

Submission 2 - CoD Covid Mode

Description of Implementation

The core architecture of the game is an “entity component system” to handle game logic and data. We have entities that can have multiple different components, which hold data. Then we have systems that iterate over all entities that match certain criteria and perform some action. For example, the RenderSystem fetches all entities containing a “Transform” and “Renderable” component and uses the data stored in those components to render those entities.

This allows us to create very flexible and decoupled code for different aspects of the game. For the implementation of the entity component system we use the library “entt” [1] which provides us the tools to create, manage and retrieve entities with our custom components.

Features

In Cough of Duty - Covid Mode you spawn in a beautiful world. But there is a twist. Suddenly you get approached by covidots from four different directions. You have two options to defend yourself: 1. Shoot with the crossbow (left click) 2. Throw the holy grenade aka the orange book (middle mouse button). You start with 100 health points and if you have 0 left you die and you have to restart the game by pressing “Enter”.

There are multiple random items that spawn after each wave:

- Shield (automatically equips itself for 30s) which gives you better damage resistance against the covidots
- The orange book (gives you one more ammo for the holy grenade).
- A Pfeiser vaccine (blue) which increases the damage you do
- An Austro Zeneca vaccine (orange) which increases your fire rate
- A Leberkasemmel, which heals you to 100 health
- A mask which blocks all damage for a short duration

Each wave the covidot count increases thus clearing waves becomes increasingly harder. The goal is to survive the longest you can.

Gameplay

Compulsory

3D Geometry

Geometry is created with blender and loaded in using assimp loader.

Min. 60 FPS and Framerate Independence

Our game runs framerate independent. This is done by multiplying everything that should happen at a fixed rate by deltaTime. The game runs at a minimum of 60 FPS as well.

Win/Lose Condition

You lose if your hp is ≤ 0 and because this is a never ending game the win condition is to survive the longest you can.

Intuitive Controls

Basic Movement - WASD

Look around - Mouse

Jump - Space

Shoot - Left click

Throw Orange Book - Hold and release middle mouse button

F1 - Toggle Wireframe Mode

F2 - Toggle Backface Culling

F3 - Show FPS

F4 - Toggle HUD

F5 - Toggle Spectator Mode (for debugging, unlimited Orange Books)

F6 - Toggle Normal Mapping

Intuitive Camera

We have a first person camera that is rotated by using mouse movement. We used the camera tutorial on learnopengl.com [4].

Illumination Model

We use the phong illumination model and have the sun as a single global directional light source. We store information about how an entity should be rendered in a material class, which sets properties like the color in the shader.

Textures

Diffuse Textures and normal maps are loaded from png files. Each model can have different Textures, which get resolved when importing an .fbx.

Moving Objects

The enemies are always moving towards the player. Items that spawn spin around the y-axis to signal that they are collectable. If you shoot or throw a grenade new physics entities will be instantiated.

Documentation

Most functions and classes have docstrings and difficult code lines are explained as comments.

Adjustable Parameters

The screen width and height, target framerate and fullscreen toggle can be controlled using a config file (settings.ini in Resources/).

Optional

Collision detection

As a physics engine we use Bullet Physics [2]. Each entity that should be affected by physics gets a RigidBody component. Our PhysicsSystem then creates a rigidbody in the bullet world for each RigidBody component. Each frame all entities with a RigidBody get their transforms updated from their equivalent rigidbody in the bullet physics world.

The player and enemies are moved using the physics engine so they can collide with the environment and are affected by gravity.

Advanced Physics

The player shoots projectiles that collide with the world and apply damage on collision with enemies. We also have a grenade that on explosion applies damage and knockback force to all enemies in range.

Enemies that die will still be present for a few seconds, so they will fly away when hit by an explosion or player projectiles.

HUD

We use the freetype library [3] to load bitmaps from font files to display the HUD. We used the tutorial on learnopengl.com [4].

The HUD shows all necessary gameplay information such as health, ammo, powerups and wave number, Orange Book ammo.

Effects

Environment map

We implemented a cubemap which can function as a skybox and use environment mapping for reflection and refraction. We used the learnopengl tutorial [4]. These reflections are visible on the crossbow.

Normal mapping

We implemented tangent space normal mapping with imported normals, tangents and bitangent from assimp loader. We used the tutorial on learnopengl [4].

Vertex skinning

We implemented vertex skinning by loading skeletal animation of a .fbx file and calculating bone transform matrices each frame. Together with a per vertex bone weight we calculate the final transformation in the vertex shader. We used the tutorial on ogldev [5]. Enemies, crossbow, book is animated.

References

- [1] <https://github.com/skypjack/entt>
- [2] <https://github.com/bulletphysics/bullet3>
- [3] <https://www.freetype.org/>
- [4] <https://learnopengl.com/>
- [5] <https://ogldev.org/>

Credits

Crossbow Model:

"Crossbow"

by Fanton

<https://sketchfab.com/3d-models/crossbow-6d2dc55ac3344c1a844b44a52ee6c4d4>

CC BY 4.0

Music:

Battle Metal by Alexander Nakarada | <https://www.serpentsoundstudios.com>

Music promoted by <https://www.chosic.com/>

Attribution 4.0 International (CC BY 4.0)

Image Karlskirche:

"Die Karlskirche in Wien im Abendlicht"

by Thomas Ledl

https://de.wikipedia.org/wiki/Wiener_Karlskirche#/media/Datei:Karlskirche_Abendsonne_1.jpg

CC BY-SA 4.0

Skybox:

Miramar skybox

by Jockum Skoglund (hipshot)

<https://opengameart.org/content/miramar-skybox>

CC BY 4.0