Documentation

Group

Csaba, 12122549
David Bauer, 12120495

Description

First person movement shooter where you try to survive for 5 minutes. The focus is quick and constant movement to dodge, while trying to shoot the enemies. Perks, multiple weapons and multiple enemies were planned but not implemented as focus was put on finishing the effects and required tasks. Additionally the map and the objects are unfortunately somewhat bland as we did not get around to texturing everything.

Controls

WASD - Movement
Space - Jump
Left Click - Shoot
F1 - Toggle fullscreen
F2 - Toggle debug collisions
F3 - Toggle HUD
ESC - Close game

Gameplay

You spawn on a map and get little time to orient yourself before the first enemies start spawning in. Enemies have to be shot while avoiding collision with them, as they deal damage to the player. Some of them are faster than you, so simply running away will not be enough, at some point you will have to dodge (jumping over them or using the map to your advantage). Picking up green orbs will heal you for 10 health.

Per default these enemies have around 100 health. You can see your weapon stats on the bottom right.

Note: The gun has a short cooldown and does not have any feedback when it is shot, so it may feel unresponsive.

3D Geometry

All our objects are loaded from external .obj files including normals and uvs.

Playable

The player can move around and interact with the game world.

Advanced Gameplay

The player can shoot enemies, who will chase the player. Both the player and enemies have health and can be killed. Enemies spawn in regular intervals near the player and the player has to shoot them to avoid losing. There are pickups that can be collected. Additionally the movement allows the player to build up speed as your movement speed is higher while in the air.

Min. 60 FPS and Framerate Independence

All our update methods for physics run on a fixed timestep. See: src/gameWorld

Win/Lose Condition

The game consists of a timer that ticks down. If the player survives for that time, they win, if they die during that time, they lose. After that the game starts again. See: src/gameplay/gameLoop

Intuitive Controls

The controls are the industry standard for an fps game (wasd, mouse, left click). They also get shown to the player before the game starts The mouse movement speed as well as the player speed and drag can be adjusted via a settings file. See: settings.toml

Intuitive Camera

The camera can be controlled by moving the mouse. The mouse sensitivity can be configured in the config. See: src/game-objects/camera

Illumination Model

We have multiple models, multiple lights and multiple materials, which all have their own distinct values. Additionally lights get created and removed during runtime.

Textures

Every object has textures attached to it, ranging from a simple single color texture, to multiple textures for PBS.

Moving Objects

The enemies use physics to move towards the player. The player can fully control his own collider as well as the camera.

Adjustable Parameters

There exists a config.toml file in which some settings can be adjusted. See: src/asset-objects/configHandler

Collision Detection (Basic Physics)

We use the ReactPhysics3D framework to create a static environment for the player to move in. It is also used for player and enemy movement by applying the necessary physics forces. There exist collision categories which enable some object to pass through one-another.

Advanced Physics

Collision categories (src/game-objects/gameObject.h), trigger colliders, collision callbacks (src/gameplay/CoolEventListener.cpp) and raycasts (src/weapon/weapon.h) have been implemented.

Heads-Up Display

We have a HUD in the form of text elements. The elements change dynamically depending on the game state. See: src/asset-objects/textHandler

Effects

Vertex Shader Animation

Vertex shader animations are present on green orbs which are health pickups. They use a simple vertex shader that transforms the sphere based on the elapsed time. See: assets/shaders/anim.vert

Procedural Texture

The enemies have a procedural texture that consists of a gradient texture generated at runtime. The texture can be seen better when the enemies are standing still next to the player. See: src/rendering/materials/proceduralTexture

Physically Based Shading

We have implemented a standard PBR shader which includes an albedo map, normal map, roughness map, metallic map and ambient occlusion map. It is used for the gun that the player is holding. See: assets/shaders/pbs

GPU Particle System using Transform Feedback

Our particle system uses a call to update (vertex/geometry) and a second call to render (vertex/geometry/fragment) many particles using transform feedback. The particle effect can be seen when an enemy is killed. See: src/particleSystem

Resources

- All models/textures/fonts are freely available for personal use
- OpenGL
- GLEW
- ReactPhysics3D
- GLM
- StbImage
- TinyObjLoader
- TOML
- Freetype
- https://learnopengl.com/ (PBS, Text rendering)
- http://www.opengl-tutorial.org/
- https://ogldev.org/index.html (Particles)