

CGUE22-ZombieSurvival

Zombie Survival ist ein Survival Game, indem der Spieler vor Zombies weglaufen und diese töten muss. Dabei darf er aber nicht sterben, ansonsten ist das Spiel vorbei. Es gibt verschiedene Waffen mit Upgrades sowie Granaten, mit denen der Spieler so lange wie möglich überleben kann. Die Zombies werden aber immer mehr und alle 5 Runden erscheint ein Boss-Zombie.

Das Spiel wurde mit Assimp sowie Bullet implementiert. Für die Sounds wurde IrrKlang verwendet. Die Struktur des Programms wurde in mehrere Klassen für die einzelnen Objekte geteilt. Fokus bei diesem Spiel lag von Anfang an beim Gameplay, da uns ein guter und vor allem interessanter Spielfluss wichtig war. Effekte wurden dennoch genügend implementiert.

Achtung: Um Sounds zu hören, muss wahrscheinlich die Lautstärke in der settings.ini File geändert werden (momentan eher leise eingestellt, „volume=0.005“).

Effects:

Model loading (no effect):	<p>Für das model loading mussten wir Assimp installieren (32-Bit Version). In Visual Studio laden wir Models mithilfe zweier Klassen „Mesh.h“ und „Model.h“. In „Model.h“ wird im Konstruktor die loadModel Methode aufgerufen, worin dann das Model mithilfe von Assimp und dem Pfad geladen wird.</p> <p>Die dazugehörigen Methoden sind alle in den 2 genannten Klassen zu finden.</p> <p>Für diese Aufgabe haben wir uns an den Code dieser Websites gehalten: https://learnopengl.com/Model-Loading/Assimp, https://learnopengl.com/Model-Loading/Mesh, https://learnopengl.com/Model-Loading/Model</p>
Lightmap using Separate Textures (8 P):	<p>Für die Lightmap haben wir uns die zur Verfügung gestellten Tutorials angesehen. Wir haben in Visual Studio einen eigenen Shader „lightmapShader“ erstellt. Beim lightmapShader gibt es keine Lichtquellen oder Ähnliches. Um die Lightmap zu erstellen haben wir in Blender ein Model erstellt, welches Texturen beinhaltet. Dieses Model haben wir mithilfe von Model loading in unser Spiel hinzufügen können, aber noch ohne Lightmap. Für diese mussten wir in Blender für jedes Objekt eine Lightmap „baken“ und die neu erhaltenen Textures dann mit den normalen Textures kombinieren.</p> <p>Unsere gesamte Map wurde mit Lightmaps erstellt.</p>

	<p>Für diese Aufgabe haben wir uns an den Code dieser Website gehalten: https://learnopengl.com/Lighting/Lighting-maps</p>
GPU Vertex Skinning (20P):	<p>Auch für diesen Effekt haben wir einen eigenen Shader erstellt. Der Unterschied hierbei sind die zusätzlichen Gewichte und Knochen, die für die Animation notwendig sind. Animationen werden zunächst genau wie beim Model loading in das Programm geladen. Allerdings handelt es sich bei Animation um .fbx Dateien, die wir in Blender erstellt haben. Die Fbx Dateien enthalten Animationen, welche wir selbst durch Skinning erstellt haben. Dafür haben wir ein Model aus dem Internet heruntergeladen und etwas angepasst und danach Knochen erstellt. Diese Knochen werden dann mit dem Körper verbunden und so können dann Animationen erstellt werden.</p> <p>Wir haben insgesamt 3 Animation: Walk-Animation, Hit-Animation und Death-Animation für die Zombies.</p> <p>Für diese Aufgabe haben wir uns an den Code dieser Website gehalten: https://ogldev.org/www/tutorial38/tutorial38.html</p>
CPU Particle System (8P) :	<p>Auch für die Particles haben wir einen eigenen Shader benötigt. Particles sind sehr viele kleine Quadrate, die für einen gewissen Zeitraum leben und dann wieder verschwinden. Angezeigt werden die Particles sobald man von einem Zombie geschlagen wird oder selbst ein Zombie abschießt. Die angezeigten Particles sollen Blut darstellen.</p> <p>Für diese Aufgabe haben wir uns an den Code dieser Website gehalten: http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/</p>

Gameplay:

3D Geometry (6P):	Wir haben viele 3D Objekte geladen. Unsere Zombies und Map sind 3D, sowie die Granaten. Nur das HUD haben wir in 2D erstellt.
Playable (3P):	Unser Programm ist spielbar.
Advanced Gameplay (3P):	Als advanced gameplay betrachten wir, dass Zombies abgeschossen werden können (Main.cpp) und diese den Spieler verfolgen. Zusätzlich fügen die Zombies Schaden zu, sobald man in der Nähe ist (Main.cpp). Man kann auch noch Granaten auf die Zombies werfen, welche auch Schaden machen und muss gegebenenfalls seine Waffe reloaden.
Min. 60 FPS and Framerate Independence (3P):	Das Spiel läuft auf 120 FPS. Wir berechnen immer die Zeit zwischen zwei Frames (deltaTime) und benutzen diese bei der Physikberechnung des Spiels.

Win/Lose Condition (3P):	Gewinnen kann man, indem man alle Zombies tötet und den Boss-Zombie in Runde 5 besiegt und verlieren kann man, indem man selbst von den Zombies getötet wird.	
Intuitive Controls (2P)	<i>[W,A,S,D] Movement</i> <i>[SPACE] Jump</i> <i>[F3] Free Camera</i> <i>[F4] Hud enable</i> <i>[ESC] Close</i> <i>[SHIFT] Sprint</i> <i>[E] Switch Weapon</i> <i>[R] Reload</i> <i>[G] Throw Grenade</i>	<i>[F] Buy, Upgrade, Freeplay</i> <i>[X] Respawn</i> <i>[B] Buy Grenade</i> <i>[MSB_LEFT] Shoot</i> <i>[MSB_RIGHT] Aim</i> <i>[I] Weapon Infos</i> <i>[Q] Show Stats</i>
Intuitive Camera (2P):	Erstellt wird die Camera in Camera.h (vom Template der LVA). Unsere Kamera ist so gewählt, dass man das Spiel in der Ego-Perspektive erlebt. Man kann sich unendlich nach links und rechts drehen, und 90 Grad nach oben bzw. unten. Mit F3 kann man in die Free Camera wechseln, in der die „Zeit gestoppt“ wird und man sich die Karte genauer ansehen kann.	
Illumination Model (2P):	Für dieses Feature haben wir den PhongShaderNormal erstellt. Dabei können wir directional light sowie point lights erstellen. Diesen Shader verwenden wir z.B. bei den geworfenen Granaten.	
Textures (2P):	Unsere Map, Granaten, Zombies und unser HUD besitzen Textures. (Main.cpp)	
Moving Objects (2P):	Sich bewegende Objekte gibt es in unserem Programm durch die Zombies, sowie die Granaten, die man werfen kann.	
Documentation (1P):	Dieses Dokument.	
Adjustable Parameters (1P):	Mit der settings.ini File kann man verschiedene Einstellungen wählen. Unter anderem z.B. die Größe, Belichtung und Lautstärke.	

Optional Gameplay:

Collision Detection (Bullet) (4P):	Es sind alle Basic Physics vorhanden. Die Bullet Objekte werden entweder in den Klassen (Zombie.h, Player.h) erstellt oder für die Map in Main. Mithilfe der setBulletModelMatrix Methode werden alle Objekte an die richtige Stelle gesetzt bzw. transformiert (Main.cpp).
Advanced Physics (6P):	Als advanced physic haben wir das Werfen der Granaten implementiert. Das Bullet Objekt dafür wird erstellt, sobald man G drückt und noch Granaten übrig hat. Danach wird eine Variable geändert, sodass in der While-Schleife des Programms alle Berechnungen und das Zeichnen der Granate durchgeführt werden können (grenadeCalc(), Main.cpp).

Heads-Up Display (4P):	Auch für das HUD haben wir einen eigenen Shader erstellt (HudShader), um die 2D Objekte zu Laden. Diese Objekte wurden mit der Geometry Klasse erstellt (neue Methoden: createPlaneGeometry und createCircleGeometry) und dann mit dem Shader geladen. In der While-Schleife des Programms werden diese Objekte dann gezeichnet (hudCalc(), Main.cpp)
-------------------------------	---

Additional Libraries:

Assimp, um 3D Modelle zu laden:	https://www.assimp.org/
Bullet, für Collision Detection:	https://github.com/bulletphysics/bullet3/releases
IrrKlang, für Sounds:	https://learnopengl.com/In-Practice/2D-Game/Audio

Features / special Features / complex interactions:

Man spawned als Spieler auf der Map und befindet sich direkt in Runde 1 von 5. Es spawnen gleichzeitig Zombies, die einen verfolgen (mithilfe von Vektorberechnungen, keine AI). Zu Beginn hat der Spieler nur eine Pistole, mit der er die Zombies töten kann (MSB_LEFT) und dann jeweils 2 Gold erhält. Es ist auch möglich genauer zu schießen und die Zombies anzuvisieren (MSB_RIGHT). Sollte man einen Zombie treffen, dann blutet dieser (Particles). Der Spieler kann zu 4 verschiedenen Wänden gehen, an denen Waffen „hängen“. Wenn man nah genug daran steht und genügend Gold hat, kann man diese Waffen kaufen bzw. upgraden (F). Jede Waffe hat unterschiedliche Eigenschaften. Unter anderem auch Munition, die pro Schuss verbraucht wird. Zombies dropen aber hin und wieder Munition, die man sammeln kann und mit der man dann nachladen (R) kann. Als Spieler ist es auch möglich Granaten zu werfen (G). Diese richten in einem großen Radius viel Schaden an. Man besitzt nur maximal 3 Granaten, kann mit 20 Gold aber wieder welche nachkaufen (B).

Die Zombies schlagen den Spieler, sobald sie nah genug herankommen, und richten dabei Schaden an. Verliert der Spieler 100 Leben, dann ist das Spiel vorbei. Man kann allerdings wieder von neu beginnen (X). Sollte man von einem Zombie-Schlag nicht sterben, dann regeneriert man nach 5 Sekunden langsam wieder das Leben. Nach 5 Runden erscheint bereits ein Boss-Zombie. Dieser hat sehr viel mehr Leben, aber dafür eine Lebensanzeige. Zusätzlich richten Schläge des Boss-Zombies sehr viel Schaden an. Sollte man den Boss und alle anderen Zombies in Runde 5 töten, dann ist das Spiel gewonnen. Man kann allerdings weiterspielen (F) und dann spawnen alle 5 Runden wieder Boss-Zombies und man hat mehr Zeit, um alle Waffen auszuprobieren und zu upgraden.

Mit der F3 Taste kann man in die Free Camera wechseln und sich die Karte ansehen. Das Spiel wird dabei „gestoppt“. Um alle Controls zu sehen kann man C drücken. Auch hier wird „gestoppt“.

Special Controls für das LVA Team:

[H] zum Healen, [T] um Gold zu erhalten (So kann man schnell bis Runde 5 spielen).