2007年度　卒　業　論　文

ゲーム制作における
汎用的マップエディタに関する研究

指導教員：渡辺 大地講師

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0104498
齋藤 めぐみ

2007 年度　　卒　業　論　文　概　要

論文題目

ゲーム制作における
汎用的マップエディタに関する研究

メディア学部 **学籍番号：M0104498**	**氏名**	齋藤 めぐみ	**指導教員**	渡辺 大地講師

キーワード	ゲーム制作支援、マップエディタ、プログラム設計、クラスライブラリ

近年、コンピュータ上で動作するゲームは、開発環境や表現技法の発達に伴い、目覚しい進歩を続けている。1 つのゲームを作成するためには、膨大な素材が必要になるため、素材を作るための負担がかかる。そこで、比較的どんなゲームでも共通に存在する、マップという素材を作るためのマップエディタというツールに着目した。マップエディタはその機能と用途の違いから汎用マップエディタ、特定のゲーム専用のマップエディタ、ツールの中のマップエディタの 3 種類に分類することができる。また、マップがもてる情報の種類として、マップの並びを指す地形、マップの性質を指す属性と、地形や属性のデータを処理することにより起こる、マップの中の動きを指すマップ上のイベントがある。しかし、3 種類のマップエディタ全て、3 種類の情報を持つことができるとは限らない。マップが持てる情報は、各マップエディタの長所、短所を明確化にした上で、それぞれの長所を考慮し、短所を補ったマップエディタを試作し、その提案手法の妥当性を検討する。

目 次

第 1 章

はじめに

1.1　研究背景と目的

近年、コンピュータ上で動作するゲームは、開発環境や表現技術の発達に伴い、目覚しい進歩を続けている。また、最近では、企業が開発したゲーム以外にも、個人または少人数で開発したゲームを数多く見かけるようになった。ゲームは一般的にゲームの流れや仕様を決めるゲームデザイン、絵や音を作る素材作成、それらを組み合わせるプログラミングの工程を経て制作する。デザイナーは、ゲームをやる人がどんなふうにゲームを楽しむかを考え出し、作曲家やグラフィッカーがデザインを動かすために必要な素材を作り、プログラマーがコンテンツの素材とゲームプレイヤーとをつなぐシステムを作成する。1 つのゲームを制作するためには、膨大な素材が必要になるため、素材を作るための負担がかかる。そこで、比較的どんなゲームでも共通に存在する、マップという素材に着目した。それを効率的に作成するツールとしてマップエディタがある。

マップエディタとは一般的に地図や地形を作成や編集するツール [1][2] を指す。マップとはゲーム中においてのキャラクタが移動する場所や背景のことである。素材作成の補助となるマップエディタには、機能の良し悪しや無料のものから有料なものまで含めると数多く存在する。それらのマップエディタを調べ、その機能と用途の違いから以下の 3 つに分類した。マップエディタを分類することで、それぞれの長所と短所を明確化する。

1. 汎用性のある単純なデータを扱うマップエディタ。汎用性を持たせるため、作成できるマップデータは単純であり、そのことがマップエディタの表現の幅を狭めている。

2. 制作するゲームを想定したそのゲーム専用のマップエディタ。あらかじめそのゲームのデータ構造を特定できるので、表現に幅のある高機能なマップエディタが多いが、このマップエディタ自体に依存するため汎用的に使うことを目的としない。

3. ゲーム制作用ツールに付随するマップエディタ。制作用ツールでは、ツール内に用意してある命令、画像、音楽などのパーツをマップ上に組み合わせながら置いていくだけでマップを作ることができるが、ツールという枠組みを越えてのマップ制作はできない。

マップエディタを分類することで、それぞれの長所と短所を明確化し、長所を考慮し、短所を補ったマップエディタを提案する。マップエディタには、マップが持てる情報の種類に関しての問題とシミュレート機能に関しての問題が挙げられる。まず、マップが持てる情報の種類に関しての問題点を述べていく。マップが持てる情報の種類として地形、属性、マップに付随している様々な処理の 3 種類がある。地形とはマップの並びのことで、属性はマップの特徴や性質のことを指す。また、マップ上で起こるできごとをイベントと呼ぶこととし、マップの中でイベントを起こすことにより地形や属性が変化する。マップを表現するには、基本的に 2 次元の方眼状のデータで管理 [3] し、1 マスごとに地形、属性、マップに付随している様々な処理の 3 種類の情報を含んでいる。複数のマスを組み合わせることによって 1 つの大きな地形にすることもできる。3 種類のマップエディタが全て、3 種類の情報を持つことができるとは限らない。マップが持てる情報の種類は、各マップエディタの特徴と情報の表現の仕方により変わってくる。

汎用マップエディタでは、基本的に 2 次元の方眼状のデータで管理しているため、地形、属性の表現はできるが、データと処理の対応関係を定義することができな

いためマップに付随する様々な処理を表現することはできない。このことが、汎用マップエディタで作るマップの表現に限界を持つことになる。特定のゲーム専用のマップエディタでは、あらかじめデータと処理を固定することができるため地形や属性の表現はもちろん、マップに付随する様々な処理も表現できる。しかし、データと処理の対応関係を固定するので汎用性に欠ける。ツールの中のマップエディタでは、もともとがゲーム制作用ツールのためマップ作成に必要なデータと処理が備わっているため、地形、属性、マップに付随している様々な処理と表現できる。しかし、ツールの中で提供している機能以上のことができないのでマップエディタ自体の拡張性に欠ける。

次にシミュレート機能に関しての問題点を述べていく。実際にマップエディタで作成したマップを読み込んだプログラムを実行した際、本来想定していたものと違う場合がある。そのため、マップが実際の実行画面に対し本当に適格かどうかの検証が必要であり、検証するためのシミュレートプログラムが必要である。本研究においてのシミュレーションとは、想定するマップ画面を描画し、マップに関する様々なマップの中の動きとマップ全体のバランスを確認することである。現存する汎用マップエディタでは、マップエディタとシミュレートプログラムがそれぞれ別々のソフトウェアとしてできている。そのため、マップ作成とマップ確認の作業を 2 つのソフトウェアの間で繰り返し行うことになるため効率が悪い。特定のゲーム専用マップエディタは、あらかじめそのゲームのデータ構造を特定してあるため、特定のゲーム専用マップエディタで作成したマップしかシミュレートすることができない問題がある。以上のことをふまえ、3 種類のマップエディタの長所を考慮し、短所を補ったマップエディタを提案する。本研究ではデータと処理の対応関係を自由に定義可能で、拡張性があることを考慮し、次の 2 点を満たすシステムを提案する。

- 任意のプログラムとの機能を共有できる。
- 機能が拡張できる。

本論文では、シミュレート機能をクラスライブラリにしてマップエディタで利用する手法を提案した。そして、クラスライブラリとマップエディタを組み合わせるという手法を用いた新しいマップエディタを試作し、提案手法の妥当性を検証する。

1.2 論文構成

本論文では第２章で既存マップエディタと本研究のマップエディタの位置づけについて述べ、第３章でクラスライブラリを用いたマップエディタの提案について述べる。第４章で、本論文の手法を実装した結果の検証と考察を行い、第５章で研究のまとめを述べる。

第 2 章

既存のマップエディタと本研究のマップエディタの位置づけ

2.1　マップエディタとは

ゲーム制作においてのマップを作成、編集するツールをマップエディタという。マップとはゲーム中においてのキャラクタが移動する場所や背景のことで、ゲームの種類によりその見た目や機能も違う。図 2.1 と図 2.2 はそれぞれ典型的なゲームのマップである。図 2.1 はロールプレイングゲーム [4][5][6]（以下、RPG）のマップで、図 2.2 はシューティングゲーム [7][8]（以下、STG）のマップである。ロールプレイングゲームはストーリー性とプレイヤーの演じるキャラクタの成長を特徴とし、シューティングゲームは弾を撃つことで敵を倒していく。

図 2.1: RPG マップの例

図 2.2: STG マップの例

表 2.1: マップチップと対応した数値

2	1	1	1	0
4	2	2	1	1
0	0	0	3	1
0	2	2	1	0
0	1	1	1	0

RPG のマップでは、何らかの意味のある物体を配置し、キャラクタがそのマップの中の世界を移動する。STG のマップでは、マップに敵を配置することができるが、自機が浮いているために自機の移動を表すための流れる背景としての要素が大きい。

ゲームで使われている 1 画面のマップをデータとして見ると、基本的に 2 次元の方眼状に置かれている物に対応した数値の羅列になっている。図 2.3 は一般的なゲームマップの一部を表したものであり、図 2.4 は「草地」「岩山」「家」のように単体の小さな画像であるマップチップの集まりである。ゲームのマップはこのマップチップをタイル状に並べて、1 枚の画像にする。マップチップを 1 つ 1 つの画像に固有の番号を割り振ることで識別し、その番号を表 2.1 はマップチップと対応した数値表を表している。プログラムにより、番号に対応した画像を貼り付けることで、マップができる。このように画像の再利用ができるため、マップを 1 枚の画像として扱うよりもデータ量が少なくてすむ。

図 2.3: ゲームのマップ

図 2.4: マップチップの集まり

しかしながら、プログラム中に数値の羅列を打ち込むには時間と手間がかかる上に数値を間違えても、実際にプログラムを実行するまではわかりづらい。表2.1のように、打ち込む数値が少なければよいが、マップを大きくすればするほど数値も増え、プログラムに打ち込む手間もかかる。そこで、マップエディタを使うことにより、この問題を解決することができる。図2.5のマップエディタでは、あらかじめ用意したマップチップ画像を読み込み、選択したチップを左のマップに置いていくことによりマップを作成するものである。マップが完成したら、マップチップ画像の並びをデータ化し、テキストファイルやCSVファイルなどの汎用的なデータとして出力できる。この、出力したデータをプログラム内で呼び出すことで、プログラム中の数値の打ち込みの手間を省くことができる。また、マップエディタを利用することにより、視覚的にマップが作成できるため、マップ全体の見た目のイメージがしやすい。

図2.5: マップエディタ

2.2　既存マップエディタの種類

マップエディタには、マップを作成する、マップデータを出力するという基本的な機能があるが、他にも様々な機能を持つマップエディタがある。マップエディタをその機能と用途の違いから以下の 3 つに分類した。

2.2.1　汎用マップエディタ

まず挙げられるのは、汎用的なマップエディタである。このマップエディタの中にはマップを制作する、マップデータを出力するという基本的な機能以外にキャラクタとマップ上に配置した物との当り判定の設定、イベントのフラグの処理などの機能があるマップエディタも存在する。また、用意する画像を変えることにより、マップエディタ 1 つでいろいろな種類のゲームマップを作成することができる。作成したマップデータは他のマップエディタでも扱えるデータとして出力し、プログラムで活用することができる。しかし、基本的に扱えるデータ構造が 2 次元の方眼状の上でのことなので、どんなマップチップでも扱えるが、地形と属性を固定するため表現できる機能は限られる。このように、1 つのマップエディタで多様なマップを作成でき、かつ、汎用的に利用できるマップデータを出力することを目的としたマップエディタを汎用マップエディタと呼ぶ。図 2.6 は汎用マップエディタの例である [9][10][11]。

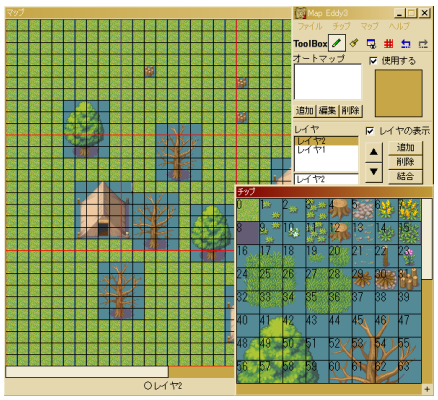

図 2.6: 汎用マップエディタ

2.2.2　特定のゲーム専用のマップエディタ

次に挙げるのは、制作するゲームを想定したそのゲーム専用のマップエディタである。このマップエディタは、あらかじめデータ構造を特定できるので、特定のゲームを作るうえで必要だと思う機能や効果などをいくらでも付けることができる。しかし、ゲーム制作をする前にマップエディタを作るという労力がかかり、制作するゲームを想定しているため、このマップエディタを使って作ったゲームは想定したゲームしか作れない。このように、1つのプログラムに依存するため汎用的に使うことを目的としないマップエディタを特定のゲーム専用のマップエディタと呼ぶ。図 2.7 は特定のゲーム専用のマップエディタの例である [12][13][14]。

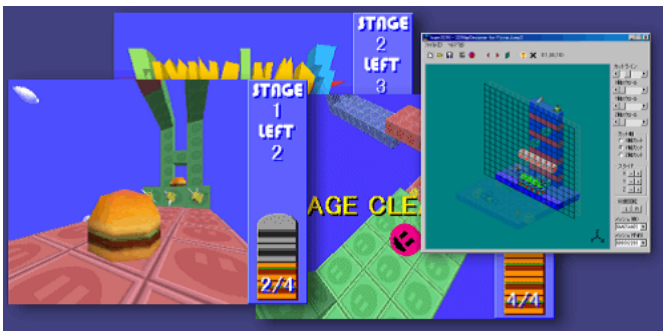

図 2.7: 特定のゲーム専用のマップエディタ

2.2.3　ツールの中のマップエディタ

最後に挙げるのは、ゲーム制作用ツールに付随するマップエディタである。特に、プログラム経験のない人でも、簡単にゲームを作ることを目的とするシステムとして、RPG ツクール [15] がある。RPG ツクールではツール内に用意してある命令、画像、音楽などのパーツをマップ上に組み合わせながら置いていくだけでゲームを作ることができる。画像、音楽はあらかじめ用意されたものの他、自分で作ったものも使用できる。また、RPG ツクール XP では RGSS[16] という Ruby[17] をベースとしたスプリクト言語を用い、それまでにできなかったゲームシステムの拡張を可能とし、命令の実行を応用すれば、RPG 以外のゲームも作ることができ

る。RGSS により、ゲームの中身を拡張することは可能だが、エディタ自体を拡張することはできない。結局のところ、RPG ツクールのエディタの中では決められたデータしか作ることができない。このように、RPG ツクールなど、ゲーム制作ツール中にあるマップエディタをツールの中のマップエディタと呼ぶ。図 2.8 は RPG ツクール XP である。

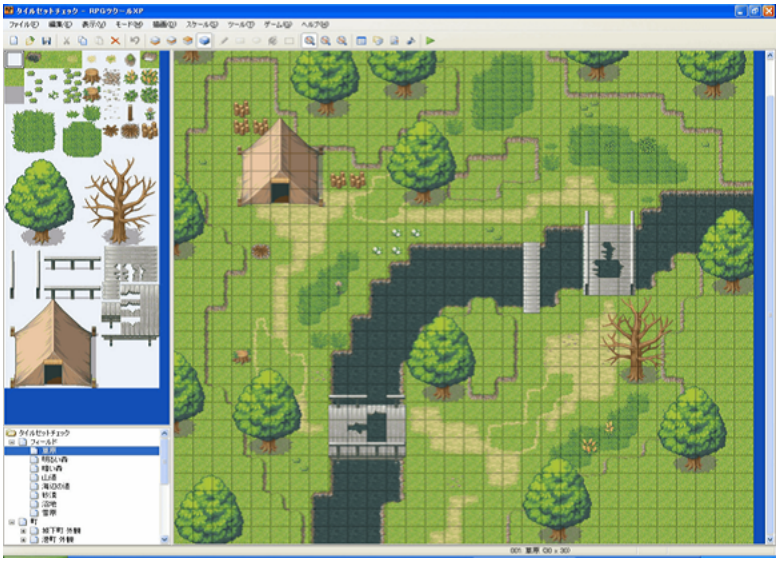

図 2.8: RPG ツクール XP

2.3　マップが持つ情報の種類

マップが持つことができる情報には地形、各マップが持っている属性、マップに付随している様々な処理の 3 種類がある。

第 1 の地形とは、土地の形状つまりここではマップチップの並びのことを指す。マップチップ画像が変わることによって、地形も変えることができる。図 2.9 はマップの地形を表したものの例である。山のマップチップや木のマップチップを用意し、単品で山や木として表現できるが、複数並べることにより、山脈や森を表現することができる。

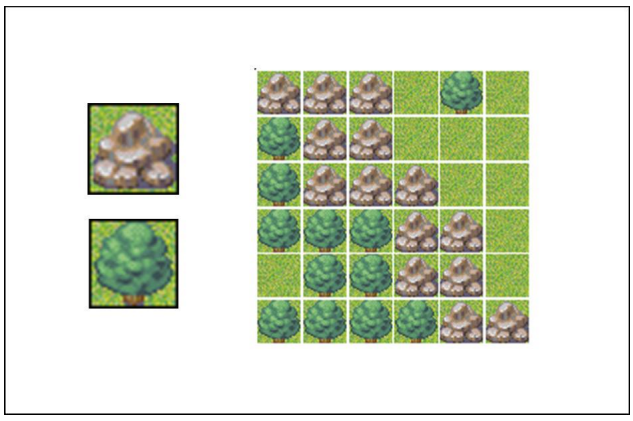

図 2.9: マップの地形

第 2 の各マップが持っている属性とは、マップチップの特徴や性質のことを指す。マップの中にはキャラクタが通行可能の場所や通行できない場所、通行するとキャラクタに何らかの影響がでる場所が存在する。図 2.10 は、マップの属性を表したものの例である。毒沼やマグマのチップ上を歩くたびにキャラクタがダメージを受けたり、氷のチップ上を歩くと滑り、一定時間キー操作を効かなくしたり、落とし穴のチップ上に行ったら、落ちて別のマップに移るなどの属性を持つ。

第 3 のマップに付随している様々な処理とは、マップに関する様々なマップの中の動きのことを指す。キャラクタがある行動をしたら、マップもその行動によって変わるということで、例えば、キャラクタがマップ上で剣を振り回すことで、草木

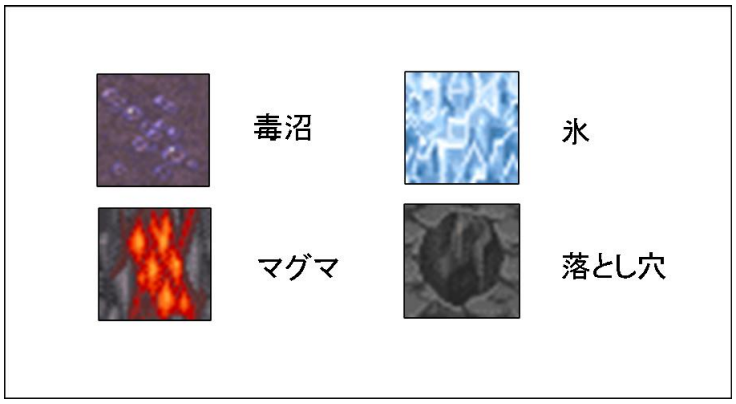

図 2.10: マップの属性

を刈ったり、マップ上に置いてある箱や樽などを壊したり、図 2.11 は、マップ中のイベントの様子を表す。水路のマップがある場合に水があると通れないが、あるスイッチを押すことにより、水が引き水路を通れるようにするなどがある。刈られた草木や壊された箱や樽は、その残骸や破片の表示もしくはマップ上から消し、水路は水がある水路から水がない水路にすることにより、マップは変化する。このように、マップ上で起こるできごとをイベントと呼び、イベントを必要なだけ作成することにより、ゲームのシナリオを表現することができる。

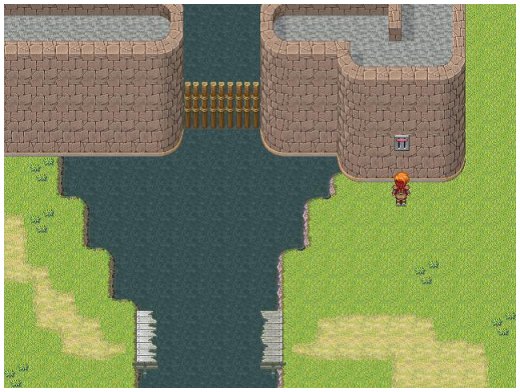

図 2.11: マップ中のイベント

2.4　本研究の位置付け

マップには地形、属性、イベントの 3 種類の情報を持つことができるが、汎用マップエディタ、特定のゲーム専用のマップエディタ、ツールの中のマップエディタのマップエディタがすべての情報を持つことができるわけではない。それは、マップエディタの特徴と情報の表現の仕方に関係してくる。地形を表現するには、草地が 0、山が 1、木が 2 で、属性を表現するには、通行不可の場所が 0、通行可能の場所が 1、毒沼が 0、マグマが 1、氷が 2、落とし穴が 3 など、自分で決めることができる。プログラム側では、地形の番号に対応したマップチップを読み込むことで、マップを表示したり、属性が 0、1（毒沼、マグマ）のときは、キャラクタの体力を減らすような処理、2（氷）のときは、滑る処理、3（落とし穴）のときは、効果音などをつけて他のマップに移動する処理を作る。このようにマップの地形や属性の情報を表現するときは、数値の羅列で表現し、そのデータを用い実際の処理はプログラム側で行う。これらは典型的なデータを扱うものなので、汎用マップエディタ、特定のゲーム専用のマップエディタ、ツールの中のマップエディタの全てで扱うことができる。

一方、マップ上で発生するイベントなどを表現するには、地形や属性のように 2 次元の方眼状の数値だけで表現することはできない。なぜならば、イベントを起こすことにより、地形や属性が変わる場合に対応できないからである。図 2.12 はイベントの情報を表した例である。あるスイッチを押すと地面にマグマが流れ、足元が盛り上るイベントを表しいている。スイッチを押す前と押した後では、地面の属性が異なり、地形も変化する。このことから、マップ上で発生するイベントでは、地形や属性などのデータが必要で、このデータを処理によって変化させることで、マップ上の見た目が変わる。

汎用マップエディタでは、基本的に 2 次元の方眼状のデータで管理しているため、地形、属性の表現はできる。しかし、その汎用性からデータと処理を固定することができないため、マップに付随する様々処理を表現することはできない。デー

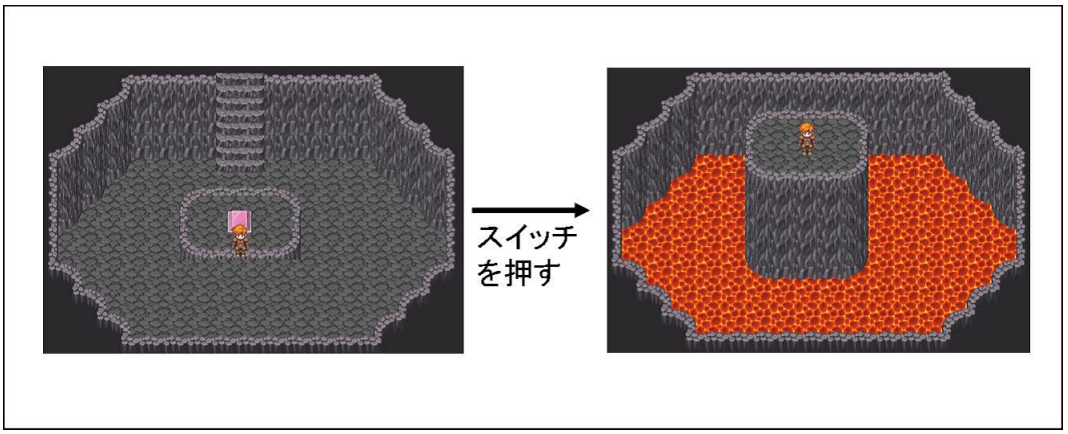

図 2.12: イベントの情報の表現

タと処理がセットになっていないと、一定の時間ごとに変化するものや、ある時間を越えたら自動的に発生するイベントなど、インタラクティブなゲーム [18] を作るのは難しい。

特定のゲーム専用のマップエディタでは、あらかじめデータと処理を定義できるため、地形や属性の表現はもちろん、マップに付随する様々な表現もできる。しかし、データと処理が固定しているためデータが変わった場合など対応できないという汎用性に問題がある。

ツールの中のマップエディタでは、もともとがゲーム制作用ツールのためゲームに必要なデータと処理が備わっているため、地形、属性、マップに付随している様々な処理を表現できる。しかし、ツールの中で提供している機能以上のことができないのでマップエディタ自体の拡張性に欠ける。マップエディタ自体が拡張できるようになれば、エディット機能をあげることができる。表 2.2 は各マップが扱える情報の種類を表している。

このように、汎用マップエディタでは、その性質からデータのみしか扱えないため、表現に限界を持ち、このことから高機能を実現することができない。特定のゲーム専用のマップエディタでは、あらかじめデータ構造を定義してしまうため、汎用性がないことに加え、ゲーム開発の度に専用のマップエディタを制作するのにゲーム制作とは別に制作工数がかかる。また、使い手の用途に合わせるために

表 2.2: 各マップエディタが扱える情報の種類

	汎用マップエディタ	専用マップエディタ	ツールマップエディタ
地形	○	○	○
属性	○	○	○
イベント	×	○	○
汎用性	○	×	△
エディタの拡張性	△	○	×

はマップエディタ自体の拡張もできる必要がある。そこで、本論文では、データと処理の対応関係を自由に定義可能で、任意のプログラムとの機能を共有かつ拡張ができるシステムを提案する。

第 3 章

クラスライブラリを用いたマップエディタの提案

本研究で想定するマップエディタとは、マップエディタで簡単なシミュレーションができることである。これを実現するために、マップエディタとクラスライブラリがセットになった新しいマップエディタを提案する。

3.1　マップエディタ上でのシミュレート

マップエディタで作成したマップを実際のプログラムで見たときに本来想定していたものと違う場合がある。そのため、マップが実際のプログラムに対し本当に適格かどうかの検証が必要であり、検証するためのシミュレートプログラムが必要である。本研究においてのシミュレーションとは、想定するゲーム画面を描画し、マップ中のあらゆる場所を任意の速さで移動できるようにして、マップに関する様々なマップの中の動きとマップ全体のバランスを確認するものである。現存する汎用マップエディタでは、マップエディタとシミュレートプログラムが別のアプリケーションソフトでできている。マップエディタで作成したマップデータをシミュレートプログラム側で呼び出してマップのシミュレーションを行う。しかし、この方法だと試行錯誤しながら作成するマップの場合、最終的にマップが完成するまでにマップ作成とマップ確認の作業を 2 つのアプリケーションの間で繰り返

し行うことになるため効率が悪い。また、汎用マップエディタでは、マップデータを汎用的に使うため、どのプログラムでも共通に扱えるデータ構造しか持つことができない。汎用マップエディタのデータ構造は2次元の方眼状になっており、このデータ構造だとデータのみしか扱えないため、マップ上での様々な機能の表現と、マップが変形する場合に対応できない問題がある。

3.2 シミュレート機能を持つライブラリの設計

汎用マップエディタのように、2つのアプリケーション間で作業を行う必要がある場合、その作業効率の悪さを解決するために単一のアプリケーション内での作業にする必要がある。また、汎用マップエディタはデータしか扱えないためにマップ上での表現に限界がある。これらの問題を解決するには、データとそのデータを制御するための処理が必要である。この方法をすでに実現しているのが、特定のゲーム専用マップエディタである。しかし、特定のゲーム専用マップエディタは、あらかじめデータ構造を特定してあるため、特定のゲーム専用マップエディタで作成したマップしかシミュレートすることができない。そこで、作成するゲームによって変わるマップデータを扱えるシミュレート機能をマップエディタにつけることで、マップ作成の作業効率を良くし、様々なマップのシミュレーションをすることができる。シミュレート機能をクラスライブラリにし、マップエディタで利用する手法を提案した。クラスライブラリにしたシミュレート機能は、ユーザプログラムなど任意のプログラムにリンクすることで、処理の共有をすることができる。図3.1はマップエディタとユーザプログラムのシミュレート機能の共有を表したものである。マップエディタはエディタで作成したマップデータを用いて描画し、ユーザプログラムでは任意のマップデータを読み込んで描画する。

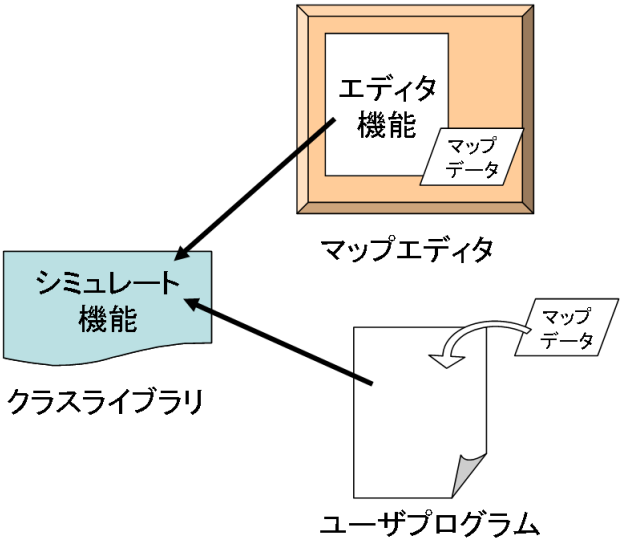

図 3.1: シミュレーション機能の共有

読み込んだマップデータを描画する機能は同じなので、マップデータが同じならば、マップエディタ上でも、任意のプログラム上でも、同じ画面を描画する。図 3.2 は、同じマップデータを使った場合のシミュレート画面を表している。

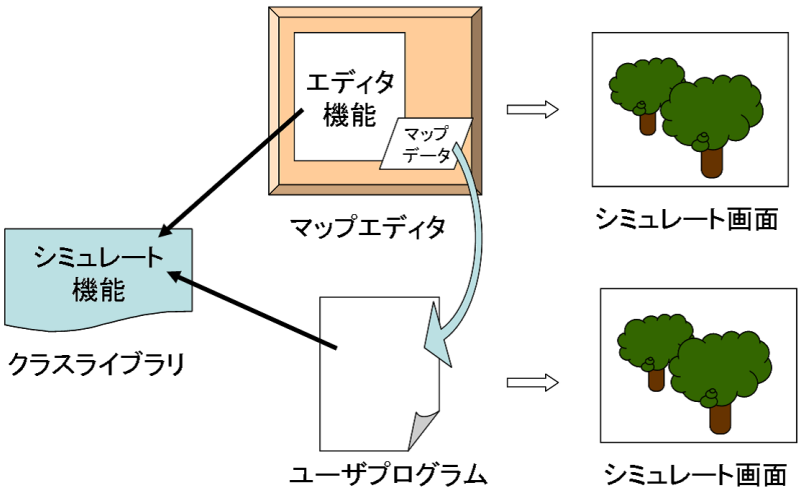

図 3.2: マップデータが同じ場合のシミュレート画面

第 4 章

動作検証

4.1 実装

前章で述べた本研究の手法を用いて実際にクラスライブラリを作ってみた。今回の開発では、開発言語としてC++を使用し、開発のための統合環境として、Microsoft Visual Studio.NET 2003[19] を、実装には、3 次元グラフィックツールキットである「Fine Kernel Tool Kit」[20] を利用し、プログラミングを行った。

4.2 実行結果

まず、マップデータを作成した。マップエディタのエディット機能で、任意の素材を置いていき、その素材データのある場所と素材を配置した場所データをテキストファイルに出力する。図 4.1 は読み込んだ素材を実際にマップ上に配置した図である。図 4.2 はマップ上に配置した素材とその位置データをテキストファイルに出力したものである。

図 4.1: マップエディタでのオブジェクトの配置

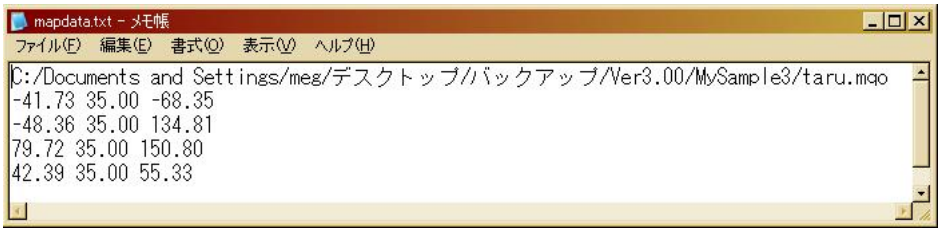

図 4.2: マップのデータが記述してあるテキストファイル

次に、シミュレート機能を持つライブラリを作成した。シミュレート機能を持つライブラリをマップエディタとユーザプログラムで結びつけることによって処理を共有する。図 4.3 はマップエディタでのシミュレート実行画面で、図 4.4 はユーザプログラムでのシミュレート実行画面である。同じマップデータを用いているため、2 つの実行画面は同じになった。

次に、マップ上でのイベント発生によるマップ地形を変形する処理を持つライブラリを作成した。マップエディタのエディット機能の地形の変形により、イベント発生後の地形の形を操作している。図 4.5 はマップエディタのエディット機能で地形を変えているところである。シミュレートボタンを押すと、シミュレート画面

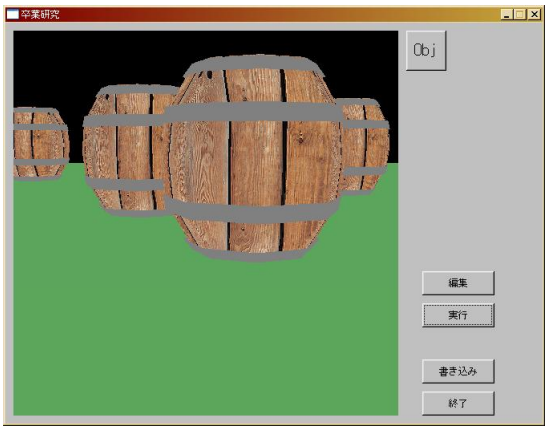

図 4.3: マップエディタでのシミュレート画面

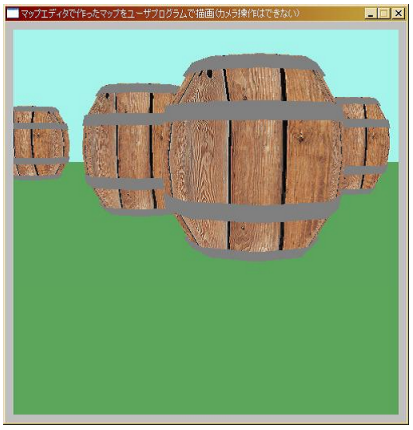

図 4.4: ユーザプログラムでの画面

に移行する。今回は、自由に動き回れるキャラクタを用意し、マップ上に設置したスイッチを押すことで、地形が変化するという簡単なイベントを想定した。図 4.6 はイベント処理のシミュレーションに移行したときの画面を表している。画面真ん中の物体がキャラクタを表し、左上の四角い物体がスイッチを表す。図 4.7 はキャラクタがスイッチを押したときの画面である。キャラクタが移動し、スイッチを押すことによって、エディット機能で操作した、地形に変わる。

エディット機能で変形した地形のデータをテキストファイルに出力し、このデータとイベント処理による地形の変化の機能を使いユーザプログラムでも、マップエディタ上で起こったことと同じことができた。図 4.8 はユーザプログラムでの画面

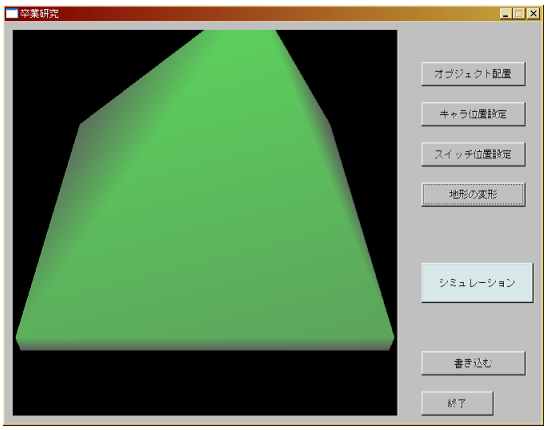

図 4.5: マップエディタのエディット機能で地形操作をしている画面

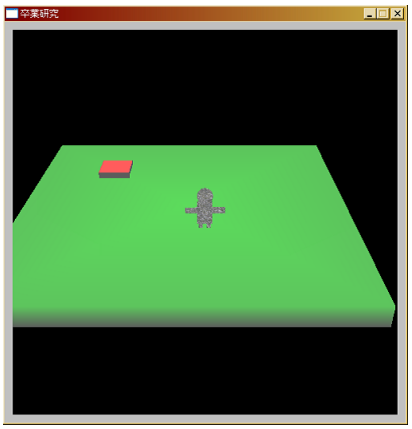

図 4.6: マップエディタでのイベント処理のシミュレート画面

を表している。図 4.9 は、ユーザプログラムのイベント処理結果を表している。

図 4.7: マップエディタでのイベント処理の結果

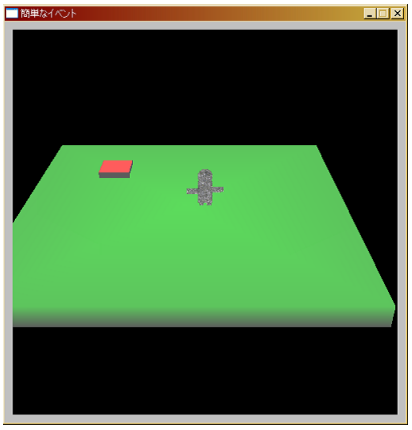

図 4.8: ユーザプログラムのイベント処理画面

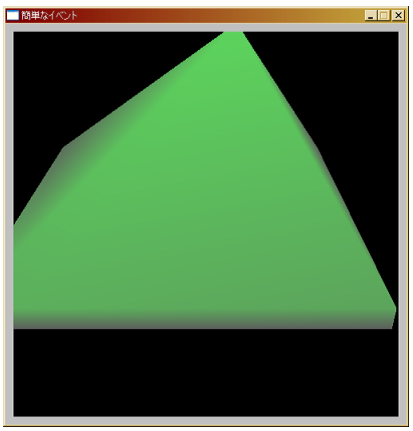

図 4.9: ユーザプログラムのイベント処理結果

第 5 章

おわりに

本手法を用いることで、汎用マップエディタでは不可能な高機能を、特定のゲーム専用のマップエディタでは不可能な汎用性を、ツールの中のマップエディタでは不可能なエディタ自体の拡張を可能にすることができた。このことにより、誰が使っても、どんなゲームを作るにしても、マップエディタを汎用的に使うことが出来る。自分の作りたいゲームに欲しいと思う機能を自分で作り、ライブラリ化することでその後、何度でもその機能を使うことができ、プログラミングの余計な手間を省くことができる。しかし、実際に自分の欲しい機能を作るときに、どのようなプログラムにも対応できる汎用性の高いものにしなくてはならないことは、いかに、熟練したプログラマと言えどもとても困難なことでである。今後、マップエディタを拡張するときのライブラリ作成を支援する概念やツールができると、作る側になって扱いやすい、さらに画期的なマップエディタになると言えるだろう。

謝辞

最後に、この論文を書くにあたり、ご指導していただいた渡辺大地講師、ならびに様々なアドバイスをしていただいた講師の方々、研究についての相談を受けてくれた院生の方々、共に励ましあい研究に協力してくれた学部生の皆様に厚く御礼申し上げます。

参考文献

[1] 谷謙二, ”学校教育用 GIS に求められる条件とその開発” 埼玉大学教育学部地理学研究報告, vol.20, pp.20-26, 2000.

[2] 地理情報分析支援システム MANDARA ”http://www5c.biglobe.ne.jp/ mandara/index.html”.

[3] 有馬元嗣, ”ゲームプログラミング 遊びのレシピ　－アルゴリズムとデータ構造” ソフトバンククリエイティブ, 2001.

[4] 坂本千尋, ”ロールプレイングゲーム プログラミング” ソフトバンククリエイティブ, 2001.

[5] アルゼ株式会社　アルゼ君　第 157 回　ロールプレイングゲームって何?ゲームアナリスト　平林久和”http://www.aruze.com/company/column/aruzekun/2005/157.pdf”.

[6] アルゼ株式会社　アルゼ君　第 158 回　RPG とは現代の神話であるゲームアナリスト　平林久和”http://www.aruze.com/company/column/aruzekun/2005/158.pdf”.

[7] 松浦健一郎, ”シューティングゲーム アルゴリズム マニアックス” ソフトバンクパブリッシング, 2004.

[8] 司ゆき, ”シューティングゲーム プログラミング” ソフトバンククリエイティブ, 2006.

[9] Map Eddy3 ”http://punk-peace.sakura.ne.jp/”.

[10] Platinum ”http://www.hyperdevice.net”.

[11] ANoME ”http://page.freett.com/aninoiser/”.

[12] Flying Jump3 ”http://www.sunflat.net/ja/win/fly3.html”.

[13] 大戦略　マップエディタ”http://www.ss-alpha.co.jp/download/d7editor.html”.

[14] Valve Hammer Editor ”http://www.geocities.co.jp/Playtown-Knight/6330/hammereditor/index.html”.

[15] ツクールWeb ”http://www.enterbrain.co.jp/tkool/”.

[16] RPGツクールXP 新機能『RGSS』”http://tkool.jp/products/rpgxp/shinkinou.html”.

[17] 高橋征義, 後藤裕蔵, ”たのしいRuby 第２版　Rubyではじめる気軽なプログラミング” ソフトバンククリエイティブ, 2006.

[18] 中野敦, 河村仁, 三浦枝里子, 星野准一, ”Spilant World: エピソードツリーによるインタラクティブなストーリー創発型ゲーム” 芸術科学会論文誌, Vol. 6, No. 3, pp.145-153, 2007.

[19] Microsoft Visual Studio ”http://www.microsoft.com/japan/msdn/vstudio/”.

[20] 渡 辺 大 地, ”Fine Kernel Tool Kit System” ”http://www.teu.ac.jp/media/ earth/FK/”.