Gravity Documentation

Controls

Implemented Controls are so far:

Key	Effect
W,A,S,D	Steer character
F	Change gravity to nearest axis
ESC	Quit game

Features

- OOP Classes for shader, camera (player), texture, face, mesh, material, text
- Using image loader devil
- Using model loader assimp
- Using Nvidia PhysX engine
- Basic shading, uv texture mapping and lighting with one material and one light source
- Vao, Vbo, Ubo, display wireframe, texture sampling, mip mapping, frustum culling, transparency
- Complex objects, animated objects

Gameplay

You can move around and reach the end of the level but if you get hit by a spiked sphere or hit the spiked wall you will die. You are able to change the gravity by looking into a direction and pressing f. The gravity will change to the nearest axis of your looking direction. Through that you are able to reach the end of the level, when you reach the end of the level a finished message appears.

HINT: Blender is buggy we changed the finish spot but it won't export it correctly so the finish spot is just a small square on the wall.

Complex Objects

There are several complex objects in the game the whole level is a complex object and we have integrated several other objects like spiked spheres and spiked walls. All those object models will be loaded via assimp model loader.

Animated Objects

We have placed a hierarchical animation right up to the player starting position. There are 2 sphere like shapes where one object is moving in a cirle and the other objects is moving around the first one in a circle.

View-Frustum-Culling

We have implemented the View-Frustum-Culling like the described here

http://www.racer.nl/reference/vfc_markmorley.htm

During the initialization we calculate the maximum length of a vertex from the model center. Afterwards we are using that radius to determine if an object is visible or not. If it is not visible it won't be drawn.

Transparency

We are using a global alpha channel for all objects which can be toggled. If transparency is on everything will be drawn half transparent.

Experimenting with OpenGL

We used VBOs, VAOs and UBOs in our game, we implemented texture mip mapping and texture sampling. Furthermore we implemented the demand key-mapping to toggle effects.

Effects

Because we didn't had the time we did not implement any special effects like particle systems and shadow mapping.

Tools

We used blender to create the models and gimp to create the textures.