Coursework Report

Rafal Ozog
40217610@napier.ac.uk
Edinburgh Napier University - Computer Graphics (SET08116)

Keywords – coursework, graphics, parishilton, trump, nvidia, siria, iluminati, cats

1 Introduction

Can you imagine 3D space, where you can create whatever you want? For ages people could only experience that in their dreams. It is really worth to think about - we are the first generation, that can create imaginated 3D spaces so easily. Computer graphics is a field of computing that allows us to do that. In my opinion it is what makes computer science great - that's a kind of magic that allows us to present our visions - and also allows us to experience others peopleâĂŹs visions.

2 New effects implemented in my project

2.1 Terrain - 3D mountains

I have decided to implement a 3D terrain instead of 2D plane from my first coursework. I have used a height map, which I have downloaded before and four textures, which we were using previously in the workbook exercises. The main part of this effect is a special function "generate terrain" in my .cpp file, which calculates positions and normals for each single point of the terrain (geometry). Then the function passes them into the shader. There are four textures binded to the terrain. The shader uses the values calculated by the function "generate terrain" and calculates a ratio of use each of the texture for each single point of the terrain. Finally the shader returns the final colour, based on the textures and their ratios.

2.2 The earthquake

One of my original effects I have implemented is **the earthquake**. We can control that using the **'M'**. When we press the key, we can notice that the whole terrain and all meshed placed on it shake. To make the effect more realistic I have decided to implement also the subsidence of the buildings (a house and a windmill). The secret of this effect lies in a **random number generator**. There is a range of positions, where the terrain and buildings can be in (when the effect is switched on). Each position is chosen by the number returned from the **number generator**. It also allow the programmer to control the power of the earthquake easily, because when we increment the range of possible numbers, we also increment the range of physical positions of the meshes (the earthquake power grows with the numbers range). The effect occurs automatically when the UFO vehicle drops the bomb - but also switches off automatically after few seconds.

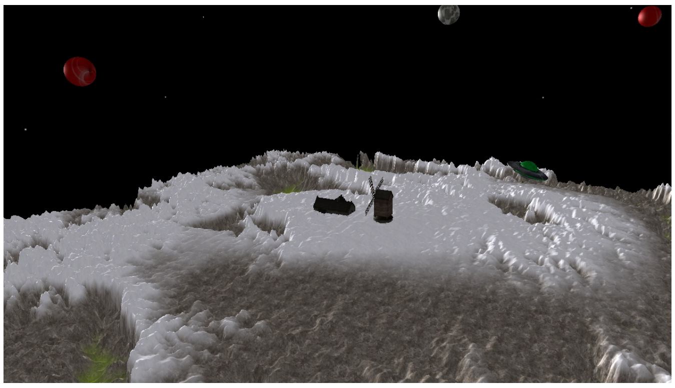
Figure 1: **Terrain 3D** - Mountains

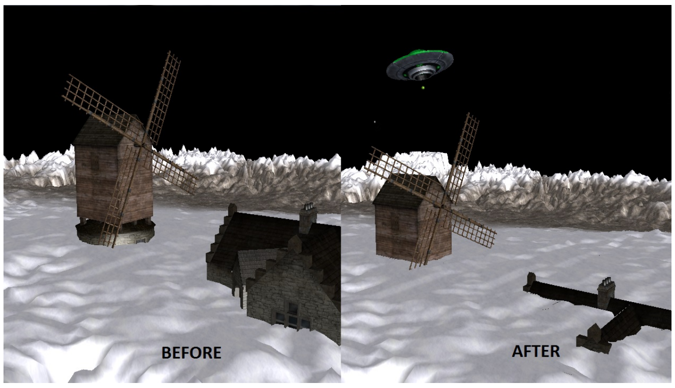

Figure 2: **The earthquake** - before and after

2.3 UFO vehicle and the dropped bomb

In the second part of the coursework I have developed the idea of UFO vehicle controlling, which was the main part of the functionality in the first part. That was the base. The first new element I have added to the vehicle is **a bomb**, which is a quite realistic, futuristic bomb, looking like a sphere (as we know from physics in space the aerodynamics does not matter, but this shape provides the best capacity of the explosive material), being placed under the vehicle, holded by a strong magnetic (invisible) stream. As we could suppose - in this version of the game **we can finally drop the bomb!** To do that we need to press the 'E' button on the keyboard. After the bomb is dropped, we can notice few special effects. Firstly, there occurs **two lighting effects**. One is a general explosive illumination, which is provided by a shader (exactly by the fragment shader in the "eff" effect) and **special float uniform, which I have added by myself**. The uniform is passing the information about a moment of explosion (how much brighter the light should be in a particular moment). The illumination takes few seconds. The second lighting effect here is a spot light, which is switched on exactly under the detonation location and faces the meshes in the middle of the map. Another effect added to the bomb detonation is **a recoil effect**. The vehicle is pushed very strongly by the force of explosion what causes its rotating and changes the trajectory of the flight.

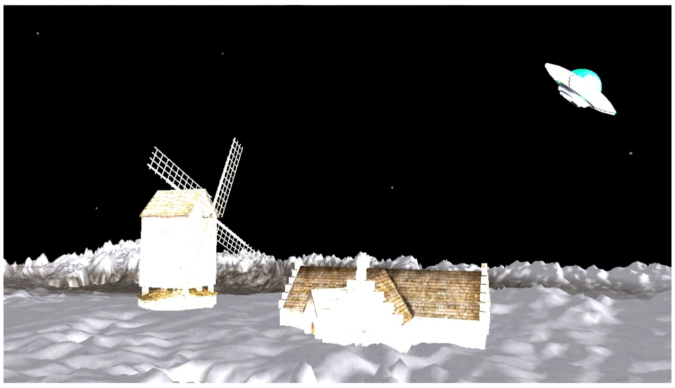

Figure 3: **The UFO vehicle** - a moment of the bomb detonation

2.4 Blur Motion effect

The Blur Motion effect is the first of **post-processing** effects implemented in my coursework. The output is rendered not directly to a screen, but to frame buffers. There are two independent of them to work. One of them stores the freshly rendered frame and the second - the previous one. The special shader uses constant called "blend factor" to mix both frames together in the ratio from that constant. The final frame is rendered as a texture on the screen quad. As the result we can see the blurring effect on the screen (in the motion of meshes and in the motion of cameras). That effect has been implemented as an another option of displaying and can be controlled by the user (can be switched by the **'B' and 'N' buttons**).

2.5 Fireworks

Another original graphic effect presents **the fireworks**. They are physically placed just near the 'house' mesh and can be activated in any moment by pressing **'H' button**. When the user presses the button, then the sequence of events hidden in the 'update' function starts. Firstly the set of meshes moves up quickly. When they reach particular height, the wooden sticks break out of the fireworks and are moved into their starting positions. In the same time the exploding (round) parts of the fire-

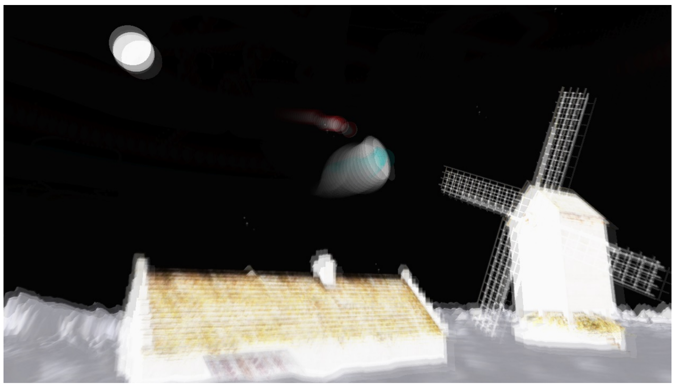

Figure 4: **Post-processing effect number 1** - Blur Motion

works... explode. To achieve that effect I have used the special geometry shader, which changes the positions of all fragments based on their **actual positions, normals and the 'explode factor'**. The factor is passed through the uniform and defines how far from the origin the particular piece of geometry should be (in each moment of explosion). The programmer could control the process

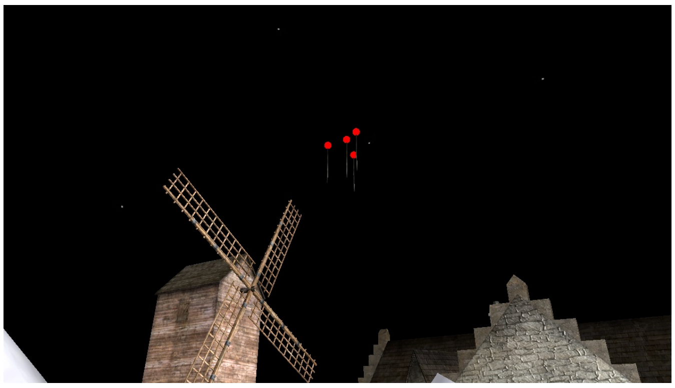

Figure 5: **Fireworks** - Flying before the explosion)

of explosion through changing the value of this factor - however, in my project it's done automatically. When the range of explosion reaches previously set value (when the pieces are almost invisible), the 'explode factor' is reseted and the exploding parts of the fireworks are translated into their starting positions. Then the whole process restarts.

2.6 Magic ball - look from the outside!

The motive popular from the films about magic inspired me to implement similar thing in my project. Everyone sometimes tries to see himself from the another perspective, from the other viepoints, from the outside. Then we imagine the perspectives in our heads. I wanted to allow users to experience that feeling in my scene. To be able to 'go out' of the physical world and look at the whole world (scene) from the outside, through the little magic ball - like through the little portal from their minds. To do that the user has to press the **'T' button**. It will move him to the rabbit hole... That's not the end! The user can still control his view (distance from the magic ball) in this pos-

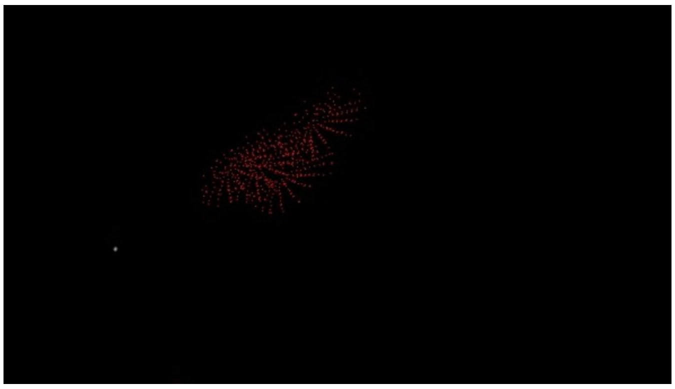

Figure 6: **Fireworks** - 'Bang'!)

Figure 7: **The magic ball - pre-idea** - 'Harry Potter and the Philosopher's Stone'

tion using **'U' and 'I' buttons** and even the view inside the ball (using the mouse).**Is it magic? No, it's just a graphic!** To come back to the real (virtual) world the user needs just to press **'Y'**. The effect is the **second post-processing effect** implemented in my code. To achieve that I have used another frame buffer, where to the output is rendered and finally - presented as a texture on the magic ball.

3 Future - what will the third version bring?

I would like to present few features that I would like to develop in the future.

3.1 Fire, fire, fire everywhere!

The first feature I would like to add in the future is **fire**. It would be added as an additional effect of explosion, but also as a part of flying fireworks animation. Until now I have placed the fire into the map, but I couldn't solve the conflict between the mouse and the partciles (they were moving with each mouse move - I don't know why). I hope it will be chenged soon.

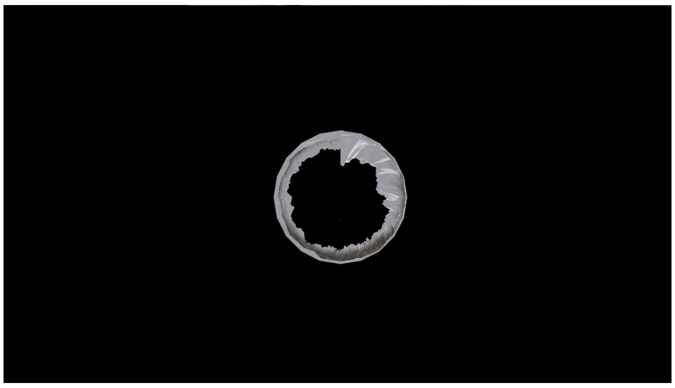

Figure 8: **The second post-processing effect** - 'the look from the outside'

3.2 UFO vehicle shooting

The second feature I tried to implement was shooting. The idea was quite trivial. I wanted to add the 3 balls (red spheres), which would change the colour after being shooted. I was going to implement that option within the vehicle control - on camera 4, but I didn't solve problem with setting the right target of the shoot. In the moment of shoot the bullet was occuring in front of the vehicle, but I couldn't order that to fly in the proper direction.

3.3 Terrain deformation

Another feature I didn't manage was the terrain deformation caused by a dropped bomb. I was going to implement that accessing the right pieces of geometry in the 'generate terrain' function, but I didn't do that successfully. I hope I will find out the proper solution as soon as possible.

4 Optimization

I tried to implement the variety of methods to optimize the code.

4.1 V and P matrices

The main change in compare to the first part is that I took all V and P matrices out of each loop placed inside of the 'render' function - and I have placed them before the loops. Now the matrices won't be set for each mesh independently, because it is not needed as they can be common for all meshes.

4.2 Lights

Another feature that is implemented in my coursework is use of one effect to provide all lighting. I have used one effect 'eff' to provide both spot light and directional light (main light source). The final light colour for the meshes is calculated within the same shaders what allows to avoid additional GPU calls (however, to be honest - this particular feature has already been implemented in the first part).

4.3 Less effects in the code

I have improved my 'render' function from the first part. There, the meshes with normal mapping were rendered by another effect than meshes with simple texturing. It has changed. Now both types of meshes are rendered through the same effect('normal eff' became a part of 'eff'). Another adventage of this approach is that the code is now much more readable and - because of that - easier to modificate. We don't have to implement changes separately for both types of meshes anymore.

4.4 Less uniform calls

Another thing that has been improved is the amount of uniform calls. I have refactorized the code looking for the repeated calls that don't change the uniform values. The example of removed uniform call is the texture uniform, which had been called separately for each mesh in my first project. Right now it is called only for the first mesh and the rest is using its unchanged form. It has also made the code more readable.

5 At the end...

Working with the computer graphics shows us the magnificence of the possibilities that are hidden within our computers - but also gives us the great opportunity of boundless creation. We have received amazing tool that allow us to create scenes, environments and worlds from our dreams and imagination. That is probably one of the tools that we've discovered, but we don't know yet what possibilities are hidden behind that. **Imagination** - that is where it has started - and where it leads us!