```
 1  using UnityEngine;
 2  using System;
 3
 4  public class BotMovement : Movement
 5  {
 6      // For the neural network input
 7      private const double yOffset = 0.61;
 8      private const double xOffset = 0.44;
 9      public bool Tester = false;
10      private NeuralNetwork Controller;
11      private float Timer;
12      private float PreviousPosition;
13
14      protected override void Start()
15      {
16          if (Tester)
17          {
18              // Initialises the player object
19              Speed = M.GetSpeed();
20              JumpHeight = M.GetJumpHeight();
21          }
22          else
23          {
24              M.InitialisePlayer(Name);
25              Controller = new NeuralNetwork("NeuralNetwork2");
26              // Initialises the player object
27
28              Speed = M.GetSpeed();
29              JumpHeight = M.GetJumpHeight();
30          }
31          Timer = Time.time;
32          PreviousPosition = transform.position.x;
33      }
34
35      public void InputNetwork(string path)
36      {
37          Controller = new NeuralNetwork(path);
38      }
39
40      public void Modify(int place)
41      {
42          Controller.Modify(Stats.BotsMade, place);
43          Stats.BotsMade++;
44      }
45
46      public void InputSpeed(float speed)
47      {
48          Speed = speed;
49      }
50
51      public void InputJumpHeight(float jumpheight)
52      {
53          JumpHeight = jumpheight;
```

```
54      }
55
56      public NeuralNetwork GetController()
57      {
58          return Controller;
59      }
60
61      public int GetGeneration()
62      {
63          return Controller.GetGeneration();
64      }
65
66      public void IncreaseGeneration()
67      {
68          Controller.IncreaseGeneration();
69      }
70
71      protected override int[] SelectedMove()
72      {
73          // Helps to prevent the bot getting stuck
74          if (transform.position.x != PreviousPosition)
75          {
76              Timer = Time.time;
77          }
78          else
79          {
80              if (Timer > Time.time + 2)
81              {
82                  int[] output = new int[] { 0, 1 };
83                  return output;
84              }
85          }
86
87          // Finds the input values
88          // 0-6 refer to tiles and 7 refers to how far along the bot is
              across a tile
89          double[] input = new double[8];
90
91          int xTile = Convert.ToInt32(Math.Floor(transform.position.x -
              xOffset));
92          int yTile = Convert.ToInt32(Math.Floor(transform.position.y -
              yOffset)) - 1;
93          for (int counter = 0; counter < 7; counter++)
94          {
95              int x = xTile;
96              int y = yTile;
97              if (counter > 4)
98              {
99                  y += 3;
100             }
101             else if (counter > 3)
102             {
103                 y += 2;
```

```
104             }
105             else if (counter > 2)
106             {
107                 y += 1;
108             }
109             else if (counter == 0)
110             {
111                 y -= 1;
112             }
113 
114             if (counter == 0 || counter == 2 || counter == 3 || counter == 4
                    || counter == 6)
115             {
116                 x += 1;
117             }
118 
119             if (y < 0 || y >= Stats.TileColumns[x].Length)
120             {
121                 input[counter] = -1;
122             }
123             else
124             {
125                 if (Stats.TileColumns[x][y] == 1)
126                 {
127                     input[counter] = 1;
128                 }
129                 else if (Stats.TileColumns[x][y] == 0)
130                 {
131                     input[counter] = -1;
132                 }
133             }
134         }
135         if (transform.position.x - xTile > 0.7)
136         {
137             input[7] = 1;
138         }
139         else
140         {
141             input[7] = -1;
142         }
143 
144         try
145         {
146             int[] decision = Controller.Decision(input);
147 
148             int[] output = new int[2];
149             // Output 0 determines whether or not the bot will jump
150             output[0] = decision[0];
151             // Decision 1 is whether or not the player moves left. Decision
                  2 is whether they move right
152             // If both are 1, then the player doesn't move
153             if ((decision[1] == 1 && decision[2] == 1) || decision[1] == 0
                  && decision[2] == 0)
```

```
154             {
155                 output[1] = 0;
156             }
157             else if (decision[1] == 1)
158             {
159                 output[1] = 1;
160             }
161             else if (decision[2] == 1)
162             {
163                 output[1] = -1;
164             }
165             return output;
166         }
167         catch
168         {
169             // If there is an error, the bot will move to the right and jump ↵
                   as this is usually a good move to make
170             int[] output = new int[] { 1, 1};
171             return output;
172         }
173     }
174 }
```