Computer Graphics UE
Submission 2
Group: Sickness

Fausto Heraldo Sifuentes Caccire, 0607000
Andrea Gangl, 1025756

Implementation

Gameplay

Our game starts in the main hall of the hospital. There are numerous medikits hidden throughout the level. Some can contain one of the 3 parts of medicine needed for the cure but some can also contain sedatives. The more time passes, the crazier the player gets. If the player picks up a sedative he regains a little bit of time before he loses. Once all three medicine parts have been acquired, the player has to locate the medicine mix table and activate it with ENTER to receive the cure.
For the implementation of the 3D game aspect we have used stairs to get to the second level of the hospital.
To pick up a medikit, get near it and when you see a text (e.g. “medicine0”), press enter. To activate the medicine mix table, get near it and when you see a text, press enter. Have fun.

Effects

The following Effects were implemented:

- Light Mapping (0.5)
  We used Blender to bake the Lightmaps of the walls. After getting the lightmap we used Photoshop to multiply the texture+lightmap and put the resulting texture back on the mesh.
- Water + Fresnel-Shading, Normal Mapping (0.5)
  Overall the following tutorial series was used:
  https://www.youtube.com/playlist?list=PLRIWtICgwaX23jiqVByUs0bqhnalNTNZh
  The steps were of course adjusted to our framework. The code can be found in the “Water.frag” and “Water.vert” files. For the waves a dudv-Map was used to distort the texture coordinates. The Fresnel effect uses the vector pointing to the camera and the normals of the water plane. The normals are extracted from a normal map. To achieve the Fresnel effect the dot product between the normals and the view vector is used. To control the effect, the “refractFactor” can be changed using the pow function.
- + Reflection (1)
  For the Reflection we used a FBO. We render the whole Scene (without the Waterplane) to the Reflection Texture but we have to consider a few things. First: We have to clip away everything that is UNDER the Waterplane. This is achieved by using “gl_ClipDistance[0]” in the “main” shader (in our case, “Phong”). We calculate the dot product between the clipping plane and the world position and render the scene and use the plane “`0, 1, 0, -waterHeight`”. The second thing we have to do is to adjust the camera. We have to translate the position of the camera downwards for the distance of the height of the Waterplane. We implemented that code in the “XCamera” class.
- + Refraction (0.5)
  Here the process is pretty much the same as the reflection. What we need to do differently here is that we have to clip away everything that is ABOVE the Waterplane. We use the plane

"`0, -1, 0, waterHeight`" here. To map the refraction and reflection texture correctly on the plane we calculated the normalized device coordinates.

- Depth of Field (1.5)
  For this effect we created another FBO. We need the Depth Texture in order to implement a correct depth of field effect. Two FBOs were used, one to render the whole scene and one to blur the whole scene. The "Gauss" and "DoF" shaders are responsible for the blur and depth of field effect respectively. Basically we use the Depth Texture to blur the pixels which are further away but not the ones which are nearer to us.

Complex Objects

We have multiple complex objects in our scene: complex chairs, wheelchairs and a humanoid creature.

Animated Objects

Our animation is a cube which rotates around the player. The rotation is independent of the player's movement, e.g. the player moves and the cube will not stop rotating. Another animation would be the head of the humanoid creature. When some time passes, the head of the creature starts moving.

View Frustrum Culling

View Frustrum Culling is implemented in the Camera class.

Transparency

We have two occurrences of this effect.

1.) When a Medikit is picked up, a GUI sprite will be rendered on the top right corner. This GUI sprite is transparent.

2.) If you look at the water we have implemented alpha blending at the edges.

The transparency can be turned off and on again with F9.

Experimenting with OpenGL

We have the following key bindings:
F1: Help
F2: Frametime on/off
F3: Wireframe mode on/off
F4: Texture Quality
F5: Mipmap Quality
F6: Cursor in Game Window or out of window
F8: Frustrum Culling (show number of meshes)
F9: Transparency on/off
TAB: Camera switch (free camera/static camera)
ENTER: Activate
ESC: Exit the game

We have used FBOs for the Water and Depth of Field effect. VAOs and VBOs were of course used for the objects.

Features

Water features: Further water features we've implemented are: lighting with respect to the waves/Normalmap, alpha blending for a softer transition from the Waterplane to object edges,

water depth and smoother waves. All those effects are in the Water fragment shader and can be seen on the water object itself.

Another feature is the Blur effect. The more time passes the more blurry the vision (DoF) gets (this simulates that the player gets sicker and sicker).

Illumination/Texture

Basically all objects are illuminated by one fixed light source. We used the Phong lighting model. All objects in the game are also textured. Please notice that also the Waterplane is correctly illuminated. The objects (with exception to the water plane) were textured using Blender and UV mapping. Afterwards they are exported to .obj and .mtl files and .dae files. These files are then loaded by the program which takes care of the rest. For the walls we have used Lightmaps with 3 fixed light sources.

Tools used for Model Creation

We used Blender to create all models. Some models were downloaded from blendswap.com. All downloaded models are CC-Zero.

Libraries

The following libraries were used:

- Assimp
- Glew
- Glfw
- FreeImage
- Freetype
- Physx