Victoria Sofie Fischer (00947837), Maria Kirschenmann (01027135)

Temple Raid – Documentation

Genre: Adventure Game (Maze game)

Goal: Find and collect all the items in the labyrinth to get out of the maze - but don‘t get caught by the ancient ghosts!

Story: Marcy Field is an archaeologist and on a field trip in Mexico to explore an old Mayan temple of an ancient goddess. During the exploration of the second chamber in the temple Marcys colleague accidentally stepped on a loose stone on the floor - suddenly a secret trapdoor opened and Marcy fell into the deep. When she woke up, she found herself in a dark and gloomy place - on each side of her are dimly lit corridors and on the wall in front of her are some very old Mayan hieroglyphs, she is able to read, because for years and years of studying the Mayan culture: „The only way to get out of this maze is to collect the holy items of the goddess herself - but beware of unwelcoming, hostile creatures!“

Gameplay: Marcy is able to run around in the maze. Some of the collectable items are there to support Marcy (for instance a map that shows her the maze as seen from above to make orientation easier). There are also dangerous things lurking in the maze. Ancient ghosts are moving through, temporarily blocking corridors. Colliding with them will cost Marcy life energy. Also some poisonous water areas make moving through the maze harder and must be jumped over, otherwise her life energy will be further reduced. If there isn’t any life left or Marcy tries to get out of the maze through the fire of doom without having collected all the items the game is over and the user lost. Once Marcy has collected all the holy items of the goddess the fire of doom disappears and Marcy is able to get out of the maze - in this event Marcy is free and the user wins.

Implementation

Character Control: The character control is implemented in “Marcy.h” according to this tutorial (https://www.youtube.com/watch?v=d-kuzyCkjoQ -last visited: 23.04.2022). The following keys move the character in the following way:

- **W**: move forward
- **D**: move to the right
- **S**: move backwards
- **A**: move to the left
- **SPACE**: jump

Additionally we have PhysX’ Kinematic Character Control. This allows us to handle collision detection with the maze walls as well as with the ghosts and poisonous waters.

Game Camera: In “GameCamera.h” the implementation of the main camera can be found (based on this tutorial: https://www.youtube.com/watch?v=PoxDDZmctnU&list=PLRIWtICgwaX0u7Rf9zkZhLoLuZVfUksDP&index=20 - last visited: 23.04.2022). The camera is controlled with the mouse:

- Left mouse button: for the yaw of the camera
- Right mouse button: for the pitch of the camera
- Mouse wheel: to zoom in and out

The camera follows the character around. We decided for a third person view. We still have the original camera from the framework but it can not be accessed when playing the game as this would make it very easy to cheat. We have used it for debugging purposes and it can be activated by either setting the “debug_mode” in our “settings.ini” to true or setting the “_debugCamMode” in the code to true after line 265.

Ghosts: Implemented in “Ghost.cpp” and “Ghost.h” were modelled in Blender with the help of this tutorial: https://www.youtube.com/watch?v=H7-OI7YM2D0 - last visited: 10.06.2022. They move between their minimum and maximum positions. Additionally a blue point light moves along with them. To make them glow

more, we implemented the ***Bloom-Effect***. The necessary framebuffers are defined in main. The "blur" and "bloom_final"-shader programs also belong to the implementation of the effect as well as the additional out-vec4 BrightColor in each shader used. The Bloom-Effect was implemented following this tutorial: https://learnopengl.com/Advanced-Lighting/Bloom - last visited 10.06.2022.

Poison Waters: The poisonous waters are implemented in "PoisonWater.cpp" and "PoisonWater.h". To animate the water the ***Vertex Shader Animation*** has already been integrated in its vertex shader "poisonWater.vert". To calculate the normal vectors the cross product of the partial derivatives in the x- and z-direction was used. To get waves in both directions the function used is:

f(y) = amp * sin((time + x)*freq) * sin((time + z)*freq)

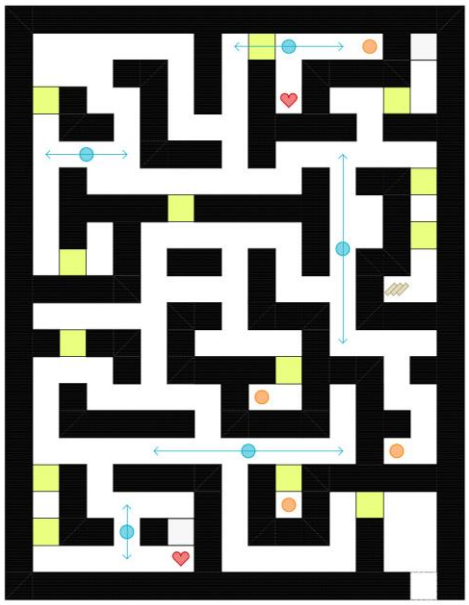
amp...height of the wave

freq...frequency of the waves

time...time passed so far in the game (what keeps the water moving)

LifeBonus: The implementation of the life boni can be found in "LifeBonus.cpp" and "LifeBonus.h". The heart-shaped model was created in Blender with the help of this tutorial: https://www.youtube.com/watch?v=TGpxr52jYHk - last visited: 25.04.2022. The hearts just rotate around themselves. When collected by the player they raise the life energy by 0.5f.

Map: The implementation of the map can be found in "Map.cpp" and "Map.h". The script roll model was created in Blender with the help of this tutorial: https://www.youtube.com/watch?v=I0aUMdrHqKI - last visited: 25.04.2022. The map also just rotates around itself. When collected by the player it shows the plan of the maze as ***HUD***. Once collected, the HUD of the map can be toggled by pressing "M" so that it does not occlude the players view.



The game starts at the top left corner. The exit is at the bottom right of the maze. The poisonous waters are the green rectangles. The ghosts and their movements are depicted with arrows and blue circles. The life boni are represented by the red hearts. The player must collect the holy items marked by orange circles. The map is the beige zig-zag-shape in the mid-right section of the maze.

Holy Items: Implemented in "holyItem.cpp" and "holyItem.h" use different imported models:

- A skull, which was downloaded from a webpage with free 3D models: https://sketchfab.com/3d-models/aztec-skull-6f7af40bd4a74066814d9b94337684d3 – last visited 10.06.2022
- A vase, that was modelled in Blender
- Coins, which were also modelled in Blender with the help of this tutorial: https://www.youtube.com/watch?v=r8ltW7pAN6M – last visited: 10.06.2022
- Gem stones, that were also modelled in Blender

Each holy item has an orange point light surrounding it, that is turned off once the item is collected.

Fire of Doom: Implemented in "FireOfDoom.cpp" and "FireOfDoom.h" is placed in front of the exit of the maze and is already implemented as ***CPU Particle System*** **using instancing**. It was written with the help of the tutorials mentioned in the Tuwel section "Effects" of this course (http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/ - last visited 25.04.2022;

https://learnopengl.com/In-Practice/2D-Game/Particles - last visited 25.04.2022). The look of it was improved by using a different shape for the individual particles and adding a color gradient (from white to yellow to red) to make it look like fire. Also the Bloom-Effect comes in handy here as the very bright particles glow.

Shadow Map with PCF: In order to improve the overall look of the maze, we decided to add shadows to our scene by implementing Shadow Map with PCF following these tutorials: https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping - last visited: 10.06.2022, http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/ - last visited: 0.06.2022. We chose to apply the "stratified Poisson sampling" as shown in the second link to take care of the edge depiction. The framebuffer for the depthmap is defined in main.

Simple Normal Mapping: This was implemented for the maze to give the walls more structure. With Photoshop a normal map texture was created out of the texture for the maze. Based on the theory explained in the tutorial (= https://learnopengl.com/Advanced-Lighting/Normal-Mapping - last visited 10.06.2022) the effect was implemented in the "mazeNormal"-Shader-program. It can be toggled by pressing "N". Because the maze only possesses normals that are axis aligned, each normal is tested with the help of the dot product in the fragment shader (in main) to find out with which axis and in which direction (positive or negative) it is aligned. The vector derived from the RGB-values from the normal map is then rotated to match the direction of the normal and finally transformed to the range [-1, 1] and normalized.

Environment Map: For the Skybox we used images from https://drive.google.com/drive/folders/1iCXmHX45OcEXBHCOw2mwliq5vMmHCSgR - but because we needed a dark cave or tomb, we edited the image of a small cave. For the implementation we followed learnopengl.com (https://learnopengl.com/Advanced-OpenGL/Cubemaps). We use the skybox for the game loop itself (so that the sky is not only black), and also for the win and loose screen. On the win/loose screen we also implemented reflection for the object.

HUD: For the HUD we used a simple text rendering – followed the tutorial of learnopengl.com (https://learnopengl.com/In-Practice/Text-Rendering) with the google font "Inika". Additionally we added a simple life/health bar to see the players health. When the player looses life the quad gets smaller – and turns red when only 30% health is left.

A special item the player can collect, is the map. When the player collected the Map, "Map" is displayed on the right side of the screen – and the player can see the map by pressing the key "M" or hide it when the key is pressed again. For the Map we render a quad and give it a .png file as texture (with transparency – so the user still sees whats going on in the maze). Pressing "H" will show a text describing the buttons that can be pressed to control the character.

Game Logic: In "GameLogic.h" the game's logic is handled. This is where the life of the player is minimized when it has the same position as a ghost or a poisonous water area. Also the condition for a lost or won game is taken care of in this class. The player wins if its life value is above 0.0, all the holy items were collected and its position equals the one from the exit. The player looses if its life is under 0.0 due to being hurt by ghosts, stepping into poisonous waters or instantly dying because of getting in contact with the fire of doom before all the items were collected.

Winner/Loser: The game is won if Marcy collected all holy items an exits the maze. In this case the user sees the text "YOU WON" in the window, which was created with Blender.
On the other side if the user lost (lifebar is empty or user walked into the fire) the text "YOU LOST" appears in the window - also created with Blender. For these screens we have implemented a really simple camera (EndCamera.h)

In order to restart the game "Y" should be pressed (if you do however have an english keyboard, you must press "Z").

Additional Libraries used apart from those already included in the OpenGL framework

Assimp: https://github.com/assimp/assimp/releases - last visited 25.04.2022

Is used to import our ".obj"-models such as the maze (which was modelled in ArchiCad 24, then exported as obj and adjusted in Blender in order to be exported again as .obj and imported into our game). In order to translate it into an object that can be rendered the tutorial for "Model Loading" described in learnopengl.com was followed (https://learnopengl.com/Model-Loading/Assimp - last visited 25.04.2022).

STB: https://github.com/nothings/stb/blob/master/stb_image.h - last visited 25.04.2022

Used to import jpg-images for the textures of the imported models.

PhysX 4.1: https://github.com/NVIDIAGameWorks/PhysX - last visited 25.04.2022

This library is used for the collision detection of the character with ghosts and the poisonous waters as well as for the interaction of the character with the physical world of the maze (not being able to go through walls or jump on top of the walls of the maze).

IrrKlang: https://www.ambiera.com/irrklang/downloads.html - last visited 31.05.2022

For the sound effects and the background music we used the irrklang library, to start playing .wav files. The background music is about 3 min and plays in loop. Additionally there are sound effects for collecting the holy items, the map or the hearts.

The music and the soundeffects we created on our own with "Apple Garageband".

FreeType: https://freetype.org/download.html - last visited 25.05.2022

For loading a font we used the library freetype, which was recommended by the learnopengl.com tutorial.