ICG - Project Documentation

Alexandre Ribeiro

April 14, 2024

1 Introduction

This document serves as a logbook to the various decisions I made during the development of my 'Introdução à Computação Gráfica' project.

Besides keeping an online version on Overleaf, I also made sure to upload a PDF version to the GitHub repository, making sure that both were updated.

2 Description

My project, as of April 14, 2024, is a 3D model of a city which contains various events to make it feel more alive.

3 Update Logs

This section documents every commit I make to the GitHub repository.

3.1 April 14, 6pm - Project Start

3.1.1 TLDR

- Initial setup (offline imports and https server)
- Defined model import method (OBJ+MTL, using obj-simplify to improve OBJ model)
- Defined rendering method (only render when needed, for now only when controling camera)

3.1.2 Log

A lot has been added in this commit.

The project file, index.html, has been added. It was taken from one of the exercises, so that I already had something functional to work with.

As I always do in class, I started by adding an offline option to the imports. Basically, I went and downloaded the files that I needed (three.module.js and OrbitControls.js), then I created an 'imports' folder where I placed the files in a way so that it would match the import map. Lastly, to make it work, I had to run a python http server, by opening PowerShell (Linux terminal also works) and typing 'python -m http.server'. All I have to do to access the project is go to 'http://localhost:8000/' and it's done.

With that done, I could start working on the project. The first step I had to make was decide what I would use for modeling my city. I did a lot of research, and came to the conclusion that if I wanted to model it myself, then I would have to choose a tool that is both easy to use and could also work in Three.js, and the best option I came across was voxel modeling, with the MagicaVoxel tool. Voxel modeling is relatively intuitive, and MagicaVoxel is a free tool that not only offers a vast amount of modeling features, but also has an implemented rendering tool that allows different materials, which includes emissive (light source). This would help in observing the effects that light has on the models I make.

The second step was making sure that I could load MagicaVoxel models to Three.js. MagicaVoxel offers a lot of export formats, but the two that were relevant to my project were OBJ and VOX. Both presented some challenges as I tried to load an example model into Three.js, but I ended up going with OBJ due to performance differences (VOX loaded each cube individually, while OBJ merges faces). OBJ is by no means perfect, I had to use an external tool in order to simplify the model so that it would take less resources to render and still it has a tough time rendering, due to other issues that I will try to solve in the future (ex: the model is somewhat separated by color).

The third and final step to this commit was solve the performance issues. Basically, every time I tried to run the project, my CPU usage would shoot up to 100%, which is unsustainable. I tried different ways of improving the performance, mainly simplifying the model as best as I could, but what ended up being the most logical solution was changing how the project renders the scene. The way it was initially meant that the scene was constantly being rendered, which is hard to maintain when the models are as complex as the ones I will be using, so, to solve it, I made it so that the scene is only re-rendered when there is camera movement. To do that, I added an event listener to the camera controls (OrbitControls). The result is very good, at least for now. I will have to revisit these methods once I start adding animations.