

玉当てゲーム説明書

～ Minedraft ～

2012年 6月 30日

工学部 助教 吉見 真聡

TA 知的システムデザイン研究室 M2 宮地 正大

目次

はじめに	2
玉当てゲーム	2
概要	2
ルール	2
フィールド	3
行動	4
風について	6
プレイヤー / フィールド情報の取得	10
プロジェクト構成	12
AI 作り方	13
ゲーム内制御変数	14
公式戦	16
よきAI作成への道	17
FAQ ～よくある質問～	18
あとがき	19

はじめに

本プログラムはJavaプログラミング1 吉見クラスの講義課題として作成されたプロジェクトです。

本プロジェクトでは講義で学んだことを生かして、対戦ゲームのAI（人工知能）を作成します。単純に与えられたことをこなす課題ではなく、制約条件の中でどうすればAIが強くなるかを考え、それを自ら実装することでプログラミング力だけでなく自由な発想を養って欲しいと考えています。講義が進むに連れて、ゲームの制約条件を増やしていき、より高度な環境でAIの思考パターンを作成してもらう予定です。

玉当てゲーム

概要

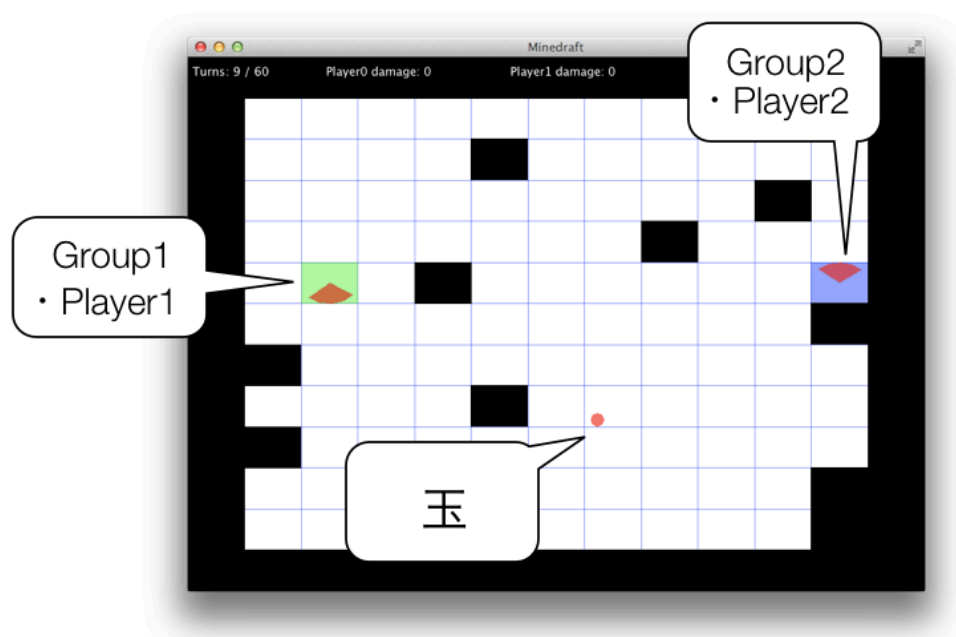
障害物のあるフィールド上で、キャラクター同士が玉を投げ合い、当てた玉の数を競う単純なゲームです。フィールドや障害物などのAI部分以外の実装はTAが行っています。そのため、皆さんにはAIファイルだけを変更することでプレイヤーの振る舞いをプログラミングしてもらいます。その際に必要なフィールドなどのデータを取得するAPIは用意してあります（詳細は後述）。

ルール

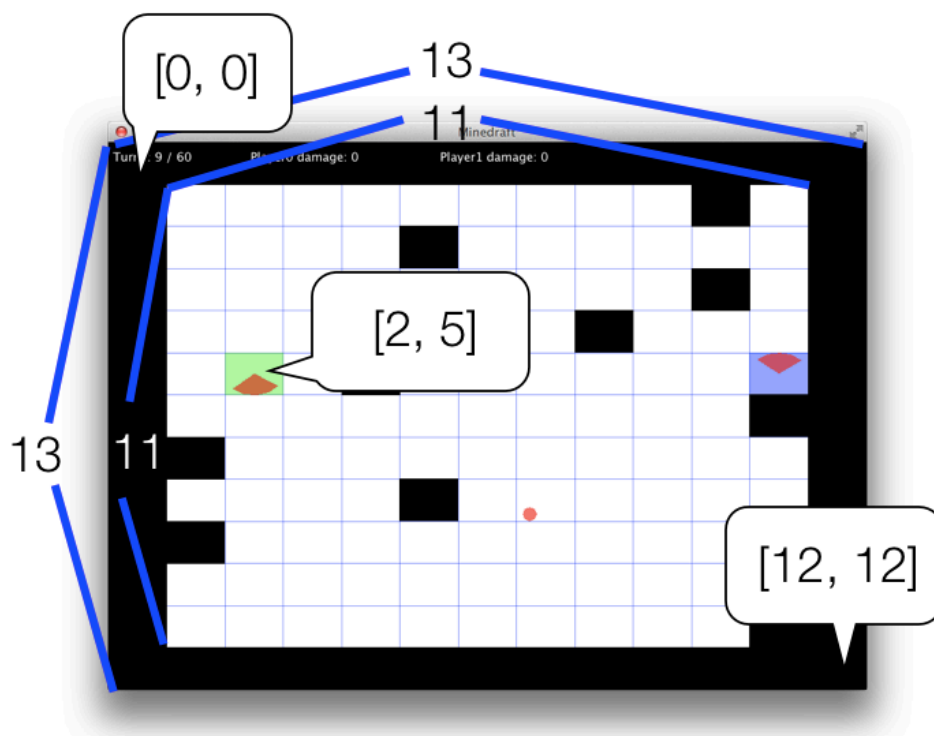
- ・ 長方形のマス上のフィールドに障害物・プレイヤーを配置
- ・ 外周には必ず壁（障害物）が配置、一列に一つランダムで障害物が配置される
- ・ すべてのプレイヤーが交代で行動する
- ・ 一回の行動でプレイヤーは 移動・方向転換・玉を投げる 動作をすることができる
- ・ 一回の行動中で玉を投げる動作は一回まで、その他は好きに使える
- ・ 玉を投げると何かに当たるまで玉は飛び続ける（自然消滅しない）。自分には当たらないが、味方（同じグループに所属しているプレイヤー）には当たる
- ・ 一人のプレイヤーが行動を終了して、次のプレイヤーに行動権利が移るまでを1ターンとする（行動中に玉を投げた場合、玉が着弾するまで次のターンに移らない）
- ・ 終了条件は指定したターン数を経過したとき
- ・ 玉を他プレイヤーに当てた場合、加点。逆に被弾した場合、減点
- ・ プレイヤーはグループで分けられており、グループ間での総得点で勝敗を決める

フィールド

可変長の二次元の盤（フィールド）上に壁（障害物）・プレーヤーが定義してあります。



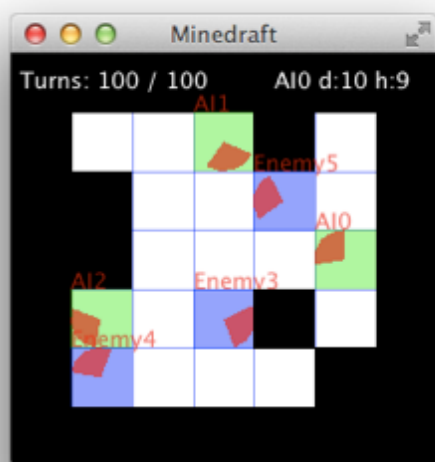
プレーヤーは何れかのグループに所属しており、グラフィック上では同一グループのプレーヤーは同じ色として描画されます。玉は赤色の○として描かれています。プレーヤー上の扇形は現在のプレーヤーの向いている方向を表しています。



フィールドは左上が原点となっており，正方形のマスで構成されています。フィールドは二次元の int型配列で構成されており，マスが空白（Piece.EMPTY：-1）壁/障害物

(Piece.WALL : -2) プレーヤー (プレーヤーの所属するグループID : 0~) で表現されています。 プレーヤーIDとは異なるので注意してください。 直接プレーヤーIDを用いずにこのような実装になっているのは、今後敵のプレーヤーIDを取得不能にして敵味方の判別だけの情報量に制限する予定があったためです。 そのため、 Piece.ME および Piece.ENEMY は非推奨の定義となりました。

```
5 public abstract class Piece {
6     public static final int EMPTY = -1;
7     public static final int WALL = -2;
```



1	-2	-2	-2	-2	-2	-2	-2
2	-2	-1	-1	0	-2	-1	-2
3	-2	-2	-1	-1	1	-1	-2
4	-2	-2	-1	-1	-1	0	-2
5	-2	0	-1	-1	-2	-1	-2
6	-2	1	-1	-1	1	-2	-2
7	-2	-2	-2	-2	-2	-2	-2

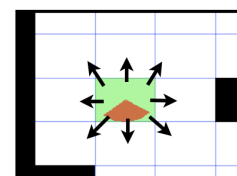
上記の図の例では、全てのマスと数字が対応しています。 グループIDが0の緑色のチームに所属するプレーヤーが存在するマスには0が、グループIDが1の青色の場合は1が入力されています。 空マス / 壁かどうかの判定式には Piece.EMPTY / Piece.WALLを用いてください。 番号の割り振りが変化する可能性があるので、数値決め打ちのハードコードしないでください。

行動

・移動

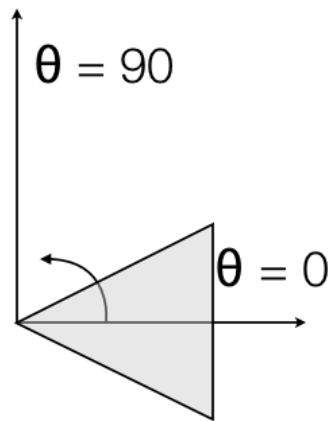
現在位置から周囲8マスのうち、空マスの場所に移動可能です。

移動用関数 move は引数にPoint型をとっており、移動可能なPointを与えるとプレーヤー位置が移動します。



・方向転換

現在のプレーヤーの向いている向きを変更します。 角度の扱いは極座標系で右が原点で反時計回りに回転します。 角度変更用の関数 angle には int型で引数を与えた場合には度数(0 ~ 360)で扱い、 float型で与えた場合にはラジアン(0 ~ 2π)として扱います。



例えば, `angle(45)` と `angle((float)(Math.PI / 4f))` は同じ挙動をします.

ここで一つ注意する必要があるのが, ゲーム中で定義しているフィールドは左上原点なので, 感覚的には上下反転した形でグラフィックが描画されます. つまり, `angle(45)` の状態で玉を投げても, 右上には飛ばず, フィールド上での扱いと同じく上方向が下に向かって表現されるので, 右下に向かって飛びます.

• 玉を投げる

指定した方向 or 現在向いている方向に玉を投げます.

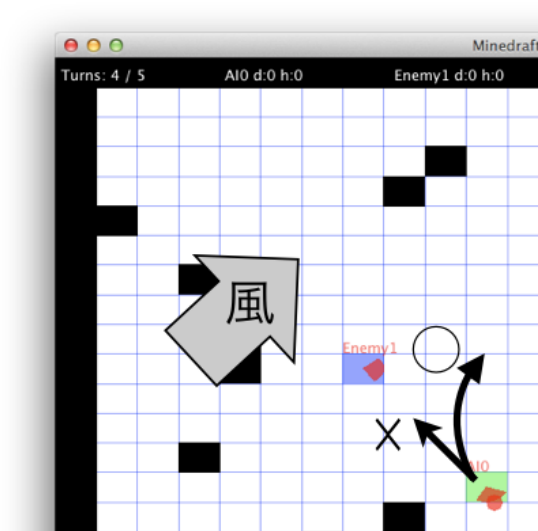
角度の扱いは上記「方向転換」と同様です. 投げる関数 `throwing` は `angle` と同じく, `int`型で引数を与えた場合には度数で扱い, `float`型で与えた場合にはラジアンとして扱います. 引数を与えない場合には, そのままの向きに玉を投げます. 引数を与えた場合には与えられた向きで `angle`関数を呼び出して向きを変えた上で玉を投げます.

その際の当たり判定方法は, `t 秒 * Piece.DYNAMICS` を加算した値で得られる玉の位置と, 物体の位置が同一マス上にある場合は当たりとします. なので, 直線軌道上ではあたってるはずでも, 運良くカスる形で当たらない可能性があります. 1マスの大きさが `1.0f`, 玉の移動に関する物理ダイナミクス (増加量) が `0.05f` で定義されているため, ほとんどありえませんが,

6/22配布版の状態では玉は何か当たるまで直進し続けましたが, 6/29版より, 風の概念を追加し玉の軌道が曲るようになっています.

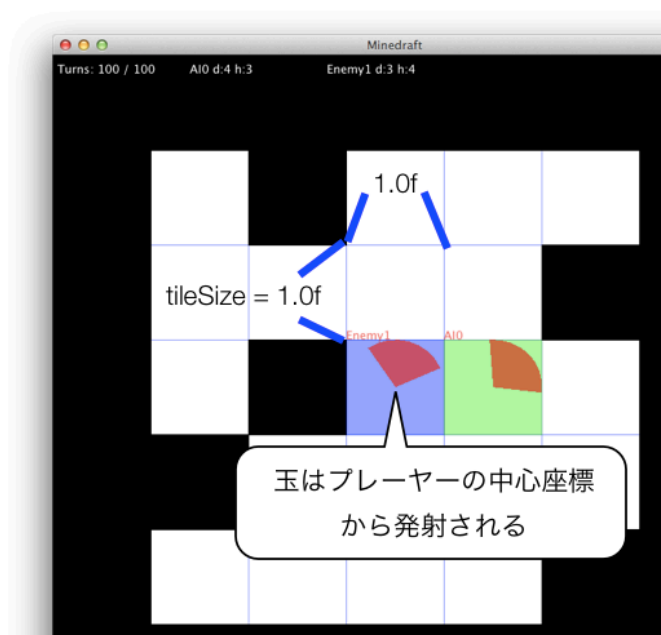
風について

前回までは玉は投げると、まっすぐ進むことが保証されていたが、特定の向き、強さで風が吹いて玉が曲がりながら飛ぶようになりました。

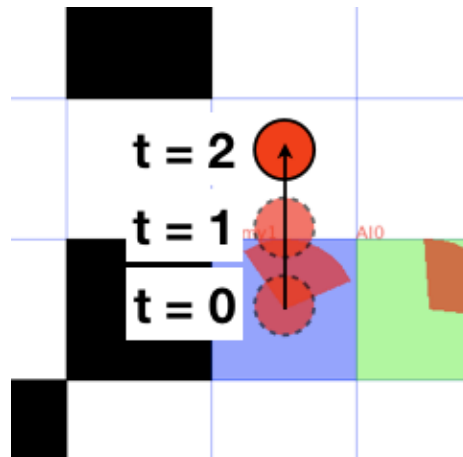


上記図のように、風によって本来の玉の軌道とは違う場所に着弾する可能性があります。また、風の情報は取得できず、玉を投げた角度、着弾位置から自分で計算しましょう。

この風の概念によって、今まで意識することがなかったフィールド内部で扱っていた座標系も考慮する必要があります。まず、大事なのが1マスの大きさをどのように定義しているかで、風が軌道に与える影響が変わってきます。ゲームで用いるマスの大きさは Boardクラスの tileSize変数で管理されており、1.0fで設定されています。



また、玉は投げた人の中心座標から発射されています。



この図のように、投げた人の中心から玉が発射($t = 0$)してから、玉の持つベクトルに従って n 秒後の玉の位置が決定します。

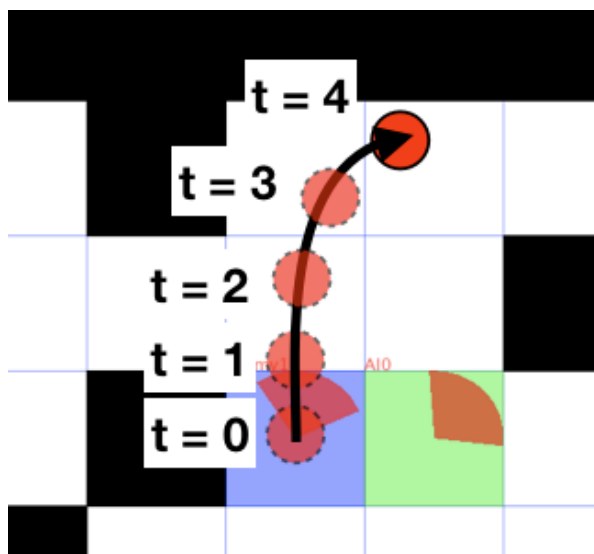
風を考慮していない場合の玉の位置計算式は以下の通りです ($t = 1$)。

```
// 自分のインスタンス取得↓
Player me = (Player)board.getME(); ↓
// 自分の位置取得↓
Point me_point = board.getPosition(me.getID()); ↓
// 玉の位置初期化 投げた人の中心座標↓
float arrow[] = { ↓
    // arrow[0] -> 玉(弓矢)のx座標, arrow[1] -> 玉のy座標 ↓
    me_point.x*board.tileSize+board.tileSize/2, ↓
    me_point.y*board.tileSize+board.tileSize/2 ↓
}; ↓
// 玉の移動に関する物理ダイナミクス分だけ移動する ↓
arrow[0] += Math.cos(angle)*Piece.DYNAMICS; ↓
arrow[1] += Math.sin(angle)*Piece.DYNAMICS; ↓
// 盤上の座標からフィールドにおけるポイントに変換 ↓
Point bpoint = board.convertRealToBoard(arrow[0], arrow[1]); ↓
```

arrowは玉の盤上における位置（Pointではありません）を表現しているfloat型の2次元配列です。玉を投げた人の中心座標に初期化して、玉の移動に関する物理ダイナミクス分だけ移動します。そして、玉の盤上の座標からフィールドにおけるポイントに変換します。以上の手順で $t = 1$ のときの玉の位置を予測することができます。 $t = n$ のときに玉の位置がどうなるかは各自考えてください。

次は風の影響を考慮した場合を考えます。風は玉に対し、その方向に応じて移動ベクトルを加算します。ただし、常に一定量を与えると仮定すると、玉の軌道は単純に斜めに移動することになります。それでは玉の軌道が曲るようなものを表現できていません。なので時間に応じて加算量が変化するようなモデルを仮定します。さらに線形な単純加算では

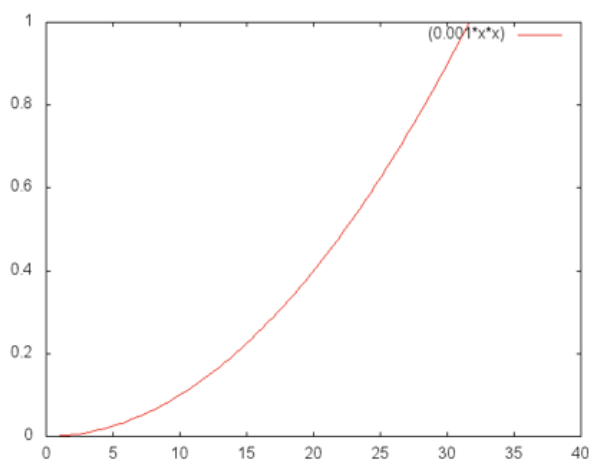
円弧を描くだけになるので、指数関数的に増加するモデルを考えます。また、現実では空気抵抗などの影響により最大速度が決まっているのでそれを表現する必要があります。



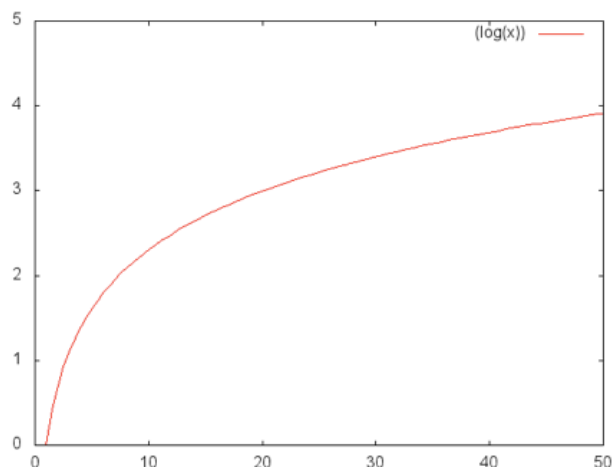
この通り、 $t=1$ ではあまり影響がないが、時間が進むに連れて影響が大きくなる状況を目指します。

以上から以下の通り、モデル化を行いました。

$$y = \begin{cases} \log t & (y > 1) \\ 0.001 * t * t & (otherwise) \end{cases}$$



$$y = 0.001 * t * t$$

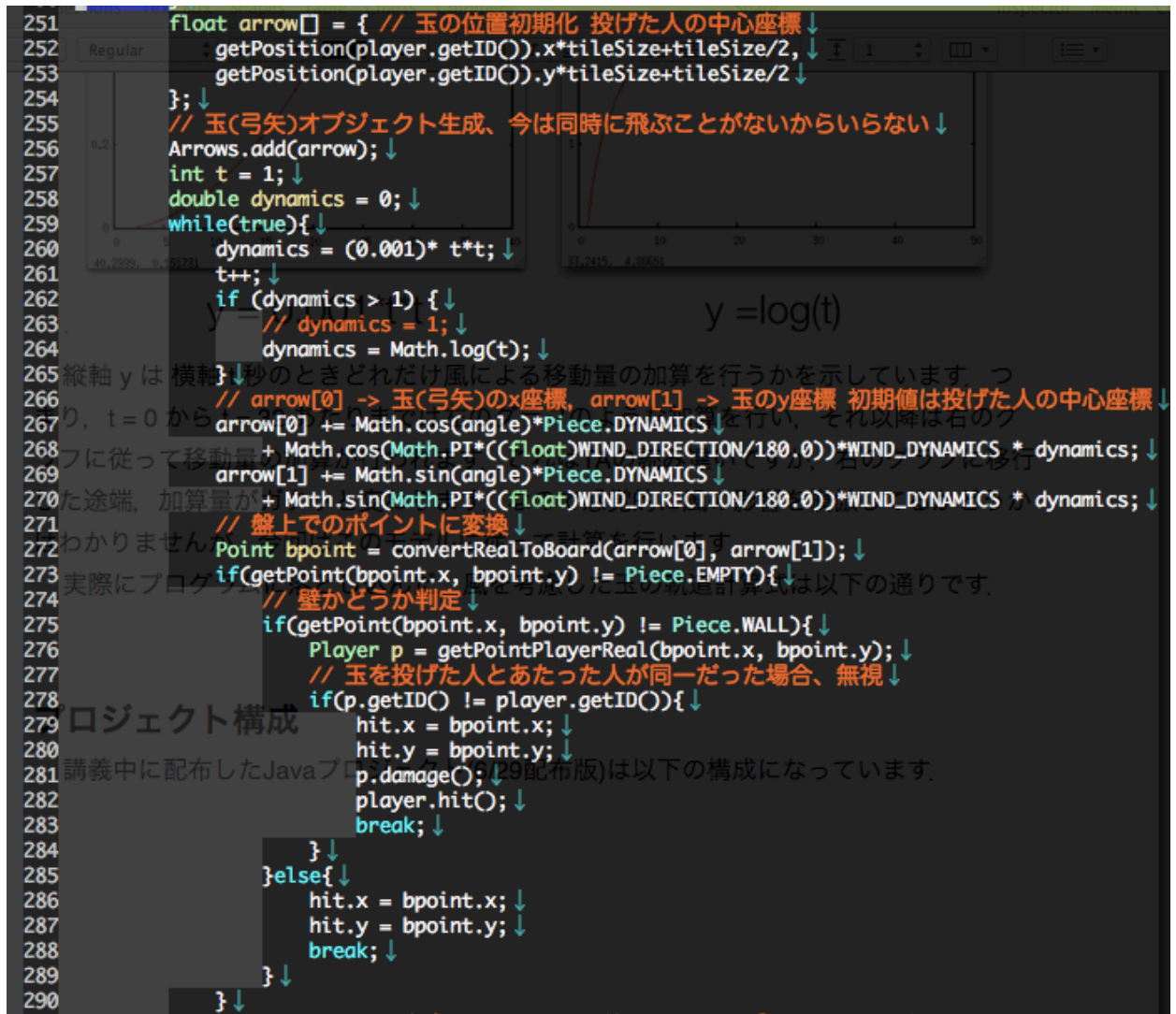


$$y = \log(t)$$

縦軸 y は 横軸 t 秒のときどれだけ風による移動量の加算を行うかの係数を示しています。つまり、 $y > 1$ のとき、 $t=0$ から $t=32$ あたりまでは左のグラフのような加算を行い、それ以降は右のグラフに従って移動量の加算が行われます。これはTAの読み違いです

が、右のグラフに移行した途端、加算量が大きく変わります。なので感覚的に風の影響を模倣してるかどうかはわかりませんが、今回はこのモデルに従って計算を行います。

実際にプログラムに落とし込んだ、風を考慮した玉の軌道計算式は以下の通りです。



```

251 float arrow[] = { // 玉の位置初期化 投げた人の中心座標 ↓
252     getPosition(player.getID()).x*tileSize+tileSize/2, ↓
253     getPosition(player.getID()).y*tileSize+tileSize/2 ↓
254 };
255 // 玉(弓矢)オブジェクト生成、今は同時に飛ぶことがないからいらない ↓
256 Arrows.add(arrow); ↓
257 int t = 1; ↓
258 double dynamics = 0; ↓
259 while(true){ ↓
260     dynamics = (0.001)* t*t; ↓
261     t++; ↓
262     if (dynamics > 1) { ↓
263         // dynamics = 1; ↓
264         dynamics = Math.log(t); ↓
265     }
266     // arrow[0] -> 玉(弓矢)のx座標, arrow[1] -> 玉のy座標 初期値は投げた人の中心座標 ↓
267     arrow[0] += Math.cos(angle)*Piece.DYNAMICS ↓
268     // 玉のx座標に風の影響を加算
269     arrow[0] += Math.cos(Math.PI*((float)WIND_DIRECTION/180.0))*WIND_DYNAMICS * dynamics; ↓
270     // 玉のy座標に風の影響を加算
271     arrow[1] += Math.sin(Math.PI*((float)WIND_DIRECTION/180.0))*WIND_DYNAMICS * dynamics; ↓
272     // 盤上でのポイントに変換 ↓
273     Point bpoint = convertRealToBoard(arrow[0], arrow[1]); ↓
274     if(getPoint(bpoint.x, bpoint.y) != Piece.EMPTY){ ↓
275         // 壁かどうか判定 ↓
276         if(getPoint(bpoint.x, bpoint.y) != Piece.WALL){ ↓
277             Player p = getPointPlayerReal(bpoint.x, bpoint.y); ↓
278             // 玉を投げた人とあたった人が同一だった場合、無視 ↓
279             if(p.getID() != player.getID()){ ↓
280                 hit.x = bpoint.x; ↓
281                 hit.y = bpoint.y; ↓
282                 p.damage(); ↓
283                 player.hit(); ↓
284                 break; ↓
285             }
286         }
287     }
288     else{ ↓
289         hit.x = bpoint.x; ↓
290         hit.y = bpoint.y; ↓
291         break; ↓
292     }
293 }

```

このプログラムはAIクラス用のサンプルプログラムではなく、実際にBoardクラスで定義されている玉の軌道計算プログラムの抜粋です。前回(6/29)配布時には風の方向計算時にキャストミスがあり、水平方向にしか風が吹かない状況になっておりましたが、これは現在上記の通り、修正済みです。公式戦はこの条件で行われます。

風の向きおよび強さを定義している WIND_DIRECTION および WIND_DYNAMICS は Boardクラスのプライベート変数として定義されています。なので、AIクラスからはこの変数の値にアクセスすることができません。なので、発射角度と着弾位置（throwing関数は返り値に着弾位置のポイントを返す）からそれらを予測するシステムが必要になります。AIクラスでどのようにして風の影響を受けた玉の軌道計算をするべきかは各自考えてください。

プレイヤー/フィールド情報の取得

AIクラスからある程度、Boardクラスで管理されているフィールドの情報にアクセス可能です。 (※ 授業中に配布したスライド中の説明が途中からずれていました。正しくはこちらです)

- `Vector<Point> getMovablePos()`
自分が移動することができるすべての位置を返す
- `Vector<Point> getPlayers()`
すべてのプレイヤーの位置を返す
- `Vector<Player> getPlayersObject()`
すべてのプレイヤー インスタンスを返す
- `Vector<Point> getPlayersPosition()`
すべてのプレイヤーの位置を返す
- `Vector<Point> getEnemies()`
敵(自分以外のグループに所属しているすべてのプレイヤー)の位置を返す
- `Vector<Player> getEnemiesObject()`
敵(自分以外のグループに所属しているすべてのプレイヤー)のインスタンスを返す
- `Vector<Point> getEnemiesPositon()`
敵(自分以外のグループに所属しているすべてのプレイヤー)の位置を返す
- `Vector<Point> getMembers()`
自分(と同じグループに所属しているプレイヤー)の位置を返す
- `Vector<Player> getMembersObject()`
自分(と同じグループに所属しているプレイヤー)のインスタンスを返す
- `Vector<Point> getMembersPositon()`
自分(と同じグループに所属しているプレイヤー)の位置を返す
- `Player getME()`
自分のプレイヤーインスタンスを返す
- `Vector<Point> getHazard()`
すべての障害物の位置を返す
- `void showBoard()`
フィールド情報を出力
- `int[][] getBoard()`
フィールドを返す(コピーなのでこれを直接弄ってもゲームに影響はない)
- `int getTurn()`
今 何ターン目か返す
- `int getMaxTurn()`
最大ターン数を返す
- `Point getPosition(int id)`
指定されたID (プレイヤーID) のオブジェクトがどこにいるのか返す
- `int getPoint(int x, int y)`
指定したフィールド上に何がいるのか返す(空白 壁 グループID のどれか)
- `Player getPointPlayer(int x, int y)`
指定した盤上の位置いるプレイヤーインスタンスを返す

- Point convertRealToBoard(float x, float y)

数値的な盤上の位置からフィールドにおけるポイントを返す

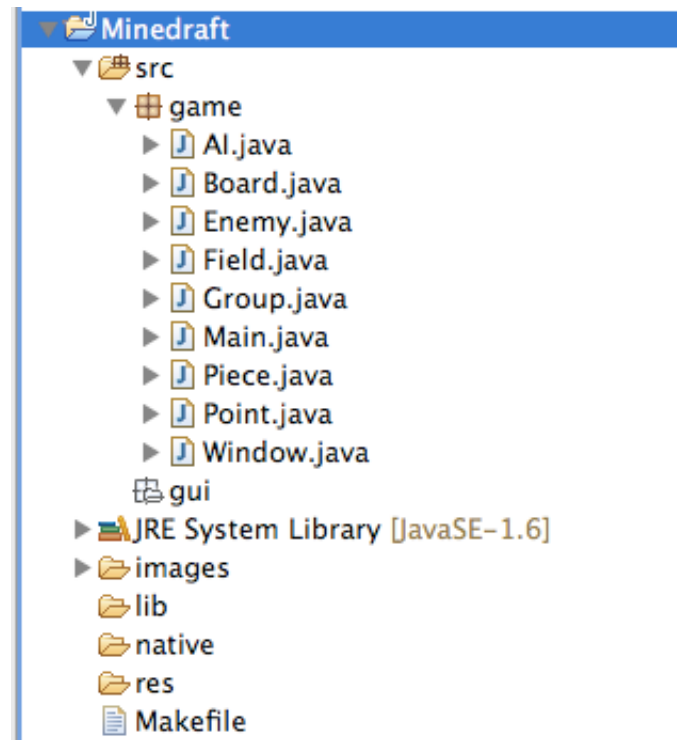
また、取得したプレイヤーインスタンスから、以下の情報も取得することができます。

```
6
7 interface Player{
8     public static final int MAX_ENERGY = 1000;
9     public static final int THROW_VAL = 300;
10    public static final int MOVE_VAL = 200;
11    public static final int REFRESH_VAL = 1000;
12    public Object clone();
13    public int getType();
14    public int getID();
15    public int getGroupID();
16    public int getDamage();
17    public int getHitCount();
18    public float getAngle();
19    public float setAngle(float angle);
20    public String getName();
21    public int getEnergy();
22    public int damage();
23    public int hit();
24    public boolean refresh();
25    public boolean spendEnergy(int energy);
26    public void onTurn(Board board) throws Exception;
27 }
```

- int getID()
このプレイヤーIDを返す
- int getGroupID()
このプレイヤーが所属するグループIDを返す
- int getDamage()
このプレイヤーが玉を受けた数を返す
- int getHitCount()
このプレイヤーが玉を当てた数を返す
- float getAngle()
このプレイヤーの向きを返す (0~360度)
- String getName()
このプレイヤーの名前を返す

プロジェクト構成

講義中に配布したJavaプロジェクト(6/29配布版)は以下の構成になっています。



基本的に src ディレクトリにある game パッケージを変更します。

- **AI.java**
プレイヤーがどういう振る舞いをするか決定するためのAIが記述されたクラス
- **Board.java**
フィールドを定義しているクラス
- **Enemy.java**
敵プレイヤーがどういう振る舞いをするか決定するためのAIが記述されたクラス
- **Field.java**
画面描画用クラス
- **Group.java**
プレイヤーを管理するグループを定義したクラス
- **Main.java**
本プロジェクトのメインクラス
ゲームのメインスレッドとAIの抽象クラスが記述されている
- **Piece.java**
フィールド上に配置しているオブジェクトのインデックスや静的な変数などを定義して

いるユーティリティクラス

- **Point.java**
ゲーム中のフィールド表現でよく使う二次元のインデックスを保持したクラス
- **Window.java**
画面描画用のフレームを生成するクラス

AI 作り方

AIは `AI.java` ファイルを編集することで作成してもらいます。ゲーム中に自分に行動権利が回ってきたら AIクラスを継承した `AiAlgorithm`クラスの `move`関数が呼ばれます。`move`関数は引数に現在の盤面を表現している`Board`インスタンスが与えられます。この`Board`クラスには便利な盤面情報を取得するAPIがあります。この詳細は本マニュアルの「プレイヤー/フィールド情報の取得」項を参照してください。このAPIからプレイヤーインスタンスやフィールドを表現した変数などが取得可能ですが、このインスタンスに対して直接ダメージを与える関数などは呼び出し可能ですが、意味がありません。返ってくるインスタンスはゲーム中で扱われているものと値は同じですが、参照を渡しているわけではなくただのコピーです。ご注意ください。

作成上の注意点としてAI中でクラス変数や内部クラスの作成は許可しています。提出していただいたファイルが与えられたメインファイルで動作すれば基本的に問題ありません。

ゲーム内制御変数

プログラム中のゲームを制御している変数について紹介します。

- GUI 機能 (Main.java : boolean iswindow = true)

```
175 private static int SLEEP_TIME = 100; ↓
176 public static final boolean iswindow = false; ↓
```

初期状態ではプログラムの実行時にGUI（グラフィカルユーザーインターフェース）が出現します。ゲームの進捗を確認したい場合には一番分かりやすいかと思いますが、対戦結果だけ知りたい場合には不要な場合があります。このiswindow変数をfalseにすることで起動時にウィンドウを生成しません。また実行時のコマンドライン変数に nw を与えることでも同様の効果があります。

- グループで対戦する機能 (Main.java : ENEMY_NUM, FRIEND_NUM = 1)

```
163 public class Main implements Runnable{ ↓
164     final static int ENEMY_NUM = 1; ↓
165     final static int FRIEND_NUM = 1; ↓
166     final static int INDENT_NUM = 0; ↓
167     final static int GROUP_NUM = 2; ↓
```

初期状態では1対1の対戦になってますが、この変数の値を変更することでチームの人数を変えることができます。ENEMY_NUM (Group1) は敵チームの数、FRIEND_NUM (Group0) は味方チームの数です。ここで言う敵、味方という概念は基本的には存在していませんが、初期状態でAIクラスで初期化されたプレイヤーがGroup0、Enemyクラスで初期化されたプレイヤーがGroup1に割り振られており、それらを制御しているため、便宜上敵 味方という表現を使っています。

- フィールドの大きさを変える (Board.java : WIDTH, HEIGHT = 20)

```
6 public class Board{ ↓
7     public static final int WIDTH = 20; ↓
8     public static final int HEIGHT = 20; ↓
9     private static float tileSize = 1.0f; ↓
10    public static final int MAX_TURNS = 60; ↓
11    private static int SLEEP_TIME = 10; ↓
```

フィールドの横、縦のカラム数を制御しています。ちなみに tileSizeはゲームフィールド上でのマスと軌道計算に用いる縮尺を定義しています。

- ゲームスピードを変える (Board.java : SLEEP_TIME, Main.java : SLEEP_TIME)

```
173 private static int SLEEP_TIME = 100; ↓
```

Board.javaのこの変数は玉を投げた時の描画スピードを制御してます。

Main.javaはプレイヤーの行動権利切替時にラグを入れるための変数です。

小さくすればそれぞれ早くなりますが、GUIで描画していても早すぎてよくわからなく

なるため挿入されています。大きくするとゲームの流れを追いやすくなるかも知れませんが、実行しているマシンの性能に大きく左右されるので最適な数値は自分で見つけると良いと思います。

- ・ゲーム終了までのターン数を変える（Board.java：MAX_TURNS = 100）

```
10 public static final int MAX_TURNS = 100; ↓
```

ゲーム終了までのターン数を表しています。

- ・風を固定する（Main.java：board）

```
209 board = new Board(ENEMY_NUM, players, this); ↓
210 // 4, 5 引数にゼロをセットすると風なし状態で初期化 ↓ トを移
211 // board = new Board(ENEMY_NUM, players, this, 0, 0f); ↓
212 window.setVisible(true);
```

今回から風の概念が追加されましたが、デバッグ時に邪魔に感じることもあります。そんなときはこのBoardクラスの初期化時に4, 5引数の値を0でセットすると無風状態を表現します。4が風の強さ、5が風の向きとなっています。

- ・プレイヤーの名前変更（Main.java）

```
193 for(int i=0; i<FRIEND_NUM; i++){ ↓
194     String name = "AI"+i; ↓
195     System.out.println("join up "+name); ↓
196     players.add(new AIPlayer(new AiAlgorithm(), name, i, 0)); ↓
197     groups.get(0).join(players.get(i)); ↓
198 } ↓
199 for(int i=FRIEND_NUM; i<ENEMY_NUM+FRIEND_NUM; i++){ ↓, i, 0)); ↓
200     String name = "Enemy"+i; ↓
201     System.out.println("join up "+name); ↓
202     players.add(new AIPlayer(new EnemyAlgorithm(), name, i, 1)); ↓
203     groups.get(1).join(players.get(i)); ↓
204 }
```

初期はAI0とかEnemy1とかの味気ない名前でしたが、設定できます。

Main.javaの初期化時にAIPlayerを生成していますが、この時に第二引数として渡している文字列がAIの名前です。本来ならば皆さんが設定できるようにAiAlgorithm側に実装を回すべきだったかも知れませんが、

ということで、本バージョンから名前をAI側に実装しました。抽象クラスで定義してあるので、ないとエラーがでます。ご注意ください。

```
4 class AiAlgorithm extends AI ↓
5 { ↓
6     public AiAlgorithm() ↓
7     { ↓
8         this.name = "AI"; ↓
9         this.start_time = 0; ↓
10        this.limit_time = Long.MAX_VALUE; ↓
11    }
```

公式戦

提出していただいたAI.javaファイルは以下の条件で総当りのリーグ対戦を行います。

形式	チーム戦
人数	3 x 3
最大ターン数	100
フィールド	20 x 20
ゲーム勝利条件	グループ間で当てた数 / 当てられた数の差が大きい方
マッチ数	3回勝負

コンパイルエラーなどは状況によってはある程度こちらで考慮するかもしれませんが、基本的に動かないプログラムは不戦敗とします。なので最低でも動くプログラム

(AI.java) を提出してください。また、勝ち点は（勝ち：2, 引き分け：1, 負け：0）として最終的に順位をつけて発表しようと思います。

よきAI作成への道

知的なAIを作成するためにはセンス・ジャッジ・アクトの3つのフェーズを理解することが重要だと考えます。例えばセンスは自分の周りに何があって、相手/味方がどこにいるか、風の方向・強さなどの情報取得があげられるでしょう。ジャッジは得られた情報から自分の振る舞いを考え、アクトで実際に行動を起こします。そして自分が環境に与えた影響を再びセンスし、ジャッジ～アクト と繰り返します。この繰り返しが基本となりますが、重要なのは行き当たりばったりに行動しないということです。すなわち自分が起こした行動の影響を考慮して、次はより効率的に行動しようとする学習システムの要素です。このゲームにおいて何を学習するべきか、という点においてはおそらく環境の情報（とくに風）が当てはまると思います。風の情報は未知の変数として与えられていますが、本ゲームにおいて玉の軌道を決定づける重要な変数であるため、環境に与える影響は大きいです。例えばプレーヤーがある角度で玉を発射したとします。環境の情報において、風以外の変数および軌道計算式は取得可能ですのである程度玉の着弾点が予測できます。加えて、自分の放った玉の着弾点が取得可能ですので予測値と実際の着弾点の誤差を考慮することで、風の未知変数の値をある程度最適化できます。風の変数さえ完璧に予測することが出来れば、あなたのAIは狙った的を外すことはなくなるでしょう。この機能を実現するためには、自分の行動履歴を保存しておくことが必要になります。本ゲームのAI作成ではAI.javaファイルを編集してもらうことになりますが、その中でクラス変数や内部クラスの作成は許可しています。それらを駆使すれば実装は可能でしょう。提出していただいたファイルが与えられたメインファイルで動作すれば基本的に問題ありません。

これ以外にも相手と自分の相対位置や、チーム対戦特有の問題などをうまく使うことでより強いAIが作成できるのではないのでしょうか。皆さんには楽しんで自分の考えたAIを実装してください。楽しみにしています。

FAQ ~よくある質問~

実際にあがった質問 and 個人的に説明するべきだったと思ってる点などを一問一答形式で記載します。AI作成の参考にしてください。

1. Q : angleに45とか入力しても、右上に玉を投げてくださいに右下に投げています。反時計回りの座標系になってませんか

A : ウィンドウ上で表現されているフィールドは左上原点です。なので縦のY軸方向は加算すると下に向かって増加します。内部的には反時計回りで扱っていますがこの理由から上下反転しているように見えます。これの関しては混乱を避けるために左下原点に変換することも考えましたが、すでにAIを作ってくれている人のことも考えて変更しないことにしました。ご了承ください。

また、プレイヤーの向きに関してはこの通り扱っていなかったのが向いている方向と上下反対方向に玉が発射されていました。これに関しては最新版で修正済みです。

2. Q : showBoard関数（現在の盤面をコンソールに出力する関数）で出力される結果とGUIで表現されている結果が、間違っていないですか

A : 間違っていました。出力時の縦横のインクリメント変数が入れ替わっていました。最新版では修正済みです。

3. Q : getPosition関数の挙動がおかしい

A : 間違っていました。一致するプレイヤーIDを持つ位置を返すつもりがオブジェクトID（障害物・空白・グループID）で一致する位置を返してました。修正済みです。

4. Q : ゲームスピード変えられないのか

A : Boardクラスで定義しているint SLEEP_TIME変数で玉の進むスピードをコントロールしています。この値を小さくすると描画のためにあえてゆっくりにしていた部分が早くなり、劇的にゲームスピードが早くなります。ただ、早くしすぎると玉が見えなくなります。

さらに早くする場合にはMainクラスで定義されているboolean iswindow変数をfalseにしてください。ゲームスタート時にウィンドウを生成しないコンソールモードで起動するので結果が早く出ます。ただ何やってるのかわかりません。

5. Q : 変更するのはAIクラスを継承したAiAlgorithmクラスのmoveメソッドだけですか

A : 提出するのがAI.javaファイルで、こちらが用意しているメイン文でそのファイルが動作すれば基本的に問題ありません。なのでAiAlgorithmクラスのクラス変数や内部クラスなどを駆使して高度なAIを記述してください。今までの行動履歴などはAPI側で用意していないので、自分でクラスを作って管理するとよいでしょう。

6. Q：初期配置ランダムとか言いながら毎回一緒じゃないですか

A：その通りです。Boardクラスのコンストラクタから初期化関数を呼び出していますが、その中でランダム関数に定数のシードを与えています。本対戦では完全ランダムに乱数を発生させます。プレイヤーの初期行動順序も同様です。インスタンスが生成された順になっていますが、本番ではランダムになります。
説明不足により、誤解を招く可能性があったので最新版では完全ランダムな状態でリリースします。

7. Q：throwing 関数中で用いられてる風の計算式間違ってますか

A：間違っていました。風の向き計算中で意図せずint型にキャストされて水平2方向しか表現できていない状態になっていました。この問題は最新版では修正済みで正しく360度風の影響がでます。

8. Q：Ai.javaファイルちょっと変わった？

A：AiAlgorithmクラスはいじってもいいけど抽象クラスのAIクラスはいじって欲しくないで、別ファイルに定義しました。動作に影響はありません。あとサンプルはたまに変わります。万全を期しているつもりですがサンプルにもミスがあることがあります。みなさんはそんなミスにも惑わされずにAIを作ってくれることを期待しています。あんまりサンプルを信頼しないでください。

あとがき

何度もプログラム修正+説明不足な点があって皆さんにはご迷惑をおかけいたしました。万全を期しているつもりですが、まだミスがあるかも知れません。TAが一人で実装しているため、たまにミスがありますがご容赦ください。なるべく皆さんの参考になるような実装を心がけていますがなかなか難しいですね。私は普段あまりJavaを使ったプログラム実装は行っていないですが、汎用性が高い良い言語だと思いますのでみなさんもこのゲームを通じて楽しんで学習してもらえると幸いです。