2006年度　卒　業　論　文

ゲームにおけるGUIツールキットに関する研究

指導教員：渡辺 大地講師

メディア学部 ゲームサイエンス
学籍番号　M0103242
鈴木 裕海

2006 年度　　卒　業　論　文　概　要

論文題目

ゲームにおける GUI ツールキット に関する研究

メディア学部 **学籍番号：M0103242**	**氏名**	鈴木 裕海	**指導教員**	渡辺 大地講師

キーワード	GUI、ゲーム、オブジェクト指向、ツールキット、開発環境

近年ハードウェアの処理能力向上と共に、高性能なゲーム機とゲームソフト（以下ゲーム）が一般家庭に普及してきている。ゲームや一般的な PC のアプリケーションでは、コンピュータの入力操作を視覚的に行えるようにした画面がよく用いられる。これらの画面はグラフィカルユーザインタフェース（以下 GUI）と呼ばれ、ゲームにおいて GUI は重要な要素の 1 つとなっている。現在ゲームにおいて GUI の開発手法を調査した結果、2 つの手法に大別できた。第 1 の手法はプログラマ自身が GUI の構成要素を定義し、画面を実装する手法である。第 2 の手法は、オーサリングツールによる実装 GUI の実装である。オーサリングツールとは、広義では「文字や画像、音声、動画といったデータを編集してソフトウェアを作ることができるアプリケーション」である。その中で本研究で扱っているオーサリングツールとは、一般に GUI ビルダと呼ばれるものであり、プログラミング能力が高くない人でも GUI 部品の一覧から利用したい部品をマウスで選択し、表示画面上に配置することにより、容易に GUI の画面設計を行うことができるものである。上記で述べたゲームにおける 2 つの GUI の実装方法には問題点がある。第 1 の手法ではプログラマ自身が GUI の構成要素のアルゴリズムや内部構造を記述しなければならず、手間がかかる。第 2 の手法では開発者の意図により大きく依存するため、自由度が著しく低下してしまう。一方で、PC では GUI を開発する有効な手法として、GUI の部品をライブラリ群として提供しているツールキットというものがある。しかし、PC の GUI のツールキットは入力デバイスの操作性に合わせた GUI の設計と、GUI に対する考え方の違いによりゲームにおける GUI の実装には適していないと考えられる。本研究では、ゲームと PC アプリケーションの違いを考慮した上で、ゲームにおける GUI のツールキットを開発し、プログラマが細かな実装過程を意識することなく簡易に画面を実装することを目的とする。そして、本研究で開発したツールキットを利用しない場合と利用した場合による実装過程の比較を行い、より簡易に実装可能であることの検証を行った。

目 次

図 目 次

第 1 章

はじめに

1.1 研究背景

近年ハードウェアの処理能力向上と共に、高性能なゲーム機とゲームソフト（以下ゲーム）が一般家庭に普及してきている。ゲームを動作させるゲーム機は、コンシューマゲーム機やコンソールゲーム機とも呼ばれており、ハードウェアをビデオ端子を用いてテレビ画面に接続し、ゲーム機に付属している専用のコントローラを使用することにより操作が可能となる。また、PC よりも安価なことから広く普及しており、コンピュータの知識が無い人でも簡易に利用することができる。ゲームや一般的な PC のアプリケーションでは、コンピュータの入力操作を視覚的に行えるようにした画面がよく用いられる。これらの画面はグラフィカルユーザインタフェース（以下 GUI）[1] と呼ばれ、ゲームに対する GUI は、画面上の大半を占める場合もあれば、画面上の一部のみに表示される場合等、様々である。本稿では、画面上の GUI を構成する一部または全体を総称して「インタフェース部品」と定義する。

現在ゲームにおいて GUI の開発手法を調査した結果、2 つの手法に大別できた。第 1 の手法はプログラマがインタフェース部品を定義し、GUI の実装を行う手法である。第 2 の手法は、オーサリングツールによる GUI の実装を行う手法である。第 1 の手法は、プログラマ自身がインタフェース部品にはどのような機能を持つかを定義し、その機能の具体的な実現方法をプログラムにより記述している。第 2

の手法であるオーサリングツールとは、広義では「文字や画像、音声、動画といったデータを編集してソフトウェアを作ることができるアプリケーション」である。その中で本研究で扱っているオーサリングツールとは、一般にGUIビルダと呼ばれるものであり、プログラミング能力が高くない人でもGUI部品の一覧から利用したい部品をマウスで選択し、表示画面上に配置することにより、容易にGUIの画面設計を行うことができるものである。例としてFlash[2]やAnark Studio[3]などがあり、この2つは実際にゲームのGUIとして用いられた事例[4]がある。上記で述べたゲームにおける2つのGUIの実装方法には問題点がある。第1の手法の問題点として、プログラマがインタフェース部品を直接定義することでGUIを実装することになるが、この時、インタフェース部品を制御するアルゴリズムや内部構造は実装者自身が考え、プログラムを記述しなければならず、非常に手間がかかるということが挙げられる。第2の手法の問題点として、GUIを構築できるオーサリングツールは、オーサリングツール開発者の意図に大きく依存する[5]ため、実装の自由度が著しく低くなってしまう。

一方で、PCアプリケーションのGUIの実装においては様々な開発環境が研究[6]し、開発されている。その1つとしてプログラミングにおいて開発効率を高める手法としてツールキット[7][8][9][10][11]というものがある。ツールキットとは、広義の意味ではアプリケーションを作成する手助けをする為のソフトウェアルーチン及びユーティリティーのセットであるが、本研究においてのツールキットは高度な処理を行う関数を用意しカプセル化して開発者に提供するライブラリ群のことを指す。ツールキットのメリットとして、提供されている高度な機能を利用する際に、内部の構造を理解したり学習したりせずに開発を行うことが出来るため、目的のアプリケーション制作までに必要な学習時間を省くことができ、効率よく開発を行うことが出来るという点がある。またツールキットを使うことにより高度な機能が少ない行数で使用できるためソースコードの量を減らすことが出来、プログラムの開発において大きな変更が行われるときでも、少量の変更で済むという点も挙げられる。GUIの要素としてボタンやスクロールバー等の画面を構

成する部品が提供されている。具体的なツールキットの例としてGUIツールキットのFLTK[12]やQt[13][14]、プログラミング言語JavaのSwing[15]等が挙げられる。FLTKは複数のOSと互換性を持つ小さくて高速なGUIツールキットである。OpenGLを通しての3Dグラフィックスをサポートしている。Qtは C++クラスライブラリであり、Unix上でのGUIプログラミングを目的としてつくられたものである。現在ではWindows環境でも利用でき、移植性に優れている。Swingは JDK(Java Development kit)の機能の1つであり、JFCのユーザインタフェース部分を担っているAWT(Abstruct Window Toolkit)の拡張である。Java言語で開発していることもあり、プラットフォームに依存しないという移植性の面でも優れている。このように、PCのアプリケーションの分野ではGUIの実装において必要とされる要素がGUI部品として定義され、プログラマが利用できる形となっている。

PCアプリケーションに対して、GUIツールキットを利用することによるGUIの構築は非常に有効であるが、ゲームにおいては、入力デバイスの違いと、GUIに対する考え方が起因となることにより、インタフェース部品の特徴が異なるものとなっている。まず、入力デバイスの違いとして、ゲームではゲーム機に付属する専用のコントローラによる操作を前提としたGUIの設計が行われ、PCではキーボードやマウスによる操作を前提としたGUIの設計が行われてきた。次に、GUIに対する考え方の違いとして、PCのGUIはアプリケーションの操作性を重視し、簡易に目的を達成することを重視しているが、ゲームでは、GUIのデザインによってどういう機能を持つかを認識させることを重視している。このことから、PCのGUIツールキットはゲームのGUIの実装には適していないと考えられる。

本研究では、ゲームとPCアプリケーションの違いを考慮した上で、ゲームにおけるGUIの要素を抽象化することにより、インタフェース部品として利用する手法を提案する。その手法として、インタフェース部品をライブラリという形で提供し、プログラマが細かな実装過程を意識することなく簡易に画面を実装することを目的とする。そして、本研究で開発したツールキットを利用しない場合と利

用した場合による実装過程の比較を行い、より簡易に実装可能であることの検証を行った。

1.2　本論文の構成

本稿では、2 章で既存の開発手法の問題点と解決法であるツールキットの概要について述べる。次に第 3 章ツールキットの有用性を示す目的で実際に複数のジャンルの画面の実装結果と検証を行う。4 章でとまとめを述べる。

第 2 章

ツールキットの概要と設計理念

本章では既存の技術による問題点と本研究での問題の解決策の紹介を行う。

2.1 現状の問題点とツールキットの必要性

これまでのゲームの GUI をインタフェース部品を用いずに実装する場合には以下の要件を必要とする。

- インタフェース部品の定義
- インタフェース部品に対するデバイス入力の検知
- インタフェース部品の入力に対する挙動の設定

第 1 のインタフェース部品の定義では、ビジネス用の PC アプリケーションを開発する場合、ボタンやメニューといった GUI 部品があらかじめ定義されており、画面上に配置し、「押された時」といったの入力に対する反応を記述することで開発を行う。しかしゲームでは、GUI を構成するインタフェース部品の定義というものはなく、ポリゴンやテクスチャなどの基本的な表示要素を DirectX[16][17] や OpenGL[18] などの API（アプリケーションプログラミングインタフェース）でプログラムを記述する必要がある。API とはアプリケーションに提供する関数セットのことである。そして、表示された表示要素に対して挙動を定義し記述するこ

とでインタフェース部品としての機能を初めて果たす。このように表示要素自体があらかじめ明確な機能を持っているわけではなく、必要な機能は開発者が独自に定義しなければならず手間がかかってしまう。

第2のインタフェース部品に対するデバイス入力の検知では、PCで提供されているGUI部品はデバイス入力がGUI部品に対して行われたことを検知する機能を備えている。例として、メッセージボックスで表示されるOKボタンでは、ボタンをマウスポインタでクリックしたときにボタンが現在押されているという情報を受け取る関数が用意されている。開発者はボタンの状態を常に取得することができ、簡易に状態に合わせてプログラムを記述することができる。ゲームのGUIでもインタフェース部品に対して入力を行い、入力に応じた処理をして欲しい場合がある。このボタンのような状態検知の機能を持つインタフェース部品を汎用プログラミング言語のみで実装する場合には、「押されている状態」や「放されている状態」といった機能を、実装者自身が定義してプログラムを記述する必要があり、多くの労力が必要となる。

第3のインタフェース部品の入力に対する挙動とは、PCアプリケーションで利用されているGUI部品のボタンは、マウスでクリックされたときには、GUI部品に対してイベントが起きたことを示すために、窪んだ画像が表示されるようになっている。ゲームにおいて、コントローラによる入力操作により現在選択中である項目を強調するために現在地点を示すカーソルが点滅している場合や選択されている表示物自体がキーフレームアニメーションによる挙動を行っている場合がある。このような選択時における挙動をプログラミングで記述する場合にはPCのGUI部品で提供されている基本的な機能のみでは行うことができない。

2.2 ツールキットの概要

本節では前節で述べた問題点を解決する手法としてGUIの実装を目的としたツールキットを提案する。ツールキットとは、広義の意味ではアプリケーションを作成する手助けをする為のソフトウェアルーチン及びユーティリティのセット

であるが、本研究においてのツールキットは高度な処理を行う関数を用意しカプセル化して開発者に提供するライブラリ群のことを指す。ツールキットを実装するにあたり、GUI を構成するものをインタフェース部品として抽象化することにより、インタフェース部品を組み合わせて簡易に GUI の実装が可能になるを目的とする。

ツールキットの設計理念として以下のことが挙げられる。

- 簡易にインタフェース部品の利用を可能とする
- 生成した単数のインタフェース部品に対して属性の変更を可能とする
- 入力による状態遷移を自動的に適応する
- インタフェース部品の汎用性とプログラムの規模に応じた段階分けを行う

簡易に利用が可能とは、インタフェース部品の生成と初期化の時点である程度の設定を自動的に行うことである。インタフェース部品によっては特有の設定を行う必要があるが、ほとんどの場合は初期化の時点で利用可能になることを想定する。生成したインタフェース部品の属性の変更は、ツールキットを利用する開発者が生成したインタフェース部品に対して属性の変更を行いたい場合に行うことができる処理である。例えば文字の色の変更を行うときには、関数 1 つで目的の処理が簡易に行うことができる設計とする。状態遷移の自動的な適応は、インタフェース部品ごとに更新という処理が備わっており、プログラムのループ中に呼び出すことによりキーフレームアニメーションや座標変換の処理がリアルタイムに変更が可能となっている。

ツールキットの設計を行うにあたり、インタフェース部品がどのような要求を満たし、システムにどのような機能が必要かを明確にすることが重要である [19]。本論文ではツールキットの開発を行うにあたり単体のインタフェース部品の大きさと機能の豊富さを考慮し、3 つの Level、アルゴリズムとシステムの機能を持った Level1 のインタフェース部品、単数で画面上に利用できる Level2 のインタフェー

ス部品、機能が豊富であり、そのままインタフェース部品が画面として利用できるLevel3 のインタフェース部品に分類した。Level1 ではアルゴリズムとシステムに特化したプログラムであり、描画の機能というものは備えていない。汎用的で小さなプログラムである為に、応用範囲は広いが単純な機能しか備えていない。Level2 は実際に表示物として利用できる形態になっており、イベント処理を付けることが可能となっている。画像の変更や移動が可能となっており、Level1 よりもゲームに利用しやすい形態となっているがその分容量が大きくなり汎用性は低下している。Level3 では Level1 と Level2 を利用して開発したものであり、このプログラムで画面レベルのものが表示・利用できる。もちろんその分汎用性は低くなっており、開発者はデザインや使いやすさといった点に力を注ぐことになる。図 2.1 は 3 つの段階を示したものである。

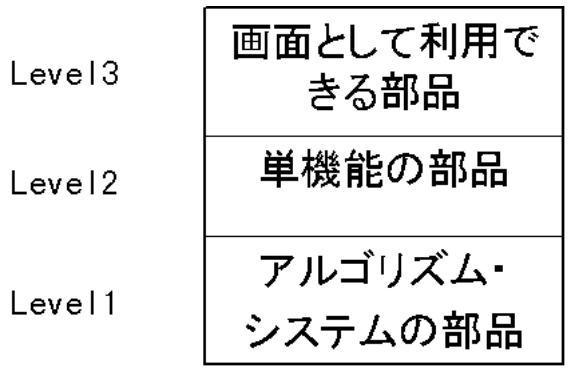

図 2.1: インタフェース部品の機能による段階分け

今回開発するツールキットは OpenGL ベースの 3DCG クラスライブラリである FK System[20][21] に対して、拡張を行うことにより実現する。FK System を利用する大きな理由としてモデルとシーンという概念がある。モデルはあらゆる表示物をモデルという大きな括りで抽象化することにより、異なる種類の表示物でも共通の手法により利用可能となっている。もう一方のシーンは複数のモデルとカメラから構成している「場面」であり、ウィンドウ上でシーンを切り替えることにより簡易に異なる画面にすることが可能となる。二つの概念を採用することに

よりGUIをアプリケーション上の表示物の一部として捉えることができ、管理・運用が容易になる。

本論文のツールキットはFK System上での利用が前提となっている。開発したツールキットの利用法として大きく分けて2つのことが考えられる。1つ目は変数として定義すること、2つ目はクラスの継承による利用である。変数としての利用は、ツールキットで開発したインタフェース部品は目的の機能を利用したい場合は細かいインタフェース部品として定義しているため、機能ごとに利用することが可能である。継承は、生成したインタフェース部品の効果的な再利用を目的としたものであり、オブジェクト指向プログラミングの代表的なメリットの1つである。ツールキットではLevelが低くなるほど汎用性が高く、継承を行うことを念頭においており、Levelが高くなるほど機能が豊富であり、GUIとして利用しやすい設計となっている。

2.3 ツールキットの構成

本節では前節の設計理念に基づいて実装したツールキットのクラス構成について述べる。本研究で開発したツールキットは、ゲームプログラムの重要な要素の1つであるGUIを実装する目的で提供されるライブラリ群である。ツールキットはゲームプログラム内の他のプログラムとも連携して利用することが可能である。例として、ゲーム上に登場するキャラクタのパラメータを変更させる画面を生成する場合にはツールキットで提供されているインタフェース部品のみではなく、ゲームプログラム内にあるキャラクタのパラメータをインタフェース部品と関連付ける必要がある。

ツールキットは表2.1のクラス群を提供しており、GUIを実装することを目的とする。Level1ではインタフェース部品の状態を制御するfk_Conditionクラス、単数のインタフェース部品同士を関連付けるfk_Rateクラスなどアルゴリズムを重視したクラスである。Level2では格闘ゲームやアクションゲーム利用されるfk_Gaugeクラスや画面上に選択肢を生成するfk_Selectクラスなど画面に表示させることが

できるインタフェース部品である。Level3 では名前入力画面を目的とした fk_Name といった画面レベルで利用できるクラスを提供する。

表 2.1: ツールキットで提供されるクラス

Level3	fk_Name	入力に用いる文字群を任意のファイルから読み出し、画面に表示された文字を選ぶことができる名前入力画面を生成する
Level2	fk_Gauge	格闘ゲームやアクションゲームで利用されるゲージを生成する
	fk_Select	質問に対して、選択して答えられるようにした
	fk_SelectArray	fk_Select を簡易に複数利用できるようにした
	fk_Slider	スライダ（滑動部）を生成する
	fk_Text	文字を簡易に画面に出力する
Level1	fk_Condition	GUI 部品の状態を定義する
	fk_Link	単数のインタフェース部品同士を関連付け、排他制御を行う
	fk_Input	入力情報をトレースする
	fk_KeyInput	fk_Input を継承してキーボードをトレースする
	fk_Rate	値に対しての割合を計算する
	KeySystem	故意的な入力の遅延を行う
	AnimationLinear	線形補間を行うクラス

2.4 ツールキットにおいて工夫した点

ここからはツールキットの実装過程において筆者が独自に工夫した点を述べる。

2.4.1 インタフェース部品のグループ化と排他制御

PC の GUI 部品にはラジオボタンというものがある。ラジオボタンには複数の GUI 部品をひとまとめにグループ化する機能や、複数の項目の中から 1 つの項目のみを選択する排他制御の機能を持っている。ラジオボタンは他の項目を選択している状態で別の項目を選ぶと今まで選択状態だった項目は空白状態に戻る。これをゲーム画面でも使えるようにするために、ツールキットではグループ化と排他制御の概念を取り入れることで、インタフェース部品をグループ化し、グループ内のどの部品が現在操作可能なのかを簡易に制御することを可能にした。プログラム上では、選択したい項目を指定するだけで他の選択されていないインタフェース部

品に対して排他的な処理が行われる。このような設計を行うことで、インタフェース部品の利用者が状態を制御する冗長なコーディングを行う必要がなくなる。

2.4.2 割合の計算

PC のアプリケーションで用いられる GUI 部品の中には数多くのアルゴリズムが用いられている。ゲーム中では機能的に PC で用いられる GUI 部品と類似しているものがあり、そのまま PC のアルゴリズムをそのまま利用できるのではないかと考えた。格闘ゲームの体力の計算を簡易に行う方法として、実数から割合を検出する数式を今回は FLTK の GUI 部品から利用した。最大値 m_1 と最小値 m_2、そして現在値 V を設定することで簡易に割合 r か現在値 V から現在の状態を導き出すことができる。以下の式 (2.1) は割合から現在値を導き出す式である。

$$V = \frac{r(m_1 - m_2)}{m_2} \tag{2.1}$$

次に式 (2.2) は値からその全体の割合 R を導き出す式である。

$$R = \frac{V - m_2}{m_1 - m_2} \tag{2.2}$$

以上の式を利用することにより、ゲージなどの割合を示すインタフェース部品を簡易に実装することができる。

第 3 章

実装結果と考察

3.1 比較検証

本節ではツールキットの有用性を示すために複数のジャンルのゲームの GUI を実装し評価を行う。本研究では 2 種類の GUI を実装した。1 つ目はツールキットのみで実装したコンフィグ画面である。2 つ目は既存のアクションゲームの GUI に、ツールキットを用いて同様の実装を行い、更に拡張を加えた画面である。

まずツールキットのみを用いて実装したコンフィグ画面の実装の検証を行う。今回実装したコンフィグ画面は FINAL FANTASY VII[22] の画面を模倣することにより実現した画面である。今回のコンフィグ画面は複数の選択肢とスライダから構成されている。

実装例として、図 3.1 の左にある複数の選択肢の列と同様の機能を GUI ツールキットを用いずに実装した場合は以下のような手順が必要となる。

- 選択肢の表示において文字画像を設定が必要
- 表示画像を画面上で等間隔の配置
- 入力による状態遷移を自動的に適応
- カーソルで使用する画像を表示させる初期設定
- デバイスによる入力検知の設定

- 選択肢に対応したカーソルの移動の指定
- インタフェース部品同士の関連付けの設定

以上のように実装には数多くの設定が必要となり、それぞれの項目は十数行のコーディングが必要となる。だが、ツールキットを用いて実装を行うことにより、簡易に表示させることができる。例として画像を画面上に等間隔に配置させるには以下のように記述する。

```
//等間隔に表示
array.SetBlank(50, 50, 0, 30);
```

array はツールキットで提供している fk_SelectArray クラスのインスタンスである。fk_SelectArray::SetBlank() は一番目と二番目の引数で初期座標を設定し、三番目と四番目の引数でそれぞれ X 座標と Y 座標の間隔を設定する。

このようにまとまった処理を 1 つのメソッドにすることで少ない命令数でコンフィグ画面を実装した。この他にも、デバイスによる入力検知の制御や単数のインタフェース部品同士の関連付けといったクラスとして使うことにより、わずか 351 行でインタラクティブなコンフィグ画面を実装可能とした。図 3.2 にその実行結果を示す。

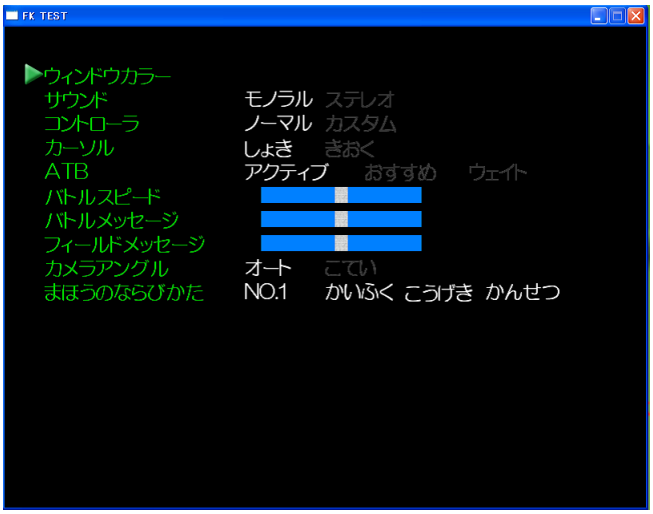

図 3.1: コンフィグ画面の実装結果

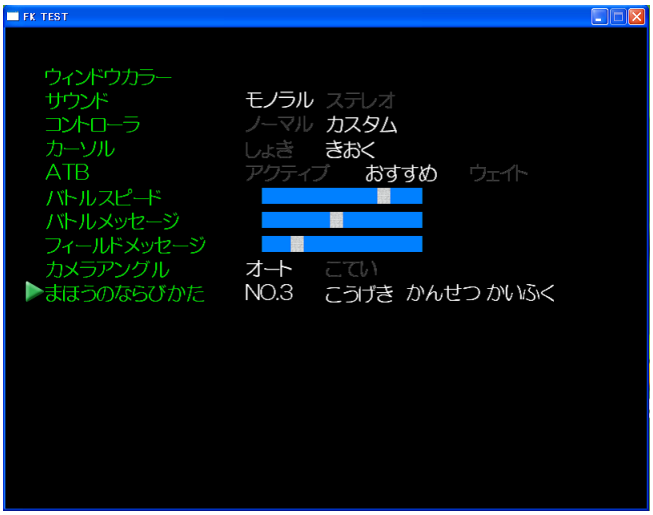

図 3.2: 入力による画面の変化

次に、既存のアクションゲームの GUI への、ツールキットの適用である。１つ目の実装とは違い、既に GUI は完成されており、より簡易に実装を行うことで有用性を実証する。アクションゲームは以下の構成となっている。

- タイトル画面 (ゲームプログラムの初期表示画面)
- アドベンチャーパート（会話画面）

- アクションパート（戦闘画面）
- 終了

タイトル画面は実装内容自体は 1 つ目の検証と同様であり、選択の列の部品を利用し、画面を実装することができた。図 3.3 にその実行結果を示す。

図 3.3: タイトル画面

アクションパートでは GUI のみではなく、ゲームのメインとなる戦闘の処理も含まれており、ツールキットで記述すべき部分を探すことが困難であった。アクションパート上で変更することとなる GUI にキャラクタの体力を表示させるゲージがある。本研究でツールキットを適用したアクションゲームのプログラムでは、画面上に表示されているゲージの減少を、ゲージに対応している現在値に対して適したゲージの長さになるようにポリゴンを伸縮する計算を考え実装していた。ツールキットではゲージクラスというものを用意していたが、これはビットマップなどの画像ファイルを用意し、画像の伸縮を簡易に行うことでゲージを操作することを想定していたため、矩形のポリゴンにより実装されている体力ゲージを既存のゲージクラスでは実装できなかった。そこで、ゲージクラスより低レベルのクラスを継承し新しいクラスを生成することにより簡易にゲージを実装し、更に拡張

を行うことを試みた。継承したクラスは割合を計算する fk_Rate クラスと線形補間によりキーフレームアニメーションを行う AnimationLinear クラスである。この 2 つのクラスをゲージを生成している fk_OverlayModel に継承させることにより、リアルタイムの伸縮機能を持ったゲージを生成することができた 。実装する際には以下のように記述する。

```
//体力ゲージの伸縮
bufferModel.SetGaugeX(lan->getHP(), 2000);
//更新
bufferModel.Update();
```

SetGaugeX() は表示しているポリゴンを X 軸に対して伸縮させる関数である。一番目の引数でキャラの体力を設定し、二番目の設定でゲージを何ミリ秒で伸縮させるかを設定する。また、Update() を呼び出すことにより伸縮の計算処理を行っている。

ゲージをリアルタイムに伸縮させる計算自体は、ツールキットで提供されているゲージクラスのアルゴリズムを用いた為簡易に実装することができた。利用時にはゲージに当てはめるべき値、体力ゲージで言えばキャラクタの体力をパラメータとして与えることで拡張したクラスのメソッドが処理を行う。アクションゲームの GUI を実装することにより、Level1 のツールキットの汎用性の高さを実証することができた。図 3.4、図 3.5、図 3.6 にその実行結果を示す。

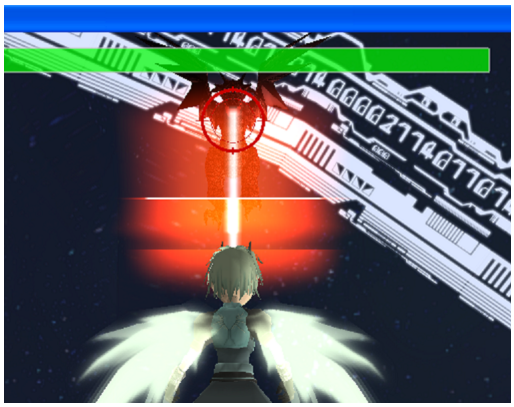

図 3.4: ゲージ変化前

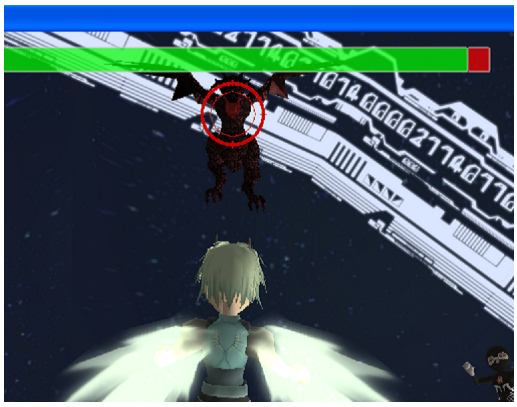

図 3.5: ゲージ変化 1 秒後

図 3.6: ゲージ変化 2 秒後

3.2 考察

今回の研究ではツールキットを用いてコンフィグ画面と実際のゲームプログラムへの利用という 2 つの実装を行った。ツールキットを利用することによって GUI を簡易に実装可能であると実証できた。しかし現状ではいくつかの問題点が残されている。ツールキットのみを利用することでの実装はツールキットの設計を熟知しているために簡易に行うことができたが、ツールキット以外のプログラムと組み合わせて利用するには、そのプログラムを理解し、拡張をする際にはその構造を理解する必要があり、設計によっては実装が困難となった。たが、Level1 のインタフェース部品を継承し、新しいクラスを生成することにより実装を行うことができたため、Level1 のインタフェース部品が軽く拡張性に優れていることが実証できた。今後は他のプログラムとの共用を考慮したインタフェース部品を設計する必要があると考えられる。

第 4 章

まとめ

本論文の締めくくりとしてまとめと今後の展望について述べる。本研究ではゲーム用 GUI ツールキットに対して GUI を実装することにより、開発者が細かい実装過程を意識することなく、アプリケーション開発を行えるようにすることを目的とした。そして、ツールキットを利用しない場合と利用した場合による実装過程の比較を行い、より簡易に実装可能であることを実証した。しかし、第 3 章でも述べたとおり問題点も残っており、より他のプログラムとの連携を考慮した設計が必要となっている。これらを踏まえた上で今後の展望について述べる。ゲーム開発を行う上で実装の手助けを行うツールは今後益々増えていくと考えられ、プログラマは内部構造を理解することなくツールから提供されている機能を利用することにより簡易に実装できるようになるであろう。しかし、ツールが増えていく一方で提供されている機能の内部構造を理解することはプログラマにとっては重要なことである。本研究の提案手法が今後のツール開発に役立つことを願い本論文の締めくくりとする。

謝辞

本研究を締めくくるにあたり、研究の指針から開発の手法、論文の執筆と幅広いご指導ご教授を頂きました、本校メディア学部の渡辺大地講師及び、卒業研究だけでなくゲームに関する研究の心得などをご指導して下さった山路和紀氏、に心より感謝いたします。在学中に研究の手助けや、メディア学の在り方、研究者としての心得などをご指導して下さった本校大学院メディア学研究科博士課程の竹内亮太さんに感謝したいと思います。さらに、研究を進めるにあたって様々な意見を交換してくれた、本校メディア学部のゲームサイエンス卒研室のメンバーに感謝します。そして、いつも私を支えてくれた家族と、全ての友人たちに感謝します。最後に、本研究にご協力いただきました全ての皆様と、この論文に目を通してくださった全ての方々に、厚くお礼を申し上げます。

参考文献

[1] 千葉 滋, 「GUI ライブラリの仕組み　—ソフトウェア設計のケーススタディー」, 朝倉書店, 2001.

[2] Adobe, Flash, <http://www.adobe.com/jp/products/flash/flashpro/>.

[3] Dstorm, Anark Corporation,
Anark Studio, <http://www.dstorm.co.jp/products/anark/>.

[4] Tim Moss, "God of War -How the left and right brain learned to love one another-", Sony Computer Entertainment America, Santa Monica Studio, GDC 2006.

[5] 馬場 昭宏, 田中 二郎 , "GUI を記述するためのビジュアル言語 ", インタラクティブシステムとソフトウェア V, 日本ソフトウェア科学会 WISS'97, 1997.

[6] 西森 丈俊, 久野 靖"アクションゲーム記述に特化した言語", 情報処理学会論文誌 VOL.44 No.SIG15, 2003.

[7] 中澤 仁, 望月 祐洋, 徳田 英幸, "Mogul: 位置透過型分散共有ツールキットライブラリ", 情報処理学会電子図書館-研究報告「システムソフトウェアとオペレーティング・システム」- No.1988-OS-078, 1998.

[8] 門田 昌哉, "仮想 3 次元空間における統合 UI 構築ソフトウェアの開発", 未踏ソフトウェア・セミナー　in 広島, 2002.

[9] 大谷 淳平, ”Lamp: 教育的かつ実用性のあるゲームミドルウェア”, 第１回「未踏ソフトウェア創造事業」, 2004.

[10] 大久保 隆, ”3 次元画像表示機能を持つ携帯電話におけるコンテンツ作成を支援するツールキットの構築に関する研究”, 東京工科大学メディア学部卒業論文, 2002.

[11] 丹治 宏文, ”リアルタイム３ＤＣＧツールキット上での絵画調レンダリングの実装とその効果に関する研究”, 東京工科大学メディア学部卒業論文, 2002.

[12] Easy Software Products, FLTK, <http://www.fltk.org/>.

[13] JasminBlanchette, Mark Summerfield,「Qt GUI プログラミング」, SoftBank Publishing, 2005.

[14] Matthias Kalle Dalheimer, 「Qt プログラミング入門」, オライリー・ジャパン, 1999.

[15] Satyaraj Pantham,「速習 Java Swing プログラミング」, SoftBank Publishing, 1999.

[16] Microsoft, DirectX, <http://www.microsoft.com/japan/windows/directx/default.mspx>.

[17] 成田 拓郎, 大川 善邦, 大澤 文孝, 登大 遊 ,「DirectX9 実践プログラミング」, 工学社, 2003.

[18] ARB, OpenGL, <http://www.opengl.org/>.

[19] Mint(経営情報研究会),「図解でわかるソフトウェア開発のすべて」, 日本実業出版社, 2000.

[20] 渡辺 大地, FK ToolKit System, <http://www.media.teu.ac.jp>.

[21] 渡辺 大地, ”リアルタイムグラフィックスのためのツールキットに関する研究”, 慶應義塾大学　政策・メディア研究科, 1996.

[22] SQUARE ENIX, FINAL FANTASY VII, 1997, <http://www.square-enix.co.jp/uh/final_fantasy_7/>.