RIBENJI

GAMEPLAY COMPULSORY:

3D Geometry:

Our game consists out of several different interesting shapes. All of which were created in Blender and are imported with ASSIMP.

Playable:

In the quest to redeem the character it is possible to move around the world with the help of teleportation trying to avoid the enemies. The game can start from a compiled .exe file.

Advanced Gameplay:

The game logic has been finalized, with ray-casted teleportation, multiple levels, and a boss fight.

Min. 60 FPS and Framerate Independence:

The game runs with a cap of 60 FPS with the help of measuring the time between renders and adapting the wait time.

Win/Lose Condition:

The set goal of the game is to redeem the character with the help of his new earned power. This is achieved by assassinating the Yakuza boss. This is no easy job as he has lots of followers which are trying to stop you.

Intuitive Controls:

Controls were kept very simple and similar to other games. Moving around is done with "**WASD**" and using the teleportation ability with "**Right Mouse button**". It is possible to hold down the Movement buttons and keep moving.

Intuitive Camera:

For our game we chose a fixed camera which is locked in a direction looking at the player. It is possible to toggle a freelook camera with "**F1**" for debug purposes.

Illumination Model:

Our Illumination Model uses one or more light sources, the main one being a directional light. Each object uses a Texture to define its diffuse color and includes normal to calculate the strength of the light.

Textures:

All textures are loaded alongside our models with the selected import library which includes mipmapping and trilinear filtering for textures.

Moving Objects:

There are several moving objects specifically the enemies around the game world move for a more dynamic feeling.

Adjustable Parameters:

It is possible to change the selected parameters in the given "**settings.ini**" file which is located in the "**assets**" folder. It is also possible to toggle Fullscreen mode in game with "**F**".

GAMEPLAY OPTIONAL:

Collision Detection (Basic Physics):

The whole game is built upon Bullet for most of the collision detection.

Advanced Physics:

For the teleportation we implemented a ray-cast which goes towards the cursor (shown as a green line) and makes a ray-cast-collision callback with bullet to determine what objects the green line collides with. After that we take the closest object (the first object the green line collides with), which will teleport the player to the specific position of that object.

Heads-Up Display:

Our HUD is made of complex shapes which then have to be blended to remove unneeded background color.

FEATURES:

Movement:

Direction:
The player can move around the game world freely using the "WASD" buttons
Teleportation:
To aid the player he can also teleport to an indicated location of the ray via the "**Right Mouse button**". Beware as he can only teleport to higher ground platforms to make the game harder. The green line indicates the teleportation spot. The first object that the green line intersects with (going from top to bottom) is the position the player will teleport to.

Enemies:

There are several enemies around all stages which must be avoided. Getting too close might end up in failing to take revenge.

Stages:

On the quest to vengeance, the player must clear stages to get to the final boss and take revenge.

Boss:

In the final stage the boss must be fought, by trying to out-maneuver him and bringing him to his fall.

EFFECTS:

Shadow Map with PCF (Lighting):

Our game is filled with shadow using a shadow with PCF. For the light source we decided on using the main directional light. Helping us achieve the outcome we mainly used a tutorial from YouTube (https://www.youtube.com/watch?v=Ut6poChkSjA), which is based on the tutorial from LearnOpenGL (https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping). We've added the ability to show the rendered shadow map with "**J**", which is used to calculate if something lies in the shadow.

Environment Map (Texturing):

For the Environment Mapping, we again followed the YouTube tutorial (https://www.youtube.com/watch?v=QYvi1akO_Po), which is also based from a tutorial from LearnOpenGL (https://learnopengl.com/Advanced-OpenGL/Cubemaps). We first created a custom scene inside of Blender and exported and created a Skybox from that. Then we implemented the Cubemap itself into our game, to show the Skybox. The final step was choosing which surfaces we wanted to be reflective and chose the water. Which then reflected the Skybox in combination with light calculation and its diffuse color. We added a small change, so that it also would have a refract effect. This was based on another tutorial (https://www.youtube.com/watch?v=xutvBtrG23A).

Cel Shading (Shading):

The game implements in its entirety Cel Shading. We simply followed a tutorial from YouTube (https://www.youtube.com/watch?v=dzItGHyteng), which explains very well how the color gradients are calculated. One important thing was that wie had to setup our scene properly, so that a normal vector between faces was averaged, especially on rounder object, to really show its effect.

Bloom/Glow (Post Processing):

For Bloom it was very important to first have a way to add Post Processing effect. This was done by following a YouTube tutorial (https://www.youtube.com/watch?v=QQ3jr-9Rc1o). It was very well explained how Post Processing works by making a "Snapshot-Texture" of the rendered scene and using it as a texture for a quad which renders across the screen. To implement bloom itself we followed this tutorial (https://www.youtube.com/watch?v=um9iCPUGyU4), which explained well how bloom is achieved by adding a blurred image, which only shows bright spots, to the quad. This effect can be toggled by pressing "**B**".

ADDITIONAL LIBRARIES:

Glfw:

Already included for this project. Used for creating windows, contexts and surfaces, receiving inputs and events. (https://www.glfw.org)

Glew:

Already included for this project. GLEW provides run-time mechanisms for determining which OPENGL extensions are supported on the target platform. (http://glew.sourceforge.net/)

ECG_LIBRARY:

Already included for this project.

Assimp for model loading (level, player, objects):

https://github.com/assimp/assimp

Freetype for 2d text rendering:

https://freetype.org

Bullet for collision detection and physics:

https://pybullet.org/wordpress

THE GUIDE TO REVENGE:

Goal:

The main goal is to travel through all the stages by going through the Torii Gate at the end. In the last stage the boss needs to be defeated, by making him to run off the stage.

Stage 1:

The easiest way is to wait for the enemy to get to the tree and teleport on the bridge. To the right is a platform which can be teleported on. From there drop down to the gate and try to enter it. Beware of enemy.

Stage 2:

To the right are gate which can be teleported on top of. At the last one wait for the enemy near the boxes to pass and run upwards. Wait behind the rocks to run across to the other rock when the enemies part. Run upwards again and beware of the quick one and try to stay close to the edge to enter the Torii Gate.

Stage 3:

Immediately drop down to the ledge, cross the plank, run back, cross the other plank and teleport up to the stone circle. There teleport on to the back wall and go to the right to the planks and drop down at the end of the planks. Then teleport on top of the stacked rocks and onto the gate. Drop down behind the bushes and teleport onto the platform. There teleport again and enter the boss arena through the Torii gate. To kill the boss, go to the edge and get out of the way as he runs towards you. Doing this three times will kill the boss and you can leave through the Torii Gate.