Description

Fully dynamic Occlusion Culling + LOD system for Unity

- Zero baking time
- Beautifully compatible with procedural scenes
- Revolutionary ray-based LOD system completely integrated
- Combine IOC and LOD as you wish, IOC only, LOD only, IOC + LOD
- Unity Terrain occlusion culling (needs multiple terrain objects or some slicing utility)
- Lights occlusion culling
- Particle effects occlusion culling (Shuriken and legacy)
- Full support of “Occludees” (useful for transparent objects)
- Full support of realtime shadows
- Full support of dynamic batching
- Extremely easy to use, yet remarkably powerful
- Astoundingly big scenes on mobile devices without killing their tiny cpus/gpus

Compatible with all versions of Unity!

How does it work?

It's based on ray-casting: the camera “fires” many rays into the view and when something get's hit, the touched object becomes visible for some time, if after a while no other ray touches the same object, it becomes invisible again (optionally checks its visibility before hiding itself if “Precull Check” is enabled).
All objects that are behind other objects and completely occluded, won't be hit from the fired rays, this way they stay invisible, reducing the number of triangles, vertices and pixel to be drawn on the screen. With the only help of the “View Frustum Culling” all these objects would have been drawn on the screen regardless of their occluded state, with a resultant performance loss.
The integrated LOD of InstantOC uses the rays already fired, to determine the distance from the camera, thus showing the right LOD level for the current distance.
The InstantOC LOD is more efficient then “classic” distance-based LOD systems, being ray-based, it can not only render the low detail version of “distant” objects, but also of near but occluded ones (not visible from the current camera position), thus avoiding waste of rendering time.
InstantOC supports also Unity Terrain, lights and particle systems occlusion culling, further improving the performance boost.
Lights occlusion culling is especially important for Unity free users, because they do not have access to the “deferred renderer” of Unity Pro, which can handle much more lights than the “forward renderer” of Unity free. Thanks to InstantOC lights occlusion culling, (which supports both Unity renderers) you can use much more lights in your scenes also with the forward renderer (Unity free) without killing your frame-rate.
InstantOC Unity terrain occlusion culling can hide occluded terrains; you need to setup your scene with multiple terrains, for instance slicing a big terrain with some terrain utility, or building your level from the beginning with multiple terrain objects.

Usage

InstantOC is extremely easy to use.
After downloading it from the Unity Asset Store, you will find a new folder in your project root called "*InstantOC*". Inside this folder you will find:

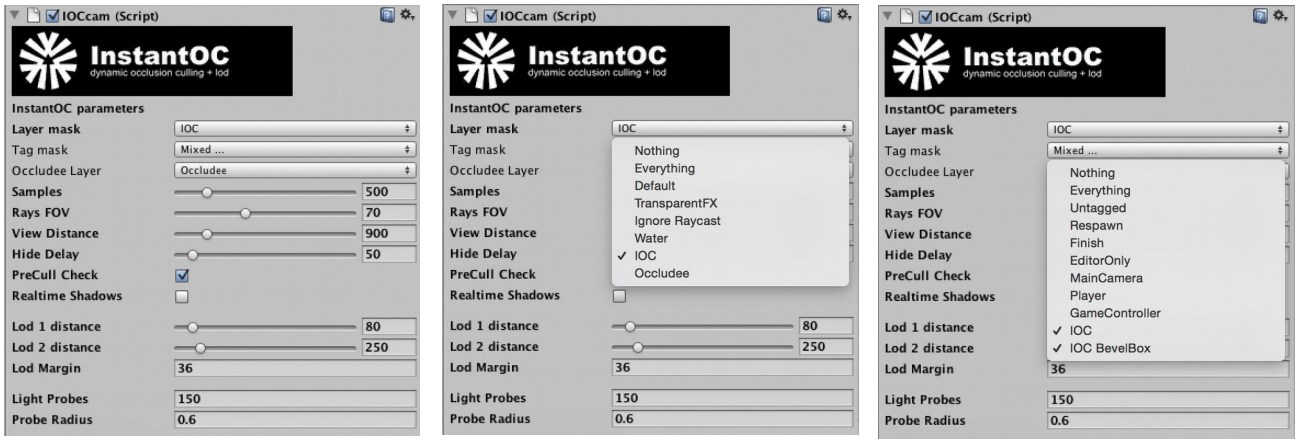
- the 4 core components of IOC ("*IOCcam"*, "*IOClod", "IOClight", "IOCterrain"*)
- the "*IOChud*" Prefab (to help you fine-tune InstantOC options at runtime)
- 2 sample scenes ("*IOC Demo*", "IOC 2 Demo")
- this manual

You can move the main folder of IOC ("*InstantOC*") wherever you want, but please maintain the inner organization as is.

Setting up your project for IOC is essentially a 3 step process:

- add the "*IOCcam"* script to your main camera
- choose the layers containing the Game Objects you want to be managed by InstantOC
- choose the TAGs (or manually add the "*IOClod*" script to your Game Objects / Prefabs)

Let's see them in detail:
Add the "*IOCcam*" script to your main camera and set the options (see below in the Tweaking section for more detail).
Choose the layers in "*Layer mask*"
Choose the TAGs in "*Tag mask*" to automatically enable InstantOC on these Game Objects.



If you don't use the automatic setup via the TAG mask, attach the "*IOClod*" script to your Game Objects / Prefabs (only to the parent Object in case they are grouped) and make sure they have a collider (either the same game object with the "*IOClod"* script, or one of its childs). The rays fired by the main camera actually sense the colliders. If you don't need full collision detection for the model, mark the collider as "Is *Trigger*" (it's sufficient for InstantOC to work correctly).
For best results the collider should resemble rather precisely the shape of the model (use a mesh collider or a good approximation with compound colliders)
The options of these script ("*IOClod*") allow you to

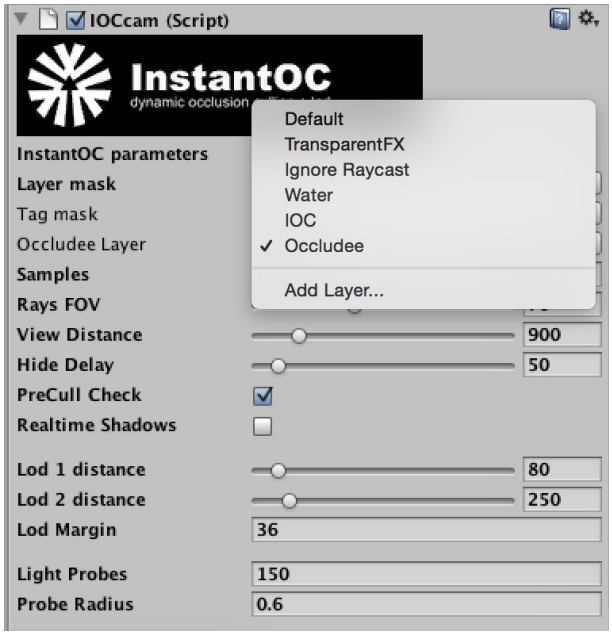
- mark the object as "Occludee"
- override the global LOD settings (which you set in the "*IOCcam*" script)
- force the "LOD only" mode for the current Object/Prefab.

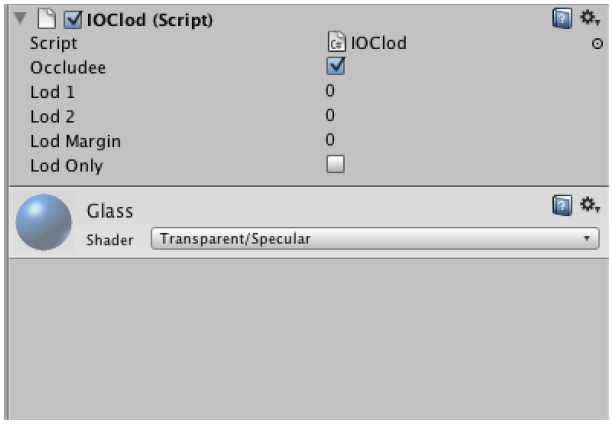
If "Occludee" is enabled, the object will be hidden if occluded by other objects, but will not occlude other objects; it's useful for instance for transparent objects.
In "LOD only" mode, the Object/Prefab will always be visible (with the correct LOD level).

It's sometimes useful for very small and/or distant objects, that are rarely hit by the IOC rays and thus tend to alternate between visible/hidden states in an undesirable fashion (flickering).

To setup Occludees in InstantOC, you need to do the following:

- create a specific layer (InstantOC will use this layer automatically)
- select this layer in the "*IOCcam"* script from the "Occludee Layer" dropdown
- enable "Occludee" on the "*IOClod"* script of the objects you want to be Occludees
- put the Occludee on a layer managed by InstantOC (not the layer you created in the first step!)





Tweaking

The options of the "*IOCcam*" script are the heart and soul of InstantOC. Let's take a closer look at what they are and how they influence the system:

Layer mask

Choose the layer(s) on which you've put the Game Objects/Prefabs you wish to manage with InstantOC.

IOC Tag

InstantOC will automatically add the IOClod component to all Game Objects/Prefabs with this Tag at startup. If you need to override global settings, add it manually and set your options.

Occludee Layer

Occludees have to be on a layer of the Layer mask, just like normal objects that you want to be managed by InstantOC, but when they get hit by the first ray, the system moves them on this "Occludee Layer" and when they later become hidden again, the system moves them back to their original layer. You can create one layer only for this purpose or choose an existing one, but it must be NOT one of the InstantOC Layer mask.

Samples

The number of rays fired by the camera every frame (or to be precise, every Update() call).
More rays, better occlusion culling, but also more overhead. Normally a value between 150 and 500 should be fine for most cases, but the right value for a specific case depends on many factors: target platform (horsepower), size and shape of Game Objects/Prefabs and distance from camera.
Feel free to experiment with different values.

Rays FOV

The Field of View of the "rays frustum".
Will greatly reduce the "popping" of objects along the borders.
You should set it with a value a little higher than the camera field of view. For instance for a camera FOV of 60 you should set the rays FOV to 65 or 70.
Greater the value (in respect to the camera FOV) less "popping" of game objects will happen along the borders when you turn your camera (left, right, up, down).

View Distance

The length of the rays. Objects further away will stay hidden. Use it like you would use the far clip plane of the camera.

Hide Delay

The amount of frames for a visible object to become hidden again. Every time an object gets hit by a ray, the countdown is reset.
A good starting point is 50-100. If distant Objects tend to flicker too much, try raising this value or the Samples value.

PreCull Check

This will greatly reduce the "flickering" effect of distant/small objects also with low Samples values. When checked, objects in the view frustum that are going to hide themselves will do a fast check to see if they are still visible from the camera, if yes, they will delay their hiding.
It's very light and optimized, you can have it always enabled without concerns.

Realtime Shadows
If you have realtime shadows in your scene, enable this option; this way occluded (invisible objects) will correctly cast shadows when needed.

Lod 1 distance
If the Game Object/Prefab has 2 level of LOD (see next section for LOD details), InstantOC will render the LODs as follows:

- Lod_0 if distance from camera >= 0 AND < Lod 1 distance
- Lod_1 if distance from camera >= Lod 1 distance

You can override this value for specific Game Objects/Prefabs through the "*IOClod*" script options.

Lod 2 distance
If the Game Object/Prefab has 3 level of LOD (see next section for LOD details), InstantOC will render the LODs as follows:

- Lod_0 if distance from camera >= 0 AND < Lod 1 distance
- Lod_1 if distance from camera >= Lod 1 distance AND < Lod 2 distance
- Lod_2 if distance from camera >= Lod 2 distance

You can override this value for specific Game Objects/Prefabs through the "*IOClod*" script options.

Lod Margin
This value gives some margin to the previous 2 (Lod 1 distance and Lod 2 distance)
It's usually best to set it to a value near the longest side of the Game Object, and if some flickering occurs (between LOD stages) try to raise it a bit. If there are many different Game Objects/Prefabs in the scene with different sizes, it's usually best to override this value on a per Game Object basis, through the "*IOClod*" script options.

Light Probes
This is the number of probes per light, that InstantOC creates at startup for doing lights occlusion culling. Probes are simple sphere colliders set as "Is Trigger", more probes → more precise results.
For spot lights, 30 is a good starting value, if you have a very wide spot light angle, use some more probes (50 – 70).
For point lights, a little higher value (90 – 100) gives usually better results.
InstantOC generates all probes for all lights with an IOC TAG at start up, so too many probes could lengthen start up time (especially if you have a ton of lights in your scene).
You can override this value for specific lights through the "*IOClight*" script options.

Probe Radius
This is the radius of the probes (sphere collider radius).
Set this to something between 0.2 – 1.0.
The bigger the Probe Radius, the less probes you will need (faster initialization), but too big probes will interfere with mesh occlusion culling.
For best results you will have to find the right balance between probes number and probes radius.
You can override this value for specific lights through the "*IOClight*" script options.

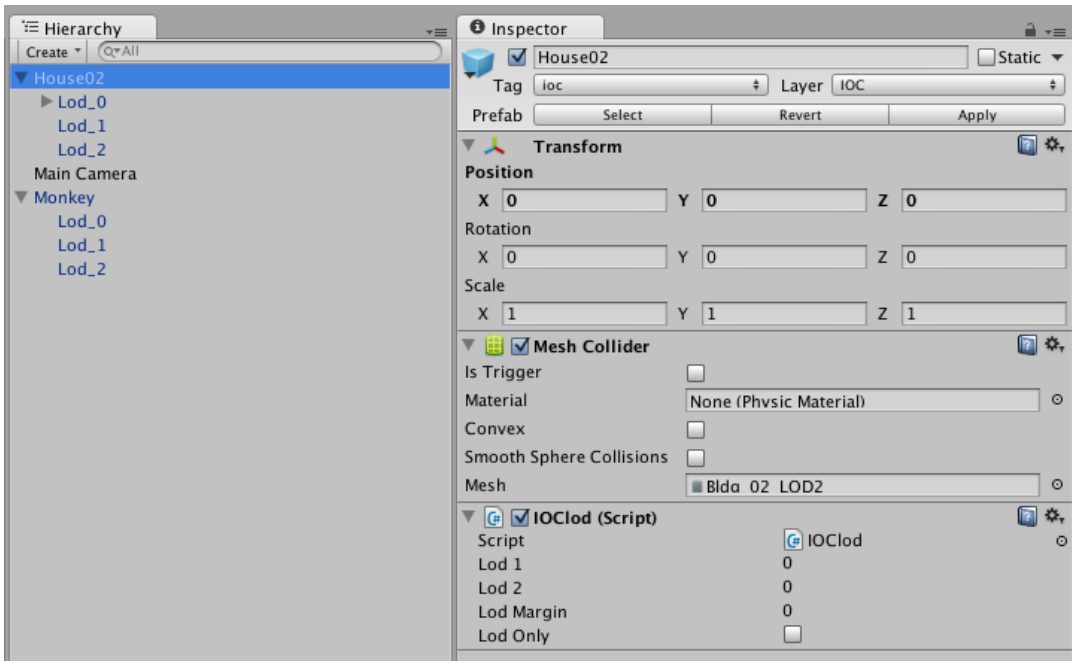
LOD

InstantOC includes an integrated LOD system which supports up to 3 LOD levels.
It is usually more efficient then "classic" distance-based LOD systems, because being ray-based, it can not only render the low detail version of distant objects, but also of near but occluded ones (not visible from the current camera position), thus avoiding waste of rendering time.

To create an InstantOC LOD Game Object/Prefab do the following:

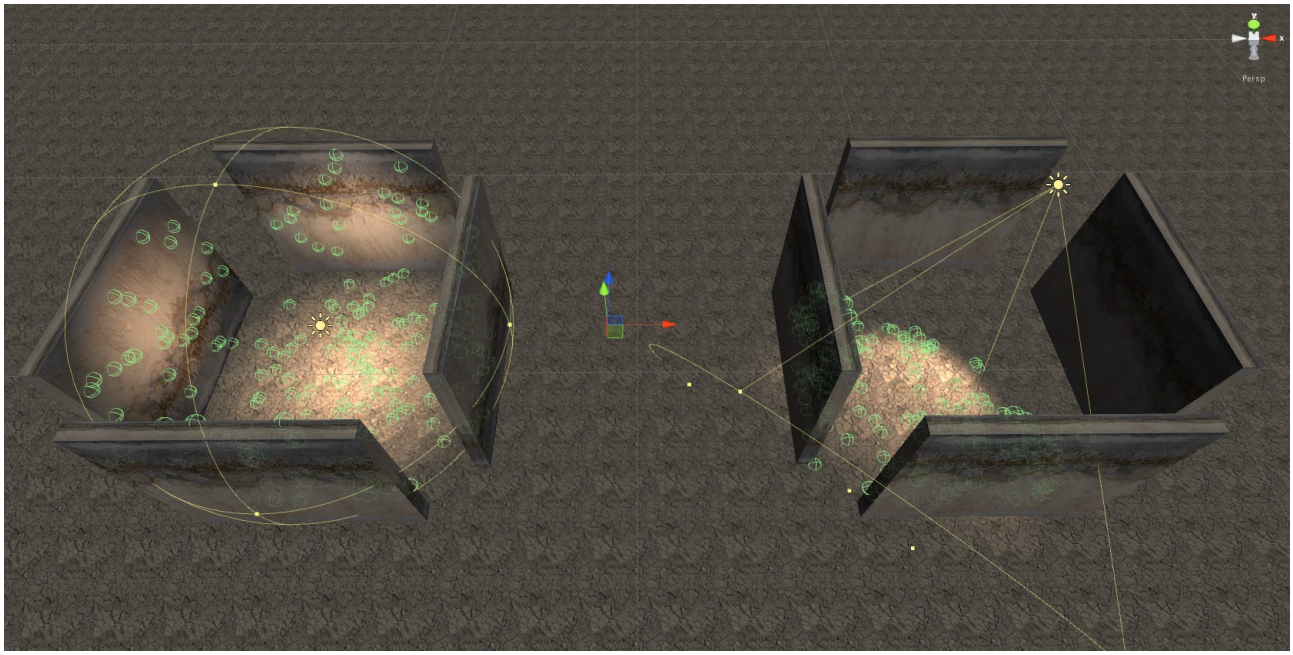
- Create an empty Game Object and name it as you wish
- Drag and drop 2 or 3 Game Objects inside it and name them "*Lod_0*", "*Lod_1*", "*Lod_2"*
- Add the "*IOClod*" script to the parent Game Object
- Be sure there is at least one Collider either attached to the parent Game Object or to one of the Lod children

The game object(s) that contains the collider(s), must be placed on one of the layers managed by InstantOC ("*IOCcam*" script, "*Layer mask*" option).



Lights Occlusion Culling

InstantOC lights occlusion culling is based on light probes.
At start up, it creates some probes for every light that you want to be managed by InstantOC.
Probes are little sphere colliders (trigger only) randomly placed in the influence area of the light.
For spot lights its a cone volume, for point lights it's a sphere.
InstantOC takes several parameters into account like light range, cone angle and obstacles in order to effectively place probes only where the surface is really affected by the light.



When the InstantOC ray-caster "senses" these probes, it activates the corresponding light, thus having always only those lights enabled that really contribute to the current view.
For special cases you can fine-tune your lights also by placing some probes manually exactly where you want them; just drag and drop one or more "probe" prefabs (InstantOC/Resources/ folder) in your scene and make them child objects of the light you are fine-tuning.

Supported lights are Point lights and Spot lights.
To enable Lights Occlusion Culling for a light, do the following:

- Set the IOC TAG on your light
- Put the light on a layer managed by InstantOC
- Set the lights occlusion culling options on the "*IOCcam*" script ("*Probes*", "*Probe radius*")

If you want to override the options for a particular light, just add manually the "*IOClight*" script to the light, and set there the preferred values ("*Probes*", "*Probe radius*").

Terrain Occlusion Culling

InstantOC fully supports terrain occlusion culling as of version 2.1.0.
You will need multiple terrains, either by slicing one big terrain with some utility (like the "Terrain Slicing & Dynamic Loading Kit"), or by starting from scratch with multiple terrains.
To enable Unity terrain occlusion culling, do the following:

- Set the IOC TAG on your terrain(s) (or add manually the "*IOCterrain*" script)
- Put the terrain(s) on a layer managed by InstantOC

IOChud Prefab

The "*IOChud*" Prefab is a helper object that lets you fine-tune InstantOC options at runtime.
Drag it in the Hierarchy Panel and hit "Play".
Write down the final values, you will have to re-enter them in the "*IOCcam*" custom inspector.

Support and Contact

If you need help or have any suggestions, please send me a mail at
frenchfaso@live.com
I will be glad to answer.

Frenchfaso Indie Apps – Smart Apps for Smart Devices
http://frenchfaso.blogspot.it/p/unity3d.html