Smurfs Gone Wild

1. Juli 2018

- Duc Vu Nguyen (Matrikelnummer: 285443)
- Torben Rogge (Matrikelnummer: 279051)

Inhaltsverzeichnis

1 Einleitung

"Poor Gargamel is under attack. The peace-loving, generous patron to the Smurfs, is under vicious bombardement by just these little bastards. They must have chewed on their mushrooms for quite too long... Help Gargamel defend himself!" In diesen Projekt "Smurfs Gone Wild" werden wir eine Auswahl relevanter Webtechnologien kennenlernen, um Gargamel zu retten.

1.1 Technologien

Im diesem Projekt haben wir folgendes benutzt:

- HTML
- CSS
- Dart
- HTTP
- JSON

2 Anforderungen

2.1 AF-1: Single-Player-Game als Single-Page-App

- Das Spiel ist als Ein-Spieler-Game zu konzipieren.
- Das Spiel ist als HTML-Single Page App zu konzipieren.
- Das Spiel muss als statische Webseite von beliebigen Webservern (HTTP) oder Content Delivery Networks (CDN) bereitgestellt werden können (bspw. mittels GitHub Pages).
- Alle Spielressourcen (z.B. Level-Dateien, Bilder, CSS-Dateien, etc.) sind daher relativ und nicht absolut zueinander zu adressieren.

2.2 AF-2: Balance zwischen technischer Komplexität und Spielkonzept

- Sie sollen ein interessantes Spielkonzept entwickeln oder ein bestehendes Spielkonzept abwandeln. Spielkonzepte sind nicht immer technisch komplex (z.B. Memory).
- Sie sollen jedoch ein Spiel konzipieren, dass eine vergleichbare Komplexität mit den Spielen der Hall-of-Fame hat (Memory wäre bspw. zu einfach; Schach zu kompliziert, da sie für ein Ein-Spieler-Game eine Gegner-KI entwickeln müssten).
- Unabhängig von der inneren Komplexität soll das Spiel schnell und intuitiv erfassbar sein und angenehm auf einem SmartPhone zu spielen sein.

2.3 AF-3: DOM-Tree-basiert

- Das Spiel soll dem MVC-Prinzip folgen (Model, View, Controller).
- Das Spiel soll den DOM-Tree als View nutzen.
- Es sind keine Canvas-basierten Spiele erlaubt. Hintergrund: Sie sollen Webtechnologien lernen und nicht wie man eine Grafikbibliothek programmiert.

2.4 AF-4: Target Device: SmartPhone

- Das Spiel soll für eine SmartPhone Bedienung konzipiert werden.
- Entsprechende Limitierungen sind zu berücksichtigen.
- Als Target Devices sind die Plattformen Android und iOS zu berücksichtigen.
- Das Spiel soll mit HTML5 mobile Browsern auf den genannten Plattformen spielbar sein.

2.5 AF-5: Mobile First Prinzip

- Das Spiel soll bewusst für SmartPhones konzipiert werden.
- Das Spiel soll auch auf Tablets und Desktop PCs spielbar sein. Einschränkungen sind zu minimieren, werden aber billigend in Kauf genommen (z.B. fehlende 3D-Lage bei Desktop Browsern).
- Sie sollen typische mobile Interaktionen sinnvoll nutzen (z.B. Swipe, Wischen, 3D Lage im Raum, etc.).
- Übertragen sie nicht eine typische Desktop-Bedienung auf Mobile.
- Sie müssen allerdings keine mobile Interaktionen sklavisch nutzen - wenn dies nicht sinnvoll für das Spielkonzept ist.

2.6 AF-6: Das Spiel muss schnell und intuitiv erfassbar sein und Spielfreude erzeugen.

- Das Spiel muss schnell und intuitiv erfassbar sein.
- Das Spiel muss Spielfreude erzeugen.
- Hintergrund: Ihre Spiele sollen ggf. als Anschauungsbeispiele für Studieninteressierte Schüler auf Messen oder ähnlichen Veranstaltungen gezeigt werden. Sie arbeiten also nicht für "die Tonne" - andere sollen ihre Spiele tatsächlich spielen.

- Tipp: Vermeintlich einfache Spielprinzipien haben ihre Stärken und sind dennoch komplex genug um die technischen Anforderungen dieses Projekts abzudecken.

2.7 AF-7: Das Spiel muss ein Levelkonzept vorsehen

- Es ist ein steigender Schwierigkeitsgrad über mindestens 7 Level vorzusehen.
- Level sollen deklarativ in Form von Textdateien beschrieben werden können (z.B. JSON, XML, CSV, etc.).
- Diese Level sollen durch das Spiel nachgeladen werden können, so dass nachträglich Level ergänzt und abgeändert werden können, ohne die Programmierung des Spiels anpassen zu müssen.

2.8 AF-8: Ggf. erforderliche Speicherkonzepte sind Client-seitig zu realisieren

- Aufgrund der Zielgruppe sollen durch das Spiel gesammelte Daten auf den Geräten bleiben.
- Aufgrund des Demonstrationscharakters auf Messen dürfen keine zentrale Server für Highscores, etc. erforderlich sein oder angebunden werden.
- Ggf. erforderliche Stateful solutions sind mittels client-seitiger Storage-Konzepte zu lösen. D.h. z.B. mittels local storage.

2.9 AF-9: Dokumentation

- Das Spiel muss nachvollziehbar dokumentiert sein.
- Das Spiel muss analog dem SnakeGame dokumentiert sein. Hinweis: Nutzen sie die SnakeGame-Dokumentation als Template.

3 Spielkonzept

Das Towerdefense Spiel enthält folgenden Komponenten und ist in 2D-Bildschirmcartoon dargestellt:

- Türme
- Spielfelder
- Gegner
- Level-Konzepte
- Lebenspunkte

- Geld
- Spielpausen - und Spielfortsetzungsknopf
- Charakterinformation
- Gegneranzahl

Aufgabe in diesem Spiel ist es, auf einer Karte, je nach Level, verschiedene Arten von Verteidigungsanlagen zu errichten, die anschließend mehrere Anstürme von unterschiedlichen Gegnern daran hindern sollen, die Karte zu durchqueren.

1. Türme

   Die Türme tauchen in Form von Gargamel, Azrael und Rotznase auf. Sie werden aus dem Navigationsbar, siehe Table 1, ins Spielfeld gezogen und dienen als Verteidigung der Gegner. Je nach Art von Turm haben sie eine bestimmte Reichweite, erzielen einen bestimmten Schaden, haben eine Kadenz und verursachen einmalig Baukosten.

2. Gegner

   Die Creeps[1] tauchen in Form von Schlümpfen auf. Sie tauchen mit einer Gruppe an einem bestimmten Eintrittspunkt der Karte auf und versuchen, zum Ausgang der Karte zugelangen. Gelingt dies einer gewissen Zahl von Gegnern verliert der Spieler die Runde. Je nach Art von Schlümpfen besitzen Sie eine bestimmte Zahl an Trefferpunkten, Geschwindigkeit und Rüstung.

   - der weißer Schlumpf stellt die grundlegendste Gegenart dar und kann von den einfachsten Verteidigungssystemen angegriffen werden.
   - der gelbe Schlumpf hat im Vergleich zum weißen Schlumpf leicht erhöhte Lebenspunkte
   - der lilane Schlumpf hat im Vergleich zum gelben- und weißen Schlumpf eine höhere Lebenspunktanzahl
   - Die Lebenspunktanzahl des grünen Schlumpfes liegt zwischen der des weißen und der des gelben Schlumpfes. Zusätzlich besitzt es eine Rüstung, welches ihm zum Tank[2] macht.

[1] (engl. creep „Kriecher/Fiesling“[2]) Wurde als Begriff zuerst in Warcraft III: Reign of Chaos verwendet, da dieses Spiel als erstes Creeps d. h. neutrale (von keinem Spieler kontrollierte) Einheiten, aufführte, welche vom Spieler bekämpft werden konnten, um die Helden zu leveln, bzw. die von den toten Creeps hinterlassenen Gegenstände zu übernehmen. Für jede getötete Einheit bekommen die Helden der Spieler je nach Fähigkeitsstufe („level“) des getöteten Creeps eine bestimmte Anzahl an Erfahrungspunkten, die wiederum benötigt werden, um den Helden auf eine höhere Stufe zu bringen. Außerdem bekommt der Spieler für jedes getötete Creep eine bestimmte Anzahl an Gold. Davon abgeleitet wird der sogenannte Creep-Jack, (englisch to jack „überfallen“) eine Situation, in der ein anderer Spieler einen gerade „creependen“ Spieler überrascht und dieser plötzlich gegen zwei Gegner kämpfen muss, Creep-Jacks werden auch Ganker genannt. Creepen steht vor allem neben „Crouchen“ und „Crawlen“ für das permanente Kriechen während einer Runde

[2] (englisch für Panzer) Ein Charakter in einem Spiel, der über extrem viel Rüstung, Leben oder Lebensregeneration verfügt und den gegnerischen Schaden auf sich lenkt, damit die restlichen Spieler ungestört angreifen können. Diese Taktik wird auch als Tanken bezeichnet.

3. Spielfeld

   Das Spielfeld besteht aus einem Pfad, Anfangspunkt, Endpunkt und eine Navigationsleiste. Die Creeps tauchen am Anfangspunkt auf und laufen den Pfand entlang bis zum Endpunkt, siehe Table 2. Die Türme können aus der Navigationsleiste beliebig auf dem Spielfeld gezogen werden. Nach einer Anzahl von erfolgreichen Verteidigung, verändert sich das Spielfeld.

4. Level-Konzepte

   Jedes Level besteht aus eine feste Spielfeldmuster und eine Abfolge von verschiedenen Creeps. Das Level steigt, falls alle Wellen erfolgreich verteidigt wurde. Jedes Level hat ein eigenes Spielfeld, sowie eine Abfolge von selben oder verschiedenen Arten von Creeps. Das Level-Konzept wird von vorne gestartet, sobald der Spieler die Invasion der Schlümpfe nicht besteht.

5. Geld

   Das Geld steigt linear an, falls ein Turm die Creeps tötet.

6. Lebenspunkte

   Zur Beginn des Spiels werden die Lebenspunkte standart auf 10 gesetzt. Sobald ein Creep durchläuft, verliert der Spieler ein Leben. Sobald die Lebenspunkte auf 0 sinkt, ist das Spiel beendet.

7. Spielpausen - und Spielfortsetzungsknopf

   Es besteht die Möglichkeit das Spiel zu stoppen, die auf der Navigationsleiste sich befindet. Falls der Spieler strategisch überlegen muss, wo er die Türme platzieren kann. Das Spiel wird dann wieder fortgesetzt, wenn die Spielfortsetzungsknopf gedrückt wird.

8. Charakterinformation

   Die Charakterinformation zeigt die Eigenschaften des Turmes an wie schnell, stark und weit der Angriff ist. Beim Klicken des Turmes werden die jeweiligen Eigenschaften im Navigationsleiste angezeigt.

9. Gegneranzahl

   Die Gegneranzahl gibt den Spieler die Information an, wie viele Creeps auf der Karte sind.

4 Architektur und Implementierung

4.1 Model

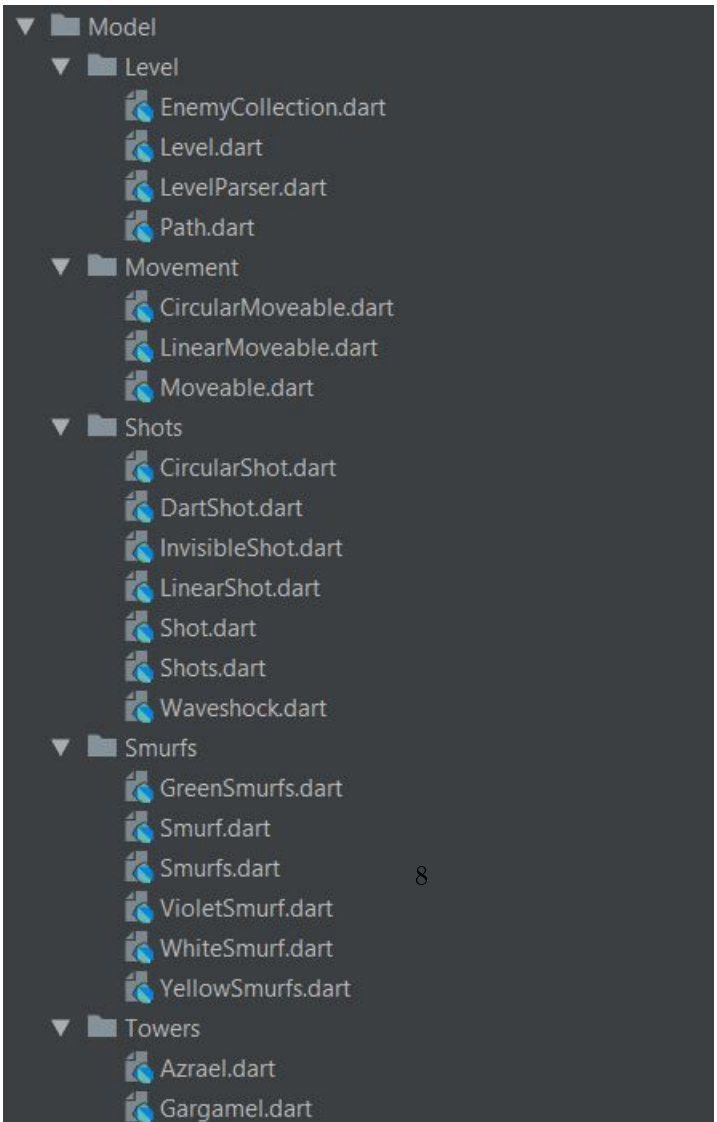
Das Model besteht aus den foldenden Klassen:

Tabelle 1: Die Navigationsleiste (Navi) beinhaltet die Türme (T), Türmeninformation (Ti), Leben (L), Geld (M), Menge an Creeps (E) und Start/Pauseknöpfe (P/S).

				Navi
				T
				L
				M
				E
				P/S

Tabelle 2: Die Creeps werden am Startpunkt (a) platziert und rennen den Pfad (-) entlang bis zum Endpunkt (e).

a-	-	-	-e	
a-	-	-	-e	



4.1.1 Erstellen einer Spielinstanz

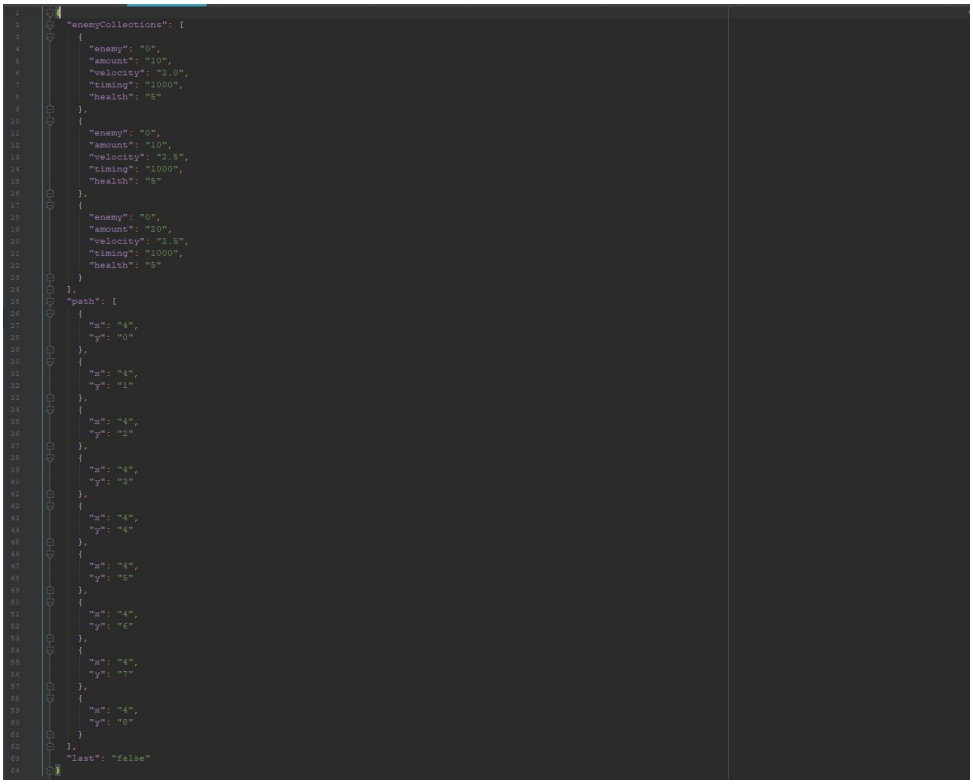
Beim starten eines neuen Spiels, also beim Laden der Website, wird eine neue Instanz der Game Klasse erzeugt. Diese erzeugt sich dann die weiteren Instanzen des Models.

4.1.2 Laden von Levels

Beim starten eines Levels, wird in einer Instanz der "LevelParser" Klasse eine neue Levelinstanz erzeugt. Die nötigen Informationen dafür, werden aus JSON Dateien geladen.

4.1.3 Aufbau der Leveldateien (JSON)

Beispielhaft die Leveldatei des ersten Levels:



- Die Abfolge der Gegner des Levels werden in Form eines JSON Arrays abgebildet. Jedes JSON Objekt in diesem Array enthält dabei unter anderem Informationen über den Gegnertyp, ihre Geschwindigkeit und wie schnell nacheinander die Gegener erscheinen. Als Gegnertyp wird der Index des Gegners im Enum "Smurfs.dart" angegeben.

- Nach der Abfolge der Gegener wird der Pfad angegeben. Dieser wird als JSON Array angegebn. die JSON Objekte in diesem Array enthalten einen x- und einen y-Wert. Der erste im JSON Dokument angegebene Punkt wird als Startpunkt interpretiert, der letzte als Endpunkt.
- Als letztes wird noch angegeben, ob das aktuelle Level das letzte Level ist.

4.2 View

Die View besteht aus zwei Teilen:

1. View.dart
2. Popup.dart

In View.dart werden alle darzustellenden Informationen des Modells mittels DOM-Tree manipulation dargestellt. Dazu gehören:

- Bewegung von Gegnern
- Bewegung von Schüssen
- Darstellung der Platzierung von Türmen
- Darstellung der Navigationsleiste:
  - Darstellung der Informationen wie die aktuelle Anzahl an Leben oder Geld
  - Darstellung der Türme, beziehungsweise Ihrer Informationen

In Popup.dart werden alle Arten con Popups beschrieben, dazu gehören:

- Welcome
- Level Start
- Game Over
- Wave Start

4.3 Controller

Der Controller besteht aus “Controller.dart”. In díeser Klasse werden alle Nutzeraktionen entgegengenommen und an das Model beziehungsweise die View weitergegeben. Zu diesen Aufgaben gehören

- Drücken des Play Buttons
- Drücken des Pause Buttons
- Drag-Start, Drag-End, Drag events um die Türme zu platzieren (Desktop)
- Touch-Start, Touch-End, Touch events um die Türme zu platzieren (Mobile)
- Drücken der Knöpfe in den Popups

5 Nachweis der Anforderungen

5.1 Nachfolgend wird erläutert wie die in Kapitel 2 eingeführten funktionalen Anforderungen, die Dokumentationsanforderungen und die technischen Randbedingungen erfüllt bzw. eingehalten werden. Abschließend wird angegeben, wer im Team welche Verantwortlichkeiten hatte.

Tabelle 3: Nachweis der Anforderungen

Id	Kurztitel	Erfüllt	Teilw. erfüllt	Nicht erfüllt	Erläuterung
AF-1	Einplayer Game	x			Smurfs Gone Wild ist ein Einpersonen Spiel, wie aus dem in Abschnitt 3 dargestellten Spielkonzepts hervorgeht.
AF-2	technischer Komplexität und Spielkonzept	x			Smurfs Gone Wild basiert auf einem Spielfeld mit absoluter Poitionierung, für Schüsse, Schlümpfe werden Positionen und Kollisionen berechnet, die Spielmechnik ist durch Platzieren der Türme noch komplexer.
AF-3	DOM-Tree basiert	x			Smurfs Gone Wild ist rein DOM-Tree aufgebaut ohne Hilfe von Canvas.
AF-4	Target Device: Smart Phone	x			Smurfs Gone Wild ist nur auf Handy spielbar. Das Spiel kann nur auf Landscape gespielt werden.
AF-5	Mobile First Prinzip	x			Das Spiel ist auf Android und IOS spielbar.
AF-6	schnell und intuitiv erfassbar und Spielfreude	x			Smurfs Gone Wild ist wie ein typisches Tower Defense aufgebaut. Die Spielfreude

5.2 Nachweis der Dokumentationsanforderungen

5.3 Nachweis der Einhaltung technischer Randbedingungen

5.4 Verantwortlichkeiten Im Projekt

Tabelle 4: Nachweis der Dokumentationsanforderungen

Id	Kurztitel	Erfüllt	Teilw. erfüllt	Nicht erfüllt	Erläuterung
D-1	Dokumentationsvorlage	x			Vorliegende Dokumentation dient als Vorlage für Spieldokumentationen.
D-2	Projektdokumentation	x			Vorliegende Dokumentation erläutert die übergeordneten Prinzipien und verweist an geeigneten Stellen auf die Quelltextdokumentation.
D-3	Quelllltextdokumentation		x		Es wurden alle Methoden und Datenfelder, Konstanten durch Inline-Kommentare erläutert.
D-4	Libraries	x			Alle genutzten Libraries werden in der pubspec.yaml der Implementierung aufgeführt. Da nur die zugelassenen Pakete genutzt wurden, sind darüber hinaus keine weiteren Erläuterungen, warum welche Pakete genutzt wurden, erforderlich.

Tabelle 5: Nachweis der Einhaltung technischer Randbedingungen

Id	Kurztitel	Erfüllt	Teilw. erfüllt	Nicht erfüllt	Erläuterung
TF-1	No Canvas	x			Die Darstellung des Spielfeldes sollte ausschließlich mittels DOM-Tree Techniken erfolgen. Die Nutzung von Canvas-basierten Darstellungstechniken ist explizit untersagt. Die Klasse Smurfs Gone Wild nutzt keinerlei Canvas basierten DOM-Elemente.
TF-2	Levelformat	x			Smurfs Gone Wild hat ein Level-Konzept
TF-3	Parameterformat	x			Das Spiel ist durch Level Dateien erweiterbar
TF-4	HTML + CSS	x			Der View des Spiels beruht ausschließlich auf HTML und CSS
TF-5	Gamelogic in Dart	x			Die Logik des Spiels ist mittels der Programmiersprache Dart realisiert worden.
TF-6	Storagelogic in Dart			x	
TF-7	Storage Referenz			x	
TF-8	Storage Referenztest			x	
TF-9	Browser Support	x			Das Spiel wurde in den Browsern Safari, Firefox und Chrome getestet. Ebenfalls wurde mit den mobilen Browsern von Android und IOS getestet.
TF-10	MVC Architektur	x			Das Spiel folgt durch Ableitung mehrerer Modell-Klassen (vgl. lib/src/model.dart), einer View Klasse (vgl. lib/src/view.dart) und dem zentralen Controller (vgl. lib/src/controller.dart) einer MVC-Architektur. Der GameKey Service (vgl. lib/src/gamekey.dart) wird ebenfalls diesem Prinzip unterworfen und wird nur vom Controller getriggert, genauso wie das Modell und der View. Der View greift zudem auf das Model nur lesend und nicht manipulierend zu.
TF-11	Erlaubte Pakete	x			Es sind nur erlaubte Pakete, dart:* packages sowie das Webframework start genutzt worden. Siehe pubspec.yaml der Implementierung.
TF-12	Verbotene Pakete	x			Es sind keine Pakete, außer den erlaubten genutzt worden. Siehe pubspec.yaml der Implementierung.
TF-13	No Sound	x			Das Spiel hat keine Soundeffekte.

Tabelle 6: Projektverantwortlichkeiten

Komponenten	Detail	Asset	Torben Rogge	Vu Nguyen
Model	Game	web/dart/Model/...	V	U
	Level		V	U
	Movement		V	U
	Shots		V	U
	Smurfs		V	U
	Tower		V	U
View	HTML-Dokument	web/styles/index.html	U	V
	Gestaltung	web/styles/style.css	U	V
		web/img	U	V
Controller	Eventhandling	web/dart/Model/Controller.dart	V	U
Dokumentation	Smurfs Gone Wild	doc/*.lyx	U	V

V = Verantwortlich

U = Unterstützend