HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent	:	**Gabriela Mai**
Vorgelegt am	:	30. November 2016
Vorgelegt von	:	Gruppe 4

Florian Durli	:	254791
Jannik Ivosevic	:	255028
Johannes But	:	254053
Marco Mayer	:	254795
Koray Emtekin	:	254816

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Team

Abbildung 1.: Florian Durli

Abbildung 2.: Jannik Ivosevic

Abbildung 3.: Johannes But

Abbildung 4.: Marco Mayer

Abbildung 5.: Koray Ektekin

1.2. Rollenverteilung

Phase	Projektleiter
Anforderungen	Johannes But
Implementation	Florian Durli
Test	Jannik Ivosevic
Dokumentation & Präsentation	Marco Mayer, Koray Emtekin

Tabelle 1.: Rollenverteilung

Der Projektleiter wechselt jede Phase. In der obigen Tabelle sind diese aufgeführt. Der Projektleiter ist jeweils für die Koordination der Aufgaben und die Organisation zuständig. Er dient als Ansprechpartner für das Projekt und gibt eine „Fahrtrichtung“ vor. Jedoch werden sämtliche wichtige Entscheidungen im Plenum getroffen.

1.3. Spielidee

1.3.1. Spielkonzept

Unser Spiel namens „InfinityRun“ wird ein Endlos-Spiel sein, bei dem es das Ziel ist die Spielfigur so lange wie möglich am Leben zu erhalten. Der dazugehörige Highscore ist abhängig von der Lebensdauer der Spielfigur. Bei ansteigender Zeit wird die Geschwindigkeit des Spiels stetig erhöht. Das Spielfeld mit den Hindernissen wird per Zufallsgenerator erzeugt, somit ist jeder Durchlauf einzigartig. Begleitend zum Spiel wird ein Soundtrack das Spielerlebnis abrunden. Bei Aufprall auf ein Hindernis, besteht die Möglichkeit das Spiel neu zu starten.

1.3.2. Entwurfsskizze

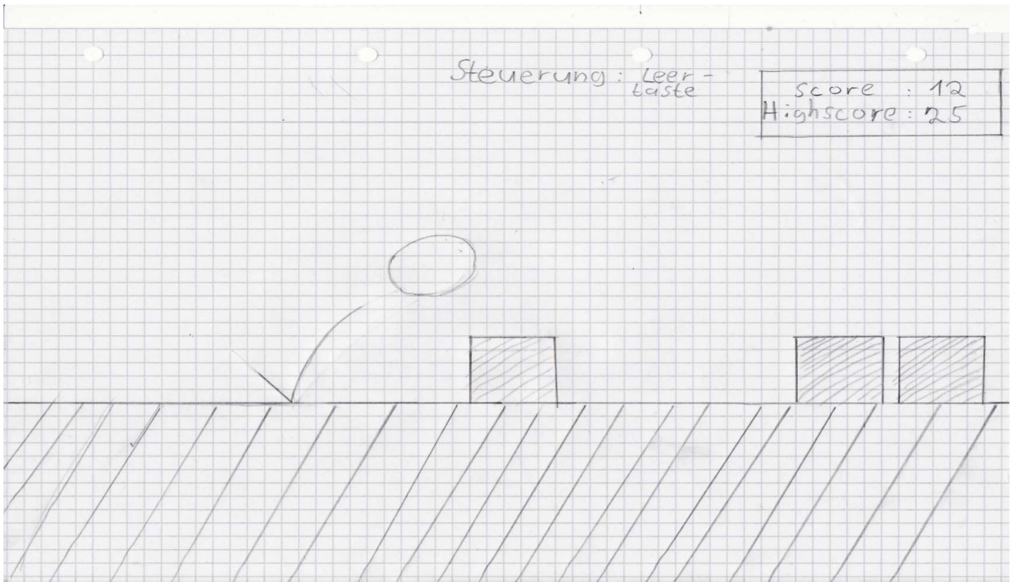

Abbildung 6.: Entwurfsskizze

Auf der abgebildeten Entwurfsskizze sehen Sie die grobe Oberfläche unseres Spieles. Der V-ähnliche Strich zeigt den Absprung eines Objektes, welches auf der Entwurfsskizze eine Kugel ist. Dies geschieht mit der Leertaste auf der Tastatur. Außerdem sind auf dem Bild noch verschiedene Blöcke zu sehen. Diese Blöcke kommen zufällig generiert von rechts in das Bild geflogen. Es können verschieden Kombinationen, z.B. ein Block, zwei Blöcke oder drei Blöcke, generiert werden. Außerdem kann man oben am rechten Rand den Score und den jeweils erreichten Highscore sehen. In unserer Entwurfsskizze ist der Score 12 und der Highscore 25. Dieser sogenannte Score berechnet sich, je nachdem über wie viele Blöcke unser Objekt gesprungen ist. Ist er über einen Block und danach über drei Blöcke gesprungen, zählt es nur zwei Punkte, da es nicht die Anzahl der Blöcke zählen soll, sondern die Anzahl der geschafften Sprünge. Der Highscore ist der jemals erreichte höchste Score in dem Spiel. Außerdem kann man neben dem Score und dem Highscore noch die Spielsteuerung sehen. Diese ist natürlich die Leertaste. Die Pausetaste wird mit der Taste P hinterlegt, womit man das Spiel pausieren kann. Man muss mit dem Objekt das richtige Timing erwischen, um über die Blöcke zu springen, anderenfalls landet man in einem oder mehreren Blöcken und darf nochmal von vorne beginnen. Um das Spiel interessanter zu gestalten wird das Spiel nach einem bestimmten Score schneller und somit schwieriger.

1.3.3. Erforderliche Software

1.3.3.1. Notepad++

Notepad++ ist ein freier Editor der es ermöglicht die Syntax von JavaScript korrekt und mit Highlighting darzustellen. Dieser Editor wird immer beliebter durch seine Unterstützung verschiedener Programmiersprachen. Quelle: [Ho]

1.3.3.2. Chrome

Chrome ist ein Webbrowser von der Firma Google der immer populärer wird. Er ist besonders benutzerfreundlich für Entwickler und bietet verschiedene Tools zum Debuggen. Quelle: [Goo]

1.3.3.3. Gimp

Zur erstellen unserer Grafiken benutzen wir das Bildbearbeitungsprogramm Gimp. Dies ist eine frei erhältliche Software, die einen erweiterten Funktionsumfang ähnlich wie das bekannte Programm Photoshop von Adobe bietet. Quelle: [Tea]

1.3.3.4. Git/Github

Wir haben uns dagegen entschieden die Softwareverwaltung der Hochschule zu nutzen und greifen nun auf eine alternative Lösung Namens Git zurück. Git ist eine freie Softwareverwaltung die durch Linus Torvalds entwickelt wurde. Github ist eine Open Source Plattform, die dieses Konzept nutzt. Somit können wir parallel an dem Projekt arbeiten und Versionsstände definieren, auf die wir jeder Zeit wieder zurück springen können. Somit ist ein Arbeiten wie in einem richtigen Softwareprojekt möglich. Quelle: [Git]

2. Phasen

2.1. Entwurf und Anforderungen

2.1.1. Funktionale Anforderungen

- Das System muss fähig sein zufällig eine Spielwelt mit Hindernissen zu generieren, welche jedoch so platziert werden müssen, dass sie immer überwindbar sind.
- Das System muss fähig sein das generierte Spielfeld durch das Bild nach links zu verschieben.
- Bei Drücken der Leertaste muss das System die Spielfigur hüpfen lassen.
- Das System muss fähig sein einen Highscore in Abhängigkeit zur Spieldauer zu generieren. Der Highscore soll proportional zum Levelfortschritt berechnet werden und dauerhaft angezeigt werden. Hierbei soll der aktuelle Score und der Highscore der Spielesession getrennt angezeigt werden. Dieser wird nur solange gespeichert, bis das Spiel beendet wird.
- Das System muss fähig sein während des Spielens eine Hintergrundmusik abzuspielen, welche sich ständig wiederholt.
- Das System muss fähig sein beim Springen der Spielfigur, beim Aufkommen der Spielfigur und beim Kollidieren der Spielfigur Effektsounds wiederzugeben.
- Das System muss die Möglichkeit bieten bei Tastendruck das Spiel zu pausieren und wieder zu starten.
- Das System muss fähig sein eine Kollision der Spielfigur mit einem Hindernis zu erkennen, nach Erkennen soll ein „Crash“ Sound abgespielt werden und sich die Spielfigur verändern.
- Das System muss fähig sein kontinuierlich die Schwierigkeit zu erhöhen. Die Schwierigkeit soll dadurch erhöht werden, dass das Spielfeld anfangs langsam nach links wandert und dies kontinuierlich immer schneller wird.
- Bei Beendigung des Spiels muss das System fähig sein das Spiel neu zu starten.
- Das System muss auf einem Gerät mit Tastatur im Browser Chrome ablaufen.

2.1.2. Nicht funktionale Anforderungen

- Das Spiel sollte intuitiv bedienbar sein.
- Die Perfomarnce des Spiels sollte so gut sein, dass keine Frame Einbrüche vorkommen.
- Auch auf den weiterverbreiteten Browsern sollte das Spiel spielbar sein.

2.1.3. Projektplan

Datum	Aufgabe
19.10.2016	Einführung in jeweilige Projekte der Gruppen
21.10.2016	Einführung in jeweilige Projekte der Gruppen
26.10.2016	Anforderungen
02.11.2016	Fertigstellung Präsentation, Ergebnispräsentation der Anforderungen
04.11.2016	Abgabe der Anforderungsspezifikation via Felix

Tabelle 2.: Phase 1: Entwurf und Anforderungen

Datum	Aufgabe
09.11.2016	Basis Implementierung
16.11.2016	Basis Implementierung + Level Design
23.11.2016	Zwischenpräsentation der Implementierung
25.11.2016	Abgabe: Zwischenstand der Implementation via Felix
30.11.2016	Level Design Verbesserungen
07.12.2016	Stabilität & Bug fixing
14.12.2016	Ergebnispräsentation der Implementierung
16.12.2016	Abgabe Implementierungsergebnisses via Felix (Code Freeze)

Tabelle 3.: Phase 2: Implementierung

Datum	Aufgabe
21.12.2016	Test und Resultate dokumentieren
11.01.2017	Ergebnispräsentation
13.01.2017	Abgabe der Ergebnisse der Testphase

Tabelle 4.: Phase 3: Test

Datum	Aufgabe
18.01.2017	Dokumentation
25.01.2017	Ergebnispräsentation Dokumentation
27.01.2017	Projektvorstellung auf der Projektmesse

Tabelle 5.: Phase 4: Dokumentation und Präsentation

2.1.4. Releaseplan

Version	Datum	Inhalt
1.0.0	09.11.16	Spiel ist startfähig mit passendem Hintergrund und Spielfigur
1.1.0	16.11.16	Automatischer Bildlauf und springen ist möglich
1.2.0	30.11.16	Beinhaltet: Zufallsgenerierte Objekte(Hindernisse) mit unendlichem Level
1.3.0	07.12.16	Highscore, Hintergrundlied, Sound beim Springen
1.4.0	14.12.16	Zeitbasierte Geschwindigkeit (Bildlauf)
1.5.0	21.12.16	Erfolgreicher Test mit behobenen Fehlern

Tabelle 6.: Releaseplan

Beim Releaseplan haben wir uns auf eine Versionierung des Programms mit aufsteigenden Nummern geeinigt. Die Erste Nummer steht hierbei für die Grundlegende Programmversion. Die Zweite für wichtige Updates und die Dritte für Bugfixes zwischendurch. Zur jeweiligen Version haben wir ein Fertigstellungsdatum festgelegt und den dann erforderlichen Inhalt festgelegt.

2.2. Implementation - Zwischenstand

2.2.1. Erfüllte Anforderungen

- Das System muss fähig sein zufällig eine Spielwelt mit Hindernissen zu generieren welche jedoch so platziert werden müssen dass sie immer überwindbar sind.
- Das System muss fähig sein das generierte Spielfeld durch das Bild nach links zu verschieben.
- Bei Drücken der Leertaste muss das System die Spielfigur hüpfen lassen.
- Das System muss die Möglichkeit bieten bei Tastendruck das Spiel zu pausieren und wieder zu starten.
- Das System muss fähig sein kontinuierlich die Schwierigkeit zu erhöhen. Die Schwierigkeit soll dadurch erhöht werden, dass das Spielfeld anfangs langsam nach links wandert und dies kontinuierlich immer schneller wird.
- Bei Beendigung des Spiels muss das System fähig sein das Spiel neu zu starten.
- Das System muss auf einem Gerät mit Tastatur im Browser Chrome ablaufen.

2.2.2. Nicht erfüllte Anforderungen

- Das System muss fähig sein eine Kollision der Spielfigur mit einem Hindernis zu erkennen, nach Erkennen soll ein „Crash“ Sound abgespielt werden und sich die Spielfigur verändern.
- Das System muss fähig sein einen Highscore in Abhängigkeit zur Spieldauer zu generieren. Der Highscore soll proportional zum Levelfortschritt berechnet werden und dauerhaft angezeigt werden. Hierbei soll der aktuelle Score und der Highscore der Spielesession getrennt angezeigt werden. Dieser wird nur solange gespeichert, bis das Spiel beendet wird.
- Das System muss fähig sein, während des Spielens eine Hintergrundmusik abzuspielen, welche sich ständig wiederholt.
- Das System muss fähig sein beim Springen der Spielfigur, beim Aufkommen der Spielfigur und beim Kollidieren der Spielfigur Effektsounds wiederzugeben.

2.2.3. Das Spiel

Hier werden zwei Screenshots des derzeitigen Spiels dargestellt. In der Abbildung 7 zu sehen, ist der Startbildschirm des Spiels. Hier gibt es verschiedene Auswahlmöglichkeiten. In der Abbildung 8 zu sehen ist der derzeitige Stand des Spiels.

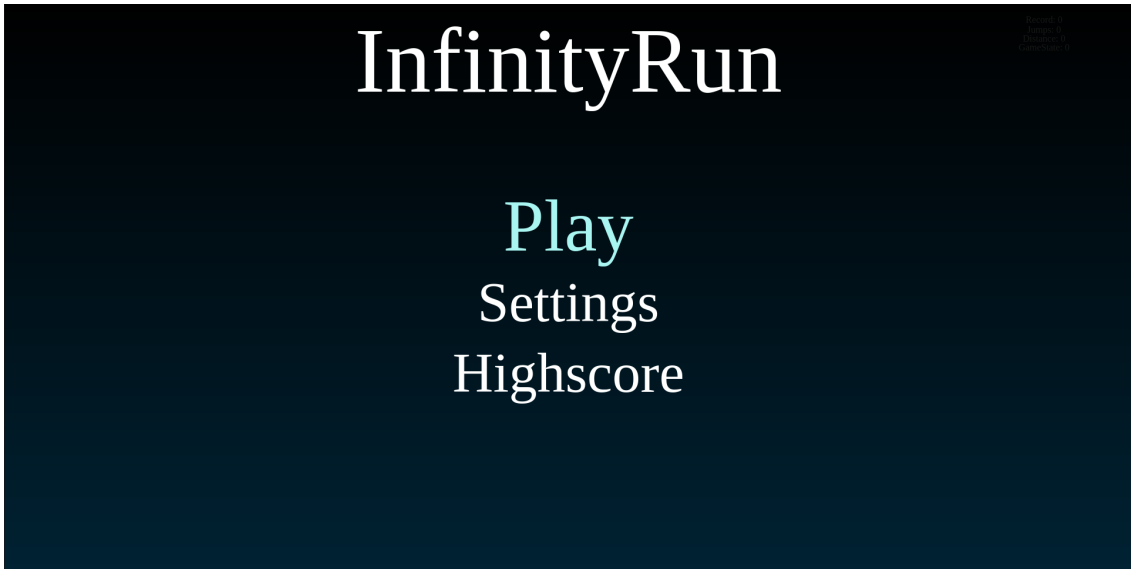

Abbildung 7.: Startbildschirm

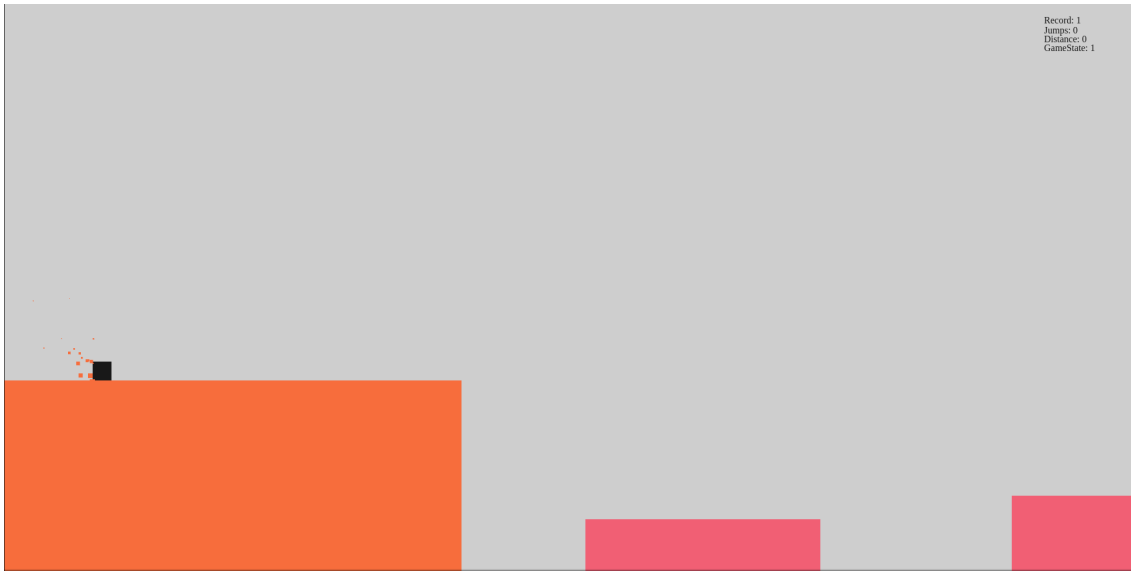

Abbildung 8.: Das Spiel

2.2.4. Bibliothek

Bei der Erstellung des Spiels greifen wir auf eine JavaScript Bibliothek namens "Sketch.js" zurück. Das Sketch.js Framework ermöglicht es uns, den Code vereinfacht und lesbarer zu schreiben. Beispiel wie Sketch.js funktioniert:

```
function start()
{
        context.now = +new Date();
    context.running = true;
}

function stop()
{
        context.running = false;
}

function toggle()
{
        ( context.running ? stop : start )();
}

function clear()
{
        if ( is2D )
        context.clearRect( 0, 0, context.width, context.
           height );
}
```

Quelle: [sou]

2.2.5. Code

2.2.5.1. Framework initialisieren

Hier in dieser Funktion wird ein Canvas-Element erstellt, dies geschieht mithilfe des Sketch-Frameworks. Dabei werden Eigenschaften wie die Höhe und Breite der Zeichenfläche übergeben.

```
var InfinityRun = Sketch.create({
fullscreen: true,
width: 640,
height: 360,
```

```
5 container: document.getElementById('container')
});
```

2.2.5.2. Spieler initialisieren

In der Player-Update-Funktion wird der Player also unsere Spielfigur aktualisiert. Damit die Schwerkraft gegeben ist, wird zuerst die Y-Geschwindigkeit um eins erhöht. Hierbei ist zu beachten, dass die Y- Koordinatenachse nach unten zeigt.Danach wird die Position des Spielers neu festgesetzt. Für den Fall, dass der Spieler verliert, welches mittels if-Entscheidung überprüft wird, werden dann anschließend sämtliche Spielwerte auf ihren Ausgangswert zurückgesetzt. Als letztes wird überprüft ob der Spieler eine Taste gedrückt um zu Springen. Falls ja und er sich nicht schon in der Luft befindet wird die Y-Geschwindigkeit in die negative Richtung erhöht und die Spielfigur springt.

```
Player.prototype.update = function() {
2 // Gravity
this.velocityY += 1;
4 this.setPosition(this.x + this.velocityX, this.y + this.
    velocityY);

6 if (this.y > InfinityRun.height || this.x + this.width <
    0)
{
8         this.x = 150;
          this.y = 50;
10        this.velocityX = 0;
          this.velocityY = 0;
12        InfinityRun.jumpCount = 0;
          InfinityRun.acceleration = 0;
14        InfinityRun.accelerationTweening = 0;
          InfinityRun.scoreColor = '#181818';
16        InfinityRun.platformManager.maxDistanceBetween =
            350;
          InfinityRun.platformManager.updateWhenLose();
18 }

20 if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
    InfinityRun.keys.W || InfinityRun.dragging) && this.
    velocityY < -8)
```

```
{
22          this.velocityY += -0.75;
}
24 };
```

2.2.5.3. Erstellen der Spielebene

In unserem Plattform-Manager werden die Plattformen initialisiert. Hierbei wird ein Wert "maxDistanceBetween" festgelegt. Ebenso werden mögliche Farben für die Plattformen gespeichert. Anschließend werden den ersten 3 Plattformen ihre Werte zugeordnet. Die erste Plattform hat hierbei feste Werte, damit der Spieler nicht sterben kann, am Anfang des Spiels. Die beiden nächsten Plattformen werden dann mit zufälligen Werten erstellt. Zum Schluss bekommt jede Plattform noch eine Höhe und Farbe zugeordnet.

```
Player.prototype.update = function() {
2 function PlatformManager()
{
4
        this.maxDistanceBetween = 300;
6
        this.colors = ['#2ca8c2', '#98cb4a', '#f76d3c', '
          #f15f74', '#5481e6'];
8
//first 3 Platforms execept the Starter Platform
10      this.first = new Platform({
        x: 300,
12      y: InfinityRun.width / 2,
        width: 400,
14      height: 70
})
16
this.second = new Platform
18 ({
        x: (this.first.x + this.first.width) + random(
          this.maxDistanceBetween - 150, this.
          maxDistanceBetween),
20      y: random(this.first.y - 128, InfinityRun.height
          - 80),
        width: 400,
```

```
22          height: 70
    })
24
    this.third = new Platform
26  ({
            x: (this.second.x + this.second.width) + random(
                this.maxDistanceBetween - 150, this.
                maxDistanceBetween),
28          y: random(this.second.y - 128, InfinityRun.height
                - 80),
            width: 400,
30          height: 70
    })
32          this.first.height = this.first.y + InfinityRun.
                height;
            this.second.height = this.second.y + InfinityRun.
                height;
34          this.third.height = this.third.y + InfinityRun.
                height;
            this.first.color = randomChoice(this.colors);
36          this.second.color = randomChoice(this.colors);
            this.third.color = randomChoice(this.colors);
38          this.colliding = false;
            this.platforms = [this.first, this.second, this.
                third];
40  }
```

2.2.5.4. Update der Plattformen

Die Plattform-Update-Funktion aktualisiert die 3 Plattformen. Sie hat zwei Aufgaben. Als erstes wird die Plattform immer, in Abhängigkeit zur Spielbeschleunigung, nach um drei nach links verschoben. Danach wird abgefragt, ob die Plattform schon ganz links aus dem Bild heraus gewandert ist und falls ja werden sämtliche Werte so zufällig neu gesetzt, dass sie wieder von rechts ins Bild laufen kann. Dies wird für alle 3 Plattformen gleich durchgeführt.

```
PlatformManager.prototype.update = function()
2   {
            this.first.x -= 3 + InfinityRun.acceleration;
4           if (this.first.x + this.first.width < 0)
```

```
        {
6               this.first.width = random(450, 
                    InfinityRun.width + 200);
                this.first.x = (this.third.x + this.third
                    .width) + random(this.
                    maxDistanceBetween - 150, this.
                    maxDistanceBetween);
8               this.first.y = random(this.third.y - 32, 
                    InfinityRun.height - 80);
                this.first.height = this.first.y + 
                    InfinityRun.height + 10;
10              this.first.color = randomChoice(this.
                    colors);
        }
12
        this.second.x -= 3 + InfinityRun.acceleration;
14      if (this.second.x + this.second.width < 0)
        {
16              this.second.width = random(450, 
                    InfinityRun.width + 200);
                this.second.x = (this.first.x + this.
                    first.width) + random(this.
                    maxDistanceBetween - 150, this.
                    maxDistanceBetween);
18              this.second.y = random(this.first.y - 32,
                     InfinityRun.height - 80);
                this.second.height = this.second.y + 
                    InfinityRun.height + 10;
20              this.second.color = randomChoice(this.
                    colors);
        }
22
        this.third.x -= 3 + InfinityRun.acceleration;
24      if (this.third.x + this.third.width < 0)
        {
26              this.third.width = random(450, 
                    InfinityRun.width + 200);
                this.third.x = (this.second.x + this.
                    second.width) + random(this.
```

```
                maxDistanceBetween - 150, this.
                maxDistanceBetween);
            this.third.y = random(this.second.y - 32,
                InfinityRun.height - 80);
            this.third.height = this.third.y +
                InfinityRun.height + 10;
            this.third.color = randomChoice(this.
                colors);
        }
};
```

2.2.5.5. Update der Plattformen

In folgender Funktion werden mithilfe einer for-Schleife zuerst alle drei Plattformen abgefragt, ob diese, anhand von: "if(this.player.intersects..) " den Spieler berühren. Falls der Spieler eine Plattform berührt, in diesem Fall " this.collidedPlatform.... " als Beispiel die zweite Plattform im Spiel berührt, so wird der Variable "collidedPlatform" ein Objekt der zweiten Plattform zugewiesen. Außerdem wird zusätzlich noch die Y-Koordinate des Spielers auf die der Plattform gesetzt, was hier die Funktion " this.player.y < this.platformManager...." ist. Zusätzlich wird wenn die Y-Koordinate des Spielers und die Y-Koordinate der Plattform übereinstimmen, die "velocityY" auf 0 gesetzt, was zur Folge hat, dass der Spieler nicht mehr fällt. Anschließend sollen die Partikel des Spielers die Farbe der Plattormen annehmen.

```
    for (i = 0; i < this.platformManager.platforms.length
        ; i++)
    {
            if (this.player.intersects(this.
                platformManager.platforms[i]))
            {
                    this.collidedPlatform = this.
                        platformManager.platforms[i];
                    if (this.player.y < this.
                        platformManager.platforms[i].y
                        )
                    {
                            this.player.y = this.
                                platformManager.
                                platforms[i].y;
                            // Gravity after
```

```
                                    Collision with Platform
                                this.player.velocityY = 0;
                            }

                            this.player.x = this.player.previousX;
                            this.player.y = this.player.previousY;

                            this.particles[(this.particlesIndex++) % this.particlesMax] = new Particle({
                            x: this.player.x,
                            y: this.player.y + this.player.height,
                            color: this.collidedPlatform.color
});
```

2.2.6. Nächste Ziele

Da die Grundlegenden Spielfunktionen implementiert sind wollen wir uns in der zweiten Phase der Implementation nun auf das Design und die Effektsounds konzentrieren.

2.3. Implementation - Endstand

2.3.1. Spielkonzept Änderungen

Folgende Spielkonzept Äbnderungen haben wir im laufe der Implementation vorgenommen:

- Die Spielebene hat anstatt Hindernisse Zufalls generierte variable Plattformen.
- Spiel-Menü eingefügt
- Spielhintergrund

2.3.2. Funktionsdiagramm

2.3.3. Grafiken

2.3.4. Sounds

Bei den implementierten Spielsounds greifen wir auf eine freie Sounddatenbank zurück. Quelle: [Fre]
Folgende Sounds werden wir verwenden:

Sounds	Links
Menu	https://www.freesound.org/people/lharman94/sounds/329597/
Main1	https://www.freesound.org/people/nicolasdrweski/sounds/179684/
Main2	https://www.freesound.org/people/joshuaempyre/sounds/251461/
Main3	https://www.freesound.org/people/Flick3r/sounds/48544/
Main4	https://www.freesound.org/people/Flick3r/sounds/45623/
Jump	https://www.freesound.org/people/Lefty_Studios/sounds/369515/
Level-Up	https://www.freesound.org/people/n_audioman/sounds/275895/
Error	https://www.freesound.org/people/SamsterBirdies/sounds/363920/
Crash	https://www.freesound.org/people/n_audioman/sounds/276341/

Tabelle 7.: Sound Links

2.4. Test

2.5. Dokumentation & Präsentation

Literaturverzeichnis

[Fre] FREESOUND: *Freesound.org* https://www.freesound.org/

[Git] GITHUB: *Softwareverwaltung* https://github.com/

[Goo] GOOGLE: *Google Chrome* https://www.google.com/chrome/

[Gru] GRUPPE4: *Changelog* https://github.com/Slay3r/InfinityRun/commits/master

[Ho] HO, Don: *Notepad++* https://notepad-plus-plus.org/

[sou] SOULWIRE: *Sketch Bibliothek* https://github.com/soulwire/sketch.js

[Tea] TEAM, The G.: *Bildbearbeitungssoftware* https://www.gimp.org/

Eidesstattliche Erklärung

Wir versichern, dass wir die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet haben. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Uns ist bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

FURTWANGEN, den 30. November 2016 Florian Durli

FURTWANGEN, den 30. November 2016 Jannik Ivosevic

FURTWANGEN, den 30. November 2016 Johannes But

FURTWANGEN, den 30. November 2016 Marco Mayer

FURTWANGEN, den 30. November 2016 Koray Emtekin

A. Anhang

A.1. Github Changelog

Der Changelog wird aus unseren Github Commits per Befehl exportiert. Derzeit ist die Quelle nicht einsehbar, da das Repository auf dem wir arbeiten auf "Private" gesetzt ist. Zur endgültigen Abgabe wird dieses natürlich Veröffentlicht.

```
$ git log --pretty=tformat:'%h %<(13)%an %cd %s%n' --date=short > CHANGELOG.md
```

Quelle: [Gru]

Github	Name
Slay3r	Florian Durli
r4qtor	Marco Maier
butjo	Johannes But
ans77	Jannik Ivosevic
Krusher999	Koray Emtekin

Tabelle 8.: Github Namen

Changelog:

```
ffe660b Slay3r          2016-11-30 Ausblick/Fazit auskommentiert

01e309a Slay3r          2016-11-30 Sounds

3a05368 Slay3r          2016-11-30 Neue Dokumentation hinzufügen

7c3596c Slay3r          2016-11-30 Neue Dokumentation hinzufügen

bfdb05c Slay3r          2016-11-30 entfernen der Dokumentation

f944707 r4qtor          2016-11-25 Querlesung - Marco

8b6bdde Florian Durli   2016-11-25 Abgabe

bc8d933 Florian Durli   2016-11-25 Code Cleanup für Doku
```

```
17 b6d7a09 butjo          2016-11-24 Rechtschreibkorrekturen

19 1faa558 r4qtor         2016-11-24 Delete phasen.tex root/

21 51f4a79 r4qtor         2016-11-24 updated phasen.tex

23 62af4b8 Krusher999     2016-11-24 Letzten Codes ↩
       geschrieben

25 93b8965 Florian Durli 2016-11-23 fix

27 9027301 Florian Durli 2016-11-23 Changelog finale Lösung

29 1392138 Florian Durli 2016-11-23 Jojos Description in ↩
       LaTeX

31 25ff037 butjo          2016-11-23 Beschreibung der ↩
       Codeteile in der phasen.tex von Johannes

33 1b2348c Florian Durli 2016-11-23 Changelog Additionally

35 cf467dc Florian Durli 2016-11-23 Changelog Additionally

37 6db1ad6 butjo          2016-11-23 Merge branch 'master' of ↩
       https://github.com/Slay3r/InfinityRun

39 b924713 Slay3r         2016-11-23 Generated Changelog

41 69dd747 butjo          2016-11-23 Merge branch 'master' of ↩
       https://github.com/Slay3r/InfinityRun

43 203ae2e Slay3r         2016-11-23 Bilder des Spiels

45 d274cfc Slay3r         2016-11-23 Anforderungen

47 a0909ac Slay3r         2016-11-23 Literaturverzichniss

49 19e2f3d Slay3r         2016-11-23 Doku update

51 21889f9 Florian Durli 2016-11-22 Add Changelog

53 743c95a butjo          2016-11-16 Formatierte sketch.min.↩
       js

55 d64d254 butjo          2016-11-16 Endlose ↩
       Schwierigkeitserhöhung

57 3707b94 butjo          2016-11-16 Präsentation ↩
```

```
    Zwischenstand

59 f23c9be Slay3r          2016-11-16 Präsentation ü
    berarbeitet

61 53aa72e butjo           2016-11-16 Präsentation und test

63 b5cb978 Florian Durli   2016-11-16 Merge pull request #1
    from r4qtor/master

65 275bd69 r4qtor          2016-11-15 tiny cleanup & incl.
    menu

67 11a5e55 Slay3r          2016-11-09 Basis implementation

69 c783850 Slay3r          2016-11-09 Bilder hinzugefügt

71 d88d0d2 Slay3r          2016-11-09 Dokumentations Basis

73 39b4705 Florian         2016-10-19 Initial Struktur der
    Ordner und files

75 093797e Florian Durli   2016-10-19 Delete Requirements

77 56c4aae Florian         2016-10-19 doc Ordner

79 b3b83fd Florian         2016-10-19 Requirements hinzugefügt

81 b4d2627 Florian Durli   2016-10-19 Initial commit
```

A.2. game.js

```
/* todo: cleanup (dirty code),
2  *
   * Put static values / vars into initialization function
4  *
   * ------------------------------
6  * Design / Graphics
   *
8  * Parallax Background?
   *
10 * ------------------------------
   * Menu
12 * Menu draw in Input & draw prototypes
   * Handle / Manage CSS or HTML variables from JavaScript
    (Fullscreen,...)
14 * ------------------------------
```

```
 *
 * Platform Schematic? - Schematic files?
 * Different Themes depending on Progress?
 *
 * ------------------------------
 * Test-Phase
 *
 * Controller: 'dragging' test Touch support
 * Browsertesting tools
 * eg.:
 * http://browserling.com/
 * http://browsershots.org/
 * https://crossbrowsertesting.com/
 * https://www.browserstack.com/
 */
var i = 0;

var State = { Menu:0, Started:1, Paused:2, Over:3 };
var GameState = State.Menu;
var MainMenu;

var debug = true;

// randomizer
function random(min, max) {
    return Math.round(min + (Math.random() * (max - min)));
}

function randomChoice(array) {
    return array[Math.round(random(0, array.length - 1))];
}


//initialize Sketch Framework
var InfinityRun = Sketch.create({
    fullscreen: true,
    width: 640,
    height: 360,
    container: document.getElementById('container')
});

//------- Vector [Get/Set] Functions ----------

//Set X,Y,Width,Height
function Vector2(x, y, width, height) {
    this.x = x;
```

```
        this.y = y;
62      this.width = width;
        this.height = height;
64      this.previousX = 0;
        this.previousY = 0;
66  };


68
    // Set X,Y
70  Vector2.prototype.setPosition = function(x, y) {

72      this.previousX = this.x;
        this.previousY = this.y;
74
        this.x = x;
76      this.y = y;

78  };
    // Set X
80  Vector2.prototype.setX = function(x) {

82      this.previousX = this.x;
        this.x = x;
84
    };
86
    // Set Y
88  Vector2.prototype.setY = function(y) {

90      this.previousY = this.y;
        this.y = y;
92
    };
94
    // Collision / Intersection Top
96  Vector2.prototype.intersects = function(obj) {

98      if (obj.x < this.x + this.width && obj.y < this.y +
           this.height &&
            obj.x + obj.width > this.x && obj.y + obj.height
               > this.y) {
100         return true;
        }
102
        return false;
104 };

106 // Collision / Intersection Left
```

```
Vector2.prototype.intersectsLeft = function(obj) {

    if (obj.x < this.x + this.width && obj.y < this.y + this.height) {
        return true;
    }

    return false;
};

//---------- Player ----------

function Player(options) {

    this.setPosition(options.x, options.y);
    this.width = options.width;
    this.height = options.height;
    this.velocityX = 0;
    this.velocityY = 0;
    this.jumpSize = -13;
    this.color = '#181818';

}

Player.prototype = new Vector2;

Player.prototype.update = function() {
    // Gravity
    this.velocityY += 1;
    this.setPosition(this.x + this.velocityX, this.y + this.velocityY);

    if (this.y > InfinityRun.height || this.x + this.width < 0) {
        this.x = 150;
        this.y = 50;
        this.velocityX = 0;
        this.velocityY = 0;
        InfinityRun.jumpCount = 0;
        InfinityRun.acceleration = 0;
        InfinityRun.accelerationTweening = 0;
        InfinityRun.scoreColor = '#181818';
        InfinityRun.platformManager.maxDistanceBetween = 350;
        InfinityRun.platformManager.updateWhenLose();
    }

    if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
```

```
        InfinityRun.keys.W || InfinityRun.dragging) &&
       this.velocityY < -8) {
        this.velocityY += -0.75;
    }

};

Player.prototype.draw = function() {
    InfinityRun.fillStyle = this.color;
    InfinityRun.fillRect(this.x, this.y, this.width, this
       .height);
};

// ---------- Platforms ----------

function Platform(options) {
    this.x = options.x;
    this.y = options.y;
    this.width = options.width;
    this.height = options.height;
    this.previousX = 0;
    this.previousY = 0;
    this.color = options.color;
}

Platform.prototype = new Vector2;

Platform.prototype.draw = function() {
    InfinityRun.fillStyle = this.color;
    InfinityRun.fillRect(this.x, this.y, this.width, this
       .height);
};

// ---------- Platform Manager ----------

function PlatformManager() {
    this.maxDistanceBetween = 300;
    this.colors = ['#2ca8c2', '#98cb4a', '#f76d3c', '#
       f15f74', '#5481e6'];


        //first 3 Platforms execept the Starter Platform
    this.first = new Platform({
        x: 300,
        y: InfinityRun.width / 2,
        width: 400,
        height: 70
    })
```

```
194     this.second = new Platform({
            x: (this.first.x + this.first.width) + random(
               this.maxDistanceBetween - 150, this.
               maxDistanceBetween),
196         y: random(this.first.y - 128, InfinityRun.height
               - 80),
            width: 400,
198         height: 70
        })
200     this.third = new Platform({
            x: (this.second.x + this.second.width) + random(
               this.maxDistanceBetween - 150, this.
               maxDistanceBetween),
202         y: random(this.second.y - 128, InfinityRun.height
                - 80),
            width: 400,
204         height: 70
        })
206
        this.first.height = this.first.y + InfinityRun.height
           ;
208     this.second.height = this.second.y + InfinityRun.
           height;
        this.third.height = this.third.y + InfinityRun.height
           ;
210     this.first.color = randomChoice(this.colors);
        this.second.color = randomChoice(this.colors);
212     this.third.color = randomChoice(this.colors);

214     this.colliding = false;

216     this.platforms = [this.first, this.second, this.third
           ];
    }
218
    PlatformManager.prototype.update = function() {
220
        this.first.x -= 3 + InfinityRun.acceleration;
222     if (this.first.x + this.first.width < 0) {
            this.first.width = random(450, InfinityRun.width
               + 200);
224         this.first.x = (this.third.x + this.third.width)
               + random(this.maxDistanceBetween - 150, this.
               maxDistanceBetween);
            this.first.y = random(this.third.y - 32,
               InfinityRun.height - 80);
226         this.first.height = this.first.y + InfinityRun.
               height + 10;
```

```
            this.first.color = randomChoice(this.colors);
228     }

230     this.second.x -= 3 + InfinityRun.acceleration;
        if (this.second.x + this.second.width < 0) {
232         this.second.width = random(450, InfinityRun.width
                + 200);
            this.second.x = (this.first.x + this.first.width)
                + random(this.maxDistanceBetween - 150, this.
               maxDistanceBetween);
234         this.second.y = random(this.first.y - 32,
               InfinityRun.height - 80);
            this.second.height = this.second.y + InfinityRun.
               height + 10;
236         this.second.color = randomChoice(this.colors);
        }
238
        this.third.x -= 3 + InfinityRun.acceleration;
240     if (this.third.x + this.third.width < 0) {
            this.third.width = random(450, InfinityRun.width
               + 200);
242         this.third.x = (this.second.x + this.second.width
               ) + random(this.maxDistanceBetween - 150, this
               .maxDistanceBetween);
            this.third.y = random(this.second.y - 32,
               InfinityRun.height - 80);
244         this.third.height = this.third.y + InfinityRun.
               height + 10;
            this.third.color = randomChoice(this.colors);
246     }

248 };


250

252 // reset / new Game: set Starting Platform Parameters
    PlatformManager.prototype.updateWhenLose = function() {
254
        this.first.x = 300;
256     this.first.color = randomChoice(this.colors);
        this.first.y = InfinityRun.width / random(2, 3);
258     this.second.x = (this.first.x + this.first.width) +
           random(this.maxDistanceBetween - 150, this.
           maxDistanceBetween);
        this.third.x = (this.second.x + this.second.width) +
           random(this.maxDistanceBetween - 150, this.
           maxDistanceBetween);
260
```

```
};

// ---------- Particle System ---------- (Sketch Docs)

function Particle(options) {
    this.x = options.x;
    this.y = options.y;
    this.size = 10;
    this.velocityX = options.velocityX || random(-(InfinityRun.acceleration * 3) + -8, -(InfinityRun.acceleration * 3));
    this.velocityY = options.velocityY || random(-(InfinityRun.acceleration * 3) + -8, -(InfinityRun.acceleration * 3));
    this.color = options.color;
}

Particle.prototype.update = function() {
    this.x += this.velocityX;
    this.y += this.velocityY;
    this.size *= 0.89;
};

Particle.prototype.draw = function() {
    InfinityRun.fillStyle = this.color;
    InfinityRun.fillRect(this.x, this.y, this.size, this.size);
};

/*************************************************/

InfinityRun.setup = function() {

    this.jumpCount = 0;
    this.acceleration = 0;
    this.accelerationTweening = 0;

    this.player = new Player({
        x: 150,
        y: 30,
        width: 32,
        height: 32
    });

    this.platformManager = new PlatformManager();

    this.particles = [];
    this.particlesIndex = 0;
```

```
304     this.particlesMax = 20;
        this.collidedPlatform = null;
306     this.scoreColor = '#181818';
        this.jumpCountRecord = 0;
308
};
310


312
InfinityRun.update = function() {
314         if (GameState == State.Started) {
    this.player.update();
316     switch (this.jumpCount) {
            case 10:
318             this.accelerationTweening = 1;
                this.platformManager.maxDistanceBetween =
                    430;
320             //this.scoreColor = '#076C00';
                break;
322         case 25:
                this.accelerationTweening = 2;
324             this.platformManager.maxDistanceBetween =
                    530;
                //this.scoreColor = '#0300A9';
326             break;
            case 40:
328             this.accelerationTweening = 3;
                this.platformManager.maxDistanceBetween =
                    580;
330             //this.scoreColor = '#9F8F00';
                break;
332     }

334     this.acceleration += (this.accelerationTweening -
            this.acceleration) * 0.01;


336


338     for (i = 0; i < this.platformManager.platforms.length
            ; i++) {
          if (this.player.intersects(this.platformManager.
              platforms[i])) {
340           this.collidedPlatform = this.platformManager.
                  platforms[i];
              if (this.player.y < this.platformManager.
                  platforms[i].y) {
342             this.player.y = this.platformManager.
                    platforms[i].y;
```

```
344                 // Gravity after Collision with Platform
                    this.player.velocityY = 0;
346             }

348             this.player.x = this.player.previousX;
                this.player.y = this.player.previousY;
350
                this.particles[(this.particlesIndex++) % this
                    .particlesMax] = new Particle({
352                 x: this.player.x,
                    y: this.player.y + this.player.height,
354                 color: this.collidedPlatform.color
                });
356
                if (this.player.intersectsLeft(this.
                    platformManager.platforms[i])) {
358                 this.player.x = this.collidedPlatform.x -
                        64;
                    for (i = 0; i < 10; i++) {
360                     // SpawnParticles @PlayerPostion with
                            intersecting Platform Color
                        this.particles[(this.particlesIndex
                            ++) % this.particlesMax] = new
                            Particle({
362                         x: this.player.x + this.player.
                                width,
                            y: random(this.player.y, this.
                                player.y + this.player.height)
                                ,
364                         velocityY: random(-30, 30),
                            color: randomChoice(['#181818',
                                '#181818', this.
                                collidedPlatform.color])
366                     });
                    };
368
                    // bounce player / push him away (effect)
370                 this.player.velocityY = -10 + -(this.
                        acceleration * 4);
                    this.player.velocityX = -20 + -(this.
                        acceleration * 4);
372             } else {

374                 // ---------- Controller ----------
                    // dragging: Mouse click & touch support
376                 if (this.dragging || this.keys.SPACE ||
                        this.keys.UP || this.keys.W) {
```

```
                        this.player.velocityY = this.player.
                            jumpSize;
378                     this.jumpCount++;

380                     if (this.jumpCount > this.
                            jumpCountRecord) {
                          this.jumpCountRecord = this.
                              jumpCount;
382                     }
                    }
384             }
        }
386     };

388     for (i = 0; i < this.platformManager.platforms.length
            ; i++) {
          this.platformManager.update();
390     };

392     for (i = 0; i < this.particles.length; i++) {
          this.particles[i].update();
394     };
}
396
};
398
var selectedItem = 0;
400
InfinityRun.keydown = function() {
402     if (InfinityRun.keys.ESCAPE && GameState==State.
            Started) {
                InfinityRun.clear();
404             GameState = State.Menu;
        } else if (InfinityRun.keys.ESCAPE && GameState==
            State.Menu) {
406             GameState = State.Started;
        }
408     if (InfinityRun.keys.UP) {
                selectedItem = (selectedItem + items.
                    length - 1) % items.length;
410     }
        if (InfinityRun.keys.DOWN) {
412             selectedItem = (selectedItem + 1) % items
                    .length;
        }
414
        if(InfinityRun.keys.ENTER) {
416             callback(selectedItem);
```

```
        }
418
}
420
Menu = function() {
422
        //this.backgroundCallback = null;
424 }


426
//---------- Draw ----------
428
InfinityRun.draw = function() {
430         if(GameState == State.Started) {
    this.player.draw();
432
    for (i = 0; i < this.platformManager.platforms.length
       ; i++) {
434         this.platformManager.platforms[i].draw();
    };
436
    //Draw particles
438 for (i = 0; i < this.particles.length; i++) {
        this.particles[i].draw();
440 };

442         //Draw menu --TODO prototype
        } else if (GameState == State.Menu) {
444
        this.title = "InfinityRun";
446         items = ["Play", "Settings", "Highscore"];

448         callback = function(numItem) { if (numItem == 0)
           GameState=State.Started };
        this.height = InfinityRun.height;
450         this.width = InfinityRun.width;
        this.size = 120;
452
        var lingrad = this.createLinearGradient(0,0,0,
           this.height);
454         lingrad.addColorStop(0, '#000');
        lingrad.addColorStop(1, '#023');
456         this.fillStyle = lingrad;
        this.fillRect(0,0,this.width, this.height)
458
        this.textAlign = "center";
460         this.fillStyle = "White";
```

```
462         var height = 150;

464         if (this.title) {
                this.font = Math.floor(this.size*1.3).
                    toString() + "px Times New Roman";
466             this.fillText(this.title, this.width/2,
                    height);
                height+= height;
468         }

470         for (var i = 0; i < items.length; ++i)
            {
472             var size = Math.floor(this.size*0.8);
                if (i == selectedItem)
474             {
                    this.fillStyle = "#A9F5F2";
476                 size = this.size+5;
                }
478             this.font = size.toString() + "px Times
                    New Roman";
                height += this.size;
480             this.fillText(items[i], InfinityRun.width
                    /2, height);
                this.fillStyle = "White";
482         }

484         }


486

488     //Debug
        if (debug) {
490         this.font = '12pt Arial';
            this.fillStyle = '#181818';
492         this.fillText('Record: ' + this.jumpCountRecord,
                this.width - 150, 33);
            this.fillStyle = this.scoreColor;
494         this.fillText('Jumps: ' + this.jumpCount, this.
                width - 150, 50);
            this.fillText('Distance: ' + 0/* -TODO- */, this.
                width - 150, 65);
496         this.fillText('GameState: ' + GameState, this.
                width - 150, 80);
        }
498
    };
500
    InfinityRun.resize = function() {
```

```
502     /* todo Windowscale optimization
         *
504      *
         */
506 }
```

A.3. game.css

```
body{
2   background: #e3e3e3;
    overflow: hidden;
4   margin: 0;
    padding: 0;
6   text-align: center;
}
8 #container{
    /*margin-top: 10%;*/
10  display: inline-block;
}
12 canvas{
    background: #cecece;
14  border: 1px solid #181818;
}
```

A.4. index.html

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" ↩
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" ↩
    lang="en">
3 <head>
    <meta http-equiv="Content-Type" content="text/html; ↩
       charset=utf-8">
5       <script type="text/javascript" src="js/sketch.min ↩
          .js" charset="utf-8"></script>

7   <title>Infinity Run</title>

9   <link href="css/game.css" rel="stylesheet" type="text ↩
       /css">
</head>
11 <body>
<!-- Game div -->
13 <div id="container">

15 </div>
<script type="text/javascript" src="js/game.js" charset=" ↩
    utf-8"></script>
```

```
17 <script type="text/javascript" src="js/menu.js" charset="
       utf-8"></script>
   </body>
19 </html>
```