平成25年度 卒業論文

クラウドソーシングを用いたゲームのレベルデザイン効率化

Improvement of game level-design via Crowdsourcing.

北海道大学　工学部
情報エレクトロニクス系　情報工学コース
複雑系工学講座　表現系工学研究室

三木 康暉

平成25年2月14日

目 次

第1章　はじめに

1.1　研究背景

今日、日々、商用、非商用問わず多数のゲーム作品が世に生み出され続けている。開発環境のオープン化と一般化が進み、個人でも簡単にゲーム作品を世に輩出することが可能になったことが一因としてあげられる。

同時に、世に出回るゲーム作品が増えたことで、商品として要求されるゲームの質、物量が一昔前と比べると、格段に肥大化し続けている。重厚長大化するゲーム作品が求められることで、ゲーム開発はますます複雑化し、多数の人手、多額の資金、長期の開発期間、熟練された開発ノウハウが求められるようになってきた。

その一方で、ゲーム作品の単価は急速に低下しており、一作品数千円というパッケージ販売の時代から、一作品数百円や無料でプレイが可能なダウンロード主体の市場に変化しつつある。そのため、ゲーム開発者は、如何に少ない労力で、高品質なゲームを生産する必要性に迫られている。

複雑化するゲーム開発の中でも、特にゲームのバランス調整や不具合検出と言ったゲームバランスの調整作業は多大な労力を要する。ゲーム中に利用されているパラメータ群の数が肥大化し、最適な組み合わせを発見する作業が困難を極めるからだ。ゲーム開発の現場において、ある特定のゲームルールにおいて、ゲームパラメータの調整を行い、プレイヤーにとって適切な難易度を提供したり、ゲームルールの解説を行えるように設計する作業はレベルデザインと呼ばれている。その中には、面クリア型のゲームにおけるステージ作成なども含まれる。レベルデザインを高品質化することは、ゲーム作品自体のおもしろさ、ひいては売り上げにまで直結する。質の高いレベルデザインを行うことはゲームを開発する意義そのものと直結すると言っても過言ではない。しかし、レベルデザインを効果的に行う手法は未だ確立しておらず、調整作業は多数のテストプレイヤーを動員し、考え得るテストパターンを網羅的に行っているのが現状だ。

仮にこのような手法が発見されれば、ゲーム開発の生産性の向上に大きく寄与するであろう。

また、ここ数年、一般的に遊ばれるゲームの質の変容も著しい。従来は、商用ゲーム業界においてはパッケージ型の販売が主流であった。そのため、一度販売し、リリース後にプレイヤーのフィードバックをゲーム自体に反映する作業は必要性に乏しく、

次回作に生かされる程度であった。それに対し、近年は世界的に、MMO[1]と呼ばれる大規模オンラインゲームや、スマートフォン環境下でプレイ可能なソーシャルゲームへの市場の移行が進んでいる。これらの形態のゲーム作品は、従来のパッケージ型の売切りのゲーム作品と異なり、リリース後もユーザーのプレイ状況やフィードバックを受け、継続的にサービスを改善、強化させ続けていくことが求められている。そのため、ゲームパラメータの調整を、実際のプレイヤーログを集計して行う手法やシステムの重要度がより一層増している。

これらの理由から、プレイヤーログを用いた効率的なゲームの不具合の検出、 レベルデザインの改善手法の提案はゲーム開発者にとって、非常に有用な物と推測できる。

1.2 研究目的

本研究では、重厚長大化が進み、ゲーム開発者による手作業での調整が困難になりつつあるゲームパラメータの調整について、プレイヤーのログを集計し、不具合検出、修正を効率的に行える手法を提案する。この検出手法は、ゲームの難易度をプレイの進捗に応じて、適切な成長曲線を描くように設定することがゲームバランスの改善に繋がる、という仮定に基づいている。同時に、従来、多数のテストプレイヤーを動員する必要があったプレイログの収集をクラウド化し、収集、解析、可視化を Web 経由で簡便に行えるシステムの構築について模索する。

1.3 論文構成

本論文の残りの構成は下記の通りである。

第 2 章　ゲームバランスの調整、ゲーム開発手法の提案、ゲームプレイヤーログの解析などと言った関連研究について触れ、これらの既存研究と本研究の差異、意義について考察する。

第 3 章　研究目的を達成するために、今回提案する手法と仮説を提示する。また、プレイヤーログ収集システムの設計について解説する。

第 4 章　実際にシステムを用いたプレイヤーログ収集について、実際に実験に利用するゲームのルール解説を交え、実験方法について述べる。

[1] Massively Multiplayer Online-Game

第 5 章　本研究の結論を述べ、今後の課題を提示する。

第2章　関連研究

2.1　プレイログの収集に関する研究

プレイヤーのプレイログの収集から、プレイヤーのゲームプレイを定性的に捉えようとした研究は多数存在する。プレイヤーのプレイを集計し、ユーザビリティの向上を目指した研究 [1][2]、任天堂の「スーパーマリオブラザーズ」を元に作られた、オープンソースソフトウェア「Infinite Mario Bros」[3] を用い、プレイヤーのプレイログをモデル化する手法を提唱した研究 [4][5] など、多数の先行研究が存在する。

2.2　レベルデザインの改善に関する研究

レベルデザインの改善をテーマにした研究としては「The 2010 Mario AI Championship Level Generation Track」[6] が挙げられる。これは先ほど挙げた「Infinite Mario Bros」を用いたコンテストである Mario AI Championship[7] の中で、とりわけ、レベルの自動生成を目的とした部門、Level Generation Track を題材とし、レベルデザインの自動化手法について多数紹介している。また、複雑化したゲームパラメータの調整を最適化することを目標として掲げた研究として、「遺伝アルゴリズムの視覚化を用いたゲームのレベルデザイン効率化技法の開発」[8] が本研究と酷似している。この研究では、複数のゲームパラメータの調整を最適化問題としてとらえ、遺伝的アルゴリズムを用いた手法の発見を模索した。

2.3　ゲームの自動生成に関する研究

レベルデザインのみならず、ゲームシステムそのものの自動生成を試みた研究も存在する。

特定の単語について、辞書データを元に適切なゲームを選び出し、当てはめる研究 [9] や、ゲームルール自体を一から自動生成する研究 [10] などが挙げられる。

2.4　既存研究との差異

これらの既存研究と比較し、本研究においては以下の特徴が挙げられる。

- プレイヤーへのゲームの配信、プレイ、評価、ログの集計など、一連の操作をクラウド上で行えるシステムを提案したこと
- このシステムを大規模に運用し、大量のプレイヤーログの収集を行ったこと
- 特定のゲーム作品だけではなく、様々なゲーム作品に汎化して適応可能なシステムを模索したこと

第3章　提案手法

3.1　用語定義

本稿で多用されている用語は定義が曖昧であり、またゲーム開発者以外には馴染み浅い言葉である。

そのため、本稿で用いられている用語について以下に解説する。

3.1.1　レベル

「レベル」とは、広義ではゲーム中における一定のある小単位を示す。

ゲームの種類によっては「ステージ」、「マップ」、「面」などと呼ばれることも多い。

一般的に扱われるレベルという単語は、難易度やプレイヤー習熟度を包括した意味合いで用いられることが多いが、本稿での「レベル」はプレイヤー熟練度の意味は成さない。

3.1.2　レベルデザイン

前項で解説した「レベル」を具体的に設計、パラメータの調整を行うことをレベルデザインと呼び、それを行う開発者をレベルデザイナーと呼ぶ。

また、広義では、「レベル」の設計のみならず、ゲームルールの設計そのものを行うことを「レベルデザイン」と呼ぶことがあるが、本項では「ゲームデザイン」の意味で用いられるレベルデザインとは区別する。

3.2　システムの目的

今回、設計を目指すシステムは、クラウドを用いてプレイヤーのプレイログを集計し、プレイログの収集や可視化、またそれを用いたゲームレベルの不具合検出や自動改善を目的とする。

3.3　仮説

3.3.1　プレイバリューを向上させるレベルデザイン

ゲーム開発の現場においては、一般的に以下の点を満たすレベルを良質なレベルと見なしている。

レベルの序列

レベルの難易度の序列が、ゲームのプレイ進行度に応じて徐々に上昇していくこと。
また、新たなゲーム中の要素が順序立てて出現し、プレイヤーへの手解きを兼ねていること。

クリア可能であるか

全てのレベルがクリア可能であること。いわゆる「ハマり」が起こらないこと

極端に高難易度なレベルがないこと

極端にクリア率の低い難易度を持つレベルが入り込まないこと。

3.4　理想的なシステムの動作

下記のようなユースケースを満たすシステムが理想的なシステムと考えられる。

1. レベルデザイナーがあるゲームのレベルを複数登録する
2. レベルデータをランダムにプレイヤーに配信する
3. プレイヤーによるプレイログ、操作履歴、プレイ後のアンケートなどがレベル毎にシステムに蓄積される
4. 蓄積されたプレイログの統計処理を行い、デザイナーが統計やログの閲覧ができる
5. 蓄積されたデータを元に、問題のあると思われるレベルを検出する
6. レベル内の問題のあるカ所をシステムが指摘する
7. 指摘された問題点を修正したレベルを生成し、システムに再登録する

本稿では、上記のユースケースのうち、4 のログの集計までを目標とし、システムを構築した。

3.5　実装したシステム

今回、実験にあたり、以下のようなシステムを設計し、LeDA(Level Design Assistant)と命名した。

図 3.1: LeDA

LeDA は以下のような機能を有する

- デザイナーによるレベルの登録

- プレイヤーへのレベルデータの配信
- プレイヤーアンケート収集
- プレイヤー毎の各種統計処理
- レベル毎の各種統計処理
- 1 ゲームのプレイログの再生
- 複数のプレイログの同時再生

3.6 システムの設計

システムは Web アプリケーションとしてサーバー上に構築されており、以下のモデルが実装されている。

この項での解説においては、多くのゲームシステムを持ったゲームに対応するため、抽象的なモデル設計について議論するが、実際の実験に利用した具体的なゲームルール上に適応できるモデルは第 4 章において詳解する。

3.6.1 Level

Level はゲームにおけるレベル 1 つ 1 つを表すモデルである。

通常はシステム管理者がシステムに登録する。

Level は通常、マップデータなどがテキスト形式で保存されているが、データ構造はゲームによって異なる。

3.6.2 Player

Player は、あるプレイヤーを一意に特定するためのモデルである。

今回は、Player は IP アドレスのみを保持し、IP アドレスが同一かどうかで、同一の Player だと仮定する

同様の Player によるプレイログを辿ることにより、プレイヤーのプレイ回数による習熟度の変化を観測することができる。

3.6.3 Metric

Metric はプレイヤーのプレイログを表すモデルで、１回のゲームプレイにつき一つが生成される。

Metric はそれぞれ以下の情報を保持している

- プレイされた Level
- プレイした Player
- Operations のリスト
- 一つ前の Metric への参照
- Metric の最終状態
- 回答された Questionnaire

1 つ前の Metric は、この Metric が生成される前に行われたゲームプレイを表す Metric への参照が張られている。これをたどることで、このゲームプレイが初回プレイなのか、別の level をプレイ後にプレイされた物なのか、前回も同様の Level をプレイしていたにもかかわらず、クリアできずにリプレイしたゲームプレイなのか、といった情報を判別できる。

Metric の最終状態には、この Metric がどのような状態で終了したかが保存される。初期状態はプレイ中 (Playing) となっているが、クリアした場合は Clear、ゲームオーバーになってしまった場合は GameOver などが記録される。今回は簡単のため、Clear, GameOver、Playing の 3 つの状態のみ仮定しているが、プレイの状態をゲームの内容に応じてさらに細分化することもできる。

3.6.4 Operation

プレイヤーの行動１つ１つを示したモデル。例えばクリック位置や、ゲーム中で選択したコマンドなど、プレイヤーの行動が記録されている。

内容や生成頻度はゲームルールによって異なる。例えば、アクションゲームの場合、毎フレーム生成されることもあるが、ターンベースのゲームの場合、各ターン終了時に生成されることもある。

行動内容の他にタイムスタンプがミリ秒単位で記録され、システム管理者は Metric に関連づけられた Operation のリストを辿ることで、プレイヤーの入力を完全に再現できる。

3.6.5 Questionnaire

プレイヤーのアンケート結果を示したモデル。

システム管理者は任意のアンケートを発行し、各プレイの終了後、プレイヤーに回答させることができる。回答は対応する Metric に関連づけられる。

アンケートの内容はゲームシステム、管理者の運用によって異なる。

3.7 システムの動作の流れ

本項では、前項で説明したデータベース (以下、DB) 上のモデルをどのように扱うか、プレイヤーの操作の順を追って、細かな処理を解説する。

ここでは、プレイヤーがプレイを開始し、ゲームオーバー、クリア、または途中での離脱など、なんらかの理由でプレイを終了するまでの単位を「ゲーム」と定義する。

図 3.2 は処理の流れを示したフローチャートである。以下、この図に沿って解説を行う。

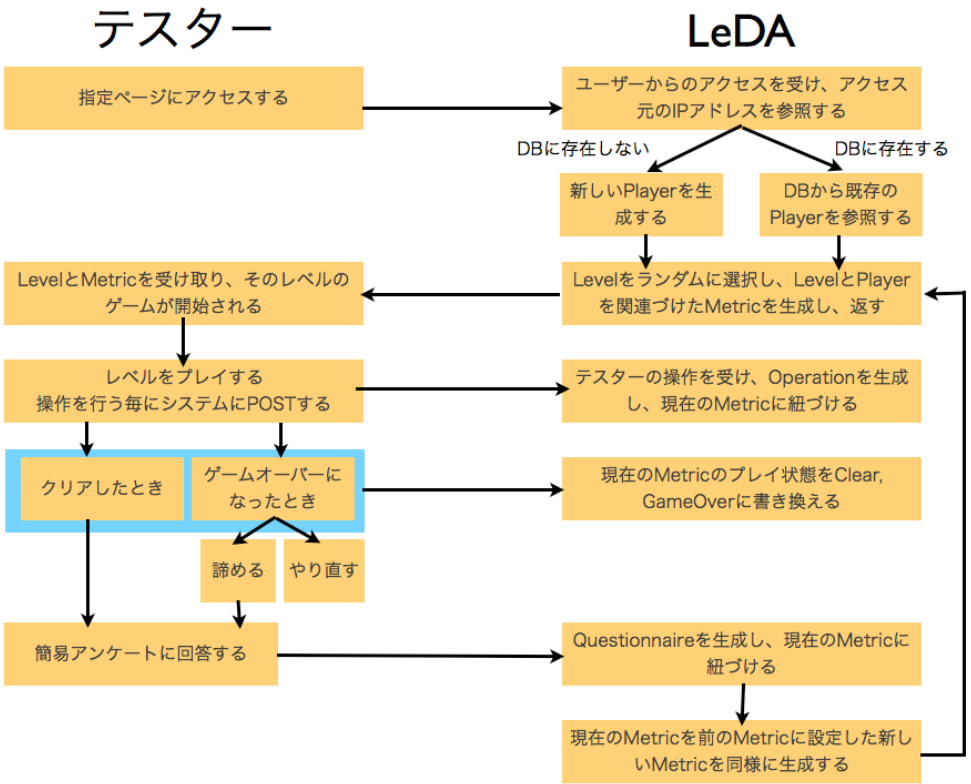

図 3.2: システムのフローチャート

3.7.1　1. アクセス後の処理

プレイヤーが Web ブラウザにアクセスすると、プレイヤーのリクエストをシステムが受け、アクセス元の IP アドレスを参照する。

IP アドレスを DB から検索し、もし存在していなければ、そのアクセスが新規プレイヤーだと仮定し、新たな Player を生成し、DB に格納する。もし、存在していれば、その Player を DB から取得する。

その後、システムに登録されている Lavel の中からランダムに 1 つを選択、Level と Player を関連づけた新しい Metric を生成する。

この Metric1 つ 1 つが 1 ゲーム分のログを表すデータである。

システムは生成された Metric の主キーと Level のデータをクライアントに返却する

3.7.2　2. ゲームの開始

サーバーから Level のデータを受け取ると、ブラウザ上ではそのデータを元に Level が再現され、即座にゲームが起動する。

プレイヤーはサーバーとの通信を意識することなくゲームをプレイすることができる。

3.7.3　3. プレイヤー操作の記録

プレイヤーのゲーム中での入力操作と現在の Metric の主キーが、操作を行う度にサーバーへ送信される。

サーバーはその情報を受け、受け取った入力操作を元に Operation を生成し、Metric に追加する。

この操作は、ゲーム終了まで何度も行われる。

3.7.4　4. ゲーム状態の変化

ゲーム中でプレイヤーがクリアやゲームオーバー状態に達したとき、その変化を現在の Metric の主キーと共にサーバーに送信する。

システムはその情報を受け、対応する Metric の状態を変更する。これによって、そのプレイの最終状態がどのような状態で終わったかをシステム管理者が容易に把握することができる。

3.7.5　5. ゲームの終了

プレイヤーがクリアする、または未クリアだがギブアップを選択したとき、プレイ後に何らかのアンケートが表示され、プレイヤーの声を収集することもできる。アンケートの結果はサーバーに送信され、Questionnaire が生成され、現在の Metric と関連づけられる。

未クリアかつ、リプレイを選択した場合、アンケートは表示されずに、1 のステップへ戻り、ランダムに選択された Level ではなく、現在と同様の Level が再び配信される。また、現在の Metric が次に生成される Metric に関連づけられる。

3.7.6　6. 次のゲームの開始

5 までのステップで、一連のプレイログの記録が終了し、ここまでのデータを一つの Metric として扱う。

プレイヤーが他のレベルに挑戦したい場合、再び 1 の手順とほぼ同様の処理が行われ、新たな Level の配信と、Metric の生成が行われる。

このとき、必ず現在遊んだ Level と異なった Level が配信されてくる。また、次に新しく生成される Metric の前の Metric として、現在の Metric が関連づけられる。

3.8　実装

実装は主にサーバーサイドとクライアントサイドに分かれて行われている。

3.8.1　サーバーサイド

サーバーサイドの実装は、Python 製の Web Application Framework である Django により実装されている。この Framework は OSS であり、簡単に利用できる。

この Framework により実装されたシステムを、Apache 上で Python を動作させるモジュール、mod_wsgi を用いてサーバー上で動作させている。

また、アプリケーション上で生成されたモデルを格納するリレーショナル・データベースは、取り扱いの容易さの観点から sqlite3 を採用している。

3.8.2　クライアントサイド

クライアントサイドでは、ゲームロジックが JavaScript で記述され、HTML5 の Canvas タグにより描画されている。

また JavaScript のソースコードは、メタ言語である CoffeeScript で記述されている。

その他、サーバーとクライアントの通信に、JavaScript 製ライブラリである jQuery を利用している。サーバーとのデータ通信は、サーバーのアプリケーション上にあるモデルを JSON というデータ記述言語としてダンプし、その文字列をやりとりすることにより行っている。

第4章　対象とするゲーム

4.1　はじめに

システムの有用性を検証するために、今回はこのシステムの利用に想定されうるサンプルゲームを実装した。

この章では、このゲームのルールや、採用理由について解説し、このゲーム固有のシステムの設計について述べる。

4.2　ゲームの概要

4.2.1　基本ルール

今回、対象としたゲームは、ターン型完全情報のパズルゲームである。このゲームは、10x10 の盤面を操作し、キャラクターがゴール状態に辿り着くまでに繰り返す。プレイヤーは一人でプレイする

4.2.2　ターンの流れ

まず、プレイヤーは盤面の中から隣接する 2x2 の任意の 4 マスについて選択する。

図 4.1: 初期状態

選択したマスについて、プレイヤーは選択したマスを右方向、ないしは左方向に回転させる。ただし、何も無い空白のマスを含む 4 マスは回転させることができない。

マップを回転させると、上に乗っているオブジェクトやキャラクターも同様のルールによって回転する。（図 4-2）

図 4.2: プレイヤー操作後の状態

次に、プレイヤーキャラクターは、現在向いている方向に自動的に移動する。移動はスキップすることができない。（図 4-3）

図 4.3: ターン終了時の状態

この一連の 2 つのステップを 1 ターンと定義する。プレイヤーはターンを終了状態に到達するまで繰り返し、これを 1 ゲームとする。

4.2.3　ゲームの終了条件

ターンを繰り返し、いずれかの状態になったとき、１ゲームが終了する。

クリア条件

以下の状態になったとき、そのゲームはクリア状態となる

- プレイヤーキャラクターがゴールマスに到達したとき

ゲームオーバー条件

以下の状態になったとき、そのゲームは失敗となり、ゲームオーバーになる

- プレイヤーキャラクターがマップ外にはみ出したとき
- 死亡判定のあるギミックに触れたとき（穴や針など）

4.2.4　マップ上のギミック

マップ上のギミックとして、下記の物が実装されている。

岩

キャラクターは進入することができず、進行方向逆向きに方向を転換し、1 マス進む。進むべき方向に別の岩がある場合、方向のみを切り替える。

図 4.4: 岩

穴

プレイヤーキャラクターが進入するとゲームオーバーとなる。プレイヤー以外のキャラクターが進入すると消滅する。

図 4.5: 穴

針

キャラクターが 4 方向のうち、針の向いている方向から進入すると、穴と同様でゲームオーバーになる。針のないその他の 3 方向から進入したとき、岩と同様の扱いとなる。

図 4.6: 針

滑る床

プレイヤーキャラクターが進入すると、自動的に向いている方向にもう 1 マス進む。進んだ後に進入した床がさら滑る床だった場合、再帰的に進み続ける。

図 4.7: 滑る床

崩れる床

1 度目の進入は通常の床として扱う。崩れる床から他のマスに移動したとき、崩れる床は穴に変化する。以降は穴として扱う。

図 4.8: 崩れる床

ジャンプ台

進入すると、向いている方向を 1 マス飛び越える。飛び越えたマスの効果は受けない。

方向パネル

進入すると、現在の矢印の方向にキャラクターの向きを変える。

4.3 対象ゲームの採用目的

このゲームを実験対象のゲームとして採用したのは、以下の理由が挙げられる。

操作が容易でプレイヤー間の実力差が出にくいため

ルールさえ習得できれば、複雑な操作が必要なく、マウスのみで遊べるため、非常にプレイしやすい。特殊な操作も必要のないため、先行研究で利用されていた『Infinite Mario』などに比べ、誰もが上達しやすいため。

確定完全情報ゲームであるため

ランダム性が一切ないため。ランダム要素を含むゲームだと、収集したログの解析が難しい物になってしまう。また、ある手を取ったとき、次の状態が確定するため、計算機上でのシミュレーションも容易であるため、ログを収集したデータの定量的評価が行いやすい。

リプレイ性が高いため

様々な手を試し、試行錯誤をはかるゲームであるため、リプレイ性が高いと予想できる。そのため、プレイ回数に応じたプレイヤー力量の変化を観測しやすい。

問題空間が小さく、解の探索が容易なため

プレイヤーの取れる行動が非常に少ないため、計算機上での全探索による解の探索、ソルバーの実装が容易である。

レベル作成が非常に困難を極めるため

レベルデータの作成は人の手によって行われているが、良質なレベルの作成が非常に困難である。そのため、収集したプレイヤーログをレベル作成に生かすことができれば、非常に有用な物となるため。レベル作成の困難性は後の項で詳しく解説する。

4.4 対象ゲームの解の探索

プレイヤーが取れる入力の総数 X は以下の数式で表される

$$X = (2((W-1)\cdot(H-1)))^T$$

ただし、W マップの横幅 H マップの高さ、T は総ターン数

しかし、実際は何も無いマスを含む回転は行えないため、回転カ所は非常に限定される。

以下執筆中

4.5 対象ゲームのレベル設計方法

この対象ゲームにおいて、良質なレベルデザインは非常に困難を極める。

この節では、このゲームにおけるレベルデザインの困難性について述べる。

4.6 対象ゲームに特化したシステム設計

対象ゲームにおいて、前章において述べたそれぞれのモデルは、以下のようなデータ構造で記述できる

4.6.1 Level

各「レベル」を定義する Level モデルは以下の情報を保持すれば良いと考えられる

map　マップ情報を定義する二次元配列を文字列として JSON 形式で格納している。ゲーム実行時にクライアント側でパースし、ステージとして読み込む

optimized turn　ソルバーにより導出された最短手数が記録される

4.6.2 Operation

プレイヤーの行動を定義する Operation モデルは以下の情報を保持すれば良いと考えられる

x　回転する 4 マスのうち、左上のマスの x 座標が記録される

y　回転する 4 マスのうち、左上のマスの y 座標が記録される

direction　回転方向が 0 または 1 で記録される

timestamp　Operation が発行された日時がミリ秒単位で記録される

このゲームにおいては、この Operation を保存しておくことで、プレイヤーの動きを完全に再現できる。

第5章　結論

5.1　結果

本研究では、クラウドソーシングを用いてプレイログの収集、ログの閲覧を目的としたシステムを開発した。ゲーム開発者の需要が概ね満たされ、開発者にとってはある程度有用なツールとなり得る。

5.2　課題

今回実装したシステムについて、現段階で考えられる問題点とその改善案について記した。

5.2.1　SPAM の判定とフィルタリング

今回開発したシステムを商用ゲーム開発の現場でも広く利用できるようにするためには、システムの信頼性を担保することが急務であると考えられる。

例えば、外部にアウトソーシングを行うに当たって、まともにプレイしたプレイヤーか、収益のために適当なプレイを行ったプレイヤーかを判別、分類する機構が必要であると考えられる。

具体的には、連続して同じ操作が繰り返されているかどうかを検出する、同じレベルが何度もプレイされているが、一向に生存手数が伸びない、平均プレイ時間が極端に短いなどと言った、プレイヤーの質のスコア化が必要となってくるであろう。

5.2.2　プレイヤー習熟度を考慮した統計処理

現段階のシステムでは、データ的には、個々のプレイヤーを個別に追跡することが可能になっているが、現在の統計処理は、個々のプレイヤー習熟度を一切考慮していないため、プレイヤー習熟度を考慮したシステム設計が必要だ。

まず、レベルの配信アルゴリズムについて、現在は完全なランダムで選択されているが、個々のプレイヤーの過去のプレイ状況から配信される問題に重み付けをする方法が考えられる。

最も単純な方法は、すでにクリア済みのレベルは配信されるレベルから除外するという手法であろう。

また、レベルの推定された難易度と比較することで、このレベルをクリアできるプレイヤーであれば、これぐらいの熟練度を持っているというようなヒューリスティックな習熟度の計測も可能であると考えられる。そのため、この推定結果から、徐々に配信されるレベル難易度を上げていくなどの手法も考えられる。

5.2.3 汎用的なシステムの模索

今回提案したシステムは、対象としたゲームに特有のものであり、多くのゲーム作品に汎用的に利用可能なシステムにする方法を模索していくことも必要であろう。

第 3 章でも、さまざまなゲームに汎用的に利用可能なシステムのデータ構造などについて示したが、

具体的には、データの保存、取得を行う API を提供し、ゲーム開発者がプレイログを収集したいタイミングで特定のコード片を埋め込むことで、汎用的にデータのやりとりを行わせる方法などが考えられる。

5.2.4 レベルの改善点検出とその修正

本研究の大目的であるレベルの改善点検出とその修正という目標は、未だ達成できていない。

このゲームに特化した手法であれば、レベル中の各マスについて、到達状況をヒートマップ化し、クリアに不要なマスを枝刈りするなどの方法が考えられるであろう。

第6章　謝辞

本研究を行うに当たり、幅広い見地から数多くの御意見、御提案を頂きました栗原正仁教授、小山聡准教授に深く感謝致します。また、研究内容に関する的確な御指導を頂きました佐藤晴彦助教に深く感謝致します。最後に、日々の研究活動において様々に御助言を頂いた表現系工学研究室の皆様に心より感謝致します。

参考文献

[1] University of Copengagen, Anders Drachen, Alessandro Canossa, Towards Gameplay Analysis via Gameplay Metrics

[2] Desurvire, Heather Wiberg, Charlotte, Game Usability Heuristics (PLAY) for Evaluating and Designing Better Games : The Next Iteration

[3] Infinite Mario Bros http://www.mojang.com/notch/mario/

[4] University of Copenhagen, Chris Pedersen, Julian Togelius, Georgios Yannakakis, Modeling player experiece in Super Mario Bros.

[5] University of Copenhagen, Chris Pedersen, Julian Togelius, Georgios Yannakakis, Optimization of Platform Game Level for Player Experience

[6] Shaker, Noor Togelius, Julian Yannakakis, Georgios N. Weber, Ben Shimizu, Tomoyuki Hashiyama, Tomonori Sorenson, Nathan Pasquier, Philippe Mawhorter, Peter Takahashi, Glen Smith, Gillian Baumgarten, Robin The 2010 Mario AI Championship: Level Generation Track

[7] Mario AI Championship http://www.marioai.org/

[8] はこだて未来大学 五木宏 松原仁 遺伝アルゴリズムの視覚化を用いたゲームのレベルデザイン効率化技法の開発

[9] Togelius, Julian Schmidhuber, Jurgen An experiment in automatic game design

[10] Georgia Institute of Technology Nelson, Mark J Mateas, Michael Towards Automated Game Design