```
1  using System;
2  using UnityEngine;
3
4  /// <summary>
5  /// CameraMovementPreview handles all player movement within the preview scene.<br/><br/>
6  /// Some behaviors (e.g. scrolling) will be enabled or disabled based on the state of several other scripts.
7  /// </summary>
8  public class CameraMovementPreview : MonoBehaviour
9  {
10     // Singleton state reference
11     private static CameraMovementPreview instance;
12
13     /// <summary>
14     /// The primary camera utilized by the player.
15     /// </summary>
16     [SerializeField]
17     Camera playerCamera;
18
19     /// <summary>
20     /// The speed under which the player can move around scenes.
21     /// </summary>
22     [SerializeField]
23     float movementSpeed;
24
25     /// <summary>
26     /// The speed under which the player can scroll around scenes.
27     /// </summary>
28     [SerializeField]
29     float scrollSpeed;
30
31     /// <summary>
32     /// How low and high the player can vertically go.
33     /// </summary>
34     [SerializeField]
35     float minHeight, maxHeight;
36
37     /// <summary>
38     /// Moves the player up and down respectively.
39     /// </summary>
40     [SerializeField]
41     KeyCode upKey, downKey;
42
43     /// <summary>
44     /// Keeps track of the mouse position in the current frame.
45     /// </summary>
46     private Vector3 mousePosCurrent;
47
```

```
48      // Enforces a singleton state pattern and initializes camera values.
49      private void Awake()
50      {
51          if (instance != null)
52          {
53              Destroy(this);
54              throw new Exception("CameraMovementPreview instance already
                  established; terminating.");
55          }
56
57          instance = this;
58          ClampPos();
59      }
60
61      private void Start() { mousePosCurrent =
            BehaviorManagerPreview.Instance.UpdateCoordinates(); }
62
63      // Listens to key inputs and updates movements each frame.
64      private void Update()
65      {
66          float x, y, z;
67
68          Vector3 mousePosPrev = mousePosCurrent;
69
70          mousePosCurrent =
              BehaviorManagerPreview.Instance.UpdateCoordinates();
71
72          // X-Z movement via mouse drag
73          if (Input.GetMouseButton(0) && !
              BehaviorManagerPreview.Instance.IsUILocked)
74          {
75              Vector3 mousePosDelta = mousePosPrev - mousePosCurrent;
76
77              x = mousePosDelta.x;
78              z = mousePosDelta.z;
79          }
80
81          // X-Z movement via WASD
82          else
83          {
84              // Obtains x and z axis values based on input
85              x = Input.GetAxisRaw("Horizontal") * movementSpeed *
                    Time.deltaTime;
86              z = Input.GetAxisRaw("Vertical") * movementSpeed *
                    Time.deltaTime;
87          }
88
89          // Y movement via scroll wheel
90          if (Mathf.Abs(Input.mouseScrollDelta.y) > 0)
```

```
91          {
92              y = -Input.mouseScrollDelta.y * scrollSpeed * Time.deltaTime;
93          }
94
95          // Y movement via upKey and/or downKey
96          else
97          {
98              y = 0;
99
100             // Determines y axis values (holding both "upKey" and
                  "downKey" will negate one another)
101             if (Input.GetKey(upKey))
102             {
103                 y += movementSpeed * Time.deltaTime;
104             }
105
106             if (Input.GetKey(downKey))
107             {
108                 y -= movementSpeed * Time.deltaTime;
109             }
110         }
111
112         // Adds obtained values and updates position
113         transform.position += x * Vector3.right + y * -CameraRay.direction
                + z * Vector3.forward;
114         ClampPos();
115         mousePosCurrent =
              BehaviorManagerPreview.Instance.UpdateCoordinates();
116     }
117
118     /// <summary>
119     /// Clamps values to ensure the user cannot traverse out of bounds.
120     /// </summary>
121     private void ClampPos()
122     {
123         Vector3 pos = transform.position;
124
125         pos.y = Mathf.Clamp(pos.y, minHeight, maxHeight);
126         transform.position = pos;
127     }
128
129     // Getter methods
130     public static CameraMovementPreview Instance { get { return
          instance; } }
131
132     public Camera PlayerCamera { get { return playerCamera; } }
133
134     private Ray CameraRay { get { return playerCamera.ScreenPointToRay
          (Input.mousePosition); } }
```

```
135 }
```