計算機科学実験 2　ソフトウェア報告書 1

新山公太（にいやまこうた）
平成 30 年度入学
1029300562

2019 年 11 月 27 日

1　課題 1

1.1　ステージパラメータの変更

1.1.1　seed 値の変更

seed 値を変更すると、出てくる敵の種類に変化は見られずコースの地形のみに変化が見られた。また地形の変化というのもブロックのパターンに変化があるのみで、落とし穴が増えたり、土管の有無が変わったりすることはなかった。また敵の種類についても seed 値には影響されなかった。以下図 1 を見ることでステージ冒頭の地形が変わっていることが確認できる。

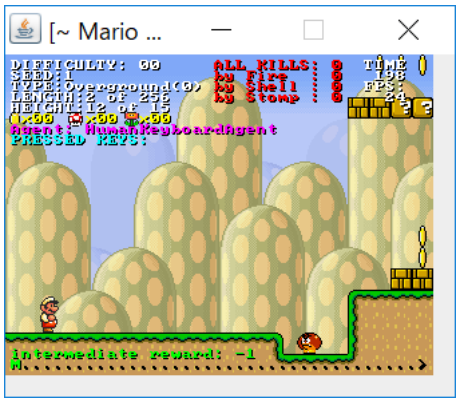

図 1　seed 値 1 のステージ冒頭

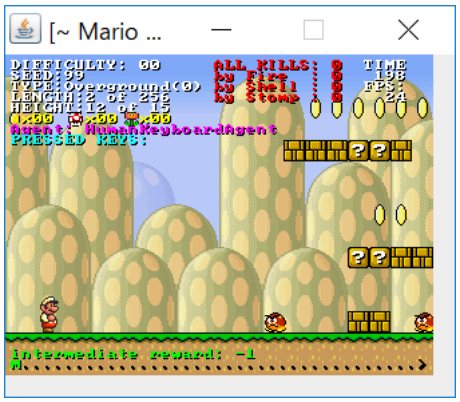

図 2　seed 値 99 のステージ冒頭

1.1.2　難易度の設定

資料に従ってステージの難易度を設定すると、地形に変化は見られず敵の数が多くなった。またデフォルトの設定では出てこなかった敵も出てきたので、一定の難易度以上でないと出現しない敵がいるのかもしれないと思って、後述する設定で出てくる敵を一種類に限定し難易度はデフォルトのままでプレイしてみたところ敵は出現したので、詳しいことはよくわからなかった。また、キラー砲台などデフォルトのステージにはなかったものが増えることもあった。以下の図 3 と図 2 を見比べればキラー砲台が表れていることがわかる。

図 3　難易度を 99 に設定した seed 値 99 のステージ

1.2　出てくる敵の指定

資料に従い様々な敵について出現するかしないかを切り替えた。敵が出てこない設定にしても、パックンフラワーは出てきたがこれは土管がステージに現れないようにすれば解決できると思われる。また、現れる敵については数文字のアルファベットで指定するが、指定する順番を変えても出てくる敵の配置に変化は見られなかった。

1.3　ForwardAgent の説明

1.3.1　reset() メソッドで行われていること

このエージェントの reset() メソッドでは、コントローラのうち「右」と「ダッシュ」のボタンが押される。これはゲーム開始と同時にマリオに右に向かって走れという命令を出していることとなる。ここでは、それらのボタンを押すだけでなく trueJumpCounter および trueSpeedCounter という変数に 0 が代入されている。これら変数は getAction() メソッドが呼び出されたときにそれぞれのボタンが押されている場合にインクリメントされていく。すなわちどれだけの間ボタンが押されていたかのパラメータとなっている。

1.3.2　DangerOfAny() メソッドが返す真偽値

このメソッドは名前の通り障害物がある場合や敵がいる場合に true を返すメソッドとなっている。具体的にはマリオの 1 マス前の地面が 2 マス以上何もない空間である、または、マリオの 1 マス前または 2 マス前が、何もない空間じゃない、または敵がいるときに true を返す。すなわち目の前に段差や障害物があったり、敵がいたりすると true を返す。

1.3.3　getAction() メソッドの説明

このメソッドでは DangerOfAny() メソッドが true を返していて、かつ目の前にある障害物がコインじゃないとき、ジャンプ可能であるかジャンプして空中にいるかならジャンプボタンを押し、目の前に障害物がないか障害物がコインの時、ジャンプボタンを離すという処理と、もしジャンプボタンを押したまま getAction() メソッドが 15 回呼び出されたらジャンプボタンを離すという処理の二つでボタンを押すか押さないかを決め

ている。要約すると、障害物と敵をジャンプで避け、できるだけ高く飛ぶようにしている。

2　課題 2

2.1　ステージをクリアするためのアイデア

まずは資料にあった障害物をよけてゴールを目指すエージェントで MainTask2 をプレイさせてみたところ、見事に落下した。(以下の図を参照)

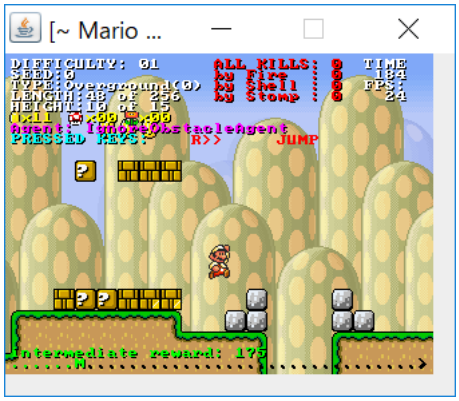

図 4　落下するマリオ 1

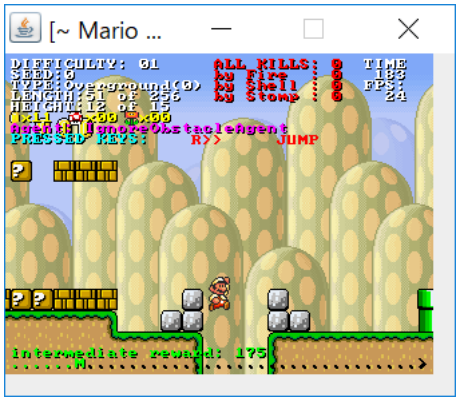

図 5　落下するマリオ 2

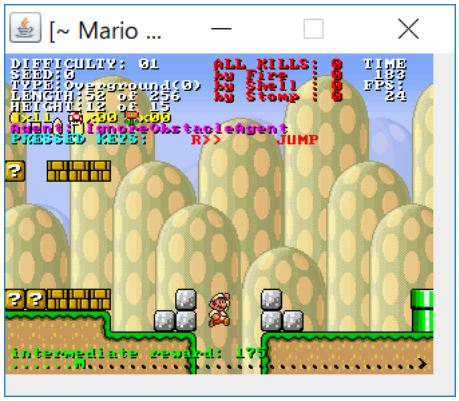

図 6　落下するマリオ 3

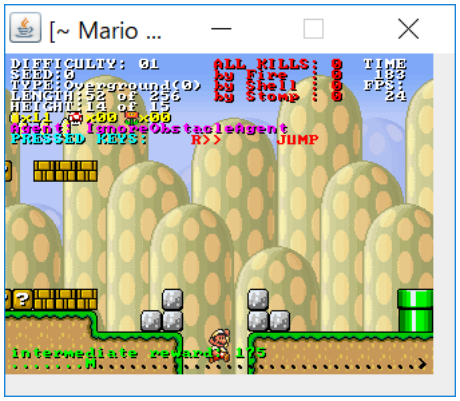

図 7　落下するマリオ 4

これは目の前の障害物を認識してジャンプボタンが押されたためである。このような事態を避けるためには穴に落ちそうなときそれ以上進まないようにすればよいと考えた。では、穴に落ちそうな時とはどのようなときだろうか。今回の図で考えてみればジャンプの飛距離が足りないことが落下の原因だと思われる。つまり、落とし穴をジャンプで避けようとするならば落とし穴の目の前でジャンプをするようにすればいいと考えた。そして飛距離が足りないときは、いったん落とし穴の目の前の足場に着地するようにすればよいのではないかと考えた。

2.2　実装方法

2.3　落とし穴を発見する方法

落とし穴があるかどうかを判定するために、ステージの空間を縦に見ていき、マリオより低い位置に足場がないときに true を返すメソッドを作った。このメソッドには int 型の整数を渡すことで、マリオから見てその分だけ先の位置に落とし穴があるかどうかを判定することができる。

2.4　危ないときに進まないようにする方法

最初は目の前が落とし穴、かつマリオから見て (0,1) 進んだ座標が足場になっているときに左に進むボタンを押すことで進まないようにしていた。しかしこの実装方法では MainTask2 で seed 値 17 の時に落とし穴に落ちてしまう。(以下の図を参照)

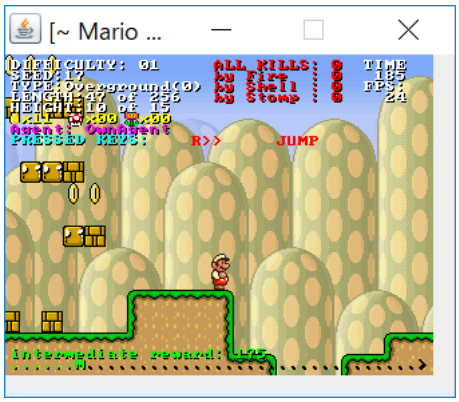

図 8　最初の実装で越えられない落とし穴

このような状況を避けるために、実装方法を少し変えた。このプログラムではマリオから見た座標ではない絶対座標といえる情報を取得することができる。今回はその情報を使うことでマリオが落下中かどうかを判定するメソッドを作った。これは getAction() メソッドが毎フレームごとに呼び出されるのを利用することで可能であった。マリオが落下中で目の前に落とし穴があるときマリオは一度その場に着地するように左のボタンが押される。そうすることで、マリオは一度地面に着地してから万全の状態で落とし穴を飛び越えることができる。実際にはさらに万全を期すために落とし穴の上を通っているときにはダッシュボタンも押すようにしている。

2.5 stdout

```
[~ Mario AI Benchmark ~ 0.1.9]

[MarioAI] ~ Evaluation Results for Task: BasicTask
        Evaluation lasted : 47972 ms
         Weighted Fitness : 7200
             Mario Status : WIN!
               Mario Mode : FIRE
Collisions with creatures : 0
     Passed (Cells, Phys) : 256 of 256, 4096 of 4096 (100%
         passed)
 Time Spent(marioseconds) : 78
  Time Left(marioseconds) : 122
             Coins Gained : 65 of 227 (28% collected)
      Hidden Blocks Found : 0 of 0 (0% found)
       Mushrooms Devoured : 0 of 0 found (0% collected)
         Flowers Devoured : 0 of 2 found (0% collected)
              kills Total : 0 of 0 found (0%)
            kills By Fire : 0
           kills By Shell : 0
           kills By Stomp : 0
    PunJ : 0

 min = 0.0
 max = 0.0
 ave = 0.0
 sd  = NaN
 n   = 1
```