Projekt Cat'n'run

Projektbeschrieb

In diesem Projekt wird ein Android Spiel auf Basis von Webtechnologien realisiert. Das Projekt wird an der BFH im Modul BTI7301 als Gruppenarbeit im Herbstsemester 2016/17 durchgeführt. Als Game wird ein *Jump and Run* realisiert. Zusätzlich zum Game soll auch ein Map-Editor erstellt werden, mit dem sich die Karten für das eigentliche Game zusammenstellen lassen.

Das Ziel dieses Projekts besteht darin, dass die Projektmitglieder Erfahrung in einem selbständig erarbeiteten und geführten Projekt sammeln können, sowie ein funktionierendes und amüsantes Android Game entwickeln.

Nebst Arbeit am Endprodukt (dem Game) selbst, gehört dazu auch die Terminplanung, das Erarbeiten von Projektdokumenten und allgemeine Dokumentationen rund um das Game (inklusive Code Kommentaren).

Projektteam

Das Projektteam besteht aus zwei gleichwertigen Projektmitgliedern: * Daniel Schmitz, Freelancer * Natalie Fioretti, Sysadmin

Daniel bringt langjährige Erfahrung aus dem Webprogramming mit, wohingegen Natalie wenig Erfahrung in der Programmierung besitzt.

Auftraggeber ist Herr Jürgen Eckerle.

Hilfsmittel

CSS

Um effizienter CSS-Dokumente zu schreiben, wird Sass verwendet und mittels Gulp-Transpiler zu CSS verwandelt.

Build-Prozess

Der Build Prozess wird mittels TypeScript-Transpiler grösstenteils erledigt. Als Packaging Software für diverse Dependencies und Versionierungen wird NPM (Node Package-Manager) verwendet.

Falls ein Backend-Server für den Map-Editor gebraucht wird, so wird dieser in NodeJS geschrieben.

Lazy-Loading

Zum Laden der benötigten JavaScript Libraries wird **SystemJS** eine JavaScript Library verwendet. Diese lädt die Dateien sobald sie benötigt werden und verringert so lange Ladezeiten & Performanceprobleme.

Assets

Da keiner der Teammitglieder über gute Design-Fertigkeiten verfügt, wird auf professionelle Assets von Unity zurückgegriffen.

Z.B. https://www.assetstore.unity3d.com/en/#!/content/31266

Charaktere und Sprites werden ähnlich organisiert.

Komponenten

Spiel

Die Teammitglieder haben sich für ein klassisches *Jump and Run* entschieden. In einem *Jump and Run* navigiert der Benutzer einen Charakter durch eine Welt. Dabei muss der Charakter oft über Hürden hinweg springen, auf andere Plattformen hochspringen und Gegnern ausweichen. In manchen Implementationen kann der Charakter Gegner auch angreifen und besiegen. Ziel des Spiels ist es, das andere Ende der Karte heil zu erreichen.

Cat'n'Run unterscheidet sich von anderen *Jump and Run* Spielen vor allem durch den spezielleren Charakter: Der Benutzer spielt eine Katze. Navigiert werden soll die Katze mit den Daumen auf dem Touchscreen, ähnlich den auf Konsolen bekannten Joysticks. Als Implementation im Webbrowser könnten die Pfeiltasten oder die bewährte WASD -Tastenkombination verwendet werden.

Features

Das Game kennt im Endzustand folgende Features:

- Katze als Spielcharakter, die hüpfen und sich ducken kann
- Gegner, die den Spieler angreifen und tödlich verletzen können. Es ist noch zu definieren, ob die Gegner fest auf der Karte einprogrammiert werden, oder ob sie dynamisch im oder vor dem Spielverlauf generiert werden
- Bossfights
- Gegner können angegriffen werden, indem der Spieler auf deren Kopf springt. Eventuell könnten auch Wurfgegenstände implementiert werden, um den Gegnern Schaden zuzufügen.
- Sammelgegenstände, wobei noch zu definieren ist, ob diese einfach Punkte geben, oder die Spielzeit zum Beenden der Karte erhöhen
- Karten laden, die der Spieler im Editor selbst erstellt hat

Map-Editor

Beschreibung

Die Teammitglieder haben sich entschieden, dass es langfristig sinnvoll ist, den Prozess zur Erstellung einer Karte mittels eines Map-Editors zu optimieren. Der Editor beinhaltet diverse graphische Elemente und soll ausschliesslich im Webbrowser verwendet werden.

Der Map Editor wird nicht für mobile Geräte optimiert. Grund dafür ist der nicht geeignete Verwendungszweck für kleinere Geräte.

Der Map Editor soll als Resultat ein JSON-File liefern, welches entweder lokal auf dem Computer abgespeichert wird, oder auf einem Webserver gesichert wird.

Designtechnisch soll der Map-Editor zum Arbeiten einladen. Allerdings liegt der Fokus auf dem Spiel und nicht dem Editor.

Features

Der Editor kennt folgende Features:

- Folgende auswählbare Elemente
  - Hintergründe
  - Charaktere (Gegner / Spielelemente)
  - Map-Elemente (Begehbarere Grund)
  - Dekorationen (rein optische Elemente)
- Möglichkeit die Karte zu vergrössern oder verkleinern
- Vergabe einfacher Properties an Gegner (Charakter-Elemente)
- Erstellen eines JSON Files (mit allen Karteninhalten)
- Erfassen von neuen Elementen (Map / Charakter / Dekorationen / Hintergründe)
- Abspeichern der JSON-Dateien aus einem Server
- Möglichkeit drei verschiedene Ansichten anzuzeigen. Dies hält den Map-Editor auch für komplexere Karten übersichtlich.
  - Grid-Ansicht: Zeigt ein Grid an, in welchem die Map-Elemente platziert werden können.
  - Dekorationsansicht: Zeigt nur den Hintergrund und Dekorationselemente an (wird gebraucht um die Map optisch ansprechend zu gestalten)
  - Charakter Ansicht: Zeigt die Charaktere an, sowie die Map-Elemente, blendet aber jegliche Dekoration aus.

Mesh Collider Editor

Dieses Feature wird umgesetzt, falls genügend Zeit vorhanden ist.

Beschreibung

Unter Umständen ist es nötig einen Mesh-Collider-Editor zu entwickeln. Dieser sollte mit einfachem GUI dem User helfen neue Gegenstände und Charaktere zu erfassen. Der dadurch zu erstellende Mesh-Collider sollte dem Element zugewiesen werden können und direkt im JSON (vom Map-Editor) integriert werden.

Mittels Mesh-Collider soll präziser (als im üblichen Rechteck) ermittelt werden, ob der Spielercharakter / Gegner getroffen wurde.

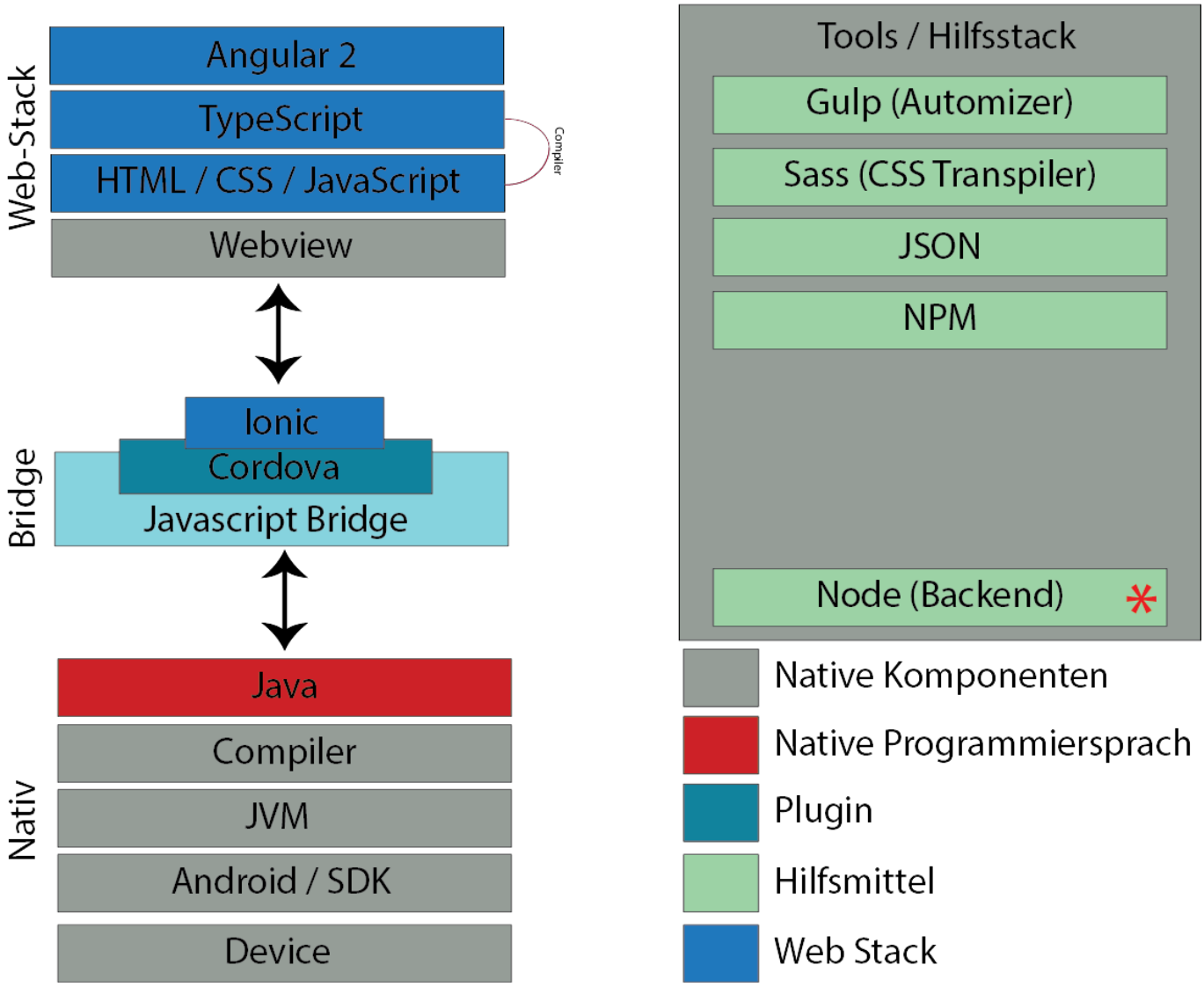
Features

- Einfaches GUI zum Erstellen eines Mesh-Colliders
- Integration im Map-Editor
- Integration des Mesh-Colliders im Spiel zur optimierten Treffererkennung

Umgebung und Technologien

Das Game soll für Android Geräte entwickelt werden. Um flexibler und mit effektiveren GUI Elementen zu arbeiten, wird das Spiel hauptsächlich in der Android Webview laufen. Dementsprechend werden diverse Webtechnologien verwendet.

Die zu verwendenden Technologien sind auf diesem Graphen dargestellt. Die unteren Schichten stellen gerätenahe Technologien dar.



Als native Gerätesprache wird **Java** verwendet. Um die Kommunikation zwischen der **Hauptkomponente (Webview)** und dem Android Gerät sicherzustellen wird eine **JavaScript Bridge** geschrieben. Dies geschieht mittels **Ionic (als Hybride-Framework)** und **Cordova Plugins** als bestandteil der **JavaScript Bridge**.

Die WebView beinhaltet die **Spielelement**. Diese Spielelemente werden als **HTML-Elemente**

abgebildet.

Das GUI wird mittels **CSS** optisch dem Spiel entsprechend angepasst.

Sämtliche Spiellogik wird in **TypeScript** geschrieben, welches ein **JavaScript** superset darstellt und zu ausführbarem **JavaScript** kompiliert wird.

Als Frontend Framework wird **Angular 2** verwendet, welches für Typescript optimiert ist.

Die Maps werden im **JSON Format** gespeichert und werden lokal auf dem Device gespeichert. Zur Aufbewahrung der Maps werden diese ggf. auf einem Webservice gespeichert (**NodeJS**).

Technologiewahl

Wir verwenden diese Technologien, damit unser Spiel platformunabhängiger läuft und um den Programmieraufwand zur portierung auf andere Systeme (iOS/Browser) möglichst gering zu halten.

Des Weiteren ermöglicht die Wahl uns, den Buildaufwand bis möglichst ende des Projekts klein zu halten. Zu guter Letzte erhöht sich mit diesem Technologiestack die Produktivität und beschleunigt den Debug-Prozess deutlich. Grund dafür ist, dass tests in den Browsern durchgeführt werden können und so aufwändige Gerätesimulatoren wegfallen.

Des Weiteren sind bei Daniel Schmitz fundierte Kenntnisse im Bereich Angular-Programming vorhanden und können mit dem Spiel vertieft werden.

Natalie Fioretti profitiert zudem von den neuen Kenntnissen eines populären Technologiestacks, welcher für die Zukunft immer relevanter wird.

Terminplanung

Semesterwoche	Soll Projektstand	Ist Projektstand
1	Projekt ausgewählt	Projekt ausgewählt
2 - 3	Projektideen gesammelt, erste Gedanken über Technologien gemacht, Projekt vorgestellt	Projektideen gesammelt, erste Gedanken über Technologien gemacht, erste Kontakte mit neuen Technologien, Projekt vorgestellt
4	Projekt-Setup & Anpassungen Projektdefinition	-
5-7	Arbeit am Projekt (Fokus: Map-Editor)	-
8	Erstellung der AI der Gegner (Map-Awareness)	-
9-10	Arbeit am Projekt (Fokus: Android Spiel) - JSON-File interpretieren und korrektes laden & darstellen	-
10-13	Arbeit am Projekt (Fokus: Android Spiel) - Optimieren des Spieles / Map-Editores	-
13-15	Portierung auf Android WebView & Integration Cordova Plugins inkl. Ionic	-
16	Projektpräsentation	-

Abgrenzungen

- Das Spielt ist:
  - Kein Shooter
  - Nicht in 3D
- Keine WegGL-Library wird verwendet
- Assets werden nicht durch uns erstellt, sondern gekauft oder je nach Lizenz verwendet
- Das Spiel soll primär auf Android Geräten laufen (später kann dies auch für den Browser / iOS optimiert werden)