Impostors - Runtime Optimization

2.0.0

Documentation

Online documentation:
https://docs.google.com/document/d/1CYzcjO9GQpGu3wsJInNHhWAZf7KHYPi_1Z4Lbmiybyg/edit?usp=sharing

Forum:
https://forum.unity.com/threads/released-impostors-runtime-optimization.759110/

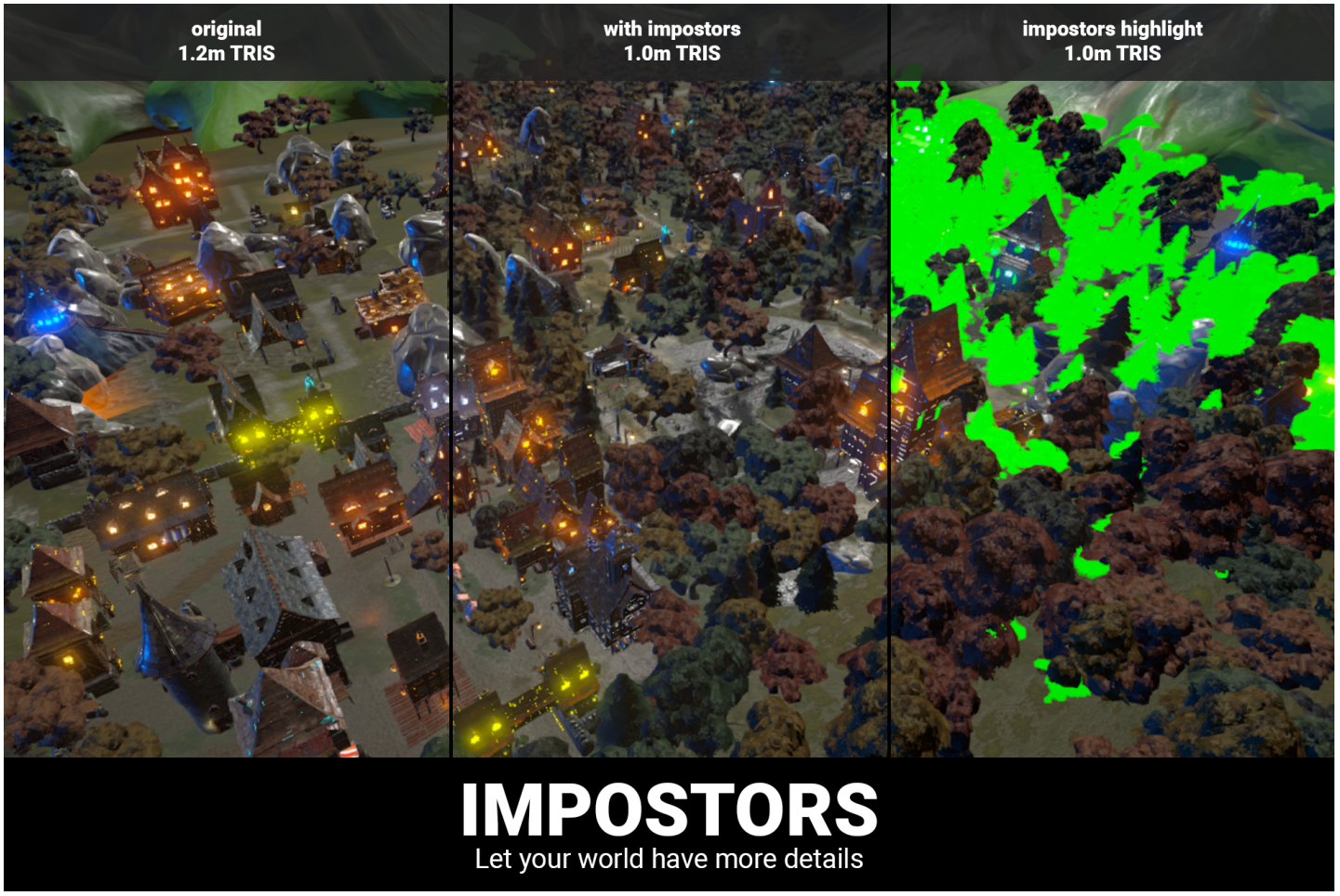


Table of contents

Dependencies

IMPORTANT: minimum required version of Unity is 2020.3.16

Package Dependencies:

- Burst
- Collections
- Mathematics

Known issues

1. HDRP is not supported.
2. Deferred rendering supported, but Impostors will be drawn using forward rendering path. If this causes problems, then please, contact me with details.
3. Built-in Render Pipeline with Scheduled impostors rendering in VR projects sometimes doesn't work properly and may cause issues. Fix: use Immediate impostors rendering.

A bit of terminology

Asset is called **Impostors – Runtime Optimization**. Most of the time I refer to it as Impostors package, or just simply Impostors with first capital letter. It's singular, not a plural term. Sometimes I also use word system.

imposter (with e instead of o at the end) – more widely used version of the same word. I use this term when referring to overal concept of imposter technique or when talking about actual imposter object/instance/texture.

– It's confusing. Wouldn't it be easier to… not do like that?
– Yes.

Contacts

Email is a preferred way, because it helps me better track and organize your requests.

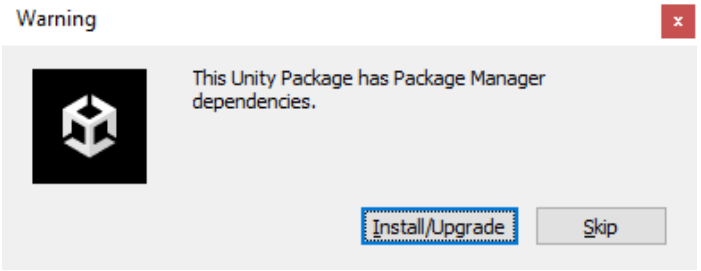
- Email - doterpoter@gmail.com
- Forum
- Private conversation on forum
- Discord

Installation

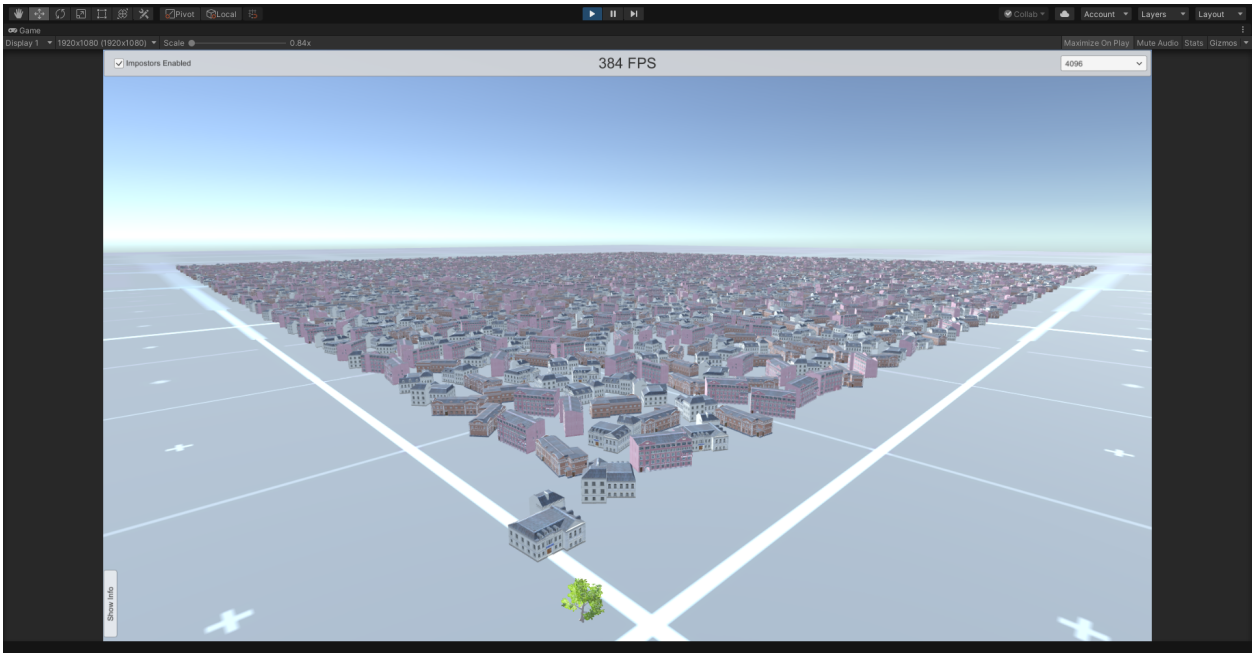
First of all, backup your project.
*Alternatively, you could create a new empty project to play around with the Impostors package using required tech-stack, target platforms and graphics APIs and see if it works for you.

Installation steps:

1. Import Impostors - Runtime Optimization using Package Manager window.
2. Unity will prompt you to install required packages.



3. Import package content.
4. Open `impostors_sample_City` scene.
5. If your project uses URP then you need to do additional setup.
6. Run example scene. Use top-right dropdown to spawn objects.



7. Verify scene works properly and example objects turns into impostors.
8. In the same scene set up some objects from your project to work with Impostors.
9. Run scene again and verify these objects work properly with Impostors.

If you have problems on any of these steps, please contact me.
Anything related to rendering in Unity is quite fragile, there are countless combinations of project settings, shaders, meshes, post processing, render pipelines, unity versions, hardwares and graphic APIs that it's just not feasible to cover all possible cases. Impostors package is heavily tested by developer and the community, but exceptions still happen requiring additional investigation of the problem.

Delete `Samples` folder

You are safe to delete the `Impostors/Samples` folder if you don't need it anymore.

Import as embedded package

Impostors package is UPM compatible, so after import you can reorganize it as upm-embedded package:
https://docs.unity3d.com/Manual/upm-embed.html

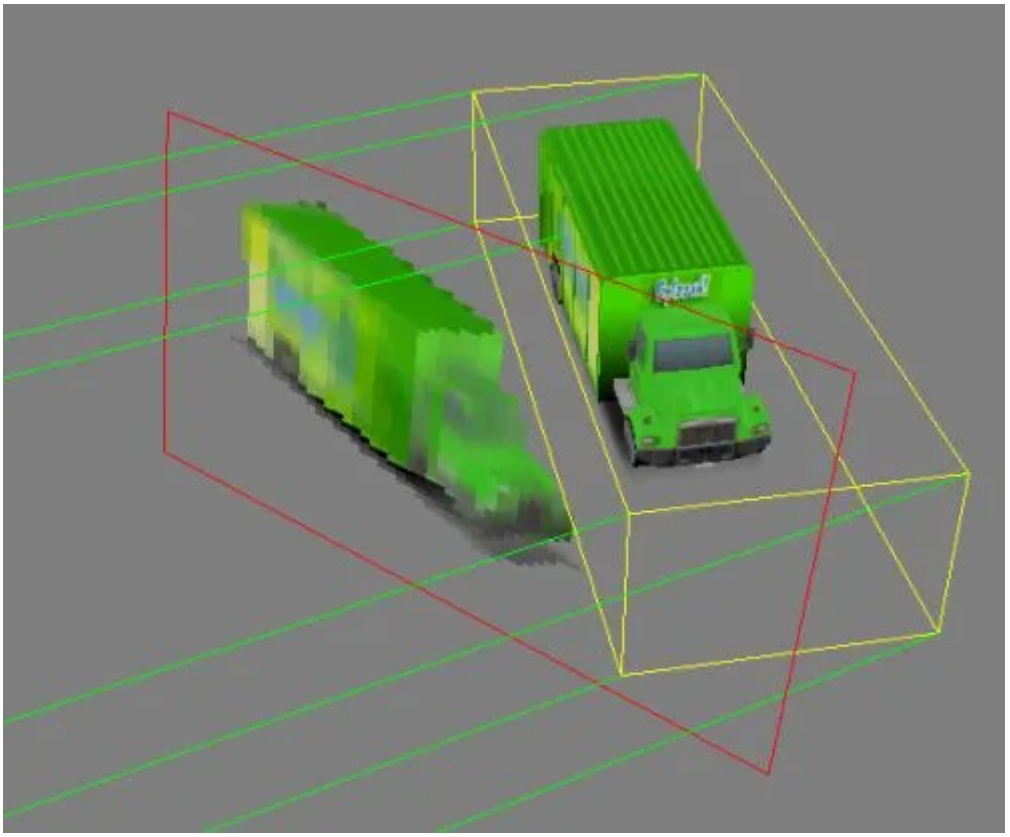
Upgrading from v1

Unfortunately, you cannot just import new version on top of existing v1 because this will create junk and duplicated assets.
To properly upgrade to v2 follow these steps:

1. **BACKUP your project.**
2. Create and open new scene using `File/New Scene`. There is no need to save this scene to disk. This is just an intermediate environment to keep safe from missing scripts.
3. Open Package Manager window and find Impostors - Runtime Optimization asset in the list.
4. Make sure Package Manager downloaded the latest available version.
5. Keep Package Manager window open.
6. Go to Unity's Project view and completely delete `Assets/Impostors` folder.
7. After deletion you may see compilation error but that's ok.
8. Don't do anything else and just import new version from Package Manager window.
9. Unity may ask to install/upgrade packages.
10. Wait for package import and script compilation.
11. Open 'impostors_sample_City' scene and run it. Verify it works properly.
12. Open your own scene that uses Impostors. Right after opening scene will be marked as dirty. Save the scene.
13. Enter playmode and verify imposters showup and work as before.

Concept explained

The main idea of imposter technique is to replace complex mesh geometry with a pre-rendered snapshot of the object.



Example of imposter. Source.

There are many implementations of the imposter technique, but ultimately they fall into two main approaches:

- Imposter texture is baked before game runs. (more info)
- Imposter texture bakes on the fly when game is running. (more info)

There are pros and cons to both approaches. But this package focuses on the second one.
I've spent last 7 years squeezing out max performance gain from using imposters.
Impostors package solves fundamental issues of creating such system.
But more importantly, it adds crucial pieces to run smoothly on any* device with any** number of objects in the scene. It does:

- Per frame load distribution
- Pooling resources for reuse
- Multithreading
- Imposters' textures atlasing
- Imposters' mesh combining
- Utilizes the nature of GPUs
- Utilizes CPU cache and Burst compiler.

Impostors focuses on one thing and does it well. Period. (epic explosion)

*any – device that supports render texture, which is almost every device nowadays

**any – any reasonable number of objects, 10.000 and more depending on device.

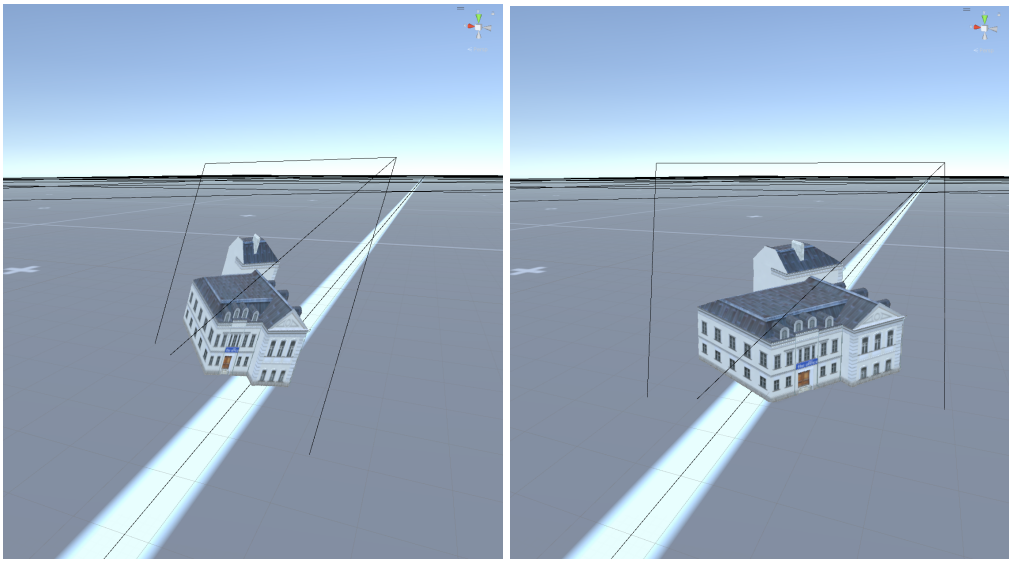
How it works

Unity's LODGroup controls ordinary object rendering prior to cull distance.
This ensures that Unity's internal systems such as occlusion culling, reflection probes, batching, instancing, etc are not affected by Impostors package.
If you are unfamiliar with LODGroup or LOD concept, please watch this video or read Unity's docs.

When object exceeds LODGroup's cull distance Unity hides original object.
At this point Impostors system starts drawing object's imposter.
Imposter is drawn until object exceeds ImpostorLODGroup's cull distance.
At further distance object is completely culled/hidden.

System knows about all active ImpostorLODGroups present in the scene.
Whenever camera is about to render the OnPreCull event is called, Impostors hooks to this event and does culling calculations to identify which imposters should and should not render, and which imposters textures to update.

Imposter texture update – is a process of creating a new snapshot of object that will be used instead of current imposter texture. Imposter is marked as RequiresTextureUpdate when its visual error exceeds specified limits. Visual error comes from camera movement, sunlight direction change and other configurable conditions.



Left – imposter with overly exaggerated visual error, clearly needs an update.
Right – happy imposter after update looking as intended.

When camera moves rapidly you can imagine how many imposters require such an update.
Spoiler: a lot.
Unfortunately, we can't afford update all of them at once, otherwise, it would be easier to just draw ordinary objects without all this imposter blsht.

That's why the next step is sorting process. It selects N imposters with the largest visual error among others, where N is a number that could be configured. These N imposters receives texture update. And other imposters wait for the next chance to be included into the update queue.

There is a ton of other stuff to cover, but these are the most important parts.

When to use

General note

Impostors works best for GPU bounded projects.
It drastically reduces triangle and drawcall count.
However, it means that if your game is already extremely-low-poly and utilizes GPU instancing, then imposters won't help much.
But in all other cases you should experience noticeable performance gains when applied properly.

1. Camera rarely changes position

The best scenario for impostors!
Updating imposter's image is a rendering operation, which we are trying to avoid as much as possible. When camera doesn't move we don't need to update anything.
Genres: VR with teleportation movement or steady gameplay, visual novels, side-scrollers.

2. First-person character camera

Works perfectly with impostors!
Genres: open-world FPS, battle-royal, survivals.

3. Flying or top-down camera

Such camera behavior requires constantly update snapshots of impostors, which leads to way less performance boost. However, I can't say that impostors cannot be used in such games! Example scene shows exactly that scenario.

4. Fast-paced camera movement

The weak part of Impostors package.
Games like racing and flying simulators might not work with Impostors as good as others. It depends on what you are trying to achieve. Feel free to contact me and we will try to understand would your project benefit from using Impostors.

5. Games with procedurally created objects

If your game has procedurally generated meshes, I doubt you have LODs for them:)
In such a situation Impostors package comes handy because it generates everything at runtime!

6. Low-poly games

It depends on the platform you targeting and the count of objects you want to present in your scene. Generally speaking, impostors are overkill for games with a low poly count.

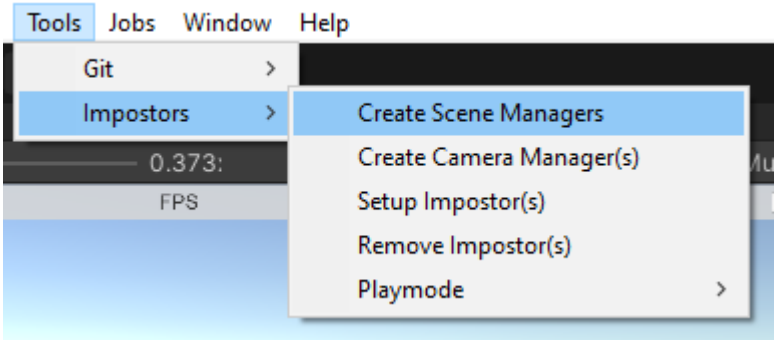
What system doesn’t do

System doesn’t unload unused assets from memory.
For example, you have one big unique tree that is placed at the center of some fantasy city. You use imposter on this tree to draw it from a further distance. But at some point, this tree is so far away from the player that it’s insufficient to render it even as imposter. Tree becomes completely culled. Even though player can’t see it, it is still in memory consumes player’s RAM. Yeah, that’s unfortunate. But loading/unloading assets is a whole other topic called world streaming.

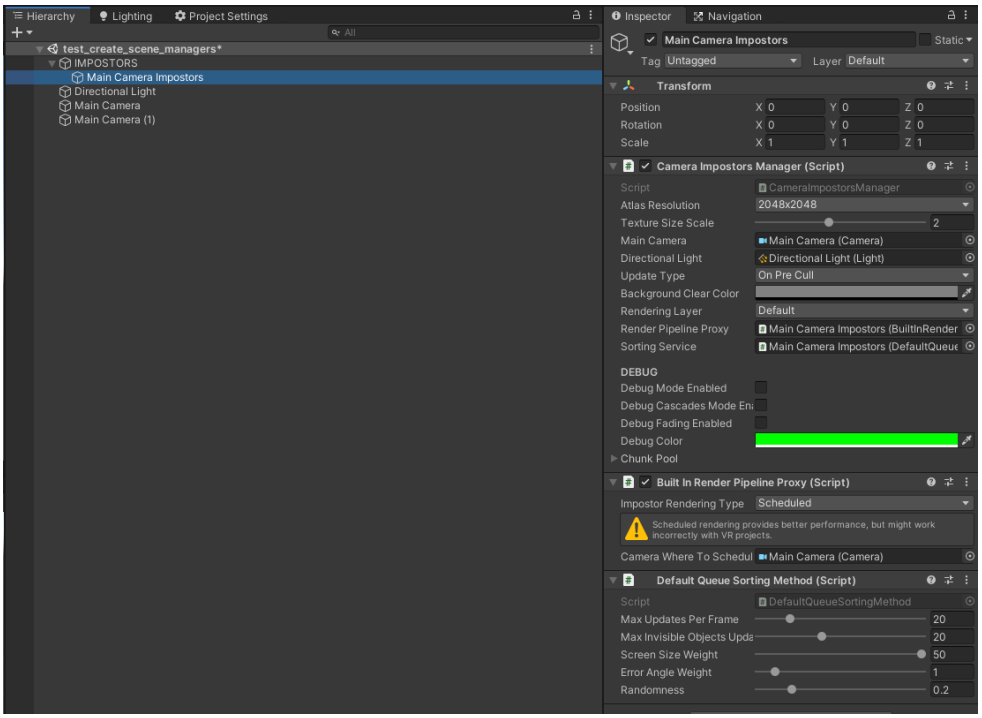
How to use

Setup Scene Managers

IMPORTANT! Please, backup your project before proceeding with this step.
Use `Tools/Impostors/Create Scene Managers` to set up your scene to use impostors.



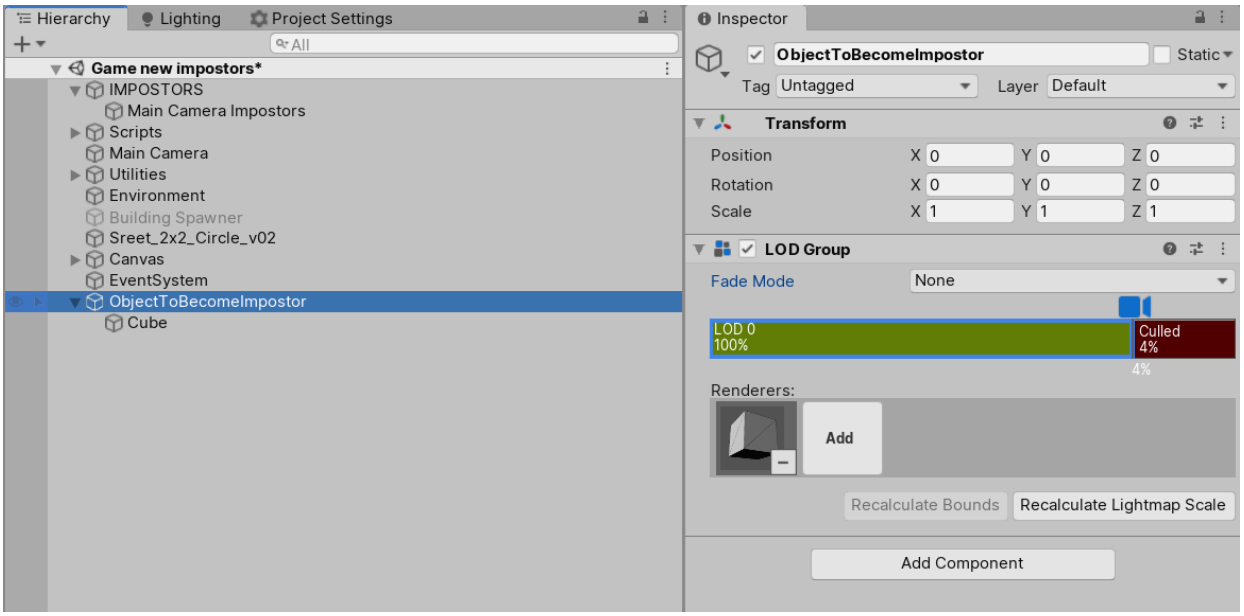
This will create a new game object “IMPOSTORS” at the top of your scene hierarchy.
Under this game object will be one child, which setups the main camera to render impostors. If you need more cameras to render impostors, then use the `Tools/Impostors/Create Camera Manager(s)` command.



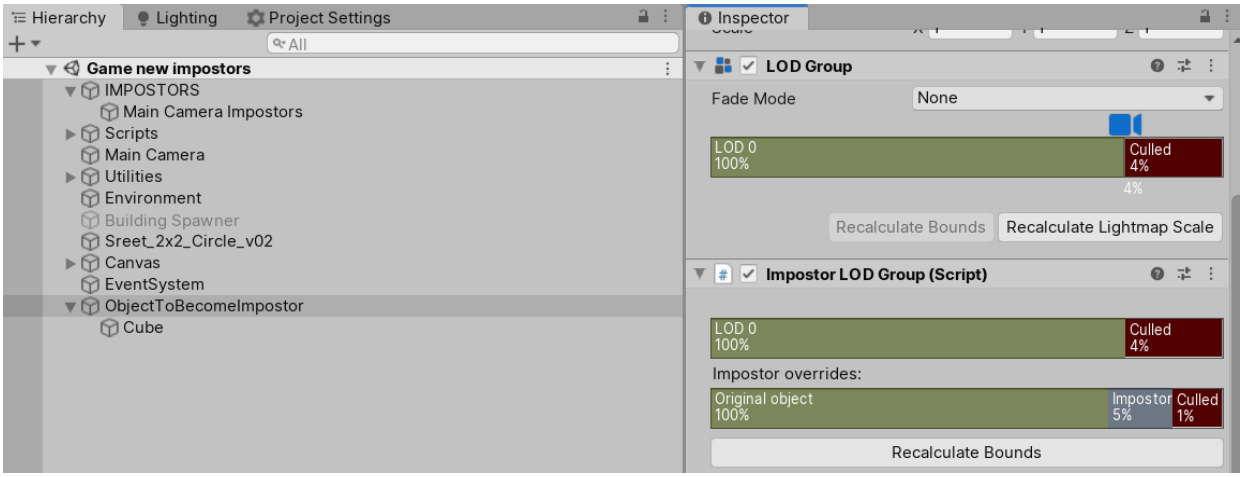
Make sure the RenderPipelineProxy is referenced in CameraImpostorsManager.
If you are using **URP** read this.

GameObject setup

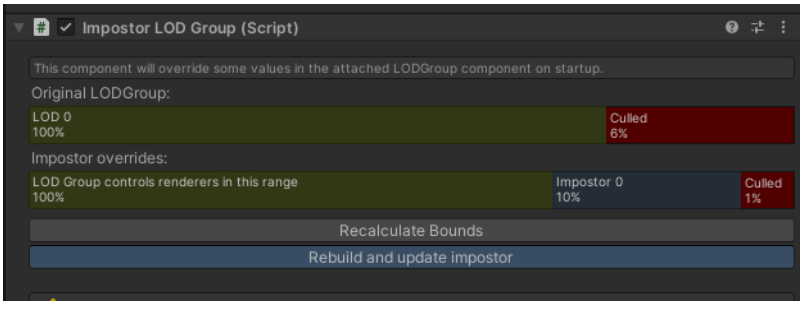
After scene setup, you need to mark objects you want to render as imposters with the `ImpostorLODGroup` component.
To work properly this component **requires a built-in LODGroup component** on the game object.
Your LODGroup has to contain at least one LOD. For example, this setup is ok:



When your object has a LODGroup component, use `Tools/Impostors/Setup Impostor(s)` to automatically handle setup of the `ImpostorLODGroup` component.
After this command gameObject's inspector should look like this:



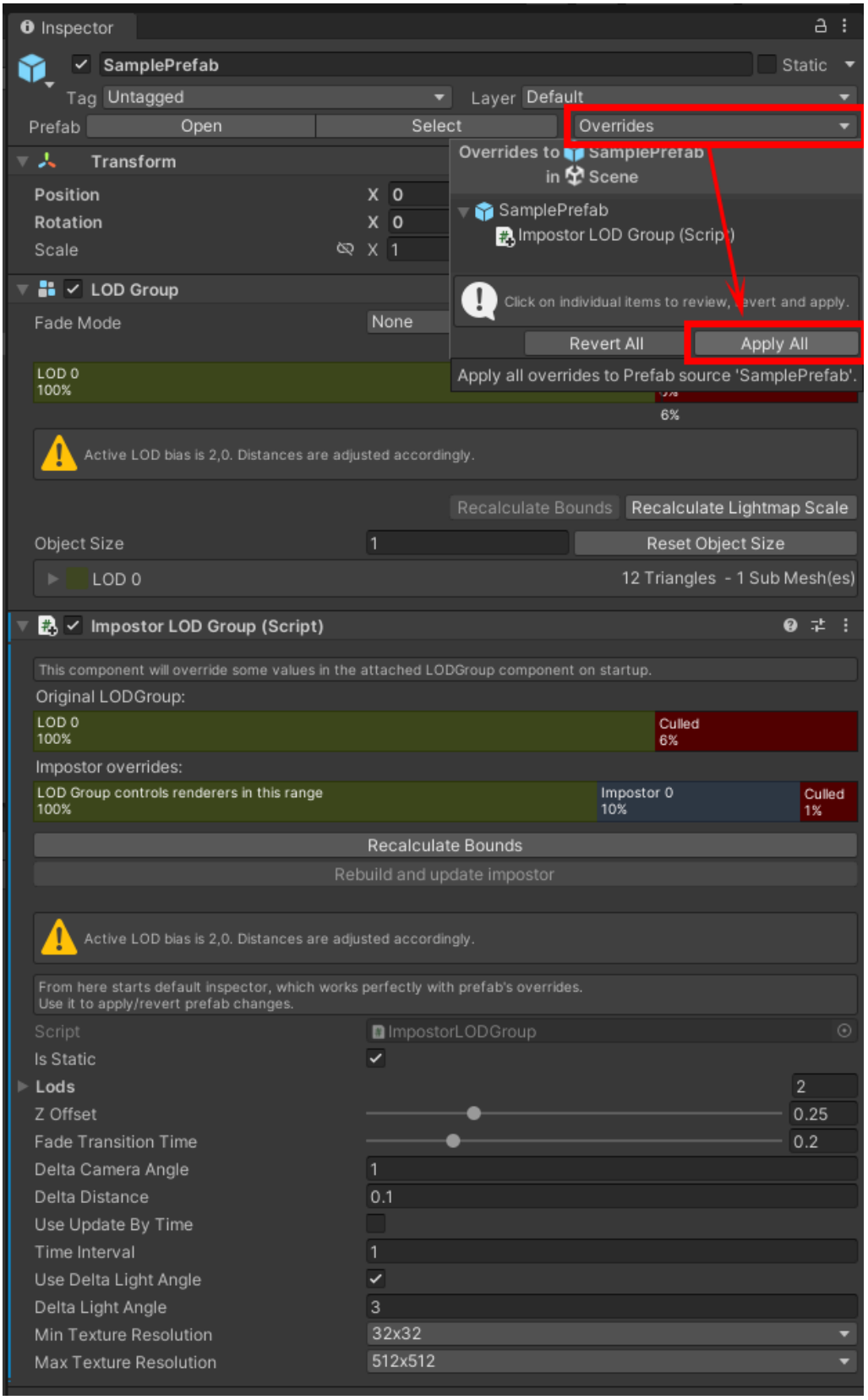
You can change properties as you want. Please note, these properties are cached, so changes are not reflected at runtime right away. To apply new settings at runtime press `Rebuild and update impostor` button.



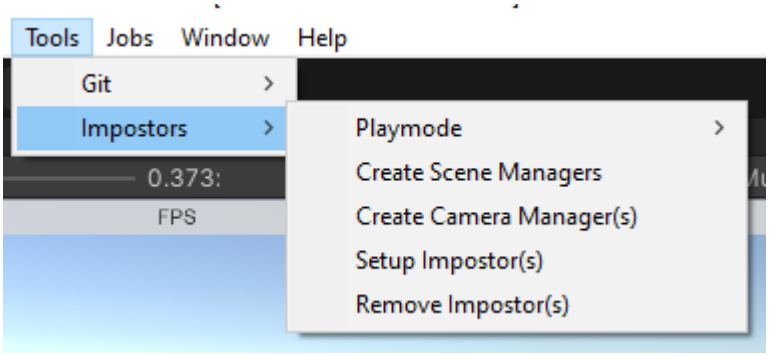
Now enter playmode and object should turn into impostor at some distance.

Prefab setup

You can set up prefab the same way as the regular GameObject from the previous section. If you are doing this in the scene, don't forget to apply changes to the prefab asset.



Menu items for automatic setup



Create Scene Managers

Creates managers in the current scene that are required for Impostors to function.

Create Camera Manager(s)

Impostors package supports multiple cameras. To set up additional cameras: select game object(s) with Camera component and run this command.

Setup Impostor(s)

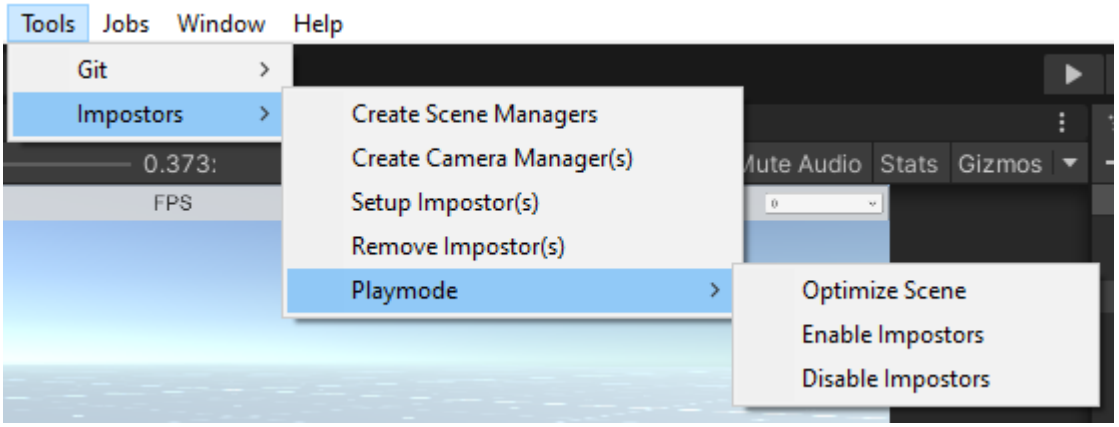
you can select one or multiple game objects in the hierarchy, and this command will try to automatically set up the ImpostorLODGroup components onto selected objects.
Note: objects must have unity's built-in LODGroup component to work properly with this command.

Remove Impostor(s)

Opposite command that removes all Impostors components from selected objects.

Playmode menu items

Under “Tools/Impostors/Playmode/...” you can find helpful operations and shortcuts.



Use at play mode
These commands are **designed to be used at runtime**, when you are running your scene. This way is preferable, because you can undo all setup operations just by stopping the play mode and prevent any chances to corrupt your scene.
This is handy when you want to quickly look at what Impostors can do with your scene.

Optimize Scene

In a nutshell, it finds all LODGroup components in the current scene and applies the “Setup Impostor(s)” command on each of them.
Warning: this command is quite heavy and might take some time. For example on 10k objects it may take 20 seconds. This setup process is required only once and doesn't impact runtime performance.

Enable Impostors

Enables all ImpostorLODGroup components in the game.

Disable Impostors

Opposite command that disables all ImpostorLODGroup components in the game. Useful when you want to look at your scene without Impostors.

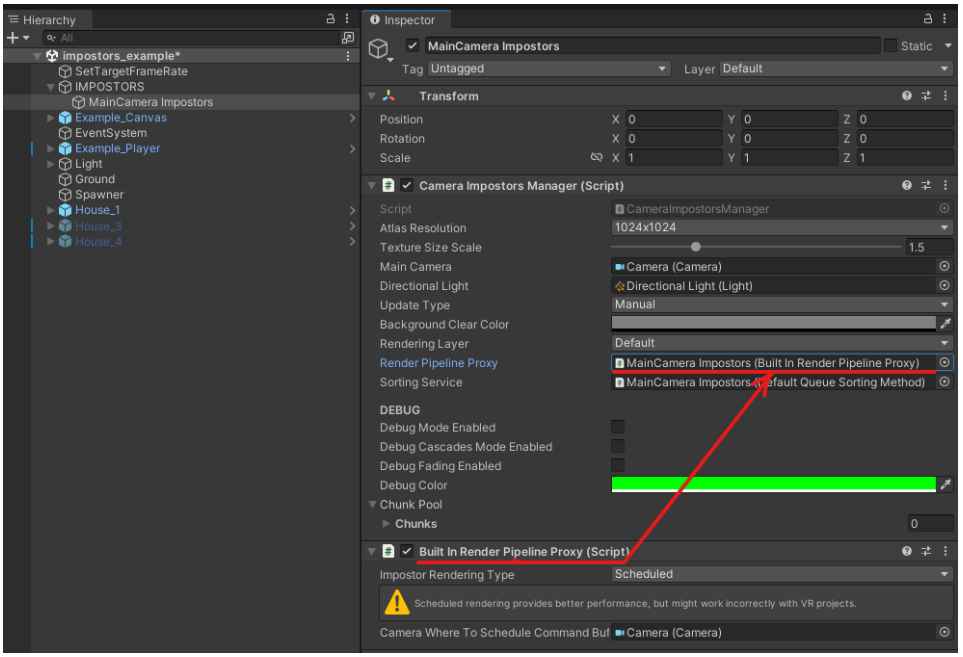
Render Pipelines

Impostors package uses RenderPipelineProxy to support multiple render pipelines.
Package provide 2 pre-made proxies:

- BuiltInRenderPipelineProxy - default
- UniversalRenderPipelineProxy

Built-In Render Pipeline

Impostors package requires the `BuiltInRenderPipelineProxy` component to work with built-in RP.
Make sure reference is set in the inspector. Drag proxy into corresponding field if it is not set.

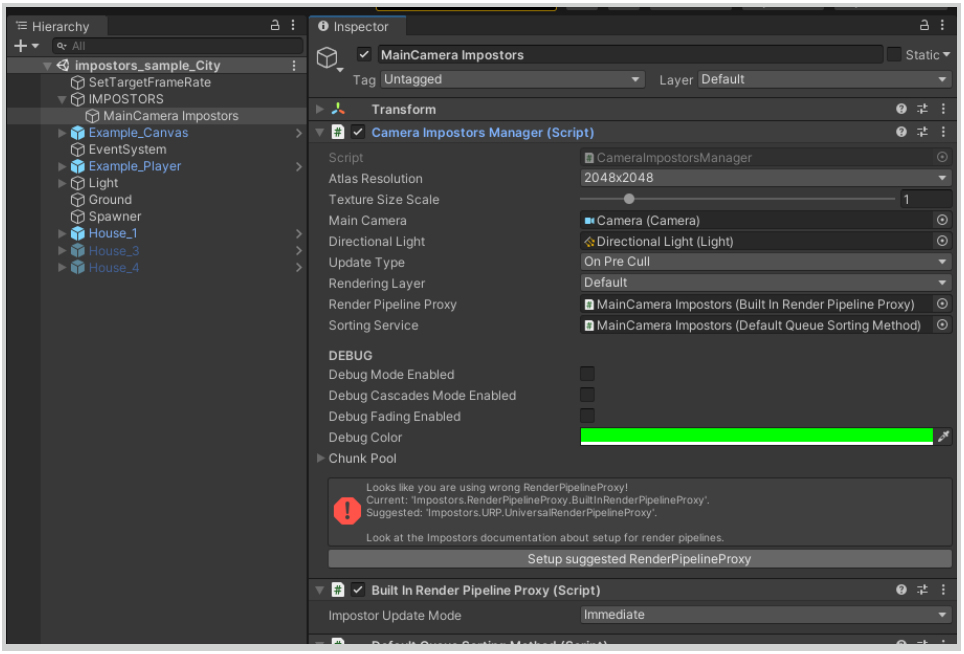


Universal Render Pipeline **(URP)**

Impostors package supports URP but requires additional setup.

Step 1 - setup URP proxy

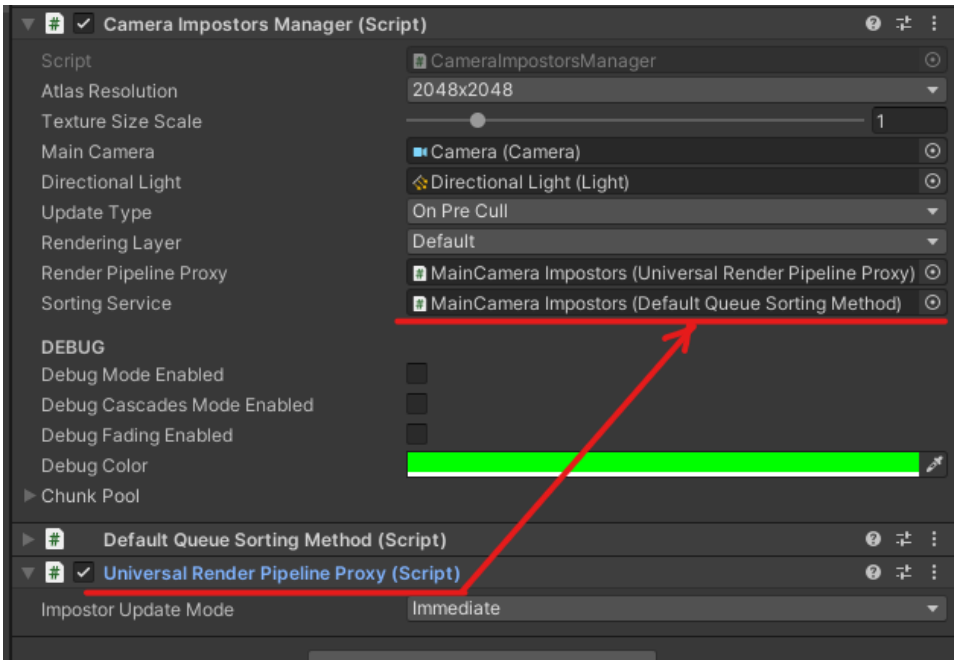
Select gameObject with `CameraImpostorsManager`.



You'll see error box at the bottom of the component. You can either manually remove `BuiltInRenderPipelineProxy` and add `UniversalRenderPipelineProxy` component or

Just click on `Setup suggested RenderPipelineProxy` which does the same but automatically.

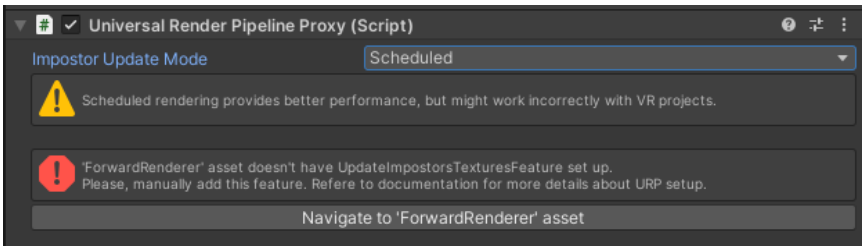
Result should look like this:



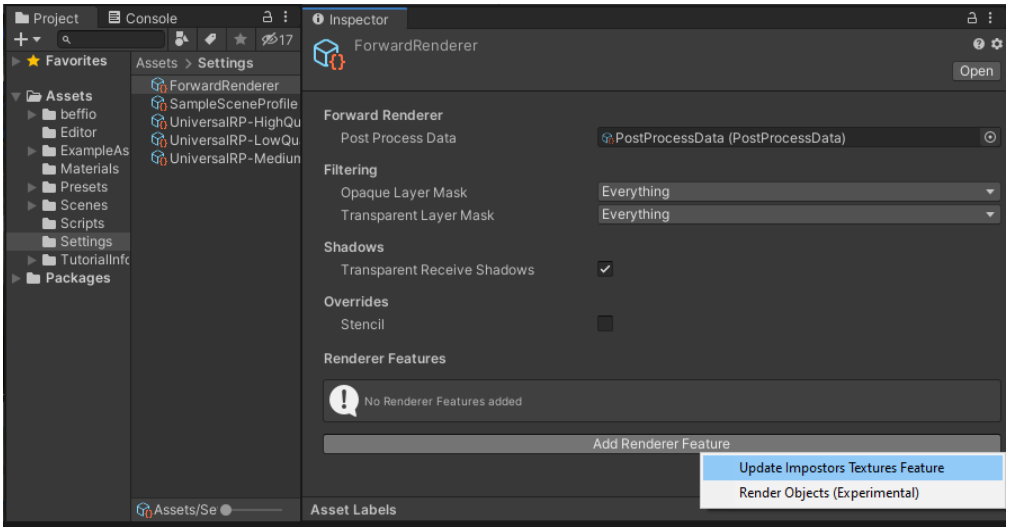
Make sure that Render Pipeline Proxy is specified on CameraImpostorsManager.

Step 2 - setup render feature (optional)

By default, URP proxy usses Immediate mode rendering which doens't require setting up render feature. However, for slightly better performance and ability to debug in Frame Debugger window you can setup render feature.
Set Impostor Update Mode to Scheduled. Inspector will notify about setup issues.



Select your Renderer Data asset. By default it is located under "Assets/Settings/" folder.
Click on "Add Renderer Feature" and select "Update Impostors Textures Feature".
IMPORTANT! Right after adding new feature click **Ctrl+S** to save changes made to renderer asset. Otherwise feature won't apply and show up in the project view.



Repeat this process for each Renderer Data asset that are used in your project.

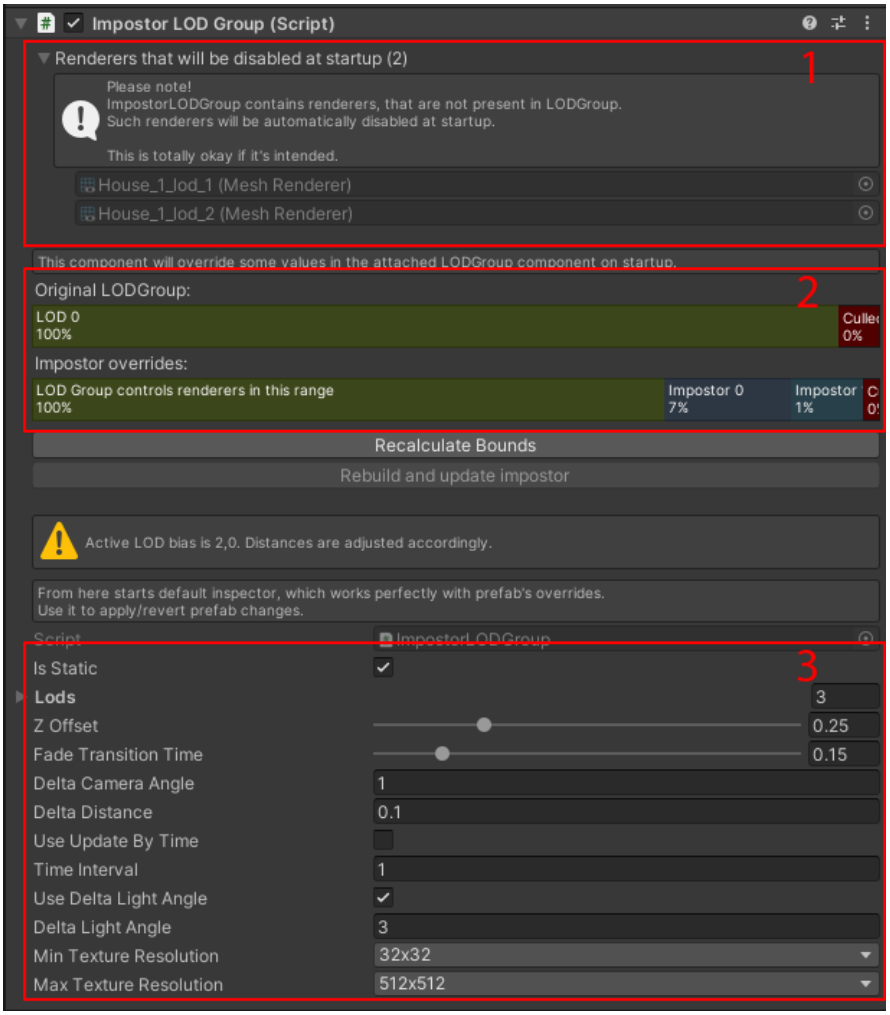
Custom Scriptable Render Pipeline

Impostors package doesn’t officially support custom SRPs.
But you can provide custom Render Pipeline Proxy.
For this you just need to extend the `RenderPipelineProxyBase` class with your own implementation.
Don’t forget to set reference to proxy in the `CameraImpostorsManager`.

Components explained

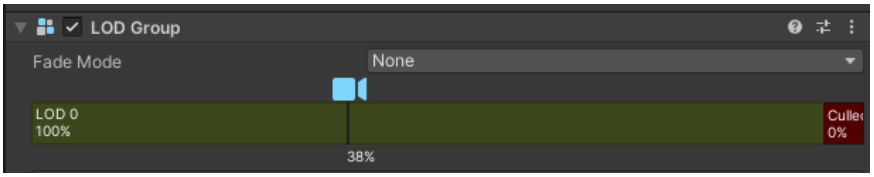
Impostor LOD Group

Serves as a marker to let system know that this object should render as imposter.
Tries to mimic Unity's LODGroup component's look and behavior for ease of use.

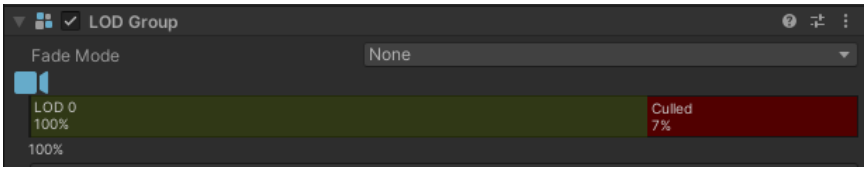


Section 1
Contains some handy messages about object's setup. Look for them, they are here to help:)

Section 2
Here you can set ranges where imposter will be drawn.
Original LODGroup's ranges are here just for a reference. This is not interactable.
Impostor overrides defines where imposter starts rendering and when object is completely culled.
You can configure these ranges as you do in LODGroup. It's called `overrides` for a reason: when game starts, LODGroup's ranges will be overridden so that ordinary object won't render simultaneously with imposter.
For example, here is how LODGroup look at edit time:



And after game is started:



Notice that now it culls at 7% instead of ~0%. This 7% comes from first imposter LOD – Impostor 0.

Section 3
Settings controlling how often imposter will be updated. The less often imposter updates the better performance, but worse visuals. Objective is to find proper values so that imposter looks good and doesn't update more than once in 4 frames.

Is Static – Enable this if object won't move at runtime. This omits some transform calculations.

Lods – Collection of imposter LODs. This is a raw representation of Impostor overrides from above. Use this to set precise transition values and apply changes to prefab. First LOD must be empty.

Z Offset – Determines how far generated imposter quad positions from center of this group. This is useful to prevent imposter ground penetration. Default value of 0.5f works in most cases.

Fade Transition Time – Time in seconds that is needed for impostor to fade in/out when impostor changes texture.

Delta Camera Angle – Angle in degrees that determines how much direction from camera should change to cause texture update. The less this value, the more often imposter's texture will be updated. Use value of 1 for important objects. Up to 5 for semi-important.

Delta Distance – Determines relative distance change between camera and object that will cause texture update. The less this value, the more often impostor's texture updates.

Use Update By Time – Enable this to update imposter's texture over time.

Time Interval – Time in seconds after which imposter will request update. The less this value, the more often impostor's texture updates. Enable 'useUpdateByTime' to take this setting into account.

Use Delta Light Angle – Enable to update imposter's texture when main directional light changes direction.

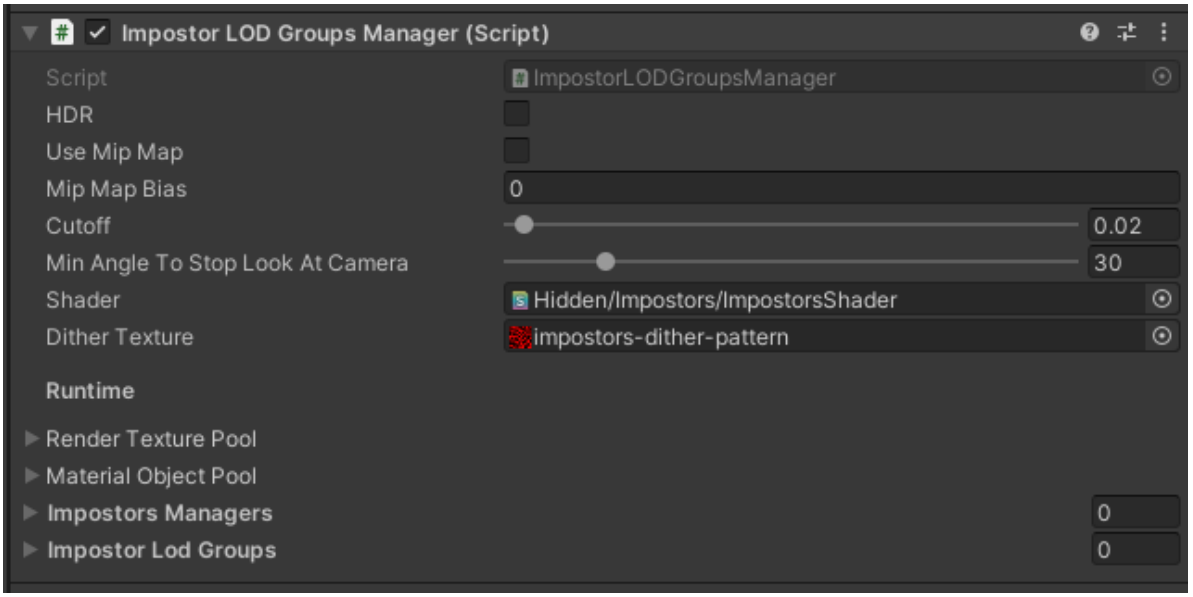
Delta Light Angle – Angle in degrees that determines how much direction of main light should change to cause texture update. The less this value, the more often impostor's texture updates.

Min Texture Resolution – Sets min allowed imposter texture. 32x32 by default. Increase it in cases when object stops rendering at low resolution. Tree leaves tend to behave this way. Better to keep this value as low as possible.

Max Texture Resolution – Sets max allowed imposter texture. 256x256 by default. Decrease it in cases when you absolutely sure object should never consume such a big texture. Better to keep this value as low as possible.

Impostor LOD Groups Manager

Manages all active instance of ImpostroLODGroup component.
Stores global settings applied to whole system.



HDR – Enable this if you want imposters to render HDR values e.g. materials with emission. Disabled by default because consumes noticeably more memory.

Use Mip Map – Enables mipmaps on imposter textures. Legacy feature, does almost nothing to imposters look. Disabled by default, doesn't affect performance that much.

Mip Map Bias – Sets mipmap bias to all imposter textures. If distant imposters looks blurrish try set slightly negative value like -0.25. By default 0. Does nothing when UseMimMap is disabled.

Cutoff – Sets global cutoff value applied to all imposters. Increase to make imposter's edges sharper. In most cases it should be 0.02.

Min Angle To Stop Look At Camera – Increase this value if imposters are weirdly rotating when camera look down from above.

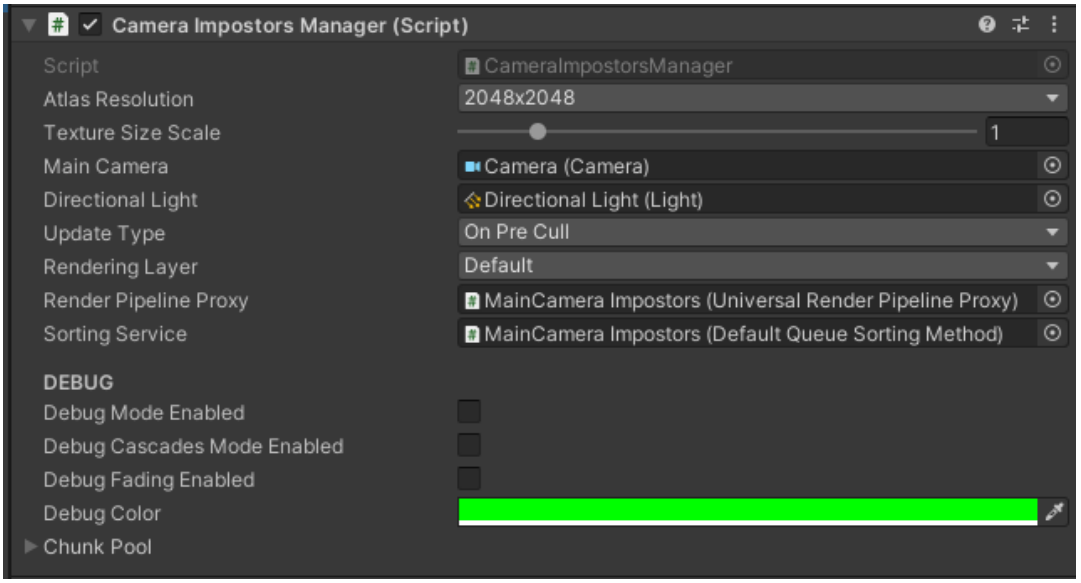
Shader – Shader that will be used to draw all imposters.

Dither Texture – Pattern that controls how imposters crossfade.

Runtime – section with runtime values, useful for debugging purposes.

Camera Impostors Manager

Contains settings that will be used for target camera.



Atlas Resolution – Resolution of atlas that contains imposter textures. 2048x2048 is a default value. Decrease it in case device doesn't support such a big textures.

Texture Size Scale – Multiplier to control per imposter texture scale. Use value bigger than 1 for more crisp imposters. Use less than 1 to make imposters a little blurier but decrease memory usage.

Main Camera – Camera for which imposter will be drawn.

Directional Light – The main directional light of the scene. In most cases sun or moon.

Rendering Layer – Imposters rendering layer.

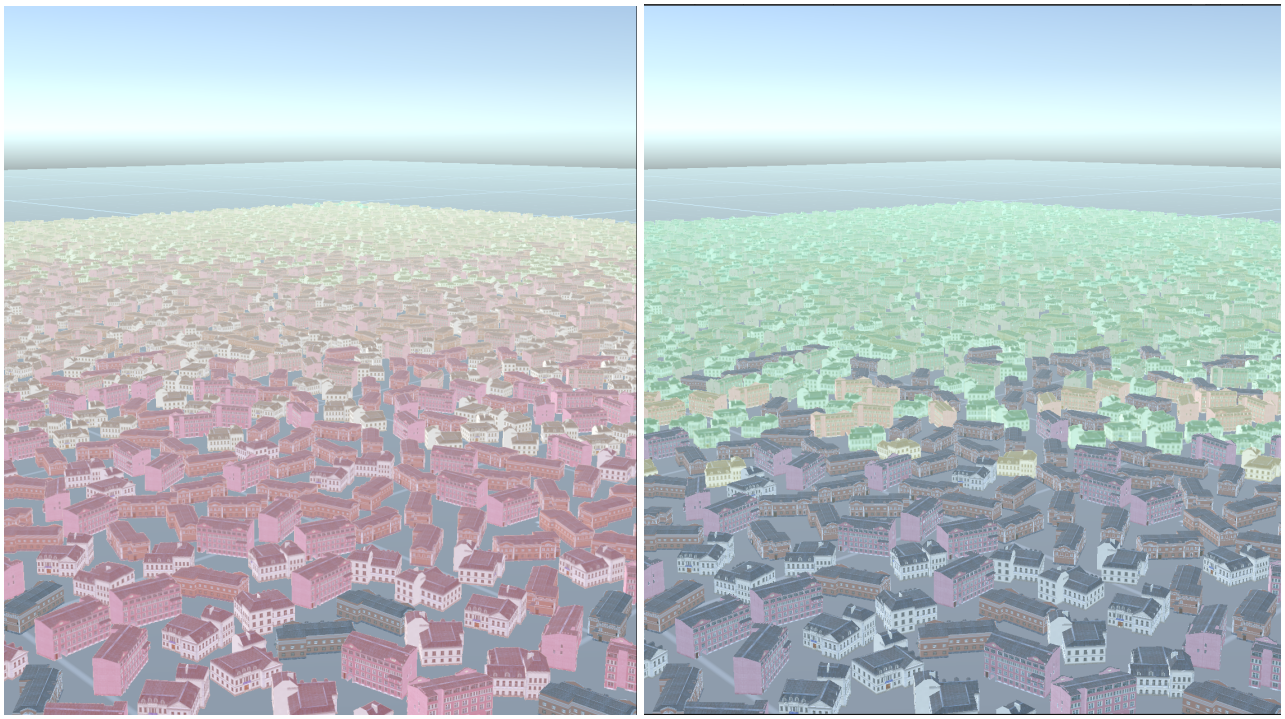
Render Pipeline Proxy – Reference to appropriate render pipeline proxy.

Sorting Service – Reference to appropriate queue sorting method.

DEBUG – Fields to help debug imposters behavior for target camera.

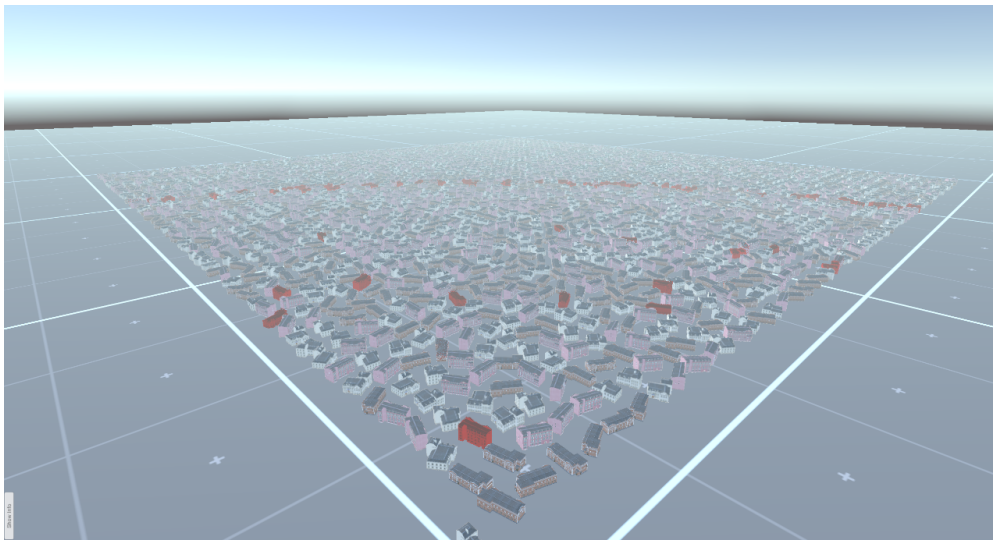
Debug Mode Enabled – Enable this to highlight imposters with Debug Color.
When enabled imposters also rendered for the Scene View camera.
All mode below doesn't work if this settings is not enabled.

Debug Cascades Mode Enabled – Uses different colors to show imposter`s textures resolution. 512, 256, 128, 64, 32. Use to verify that imposters have reasonable resolution.



Bad(left) and healthy(right) state in cascades mode

Debug Fading Enabled – Enable to highlight imposters that are fading. Imposter fades when texture is updated.

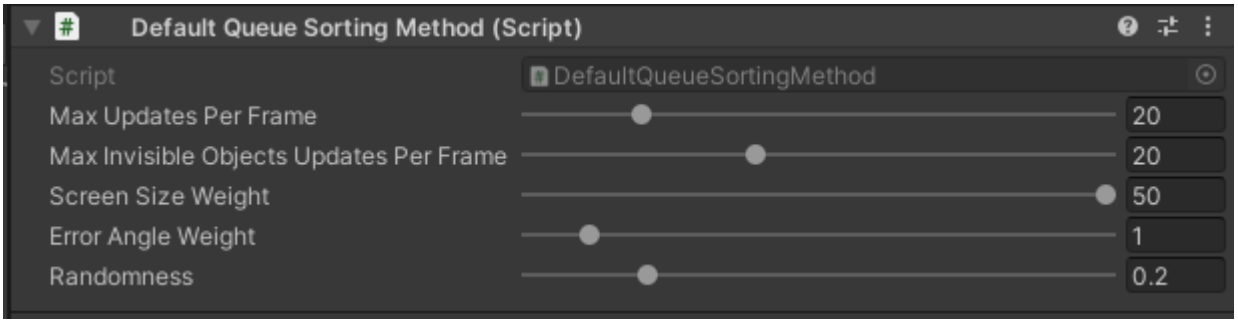


Debug Color – Color used to tint imposters when debug mode is enabled. Set it to black to remove tint.

Chunk Pool – For advanced debugging purposes.

Default Queue Sorting Method

Important component that controls system load on GPU and decides which imposters to update first.



Max Updates Per Frame – Controls how many imposters should be updated per frame. 20 by default. Most of modern middle-end PCs could handle up to 100 updates.
On mobile platforms it's worth testing but 20 should be enough in most cases.
If camera is moving relatively slow (not flying etc) you can set quite low values to decrease load even more. There is no universal value, each case requires finetuning a bit.

Max Invisible Objects Updates Per Frame – By "invisible" it means objects that are not in camera's frustum(outside the screen, behind the camera). No invisible objects updates are performed when regular updates exceed Max Updates Per Frame.

Screen Size Weight – Controls how imposter's size on screen affects update priority.

Error Angle Weight – Controls how imposter's visual error angle affects update priority.

Randomness – Adds a bit of randomness into sorting to prevent noticable update pattern over large areas.

Best practices

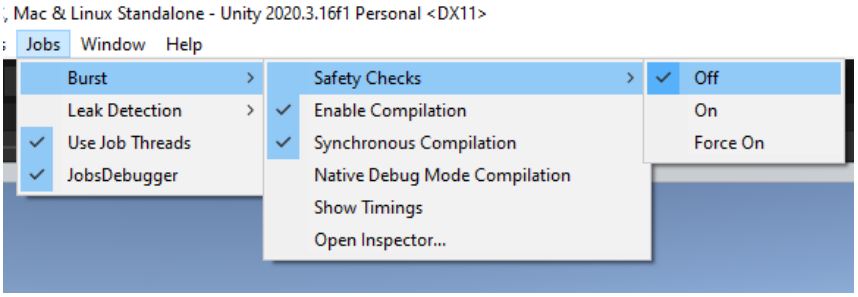
Mobile devices

Mobile GPUs are very different from standalone GPUs, so it requires additional know-hows on setting up your project using the Impostors package.

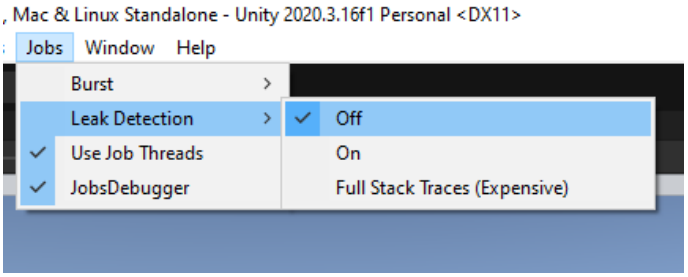
1. Don't use HDR impostors if possible.
2. Don't use Atlas Resolution bigger than 2048. 1024x1024 provides best performance on mobile devices.
3. Limit `MaxUpdatesPerFrame` to something like 20, not more.
4. It's better to use `TextureSizeScale = 1`. Or you can use values below 1 to save some memory. You can use `debugCascadesModeEnabled` on device to investigate impostors' texture sizes.
5. On mobile devices scheduled impostors rendering is crucial to achieving max performance.
6. In my tests and on my devices I got better performance using OpenGLES3 GraphicsApi in comparison to Vulkan. It depends on your project and devices you are targeting so it's worth testing.

Max performance by disabling safety checks

By default Unity keeps safety checks enabled, but they slow down Impostors system more than 2 times. Disabling them is easy:



Also, there is Leak Detection that also could be disabled to max out performance.



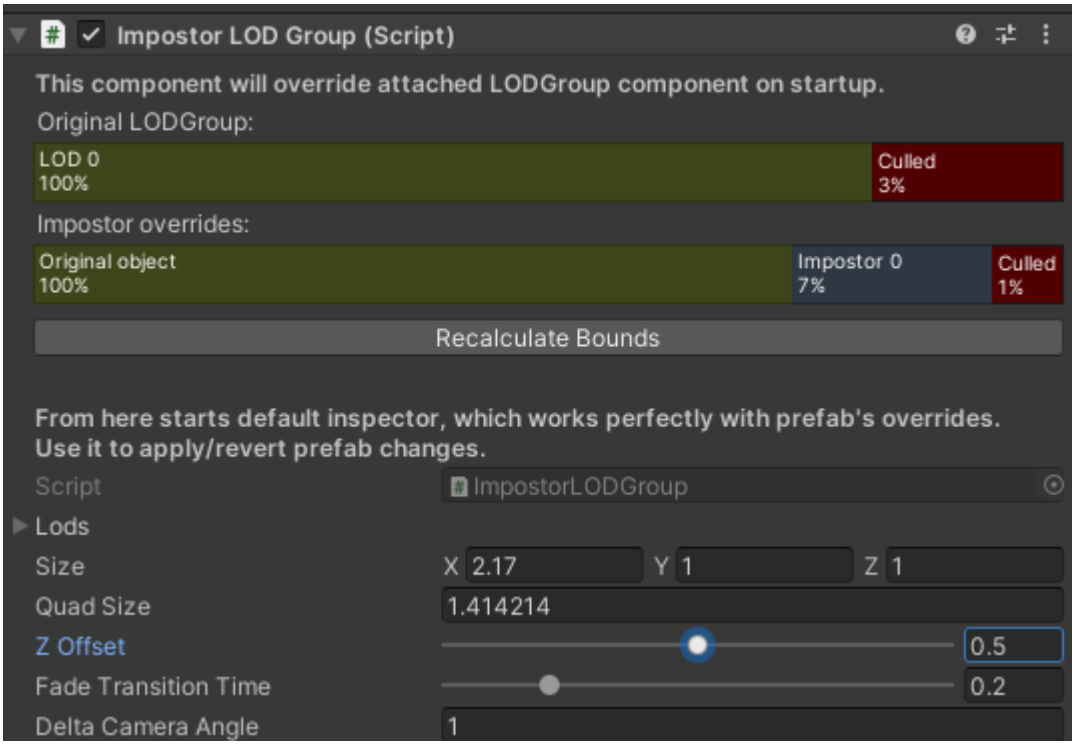
Use MSAA to smooth out imposters edges

Impostors shader is a cutout (AlphaTest) shader, that's why it suffers from the same problems as other cutout shaders (vegetation for example). There is a way to improve cutout visuals by using AlphaToMask (AlphaToCoverage) feature that works only when MSAA is enabled. Impostors shader supports this technique. Even 2x MSAA provides noticably better visuals.

FAQ

Bottom part of the imposter is under the ground. How to fix?

It's a common issue and fix is very simple. Exit play-mode, go to your object with ImpostorLODGroup component. Increase Z Offset and try to run the scene again. Problem should be fixed.



How to spawn/destroy objects with ImpostorLODGroup at runtime?

Simply Instantiate()/Destroy() them.
Additionally, `ImpostorLODGroup` has method `MarkGameObjectWillBeDestroyed()` that removes waste allocations on destroy.

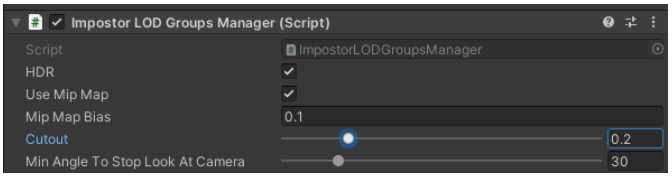
How to setup object to use the ImpostorLODGroup component at runtime?

Take a look at `ImpostorsEditorTools.SetupImpostorLODGroupToObject()` method.
This is an example on how to properly add and setup ImpostorLODGroup to gameObject at runtime.
If you are instantiating prefabs then prefab assets can have ImpostorLODGroup setup previously in editmode.

Impostors have outlines. How to fix?

In previous versions of Impostors package prior to v1.1.0 there was issue with imposters outline. Now it shouldn't be the case.
However, if you are facing this problem then please report this with some video/screenshot.

Also, you can increase 'Cutoff' setting on `ImpostorLODGroupsManager`.

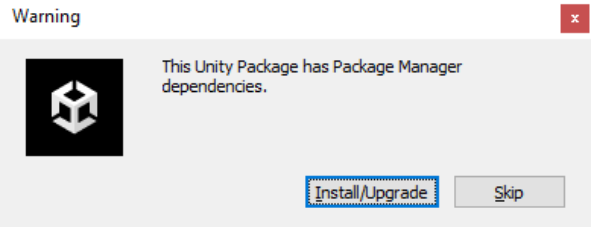


But this will lead to harsh imposters' edges.

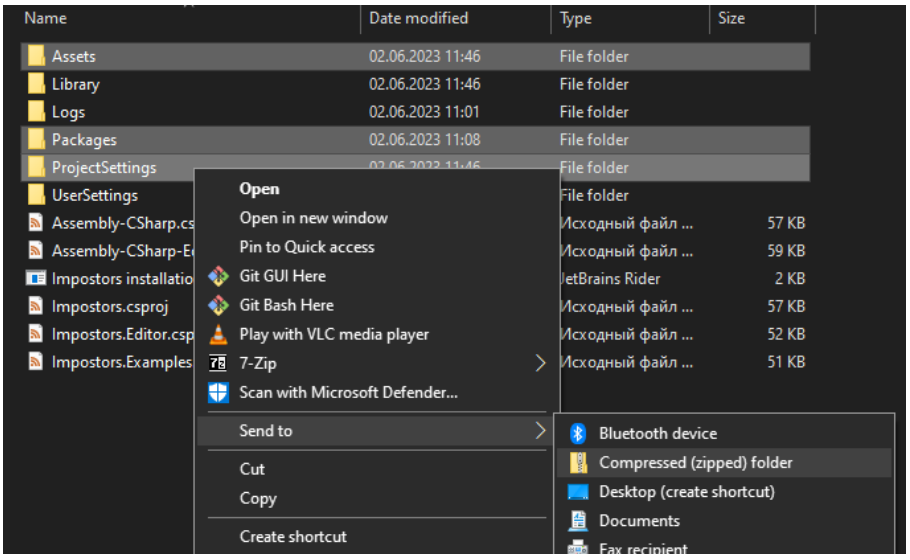
How to create reproduce-issue project

The easiest way would be to send the whole project that has a problem.
If it's not possible due to NDA, project size, or other reasons, then please, follow these steps:

1. Create a new empty project with the same Unity and render pipeline version as your main project.
2. Copy Project Settings and Quality/Fidelity levels from main project.
3. If you have specific assets(models, shaders) that do not work properly with Impostors, then import them.
4. Verify these assets look correctly in the game without impostors.
5. Import Impostors using Package Manager window.
6. Unity will prompt you to install required packages.



7. Import package content.
8. Open `impostors_sample_City` scene.
9. If your project uses URP then you need to do additional setup.
10. Run/build example scene. Use top-right dropdown to spawn objects.
11. Verify scene works properly with sample objects.
12. Setup troublesome objects(that you imported on step 3) to work with Impostors.
13. Run/build scene and verify it does not work properly with Impostors.
    In case scene is not broken as in main project, look again for project/quality settings or for any additional rendering plugins/packages/features you use in the main project until you able to reproduce the issue.
14. Archive project as **.zip** to send it over.
    You only need to archive these folders: Assets, Packages, ProjectSettings.



    (archive using any application of your choice)
15. Send this **.zip** archive to me :)
    Info about operating system and hardware specs much appreciated.