RMA: 2D-Adventure

René Ruprecht

19. Januar 2022

Inhaltsverzeichnis

1 Einleitung

In dem Modul soll ein 2D-Multilevel Spiel mittels Javascript entwickelt werden. Das Spielkonzept ist eine Art Zelda Klone. In dem Spiel wird es einen spielbaren Charakter geben, der sich durch mehrere Level bewegen kann. In den verschiedenen Level wird es dann Gegner geben, die der spielbare Charakter besiegen kann. Für jeden besiegten Gegner wird der Spieler mit Punkten belohnt. Sollte der Spieler von einem Gegner getroffen werden, dann wird diesem ein Leben angezogen. Ein Spieler wird eine vorher definierte Anzahl von Leben besitzen. Sind die Leben aufgebraucht, so wechselt das Spiel in die "Ende" Szene und die erreichte Punktzahl wird angezeigt. Die Punkte berechnen sich aus besiegten Gegnern sowie verbleibende Leben. Pro verbleibendes Leben gibt es einen extra Punkt.

2 Planung

2.1 Minimal Kriterien

In dem Spiel sollen folgende Kriterien abgedeckt werden:

- 2 Charaktere mit Animationen
- mehr als ein Level
- Start- / Endszene
- Möglichkeit, Neustart des Spiels

2.2 Erweiterte Kriterien

Neben den vorgegebenen Anforderungen habe ich mich für folgende “kann“ Kriterien entschieden:

- Items im Spiel vom Spieler aufhebbar (z. B. Waffen)
- Anzeige des Lebens vom Spielercharakter
- Anzeige des Punktestands
- Hintergrundmusik
- Soundeffekte
- mehrere Gegner mit verschiedenen Eigenschaften (z. B. geben mehr Punkte, bewegen sich schneller, machen mehr Schaden)
- Items mit verschiedenen Eigenschaften
- Boss Gegner

2.3 Programmiersprache

Da das Spiel innerhalb des Browsers spielbar sein soll, habe ich mich für Javascript entschieden. Javascript wird ohne weitere Installationen in jedem Browser unterstützt.

2.4 Framework/Bibliothek

Um nicht jede Funktion vom Grund auf zu implementieren, wird ein Framework oder eine Bibliothek benötigt. Nach einiger Recherche habe ich mich hierbei für KaboomJS entschieden. Diese Bibliothek bringt eine Vielzahl von Funktionen mit, die für die Umsetzung benötigt wird. Hierzu zählen unteranderem das Animieren von Spritesheets oder das Abspielen von Musik.

2.5 Sprites

Für das Spiel werden Sprites benötigt. Sprites sind Grafikobjekte, die geladen werden können. Ich habe mich für ein Spriteset von 0x72 entschieden. Das Spriteset bringt unteranderem eine Vielzahl von Levelelementen sowie Charaktere, NPCs und Gegenstände mit.

3 Spielaufbau

Das Spiel ist so aufgebaut, das die Kamera immer dem Spielercharakter folgt. Ein User-Interface wird außerhalb des Levels angezeigt. Dies bietet die Möglichkeit, dass die spielende Person nicht immer genau weiß, wie viele Leben noch verfügbar sind und erhöht etwas den Schwierigkeitsgrad. Die Anzeige der verbleibenden Leben sowie der Punktezähler werden sich ständig nebeneinander an der oberen Seite des Levels befinden. Außerhalb von Menüs spielt eine ambient Musik. Im ersten Level hat der Spieler die Möglichkeit, sich ein Schwert als Waffe zu nehmen. Das Schwert hat die Eigenschaft, das dieses schnell zuschlagen kann mit einem geringen Schadenswert. Es gibt drei verschiedene Gegnertypen. Alle drei Gegnertypen haben verschiedene Eigenschaften wie z. B. wie viel Schaden diese dem Spieler zufügen, wie viele Punkte diese geben oder wie schnell diese sich bewegen. Wenn der Spieler eine Attacke ausführt, so wird ein Sound abgespielt. Trifft der Spieler einen Gegner, so wird ein Sound abgespielt, um den Treffer zu signalisieren. Wird der Spieler von einem Gegner getroffen, so wird auch hier ein Sound abgespielt, um den Treffer zu signalisieren. Auf der zweiten Ebene liegt am Ausgang eine weitere wählbare Waffe, ein Hammer. Der Hammer braucht mehr Zeit als das Schwert, bis dieser erneut zuschlagen kann, macht dafür aber mehr Schaden und wirft die Gegner zurück. Das Spiel endet, sobald der Spieler keine Leben mehr hat oder das letzte Level durchlaufen wurde. Es muss kein Gegner besiegt werden, man kann das Spiel auch als Pazifist durchspielen und lediglich mit den verbleibenden Herzen die Punkte bekommen.

4 Starten der Anwendung

4.1 Anforderungen

- Git (optional)
- Docker
- Webbrowser
- Monitor Auflösung 1920x1080

4.2 Installation

Als Erstes muss das Repository heruntergeladen werden. Danach ist es am einfachsten, wenn man die beigefügte docker-depl Datei builden lässt. Dies funktioniert mittels docker und dem Command „docker build -f docker-depl . -t rma-2d-adventure“ innerhalb des heruntergeladenen Verzeichnisses. Danach wird der Container ausgeführt mit dem Command „docker run -p 4567:80 –name rma-2d-adventure rma-2d-adventure“. Danach kann man über dem Webbrowser auf die localhost:4567 zugreifen und spielen. Alle Commands werden ohne Anführungszeichen ausgeführt.

4.3 Tastenbelegung

- Pfeiltasten zum bewegen
- E zum Item aufheben und Ebenen wechseln
- Leertaste zum Waffen einsetzen

5 Spielablauf

5.1 Erste Szene

Auf der ersten Szene wird eine kurze Erklärung für die Tastenbelegung angezeigt. Auf dieser Szene gibt es die Möglichkeit, dass man auf die Game-Szene wechseln kann.

5.2 Game-Szene Ebene 0

Innerhalb der Game-Szene wird dann die erste Ebene geladen. Hierzu zählen die grafischen Elemente der Ebene,
platzieren des Spielercharakters, platzieren der Items und anzeigen des User-Interfaces (Leben, Punktestand). Der Spielercharakter hat auf der ersten Ebene die Möglichkeit, sich ein Schwert zu nehmen. Der Spielercharakter kann dann über eine klar ersichtliche Möglichkeit in das nächste Ebene wechseln.

5.3 Game-Szene weitere Ebenen

Ab der Ebene 1 hat der Spielercharakter die Möglichkeit, gegen Gegner zu kämpfen. Die Gegner sind so eingestellt, dass diese sich immer auf die Position vom Spielercharakter zubewegen. Der Spielercharakter hat immer die Möglichkeit, der Ebene über eine klar ersichtliche Position innerhalb der Ebenen zum nächsten zu wechseln.

5.4 Spielende

Das Spiel endet, sobald die Leben des Spielercharakters aufgebraucht oder die letzte Ebene durchlaufen worden ist. Danach wird die Szene gewechselt und die erreichte Punkteanzahl angezeigt. Hier gibt es dann die Möglichkeit, das Spiel von erneut zu starten oder das letzte Ebene zu wiederholen.

6 Entwicklung

6.1 Verwendete Tools

- VSCode, Entwicklungsumgebung
- Parcel, Bundler
- Webbrowser, Chrome
- Docker

6.2 Entwicklungsablauf

Als Entwicklungsumgebung wurde VSCode verwendet. Die Bibliothek KaboomJS wurde mittels des Node Package Managers dem Projekt hinzugefügt. Damit die Bibliothek während der Entwicklung Live getestet werden kann, wurde ein Bundler verwendet. Ich habe mich für Parcel als Bundler entschieden, da dieser einfach und sehr schnell zu integrieren ist. Parcel bietet die Möglichkeit, das die Anwendung nicht immer erneut lange compilen muss, dies geschieht sehr effizient. Es reicht aus, lediglich die Webanwendung erneut zu laden. Da die Anwendung auf einem Linux-Server laufen soll, wird die Anwendung nach jedem größeren Feature auf einer Docker-Instanz getestet. Docker bietet die Möglichkeit, eine Linux-Server Umgebung zu erstellen, worauf die Anwendung dann getestet werden kann. Dies ist bei der Entwicklung auf Windows-System sehr ratsam, da es innerhalb meiner Entwicklungszeit Probleme mit den Pfaden gab.

6.3 Ordnerstruktur

- assets, Grafiken und Audio Content
- components, Userinterface sowie Listener
- config, jegliche Konfigurationen
- helper, Funktionen die z. B. das Level erstellen
- objects, Elemente wie Spieler, Gegner und Waffen
- scenes, die einzelnen Scenen

7 Probleme / Lösungen

7.1 Pfad Probleme

Während der Entwicklung gab es unteranderem Probleme mit den Pfaden. Es wurden z. B. Assets nicht gefunden, obwohl diese über die Entwickler-Optionen in Chrome sichtbar waren. Abhilfe hat es geschafft, die Anwendung auf eine Docker-Instanz mit einem Linux-Webserver zu kopieren und diese da Final zu testen, bevor diese hochgeladen wurde.

7.2 Fehlende Events

Die Bibliothek bietet nicht die Möglichkeit, über ein Event zu überprüfen, ob sich zwei Objekte berühren. Dies wird z. B. dafür benötigt, wenn ein Gegner innerhalb des Spielers steht, dass dieser mehrmals Schaden zufügen soll. Es musste also eine separate Funktion geschrieben werden die überprüft ob sich der Spieler mit dem Gegner überlagert. Das Gleiche musste auch für Items gemacht werden. Es gibt ein onCollide Event, aber hierbei könnte der Spieler das Item auch dennoch aufheben, wenn dieser nicht mehr über diesem steht.

7.3 Kollisionen unter den Gegner

Ein weiteres Problem waren die Kollisionsboxen. Die Gegner konnten durch Wände laufen, da dies nicht gewollt war, mussten die Gegner als “solide“ Entität gekennzeichnet werden. Nun kann der Spieler aber nicht mehr durch diese durchlaufen. Dazu kam, dass es passieren konnte, das sich die Gegner verkantet haben und sich somit nicht mehr bewegen konnte. Es musste also eine Möglichkeit gefunden werden, dass die Gegner nicht mehr als solide

Entität zählten, sobald diese sich gegenseitig berühren oder vom Spieler schaden erlitten haben. Es wird also nun überprüft, ob sich die Gegner berühren. Sollte dies der Fall sein, so wird von einem der Gegner die solide Eigenschaft entfernt und erneut gesetzt. Dies löste das Problem, das diese nicht mehr verkanteten. Wenn die Gegner schaden erleiden, wurde auch hinzugefügt, dass die solide Eigenschaft entfernt wird, solange diese transparent sind. So hat der Spieler die Möglichkeit, durch diese hindurch zu bewegen.

7.4 Bundler Probleme

Kurz vor der Abgabe ist mir aufgefallen, dass die Builds vom Projekt nicht korrekt ausgeführt worden sind. Dies fiel mir auf, als ich einen selbst erstellten Docker-Container fürs einfachere builden bauen wollte. Als ich dann über das Gitlab-Portal schaute, war auch da das Projekt nicht funktionsfähig. Die Builds schlossen allerdings immer ohne jegliche Fehlermeldung ab. Ich habe keine Erklärung gefunden, wieso dies von jetzt auf gleich nicht mehr ging. Ein erneutes clonen der Repository und das Installieren der package.json erzeugte noch viel mehr Probleme beim builden des Projektes. Die Lösung war es den Bundler auszuwechseln und den Code anzupassen. Als Schlussfolgerung ziehe ich daraus, dass es besser wäre, direkt in einem Container zu entwickeln, da so alle Abhängigkeiten an einem Ort installiert sind und auch zuverlässig funktionieren.

8 Erweiterungen

Mögliche Erweiterungen wären unteranderem das responsive Design. Dies ist leider derzeit durch die Bibliothek nicht gegeben. Des Weiteren kann das Spiel durch weitere spielbare Charaktere erweitert werden. Diese könnten unterschiedliche Starteigenschaften haben, wie z. B. einen anderen Lebenswert. Dazu könnten auch weitere Gegner hinzugefügt werden, ebenfalls mit verschiedenen Eigenschaften. Weitere Items könnten hinzugefügt und auf den Ebenen verteilt werden, hierbei wären Items, die z. B. ein Lebenspunkt wiederherstellen, denkbar. Weitere Tasten könnten ebenfalls belegt oder gar eine Unterstützung für alterative Eingabemöglichkeiten implementiert werden.

9 Quellen

- Spriteset: https://0x72.itch.io/dungeontileset-ii
- Music by Alex MakeMusic soft-ambient-10782 from https://pixabay.com/music/?utm_source=link-attribution&utm_medium=referral&utm_campaign=music&utm_content=10782