計算機科学実験及演習 2
ソフトウェア報告書 1

谷　勇輝

入学年 平成 27 年
学籍番号 1029-27-2870

提出日: 2016 年 10 月 18 日

1　課題 1

1.1　実施内容

(1)

ステージパラメータ（seed、難易度、敵の有無、その他地形等の変数）と既存エージェントを様々に変更し、動作を観察した。

(2)

ch.idsia.agents.controllers パッケージ内に用意された Agent プログラムのうち、ForwardAgent についてソースコードと動作の比較からその内容を理解した。

(3)

各 Agent プログラムに頻繁に登場する isMarioAbleToJump と isMarioOnGround の 2 つのパラメータについて理解を深めるため、以下のソースコードを Agent プログラムの適切な箇所にアサーションし、その内容について考察した。

```
System.out.println(isMarioAbleToJump + "/" + isMarioOnGround + " = " + action[Mario.KEY_JUMP]);
```

1.2　実行結果

(1)

それぞれのパラメータについて、難易度、シードの値を様々に変更し観察し意味を理解した。また、概ね自分の望むタイプのコースを用意できるようになった。特記すべきパラメータについては以下に詳細を記述する。

Hill(丘)

デフォルトで全ての難易度で出現する。

下からのジャンプを透過し、床として使用できる。
marioAIOptions.setHillStraightCount メソッドで false に設定することで出現しなくなる。

Tubes(土管)

デフォルトで全ての難易度で出現する。 パックンフラワーの出現率は難易度で異なるように思われる。
marioAIOptions.setTubesCount メソッドで false に設定することで出現しなくなる。

Gaps(落とし穴)

難易度 1 以上で出現する。
marioAIOptions.setGapsCount メソッドを false にすることで出現しなくなる。
難易度 0 では値を true にしても出現しない。

Cannons(砲台)

難易度 2 以上で出現する。
marioAIOptions.setCannonsCount メソッドを false にすることで出現しなくなる。
難易度 1 以下では値を true にしても出現しない。

DeadEnds(行き止まり)

デフォルトでは出現しない。
難易度に関わらず、marioAIOptions.setDeadEndsCount メソッドで有無を操作できる。
以下のいずれか、もしくは複数の地形が発生する。

・マリオがジャンプによって越えることのできない、地面からの壁
・上方画面外から続く、空中をふさぐ壁。
・上記の 2 つの地形に、その壁に密着するように延びる地面を追加した鍵型の地形。

袋路を形成する可能性があり、後戻りが必要になりうる。

(2)

reset メソッド　以下の五つの設定を行っている。

1. 配列 action の生成（中身は false で初期化されている）
2. 要素 RIGHT を true にする
   この Agent では右ボタンは常に押された状態である。
3. 要素 SPEED を true にする
   この Agent ではダッシュ＆ファイアは常に押された状態である。
4. trueJumpCounter を 0 に初期化する
5. trueSpeedCounter を 0 に初期化する
   これらの 2 変数の内容については後述する。

getAction メソッド

マリオの毎ターンの行動を定めている。まず、マリオが「危険状態」かどうかを判断する。DangerOfAny メソッドが true で、進行方向 1 マス目がコインではないとき、それを「危険状態」としている。

危険状態の時

マリオがジャンプ可能（isMarioAbleToJump==true）の時には要素 JUMP を true とする。また、マリオが空中にいて（!isMarioOnGround==true）直前もジャンプキーを押している際はジャンプキーを押しつづける。

危険状態にないとき

マリオはジャンプボタンを離す。

また、マリオの目の前が障害物で着地しているのにジャンプボタンを離しそこねた場合の詰みを防ぐため、17 ターン連続でジャンプボタンを押している際はボタンを離すようになっている。

DangerOfAny メソッド

以下のいずれかに当てはまる時、true を返す。

マリオの 1 マス前に 2 マス以上の穴がある。（これには空中も含まれる。）

マリオの 1 マスまたは 2 マス前に障害物がある。

マリオの 1 マスまたは 2 マス前に敵がいる。

(3)

一部を抜粋する。

```
<FowardJumprinAgent>
....
false/false = true
false/false = true
false/false = true
false/true = false
true/true = true
false/false = true
false/false = true
...
```

<FowardAgent>
...
false/false = false
false/true = false
true/true = false
true/true = true
false/false = false
false/false = false
false/false = false
false/false = false
false/false = false
false/false = false
false/false = false
true/false = true
false/false = true
false/false = true
false/false = true
...

1.3 結論と考察

(1)

ステージパラメータについては、以下の三つのタイプのパラメータがあると分類することができる。

1. 難易度に関わらずデフォルトで出現するもの
   コイン、ブロック、丘、土管
2. 特定難易度以上でしか出現しないもの落とし穴、砲台
3. 値を true に設定した場合のみ出現するもの行き止まり、Flat、隠しブロック　（隠しブロックについては未検証）

難易度とそれによるステージの変化については、続く課題を行う際に随時確認していきたい。詳しく調査を行うことができたので、適宜適切なステージを構築し、人工知能プログラムの検証等に役立てたいと思う。　また、今回の分析をふまえ、Main クラス内を望むステージを作り易いよう改良した。

```
public final class Main

public static final int SEED = 30,
DIFFICULTY = 0;

public static void main(String[] args)

final MarioAIOptions marioAIOptions = new MarioAIOptions(args);

//ステージパラメータ
marioAIOptions.setLevelRandSeed(SEED);
marioAIOptions.setLevelDifficulty(DIFFICULTY);

// marioAIOptions.setDeadEndsCount(true); //dead_ends
// marioAIOptions.setHiddenBlocksCount(true); //hidden_blocks
// marioAIOptions.setFlatLevel(true); //flat

// marioAIOptions.setCoinsCount(false); //coins
// marioAIOptions.setHillStraightCount(false); //hill
// marioAIOptions.setBlocksCount(false); //blocks
// marioAIOptions.setTubesCount(false); //tubes
// marioAIOptions.setGapsCount(false); //gaps
// marioAIOptions.setCannonsCount(false); //cannons

//敵の有無
// marioAIOptions.setEnemies("off"); //キラーとパックンのみ
// marioAIOptions.setEnemies("g"); //クリボー
// marioAIOptions.setEnemies("ggk");

// エージェントの追加
final Agent agent = new ForwardJumpingAgent();
marioAIOptions.setAgent(agent);

final BasicTask basicTask = new BasicTask(marioAIOptions);
basicTask.setOptionsAndReset(marioAIOptions);
basicTask.doEpisodes(1,true,1);
System.exit(0);
```

SEED,DIFFICULTY の値を変更しやすい位置に移し、各ステージパラメタをコメント扱い

で待機させている。例えば土管を消したいのであれば //tubes の位置のコメントを外せばすぐに実現できる。

(2)

ForwardAgent プログラムは、基本的な危機回避動作を組み込んだプログラムと言える。具体的には以下の三つに対応する事を目的としているようである。それぞれの観点から、実装のよい点と悪いところを述べる。

1. 障害物を回避し、ゴールまで到達すること。
   ジャンプによっておおむね全ての障害物を回避できる。変数 trueJumpCounter を利用した詰み防止も良く機能している。マリオが LARGE の状態の際に、高さの幅がブロックによって 2 に縮まった道が出現すると、小さなジャンプではなく大きなジャンプを選択してしまうことから詰みがおきてしまう。また、袋地に対しては完全に無力である。

2. 敵を回避し、死亡しないこと。
   単純に歩いてくる敵に対してはおおむね対処できる。着地位置に敵がいるかどうかの判定はしていないので、着地後に敵にぶつかることはままある。また、特に上から降りてくる敵に対して脆弱であり、大抵の死亡要因はこれである。

3. 穴を回避し、死亡しないこと。　基本的な穴は回避できる。着地判定をしていないことから、大きなジャンプの先に穴が存在し死亡してしまうことは良く起きる。

レベル 0 のコースをクリアするだけの力はあるが、よりレベルの高いコースに対しては精度の高い実装が必要となる。また、コーディングも全体として甘く、目的にあったプログラムが書けているとは言い難い。getAction メソッド内の「 =!1」の明らかなミスに加え（コインを危険として判断しないことを意図するならば !2 が正しい）、ジャンプの最中も「穴」の危険があると判定してしまっている所など、精密さに欠けているところなどが改善すべき点と言える。

(3)

ForwardJumpingAgent は、できる限り大きなジャンプを行うエージェントである。着地の直後にジャンプボタンを離し、再びジャンプボタンを押す。

アサーションしたログを確認すると、ジャンプ不可/着地済みの場合のみ要素 JUMP が false になっているのが良く分かる。またマリオの状態は、基本的には以下の 3 ステップを繰り返すことが分かった。

1. ジャンプ不可/空中
2. ジャンプ不可/地上
3. ジャンプ可能/地上

敵を踏む際も状態はジャンプ不可/空中である。また、ForwardAgent が観測した「ジャンプ可能/空中」となりうるのは確認できた限り以下の 2 つの場合に限られる。

1. 壁づたいにいて、壁方向にキーを入れている時
2. ブロックなどの先端を擦るように動いた時

前者はいわゆる「壁キック」可能な状態であり、この時にジャンプキーを入れると逆方向に飛び上がることができる。後者は既にブロックに乗っているか微妙な状態での判定のすれ違いによるものと考えられる。

2 課題2

2.1 実施内容

2.2 実行結果

2.3 結論と考察

3 セクション

内容
段落改行は一行あける

3.1 サブセクション

3.2 サブセクション

3.3 箇条書き

表題
内容はここに書く

表題
内容はここに書く

表題に括弧を使う < 括弧 >

3.4 箇条書き（番号付き）

1. 表題
   内容はここに書く
2. 表題
   内容はここに書く

3.5　図の挿入

3.6　スクリーン

単純改行
【括弧】
< 括弧 >