Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent	:	**Gabriela Mai**
Vorgelegt am	:	11. Januar 2017
Vorgelegt von	:	Gruppe 4

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

Inhaltsverzeichnis

Inhaltsverzeichnis ii

Abbildungsverzeichnis iii

Tabellenverzeichnis v

1 Einleitung 1

1.1 Team 1

1.2 Rollenverteilung 2

1.3 Spielidee 2

1.3.1 Spielkonzept 2

1.3.2 Entwurfsskizze 3

1.3.3 Erforderliche Software 4

2 Phasen 5

2.1 Entwurf und Anforderungen 5

2.1.1 Funktionale Anforderungen 5

2.1.2 Nicht funktionale Anforderungen 6

2.1.3 Projektplan 6

2.1.4 Releaseplan 7

2.2 Implementation - Zwischenstand 8

2.2.1 Erfüllte Anforderungen 8

2.2.2 Nicht erfüllte Anforderungen 8

2.2.3 Das Spiel 9

2.2.4 Bibliothek 10
2.2.5 Code 10
2.2.6 Nächste Ziele 16
2.3 Implementation - Endstand 16
2.3.1 Spielkonzept Änderungen 16
2.3.2 Funktionsdiagramm 16
2.3.3 Grafiken 18
2.3.4 Code Änderungen 19
2.3.5 Das Spiel - Endstand 23
2.3.6 Sounds 25
2.4 Test 25
2.4.1 Umfang 25
2.4.2 Testplan 25
2.4.3 Testberichte 26
2.4.4 Testberichte analysieren 31
2.4.5 Browsertest 31
2.4.6 Bug Behebung 31
3 Ausblick 33
4 Fazit 35
Literaturverzeichnis 37
Eidesstattliche Erklärung 39
A Anhang 41
A.1 Github Changelog 41
A.2 game.js 45
A.3 game.css 80
A.4 index.html 81

Abbildungsverzeichnis

Abbildung 1: Florian Durli 1
Abbildung 2: Jannik Ivosevic 1
Abbildung 3: Johannes But 1
Abbildung 4: Marco Mayer 1
Abbildung 5: Koray Ektekin 1
Abbildung 6: Entwurfsskizze 3
Abbildung 7: Startbildschirm 9
Abbildung 8: Das Spiel 9
Abbildung 9: Funktionsdiagramm 17
Abbildung 10: Logo 18
Abbildung 11: Startbildschirm - Endstand 23
Abbildung 12: Das Spiel - Endstand 23

Tabellenverzeichnis

Tabelle 1: Rollenverteilung 2
Tabelle 2: Phase 1: Entwurf und Anforderungen 6
Tabelle 3: Phase 2: Implementierung 6
Tabelle 4: Phase 3: Test 6
Tabelle 5: Phase 4: Dokumentation und Präsentation 7
Tabelle 6: Releaseplan 7
Tabelle 7: Funktionsbeschreibung 18
Tabelle 8: Menü-Steuerung 24
Tabelle 9: Spiel-Steuerung 24
Tabelle 10: Sound Links 25
Tabelle 11: Bugs 31
Tabelle 12: Github Namen 41

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

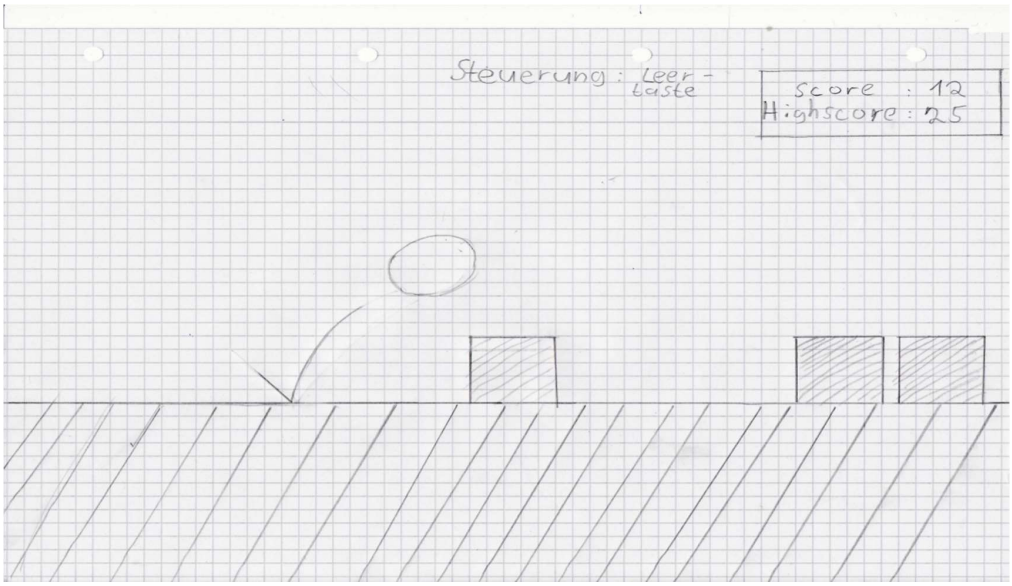

Abbildung 6.: Entwurfsskizze

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.

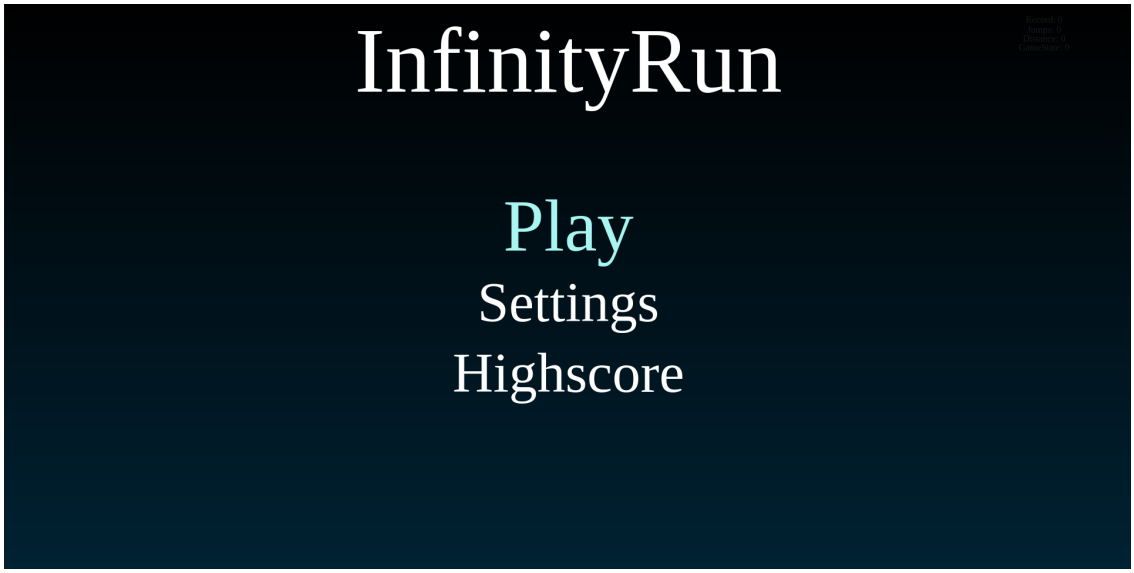

Abbildung 7.: Startbildschirm

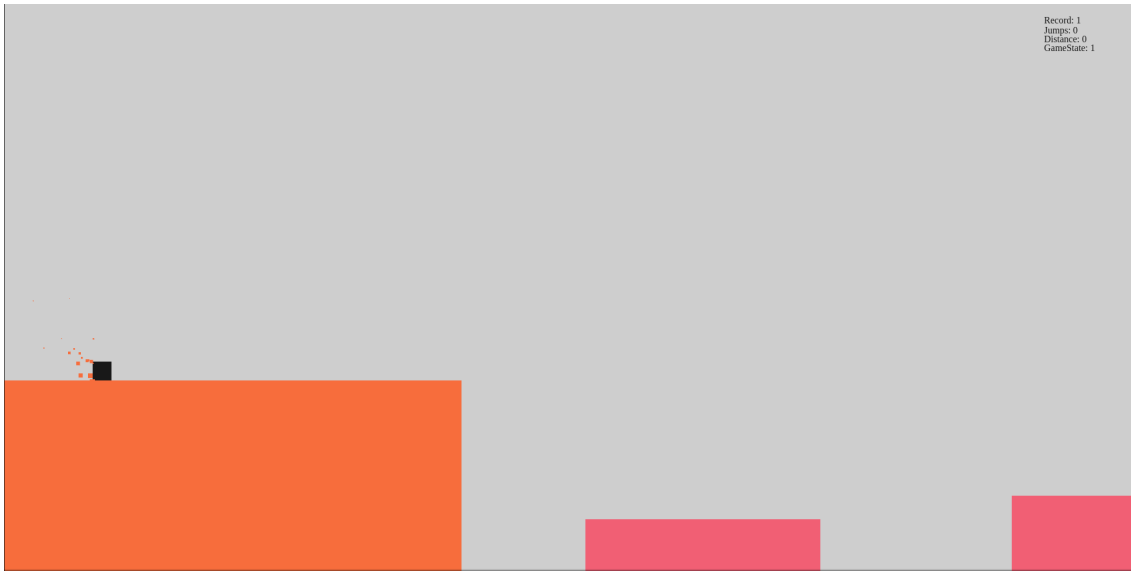

Abbildung 8.: Das Spiel

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
- Spielfigur ändert sich nicht nach Aufprall

2.3.2. Funktionsdiagramm

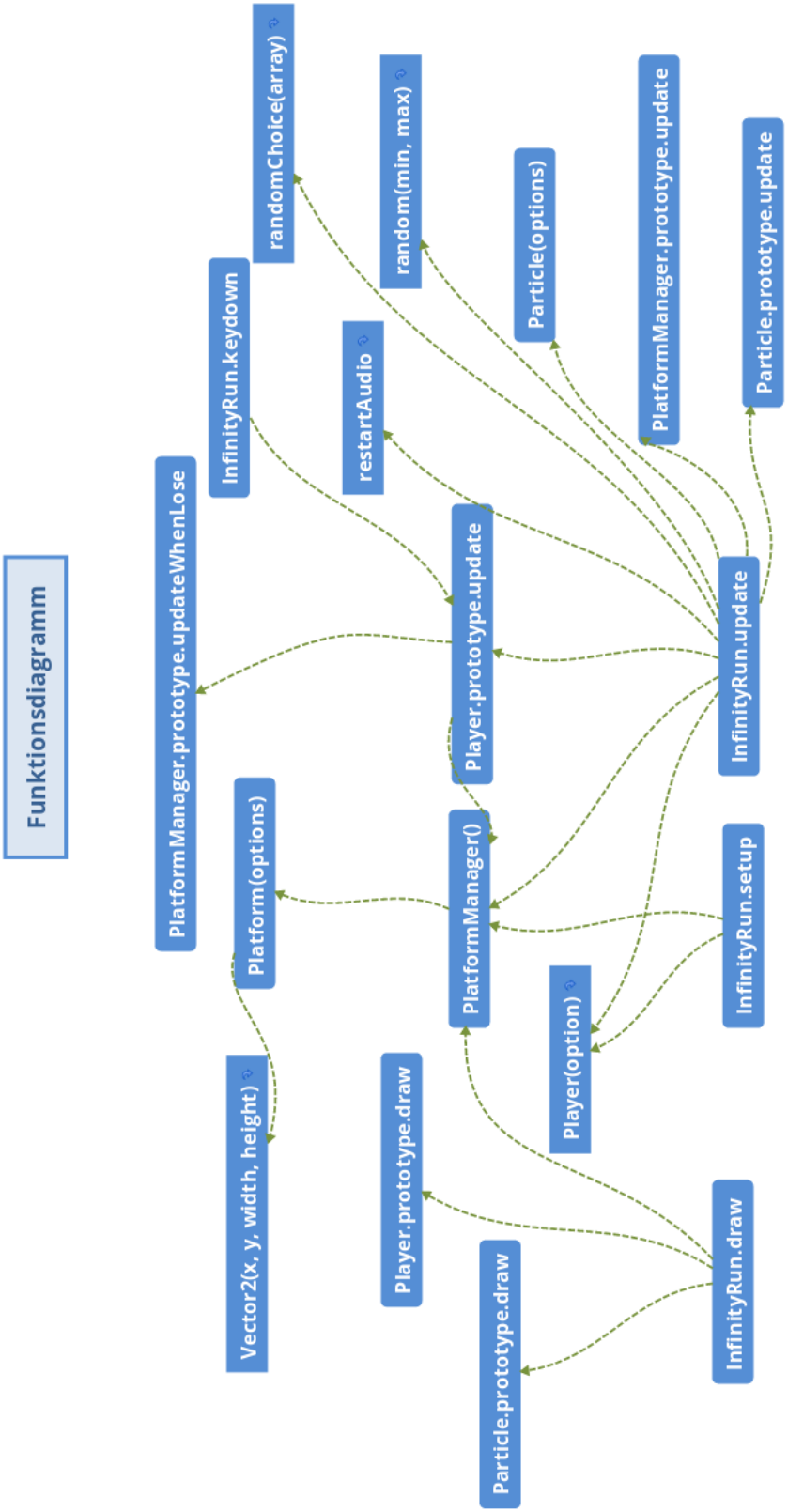

Abbildung 9.: Funktionsdiagramm

Beschreibung der Funktionen aus Abbildung 9

Funktion	Erklärung
InfinityRun.draw	Spielfläche wird gezeichnet
InfinityRun.setup	Grundeinstellungen des Spiels
InfinityRun.update	Aktualisierung der Spielfläche
Particle.prototype.update	Aktualisierung der Partikel
PlatformManager.prototype.update	Neue Position der Plattformen
Particle(Option)	Einstellungen der Partikel
random(min, max)	Erstellen der Zufallszahl
randomChoice(array)	Zufälliger Wert aus dem Array
restartAudio	Neustand der Audiosequenz
InfinityRun.keydown	Festlegung der Spieltasten
PlatformManager.prototype.updateWhenLose	Setzt die Plattformen zurück
Player.prototype.update	Aktualisierung der Spielfigur
PlatformManager()	Verwalten der Plattformen
Platform(options)	Erzeugt eine Plattform
Player(option)	Erstellt die Spielfigur
Vector2(x, y, width, height)	Verwaltungen der Koordinaten
Player.prototype.draw	Zeichnen der Spielfigur
Particle.prototype.draw	Zeichnen der Partikel

Tabelle 7.: Funktionsbeschreibung

2.3.3. Grafiken

2.3.3.1. Spiel

Derzeit haben wir keine Grafiken implementiert, da unsere Objekte und Hintergründe mittels Canvas gezeichnet werden.

2.3.3.2. Logo

Wir haben für unser Spiel zusätzlich ein Logo erstellt dieses Logo wurde mit Gimp erstellt und wird in unserem Spiel, wie auch auf dem Deckblatt dieser Dokumentation angezeigt.

Abbildung 10.: Logo

2.3.4. Code Änderungen

2.3.4.1. Musik

Die ausgewählte Musik wurde per Audio-Element in JavaScript implementiert. Es gibt zwei Audio-Elemente, einen für den Hintergrund und einen für die Effekte. Folgende Code Beispiele zeigen diese Elemente.
Erstellen der Audio-Elemente:

```
var bgaudio = document.getElementById('backgroundmusic');
var fxaudio = document.getElementById('fxaudio');
```

Zugriff auf Element per Funktion zum Neustarten der Musik:

```
function restartAudio()
{
        // Check for audio element support.
        if (window.HTMLAudioElement)
        {
                try
                {
                        // Tests the paused attribute and set state.
                        if (bgaudio.ended)
                        {
                                bgaudio.currentTime = 0;
                                bgaudio.play();
                        }
                }
                catch (e)
                {
                        // Fail silently but show in F12 developer tools console
                        if(window.console && console.error("Error:" + e));
                }
        }
}
```

Beispiel für die Audio Wiedergabe:

```
if (this.dragging || this.keys.SPACE || this.keys.UP || this.keys.W)
```

```
{
        this.player.velocityY = this.player.jumpSize;
        this.jumpCount++;
        fxaudio.pause();
        fxaudio.src = 'sounds/jump.wav';
        fxaudio.load();
        fxaudio.play();
}
```

2.3.4.2. Hintergrund

Unser Hintergrund stellt in drei verschiedenen Layern Hochhäuser dar. Diese Hochhäuser werden ähnlich wie unsere Plattformen generiert und von links nach rechts auf dem Bildschirm dargestellt. Der Hintergrund reagiert zusätzlich auf Sprünge der Spielfigur. Inspiriert von "Canvas Parallax Skyline" [dis] Erstellt die Hochhäuser mit ihren Eigenschaften:

```
Street.prototype.populate = function()
{
        var newHeight, newWidth, results, totalWidth;
        totalWidth = 0;
        results = [];
        while (totalWidth <= InfinityRun.width + (this.
           width.max * 2))
        {
                newWidth = round(random(this.width.min,
                   this.width.max));
                newHeight = round(random(this.height.min,
                    this.height.max));
                this.alltowers.push(new Tower({
                        layer: this.layer,
                        x: this.alltowers.length === 0 ?
                           0 : this.alltowers[this.
                           alltowers.length - 1].x + this
                           .alltowers[this.alltowers.
                           length - 1].width,
                        y: InfinityRun.height - newHeight
                           ,
                        width: newWidth,
                        height: newHeight,
                        color: this.color
                }));
                results.push(totalWidth += newWidth);
        }
        return results;
};
```

Aktualisieren der Hochhäuser, für neues erscheinen am rechten Spielrand:

```
Street.prototype.update = function()
{
```

```
3           var firstTower, lastTower, newHeight, newWidth;
            if (InfinityRun.accelerationTweening==0)
5           {
                    this.x-=((150) * this.speed) * dt;
7           }
            else
9           {
                    this.x -= ((InfinityRun.
                       accelerationTweening*330) * this.speed
                       ) * dt;
11          }

13          firstTower = this.alltowers[0];
            if (firstTower.width + firstTower.x + this.x < 0)
15          {
                    newWidth = round(random(this.width.min,
                       this.width.max));
17                  newHeight = round(random(this.height.min,
                        this.height.max));
                    lastTower = this.alltowers[this.alltowers
                       .length - 1];
19                  firstTower.reset({
                            layer: this.layer,
21                          x: lastTower.x + lastTower.width,
                            y: InfinityRun.height - newHeight
                               ,
23                          width: newWidth,
                            height: newHeight,
25                          color: this.color
                    });
27          return this.alltowers.push(this.alltowers.shift()
               );
            }
29  };
```

2.3.5. Das Spiel - Endstand

Hier werden zwei Screenshots des derzeitigen Spiels dargestellt. In der Abbildung 11 zu sehen, ist der endgültige Startbildschirm des Spiels. Hier gibt es verschiedene Auswahlmöglichkeiten, die das Spielerlebnis ergänzen. In der Abbildung 12 zu sehen ist der endgültige Stand des Spiels. Der Hintergrund reagiert hierbei auf den Sprung des Spielers und ein Partikeleffekt hinter dem Spieler ist ebenfalls implementiert.

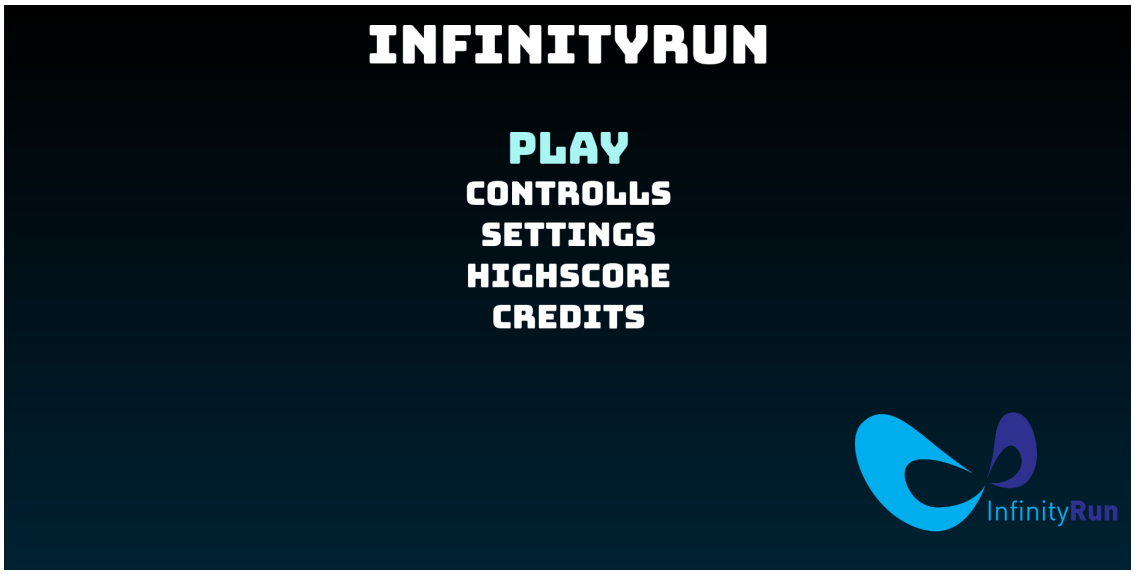

Abbildung 11.: Startbildschirm - Endstand

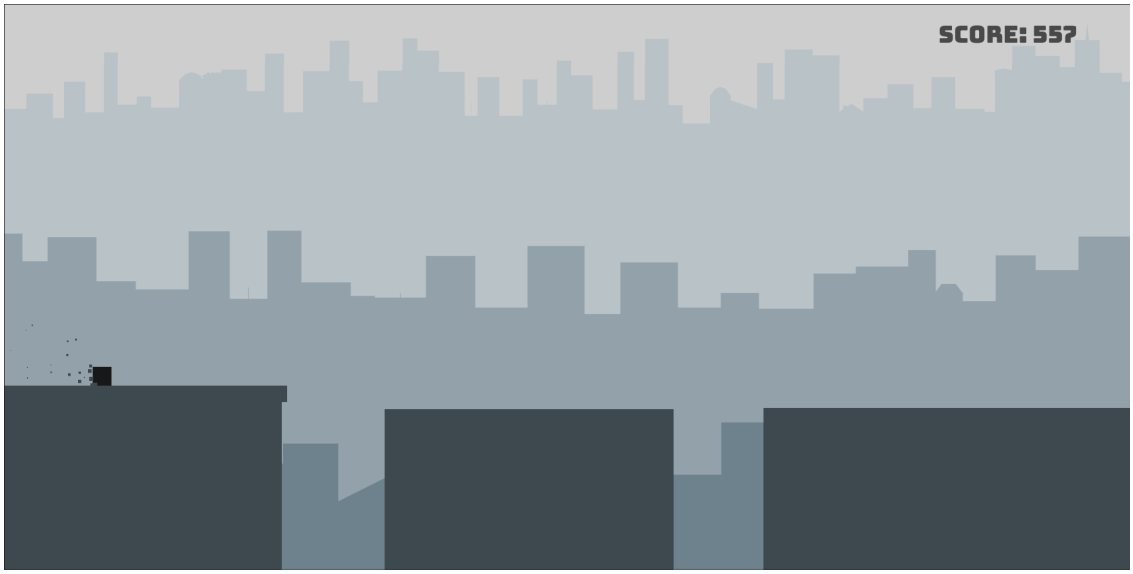

Abbildung 12.: Das Spiel - Endstand

2.3.5.1. Steuerung

Menü-Steuerung:

Taste	**Funktion**
Pfeiltaste Hoch,Runter	Hoch,Runter navigieren im Menü
Enter	Bestätigen
ESC	Zurück ins Hauptmenü

Tabelle 8.: Menü-Steuerung

Spiel-Steuerung:

Taste	**Funktion**
Pfeiltaste Hoch,Runter	Sprung und schneller Fallen lassen
Leertaste, W, Mausklick links	Sprung
ESC	Zurück ins Hauptmenü

Tabelle 9.: Spiel-Steuerung

2.3.5.2. Webserver und Smartphone

Zusätzlich habe wir das Spiel nun auf einem freien Webserver(Quelle:[Are]) implementiert. Dies ist vor allem für unsere Testphase wichtig, in der wir nur noch den Link zum testen schicken müssen. Der Webserver macht es möglich das Spiel auch problemlos auf dem Smartphone zu spielen. Somit unterstützen wir unterschiedliche Plattformen mit unserem Spiel um den Spiel Spaß auch unterwegs zu genießen.
Spiel: `http://infinityrun.freevar.com/`

2.3.6. 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 10.: Sound Links

2.4. Test

In der Phase "Test" erstellen wir einen Testplan der auch an dritte ausgegeben wird.

2.4.1. Umfang

Das Spiel wird ausführlich von uns getestet und zusätzlich ein Feedback von dritten eingeholt.

2.4.2. Testplan

2.4.2.1. Testumgebung

- Browser
- Betriebssystem
- System
- Auflösung

2.4.2.2. Modultest

- Steuerung
- Sound

- Spieloberfläche
- Spielverlauf

2.4.2.3. Systemtest

- Performance

2.4.3. Testberichte

Testberichte werden an dritte ausgehändigt. Nachdem wir die Berichte zurück bekommen haben, werden wir diese digitalisieren. Unser Testbericht haben wir zusätzlich noch angehängt.

Testbericht

Name: Felix Duffner, Beruf: Zimmermann Datum: 28.12.2016

1 Testumgebung

Browser	Chrome
Betriebssystem	Windows 7
System	Laptop
Auflösung	1280x720

2 Modultest

2.1 Steuerung

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Ja	
Intuitiv?	Ja	

2.2 Sound

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Ja/Nein	Musik hört auf und startet irgendwann neu
Passend?	Ja	

2.3 Spieloberfläche

	Erfüllt Ja/Nein?	**Anmerkungen**
Farbgestaltung passend?	Ja	
Darstellung der Komponenten	Ja	

2.4 Spielverlauf

	Erfüllt Ja/Nein?	**Anmerkungen**
Spielkonzept passend?	Ja	
Funktioniert der Highscore	Ja	
Schwierigkeit passend?	Nein	Zu langsam

3 Systemtest

Performance: Ab und zu kleinere Ruckler. Sonst gut.

Testbericht

Name: Maurice Ketterer, Beruf: Verwaltungs Azubi Datum: 30.12.2016

1 Testumgebung

Browser	Firefox
Betriebssystem	Windows 8.1
System	Desktop-PC
Auflösung	Full-HD

2 Modultest

2.1 Steuerung

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Ja	
Intuitiv?	Ja	Wie ähnliche Mini-Games

2.2 Sound

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Ja/Nein	Effekt Sound lässt sich nicht ausstellen
Passend?	Ja	

2.3 Spieloberfläche

	Erfüllt Ja/Nein?	**Anmerkungen**
Farbgestaltung passend?	Ja/Nein	Könnte Bunter sein
Darstellung der Komponenten	Ja	

2.4 Spielverlauf

	Erfüllt Ja/Nein?	**Anmerkungen**
Spielkonzept passend?	Ja	
Funktioniert der Highscore	Ja	
Schwierigkeit passend?	Ja	

3 Systemtest

Performance: Läuft gut.

Testbericht

Name: Tomas Müller, Beruf: Mechatroniker Datum: 03.01.2017

1 Testumgebung

Browser	Chrome
Betriebssystem	Windows 10
System	Alter Laptop
Auflösung	Full-HD

2 Modultest

2.1 Steuerung

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Ja	
Intuitiv?	Ja	

2.2 Sound

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Nein	Aussetzer!
Passend?	Nein	

2.3 Spieloberfläche

	Erfüllt Ja/Nein?	**Anmerkungen**
Farbgestaltung passend?	Ja	
Darstellung der Komponenten	Ja	

2.4 Spielverlauf

	Erfüllt Ja/Nein?	**Anmerkungen**
Spielkonzept passend?	Ja	
Funktioniert der Highscore	Ja	
Schwierigkeit passend?	Ja	

3 Systemtest

Performance: Kaum Spielbar durch ruckeln.

Testbericht

Name: Gruppe 4, Beruf: Studenten Datum: 21.12.2016

1 Testumgebung

Browser	Chrome
Betriebssystem	Linux
System	Laptop
Auflösung	Full-HD

2 Modultest

2.1 Steuerung

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Ja	
Intuitiv?	Ja	

2.2 Sound

	Erfüllt Ja/Nein?	**Anmerkungen**
Funktioniert	Ja/Nein	Probleme bei dauerhafter Wiedergabe
Passend?	Ja	

2.3 Spieloberfläche

	Erfüllt Ja/Nein?	**Anmerkungen**
Farbgestaltung passend?	Ja	
Darstellung der Komponenten	Ja	

2.4 Spielverlauf

	Erfüllt Ja/Nein?	**Anmerkungen**
Spielkonzept passend?	Ja	
Funktioniert der Highscore	Ja	
Schwierigkeit passend?	Ja	

3 Systemtest

Performance: Nicht durchgehend Flüssig Spielbar.

2.4.4. Testberichte analysieren

Anhand der Testberichte und aus unserem eigenen Test sind uns einige Bugs aufgefallen. Diese Bugs werden wir in einer Liste mit Priorität abarbeiten.

2.4.4.1. Bugliste

Priorität	**Bug**	**Reproduzierbar?**	**Ursache**
1	Performance	Ja	Hintergrund
2	Hintergrundmusik	Ja/Nein	HTML-Audio Element
3	Effektlautstärke	Ja	Nicht einstellbar
3	Menüsound	Ja	Aktualisierung

Tabelle 11.: Bugs

Legende:

- Priorität 1: Sehr Wichtig
- Priorität 2: Wichtig
- Priorität 3: Optional

2.4.5. Browsertest

Unseren Browsertest haben wir mittels eines Online Tools gemacht. Browsershots.org [bro] erlaubte uns verschiedene Browser und auch verschiedene Betriebssysteme zu testen. So kam das Ergebnis heraus dass unser Spiel auf den gängigsten Browsern spielbar ist, jedoch einige Ausnahmen wie z.B. Konqueror und Dillo nicht funktionieren.

2.4.6. Bug Behebung

Wir haben die Bugs nach Priorität behoben und die Nachfolgende Auflistung orientiert sich an dieser.

2.4.6.1. Performance - Priorität 1

Wir konnten diesen Bug beheben in dem wir in das Menü eine Grafikoption implementiert haben in der man zwischen "Low, Mid, High" auswählen kann. Je nach Option werden mehrere Layer im Hintergrund erzeugt und auch die Komplexität der Häuser im Hintergrund ändert sich.

2.4.6.2. Hintergrundmusik - Priorität 2

Die Hintergrundmusik haben wir nun durch eine Audio-Bibliothek behoben. Diese Technologie ist ausgereifter als die Musik per HTML-Audioelemente zu implementieren.

2.4.6.3. Effektlautstärke - Priorität 3

Die Effektlautstärke wurde von uns nun perfekt zum Spiel angepasst.

2.4.6.4. Menüsound - Priorität 3

Diesen Fehler konnten bei der Behebung der Hintergrundmusik zusätzlich lösen.

3. Ausblick

In der Zukunft könnte das Spiel zusätzliche Schwierigkeitsgrade, eine Auswahl von Spielfiguren, ein Umgekehrter Spielverlauf und einen Globalen Score der bei Neustart verfügbar ist könnte implementiert werden. Wir haben uns bis zum derzeitigen Stand des Spiels rein um das Grundspiel und die Stabilität gekümmert, somit blieben solche "Nice to have" Features aus. Zusätzlich könnte dieses Spiel vor allem auf mobilen Plattformen als App Angeboten werden können.

4. Fazit

Der Informatik Workshop hat uns gezeigt das Teamwork eines der wichtigsten Eigenschaften einer solchen Projektarbeit ist. Wir haben gelernt gemeinsam an einem Strang zu ziehen und hatten keinerlei Probleme unseren Projektplan einzuhalten. Selbst Probleme bei der Umsetzung des Spiel konnten wir wie geplant umsetzen. Das benutzen von Github erleichterte das Arbeiten im Team ungemein, so konnten wir parallel an Doku und Spiel arbeiten. Abschließend können wir sagen, dass wir ein nettes kleines Spiel auf die Beine gestellt haben.

Literaturverzeichnis

[Are] Area, Free Web H.: *Free Hosting* http://freevar.com/

[bro] browsershots.org: *Browsershoots* http://browsershots.org/

[dis] dissimulate: *Skyline* https://codepen.io/dissimulate/pen/CAzlt

[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 11. Januar 2017 Florian Durli

FURTWANGEN, den 11. Januar 2017 Jannik Ivosevic

FURTWANGEN, den 11. Januar 2017 Johannes But

FURTWANGEN, den 11. Januar 2017 Marco Mayer

FURTWANGEN, den 11. Januar 2017 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 12.: Github Namen

Changelog:

```
86c56e1 Slay3r         2016-12-16 Doku final

c4876c7 Florian Durli  2016-12-16 Controlls

9ce6d99 Florian Durli  2016-12-16 Doku Server,Steuerung

e1c7138 r4qtor         2016-12-15 acceleration fix

188bcdb r4qtor         2016-12-15 Delete sketch.minlesbar.js (RE: unbenutzt)

55b3162 r4qtor         2016-12-15 bug fixes, 'Farbänderung', kleines 'Hindernis'

0f5b9cc butjo          2016-12-15 HighScore gefixed Plattformen erhöht

414cbc9 butjo          2016-12-14 Creditsverbesserungen

fc96b76 butjo          2016-12-14 Credits eingefügt
```

```
19 3b990da butjo          2016-12-14 Audioeinstellungen ↩
       implementiert

21 6fb84f9 Slay3r         2016-12-14 Doku Quellen angepasst

23 02cecd1 butjo          2016-12-14 Merge branch 'master' of ↩
       https://github.com/Slay3r/InfinityRun

25 682c0d2 butjo          2016-12-14 Farbschema geändert

27 449cf21 Slay3r         2016-12-14 Doku

29 288bb25 butjo          2016-12-14 Merge

31 2003a79 butjo          2016-12-13 Dacharten erweitert

33 42e9a8f butjo          2016-12-13 Code verbessert/↩
       Optiemiert

35 67cc215 Slay3r         2016-12-13 Doku logo

37 2d7d953 butjo          2016-12-12 Favicon und Logo im Menü

39 2b8a139 ans77          2016-12-12 Logos hinzugefügt

41 92b7b28 Slay3r         2016-12-12 Code Cleanup

43 30b0c6f Slay3r         2016-12-12 Clean up

45 37f6e94 butjo          2016-12-12 Favicon hinzugefügt

47 35801ac butjo          2016-12-11 Hintergrund ist nun abhä↩
       nig von der Fenstergröße und reagiert auf den Player ↩
       nicht mehr auf die Maus sowie noch die Dacharten ↩
       erweitert und bugs behoben

49 8d06d8a butjo          2016-12-09 skyline eingefügt

51 4e52883 Slay3r         2016-12-07 Doku + Alte Dateien

53 2c3491b butjo          2016-12-05 Sounds hinzugefügt ↩
       teilweise noch ein paar Bugs und dirty code

55 2e6dfbe butjo          2016-11-30 Dirty Code mit versuch ↩
       den background zu implementieren files sind mit prefix↩
        back gekennzeichnetund liegen im Hauptverzeichnis.
```

```
57 dda171b Slay3r          2016-11-30 Changelog geändert

59 ffe660b Slay3r          2016-11-30 Ausblick/Fazit ↩
       auskommentiert

61 01e309a Slay3r          2016-11-30 Sounds

63 3a05368 Slay3r          2016-11-30 Neue Dokumentation ↩
       hinzufügen

65 7c3596c Slay3r          2016-11-30 Neue Dokumentation ↩
       hinzufügen

67 bfdb05c Slay3r          2016-11-30 entfernen der ↩
       Dokumentation

69 f944707 r4qtor          2016-11-25 Querlesung - Marco

71 8b6bdde Florian Durli   2016-11-25 Abgabe

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

75 b6d7a09 butjo           2016-11-24 Rechtschreibkorrekturen

77 1faa558 r4qtor          2016-11-24 Delete phasen.tex root/

79 51f4a79 r4qtor          2016-11-24 updated phasen.tex

81 62af4b8 Krusher999      2016-11-24 Letzten Codes ↩
       geschrieben

83 93b8965 Florian Durli   2016-11-23 fix

85 9027301 Florian Durli   2016-11-23 Changelog finale Lösung

87 1392138 Florian Durli   2016-11-23 Jojos Description in ↩
       LaTeX

89 25ff037 butjo           2016-11-23 Beschreibung der ↩
       Codeteile in der phasen.tex von Johannes

91 1b2348c Florian Durli   2016-11-23 Changelog Additionally

93 cf467dc Florian Durli   2016-11-23 Changelog Additionally

95 6db1ad6 butjo           2016-11-23 Merge branch 'master' of ↩
       https://github.com/Slay3r/InfinityRun
```

```
97  b924713 Slay3r          2016-11-23 Generated Changelog

99  69dd747 butjo           2016-11-23 Merge branch 'master' of
        https://github.com/Slay3r/InfinityRun

101 203ae2e Slay3r          2016-11-23 Bilder des Spiels

103 d274cfc Slay3r          2016-11-23 Anforderungen

105 a0909ac Slay3r          2016-11-23 Literaturverzichniss

107 19e2f3d Slay3r          2016-11-23 Doku update

109 21889f9 Florian Durli   2016-11-22 Add Changelog

111 743c95a butjo           2016-11-16 Formatierte sketch.min.
        js

113 d64d254 butjo           2016-11-16 Endlose
        Schwierigkeitserhöhung

115 3707b94 butjo           2016-11-16 Präsentation
        Zwischenstand

117 f23c9be Slay3r          2016-11-16 Präsentation ü
        berarbeitet

119 53aa72e butjo           2016-11-16 Präsentation und test

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

123 275bd69 r4qtor          2016-11-15 tiny cleanup & incl.
        menu

125 11a5e55 Slay3r          2016-11-09 Basis implementation

127 c783850 Slay3r          2016-11-09 Bilder hinzugefügt

129 d88d0d2 Slay3r          2016-11-09 Dokumentations Basis

131 39b4705 Florian         2016-10-19 Initial Struktur der
        Ordner und files

133 093797e Florian Durli   2016-10-19 Delete Requirements

135 56c4aae Florian         2016-10-19 doc Ordner
```

```
137 b3b83fd Florian       2016-10-19 Requirements hinzugefügt

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

A.2. game.js

```
/* todo: cleanup (dirty code),
 *
 * Put static values / vars into initialization function
 *
 * ---------------------------------
 * Design / Graphics
 *
 * Parallax Background?
 *
 * ---------------------------------
 * Menu
 * Menu draw in Input & draw prototypes
 * Handle / Manage CSS or HTML variables from JavaScript (Fullscreen,...)
 * ---------------------------------
 *
 * 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/
 */
//testweise rausgenommen verändert nix
//var i = 0;

var debug = false;

var State = { Menu:0, Controlls:1,Started:2, Paused:3, Over:4 };
var GameState = State.Menu;
var MainMenu;
var MenuTab = {Main:0, Controlls:1,Settings:2, Highscore:3, Credits:4};
```

```
var curMenuTab = MenuTab.Main;

var vgaquality = 0; //0=low 1=mid 2=high
var settingsItem = 0; // 0=audiosettings 1= Graphicsettings 2= filtersettings
var setFilters = true; //set filter on or off
var freshStart = 0;

//timer
var s = 0,
ms = 0,
playTimer = false;

var highScore = new Array(10);
highScore = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];


//-------------------------------------------------------------------------

//vars background
var Tower, Street, dt, Town;
//Building

Town = [];

dt = 1;
 var jumpheight = 0
 //logoimage
var bglogo = new Image();
bglogo.src = 'image/logo.png';
//-------------------------------------------------------------------------

var bgFX;
var sfx;

//playSound
function playMenuFX () {
        menuFX = createjs.Sound.play("MainMenu", {loop:-1});
}

function playbgFX (soundID) {
        bgFX = createjs.Sound.play(soundID, {loop:-1});
}


function playSFX (soundID) {
        sfx = createjs.Sound.play(soundID);
```

```
}

//-------------------------------------------------------------------------

// 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')
});
var qs = document.querySelector('canvas');

//-------------------------------------------------------------------------

//bg func
Tower = function(config)
{
        return this.reset(config);
};

Tower.prototype.reset = function(config)
{
        this.layer = config.layer;
        this.x = config.x;
        this.y = config.y;
        this.width = config.width;
        this.height = config.height;
        this.color = config.color;
                this.summitTop = floor(random(0, 15)) ===
                    0;
        this.summitTopWidth = random(this.width * .01,
           this.width * .07);
        this.summitTopHeight = random(10, 20);
```

```
        this.singleroofTop = floor(random(0, 10)) === 0;
126     this.singleroofTopHeight = this.width / random(2,
            4);
        this.singleroofTopDirection = round(random(0, 1))
            === 0;
128             this.normalTop = !this.singleroofTop &&
                    floor(random(0, 10)) === 0;
        this.normalTopHeight = this.width / random(2, 4);
130     this.normalTopchimney = round(random(0, 1)) ===
            0;
                this.coneTop = !this.singleroofTop && !
                    this.normalTop && floor(random(0, 10))
                     === 0;
132     this.coneTopHeight = this.width / random(3, 4);
                this.coneTopWidth = this.width / random
                    (1, 2);
134     this.coneTopeflat = round(random(0, 1)) === 0;
                this.companyTop = !this.singleroofTop &&
                    !this.summitTop && !this.radioTop && !
                    this.normalTop && floor(random(0, 10))
                     === 0;
136     this.companyTopHeight = this.width / random(4, 6)
            ;
        this.companyTopcount = 4;//round(random(3, 6));
138     this.radioTop = !this.summitTop && floor(random
            (0, 10)) === 0;
        this.radioTopWidth = this.layer / 2;
140     return this.radioTopHeight = random(6, 30);
};
142
Tower.prototype.render = function()
144 {
        InfinityRun.fillStyle = InfinityRun.strokeStyle =
            this.color;
146     InfinityRun.lineWidth = 2;
        InfinityRun.beginPath();
148     InfinityRun.rect(this.x, this.y, this.width, this
            .height);
        InfinityRun.fill();
150     InfinityRun.stroke();

152 if (vgaquality > 0){ //graphics higher then low
        if (this.singleroofTop)
154     {
                InfinityRun.beginPath();
156             InfinityRun.moveTo(this.x, this.y);
                InfinityRun.lineTo(this.x + this.width,
                    this.y);
```

```
158                 if (this.singleroofTopDirection)
                    {
160                         InfinityRun.lineTo(this.x + this.
                               width, this.y - this.
                               singleroofTopHeight);
                    }
162                 else
                    {
164                         InfinityRun.lineTo(this.x, this.y
                               - this.singleroofTopHeight);
                    }
166                 InfinityRun.closePath();
                    InfinityRun.fill();
168                 InfinityRun.stroke();
            }
170
            if (this.normalTop)
172         {
                    InfinityRun.beginPath();
174                 InfinityRun.moveTo(this.x, this.y);
                    InfinityRun.lineTo(this.x + this.width,
                       this.y);
176                         InfinityRun.lineTo(this.x + (this
                               .width/2), this.y-this.
                               normalTopHeight);
                    InfinityRun.closePath();
178                 InfinityRun.fill();
                    InfinityRun.stroke();
180                         if(this.normalTopchimney)
                            {
182                                 InfinityRun.beginPath();
                                    InfinityRun.moveTo(this.x
                                       +(this.width/5), this.
                                       y);
184                                 InfinityRun.lineTo(this.x
                                       +(this.width/5), this.
                                       y- 0.8*(this.
                                       normalTopHeight));
                                    InfinityRun.lineTo(this.x
                                        + (this.width/5)+(
                                       this.width/10), this.y
                                       - 0.8*(this.
                                       normalTopHeight));
186                                 InfinityRun.lineTo(this.x
                                        + (this.width/5)+(
                                       this.width/10), this.y
                                       );
                                    InfinityRun.closePath();
```

```
188                                 InfinityRun.fill();
                                    InfinityRun.stroke();
190                         }
                }
192             if (vgaquality > 1 && this.coneTop)
                {
194                     InfinityRun.beginPath();
                        InfinityRun.moveTo(this.x, this.y);
196                     InfinityRun.lineTo(this.x + (this.width-
                           this.coneTopWidth)/2, this.y-this.
                           coneTopHeight);
                                if(!this.coneTopeflat)
198                             {
                                    InfinityRun.lineTo(this.x
                                       +(this.width/2), this.
                                       y-(this.coneTopHeight
                                       *1.3));
200                             }
                                InfinityRun.lineTo(this.x + ((
                                   this.width-this.coneTopWidth)
                                   /2)+this.coneTopWidth, this.y-
                                   this.coneTopHeight);
202                             InfinityRun.lineTo(this.x + this.
                                   width, this.y);
                        InfinityRun.closePath();
204                     InfinityRun.fill();
                        InfinityRun.stroke();
206
                }
208             if (vgaquality > 1 && this.companyTop)
                {
210                     var ctc = 1;
                        while (ctc<=this.companyTopcount)
212                     {
                                InfinityRun.beginPath();
214                             InfinityRun.moveTo(this.x , this.
                                   y);
                                InfinityRun.lineTo(this.x + ctc*(
                                   this.width/this.
                                   companyTopcount), this.y-this.
                                   companyTopHeight);
216                             InfinityRun.lineTo(this.x + ctc*(
                                   this.width/this.
                                   companyTopcount), this.y+this.
                                   companyTopHeight);
                                InfinityRun.closePath();
218                             InfinityRun.fill();
                                InfinityRun.stroke();
```

```
220                             ctc++;
                        }
222             }
                if (vgaquality > 1 && this.summitTop)
224             {
                        InfinityRun.beginPath();
226                     InfinityRun.moveTo(this.x + (this.width /
                            2), this.y - this.summitTopHeight);
                        InfinityRun.lineTo(this.x + (this.width /
                            2) + this.summitTopWidth, this.y);
228                     InfinityRun.lineTo(this.x + (this.width /
                            2) - this.summitTopWidth, this.y);
                        InfinityRun.closePath();
230                     InfinityRun.fill();
                        InfinityRun.stroke();
232             }

234             if (vgaquality > 1 && this.radioTop)
                {
236                     InfinityRun.beginPath();
                        InfinityRun.moveTo(this.x + (this.width /
                            2), this.y - this.radioTopHeight);
238                     InfinityRun.lineTo(this.x + (this.width /
                            2), this.y);
                        InfinityRun.lineWidth = this.
                           radioTopWidth;
240                     return InfinityRun.stroke();
                }
242 }
    };
244
    Street = function(config)
246 {
                this.x = 0;
248             this.alltowers = [];
                this.layer = config.layer;
250             this.width = {
                min: config.width.min,
252             max: config.width.max
    };
254
    this.height = {
256             min: config.height.min,
                max: config.height.max
258 };

260     this.speed = config.speed;
        this.color = config.color;
```

```
262     this.populate();
        return this;
264 };

266 Street.prototype.populate = function()
    {
268         var newHeight, newWidth, results, totalWidth;
            totalWidth = 0;
270         results = [];
            while (totalWidth <= InfinityRun.width + (this.
               width.max * 2))
272         {
                    newWidth = round(random(this.width.min,
                       this.width.max));
274                 newHeight = round(random(this.height.min,
                        this.height.max));
                    this.alltowers.push(new Tower({
276                 layer: this.layer,
                    x: this.alltowers.length === 0 ? 0 : this
                       .alltowers[this.alltowers.length - 1].
                       x + this.alltowers[this.alltowers.
                       length - 1].width,
278                 y: InfinityRun.height - newHeight,
                    width: newWidth,
280                 height: newHeight,
                    color: this.color
282                 }));
                    results.push(totalWidth += newWidth);
284         }
            return results;
286 };

288 Street.prototype.update = function()
    {
290         var firstTower, lastTower, newHeight, newWidth;
            if (InfinityRun.accelerationTweening==0)
292         {
                    this.x-=((150) * this.speed) * dt;
294         }
            else
296         {
      this.x -= ((InfinityRun.accelerationTweening*330) *
         this.speed) * dt;
298         }

300         firstTower = this.alltowers[0];
            if (firstTower.width + firstTower.x + this.x < 0)
302         {
```

```
                newWidth = round(random(this.width.min,
                    this.width.max));
304             newHeight = round(random(this.height.min,
                    this.height.max));
                lastTower = this.alltowers[this.alltowers
                    .length - 1];
306             firstTower.reset({
                        layer: this.layer,
308                     x: lastTower.x + lastTower.width,
                        y: InfinityRun.height - newHeight
                            ,
310                     width: newWidth,
                        height: newHeight,
312                     color: this.color
                });
314         return this.alltowers.push(this.alltowers.shift()
                );
            }
316 };

318 Street.prototype.render = function()
    {
320         var i;
            i = this.alltowers.length;
322         InfinityRun.save();
                InfinityRun.translate(this.x, (
                    InfinityRun.height - (InfinityRun.
                    height-(-jumpheight*0.5)-400)) / 20 *
                    this.layer);
324
                while (i--) {
326             this.alltowers[i].render(i);
            }
328         return InfinityRun.restore();
    };
330 //-------------------------------------------------------

    //-------- Vector [Get/Set] Functions ----------
332
    //Set X,Y,Width,Height
334 function Vector2(x, y, width, height) {
        this.x = x;
336     this.y = y;
        this.width = width;
338     this.height = height;
        this.previousX = 0;
340     this.previousY = 0;
    };
```

```
342
344 // Set X,Y
    Vector2.prototype.setPosition = function(x, y) {
346
        this.previousX = this.x;
348     this.previousY = this.y;

350     this.x = x;
        this.y = y;
352
    };
354 // Set X
    Vector2.prototype.setX = function(x) {
356
        this.previousX = this.x;
358     this.x = x;

360 };

362 // Set Y
    Vector2.prototype.setY = function(y) {
364
        this.previousY = this.y;
366     this.y = y;

368 };

370 // Collision / Intersection Top
    Vector2.prototype.intersects = function(obj) {
372
        if (obj.x < this.x + this.width && obj.y < this.y + ↩
           this.height &&
374          obj.x + obj.width > this.x && obj.y + obj.height ↩
               > this.y) {
             return true;
376     }

378     return false;
    };
380
    // Collision / Intersection Left
382 Vector2.prototype.intersectsLeft = function(obj) {

384     if (obj.x < this.x + this.width && obj.y < this.y + ↩
           this.height) {
             return true;
386     }
```

```
388     return false;
};
390
//---------- Player ----------
392
function Player(options) {
394
    this.setPosition(options.x, options.y);
396     this.width = options.width;
    this.height = options.height;
398     this.velocityX = 0;
    this.velocityY = 0;
400     this.jumpSize = -13;
    this.color = '#181818';
402
}
404
Player.prototype = new Vector2;
406
Player.prototype.update = function() {
408     // Gravity
    this.velocityY += 1;
410         //um bg zu ändern
        jumpheight=(this.y);
412     this.setPosition(this.x + this.velocityX, this.y + ↩
       this.velocityY);

414     if (this.y > InfinityRun.height || this.x + this.↩
       width < 0) {
         this.x = 150;
416         this.y = 50;
         this.velocityX = 0;
418         this.velocityY = 0;
         InfinityRun.jumpCount = 0;
420         InfinityRun.acceleration = 0;
         InfinityRun.accelerationTweening = 0;
422         InfinityRun.scoreColor = '#181818';
         InfinityRun.platformManager.maxDistanceBetween = ↩
            350;
424
                //InfinityRun.pause();
426
                //highscore update↩
                    ------------------------------------------------------------↩

428             if (timePassed>highScore[0]){
                        var help =highScore[0];
```

```
430                             var help2=highScore[1];
                                highScore[0]=timePassed;
432                             for(i=1; i<=9;i++){
                                help2 = highScore[i];
434                             highScore[i]=help;
                                help=help2;
436
                                }
438
                        }
440             InfinityRun.platformManager.updateWhenLose();

442                     playSFX("Crash");
                        bgFX.stop();
444                     difficulty = 0;
                        ms = 0;
446         }

448         if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
                InfinityRun.keys.W || InfinityRun.dragging) &&
               this.velocityY < -8) {
                this.velocityY += -0.75;
450
        }
452

454         if (InfinityRun.keys.DOWN) {
                this.velocityY += 1;
456         }

458 };

460 Player.prototype.draw = function() {
        InfinityRun.fillStyle = this.color;
462     InfinityRun.fillRect(this.x, this.y, this.width, this
           .height);
    };
464
    // ---------- Platforms ----------
466
    function Platform(options) {
468     this.x = options.x;
        this.y = options.y;
470     this.width = options.width;
        this.height = options.height;
472     this.previousX = 0;
        this.previousY = 0;
474     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 = ['#3D494F'];

        //first 3 Platforms execept the Starter Platform
    this.first = new Platform({
        x: 300,
        y: 600,
        width: 400,
        height: 70
    })
    this.second = new Platform({
        x: (this.first.x + this.first.width) + random(this.maxDistanceBetween - 150, this.maxDistanceBetween),
        y: 570,//y: random(this.first.y - 128, InfinityRun.height - 80),
        width: 400,
        height: 70
    })
    this.third = new Platform({
        x: (this.second.x + this.second.width) + random(this.maxDistanceBetween - 150, this.maxDistanceBetween),
        y: 540,//y: random(this.second.y - 128, InfinityRun.height - 80),
        width: 400,
        height: 70
    })

    this.first.height = this.first.y + InfinityRun.height;
    this.second.height = this.second.y + InfinityRun.height;
    this.third.height = this.third.y + InfinityRun.height;
```

```
    this.first.color = randomChoice(this.colors);
514     this.second.color = randomChoice(this.colors);
    this.third.color = randomChoice(this.colors);
516
    this.colliding = false;
518
    this.platforms = [this.first, this.second, this.third
        ];
520 }

522 PlatformManager.prototype.update = function() {

524     this.first.x -= 3 + InfinityRun.acceleration;
    if (this.first.x + this.first.width < 0) {
526         this.first.width = random(450, 800);
        this.first.x = (this.third.x + this.third.width)
            + random(this.maxDistanceBetween - 150, this.
            maxDistanceBetween);
528         //this.first.y = random(this.third.y - 32,
            InfinityRun.height - 80);
                this.first.y = random(this.third.y - 32,
                    InfinityRun.height - 200);
530         this.first.height = this.first.y + InfinityRun.
            height + 10;
        this.first.color = randomChoice(this.colors);
532     }

534     this.second.x -= 3 + InfinityRun.acceleration;
    if (this.second.x + this.second.width < 0) {
536         this.second.width = random(450, 800);
        this.second.x = (this.first.x + this.first.width)
             + random(this.maxDistanceBetween - 150, this.
            maxDistanceBetween);
538                 //this.first.y = random(this.third.y -
                    32, InfinityRun.height - 80);
        this.second.y = random(this.first.y - 32,
            InfinityRun.height - 200);
540         this.second.height = this.second.y + InfinityRun.
            height + 10;
        this.second.color = randomChoice(this.colors);
542     }

544     this.third.x -= 3 + InfinityRun.acceleration;
    if (this.third.x + this.third.width < 0) {
546         this.third.width = random(450, 800);
        this.third.x = (this.second.x + this.second.width
            ) + random(this.maxDistanceBetween - 150, this
            .maxDistanceBetween);
```

```
548                 //this.first.y = random(this.third.y -
                        32, InfinityRun.height - 80);
            this.third.y = random(this.second.y - 32,
                InfinityRun.height - 200);
550         this.third.height = this.third.y + InfinityRun.
                height + 10;
            this.third.color = randomChoice(this.colors);
552     }

554 };


556

558 // reset / new Game: set Starting Platform Parameters
    PlatformManager.prototype.updateWhenLose = function() {
560
        this.first.x = 300;
562     this.first.color = randomChoice(this.colors);
            this.first.y = 700;
564     //this.first.y = InfinityRun.width / random(2, 3);
            this.second.y = 650;
566         this.third.y = 600;
        this.second.x = (this.first.x + this.first.width) +
            random(this.maxDistanceBetween - 150, this.
            maxDistanceBetween);
568     this.third.x = (this.second.x + this.second.width) +
            random(this.maxDistanceBetween - 150, this.
            maxDistanceBetween);


570 };

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

574 function Particle(options) {
        this.x = options.x;
576     this.y = options.y;
        this.size = 10;
578     this.velocityX = options.velocityX || random(-(
            InfinityRun.acceleration * 3) + -8, -(InfinityRun.
            acceleration * 3));
        this.velocityY = options.velocityY || random(-(
            InfinityRun.acceleration * 3) + -8, -(InfinityRun.
            acceleration * 3));
580     this.color = options.color;
    }
582
    Particle.prototype.update = function() {
584     this.x += this.velocityX;
```

```
    this.y += this.velocityY;
586     this.size *= 0.89;
};
588
Particle.prototype.draw = function() {
590     InfinityRun.fillStyle = this.color;
    InfinityRun.fillRect(this.x, this.y, this.size, this.
        size);
592 };

594 /*********************************************/


596
InfinityRun.setup = function() {
598
    this.jumpCount = 0;
600     this.acceleration = 0;
    this.accelerationTweening = 0;
602     this.player = new Player({
        x: 150,
604         y: 30,
        width: 32,
606         height: 32
    });
608
        setTimeout(function (){
610         playMenuFX("MainMenu");

612         }, 200);


614
    this.platformManager = new PlatformManager();
616
    this.particles = [];
618     this.particlesIndex = 0;
    this.particlesMax = 20;
620     this.collidedPlatform = null;
    this.scoreColor = '#181818';
622     this.jumpCountRecord = 0;
        //-----------------------------------------
624     var i, results;
    i = 3;
626     results = [];
    while (i--) {
628       results.push(Town.push(new Street({
        layer: i + 1,
630         width: {
          min: (i + 1) * 20,
```

```
632             max: (i + 1) * 50
            },
634             height: {
              min: InfinityRun.height-200 - (i * round(
                  InfinityRun.height/3)),
636             max: InfinityRun.height-50 - (i * round(
                  InfinityRun.height/3))
            },
638             speed: (i + 1) * .003,
            color: 'hsl( 200, ' + (((i + 1) * 1) + 10) + '%, '
                + (75 - (i * 13)) + '% )'
640       }));
      }
642         return results;
          //------------------------------------------

644


646 };
  //------------------------------------------------------
648 InfinityRun.clear = function() {
      return InfinityRun.clearRect(0, 0, InfinityRun.width,
          InfinityRun.height);
650   };
  //------------------------------------------------------
652
  Array.max = function( array ){
654     return Math.max.apply( Math, array );
  };
656
  var sc = 0;
658 var sx = 0;
  var sy = 0;
660 var sz = 0;
  var invertRunning = false;
662 var sunsetRunning = false;
  timer = setInterval(function() {
664   if (!playTimer) return;
      ms += 1;
666         sc += 1;
          sy += 1;
668         sz += 1;
          if (sc == 99) {
670                 s+=1;
                  sx+=1;
672                 sc = 0;
          }
674
          updateTimer();
```

```
676
}, 1);
678
function randomIntFromInterval(min,max)
680 {
        var milliseconds = new Date().getMilliseconds();
682     return Math.floor(Math.random()*(max-min+1)+min);
}
684

686 var rng = random(115,124);
var rng2 = random(13,16)
688 function updateTimer() {
  if (s==rng) {
690         if(!invertRunning) {
            invertRunning = true;
692         rng = random(30,50);
            if(setFilters) //toggle filters
694         {
                qs.classList.toggle('invertFilter');
696         }
            }
698         s=0;
  }
700 if (sx==rng2) {
            if(!sunsetRunning) {
702         sunsetRunning = true;
            rng2 = random(2,5);
704         if(setFilters) //toggle filters
            {
706             qs.classList.toggle('sunsetFilter');
            }
708         }
      sx=0;
710 }
  if (sz==70) {
712         invertRunning = false;
            sz = 0;
714 }
  if (sy==70) {
716         invertRunning = false;
            sy = 0;
718 }


720
        timePassed = ms;
722 }
```

```
724 function toggleTimer() {
    if (!playTimer) {
726   //s = 0, ms = 0;

728   updateTimer();
    }
730 playTimer = !playTimer;
  }
732
  var difficulty = 0;
734
  InfinityRun.update = function() {
736       if (GameState == State.Started) {
          //-------------------------------------------------------
738       //clear func bg

740       var i, results;
    dt = InfinityRun.dt < .1 ? .1 : InfinityRun.dt / 16;
742 dt = dt > 5 ? 5 : dt;
    i = Town.length;
744 results = [];
    while (i--) {
746   results.push(Town[i].update(i));
    }
748       //-------------------------------------------------------

750       if(document.hasFocus()) {
              toggleTimer();
752       } else {
              toggleTimer();
754       }


756
    this.player.update();
758       if(difficulty ==0) {

760           playbgFX("Main1");
              difficulty = 1;
762       } else if (timePassed>1000 && timePassed < 5000
            && difficulty == 1) {
              this.accelerationTweening = 1.5;
764       this.platformManager.maxDistanceBetween = 430;

766           bgFX.stop();
              playbgFX("Main2");
768           playSFX("LevelUP");

770           difficulty = 2;
```

```
        } else if (timePassed>5000 && timePassed < 10000
           && difficulty == 2) {
772             this.accelerationTweening = 2.7;
        this.platformManager.maxDistanceBetween = 530;
774

776             bgFX.stop();
                playbgFX("Main3");
778             playSFX("LevelUP");

780             difficulty = 3;
        } else if (timePassed>10000 && timePassed < 15000
            && difficulty == 3) {
782             this.accelerationTweening = 3.8;
        this.platformManager.maxDistanceBetween = 580;
784

786             bgFX.stop();
                playbgFX("Main4");
788             playSFX("LevelUP");

790             difficulty = 4;
        } else if (timePassed>15000 && timePassed < 20000
            && difficulty == 4) {
792             this.accelerationTweening = 4.4;
                this.PlatformManager.maxDistanceBetween =
                    610;
794

796             playSFX("LevelUP");

798             difficulty = 5;
        } else if (timePassed>20000 && difficulty == 5) {
800             this.accelerationTweening = 5;
                this.PlatformManager.maxDistanceBetween =
                    620;
802

804             playSFX("LevelUP");

806             difficulty = 6;
        }
808     this.acceleration += (this.accelerationTweening -
           this.acceleration) * 0.01;

810     for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
          if (this.player.intersects(this.platformManager.
```

```
            platforms[i])) {
812          this.collidedPlatform = this.platformManager.
                platforms[i];
             if (this.player.y < this.platformManager.
                platforms[i].y) {
814              this.player.y = this.platformManager.
                    platforms[i].y;

816              // Gravity after Collision with Platform
                 this.player.velocityY = 0;
818          }

820          this.player.x = this.player.previousX;
             this.player.y = this.player.previousY;
822
             this.particles[(this.particlesIndex++) % this.
                .particlesMax] = new Particle({
824              x: this.player.x,
                 y: this.player.y + this.player.height,
826              color: this.collidedPlatform.color
             });
828
             if (this.player.intersectsLeft(this.
                platformManager.platforms[i])) {
830              this.player.x = this.collidedPlatform.x -
                     64;
                 for (i = 0; i < 10; i++) {
832                  // SpawnParticles @PlayerPostion with
                         intersecting Platform Color
                     this.particles[(this.particlesIndex
                        ++) % this.particlesMax] = new
                        Particle({
834                      x: this.player.x + this.player.
                            width,
                         y: random(this.player.y, this.
                            player.y + this.player.height)
                            ,
836                      velocityY: random(-30, 30),
                         color: randomChoice(['#181818',
                            '#181818', this.
                            collidedPlatform.color])
838                  });
                 };
840
                 // bounce player / push him away (effect)
842              this.player.velocityY = -10 + -(this.
                    acceleration * 4);
                 this.player.velocityX = -20 + -(this.
```

```
                    acceleration * 4);
844
                                    if (timePassed > this.
                                       jumpCountRecord) {
846                             this.jumpCountRecord = timePassed
                                   ;
                        }
848                 } else {

850                     // ---------- Controller ----------
                        // dragging: Mouse click & touch support
852                     if (this.dragging || this.keys.SPACE ||
                           this.keys.UP || this.keys.W) {
                            this.player.velocityY = this.player.
                               jumpSize;
854                         this.jumpCount++;

856                                             playSFX("Jump");
                        }
858                 }
            }
860     };

862     for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
            this.platformManager.update();
864     };

866     for (i = 0; i < this.particles.length; i++) {
            this.particles[i].update();
868     };
            //-------------------------------------------------------
870         //bg
            return results;
872         //-------------------------------------------------------
}
874
};
876


878


880 var selectedItem = 0;
    var audioItem = 10;
882


884 InfinityRun.keydown = function() {
        if (InfinityRun.keys.ESCAPE && GameState==State.
```

```
        Started) {
886                 InfinityRun.clear();
                    GameState = State.Menu;
888
                    bgFX.setPaused(true);
890                 //playMenuFX("MainMenu");
                    menuFX.setPaused(false);
892                 toggleTimer();

894                 freshStart = 1;

896         } else if (InfinityRun.keys.ESCAPE && GameState==
               State.Menu && curMenuTab==MenuTab.Main &&
               freshStart == 1) {
                    GameState = State.Started;
898                 toggleTimer();
                    //menuFX.stop();
900                 bgFX.setPaused(false);
                    menuFX.setPaused(true);
902


904
         }else if (InfinityRun.keys.ESCAPE && GameState==
               State.Menu && curMenuTab==MenuTab.Controlls) {
906                 curMenuTab = MenuTab.Main;
      }else if (InfinityRun.keys.ESCAPE && GameState==State
         .Menu && curMenuTab==MenuTab.Settings) {
908                 curMenuTab = MenuTab.Main;
         } else if (InfinityRun.keys.ESCAPE && GameState==
               State.Menu && curMenuTab==MenuTab.Highscore) {
910                 curMenuTab = MenuTab.Main;
         } else if (InfinityRun.keys.ESCAPE && GameState==
               State.Menu && curMenuTab==MenuTab.Credits) {
912                 curMenuTab = MenuTab.Main;
         }
914
         //main menu controls
916      if (InfinityRun.keys.UP && GameState == State.
            Menu) {
                    selectedItem = (selectedItem + items.
                       length - 1) % items.length;
918      }
         if (InfinityRun.keys.DOWN && GameState == State.
            Menu) {
920                 selectedItem = (selectedItem + 1) % items
                       .length;
         }
922      //general settings choose
```

```
        if (InfinityRun.keys.UP && curMenuTab==MenuTab.
           Settings && settingsItem!=0) {
924             settingsItem-=1;
        }
926     if (InfinityRun.keys.DOWN && curMenuTab==MenuTab.
           Settings && settingsItem!=2) {
                settingsItem+=1;
928     }
        // settings audio change
930     if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.
           Settings && audioItem !=0 && settingsItem ==0)
            {
                audioItem = (audioItem + items.length -
                   1) % items.length;
932             createjs.Sound.volume -= 0.1;
        }
934
        if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab
           .Settings && audioItem !=10 && settingsItem
           ==0) {
936             audioItem = (audioItem + 1) % items.
                   length;
                createjs.Sound.volume += 0.1;
938     }
        //graphic settings change
940     if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.
           Settings && vgaquality!=0 && settingsItem ==1)
            {
                vgaquality-=1;
942     }

944     if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab
           .Settings && vgaquality!=2 && settingsItem
           ==1) {
                vgaquality+=1;
946     }
        //filter settings change
948     if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.
           Settings && !setFilters && settingsItem ==2) {
                setFilters=true;
950     }

952     if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab
           .Settings && setFilters && settingsItem ==2) {
                setFilters=false;
954             if (invertRunning) {
                        qs.classList.toggle('invertFilter
                           ');
```

```
956                 }
                    if (sunsetRunning) {
958                         qs.classList.toggle('sunsetFilter
                            ');
                    }
960         }
            if(InfinityRun.keys.ENTER && GameState == State.
               Menu) {
962                 callback(selectedItem);
            }
964
    }
966
    Menu = function() {
968 }

970
    //---------- Draw ----------
972

974 InfinityRun.draw = function() {

976         if(GameState == State.Started) {
            //-------------------------------------------------------------------------

978         //bg draw

980         var i, results;
        i = Town.length;
982     results = [];
            if(vgaquality>1){
984     while (i--) {
                results.push(Town[i].render(i));
986
        }
988         }
        if (vgaquality==1)
990         {
        i-=1; //only upper towers will be drawn
992         while (i--) {
                results.push(Town[i+1].render(i+1));
994
        }
996         }
        if (vgaquality<1)
998         {
        i-=2; //only one street will be drawn
1000        while (i--) {
```

```
                results.push(Town[i+1].render(i+1));
        }

            }
            //———————————————————————————————————————————————

        this.player.draw();

        for (i = 0; i < this.platformManager.platforms.length
            ; i++) {
            this.platformManager.platforms[i].draw();
        };

        //Draw particles
        for (i = 0; i < this.particles.length; i++) {
            this.particles[i].draw();
        };

            //draw score
            this.font = '30pt Bungee';
            this.textAlign="left";
            this.fillStyle = '#494949';
            this.fillText('Score: ', this.width - 330, 65);
            this.fillText(timePassed , this.width - 170, 65);

            /*
             * Main Menu
             *
             */
            } else if (GameState == State.Menu && curMenuTab
                ==MenuTab.Main) {

            this.title = "InfinityRun";
            items = ["Play","Controls", "Settings", "
                Highscore", "Credits"];

            callback = function(numItem) { //if (numItem ==
                0) GameState=State.Started

            switch (numItem) {
              case 0:
                    GameState=State.Started;
                    toggleTimer();
                    //bgFX.stop();
                    menuFX.setPaused(true);
                    bgFX.setPaused(false);
```

```
1044                    break;
                case 1:
1046                  curMenuTab=MenuTab.Controlls;
                    break;
1048                case 2:
                        curMenuTab=MenuTab.Settings;
1050                    break;
                case 3:
1052                  curMenuTab=MenuTab.Highscore;
                        break;
1054                case 4:
                  curMenuTab=MenuTab.Credits;
1056                    break;

1058          }



1062          };
              this.height = InfinityRun.height;
1064          this.width = InfinityRun.width;
              this.size = 70;
1066
              var lingrad = this.createLinearGradient(0,0,0,
                 this.height);
1068          lingrad.addColorStop(0, '#000');
              lingrad.addColorStop(1, '#023');
1070          this.fillStyle = lingrad;
              this.fillRect(0,0,this.width, this.height)
1072
              this.textAlign = "center";
1074          this.fillStyle = "White";

1076          var height = 100;
              //logo
1078          this.drawImage(bglogo,this.width-500,this.height
                 -300);
             //-----------------------------------
1080          if (this.title) {
                    this.font = Math.floor(this.size*1.3).
                       toString() + "px Bungee";
1082                this.fillText(this.title, this.width/2,
                       height);
                    height+= height;
              }
1084
1086          for (var i = 0; i < items.length; ++i)
              {
```

```
1088                var size = Math.floor(this.size*0.8);
                    if (i == selectedItem)
1090                {
                            this.fillStyle = "#A9F5F2";
1092                        size = this.size+5;
                    }
1094                this.font = size.toString() + "px Bungee";
                    height += this.size;
1096                this.fillText(items[i], InfinityRun.width/2, height);
                    this.fillStyle = "White";
1098            }
            //-------------------------------------------------------------------

1100        //bg dd <-- ??
            return results;
1102        //-------------------------------------------------------------------


1104        /*
             * Settings Tab
1106         *
             */
1108        }else if (GameState == State.Menu && curMenuTab==MenuTab.Controlls){
                    this.title = "Controlls";
1110        items = highScore;


1112
            callback = function(volume) { //if (numItem == 0)
                GameState=State.Started
1114
            switch (volume) {
1116
            }
1118



1120
            };
1122        this.height = InfinityRun.height;
            this.width = InfinityRun.width;
1124
            var lingrad = this.createLinearGradient(0,0,0,this.height);
1126         lingrad.addColorStop(0, '#000');
            lingrad.addColorStop(1, '#023');
1128         this.fillStyle = lingrad;
```

```
            this.fillRect(0,0,this.width, this.height, items[
                i]);
1130
            this.textAlign = "center";
1132        this.fillStyle = "White";

1134        var width = 10;
            var height = 150;
1136
            if (this.title) {
1138                this.font = Math.floor(this.size*1.3).
                        toString() + "px Bungee";
                    this.fillText(this.title, this.width/2,
                        150);
1140                height+= height;
            }
1142        var distanceText = 50
            this.font = Math.floor(40).toString() + "px
                Bungee";
1144        this.textAlign = "left";
            //Names
1146        this.fillText("Menu:", this.width/5, 300);
            this.font = Math.floor(20).toString() + "px
                Bungee";
1148        this.fillText("[ESC] - Menu", this.width/5, 300+
                distanceText);
            this.fillText("[Arrow up/down] - To navigate",
                this.width/5, 300+2*distanceText);
1150        this.fillText("[Enter] - Accept", this.width/5,
                300+3*distanceText);

1152        this.font = Math.floor(40).toString() + "px
                Bungee";
            this.textAlign = "left";
1154        //Names
            this.fillText("Game:", this.width/5, 300+5*
                distanceText);
1156        this.font = Math.floor(20).toString() + "px
                Bungee";
            this.fillText("[ESC] - Menu", this.width/5,
                300+6*distanceText);
1158        this.fillText("[W], [Arrow up], [Leertaste] -
                Jump", this.width/5, 300+7*distanceText);
            this.fillText("[Arrow down] - increase falling
                speed", this.width/5, 300+8*distanceText);
1160        }
            else if (GameState == State.Menu && curMenuTab==
                MenuTab.Settings){
```

```
1162
        this.title = "Settings";
1164    items = [0,10, 20, 30, 40, 50, 60, 70, 80, 90 , 100];

1166    callback = function(volume) { //if (numItem == 0) GameState=State.Started

1168    switch (volume) {

1170    }


1172


1174    };

1176    this.height = InfinityRun.height;
        this.width = InfinityRun.width;
1178
        var lingrad = this.createLinearGradient(0,0,0, this.height);
1180    lingrad.addColorStop(0, '#000');
        lingrad.addColorStop(1, '#023');
1182    this.fillStyle = lingrad;
        this.fillRect(0,0,this.width, this.height, items[i]);
1184
        this.textAlign = "center";
1186    this.fillStyle = "White";

1188    var width = 10;
        var height = 10;
1190    var posx = 130;
        var posy = 380;
1192    this.space = 15;
        this.heightincr = 4;
1194
        if (this.title) {
1196            this.font = Math.floor(this.size*1.3).toString() + "px Bungee";
                this.fillText(this.title, this.width/2, 150);
1198            height+= height;
        }
1200
        this.font = "55px Bungee";
1202    if (settingsItem==0) {
        this.fillStyle = "#A9F5F2";
```

```
1204            }
                this.fillText('Volume', 240, 300);
1206
                this.fillStyle = "White";
1208


1210            for (var i = 0; i < items.length; ++i) {
                        var size = Math.floor(this.size*0.8);
1212                    if (i == audioItem && settingsItem==0)
                        {
1214                            this.fillStyle = "#A9F5F2";
                                //size = this.size+5;
1216                    }
                        this.font = size.toString() + "px Bungee"
                           ;
1218                    posx += this.space;
                        posy -= this.heightincr;
1220                    height += this.heightincr;

1222                    items[i] = this.fillRect(posx,posy,width,
                           height);

1224                    //this.fillText(items[i], InfinityRun.
                           width/2, height);
                        this.fillStyle = "White";
1226
                }
1228            //-----------------------------------------------------------------------------

                //Graphic Settings
1230            this.fillStyle = "White";
                if (settingsItem==1) {
1232            this.fillStyle = "#A9F5F2";
                }
1234            this.fillText('Graphics', 240, 500);
                this.fillStyle = "White";
1236            switch (vgaquality) {
                        //Low
1238              case 0:
                    this.fillText('Mid', 500,600 );
1240                    this.fillText('High', 700, 600);
                        if (settingsItem == 1){
1242                    this.fillStyle = "#A9F5F2";
                        }
1244                    this.fillText('Low', 240, 600);
                        break;
1246                    //mid
                  case 1:
```

```
1248
                this.fillText('Low', 240, 600);
1250            this.fillText('High', 700, 600);
                if (settingsItem == 1){
1252            this.fillStyle = "#A9F5F2";
                }
1254            this.fillText('Mid', 500,600 );

1256          break;
                //high
1258        case 2:
              this.fillText('Mid', 500,600 );
1260            this.fillText('Low', 240, 600);
                if (settingsItem == 1){
1262            this.fillStyle = "#A9F5F2";
                }
1264            this.fillText('High', 700, 600);

1266            break;

1268      }
          //-------------------------------------------------------------------------------

1270      //Filter settings
          this.fillStyle = "White";
1272      if (settingsItem==2) {
          this.fillStyle = "#A9F5F2";
1274      }
          this.fillText('Filters', InfinityRun.width-300, 300);
1276      this.fillStyle = "White";
          if(setFilters)
1278      {
            this.fillText('Off', InfinityRun.width-200, 400);
1280        if (settingsItem==2)
            {
1282        this.fillStyle = "#A9F5F2";
            }
1284      this.fillText('On', InfinityRun.width-350, 400);
          }
1286      else
          {
1288        this.fillText('On', InfinityRun.width-350, 400);
          if (settingsItem==2)
            {
1290        this.fillStyle = "#A9F5F2";
```

```
1292            }
          this.fillText('Off', InfinityRun.width-200, 400);
1294        }

1296        //-----------------------------------------------------------------------------

            /*
1298         * Highscore Tab
             *
1300         */

1302        } else if (GameState == State.Menu && curMenuTab ↩
               == MenuTab.Highscore) {


1304

            this.title = "Highscore";
1306        items = highScore;


1308

            callback = function(volume) { //if (numItem == 0) ↩
                GameState=State.Started
1310

            switch (volume) {

1312

            }

1314



1316

            };
1318        this.height = InfinityRun.height;
            this.width = InfinityRun.width;
1320

            var lingrad = this.createLinearGradient(0,0,0, ↩
               this.height);
1322        lingrad.addColorStop(0, '#000');
            lingrad.addColorStop(1, '#023');
1324        this.fillStyle = lingrad;
            this.fillRect(0,0,this.width, this.height, items[ ↩
               i]);
1326

            this.textAlign = "center";
1328        this.fillStyle = "White";

1330        var width = 10;
            var height = 100;
1332

1334        if (this.title) {
```

```
                this.font = Math.floor(this.size*1.3).toString() + "px Bungee";
1336            this.fillText(this.title, this.width/2, 150);
                height+= height;
1338        }

1340        var rank = 1;

1342        for (var i = 0; i < items.length; ++i)
            {
1344
                var size = Math.floor(this.size*0.8);
1346            if (i == selectedItem)
                {
1348                    this.fillStyle = "#A9F5F2";
                        size = this.size+5;
1350            }
                this.font = 0.6*size.toString() + "px Bungee";
1352            height += 50;
                this.fillText(rank + ". " + items[i], InfinityRun.width/2, height);
1354            this.fillStyle = "White";

1356            rank++;
            }
1358
            }
1360    // Credits Menu ----------------------------------------------------------

        else if (GameState == State.Menu && curMenuTab == MenuTab.Credits) {
1362
        this.title = "Credits";
1364    items = highScore;

1366    callback = function(volume) { //if (numItem == 0) GameState=State.Started

        switch (volume) {
1368
        }
1370

1372


1374    };
```

```
        this.height = InfinityRun.height;
1376    this.width = InfinityRun.width;

1378    var lingrad = this.createLinearGradient(0,0,0,
           this.height);
        lingrad.addColorStop(0, '#000');
1380    lingrad.addColorStop(1, '#023');
        this.fillStyle = lingrad;
1382    this.fillRect(0,0,this.width, this.height, items[
           i]);

1384    this.textAlign = "center";
        this.fillStyle = "White";
1386
        var width = 10;
1388    var height = 150;


1390
        if (this.title) {
1392            this.font = Math.floor(this.size*1.3).
                   toString() + "px Bungee";
                this.fillText(this.title, this.width/2,
                   150);
1394            height+= height;
        }
1396    var distanceText = 50
        this.font = Math.floor(50).toString() + "px
           Bungee";
1398    this.textAlign = "left";
        //Names
1400    this.fillText("Group Members:", this.width/5,
           300);
        this.font = Math.floor(40).toString() + "px
           Bungee";
1402    this.fillText("  Florian Durli", this.width/5,
           300+distanceText);
        this.fillText("  Koray Emtekin", this.width/5,
           300+2*distanceText);
1404    this.fillText("  Jannik Ivosevic", this.width/5,
           300+3*distanceText);
        this.fillText("  Marco Mayer", this.width/5,
           300+4*distanceText);
1406    this.fillText("  Johannes But", this.width/5,
           300+5*distanceText);
        //bottom info
1408    this.font = Math.floor(15).toString() + "px Times
           New Roman";
        this.textAlign = "center";
```

```
1410            distanceText = 20;
                this.fillText("InfinityRun␣is␣a␣nonprofit␣students␣project␣at␣\"Hochschule Furtwangen\"/\"Furtwangen University.\"␣Special␣thanks␣to␣\"Soulwire\"␣for␣his␣Sketch.js␣Minimal␣JavaScript␣Creative␣Coding␣Framework",this.width/2, this.height-2.2*distanceText);
1412            this.fillText("Sounds:␣freesounds.org␣Special␣thanks␣to␣Jack␣Rugile␣for␣his␣Parrallax␣Skyline",this.width/2, this.height-distanceText-5);
                this.fillText("2016",this.width/2, this.height-8);

1414

1416            }

1418        //Debug
            if (debug) {
1420            this.font = '16pt Arial';
                this.fillStyle = '#181818';
1422            this.fillText('Record: ' + s + "␣" + sc/*this.jumpCountRecord*/, this.width - 150, 33);
                this.fillStyle = this.scoreColor;
1424            this.fillText('Jumps: ' + this.jumpCount, this.width - 150, 50);
            }
1426
    };
1428
    InfinityRun.resize = function() {
1430        /* todo Windowscale optimization
             *
1432         *
             */
1434    }
```

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{
14   font-family: 'Bungee', cursive;
     background: #cecece;
16   border: 1px solid #181818;
  }
18
  canvas.sunsetFilter {
20   -webkit-animation: sunset-animation 70s;
  }
22
  canvas.invertFilter {
24   -webkit-animation: invert-animation 20s;
  }
26


28
  @-webkit-keyframes sunset-animation {
30   0% {
       -webkit-filter: sepia(0) saturate(2);
32   }

34   50% {
       -webkit-filter: sepia(1) saturate(15);
36   }

38   100% {
       -webkit-filter: sepia(0) saturate(2);
40   }
  }
42


44
  @-webkit-keyframes invert-animation {
46   0% {
       -webkit-filter: invert(0);
48   }

50   50% {
       -webkit-filter: invert(.8);
52   }

54   100% {
       -webkit-filter: invert(.0);
56   }
  }
```

A.4. index.html

```
<!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">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=utf-8">
        <script type="text/javascript" src="js/sketch.min
            .js" charset="utf-8"></script>
        <script type="text/javascript" src="js/soundjs
            -0.6.2.min.js" charset="utf-8"></script>
        <link href="https://fonts.googleapis.com/css?
            family=Bungee" rel="stylesheet">

    <title>Infinity Run</title>

    <link href="css/game.css" rel="stylesheet" type="text
        /css">
        <link rel="shortcut icon" type="image/x-icon"
            href="image/favicon.png">
</head>
<body onload="loadSound();">

<script>
// internal load sound files (called in onload (body))
function loadSound() {
    if (!createjs.Sound.initializeDefaultPlugins()) {
        return;}

    var audioPath = "sounds/";
    var sounds = [
        {id:"MainMenu", src:"menu.wav"},
                {id:"Crash", src:"crash.wav"},
                //{id:"", src:"error.wav"},     ::unused
        {id:"Jump", src:"jump.wav"},
                {id:"LevelUp", src:"levelup.wav"},
        {id:"Main1", src:"main1.wav"},
                {id:"Main2", src:"main2.wav"},
        {id:"Main3", src:"main3.wav"},
                {id:"Main4", src:"main4.wav"}

    ];

    createjs.Sound.registerSounds(sounds, audioPath);
}
</script>
<!-- Game div -->
<div id="container">
```

```
41 </div>
  <audio id="backgroundmusic" ></audio>
43 <audio id="fxaudio" ></audio>
  <script type="text/javascript" src="js/game.js" charset="
     utf-8"></script>
45 </body>
  </html>
```