Manual

QuickLod is an easy to use and superfast asynchronous lod system that will make your life easier. You don't need to worry any longer about the triangle count, frame stutters in the games or slow editors.

In this manual, you will learn how to setup QuickLod nice and easily.

Have fun!

More information's:

Website:	http://tirelessart.ch/en/quickcode/quicklod/
Forum:	http://forum.unity3d.com/threads/237196-QuickLod
Demos:	http://tirelessart.ch/en/quickcode/quicklod/demo/
API documentation:	http://chillersanim.bplaced.net/

Contents

Introduction

In modern games, high quality, realism and worlds full of objects are common. But while the quality gets better, the performance usage of the games grows exponentially.
There are numerous ways to reduce this performance hunger of games; most of them don't even reduce the graphic quality. One of these methods is called level of detail, in short "lod".

Imagine a large open world game. You are on top of a mountain and you can see a city far away. This city contains multiple thousand objects and millions of vertices. It's a beautiful city, but you are far away from this city. In fact, you are so far away, that the city covers just one single pixel on your screen. It is thus a total waste to render the whole city!

That's where LOD systems are used. They reduce the graphic quality of objects when they get further away and vice versa. A tree directly in front of your nose has the maximum amount of details. But when the same tree is 100 meters (~328ft) away, then some of its details are removed and at 500 meters (~1640ft) only a billboard is left.
That was just an example; you can use this principle on every object in your game!

In theory, you could just calculate the distance between the camera and the object and modify the quality based on that distance. In real life, this doesn't work, as most games contain multiple thousands of objects, some even millions of objects!
Calculating the distance for so many objects can't be done in real time at 60fps, so you need some clever algorithms.

QuickLod offers you a solution, which is blazing fast and independent of the amount of managed objects. Thanks to many optimizations, you can run a game with tons of objects smoothly and don't have to care about performance anymore.

QuickLod is a level of detail implementation that is usable in 3D games. It automates most of the LOD setup, and allows you to focus on creating the game. With many useful features, you will be able to use it in all sorts of games, and you can always get the most out of it.

In the following chapters, you will learn how to use the different components and how to setup QuickLod.
If you still have questions, feel free to ask them on the forum.

Getting started

QuickLod needs three components in order to work:

- **LodManager**
  The central manager that manages all lod objects
  You need a lod manager in the scene for QuickLod to work
- **LodObject**
  Is needed by each game object that should have distance based quality levels
- **LodSource**
  Is needed by each camera that should be taken into account by the lod system

When you add a component, you can expand the “Help” category to get short information how the component works and how you should setup the component.

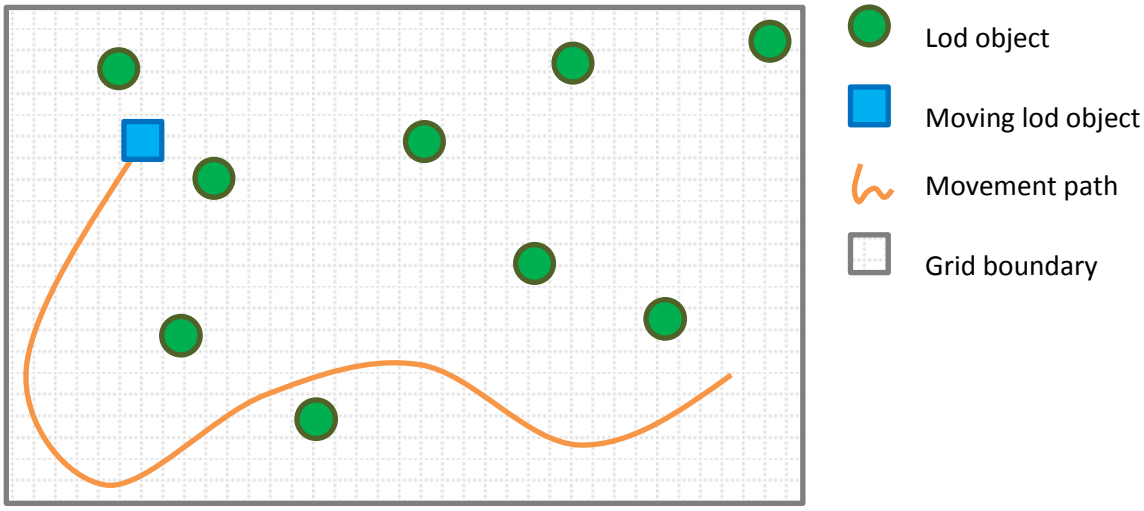
Each setting has a tooltip that describes the functionality of this setting. To see the tooltip, just hoover the cursor a moment over the name of the option.
Some tooltips have extended information, which describe the currently selected value. To see the information for another value, you need to switch the value.

God to know: It is not important in what order you setup the components.

Lod manager

To setup QuickLod, add the LodManager component to a game object. This game object should not be deactivated or deleted during the game.
Then you need to setup the boundary for the LodManager which is defined by a starting point and a size. When the editor setting “draw grid” is activated you can directly check if the boundary is correct or not.

The size of the boundary depends on where lod objects can be. It should just cover all the space where ever a lod object might be, but should not be larger than that.



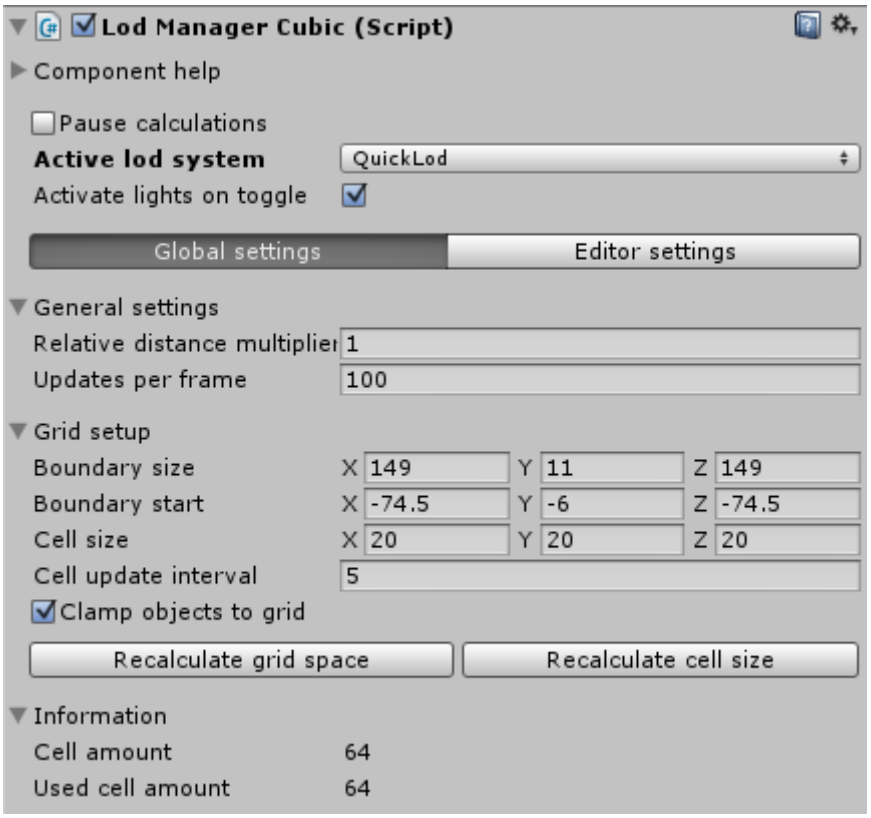
Picture 1 - Grid space example

In this example, you can see that the grid covers all the space, where ever a lod object can be. When a lod object moves, then the space in which it moves must be covered to. In QuickLod the grid is 3-dimensional, but the principle is the same.

You also have to setup the chunk-size; this value determines the resolution of the grid. The better the resolution is, the better can QuickLod optimize the managing process. But a better resolution means also more pre-processing and more RAM usage. You should find a setting that is optimal for your scene, which depends on the density of your game objects and the size of the grid. A chunk should contain in average between 10 and 100 objects. That value also depends on the target platform.

God to know:
Only used chunks will be stored in the memory. So if you have defined a grid size that contains 64000 cells, but only 3 cells are used, then the memory only stores 3 cells.

QuickLod can calculate a possible grid size and chunk size for you. This value is only estimation, as QuickLod doesn’t know how your scene will change in the game. It is probably a good idea to play with the grid values until you find an optimal setting for your game.



Picture 2 - The lod manager

Lod object

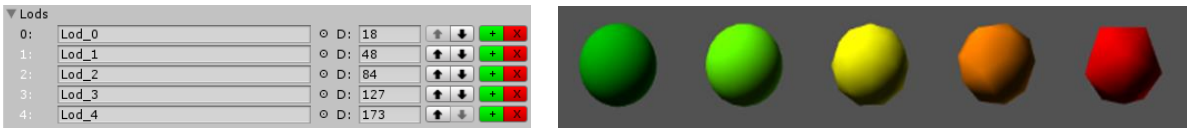
You have to setup all game object which needs to have different quality levels depending on the distance to the nearest camera.
There are two LodObjects, each of them has advantages and disadvantages. Because of that, it's important that you understand which LodObject is the right choice for which situation.

The simplest LodObject is the **LodObjectMesh**. This one replaces the mesh in the mesh filter component. That is fast and uses no additional ram, but it cannot manage components.
A similar component is the **LodObjectSkinnedMesh**, which is used for the SkinnedMeshRenderer. It allows you to apply lod levels to your animated characters.

The more advanced LodObject is the **LodObjectReplacement**. It activates and deactivates referenced game objects. That allows you to use different game objects for each lod level. You could use a high quality particle emitter for nearby objects and a low quality particle emitter for far away objects. Because you need to have one game object for each lod level and a parent object, it needs more ram and can slow Unity down, as Unity is not optimized for large amounts of objects.

If you just want to activate or deactivate a bunch of objects depending on the distance, you can use the **LodObjectChildSwitch** which manages its children. This is a fast and easy way to deactivate a whole complex hierarchy with just one component.

Recommendation:
You should use the LodObjectMesh whenever possible. Use the LodObjectReplacement only, if you really need to use different components for different distances.



Picture 3 – Different lod levels inside the lod object component

As soon as you add a LodObject to a game object, it will try to automatically setup the lod levels depending on the given circumstances. You should verify that the lod levels are setup correct. The lod levels needs to be sorted depending on the quality, so that the lod level with the highest quality is at the top.

Each lod level has a distance. This distance is the maximum distance at which the lod level can be shown. When the distance to the nearest camera surpasses the largest defined distance, then all lod levels will be hidden.

When the game object doesn't move, then you should activate the "Is static" option. This prevents the object from checking if it has moved and saves some performance. Only the movement is important, if the game object scales or rotates, then you can still activate this option.

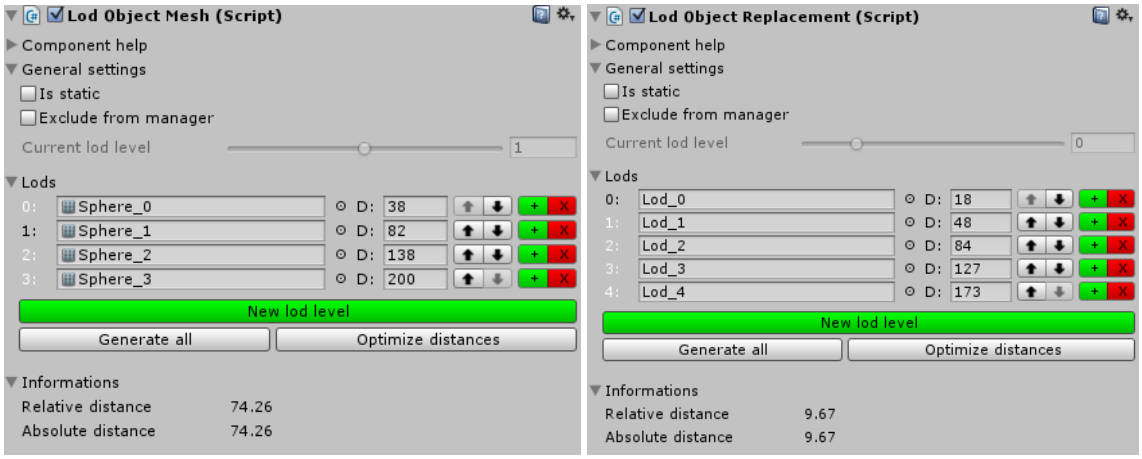
God to know:
When you move the object by code, you can still activate the option "Is static". But you need to call the "UpdatePositionChange()" method of the lod object.

You cannot set the current lod level manually when the lod object is being managed by a lod manager. If it is necessary to set it manually, you can activate the option “Exclude from manager”. When you activate this option, the lod object will be effectively ignored by the lod manager.

Recommendation:
When you use the LodObjectReplacement, you can reference any game object you want. It is still recommended to use a parent-child hierarchy where all lod level objects are childs of one parent object. This parent object should contain the LodObjectReplacement.

The lod object shows you two distances which can be different from each other. The relative distance is the distance to the nearest lod source, where the lod distance multiplier is taken into account. On the other hand, the absolute distance shows you the real distance to the nearest camera without using the lod distance multiplier.

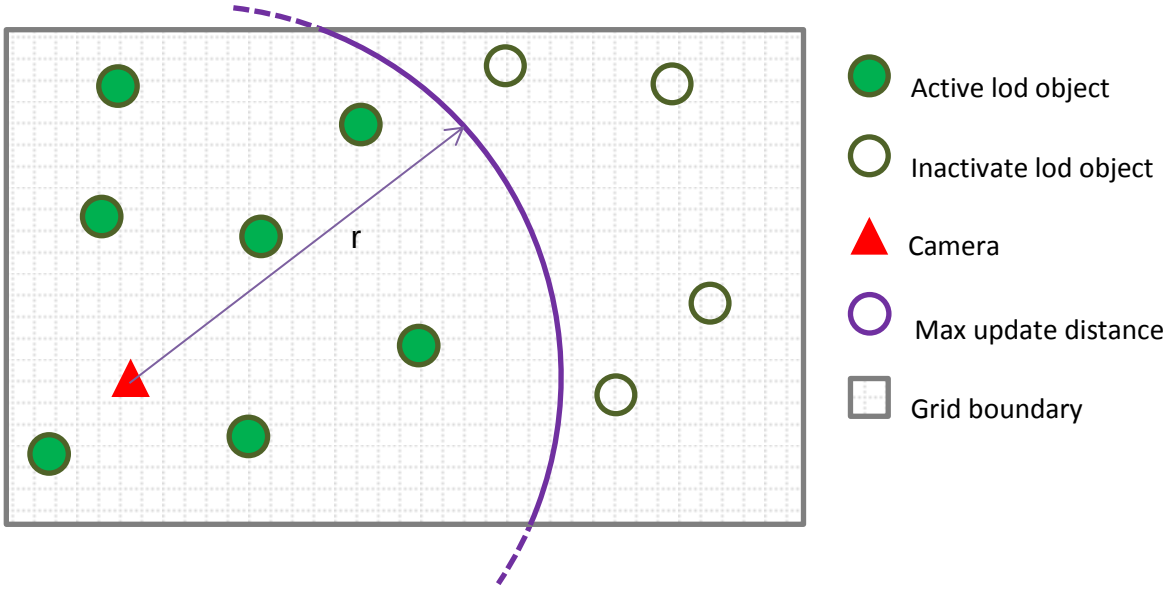
To calculate the current lod level, only the relative distance is used. But it can be useful to see the real distance. Also it can be, that the nearest camera for the relative distance is not the same camera as the nearest camera for the absolute distance, as you can disable the lod distance multiplier for individual lod sources.



Picture 4 – Some lod objects

Lod source

Now you have to setup all used cameras. Each camera that should be taken into account by the lod system needs a LodSource component. The most important setting of this component is the “Max update distance”, this settings defines at what distance the lod system should stop updating the lod objects. The value should be a bit larger than the largest used lod level distance.



Using the max update distance allows QuickLod to only recalculate objects when necessary. As no object outside this distance will be visible, it’s not necessary to recalculate the distance to them.

Because QuickLod updates the lod objects asynchronous, it’s important that nearby objects are updated more often than far away objects. That is due to the fact that changes made to nearby objects are better visible than changes made to far away objects.
QuickLod priorities nearby objects, with the help of the maximum update distance.

You can disable the lod distance multiplier for the lod source. That can be especially useful when you are indoors and cannot see further than a certain distance.

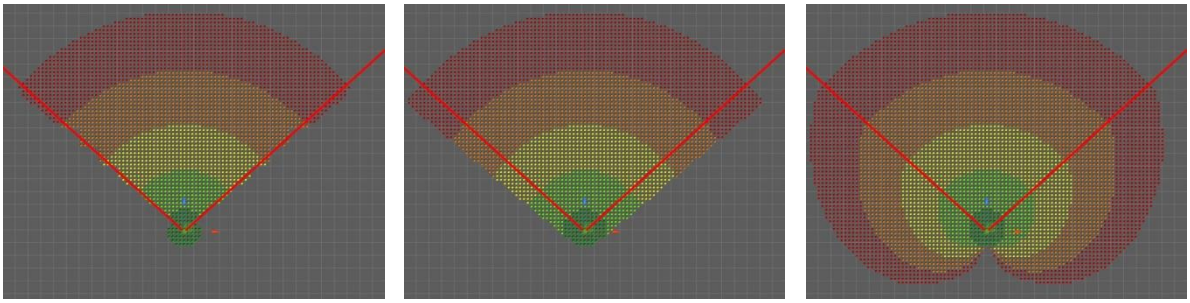
To focus the update calls even more on visible objects, you can use the option “Use view angle”. With this option, QuickLod will only recalculate objects in a certain view angle of the object.
That can be especially useful when you want to zoom in a certain area and have to increase the lod distance multiplier.

While the option is active, camera rotation can cause objects to pop in the screen, as they weren’t visible before and are recalculated asynchronous.
For that, QuickLod offers you three modes, called “Falloff mode”. These three modes give you more control about how QuickLod should manage objects outside the view angle. With that, you can prevent most too all object pop in.

If you want the camera to automatically update some values depending on the Field of View (FoV), then you can activate the “Observe Camera” option. This is useful if you want to zoom or the aspect ratio changes during the game.

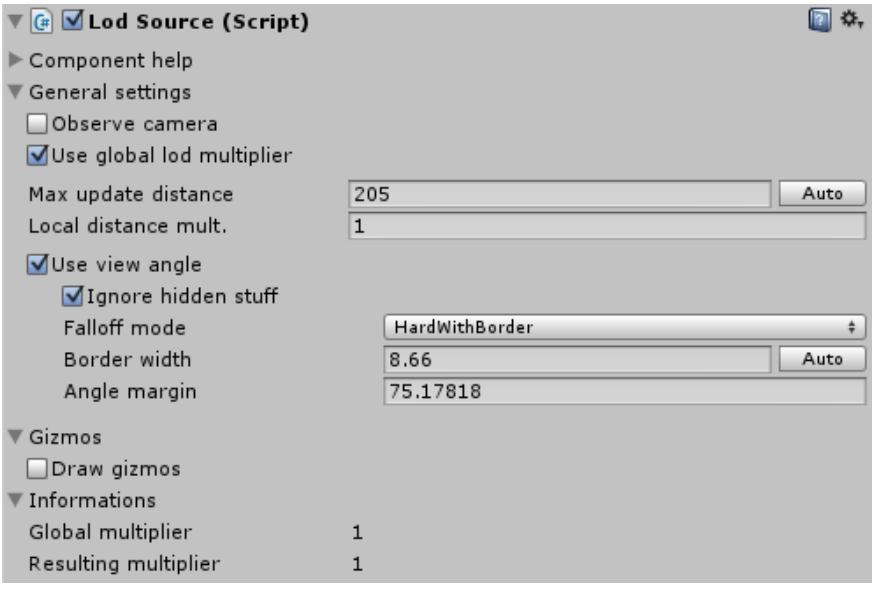
The three falloff modes are:

- **Hard**
  Simple view angle based calculation with a hard border. It's fast, but not precise and also not good with rotating cameras.
- **Hard with borders**
  It's like the hard mode, but adds a border that prevents objects to pop in. It offers the best optimization but the calculations aren't that fast and objects can still pop in.
- **Smooth**
  With an smoothed falloff, you will nearly never see objects pop into the screen. As a down side, it hides much less objects and has thus a lower optimization.



Picture 5 - Falloff modes (from l. to r.): Hard, Hard with borders, Smooth

The camera distance can be visualized by activating the "Draw gizmos" option. That will draw 4 spheres around the lod source which indicates the different priority levels.



Picture 6 – The lod source

Group Switch

QuickLod offers you an extended way to cull unused objects with the GroupSwitch.
The GroupSwitch activates and deactivates objects depending on trigger inputs, these triggers can be everything, from distance based to collision based or even something more fancy.

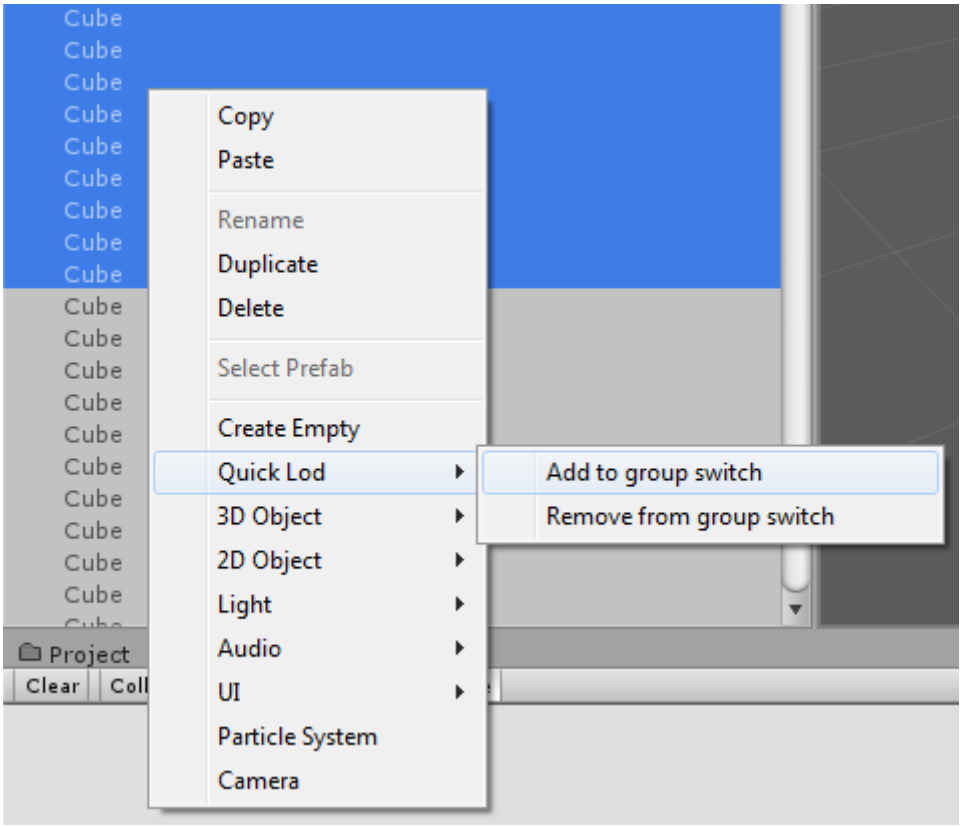
This allows you to precisely control when which objects are activated or deactivated. That can be used for example in caves, houses, cities, and much more.

To add a group switch, you need to add the **GsGroupSwitch** component to any game object.
Then you need to define which objects it should manage.

There are four ways to add game objects to the managed:

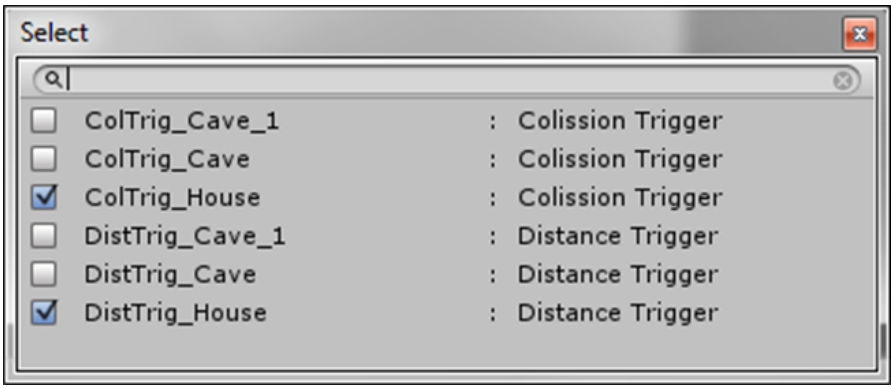
- Add them to the list by selecting them in the object selector of Unity
- Drag and drop them on the list
- Right click them in the hierarchy and use the option “Quick Lod/Add to group switch”
- Add them with a script, accessing the “ManagedObjects” property

It’s recommended to use the right click option from the hierarchy, as this will also check if the objects are already managed by any other group switch. This prevents mistakes and strange behaviours when multiple group switches manage the same objects.



Now you need to add some triggers to the scene.
For more information about triggers, see the chapter Triggers.

When you created your triggers, you need to add them to the group switch. For that you need to select the group switch and in the category “General settings/Triggers” select the button “Select triggers”.
This will open a new window where you can select the triggers from the scene, just tick the ones you want to use.



You can define how long the group switch should wait with deactivating the managed objects after it is no longer triggered. For that use the property “Deactivation delay”.

When you want to enforce a certain trigger state for the group switch, you can activate the property “Force trigger state”. Then you can use the property “Forced state” to control the trigger state.
This overrides the trigger input from all triggers.
Note that the deactivation delay still works.

Triggers

Triggers are components which implement the IGsTrigger interface.
They are used to check if defined conditions are met.

You can think of triggers as switches, which are turned on when a condition is met and turned off when not. What this condition is depends on the trigger.
For example a distance trigger would check if an object is inside a given range. If yes, then it would be triggered, else not.

A trigger consists of two properties:

- **IsTriggered : bool**
  Tells you if this trigger is currently triggered or not
- **Name : string**
  The name of the trigger type. Can be used to explain the function of this trigger.

In QuickLod, triggers are used for the group switch.

Editor remarks

Unity has troubles working with interfaces, as they aren't real objects.

You can't serialize an interface reference, so you need to store a reference to the component implementing the interface.

The object selection window in the Unity editor doesn't support selecting interfaces.
Because of this, QuickLod offers its own window, which can be used to select triggers in the scene.
You can use the GsTriggerSelector class to show a selection window.
You need to call the method "Create" and pass a list of triggers by reference.
The window will update the list on the fly and close when it loses the focus.

Editor

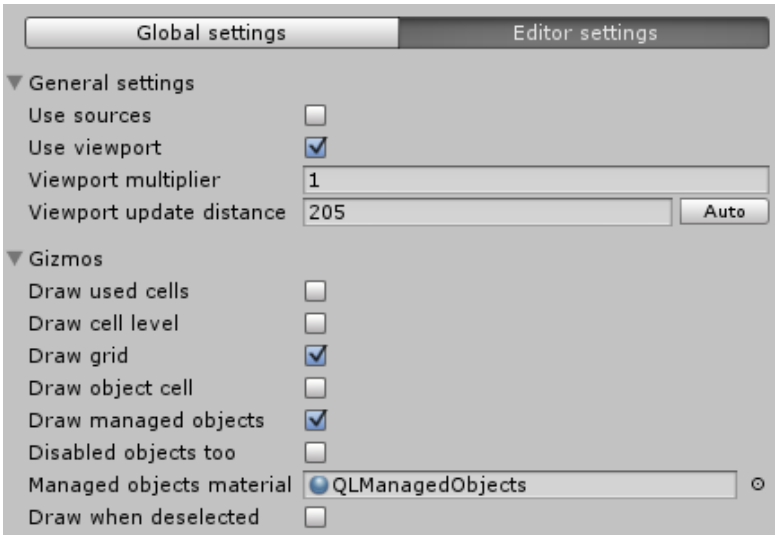
QuickLod offers some settings for the editor behaviour. Those settings aren't relevant in the build game, but are very useful for developing with QuickLod. You can find those settings in the lod manager under the editor settings.

In the editor, QuickLod automatically adds a new game object that is used as a camera component in order to use QuickLod with the current scene view. This game object is managed separately from the other camera components and has additional settings. You need to define the maximum update distance here as well. The value can be defined in the “Viewport update distance” field.

You can activate or deactivate this viewport camera using the setting "Use viewport". Also all other camera components can be activated or deactivated using the setting "Use sources". The "Viewport multiplier" is a distance multiplier that is only applied to the viewport camera. It is useful when you want to see further inside the editor view without affecting all lod sources.

QuickLod also offers some visualization, to help you find wrong settings and optimize them. The following visualizations and settings are supported:

- **Draw used chunks**
  Draws all chunks which have at least one lod object inside
- **Draw chunk level**
  Draws the priority of each chunk. Chunks that are outside the maximum update distance of any camera don't have any priority and are ignored when calculating the new lod levels.
- **Draw grid**
  Draws the whole grid with all chunks
- **Draw object chunk**
  Draws each chunk that contains at least one selected lod object.
- **Draw managed objects**
  Highlights all managed objects with a given material. Can be used to search for unmanaged objects.
    - **Disabled objects too**
      When activated, disabled objects will be drawn too. Be aware that LodObjectMesh and LodObjectSkinnedMesh remove the mesh when disabled, so those lod objects cannot be drawn.
    - **Managed objects material**
      Define which material should be used to highlight the objects. That allows you to customize the look.
- **Draw when deselected**
  When activated, then the visualizations are also drawn when the lod manager is not selected.



Picture 7 – The editor settings

Compatibility

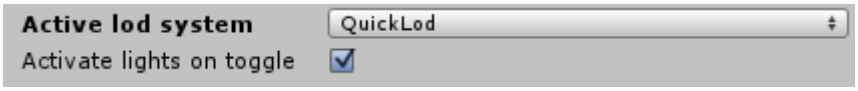
QuickLod offers compatibility with the tool “InstantOC” from *Frenchfaso Indie Apps*.

Both tools have their advantages and are strong in their own way.
QuickLod is great in managing many objects and offers many settings and tools for distance based lod.
InstantOC is great in occlusion culling for objects, and also offers a simple lod system.

You can setup both systems independently and use one system or the other. You cannot use both systems at the same time, but switching the systems is easy and allows you to use the different systems in certain areas.

Note: In the editor, QuickLod is always active. The lod system selection is only taken into account while the game is running.

To define which lod system is currently active, you can access the option “Active lod system” in the lod manager. InstantOC can disable light sources, if you want them to be reactivated when switching to QuickLod; you can activate the “Activate lights on toggle” option.



Picture 8 – Set the active lod system

Brought to you by: