「映像を用いないゲーム」の開発

環境情報学部 4 年　橋山牧人

概要

株式会社ユードーの南雲氏は，映像をまったく用いずに音だけで遊ぶことができる新しいゲームを作りたいという企画を持っていた．その企画に興味を持った私は，研究プロジェクトで南雲氏と一緒に「映像を用いないゲーム」を開発することにした．本文書は，南雲氏や各企業の技術協力を得て開発を行った「映像を用いないゲーム」の開発についてまとめたものである．

「映像を用いないゲーム」によって，従来のゲームでは非常に難しかった健常者と視覚にハンディキャップを有する人が一緒に遊ぶことが可能になり，社会的に大きな貢献ができる．また，音の持つ役割や影響を調べることで，映像だけで表現することが難しかった場の雰囲気や人の感情を，より豊かに表現することができるようになる．さらに，映像を用いないことで目を使わない新しいゲームを提供できるという意義がある．

目次

第 1 章.　「映像を用いないゲーム」概要 3
1.1.　開発の背景 3
1.1.1.　私がゲームを好む理由 3
1.1.2.　携帯ゲームアプリの開発経験 3
1.1.3.　南雲氏との出会い 4
1.2.　「映像を用いないゲーム」とは 4
1.2.1.　ゲームの定義 4
1.2.2.　ゲームの対象 5
1.2.3.　ゲームを行う環境 5
1.3.　「映像を用いないゲーム」を開発する意義 6
1.3.1.　社会的な貢献 6
1.3.2.　映像を用いたゲームへの応用 6
1.3.3.　新しいゲームへの応用 6
第 2 章.　「映像を用いないゲーム」の開発 7
2.1.　「さうんど おんりぃ」プロジェクト 7
2.2.　「さうんど おんりぃ 2」プロジェクト 7
第 3 章.　今後の展望 8

第1章. 「映像を用いないゲーム」概要

1.1. 開発の背景

1.1.1. 私がゲームを好む理由

私は無類のゲーム好きである．休みには自宅でロールプレイングゲーム（RPG）をプレイする．また，大学の帰りにゲームセンターで音楽体感シミュレーションゲーム（音ゲー）をプレイすることも多い．

私が RPG を好むのは，現実とは大きく異なる世界で人々と会話し，モンスターを撃退しながら旅をする，というシチュエーションに憧れるからである．RPG とは名前の通りプレーヤーがゲームの中で特定の役を演じながら物語を進めていくゲームだ．自分とはまったく違う立場に置かれている主人公に感情移入するという点で，RPG は映画に似ている．しかし，私は映画を見ることよりも RPG をプレイすることの方が好きである．

また，私が音ゲーを好むのは，音楽に合わせてリズムを取るということが好きだからである．音ゲーとは流れてくる音楽のリズムに合わせてタイミングよくボタンを押したり，踊ったり，楽器を演奏したりするゲームだ．元々アップテンポの音楽が好きだった私は，音ゲーで用いられている曲が気にいって音ゲーを始めた．しかし，次第に音楽を聞くことより，リズムに合わせてゲームを行うことを好きになっていった．

私が映画や音楽を見聞きするよりもゲームをプレイすることが好きな理由は，そこにプレーヤーの意思が介在するからである．映画や音楽はそれを見聞きする人の意思とは関係なく進んでいくが，ゲームはプレーヤーが意図しない限り進まない．つまり，ゲームはある程度自分の意思で進行を操作することができるのである．

1.1.2. 携帯ゲームアプリの開発経験

ゲームをプレイすることが中心だった私は，自分でもゲーム開発に携わってみたいと考えた．そこで，2004 年より 1 年ほど携帯のゲームアプリを開発している会社でアルバイトをした．アルバイトの経験を通じて分かったことは，日本のゲーム開発の現場ではアイデアを重視するあまり，コーディングの前にしっかりとした設計を行わないということだ．特に，ソースコードの再利用性や可読性を考えていないことが多い．後からゲームを拡張するときに，ソースコードを理解するだけで必要以上に時間と労力がかかってしまう．

1.1.3. 南雲氏との出会い

2005 年の冬に，大岩先生の紹介により，音ゲーの創始者の一人である株式会社ユードーの南雲氏と出会った．南雲氏は，映像をまったく用いずに音だけで遊ぶことができる新しいゲームを作りたいという企画を持っていた．私は，音だけでゲームを作るとはどういうことであるかを考えているうちに，ゲームにおいて音がどのような役割を持ち，音はプレーヤーにどのような影響を与えるのかということに興味を抱くようになった．これをきっかけとして，南雲氏と協力して「映像を用いないゲーム」のプロトタイプを開発する「さうんど おんりぃ」プロジェクト（詳細は第 2 章で述べる）が発足したのである．

1.2. 「映像を用いないゲーム」とは

1.2.1. ゲームの定義

現在，発売されている多くのゲームでは，ゲームから出力される情報は映像と音が中心である．「映像を用いないゲーム」では，ゲームからの出力を音のみに限定する．また，プレーヤーからの入力を，プレーヤーが行ったコントローラーの操作とする（図 1-1 参照）．

ゲームによっては，コントローラーに振動を伝え，プレーヤーの触覚を利用して情報を伝える方法もある．しかし，このような情報は映像や音に比べて伝えられる情報が限定されるため，ここでは考慮しないものとする．

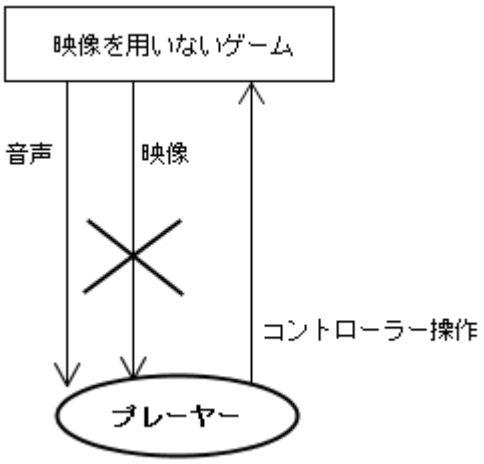


図 1-1：「映像を用いないゲーム」のモデル

1.2.2. ゲームの対象

「映像を用いないゲーム」は健常者と視覚にハンディキャップを有する人を対象とする．

1.2.3. ゲームを行う環境

「映像を用いないゲーム」では，ゲームからの出力が音のみに限定される．音を表現するために音響環境として 5.1ch サラウンドを使用する．5.1ch サラウンドでは，前方に 3 つのスピーカー（3ch）とサブウーハー（0.1ch），後方に 2 つのスピーカー（2ch）を設置する．これによって，音に前後の広がりを与えることで，ステレオでは表現し切れなかった 360 度の立体感を表現できるようになる（図 1-2 参照）．また，「映像を用いないゲーム」では，入力機器としてコントローラーを想定している（図 1-3 参照）．

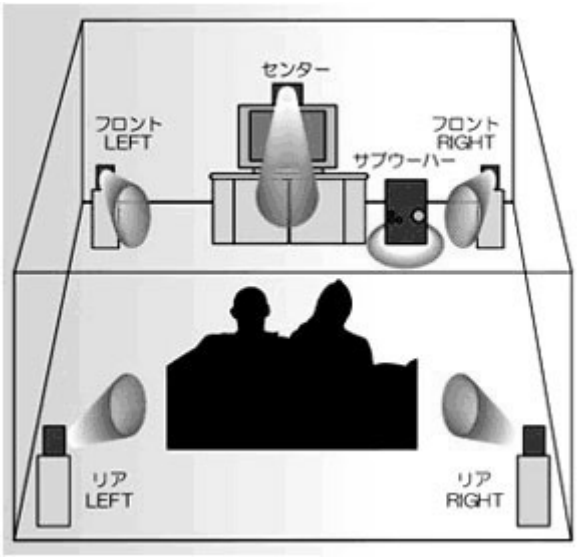


図 1-2：5.1ch サラウンドイメージ[1]

図 1-3：Xbox360 のコントローラー[2]

1.3. 「映像を用いないゲーム」を開発する意義

「映像を用いないゲーム」を開発することには，大きく分けて以下の 3 つの意義がある．

1.3.1. 社会的な貢献

「映像を用いないゲーム」では，ゲームからプレーヤーに出力される情報が音のみであるため，健常者と視覚にハンディキャップを有する人が一緒に遊ぶことができる．従来の映像を用いたゲームでは，ゲームからプレーヤーに出力される情報のほとんどが映像である．そのため，健常者と視覚にハンディキャップを有する人が一緒に遊ぶことは非常に難しい．従って，健常者と視覚にハンディキャップを有する人が一緒に遊ぶことができる「映像を用いないゲーム」を提供できることは，社会的に大きな意義を持つのである．

1.3.2. 映像を用いたゲームへの応用

映像を用いないことで，ゲームにおける音の役割や，音がプレーヤーにどのような影響を与えているかが，映像を用いたゲームに比べて一段と分かりやすくなる．そのため，ゲームを開発するときに，音の役割や影響を詳細に調べることができる．その成果を元にして，従来のゲームにおいて映像だけでは表現することが難しかった「場の雰囲気」や「人の感情」などに対して，音を組み合わせることでより豊かに表現することができるようになる．

1.3.3. 新しいゲームへの応用

映像を用いないため，長時間ゲームをプレイすると目が疲れて，視力が低下するという常識を覆すことになる．その結果，目を使わないという今までに無かった全く新しいゲームを提供することができる．例えば，目が疲れたときにプレイすることで目を休めてリラックスできるといったゲームが考えられる．

また，映像を用いないので，音さえ聞こえれば周りの環境に左右されない．例えば，夜行バスの中で消灯時間が過ぎた後に眠れないとき，イヤホンさえあれば誰にも迷惑をかけることなくゲームを楽しむという状況が想定できる．

第2章. 「映像を用いないゲーム」の開発

2.1. 「さうんど おんりぃ」プロジェクト

「映像を用いない」ゲームのプロトタイプとして，音を頼りにテロリストの仕掛けた時限爆弾を解除するというゲームを開発した．このプロトタイプの開発に関する報告は，すべて**「さうんど おんりぃ」プロジェクト最終報告書**にまとめられているので，そちらを参考にして頂きたい．

2.2. 「さうんど おんりぃ２」プロジェクト

「さうんど おんりぃ」プロジェクトの開発経験を踏まえて，「映像を用いない」ゲームの第 1 弾として，「ForestWalking」を開発した．このゲームは，森の中を散歩しながら，聞こえてくる昆虫の鳴き声を聞き分けて，虫捕りを行うというゲームである．「ForestWalking」の開発に関する報告は，すべて**「さうんど おんりぃ２」プロジェクト最終報告書**にまとめられているので，そちらを参考にして頂きたい．

第3章. 今後の展望

今回，卒業制作として今まで開発してきた「映像を用いないゲーム」の開発報告を行った．今後の展望として，まずは，今回の評価をもとに「ForestWalking」を一般公開し，色々な立場の方からご意見を頂きながら改良を重ね，商品として売れるものを作りたい．そして，ゆくゆくは「映像を用いない携帯アプリ」への応用や，ゲーム以外のリラックスできる商品として「映像を用いない作品」などの研究・開発も視野に入れている．

そのために，私自身は慶応義塾大学大学院政策・メディア研究科に進学し，株式会社ユードーの南雲氏や横浜市立盲学校の松田氏，並びに同校の生徒の皆様にも引き続きご協力頂き，「映像を用いないゲーム」の研究・開発を引き続き行う予定である．

2006 春学期　大岩研究プロジェクト 2

さうんど おんりぃ班　最終報告書

PM　三菱スペース・ソフトウエア　中川誠

環境情報学部 4 年　橋山牧人

環境情報学部 3 年　佐藤俊之

環境情報学部 3 年　堀正人

総合政策学部 2 年　渡辺士郎

1. プロジェクトの概要

さうんど おんりぃプロジェクトは、(1) 学習プロジェクト、(2) ゲームプロジェクトの 2 つのサブプロジェクトから構成される。それぞれのプロジェクトごとに目標、成果物スコープが定義されている。

・参考資料（添付資料 5.1：プロジェクト定義書）

1.1. 背景

株式会社ユードー社の南雲氏は、視覚にハンディキャップを有する人と健常者との両方が楽しむ時間を共有することができるゲームを開発したいと考えている。このプロジェクトでは、サラウンド音響を用いた映像の無いゲームの開発をして欲しいという南雲氏の依頼をもとに、「映像のないゲーム」を開発していく。

1.2. 目的

南雲氏から依頼された「映像のないゲーム」を開発する。

1.3. 目標

プロジェクトの目標について説明する。

参考資料（添付資料 5.2：スコープ記述書）

1.3.1. 学習プロジェクト目標

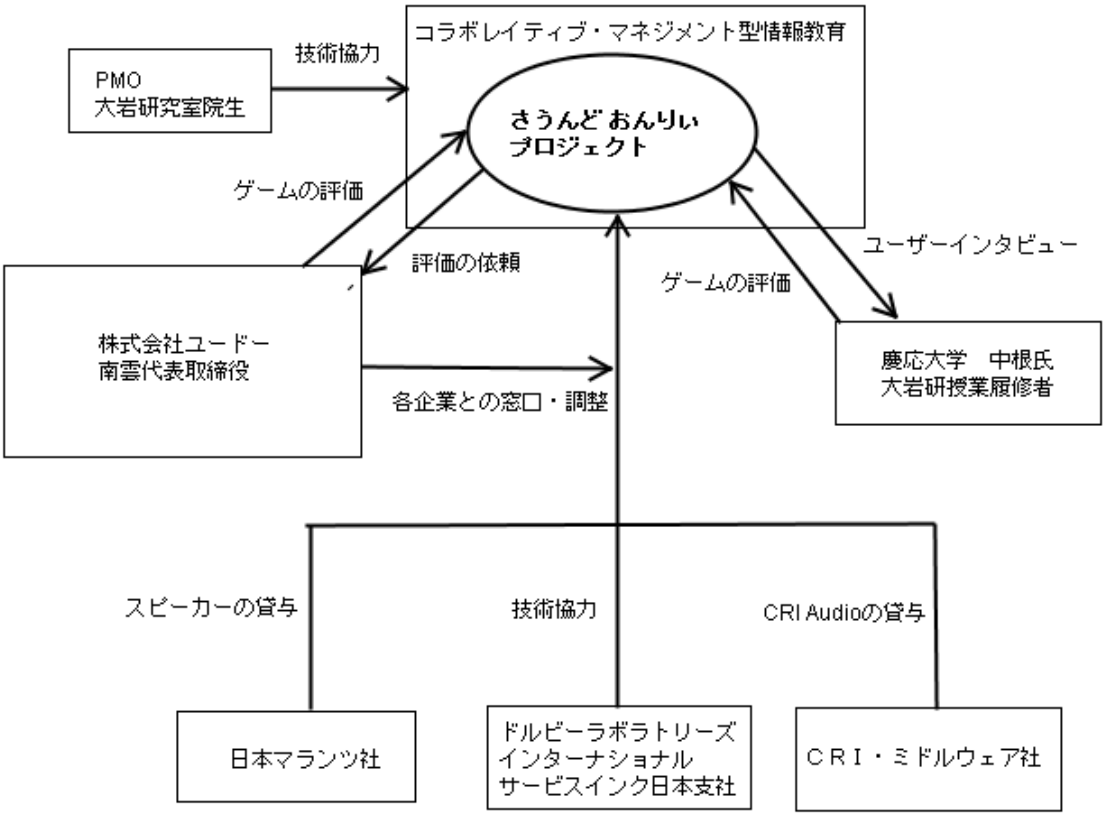
プロジェクト活動について学習するため、本プロジェクトでは、適切な計画を立案し、適切なプロセスでプロジェクト活動を実施することを目標とする。また、学生、PM がプロジェクト活動を通して、自己の目標を達成し、“ 成長 ” することを目標とする。

1.3.2. 学習プロジェクト目標

視覚にハンディキャップを有する人、健常者がともに楽しい時間を共有することができる、「映像のないゲーム」を開発するための課題を解決すること目標とする。

1.4. 体制

さうんど おんりぃは大岩研究室の研究プロジェクトにおけるプロジェクトの 1 つである。プロジェクト・マネジメント・オフィス（PMO）や南雲氏を通じた各社から技術協力や機材の貸し出しをしてもらう。また、慶応大学内の視覚にハンディキャップを有する方へユーザーインタビューを行い、ゲームの評価をして頂く。



さうんど おんりぃプロジェクトの体制

1.5. 開発環境・設備

実装環境として、Visual Studio2003.NET を利用して、C++で行う。また、音をサラウンドで再生するときに、株式会社 CRI・ミドルウェア社の「CRI Audio」を利用する。さらに、Subversion によってプロジェクト内の UML やソースコード、ミーティングログを始めとするすべての資料をプロジェクトメンバー内で共有する。

音響設備として、開発用に 5.1ch 対応のヘッドホンを使用する。

2. 企画詳細

2.1. 企画決定までの変遷

2.1.1. 企画構想

2.1.1.1. 企画構想

今回、南雲代表取締役に依頼されたのは「映像のないゲーム」の開発と、非常に漠然としたイメージのものであった。よって、企画の決定を行うのも今プロジェクトの一環であった。

企画の決定は以下の手順で行われた。

2.1.1.2. 第 1 回企画考案会

まず、各プロジェクトメンバーが企画を持ち寄り、考案会という形で、プロジェクトミーティング内で各自 5 分程度の発表を行った。発表後、各案の良かった点を挙げていき、それぞれの良い点を抽出してそれを元に企画を決定させようとした。

この際、「何が出来そうか」という考えは排除して、単純にどういうものが作りたいか、どういうゲームなら遊びたいかという考えに基づいて案を考えることにした。初めからアイデアが制限されていては良い案を見落としてしまう可能性があるからだ。

このミーティングでは、宝探しゲームや、動き回る対象を追跡するゲーム、銃型コントローラーを用いたガンシューティングゲームと言った様々な案が出た。この内ガンシューティングゲームはハード面を実現するのが難しそうという理由で断念することになった。

2.1.1.3. 第 2 回企画考案会

続いて、第 1 回企画考案会に出た「良かった点」を元に、再び各自で案を練り、第 2 回考案会にて発表を行うこととなった。考案会では、既存の宝探しゲームや追跡ゲーム、複数人での鬼ごっこゲーム等が立案された。

2.1.1.4. 企画決定

第 2 回企画考案会で出た案を元に議論した結果、対戦型ゲームの方が健常者と視覚にハンディキャップを有する人が同時に楽しむことができるという考えもあり、鬼ごっこゲームに他の案の良いところを搭載させたものが企画として決定した。

また、対戦は不特定多数の人間が同時に遊べる環境ということでネット対戦で行おうということになった。

企画の詳細は **2.2 スコープ定義前の全体企画**にて記す。

2.1.2. スコープ定義

2.1.2.1. 目的

企画の構想と平行して、スコープの構想も行われた。今回の製作期間 3 ヶ月と短いため、実装する範囲を決定する必要があるからである。

2.1.2.2. 経緯

第 2 回企画考案会において、各自、自分の案が今学期中にどれだけの範囲が実装できそうかということを発表した。先述の通り、話し合いの結果企画が決定したが、その後もスコープの定義についての議論を行った。

2.1.2.3. スコープ決定

操作可能なのは鬼ごっこの「追う側」のみという点はすぐに決定したのだが、「逃げる側」を固定とするか、AI を搭載して逃げるようにするかという議論が行われ、最終的にとりあえずは根幹部分を作成しようということで逃げる側は固定することにした。また、ゲームを彩るトラップやボーダーラインといったオブジェクトは段階を踏んで実装しようということにし、スコープに優先順位を付け、作成の順番を付けた。

決定したスコープの詳細は **2.3 スコープ定義後の企画**にて記す。

2.1.3. 学んだ点・苦労した点

2.1.3.1. 企画の決定

今プロジェクトのモットーとして、全員が納得して作業に取り掛かりたいというものがある。自分の本意ではないものを作成しようとしてもモチベーションは上がらないし、ストレスも溜まってしまうからだ。なので、企画が決定するまで全員が納得するまで討論を行った。

当然、今プロジェクトに参加したメンバーは何らかの「やりたいこと」を持っていた訳なので、その「やりたいこと」ができなくなってしまうこともある。やはり、全員が納得する企画が決定するまでには時間がかかった。今回のことで、誰もが気持ちよく作業を行うことの重要さ、及びその難しさを学んだ。

2.1.3.2. スコープの決定

スコープを決定するのにあたり難しかったことは、殆どのメンバーが VisualC++ で作業をしたことが無く、またサラウンド音響に触れたことが無かった、ということが挙げられる。これらの原因で、自分たちがどこまで実装できるか分からないという不確定要素が発生した。スコープを決定するに当たって、まず何を達成すれば良いとするかということよりも、自分たちがどこまで達成することができるか、ということが焦点になって話し合いが行われた。

この決定したスコープ後の企画では、自分たちがまず達成すべきであろう範囲と、最終的に達成させたい範囲の二つに範囲を分けることにした。このようにスコープに段階をつけることにより、まず何からすればいいのかという明確なゴールラインを作成することができた。

2.2. スコープ定義前の全体企画

2.2.1. ゲームの概要

複数対複数で対戦可能な「鬼ごっこゲーム」

2.2.2. プレイヤーの役割

このゲームには「追う側」と「逃げる側」の 2 つの役割があり、プレイヤーはどちらかを選択してプレイすることになる。

追う側のプレイヤーの役割は時間内に逃げる側のプレイヤーを捕まえることで、逃げる側のプレイヤーの役割は時間制限まで追う側のプレイヤーから逃げ切ることや、追う側のプレイヤーをトラップにはめて撃退するということである。

複数対複数ということで、追う側のプレイヤー及び逃げる側のプレイヤーが複数存在することもある。

2.2.3. ゲームの特徴

サラウンド音響を利用し、音による空間把握ができる。つまり、相手が発信する音で、相手の現在地がわかる。また、フィールドに点在するトラップも音を発し、周囲に何があるのかということを常に見張ることが可能である。

また、多数の人間が 1 つのフィールドで同時に遊ぶことになるので、ネット対戦という形で対戦を行う。

2.2.4. ゲームを盛り上げる要素

一部に記述したが、トラップという要素がある。これは、逃げる側がトラップを置き、追う側がその設置したトラップに接触したときにある効果をもたらすというものである。この効果の案として、場所を移動させたり、動きを止めたり、即ゲームオーバーにするという案がある。

また、フィールドは平面であり、一定のマスに仕切られている。これは、音のみでは地面の高低や壁の表現をすることが難しいためである。また、果てしない平面では面白みが無いため、ボーダーラインという要素を追加する。一定の範囲外に出たプレイヤーはその場でゲームオーバーになると言うものである。また、この他に時間が経過すると進入禁止になるエリア（ボーダーラインを狭める）といった要素が考案された。

2.3. スコープ定義後の企画

上記の企画を元にスコープ定義を行った。今回、上記の企画の範囲内で実装する範囲は以下である。

2.3.1. ゲームの概要

逃げる側の場所を固定し、プレイヤーは追う側のみとする。

プレイヤーは音だけを頼りに，フィールド上のトラップを避けつつ制限時間以内に目標物に到達するという一人用のゲームとする。（スイカ割りのようなもの）

2.3.2. プレイヤーの役割

プレイヤーは目的地点へ辿り着くことを目標とする。

2.3.3. ゲームを盛り上げる要素

トラップ及びボーダーラインは実装する。しかし、トラップは逃げる側が設置するのではなく、初めから置いてあるようにした。

2.3.4. ストーリー

上記の企画を踏まえ、以下のようなストーリーも作成した。

「テロリストが地下駐車場に爆弾を設置したので、それを解除しに行かなきゃいけない。またテロリストがそこの電気を切ったので、暗闇を模索しながら行く。」

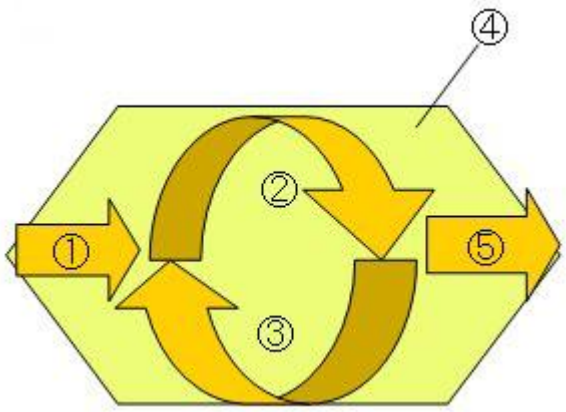
以上のプロセスを経て、企画・立案フェーズは終了した。

3. プロジェクトの進行内容

3.1. プロジェクトの流れ

さうんど おんりぃプロジェクトは、プロジェクトのマネジメント手法であり、世界中で広く認知されている知識体系である PMBOK に従って進めていく。メンバーの半分はプロジェクト未経験者で、残り半分は経験者である。PMBOK に従ってプロジェクトを進めることで、未経験者はプロジェクトがどのようなものであるのかということを体系的に学ぶことができる。また経験者は、過去に経験したプロジェクトとの違いを比較することで、プロジェクトの全体像が見え、プロジェクトの管理方法を学ぶことができる。以上が本プロジェクトを PMBOK に従って進めていく理由である。

PMBOK では以下の 5 つのフェーズがある。



図：3.1-1 PMBOK の 5 つのフェーズ

① 立ち上げフェーズ
② 計画フェーズ
③ 実行フェーズ
④ 監視・コントロールフェーズ
⑤ 終結フェーズ

・ 参考資料
  - 添付資料 4.1 目標管理シート
  - 添付資料 5.1：プロジェクト定義書
  - 添付資料 5.7 WBS・スケジュール

3.2. 反復作業

3.2.1. 全体設計

3.2.1.1. 方針

はじめに、最初の成果物となるスコープ第 1 段階の設計に取り掛かるのではなく、全体の設計を行うことにした。理由として、将来的な拡張性が挙げられる。一番小さいものをベースにすると、それを拡大する際に仕様変更等のリスクが生じることがある。それを防止するため、全体の設計から、それをベースに実際に実装する箇所だけを抜き出すという形式で実装を行うことにした。

3.2.1.2. 作業概要

まず要求分析を行い、その結果を基に仕様を決定した。その仕様を基に更に設計を行い、反復第 1 回目の範囲の設計へと移行した。

3.2.1.3. 作業詳細

3.2.1.1.1. 要求分析

3.2.1.1.1.1. 目的

どのような機能があればシステムを満足することができるのかということを分析した。

3.2.1.1.1.2. 経緯

まずメンバー全員が各々考えたユースケース図を作成し、それを基に討論を行った。

この議論において、各人がユースケース図のみを作成し、それに対応するユースケース記述を作成していない、という点が明らかになった。確かに、ユースケース図だけで要求の理解に導くのは不可能であるという解釈を全員が持ち、ユースケース図を作成するときはユースケース記述も忘れないようにしよう、という意識が全体に浸透した。

またこの時、「ゲーム」というシステムの特異性（目的が曖昧）から、通常のユースケース図では表現が難しいという話になった。そこで、我々は 2 つの手段を用いて要求の分析を行うこととした。1 つは、ゲームに向いていると言われる組み込み UML について調査し、それにおけるユースケースの記述方法に基づいてユースケース図及びユースケース文書を記述する、という手段である。もう一方は、プレイヤーにゲームの説明や、ゲーム内で出来ることを著したマニュアルを作成する、という手段である。それぞれをメンバーが作成し、討論を基に双方の作成を行った。

3.2.1.1.1.3. 成果物

3.2.1.1.1.3.1. ユースケース

まず、作成したユースケース図について説明したい。組み込み UML においては、システム内に存在するオブジェクトもアクターの一部となり、それぞれのユースケースも記述する。通常と異なっている点として、ユーザが何を

したいかを分析するのではなく、どのような機能があればシステム自体が満足できるのかということを分析するという点が挙げられる。

なお今回作成したユースケース図、及びユースケース記述は 5.4.1.1 項の資料 3.2.1.3.1.3.1-1 と 3.2.1.3.1.3.1-2 に添付した。

3.2.1.1.1.3.2. マニュアル

続いて、マニュアルの記述を行った。これもユースケース図と同様に複数のメンバーが作成し、検討を基に決定した。プレイヤーにとって理解がし易い、逆に言えばプレイヤーが理解するのに必要な範囲をマニュアルから分析するという狙いがある。なお、マニュアルは 5.5 項に添付した。

3.2.1.1.1.3.3. 苦労点や発見点

ここで議論しているうちに、ユースケースとマニュアルを比較しての発見点が見つけられた。まず、ユースケースはどちらかというと製作者（システム寄り）の視点で書かれたものであるのに対し、マニュアルはプレイする側の立場に立って書かれたものである。プレイヤーが何かをするには、何をできるようにすべきか、といったゲームに必須となる動作の要求を分析するにはマニュアルは向いており、逆にシステム内でどういった作業が行われればプレイヤーがやって欲しいことを解決できるか、といった点を分析するにはユースケースが向いている、という結論に達した。

この結果、実装にはユースケースが向いているかもしれないが、細かな仕様の確認をするためにマニュアルを用いるのも有効だと言う結論に達した。

3.2.1.1.2. 仕様

3.2.1.1.2.1. 目的

どのような機能があればユースケースを要求を満足させることができるかをリストアップする。

3.2.1.1.2.2. 経緯

先述のユースケース及びマニュアルに記した項目を全て満足させるために必要な機能を全て挙げ、1 つに纏めた「機能仕様一覧」を作成した。

3.2.1.1.2.3. 成果物

作成した機能仕様一覧は 5.4.1.1 項にある表 3.2.2.3.4.3. -1 である。

3.2.1.1.2.4. 苦労点や発見点

2 つの要求分析に用いたツールの整合性を取りつつ、必要だと思われる全ての機能及びその条件を記述するのに苦労し、当初は明らかに仕様の数が少なかった。最終的には、作成の結果、全員が何をつくるのかという点を共有することが出来た。

3.2.1.1.3. 設計

3.2.1.1.3.1. 目的

決定した仕様を基に、全体の設計を行う。

3.2.1.1.3.2. 経緯

設計も、これまでのプロセスと同様に全員の作成したモデルを基に討論した。その際、論点となったのは各種トラップを「トラップ」という抽象クラスのサブクラスとするかということ、及びプレイヤーと時限爆弾を「キャラクター」という抽象クラスのサブクラスにするか、と言った点である。これらのことは、キャラクタークラスを実装するメリットが現時点では見当たらないといった点から、とりあえずのクラス図を作成するという形になった。理由として、実際に実装をしてみないと分からないという点が考えられる。反復第 2 回の冒頭に再び全体設計を行うのだが、これらの点はその時に改めて議論され、最終的な結論へと達する。

成果物として、クラス図・オブジェクト図・シーケンス図を作成した。手順としてはまずオブジェクト図を作成し、ゲーム内にどのようなオブジェクトが登場するかを考えた。続いてクラス図を作成し、要求を実現するためにはどのようなクラスが必要となるかをまとめた。

最後にシーケンス図に全体的な処理の流れや各プロセスにおける処理の段階を記し、要求を実現するための手順を纏めた。当初はシーケンス図は 1 つだけであったが、そのままでは処理の流れが膨大であり、また条件分岐も含めて全て一枚にまとめてしまったので、非常に読みにくく、処理の流れもつかみづらいと言うレビューが入った。そこで、複数の図に分けて書いたという経緯がある。

3.2.1.1.3.3. 成果物

作成したオブジェクト図、クラス図、シーケンス図は 5.4.1.1.項の図 3.2.1.3.3.3-1 から 3.2.1.3.3.3-6 を参照のこと。

3.2.1.1.3.4. 苦労点や発見点

上記のように、UML に詳しいメンバーとそうでないメンバーがおり、各々が図を作成し、それについて討論するだけでも、考え方などを参考にすることができ、勉強になった。

3.2.2. 反復 1 回目

3.2.2.1. 方針

反復第一回目の方針は、まずゲームの根幹部分であるスコープ第 1 段階を作成しようというものである。特に、このゲームの特徴的な部分でもある、音響の出力によってキャラクターと対象物の相対的な距離と方向の把握を可能とする機能の実装に注力することとなった。

3.2.2.2. 作業概要

まず要求分析を行い、その結果を基に仕様を決定した。その仕様を基に更に設計を行い、反復第 1 回目の範囲の設計へと移行した。設計が終了してから実装に取り掛かり、試験を行ってスコープ第 1 段階は終了した。

3.2.2.3. 作業詳細

3.2.2.3.1. 要件定義

3.2.2.3.1.1. 目的

反復第 1 回目において、どこまでの機能を実装するかを決定する。

3.2.2.3.1.2. 経緯

反復第 1 回目がどのような仕様を持つかは全員の話し合いを基に決定された。

3.2.2.3.1.3. 成果物

仕様を絞りこんだ結果、全体仕様と以下の点が異なった。

・ゲーム内オブジェクト：プレイヤー、時限爆弾（目的地）

・機能は以下の点を達成することを目的とする

(1) 音響出力：時限爆弾（目的地）が音を発し、位置がわかる

(2) キャラクターの移動：プレイヤーの入力により、キャラクターが移動する

(3) 描画：デモ用、デバッグ用に、キャラクターと目的地の存在地点が画される

要件を達成する目標として、ユーザーがプレイヤーを操り、音のみでプレイヤーを時限爆弾のある場所へと到達できることが挙げられる。具体的には、音の出力によって時限爆弾のある場所が把握できることである。

当初は、目標地点に到達したらゲームクリアにするといった案もあったが、音響出力の手段であるミドルウェアの使用方法や使用具合が不透明であったため、まずは根幹部分を作成しよう、ということになった。

3.2.2.3.1.4. 苦労点・発見点

どこまでを最初に作ると楽か、どこまでなら決められた期限までに実

装を完了できるか、といった点を基に話し合ったのだが、なかなか纏まらなかった。焦点となったのは当たり判定、及びそれに付随するクリア判定やゲームオーバーの判定であるが、実装しなければゲームとして成り立たないのではないかという意見も出た。しかし、とりあえずは動くものを作成することを目標にしよう、ということで上記のような仕様が出来上がった。

　自分たちがどこまでやれるのかわからない場合、要件 1 つを決めるのにも苦戦するのだという勉強になった。

3.2.2.3.2. 要求分析

3.2.2.3.2.1. 目的

　要件定義の結果を受け、要件を満足する機能の洗い出しを実施した。

3.2.2.3.2.2. 経緯

　このような要件を絞ったバージョンでは、システムが行える事は当然のことながら全体設計で分析した物とは異なる。その変更点をユースケース図に反映させた。

3.2.2.3.2.3. 成果物

　ユースケース図を作成した（5.4.1.2 項の図 3.2.2.3.2.3-1）。なお、ユースケース記述は全体設計のものと同一であるので、そちらを参照していただきたい。

3.2.2.3.2.4. 苦労点・発見点

　特になかった。

3.2.2.3.3. 設計

3.2.2.3.3.1. 目的

　スコープ第 1 段階の実装のため、詳細設計を行った。

3.2.2.3.3.2. 経緯

　上記の要求分析の決定を受け、設計を行った。設計の成果物として、全体分析時に作成したクラス図・オブジェクト図・シーケンス図の仕様を絞ったバージョンを作成した。

　これらも、全体設計におけるモデルに疑問点が生じる度に変更を加えられた。

3.2.2.3.3.3. 成果物

　オブジェクト図、クラス図、シーケンス図を作成した。5.4.1.2 項の図 3.2.2.3.3.3-1 から 3.2.2.3.3.3-5 を参照のこと。

3.2.2.3.3.4. 苦労点・発見点

　実装の途中に疑問点が加わるたびに修正を行った。その度に全ての図

を変更する必要があったということ、またちゃんと図の間の整合性が取れてなければならないといった点で苦労をした。

3.2.2.3.4. システム仕様決定

3.2.2.3.4.1. 目的

スコープ第 1 段階に存在する機能の仕様を挙げ、満足させるべき項目のリストアップを行う。

3.2.2.3.4.2. 経緯

上記の設計の結果、明確化されたシステム仕様の一覧を作成した。全体の要求仕様と比較すると、今期は実装しない部分や、仕様を変更して実装する部分が出てくる。これらの差分を色分けすることにより、今回の開発範囲を把握できるようにした。

3.2.2.3.4.3. 成果物

仕様一覧は 5.4.1.2 項の表 3.2.2.3.4.3 -1 として添付した。

3.2.2.3.4.4. 苦労点・発見点

特に無かった。

3.2.2.3.5. 実装

3.2.2.3.5.1. 目的

実装を行い、スコープ第 1 段階の成果物を作成する。

3.2.2.3.5.2. 経緯

上記の成果物を以て、反復第 1 回目の設計を終了とした。ただ、実装の途中にも細かな仕様の変更や設計の変更は行われた。数多くのレビューを経た、最終的なモデルが上記のモデルであることを明記しておく。

設計の変更が行われた理由として、実際に実装して見なければ分からなかった点、特にミドルウェア付近の扱いがある。実装するにあたって疑問点が出る毎に設計のレビューを行い、その後に実装を再開するという形式によって実装は行われた。

3.2.2.3.5.3. 成果物

映像ないゲーム（第 1 版）

3.2.2.3.5.4. 苦労点・発見点

実装は、1 人のエキスパートを中心に進められることになった。しかし、全てを 1 人に任せると何の学習にもならないので、全員に作業を分担させる必要があったのだが、どこまでが各自で作業することが可能なレベルであるかが難しく、あまり作業の分担ができなかった。

3.2.2.3.6. 試験

3.2.2.3.6.1. 目的

実装による成果物が、要件を満たすものであるか確認する。

3.2.2.3.6.2. 経緯

実装の終了後、試験を行った。試験は、まず試験手順書を作成し、各項目に対応する箇所を埋めていく、という形式で行われた。

3.2.2.3.6.3. 成果物

試験手順書・成績書試験手順書は 5.4.1.2 項に表 3.2.2.3.6.3-1 に添付した。

3.2.2.3.6.4. 苦労点・発見点

今回はシンプルな機能しか持ち合わせていなかったため大きな問題にはならなかったが、試験項目があまり詳細まで行き渡ってなかったため、変更を加えた後もまだ細分化して評価できる項目があった。

3.2.3. 視覚にハンディキャップを有する人に対するインタビュー

3.2.3.1. 目的

視覚にハンディキャップを有する方に現状の成果物をプレイしてもらってその内容を評価してもらう。

3.2.3.2. 経緯

今回のインタビューは、以前より協力をお願いしていた慶應義塾大学大学院、政策・メディア研究科の特別研究助手である中根雅文氏にお願いした。実際のインタビューは 2006 年 6 月 28 日の 13 時 30 分頃から慶應大学 SFC のイオタの 308 で行われた。

インタビューの内容は大きくわけて以下の 3 つだった。

(1) 現状のゲームをプレイしてもらい、評価してもらう
(2) ゲームをより良いものにするにはどうすれば良いかを乞う
(3) 視覚にハンディキャップを持つ人とコンピューターゲームの関係が一般的にどういうモノかを聞く

3.2.3.3. 成果物

以上のようなインタビューを経て、我々が反復第 2 回目に反映することになった仕様は以下の 2 点である。

(1) ステージ単位でのハイスコアを記録・提示する
(2) ステージ開始前にチュートリアルを設置する

3.2.3.4. 苦労点や発見点

結果として、あまり新しい情報が得られることは無かった。主に操作方法やゲーム性に関する質問を投げたものの、ゲームそのものがまだ単純だったからか、ゲーム一般にも言えるような回答しか得ることができなかった。

例えば「ゲームをより面白くするにはどうすれば良いか」という質問に対し、中根氏は「スコアなど、人と競えるものがあれば良い」と答えたが、それは別に今回のゲームに限った話ではなく、ゲームでは一般的に使われている手法である。また、音の種類や定位を認識するためにチュートリアルがあった方が良い、という意見も出たが、それもやはり事前にチーム内で考慮されていたことで、特別真新しい話ではなかった。

ただ、1 つだけ面白い話を聞くことができた。その内容は、視覚にハンディキャップを有する人でも一般的なコンピューターゲーム（の一部）はプレイすることがある、ということだ。詳しく話を聞くと、やはり視覚情報が得られないため、（敵の動きなどで）ランダムな要素が絡んでくるゲームは難しいが、パターン化できるゲームだったら充分プレイ可能である、とのことだった。例としては横スクロール型のアクションゲームなど、敵の出現位置や動きがパターン化されているゲームなどが挙げられた。念頭に置いておくこととしては貴重な意見だった。

全体としては、「これだけはこの人に問いたい」というような事項がなかったため、効果的とは言えないインタビューとなってしまった。実際に追加された仕様が一般的なゲームで使われている手法であることを考えると、やはりインタビューを行うには早すぎたのではないか、あるいはそもそもインタビューは必要でなかったのではないか、と感じた。

3.2.4. 反復 2 回目

3.2.4.1. 方針

第 2 段階の反復作業は、第 1 段階の成果物があるので、それを基に拡張していく事にした。そして第 2 段階の反復作業の設計・実装は第 1 段階の成果物を作っていく際に気付いたことや第 1 段階の成果物を視覚にハンディキャップを有する人に評価をしていただいた結果を盛り込んだ内容とした。これは当たり前のことだが、第 1 段階と第 2 段階とはっきり作業をわけているのは実装以前にわからなかった問題や第 1 段階の実装をしてみて生じた問題を第 2 段階で解決するためである。これは、今回の映像のないゲームという前例のないものを作るにあたり、PM が最初から問題が起こるだろうということを先のことを想定した方法であり、ベストではないかもしれないがベターな方法であったといえる。

3.2.4.2. 作業概要

まず第 1 段階の成果物が出来たことによる追加された要求分析を行い、その結果を基に仕様を決定する。その仕様を基にスコープ第 2 段階の範囲の設計へと移行する。設計が全部終了すると実装に取り掛かり、試験を行ってそれを満足させればスコープ第 2 段階の実装は終了となる。としたいところであったが、実際は日程の関係や「面白い」ゲームを作ることを優先するために設計が終わった部分から実装に移ることとなった。

3.2.4.3. 作業詳細

3.2.4.3.1. 要件定義

3.2.4.3.1.1. 目的

スコープの第 2 段階において、どこまでの機能を実装するかを決めた。

3.2.4.3.1.2. 経緯

スコープ第 2 段階がどのような仕様を持つかは､ユーザーインタビューの結果や第 1 段階の実装の結果判明した問題点について話し合いをし､その結果決定された｡具体的には目的地から遠くにいると音の聞こえ方の変化が小さいためわかりにくい事や、キー入力の改善、トラップ、ボーダーラインの追加などである。

3.2.4.3.1.3. 成果物

・ゲーム内オブジェクトとはキャラクター、時限爆弾(目的地)、ボーダーライン、トラップをさす。
・機能は以下の点を達成することを目的とする。

(1) 音響出力：時限爆弾(目的地)、ボーダーライン、トラップがそれぞれ音を発し、位置がわかる。
(2) キャラクターの移動：プレイヤーの入力により、キャラクターが移動する(第 1 段階と同じであるが、第 1 段階で発覚したキーの入力の

改善やキャラクターの移動量の調整もここに含まれると言える)。

(3) 描画：デモ用、デバッグ用に、すべてのゲーム内オブジェクトの存在地点がフィールドに描画される。

(4) 状態：第 1 段階では起動した時点でゲーム状態であったが、タイトル状態、ステージセレクト状態、チュートリアル状態、ゲーム状態、ヘルプ状態、ゲームクリア状態、ゲームオーバー状態が描画(デモ・デバッグ用に)かつ音声で知らされる。ここでは状態と書いたが、これは他のシステムでいう画面という言葉で説明されるものである。以後もわかりやすさを優先して「～画面」と記述する。

(5) ステージ読み込み：外部でつくった txt 形式ファイルを読み込み、様々なゲーム内ステージを作成できるようにする。

(6) スコア、ハイスコア：ステージ読み込み同様、ゲームを面白くするためにクリア時間を利用したスコアを作る。

要件を達成する目標として、プレイヤーがキャラクターを操り、タイトル画面からゲームクリア or ゲームオーバー画面まで音のみで遷移できること。これが最低条件で、加えて具体的ではないが「面白い」ゲームに仕上げること。この二点を目標とした。

3.2.4.3.1.4. 苦労点・発見点

最終的にどこまで出来るか、といった点を基に話し合ったのだが、第 1 段階同様なかなか纏まらなかった。第 1 段階の成果物を受けて色々修正すべき点などはあっさり決まったが、今回の焦点となったのは「面白い」ゲームとはそもそも何か、という人によって感覚が違うためそれを文章や数値で表すのが難しい問題であった。

もし「面白い」ということを要件に盛り込むことが出来るのならば、世の中の存在するゲームは殆ど面白くなるはずであるが、現実にはそんな事はない。このゲームを私たちは「音だけ」という点を「斬新」で面白いと評価したが、それはあくまで私達の主観であるので、それ以外にも何かゲームらしい要素が必要だろうということになった。

ある班員は基本的にどんなゲームでもリスクとリターンがあり、例えば競馬でも勝ちそうな馬は儲けが低いが勝てそうにない馬は儲けが高いという、低リスクには低リターンしかかえってこないし高リスクには高リターンがかえってくるものであると主張した。虎穴にいらずんば虎児を得ずである。

よって、このゲームでそのリスクとリターンを天秤にかけるようなことをどのように仕様に盛り込むかという議論になった。

Q. このゲームにおけるリターン(目標)とは何か

A. ゲームクリアもしくはゲームクリア時のハイスコア

Q. このゲームにおけるリスク(目標が達成できないこと)とは何か

A. ボーダーラインやトラップにひっかかったり時間切れで死ぬこと

これらの考えから接触すれば制限時間が増加するアイテムの追加を決めた。これは、残りの制限時間がそのままスコアとなる仕様としたので、そのままクリアするかそれとも危険を冒してアイテムをとりに行ってスコアの上昇を狙うか、というプレイヤーにリスクとリターンを迫ることが出来る方法であった。この仕様にたどりつくまでにかなりの議論を要した。

3.2.4.3.2. 要求分析

3.2.4.3.2.1. 目的

第 1 段階の成果物の完成をうけて、実装に必要な機能を洗い出し直した。

3.2.4.3.2.2. 経緯

第 2 段階の反復作業における成果物は、第 1 段階の結果などをうけて当然のことながら全体設計で分析した物とは異なってきた。その変更点をユースケース図に反映させるということであった。

3.2.4.3.2.3. 成果物

第 2 段階目のユースケース図は 5.4.2.1.項の資料 3.2.4.3.2.3.-1 に添付した。

3.2.4.3.2.4. 苦労点・発見点

ユースケース図にどこまで記述するかという点についてプロジェクト員だけでは判断できなかったので大岩研究室の松澤氏に意見を求めるなど、班員だけでは出来ないことがあったが、全員が同時に集まるのが難しいなど日程に調整などに苦労した。

3.2.4.3.3. 設計

3.2.4.3.3.1. 目的

スコープ第 2 段階の実装のため、詳細設計を行った。

3.2.4.3.3.2. 経緯

上記の要求分析の決定を受け、設計を行った。設計の成果物として、第 1 段階の反復作業時に作成したクラス図の修正を行った。また、ゲームを実装するのにわかりやすい状態遷移図の作成も行った。これらは、できあがるたびに実装にすぐ移るという形であった。よってじっくり修正をかける時間がなくモデルに疑問点が生じる度に適時変更が加えられた。

3.2.4.3.3.3. 成果物

第 2 段階のクラス図は 5.4.2.2.項の 3.2.4.3.3.2.3.-1 に、状態遷移図は 3.2.4.3.3.2.3.-2 に添付した。

3.2.4.3.3.4. 苦労点・発見点

実装の途中に疑問点が加わるたびに修正を行った。特にクラス図でのトラップオブジェクトの上に抽象クラスを作るかどうかという点で議論がわかれた。現在の仕様ではトラップが一つ(KillTrap)しかないため、わざわざ抽象化する必要性が感じられないというメイン実装者の意見があったが、最終的には今後の拡張性を考えてトラップを抽象化し、第 2 段階の範囲ではボーダーラインとトラップとアイテムはキャラクターと接触して何らかの反応を起こすという同様の機能であるので、抽象化されたトラップクラスに属する形となった。

3.2.4.3.4. システム仕様決定

3.2.4.3.4.1. 目的

スコープ第 2 段階に存在するシステム仕様を挙げ、満足させるべき項目のリストアップを行う。

3.2.4.3.4.2. 経緯

上記の設計の結果、明確化されたシステム仕様を決定した。

3.2.4.3.4.3. 成果物

仕様一覧は 5.4.2.3.項の 3.2.4.3.4.3.-1 に添付した。

3.2.4.3.4.4. 苦労点・発見点

特になし。

3.2.4.3.5. 実装

3.2.4.3.5.1. 目的

実装を行い、スコープ第 2 段階の成果物を作成する。

3.2.4.3.5.2. 経緯

上記の成果物を以て、反復第 2 回目の設計を終了とした。ただ、第 2 段階は設計ができた部分から実装に移ると決まっていたので、実装の途中にも細かな仕様の変更や設計の変更は行われた。数多くの討論を経た、最終的なモデルが上記のモデルであることを明記しておく。

設計の変更が行われた理由として、「人に使ってもらうソフトウェア」であるので、実際に実装してみてユーザーインターフェース的によろしくない点があれば、実装主導で設計の変更が行われたからである。

3.2.4.3.5.3. 成果物

映像ないゲーム「さうんどおんりぃ」（第 2 版）

3.2.4.3.5.4. 苦労点・発見点

実装は、第 1 段階同様基本的には一人のエキスパートを中心に進められ

た。

なお、今回の「映像のないゲーム」プロジェクトにおいて仕様が変わりやすいゲームにはむいていない、そもそも完全に UML を理解していないなどの理由から、実装に対して UML はあまり必要でなかったのでは、という考えがメンバーたちにあった。

しかし、UML はプログラム設計のために使用されるが、全員がそれを通じてシステムの内容が理解できるのであれば、例えば実装終了後に行われる UML でも意味があると言えるのではないだろうか、と感じた。

3.2.4.3.6. 試験

3.2.4.3.6.1. 目的

実装による成果物が、仕様を満たすものであるか確認する。

3.2.4.3.6.2. 経緯

実装の終了後、試験を行った。試験は、まずシステム仕様一覧を参考とした試験手順書を作成し、各項目に対応する箇所を埋めていく、という形式で行われた。項目として特に重要なのは第 1 段階の反復作業以降に行われた部分である。

3.2.4.3.6.3. 成果物

試験手順書に対応する試験成績書は 5.4.2.3.項の 3.2.4.3.6.3.-1 に添付した。

3.2.4.3.6.4. 苦労点・発見点

第 2 段階の反復作業の試験手順書には当初 53 個の機能がリストアップされていたが、必要ないため削った機能、後から追加された機能、仕様の変更など、終盤になってかなりの作業が立て込み、(現在のものは修正されているが)試験手順書の試験項目に若干のズレがみられた。

3.3. 成果物の評価

3.3.1. 発注者評価

3.3.1.1. 概要

7 月 13 日、発注者である株式会社ユードー（以下、ユードー社）の南雲代表取締役（以下、南雲氏）に、成果物が発注の基準を満たしている物かどうか評価をして頂きたく、ユードー社を訪問して評価会を行った。

この時点ではまだゲーム性を発展させている途中であり、チュートリアルやガイド用の音声は無かったが、ゲームの本質的な部分は出来上がっていた。また、どのようにしたらゲームを面白くすることができるだろうかということを、ゲーム製作のプロフェッショナルである南雲氏から意見を頂きたいという目的もあった。

3.3.1.2. 評価の様子

評価には南雲氏のほかに、ドルビーラボラトリーズインターナショナルサービスインク日本支社から 2 名、CRI・ミドルウエア社から 3 名、ユードー社から 1 名が参加した。

評価会ではまず参加していただいた皆様にゲームをプレイしてもらった。そのテストプレイでは、大半の方が前後の音が把握できないという意見を言っていたが、南雲氏は簡単にクリアできた。スピーカーの性能が悪くても、近づけば音が大きくなり、遠ざかれば音が離れるということに気付けばプレイできるということが分かった。

その後、ゲーム性を発展させる意見を数多く頂いた。チュートリアルの内容や難易度の設定は、この時に頂いた意見を反映した部分もある。他に、今期の開発範囲外の意見（対戦の形式や、3D のゲームを作成する場合の計算方法）も頂き、今後の発展の参考にすることができた。

3.3.1.3. 評価

南雲氏の評価として、「技術的な問題もあるし、音だけのゲームが面白いのか不安だったが、この日出来たものを見て発展できそうだと感じた」という意見を頂くことができた。

今後の発展材料として「映像の表現によるプロモーションのおかげで売れたゲームもあったが、映像のないゲームとして情緒を感じさせるようなゲームを作成することができれば良い」という意見を頂くことができた。

3.3.1.4. 結論

発注者である南雲氏に満足していただけたと言うこと、数多くの発展材料があり我々の成果物が今後進化をすることができるということがあり、我々のゲームプロジェクト目的であった「視覚にハンディキャップを有する人、健常者がともに楽しい時間を共有することができる、「映像のないゲーム」を開発するための課題を解決すること目標とする」(1.3.2 参照)を達成することができたと考えている。

3.3.2. 健常者の評価

3.3.2.1. 概要

成果物の評価をしてもらうため、大岩研究室が開催している授業の受講者を対象に 7 月 19 日〜26 日の間、ユーザインタビューを行った。ゲームを 5 分程度プレイし、用意したアンケート用紙に回答する、という形式で行われた。

3.3.2.2. 手順

ユーザインタビューを行うため、まずユーザインタビューを行うための場所を確保するために研究室のスペースを借用し、また機材としてデスクトップ PC を 1 台借用した。PC にはヘッドホン及び我々の成果物を搭載し、実際にプレイできる環境を用意した。続いて、先述の授業の受講者を対象にアンケートに協力を求めるメールを送信した。

3.3.2.3. アンケート内容

アンケートには以下の項目が含まれている。1〜4 用意した回答から選択する形式になっている。以下が、項目と回答の対応である。

問 1. 映像を用いないゲームをプレイした感想
- 非常に面白い
- 面白い
- 普通
- つまらない
- 非常につまらない

問 2. 映像を用いないゲームをまたプレイしたいと思うか
- 是非プレイしたい
- 機会があればプレイしたい
- どちらでも良い
- あまりプレイしたくない
- 二度とプレイしたくない

問 3. 面白かったと感じた部分（複数回答可）
- 映像を用いないという新しいゲームを体験できる
- 音がサラウンドで聞こえる
- 音声による説明が充実している
- ハイスコアを競うことができる
- ゲームの難易度･バランスがちょうど良い
- その他（記入可）

問 4. 改善･追加するべきだと感じた部分
- マップ上のトラップや時限爆弾などの聞こえ方
- 音声によるガイド
- ゲームの操作性
- ゲームの難易度･バランス

- 他人と競う部分（スコア以外）
- その他

問 5．その他、自由意見

3.3.2.4．結果

合計 9 名の人にアンケートを回答してもらうことができた。

問 1 の回答結果は、「非常に面白い」が 1 名、「面白い」が 3 名、「普通」が 3 名、「つまらない」が 2 名であった。普通から高めの評価を受けることができていることがわかった。

問 2 の回答結果は「是非プレイしたい」が 4 名、「機会があればプレイしたい」が 3 名、「あまりプレイしたくない」が 2 名であり、映像の無いゲームに面白いと感じる要素があったと感じてもらえることができたことがわかった。

問 3 の回答結果は「映像を用いないという新しいゲームを体験できる」が 6 名、「音がサラウンドで聞こえる」が 5 名であり、既存のゲームとは異なる点を高評価してもらうことができた。

問 4 の回答結果は「マップ上のトラップや時限爆弾などの音の聞こえ方」が 7 名、「音声によるガイド」が 6 名、「ゲームの操作性」が 5 名、「ゲームの難易度・バランス」が 3 名であり、まだまだ発展の余地が残っていることが分かった。特に音の聞こえ方に問題があるという意見が多くあったが、5 分程度のプレイでは慣れることが難しく、またヘッドホンの影響もあるため、発展に反映するのが難しい部分であると感じた。また、ガイド部分や操作性については、今後複雑なゲームになるにつれて重要となる箇所なので、更に考慮する必要があると感じた。

問 5 の回答としては、チュートリアルについてや難易度、環境についてといった、上記の回答をフォローする物が多かった。

3.3.2.5．結論

「面白い」という意見や「またプレイしたい」という意見が多く、我々の成果物は「面白そうなものである」と結論付けることができた。

また、初めてプレイした人がどのように感じているかという貴重な意見を頂くことができた。優秀なゲームに必要な要素として「何度もやってもらえる」ものでなければならない、というものがある。ゲームとしての質を向上するため、今後の発展材料となる意見を数多く得ることができた。

4. 個人レポート

4.1. 個人レポート（橋山）

4.1.1. プロジェクトを通しての活動内容

私は、株式会社ユードーの南雲代表取締役とともに「映像のない」ゲームを開発していくための問題を解決するためのプロジェクトである「さうんど おんりぃ」プロジェクトに所属して、活動を行った。ここでは、その活動の詳細を述べていく。

4.1.1.1. 企画決定まで

まず、「映像のないゲーム」の企画をメンバーで持ち寄って、それを発注者である南雲代表取締役に確認してもらうことになった。私は、映像を用いずにゲームを行うということを考えたとき、真っ先に探検をするといったゲームが思いついた。これは、私が自然の中など未知の領域を探検することが好きだからである。例えば、川のせせらぎや虫の鳴き声、葉っぱを踏みしめる足音などから、自分が森にいて近くに川が流れているところが想像できる。そして、音を頼りに森の中を探検して、色々なものを発見するというゲームである。しかし、森の中ではゲームの目標が設定しにくいため、宝探しという分かりやすい目標に変更した。それに伴い、舞台も森の中から古代遺跡に変更した。

プレーヤーは宝を探しに遺跡に入ったが、途中で落とし穴に落ちてしまい、持っていた照明器具を壊してしまう。真っ暗闇の中、プレーヤーは金属探知機の音だけを頼りに宝を目指す、といったストーリーである。遺跡の中では水がしたたる音やコウモリの羽ばたく音などが聞こえている。また、金属探知機はトラップや宝に反応して、色々な音を鳴らす。音の種類と大きさから、自分の近くに何があるのかを判断して、トラップだったら避けて、宝だったらその方向に向かう。

メンバー全員の企画自体は南雲代表取締役に認められたので、他のメンバーの企画と比べてどの企画を採用するかみた。私の企画では、壁があることをプレーヤーにどのように知らせるか、という問題があった。壁が音を反射することで存在を知らせるという意見もあったが、その場合は音の種類が多くなりすぎて煩雑になりすぎる。検討の結果、堀氏の提案した対戦型ゲームに他の 3 人のゲームの要素を少しずつ取り入れたような企画にすることが決定した。企画詳細については、本書の「2。企画詳細」を参照していただきたい。

4.1.1.2. ユースケース vs. マニュアル

企画が決定したことで、実際に要求分析に入った。メンバー全員でユースケースを書いたが、いまいちどんなものかゲームの全体像が見えてこない。そこで、UML は人に伝わる手段であるという認識をメンバーで共有して、ユースケースの代わりにマニュアルを書いてみるという作戦になった。メンバーを 2 グループに分けて、1 グループはマニュアルを書き、もう 1 グループは組み込み UML を使ってユースケースを書くことにした。これは、普通の UML よりも組み込み UML のほうがゲームの開発に近いのではないか、という疑問を検証するためである。

私は、組み込み UML を用いたユースケースを書くグループとなった。研究室にあ

った組み込み UML の本を借りて、そこにあったキャンディソーターのユースケースを参考に「映像のないゲーム」のユースケースを書いた。キャンディソーターでは、キャンディソーターを動かす人間をアクター、キャンディソーター内のモーターや IC チップが副アクターとなっていた。ここから、「映像のないゲーム」をプレイする人間をアクター、「映像のないゲーム」の中で動作するキャラクターやトラップなどを副アクターと捉えることで、前回書いた UML より分かりやすいものが書けたのである。

各グループが持ち寄ったユースケースとマニュアルを元に、どちらがゲームの開発に適しているか、という議論が行われた。組み込み UML で書いたユースケースは前回よりも分かりやすくなったが、それでも、まだ伝わりにくい。また、マニュアルは伝わりやすいのだが、詳細な仕様を書かないためユースケースが必要となる。結論として、マニュアルとユースケースの両方を用いて要求分析を行うということになった。ゲーム開発が今までのソフトウェア開発のプロセスとは異なると感じた最初の
ポイントであり、組み込み UML のことも知る良いきっかけとなった。

4.1.1.3.スコープ記述書の作成

全体分析と同時に、このプロジェクトでどこまで作るか、という範囲を決めるためスコープ定義書を作成した。この仕事は私が担当となったが、スコープ定義書を書くのは初めてだったので PMBOK を参考にしてどのようなことを書けばいいのかを調べた。その結果、プロジェクトの目標や成果物スコープ、成果物の受け入れ基準などを書いたスコープ定義書の第 1 稿が出来上がった。今後、プロジェクトで「映像のないゲーム」を作っていくときに、このスコープ定義書がすべての基準になるとのことであったので、そのような重要な文書を作ったことでプロジェクトの管理に貢献できたのではないかと考えている。

4.1.1.4.反復作業 1 回目

全体としての分析が終わったところで、いよいよスコープを絞った反復作業 1 回目に入った。ここではゲームの根幹部分を作って、とりあえず動くものを完成させようという方針であった。そのため、プレーヤーと目標（時限爆弾）が存在して目標から音が聞こえてくる、という部分のみを作ることにした。

ここで実装を行うにあたって、Visual Studio 。NET 2003 を使うこととなり、メンバー全員が研究室でこれをインストールすることにした。しかし、なぜか私のパソコンにだけインストールすることができず、ウェブ上で色々原因を探っても一向に分からなかった。VS は Windows のコアな部分と絡んでいることが多いため、インストールやアンインストールに失敗したら、OS を再インストールするのが一番早いという意見が多かった。結局 1 週間ほど試行錯誤したが解決しなかったため、2 日がかりで OS を再インストールして環境を構築した。プロジェクトの進行に影響が出る前に、OS を再インストールする決断が出来てよかったと思う。

また、実装時のソースコードやその他文書をメンバーで共有するために SVN を利用してファイルのバージョン管理をすることとなった。当初、VS 上から Subversion

を利用する予定であったが、VS のプラグインでは日本語ファイルが文字化けしてしまうため、TortoiseSVN というクライアントを利用してバージョン管理をすることにした。私はSubversionの設定やTortoiseSVNのインストール方法などをまとめた。

実装のための準備を終えて実装に入った。私は、デバッグ用の画面を作成して、キー操作によってキャラクターが動くという部分を実装した。C++に不慣れだったため、メンバーの堀氏や大岩研究室の篠崎氏に協力して頂き、なんとか実装することができた。また、音を鳴らすときに使っていた CRI Audio の仕様で不明点があったため、メンバーの堀氏とともに CRI・ミドルウェアに訪問した。今後不明点や質問があった場合は、調整役であった南雲代表取締役を通さずに、メンバーが直接 CRI・ミドルウェアの担当者に問い合わせても良いという許可を頂いた。気軽に質問ができるという状況を作ることが出来たことは、今後の開発において非常に心強いことであった。

4.1.1.5.ユーザーインタビュー

反復作業 1 回目が終了したのを受けて、慶応大学の中根氏にその時点で出来上がっていたゲームを遊んでいただき、意見や感想をもらった。私は中根氏とのスケジュールの調整を行ったが、メンバー全員の都合がなかなか合わずに、インタビューの実施時期が遅くなってしまったことが反省点である。

実際のインタビューでは、「映像のないゲーム」のこと以外にも、最近のゲームではどのようなものをプレイするのかということを聞いた。すると、「電車で GO」という答えが返ってきた。以前、横浜市立盲学校の生徒達も同じように「電車で GO」がプレイしたいと言っていたことを思い出した。「電車で GO」は電車が止まる時の音などがパターン化されている。そのようなゲームであれば、映像がなくても十分にプレイできるということを改めて知ることが出来た。

4.1.1.6.反復作業 2 回目

中根氏のインタビューを元に、ゲームを面白くするスコアやゲームを分かりやすくするチュートリアルの実装を中心に反復作業 2 回目が開始された。さらにメンバー内でプレイした結果、いくつかの改善点や追加仕様も決まった。

反復作業 2 回目では、要求分析を担当した。それまでの設計を大岩研究室の松澤氏にレビューしてもらった。詳細クラス図を評価するためには、その機能を網羅した詳細なユースケースが必要であるというレビューを踏まえて、ユースケース図と文書を書き直した。

4.1.1.7.成果物評価会

反復作業 2 回目がある程度終盤に近づいた頃、発注者である南雲代表取締役をはじめとして、ドルビー社や CRI・ミドルウェア社の方々を交えた成果物評価会を行った。ここでは、音の聞こえ方に関する問題と、ゲーム性を向上させるための工夫について色々な意見を伺うことができた。中でも、印象に残っているのは、5.1ch サラウンドのスピーカーを使って対戦を行うときの方法である。私は、プレーヤーが全員 5.1ch サラウンド対応のヘッドフォンをかけてゲームを行うか、通信対戦に

するくらいしか解決方法が思い浮かばなかった。しかし、人間のいる位置に応じてスピーカーから聞こえてくる音の聞こえ方を変えることができるといった技術があることを教えて頂いた。例えば、ゲーム内で音の発するものを投げたとする。投げた側の人は音が遠くなるように聞こえて、投げられた側の人は音が近づいているように聞こえるということが、同じ空間にいても可能なのである。他にもゲーム性の向上において役に立つ情報をたくさん頂き、とても有意義な評価会となった。

4.1.1.8.最終発表会に向けて

最終発表会のプレゼン資料や最終報告書の目次案は 6 月半ばから作り始めていたため、多少余裕があった。しかし、最終発表会のプレゼンを直前になって変更することにした。今までは、最終報告書に書くことをまとめた活動報告であったが、新しいプレゼンでは今後同じようなゲームを開発する人たちに向けての提言が中心となっている。これは、単なる活動報告では聞いている方が退屈してしまうのではないかと考え、他のプロジェクトとは異なるプレゼンをしようというメンバーの意向である。

この決定を行うために、5 時間に及ぶミーティングを行った。誰に対してどのようなプレゼンを行うのか、ということを念頭に置き、10 分という限られた時間で自分たちが伝えたいことを最大限にアピールするためのプレゼンを考えた。

また、最終報告書ではプロジェクトの概要と流れを記述した。これは目標の設定とも関連しているため、次項で詳しく述べる。

4.1.2.目標の設定と達成について

4.1.2.1.設定した目標（目標管理シートより抜粋）

（1）今までゼロから物を作った経験があまりないので、チーム開発を通してソフトウェア（今回はゲーム）開発プロジェクト全体の流れを把握する。本プロジェクトを知らない第三者に概要が説明できることが、プロジェクトの流れを把握しているという目標の評価基準となる。また、アウトプットとして、最終報告書にプロジェクトの流れを記述する。

（2）PM の補佐をしながら、プロジェクトの管理方法を学習する。本プロジェクトでは、PMBOK の定義の監視・コントロールフェイズについて学習することを目標とする。これを達成するために、スケジュール管理やコミュニケーション管理について、PM に具体案を提示する。プロジェクト終結時に、私の提案がプロジェクトにどれだけ貢献したかということを評価基準とする。アウトプットとしては、最終報告書の個人レポートとして、プロジェクト内でどれだけプロジェクトの管理が補佐できたかを記述する。

4.1.2.2.目標の達成について

（1）について、最終報告書のプロジェクトの概要という項目にプロジェクトの全体像をまとめた。

また、プロジェクトの流れについては PMBOK におけるプロジェクトの流れを整理し

た。最終プレゼンテーションでプロジェクトの概要を説明することで、プロジェクトの流れを把握しているかどうか評価できると思われる。

（2）について、ユーザーインタビューの調整、終盤スケジュールの作成などスケジュールに関しては大きく貢献した。また、メールや実際に会うことで頻繁にコミュニケーションを取り、情報の誤解や行き違いが起こらないように努力した。また、報告作業を積極的に行うことで新規履修者の参考になるようにした。

4.1.3.感想

4.1.3.1.プロジェクト全体を通して

今回、このプロジェクトが始まる前に、株式会社ユードーの南雲代表取締役と下準備を進めてきた。ドルビー社や CRI・ミドルウェア社など各企業にも訪問して、春休みを費やして企画してきた。実際に 4 月になって学生が集まるかどうかという心配があったが、その心配は杞憂に終わった。

実際にプロジェクトが動き出してからは、とにかく動くものを作ることが最重要であった。いくら面白い企画であっても実際に音だけでゲームを行うことが確認できなければ、それは画餅になってしまう。幸い、プロジェクトメンバー全員が週に 15 時間以上このプロジェクトに貢献できると回答していたので、時間的な余裕はあったと思う。初めのうちは新規履修者 2 名にプロジェクトがどのように進んでいくかということを知ってもらうために、率先して進捗報告をしたり、メールを送ったりした。プロジェクトメンバー全員が個性的で自己主張をするため、良い意味で常に議論が絶えないプロジェクトであった。

PM の中川氏も積極的にメンバーとコミュニケーションを取ったり、資料をレビューしてくれたりした。管理は今までのどのプロジェクトよりも厳しかったが、毎週出されるアクションアイテムやリビジョンされるスケジュールは、自分たちの立ち位置を明確にしてくれた。今、何をやらなければいけないのかということを各メンバーが認識しながら作業を行うことができたため、スケジュール的にも余裕ができたのである。

また、新規履修者であった堀氏が C++に精通していたため、一番困難だと思われた実装の面もスムーズに行うことができた。6 月の半ばでとりあえず最低限遊べそうなものが出来たので、一安心した。

今回のプロジェクトで特に気を使ったのは時間の配分である。4 年になって授業は減ったものの、大学院の入試準備や 2 輪の免許取得、研究室の仕事など、本プロジェクト以外にも時間を割かなければいけないことが多かった。昔から時間に追われたときに自分が使う方法は、やることをリストアップして優先順位をつけることである。優先順位をつけた結果やらなければいけないことは、いつまでにやる必要があるかを確認して、いつやるかを決めるのである。やると決めたら徹夜してでもその日中に終わらせることで、過密スケジュールでもなんとか作業をこなしていけた。

7 月 18 日に大学院の合格発表が行われ無事合格することができたので、「映像のないゲーム」の開発は大学院の私の研究として引き継がれる。そのことを考えても、今回のプロジェクトで発注者に一定の評価をもらえたものを開発できたことは非常

に大きな成果である。

また 3 ヶ月という限られた時間ではあったが、自分たちが作ったゲームに対して視覚にハンディキャップを有する人やプロのゲーム開発者の人たちから意見をもらうという貴重な経験ができた。映像を用いないでゲームを開発して面白いゲームができるのか、という問いを持ち続けながらこのプロジェクトを続けてきたが、発注者やユーザーの評価を受けて「映像のないゲーム」は面白いゲームになるだろうと確信することができた。今後も、このプロジェクトのメンバーが望めば、一緒に映像のないゲームについての開発・研究を進めていきたいと思う。

4.1.3.2. PMBOK について

今回、世界中で広く認知されているプロジェクトマネジメントの知識体系である PMBOK に従ってプロジェクトを進めてきた。PM からこのことを聞いたとき、最初はなぜ PMBOK を使うのか、PMBOK を使ったらプロジェクトはどのように変化するのかという疑問が頭の中を渦巻いていた。

しかし、プロジェクトが進むにつれて PMBOK に従っているということが頭の中から消えていった。前回までのプロジェクトでやってきたことと、今回のプロジェクトでやっていることに大きな差を感じないのである。そのことを改めて考えた時に、PMBOK は何も特別なことをしていないということに気づいた。

PMBOK は世界中で日常的に使われているプロジェクトマネジメントにおける知識体系をまとめただけである。私たちは日常生活の中でプロジェクトと呼ばれるものを行うときには無意識のうちに PMBOK に書かれているような内容を実行しているのである。つまり、逆に言うと私たちがプロジェクト内で意識せずに行っていることに名前をつけて体系づけてまとめたものが PMBOK なのである。

これは私個人の意見だが、PMBOK とはプロジェクトを行っていて何か問題が発生したときにその解決策として最も一般的な解を示しているだけであり、それに従うかどうかは本人の自由である。プロジェクトの初心者は分からないことが多いため PMBOK を参考にすると効率が良い。PMBOK に従ってプロジェクトを行うことで、一般的なプロジェクトというものを知ることが出来る。その上で、自分に合った別の方法があればそちらを採用して、自分なりのプロジェクトマネジメントを作り上げる。そのときに、参考として PMBOK を利用するのが良いのではないかという結論に至った。

4.1.3.3.最後に

本プロジェクトの発注者であり、協力企業との調整を行ってくださった株式会社ユードーの南雲代表取締役、プロジェクトを統括してくれた PM の中川氏、一緒にプロジェクトを行ったメンバーの佐藤氏、堀氏、渡辺氏、その他レビューや実装のときに協力して頂いた大岩研究室と PMO の皆様、他本プロジェクトに協力して頂いたすべての人々に感謝します。ありがとうございました。

4.2. 個人レポート（佐藤）

4.2.1. 目標の確認と達成の如何

4.2.1.1. 目標の確認

私がこのプロジェクトに望むにあたり、目標としていた点が２つある。それを今もう一度確認してみる。

- 今までに体験したプロジェクトは全て「元あるモノの改良」だったということもあり、何も無い段階からモノを作る場合の手順・手法を学びたい。最終的にはプロジェクトで取った手法やマネジメント方法について纏めたレポートを記述することを目標とする。
- 3 期目ということ、また就職活動を目前に控えているということもあり、制作活動を通じて自分の適性を把握したい。プロジェクトを通じて、自分のプロジェクト内での役割と仕事内容、またその進行度合を記録し、自らの仕事を振り返っての評価を記したレポートを作成し、自分の適性を確かめることを目標とする。

では、続いて自分がどれだけこの目標を達成できたかを確認したい。

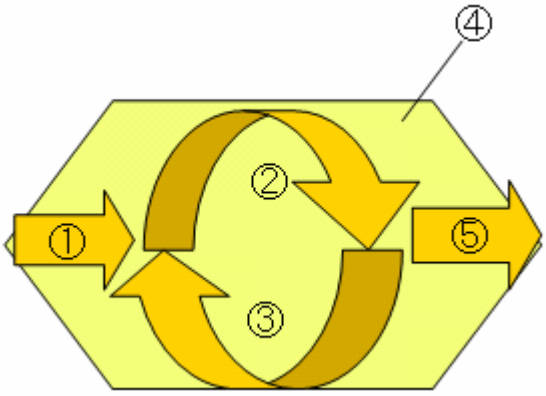
4.2.1.2. プロジェクトで取った手法について

4.2.1.2.1. 用いた手法についての簡略なまとめ

ここでは、今回さうんど おんりぃプロジェクトで用いられた手法について、私自身の体験も交えて纏める。

1. PMBOK

今プロジェクトは PMBOK に従って活動が行われた。PMBOK とは、プロジェクトのマネジメント手法であり、世界中で広く認知されている知識体系である。



PMBOK には、5 つのフェーズがある。①立ち上げ ②計画 ③実行 ④監視・コントロール ⑤終結である。まず立ち上げフェーズから入り、計画フェーズと実行フェーズをループして実施し、目標を達成した時点で終結フェーズへと移行する。これらの全てのフェーズを監視するのが監視・コントロールフェーズとなっている。

今回の我々のプロジェクトでは、企画・立案が立ち上げフェーズ、学習・準備及び設計が計画フェーズ、反復実装作業が実行フェーズ、そして最終的なドキュメントの作成といった作業が終結フェーズに属すると考えている。また、計画とのずれが生じていないか、設計通り行っていないのではないかといった点を管理するのが

監視・コントロールフェーズであり、意識はしなかったものの常に行われていたということになる。

各フェーズが終了する毎に、成果物を製作する。例えば、我々のケースで言うと、企画・立案フェーズの成果物としてスコープ記述書があり、計画フェーズの成果物として設計書、実行フェーズの成果物として製作したさうんど おんりぃのゲームが成果物といったような成果物がある。これらは、フェーズを終了させたという証拠となり、曖昧さ回避のためにも必要な物であると考えている。

2．WBS

また、今回のプロジェクトでは WBS によるスケジュール管理が行われた。WBS とは Work Breakdown Structure の略で、プロジェクト全体の大きな仕事を細かい作業に分割したものである。各 WBS の達成時期や担当者を決定することにより、進捗管理や作業漏れが無くて済み、リスクが軽減される。また、一目で現在の進捗が把握できるため、何をすべきかが分かりやすく、作業が円滑になる。

3．EVM

EVM の活用によるスケジュール管理も行われた。EVM とは Earned Value Management の略であり、プロジェクトのパフォーマンスを費やしたコスト（時間）で定量的に測定し、作業の進捗や達成度の管理を行うプロジェクト管理法である。これを活用することにより、進捗の状況をグラフという形で目に見えて確認することができる。

4.2.1.2.2.　用いた手法について学んだこと

1．PMBOK

PMBOK で用いた手法についてであるが、はっきり言って全てを理解できていないと思っている。しかし、このようなプロセスでプロジェクトが進められるのであるということが分かっただけでも良かったと思うし、はっきりとではないが「何も無い段階からモノを作る場合の手順・手法」がどのようなものであるかと言うことは把握できたと思っている。

特に、プロジェクトが一連の流れで回っているのだと言うことが理解できたことは大きいと思っている。異なる作業の連なりなのではなく、内容は違っているが、外観的には全て同じプロセスの繰り返しであり、大差が生じることはないということを理解できたということ。この経験は、他のプロジェクトに取り組む際に、前と全然違う性質のプロジェクトであるとか、細かな作業の内容にとらわれず、何をすればプロジェクトの目標を達成できるかということが理解できているため、作業への取り組み方が以前とは異なるのではないかと思う。

また、自分が現在どのフェーズにいるかと言うことを意識し、それぞれに達成条件を設けるということにより、プロジェクト内にはっきりとした役割意識があったと思う。やはり、何事もゴールがはっきりしていると遂行がしやすいものであるということを再確認した。

2．WBS

今回のこのWBSによるスケジュール管理であるが、かなり便利なものであると感じた。進捗具合による記入をするのが面倒ではあるという欠点はあるが、何を誰がどこまで行うかということが簡単に見ることができるし、どの項目の事項を達成することが目標の達成に必要なのかということも理解することができる。自分たちのスケジュール管理のためだけではなく、ドキュメントを記述する際にも助けとなることであるし、急な変更に対する対応性もある。今後のプロジェクトでも是非活用して欲しいと思った。

3．EVM

EVMに関しては、たまに中川さんが見せてくれるという程度であり、そこまでじっくりと効果を意識して閲覧したことは無かったが、どれだけの効率で作業を行い、また進捗はどれくらいなのかと言うことが理解することができたのはよいことだと思う。しかし、反復第二回に入った時点付近の費用対効果がそれまでに比べれば優れている、ということは分かったのだが、その理由が知りたいと感じた場合、どのようにすればその理由を抽出できるのかが知りたいと思った。しかし、現段階で私がコストについて理解できているとはいいがたく、まだ実用するには勉強が必要だと感じた。

4.2.1.2.3．用いた手法について学んだこと

例えばの話であるが、私が社会人になり、プロジェクトマネージャーをすることになったとすれば、現段階では私は今回中川さんが取った手法と全く同じ手法を取ろうと思うだろう。それは、私が把握しているマネージングの方法が他にないと言うこと以外に、今回この手法を用いることによって生じたマイナス点が、私の立場からは感じられなかったということが挙げられる。どのようにすればプロジェクトを遂行することができるのか、という漠然なイメージではあるが理解できたと言うことは今後の役に立つと考えている。

4.2.1.3.　自分の適性について

4.2.1.3.1．今期行った主な作業

まず、今期のさうんど おんりぃプロジェクトにおいて自分がどれだけの作業を行うことができ貢献することが出来たかを考えるため、私が携わった主な作業内容を挙げる。

・ 企画・スコープ範囲決定

1. 企画案を持ち寄り、発表を行った
2. 全員の案を下に議論を行い、企画とスコープの決定に参加した

- 全体設計
    1. インスタンス図・ユースケース図・クラス図・シーケンス図を作成した
    2. 各自の作成した UML 図を下に、最終的な UML 図の作成を行った
    3. 作業中に変更となった点を設計図に反映させた
- 反復第一回目
    1. 全体設計における設計図を元にした、反復第一回目の設計図の作成
    2. スペック決定の議論への参加
    3. マップ内オブジェクトクラスと言った、小範囲の箇所の実装
    4. システム仕様一覧の製作
    5. 試験項目の製作
    6. 反復第一回目の成果物ドキュメントを作成した
- 反復第二回目
    1. 音の連続再生やタイトル画面といった小範囲の箇所の実装
    2. システム仕様一覧の製作
    3. 試験項目の製作
    4. ユーザーレビューの準備
- その他
    1. 進捗報告資料制作・発表
    2. 中間発表資料製作・発表
    3. 視覚にハンディキャップを有する人への評価時、ビデオで手元やプレイ中画面の撮影をした
    4. 発注者評価のための発表資料の作成
    5. 進捗報告の資料作成・発表担当（2 度）

4.2.1.3.2. 作業のまとめ

上記の通り、今プロジェクトでは満遍なく異なる作業を行った。各分野の作業に置いて、私がどれだけの仕事ができたのかを自己分析してみる。

4.2.1.3.2.1. 企画

企画においては、私は第 1 回企画考案会では「鬼ごっこゲーム」、第 2 回企画考案会では「捕獲ゲーム」を考案した。

「鬼ごっこゲーム」は、1～2 人のプレイヤーで遊ぶもので、一方が逃げ、一方が追いかける、というゲームである。全体企画に採用された部分も多かった。

「捕獲ゲーム」は 1 人用ゲームで、逃げる対象を追いかけるというゲームであった。対象は AI を搭載し、また障害物も自分を追いかけてくるというゲームであった。

統括として、どちらもオリジナリティという点では低かった。私は発想力が豊かなほうではないことは自分でも分かっていた。授業の最終課題でも、ありきたりな物や元ある物のリメイク・再現が多く、オリジナルの何かを作るのは苦手分野であった。

キックオフミーティング時、チーム名を決める話し合いが合ったのだが、他のメンバーが面白いアイデアをポンポン出している中、自分も必死に考えたが何も

思い浮かばず、歯がゆい思いをしたこともあった。
　まとめとして、「もともとあるそれっぽいものを考案するのはできるが、斬新なアイデアを考案する発想力は無い」ということが言える。

4.2.1.3.2.2.　設計

　今回、設計は全体設計と反復作業第 1 回目において手がけた。自分がどれだけ UML や設計について理解できているか不安なところはあったが、割と問題なく設計をすることができた。完璧ではなかったが、自分の理解が以前に比べて格段に深まっているということが理解できたのは収穫であった。
　ただ、そこまで複雑なプログラムでなかったということは作用していると思う。例えば、対戦に対応したモノを作らなければならない場合、設計を上手く行うことができたかは不安なところがある。
　まとめとして、「今回は規模の問題で正確な技量はわからないが、全く話にならないというレベルではない」ということがわかった。

4.2.1.3.2.3.　実装

　今プロジェクトにおける実装は殆ど堀君に担当させてしまい、あまり手がける部分は無かった。数少ない手がけた部分を思い起こしてみると、C++に触れるのが初めてということもあり、予想よりも時間がかかったというケースが多くあった。例えばステージ選択を作成する、というタスクでは、他の授業で学んだ C の技法を使用したためシステム終了時にエラーが出てしまう、というケースがあった。C++に対応した記述法を探すのに結構な時間がかかり、慣れの問題もあるだろうが、効率のよう実装ができたとは言えなかった。
　ただ、振られた実装タスクは全てこなしているので、これも全く話にはならないというレベルではなかったと思う。しかし、最終的な成果物のソースコードの全てを理解しているわけではない。
　まとめとして、「今回のプロジェクトで実装のスキルアップは殆ど無かった。レベルとしても、全く分からないというレベルでも良く理解できているというレベルでもない」という結論に至った。

4.2.1.3.2.4.　発表

　今プロジェクトでは、中間発表の担当をしたことが一番の思い出である。資料を何度も作成しなおし、練習もみっちりやったおかげで、時間通ピッタリに発表を終わらせることができた。
　しかし、発表内容は「つまらない」ものであり、もっと考える余地があったと思う。相手は何を見たいのか、自分たちは何を見せたいのか、ということを考えた資料作りをしなければならないということを学んだ。
　最終プレゼンテーションの内容決めのミーティングでも、「君は何を見せたいの」という問いに対し、全く何も思い浮かばなかった。成果物を見せ、その経緯を見せるというものが最終プレゼンテーションであるという既成概念が出来上がっていたからだと思う。もうちょっと柔らかい頭というか、せめて自分がどう

したいのかということは理解しておきたいと思った。
　まとめとして、「お堅いプレゼンテーションならできるが、型を破ったプレゼンテーションはできない」ということがわかった。

4.2.1.3.2.5.　ドキュメンテーション

　ドキュメンテーションであるが、反復第 1 回目の成果物ドキュメントや、ユーザーレビューのための資料の作成などを行った。
　成果物ドキュメントの方は、メンバーに言われるまで自分たちの意見を入れようというアイデアは思い浮かばなかった。読む立場のことを考えたドキュメント作成が行えていないということである。
　ドキュメント作成は苦ではなく、それまでの作業を思い起こしてそれを記述するという作業は楽しく感じた。
　まとめとして、「つまらないドキュメントを作成するのはうまい」という結論に達した。

4.2.1.3.2.6.　その他

　最終ドキュメントのマージング作業や、試験項目の書き出しといった、割と機械的な作業は、確かにつらいとは感じたが、スムーズに行うことができたと思う。

4.2.1.3.3. 自分の適正について

　上記のことから、自分は「斬新な考えを考案したり、既存と異なる手段を取るということが苦手である。しかし、機械的な作業や形式の決まっている仕事をすることは得意」な人間であることがわかった。
　自分で書いておいてなんだが、悲しいとは思う。どうせなら逆のタイプの人間になりたいと思うが、努力次第でなんとかできる問題なのかと言えば私はそうではないと思う。発想力が豊かになる本などがあれば是非読んでみたいが、小学生のナゾナゾにも引っかかる自分が発想力の豊かな人間になれるとは思えない。
　なので、私は既存の手段を用いてタスクをクリアするという、地道な方向を歩むべきなのではないかと思っている。手段を問わず何かをしろ、という指令があるとしたら、既存の手段を用いてどうにかする。一応、最低限の「自分なりの考え」は反映させつつ、周囲が思っている通りの成果が挙げられる、そのような人間になることができればと思う。
　このタイプは決してネガティブな考えに基づいて生まれたものではなく、努力次第で幾らでも貢献することができる、周りにいたらありがたいタイプだと思う。無難かも知れないが、仕事をこなすことができる人間、それが私が将来的に辿り着くべき終着点なのではないかと考える。もちろん、発想力や独創性はもっと欲しいと思っていることには変わりは無いが。

4.2.2. プロジェクトを通しての感想

　一言で言えば、とても楽しいプロジェクトであったと思う。プロジェクトメンバーは全員頼りになる存在であったし、PM の中川さんも凄く真剣に取り組んでくれ、また

非常に気を使ってくれたので、とても良い環境で仕事ができたと思う。

この「音のないゲーム」は今後橋山君が研究していくと言うことになっているそうだが、将来自分が携わっていたということが自慢できるような物に仕上げて欲しいと思う。決してプレッシャーをかけるわけではないが。

何も無い段階からモノを作成すると言うこと、企業の方を PM にしての作業、そして 3 期目としての自覚を持っての取り組みと、あらゆる要素が新しいプロジェクトで、自分の目標を全て達成できたわけではないが、目標にしていた事柄以外にも非常に多くのことを学べたと満足している。

4.2.3. 最後に

共にプロジェクトに取り組んだ橋山君、堀君、渡辺君、そして中川さんに素晴らしいプロジェクト活動が行えたことを感謝したい。

「1 学期間、ありがとうございました。」

個人レポート（堀）

4.2.4. プロジェクト総括

今回、私はプロジェクトを通して 2 つのことを見極めたいと思っていた。1 つはゲームの設計における UML の有効性、もう 1 つは集団でコーディングを行う上で重要なことである。また、プロジェクト管理手法がどういったものかも興味があった。それぞれ考えたこと、得たことを見ていきたい。

4.2.4.1. ゲームの設計における UML の有効性

結論から言うと、プロジェクトの設計段階で利用した UML の大部分で、その有効性を実感することはできなかった。我々は結果的にユースケース図、オブジェクト図、クラス図、シーケンス図、状態遷移図の 5 種類を作成したのだが、ユースケース図と状態遷移図以外はほとんど有効利用されることなく終わってしまった。そのため、小規模なゲームのプロジェクトであれば要件定義でユースケース図・ユースケース記述を、設計段階で状態遷移図を記述し、それ以外は（設計段階では）むしろ使わない方が効率的に進む、という結論に達した。

その理由を、各図の作成状況・利用状況と併せて以下にまとめた。

4.2.4.1.1. ユースケース図

今回の制作にあたって、我々はまずユースケース図を作成した。ゲームは通常の情報システムとは方向性が異なるためユースケース図が有効かは怪しかったが、今になって考えるとユースケース図・ユースケース記述は要件定義に役立っていたと考えられる。というのも、ユースケース図やユースケース記述を作成する過程でゲームの要件が少しづつ固まっていったからだ。

目標としていたゲームがかなり単純だったため、あえてユースケース図を利用する必要は無かったのかもしれないが、その理念である「ユーザーの立場からの分析」という手法はゲームの要件定義に充分有効だと感じた。

つまり、あえてユースケース図という形態を取る必要は無いが、ユーザーの立場から要件を定義していく、という手法は使える、という結論に達した。

4.2.4.1.2. オブジェクト図

オブジェクト図は反復第 1 回目の設計段階で制作したが、これは作りっぱなしで活用されることはほとんど無かった。正確に言えばクラス図の作成において使われたが、そのクラス図が有効利用されなかったため実質活用されていない。

4.2.4.1.3. クラス図

クラス図は反復第 1 回目と 2 回目の両方で制作され、何度か作り直された。しかしこの作り直された、というのが問題で、結局クラス図は実装に合わせる形で改変されていたのだ。だとすれば、設計段階からクラス図を利用する必要はなかったのではないだろうか、というのが私の見解だ。設計ではクラス図は制作せず、実装が固まった時点で実装状況のリファレンス用としてクラス図を制作すれば充分だった。

その理由は、クラス図を先に書いたからといって実装の細部に渡って見通すことはできないからだ。今回のプロジェクトでも、いくら時間をかけてクラス図を書いたところでいざ実装してみると具合の悪い部分はいくつか発生し、そのたびにクラス図を修正する手間が発生していた。初期段階に記述したクラス図に限っては概念的な整理しか行っていなかったため、そのクラス図を実装に反映したがために無駄な抽象クラスをいくつか作成してしまった。

つまり、クラス図は実装ではほとんど役に立たず、むしろ手間を増やすだけの存在になっていたのである。

そのため、設計段階ではクラス図は作成せず、実装が固まった時点でリファレンス用として作成するやり方が適切だ、という結論に達した。

4.2.4.1.4. シーケンス図

今回使用した図の中で最も影が薄かったのはこのシーケンス図で、反復第 1 回目で作ったものの、本当に作っただけで以後は完璧に放置されていた。理由は、シーケンス図はゲームという形態と合わなかったからだと考えられる。

ゲームは基本的に自律的に活動するオブジェクトが大半である。ユーザーの操作に呼応して反応を示す部分もあるが、大半はユーザーの意思や操作とは無関係に行動する。同様に、1 つのオブジェクトは独立して活動する場合が多いため、他のオブジェクトと呼応して何かをする場合も少ない。そのため、時系列でオブジェクト同士のメッセージ活動をまとめるシーケンス図は作ったところで役に立たなかったのだと考えられる。

また、そもそもシーケンス図に記述された内容があまりに当たり前だったので、あえて参照する必要が無かった、というのも大きな理由だったと考えられる。

4.2.4.1.5. 状態遷移図

状態遷移図は反復 2 回目になって初めて作成された。反復 1 回目では状態が 1 つしかなかったため、作る必要性を感じなかったからだ。

結果として、この状態遷移図は後の実装にかなり役立つことになった。理由は、状態を整理したことで独立したオブジェクトが明らかになり、実装作業を分担することができたからだ。

それまでは明確に状態を整理していなかったため、他の箇所の変更により実装作業がムダになる可能性が高かった。そのため、私が 1 人で大半の実装作業を行っていたのだが、状態遷移図のおかげで他と干渉しない・されない独立したオブジェクトが明らかになり、その部分の実装作業を他のメンバーにふることができた。

4.2.4.2. ゲームの設計における UML の有効性まとめ

まとめると、ゲームの設計においてはユースケース図、状態遷移図が有効で、オブジェクト図、クラス図、シーケンス図があまり有効でない、という結論になった。

4.2.4.3.　集団でのコーディング

前項では UML による事前設計の必要性に疑問を投じたが、やはり設計そのものは必要だ、ということも今回実感することになった。その理由が、集団でのコーディングである。設計書が無ければ集団で同時並行的にコーディングをすることができないのだ。

「とりあえず作ってみないとわからない・使ってみないとわからない」という面が強かった今回のプロジェクトも、複数人でコーディングを行った部分はごくわずかで、大半のコードは私 1 人が書くことになった。磐石な設計ができてなかったため、他のメンバーと同時に進行しても結局ムダなコーディングになる可能性が高かったからだ。そして実際、反復第 2 回目では設計の根幹であるライブラリが変わったため、それまでのコードで不要になる部分が 200 行ほど出てきた。設計が脆かった結果である。

結局のところ、今回のようなノウハウの少ないソフトを作る場合はどうしても大きな変更点が多発するため、集団でのコーディングには不向きである。1 人、できれば 2 人のプログラマで色々と検討をしながらコーディングを行い、使用するライブラリ等と大枠の状態遷移とを確立させてから個別の独立した部分を集団でコーディングする、というのが最も現実的なやり方だろう。

今回に限ってはそもそもゲームとして成り立つかわからなかったためゲームプレイの部分から作りはじめたが、理想を言えば先にタイトル、ステージセレクト、チュートリアル、ゲームプレイ、ヘルプなどの状態遷移間の関係を設計した方が良かった。そうすれば各状態は独立したモジュールになるので、分割して集団でコーディングすることもできたはずだ。

4.2.4.4.　プロジェクト管理手法

私は今回初めてプロジェクト管理手法（PMBOK）なるものに触れたのだが、その雑感と、今回学んだ手法そのものの概要をまとめて行きたい。

4.2.4.4.1.　雑感

まずなにより、その特殊な用語の多さに辟易した。アクションアイテム、WBS（Work Breakdown Structure）、PV（Planned Value）、フィージビリティスタディ、など、知らなければ全く意味のわからない略語や用語が頻発していた。進捗報告会などでも「アクションアイテムとは何か」といった質問も出てきて、そのわかりづらさが目立っていたと考えられる。起源が海外なのでしょうがない面もあるが、もう少しわかりやすい用語に直せないものかと思った。

とはいえ、色々と小難しい用語で説明されてはいるものの手法の内容そのものはわかりやすいと感じた。(1)プロジェクトの目標と期限を決め、(2)期限までに扱う範囲を決め、(3)その上で人や資源を配分し（つまりスケジュールを練り）、(4)適宜監視・調整を行いつつプロジェクトを終結させる。感覚的に理解していることを厳密に定義し、体系だててまとめただけだ。工数やコストの計算など、ビジネス的な要素に馴染めなかったくらいだろうか。

4.2.4.4.2.　手法の概要

前項でも軽く説明したが、PMBOK とは我々が感覚的に理解しているプロジェクトの管理方法をまとめたものだ。大体のプロジェクト――それこそダムの製作から飲み会の開催まで――に共通している要素を抽出し、各要素・各ステップを厳密に定義し、その上でコストの見積りやスケジューリングで使える手法をくっつけた、というものだと考えていいだろう。

例として、学校のレポートを執筆する、というプロジェクトを考えてみる。「現代日本のソフトウェア業界が抱える問題点を 1 つ挙げ、その考えうる解決方法を 3000 字以内でまとめよ。提出は 2 週間後」、というレポートが出たとする。

4.2.4.4.2.1.　立ち上げ

これからそのレポートを執筆するわけだが、実はこのレポートが出題された時点ですでにプロジェクトで言う「立ち上げ」は完了している。

立ち上げでは本来プロジェクトの認可を得るため、プロジェクト憲章や成果物記述書などを出力しなければならないのだが、今回は執筆者がレポート執筆の認可を得なければならない存在は居ないため(あえて言えば、その存在は執筆者本人である)、それらの作成は全て省略できる。また、プロジェクトの目標や制約も(1)ソフトウェア業界が抱える問題点と解決方法が挙げられている、(2)2 週間後に提出、(3)3000 字以内、と必要なものは与えられているので、新たに制約を考える必要も無い。

4.2.4.4.2.2.　スコープの定義

次に考えなければならないのは、今回のプロジェクトで扱う範囲（スコープ）、言いかえればレポートで扱う具体的な題材を決定することだ。例で言えば、ソフトウェア業界が抱えるどの問題を扱うか、という点だ。これはもちろん、与えられた制約条件の中でできる範囲でなくてはならない。レポート側で提示されたテーマと文字数に沿っていることは当然のことながら、自分が執筆に取れる時間などの資源や、最終的に目標とする品質も考慮しなくてはならない。可能なら過去のレポートなども調査し、どういったレポートの評価が高く、どういったレポートの評価が低いか、なども調べた方が良い。一度決めた題材を教授にレビューしてもらえればなお良い。

ちなみに、品質の悪い成果物をスコープとして定義したとても、自分がそれを認可し、設定した目標を達成できてればそれはプロジェクト的には成功である。この場合で言えば、単位さえくれば良いというスタンスならスコープの品質を低く設定し、自分の時間と労力というコストを低く抑えて執筆する、というのも手ではある。

また本来スコープ定義が終わった段階でも認可が必要なのだが、立ち上げと同じく認可するのは自分なので、自分がそれで良ければ問題は無い。

4.2.4.4.2.3.　計画

次に行うのは計画だ。計画、つまりスケジューリングのことだが、具体的に

やるのは作業全体の設計と資源の配分である。例で言えば、いつ目次を作成するのか、いつ資料を調査するのか、いつ執筆を行うのか、といったことの決定である。大抵我々はこのあたりをあまり厳密に決めず、「何曜日の何時ごろに作業する」くらいのことしか決めないが、ビジネスでのプロジェクトでは WBS という詳細な作業項目と、その実施スケジュールを定義する。そのことでスケジュールが明確になるとともに、各作業担当者の責任追求もしやすくなるからだ。しかし、今回の例ではスケジューリングも作業も全て自分 1 人で行い、その責任も自分で負うため、あえて WBS の形で文書化する必要は無い。

なお、PMBOK ではこの計画を非常に重視しており、プロジェクトを左右する活動だとしている。そのため、いくら期限が迫っていたとしても計画は入念に行う方が結局は良い、ということになっている。

4.2.4.4.2.4. 実行・監視

計画まで完了すれば、あとはその計画に基づいて実行するのみである。目次を考え、図書館へ行って資料を漁り、執筆を行い、推敲を重ねる。作業内容としてはこのようなものになるだろう。その際、常に監視をすることを忘れてはならない。つまり、自分の現状とスケジュールとを比べ、どの程度の遅れがあるかを自覚し、その調整を行うことである。例えば、必要な資料が手に入らなくて、調査が長びきそうだからその分先に書けるところから執筆を始める、といったようなことだ。

また、推敲作業を行うことも非常に重要だ。予定されている内容を網羅しているかを確かめ、もし足りないようなら追加する。あるいは不要な部分を削除する。これらの作業はプロジェクトで言うところの試験にあたり、全体の品質を保証・向上する重要な手順となる。本来は試験の手順書なども作成し、その結果も文書として残す必要があるのだが、今回の例ではやはり不要だろう。

4.2.4.4.2.5. 終結

以上のような手順を踏み、実行と監視を繰り返すとやがてプロジェクトは終結する。正確に言えば、設定された期限に合わせてプロジェクトは組まれているので、期限が近づくとプロジェクトは終了する。最後に残っているのは成果物を提出することと、その評価を行うことである。今回の例でも文字通り、レポートの提出とその評価が行われる。ビジネス上のプロジェクトでは評価も含め、未来のために様々なことを分析し、文書として残しておくが、今回の例では特にそういったことを行う必要もないだろう。もちろん、そういったことも行うのがベストである。

4.2.4.4.3. プロジェクト管理手法まとめ

以上が、PMBOK から学んだプロジェクト管理手法の概要である。多くの点を省略したが、大筋としてこのようなものだと認識した。

4.2.4.4.4. 参考文献

• 『PMBOK ガイド 2000 年版』 2002 Project Management Institute Inc.

4.3. 個人レポート(渡辺)

4.3.1. 学習プロジェクト

4.3.1.1. 学習プロジェクトにおける個人目標の確認とその目標に至った背景

私がこのプロジェクトに望むにあたり、学習プロジェクトとして目標としていたのは「システム開発に利用する UML(Unified Modeling Language)の基礎的な理解を目標とし、インターネットで公開されている無料 UML 技術認定試験(http://naha.cool.ne.jp/overdrive/UmlExam/index.htm)を利用して理解度を示し、それについてレポートをまとめる」ということであった。

この目標に至った背景として、大岩研究会 1 期目である私は、システム(今回はゲームだったが)開発において右も左もわからない状態であり、今回のゲーム開発プロジェクト「さうんど おんりぃ」の開発環境である VC++や設計に利用する UML の勉強会に参加した。そこではじめて UML を知り、UML は作る予定のシステムについて共通の認識を得るものだとわかった。結果的には「さうんど おんりぃ」は今までにないものかつゲームであるため、UML 全てが役立ったとは言い難かったが、設計かつコミュニケーションの重要性を私に教えた重要なツールであった。

4.3.1.2. 学習プロジェクトにおける個人目標の達成

テストを受けた結果、48 問中 40 問正解で、合格という一定の基準を得られた。

4.3.1.3. 学習プロジェクトの感想

このテストの中にあった、

あなたは、ソフトウェア開発プロジェクトのプロジェクトマネジャーです。先日、品質監査のために、設計に関する成果物ドキュメントである約 30 枚の UML の図を提出しました。ところが監査の結果は「成果物ドキュメントに問題あり」という判定でした。最も考えられる問題点は？

1)UML の図の枚数が少なすぎる

2)UML の図の枚数が多すぎる

3)UML の図で書けないことが成果物ドキュメントとして提出されていない

4)内部設計が行われていない

という問題をみて、直接は関係ないが、ユースケース図を初めて書いたときにユースケース記述なしで図だけで表わそうとし、「情報量が足りなくてこの図は駄目」というレビューをうけたことが思い出された。これはもちろん単なる私のミスだが、「情報量が足りなくてわからない」というレビューで頂いた言葉は、UML で表わすことのできないものの文書・図化も大切であると考えさせられた。特に今回は細かい仕様変更が起こりやすいゲーム開発であり、UML のユースケース図に当たるのはゲームでいうとマニュアルだからそれを作ってみようなど様々なことをしたのが良い経験になった。

4.3.2. ゲーム開発プロジェクト「さうんど おんりぃ」

4.3.2.1. プロジェクトの流れとそれにたいする当時の感想

「やったこと」はほかの書類にあるので時系列に沿って当時の考えなどを書く。

4.3.2.1.1. 4 月編

第 1 回目の講義ではメンバーと PM が集まり、自己紹介などや当時は決まっていなかったプロジェクト名を決めた。その時に「さうんど おんりぃ」か「sound only」と表記方法で議論が起こった。議論というほどのものではなく、要するに揉めたのだが、そこで PM がその単に揉めただけの事についてすら問題の解決方法として

・強制…その中で最大の権力者がある解決方法を選択すること

・撤退…自分の意見を放棄すること

・妥協…折衷案などでお互い歩み寄ること

・対決…納得のいくまで話し合うこと

など用語があるという事を我々に教えてくれ、私はこれ以後毎回何か活動を行うたびに、その活動の概念を共有するための用語が既につけられていることに驚きながら活動することになった。また、初めての研究会である私は(当たり前のことであるのだろうが)毎回ミートログをつけるという細かさにも驚いた。

第 2 回目の講義ではどんなゲームを作りたいかというのをそれぞれ発表しあった。一番おもしろそうだった堀さんの全方向ガンシューティングをハードの問題で試すことなくあっさり切ったのが残念であったが、どう考えてもソフトウェアから離れた問題であったので仕方がなかった。なお、今回初出の用語としては

・スコープ…プロジェクトで何をやるか。実質自分たちがやる範囲

・WBS…ワークブレークダウンシステム。仕事を噛み砕いて構造化するシステムがあった。WBS 詳細、つまりこのプロジェクトで何をやるかは PM が来週までに作ってくるということであった。これが決まらないと動けない私たちにとって最優先すべきのことであり、後に PM はこの時期(の作業量)は辛かった、と述べている。

4.3.2.1.2. 5 月編

第 3 回目の講義では今回立体音響を表現するための CRI・ミドルウェアの導入などが設定されていたが VS Express Edition ではコンパイルできず、最終的には研究会に掛け合って製品版をもらうことになった。この件に関して PM が述べた言葉は「(人件費が一番高いので)金で解決できることは金で解決する」であった。やはり働くということはそういうことなのだろうかと考えさせられた。また、その話の内容の中にそれに関して

・ステークホルダー…利権に絡む人

という用語が出た。「さうんど おんりぃ」プロジェクトであれば南雲氏や我々メンバーや PM や大岩先生など要するに関わっている人のことである。

第 4 回目の講義辺りでは正直毎回 1 コマ目にある他のプロジェクトの進捗報告及び自分たちのプロジェクトの進捗報告の時間がもったいないと思っていた。そんな時間があるのであれば、もっと自分たちのミーティングや設計実装に充てたいと考えていた。

第 5 回目の講義は体調不良で休んだ。PM もウチの両親が言うように体が資本、健康が第一だとメールやミーティングでもよく言っていた。やはり年上の方は生きていく中でなんらかの体を壊した経験があり、そのような結論に至るらしいとわかった。

4.3.2.1.3. 6 月編

第 1 反復の実装編であるが、最初の部分以外ほとんど実装にかかわっていない上に途中でその部分は変更され、この時期は講義そのもの以外の時間、アクションアイテム時間はかなり少なめであった。なお、進捗報告の発表時に突っ込まれたアクションアイテムとは何なのか、という質問に対して 2 コマ目のミーティング中に

・アクションアイテム…締め切りがはっきりしているやるべきタスク(作業)

であると定義された。そのアクションアイテムが割り当てられてはいたが、実時間は少なめであったので、ゲームに必要になるだろう(無断で使用していい)フリーの音素材をゲームのストーリーに合わせて収集したりしていた。しかしこの時点で音声が必要であるとわかっていたが、人工音声にするのかいっそ人の声で録音するのかという点は曖昧なまま放置された。最初の WBS にない活動(後から必要だと判明したものなど)は優先度が下がったり、後手後手になってしまう事があると感じた。

この時期に自分が前に立って初めて進捗報告をして、毎週の進捗報告の重要性を感じた。毎週の自分たちの作業量がはっきりするし、どうしても出てくる内部では見過ごしてしまうような問題を外部から見てすぐさま解決に持っていける

というのが素晴らしい点であるといえる。欠点は時間がかかる事で、これは実際に発表する時間やその発表のための資料作りの時間である。解決方法としては、この発表のための資料はできるところまでテンプレート化してプロジェクトメンバーや研究会全体で共有すると時間の削減になるのではと思った。

4.3.2.1.4. 7月編

第1反復で出来た成果物を第2反復で改良するという作業に入った。私は説明文を人工音声で作ったものやゲーム内の音(爆弾の音とか)をCRI Audioで使えるように.wav形式から.csb形式に変換し、実際に聞いてみて調整する作業がメインであった。この作業はCRIAudioをβ版から正式版に変更し、CriAudioCraftという変換ソフトを利用した。当初β版で作成していたのだが、実装したプログラム正式版を利用しており、そのプログラム上ではならないという事があったためである。ミート外でも作業を効率化するためのコミュニケーションが必須であると感じた。

音関連で苦労したのは、日本語で人工音声を作ってくれるWebページを探すことであった。最初は自分の声で作っていたが、男の声だと耳障りではないように音の波をいじるソフトでいじくった際にもノイズがのりやすいため断念した。また、人工音声は基本的に滅茶苦茶高いソフトを利用するのが主流であり、無料でサンプルを作成してくれるWebページは、そのサンプルを保存させない(しにくい)ようにしてあるのが普通だからだ。今回は運良く見つかったからよかったが、あらかじめ必要だとわかっているので音声作り担当の私がそういう人脈作りもはやめにしておくべきであったと反省した。

4.3.3. まとめ

はじめての「学習・ゲーム作り」のプロジェクトであったが、ユードー社に伺うなど、非常に楽しく行うことができた。今更だが、何度も出てきたプロジェクトとは

・プロジェクト…有期性と独自性を持つ活動

と私たちの中では定義されている。

有期性があるのは当然だが､私たちは特に独自性という点において胸を張れるプロジェクトであったといえる。

5. 添付資料

5.1. プロジェクト定義書

次ページからは、プロジェクト定義書である。

5.2. スコープ記述書

次ページからは、スコープ記述書である。

5.3. マススケジュール

次ページからは、今期のプロジェクトのマススケジュールである。

5.4. UML・試験手順書

5.4.1. 全体分析

図 3.2.1.3.1.3.1-1 ユースケース図

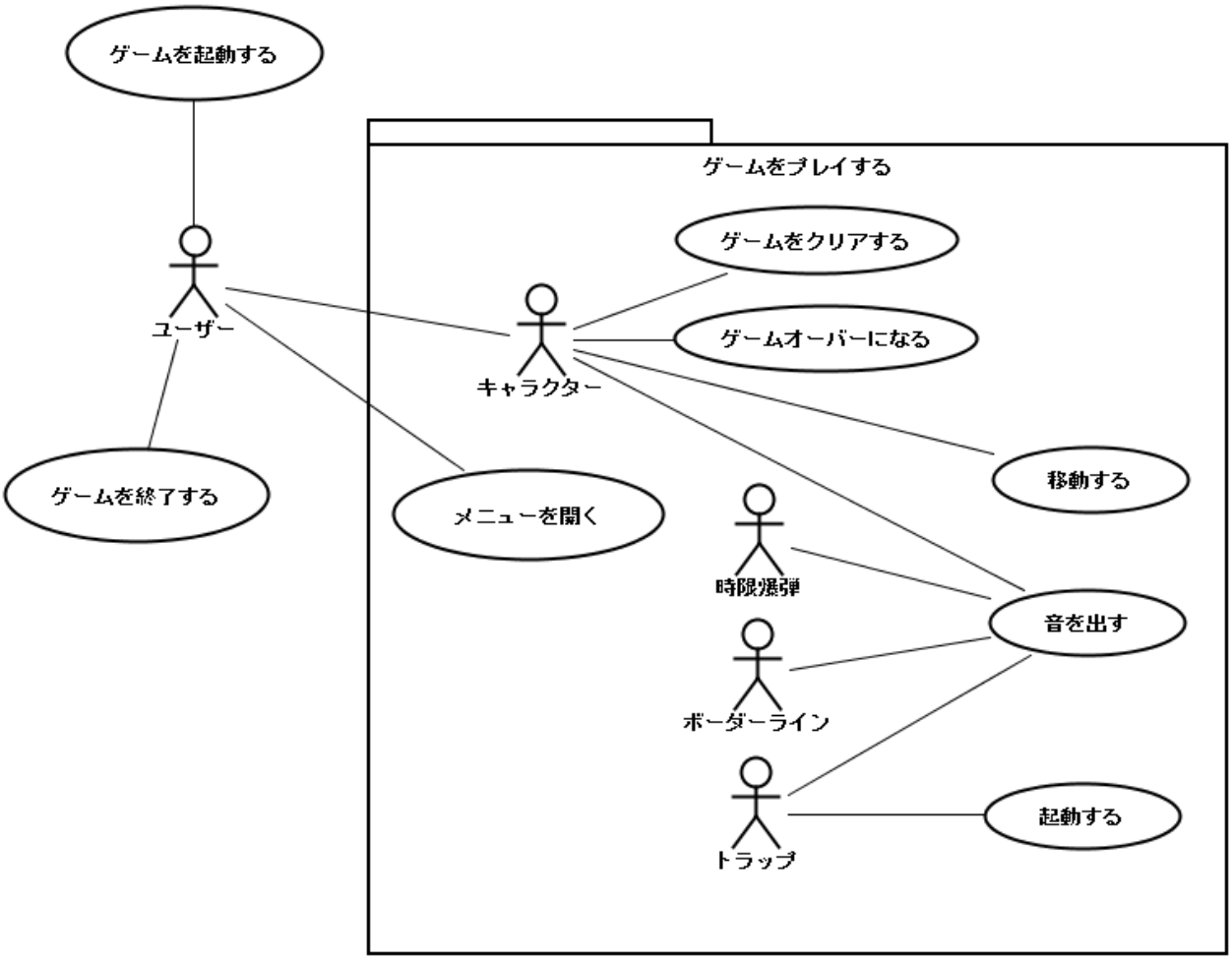


表 3.2.1.3.1.3.1-2 ユースケース記述

ユースケース番号	1
ユースケース名	ゲームを起動する
アクター	ユーザ
目的	ユーザはゲームを起動して遊びたい。
事前条件	4．ユーザはゲームを起動していない
基本系列	1　アクターは、アプリケーションを起動する 2　ゲームのシステムは立ち上がる
事後条件	ゲームのシステムは起動している
代替系列	基本系列 2 で、システムが既に立ち上がっている場合は警告をする
備考	無し
シナリオ	1　橋山はゲームがやりたいと思ったのでアプリケーションを起動させた 2　渡辺はゲームがやりたいと思ったのでアプリケーションを起動させようとしたが、最近痴呆気味のため既に起動させていたことを忘れていたため、既に起動済みであると警告をされた

ユースケース番号	2
ユースケース名	ゲームをプレイする
アクター	ユーザー
目的	ユーザーはゲームで遊びたい
事前条件	・ ゲームが起動して，タイトルへ移動している
基本系列	1　アクターは Enter を押してゲームを開始する 2　システムはゲームの説明をアナウンスする 3　システムはゲーム中の状態になる
事後条件	ゲームがプレイ中の状態になっている
代替系列	基本系列 2 で、ゲーム中の状態でない場合は何も起こさない
備考	・ゲームはテンキーとマウスを用いて遊ばれる
シナリオ	(7) 佐藤はゲームを起動したあとに操作音声が流れたので，Enter を押した．ゲームが開始され，「説明を飛ばす場合は Enter を押してください」という音声が流れた．その後，ストーリー紹介が始まったが面倒なので，Enter キーを押すとキャラクターの操作ができるようになった．

ユースケース番号	3
ユースケース名	ゲームを終了する
アクター	ユーザー
目的	ユーザーはゲームをやめたい。
事前条件	・ システムは起動している ・ システムはゲーム中の状態ではない
基本系列	1 アクターは、0 キーを押してゲームを終了しようとする 2 システムは、ゲームをやめるがいいかと聞く 3 アクターは、Enter を押してゲームを終了する 4 システムは終了する
事後条件	システムは終了している
代替系列	基本系列 3 で、再度 0 キーが押された場合は元の状態に戻る
備考	無し
シナリオ	佐藤はゲームに飽きたので、終了したいと思った。 おもむろに 0 キーを押すと、「ゲームを終了します。終了する場合は Enter を押してください。キャンセルしたい場合は 0 キーを押してください」と尋ねられた。 佐藤はゲームを終了したいと思ったので Enter を押し、ゲームを終了させた。

ユースケース番号	4
ユースケース名	ゲームをクリアする
アクター	キャラクター
目的	ゲームの目的を達成したことをプレイヤーに知らせる
事前条件	ゲームがプレイ中である
基本系列	1 アクターは、実行可能な動作を用いて時限爆弾に接触する 2 システムは、プレイヤーと時限爆弾が触れたか判定する。 触れていた場合、時限爆弾は解除されクリアとなる 3 システムは、触れたことをアナウンスし、クリアまでにかかった時間を知らせる。 4 システムはタイトルに戻る。
事後条件	ゲームはタイトル画面へ移動している
代替系列	2 で、触れていなかった場合はそのままゲームを続行する
備考	無し
シナリオ	渡辺は時限爆弾の音が聞こえたので，そちらへ移動した． 時限爆弾の音が大きくなったところでゲームクリアの音声が流れて ファンファーレが鳴った．3 分 40 秒でクリアしたことが流れて， タイトルへ戻ることが音声で伝えられた．その後すぐに，タイトル における操作音声が流れ始めた．

ユースケース番号	5
ユースケース名	ゲームオーバーになる
アクター	キャラクター
目的	ユーザーにゲームの目的が達成できなかったことを知らせる
事前条件	ゲームがプレイ中である。
基本系列	1　アクターは、 ・ 即死トラップにひっかかる ・ ボーダーラインを超える ・ 制限時間がなくなる のいずれかの条件を満たす 2　システムは、ゲームオーバーになったことをアナウンスする 3　システムは、ゲームをタイトル画面に戻す
事後条件	ゲームはタイトル画面へ移動している
代替系列	無し
備考	無し
シナリオ	橋山は左から時限爆弾の音がしていると思い左へ向かった． 途中でボーダーラインの警告音が鳴ったが，時限爆弾があると 信じて左へ向かい続けると警告音とともにゲームオーバーになった． タイトルへ戻ることが音声で伝えられて，その後操作音声が 流れたのでタイトルへ戻ったことが分かった．

ユースケース番号	6
ユースケース名	メニューを開く
アクター	ユーザー
目的	操作説明の確認や，ゲームの途中終了を行う
事前条件	ゲームがプレイ中である
基本系列	1　アクターは、Enter を押す 2　システムは、「チュートリアルの場合は 1 を，タイトルに戻る場合は 2 を， メニューを閉じる場合は Enter を押してください」という音声を流す 3　アクターは、思うままにキーを押す 4　システムは、押されたキーに応じて反応を起こす
事後条件	メニュー項目ごとのアクションが行われ
代替系列	無し
備考	・ 1 の場合、システムはゲーム開始時に流れたチュートリアルを流す。 ・ 2 の場合、システムはゲームをタイトル画面に移動する ・ Enter の場合、システムはメニューを閉じ、ゲームを続行する
シナリオ	佐藤はワープトラップと即死トラップの音の違いを忘れてしまい，Esc を押してメニューを開いた．操作音声が流れたので，1 を押してチュートリアルを選択して，その中でトラップの音の違いを覚えた．その後メニューに戻り，3 を押してメニューを閉じた．

ユースケース番号	7
ユースケース名	移動する
アクター	キャラクター
目的	トラップを避け、時限爆弾にたどり着きたい
事前条件	ゲーム中である
基本系列	1　ユーザーは、テンキーやマウスで移動したい方向を入力する。 2　システムは、入力通りにプレイヤーを移動させる
事後条件	キャラクターが移動している
代替系列	無し
備考	・　移動時には足音が鳴る ・　移動はテンキーで行い、それぞれ 5 が前、2 が後ろ、1 が左、3 が右に対応している。 ・　方向はマウスで変えられ、マウスを動かすとキャラクターの向きをその方向へ回すことができる
シナリオ	渡辺は正面からトラップの音が聞こえてきたので，キャラクターを迂回させようと←を押した．すると，キャラクターの足音とともにトラップの音が右方向へずれていった．

ユースケース番号	8
ユースケース名	起動する
アクター	トラップ
目的	キャラクターのゲームクリアを邪魔する
事前条件	・　ゲームがプレイ中である ・　対象のトラップが起動前である
基本系列	1　アクターは、キャラクターがアクターの座標上に移動すると起動する 2　アクターは警告音を出し、トラップの効果を発動させる
事後条件	トラップが起動して，トラップごとのアクションが行われる
代替系列	音が聞ける状態でない場合（罠にはまっている等)、音は聞こえない
備考	・　トラップは 2 種類ある ・　ワープトラップ：キャラクターをフィールド内のどこかに強制的に移動する．場所は毎回ランダムで決定する． ・　即死トラップ：ただちにゲームオーバーになる ・　ワープトラップが起動した場合はトラップは起動済みとなり，以降同じトラップにひっかかることはない．また，起動済みのトラップは音を発しなくなる．
シナリオ	堀はトラップの音と時限爆弾の音が同じ正面方向から聞こえてくるので，一か八かそちらへ向かってみた．トラップの音が若干大きかったように聞こえたのはトラップが時限爆弾の真正面にあったからで，トラップにひっかかってしまった．警告音声とともにワープトラップが起動して，時限爆弾の音はまったく聞こえなくなってしまった．

ユースケース番号	9
ユースケース名	音を出す
アクター	キャラクター，時限爆弾，トラップ，ボーダーライン
目的	ユーザーに存在を知らせる
事前条件	・　全体：ゲームがプレイ中である ・　キャラクター：移動中である ・　トラップ：起動前である ・　時限爆弾・トラップ・ボーダーライン：一定距離内に存在する
基本系列	1　アクターは音を鳴らす必要がある場合、音を発する
事後条件	アクタごとにそれぞれの音が鳴る
代替系列	無し
備考	・　キャラクター：移動すると足音がなる ・　時限爆弾：毎秒，時計が動く音が鳴る（残り時間が近づくと，カウントダウンを始める） ・　トラップ：断続的に電子音が鳴る ・　ボーダーライン：警告音が鳴る ・　時限爆弾とトラップとボーダーラインは，キャラクターとの距離に反比例して発する音が大きくなる．
シナリオ	中川はゲームを始めたものの，どこに時限爆弾があるか分からず， とりあえずうろうろしていた．すると，足音に混じって右から 時計が動くような音が聞こえてきたので，そちらへ移動すると 時限爆弾を発見することができた．

図 3.2.1.3.3.3-1 オブジェクト図

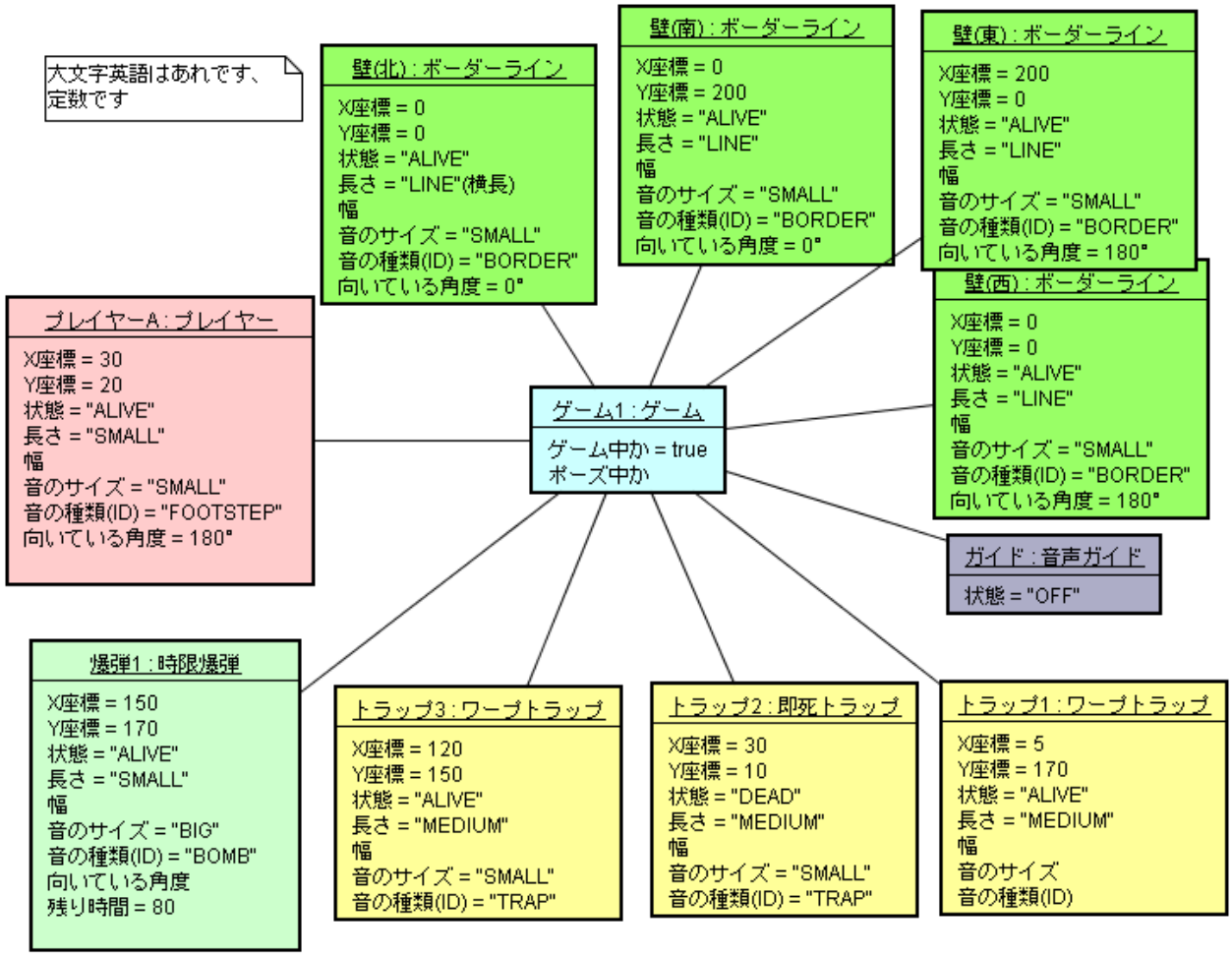


図 3.2.1.3.3.3-2 クラス図

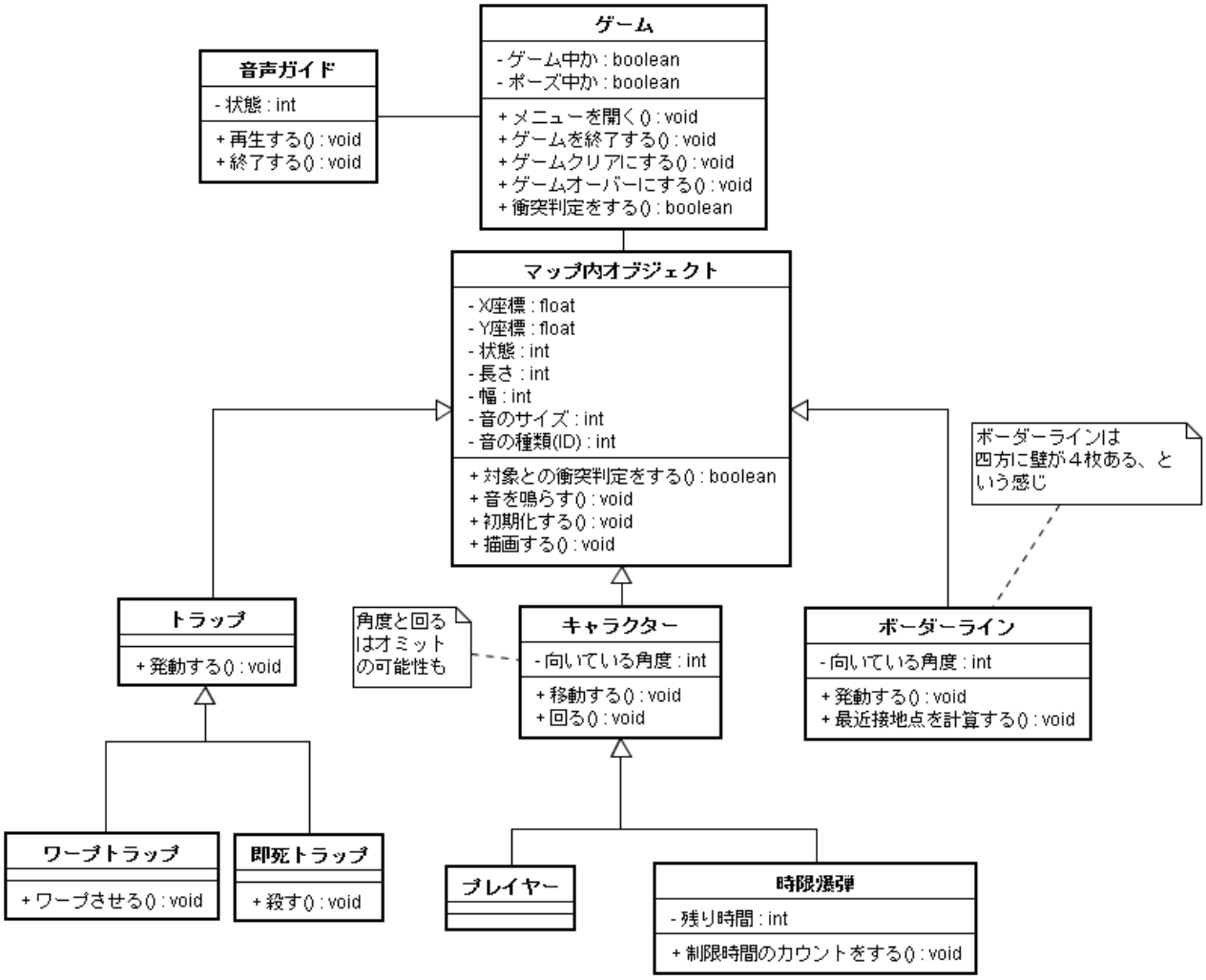


図 3.2.1.3.3.3-3 シーケンス図 1 処理の流れ

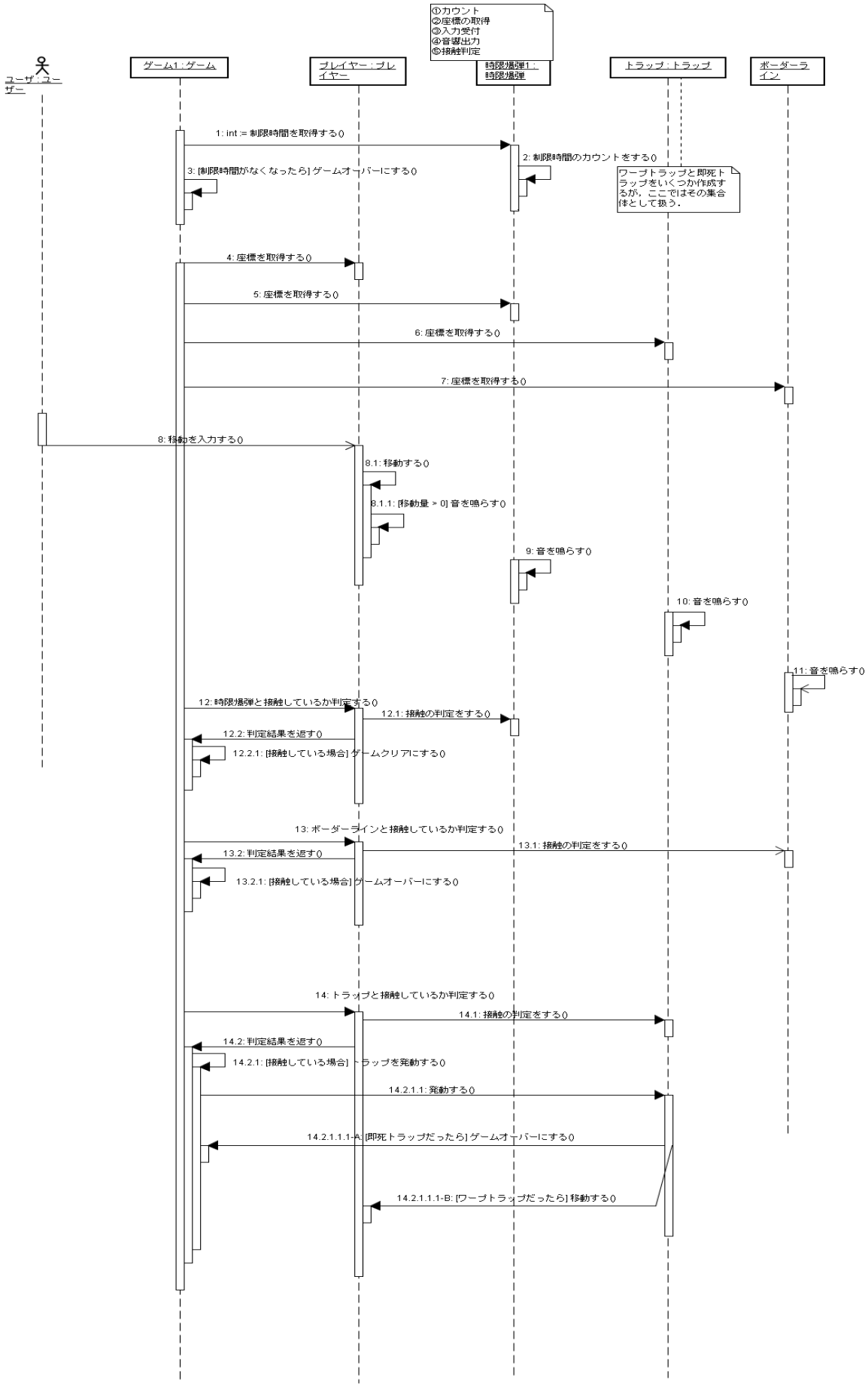


図 3.2.1.3.3.3-4 シーケンス図 2 ゲームを開始する

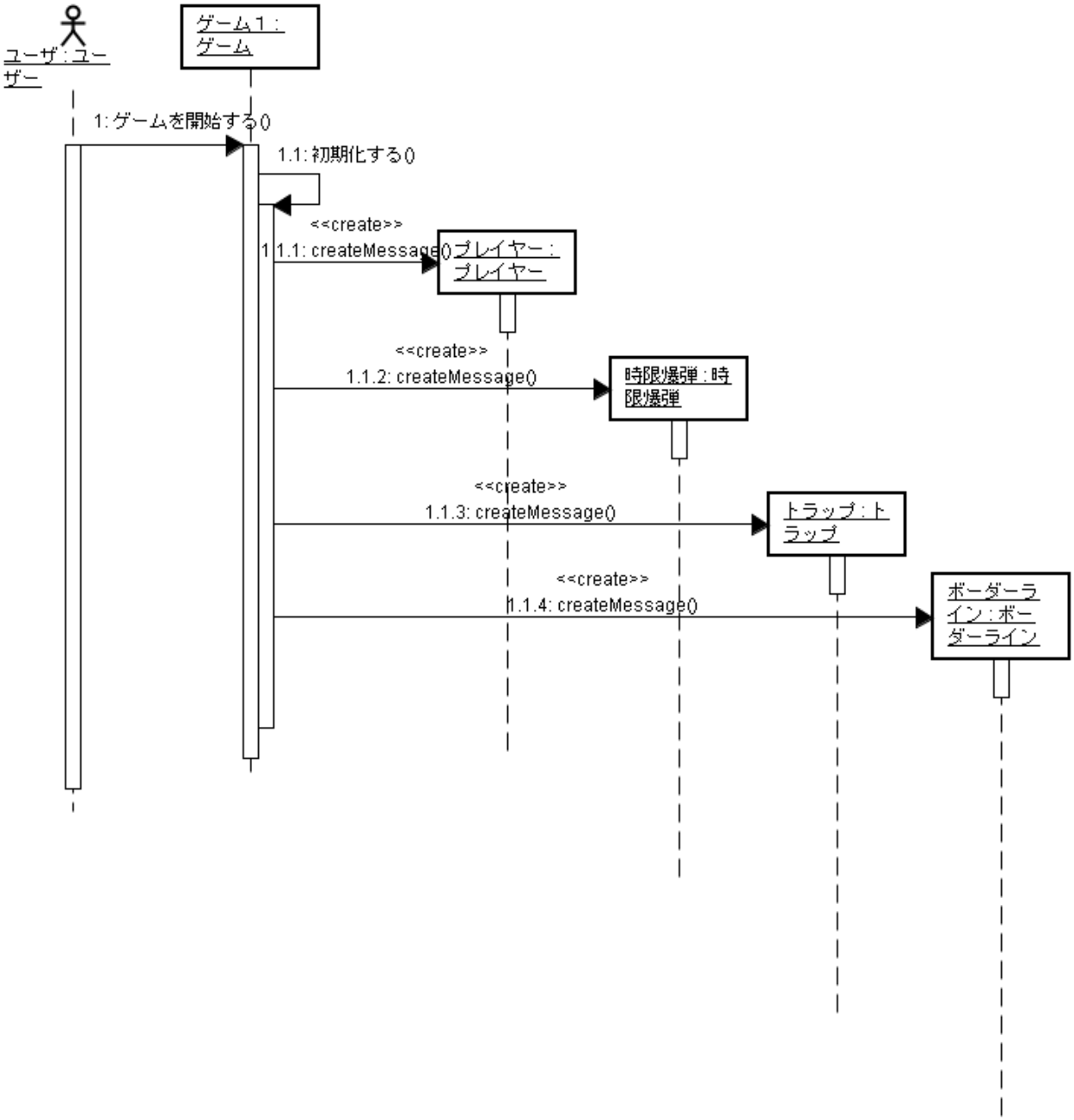


図 3.2.1.3.3.3-5 シーケンス図 3 ゲームをクリアする

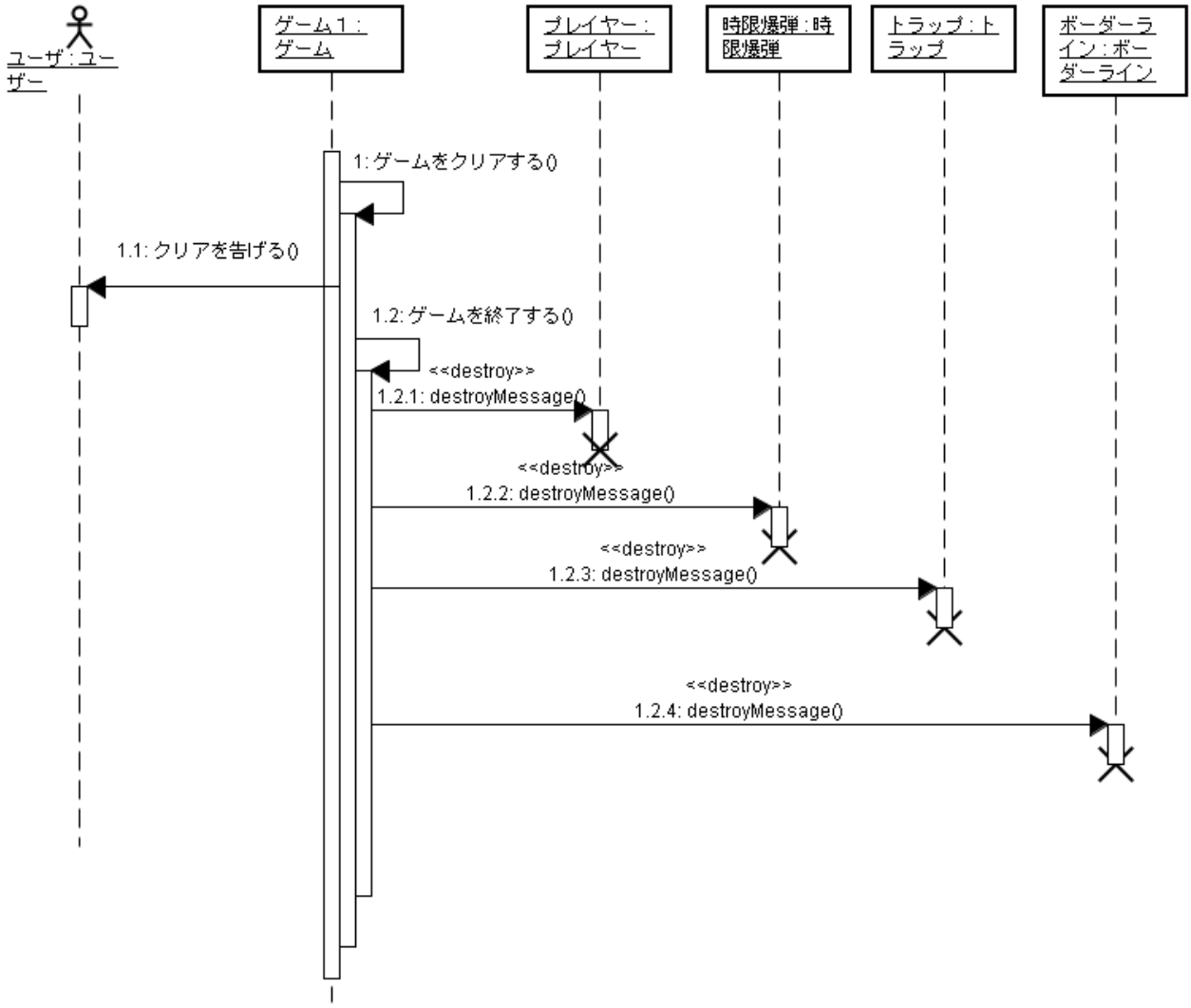
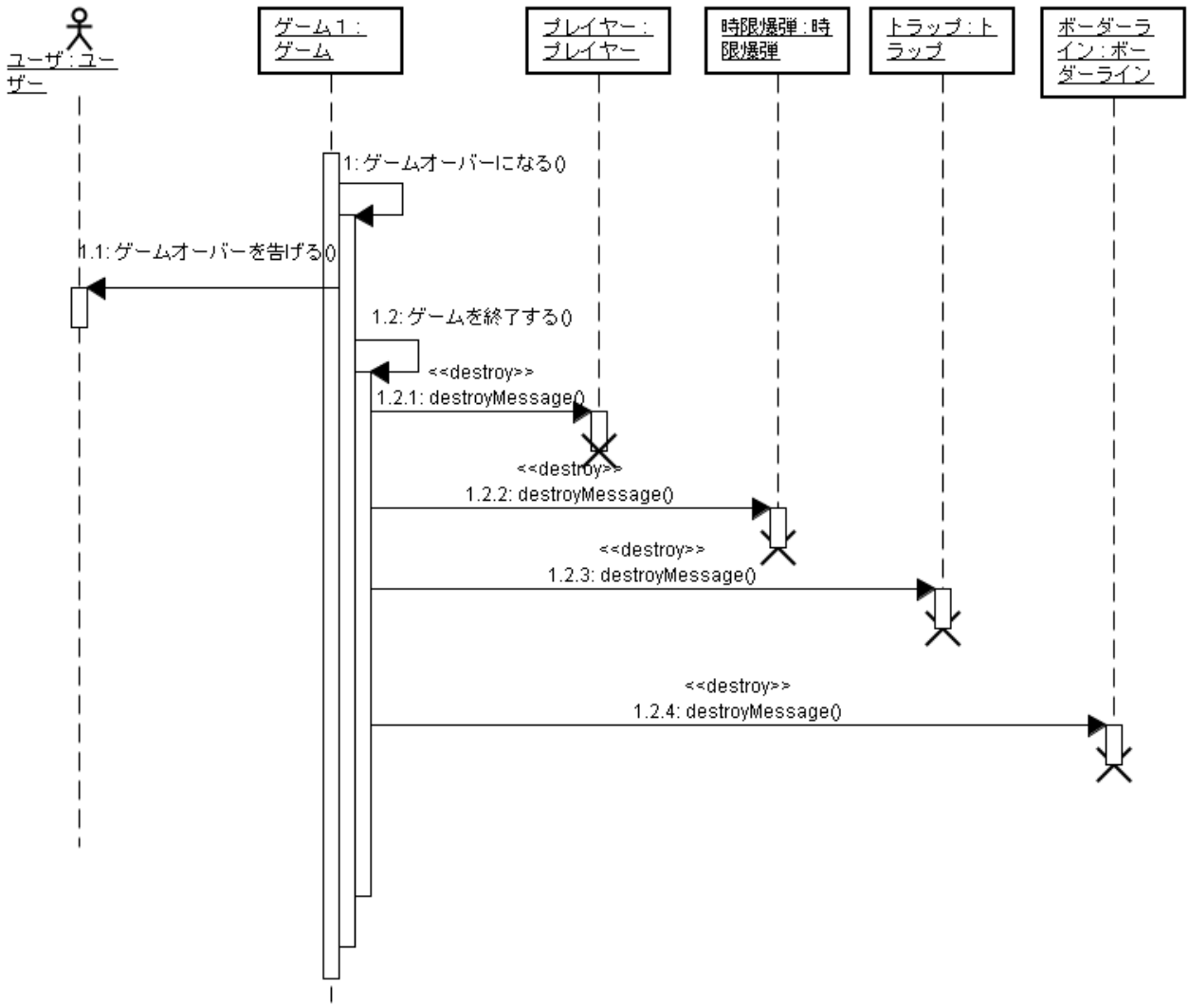


図 3.2.1.3.3.3-6 シーケンス図 4 ゲームオーバーになる



5.4.2. 反復作業 1 回目

図 3.2.2.3.2.3-1 ユースケース図

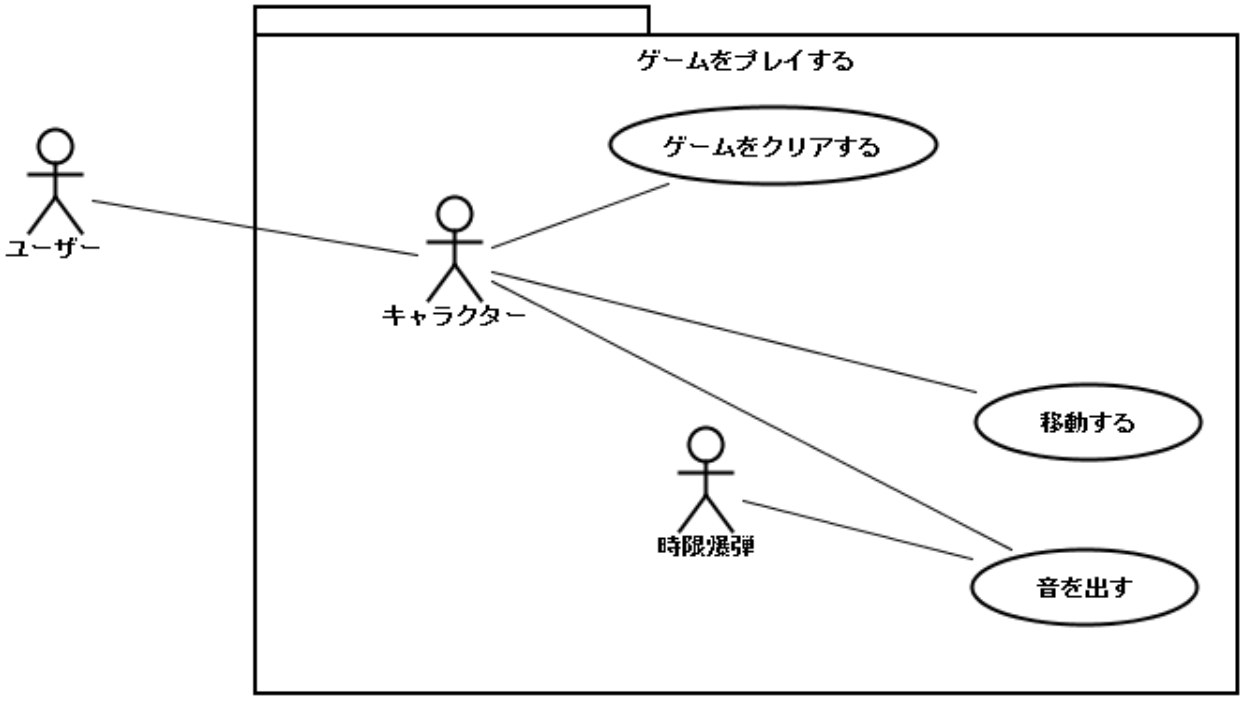
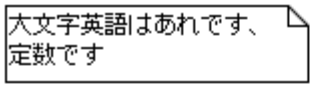


図 3.2.2.3.3.3-1 オブジェクト図



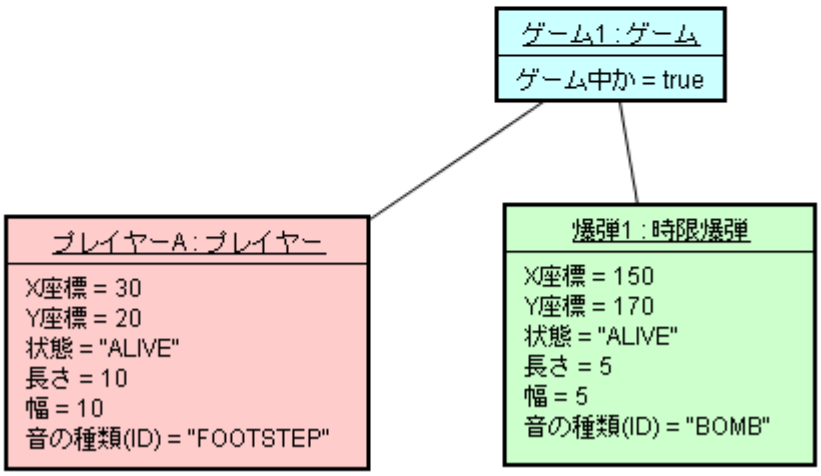
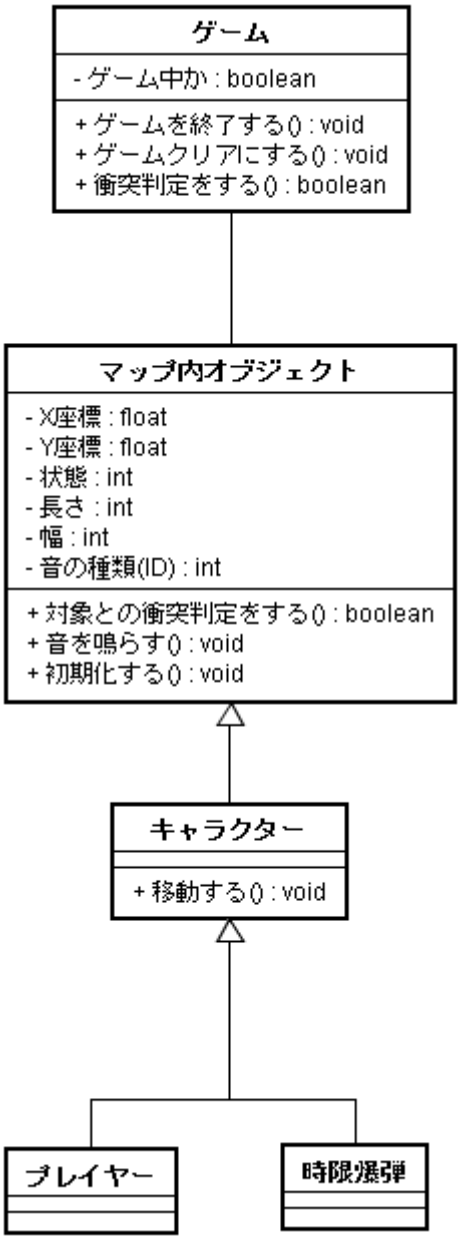


図 3.2.2.3.3.3-2 クラス図



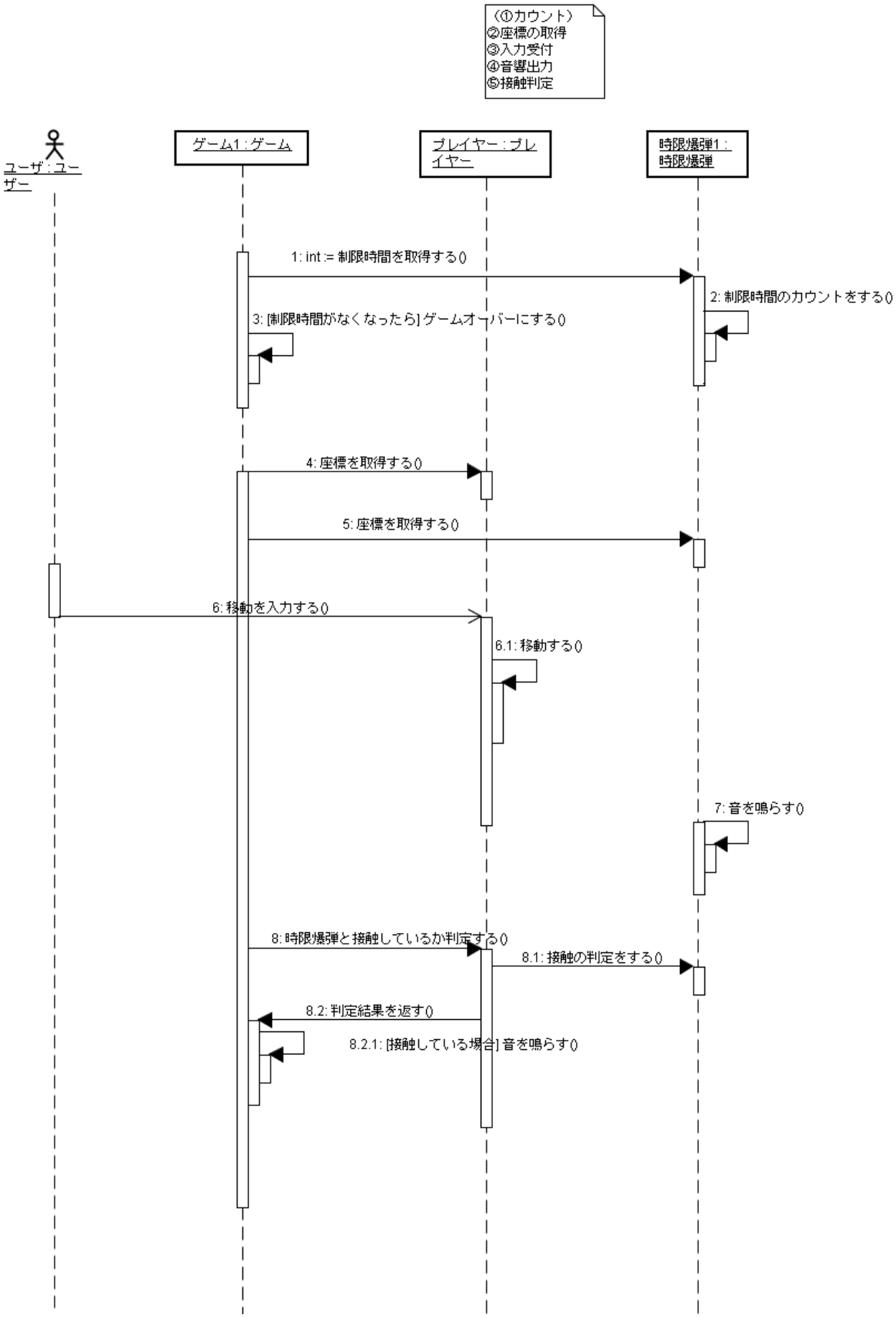


図 3.2.2.3.3.3. -3 シーケンス図 1　処理の流れ

図 3.2.2.3.3.3 -4 シーケンス図 2　ゲームを開始する

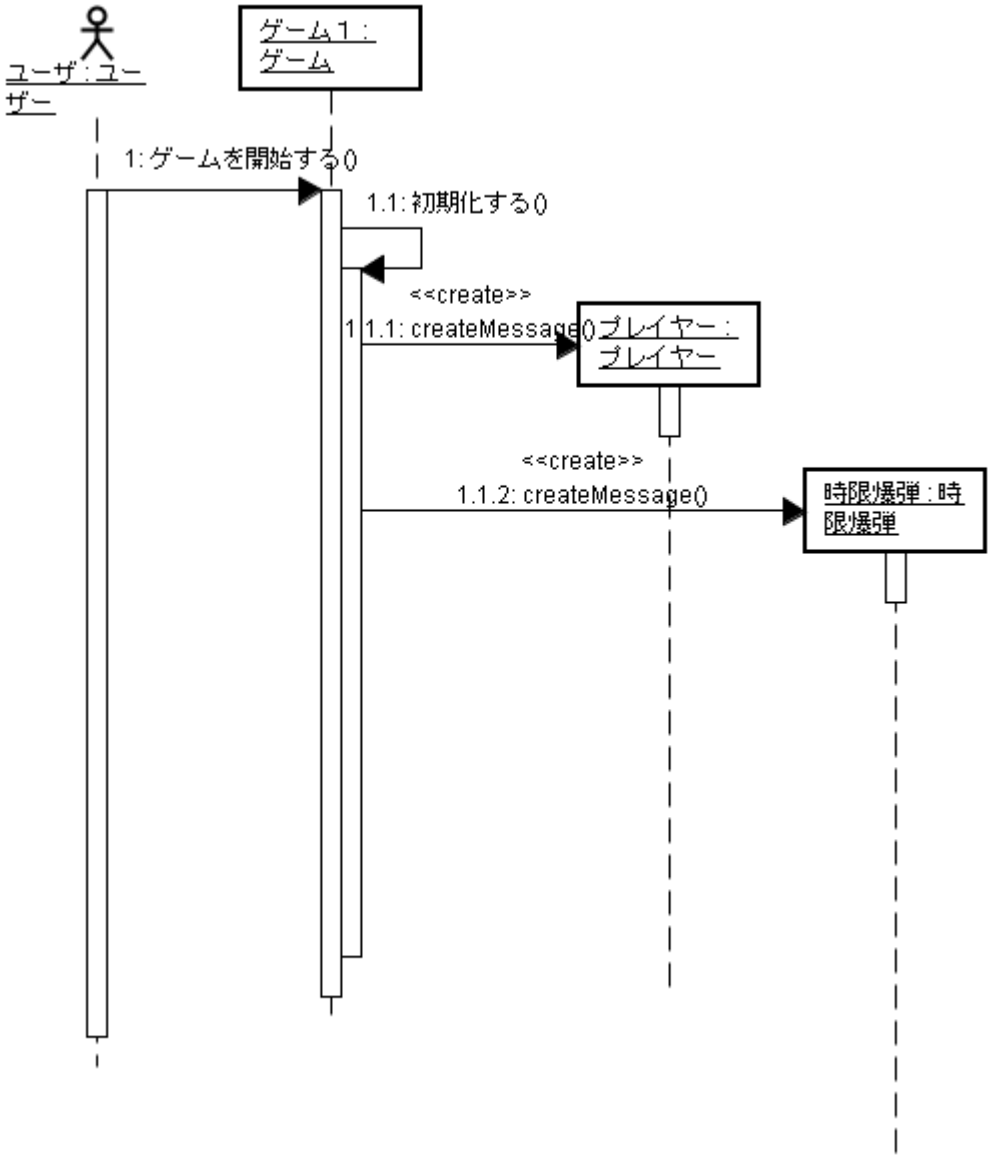
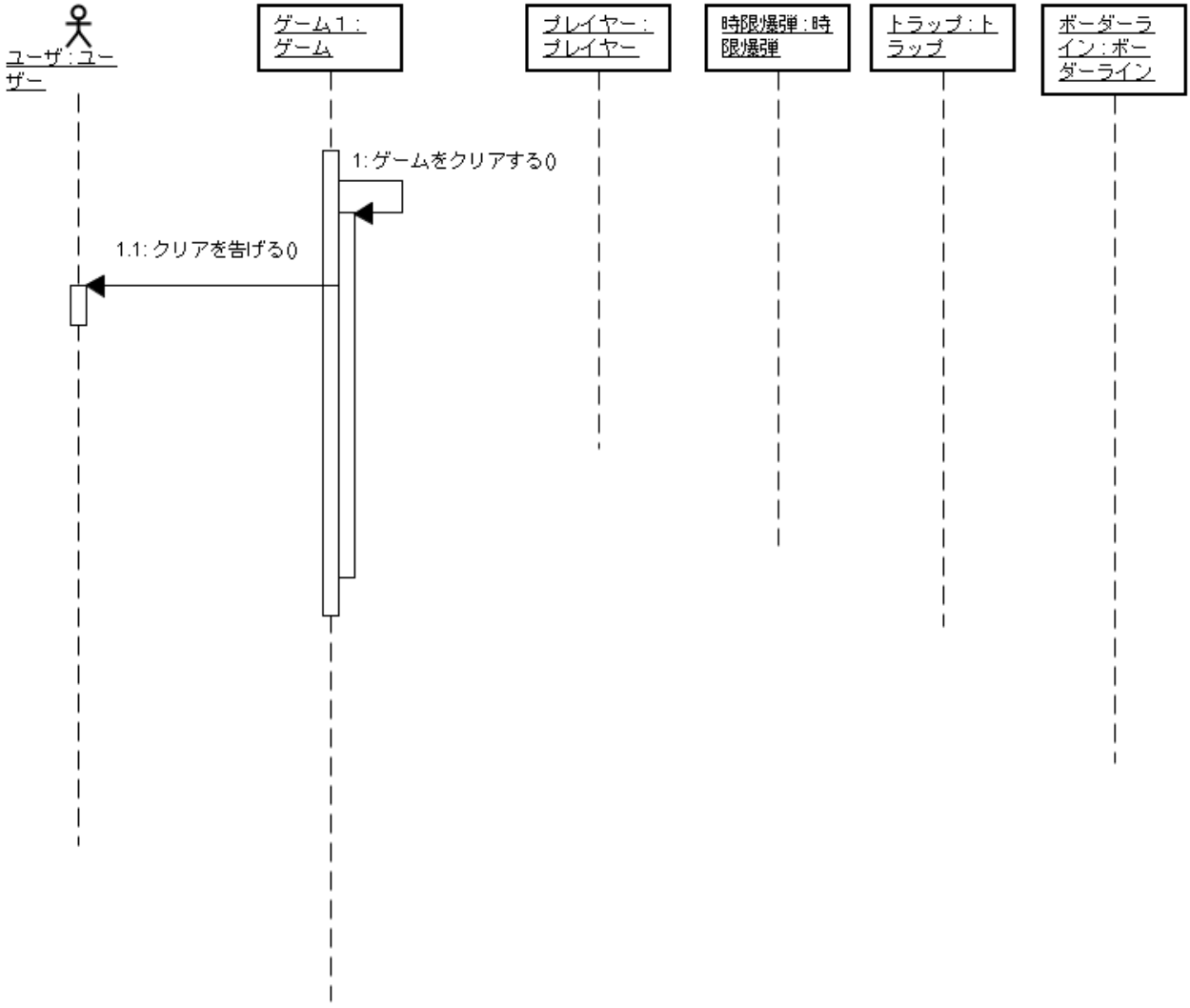


図 3.2.2.3.3.3 -5 シーケンス図 3　ゲームをクリアする



No	システム仕様				備考
	分類		要求仕様との対応	要求事項	
1	全般	システム概要		・映像の無いゲームであること。 ・健常者と目の不自由な者の両方が同様に楽しむことのできるゲームであること。	
2		制約条件		・画面には何も表示されないこと。(ただし、製作中の段階に限ってデバッグ中の画面を表示してもよい) ・音だけでゲームの状態が理解できること。 ・サラウンド音響を用いて、ゲーム中の各オブジェクトの存在地点を音だけで把握できること。	
3		インタフェース	マニュアル「操作方法」	・入力はマウスとテンキーによって行われる。 ・ゲーム中のキャラクターの操作は、キャラクターの移動はテンキーで行われること。 ・状態遷移の選択(ゲームの開始やメニューの表示他)はテンキーの入力によって行われること。 ・詳細な入力対応はマニュアル及び各操作時のユースケース記述を参考にすること。	
4	ゲーム制御	システムを起動する	ユースケース番号1	・システムが起動前であること。 ・システムのアイコンをダブルクリックするとウィンドウが開き、システムは起動すること。 ・プレイヤーがシステムを起動すると、ゲームは起動した状態となること。この際、ゲームの状態はゲーム前(タイトル画面)とすること。 ・プレイヤーがシステムを起動すると、ゲームは起動した状態となること。この際、ゲームの状態はゲーム中(タイトル画面)とすること。	
5		ゲームを開始する	ユースケース番号2	・ゲームの状態がゲーム前(タイトル画面)であること。 ・プレイヤーが入力(Enterキー)を行った場合、「ゲームを開始します」という音声と共にゲームの状態をゲーム中に遷移させること。 ・開始の際は、全てのオブジェクトの位置・状態を初期化すること。 ・ゲームを起動させた際、ゲームの状態は中断中(メニュー表示中)であり、チュートリアルを流すこと。	
6		システムを終了する	ユースケース番号3	・ゲームの状態がゲーム前(タイトル画面)であること。 ・プレイヤーが入力(0キー)を行った場合、「ゲームを終了します」という音声と共にシステムを終了した状態とすること。	
7		メニューを表示する	ユースケース番号6	・ゲームの状態がゲーム中であること。 ・プレイヤーが入力(Enterキー)を行った場合、ゲームを中断中にし、メニューを表示させること。	
8		メニューを閉じる	ユースケース番号6	・ゲームの状態がゲーム中で、中断中(メニュー表示中)であること。 ・プレイヤーが入力(Enterキー)を行った場合、「メニューを閉じます」という音声と共に中断を解くこと。	
9		チュートリアルを流す	ユースケース番号6	・ゲームの状態がゲーム中で、中断中(メニュー表示中)であること。 ・プレイヤーが入力(1キー)を行った場合、ゲームの説明をするチュートリアルを流すこと。なお、ゲームを開始した際は自動的にチュートリアルを流す段階とすること。 ・チュートリアル中にプレイヤーが入力(Enterキー)を行うと、チュートリアルは終了され、中断は解除されること。	
10		ゲームを終了する	ユースケース番号6	・ゲームの状態がゲーム中で、中断中(メニュー表示中)であること。 ・プレイヤーが入力(2キー)を行った場合、「ゲームを終了します」という音声と共にゲームの状態をゲーム前(タイトル画面)に遷移させること。 ・Ctrl+Alt+Del操作を行い、「タスクの終了」を選択するとゲームが終了すること	
11		ゲームクリアにする	ユースケース番号4、マニュアル「ゲームの流	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと ・プレイヤーがゲームクリアの条件を満たした場合、「ゲームクリアです」という音声と共にゲームの状態をゲーム前(タイトル画面)に遷移させること。 ・プレイヤーがゲームクリアの条件を満たした場合、効果音を流すこと。	
12		ゲームオーバーにする	ユースケース番号5、マニュアル「ゲームの流	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと ・プレイヤーがゲームオーバーの条件を満たした場合、「ゲームオーバーです」という音声と共にゲームの状態をゲーム前(タイトル画面)に遷移させること。	
13		接触判定をする	ユースケース番号4	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと ・キャラクターが移動を行い、座標が変わったこと。 ・キャラクターと他のオブジェクトが接触したかどうか、両方の座標とサイズを取得した上で計算を行ってを判定をすること(ここで言うオブジェクトとは時限爆弾、トラップ、ボーダーラインを指す)。 ・判定の結果、接触が有効だと判断された場合、各オブジェクトに用意されたリアクションを起こさせること。	
14	キャラクター	音響出力する	ユースケース番号9	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと ・キャラクターが移動を行い、一定距離動く度にキャラクターの座標から「コツ、コツ」という足音を鳴らすこと。	
15		移動する	ユースケース番号7、マニュアル「操作方法」	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと ・プレイヤーキー/マウスを用いて入力を行い、その入力に応じてキャラクターは動作すること。具体的な操作と行動の対応はユースケース・マニュアルを参照。	
16		描画する	―	・ゲームの状態がゲーム中であること。 ・キャラクターが存在する地点を確認するため、ウィンドウに位置を描画すること。なお、デバッグ用の機能である。	
17	時限爆弾	音響出力	ユースケース番号9	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと ・時限爆弾が有効であること(解除もしくは爆破していない状態であること)。 ・時限爆弾が存在する地点から、キャラクターの存在する地点までの距離と角度(座標)を計算し、その地点から恒久的に「カチ、カチ」という音を発すること。	
18		カウントする	マニュアル「ゲームの流れ」	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと ・時限爆弾が有効であること(解除もしくは爆破していない状態であること)。 ・設定された時間から経過した時間を刻むこと。	
19		爆発する	ユースケース番号5	・ゲームの状態がゲーム中であること。 ・時限爆弾が有効であること(解除もしくは爆破していない状態であること)。 ・経過時間が設定した時間を超過した場合、爆発の効果音と共にゲームの状態をゲームオーバーとすること。(No12参照)	
20		解除する	ユースケース番号4	・ゲームの状態がゲーム中であること。 ・時限爆弾が有効であること(解除もしくは爆破していない状態であること)。 ・設定した時間内に時限爆弾とキャラクターの接触判定が有効だと判断された場合、解除音と共にゲームをクリアすること。(No11参照)	
21		描画する	―	・ゲームの状態がゲーム中であること。 ・時限爆弾が有効であること(解除もしくは爆破していない状態であること)。 ・時限爆弾が存在する地点を確認するため、ウィンドウに位置を描画すること。なお、デバッグ用の機能である。	
22	トラップ	音響出力する	ユースケース番号9	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと。 ・トラップが有効であること(起動前の状態であること)。 ・トラップが存在する地点から、キャラクターの存在する地点までの距離と角度(座標)を計算し、その地点から恒久的に「ピー」という警告音を発すること。	
23		起動する	ユースケース番号8	・ゲームの状態がゲーム中であること。 ・トラップが有効であること(起動前の状態であること)。 ・トラップとキャラクターの接触判定が有効だと判断された場合、トラップ起動音と共に下記のトラップの効果を発動させること。 ・トラップがワープトラップであった場合、キャラクターをフィールド内のランダムな地点に移動させること。 ・トラップが即死トラップであった場合、ゲームをゲームオーバーにすること。(No12参照)	
24		描画する	―	・ゲームの状態がゲーム中であること。 ・トラップが有効であること(解除もしくは爆破していない状態であること)。 ・トラップが存在する地点を確認するため、ウィンドウに位置を描画すること。なお、デバッグ用の機能である。	
25		音響出力する	ユースケース番号9	・ゲームの状態がゲーム中で、中断中(メニュー表示中)ではないこと。 ・部屋の四辺の端(ボーダーラインが存在する場所)から、キャラクターの存在する地点までの最短距離と角度を計算し、その地点から恒久的に警告音を発すること。	

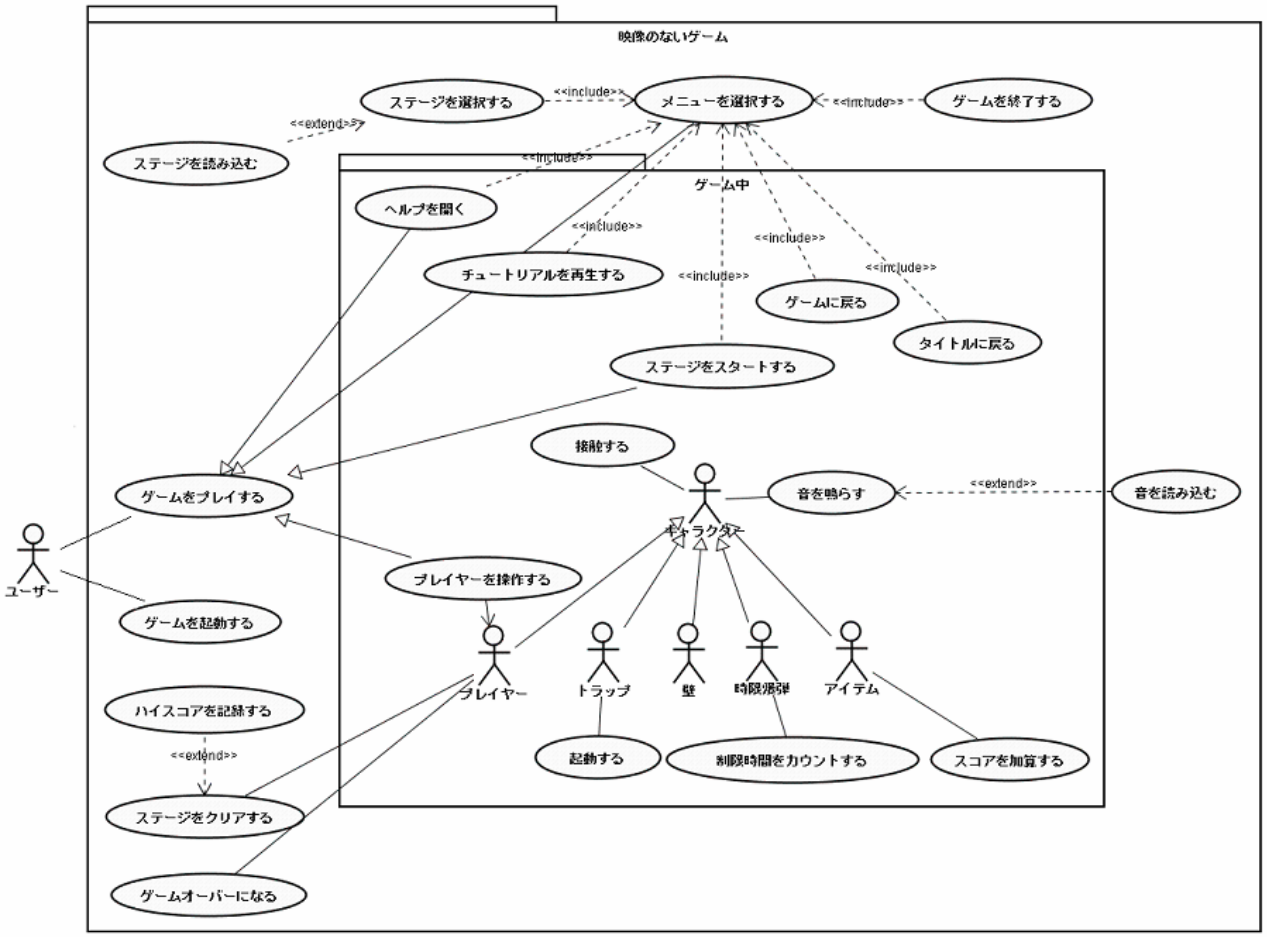
表 3.2.2.3.4.3 -1 システム仕様一覧

表 3.2.2.3.6.3-1　試験手順書・成績書

No	試験項目			試験手順	判定 基準	試験 結果	判定	備考	試験 担当
	大区分 （システム仕様）	中区分	小区分						
1	ゲーム制御	システムを起動する		SoundOnlyFrameworkexeを開くと、 システムを起動し、ウィンドウが開くことを確認する。	起動すればウィンドウが開くかどうか	ウィンドウが自動でひらいた	○		渡辺
2		システムを終了する		ゲームが起動してある状態で、 Ctrl+Alt+Del操作を行い、「タスクの終了」を選択すると、 システムは終了し、ウィンドウが閉じることを確認する。	「タスクの終了」でシステムを終了してウィンドウが閉じるかどうか	ウィンドウが閉じた	○	ウィンドウ右上の×を押下することでも終了できる	渡辺
3	キャラクター	移動する	前に移動	ゲームが起動してある状態で、 上キーを押すと前方(デバッグ画面上では上方) に移動するか確認する。	デバッグ画面上で上に動くか確認できること	上キーを押すと上に動いた	○	移動する量が決まっており、そこまで細かい動きはできない(以下3つも同様)	渡辺
4			右に移動	ゲームが起動してある状態で、 右キーを押すと右方向(デバッグ画面上でも右方) に移動するか確認する。	デバッグ画面上で右に動くか確認できること	右キーを押すと右に動いた	○		渡辺
5			左に移動	ゲームが起動してある状態で、 左キーを押すと左方向(デバッグ画面上でも左方) に移動するか確認する。	デバッグ画面上で左に動くか確認できること	左キーを押すと左に動いた	○		渡辺
6			後ろに移動	ゲームが起動してある状態で、 下キーを押すと後ろ(デバッグ画面上では下方) に移動するか確認する。	デバッグ画面上で下に動くか確認できること	下キーを押すと下に動いた	○		渡辺
7		描画する		ゲームが起動してある状態のとき、 キャラクターが位置する場所に四角形が描画されているか確認する	デバッグ画面上でキャラクターの位置を把握できるものが描画されていること	四角形ではなく横線だが描画された	△		渡辺
8	時限爆弾	音響出力	音の出力	ゲームが起動してある状態のとき、 時限爆弾が絶えず音を発しているか確認する。	起動している間ヘッドフォンから音が聞こえていること	間隔をあけてではあるが常時音が鳴った	○		渡辺
9			音の変異	ゲームが起動してある状態のとき、 プレイヤーが座標を変えるごとに、 時限爆弾から出ている音も変わっているか確認する。					渡辺
10			音の方向	ゲームが起動してある状態のとき、 時限爆弾から出てくる音によって 時限爆弾が位置する場所が分かるか確認する。 例えば、爆弾が左にある時は左から音が出て、 左前方にある場合は左前方から音が出ているか確認する。	デバッグ画面上でキャラクターが爆弾の周りを動いて8方向から音が聞こえていること	5.1chサラウンドヘッドフォンを使用すると8方向から聞こえた	○	プレイヤーが爆弾に重なっている状態だと左から聞こえる　正面だと音が小さく聞こえる	渡辺
11			音の減衰処理	ゲームが起動してある状態のとき、 時限爆弾が位置する距離から音が出ているか確認する。 例えば、近づけば近づくほど音が大きくなり、 遠ざかれば音が小さくなる、というようになっているか確認する。	デバッグ画面上でキャラクターと爆弾の距離を変化させて音が変化していること	デバッグ画面上で爆弾とキャラクターの位置を離していくと音が小さくなった	○	ある程度距離が離れると一切聞こえなくなる	渡辺
12		描画する		ゲームが起動してある状態のとき、 時限爆弾が位置する場所に四角形が描画されているか確認する	デバッグ画面で爆弾の位置を把握できる何かが描画されていること	四角形ではなく横線だが描画された	△		渡辺

5.4.1.1. 反復作業第二段階のユースケース図

図 3.2.4.3.2.3.-1 ユースケース図



5.4.1.2. 反復作業第 2 回目分析のクラス図・状態遷移図

図 3.2.4.3.3.2.3.-1 クラス図

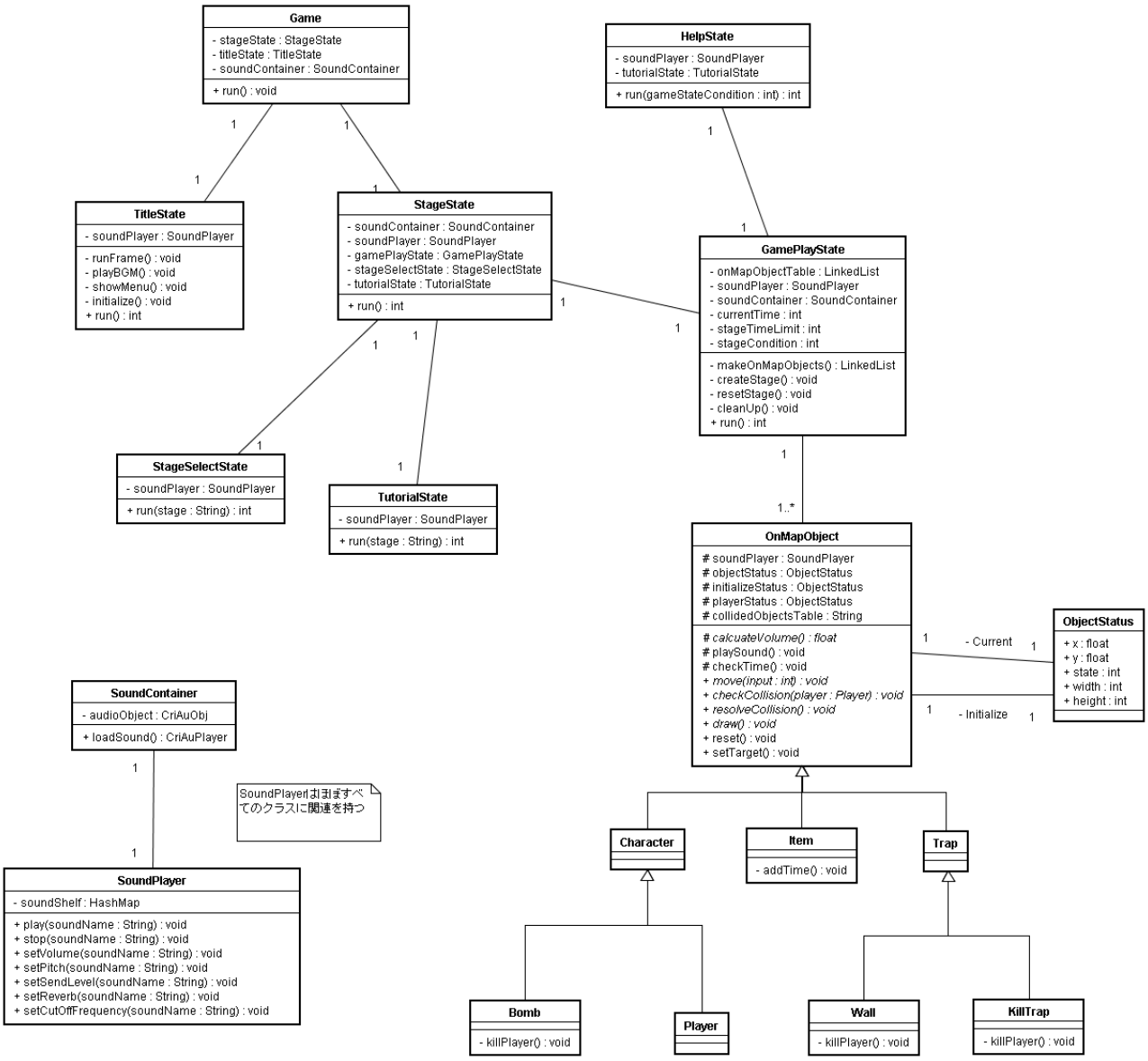
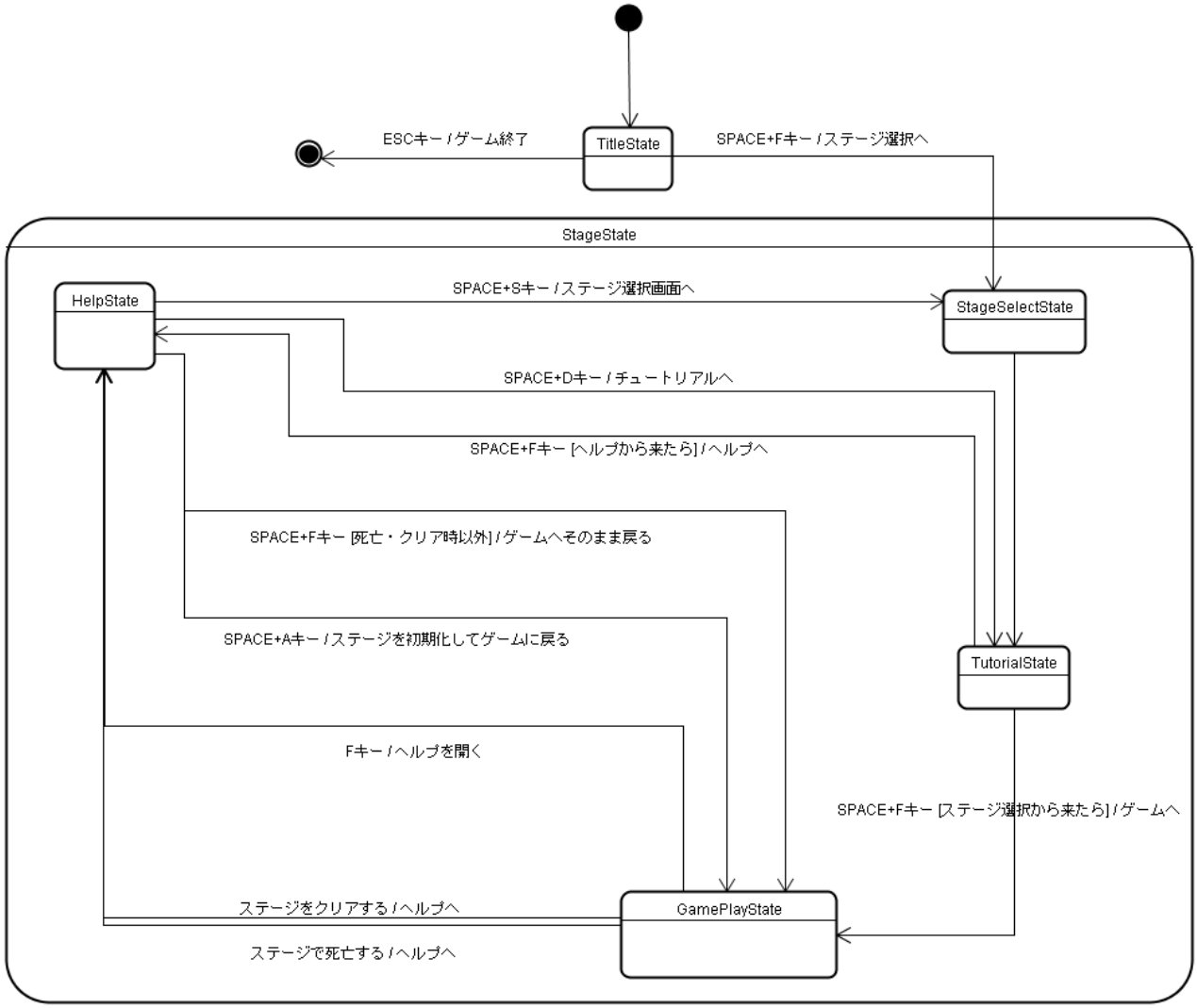


図 3.2.4.3.3.2.3.-2　反復作業第 2 回目分析の状態遷移図



5.5. ユーザーマニュアル

■ ***ストーリー***

時は 2XXX 年、惑星間での交流が発展している時代。
人類は地球外に文明に触れ、様々な技術を手に入れ、生活も豊かになった。

しかし、いいことばかりではなかった。

2XXX 年 ι 月 308 日、一つのニュースが地球に伝えられた。予てから地球を敵対視する火星のテロリスト達が月を荒らしつくし、更には月内部の空洞に大量の罠と超強力な時限爆弾を仕掛けたという。

月内部は、電気系統が破壊されたため真っ暗だ。その上、超電磁波の影響で灯りを灯すことができない。爆弾処理チームは、センサーの音だけを頼りに爆弾を解除することになる。無謀かも知れないが、失敗すれば地球は滅亡だ。なんとかしなければならない。

なんで懐中電灯がダメでセンサーが大丈夫なのかは未来の技術者に聞いてください。

■ ***ゲーム目的***

あなたは爆弾処理班として爆弾を解除することを目的とします。
フィールド上にある罠を避けつつ、時間内に音だけを頼りに爆弾を探してください。
無事に爆弾まで辿りつければ爆弾は解除できます。

■ ***ゲームの流れ***

☆ まず、ゲームを起動すると、「さうんど おんりぃ」というアナウンスが流れます。
☆ ステージ選択画面では、ステージナンバーを再生します。SPACE キーを押しながら A キーや S キーで好きなステージを選択してください。
☆ ゲームが始まると、刻一刻と制限時間が近づきます。落ち着きたい場合や、チュートリアルを確かめたい場合、ゲームを中断したい場合は F キーを押すとメニューが開きます。メニューを開いている間は時間が流れないので、じっくり英気を養ってください。
☆ カチッカチッと冷酷に時間を刻んでいる爆弾に辿り着くとクリアですが、制限時間内にたどりつけなかったり、トラップや壁に接触した場合はゲームオーバーとなります。

■ ***操作方法***

- **タイトル画面**

  Space を押しながら F を押して離す：ステージ選択画面へ

  Esc：ゲーム終了

  ↓

- **ステージ選択画面**

  Space を押しながら S を押して離す：次のステージナンバーを選択

  Space を押しながら A を押して離す：前のステージナンバーを選択

  Space を押しながら F を押して離す：チュートリアルへ

  ↓

- **チュートリアル画面**

  F を押して離す：チュートリアルを流す

  Space を押しながら F を押して離す：ゲーム開始

  ↓

- **ゲーム中**

  F ：ヘルプを開く

  ↑：前に進む

  ↓：後ろに進む

  ←：左に進む

  →：右に進む

- **ヘルプ中**

  Space を押しながら A を押して離す：ステージリセット

  Space を押しながら S を押して離す：ステージセレクトに戻る

  Space を押しながら D を押して離す：ヘルプ内のチュートリアルへ

  Space を押しながら F を押して離す：ゲームに戻る

- **ヘルプ内のチュートリアル中**

  Space を押しながら F を押して離す：ヘルプに戻る

- **ゲーム終了時（クリア・ゲームオーバー）**

  Space を押しながら A を押して離す：同じステージを始める

  Space を押しながら S を押して離す：ステージセレクトに戻る

■ ***トラップの種類***

キュンキュンなる怪しい音のトラップに接触すると罠が発動します。

・即死トラップ…接触するとゲームオーバーになる

■ ***アイテム***

アイテムに触れると、ピンポーンとなって制限時間が増えます。クリア時の残りタイムがスコアとなるのでできるだけ多くのアイテムを集めてみましょう。

■ ***ボーダーライン***

シューシューなる部屋の壁（ボーダーライン）に触れると、ゲームオーバーになります。壁も近づくと警告音を出しますので、注意してください。

■ ***ワンポイント・アドバイス***

トラップやボーダーラインは、近づくと感知器がその位置を察知して、罠のある辺りから音が聞こえるようになります。カチカチなる時限爆弾よりも小さい範囲でしか音が聞こえないので、注意して進みましょう。

さうんど おんりぃ２　最終報告書

PM　株式会社インテム　菊地徹也
環境情報学部４年　橋山牧人
環境情報学部４年　藤原育実

目次

第 1 章.　プロジェクトの概要 4
1.1.　背景 4
1.2.　目的 4
1.3.　目標 4
1.4.　体制 4
1.5.　開発環境 5
1.6.　開発期間 6
1.7.　成果物 6
1.8.　プロジェクトの評価 7
1.8.1.　評価基準 7
1.8.2.　評価の実施について 7
1.9.　プロジェクト運用のルール 7
第 2 章.　プロジェクト実績報告 8
2.1.　スケジュール 8
2.1.1.　α版開発 8
2.1.2.　β版開発 8
2.2.　プロジェクト実績 8
2.2.1.　プログラムステップ数（α版開発） 8
2.2.2.　プログラムステップ数（β版開発） 10
2.3.　作業工数 10
2.3.1.　α版作業工数 10
2.3.2.　β版作業工数 11
2.4.　ユーザ評価 11
2.4.1.　実施内容 11
2.4.2.　成功目標 11
2.4.3.　アンケート結果 12
2.5.　総合評価 12
第 3 章.　α版開発 14
3.1.　開発方針 14
3.1.1.　開発の流れ 14
3.1.2.　ミニゲームの量産 14
3.1.3.　ゲーム特有の開発手法 15
3.1.4.　ミニゲーム企画立案 15
3.1.5.　Java による小規模ゲームの開発 17

3.1.6. C++による小規模ゲームの開発 19
3.2. ミニゲーム評価 22
3.2.1. クライアント評価 22
3.2.2. 中間報告会 23
3.3. ミニゲーム開発を踏まえて 24
3.3.1. 質の高い音の重要性 25
3.3.2. 企画に対するモチベーション 25
3.3.3. 新しい企画の模索 26
3.4. 虫捕りゲーム開発 28
3.4.1. 企画書の作成 28
3.4.2. 開発方針の確認 38
3.4.3. 音環境の作成 38
3.4.4. ゲームの骨格部分の作成 41
3.5. α版評価（ユーザレビュー） 42
第 4 章. β版開発 45
4.1. β版実装 45
4.1.1. ユーザインタビューの反映 45
4.1.2. デバッグ・修正およびリファクタリング１ 46
4.1.3. ゲームを面白くするためのギミックの追加 53
4.1.4. デバッグ・修正およびリファクタリング２ 66
4.1.5. ライブラリについて 76
4.1.6. 最終的なクラス構成とソースコードの規模 80
4.2. β版評価 88
4.2.1. 横浜市立盲学校の生徒の方々からの意見・指摘 93
4.2.2. 大岩研関係者の方々からの意見・指摘 94
4.2.3. クライアントからの意見・指摘 95
4.2.4. 「虫捕りゲーム」から得られたもの 96
第 5 章. 添付資料 98

第1章. プロジェクトの概要

本章では本プロジェクトにおけるプロジェクトの定義，及びプロジェクトの計画について説明する．

1.1. 背景

「映像を用いないゲーム」は，5.1ch のサラウンド音響環境を利用することで，音源の発音する定位や位置情報，音色，リズムなどを聞きわけ，仮想の空間を体感するという新しいギミックを有するゲームである．映像のある一般的なゲームをプレイするよりも想像力を高めることができるなど，シリアスゲームの一環としての利用が想定されている． また，視覚にハンディキャップを有する方々にもゲームの面白さを伝えることができ，ゲームを楽しめるターゲットを増やすことができる．

1.2. 目的

- ユーザが繰り返し遊んでくれるような面白いと感じられるゲームを開発するその開発を通じて，プロジェクト進行，ゲーム製作を勉強していく．
- 人に使ってもらえるもの，人に楽しんでもらえるものを開発する．

1.3. 目標

ゲーム開発のプロセスやプロジェクトの進行を学習し，それによって，ソフトウェア開発手法との違いを学習する．それをメンバーごとに最終報告書にまとめる．

1.4. 体制

全体的な体制については以下の図を参照のこと．

- プロジェクトメンバー
  菊地　徹也（ＰＭ）
  橋山　牧人
  藤原　育実
- クライアント
  南雲　玲生（株式会社ユードー）

- ユーザ

  健常者と視覚にハンディキャップを有する方々

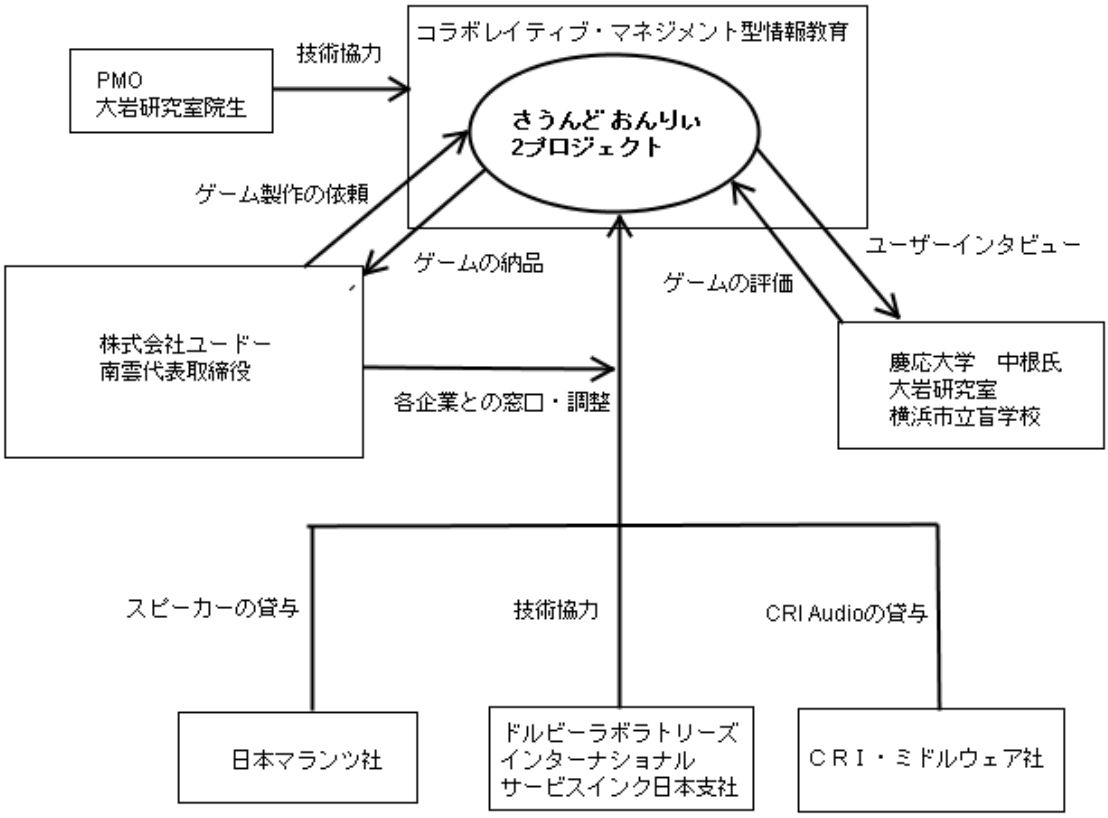


図 1-1　さうんど おんりぃ２プロジェクト　体制図

1.5. 開発環境

- コーディング
  - VisualStudio2003.NET（C++）
- ライブラリ各種
  - CRI Audio（5.1ch サラウンドライブラリ）
  - SDL（Simple Directmedia Layer）
  - Boost
- ヘッドフォン
  - 5.1ch サラウンドヘッドフォン
- データ共有
  - Wiki
  - SVN

1.6. 開発期間

開発期間は，2006 年 10 月 5 日から 2007 年 1 月 31 日である．

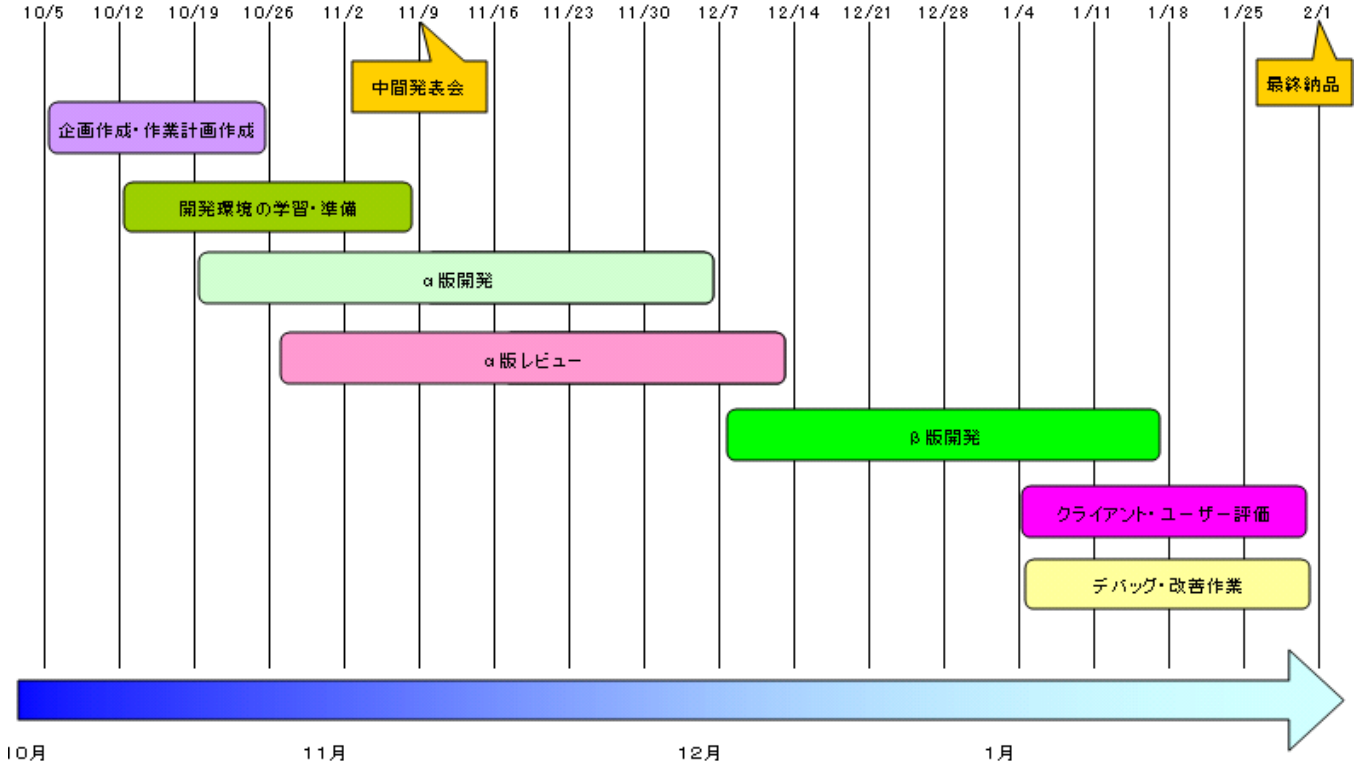


図 1-2　全体スケジュール（予定）

1.7. 成果物

- ソフトウェア (映像を用いないゲーム)
- ゲーム企画書
- ソースコード
- リソースデータ (音楽, 効果音, イメージ)
- 取扱説明書
- プロジェクト定義書

1.8. プロジェクトの評価

1.8.1. 評価基準

- ユーザにゲームをプレイしてもらい，アンケートを記入してもらう
- アンケートの評価が去年の評価を上回ることを目標とする
  - 面白いと思う人が全体の 7 割以上（前回は 5 割）
  - もう一度遊びたいと思う人が全体の 8 割以上（前回は 7 割）
- 健常者，視覚にハンディキャップを有する人が共に面白いと感じるものを作る
  - 両者の評価に明確な差がなく，両者から目標とする評価を得る

1.8.2. 評価の実施について

- α版の評価を 11 月中旬に行う（慶応大学中根様，大岩研究室）
  - 意見を聞く
- β版の評価を 12 月下旬～1 月初旬（横浜市立盲学校，大岩研究室）
  - アンケートについて，先学期作成したものを改良して利用する

1.9. プロジェクト運用のルール

- 毎週木曜日の進捗発表会の発表者はプロジェクトメンバーが持ち回りで担当しその準備も行う．
- メンバーは，毎週月曜5限目にミーティングを行う．
- プロジェクトメンバーのコミュニケーションには，さうんどおんりぃ2のメーリングリスト，Wiki を利用し，どれでも連絡とれないようであれば，携帯電話で連絡を取り合う．
- ファイルの共有は，Wiki，WebDAV，SVN などを利用し共有する．

第2章. プロジェクト実績報告

2.1. スケジュール

2.1.1. α版開発

当初計画した通りに進めることができなかった.

たくさんのゲームを中間報告会までに作成することを目的として開始したが，メンバーの作業時間が思うように取れないことが多く，実際には，ゲームとして，中途半端なものを中間報告会で発表することになってしまった.

中間報告会の反省を受けてα版の開発期間を延ばし，中間報告会までの反省と学習した知識で，新たな企画を作成し，現在の企画「虫捕りゲーム」の企画をまとめた.

2.1.2. β版開発

作業開始は，遅れてしまったが，α版開発時からの技術向上もあり，順調に進むことができた.

開発も後半に差し掛かると，自分達で，ゲームが面白くなるにはどうしたらいいのかと考え，さまざまな調節や工夫を取り入れ面白くするにはどうするかを考え作業をするようになった.

2.2. プロジェクト実績

このプロジェクトは，見積もり行っておりませんので実績だけ記載します.

2.2.1. プログラムステップ数（α版開発）

α版開発にあたって，いきなり大きな規模のゲームを作るのではなく，小規模なゲームを複数作ることにした. α版の期間で開発したのは，下記のゲームとなる.

- 方向当てゲーム
- 聖徳太子ゲーム

- サウンドシュート
- 聖徳太子ゲーム改

はじめはすぐ開発ができるということで java を用いて開発を行ってもらった．その後，C++での開発となり規模がだんだんと大きくなっていった. 以下はα版で開発したゲームのステップ数となる．

方向当てゲーム

ファイル形式	行数	コメント	空行	Logic	ファイル数
java	289	52	43	194	3
合計	289	52	43	194	3

聖徳太子ゲーム

ファイル形式	行数	コメント	空行	Logic	ファイル数
java	338	70	29	239	3
合計	338	70	29	239	3

さうんどシュート

ファイル形式	行数	コメント	空行	Logic	ファイル数
cpp	967	194	151	622	9
h	355	12	63	280	10
合計	1,322	206	214	902	19

聖徳太子ゲーム改

ファイル形式	行数	コメント	空行	Logic	ファイル数
cpp	1,025	199	154	672	9
h	360	12	59	289	10
合計	1,385	211	213	961	19

2.2.2. プログラムステップ数（β版開発）

α版の開発のあと，β版の開発となった．α版での反省，改善点などを踏まえ企画を考えひとつのゲームの完成を目指した．以下はβ版で開発したゲームのステップとなる．

β版開発ゲーム

ファイル形式	行数	コメント	空行	Logic	ファイル数
cpp	3,359	802	458	2,099	31
h	1,235	197	224	814	33
合計	4,594	999	682	2,913	64

2.3. 作業工数

2.3.1. α版作業工数

α版開発における作業工数は下記の表の通りである．作業的にはゲームの開発工数というよりは，どちらかというと勉強会のほうに時間を割かれている．実際，ミニゲームを複数開発する予定ではあったが，作業時間がとれず，ミニゲームを量産することができなかった．

α版作業工数表

作業項目	実績	
	橋山	藤原
ミーティング	12.0	10.5
ユーザレビュー	2.0	2.5
環境整備･学習	7.0	8
開発･企画	14.0	12.5
勉強会	28.5	14.5
その他･資料作成	2.5	4
合計	66.0	52.0

2.3.2. β版作業工数

実際にスタートする企画が決まり，作業を開始した．α版に比べると，作業時間も増え，プロジェクトの活動が活発になったように感じられる．

β版作業工数

作業項目	実績	
	橋山	藤原
ミーティング	13.5	13
ユーザレビュー	5	5
勉強会	11	-
環境準備	7.5	7
開発･企画	150.5	74
その他	47	39
合計	234.5	138

2.4. ユーザ評価

「映像のないゲーム」の成功評価をユーザに遊んでもらいアンケートを記入してもらい，その結果で，今回のゲーム開発の成功・失敗を判定することにした．

2.4.1. 実施内容

アンケートは下記の方々にご協力をいただいて，実際にプレイしてもらい，アンケートの記入をしていただいた．

- 大岩研究室の方々
- 横浜市立盲学校の方々
- 株式会社ユードーの方々

2.4.2. 成功目標

成功目標としては，アンケート項目の「ゲームを遊んでみて，面白い・もう一度プ

レイしてみたいと思う人の割合が，前期の『さうんど おんりぃ』プロジェクトの評価を超えること，を目的とし以下の目標を設定した．

- 面白いと思う人が 7 割以上
- もう一度やりたいと思う人が 8 割以上

これは前期での「映像のないゲーム」の最終アンケートでの結果

- 面白いと思う人が 5 割
- もう一度やりたいと思う人が 7 割

を超えることを目的として設定をした．

2.4.3. アンケート結果

アンケートを実施した結果は下記の通りである．

- 面白いと思う人が 8 割 3 分
- もう一度やりたいと思う人は 10 割

目標と設定していた回答を上回ることができ，プロジェクトは成功ということになった．ただ，残念なことに「非常に面白い」という回答を得ることが出来なかったというのが非常に残念ではあるが，まだまだ改善の余地はあるので，今後，改善を重ねていけば挽回できると思う．

また，アンケート項目のなかに，「面白いと感じた部分」を聞く項目があったが，この結果が，大岩研と盲学校で違いが現れた．大岩研の方々が，音に臨場感があるというところを面白く感じているのに対し，盲学校の方々はそれほどまで臨場感があるようには感じていないようだった．

そしてこのゲームの一番の肝となる部分であるため，場の臨場感をいかに出すかが，まだ研究，改善が必要ということの結果だと思う．

2.5. 総合評価

まず，プロジェクトの目的であるユーザが繰り返し遊んでくれるような面白いと感じられるゲームを開発することはできたのではないのかと思う．

それは，アンケート結果での目標値を大きく上回ることができたことで達成ができた．
また，プロジェクト進行，ゲーム製作を勉強していくという目標についても，ゲームを完成まで進むことができたことで十分な達成だと思える．

目標の達成に十分なほどの結果を得られたと思う．

第3章. α版開発

3.1. 開発方針

今回のプロジェクトでは，前回の「さうんど おんりぃ」プロジェクトで得られたゲーム作成の方法や技術的なノウハウを活かし，より楽しく遊ぶことのできるゲーム開発を目指した．ただし，前回のゲームの企画を元に開発するわけではなく，ゲームの企画や内容については一新し，今までにないような斬新で面白いゲームを開発することを目標とした．

そこで，ゲーム開発期間の前半は，主にゲームの企画を出すことに費やし，企画が確定した段階で本格的に実装を行い，ゲームを作り込んでいくという方針で進めることにした．

3.1.1. 開発の流れ

今回のプロジェクトでは，ゲーム開発のフェーズを大きく二つのフェーズにわけた．前半のフェーズでは，α版の開発を行い，後半のフェーズではβ版の開発を行った．

α版は，斬新なゲームのアイデアを模索したり，ゲームの方向性を確認したりするために開発する試作版である．そのため，α版では，ゲームに多少のバグや不具合があっても，遊ぶのに支障が出ない程度であれば許容するという方針で開発した．また，ゲームの完成度についても高いものは求めず，ゲームの大まかなコンセプトや雰囲気がつかめる程度のものであればよいということにした．α版の開発で，企画がほぼ固まった段階で，β版の開発に移行する．

β版は，ゲームの内容を評価していただくことを目的とした，完成品に近いものである．そのため，β版では，ゲームの根幹に関わるような重大なバグは全て解消し，完成度についても正式版の機能を一通り備えた，クオリティの高いものを完成させるという方針で開発した．

3.1.2. ミニゲームの量産

α版の開発では，斬新で面白いゲームのアイデアを出すために，また，ゲームの開発に慣れる，ゲーム開発のイメージをつかむために，手軽に開発できる規模のミニゲームを多数開発することにした．ミニゲームは，基本的にチームメンバーが個々に別々のアイデアを出し，別々に開発するという方法を採った．

出来上がったゲームはチームメンバー，または研究室の方々にプレイしてもらい，面白かった点は残す，あるいは強化し，不満や要望があれば改善しながら，また新しいゲームを開発する．そして，ある程度ゲームの方向性やコンセプト等が固まった段階で，最終的に一つの作品にまとめ上げるという方法を採ることにした．

3.1.3. ゲーム特有の開発手法

今回の開発は，PM がゲーム業界の方ということもあり，一般的にゲーム業界で行われている開発方法に従って行うことにした．

企画書については，ゲーム開発が進むにつれて新しいアイデアの追加や修正を行う必要性が生じるので，最初の段階かつ詳細で綿密な企画書を書いてもあまり意味がない．そのため，企画書は概要を記述する程度にとどめ，開発が進み，企画に変更が生じた段階で随時書き直していくという方針で進めることにした．

設計書については，企画に変更が生じ，そのたびに設計書を書き直すのは非効率であり，また，設計書自体の必要性も低いため，設計書は作成しないことにした．ただし，設計自体を行わないわけではなく，設計についての議論はチーム内で行い，チームメンバー全員が設計の内容については共有していた．また，後々の混乱を招くことがないように，重要な事柄はソースコード内にコメントとして記述しておくよう心がけた．

3.1.4. ミニゲーム企画立案

ミニゲームを作成するにあたって，まずゲームの企画を思いつく限り挙げていき，その中から面白そうなものを選び，実装していった．

企画立案の段階で挙がっていたものとしては，以下のようなものがあった．

- 旗揚げゲーム
  - 「右上げて」,「左上げて」,「右下げて」,「左下げて」といった音に合わせて，左右のキーを押して旗揚げゲームをするゲーム．
  - ゲームとしては面白そうだが，ゲームの幅を広げるのが難しそうで，単純になりそうであるという欠点があった．また，音の鳴る方向を変える必要がないので，サラウンド音声を活かすのが難しそうであるという問題もあった．

- 落ちてくる障害物をよけるゲーム
  - 複数の音が上から下に移動してきて，それらの音にあたらないように移動してよけるゲーム．音にあたってしまうとゲームオーバーとなる．ゲームのプレイ時間

が長くなるほど，音の数が増え，難易度が上がる．

- リズムゲーム
  - 上下左右から，様々なタイミングで音が鳴ってくるので，タイミングよく音のなる方向のキーを押すゲーム．タイミングが正確なほど高得点が得られ，方向を間違えたり，タイミングがずれたりするほど，得点が低くなる．

- モグラ叩き
  - 様々な方向から，モグラの鳴き声が聞こえてくるので，聞こえたところのモグラを叩いていくゲーム．
  - どのようにして，鳴いているモグラを選択するのか等が難しそうであるという問題がある．

- サウンドノベル
  - ストーリーを音声で読み上げ，ストーリーの途中で選択肢が現れ，どの選択肢を選ぶかによってストーリーが変化していくゲーム．
  - 使用する音は主に声だけなので，サラウンド音声を使う必要性が少なそうであるという問題がある．

- スポーツゲーム
  - 音のみでボウリングやテニス，サッカーやボクシング等のスポーツを再現するゲーム．
  - 音のみでどこまで現実のスポーツに近い臨場感を出せるかが課題となりそう．

- 鬼ごっこ
  - 音を頼りに鬼を追いかけ，鬼を捕まえるゲーム．
  - 前期の「さうんど おんりぃ」に近い

- 聖徳太子ゲーム
  - 一度に複数の音声が同時に読み上げられ，それぞれの音声で何をしゃべっていたのかを当てるゲーム．
  - 使用する音声は人の声だけなので，サラウンドを使う必要性が少なそうであるという問題がある．

- 記憶ゲーム
  - いくつかの音が様々な方向から聞こえてくるので，その音がどちらから聞こえて

きてかを当てるゲーム．正解数が増えるごとに記憶しなくてはならない音が多くなっていき，難易度が上がる．

この中から，チームメンバー各自が面白そうだと思うものを選び，それをベースにミニゲームを開発することにした．メンバー内で相談した結果，最終的に橋山は記憶ゲーム，藤原は聖徳太子ゲームをそれぞれ選び，開発を行った．

3.1.5. Java による小規模ゲームの開発

今回のゲーム開発では，5.1ch のサラウンド音声を使用する．しかし，サラウンド音声を扱うためのライブラリが Java にはなく，C++用のものしかなかったため，開発言語は C++を使用するということに決定した．ただし，チームメンバーは C++によるソフトウェア開発の経験がなく，慣れていなかったため，当初はチームメンバーが慣れ親しんでいる Java を使用してミニゲームを開発することにした．

ミニゲームの開発は，チームメンバーがそれぞれ別々のものを個別に開発することにした．以下，チームメンバーがそれぞれ開発したゲームについて説明する．

- 方向当てゲーム（記憶ゲーム）　担当：橋山

  方向当てゲームは，上下左右のいずれかの方向から音を流し，その音がどの方向から聞こえたかを記憶していくゲームである．正解すると，直前に流れた音に新しい音が一つ追加され，記憶する音の数が増えていく．間違えたところでゲームオーバーとなり，いくつまで正解できるかを競うゲームである．

  このゲームは Java で開発したため，サラウンド音声用のライブラリを使用することができなかったので，初めから定位や音量を決めた音を方向ごとに用意し，それらを流すという方法を採っていた．しかし，サラウンドを使用しないとどうしても音の鳴っている方向がわかりづらくなってしまい，ゲームとしてはプレイしづらいものになってしまった．

  また，音が上下左右の固定した場所から聞こえてくるだけでは単調で飽きてしまうので，音の聞こえてくる方向を移動させるといった要素があると面白くなるのではないかという意見も出た．それをふまえて，次はサラウンド音声を活かして，音の移動を取り入れて新しいゲームを開発することとなった．

- 方向当てゲーム（記憶ゲーム）の規模

  方向当てゲーム（記憶ゲーム）の規模については，以下の通りである．

ファイル形式	行数	コメント	空行	Logic	ファイ￥ル名
java	162	28	22	112	MemoryGame.java
java	62	12	8	42	MidiPlayer.java
java	65	12	13	40	Sound.java

ファイル形式	行数	コメント	空行	Logic	ファイル数
java	289	52	43	194	3
合計	289	52	43	194	3

- 聖徳太子ゲーム　担当：藤原

  聖徳太子ゲームは，同時に複数の果物の名前を読み上げ，それを全て聞き分けるゲームである．読み上げられた名前はキーボードで入力し，それが全て合っていたら正解となり，新しい問題が出題される．また間違っていたら，ゲームオーバーとなる．

  このゲームは，内容としては面白いが，キーボードを使った入力が大変であるという問題があった．例えば，タッチタイピングができない人にとっては，正しい文字を入力するのが困難であるという問題がある．また，一度入力した文字は修正ができない等の不具合もあった．

  また，この時点では読み上げる果物の音声に，機械的な合成音声を使用していたため，イントネーションが不正確で，聞き取りづらいという問題があった．さらに，全ての音声が同じ位置から聞こえてくると，言葉が混じって聞こえてしまうため，それぞれの音声は別々の位置から聞こえるようにした方がよいという意見が出た．それを踏まえて，次はこのゲームの基本ルールは変えずに，より遊びやすいものを開発することとなった．

- 聖徳太子ゲームの規模

  聖徳太子ゲームの規模については，以下の通りである．

ファイル形式	行数	コメント	空行	Logic	ファイル名
java	81	26	11	44	Input.java
java	133	20	7	106	SoundManager.java
java	124	24	11	89	SyotokutaishiGame.java

ファイル形式	行数	コメント	空行	Logic	ファイル数
java	338	70	29	239	3
合計	338	70	29	239	3

3.1.6. C++による小規模ゲームの開発

一度目のミニゲーム作成では，普段から慣れている Java で実装を行い，手っ取り早く簡単なゲームを開発し，ゲームのイメージをつかんだ．それをふまえて，二度目のミニゲーム作成は，C++のサラウンド音声ライブラリを使用して行った．

二度目のミニゲーム作成も，チームメンバーがそれぞれ別々のものを個別に開発することにした．以下，チームメンバーがそれぞれ作成したゲームについて説明する．

- さうんどシュート　担当：橋山

  さうんどシュートは，音が左から右に移動しながら鳴り続けており，自分の正面に音が来たと思ったらキーを押し，その精度を競うゲームである．

  最初は左の方から音が鳴っており，音量も小さいが，だんだんと音が右に移動していき，音量も大きくなってくる．音が自分の目の前を通り過ぎると，音は自分の右側に遠ざかっていき，音量も小さくなっていく．音が自分の目の前に近く，音量が大きいときにキーを押すほど高いスコアを獲得でき，音が自分の目の前から遠く，音量が小さいほど，スコアは低くなるというゲームである．

- さうんどシュートの規模

  さうんどシュートの規模については，以下の通りである．

ファイル形式	行数	コメント	空行	Logic	ファイル名
h	32	0	7	25	SoundContainer.h
h	39	3	4	32	SoundPlayer.h
h	14	1	1	12	StateMessages.h
h	38	3	9	26	TitleState.h
cpp	57	16	6	35	Game.cpp

cpp	307	64	52	191	GamePlayState.cpp
cpp	44	15	4	25	KeyFlag.cpp
cpp	65	21	9	35	KeyState.cpp
cpp	45	6	9	30	Main.cpp
cpp	85	24	13	48	OnMapObject.cpp
cpp	91	17	16	58	SoundContainer.cpp
cpp	159	6	21	132	SoundPlayer.cpp
cpp	114	25	21	68	TitleState.cpp
h	23	0	5	18	Game.h
h	58	0	9	49	GamePlayState.h
h	18	0	3	15	KeyFlag.h
h	33	0	7	26	KeyState.h
h	64	4	14	46	OnMapObject.h
h	36	1	4	31	SDLCommonFunctions.h

ファイル形式	行数	コメント	空行	Logic	ファイル数
cpp	967	194	151	622	9
h	355	12	63	280	10
合計	1,322	206	214	902	19

- 聖徳太子ゲーム改　担当：藤原

  聖徳太子ゲーム改は，Java で開発した聖徳太子ゲームをベースにして，読み上げられる音声が様々な位置から流れてくるようにしたものである．

  Java で作成したゲームは，音の定位が一定だったため，音声が聞き取りづらいという問題があったが，C++版では，サラウンド音声を使用して，それぞれの音声が別々の位置から聞こえてくるように改良したため，Java で作成したものよりも音声が聞き取りやすくなっている．

- 聖徳太子ゲーム改の規模

  聖徳太子ゲーム改の規模については，以下の通りである．

ファイル形式	行数	コメント	空行	Logic	ファイル名
h	39	3	4	32	SoundPlayer.h
h	14	1	1	12	StateMessages.h
h	35	3	6	26	TitleState.h
cpp	57	16	6	35	Game.cpp
cpp	365	69	55	241	GamePlayState.cpp
cpp	44	15	4	25	KeyFlag.cpp
cpp	65	21	9	35	KeyState.cpp
cpp	45	6	9	30	Main.cpp
cpp	85	24	13	48	OnMapObject.cpp
cpp	91	17	16	58	SoundContainer.cpp
cpp	159	6	21	132	SoundPlayer.cpp
cpp	114	25	21	68	TitleState.cpp
h	23	0	5	18	Game.h
h	66	0	8	58	GamePlayState.h
h	18	0	3	15	KeyFlag.h
h	33	0	7	26	KeyState.h
h	64	4	14	46	OnMapObject.h
h	36	1	4	31	SDLCommonFunctions.h
h	32	0	7	25	SoundContainer.h

ファイル形式	行数	コメント	空行	Logic	ファイル数
cpp	1,025	199	154	672	9
h	360	12	59	289	10
合計	1,385	211	213	961	19

3.2. ミニゲーム評価

3.2.1. クライアント評価

本プロジェクトのクライアントである株式会社ユードーの南雲玲生代表取締役に，ミニゲームについての意見をいただいた.

南雲代表取締役に指摘していただいた点は，主に以下の二点であった.

- 音の質にこだわるべきである

  我々が開発したゲームは，どれも音声に合成音声を用いており，機械的であじけないものになってしまっていた．映像を使わない音だけのゲームにおいては，ゲームの面白さや迫力，魅力は音の質によって左右される部分が非常に大きいというお話しをいただいた．特に，電子的な音は，人間の耳にとって非常に不自然で耳障りであるため，今回のゲームで使用する音は，全て自然の音を使用した方がよいだろうとのことであった.

  例えば，音声は実際に人がしゃべった声を録音する，とりわけ女性の声にするとよいのではないかという意見や，必要な音はマイクを持って出かけていき，自分たちで録音するとよいのではないかという意見をいただいた. また, どうしても手に入らない音や，手軽に手に入らない音があった場合は，南雲代表取締役が相談に乗ってくださるというお話しもいただいた.

- 今までにないような斬新なゲームにして欲しい

  また, 我々が作成したゲームは, 内容がありきたりで, 斬新さに欠けるものであった. そのため，インパクトが弱く，飽きるのも早くなってしまうという欠点があった.

  南雲代表取締役からは，せっかく「映像を用いないゲーム」という新しい試みに挑戦しているのであるから，ゲーム自体の企画についてもより斬新で，今までに例を見ないようなものを作成した方が，作り甲斐もあり，プレイする側としても楽しめるのではないか, また, ゲームが投げかける影響力も大きいのではないかという意見をいただいた.

  例えば，商店街の中を散歩するゲームなどの案を提示していただいた．自分たちで商店街の中を歩きながら商店街の音を録音し，その音をもとに実際に商店街に出かけている様子を音だけで再現でき，さらに商店街の移りゆく音を環境音楽や BGM としても楽しめるようなゲームにすれば，既存のゲームの枠を取り払った，全く新しいジャンルを確立できるのではないかという意見をいただいた.

上記のようなクライアント評価の結果をふまえ，次の開発では南雲代表取締役からいただいた意見を参考に，音の質にこだわり，より斬新なゲームを作ることを目標にした.

3.2.2. 中間報告会

中間報告会では，評価委員会の方々から大変的確かつ有用な意見や指摘をいくつかいただくことができた．中間報告会でいただいた意見は，主に企画について，もう少し洗練された，面白いものを考えてみて欲しいというものであった．また，具体的にいくつかアイデアを出していただいた．評価委員会の方からいただいたアイデアは，以下のようなものである．

- 絶対音感育成ゲーム

  絶対音感育成ゲームは，音を鳴らして，その音の音程を当てるゲームである．ゲームをプレイしているうちに，絶対音感を身につけられるようなものだと，面白そうだし需要も高いのではないかという意見をいただいた．

- 音キムス

  音キムスは，あるものが振動する音を鳴らし，それが何の音であるのかを当てるゲームである．例えば，小豆やビー玉等の入った容器を振ったり動かしたりしたときの音を鳴らし，それらの音の正体が何なのかを正しく判別し，当てるゲームである．

  臭いを当てる臭いキムス等は，すでにあるそうだが，音キムスは音の大きさや中身に使うものの準備など色々と問題が多く，ゲームとして遊ぶのはなかなか難しいとのことである．そのため，音が準備されていて音の大きさが自由に変えられるゲームであれば，より簡単に音キムスが実現できると思われるので，今回のプロジェクトでチャレンジしてみるのも面白いのではないかという意見をいただいた．

- ファンタジックアドベンチャーゲーム

  ファンタジックアドベンチャーゲームは，映像での描写が難しい世界を，音だけを使って体感できるようなゲームである．例えば，ハリーポッターの魔法の世界などを音だけで再現し，実際に冒険しているような気分が味わえるようなゲームである．

- バーチャルシミュレーションゲーム

  バーチャルシミュレーションゲームは，現実世界で行われていることを音だけでシミュレートして体感できるようなゲームである．例えば，音だけを使って電車の運転を疑似体験できるような，音だけでプレイできる電車で GO のようなゲームである．

  実際に視覚にハンディキャップを有する方々の間では，電車で GO は人気が高いらしく，バーチャルシミュレーションゲームならば，視覚にハンディキャップを有する方々にも楽しく遊んでもらえるのではないかという意見をいただいた．

  我々がβ版で開発することにした虫捕りゲームも，このバーチャルシミュレーショ

ンゲームに近いものである．虫捕りゲームについての詳細は後述する．

また，中間報告会でいただいた質問の内容は，主に以下の 2 点であった．

- 前期のプロジェクトよりも優れている点，進歩した点は何か？

今回の「さうんど おんりぃ 2」プロジェクトは，前期の「さうんど おんりぃ」プロジェクトの続編としてスタートしたものであり，映像を用いないゲームを開発するという目的については共通だった．中間報告会では，前期プロジェクトとの違いに関する説明が欠けていたため，誤解や疑問を生じさせる結果となってしまった．

この質問に対する答えとしては，次のようになる．前期のプロジェクトはサラウンド音声を使ったゲーム作成を行うための技術調査を行うことがメインであり，ゲームの品質についてはあまり高いものを求めていなかった．一方，今期のプロジェクトでは，前期のプロジェクトを通して得られた知識や技術，ノウハウなどを活かし，前期のプロジェクトよりも面白く，完成度の高いゲームを作成することを主眼においており，この点が前期のプロジェクトと異なる点，進歩したといえる点である．

しかし，中間報告会で発表したゲームはクオリティが低く，面白みにも欠けていたため，今期のプロジェクトではゲームの品質を重視するという点をアピールすることができなかった．改めて，ゲームアイデアの抜本的見直しと，音質向上の必要性を認識させられた．

- PM は何をマネージメントするのか？

質問を受けた時，明確な回答を持って回答することが出来なかった．そのことで，私自信，PM のことをまるで理解をしていないということを痛いほど感じ，考えさせられた．

この質問について改めて回答すると，PM とは，プロジェクトの計画とそれに対する現実との差異を認識し，プロジェクトの最終ゴール達成のために，継続的に，適切な軌道修正を行っていくことが PM としての役割である．

今回のプロジェクトでは，面白いゲームを作成する（プロジェクトの最終ゴール）のために，何が面白いのか，そのアイデアで，本当に面白いのかなど，ゲームとしての面白さの修正（プロジェクトの軌道修正）を行っていくのが，私が PM としてのマネージメントだと考える．

3.3. ミニゲーム開発を踏まえて

これまではいくつものゲームを作っては壊すということを繰り返した．それによって「映像を用いないゲーム」を作るための全体的なイメージは掴めたものの，実際に出来上がったゲームは自分たちで遊んでみても正直つまらないものばかりであった．ゲーム開発にお

いて，開発したゲームが面白くないということはその開発の失敗を意味する．そこで，我々は今後の開発において同じ失敗を繰り返さないように，ミニゲームの開発で失敗した原因とそこから得られた反省点について考察した．

3.3.1. 質の高い音の重要性

前回の「さうんど おんりぃ」にも言えることだが，ミニゲームを作る時に我々は動くものを作ることを優先した．表現力が乏しい電子音や合成音声を多用して，音にはまったくこだわらなかった．これは，我々が普段遊ぶ一般的なゲームがイメージとしてあったからである．

映像があるゲームでは，映像が多くの情報を持っており音は副次的な要素である．音自体に情報が少なくても映像でカバーすることが出来るため，結果として面白いゲームになることがありうる．しかし，我々が作っている「映像を用いないゲーム」は文字通り音がすべてである．そのため，ゲーム中の音が表現力に乏しい電子音や合成音声では，そこからゲーム全体をイメージとして捉えることが非常に困難になる．

例えば，ゲームの内容を説明する音声が合成音声であるときと人間の生の声を録音したものであるときを比較してみるとよく分かる．現在無料で利用できる合成音声は，記述した文章を読み上げるといったタイプのものだが，これは基本的に文章を棒読みする．また，文章全体を読ませるとほぼ間違いなくイントネーションが不自然となる．合成音声の説明ではゲームの内容は伝わるかもしれないが，平坦な音声ではゲームの重要な部分が曖昧になってしまう恐れがある．これが人間の生の声であれば，アクセントやイントネーション，微妙な間などを上手に使うことで，どの情報を一番伝えたいのかということを明確にすることができる．ゲームの設定が近未来でロボットが説明しているというような特殊な状況下でない限り，合成音声では人間の生の声に比べて情報がうまく伝わらないのである．

以上のことから我々は，「映像を用いないゲーム」を作る際にはまず質の高い音を収集し，それらを使ってゲームを開発していくべきである，という結論に至った．

3.3.2. 企画に対するモチベーション

ミニゲームを開発するにあたり企画自体はいくつか挙がったものの，どれも似通ったもので自分たちが面白いと思う企画を考えることが出来なかった．これはメンバーに企画力が足りなかったことが原因である．そのような企画を基に開発を進めていったため，当然納得できるような面白さを備えたゲームは出来なかった．

しかし，ゲームを開発しているうちに研究室にいる人や評価委員の方から企画に対する様々なアドバイスを頂いた．中にはとても面白そうな企画を持っている人がいて，是非そ

れを作って欲しいとお願いされたこともあった．そういった中で様々な企画に触れているうちに，自分たちだけで企画を考えようとしていたことが失敗の原因であるということが分かった．我々は企画を実装に結び付けて考えてしまうので，実現可能性や実装の難易度が頭にあるためどうしても視野が狭くなってしまう．しかも，メンバー内でしか企画を話し合っていなかったので，ますますその傾向は強くなっていったのである．

我々には「映像を用いないゲーム」という未知のジャンルに対して，面白い企画を持っている人はたくさんいるという認識が欠けていた．開発者という立場に縛られない自由な発想をする外部の意見や企画を取り入れることで，より魅力的な企画が出来上がるのだということを改めて発見した．

3.3.3. 新しい企画の模索

以上のことを活かし，企画力に定評のある大岩研究室の荒木氏から，色々と意見やアイデアを提供してもらい，それを参考に企画を考えるという方法を採ることにした．荒木氏からは，新鮮で率直な発想と，いくつかの大変興味深いアイデアを提供していただいた．例えば，以下のようなものである．

背景：

荒木氏には，視覚にハンディキャップを有する友達がいて，その友人に子供が生まれた．そこで，健常者の子供と視覚にハンディキャップを有する親が全くストレスを感じずに遊べるゲームがあるとよいのではないかと考えた．視覚にハンディキャップを有する人は，音を立体的に捉えることができるので，その感覚を追体験できるようなゲームがあると面白いのではないだろうか．

現実的には，健常者と視覚にハンディキャップを有する人が一緒に遊ぶことは難しいが，音だけの世界であれば，一緒に遊ぶことが可能である．音を介して，一緒にどこかに出かけたり，遊んだりすることが，何の抵抗も無く楽しむことができる．現実にある楽しい時間を，視覚にハンディキャップを有するか有しないかに関わらず楽しむことができる．そこで，森の中や海の中や街の中，あるいは現実には体験できない，コンピュータの中や身体の中などの場所を移動し，そこで一緒に遊べるようなゲームを作ってみると楽しいのではないだろうか．

具体案：

日常的で，臨場感のあるゲーム：

- 虫捕り
  - 市販されているゲームである「僕の夏休み」のように，季節感や自然の雰囲気が体感できるようなものは，健常者にとっても面白く，魅力的である．虫捕り

という内容ならば，様々な季節や時間の移り変わりが再現でき，森に生息する様々な生き物の存在が感じられるので，面白いのではないだろうか．

- 焼肉
  - 視覚にハンディキャップを有する方は，鉄板がどこにあるかわからないため火傷をしてしまう可能性があるので，焼き肉を食べにいくことができないという．そこで，音で焼き肉を再現し，焼き肉の楽しさを体験できるようなものができたら嬉しいのではないだろうか．

- オリエンテーリング
  - 夏のキャンプに参加し，クイズに答える等．

- 特定の人物を捜すゲーム
  - 人混みの中で迷子になった子供を捜す等．

- 料理
  - 音だけで料理の臨場感が味わえるようなゲーム．

- 釣り
  - 音だけで釣りの臨場感が味わえるようなゲーム．

- 肝試し
  - 音だけで肝試しの臨場感が味わえるようなゲーム．

対戦要素のあるゲーム：

- 雪合戦
- スイカ割り
- ビーチバレー
- ピンポン（卓球）
- 蚊やハエを叩く
- 射的

現実では体験できない世界をシミュレートするようなゲーム：

- 人間の体内を探検する
  - 自分が小さな医者になって，病原菌と戦うことができる．

- 宇宙空間で敵と戦う

- 人間以外の生き物になって，人間以外の生き物の世界を疑似体験する
  - 野良猫になって，夜に人間から食べ物を奪う等.

以上のような案の中から，チームメンバーが最も興味を示し，最もアイデアの膨らんだ虫捕りゲームの案を採用した．この虫捕りゲームのアイデアを基に再度ゲームを作り直し，ある程度作成してみて面白くなりそうであれば，企画を虫捕りゲームに絞り，α版最後の開発とすることにした.

3.4. 虫捕りゲーム開発

α版として最終的に企画がまとまった虫捕りゲームについて，開発方針を決めて，企画書を作成した.

3.4.1. 企画書の作成

「虫捕り」とキーワードをベースにメンバー間で議論した結果,「森の中を散歩しながら虫を捕まえていく」という基本的なゲームの企画が出来上がった．最初に作った企画書は以下の通りである（図 2-1～2-5 参照）.

図 2-1　虫捕りゲーム企画書（1 ページ）

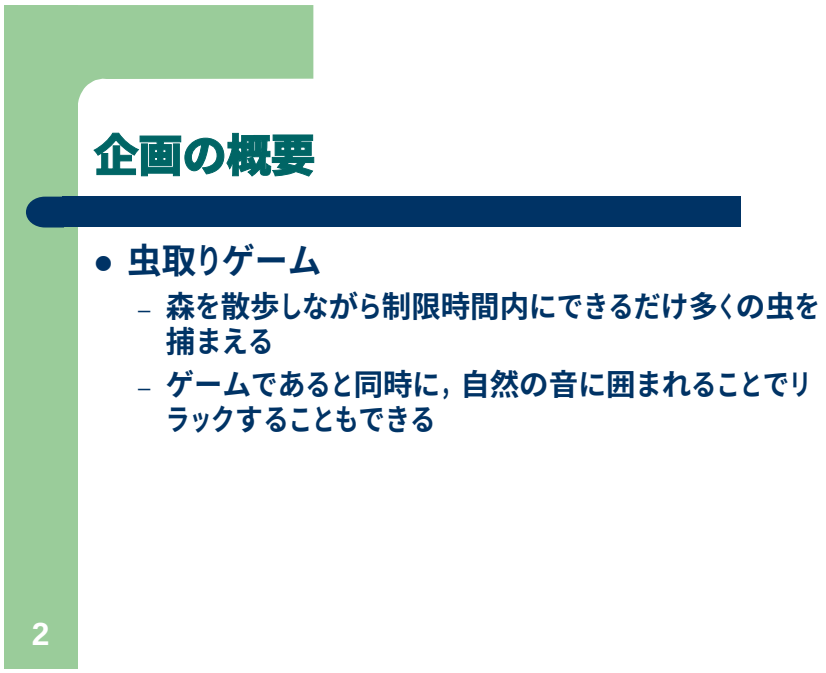


図 2-2　虫捕りゲーム企画書（2 ページ）

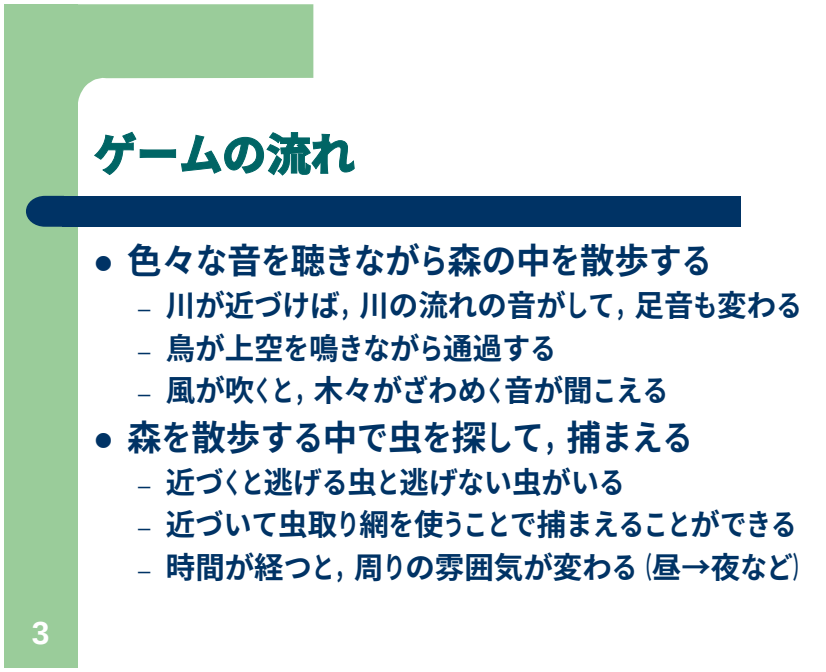


図 2-3　虫捕りゲーム企画書（3 ページ）

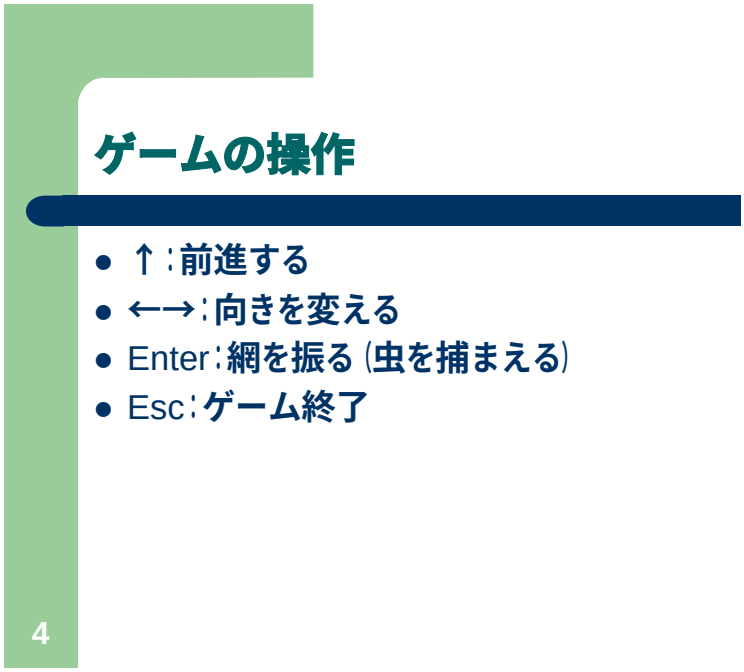


図 2-4　虫捕りゲーム企画書（4 ページ）

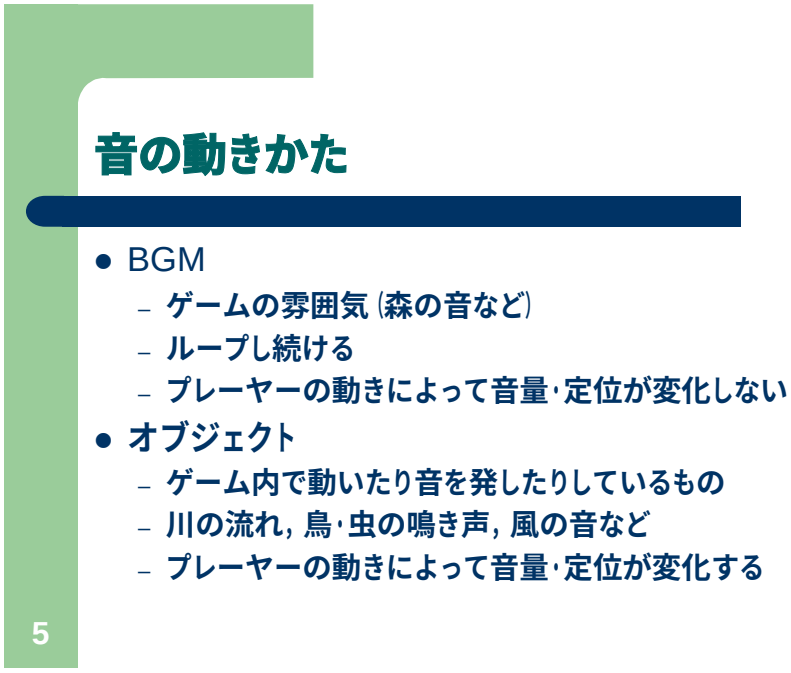


図 2-5　虫捕りゲーム企画書（5 ページ）

本プロジェクトでは企画書が仕様書を兼ねるため，基本的な企画に加えてゲームの流れや操作方法，登場するゲーム要素に対しての大まかな仕様と，シーンごとに必要となる音のリストなどを併せて記載した. 修正した企画書は以下の通りである（図 2-6～2-19 参照）.

図 2-6　虫捕りゲーム企画書修正後（1 ページ）

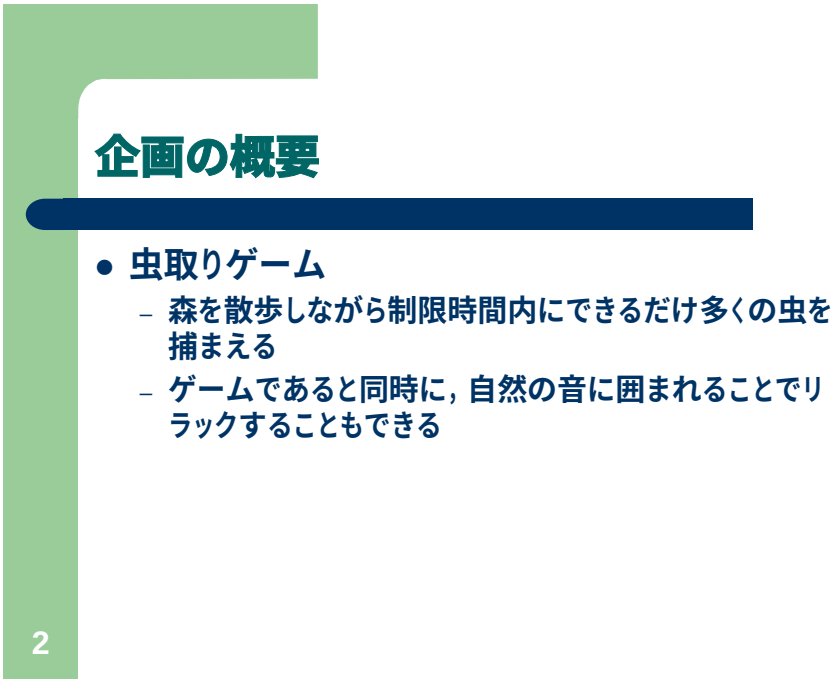


図 2-7　虫捕りゲーム企画書修正後（2 ページ）

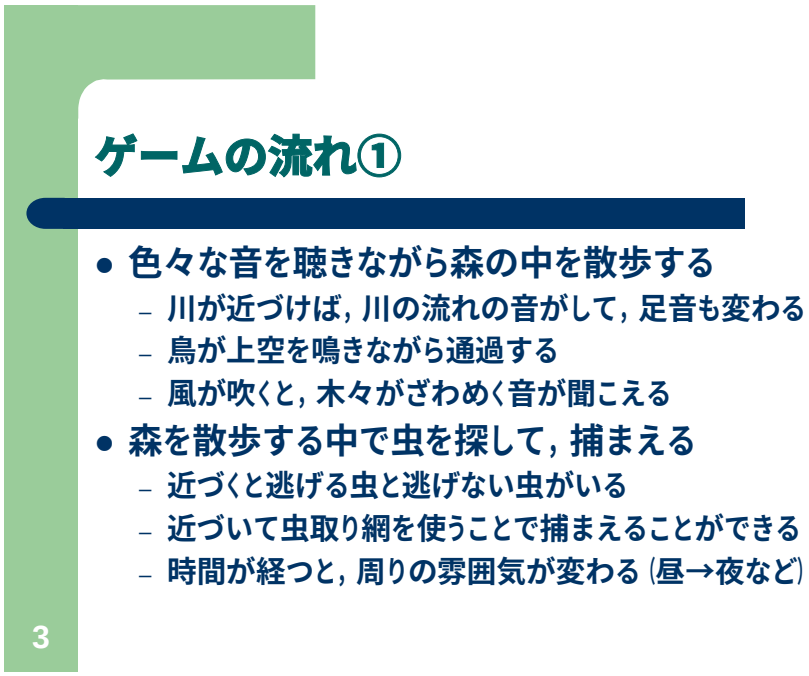


図 2-8　虫捕りゲーム企画書修正後（3 ページ）

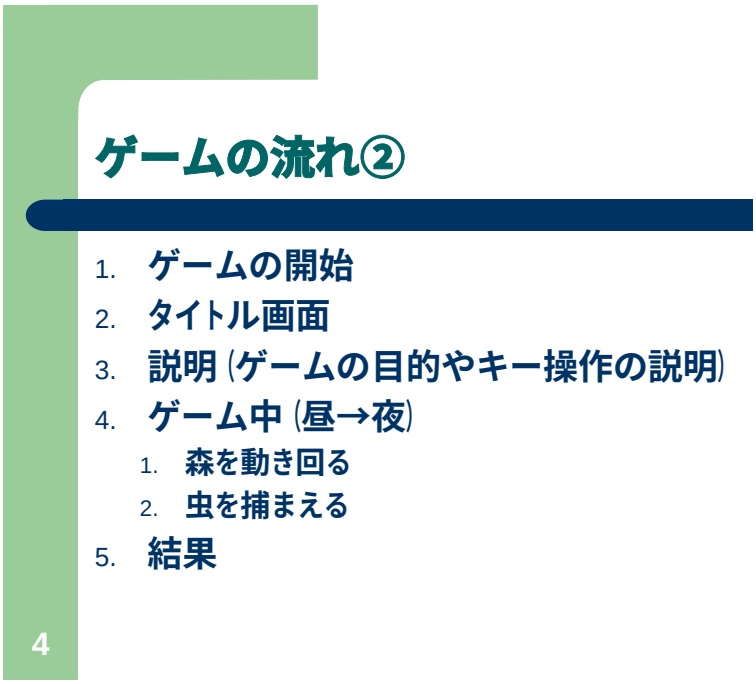


図 2-9　虫捕りゲーム企画書修正後（4 ページ）

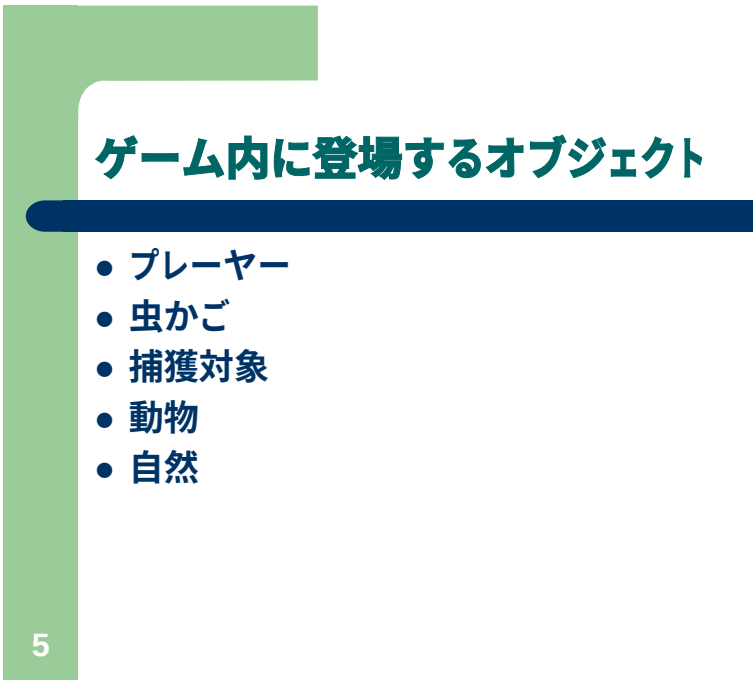


図 2-10　虫捕りゲーム企画書修正後（5 ページ）

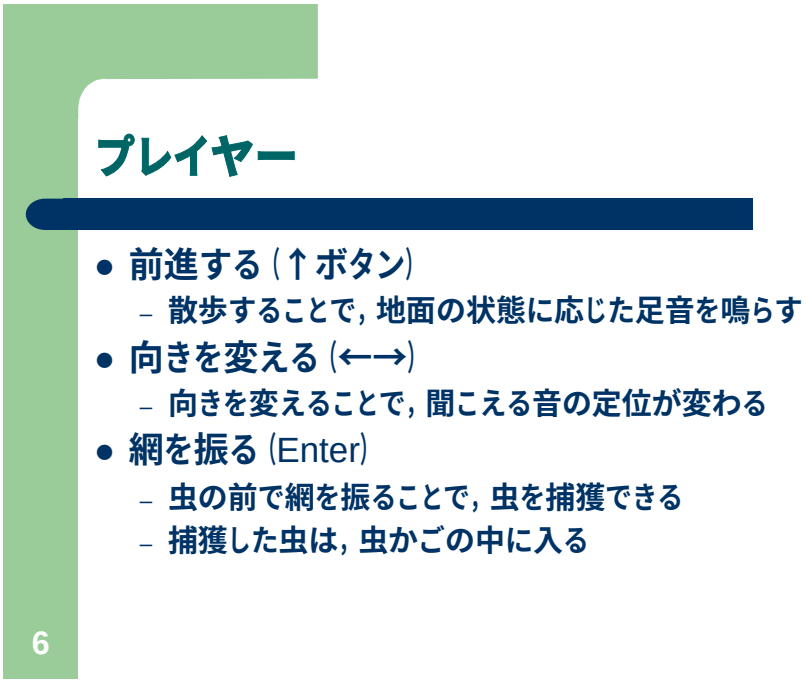


図 2-11　虫捕りゲーム企画書修正後（6 ページ）

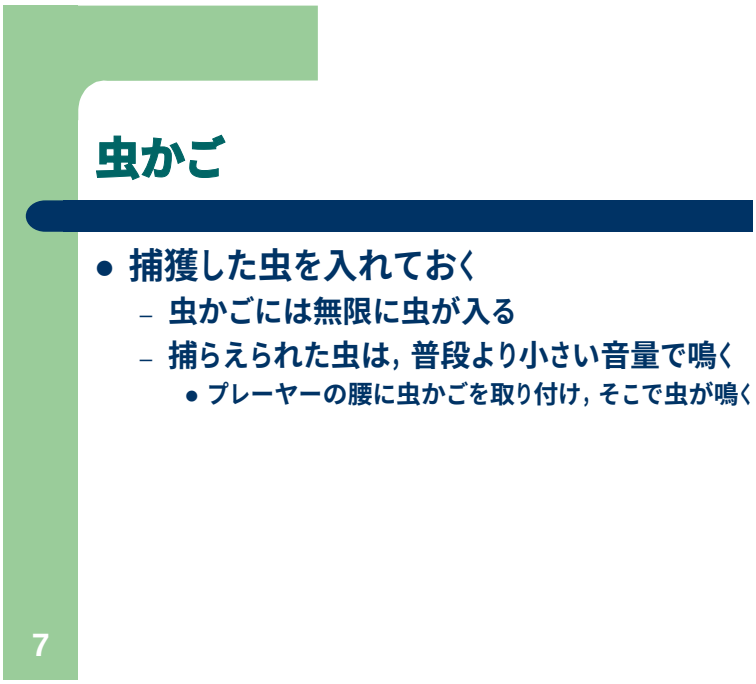


図 2-12　虫捕りゲーム企画書修正後（7 ページ）

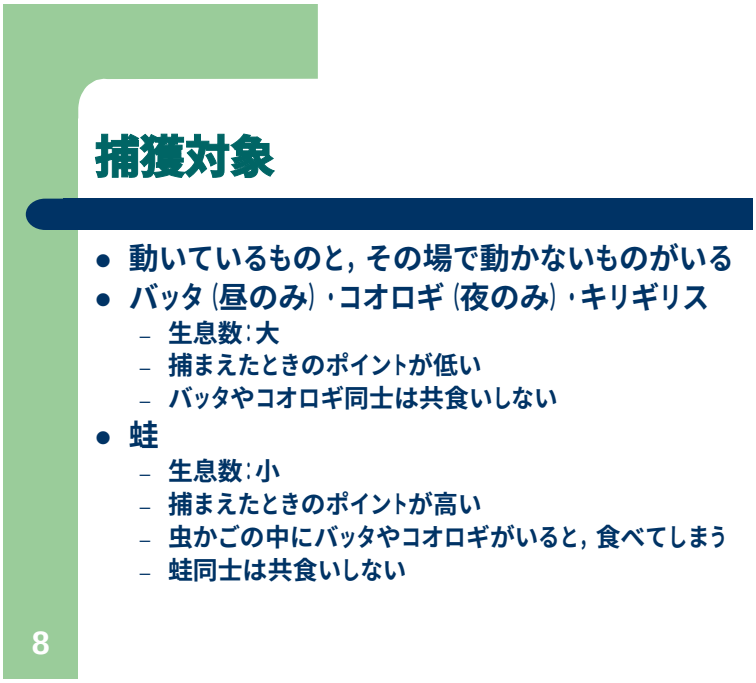


図 2-13　虫捕りゲーム企画書修正後（8 ページ）

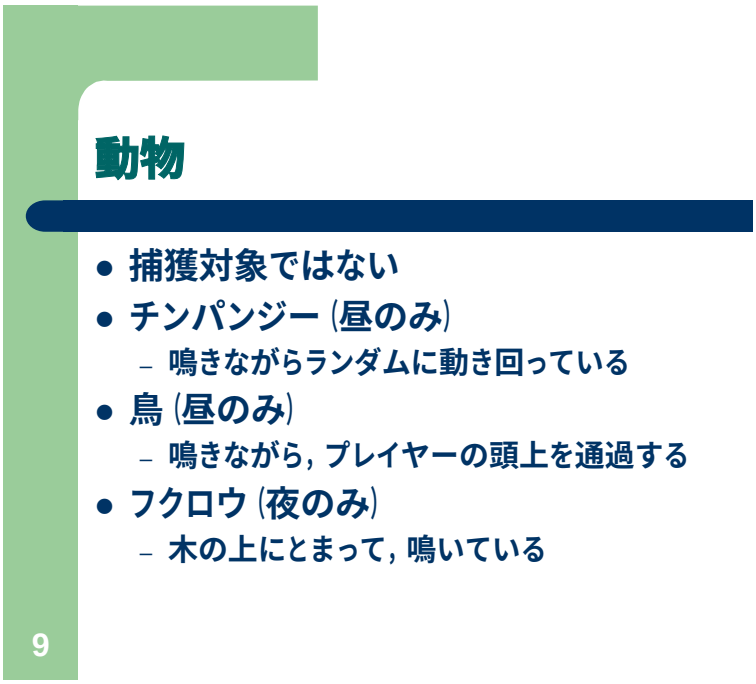


図 2-14　虫捕りゲーム企画書修正後（9 ページ）

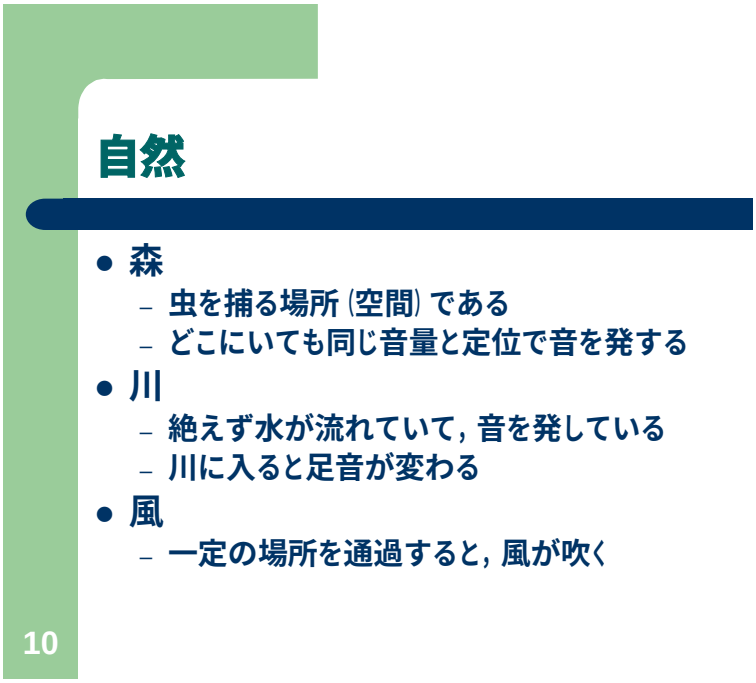


図 2-15　虫捕りゲーム企画書修正後（10 ページ）

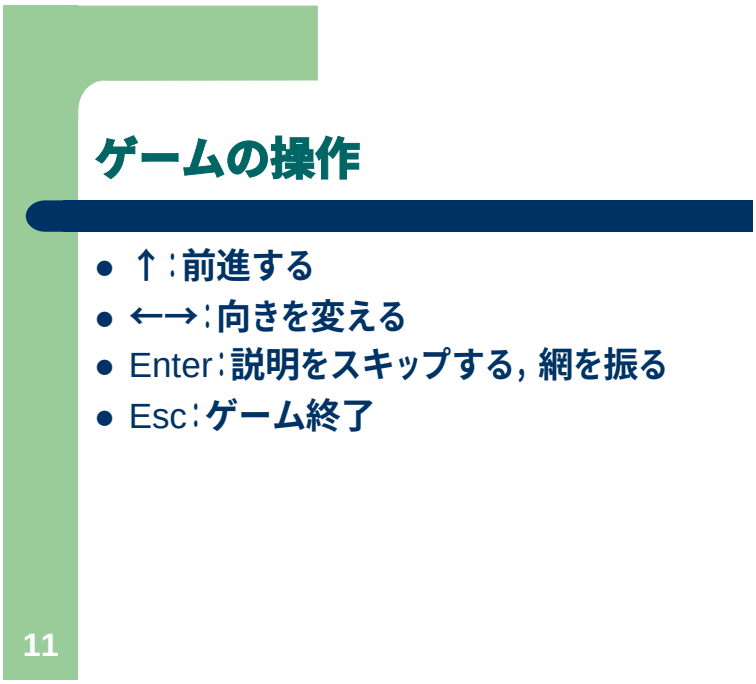


図 2-16　虫捕りゲーム企画書修正後（11 ページ）

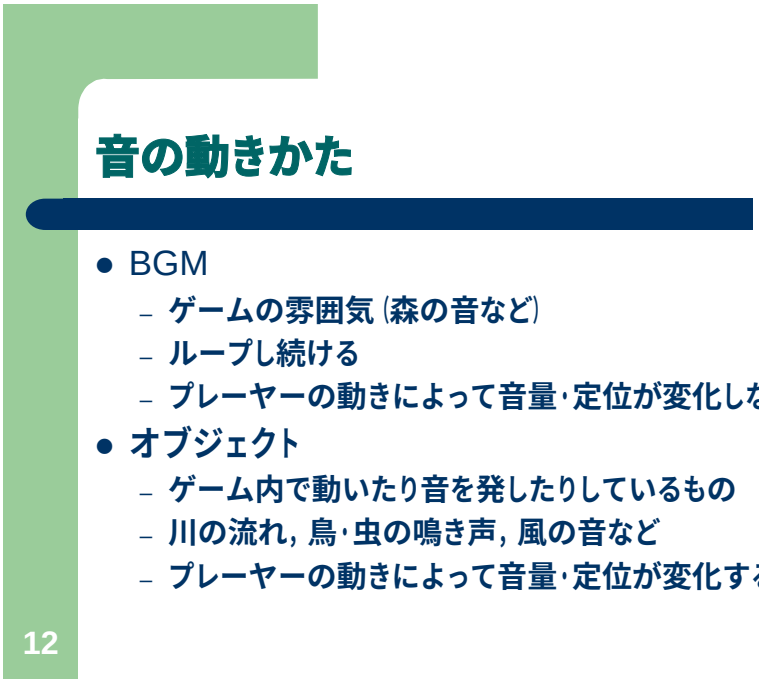


図 2-17　虫捕りゲーム企画書修正後（12 ページ）

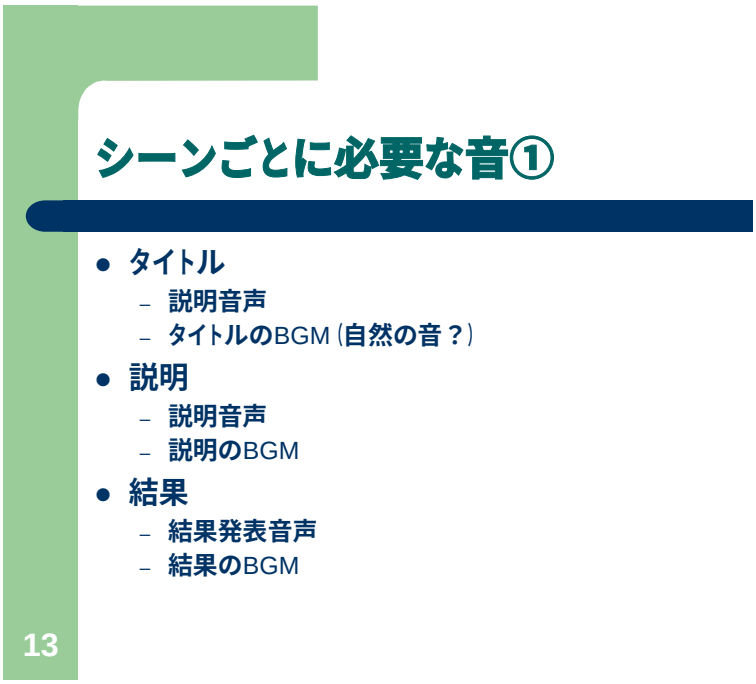


図 2-18　虫捕りゲーム企画書修正後（13 ページ）

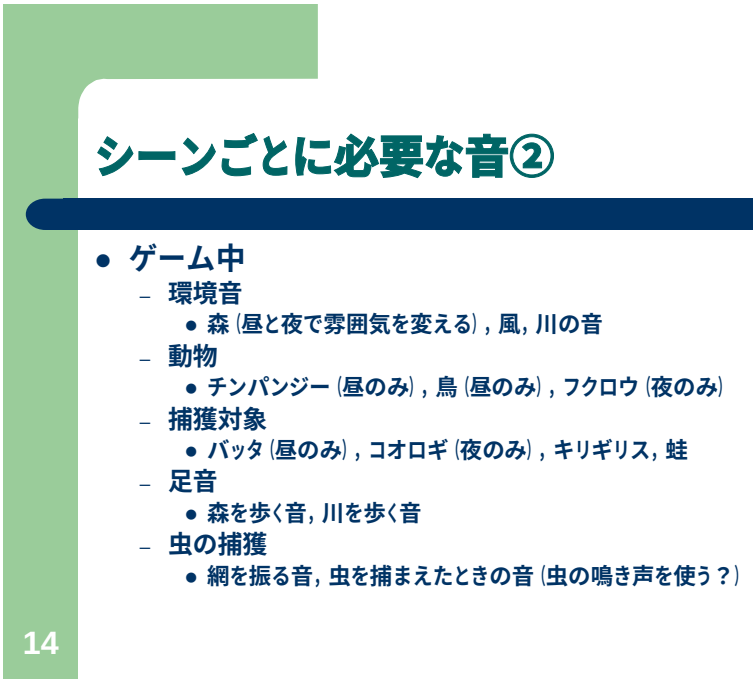


図 2-19　虫捕りゲーム企画書修正後（14 ページ）

3.4.2. 開発方針の確認

ミニゲームの開発ではいくつものゲームを作って，そこから面白そうな要素を抽出してひとつのゲームを作り上げるという予定だった．ミニゲームの開発で作られたゲームの中には面白さが期待される要素を含んだゲームはあるにはあったが，決定的なものではなかった．そこで方針を変更し，研究会同期の荒木氏が提案した数ある企画の中でメンバーが興味を持った虫捕りという要素をベースに開発することにした．

ミニゲームの開発を踏まえて，虫捕りゲームを開発する際には以下の点に留意した．

- 虫捕りのイメージを喚起するような良質の音を集めることを優先する
- 企画に詰まったら積極的に他人に相談し，魅力的な意見を取り入れていく

以上の開発方針を踏まえて，次のような実装手順で開発を行った．

1. 音環境の作成
   - 良質な音を収集する
   - 虫捕りを行う場所である森を作る
   - 森の中を散歩できるようにする
2. ゲームの骨格部分の作成
   - 虫捕りを行えるようにする

3.4.3. 音環境の作成

まずは，ゲームの要となる音を探すところから始めた．知り合いや大学の図書館から効果音を集めた CD を借りたり，クライアントである南雲氏に使えそうな音をもらったりした．音の中にはそのまま使えないものもあったので，フリーの音声加工ツールである SoundEngine（http://www.cycleof5th.com/download/）を利用して，適切な加工を施した．

音がある程度揃ったところで，いよいよコーディングに入った．音を鳴らす部分については，前回の「さうんど おんりぃ」プロジェクトで利用した SoundPlayer と SoundContainer というクラスがあったので，これを利用することで難なく実装できた．これらは，ミドルウェアである CRI Audio の中の CriAuPlayer をラップしたもので，音の再生や停止を管理している．BGM として森の雰囲気を出す音をループ再生しておき，その中で虫や動物の音が点在するようにした．同時に，デバッグのためにゲーム中のプレーヤーや動物などの位置関係が把握できるように簡単な描画が行うようにした．

次に行ったのはプレーヤーが散歩できる，という部分である．前回の「さうんど おんりぃ」ではプレーヤーは上キーを押したら上に，左キーを押したら左へ移動するというカニ歩きの移動方法を採用していた．しかし，今回はリアリティを追及するために，左右キー

を押したら自分の向きを変えるという移動方法を採用することにした.

プレーヤーの移動方法を決定したところで，次に音の聞こえ方について考えた．今回採用した移動方法では，プレーヤーが向きを変えるとそれに応じて音の聞こえ方を修正しなければならない．それを実現するために行った実装と数式を以下に記す.

1. プレーヤー（x，y）と，音を鳴らしている対象（x_1，y_1）との角度θを計算する
   $\theta = \mathrm{atan2}(x_1 - x,\ y_1 - y)$
   ※原点から引数に与えられた座標までの角度をラジアンで返すatan2関数を利用した.

2. プレーヤーと，音を鳴らしている対象との距離dを計算する
   $d^2 = (x_1 - x)^2 + (y_1 - y)^2$

3. プレーヤーが変えた向きαと距離dをもとに，対象の新座標（x_2，y_2）を計算する
   $x_2 = d * \cos(\theta + \alpha) + x$
   $y_2 = d * \sin(\theta + \alpha) + y$

4. 新座標を元に，各スピーカーに送る音の大きさを再計算して設定する
   ※この処理は，CRI Audio 側の関数を利用した

また，虫や動物などプレーヤーとの角度や距離によって音の聞こえ方が変わるものはBGMと分ける必要があると考え，MovableSoundというクラスを新たに作成した．この時点でのクラス図が図 2-20 である．GamePlayState がゲームそのものであり，音としてSoundMaterial と MovableSound を持っている．GamePlayState は MovableSound のmoveSound()メソッドを逐次呼び出し，その度に各オブジェクトがプレーヤーとの距離や角度を再計算する．ここで再計算された情報は再生中の音にも適用されるので，音の定位や音量がリアルタイムで変化し続けるのである.

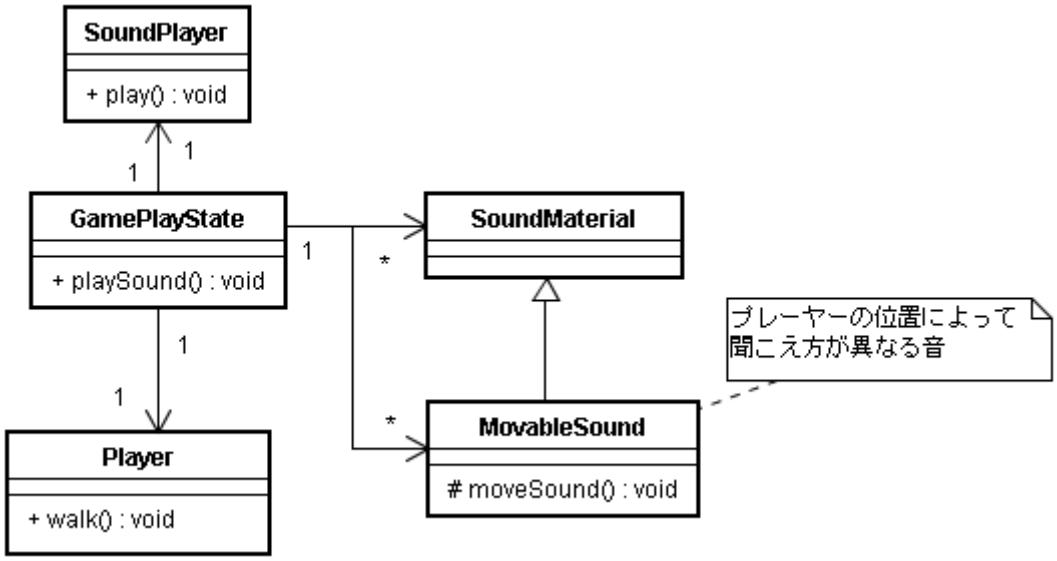


図 2-20　音環境作成時のクラス図

最後に，音量について述べておく．まずは，表 2-1 を見てもらいたい．この表はプレーヤーと音を鳴らす対象物の距離と音量を表している．青い線が改善前の音量の計算式を，赤い線が改善後の音量の計算式をもとに描いた距離と音量の関係である．

- 改善前の式：(音量) = 1 / sqrt(距離) * 2
- 改善後の式：(音量) = －sqrt(距離) / 10 + 2

※sqrt は平方根を表す

※音量の単位はリニアスケールである（1.0 が本来の音量）

グラフを見ると分かるように，改善前の式では距離が離れているうちは音量も緩やかに大きくなっていくが，距離が 50 を切ると音量は急激に大きくなる．これは，自然界で実際に聞こえる距離と音の関係を数式化したものである．しかし，実際にこの式を用いてゲームを行ってみると音量の変化が乏しく，距離が近づいても判断することが非常に困難であることが分かった．そのため，我々はゲームにふさわしい音量の計算式を改善した．それが改善後の数式である．

改善後の数式では，改善前の数式に比べて距離が離れていても音量が変化する様子が分かるようになった．また，距離が一定以上開くと音が完全に消えるため，音の種類や数が増えても煩雑さが減って聞き取りやすくなった．

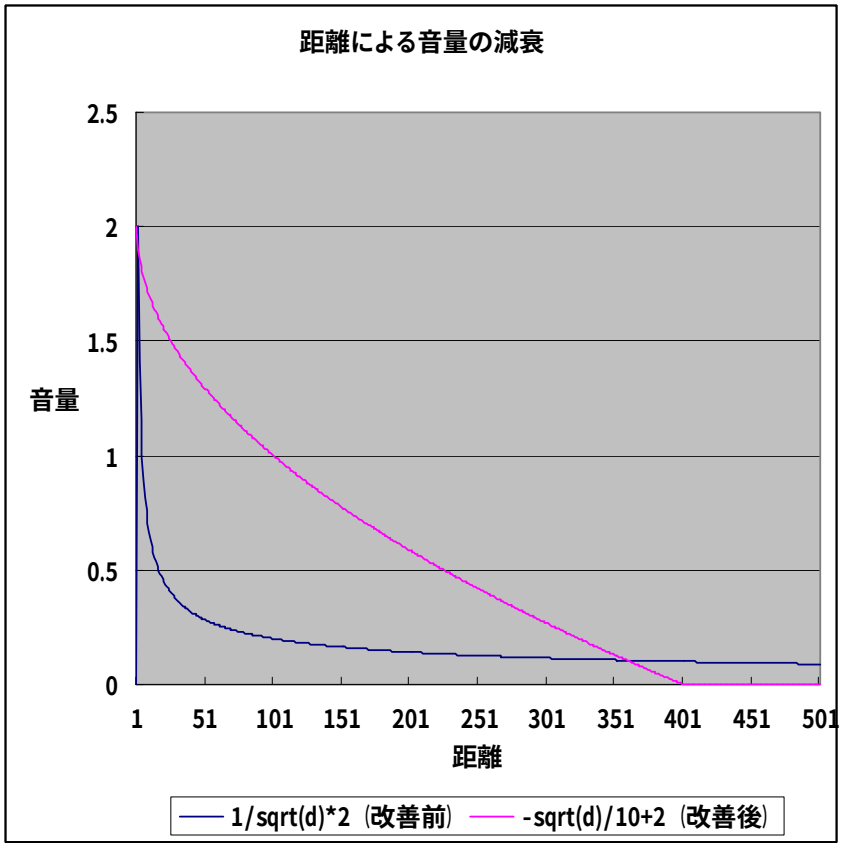


表 2-1 距離による音量の減衰

3.4.4. ゲームの骨格部分の作成

虫を捕獲する機能を追加する前に，MovableSound クラスを継承して，Creature クラスを作った．さらに Creature クラスを継承して Insect クラスを作った．これは，ゲーム中に存在するオブジェクトの持つ役割を明確に分けるためである．ここで，現在の音オブジェクトについて整理しておく．

- SoundMaterial…ゲーム中に存在する音オブジェクトすべてを表す
- MovableSound…プレーヤーの座標によって音の聞こえ方が変わるもの（川・泉など）
- Creature…MovableSound のうち，自ら移動するもの
- Insect…Creature のうち，プレーヤーが捕まえることが出来るもの

この時点で，クラス図は図 2-22 のようになった．

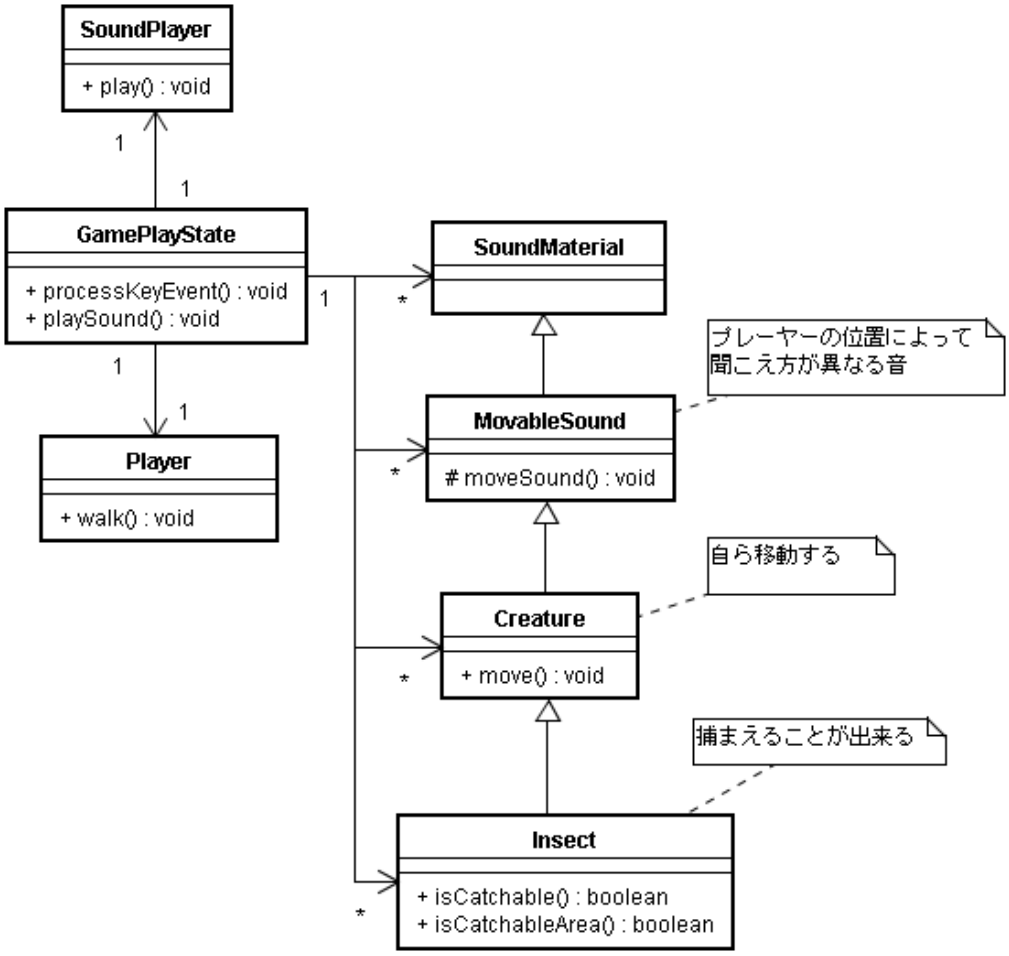


図 2-22　ゲームの骨格部分作成時のクラス図

この設計を元に，プレーヤーが網を振ると，GamePlayState が Player と Insect の距離を計算して，一定の範囲内に捕まえられる虫がいれば捕まえる，という実装を行った．

3.5. α版評価（ユーザレビュー）

ゲームの骨格部分が完成した時点で，慶応義塾大学の中根様にプレイしていただき，レビューをいただいた．実際に視覚にハンディキャップを有する方の視点から，プレイする上での不具合や不足点，要望や意見等をいただき，ゲームの開発に反映させることで，視覚にハンディキャップを有する方々により快適に遊んでいただけるような作品にしていきたいと考えた．

中根様からは，いくつかの非常に貴重なご意見をいただくことができた．具体的には以下のような点である．

- ゲームのフィールド全体が，どのくらいの大きさか分からない．

中根様にユーザレビューを依頼した時点の虫捕りゲームでは，プレーヤーは際限なくどこまでも移動できるようになっていた．そのため，プレーヤーが遠くに移動しすぎると虫が全くいない状態になってしまい，戻る道もわからず迷子になってしまう．

そこで，移動できる範囲を定めて，移動できる範囲の限界まで移動したら，障害物などによってそれ以上移動できないことがわかるようにするとよいのではないかという意見をいただいた．

- 森の中を歩いていても，何にもぶつからない．

中根様にユーザレビューを依頼した時点の虫捕りゲームでは，木や川などの障害物は実装されていたが，障害物と同じ座標までプレーヤーが移動しても，ぶつかる音や，水の中に入る音などはならないようになっていた．そのため，実際に森の中を散歩しているような臨場感が得られず，不自然さが残ってしまっていた．

そこで，木にぶつかる音や，草を掻き分ける音などを追加すれば，よりリアルに森の中を散歩している状況が再現でき，面白さも増すのではないかという意見をいただいた．

- 足音がない

中根様にユーザレビューを依頼した時点の虫捕りゲームでは，プレーヤーが歩いても足音がしないようになっていた．そのため，自分が移動しているということがわかりづらく，位置感覚がつかみにくいという問題があった．

そこで，プレーヤーが歩いているときは足音を鳴らすようにすれば，プレーヤーが移動していることがわかりやすいし，位置感覚も把握しやすくなるのではないかという意見をいただいた．

- 川が不自然である．

中根様にユーザレビューを依頼した時点の虫捕りゲームでは，川の音が一点からしか聞こえず，泉のように聞こえてしまっていた．また，プレーヤーが川の上を通り過ぎても，何も音がしないので，不自然であるという問題があった．

そこで，川の音が一点から聞こえるのではなく，線上から聞こえるようにした方がよいという意見をいただいた．また，プレーヤーが川の上を移動したときには，なんらかの音を鳴らす，あるいは川の上は移動できないようにするとよいのではないかという意見をいただいた．

- 虫が自動的にうごく

中根様にユーザレビューを依頼した時点の虫捕りゲームでは，虫は一定の場所から動かず，止まったままになっていた．虫は動かなくても，最初からそういうものだと思ってプ

レイすれば不自然には感じないが，多少動く虫がいた方が，自然の虫捕りに近くなるのではないかという意見をいただいた．

- ただ虫を捕り続けるというだけでは飽きる

中根様にユーザレビューを依頼した時点の虫捕りゲームでは，ひたすら虫を捕り続け，ゲームの終了などもなかった．そこで，ゲームの終了条件や，虫を捕る上で目標となるようなスコアを付けた方がより楽しめ，飽きもこないのではないかという意見をいただいた．また，虫捕り以外のイベントも取り入れると面白いのではないかという意見をいただいた．例えば，虫捕りをする前に虫捕り網を探すイベントを追加したり，ゲームが進むにつれて虫捕り以外のことができるようにしたりするといった要素を取り入れることで，飽きることなく，長時間プレイできるゲームになるのではないかという意見をいただいた．

- 説明音声が足りない

中根様にユーザレビューを依頼した時点の虫捕りゲームでは，ゲームのルールや，操作方法の説明等の音声が一切入っておらず，ゲームのルールや操作方法がわかりづらいといった問題があった．そこで，ゲームのルールや，操作方法の説明を音声で付けた方よいという意見をいただいた．

音声については，人間の声で録音した方がいいが，あまり感情を込めたり抑揚を付けたりしない方がよいのではないかとのことだった．アクションゲームや RPG では，感情を込めた音声の方が，臨場感が出るが，虫捕りゲームの場合は，自然の音によるリアリティを求めた方がよいので，音声にはあまり抑揚は付けなくてもよいのではないかという意見をいただいた．

また，合成音声を使いたい場合は，ペンタックスが開発しているライブラリが，現行では一番クオリティが高いとのアドバイスもいただいた．

中根様からいただいた，これらのアドバイスを参考にして，虫捕りゲームに改良を加えより面白く，遊びやすいものにすることを目指した．またこのユーザレビューをもって，α版の開発を終了とした．

第4章. β版開発

4.1. β版実装

α版の開発を引き継いで，β版の開発を行った．主にユーザインタビューの反映とゲームとして面白くするためのギミックの追加という視点から，次のような項目の実装を行った．

1. ユーザインタビューの反映
   - 足音の実装
   - 川の音の修正
2. バグの修正
   - 虫を捕る方法の変更
   - 音の鳴らし方の変更
   - 音ファイルの読み込みにおけるバグの修正
   - 実行速度が遅い問題の解決
3. ゲームとして面白くするためのギミックの追加
   - 制限時間やスコアなどを追加する
   - 虫が簡単に捕まえられないようにする
   - 音声によって，ゲーム全体を分かりやすくする
   - タイトルやチュートリアルなどを追加する

以下では，各項目について実装の経緯や悩んだ点について説明していく．

4.1.1. ユーザインタビューの反映

ここでは，ユーザインタビューで得られた評価のうち，特に優先順位が高いと判断した 2 項目について実装・修正を行った．

- 足音の実装

これまでは，プレーヤーはただ移動するだけだったが，自分たちでゲームをプレイしてみると確かに動いたことがわからないので距離が掴みづらい．そこで，一定距離を動くごとに足音をつけることにした．

効果音 CD から足音の音を探して移動中はループ再生するようにしたが，不自然さが残っていた．その理由を考えていると，足音がずっと同じ音であることが原因になっていることが分かったので，左右の足で違う足音を用意して，歩くたびに左右の足が入れ替わるといった実装にした．すると，以前に比べて自然な足音らしく聞こえるようになった．

また，森を歩くときと川を歩くときで音を分けた．森を歩くときは落ち葉を踏んだときの音，川を歩くときは水辺を歩く音を使うことで，ゲームに臨場感が出た．

- 川の音の修正

次に川の実装であるが，今までの実装では川は点（0 次元）で表されていい．しかし，ユーザインタビューで指摘があったとおり元来川は線（１次元）である．これを解決するためには，川の端から端まで至るところで川が流れる音が同時に聞こえる必要がある．しかし，音を複数同時に再生するためにはどのように行えばよいかというライブラリ依存の技術的な問題があったため，ライブラリの提供元である CRI・ミドルウェア社に質問のメールを送った．

その結果，複数の SoundPlayer を作成することで同時再生が可能だということが分かったので，早速川における音の聞こえ方を修正した．結果として，川はそれらしく聞こえるようにはなったが，近づくと複数の川の音が干渉し合い，音が濁ってしまうという不具合が発生した．この問題については現在技術調査中であり，未解決となっている．

4.1.2. デバッグ・修正およびリファクタリング１

ユーザインタビューで得られた評価を反映した後に，一度バグの修正とリファクタリングを行うことした．これは，パソコンによって実行速度が著しく異なるためである．まず，全体のリファクタリングを行ってソースの状況を整理して，その後実行速度を落としている影響を調べるという順序で作業を行うことにした．

リファクタリングした項目はいくつかあるが，その中でも大きな変更を加えたものを記しておく．

- 虫を捕る方法の変更

従来は，GamePlayState 内で虫を捕まえるキーが押された場合，直接虫に対して捕まえられるかを調べて，捕獲範囲内にいれば捕まえるという方法を取っていた（図 3-1 参照）．しかし，プレーヤーを介さずに直接虫を捕まえると，将来虫かごを実装する場合に拡張性がなく，また意味的にも不自然である．そのため，GamePlayState 内で捕獲範囲内にいる虫をプレーヤーに知らせて，プレーヤーが虫を捕まえるように変更した（図 3-2 参照）．

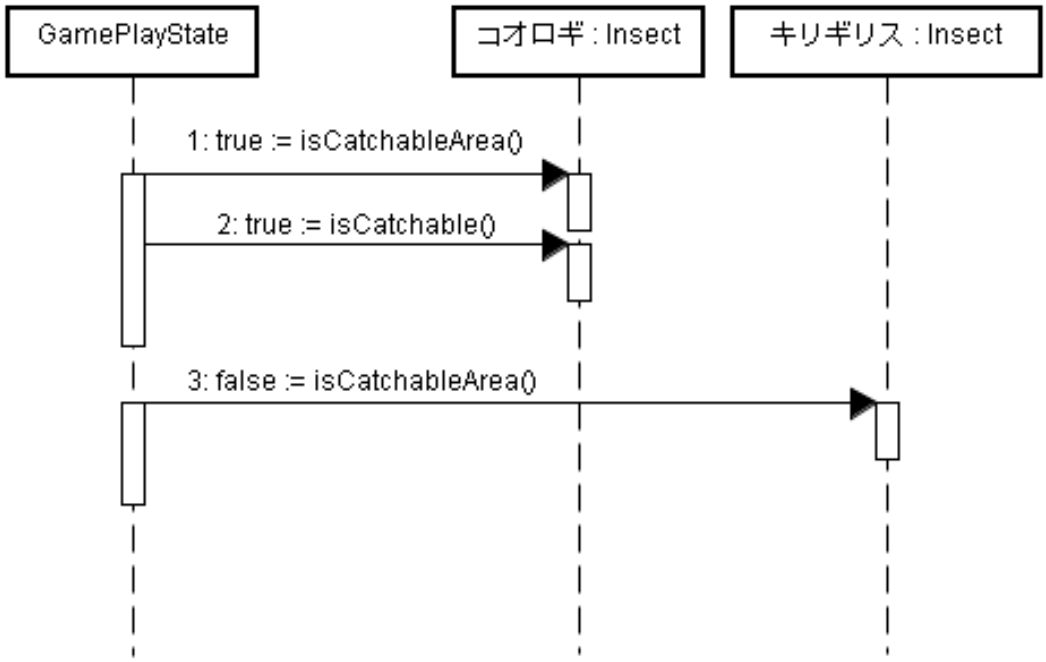


図 3-1：虫捕り変更前のシーケンス図

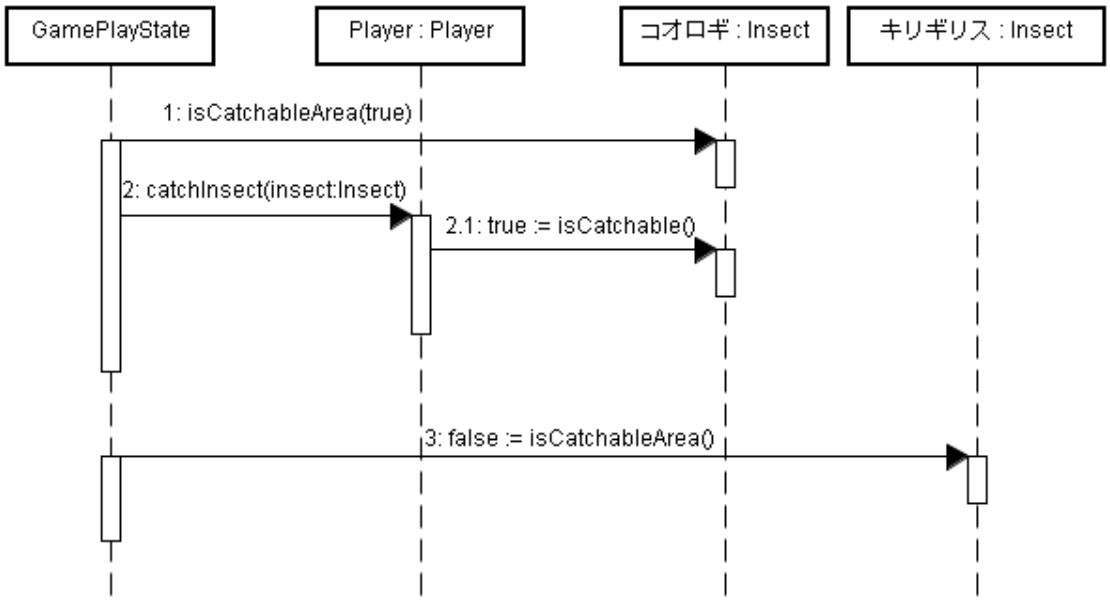


図 3-2：虫捕り変更後のシーケンス図

- 音の鳴らし方の変更

音を鳴らす際，今までは GamePlayState が SoundPlayer を持っており，BGM や虫，動物などすべての音を鳴らすようにしていた（図 3-3 参照）．しかし，GamePlayState の責任が大きくなってしまい，また GamePlayState 自体も肥大化していたため，虫や動物については自分で SoundPlayer を持ち，オブジェクトごとに音を鳴らすように変更した．

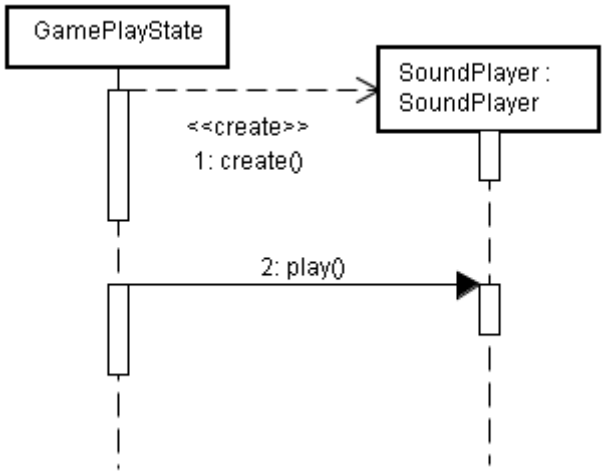


図 3-3：音関係リファクタリング前のシーケンス図

まず，GamePlayState で音が鳴るオブジェクトを GamePlayStateSoundFactory で作成する．このオブジェクトはすべて SoundMaterial のサブクラスであり，SoundPlayer を所持している．GamePlayState は逐次，自分の持つ SoundMaterial のリストに対して音を鳴らす play メソッドを呼び，必要に応じて音を鳴らすようにした．音関係のリファクタリングを行った後のクラス図，シーケンス図はそれぞれ以下の通りである（図 3-4, 3-5 を参照）．

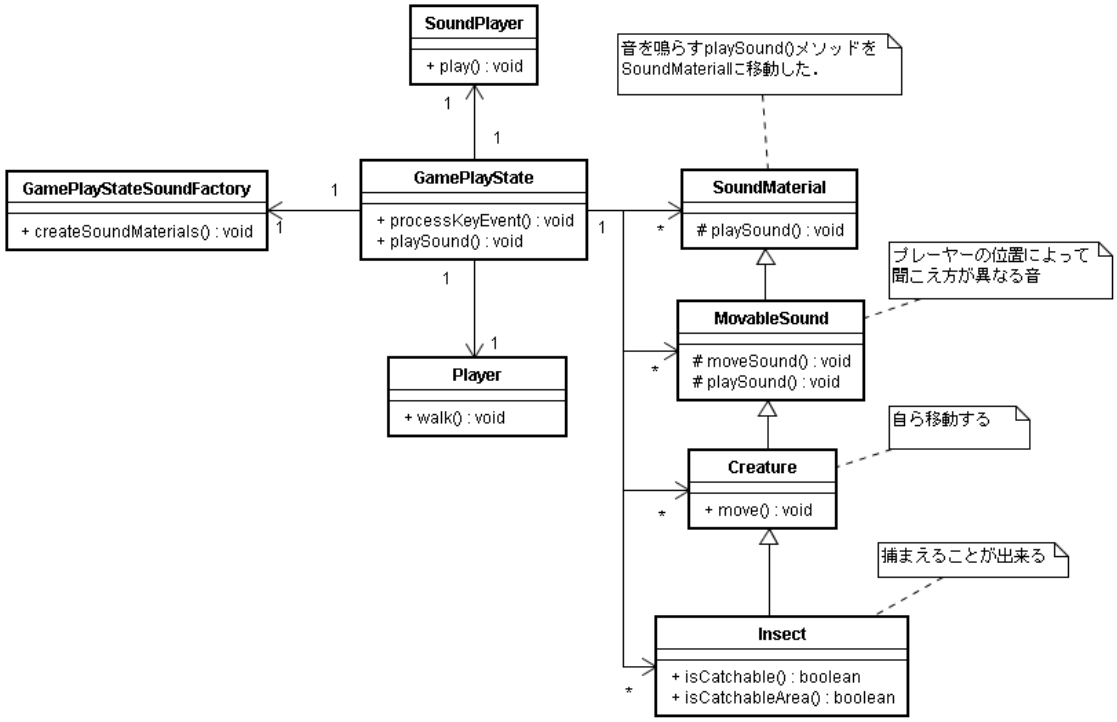


図 3-4：音関係リファクタリング後のクラス図

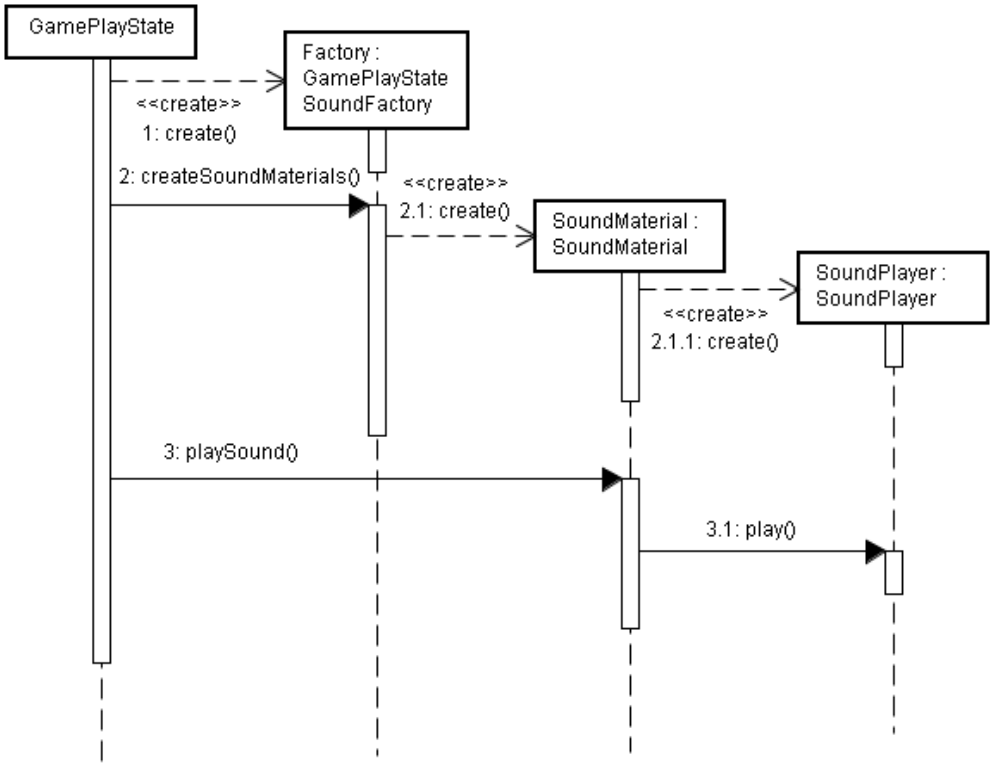


図 3-5：音関係リファクタリング後のシーケンス図

- 音ファイルの読み込みにおけるバグの修正

これまでの実装で多数の音を利用するようになったが，1）csb ファイルの容量が大きすぎると音が再生されなくなる，2）複数の csb ファイルを読み込もうとすると実行時にエラーが出る，という問題が生じた．これについては，直接 CRI・ミドルウェア社に質問し，回答をもらうことができた．

前者の問題はヒープを確保する際のバッファサイズが初期設定（3MB 程度）のままだったのが原因だった．実際にゲームで利用していた ForestWalking.csb は 7MB を超えていたので，このバッファサイズを増やすことで音を再生することに成功した．

また，後者の問題については CriAuObj をラップした関数である SoundContainer の呼び出し方に問題があることが分かった．そのため，一度現在のライブラリの利用状況とラップしている関数の関係性を整理した．

虫捕りゲームでは，音の再生に CRI Audio ライブラリを利用している．まず， CRI AudioCRAFT というデータ作成ツールを利用してゲーム内で使用する wav ファイルをキューシートバイナリ（csb）という形式に変換する（図 3-6 参照）．

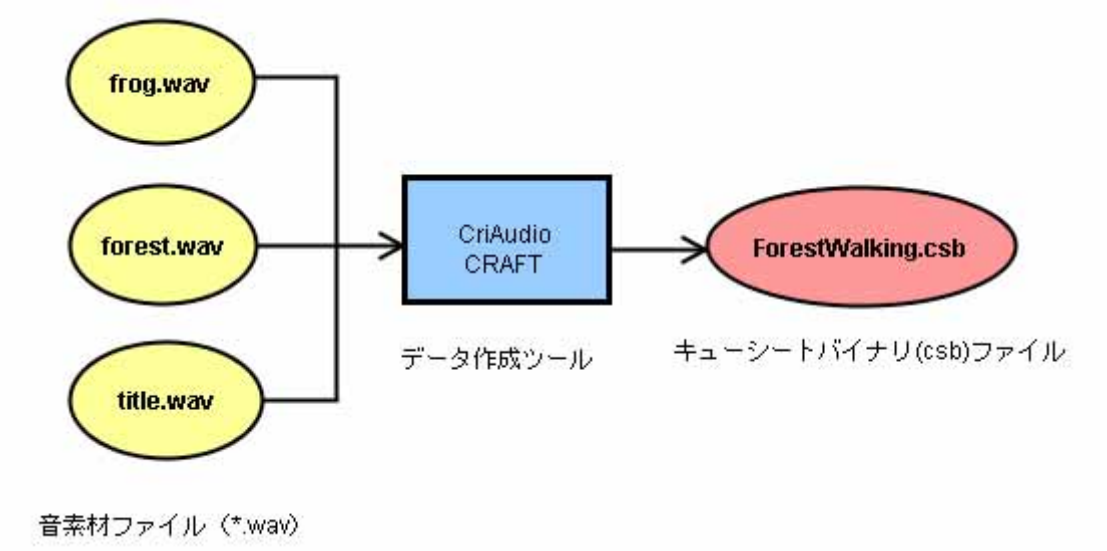


図 3-6：CRI Audio ワークフロー

プログラム内では，CriAuObject という音の再生をコントロールするオブジェクトを作成し，そこにキューシートバイナリから読み込んだデータを登録することで音の再生が可能となる．さらに，CriAuPlayer というサブクラスを利用することによって，再生中にボリュームや定位を変更するなどといった細かい制御ができるようになる．

虫捕りゲームでは，CriAuObj のほかにキューシートバイナリをロードする関数を実装してある CriAuCueSheet やサウンドを出力するための関数を実装してある CriSoundRendererBasic などをラップした SoundContainer というクラスを用意してあ

る．また，SoundPlayer が，文字列と CriAuPlayer をマッピングした連想配列である SoundShelf を所持している．この SoundPlayer が初期化される際，引数として SoundContainer を受け取り，その中でキューシートバイナリがロードされて，再生や停止が行えるようになるのである．なお，連想配列に読み込む音の役割と名前は外部ファイルに記述してある．虫捕りゲーム（ForestWalking）と CRI Audio ライブラリの利用関係を図 3-7 に図示した．

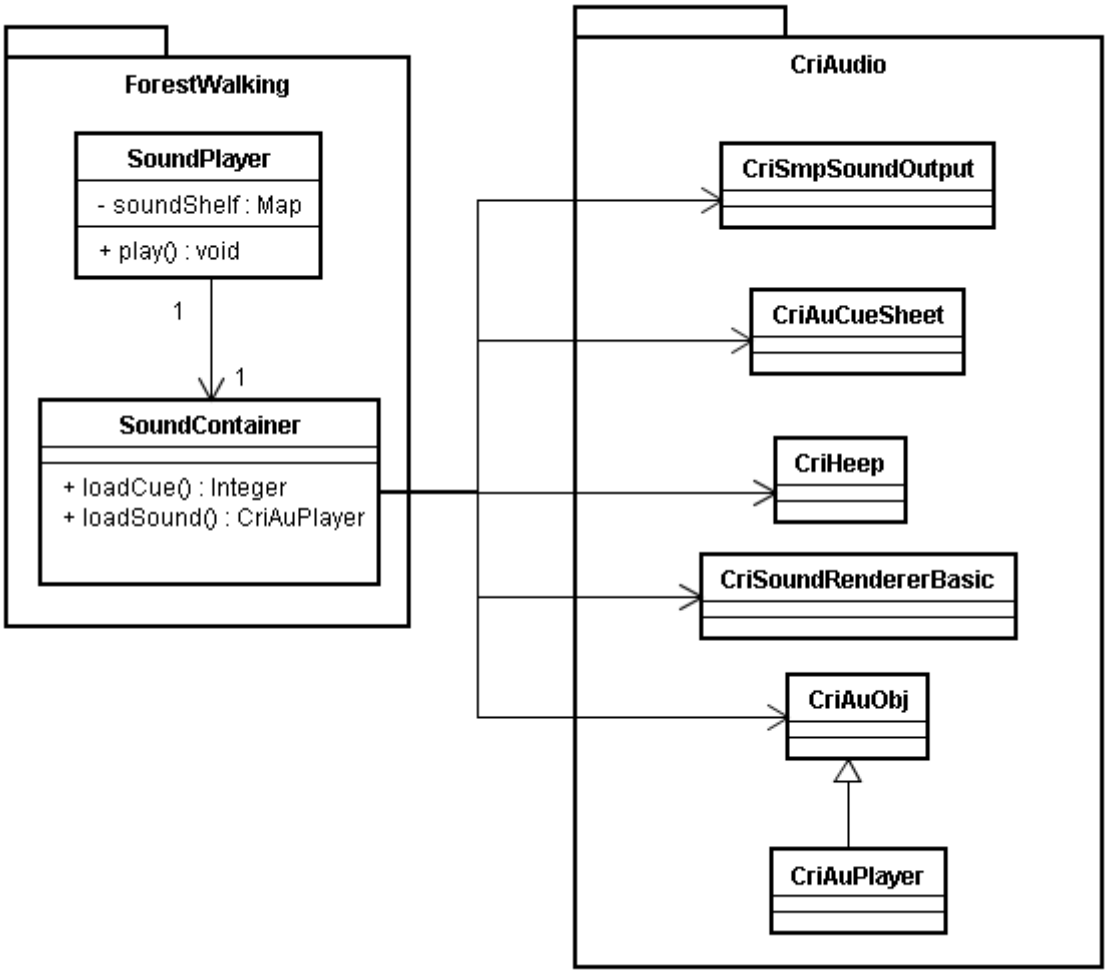


図 3-7：音関連ライブラリとの関係を表すクラス図

また，音の登録から再生・停止の手順を具体的に説明すると以下のようになる．

1. csb ファイルを作成する

CRI AudioCRAFT を使用して，キューシートとして frog.wav が登録されている Sound.csb を作成する．

2. 音の役割とファイル名を外部ファイルに記述する

ゲーム中での役割を表す名前が sound_of_frog（蛙の鳴き声）とするため，外部ファイル（sound.txt とする）に，【sound_of_frog frog】と半角スペース区切りで記述する．

3. csb ファイルのパスを引数にして，SoundContainer を作成する

```
soundContainer = new SoundContainer("Sound.csb");
```

SoundContainer の初期化の際, csb ファイルのロードやキューシートの登録が行われる．

4. SoundContainer と外部ファイルのパスを引数として，SoundPlayer を作成する

```
soundPlayer = new SoundPlayer(soundContainer, "Sound.txt");
```

SoundPlayer の初期化の際，Sound.txt を解析して，SoundShelf という連想配列に格納する. ここでは，frog という名前の音ファイルの再生を管理する CriAuPlayer が作成され，sound_of_frog という文字列とマッピングされる．

5. 役割の名前を引数として，SoundPlayer の play()関数を呼び出す

ここでは，SoundPlayer->play(“sound_of_frog”)と記述する．すると，SoundShelf の中から sound_of_frog という名前にマッピングされている frog の CriAuPlayer を探す．CriAuPlayer が見つかった場合，その play()関数を呼び出すことで，frog を再生する．

複数の csb ファイルを読み込もうとすると実行時にエラーが出る, という問題については SoundContainer を作成する際に，サウンド出力を行う CriSmpSoundOutput を複数作成しているためにリソースの確保が正しく行えていないということが原因であった．そのため，複数の csb ファイルを読み込む場合はサウンド出力の初期化を 1 回行い，キューシートハンドルの作成とキューシートロードを複数行い, 1 つのオーディオオブジェクトに対して複数アタッチするという方法に変更することで問題を解決した．

- 実行速度が遅い問題の解決

既存のバグとして最後に残ったのが，実行速度が遅いという問題である．マシンスペックにある程度影響されるのだが，Pentium3（933MHz）・メモリ 512MB のパソコンで正常に動作しないのは問題があると考えた．デバッガを利用してどの関数が動作を重くする原因となっているのかを調べた．その結果，デバッグ用に文字を描画している関数が動作を重くしている原因だということが分かった．

虫捕りゲームでは，文字を描画する際に SDL（Simple DirectMedia Layer）というライブラリを利用している．これはゲームなどのマルチメディア関連のソフトウェアを開発するための，グラフィックやサウンド等の API を提供するライブラリである．このライブラリの中では文字を作るときに以下のようなプログラムを書く．

```
1  SDL_Color textColor = {255, 255, 255}; // 文字の色を設定する（ここでは白）
2  TTF_Font *font = TTF_OpenFont("Headache.ttf", 28); // フォントを設定する
3  string textString = "Test"; // 描画する文字列
4  SDL_Surface *text = TTF_RenderText_Blended(font, textString.c_str(), textColor);
   // フォントと文字列から描画する文字を作成する
5  drawText(10, 10, text, mainScreen); // 文字を描画する
6  TTF_CloseFont(font); // フォントを解放する
7  SDL_FreeSurface(text); // テキストが占有していたメモリを解放する
```

虫捕りゲームでは，この文字作成を簡易化するために createText()という関数を作成してその中で 1~4 行目の処理を行っていた．しかしこの関数では文字列の作成は行うものの，その後に行わなければならないメモリの解放を行っていなかった．これは，5 行目の文字の描画を行う前に 7 行目のメモリ解放を行ってしまうと，本来描画するべき文字が描画できなくなってしまうためである．

関数化したことで文字の作成と描画を分けてしまったため，文字を描画するたびにメモリが解放されずにどんどんメモリリークが起こるという状態になっていたのが，実行速度低下の原因であった．この問題を解決するために，createText()関数は廃止し，文字を描画するときは必ずメモリの解放を行っているかを確認することにした．

4.1.3. ゲームを面白くするためのギミックの追加

- 制限時間とスコアの追加

虫を捕まえることができても，具体的な目標やゴールがないと，飽きるのが早くなってしまうし，プレイする気も起きなくなってきてしまう．そこで，ゲームに制限時間とスコアを追加し，ゲームを遊ぶ人が目標をもってプレイできるようにした．

ゲーム開始時は昼から始まり，一定の時間が経過すると，昼から夜になる．夜になると登場する虫や BGM が変化する．夜になってから，さらに一定時間が経過すると，ゲーム終了となり，最後に捕まえた虫の数と，スコアが読み上げられる．ゲームを遊ぶ人は，より多くの虫を捕まえ，高いスコアを獲得することを目標にプレイすることで，よりゲームに対する熱中度が向上する．

まず，時間帯の変更という要素を追加するために，今までの設計を拡張した（図 3-8 参照）．具体的には，GamePlayState の親クラスとして State クラスを，子クラスとして AftenoonState と NightState を追加した．AftenoonState は昼に登場する音を管理し，NightState は夜に登場する音を管理する．GamePlayState では，昼夜問わず登場する音を管理する．また，それぞれの時間帯が持つ音を作成するクラスとして SoundFactory を作成した．

時間帯の切り替えを明確にするために，昼から夜に切り替わる際には昼のみに登場する動物や昆虫の鳴き声をフェードアウトさせた．さらに，プレーヤーの声と鐘の音，カラスの鳴き声などで夕方ということを演出した．また，夜にはすべての音をフェードアウトさせ，プレーヤーの声でゲーム終了が近づいていることを喚起することで，ゲームが終了することを分かりやすくした．昼と夜の切り替えの方法については次項の「タイトルやチュートリアルを追加する」で説明する．

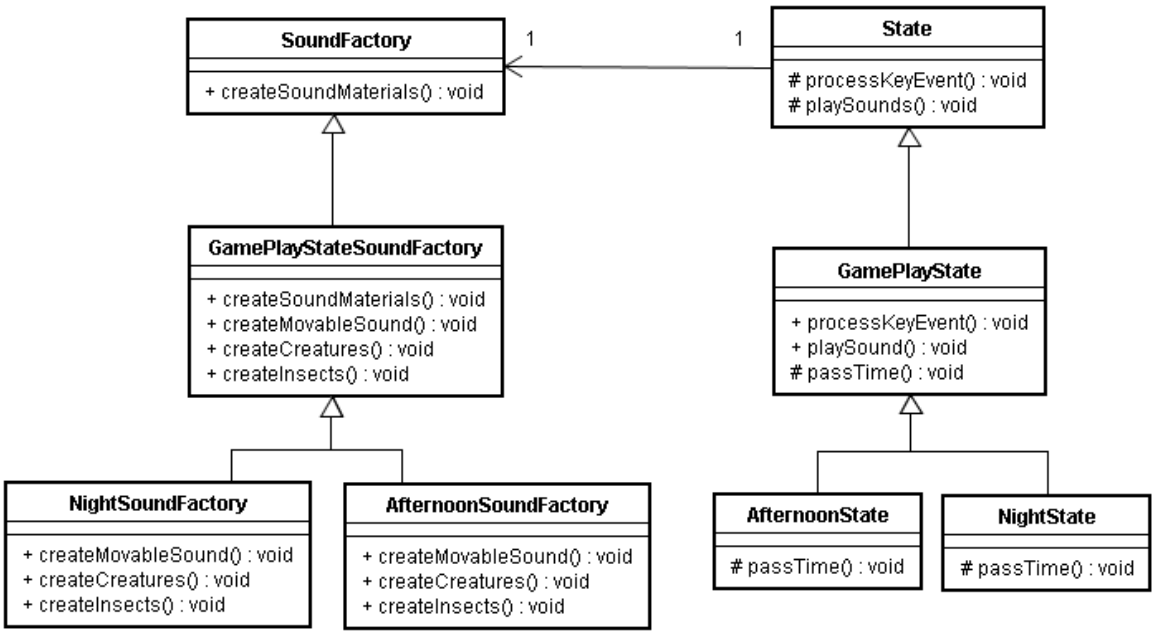


図 3-8：時間帯要素を追加したクラス図

次にスコア機能を実装するために，ResultState を追加した．捕らえた虫を虫かごに入れておくことで，ResultState でその虫かごに入っている虫の名前と数を読み上げるようにした．また，虫ごとにスコアを設定することで，合計スコアを発表するようにした．スコアの読み上げについては，現在は 0～10 までの音声を組み合わせて対応しているが，イントネーションや各桁のつながりが不自然である．これについては，各桁の数字を読み上げた音声を用意すれば解決できそうであるが，これはまだ未実装である．

（例）765 という数字を読むとき

現状：1～10 までしか音声がないので「なな，ろく，ご」と読む

改善後：１～9，10～90（10 ずつ），100～900（100 ずつ）の計 30 種類の音声を用意することで，「ななひゃく，ろくじゅう，ご」と読むことができる

- タイトルやチュートリアルなどを追加する

従来の虫捕りゲームは，説明や練習が無く突然ゲームが始まってしまうため，操作方法がわからなかったり，虫の鳴き声が聞き分けられなかったりすることが多い．そのため，ゲームの操作方法を把握しゲームに慣れてもらうために，ゲームの解説や練習モードを追加した．ゲームの解説では，ゲームの目的や操作方法や登場する虫の解説等をする．練習モードは，虫を一匹だけ登場させその虫を捕まえることでゲームの雰囲気を知ってもらう初心者のためのモードである．解説と練習モードがあることで，ゲームを遊ぶ人は操作方法を理解することができる．またゲームに慣れることができ，スムーズにゲームの本編に入ることができるようになる．

前項の制限時間やスコアを追加する際にゲームの状態を追加したので，一度状態遷移図を書くことで全体を整理した（図 3-9 参照）．

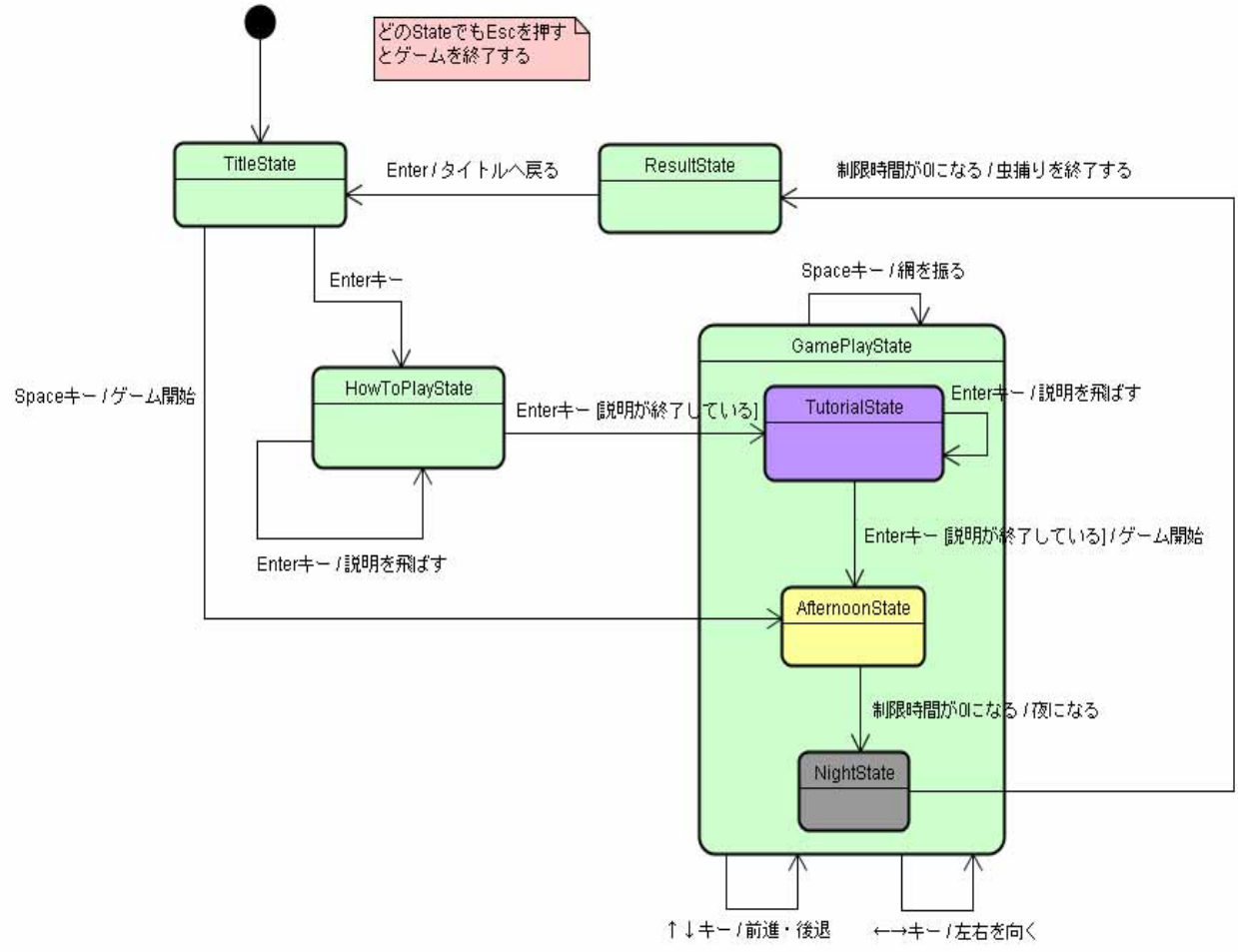


図 3-9：状態遷移図（ゲームの流れ）

この状態遷移図に従い，状態を遷移させるために State パターンを適用した．これは

State が増えることによって if 文が増えることを避けるためである．例えばある状態から別の状態へ遷移するときは以下のようなコードを書く．

```
changeState(new AfternoonState(soundContainer, mainScreen)); // 昼に遷移する
```

changeState()はすべての状態の親クラスである State クラスが持っている関数で，引数に遷移先の状態のインスタンスを受け取る．ここでは，AfternoonState に遷移させたいので AfternoonState クラスのインスタンスを作って渡している．State クラスを拡張したクラス図については図 3-10 を参照してほしい．

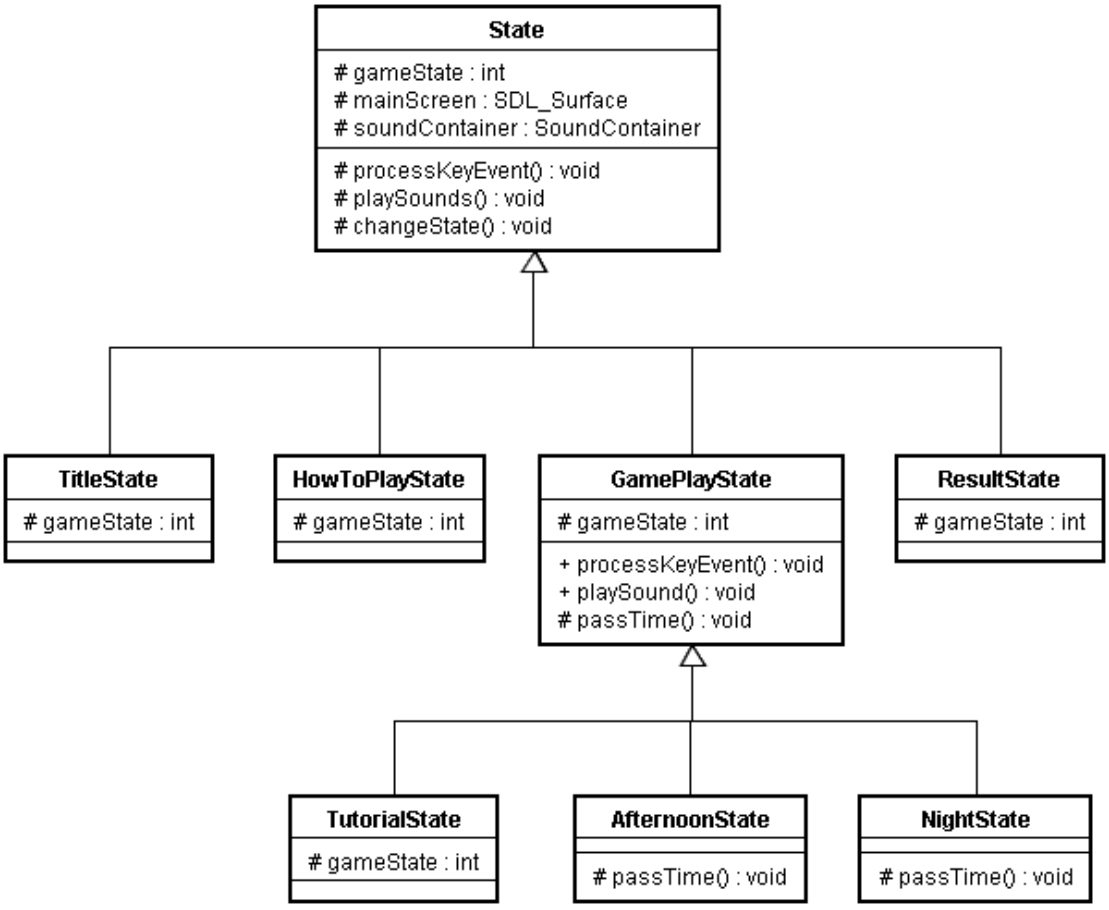


図 3-10：State パターン適用後のクラス図

- 虫が簡単に捕まえられないようにする

今までは，ある程度虫に近づいて網を振るだけで確実に虫が捕まえられた．しかし，あまり簡単に虫を捕まえることができてしまうと，張り合いがなくなり，ゲームとしても単調になってしまい，すぐに飽きてしまう．そのため，簡単には虫を捕まえることができないようにするために，虫を捕まえようとしても逃げるようにした．

プレーヤーが虫を捕まえることができる範囲にいたとしても，虫から遠ければ捕まえようとしても虫が逃げる可能性が高くなる．逆に，虫の近くに居るときほど，虫を捕まえようとしたときに，捕まえられる可能性が高くなる．このとき虫を捕まえることが出来る確率の計算には以下の式を使っている．

```
int probability = 100 - distance; // 距離から確率を計算する
```

ここでは，distance はプレーヤーと虫との Y 座標の差である．100 以上ドット離れていれば probability の値は 0 となり，近づくにつれて propbability の値が大きくなっていく．probability が 0～100 の範囲にない場合は 0 とする．こうして計算された probability を 0～100 までの間で発生させた乱数と比較することで，虫を捕まえられるかどうかを計算している．また，虫を捕まえるのに失敗したときは，虫はプレーヤーの左，左上，上，右上，右のいずれかの方向にランダムで逃げる．

- プレーヤーが移動できる範囲を限定する

今までは，プレーヤーは際限なくマップ上を移動できてしまい，あまり遠くまで行ってしまうと全く虫や川などの音が聞こえなくなってしまい，迷子になってしまうことがあった．そこで，プレーヤーが移動できる範囲を制限し，虫や川などの音が聞こえる範囲内のみ移動できるようにした．

移動できる範囲は，Config.txt で指定できる．Config.txt の ariaSize に 800～3000 の間の数値を記述することで，その範囲が移動できる範囲となる．この範囲を超えて移動しようとすると，虎の鳴き声とアナウンスが入り，プレーヤーが先に進めないようになっている．

また，マップ上のオブジェクトも，移動できる範囲外に生成されることが内容にしているが，その際にある問題点が発生した．プレーヤーが移動する際には，プレーヤー自体を動かすのが一般的であろう（図 3-11 参照）．

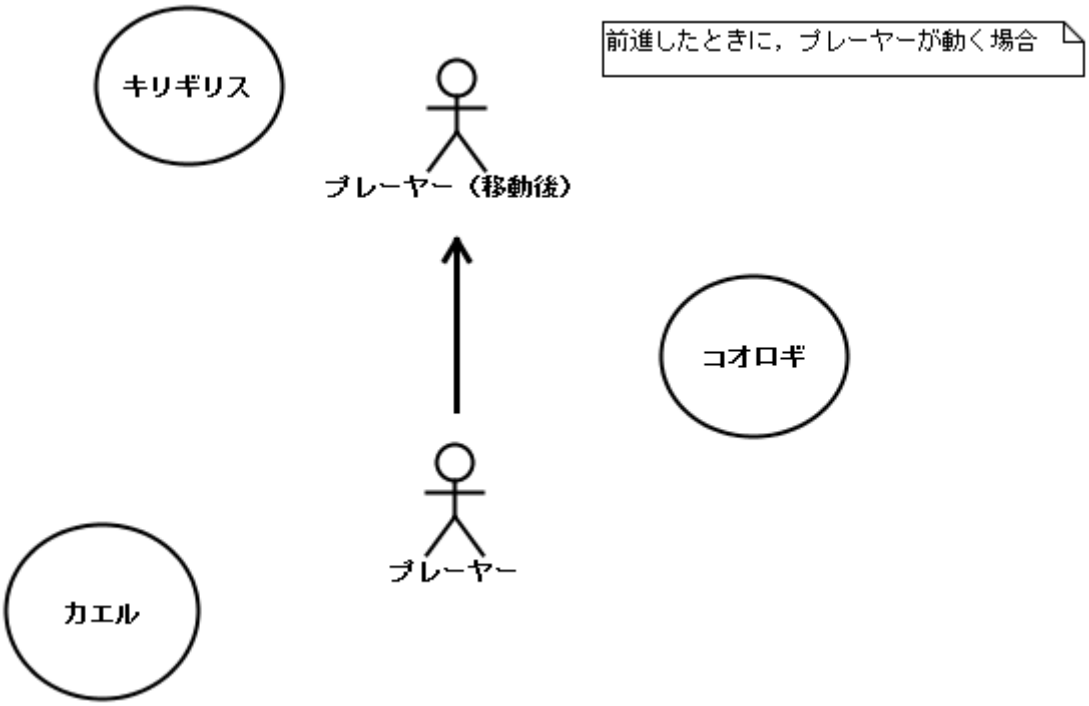


図 3-11：前進する際にプレーヤーが動く場合のイメージ

しかし，虫捕りゲームでは，プレーヤー自体は移動させず，プレーヤー以外の全てのオブジェクトを動かすことで，プレーヤーが動いているように見せている（図 3-12 参照）.

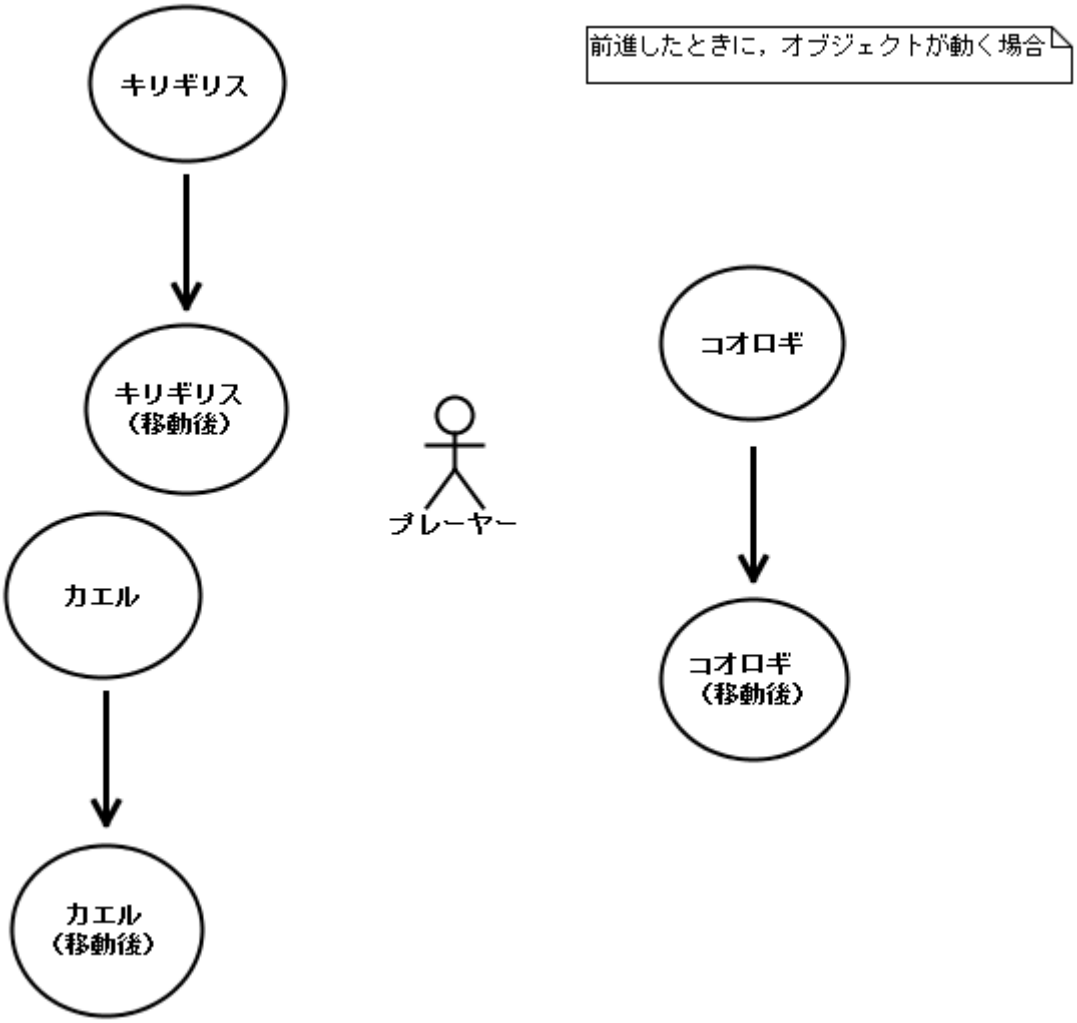


図 3-12：前進した際に虫が動く場合のイメージ

そのようにした理由は，プレーヤー自体が動いてしまうと，プレーヤーから見たオブジェクトの位置関係や角度等の特定が難しくなってしまい，位置関係に応じて音の音量や定位を調節するのが難しくなってしまうためである．

しかし，虫が生成される時の虫の座標は，画面の中心の座標を基準にして決定されるようになっていた．そのため，プレーヤーが昼の間に移動可能範囲ぎりぎりのところまで移動していた場合，夜に移り変わるときに，プレーヤーの位置ではなく，現在の画面の中心を基準に夜の虫の座標が決まってしまうので，プレーヤーが移動できる範囲を超えて虫が生成されてしまっていた．この問題を解決するために，夜に虫が生成される際には，プレーヤーの移動距離を減算して虫の座標を決定するように改良した．そうすることにより，プレーヤーが移動可能範囲ぎりぎりに移動していた場合でも，移動範囲を超えて虫が生成されてしまうという問題は発生しなくなった．

- 音の充実化

ゲームに使われている音や音声の種類を増やしたり，現状の音を洗練させたりすることで，より臨場感あふれる音環境を作成することができると考えた．そのために，現在ゲーム内で使われている音を表にまとめて，音の名前と目的などを記述した．またセリフの場合はその内容も併せて記述した（表 3-1 参照）．

この表を作ることで，後で他の人に音だけを発注するといったことができるようになったと同時に，ゲーム内で使われているすべての音と役割が一目瞭然となったので，利用できる音ファイルの名前を探すといった時間の浪費を節約することができた．

表 3-1：ゲーム中で使用する音のリスト

No.	カテゴリ	ファイル名	説明
1	背景音	forest1	森
2		forest2	森
3		forest3	森
4		forest4	森
5		forest5	森
6		forest6	森
7		night	夜の森
8		bell	寺の鐘
9		crow	カラス（昼→夜を表す）
10			
11	効果音	river	川
12		wind	風が吹く
13			
14	プレーヤーの動作	step_leaves_left	左足で歩く
15		step_leaves_right	右足で歩く
16		step_water_left	左足で歩く（水の中）
17		step_water_right	右足で歩く（水の中）
18		swing	網を振る
19			
20	プレーヤーの声	yell1	網を振るとき
21		yell2	網を振るとき
22		catch_frog	蛙を捕まえたとき
23		catch_grasshopper	キリギリスを捕まえたとき
24		catch_cricket	コオロギを捕まえたとき
25		catch_mole_cricket	オケラを捕まえたとき

26		fail_in_catch1	失敗したとき
27		fail_in_catch2	失敗したとき
28		fail_in_catch3	失敗したとき
29		cannot_go_forward	移動可能エリアの限界に来たとき
30		find_net	虫捕り網を見つけたとき
31		start_collect_insects	虫捕りを始めるときのセリフ
32		get_dark	昼→夜になったときのセリフ
33		walk_home	虫捕りを終わるときのセリフ
34			
48	TitleState 関連	title	ゲームタイトル
49		how_to_start	ゲームの始め方
50			
51	HowToPlayState 関連	explanation_of_skip	説明をスキップできることを知らせる
52		explaination_of_goal	目的の説明
53		how_to_play	操作説明
54		explaination_of_insects	昆虫の鳴き声の説明
55		sound_of_frog	カエルの声の説明
56		sound_of_grasshopper	バッタの声の説明
57		sound_of_cricket	コオロギの声の説明
58		sound_of_mole_cricket	オケラの声の説明
59		go_tutorial	チュートリアルへ行くことを知らせる
60			
61	TutorialState 関連	tutorial	移動のチュートリアル
62		end_of_tutorial	ゲームを開始することを知らせる
63			

64	ResultState 関連	go_title	タイトルへ行くことを知らせる
65		0	数字を読み上げる音
66		1	数字を読み上げる音
67		2	数字を読み上げる音
68		3	数字を読み上げる音
69		4	数字を読み上げる音
70		5	数字を読み上げる音
71		6	数字を読み上げる音
72		7	数字を読み上げる音
73		8	数字を読み上げる音
74		9	数字を読み上げる音
75		10	数字を読み上げる音
76		20	数字を読み上げる音
77		30	数字を読み上げる音
78		40	数字を読み上げる音
79		50	数字を読み上げる音
80		60	数字を読み上げる音
81		70	数字を読み上げる音
82		80	数字を読み上げる音
83		90	数字を読み上げる音
84		100	数字を読み上げる音
85		200	数字を読み上げる音
86		300	数字を読み上げる音
87		400	数字を読み上げる音
88		500	数字を読み上げる音
89		600	数字を読み上げる音
90		700	数字を読み上げる音
91		800	数字を読み上げる音
92		900	数字を読み上げる音
93		result	結果を知らせる音声
94		pre_tag_of_score	結果を知らせる音声
95		post_tag_of_score	結果を知らせる音声
96		number_of_frogs	蛙の数を知らせる音声
97		number_of_grasshoppers	キリギリスの数を知らせる音声
98		number_of_crickets	コオロギの数を知らせる音声
99		number_of_mole_crickets	オケラの数を知らせる音声

表 3-1：ゲーム中で使用する音のリスト（続き）

No.	備考
1	
2	現在未使用
3	現在未使用
4	現在未使用
5	現在未使用
6	現在未使用
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	「えいっ」
21	「えいっ」
22	「蛙をつかまえた」
23	「キリギリスをつかまえた」
24	「コオロギをつかまえた」
25	「オケラをつかまえた」
26	「逃げられた」
27	「あ゛ー」
28	「もう少し近づかなくちゃ」
29	「これ以上進むのは危険そうだ，引き返そう」
30	虫捕り網を見つけるというギミックを実装した後で
31	「さあ，がんばってたくさん捕まえるぞ！」
32	「大分暗くなってきたな」

```
33 「夜も更けてきたし，そろそろ帰ろう」
34
48 ゲーム名「ForestWalking」を入れる
   「ゲームの説明を聞く場合は，enter キーを押してください．
49 説明を聞かずにゲームを始める場合は，space キーを押してください」
50
51 「ゲームの説明を開始します．説明をスキップしたい方は enter を押してください」
   「初めにゲームの目的を説明します．あなたは昼間の森の中を歩いています．
   聞こえてくる様々な音から虫の鳴き声を探し出して，虫を捕まえてください．
   夜になるまでにできるだけたくさんの虫を捕まえましょう．
52 捕まえた虫の種類と数に応じて得点が計算されますので，高得点を目指してください．」
   「次に操作方法を説明します．上カーソルは前進，下カーソルは後退です．
   左右のカーソルでは自分の向いている向きを変えることができます．
   Space キーを押すと虫捕り網を振ります．虫を捕まえるためには上下左右の
   カーソルを使って出来るだけ虫に近づいて網を振ってください．
53 また，Esc キーを押すとゲームを終了します．」
   「続いて，捕まえられる虫について説明します．
   捕まえられるのはキリギリス，コオロギ，オケラ，カエルです．
54 それぞれの鳴き声を聞いてみましょう．」
55 「これはカエルの鳴き声です」
56 「これはキリギリスの鳴き声です」
57 「これはコオロギの鳴き声です」
58 「これはオケラの鳴き声です」
   「最後に，ゲームを始める前に少し練習してみましょう．
59 練習を始めるには Enter を押してください．」
60
   「虫の鳴き声を頼りに上下左右のカーソルを使って移動しましょう．
   虫が近くにいると思ったら space キーで網を振ってください．
   ただし，虫に十分近づいていないと虫が逃げてしまうので注意してください．
61 それでは練習してみましょう．」
   「虫の捕まえ方は分かりましたか？　以上で練習は終わりです．
   実際のゲームを開始するには enter キーを押してください．
62 もう一度説明を聞きたい場合は，space キーを押してください．」
63
64 「enter キーを押すと，タイトルに戻ります．」
65 数字は自然に聞こえるようにしたい
```

```
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93  「結果発表です」
94  「スコアは」
95  「～です」
96  「蛙を捕まえた数」
97  「キリギリスを捕まえた数」
98  「コオロギを捕まえた数」
99  「捕まえたオケラの数」となっているので，「オケラを捕まえた数」に直したい
```

4.1.4. デバッグ・修正およびリファクタリング２

横浜市立盲学校の生徒達や大岩研究室関係者，クライアントへのユーザレビュー（詳細は 3-2 節を参照）を通じて得られた意見をもとに，納品前に最後のデバッグ・修正およびリファクタリングを行った．またそのとき，何がタスクとして残っていてどの優先度が高いかを明確化するために，デバッグ時 TODO リスト作成した（表 3-2 参照）．

機能追加・バグ修正・リファクタリングの 3 つのカテゴリを作り，それぞれのタスクと重要度を書き込んでいった．また，そのタスクの着手状況や優先度がわかるように，未修正のもので重要度が高いものは赤，未修正のもので重要度が中程度のもの，または修正中や保留は黄，修正済みのものは青という色分けを行った．

表 3-2 デバッグ時 TODO リスト

作業カテゴリ	作業内容	状態	優先度
機能追加	練習モードを作成する	修正済み	高
	説明をつける	修正済み	高
	音の高低差をつける	未修正	低
	音の種類を増やす	修正済み	中
	虫取り網を探す	未修正	低
	森の一定範囲外に出られないようにする	未修正	高
	デバッグモードをオフにする機能をつける	修正済み	中
	スコアの読み上げを自然にする	修正済み	中
	スコアの評価をつける（がんばりました，もっとがんばりましょう等）	未修正	中
	ステージ編成を追加する	未修正	中
バグ修正	Result がおかしいバグを修正する	修正済み	高
	フェードイン･フェードアウトを修正する	未修正	中
	KeyRelease を実装する	修正済み	高
	川の音の聞こえ方を修正する	未修正	高
	一定範囲外からはみ出てしまうのを修正する	修正済み	高
	夜になると一定範囲外に虫ができてしまうのを修正する	修正済み	高
	昼，夜のコンストラクタが重い（new するときが原因？）	未修正	高
リファクタリング	外部ファイルからの読み込み（制限時間，エリアサイズなど）	修正済み	中
	マジックナンバーの整理	修正中	中
	SoundPlayer に fade 系をもたせる	保留	中
	Factory の修正	修正済み	高
	KeyEventListener で Exit は親に書く	修正済み	中
	esc を押したときの exit(0)をやめる．	保留	高
	include の整理	修正済み	中
	移動の際，Player を動かしたい	修正済み	中
	音声読み込みのテキストの書式を分かりやすくする	未修正	低
	外部テキストから，虫の数などを読み込む	未修正	中

- ResultState のバグの修正

ResultState ではスコアの読み上げなどの結果発表を担当しているが，このときに読み上げの声がおかしくなってしまうというバグがあった．これは読み上げの際に，C++標準ライブラリの list を使っていた. しかし iterator の初期化に失敗しているようで，デバッグではうまく動くが，バイナリにすると動かなくなってしまう．このエラーの原因は未だ分かっていないが, 暫定的な対処として list の内容を一度配列に読み込んで, iterator ではなく普通の for 文を使うことにした.

- Release 環境で動作しないバグについて

本来，VisualStudio で開発したプログラムを配布するときは，コンパイル時にデバッグ版ではなくリリース版を選択するべきである．しかし，リリース版としてコンパイルすると実行時に「フォントが開けない」というエラーが出てしまう．エラーの原因について Web などを調べてみたが，どうしても原因が特定できなかったため，現在配布してあるものはデバッグ版でコンパイルしてある（デバッグ版とリリース版で動作に差異などは生じていないことを確認してある）.

- ヘッダーファイルのインクルード関係の整理

C++では，あるクラスで必要となるクラスは，ヘッダーファイルをインクルードする必要があるが，ヘッダーファイルを相互にインクルードし合ってしまうと，インクルード関係が循環してしまい，コンパイルエラーとなってしまう．この問題は，相互インクルードの関係にあるヘッダーファイルそれぞれに，直接#include の記述がある場合と，あるヘッダーファイルをインクルードした別のヘッダーファイルを介して，相互インクルードになってしまう場合がある．前者に関しては，問題となる箇所が発見しやすく，対処も容易に施せるが，後者に関しては，問題箇所を特定するために複数のインクルードファイルを点検する必要があり，インクルード関係が複雑になっているほど，問題箇所の特定と対処に手間がかかる.

そこで，このような問題が起こりにくいよう，また，起きた場合にも容易に対処できるよう，インクルード関係がなるべく簡潔になるように整理した．以下が，整理する前のヘッダーファイル間のインクルード関係を示した図である（図 3-13 参照）.

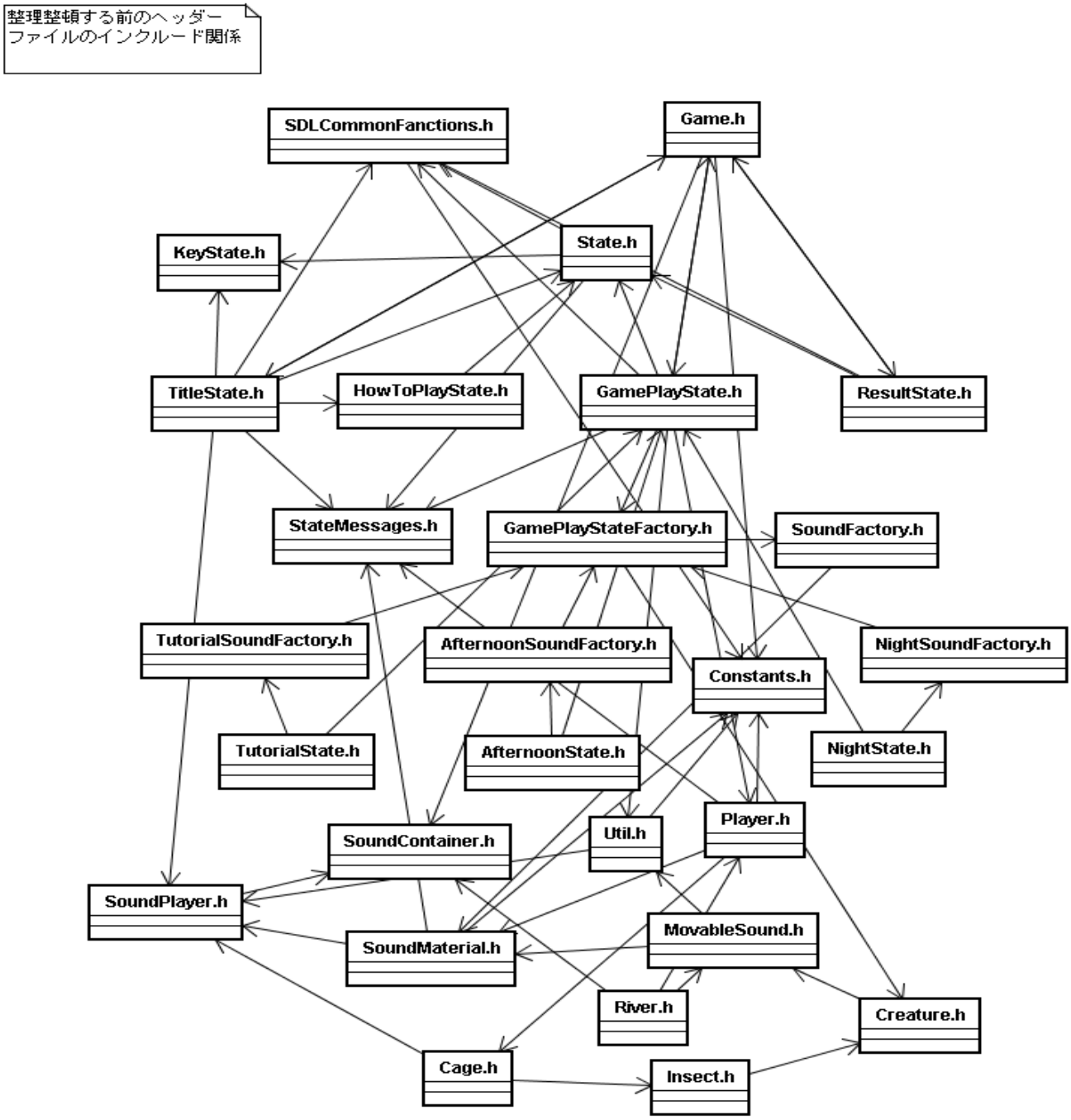


図 3-13：整頓前のインクルードファイル相関図

この図では，あるヘッダーファイルが，そこから伸びている矢印の向いている先のヘッダーファイルをインクルードしていることを示している．この図を一見してわかる通り，ヘッダーファイル間のインクルード関係が非常に複雑になっている．このような状態では，上記のような相互インクルード問題が発生した際に，問題箇所の特定と対処が非常に困難になると予想される．

そこで，ヘッダーファイル間のインクルード関係を簡潔に整理した．以下が整理後のヘッダーファイル間のインクルード関係を示した図である（図 3-14 参照）．

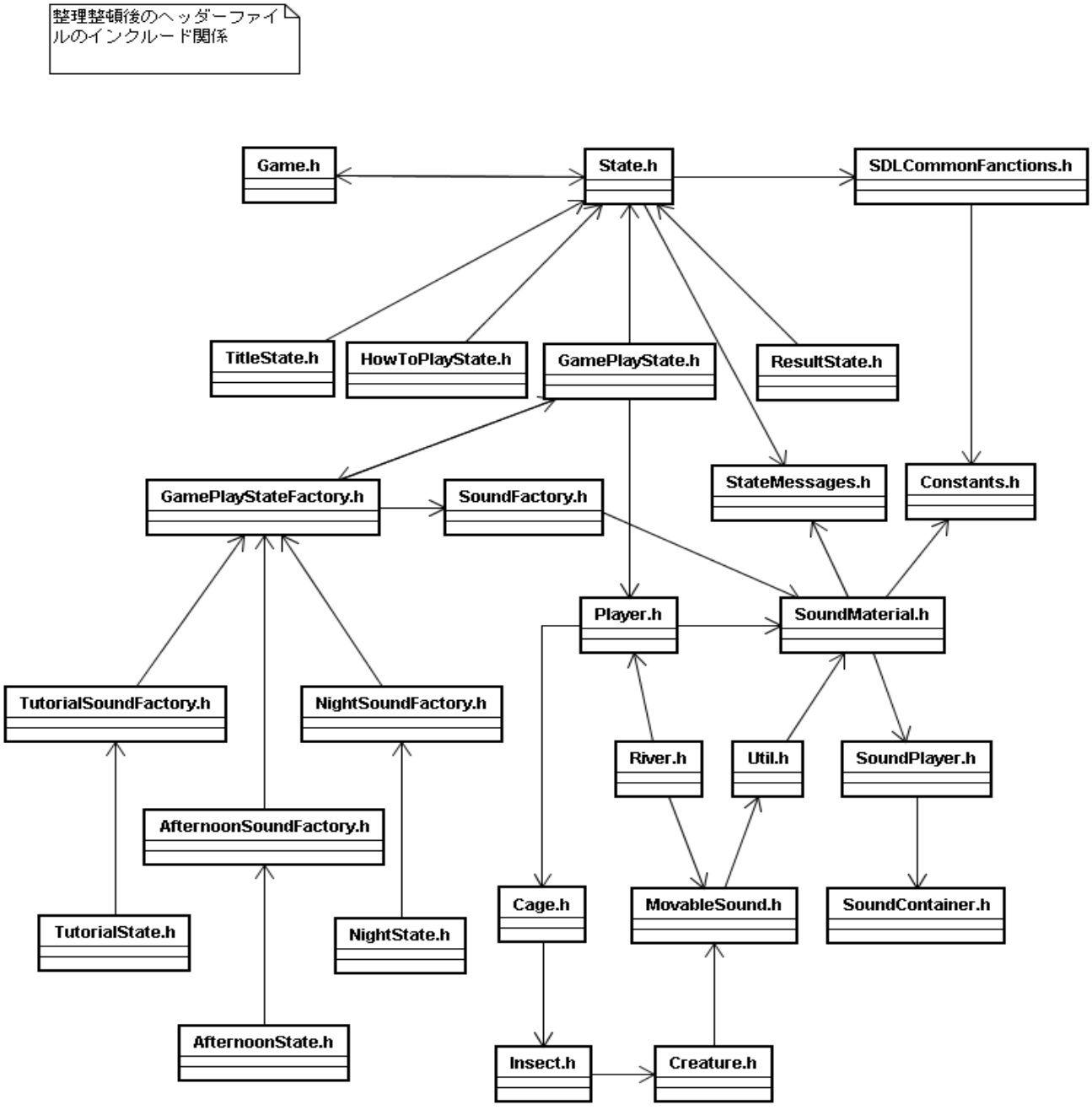


図 3-14：整頓後のインクルードファイル相関図

整理する前のインクルード関係と比べて，非常に簡潔にまとまっているのがおわかりいただけると思う．このくらい簡潔であれば，相互インクルード問題が発生した際に問題箇所の特定と対処が比較的楽に行えるだろう．また，今後新しいヘッダーファ

イルが追加されたり，ヘッダーファイル間のインクルード関係に変更が生じたりする場合にも，上記のインクルード関係を参照すれば，不要なインクルードを行うことなく，インクルード関係を簡潔に保ったまま，追加や変更ができるようになるだろう．

- 定数の管理について

従来のソースコードでは各クラスに定数が散逸していたため，ゲームの設定を変更するときに関係している定数を探しながら値を変更する必要があった．これはデバッグ時に相当な労力がかかることで，これらの定数を 1 つのファイルにまとめてしまえば，わざわざ他のファイルを開く必要がなくなると考えて，定数だけをまとめたヘッダである Constants.h を用意した．

しかし，Constants.h はその性質から様々なクラスにインクルードされるため，値を書き換えるとインクルードしているすべてのクラスをコンパイルしなければならず，結果的にコンパイルに多くの時間がかかるようになってしまった．この問題を解決するために，Constants.h の中に記述されている定数のうち特に変更回数が多い「制限時間」「マップの広さ」については，外部ファイルから読み込むことで動的に変更できるようにした．これによって，コンパイルにかかる時間が圧倒的に短縮されて，デバッグがスムーズに行えるようになった．

- CriAuPlayer の仕様と，読み上げ時の状態遷移について

CRI Audio では, CriAuPlayer に音ファイルのキューをセットして音を再生するが，そのときの CriAuPlayer の状態遷移は図 3-15 のようになっている．

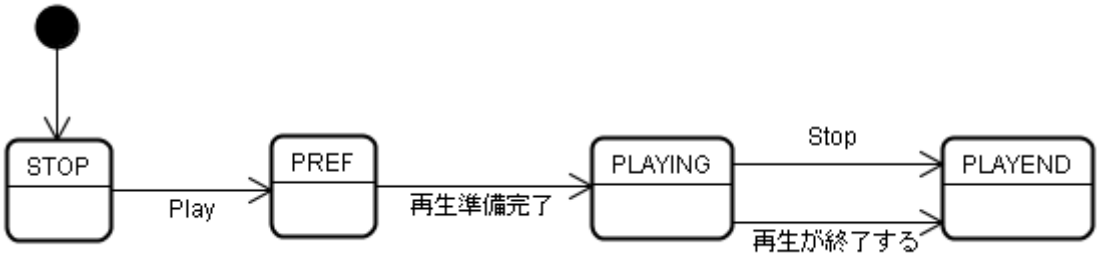


図 3-15：CriAuPlayer の状態遷移図

この状態遷移図を見ると分かるように，音の再生が終了すると CriAuPlayer は PLAYEND 状態になる．例えば，虫の数を表す時に“７”という音を再生するときは以下のような順序になる．

1. “７”がセットされた CriAuPlayer は，再生前なので STOP である
2. CriAuPlayer が STOP ならば,“７”を再生する

3. “７”の再生が終わると，CriAuPlayer は PLAYEND という状態になる

２の処理は，ゲーム内で音を鳴らしている関数である playSounds は while ループの中で逐一呼ばれているため，1 度再生している間は同じ音は再生しないようにするために必要である．しかし，このままでは再生前と再生後の状態が異なるため，同じ音を 2 回目以降に再生しようすると，2 の条件に合致せずに音が再生されない．この状況を改善するためには CriAuPlayer が PLAYEND かどうかを調べる必要がある．上記と同じ例を用いると同じ音を 2 回以上鳴らす場合は以下のようになる．

1. “７”がセットされた CriAuPlayer は，再生後なので PLAYEND である
2. CriAuPlayer が STOP か PLAYEND ならば，“７”を再生する
3. “７”の再生が終わると，CriAuPlayer は PLAYEND という状態になる

２回目以降でも音が再生するように，2 の条件を変えた．しかし，ここで新たな問題が発生した．この場合では，PLAYEND になった直後に 2 の条件に合致するため，同じ音が意図しなくても何度も再生されてしまうのである．このため，同じ音を 2 回以上鳴らし，さらに何度も再生しないようにするためには，CriAuPlayer の状態とは別に再生状態を持つ必要が出てくる．そこで，SoundPlayerState という状態を追加して，再生状態の管理を行うことにした．例として，ResultState の読み上げ時の状態遷移を挙げる．まず，下図を見て頂きたい．

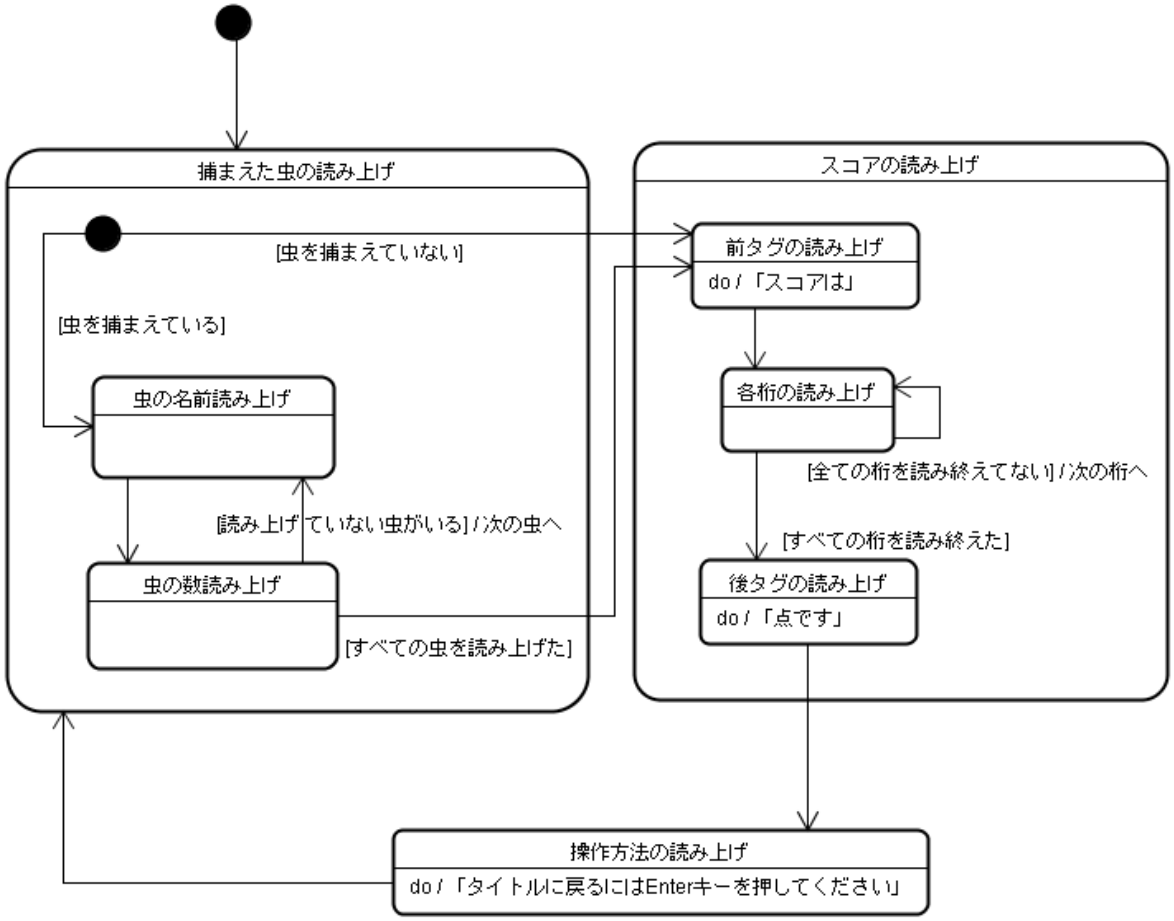


図 3-16：ResultState の状態遷移図

図 3-16 は ResultState における読み上げの状態遷移を表したものである. この中で, 捕まえた虫の名前・数を読み上げる際とスコアを読み上げる際に SoundPlayerState が必要になる. それぞれの場合における SoundPlayerState の状態遷移を表したものが図 3-17，3-18 である.

図 3-17：SoundPlayerState の状態遷移図（虫の名前読み上げ時）

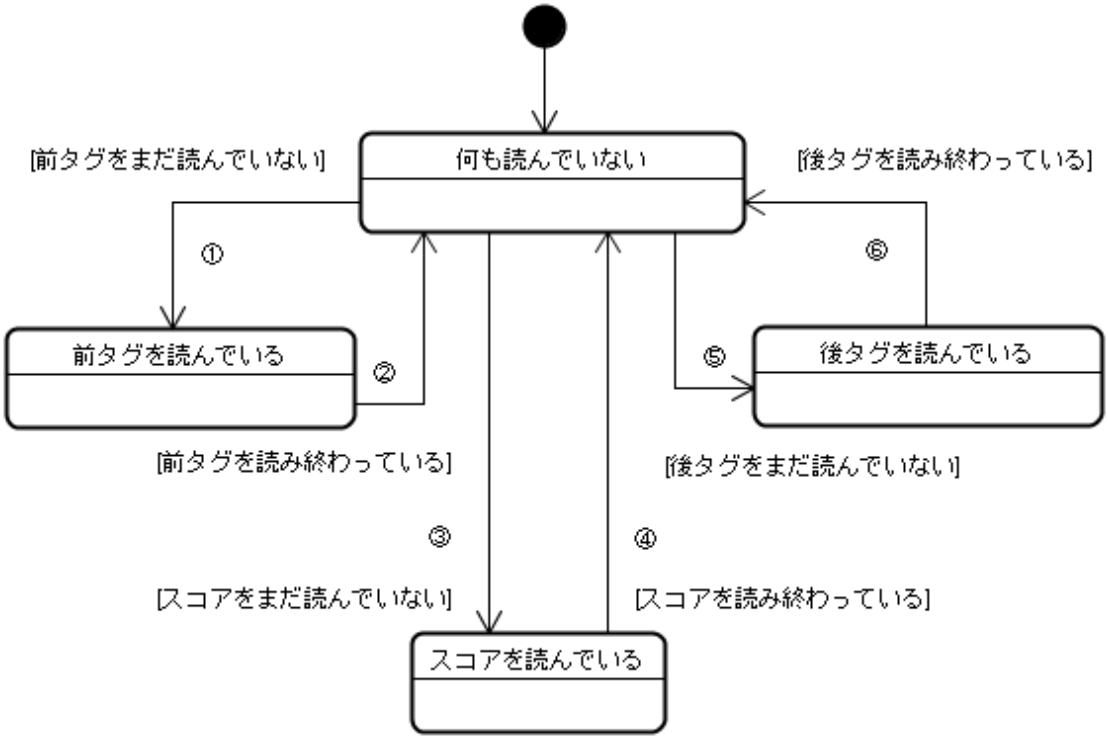


図 3-18：SoundPlayerState の状態遷移図（スコア読み上げ時）

SoundPlayerState を用いて音の再生状態を管理することで，音を再生するときの順序は以下のように改善された．もう一度同じ例を用いて説明する．

1. “７”がセットされた CriAuPlayer は，再生後なので PLAYEND である．また SoundPlayerState は「何も読んでいない状態」である．
2. CriAuPlayer が STOP か PLAYEND で，かつ SoundPlayerState が「何も読んでいない状態」ならば，“７”を再生し，SoundPlayerState を「“７”を読んでいる状態」にする
3. “７”の再生が終わると，CriAuPlayer は PLAYEND という状態になり，SoundPlayerState は「何も読んでいない状態」になる

一見，何も変わっていないようだが，例えば“７”の音が再生し終わったあとに別の音を再生したい，というような状況を考えると違いがよく分かる．SoundPlayerState を導入したことで，CriAuPlayer が PLAYEND になっても即座に同じ音がもう一度再生されるという状況を防止できる．これによって，同じ音を 2 回以上再生し，なおかつ意図しないのに何度も再生されてしまうという状況を改善できた．

- キーが過剰反応してしまうバグの修正

キーが押されているかどうかを判断するために，SDL を用いて以下のような関数を用いていた．

```
bool Util::isPressed(int id) {
    Uint8* keys = SDL_GetKeyState(NULL); // すべてのキーの状態を取得する
    return keys[id] == SDL_PRESSED; // 対象のキーが押されているかどうかを返す
}
```

しかし，この関数を使っているとキーを押しっぱなしにしたときに過剰反応してしまうというバグが見つかった．たとえば説明を読み上げているときに Enter を押すと説明を 1 つ飛ばすのだが，キーを押しっぱなしにすると Enter が何度も押されていると判断されて説明を全部飛ばしてしまうのである．押しっぱなしであると判断されない間隔がかなりシビアで，普通にキー操作しているだけでも押しっぱなしであると判断されて，ゲームの実行に支障をきたしていた．そのため，キーが押されているかを判断する関数に改良を加えて，次のような関数を作った．

```
bool Util::isPressedOnce(int id) {
    Uint8* keys = SDL_GetKeyState(NULL);
    if (keys[id] == SDL_PRESSED) {
        // キーが過剰に反応することを防ぐ
        if (!isKeyPressed) {
            isKeyPressed = true;
            return true;
        }
    } else {
        isKeyPressed = false;
    }
    return false;
}
```

まず，グローバル変数としてキーが押されたかどうかを判断する isKeyPressed という bool 型のフラグを用意して，初期値として false を与えておく．そして，キーが押された場合はこのフラグを true にする．フラグが false である最初の 1 回だけはキーが押されていると判断する．また，キーが離された時点で isKeyPressed は false に戻る．この関数を用いることで，キーの過剰反応を防ぐことができた．

4.1.5. ライブラリについて

今回使用したライブラリは，CriAudio，SDL(Simple Directmedia Layer)，Boost の 3 種類である．この節では，これらのライブラリについて解説する．

- CriAudio について

CriAudio は，サラウンド対応の音声を作成し，再生するためのミドルウェアである．C++から CriAudio のサウンドデータを再生するには, CriAudio ライブラリを使用する．

映像を用いないゲームにおいては，CriAudio は非常に利用価値が高く，有用なものであった．以下 CriAudio について解説する．

➢ CriAudio のワークフロー

wav 等のサウンドファイルを，CriAudioCRAFT を使用してキューシートバイナリファイルに変換し，キューシートバイナリファイルを，CriAudio ライブラリを使用して再生する（図 3.6 参照）．

➢ CriAudio のアーキテクチャ

CriAudio は，大きく分けて 2 つの部分から構成されている．一つは音をどのように鳴らすかを制御するボイスコントロール部，もう一つは鳴らしたい音を発生させるサウンドレンダラ部である．この二つはそれぞれ演奏者と楽器に例えることができる．

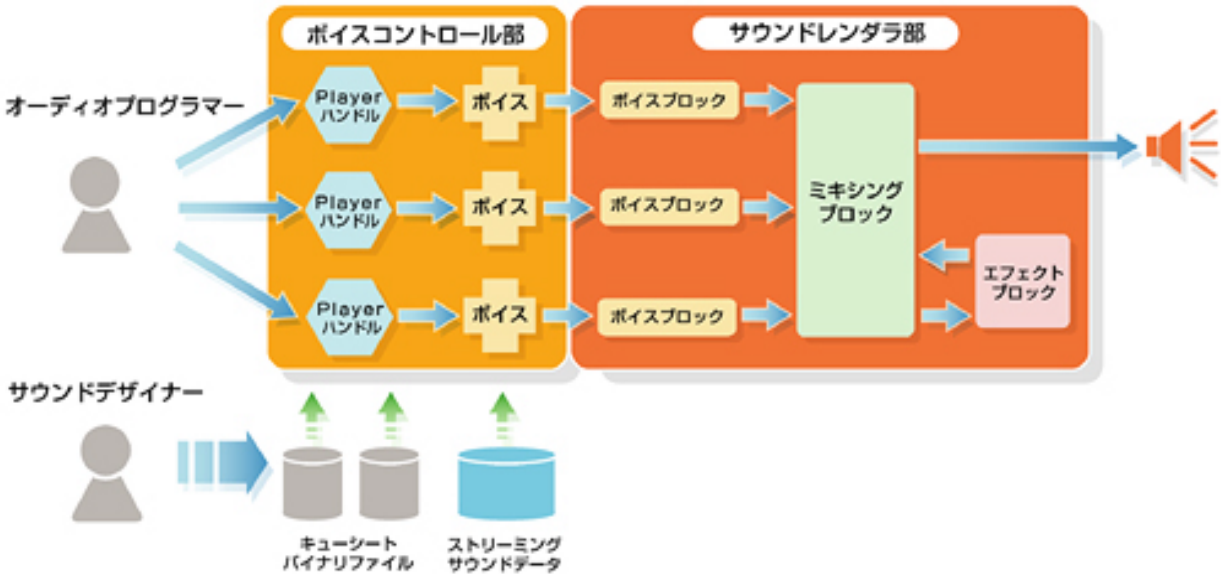


図 3-19：CriAudio のアーキテクチャ

➢ AISAC

AISAC（Advanced Interactive Sound and Active Controller）とは，回転数に応じたエンジン音や，試合の盛り上がりに呼応した競技場の歓声，発音源までの距離に応じて変化する足音や声など，ゲームの状況に応じて変化するインタラクティブなサウンドを設計するためのコンポーネントである．このコンポーネントを使うことで，虫の距離に応じて音量や定位を変化させる，音に奥行きを持たせるためにリバーブをかける等といった複雑な計算や処理を簡単に実現することができる．

➢ CriAudio を導入するメリット

CriAudio を導入するメリットは，大きく 3 つ挙げられる．

一点目は，音を作成する自由度が高いことである．定位やリバーブのコントロールはもちろん，ドップラー効果やオクルージョン（こもり音）等の音が自由に作成でき, 微調整も簡単に行える. また, 直接プログラムをいじらずに, CriAudioCRAFT 等のツールを使用することにより，GUI で簡単に音声ファイルを作成することができるので，プログラムに精通していない人が音ファイルの作成を担当することもできる．

二点目は，音の再生が容易に行えることである．作成したキューシートバイナリファイル上のサウンド名を指定するだけで再生が行える．そのため，非常に高度に作り込まれた音ファイルであっても，それを利用する側は何も意識せずにその音ファイルを利用することができる．虫捕りゲームを作成する際は，実際に音声ファイルを作る人とプログラムを作成する人を分けたが，特に不自由なく音声ファイルを扱うことができた．

三点目は，マルチコア CPU に最適化してくれることである．今回はマルチコア CPU を用いなかったが，マルチコア CPU を用いた場合，ゲームのメイン処理を行うコアとは別のコアで CriAudio の処理を行うことができるので, メイン処理に負担をかけずに音声の再生をすることができる．

➢ その他の機能

CriAudio には，ポリフォニック再生機能，ランダム再生機能，シーケンシャル再生機能，4ch サラウンドリバーブ機能，クロスフェーディングやボイスハイライトを実現するシネマティックサウンド機能，音声再生中のファイル読み込み機能といった様々な機能が搭載されている．今後，虫捕りゲームをバージョンアップさせる際に，より高度な音声処理を行う必要があるかもしれないが，そのような場合でも，CriAudio の各種機能を使用して様々な音声を作成することが可能である．

➢ 参考 URL

CriAudio に関する情報は，以下のサイトから取得することができる．

http://criware.jp/products/product_criaudio_j.htm

● SDL について

SDL(Simple Directmedia Layer)は，フリーなクロスプラットフォームの マルチメディア開発用 API である．SDL は，主にデバッグ用の画面を描画するために用いた．以下 SDL について解説する．

➢ 主な機能

SDL の主な機能は，グラフィックスの描画，キーボード入力，マウス入力，ユーザからの要求による終了といったイベント処理，オーディオの再生，CD-ROM オーディオの制御，タイマー等がある．今回使用したのは，主にグラフィックスを描画する機能である．

➢ 虫捕りゲームにおいて SDL を利用した箇所

虫捕りゲームでは，スクリーンの表示や，スクリーンの状態の更新，デバッグ用に文字を描画する必要のある箇所，キーイベントの部分で，SDL を使用した．

スクリーンのサイズ等についての情報やスクリーンの状態については，SDL_Surface 構造体が管理する．Game.cpp のコンストラクタで SDL_SetVideoMode 関数を呼び, スクリーンの幅, 高さ, ピクセル深度等を設定し，SDL_Surface を作成している．State クラスでは，ゲーム中のスクリーンの表示を更新するために，SDL_Flip 関数を呼び，また，ClearScreen 関数内で一度スクリーン全体を黒く塗りつぶして，画面を再描画している．

文字の描画については，TTF_RenderText_Blended 関数でフォントや色の情報を保持したテキストを作成し，SDL_BlitSurface 関数で画面上に文字列を描画している． SDL ライブラリを利用せずに文字列を描画する場合は，WindowsAPI を利用して複雑で手間のかかる処理を記述しなければならないが，SDL ライブラリ使用することで，非常に少ない手間で文字列の描画をすることができる．

キーイベントについては，当初 SDLKey 構造体を使ってキーイベントの処理を行っていたが，SDLKey 構造体では，上下左右の方向キーが使用できなかったため，この構造体は使わないことにした．最終的には，SDL_GetKeyState 関数でキーの状態を監視し，指定されたキーが押されていたら，イベントを発生させるようにしている．また，イベントが発生する度にイベントキューに追加されていくので，State クラスでは, 定期的に SDL_PollEvent 関数を呼び, キューからイベントを取り出し，処理するようにしている．

➢ CriAudio との競合

SDL を導入する際に，一つ問題が発生した．Uint32 と Sint32 という名前の構造体が，CriAudio と SDL の双方で定義されていたため，定義が重複してしまい，コンパイルエラーになってしまった．この問題を回避するために，当初は片方の定義をコメントアウトしていたが，両方とも定義されている内容は同様だったために，すでに定義済みの場合のみ定義を有効にするように，プリプロセッサを記述した．

具体的には，以下のような記述をした．

```
#if !defined(_TYPEDEF_Uint32)
#define _TYPEDEF_Uint32
typedef unsigned long Uint32;
#endif

#if !defined(_TYPEDEF_Sint32)
#define _TYPEDEF_Sint32
typedef signed long Sint32;
#endif
```

このように記述することで，まだ構造体が定義されていないときのみ，定義を行うようにした．

➢ 参考 URL

下記の URL は，SDL のオフィシャルサイトの URL である．SDL のオフィシャルサイトから，SDL のダウンロードと，SDL についての情報を得ることが可能である．

http://www.libsdl.org/

- Boost について

Boost は，国際規格で定められた C++標準ライブラリの他に，さらに有用で移植性のあるライブラリを提供することを目的に，標準化委員会のメンバーによって作成された．以下 Boost について解説する．

➢ 主な機能

Boost は非常に多機能であり，様々なことができるが，代表的な機能としては以下のようなものが挙げられる．

✧ 自動的にオブジェクトを delete してくれるスマートポインタ

- ✧ 文字列・数値間のキャストや正規表現
- ✧ イテレータやコンテナ等のデータ構造とアルゴリズム
- ✧ グラフ，有理数，乱数，GCD，LCM，四元数等の数学関係の処理
- ✧ 日時・時刻・時間・時間経過等の制御

これらの機能うち，虫捕りゲームでは，文字列・数値間のキャストの機能を主に使用した.

- ➢ 虫捕りゲームにおいて Boost を利用した箇所

虫捕りゲームでは，主に文字列を数値に変換する，または数値を文字列に変換する必要のある箇所で，Boost の lexical_cast を用いた．虫捕りゲームでは，デバッグ用に SDL を利用して虫の座標や経過した時間等を描画するようにしているが，描画をするデータは文字列型として扱わなくてはならないため，数値を描画したい場合でもそれを一度整数型に変換する必要があった. しかし, C++の通常のキャストでは, 数値から文字列型への変換はサポートされていないので，Boost の lexical_cast を用いる必要があった. Boost の lexical_cast を用いることで, 文字列を数値に変換する, または数値を文字列に変換することができる.

- ➢ 参考 URL

下記の URL は，Boost のオフィシャルサイトの URL である．Boost のオフィシャルサイトから，Boost のダウンロードと，Boost についての情報を得ることが可能である.

http://boost.org/

4.1.6. 最終的なクラス構成とソースコードの規模

- クラス構成

β版のクラス構成は，最終的に図 3-20 のようになった.

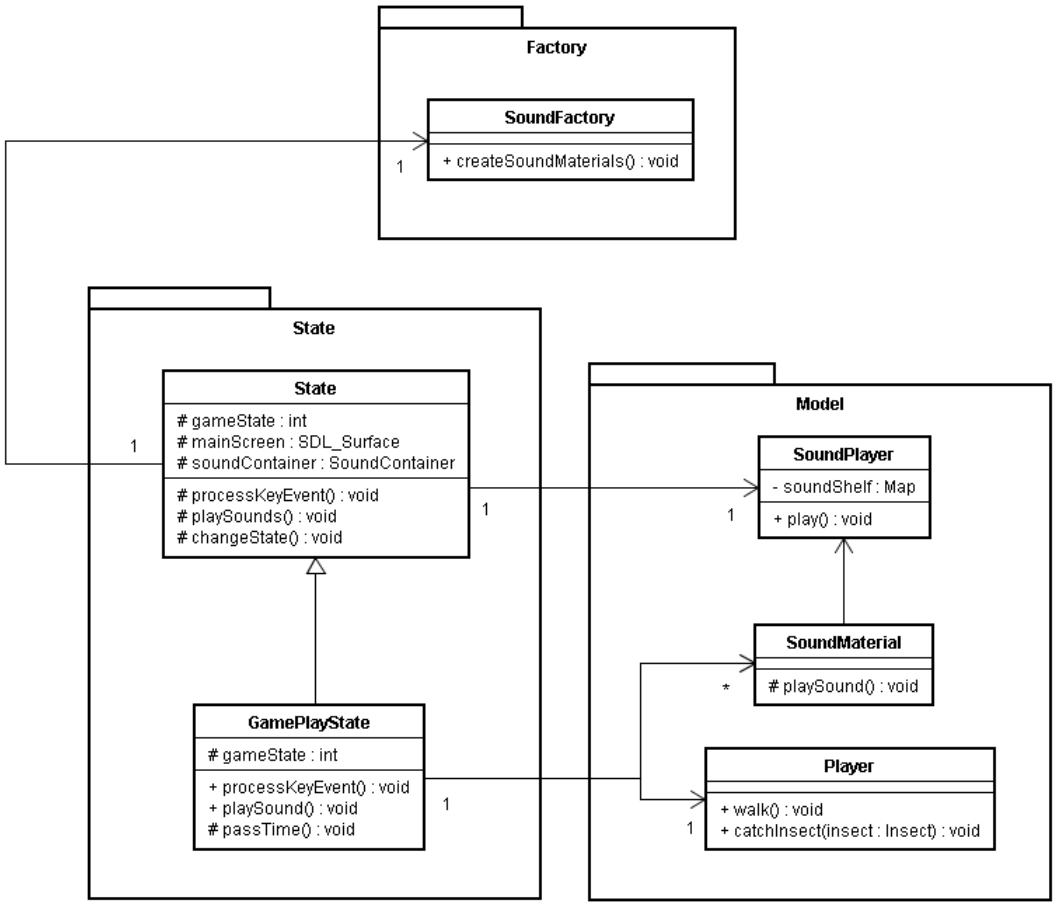


図 3-20：全体のクラス図

基本的な構造は, State クラスが SoundFactory クラスと SoundPlayer クラスを参照し, State クラスを継承した GamePlayState クラスが，SoundMaterial クラスと Player を参照する．SoundFactory クラス，State クラス，GamePlayState クラス，SoundMaterial クラスは，それぞれサブクラスを持つが，図 3-20 では省略している．

以下に，各パッケージのサブクラスを含めたクラス図を示す．

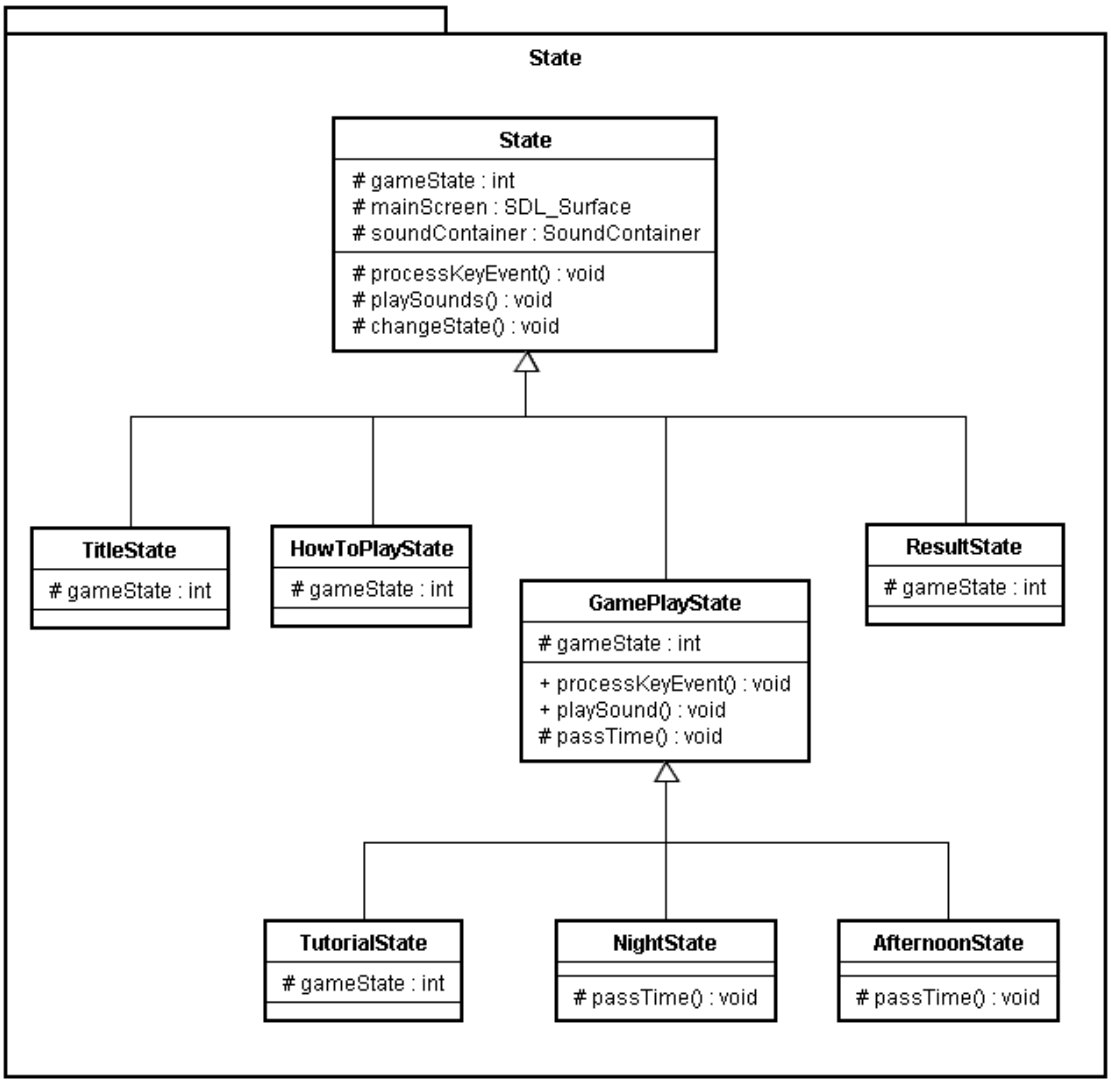


図 3-21：State パッケージ内のクラス図

図 3-21 は，State パッケージ内のクラス図である．各 State クラスは，ゲーム中のそれぞれの状態を管理する．各 State の遷移については，図 3-9 を参照していただきたい．

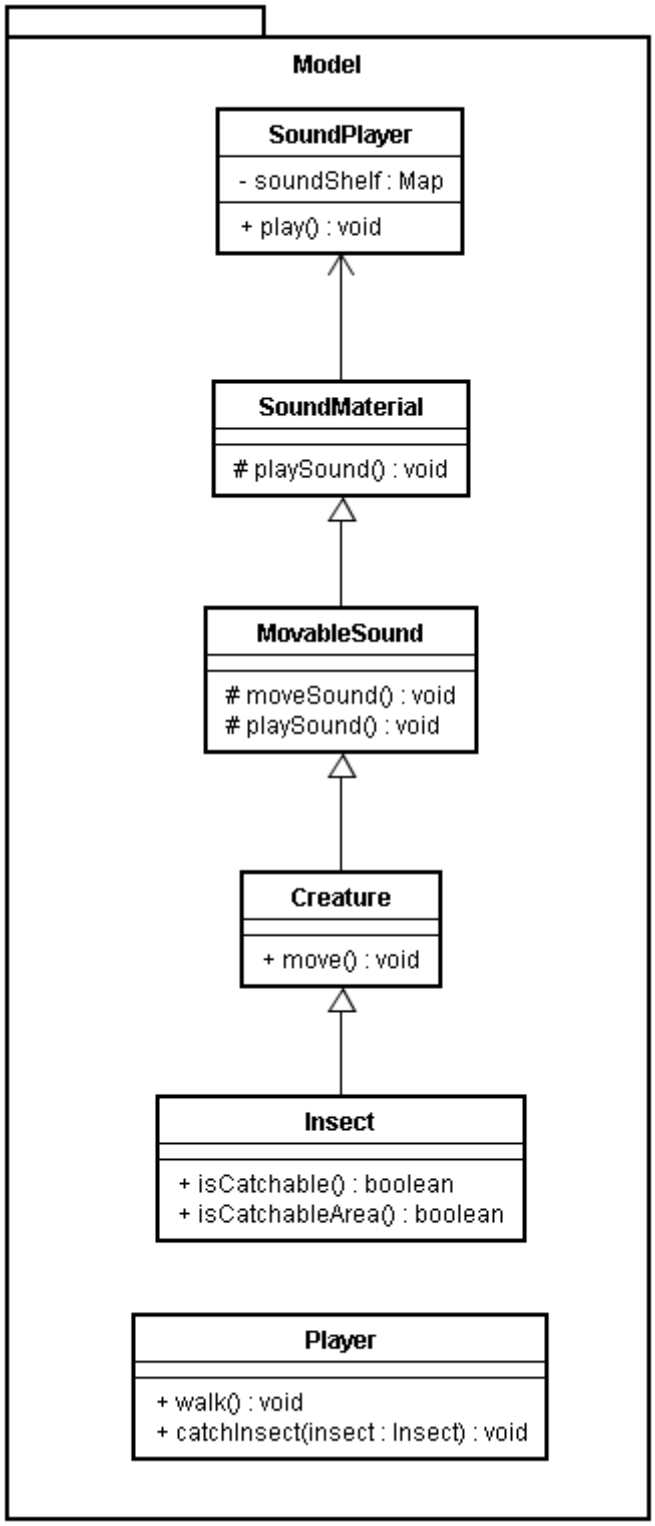


図 3-22：Model パッケージ内のクラス図

図 3-22 は，Model パッケージ内のクラス図である．SoundMaterial クラスが個々の音の

振る舞いを管理し，音を再生する・停止するといった処理は SoundPlayer クラスに委譲している．必要となる音の振る舞いによって，SoundMaterial クラスをベースに，継承して新たなクラスを作成している．

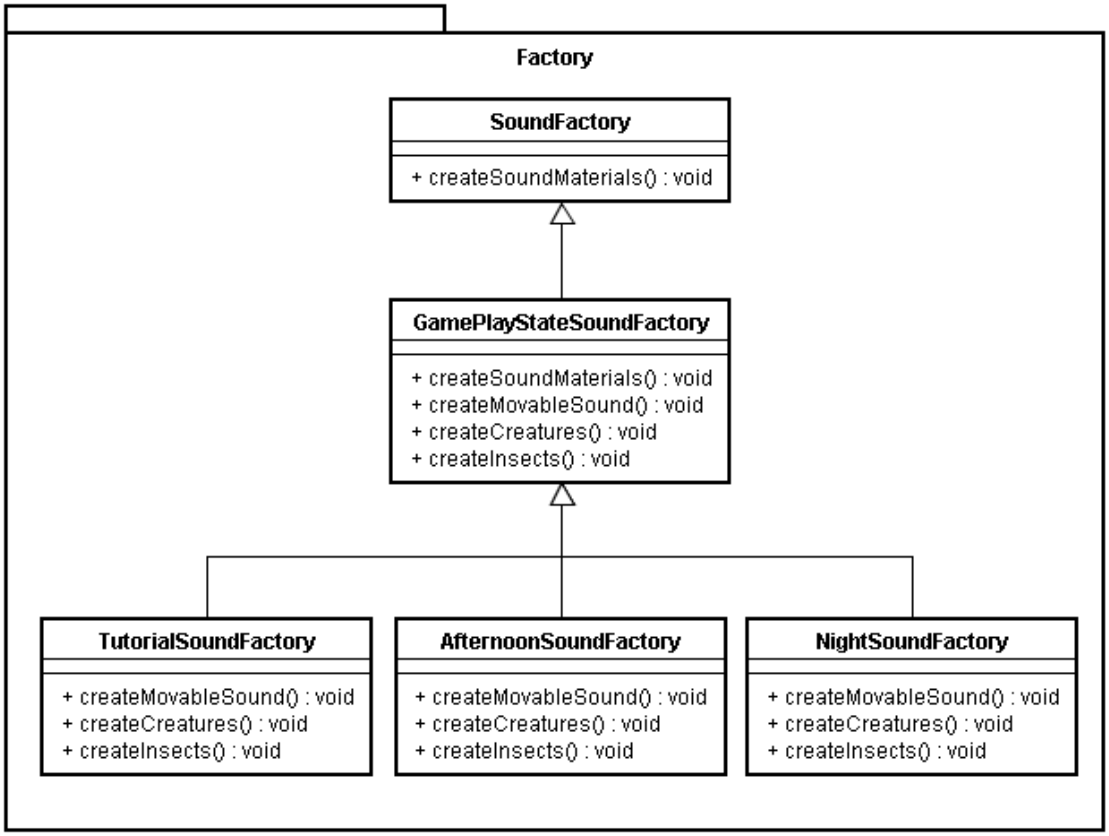


図 3-23：State パッケージ内のクラス図

図 3-23 は，Factory パッケージ内のクラス図である．State で使用する音の数が多く，生成が複雑になってしまうので，各 State クラスで必要な音の生成は，Factory クラスに委譲している．各 Factory クラスは，State パッケージのクラスごとに分割されている．TitleState クラス，HowToPlayState クラス，ResultState クラスは現在 Factory クラスを利用していないが，将来的に利用したくなったときのために，SoundFactory クラスを用意してある．なので，SoundFactory クラスを継承して各 State に必要な Factory クラスを作成することができる．

- ソースコードの規模

β版のソースコードの規模については，以下の通りである．

ファイル形式	行数	コメント	空行	Logic	ファイル名
cpp	75	20	9	46	State.cpp
cpp	26	18	3	5	TextCreator.cpp
cpp	77	16	10	51	TitleState.cpp
cpp	41	17	6	18	TutorialSoundFactory.cpp
cpp	199	38	32	129	TutorialState.cpp
cpp	205	56	28	121	Util.cpp
h	19	6	3	10	AfternoonSoundFactory.h
h	27	8	4	15	AfternoonState.h
h	32	6	9	17	Cage.h
h	37	11	6	20	Constants.h
h	37	8	7	22	Creature.h
h	31	6	6	19	Game.h
h	67	9	11	47	GamePlayState.h
h	27	7	4	16	GamePlayStateSoundFactory.h
h	22	6	3	13	GameStates.h
h	47	9	11	27	HowToPlayState.h
h	53	9	11	33	Insect.h
h	18	0	3	15	KeyFlag.h
h	33	0	7	26	KeyState.h
h	63	10	11	42	MovableSound.h
h	25	6	5	14	NightSoundFactory.h
h	27	8	4	15	NightState.h
h	64	4	14	46	OnMapObject.h
h	64	8	8	48	Player.h
h	63	11	14	38	ResultState.h
h	12	0	3	9	River.h

h	30	1	4	25	SDLCommonFunctions.h
h	36	0	7	29	Sound.h
h	38	5	8	25	SoundContainer.h
h	25	7	5	13	SoundFactory.h
h	39	6	6	27	SoundMaterial.h
h	44	5	7	32	SoundPlayer.h
h	52	6	7	39	State.h
h	62	5	8	49	StateMessages.h
h	24	10	4	10	TextCreator.h
h	26	7	5	14	TitleState.h
h	13	0	3	10	TutorialSoundFactory.h
h	40	7	7	26	TutorialState.h
h	38	6	9	23	Util.h
cpp	43	9	6	28	AfternoonSoundFactory.cpp
cpp	113	31	16	66	AfternoonState.cpp
cpp	45	17	6	22	Cage.cpp
cpp	63	20	7	36	Creature.cpp
cpp	54	10	9	35	Game.cpp
cpp	427	86	50	291	GamePlayState.cpp
cpp	53	19	7	27	GamePlayStateSoundFactory.cpp
cpp	194	34	26	134	HowToPlayState.cpp
cpp	1	0	1		InputDevice.cpp
cpp	138	30	16	92	Insect.cpp
cpp	44	15	4	25	KeyFlag.cpp
cpp	57	21	9	27	KeyState.cpp
cpp	34	5	6	23	Main.cpp
cpp	200	48	35	117	MovableSound.cpp
cpp	52	9	8	35	NightSoundFactory.cpp
cpp	150	33	18	99	NightState.cpp
cpp	85	24	13	48	OnMapObject.cpp

cpp	253	54	37	162	Player.cpp
cpp	318	70	36	212	ResultState.cpp
cpp	31	16	4	11	River.cpp
cpp	26	8	4	14	Sound.cpp
cpp	93	24	15	54	SoundContainer.cpp
cpp	13	6	2	5	SoundFactory.cpp
cpp	86	23	12	51	SoundMaterial.cpp
cpp	163	25	23	115	SoundPlayer.cpp

ファイル形式	行数	コメント	空行	Logic	ファイル数
cpp	3,359	802	458	2,099	31
h	1,235	197	224	814	33
合計	4,594	999	682	2,913	64

全体の規模については，プロジェクトメンバー各々が C++で開発したα版と比較して，3 倍以上大きくなっている．

また，β版の開発では設計書を作成しない代わりに，必要事項はソースコードにコメントとして記述し，ソースコードの可読性を高めるよう努めた．そのため，コメントの行数に関しては，C++で開発したα版と比較して約 5 倍多くなっている．また，ソースコード全体に占めるコメントの割合は，C++で開発したα版が約 15%だったのに対し，β版では約 22%に増加している．

4.2. β版評価

虫捕りゲームのβ版を，大岩研究室の方々，横浜市立盲学校の方々，株式会社ユードーの方々にプレイしていただき，アンケートに答えていただいた．アンケートの質問項目と選択肢は以下の通りである．

- 質問 1.「映像を用いないゲーム」をプレイした感想は？
  - ① 非常に面白い
  - ② 面白い
  - ③ 普通
  - ④ つまらない
  - ⑤ 非常につまらない　の 5 段階

- 質問 2.「映像を用いないゲーム」をまたプレイしたいと思うか？
  - ① 是非プレイしたい
  - ② 機会があればプレイしたい
  - ③ どちらでもよい
  - ④ あまりプレイしたくない
  - ⑤ 二度とプレイしたくない　の 5 段階

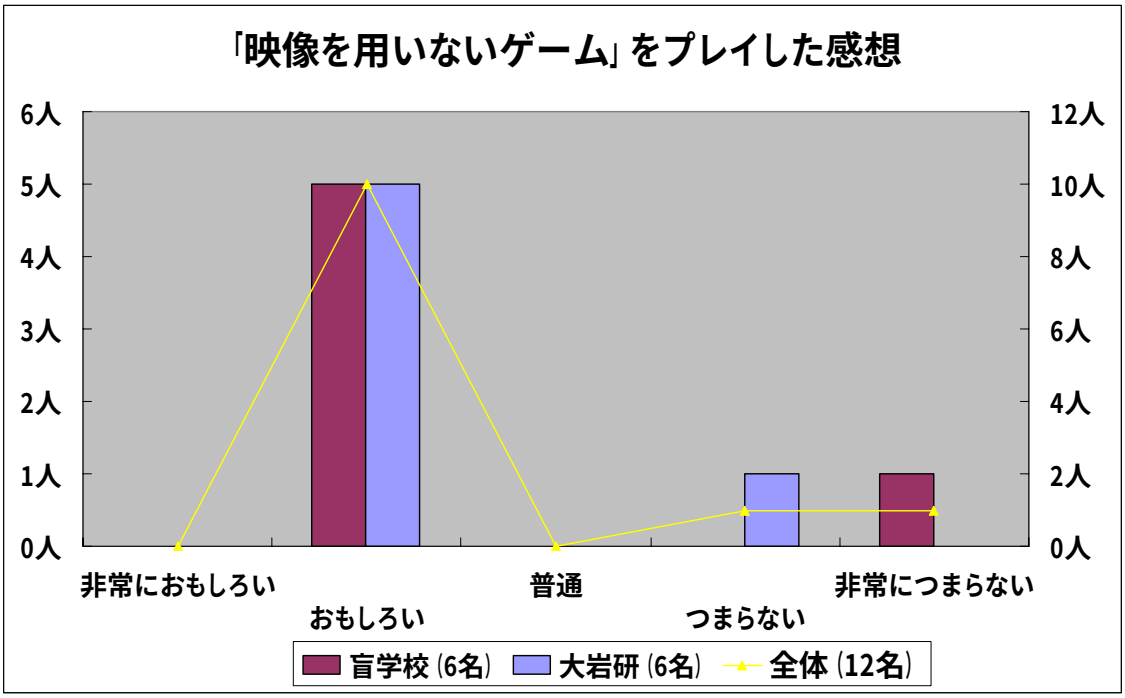
- 質問 3.「映像を用いないゲーム」をプレイして面白かった点は？（複数回答可）
  - ① 映像を用いないという新しいゲームを体験できる
  - ② 音に臨場感がある
  - ③ 捕った虫の数やスコアを競うことが出来る
  - ④ ゲームの難易度・バランスがちょうど良い
  - ⑤ その他　の 5 項目

- 質問 4.「映像を用いないゲーム」について改善・追加した方がよいと感じた点は？（複数回答可）
  - ① 虫や自然の音の聞こえ方
  - ② 音声による説明
  - ③ ゲームの操作性
  - ④ ゲームの難易度・バランス
  - ⑤ 虫捕り以外の要素を追加した方がよい
  - ⑥ その他　の 6 項目

以上の 4 項目と，自由記述欄である．

今期の「さうんど おんりぃ 2」プロジェクトでは，ゲームの開発が成功したか失敗したかを判断するための指標として，「ゲームを遊んでみて，面白い・もう一度プレイしてみたいと思う人の割合が，前期の『さうんど おんりぃ』プロジェクトよりも大きいかどうか」という判断基準を設けた．具体的には，前期の「さうんど おんりぃ」プロジェクトでは，面白いと思う人が 5 割，もう一度やりたいと思う人が 7 割という結果だったので，今期の「さうんど おんりぃ 2」プロジェクトでは，面白いと思う人が 7 割以上，もう一度やりたいと思う人が 8 割以上いるという数値を目標として設けた．以下，アンケートの結果を示す（添付資料：アンケート結果参照）．

表 3-3：「映像を用いないゲーム」をプレイした感想



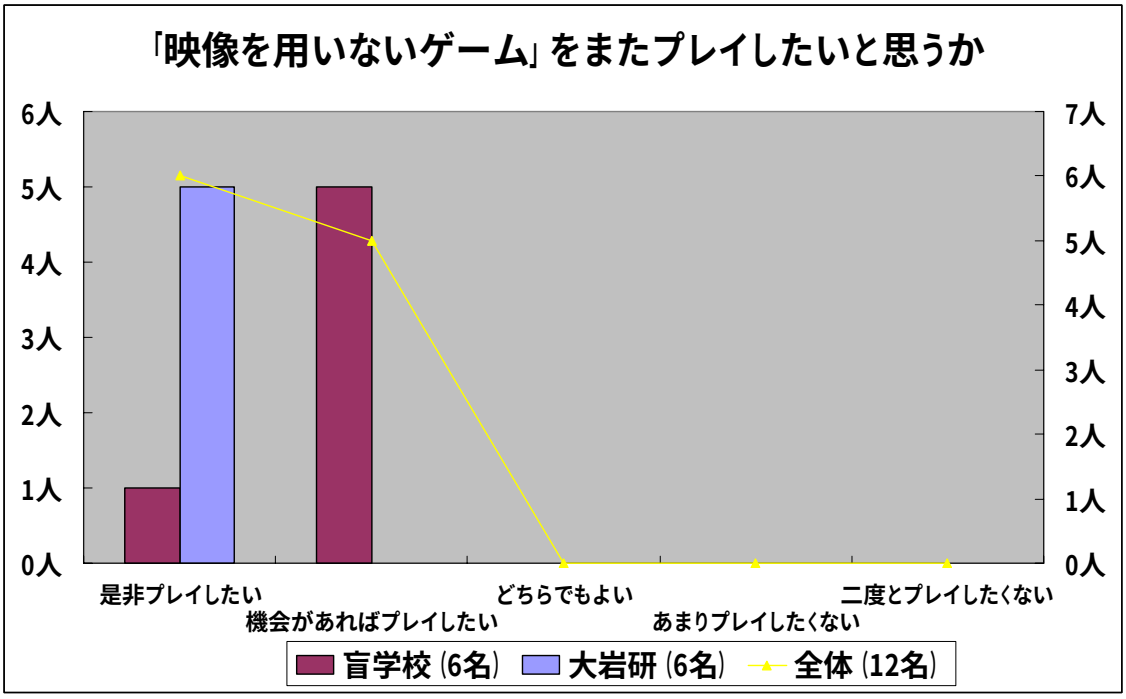
「映像を用いないゲーム」をプレイしてみて，面白いと感じたという回答を行ったのが横浜市立盲学校の生徒・大岩研究室関係者ともに 6 人中 5 人で，その割合は 8 割 3 分であった．

面白いと感じた人の割合は多かったが，非常に面白いという評価をした人はいなかった．これは，ゲームとしては未完成で十分に遊べるレベルではないが，今後開発が進んでゲームがより進化すれば，十分に遊べるものができそうであるという印象を受けた方が多かったということであると考えている．実際に，改良を加えれば非常に面白いものになりそうであるという意見は大岩研究室・盲学校の両方から多数頂いた．今後もより面白く遊んでもらえるゲームになるように改良を加えていきたい．

つまらない，非常につまらないと感じた人は，難易度が高くてなかなか虫が捕まらない

ことや，操作性の悪さにストレスを感じたようである．難易度が高い・操作性がよくないという指摘は盲学校・大岩研に関わらず多く寄せられたので，この点に関しては特に改善を要するところである．

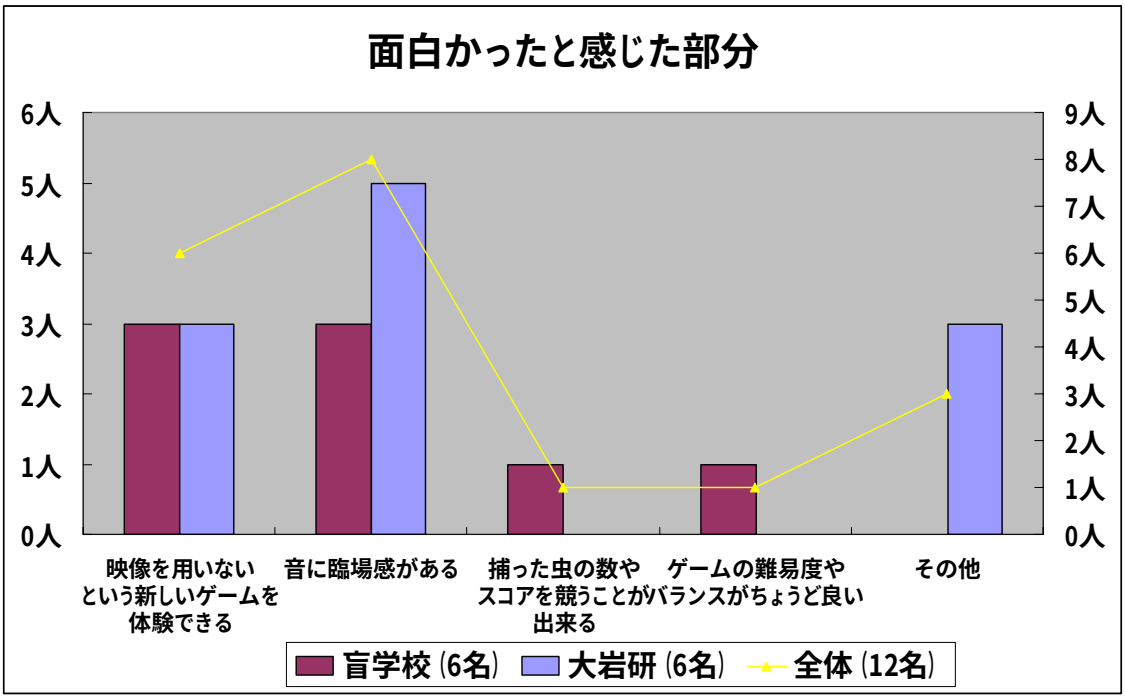
表 3-4：「映像を用いないゲーム」をまたプレイしたいと思うか



「映像を用いないゲーム」をプレイしてみて，是非またプレイしたい・機会があればプレイしたいと感じた人は，横浜市立盲学校の生徒・大岩研究室の関係者ともに 6 人中 6 人で，10 割であった．

全ての人が「是非プレイしたい」，もしくは「機会があればプレイしたい」という項目にチェックを入れた（どの項目にも記入していない人が一名いた．記入し忘れたものと思われる）．この結果から，今回作成した「映像を用いないゲーム」は，一度遊んだだけで飽きてしまうようなものではなく，何度も遊びたいと思えるようなゲームであるということができる．今後もより長く遊べるように改良を加え，繰り返し楽しんでもらえるようなゲームにしていきたい．

表 3-5：面白かったと感じた部分

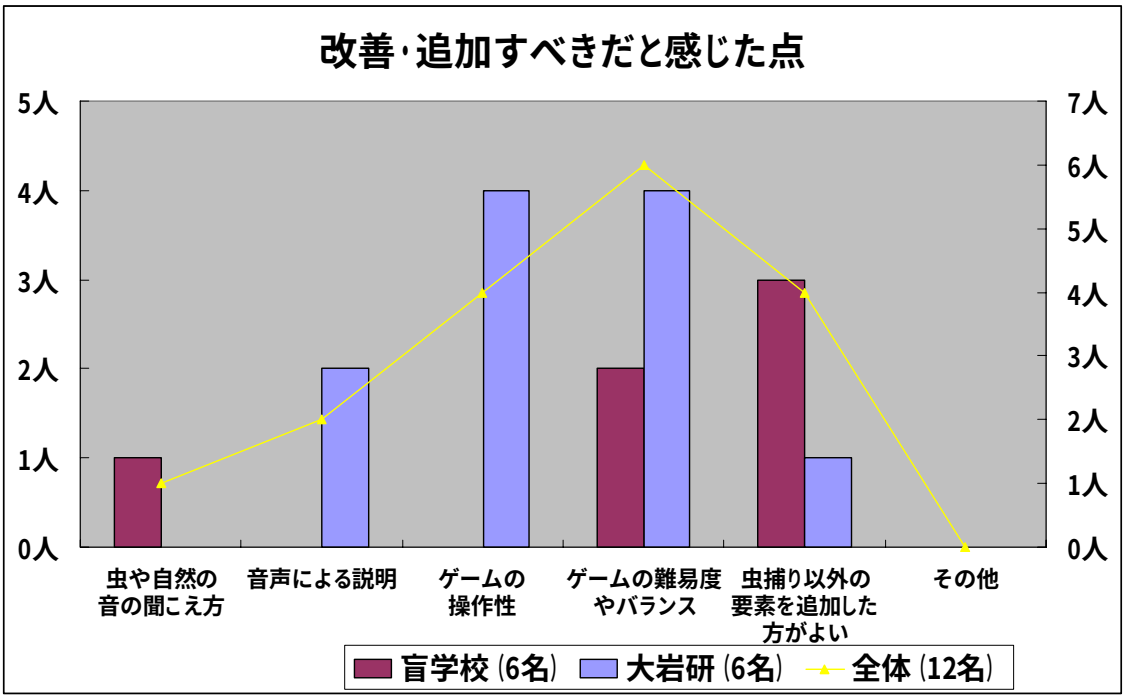


「映像を用いないゲーム」をプレイしてみて，面白いと感じた点についてであるが，「映像を用いないという新しいゲームを体験できる」点が面白いと感じた人は，横浜市立盲学校の生徒・大岩研究室関係者ともに 3 人であった．「音に臨場感がある」点が面白いと感じた人は，横浜市立盲学校の生徒が 3 人，大岩研究室関係者が 5 人であった．「捕った虫の数やスコアを競うことができる」点が面白いと感じた人は，横浜市立盲学校の生徒が 1 人であった．「ゲームの難易度・バランスがちょうど良い」点が面白いと感じた人は，横浜市立盲学校の生徒が 1 人であった．「その他」の点が面白いと感じた人は，大岩研究室関係者が 3 人であった．

「音に臨場感がある」と感じた人は，横浜市立盲学校の生徒の方々よりも，大岩研究室の方々の方が多かった．横浜市立盲学校の方々は，日頃から聴覚を研ぎ澄ませて生活しているため，音に関する感覚が鋭く，少しでも不自然な点があるととても気になってしまうため，大岩研究室の方々よりも臨場感を感じられなかったのではないかと考えられる．

「捕った虫の数やスコアを競うことができる」と，「ゲームの難易度・バランスがちょうど良い」にチェックを入れた方は，それぞれ一人しかいなかった．そのため，これらの項目に関しては，より改善していく必要があると感じている．

表 3-6：改善・追加すべきだと感じた部分



「映像を用いないゲーム」をプレイしてみて，改善・追加すべきだと感じた部分についてであるが，「虫や自然の音の聞こえ方」を改善すべきだと感じた人が，横浜市立盲学校の生徒が 1 人であった．「音声による説明」を改善・追加すべきだと感じた人は，大岩研究室関係者が 2 人であった．「ゲームの操作性」を改善した方がよいと感じた人は，大岩研究室関係者が 4 人であった．「ゲームの難易度・バランス」を改善すべきだと感じた人は，横浜市立盲学校の生徒が 2 人，大岩研究室関係者が 4 人であった．「虫捕り以外の要素を追加した方がよい」と感じた人は，横浜市立盲学校の生徒が 3 人，大岩研究室関係者が 1 人であった．

「音声による説明」や，「ゲームの操作性」を改善・追加すべきであるという項目にチェックしたのは，大岩研究室関係者が多かった．大岩研究室では，使いやすいソフトウェアを開発することを心がけている方が多いので，ソフトウェアの使い方の説明や，ゲームの操作性に関しては，シビアな視点を持っている方が多い．そのため，これらの項目について不満を持った方が多かったのではないかと考えられる．しかし，シビアな目を持つ方々にも満足していただける品質のゲームを作成するために，今後さらなる改良を加えていきたいと考えている．

「ゲームの難易度・バランス」については，全体的に不満を持った方が多かった．「難しすぎる，どこまで虫に近づいたら捕まえられるのかよくわからない」といった意見が多数寄せられたので，「ゲームの難易度・バランス」については特に改善を要する箇所であると思われる．具体的な改善案として，誰でも簡単に虫を捕まえられる初心者モードを追加し，

慣れた人はより難しいモードを選択できるようにするということが上げられる．

アンケートの結果，面白いと思う人が 8 割 3 分，もう一度やりたいと思う人は 10 割であった．そのため，面白いと思う人が 7 割以上，もう一度やりたいと思う人が 8 割以上いることという当初の目標は完全に達成された．

4.2.1. 横浜市立盲学校の生徒の方々からの意見・指摘

横浜市立盲学校の生徒の方々からいただいた主な意見・指摘は，以下のようなものである．それぞれの意見・指摘について，横浜市立盲学校図書館司書教諭の松田基章様から補足意見をいただいた．

- 虫以外の音が目立ちすぎていて，虫の鳴き声がよくわからなくなってしまう

この指摘について，松田教諭から，「視覚にハンディキャップを有する方々は，視覚から情報を得ることができない代わりに，聴覚に頼って情報を得ていることが多い．そのため，聴覚が一般の人に比べて発達している人が多く，少しでも音に違和感があったり，煩雑だったりすると，非常に不快である．そのため，ゲーム中の音のバランスの悪さが気になってしまったのではないか．」という意見をいただいた．この点については，自分たちがプレイしたときも，注意深く聞いてみると，音によって音量の大きさにバラつきがあったり，音に不自然さがあったりすることがわかる．何気なくプレイしていると中々気付きづらい点であるが，このゲームの善し悪しを決める非常に大きな要素なので，少しでも不自然に感じる点は改良していく必要がある．

- ゲームの難易度が高すぎる

この指摘について，松田教諭から，「普段あまりゲームに触れる機会のない生徒も居るので，初めは非常に簡単なところから始めて，慣れてきたら徐々に難易度を上げていけるようにした方が楽しめるのではないか．」という意見をいただいた．この点に関しては，視覚にハンディキャップを持つ方に限らず，多くの方から同じような指摘をいただいているため，大いに改善を要する点である．このゲームでは，簡単な練習ができるようにはなっているが，ゲーム自体の難易度の設定等ができず，慣れないと中々虫が捕まえられず，ストレスやフラストレーションがたまってしまう．そのため，最初は虫が逃げなかったり，移動範囲が狭かったり，虫を捕るタイミングをアドバイスしてくれる等の簡単なモードを作成した方がよかった．

- 弱視の方が対象であっても，映像があった方がよい

この指摘について，松田教諭から，「横浜市立盲学校の中では，全盲の方の割合は低く，

弱視の方の割合の方が高い．そのため，弱視の方にとっては，ゲームに映像が付いていた方が面白いと感じるのだろう．また，視覚にハンディキャップを有する方々は，視覚にハンディキャップを有する方のみを対象に作られたゲーム，あるいは健常者が遊ぶ気にならないようなゲームには，魅力を感じない傾向がある．多くの人にとって魅力的なものを，視覚にハンディキャップを持つ方々も好む傾向がある.」という意見をいただいた．本プロジェクトのコンセプトとして,「音だけを使って健常者も視覚障害者も同じように楽しめるゲーム」を目指して開発していたのだが，映像がないとどうしても印象が地味になってしまい，魅力を感じにくくなってしまう面がある．この点に関しては,「ゲームには音だけしか使用しない」という方針を変更して多少の映像を使用するか，あるいは音だけを使ってより魅力的なゲームになるように改良していく必要がある．

4.2.2. 大岩研関係者の方々からの意見・指摘

大岩研関係者の方々からいただいた主な意見・指摘は，以下のようなものである．

- 距離感がわかりづらく，自分がどこにいるのかよくわからない

この点に関しては，移動をしているうちに自分がどちらの方角を向いているのか，マップ上のどの辺にいるのかがわからなくなってしまい，迷ってしまうことが多いというものである．現状のゲームでは，プレーヤーの方向や位置を知らせる仕組みがないため，混乱を招いてしまっているのだと考えられる．そのため，一定の方向から目印となるような音を鳴らしたり，虫に対しても捕まえられる範囲にいたら知らせてくれたりするような仕組みを用意して改善したいと考えている

- ゲームの難易度が高すぎる

この点に関しては，横浜市立盲学校の生徒の方々からいただいた指摘と共通である．現状のゲームは，誰がやっても難しいと感じてしまうようなので，改善したい．

- 移動できる範囲が狭い

この点に関しては，行き当たりばったりで移動しているとすぐに行き止まりになってしまい，戻らなくてはならないので，鬱陶しいと感じる方が多かったようである．この点に関しては，Config.txt でマップの広さを指定できるようになっているので，Config.txt の値をユーザが書き換えることでマップの広さを変更することができる．大岩研究室の関係者の方々にプレイしていただいた際には，その点をアナウンスし忘れてしまったので，アナウンスすべきであったと反省している．

- もっと虫のバリエーションが欲しい

この点に関しては，まだ捕まえられる虫が少ないので，捕まえられる虫の種類を増やしたいと思う．また，マップ上のエリアによって生息している虫の種類が異なっている等の工夫があるとより面白くなるのではないかという意見もいただいたので，参考にさせていただきたい．

- 虫を捕まえたことに対する感動があまり感じられない

この点に関しては，虫を捕っても「○○を捕まえた」という声だけで，もっと派手な演出が欲しいというものである．例えば，虫を捕まえたら，捕まえた虫に関する豆知識等を教えてくれたり，捕まえた虫の数によってスコアだけでなく，ランクも付けて，より高いランクを目指せるようにしたりすると，虫を捕まえるモチベーションが上がるのではないかという意見をいただいた．

4.2.3. クライアントからの意見・指摘

本プロジェクトのクライアントである株式会社ユードーの南雲玲生代表取締役から頂いた主な意見・指摘は，以下の通りである．

- 今回のテーマであった，演出や企画の質の向上

この点について，南雲代表取締役から，「具体的に，屋久島の音や，虫の効果音などの使用により，空間の演出がうまくできています．FPS の設計も理解できます．これをベースに，都市などのバージョンができますね．音が変わるだけでも，ユーザへ想像性や，美しさ，心地よさを提供できます．」というコメントをいただいた．空間の演出についてはうまくいっているという評価をいただけたので，ゲームの幅を広げていくために，今後はより様々な場面を再現することを考慮していきたい．

- ゲームバランスの調整が必要

この点について，南雲代表取締役から，「しばらく慣れるまでに，物標とプレーヤーの距離感がつかめません．どの音量レベルだと，目の前にいるという感覚が難しいのです．感覚をユーザへ与えるには，ゲームの遊び方，や，ゲーム開始時に，目の前に人を登場させ，この人が，”ゲーム開始”というアナウンスや，プレーヤーの属性をアナウンスします．その人との音量の距離感をデフォルト値にして，ゲームプレイヤーは，進められるのでしょう．」というコメントをいただいた．南雲代表取締役も，ゲームのバランスや操作性に問題を感じたようなので，この点については最優先で改善を要することを再認識させられた．また，「目の前に人を登場させてアナウンスさせることで，ゲームをプレイする人に距離感を教える」というアイデアを提案していただいたので，ゲーム改善の際には是非参考にさせて

いただきたい.

- サウンドの制作環境の向上

この点について，南雲代表取締役から，「基本的に，音の所在は，音のアタック（立ち上がりが早い）が強い，音圧が高いとわかり易いはずです．一部，存在が弱い音の虫がいた用に思います．あと，サウンドの編集ですね．品質の高い機器で，編集を行うと，圧倒的にクオリティが変わります.」というコメントをいただいた．β版のゲームで使用している音は，フリーで利用できるツールを使用して編集したものである．そのため，高度な編集ができず，音のクオリティは決して高いものではなかった．南雲代表取締役からは，株式会社ユードーの機材をお貸ししてくださると申し出ていただけたので，南雲代表取締役と相談しながら，よりクオリティの高い音を作成していきたい.

- ゲームイメージ（ブランド）の確立

この点について，南雲代表取締役から，「ゲームプランニングになりますが，ユーザに，想像力を提供できるゲーム名や，ストーリーを提供してもよいのでは．森ではなく，せっかく屋久島の音ですので，屋久島の冒険とか．インパクトが与えられますね.」というコメントをいただいた．β版のゲームでは，名もない森の中で漠然と虫を捕まえるだけなので，もっと具体的な場所で，なおかつ多くの人が魅力を感じるような場所を音で再現できるとよりおもしろくなりそうである．このようにゲームのイメージを鮮明にすることは重要なポイントであると思うので，今後考えていきたい.

4.2.4. 「虫捕りゲーム」から得られたもの

- ゲーム開発に向いているプロセス

今回の「さうんど おんりぃ２」の開発を行うにあたり，ゲーム開発特有のプロセスを用いた．企画書・仕様書は面白いと思う要素が思い浮かべば，どんどん改変していった．また，設計書は最低限手書きのメモ程度に残し，実装の時間をできるだけ多く捕ることでゲームを開発しながらその面白さを追及した.

前回のプロジェクトでは，PMBOK に従って要件定義，設計，実装，評価というフェーズを通じて，ソフトウェア開発手法を用いて「映像のないゲーム」を開発した．その結果，プロジェクトとしては期間内にスコープで定義したものが出来上がり成功を収めたが，ゲームとしてはそれほど面白いものが出来上がらなかった.

今回，ゲーム特有の開発手法を実践したことが，そこにはソフトウェア開発にはない様々な発見があった．詳細については個人レポートを参照して頂きたいが，ソフトウェア開発手法のスケジュール管理やタスク管理方法と，ゲーム開発手法の設計書に振り回されない実装方法をうまく組み合わせることで，ゲーム開発により適した新しいプロセスを組み合

わせて主観的にも客観的にも面白いと評価することが出来るゲームを開発できたことが大きな成果である．

- 「映像のないゲーム」のアイデアを実現するためのツール・環境の開発

虫捕りゲームは音を変えるだけで，比較的簡単に色々なゲームを開発することができる．たとえば，海中の音，砂漠の音などを使えば，様々な自然環境で魚や虫を捕まえるというゲームに出来る．また，発想を変えて，森の音を商店街の雑踏の音に変え，動物や虫の声を店先で商品を売る店員の声にすれば，店を探して商店街を歩き回るといったゲームも作ることができる．

つまり，虫捕りゲームの存在によって「映像のないゲーム」に様々な可能性を見出すことができるのである．虫捕りゲームはゲームとしてはまだまだ改良の余地が多く残されているが，「映像のないゲーム」の基礎が出来たことで，今回他にも挙がった様々な企画をゲームとして実現する目処が立った．色々な人が思い浮かんだ「映像のないゲーム」のアイデアを実現させることができるツールや環境の開発ができたことが，「虫捕りゲーム」開発の大きな成果である．

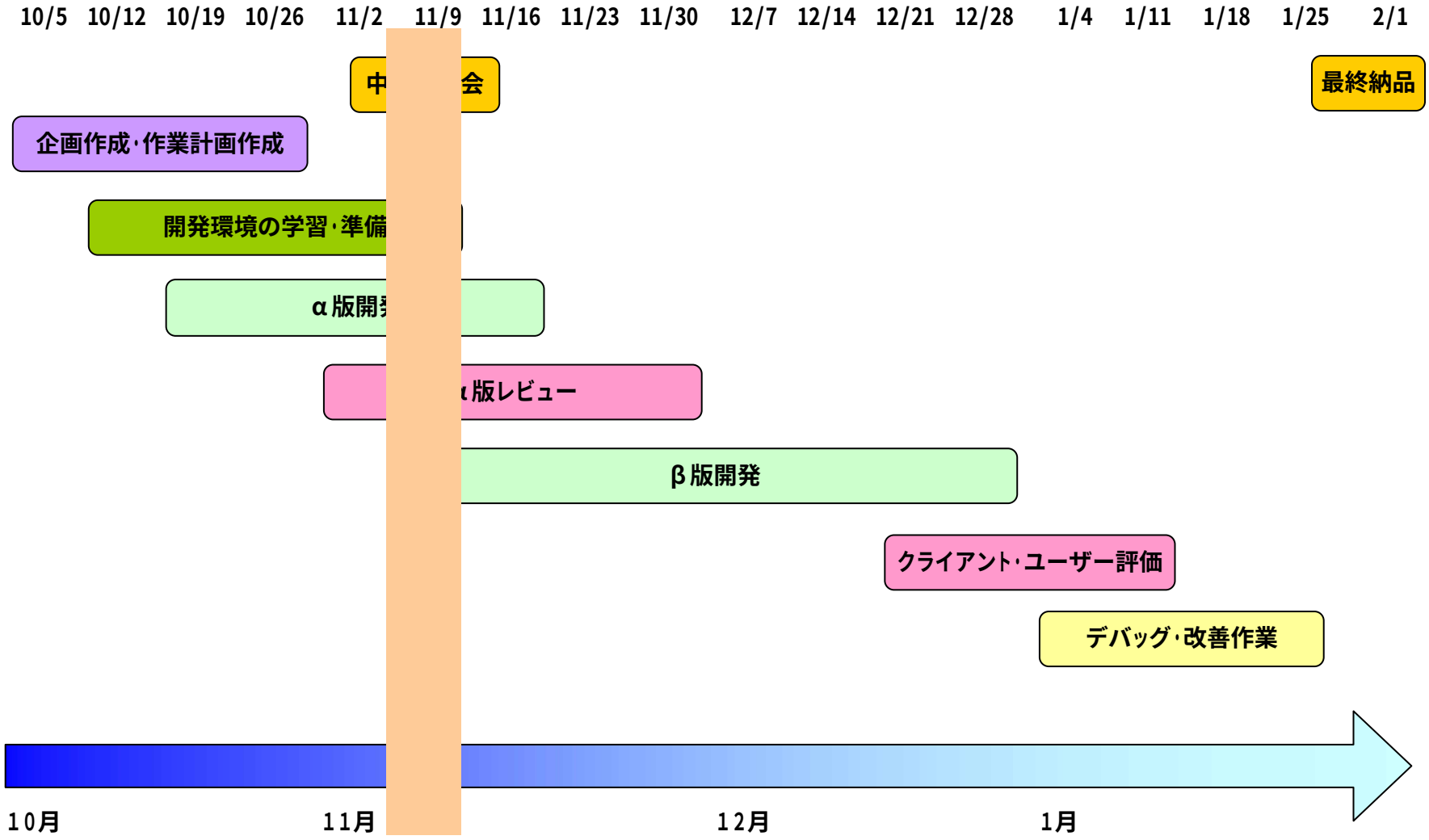
第5章. 添付資料

- 全体スケジュール
- 見積もりと実績
- ミーティングログ
- 進捗報告会資料
- 週報
- ソースコード（音記憶ゲーム：Java）
- ソースコード（聖徳太子ゲーム：Java）
- ソースコード（さうんどシュート：C++）
- ソースコード（聖徳太子ゲーム改：C++）
- ソースコード（ForestWalking：C++）
- アンケート結果
- プロジェクト宣伝用資料

2007/01/31
さうんど おんりぃ
最終報告書

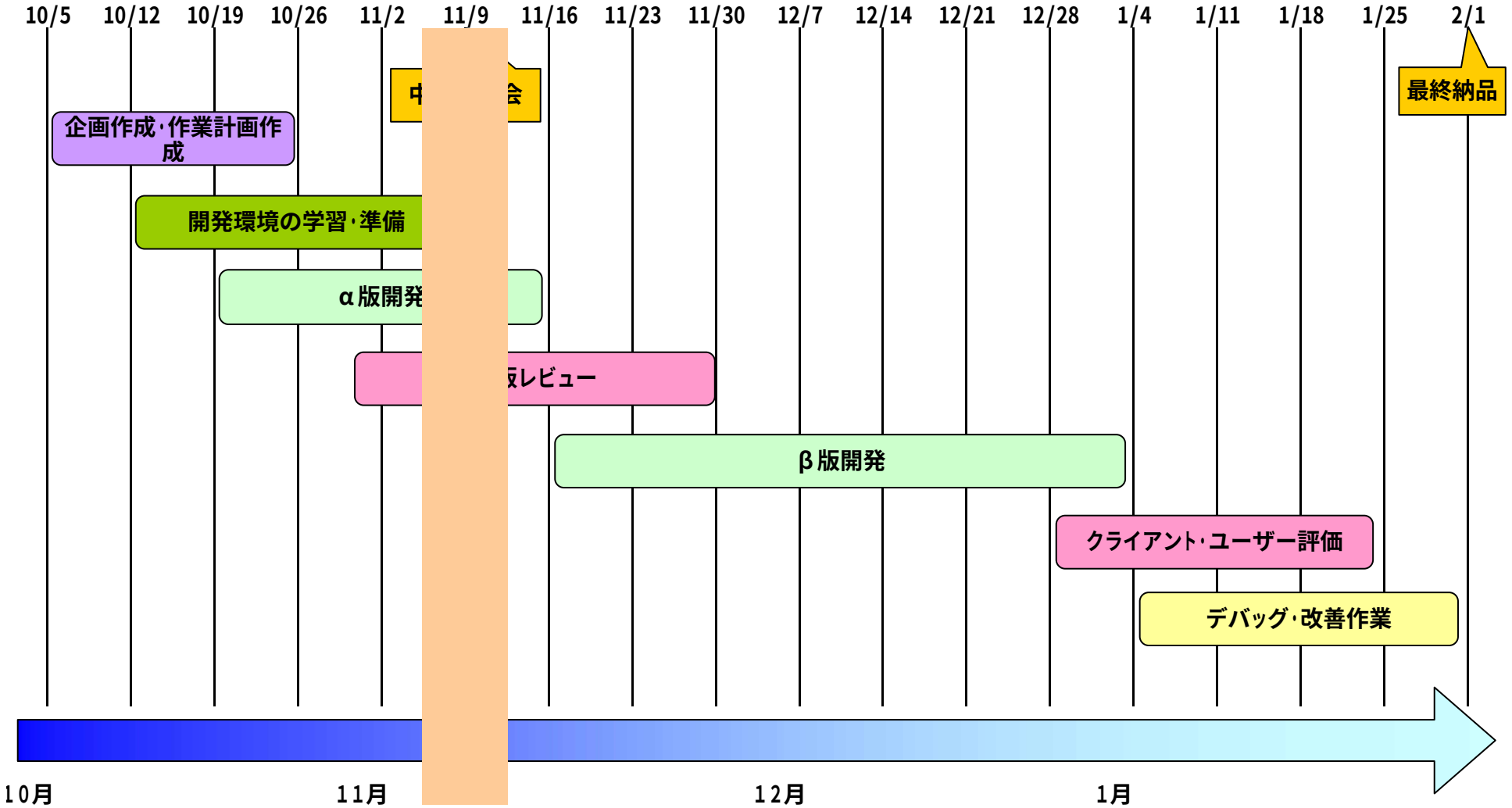
全体スケジュール

さうんどおんりぃ2　WBS



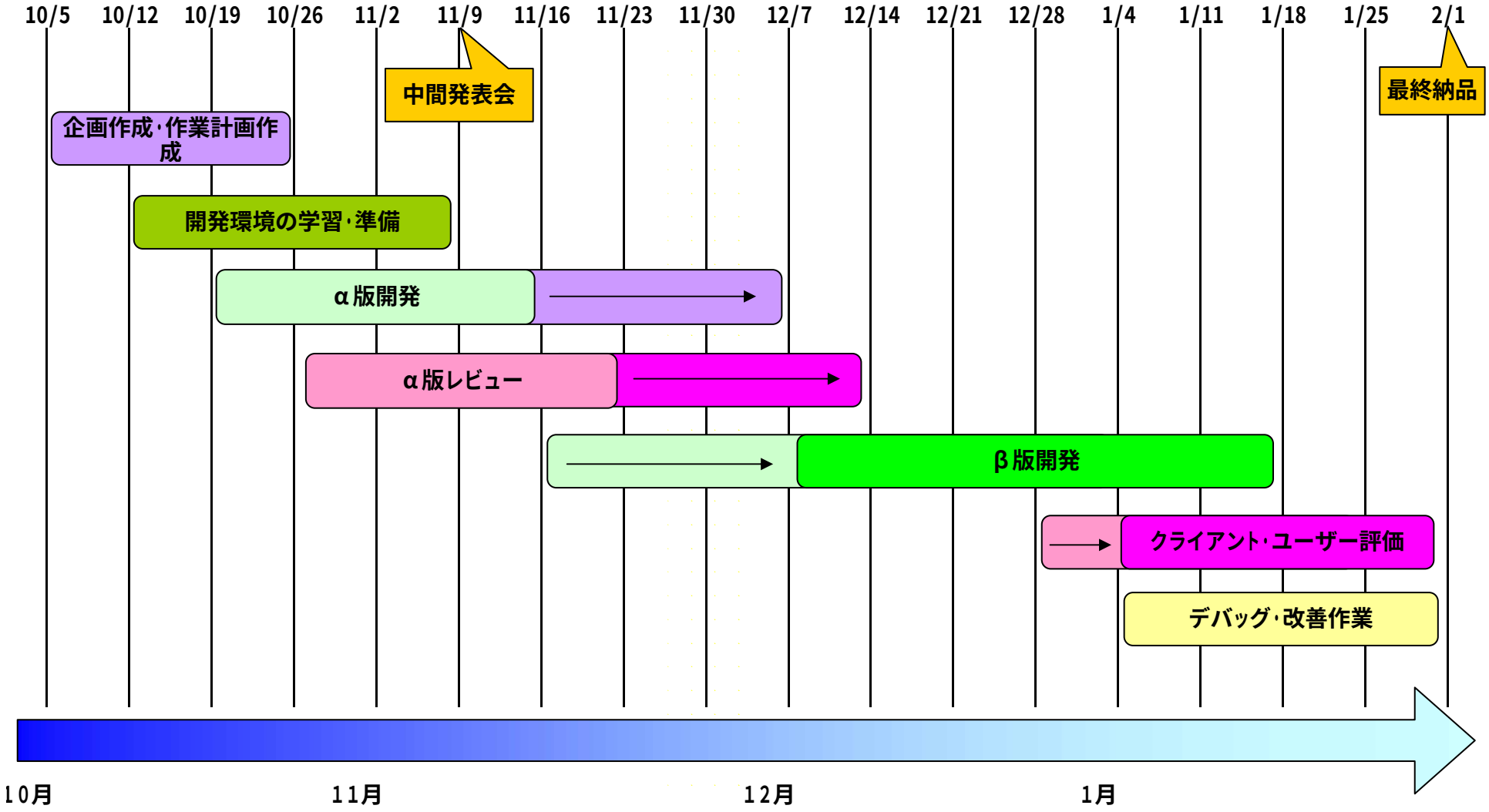
作成日：2006/10/25
作成者：菊地徹也

さうんどおんりぃ2　WBS



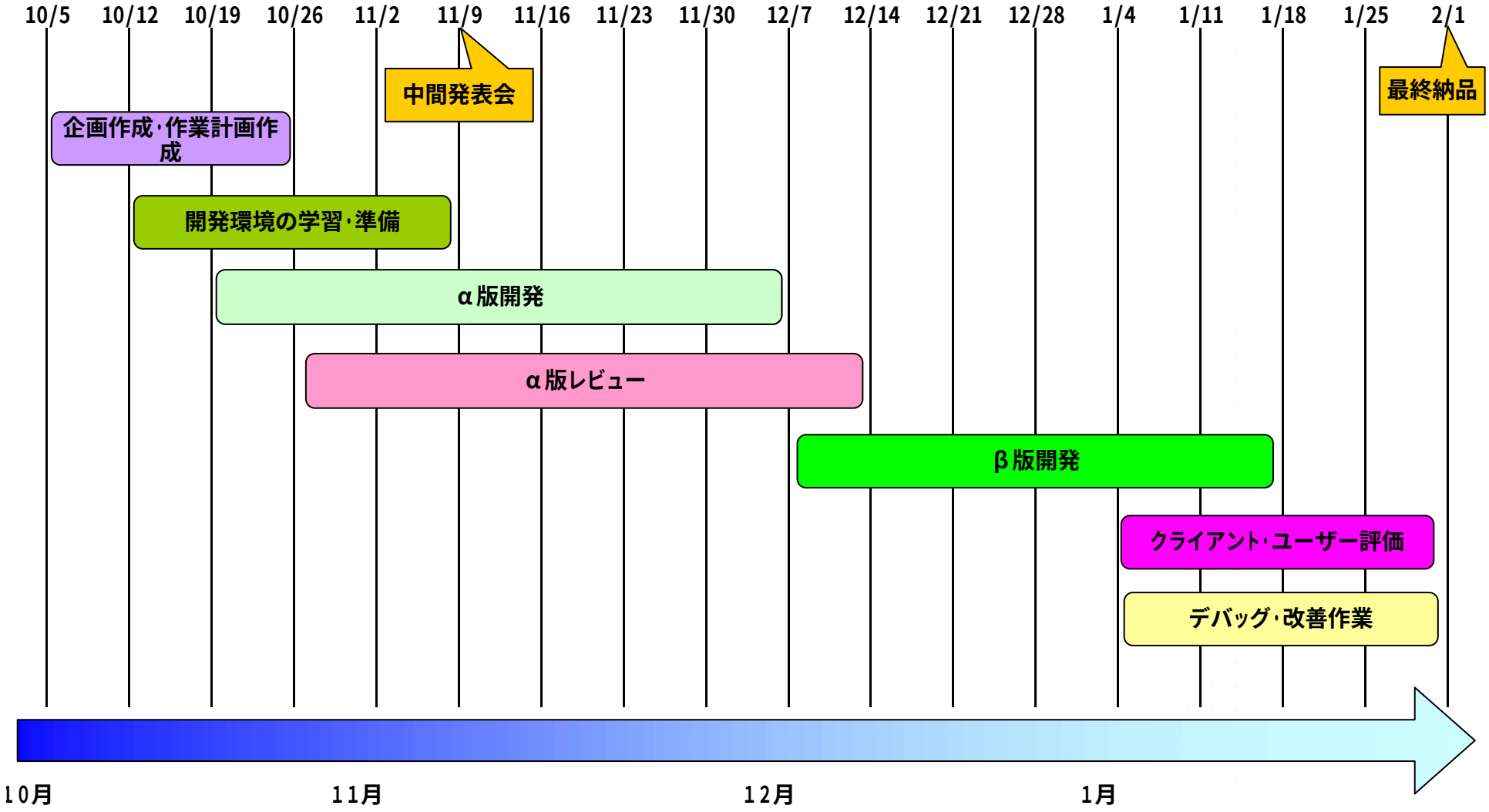
作成日：2006/11/30
作成者：菊地徹也

さうんどおんりぃ2　WBS



作成日：2006/11/30
作成者：菊地徹也

さうんどおんりぃ2　WBS



2007/01/31
さうんど おんりぃ
最終報告書

見積もりと実績

作業時間報告

第2週	10/5～10/11		
チーム	計1.5h	作業内容	備考
10/5	1.5h	ミーティング	橋山欠席
菊地	計1.5h(0.0h)		
橋山	計1.0h(1.0h)		
10/11	1.0h	第2回進捗報告会資料作成	
藤原	計2.5h(1.0h)		
10/11	1.0h	さうんどおんりぃ前作のソース確認	

第3週	10/12～10/18		
チーム	計2.5h	作業内容	備考
10/12	1.5h	ミーティング	
10/16	1.0h	ミーティング	メンバーのみ
菊地	計1.5h(0.0h)		
橋山	計4.0h(1.5h)		
10/15	0.5h	企画資料作成	
	1.0h	環境整備	
藤原	計4.5h(2.0h)		
10/12	1.0h	環境整備	
10/18	0.5h	企画資料作成	
10/18	0.5h	第3回進捗報告会資料作成	

第4週	10/19～10/25		
チーム	計2.5h	作業内容	備考
10/19	1.5h	ミーティング	
10/23	1.0h	ミーティング	メンバーのみ
菊地	計4.5h(2.0h)		
10/25	1.5h	PM資料作成	
	0.5h	Wikiの整備	
橋山	計7.0h(4.5h)		
10/21	3.0h	音記憶ゲーム作成	
10/25	0.5h	Wikiの整備	
	1.0h	第4回進捗報告会資料作成	
藤原	計6.0h(3.5h)		
10/22	0.5h	企画資料作成	
10/25	3.0h	聖徳太子ゲーム作成	

第5週	10/26～11/1		
チーム	計3.0h	作業内容	備考
10/26	2.5h	ミーティング	
10/30	0.5h	ミーティング	メンバーのみ
菊地	計2.5h(0.0h)		
橋山	計13.5h(10.5h)		
10/28	5.0h	PSP勉強会資料作成	学習
10/29	4.0h	PHP勉強会参加	学習
10/31	0.5h	南雲さんとのミーティング調整	
11/1	1.0h	C++勉強	学習
藤原	計13.5h(10.5h)		
10/27	6.0h	VS・デザインパターンの勉強	学習
10/28	4.0h	VS・デザインパターン勉強会資料作成	学習
11/1	0.5h	第5回進捗報告会資料作成	

第6週	11/2～11/8		
チーム	計12.5h	作業内容	備考
11/2	1.0h	ミーティング	
11/4	3.0h	CRI Audioについての勉強	メンバーのみ・学習
	3.0h	ペアプログラミング	メンバーのみ
11/6	2.0h	南雲さんとのミーティング	
11/7	2.5h	ミーティング	
11/9	1.0h	中間報告会リハーサル	
菊地	計9.0h(2.5h)		
11/6	1.0h	資料修正	
11/7	1.5h	中間報告会資料作成	
橋山	計27.5h(15.0h)		
11/1	4.0h	PSP勉強会資料作成	学習
11/3	1.0h	CRI Audioについての勉強	学習
11/5	2.0h	VS・C++の勉強	学習
	6.0h	Sound ShootをC++で作成	
11/6	1.5h	PSP勉強会パイロットテスト	学習
11/7	0.5h	中間報告会資料作成	
藤原	計26.0h(13.5h)		
11/5	2.0h	VS・C++の勉強	学習
	4.0h	Command InputをC++で作成	
11/6	2.0h	デザインパターン勉強会資料作成	学習
11/7	0.5h	中間報告会資料作成	
11/8	2.0h	デザインパターン勉強会資料作成	学習
11/8	3.0h	Command Input作成	

第7週	11/9～11/15		
チーム	計0.5h	作業内容	備考
11/9	0.5h	チームミーティング	
11/13	1.0h	メンバーミーティング	
菊地	計0.5h(0h)		
橋山	計13.5h(12.0h)		
11/14	1.0h	中根さんとのスケジュール調整	
	3.0h	PSP勉強会資料作成	学習
11/15	8.0h	PSP勉強会資料作成	学習
藤原	計16.0h(14.5h)		
11/10	3.0h	デザパタ勉強会資料作成	学習
11/13	2.0h	デザパタ勉強会資料作成	学習
11/14	4.0h	デザパタ勉強会資料作成	学習
11/15	5.5h	デザパタ勉強会資料作成	学習

第8週	11/16～11/22		
チーム	2.5h	作業内容	備考
11/16	1.5h	チームミーティング	
11/20	1.0h	メンバーミーティング	メンバーのみ
菊地	-		
橋山	計2.5h(0.0h)		
藤原	計5.5h(3.0h)		
11/22	3.0h	ユードーで音素材集め	

第9週	11/23～11/29		
チーム	5.0h	作業内容	備考
11/27	1.0h	メンバーミーティング	メンバーのみ
	4.0h	音環境作成	メンバーのみ
菊地	-		
橋山	計8.0h(3.0h)		
11/27	1.0h	中根さんとのスケジュール調整	
11/29	2.0h	企画書作成	
藤原	計9.0h(4.0h)		
11/29	4.0h	ForestWalk実装(対象の音が回転するように)	

第10週	11/30～12/6			
チーム	予定	実績	作業内容	備考
11/30	-	1.0h	ミーティング	
12/4	1.0h	1.0h	ミーティング	メンバーのみ
菊地	-			
橋山	計8.5h(6.5h)	計5.5h(3.5h)		
12/3	0.5h	0.5h	ユーザーインタビュー調整	
11/30	1.0h	0.5h	音ファイルの修正	
12/6	1.0h	0.5	音関連のプログラムの修正	
12/3	1.0h	1.0h	ギミック造り(企画)	
12/6	1.0h	0.5h	音の準備	
12/6	1.0h	0.5h	音データ修正の依頼	
藤原	計11.0h(9.0h)	計15.0h(13.0h)		
11/30	2.0h	2.0h	音関連のプログラムの修正	
12/2	-	6.0h	音データの準備・バグの調査・リファクタリング	
12/3	2.0h	3.0h	虫の実装	
12/6	3.0h	3.0h	虫を取る	
12/6	1.0h	1.0h	発表資料作成	

第11週	12/7～12/13			
チーム	予定	実績	作業内容	備考
12/7	1.0h	1.0h	ミーティング	
12/8	0.5h	0.5h	ユーザーインタビュー実施	
	1.0h		ユーザーインタビュー分析	
12/11	1.0h	-	ミーティング	メンバーのみ
菊地	3.5h(2.5h)	-		
	2.0h	-	スケジュールの作成	
	0.5h	-	企画書のレビュー	
橋山	計15.5h(12.0h)	8.5h(5.0h)		
	2.0h	2.0h	企画書修正・仕様書作成	
	3.0h	-	音の聞こえ方を調整する	ペンディング
	3.0h	-	バグ修正	
	1.0h	1.0h	音ファイルの作成	
	1.0h	-	横浜市立盲学校との連絡	
	1.0h	1.0h	CRI・ミドルウェアに質問する	
	1.0h	1.0h	発表資料作成	
藤原	計15.0h(11.5h)	9.5h(6.0h)		
	0.5h	-	企画書のレビュー	
	(3.0h)	-	オブジェクトを動かす	ペンディング
	(2.0h)	-	音に高低差を付ける	ペンディング
	3.0h	-	音の聞こえ方を調整する	ペンディング
	3.0h	2.0h	バグ修正	大きいcsbファイルも読み込めるようにした
	4.0h	4.0h	ユーザーインタビューの結果を反映させる	川を線上に並べる・虫を捕まえるときの音(バグ有り)

第12週	12/14～12/20			
チーム	予定	実績	作業内容	備考
12/14	1.0h	1.0h	ミーティング	
12/14	1.0h	1.0h	足音・川の実装	メンバーのみ
12/15	4.0h	4.0h	ペアプロ(設計方針について・バグの修正)	メンバーのみ
12/18	1.0h	-	ミーティング	メンバーのみ
菊地	11.0h(10.0h)	-		
	2.0h	-	スケジュールの作成	
	8.0h	-	PM資料の作成	
橋山	計19.0h(12.0h)	-		
	2.0h	1.5h	音の準備	
	1.0h	-	サラウンドスピーカーの準備	
	1.0h	-	企画書・状態遷移図の修正	
	1.0h	0.5h	横浜市立盲学校に連絡	
	0.5h	-	武藤先生に連絡	
	-	2.0h	C++の勉強	勉強
	-	4.0h	リファクタリング	
	-	2.0h	川に入ったときの足音の実装	
	-	2.0h	音量の修正	
藤原	計15.5h(8.5h)	11.0h(4.0h)		
	0.5h	-	企画書・状態遷移図のレビュー	
	3.0h	-	オブジェクトを動かす	
	2.0h	-	音に高低差を付ける	
	3.0h	-	音の聞こえ方を調整する	
	-	4.0h	文字表示関連のバグを修正する	

第13週	12/21～12/27			
チーム	予定	実績	作業内容	備考
12/21	1.0h	1.0h	ミーティング	
菊地	-	-		
橋山	7.0h(6.0h)	-		
1.0h	-	企画書・状態遷移図の修正		
2.0h	-	音の聞こえ方の修正(音量・高低)		
3.0h	-	音声の録音・実装		
藤原	7.0h(6.0h)	-		
3.0h	-	オブジェクトを動かす		
3.0h	-	制限時間の導入(朝→夜)		

第14週	12/28～1/3			
チーム	予定	実績	作業内容	備考
12/29	-	3.0h	ミーティング・ペアプログラミング	
菊地	-	-		
橋山	-	3.0h(0.0h)		
藤原	-	3.0h(0.0h)		

第15週	1/4～1/10			
チーム	予定	実績	作業内容	備考
1/7	-	6.0h	設計・バグ修正の相談	
菊地	-	-		
橋山	-	14.0h(8.0h)		
1/6	-	6.0h	音の録音	
1/10	-	2.0h	進捗報告会資料作成	
藤原	-	-		

第16週	1/11～1/17			
チーム	予定	実績	作業内容	備考
菊地	-	-		
橋山	-	27.0h(27.0h)		
1/12	-	3.5h	GamePlayStateが持っていたSoundPlayerをPlayerクラスに移譲	
1/14	-	1.5h	処理が重くなっていた問題を調査	
1/14	-	2.0h	音がフェードアウトするようにした	
1/14	-	1.0h	虫が逃げる確率を求める関数の周りを抽象化	
1/14	-	1.0h	WalkingState.h→GameStates.hに変更	
1/16	-	2.0h	一度捕まえた虫を2度以上捕まえられてしまうバグを修正	
1/16	-	1.5h	テキスト関連のメモリ操作を修正	
1/16	-	4.5h	GamePlayStateをAfternoonStateとNightStateに切り分けた	
藤原	-	-		

第17週	1/18～1/24			
チーム	予定	実績	作業内容	備考
菊地	-	-		
橋山	-	34.5h(34.5h)		
1/18	-	1.0h	虫を捕まえる関数を虫自体が持っていたので修正	
1/18	-	2.0h	結果画面の文字表示を実装	
1/18	-	1.5h	虫ごとにポイントを追加	
1/19	-	1.5h	soundList.xslを作成	
1/19	-	1.5h	必要な音を録音	
1/20	-	2.0h	時間帯が変わったときの音を実装	
1/20	-	2.0h	結果画面におけるスコアの読み上げ機能を追加	
1/21	-	2.5h	音ファイルを追加	
1/21	-	2.5h	結果画面で捕まえた虫とスコアを音声で読み上げるように修正	
1/21	-	1.5h	タイトルとその遷移を追加	
1/21	-	1.5h	画面が切り替わる際の音を追加	
1/21	-	1.5h	各種設定を変更してバイナリを作成	
1/22	-	5.0h	最終報告書作成	
1/23	-	8.5h	todoリストの優先度高のものを修正	
藤原	-	35.0h(35.0h)		
1/18	-	5.0h	昼と夜の状態の切り替えを修正	
1/18	-	5.0h	フェードイン，フェードアウトの処理を修正	
1/19	-	2.0h	フェードイン・フェードアウトの修正	
1/19	-	2.0h	昼・夜ごとのサウンド作成クラス(SoundFactory)を作成	
1/20	-	1.0h	リプレイできない問題を解決	
1/20	-	0.5h	floatをdoubleに直す	
1/20	-	1.0h	GamePlayState内のいらない変数・関数を整理	
1/20	-	1.0h	デストラクタを記述	
1/20	-	0.5h	警告を消す	
1/20	-	1.0h	虫が色々な方向に逃げていくように修正	
1/20	-	1.0h	フォントのクローズでバグる問題を修正した	
1/20	-	1.0h	夜は自然物や風の音などもフェードアウトするように修正	
1/21	-	1.5h	虫が重なっていたら，一つの虫のみを捕まえるように修正	
1/21	-	1.0h	フェードイン，フェードアウトの処理をGamePlayStateのpassTime関数内で行うよう修正	
1/21	-	1.0h	keyStateを使わないように修正	
1/21	-	1.5h	虫を取り損ねても逃げないことがあるバグを修正	
1/22	-	5.0h	最終報告書作成	
1/22	-	1.0h	実装のTODOリストの作成	
1/22	-	1.0h	設計の見直し	
1/22	-	1.0h	チュートリアルの追加	
1/23	-	1.0h	チュートリアルの実装	

第18週	1/25～1/31			
チーム	予定	実績	作業内容	備考
菊地	-	-		
橋山	-	45.5h(45.5h)		
1/25	-	1.0h	アンケート作成	
1/25	-	5.0h	HowToPlayStateの実装	
1/26	-	5.0h	最終報告書作	
1/27	-	0.5h	大岩研・南雲さんにアンケート回答の依頼	
1/27	-	5.0h	最終報告書作成	
1/28	-	5.0h	最終報告書作成	
1/29	-	8.0h	最終報告書作成	
1/30	-	8.0h	最終報告書作成	
1/31	-	8.0h	最終報告書作成	
藤原	-	42.5h(42.5h)		
1/25	-	1.0h	アンケート作成	
1/25	-	5.0h	TutorialStateの実装	
1/26	-	4.0h	最終報告書作成	
1/27	-	0.5h	松田さんに盲学校ユーザインタビューを依頼	
1/27	-	4.0h	最終報告書作成	
1/28	-	4.0h	最終報告書作成	
1/29	-	3.0h	横浜市立盲学校にてユーザーインタビュー実施	
1/29	-	7.0h	最終報告書作成	
1/30	-	7.0h	最終報告書作成	
1/31	-	7.0h	最終報告書作成	

2007/01/31
さうんど おんりぃ
最終報告書

ミーティングログ

10/05会議まとめ

参加：菊地　藤原

■　プロジェクト名の決定

前回のプロジェクトを引き継ぎ進めて行くということで、前回と同様の
「さうんどおんりぃ」と決定いたしました。

■　プロジェクトの目的の決定

高品質のゲーム作りを通じて、プロジェクト進行、ゲーム製作を勉強していく

を今回のプロジェクトの目的として進めていきます。

■　ML名の決定

「sonly」と決定いたしました。

■　次回までの各自課題

まずはゲームを作成する目的が無ければいけませんので、ゲームの企画の作成をお願い　します。

自由なサウンドの環境を扱うために、C言語を利用し、作成を進めていくことになるか　と思いますので、各自言語の勉
をお願いします。

2006/10/06　作成：菊地

参加者：菊地，橋山，藤原

・ゲームの企画
前回の開発を参考に，映像のないゲームを2本（継続，新規）で作ってみる
そのうち，良かったものを採用して，開発を継続していく

メンバー1人ずつが1本ずつゲームを作って行く．

映像のないゲームは既存のものと比較ができないので，同時に2本走らせて
比較しながら開発をしていくのはどうか？

今あるプロトタイプをベースにして，別のものを作っていく．

前回大変だったこと
　サラウンドの再生（技術的な問題）
　ユーザーインターフェースについて

・スケジュール
10月12日～11月9日：それぞれ，ゲームを1本ずつ作る（α版）
　10月19日：企画決定
　10月26日：設計・実装
　11月2日：テスト
　11月9日：中間発表会

11月9日～12月21日：ゲームを絞って，1本にする（β版）
　11月16日：α版レビュー
　11月23日：企画統合
　11月30日：設計
　12月7日：実装
　12月14日：実装
　12月21日：実装

1月11日～1月末：ユーザー評価，クライアント評価，テスト（最終版）

毎週ごとにプロジェクト内で評価を行う．

・コミュニケーション計画
ミーティング
　木曜4限，研究室にて（PM，メンバー）
　月曜5限，研究室にて（メンバー）

連絡方法（育実君はメールが使えるように！！）
　メーリングリスト
　Wiki
　携帯

ファイルの共有
　Wiki
　WebDAV
　SVN（未定）

・環境の整備
音声は南雲さんに頼むこともできる
Visual Studio .NET2003（研究室にある）
勉強は各自で前回のソースコードを見ながら行う

・目的の再検討
ユーザーが繰り返し遊んでくれるような面白いと感じられるゲームを開発する．
その開発を通じて，プロジェクト進行，ゲーム製作を勉強していく

・月曜までのタスク
企画を考えてくる（各自）
ソースコードを読んでくる（各自）
作業時間のログを取る（1日単位）

19：40開始

【ソースについて】
　二人とも多少読んだ程度
　藤原は環境設定が終了（VS2003にて開発が可能に）

【企画】
　橋山…宝探しゲーム（対戦アクション）
　　これが作りたかった

　藤原…宝探しゲーム（スコアゲー）
　　消去法で，これしか思いつかなかった

二人の企画は，共に現状の映像のないゲームを再利用して作ることが出来る
→現状の映像のないゲームを参考に，中間報告までに1本ずつ作る

南雲さんに説明するべき？→ミーティングが必要

【進捗報告】
　藤原が担当
　今週やったこと（企画，スケジュール）
　来週やること（南雲さんとのミーティング，設計）

【次回以降のミーティングについて】
　月曜の19:30〜か，火曜の11:30〜

20：00終了

15:25開始
参加者：菊地，藤原，橋山

・企画について
　二人の企画にもう少し違いが欲しい（特にサウンド面で）
　　映像が無くても出来そうな企画
　　　旗揚げゲーム
　　　落ちてくる障害物をよけるゲーム
　　　リズムゲーム
　　　モグラ叩き
　　　サウンドノベル
　　　ボウリング
　　　テニスゲーム
　　　ボクシング
　　　サッカー
　　　鬼ごっこ
　　　聖徳太子ゲーム
　　　記憶ゲーム
　　→以上のような細かいゲームを来週までに作ってみる
　二週間でゲームを作れるかどうか
　　設計は行わない
　　まだソースを詳細に見たわけではないので，どの程度リスクがあるのかがまだ不明
　　素材の収集が大変
　　音の聞こえ方の調整
　　サラウンド環境・・・ヘッドフォンが一つしかない
　　　もう一つヘッドフォンを提供してもらう？
　仕様書を作成する→企画書から詰めていく
　　共通部分についてはお互いが重複して作らなくてもよいようにしておく

・スケジュールについて
　既に決まっているものがあるので，表にする（菊地さん）

・中間発表までに作るゲームの完成度について
　簡単なミニゲームをいくつか作り，それらのおもしろさを比較検討する
　最低各自二本作成する

・作業中のコミュニケーション手段について
　メールで随時連絡を行う
　月曜の定例ミーティング
　メッセンジャー（ログを残しておく）
　電話（メモを取っておく）
　11月末までは橋山君は家にネット環境がない

・南雲さんとのミーティングについて
　現状報告も兼ねてミーティングを行う
　連絡は橋山君が行う

・来週までにやること
　ミーティングの調整及び実施
　企画書の作成
　C++の調査
　実装

・資料の共有について
　資料はWikiとメール両方で共有する
　資料を更新した場合は，Wikiの旧バージョンは残しておく（タイムスタンプを押す）

20:10開始

【ゲーム開発の進捗】
橋山→記憶ゲームをJavaで作成
藤原→上下左右ゲーム企画作成

橋山はゲームに声などを入れる
藤原は木曜までにゲームをJavaで完成させる

完成したゲームは，jar化してWikiにアップする予定

【テキストを読み上げて，Wavで保存するツール】

・楽SpeechS
http://my.vector.co.jp/servlet/System.FileDownload/download/http/0/306183/pack/win95/art/sound/rsp.exe

【南雲さんとのミーティング】
第一候補：30日（月）20時頃より湘南台にて

15:00開始

・ゲームについて
　中間報告までにC++で動くものを作る
　来週までに・・・
　　C++で作ってみる
　　藤原：今のゲームを発展させる
　　橋山：音が動くようなゲーム（アクション系）を作ってくる
　サラウンドを使うことを前提にしているので，C++で実現したい

・中間報告会
　PMと学生合わせて20分で発表
　質問が10分
　発表方法・内容
　　デモ・開発の経緯（二人それぞれが作ったものについて発表）
　　全体的なまとめ・今後の方針を菊地さんが発表
　　来週詳細に決める
　11/2〜11/9の間に全員でミーティングを行う（場所は大学）
　菊地さんの都合に合わせて日時を調整
　各自がゲーム作成を進める（できればC++，最悪Javaで）

・視覚にハンディキャップを持つ人たちの位置づけ
　α版のレビューに，中根さんに協力をお願いする
　β版の評価に，横浜市立盲学校の方々にお願いする
　　→橋山が林太郎さんに連絡する
　アイデアなどもインタビューしたらどうか，どういうゲームがやりたいか
　　→前回の結果が残っている．前回のインタビューでは電車でGOがやりたいという意見だった．

・南雲さん
　現状では10/30を予定している
　　→菊地さんが厳しい
　金沢さんも参加したがっている
　　→明日中に菊地さんが予定を聞く
　できればプロジェクト定義書を渡してからにしたい
　来週末もしくは11月頭に行う
　場所はユードー本社がよいのでは
　遅くとも11/9までには一度ミーティングを行う

16:30中断

19:00再開

・α版，β版の定義
　みんなのイメージが違っている
　α版…ゲームの方向性がわかるもの
　　性能や機能、使い勝手に対する要望を受け入れるための開発初期のもの
　　完成度については，20〜30%程度
　　バグが潜んでいる可能性はあるものの，ゲームのイメージがわかる程度に動作しているもの
　β版…テスト前の段階，ほぼ完成している状態
　　正式版の機能を一通り備えた完成品に近いもの
　　完成度については，80〜90%程度
　　ゲームの根幹に関わるようなバグは解消されていて，テストができるもの

・評価基準，評価項目
　盲学校でのユーザインタビュー
　起動時間・起動回数を内部的に記録しておく
　横浜市立盲学校に環境を構築し，一週間程度プレイしてもらう
　　→サラウンドヘッドフォン・スピーカー
　ユーザのアンケートの質問
　　5段階くらいの選択式
　　　普段からゲームをしているか
　　　もう一度プレイしたいと思うか
　　　面白いかどうか

　　→どこが面白かったか
　評価基準
　　面白いと感じた人が7割以上
　　→前回は5割だった
　　もう一度やりたいと思った人が8割以上
　　→前回は7割だった
　　ユーザをどこに置くのか
　　障害者と健常者の両者が面白いと言うようなゲーム

・プロジェクト定義書，プロジェクト憲章
　来週までに作らなくてはならない
　ユーザ等を決める必要がある
　できれば来週の月曜日までに用意する

・作業ログ
　勉強にかかった時間も別途記録する

20:00終了

19:30開始

・南雲さんの都合について
　11月6日（月曜）の18時以降でアポを取れるか
　橋山が電話→繋がらず
　後で改めて電話する

・今日までの活動について
　二人とも忙しく，ほとんど作業ができなかった

・木曜までにすること
　C++でゲームを作る
　無理そうだったら，Javaで進化させる

・作業環境について
　橋山→11/6まで家にネット環境がない

19:40終了

出席者：菊地，橋山，藤原

15：35～

【中間報告会】
・流れ
①プロジェクトの概要説明（2分）：菊地

②全体のスケジュール（3分）：橋山
全体のスケジュールの説明（活動内容）
今の進捗状況

③開発の経緯（4分）：藤原
メンバー各自がそれぞれゲームを作ることにした
クライアントの反応

④デモ（合わせて5分）：橋山，藤原

⑤今後の方針（5分）：菊地
中根さんへの評価（評価項目・評価基準について）
ゲームの作り方の話

資料作成期限：11月6日（月）23：59まで（Wikiにあげる）
リハーサル：11月7日（火）17：00～　研究室にて

【C++の学習について】
11月4日（土）：10：00～16：00　研究室にて
目標はサラウンド対応のゲームを1つ作る

【プロジェクト定義書の作成】
やりました

【南雲さんのミーティングについて】
11月6日（月）19：00　横浜ユードー本社にて
土曜日に作ったゲームを持っていく
プロジェクト定義書のレビューをしてもらう
今後の開発について，体制などを確認をする

【中根さんに連絡】
橋山がやっておきます

～16：30

経緯の説明

・人に使ってもらえるというゲームはどういうものか？

環境，用意してもらいたいものなど

元ファイルがステレオだった→モノラルでいきましょう
ステレオをサラウンドにすると音が歪んでしまう

合成音声
　マッキントッシュ（マッキントーク）

人間の声は個性が出てしまう

ーサウンドシュートー
効果音次第
　怖いものが迫ってくる
　聞いていると嫌な音　などにする

　絶えるほど得点が高くなる

ー聖徳太子ー
　読み上げ音を変える
　皆で読み上げる？

ゲームのアイデア
　迷路ゲーム
　既存のゲームを音だけにしてみる
　右脳系のゲーム

　既存の概念にこだわらずに
　　商店街のサウンドスケッチなど

演出面の課題
　訓練が必要
　音をそのまま使う

ブレストをして企画だす

プレゼンテーションで自信を持つこと！

18:00開始

・企画について
絶対音感ソフト
　どうサラウンドに結びつけるか
　ゲームというよりはツールっぽい？
キムス
　面白そう
　振ったり，動かしたりというところをどうするのか
　動かしたり，ものにぶつかったりというのが技術的に難しそう
ハリーポッター
バーチャルシミュレーション
　電車にGOに近い？

最終版はミニゲーム集にするのか，一本化するのか
一本にするとしたら，何が面白いかを考えなくてはならない
もう少しα版の期間を延ばして，企画を色々出してみたい
多数の企画をブレストする
南雲さんの商店街のやつは面白そう
　自分がどこにいるのかということを認識して動く
　フィールドワークとして，横浜中華街などで録音をする？
　中華街を紹介する作品であり，ゲームでもある
　録音については南雲さんに協力を依頼する
　中華街だとにおいが重要だが，それは再現できない
　商店街のある場所に行くと，ミニゲームが遊べる
今まで作った迷路ゲームに色々要素を追加していくと面白くなるのではないか

・来週までにすること
今までの案の中で，やりたいものを詰めてくる（月曜まで）
今週は企画を中心に行う

18:15

ーアイデアー

・音キムス
・絶対音感育成ゲーム
・ハリーポッター
・ヴァーチャルシミュレーション（車とか）

荒木さんアイデア

目的：
目の見えない友達がいて、子供が生まれた
そこで、健常者の子供と目の見えない親が全くストレスを感じずに遊べるゲームがあるとよいのではないかと思った
目の見えない人は立体的に音を捉えており、その感覚を追体験できるようなゲームがあると面白そう

どこにいるかがわかるようなもの
　森の中、町の中、コンピュータの中、体の中、海の中　が　音で分かる
　夏休み　どこにいるかがわかる

実際に一緒に遊べないが、音だけの世界だったら遊べる
現実的には健常者と一緒に遊ぶことが難しい
音で一緒に海で遊ぶということが、何の抵抗も無く楽しめる
現実にある楽しい時間を目が見える見えないなどに関わらず楽しめる
（レジャーに出かける、など）

現実にはボールの位置や人の位置や、人の「もっと右」などの声を海で聞き分けることが困難だが、音だけで判別できるようにする

具体例：
・虫取り
・雪合戦
・スイカ割り
・ビーチバレー
・焼肉
・凧上げ
・人間の体内を探検する
　自分が小さな医者になって病原菌と戦う
・人ごみの中で迷子の子供を捜す
・ピンポン
・肝試しをして、何かを取ってくる
・サマーキャンプでオリエンテーリングをして、クイズに答える
・料理
・釣り
・宇宙空間で敵と戦う
・野良猫になって、夜、人間から食べ物を奪う
・蚊を叩く
・射的
・恋人の足跡を聞き分ける

二人で楽しめる（対戦）ゲームがあったら面白そう
ゲーム内で同じ空間を共有する必要がある
荒木さん的には、日常と同じものがよい
かつ、日常で行うには制約上実現が難しいもの
音空間の中であれば、実現可能になるもの
「僕の夏休み」は夏の雰囲気が緻密に再現されていた
音の専門化が必要かも
夏休みとか体の中が魅力的
夜の街で生きている小さな生き物
　車の危険を察知してよける　など

音を音に当てると別の音に変わり、何か（音楽など）を作っていくパズルゲーム
最初はいやな音だが、だんだんといい音になっていく

最初は波形が一定の音

そこに何かを当てて、波形を変えていく
作るべき波形を最初に見せて、それに近づけていくパズルゲーム

ーアイデアをまとめるー

藤原
　・射的
　・蚊を叩く
　・ピンポン
　・虫取り

橋山
　・蚊を叩く
　・焼肉
　・虫取り

虫取りになりそうだ．

【虫取り】
　音空間を大事にしたい．
　　部屋の中で蚊をとる
　　夏にセミを捕まえる
　　秋に蛍を捕まえる　　etc

　音で空間を再現することが先決？
　　その後にゲームとして考えていく

　必要なこと→音の分析

設定
　季節：夏
　場所：森
　時間：昼

　虫取り網で，セミを捕まえる

必要な音
　森
　　葉っぱ
　　草

　セミ
　　鳴き声
　　飛ぶ音

　虫
　　鳴き声

　虫取り
　　歩く音
　　　草の上
　　　土の上
　　虫取りの音

今後の予定
　南雲さんと音の相談をする

チームの学習目標
　ゲーム開発のプロセスやプロジェクトの進行を学習する．
　それによって，ソフトウェア開発手法との違いを学習する．
　それをメンバーごとに最終報告書にまとめる．

チームの挑戦
　売れるゲームを作りたい
　値段目標：500円以上

中間報告の反省
　前回との違いは…前回は技術調査，今回は品質重視
　終わりは？…タイムライン
　　納期が来るまで，作ったり壊したりする

　開発手法（たくさん作ってみる）ことの意義は？
　　これから考える

15:30〜

反省の残り（開発手法の意義）
　比較検討で，面白い部分を集めてみる
　どういうことができるか（技術調査の続き）

ゲーム開発進行について
　中間報告会以降は1本の筋を通した企画を考える

　橋山
　　企画からやりたいと思っていたが，人の企画を集めて面白そうなものをカスタマイズしてオリジナルのものを作りたい
　　インドアとアウトドアの融合，音の聞こえ方に興味がある

　藤原
　　企画にはあまり興味はない（現在の企画で良い）
　　どちらからというと，与えられたものを作ることに興味がある

　企画をPMから提供する？
　他の人から企画を提供してもらう

企画について
　軸になる企画…虫取り

　・企画を詰めてから音を集めるのではなく，音を聞いてから企画を決める
　・ゲームとしての虫取りだけではなく，湘南台のサウンドスケッチを行う
　　学校紹介のような作品兼ゲームに発展できるかも？

　環境や設定は作りながら面白いと思うものに随時変えていく

　企画書を橋山が作る（月曜）

来週までに
　来週は進捗報告会がない
　月曜までに，南雲さんに必要な音をお願いする
　音取りの算段をつける
　　効果音CDを聞きまくってみる
　　Webで探してみる
　　生録の場所，時間を決める（大学，湘南台近辺など）

　藤原：ゲームの基礎（プレーヤーが動く，虫を取るなど）を開発
　橋山：音の聞こえ方を評価・改良

〜16:10

【ゲームの企画】
時間がなくて作れなかった
橋山ができるだけ早く作る
　森の中を散歩しながら，虫を捕まえるようなゲーム

【効果音CDから使えそうな音を探した】
小川のせせらぎ，虫の声，鳥の鳴き声などは効果音CDから入手

手に入っていない音
　森の音（生録？）
　　木々の葉がこすれ合う音など
　足音
　　歩く音
　　葉を踏みしめる音

欲しい音リストを作ったので，南雲さんに発注する
手に入らない場合は，生録音を行う（場所は未定）

【中根さんへのインタビュー】
12月1日（金）に行う
日程の調整はできているので，確認のメールを橋山が送る

インタビュー実施までに，音が移動して森の中を散歩しているような雰囲気が出せるものを作成する（臨場感があるかどうをレビューしてもらう）

【今後の予定】
1. 橋山が火曜までに企画を作成する
2. 藤原が橋山の企画に沿って，今ある音で簡単な音環境を作る（人間は止まっている）
3. 音の聞こえ方のレビューを重ねて，音が自然に聞こえるようにする
（音の変化は，さうんどおんりぃなどを参考にする）

音環境をある程度作成してから，ゲーム作成に取り掛かる

森を散歩しているような環境を音だけで作る
　南雲さんのおかげで，音はいろいろ揃った

【具体案】

・移動方法
　基本的に移動は前進のみ
　左右に向きを変えることが出来る
　後退する場合は，後ろを向いてから前進する

・音の聞こえ方
　前進することで，音量を調整する
　左右に向きを変えることで，音の定位が移動する
　（例：後ろを向くと，定位が180度変わる）

・環境音
　森の音をループで再生する
　　自然に聞こえるように

　複数の森の音を使う
　　ある程度距離を移動したら，今までの森の音を小さくして新しい森の音を鳴らす
　　※同じ音だと聞き飽きてしまう可能性がある

　環境音の定位や音量の変更
　　別の森の音に変えるときにのみ音量を変更する
　　向いている方向で定位を変更する

・効果音
　鳥の鳴き声
　　自分の上空を一定間隔で移動する（例：左手前から右奥へ）

　滝・川の音
　　近づくと大きくなる

　虫の音
　　近づくと大きくなる
　　虫によっては，近づくと逃げる

　足音
　　自分の足元から一定間隔で聞こえる
　　音量や定位は基本的には変更しない
　　歩く地面の状態によって変える
　　　まずは，森を歩く音を川を歩く音の2音

【ゲーム作り（環境作り）】
　森の音を鳴らす
　　森の音をループ再生する
　　音のつなぎ目でブツブツ切れる→ループの方法に問題あり？
　　　案1：1つの音の再生が終わりそうになったら，他の音を再生する
　　　問題点：音を切り替えることで，定位やボリュームの設定がめんどそう

　　　案2：再生時間の長い音を使うことで，切れ目を少なくする
　　　問題点：根本的な解決にはならない．ファイルサイズが大きくなる

　　　案3：ループ方法を改良する（再生が終了する前に，同じ音の再生を開始する）
　　　問題点：技術的な問題がある．

　　現状は，フェードインとフェードアウトをすることで，ブツブツ切れるのを回避している

　川の音を鳴らす
　　川の音をループ再生する
　　森と同様に，ループについての問題があり

　　最初は聞こえないが，ある程度歩くと，川の音が聞こえてくる

　効果音について
　　風の音
　　チンパンジーの音

　音の移動
　　プレーヤーが向きを変えることで，音を回転させる
　　円周上を音が移動するような計算式を考える

【ユーザーインタビューについて】
　予定調整中（来週実施）

・スケジュールについて
　α版の開発期間・レビュー期間が延びた
　12/7までに方針を固めて，β版の開発に移行する

・前回の失敗について
　音がしょぼかった
　ゲームの先行きが見えなかった
　それ自体ではゲームにならない
　メンバーの企画力が足りず，面白い企画が浮かばなかった
　あまり作業時間を取らなかったので，あまりたくさんのゲームを作れなかった
　前回と大して変わらないものしかできなかった

・前回の失敗から得たもの
　音が充実していることが重要
　企画はできる人に任せる（他の人の意見を取り入れる）
　作るものを明確にさせることが重要
　企画に対するモチベーションが足りなかった
　ゲーム作りのイメージはつかめた

・作業見積もり
　来週までの作業(計16.5h)
　・ユーザインタビュー(計3.5h)
　　調整(橋山・0.5h)
　　実施(メンバー・1.0h)
　　結果の分析(メンバー・0.5h)
　・実装
　　音関連のバグの修正(計4.0h)
　　音ファイルの修正(橋山・1.0h)
　　プログラムの修正(藤原・2.0h)(橋山・1.0h)
　　ゲームのギミック作り(計7.0h)
　　企画(橋山・1.0h)
　　音の準備(橋山・1.0h)
　　プログラム(藤原・5.0h)
　　虫の実装(藤原・2.0h)
　　虫を採る(藤原・3.0h)
　　（虫かごに入れる）
　　（時間の経過）
　　音データの修正→南雲さんに依頼(橋山・1.0h)
　・発表の準備
　　資料作成・デモの準備(藤原・1.0h)

・β版移行のタイミング
　12/7までにまとめた企画をもとに，β版に移行する
　中根さんのレビューが終了した時点で，α版完了とする
　レビューをもとに手直しを加えて，来週の発表会に臨めればベスト

【虫の種類】
・木に止まっている虫
　川などのオブジェクトと同じ扱い

・動き回る虫
　ランダムに動き回る

【虫を捕る】
虫の手前（完全に重なると音が聞こえなくなるため）でEnterを押すと，虫を捕まえられる．

捕まえた虫は，虫かごに入る→虫かごはプレーヤーが持っている．

【虫かご】
虫かごには何匹でも虫が入る．
虫かごからは捕まえた虫の声が定期的に聞こえる（ただし音量小さめ，定位は一定）

【現状の問題点】
なぜか音のファイルが読み込めない
→メンバーで原因を究明する

各自の作業
　橋山：音の準備
　藤原：虫，虫捕りの実装

【ユーザーインタビュー】
12月8日（金）：16：00～

【実行速度】
・調査の必要あり

【企画書・仕様書】
・企画に対するモチベーションが足りない

・現状の企画書を修正して，企画書兼仕様書を作成する
　画面構成，シーン構成，必要なリソース（音）について
　　→状態遷移図を作る
　状態数はタイトル，説明，ゲーム中，結果程度
　　→あまり複雑にしない
　作業分担を行う（ゲームメイン：藤原，その他：橋山）

【ユーザーインタビュー】
・明日，中根様に行う
　α版のゲーム内容について
・武藤先生が，虫の音に対する物理学の研究を行っている
　→連絡を取ってみる

【スケジュール】
・企画書，仕様書を元にもう一度スケジュールを引く

【音の聞こえ方について】
・音量や定位がつかみにくい
・ライブラリの中をいじることも検討する
　→現状をCRI・ミドルウェア社にメールして，相談する
・音の高低をつける（鳥の音など）
・オブジェクトを動かす（虫など）

【来週までの予定】

・ミーティング（5.0h）
　木曜ミーティング（メンバー：1.0h）
　月曜ミーティング（橋山・藤原：1.0h）

・企画関連（3.0h）
　企画書修正・仕様書作成（橋山：2.0h）
　企画のレビュー（菊地・藤原：0.5h）

・実装関連（18.0h）
　オブジェクトを動かす（藤原：3.0h）
　高低差をつける（藤原：2.0h）
　音の聞こえ方を調整する（橋山・藤原：3.0h）
　バグ修正（橋山・藤原：3.0h）
　音ファイルの作成（橋山：1.0h）

・ユーザーインタビュー関連（4.0h）
　中根様
　　ユーザーインタビュー実施（橋山・藤原：1.0h）
　　ユーザーインタビュー分析（橋山・藤原：0.5h）

　横浜市立盲学校との連絡（橋山：1.0h）

・技術調査関連（1.0h）
　CRI・ミドルウェアに質問する（橋山：1.0h）
　　実行速度について
　　CSBファイルの限界容量について

・その他（3.0h）
　スケジュールを引く（菊地：2.0h）
　発表資料作成（橋山：1.0h）

菊地：3.5h
藤原：15h
橋山：15.5h
計：34h

【インタビュー実施】
ーインタビューの趣旨の説明
ー前回との違いの説明（企画やゲーム製作の方針など）
ー操作の説明

音を聞き分けて，虫を捕まえることができた

【インタビュー内容】
ー質問・要望
説明の情報が必要
　マップの広さ
　キーの操作の問題
　　カニ歩きでも，向きを変えてもどちらでも良い

虫が動く
動物が動く
　一度に動くオブジェクトの数を制限する

慣れの問題でもある

足音が必要！
・フィードバックが欲しい

遊ばせ方の問題
　全体がどのくらいの大きさか分からない
　端に来たかどうか分からない

適当に歩いていて，何にもぶつからないことの不自然さ
　障害物の並びが目印になる

感覚的に理解しやすいものができる

臨場感
　ある程度のレベルではできている
　川が線上ではない

近づいていって，落ちるかなといったところで音が後ろへ来てしまう

今の状況
　近づいたとき音が大きすぎる
　定位の動かし方がよくない
　斜め右前でも斜め左前でも正面でも同じように聞こえる

　近づく
　　ボリュームの調整が必要である

ゲームギミックについて
　ゲームそのものとしては，イベントが欲しい
　何かを見つける

最初は，ある程度情報を与えるという前提で
　虫捕り網や虫かごを探すところから
　ある段階になったら，別の要素が必要

アクションに対して，積極的なフィードバックがあるとよい
　飽きさせないために

合成音声
　ペンタックスが出しているライブラリ
　合成サンプル

ロールプレイング的なものは，感情が入っていたほうがいい場合もある
感情云々で大きな問題がでることはない

【現在の問題点】

全体がどのくらいの大きさか分からない
　端に来たかどうか分からない
　障害物があると目印になる

適当に歩いていて，何にもぶつからないことが不自然である
　木にぶつかる
　草を掻き分ける

音の聞こえ方
　足音がない
　　歩いたことに対するフィードバックがない

　川に関する問題
　　川の音が1箇所から聞こえる（泉のようになっている）
　　　川は線上にあるべき
　　川に入れない（川に入ると，川の音が後ろになってしまう）

ゲームデザイン
　オブジェクトの動き
　　大量のオブジェクトが同時に動かない限りは，問題ない
　　動かなくても，そういうものだと理解すれば良い

　飽きる
　　イベントが必要
　　　最初は虫捕り網から探すなど
　　　虫捕り網は泉の傍に落ちているなど
　　　ある段階になったら別の要素をとりいれる

　説明音声が足りない
　　最低限ゲームについての説明が必要

　　合成音声
　　　ベンタックスが出しているライブラリが，現行で一番クオリティが高い
　　　ただし値段もそれなりにする
　　人の音声
　　　ロールプレイング的なものは，感情が入っていたほうがいい場合もある
　　　感情云々で大きな問題がでることはない

【β版開発に向けて】
（優先順位1）
足音の実装（森，川）
川に関する問題の解決

（優先順位2）
オブジェクトを動かす
自分のいる位置を把握するギミックを考える（障害物など）

（優先順位3）
音声の実装
イベントの追加

■先週のPMミーティングでの指摘
もう少し遊びを入れて欲しい
　企画＋α
　二人でアイデアを出して，とりあえず形にしてみる
　もう少し柔軟に作っていく
　虫の網を見つけるところから始める，等のイベントを追加する

発表はデモだけでいいのでは

■先週行う予定だったこと
バグの修正
ユーザレビュー

■今週の予定
土日に一日ペアで作業をする時間を作りたい
音源の修正
実行速度の問題の修正
発表はデモだけ
　変わったところを見せられるように
　虫を捕るところ
サラウンドスピーカーを研究室で用意できないか？
　値段を調査する
企画書・状態遷移を洗練させる
　状態遷移図で終了条件を入れる
実装
　端っこは環境音または音声などで知らせる
　デバッグ用に当たり判定を出してみては

■今後の見通しについて
冬休み中に優先順位2のタスクまでは達成したい

・面白さの追求
追加項目：
　競う要素をつける（点数など）
　虫捕りに失敗したら、虫が逃げる
　　遠いところで捕ると失敗する確率が高くなるなど
　時間の経過がわかるようにする
　　鐘を鳴らす
　　狼がほえる
　　夜が明けそうになったら鶏が鳴く
　タイトルをつける

　後回し：
　網を見つける
　虫かご内の虫が共食いをする

修正項目：
　虫をたくさん作る
　虫がいきなり消えるのはヘン
　　虫が画面外に逃げる
　　フェードアウトする
　自然物の音をもっと増やしたい
　夜行性の動物を追加する

・ユーザインタビュー
　ゲームの説明の仕方
　アンケートをどうするか
　　学校の先生に相談する

・バグ修正
　サウンドプレーヤーが複数ないと同じ音を同時に鳴らせない
　　川が不自然に聞こえる

・南雲さんに連絡
　南雲さんに現状を見せに行く（1/23まで）
　CriAudioライブラリの話
　1/20くらい
　盲学校のレビューにも参加してもらえたら参加してもらう

・ユーザレビュー
　1/23までに連絡を取る
　1/20くらいを目途に

・最終報告書の作成
　できれば23日くらいまでに終わらせたい
　土曜日くらいまでに目次を作る
　来週の木曜までに骨組みを作る

2007/01/31
さうんど おんりぃ
最終報告書

進捗報告会資料

さうんど おんりぃ2　進捗報告

2006/10/12

㈱インテム　菊池徹也（PM）
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

本日の報告内容

- 今日までに行ったこと
- 来週までに行うこと

今日までに行ったこと

- プロジェクト名の決定
  - 前回のプロジェクトを引き継いで進めて行くということで，メンバー間では前回と同様の「さうんど おんりぃ」と決定した
  - しかし，前期のプロジェクトと区別がつかなくなるので，最終的に「さうんど おんりぃ2」となった
- プロジェクトの目的の決定
  - 高品質のゲーム作りを通じて，プロジェクト進行，ゲーム製作を勉強していく

来週までに行うこと

- ゲームの企画決定
  - 前回の企画を拡張して，新しい企画を考える
  - 各自考えてきた企画を元に，具体的な内容を詰めていく
- コミュニケーション計画
  - 授業時間以外にミーティングを行うかどうか
  - ファイルの共有方法について（Wiki，WebDAV，SVN…）
- 環境の整備
  - 前期の製作環境の整備（VS2003，各種ライブラリなど）
  - C++の勉強（前期のソースコードが読める程度）

さうんど おんりぃ2　進捗報告
2006/10/19

㈱インテム　菊池徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

CreW
Creative Workspace

本日の報告内容

- 今週行ったこと
  - 環境整備
  - ゲームの企画書作成
  - 今後のスケジュール決定
- 来週までに行うこと
  - 南雲さんとミーティング
  - 設計

CreW
Creative Workspace

環境整備について

- Visual Studio2003にて開発を行う
- 藤原･橋山共にインストール済み
- 前回のプログラムをコンパイルして起動することができる

CreW
Creative Workspace

ゲームの企画について

- 藤原･橋山各々が中間報告会までに一本ずつゲームを作る
- 藤原･･･宝探しゲーム (スコアゲー)
  - 制限時間内にいくつの宝を集められるのかを競う
- 橋山･･･宝探しゲーム (対戦アクション)
  - CPUと一つの宝を奪い合う
- ゲームの基本部分は二人とも共通

CreW
Creative Workspace

今後の方針について

- 中間報告会までに最低二本のゲームを仕上げる
- 現状のソースを参考に，アイデアを形にしてみる
- その上で，藤原･橋山の企画の中からよいものを選ぶ，あるいは両者の企画を統合する

CreW
Creative Workspace

来週までに行うことについて

- 南雲さんとミーティングを行う
  - 必要があるかどうかを本日のミーティングで話し合う
- 設計を行う

CreW
Creative Workspace

さうんど おんりぃ2　進捗報告

2006/10/26

㈱インテム　菊池徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

本日の報告内容

- 全体スケジュールについて
- 今日までに行ったこと
- 来週までに行うこと
- 視覚にハンディキャップを有する人達との関わり方について

全体スケジュールについて

今日までに行ったこと

- 企画の再考
  - 先週考えた二人の企画は似通っていた
  - 二人で同じようなゲームを作るよりも，色々なゲームを作ってみることでアイデアや技術を蓄積したほうが良い
- 小規模なゲームの作成
  - 映像がなくても遊べるゲームのアイデアを10個程度挙げた
  - その中からメンバーが1つずつ1週間で作成した
  - 仕様書の作成・設計などは行わず，いきなり実装した
  - 実装にはJavaを使用した

今日までに行ったこと

- 南雲さんとのミーティングの予定調整
  - 現状報告も兼ねて，ミーティングを行う
  - 第一候補は30日 (月)
- Wikiの整備
  - 資料の共有場所を整備
  - 作業時間の報告場所を整備

来週までに行うこと

- 中間発表用のゲーム作成
  - 作った小規模なゲームの面白さを比較検討する
  - ゲームの規模は最低限遊べる程度の小さなものにする
  - 設計は行わず，共通部分についてはお互いが重複しないように仕様書を作成する (企画書を詰めていく)
- 問題点
  - 音素材の収集方法について
  - 音の聞こえ方の調整
  - サラウンド環境の準備

視覚にハンディキャップを有する人達との関わり方について

- α版レビュー期間中に，慶應大学の中根様にユーザーインタビューを依頼する予定である
- 横浜市立盲学校とも連絡を取って，クライアント・ユーザー評価に協力してもらいたい

さうんど おんりぃ2　進捗報告

2006/11/02

㈱インテム　菊池徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

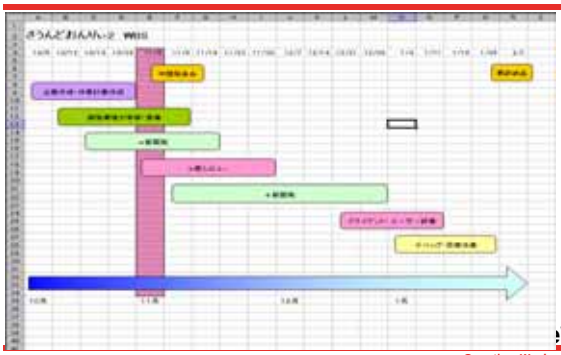
CreW
Creative Workspace

本日の報告内容

- 全体スケジュール
- 今週行ったこと
  - ゲームの評価対象，評価方法について
  - α版，β版の定義
  - 南雲さんを交えたミーティング
  - プロジェクト定義書の作成
  - 実装について
  - 前回作ったゲームの評価
- 来週までに行うこと
  - 南雲さんとミーティング
  - C++で実装

CreW
Creative Workspace

全体スケジュールについて



CreW
Creative Workspace

ゲームの評価対象，評価基準について

- 評価対象
  - 大岩研関係者，慶應大学の中根様，横浜市立盲学校の方々
- 評価方法
  - ユーザアンケート
    - 前期プロジェクトで作成した選択式のアンケートを改良
  - 面白いと感じた人が7割以上，もう一度やりたいと思う人が8割以上を目指す
    - 前回は，面白いと感じた人が5割，もう一度やりたいと感じた人が7割だった
  - 障害者と健常者の両方が面白いというゲーム

CreW
Creative Workspace

α版，β版の定義

- α版
  - ゲームの方向性を確認することが目的
  - 完成度については，20～30%程度
  - バグが潜んでいる可能性はあるものの，ゲームのイメージがわかる程度に動作しているもの
- β版
  - ゲーム内容の評価をすることが目的
  - 正式版の機能を一通り備えた完成品に近いもの
  - 完成度については，80～90%程度
  - ゲームの根幹に関わるようなバグは解消されていて，プレイできるもの

CreW
Creative Workspace

南雲さんを交えたミーティング

- 今週南雲さんとミーティングを行う予定だったが，延期になった
  - 南雲さんが屋久島に出張中で，連絡が付かなかったため
- インテムの金澤さんも交えて行う
- 11/6 (月) の19:00からユードー横浜本社にて行うということで，アポイントを取得済み

CreW
Creative Workspace

プロジェクト定義書の作成

- 暫定版の作成
  - 作成途中なので，項目の追加等を行っていく

CreW
Creative Workspace

実装について

- 前回まではJavaで簡単なゲームを二人で一つずつ作成した
- 前回作ったゲームをもとに，C++で実装を進める予定
  - 今週は，藤原･橋山共に勉強会の準備等に時間を取られ，ほとんど進められず

CreW
Creative Workspace

前回作ったゲームの評価

- 藤原のゲーム→聖徳太子ゲーム
  - 同時に三つの果物の名前を読み上げ，それを全て当てるゲーム
    - 内容としては面白いが，入力が面倒
    - 音声が合成なので，聞き取りづらい
    - サラウンドで音声を聞こえるようにしたい
- 橋山のゲーム→音記憶ゲーム
  - 上下左右から音が聞こえ，どこから聞こえたかを記憶していくゲーム
    - サラウンドを使っていないので，方向がよくわからない
    - サラウンドを活かして，音が移動していくようなものを作る

CreW
Creative Workspace

来週までに行うこと

- 南雲さんとミーティング
  - 11/6を予定
- C++によるゲームの実装

CreW
Creative Workspace

さうんど おんりぃ2　中間報告
2006/11/09

㈱インテム　菊地徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

CreW
Creative Workspace

発表の流れ

- 「映像のないゲーム」概要
- 「さうんど おんりぃ2」プロジェクトについて
- スケジュール
- デモンストレーション
- α版レビュー
- 今後の予定

CreW
Creative Workspace

「映像のないゲーム」概要

CreW
Creative Workspace

映像のないゲームとは？

- サラウンドスピーカーから流れてくる音の大きさや定位を聞くことでプレイできる
- 健常者にも視覚にハンディキャップを有する方々にも楽しんでもらうことができる

CreW
Creative Workspace

「映像のないゲーム」イメージ

CreW
Creative Workspace

さうんど おんりぃ2プロジェクトについて

CreW
Creative Workspace

今回のプロジェクトについて

- 先学期，株式会社ユードーの南雲氏より，「映像のないゲーム」を作って欲しいという依頼を受けた
- 「さうんど おんりぃ」というプロジェクトを立ち上げ，先学期で「映像のないゲーム」のプロトタイプを製作した
- 今期は，先学期のプロジェクトを引き継いで「映像のないゲーム」の開発を行う

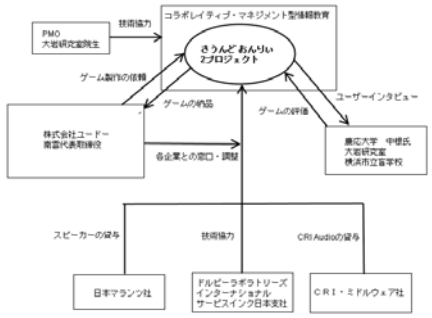
CreW
Creative Workspace

プロジェクトの目的

- ユーザーが繰り返し遊んでくれるような面白いと感じられるゲームを開発する
- 「映像のないゲーム」の開発を通じて，プロジェクト進行・ゲーム製作を勉強する

CreW
Creative Workspace

プロジェクトの体制



CreW
Creative Workspace

開発手法

- 機能仕様書や設計書を作らずに進める
  - 企画書のみを作成する
  - 設計は行うが，設計書という形では残さない
- メンバー各自がゲームをそれぞれ作る
- それぞれの作品の感触を比較しながら企画を詰めていく
- 最終的に，それぞれのゲームの面白い部分を組み合わせて1つのゲームを作る

CreW
Creative Workspace

ゲームの評価対象，評価基準について

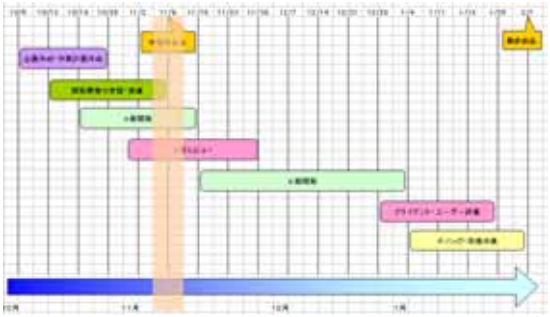
- 評価対象
  - 大岩研関係者，慶應大学の中根様，横浜市立盲学校の方々
- 評価方法
  - ユーザーアンケート
    - 前期プロジェクトで作成した選択式のアンケートを改良
    - 面白いと感じた人が7割以上，もう一度やりたいと思う人が8割以上を目指す
  - 健常者と視覚にハンディキャップを有する方々の両方が面白いというゲーム

CreW
Creative Workspace

スケジュール

CreW
Creative Workspace

全体スケジュールについて



CreW
Creative Workspace

現在の進捗状況

- 開発環境の学習・準備 (90%)
  - 追加のサラウンドヘッドフォンを南雲さんに発注した
- α版開発 (70%)
  - C++の学習が不足していたので，メンバーがそれぞれ得意とするJavaでゲームを作成した
  - ペアプログラミングにより，C++で1本のゲームを作ることで，学習の遅れを取り戻した
  - その後，メンバー2人でそれぞれC++で1本ずつ簡単なゲームを作成した

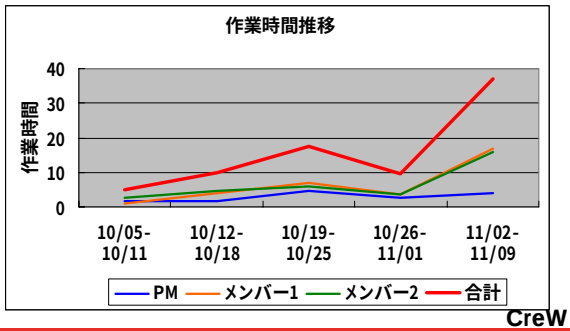
CreW
Creative Workspace

現在の進捗状況

- α版レビュー (30%)
  - Javaゲームのチーム内レビューを実施
  - C++ゲームのクライアントレビューを実施
  - C++ゲームのユーザーレビューの予定を調整中

CreW
Creative Workspace

作業時間推移



CreW
Creative Workspace

デモンストレーション

CreW
Creative Workspace

α版レビュー結果

CreW
Creative Workspace

チーム内レビューの結果

- Javaで作ったゲーム
  - メンバーが作った2つのゲームは両方とも音が一定の方向から聞こえていた
  - 音が動いている方が，場所を掴みやすい
  - 音がサラウンドで聞こえれば面白い

CreW
Creative Workspace

クライアント（南雲氏）の反応

- クライアントの承認はもらえた
- ゲーム自体は面白いが，既存のゲームの殻を破るようなアイデアがあるとより良くなる
  - 商店街のサウンドスケッチなど
- 音の演出次第でもっと面白くなる
  - 電子的な音は不自然なことが多い
  - 自然の音を取り入れる
  - 日常聞こえてくる音を聞き，感性を磨く
- モノラルの音を使うこと
  - ステレオだと音が歪む

CreW
Creative Workspace

今後の予定

CreW
Creative Workspace

今後の予定

- 慶応大学中根様に，ユーザーレビューを依頼する
- C++で作ったゲームのチーム内レビューを行う
- α版レビューを元に，企画書をまとめる
- その後，β版の開発を開始する

CreW
Creative Workspace

さうんど おんりぃ2　進捗報告
2006/11/16

㈱インテム　菊地徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

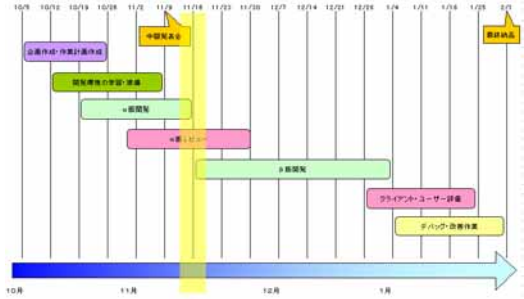
CreW
Creative Workspace

本日の報告内容

- 全体スケジュール
- 今週行ったこと
  - 中間報告会
  - ゲーム企画の作成
- 来週までに行うこと
  - 慶応大学中根様に，ユーザーレビューを依頼
  - β版開発に向けての企画の作成

CreW
Creative Workspace

全体スケジュールについて



CreW
Creative Workspace

中間報告会

- プロジェクトについて質問
  - 前期のプロジェクトとよりも優れている点，進歩した点はなにか？
  - Pmは何をマネージメントをするのか？
- 質問についての検討
  - 本日のミーティングで行う。

CreW
Creative Workspace

企画の作成

- B版に向けての企画案を考えた
  - 射的･蚊を叩く･ピンポン･虫取り
  - 人間の体内を探検する･宇宙空間で敵と戦う
- 企画案1
  - 虫取りゲーム
    - 音で空間を把握し、部屋の中や森の中で虫を捕まえる。
    - 季節なども用意し、夏にセミや蚊、秋には蛍やトンボを捕まえる。

CreW
Creative Workspace

来週までに行うこと

- 慶応大学中根様に，ユーザーレビューを依頼
  - 現在スケジュールを調整中
- β版開発に向けての企画の作成

CreW
Creative Workspace

さうんど おんりぃ2　進捗報告
2006/11/30

㈱インテム　菊地徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

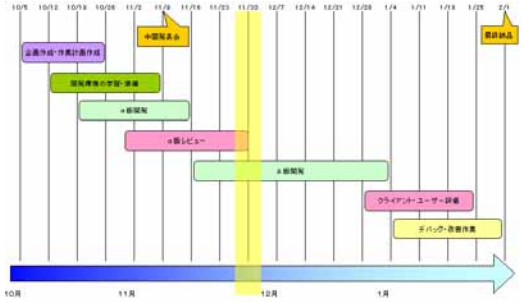
CreW
Creative Workspace

本日の報告内容

- 全体スケジュール
- 今週行ったこと
  - 開発環境の調査
  - ゲーム企画の作成
- 来週までに行うこと
  - 慶応大学中根様に，ユーザーレビューを依頼
  - α版企画を元に大まかな開発を行う
  - α版レビュー

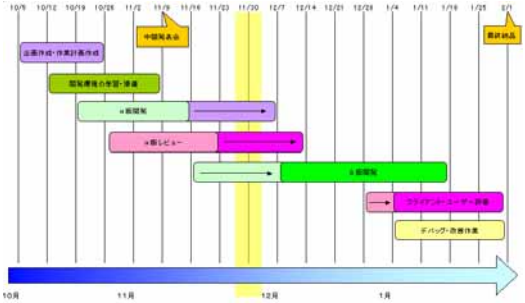
CreW
Creative Workspace

全体スケジュールについて (変更前)



CreW
Creative Workspace

全体スケジュールについて (変更後)



CreW
Creative Workspace

環境の調査、収集

- 効果音の収集
  - 効果音CDを入手した
  - 現在ゲームに必要だと思われる音で、手に入っていない音をユードー南雲氏に提供してもらった
- ゲーム開発のベースになる環境の作成
  - 音を鳴らし、音を動かすことが可能なベースを準備

CreW
Creative Workspace

企画の作成 (虫取りゲーム)

- 森を散歩しながら、制限時間内にできるだけ多くの虫を捕まえる
- ゲームであると同時に、自然の音に囲まれることでリラックスすることもできる

CreW
Creative Workspace

現状の問題点

- 音のループが不自然に聞こえる
- プレーヤーの向きが変わるときの定位の変更がうまくいかない→解決

CreW
Creative Workspace

来週までに行うこと

- 慶応大学中根様に、ユーザーレビューを依頼
  - 現在、スケジュール調整中
- α版の企画を元にゲームの基本部分を作成し、α版レビューを行う

CreW
Creative Workspace

さうんど おんりぃ2　進捗報告
2006/12/07

㈱インテム　菊地徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

CreW
Creative Workspace

本日の報告内容

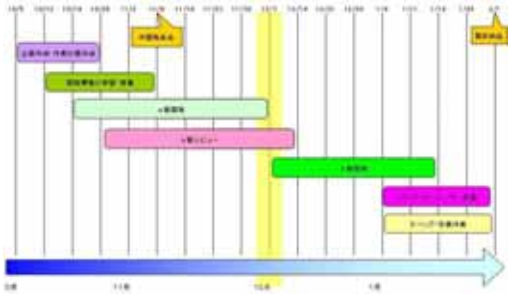
- 全体スケジュールについて
- β版移行について
  - α版の制作を通してわかったこと
  - β版移行のタイミングについて
- 今週行ったこと
  - 全体の作業見積もりと実績について
  - 実装について
- 来週までに行うこと
  - バグフィックスを行う
  - 慶応大学中根様に，ユーザーレビューを実施する
  - β版の開発を開始する

CreW
Creative Workspace

全体スケジュールについて

CreW
Creative Workspace

全体スケジュールについて



CreW
Creative Workspace

β版移行について

CreW
Creative Workspace

α版作成を通してわかったこと

- α版作成の失敗点について
  - 当初はα版ではアイデアをたくさん出し，洗練させていくという予定
  - 音に対するこだわりが無かった
    - 電子音声では迫力や臨場感に欠ける
  - 企画に力を入れなかった
    - メンバーの企画力が足りず，面白い企画が浮かばなかった
    - あまり作業時間を取らなかったので，たくさんのゲームを作れなかった
  - 結果として，面白そうなゲームの土台ができなかった

CreW
Creative Workspace

α版作成を通してわかったこと

- α版作成の失敗から学んだこと
  - 音が充実していることが重要
  - 企画に対するモチベーションが足りなかった
  - 企画はできる人に任せる（他の人の意見を取り入れる）
  - 作るものを明確にさせることが重要
    - 世界観，コンセプトなど
  - ゲーム作りのイメージはつかめた

CreW
Creative Workspace

β版移行のタイミングについて

- 大まかな企画はまとまった
- 12/8に中根様にユーザレビューを行う
- その結果を受けて，今後はβ版としてゲームを開発する

CreW
Creative Workspace

今週行ったこと

CreW
Creative Workspace

全体の作業見積もりと実績について

- Wikiを参照

CreW
Creative Workspace

実装について

- バグの修正
  - プレイヤーが移動しても音の聞こえてくる位置や角度がおかしくなる（解決）
  - 新規に作成した音ファイル（csbファイル）を再生できない（解決）
- 虫の実装
  - 虫を捕まえる

CreW
Creative Workspace

来週までに行うこと

CreW
Creative Workspace

来週までに行うこと

- 慶応大学中根様に，ユーザーレビューを実施
  - 12月8日　16時～　アポイント取得済み
- 音データの修正を南雲さんに依頼
  - ループ再生してしまうと，音が不自然にとぎれてしまう
- β版の開発を開始

CreW
Creative Workspace

さうんど おんりぃ2　進捗報告
2006/12/14

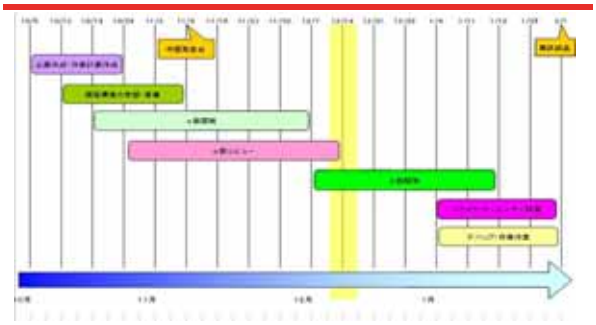
㈱インテム　菊地徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

本日の報告内容

- 全体スケジュールについて
- 今週行ったこと
  - 全体の作業見積もりと実績について
  - 企画書の修正，仕様書の作成
  - ユーザーインタビューについて
  - 技術的な問題の解決
- 来週までに行うこと

全体スケジュールについて

全体スケジュールについて



今週行ったこと

全体の作業見積もりと実績について

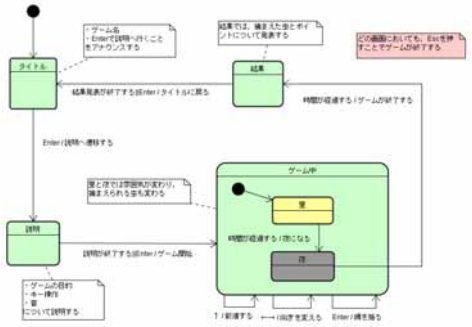
- Wiki参照

企画書（兼仕様書）の修正

- 制限時間の概念を取り入れた
  - 昼と夜によって制限時間を表現する
  - 昼と夜で捕まえられる虫が違う，など
- 画面構成，シーン構成，必要なリソース（音）についてまとめた
- 状態遷移図を作成して，ゲームの全体の流れを共有した

CreW
Creative Workspace

状態遷移図



CreW
Creative Workspace

ユーザーインタビューについて

- 日時：12月8日（金）　16：00より
- 協力：慶応大学村井研究室中根様
- 概要：映像のないゲームα版を実際に遊んで頂き，感想や改善点，問題点などを議論する

CreW
Creative Workspace

インタビュー結果分析①

- マップ全体がどのくらいの大きさか分からない
  - 端に来たかどうか分からない
  - 障害物があると目印になる
- 歩いていて，何にもぶつからないことが不自然
  - 木にぶつかる
  - 草を掻き分ける
- 音の聞こえ方が不自然
  - 足音がない
    - 歩いたことに対するフィードバックがない
  - 川に関する問題
    - 川の音が1箇所から聞こえる（泉のようになっている）
    - 川に入れない（川に入ると，川の音が後ろになってしまう）

CreW
Creative Workspace

インタビュー結果分析②

- オブジェクトの動き
  - 大量のオブジェクトが同時に動かなければ，問題ない
  - 動かなくても，そういうものだと理解すれば良い
- 慣れると飽きてしまう
  - イベントが必要
    - 虫捕り網は泉の傍に落ちているなど
  - ある段階になったら別の要素をとりいれる
- 説明音声が足りない
  - 最低限ゲームについての説明が必要
  - 人の音声を録音する
    - ゲームによっては，感情が入っていたほうがいい場合もある
    - 感情云々で大きな問題がでることはなさそう

CreW
Creative Workspace

インタビューの結果を受けて①

- 実装に優先順位をつけた
  - 優先順位1
    - 足音の実装（森，川）
    - 川に関する問題の解決
  - 優先順位2
    - オブジェクトを動かす
    - 音の聞こえ方を修正する
    - 自分のいる位置を把握するギミックを考える
      - 障害物の用意など
  - 優先順位3
    - 音声の実装
    - イベントの追加
    - 制限時間の追加

CreW
Creative Workspace

インタビューの結果を受けて②

- 実装の順番を変更した
  - 優先順位1の項目を実装
    - 足音の音素材を探した
    - 足音を実装（未実装）
  - ペンディング（保留）した項目
    - オブジェクトを動かす
    - 音に高低差をつける
    - 音の聞こえ方の修正

CreW
Creative Workspace

技術的な問題の解決

- 音の加工について
  - ループすると音が不自然に聞こえる（解決）
    →南雲さんに解決方法を教えてもらう
- CRI・Audioの問題について
  - 実行速度が遅くなる（未解決）
    →どの関数が原因なのかを調べる
  - 音ファイルの読み込みのサイズ（解決）
    →バッファサイズを大きくすれば良い
  - 複数の音を同時に再生できない（解決）
    →リソースの確保が正しく行えていない

CreW
Creative Workspace

来週までに行うこと

CreW
Creative Workspace

来週までに行うこと

- 優先順位1の実装の残り
  - 足音の実装など
- 優先順位2の実装
  - オブジェクト（虫，動物）の移動
  - 音の聞こえ方を修正する
  - 障害物などのギミックを考える（企画）
- 解決した技術的な問題の修正
  - 音ファイルや，実行速度など
- 横浜市立盲学校とのアポイント

CreW
Creative Workspace

さうんど おんりぃ2　進捗報告
2006/1/11

㈱インテム　菊地徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

CreW
Creative Workspace

本日の報告内容

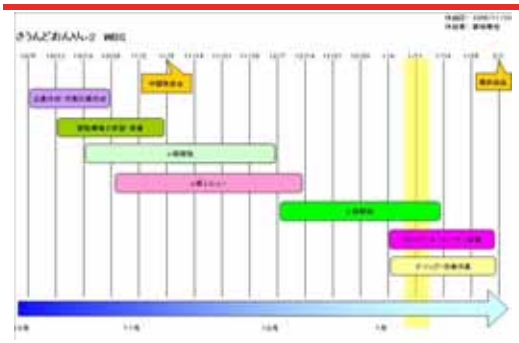
- 全体スケジュールについて
- 今週行ったこと
- デモンストレーション
- 来週までに行うこと

CreW
Creative Workspace

全体スケジュールについて

CreW
Creative Workspace

全体スケジュールについて



CreW
Creative Workspace

今週行ったこと

CreW
Creative Workspace

全体の作業見積もりと実績について

- 見積もりは失念していました
- 橋山 (35h)
  - 時間帯の実装
  - 動物の移動
  - 音声の録音
  - メモリリーク問題の修正
  - 最終発表会場の下見
- 藤原 (20h)
  - 向きを変えることによる定位の変更
  - 昆虫の移動
  - 状態管理の修正
  - 虫かごの実装

CreW
Creative Workspace

時間帯の概念の導入

- 昼と夜という概念を導入した
  - ゲームに進行
    - 昼から始まって，一定時間後に夜になる
    - 夜になって一定時間後にゲームが終了する
  - 昼と夜の違い
    - 昼は動物が動き回っていて，BGMも明るめ
    - 夜は動物は寝静まり，BGMも静か
    - 時間帯ごとに活動する生物のリストは外部ファイルから読み込む

CreW
Creative Workspace

状態管理の修正

```
int titleMessage = titleState->run();
switch(titleMessage) {
    case START_GAME: {
        int gameMessage = gamePlayState->run();
        if (gameMessage == QUIT_GAME) {
            return;
        } else if (gameMessage == GO_RESULT) {
            int resultMessage = resultState->run();
            if (resultMessage == QUIT_GAME) {
                return;
            }
            break;
        }
        break;
    }
}
```

CreW
Creative Workspace

状態管理の修正

- 今までのソースには拡張性がない
  - 状態を増やすたびに，Game.cppを修正する必要があり，if文のネストが増える
- 今までは状態が1つしかなかったが，タイトルや結果など状態が増えそうであった
  - Stateパターンを適用した
  - Game.cppを修正する必要がなくなったため，状態の追加が容易となった

CreW
Creative Workspace

横浜市立盲学校との連絡

- ユーザーレビューの依頼をした
- 以下のような返答があった（抜粋）
  - 完成前に意見を言いたい
  - 盲学校の全盲の在籍は1・2割である
  - 映像がなくてもセンスのよいもの（イメージとしてかっこいいもの）が好ましい
  - 画像はあっても構わないので，弱視の子にも見やすいもの

CreW
Creative Workspace

デモンストレーション

CreW
Creative Workspace

来週までに行うこと

CreW
Creative Workspace

来週までに行うこと

- ゲームギミックの追加
  - 虫を捕まえたことによるポイントを結果で知らせる
  - 虫捕りに加えて，アミを探すという要素の追加
- 音声の追加
  - タイトルなど，説明音声を録音･追加する
- 既存のバグの修正
  - 川の音が自然に聞こえるように修正する
- ユーザーレビューの詳細を詰める
- 最終報告書目次の作成

CreW
Creative Workspace

2007/01/31
さうんど おんりぃ
最終報告書

週報

作成日時　2006/10/13

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/10/5　～　2006/10/11

■ 作業報告

No.	内容	作業時間(h)	備考
1	プレゼン内容作成	5.0	
2	第2回授業(週間報告会)	1.5	
3	プロジェクトミーティング	1.5	
4	PMミーティング	1.5	
5	プロジェクト定義書作成	4.5	
6	WBS作成	3.5	
7	前期プロジェクトの資料確認	3.0	
11	メール確認、メール作成	5.5	
	合計	26.0	

■ 次週作業予定

No.	内容	備考
1	企画の作成、決定	
3	C++開発についての学習	

■ 反省･課題

No.	反省･課題	対策
1	コミュニケーションの方法 メールが使えない状態になってしまい連絡を取ることが出来なかった。	メールが使用できない場合の対処も考えておけば、このようなことになっても十分対応が可能だった。

■ その他

備考

作成日時　2006/10/18

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/10/12　～　2006/10/18

■ 作業報告

No.	内容	作業時間(h)	備考
1	PM勉強会	2.5	
2	第2回授業(週間報告会)	1.5	
3	プロジェクトミーティング	1.5	
4	PMミーティング	1.5	
5	企画作成		
6	メンバーミーティング	0.5	
7	企画確認	1.0	
8	前回の「映像のないゲーム」のソースコード確認	1.0	
9	製作環境準備	0.5	
10	週間報告書作成	1.0	
11	メール確認、メール作成	1.0	
	合計	12.0	

■ 次週作業予定

No.	内容	備考
1	企画の作成、決定	
2	開発の開始	
3	C++開発についての学習	

■ 反省･課題

No.	反省･課題	対策
1	企画の内容に差がある。 (ゲームのイメージの差) 橋山さんは作りたいものがはっきりしている一方で、藤原さんはまだ何を作りたいのかがイメージできていない。	開発に対するやる気の差にもなっていくので、なんとかその差を埋めたい。 一緒に話をする中で、アイデアを出して形にしていければと思うので、これからのコミュニケーションを円滑にするようにする。

■ その他

備考

作成日時　2006/10/25

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/10/19　～　2006/10/25

■ 作業報告

No.	内容	作業時間（h）			備考
		菊地	橋山	藤原	
1	PM勉強会	2.5	-	-	
2	第3回授業（週間報告会）	1.5	1.5	1.5	
3	プロジェクトミーティング	1.5	1.5	1.5	
4	PMミーティング	1.5	-	-	
5	メンバーミーティング	-	1.0	1.0	
6	企画作成、ゲーム開発	-	3.0	3.5	
7	週間報告書作成	0.5	-	-	
8	週間報告会資料作成	-	1.0	-	
9	資料修正（wbs、プロジェクト定義書）	1.0	-	-	
10	メール確認、メール作成	1.0	-	-	
11	wiki整備	0.5	0.5	-	
	合計	10.0	8.5	7.5	
		26.0			

■ 次週作業予定

No.	内容	備考
1	企画の作成	
2	ゲーム開発	
3	C++開発についての学習	
4	クライアントとの打ち合わせ	

■ 反省･課題

No.	反省･課題	対策
1		

■ その他

備考

作成日時　2006/11/2

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/10/26　～　2006/11/1

■ 作業報告

No.	内容	作業時間（h）			備考
		菊地	橋山	藤原	
1	PM勉強会	2.5	-	-	
2	第3回授業（週間報告会）	1.5	1.5	1.5	
3	プロジェクトミーティング	2.5	2.5	2.5	
4	PMミーティング	1.5	-	-	
5	メンバーミーティング	-	0.5	0.5	
6	企画作成、ゲーム開発	-	-	-	
7	週間報告書作成	0.5	-	-	
8	週間報告会資料作成	-	-	0.5	
9	資料修正（wbs、プロジェクト定義書）	1.0	-	-	
10	メール確認、メール作成	0.5	0.5	0.5	
11	C++学習	-	1.0	-	
	合計	10.0	5.0	5.5	
		20.5			

■ 次週作業予定

No.	内容	備考
1	企画の作成	
2	ゲーム開発	
3	C++開発についての学習	
4	プロジェクト定義書の作成	
5	中間報告会資料･打ち合わせ	
6	クライアントとの打ち合わせ	

■ 反省･課題

No.	反省･課題	対策
1	作業時間について 今週はメンバーが勉強会を開催するため、そちらの準備に時間を消費してしまい、ゲーム開発に時間を使うことができなかった。 また、私自身の作業で今週は手一杯になってしまい、PMの資料作成などに時間を作ることができなかった。	作業の遅れを踏まえ、作業スケジュールの見直しを行う。

■ その他

備考

作成日時　2006/11/8

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/11/2　～　2006/11/8

■ 作業報告

No.	内容	作業時間（h）			備考
		菊地	橋山	藤原	
1	PM勉強会	2.5	-	-	
2	第5回授業（週間報告会）	1.5	1.5	1.5	
3	プロジェクトミーティング	2.5	2.5	2.5	
4	PMミーティング	1.5	-	-	
5	企画作成、ゲーム開発	-	9.0	10.0	ペアプログラミング（各3.0h）
6	クライアントとの打ち合わせ	2.0	2.0	2.0	
7	中間報告会資料作成	1.5	0.5	0.5	
8	プロジェクトミーティング（中間報告会準備）	2.5	2.5	2.5	
9	週間報告書作成	0.5	-	-	
10	C++学習、CRI Audio学習	-	6.0	2.0	
11	勉強会準備	-	5.5	4.0	
12	メール確認、メール作成	0.5	0.5	0.5	
	合計	14.5	24.0	21.0	
		59.5			

■ 次週作業予定

No.	内容	備考
1	α版企画、開発	
2	α版レビュー	
3		
4		
5		
6		

■ 反省･課題

No.	反省･課題	対策
1		

■ その他

備考

作成日時　2006/11/16

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/11/9　～　2006/11/15

■ 作業報告

No.	内容	作業時間(h)			備考
		菊地	橋山	藤原	
1	中間報告会準備	0.5	0.5	0.5	
2	中間報告会	3.0	3.0	3.0	
3	プロジェクトミーティング	0.5	0.5	0.5	
4	メンバーミーティング	-	1.0	1.0	
5	週間報告書作成	0.5	-	-	
6	週間報告会資料作成	1.5	-	0.5	
7	メール確認、メール作成	0.5	1.0	-	
8	勉強会準備	-	11.0	14.5	
	合計	6.5	17.0	20.0	
		43.5			

■ 次週作業予定

No.	内容	備考
1	α版のユーザーレビュー	
2	β版企画作成	
3		
4		
5		
6		

■ 反省･課題

No.	反省･課題	対策
1	企画案が誰かに言われたら思いつくというような状態に感じられる。 自分でこれをというものをどんどん出してほしい	各個人と話をしてどう考えているのかを聞いてみる。 松澤さんにもいわれたが、一プログラマとしてやりたいのかゲームの企画を含めてやりたいのかなどきいてみて対処を考えたい。

■ その他

備考

作成日時　2006/11/16

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/11/16　～　2006/11/29

■ 作業報告

No.	内容	作業時間 (h)			備考
		菊地	橋山	藤原	
1	第6回授業 (週間報告会)	1.5	1.5	1.5	
2	プロジェクトミーティング	1.5	1.5	1.5	
3	PMミーティング	2.5	-	-	
4	PSP勉強会	3.0	3.0	3.0	
5	メンバーミーティング (11/20)	-	1.0	1.0	
6	メンバーミーティング (11/27)	-	1.0	1.0	
7	音収集	-	-	3.0	
8	音環境作成	-	4.0	8.0	
9	企画作成	-	2.0	-	
10	週間報告会資料作成	1.0	-	-	
11	メール確認、メール作成	0.5	1.0	1.0	
	合計	10.0	15.0	20.0	
		45.0			

■ 次週作業予定

No.	内容	備考
1	ユーザーレビューの実施	
2	α版企画の作成	
3	α版企画を元に実装	
4	α版レビュー	
5		
6		

■ 反省･課題

No.	反省･課題	対策
1	先週の作業メールが届かなく、作業内容を知ることができなかった	wikiに一言コメントを追加したのでメールを送ったり受信した際に書き込むようにしてもらい最悪受信できなくなったとしてもメーリングリストの倉庫で確認できるようにする。

■ その他

備考

作成日時　2006/12/7

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/11/30　～　2006/12/6

■ 作業報告

No.	内容	作業見積もり		作業時間（h）			備考
		橋山	藤原	菊地	橋山	藤原	
1	第7回授業（週間報告会）	-	-	1.5	1.5	1.5	
2	プロジェクトミーティング	-	-	1.0	1.0	1.0	
3	PMミーティング	-	-	1.5	-	-	
4	テスト勉強会	-	-	3.0	3.0	3.0	
5	メンバーミーティング	1.0	1.0	-	1.0	1.0	
6	ユーザーレビュー						
7	スケジュール調整	0.5	-	-	0.5	-	
8	ユーザーレビュー実施	1.0	1.0	-	-	-	
9	結果のレビュー	0.5	0.5	-	-	-	
10	実装						
11	音関連のバグの修正	2.0	2.0	-	0.5	2.0	
12	ゲームのギミック作り	2.0	5.0	-	2.0	12.0	
13	週間報告会資料作成	-	1.0	-	-	1.0	
14	PM資料作成	-	-	6.0	-	-	
	合計	7.0	10.5	13.0	9.5	21.5	
		17.5		44.0			

■ 次週作業予定

No.	内容	備考
1	ユーザーレビューの実施	
2	β版企画・仕様書作成	
3	β版実装	
4	技術調査（CRIミドルウェアについて）	
5		
6		

■ 反省・課題

No.	反省・課題	対策
1		

■ その他

備考

作成日時　2006/12/13

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/12/8　～　2006/12/13

■ 作業報告

No.	内容	作業見積もり		作業時間（h）			備考
		橋山	藤原	菊地	橋山	藤原	
1	第7回授業（週間報告会）	-	-	1.5	1.5	1.5	
2	プロジェクトミーティング	-	-	1.0	1.0	1.0	
3	PMミーティング	-	-	1.5	-	-	
4	メンバーミーティング	1.0	1.0	-	1.0	1.0	
5	企画関連						
6	企画書修正・仕様書作成	2.0	-	-	2.0	-	
7	企画のレビュー	-	0.5	-	-	-	
8	実装関連						
9	オブジェクトを動かす	-	3.0	-	-	-	ペンディング
10	高低差をつける	-	2.0	-	-	-	ペンディング
11	音の聞こえ方を調整する	3.0	3.0	-	-	-	ペンディング
12	バグ修正	3.0	3.0	-	-	2.0	
13	音ファイルの作成	1.0	-	-	1.0	-	
	ユーザーインタビューの結果を反映	-	-	-	-	4.0	
14	ユーザーインタビュー関連						
15	中根様ユーザーインタビュー実施	1.0	1.0	-	1.0	1.0	
16	中根様ユーザーインタビュー分析	0.5	0.5	-	-	-	
17	横浜市立盲学校との連絡	1.0	-	-	-	-	
18	技術調査関連						
19	CRI・ミドルウェアに質問する	1.0	-	-	1.0	-	
20	その他						
21	発表資料作成	1.0	-	-	1.0	-	
22	PM資料作成	-	-	0.1	-	-	
	合計	14.5	14.0	4.1	9.5	10.5	
		28.5		24.1			

■ 次週作業予定

No.	内容	備考
1	ユーザーレビューの実施	
2	β版企画・仕様書作成	
3	β版実装	
4	技術調査（CRIミドルウェアについて）	
5		
6		

■ 反省・課題

No.	反省・課題	対策
1	見積もりをした作業内容を実現することができなかった。	何か原因があたと思われるのでその部分の確認と把握をしておきたい。

■ その他

備考

作成日時　2006/12/20

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/12/14　～　2006/12/20

■ 作業報告

No.	内容	作業見積もり		作業時間（h）			備考
		橋山	藤原	菊地	橋山	藤原	
1	第7回授業（週間報告会）	-	-	1.5	1.5	1.5	
2	プロジェクトミーティング	-	-	1.0	1.0	1.0	
3	PMミーティング	-	-	1.5	-	-	
4	メンバーミーティング	1.0	1.0	-	-	-	
5	企画関連						
6	企画書修正・仕様書作成	1.0	-	-	-	-	
7	企画のレビュー	-	0.5	-	-	-	
8	実装関連						
9	サウンド準備	2.0	-	-	1.5	-	
10	オブジェクトを動かす	-	3.0	-	-	-	
11	音に高低差を付ける	-	2.0	-	-	-	
12	音の聞こえ方を調整する	-	3.0	-	-	-	
13	足音・川の実装	1.0	1.0	-	1.0	1.0	
14	設計方針について・バグの修正	4.0	4.0	-	-	4.0	
15	川に入ったときの足音の実装	-	-	-	2.0	-	
16	ユーザーインタビュー関連						
17	横浜市立盲学校との連絡	1.0	-	-	0.5	-	
18	技術調査関連						
20	サラウンド環境の準備	1.0	-	-	-	-	
21	その他						
22	C言語学習	-	-	-	2.0	-	
23	発表資料作成	-	-	-	-	-	
	合計	11.0	14.5	4.0	9.5	7.5	
		25.5		21.0			

■ 次週作業予定

No.	内容	備考
1	ユーザーレビューの実施	
2	β版企画・仕様書作成	
3	β版実装	
4	技術調査（サラウンド環境）	
5		
6		

■ 反省・課題

No.	反省・課題	対策
1		

■ その他

備考

作成日時　2006/12/20

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/12/21　～　2007/1/11

■ 作業報告

No.	内容	作業見積もり		作業時間（h）			備考
		橋山	藤原	菊地	橋山	藤原	
1	授業（週間報告会）	-	-	1.5	1.5	1.5	
2	プロジェクトミーティング	-	-	1.0	1.0	1.0	
3	PMミーティング	-	-	1.5	-	-	
5	橋山作業						
6	時間帯の実装	-	-				
8	動物の移動	-	-	-	35.0	-	
9	音声の録音	-	-				
10	メモリリーク問題の修正	-	-				
11	最終発表会場の下見	-	-				
12	藤原作業						
16	向きを変えることによる定位の変更	-	-				
17	昆虫の移動	-	-	-	-	20.0	
18	状態管理の修正	-	-				
20	虫かご実装	-	-				
21	その他						
	PM資料作成	-	-	8.0	-	-	
23	発表資料作成	-	-	-	1.0	-	
	合計	0.0	0.0	12.0	38.5	22.5	
		0.0		73.0			

■ 次週作業予定

No.	内容	備考
1	ユーザーレビューの実施	
2	実装･修正	
3	バグ修正	
4	最終報告資料作成	
5		
6		

■ 反省･課題

No.	反省･課題	対策
1		

■ その他

備考

作成日時　2006/12/20

週間報告書

プロジェクト名　：　さうんど おんりぃ2
報告者　：　菊地　徹也
作業期間　：　2006/12/21　～　2007/1/11

■ 作業報告

No.	内容	作業見積もり		作業時間（h）			備考
		橋山	藤原	菊地	橋山	藤原	
1	授業（週間報告会）	-	-	1.5	1.5	1.5	
2	プロジェクトミーティング	-	-	1.0	1.0	1.0	
3	PMミーティング	-	-	1.5	-	-	
5	橋山作業			-	-	-	
6	実装作業	-	-		63.0		
12	藤原作業			-	-	ー	
16	実装作業	-	-	-	-	-	
21	その他						
	ユーザーインタビュー						
	PM資料作成	-	-	-	-	-	
23	最終発表資料作成	-	-	4.0	15.0	7.0	
合計		0.0	0.0	8.0	80.5	9.5	
		0.0		98.0			

■ 次週作業予定

No.	内容	備考
1	ユーザーレビューの実施	
2	実装･修正	
3	バグ修正	
4	最終報告資料作成	
5		
6		

■ 反省･課題

No.	反省･課題	対策
1		

■ その他

備考

2007/01/31
さうんど おんりぃ
最終報告書

ソースコード：音記憶ゲーム（Java）

```
package memory;

import java.awt.Color;
import java.awt.Font;

/**
 * キャンバスを表現するクラス
 *
 * 各種書き込みメソッドにより，GUI描画を行なうことができます．
 * 実際の書き込み処理はCanvasPanelに委譲します．
 * （余計な処理をカプセル化していますので，中身を知りたい人はCanvasPanel(BWindow.java内)を参照せよ）
 *
 * @author macchan
 * @version 2.0
 */
public class BCanvas {

    private CanvasPanel canvasPanel;
    private CanvasKeyEventHandler keyHandler;
    private CanvasMouseEventHandler mouseHandler;

    /**
     * コンストラクタ
     */
    public BCanvas(CanvasPanel canvasPanel, CanvasKeyEventHandler keyHandler,
            CanvasMouseEventHandler mouseHandler) {
        this.canvasPanel = canvasPanel;
        this.keyHandler = keyHandler;
        this.mouseHandler = mouseHandler;
    }

    /**************************************************
     * 描画関連（第7,8回）
     **************************************************/

    /**
     * 線を引きます
     * 使用例:
     * 座標(10, 10) から 座標(20, 20)へ黒い線を引く場合
     * drawLine(Color.BLACK, 10, 10, 20, 20);
     */
    public void drawLine(Color color, double x1, double y1, double x2, double y2) {
        canvasPanel.drawLine(color, x1, y1, x2, y2);
    }

    /**
     * 塗りつぶした三角形を書きます
     * 使用例:
     * 座標A(10, 10), 座標B(20, 20), 座標C(30,30)を頂点とする三角形を書く場合
     * drawFillTriangle(Color.BLACK, 10, 10, 20, 20, 30, 30);
     */
    public void drawFillTriangle(Color color, double x1, double y1, double x2,
            double y2, double x3, double y3) {
        canvasPanel.drawFillTriangle(color, x1, y1, x2, y2, x3, y3);
    }

    /**
     * 円弧を書きます
     * 角度の単位は度です（0〜360度）
     * startAngleには弧を描き始める角度
     *         90
     * 180           0
     *       270
```



```
     * arcAngleには，弧全体の角度を書きます．弧は反時計回りに書かれます
     * 使用例:
     * 座標(10, 10)を左上として，高さ100, 幅100 の円弧を書く場合
     * drawDrawArc(Color.BLACK, 10, 10, 100, 100, 0, 360);
     */
    public void drawArc(Color color, double x, double y, double width,
            double height, double startAngle, double arcAngle) {
        canvasPanel.drawArc(color, x, y, width, height, startAngle, arcAngle);
    }

    /**
     * 塗りつぶした円を書きます
     * startAngleには弧を描き始める角度
     *         90
     * 180           0
     *       270
     * arcAngleには，弧全体の角度を書きます．弧は反時計回りに書かれます
     * 使用例:
     * 座標(10, 10)を左上として，高さ100, 幅100 左半分の円を書く場合
     * drawDrawArc(Color.BLACK, 10, 10, 100, 100, 90, 180);
     */
    public void drawFillArc(Color color, double x, double y, double width,
            double height, double startAngle, double arcAngle) {
        canvasPanel.drawFillArc(color, x, y, width, height, startAngle,
                arcAngle);
    }

    /**************************************************
     * 描画関連（第9回以降）
     **************************************************/

    /**
     * 文字を書きます
     */
    public void drawText(Color color, String text, double x, double y) {
        canvasPanel.drawText(color, text, x, y);
    }

    /**
     * （フォントサイズを指定して）文字を書きます
     */
    public void drawText(Color color, String text, double x, double y, Font font) {
        canvasPanel.drawText(color, text, x, y, font);
    }

    /**
     * 画像を書きます
     */
    public void drawImage(String filename, double x, double y) {
        canvasPanel.drawImage(filename, x, y);
    }

    /**
     * 画像を書きます
     * （幅と高さを引数にとり，その大きさに拡大，縮小します）
     */
    public void drawImage(String filename, double x, double y, double width,
            double height) {
        canvasPanel.drawImage(filename, x, y, width, height);
    }

    /**************************************************
     * フォント文字サイズの取得
```

```
     ****************************************************/

    /**
     * テキストの幅を取得します
     */
    public int getTextWidth(String text, Font font) {
        return canvasPanel.getTextWidth(text, font);
    }

    /**
     * テキストの高さを取得します
     */
    public int getTextHeight(String text, Font font) {
        return canvasPanel.getTextHeight(text, font);
    }

    /****************************************************
     * 画像サイズの取得
     ****************************************************/

    /**
     * 画像の幅を取得します
     */
    public int getImageWidth(String filename) {
        return canvasPanel.getImageWidth(filename);
    }

    /**
     * 画像の高さを取得します
     */
    public int getImageHeight(String filename) {
        return canvasPanel.getImageHeight(filename);
    }

    /****************************************************
     * 更新関連
     ****************************************************/

    /**
     * キャンバス全体を白く塗りつぶします
     */
    public void clear() {
        canvasPanel.clear();
    }

    /**
     * キャンバスを更新（再描画）します
     */
    public void update() {
        canvasPanel.update();
        keyHandler.update();
        mouseHandler.update();
    }

    /****************************************************
     * キー入力関連
     ****************************************************/

    /**
     * 押されたキーのコードを取得します
     */
    public int getKeyCode() {
        return keyHandler.key();
```



```
    }

    /**
     * 何らかのキーが押されたかどうかを調べます（継続は含まない）
     */
    public boolean isKeyDown() {
        return keyHandler.isKeyDown();
    }

    /**
     * 指定されたキーが押されている状態かどうかを調べます（継続も含む）
     */
    public boolean isKeyPressing(int keycode) {
        return keyHandler.isKeyPressing(keycode);
    }

    /****************************************************
     * マウス入力関連
     ****************************************************/

    /**
     * マウスのX座標を取得します
     */
    public int getMouseX() {
        return mouseHandler.mouseX();
    }

    /**
     * マウスのY座標を取得します
     */
    public int getMouseY() {
        return mouseHandler.mouseY();
    }

    /**
     * マウスが押されているかどうか調べます
     */
    public boolean isMouseDown() {
        return mouseHandler.isMouseDown();
    }

    /**
     * 右のマウスボタンが押されているかどうか調べます
     */
    public boolean isRightMouseDown() {
        return mouseHandler.isRightMouseDown();
    }

    /**
     * 左のマウスボタンが押されているかどうか調べます
     */
    public boolean isLeftMouseDown() {
        return mouseHandler.isLeftMouseDown();
    }

    /**
     * クリックかどうか調べます
     * (何回でのクリックも反応します)
     */
    public boolean isClick() {
        return mouseHandler.isClick();
    }
```

```
    /**
     * シングルクリックかどうか調べます
     */
    public boolean isSingleClick() {
        return mouseHandler.isSingleClick();
    }

    /**
     * ダブルクリックかどうか調べます
     */
    public boolean isDoubleClick() {
        return mouseHandler.isDoubleClick();
    }

    /**
     * ドラッグ中かどうか調べます
     */
    public boolean isDragging() {
        return mouseHandler.isDragging();
    }

    /*****************************************************
     * その他
     *****************************************************/

    /**
     * 指定された秒数待ちます
     */
    public void sleep(double seconds) {
        try {
            Thread.sleep((long) (seconds * 1000));
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /**
     * キャンパスの幅を取得します
     */
    public int getCanvasWidth() {
        return canvasPanel.getWidth();
    }

    /**
     * キャンパスの高さを取得します
     */
    public int getCanvasHeight() {
        return canvasPanel.getHeight();
    }
}
```

```
package memory;

import bsound.BSoundSystem;
import bsound.framework.BSoundPlayer;

/**
 * 初心者用　音出しクラス．　オブプロ履修者のために．
 * mp3, wav, midファイルを簡単に制御できます．
 *
 * 1．基本的な使い方
 * BSound sound = new BSound("sample.wav");
 * sound.loop();
 *
 * 2．いちいちインスタンスを生成しない簡易メソッドを使う場合
 * BSound.play("sample.wav");
 *
 * 1．はＢＧＭ，2．は効果音に最適です．
 * サンプルコードBSoundTestを参照ください．
 *
 * このクラスで音を再生した場合デフォルトでストリーミング再生を行いますが，
 * 反応速度が重要な場合は，メモリにロードしておく必要があります．
 * BSound.load("sample.wav");
 * 当然ながら，BGM等の長いファイルは，ロードするとメモリを圧迫します．気をつけてください．
 *
 * なお，現在のバージョンでは，midiファイルの音量調節はできません．
 *
 * @author macchan
 */
public class BSound {

    /*********************************************
     * クラスメソッド
     *********************************************/

    /**
     * 再生する（止められません）
     */
    public static final void play(String filename) {
        new BSound(filename).play();
    }

    /**
     * ボリュームを指定して再生する（止められません）
     */
    public static final void play(String filename, int volume) {
        BSound sound = new BSound(filename);
        sound.setVolume(volume);
        sound.play();
    }

    /**
     * メモリ上にサウンドデータを読み込みます(反応が早くなりますが，メモリ領域が必要です)
     */
    public static final void load(String filename) {
        BSoundSystem.load(filename);
    }

    /*********************************************
     * BSound本体
     *********************************************/

    private BSoundPlayer player = null;
```



```
    public BSound(String filename) {
        player = BSoundSystem.createPlayer(filename);
    }

    /*
     * ------------------------ 操作系 ------------------------
     */

    /**
     * 再生します
     */
    public void play() {
        player.setLoop(false);
        player.play();
    }

    /**
     * ループ再生します
     */
    public void loop() {
        player.setLoop(true);
        player.play();
    }

    /**
     * 停止します
     */
    public void stop() {
        player.stop();
    }

    /**
     * 再生中かどうか調べます
     */
    public boolean isPlaying() {
        return player.getState() == BSoundPlayer.PLAYING;
    }

    /* -------------------------
     * ボリュームコントロール系（ボリュームは0-100の100段階設定ができます）
     * ------------------------- */

    /**
     * 現在のボリュームを取得します．
     */
    public int getVolume() {
        return player.getVolume();
    }

    /**
     * ボリュームを設定します．
     */
    public void setVolume(int volume) {
        player.setVolume(volume);
    }

    /**
     * 初期ボリュームを取得します
     */
    public int getDefaultVolume(){
        return player.getDefaultVolume();
    }
```

```
127     }
```

```java
package memory;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import javax.imageio.ImageIO;
import javax.swing.JFrame;

/**
 * ウインドウを表現するクラス
 *
 * @author macchan
 * @version 2.0
 */
public class BWindow {

    private JFrame frame;
    private BCanvas canvas;

    /**
     * コンストラクタ
     */
    public BWindow() {
        //フレーム生成
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //パネル＆イベントハンドラ生成
        CanvasPanel canvasPanel = new CanvasPanel();
        frame.getContentPane().add(canvasPanel);

        CanvasKeyEventHandler keyHandler = new CanvasKeyEventHandler();
        CanvasMouseEventHandler mouseHandler = new CanvasMouseEventHandler();
        frame.addKeyListener(keyHandler);
        frame.getContentPane().addKeyListener(keyHandler);
        canvasPanel.addKeyListener(keyHandler);
        canvasPanel.addMouseListener(mouseHandler);
        canvasPanel.addMouseMotionListener(mouseHandler);

        //キャンバス生成
        canvas = new BCanvas(canvasPanel, keyHandler, mouseHandler);
    }
```



```java
    /**
     * 位置を設定する
     */
    public void setLocation(int x, int y) {
        frame.setLocation(x, y);
    }

    /**
     * 大きさを設定する
     */
    public void setSize(int width, int height) {
        frame.setSize(width, height);
    }

    /**
     * (この) ウインドウを表示する
     */
    public void show() {
        frame.setVisible(true);
    }

    /**
     * 書き込みができるCanvasインスタンスを取得する
     */
    public BCanvas getCanvas() {
        return canvas;
    }

}

/**
 * キーのイベントを拾うクラス
 */
class CanvasKeyEventHandler implements KeyListener {

    //定数
    public static final int NULL_KEY_CODE = -1;
    public static final int NULL_MOUSE_LOCATION = -1;

    //入力イベント関連
    private KeyEvent bufferKeyEvent = null;
    private KeyEvent capturedKeyEvent = null;

    private Set pressingKeys = new HashSet();

    /*****************************************************
     * リスナインターフェイスの実装
     *****************************************************/

    public void keyPressed(KeyEvent e) {
        bufferKeyEvent = e;
        pressingKeys.add(new Integer(e.getKeyCode()));
    }

    public void keyReleased(KeyEvent e) {
        pressingKeys.remove(new Integer(e.getKeyCode()));
    }

    public void keyTyped(KeyEvent e) {
    }

    /*****************************************************
     * 公開インターフェイス
```

```
127         ************************************************/
128
129         public int key() {
130             if (capturedKeyEvent != null) {
131                 return capturedKeyEvent.getKeyCode();
132             } else {
133                 return NULL_KEY_CODE;
134             }
135         }
136
137         public boolean isKeyDown() {
138             return capturedKeyEvent != null;
139         }
140
141         public boolean isKeyPressing(int keycode) {
142             return pressingKeys.contains(new Integer(keycode));
143         }
144
145         /************************************************
146          * 更新関連
147          ************************************************/
148
149         public void update() {
150             capturedKeyEvent = bufferKeyEvent;
151             bufferKeyEvent = null;
152         }
153
154     }
155
156     /**
157      * マウスのイベントを拾うクラス
158      */
159     class CanvasMouseEventHandler implements MouseListener, MouseMotionListener {
160
161         //定数
162         public static final int NULL_MOUSE_LOCATION = -1;
163
164         //入力イベント関連
165         private int mouseX = NULL_MOUSE_LOCATION;
166         private int mouseY = NULL_MOUSE_LOCATION;
167         private boolean isDragging = false;
168         private MouseEvent bufferMouseEvent = null;
169         private MouseEvent capturedMouseEvent = null;
170
171         private Set pressingMouses = new HashSet();
172
173         /************************************************
174          * リスナインターフェイスの実装
175          ************************************************/
176
177         public void mousePressed(MouseEvent e) {
178             bufferMouseEvent = e;
179             pressingMouses.add(new Integer(e.getButton()));
180         }
181
182         public void mouseReleased(MouseEvent e) {
183             bufferMouseEvent = e;
184             pressingMouses.remove(new Integer(e.getButton()));
185             isDragging = false;
186         }
187
188         public void mouseClicked(MouseEvent e) {
189             bufferMouseEvent = e;
```



```
190         }
191
192         public void mouseEntered(MouseEvent e) {
193             bufferMouseEvent = e;
194         }
195
196         public void mouseExited(MouseEvent e) {
197             bufferMouseEvent = e;
198             isDragging = false;
199         }
200
201         public void mouseMoved(MouseEvent e) {
202             bufferMouseEvent = e;
203         }
204
205         public void mouseDragged(MouseEvent e) {
206             bufferMouseEvent = e;
207             isDragging = true;
208         }
209
210         /************************************************
211          * 公開インターフェイス
212          ************************************************/
213
214         public int mouseX() {
215             return mouseX;
216         }
217
218         public int mouseY() {
219             return mouseY;
220         }
221
222         public boolean isMouseDown() {
223             return isRightMouseDown() || isLeftMouseDown();
224         }
225
226         public boolean isRightMouseDown() {
227             if (pressingMouses.contains(new Integer(MouseEvent.BUTTON3))) {
228                 return true;
229             } else {
230                 return capturedMouseEvent == null ? false : capturedMouseEvent
231                         .getButton() == MouseEvent.BUTTON3;
232             }
233         }
234
235         public boolean isLeftMouseDown() {
236             if (pressingMouses.contains(new Integer(MouseEvent.BUTTON1))) {
237                 return true;
238             } else {
239                 return capturedMouseEvent == null ? false : capturedMouseEvent
240                         .getButton() == MouseEvent.BUTTON1;
241             }
242         }
243
244         public boolean isClick() {
245             return capturedMouseEvent == null
246                     ? false
247                     : capturedMouseEvent.getID() == MouseEvent.MOUSE_CLICKED;
248         }
249
250         public boolean isSingleClick() {
251             return capturedMouseEvent == null
252                     ? false
```

```
253                     : capturedMouseEvent.getID() == MouseEvent.MOUSE_CLICKED
254                             && capturedMouseEvent.getClickCount() == 1;
255         }
256
257         public boolean isDoubleClick() {
258             return capturedMouseEvent == null
259                     ? false
260                     : capturedMouseEvent.getID() == MouseEvent.MOUSE_CLICKED
261                             && capturedMouseEvent.getClickCount() == 2;
262         }
263
264         public boolean isDragging() {
265             return isDragging;
266         }
267
268         /**************************************************
269          * 更新関連
270          **************************************************/
271
272         public void update() {
273             capturedMouseEvent = bufferMouseEvent;
274             if (capturedMouseEvent != null) {
275                 mouseX = capturedMouseEvent.getX();
276                 mouseY = capturedMouseEvent.getY();
277             }
278
279             bufferMouseEvent = null;
280         }
281
282     }
283
284     /**
285      * Canvasの委譲先クラス
286      * Canvasに書かれる形式をバッファし，Swing形式に変換し出力します．
287      */
288     class CanvasPanel extends Canvas implements ComponentListener {
289
290         //定数
291         private static final int FILP_BUFFERSTRATEGY = 3;
292         private static final Graphics2D NULL_GRAPHICS = (Graphics2D) (new BufferedImage(
293                 1, 1, BufferedImage.TYPE_3BYTE_BGR).createGraphics());
294
295         //属性
296         private Graphics2D offGraphics;
297
298         /**
299          * コンストラクタ
300          */
301         public CanvasPanel() {
302             addComponentListener(this);
303             refreshOffGraphics();
304         }
305
306         /**************************************************
307          * BufferStrategy関連
308          **************************************************/
309
310         private void initializeBufferStrategy() {
311             createBufferStrategy(FILP_BUFFERSTRATEGY);
312             refreshOffGraphics();
313         }
314
315         private void refreshOffGraphics() {
```



```
316             offGraphics = getGraphics2D();
317             offGraphics.setColor(Color.WHITE);
318             offGraphics.fillRect(0, 0, getWidth(), getHeight());
319         }
320
321         private Graphics2D getGraphics2D() {
322             Graphics2D g2d = (Graphics2D) getBufferStrategy().getDrawGraphics();
323             if (g2d != null) {
324                 return g2d;
325             } else {
326                 return NULL_GRAPHICS;
327             }
328         }
329
330         private void flip() {
331             getBufferStrategy().show();
332         }
333
334         /**************************************************
335          * Component Listener関連
336          **************************************************/
337
338         public void componentHidden(ComponentEvent e) {
339         }
340         public void componentMoved(ComponentEvent e) {
341         }
342         public void componentResized(ComponentEvent e) {
343             initializeBufferStrategy();
344         }
345         public void componentShown(ComponentEvent e) {
346         }
347
348         /**************************************************
349          * 描画関連
350          **************************************************/
351
352         public void drawLine(Color color, double x1, double y1, double x2, double y2) {
353             offGraphics.setColor(color);
354             offGraphics.drawLine((int) x1, (int) y1, (int) x2, (int) y2);
355         }
356
357         public void drawFillTriangle(Color color, double x1, double y1, double x2,
358                 double y2, double x3, double y3) {
359             offGraphics.setColor(color);
360             offGraphics.fillPolygon(new int[]{(int) x1, (int) x2, (int) x3},
361                     new int[]{(int) y1, (int) y2, (int) y3}, 3);
362         }
363
364         public void drawArc(Color color, double x, double y, double width,
365                 double height, double startAngle, double arcAngle) {
366             offGraphics.setColor(color);
367             offGraphics.drawArc((int) x, (int) y, (int) width, (int) height,
368                     (int) startAngle, (int) arcAngle);
369         }
370
371         public void drawFillArc(Color color, double x, double y, double width,
372                 double height, double startAngle, double arcAngle) {
373             offGraphics.setColor(color);
374             offGraphics.fillArc((int) x, (int) y, (int) width, (int) height,
375                     (int) startAngle, (int) arcAngle);
376         }
377
378         public void drawText(Color color, String text, double x, double y) {
```

```
379             offGraphics.setColor(color);
380             offGraphics.drawString(text, (int) x, (int) y);
381         }
382 
383         public void drawText(Color color, String text, double x, double y, Font font) {
384             FontMetrics fontMetrics = offGraphics.getFontMetrics(font);
385             int topY = (int) y + fontMetrics.getAscent();
386 
387             //前処理
388             offGraphics.setColor(color);
389             Font originalFont = offGraphics.getFont();
390             offGraphics.setFont(font);
391 
392             //処理
393             offGraphics.drawString(text, (int) x, topY);
394 
395             //後処理
396             offGraphics.setFont(originalFont);
397         }
398 
399         public void drawImage(String filename, double x, double y, double width,
400                 double height) {
401             BufferedImage image = BImageProvider.getInstance().getImage(filename);
402             double scaleX = width / image.getWidth();
403             double scaleY = height / image.getHeight();
404             AffineTransform transform = AffineTransform.getScaleInstance(scaleX,
405                     scaleY);
406             AffineTransformOp transformOp = new AffineTransformOp(transform,
407                     AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
408             drawImage(image, transformOp, x, y);
409         }
410 
411         public void drawImage(String filename, double x, double y) {
412             BufferedImage image = BImageProvider.getInstance().getImage(filename);
413             drawImage(image, null, x, y);
414         }
415 
416         private void drawImage(BufferedImage image, AffineTransformOp transformOp,
417                 double x, double y) {
418             offGraphics.drawImage(image, transformOp, (int) x, (int) y);
419         }
420 
421         /**************************************************
422          * フォント文字サイズの取得
423          **************************************************/
424 
425         public int getTextWidth(String text, Font font) {
426             FontMetrics fontMetrics = offGraphics.getFontMetrics(font);
427             return fontMetrics.stringWidth(text);
428         }
429 
430         public int getTextHeight(String text, Font font) {
431             FontMetrics fontMetrics = offGraphics.getFontMetrics(font);
432             return fontMetrics.getHeight();
433         }
434 
435         /**************************************************
436          * 画像サイズの取得
437          **************************************************/
438 
439         public int getImageWidth(String filename) {
440             BufferedImage image = BImageProvider.getInstance().getImage(filename);
441             return image.getWidth();
```



```
442         }
443 
444         public int getImageHeight(String filename) {
445             BufferedImage image = BImageProvider.getInstance().getImage(filename);
446             return image.getHeight();
447         }
448 
449         /**************************************************
450          * 更新関連
451          **************************************************/
452 
453         public void update() {
454             flip();
455         }
456 
457         public void clear() {
458             refreshOffGraphics();
459         }
460 
461     }
462 
463     /**
464      * 画像読み込みクラス
465      */
466     class BImageProvider {
467 
468         //定数
469         private static final int DUMMY_IMAGE_FONT_SIZE = 12;
470         private static final Font DUMMY_IMAGE_FONT = new Font("Dialog", Font.PLAIN,
471                 DUMMY_IMAGE_FONT_SIZE);
472 
473         /*********************************
474          * SingleTonの実装
475          *********************************/
476 
477         private static BImageProvider instance;
478 
479         public static BImageProvider getInstance() {
480             if (instance == null) {
481                 instance = new BImageProvider();
482             }
483             return instance;
484         }
485 
486         //属性
487         private Map images = new HashMap();
488 
489         /**
490          * コンストラクタ
491          */
492         private BImageProvider() {
493             super();
494         }
495 
496         /**
497          * 画像を取得する（なければ新しく生成）
498          */
499         public BufferedImage getImage(String filename) {
500             if (!images.containsKey(filename)) {
501                 images.put(filename, prepareImage(filename));
502             }
503             return (BufferedImage) images.get(filename);
504         }
```

```
505
506         /*********************************
507          * 以下, 画像読み込み処理
508          *********************************/
509
510         private BufferedImage prepareImage(String filename) {
511             try {
512                 return loadImage(filename);
513             } catch (Exception ex) {
514                 return createDummyImage(filename);
515             }
516         }
517
518         private BufferedImage loadImage(String filename) throws IOException {
519             File f = new File(filename);
520             BufferedImage image = ImageIO.read(f);
521             return image;
522         }
523
524         private BufferedImage createDummyImage(String filename) {
525             //画像を生成する
526             int width = DUMMY_IMAGE_FONT_SIZE * filename.length();
527             int height = DUMMY_IMAGE_FONT_SIZE * 2;
528             BufferedImage image = new BufferedImage(width, height,
529                     BufferedImage.TYPE_4BYTE_ABGR);
530
531             //ダミー画像を書き込む
532             Graphics g = image.getGraphics();
533             g.setColor(Color.WHITE);
534             g.fillRect(0, 0, width - 1, height - 1);
535             g.setColor(Color.BLACK);
536             g.setFont(DUMMY_IMAGE_FONT);
537             g.drawRect(0, 0, width - 1, height - 1);
538             g.drawString(filename, 10, height / 2);
539             g.dispose();
540
541             return image;
542         }
543     }
```

```
1   package memory;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   public class MemoryGame {
7       private final String RESOURCE_PATH = "res/memory/";
8
9       private final int KEY_UP = 38;
10      private final int KEY_DOWN = 40;
11      private final int KEY_LEFT = 37;
12      private final int KEY_RIGHT = 39;
13
14      public static void main(String args[]) {
15          MemoryGame game = new MemoryGame();
16          game.run();
17      }
18
19      public void run() {
20          showTitle();
21          doMemoryGame();
22          showEndTitle();
23      }
24
25      /**
26       * ゲーム開始を知らせる
27       */
28      private void showTitle() {
29          System.out.println("音記憶ゲームを開始します．");
30      }
31
32      /**
33       * ゲーム中の処理を行う
34       */
35      private void doMemoryGame() {
36          // ウインドウを初期化する
37          BWindow window = openWindow();
38          BCanvas canvas = window.getCanvas();
39
40          loadSounds();
41          List soundList = new ArrayList(); // 鳴った音を保存するリスト
42
43          while (true) { // 正解している限り，繰り返す
44              Sound currentSound = createSound();
45              soundList.add(currentSound); // 音を記憶する
46              playSound(canvas, soundList);
47
48              List keyList = new ArrayList(); // 入力したキーを保存するリスト
49              keyList = getKeyCode(canvas, keyList, soundList.size());
50
51              if (judge(soundList, keyList)) {
52                  BSound.play(RESOURCE_PATH + "right.wav", 70);
53                  System.out.println(soundList.size() + "問正解！");
54                  canvas.sleep(0.5);
55              } else {
56                  BSound.play(RESOURCE_PATH + "wrong.wav", 70);
57                  System.out.println("残念でした．不正解です．");
58                  System.out.println("正解数は" + (soundList.size() - 1) + "問でした．");
59                  canvas.sleep(0.5);
60                  break;
61              }
62          }
63      }
```



```
64
65      /**
66       * 効果音をロードする
67       */
68      private void loadSounds() {
69          BSound.load(RESOURCE_PATH + "up.mid");
70          BSound.load(RESOURCE_PATH + "down.mid");
71          BSound.load(RESOURCE_PATH + "left.mid");
72          BSound.load(RESOURCE_PATH + "right.mid");
73          BSound.load(RESOURCE_PATH + "right.wav");
74          BSound.load(RESOURCE_PATH + "wrong.wav");
75      }
76
77      /**
78       * ウィンドウを開く
79       */
80      private BWindow openWindow() {
81          BWindow window = new BWindow();
82          window.setLocation(150, 150);
83          window.setSize(300, 300);
84          window.show();
85
86          return window;
87      }
88
89      /**
90       * ランダムな音を生成する
91       */
92      private Sound createSound() {
93          return new Sound();
94      }
95
96      /**
97       * 音を鳴らす
98       */
99      private void playSound(BCanvas canvas, List soundList) {
100         for (int i = 0; i < soundList.size(); i++) { // 記憶されている音を順番に鳴らす
101             String noteDirection = ((Sound) soundList.get(i)).getDirection();
102
103             if (noteDirection == "↑") { // 音の方向に合わせて，再生するファイルを変える
104                 BSound.play(RESOURCE_PATH + "up.mid");
105             } else if (noteDirection == "↓") {
106                 BSound.play(RESOURCE_PATH + "down.mid");
107             } else if (noteDirection == "←") {
108                 BSound.play(RESOURCE_PATH + "left.mid");
109             } else if (noteDirection == "→") {
110                 BSound.play(RESOURCE_PATH + "right.mid");
111             }
112             canvas.sleep(0.3); // 音同士の間隔を空ける
113         }
114     }
115
116     /**
117      * 押されたキーのリストを取得する
118      */
119     private List getKeyCode(BCanvas canvas, List keyList, int soundListSize) {
120         while (keyList.size() < soundListSize) { // キーの入力が終わるまで繰り返す
121             canvas.update();
122
123             if (canvas.isKeyDown()) {
124                 if (canvas.getKeyCode() == KEY_UP) {
125                     keyList.add("↑");
126                 } else if (canvas.getKeyCode() == KEY_DOWN) {
```

```
127                     keyList.add("↓");
128                 } else if (canvas.getKeyCode() == KEY_LEFT) {
129                     keyList.add("←");
130                 } else if (canvas.getKeyCode() == KEY_RIGHT) {
131                     keyList.add("→");
132                 }
133                 System.out.print(keyList.get(keyList.size() - 1).toString()); // 入力したキーを表示する
134             }
135             canvas.sleep(0.1);
136         }
137         System.out.println("");
138         return keyList;
139     }
140 
141     /**
142      * 正誤を判定する
143      */
144     private boolean judge(List soundList, List keyList) {
145         boolean result = true;
146 
147         for (int i = 0; i < soundList.size(); i++) { // 音とキーのリストを一つずつ比較する
148             if ( ((Sound) soundList.get(i)).getDirection() != keyList.get(i)) {
149                 result = false;
150             }
151         }
152         return result;
153     }
154 
155     /**
156      * ゲーム終了を知らせる
157      */
158     private void showEndTitle() {
159         System.out.println("音記憶ゲームを終了します．");
160         System.exit(0);
161     }
162 }
```

```
1   package memory;
2
3   import javax.sound.midi.Instrument;
4   import javax.sound.midi.MidiChannel;
5   import javax.sound.midi.MidiSystem;
6   import javax.sound.midi.Soundbank;
7   import javax.sound.midi.Synthesizer;
8
9   /**
10   * Midiを演奏するクラス
11   *
12   * @author hashiyaman
13   * @version $Id: MidiPlayer.java,v 1.2 2006/10/26 07:33:56 hashiyaman Exp $
14   */
15
16  public class MidiPlayer {
17      MidiChannel channel = null;
18      Synthesizer synthesizer = null;
19      Soundbank soundbank = null;
20
21      /**
22       * コンストラクタ
23       */
24      public MidiPlayer(int instrumentID) {
25          try {
26              synthesizer = MidiSystem.getSynthesizer();
27              soundbank = synthesizer.getDefaultSoundbank();
28              synthesizer.open();
29
30              Instrument[] instruments = synthesizer.getDefaultSoundbank()
31                      .getInstruments();
32              synthesizer.loadInstrument(instruments[instrumentID]);
33              channel = synthesizer.getChannels()[0];
34          } catch (Exception ex) {
35              ex.printStackTrace();
36          }
37      }
38
39      /**
40       * ノートの音を上げる
41       */
42      public void noteOn(int noteNumber, int velocity) {
43          if (channel != null) {
44              channel.noteOn(noteNumber, velocity);
45          }
46      }
47
48      public void noteOff(int noteNumber, int velocity) {
49          if (channel != null) {
50              channel.noteOff(noteNumber, velocity);
51          }
52      }
53
54      public void close() {
55          try {
56              synthesizer.close();
57          } catch (Exception e) {
58              if (channel != null)
59                  channel.allNotesOff();
60          }
61      }
62  }
```

```java
package memory;

import java.util.Random;

/**
 * ゲーム中に記憶する音の音階と方向を表すクラス
 *
 * @author hashiyaman
 * @version $Id: Sound.java,v 1.2 2006/10/26 07:33:56 hashiyaman Exp $
 */
public class Sound {
    private final int NUMBER_OF_KEYS = 4;

    private int key; // 音階
    private String direction; // 音の方向

    /**
     * コンストラクタ
     */
    public Sound() {
        createSound();
    }

    /**
     * ランダムな音を生成する（音階と音の方向はセットで決まる）
     */
    private void createSound() {
        Random random = new Random();
        this.key = random.nextInt(NUMBER_OF_KEYS);

        switch (this.key) {
            case 0:
                this.direction = "↑";
                break;

            case 1:
                this.direction = "↓";
                break;

            case 2:
                this.direction = "←";
                break;

            case 3:
                this.direction = "→";
                break;
        }
    }

    public String getDirection() {
        return direction;
    }

    public void setDirection(String direction) {
        this.direction = direction;
    }

    public int getKey() {
        return key;
    }

    public void setKey(int key) {
        this.key = key;
```



```java
    }
}
```

2007/01/31
さうんど おんりぃ
最終報告書

ソースコード：聖徳太子ゲーム（Java）

```
package syotokutaishi;

import java.awt.Color;
import java.awt.Font;

/**
 * キャンバスを表現するクラス
 *
 * 各種書き込みメソッドにより，GUI描画を行なうことができます．
 * 実際の書き込み処理はCanvasPanelに委譲します．
 * （余計な処理をカプセル化していますので，中身を知りたい人はCanvasPanel(BWindow.java内)を参照せよ）
 *
 * @author macchan
 * @version 2.0
 */
public class BCanvas {

    private CanvasPanel canvasPanel;
    private CanvasKeyEventHandler keyHandler;
    private CanvasMouseEventHandler mouseHandler;

    /**
     * コンストラクタ
     */
    public BCanvas(CanvasPanel canvasPanel, CanvasKeyEventHandler keyHandler,
            CanvasMouseEventHandler mouseHandler) {
        this.canvasPanel = canvasPanel;
        this.keyHandler = keyHandler;
        this.mouseHandler = mouseHandler;
    }

    /**************************************************
     * 描画関連（第7,8回）
     **************************************************/

    /**
     * 線を引きます
     * 使用例:
     * 座標(10, 10) から 座標(20, 20)へ黒い線を引く場合
     * drawLine(Color.BLACK, 10, 10, 20, 20);
     */
    public void drawLine(Color color, double x1, double y1, double x2, double y2) {
        canvasPanel.drawLine(color, x1, y1, x2, y2);
    }

    /**
     * 塗りつぶした三角形を書きます
     * 使用例:
     * 座標A(10, 10), 座標B(20, 20), 座標C(30,30)を頂点とする三角形を書く場合
     * drawFillTriangle(Color.BLACK, 10, 10, 20, 20, 30, 30);
     */
    public void drawFillTriangle(Color color, double x1, double y1, double x2,
            double y2, double x3, double y3) {
        canvasPanel.drawFillTriangle(color, x1, y1, x2, y2, x3, y3);
    }

    /**
     * 円弧を書きます
     * 角度の単位は度です（0〜360度）
     * startAngleには弧を描き始める角度
     *         90
     * 180           0
     *       270
```



```
     * arcAngleには，弧全体の角度を書きます．弧は反時計回りに書かれます
     * 使用例:
     * 座標(10, 10)を左上として，高さ100, 幅100 の円弧を書く場合
     * drawDrawArc(Color.BLACK, 10, 10, 100, 100, 0, 360);
     */
    public void drawArc(Color color, double x, double y, double width,
            double height, double startAngle, double arcAngle) {
        canvasPanel.drawArc(color, x, y, width, height, startAngle, arcAngle);
    }

    /**
     * 塗りつぶした円を書きます
     * startAngleには弧を描き始める角度
     *         90
     * 180           0
     *       270
     * arcAngleには，弧全体の角度を書きます．弧は反時計回りに書かれます
     * 使用例:
     * 座標(10, 10)を左上として，高さ100, 幅100 左半分の円を書く場合
     * drawDrawArc(Color.BLACK, 10, 10, 100, 100, 90, 180);
     */
    public void drawFillArc(Color color, double x, double y, double width,
            double height, double startAngle, double arcAngle) {
        canvasPanel.drawFillArc(color, x, y, width, height, startAngle,
                arcAngle);
    }

    /**************************************************
     * 描画関連（第9回以降）
     **************************************************/

    /**
     * 文字を書きます
     */
    public void drawText(Color color, String text, double x, double y) {
        canvasPanel.drawText(color, text, x, y);
    }

    /**
     * （フォントサイズを指定して）文字を書きます
     */
    public void drawText(Color color, String text, double x, double y, Font font) {
        canvasPanel.drawText(color, text, x, y, font);
    }

    /**
     * 画像を書きます
     */
    public void drawImage(String filename, double x, double y) {
        canvasPanel.drawImage(filename, x, y);
    }

    /**
     * 画像を書きます
     * （幅と高さを引数にとり，その大きさに拡大，縮小します）
     */
    public void drawImage(String filename, double x, double y, double width,
            double height) {
        canvasPanel.drawImage(filename, x, y, width, height);
    }

    /**************************************************
     * フォント文字サイズの取得
```

```
     ****************************************************/

    /**
     * テキストの幅を取得します
     */
    public int getTextWidth(String text, Font font) {
        return canvasPanel.getTextWidth(text, font);
    }

    /**
     * テキストの高さを取得します
     */
    public int getTextHeight(String text, Font font) {
        return canvasPanel.getTextHeight(text, font);
    }

    /****************************************************
     * 画像サイズの取得
     ****************************************************/

    /**
     * 画像の幅を取得します
     */
    public int getImageWidth(String filename) {
        return canvasPanel.getImageWidth(filename);
    }

    /**
     * 画像の高さを取得します
     */
    public int getImageHeight(String filename) {
        return canvasPanel.getImageHeight(filename);
    }

    /****************************************************
     * 更新関連
     ****************************************************/

    /**
     * キャンバス全体を白く塗りつぶします
     */
    public void clear() {
        canvasPanel.clear();
    }

    /**
     * キャンバスを更新（再描画）します
     */
    public void update() {
        canvasPanel.update();
        keyHandler.update();
        mouseHandler.update();
    }

    /****************************************************
     * キー入力関連
     ****************************************************/

    /**
     * 押されたキーのコードを取得します
     */
    public int getKeyCode() {
        return keyHandler.key();
```



```
    }

    /**
     * 何らかのキーが押されたかどうかを調べます（継続は含まない）
     */
    public boolean isKeyDown() {
        return keyHandler.isKeyDown();
    }

    /**
     * 指定されたキーが押されている状態かどうかを調べます（継続も含む）
     */
    public boolean isKeyPressing(int keycode) {
        return keyHandler.isKeyPressing(keycode);
    }

    /****************************************************
     * マウス入力関連
     ****************************************************/

    /**
     * マウスのX座標を取得します
     */
    public int getMouseX() {
        return mouseHandler.mouseX();
    }

    /**
     * マウスのY座標を取得します
     */
    public int getMouseY() {
        return mouseHandler.mouseY();
    }

    /**
     * マウスが押されているかどうか調べます
     */
    public boolean isMouseDown() {
        return mouseHandler.isMouseDown();
    }

    /**
     * 右のマウスボタンが押されているかどうか調べます
     */
    public boolean isRightMouseDown() {
        return mouseHandler.isRightMouseDown();
    }

    /**
     * 左のマウスボタンが押されているかどうか調べます
     */
    public boolean isLeftMouseDown() {
        return mouseHandler.isLeftMouseDown();
    }

    /**
     * クリックかどうか調べます
     * (何回でのクリックも反応します)
     */
    public boolean isClick() {
        return mouseHandler.isClick();
    }
```

```
    /**
     * シングルクリックかどうか調べます
     */
    public boolean isSingleClick() {
        return mouseHandler.isSingleClick();
    }

    /**
     * ダブルクリックかどうか調べます
     */
    public boolean isDoubleClick() {
        return mouseHandler.isDoubleClick();
    }

    /**
     * ドラッグ中かどうか調べます
     */
    public boolean isDragging() {
        return mouseHandler.isDragging();
    }

    /*****************************************************
     * その他
     *****************************************************/

    /**
     * 指定された秒数待ちます
     */
    public void sleep(double seconds) {
        try {
            Thread.sleep((long) (seconds * 1000));
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /**
     * キャンパスの幅を取得します
     */
    public int getCanvasWidth() {
        return canvasPanel.getWidth();
    }

    /**
     * キャンパスの高さを取得します
     */
    public int getCanvasHeight() {
        return canvasPanel.getHeight();
    }
}
```

```
package syotokutaishi;

import bsound.BSoundSystem;
import bsound.framework.BSoundPlayer;

/**
 * 初心者用　音出しクラス．　オブプロ履修者のために．
 * mp3, wav, midファイルを簡単に制御できます．
 *
 * 1．基本的な使い方
 * BSound sound = new BSound("sample.wav");
 * sound.loop();
 *
 * 2．いちいちインスタンスを生成しない簡易メソッドを使う場合
 * BSound.play("sample.wav");
 *
 * 1．はＢＧＭ，2．は効果音に最適です．
 * サンプルコードBSoundTestを参照ください．
 *
 * このクラスで音を再生した場合デフォルトでストリーミング再生を行いますが，
 * 反応速度が重要な場合は，メモリにロードしておく必要があります．
 * BSound.load("sample.wav");
 * 当然ながら，BGM等の長いファイルは，ロードするとメモリを圧迫します．気をつけてください．
 *
 * なお，現在のバージョンでは，midiファイルの音量調節はできません．
 *
 * @author macchan
 */
public class BSound {

    /*********************************************
     * クラスメソッド
     *********************************************/

    /**
     * 再生する（止められません）
     */
    public static final void play(String filename) {
        new BSound(filename).play();
    }

    /**
     * ボリュームを指定して再生する（止められません）
     */
    public static final void play(String filename, int volume) {
        BSound sound = new BSound(filename);
        sound.setVolume(volume);
        sound.play();
    }

    /**
     * メモリ上にサウンドデータを読み込みます(反応が早くなりますが，メモリ領域が必要です)
     */
    public static final void load(String filename) {
        BSoundSystem.load(filename);
    }

    /*********************************************
     * BSound本体
     *********************************************/

    private BSoundPlayer player = null;
```



```
    public BSound(String filename) {
        player = BSoundSystem.createPlayer(filename);
    }

    /*
     * ------------------------ 操作系 ------------------------
     */

    /**
     * 再生します
     */
    public void play() {
        player.setLoop(false);
        player.play();
    }

    /**
     * ループ再生します
     */
    public void loop() {
        player.setLoop(true);
        player.play();
    }

    /**
     * 停止します
     */
    public void stop() {
        player.stop();
    }

    /**
     * 再生中かどうか調べます
     */
    public boolean isPlaying() {
        return player.getState() == BSoundPlayer.PLAYING;
    }

    /* -------------------------
     * ボリュームコントロール系（ボリュームは0-100の100段階設定ができます）
     * ------------------------- */

    /**
     * 現在のボリュームを取得します．
     */
    public int getVolume() {
        return player.getVolume();
    }

    /**
     * ボリュームを設定します．
     */
    public void setVolume(int volume) {
        player.setVolume(volume);
    }

    /**
     * 初期ボリュームを取得します
     */
    public int getDefaultVolume(){
        return player.getDefaultVolume();
    }
```

```
127     }
```

```java
package syotokutaishi;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import javax.imageio.ImageIO;
import javax.swing.JFrame;

/**
 * ウインドウを表現するクラス
 *
 * @author macchan
 * @version 2.0
 */
public class BWindow {

    private JFrame frame;
    private BCanvas canvas;

    /**
     * コンストラクタ
     */
    public BWindow() {
        //フレーム生成
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //パネル＆イベントハンドラ生成
        CanvasPanel canvasPanel = new CanvasPanel();
        frame.getContentPane().add(canvasPanel);

        CanvasKeyEventHandler keyHandler = new CanvasKeyEventHandler();
        CanvasMouseEventHandler mouseHandler = new CanvasMouseEventHandler();
        frame.addKeyListener(keyHandler);
        frame.getContentPane().addKeyListener(keyHandler);
        canvasPanel.addKeyListener(keyHandler);
        canvasPanel.addMouseListener(mouseHandler);
        canvasPanel.addMouseMotionListener(mouseHandler);

        //キャンバス生成
        canvas = new BCanvas(canvasPanel, keyHandler, mouseHandler);
    }
```



```java
    /**
     * 位置を設定する
     */
    public void setLocation(int x, int y) {
        frame.setLocation(x, y);
    }

    /**
     * 大きさを設定する
     */
    public void setSize(int width, int height) {
        frame.setSize(width, height);
    }

    /**
     * (この) ウインドウを表示する
     */
    public void show() {
        frame.setVisible(true);
    }

    /**
     * 書き込みができるCanvasインスタンスを取得する
     */
    public BCanvas getCanvas() {
        return canvas;
    }

}

/**
 * キーのイベントを拾うクラス
 */
class CanvasKeyEventHandler implements KeyListener {

    //定数
    public static final int NULL_KEY_CODE = -1;
    public static final int NULL_MOUSE_LOCATION = -1;

    //入力イベント関連
    private KeyEvent bufferKeyEvent = null;
    private KeyEvent capturedKeyEvent = null;

    private Set pressingKeys = new HashSet();

    /*****************************************************
     * リスナインターフェイスの実装
     *****************************************************/

    public void keyPressed(KeyEvent e) {
        bufferKeyEvent = e;
        pressingKeys.add(new Integer(e.getKeyCode()));
    }

    public void keyReleased(KeyEvent e) {
        pressingKeys.remove(new Integer(e.getKeyCode()));
    }

    public void keyTyped(KeyEvent e) {
    }

    /*****************************************************
     * 公開インターフェイス
```

```
127         ************************************************/
128
129         public int key() {
130             if (capturedKeyEvent != null) {
131                 return capturedKeyEvent.getKeyCode();
132             } else {
133                 return NULL_KEY_CODE;
134             }
135         }
136
137         public boolean isKeyDown() {
138             return capturedKeyEvent != null;
139         }
140
141         public boolean isKeyPressing(int keycode) {
142             return pressingKeys.contains(new Integer(keycode));
143         }
144
145         /************************************************
146          * 更新関連
147          ************************************************/
148
149         public void update() {
150             capturedKeyEvent = bufferKeyEvent;
151             bufferKeyEvent = null;
152         }
153
154     }
155
156     /**
157      * マウスのイベントを拾うクラス
158      */
159     class CanvasMouseEventHandler implements MouseListener, MouseMotionListener {
160
161         //定数
162         public static final int NULL_MOUSE_LOCATION = -1;
163
164         //入力イベント関連
165         private int mouseX = NULL_MOUSE_LOCATION;
166         private int mouseY = NULL_MOUSE_LOCATION;
167         private boolean isDragging = false;
168         private MouseEvent bufferMouseEvent = null;
169         private MouseEvent capturedMouseEvent = null;
170
171         private Set pressingMouses = new HashSet();
172
173         /************************************************
174          * リスナインターフェイスの実装
175          ************************************************/
176
177         public void mousePressed(MouseEvent e) {
178             bufferMouseEvent = e;
179             pressingMouses.add(new Integer(e.getButton()));
180         }
181
182         public void mouseReleased(MouseEvent e) {
183             bufferMouseEvent = e;
184             pressingMouses.remove(new Integer(e.getButton()));
185             isDragging = false;
186         }
187
188         public void mouseClicked(MouseEvent e) {
189             bufferMouseEvent = e;
```



```
190         }
191
192         public void mouseEntered(MouseEvent e) {
193             bufferMouseEvent = e;
194         }
195
196         public void mouseExited(MouseEvent e) {
197             bufferMouseEvent = e;
198             isDragging = false;
199         }
200
201         public void mouseMoved(MouseEvent e) {
202             bufferMouseEvent = e;
203         }
204
205         public void mouseDragged(MouseEvent e) {
206             bufferMouseEvent = e;
207             isDragging = true;
208         }
209
210         /************************************************
211          * 公開インターフェイス
212          ************************************************/
213
214         public int mouseX() {
215             return mouseX;
216         }
217
218         public int mouseY() {
219             return mouseY;
220         }
221
222         public boolean isMouseDown() {
223             return isRightMouseDown() || isLeftMouseDown();
224         }
225
226         public boolean isRightMouseDown() {
227             if (pressingMouses.contains(new Integer(MouseEvent.BUTTON3))) {
228                 return true;
229             } else {
230                 return capturedMouseEvent == null ? false : capturedMouseEvent
231                         .getButton() == MouseEvent.BUTTON3;
232             }
233         }
234
235         public boolean isLeftMouseDown() {
236             if (pressingMouses.contains(new Integer(MouseEvent.BUTTON1))) {
237                 return true;
238             } else {
239                 return capturedMouseEvent == null ? false : capturedMouseEvent
240                         .getButton() == MouseEvent.BUTTON1;
241             }
242         }
243
244         public boolean isClick() {
245             return capturedMouseEvent == null
246                     ? false
247                     : capturedMouseEvent.getID() == MouseEvent.MOUSE_CLICKED;
248         }
249
250         public boolean isSingleClick() {
251             return capturedMouseEvent == null
252                     ? false
```

```
253                     : capturedMouseEvent.getID() == MouseEvent.MOUSE_CLICKED
254                             && capturedMouseEvent.getClickCount() == 1;
255         }
256
257         public boolean isDoubleClick() {
258             return capturedMouseEvent == null
259                     ? false
260                     : capturedMouseEvent.getID() == MouseEvent.MOUSE_CLICKED
261                             && capturedMouseEvent.getClickCount() == 2;
262         }
263
264         public boolean isDragging() {
265             return isDragging;
266         }
267
268         /**************************************************
269          * 更新関連
270          **************************************************/
271
272         public void update() {
273             capturedMouseEvent = bufferMouseEvent;
274             if (capturedMouseEvent != null) {
275                 mouseX = capturedMouseEvent.getX();
276                 mouseY = capturedMouseEvent.getY();
277             }
278
279             bufferMouseEvent = null;
280         }
281
282     }
283
284     /**
285      * Canvasの委譲先クラス
286      * Canvasに書かれる形式をバッファし，Swing形式に変換し出力します．
287      */
288     class CanvasPanel extends Canvas implements ComponentListener {
289
290         //定数
291         private static final int FILP_BUFFERSTRATEGY = 3;
292         private static final Graphics2D NULL_GRAPHICS = (Graphics2D) (new BufferedImage(
293                 1, 1, BufferedImage.TYPE_3BYTE_BGR).createGraphics());
294
295         //属性
296         private Graphics2D offGraphics;
297
298         /**
299          * コンストラクタ
300          */
301         public CanvasPanel() {
302             addComponentListener(this);
303             refreshOffGraphics();
304         }
305
306         /**************************************************
307          * BufferStrategy関連
308          **************************************************/
309
310         private void initializeBufferStrategy() {
311             createBufferStrategy(FILP_BUFFERSTRATEGY);
312             refreshOffGraphics();
313         }
314
315         private void refreshOffGraphics() {
```



```
316             offGraphics = getGraphics2D();
317             offGraphics.setColor(Color.WHITE);
318             offGraphics.fillRect(0, 0, getWidth(), getHeight());
319         }
320
321         private Graphics2D getGraphics2D() {
322             Graphics2D g2d = (Graphics2D) getBufferStrategy().getDrawGraphics();
323             if (g2d != null) {
324                 return g2d;
325             } else {
326                 return NULL_GRAPHICS;
327             }
328         }
329
330         private void flip() {
331             getBufferStrategy().show();
332         }
333
334         /**************************************************
335          * Component Listener関連
336          **************************************************/
337
338         public void componentHidden(ComponentEvent e) {
339         }
340         public void componentMoved(ComponentEvent e) {
341         }
342         public void componentResized(ComponentEvent e) {
343             initializeBufferStrategy();
344         }
345         public void componentShown(ComponentEvent e) {
346         }
347
348         /**************************************************
349          * 描画関連
350          **************************************************/
351
352         public void drawLine(Color color, double x1, double y1, double x2, double y2) {
353             offGraphics.setColor(color);
354             offGraphics.drawLine((int) x1, (int) y1, (int) x2, (int) y2);
355         }
356
357         public void drawFillTriangle(Color color, double x1, double y1, double x2,
358                 double y2, double x3, double y3) {
359             offGraphics.setColor(color);
360             offGraphics.fillPolygon(new int[]{(int) x1, (int) x2, (int) x3},
361                     new int[]{(int) y1, (int) y2, (int) y3}, 3);
362         }
363
364         public void drawArc(Color color, double x, double y, double width,
365                 double height, double startAngle, double arcAngle) {
366             offGraphics.setColor(color);
367             offGraphics.drawArc((int) x, (int) y, (int) width, (int) height,
368                     (int) startAngle, (int) arcAngle);
369         }
370
371         public void drawFillArc(Color color, double x, double y, double width,
372                 double height, double startAngle, double arcAngle) {
373             offGraphics.setColor(color);
374             offGraphics.fillArc((int) x, (int) y, (int) width, (int) height,
375                     (int) startAngle, (int) arcAngle);
376         }
377
378         public void drawText(Color color, String text, double x, double y) {
```

```
379             offGraphics.setColor(color);
380             offGraphics.drawString(text, (int) x, (int) y);
381         }
382 
383         public void drawText(Color color, String text, double x, double y, Font font) {
384             FontMetrics fontMetrics = offGraphics.getFontMetrics(font);
385             int topY = (int) y + fontMetrics.getAscent();
386 
387             //前処理
388             offGraphics.setColor(color);
389             Font originalFont = offGraphics.getFont();
390             offGraphics.setFont(font);
391 
392             //処理
393             offGraphics.drawString(text, (int) x, topY);
394 
395             //後処理
396             offGraphics.setFont(originalFont);
397         }
398 
399         public void drawImage(String filename, double x, double y, double width,
400                 double height) {
401             BufferedImage image = BImageProvider.getInstance().getImage(filename);
402             double scaleX = width / image.getWidth();
403             double scaleY = height / image.getHeight();
404             AffineTransform transform = AffineTransform.getScaleInstance(scaleX,
405                     scaleY);
406             AffineTransformOp transformOp = new AffineTransformOp(transform,
407                     AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
408             drawImage(image, transformOp, x, y);
409         }
410 
411         public void drawImage(String filename, double x, double y) {
412             BufferedImage image = BImageProvider.getInstance().getImage(filename);
413             drawImage(image, null, x, y);
414         }
415 
416         private void drawImage(BufferedImage image, AffineTransformOp transformOp,
417                 double x, double y) {
418             offGraphics.drawImage(image, transformOp, (int) x, (int) y);
419         }
420 
421         /**************************************************
422          * フォント文字サイズの取得
423          **************************************************/
424 
425         public int getTextWidth(String text, Font font) {
426             FontMetrics fontMetrics = offGraphics.getFontMetrics(font);
427             return fontMetrics.stringWidth(text);
428         }
429 
430         public int getTextHeight(String text, Font font) {
431             FontMetrics fontMetrics = offGraphics.getFontMetrics(font);
432             return fontMetrics.getHeight();
433         }
434 
435         /**************************************************
436          * 画像サイズの取得
437          **************************************************/
438 
439         public int getImageWidth(String filename) {
440             BufferedImage image = BImageProvider.getInstance().getImage(filename);
441             return image.getWidth();
```



```
442         }
443 
444         public int getImageHeight(String filename) {
445             BufferedImage image = BImageProvider.getInstance().getImage(filename);
446             return image.getHeight();
447         }
448 
449         /**************************************************
450          * 更新関連
451          **************************************************/
452 
453         public void update() {
454             flip();
455         }
456 
457         public void clear() {
458             refreshOffGraphics();
459         }
460 
461     }
462 
463     /**
464      * 画像読み込みクラス
465      */
466     class BImageProvider {
467 
468         //定数
469         private static final int DUMMY_IMAGE_FONT_SIZE = 12;
470         private static final Font DUMMY_IMAGE_FONT = new Font("Dialog", Font.PLAIN,
471                 DUMMY_IMAGE_FONT_SIZE);
472 
473         /*********************************
474          * SingleTonの実装
475          *********************************/
476 
477         private static BImageProvider instance;
478 
479         public static BImageProvider getInstance() {
480             if (instance == null) {
481                 instance = new BImageProvider();
482             }
483             return instance;
484         }
485 
486         //属性
487         private Map images = new HashMap();
488 
489         /**
490          * コンストラクタ
491          */
492         private BImageProvider() {
493             super();
494         }
495 
496         /**
497          * 画像を取得する（なければ新しく生成）
498          */
499         public BufferedImage getImage(String filename) {
500             if (!images.containsKey(filename)) {
501                 images.put(filename, prepareImage(filename));
502             }
503             return (BufferedImage) images.get(filename);
504         }
```

```
505
506         /*********************************
507          * 以下, 画像読み込み処理
508          *********************************/
509
510         private BufferedImage prepareImage(String filename) {
511             try {
512                 return loadImage(filename);
513             } catch (Exception ex) {
514                 return createDummyImage(filename);
515             }
516         }
517
518         private BufferedImage loadImage(String filename) throws IOException {
519             File f = new File(filename);
520             BufferedImage image = ImageIO.read(f);
521             return image;
522         }
523
524         private BufferedImage createDummyImage(String filename) {
525             //画像を生成する
526             int width = DUMMY_IMAGE_FONT_SIZE * filename.length();
527             int height = DUMMY_IMAGE_FONT_SIZE * 2;
528             BufferedImage image = new BufferedImage(width, height,
529                     BufferedImage.TYPE_4BYTE_ABGR);
530
531             //ダミー画像を書き込む
532             Graphics g = image.getGraphics();
533             g.setColor(Color.WHITE);
534             g.fillRect(0, 0, width - 1, height - 1);
535             g.setColor(Color.BLACK);
536             g.setFont(DUMMY_IMAGE_FONT);
537             g.drawRect(0, 0, width - 1, height - 1);
538             g.drawString(filename, 10, height / 2);
539             g.dispose();
540
541             return image;
542         }
543     }
```

```java
package syotokutaishi;

import java.io.*;

/**
 * コンソールからの入力メソッドを提供するクラス
 *
 * @author Manabu Sugiura
 * @version $Id: Input.java,v 1.1 2006/10/26 07:33:57 hashiyaman Exp $
 */
public class Input {

    private static BufferedReader br;

    /**
     * コンソールから文字を読み込む
     * @return String
     */
    public static String getString() {
        String returnString = null;
        try {
            if (br == null) {
                br = new BufferedReader(new InputStreamReader(System.in));
            }
            returnString = br.readLine();
            return returnString;

        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    /**
     * コンソールから数字を読み込む
     * @return int
     */
    public static int getInt() {
        int returnInt = 0;
        returnInt = Integer.parseInt(getString());
        return returnInt;

    }

    /**
     * コンソールからdouble型の数字を読み込む
     * @return double
     */
    public static double getDouble() {
        double returnDouble = 0.0;
        returnDouble = Double.parseDouble(getString());
        return returnDouble;

    }

    /**
     * 入力された文字列がint型に変換できるかどうかを調べる
     * @return int
     */
    public static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
            return true;
```



```java
        } catch (NumberFormatException ex) {
            return false;
        }
    }

    /**
     * 入力された文字列がint型に変換できるかどうかを調べる
     * @return int
     */
    public static boolean isDouble(String str) {
        try {
            Double.parseDouble(str);
            return true;
        } catch (NumberFormatException ex) {
            return false;
        }
    }
}
```

```java
package syotokutaishi;

import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import static syotokutaishi.BSound.*;

public class SoundManager {
    private static final int NUMBER_OF_FLUITS = 8; // 問題に使うサウンドの数
    public static final int NUMBER_OF_QUESTIONS = 3; // 当てさせる音の数
    private static List<String> soundList = new ArrayList<String>(); // サウンドファイル

    /**
     * 音声をロードする
     *
     * @param サウンドファイルのパス
     */
    public static void loadSounds(String soundPath) {
        File soundDirectory = new File(soundPath);
        File[] sounds = soundDirectory.listFiles();

        // サウンドディレクトリ内の音声ファイルを全てロードする
        for (File sound : sounds) {
            load(soundPath + sound.getName());
        }
    }

    /**
     * ランダムな音声を生成する
     */
    public static List createSounds() {
        soundList.clear();

        while (soundList.size() < NUMBER_OF_QUESTIONS) {
            Random random = new Random();
            int fluit = random.nextInt(NUMBER_OF_FLUITS);
            switch (fluit) {
                case 0 :
                    if (!soundList.contains("ばなな")) {
                        soundList.add("ばなな");
                    }
                    break;
                case 1 :
                    if (!soundList.contains("ぶどう")) {
                        soundList.add("ぶどう");
                    }
                    break;
                case 2 :
                    if (!soundList.contains("いちご")) {
                        soundList.add("いちご");
                    }
                    break;
                case 3 :
                    if (!soundList.contains("きうい")) {
                        soundList.add("きうい");
                    }
                    break;
                case 4 :
                    if (!soundList.contains("れもん")) {
                        soundList.add("れもん");
                    }
                    break;
```



```java
                case 5 :
                    if (!soundList.contains("めろん")) {
                        soundList.add("めろん");
                    }
                    break;
                case 6 :
                    if (!soundList.contains("みかん")) {
                        soundList.add("みかん");
                    }
                    break;
                case 7 :
                    if (!soundList.contains("りんご")) {
                        soundList.add("りんご");
                    }
                    break;
                case 8 :
                    if (!soundList.contains("すいか")) {
                        soundList.add("すいか");
                    }
                    break;
            }
        }
        return soundList;
    }

    /**
     * 生成された音声を再生する
     *
     * @param サウンドファイルのパス
     */
    public static void playSound(String soundPath) {
        for (String fluit : soundList) {
            if (fluit.equals("ばなな")) {
                BSound.play(soundPath + "banana.wav");
            } else if (fluit.equals("ぶどう")) {
                BSound.play(soundPath + "budou.wav");
            } else if (fluit.equals("いちご")) {
                BSound.play(soundPath + "itigo.wav");
            } else if (fluit.equals("きうい")) {
                BSound.play(soundPath + "kiui.wav");
            } else if (fluit.equals("ぶどう")) {
                BSound.play(soundPath + "budou.wav");
            } else if (fluit.equals("れもん")) {
                BSound.play(soundPath + "lemon.wav");
            } else if (fluit.equals("めろん")) {
                BSound.play(soundPath + "meron.wav");
            } else if (fluit.equals("みかん")) {
                BSound.play(soundPath + "mikan.wav");
            } else if (fluit.equals("りんご")) {
                BSound.play(soundPath + "ringo.wav");
            } else if (fluit.equals("すいか")) {
                BSound.play(soundPath + "suika.wav");
            }
        }
    }
    /**
     * 正解・不正解の判定をする
     *
     * @param プレーヤーの答え
     * @return 正解・不正解
     */
    public static boolean judge(List<String> answerList) {
        for (String answer : answerList) {
```

```
127                 if (!soundList.contains(answer)) {
128                     return false;
129                 }
130             }
131             return true;
132         }
133     }
```

```java
package syotokutaishi;

import java.util.ArrayList;
import java.util.List;
import static syotokutaishi.SoundManager.*;

public class SyotokutaishiGame {
    private final String SOUND_PATH = "res/syotokutaishi/sound/"; // サウンドファイルのパス
    private int numberOfCorrect = 0; // 正解数

    /**
     * @param args
     */
    public static void main(String[] args) {
        SyotokutaishiGame game = new SyotokutaishiGame();
        game.run();
    }

    /**
     * 聖徳太子ゲームを実行する
     */
    public void run() {
        loadSounds(SOUND_PATH);
        showTitle();
        playGame();
        showEndTitle();
        System.exit(0);
    }

    /**
     * ゲームの開始を知らせる
     */
    private void showTitle() {
        System.out.println("聖徳太子ゲームを始めます．");
        BSound.play(SOUND_PATH + "start.wav");
        sleep(4);
        BSound.play(SOUND_PATH + "rule.wav");
        System.out.println("三種類の果物の名前を聞き分け，正解をひらがなで入力してください．");
        sleep(8);
        BSound.play(SOUND_PATH + "do.wav");
        System.out.println("それでは始めます．");
        sleep(4);
    }

    /**
     * ゲームを開始する
     */
    private void playGame() {
        createSounds();

        // 正解している限り，ゲームを続行する
        while (true) {
            playSound(SOUND_PATH);
            if (judge(getAnswer())) { // 正解だったら
                numberOfCorrect++;
                System.out.println("正解です！！");
                BSound.play(SOUND_PATH + "right.wav");
                sleep(1);
                BSound.play(SOUND_PATH + "correct.wav");
                sleep(3);
                System.out.println("次の問題に移ります．");
                BSound.play(SOUND_PATH + "next.wav");
                sleep(3);
```



```java
                SoundManager.createSounds();
            } else { // 不正解だったら
                System.out.println("不正解です！！");
                BSound.play(SOUND_PATH + "wrong.wav");
                sleep(1);
                BSound.play(SOUND_PATH + "incorrect.wav");
                sleep(3);
                System.out.println("正解数は" + numberOfCorrect + "問でした．");
                BSound.play(SOUND_PATH + "correctNum1.wav");
                sleep(2);
                BSound.play(SOUND_PATH + numberOfCorrect + ".wav");
                sleep(1);
                BSound.play(SOUND_PATH + "correctNum2.wav");
                sleep(2);
                break;
            }
        }
    }

    /**
     * プレーヤーの入力した答えを取得する
     *
     * @return プレーヤーの答え
     */
    private List<String> getAnswer() {
        List<String> answerList = new ArrayList<String>();

        while (answerList.size() < NUMBER_OF_QUESTIONS) {
            String answer = Input.getString();
            if (answerList.contains(answer)) { // 答えの重複をはじく
                System.out.println("同じ答えを入力しないでください．");
                continue;
            } else if (answer.equals("r")) { // rを入力するともう一度音声を再生する
                SoundManager.playSound(SOUND_PATH);
                continue;
            }
            answerList.add(answer);
        }
        return answerList;
    }

    /**
     * ゲームの終了を知らせる
     */
    private void showEndTitle() {
        System.out.println("聖徳太子ゲームを終了します．");
        BSound.play(SOUND_PATH + "end.wav");
        sleep(4);
    }

    /**
     * 指定された秒数待ちます
     */
    private void sleep(double seconds) {
        try {
            Thread.sleep((long) (seconds * 1000));
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

2007/01/31
さうんど おんりぃ
最終報告書

ソースコード：さうんどシュート（C++）

```
/**
 * ゲーム本体を実行するクラス
 *
 * @date 2006/11/05
 * @author hashiyaman
 */

#include "Game.h"

/*
 * コンストラクタ
 */
Game::Game(SDL_Surface *mainScreen)
{
    this->mainScreen = mainScreen;
    soundContainer = new SoundContainer("SoundShoot.csb");
    titleState = new TitleState(soundContainer,mainScreen);
    gamePlayState = new GamePlayState(soundContainer,mainScreen);
}


/*
 * ゲームを実行する
 */
void Game::run()
{
    while(true)
    {
        // タイトルを表示する
        int titleMessage = titleState->run();
        switch( titleMessage )
        {
        case START_GAME:
            {
                int GameMessage = gamePlayState->run();
                if( GameMessage == QUIT_GAME )
                {
                    return;
                }
            }
            break;
        case QUIT_GAME:
            return;
        }
    }
}

/*
 * デストラクタ
 */
Game::~Game()
{
    delete soundContainer;
    delete titleState;
    delete gamePlayState;
}

```

```
1   #ifndef _GAME_H_
2   #define _GAME_H_
3
4   #include <string>
5   #include "SoundContainer.h"
6   #include "TitleState.h"
7   #include "GamePlayState.h"
8
9   class Game
10  {
11      SoundContainer *soundContainer;
12      SDL_Surface *mainScreen;
13      TitleState *titleState;
14      GamePlayState *gamePlayState;
15
16  public:
17      Game(SDL_Surface *mainScreen);
18      ~Game();
19
20      void run();
21  };
22
23  #endif
```

```
/**
 * ゲーム中を表現するクラス
 */

#include "GamePlayState.h"

const Uint8 GamePlayState::SHOOT = SDLK_RETURN;
const Uint8 GamePlayState::QUIT = SDLK_ESCAPE;
const Uint8 GamePlayState::SKIP = SDLK_SPACE;

const double GamePlayState::M_PI = 3.14159265358979323846264338327 9;
const float GamePlayState::PLAYER_X = 320;
const float GamePlayState::PLAYER_Y = 240;

/*
 * コンストラクタ
 */
GamePlayState::GamePlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen)
{
    this->soundPlayer = new SoundPlayer(soundContainer,"GamePlayStateSound.txt");
    this->soundContainer = soundContainer;
    this->mainScreen = mainScreen;

    // 状態の監視をしたいキーを列挙して渡す
    KeyTable useKeyTable;
    useKeyTable.push_back(QUIT);
    useKeyTable.push_back(SHOOT);
    useKeyTable.push_back(SKIP);
    keyState = new KeyState(useKeyTable);
}

/*
 * デストラクタ
 */
GamePlayState::~GamePlayState()
{
    delete keyState;
    delete soundPlayer;
}

/*
 * ゲーム中の処理を行う
 */
int GamePlayState::run()
{
    return this->runFrame();
}

/*
 * 一フレーム内の処理を行う
 */
int GamePlayState::runFrame()
{
    // フォントと色と内容の設定を行う
    SDL_Color textColor = {255,255,255};
    TTF_Font *font = TTF_OpenFont( "Headache.ttf", 28 );
    SDL_Surface *text = TTF_RenderText_Blended( font, "GAME PLAYING", textColor );

    int gameState;
    soundPlayer->loopPlay("START_GAME");

    int score = -1;
    float targetX = 0;
```



```
    float targetY = 230;

    while(true)
    {
        CriError error = CRIERR_OK;

        // 入力を取得
        keyState->update();
        Uint8 *keys = SDL_GetKeyState(NULL);

        // ゲームが終了された場合
        if( !PollEvent() || keyState->down(QUIT) )
        {
            gameState = QUIT_GAME;
            break;
        }

        // 説明をスキップする
        if( soundPlayer->getStatus("START_GAME") == CriAuPlayer::STATUS_PLAYING
            && keyState->down(SKIP) )
        {
            soundPlayer->stop("START_GAME");
        }

        // ゲームの説明を終えた後に，ゲーム中の音声の再生を開始する
        if( (targetX < 640 && targetY < 480)
            && soundPlayer->getStatus("START_GAME") == CriAuPlayer::STATUS_PLAYEND )
        {
            moveSound(&targetX, &targetY);
        }
        else if (targetX >= 640 || targetY >= 480)
        {
            showResult(score);

            if ( keyState->down(SKIP) )
            {
                gameState = GO_TITLE;
                break;
            }
        }

        CriAuObj::ExecuteMain(error); // 音声の状態を更新する

        // 音を狙って，得点を計算する(1ゲームにつき1回のみ)
        if( keyState->down(SHOOT) && score < 0
            && soundPlayer->getStatus("START_GAME") == CriAuPlayer::STATUS_PLAYEND)
        {
            score = calculateScore(targetX, targetY);
        }

        ClearScreen(mainScreen); // 画面をクリアする

        this->draw(10,10,text,mainScreen); // 文字の描画


        if( score >= 0 ) // スコアを表示する
        {
            string scoreText = "SCORE:" + boost::lexical_cast<string>(score);
            SDL_Surface *text = TTF_RenderText_Blended( font, scoreText.c_str(), textColor );
            this->draw(10,30,text,mainScreen); // スコアの描画
        }

        SDL_Flip(mainScreen);
```

```
127
128             SDL_Delay(50); // CPU使用率が100%になるのを防ぐ
129         }
130
131         // 音の停止
132         CriError error = CRIERR_OK;
133         soundPlayer->stopAll();
134         CriAuObj::ExecuteMain(error);
135
136         TTF_CloseFont(font);
137         SDL_FreeSurface(text);
138
139         return gameState;
140     }
141
142     /*
143      * 音を動かす
144      */
145     void GamePlayState::moveSound(float *targetX, float *targetY)
146     {
147         *targetX += 2;
148         *targetY += 0;
149
150         // 距離に応じてボリュームを調整する
151         Float32 volume = this->calculateVolume( *targetX, *targetY );
152         soundPlayer->setVolume("MOVE1", volume);
153
154         CriAuSendLevel sendLevel = this->calculateSpeakerSendLevel( *targetX, *targetY );
155         playSound("MOVE1", sendLevel);
156     }
157
158     /*
159      * ボリュームを計算する
160      */
161     Float32 GamePlayState::calculateVolume(float targetX, float targetY)
162     {
163         float distance = calculateDistance(targetX, targetY);
164
165         // 距離に応じてボリュームを調整する
166         Float32 volume;
167         if( distance == 0 )
168         {
169             volume = 1.0f;
170         }
171         else
172         {
173             //volume = 1.0f / sqrtf(distance) * 2.0f;
174             volume = sqrtf(distance) / -10.0f + 2.0f;
175         }
176
177         if( volume < 0.0f)
178             volume = 0.0f;
179
180         return volume;
181     }
182
183     /*
184      * 距離を計算する
185      */
186     Float32 GamePlayState::calculateDistance(float targetX, float targetY)
187     {
188         // 距離を計算する
189         float xDistance = targetX - PLAYER_X;
```



```
190         float yDistance = targetY - PLAYER_Y;
191         float distance = xDistance * xDistance + yDistance * yDistance;
192
193         return sqrtf(distance);
194     }
195
196     /*
197      * オブジェクト間の角度を計算する
198      */
199     Float32 GamePlayState::calculateInterObjectAngle(float targetX, float targetY)
200     {
201         // オブジェクト間の角度を計算する
202         float radian = atan2( PLAYER_Y - targetY, PLAYER_X - targetX);
203
204         // 弧度法から度数法へ変換
205         // CRIAudioでは正面が0度なので、画面と音の整合性を取るため角度を足す
206         int angle = static_cast<int>( ( radian / M_PI * 180 ) + 90 );
207         angle = angle % 360;
208
209         return static_cast<Float32>(-angle);
210     }
211
212     /*
213      * スピーカーのセンドレベルを計算する
214      */
215     CriAuSendLevel GamePlayState::calculateSpeakerSendLevel(float targetX, float targetY)
216     {
217         // オブジェクト間の角度を計算する
218         Float32 angle = this->calculateInterObjectAngle(targetX, targetY);
219
220         // 角度からセンドレベルを計算する
221         Float32 left;
222         Float32 right;
223         Float32 leftSurround;
224         Float32 rightSurround;
225         Float32 center;
226         CriAuUty::CalcSendLevel5Speakers(angle, &left, &right, &leftSurround, &rightSurround, &center);
227
228         // センドレベルを設定する
229         CriAuSendLevel sendLevel;
230         sendLevel.SetLeft(left);
231         sendLevel.SetRight(right);
232         sendLevel.SetLeftSurround(leftSurround);
233         sendLevel.SetRightSurround(rightSurround);
234         sendLevel.SetCenter(center);
235
236         return sendLevel;
237     }
238
239     /*
240      * オブジェクトの音を鳴らす
241      */
242     void GamePlayState::playSound(string soundName, const CriAuSendLevel &sendLevel)
243     {
244         soundPlayer->setSendLevel(soundName, sendLevel);
245         soundPlayer->loopPlay(soundName);
246     }
247
248     /*
249      * 得点を計算する
250      */
251     int GamePlayState::calculateScore(float targetX, float targetY)
252     {
```

```
253         float distance = calculateDistance(targetX, targetY);
254         int score = 100 - distance; // 得点
255 
256         if( score < 0)
257         {
258             score = 0;
259         }
260 
261         return score;
262     }
263 
264     /*
265      * 描画を行う
266      */
267     void GamePlayState::draw(int x, int y, SDL_Surface *source, SDL_Surface *destination )
268     {
269         SDL_Rect position;
270         position.x = x;
271         position.y = y;
272         SDL_BlitSurface(source, NULL, destination, &position);
273     }
274 
275     /*
276      * 結果を知らせる
277      */
278     void GamePlayState::showResult(int score)
279     {
280         if( soundPlayer->getStatus("END_GAME") == CriAuPlayer::STATUS_PLAYEND
281             || soundPlayer->getStatus("END_GAME") == CriAuPlayer::STATUS_STOP )
282         {
283             soundPlayer->play("END_GAME");
284             //readNumber(score);
285         }
286     }
287 
288     /*
289      * 得点を読み上げる（未完成）
290      */
291     void GamePlayState::readNumber(int score)
292     {
293         string scoreString = boost::lexical_cast<string>(score);
294         static string currentDigit = scoreString.substr(0,1); // 現在読み上げているケタ
295 
296         if( soundPlayer->getStatus("SCORE") == CriAuPlayer::STATUS_PLAYEND
297             || soundPlayer->getStatus("SCORE") == CriAuPlayer::STATUS_STOP )
298         {
299             for( int i = 0; i < scoreString.length(); i++ )
300             {
301                 currentDigit = scoreString.substr(i,1);
302                 soundPlayer->play(currentDigit);
303             }
304             soundPlayer->play("SCORE");
305 
306         }
307     }
```

```
1   #ifndef _GAMEPLAYSTATE_H_
2   #define _GAMEPLAYSTATE_H_
3
4   #include "SoundPlayer.h"
5   #include "KeyState.h"
6   #include "StateMessages.h"
7
8   #include <SDL.h>
9   #include <SDL_ttf.h>
10  #include <string>
11  #include <list>
12  #include <boost/bind.hpp>
13  #include <boost/lexical_cast.hpp>
14  #include <algorithm>
15  #include <cstdlib>
16
17  #include "SDLCommonFunctions.h"
18  using std::string;
19
20  enum ObjectCondition
21  {
22      ALIVE,
23      DEAD
24  };
25
26  class GamePlayState
27  {
28      SoundPlayer *soundPlayer;
29      SoundContainer *soundContainer;
30      SDL_Surface *mainScreen;
31
32      KeyState *keyState;
33      static const Uint8 SHOOT;
34      static const Uint8 QUIT;
35      static const Uint8 SKIP;
36      static const double M_PI;
37      static const float PLAYER_X;
38      static const float PLAYER_Y;
39
40      int runFrame();    // 1フレーム内の処理を行う
41      void draw(int x, int y, SDL_Surface *source, SDL_Surface *destination );
42      void moveSound(float *targetX, float *targetY);
43      Float32 calculateInterObjectAngle(float targetX, float targetY);
44      Float32 calculateDistance(float targetX, float targetY);
45      Float32 calculateVolume(float targetX, float targetY);
46      CriAuSendLevel calculateSpeakerSendLevel(float targetX, float targetY);
47      void playSound(string soundName, const CriAuSendLevel &sendLevel);
48      int calculateScore(float targetX, float targetY);
49      void showResult(int score);
50      void readNumber(int score);
51
52  public:
53      GamePlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
54      ~GamePlayState();
55      int run();
56
57  };
58  #endif
```

```
1   #include "KeyFlag.h"
2
3   /**
4    * キーの状態を管理するクラス
5    *
6    * @date 2006/11/05
7    * @author hashiyaman
8    */
9
10  /*
11   * キーの状態を更新する
12   */
13  void KeyFlag::update(Uint8 *keyState)
14  {
15      if(keyState[id]) // キーが押されてたら
16      {
17          down = true;
18      }
19      else if( !keyState[id] && down) // キーが離された瞬間だったら
20      {
21          released = true;
22          down = false;
23      }
24      else if( !keyState[id] && !down ) // キーが押されてなかったら
25      {
26          released = false;
27      }
28  }
29
30  /*
31   * キーが押されているかを返す
32   */
33  bool KeyFlag::isDown()
34  {
35      return down;
36  }
37
38  /*
39   * キーが離されているかを返す
40   */
41  bool KeyFlag::isReleased()
42  {
43      return released;
44  }
```

KeyFlag.h (1/1)

```
1       #ifndef _KEYFLAG_H_
2       #define _KEYFLAG_H_
3
4       #include <SDL.h>
5
6       class KeyFlag
7       {
8           Uint8 id;
9           bool down;
10          bool released;
11      public:
12          bool isDown();
13          bool isReleased();
14          void update(Uint8 *keyState);
15          KeyFlag(Uint8 keyId): id(keyId), down(false), released(false){};
16      };
17
18      #endif
```

```cpp
#include "KeyState.h"

/**
 * キーの状態を監視するクラス
 *
 * @date 2006/11/05
 * @author hashiyaman
 */

/*
 * コンストラクタ
 */
KeyState::KeyState(KeyTable keyTable)
{
    KeyTableItr itr;

    for(itr = keyTable.begin(); itr != keyTable.end(); itr++)
    {
        keys[*itr] = new KeyFlag(*itr);
    }

}

/*
 * キーの状態を監視する
 */
void KeyState::update()
{
    Uint8* keyState = SDL_GetKeyState(NULL);
    KeyFlagTableItr itr;
    for( itr = keys.begin(); itr != keys.end(); itr++ )
    {
        (*itr).second->update(keyState);
    }
}

/*
 * デストラクタ
 */
KeyState::~KeyState()
{
    KeyFlagTableItr itr;
    for( itr = keys.begin(); itr != keys.end(); itr++ )
    {
        delete (*itr).second;
    }

    keys.clear();
}

/*
 * キーが押されているかを返す
 */
bool KeyState::down(Uint8 key)
{
    return keys[key]->isDown();
}

/*
 * キーが離されているかを返す
 */
bool KeyState::released(Uint8 key)
{
```



```cpp
    return keys[key]->isReleased();
}
```

```
1   #ifndef _KEYSTATE_H_
2   #define _KEYSTATE_H_
3
4   #include <SDL.h>
5   #include <vector>
6   #include <map>
7   #include <algorithm>
8   #include <boost/bind.hpp>
9
10  #include "KeyFlag.h"
11
12  namespace
13  {
14      typedef std::vector<Uint8> KeyTable;
15      typedef KeyTable::iterator KeyTableItr;
16
17      typedef std::map<Uint8,KeyFlag*> KeyFlagTable;
18      typedef KeyFlagTable::iterator KeyFlagTableItr;
19
20  }
21
22  class KeyState
23  {
24      KeyFlagTable keys;
25  public:
26      KeyState(KeyTable keyTable);
27      ~KeyState();
28      void update();
29      bool down(Uint8 key);
30      bool released(Uint8 key);
31  };
32
33  #endif
```

```
1   #include <SDL.h>
2   #include <SDL_ttf.h>
3   #include "Game.h"
4
5   #pragma comment(lib, "SDL.lib")
6   #pragma comment(lib, "SDLmain.lib")
7   #pragma comment(lib, "SDL_ttf.lib")
8   #pragma comment(lib, "cri_audio_pc.lib")
9   #pragma comment(lib, "cri_base_pc.lib")
10  #pragma comment(lib, "dsound.lib")
11  #pragma comment(lib, "winmm.lib")
12
13  int main(int argc, char* argv[])
14  {
15      // 初期化
16      if( SDL_Init( SDL_INIT_AUDIO|SDL_INIT_VIDEO ) == -1
17       || TTF_Init() == -1
18       )
19      {
20          fprintf(stderr,"初期化に失敗¥n");
21          exit(1);
22      }
23
24      // キャプションの設定
25      SDL_WM_SetCaption( "SoundShoot", NULL );
26
27      // マウスカーソルを消す
28      SDL_ShowCursor(SDL_DISABLE);
29
30      // ウィンドウの初期化
31      SDL_Surface *mainScreen = SDL_SetVideoMode(640, 480, 32, SDL_SWSURFACE);
32
33      // メインループ
34      Game *game = new Game(mainScreen);
35      game->run();
36
37      delete game;
38
39
40      // 終了処理
41      SDL_FreeSurface( mainScreen );
42      TTF_Quit();
43      SDL_Quit();
44      return 0;
45  }
```

```
#include "OnMapObject.h"

const double OnMapObject::M_PI = 3.14159265358979323846264338327 9;

/*
 * コンストラクタ
 */
OnMapObject::OnMapObject(float x, float y, SoundContainer *soundContainer, string soundListPath)
{
    this->soundPlayer = new SoundPlayer(soundContainer,soundListPath);
    this->initObjectStatus(x,y);
    initializeStatus = objectStatus;
}

/*
 * 初期化する
 */
void OnMapObject::initObjectStatus(float x, float y){
    this->objectStatus.x = x;
    this->objectStatus.y = y;
    this->objectStatus.state = ALIVE;
    this->objectStatus.width = 10;
    this->objectStatus.height = 10;
}

/*
 * オブジェクト間の角度を計算する
 */
Float32 OnMapObject::calculateInterObjectAngle(const ObjectStatus *subject, const ObjectStatus *target)
{
    // オブジェクト間の角度を計算する
    float radian = atan2( subject->y - target->y, subject->x - target->x);

    // 弧度法から度数法へ変換
    // CRIAudioでは正面が0度なので、画面と音の整合性を取るため角度を足す
    int angle = static_cast<int>( ( radian / M_PI * 180 ) + 90 );
    angle = angle % 360;

    return static_cast<Float32>(-angle);

}

/*
 * スピーカーのセンドレベルを計算する
 */
CriAuSendLevel OnMapObject::calculateSpeakerSendLevel(const ObjectStatus *subject, const ObjectStatus
target)
{
    // オブジェクト間の角度を計算する
    Float32 angle = this->calculateInterObjectAngle(subject, target);

    // 角度からセンドレベルを計算する
    Float32 left;
    Float32 right;
    Float32 leftSurround;
    Float32 rightSurround;
    Float32 center;
    CriAuUty::CalcSendLevel5Speakers(angle, &left, &right, &leftSurround, &rightSurround, &center);

    // センドレベルを設定する
    CriAuSendLevel sendLevel;
    sendLevel.SetLeft(left);
    sendLevel.SetRight(right);
```



```
    sendLevel.SetLeftSurround(leftSurround);
    sendLevel.SetRightSurround(rightSurround);
    sendLevel.SetCenter(center);

    return sendLevel;
}

/*
 * オブジェクトの音を鳴らす
 */
void OnMapObject::playSound(string soundName, const CriAuSendLevel &sendLevel)
{
      soundPlayer->setSendLevel(soundName, sendLevel);
    soundPlayer->play(soundName);
}

/*
 * デストラクタ
 */
OnMapObject::~OnMapObject()
{
    delete soundPlayer;
}
```

```
#ifndef _ONMAPOBJECT_H_
#define _ONMAPOBJECT_H_

#include <SDL.h>
#include <windows.h>
#include "SoundPlayer.h"
#include "cri_audio.h"

enum ObjectCondition
{
    ALIVE,
    DEAD
};

struct ObjectStatus
{
    ObjectStatus(){};
    ObjectStatus(float x, float y, int state,    int width, int height):
    x(x), y(y), state(state), width(width), height(height){};
    float x;
    float y;
    int state;
    int width;
    int height;
};

class OnMapObject
{
protected:
    static const double M_PI;

    ObjectStatus objectStatus;
    ObjectStatus initializeStatus;
    ObjectStatus *playerStatus;

    SoundPlayer* soundPlayer;

    // subjectから見たtargetの位置に応じて各スピーカーへのセンドレベルを計算する
    CriAuSendLevel calculateSpeakerSendLevel(const ObjectStatus *subject, const ObjectStatus *target);

    // subjectとtarget間の角度を計算する
    Float32 calculateInterObjectAngle(const ObjectStatus *subject, const ObjectStatus *target);
    void playSound(string soundName, const CriAuSendLevel &sendLevel);

    // 距離に応じたボリュームを計算する
    virtual Float32 calculateVolume(const ObjectStatus *subject, const ObjectStatus *target) = 0;

    //初期化する
    void initObjectStatus(float x, float y);

public:

    void setTarget(ObjectStatus *target){playerStatus = target;};
    void reset(){ objectStatus = initializeStatus; };
    ObjectStatus* getObjectStatus(){return &objectStatus;};

    virtual void move(Uint8 *keys) = 0;
    virtual void resolveCollision() = 0;
    virtual void draw(SDL_Surface *targetScreen) = 0;
    OnMapObject(float x, float y, SoundContainer *soundContainer, string soundListPath, int *time);
    virtual ~OnMapObject();
};
```



```
#endif
```

SDLCommonFunctions.h (1/1)

```
#ifndef _SDLCOMMONFUNCTIONS_H_
#define _SDLCOMMONFUNCTIONS_H_
#include <SDL.h>

namespace
{
    bool PollEvent()
    {
        SDL_Event ev;
        while(SDL_PollEvent(&ev))
        {
            switch(ev.type)
            {
            case SDL_QUIT:// ウィンドウの×ボタンが押された時など
                return false;
                break;
            }

        }
        return true;
    }

    void ClearScreen(SDL_Surface *target)
    {
        // サーフェスを黒で初期化
        SDL_Rect dest;
        dest.x = 0;
        dest.y = 0;
        dest.w = 640;
        dest.h = 480;
        Uint32 color= 0x00000000;
        SDL_FillRect( target, &dest, color );
    }

}
#endif
```

```
#include "SoundContainer.h"

SoundContainer::SoundContainer(string cuePath)
{
    CriError error = CRIERR_OK; // エラー検出用のオブジェクト

    //音声出力の初期化
    soundOut =  CriSmpSoundOutput::Create();
    heap = criHeap_Create(buf, sizeof(buf));
    soundRenderer = CriSoundRendererBasic::Create(heap, error);
    soundOut->SetNotifyCallback( soundOutCallBack, static_cast<void *>(soundRenderer) );
    soundOut->Start();

    // キューの読み込み
    Uint8 *csbdata;
    unsigned long csbsize;
    csbdata = this->loadCue(cuePath, &csbsize);
    cueSheet = CriAuCueSheet::Create(heap, error);
    cueSheet->LoadCueSheetBinaryFileFromMemory("Tutorial", csbdata, csbsize, error);
    free(csbdata);

    // 音声オブジェクトの生成
    audioObject = CriAuObj::Create(heap, soundRenderer, "Tutorial", error);
    audioObject->AttachCueSheet(cueSheet, error);

    // エラー処理
    if( error != CRIERR_OK )
        exit(1);
}

/*
 * 良くわからないけど必要な処理
 */
unsigned long SoundContainer::soundOutCallBack(void *obj, unsigned long nch, Float32 *sample[], unsigned lo
g nsmpl)
{
    CriSoundRendererBasic* sndrndr=(CriSoundRendererBasic*)obj;
    CriError err;

    //    Getting PCM Data from CRI Sound Renderer
    //    nsmpl has to be 128*N samples. (N=1,2,3...)
    sndrndr->GetData(nch, nsmpl, sample, err);

    return nsmpl;
}

/*
 * キューファイルをロードする
 */
Uint8* SoundContainer::loadCue(string path, unsigned long *idtsize)
{
    FILE *fp;
    signed long size;
    Uint8 *idt;

    fp = fopen(path.c_str(), "rb");
    fseek(fp, 0, SEEK_END);
    size = ftell(fp);
    idt = (Uint8*)calloc(size, 1);
    fseek(fp, 0, SEEK_SET);
    fread(idt, size, 1, fp);
    fclose(fp);
```



```
    *idtsize = (unsigned long)size;
    return idt;
}

/*
 * 音をロードする(つまり音声プレーヤーを生成する)
 */
CriAuPlayer* SoundContainer::loadSound(string soundName)
{
    CriError error = CRIERR_OK;
    CriAuPlayer* player = CriAuPlayer::Create(audioObject, error);
    player->SetCue(soundName.c_str(), error);

    return player;
}

SoundContainer::~SoundContainer()
{
    CriError error = CRIERR_OK;

    audioObject->Destroy(error);
    cueSheet->Destroy(error);
    soundOut->SetNotifyCallback(NULL, NULL);
    soundRenderer->Destroy(error);

    //    Print Heap Status
    //criHeap_DebugPrintBlockInformationAll(heap);
    criHeap_Destroy(heap);
}
```

```
1   #ifndef _SOUNDCONTAINER_H_
2   #define _SOUNDCONTAINER_H_
3
4   #include <string>
5   #include <iostream>
6   #include "cri_audio.h"
7   #include "cri_xpt.h"
8   #include "CriSmpSoundOutput.h"
9
10  using std::string;
11
12  class SoundContainer
13  {
14      CriHeap heap;
15      CriSoundRendererBasic* soundRenderer;
16      CriAuObj* audioObject;
17      Uint8 buf[10*1024*1024];
18      CriSmpSoundOutput *soundOut;
19      CriAuCueSheet* cueSheet;
20
21      Uint8* loadCue(string path, unsigned long *idtsize);
22      void createCueSheet();
23      static unsigned long soundOutCallBack(void *obj, unsigned long nch, Float32 *sample[], unsigned long nsmp
    );
24
25  public:
26      ~SoundContainer();
27      SoundContainer(string cuePath);
28      CriAuPlayer* loadSound(string soundName);
29
30  };
31
32  #endif
```

```cpp
#include "SoundPlayer.h"

SoundPlayer::SoundPlayer(SoundContainer *soundContainer, string soundListPath)
{
    // 音声を取得する
    soundShelf = this->loadSound(soundContainer,soundListPath);
}

SoundPlayer::~SoundPlayer()
{
    soundShelf.clear();
}


SoundShelf SoundPlayer::loadSound(SoundContainer *soundContainer, string soundListPath)
{
    // ファイルを開く
    std::ifstream list(soundListPath.c_str());

    // 役割ごとの音声の名前を連想配列に読み込む
    SoundShelf soundList;
    string line;
    while( getline(list,line) )
    {
        // 取得した文の空白位置を探す
        string::size_type splitIndex = line.find(" ",0);

        // 役割と名前が空白で区切られてなかったらスキップする
        if( splitIndex == string::npos )
            continue;

        string soundJobName = line.substr(0,splitIndex);
        string soundName = line.substr(splitIndex+1);
        soundList[soundJobName] = soundContainer->loadSound(soundName);
    }
    list.close();

    return soundList;
}

int SoundPlayer::getStatus(string soundJobName)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        return soundShelf[soundJobName]->GetStatus(error);
    }
    else
    {
        return -1;
    }
}

void SoundPlayer::play(string soundJobName)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->Play(error);
    }
}

void SoundPlayer::loopPlay(string soundJobName)
```



```cpp
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        int soundStatus = soundShelf[soundJobName]->GetStatus(error);
        if( soundStatus == CriAuPlayer::STATUS_STOP || soundStatus == CriAuPlayer::STATUS_PLAYEND )
        {
            soundShelf[soundJobName]->Play(error);
        }
    }
}

void SoundPlayer::stop(string soundJobName)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->Stop(error);
    }
}

void SoundPlayer::stopAll()
{
    CriError error = CRIERR_OK;
    SoundShelfIterator itr;
    for( itr = soundShelf.begin(); itr != soundShelf.end(); itr++ )
    {
        (*itr).second->Stop(error);
    }
}

void SoundPlayer::stopAllExcept(string excludeSoundName)
{
    CriError error = CRIERR_OK;
    SoundShelfIterator itr;

    // 指定された音以外を止める
    for( itr = soundShelf.begin(); itr != soundShelf.end(); itr++ )
    {
        if( (*itr).first != excludeSoundName )
        {
            (*itr).second->Stop(error);
        }
    }
}


void SoundPlayer::setVolume(string soundJobName, float volume)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->SetVolume(volume, error);
        soundShelf[soundJobName]->Update(error);
    }
}

void SoundPlayer::setPitch(string soundJobName, float pitch)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->SetPitch(pitch,error);
```

```
127             soundShelf[soundJobName]->Update(error);
128         }
129     }
130
131     void SoundPlayer::setSendLevel(string soundJobName, const CriAuSendLevel &sendLevel)
132     {
133         if( soundShelf.find(soundJobName) != soundShelf.end() )
134         {
135             CriError error = CRIERR_OK;
136             soundShelf[soundJobName]->SetDrySendLevel(sendLevel, error);
137             soundShelf[soundJobName]->Update(error);
138         }
139     }
140
141     void SoundPlayer::setReverb(string soundJobName, float reverb)
142     {
143         if( soundShelf.find(soundJobName) != soundShelf.end() )
144         {
145             CriError error = CRIERR_OK;
146             soundShelf[soundJobName]->SetWetSendLevel(CriAuSendLevel::WET_0, reverb, error);
147             soundShelf[soundJobName]->Update(error);
148         }
149     }
150
151     void SoundPlayer::setCutOffFrequency(string soundJobName, float lowerFrequency, float upperFrequency)
152     {
153         if( soundShelf.find(soundJobName) != soundShelf.end() )
154         {
155             CriError error = CRIERR_OK;
156             soundShelf[soundJobName]->SetFilterCutoffFrequency(lowerFrequency, upperFrequency, error);
157             soundShelf[soundJobName]->Update(error);
158         }
159     }
```

```
#ifndef _SOUNDPLAYER_H_
#define _SOUNDPLAYER_H_

#include "SoundContainer.h"
#include <string>
#include <fstream>
#include <map>
using std::string;

namespace
{
    typedef std::map<string, CriAuPlayer*> SoundShelf;
    typedef SoundShelf::iterator SoundShelfIterator;
}
/*
 * 音声プレイヤー
 */
class SoundPlayer
{
    SoundShelf soundShelf;    // 音声を管理する連想配列
    SoundShelf loadSound(SoundContainer *soundContainer, string soundListPath);

public:
    SoundPlayer(SoundContainer *soundContainer, string soundListPath);
    ~SoundPlayer();
    int getStatus(string soundName);
    void play(string soundName);
    void loopPlay(string soundJobName);
    void stop(string soundName);
    void stopAll();
    void stopAllExcept(string excludeSoundName);
    void setVolume(string soundName, float volume);
    void setPitch(string soundName, float pitch);
    void setSendLevel(string soundName, const CriAuSendLevel &sendLevel);
    void setReverb(string soundName, float reverb);
    void setCutOffFrequency(string soundName, float lowerFrequency, float upperFrequency);
};

#endif
```

```
1       #ifndef _STATEMESSAGES_H_
2       #define _STATEMESSAGES_H_
3
4       namespace
5       {
6           // 状態間で行き交うメッセージ
7           enum StateMessage
8           {
9               QUIT_GAME,    // escが押された・ウィンドウが閉じられた時
10              START_GAME, // タイトルでゲームを開始した
11              GO_TITLE,    // 結果表示を終えた
12          };
13      }
14      #endif
```

```
/**
 * タイトルを表現するクラス
 *
 * @date 2006/11/05
 * @author hashiyaman
 */

#include "TitleState.h"

const Uint8 TitleState::PLAY_GAME = SDLK_RETURN;
const Uint8 TitleState::QUIT = SDLK_ESCAPE;

/*
 * コンストラクタ
 */
TitleState::TitleState(SoundContainer *soundContainer, SDL_Surface *mainScreen)
{
    this->soundPlayer = new SoundPlayer(soundContainer,"TitleStateSound.txt");
    this->mainScreen = mainScreen;

    // 状態の監視をしたいキーを列挙して渡す
    KeyTable useKeyTable;
    useKeyTable.push_back(PLAY_GAME);
    useKeyTable.push_back(QUIT);
    keyState = new KeyState(useKeyTable);
}

/*
 * デストラクタ
 */
TitleState::~TitleState()
{
    delete soundPlayer;
    delete keyState;
}

/*
 * タイトルでの処理を行う
 */
int TitleState::run()
{
    return this->runFrame();
}

// 1フレーム内の処理を行う
int TitleState::runFrame()
{
    // テキストの色とフォントと内容を設定する
    SDL_Color textColor = {255,255,255};
    TTF_Font *font = TTF_OpenFont( "Headache.ttf", 28 );
    SDL_Surface *stateText = TTF_RenderText_Blended( font, "IN TITLE", textColor );

    int gameState;
    soundPlayer->loopPlay("TITLE");

    while(true)
    {
        SDL_PumpEvents(); // イベント状態を更新
        SDL_Delay(50); // CPU使用率が100%になるのを防ぐ

        keyState->update(); // キー入力を取得

        // ゲームが終了された場合
```



```
        if( !PollEvent() || keyState->down(QUIT) )
        {
            gameState = QUIT_GAME;
            break;
        }

        CriError error = CRIERR_OK;

        // タイトルを再生し終えてから，開始方法を再生する
        if ( soundPlayer->getStatus("TITLE") == CriAuPlayer::STATUS_PLAYEND )
        {
            soundPlayer->loopPlay("HOW_TO_START");
        }

        CriAuObj::ExecuteMain(error); // 音声の状態を更新する

        ClearScreen(mainScreen); // 画面をクリアする
        this->draw(10,10,stateText,mainScreen);    // 描画を行う

        SDL_Flip(mainScreen); // 画面の状態を更新する

        if( keyState->released(PLAY_GAME) )
        {
            gameState = START_GAME; // ゲームを開始する
            soundPlayer->stopAll();
            break;
        }
    }

    // 音の停止
    CriError error = CRIERR_OK;
    soundPlayer->stopAll();
    CriAuObj::ExecuteMain(error);

    // 後処理
    TTF_CloseFont(font);
    SDL_FreeSurface(stateText);

    return gameState;
}

/*
 * 描画する
 */
void TitleState::draw(int x, int y, SDL_Surface *source, SDL_Surface *destination )
{
    SDL_Rect position;
    position.x = x;
    position.y = y;
    SDL_BlitSurface(source, NULL, destination, &position);
}
```

```
1   #ifndef _TITLESTATE_H_
2   #define _TITLESTATE_H_
3
4   #include <SDL.h>
5   #include <SDL_ttf.h>
6   #include <string>
7   #include <list>
8   #include "KeyState.h"
9   #include "StateMessages.h"
10  #include "SoundPlayer.h"
11  #include "SDLCommonFunctions.h"
12  using std::string;
13
14  /*
15   * ゲーム部分を管理するクラス
16   */
17  class TitleState
18  {
19      SDL_Surface *mainScreen;
20      SoundPlayer *soundPlayer;
21      KeyState *keyState;
22
23      static const Uint8 PLAY_GAME;
24      static const Uint8 QUIT;
25
26      int runFrame();    // 1フレーム内の処理を行う
27
28      void draw(int x, int y, SDL_Surface *source, SDL_Surface *destination );
29  public:
30      TitleState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
31      ~TitleState();
32      int run();    // ゲームを実行する
33
34
35
36  };
37
38  #endif
```

2007/01/31
さうんど おんりぃ
最終報告書

ソースコード：聖徳太子ゲーム改（C++）

```
1   /**
2    * ゲーム本体を実行するクラス
3    *
4    * @date 2006/11/05
5    * @author ikumin
6    */
7
8   #include "Game.h"
9
10  /*
11   * コンストラクタ
12   */
13  Game::Game(SDL_Surface *mainScreen)
14  {
15      this->mainScreen = mainScreen;
16      soundContainer = new SoundContainer("CommandInput.csb");
17      titleState = new TitleState(soundContainer,mainScreen);
18      gamePlayState = new GamePlayState(soundContainer,mainScreen);
19  }
20
21
22  /*
23   * ゲームを実行する
24   */
25  void Game::run()
26  {
27      while(true)
28      {
29          // タイトルを表示する
30          int titleMessage = titleState->run();
31          switch( titleMessage )
32          {
33          case START_GAME:
34              {
35                  int GameMessage = gamePlayState->run();
36                  if( GameMessage == QUIT_GAME )
37                  {
38                      return;
39                  }
40              }
41              break;
42          case QUIT_GAME:
43              return;
44          }
45      }
46  }
47
48  /*
49   * デストラクタ
50   */
51  Game::~Game()
52  {
53      delete soundContainer;
54      delete titleState;
55      delete gamePlayState;
56  }
57
```

```
1   #ifndef _GAME_H_
2   #define _GAME_H_
3
4   #include <string>
5   #include "SoundContainer.h"
6   #include "TitleState.h"
7   #include "GamePlayState.h"
8
9   class Game
10  {
11      SoundContainer *soundContainer;
12      SDL_Surface *mainScreen;
13      TitleState *titleState;
14      GamePlayState *gamePlayState;
15
16  public:
17      Game(SDL_Surface *mainScreen);
18      ~Game();
19
20      void run();
21  };
22
23  #endif
```

```
/**
 * ゲーム中を表現するクラス
 */

#include "GamePlayState.h"

const Uint8 GamePlayState::SHOOT = SDLK_RETURN;
const Uint8 GamePlayState::QUIT = SDLK_ESCAPE;
const Uint8 GamePlayState::SKIP = SDLK_SPACE;
const Uint8 GamePlayState::DOWN = SDLK_x;
const Uint8 GamePlayState::UP = SDLK_e;
const Uint8 GamePlayState::RIGHT = SDLK_d;
const Uint8 GamePlayState::LEFT = SDLK_s;

const double GamePlayState::M_PI = 3.14159265358979323846264338327 9;
const float GamePlayState::PLAYER_X = 320;
const float GamePlayState::PLAYER_Y = 240;

/*
 * コンストラクタ
 */
GamePlayState::GamePlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen)
{
    this->soundPlayer = new SoundPlayer(soundContainer,"GamePlayStateSound.txt");
    this->soundContainer = soundContainer;
    this->mainScreen = mainScreen;

    // 状態の監視をしたいキーを列挙して渡す
    KeyTable useKeyTable;
    useKeyTable.push_back(QUIT);
    useKeyTable.push_back(SHOOT);
    useKeyTable.push_back(SKIP);

    useKeyTable.push_back(DOWN);
    useKeyTable.push_back(UP);
    useKeyTable.push_back(RIGHT);
    useKeyTable.push_back(LEFT);
    keyState = new KeyState(useKeyTable);
}

/*
 * デストラクタ
 */
GamePlayState::~GamePlayState()
{
    delete keyState;
    delete soundPlayer;
}

/*
 * ゲーム中の処理を行う
 */
int GamePlayState::run()
{
    return this->runFrame();
}

/*
 * ーフレーム内の処理を行う
 */
int GamePlayState::runFrame()
{
    // フォントと色と内容の設定を行う
```



```
    SDL_Color textColor = {255,255,255};
    TTF_Font *font = TTF_OpenFont( "Headache.ttf", 28 );
    SDL_Surface *text = TTF_RenderText_Blended( font, "GAME PLAYING", textColor );

    int gameState;
    bool isLeftPushed = false;
    bool isUpPushed = false;
    isRight = true;
    isCollect = true;
    soundState = false;
    soundPlayer->loopPlay("START_GAME");

    int score = -1;
    float targetX = 0;
    float targetY = 230;

    while(true)
    {
        CriError error = CRIERR_OK;

        // 入力を取得
        keyState->update();
        Uint8 *keys = SDL_GetKeyState(NULL);

        // ゲームが終了された場合
        if( !PollEvent() || keyState->down(QUIT) )
        {
            gameState = QUIT_GAME;
            break;
        }
        if (keyState->down(SHOOT) && !getCommandSoundStatus()) {
            soundState = true;
        }

        // 説明をスキップする
        if( soundPlayer->getStatus("START_GAME") == CriAuPlayer::STATUS_PLAYING
            && keyState->down(SKIP) )
        {
            soundPlayer->stop("START_GAME");
        }

        // ゲームの説明を終えた後に，ゲーム中の音声の再生を開始する
        if( (targetX < 640 && targetY < 480)
            && soundPlayer->getStatus("START_GAME") == CriAuPlayer::STATUS_PLAYEND
            && soundState)
        {
            anounceCommand();
        }
        if (isLeftPushed && isUpPushed)
        {
            string resultText = "Correct!!";
            SDL_Surface *text = TTF_RenderText_Blended( font, resultText.c_str(), textColor );
            this->draw(10, 30, text, mainScreen); // 結果の描画

            showResult(score);
            if ( keyState->down(SKIP) )
            {
                gameState = GO_TITLE;
                break;
            }
        }

        CriAuObj::ExecuteMain(error); // 音声の状態を更新する
```

```
127
128         // 音を狙って，得点を計算する(1ゲームにつき1回のみ)
129         if( keyState->down(LEFT)
130             && soundPlayer->getStatus("START_GAME") == CriAuPlayer::STATUS_PLAYEND)
131         {
132             isLeftPushed = true;
133             //score = calculateScore(targetX, targetY);
134         }
135         if( keyState->down(UP)
136             && soundPlayer->getStatus("START_GAME") == CriAuPlayer::STATUS_PLAYEND)
137         {
138             isUpPushed = true;
139             //score = calculateScore(targetX, targetY);
140         }
141
142         ClearScreen(mainScreen); // 画面をクリアする
143
144         this->draw(10,10,text,mainScreen); // 文字の描画
145
146
147         if( score >= 0 ) // スコアを表示する
148         {
149             string scoreText = "SCORE:" + boost::lexical_cast<string>(score);
150             SDL_Surface *text = TTF_RenderText_Blended( font, scoreText.c_str(), textColor );
151             this->draw(10,30,text,mainScreen); // スコアの描画
152         }
153
154         SDL_Flip(mainScreen);
155
156         SDL_Delay(50); // CPU使用率が100%になるのを防ぐ
157     }
158
159     // 音の停止
160     CriError error = CRIERR_OK;
161     soundPlayer->stopAll();
162     CriAuObj::ExecuteMain(error);
163
164     TTF_CloseFont(font);
165     SDL_FreeSurface(text);
166
167     return gameState;
168 }
169
170 /*
171  * 命令をアナウンスする
172  */
173 void GamePlayState::anounceCommand() {
174     soundPlayer->play("push_left");
175     soundPlayer->play("push_up");
176     soundState = false;
177 }
178
179 bool GamePlayState::getCommandSoundStatus() {
180     if (soundPlayer->getStatus("push_down") == CriAuPlayer::STATUS_PLAYING
181         || soundPlayer->getStatus("push_left") == CriAuPlayer::STATUS_PLAYING
182         || soundPlayer->getStatus("push_up") == CriAuPlayer::STATUS_PLAYING
183         || soundPlayer->getStatus("push_right") == CriAuPlayer::STATUS_PLAYING) {
184             return true;
185         } else {
186             return false;
187         }
188 }
189
```



```
190 /*
191  * 音を動かす
192  */
193 void GamePlayState::moveSound(float *targetX, float *targetY)
194 {
195     *targetX += 2;
196     *targetY += 0;
197
198     // 距離に応じてボリュームを調整する
199     Float32 volume = this->calculateVolume( *targetX, *targetY );
200     soundPlayer->setVolume("MOVE1", volume);
201
202     CriAuSendLevel sendLevel = this->calculateSpeakerSendLevel( *targetX, *targetY );
203     playSound("MOVE1", sendLevel);
204 }
205
206 /*
207  * ボリュームを計算する
208  */
209 Float32 GamePlayState::calculateVolume(float targetX, float targetY)
210 {
211     float distance = calculateDistance(targetX, targetY);
212
213     // 距離に応じてボリュームを調整する
214     Float32 volume;
215     if( distance == 0 )
216     {
217         volume = 1.0f;
218     }
219     else
220     {
221         //volume = 1.0f / sqrtf(distance) * 2.0f;
222         volume = sqrtf(distance) / -10.0f + 2.0f;
223     }
224
225     if( volume < 0.0f)
226         volume = 0.0f;
227
228     return volume;
229 }
230
231 /*
232  * 距離を計算する
233  */
234 Float32 GamePlayState::calculateDistance(float targetX, float targetY)
235 {
236     // 距離を計算する
237     float xDistance = targetX - PLAYER_X;
238     float yDistance = targetY - PLAYER_Y;
239     float distance = xDistance * xDistance + yDistance * yDistance;
240
241     return sqrtf(distance);
242 }
243
244 /*
245  * オブジェクト間の角度を計算する
246  */
247 Float32 GamePlayState::calculateInterObjectAngle(float targetX, float targetY)
248 {
249     // オブジェクト間の角度を計算する
250     float radian = atan2( PLAYER_Y - targetY, PLAYER_X - targetX);
251
252     // 弧度法から度数法へ変換
```

```
253         // CRIAudioでは正面が0度なので、画面と音の整合性を取るため角度を足す
254         int angle = static_cast<int>( ( radian / M_PI * 180 ) + 90 );
255         angle = angle % 360;
256 
257         return static_cast<Float32>(-angle);
258     }
259 
260     /*
261      * スピーカーのセンドレベルを計算する
262      */
263     CriAuSendLevel GamePlayState::calculateSpeakerSendLevel(float targetX, float targetY)
264     {
265         // オブジェクト間の角度を計算する
266         Float32 angle = this->calculateInterObjectAngle(targetX, targetY);
267 
268         // 角度からセンドレベルを計算する
269         Float32 left;
270         Float32 right;
271         Float32 leftSurround;
272         Float32 rightSurround;
273         Float32 center;
274         CriAuUty::CalcSendLevel5Speakers(angle, &left, &right, &leftSurround, &rightSurround, &center);
275 
276         // センドレベルを設定する
277         CriAuSendLevel sendLevel;
278         sendLevel.SetLeft(left);
279         sendLevel.SetRight(right);
280         sendLevel.SetLeftSurround(leftSurround);
281         sendLevel.SetRightSurround(rightSurround);
282         sendLevel.SetCenter(center);
283 
284         return sendLevel;
285     }
286 
287     /*
288      * オブジェクトの音を鳴らす
289      */
290     void GamePlayState::playSound(string soundName, const CriAuSendLevel &sendLevel)
291     {
292           soundPlayer->setSendLevel(soundName, sendLevel);
293         soundPlayer->loopPlay(soundName);
294     }
295 
296     /*
297      * 得点を計算する
298      */
299     int GamePlayState::calculateScore(float targetX, float targetY)
300     {
301         float distance = calculateDistance(targetX, targetY);
302         int score = 100 - distance; // 得点
303 
304         if( score < 0)
305         {
306             score = 0;
307         }
308 
309         return score;
310     }
311 
312     /*
313      * 描画を行う
314      */
315     void GamePlayState::draw(int x, int y, SDL_Surface *source, SDL_Surface *destination )
```



```
316     {
317         SDL_Rect position;
318         position.x = x;
319         position.y = y;
320         SDL_BlitSurface(source, NULL, destination, &position);
321     }
322 
323     /*
324      * 結果を知らせる
325      */
326     void GamePlayState::showResult(int score)
327     {
328         if (isRight) {
329             soundPlayer->play("right");
330             isRight = false;
331         }
332         if (soundPlayer->getStatus("right") == CriAuPlayer::STATUS_PLAYEND
333             && isCollect) {
334             soundPlayer->play("SEIKAI");
335             isCollect = false;
336         }
337         if(soundPlayer->getStatus("SEIKAI") == CriAuPlayer::STATUS_PLAYEND
338             && (soundPlayer->getStatus("END_GAME") == CriAuPlayer::STATUS_PLAYEND
339             || soundPlayer->getStatus("END_GAME") == CriAuPlayer::STATUS_STOP) )
340         {
341             soundPlayer->play("END_GAME");
342             //readNumber(score);
343         }
344     }
345 
346     /*
347      * 得点を読み上げる（未完成）
348      */
349     void GamePlayState::readNumber(int score)
350     {
351         string scoreString = boost::lexical_cast<string>(score);
352         static string currentDigit = scoreString.substr(0,1); // 現在読み上げているケタ
353 
354         if( soundPlayer->getStatus("SCORE") == CriAuPlayer::STATUS_PLAYEND
355             || soundPlayer->getStatus("SCORE") == CriAuPlayer::STATUS_STOP )
356         {
357             for( int i = 0; i < scoreString.length(); i++ )
358             {
359                 currentDigit = scoreString.substr(i,1);
360                 soundPlayer->play(currentDigit);
361             }
362             soundPlayer->play("SCORE");
363 
364         }
365     }
```

GamePlayState.h (1/2)

```
#ifndef _GAMEPLAYSTATE_H_
#define _GAMEPLAYSTATE_H_

#include "SoundPlayer.h"
#include "KeyState.h"
#include "StateMessages.h"

#include <SDL.h>
#include <SDL_ttf.h>
#include <string>
#include <list>
#include <boost/bind.hpp>
#include <boost/lexical_cast.hpp>
#include <algorithm>
#include <cstdlib>

#include "SDLCommonFunctions.h"
using std::string;

enum ObjectCondition
{
    ALIVE,
    DEAD
};

class GamePlayState
{
    SoundPlayer *soundPlayer;
    SoundContainer *soundContainer;
    SDL_Surface *mainScreen;
    bool soundState;
    bool isRight;
    bool isCollect;

    KeyState *keyState;
    static const Uint8 SHOOT;
    static const Uint8 QUIT;
    static const Uint8 SKIP;
    static const Uint8 DOWN;
    static const Uint8 UP;
    static const Uint8 RIGHT;
    static const Uint8 LEFT;
    static const double M_PI;
    static const float PLAYER_X;
    static const float PLAYER_Y;

    int runFrame();    // 1フレーム内の処理を行う
    void draw(int x, int y, SDL_Surface *source, SDL_Surface *destination );
    void moveSound(float *targetX, float *targetY);
    void anounceCommand();
    bool getCommandSoundStatus();
    Float32 calculateInterObjectAngle(float targetX, float targetY);
    Float32 calculateDistance(float targetX, float targetY);
    Float32 calculateVolume(float targetX, float targetY);
    CriAuSendLevel calculateSpeakerSendLevel(float targetX, float targetY);
    void playSound(string soundName, const CriAuSendLevel &sendLevel);
    int calculateScore(float targetX, float targetY);
    void showResult(int score);
    void readNumber(int score);

public:
    GamePlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
    ~GamePlayState();
```


GamePlayState.h (2/2)

```
    int run();
};
#endif
```

```
1   #include "KeyFlag.h"
2
3   /**
4    * キーの状態を管理するクラス
5    *
6    * @date 2006/11/05
7    * @author hashiyaman
8    */
9
10  /*
11   * キーの状態を更新する
12   */
13  void KeyFlag::update(Uint8 *keyState)
14  {
15      if(keyState[id]) // キーが押されてたら
16      {
17          down = true;
18      }
19      else if( !keyState[id] && down) // キーが離された瞬間だったら
20      {
21          released = true;
22          down = false;
23      }
24      else if( !keyState[id] && !down ) // キーが押されてなかったら
25      {
26          released = false;
27      }
28  }
29
30  /*
31   * キーが押されているかを返す
32   */
33  bool KeyFlag::isDown()
34  {
35      return down;
36  }
37
38  /*
39   * キーが離されているかを返す
40   */
41  bool KeyFlag::isReleased()
42  {
43      return released;
44  }
```

KeyFlag.h (1/1)

```
1       #ifndef _KEYFLAG_H_
2       #define _KEYFLAG_H_
3
4       #include <SDL.h>
5
6       class KeyFlag
7       {
8           Uint8 id;
9           bool down;
10          bool released;
11      public:
12          bool isDown();
13          bool isReleased();
14          void update(Uint8 *keyState);
15          KeyFlag(Uint8 keyId): id(keyId), down(false), released(false){};
16      };
17
18      #endif
```

```cpp
#include "KeyState.h"

/**
 * キーの状態を監視するクラス
 *
 * @date 2006/11/05
 * @author hashiyaman
 */

/*
 * コンストラクタ
 */
KeyState::KeyState(KeyTable keyTable)
{
    KeyTableItr itr;

    for(itr = keyTable.begin(); itr != keyTable.end(); itr++)
    {
        keys[*itr] = new KeyFlag(*itr);
    }

}

/*
 * キーの状態を監視する
 */
void KeyState::update()
{
    Uint8* keyState = SDL_GetKeyState(NULL);
    KeyFlagTableItr itr;
    for( itr = keys.begin(); itr != keys.end(); itr++ )
    {
        (*itr).second->update(keyState);
    }
}

/*
 * デストラクタ
 */
KeyState::~KeyState()
{
    KeyFlagTableItr itr;
    for( itr = keys.begin(); itr != keys.end(); itr++ )
    {
        delete (*itr).second;
    }

    keys.clear();
}

/*
 * キーが押されているかを返す
 */
bool KeyState::down(Uint8 key)
{
    return keys[key]->isDown();
}

/*
 * キーが離されているかを返す
 */
bool KeyState::released(Uint8 key)
{
```



```cpp
    return keys[key]->isReleased();
}
```

```
1   #ifndef _KEYSTATE_H_
2   #define _KEYSTATE_H_
3
4   #include <SDL.h>
5   #include <vector>
6   #include <map>
7   #include <algorithm>
8   #include <boost/bind.hpp>
9
10  #include "KeyFlag.h"
11
12  namespace
13  {
14      typedef std::vector<Uint8> KeyTable;
15      typedef KeyTable::iterator KeyTableItr;
16
17      typedef std::map<Uint8,KeyFlag*> KeyFlagTable;
18      typedef KeyFlagTable::iterator KeyFlagTableItr;
19
20  }
21
22  class KeyState
23  {
24      KeyFlagTable keys;
25  public:
26      KeyState(KeyTable keyTable);
27      ~KeyState();
28      void update();
29      bool down(Uint8 key);
30      bool released(Uint8 key);
31  };
32
33  #endif
```

```
#include <SDL.h>
#include <SDL_ttf.h>
#include "Game.h"

#pragma comment(lib, "SDL.lib")
#pragma comment(lib, "SDLmain.lib")
#pragma comment(lib, "SDL_ttf.lib")
#pragma comment(lib, "cri_audio_pc.lib")
#pragma comment(lib, "cri_base_pc.lib")
#pragma comment(lib, "dsound.lib")
#pragma comment(lib, "winmm.lib")

int main(int argc, char* argv[])
{
    // 初期化
    if( SDL_Init( SDL_INIT_AUDIO|SDL_INIT_VIDEO ) == -1
     || TTF_Init() == -1
     )
    {
        fprintf(stderr,"初期化に失敗¥n");
        exit(1);
    }

    // キャプションの設定
    SDL_WM_SetCaption( "CommandInput", NULL );

    // マウスカーソルを消す
    SDL_ShowCursor(SDL_DISABLE);

    // ウィンドウの初期化
    SDL_Surface *mainScreen = SDL_SetVideoMode(640, 480, 32, SDL_SWSURFACE);

    // メインループ
    Game *game = new Game(mainScreen);
    game->run();

    delete game;


    // 終了処理
    SDL_FreeSurface( mainScreen );
    TTF_Quit();
    SDL_Quit();
    return 0;
}
```

```
#include "OnMapObject.h"

const double OnMapObject::M_PI = 3.14159265358979323846264338327 9;

/*
 * コンストラクタ
 */
OnMapObject::OnMapObject(float x, float y, SoundContainer *soundContainer, string soundListPath)
{
    this->soundPlayer = new SoundPlayer(soundContainer,soundListPath);
    this->initObjectStatus(x,y);
    initializeStatus = objectStatus;
}

/*
 * 初期化する
 */
void OnMapObject::initObjectStatus(float x, float y){
    this->objectStatus.x = x;
    this->objectStatus.y = y;
    this->objectStatus.state = ALIVE;
    this->objectStatus.width = 10;
    this->objectStatus.height = 10;
}

/*
 * オブジェクト間の角度を計算する
 */
Float32 OnMapObject::calculateInterObjectAngle(const ObjectStatus *subject, const ObjectStatus *target)
{
    // オブジェクト間の角度を計算する
    float radian = atan2( subject->y - target->y, subject->x - target->x);

    // 弧度法から度数法へ変換
    // CRIAudioでは正面が0度なので、画面と音の整合性を取るため角度を足す
    int angle = static_cast<int>( ( radian / M_PI * 180 ) + 90 );
    angle = angle % 360;

    return static_cast<Float32>(-angle);

}

/*
 * スピーカーのセンドレベルを計算する
 */
CriAuSendLevel OnMapObject::calculateSpeakerSendLevel(const ObjectStatus *subject, const ObjectStatus
target)
{
    // オブジェクト間の角度を計算する
    Float32 angle = this->calculateInterObjectAngle(subject, target);

    // 角度からセンドレベルを計算する
    Float32 left;
    Float32 right;
    Float32 leftSurround;
    Float32 rightSurround;
    Float32 center;
    CriAuUty::CalcSendLevel5Speakers(angle, &left, &right, &leftSurround, &rightSurround, &center);

    // センドレベルを設定する
    CriAuSendLevel sendLevel;
    sendLevel.SetLeft(left);
    sendLevel.SetRight(right);
```



```
    sendLevel.SetLeftSurround(leftSurround);
    sendLevel.SetRightSurround(rightSurround);
    sendLevel.SetCenter(center);

    return sendLevel;
}

/*
 * オブジェクトの音を鳴らす
 */
void OnMapObject::playSound(string soundName, const CriAuSendLevel &sendLevel)
{
      soundPlayer->setSendLevel(soundName, sendLevel);
    soundPlayer->play(soundName);
}

/*
 * デストラクタ
 */
OnMapObject::~OnMapObject()
{
    delete soundPlayer;
}
```

```
#ifndef _ONMAPOBJECT_H_
#define _ONMAPOBJECT_H_

#include <SDL.h>
#include <windows.h>
#include "SoundPlayer.h"
#include "cri_audio.h"

enum ObjectCondition
{
    ALIVE,
    DEAD
};

struct ObjectStatus
{
    ObjectStatus(){};
    ObjectStatus(float x, float y, int state,    int width, int height):
    x(x), y(y), state(state), width(width), height(height){};
    float x;
    float y;
    int state;
    int width;
    int height;
};

class OnMapObject
{
protected:
    static const double M_PI;

    ObjectStatus objectStatus;
    ObjectStatus initializeStatus;
    ObjectStatus *playerStatus;

    SoundPlayer* soundPlayer;

    // subjectから見たtargetの位置に応じて各スピーカーへのセンドレベルを計算する
    CriAuSendLevel calculateSpeakerSendLevel(const ObjectStatus *subject, const ObjectStatus *target);

    // subjectとtarget間の角度を計算する
    Float32 calculateInterObjectAngle(const ObjectStatus *subject, const ObjectStatus *target);
    void playSound(string soundName, const CriAuSendLevel &sendLevel);

    // 距離に応じたボリュームを計算する
    virtual Float32 calculateVolume(const ObjectStatus *subject, const ObjectStatus *target) = 0;

    //初期化する
    void initObjectStatus(float x, float y);

public:

    void setTarget(ObjectStatus *target){playerStatus = target;};
    void reset(){ objectStatus = initializeStatus; };
    ObjectStatus* getObjectStatus(){return &objectStatus;};

    virtual void move(Uint8 *keys) = 0;
    virtual void resolveCollision() = 0;
    virtual void draw(SDL_Surface *targetScreen) = 0;
    OnMapObject(float x, float y, SoundContainer *soundContainer, string soundListPath, int *time);
    virtual ~OnMapObject();
};
```



```
#endif
```

SDLCommonFunctions.h (1/1)

```
#ifndef _SDLCOMMONFUNCTIONS_H_
#define _SDLCOMMONFUNCTIONS_H_
#include <SDL.h>

namespace
{
    bool PollEvent()
    {
        SDL_Event ev;
        while(SDL_PollEvent(&ev))
        {
            switch(ev.type)
            {
            case SDL_QUIT:// ウィンドウの×ボタンが押された時など
                return false;
                break;
            }

        }
        return true;
    }

    void ClearScreen(SDL_Surface *target)
    {
        // サーフェスを黒で初期化
        SDL_Rect dest;
        dest.x = 0;
        dest.y = 0;
        dest.w = 640;
        dest.h = 480;
        Uint32 color= 0x00000000;
        SDL_FillRect( target, &dest, color );
    }

}
#endif
```

```
#include "SoundContainer.h"

SoundContainer::SoundContainer(string cuePath)
{
    CriError error = CRIERR_OK; // エラー検出用のオブジェクト

    //音声出力の初期化
    soundOut =  CriSmpSoundOutput::Create();
    heap = criHeap_Create(buf, sizeof(buf));
    soundRenderer = CriSoundRendererBasic::Create(heap, error);
    soundOut->SetNotifyCallback( soundOutCallBack, static_cast<void *>(soundRenderer) );
    soundOut->Start();

    // キューの読み込み
    Uint8 *csbdata;
    unsigned long csbsize;
    csbdata = this->loadCue(cuePath, &csbsize);
    cueSheet = CriAuCueSheet::Create(heap, error);
    cueSheet->LoadCueSheetBinaryFileFromMemory("Tutorial", csbdata, csbsize, error);
    free(csbdata);

    // 音声オブジェクトの生成
    audioObject = CriAuObj::Create(heap, soundRenderer, "Tutorial", error);
    audioObject->AttachCueSheet(cueSheet, error);

    // エラー処理
    if( error != CRIERR_OK )
        exit(1);
}

/*
 * 良くわからないけど必要な処理
 */
unsigned long SoundContainer::soundOutCallBack(void *obj, unsigned long nch, Float32 *sample[], unsigned lo
g nsmpl)
{
    CriSoundRendererBasic* sndrndr=(CriSoundRendererBasic*)obj;
    CriError err;

    //    Getting PCM Data from CRI Sound Renderer
    //    nsmpl has to be 128*N samples. (N=1,2,3...)
    sndrndr->GetData(nch, nsmpl, sample, err);

    return nsmpl;
}

/*
 * キューファイルをロードする
 */
Uint8* SoundContainer::loadCue(string path, unsigned long *idtsize)
{
    FILE *fp;
    signed long size;
    Uint8 *idt;

    fp = fopen(path.c_str(), "rb");
    fseek(fp, 0, SEEK_END);
    size = ftell(fp);
    idt = (Uint8*)calloc(size, 1);
    fseek(fp, 0, SEEK_SET);
    fread(idt, size, 1, fp);
    fclose(fp);
```



```
    *idtsize = (unsigned long)size;
    return idt;
}

/*
 * 音をロードする(つまり音声プレーヤーを生成する)
 */
CriAuPlayer* SoundContainer::loadSound(string soundName)
{
    CriError error = CRIERR_OK;
    CriAuPlayer* player = CriAuPlayer::Create(audioObject, error);
    player->SetCue(soundName.c_str(), error);

    return player;
}

SoundContainer::~SoundContainer()
{
    CriError error = CRIERR_OK;

    audioObject->Destroy(error);
    cueSheet->Destroy(error);
    soundOut->SetNotifyCallback(NULL, NULL);
    soundRenderer->Destroy(error);

    //    Print Heap Status
    //criHeap_DebugPrintBlockInformationAll(heap);
    criHeap_Destroy(heap);
}
```

```
1   #ifndef _SOUNDCONTAINER_H_
2   #define _SOUNDCONTAINER_H_
3
4   #include <string>
5   #include <iostream>
6   #include "cri_audio.h"
7   #include "cri_xpt.h"
8   #include "CriSmpSoundOutput.h"
9
10  using std::string;
11
12  class SoundContainer
13  {
14      CriHeap heap;
15      CriSoundRendererBasic* soundRenderer;
16      CriAuObj* audioObject;
17      Uint8 buf[10*1024*1024];
18      CriSmpSoundOutput *soundOut;
19      CriAuCueSheet* cueSheet;
20
21      Uint8* loadCue(string path, unsigned long *idtsize);
22      void createCueSheet();
23      static unsigned long soundOutCallBack(void *obj, unsigned long nch, Float32 *sample[], unsigned long nsmp
    );
24
25  public:
26      ~SoundContainer();
27      SoundContainer(string cuePath);
28      CriAuPlayer* loadSound(string soundName);
29
30  };
31
32  #endif
```

```cpp
#include "SoundPlayer.h"

SoundPlayer::SoundPlayer(SoundContainer *soundContainer, string soundListPath)
{
    // 音声を取得する
    soundShelf = this->loadSound(soundContainer,soundListPath);
}

SoundPlayer::~SoundPlayer()
{
    soundShelf.clear();
}


SoundShelf SoundPlayer::loadSound(SoundContainer *soundContainer, string soundListPath)
{
    // ファイルを開く
    std::ifstream list(soundListPath.c_str());

    // 役割ごとの音声の名前を連想配列に読み込む
    SoundShelf soundList;
    string line;
    while( getline(list,line) )
    {
        // 取得した文の空白位置を探す
        string::size_type splitIndex = line.find(" ",0);

        // 役割と名前が空白で区切られてなかったらスキップする
        if( splitIndex == string::npos )
            continue;

        string soundJobName = line.substr(0,splitIndex);
        string soundName = line.substr(splitIndex+1);
        soundList[soundJobName] = soundContainer->loadSound(soundName);
    }
    list.close();

    return soundList;
}

int SoundPlayer::getStatus(string soundJobName)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        return soundShelf[soundJobName]->GetStatus(error);
    }
    else
    {
        return -1;
    }
}

void SoundPlayer::play(string soundJobName)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->Play(error);
    }
}

void SoundPlayer::loopPlay(string soundJobName)
```



```cpp
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        int soundStatus = soundShelf[soundJobName]->GetStatus(error);
        if( soundStatus == CriAuPlayer::STATUS_STOP || soundStatus == CriAuPlayer::STATUS_PLAYEND )
        {
            soundShelf[soundJobName]->Play(error);
        }
    }
}

void SoundPlayer::stop(string soundJobName)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->Stop(error);
    }
}

void SoundPlayer::stopAll()
{
    CriError error = CRIERR_OK;
    SoundShelfIterator itr;
    for( itr = soundShelf.begin(); itr != soundShelf.end(); itr++ )
    {
        (*itr).second->Stop(error);
    }
}

void SoundPlayer::stopAllExcept(string excludeSoundName)
{
    CriError error = CRIERR_OK;
    SoundShelfIterator itr;

    // 指定された音以外を止める
    for( itr = soundShelf.begin(); itr != soundShelf.end(); itr++ )
    {
        if( (*itr).first != excludeSoundName )
        {
            (*itr).second->Stop(error);
        }
    }
}


void SoundPlayer::setVolume(string soundJobName, float volume)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->SetVolume(volume, error);
        soundShelf[soundJobName]->Update(error);
    }
}

void SoundPlayer::setPitch(string soundJobName, float pitch)
{
    if( soundShelf.find(soundJobName) != soundShelf.end() )
    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->SetPitch(pitch,error);
```

```
127             soundShelf[soundJobName]->Update(error);
128         }
129     }
130
131     void SoundPlayer::setSendLevel(string soundJobName, const CriAuSendLevel &sendLevel)
132     {
133         if( soundShelf.find(soundJobName) != soundShelf.end() )
134         {
135             CriError error = CRIERR_OK;
136             soundShelf[soundJobName]->SetDrySendLevel(sendLevel, error);
137             soundShelf[soundJobName]->Update(error);
138         }
139     }
140
141     void SoundPlayer::setReverb(string soundJobName, float reverb)
142     {
143         if( soundShelf.find(soundJobName) != soundShelf.end() )
144         {
145             CriError error = CRIERR_OK;
146             soundShelf[soundJobName]->SetWetSendLevel(CriAuSendLevel::WET_0, reverb, error);
147             soundShelf[soundJobName]->Update(error);
148         }
149     }
150
151     void SoundPlayer::setCutOffFrequency(string soundJobName, float lowerFrequency, float upperFrequency)
152     {
153         if( soundShelf.find(soundJobName) != soundShelf.end() )
154         {
155             CriError error = CRIERR_OK;
156             soundShelf[soundJobName]->SetFilterCutoffFrequency(lowerFrequency, upperFrequency, error);
157             soundShelf[soundJobName]->Update(error);
158         }
159     }
```

```
#ifndef _SOUNDPLAYER_H_
#define _SOUNDPLAYER_H_

#include "SoundContainer.h"
#include <string>
#include <fstream>
#include <map>
using std::string;

namespace
{
    typedef std::map<string, CriAuPlayer*> SoundShelf;
    typedef SoundShelf::iterator SoundShelfIterator;
}
/*
 * 音声プレイヤー
 */
class SoundPlayer
{
    SoundShelf soundShelf;    // 音声を管理する連想配列
    SoundShelf loadSound(SoundContainer *soundContainer, string soundListPath);

public:
    SoundPlayer(SoundContainer *soundContainer, string soundListPath);
    ~SoundPlayer();
    int getStatus(string soundName);
    void play(string soundName);
    void loopPlay(string soundJobName);
    void stop(string soundName);
    void stopAll();
    void stopAllExcept(string excludeSoundName);
    void setVolume(string soundName, float volume);
    void setPitch(string soundName, float pitch);
    void setSendLevel(string soundName, const CriAuSendLevel &sendLevel);
    void setReverb(string soundName, float reverb);
    void setCutOffFrequency(string soundName, float lowerFrequency, float upperFrequency);
};

#endif
```

```
1       #ifndef _STATEMESSAGES_H_
2       #define _STATEMESSAGES_H_
3
4       namespace
5       {
6           // 状態間で行き交うメッセージ
7           enum StateMessage
8           {
9               QUIT_GAME,    // escが押された・ウィンドウが閉じられた時
10              START_GAME, // タイトルでゲームを開始した
11              GO_TITLE,    // 結果表示を終えた
12          };
13      }
14      #endif
```

```
/**
 * タイトルを表現するクラス
 *
 * @date 2006/11/05
 * @author hashiyaman
 */

#include "TitleState.h"

const Uint8 TitleState::PLAY_GAME = SDLK_RETURN;
const Uint8 TitleState::QUIT = SDLK_ESCAPE;

/*
 * コンストラクタ
 */
TitleState::TitleState(SoundContainer *soundContainer, SDL_Surface *mainScreen)
{
    this->soundPlayer = new SoundPlayer(soundContainer,"TitleStateSound.txt");
    this->mainScreen = mainScreen;

    // 状態の監視をしたいキーを列挙して渡す
    KeyTable useKeyTable;
    useKeyTable.push_back(PLAY_GAME);
    useKeyTable.push_back(QUIT);
    keyState = new KeyState(useKeyTable);
}

/*
 * デストラクタ
 */
TitleState::~TitleState()
{
    delete soundPlayer;
    delete keyState;
}

/*
 * タイトルでの処理を行う
 */
int TitleState::run()
{
    return this->runFrame();
}

// 1フレーム内の処理を行う
int TitleState::runFrame()
{
    // テキストの色とフォントと内容を設定する
    SDL_Color textColor = {255,255,255};
    TTF_Font *font = TTF_OpenFont( "Headache.ttf", 28 );
    SDL_Surface *stateText = TTF_RenderText_Blended( font, "IN TITLE", textColor );

    int gameState;
    soundPlayer->loopPlay("TITLE");

    while(true)
    {
        SDL_PumpEvents(); // イベント状態を更新
        SDL_Delay(50); // CPU使用率が100%になるのを防ぐ

        keyState->update(); // キー入力を取得

        // ゲームが終了された場合
```



```
        if( !PollEvent() || keyState->down(QUIT) )
        {
            gameState = QUIT_GAME;
            break;
        }

        CriError error = CRIERR_OK;

        // タイトルを再生し終えてから，開始方法を再生する
        if ( soundPlayer->getStatus("TITLE") == CriAuPlayer::STATUS_PLAYEND )
        {
            soundPlayer->loopPlay("HOW_TO_START");
        }

        CriAuObj::ExecuteMain(error); // 音声の状態を更新する

        ClearScreen(mainScreen); // 画面をクリアする
        this->draw(10,10,stateText,mainScreen);    // 描画を行う

        SDL_Flip(mainScreen); // 画面の状態を更新する

        if( keyState->released(PLAY_GAME) )
        {
            gameState = START_GAME; // ゲームを開始する
            soundPlayer->stopAll();
            break;
        }
    }

    // 音の停止
    CriError error = CRIERR_OK;
    soundPlayer->stopAll();
    CriAuObj::ExecuteMain(error);

    // 後処理
    TTF_CloseFont(font);
    SDL_FreeSurface(stateText);

    return gameState;
}

/*
 * 描画する
 */
void TitleState::draw(int x, int y, SDL_Surface *source, SDL_Surface *destination )
{
    SDL_Rect position;
    position.x = x;
    position.y = y;
    SDL_BlitSurface(source, NULL, destination, &position);
}
```

```
1       #ifndef _TITLESTATE_H_
2       #define _TITLESTATE_H_
3
4       #include <SDL.h>
5       #include <SDL_ttf.h>
6       #include <string>
7       #include <list>
8       #include "KeyState.h"
9       #include "StateMessages.h"
10      #include "SoundPlayer.h"
11      #include "SDLCommonFunctions.h"
12      using std::string;
13
14      /*
15       * ゲーム部分を管理するクラス
16       */
17      class TitleState
18      {
19          SDL_Surface *mainScreen;
20          SoundPlayer *soundPlayer;
21          KeyState *keyState;
22
23          static const Uint8 PLAY_GAME;
24          static const Uint8 QUIT;
25
26          int runFrame();    // 1フレーム内の処理を行う
27
28          void draw(int x, int y, SDL_Surface *source, SDL_Surface *destination );
29      public:
30          TitleState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
31          ~TitleState();
32          int run();    // ゲームを実行する
33      };
34
35      #endif
```

2007/01/31
さうんど おんりぃ
最終報告書

ソースコード：ForestWalking（C++）

```
1   #include "AfternoonSoundFactory.h"
2
3   AfternoonSoundFactory::AfternoonSoundFactory(void) {
4   }
5
6   AfternoonSoundFactory::~AfternoonSoundFactory(void) {
7   }
8
9   /**
10   * 背景音を作成する.
11   */
12  list<SoundMaterial> AfternoonSoundFactory::createSoundMaterials() {
13      list<SoundMaterial> soundMaterials;
14      soundMaterials.push_back(*(new SoundMaterial("forest1")));
15      return soundMaterials;
16  }
17
18  /**
19   * 動物を作成する
20   */
21  list<Creature> AfternoonSoundFactory::createCreature() {
22      list<Creature> creatures;
23      for (int i = 0; i < 2; i++) {
24          creatures.push_back(*(new Creature("chimpanzee", getRandomCoordinates(), getRandomCoordinates() )
    );
25          creatures.push_back(*(new Creature("ibis", getRandomCoordinates(), getRandomCoordinates() )));
26      }
27      return creatures;
28  }
29
30  /**
31   * 昆虫を作成する
32   */
33  list<Insect> AfternoonSoundFactory::createInsects() {
34      list<Insect> insects;
35      for (int i = 0; i < 5; i++) {
36          insects.push_back(*(new Insect("grasshopper", getRandomCoordinates(), getRandomCoordinates(), 30)));
37      }
38
39      for (int i = 0; i < 3; i++) {
40          insects.push_back(*(new Insect("frog", getRandomCoordinates(), getRandomCoordinates(), 100)));
41      }
42      return insects;
43  }
```

```
#pragma once

#include "GamePlayStateSoundFactory.h"

/**
 * 虫捕り中の昼の音を作成するクラスです.
 *
 * @author ikumin
 * @date 2006/12/15
 */
class AfternoonSoundFactory : public GamePlayStateSoundFactory {
public:
    AfternoonSoundFactory(void);
    ~AfternoonSoundFactory(void);

    list<SoundMaterial> createSoundMaterials();
    list<Creature> createCreature();
    list<Insect> createInsects();
};
```

```
#include "AfternoonState.h"
#include "NightState.h"

/**
 * コンストラクタ
 */
AfternoonState::AfternoonState(SoundContainer *soundContainer, SDL_Surface *mainScreen) : GamePlaySta
e(soundContainer, mainScreen) {
    // 昼の音を生成する
    factory = new AfternoonSoundFactory();
    movableSounds = ((GamePlayStateSoundFactory *) factory)->createMovableSounds();
    soundMaterials =((AfternoonSoundFactory *) factory)->createSoundMaterials();
    Creatures = ((AfternoonSoundFactory *) factory)->createCreature();
    insects = ((AfternoonSoundFactory *) factory)->createInsects();
    rivers = ((GamePlayStateSoundFactory *) factory)->createRiver(soundContainer, GAME_SOUND_PATH);

    gameState = START_GAME;

    // プレーヤーを取得する
    player = new Player();
}

/**
 * デストラクタ
 */
AfternoonState::~AfternoonState(void) {
}

/**
 * 昼の処理を行う(オーバーライド)
 */
void AfternoonState::run() {
    player->speak("start_collect_insects");
    GamePlayState::run();
}

/**
 * 音を描画する
 */
void AfternoonState::draw() {
    if (debugMode == ON) {
        GamePlayState::draw();

        string stateText = "AFTERNOON";
        text = TTF_RenderText_Blended(font, stateText.c_str(), textColor);
        drawText(10, 10, text, mainScreen); // 時間帯の描画
        SDL_FreeSurface(text);
    }
}

/**
 * 時間を経過させる.
 * 一定時間が経過すると, 夜となる.
 */
void AfternoonState::passTime() {
    GamePlayState::passTime();

    if (passingTime == limitTime - FADE_TIME) {
        soundPlayer->play("bell");
    }

    // 夜にする
    if (passingTime >= limitTime) {
```



```
        finalize();
        changeState(new NightState(soundContainer, mainScreen));
        gameState = GO_NIGHT;
    }
}

/**
 * 音とテキストをフェードアウトさせる
 */
void AfternoonState::fadeOut() {
    for (list<SoundMaterial>::iterator i = soundMaterials.begin(); i != soundMaterials.end(); i++) {
        i->setFadeOut(true);
    }
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->fadeOutColor();
        i->setFadeOut(true);
    }
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->fadeOutColor();
        i->setFadeOut(true);
    }
}

/**
 * 音とテキストをフェードインさせる
 */
void AfternoonState::fadeIn() {
}

/**
 * 音とテキストのフェードインを止める
 */
void AfternoonState::stopFadeIn() {
}

void AfternoonState::finalize() {
    // 背景音を止める
    for (list<SoundMaterial>::iterator i = soundMaterials.begin(); i != soundMaterials.end(); i++) {
        i->stopSound();
    }

    // 動物の音を止める
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->stopSound();
    }

    // 昆虫の音を止める
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->stopSound();
    }
}
```

```
#pragma once

#include "AfternoonSoundFactory.h"

/**
 * 昼を表すクラス
 *
 * @author hashiyaman
 * @date 2007/1/16
 */
class AfternoonState : public GamePlayState {
protected:
    // オーバーライド
    void draw();
    void passTime();
    void fadeOut();
    void fadeIn();
    void stopFadeIn();
    void finalize();

public:
    AfternoonState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
    ~AfternoonState();

    // オーバーライド
    void run();
};
```

```
#include "Cage.h"

/**
 * コンストラクタ
 */
Cage::Cage(void) {
}

/**
 * デストラクタ
 */
Cage::~Cage(void) {
}

/**
 * 捕まえた虫を虫かごに入れる
 */
void Cage::putInsect(Insect* insect) {
    catchedInsects.push_back(*insect);
}

/**
 * 虫かごの虫が鳴く
 */
void Cage::playCatchedInsects() {
    for (list<Insect>::iterator i = catchedInsects.begin(); i != catchedInsects.end(); i++) {
        //soundPlayer->setVolume(*i, 0.2);
        //soundPlayer->loopPlay(*i);
        i->getSoundPlayer()->setVolume(i->getName(), 0.2);
        i->getSoundPlayer()->loopPlay(i->getName());
    }
}

/**
 * 虫かごの虫を黙らせる
 */
void Cage::stopCatchedInsects() {
    for (list<Insect>::iterator i = catchedInsects.begin(); i != catchedInsects.end(); i++) {
        i->getSoundPlayer()->stop(i->getName());
    }
}

list<Insect> Cage::getCatchedInsects() {
    return catchedInsects;
}
```

```
1   #pragma once
2
3   #include <list>
4   #include <string>
5
6   #include "Insect.h"
7
8   using std::string;
9   using std::list;
10
11  /**
12   * 虫かごのクラス
13   *
14   * @author ikumin
15   * @version 2006/01/10
16   */
17
18  class Cage {
19
20  private:
21      list<Insect> catchedInsects;
22
23  public:
24      Cage(void);
25      ~Cage(void);
26
27      void putInsect(Insect* insect);
28      void playCatchedInsects();
29      void stopCatchedInsects();
30
31      list<Insect> getCatchedInsects();
32  };
```

```
#pragma once

#include <string>

using std::string;

/**
 * 定数をまとめて管理しておきます.
 * TODO 必要なものは外部ファイル化するべし
 *
 * @author hashiyaman
 * @date 2007/1/15
 */
namespace {
    // 画面関連
    static const int WINDOW_WIDTH = 1024; // ゲームウィンドウの幅
    static const int WINDOW_HEIGHT = 768; // ゲームウィンドウの高さ
    static const int AREA_SIZE_MAX = 3000; // マップの広さ上限
    static const int AREA_SIZE_MIN = 800; // マップの広さ下限

    // ゲーム関連
    static const int LIMIT_TIME_MAX = 10000; // 制限時間上限
    static const int LIMIT_TIME_MIN = 100; // 制限時間下限
    static const int FADE_TIME = 85; // フェードイン・フェードアウトの始まる時間

    // プレーヤー関連
    static const float PLAYER_X = WINDOW_WIDTH / 2; // プレーヤーのX座標
    static const float PLAYER_Y = WINDOW_HEIGHT / 2; // プレーヤーのY座標
    static const int PLAYER_SPEED = 4; // プレーヤーの移動速度

    // パス関連
    static const string TEXTS_PATH = "texts/";
    static const string TITLE_SOUND_PATH = TEXTS_PATH + "TitleStateSound.txt";
    static const string HOW_TO_PLAY_SOUND_PATH = TEXTS_PATH + "HowToPlayStateSound.txt";
    static const string GAME_SOUND_PATH = TEXTS_PATH + "GamePlayStateSound.txt";
    static const string RESULT_SOUND_PATH = TEXTS_PATH + "ResultStateSound.txt";
};
```

```
#include "Creature.h"

/**
 * コンストラクタ
 */
Creature::Creature(string soundName, double x, double y) : MovableSound(soundName, x, y) {
    speed = 3;
    state = WALKING;
    changeMovement();
    //srand((unsigned int) time(NULL));
}

/**
 * デストラクタ
 */
Creature::~Creature(void) {
}

/**
 * 動き方を変える
 */
void Creature::changeMovement() {
    count = rand() % 30;

    // -1～1までの値で，移動方向を表す
    abscissa = (rand() % 2) - 1;
    ordinate = (rand() % 2) - 1;
}

/**
 * 動く
 */
void Creature::move() {
    if (count > 0) {
        if (state == WALKING) {
            moveRandom();
        }
        count--;
    } else {
        changeState();
        changeMovement();
    }
    updateLocation();
}

/**
 * ランダムに動く（8方向）
 */
void Creature::moveRandom() {
    x += speed * abscissa;
    y += speed * ordinate;
}

/**
 * 状態を変える
 */
void Creature::changeState() {
    if (state == WALKING) {
        state = STAYING;
    } else {
        state = WALKING;
    }
}
```

```
#pragma once
#include "MovableSound.h"

#include <time.h>

// 生物の状態
enum CreatureState {
    WALKING, STAYING
};

/**
 * 生物を表すクラスです.
 * 生物を新たに増やす場合は, このクラスを継承してください.
 *
 * @author hashiyaman
 * @date 2007/1/22
 */
class Creature : public MovableSound {

private:
    int state;
    int count; // 状態を維持している時間
    int abscissa; // 進んでいる方向(横)
    int ordinate; // 進んでいる方向(縦)

protected:
    int speed; // 動く早さ

public:
    Creature(string soundName, double x, double y);
    ~Creature(void);

    void move();
    void changeMovement();
    void moveRandom();
    void changeState();
};
```

```
1   #include "Game.h"
2   #include "TitleState.h"
3
4   Game* Game::game = NULL;
5
6   /*
7    * コンストラクタ
8    */
9   Game::Game() {
10      // ウィンドウの初期化
11      SDL_Surface *mainScreen = SDL_SetVideoMode(WINDOW_WIDTH, WINDOW_HEIGHT, 32, SDL_SWSURFAC
    );
12
13      this->mainScreen = mainScreen;
14      soundContainer = new SoundContainer("ForestWalking.csb");
15      state = new TitleState(soundContainer, mainScreen);
16  }
17
18  /*
19   * ゲームを実行する
20   */
21  void Game::run() {
22      while(true)    {
23          state->run();
24      }
25  }
26
27  /*
28   * デストラクタ
29   */
30  Game::~Game() {
31      delete soundContainer;
32      delete mainScreen;
33      delete state;
34      delete game;
35  }
36
37  Game* Game::getInstance() {
38      if (game == NULL) {
39          game = new Game();
40      }
41      return game;
42  }
43
44  void Game::changeState(State *state) {
45      this->state = state;
46  }
47
48  SoundContainer* Game::getSoundContainer() {
49      return soundContainer;
50  }
51
52  SDL_Surface* Game::getMainScreen() {
53      return mainScreen;
54  }
```

```
1   #pragma once
2
3   #include <string>
4   #include "State.h"
5
6   class State;
7
8   /**
9    * ゲーム本体を実行するクラス
10   *
11   * @author ikumin
12   * @date 2006/11/05
13   */
14  class Game {
15  public:
16      ~Game();
17
18      static Game* getInstance();
19      void changeState(State *state);
20      void run();
21      SoundContainer* getSoundContainer();
22      SDL_Surface* getMainScreen();
23
24  private:
25      Game();
26
27      SoundContainer *soundContainer;
28      SDL_Surface *mainScreen;
29      State *state;
30      static Game *game;
31  };
```

```
#include "GamePlayState.h"
#include <iostream>

list<MovableSound> GamePlayState::movableSounds;
list<River> GamePlayState::rivers;

Player* GamePlayState::player;

/**
 * コンストラクタ
 */
GamePlayState::GamePlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen) : State() {
    this->soundPlayer = new SoundPlayer(soundContainer, GAME_SOUND_PATH);
    this->soundContainer = soundContainer;
    this->mainScreen = mainScreen;

    // 設定ファイルから値を取得する
    limitTime = Util::getLimitTime();

    // テキストを初期化する
    textColor.r = 255;
    textColor.g = 255;
    textColor.b = 255;
    font = TTF_OpenFont("Headache.ttf", 28);
}

/**
 * デストラクタ
 */
GamePlayState::~GamePlayState() {
}

/**
 * ゲーム中の処理を行う
 */
void GamePlayState::run() {
    initialize();

    while(gameState != QUIT_GAME && gameState != GO_NIGHT && gameState != GO_RESULT) {
        processKeyEvent();
        playSounds();
        draw();
        moveCreatures();
        update();
        passTime();
    }
}

/**
 * 初期化を行う
 */
void GamePlayState::initialize() {
    // 時間を初期化する
    passingTime = 0;

    // 音を初期化する
    for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
        i->updateLocation();
        i->moveSound();
    }
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->updateLocation();
        i->moveSound();
```



```
    }
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->updateLocation();
        i->moveSound();
    }
    for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
        i->updateLocation();
        i->moveSound();
    }
}

/**
 * 左を向く
 */
void GamePlayState::moveLeft() {
    for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
        i->moveLeft();
        i->moveSound();
    }
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->moveLeft();
        i->moveSound();
    }
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->moveLeft();
        i->moveSound();
    }
    for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
        i->moveLeft();
        i->moveSound();
    }
}

/**
 * 右を向く
 */
void GamePlayState::moveRight() {
    for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
        i->moveRight();
        i->moveSound();
    }
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->moveRight();
        i->moveSound();
    }
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->moveRight();
        i->moveSound();
    }
    for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
        i->moveRight();
        i->moveSound();
    }
}

/**
 * 前に移動する
 */
void GamePlayState::moveFront() {
    for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
        i->moveFront();
        i->moveSound();
    }
```

```
127         for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
128             i->moveFront();
129             i->moveSound();
130         }
131         for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
132             i->moveFront();
133             i->moveSound();
134         }
135         for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
136             i->moveFront();
137             i->moveSound();
138         }
139     }
140
141     /**
142      * 後ろに移動する
143      */
144     void GamePlayState::moveBack() {
145         for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
146             i->moveBack();
147             i->moveSound();
148         }
149         for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
150             i->moveBack();
151             i->moveSound();
152         }
153         for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
154             i->moveBack();
155             i->moveSound();
156         }
157         for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
158             i->moveBack();
159             i->moveSound();
160         }
161     }
162
163     /**
164      * 動物や虫が移動する
165      */
166     void GamePlayState::moveCreatures() {
167         for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
168             i->move();
169             i->moveSound();
170         }
171
172         for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
173             if (!(i->getIsCatched())) { // 捕まってなければ移動する
174                 if (!(i->getIsRunAway())) {
175                     i->move();
176                 } else {
177                     i->runAway();
178                 }
179                 i->moveSound();
180             }
181         }
182     }
183
184     /**
185      * 歩いている場所の状態を更新する.
186      * 歩いている場所によって，足音を変えるために利用する.
187      */
188     void GamePlayState::updateWalkingState() {
189         for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
```



```
190             if (i->isCrossingRiver()) {
191                 player->setWalkingPlace(RIVER);
192                 break;
193             } else {
194                 player->setWalkingPlace(FOREST);
195             }
196         }
197     }
198
199     /**
200      * キーイベントリスナー
201      */
202     void GamePlayState::processKeyEvent() {
203         State::processKeyEvent();
204
205         // 左を向く
206         if(Util::isPressed(SDLK_LEFT)) {
207             moveLeft();
208             player->turnLeft();
209         }
210
211         // 右を向く
212         if(Util::isPressed(SDLK_RIGHT)) {
213             moveRight();
214             player->turnRight();
215         }
216
217         // 前進する
218         if(Util::isPressed(SDLK_UP)) {
219             moveFront();
220             updateWalkingState();
221             player->walkFront();
222             if (!isInsideMovableArea()) {
223                 warn();
224                 player->walkBack();
225                 moveBack();
226             }
227         }
228
229         // 後退する
230         if(Util::isPressed(SDLK_DOWN)) {
231             moveBack();
232             updateWalkingState();
233             player->walkBack();
234             if (!isInsideMovableArea()) {
235                 warn();
236                 player->walkFront();
237                 moveFront();
238             }
239         }
240
241         // 虫を捕まえる
242         if(Util::isPressed(SDLK_SPACE)) {
243             for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
244                 if (!(i->getIsCatched()) && !(i->getIsRunAway()) && (i->isInsideCatchableArea()) && !player->getIsSwi
    nging()) { // 捕まえられる条件が揃っている場合
245                     player->catchInsect(&(*i));
246                     break;
247                 }
248             }
249             if (!player->getIsSwinging()) {
250                 player->swingNet();
251                 player->setSwinging(true);
```

```
            }
        } else {
            player->setSwinging(false);
        }
    }

    /**
     * 移動可能範囲内にいるか
     */
    bool GamePlayState::isInsideMovableArea() {
        int areaSize = Util::getAreaSize();
        int areaStartX = -(areaSize - WINDOW_WIDTH) / 2;
        int areaStartY = -(areaSize - WINDOW_HEIGHT) / 2;
        int areaEndX = areaStartX + areaSize;
        int areaEndY = areaStartX + areaSize;

        return (areaStartX < player->getX() && player->getX() < areaEndX)
            && (areaStartY < player->getY() && player->getY() < areaEndY);
    }

    /**
     * 移動可能範囲外に出そうであることを警告する
     */
    void GamePlayState::warn() {
        if (soundPlayer->getStatus("tiger") != CriAuPlayer::STATUS_PLAYING) {
            soundPlayer->play("tiger");
        }

        if (soundPlayer->getStatus("cannot_go_forward") != CriAuPlayer::STATUS_PLAYING) {
            soundPlayer->play("cannot_go_forward");
        }
    }

    /**
     * 森の時間帯にあった音を鳴らす
     */
    void GamePlayState::playSounds() {
        // 背景音を鳴らす
        for (list<SoundMaterial>::iterator i = soundMaterials.begin(); i != soundMaterials.end(); i++) {
            i->playSound();
        }

        // 自然物の音を鳴らす
        for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
            i->playSound();
        }

        // 動物の音を鳴らす
        for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
            i->playSound();
        }

        // 昆虫の音を鳴らす
        for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
            i->playSound();
        }

        // 川の音を鳴らす
        for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
            i->playSound();
        }

        // 虫かごに捕らえている昆虫の音を鳴らす
```



```
        player->playCatchedInsects();
    }

    /**
     * 時間を経過させる
     */
    void GamePlayState::passTime() {
        // 時間を進める
        if (passingTime < limitTime) {
            passingTime++;
        }

        // 音とテキストをフェイドインさせる
        if (passingTime <= FADE_TIME) {
            fadeIn();
        } else {
            stopFadeIn();
        }

        // 音とテキストをフェイドアウトさせる
        if (passingTime >= limitTime - FADE_TIME) {
            fadeOut();
        }
    }

    /**
     * 描画を行う(デバッグ用)
     */
    void GamePlayState::draw() {
        if (debugMode == ON) {
            // 制限時間の描画
            string timeText = "TIME: " + boost::lexical_cast<string>(limitTime - passingTime);
            text = TTF_RenderText_Blended(font, timeText.c_str(), textColor);
            drawText(10, 30, text, mainScreen);

            // プレーヤーの描画
            drawText((int) PLAYER_X, (int) PLAYER_Y, Player::getText(), mainScreen);
            SDL_FreeSurface(text);

            // 現在位置(X座標)の描画
            string xText = "X:" + boost::lexical_cast<string>(player->getX());
            text = TTF_RenderText_Blended(font, xText.substr(0, 6).c_str(), textColor);
            drawText(10, 60, text, mainScreen);
            SDL_FreeSurface(text);

            // 現在位置(Y座標)の描画
            string yText = "Y:" + boost::lexical_cast<string>(player->getY());
            text = TTF_RenderText_Blended(font, yText.substr(0, 6).c_str(), textColor);
            drawText(90, 60, text, mainScreen);
            SDL_FreeSurface(text);

            // 現在いる位置が移動可能範囲内かどうか
            string areaText = "AREA:";
            if (isInsideMovableArea()) {
                areaText += "Inside";
            } else {
                areaText += "OutSide";
            }
            text = TTF_RenderText_Blended(font, areaText.c_str(), textColor);
            drawText(170, 60, text, mainScreen);
            SDL_FreeSurface(text);

            // プレーヤーの向いている角度を描画
```

```
378             string angleText = "ANGLE:" + boost::lexical_cast<string>(player->getAngle());
379             text = TTF_RenderText_Blended(font, angleText.substr(0, 9).c_str(), textColor);
380             drawText(10, 90, text, mainScreen);
381             SDL_FreeSurface(text);
382 
383             // 音の描画
384             drawMovableSounds();
385         }
386     }
387 
388     /**
389      * 森の時間帯にあった描画を行う
390      */
391     void GamePlayState::drawMovableSounds() {
392         // 音を発する川以外の自然物を描画する
393         for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
394             drawText((int) i->getX(), (int) i->getY(), i->getText(), mainScreen);
395         }
396 
397         // 動物を描画する
398         for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
399             drawVolume((int) i->getX(), (int) i->getY(), i->getVolume());
400             drawText((int) i->getX(), (int) i->getY(), i->getText(), mainScreen);
401         }
402 
403         // 昆虫を描画する
404         for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
405             drawVolume((int) i->getX(), (int) i->getY(), i->getVolume());
406             drawText((int) i->getX(), (int) i->getY(), i->getText(), mainScreen);
407         }
408 
409         // 川を描画する
410         for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
411             drawText((int) i->getX(), (int) i->getY(), i->getText(), mainScreen);
412         }
413     }
414 
415     /**
416      * 音量の表示を行う(デバッグ用)
417      */
418     void GamePlayState::drawVolume(double targetX, double targetY, double volume) {
419         // 音量を取得する
420     string volumeText = boost::lexical_cast<string>(volume);
421 
422         // 音量を表示する
423         text = TTF_RenderText_Blended(font, ("(" + volumeText.substr(0, 4) + ")").c_str(), textColor);
424         drawText((int) (targetX + 30), (int) targetY, text, mainScreen);
425 
426         SDL_FreeSurface(text);
427     }
```

```
#pragma once

#include "GamePlayStateSoundFactory.h"
#include "Player.h"
#include "State.h"

#include <fstream>
#include <boost/bind.hpp>
#include <boost/lexical_cast.hpp>
#include <algorithm>
#include <cstdlib>

using std::string;

/**
 * 虫捕り中を表現するクラスです.
 * プレーヤーの動き, 各オブジェクトの定位や音量の変更などを管理します.
 *
 * @author ikumin
 * @date 2006/12/15
 */
class GamePlayState : public State {
private:
    void moveLeft();
    void moveRight();
    void moveFront();
    void moveBack();

    bool isInsideMovableArea();
    void warn();
    void drawVolume(double targetX, double targetY, double volume);

protected:
    static Player *player;
    int gameState;
    int limitTime; // 制限時間
    int passingTime; // 経過時間

    // ゲーム中の音
    list<SoundMaterial> soundMaterials;
    static list<MovableSound> movableSounds;
    list<Creature> Creatures;
    list<Insect> insects;
    static list<River> rivers;

    GamePlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen);

    void updateWalkingState();
    void initialize();
    virtual void moveCreatures();
    virtual void passTime();
    virtual void drawMovableSounds();
    virtual void fadeOut() = 0;
    virtual void fadeIn() = 0;
    virtual void stopFadeIn() = 0;

    // オーバーライド
    void playSounds();
    virtual void processKeyEvent();
    virtual void draw();
    virtual void finalize() = 0;

public:
```



```
    ~GamePlayState();
    virtual void run();
    bool existsSound(string name);
};
```

```
1   #include "GamePlayStateSoundFactory.h"
2
3   const float GamePlayStateSoundFactory::RIVER_SIZE = 50;
4   const int GamePlayStateSoundFactory::RIVER_LENGTH = 20;
5
6   /**
7    * コンストラクタ
8    */
9   GamePlayStateSoundFactory::GamePlayStateSoundFactory(void) {
10      srand((unsigned int) time(NULL));
11  }
12
13  /**
14   * デストラクタ
15   */
16  GamePlayStateSoundFactory::~GamePlayStateSoundFactory(void) {
17  }
18
19  /**
20   * 自然物を作成する.
21   * ここで作成するのは，背景音でも昆虫でもない音（風・滝など）である.
22   */
23  list<MovableSound> GamePlayStateSoundFactory::createMovableSounds() {
24      list<MovableSound> movableSounds;
25      for (int i = 0; i < 2; i++) {
26          movableSounds.push_back(*(new MovableSound("wind", getRandomCoordinates(), getRandomCoordinat
    s())));
27      }
28      return movableSounds;
29  }
30
31  /**
32   * 川を作成する.
33   * 川は線状に配置しているため，複数作成する必要がある.
34   * そのため，現在は効果音とは別に作成している.
35   */
36  list<River> GamePlayStateSoundFactory::createRiver(SoundContainer *soundContainer, string soundListPath)
    {
37      list<River> rivers;
38      double riverX = getRandomCoordinates();
39      double riverY = getRandomCoordinates();
40
41      // 川を線状に配置する
42      for (int i = 0; i <= RIVER_LENGTH; i++) {
43          rivers.push_back(*(new River(riverX, riverY + (i * RIVER_SIZE))));
44      }
45      return rivers;
46  }
47
48  /**
49   * ランダムな座標を取得する
50   */
51  double GamePlayStateSoundFactory::getRandomCoordinates() {
52      return rand() % Util::getAreaSize();
53  }
```

```
1   #pragma once
2
3   #include "GamePlayState.h"
4   #include "SoundFactory.h"
5
6   #include <time.h>
7
8   /**
9    * 虫捕り中に共通して使われる音を作成するクラスです.
10   *
11   * @author ikumin
12   * @date 2006/12/15
13   */
14  class GamePlayStateSoundFactory : public SoundFactory {
15  public:
16      // 川
17      static const float RIVER_SIZE;
18      static const int RIVER_LENGTH;
19
20      GamePlayStateSoundFactory(void);
21      ~GamePlayStateSoundFactory(void);
22      virtual list<MovableSound> createMovableSounds();
23      virtual list<Creature> createCreature() = 0;
24      virtual list<Insect> createInsects() = 0;
25      virtual list<River> createRiver(SoundContainer *soundContainer, string soundListPath);
26      double getRandomCoordinates();
27  };
```

```
#pragma once

/**
 * 複数のクラスで行きかう移動状態を管理する.
 *
 * @author hashiyaman
 * @version 2006/12/16
 */

namespace {
    enum WalkingState {
        STEP_LEFT, // 左足を踏み出している
        STEP_RIGHT, // 右足を踏み出している
        FOREST, // 森を歩いている
        RIVER // 川を歩いている
    };

    enum ForestState {
        AFTERNOON, // 昼
        NIGHT // 夜
    };
}
```

```
#include "HowToPlayState.h"
#include "TutorialState.h"

const string insects[] = {"grasshopper", "cricket", "mole_cricket", "frog"};

/**
 * コンストラクタ
 */
HowToPlayState::HowToPlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen) : State() {
    this->soundPlayer = new SoundPlayer(soundContainer, HOW_TO_PLAY_SOUND_PATH);
    this->mainScreen = mainScreen;
    this->soundContainer = soundContainer;

    readingState = EXPLANATION_OF_SKIP;
    soundPlayerState = NOT_READING;
    readingSoundOfInsect = 0;

    // テキストを初期化する
    textColor.r = 255;
    textColor.g = 255;
    textColor.b = 255;
    font = TTF_OpenFont("Headache.ttf", 28);
}

/**
 * デストラクタ
 */
HowToPlayState::~HowToPlayState(void) {
}

void HowToPlayState::run() {
    while(gameState != START_TUTORIAL) {
        processKeyEvent();
        playSounds();
        draw();
        update();
    }
    finalize();

    changeState(new TutorialState(soundContainer, mainScreen));
}

void HowToPlayState::processKeyEvent() {
    State::processKeyEvent();

    // 説明を飛ばす
    if (Util::isPressedOnce(SDLK_RETURN)) {
        skip();
    }
}

/**
 * 説明を飛ばす
 */
void HowToPlayState::skip() {
    soundPlayer->stopAll(); // 再生中の説明を終了する
    soundPlayerState = NOT_READING;

    // 次の説明に行く
    switch (readingState) {
        case END_OF_HOW_TO_PLAY:
            gameState = START_TUTORIAL;
            break;
```



```
        default: // switch文で一つずつ書いたほうが分かり易いのかも
            readingState++;
    }
}

/**
 * 説明を読み上げる
 */
void HowToPlayState::playSounds() {
    switch (readingState) {
        case EXPLANATION_OF_SKIP:
            readExplanationOfSkip();
            break;

        case EXPLANATION_OF_GOAL:
            readExplanationOfGoal();
            break;

        case HOW_TO_PLAY:
            readHowToPlay();
            break;

        case EXPLANATION_OF_INSECTS:
            readExplanationOfInsects();
            break;

        case SOUND_OF_INSECT:
            readSoundOfInsects();
            break;

        case END_OF_HOW_TO_PLAY:
            readEndOfHowToPlay();
            break;
    }
}

/**
 * スキップについて説明する
 */
void HowToPlayState::readExplanationOfSkip() {
    if (soundPlayer->getStatus("explanation_of_skip") != CriAuPlayer::STATUS_PLAYING && soundPlayerState
= NOT_READING) {
        soundPlayer->play("explanation_of_skip");
        soundPlayerState = READING_RULE;
    } else if (soundPlayer->getStatus("explanation_of_skip") != CriAuPlayer::STATUS_PLAYING && soundPlayer
tate == READING_RULE) {
        readingState = EXPLANATION_OF_GOAL;
        soundPlayerState = NOT_READING;
    }
}

/**
 * ゲームの目的を読み上げる
 */
void HowToPlayState::readExplanationOfGoal() {
    if (soundPlayer->getStatus("explanation_of_goal") != CriAuPlayer::STATUS_PLAYING && soundPlayerState
= NOT_READING) {
        soundPlayer->play("explanation_of_goal");
        soundPlayerState = READING_RULE;
    } else if (soundPlayer->getStatus("explanation_of_goal") != CriAuPlayer::STATUS_PLAYING && soundPlayer
tate == READING_RULE) {
        readingState = HOW_TO_PLAY;
```

```
123             soundPlayerState = NOT_READING;
124         }
125     }
126
127     /**
128      * 操作方法を読み上げる
129      */
130     void HowToPlayState::readHowToPlay() {
131         if (soundPlayer->getStatus("how_to_play") != CriAuPlayer::STATUS_PLAYING && soundPlayerState == NO
    _READING) {
132             soundPlayer->play("how_to_play");
133             soundPlayerState = READING_RULE;
134         } else if (soundPlayer->getStatus("how_to_play") != CriAuPlayer::STATUS_PLAYING && soundPlayerState =
     READING_RULE) {
135             readingState = EXPLANATION_OF_INSECTS;
136             soundPlayerState = NOT_READING;
137         }
138     }
139
140     /**
141      * 虫の鳴き声の説明を読み上げる
142      */
143     void HowToPlayState::readExplanationOfInsects() {
144         if (soundPlayer->getStatus("explanation_of_insects") != CriAuPlayer::STATUS_PLAYING && soundPlayerSt
    te == NOT_READING) {
145             soundPlayer->play("explanation_of_insects");
146             soundPlayerState = READING_RULE;
147         } else if (soundPlayer->getStatus("explanation_of_insects") != CriAuPlayer::STATUS_PLAYING && soundPla
    erState == READING_RULE) {
148             readingState = SOUND_OF_INSECT;
149             soundPlayerState = NOT_READING;
150         }
151     }
152
153     /**
154      * 虫の鳴き声を聞かせる
155      */
156     void HowToPlayState::readSoundOfInsects() {
157         if (readingSoundOfInsect < LENGTH(insects)) {
158             string insectName = insects[readingSoundOfInsect];
159             string soundName = "sound_of_" + insectName;
160
161             // 虫ごとに鳴き声と名前を読み上げる
162             if (soundPlayer->getStatus(insectName) != CriAuPlayer::STATUS_PLAYING && soundPlayerState == NO
    _READING) {
163                 soundPlayer->play(insectName);
164                 soundPlayerState = READING_SOUND_OF_INSECT;
165             } else if (soundPlayer->getStatus(insectName) != CriAuPlayer::STATUS_PLAYING && soundPlayer->get
    tatus(soundName) != CriAuPlayer::STATUS_PLAYING && soundPlayerState == READING_SOUND_OF_INSECT)
    {
166                 soundPlayer->play(soundName);;
167                 soundPlayerState = READING_NAME_OF_INSECT;
168             } else if (soundPlayer->getStatus(soundName) != CriAuPlayer::STATUS_PLAYING && soundPlayerState
    = READING_NAME_OF_INSECT) {
169                 readingSoundOfInsect++; // 次の虫に行く
170                 soundPlayerState = NOT_READING;
171             }
172         } else {
173             readingState = END_OF_HOW_TO_PLAY;
174         }
175     }
176
177     /**
```



```
178      * 説明が終わることを知らせる
179      */
180     void HowToPlayState::readEndOfHowToPlay() {
181         if (soundPlayer->getStatus("go_tutorial") != CriAuPlayer::STATUS_PLAYING && soundPlayerState == NOT_
    EADING) {
182             soundPlayer->play("go_tutorial");
183             soundPlayerState = READING_RULE;
184         }
185     }
186
187     void HowToPlayState::draw() {
188         if (debugMode == ON) {
189             string stateText = "How_to_play";
190             text = TTF_RenderText_Blended(font, stateText.c_str(), textColor);
191             drawText(10, 10, text, mainScreen); // 文字の描画
192             SDL_FreeSurface(text);
193         }
194     }
```

```
1   #ifndef _HOWTOPLAYSTATE_H_
2   #define _HOWTOPLAYSTATE_H_
3
4   #pragma once
5
6   #include "State.h"
7
8   /**
9    * ゲームの説明を行います.
10   *
11   * @author hashiyaman
12   * @date 2006/1/22
13   */
14
15  /* 配列の大きさを調べるマクロ */
16  #define LENGTH(array) (sizeof (array) / sizeof *(array))
17
18  class HowToPlayState : public State {
19
20  private:
21      int readingState; // どの説明を読み上げているか
22      int soundPlayerState; // CriAuPlayerのStatusで対応できない部分をカバーする
23      int readingSoundOfInsect; // どの虫の鳴き声を聞かせているか
24
25      // 説明を読み上げる
26      void readExplanationOfSkip();
27      void readExplanationOfGoal();
28      void readHowToPlay();
29      void readExplanationOfInsects();
30      void readSoundOfInsects();
31      void readEndOfHowToPlay();
32
33      void skip();
34
35  protected:
36      // オーバーライド
37      void processKeyEvent();
38      void playSounds();
39      void draw();
40
41  public:
42      HowToPlayState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
43      ~HowToPlayState(void);
44      void run();
45  };
46
47  #endif
```

```cpp
#include "Insect.h"

const int Insect::RUN_AWAY_TIME = 10;
const int Insect::RUN_AWAY_SPEED = 10;

const int Insect::CATCHABLE_AREA_X = -30;
const int Insect::CATCHABLE_AREA_Y = -100;
const int Insect::CATCHABLE_AREA_SIZE = 100;

/**
 * コンストラクタ（点数なし）
 */
Insect::Insect(string soundName, double x, double y) : Creature(soundName, x, y) {
    //Insect(soundName, x, y, 0);
    point = 0;
    isCatched = false;
    isRunAway = false;
    runAwayTime = 0;
    speed = 1;
    runAwayAbscissa = 0;
    runAwayOrdinate = 0;
}

/**
 * コンストラクタ（点数あり）
 */
Insect::Insect(string soundName, double x, double y, int point) : Creature(soundName, x, y) {
    this->point = point;
    isCatched = false;
    isRunAway = false;
    runAwayTime = 0;
    speed = 1;
    runAwayAbscissa = 0;
    runAwayOrdinate = 0;
    //srand((unsigned int) time(NULL));
}

/**
 * デストラクタ
 */
Insect::~Insect(void) {
}

/**
 * 捕まるかどうか返す
 */
bool Insect::isCatchable() {
    //float distanceX = this->x - PLAYER_X;
    double distanceY = PLAYER_Y - this->y;

    int uncatchableProbability = rand() % 100; // 逃げられる確率
    if (uncatchableProbability < getCatchableProbability(distanceY)) {
        isCatched = true;
        soundPlayer->stop(soundName); // 捕まると鳴き声が止む
        return true;
    } else {
        isRunAway = true;
        do {
            runAwayAbscissa = (rand() % 3) - 1;
            runAwayOrdinate = (rand() % 2) - 1;
        } while (runAwayAbscissa == 0 && runAwayOrdinate == 0);
    }
    return false;
```



```cpp
}

/**
 * 虫が捕まえることができる範囲内にいるかどうかを返す
 */
bool Insect::isInsideCatchableArea() {
    double distanceX = this->x - PLAYER_X;
    double distanceY = this->y - PLAYER_Y;

    return (CATCHABLE_AREA_X <= distanceX && distanceX <= CATCHABLE_AREA_X + CATCHABLE_AREA_
IZE)
        && (CATCHABLE_AREA_Y <= distanceY && distanceY <= CATCHABLE_AREA_Y + CATCHABLE_AREA_
IZE);
}

/**
 * 虫を捕まえることができる確率を計算する
 */
int Insect::getCatchableProbability(double distance) {
    int probability = 100 - distance; // 距離から確率を計算する
    if (probability < 0 || probability > 100) {
        probability = 0;
    }
    return probability;
}

/**
 * 鳴く(オーバーライド)
 */
void Insect::playSound() {
    if (!isCatched) { // 捕まえられていなければ鳴く
        if (isFadeOut) {
            fadeOutSound();
        } else if (isFadeIn) {
            fadeInSound();
        }
        soundPlayer->setVolume(soundName, volume);
        soundPlayer->loopPlay(soundName);
    }
}

/**
 * 逃げる
 */
void Insect::runAway() {
    x += RUN_AWAY_SPEED * runAwayAbscissa;
    y += RUN_AWAY_SPEED * runAwayOrdinate;
    updateLocation();
    runAwayTime++;
    if (runAwayTime > RUN_AWAY_TIME) {
        isRunAway = false;
        runAwayTime = 0;
    }
}

/**
 * 虫の名前を取得する
 */
SDL_Surface* Insect::getText() {
    if (!isCatched) {
        return text;
    } else { // 捕まっている場合は，名前は出ない
        return NULL;
```

```
125         }
126     }
127
128     bool Insect::getIsRunAway() {
129         return isRunAway;
130     }
131
132     bool Insect::getIsCatched() {
133         return isCatched;
134     }
135
136     int Insect::getPoint() {
137         return point;
138     }
```

```
#pragma once

#include "Creature.h"

#include <stdlib.h>
#include <list>

using std::list;

/**
 * 昆虫を表すクラスです.
 * 自立的に動いたり, 捕まえることができます.
 *
 * @author ikumin
 * @version 2006/12/15
 */

class Insect : public Creature {
private:
    static const int RUN_AWAY_TIME;
    static const int RUN_AWAY_SPEED;

    // 虫を捕まえることができる範囲
    static const int CATCHABLE_AREA_X;
    static const int CATCHABLE_AREA_Y;
    static const int CATCHABLE_AREA_SIZE;

    int runAwayTime;
    int point; // 虫を捕まえたときのポイント
    bool isCatched; // 捕まえられたかどうか
    bool isExist;
    bool isRunAway; // 捕まえ損ねたときに, 逃げるかどうか

    int runAwayAbscissa; // 逃げていく方向(横)
    int runAwayOrdinate; // 逃げていく方向(縦)

public:
    Insect(string soundName, double x, double y);
    Insect(string soundName, double x, double y, int point);
    ~Insect(void);
    bool isCatchable();
    bool isInsideCatchableArea();
    int getCatchableProbability(double distance);
    void runAway();
    SDL_Surface* getText();

    virtual void playSound();

    // getter & setter
    bool getIsRunAway();
    bool getIsCatched();
    int getPoint();
};
```

```cpp
#include "KeyFlag.h"

/**
 * キーの状態を管理するクラス
 *
 * @date 2006/11/05
 * @author hashiyaman
 */

/*
 * キーの状態を更新する
 */
void KeyFlag::update(Uint8 *keyState)
{
    if(keyState[id]) // キーが押されてたら
    {
        down = true;
    }
    else if( !keyState[id] && down) // キーが離された瞬間だったら
    {
        released = true;
        down = false;
    }
    else if( !keyState[id] && !down ) // キーが押されてなかったら
    {
        released = false;
    }
}

/*
 * キーが押されているかを返す
 */
bool KeyFlag::isDown()
{
    return down;
}

/*
 * キーが離されているかを返す
 */
bool KeyFlag::isReleased()
{
    return released;
}
```

KeyFlag.h (1/1)

```
1       #ifndef _KEYFLAG_H_
2       #define _KEYFLAG_H_
3
4       #include <SDL.h>
5
6       class KeyFlag
7       {
8           Uint8 id;
9           bool down;
10          bool released;
11      public:
12          bool isDown();
13          bool isReleased();
14          void update(Uint8 *keyState);
15          KeyFlag(Uint8 keyId): id(keyId), down(false), released(false){};
16      };
17
18      #endif
```

```
#include "KeyState.h"

/**
 * キーの状態を監視するクラス
 *
 * @date 2006/11/05
 * @author hashiyaman
 */

/*
 * コンストラクタ
 */
KeyState::KeyState(KeyTable keyTable) {
    KeyTableItr itr;

    for(itr = keyTable.begin(); itr != keyTable.end(); itr++) {
        keys[*itr] = new KeyFlag(*itr);
    }

}

/*
 * キーの状態を監視する
 */
void KeyState::update() {
    Uint8* keyState = SDL_GetKeyState(NULL);
    KeyFlagTableItr itr;
    for( itr = keys.begin(); itr != keys.end(); itr++ )    {
        (*itr).second->update(keyState);
    }
}

/*
 * デストラクタ
 */
KeyState::~KeyState() {
    KeyFlagTableItr itr;
    for(itr = keys.begin(); itr != keys.end(); itr++) {
        delete (*itr).second;
    }

    keys.clear();
}

/*
 * キーが押されているかを返す
 */
bool KeyState::down(Uint8 key) {
    return keys[key]->isDown();
}

/*
 * キーが離されているかを返す
 */
bool KeyState::released(Uint8 key) {
    return keys[key]->isReleased();
}
```

```
1   #ifndef _KEYSTATE_H_
2   #define _KEYSTATE_H_
3
4   #include <SDL.h>
5   #include <vector>
6   #include <map>
7   #include <algorithm>
8   #include <boost/bind.hpp>
9
10  #include "KeyFlag.h"
11
12  namespace
13  {
14      typedef std::vector<Uint8> KeyTable;
15      typedef KeyTable::iterator KeyTableItr;
16
17      typedef std::map<Uint8,KeyFlag*> KeyFlagTable;
18      typedef KeyFlagTable::iterator KeyFlagTableItr;
19
20  }
21
22  class KeyState
23  {
24      KeyFlagTable keys;
25  public:
26      KeyState(KeyTable keyTable);
27      ~KeyState();
28      void update();
29      bool down(Uint8 key);
30      bool released(Uint8 key);
31  };
32
33  #endif
```

```
1   #include <SDL.h>
2   #include <SDL_ttf.h>
3   #include "Game.h"
4
5   #pragma comment(lib, "SDL.lib")
6   #pragma comment(lib, "SDLmain.lib")
7   #pragma comment(lib, "SDL_ttf.lib")
8   #pragma comment(lib, "cri_audio_pc.lib")
9   #pragma comment(lib, "cri_base_pc.lib")
10  #pragma comment(lib, "dsound.lib")
11  #pragma comment(lib, "winmm.lib")
12
13  int main(int argc, char* argv[]) {
14      // 初期化
15      if(SDL_Init( SDL_INIT_AUDIO|SDL_INIT_VIDEO ) == -1 || TTF_Init() == -1) {
16          fprintf(stderr,"初期化に失敗¥n");
17          exit(1);
18      }
19
20      // キャプションの設定
21      SDL_WM_SetCaption( "Forest Walking", NULL );
22
23      // マウスカーソルを消す
24      SDL_ShowCursor(SDL_DISABLE);
25
26      // メインループ
27      Game *game = Game::getInstance();
28      game->run();
29
30      // 終了処理
31      TTF_Quit();
32      SDL_Quit();
33      return 0;
34  }
```

```cpp
#include "MovableSound.h"
#include "Game.h"

/**
 * コンストラクタ
 */
MovableSound::MovableSound(string soundName, double x, double y) : SoundMaterial(soundName) {
    this->x = x;
    this->y = y;

    createNameText();

    appearanceColor = 255;
    disappearanceColor = 0;
}

/**
 * デストラクタ
 */
MovableSound::~MovableSound(void) {
    //TTF_CloseFont(font);
}

/**
 * 虫の名前（テキスト）を作る
 */
void MovableSound::createNameText() {
    // 色の設定
    textColor.r = 255;
    textColor.g = 255;
    textColor.b = 255;

    // フォントの作成
    font = TTF_OpenFont("Headache.ttf", 28);

    // テキストの作成
    string soundNameText = boost::lexical_cast<string>(soundName).substr(0, 2);
    text = TTF_RenderText_Blended(font, soundNameText.c_str(), textColor);

    // フォントの開放
    TTF_CloseFont(font);
}

/**
 * 位置を更新する
 */
void MovableSound::updateLocation() {
    angle = Util::calculateInterObjectAngle(x, y);
    distance = Util::calculateDistance(x, y);
}

/**
 * 左を向く
 */
void MovableSound::moveLeft() {
    angle += PLAYER_SPEED;
    x = distance * cos(Util::angleToRadian(angle)) + PLAYER_X;
    y = distance * sin(Util::angleToRadian(angle)) + PLAYER_Y;
}

/**
 * 右を向く
 */
```



```cpp
void MovableSound::moveRight() {
    angle -= PLAYER_SPEED;
    x = distance * cos(Util::angleToRadian(angle)) + PLAYER_X;
    y = distance * sin(Util::angleToRadian(angle)) + PLAYER_Y;
}

/**
 * 前進する
 */
void MovableSound::moveFront() {
    y += PLAYER_SPEED;
    updateLocation();
}

/**
 * 後退する
 */
void MovableSound::moveBack() {
    y -= PLAYER_SPEED;
    updateLocation();
}

/**
 * 音を動かす
 */
void MovableSound::moveSound() {
    volume = Util::calculateVolume(x, y);

    CriAuSendLevel sendLevel = Util::calculateSpeakerSendLevel(x, y);
    soundPlayer->setSendLevel(soundName, sendLevel);
}

/**
 * 色をフェードアウトさせる
 */
void MovableSound::fadeOutColor() {
    if (appearanceColor > 0) {
        SDL_FreeSurface(text);

        // フォントを初期化する
        if (appearanceColor == 255) {
            font = TTF_OpenFont("Headache.ttf", 28);
        }
        textColor.r = appearanceColor;
        textColor.g = appearanceColor;
        textColor.b = appearanceColor;

        string soundNameText = boost::lexical_cast<string>(soundName).substr(0, 2);
        text = TTF_RenderText_Blended(font, soundNameText.c_str(), textColor);

        appearanceColor -= 3; // 色を減少させる
    } else {
        // フォントを開放する
        TTF_CloseFont(font);
    }
}

/**
 * 色をフェードインさせる
 */
void MovableSound::fadeInColor() {
    if (disappearanceColor <= 255) {
        SDL_FreeSurface(text);
```

```
127
128             // フォントを初期化する
129             if (disappearanceColor == 0) {
130                 font = TTF_OpenFont("Headache.ttf", 28);
131             }
132             textColor.r = disappearanceColor;
133             textColor.g = disappearanceColor;
134             textColor.b = disappearanceColor;
135
136             string soundNameText = boost::lexical_cast<string>(soundName).substr(0, 2);
137             text = TTF_RenderText_Blended(font, soundNameText.c_str(), textColor);
138
139             disappearanceColor += 3; // 色を増加させる
140         } else {
141             // フォントを開放する
142             TTF_CloseFont(font);
143         }
144     }
145
146     /**
147      * 音をフェードインさせる(オーバーライド)
148      */
149     void MovableSound::fadeInSound() {
150         if (Util::calculateVolume(x, y) >= (increasingWidth += 0.001)) {
151             increasingWidth += 0.001;
152             volume = increasingWidth;
153         } else {
154             volume = Util::calculateVolume(x, y);
155         }
156     }
157
158     /************************************************
159      *              Getter & Setter                 *
160      ************************************************/
161     SDL_Surface* MovableSound::getText() {
162         return text;
163     }
164
165     double MovableSound::getX() {
166         return x;
167     }
168
169     void MovableSound::setX(double x) {
170         this->x = x;
171     }
172
173     double MovableSound::getY() {
174         return y;
175     }
176
177     void MovableSound::setY(double y) {
178         this->y = y;
179     }
180
181     double MovableSound::getAngle() {
182         return angle;
183     }
184
185     void MovableSound::setAngle(double angle) {
186         this->angle = angle;
187     }
188
189
```



```
190     double MovableSound::getDistance() {
191         return distance;
192     }
193
194     void MovableSound::setDistance(double distance) {
195         this->distance = distance;
196     }
197
198     SoundPlayer* MovableSound::getSoundPlayer() {
199         return soundPlayer;
200     }
```

```
#pragma once

#include "Util.h"

#include <SDL.h>
#include <SDL_ttf.h>
#include <boost/lexical_cast.hpp>

/**
 * プレーヤーの向きに応じて，音の定位や音量が変わるオブジェクトを表すクラスです．
 * 動物や虫，川，泉などはこのクラスを継承してください．
 *
 * @author ikumin
 * @date 2006/12/15
 */
class MovableSound : public SoundMaterial {

private:
    SDL_Color textColor;
    TTF_Font *font;

protected:
    double x;
    double y;
    double angle; // プレーヤーからの角度
    double distance; // プレーヤーからの距離
    int appearanceColor; // 出現時の色
    int disappearanceColor; // 消失時の色

    SDL_Surface *text;

public:
    MovableSound(string soundName, double x, double y);
    ~MovableSound(void);

    void createNameText();
    void updateLocation();
    void moveLeft();
    void moveRight();
    void moveFront();
    void moveBack();
    void moveSound();
    void fadeInColor();
    void fadeOutColor();

    // オーバーライド
    void fadeInSound();

    //getter
    double getX();
    double getY();
    SDL_Surface* getText();
    double getAngle();
    double getDistance();
    SoundPlayer* getSoundPlayer();

    //setter
    void setX(double x);
    void setY(double y);
    void setVolumeText(SDL_Surface* volumeText);
    void setAngle(double angle);
    void setDistance(double distance);
};
```

```
#include "NightSoundFactory.h"

NightSoundFactory::NightSoundFactory(Player *player) : GamePlayStateSoundFactory() {
    this->player = player;
}

NightSoundFactory::~NightSoundFactory(void) {
}

/**
 * 背景音を作成する.
 */
list<SoundMaterial> NightSoundFactory::createSoundMaterials() {
    list<SoundMaterial> soundMaterials;
    soundMaterials.push_back(*(new SoundMaterial("night")));
    return soundMaterials;
}

/**
 * 動物を作成する
 */
list<Creature> NightSoundFactory::createCreature() {
    list<Creature> creatures;
    for (int i = 0; i < 2; i++) {
        creatures.push_back(*(new Creature("wolf", getSoundX(), getSoundY())));
        creatures.push_back(*(new Creature("owl", getSoundX(), getSoundY())));
    }
    return creatures;
}

/**
 * 昆虫を作成する
 */
list<Insect> NightSoundFactory::createInsects() {
    list<Insect> insects;
    for (int i = 0; i < 5; i++) {
        insects.push_back(*(new Insect("cricket", getSoundX(), getSoundY(), 50)));
    }

    for (int i = 0; i < 4; i++) {
        insects.push_back(*(new Insect("mole_cricket", getSoundX(), getSoundY(), 40)));
    }
    return insects;
}

int NightSoundFactory::getSoundX() {
    return getRandomCoordinates() + PLAYER_X - player->getX();
}

int NightSoundFactory::getSoundY() {
    return getRandomCoordinates() + PLAYER_Y - player->getY();
}
```

```
1   #pragma once
2
3   #include "GamePlayStateSoundFactory.h"
4
5   /**
6    * 虫捕り中の夜の音を作成するクラスです.
7    *
8    * @author ikumin
9    * @date 2006/12/15
10   */
11  class NightSoundFactory : public GamePlayStateSoundFactory {
12  public:
13      NightSoundFactory(Player *player);
14      ~NightSoundFactory(void);
15
16      list<SoundMaterial> createSoundMaterials();
17      list<Creature> createCreature();
18      list<Insect> createInsects();
19
20  private:
21      Player *player;
22
23      int getSoundX();
24      int getSoundY();
25  };
```

```
#include "NightState.h"
#include "ResultState.h"

/**
 * コンストラクタ
 */
NightState::NightState(SoundContainer *soundContainer, SDL_Surface *mainScreen) : GamePlayState(soundContainer, mainScreen) {
    // 夜の音を生成する
    factory = new NightSoundFactory(player);
    soundMaterials =((NightSoundFactory *) factory)->createSoundMaterials();
    Creatures = ((NightSoundFactory *) factory)->createCreature();
    insects = ((NightSoundFactory *) factory)->createInsects();
}

/**
 * デストラクタ
 */
NightState::~NightState(void) {
}

/**
 * 夜の処理を行う(オーバーライド)
 */
void NightState::run() {
    soundPlayer->play("crow");
    player->speak("get_dark");
    GamePlayState::run();
}

/**
 * 音を描画する
 */
void NightState::draw() {
    if (debugMode == ON) {
        GamePlayState::draw();

        string stateText = "NIGHT";
        text = TTF_RenderText_Blended(font, stateText.c_str(), textColor);
        drawText(10, 10, text, mainScreen); // 時間帯の描画
        SDL_FreeSurface(text);
    }
}

/**
 * 時間を経過させる.
 * 一定時間が経過すると, ゲームが終了する.
 */
void NightState::passTime() {
    GamePlayState::passTime();

    // ゲーム終了が近づいていることを知らせる
    if (passingTime == limitTime - FADE_TIME) {
        player->speak("walk_home");
    }

    // ゲームを終了させる
    if (passingTime == limitTime) {
        finalize();
        changeState(new ResultState(soundContainer, mainScreen, player));
        gameState = GO_RESULT;
    }
}
```



```

/**
 * 音とテキストをフェードアウトさせる
 */
void NightState::fadeOut() {
    for (list<SoundMaterial>::iterator i = soundMaterials.begin(); i != soundMaterials.end(); i++) {
        i->setFadeOut(true);
    }
    for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
        i->fadeOutColor();
        i->setFadeOut(true);
    }
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->fadeOutColor();
        i->setFadeOut(true);
    }
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->fadeOutColor();
        i->setFadeOut(true);
    }
    for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
        i->fadeOutColor();
        i->setFadeOut(true);
    }
}

/**
 * 音とテキストをフェードインさせる
 */
void NightState::fadeIn() {
    for (list<SoundMaterial>::iterator i = soundMaterials.begin(); i != soundMaterials.end(); i++) {
        i->setFadeIn(true);
    }
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->fadeInColor();
        i->setFadeIn(true);
    }
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->fadeInColor();
        i->setFadeIn(true);
    }
}

/**
 * 音とテキストのフェードインを止める
 */
void NightState::stopFadeIn() {
    for (list<SoundMaterial>::iterator i = soundMaterials.begin(); i != soundMaterials.end(); i++) {
        i->setFadeIn(false);
    }
    for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
        i->setFadeIn(false);
    }
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        i->setFadeIn(false);
    }
}

void NightState::finalize() {
    State::finalize();

    // 背景音を止める
    for (list<SoundMaterial>::iterator i = soundMaterials.begin(); i != soundMaterials.end(); i++) {
```

```
126             i->stopSound();
127         }
128
129         // 自然物の音を止める
130         for (list<MovableSound>::iterator i = movableSounds.begin(); i != movableSounds.end(); i++) {
131             i->stopSound();
132         }
133
134         // 動物の音を止める
135         for (list<Creature>::iterator i = Creatures.begin(); i != Creatures.end(); i++) {
136             i->stopSound();
137         }
138
139         // 昆虫の音を止める
140         for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
141             i->stopSound();
142         }
143
144         // 川の音を止める
145         for (list<River>::iterator i = rivers.begin(); i != rivers.end(); i++) {
146             i->stopSound();
147         }
148
149         player->stopCatchedInsects();
150     }
```

```
#pragma once

#include "NightSoundFactory.h"

/**
 * 夜を表すクラス
 *
 * @author hashiyaman
 * @date 2007/1/16
 */
class NightState : public GamePlayState {
protected:
    // オーバーライド
    void draw();
    void passTime();
    void fadeOut();
    void fadeIn();
    void stopFadeIn();
    void finalize();

public:
    NightState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
    ~NightState();

    // オーバーライド
    void run();
};
```

```
#include "OnMapObject.h"

const double OnMapObject::M_PI = 3.14159265358979323846264338327 9;

/*
 * コンストラクタ
 */
OnMapObject::OnMapObject(float x, float y, SoundContainer *soundContainer, string soundListPath)
{
    this->soundPlayer = new SoundPlayer(soundContainer,soundListPath);
    this->initObjectStatus(x,y);
    initializeStatus = objectStatus;
}

/*
 * 初期化する
 */
void OnMapObject::initObjectStatus(float x, float y){
    this->objectStatus.x = x;
    this->objectStatus.y = y;
    this->objectStatus.state = ALIVE;
    this->objectStatus.width = 10;
    this->objectStatus.height = 10;
}

/*
 * オブジェクト間の角度を計算する
 */
Float32 OnMapObject::calculateInterObjectAngle(const ObjectStatus *subject, const ObjectStatus *target)
{
    // オブジェクト間の角度を計算する
    float radian = atan2( subject->y - target->y, subject->x - target->x);

    // 弧度法から度数法へ変換
    // CRIAudioでは正面が0度なので、画面と音の整合性を取るため角度を足す
    int angle = static_cast<int>( ( radian / M_PI * 180 ) + 90 );
    angle = angle % 360;

    return static_cast<Float32>(-angle);

}

/*
 * スピーカーのセンドレベルを計算する
 */
CriAuSendLevel OnMapObject::calculateSpeakerSendLevel(const ObjectStatus *subject, const ObjectStatus
target)
{
    // オブジェクト間の角度を計算する
    Float32 angle = this->calculateInterObjectAngle(subject, target);

    // 角度からセンドレベルを計算する
    Float32 left;
    Float32 right;
    Float32 leftSurround;
    Float32 rightSurround;
    Float32 center;
    CriAuUty::CalcSendLevel5Speakers(angle, &left, &right, &leftSurround, &rightSurround, &center);

    // センドレベルを設定する
    CriAuSendLevel sendLevel;
    sendLevel.SetLeft(left);
    sendLevel.SetRight(right);
```



```
    sendLevel.SetLeftSurround(leftSurround);
    sendLevel.SetRightSurround(rightSurround);
    sendLevel.SetCenter(center);

    return sendLevel;
}

/*
 * オブジェクトの音を鳴らす
 */
void OnMapObject::playSound(string soundName, const CriAuSendLevel &sendLevel)
{
      soundPlayer->setSendLevel(soundName, sendLevel);
    soundPlayer->play(soundName);
}

/*
 * デストラクタ
 */
OnMapObject::~OnMapObject()
{
    delete soundPlayer;
}
```

```
#ifndef _ONMAPOBJECT_H_
#define _ONMAPOBJECT_H_

#include <SDL.h>
#include <windows.h>
#include "SoundPlayer.h"
#include "cri_audio.h"

enum ObjectCondition
{
    ALIVE,
    DEAD
};

struct ObjectStatus
{
    ObjectStatus(){};
    ObjectStatus(float x, float y, int state,    int width, int height):
    x(x), y(y), state(state), width(width), height(height){};
    float x;
    float y;
    int state;
    int width;
    int height;
};

class OnMapObject
{
protected:
    static const double M_PI;

    ObjectStatus objectStatus;
    ObjectStatus initializeStatus;
    ObjectStatus *playerStatus;

    SoundPlayer* soundPlayer;

    // subjectから見たtargetの位置に応じて各スピーカーへのセンドレベルを計算する
    CriAuSendLevel calculateSpeakerSendLevel(const ObjectStatus *subject, const ObjectStatus *target);

    // subjectとtarget間の角度を計算する
    Float32 calculateInterObjectAngle(const ObjectStatus *subject, const ObjectStatus *target);
    void playSound(string soundName, const CriAuSendLevel &sendLevel);

    // 距離に応じたボリュームを計算する
    virtual Float32 calculateVolume(const ObjectStatus *subject, const ObjectStatus *target) = 0;

    //初期化する
    void initObjectStatus(float x, float y);

public:

    void setTarget(ObjectStatus *target){playerStatus = target;};
    void reset(){ objectStatus = initializeStatus; };
    ObjectStatus* getObjectStatus(){return &objectStatus;};

    virtual void move(Uint8 *keys) = 0;
    virtual void resolveCollision() = 0;
    virtual void draw(SDL_Surface *targetScreen) = 0;
    OnMapObject(float x, float y, SoundContainer *soundContainer, string soundListPath, int *time);
    virtual ~OnMapObject();
};
```



```
#endif
```

```
#include "Player.h"
#include "Game.h"

static SDL_Surface *text;

/**
 * コンストラクタ
 */
Player::Player() {
    x = PLAYER_X;
    y = PLAYER_Y;
    angle = 0;
    steppingFoot = STEP_RIGHT;
    walkingPlace = FOREST;
    isSwinging = false;
    cage = new Cage();
    soundPlayer = new SoundPlayer(Game::getInstance()->getSoundContainer(), GAME_SOUND_PATH);
}

/**
 * デストラクタ
 */
Player::~Player(void) {
    delete cage;
    delete soundPlayer;
}

/**
 * 前進する
 */
void Player::walkFront() {
    x += PLAYER_SPEED * cos(Util::angleToRadian(angle));
    y += PLAYER_SPEED * sin(Util::angleToRadian(angle));
    step();
}

/**
 * 後退する
 */
void Player::walkBack() {
    x -= PLAYER_SPEED * cos(Util::angleToRadian(angle));
    y -= PLAYER_SPEED * sin(Util::angleToRadian(angle));
    step();
}

/**
 * 左を向く
 */
void Player::turnLeft() {
    angle += PLAYER_SPEED;
    if (angle >= 360) {
        angle = 0;
    }

    // 場所に応じて足音を鳴らす
    if (walkingPlace == FOREST) {
        if (soundPlayer->getStatus("turn_leaves") != CriAuPlayer::STATUS_PLAYING) {
            soundPlayer->play("turn_leaves");
        }
    } else {
        if (soundPlayer->getStatus("turn_water") != CriAuPlayer::STATUS_PLAYING) {
            soundPlayer->play("turn_water");
        }
    }
}
```



```
}

/**
 * 右を向く
 */
void Player::turnRight() {
    angle -= PLAYER_SPEED;
    if (angle <= 0) {
        angle = 360;
    }

    // 場所に応じて足音を鳴らす
    if (walkingPlace == FOREST) {
        if (soundPlayer->getStatus("turn_leaves") != CriAuPlayer::STATUS_PLAYING) {
            soundPlayer->play("turn_leaves");
        }
    } else {
        if (soundPlayer->getStatus("turn_water") != CriAuPlayer::STATUS_PLAYING) {
            soundPlayer->play("turn_water");
        }
    }
}

/**
 * 場所に応じて足音を変える
 */
void Player::step() {
    if (walkingPlace == FOREST
        && soundPlayer->getStatus("step_water_left") != CriAuPlayer::STATUS_PLAYING
        && soundPlayer->getStatus("step_water_right") != CriAuPlayer::STATUS_PLAYING) {
        stepForest();
    } else if (walkingPlace == RIVER
        && soundPlayer->getStatus("step_leaves_left") != CriAuPlayer::STATUS_PLAYING
        && soundPlayer->getStatus("step_leaves_right") != CriAuPlayer::STATUS_PLAYING) {
        stepRiver();
    }
}

/**
 * 森の中を歩く
 */
void Player::stepForest() {
    // 左足で歩く
    if (steppingFoot == STEP_LEFT && soundPlayer->getStatus("step_leaves_right") != CriAuPlayer::STATUS_PLAYING) {
        soundPlayer->play("step_leaves_left");
        steppingFoot = STEP_RIGHT;

    // 右足で歩く
    } else if (steppingFoot == STEP_RIGHT && soundPlayer->getStatus("step_leaves_left") != CriAuPlayer::STATUS_PLAYING) {
        soundPlayer->play("step_leaves_right");
        steppingFoot = STEP_LEFT;
    }
}

/**
 * 川を渡る
 */
void Player::stepRiver() {
    // 左足で歩く
    if (steppingFoot == STEP_LEFT && soundPlayer->getStatus("step_water_right") != CriAuPlayer::STATUS_P
```

```
AYING) {
125             soundPlayer->play("step_water_left");
126             steppingFoot = STEP_RIGHT;
127
128         // 右足で歩く
129         } else if (steppingFoot == STEP_RIGHT && soundPlayer->getStatus("step_water_left") != CriAuPlayer::STA
    US_PLAYING) {
130             soundPlayer->play("step_water_right");
131             steppingFoot = STEP_LEFT;
132         }
133     }
134
135     /**
136      * セリフを話す
137      */
138     void Player::speak(string words) {
139         soundPlayer->play(words);
140     }
141
142     /**
143      * 網を振る
144      */
145     void Player::swingNet() {
146         soundPlayer->play("swing");
147         soundPlayer->play("yell1");
148     }
149
150     void Player::stopSwing() {
151         soundPlayer->stop("swing");
152     }
153
154     /**
155      * 虫を捕まえたときの処理を行う
156      */
157     void Player::catchInsect(Insect* insect) {
158         if (insect->isCatchable()) {
159             // 捕まえたことを知らせる
160             string catchedSoundName = "catch_" + insect->getName();
161             if (soundPlayer->getStatus(catchedSoundName) != CriAuPlayer::STATUS_PLAYING) {
162                 soundPlayer->play(catchedSoundName);
163             }
164
165             // 虫を虫かごへ入れる
166             cage->putInsect(insect);
167
168         } else {
169             // 逃げられたことを知らせる
170             int id = rand() % 3 + 1;
171             string failureVoice = "fail_in_catch" + boost::lexical_cast<string>(id);;
172             soundPlayer->play(failureVoice);
173         }
174     }
175
176     /**
177      * 捕まえた虫が鳴く
178      * (逆に分かりにくくなるので，現在未使用)
179      */
180     void Player::playCatchedInsects() {
181         //cage->playCatchedInsects();
182     }
183
184     void Player::stopCatchedInsects() {
185         //cage->stopCatchedInsects();
```



```
186     }
187
188     /***********************************************
189      *             Getter & Setter                 *
190      ***********************************************/
191
192     double Player::getX() {
193         return x;
194     }
195
196     void Player::setX(double x) {
197         this->x = x;
198     }
199
200     double Player::getY() {
201         return y;
202     }
203
204     void Player::setY(double y) {
205         this->y = y;
206     }
207
208     double Player::getAngle() {
209         return angle;
210     }
211
212     int Player::getSteppingFoot() {
213         return steppingFoot;
214     }
215
216     void Player::setSteppingFoot(int steppingFoot) {
217         this->steppingFoot = steppingFoot;
218     }
219
220     bool Player::getIsSwinging() {
221         return isSwinging;
222     }
223
224     void Player::setSwinging(bool isSwinging) {
225         this->isSwinging = isSwinging;
226     }
227
228     SDL_Surface* Player::getText() {
229         if (text == NULL) {
230             SDL_Color textColor = {255, 255, 255};
231             TTF_Font *font = TTF_OpenFont("Headache.ttf", 28);
232             string playerText = "P";
233             text = TTF_RenderText_Blended(font, playerText.c_str(), textColor);
234             TTF_CloseFont(font);
235         }
236         return text;
237     }
238
239     int Player::getWalkingPlace() {
240         return walkingPlace;
241     }
242
243     void Player::setWalkingPlace(int walkingPlace) {
244         this->walkingPlace = walkingPlace;
245     }
246
247     Cage* Player::getCage() {
248         return cage;
```

```
249     }
250
251     SoundPlayer* Player::getSoundPlayer() {
252         return soundPlayer;
253     }
```

```
#pragma once

#include <SDL.h>
#include <SDL_ttf.h>

#include "SoundMaterial.h"
#include "Cage.h"

/**
 * プレーヤーを表すクラスです.
 *
 * @author ikumin
 * @version 2006/12/15 hashiyaman
 */
class Player {
private:
    double x;
    double y;
    double angle; // プレーヤーが向いている角度
    int steppingFoot; // 踏み出している足
    int walkingPlace; // 歩いている場所
    bool isSwinging; // 棒を振っているかどうか
    Cage *cage;
    SoundPlayer *soundPlayer;
    list<SoundMaterial> soundMaterials;

public:
    static const int SPEED;

    Player();
    ~Player(void);

    void walkFront();
    void walkBack();
    void step();
    void stepForest();
    void stepRiver();
    void turnLeft();
    void turnRight();
    void swingNet();
    void stopSwing();
    void speak(string words);
    void catchInsect(Insect* insect);
    void playCatchedInsects();
    void stopCatchedInsects();

    // getter
    double getX();
    double getY();
    double getAngle();
    int getSteppingFoot();
    int getWalkingPlace();
    bool getIsSwinging();
    static SDL_Surface* getText();
    Cage* getCage();
    SoundPlayer* getSoundPlayer();

    // setter
    void setX(double x);
    void setY(double y);
    void setSteppingFoot(int steppingFoot);
    void setSwinging(bool isStick);
    void setWalkingPlace(int walkingPlace);
```



```
};
```

```
#include "ResultState.h"
#include "TitleState.h"

/**
 * コンストラクタ
 */
ResultState::ResultState(SoundContainer *soundContainer, SDL_Surface *mainScreen, Player *player) : State
) {
    this->soundPlayer = new SoundPlayer(soundContainer, RESULT_SOUND_PATH);
    this->soundContainer = soundContainer;
    this->mainScreen = mainScreen;
    this->player = player;

    // 状態の初期化
    readingState = NUMBER_OF_INSECTS;
    //readingState = PRE_TAG_OF_SCORE;
    soundPlayerState = NOT_READING;
    readingDigit = 0;
    readingInsect = -1;
    //readingInsect = NULL;

    // テキストを初期化する
    textColor.r = 255;
    textColor.g = 255;
    textColor.b = 255;
    font = TTF_OpenFont("Headache.ttf", 28);
}

/**
 * デストラクタ
 */
ResultState::~ResultState() {
    delete soundPlayer;
}

/*
 * ゲーム中の処理を行う
 */
void ResultState::run() {
    soundPlayer->play("result");

    while(gameState != END_OF_TITLE) {
        processKeyEvent();
        playSounds();
        draw();
        update();
    }
    finalize();
    changeState(new TitleState(soundContainer, mainScreen));
}

/**
 * キーイベントリスナー
 */
void ResultState::processKeyEvent() {
    State::processKeyEvent();

    // タイトルに戻る
    if(Util::isPressedOnce(SDLK_RETURN)) {
        gameState = END_OF_TITLE;
    }
}
```



```
/**
 * 結果画面の音を鳴らす(オーバーライド)
 */
void ResultState::playSounds() {
    if (soundPlayer->getStatus("result") != CriAuPlayer::STATUS_PLAYING) {
        readResult();
    }
}

/**
 * 結果を読み上げる
 */
void ResultState::readResult() {
    switch (readingState) {
        case NUMBER_OF_INSECTS:
            readNumberOfInsects();
            break;

        case PRE_TAG_OF_SCORE:
            readPreTagOfScore();
            break;

        case NUMBER_OF_SCORE:
            readNumberOfScore();
            break;

        case POST_TAG_OF_SCORE:
            readPostTagOfScore();
            break;

        case GO_TITLE:
            readGoTitle();
            break;
    }
}

/**
 * 捕まえた虫の名前を数を読む
 */
void ResultState::readNumberOfInsects() {
    list<Insect> catchedInsects = player->getCage()->getCatchedInsects();
    list<Insect> unifiedInsects = unifyInsects(catchedInsects);

    // 捕まえた虫の名前を配列に取り込む
    if (readingInsect == -1) {
        int index = 0;
        for (list<Insect>::iterator i = unifiedInsects.begin(); i != unifiedInsects.end(); i++) {
            insectNames[index++] = i->getName();
        }
        readingInsect = 0;
    }

    // 虫ごとに，名前と捕まえた数を読み上げる
    if (readingInsect < unifiedInsects.size()) {
        string soundName = "number_of_" + insectNames[readingInsect] + "s"; // 捕まえた昆虫の名前
        int numberOfInsects = getNumberOfInsects(catchedInsects, insectNames[readingInsect]); // 捕まえた昆
の数
        string soundScore = boost::lexical_cast<string>(numberOfInsects);

        if (soundPlayer->getStatus(soundName) != CriAuPlayer::STATUS_PLAYING && soundPlayerState == NO
_READING) {
            soundPlayer->play(soundName);
            soundPlayerState = READING_NAME_OF_INSECT;
```

```
124         } else if (soundPlayer->getStatus(soundName) != CriAuPlayer::STATUS_PLAYING && soundPlayer->get
    tatus(soundScore) != CriAuPlayer::STATUS_PLAYING && soundPlayerState == READING_NAME_OF_INSECT) {
125             soundPlayer->play(soundScore);
126             soundPlayerState = READING_NUMBER_OF_INSECTS;
127         } else if (soundPlayer->getStatus(soundScore) != CriAuPlayer::STATUS_PLAYING && soundPlayerState
    = READING_NUMBER_OF_INSECTS) {
128             readingInsect++; // 次の虫に行く
129             soundPlayerState = NOT_READING;
130         }
131     } else {
132         readingState = PRE_TAG_OF_SCORE;
133         readingInsect = NULL; // 虫の参照を初期化する
134     }
135 }
136
137 /**
138  * スコアの冒頭部分を読む
139  */
140 void ResultState::readPreTagOfScore() {
141     if (soundPlayer->getStatus("pre_tag_of_score") != CriAuPlayer::STATUS_PLAYING && soundPlayerState ==
    NOT_READING) {
142         soundPlayer->play("pre_tag_of_score");
143         soundPlayerState = READING_SCORE;
144     } else if (soundPlayer->getStatus("pre_tag_of_score") != CriAuPlayer::STATUS_PLAYING && soundPlayerSt
    te == READING_SCORE) {
145         readingState = NUMBER_OF_SCORE;
146         soundPlayerState = NOT_READING;
147     }
148 }
149
150 /**
151  * スコアの数字部分を読む
152  */
153 void ResultState::readNumberOfScore() {
154     int score = calculateScore(player->getCage()->getCatchedInsects());
155     string scoreText = boost::lexical_cast<string>(score);
156
157     // スコアを桁ごとにばらして，読み上げる
158     if (readingDigit < scoreText.size()) {
159         string digit = scoreText.substr(readingDigit, 1);
160         if (soundPlayer->getStatus(digit) != CriAuPlayer::STATUS_PLAYING && soundPlayerState == NOT_REA
    ING) {
161             soundPlayer->play(digit);
162             soundPlayerState = READING_SCORE;
163         } else if (soundPlayer->getStatus(digit) != CriAuPlayer::STATUS_PLAYING && soundPlayerState == REA
    ING_SCORE) {
164             readingDigit++; // 次の桁へ行く
165             soundPlayerState = NOT_READING;
166         }
167     } else { // 数字を読み終えたら
168         readingState = POST_TAG_OF_SCORE;
169         readingDigit = 0; // 桁数を初期化する
170     }
171 }
172
173 /**
174  * スコアの末尾部分を読む
175  */
176 void ResultState::readPostTagOfScore() {
177     if (soundPlayer->getStatus("post_tag_of_score") != CriAuPlayer::STATUS_PLAYING && soundPlayerState =
     NOT_READING) {
178         soundPlayer->play("post_tag_of_score");
179         soundPlayerState = READING_SCORE;
```



```
180     } else if (soundPlayer->getStatus("post_tag_of_score") != CriAuPlayer::STATUS_PLAYING && soundPlayerS
    ate == READING_SCORE) {
181         //readingState = PRE_TAG_OF_SCORE;
182         readingState = GO_TITLE;
183         soundPlayerState = NOT_READING;
184     }
185 }
186
187 /**
188  * タイトルに戻ることを知らせる
189  */
190 void ResultState::readGoTitle() {
191     if (soundPlayer->getStatus("go_title") != CriAuPlayer::STATUS_PLAYING && soundPlayerState == NOT_RE
    DING) {
192         soundPlayer->play("go_title");
193         soundPlayerState = READING_SCORE;
194     } else if (soundPlayer->getStatus("go_title") != CriAuPlayer::STATUS_PLAYING && soundPlayerState == R
    EADING_SCORE) {
195         readingState = NUMBER_OF_INSECTS;
196         //readingState = PRE_TAG_OF_SCORE;
197         soundPlayerState = NOT_READING;
198     }
199 }
200
201 /**
202  * 描画を行う(デバッグ用)
203  */
204 void ResultState::draw() {
205     if (debugMode == ON) {
206         string stateText = "Result";
207         text = TTF_RenderText_Blended(font, stateText.c_str(), textColor);
208         drawText(10, 10, text, mainScreen); // 文字の描画
209         SDL_FreeSurface(text);
210
211         drawResult();
212     }
213 }
214
215 /**
216  * 結果の表示を行う(デバッグ用)
217  */
218 void ResultState::drawResult() {
219     list<Insect> catchedInsects = player->getCage()->getCatchedInsects();
220     list<Insect> unifiedInsects = unifyInsects(catchedInsects);
221
222     drawCatchedInsectsName(unifiedInsects);
223     drawNumberOfCatchedInsects(catchedInsects, unifiedInsects);
224
225     drawScore(catchedInsects);
226 }
227
228 /**
229  * 捕らえた昆虫の名前を表示する(デバッグ用)
230  */
231 void ResultState::drawCatchedInsectsName(list<Insect> insects) {
232     int index = 1;
233     for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
234         string insectName = i->getName();
235         text = TTF_RenderText_Blended(font, insectName.c_str(), textColor);
236         drawText(10, (30 * index) + 10, text, mainScreen); // 名前の描画
237         SDL_FreeSurface(text);
238         index++;
239     }
```

```
}

/**
 * 捕らえた昆虫の数を表示する(デバッグ用)
 */
void ResultState::drawNumberOfCatchedInsects(list<Insect> catchedInsects, list<Insect> unifiedInsects) {
    int index = 1;
    for (list<Insect>::iterator i = unifiedInsects.begin(); i != unifiedInsects.end(); i++) {
        int numberOfInsects = getNumberOfInsects(catchedInsects, i->getName());
        string insectName = boost::lexical_cast<string>(numberOfInsects);
        text = TTF_RenderText_Blended(font, insectName.c_str(), textColor);
        drawText(150, (30 * index) + 10, text, mainScreen); // 数の描画
        SDL_FreeSurface(text);
        index++;
    }
}

/**
 * 同じ名前の昆虫を一緒にする
 */
list<Insect> ResultState::unifyInsects(list<Insect> insects) {
    list<Insect> unifiedInsects; // 同じ名前の昆虫をまとめたリスト

    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        if (!existsSameInsect(unifiedInsects, i->getName())) {
            unifiedInsects.push_back(*i);
        }
    }
    return unifiedInsects;
}

/**
 * 同じ種類の昆虫の数を調べる
 */
int ResultState::getNumberOfInsects(list<Insect> insects, string name) {
    int numberOfInsects = 0;
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        if (i->getName() == name) {
            numberOfInsects++;
        }
    }
    return numberOfInsects;
}

/**
 * 同じ名前の昆虫を捕まえているかを調べる
 */
bool ResultState::existsSameInsect(list<Insect> insects, string name) {
    bool exists = false;
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        if (i->getName() == name) { // 既に同じ名前の昆虫がいたら
            exists = true;
            break;
        }
    }
    return exists;
}

/**
 * スコアを表示する(デバッグ用)
 */
void ResultState::drawScore(list<Insect> insects) {
    int score = calculateScore(insects);
```



```
    string scoreText = boost::lexical_cast<string>(score);
    text = TTF_RenderText_Blended(font, ("SCORE: " + scoreText).c_str(), textColor);
    drawText(10, 250, text, mainScreen); // スコアの描画
    SDL_FreeSurface(text);
}

/**
 * スコアを計算する
 */
int ResultState::calculateScore(list<Insect> insects) {
    int score = 0;
    for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
        score += i->getPoint();
    }
    return score;
}
```

```
#ifndef _RESULTSTATE_H_
#define _RESULTSTATE_H_

#pragma once

#include "State.h"
//#include "Game.h"

#include <list>

using std::string;

/**
 * ゲームの結果を知らせるクラスです.
 * 取った虫の通知をします.
 *
 * @author hashiyaman
 * @version 2006/1/6
 */

class ResultState : public State {

private:
    Player *player;

    int gameState;
    int readingState; // 結果発表で何を読み上げているか
    int soundPlayerState; // CriAuPlayerのStatusで対応できない部分をカバーする
    int readingDigit; // 読んでいる数字が何桁目か
    int readingInsect;
    string insectNames[10];

    // 音声による結果発表
    void readResult();
    void readNumberOfInsects();
    void readPreTagOfScore();
    void readNumberOfScore();
    void readPostTagOfScore();
    void readGoTitle();

    // 文字による結果発表
    void drawResult();
    void drawCatchedInsectsName(list<Insect> insects);
    void drawNumberOfCatchedInsects(list<Insect> catchedInsects, list<Insect> unifiedInsects);
    void drawScore(list<Insect> insects);

    list<Insect> unifyInsects(list<Insect> insects);
    int getNumberOfInsects(list<Insect> insects, string name);
    bool existsSameInsect(list<Insect> insects, string name);
    int calculateScore(list<Insect> insects);

public:
    // オーバーライド
    void processKeyEvent();
    void playSounds();
    void draw();

    void run();
    ResultState(SoundContainer *soundContainer, SDL_Surface *mainScreen, Player *player);
    ~ResultState();
};

#endif
```

```
1   #include "River.h"
2   #include "Game.h"
3
4   /**
5    * 川を表すクラスです.
6    * 現在は線状の川を表すために, 複数の音を鳴らしています.
7    *
8    * @author ikumin
9    * @version 2006/12/15
10   */
11
12  /**
13   * コンストラクタ
14   */
15  River::River(double x, double y) : MovableSound("river", x, y) {
16  }
17
18  /**
19   * デストラクタ
20   */
21  River::~River(void) {
22  }
23
24  /**
25   * 川を渡っているかどうか
26   */
27  bool River::isCrossingRiver() {
28      double distanceX = this->x - PLAYER_X;
29      double distanceY = this->y - PLAYER_Y;
30      return (distanceX > -20 && distanceX < 20) && (distanceY > -20 && distanceY < 20);
31  }
```

```
1       #pragma once
2
3       #include "movablesound.h"
4       #include "player.h"
5
6       class River : public MovableSound {
7       public:
8           River(double x, double y);
9           ~River(void);
10
11          bool isCrossingRiver();
12      };
```

```
#pragma once

#include <SDL.h>

#include "Constants.h"

namespace {
    bool PollEvent() {
        SDL_Event ev;
        while(SDL_PollEvent(&ev)) {
            switch(ev.type) {
            case SDL_QUIT:// ウィンドウの×ボタンが押された時など
                return false;
                break;
            }
        }
        return true;
    }

    void ClearScreen(SDL_Surface *target) {
        // サーフェスを黒で初期化
        SDL_Rect dest;
        dest.x = 0;
        dest.y = 0;
        dest.w = WINDOW_WIDTH;
        dest.h = WINDOW_HEIGHT;
        Uint32 color= 0x00000000;
        SDL_FillRect( target, &dest, color );
    }
}
```

```
1       #include "Sound.h"
2
3       /*
4        * コンストラクタ
5        */
6       Sound::Sound(string soundName, float x, float y) {
7           this->soundName = soundName;
8           this->x = x;
9           this->y = y;
10      }
11
12      float getX() {
13          //return x;
14          return 0;
15      }
16
17      float getY() {
18          //return y;
19          return 0;
20      }
21
22      /*
23       * デストラクタ
24       */
25      Sound::~Sound() {
26      }
```

```
1   #ifndef _GAMEPLAYSTATE_H_
2   #define _GAMEPLAYSTATE_H_
3
4   #include "SoundPlayer.h"
5   #include "KeyState.h"
6   #include "StateMessages.h"
7
8   #include <SDL.h>
9   #include <SDL_ttf.h>
10  #include <string>
11  #include <list>
12  #include <boost/bind.hpp>
13  #include <boost/lexical_cast.hpp>
14  #include <algorithm>
15  #include <cstdlib>
16
17  #include "SDLCommonFunctions.h"
18  using std::string;
19
20  enum ObjectCondition {
21      ALIVE, DEAD
22  };
23
24  class Sound {
25      string soundName;
26      float x;
27      float y;
28
29      float getX();
30      float getY();
31
32      public:
33          Sound(string soundName, float x, float y);
34          ~Sound();
35  };
36  #endif
```

```
#include "SoundContainer.h"

/**
 * コンストラクタ
 */
SoundContainer::SoundContainer(string cuePath) {
    CriError error = CRIERR_OK; // エラー検出用のオブジェクト

    // 音声出力を初期化する
    soundOut =  CriSmpSoundOutput::Create();
    heap = criHeap_Create(buf, sizeof(buf));
    soundRenderer = CriSoundRendererBasic::Create(heap, error);
    soundOut->SetNotifyCallback(soundOutCallBack, static_cast<void *>(soundRenderer));
    soundOut->Start();

    // csbファイルを読み込む
    Uint8 *csbdata;
    unsigned long csbsize;
    csbdata = this->loadCue(cuePath, &csbsize);
    cueSheet = CriAuCueSheet::Create(heap, error);
    cueSheet->LoadCueSheetBinaryFileFromMemory(csbdata, csbsize, error);
    free(csbdata);

    // 音声オブジェクトを生成し，キューシートを登録する
    audioObject = CriAuObj::Create(heap, soundRenderer, "ForestWalking", error);
    audioObject->AttachCueSheet(cueSheet, error);

    // エラー処理
    if (error != CRIERR_OK) {
        exit(1);
    }
}

/*
 * サウンドレンダラで生成されたサウンドデータを取得する
 *
 * Getting PCM Data from CRI Sound Renderer
 * numberOfSamples has to be 128*N samples. (N=1,2,3...)
 */
unsigned long SoundContainer::soundOutCallBack(void *obj, unsigned long numberOfChannels, Float32 *sampl
[], unsigned long numberOfSamples) {
    CriSoundRendererBasic* soundRenderer = (CriSoundRendererBasic*) obj;
    CriError error;

    soundRenderer->GetData(numberOfChannels, numberOfSamples, sample, error);
    return numberOfSamples;
}

/*
 * キューファイルをロードする
 */
Uint8* SoundContainer::loadCue(string path, unsigned long *idtsize) {
    FILE *fp;
    signed long size;
    Uint8 *idt;

    fp = fopen(path.c_str(), "rb");
    fseek(fp, 0, SEEK_END);
    size = ftell(fp);
    idt = (Uint8*)calloc(size, 1);
    fseek(fp, 0, SEEK_SET);
    fread(idt, size, 1, fp);
    fclose(fp);
```



```

    *idtsize = (unsigned long)size;
    return idt;
}

/*
 * 音をロードする（音声プレーヤーを生成する）
 */
CriAuPlayer* SoundContainer::loadSound(string soundName) {
    CriError error = CRIERR_OK;
    CriAuPlayer* player = CriAuPlayer::Create(audioObject, error);
    player->SetCue(soundName.c_str(), error);

    return player;
}

/**
 * デストラクタ
 */
SoundContainer::~SoundContainer() {
    CriError error = CRIERR_OK;

    audioObject->Destroy(error);
    cueSheet->Destroy(error);
    soundOut->SetNotifyCallback(NULL, NULL);
    soundRenderer->Destroy(error);

    //    Print Heap Status
    //criHeap_DebugPrintBlockInformationAll(heap);
    criHeap_Destroy(heap);
}
```

```
1   #ifndef _SOUNDCONTAINER_H_
2   #define _SOUNDCONTAINER_H_
3
4   #include <string>
5   #include <iostream>
6   #include "cri_audio.h"
7   #include "cri_xpt.h"
8   #include "CriSmpSoundOutput.h"
9
10  using std::string;
11
12  /**
13   * CSBファイルの読み込みを行い, 音ファイルの準備をします.
14   *
15   * @author riho
16   */
17  class SoundContainer {
18
19  private:
20      CriHeap heap;
21      CriSoundRendererBasic* soundRenderer;
22      CriAuObj* audioObject;
23      Uint8 buf[50*1024*1024];
24      CriSmpSoundOutput *soundOut;
25      CriAuCueSheet* cueSheet;
26
27      Uint8* loadCue(string path, unsigned long *idtsize);
28      void createCueSheet();
29      static unsigned long soundOutCallBack(void *obj, unsigned long numberOfChannels, Float32 *sample[], unsi
    ned long numberOfSamples);
30
31  public:
32      ~SoundContainer();
33      SoundContainer(string cuePath);
34      CriAuPlayer* loadSound(string soundName);
35
36  };
37
38  #endif
```

```
1       #include "SoundFactory.h"
2
3       /**
4        * コンストラクタ
5        */
6       SoundFactory::SoundFactory(void) {
7       }
8
9       /**
10       * デストラクタ
11       */
12      SoundFactory::~SoundFactory(void) {
13      }
```

```
1   #pragma once
2
3   #include "soundmaterial.h"
4   #include "movablesound.h"
5   #include "river.h"
6   #include "insect.h"
7
8   #include <list>
9
10  using std::list;
11
12  /**
13   * 効果音を作成するクラスです.
14   * 各状態で効果音を作成するときは, このクラスを継承して下さい.
15   *
16   * @author ikumin
17   * @version 2006/12/15
18   */
19
20  class SoundFactory {
21  public:
22      virtual list<SoundMaterial> createSoundMaterials() = 0;
23      SoundFactory(void);
24      ~SoundFactory(void);
25  };
```

```cpp
#include "SoundMaterial.h"
#include "Game.h"

/**
 * コンストラクタ
 */
SoundMaterial::SoundMaterial(string soundName) {
    this->soundName = soundName;
    soundPlayer = new SoundPlayer(Game::getInstance()->getSoundContainer(), GAME_SOUND_PATH);
    volume = 1.0;

    isFadeOut = false;
    isFadeIn = false;

    // 変数の初期化
    increasingWidth = 0;
    decreasingWidth = 0;
}

/**
 * デストラクタ
 */
SoundMaterial::~SoundMaterial(void) {
    //delete soundPlayer;
}

/**
 * 音の名前を取得する
 */
string SoundMaterial::getName() {
    return soundName;
}

/**
 * 音を鳴らす
 */
void SoundMaterial::playSound() {
    if (isFadeOut) {
        fadeOutSound();
    } else if (isFadeIn) {
        fadeInSound();
    }
    soundPlayer->setVolume(soundName, volume);
    soundPlayer->loopPlay(soundName);
}

/**
 * 音を止める
 */
void SoundMaterial::stopSound() {
    soundPlayer->stop(soundName);
}

/**
 * 音をフェードインさせる
 */
void SoundMaterial::fadeInSound() {
    if ((increasingWidth += 0.01) <= 1) {
        increasingWidth += 0.01;
    }
    volume = increasingWidth;
}
```



```cpp
/**
 * 音をフェードアウトさせる
 */
void SoundMaterial::fadeOutSound() {
    if (volume - decreasingWidth > 0) {
        decreasingWidth += 0.015;
        volume -= decreasingWidth;
    } else {
        volume = 0;
    }
}

double SoundMaterial::getVolume() {
    return volume;
}

void SoundMaterial::setFadeOut(bool isFadeOut) {
    this->isFadeOut = isFadeOut;
}

void SoundMaterial::setFadeIn(bool isFadeIn) {
    this->isFadeIn = isFadeIn;
}
```

```
1   #pragma once
2
3   #include "SoundPlayer.h"
4   #include "StateMessages.h"
5   #include "Constants.h"
6
7   #include <string>
8
9   using std::string;
10
11  /**
12   * プレーヤーの位置が定位や音量に影響しない音（背景音）を表します.
13   *
14   * @author ikumin
15   * @date 2006/12/15
16   */
17  class SoundMaterial {
18  protected:
19      string soundName; // 音の名前
20      SoundPlayer *soundPlayer;
21      double increasingWidth; // 音量の増加幅
22      double decreasingWidth; // 音量の減少幅
23      double volume;
24      bool isFadeOut;
25      bool isFadeIn;
26
27  public:
28      SoundMaterial(string soundName);
29      ~SoundMaterial(void);
30      string getName();
31      void playSound();
32      void stopSound();
33      virtual void fadeInSound();
34      void fadeOutSound();
35
36      void setFadeOut(bool isFadeOut);
37      void setFadeIn(bool isFadeIn);
38      double getVolume();
39  };
```

```
#include "SoundPlayer.h"

/**
 * コンストラクタ
 */
SoundPlayer::SoundPlayer(SoundContainer *soundContainer, string soundListPath) {
    // 音声を取得する
    soundShelf = this->loadSound(soundContainer,soundListPath);
}

/**
 * デストラクタ
 */
SoundPlayer::~SoundPlayer() {
    soundShelf.clear();
}

SoundShelf SoundPlayer::loadSound(SoundContainer *soundContainer, string soundListPath) {
    // ファイルを開く
    std::ifstream list(soundListPath.c_str());

    // 役割ごとの音声の名前を連想配列に読み込む
    SoundShelf soundList;
    string line;
    while(getline(list, line)) {
        // 記述形式が有効でない場合はスキップする
        if (!validate(line)) {
            continue;
        }

        string::size_type splitIndex = line.find(" ", 0); // 半角で区切る

        string soundJobName = line.substr(0, splitIndex);
        string soundName = line.substr(splitIndex + 1);
        soundList[soundJobName] = soundContainer->loadSound(soundName);
    }
    list.close();

    return soundList;
}

/**
 * 記述形式の有効性を調べる
 */
bool SoundPlayer::validate(string line) {
    // コメントの場合
    string::size_type text = line.find("#", 0);
    if (text != string::npos) {
        return false;
    }

    // 役割と名前が空白で区切られてない場合
    text = line.find(" ", 0);
    if (text == string::npos) {
        return false;
    }

    return true;
}

/**
 * サウンドプレーヤーの状態を取得します.
 * 取得される状態は以下の4つです.
```



```
 *
 * CriAuPlayer::STATUS_PLAYEND(再生終了)
 * CriAuPlayer::STATUS_PLAYING(再生中)
 * CriAuPlayer::STATUS_PREP(再生準備中)
 * CriAuPlayer::STATUS_STOP(停止中)
 */
int SoundPlayer::getStatus(string soundJobName) {
    if(soundShelf.find(soundJobName) != soundShelf.end())    {
        CriError error = CRIERR_OK;
        return soundShelf[soundJobName]->GetStatus(error);
    } else {
        return -1;
    }
}

void SoundPlayer::play(string soundJobName) {
    if(soundShelf.find(soundJobName) != soundShelf.end())    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->Play(error);
    }
}

void SoundPlayer::loopPlay(string soundJobName) {
    if( soundShelf.find(soundJobName) != soundShelf.end() )    {
        CriError error = CRIERR_OK;
        int soundStatus = soundShelf[soundJobName]->GetStatus(error);
        if(soundStatus == CriAuPlayer::STATUS_STOP || soundStatus == CriAuPlayer::STATUS_PLAYEND)    {
            soundShelf[soundJobName]->Play(error);
        }
    }
}

void SoundPlayer::stop(string soundJobName) {
    if(soundShelf.find(soundJobName) != soundShelf.end())    {
        CriError error = CRIERR_OK;
        soundShelf[soundJobName]->Stop(error);
    }
}

void SoundPlayer::stopAll() {
    CriError error = CRIERR_OK;
    SoundShelfIterator itr;
    for(itr = soundShelf.begin(); itr != soundShelf.end(); itr++) {
        (*itr).second->Stop(error);
    }
}

void SoundPlayer::stopAllExcept(string excludeSoundName) {
    CriError error = CRIERR_OK;
    SoundShelfIterator itr;

    // 指定された音以外を止める
    for( itr = soundShelf.begin(); itr != soundShelf.end(); itr++ )    {
        if((*itr).first != excludeSoundName) {
            (*itr).second->Stop(error);
        }
    }
}


void SoundPlayer::setVolume(string soundJobName, double volume) {
    if(soundShelf.find(soundJobName) != soundShelf.end()) {
        CriError error = CRIERR_OK;
```

```
127             soundShelf[soundJobName]->SetVolume((Float32) volume, error);
128             soundShelf[soundJobName]->Update(error);
129         }
130     }
131
132     void SoundPlayer::setPitch(string soundJobName, float pitch) {
133         if(soundShelf.find(soundJobName) != soundShelf.end()) {
134             CriError error = CRIERR_OK;
135             soundShelf[soundJobName]->SetPitch(pitch, error);
136             soundShelf[soundJobName]->Update(error);
137         }
138     }
139
140     void SoundPlayer::setSendLevel(string soundJobName, const CriAuSendLevel &sendLevel)
141     {
142         if(soundShelf.find(soundJobName) != soundShelf.end()) {
143             CriError error = CRIERR_OK;
144             soundShelf[soundJobName]->SetDrySendLevel(sendLevel, error);
145             soundShelf[soundJobName]->Update(error);
146         }
147     }
148
149     void SoundPlayer::setReverb(string soundJobName, float reverb) {
150         if(soundShelf.find(soundJobName) != soundShelf.end()) {
151             CriError error = CRIERR_OK;
152             soundShelf[soundJobName]->SetWetSendLevel(CriAuSendLevel::WET_0, reverb, error);
153             soundShelf[soundJobName]->Update(error);
154         }
155     }
156
157     void SoundPlayer::setCutOffFrequency(string soundJobName, float lowerFrequency, float upperFrequency) {
158         if(soundShelf.find(soundJobName) != soundShelf.end()) {
159             CriError error = CRIERR_OK;
160             soundShelf[soundJobName]->SetFilterCutoffFrequency(lowerFrequency, upperFrequency, error);
161             soundShelf[soundJobName]->Update(error);
162         }
163     }
```

```
#ifndef _SOUNDPLAYER_H_
#define _SOUNDPLAYER_H_

#include "SoundContainer.h"
#include <string>
#include <fstream>
#include <map>
using std::string;

namespace {
    typedef std::map<string, CriAuPlayer*> SoundShelf;
    typedef SoundShelf::iterator SoundShelfIterator;
}

/*
 * 音声プレイヤーです. CriAuPlayerをラップしています.
 *
 * @author riho
 */
class SoundPlayer {

private:
    SoundShelf soundShelf;    // 音声を管理する連想配列
    SoundShelf loadSound(SoundContainer *soundContainer, string soundListPath);

    bool validate(string line);

public:
    SoundPlayer(SoundContainer *soundContainer, string soundListPath);
    ~SoundPlayer();
    int getStatus(string soundJobName);
    void play(string soundJobName);
    void loopPlay(string soundJobName);
    void stop(string soundJobName);
    void stopAll();
    void stopAllExcept(string excludeSoundName);
    void setVolume(string soundName, double volume);
    void setPitch(string soundName, float pitch);
    void setSendLevel(string soundName, const CriAuSendLevel &sendLevel);
    void setReverb(string soundName, float reverb);
    void setCutOffFrequency(string soundName, float lowerFrequency, float upperFrequency);
};

#endif
```

```
#include "State.h"
#include "Game.h"

/**
 * コンストラクタ
 */
State::State(void) {
    error = CRIERR_OK;

    // 設定ファイルからデバッグモードの情報を読み込む
    string value = Util::getConfigurationValue("debug");
    if (value == "ON") {
        debugMode = ON;
    } else {
        debugMode = OFF;
    }
}

/**
 * デストラクタ
 */
State::~State(void) {
    TTF_CloseFont(font);
    delete soundContainer;
    delete mainScreen;
    delete soundPlayer;
    delete factory;
}

void State::drawText(int x, int y, SDL_Surface *source, SDL_Surface *destination) {
    SDL_Rect position;
    position.x = x;
    position.y = y;
    SDL_BlitSurface(source, NULL, destination, &position);
}

/**
 * キーを監視する
 */
void State::processKeyEvent() {
    if (!PollEvent() || Util::isPressed(SDLK_ESCAPE)) {
        exit(0); // ゲームを終了する
    }
}

/**
 * 各種状態を更新する
 */
void State::update() {
    CriAuObj::ExecuteMain(error); // 音声の状態を更新する
    SDL_Flip(mainScreen); // 画面の状態を更新する
    SDL_Delay(30); // CPU使用率が100%になるのを防ぐ
    ClearScreen(mainScreen); // 画面をクリアする
}

/**
 * 状態を変える
 */
void State::changeState(State *state) {
    Game::getInstance()->changeState(state);
}

/**
```



```
 * 後処理を行う
 */
void State::finalize() {
    // 音の停止
    CriError error = CRIERR_OK;
    soundPlayer->stopAll();
    CriAuObj::ExecuteMain(error);
}

int State::getGameState() {
    return gameState;
}
```

```
#pragma once

#include "SoundPlayer.h"
#include "StateMessages.h"
#include "Util.h"
#include "SoundFactory.h"
#include "SDLCommonFunctions.h"
#include "CriSmpVideoOutput.h"
#include "Game.h"

#include <SDL.h>
#include <SDL_ttf.h>
#include <string>
#include <list>

class Game;

/**
 * ゲーム中の状態を表すクラス
 *
 * @author ikumin
 * @date 2006/12/21
 */
class State {
protected:
    SDL_Event event;
    SoundContainer *soundContainer;
    SDL_Surface *mainScreen;
    SoundPlayer *soundPlayer;
    SoundFactory *factory;
    SDL_Surface *text;
    SDL_Color textColor;
    TTF_Font *font;
    CriError error;
    int gameState; // ゲームの現在の状態を表す
    int debugMode; // デバッグモード(描画を行う)かどうか

    virtual void processKeyEvent();
    virtual void playSounds() = 0;
    virtual void draw() = 0;

    void finalize();
    void drawText(int x, int y, SDL_Surface *source, SDL_Surface *destination);
    void update();
    void changeState(State *state);

public:
    State(void);
    ~State(void);
    virtual void run() = 0;
    int getGameState();
};
```

```
#ifndef _STATEMESSAGES_H_
#define _STATEMESSAGES_H_

namespace {
    // 状態間で行き交うメッセージ
    enum StateMessage {
        QUIT_GAME,    // escが押された・ウィンドウが閉じられた時
        START_GAME, // ゲーム開始時
        GO_HOW_TO_PLAY, // 説明開始時
        START_TUTORIAL, // チュートリアル開始時
        GO_NIGHT, // ゲーム中に時間帯が変わった
        GO_RESULT, // ゲームが終了した
        END_OF_TITLE // タイトルへ遷移した
    };

    enum WalkingState {
        STEP_LEFT, // 左足を踏み出している
        STEP_RIGHT, // 右足を踏み出している
        FOREST, // 森を歩いている
        RIVER // 川を歩いている
    };

    enum ForestState {
        AFTERNOON, // 昼
        NIGHT // 夜
    };

    enum ReadingState {
        // HowToPlayStateで使用
        EXPLANATION_OF_SKIP, // スキップの説明
        EXPLANATION_OF_GOAL, // 目的の説明
        HOW_TO_PLAY, // 操作方法
        EXPLANATION_OF_INSECTS, // 虫の鳴き声の説明
        SOUND_OF_INSECT, // 虫の鳴き声を聞かせる
        END_OF_HOW_TO_PLAY, // チュートリアルへ行くことを知らせる

        // TutorialStateで使用
        EXPLANATION_OF_TUTORIAL, // チュートリアルの説明
        DO_TUTORIAL, // チュートリアルの実行
        END_OF_TUTORIAL, // チュートリアルが終わることを知らせる

        // ResultStateで使用
        NUMBER_OF_INSECTS, // 捕まえた虫と数を知らせる
        PRE_TAG_OF_SCORE, // 冒頭部分
        NUMBER_OF_SCORE, // スコアの数字部分
        POST_TAG_OF_SCORE, // 末尾部分
        GO_TITLE
    };

    enum SoundPlayerState {
        // HowToPlayState, TutorialState, ResultStateで使用
        NOT_READING, // 何も読んでいない
        READING_RULE, // ゲームルールを読み上げ中
        READING_TUTORIAL, // チュートリアルを読み上げ中
        READING_SOUND_OF_INSECT, // 虫の鳴き声読み上げ中
        READING_NAME_OF_INSECT, // 虫の名前読み上げ中
        READING_NUMBER_OF_INSECTS, // 虫の数読み上げ中
        READING_SCORE // スコア読み上げ中
    };
}

#endif
```

```
#include "TextCreator.h"

/**
 * コンストラクタ
 */
TextCreator::TextCreator(void) {
}

/**
 * デストラクタ
 */
TextCreator::~TextCreator(void) {
}

/**
 * 文字列から, テキストを作り出す.
 * テキストの色とフォントは基本的に固定です.
 */
/*
SDL_Surface* TextCreator::createText(string textString) {
    SDL_Color textColor = {255, 255, 255};
    TTF_Font *font = TTF_OpenFont("Headache.ttf", 28);
    SDL_Surface *text = TTF_RenderText_Blended(font, textString.c_str(), textColor);
    return text;
}
*/
```

```
1   #pragma once
2
3   #include <string>
4   #include <SDL.h>
5   #include <SDL_ttf.h>
6
7   using std::string;
8
9   /**
10   * 文字列から表示するためのテキストを作り出すクラスです.
11   * テキストは, デバッグ用に表示します.
12   *
13   * 現在メモリリークの原因となっているため, 使用禁止にします.
14   *
15   * @author ikumin
16   * @version 2006/12/15
17   */
18
19  class TextCreator {
20  public:
21      //static SDL_Surface* createText(string textString);
22      TextCreator(void);
23      ~TextCreator(void);
24  };
```

```
#include "TitleState.h"
#include "HowToPlayState.h"
#include "AfternoonState.h"

/*
 * コンストラクタ
 */
TitleState::TitleState(SoundContainer *soundContainer, SDL_Surface *mainScreen) : State() {
    this->soundPlayer = new SoundPlayer(soundContainer, TITLE_SOUND_PATH);
    this->mainScreen = mainScreen;
    this->soundContainer = soundContainer;

    // テキストを初期化する
    textColor.r = 255;
    textColor.g = 255;
    textColor.b = 255;
    font = TTF_OpenFont("Headache.ttf", 28);
}

/*
 * デストラクタ
 */
TitleState::~TitleState() {
    delete soundPlayer;
}

/*
 * タイトルでの処理を行う
 */
void TitleState::run() {
    soundPlayer->play("title");

    while(gameState != START_GAME && gameState != GO_HOW_TO_PLAY) {
        processKeyEvent();
        playSounds();
        draw();
        update();
    }
    finalize();

    if (gameState == START_GAME) {
        changeState(new AfternoonState(soundContainer, mainScreen));
    } else if (gameState == GO_HOW_TO_PLAY) {
        changeState(new HowToPlayState(soundContainer, mainScreen));
    }
}

/**
 * キーイベントリスナー
 */
void TitleState::processKeyEvent() {
    State::processKeyEvent();

    if (Util::isPressed(SDLK_SPACE)) { // ゲームを開始する
        gameState = START_GAME;
    } else if (Util::isPressedOnce(SDLK_RETURN)) { // 説明を開始する
        gameState = GO_HOW_TO_PLAY;
    }
}

void TitleState::playSounds() {
    if (soundPlayer->getStatus("title") != CriAuPlayer::STATUS_PLAYING) {
        soundPlayer->loopPlay("how_to_start");
```



```
    }
}

/*
 * 描画する(デバッグ用)
 */
void TitleState::draw() {
    if (debugMode == ON) {
        string stateText = "Title";
        text = TTF_RenderText_Blended(font, stateText.c_str(), textColor);
        drawText(10, 10, text, mainScreen); // 文字の描画
        SDL_FreeSurface(text);
    }
}
```

```
1   #pragma once
2
3   #include "State.h"
4   using std::string;
5
6   /**
7    * タイトルを表現するクラス
8    *
9    * @date 2006/11/05
10   * @author hashiyaman
11   */
12  class TitleState : public State{
13
14  private:
15      int gameState;
16
17  public:
18      // オーバーライド
19      void processKeyEvent();
20      void playSounds();
21      void draw();
22
23      void run();    // ゲームを実行する
24      TitleState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
25      ~TitleState();
26  };
```

```cpp
#include "TutorialSoundFactory.h"

/**
 * コンストラクタ
 */
TutorialSoundFactory::TutorialSoundFactory(void) {
}

/**
 * デストラクタ
 */
TutorialSoundFactory::~TutorialSoundFactory(void) {
}

/**
 * 背景音を作成する
 */
list<SoundMaterial> TutorialSoundFactory::createSoundMaterials() {
    list<SoundMaterial> soundMaterials;
    return soundMaterials;
}

/**
 * 動物を作成する
 */
list<Creature> TutorialSoundFactory::createCreature() {
    list<Creature> creatures;
    return creatures;
}

/**
 * 昆虫を作成する
 */
list<Insect> TutorialSoundFactory::createInsects() {
    list<Insect> insects;

    insects.push_back(*(new Insect("frog", 400, 250)));
    //insects.push_back(*(new Insect("cricket", 200, 100)));
    //insects.push_back(*(new Insect("cricket", 400, 100, 10)));
    return insects;
}
```

```
1   #pragma once
2
3   #include "GamePlayStateSoundFactory.h"
4
5   class TutorialSoundFactory : public GamePlayStateSoundFactory {
6   public:
7       TutorialSoundFactory(void);
8       ~TutorialSoundFactory(void);
9
10      list<SoundMaterial> createSoundMaterials();
11      list<Creature> createCreature();
12      list<Insect> createInsects();
13  };
```

```
#include "TutorialState.h"
#include "HowToPlayState.h"
#include "AfternoonState.h"

/**
 * コンストラクタ
 */
TutorialState::TutorialState(SoundContainer *soundContainer, SDL_Surface *mainScreen) : GamePlayState(soundContainer, mainScreen) {
    // チュートリアルの音を生成する
    factory = new TutorialSoundFactory();
    insects = ((TutorialSoundFactory *) factory)->createInsects();

    gameState = START_TUTORIAL;

    readingState = EXPLANATION_OF_TUTORIAL;
    soundPlayerState = NOT_READING;

    // プレーヤーを取得する
    player = new Player();
}

/**
 * デストラクタ
 */
TutorialState::~TutorialState(void) {
}

/**
 * チュートリアルの処理をする
 */
void TutorialState::run() {
    initialize();

    while(gameState != START_GAME && gameState != GO_HOW_TO_PLAY) {
        processKeyEvent();
        playSounds();
        draw();
        moveCreatures();
        update();
    }

    finalize();

    if (gameState == START_GAME) {
        changeState(new AfternoonState(soundContainer, mainScreen));
    } else if (gameState == GO_HOW_TO_PLAY) {
        changeState(new HowToPlayState(soundContainer, mainScreen));
    }
}

void TutorialState::processKeyEvent() {
    State::processKeyEvent();

    // 説明以外の場面でのみ，プレーヤーの操作を可能にする
    if (readingState == DO_TUTORIAL) {
        GamePlayState::processKeyEvent();
    }

    // 説明を飛ばす
    if (Util::isPressedOnce(SDLK_RETURN)) {
        skip();
    }
```



```

    // もう一度説明を行う
    if (Util::isPressed(SDLK_SPACE) && readingState == END_OF_TUTORIAL) {
        gameState = GO_HOW_TO_PLAY;
    }
}

void TutorialState::moveCreatures() {
    if (readingState == DO_TUTORIAL) {
        GamePlayState::moveCreatures();
    }
}

/**
 * 説明を飛ばす
 */
void TutorialState::skip() {
    soundPlayer->stopAll(); // 再生中の説明を終了する
    soundPlayerState = NOT_READING;

    // 次の説明に行く
    switch (readingState) {
        case END_OF_TUTORIAL:
            gameState = START_GAME;
            break;

        case DO_TUTORIAL:
            // 昆虫の音を止める
            for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
                i->stopSound();
            }
            // プレーヤーの声を止める
            player->getSoundPlayer()->stopAll();
            readingState++;
            break;

        default: // switch文で一つずつ書いたほうが分かり易いのかも
            readingState++;
    }
}

/**
 * チュートリアルを読み上げる
 */
void TutorialState::playSounds() {
    switch (readingState) {
        case EXPLANATION_OF_TUTORIAL:
            readExplanationOfTutorial();
            break;

        case DO_TUTORIAL:
            readDoTutorial();
            break;

        case END_OF_TUTORIAL:
            readEndOfTutorial();
            break;
    }
}

/**
 * チュートリアルの説明を読み上げる
 */
```

```
126 void TutorialState::readExplanationOfTutorial() {
127     if (soundPlayer->getStatus("explanation_of_tutorial") != CriAuPlayer::STATUS_PLAYING && soundPlayerSt
    te == NOT_READING) {
128         soundPlayer->play("explanation_of_tutorial");
129         soundPlayerState = READING_TUTORIAL;
130     } else if (soundPlayer->getStatus("explanation_of_tutorial") != CriAuPlayer::STATUS_PLAYING && soundPla
    erState == READING_TUTORIAL) {
131         readingState = DO_TUTORIAL;
132         soundPlayerState = NOT_READING;
133     }
134 }
135
136 /**
137  * チュートリアル実行時の音声を鳴らす
138  */
139 void TutorialState::readDoTutorial() {
140     if (!isClear()) {
141         // 昆虫の音を鳴らす
142         for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
143             i->playSound();
144         }
145
146         // 虫かごに捕らえている昆虫の音を鳴らす
147         player->playCatchedInsects();
148     } else if (player->getSoundPlayer()->getStatus("catch_" + lastCatchedInsectName) != CriAuPlayer::STATU
    _PLAYING) {
149         readingState = END_OF_TUTORIAL;
150     }
151 }
152
153 /**
154  * チュートリアルが終わることを読み上げる
155  */
156 void TutorialState::readEndOfTutorial() {
157     if (soundPlayer->getStatus("end_of_tutorial") != CriAuPlayer::STATUS_PLAYING && soundPlayerState == N
    T_READING) {
158         soundPlayer->play("end_of_tutorial");
159         soundPlayerState = READING_TUTORIAL;
160     }
161 }
162
163 void TutorialState::draw() {
164     if (debugMode == ON) {
165         GamePlayState::draw();
166
167         string stateText = "TUTORIAL";
168         text = TTF_RenderText_Blended(font, stateText.c_str(), textColor);
169         drawText(10, 10, text, mainScreen); // 時間帯の描画
170         SDL_FreeSurface(text);
171     }
172 }
173
174 /**
175  * すべての虫を捕まえたかどうか
176  */
177 bool TutorialState::isClear() {
178     for (list<Insect>::iterator i = insects.begin(); i != insects.end(); i++) {
179         if (!i->getIsCatched()) {
180             return false;
181         }
182         lastCatchedInsectName = i->getName(); // 虫の名前を記憶しておく
183     }
184     return true;
```



```
185 }
186
187 void TutorialState::fadeOut() {
188 }
189
190 void TutorialState::fadeIn() {
191 }
192
193 void TutorialState::stopFadeIn() {
194 }
195
196 void TutorialState::finalize() {
197     // 説明音声を止める
198     soundPlayer->stopAll();
199 }
```

```
#pragma once

#include "TutorialSoundFactory.h"

/**
 * ゲームのチュートリアル(虫取り練習モード)です.
 *
 * @author ikumin
 * @date 2007/1/22
 */
class TutorialState : public GamePlayState {

private:
    bool isClear(); // クリアしたかどうか
    int readingState; // どの説明を読み上げているか
    int soundPlayerState; // CriAuPlayerのStatusで対応できない部分をカバーする
    string lastCatchedInsectName; // 最後に捕まえた虫の名前

    void readExplanationOfTutorial();
    void readDoTutorial();
    void readEndOfTutorial();

    void skip();

protected:
    // オーバーライド
    void draw();
    void processKeyEvent();
    void playSounds();
    void moveCreatures();
    void fadeOut();
    void fadeIn();
    void stopFadeIn();
    void finalize();

public:
    TutorialState(SoundContainer *soundContainer, SDL_Surface *mainScreen);
    ~TutorialState(void);
    void run();
};
```

```
#include "Util.h"

const double Util::M_PI = 3.14159265358979323846264338327 9;
bool Util::isKeyPressed = false;

/**
 * 度数法から弧度法へ変換する
 */
double Util::angleToRadian(double angle) {
    return ((angle - 90) * M_PI / 180);
}

/**
 * 弧度法から度数法へ変換する
 */
double Util::radianToAngle(double radian) {
    return ((radian / M_PI * 180) + 90);
}

/**
 * ボリュームを計算する
 */
double Util::calculateVolume(double targetX, double targetY) {
    double distance = calculateDistance(targetX, targetY);

    // 距離に応じてボリュームを調整する
    double volume;
    if( distance == 0 )   { // 距離が0の場合
        volume = 1.0;
    } else {
        //volume = 1.0f / sqrtf(distance) * 2.0f;
        volume = sqrt(distance) / -10.0 + 2.0;
    }
    if( volume < 0.0f) {
        volume = 0.0f;
    }

    return volume;
}

/**
 * 距離を計算する
 */
double Util::calculateDistance(double targetX, double targetY) {
    // 距離を計算する
    double xDistance = targetX - PLAYER_X;
    double yDistance = targetY - PLAYER_Y;
    double distance = sqrt(xDistance * xDistance + yDistance * yDistance);

    return distance;
}

/**
 * オブジェクト間の角度を計算する
 */
double Util::calculateInterObjectAngle(double targetX, double targetY) {
    // オブジェクト間の角度を計算する
    double radian = atan2(targetY - PLAYER_Y, targetX - PLAYER_X);

    // 弧度法から度数法へ変換する
    double angle = radianToAngle(radian);

    return angle;
```



```
}

/*
 * スピーカーのセンドレベルを計算する
 */
CriAuSendLevel Util::calculateSpeakerSendLevel(double targetX, double targetY) {
    // オブジェクト間の角度を計算する
    Float32 angle = (Float32) calculateInterObjectAngle(targetX, targetY);

    // 角度からセンドレベルを計算する
    Float32 left;
    Float32 right;
    Float32 leftSurround;
    Float32 rightSurround;
    //Float32 center;
    //CriAuUty::CalcSendLevel5Speakers(angle, &left, &right, &leftSurround, &rightSurround, &center);
    CriAuUty::CalcSendLevel4Speakers(angle, &left, &right, &leftSurround, &rightSurround);//, &center);

    // センドレベルを設定する
    CriAuSendLevel sendLevel;
    sendLevel.SetLeft(left);
    sendLevel.SetRight(right);
    sendLevel.SetLeftSurround(leftSurround);
    sendLevel.SetRightSurround(rightSurround);
    //sendLevel.SetCenter(center);

    return sendLevel;
}

/**
 * そのキーが押されたかどうか
 */
bool Util::isPressed(int id) {
    Uint8* keys = SDL_GetKeyState(NULL);
    return keys[id] == SDL_PRESSED;
}

/**
 * そのキーが一度だけ押されたかどうか
 */
bool Util::isPressedOnce(int id) {
    Uint8* keys = SDL_GetKeyState(NULL);
    if (keys[id] == SDL_PRESSED) {
        // キーが過剰に反応することを防ぐ
        if (!isKeyPressed) {
            isKeyPressed = true;
            return true;
        }
    } else if (isReleased(id)) {
        isKeyPressed = false;
    }
    return false;
}

/**
 * そのキーが離されたかどうか
 */
bool Util::isReleased(int id) {
    Uint8* keys = SDL_GetKeyState(NULL);
    return keys[id] == SDL_RELEASED;
}

/**
```

```
 * 制限時間を取得する
 */
int Util::getLimitTime() {
    int limitTime = 1000; // 初期値
    string value = getConfigurationValue("limitTime");
    if (value != "") {
        limitTime = boost::lexical_cast<int>(value);
        if (limitTime > LIMIT_TIME_MAX) {
            limitTime = LIMIT_TIME_MAX;
        } else if (limitTime < LIMIT_TIME_MIN) {
            limitTime = LIMIT_TIME_MIN;
        }
    }
    return limitTime;
}

/**
 * マップの広さを取得する
 */
int Util::getAreaSize() {
    int areaSize = 1300; // 初期値
    string value = getConfigurationValue("areaSize");
    if (value != "") {
        areaSize = boost::lexical_cast<int>(value);
        if (areaSize > AREA_SIZE_MAX) {
            areaSize = AREA_SIZE_MAX;
        } else if (areaSize < AREA_SIZE_MIN) {
            areaSize = AREA_SIZE_MIN;
        }
    }
    return areaSize;
}

/**
 * デバッグモードを取得する
 */
string Util::getConfigurationValue(string propertyName) {
    string configPath = "Config.txt";

    // 設定ファイルを開く
    std::ifstream list(configPath.c_str());

    string line;
    while(getline(list, line)) {
        // 記述が有効でない場合，読み飛ばす
        if (!validate(line)) {
            continue;
        }

        // 設定ファイルに指定されたプロパティがあるか探す
        string::size_type text = line.find(propertyName, 0);
        if (text != string::npos) { // 存在する場合，設定された値を返す
            string::size_type splitIndex = line.find("=", 0);
            return line.substr(splitIndex + 1);
        }
    }

    list.close();
    return ""; // プロパティ名が見つからないときは空文字を返す
}

/**
 * 設定ファイルの記述が有効かどうか調べる
```



```
 */
bool Util::validate(string text) {
    // コメントの場合
    string::size_type result = text.find("#", 0);
    if (result != string::npos) {
        return false;
    }

    // プロパティ名と値が'='で区切られてない場合
    result = text.find("=", 0); // '='で区切る
    if (result == string::npos) {
        return false;
    }

    return true;
}
```

```
#pragma once

#include "SoundMaterial.h"

#include <SDL.h>
#include <boost/lexical_cast.hpp>

/**
 * 便利な関数をまとめておくクラスです.
 *
 * @author ikumin
 * @version 2006/12/15
 */

class Util {

private:
    static bool validate(string text);

public:
    static const double M_PI;
    static bool isKeyPressed;

    static double radianToAngle(double radian);
    static double angleToRadian(double angle);
    static double calculateVolume(double targetX, double targetY);
    static double calculateDistance(double targetX, double targetY);
    static double calculateInterObjectAngle(double targetX, double targetY);
    static CriAuSendLevel calculateSpeakerSendLevel(double targetX, double targetY);

    static bool isPressed(int id);
    static bool isPressedOnce(int id);
    static bool isReleased(int id);

    static int getLimitTime();
    static int getAreaSize();
    static string getConfigurationValue(string propertyName);
};
```

2007/01/31
さうんど おんりぃ
最終報告書

進捗報告会資料

「映像を用いないゲーム」アンケート

評価者：大岩研関係者

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

ゲームに関して：

- 夜になるとちょっとドキドキするのだが，基本的に昼間とあんまりかわっていない点をもうちょっとなんとかできたかな，と思います．
- 虫の生息場所がいまいちわからなかったかな．カエルは川の近くにいるのはなんとなくわかったけど．マップにいろいろ属性があったらおもしろかったかも．沼とか草地とか林とか．それぞれで生息している虫が違ったらおもしろくなりそう．音を配置する前にマップの絵を書いてから作るといいのかなあ．
- 難易度がちょっと高いかな．このゲームがいわゆる逃げゲーではなく取りゲーならば，一番

はじめに馬鹿でもとれそうな場所に簡単に取れる虫を配置して，虫取りのおもしろさがわかるようにしなければならないと思います．

- とれたときの感動がちょっと弱い．もうちょっと派手めの演出があるとおもしろい．とった虫に関する豆知識とかトリビアを言ってくれるとおもしろいよね．そういうのあると先生方も勉強になるとか言ってさらにおもしろがってくれるかも．
- 古典的なやり方だけどスコアに応じてランク付けをするとユーザはもう一回チャレンジしたくなるんじゃないかな
- さらに虫の大きさとかもあってそれで点数変わると燃えるよね．
- もう少し目印的な音があるとだいぶマップが想像しやすいと思います．ベースキャンプとかがあったほうがいいかも．

音に関して：

- ゲームとは直接関係ないのですが，音声の特にカ行が耳に刺さって痛かったです．コンプレッサなどのエフェクタをかけてバランスを調節してください
- 音声のエコーが安っぽい．普通にＢＧＭ流して普通にしゃべったほうがいいソフトウェアに見える（いや，聞こえる）と思うんだけどなあ．
- 足音の聞こえ方がおかしい．左右から交互に聞こえてくるからどんだけガニまたなんだと突っ込んでしまいたくなる（ステレオヘッドフォンだから？）．そのわりには方向転換したときの音が真ん中から聞こえる．むしろ方向転換のときこそ左右から音が聞こえてほしい．普段の足音は真ん中から，しかももっと小さくていいと思う．

「映像を用いないゲーム」アンケート

評価者：大岩研関係者

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（虎がいきなり出てきてびっくりした　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

・練習のときに「このぐらい接近してれば捕まえられますよ」ってわかるようにしてもらいたいです！
・一度スペースを押したときに回転する角度が何度かわかった方が操作しやすいかなと思いました．

「映像を用いないゲーム」アンケート

評価者：大岩研関係者

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（いろんな動物がいるところ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

・ すぐ捕らえられるザコ虫をふやしてもいいのかと思います．捕らえられる虫が少ないのはショックです．ちなみに私は 0 点でした．そうすると，勝ちのある虫という存在が際立つのではないかと思います．
・ 意外にプレイできる範囲が小さいなと思いました．

「映像を用いないゲーム」アンケート

評価者：大岩研関係者

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

- 道に迷った．
  - どの向きを向いているかがわからない．
  - どのくらい角度が変わったかわからない．
  - 一回北を向くなど，向きのリセットとかしたいです．
- ライオンがものすごく多い．
- 本当に虫に近づいているみたいですごい．
- 範囲がない方がうれしい．

「映像を用いないゲーム」アンケート

評価者：大岩研関係者

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

・ 制限時間が近づくと，コチコチ言い出す．
・ 裏技を作る（コナミコマンド）．
・ 音声による説明に，プロの声優を起用する．
・ 「走り」を導入する．

「映像を用いないゲーム」アンケート

評価者：大岩研関係者

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

- 距離感がよくわからない．
- 操作性もあまりよくない．
- 森の中にいる感じはするが，ゲームとしては疑問である．
- というか捕まえられないので，網振りながら歩くことになってつまらない．

「映像を用いないゲーム」アンケート

評価者：横浜市立盲学校生徒

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

「映像を用いないゲーム」アンケート

評価者：横浜市立盲学校生徒

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

- 「虫捕り」という名前（内容）にも関わらず他の音の方が目立ち，意味がわからない．
- 「オケラ」の鳴き声が大きく耳障りだ！
- 虫の音より，ライオンなどの音の方が目立っていた．
- 全盲者には今のやり方でよいと思うが，弱視者には映像が必要だと思う．
  - 自然の絵などを入れる
  - 音声スポーツゲーム（バレーボール・バスケなど）

「映像を用いないゲーム」アンケート

評価者：横浜市立盲学校生徒

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

・ もう少し時間が欲しい．

「映像を用いないゲーム」アンケート

評価者：横浜市立盲学校生徒

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

・ もう少し時間が欲しい．

「映像を用いないゲーム」アンケート

評価者：横浜市立盲学校生徒

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

- 虫がいる場所がわかりにくかった．
- 押すところは上下左右なので分かりやすい．
- 虫の声を頼りにできるから楽しい．
- これからももっとおもしろい良いゲームを作ってください．またやりたい．

「映像を用いないゲーム」アンケート

評価者：横浜市立盲学校生徒

1. 以下の質問に対して，該当する番号に〇をつけて下さい．質問に対して，「その他」を選んだ方はその理由についてもお答え下さい．

(ア) 「映像を用いないゲーム」をプレイした感想を教えてください．
①非常に面白い　②面白い　③普通　③つまらない　④非常につまらない

(イ) 「映像を用いないゲーム」をまたプレイしたいと思いますか？
①是非プレイしたい　②機会があればプレイしたい　③どちらでもよい
④あまりプレイしたくない　⑤二度とプレイしたくない

(ウ) 面白かったと感じた部分があればお答え下さい．（複数回答可）
① 映像を用いないという新しいゲームを体験できる
② 音に臨場感がある
③ 捕った虫の数やスコアを競うことが出来る
④ ゲームの難易度・バランスがちょうど良い
⑤ その他（音声による説明・セリフ　　）

(エ) 改善・追加するべきだと感じた部分があればお答え下さい．（複数回答可）
① 虫や自然の音の聞こえ方
② 音声による説明
③ ゲームの操作性
④ ゲームの難易度・バランス
⑤ 虫捕り以外の要素を追加した方がよい
⑥ その他（　　）

2. このようにすればもっと面白くなるのではないか，自分だったらこのように作るといった，意見・感想があれば自由にお書きください．

2007/01/31
さうんど おんりぃ
最終報告書

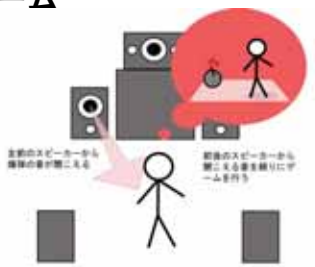
プロジェクト宣伝用資料

映像のないゲーム

環境情報学部4年　橋山牧人

映像のないゲームとは？

- 5.1chサラウンドスピーカーを用いた立体音響から流れてくる音の大きさや定位を聞いてプレイするゲーム



「さうんど おんりぃ」について

- 先学期，株式会社ユードーの南雲代表取締役より，「映像のないゲーム」を作って欲しいという依頼を受けた
- 「さうんど おんりぃ」というプロジェクトを立ち上げ，先学期で「映像のないゲーム」のプロトタイプを製作した

体制

今期やりたいこと

- 先学期の成果物をパワーアップさせる
  - ゲームデザインの向上
  - 音の聞こえ方の改善
  - インターフェイスの改良
  - 横浜市立盲学校への生徒にヒアリング

etc…