EURECOM
Sophia Antipolis

Final report

Lab 3 : Hammer

*3*DGraph

Student Author
Louis BARBE
Louis GALLAND
Irina MOSCHINI
Guillaume REQUENA

Teacher
Pascal GROS

Sommaire

1. Step 1 : build an office

Here is our reproduction of the office :

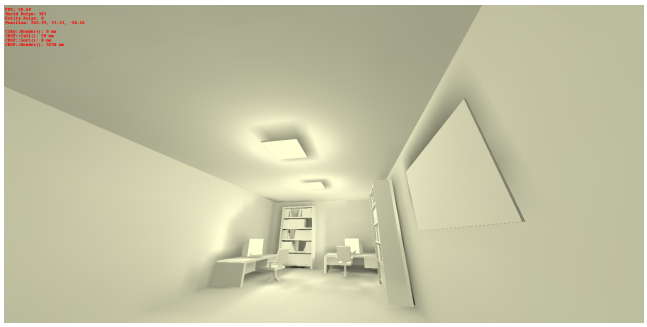
Figure 1: Final result of our office

We can define a hierarchy so that things can be moved or duplicated more easily.
- The root of the hierarchy would be the room (the initial box we created), the window and the door
- The children would be the first cupboard, the second cupboard, the *first office equipment* and the *second office equipement*
- In each *office equipement* , there would be the chair, the desk and the computer

With this hierarchy, we can move the entire room, but still have the freedom to move the cupboards or the office equipment if we like to.

2. Step 2 : compile an office

Here is what we obtain by playing with the *render* menu :



(a) `Result with the "Lightmaps" activated`

(b) `Result with the "Texture" activated`

The `Lightmaps` item allows us to see the impact of the lights we have putted as it displays the scene without the textures (everything has the same white color). Thus, the intensity of the white color is stronger when it is near a light source (the lights on the ceiling for example), while it is darker when the area is far from a light source (the ceiling near the camera is a dark gray).

While the `Lightmaps` item allows to study only the luminosity, the `Texture` item allows to study the textures we have putted without any luminosity. That is why the texture of the wall on the right of the camera is exactly the same as the wall in front of it. It allows to emphasize that, without any lights, the scene is not realistic.

Combining the `Lightmaps` and the `Texture` allows to have the final result, which is a lot better than having just one of these two items. Adding lights to the textures allows to create a realist room (put light on the screens of the computers for example).

3. Step 3 : compile a floor

Here is our reproduction of the Eurecom floor :

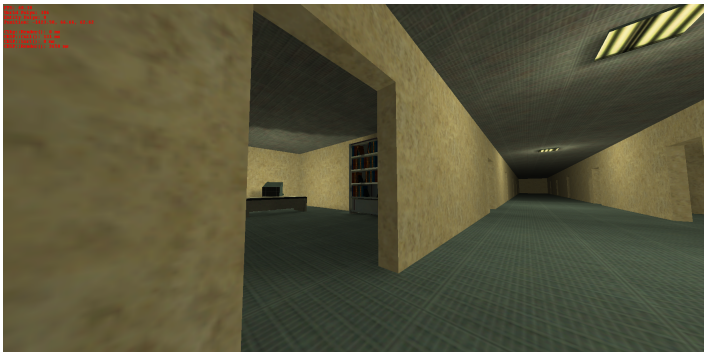


Figure 3: Final result of our office

4. Step 6 : analyse the scene

1. We can see that playing with the 3 items present in the `Cull` menu (that is to say `Backface`, `Frustom` and `VIS`) doesn't change anything visually. However, the number of polygons computed differs. Here is a table that recaps the number of polygons computed when different items are toggled :

Toggled items	Number of polygons computed
Backface, Frustom, VIS	386 polygons
Frustom, VIS	564 polygons
Backface, VIS	599 polygons
Backface, Frustom	833 polygons
VIS	964 polygons
Backface	1220 polygons
Frustom	1344 polygons
None	2075 polygons

We can see that toggling items allows to optimize the number of displayed polygons, as the more items are toggled, the less polygons are displayed.

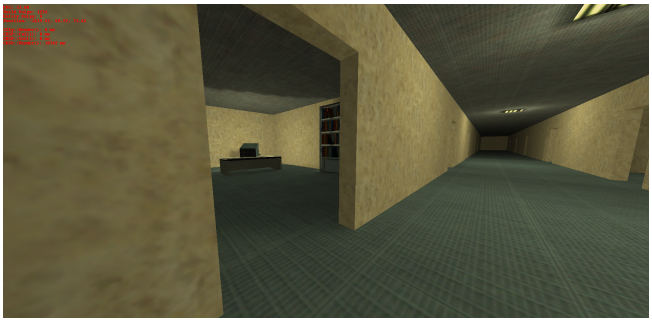
As learnt in the course, the backface optimization consists in computing only the visible faces of each object from the camera point of view. In our case, applying this backface optimization allows to compute 1220 polygons instead of 2075 polygons (which means that 855 polygons are backface polygons, whereas 1220 are frontface polygons).
The `Frustom` item performs optimization by computing only the polygons belonging to the view volume (and not those behind us, or next to us). In our example, it allows to compute 1344 polygons instead of 2075 polygons.
The `VIS` item disables the render for polygons that are not in the same room as the camera. It allows to compute 964 polygons instead of 2075.

As these 3 optimizations do not act on the same things, it is very efficient to performs all of them (compute 386 polygons instead of 2075 polygons without any optimization).

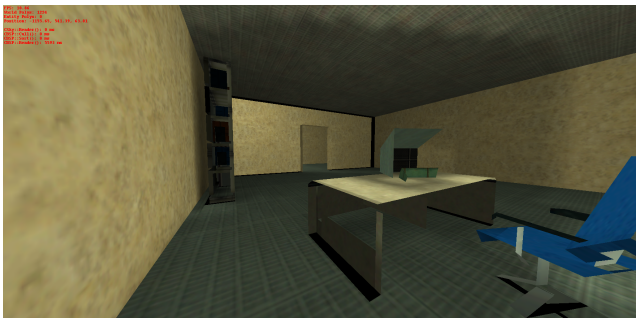
2. The `Freeze tool` allows to freeze the computation of the polygons from a point of view, and move in the scene after that. That is why after freezing, the number of polygons computed stays fix while moving. It is very useful to understand the optimizations by looking at which polygons are computed for a given optimization.

Without any optimization, freezing and then moving doesn't change anything concerning the visual. However, the price to pay is that a lot of polygons has to be computed when we freeze from point of view : 2075 in our case.

Here are the images we obtained when we activate the Backface optimization, then the Frustum optimization and finally VIS optimization using the Freeze tool :



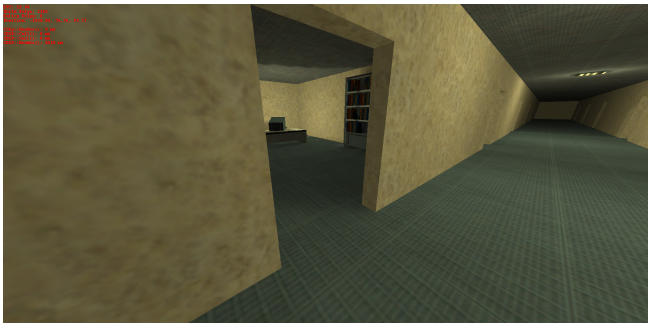
(a) Point of view 1: toggling of the Freeze item



(b) Point of view 2

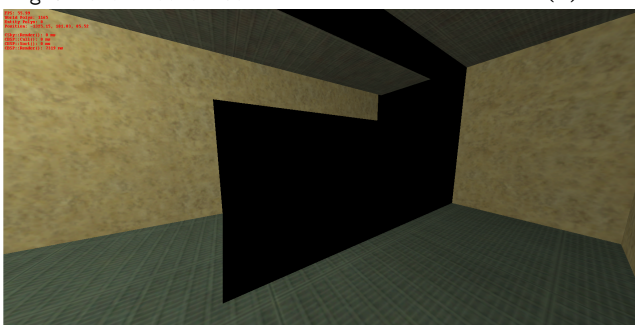
Figure 4: Backface optimization

We can see that from the point of view 1, all polygons are displayed, whereas when we move forward to look the backfaces of the objects, we can't see them : they haven't been computed. For example, the right side of the computer (that can be seen from the point of view 2) is not computed. This allows to display only 1224 polygons.



(a) Point of view 1: toggling of the Freeze item

(b) Point of view 2

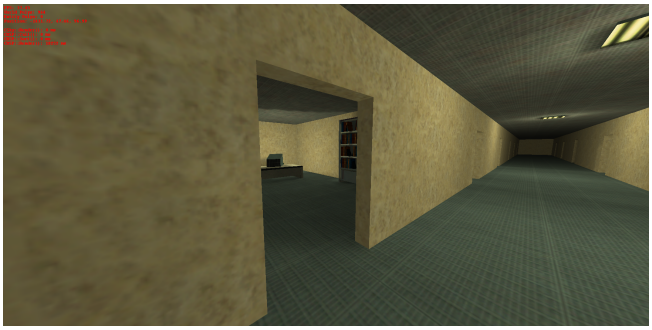


(c) Point of view 3

Figure 5: Frustum optimization

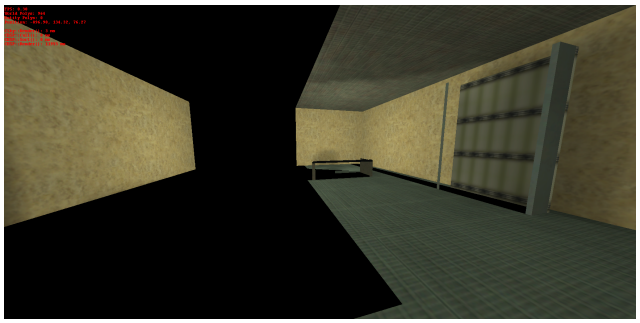
We can see that while everything is computed from the point of view 1, what is behind the camera is not always computed. For example, from the point of view 2, the room in front of us is totally black, not because it is dark, but because the polygons have not been computed. When we get into that room, a big black area is present (point of view 3) as the polygons have not been computed because of this optimization. It allows to display 1528 polygons instead of 2075 without optimization.

The Freeze item here highlights that all polygons of the room we saw when we toggled the item are computed (frontfaces and packfaces polygons). However, moving to the room next to it shows that not all polygons are computed (there is a big black area). The more we move far from this original room, the less the number of



(a) Point of view 1: toggling of the Freeze item

(b) Point of view 2

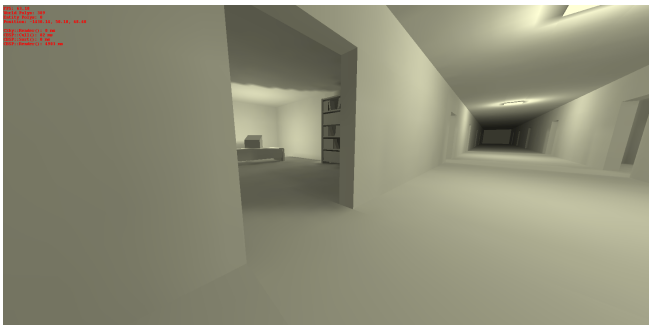


(c) Point of view 3

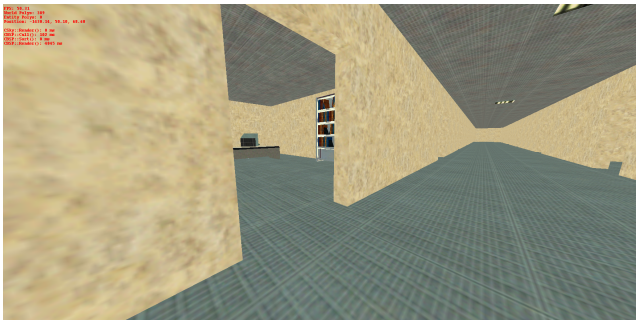
Figure 6: VIS optimization

polygons are computed. It allows therefore to compute only 964 polygons (the best of the 3 optimizations proposed).

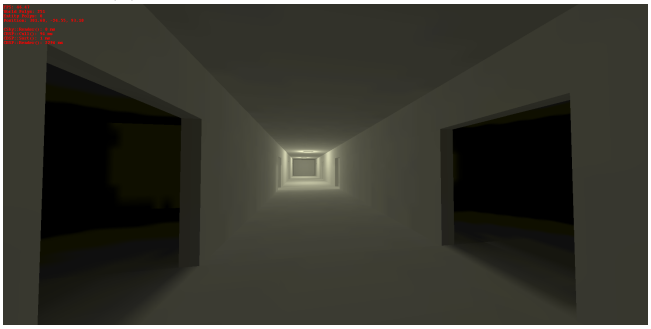
3. Here is the result obtained by toggling the items Lightmap and Texture from two different points of view.



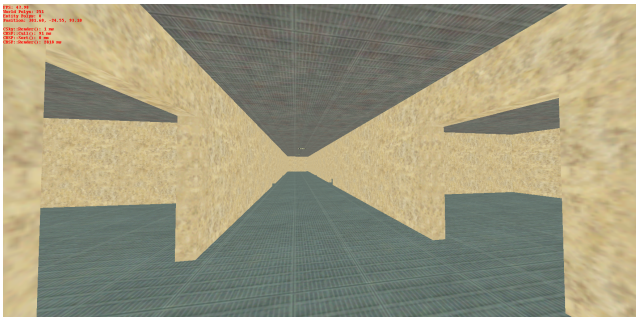
(a) Point of view 1: toggling Lightmaps



(b) Point of view 1: toggling Texture



(c) Point of view 1: toggling Lightmaps



(d) Point of view 1: toggling Texture

These pictures show that when we put a light in an area, the lightmaps displays it very bright (figure (a), we can see clearly the room and the objects inside) whereas it is extremely dark when there is no light (figure (c), we

can't see what is inside the rooms). On the opposite, the Texture doesn't emphasizes the presence of lights, but displays only the texture. That is why the picture (d) looks like having the same luminosity that the picture (b), even if there is actually no light.

4. The `Lightmaps` is a texture that is computed only one time, that gathers information about the luminosity of a scene. It allows to reduce the computational costs compared to the computation of the luminosity in real-time. To have the final result (with the good texture and luminosity), a multiplication is done between the original texture, and the the texture of the Lightmaps.

However, as this Lightmaps freezes the values of the luminosity, the price to pay is that it becomes impossible to change the parameters of the light : it is static.

5. The resolution of a Lightmaps is the number of pixels available to store our Lightmaps. The more pixels we have, the finest the details of the light there are. But a high resolution can reduce performance and increase memory usage.

However, the resolution of the Lightmaps is not necessary the same as the texture it is going to be mixed with. Thus, some artefacts can appear.