Level Design - Indoor Level Performance

Introduction

This tutorial will show some steps on how to optimize an indoor level to get better performance.

Target

Measuring performance can be hard. The main target is 60 FPS on a high end computer running at 1080p and 30 FPS on a low end computer running at 720p with low quality settings.

There are other things that are much easier to measure. Here is a list of goals for different parameters (in 1080p):

- Total Memory Usage: ~700 mb (max 1024 mb)
- Vertex Memory Usage: ~64 mb (max 128mb)
- Draw Calls: ~400 (max 500)
- Rendered Triangles: 1.0 million (max 1.5million)
- Queries: ~50 (max 100)

Occlusion Culling

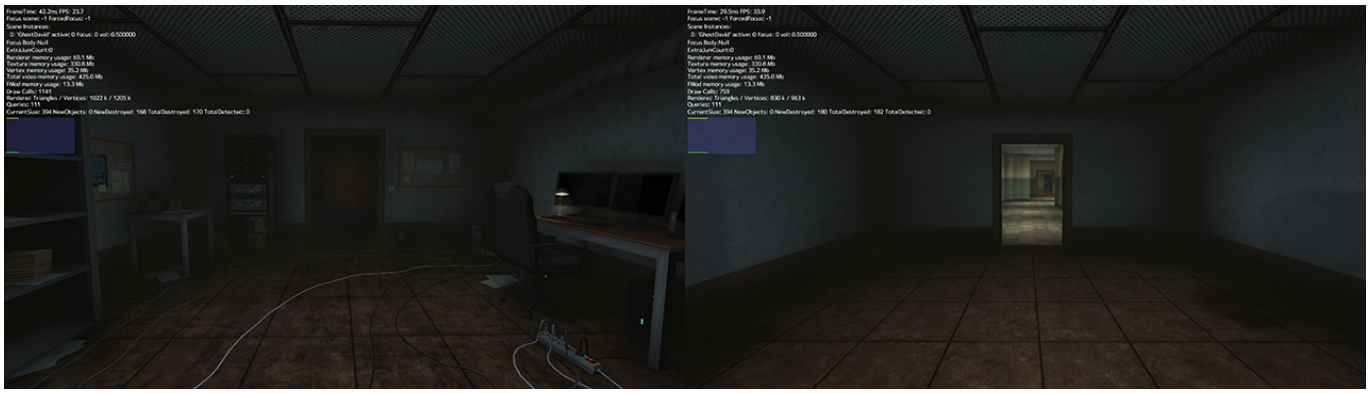
The only way to get good performance on an indoor level is to design it to make use of occlusion culling. Occlusion culling reduces the number of objects that needs to be rendered.

Occlusion culling is a way for the graphics engine to calculate which objects are visible from a certain position in the level. This is a complicated problem and the engine will not be able to solve this perfectly on its own. There are a few things that can be done to increase the efficiency of the culling.

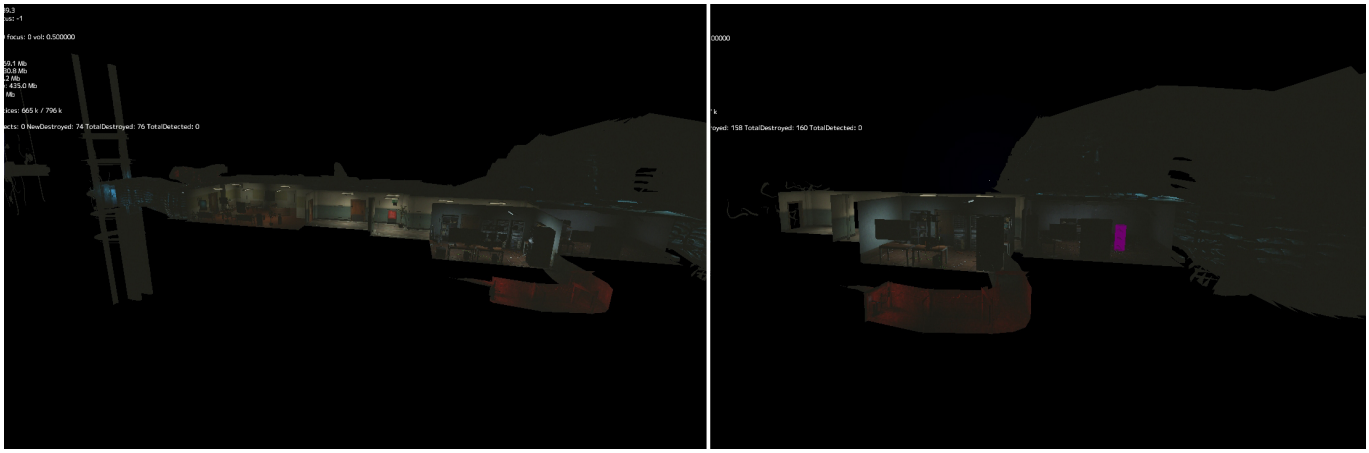
Occluders

An occluder is an object that is used to block the vision of the engine. All static objects are automatically set to be occluders, but dynamic objects are not. Some type of dynamic entities should be set as occluder to help the game. The most important ones are doors or objects that act as blockades. You should not set to many dynamic objects as occluders since it increases the draw call and queries. But it is better with to many than to few.

In order to see where the occlusion culling fails you can use the Debug Menu (F1) in Depth and select Render Only Occluders. This will show you what the engine sees when determining what is visible.

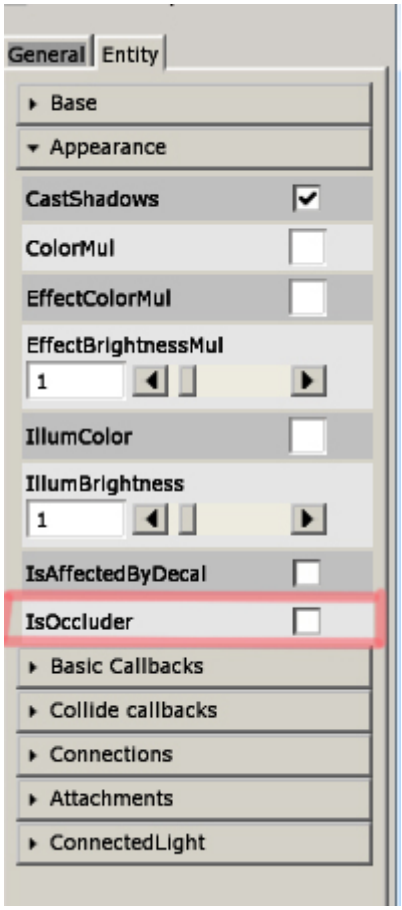


The image to the left shows what the player sees. It looks like only a single room is rendered. On the right you can see what the engine sees when calculating what is visible.



On the left you can see everything that is rendered when the door is not set to be an occluder, which is pretty much the whole level. On the right the door is set to be an occluder and most of the objects are removed. But it is still not perfect.

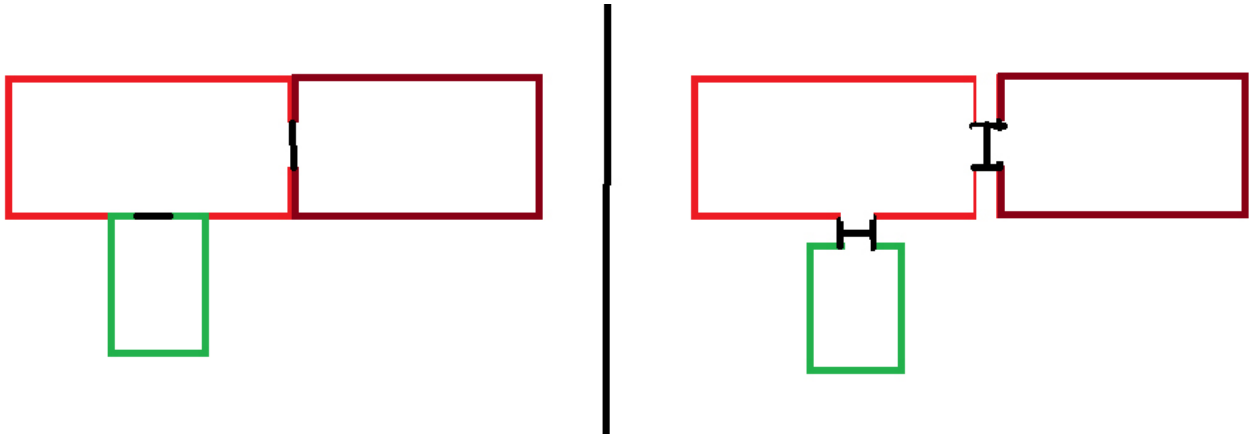
IsOccluder setting can be found under the Apperance tab for dynamic entities.



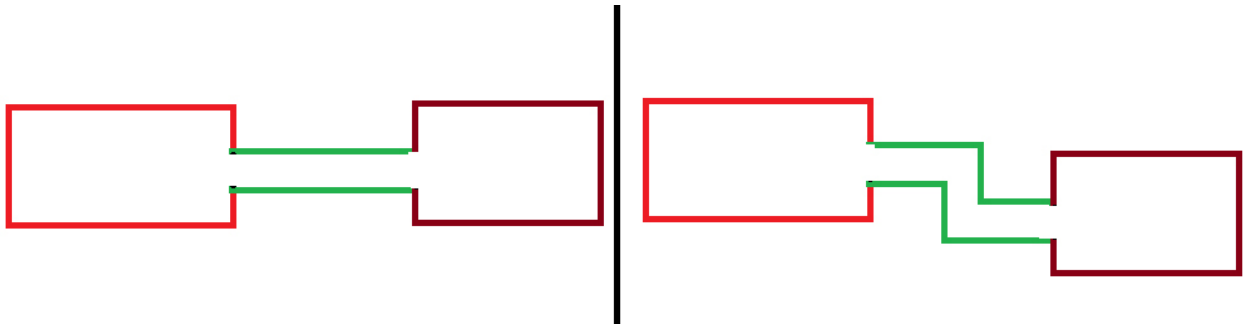
Room Positioning

Another way of making sure that the occlusion culling is effective is to try to separate rooms and corridors as much as possible.

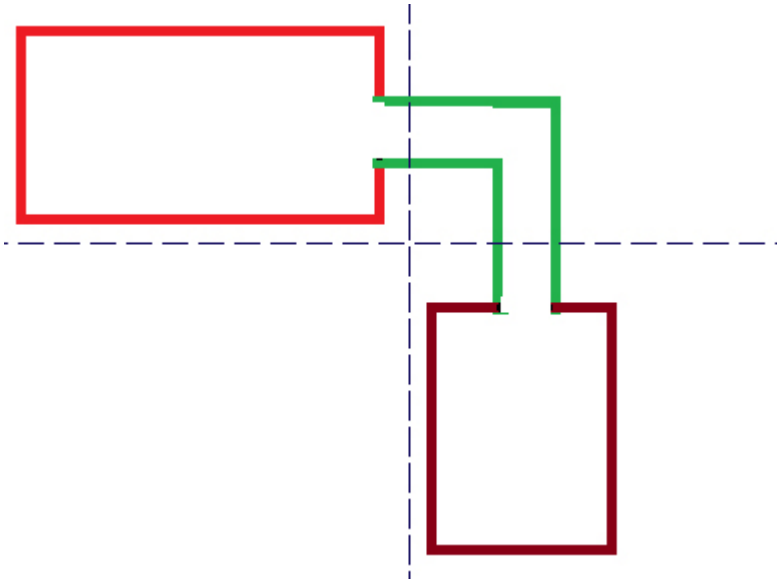
The following images only show 2D setups, but they work on all axis and should be applied to 3D!



Adding some space between rooms can help the occlusion culling determining which objects belong to which part of the level. The space does not need to be as big as in the image. The important part here is that the bounding volumes of the objects in one room should not intersect with the other room. This is very important for shadow casting lights.



In this image the two rooms are connected with a corridor without doors. If the player stands in one room he would be able to see the other room. The best solution here would be to add doors or other occluders. If that is not possible you can change the look of the corridor so that the player does not have direct vision between the rooms. Using a diagonal corridor would work as good as a zigzag one.



If the performance is really bad and the culling fails then you should try to fully separate rooms in at least two axis. The dotted blue line show that the two rooms are not intersecting in the x- and y-axis. This kind of separation really helps the space partitioning algorithm.

Leaks

It is important to fix any small gaps and leaks in the geometry between two rooms.

Here you can see some geometry leaks from one of the doors. These leaks can cause the occlusion culling to fail and should be fixed.

Shadows

Spot light shadows are the second most expensive part of a indoor level. There are a few ways to reduce the cost of shadow casting.

If your light is not used for anything gameplay related then you should disable “Shadow affects dynamic”. This greatly increases the speed of the shadow casting because the shadows only have to be generated one time. If the shadow affect dynamic entities it has to be generated again each frame.

Optimize the radius and FOV of the light. The larger the radius the more objects has to be rendered to the shadow map. You should also try to keep the bounding volume of a light from leaking into multiple rooms.

If you have multiple shadow casting lights in the same room you can set the shadow fade range (not in yet) to a low value. This will make the shadows fade in/out when the camera gets further away than the value.

Reducing the number of shadow casting lights and replacing them with one bigger light is also a great way to increase performance

Use a gobo texture instead of shadow casting if possible.

If a room requires the use of the flashlight then it needs to be turned on when calculating performance.

You can increase the speed of shadow rendering by removing shadow casting on some object. You only have to do this on objects that are near a shadow casting light. Here is a list of object that should

never cast shadows:

- Floors that have nothing beneath them
- Outer walls that have nothing on the other side
- Small objects near walls and floors
- Objects that are guaranteed to be in the shadow of all lights