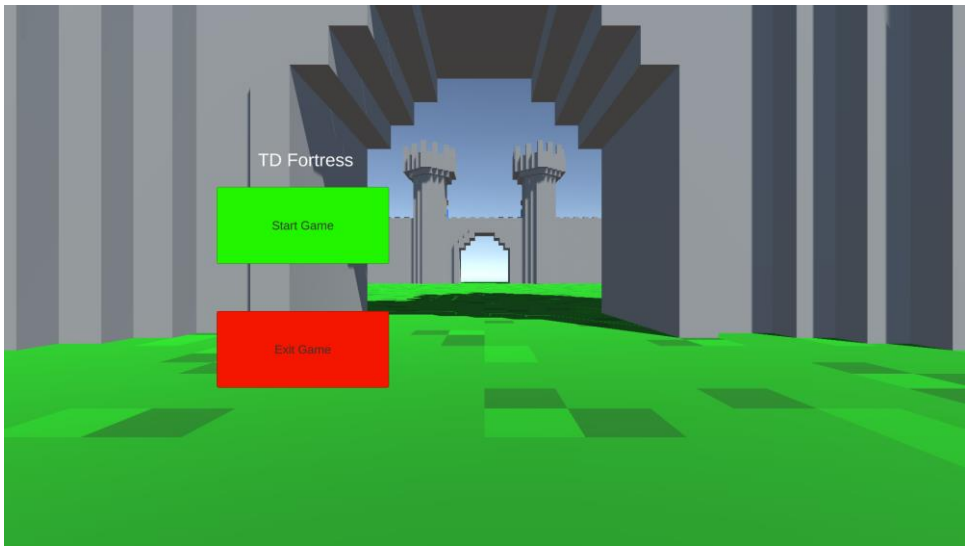
TD Fortress Game – Dokumentation

Spieleprogrammierung

Prof. Dr. Carsten Lecon

Gruppe 13:

Marc Dommröse 84049 (Allg. Informatik)



Inhaltsverzeichnis

Projekt-Idee

Die Projekt-Idee ist ein Tower Defense Spiel. Dabei soll der Spieler die Möglichkeit haben auf Maus-Klick einen Turm zu setzen und damit die Reihen an durchlaufenden Gegnern zu zerstören. Das Spiel benutzt dabei einen Pathfinding Algorithmus damit der Spieler seinen eigenen Weg zusammenbauen kann. Das Spiel soll im Endlos-Modus laufen, damit der Spieler so lange wie möglich spielen kann. Die Gegner werden hierbei immer stärker. Somit muss der Spieler eine lange Bahn selber bauen wodurch die Gegner lange laufen müssen.

Das Spiel ist verloren, sollte der Spieler kein Geld mehr haben und dann von einem Gegner Schaden bekommen. In diesem Fall soll das Spiel neu gestartet werden.

Projekt-Umsetzung

Tools, die zur Umsetzung genutzt wurden:

- Unity (2022.3.4f)
- MagicaVoxel

Das Spiel wurde mit der Game-Engine Unity umgesetzt. Es wurden einige Assets genutzt aber auch selber kreiert oder bearbeitet mit der MagicaVoxel Software.

Die meisten Assets wurden aus einem Asset Pack genommen und überarbeitet. Das Asset Pack wurde mir von einem Freund gegeben als ZIP.

Die Welt wurde komplett selbst erstellt.

Es wurde mit sehr viel Tutorials aus dem Internet gearbeitet, da ich mich mit Unity etc zuvor nicht ausgekannt habe. Ich wollte außerdem unbedingt einen Pathfinding Algorithmus einsetzen, und habe hierzu mich an sehr viele Tutorials für Unity gehalten, welche dieses Problem sehr gut umsetzen konnten.

Assets

Voxel Castle Asset Pack

Main Game Music: Age of War – Main Theme Soundtrack

Skripte

Menu.cs:

Simples Skript, um ein Menü während dem Spiel aufzurufen

Tower.cs:

Erstellt den Turm des Spielers mit ein wenig Verzögerung, um das Spiel etwas schwerer zu machen, da der Turm nicht direkt schießen kann. Außerdem werden Gold-Kosten beachtet

TargetLocator.cs:

Ziel auf den Gegner in der Nähe und schießt wenn dieser nah genug ist.

CoordinateLabler.cs:

Skript welches zum Debuggen genutzt wurde. Zeigt im Editor Pathfinding etc an. Kann während dem Spiel mit “C” aufgerufen werden. Der Spieler sollte es eigentlich nicht benutzen können, aber im Sinne dieses Projekts wurde es drinnen gelassen

Waypoint.cs:

Bestimmt, ob ein Turm erlaubt werden soll, gesetzt zu werden oder nicht mit Linksklick.

GridManager.cs:

Der GridManager erstellt ein Grid für den Pathfinding Algorithmus.

Pathfinder.cs:

Pathfinder.cs implementiert den relativ komplizierten BreathFirstSearch Algorithmus. Hierzu wurde Code aus verschiedenen Tutorials verwendet und angepasst.

Node.cs:

Eine Custom-Klasse, welche die Nodes zum Traversieren mit dem Pathfinding-Algorithmus erstellt btw darstellt.

Gold.cs:

Dieses Skript zeigt die Anzahl an Gold an die der Spieler besitzt und fügt dieses hinzu bzw. nimmt es weg, wenn man Türme kauft oder Schaden nimmt.

Enemy.cs:

Simples Skript das dem Gegner erlaubt Gold zu “klauen” oder hinzuzufügen, wenn dieser Gegner zerstört wird

EnemyHealth.cs:
Simples Skirpt, welche die Leben des Gegners beachtet und das Spiel schwerer macht, indem es die MaxHP des Gegners erhöht mit der Zeit bzw. je öfter der Gegner stirbt.

EnemyMover.cs:

Entscheidet darüber, wohin sich der Gegner bewegen kann. Es wird mit der Lerp Funktion von Unity gearbeitet, um den Gegner schön von einem Feld zum anderen zu bewegen. Es beachtet hier die zuvor erwähnten Pathfinding Skripte

ObjectPool.cs:

Code ist größtenteils aus Tutorials. Es wurde erwähnt, dass ObjectPools das Spiel effizienter laufen lassen, da Objekte wiederholt genutzt werden, und nicht immer wieder neu geladen werden müssen. Dies wurde implementiert, um zu lernen, damit umzugehen. Es ist allerdings für ein kleineres Spiel dieses Maßstabs nicht wirklich nötig.

Alle Skripte und Assets können im Asset Folder gefunden werden. Skripte sind zusätzlich in passende Ordner unterteilt.

Herausforderungen

Beim Bearbeiten des Projekts gab es einige Herausforderungen:

- Z-Fighting: Wenn man in Unity Objekte auf der gleichen Höhe erstellt, wird Z-Fighting provoziert. Dabei gibt es graphische Fehler. Da irgendwann mit einem Grid gearbeitet wurde, war dies kein großes Problem mehr
- Game-Design: Es konnte noch nicht evaluiert werden, ob das Spielprinzip verständlich dargestellt wird, da es nur eine sehr kleine Spieltestgruppe gab. Als Entwickler des Spiels hat man hier eine andere Sicht.
- Game-Test: Das Spiel wurde noch nicht von einer größeren Testgruppe getestet, weshalb ich nicht 100% sicher sein kann, ob alles so läuft wie es laufen soll. Auch hier hat man eine andere Sicht als der Spieler als Entwickler
- Pathfinding: Das Pathfinding war eine Herausforderung, die ich unbedingt einmal erledigen wollte. Es gibt hier leider ein paar geringfügige Probleme mit dem Umdrehen der Gegner

wegen des Pathfindings aber ich bin größtenteils sehr glücklich, wie es nun aussieht und funktioniert
- Unity-Probleme: Es gab öfters ein Problem, dass durch Neustarten von Unity behoben werden konnte.

Abgabe

Das Projekt kann unter dem Git Repository gefunden werden:

https://github.com/MarcDiDo/TDFortressGame

Hier befindet sich diese Dokumentation und eine Spieleanleitung, sowie die Zwischenpräsentation, Abschlusspräsentation, die Exe-Files (Gebuildetes Spiel von Unity) und Unity Files mit denen man das Spiel in Unity editieren kann. Dabei sind die Assets sowie Skripte, etc.