

Fakultät 4 – Elektrotechnik und Informatik

**Medieninformatik 2**

**Projekt: Webgame mit Phaser.js**

Jump-Adventure: „Possible Game“

**Verfasser:**  Philipp Moritzer, 5034255

Hannes Lesemann, 5017055

Pascal Seegers, 5051429

**Modul:**  Medieninformatik 1

**Dozent:** Prof. Dr. Volker Paelke

**Labor:** Dipl.-Inf. Andreas Lochwitz

**Abgabedatum:** 28.06.2019

Inhalt

[1 Spielidee 1](#_Toc9525077)

[2 Handlung 1](#_Toc9525078)

[2.1 Story 1](#_Toc9525079)

[2.2 Interaktion 1](#_Toc9525080)

[3 Grafische Gestaltung 1](#_Toc9525081)

[3.1 Layout und Gestaltungsmerkmale 1](#_Toc9525082)

[3.1.1 Hauptmenü 1](#_Toc9525083)

[3.1.2 Level-Selektion 1](#_Toc9525084)

[3.1.3 Spiel 1](#_Toc9525085)

[3.1.4 Modal-Dialoge 1](#_Toc9525086)

[3.1.5 Inventar 1](#_Toc9525087)

[3.1.6 Button-Effekt 1](#_Toc9525088)

[3.1.7 Outro 1](#_Toc9525089)

[3.2 Schriften 1](#_Toc9525090)

[3.3 Icons 1](#_Toc9525091)

[3.4 Musik 1](#_Toc9525092)

[4 Verwendete Technologien 2](#_Toc9525093)

[4.1 Verwendete Sprachen 2](#_Toc9525094)

[4.2 Bibliotheken 2](#_Toc9525095)

[4.3 Editor und Tools 2](#_Toc9525096)

[4.4 Versionsverwaltung 2](#_Toc9525097)

[4.5 Konventionen 2](#_Toc9525098)

[5 Architektur 3](#_Toc9525099)

[5.1 Aufbau des Projektes 3](#_Toc9525100)

[5.2 index.html als Einstiegspunkt 3](#_Toc9525101)

[5.3 Seiten und Komponenten 3](#_Toc9525102)

[5.3.1 Seiten 3](#_Toc9525103)

[5.3.2 Komponenten 3](#_Toc9525104)

[5.4 Zentrales „State-Management“ 3](#_Toc9525105)

[5.4.1 GameStateManager 3](#_Toc9525106)

[5.4.2 UIManager 3](#_Toc9525107)

[5.4.3 SoundManager 3](#_Toc9525108)

[5.5 JSON-Daten 3](#_Toc9525109)

[5.6 Models 3](#_Toc9525110)

[5.7 Helfer 3](#_Toc9525111)

[5.8 Layout-Spezifisches 3](#_Toc9525112)

[6 Projektplanung und Arbeitsteilung 3](#_Toc9525113)

[6.1 Zeitlicher Ablauf und Arbeitsteilung 3](#_Toc9525114)

[7 Post Mortem 3](#_Toc9525115)

[7.1 Was hast gut funktioniert 3](#_Toc9525116)

[7.2 Was hat nicht funktioniert 4](#_Toc9525117)

[7.3 Rückblick 4](#_Toc9525118)

[7.4 Ausblick 4](#_Toc9525119)

[Anhang I](#_Toc9525120)

[i. Projektplanung I](#_Toc9525121)

[ii. Projektstruktur II](#_Toc9525122)

[iii. Tiled Map Building III](#_Toc9525123)

[iv. Tileset IV](#_Toc9525124)

[vi. Auszug Tilemap als .json IV](#_Toc9525125)

[iv. Collision debugging V](#_Toc9525126)

[iii. Desktop Menu Screenshots VI](#_Toc9525127)

[iv. Desktop Game Screenshots VII](#_Toc9525128)

[v. Mobile Ansichten IX](#_Toc9525129)

**Abbildungsverzeichnis**

[Abbildung 1: GANTT-Diagramm Planung I](#_Toc9525130)

[Abbildung 2: Projektstruktur II](#_Toc9525131)

[Abbildung 3: Erstellen von Tiledmaps III](#_Toc9525132)

[Abbildung 4: Tiled layers III](#_Toc9525133)

[Abbildung 5: Tiled Attribute IV](#_Toc9525134)

[Abbildung 6: Tileset IV](#_Toc9525135)

[Abbildung 7: Auszug Tilemap .json-Export V](#_Toc9525136)

[Abbildung 8: Collision Debugging V](#_Toc9525137)

[Abbildung 9: Desktop-Hauptmenü VI](#_Toc9525138)

[Abbildung 10: Dektop-Level-Select VI](#_Toc9525139)

[Abbildung 11: Stage 1 – Screenshot VII](#_Toc9525140)

[Abbildung 12: Stage 2 – Screenshot VII](#_Toc9525141)

[Abbildung 13: Stage 3 – Screenshot VIII](#_Toc9525142)

[Abbildung 14: Stage 4 – Screenshot VIII](#_Toc9525143)

[Abbildung 15: Menu-Screenshot Phone IX](#_Toc9525144)

[Abbildung 16: Select-Level Screenshot Phone IX](#_Toc9525145)

[Abbildung 17: Game Screenshot Phone X](#_Toc9525146)

[Abbildung 18: Game Screenshot Tablet X](#_Toc9525147)

# 1 Spielidee

* Plattformer mit einer Eingabemöglichkeit
* Schwierigkeit kommt über das perfekte Timing
* Gegen den üblichen menschlichen Rhythmus
* Fiese Obstacles und twists
* Anzahl an Versuche reduzieren
* Theming der Maps
* „Kleine“ Story
* Spiel für nebenbei / mobile

# 2 Handlung

## 2.1 Story

- Astronaut

- Zürck zum Mond

- mehr noch unsicher

## 2.2 Interaktion

- Drücken einer einzelnen Pfeiltaste

- oder Touch-input

# 3 Grafische Gestaltung

## 3.1 Layout und Gestaltungsmerkmale

### 3.1.1 Hauptmenü

Für Screenshots der Intro-Seite siehe im Anhang ii. Screenshots - Intro.

### 3.1.2 Level-Selektion

Für Screenshots der Charakter-Selektions-Seite siehe im Anhang iii. Screenshots - Charakterauswahl.

### 3.1.3 Spiel

Für Screenshots der Game-Seite siehe im Anhang iv. Screenshots – Game.

## Parallax

## 3.3 Mobile Ansichten

Für Screenshots der mobilen Ansicht siehe xyz

Responsive

## 3.4 Schriften

Im Spiel werden vier unterschiedliche Schriftarten verwendet. Diese sind Arcade Classic, Zorque, Arial und Helvetica. Während Arcade Classic einen gewissen nostalgischen Effekt mit sich bringt

## Tiles / Sprites

* „Pixel“-Art
* Theming durch verschiedene Enviromental-Tiles
* 16x16
* Astronaut

## Musik

* Passende Musik
* Hills
* Desert
* Moon
* Ice
* Jump-Sound
* Lose-Sound

# 4 Verwendete Technologien

## 4.1 Verwendete Sprachen

Für dieses Projekt werden die drei gängigen Sprachen in der Webentwicklung verwendet: HTML, JavaScript und CSS. Die Sprachen wurden in ihrer Grundform genutzt und es werden keine Abwandlungen, wie TypeScript oder SASS verwendet. Der Anteil von HTML und CSS ist sehr gering, da die Elemente über die Game-Engine Phaser geregelt werden, welche es erlaubt die Implementierung fast ausschließlich in JavaScript vorzunehmen. Es muss lediglich ein Container und die Rahmenbedingungen der Spielszene in HTML und CSS umgesetzt werden. Es werden Teile der ES6-Syntax verwendet, beispielsweise Arrow-Funktionen.

## 4.2 Bibliotheken

Als einzige Bibliothek wurde Phaser verwendet. Phaser ist ein Framework, welches zur Entwicklung von Spielen in HTML5 dient. Als Renderer wird WebGL verwendet.

## 4.3 Editor und Tools

Als Editor wurde Visual Studio Code mit folgenden Plugins verwendet:

* Live Server
* Prettier
* Debugger for Chrome

Hauptsächlich wurde die Applikation auf Google Chrome getestet, von Zeit zu Zeit auf Mozilla Firefox Safari (iOS) oder dem Standard Android Browser. Es ist kein Support für Internet Explorer oder Microsoft Edge vorgesehen, dementsprechend wurden diese Browser auch nicht zum Testing verwendet.

Zur Erstellung von Tilemaps wurde das Programm „Tiled“ verwendet.

## 4.4 Versionsverwaltung

Zur Versionsverwaltung wurde die git-Technologie verwendet. Hier wurde ein Repository auf GitHub (vgl. <https://www.github.com/> ) erstellt. Der Vorteil ist, dass parallel an dem Projekt gearbeitet werden kann, und der Source-Code in der aktuellsten Version zentral verfügbar ist. Des Weiteren können bei einer Entwicklung in die falsche Richtung oder einem unlösbar scheinenden Bug alte Entwicklungsstände zurückgeholt werden.

## 4.5 Konventionen

Es wurden folgende Naming-Konventionen vor dem Beginn des Projekts besprochen:

**CSS-Klassen:** Camel-Case, **Dateinamen** (inkl. Bilder und Audio, mit Ausnahme von Klassen): kleingeschrieben, mehrere Wörter mit Bindestich verbunden, **Dateien die nur Klassen sind** und keine Komponenten sind: Camel-Case mit großem Anfangsbuchstaben, **Variablen und Methoden**: Camel-Case.

# 5 Architektur

## 5.1 Aufbau des Projektes

Das Projekt ist übersichtlich und erweiterbar aufgebaut. Die verschiedenen Dateien sind nach verschiedenen Funktionen getrennt und liegen in nach Funktionen sinnvoll benannten Ordnern. Die Projektstruktur (s. Anhang TODO) ist folgendermaßen aufgebaut: Die Quelltext-Dateien liegen im Ordner „/src/“, die Assets (Bilder, Audio, Tilemaps) liegen im Ordner „/assets/“, die verwendeten Bibliotheken unter „/lib/“ und die Dokumenation unter „/doc/“.

## 5.2 Einstiegspunkt

Zunächst wurde in der index.html ein Container erzeugt, in dem das Spielgeschehen ablaufen soll. Über den Script-Tag im Head wird die Phaser-Bibliothek geladen. Außerdem wurden die weiteren benötigten Abhängigkeiten geladen.

...

<script src="./lib/phaser.min.js"></script>

...

<body>

<div id="game-container"></div>

<script src="./src/managing/game-state.js"></script>

......

</body>

...

Abbildung 1: index.html Auszug

In der main.js wird zunächst ein config-Objekt (s. Abbildung 4: config-Objekt) initialisiert, welches dann dem Phaser-Game übergeben wird

const game = new Phaser.Game(config);

## 5.3 Scenes

### 5.3.1 Preload

### 5.3.2 Menu

### 5.3.3 Stages

## 5.4 JavaScript

### 5.4.1 game-state.js

#### 5.4.2 state-managing.js

#### 5.4.3 music-config.js, constants.js, util.js

## 5.4 Scaling

## 5.5 JSON-Tilemaps

# 6 Projektplanung und Arbeitsteilung

## 6.1 Zeitlicher Ablauf und Arbeitsteilung

Der zeitliche Ablauf wurde in im Anhang unter i. Projektplanung mithilfe eines Gantt-Diagramms dargestellt. Die Arbeitsteilung erfolgte zunächst nach dem was derjenige sich zutraute. So wurde die Architektur des Programmes an den Entwickler gegeben, der das meiste Vorwissen hatte, allerdings immer in Absprache mit den Gruppenmitgliedern. Nach und nach konnten die Aufgaben besser verteilt werden und in fester Absprache konnten einzelne Teilpakete des Projektes getrennt bearbeitet werden. Am Ende zum „Zusammenfügen“ der Puzzleteile und Testen wurde die Arbeit wieder gemeinsam erledigt. Das Projekt wurde in erfolgreicher Zusammenarbeit abgeschlossen.

# 7 Post Mortem

## 7.1 Was hast gut funktioniert

Die Komponentenbasierte Architektur wurde sehr ordentlich durchgeplant und hat letztendlich auch den Rahmen des Projektes getragen. Durch den strukturierten Aufbau war es leicht neue Komponenten hinzuzufügen und die Arbeit aufzuteilen.

Die Versionsverwaltung sorgte ebenfalls für ein gutes teamorientiertes Arbeiten und einen Austausch von Entwicklungsergebnissen, ohne dass man nebeneinandersitzen musste. Dadurch hat die Aufteilung der Aufgaben auch gut funktioniert und die Arbeit gleichmäßig aufgeteilt.

Die Versionsverwaltung sorgte auch einmal dafür, dass Projekt zu „retten“, als die Applikation sich immer in einer Endlosschleife befand, und nicht herausgefunden werden konnte, woran das Problem lag. Hier wurde das Projekt auf den vorherigen Commit zurückgespielt.

## 7.2 Was hat nicht funktioniert

Durch das dynamische Laden der Daten gab es etwas größere Schwierigkeiten mit der Asynchronität. Es musste sich Wissen über „Promises“ angeeignet werden, welches im Nachhinein schwierig war in das Projekt einzupflegen. So kommt es sehr selten immer noch vor, dass das Projekt nicht startet, da die Daten nicht rechtzeitig geladen werden. Ein einfacher Reload löst das Problem allerdings meistens schon.

Des Weiteren hat die Eigengestaltung von Grafiken nicht funktioniert, da zeitlich wenig Platz und generell viel Misserfolg in der Kreation stattfand. So wurde entschieden auf fertige Icons, Bilder und Musik aus dem Internet zuzugreifen und diese zu verwenden und die Autoren zu kreditieren. Das macht das Spiel nicht ganz „stimmig“ auch wenn auf identischen Stil der Icons geachtet wurde.

Des Weiteren gab es kleinere Probleme mit dem Balancing. Dadurch dass die Aktionen zufällig geladen werden, ist es möglich, dass ein Spiel unschaffbar ist.

## 7.3 Rückblick

Obwohl die Architektur dieses Projekts gut getragen hat und gut geplant und strukturiert wurde, würde nächstes Mal mehr Wert auf Data-Binding gelegt werden. Jede Änderung muss über den UI-Manager aufgerufen werden. Einen Weg die Oberfläche zu aktualisieren, wenn sich die Daten ändern, würde die Menge an Code deutlich schmälern und das Projekt leichter erweiterbar machen.

Des Weiteren muss beim nächsten Projekt darauf geachtet werden, Probleme von Asynchronität und großen Dateien (Stichworte Lazy-Loading, Pre-Loading, Promises) mit einzuplanen und die Architektur und das Gesamtprodukt stabiler zu machen und zu verbessern.

Außerdem würde, wenn dieses Projekt erneut durchgeführt werden würde, mehr Wert auf Grafik gelegt werden, um eine stimmigere Atmosphäre und ein besser abgestimmtes Produkt zu erstellen.

Um das Balancing zu verbessern, ist es gegebenenfalls sinnvoll den Aktionen eine Häufigkeit zu geben, damit mehr Kontrolle vorhanden ist, die Aktionen und Belohnungen besser ins Gleichgewicht zu bringen und das Spiel auch durch mehr Strategie zu gewinnen.

## 7.4 Ausblick

Denkbare Erweiterungen des Spiels sind weitere Charaktere, die unterschiedliche Schwierigkeitsstufen beinhalten. Diese Charaktere können wiederum mit weiteren Aktionen, die charakterspezifisch sind erweitert werden.

Außerdem können die Locations und die Items erweitert werden, um mehr Möglichkeiten im Spiel zu haben und das Universum zu erweitern.

Denkbar ist auch, eine neue Art von Ressourcenleiste hinzuzufügen, um die Komplexität des Spiels zu erweitern und interessanter und ggf. schwieriger oder leichter zu machen.

# Anhang

# i. Projektplanung

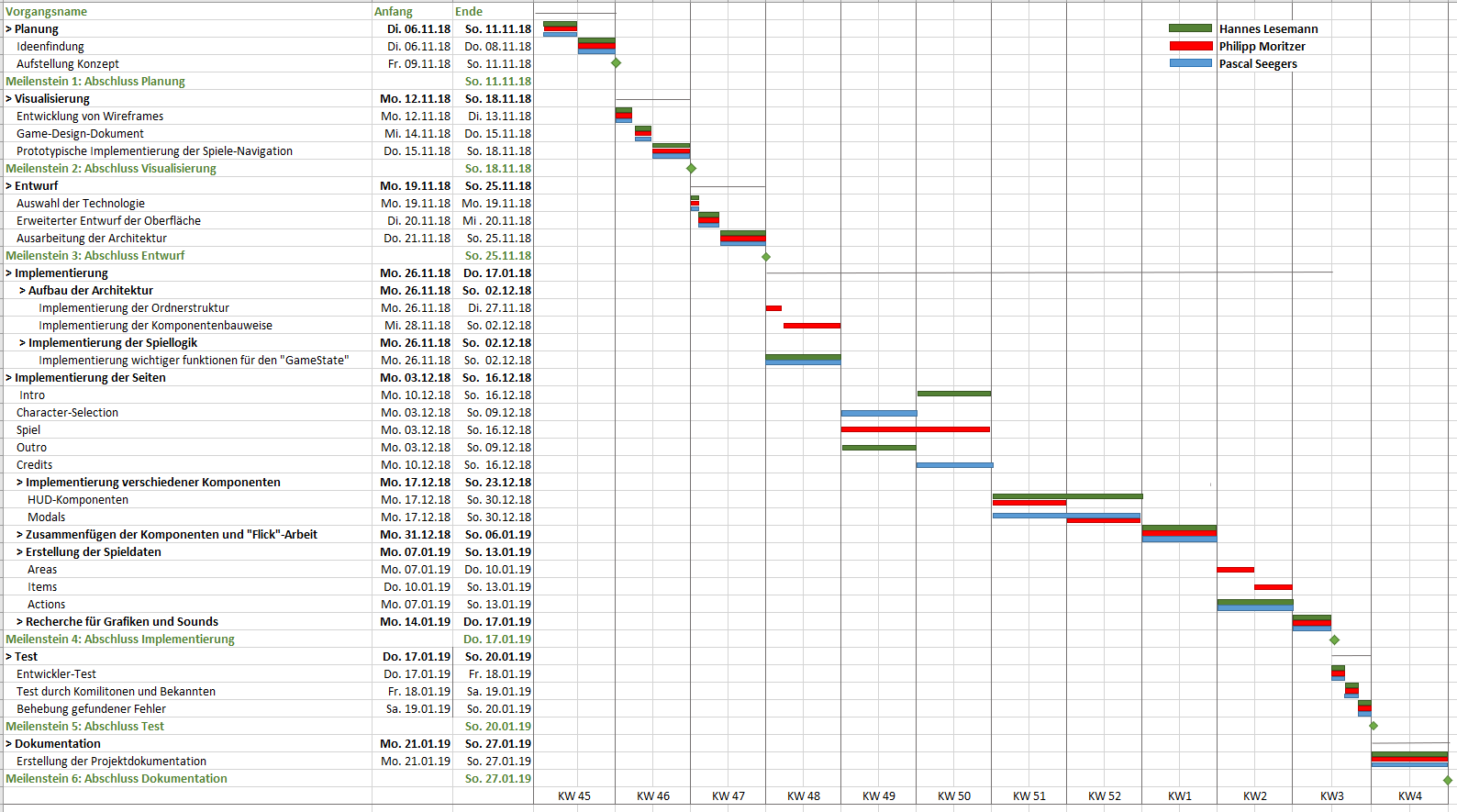


Abbildung 2: GANTT-Diagramm Planung

# ii. Projektstruktur

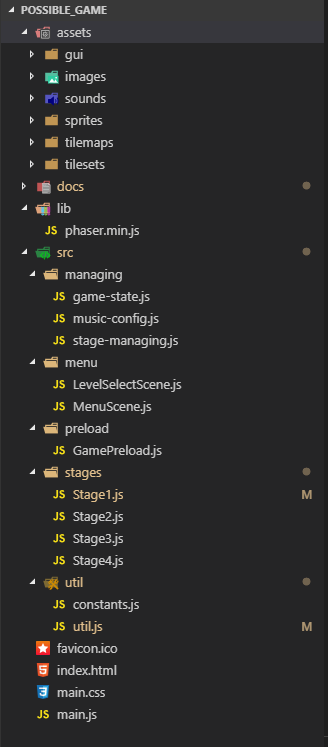


Abbildung 3: Projektstruktur

const config = {

type: Phaser.AUTO,

scale: {

mode: Phaser.Scale.FIT,

autoCenter: Phaser.Scale.CENTER\_BOTH,

width: GAME\_WIDTH,

height: GAME\_HEIGHT

},

parent: "game-container",

backgroundColor: "#E4F6F8",

zoom: 1,

physics: {

default: "arcade",

arcade: {

tileBias: 32,

gravity: { y: 5000 }

}

},

scene: [

GamePreload,

MenuScene,

LevelSelectScene,

Stage1,

Stage2,

Stage3,

Stage4

]

};

Abbildung 4: config-Objekt

# iii. Tiled Map Building

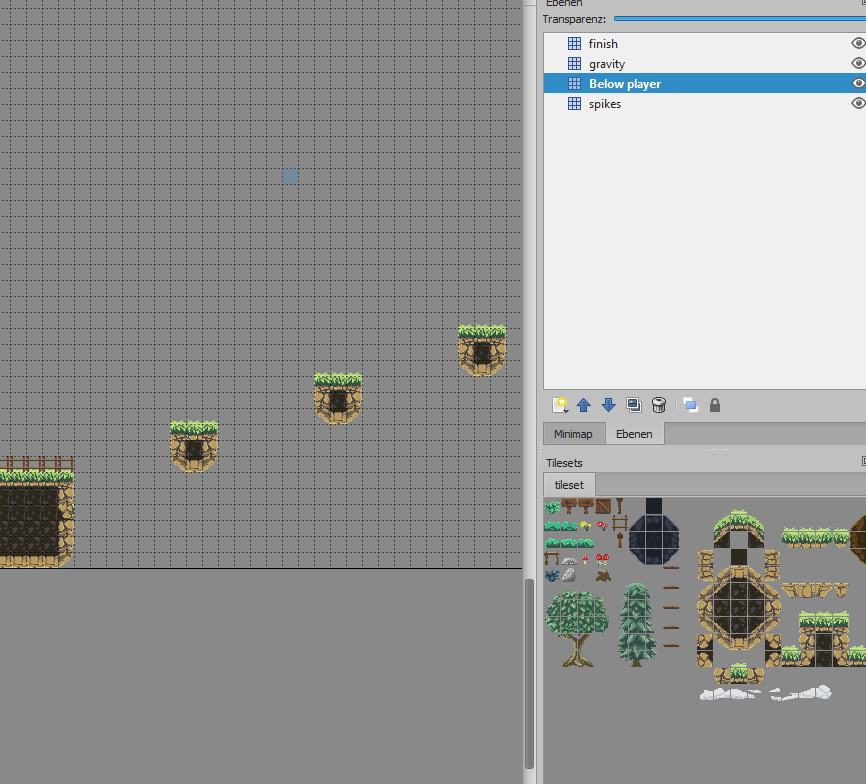


Abbildung 5: Erstellen von Tiledmaps

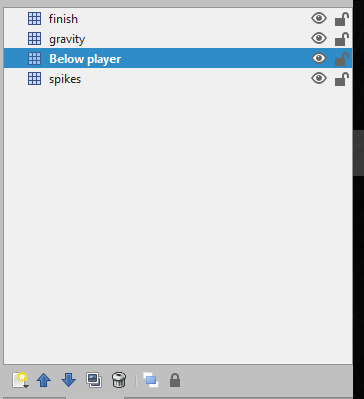


Abbildung 6: Tiled layers

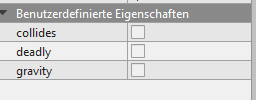


Abbildung 7: Tiled Attribute

# iv. Tileset



Abbildung 8: Tileset

# vi. Auszug Tilemap als .json

{ "height":150,

"infinite":false,

"layers":[

{

"data":[154, 155, 155, 3221225627, 155, 155, 3221225627, 155, 155, 3221225627, 155, 155, 3221225627, 155, 155, 3221225627, 155, 155, 3221225627

...

"height":150,

"id":16,

"name":"gravity",

"opacity":1,

"type":"tilelayer",

"visible":true,

"width":550,

"x":0,

"y":0

...

"id":35,

"properties":[

{

"name":"collides",

"type":"bool",

"value":false

}]

},

...

Abbildung 9: Auszug Tilemap .json-Export

# iv. Collision debugging

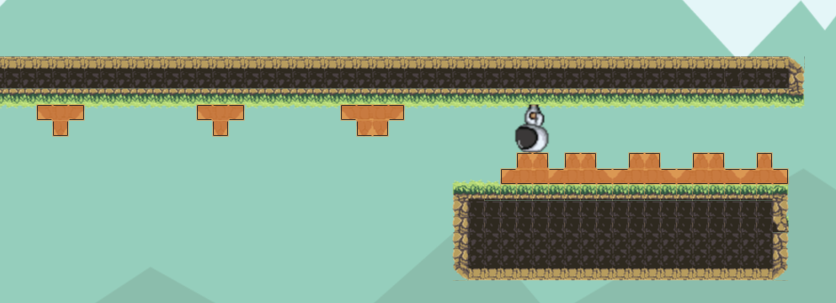


Abbildung 10: Collision Debugging

# iii. Desktop Menu Screenshots



Abbildung 11: Desktop-Hauptmenü



Abbildung 12: Dektop-Level-Select

# iv. Desktop Game Screenshots



Abbildung 13: Stage 1 – Screenshot



Abbildung 14: Stage 2 – Screenshot

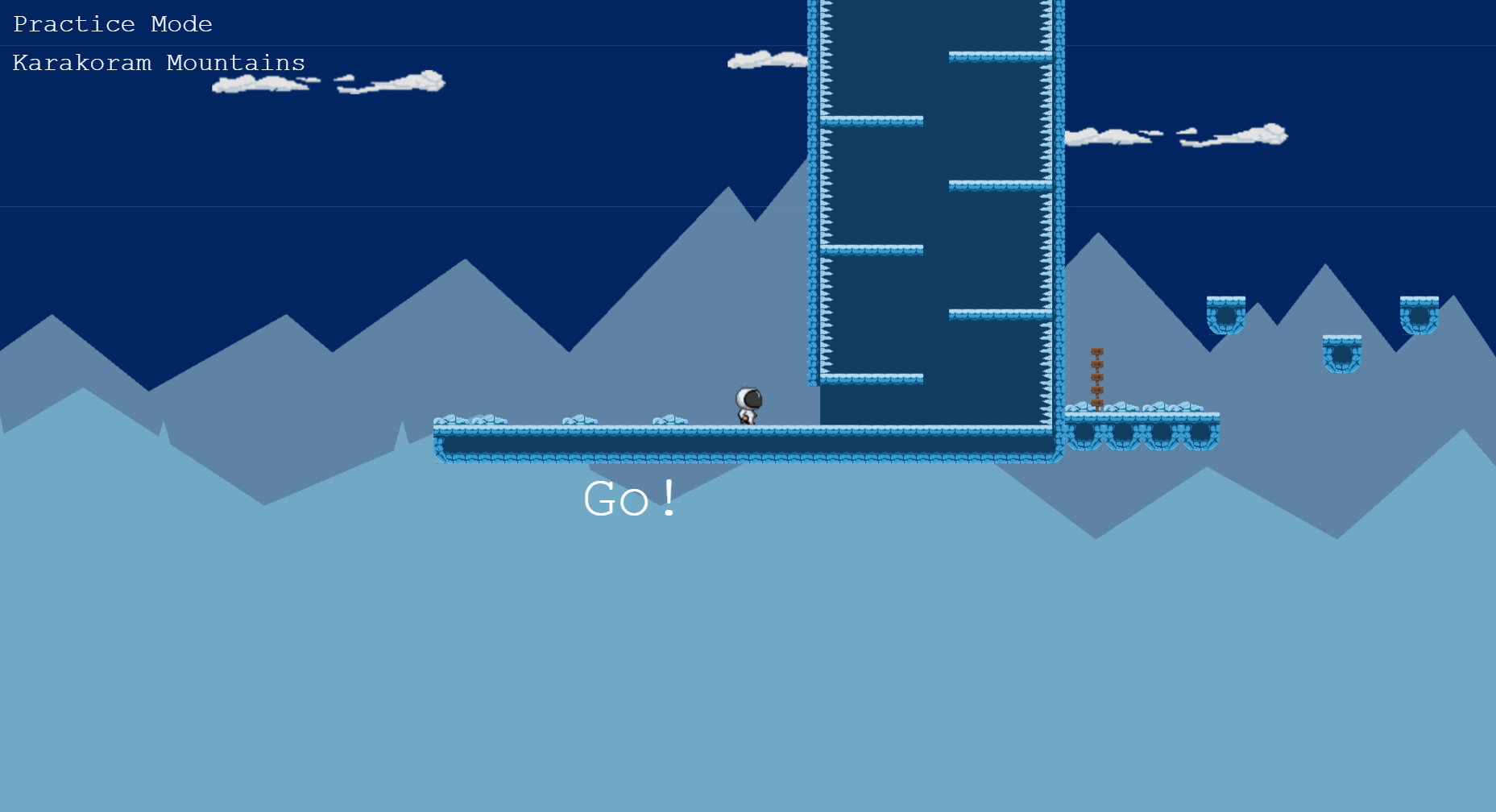


Abbildung 15: Stage 3 – Screenshot

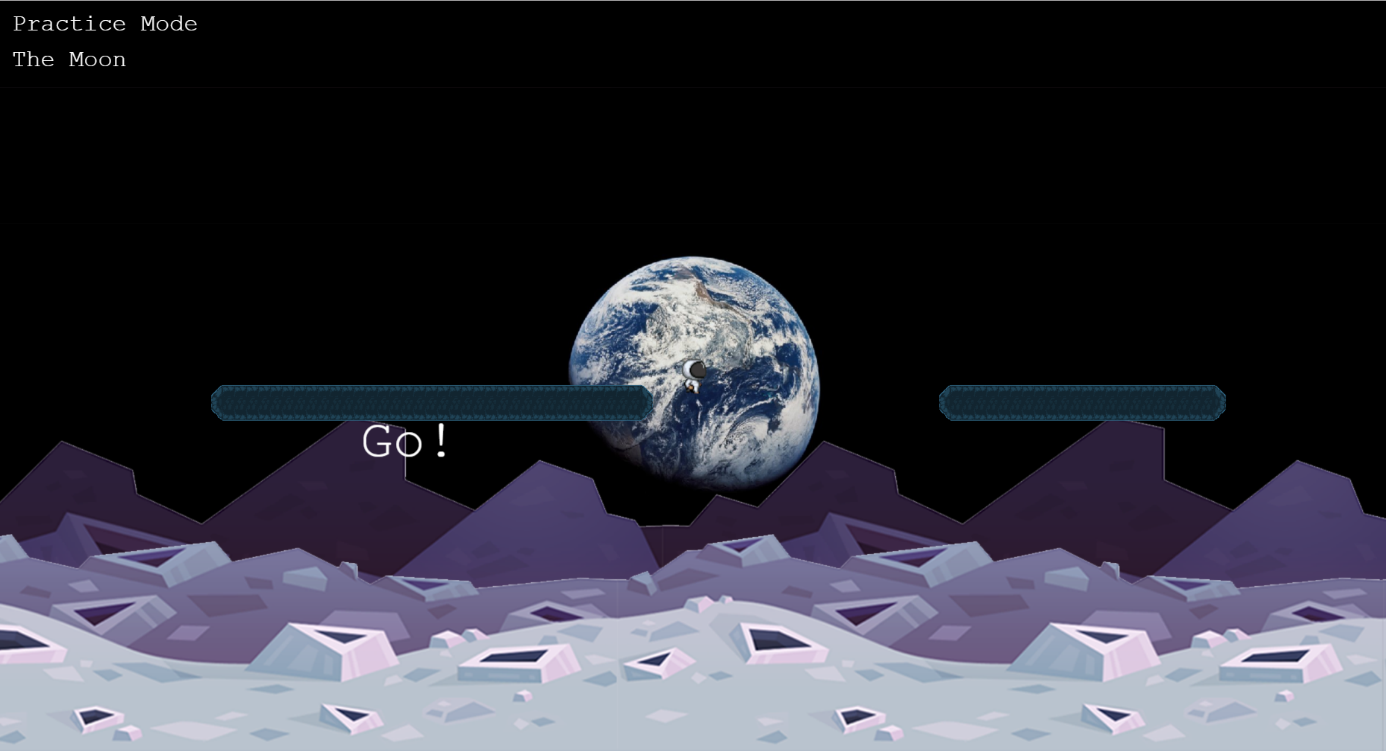


Abbildung 16: Stage 4 – Screenshot

# v. Mobile Ansichten

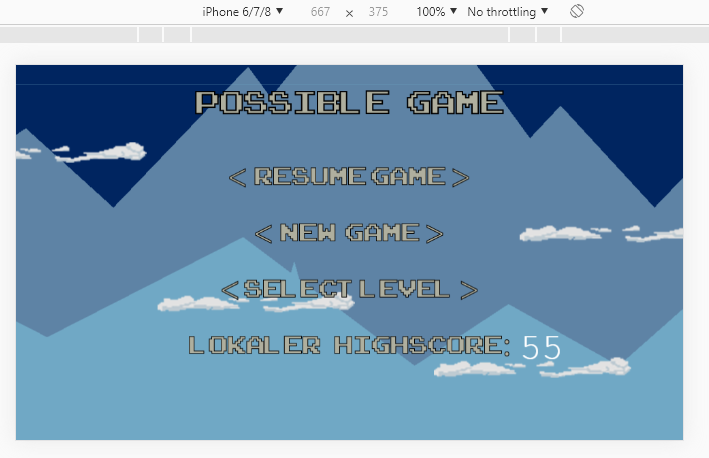


Abbildung 17: Menu-Screenshot Phone

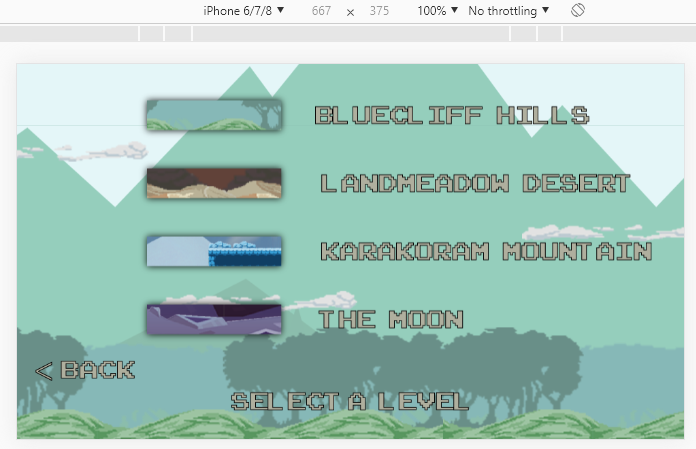


Abbildung 18: Select-Level Screenshot Phone

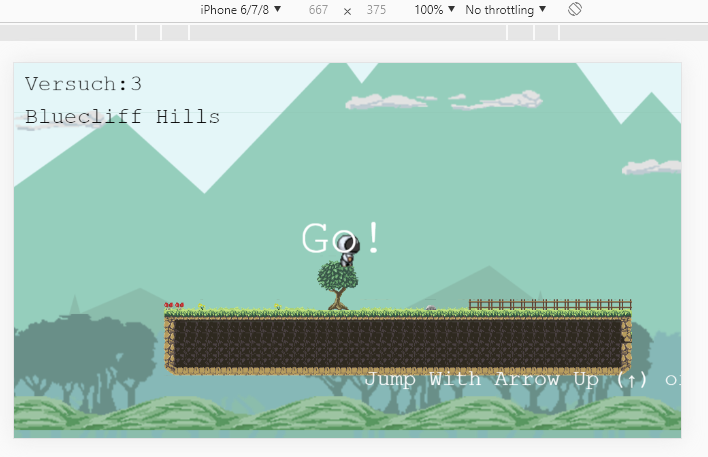


Abbildung 19: Game Screenshot Phone

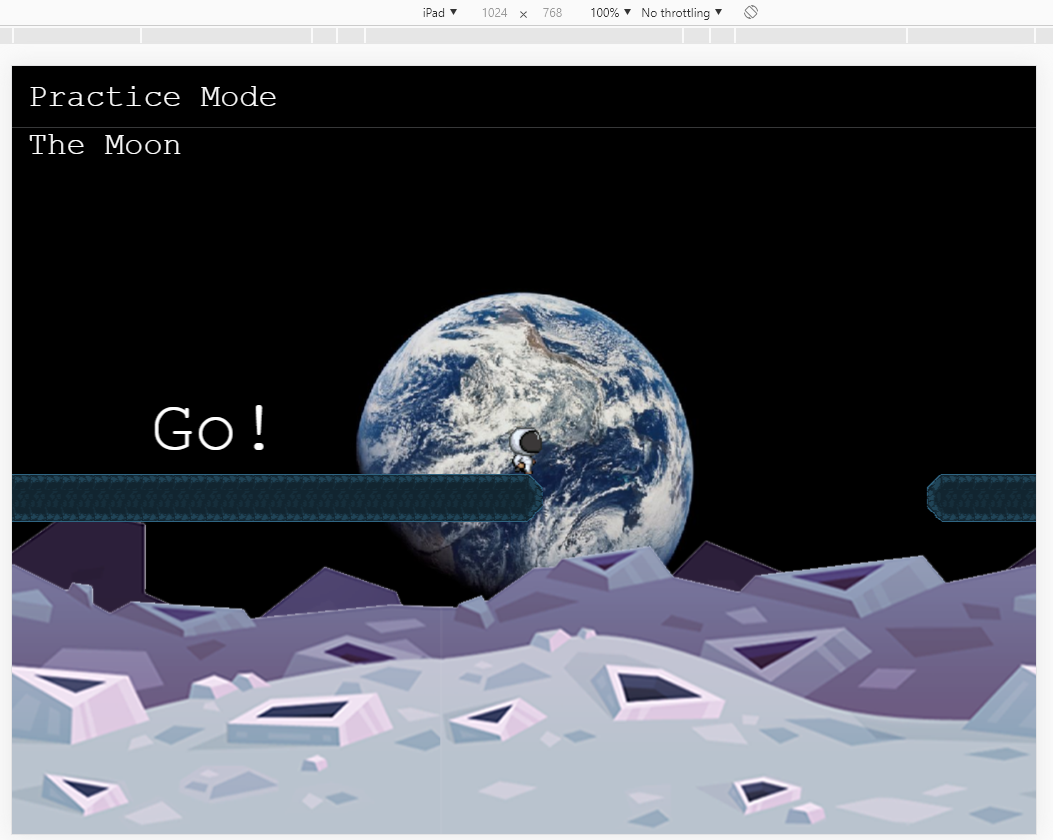


Abbildung 20: Game Screenshot Tablet