Computer Graphics CSE 40166

Final Project Report

Artur Pimentel de Oliveira

Lucas Barbosa Parzianello

Introduction – the project

In this project, the goal was to build a 3-dimensional demonstration that contains a controllable 3D vehicle over an uneven terrain with obstacles. Collisions are detected and proper physics applied.

Main functions

During the development, a few changes on the task division previously stablished were made. Originally, Artur was the responsible for the physics and car controls and Lucas, the responsible for the scene building and camera view. The usage of libraries to take care of the physics of the demonstration and texturing changed this division.

Different improvements were made by both of us. The physics part – one of the biggest challenges – was almost entirely taken care by the Physijs plugin for the Three.js library. Our work was to adjust some necessary parameters – trying to simulate a realistic environment. The camera 'lock' on the car was stablished by Artur and Lucas did the car construction, part of the scene building. The camera lock was done using a Three.js feature that allows the link of a camera to a mesh object (the car in this case). The car object comes from a JSON file containing its vertices, faces and normal vectors. Then this file is parsed by a Three.js function to retrieve this information and build a mesh. The file also contains a list of materials, which are used to replace textures and specify values such as specular shininess, color diffuse and opacity.

The height map feature – reading height values for the terrain building – could not be accomplished on time. Some problems of synchronization doing a XML HTTP request led to the discard of this feature.

Running the program

The program uses a minimalist version of the famous Three.js library, and therefore objects are loaded from JSON files rather than OBJ ones. To run the program, the user must be sure to enable his or her browser to access local files, or run a simple http server in the project folder and access the html file from the local server via browser. To apply physics we use the Physi.js plugin for Three.js.

Functioning

The program setup is very simple. A car is dropped from a pre-defined height in the middle of an empty terrain. Then, the user can drive the car using the keyboard's arrow keys. Up key accelerates the car and down key breaks/backs up the car. Left and right keys can be used to steer the car. While driving, the camera follows the car showing a third-person perspective. The screenshots below show this functioning. Note that the car body and wheels are different objects related by a hierarchical modelling.

Figure 1: The car is instantiated above the terrain

Figure 2: Camera position relative to the car during driving

Challenges

Using external libraries helped our implementation in many critical aspects of the coding, such as JSON parsing and loading, camera and lightning controls, texturing and collision detection. There is a drawback in this, though: to use external libraries also means the team must expend some time getting familiar with them, which would be unnecessary if we had stuck with the WebGL code presented during the semester. Poor documentation always posed a problem, and numerous tests were needed in order to fine-tune our project.

In addition, we did not apply all features intended successfully. There are no obstacles aside the terrain itself. Lastly, the heightmap used to generate the terrain could not be imported from an external file.

Learning experience

Having all the theoretical and practical experience from the programming assignments gave a good start for us and helped immensely with all computer graphics notion for implementing the functions efficiently. Besides, applying those concepts—such as hierarchical modeling, texturing, lightning, camera handling—on our own in this project made us incorporate them in a much more natural manner.