

TISS SIMULATOR

Katharina Weindl (11777728)

Gameplay

„TISS Simulator“ ist ein Puzzle Runner Game.

Spielumgebung ist das Gußhaus der TU Wien in welcher die spielende Person versuchen muss sich bei TISS für eine Lehrveranstaltung anzumelden, deren Anmeldung um 12:00:00 beginnt. Um einen Platz in der LVA zu bekommen müssen Laptops, die in den Räumen stehen, kaputt gemacht werden. Ein Laptop steht hierbei für einen Studierenden der sich anmelden will. Ist die Anzahl der Studierenden eins weniger als die Anzahl der Plätze für die LVA, kann man sich anmelden, jedoch muss das genau um 12:00:00 sein, ansonsten bekommt den Platz jemand anders der schneller war.

HUD

Das HUD besteht aus Alpha geblendetem Text der im Screen ganz oben links platziert wird. Die dafür verwendete Schrift wird mittels der Bibliothek FreeType geladen, welche aus einer TrueType-Schrift (eine durch mathematische Gleichungen definierte Schrift) Bitmaps rendert.

Um immer genau zu wissen wie spät es ist, gibt es links oben im Screen eine Anzeige der wichtigsten Spielinformationen: Zeit, Studierendenanzahl und fps Anzahl.

Die Zeit zeigt an wie spät es gerade ist und veranschaulicht wie viel Zeit einem noch bis zur Anmeldung bleibt. Sind es nur noch 10 Sekunden bis zur Anmeldung wird jene rot angezeigt.

Die Studierendenanzahl gibt an wie viele sich anmelden wollen und wie viele Plätze die LVA zur Verfügung hat.

Die fps geben an wie viel Bilder pro Sekunde gezeichnet werden.



Spielanleitung

Gestartet wird im Eingangsbereich vom Gußhaus, mittels den Tasten W,A,S,D kann durch die Räume navigiert werden und mittels der Maus kann das Blickfeld verändert werden.

Wird ein Laptop entdeckt kann er mittels der Taste E kaputt gemacht werden, wodurch die Anzahl der Studierenden die sich anmelden wollen um eins weniger wird, da sich der Studierende, dem der Laptop gehört, nun nicht mehr anmelden kann. Ziel ist es alle Laptops kaputt zu machen um sicher zu stellen einen Platz in seiner gewünschten LVA zu erhalten.



3D Geometrie

Die 3D Modelle wurden in Blender erstellt. Das größte Model ist der Gußhaus Nachbau, weiters existieren noch Laptops mit unterschiedlichen Screens als 3D Geometrie die im Spiel verwendet werden. Die Modelle werden als obj exportiert mit einem zugehörigen mtl File. Um diese im Spiel laden zu können wird die Bibliothek Assimp verwendet und ein Model Loader (Mesh.h/Model.h) wurde implementiert.

Collision Detection

Die Collision Detection funktioniert über eine 2D Map. Diese ist ein selbst erstelltes PNG Bild, bei dem die Pixelkoordinaten durch die Kameraposition umgerechnet werden. Der Rot Wert (0-255) gibt die Höhe der spielenden Person an, der Grün Wert wird verwendet um die Position der Laptops festzustellen. Ist ein Pixel Schwarz (Werte sind alle 0) befindet sich an dieser Stelle ein nicht durchdringbares Objekt. Die spielende Person wird, sobald sie auf einem schwarzen Pixel steht, an die letzte Position zurück teleportiert. Zudem wird eine „Danger Zone“ mittels eines bestimmten Grünwertes um die Wände gelegt. Die letzte gespeicherte Position, die nicht in der „Danger Zone“ liegt, also die in Sicherheit liegende „Last Save Position“, wird genutzt um den Spieler zurück zu teleportieren falls das vorhergehende "Last Position" Teleportieren gliched.

Controls

Key	Effect
W, A, S, D	Bewegen
E	Laptop kaputt machen
W + SHIFT	Rennen
ESC	zurück zum Startscreen/Spiel beenden

Key	Effect
H	HUD on/off
Mausbewegen	Blickrichtung

Camera

Die Kamera ist eine First Person Camera die mittels Tasteninput bewegt und gesteuert werden kann.

Textures

Texturen werden verwendet um die unterschiedlichen Screens (start-screen, control-screen,...) darzustellen. Dafür wird ein Rechteck in der Größe des Fensters erstellt und die Textur wird unter Verwendung von UV-Koordinaten gemappt. Trilinear filtering und mipmapping wird dabei verwendet.

Moving object

Der Anmeldelaptop dreht sich während dem Spiel um ihn besser von den anderen Laptops unterscheiden zu können. Sind es nur noch 10 Sekunden bis zur Anmeldung wird er passend rotiert und verändert seine Position bis die Zeit um ist nicht mehr.

Effekte

Video Textures

Im großen Raum vor dem EI7 gibt es einen Bildschirm auf dem ein Video am Anfang des Spiels gezeigt wird. Dafür wurde das Video in einzelne Frames zerlegt, die als Texturen dann jeweils geladen werden. Wann welcher Frame angezeigt wird, wird abhängig von der Zeit, die bereits seit Start des Spieles vergangen ist, und der Anzahl der Frames, die pro Sekunde gezeigt werden sollen, berechnet.

CPU Particle System

Wird das Spiel gewonnen werden Particle mit Instancing generiert, die buntes Konfetti darstellen sollen. Das Konfetti erscheint für 3 Sekunden, dann wird der won-screen angezeigt. Um nur die Partikel anzuzeigen, kann auch im start-screen G gedrückt werden. Quelle: <http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/>

Hierarchical Animation

Dieser Effekt wurde durch ein Auto, dass an der Uni beim Eingang vorbeifährt, realisiert.

Implementierung

Main.cpp

Enthält die Game-Loop und hier werden alle Shader und Modelle geladen.

Window.h/ Window.cpp

In der Window.h und Window.cpp Datei wird das Fenster erstellt und alles dafür wesentliche (OpenGL Version, Callbacks, Cursor,..) gesetzt und initialisiert. Aufgerufen wird das Fenstererstellen in der Main.cpp.

Settings.h

Diese Datei enthält den Debug Kontext und die Methoden um sich im Spiel bewegen zu können, also die gesamte Steuerung der Maus und der Tasten. Weiters werden hier die Werte aus dem settings.ini File geladen. (width, height, refresh-rate, fullscreenmodus, brightness).

Camera.h

Enthält die Implementierung der First Person Camera.

Collision.h/Collision.cpp

Diese Dateien enthalten die Implementierung unserer collision detection.

Gameplay.h/Gameplay.cpp

Hier ist der gesamte Spielablauf implementiert, unter anderem die Win/Loose Conditions, die Screens (Startscreen, Controlscreen,..), die Zeitanzeige, der Counter für die Studierendenanzahl, der fps Zähler und das Laptop kaputt machen.

Mesh.h/Model.h

Enthält alle Methoden um ein 3D Modell in das Spiel hinein zuladen unter Verwendung der Assimp Bibliothek.

Text.h/Text.cpp

Hier sind alle Methoden enthalten, um Text am Bildschirm anzeigen zu können. Dafür wird die FreeType Bibliothek verwendet.

Particle.h/Particle.hpp

Enthalten die Methoden für das Particle System.

Quad.h/Quad.cpp

Hier sind alle Methoden enthalten, um die unterschiedlichen Screens darzustellen und hier ist die Implementierung vom video textures Effekt enthalten.

Animation.h/Animation.cpp

Hier wurde ein Teil des hierarchical animation Effekts implementiert.

Bibliotheken

Assimp (<http://www.assimp.org>)

FreeType (<https://www.freetype.org>)

GLFW (<https://www.glfw.org>)

GLEW (<http://glew.sourceforge.net>)

Quellen:

Als Hilfestellung zur Implementierung wurde LearnOpenGL (<https://learnopengl.com>) und opengl-tutorial verwendet (<http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/>).

Schriftart

Für das HUD wird die Schrift Steelfish (<https://fontmeme.com/fonts/steelfish-font/>) verwendet.

Texturen

Alle Texturen, die für die Modelle verwendet werden, stammen von der Webseite Pexels und sind selbstständig nacheditiert (<https://www.pexels.com/de-de/>).

Die Screens (start-screen, control-screen,...) sind eigenständig editierte Kompositionen aus Bildern von Pexels und Adobe Stock mit der Schriftart Steelfish. Das Video wurde ebenfalls selbständig erstellt.

Model

Das 3D Mensch Model stammt von der Website Free3D (<https://free3d.com/de/3d-model/male-base-mesh-6682.html>).