```csharp
using System;
using UnityEngine;

/// <summary>
/// CameraMovement handles all player movement within the editor
  scene.<br/><br/>
/// Some behaviors (e.g. scrolling) will be enabled or disabled based on
  the state of several other scripts.
/// </summary>
public class CameraMovement : MonoBehaviour
{
    // Singleton state reference
    private static CameraMovement instance;

    /// <summary>
    /// The primary camera utilized by the player.
    /// </summary>
    [SerializeField]
    Camera playerCamera;

    /// <summary>
    /// The speed under which the player can move around scenes.
    /// </summary>
    [SerializeField]
    float movementSpeed;

    /// <summary>
    /// The speed under which the player can scroll around scenes.
    /// </summary>
    [SerializeField]
    float scrollSpeed;

    /// <summary>
    /// How low and high the player can vertically go.
    /// </summary>
    [SerializeField]
    float minHeight, maxHeight;

    /// <summary>
    /// Moves the player up and down respectively.
    /// </summary>
    [SerializeField]
    KeyCode upKey, downKey;

    /// <summary>
    /// Keeps track of the mouse position in the current frame.
    /// </summary>
    private Vector3 mousePosCurrent;

    // Enforces a singleton state pattern and initializes camera values.
    private void Awake()
    {
        if (instance != null)
        {
            Destroy(this);
            throw new Exception("CameraMovement instance already established; terminating.");
        }

        instance = this;
        ClampPos();
    }

    private void Start() { mousePosCurrent = Coordinates.Instance.MousePos; }

    // Listens to key inputs and updates movements each frame.
    private void Update()
    {
        float x, y, z;

        Vector3 mousePosPrev = mousePosCurrent;

        mousePosCurrent = Coordinates.Instance.MousePos;

        // If the scene is paused or there is an override, no movement can occur.
        if (BehaviorManager.Instance.CurrentStateType == BehaviorManager.StateType.PAUSED && !IOAssigner.Instance.MovementOverride) return;

        // Otherwise, some/all movement features are allowed depending on whether the game is unrestricted or locked.
        // X-Z movement via mouse drag
        if (Input.GetMouseButton(0) && BehaviorManager.Instance.CurrentStateType == BehaviorManager.StateType.UNRESTRICTED && BehaviorManager.Instance.CurrentGameState != BehaviorManager.GameState.CIRCUIT_HOVER)
        {
            Vector3 mousePosDelta = mousePosPrev - mousePosCurrent;

            x = mousePosDelta.x;
            z = mousePosDelta.z;
        }

        // X-Z movement via WASD
        else
```

```csharp
        {
            // Obtains x and z axis values based on input
            x = Input.GetAxisRaw("Horizontal") * movementSpeed * Time.deltaTime;
            z = Input.GetAxisRaw("Vertical") * movementSpeed * Time.deltaTime;
        }

        // Y movement via scroll wheel
        if (Mathf.Abs(Input.mouseScrollDelta.y) > 0)
        {
            y = -Input.mouseScrollDelta.y * scrollSpeed * Time.deltaTime;
        }

        // Y movement via upKey and/or downKey
        else
        {
            y = 0;

            // Determines y axis values (holding both "upKey" and "downKey" will negate one another)
            if (Input.GetKey(upKey))
            {
                y += movementSpeed * Time.deltaTime;
            }

            if (Input.GetKey(downKey))
            {
                y -= movementSpeed * Time.deltaTime;
            }
        }

        // Adds obtained values and updates position
        transform.position += x * Vector3.right + y * -CameraRay.direction + z * Vector3.forward;
        ClampPos();
        mousePosCurrent = Coordinates.Instance.MousePos;
    }

    /// <summary>
    /// Clamps values to ensure the user cannot traverse out of bounds.
    /// </summary>
    private void ClampPos()
    {
        Vector3 pos = transform.position;

        pos.y = Mathf.Clamp(pos.y, minHeight, maxHeight);
        transform.position = pos;
    }

    // Getter methods
    public static CameraMovement Instance { get { return instance; } }

    public Camera PlayerCamera { get { return playerCamera; } }

    private Ray CameraRay { get { return playerCamera.ScreenPointToRay
      (Input.mousePosition); } }
}
```