Ziad Beyens
Hamza Nougba

INFO-H402 Virtual Reality: Driving Car

I. Introduction

As part of the course "INFO-H402 Virtual Reality", a project must be done by the students. This project can be either a game or a medical visualization following the section of the student. In our case, the implemented game basically consists in a car driving in a town. OpenGL v4.5 with GLU, GLM and GLAD librairies has been used for that purpose. The game illustrates many concepts seen in the course such as the lightning, shaders, etc.

II. Description

A. Model loading

Free car, town and street lamp model have been loaded using assimp.

B. Lighting

The car uses headlights during the night. There are also 4 street lamps that enlighten the town. All the lights have ambient, diffuse and specular components. By default, Blinn-Phong lighting is used but the user can enable Phong lighting by pressing the 'B' key. Also, gamma correction can be used by pressing the 'G' key.

The car headlights imply a moving 'spotlight'. The car position and its front vectors are sent to the 'carShader' for this purpose.
Refraction and reflection can be applied on the models by pressing the 'R' key. Also, a directional light is used from the sky to apply the light of the sky.

C. Environmental Map

The map represents a small town with a few houses and street lights in a brightful night. For the sky, a free cubemap has been used.

List of some shaders used:

- The town: ‘streetShader’.
- The lamps: ‘lampStreetShader’.
- The floor: ‘floorShader’.
- The sky cubemap: ‘skyboxShader’.

D. Animation

The user can drive the car by using ‘W’ to go forward, ‘S’ to go backward, ‘A’ to turn left, ‘D’ to turn right. The camera is locked to the car.
By clicking left, the camera can be locked at the same place while seeing the car driving away.
As in all car games, by pressing the car pedal, the car gains acceleration. The car loses acceleration over time otherwise. You can turn left the car only when moving.
You can also brake by pressing the opposite direction key.
This logic has been implemented in Camera.h.

E. Post-Processing

To apply post-processing filters, we use ‘screenShader’. It uses ‘quadVAO’ that fills the entire screen. We use a framebuffer to put the result of the previous fragment shader as a texture to this shader. Once done, post-processing is an easy task by using kernels. Grayscale, sharpen and edge detection filters have been implemented.
The user can switch the filter applied by pressing the ‘F’ key.

III. Demonstration

The source code is available here:
https://github.com/zbeyens/opengl-game

The file demo.mp4 or the file can be download here:

https://drive.google.com/file/d/1AWVdWlhsg6jstHiQv_WU6w3TWHTMmuho/view

IV. Conclusion

Features:	
OpenGL v4 using shaders for lighting	done
Environmental map	done
Animation	done
Any viewpoint around the objects/scene	done
Model loading and textures	done
Reflection and refraction	done
Specular and diffuse light	done
Phong and Blinn-Phong lighting	done
Gamma correction	done
Post-processing filters	done