HOCHSCHULE FURTWANGEN UNIVERSITY HFU

Designdokumentation

in

Prototyping interactive media-applications and games

# Tower

Ein Prototyp


| | |
|---|---|
| Referent: | Jirka Dell'Oro |
| Vorgelegt am: | 16.02.2020 |
| Vorgelegt von: | Jonas Plotzky, 256539<br>Bilel Spannagel, 256661 |

# 1 Herangehensweise

## 1.1 Was war geplant

Von vornherein war uns klar, dass wir uns mehr mit dem technischen beschäftigen wollten, als mit dem visuellen, um uns von den Studenten in Digitalen Medien abzuheben. Somit war unser erstes Konzept ein Jump-and-Run Spiel, in dem man als Spielercharakter einen Turm von außen erklimmt, während man sich um diesen dreht. Dies wäre dann in einer 2,5D Ansicht gewesen, wobei der Turm 3D wäre aber man bewegt sich zweidimensional über die Oberfläche des Turms.

Abbildung 1: Turm aus “Upwards, Lonely Robot”

Je höher der Spielercharakter den Turm besteigt, desto schwerer würde das Spiel werden, zudem würde sich auch noch der Hintergrund ändern, am Anfang wären zum Beispiel ein Wald im Hintergrund, der dann von Bergen ersetzt wird, bis dann nur noch der Himmel sichtbar ist und zum Schluss sieht man dann die Sterne.

Als wir uns dann jedoch am Anfang mit den Basics beschäftigten wurde uns jedoch schnell klar, dass dieses 2,5D zwar möglich ist, aber viele Probleme mit sich bringt.

Zum einen brauchten wir dann dreidimensionale Modelle für Objekte und den Spielercharakter, während wir in 2D einfach Sprites verwenden konnten, zum anderen wurde die Collision detection weitaus komplizierter durch die Bewegung

im Kreis, weshalb wir uns schlussendlich dagegen entschieden haben.

### 1.2 Was haben wir gemacht

Somit kamen wir zum momentanen Spielkonzept. Wir wollten weiterhin ein pures Jump-and-Run Spiel machen, ohne Kampf oder Gegner, das aber leicht erweiterbar ist. Somit beschäftigten wir uns, nachdem wir mit der Steuerung und den Collisions weitestgehend fertig waren, mit der Objekt- und Levelgenerierung.

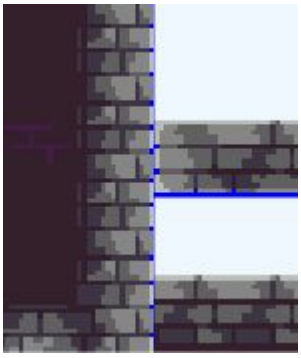

Abbildung 2: Frühes Leveldesign

Unsere Priorität lag weiterhin beim technischen Teil, weshalb wir uns Gedanken machten, wie wir das Level generieren lassen können, ohne uns dabei einzuschränken.

Zunächst hatten wir uns zwei ähnliche Konzepte überlegt:

Fest definierte Bausteine mit denen wir das Level aufbauen. Dem Levelgenerator müsste man dann nur noch die Reihenfolge dieser Bausteine übergeben. Die Abbildung 2 würde dann nach diesem Schema aufgebaut sein:

1,2,0,0
1,2,3,3
1,2,0,0
4,3,3,3

Diese Vorgehensweise brachte jedoch ein paar Probleme mit, zum Einen war das Spiel somit auf gleichgroße Ebenen limitiert, von denen man nicht abweichen konnte und zum Anderen musste man leere Blöcke auch mit einplanen und verarbeiten.
Der Vorteil jedoch wäre, dass das Design vereinfacht wäre.

Die Alternative dazu wäre, die Informationen für die Bausteine mitzugeben, wodurch deren Positionen leicht veränderbar wären und nur Bausteine die gesetzt werden müssten auch angegeben werden.

Die Abbildung 2 wäre dadurch nach diesem Schema aufgebaut:


Abbildung 3: Aufbau von level.json

Diese Ausführung bringt aber auch mehr Aufwand mit sich, da die Koordinaten von Hand bestimmt werden müssen.

Schlussendlich entschieden wir uns für die zweite Variante, weil wir dadurch Einschränkungen vermeiden konnten.

Nachdem die Levelgenerierung geklärt war, wollten wir diese natürlich mit verschiedenen Bausteinen füllen. Abhängig von dem Baustein sollte dann auch ein Sprite für dieses Baustein generiert werden, das unabhängig vom Baustein an sich ist, weshalb die Selbe Node Subklasse zum Beispiel für mehrere verwendet werden kann. Als Proof of Concept erstellten wir einen Windböengenerator der dynamisch Windböen erstellt, die in eine vorgegebene Richtung sich bewegen und den Spieler beeinflussen.

# 2 Spielkonzept

Tower ist ein Platformer und Jump'n'Run in dem der Spieler eine vermummte Spielfigur spielt, die versucht einen Turm zu erklimmen. Der Spieler muss immer schwerer werdende Sprünge meistern, während weitere Spielmechaniken langsam eingeführt werden. Die Spielumgebung ist harmlos, das heißt der Spieler kann nicht sterben; jegliche Interaktion beeinflusst nur die Bewegung und Sprünge des Spielers. Sollte der Spieler fallen, so muss er einen oder mehrere Sprünge erneut schaffen, es sollte jedoch Checkpoints geben, von denen der Spieler nicht weiter runterfallen kann, sodass er sicher ist sobald er diese erreicht.

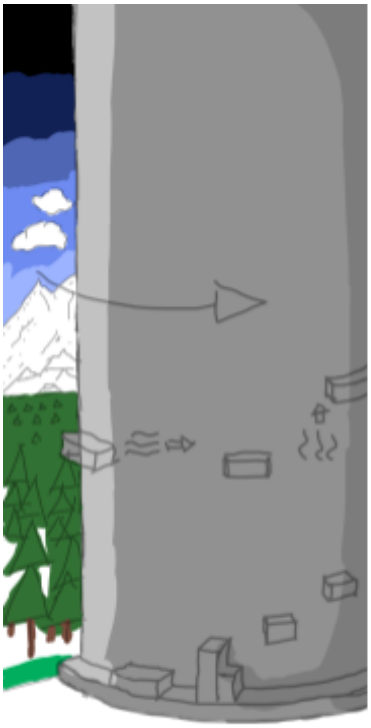

Abbildung 4. Erstes Spielkonzept

Der Spieler kann sich nach links oder rechts bewegen und springen. Der Spieler interagiert auch noch mit der Welt indem er Plattformen erklimmt oder von Windböen beeinflusst wird. Diese können den Spieler verlangsamen, beschleunigen, nach unten drücken oder helfen, höher zu springen als er es alleine kann.

Während der Spieler den Turm erklimmt ändert sich der Hintergrund um zu zeigen wie hoch der Spieler bereits ist.

## 2.1 Leveldesign

Das Leveldesign sollte zumindest für einen Prototypen darstellen, was alles mit den integrierten Bausteinen möglich ist. So wird ein Sprung den man ohne Probleme schaffen sollte durch eine Windböe aus der entgegengesetzten Richtung gestört. Im nächsten Moment muss man eine Windböe aus derselben Richtung aber verwenden um einen weiten Sprung zu schaffen, der ohne die Hilfe nicht machbar wäre. Diese beeinträchtigung von Sprüngen ist aber nicht nur seitwärts möglich sondern auch von

oben oder von unten was mit dem nächsten Sprung gezeigt wird. Zuletzt wird noch gezeigt, dass die Windböen helfen können, höher zu springen als alleine möglich wäre bevor das Level auch schon zu Ende ist. Diese Mechaniken kann man noch frei erweitern indem man sie mit dem Leveldesign und miteinander kombiniert

## 2.2 Klassendiagramm