SCENE OPTIMIZER

By Procedural Worlds

The Scene Optimizer is a collection of useful tools for optimizing the performance in your scenes.

Contents

About Procedural Worlds

Powerful, simple, beautiful. Friendly tools, gorgeous games!

Procedural Worlds empowers artists and developers to bring their vision to life by making it easy to create beautiful worlds. Leverage the latest procedural generation techniques to take the pain out of creating stunning environments and focus on creating amazing games.

NEW – Canopy community site

Canopy is our new community site for all Procedural Worlds Tools with support forums, knowledge base library, tutorials, and much more.
Register with Canopy to receive FREE stamp packs for Gaia and get the best out of Ambient Sounds.

The only end to end environmental generation and delivery suite:

Gaia Pro 2021 - A world generation system for creating, texturing, planting and populating scenes from low poly mobile, VR and through to high end desktop.

GeNa Pro - A sophisticated localised level design tool that augments Gaia's broad-brush strokes, by working intuitively to give fine grained control.

SECTR - A suite of performance-enhancing tools that enable open world streaming, massive mobile games and includes the latest techniques in audio occlusion and propagation.

Ambient Sounds - Lets you configure music and sounds to create a unique atmosphere for each region in your game, which can react to changes in your gameplay instantly.

Pegasus - A cut scene and fly through creator that makes it easy to show off gorgeous environments and drive characters through scenes with localised avoidance and Mecanim animation support.

Spawner Packs – You can save time by using our pre-configured Procedural Worlds Spawner packs (PWS). The packs contain configurations for our tools Gaia and GeNa, and are designed to work with popular asset packs from the Unity Asset Store. Currently available:

PWS – POLYGON Fantasy Kingdom - Spawner Pack
PWS – POLYGON Nature - Spawner Pack

Micro Biomes - Let us inspire you with our new Micro Biome series where we put together groups of matching assets that cover one specific aspect of environmental design and release them in targeted high quality packs:
Micro Biomes – Fields of Color

Introduction

Thanks for purchasing the Scene Optimizer! The Scene Optimizer will help you with optimizing your projects to gain better performance in your final product.
Additional performance allows you to increase the graphical fidelity in other aspects of your project, or to build up a “performance buffer” that allows users with weaker hardware to enjoy your product as well.

Installation

You can install the Scene Optimizer as any other product via the package manager in the unity editor, or by importing the .unitypackage file

Usage

To start using the Scene Optimizer, simply open the Main Window by going to:

Window > Procedural Worlds > Scene Optimizer > Main Window .

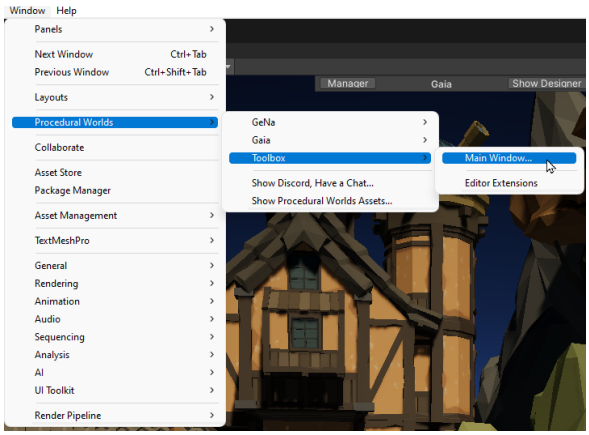
This will open the main window of the scene optimizer that lets you access most of its functionality. It is recommended to follow along with the Quickstart Guide in the next chapter to gain a basic understanding of the tool.

Quick Start

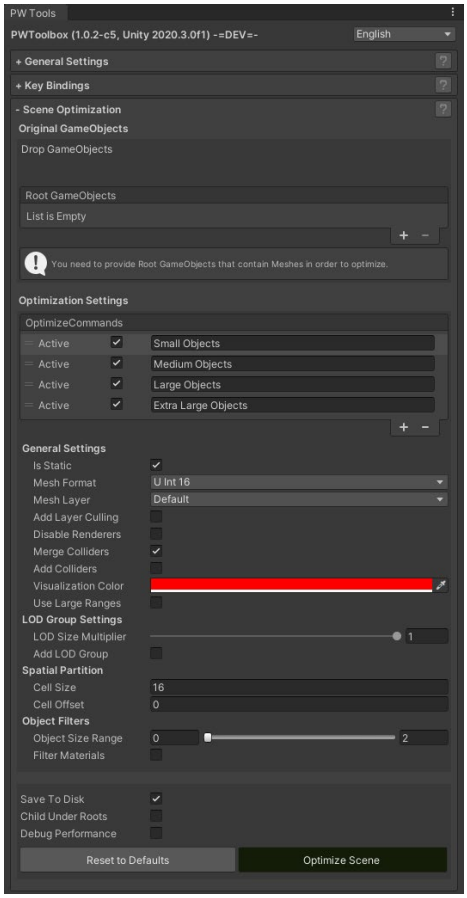
The Scene Optimizer can be accessed from a separate window inside the Unity Editor.

1. Once you have the Scene Optimizer imported into your project. It can then be opened through

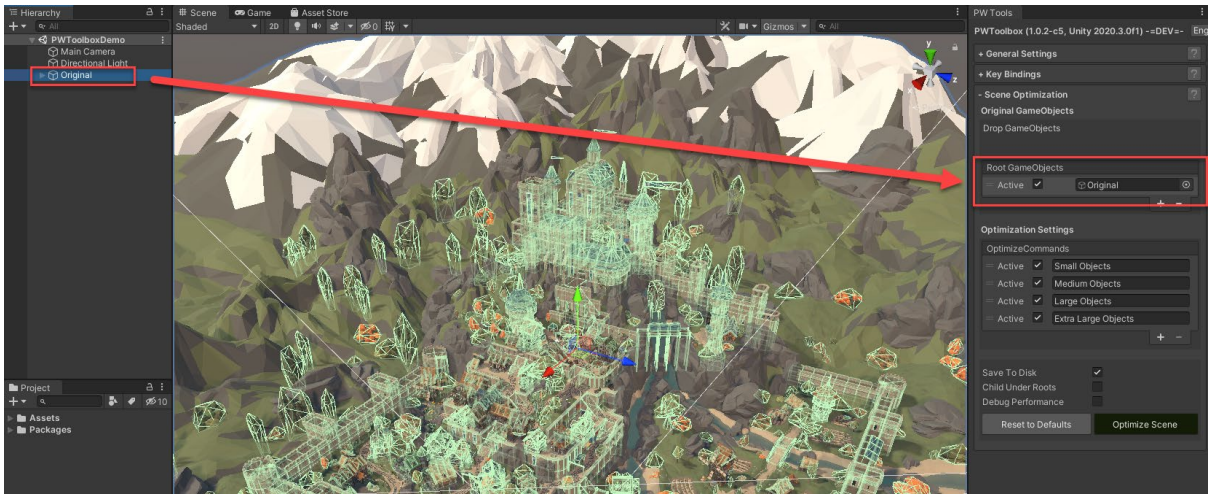
   Window -> Procedural Worlds -> Scene Optimizer -> Main Window .



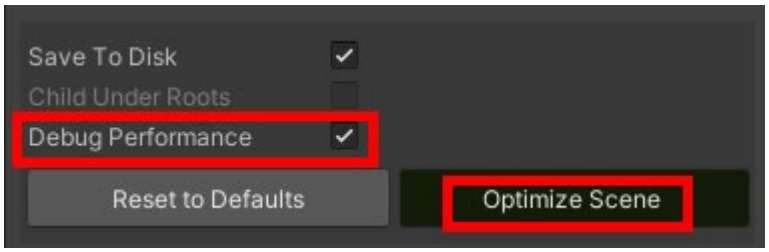
Once opened, you'll be presented with a series of settings to configure:



2. For mesh combination, which is the main feature of the scene optimizer, you need to add a 'Root GameObject' that contains all of the meshes are that you wish to combine. In this example, we have an 'Original' GameObject that contains all the original meshes as separate objects.

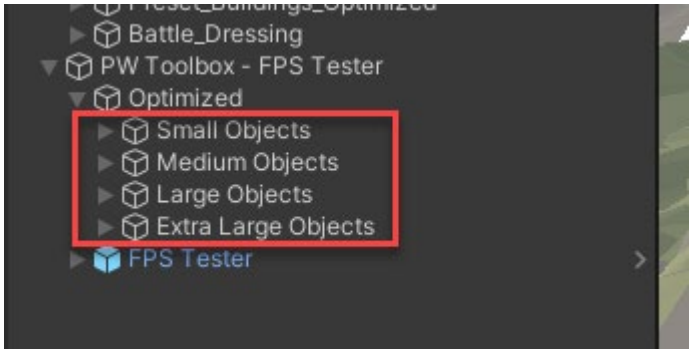


3. Normally you would configure some of the additional settings, but for the purpose of the quickstart, the default settings are fine. Select the 'Debug Performance' checkbox. This will create a test environment for us to compare quickly between the original Game Object setup vs. the output of the Scene Optimizer which will allow us to measure FPS improvements.

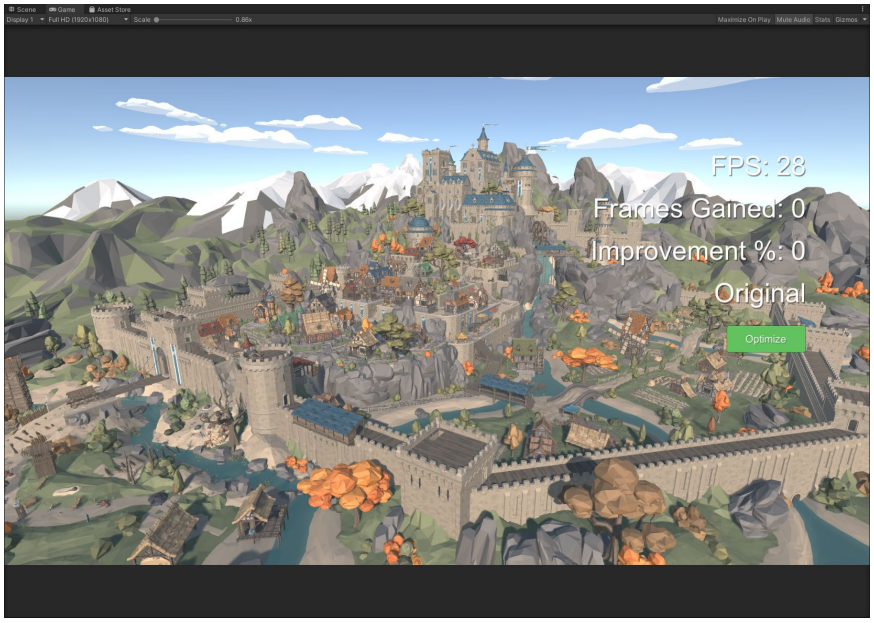


4. Then, click on 'Optimize Scene'. This will create a new GameObject in the scene with a pre-configured UI to demonstrate the FPS improvement over the original GameObjects during Runtime.

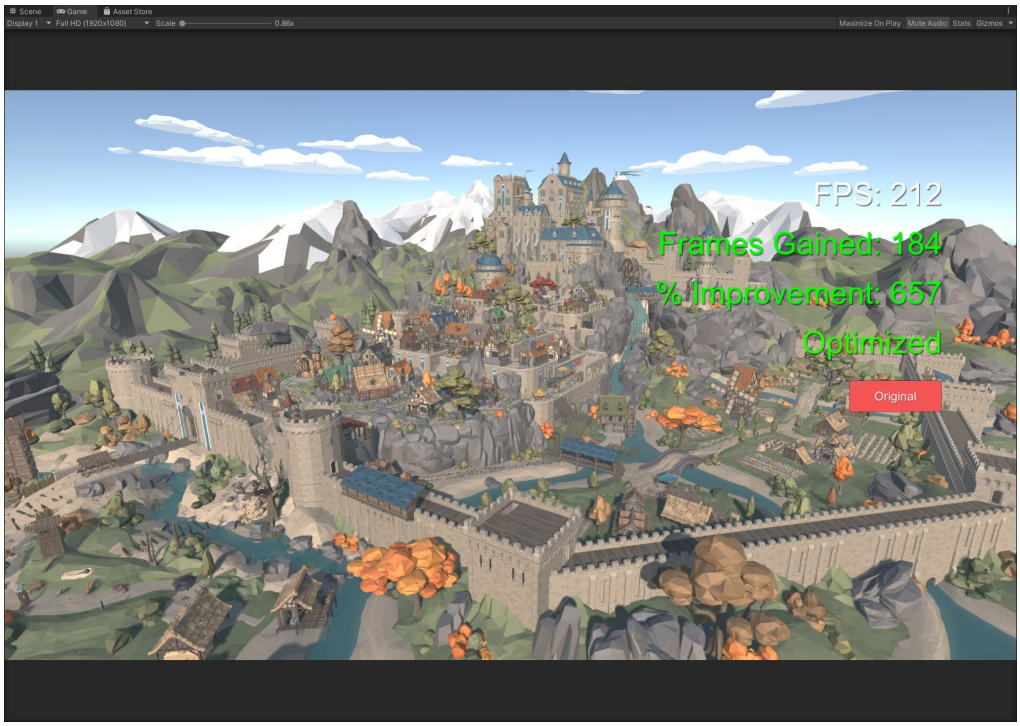
   You now have a bunch of optimized GameObjects in your scene!



5. Hit play and see the performance improvements:



As you can see, the Original GameObjects are running at roughly 28 frames per second.
We have quite a lot of separate GameObjects in this scene so that's not surprising that Unity has a lot of overhead.



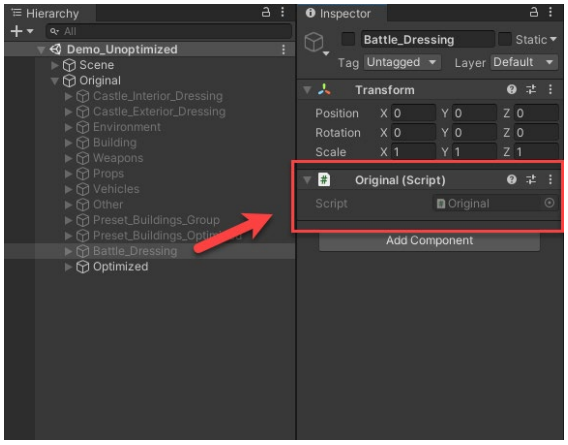
If we click on the green 'Optimize' button, you'll notice that we are getting 212 frames per second! That's exactly a 657% improvement over the original GameObjects in our scene!

This quickstart introduction was the quickest possible way to perform an optimization for your scene. You will be able to tweak your optimizations further by tuning the settings and especially setting up layer based distance culling and tweaking it according to the objects being used in the scene. For more information please see the tutorial video, and take a look at the description of the Interface in the next section.

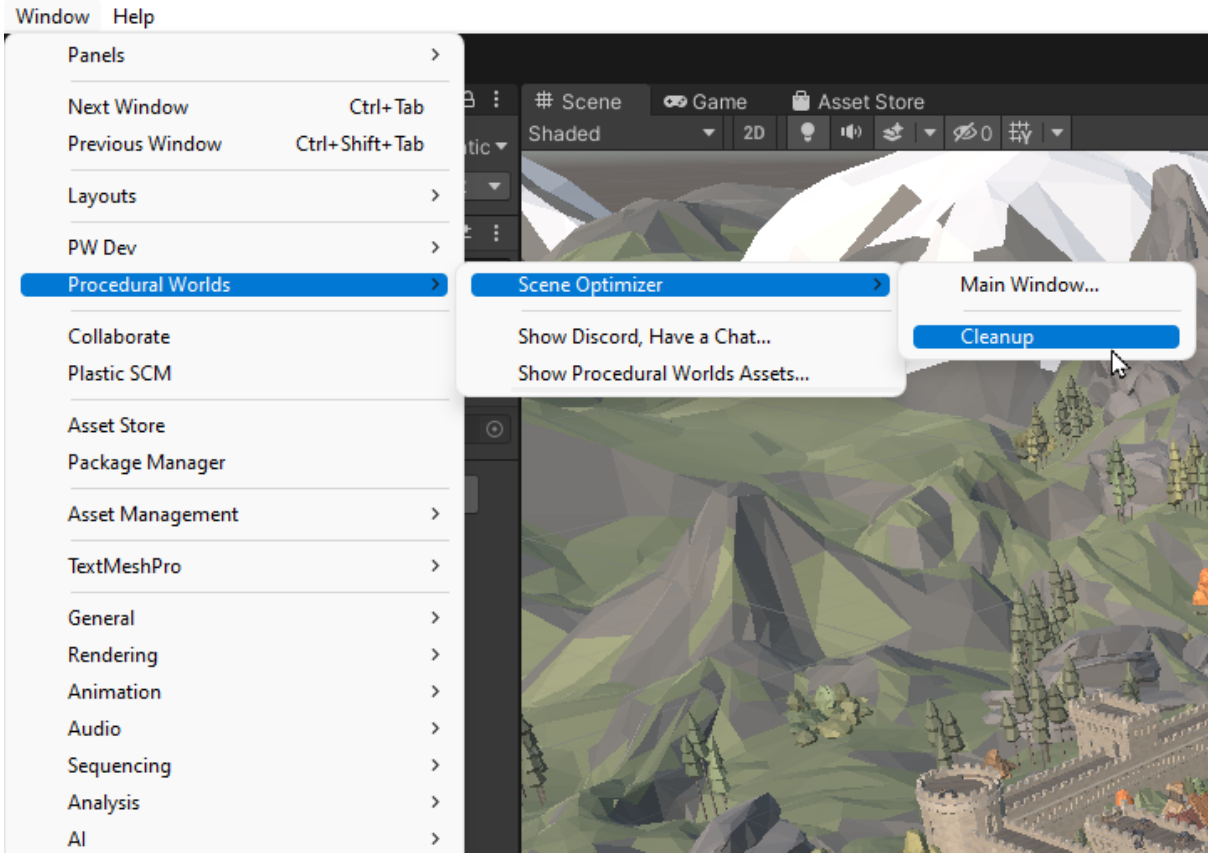
Clean Up

The optimiser can create a lot of data depending on the complexity of your scene. You should do a 'Clean up' operation after you have optimised your scene.

This will delete all the components that were added to your original objects and other kinds of scene data clean-up.



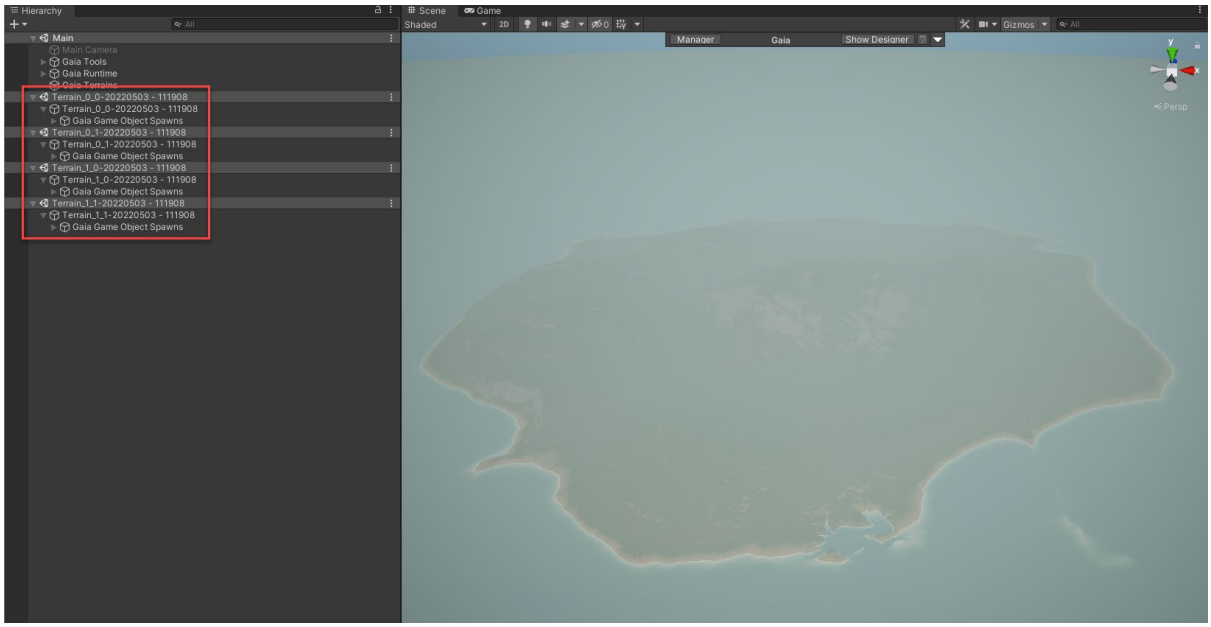
To do this, go to **Window > Procedural Worlds > Scene Optimizer > Clean up**. And wait for the process to finish.



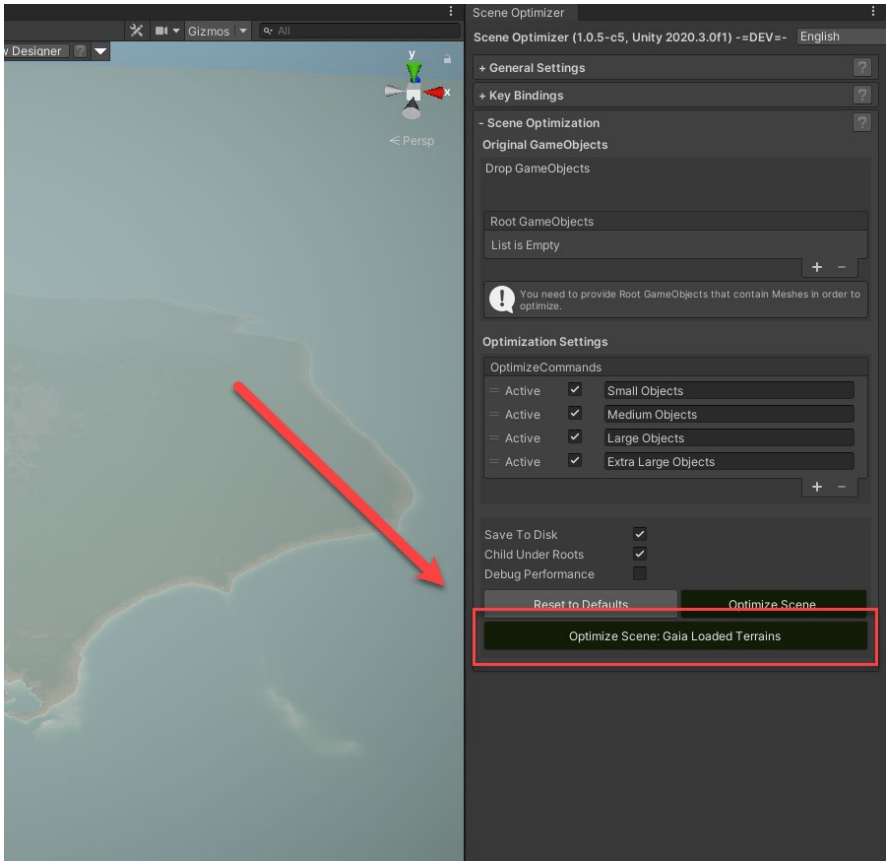
Gaia Pro Support

By default, the Scene Optimizer supports optimizing Gaia Terrain Streaming in your project.

To see this in action, go ahead and create your Gaia Environment using the Terrain Streaming system as you normally would.



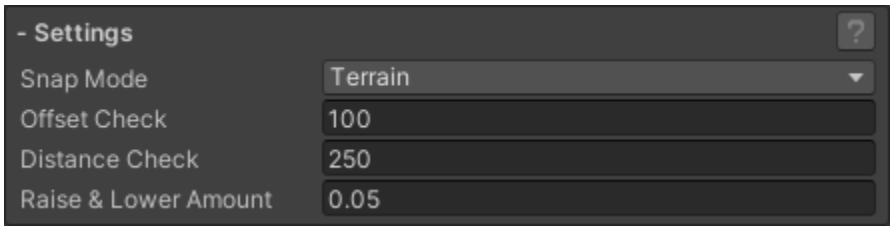
You should then notice a new option open in the Scene Optimizer window. This operation performs optimizations on the Gaia loaded terrains.



Interface

This chapter details the individual settings found in the Scene Optimizer window.

Global Settings Panel



The Settings panel shows the global settings for the various features of the Scene Optimizer.

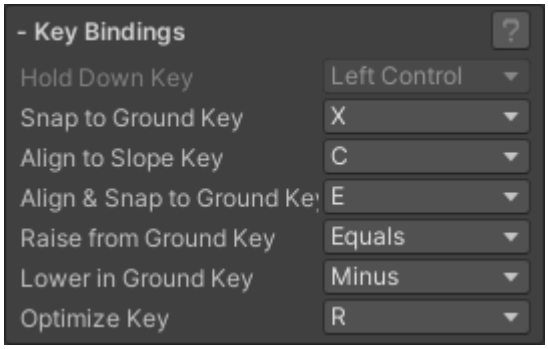
Snap Mode: If enabled, snapping will only happen on Terrains, otherwise snapping will be on Meshes.

Offset Check: Position offset of the selected object, helps with checking if objects are under the ground.

Distance Check: How far the raycast check will go. The higher the value, the more it will affect performance.

Raise & Lower Amount: How much the object is raised or lowered into the ground.

Key Bindings Panel



Key bindings settings to control various features of the Scene Optimizer.

Snap to Ground Key: Button used to snap objects to ground.

Align to Slope Key: Button used to align objects to slope.

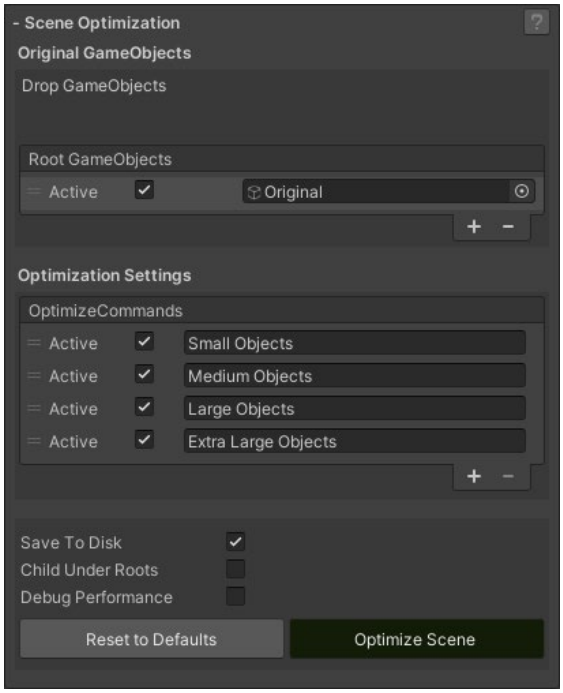
Align & Snap to Ground Key: Button used to align object to slope & snap object to ground.

Raise from Ground Key: Button used to raise the object up from the ground.

Lower in Ground Key: Button used to lower the object to the ground.

Optimize Key: Button used to perform Scene Opimization.

Scene Optimization Panel



This panel shows all settings pertaining to the operation of combining meshes to optimize your Mesh Renderers in your scene.

Drop GameObjects (Drop Area): GameObjects dropped here will be added to the 'Root GameObjects' array.

Root GameObjects: GameObjects added to this list will be used in the process of optimizing your scene.

Optimize Commands: A list of optimization instructions to be performed with all of the gathered objects within the Root GameObjects.

Save To Disk: Saves the optimized object's assets to disk (significantly reduces the load time of scenes).

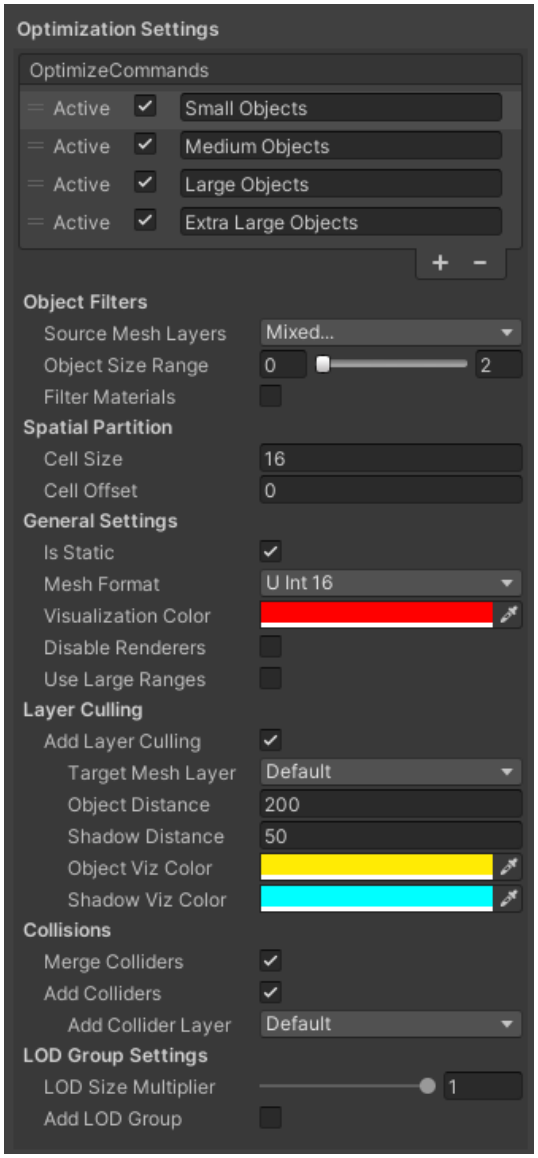
Child Under Roots: Childs the optimized objects under the root GameObjects. Note: This is ignored when 'Debug Performance' is enabled.

Debug Performance: Creates FPS UI and environment for testing performance when optimizing.

Reset to Defaults: Reset to Defaults: Resets the Scene Optimization settings back to the default settings.

Optimize Scene: Performs the Combine Mesh Operation using all the configurations above.

Optimization Settings



Combine Commands are a set of instructions to be performed (top down) and filters GameObjects in the process based on Cell Size, Object Size, Filtered Materials, etc.

Object Filters

Source Mesh Layers: The Layers to source Meshes from for this Optimize Command.

Object Size Range: The minimum size of objects that will be collected for this Optimize Command (Inclusive).
Filter Materials: Filter the Optimizing of objects based on Material.

Material Entries: List of Material Entries to be used in the Scene Optimizing Process. Provides control over what Shaders can be used or not.

Add new Material: Adds a new Material Entry for Manually filtering out GameObjects.

Scan for Materials: Scans the Root GameObjects for all Materials being used in the current operation.

Spatial Partition

Cell Size: The size of the cell partition to optimize your scene (in world units).

Cell Offset: The offset of the cell partition to optimize your scene (in world units).

General Settings

Is Static: Static Meshes are required for Occlusion Culling to work properly.

Mesh Format: The index format to restrict meshes to. UInt16 is recommended for Mobile builds. Note: See 'Mesh.IndexFormat' in Unity's Scripting API.

Visualization Color: The global visualization color for the Optimized Objects.

Disable Renderers: Disables the original mesh renderers.

Use Large Ranges: Enables the use of larger values in the following properties.

Layer Culling

Add Layer Culling: Adds Layer Culling to the Camera.

Target Mesh Layer: The target layer to combine all meshes to.

Object Distance: The distance that Objects will render for this layer.

Shadow Distance: The distance that Shadows will render for this layer.

Object Viz Color: The visualization color for the object distance.

Shadow Viz Color: The visualization color for the shadow distance.

Collisions

Merge Colliders: Merges Colliders to the optimized from the original colliders.

Add Colliders: Adds Mesh Colliders to Optimized (if missing).

Collider Layer: The Layer to put all the newly created Colliders onto.

LOD Group Settings

LOD Size Multiplier: Applies a multiplier to the final calculated LOD Size. Useful for controlling the render distance of generated LOD Groups.

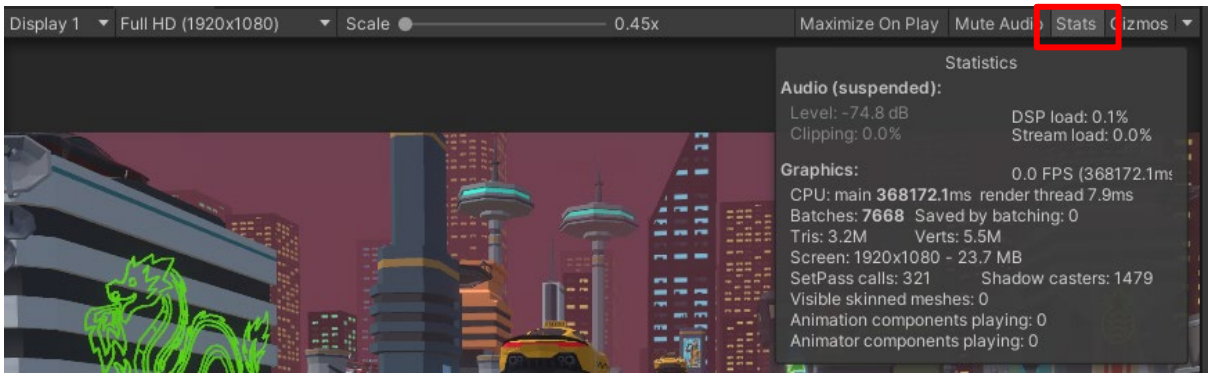
Add LOD Group: Adds LOD Groups to objects that do not have existing LOD Groups.

LOD Cull Percentage: The percentage to cull renderers in the LOD Group.

Optimization Information

Keeping track of many individual GameObjects in a scene has a cost to performance. There is a lot of data often associated with each mesh that gets sent to the GPU to render and reducing the amount of data that gets sent to the GPU improves performance. One way of reducing this data is by combining meshes, so instead of information such as position, scale and rotation being sent per mesh, this data is only sent once in this combined mesh.

Unity provides a statistics window to profile performance quickly in a scene. It can be accessed through the Game window.



Statistics Window Information

A brief explanation of the most important data represented in the statistics window is summarized.

FPS / ms

FPS stands for Frames per Second, which is an indication of how many times Unity can render a single frame in a second. The more Frames per Second, the smoother a scene will run. The small ms value pictured in the brackets next to the FPS counter represents milliseconds, which is a unit of measurement showing how long it took Unity to render each frame. So, milliseconds and Frames per Second are directly related to each other.

There is 1000 milliseconds in a second. If your scene is targeting a minimum of 60 Frames per Second, the target milliseconds would be (1000/ 60) = 16.667 milliseconds.

Batches

Batches refer to groups of data that can be sent by the CPU to the GPU to render. Sometimes objects are grouped together because they share the same material, as this saves the material being included in many different individual gameobjects being sent to the GPU. As a rule, the less batches the better the performance, but it can depend on how certain hardware is able to handle certain aspects within these batches. The process of the CPU telling the GPU to render a mesh is referred to as a Draw Call, so the less Draw Calls

the CPU must send the better. 3 objects with the same material may take 3 Draw Calls to render, whilst combining these objects into 1 single object can be drawn with just 1 Draw Call.

Tris / Verts

Tris is shorthand for Triangles, and Verts is shorthand for Vertices. These refer to the total number of triangles and vertices being rendered in the frame. Generally, the less triangles and vertices in the frame, the better the performance, but certain hardware is better suited to handling higher numbers of triangles and vertices more efficiently than others. To see if triangles and vertices are the main cause of loss in performance, try changing the resolution in the Game view to a smaller value. If the Frames per Second doesn't increase, it's possible that the number of vertices and triangles being rendered will be main bottleneck on your hardware.

SetPass calls

SetPass calls refers to number of times the renderer needs to setup information about the current material being drawn. Having multiple different materials on many individual meshes will increase this number, as the renderer is constantly changing what material needs to be drawn to complete the frame.

Shadow casters

To render shadows, each individual object is rendered again but with a simpler shading. This also adds up in triangle / vertex count, so turning shadow casting off will often reduce these numbers. Combining meshes reduces the amount of individual shadow casters required, because the amount of individual meshes have also been reduced.

Combining Meshes based on Cell Sizes

The Scene Optimization tool can group together meshes based on a configurable spatial partition. The spatial partition is a 3D volume size that is configurable in the menu. Items that occupy each partition are combined based on unique materials, size and LOD groups. Due to the way that different hardware can be better suited for many pixels rendering on screen or the number of vertices, different cell sizes may be better suited for different hardware.

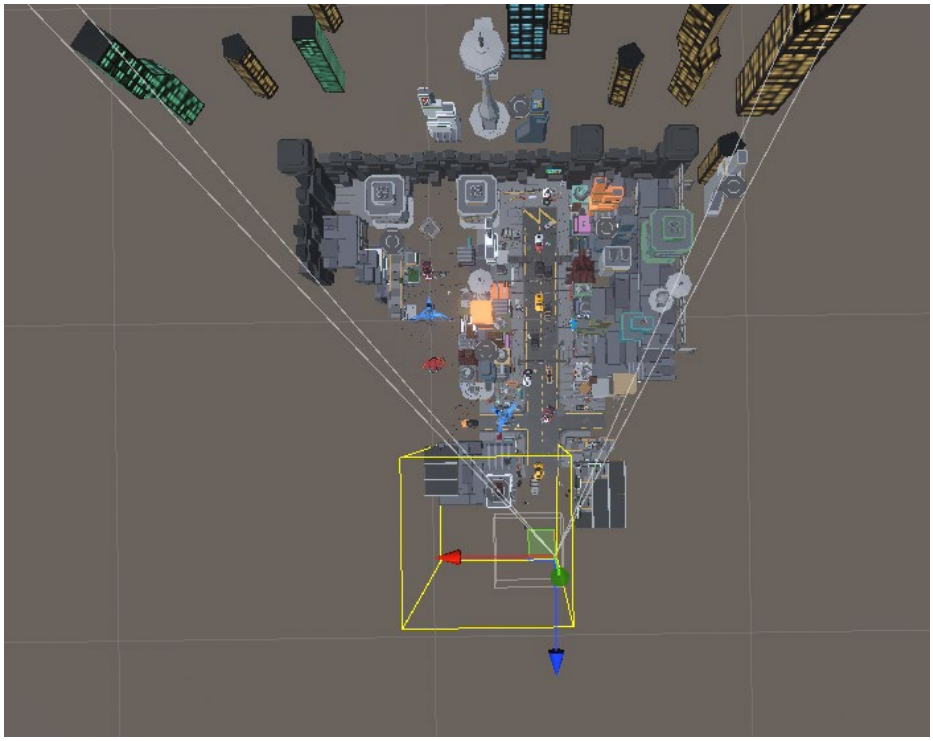
Below is a comparison table showing how the data in the statistics tab will change based on different cell sizes chosen. Frames per Second and milliseconds are excluded as this data changes between hardware.

Cell Size	Large (~128+)	Small (~8+)	Tradeoffs
Batches	Less batches than original scene	More batches than a Larger cell size, but still less batches than original	If your CPU performs better with less batches, it may be CPU bound, meaning sticking to a higher cell size may give you the best performance, all other aspects considered.
Tris / Verts	More Tris / Verts than original scene	Less Tris / Verts than Large cell size, but more Tris / Verts than original scene	The cost of having a larger cell size may be worth using if your hardware can support more tris / verts without decreasing performance.
SetPass Calls	Less Setpass Calls than original scene	More Setpass Calls than a Larger cell size, but less Setpass Calls than original scene	Much like the number of batches, if the scene runs better with less SetPass calls, it might be worth sticking with a higher cell size for best performance, all other aspects considered.

Shadow Casters	Less Shadow Casters than original scene	More Shadow Casters than a larger cell size, but less Shadow Casters than original scene	The cost of having a larger cell size may be worth using if your hardware can support more tris / verts without decreasing performance.

Frustum Culling with Cell Sizes

Because this tool combines meshes that fall within the bounds of each spatial partition, they are treated like one object when it comes to rendering. Unity's renderer automatically culls (hides) objects that do not fall into the rendered camera's view frustum. Because the combined objects are considered one object, even if one small part of it is occupying the screen, the whole object will be drawn. This causes extra vertices to be rendered even if they are not visible in the camera's view. A smaller cell size will reduce the visible size of each combined object, thus providing better frustum culling with less vertices rendered on screen. If your scene if vertex bound, i.e., performance slows down with more vertices, than choosing a smaller cell size may suit this scenario.



Frustum culling roughly visualized in a scene without any combined objects.

Frustum culling roughly visualized in a scene with combined objects Cell Size = 64. Because of the bigger bounds of the combined objects, more is being drawn.

Frustum culling roughly visualized in a scene with combined objects Cell Size = 16. It draws less vertices in total than the bigger cell size of 64, but still draws more vertices than the scene without any combined meshes.

Using the Mesh Combine tool

The combine meshes tool works on the currently selected root elements of all gameobjects that should be combined in the hierarchy. To combine meshes, the default hotkey is **Ctrl + T**.