情報環境実験II (香川担当分)
Dronの改良

難民チーム
s12t252 畠山 侑也
s12t253 花川 直己
s12t270 森垣 航太
s12t272 山形 悠人

目次

1. 改良したゲームの概要
2. Dronのクラス図
3. 実装した機能の紹介
4. ゲーム画面の紹介
5. やりたかったこと
6. 担当紹介

1.1 元のDron

- 2人用
- どちらが長く棒を伸ばせるか競う
- 壁に衝突すると負け
- 棒に衝突すると負け

1.2 改良したDron

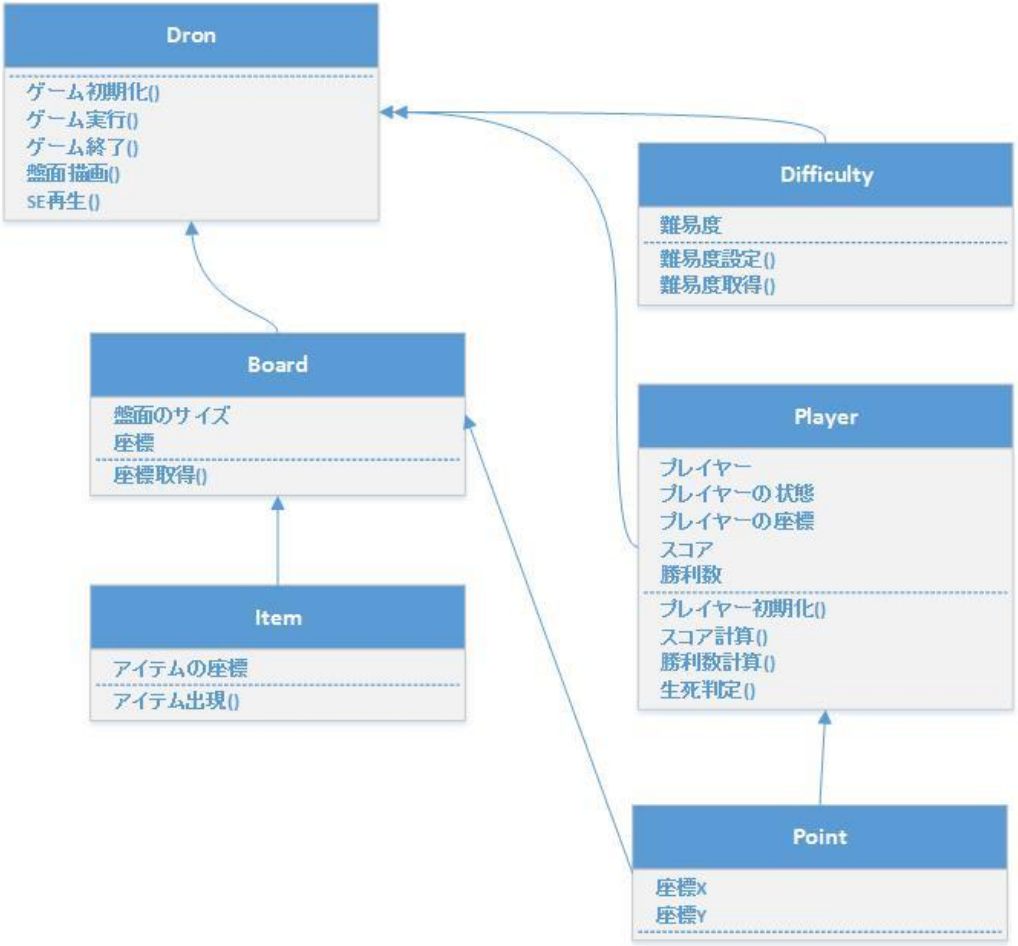
ゲーム内容

- 2人用
- スコアを競う
- 壁に衝突すると負け
- 相手の棒に衝突すると減点

ビジュアル,機能

- SEを追加
- 難易度を追加
- アイテムの追加
- ユーザビリティの向上

2.1 Dronのクラス図



3.1 Itemについて

乱数を発生させ、その値がアイテムの出現確率を超えているならアイテムを出現させる位置を返す。

```
27  public void getOfItemPos (Color[][] state) {
28    randTmp = Math.random();
29    itemPos.x = 0;
30    itemPos.y = 0;
31    flag = false;
32    cItem++;
33
34    if ( randTmp <= difficultyOnus && nItem <= 30 && cItem % iDifficulty == 0 ) {
35      while ( flag == false ) {
36        itemPos.x = (int)( Math.random() * xMax + 1 );
37        itemPos.y = (int)(Math.random() * yMax + 1 );
38        if ( state[itemPos.y][itemPos.x] == Color.WHITE ) {
39          nItem = nItem + 1;
40          flag = true;
41        }
42      }
43    }
44  }
```

3.1 Itemについて

乱数を発生させ、アイテムを出現させる位置に出現するアイテムの種類を乱数で分ける。

```
46  public Color getOfItemType () { // 出現するアイテムの種類の取得
47    randTmp = Math.random();
48
49    if ( randTmp >= 0.4 ) {  return Color.GREEN; }
50    if ( randTmp >= 0.2 && iDifficulty == 3 ) {  return Color.GRAY; }
51    return Color.ORANGE;
52  }
```

3.2 ユーザビリティについて

- スペースキーでゲームを開始するよう変更
- 勝利数の表示
- ゲーム領域の拡張
- SEの追加
- 音量のON/OFF機能を追加

3.3 その他について

- ゲームに制限時間を設定
- 難易度を設定

プレイしながらお聞き下さい

http://naosuke.me/Dron/dron.html

4.1 ゲーム画面

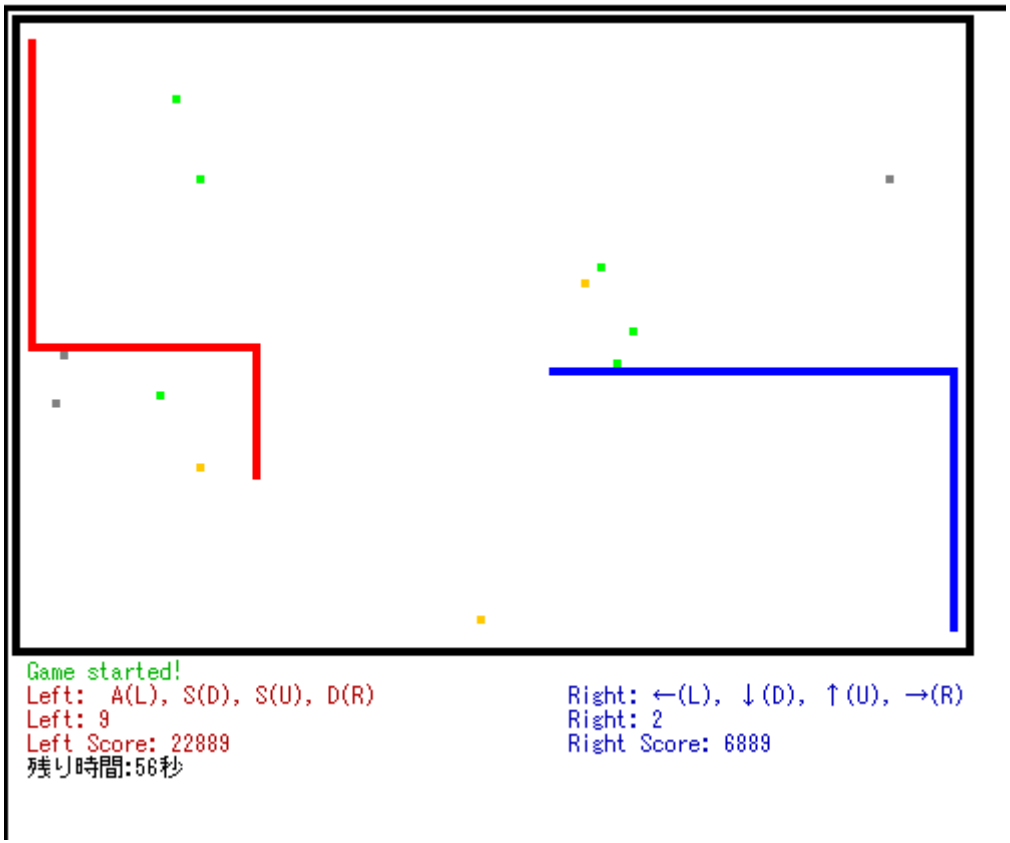
難易度を選択してください

easy:1 normal:2 hard:3

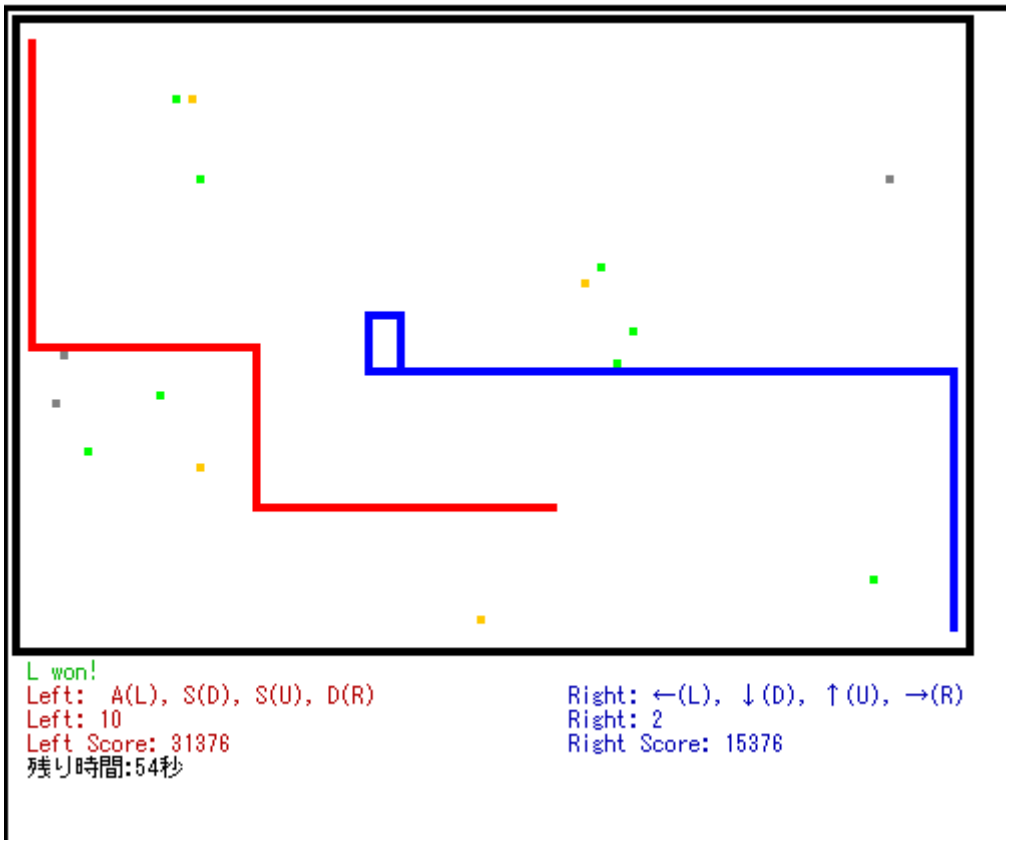
4.2 ゲーム画面


スペースキーでゲームスタート
Left: A(L), S(D), S(U), D(R)
Left: 0
Left Score: 0
残り時間:60秒
Right: ←(L), ↓(D), ↑(U), →(R)
Right: 0
Right Score: 0

4.3 ゲーム画面


Game started!
Left: A(L), S(D), S(U), D(R)
Left: 9
Left Score: 22889
残り時間:56秒
Right: ←(L), ↓(D), ↑(U), →(R)
Right: 2
Right Score: 6889

4.4 ゲーム画面


L won!
Left: A(L), S(D), S(U), D(R)
Left: 10
Left Score: 31376
残り時間:54秒
Right: ←(L), ↓(D), ↑(U), →(R)
Right: 2
Right Score: 15376

4.5 ゲーム画面

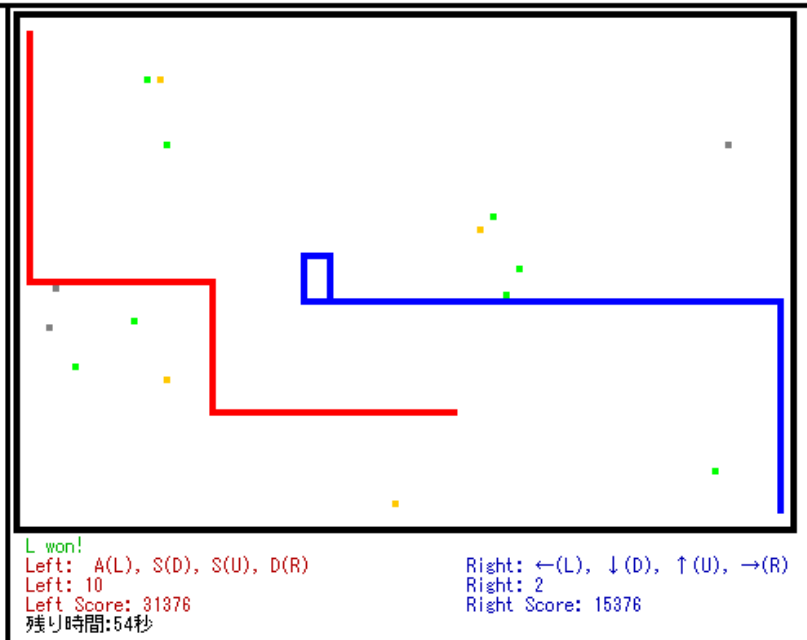
情報環境実験2 難民チーム

◉ルール説明
キーボードを操作して移動します。
黒い壁にぶつかると、ゲームが終了します。
相手のバーにぶつかると、スコアが減点されます。
自分のバーにぶつかると、その時点で負けです。

制限時間内に、相手より多くアイテムを集めましょう。

◉操作方法
1P(赤)
Wキー:上移動
Aキー:左移動
Sキー:下移動
Dキー:右移動

2P(青)
↑キー:上移動
←キー:左移動
↓キー:下移動
→キー:右移動



5.1 やりたかったこと

- Barクラスを実装して、バーの長さの制限
- そのために、バーの移動履歴を保持する、Queueクラスの実装

5.2 やりたかったこと

キューを実装して、移動履歴を格納

```
1  public class Queue {
2    final int SIZE = 50;
3    private Point[] values = new Point[SIZE+1];
4    private Point t = new Point();
5    private int head;
6    private int tail;
7
8    Queue() {
9      for ( int i = 0; i < SIZE+1; i++ ) {
10       values[i] = new Point();
11       values[i].x = 0;
12       values[i].y = 0;
13     }
14     head = tail = 0;
15   }
```

5.2 やりたかったこと

バーの長さが一定値を超えたら、キューを参照して最古の位置から削除していく

```
191             if ( countMove > barSize ) {
192               bMoveP1 = barP1.queue.dequeue();
193               state[bMoveP1.y][bMoveP1.x] = Color.GREEN;
194             }
195             barP1.queue.enqueue(currentPoint1);
196           }
```

6.1 担当紹介

12t252 畠山侑也
担当 : Barクラスの実装(失敗)、資料作成
12t253 花川直己
担当 : 音声追加、リファクタリング
12t270 森垣航太
担当 : ユーザビリティの向上
12t272 山形悠人
担当 : Itemクラスの実装

ソースコード公開中

http://github.com/hanasuke/Dron