落ちものパズルゲーム 共通ルール記述言語の提案

栗原 一浩 [1,a)]　阿部 雅樹 [2]　渡辺 大地 [2]

概要：ゲームの自動生成技術は、主にフィールドやダンジョンを対象とした生成を行っているが、ゲームのルール自体を自動生成した生成実例は少ない。GDL(Game Description Language)はゲームルールを体系化し、簡易的にゲームを形作るための記述言語である。 GDL を用いることで AI によるゲームルールの自動生成や、ゲームの面白み測定という研究も存在するが、対象はボードゲームに留まっている。 そこで、我々はデジタルゲームの中から落ちものパズルゲームに着目した。落ちものパズルゲームはアクション性が高くゲームルールを体系化でき、GDL として記述できる可能性がある。本研究では落ちものパズルゲームの自動生成を目的とし、本稿では落ちものパズルゲームを GDL として記述できるシステムを開発した。その結果、本提案システムを用いて落ちものパズルゲームを幅広く実装できることが分かった。

キーワード：落ちものパズルゲーム, デジタルゲーム, パズルゲーム, Game Description Language, 共通ルール記述言語

A proposal for game description language of Falling block puzzle games

KAZUHIRO KURIHARA [1,a)]　MASAKI ABE [2]　TAICHI WATANABE [2]

Abstract: Automatic game generation technology automatically generates fields and dungeons, however, there are few examples of generating game rules. GDL(Game Description Language) is a description language that is organized game rules for generating game easily. There are researches on AI uses GDL to automatically generate game rules and measure the fun of games, but the subject remains limited to board games. Therefore, we focused on Falling block puzzle games in digital games. The purpose of this research is to generate Falling block puzzle games automatically. In this paper, we developed a system that can describe Falling block puzzle games as GDL. As a result, We found that Falling block puzzle games can be widely implemented using the proposed system.

Keywords: Falling block puzzle games, Digital game, Puzzle Game, Game description language,

1. 背景

ゲームの自動生成の基盤技術としてゲームルール記述言語は重要であり、ゲームルール記述言語を用いたゲームの自動生成を行った研究も存在する。Cameron Browne ら[1]による研究では、実際に「Ludi」と呼ばれるゲームルール記述言語を用いて AI によるゲームルールの自動生成・そのゲームが人間プレイヤーにとって興味をひくものであるかの計測を行っている。ゲーム記述言語の研究は Nathaniel Love ら[2]により研究されている。しかし、これらの研究ではボードゲームの範疇に収まっており、デジタルゲームにおけるアクション性のあるゲームは実現されていない。ビデオゲ

1　東京工科大学大学院バイオ・情報メディア研究科
Graduate School of Bionics, Computer and Media Sciences, Tokyo University of Technology
2　東京工科大学メディア学部
School of Media Science, Tokyo University of Technology
a)　g31180086c@edu.teu.ac.jp

ームをゲームルール記述言語 Video Game Description Language(VGDL)で記述できるよう研究された事例もいくつか存在するが、アクション性のあるゲームが記述できるものは限定的である。そこで、本研究ではデジタルゲームにおけるアクション性のあるゲームとして「落ちものパズルゲーム」に着目した。落ちものパズルゲームを基本的なルールを編集し、作成できるツールもいくつか存在しており、「電撃コンストラクション　落ちゲーやろうぜ！」や「落ちゲーデザイナー作ってポン！」など存在するが、一般的な落ちものパズルゲームであるぷよぷよ[3]は再現できてもテトリス[4]を再現することは不可能であり、ゲームルール記述言語やツールを含めて、これまで落ちものパズルゲームを一つのシステムで多くのルールを実現した事例はない。落ちものパズルゲームをゲームルール記述言語として記述できるようにすることで、落ちものパズルゲームの自動生成を目標とし、本稿では落ちものパズルゲームのゲームルール記述言語の開発をし、実際にリリースされている一般的な落ちものパズルゲームが実現できることを検証した。

2. 落ちものパズルゲームの定義

落ちものパズルゲームとは役割を与えられたブロックが複数種類存在し、ブロックが複数個集まっているツモと呼ばれるものをプレイヤーが操り、ブロックをとある条件を成立させ消去・変化させ得点を稼いでいくというアクションパズルゲームのことである。また、落ちものパズルゲームから派生したアクションパズルゲームもいくつか存在するが、本研究では対象外とする。

本研究における落ちものパズルゲームとは、フィールドと呼ばれる格子状のプレイエリアにツモを操作してブロックを設置していき、ブロックを特定条件で消去するゲームのことを指す。ツモは時間経過、もしくはプレイヤーの操作により落下するものであり、ツモのブロックはフィールドの一番下まで落下しツモのブロックがどれか一つでもフィールドの一番下に衝突した際に設置、もしくはツモのブロックどれか一つでも設置されているブロックに対して下方向に衝突した際にプレイエリアに設置される。プレイヤーができるツモの操作は左右移動、落下、回転（ゲーム毎の法則にしたがう）が可能である。ブロックにはそれぞれ落下法則があり、ブロック一つ一つがつながりを持たず、下方向に空間のできないよう落下するものと、ブロックとブロック同士で繋がりがあり、繋がりをもったブロックが削除されない限りつながりを持ったブロックよりも落下することはないが、つながりを持ったブロックすべてが下方向に空間ができていた際に落下するもの、ある特定の条件下のみで落下するものがある。一般的な落ちものパズルゲームで例を挙げると、ぷよぷよやテトリスは対象となり、パズルボブルやパネルでポンなどは対象外となる。

3. 落ちものパズルゲームのルール体系化

落ちものパズルゲームを GDL として記述できるようにするためには、ゲームルールを体系化する必要がある。落ちものパズルゲームはプレイヤーが操作する「ツモ」、ピースごとの「消滅条件」、「落下法則」、「消滅方法」の大まかに 4 つに分類することができる。そこからまた更に細分化していくと、ツモには「形の分類」と「回転法則」を決めるとほとんどの落ちものゲームに用いられるツモは表現が可能である。落下法則は、大きく二つの分類がされ、「ちぎれる」ものと「ちぎれない」ものに分類できる。消滅条件は大きく 4 つに分類でき、「連結型」、「直列型」、「固定形型」、「起爆（特殊）型」がある。連結型は形はとはず、上下左右で同じ役割を持つブロック同士が繋がっており、一定以上で消滅判定が発生するものである。直列型は同じ役割のブロックが直線上に一定以上並んでいたら消滅判定が発生するものである。固定形型は同じ役割を持つブロック同士で特定の形を作る事で消滅判定が発生するものである。しかし、消滅条件においては、落ちものパズルゲームにおいて最も体系化が複雑化するものであり、これらの分類には縛られないこともあると考えられる。消滅方法は主に 3 つに分類することができ、消滅判定がでたら即消滅する「即時消滅」。消滅判定が出たら他のブロックに変化する、もしくは形態が進む「変化」。消滅判定が出ても留まり、さらに特定の条件を成立させることで消滅する「貯蓄」があり、これらでほとんどの落ちものパズルゲームを再現できると考えられる。

4. 提案ゲームルール記述言語仕様

4.1 初期設定系

初期設定系は、ゲームの初期設定として指定する機能である。

- フィールド（パズルのプレイエリアとなる格子状の領域）のサイズ指定
- ブロックの描画サイズ指定
- 削除条件テンプレートの使用モードの変更（無効化可能）
- 同色が一定以上繋がっていた際に削除判定を出すテンプレートモード指定（無効化可能）
- ブロックの種類の作成、パラメータの指定
- ツモの種類と回転法則の指定
- ツモ回転時のハイパーローテーションの使用有無
- ツモ回転時のクイックターンの使用有無
- ツモ回転時の壁蹴りの使用有無
- ツモ回転時の床蹴りの使用有無
- ハードドロップの使用有無
- ソフトドロップの使用有無
- ソフトドロップを使い着地した時の即設置するかどう

かの設定、また猶予時間
- 横移動速度（一度押してから押しっぱなし移動になるまでの待機時間、押しっぱなし移動時の速度）の指定
- ソフトドロップ速度の指定
- ブロック描画時のマス間描画補完の有無

4.2 取得・制御系

取得・制御系は、主にゲームの状況取得や制御を行う機能である。
- フィールドのサイズ取得
- フィールド上の指定座標からブロックの状態を取得
- 指定座標ブロックへの削除命令
- 指定座標ブロックへの削除待機時間指定
- 指定座標ブロックへの落下可能フラグの指定
- 指定座標ブロックへの強制落下段数指定
- フィールド上の指定座標へブロックを強制配置
- 同色ブロックがどれだけつながっているかの個数取得（ぷよぷよ系、コラムス系）
- 同色のブロックが繋がっている一つのグループに対して一気に削除命令を送る

5. サンプル

以下のサンプルは、テトリスの削除判定をしたものである。テトリスの削除条件である横一列にブロックが存在した場合の条件は以下のように記述する。

```
--横列にブロックが何個いるか捜査
for y = 0 , getFieldHeigt() do
    for x = 0 , getFieldWidth() do
        if getFieldBlock( x , y ) > 0 and
        getFieldBlock( x , y ) < 100 then
            checknum = checknum + 1
        end
    end
end
```

横一列にブロックが存在した際に、削除命令を送る場合は以下のように記述する。

```
--消去処理をする
for y = 0 , getFieldWidth() do
    if checknum >= 10 then
        checknum = 10
        downnum = downnum + 1
        for x = 0 , getFieldWidth() do
            setBlockDeleteFlag( x , y )
        end
    end
end
```

テトリスのような複雑なツモの形状をしたゲームでも対応できるようにするため、ツモの種類や回転法則を自身で設定することもできる。以下のサンプルコードは実際にテトリスのツモ「T」の形を設定したものである。

```
--配列準備
arrays = {}
for p = 1 , 4 do
      arrays[p] = {}
end

--ツモの設定
arrays[0] = { 0 , 0 , 0 , 0 }
arrays[1] = { 0 , 1 , 0 , 0 }
arrays[2] = { 1 , 1 , 1 , 0 }
arrays[3] = { 0 , 0 , 0 , 0 }

--ツモ0種類目の0回転目に登録
setTumo( arrays , 0 , 0 )

arrays[0] = { 0 , 0 , 0 , 0 }
arrays[1] = { 0 , 1 , 0 , 0 }
arrays[2] = { 0 , 1 , 1 , 0 }
arrays[3] = { 0 , 1 , 0 , 0 }

--ツモ0種類目の1回転目に登録
setTumo( arrays , 0 , 1 )

arrays[0] = { 0 , 0 , 0 , 0 }
arrays[1] = { 0 , 0 , 0 , 0 }
arrays[2] = { 1 , 1 , 1 , 0 }
arrays[3] = { 0 , 1 , 0 , 0 }

--ツモ0種類目の2回転目に登録
setTumo( arrays , 0 , 2 )

arrays[0] = { 0 , 0 , 0 , 0 }
arrays[1] = { 0 , 1 , 0 , 0 }
arrays[2] = { 1 , 1 , 0 , 0 }
arrays[3] = { 0 , 1 , 0 , 0 }

--ツモ0種類目の3回転目に登録
setTumo( arrays , 0 , 3 )
```

その他の形も同様に実装でき、テトリスのツモの形状をすべて実現できる。

6. 検証

実際に提案ゲームルール記述言語を用いて一般的な落ちものパズルゲームの代表として、ぷよぷよ、コラムス、テトリスを実装した。 図 1 はぷよぷよを実際に実装し、プレイした画面のスクリーンショットである。

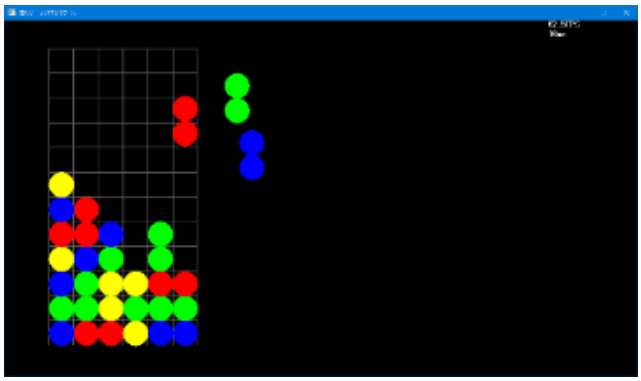

図 1 ぷよぷよを実装した画面

図 2 はコラムスを実装した画面のスクリーンショットである。

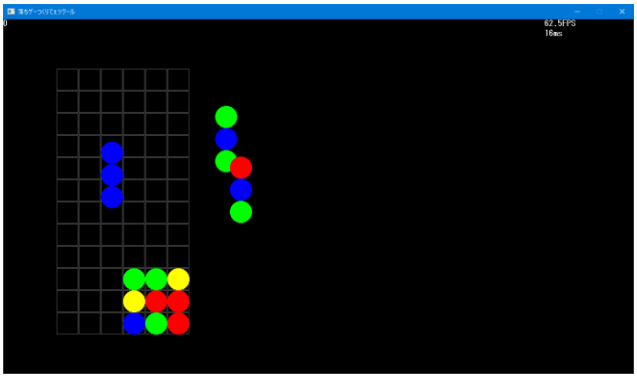

図 2 コラムスを実装した画面

図 3 はテトリスを実装した画面のスクリーンショットである。

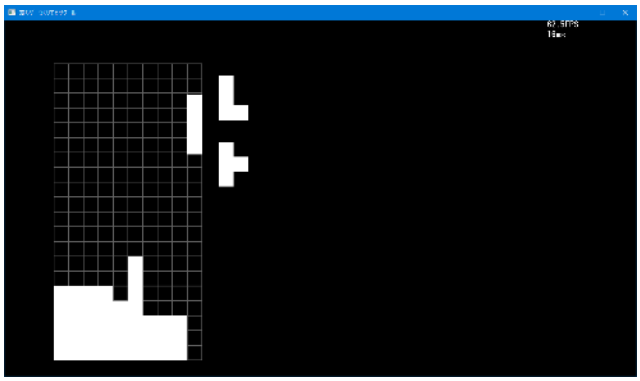

図 3 テトリスを実装した画面

検証の結果、ぷよぷよ、コラムス、テトリスのルールを提案ゲームルール記述言語を用いて実装することができた。テトリスにおけるツモの HOLD 機能などは考慮していない。

7. まとめ

本研究では落ちものパズルゲームの自動生成を目的とし、本稿では落ちものパズルゲームの共通ルール記述言語を提案、開発した。提案落ちものパズルゲーム共通ルール記述言語はぷよぷよ、テトリスなどの一般的な落ちものパズルゲームの実装が可能なシステムであることが検証できた。
今後の展望として、落ちものパズルゲームの自動生成や、このシステムを用いてゲームの面白さの計測、AI の開発支援として用いることができるシステムとして実装・検証していく。

8. 参考文献

[1] C. Browne and F. Maire: Evolutionary Game Design, *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 1-16 (2010).

[2] Nathaniel Love et al.: General Game Playing: Game Description Language Specification, *Stanford University, Tech.Rep.,* LG-2006-11(2006).

[3] “ぷよぷよポータルサイト”. http://puyo.sega.jp/portal/index.html , (参照 2018-12-18).

[4] “The addictive puzzle game that started it all!”. https://tetris.com/ , (参照 2019-7-12)