```
 1  using System.Collections.Generic;
 2  using System;
 3  using UnityEngine;
 4
 5  public abstract class Master : MonoBehaviour
 6  {
 7      public GameObject VirtualCamera;
 8      protected List<Player> Players = new List<Player>();
 9      public ProceduralGeneration ProGen;
10      protected int ChunkLength;
11      public float Speed = 1f;
12      public float JumpHeight = 1f;
13      protected int CameraLine = 1000;
14      public bool Arena = false;
15      protected int ArenaCounter = 0;
16
17      protected virtual void Start()
18      {
19          InitialiseCamera();
20      }
21
22      // Moves the camera to the correct position
23      protected void InitialiseCamera()
24      {
25          float width = Camera.main.orthographicSize * Camera.main.aspect;
26          float height = Camera.main.orthographicSize;
27          VirtualCamera.transform.position += new Vector3(width + 1, height,
              0f);
28          ChunkLength = (int)Math.Ceiling(width * 2);
29          CameraLine = ChunkLength - 10;
30      }
31
32      public virtual void InitialisePlayer(string name)
33      {
34          Players.Add(new Player(name, ChunkLength));
35      }
36
37      public float GetSpeed()
38      {
39          return Speed * 4;
40      }
41
42      public float GetJumpHeight()
43      {
44          return JumpHeight * 7;
45      }
46
47      // Resolves the movement of a player
48      public void MovePlayer(string name, float distance)
49      {
50          var currentPlayer = GetPlayer(name);
51
52          currentPlayer.IncreaseDistance(distance);
```

```
53          MoveGeneration(name);
54          MoveCamera();
55      }
56
57      // Checks if either player is far away from the camera enough that it
          needs to move
58      private void MoveCamera()
59      {
60          foreach (Player player in Players)
61          {
62              if (player.CameraDistance() > CameraLine)
63              {
64                  float distanceChange = player.CameraDistance() - CameraLine;
65                  VirtualCamera.transform.position += new Vector3
                      (distanceChange, 0f, 0f);
66                  foreach (Player p in Players)
67                  {
68                      p.MoveCamera(distanceChange);
69                  }
70                  return;
71              }
72          }
73      }
74
75      // Checks if any player has moved the minimum tiles that are needed for
          another generation to be needed
76      private void MoveGeneration(string name)
77      {
78          if (GetPlayer(name).GenerationDistance() > ChunkLength)
79          {
80              if (ArenaCounter <= 0)
81              {
82                  ProGen.CreateNewTiles();
83              }
84              else
85              {
86                  ArenaCounter--;
87              }
88              foreach (Player player in Players)
89              {
90                  player.MoveGeneration(ChunkLength);
91              }
92          }
93      }
94
95      // Returns a player based on their name
96      protected Player GetPlayer(string name)
97      {
98          foreach (Player player in Players)
99          {
100             if (player.GetName() == name)
101             {
102                 return player;
```

```
103                 }
104             }
105             throw new Exception();
106         }
107 
108         // Checks if the player is too far to the left or too far down
109         public bool CheckDeath(string name, float yPosition)
110         {
111             var currentPlayer = GetPlayer(name);
112 
113             if (currentPlayer.CameraDistance() < 0.5 || yPosition < -1)
114             {
115                 ResolveDeath(name);
116                 return true;
117             }
118             return false;
119         }
120 
121         protected abstract void ResolveDeath(string name);
122     }
```