ECOLE
POLYTECHNIQUE
DE BRUXELLES

INFO-H-502 « Virtual Reality »

Project report

Van Heirstraeten Arthur

000408667

ULB

Introduction

In the context of the Virtual Reality course given at the *Ecole Polytechnique de Bruxelles*, students were asked to to develop their programming skills and more specifically their OpenGL skills by taking part to a project. This project takes the form of a game to be developed using only OpenGL through implementation of multiple functionalities seen during the course and the practical sessions. These functionalities will be detailed further and the way they are implemented explained in this report.

Game description

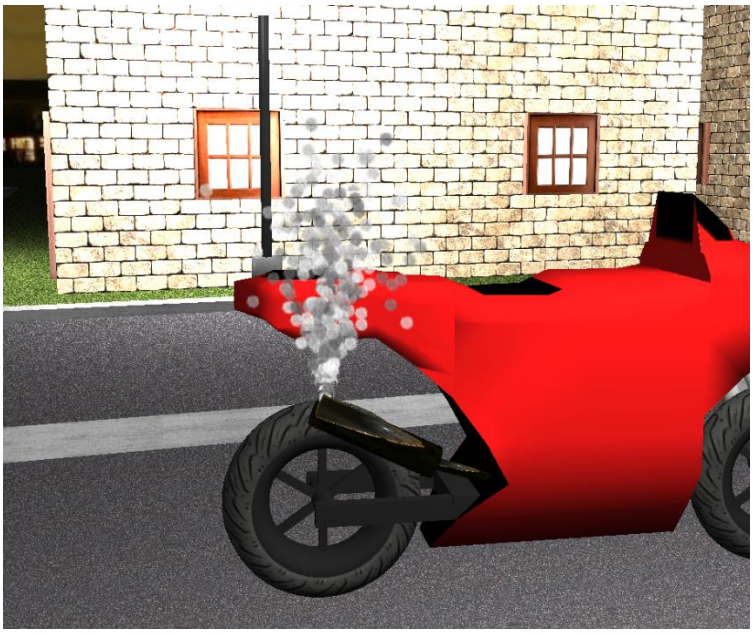
As there was no specific guidelines for the project' theme, it was decided to create a game based on a personal point of interest. This is why the game represents a bike driving around a small residential area. The bike is able to move forward, backwards and to turn. The game is taking place at night in order to be able to play with multiple light effects. All models have been created for the project but most of the textures come from online sources. The game is surrounded by a skybox following the camera movement and a giant tile of grass for base floor.

The colour of each model is rendered using Phong's Shading. However there are multiple types of light. All of them are white lights but we have spherical lights representing the light posts and one spotlight on the motorbike following its movement. Considering the large amount of lights in the scene, rendering all of them in Forward Rendering would have been very computationally heavy so a workaround was used. In fact only the ten closest to the camera light sources are processed. Another solution which has not been implemented here would have been to use Deferred Rendering. The lights are attenuated over distance and each element has its own parameters of diffuse and specular rendering, giving elements a more natural look.

In order to generate the whole game scene, a lot of vertices are required but instead of storing all of them, an intermediate class was created. It has the objective of storing the model matrix of an element but by referencing an already existing VAO. This means that each time an object is rendered, it is the same VAO that is just moved. This is done for each road and house elements.

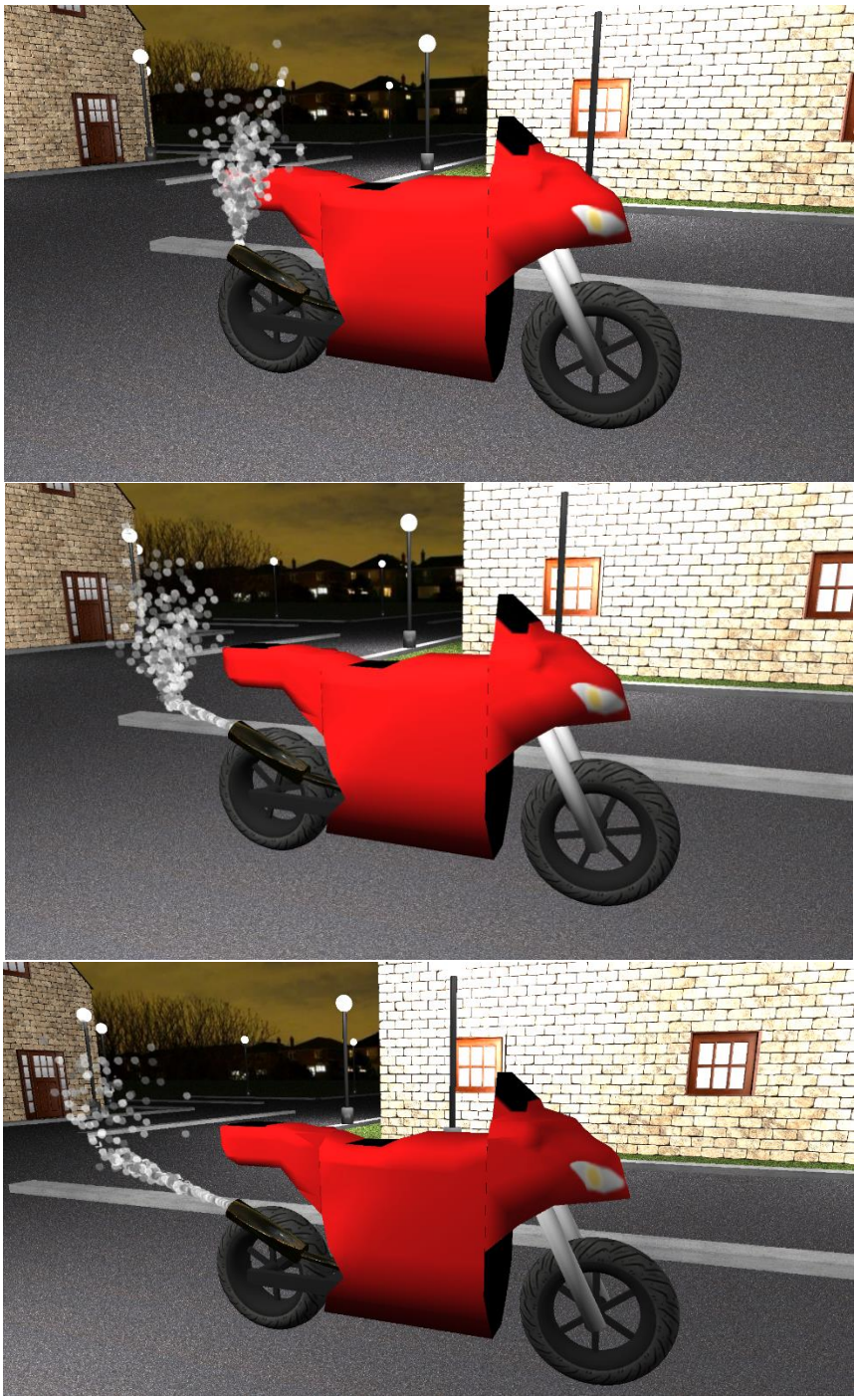
The bike comes with a smoke particle effect. Those particles are emitted at the exhaust port of the bike and have their own life cycle once generated. It uses the same principle of instantiation as explained previously. They each have a random aspect with different transparency and small random movements making them more realistic. They always face the camera horizontally but not vertically.

On the exhaust of the bike, a reflection of the skybox is done to give it a metallic mirror aspect.

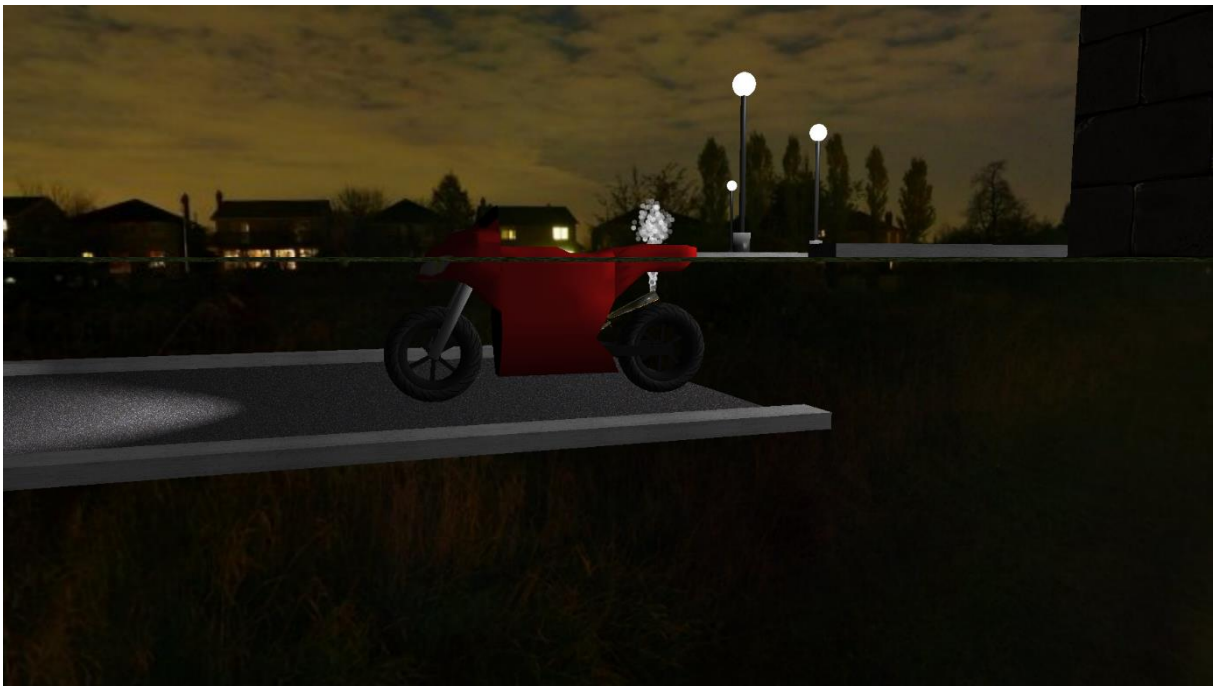


Instead of using complex 3D models for the wall, normal mapping is used. It gives the illusion of relief on the bricks. However it is just an optimisation using a normal texture.

The bike can be controlled through keyboards input by the user for both acceleration, braking and turning. As the bike is a complex model composed of multiple elements which have to stand as one piece, multiple rotations equations are used. Each sub-model is rotated around the main axis of the bike when the bike has to turn and when the bike drives, the wheels also rotate according to the speed of the bike. This is done using Rodriguez equation.



Two different types of collisions have been implemented, each one using a different technique. The first one is the gravity. In fact the bike checks at each cycle if it is still on a road by using line interception equations with road planes. This calculation is done at the center of the bike. If the bike is no more on a road, it will fall down until it encounters another road if it encounters one.



The second type of collision is the border collision. To avoid the bike from falling all the time by going through borders, another collisions detection method was used. It detects from a defined trajectory if a point goes through the plane of the border. If it does, it process the intermediate length at which the object should stop. It will then stop the bike at this collision spot. This is done for front and rear wheel depending on the direction and for both wheels when turning.