Visibility Culling

By: Maurits Dijkman - 495475 - ECM2V.Ea

Enschede - March 3, 2022

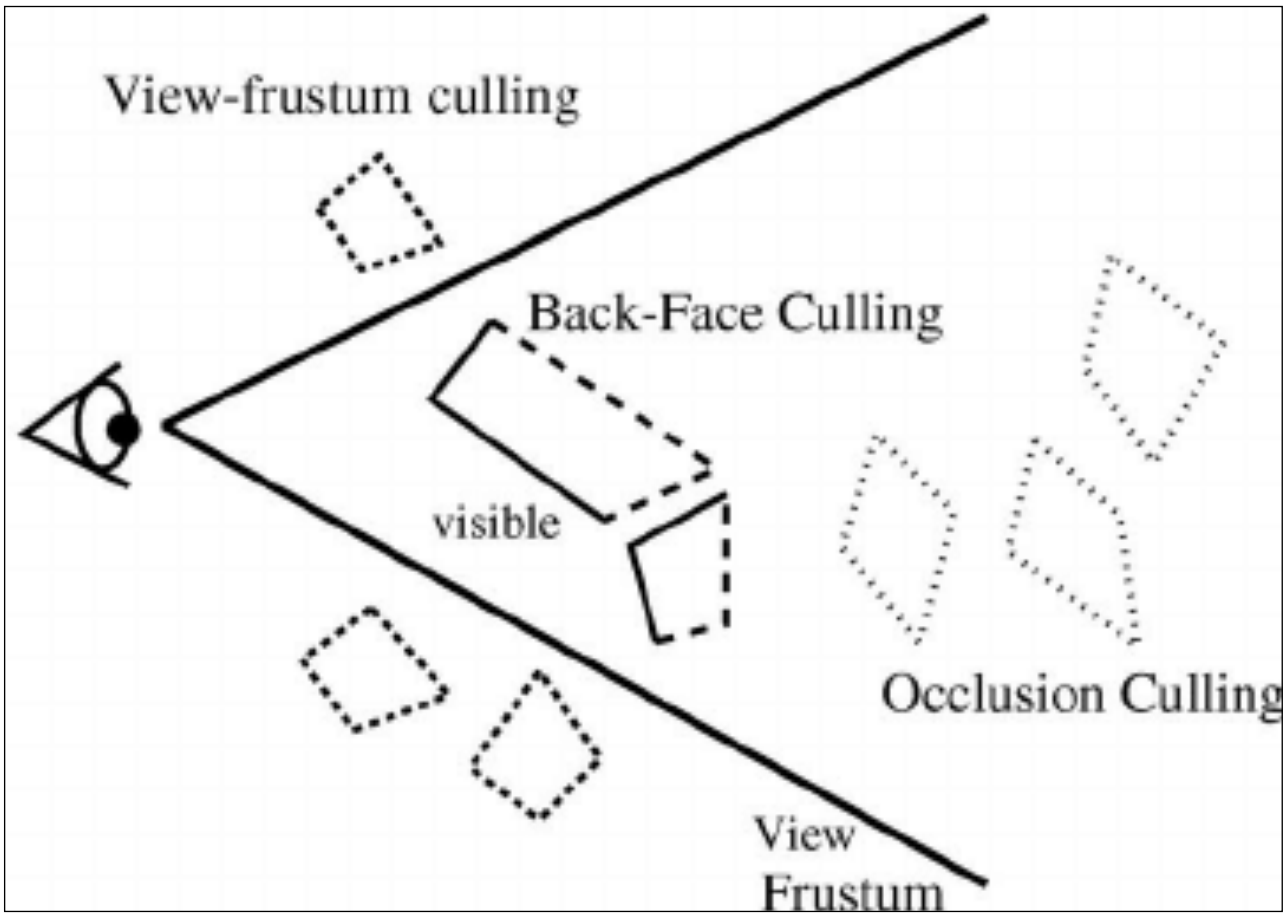


Table of Contents

Culling methods

Why culling methods?

3D Objects in games consist out of little triangles. These triangles together give the shape of the object. The computer needs to calculate and show these rectangles. The more rectangles the computer needs to calculate, the slower the computer will perform.
To make a game run smooth, some clever programming came up with culling methods. These methods remove triangles that are not in view from the calculations the computer is doing. With the triangle count downsized, a computer will run smoother.

There are three different methods of culling;
- View Frustum Culling (VFC)
- Backface Culling (BFC)
- Occlusion Culling (OC)

View Frustum Culling (VFC)

This type of culling method removes everything outside the viewing frustum. This means that every object that is not in the view of the camera (what the player sees), will not be calculated and shown in the game.

Backface Culling (BFC)

Backface culling removes all faces that are in the same direction as the viewer. If the viewer only sees the front of an object, the back is not rendered (calculated) by the computer.

Occlusion Culling (OC)

This culling method removes all objects occluded by other object in front of viewer.
This means that only the objects that are in view are rendered. These objects, in contrast to Backface Culling, are rendered totally. When an object overlaps another object totally in the view of the camera, the object that is covered totally will not be rendered.

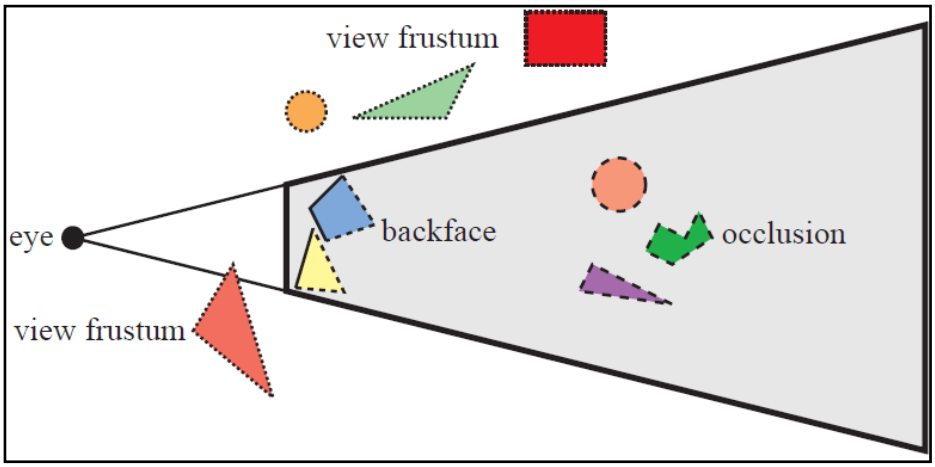


Figure 1 - VFC, BFC and OC

Evaluation Proposal

The evaluation proposal that I proposed to my teacher was:
"Implement View-Frustum Culling (VFC), Occlusion Culling (OC) and Back-Face Culling (BFC) into a Unity project and compare the FPS using different scene sizes for its properties."

The response my teacher gave me was:
"I think that should be fine, but VFC is the only one that I think you can more reliably do. Unity already does VFC and BFC. BFC you cannot really implement because you would need to remove triangles from the mesh and I don't think Unity allows you to. Occlusion Culling is a property in Unity that you need to turn on/off and bake. What you could do is implement VFC and make tests with different scenes and other aspects of OC. Also, you could play with multiple types of shaders, marking objects statics, marking lights, and so on."

Test 1 - Setup

Topic

For my first test I wanted to compare Occlusion Culling and baked lighting with Unity. To also test if the size of the light maps (quality of the lighting) influence the average FPS, I also did tests with heavier lighting (all values from the original lighting multiplied by two). I will perform the tests as a build and in the Unity editor.

To test the impact of Occlusion Culling in combination with baked lighting, I created the following test scenes:

- No baked lighting and no baked Occlusion Culling
- No baked lighting and baked Occlusion Culling
- Baked original lighting and no baked Occlusion Culling
- Baked original lighting and baked Occlusion Culling
- Baked heavier lighting and no baked Occlusion Culling
- Baked heavier lighting and baked Occlusion Culling

Setup

For the test scene I imported the Viking scene created by Unity (Figure 2). This scene contains a small Viking village and has an animated camera and a first person controller. For my tests I used the animated camera. By doing this, the tests would always be the same in terms of objects seen by the camera.
For the original lighting settings, I used the "The_Viking_VillageSettings" that was imported together with the Viking scene. For the heavier lighting settings I have multiplied all values from the original lighting by two.

To get the average FPS, I wrote a script that stores the current FPS value in a list. After 60 seconds (the duration of the camera animation), the values are stored in a text (.txt) file automaticly. The average FPS is also noted in the document.
To have more reliable results, I ran every setup 5 times. From the five values I took the average amount of FPS. The results can be seen in the 'Test 1 - Results' chapter.

Hypothesis

- Baked Occlusion Culling will increase FPS
- Heavier baked lighting will improve FPS

Figure 2 - Viking Village created by Unity - Asset Store: https://assetstore.unity.com/packages/essentials/tutorial-projects/viking-village-urp-29140

Test 1 - Results

No lighting - Build

	No baked lighting & No baked OC (Average FPS)	No baked lighting & Baked OC (Average FPS)
Test 1	124	130
Test 2	124	128
Test 3	130	128
Test 4	128	128
Test 5	125	130
Average	126,2	128,8

Table 1 - No baked lighting & different OC (Build)

Original lighting - Build

	Baked lighting & No baked OC (Average FPS)	Baked lighting & Baked OC (Average FPS)
Test 1	127	130
Test 2	126	127
Test 3	125	125
Test 4	125	130
Test 5	125	125
Average	125,6	127,4

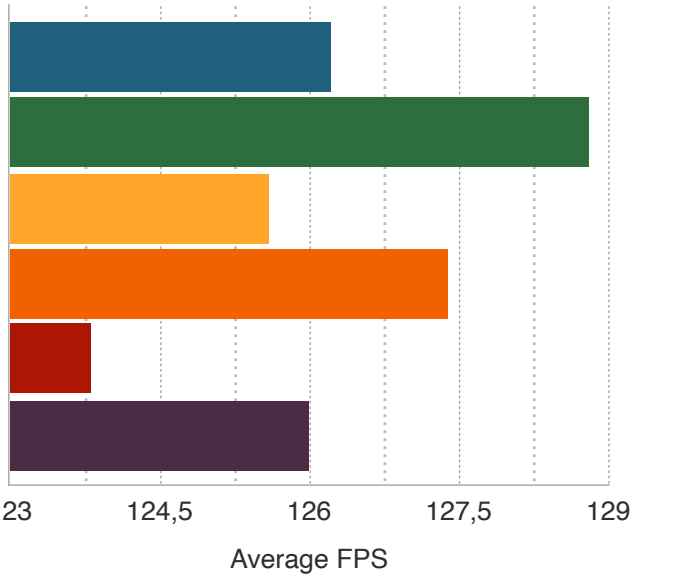
Table 2 - Original baked lighting & different OC (Build)

Heavier lighting - Build

	Baked lighting & No baked OC (Average FPS)	Baked lighting & Baked OC (Average FPS)
Test 1	126	125
Test 2	122	127
Test 3	123	124
Test 4	122	128
Test 5	126	126
Average	123,8	126

Table 3 - Heavier baked lighting & different OC (Build)

Diagram 1 - Results test 1 (Build)



Results build

As can be seen in diagram 1, the test with no baked lighting and baked Occlusion Culling got the highest average frames per second (FPS).

There is one issue with these test results. The results where gathered from a build of each scene. When an .exe file is builded in Unity, some magic is happening with the scenes that are selected for the build. This influences the performance of the scenes, and so also the average FPS. The FPS values will be higher than normal. This is why I also ran the tests in the Unity Editor. The results I get from theses tests will be different and more useful than a build.

No lighting (Editor)

	No baked lighting & No baked OC (Average FPS)	No baked lighting & Baked OC (Average FPS)
Test 1	119	121
Test 2	120	121
Test 3	120	121
Test 4	119	120
Test 5	119	122
Average	119,4	121

Table 4 - No baked lighting & different OC (Editor)

Original lighting (Editor)

	Baked lighting & No baked OC (Average FPS)	Baked lighting & Baked OC (Average FPS)
Test 1	118	120
Test 2	119	122
Test 3	119	122
Test 4	118	121
Test 5	118	120
Average	118,4	121

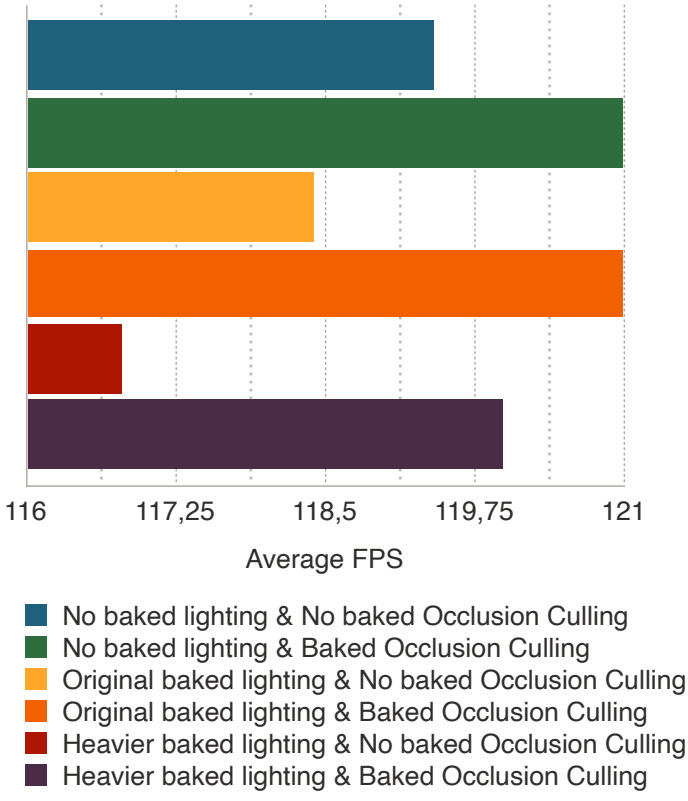
Table 5 - Original baked lighting & different OC (Editor)

Heavier lighting - Editor

	Baked lighting & No baked OC (Average FPS)	Baked lighting & Baked OC (Average FPS)
Test 1	117	120
Test 2	116	121
Test 3	118	120
Test 4	116	120
Test 5	117	119
Average	116,8	120

Table 6 - Heavier baked lighting & different OC (Build)

Diagram 2 - Results test 2 (Editor)



Results editor

As can be seem in diagram 2, the tests 'No baked lighting & Baked Occlusion Culling' and 'Original baked lighting & Baked Occlusion Culling' have a shared first place. 'Heavier baked lighting & Baked Occlusion Culling' has a second place. In fact, all the results are quite close to each other. It is hard to point out a direct winner, just because it has 1 FPS more. The conclusion I can draw from the results, is that baked Occlusion Culling always improves the FPS results. This can be seen with the build and editor results.

Results hypothesis

When looking at the results I can say that my hypothesis is 50% correct. The results from the tests with baked Occlusion Culling are higher than the ones without.
My hypothesis about the results of the baked heavier lighting are not correct. I always learned that big lightmaps can be ran better than smaller ones. If I look at the results, this statement can be discussed (since the difference in FPS is very small).

Test 2 - Setup

Topic

For my second test, I wanted to compare Occlusion Culling with realtime lighting in Unity. For this test I used the same setup as for test 1. I will do the tests as a build and in the editor itself.

To test the impact of Occlusion Culling in combination with realtime lighting, I created the following test scenes:

- Realtime lighting and no baked Occlusion Culling
- Realtime lighting and baked Occlusion Culling

Setup

I used the Viking Scene and used the animated camera. I changed the directional lighting mode to 'Realtime'. For the reflection probes I changed the type to 'Realtime' and the refresh mode to 'Every frame'. In the quality settings of the project I also enabled 'Realtime Reflection Probes'. Before I did the tests I ran the scene ones, so that the textures and lighting data could be loaded.

To store the FPS, I used the same script that I used for test 1. I ran every setup 5 times. From the five values I took the average amount of FPS. The results can be seen in the 'Test 1 - Results' chapter.

Hypothesis

The FPS of the build will be a lot higher than the FPS in the Unity editor. Since performing runtime calculation for Realtime Lights might be costly (https://docs.unity3d.com/Manual/LightMode-Realtime.html), the FPS will be below 100 frames per second.
The tests with baked Occlusion Culling will have a higher frame-rate then the ones with no baked Occlusion Culling.

Test 2 - Results

Realtime lighting - Build

	Realtime lighting & No baked OC (Average FPS)	Realtime lighting & Baked OC (Average FPS)
Test 1	125	126
Test 2	130	132
Test 3	130	125
Test 4	126	131
Test 5	125	128
Average	127,2	128,4

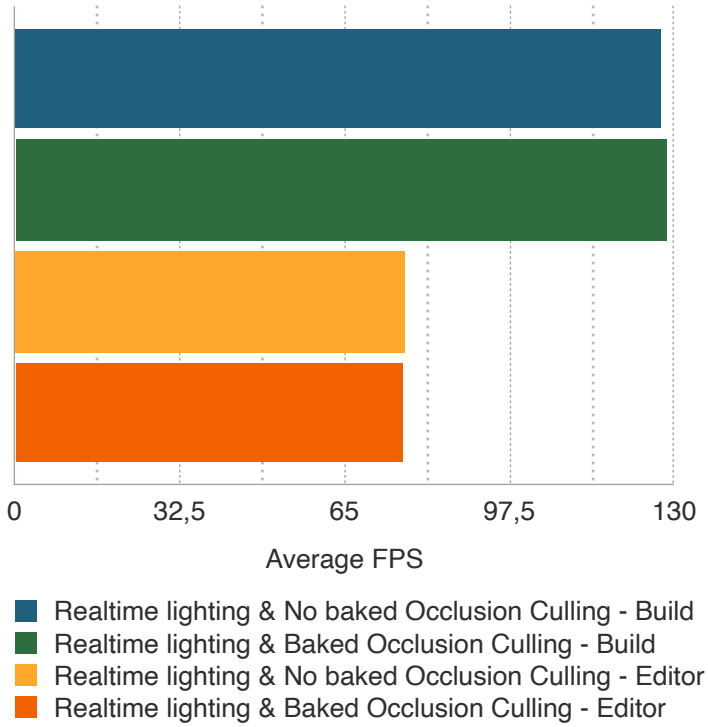
Table 6 - Realtime lighting & different OC (Build)

Realtime lighting - Editor

	Realtime lighting & No baked OC (Average FPS)	Realtime lighting & Baked OC (Average FPS)
Test 1	79	76
Test 2	78	76
Test 3	76	76
Test 4	77	76
Test 5	76	77
Average	77,2	76,2

Table 6 - Realtime lighting & different OC (Editor)

Diagram 3 - Results test 2



Results

As can be seen in diagram 3, the results of the realtime lighting from the build and in the editor differ a lot. This is due to the fact that a build has less to calculate than played in the editor.
The results from the build with baked Occlusion Culling are surprisingly almost the same or lower than the results from the tests with no Occlusion Culling. This can be due to the fact that Occlusion Culling not always increases performance. This depends on the specifications of the hardware of the computer.

Results hypothesis

My hypothesis was almost right. The build of the FPS is higher than the ones from the editor. My prediction that the FPS will be below 100 only applies to the results from the editor. So that one was 50% true. The results from the tests with baked Occlusion Culling are almost the same as the ones without. So with realtime lighting, baked Occlusion Culling doesn't really improve the FPS.