Demo my Sponza

Ignacio Cortizo Pol (S6088748)
Teesside 2016-2017

For the last assessment of the Graphics course, we had to render the Sponza scene using deferred. Also we must use some kind of physically based rendering as well as some anti-aliasing technique. All this should run at interactives frame rates at a modest resolution of 1280x720 in the computers at the lab (which have equipped an old Nvidia Quadro). I will analyse three different parts of the engine:

- **Shadow map generation:** at the beginning of the frame I render the shadow maps using a 1024x1024 DEPTH_COMPONENT32 texture.
- **Filling G-Buffer:** store all the the geometry information in a set of textures. I´m using a full fat G-buffer (Positions: RGB32F, Normals + roughness: RGBA16F, Albedo + Metallic: RGBA16F and Depth: DEPTH_COMPONENT24).
- **Drawing light volumes:** I´m using a full screen quad for directional lights a cone for the spot lights and spheres for the point lights. In this step, I also perform the shading of the scene.
- **Post processing and scene compositing:** at this step, I perform SSAO, a basic approach of DOF, FXAA and finally add a bit of ambient lightning taken from a cube-map (basic approach to environment mapping).

To time the application, I will be using glQueries (with the timestamp). This should give an accurate GPU timing. The application is being tested in Release (x64).

The following times are taken from my home computer:

- **CPU:** i5-4690
- **GPU:** Nvidia 960

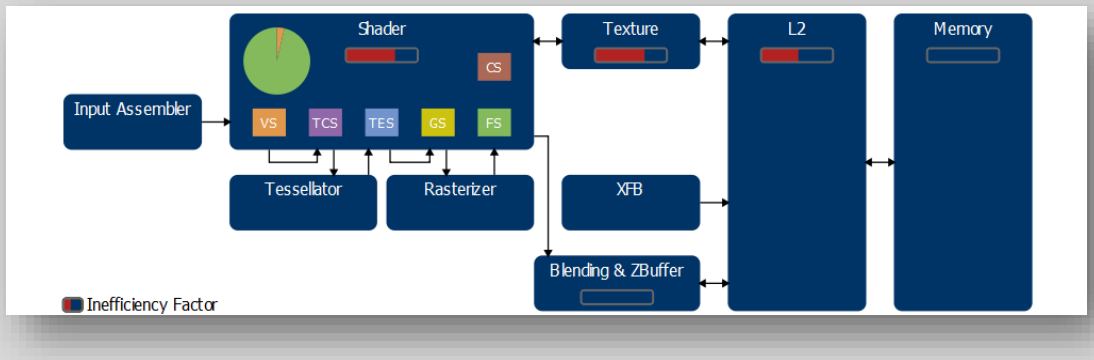
Timing (~8ms to compute the frame)

- **Shadow map generation:** 1.45ms – 18.1%
- **Filling G-Buffer:** 2ms – 25%
- **Drawing light volumes:** 2.8ms – 35%
- **Postprocessing and compositing the scene:** 1.9 ms – 23.7%

Looking at this values we can see that "most" of the time is spent on the light volumes. This is no surprise as there is where all the shading is being done. I´m using a microfacet model and also sampling 5 textures! The lowest mark is from the shadow map generation, which in my opinion is quite height (taking into account that only 3 shadows are being generated). The rest of the frame is spent filling the G-buffer and performing some postprocessing.

My thoughts about the result and how to improve it

First, I think that generating the shadow maps and filling the g-buffer could be done more efficiently as I´m not performing any kind of batch rendering or at least instanced rendering. I don´t think instanced will make a lot of an improvement as the maximum number of instances per mesh is just 6, doing those calls in a single batch will improve the performance by reducing draw calls (the meshes are not high poly so bandwidth won´t be a problem either). On the other hand if I had to improve some part should be on the shading. As we can see on the following image, representing the pipeline overview of a point light volume:



Most of the time is being used on the pixel shader (FS) and also we are hitting the texture manager and L2 cache hard. This is caused by the fact that we are using a full fat g-buffer and we are performing a lot of texture samples. To improve this I could pack some of the data or getting rid of the positions texture (and find the positions using the depth). I´m also performing smooth shadows here (Poisson Sampling(9 samples) + interpolating neighbour shadows) .Shadows could be precomputed for static objects which will remove all the Poisson sampling & linear interpolation (it should be still done for dynamic objects in the scene).

Finally we have the postprocessing and compositing part, and most of those 1.9 ms are spent calculating the ambient occlusion factor: 0.9ms. This time only represents the raw calculation of the AO, adding some blur takes other 0.3ms, making a total of 1.2ms (64% of all the postprocessing). DOF, ambient lightning (sampling a cubemap), FXAA and compositing all the textures take just 0.7 ms. The “hot spot” of this step is of course the AO factor calculation, I´ve been tweaking its values since I added it into the engine. I´m doing 16 samples, which initially looked really bad but increasing the radius of the sampling made it look much better. Also, I removed all the divisions of the code and added some constants (which improved a bit the performance).

I would like to mention that the antialiasing technique I´m using (FXAA) is relative cheap and it blurs the scene and also doesn´t handle well all the aliasing. Other techniques like MSAA,SMAA etc. will give better results but with an extra time cost.

I will show now the times of the scene in the university computer:

- **CPU:** Intel Xeon
- **GPU:** Nvidia Quadro 4000

Timing (~32ms to compute the frame)

- **Shadow map generation:** 5.07ms – 15.84%
- **Filling G-Buffer:** 4.27ms – 13.34%
- **Drawing light volumes:** 14.6ms – 45.6%
- **Postprocessing and compositing the scene:** 8.4ms – 26.25%

I leave this results to the end as I don’t think they are relevant. Nvidia Quadro series are a set of graphics cards used in workstations (CAD) and the drivers used in those cards are not developed to run games. Anyway, I tried to improve the performance of the engine in this

card, I first reduced the number of samples used for the softshadows (from 9 to 5) this made the scene reach the 30FPS (it was going at 20FPS), I tried reducing the samples of the SSAO from 16 to 8 but I only gained around 2FPS, this low improvement is normal as most of the time (45.6%) is spent drawing the light volumes. To improve more the performance, I would create a custom set of shaders for this type of cards (using some kind of normalized Blinn-Phong) because the PBR approach that I'm using is expensive. Also, the other improvements that I exposed previously will also improve the performance on this card.