Illuminator

Implementation

Free movable camera

Every frame the camera is updated from keyboard and mouse and creates a view matrix. In normal camera mode player movement follows rules of gravity (camera attached to rigid body – Bullet Physics) and collides with other objects. In Ghost Mode (toggle with [G]) gravity and collisions are disabled. View frustum culling is already implemented.

Moving objects

The position of the All-seeing Eyes and Lasers are updated every frame. In addition the view direction of All-seeing Eyes will match their movement direction after a few frames.

Texture Mapping

An All-seeing Eye contains two textures, one for the eye and one for the stones. The level mesh uses 3 textures. One for the floor, one for the ceiling and one for the walls.

Simple lighting and materials

Currently there is only one light source. A point light positioned in the middle of the room. Each object has a material that influences its shininess. Normals are also included, but are only rendered flat at the moment.

Controls

WSAD – move

Mouse movement – look around

Space – jump/Ghost mode: move up

Left Control – Ghost mode: move down

Left Click – shoot laser

Right Click and hold – grappling hook (at the moment: flying)

E – Open doors

R – Reset Game

ESC – End game

1 – Deactivate Enemy Movement

2 – Disable Normal Mapping

G – toggle Ghost mode

F2 – Statistics on/off

F3 – Wire Frame on/off

F4 – Textur-Sampling-Quality: Nearest Neighbor/Bilinear

F5 – Mip Mapping-Quality: Off/Nearest Neighbor/Linear

F8 – Viewfrustum-Culling on/off

F9 – Transparency on/off

Gameplay

Shoot an All-Seeing Eye to convert it (make it shoot other All-Seeing Eyes). Converted All-Seeing Eyes also convert enemies they hit. Currently you only have 10 shots (except for when you enable the "rapid fire"-power-up. If an converted All-Seeing Eye is hit 3 times it vanishes. If you get hit 3 times you lose. There is no win condition at the moment.

Effects

Shadow Mapping with PCF

Shadow Mapping was implemented for directional perspective light (a spotlight). It uses a perspective projection matrix based on the spotlight's angle and the maximum shadow distance. With my implementation the light could easily be moved every frame, but we do not use this feature in our game. I used the tutorials by ThinMatrix (https://www.youtube.com/channel/UCUkRj4qoT1bsWpE_C8lZYoQ) as a reference.

Normal Mapping

Normal mapping was implemented by using normal maps from Total Textures. Lighting calculation was changed from world space to eye space. Additionally some calculations are performed in tangent space. I used the tutorials by ThinMatrix (https://www.youtube.com/channel/UCUkRj4qoT1bsWpE_C8lZYoQ) as a reference.

Bloom

Bloom was implemented by using an additional render target. Our fragment shader renders two textures into a frame buffer object, one containing the rendered scene and one containing all the pixels, which brightness exceeds a certain threshold. The texture containing all the bright values is then blurred using a 2-pass gaussian blur, implemented with two additional shaders. The original scene texture and the blurred brightness texture are then blended together to create a glowing effect for all bright objects in the game (lasers, shining enemies after death, ceiling on top floor of level). For the implementation I used the following tutorial:

http://learnopengl.com/#!Advanced-Lighting/Bloom

Deferred Shading

Deferred Shading was implemented using Multi-Render-Targets in the first step. I discarded the original renderer we used, which was a forward renderer and split the rendering process into the scenes geometry and the lighting (geometry pass + lighting pass). With the scenes geometry we rendered multiple textures into a FBO, called the gBuffer, containing amongst others, a position texture, a normal texture and a diffuse/spec texture. Further textures needed to be used for normal mapping and shadow mapping to work together with the new rendering pipeline. With these textures, which we acquired in the previous step, called the geometry pass, we then took in our light sources and in this last step, called the lighting pass, calculated the final result. By postponing the lighting calculations to the end of the pipeline, the calculations aren't performed for each object of the scene, but rather for each pixel of the texture, which is then rendered onto the screen (post processing), which improves the performance of the game by a lot, when using a high amount of light sources. I used the following tutorials for the implementation:

http://www.codinglabs.net/tutorial_simple_def_rendering.aspx

http://ogldev.atspace.co.uk/www/tutorial35/tutorial35.html

http://learnopengl.com/#!Advanced-Lighting/Deferred-Shading

Special Features

Powerups

The game features a rapid fire, a hyper jump, a shield and a GODMODE powerup. (The power ups with the models are existing, however they aren't part of the level yet and can't be collected)

Ammo

The game features an ammo system. You have 10 shots, as displayed in the bottom right corner of the game screen. Once you are out of ammo, you can't shoot anymore.

Material changing of converted enemies

If you convert a pyramid, by shooting it with your lasers, it will start to shine and start firing at other enemies alongside you.

Ghost Mode

If you want to explore our amazing level without being shot be the pyramids, use G to toggle between the ghost mode, which will make you invincible and also allows you to fly and go through walls.

Modeling Tool

Blender

Complex interaction sequences

Press E to open the door.

Features

- OBJ loader
- View frustum Culling
- Physics (Collision, Gravity)
- Ground contact check with Raycasting (implemented with Bullet Physics)
- Font Rendering with texture atlas (BMFont format)
- All-Seeing Eyes shoot at closest hostile target
- Rendering images to HUD
- Shadow Mapping with PCF
- Normal Mapping
- Deferred Shading
- Bloom

Illumination and texturing

There is one red spotlight that is used for shadow mapping. The All-Seeing Eyes and the level cast shadows. Everything receives shadows. Additionally there is one point light on the top floor that is used to light up the whole game a little. Since the point light is not used for shadow mapping walls don't stop it.

An All-seeing Eye contains two textures, one for the eye and one for the stones. The level mesh uses 3 textures. One for the floor, one for the ceiling and one for the walls. One texture is used for the doors and one for the gears.

Libraries

FreeImage (http://freeimage.sourceforge.net/)

Assimp (http://www.assimp.org/)

GLEW (http://glew.sourceforge.net/)

GLFW (http://www.glfw.org/)

GLM (http://glm.g-truc.net/0.9.7/index.html)

Bullet Physics (http://bulletphysics.org/wordpress/)