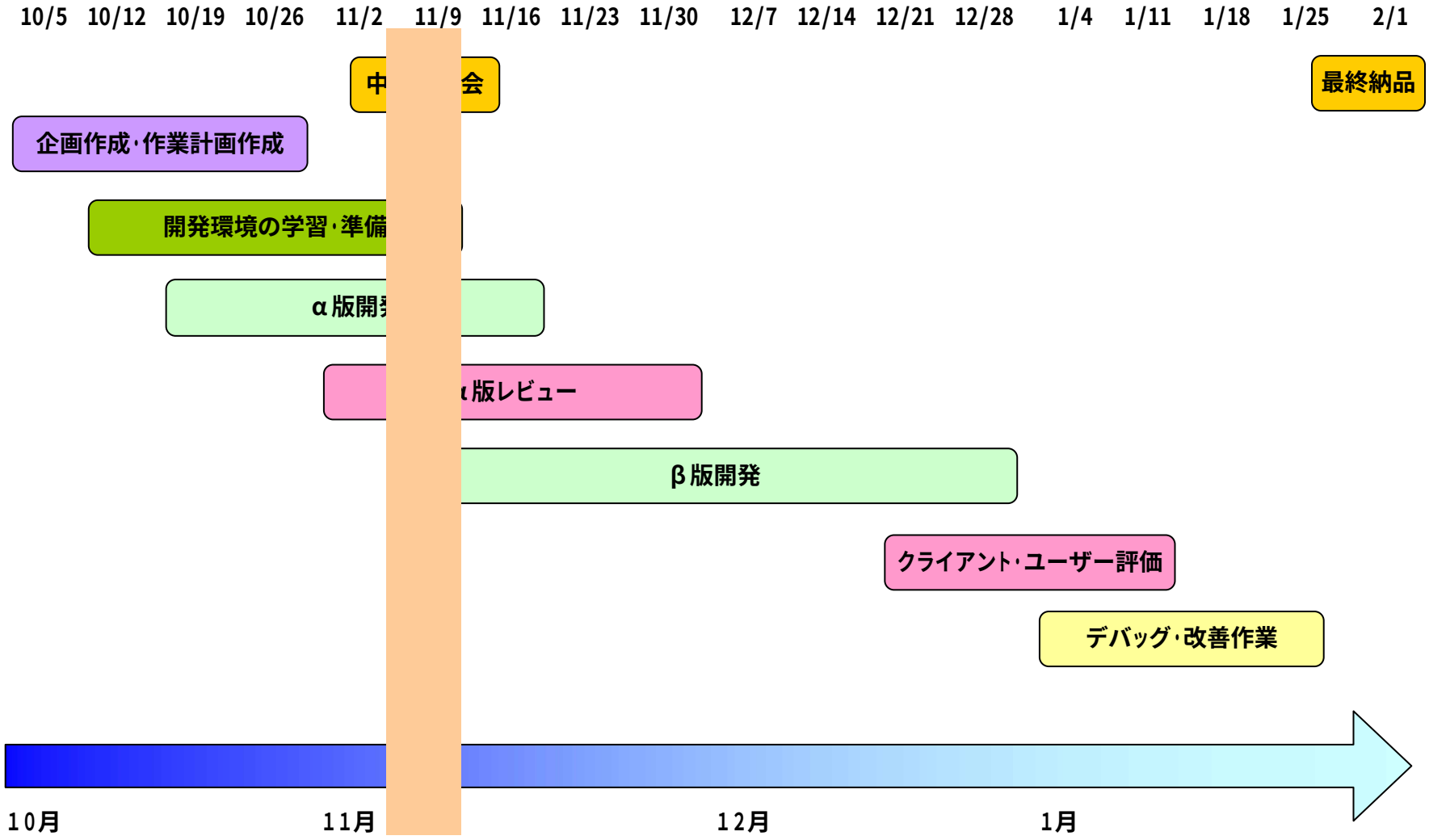
2007/01/31
さうんど おんりぃ
最終報告書

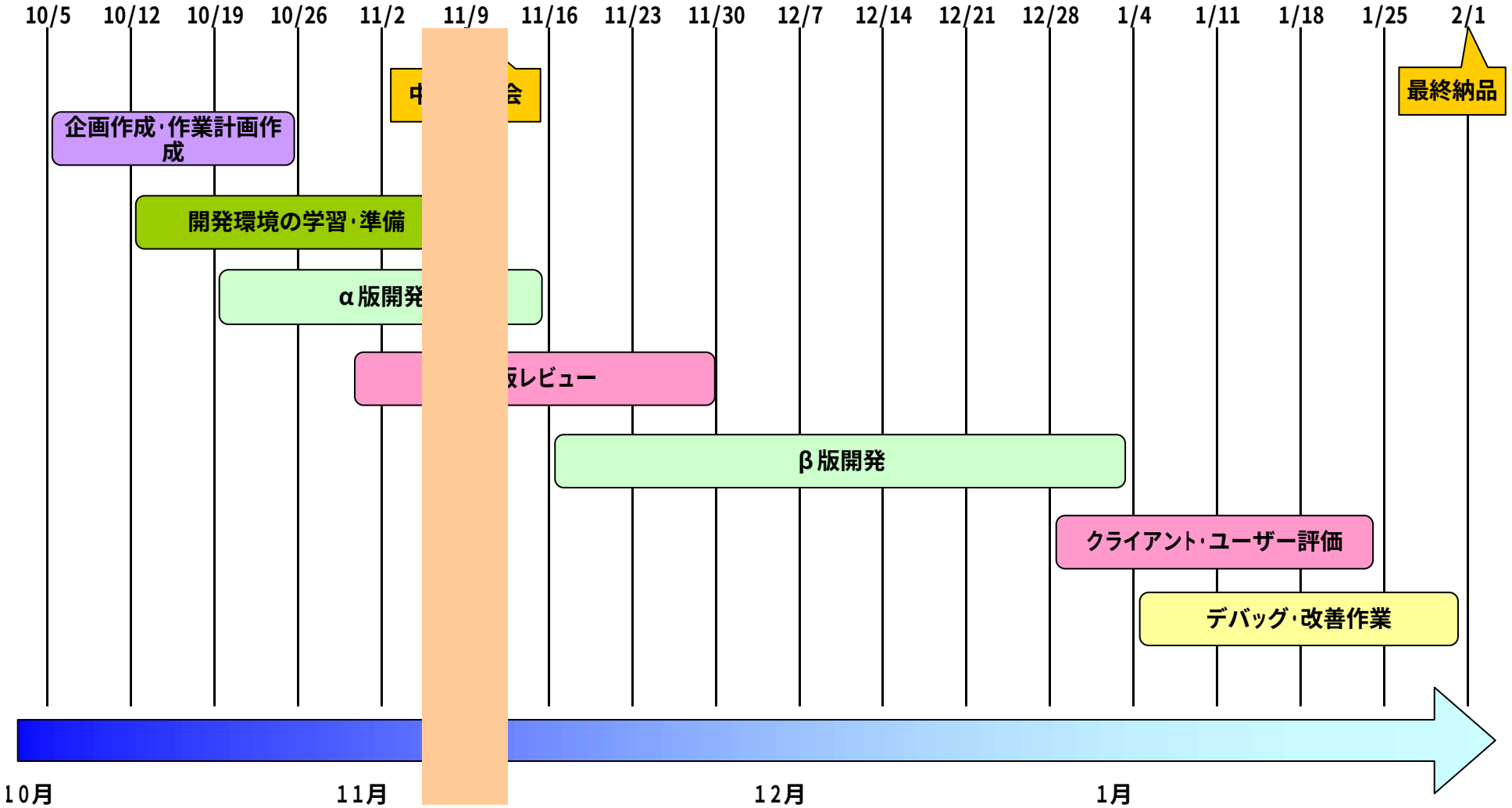
全体スケジュール

さうんどおんりぃ2　WBS



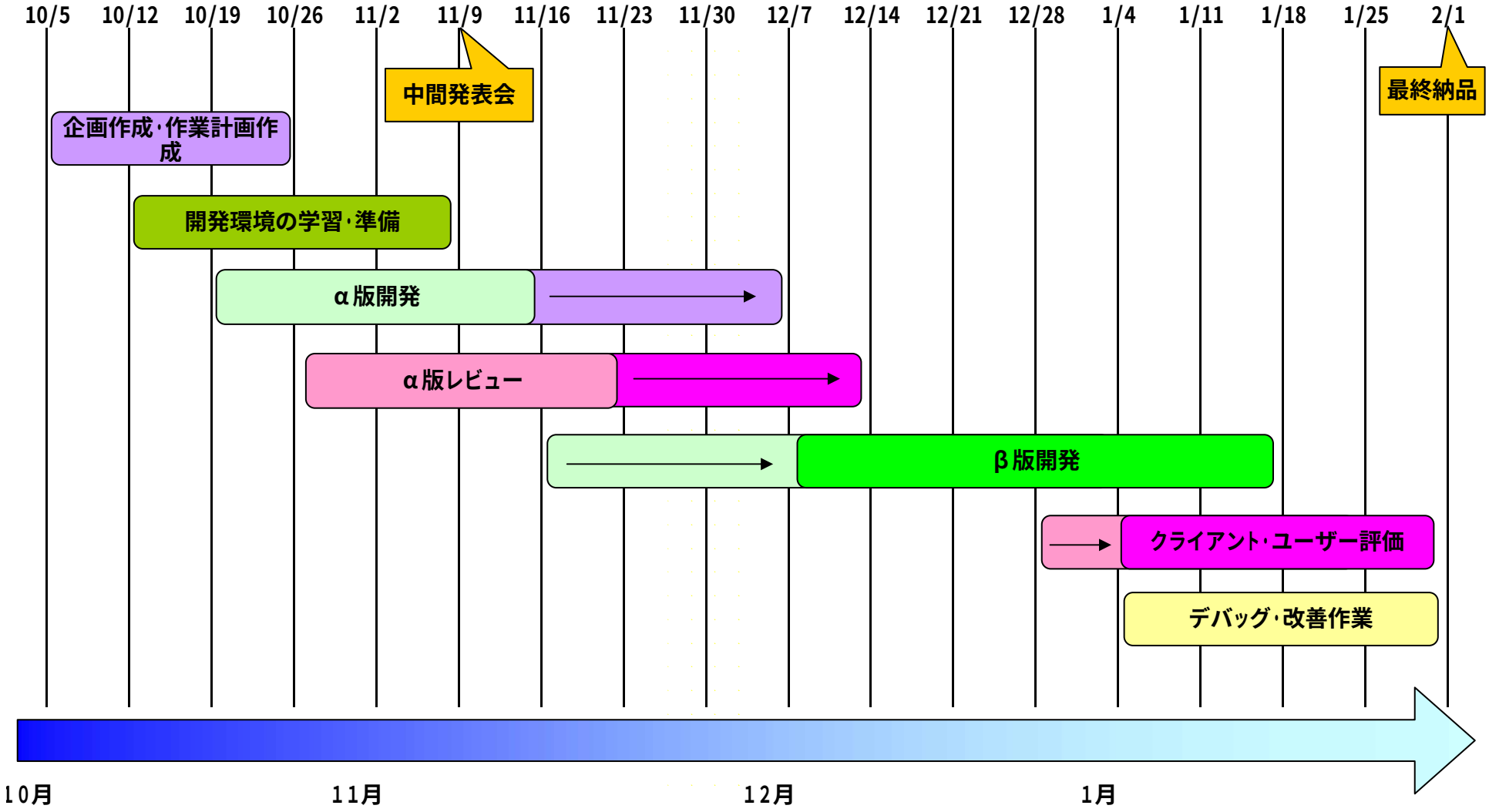
作成日：2006/10/25
作成者：菊地徹也

さうんどおんりぃ2　WBS



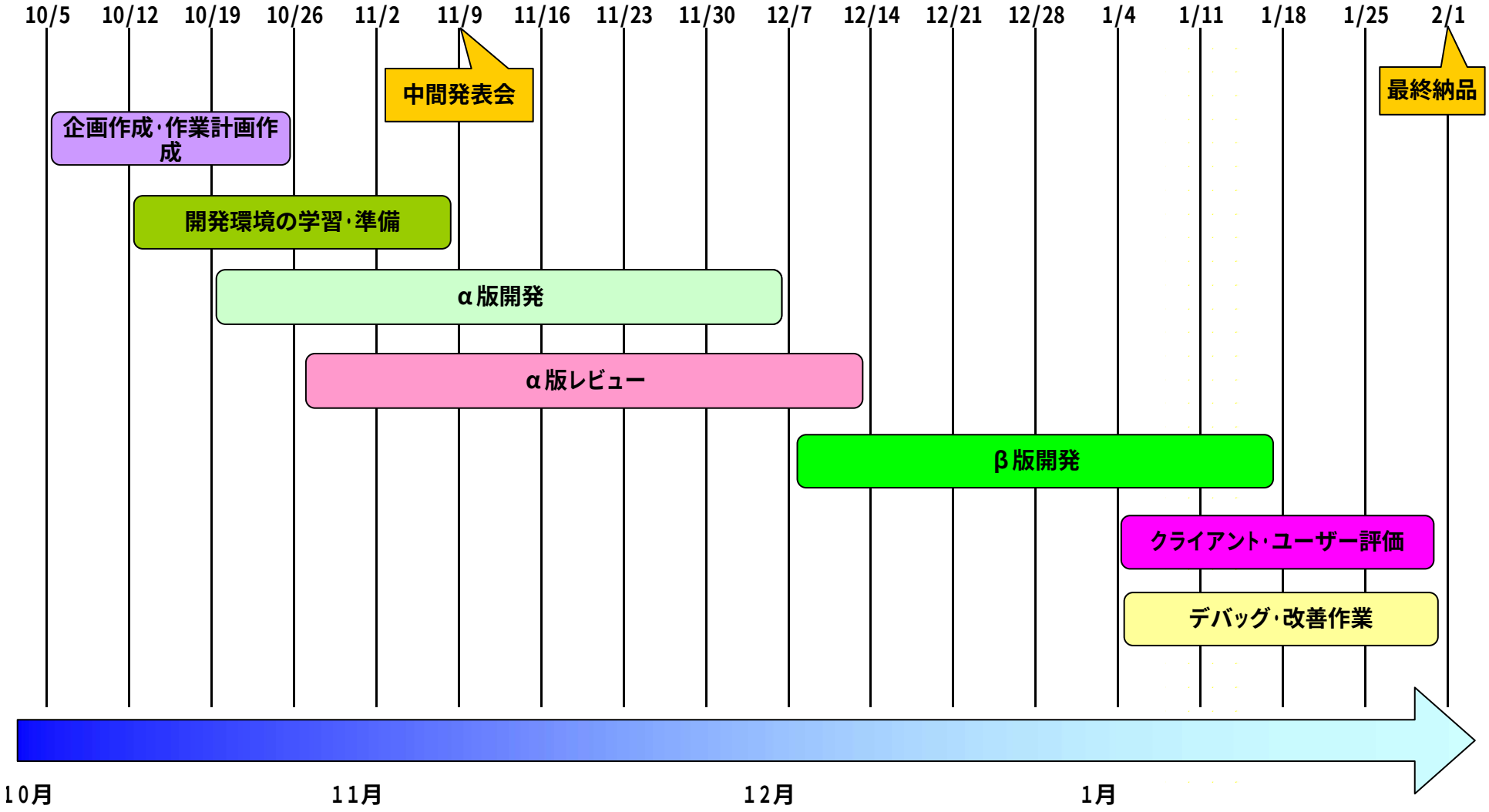
作成日：2006/11/30
作成者：菊地徹也

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	さうんどおんりい前作のソース確認	

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

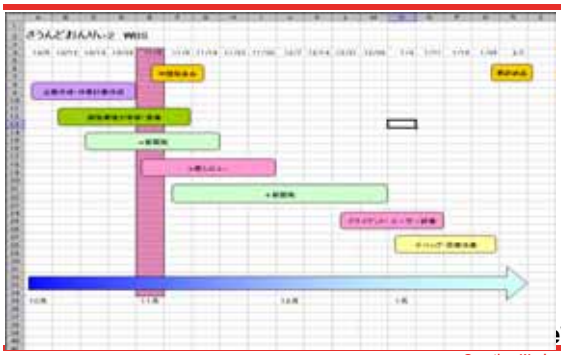
CreW
Creative Workspace

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

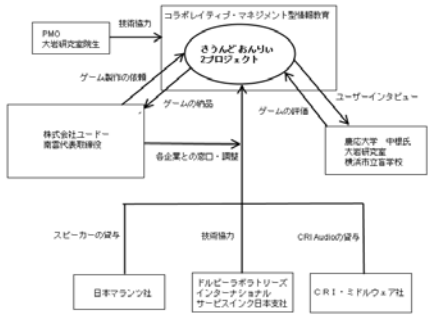
CreW
Creative Workspace

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

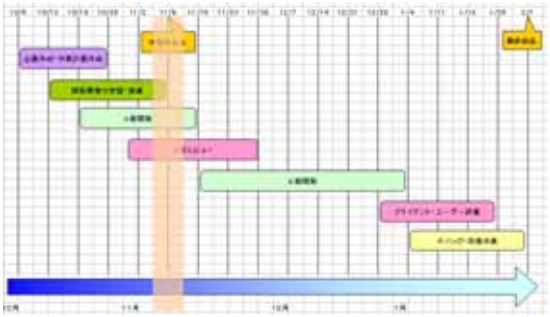
- 評価対象
  - 大岩研関係者，慶應大学の中根様，横浜市立盲学校の方々
- 評価方法
  - ユーザーアンケート
    - 前期プロジェクトで作成した選択式のアンケートを改良
    - 面白いと感じた人が7割以上，もう一度やりたいと思う人が8割以上を目指す
  - 健常者と視覚にハンディキャップを有する方々の両方が面白いというゲーム

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

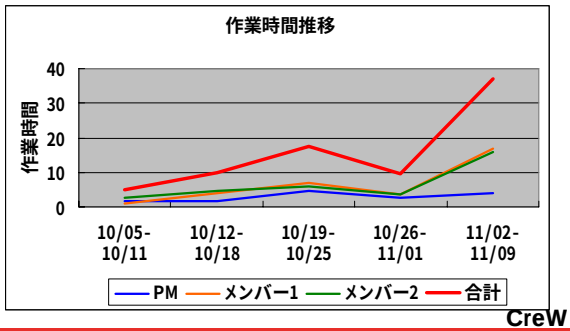
CreW
Creative Workspace

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

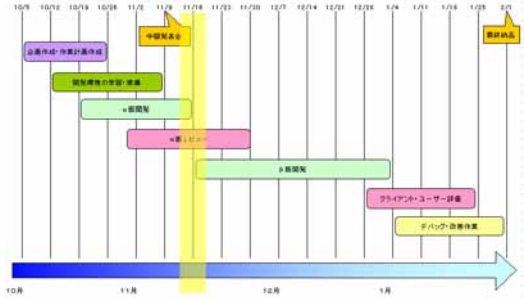
CreW
Creative Workspace

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

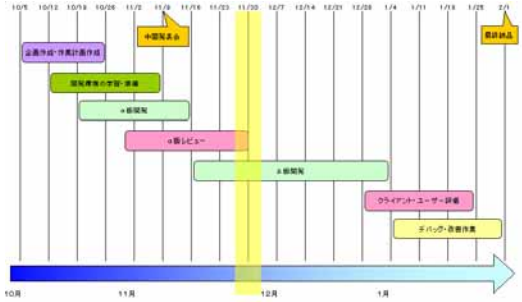
CreW
Creative Workspace

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

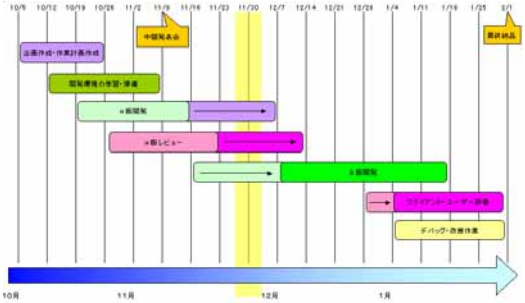
CreW
Creative Workspace

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

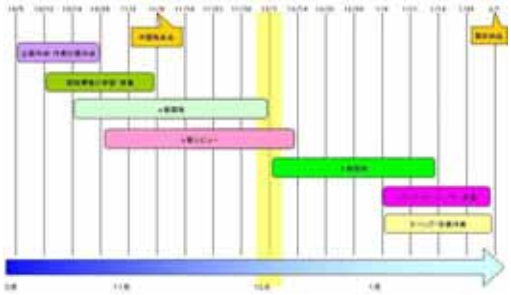
- 全体スケジュールについて
- β版移行について
  - α版の制作を通してわかったこと
  - β版移行のタイミングについて
- 今週行ったこと
  - 全体の作業見積もりと実績について
  - 実装について
- 来週までに行うこと
  - バグフィックスを行う
  - 慶応大学中根様に，ユーザーレビューを実施する
  - β版の開発を開始する

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

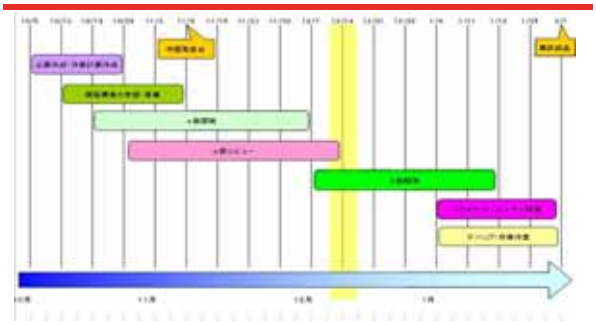
㈱インテム　菊地徹也 (PM)
環境情報学部4年　橋山牧人
環境情報学部4年　藤原育実

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

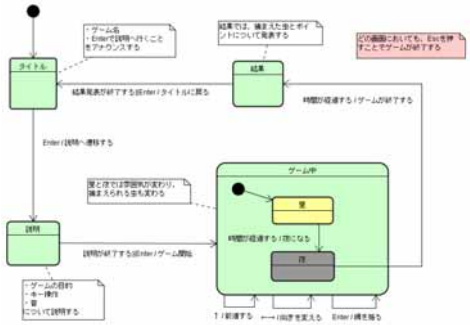
- Wiki参照

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

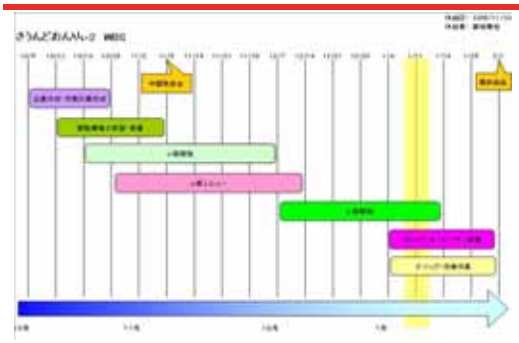
- 全体スケジュールについて
- 今週行ったこと
- デモンストレーション
- 来週までに行うこと

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 );
    }
}
```

```
#include "Sound.h"

/*
 * コンストラクタ
 */
Sound::Sound(string soundName, float x, float y) {
    this->soundName = soundName;
    this->x = x;
    this->y = y;
}

float getX() {
    //return x;
    return 0;
}

float getY() {
    //return y;
    return 0;
}

/*
 * デストラクタ
 */
Sound::~Sound() {
}
```

```
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

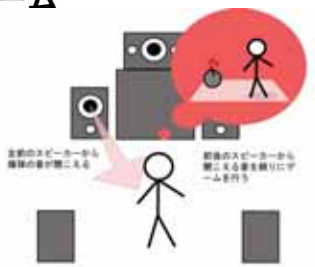
プロジェクト宣伝用資料

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…