MicroVerse & SAM General Setup (v1 – March 13, 2024)

To make things easier, we will be utilizing the MicroVerse Demo scene, which most of you should have access to. This demo is called 2x2 Demo and can be found in the Packages/MicroVerse-Demo/Scenes/2x2 Demo folder. If you already have a MicroVerse scene (with MicroVerse hierarchy), you are free to use that scene instead, skipping any steps that do not apply.

We will also operate under the assumption that this demo scene is using a Fixed (non-floating origin) World. Do not worry if you plan on your own game world having a floating origin; with the instructions below you will be able to use either type of World.

1) Before proceeding, re-save the 2x2 demo scene under a new name and location. From here on out, we will use the new scene rather than the 2x2 scene so that the original demo scene is preserved. I will refer to this new scene as the Demo scene.

2) Once the new scene is saved, add it to your Build Settings by clicking **File -> Build Settings**, and then using the **Add Open Scenes** button.

   When you build your project, the scene with index 0 will be the first scene that Unity runs, so if you plan on making a test build with this demo scene later, you should make sure it is the first scene (index 0).

3) Before we start setting up SAM, we need to make one modification to MicroVerse settings. Click on the MicroVerse game object and then find the **Terrain Search Method** setting (it can be found under **Options/Settings**). Change the method to **All In Scene**.

   Changing this setting will ensure that the Terrain objects can be found by MicroVerse later on, after we have converted them to Asset Chunks.

4) Now we will add the SAM default objects and components to the scene using the **GameObject -> Create Other -> Deep Space Labs -> SAM -> Default SAM Setup** menu command.

5) If you are not using the Demo scene, and you are currently utilizing a single Unity Terrain for your game World, do note that MicroVerse is configured to work with

multiple terrains. As such, if you have access to **Terra Slicer**, you should feel free to slice you terrain into multiple terrains!

6) In the Demo scene, the scene already contains a 4x4 group of terrains for a total of 16 terrains, so using Terra Slicer is not necessary.

   However, these terrains are not in the correct format required by SAM. We will fix that later. For now, let's continue with setting up the demo scene.

7) We will start with the World component. The first thing we want to do is configure our Base Grouping, which will represent the terrains in the scene.

   Navigate to the **Groupings** tab, which will have the **1 Base Grouping** tab automatically selected (since it is the only Grouping). You should also be in this Grouping's **General Settings** sub-tab. Change the **World Grouping Name** to something that will help you relate it to the terrain (**Terrains** is as good of a name as any).

8) Create a Streamable Grid asset to represent the terrains. To do so, right click the folder where you want to create this asset and choose **Create -> Deep Space Labs -> SAM -> Streamable Grid**.

   Rename the asset to help differentiate it from other Streamable Grid assets.

9) The Demo scene terrain group is a 4x4 terrain group, which happens to be the same number of **Rows** and **Columns** that the default Streamable Grid asset starts with, so there is no need to change those values.

   You will, however, will likely wish to change the **Axes** type to **Two Dimensional Using XZ Axes**, although if you want the Player's Y Axis position to effect whether terrain Cells are loaded or unloaded, you can also choose to keep the type set to **Three Dimensional**.

10) Later on, we will be using the World Designer Tool to assign the terrain to this Streamable Grid's Cells, so it's important to disable all Cells to begin with. To do this, hold the shift key and right click the **Everything** button.

11) We need to adjust the Cell dimensions to match the Terrain size. The terrains are 1000 x 1000, so that's what all of the Cell's should be as well. Left click the **Everything** button (without shift held down), and enter 1000 into the popup box that appears. Press the

Apply Changes button or enter on your keyboard to commit the change.

This will set all Row Lengths and Column Widths to **1000**, which is what we want, however it will also set the Layer 1 (L1) height to 1000.

While this is not especially problematic, it is usually a good idea to set the Layer height equal to the Terrain's height value, which in this case is 600. Press the button that says **L1 (1000)** and enter **600** in the popup box, then press **Apply Changes**.

12) Now navigate to the **LOD Groups** tab and modify the **Group Name** field to something that will help you identify these terrain Cells at runtime. You are free to use whatever name you wish!

13) Additionally, change the **Chunk Type** to **Unity Terrain**, which will ensure the Terrains are neighbored properly at runtime. After that, there shouldn't be a need to adjust any more settings on this Streamable Grid.

14) Navigate back to the World component in the Demo scene and assign the newly created Streamable Grid to Grouping 1's **Streamable Grid** field (again, this is found in the Grouping's **General Settings** sub-tab).

15) Now create another new World Grouping, which will contain just a single World Cell in order to house the MicroVerse hierarchy. Use the **Add World Grouping** button to do so.

16) Repeat step 8 to create a new Streamable Grid asset for this grouping, then adjust the following settings:

    a) Set **Axes** to the same value you set the first Streamable Grid's Axes to.

    b) Set **Rows** and **Columns** to 1, then press one of the **Update** buttons (it doesn't matter which, since no new Cells are being added).

    c) Click the R1 (500) button and set the value to 4000, then click the C1 (500) button and also set it to 4000. This cell is intended to be the same size as the entire World, which in this case is 4000 x 4000 (4 x 4 terrain group which are all 1000 x 1000).

Click the L1 (500) button and set it to 600, the same height as the Terrain.

d) Disable the single Cell by right click on it (so it turns red).

e) In the **LOD Groups** tab, give the LOD a distinct **Group Name** like MV_Hierarchy. No other settings need to be adjusted.

17) Assign the new Streamable Grid asset to the **Streamable Grid** field of World Grouping 2.

18) Last but not least, navigate to the **Origin Settings** tab. The terrain group of the Demo Scene begins at -2000, -2000, so the current Origin Settings would cause the terrain to be shifted at runtime.

    There are several ways we can modify the Origin settings to correct this, however the simplest solution is to simply modify the **Origin Row** and **Column** to be equal to the Cell who's terrain is currently positioned at x = 0, z = 0.

    This, as it turns out, is the terrain named **Terrain**, which is the 3rd terrain from the bottom/left (so Cell Row = 3, Column = 3). Set both **Row** and **Column** to **3**.

19) Now navigate to the Active Grid game object where the **ActiveGrid** component resides.

    The first thing we want to do is to assign the Player to the Active Grid, which the Active Grid will track in order to decide which Cells it needs loaded.

    In the Demo scene, the **Player** game object does not actually change as the player moves, so instead of assigning that object we will assign the **Challenger** object to the **Player Transform** field (note, if using the Roads Demo scene, the object is called **SkyCar**).

20) Next we want to assign a Blueprint to Grouping 1, however to do so we will need to assign a **Loading Blueprint Repository**.

    If you already have one of these created, feel free to use it, however if you don't, or just want to use a different one for the demo scene, you can do so by right clicking a folder in your Project and selecting **Create -> Deep Space Labs -> SAM -> Loading Blueprint Repository** (rename it after creating it).

Assign the repository to the **Loading Blueprint Repository** field on the **ActiveGrid**.

Next, select the Repository asset and click on the **Edit Repository** button in the inspector, which will open the Loading Blueprint Editor.

Click the **Create Loading Blueprint** button and leave the Blueprint as its default 3x3 Uniform Ring (you can come back and change this later if you wish).

Close the Loading Blueprint Editor and navigate back to the Active Grid component. If you click the **Groupings** tab, you will notice that Grouping 1's **Loading Blueprint** field has automatically been set to the new Loading Blueprint you created (if you are using a pre-existing Repository, make sure to click the dropdown and set the Blueprint to whichever one you wish to use).

21) Adjust the **Grouping Name** field to help relate it to the terrains from World Grouping 1.

22) Before leaving the **Groupings** tab, we are going to add an additional Grouping, which will represent the MicroVerse hierarchy (i.e., the World Grouping 2 created in step 15).

    To do so, click the **Add Grouping** button, then give it a new **Grouping Name** (MV Hierarchy, for instance).

    This Grouping should be fine to use the same 3x3 Uniform Ring Loading Blueprint, however if you are not planning on ever using an Endless World, you can also create a new Uniform Ring Blueprint with a single Inner Area Cell to use with this Grouping (this should make more sense later).

23) At this point, we recommend creating a Prefab out of the Streamable Assets Manager object by dragging it from the Scene Hierarchy into a Project folder.

    Doing this will allow you to re-use this prefab if something goes wrong later on, as you'll be able to reopen the 2x2 Demo scene, resave it under a new name, and then drag the prefab into it (note, if you do need to do this, you will need to drag the **Challenger** player object onto the **Player Transform** field of the Active Grid component, and also re-set the **Terrain Search Method** on the MicroVerse script to **All In Scene**).

24) Our scene is now fully setup, so we can proceed with creating the Asset Chunks that will hold the Terrains and MicroVerse Hierarchy. Navigate back to the World component and click on the **World Designer** tab.

25) You will see a red warning message telling you that "World Designer Data has not been assigned." To fix this, press the **Create New Data** button. You can also use the **Bind Data To World** button.

26) The scene always needs to be in a non-dirty state before launching the World Designer tool, so press the **Save Scene** button now. Once you do so, you should see the **Launch World Designer** button, which you can use to open the tool.

27) The tool will be opened with Grouping 1 as the **Grouping To Edit**, so the red Cells you see in the middle of the window represent the terrain cells (which we have not created Asset Chunks for yet). You can minimize the **View Controls** area by clicking on the drop down arrow, as we will not need to make use of these Controls.

28) Click on Cell **R1 | C1** (the bottom left most cell). Now observe the Scene view window.

    You'll notice that a white outline appears in this view, which represents the area that selected/highlighted Cells occupy in the game world. Unfortunately, this outline is appearing over the wrong terrain, even though we changed our World's Origin Cell to Row 3, Column 3.

    Don't worry, this is actually by design! The World Designer Tool does not use the World's Origin Cell (though it does use its Origin Position), but rather its own unique value that is stored with the World Designer Data asset.

    Therefore, to fix the issue, navigate to the **Global Settings** tab and adjust the **Origin Row** and **Origin Column** values to 3 and 3 using the slider. To commit the changes, press the **Update Origin Cell/Position*** button.

    After doing so, you will notice the Scene View outline now appears over the first terrain in the group.

29) We are almost ready to start assigning the terrain and MicroVerse hierarchy to Asset Chunks. Before doing so, however, we need to create 1 or 2 Default Asset Creators and assign them to each Grouping's **Asset Creator** field. These Creators are responsible for

creating the Asset Chunks that the terrain and hierarchy will be assigned to.

30) To create the first Default Asset Creator, right click a folder in your project and select **Create -> Deep Space Labs -> SAM -> Default Asset Creator**.
Rename this asset and then click it.

31) This will be used to create the Asset Chunks for the terrain cells and possibly the MicroVerse hierarchy cell.

Since our Demo scene is configured to use Scene based Asset Chunks, you should set the **Asset Creation Mode** to either **Prefabs_And_Scenes** or **Only_Scenes**. If you use **Prefabs_And_Scenes**, set the **Primary Assets** setting to **Scenes**.

Configure the rest of the setting as you wish, being sure to leave the **Add To Build Settings** option enabled.

32) For the MicroVerse Hierarchy Asset Chunk, we highly recommend only creating scene Asset Chunks, as using Prefabs causes some issues due to how Micro Verse functions.

If in step 30 you used the **Only_Scenes** Asset Creation Mode, you can use the same Asset Creator created in step 30, however note that this will cause the terrain related scene Asset Chunks and Hierarchy related Asset Chunk to be generated in the same folder. This does not present any issues for SAM, it is only a matter of whether you like things to be organized in a certain way.

If you did not set **Asset Creation Mode** to **Only_Scenes** or want to use a different scene output folder, feel free to create a new Default Asset Creator (being sure to set **Asset Creation Mode** to **Only_Scenes**).

33) Heading back to the World Designer Tool, assign the Asset Creator(s) you created to the **Asset Creator** fields for both Grouping 1 and Grouping 2 (for Grouping 2, you will first need to select Grouping 2 from the **Grouping To Edit** dropdown).

34) When assigning Terrain, it can be useful to add a slight offset to each terrain's position when calculating which Cell it should be assigned to, since the Terrain's raw position typically falls along the border of Cells (which means floating point inaccuracies can cause the terrain to be assigned to the wrong Cells).

Navigate to the **General Settings** tab (in the top middle **Advanced Operation** Settings area) and look for the **Assignment/Transfer Position Offset** setting. Set the X, Y, and Z values all to 1, making sure to hit Enter after each change.

35) Now navigate to the **Assignment Settings** tab. You can use any number of **What To Assign** options to accomplish the terrain assignment, however for simplicities sake we recommend leaving it set to **Selected Objects**.

36) Make sure **Grouping To Edit** is set to Grouping 1, then the first terrain in the group (the one named **Terrain**).

37) Press the Assignment button (**A*)**, which can be found to the right of **Loaded LOD 1** in the **Cell Color Key** section of the **Main Controls** area.

    The assignment operation will complete very quickly. Afterwards, you should see the **R3 | C3** Cell turn blue, indicating that it is now enabled and has Asset Chunks (or in this case, just a single Asset Chunk).

    Click on this cell and press the Load button (**L***). The former Terrain game object should be loaded into the scene (either as a scene or as a prefab depending on the Asset Creator you created in step 30). This game object will now be the child of another game object, which is the root Asset Chunk.

38) If something went wrong and the terrain was not assigned to the correct Cell, take a look at the website FAQ to try and diagnose why.

    If the terrain was assigned correctly, proceed by selecting all of the remaining terrains and pressing the assignment button (**A***) again.

39) Afterwards, all of the Cells should turn blue, and if you highlight them all and press the **L*** button, all of the terrain should be loaded into the editor (do note, in some cases when creating both prefabs and scenes, the World Designer Tool may load the prefab form of the asset rather than the scene. This is by design and nothing to be alarmed by! At runtime the scenes will be loaded rather than the prefabs directly).

40) Once the assignment operation is complete and all terrains have been assigned to the correct Cells, you can delete the **Terrains** game object that is a child of the MicroVerse

game object, as it will no longer be needed.

41) You can also reset the **Assignment/Transfer Position Offset** settings back to 0,0,0.

42) Now we will create the Asset Chunk to store the MicroVerse hierarchy. While we could leave the hierarchy in the main scene, placing it inside of an Asset Chunk offers two advantages:

    a) It allows the MicroVerse hierarchy to be repositioned whenever the Origin Cell is changed (both in the editor and at runtime), which is crucial to ensuring MicroVerse works with floating origin based Worlds.

    b) It allows you to unload the MicroVerse hierarchy if you want to temporarily disable its editing capabilities.

43) Before doing this, however, drag the Probes game object onto the MicroVerse game object so it becomes a child of it. This will ensure that the probes stay in the same scene as the Water Plane, although to be honest I do not know if this will actually still allow the Reflection Probes to function properly.

    Save the Demo scene before proceeding (by right clicking the scene in the Scene Hierarchy and clicking **Save Scene)**.

44) Change the **Grouping To Edit** to Grouping 2. Select the MicroVerse game object and use the **A*** button to assign it. The single **R1 | C1** Cell should turn blue after it completes.

45) Select the Cell and use the **L*** button to load it, in order to ensure that the hierarchy loads correctly into the editor. Remember that the MicroVerse object will not be the root game object in the scene, but will instead be a child of another empty game object (which is the root Asset Chunk).

Additional Notes

No testing has been done on the **Challenger** character to determine if special code is required to shift or move the Player during Origin Cell changes or transportation method calls, although we see it as likely.

If you notice the player falling through the world during these operations, you can try to add

the **Fixed Update Player Mover** script to the Challenger game object, and then assign the Challenger game object to the **Player Mover** field on the Active Grid component.

If that does not work, you will need to modify either the **Third Person User Controller** or **Third Person Character** script to derive from **PlayerMover**, and then implement the **MovePlayerByAmount** and **MovePlayerToPosition** methods to function correctly with the script's movement code.