

# Documentation

## Admin

**Group/Game Name:** Group 15 - Polygon Survivors

**Github Link** (invite *ptvc25-tutors!*): <https://github.com/lucio1199/ptvc25-polygonsurvivors>

Link from last year: <https://github.com/AdrianKzbr/ptvc24-polygonsurvivors>

**Students** (names & student ID numbers): Delen Lucio 11713066, Adrian Kurzbauer 12024005

**Graphics API:** Vulkan

---

**Note:** We are using a project we used in the last semester of PTVC.

## Description

**Story** (max. 4 sentences):

You find yourself in a digital dream world as a sphere. But the other simpler objects realize your anomalous form and higher computational load and want to reduce your complexity. Fight them and survive as long as possible. Be vary of their dangerous bullets and use more complex items to your advantage.

**Gameplay** (max. 4 sentences):

Move around and jump in order to dodge attacks. Shoot the enemies first or get eliminated when they catch you.

## Controls

### General Controls

- **ESC:** Quit
- **F1:** Toggle polygon draw modes (fill, wireframe)
- **F2:** Toggle culling modes (none, backface, frontface)
- **F3:** Switch camera modes (player tracking, free cam, drag & strafe cam)

### Effect/Debug Controls

- **T:** Toggle texture uv display
- **N:** Toggle normals display
- **B:** Toggle lightmap sampling from baked texture
- **M:** Toggle normalmapping

### Player Controls

- **WASD:** Move player
- **Space:** Jump
- **RMB + Mouse Movement:** Rotate player and camera (in tracking / free cam)
- **LMB:** Shoot

## Camera Modes

- **Player Tracking Camera:** Follows and rotates with player
  - Mouse Scroll change distance
- **Free Camera:**
  - WASD to move
  - RMB + Mouse to rotate
- **Drag & Strafe Camera:**
  - LMB + Mouse to move
  - RMB + Mouse to rotate
  - Mouse Scroll change distance

## Additional libraries

assimp, bullet3 physics, shaderc

## Gameplay

### Mandatory

#### **3D Geometry (6 points)**

- Included assimp library
- Import obj file from dynamic path
- Convert assimp mesh to geometry data
- Used for pistol model

#### **Playable (3 points)**

- Physics based player movement, camera tracking player
- Shoot enemies to win or lose by getting hit by enemy

#### **Min. 60 FPS and Framerate Independence (3 Points)**

- Implemented time handler that gets updated at the start of the render loop
- Enemy movement, player and bullet physics and particles based on deltaTime

#### **Win/Lose Condition (3 Points)**

- Win by hitting all enemies with pistol shots
- Lose by getting hit by an enemy

#### **Intuitive Controls (2 Points)**

- Implemented input system
- Input system sets up key callbacks
- Objects like player and pistol check for input
- Switching through render modes enabled by sending inputs to shaders

#### **Intuitive Camera (2 Points)**

- Camera tracking the player “over the shoulder”
- Rotation only limited by the floor
- Scroll limited to not clip into the player and to a reasonable distance overlooking the arena

#### **Illumination Model (2 Points)**

- Material on all objects
- 1 Point light in the middle of the arena
- 1 Directional light shining straight down
- Normal vectors generated by geometry or imported

### **Textures (2 Points)**

- Textures on all objects
- UV coordinates generated by geometry or imported
- Mipmapping and trilinear filtering enabled for all textures loaded - Lightmaps and generated Images excluded

### **Moving Objects (2 Points)**

- Moving player, projectiles and enemies

### **Documentation (1 Point)**

### **Adjustable Parameters (1 Points)**

- Screen resolution and fullscreen-mode changeable in assets/settings/windows.ini file

## **Optional Gameplay Features**

### **Collision Detection (6 Points)**

- Implemented Physics with bulletPhysics
- Setup physics world
- Custom Rigidbody System handling movement, transforms and collisions
- <https://www.kodeco.com/2606-bullet-physics-tutorial-getting-started/page/3>
- started with the reference, expanded to resemble a Rigidbody from Unity Engine with transforms to allow object hierarchy and a closed system handling all physics

### **Advanced Physics (4 Points)**

- Projectile handled dynamically, as well as player in a controlled way
- Collision callbacks and triggers not implemented

## **Effect Features**

### **Lighting: Lightmap using In-Game Calculation (12 Points)**

- included shaderc library to import text shaders
- Implemented lightmap baking at startup for the background
- Saves the ambient and diffusion phong part to an image, which is passed to the shader for sampling and rendering the specular phong part
- <https://www.youtube.com/watch?v=CURcteWRmKs>
- Used the videos in the playlist as starting point and built around it

### **Advanced Modelling: CPU Particle System (8 Points)**

- Particles handled on the cpu, then sent to shader
- Spawns different oriented and colored particles to signal victory and game over

### **Animation: Hierarchical Animation (4 Points)**

- Hierarchy based on the custom rigidbodies and transforms, allowing to easily attach objects to other objects
- Pistol is attached to player, but always aims up and down to follow the camera

### **Texturing: Procedural Texture (8 Points)**

- Quantized Perlin Noise for the Background
- Red and blue colors shifted slightly to create a “dreamy” effect

### **Shading: Simple Normal Mapping (4 Points)**

- Load normal map from texture and send to shader for sampling
- Normals created with geometry

## **Walk-through**

Shoot all 10 enemies to win or touch one to lose.