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

There are two different techniques for handling MicroVerse Roads in S.A.M.

The first is intended to be used during active development before your road system and world building is finalized, while the second is mainly designed to be used after you have finalized your road/world, although it can also be used during development for performance testing.

Both techniques have been designed to work with Endless Worlds.

With either technique, it's very important that you have read the previous section on **General Setup** before proceeding, as some steps will assume you have done so.

## Pre-Setup

The RoadEditor.cs file (found in the Packages/MicroVerse-Roads/Scripts/Editor folder) contains some asset/scene processing code that is intended to enable colliders for meshes related to the MicroVerse Road System. While this is normally not a problem, for S.A.M. it will cause significant performance problems in Play Mode and possibly slow down the World Designer Tool.

This code is also not needed when using either of the Techniques described above, so while I do not normally recommend modifying 3rd party code, I will in this case. You can comment out both the RoadAssetProcessor and RoadBuildProcessing classes at this time! Just remember to uncomment these classes in the future, if you remove S.A.M. or otherwise decide to stop using the techniques described below.

**Also, if you do not currently have a scene with a Road System, you can perform the rest of these instructions using the Roads Demo scene (found in Packages/MicroVerse-Roads-Demo/Scenes). The instructions below assume that your scene is already configured to work with SAM. If that is not the case, use the General Setup instructions above to configure this scene to work with SAM, before proceeding with the remaining steps.**

# Technique 1 – Culling System

## About

The first technique utilizes the Bounds Based Road Culling System, which as the name implies serves as a rudimentary culling system. This system constructs a set of composite Bounds for each Road/Intersection at runtime by examining all MeshRenderer children of each Road/Intersection. Once each Bounds is computed, each Road/Intersection is activated/deactivated as its Bounds falls into and out of a specific range from the Player.

The intention of this system is to allow you to preserve each of your Road Systems completely, in order to make editing the Road System work exactly like if you were not using SAM, while also granting a slight performance boost by deactivating those segments of the Road System that are far away from the player.

The drawback of this system is twofold. First, when the Road System is loaded for the first time, all procedural meshes and material overrides need to be generated/set, which in some cases may have an impact on performance. Second, because the system generates all procedural meshes from the get go, in terms of memory it will be like you are not even using S.A.M.

For this reason, while you can use this system in a live game, it is not recommended.

## Configuring The Culling System

In order to use this system, you must add the Bounds Based Road Culling System to the scene (**Add Component -> Deep Space Labs -> SAM -> 3rd Party -> MicroVerse -> Bounds Based Road Culling System**). Note that each Culling System is intended to be used with a single RoadSystem, so if you are using more than one Road System, you will need to add a unique Bounds Based Road Culling System for each one! Set up each Culling System as follows:

1) Copy the name of the game object with the RoadSystem script exactly, and paste it into the **Road System Name** field.

2) Decide whether you want to track the same Player used by an Active Grid, or a Transform based Player (enable/disable **Track Active Grid Player** depending on what

you choose). Assign the Active Grid or Transform to track.

If the Active Grid or Transform is not created until runtime, you can also call the **TrackActiveGrid** or **TrackPlayerTransform** method to assign these references at runtime!

3) Set the **Active Range**. Once the composite Bounds for a Road/Intersection falls out of this range, that Road/Intersection will be deactivated and only re-activated if it re-enters that range.

4) **Frequency** defines how often (in seconds) the Road/Intersection composite Bounds are checked to see if they fall in/out of the Active Range. For faster players you will likely need to use a smaller frequency; however this will result in more checking and more activating/deactivated of game objects, and thus may negatively impact performance.

5) **Max State Changes Per Frame** is intended to limit the number of game object activations/deactivations that are performed in a single frame, allowing you some control over the performance impact of these operations.

6) Once the System is configured, you need to add it as a World Grouping Listener to the World Grouping that contains your MicroVerse Asset Chunk (the one with the MicroVerse hierarchy).

   You can find the Listeners under the **Optional Components** tab of the World Grouping.

# Technique 2 – Road Asset Chunk Manager

## About

This technique leverages MicroVerse's ability to generate its procedural road and intersection meshes in the editor so that they can be saved with scene data.

It basically works the same way that converting a normal scene to Asset Chunks works in S.A.M., except in this case we generate the procedural meshes so that all RoadSystem related meshes can be converted to Asset Chunks.

These Asset Chunks can then be used with S.A.M. like normal, with the addition of a special World Grouping Listener that takes care of some configuration work needed to make the roads work correctly.

In order to use this technique, please follow the steps outlined in **Creating The Road Asset Chunks** and **Configuring The Road Asset Chunk Manager** below.

### Creating the Road Asset Chunks

1) Before proceeding, it is recommended to duplicate the Asset Chunk Scene that contains your MicroVerse Hierarchy, just in case something goes wrong during this process.

2) Create a new Streamable Grid that will represent the road meshes. The dimensions of this Grid are up to you, however we generally recommend using smaller sized cells (125 x 125 for instance). Just remember that the total width, length, and (for 3D grids) height of the Grid must equal the total width, length, and (for 3D Grids) height of all other Streamable Grids being used by World Groupings on the same World. In addition:

    - In most cases it makes the most sense to use Two Dimensional Using XZ Axes for the Axes Type, however if your terrain vary significantly across the Y Axes, using a Three Dimensional Axes Type can also make sense.

    - Only one LOD Group is needed, with its Chunk Type set to **Non Terrain Game Object**.

    - The Group Name can be set to whatever you'd like, thought generally we recommend it to be related to the name of the game object that contains your

Road System.

- Finally, using Multi-Chunking is recommended, since each Cell can easily end up with 30+ child objects.

3) Add a new World Grouping to the World component, then assign the Streamable Grid you created in Step 2 to this Grouping's **Streamable Grid** field (found in the **General Settings** tab of the Grouping).

4) Also add a new Grouping to the Active Grid component, which is the counterpart to the World Grouping. Use your discretion to choose an appropriate Loading Blueprint to use with this new Grouping.

5) The road Asset Chunks we will be creating further on must be scene based rather than prefab based, in order for the procedural mesh data to be saved correctly. As such, you may need to create a new Default Asset Creator (which is set to only output scenes) and Scene Chunk Streamer if no such assets/components exist already (although if you have followed the General Setup instructions, you should have such an Asset Creator and Scene Chunk Streamer already present).

   If the **Default Chunk Streamer** found in the World's **Overridable Settings** tab is not set to this Scene Chunk Streamer, you will need to assign the Streamer to the new Grouping by navigating to that Grouping, then its **Grouping Overrides** tab. Assign the Streamer to the **Chunk Streamer** field.

6) Open the World Designer Tool (saving your main scene first) and load the Asset Chunk containing your MicroVerse Hierarchy (there should only be a single Cell on this Grouping, so the Cell should be 1_1 or 1_1_1).

7) If you have made changes to your road that would affect one or more terrains, and have not loaded those terrains in order that those changes are applied, load those terrains now using the World Designer Tool by navigating to the World Grouping that contains the terrain and using the **L*** button.

8) Navigate to the RoadSystem you are going to be creating Asset Chunks for and disable both **Hide Game Objects** and **Generate At Load**.

9) Press the **Update System** button on the RoadSystem script.

10) If you loaded one or more terrains in step 7, unload those terrains now before proceeding. If you do not follow this step, when the road meshes are converted to Asset Chunks and removed from the MicroVerse hierarchy, the terrains will be modified in a way that is not desirable.

11) The World Designer Tool should still be open, but if it is not, open it now. Change the **Grouping To Edit** (in the **Editing Controls** tab) to the Grouping you created in step 3.

12) Assign the Default Asset Creator you wish to use to create Asset Chunks for this Grouping. **Make sure this Asset Creator is configured to only output scenes!**

13) If you followed our recommendation and enabled Multi-Chunking for the Streamable Grid created in step 2, set the **Max Children/Chunk** value to whatever you wish (you can start with a higher value like 20 or 30 first and see if it presents any performance issues).

14) Enable **Try Use Mesh Bounds**, which can be found within the **General Settings** tab of the **Advanced Operation Settings** window of the World Designer Tool (this step is crucial to ensure the meshes are assigned to the correct Cells!).

15) Make sure the **Assignment/Transfer Position Offset** value is set to 0,0,0.

16) In the **Assignment Settings** tab of the **Advanced Operation Settings** window, set **What To Assign** to **Game Objects With Component**, then enter **UnityEngine.MeshRenderer** into the Component Name field.

17) Drag and drop the game object containing your Road System onto the **Assignment Parent** field, which will ensure that only meshes that are descendants of that Road System will be assigned.

18) Press the **Test Hierarchy Search** button and look in the Console Window. You should see a message detailing how many objects were identified. These are all of the game objects with MeshRenderers that are children of the Road System game object. If the number of game objects found is less than you think it should be, repeat steps 7-17 (except steps 12-13) and press the **Test Hierarchy Search** button again.

19) Once you are sure everything is correct, press the **A*** button to begin the Assignment Operation. This operation will likely 30+ seconds to complete, depending on how large your Road System is.

20) After the Assignment Operation is complete, navigate to the World Designer Tool's **Global Settings** tab and set **Unsaved Asset Handling Method** to **Unload Without Saving**.

21) Within the **Editing Controls** tab, change the **Grouping To Edit** back to the Grouping containing your MicroVerse Hierarchy. Unload the 1_1 (or 1_1_1_) cell using the **U*** button.

    This step is required because the Road System generally contains some non-procedural meshes (like parking spaces) that are removed during the Asset Chunk conversion process. If were to save the MicroVerse Hierarchy after removing those objects, they would be lost for good in that scene (though they should still exists in the backup scene created in step 1).

    If done correctly, this step will ensure that you do not need to use the backup scene, as the original scene containing the MicroVerse hierarchy should be exactly as it was before starting this process, with all non-procedural road system related meshes intact.

    This allows you to make further edits to the Road System if needed, without having to resort to the backup scene.

22) Reload the Cell containing the MicroVerse hierarchy and confirm that the non-procedural meshes are still present (parking spaces are a good place to look). If they are not, you should delete this scene and rename your backup scene to have the Cell 1_1 (or 1_1_1) Asset Chunk name.

23) Reset **Unsaved Asset Handling Method** to either **Display Prompt** or **Auto Save**.

24) Check to make sure that the Asset Chunks for the roads/intersections were created correctly by loading all of the Cells on the new Grouping you created in step 3.

    Do not worry that some of the meshes are pink. This is due to those meshes using RoadMaterialOverride scripts, which will be configured correctly at runtime after following the steps in **Configuring The Road Asset Chunk Manager** below (similarly, the Colliders for these scripts will also be enabled.

**Configuring The Road Asset Chunk Manager**

The Road Asset Chunk Manager performs several tasks:

1) It locates the RoadSystem script needed by each set of Road Asset Chunks (in order to configure RoadMaterialOverrides)

2) It removes unnecessary child game objects of this Road System and also ensures it is configured not to general meshes at runtime.

3) It can (optionally) enable the colliders of the road/intersection meshes, which is typically needed since MicroVerse disables these colliders by default.

4) It configures the materials for game objects that contain RoadMaterialOverride scripts.

Each Road Asset Chunk Manager is designed to work with a single Road System, so if you are using multiple Road Systems, you will need to create a unique Road Asset Chunk Manager for each System (**Add Component -> Deep Space Labs -> SAM -> 3rd Party -> MicroVerse -> Road Asset Chunk Manager**).

Please follow these steps to configure each Road Asset Chunk Manager:

1) Set the **MicroVerse Grouping** to the index of the Grouping that contains the Asset Chunk with the MicroVerse Hierarchy. This is the Asset Chunk that contains the actual RoadSystem script.

2) Set the **Road System Name** to the exact name of the game object that contains the Road System. This is needed for the Manager to identify the Road System.

3) Enable the **Enable Colliders** setting in most instances, unless you've somehow ensured that the meshes contained in the Asset Chunks already have their colliders enabled.

4) Set the other options as you see fit; you will likely need to play around with these values to find settings that work best (aim for the largest values that don't reduce performance).

5) Each Road Asset Chunk Manager needs to be added as a World Grouping Listener to **two** World Groupings. The first is the World Grouping containing the MicroVerse hierarchy, while the second is the World Grouping that represents the actual Road Asset Chunks.

Listeners can be added by navigating to the Grouping, then to that Grouping's **Optional Components** tab.

And that's it! You should be ready to enter Play Mode at this point. If something doesn't work right (for instance, you see pink meshes after entering Play Mode), or you get console log errors/exceptions, please don't hesitate to post in the beta-general-sam channel on Discord (include any error/exceptions you see!).

# Final Word

There are other MicroVerse systems that will likely not work out of the box. Rest assured that this is just the beginning of the MicroVerse support. We will continue our work with it by testing those systems and will also revise the instructions and/or techniques described in this document based on the feedback of our users.

Please join the discussion and let us know what you think of these instructions using the beta-general-sam channel on Discord!