Universität Hamburg
Department Informatik

Projektdokumentation: Tower Defense 2D

Programmiertechnisches Praktikum 2019 64-146b

Christoph Lankau
Matr.Nr. 7195269
8lankau@informatik.uni-hamburg.de

Michal Zlotnik
Matr.Nr. 7054053
michal.zlotnik@studium.uni-hamburg.de

15.08.2020

Grobbeschreibung

Die Idee von uns ist es ein einfaches kleines Tower Defense Game zu erstellen. Die grundlegende Elemente sind dabei: Beim Starten des Programmes soll das Fenster schon eine feste Größe haben. In dem Fenster ist dann schon das Spielfeld zu sehen. Das Spielfeld ist in kleine Rechtecke (Grid) unterteilt, dabei gibt es drei unterschiedliche Farben. Die Farben stellen einmal die Route für die Gegner, Platzierbare und nicht platzierbare Objekte(Türme) da. Die Gegner gehen von einer zur anderen Seite. Dabei soll der Spieler Türme mit einem Klick auf einem Rechteck Platzieren können. Diese Türme haben eine bestimmte Reichweite und wenn ein Gegner den Turm zu nahe kommt verliert er Leben und verschwindet, wenn dieser genug schaden abbekommen hat. Schafft es ein Gegner von einer zur anderen Seite dann verliert der Spieler das Spiel.

Zur Erweiterung des Programms würde dann zählen: Start Menü, unterschiedliche Schwierigkeit Stufen der Gegner, mehr Gegner und Türme mit unterschiedliche Funktionen/Fähigkeiten, Geld/Kosten System (dabei geben Gegner beim Zerstören Geld und beim Platzieren von Türmen verbraucht man das Geld), zum Platzieren der Türme werden Türme aus einer Box ins Spielfeld gezogen, eine einfache AI/KI für die Gegner um Pfade zu folgen, unterschiedliche Karten.

Inhaltsverzeichnis

1 Sollkonzept

1.1 Anwendungsszenario

Das Spiel hat eine Reihe von Funktionen, die ein Benutzer verwenden kann. Erstens kann ein einfaches Spiel mit einer vordefinierten Karte gespielt werden. Dann kann eine neue Karte bearbeitet, gespeichert und dann gespielt werden. Schließlich gibt es drei Schwierigkeitsstufen, in denen Karten und Gegner vordefiniert sind

Die einzelne Level besteht aus einer Karte und feindlichen Wellen. Der Spieler hat eine vordefinierte Anzahl von Leben und Geld für Türme. Ein Level ist abgeschlossen, wenn alle feindlichen Wellen besiegt sind oder die Wellen enden und der Spieler genügend Leben hat.

1.2 Userstories

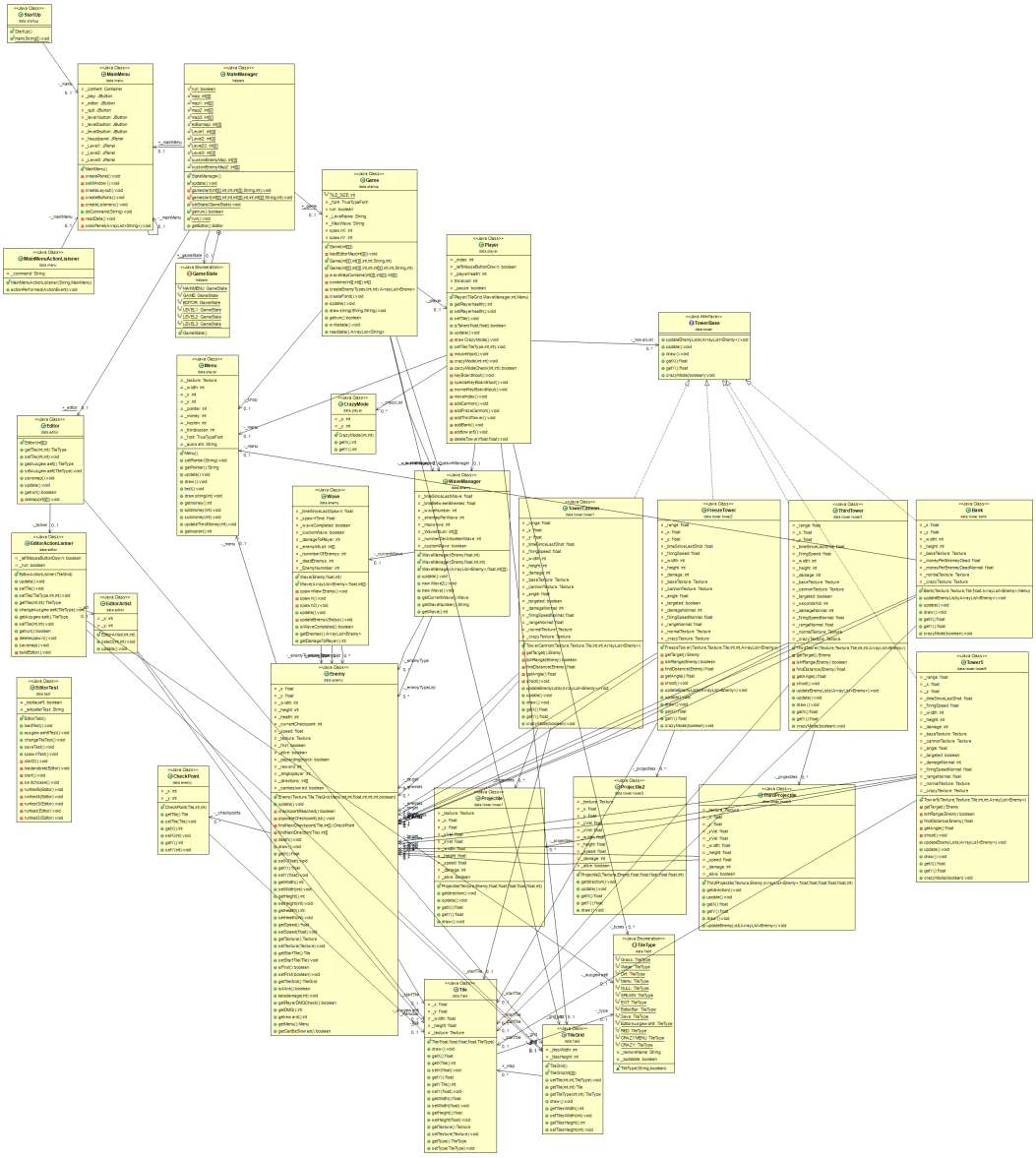
Beschreibung der einzelne Teile der Userstories
Einfache Texturen: Einfache Texturen sind Dummy Texturen, die nur für das Programm wichtig sind.
Aufbauen des Spielfelds: Ordnen der Textruen
Timer(Clock): Für bewegliche Einheiten, bewegliche Türme. Zeit zwischen der aktuellen Zeit und der letzten Aktualisierung des Spiels.
Level erstellen: Dummy-Ebene für die weitere Implementierung.
Gegener und Generwellen: Feinde, die sich in der Geschwindigkeit, mit der sie sich bewegen, und in der Anzahl der Leben unterscheiden. Die Wellen unterscheiden sich in der Anzahl der Feinde.
Gegener KI: Der Feind rückt vor, wenn das nächste Feld vom gleichen Typ wie sein aktuelles ist. Andernfalls prüft er oben, links/rechts und unten. **Turm Base:** Basis des Turms.
Turm Cannon: Kanone des Turms.
Projektile: Projektil, das vom Turm auf den Feind geschossen wird.
Kollsion: Projektil, das den Feind trifft.
Gewinnen/Verlieren: Gewinnen: die feindlichen Wellen sind beendet und der Spieler hat genug Leben. Verloren: Spieler hat alle Leben verbraucht.
Refactoring: Verbesserte Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit des Codes.
Zweiter Turm: Zweiter Turmtyp.
Zweiter Feind: Zweiter Feindtyp
Anzeigen: Platzieren von Texturen auf dem Feld.
Menü: Hauptmenü des Spiels.
Level 1-3: Drei Stufen des Spiels.
Endlose Wave: Endlose Welle von Feinden.
Dritter Turm: Dritter Turmtyp.
Editor: Ermöglicht es dem Benutzer, eine eigene Karte zu erstellen.
Editor Test: Test des Editors.
Turm 4: Vierter Turmtyp.

Turm 5: Fünfter Turmtyp.
Level/Gegner Anpassung: Anpassung der feindlichen Wellen an das Niveau.

1.3 Zeitplanung

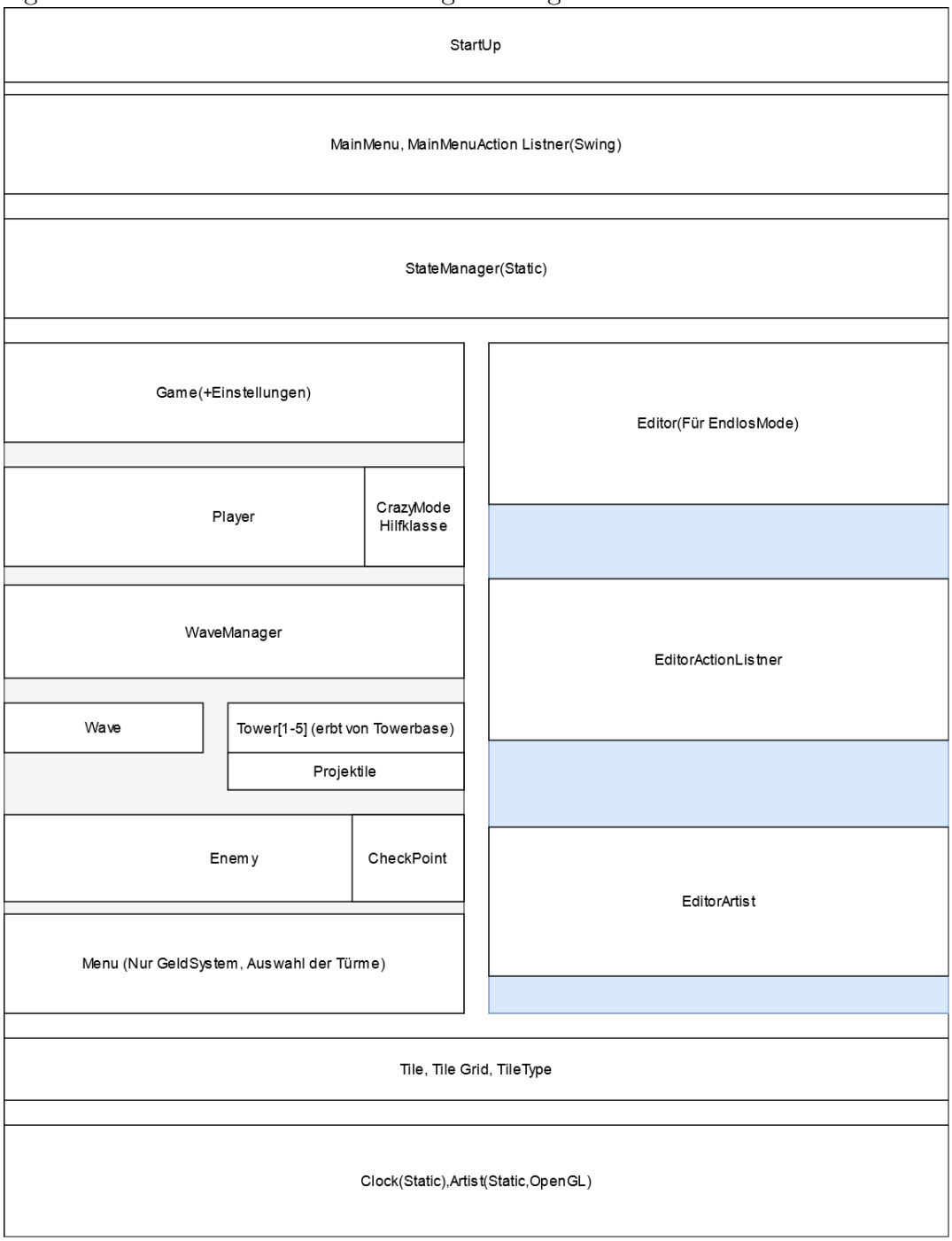
Prio.	User Story	Zeitaufwand(Soll)	Zeitaufwand(Ist)
1	Einfache Texturen	1h	1h
2	Aufbauen des Spielfelds(Grid)	4h	5h
3	Timer(Clock)	2h	2h
4	Level erstellen	30min	20min
5	Gegener und Generwellen	3h	4h
6	Gegener KI	3h	2h
7	Turm Base	2h	3h
8	Turm Cannon	2h	3h
9	Projektile	2h	1h 30 min
10	Kollision	2h	3h
11	Gewinnen/Verlieren	2h	2h
12	Refactoring	-	4h
13	Zweiter Turm	2h	2h
14	Zweiter Feind	1h	45min
15	Anzeigen	5h	5h
16	Menü	4h	5h
17	Level 1-3	4h	2h
18	Endlose Wave	1h	1h
19	Dritter Turm	2h	1h 30min
20	Editor	4h	5h
21	EditorTest	1h	1h
22	Turm 4	2h	1h 30min
23	Turm 5	1h	30min
24	CrazyMode	3h	4h
25	Level/Gegner Anpassung	1h	40min
	Gemsammtzeit	54h 30min	60 h 45min

2 UML Diagramm



3 Übersicht

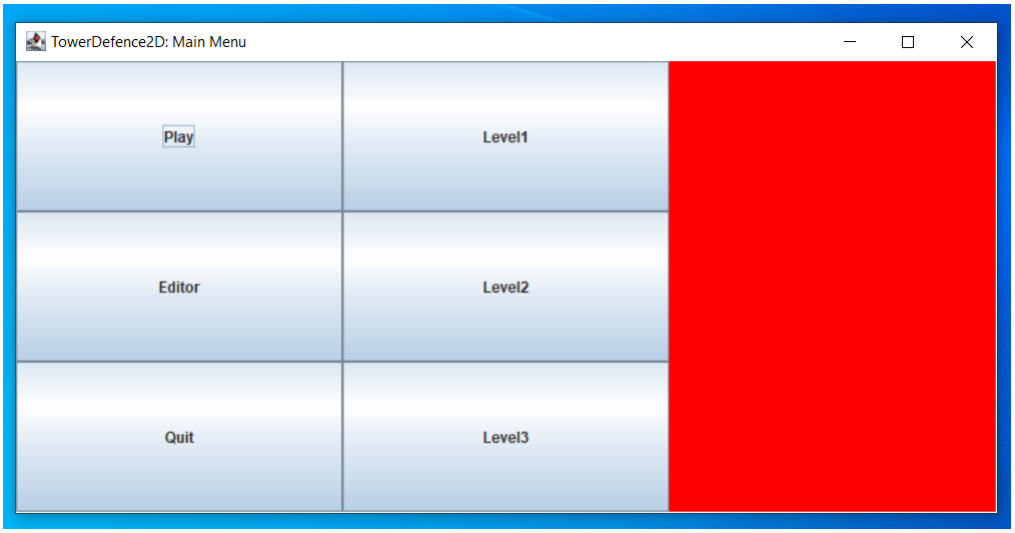
Darstellung aller Klassen(bis auf die Tests). Klassen können nur auf Klassen der eigenen Schicht und der darunter liegende zugreifen.



4 Testing

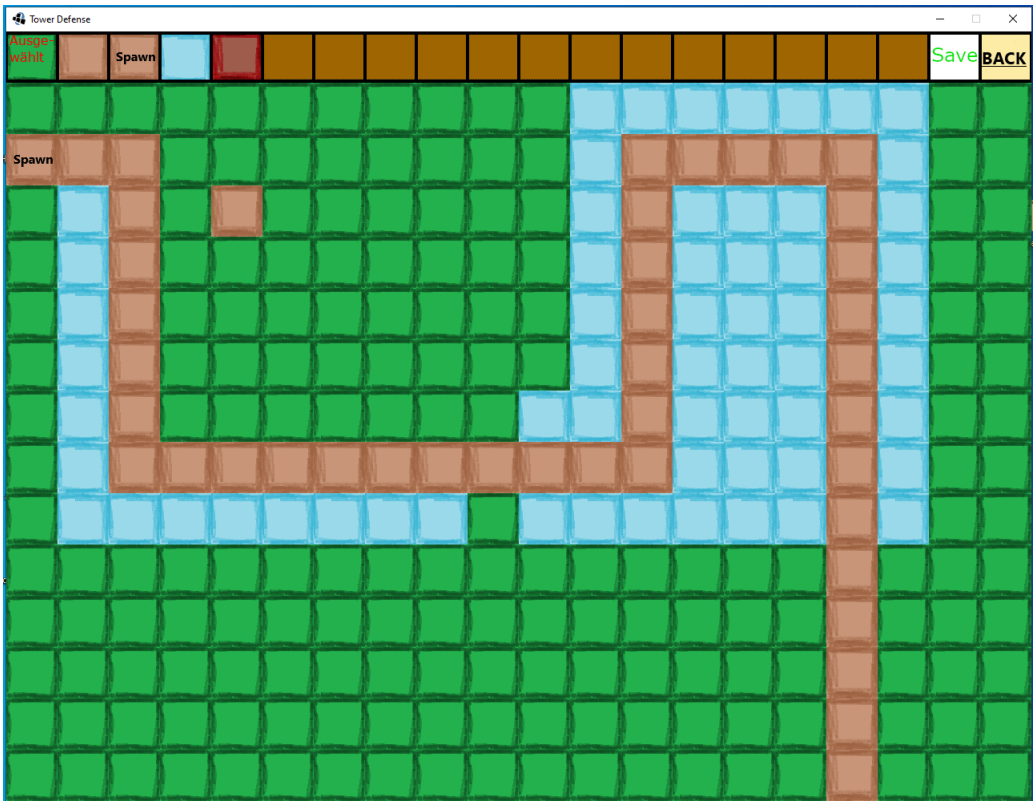
Implementierte Tests gibt es nur für den Editor-Modus. Sie testen die erfolgreiche Extraktion des Feldes und die Platzierung von Texturen auf der Karte. GUI und Win/Loss-Bedingungen sollten ebenfalls Bestandteil der weiteren Tests sein.

5 Beschreibung für Benutzer



Hauptmenü:

- Play: vordefinierte Karte mit endloser Welle von Feinden.
- Editor: Erstellen benutzerdefinierter Karten.
- Level 1- 3: vordefinierte Karten mit zunehmendem Schwierigkeitsgrad.



Editor: Um eine Textur auf der Karte zu platzieren, muss der Benutzer eine Textur aus dem oberen Menü auswählen (linker Mausklick) und dann auf ein Feld klicken. **Spawn** ist der Startpunkt für die feindlichen Wellen. Um eine Karte zu speichern, muss der Benutzer auf die Schaltfläche **Save** im Menü klicken. Die benutzerdefinierte Karte kann abgespielt werden, wenn der Benutzer zurückklickt und dann auf **Play** klickt.

Levels 1-3:

Um Verteidigungstürme zu aktivieren, müssen tastaturrelevante Tasten gedrückt werden. Dann wird sie durch Anklicken mit der linken Maustaste auf das gewählte Feld auf der Karte platziert. Türme können nicht auf Wassertexturen (blau) platziert werden. Türme unterscheiden sich in Kosten und Schaden, den sie den Feinden zufügen.

 Gedrückte **Taste '1'**.

 Gedrückte **Taste '2'**.

 Gedrückte **Taste '3'**: spezieller Turmtyp. Er kann nur auf ein rotes Feld gestellt werden, schießt zwei Projektile ab, die beim Einschlag die Feinde sofort zerstören.

 Gedrückte **Taste '4'**: schießt nicht auf die Feinde, sondern fügt nach der Vernichtung eines Feindes zusätzliches Geld hinzu.


Gedrückte **Taste '5'**.

Gedrückte **Taste '9'**: schaltet das CrazyMode ein, der das TowerUpgrade aktiviert. Türme schießen schneller und richten mehr Schaden bei den Gegnern an.


Gedrückte **Taste '0'**: aktiviert den Modus, mit dessen Hilfe ein ausgewählter Turm aus dem gewählten Feld gelöscht werden kann. Dabei werden die Kosten für diesen Turm nicht zum Gesamtbetrag der verfügbaren Gelder addiert.
Gedrückte **Taste 'ESC'**: gibt das Hauptmenü des Spiels zurück.
Gedrückte Taste **'SPACE'**: mit dem erstenmal Drücken von 'Space' kann man das Spiel stoppen und mit dem zweitenmal wieder starten.

6 Nächste Schritte und Lessons Learned

Die nächsten Schritte sollten das Refactoring und die Implementierung in ein Design-Pattern (z.B. MVC), die Verringerung der Anzahl der Redundanzen im Code, das Hinzufügen von Schnittstellen, die Verbesserung der Lesbarkeit des Codes und die Erweiterung der Tests umfassen.

Ein funktionierendes Spiel/Software und relativ fehlerfrei bedeutet keine ausreichende innere Struktur, die eine einfache weitere Implementierung ermöglicht. Fehlende Entwurfsmuster führen zu einer chaotischen Implementierung beim Hinzufügen zusätzlicher Funktionen.

Der Mangel an ausreichenden Kommentaren, lesbarem Code und entsprechenden Tests führt zu technischer Verschuldung. Folglich erfordert die weitere Implementierung langfristig zusätzliche Zeit.

Gute Planung, Aufgabenzuweisung und vor allem Kommunikation sind die Grundlagen, wenn es um die Arbeit an Projekten geht. Das Auslassen eines der oben genannten Punkte führt zu unklarem Projektstatus, unbekannten Fehlern, die plötzlich entdeckt werden, und einer chaotischen Projektstruktur.