Tisstracted Documentation

CG UE SS 2020
Final Submission

Andreea Bogdan 11718711
Benedikt Klinglmayr 11808590

Implementation and Features

Our game is based on the provided framework. We started by building a simple room from basic shapes. We quickly reached the point where we were ready to load complex objects. We decided to use the .obj file format. This is very well known and widespread. This enabled us to create our own objects with Blender as well as to use many freely available models. A library was used to load the complex shapes [1].

For the illumination we have used directional lights and point lights. To others the bloom ceiling lamp donates a discreet ambient light.

The next step was the basic implementation of the game logic. We decided to introduce the Physics engine bullet [2] as soon as possible. On the one hand we wanted to minimize the integration effort, on the other hand collision detection is an elementary part of our basic gameplay. We implemented a simple collision detection to detect the player's touch with a table. As soon as a player touches the table we can activate the items. We also use Bullet to place the items on the table.

The next major step was to display the game data. Information such as time, items, waiting list places etc. should be displayed using text. With freetype [3] and the corresponding documentation we were able to implement this feature quickly.

After we had implemented our game logic, we could provide the first prototype for the first release. After that, we concentrated primarily on incorporating the corresponding effects. The implementation of these effects will be discussed in more detail later.

Gameplay

The game starts with a simple textual tutorial (Skip with J)
A user can move freely in space (W,A,S,D; U = Jump)
When a user touches a table, the item texts change colour to green and the user can insert an item by using the number keys (1,2,3,4,5)
if the user is lucky, some students leave the table. If he/she succeeds in chasing away enough students, he/she advances in the waiting list and receives one of the coveted examination places.
If there are still too many students in the room after the end of the playing time, the player has lost.

3rd Party Libraries

A lightweight single-h-file object loader. [1]

Bullet Physics SDK [2]

Freetype Font Rendering [3]

irrKlang Audio Engine [4]

[1] https://github.com/Bly7/OBJ-Loader
[2] https://github.com/bulletphysics/bullet3
[3] https://www.freetype.org/
[4] https://www.ambiera.com/irrklang/downloads.html

Effects

Bloom

We have implemented bloom with the help of the corresponding tutorial on learnopengl [5]. Especially the explanation of the shaders were a great help for the implementation. It was necessary to integrate the described system accordingly. For this purpose we used a common buffer for all objects of the game. To display the depth information correctly we used a depth buffer. If bloom objects are drawn, we write the corresponding information into the 1st and 2nd colour buffer. If "normal" objects are rendered, the data is placed in the first colour buffer, in the second one black pixels are inserted to display the depth information.

During the implementation we encountered 2 significant problems. At the beginning we always cleaned the wrong buffer, so bloom objects were never deleted and always just added to the image. Problem number 2 resulted from the missing color in the light shaders. Since this value was not set for non-bloom values, the depth information was not updated and therefore any object with bloom effect was always in the foreground. We could easily fix this by setting the bright value to black.

WHERE?

The bloom effect can be seen on the ceiling of the room. The large, cuboid ceiling lamp gives a more realistic picture than without bloom.

Video Textures

For the implementation of the video textures we have not used a special source. For the implementation we have used already existing methods and classes of the ECG framework. Basically we load a set of frames, and react to new frames accordingly. In regular intervals we play the next frame, thus the impression of a video is created. The whole thing was packed as material and can be assigned to an object.

WHERE?

Several monitors were placed in the room. These show a dynamic texture.

Particle System

For the particle system we used a tutorial which was linked on the LVA Tuwel page [6]. There was a quick overview of the basic functioning of a particle system. Our goal was to build a simple system that allows a comfortable use of particles in the code. We implemented a **ParticleGenerator** class. This takes care of all stages of the life cycle of a particle. Basically, it was necessary to create particles dynamically, draw them and then discard them again.

WHERE

The particle system is the 5th item that a user can use. It is an abstract representation of water. To display the particle system you have to move to a table until the item writing turns green and then use the corresponding item (key "5").

Hierarchical Animations

For the implementation of the hierarchical animations we used the slides of the theoretical lecture on the corresponding topic [7]. Basically we wanted to simulate cars whose tires move to their parents (car) accordingly. We have created a class of car that takes care of its own geometry as well as that of your children, i.e. the tyre. If the position of the tire is updated depending on the frame time, all corresponding tires will be notified. These then calculate the new position and the rotation independently. The result is a vehicle where the tyres behave accordingly.

WHERE

If you look out the window, you can see vehicles driving by.

Normal Mapping

Unfortunately we failed to implement normal mapping due to lack of time. Although we successfully implemented normal mapping, after the integration of Bloom there were problems that could not be solved in time. The implementation was kept in the code, but the functionality was not used.

[5] https://learnopengl.com/Advanced-Lighting/Bloom

[6] http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/

[7] https://tuwel.tuwien.ac.at/pluginfile.php/1721131/mod_page/content/37/Animation_SS18.pdf