INFO-H502 Project Report

Stargate – Maxime Verstraeten

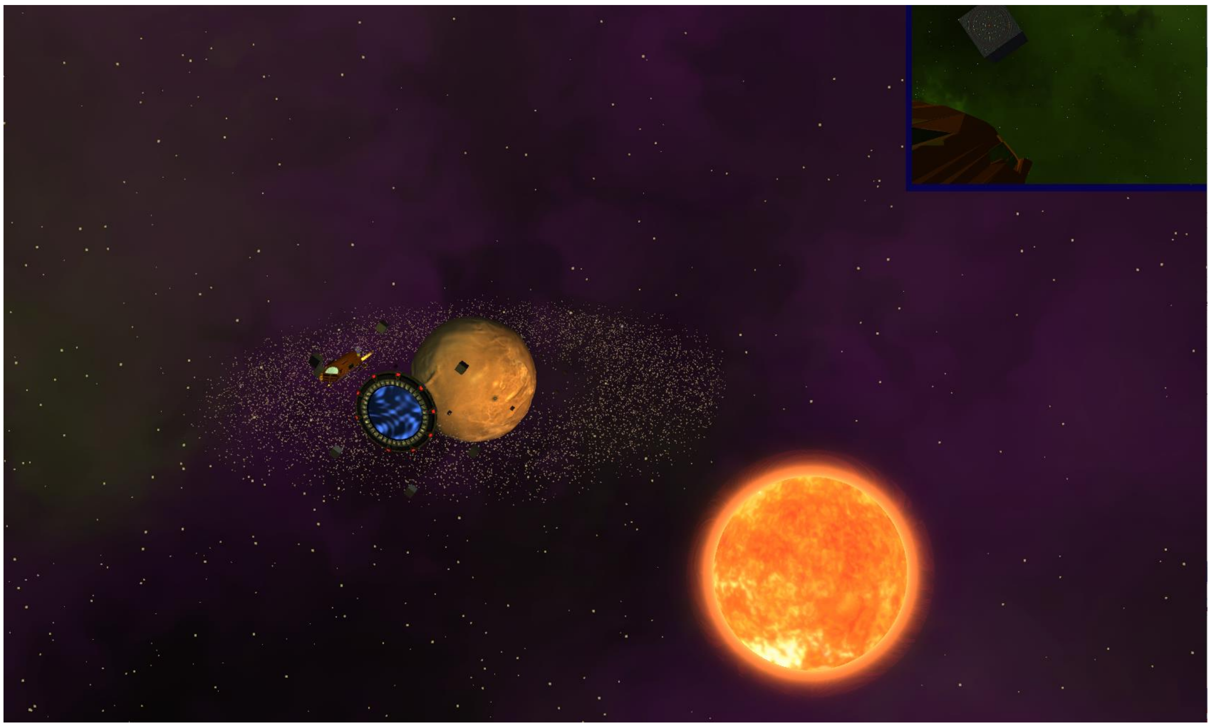
This report describes the functionalities implemented in this project.

The “MainFunctionalities.txt” file in the root folder contains a recap of the implementations and all the key bindings available.

The game takes place in space, in the TV series Stargate context. It includes different models such as a Jumper (spaceship), a stargate, a planet, a sun and some weird-looking cubes. All models except the missile, the asteroids and the light bulb have been done in Blender 2.80.75 by myself.

The player moves in the environment with a 6DOF camera and can take control of the spaceship with also 6DOF.

I used a skybox found online (http://wwwtyro.github.io/space-3d/) and I implemented an automatic procedural generation of stars using a Python script to generate coordinates and size. The output file is read to create multiple instances (to increase performance) of a yellow cube all around the scene.

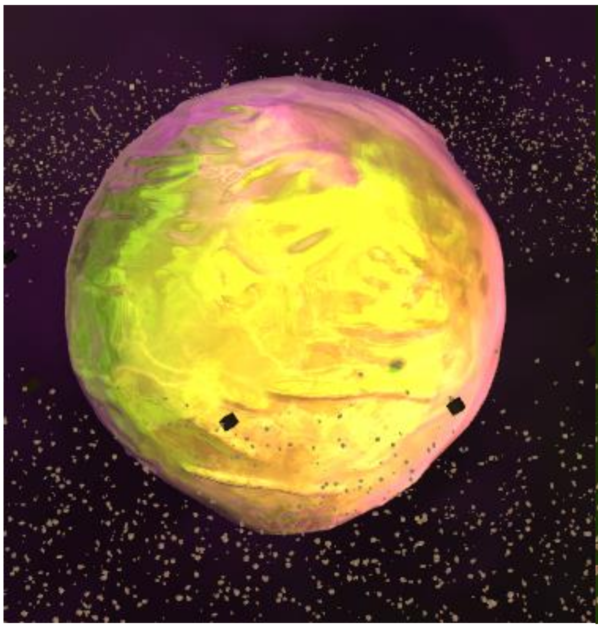
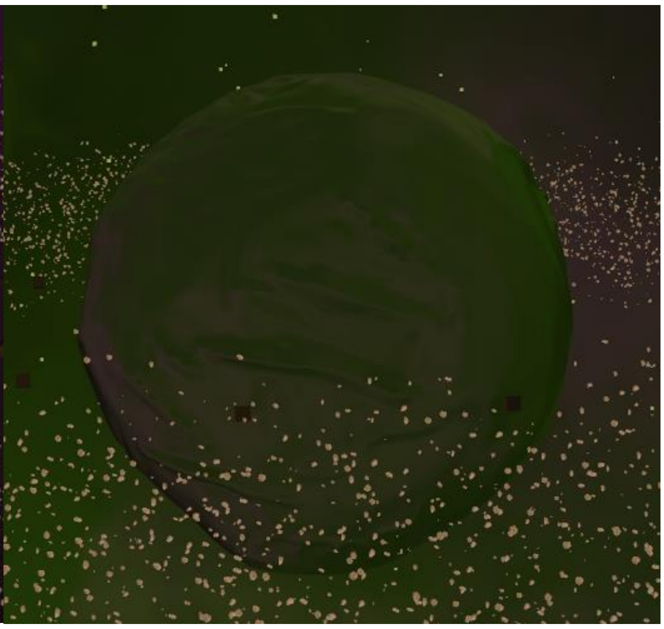


This gives the impression that the stars are real stars as they slightly move around when travelling instead on staying fixed with a classical skybox.

The asteroids around the planet are also rendered using instancing, allowing to render 10.000 of them with only a slight performance drop.

The color of each model is computed by combining Phong’s shading (ambient, diffuse and specular lighting) as well as emission. Emission maps along with diffuse and specular maps allow for detailed models.

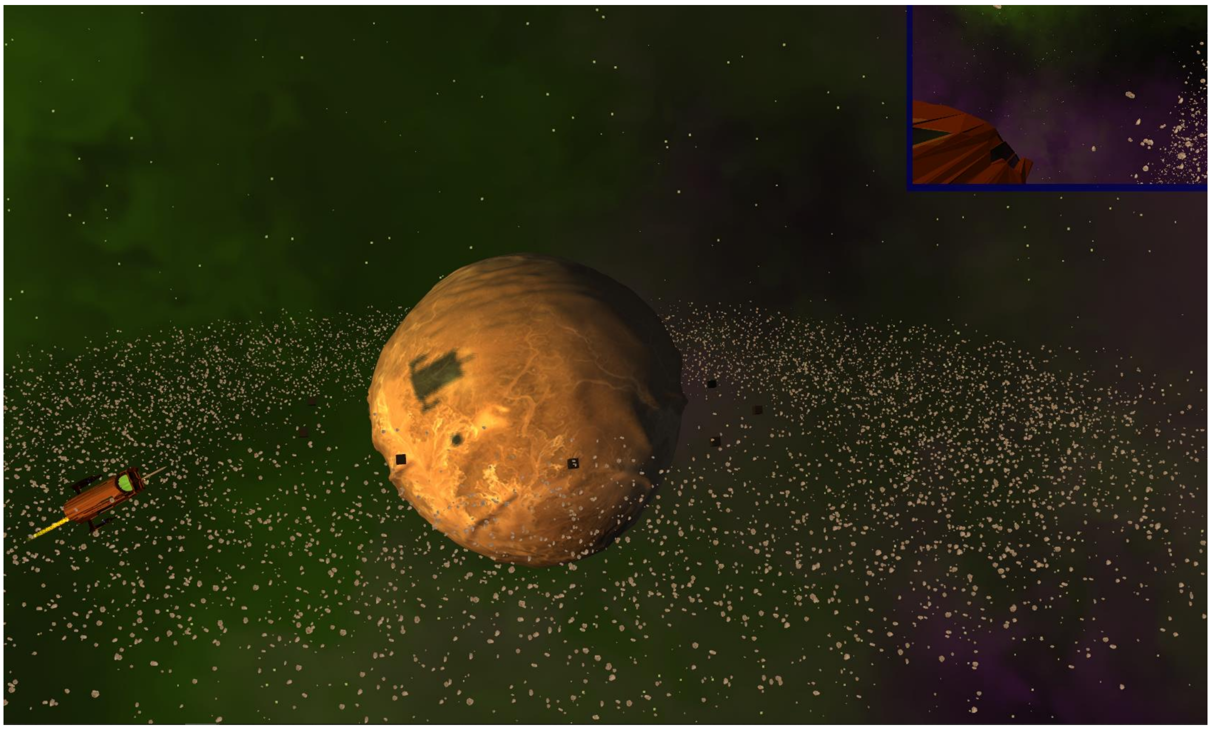
Environment mapping is also available: a reflection map is used on certain parts of the ship to give it a metallic/chrome aspect. To fully appreciate the effect, refraction and reflection can be toggled on the full planet (using U and I keys).

A few light sources are available. The sun itself is of course a very strong (and reddish) one. An Edisson type lightbulb is turning around the ship at the start, giving the opportunity to observe the specular and diffuse lighting on the ship. The stargate is emitting some low-intensity blue light from its center (the event horizon). The ship itself got a small spotlight under its nose and finally, the camera itself got a small flashlight. I used forward rendering as I have only a few light sources.

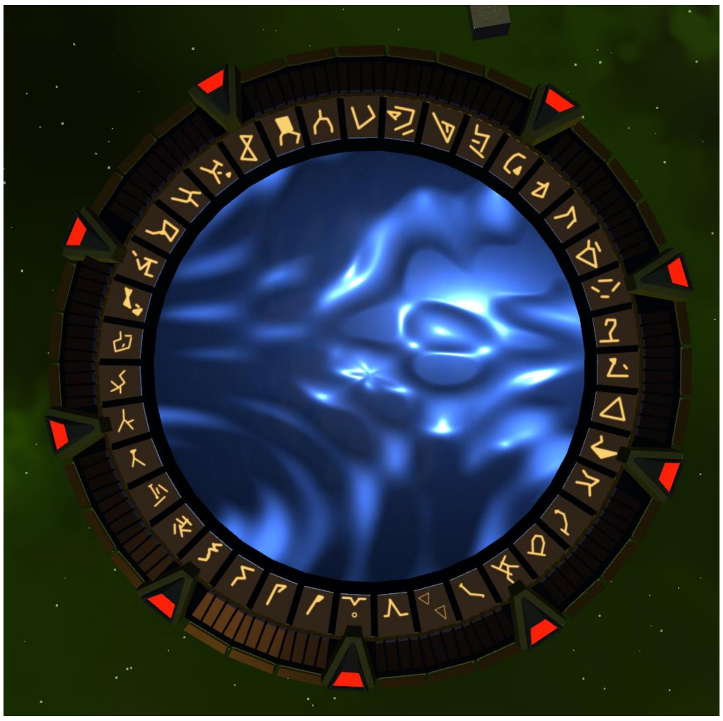
The lights are, depending on their strength, attenuated over a distance. This is not highly realistic but a very interesting results as, for example, the ship will turn red close to the sun.

The sun will also project soft shadows on every model. This can be seen with the weird cubes, the ship and the missile all projecting shadows onto the planet, but also the planet projecting shadows onto the ship when behind its dark side. Same goes for the stargate hiding the ship from the sun and so on…

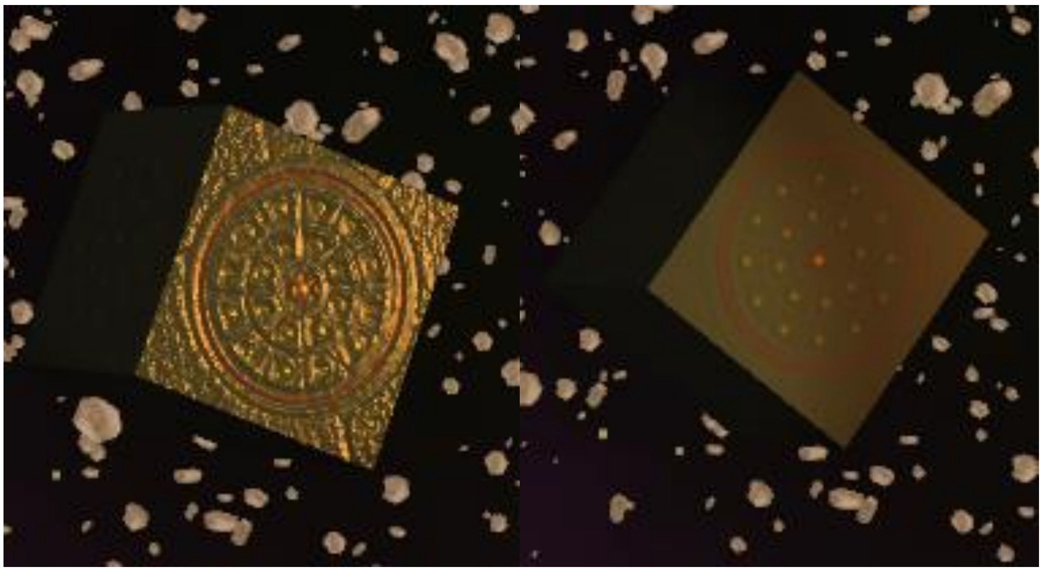


The sun got its own shader, inspired from a lot of shaders on shaderToy, to give him a corona and some glow, which changes over time.

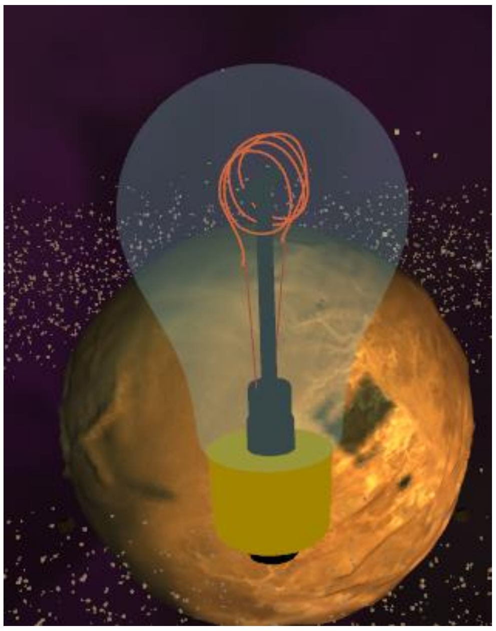
Same thing is done for the event horizon in the stargate which is a combination of 4 circular waves and a spiral.



The weird cubes feature normal mapping.



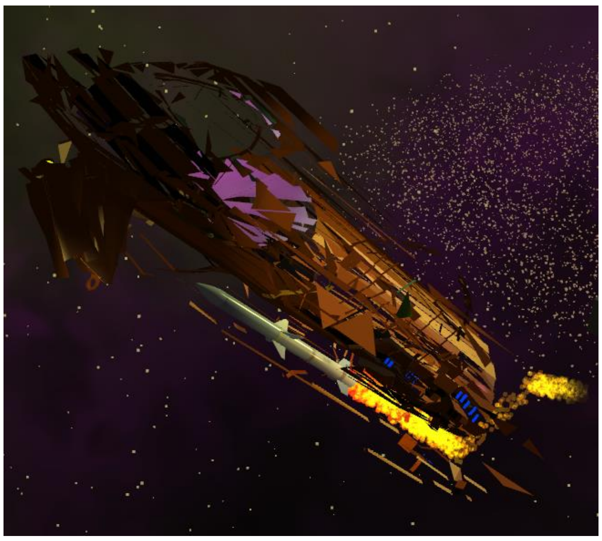
And the light bulb features blending.

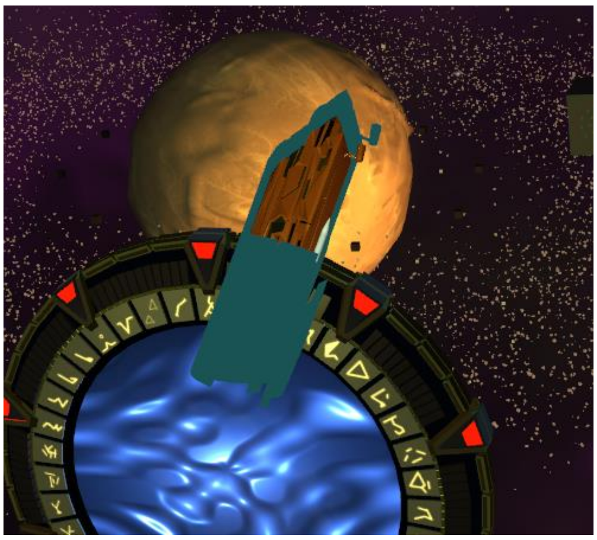


The missile comes with some particle effects with additional random on color, transparency, position and velocity to give it a somewhat realistic effect.

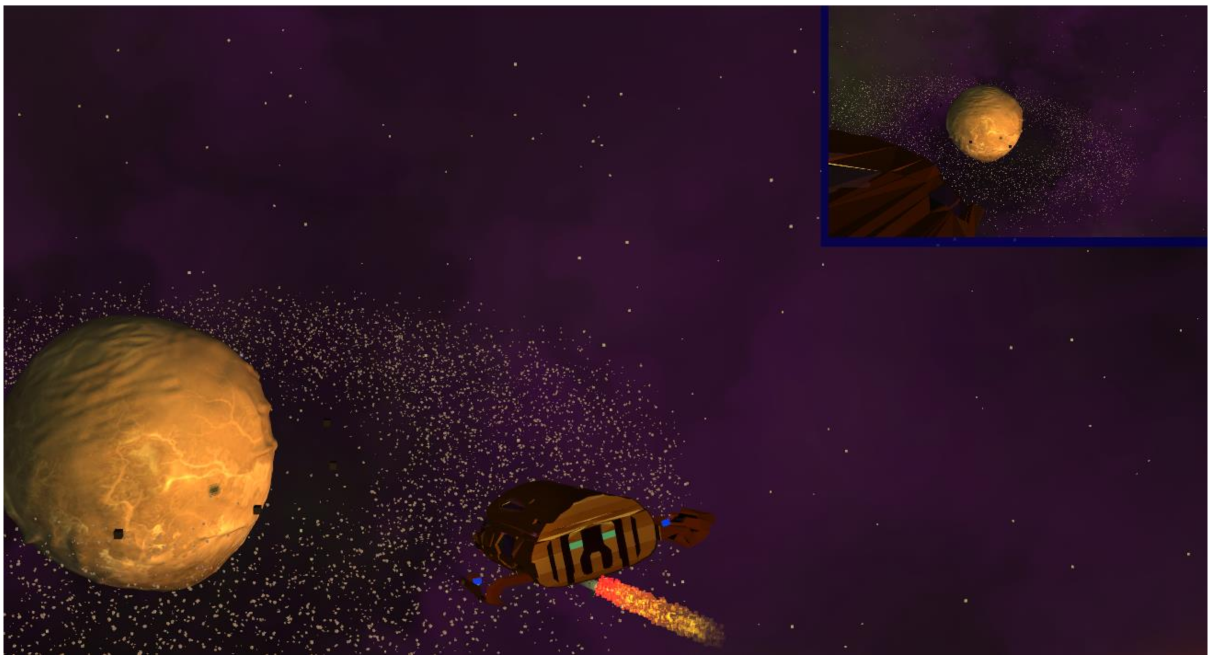
The ship can “explode”. Basically, this is just the fragments moving along their normal in the geometry shader. The effect is interesting albeit not working properly due to some normals being wrong in the model (Blender design mistake).

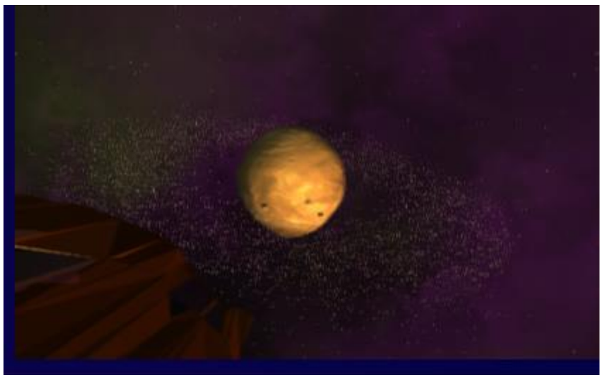
It can also be outlined by using the stencil buffer, so that the ship stays visible behind other objects.

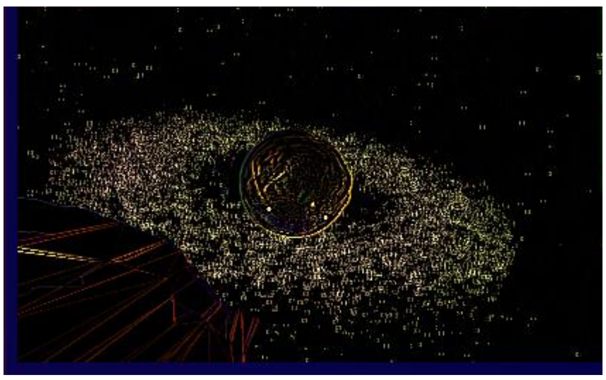




I used framebuffers to have a second POV being on the right side of the ship. On this framebuffer can also be applied some post-processing effects such as sharpening, blur, edge detection and grayscale.







To increase performance, I used face culling when relevant (i.e. I couldn't use it on my complex models as some normals are facing the wrong way). It was also sometimes mandatory to use it in order to avoid artefacts when using transparency (for the sun and the light bulb glass).

I also implemented a very basic anti-aliasing method (multisampling so MSAA) to remove some jagged edges and improve the rendering of small details such as small stars and asteroids.

As a bonus, I added some background music, which really helps when waiting at startup… Indeed, beginner mistake, my models are very high-poly and tend to take some time to load.