```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  /* Handles player movement given data from other scripts(selectedCharacter,
     FloorDetection).
5   *
6   * NOTE: FIX BOOLEANS, we can be using selectedCharacter's boolean here and im
     sure
7   *       there are more improvements that can be made.
8   *
9   *       FIX IF STATEMENTS, Im sure these are just repetitive and unecessary.
10  */
11 public class PlayerController2D : MonoBehaviour
12 {
13     GameObject[] levelStairs;                   // Stores coordinates of our
       stairs' start and end here.
14
15     [SerializeField] float moveSpeed = 10f;
16     bool isMoving = false;                      // FIX ALL THIS~~~~
17     bool climbing = false;
18     bool selected = false;
19
20     void Start()
21     {
22         levelStairs = GameObject.FindGameObjectsWithTag("Stair"); // Load
           levelStairs
23     }
24     void Update()
25     {
26
   // Checks if we are going up stairs or selected by player
27         if (Input.touchCount > 0 && climbing == false &&
           SelectedCharacter.isSelected == true)
28         {
29             Touch touch = SelectedCharacter.touch;
30             Vector3 touchPos = Camera.main.ScreenToWorldPoint(touch.position);
31             touchPos.z = 0;
32             if (touch.phase == TouchPhase.Began)
33             {
34                 int currentLevelNumber = FloorDetection.CurrentLevelNumber();
                   // Finds the floor our player is on.
35                 int targetLevelNumber = FloorDetection.TargetLevelNumber
                   (touch);// Finds the level our player wants to go to.
36
37                 int distance = currentLevelNumber - targetLevelNumber;
                   // Determines if we need to go up or down and how many stairs
38
   // we need to climb or descend.
39                 if (distance == 0 && selected)
                   // ~~~~~
40                 {
41                     if(isMoving == true)
```

```
42                          StopAllCoroutines();
43                      StartCoroutine(Move(touchPos, Vector3.zero));

44                  }
45                  if (distance < 0 &&
                      selected)                                   // ~~~~~
46                  {
47                      if (isMoving == true)
48                          StopAllCoroutines();
49                      StartCoroutine(AscendingLevels(levelStairs,
                         currentLevelNumber, targetLevelNumber, touchPos));

50                  }
51                  else if (distance > 0 &&
                      selected)                                 // ~~~~~
52                  {
53                      if (isMoving == true)
54                          StopAllCoroutines();
55                      StartCoroutine(DescendingLevels(levelStairs,
                         currentLevelNumber, targetLevelNumber, touchPos));

56                  }
57                  selected = !selected;
                      // FIX FIX ~~~~
58              }
59          }
60      }
61
62
63      /*
64       * Handles climbing stairs by looping through our levelStairs, we use
           coroutines here since we need to finish
65       * movement before executing the next loop. This is done through yield return
            null.
66       */
67      IEnumerator AscendingLevels(GameObject[] levelStairs, int currentLevelNumber,
           int targetLevelNumber, Vector3 touchPos)
68      {
69          for (int x = currentLevelNumber - 1; x < targetLevelNumber - 1; x++) //
              Determines how many times we need to loop through code.
70          {
71              GameObject stairStart = levelStairs[x * 2];                    //
                  Grabs the start of our stair(position).
72              GameObject stairEnd = levelStairs[(x * 2) + 1];                //
                  Grabs the end of our stair(position).
73
74                                                                             //
                          Convert to vectors for transforming
75              Vector3 stairStartPos = new Vector3(stairStart.transform.position.x,
                  stairStart.transform.position.y, 0f);
76              Vector3 stairEndPos = new Vector3(stairEnd.transform.position.x,
                  stairEnd.transform.position.y, 0f);
```

```
77
78                                                      // If we are moving
                        (determined through boolean affected in move coroutine
79              StartCoroutine(Move(stairStartPos, stairEndPos)); // we pause our
                  code using the while loop.
80              while (isMoving == true)                // These two
                  coroutines handle movement to our stair start,
81                  yield return null;                  // then the
                      movement to the top of the stairs.
82
83              StartCoroutine(Move(touchPos, Vector3.zero));
84              while (isMoving == true)
85                  yield return null;
86          }
87      }
88      /*
89       * Handles descending stairs by looping through our levelStairs, we use
           coroutines here since we need to finish
90       * movement before executing the next loop. This is done through yield return
            null.
91       */
92      IEnumerator DescendingLevels(GameObject[] levelStairs, int
          currentLevelNumber, int targetLevelNumber, Vector3 touchPos)
93      {
94          for (int x = currentLevelNumber - 1; x > targetLevelNumber - 1; x--) //
              Determines how many times we need to loop through code.
95          {
96              GameObject stairStart = levelStairs[(x * 2) - 1];               //
                  Grabs the start of our stair(position).
97              GameObject stairEnd = levelStairs[(x * 2) - 2];                 //
                  Grabs the end of our stair(position).
98
99                                                                              //
                        Convert to vectors for transforming.
100             Vector3 stairStartPos = new Vector3(stairStart.transform.position.x,
                  stairStart.transform.position.y, 0f);
101             Vector3 stairEndPos = new Vector3(stairEnd.transform.position.x,
                  stairEnd.transform.position.y, 0f);
102
103                                                     // If we are moving
                        (determined through boolean affected in move coroutine
104             StartCoroutine(Move(stairStartPos, stairEndPos)); // we pause our
                  code using the while loop.
105             while (isMoving == true)                // These two
                  coroutines handle movement to our stair start,
106                 yield return null;                  // then the
                      movement to the top of the stairs.
107
108             StartCoroutine(Move(touchPos, Vector3.zero));
109             while (isMoving == true)
110                 yield return null;
111         }
```

```
112     }
113     /*
114      * Moves our player depending on which vectors we are given
115      *
116      * Ex. If only one Vector3 is given(Pos1) we move once
117      *     If two Vector3s' are given(Pos1, Pos2) we move twice and access another coroutine(ClimbLevel)
118      */
119     IEnumerator Move(Vector3 Pos1, Vector3 Pos2)
120     {
121         SelectedCharacter.isSelected = false;   // If our player is issued a command they are no longer selected.
122
123         isMoving = true;                        // Used to inform our code if the player is currently issued a command.
124                                                 // This prevents buggy movement by allowing us to cancel that movement before issuing
125                                                 // another command.
126
127         while (Pos1.x != transform.position.x)  // Moves player to our position until we reach it. (Can result in a lot of soft locks if player cant reach position)
128                                                 // so keep an eye out here.****
129         {
130             transform.position = Vector2.MoveTowards(transform.position, new Vector2(Pos1.x, transform.position.y), moveSpeed * Time.deltaTime);
131             yield return null;
132         }
133         if (Pos2 != Vector3.zero)               // We determine if our player is still moving by checking if we have another position to move to.
134             StartCoroutine(ClimbLevel(Pos2));
135         else                                    // Otherwise we end our movement here.
136             isMoving = false;
137     }
138     IEnumerator ClimbLevel(Vector3 Pos2)
139     {
140         climbing = true;                        // Handles climbing boolean (active only when moving between floors.
141         while (Pos2.y != transform.position.y)
142         {
143             transform.position = Vector2.MoveTowards(transform.position, new Vector2(Pos2.x, Pos2.y), moveSpeed * Time.deltaTime);
144             yield return null;
145         }
146         climbing = false;
147         isMoving = false;
148     }
149
150
151 }
152
```