InstantOC
dynamic occlusion culling + lod

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
- Full support of realtime shadows (Unity Pro)
- 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! (Free/Pro/Mobile/Flash/etc..)

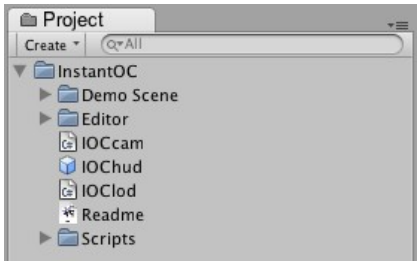
How does it work?

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.
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.
When used with the "Lod Only" option, the InstantOC LOD is far 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.

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:

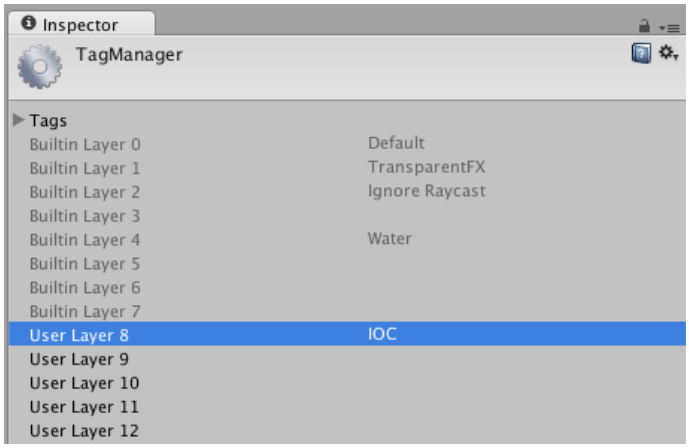
- the 2 core components of IOC ("*IOCcam"*, "*IOClod"*)
- the "*IOChud*" Prefab (to help you fine-tune InstantOC options at runtime)
- the sample scene ("*IOC Demo*")
- this manual



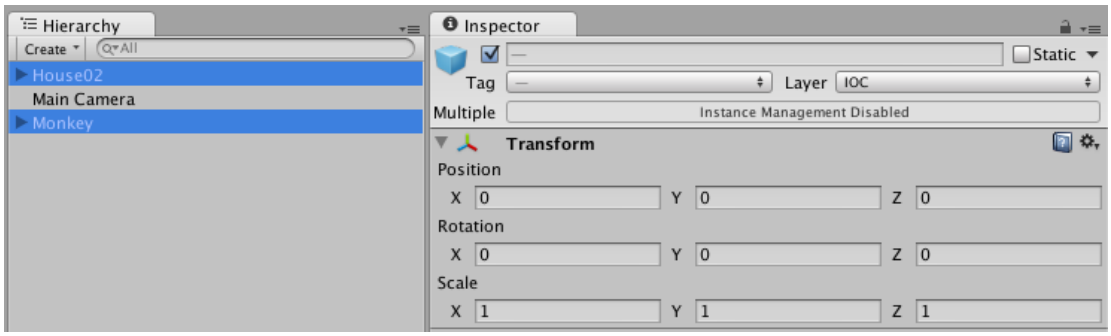
You can move the main folder of IOC ("*InstantOC*") wherever you want, but please maintain the inner organization as is.
Setting up your project for IOC is essentially a 3 step process:

- create a new layer and put all your Game Objects/Prefabs you wish to use with IOC on it
- add the "*IOClod"* script to these Game Objects/Prefabs
- add the "*IOCcam"* script to your main camera and set your options

Let's see them in detail:
Create a new layer, in these sample screenshot it's named IOC, but you can name it as you wish.



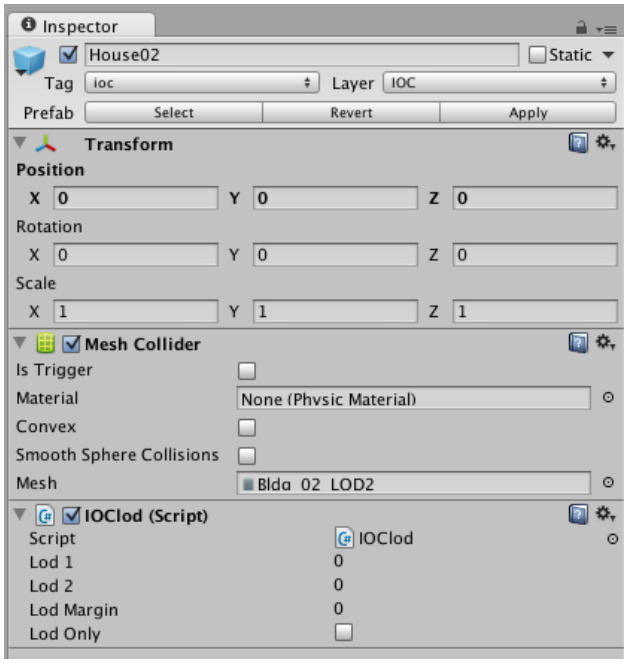
Put the Game Objects/Prefabs that you wish to use with IOC on these layer.



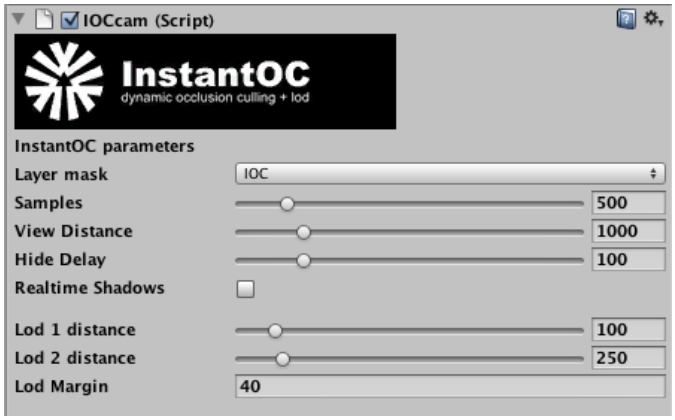
Add the "*IOClod*" script to your Game Objects/Prefabs making sure they have a collider. The rays fired by the main camera actually sense the colliders. If you don't need full collision detection for the model, mark the collider as "Is *Trigger*" (it's sufficient for InstantOC to work correctly).
For 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 allow you to

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

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).



Add the "*IOCcam*" script to your main camera and set the options (see below in the Tweaking section for more detail).



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.

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 different values.

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 100-150. If distant Objects tend to flicker too much, try raising this value or the Samples value.

Realtime Shadows
If you have a Unity Pro license and have realtime shadows in your scene, enable this option, this way occluded (invisible objects) will correctly cast shadows.

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.

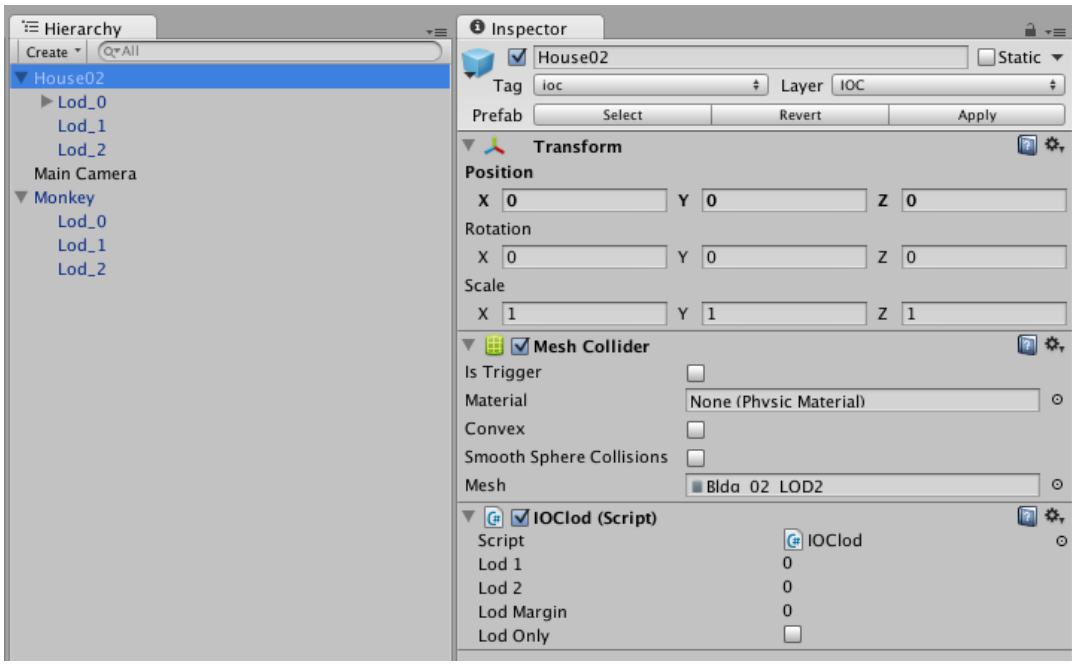
Lod Margin
This value gives some margin to the previous 2 (Lod 1 distance and Lod 2 distance)
It's usually best to set it to a value near the longest side of the Game Object, and if some flickering occurs (between LOD stages) try to raise it a bit. If there are many different Game Objects/Prefabs

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.

LOD

InstantOC includes an integrated LOD system which supports up to 3 LOD levels.
When used with the "Lod Only" option, the InstantOC LOD is far 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.
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 one Collider either attached to the parent Game Object or to one of the Lod children



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