# 対戦型 2048

寺田 実[1,a)]

**概要**：2048 という一人遊び用のパズルゲームが存在する．それを対戦型としてプレーヤプログラムを作成するテーマが, 2015 年 1 月の第 56 回プログラミングシンポジウムにおいて GPCC 分科会 (Games and Puzzles Competitions on Computers) の一課題に選ばれた．今回の夏のシンポジウムで募集された「お題」のひとつとしてそれを提案したところ採用となったので, 何人かの参加者にそのプレーヤプログラムを作成していただき対戦を行った．本稿ではそれを報告する．

**キーワード**：2048, GPCC

## 1. 2048 について

2048 は Gabriele Cirulli が作成した一人遊び用のパズルゲームである．作者のウェブサイト [1] によれば Veewo Studio による 1024, Asher Vollmer による Threes というゲームに基づくものとされている．

### 1.1 ゲームのルール

4 × 4 の盤面上に 2 から始まる 2 のべき乗の値が記入されたタイルが置かれている (図 1)．初期盤面では, ランダムな位置に 2 または 4 のタイルが合計 2 枚置かれている．ゲームの目的は, タイルを移動させてタイル同士を合体させることにより, 大きな値をもつタイルを作成していくことにある．


**図 1** 2048 の外観 ([1] のサイトでのプレイ画面; 他の図も)

#### 1.1.1 タイルの移動

プレーヤが上下左右のいずれかの方向を指定すると, すべての動けるタイルが「盤面をその方向に傾けたように」[*1] 移動する．そのあと, ランダムな空き位置に新規タイルが一枚出現する．

#### 1.1.2 タイルの合体

タイルの移動の結果として, 同じ値を持つタイルが移動方向に隣接した場合, それら 2 枚のタイルは

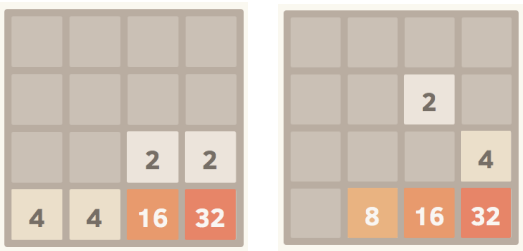


**図 2** タイルの合体 (左: 移動前, 右: 右方向への移動後)

[1] 電気通信大学 情報・通信工学科
[a)] terada.minoru@uec.ac.jp

[*1] 和田英一先生による表現

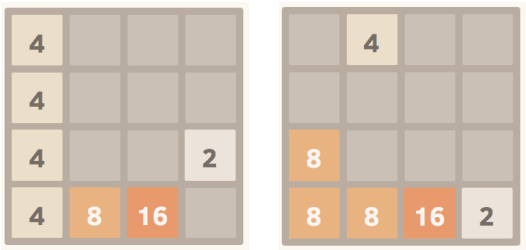


**図 3** タイルの合体は一度だけ (左: 移動前, 右: 下方向へ移動後)

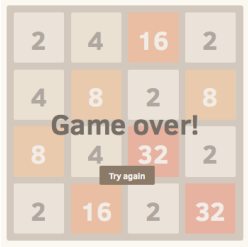


**図 4** ゲーム終了時の盤面

合体し, 2 倍の値を持つタイル 1 枚に変化する. 図 2 がその例で, 左の状況でタイルを右方向に動かすと, 隣接している (4, 4) (2, 2) がそれぞれ合体して右図のようになる. 右図の 2 が新規に出現したタイルである.

一回の移動操作ではタイルは一度しか合体しない. 図 3 がその例で, 左の状況でタイルを下方向に移動した場合, (4, 4) と (4, 4) がそれぞれ合体して 8 と 8 になるが, その 8 どうしは合体しない.

プレーヤはタイルの合体の際に合体によってできたタイルの値をスコアとして受け取る.

#### 1.1.3 ゲームの終了

ゲームは, タイルの移動操作とそれに続く新規タイルの出現を繰り返す. タイルの移動ができなくなった時点でゲーム終了である. このとき, 盤面上には空白がなくなり, さらに合体できるタイルもなくなっている (図 4). この時のスコアがプレイヤーの得点となる.

## 2. 対戦型のアイデア

### 2.1 守備側と攻撃側

オリジナルの 2048 では, 新規に生成されるタイルの位置と値 (2 または 4) はランダムに選ばれた. この新規タイルの位置を相手プレーヤが選択できるようにする, というのが対戦型のアイデアである. 対戦型 2048 においては, 従来のプレーヤ (タイルを移動する側) を守備側とし, 新規タイルの出現位置を指定する側を攻撃側とする.

盤面は空白の状態からスタートし, 攻撃側がタイルを置くことでゲームを開始する. これ以降ふたりのプレーヤが交互に手番をおこない, 守備側がゆきづまった (オリジナル 2048 と同様, タイルの移動ができなくなった) 時点で試合を終了する. 守備側はそこまでの得点を得る.

これを攻守交替して 2 回行い, 得た得点の多いプレーヤを勝ちとする.

### 2.2 新規タイルの値

攻撃側が新規タイルを置くときに, タイルの値をいくつとするかについて, ルール上の判断が必要である. オリジナルでは, 新規タイルの値は 90% の確率で 2, 10% で 4 がランダムに選ばれる.

これについて, 当初の考えでは, プログラミングを容易にする観点から, 新規タイルは 2 に限定することにしていた.

しかし, この制限は守備側を著しく有利にし, 試合を長引かせる原因にもなりうる. そこで, シンポジウム会場では, 新規タイルとして 2 と 4 のいずれかを攻撃側が自由にえらべるルール拡張も実験的に実装してみた.

### 2.3 ゲームプログラムの実装

対戦を管理するサーバは Java で実装し, プレーヤプログラムも Java のクラスとして実現することとした. 対戦はサーバプログラムがプレーヤプログラムのクラスをロードして自分のメモリ空間で対戦させる. プレーヤクラスのインスタンスはサーバオブジェクトからメソッド呼出を受けて動作する. 基本的なメソッドは以下の通りである.

`public int offend(int[] vec, int score)`
: 攻撃側の手番に呼び出されるメソッドで, 引数として現在の盤面 (16 要素の整数配列) と現在の守備側の得点を受け取る. 攻撃側が置く新規タイルの位置は, 盤面

**表 1** 対戦結果
プログラムのあとの括弧内は, 攻撃側が置ける新規タイルの種類に関するものである (2.2 参照).

| | プログラム | スコア |
|---|---|---|
| 第一試合 | 三廻部さん (2 限定) | 2688 |
| | 三好さん (2 限定) | 16760 |
| 第二試合 | 寺田 (2,4 任意) | 1040 |
| | 八木原さん (2,4 任意) | 1256 |
| 第三試合 | 三好さん (2 限定) | 504 |
| | 山口さん (2,4 任意) | 12640 |
| 決勝 | 八木原さん (2,4 任意) | 1652 |
| | 山口さん (2,4 任意) | 2776 |

配列の添字である 0 から 15 までの整数で返す. 新規タイルとして 4 も置けるルールの際には, 16 から 31 までを返させることとした.

`public Direction`
`defend(int[] vec, int score)`

守備側の手番に呼び出されるメソッドで, 引数として現在の盤面と現在の守備側の得点を受け取る.
盤面の移動方向を `enum Direction` のよっつの定数 (`Direction.RIGHT` など) のいずれかで返す.

`public static Integer`
`move(int[] board, Direction d)`

ユーティリティクラスのメソッドで, 指定した盤面でタイルを指定方向に移動させる。引数で与えた盤面の配列が変更を受ける。
返り値は, その移動によって得られた得点 (`Integer`) またはその方向に移動不可だった場合には `null` である.

また, サンプルプレーヤとして, ランダムな操作を行うプレーヤ, 単純な Minimax 戦略をもちいてプレイするプレーヤも用意した.

これに加えて, TCP で通信するスタブプレーヤも Java で作成し, 簡単な通信プロトコルも定義した. これを利用することで, 任意のプログラム言語で書かれたプレーヤも通信によってゲームに参加できる.

## 3. シンポジウムでの対戦結果

作成したプレーヤプログラムの対戦をシンポジウム会場で二日目の夜に行った. 対戦には 5 名が参加し, トーナメント方式で実施した. 前節で述べたとおり, 攻撃側が新規に置けるタイルの値について 2 のみに限定するルールと 2, 4 を任意に選択できるルールが併存している. 今回の対戦でも 5 名中 2 名が「2 限定ルール」, 3 名が「2, 4 任意ルール」でのプレーヤを作成した.

対戦結果を表 1 に示す. 山口さん作成のプログラムが優勝となった.

## 4. いただいたご意見など

シンポジウム会場でいただいたご意見には以下のようなものがあった.

- プログラム構成に関して
  - 標準入出力によるプレーヤプログラムとのインタフェースもあるとよい
- 初期盤面を空白で開始することについて
  - 最初の 2 枚のタイルはランダムに置くのはどうか
  - 最初の 2 枚のタイルを攻撃側が続けて置くのはどうか
- 4 のタイルを無制限に置けるのは攻撃側を有利にしすぎる
  - 置くことのできる 4 の枚数に上限を設ける
  - 2 を一定枚数置くと 4 のタイルが手に入る
  - 盤面上のタイルの最大値を守備側が更新したら 4 が一枚手に入る

## 5. 今後に向けて

対戦型 2048 を行うについて, いくつか課題となる点がある. 以下でそれを検討する.

### 5.1 ゲームとして成立するか

当初から懸念されたこととして, プレーヤプログラムどうしの相性のみで勝敗が決まってしまい, 一般的な強さのようなものがない可能性がありえた. 厳密なことはいえないが, 今回行った試合の結果を見ると, 強いプログラムはそれなりに強いと

言えよう.

### 5.2 ゲームのルールの検討

攻撃側が新規に置けるタイルを「2 限定」とするか「2,4 任意」とするかは大きな問題である. 試合の結果を見ると, 「2,4 任意」は「2 限定」にくらべて明らかに強く, 得点が低く抑えられる (攻撃側が有利で, 試合時間も短い) ことがわかる. したがって「2,4 任意」のほうが望ましいのではないかと思う.

### 5.3 試合の実施法

今回はプレーヤのクラスファイルを作成者がファイル共有サイトにアップロードし, 試合を実施する PC にそれらを人手でダウンロードして実行した. (なお, 山口さんのプレーヤプログラムだけは TCP で通信する仕様であるので, ご自身の PC 上で実行した) 参加プレーヤが多数となった場合には, この手順を自動化するなどの仕組みが必要になるであろう.

試合の形態としてトーナメント方式を採用した. これは試合経過を画面上で「実況中継」するために試合数を抑えたかったためであるが, プログラムどうしの相性に起因する勝敗のバイアスを減らすためにはリーグ戦も必要かもしれない. 実況中継を行わなければある程度の数のリーグ戦は可能であろう. 多数の参加プレーヤがある場合には, 予選リーグとその通過者によるトーナメントを併用するのも良いかもしれない.

### 5.4 試合の視覚化

試合の様子は盤面をプロジェクタで映すことで実況中継を行った. ただその表示速度の設定に問題があった. 序盤の空白の多い状態での淡々としたゲーム展開と, 終盤に近づいて緊迫した状況とでは望ましい表示速度が異なるのである. 盤面の緊張度をなんらかの方法で計算し, それに基づいた表示速度の自動調整や自動的なリプレイなどが行えると面白いであろう.

## 6. まとめ

シンポジウムのお題のひとつとして実施した対戦型 2048 についてその概略を示した.

**謝辞** プレーヤプログラムの作成に参加いただいたかたがたに感謝します.

### 参考文献

[1] Gabriele Cirulli, 2048, http://gabrielecirulli.github.io/2048/.