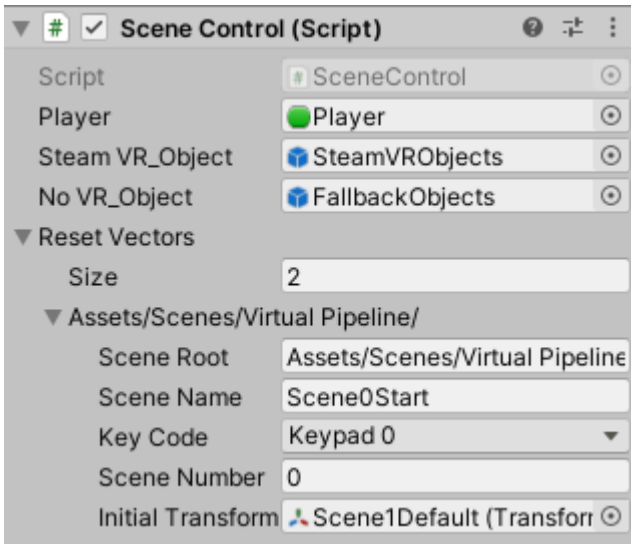
Scene Control Script

Description:

The Scene Control script allows for moving the player to different sub scenes within the main scene. When activating the different scenes, the player is transported to a point within that scene, and the scene is initialized to a beginning state so that the interactions can play out from the beginning.

Interface:



The Scene Control script has several parameters that are made public to the developer to provide a simple interface to create subscene transitions. This script was designed to work with the Steam VR player character system. As such, you will find that the scene control must reference key aspects of this prefab. Firstly, it must reference the base 'Player' frame. This is used to reset the player to the desired position within the subscene. It must also make references to 'SteamVRObjects' and 'FallbackObjects' that are contained within the 'Player' game object. These are used to further set up the player position on a scene reset, whether VR is present or not.

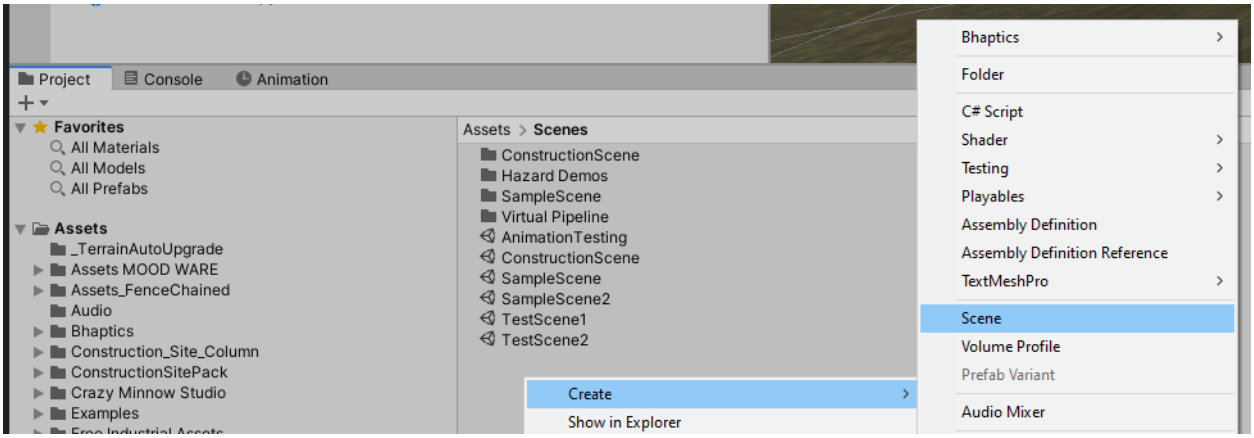
Following these references, an array of 'Reset Vectors' is exposed to the developer. Each of the elements in this array contains key information about a subscene to include its root directory, its name, the keyboard key to press to actuate the reset for it, a numeric value to reference for the animation system, and a position transform that is used for resetting the player to a desired position within the scene. The following is a look at the structure in the code:

```
//ResetVector: holds data needed to reset a given sub-scene
[System.Serializable]
4 references
public class ResetVector
{
    [Tooltip("The root path of the scene.")]
    public string sceneRoot;
    [Tooltip("Name of the scene to reset to (without '.unity' suffix).")]
    public string sceneName;
    [Tooltip("The keyboard key that will reset this scene.")]
    public KeyCode keyCode;
    [Tooltip("The number you wish to identify this scene with.")]
    public int sceneNumber;
    [Tooltip("GameObject transform used as an initial position for the player in the scene.")]
    public Transform InitialTransform;
}
```

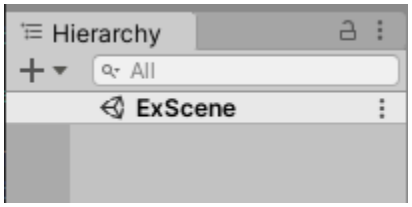
To set up a new interface start by creating an empty object and adding the script to it. Then place references for the Player, SteamVRObjects, and FallbackObjects within named regions of the script interface. Next, change the size of the 'Reset Vectors' array to the number of subscenes that you want to track with the system. This can be changed at any time to add more scenes. Once you have empty vectors, fill in their parameters accordingly.

Scene Creation:

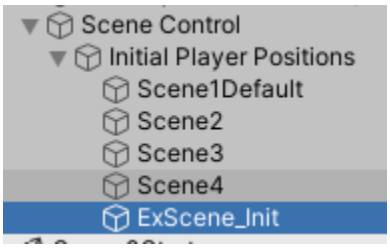
To create a new scene simply go to the Project overview tab in the Unity Editor. It is recommended to place them within the Assets/Scenes/ hierarchy. Once there, right click within the desired directory and select Create->Scene.



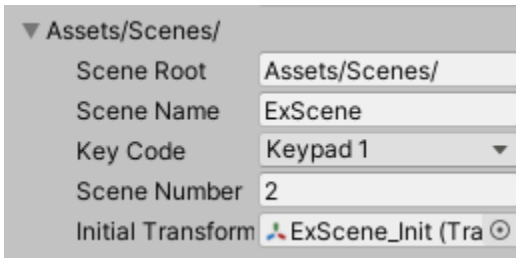
Open the scene for editing. Delete the camera and directional light. Save the scene. The scene hierarchy should look like this:



Load the main scene and open the hierarchy for the 'Scene Control' object. Then open the 'Initial Player Positions' object. Within this object create a new object. Its transform will be used as the reset transform for your new scene:



Next, in the main scene, go to the Scene Control object and increment the size of the ‘Reset Vectors’ array. This will generate a new ‘Reset Vector’. Now fill in the appropriate data for the scene. Save the main scene.



Now open a different scene, and then open the main scene again. The newly created scene will show up in the hierarchy. You have now finished the setup for the new scene.

Code:

1. **Start():** Begins by calling a separate function called ‘Setup’. It then loads in each of the subscenes described within the ‘ResetVectors’ array.
   a.
```
    //START: loads the sub-scenes when the main scene is loaded
    Unity Message | 0 references
    void Start()
    {
        Setup();

        foreach (ResetVector resetVector in ResetVectors)
        {
            Debug.Log("Loading scene: " + resetVector.sceneName);
#if UNITY_EDITOR
            EditorSceneManager.OpenScene(resetVector.sceneRoot + resetVector.sceneName + ".unity", OpenSceneMode.Additive);
#else
            SceneManager.LoadScene(resetVector.scene, LoadSceneMode.Additive);
#endif
        }

    }
```
2. **Setup():** Finds the active frame of reference for the Steam VR system. This is used in the rest of the code for resetting the player’s position. This also records the initial position for the player.

a.

```
//Setup: performs initialization of the activeFrame and initial positioning data on game start
1 reference
void Setup()
{
    VRLoader = GetComponent<SteamVR_LoadLevel>();
    if (SteamVR_Object.activeInHierarchy)
    {
        Debug.Log("SceneReset.Setup(): activeFrame set to SteamVR_Object");
        activeFrame = SteamVR_Object;
    }
    else
    {
        Debug.Log("SceneReset.Setup(): activeFrame set to NoVR_Object");
        activeFrame = NoVR_Object;
    }
    initPos = activeFrame.transform.localPosition;
    initAngles = activeFrame.transform.localEulerAngles;
    Debug.Log("SceneReset.Setup(): initial position: " + initPos.ToString());
    Debug.Log("SceneReset.Setup(): initial euler angles: " + initAngles.ToString());
}
```

3. **Update():** Checks whether one of the keys in the reset vectors has been pressed, if it has it initiates a reset to that subscene.

a.

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    foreach (ResetVector resetVector in ResetVectors)
    {
        if (Input.GetKeyDown(resetVector.keyCode))
        {
            SceneControlVars.currentScene = resetVector.sceneNumber;

            StartCoroutine(ResetScene(resetVector));
            break;
        }
    }
}
```

4. **ResetScene():** Carries out the scene reset. Will not reset to a scene if another reset is in progress. This reloads the indicated scene, and sets up the player's position to the indicated reset transform.

a.

```
Debug.Log("SceneReset.ResetScene(): loading scene" + resetVector.sceneName);
VRLoader.levelName = resetVector.sceneName;
VRLoader.Trigger();


while (SteamVR_LoadLevel.fading)
    yield return null;

SceneControlVars.currentScene = resetVector.sceneNumber;
Player.transform.position = resetVector.InitialTransform.position;
Player.transform.eulerAngles = resetVector.InitialTransform.eulerAngles;
activeFrame.transform.localPosition = initPos;
activeFrame.transform.localEulerAngles = initAngles;

while (SteamVR_LoadLevel.loading)
    yield return null;

Debug.Log("Finished.");
```