Echoes of the Labyrinth

Brief description of implementation

Dive into an exciting adventure in the mysterious labyrinth! As the game begins, all models are imported and rendered, placing the player at the center of the maze. Use the WASD keys to move and the SPACE key to jump. You can adjust the camera with the mouse and zoom in and out using the scroll wheel.

Your mission: Collect the glowing diamond, which shines brilliantly and is surrounded by dangerous lava. Beware, a single misstep can make you fall into the lava, causing you to respawn. To reach the diamond, you must first find keys hidden throughout the labyrinth.

As you navigate the labyrinth, you can keep track of your progress on the screen. The number of collected keys, your remaining lives, and the current FPS are all displayed for your convenience.

Once you have collected at least four keys, a bridge over the lava will appear, providing a safe path to the diamond. Cross the bridge, grab the diamond, and win the game!

The game is developed with a focus on immersive gameplay and advanced computer graphics techniques. The development environment utilizes various libraries to meet the requirements and enhance the game's features. Below is a detailed description of the implementation of different aspects of the game:

Features and implementation details

Gameplay:

- 3D Geometry: Objects are imported with Assimp.
- Playable: You can move the Player with **WASD and SPACE** and we have an exe to start the game.
- Min. 60 FPS and Framerate Independence: The Player and the Camera move independently and the game runs at a constant 60 fps in 1920x1080 resolution. We also added an FPS- Counter to the HUD that averages the FPS that were achieved in the last 3 seconds and displays them in the bottom left corner.
- Win/Lose Condition: If you collected 4 keys at least the bridge will spawn, allowing you to grab the diamond. If you fall into the lava, you will be sent back to the spawn.
- Intuitive Controls: Character Controls with **WASD, SPACE and the mouse.**
- Intuitive Camera: controlled via mouse.
- Illumination Model: We used a mix of a few Lighting models, most objects use Phong shading with a vector that changes the material properties, some objects use Physically based Shading, and some Objects use a blur shader to simulate bloom. Normal vectors are also provided **(you can toggle them with N).**
- Textures: Imported Models all have Textures. The torch and the skybox **(toggle with T).**
- Moving Objects: The Player can be moved, and the torch is moved with it.
- Documentation: see here.
- Adjustable Parameters: you can change them in windows.ini in assets/settings.

Optional Gameplay:

- Collision Detection (Basic Physics): We used PhysX to stop the player from traversing the walls.
- Heads-Up Display: We added a HUD where you can see your Health, the number of keys you have found, a timer which shows you how long you have been in game and an FPS- Counter **(toggle with H).**
- Camera Object Tracking: We track our Character with our Camera and update the direction the character will walk to based on the camera direction. The rotation of the camera is also updated using the camera direction, so the character is always facing in the same direction as the camera.

Effects:

Lighting:

- Shadow Map with PCF: we included a Shadow map for a directional light source which is computed and added to the overall scene color at runtime.

Animation:

- GPU Vertex Skinning: We added skeletal animation to allow our character to switch between different animations based on the keyboard input **(toggled with WASD)**

Texturing:

- Environment Map: the map, Player, pedestal, bridge and floor are affected by the skybox. Also, the pbs- Demo uses reflections on the keys.

Shading:

- Physically Based Shading: the pedestal and 2 keys for demo **(toggle with P)** are rendered with pbs. Also, the pedestal is rendered using pbs.

Post Processing:

- Bloom/Glow: keys, lava and the fire of the torch are rendered with bloom **(toggle with B)**

Special Features

- HDR with exposure tone mapping: HDR is enabled for all objects in our scene **(can be toggled with R).**
  You can change the exposure with the **keys up (exposure increases) and down (exposure decreases)**
- Gamma correction: **toggle with G**
- Skybox
- Wireframe Mode: **toggle with F1**
- Culling: **toggle with F2**
- Close the window with escape key
- You can update Player Health with the **J and K key** to see the health increase/ decrease on the HUD.

Used libraries

Following libraries are used:

- Nvidia PhysX: Used for collision detection between static actors and the player controller. https://nvidia-omniverse.github.io/PhysX/physx/5.4.0/
- Assimp: Used for 3D model importing. https://github.com/assimp/assimp
- GLFW: https://github.com/glfw/glfw

- GLM: https://github.com/g-truc/glm
- GLI: https://github.com/g-truc/gli
- FreeType: https://freetype.org/
- ImGUI: https://github.com/ocornut/imgui

Additional Information

To achieve the various effects in our game, we used multiple resources and tools. For model importing and many of our effects, we used code from LearnOpenGL, which we then customized to fit into the implementation.

We also benefited greatly from educational YouTube videos by creators such as Brian Will, Victor Gordan, and OGLDEV, which provided in-depth tutorials and insights into OpenGL programming.

Our models were primarily created using Blender. For the player model we downloaded it from a website called BlendSwap (https://www.blendswap.com/blend/7233). For physics simulations, we relied on the PhysX documentation to guide our implementation.

Here are the tutorials we used:

- Model loading: https://learnopengl.com/Model-Loading/Model
- Vertex Skinning: https://www.youtube.com/watch?v=r6Yv_mh79PI (Part 1-5), https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation
- Shadow Map: https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping
- Bloom: https://learnopengl.com/Advanced-Lighting/Bloom
- Physically based Shading: https://learnopengl.com/PBR/Lighting
- HUD: We referred to the Dear ImgUi GitHub- Page and the docs, https://github.com/ocornut/imgui

Walk-through:

The game starts with the player in the middle of a labyrinth, using WASD keys for movement and SPACE for jumping. The player navigates the maze, finding and collecting keys, solving puzzles to unlock doors, and avoiding lava to reach the glowing diamond. With at least four keys, a bridge appears over the lava, allowing the player to safely collect the diamond and win the game. By pressing F6 you can enable flight and collect the keys faster. You can fly by holding down the Spacebar.