WORLD STREAMER MANUAL v.2.1

Support: **Discord Channel**|**Website & Support**|**Facebook**|**Twitter**
Manual author: Bartłomiej Galas

Introduction to World Streamer

World Streamer is a memory ASYNC streaming system. By using it you can stream your whole game from a disc in any axis and space. You can create endless space games, 2D platformers, 3rd person, open worlds, mobile apps, architectural presentations, single, multiplayer, or any kind of game you want without loading screens during player movement. You are also able to create endless looped worlds or planet imitations. You also could generate far distance background for your game with it.

All scene assets will be stored on the disk as scene files and they will be loaded only when they are close enough or when they meet the required conditions. This solution reduces CPU, RAM, VRAM, and GPU usage, and performance in your game will be much, much better and your application will load much faster. With streaming you will reduce the object count in your scene and you can create very detailed and large scenes without the performance drop. World streamer gives you the ability to create your scenes, as big as you want, without any limits.

"Floating Point Fix System" removes restrictions and errors caused by huge coordinates in the engine. With "Local Area Updater" you can work on a small, local area without loading the entire world. Easily add these changes to the world by clicking just a few buttons. This means you will not have Unity Editor slow down, because of loading the entire world into the editor scene.

You can also separate your world into layers such as Lighting, AI, Geometry far in the distance, Geometry Near, Trees, etc. Ring streaming gives you the ability to replace objects in the distance with lower Level Of Detail models. You can replace detailed terrain with foliage by lower poly mesh when it's far in the distance.
Collider streaming gives you the ability to load buildings, rooms, caves, interiors, etc, when you are close enough to them and when you enter/hit the collider or portal that triggers them.

World Streamer is a perfect solution for teams that work on the same project at the same time, even in the same area. Team members can work on separate areas of the world and then save their changes into the world when ready using Local Area Updater. Team members can work in the same area at the same time as well, by working on different layers.

We have provided an extensive manual as per the community's request to thoroughly explain everything, but once you understand the basics you will be able to prepare your game for streaming with only a few buttons and in a few minutes or even seconds.

If you want to use HD or URP please open "HD and URP Support Packs" folder and follow the readme files, import support packs from that folder for your proper HD and URP version.

Features:

- Floating point error fix system.
- Looped world system.
- Multiplayer support.
- Multi-scene editing support.
- Stream in multiple independent layers.
- Support for multiple developers working in one area.
- Terrain split tool
- Low Poly mesh generator from terrain and foliage
- Removes world size restrictions.
- Ability to work in Unity huge or extremely detailed scenes without slowing down the editor.
- Support for massive worlds.
- Easy automatic processes will save your development time.
- In a few clicks, you can stream an endless world.
- Work in any axis and space: 2d, 3d, vertical, horizontal.
- Reduce RAM, CPU, GPU, and VRAM usage of your game.
- Reduce the number of objects in your scene.
- Load your world in the background (async loading).
- Compatible with Amplify, Granite, and any other texture streaming system.
- Loading screen system included.
- LOD system for terrains.
- Low poly mesh generation for terrain and it's foliage
- Terrain manager – split, manage your terrain for streaming
- Additional terrain culling system, it reduce 10-90% of terrain CPU/GPU usage
- Ring streaming can replace objects with objects in the distance with lower Levels of Detail models. For example, high-quality terrain can be changed to low poly mesh when it's far in the distance.
- Collide streaming option, stream interior areas when you walk through doors or trigger a collider.
- Streamed scenes could stream scenes (matrioshka construction).
- Webplayer support.
- An innovative local area updater allows Unity Editor to load only a specific area/layer of a massive world.
- Compatible with Terrain Stitcher and Multiple Terrain Brush,
- Fully commented opensource code that will help you to work with and expand World Streamer!,
- Batching support,
- GI and Lighting manager
- The player at the safe place during the initial or teleport data loading system,
- Physics manager which will freeze dynamic objects if it's needed,
- Occlusion Culling support
- Unity navmesh with floating point fix support

Table of Contents

1. Idea of streaming - very important

The idea is pretty simple. Your huge world will be split into smaller parts by our tools. These parts are elements on a virtual grid. World Streamer will load these virtual grid elements when an "object" (player or other object) is close enough to the grid element. You can also load grid elements when the player hits a collider (collider streaming).

In most cases, we recommend using the player as this "object", not the camera. For RTS games or 2D platform games, setting the camera as the following object is a better solution. Because we save everything as a scene, you can still use lightmaps, navmesh, GI, batching, baked colliders, and everything that you can save in a scene. You could use a custom culling solution or cull objects by removing them from memory and the scene when they are far enough away or behind a wall. For an example of objects culled by streaming, you could load small stones only when you are close to them or use collider streaming to load the building interior only when you are on the correct floor or close to the entrance.

With World Streamer you can replace grid elements with grid elements from another layer depending on distance. You can use this as a LOD solution for objects that Unity's LOD system doesn't support or for objects where LOD0 is heavy - like Unity terrain objects. You can replace your Unity Terrain with a simple mesh when it's at a far distance, such mesh can even contain trees, this will greatly reduce culling and refresh operations (CPU) and memory usage. Unity terrains can be very heavy so this can save hundreds of MB of RAM. More info about working with Unity terrains, objects, etc are found further in this manual.

World Streamer can stream in layers, which means that you can put different objects on different layers. For example, you could have a layer for terrain, one for trees, one for rocks, and one for sound effects. Each layer will load objects independently.

A good way to use layers is to group objects together by size and load only small objects near the player, but have larger objects loaded in the distance. For example, buildings that need to be seen in the distance can be on their layer, medium-sized details like boxes and lanterns could be added to a 'Medium' distance layer, while smaller details like papers, dirt, and small stones are added to their own layer as well. For the terrain, a layer containing a detailed terrain can be used when the player is near, and when the player is far away from the terrain, a separate layer with a low poly mesh of the terrain can be shown.

If you put all these things on different layers, you could setup how far each object (layer) will be visible and stay in RAM, VRAM, CPU, and GPU. This is a dynamic setting so you can manage this during gameplay for the player. Each layer and its range of streaming can be configured differently. Even each axis direction X,Y,Z can have different settings for the range of streaming. Layers also can be used to separate lights, particles, and other effects from 3D content. This allows multiple team members to work on the same map and region at the same time.
After you have split your world into parts you are always able to recreate one big, un-split scene by using the Local Area Updater tool. This means that you can always undo the fragmentation of your world and merge any changes into a whole scene again.
The basic idea is to split your world into many virtual grid elements in many different layers and then load them asynchronously without any CPU impact to save a lot of performance in CPU, GPU, RAM, and VRAM.

REMEMBER: The real power of World Streamer is at build. During the build process, Unity Editor bakes many things like collisions and batching. If you use World Streamer in the editor, these operations must be done at runtime and sometimes they are not asynchronous or simply put they are heavy. You could reduce these spikes by temporarily turning off static batching, etc, but only while in the editor. You will lose performance (draw calls will grow up) in the editor. However, at build everything will be fine.

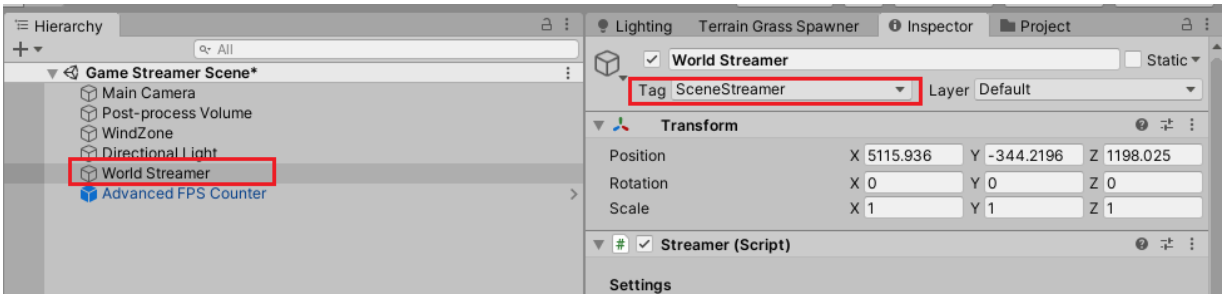
2. World System project requirements

- You must add the following tags to the project: **SceneStreamer, WorldMover**, **ColliderStreamerManager.**

  ✓ **SceneStreamer**
  The SceneStreamer tag must be assigned to any GameObject that uses a Streamer.cs script.
  The _Streamer_Major and _Streamer_Minor prefabs must have the SceneStreamer tag.

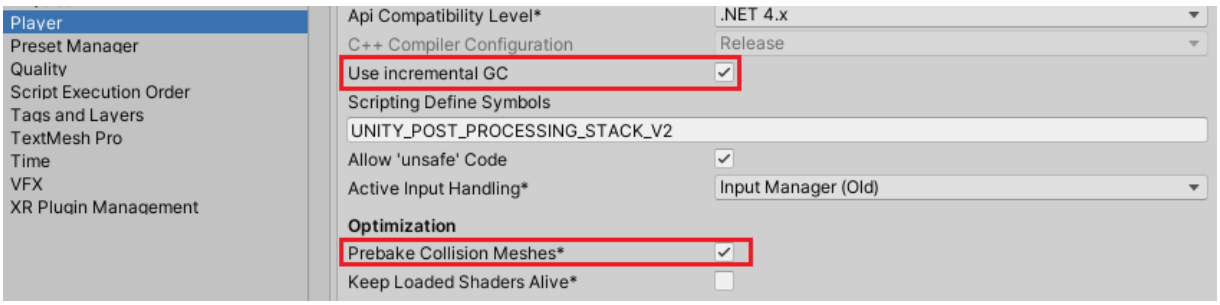
  ✓ **WorldMover**
  The WorldMover tag must be assigned to any GameObject that uses a WorldMover.cs Script.
  The _World_Mover prefab must have the WorldMover tag.

  ✓ **ColliderStreamerManager**
  The ColliderStreamerManager tag must be assigned to any GameObject that uses a ColliderStreamerManager.cs script.
  The _Collider_Streamer_Manager prefab must have the ColliderStreamerManager tag.



These streamer system prefabs must all have tags! If you do not add them, World Streamer cannot load anything at build or even in the editor. You will get many errors if you have not added the required tags. ***Always check tags in new projects!***

- **Use incremental GC** this option in project settings - player setting speeds up streaming a lot – remember to keep it turned on.
- **Prebake Collision Meshes** This option speeds up streaming at build but slowdown at editor, because unity bakes meshes into blocks. It reduces physics usage a lot.



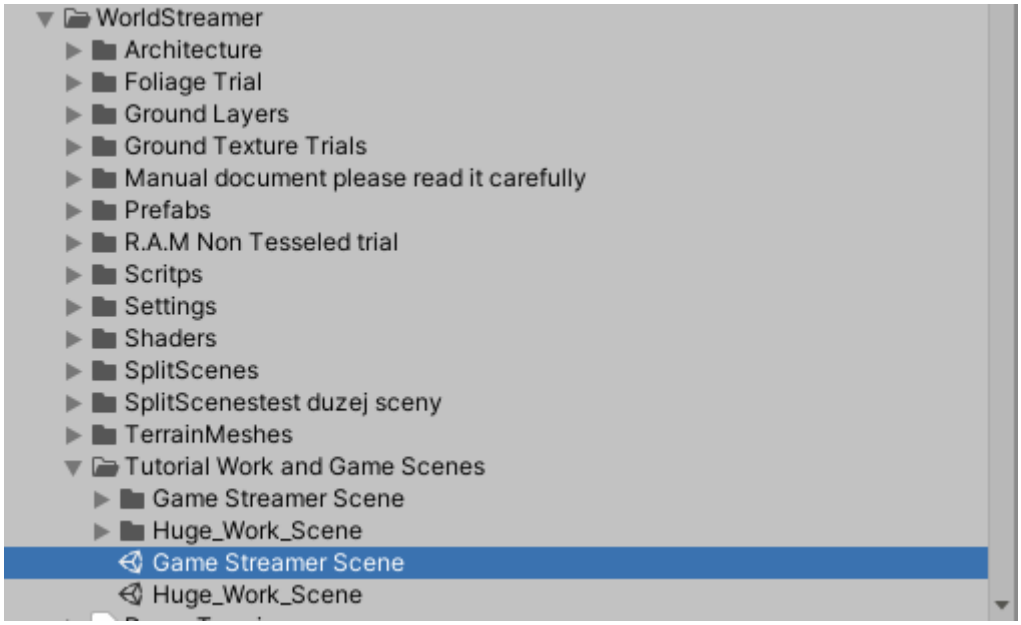
- **To start you need a "Work"** scene where you hold all models that will be streamed from the disc. This scene or scenes will be used to split your world into virtual grid elements, multiple layers
- All info about how streamer holds recognizes your models/objects, and terrains you will find in manual in the proper section.
- **To stream you need a "Gameplay"** scene where you hold: player, direction light, camera, your UI, and everything else (world) could be loaded from the disc. You will put streamer prefab there and it will load everything that you want. Please prepare it before you start. You will fill streamer prefab when your world will be split into many parts.

3. Quick Start - Test Scene

1. You must configure the proper Tags to use the demo scenes please check "2 World Streamer project requirements." section.

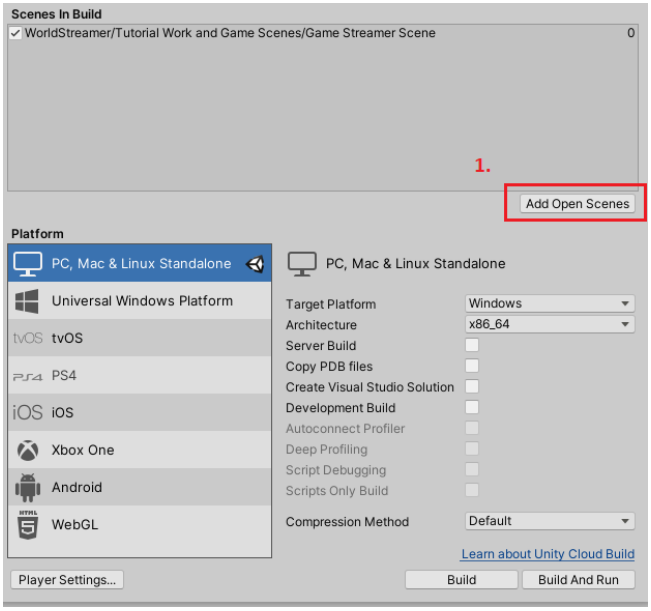
   All Streamer system-mentioned prefabs must have tags! If you will not add it, **World Streamer couldn't load anything** at build or even at editor. You will get many errors from missed tags. Always check that in new project!

2. Open any Gameplay world streamer demo scene. "WorldStreamer" catalog→"Tutorial Work and Game Scenes"→Choose Game scene.

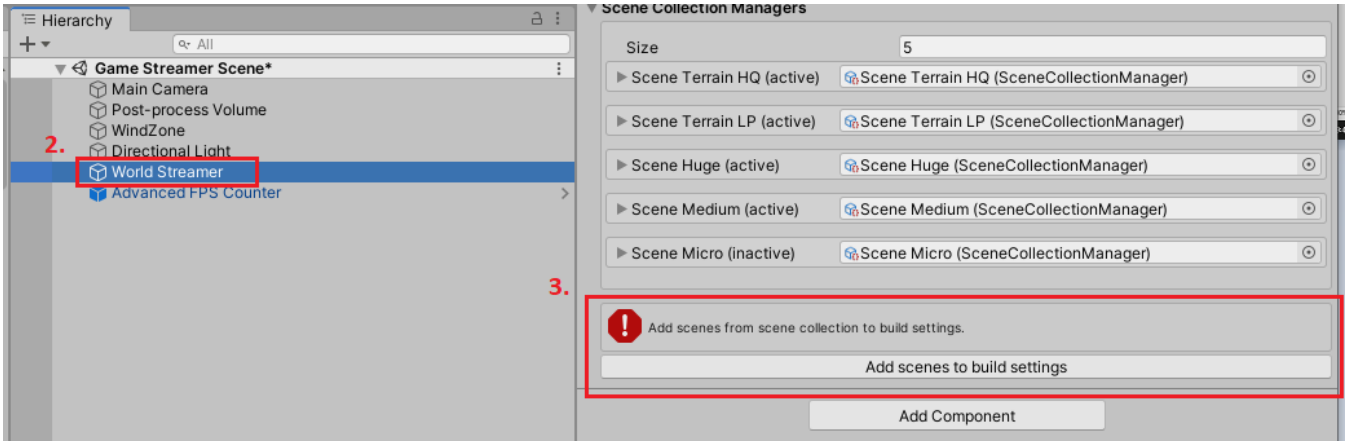


   We prepared mobile and standalone scenes to show in a few variants, with different features turned on so chose scene which will fit you.

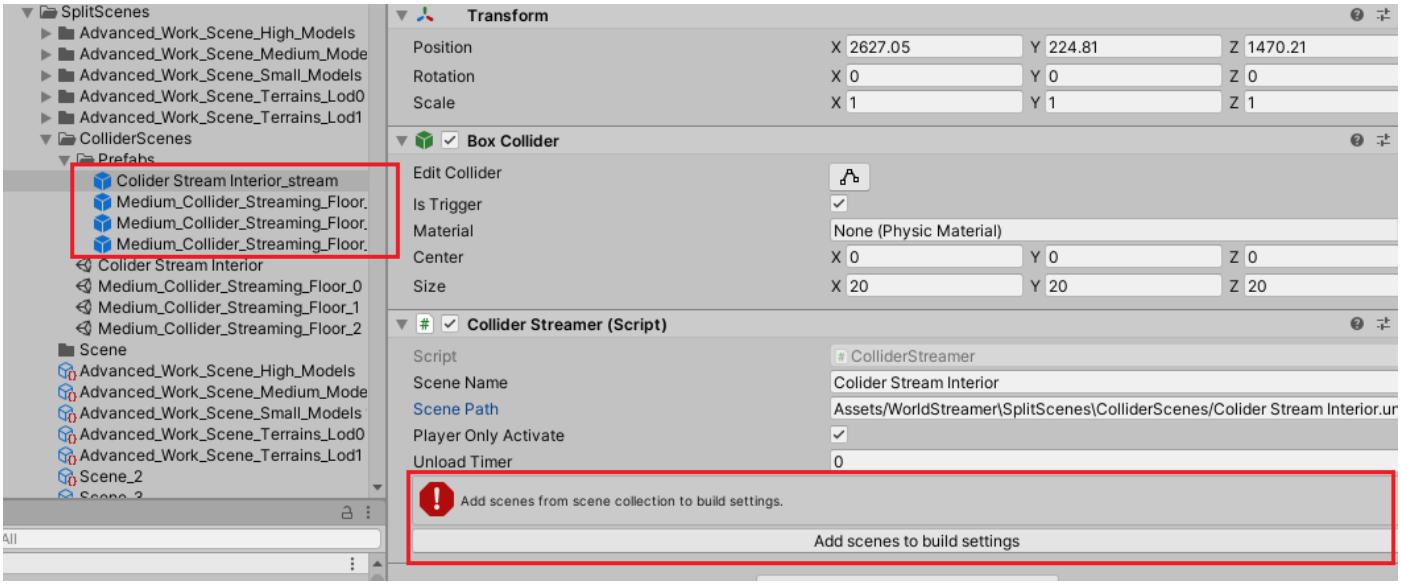
3. Add current scene to the build settings:



   Click streamer object in your hierarchy and click "add to build settings" button. We must have all grid elements (scenes) in our build settings. Without this, we could not load them.



4. As we also stream models by collider streaming in our scene, we have to add our scene with building interior to build settings so: "WorldStreamer" catalog → "SplitScenes" → "ColliderScenes" → "Prefabs" → click on "Colider Stream Interior_stream" prefab and others.
   Then you have to click add to build settings button. Do the same operations for others".



Of course, there is easier, more automatic and faster way to do this but now we want to show some basic information about how system works.

5. **Hit play or build project and run around**, take a look how everything is streamed. We suggest making a build because play mode is overloaded via operations that normally are made during build like batching, collision bake etc.

4. World Streamer details and settings

Streaming solution consists of:

- **Scene manager.** This big tool is used to create virtual grids and layers from your scene objects, it's also used to manage light settings, split terrain, manage terrain in chunks. Scene manager also generates scenes from virtual grid elements. During scene generating process scene splitter create also scene collections. Scene collection is created for each layer.

- **Scene collections.** This prefabs hold information about scenes, grid element size, world size and much more details that are useful for streaming process. As we mentioned they are formed and refreshed by scene splitter or local area updater tool, during scene generation process.

- **Scene objects**. They are connected to scene collections, they are formed and refreshed by scene splitter or local area updater tool during scene generation process.

- **Streamer prefabs** They hold and use scene collections, in your "Gameplay" scene and they stream content according to options that you set. It hold layers that you want to stream. You can hold everything in one streamer or multiple streamer objects.

- **Collider streamers.** They holds scene which will be loaded after chosen object hit his collider.

- **Collider_Stream_Manager.** Is a prefab that give you ability to stream objects by Collider streamers. This object also gives you ability to stream scenes from colliders which are hidden in grid elements. Yes, you could create matrioshka streaming construction (streamed scene that stream scenes).

- **Local area updater.** This tool gives you ability to merge scenes, load only local area, refresh/modify/remove parts of the already split world. It gives developers the ability to work on the same area, at the same time.

- **Streamer GUI.** Our loading screen, teleport, respawn, position debug, UI which you could modify and adapt to your game construction.

- **World_Mover.** This object is used in floating point fix system to restart world position and it also holds connection between real and local restarted world position. This connection is useful for spawned objects or server communication.

- **Player_Mover.** This script is used to move player into safe place until initial data loading/teleport will end.

- **Player teleport.** It's an object that holds respawn/teleport point position. This could be dynamic object.

- **Object to move**. This script holds relation between non-streamed object and player, while floating point fix system resets the position.

- **TerrainCullingSystem.** This script gives additional performance boost to terrain rendering and streaming.

- **Physic Culling System.** This script hold/freeze dynamic physical objects until player will be close enough. It saves object direction and speed vector. This help to avoid situations with missed collisions because data haven't been loaded for this area yet.

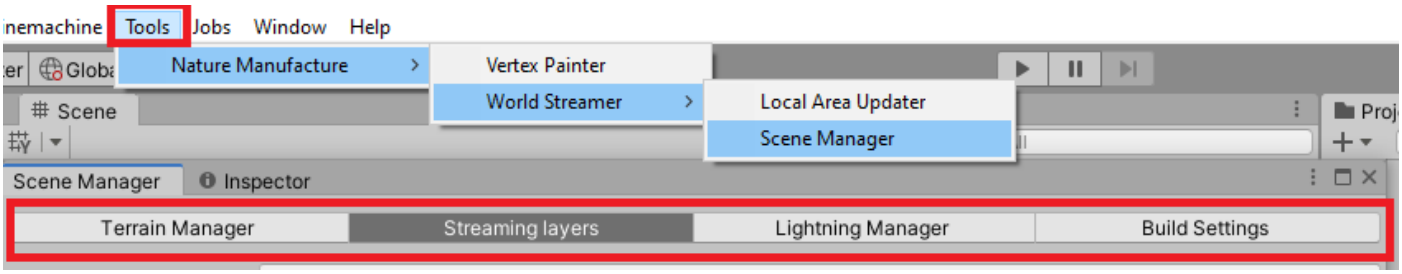
- **Object Parent.** This script should be attached to hierarchy catalog (parent), if you want to use catalogs during split process. This give you ability to hold objects in parents even during split process.

4.1 Scene Manager

This tool is used to split your world objects into virtual grids and prepare them into streaming. As objects we mean: models, objects, lights, particles, terrains. Every grid element "grid eye" will be a "pack" of data that World Streamer load into memory as one object.

We always advice you to open "scene manager" at empty scene that we will call "Work" scene. You should copy into it whole content that you want to stream. Directional lights, player, UI should be at your "Gameplay" scene, so remove them from this "Work" scene or hide during splitting process. If you will properly set your layers, this objects could stay here.

To open "Scene Splitter" window click at Tools -> Nature Manufacture -> World Streamer->"Scene Manager



When you open "Scene Manager" window you will find few sections.

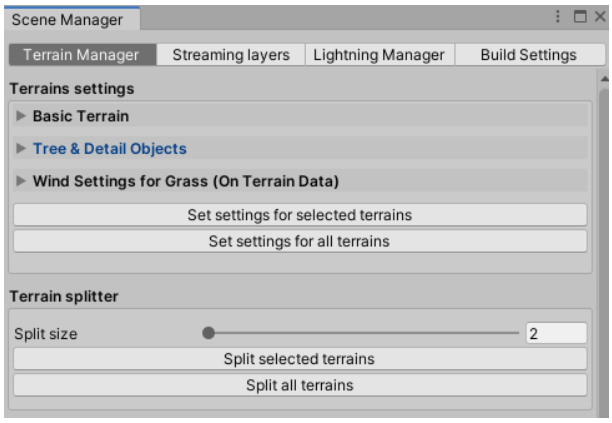
Terrain manager – it will prepare terrains for streaming – cut, generate low poly meshes etc

Streaming layers – it will prepare streaming chunks and generate scenes

Light manager – You can set light settings for your streaming scenes

Build Settings – you can add scenes, collider streamers to build settings

Terrain Manager



Terrain Manager which will prepare your terrains for streaming for example cut it. It will also be helpful for terrain chunks estimation, change terrain grass, tree settings globally, bake low poly meshes for terrain and trees in far distance.

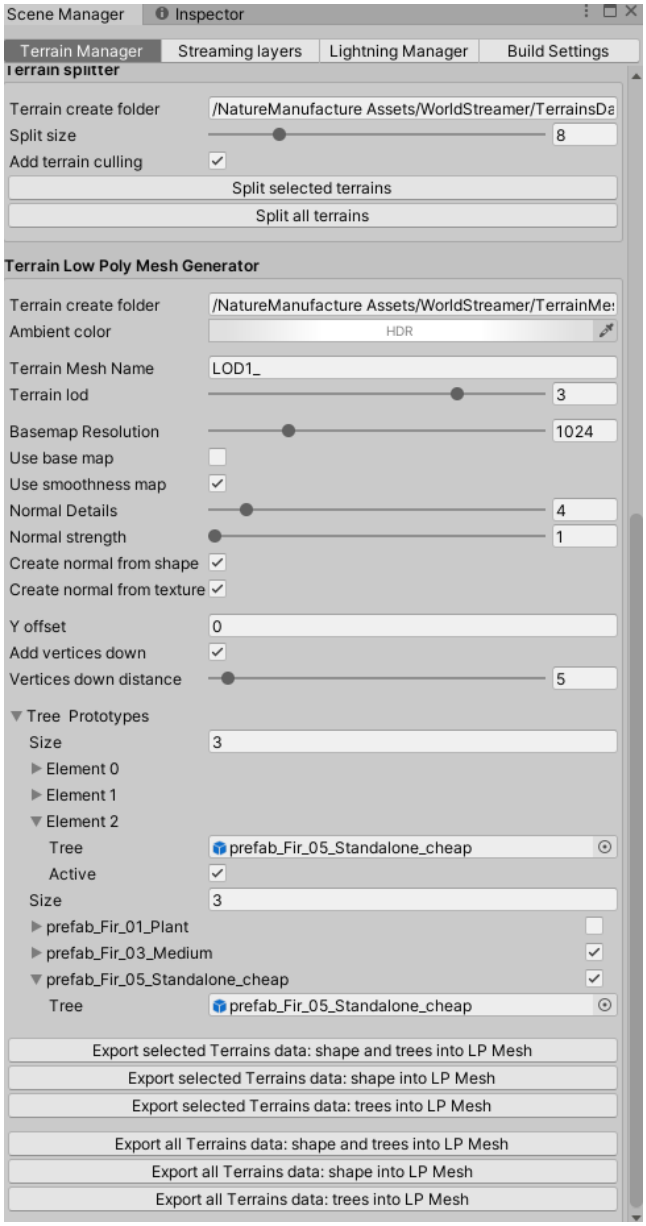
Terrain Settings are basic terrain settings that you will get after terrain split in terrain parts. If you have terrains already in chunks you can change terrain settings for all terrains at the same time.

Terrain splitter will split your terrains into chunks without any external tools. You chose how many times you will split each terrain. For example 1000x1000 terrain with 513 heightmap, split by 2 will result in 4 terrains with size 500x500 and 257 heightmap resolution.

Add Terrain Culling – Add culling script for all terrains, it give additional performance boost as it turn off terrains behind camera.

Terrain Low Poly Mesh Generator - This very important tool is able to create low poly meshes from your terrain including trees. Remember that low poly meshes from trees works only with our NM trees as they got proper shader. They may work also with trees that have cross models as last LODs. Basically for trees we take last LOD and bake into single mesh with proper shader. Our mobile demo example contain cross models from not NatureManufacture technology trees.

Settings:



Ambient Color -it's set automatically to achive best result but you can manage low poly generated terrain color by it.

Terrain Mesh Name - future name of low poly mesh

Terrain LOD – how dense low poly mesh should be, we set it to 3 as standard and in most cases it's enough.

Use Basemap – you can decide if generated texture from terrain will be result from basmap or splatmaps. For custom shaders like cts and other we advise to do splatmaps. If you render basemap in shorter distance then terrain tile size check this option.

Use Smoothness – decide if smoothness map from terrain will be saved on basemap alpha channel.

Basemap Resolution – chose texture resolution at low poly meshes generated from terrain.

Normal Details – our system also bake normalmap for low poly meshes to give most accurate terrain representation. With this value you decide how detailed normalmap at terrains should be. We set it to best values as default. Result should be best.

Normal Strength – value which decide how sharp normal map at low poly terrains should be.

Create normal from shape – you can create normalmap from terrain shape so it will be more accurate.

Create normal from texture – it will create normal from terrain textures and blend it with shape normal if it's checked.

Y Offset – This values move a bit down low poly meshes in reference to original terrains.

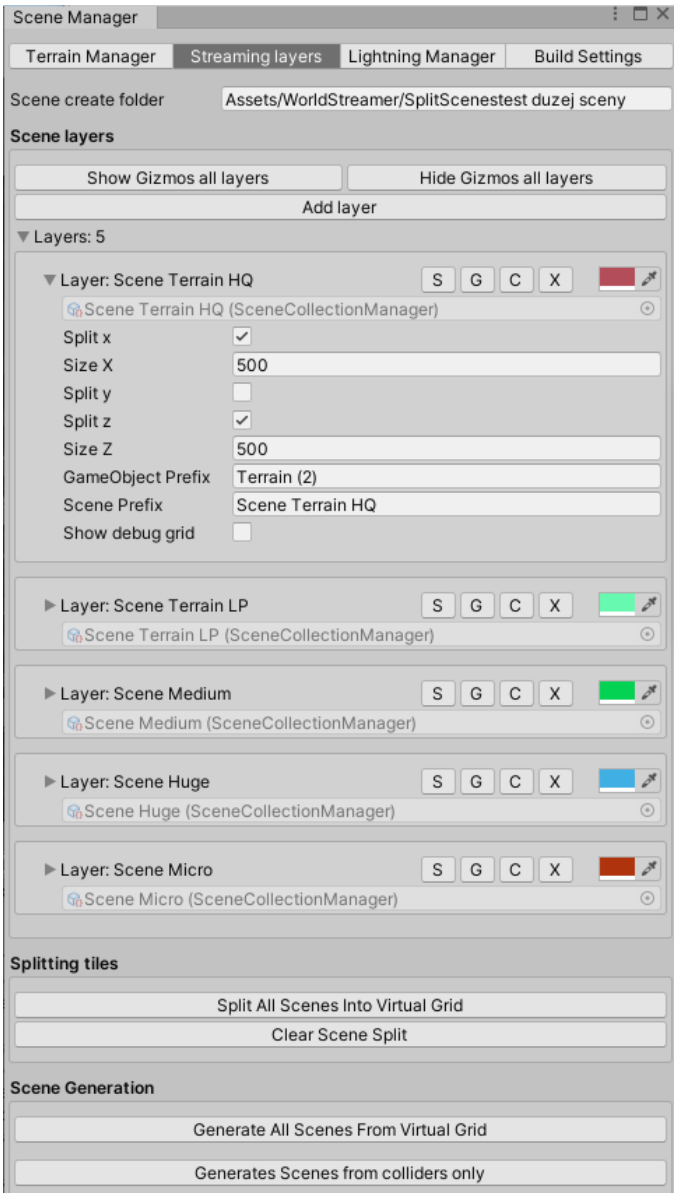
Vertex down distance – To avoid gaps between low poly meshes and original terrains we move down verts at terrain borders. This value decide how long this "wall" around terrain should be.

Tree Prototypes – System read tree prototypes from terrain, you can change it into low poly meshes so trees will not disappear in far distance and they will take only few batches. Remember that not all trees should be converted into low poly mesh. For example small trees which disappear earlier are rather waste of source. You can switch off trees that should be baked in our example small tree plant. System will take last lod's from prefab and bake one big single mesh from them. With our cross solution it works perfect.

Export buttons – they export terrains into low poly meshes, with trees or without or only trees. You can decide what to do.

Streaming Layers

This part will split all objects into virtual grids and create streamed scenes from it.



- **Add layer** button creates a group/layer of objects. Layer separates specified objects from the rest of objects, this layer/group will be streamed independently from others. As an example of layers/groups we could use: Terrains, Small models, Big models, Medium models, Far low poly Terrains, Lights, Particles, FX.
- **Split X, Split Y, Split Z** buttons give you ability to chose what Axis will be used during virtual grid generation. If you create space game you will probably check X,Y,Z . If you create RTS of RPG, Car game you probably will check X and Z because height is not so important. For unity terrain you should check only X and Z. If you want to unload terrains when player is to high over terrains you could check Y too.

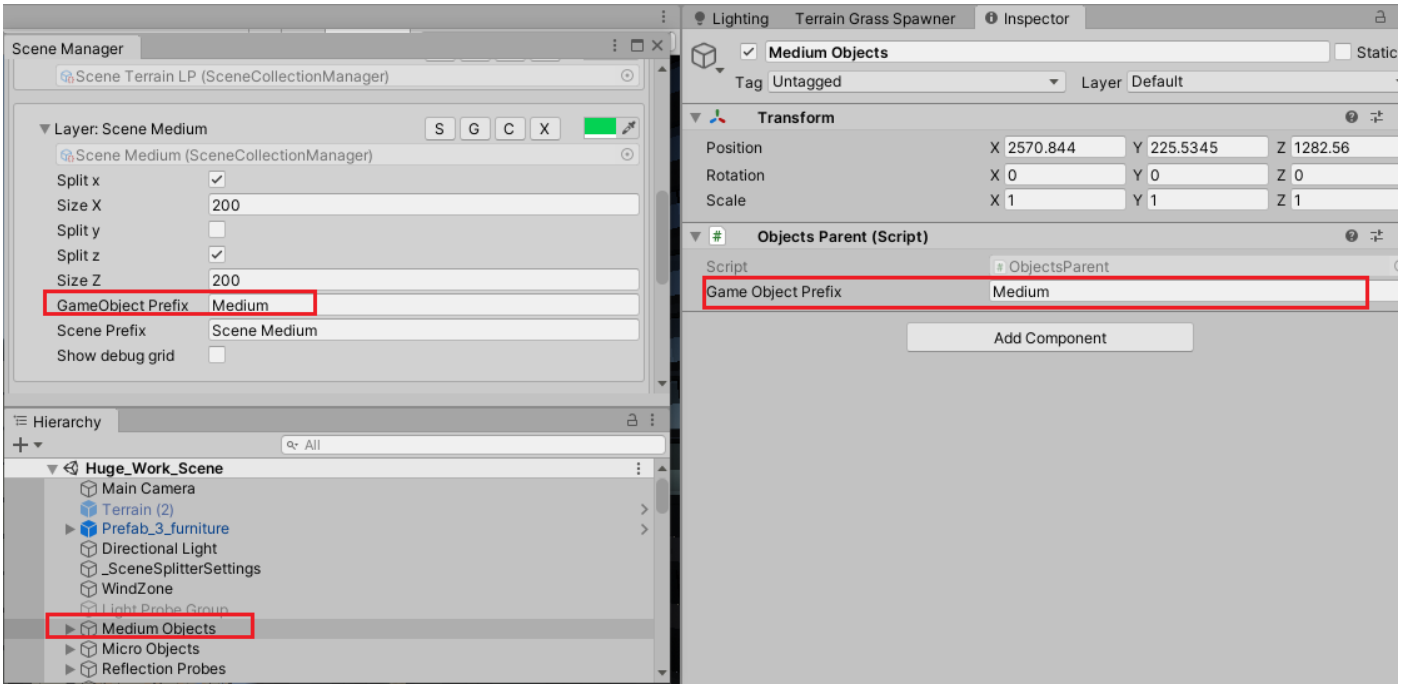
- **X size, Y size, Z size** values are responsible for size of one virtual grid element "eye". You are creating virtual grid which is based on your checked Split X, Split Y, Split Z axis and their size which you chose right here. Grid element size should be estimated by you. You could easily change it and test different values. They shouldn't be too big because you will load too much objects at once. They shouldn't be too small because you will have huge amount of scenes, which are probably unnecessary. You have to confront that also to object size, if your big building is 200x200 units you shouldn't stream it in 20x 20 grid element size, but you could stream a group of these buildings in 1000x1000 virtual grid element. Of course if they are not too heavy as one grid element.

For **unity terrain** "X size" and "Z size" should/**MUST** be the same size as terrain length and width. More about unity terrains you will find in **"5. Terrain streaming."** exactly in **"5.2 Scene split and virtual grid setup."** section.

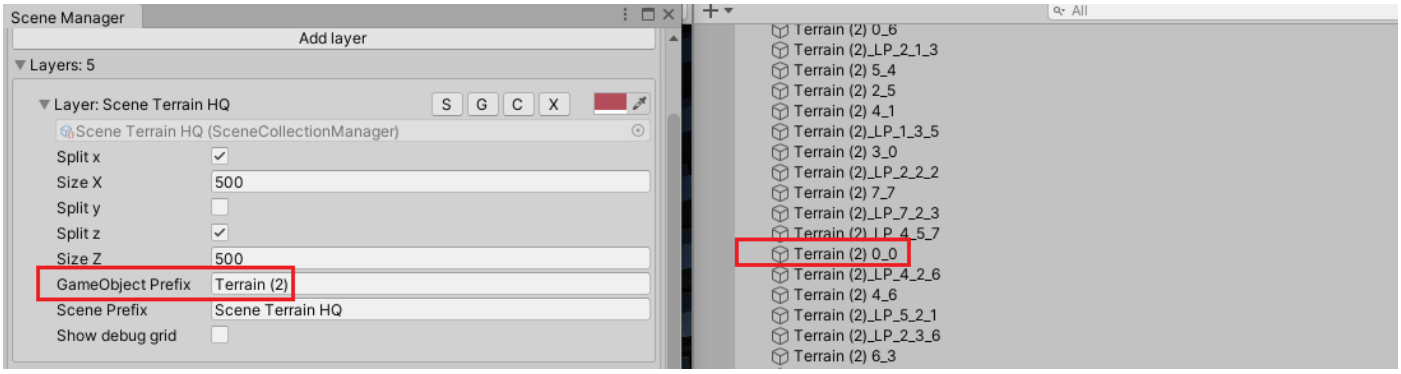
You could **visualize your virtual grid element size** by using our gizmo. After you click "Split Scene" or "S" button, when you click at virtual grid element that system just created, you will see its boundaries. **Before this** you have to fill all info about virtual grid and layer to create first split.

- **GameObject Prefix** is really important value. While splitting system is searching for objects, because it wants to put them into correct layers, it's checking each object's name. If you will leave game object prefix as empty, system will put all objects at the scene into one layer. If you want to split objects from specified folder in hierarchy you have to add script into folder “Object Parent” and set same prefix in script as it’s in scene layer.

Example1.:



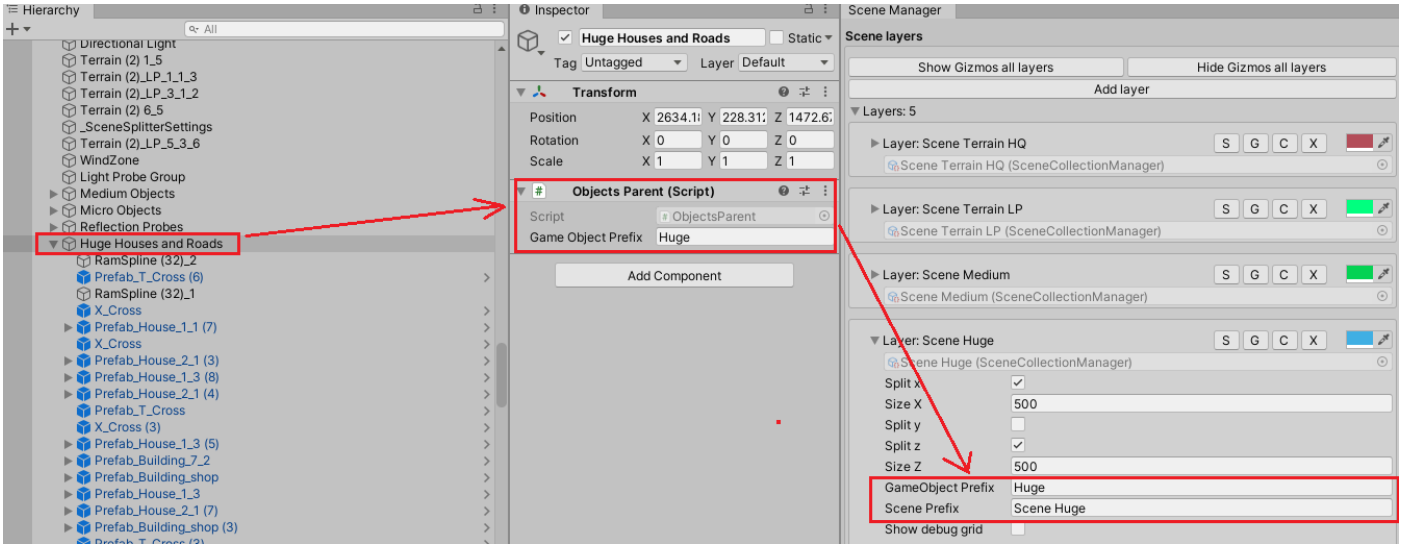
If for **Example.2** you want to put your unity terrains into layer "Terrain (2)", fill game object prefix for example by "Terrain (2)". Add "Terrain" word at beginning of each terrain object at the scene.



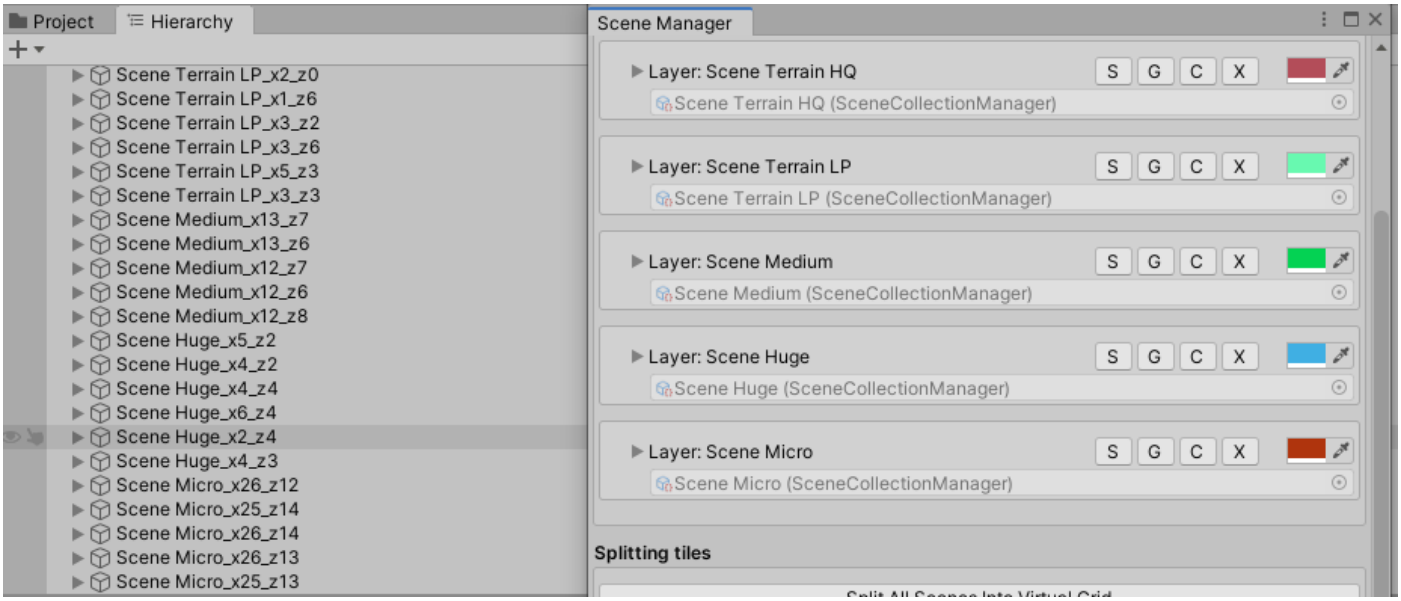
If for **Example.3** you want to put your small stones, dirt, papers into layer "Scene Micro", fill game object prefix for example by "Micro". Add "Micro" word at the beginning of each object at the scene that you want to put in this layer OR put it in folder Micro with “object parent script”. No matter which solution will chose, I guess folder is more user friendly.

- **Scene Prefix (layer name)**, in this place you choose name of your virtual grid and scenes (virtual grid elements) for each layer. For example you could leave it the same as "GameObject Prefix" or add info about scene/world that you are currently working on, like: World_A_Small_Models, World_A_Terrain_Lod0. After you click split, virtual grid elements will appear and they will hold "Scene Prefix" in name + number of row and column that they have in virtual grid. Look at images below.

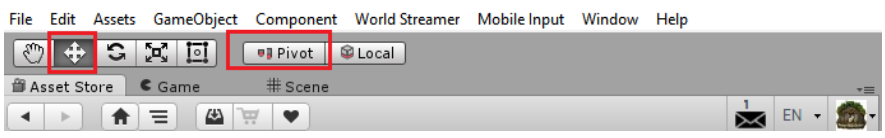
Before **split:**



After split:



- **Split Scene into virtual Grid** when you hit this button, system will check all objects in the scene and put them in correct virtual grid element and layer. This will be done for all layers and objects. In first step scene splitter will check all objects names, if they are contain correct/specified prefix, it will put this objects to layer which contain this prefix. In next step system will check objects position (**their pivot!**) in layer and will move them to correct virtual grid element. Look at image above. Remember If you put your object into other object as child, from this moment his pivot position will be taken from his parent/mother object. This could change his position in virtual grid. It depends how big difference is between parent/mother and child objects pivots positions. To check your object pivot position click at your object in scene hierarchy and you need this option to be enabled:

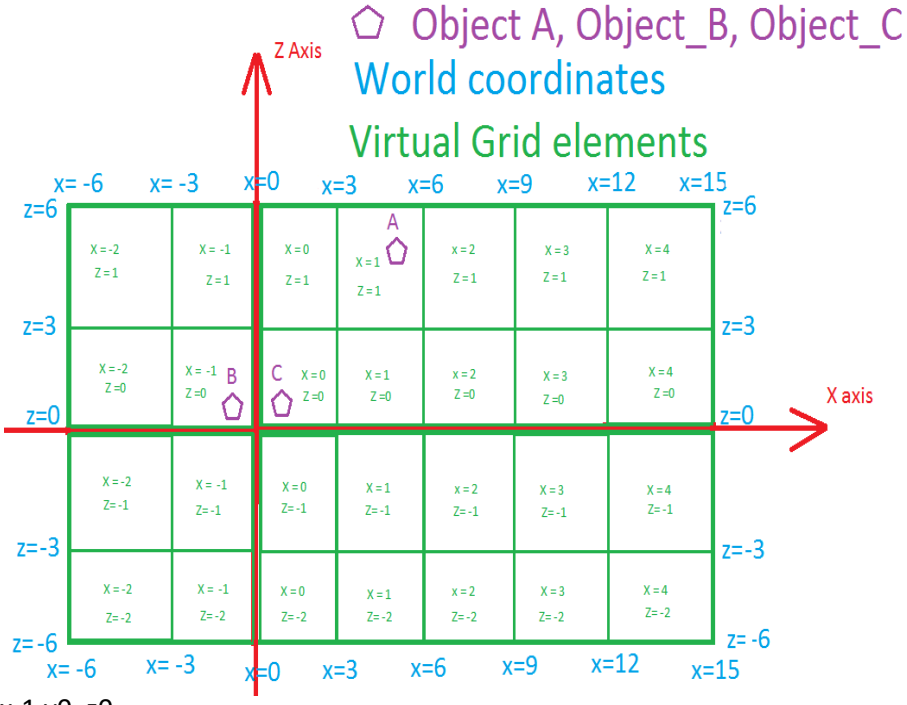


Example:

We have Layer called "Models", it has Game object prefix "Object", virtual grid size is X=3 Y=3 Z=3. So we have virtual grid with grid elements that are 3x3x3 units each.

At scene there are 3 objects:
Object_A with position x=5, y=5, z=5 ,
Object_B with position x=-1 y=,1, z=1,
Object_C with position x=0, y=0, z= 0,

Take a look at image which will help to understand example.



Important: Virtual grid elements starts from 0, 0, 0.
After click "Split scene into virtual Grid " button system will move:
Object_A to layer "Models" and to virtual grid element with name Models_x1_y1_z1,
Object_B to layer "Models" and to virtual grid element with name Models_x-1,y0_z0,
Object_C into layer "Models" and to virtual grid element with name Models_x0_y0_z0
Part of name "x-1,y0,z0" in virtual grid element is number of column and row in virtual grid.

- **Clear Scene Split**, when you hit this button you will undo whole split operation. All objects will be unpacked from virtual grid elements and virtual grid will disappear.

- **Generate Scenes from Virtual Grid** when you are sure about virtual grids and their elements and you hit this button, system will prepare/generate scenes from virtual grids elements for each layer. It will also generate scene collection prefab in scenes catalog. Scene collection prefab holds whole information about layer and scenes/virtual grid elements connected to this layer. In short we could say, when you click "generate scenes" button, system will change virtual grid elements into scenes with information about layer that they belong to. This scene collection will be used by streamer to stream entire world.

- **Scene create folder** is a place at your project where your scenes and Scene Collections prefabs will be stored after you click "Generate Scenes" button.

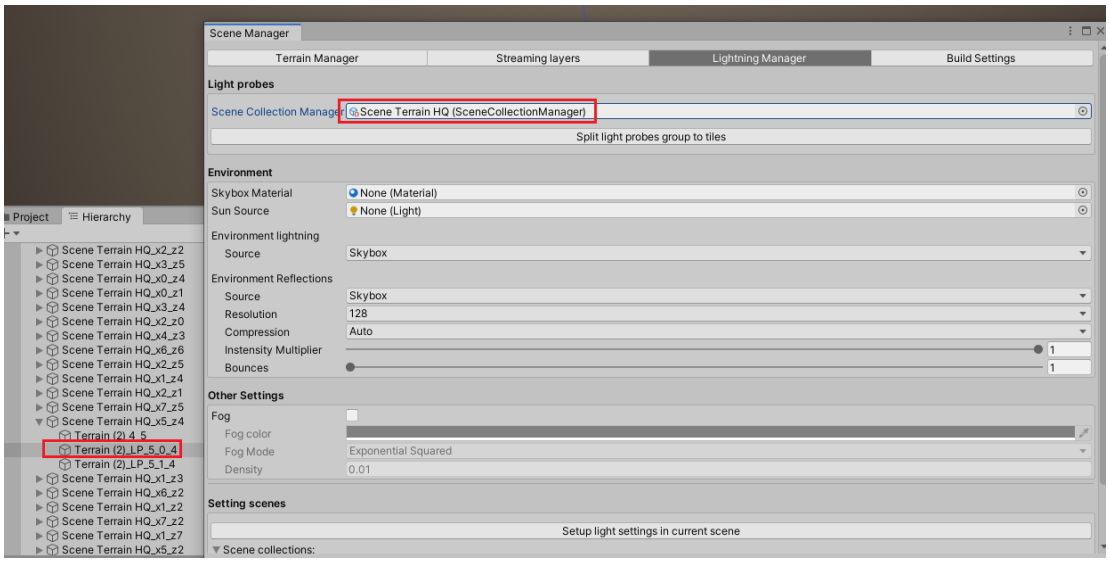
- **Buttons S,G,C,X**
  S - you are able to split/generate virtual grid elements only for this layer separately from others
  G - you are able to generate scenes from virtual grid elements only from this layer separately from others
  C - you are able to unsplit/remove virtual grid elements only for this layer separately from others
  X - you are able to delete this layer separately from the others

- **Layer order** is really important, but only when you create a layer with empty "GameObject prefix". If you put layer with empty "GameObject prefix" as first, system will move all objects at the scene and create virtual grid elements with them. Other layers will have empty list to check and as a result they will be empty

- **Gizmo color** when you click on virtual grids, scene splitter will show you their boundaries in color that you specified. This gizmo color will be also useful to visualize streaming range for each layer in "Gameplay" scene.
- **Generate Scenes from Colliders only,** when you hit this button and you have prepared objects for collider streaming, system will generate "Collider_Stream" prefabs and scenes for collider streaming system. More info about this you could find in "7. Streaming by collider" section.

Lightning Settings options

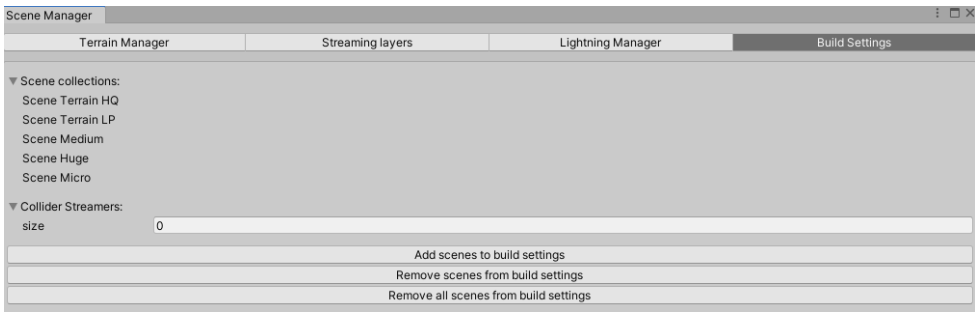
It's setup for generated scenes, you can set specific light settings for them so all scenes will contain same lightmap, baked ambient etc settings. This will avoid warnings and errors during streaming. It's also used to split your light probes and distribute them into streamed scenes.



Scene collection manager – here you drag and drop scene collection which should be corelated with light probes, Light probes will be cut into parts in same size as chosen layer and moved , renamed into proper layer split folder. Look at image above. Layer contain light probes and terrain. Such layer must be regenerated after this process. Source light probe grid will be disabled/turned off at scene and only probes in parts will stay to avoid duplicated messed probes.

Rest setup is used to set specific light settings at scenes, like fog etc. Basically initial settings will be taken from current work scene, so if setup at work scene is properly you don't need to use it.

Build Settings options



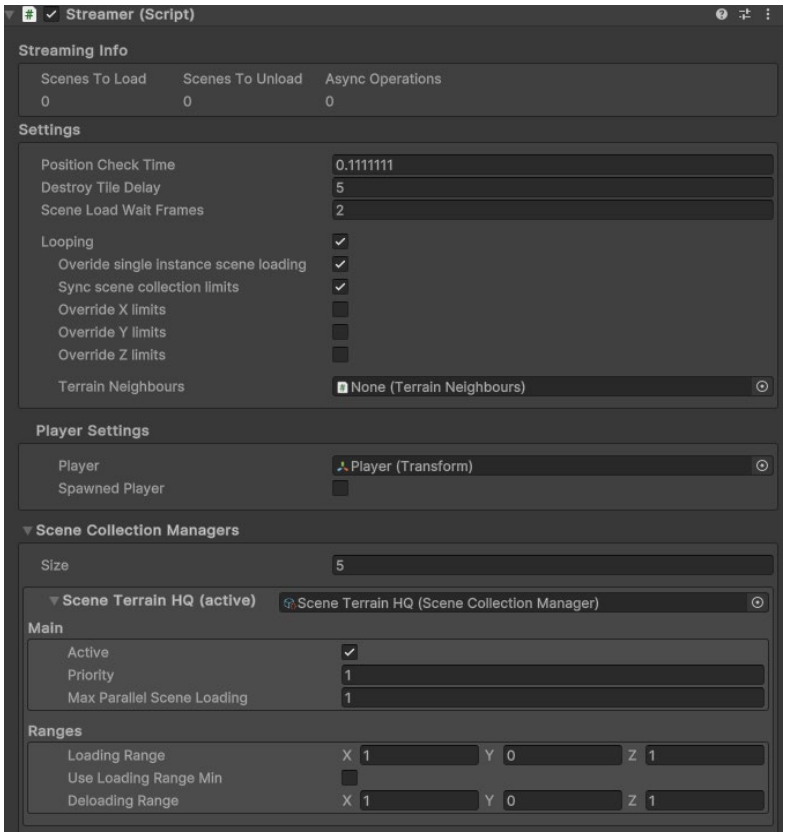
- **Scene collections,** here you can drag and drop your scene collections prefabs to add massively their content to build settings. If you open this window at "Work" scene it will be automatically filled by all scene collections that were created at this "Work" scene.

- **Collider Streamers** here you could drag and drop your "Collider_Stream" prefabs to add massively their content to build settings. If you open this window at "Work" scene it will be automatically filled by all "Collider_Stream" prefabs that were created at this "Work" scene.

- **Add scenes to build settings** by clicking this button you will add to build settings all scenes from "Scene Collections" and "Collider Streamers" list.

- **Remove scenes from build settings** by clicking this button you will remove from build settings all scenes from "Scene Collections" and "Collider Streamers" list.

4.2 Streamer objects/prefabs and their settings



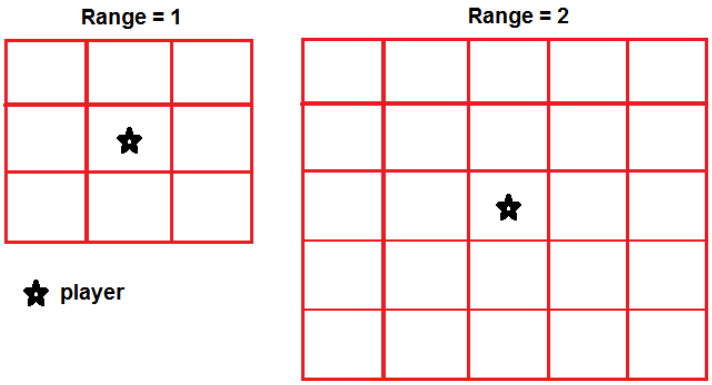
- **Active** you can activate or deactivate layer in streaming.

- **Priority** will set order of streamed layers, system start from lowest value into higer.

- **Scene Collection** is a place where you connect streaming system with your scene collections prefabs, which holds information about virtual grid elements(scenes). Simply drag and drop scene collection prefab into this window. Scene collection prefab is in catalog with generated scenes, which it belongs to, so WorldStreamer→SplitScenes→Layer name

- **Loading range** is an integer value in each axis that precise virtual grid elements radius that you want hold at RAM memory at once. Loading range could be different for every axis in your world. When you click on Streamer object you could see gizmo at your screen with current loading range area, look at screen above.

Short interpretation:
"0"- 1 virtual grid elements,
"1" - 9 virtual grid elements,
"2" - 25 virtual grid elements,
"3" - 49 virtual grid elements etc...



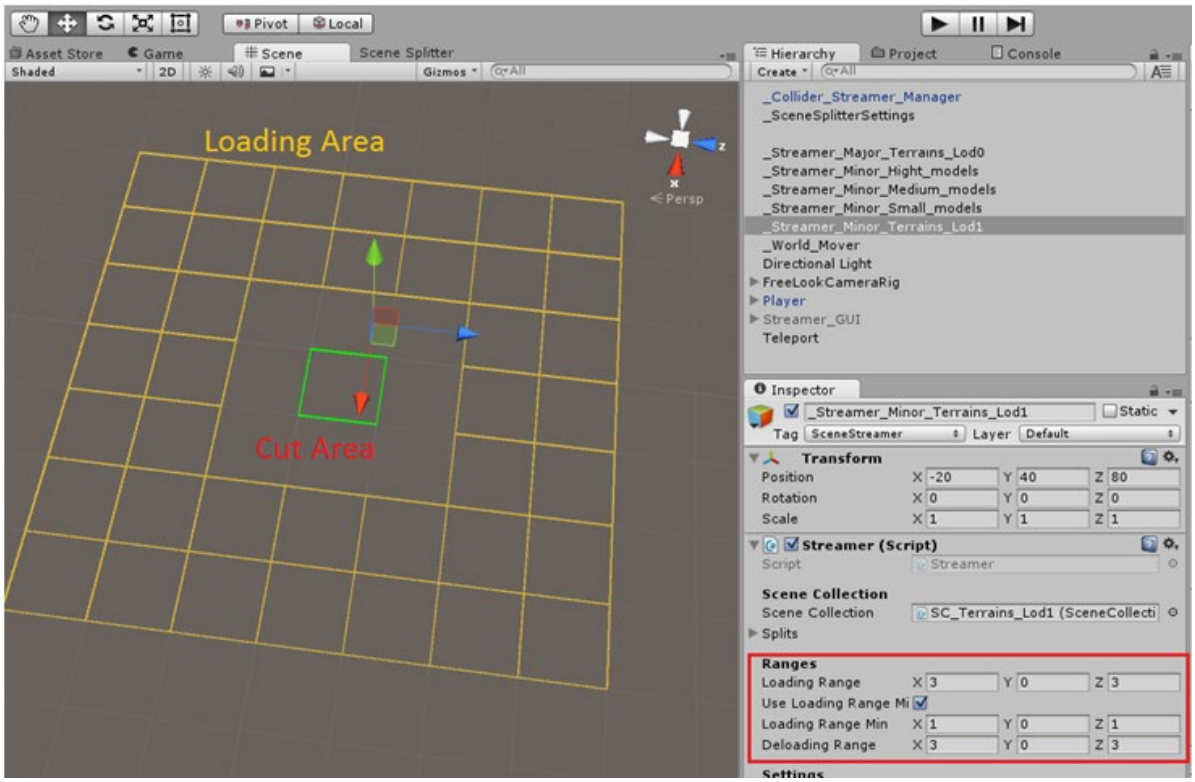
Loading range could be different for every axis in your world. If world is looped loading range must be shorter than half your world in virtual grid elements minus 1 virtual grid element.

- **Use Loading Range Min**, checkbox it allows you to use "Loading Range Min" we called this Ring streaming. This is our fresh system that allows you to load objects in specific range "from - to". We called it ring

streaming because streaming area looks like ring. If objects are too close they will be removed, if they will be too far they also will be removed.

- **Loading Range Min**, **(Ring Streaming)** is an integer value in each axis that precise virtual grid elements radius that you want to cut out from "Loading Range". Streamer will load everything in the ring. If you will be close enough it will remove models from memory or doesn't load them. It's useful to replace models/objects in close distance by models/objects from other layers. When you click on Streamer object you could see gizmo at your screen with current loading range area. Look at image below, we have cut area, from loading area.

- **Sync with other layer** – this option appear when you check loading range min. It sync loading and unloading between 2 layers to avoid any gaps and streaming problems. Data will not going to be unloaded until other layer will be ready to replace it.



- **Deloading range** is an integer value that precise radius after which system will unload virtual grid elements(scene). Remember! Virtual grid element will be unloaded after time that you set in "Destroy tile delay", so first criterion is distance then time. **If world is looped deloading range must be shorter than half of your world in virtual grid elements minus 1 virtual grid element.**

- **Position check** is the position refreshing time in seconds. Streamer is loading/unloading your world saved in virtual grid elements(scenes) depending on the following object "Player/Camera" position in XYZ world.

- **Destroy tile delay**, value/time, in seconds, is useful to precise time, after which your virtual grid element(scene) will be unloaded from RAM memory, when virtual grid element(scene) will leave "Loading range". This value is useful when player is moving along the border of virtual grid elements or the player often moves between the same virtual grid elements. This will help you to avoid numerous load/unload actions in short time. This is also useful at "ring" streaming, old models will wait some time until they will be removed, in this time next layer will fill this place with other objects.

- **Max Parallel Scene Loading** is an integer value that gives you the opportunity to decide how many virtual grid elements(scenes) you want to stream at the same time. This option is useful to optimize streaming. Low

value will cause longer loading but you won't get any performance drop. You have to estimate and check few values of this to fit your purposes.

- **Scene Load Wait Frames** is value in frames useful to set delay between virtual grids elements loading. It's recommended to set few empty frames during streaming process, this will save streaming pipeline from overload. If you stream small virtual grid elements you could set 1 or 0. If time that you chose will be too long, streaming will be very slow only because of empty frames. You have to estimate and check few values this for your purposes.

- **Player** is place where you put object that represent player position for streaming system.
- **Looping** checkbox enables looping for this scene collection, this creates endless world. Each streamer with its scene collection is looped independently.
    - if all layers have different amount of grid please check : **sync scene collection limits,** this will sync them
    - if world is not huge and you allow to load scene multiple times at the same time, simply check **override single instance scene.** In such case you can loop any size of the world.
    - You can force looping on specific distance value **"Override X, Y ,Z limits**". For example if you have loading range "0" on "Y" you can loop world on Y via specific world distance.
- **Spawned Player** checkbox which means that streamers should wait until player will spawn at gameplay scene. System will wait for object with selected tag "**Player tag**" window. **If this option is enabled leave all "player" windows in each streamer and rest of the scripts that use player object as empty.**

4.3 Teleport or Respawn

Drag and drop into your "Gameplay" scene "_PlayerTeleport: prefab. Player will be teleported into position that is equal to this prefab's XYZ coordinates. Fill streamers windows with all streamer objects that you use in this scene (from hierarchy). If you want to use loading screen at teleport/respawn, then fill "UI Loading Streamer" window. If you are going to use floating fix system please fill "World Mover" window. Below is the information about each setting, you will also find images with information about how to setup your respawn/teleport into your game.

- **Ui Loading Streamer** is a place where you can connect your UI canvas and loading screen with teleport/respawn system. This allows you to use loading screen system after player will be respawned or teleported. Simply drag and drop object from your scene hierarchy that contains UI Loading Streamer Script into this window. In our prefabs it's child of "Streamer_GUI" object.

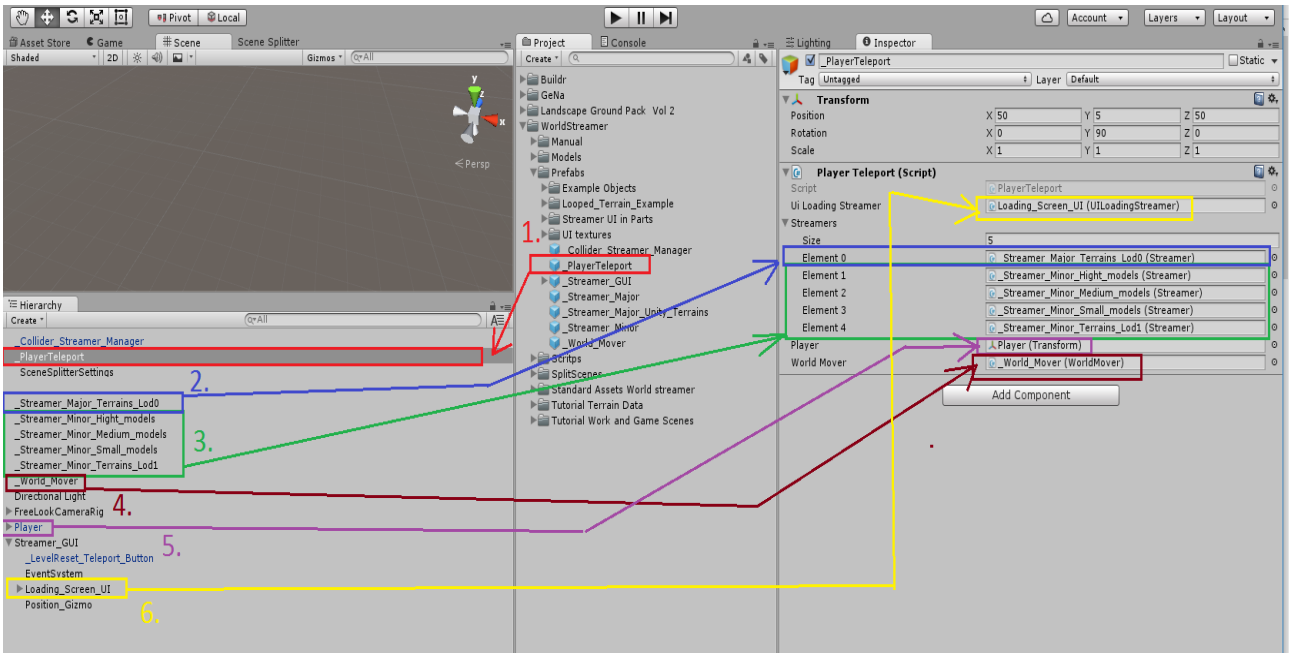
- **Streamers** is place where you should put all streamer objects that you have at scene hierarchy. If you have more than one, grow up size and drag and drop all of them into this place.

- **World Mover** you should fill it by World Mover object (object that contains world mover script) from your scene hierarchy, only if you use floating point fix system.

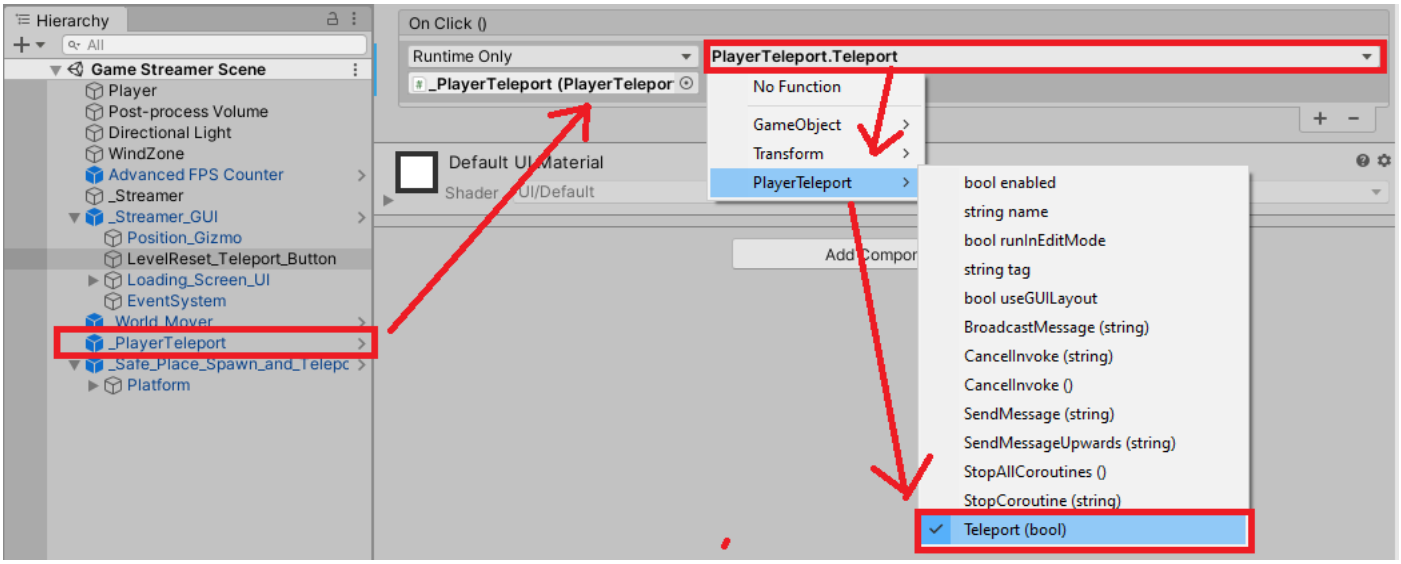
- **Player** is a place where you drag and drop object from your scene hierarchy that represents player position for stream system. This object will be moved during respawn/teleport process.

- **Player Mover** - drag and drop object which contains player mover script from scene hierarchy if you use safe place system during loading/teleport. In our prefabs "_Safe_Place_Spawn_and_Teleport" contain this script.

Image with quick information how to setup respawn/teleport



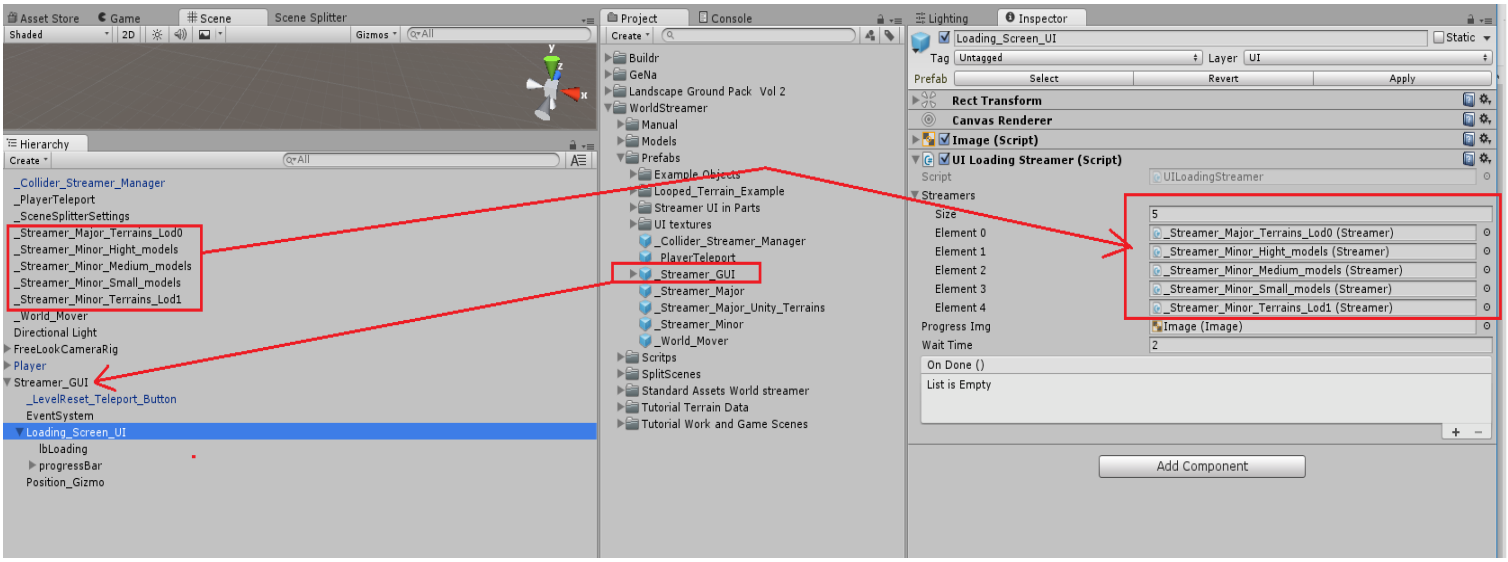
If you want to use our teleport button simply drag and drop our "LevelReset_Teleport_button" prefab and "EventSystem prefab" into your scene. Fill "on click" window by "_PlayerTeleport" object from your scene hierarchy. Chose button function PlayerTeleport→Teleport(bool). Button and Event system is also attached to "Streamer_GUI prefab" Look at the image below.



4.4 Loading Screen UI for game start/teleport/respawn

For start you could use our simple loading screen system. It contains loading progress bar and it's easy to apply. You also could use our scripts to build your own loading screen system and connect it to world streamer.

Drag and drop into your scene "Streamer_GUI" prefab or more simple "Loading_Screen_UI". "Streamer_GUI" is more advanced in relation to "Loading_Screen"UI", it contains additional things like "EventSystem", "Position_Gizmo", "LevelReset_Teleport_Button", and it also contains "Loading_Screen"UI". Fill "Loading_Screen"UI" with streamer objects from your scene hierarchy, which affect loading screen. In some cases not all streamers should affect loading screen. As an example we could give you small objects which are in memory only at close range, they are not so important so they could be loaded during gameplay.



- **Streamers** is a place where you should put all streamer objects that you have at scene hierarchy. In this place you should drag and drop only this streamers, which should affect loading screen. If you have more than one, grow up size and drag and drop all of them.

- **Wait frame is a** very important value in seconds, it should be higher than 1-2. This is the time that loading screen system need have to get info about how many virtual grid elements(scenes) from whole streamers will affect loading screen. If it will be too short loading screen could end before all virtual grid elements(scenes) are loaded, but if it will be to long you will simply waste time. Probably it should be about 2-3 seconds.

- **Progress Img**, here you drag and drop your image for progress bar if this window haven't been filed automatically or you want to change it.

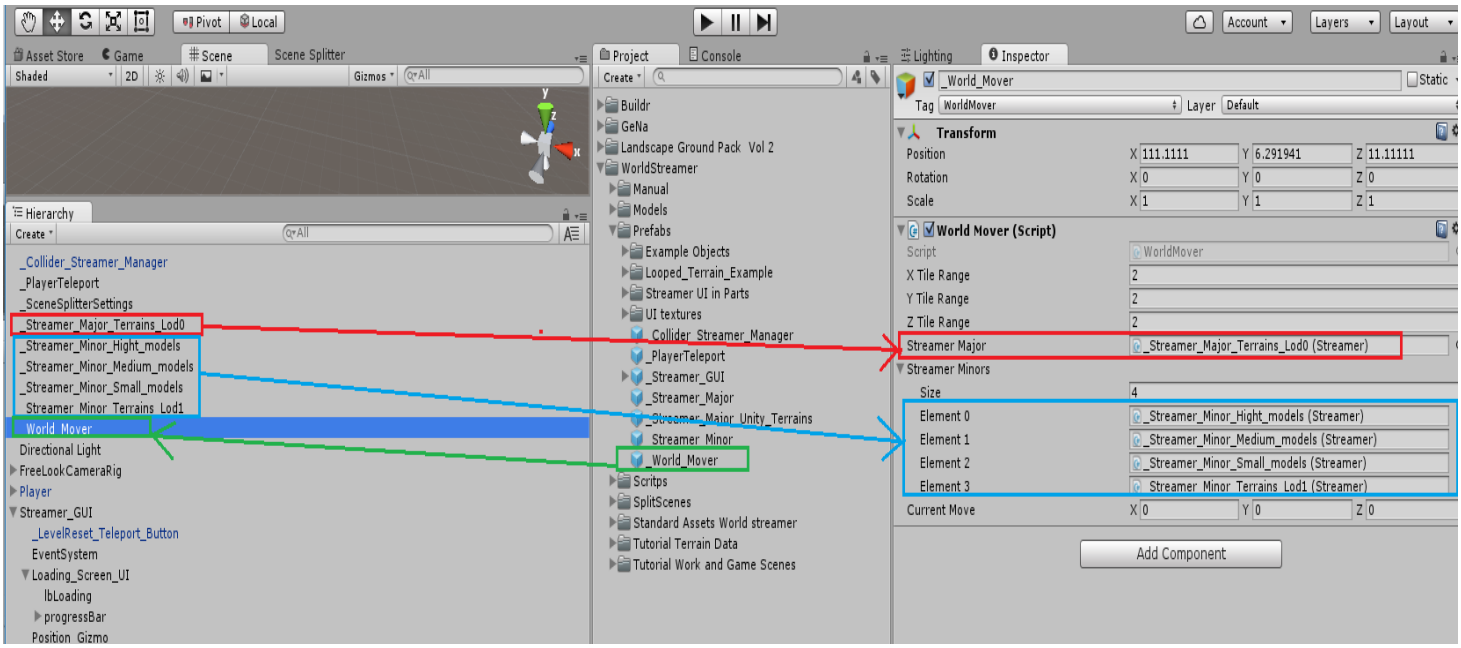
4.5 Floating Point Fix system

If your game area is too big or you use looping system, you should use the floating point fix system. Basically if coordinates are too big in engine, physics and every unity system that is based on object position begins to lose accuracy in calculations. This may create strange situations in physics and create graphical bugs like flickering shadows. That's why we created system which removes this problem. World Mover will restart your world's position from time to time. While we use word "time" we mean a distance that is represented by virtual grid elements(scenes) interval. We have to use tile count/interval instead of xyz distance because it's easy and clear for whole system when restart should happen. You shouldn't restart world position too often because it's expensive process for most games. Floating Point Fix system also have solutions that support multiplayer games so check "current move" position.

Floating Point Fix system and Looping do not support:

- Static batching, because we are not able to move these objects at the moment - unity blocks that. You could use custom batching system or batch your objects by combining them manually. You could use tools that are available at asset store. Your unity terrain should have static batching checked on, we are still able to move batched with static batching.

To add floating point fix system to your game simply drag and drop "_World_Mover" Prefab into your scene and fill it with all necessary information. If you use teleport/respawn system check "Important1" information below. If your camera is separated from the player or there are things that should have position reset together with the player like: camera, AI at single player, weather or sky horizon objects (which are not streamed by streamers, but need to hold them in relation to player position) please check "Important2" information which is below.



- **XYZ Tile Range**, it's used to set amount of virtual grid elements after which system will reset your coordinates. For example if tile range is "1"and major streamer has got each virtual grid element size 200x200 units, streamer will set 0,0,0 position to your player and to whole world after 2 virtual grid elements (we are counting from 0). This means after you move 400 units. So after each 400 units system will restart whole world position.

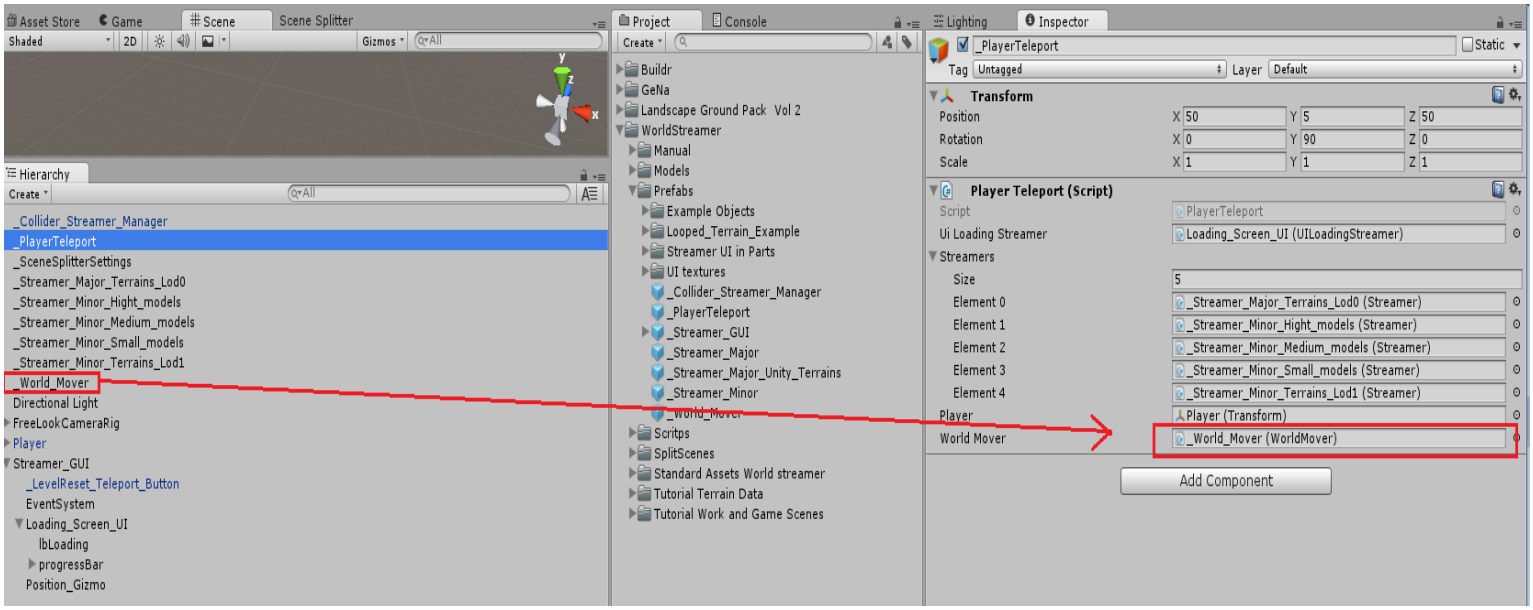
- **Streamer Main**, in this place you have to choose which streamer will decide how often system will reset your world position and which virtual grid (from which streamer) will decide when this will happen. We advice to use the streamer with the biggest virtual grid elements or if you stream unity terrain, you should use streamer with terrains.

- **Streamers Additional**, you should put here all the remaining streamers from your scene hierarchy, which should be affected by floating point system. In 99% cases you should drag and drop all of them.

- **Current move** is a difference between player's real and local position that is useful for correcting RPC with server communication or correct objects spawned from code. Local player position - current move = real player position. Local player position is the actual player game object position, it has been reset few times so that's why we call it as local.

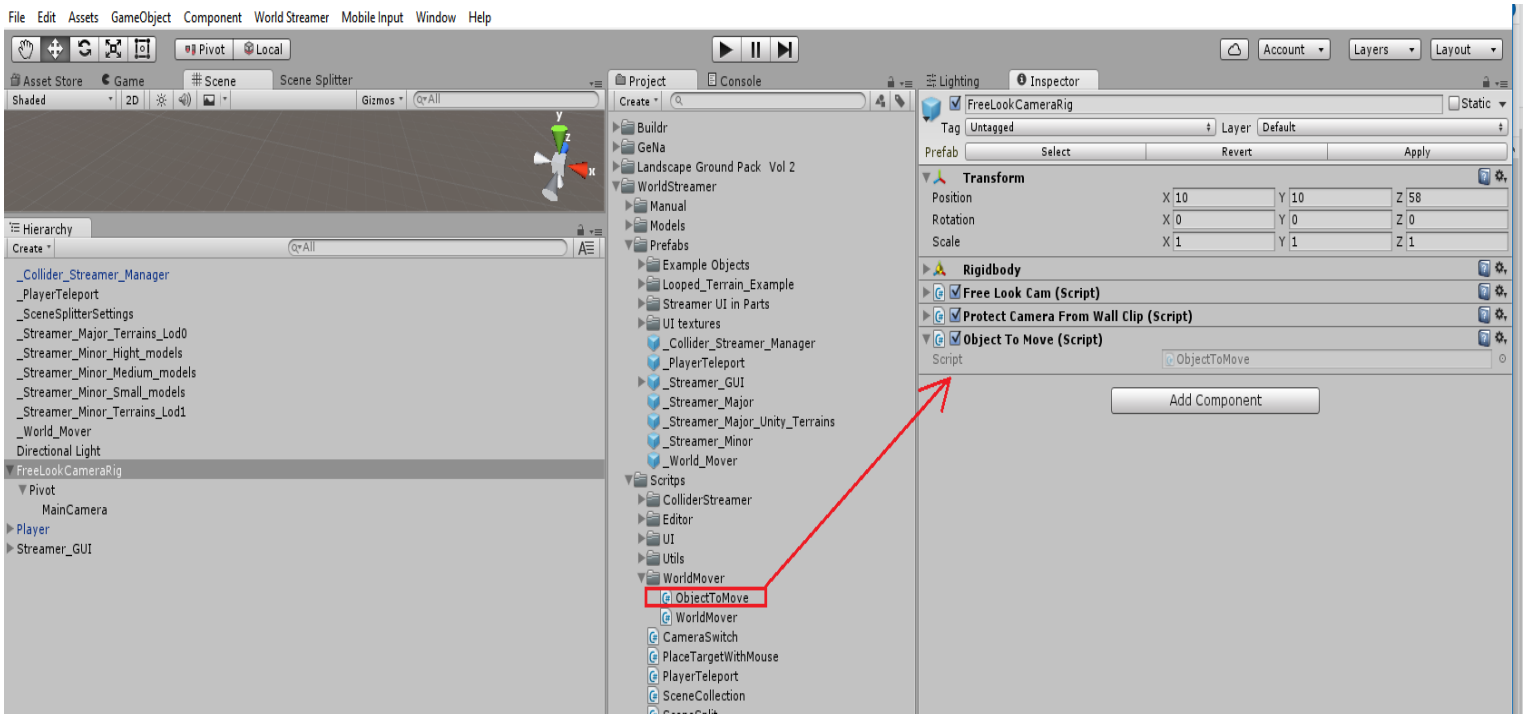
Important1!

If you use teleport/respawn system you have to connect "_World_Mover" object at your scene hierarchy to "_PlayerTeleport" prefab also at your scene hierarchy. Simply drag "_World_Mover" object from your hierarchy and drop into "_PlayerTeleport" script to "World mover" window.



Important2!

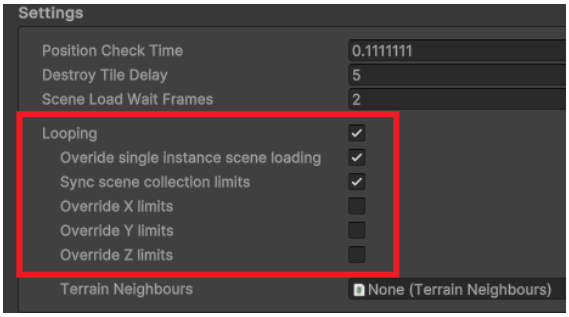
For objects that follow player or stay in important relation with the player or world we have "Object to move" script. Simply attach script to the objects that should follow player/world during position reset so that the relation between player and these objects will stay untouched. If you have particles that follow player or objects, please change their simulation space to local, or if this is not possible change that only for the moment of position reset. If you will not change this, particles will have small break because they need to spawn in new place.



4.6 How to loop your whole world or specific object/models

World streamer gives ability to loop your world or imitate planet. That means, world could be endless Loop could be also used for repeatable things like asteroids in space.

To loop your world simply check the checkbox "looping" at your streamers in scene hierarchy. You should use floating point system for your looped world, so you also have to do everything that is in "4.5 Floating Point Fix system" section.



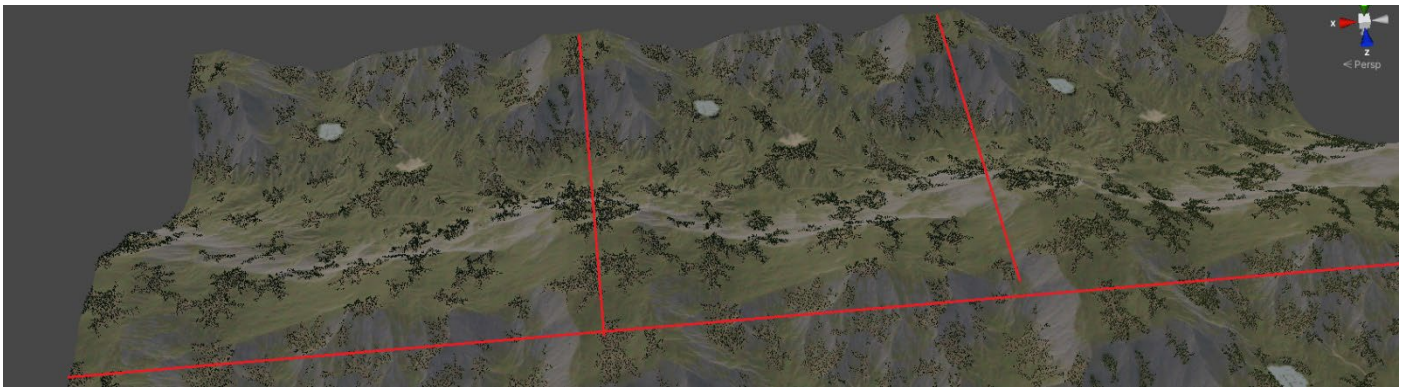
Looping checkbox enables looping for this scene collection, which creates endless world. Each streamer with its scene collection is looped independently, which means if you want to synchronize them they must have the same world size or you simply have to check checkbox to synchronize I them. You also can force world looping via distance by Override X, Y, Z limits for example in case there is no data in specific axis.

One layer could be looped in different rage from others for example if you uncheck sync scene collection limits town will be looped more often then whole world.

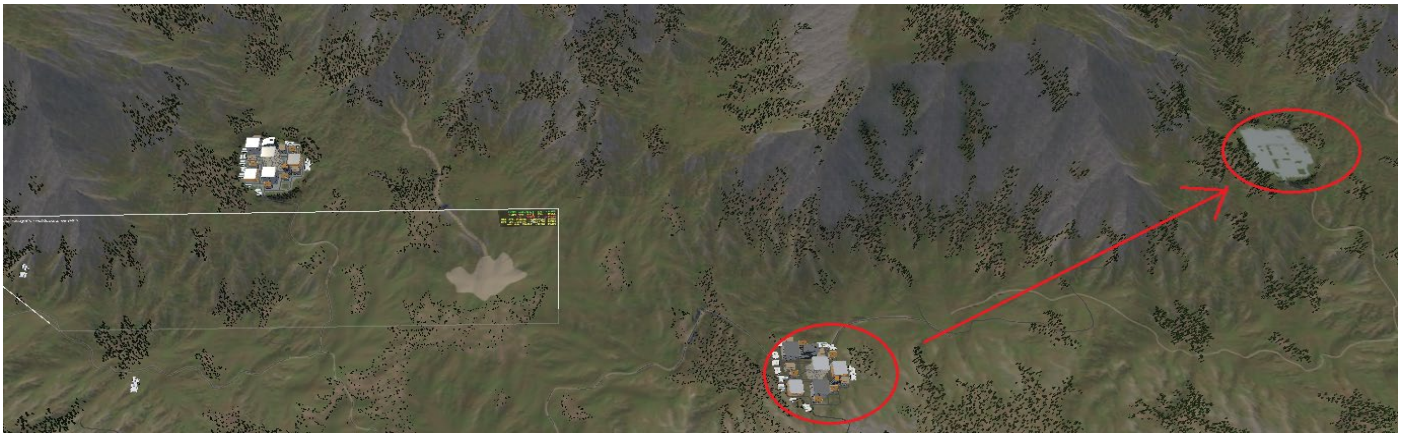
As we move world objects during looping, our loop system has the same limitations as Floating Point Fix system. Floating Point Fix system and Looping **do not support**:

- Static batching, because we are not able to move these objects at the moment - unity blocks that. You could use custom batching system or batch your objects by combining them manually. You could use tools that are available at asset store. Your unity terrain should have static batching checked on, we are still able to move these batched terrains.

Here is an image that shows how looping system idea works:



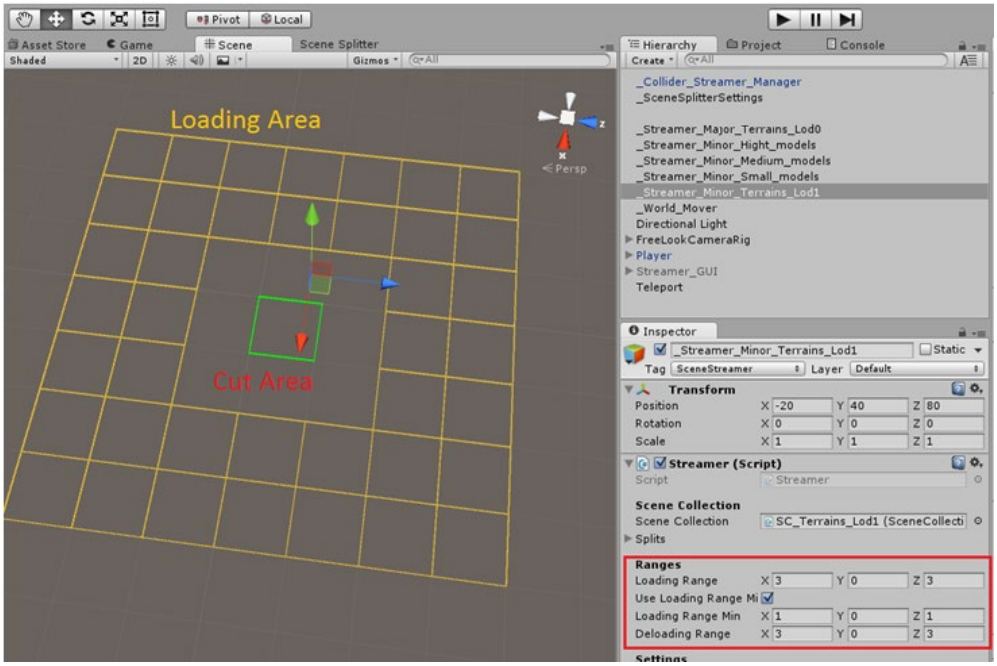
No sync scene collection limits – town is desynced from other layers, but for example for asteroids and space game this could be good idea to loop them more often then other parts of the world.



4.7 Ring streaming idea

This system allows you to load objects in specific range "from - to". We called it ring streaming because streaming area looks like ring. If objects are too close they will be removed, if they will be too far they also will be removed. Ring streaming could be useful to replace objects at close or far distance. If 1st streamer will stream layer with high quality detailed objects/terrains only at short distance, then 2nd streamer could stream low quality objects/terrains at far distance only. Low quality objects/terrains will disappear in place where high quality objects/terrains will be loaded. You will get far view with extremely good performance. This is very good solution if object - like unity terrain doesn't support LOD system and its data (regardless of visibility) is very heavy to load

To turn on "Ring streaming" simply check "Use Loading Range Min" and “check sync with other layer” at your streamer object in hierarchy and fill "Loading Range Min values". Chose also layer that you want to sync with.

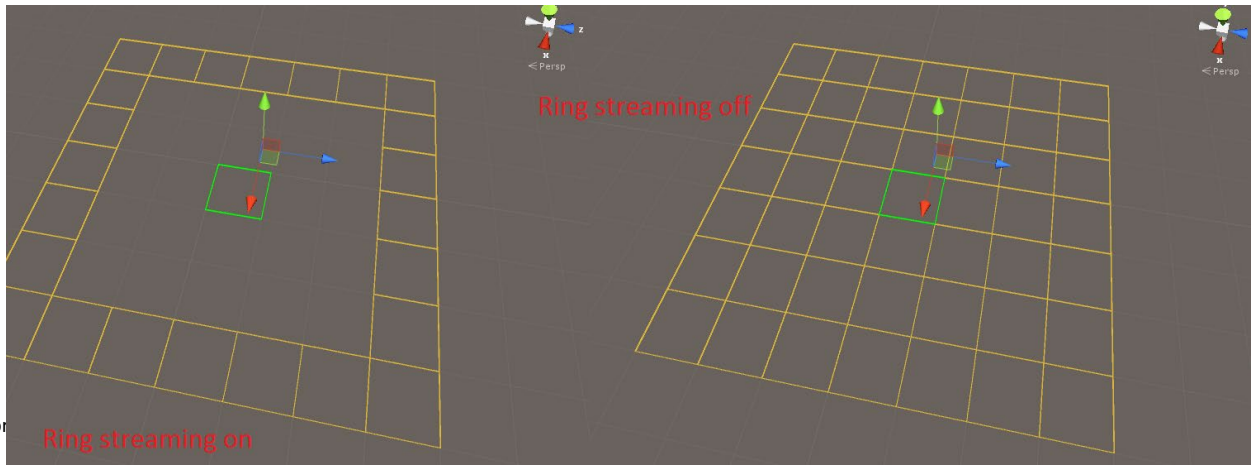


- **Use Loading Range Min** checkbox allows you to use "Loading Range Min" which we called - Ring streaming.

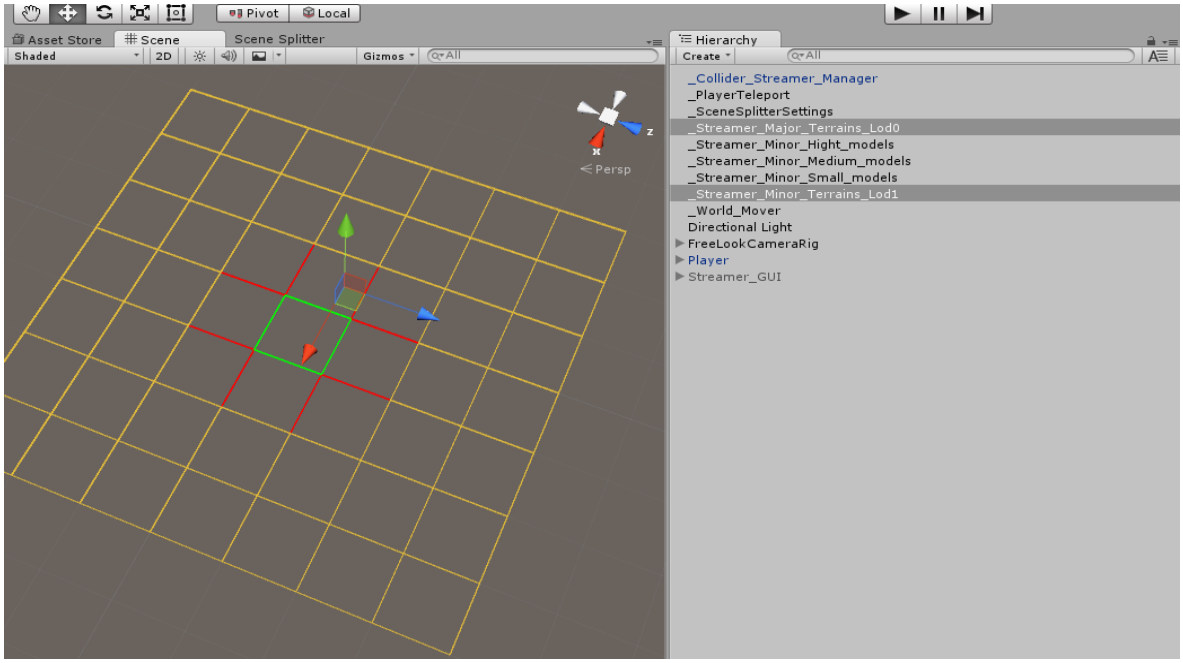
- **Sync with other layer** allows you to sync streaming between2 layer so data will not going to be unloaded until synced layer will be ready to replace it.

- **Loading Range Min** (Ring Streaming) is an integer value in each axis that precise virtual grid elements radius that you want to cut out from the loading range. Streamer will load everything in the ring. If you will be close enough it will remove or doesn't load objects from memory. It's useful to replace objects in close distance by other layers. When you click on Streamer object you could see gizmo at your screen with current loading range area, like you see at the image above..

Here is streamer gizmo from ring streaming ON and OFF:



Here is streamer gizmo, from 2 streamers which complement each other. On first ring streaming is ON and it streams low quality terrains Lod1 (yellow), on second is OFF and it stream high quality terrains Lod0 (red):



4.8 Physics manager

If scene or streamed world contains physical object and you want to defend from falling or moving because of removed/unloaded environment simply add "**Physic Culling System"** script to each physical object. System will hold and save speed/direction vector when player will be to far. If player will back to specified area speed/direction vector will be read and object will behave natural. **Physic Distance** - after crossing this distance object will be frozen. In our prefabs catalog you could find "_PhysicCulling_Test" prefab which will help you to understand how it works.

4.9 Terrain culling system

This system will hide unity terrain objects, when they will be not visible at camera view or they will be too far from player. "**Terrain Culling System**" script must be added to each terrain chunk. It's good to add it massively during split process. System react with main game camera so terrain could be invisible at scene view. It save CPU usage because invisible tree, terrain parts simply are not refreshed with their trees and LODs. System will detect main camera via **Tag "MainCamera"** and use it's view to cull or show terrain object.

4.10 Player in safe place during data loading

If you want to move player to safe place until loading/teleport will end, you have drag and drop our "_Safe_Place_Spawn_and_Teleport" prefab in to your gameplay scene hierarchy. This system also support spawned player. You have to fill:

- "**Streamers**" list by streamers from gameplay scene which should affect safe place system.
- "**Safe Position**" is an object with position where player will be moved until system will read data selected by streamers.
- **"Player"** you have to drag and drop your player here which should be moved. If player is spawned system will take over of this so simply leave this place as empty. Immediately after spawn it will be wait for game start at safe place.

If you want to use this system during teleport you also have to fill "player mover" window at "_PlayerTeleport" object by "_Safe_Place_Spawn_and_Teleport" object from scene hierarchy.

5. Terrain Streaming

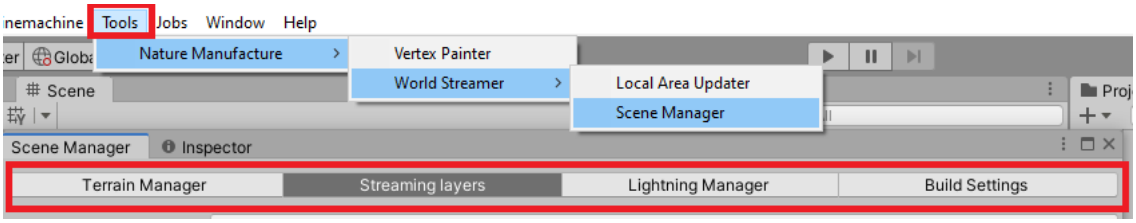
First of all, there are few things that need to be said before you start to stream your terrain. If you want to stream unity terrain, it must be in parts. Each unity terrain part generates static CPU usage and dynamic when you load next tile. All of this performance eaters could be solved or reduced. Information about how to solve/reduce this, you will find in "5.5 Huge amount of details at unity terrain and performance solutions" section. Terrain in our system could be looped and replaced by low poly meshes in far distance. Read carefully whole info that we give you in "5. Terrain Streaming" section, it could save you a lot of time and guarantee extremely good performance in relation to any other available solutions.

5.1 Terrain preparation - unity terrain, low poly mesh generartion

To stream your terrains you have prepare them for this process.

1. **Idea**

   When you are starting streaming unity terrains you need terrains in parts/chunks.
   In our tools -> nature manufacture -> world streamer -> scene manager -> terrain manager section you could cut, estimate size of your terrain and generate low poly versions for far distance.



   Remember to leave your source terrain untouched because you could change your mind about virtual grid element size and check how it works on bigger or smaller chunks!

   You could generate your world already in tiles by available terrain systems at unity and process it by our tools to get for example low poly meshes for far distance.

   Developers always ask how big this terrain chunks should be? Basically it's your choice we could give you a few tips, depending on way that you chose.

   (a) If you don't want to use ring streaming and low poly mesh at far distance instead of unity terrain, your each terrain tile should be size 110% of your camera far view. This will guarantee low static cpu usage, because you will set loading range to 1 and only 9 terrain tiles will be loaded. Such big tiles also mean quite big memory consumption but not tragic and quite big CPU consumption from culling (many tree objects). We advise size 110% of camera far view because nobody wants to see how terrains are loading (appear). This could happened from time to time. With 110% you leave this 10% of "time" for loading process.

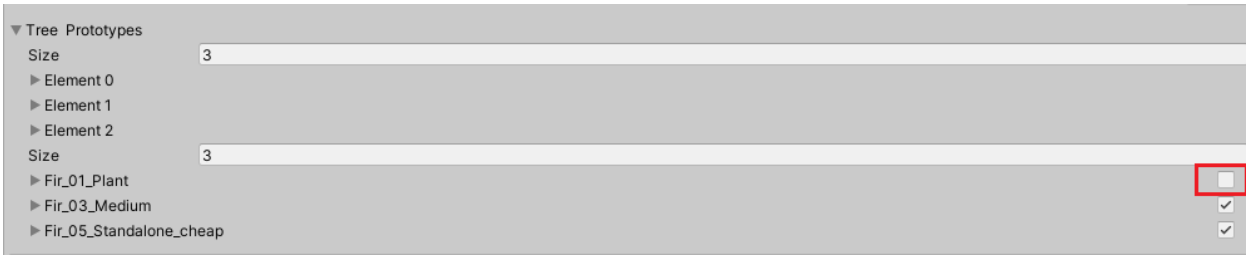
   (b) If you use ring streaming to replace far terrains by low poly mesh which is much, much better solution, your terrain chunks/parts should be 10-50% of camera far view. 10% is good if you want to have control on terrain details by in game options and low poly meshes contain trees like with our example. 30-50% is good if you just want to leave terrains like they are. Advantages of this solution is that you have very far view and you spent only few verts/tris and 1 drawcall per each low poly mesh as terrain. You could handle many terrain tiles because many of them are simple meshes and unity terrains are small, they hold smaller amount of data and their refreshing time is short. By Streamer

loading range and loading range min you could adjust when your terrain will become low poly mesh. This is dynamic option so could be changed during gameplay. Player could choose the level of details.

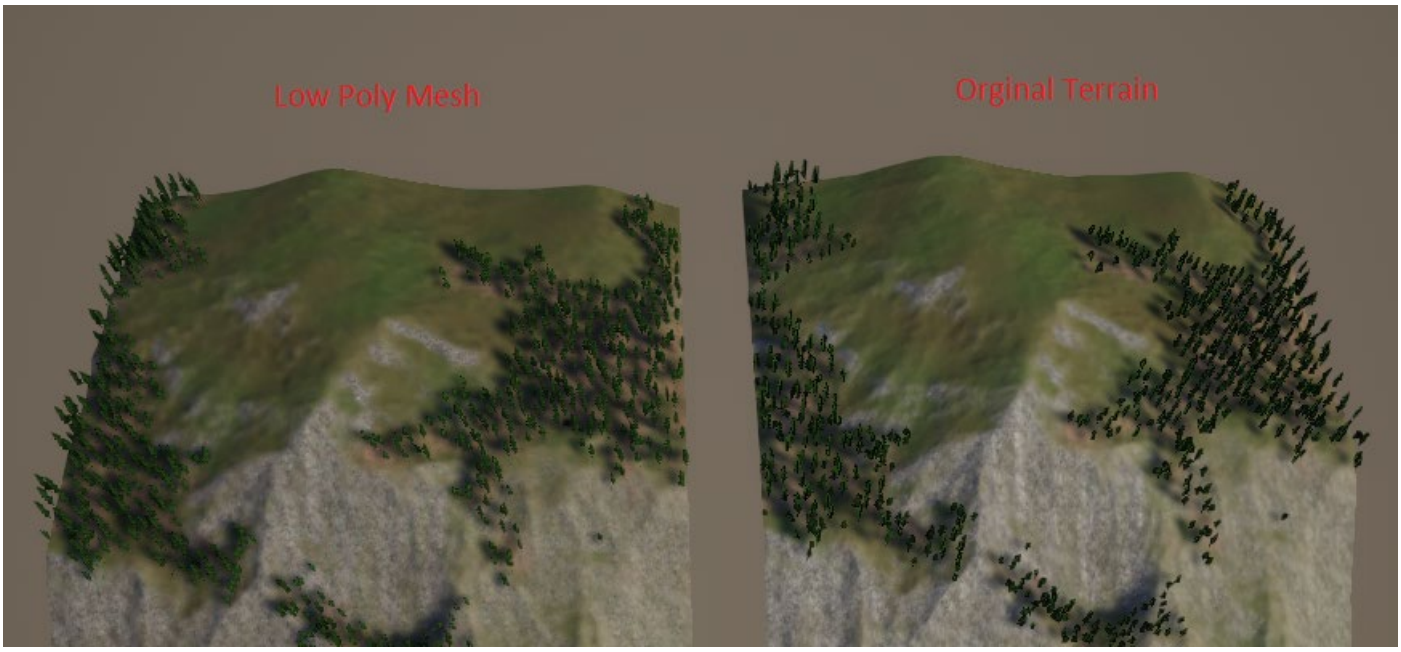
(c) If you use only low poly mesh instead of unity terrains (probably mobile games), you could use any size but remember to engage number of tiles with camera far view. It's good to hold amount and size of loaded tiles in relation 110% of camera far view, only because of terrain appears/show on. You left this 10% of "time" for loading new tiles.

Custom terrain solutions are always difficult because we don't test all of them so probably it will be combination of "b" and "c" point in this section.

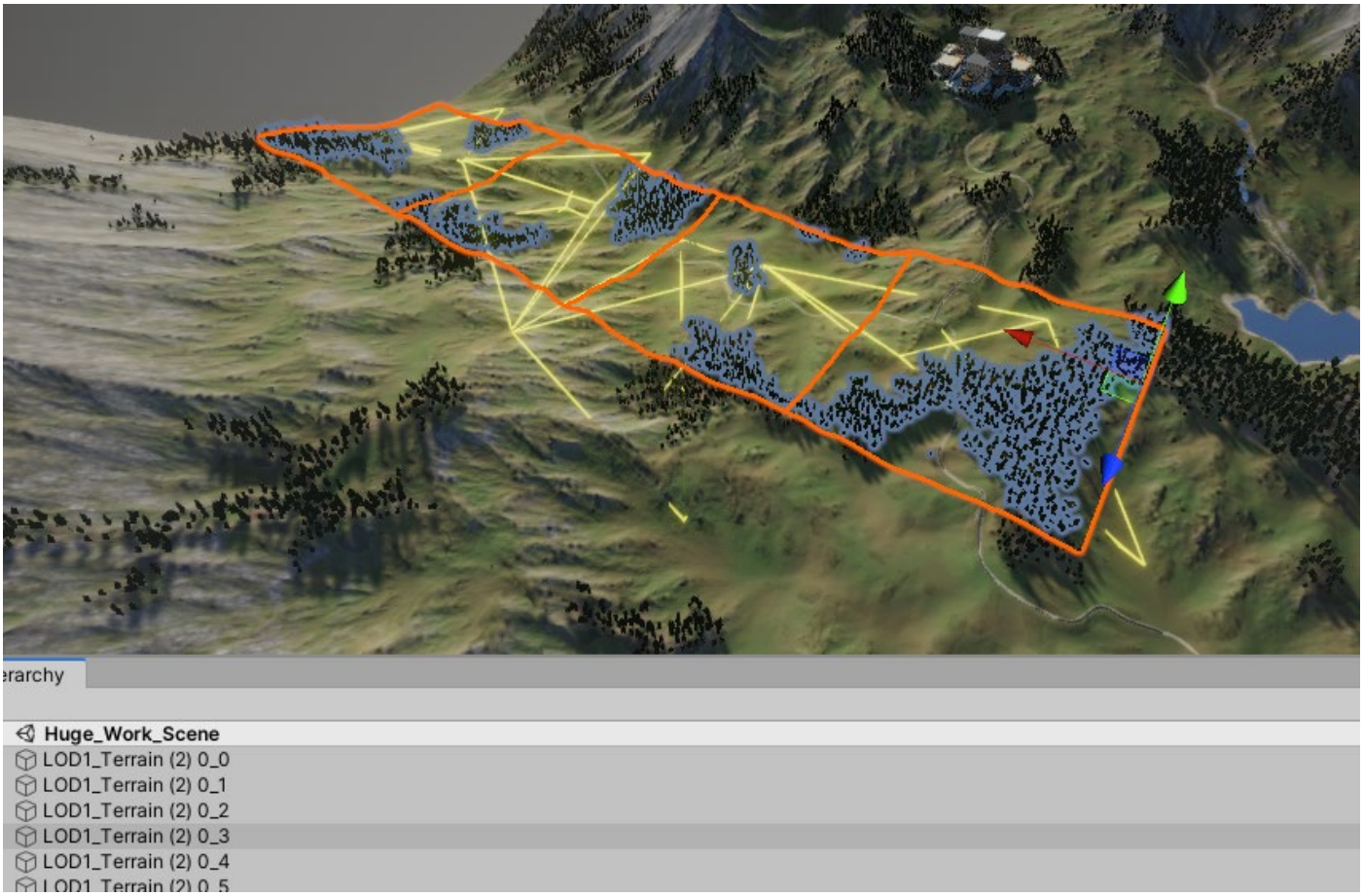
2. Terrain Preparation
    1. To prepare terrain from streaming there are few steps. Fist step is to open Terrain Manager
    2. Set Options that terrain in parts after split should have so.. draw instanced, pixel error, settings for grass and trees, wind.
    3. Chose amount of splits on your terrain/terrains. 2 means that terrain will by split by 2 in x and z axis so you will get 4 tiles, etc. Click split and our system will split terrain into parts.
    4. If you want to get best performance you should generate low poly meshes from terrain and trees. They will replace in ring streaming heavy terrain data without visible impact. It will dramatically reduce used drawcalls and triangles. You could set few thing here.
        - **Terrain LOD =3** because we do not need more dense shape for LP terrains, at value = 4 probably it will lose too much shape.
        - **Basemap** resolution is set to 512 as higher texture resolution for far distance is rather useless and only takes memory. Probably 1024 would be a reasonable limit as well.
        - **Normal details and strength** we estimate for unity terrain but you always could play with this values.
        - If you use NM trees or trees that have cross model as last LOD (NOT billboard) you can chose bigger trees and bake them with terrain as well. Rather avoid to add there small trees as they become invisible very fast and they are not worth of putting in far distance. This will avoid situation when tree disappear because of LOD cull distance and it back in low poly mesh again. Do not add unnecessary data in far distance. Simply turn of checkbox for smaller trees.



After you generate low poly meshes you should get huge matrix of low poly meshes and unity terrains in scene. Low poly meshes if they use our trees they should contain crosses on surface. Here is result of such operation:



3. We can move to next step when your terrains are in parts



4. First terrain should have a starting position XYZ from full rounded position like : 5000, -5000, 0, 1000, 500, -100 etc not like 99.98 , 0.1, -0.018. This kind of wrong positions, could effect in unexpected terrain offsets in virtual grid elements. Remember virtual grid elements starts from 0, 0, 0 and unity terrain pivot is in it's corner, this sometimes could effect in offset between terrain and virtual grid borders. You could simply remove this offset by placing terrains on positions which are multiples of their size. For example if terrain tile size is 500x500, virtual grid element size is 500x500 ,then terrain objects must lie on positions like 500, 0, -500, 1000,-1000 and other multiples of 500. This will synchronize virtual grid elements with terrains.

5. Rename your terrain objects by adding word "LOD0_Terrain at their names beginning or "LOD1_Terrain" for low poly terrain mesh. Of course this could be any word , we give just an example. This word will be used during splitting/segregation process which put's terrains in correct virtual grid elements and layers. You always could use “Object Parent” script and hold terrains in parent game objects, then you do not need any name convention and prefix on objects.

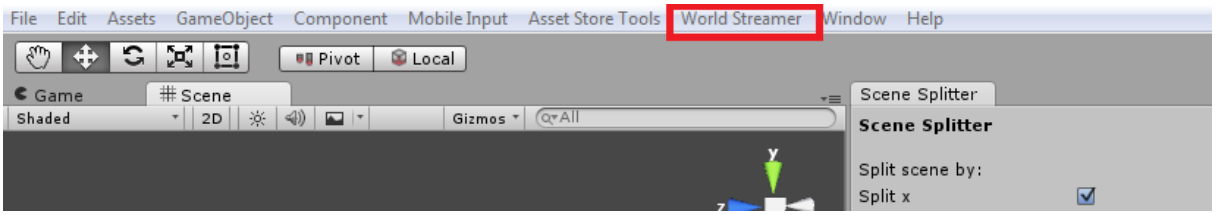
6. Create prefabs from these terrains, **this is not necessary** but it gives access to them without opening the scene so it's useful. You could create prefabs when you will be sure that terrain parts size are fit for your game. Probably you will be sure of them after you will see how streaming goes.

5.2 Scene split and virtual grid setup

We are going to put terrains into correct virtual grid element automatically. Now you should follow this steps:

1. Open empty scene and save it, then you should copy your terrains here. You could call this scene as "Work" scene you also could work on current scene.

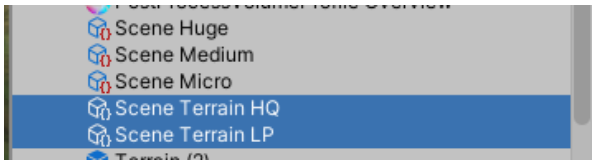
2. Open "Scene Manager" window by clicking at World Streamer =>Scene Manager



3. Fill scene Splitter window:
   - Click add layer button
   - Check "Split X" and "Split Z" don't check "Split Y" axis if it is not so important for streaming (like in Space Games). If you don't want to remove terrains because of player Y position don't check "Split Y" axis.
   - Fill X,Z, optional Y size. Size MUST be equal to terrain size. If your terrain is 200 width, 300 length please fill "X size" with 200 value and "Z size" by 300
   - Fill GameObject Prefix by word that we add to our terrains names, so "LOD0_Terrain". Splitter will check all objects at the scene and if any object will contain "LOD0_Terrain" at the beginning of the name, splitter will use it in this layer. If you use object parent and parent objects use same prefix as object parent script have inside.
   - Fill Scene Prefix (layer name) by name that will be clear for you, it could be: " Scene Terrain HQ " or you also could add info about which world it belongs to, if you got more than 1 world. So name with this info will look like: "World_1_ Scene Terrain HQ ".
   - Do similar operation for low poly terrains if you generate them in our example we used "LOD1_Terrain".

4. If you are sure about everything that you filled in Splitter window, you could click "S" button or "Split Scene". You could undo that operation by "C" button or "Clear Scene Split". System during split will prepare virtual grid elements/future scenes from your chosen assets. For more info about this section "4.1 Scene Splitter"
Do the same operation for low poly terrains if you build them, just create layer for them.

5. If you are sure about virtual grid elements that system made, click "Generate Scenes from virtual Grid". This is the last thing that you have to do with your assets. System will start generating scenes from virtual grid elements. It will create scenes in specified directory or in default one. System will also create scene collection prefab called same as you chose scene prefix. In our example we will get scene collection prefab called: "Scene Terrain HQ " or "Sene Terrain LP". This file contains all fresh info about your scenes (virtual grid elements): how many are they, how they were split, when they start and end (world borders). This important scene collection prefab called will be connected by us to streamer objects at your "Gameplay" scene. It allows you to stream this whole content in proper time and place.



5.3 Streamer setup at your game scene

After everything in "5.2 Scene split and virtual grid setup" section is done and we have our scene collection or collections (if we made the same with low poly terrain but with other scene and object prefix), we could start to setup our "Gameplay" scene. For start we should open/create empty scene with player, camera and directional light and follow these steps:

1. Drag and drop into your "Gameplay" scene "_Streamer" prefab.

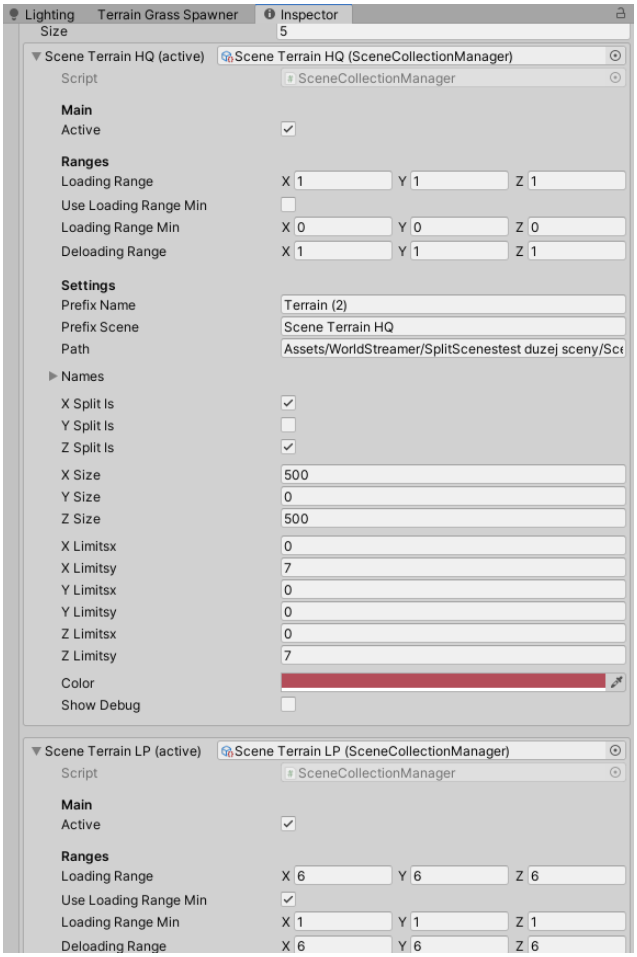
2. Drag and drop into your game scene "_Streamer_GUI" prefab. This prefab contains graphical UI, Loading Screen, Teleport/Respawn and gizmo "Position" viewer for floating point fix or looping system.

3. Click on Streamer prefab at your scene hierarchy. Drag and drop your scene collections, that you made in "5.2 Scene split and virtual grid setup" section into "Scene Collection" window. In our example it was: "Scene Terrain HQ" and " Scene Terrain LP"
4. By this moves you connected your generated scenes with streamer.

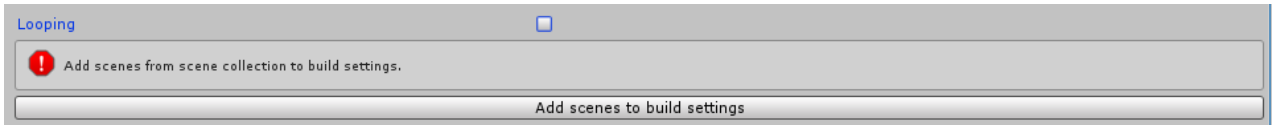
5. Fill streamer settings according to "4.2 Streamer objects" section. For this manual example we will set:

- Loading range to X= 1, Y = 0, Z = 1
- Use Loading Range Min = false
- Deloading Range X= 1, Y = 0, Z =1
- Destroy Tile delay = 2 second
- Position check time = 0.1 seconds
- Max Parallel Scene Loading = 1
- Scene Loading Wait frames = 2 frames

  Summing that means we will load area 3x3 of terrains parts/virtual grid elements/scenes and unload every virtual grid element that is further than this. We will check our position every 0.1 second. Streamer will load max 1 terrain part during one frame and we left 2 frames to finish asynchronous stream operation after each virtual grid element was loaded and before we load next one.
- We fill "Player" window by drag and drop our player from scene hierarchy into "Player" window. Our system will follow that player and load terrain tiles if they will be close enough.
- For low poly meshes if you build them we set loading range min =1 (ring streaming) and loading range , deloading range = 6.



6. Now we will set up our UI if we want to use loading screens.
   - Drag and drop "_Streamer_GUI" or "_Loading_Screen_UI" from project into scene hierarchy.
   - We click at "_Loading_Screen_UI" object (child of "Streamer_GUI") in scene hierarchy and we write Size =1 or more (if we have more streamers) in Streamer value.
   - Now we need to drag and drop our "_Streamer" from scene hierarchy as element 0 in "UI Loading Streamer Script". Element 0 holds high quality terrains. Loading bar in loading screen will release player into game only when all scenes from element 0 will be loaded.
7. We click at "_Streamer" game object and if we didn't added our scenes from scene collections to the build setting, we should see a warning with button. Click on it and system will automatically add missed scenes to the build settings. Each time you change number and size of virtual grids elements you have to click that button!



8. **Click play and it works!**

Important1.

Now you could easily split/unsplit/add new terrains/remove content/move content and you could refresh layer with terrains and streamed world in few seconds by clicking split and then generate again at your work scene or by our "local area updater". System will refresh your scenes. You only need to refresh "_Streamer" prefabs by clicking add to build settings button. That means you could easily test many configurations!

Important2.

When you modify your terrains, by adding plants, changing shape, painting textures you don't have to split again your scene and refresh your streamers. Terrains are saved at scene and they hold reference to your terrain data. That mean everything will be actual all the time.

5.4 Huge amount of details at Unity Terrain and performance solutions

It is possible to stream and hold terrains with huge amount of details without big CPU usage!

- You could use ring streaming, low poly terrains could even contain trees baked. Far distance foliage and terrain with such config take only few batches and triangles. Check our huge world demo.
- Turn of terrains rendering that are not visible, by your own or our script "TerrainCullingSystem" Simply drag and drop it on each terrain prefab/object and set max visibility distance for this terrain.
- Both solutions, ring streaming and culling by script.
- Vegetation Studio or Vegetation Studio Pro – removes whole foliage and move it into instanced indirect. So terrain only contain heightmap and textures. It also support our custom shaders which we made in our other products.
  https://assetstore.unity.com/packages/tools/terrain/vegetation-studio-103389?aid=1011l7vcu
  https://assetstore.unity.com/packages/tools/terrain/vegetation-studio-pro-131835?aid=1011l7vcu

This tips reduce static terrain CPU and memory usage a lot! By using this solution you could render your terrains in very far view even at mobiles!

If you left low amount of objects (detailed terrains) at the scene, some scripts could work faster. When you call garbage collector, his spike will be reduced to value that you could skip. If you want to remove objects from memory you have to call GC (garbage collector) from time to time. We call it in our system, you always could change calling density in our code.

If texture streaming is slow/generates spikes, please adjust "Async Upload Time Slice" and "Async Upload Buffer Size" in Quality settings.

Remember to use incremental GC in player settings, this increase GC a lot.

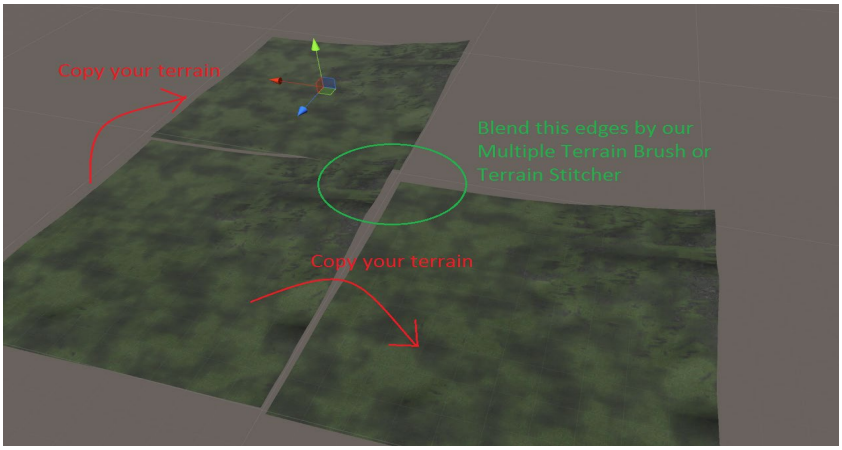
So too summing up, you could use unity terrains for close distance and turn off terrains that are not visible at your camera (smart culling script). For far distance you could use low poly meshes with baked trees, custom tree system or without trees at all. In this setting you could remove millions of vertexes and thousands of drawcalls. Your loading will be smooth, it will not affect players CPU. Memory usage will be very low probably you will cut few hundreds of MB that you could spent on textures and models. Isn't that promising? It is and this is how console and MMO, AAA games works and handle terrains and models.

5.5 Create looped terrain

Probably small amount of developers want to create endless looped world with terrains, something like a planet. Wherever you go you will eventually come back to the same point

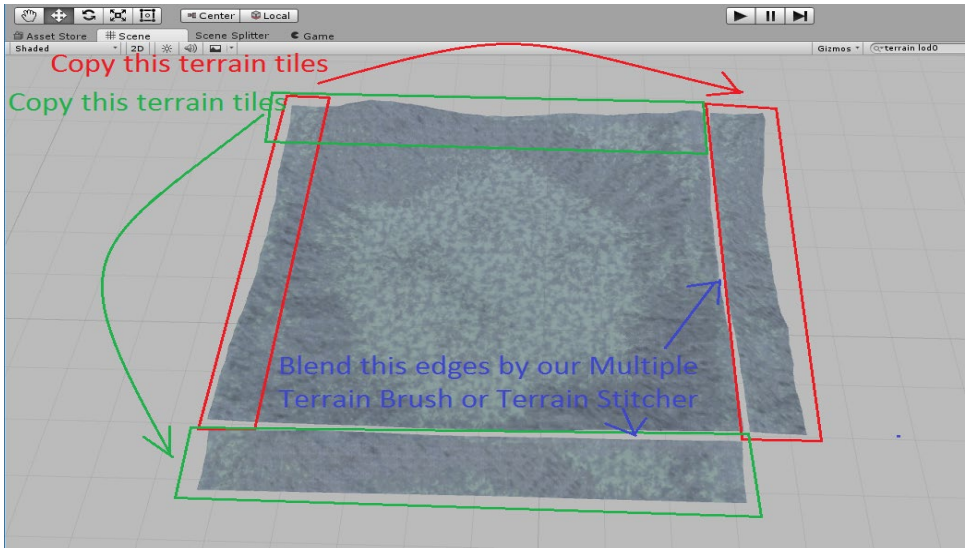
There are two moments when you could loop your terrains:

1. You have your terrains in one chunk, you didn't split it yet. Probably it's easiest way.
    - Simply copy and move your terrain like in this image (yes you to need blend only 2 borders):



    Borders must be in contact. That means if terrain ends at position 200, next (copied) terrain must start from position 200.
    - Use unity terrain brush or Terrain Stitcher and simply blend your terrain edges by this tools.
    - When you finish, remove additional terrains (which were copied only to have perfect terrain borders) and terrain is ready!
    - Do all points from "5.1 Terrain preparation", "5.2 Scene split and virtual grid setup" sections. At "5.3 Streamer setup at your scene" section, the only change is that you have to check checkbox "looping" at your "Streamer" in gameplay scene hierarchy.
    - Probably you should use floating point fix system. You have to use it, because while player will move over your world all the time in one direction, his position XYZ will still grow. If position will be too big physics, shadows will stop working properly. Please check "4.5 Floating Point Fix System" section to add this. It's easy and it will only take you few minutes/seconds to setup.
    - Done, you have your world looped, hit play and test!

2. You have your terrains in many parts
    - Simply copy and move your border terrains like in this image:



Borders must be in contact. That means if last original terrain ends at position 200, next (copied) terrain tile must start from position 200.

- Use unity terrain tools or Terrain Stitcher and simply blend your terrain edges by our tools
- When you finish, remove all additional terrain tiles (which were copied only to have perfect terrain borders)
- Your terrain data at these terrains have been actualized.
- Do all points from this list, but only if you haven't done theme before:
  - "5.1 Terrain preparation" Probably you have done it before, because you have your terrains in parts.
  - "5.2 Scene split and virtual grid setup" , if you haven't split your terrain tiles before.
  - "5.3 Streamer setup at your scene" section, **the only change is** that you have to check checkbox "looping" at your "Streamer" in "Gameplay" scene hierarchy.
- Probably you should use floating point fix system. You should use it because while player will move over your world all the time in one direction, his position XYZ will still grow. If position will be too big physics, shadows will stop working properly. Please check "4.5 Floating Point Fix System" section to add floating point fix system. It's easy and it will takes you only few minutes/seconds to setup.
- Done, you have your world looped, hit play and test!.

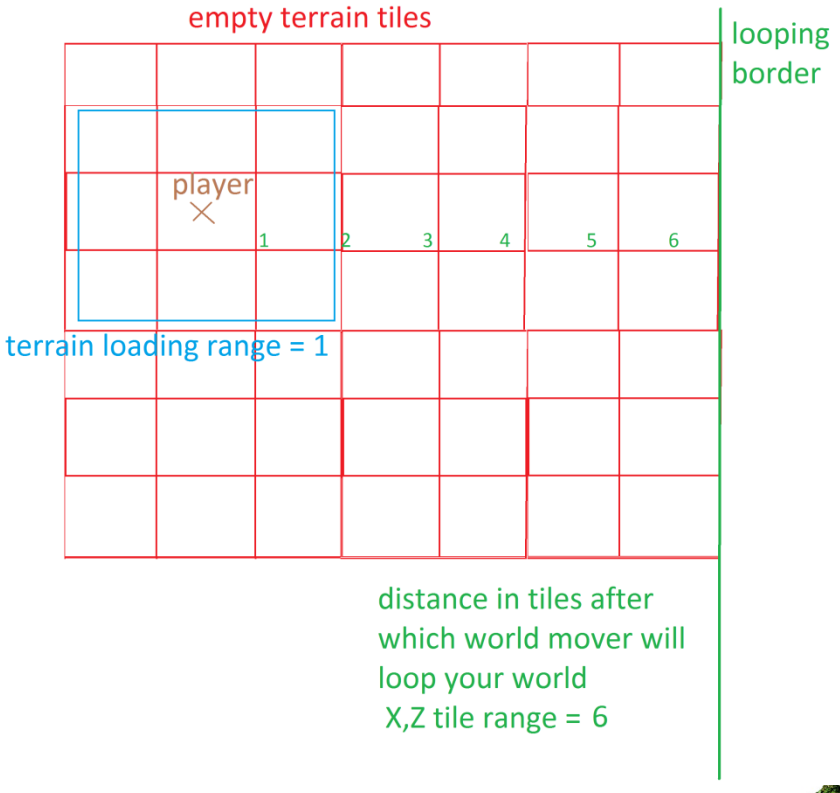
5.6 Real-time generated terrains support

There is possibility to stream real-time endless generated terrains. We will discuss this very short, here is the idea of how to handle this and use world streamer. We discussed this idea at forum and we would like to share the result.

1. You have to create big group of empty terrains. You must generate 2,5x more terrain tiles then you want have loaded at one time, this because of looping system. Place them as grid and split with proper settings. Check sections: "5.1 Terrain preparation (unity terrain, low poly mesh, custom solutions)" and "5.2 Scene split and virtual grid setup".

   Here is a concept example:
   a) Looping will guarantee unlimited amount of generated tiles. That's why your realtime generated world will be endless and you will get unlimited tile number to fill by your generated data,

   b) You have to generate 7x7 empty terrain tiles, because looped world must be:
   2 * loading range +1 bigger, then loading range otherwise you will get a warning about bad settings.



2. Then you have to generate scenes, check: "5.3 Streamer

setup at your game scene" section.

3. Check looping at streamer prefab, you could also use floating point fix system which optional.

4. When your blank terrain streaming is ready, you have to modify our world streamer script or attach your own script to blank terrains (probably easiest way). You have to fill/replace every blank terrain by your data generated from runtime generator. This will work properly.

Of course you could create your own idea how to merge terrain streaming with real-time generation. If you have it, you could share your idea with us, so we could discuss it.

6. Model/Objects Streaming

Models and objects streaming is the main feature of World Streamer. With our solution you are able to read from the disc your objects that have been placed in your world and split into virtual grid and layers by our splitting tool. Our splitting system will easily refresh, grow, reduce your world library (virtual grid elements/scenes). You can put huge amount of different objects like: lights, particles, models, colliders, reflection probes and everything will be read from the disc in the proper moment. We highly advice you to stream objects/models in layers. This give you ability to get better performance and optimize some unity functions that are not supported by unity LODs system. Of course everything is compatible with unity LOD system but we offer much more. If you left low amount of objects at the scene some scripts could work faster. When you call garbage collector, its spike will be reduced to value that you could skip. If you want to remove objects from memory you have to call GC (garbage collector) from time to time.

Optimization which we are talking about are specially touching particles and lights. As you had seen or not, if you put Lights or Particles at unity LODs system, when become culled but:

- Lights are still active and give light and grow drawcalls at objects that they touch.
- Particles are still active at CPU. If you check profiler your particles will still take CPU because of update and generating process

All this CPU,RAM,VRAM,GPU usage could be remove by world streamer. Light and particles will only use your sources when they become close enough to be visible.

If texture streaming is slow/generates spikes which calls "GFX wait for present" please adjust "Async Upload Time Slice" and "Async Upload Buffer Size" in Quality settings.

Let's get back to our objects and streaming. This small tutorial will help you to stream your objects during gameplay. There will be a lot of text like you probably see, but everything must be clear. After this tutorial you will be able to work/refresh/add/remove streamed models/objects very fast.

6.1 Model/Object preparation

We always advise you to open/create empty scene that we will call "Work" scene, copy your whole content (which you want to stream) there. Directional lights, player, UI should be at your game scene, so remove them from this scene or hide during splitting process. If you will properly set your layers, this objects could stay here.

1. Think about in how many layers you would like stream to your objects. Everything depends on your game style. We will give example for GTA style game. This will help you to visualize layer and streaming idea. We recommend you to use architecture like this:
    - **Big objects** - their shape is visible from long distance (buildings, roads, huge rock structures).
    - **Medium objects** - objects that are quite big and it's important to see them in quite long distance (static cars, building add-on, fences, lanterns, bushes, boxes, street signs)
    - **Small objects** - objects that are really small, not important, details. (papers, dirt, street wells, flowers, trash, dumpsters, small street signs)
    - **Lights and Particles** - you could put them at medium or small objects but also create individual layer for them.

If you create RTS game or game with top down view, you will probably use only one layer or less than this list above. For space game you should separate stars from planets, asteroids etc... Simply you should have catalogued objects by size and visibility distance. Anyway number of layers is your independent choice.

2. After you have decided how many layers you are going to use, you should rename your objects at the scene by giving them additional word at object name beginning. We call this as prefix in further doc parts. You should start from prefab renaming. When you rename your prefabs in project, objects at scene hierarchy also should be renamed automatically too. Look at examples:

   Example1. Building→ Big_Objects _Building

   Example2. Fence→ Medium_Objects_Fence

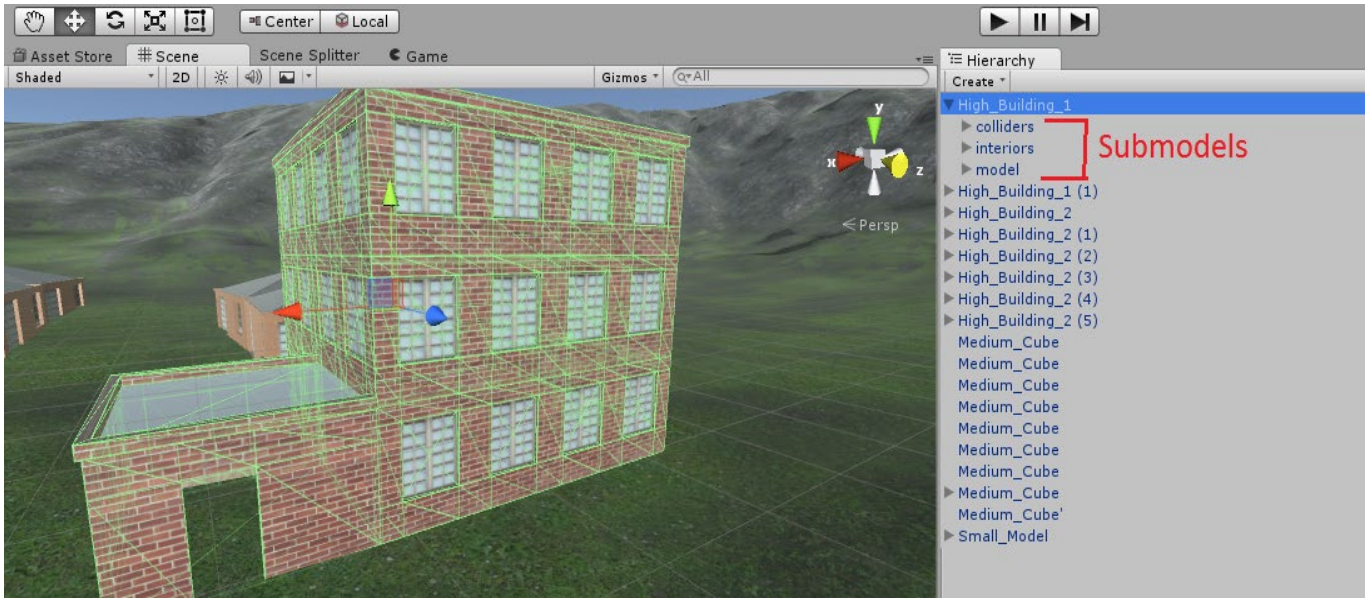
   Example3. Flower→ Small_Objects _Flower

   These names will be useful in automatic segregation process. Renamed assets will save you huge amount of time and they give ability to spreading models, without thinking what layer they belongs too.

   You don't have to rename objects, you could use parent game objects where you will put your models via category. For example all buildings into "Big objects" , all medium size objects into "Medium Objects" etc. If you put script "Objects parent" and set proper prefix like "Big_Objects_Building" system will simply take objects from proper parent into proper layer and ignore name of the objects inside. Probably it's **faster method** for most users but this force you to keep an order in hierarchy.

   If you are going to stream everything at **one layer** forget about renaming, you don't need this.

3. Unpack your models from all mother folders like at this screen or use our "Object_Parent" script like in "Tutorial World Streamer - Advanced_Work_Scene_Catalogs" demo scene:



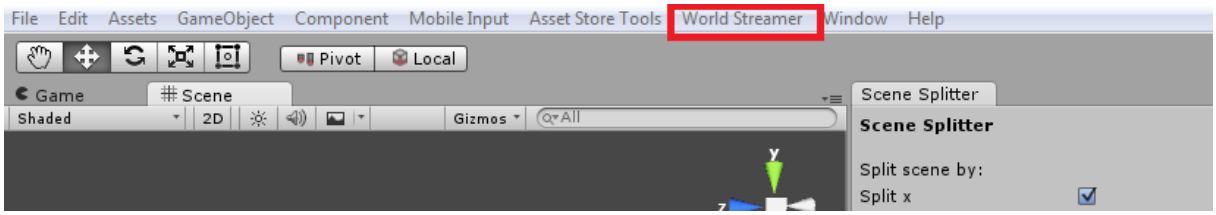
   Leave your sub-models and prefabs untouched if you want to stream them as one object with its mother/parent object. If you decide to stream building details separately and in shorter distance than building general shape you should also unpack them from building. If not, you always could leave them as LOD. Probably LOD will be smarter if details are part of building prefab but that means you leave them at GC list. That means you decided to hold unnecessary objects at the scene which could be heavy in big numbers for Garbage Collector "GC".

4. Done, now it's time for splitting, layers and streaming.

6.2 Scene split and virtual grid setup

When your Objects are prepared like in "6.1 Model/Object preparation" section, you are ready to split them, which means we are going to put them into correct virtual grid element and layer automatically. We will continue here, the example what we started in "6.1 Model/Object preparation" section. Now you should follow these steps:

1. Open "Scene Manager" window by clicking at World Streamer =>Scene Splitter



2. Fill scene Splitter window:
   - Add layer
   - Check "Split X" and "Split Z" don't check "Split Y" axis, if is not so important for streaming - like in Space Games is. If you don't want to remove models because of player Y position don't check "Split Y" axis. Probably for "Small_Objects", "Medium_Objects" "Split Y" axis could be useful.
   - Fill X, Z, optional Y size. Compare these with objects size and density. Don't put objects that are 100x100 units size in 10x10 virtual grid element size. Model streaming is very, very efficient, especially in build mode. At editor could be slow because of runtime batching and colliders baking. Don't worry about that, build is free from these. You could stream many buildings at once so maybe for a start try "X size" with 300 value and "Z size" also with 300 units. You can change that anytime, in few seconds you will have new setup ready to runtime test.
     The thing that you should take into account is: If you want to see your building from 1500 units distance, 1500(max view distance)/300 (virtual grid element size) = 5. You will have to set streaming distance (loading range at streamer object) to 5→ 300x5 = 1500.
   - Fill GameObject Prefix by a word that we add to recognize our big objects so by "Big_Objects". Splitter will check all objects at the scene and if any will contain "Big_Objects" at the beginning of the name it will use it in this layer.
   - Fill Scene Prefix (layer name) by name which will be clear for you, it could be the same as layer name like: " Far_View" or you could add any kind of info about which world it belongs to, if you got more than 1 world. So name with this info will look like: "World_1_ Far_view ".
   - Do the same operation for the rest of layers that you want to create. If you have only one layer, miss this point.
     "Medium_View" for objects that contain → "Medium_Objects" prefix
     "Short_View" for objects that contais → "Small_Objects"
     You could change virtual grid element size for each layer or leave it the same.
   - **Important:** There is no scenes/virtual grid elements limit but if you create too small virtual grid elements you will improve drawcalls count and also streaming will be slower only because of many parts to load/unload. Don't create to big virtual grid elements because you will hold in memory and garbage collector many unnecessary assets. This could affect bigger garbage collector spike. You should estimate your virtual grid size through few test. This will take only few min.

3. If you are sure about everything that you filled in Scene Manager window you could click "S" button for each layer separately or click "Split Scene" it will affect all layers. You could undo that operation by "C" button for specific layer or "Clear Scene Split" for all layers. System during split will prepare virtual grid elements from chosen assets. Virtual grid elements will become scenes in the future (next step). For more info about this section "4.1 Scene Splitter"

4. If you are sure about virtual grid elements that system made, click "Generate Scenes from virtual Grid". This is the last thing that you have to do with your assets. System will start generating scenes from virtual grid elements. It will create scenes in specified directory or in default one. System will also create scene collection prefab. In our example we will get scene collections prefabs called: Scene_World_1_ Far_View or Scene_ Far_view; Scene_World_1_ Medium_View or Scene_ Medium_view etc...

   This scene collection prefab will be stored in catalog with generated scenes. In our example WorldStreamer→SplitScenes→ "Scene_World_1_Far_View" or "Scene_ Far_view"; "Scene_World_1_ Medium_View" or "Scene_ Medium_view" etc...

5. These files (scene collections) contain all fresh info about your scenes: how many there are, how they were split, when they start and end (positions). This important scene collections prefabs will be connected by us to streamers objects at your "Gameplay" scene. It allows you to stream this whole content in proper time and place.

6.3 Streamer setup at your game scene

After everything in "6.2 Scene split and virtual grid setup" section is done and we have our scene collection or collections (if we made few layers), we could start to setup our "Gameplay scene". We should open our "Gameplay" scene if we have it or create it. We are going to do all points below for each layer. So step by step:

1. Open your "Gameplay" scene, this is the place where you have player, directional light and camera

2. Drag and drop into your game scene hierarchy “Streamer” prefab but if you use also terrain streaming, you should use existing one and simply add another layer/scene collection.

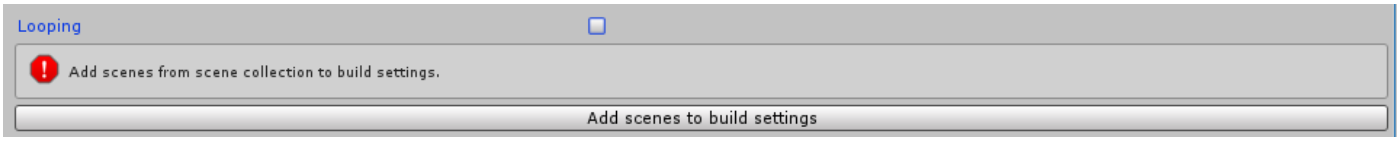
3. If you are going to use loading screen, drag it and drop into your game scene hierarchy "_Streamer_GUI" or "_Loading_Screen_UI" prefab. Do this only if there is no "_Streamer_GUI" or "_Loading_Screen_UI" already at the scene. This prefab contains graphical UI, Loading Screen, Teleport/Respawn and gizmo "Position" viewer for floating point fix or looping system. If this is your second or next layer you don't need to do this. We don't want to duplicate UI.

4. Click on "Streamer" at your hierarchy. Drag and drop your scene collection, that you made in "6.2 Scene split and virtual grid setup" section into "Scene Collection" window. By this move you connected your generated scenes with streamer.

5. Fill streamer or streamers settings with knowledge from "4.2 Streamer objects" section. For this manual example we will set:
   - Loading range to X= 5, Y = 0, Z = 5 for buildings that must be visible from 1500units and each virtual grid element(scene) is 300x300 units 300x5=1500units
   - Use Loading Range Min = false
   - Deloading Range X= 5, Y = 0, Z =5
   - Destroy Tile delay = 1 second

- Position check time = 0.1 seconds
- Max Parallel Scene Loading = 1
- Looping = false
- Scene Loading Wait frames = 1 or 2 frames
  Summing up that means we will load area 5x5 of virtual grid elements(scenes) where each virtual grid element is 300x300 and unload every virtual grid element that is further than this. We will check our position every 0.1 second. We will load max 1 virtual grid element during one frame and we left 1-2 frame to finish asynchronously stream operation, after virtual grid element was loaded and before we load next one.
- We fill "Player" window by drag and drop our player from scene hierarchy into "Player" window. Our system will follow that player and load terrain tiles if they will be close enough.

6. Now we will fill our UI, of course only if we want to use loading screens.
   - We click at "_Loading_Screen_UI" object at "Streamer_GUI" in scene hierarchy and we write Size = 1 or more (if we have more streamers) in Streamer value. You could find this value in "UI Loading Streamer Script".
   - Now we need to drag and drop our "Streamer" from scene hierarchy as element 0 in "UI Loading Streamer Script". Please drag and drop into this window rest of the streamers if you want to use them in Loading Screen.
7. Now we click at “Streamer" and at every streamer prefab “Streamer" at the scene and if you didn't added scenes from scene collections to the build setting, you should see a warning with button. Click on it and system will automatically add missing scenes or refresh them in build settings. Each time you change number and size of virtual grid elements you have to clicking that button!



8. **Click play and it works!**

Important!

Now you could easily split/unsplit/add new objects/remove content/move content and you could refresh layer with objects and streamed world in few seconds by click split and generate again at your work scene or by our "local area updater". System will refresh your scenes and "Streamer" at your game scene. You only need to refreshed them by clicking add to build settings button. That mean you could easily test many configurations!

7. Streaming by colliders

Collider streaming is useful to load/unload/cull interiors/rooms/floors or you could simply just load something only when player/object hit the collider. This function could save you a lot of cpu/gpu/ram/vram usage, you are able to fill whole town by interiors and load them only when player hit the doors or nearby area. It's really easy and fast in use.

7.1 Object preparation, split and scene generation

It's very simple. Let's do it step by step.

1. Create empty game object near objects that you want to stream by collider. Let's name it "Collider_Stream_Interior _1"

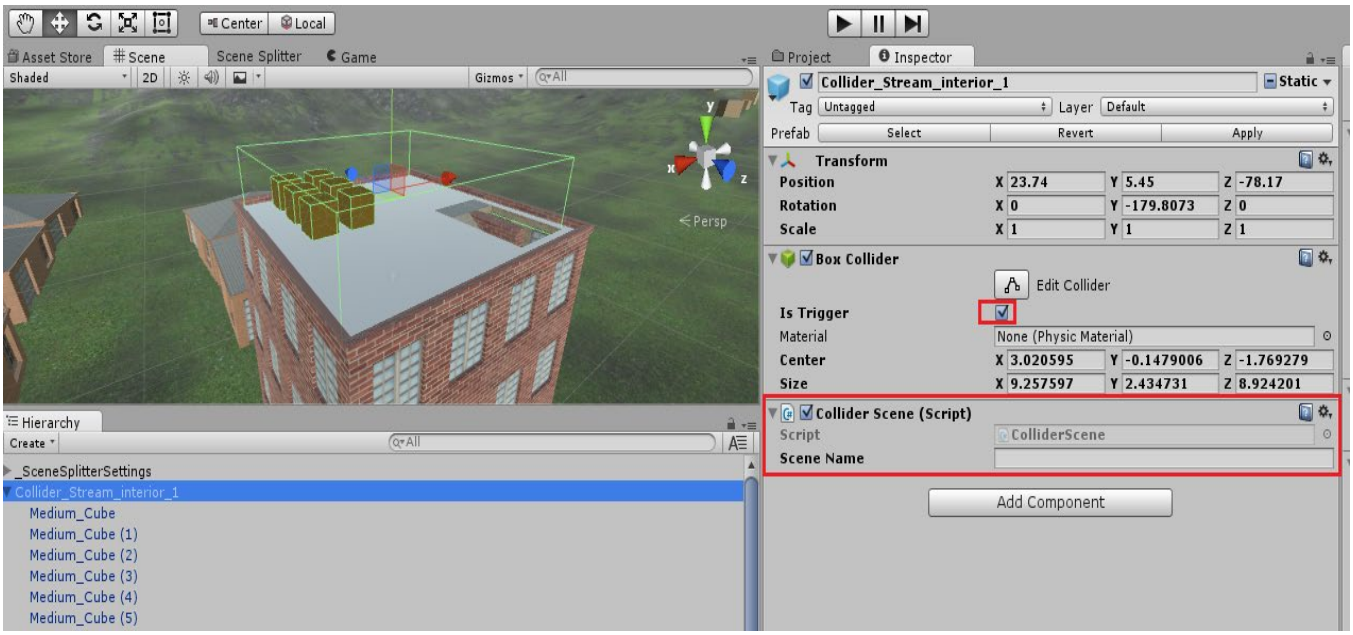
2. Attach collider to this game object and check "is trigger".

3. Adjust collider size to the area in which streamed objects should be loaded and stay in memory

4. Move objects that you want to stream into "Collider_Stream_Interior _1" object, let these objects become their children in scene hierarchy

5. Attach ColliderScene object to our "_Interior _1".

6. After all these steps situation should look like this:



7. Open Scene Manager window
8. Hit "Generate scenes from colliders only" button.
9. Generation process, just generated scene "Collider_Stream_Interior _1" scene and prefab with collider "Collider_Stream_Interior _1_stream". You could find them in WorldStreamer catalog → SplitScenes → ColliderScenes. Prefab contains connection to the generated scene and it holds original object position. If you move this prefab from project window to hierarchy at any other scene, objects will be in the same place as in scene where they have been generated. Of course you could change prefab and spawning position.

- **Important!!!** Our system gives ability to hide your objects streamed by collider in scenes that are streamed by streamer from scene collections (virtual grids). This looks like small matrioshka, scene that stream scenes etc... Simply put them in any layer, during generation process system will recognize that it found objects that are streamed by colliders. It will create scenes for this objects and put in correct places. The only thing that you have to do is to add them to build settings by Scene Manager→BuildSettings→"Add scenes to build settings" button. After you generate scenes this window should be automatically filled with proper objects. Another advantage of this construction is, if layer in which you left your collider streamed objects is looped and it's using floating point fix, your collider streamed objects will be synchronized with this layer. Objects will properly follow world and player. **If you only use this construction** in your project, you don't have to pass through "7.2 Collider streamer setup at you game scene" section. The only thing to have to do from this section is: Drag and drop from project into "Gameplay" scene hierarchy "_Collider_Streamer_Manager". This prefab must have **"ColliderStreamerManager" tag**. Check it! Fill it with player object from your scene hierarchy.

7.2 Collider streamer setup at your game scene

It is much easier than generating.

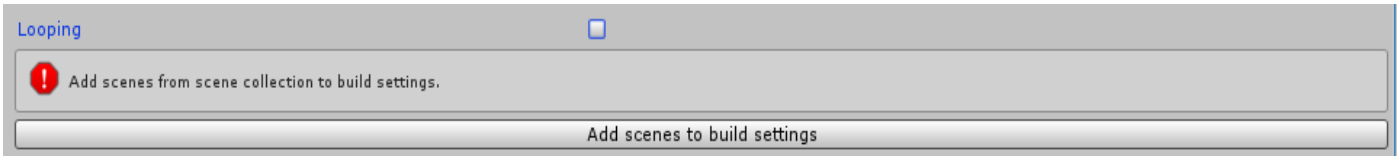
1. Open "Gameplay" scene

2. Drag and drop from project into scene hierarchy "_Collider_Streamer_Manager". This prefab must have **"ColliderStreamerManager" tag**. Check it! Fill it with player object from your scene hierarchy.

3. Set object tag that should initiate streaming after hit – for example player or main camera. This player or camera object must have collider with rigidbody to initiate streaming. In most cases is good to turnb on kinematic and off gravity.

4. Simply move generated prefab "Collider_Stream_Interior _1_stream" from project window into your scene hierarchy window.

5. Hit add to build settings button. If you forget about this, scene will not load.



6. Decide if only player during hit should initiate streaming by checking "Player Only Activate" or not.

7. Set up unload Timer. After this time scene will be unloaded, but this will happen only when streaming initiator leaves collider area.

8. Click PLAY and Move your player/object and hit collider. Objects will become loaded.

8. Multiple layer streaming - why you need this.

We put a lot of attention into layers at section "6.Model/Objects Streaming" and "5. Terrain Streaming". Everything is about performance.

The idea is to hold as low amount of objects at your game, as it's possible. If you left low amount of objects at the scene some scripts could work faster. When you call garbage collector, its spike will be reduced to value that you could skip. If you want to remove objects from memory/CPU you have to call GC (garbage collector) from time to time. We could look at this from second side. If your GC is still too heavy, with layers you will not have to call it so often.

Layers give ability to hold this assets that are really necessary for you at current moment. Small objects will be removed from your memory and CPU (also GC list), if they are far enough. We give ability to hold big shapes like buildings, terrains in far distance, and enjoy the big amount of details that will eat your resources only at short distance.

Layers with ring streaming give ability to replace detailed and heavy unity terrains by low poly meshes. Unity LODs system don't support this construction. We will say more, in this construction you don't hold/load LOD0 at/to memory if it's not necessary. This dosen't have to be terrain, what do you think about replacing particles by static billboard mesh at far distance? With unity LODs even if you switch to billboard, particles will still eat your CPU. With many particle sources you could save a lot by our ring streaming.

Layers could be useful to separate following objects: Lights, FX, Models, AI. While they are separated, multiple developers with different specializations could work at the same area, without interfering with each other.

Summing up, layers give ability to optimize your game a lot and hold very detailed world without any performance issues. Users have performance increase by more than 200-300%. As layer streaming range is a dynamic value, player could have this range adjustable at setting

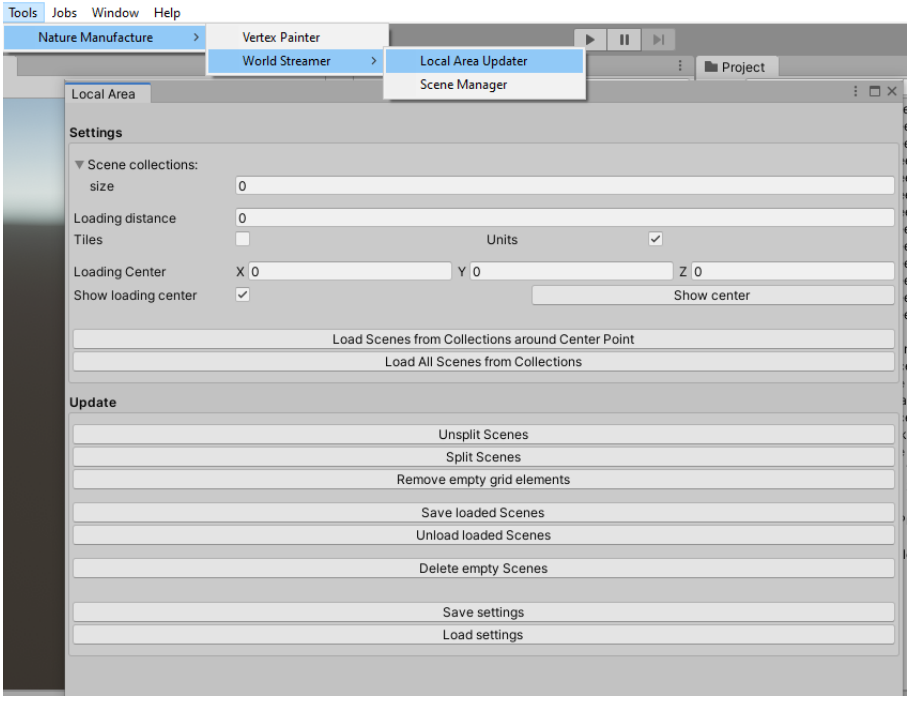
Remember to do not load all layers at the same time, for example when you load terrain in 500x500 grid then try to stream objects in 350x350 grid so 500 and 350 layers will not load at the same time, because terrain will be loaded at position: 0 , 500, 1000 while other objects on 0 , 350, 700, 1050 etc.

9. Local Area updater

Local area updater could be useful when few developers work at the same area, but on different layers or they are working at the same layers at the same world but at its different places/regions. By layers word we mean - AI, FX, Models, Light , Big models, small models, terrains.

9.1 Local Area updater options

- Open empty scene and save it. You could call it "Local Area Updater Scene"



- Open Local Area Updater window. Tools -> Nature Manufacture -> World Streamer→ Local Area Updater

- **Scene collections** is a place where you can add your multiple or one scene collection prefabs. From this "scene collections", system will load split scenes (virtual grid elements) for you. Remember about layer correct order if you use multiple layers. When you use layer with empty prefix, system could put your models to wrong scenes when you hit "Split Scenes"! Scene collection with empty prefix should be last one on the list.

- **Loading distance** is a range of loaded world. It could be in virtual grid elements (scenes) or unity units you could chose by checkbox below. When unity units option is checked, unity units distance will be recalculated into virtual grid elements. Unity units are useful when layers which we want to load have different virtual grid elements size. As an example we want to load terrain, models and fx that belongs to this terrain area. Terrain is at different layer than models and fx and it has different virtual grid elements size than them. In this case we should use Unit size loading distance.

- **Loading center** is the XYZ position in around which local area updater will load models/objects, in range that you specified above. Soon we will add also streaming that follow unity camera in editor.
- **Load scenes from collections around Center point** will immediately load your scenes, from chosen scene collections, around the point you chose and in range that you setup.

- **Load all scenes from collections** will immediately load **all** scenes that are connected to chosen scene collections.

- **Unsplit scenes** button it's useful when you want to create your big scene from split scenes(virtual grid elements). You simply need to load all scenes from collections then click "unsplit" and you have your big scene with all models in one place again. By this button you could merge your layers, because you simply remove their virtual grid elements membership.

- **Split scenes** button will put new added or moved models to correct scenes, layers and virtual grid elements.

- **Save loaded scenes** button will immediately create new or actualize old scenes with changes that you made in virtual grid elements. It will actualize your scenes **but** you have to click "Split scenes" button before you click "save loaded scenes". It will show you bugged unity window, simply click ok or enter. This window contain list of scenes that you updated. If list is too big unity window is bugged, we have no access to it, to fix that. Just don't care about this and click enter.

- **Unload loaded scenes** will remove your loaded scenes and you will get back to clean empty scene.

- **Remove empty virtual grid elements** by this button you could remove empty scenes from scene collections. This will avoid empty scene loading during gameplay.

- **Delete empty scenes** will delete empty scenes from your project, sometimes it's useful to hold them. As an example if you are going to put new models in this area, system will not have to create new scenes again. So you could hold empty scenes and clean them at the end of work.

9.2 Workplace setup, virtual grids loading, world refreshing

1. Open empty scene and save it as "Local Area Updater Scene".

2. Open Local Area Updater window

3. Set number of layers that you want to load. Scene collections→ click on the arrow →type number/size

4. Drag and drop into list your scene collection prefabs. Remember about layer/scene collections correct order if you use multiple layers. When you use scene collection with layer that contains empty prefix, system could put your models to wrong scenes when you hit "split scenes"! Scene collection with empty prefix should be last one on the list.

5. Set size of the area that you want to load in virtual grid elements (scenes) or in unity units.

6. Set point from which you want to load your area "Loading Center"

7. Click "Load Scenes from Collections around Center Point", system will immediately load virtual grid elements (scenes) from chosen scene collection in specified range.

8. Make changes that you want to do for example: add objects, remove, rise your world by new regions, move objects.

9. Hit "Split scenes" button, it will sort, move new and old content into correct layers and virtual grid elements. In this place if you use layer with empty prefix and it is on the top of the scene collection list, everything will be moved only to this layer and nothing will be left for the rest of the list.

10. Click "Save loaded Scenes", this will save your changes and refresh/actualize scene collection prefabs.

11. "Unload loaded scenes" this will remove all content and clean your "Local Area Updater Scene".

12. If system had to create new scenes, you should add them to build setting. You could do this by our tool scene splitter or at "Gameplay" scene by clicking at each streamer object and clicking at error button "Add to build settings". Button will show up only if there will be changes in amount of scenes.

10. AI and navmesh solutions

10.1 Unity navmesh for Unity 2022.3 and higher

For Unity 2022.3 and higher engine versions with the new Unity navmesh system. Navmesh is a hierarchy game object with position. You can stream it like any other object. Just keep it in grid. You can create a special layer for it or stream with terrain if it's not to heavy to load them together. You also can keep it in one big navmesh and with "object to move" script move it during the floating point fix process. Everything depends on world size. For lower engine versions look below:

10.2 Custom navmesh solutions

This solution is the easiest. There are few AI solutions that don't need navmesh or they give ability to save it in parts and load.

- **APEX** solution, don't require navmesh it even react geometry changes. Fast and very advanced, it fits most projects.
  https://assetstore.unity.com/publishers/6827
- If you have tried more solutions and they work well for you give us a sign and contact us, we would love to include them here. Probably most solutions that give ability to save navmesh in parts and connect parts at runtime will work fine. All solutions that do not require navmesh for pathfinding will work fine too, assetstore is full of them.

10.3 Unity navmesh for lower engine versions

With streamed world you could use unity navmesh. It could be loaded and removed with scenes, it also works with floating point fix and looping systems - unity 5.6+.

Here are the options that you could use at the moment with unity navmesh:

- **Dynamic navmesh**
- **Navmesh in one part:** You also could always hold navmesh for whole world in gameplay scene. Navmesh is an object that you could move between unity catalogues. You could bake it at "Work" scene with all objects, and after you finish baking, you could copy it and paste into "Gameplay" scene catalog. If you don't have actual work scene you could rebuild it at empty saved scene, by our local area updater tool. This tool loads all scenes from scene collections that you decide to use. For more info check **"9. Local area updater"** section.

- Navmesh in many parts: Load specified area by local area updater. For more info check "9. Local area updater" section. When you load your area you could bake navmesh for it.

Floating point fix and looping

It's pretty simple to handle this. Here is pack of details that will help you.

- **AI spawn**. While player is moving and we use floating fix system with looping or not, our player position is reset from time to time. World Mover script holds position differences between local player position (after reset) and real player position. Player position (XYZ) - Current Move (XYZ) = Player real position (without reseting). This values are useful for spawners to correct spawn position to actual player position. If it's single player you could move spawners with player by attaching and filling "object to move" script. Check **"4.5 Floating Point Fix system"** section.

- **AI pursuit**. While player is moving and we use floating fix system with looping or not, our player position is reset from time to time. To avoid dropping our AI followers:
  - Single player: you have to attach "object to move" script to each NPC object and fill it on start by actual "_World_Mover" object at your scene. For more info check: **"4.5 Floating Point Fix system"** section.
  - Multiplayer: simply correct rpc/message with position by "Current Move" value from client World Mover script. For more info check: **"4.5 Floating Point Fix system"** section.

Version changes

Version 2.0 changes

- Totally new UI
- Terrain manager system
- Terrain split system
- Low poly mesh generator from terrain shape
- Low poly mesh generator from terrain trees for NM foliage and assets that contain low poly last lod.
- One streamer to rule whole streaming instead of many
- Split option from parent folders
- System speed up
- New huge 4k demo scene with trial objects
- Versions for hd and urp 7.2+

12. TIPS, additional info, performance problems and solutions

- If texture streaming is slow/generates spikes "**GFX wait for present**" adjust "Async Upload Time Slice" and "Async Upload Buffer Size" in Quality settings. You can also play with texture streaming in player settings “Virtual texturing”
- You can check performance only in the development build with the auto-connect profiler option. Access to memory in unity editor is very slow and generates many CPU spikes that will not show up at game build. To reduce spikes at the editor you could turn off static batching for a while.
- If you want to remove textures from memory at your games, use Amplify, granite other texture streaming system. Check 14. Third party assets that might be helpful section. This will allow you to use unlimited resolution of textures without memory grow. Streamer will work extremely faster, because it will load only colliders, meshes, terrains. Textures will be streamed from the block by other outside systems. Unity bought Granite system into engine and you can now play with texture streaming in player settings “Virtual texturing”
- REMEMBER to check your static objects as static! It will reduce CPU use during scene streaming process (build only). If you use floating point system fix, you have to batch/bake object by using custom tool.
- REMEMBER to check "Prebake Collision Meshes" box at player settings, to stream colliders much faster! This option will make your game build time much longer! Best option is to use it at final builds.
- We advise you to check "optimize mesh data" box in player settings, this will reduce game build size and stream time, but it will extremely lengthen the build time! So check it only on final build or when you have much time.
- Remember to **use incremental GC** in your project, it speed up GC a lot. You can find it in player settings.
- Every error or warning at your scenes will slow down stream system. Remember to clean your scenes from bugs and mistakes.
- Use as simple colliders for your objects as possible! If colliders are baked (player settings) they are loading much, much faster. Anyway simple colliders are healthier for your project.
- If your scene is to heavy, cut it into smaller parts and load with bigger range. You will load many small parts and this way guarantee better performance. Good way is also to load terrain separately by second world streamer prefab. The clue is do not load everything at once!
- **AVOID! Loading huge meshes, unity stuck in loading such mesh and generate spike. Size of the mesh depend on goal device**. Even engine import process is slow on huge meshes.
- Remember that terrain detail resolution, control texture resolution, base texture resolution, and heightmap resolution also affect streaming performance so adjust it wise. If you don't need so much details, cut unused data by setting lower value.
- **If Garbage Collector "GC" function spikes too heavy (spikes)**, you probably have to much objects at your scene or unity terrain have to big splatmaps (that is silly but yes that how it works). You probably have spike at **GC.Mark.Dependencies** function which is related to the amount of game objects at the scene. You could reduce it by layer adjustment. You should hold in scene only things that you actually need. This could be also your/unity scripts memory leak bug. You could change the density of GC by yourself in our scripts or remove it and call from time to time by your own script. Anyway remember if you got GC spikes, you made something wrong. It could be also Unity bug, sometimes unity functions also have memory leaks.
- With ring streaming you could replace heavy unity terrains by low poly mesh at far distance. This could save hundreds MB of memory, avoid CPU spikes and remove static CPU usage that unity terrain refreshing generates all the time… By replace trees by low poly meshes you save few thousand of batches.

- Use rather instanced objects for trees and grass and big amount of one type objects. Loading huge static batched baked mesh is slower in engine. Instanced object it loaded once and really fast. It becomes populated in the scene. Instead of loading mb you load only few kb into memory.