*CGUE SS2014* *2nd Submission*

# Wilderock Asylum

## Group Members

- Felix Hochgruber, 1226656, 532
- Jonas Engl, 1226658, 532

## Description of the Implementation

### Gameplay

The final game feauteres now a big level, which consists of different rooms and corridors. You control a police officer in first-person view. The only weapon you have is a torch and a camera, but that isn't enough for the thing that is haunting you.

The goal is to photograph every piece of evidence you find in the said level. Every piece is marked, so the player can identify easily IF he founds it. The thing which haunts you can*t see anything, so it follows your trails, so don't look back!

The game ends when every item is found or when the monster catches you.

### Complex Objects / Animated Objects

We have many complex objects in our game, like the surgical lamp, the moosehead or the skeleton. We added one animated object, which is the monster. It is animated using full skeletal animations and moves around, trying to catch you.

### Transparency

We discussed a lot, where to use transparency and decided to use it for the help/win/loose screen. The help screen is shown when pressing F1, but the game doesn' stop, so you have to keep moving (=> impossible without transparency). It is also applied to the win/loose screen, so you can catch a view of the monster when you die ;)

### Experimenting with OpenGL

We experimented with OpenGL, based on the needed functions. For example you can turn on/off Mip Mapping with F5 to see the changes live in the game. You can also toggle the texture-sampling-quality, turning transparency on and off, etc.

Experimenting with OpenGl showed as what it is really capable of and how skillfull it is.

### Lighting and materials

As an alternative to adding multiple light sources to our game, we decided to go for light mapping. In addition to that, the scene is lit by a dynamic light source, which is the flashlight. The lighting calculation is done in the fragment-stage of the shader. The shader is able to handle materials, which are read from the model file. Not just every object, but every mesh can have a different material and texture. If a material attribute is not set, a default value is used.

### Controls

These are the currently implemented controls. They are implemented with polling.

| | |
|---|---|
| W,A,S,D | Control player character |
| Mouse | Look around |
| RMB | Look through viewfinder |
| LMB | Detect if an object of interest is photographed |
| P | Save screenshot |
| Space | Flashlight on/off |
| Shift | Run |

## Features

- Multiple models loaded from different file-formats
- Textures and materials for each mesh
- 3D Picking by offscreen-rendering and color detection, to check if you photographed a piece of evidence
- Monster, which follows your path
- Saving screenshots
- Normal Mapping to get a more realistic feeling
- Light Mapping
- Motion Blur while running (can be turned on and off)
- full skeletal animations
- Background sound, Sound for walking/running and when you're exhausted
- Freely movable torch, to explore the dark parts of the level

## Effects

### Motion Blur

For the motion blur i followed this tutorial:

http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html

I started by passing my movement vector to the fragment-shader, to know in which direction i should blur. After that, I can sum up a fixed number of neighbour pixels and divide it by the same number again. This needs to be done, to get the blurry effect.

Instead of the depth buffer I calculated, with help from the normals, if a wall/object is in front of me or on the side. I did this because it looked wrong, when the wall ahed of me was blurred. With this technique only the walls/things on my left/right side get blurred.

### Animations

We implemented full skeletal animations in our game. The bone- and weights-data is directly imported from the model file, using AssImp's animation loading capabilities. They are then stored in an internal structure. The bone-transform-matrices are calculated on each draw step with the help of an external class. The whole matrix array is then pushed to the shader, where the corresponding matrices are multiplied with the model matrix.

The external classes were obtained from this website: http://nolimitsdesigns.com/game-design/open-asset-import-library-animation-loader/

### Light Mapping

The light maps were generated in Cinema 4D by using the “Bake Object” function. They are loaded like a texture in the game and passed to the shader before drawing the mesh. If a material does not have a lightmap assigned to it, a default lightmap (1x1 pixel black image) is used.

### Normal Mapping

Because flat surfaces do not look very realistic if an old wall texture is applied to them, we decided to implement normal mapping to create the impression of depth. Almost every mesh in our game has a nomal map assigned to it. Since Collada 1.4 (we decided to go for this format for our models) doesn't support normal maps, they are saved as a reflection-map in the model file. The normal maps are, like the light maps, loaded as textures and passed to the shader. The tangents and bitangents needed for the calculation in the shader are generated by AssImp and they are bound to a VBO.

The implementation in the shader mostly relies on this tutorial: http://ogldev.atspace.co.uk/www/tutorial26/tutorial26.html

### Stencil Buffer Outlines

In order to make the pieces of evidence more visible to the player, we decided to create outlines around the objects. For that purpose we firstly rendered the object in a distinctive color slightly bigger to a stencil buffer, and then rendered the object regularly on top of that.

## External Libraries

- **glew32**
- **glfw3**
- **SOIL**: for image loading

  http://www.lonesock.net/soil.html
- **Assimp**: for model and material loading

  http://assimp.sourceforge.net/
- **GLM**: for math stuff

  http://glm.g-truc.net/0.9.5/index.html
- **Bullet Phyics:** for Collision Detection and Physics (http://bulletphysics.org/wordpress/)
- **FMOD:** for the sound (http://www.fmod.org/)

## Tools used for content creation

- Cinema 4D for creating the level and 3D models
- Adobe Photoshop for editing textures
- SS Bump Generator for generating normal maps