Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent	:	**Gabriela Mai**
Vorgelegt am	:	20. Januar 2017
Vorgelegt von	:	Gruppe 4

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

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Team

Abbildung 1.: Florian Durli

Abbildung 2.: Jannik Ivosevic

Abbildung 3.: Johannes But

Abbildung 4.: Marco Mayer

Abbildung 5.: Koray Ektekin

1.2. Rollenverteilung

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

Tabelle 1.: Rollenverteilung

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

1.3. Spielidee

1.3.1. Spielkonzept

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

1.3.2. Entwurfsskizze

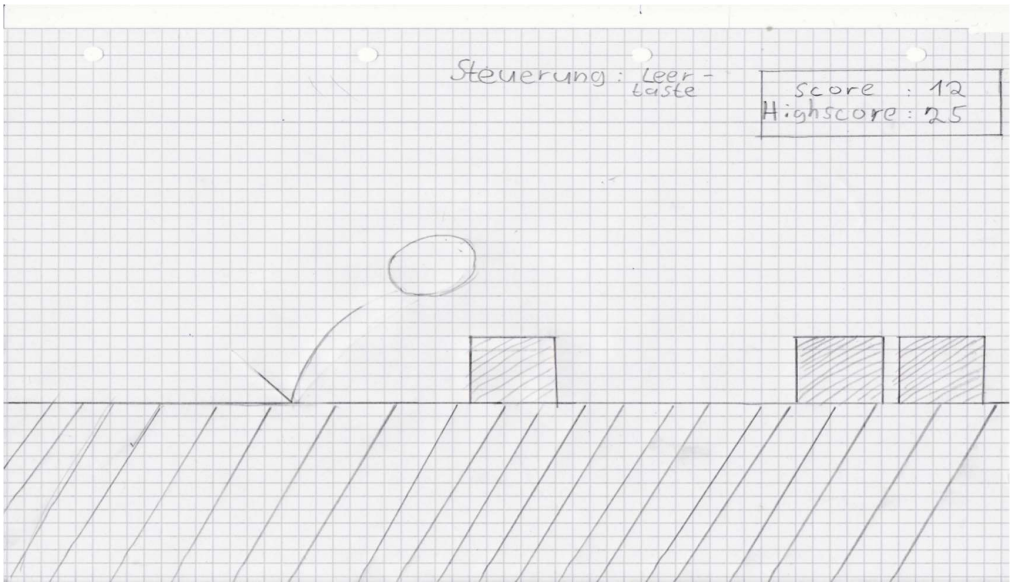

Abbildung 6.: Entwurfsskizze

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

1.3.3. Erforderliche Software

1.3.3.1. Notepad++

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

1.3.3.2. Chrome

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

1.3.3.3. Gimp

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

1.3.3.4. Git/Github

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

2. Phasen

2.1. Entwurf und Anforderungen

2.1.1. Funktionale Anforderungen

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

2.1.2. Nicht funktionale Anforderungen

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

2.1.3. Projektplan

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

Tabelle 2.: Phase 1: Entwurf und Anforderungen

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

Tabelle 3.: Phase 2: Implementierung

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

Tabelle 4.: Phase 3: Test

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

Tabelle 5.: Phase 4: Dokumentation und Präsentation

2.1.4. Releaseplan

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

Tabelle 6.: Releaseplan

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

2.2. Implementation - Zwischenstand

2.2.1. Erfüllte Anforderungen

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

2.2.2. Nicht erfüllte Anforderungen

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

2.2.3. Das Spiel

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

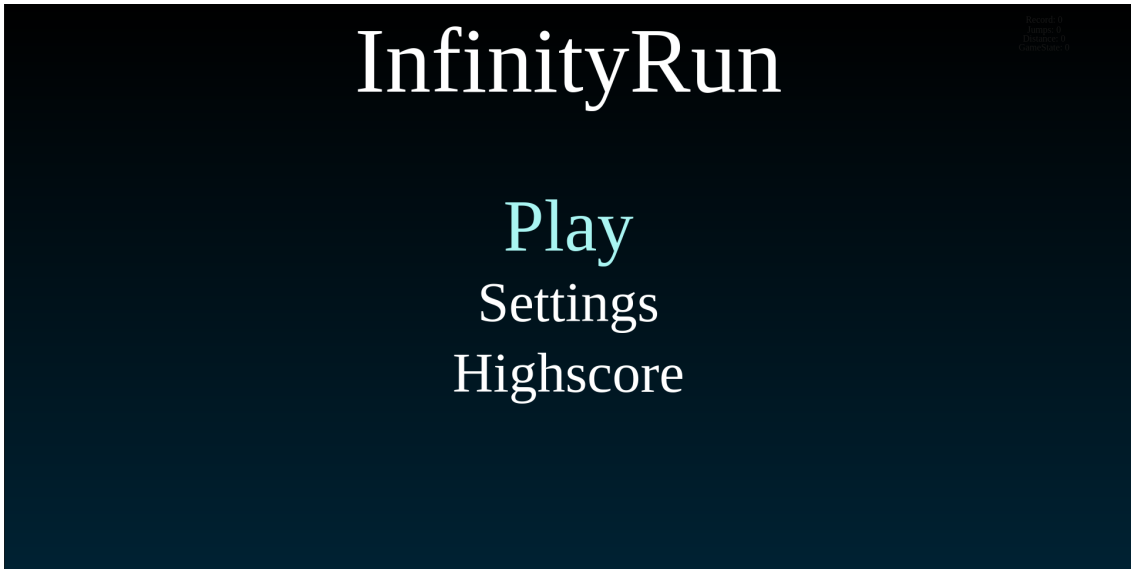

Abbildung 7.: Startbildschirm

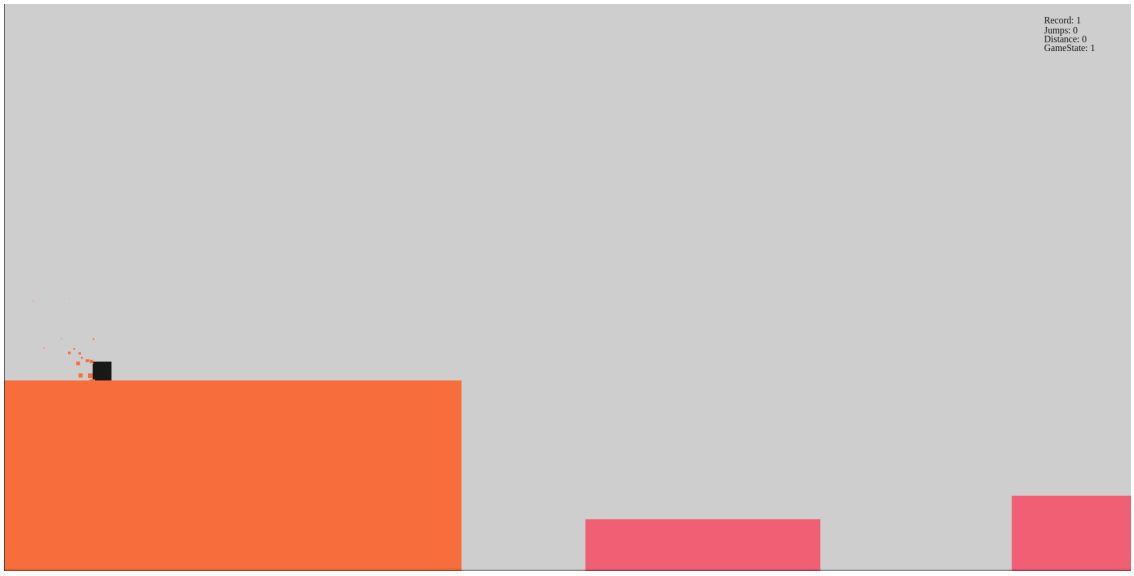

Abbildung 8.: Das Spiel

2.2.4. Bibliothek

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

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

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

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

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

Quelle: [sou]

2.2.5. Code

2.2.5.1. Framework initialisieren

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

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

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

2.2.5.2. Spieler initialisieren

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

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

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

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

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

2.2.5.3. Erstellen der Spielebene

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

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

```
22          height: 70
})

this.third = new Platform
({
          x: (this.second.x + this.second.width) + random(
              this.maxDistanceBetween - 150, this.
              maxDistanceBetween),
          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);
          this.second.color = randomChoice(this.colors);
          this.third.color = randomChoice(this.colors);
          this.colliding = false;
          this.platforms = [this.first, this.second, this.
              third];
}
```

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, 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()
{
          this.first.x -= 3 + InfinityRun.acceleration;
          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 Spielkonzeptänderungen haben wir im Laufe der Implementation vorgenommen:

- Die Spielebene hat anstatt Hindernisse zufallsgenerierte variable Plattformen.
- Spiel-Menü eingefügt
- Spielhintergrund
- Spielfigur ändert sich nicht nach Aufprall

2.3.2. Funktionsdiagramm

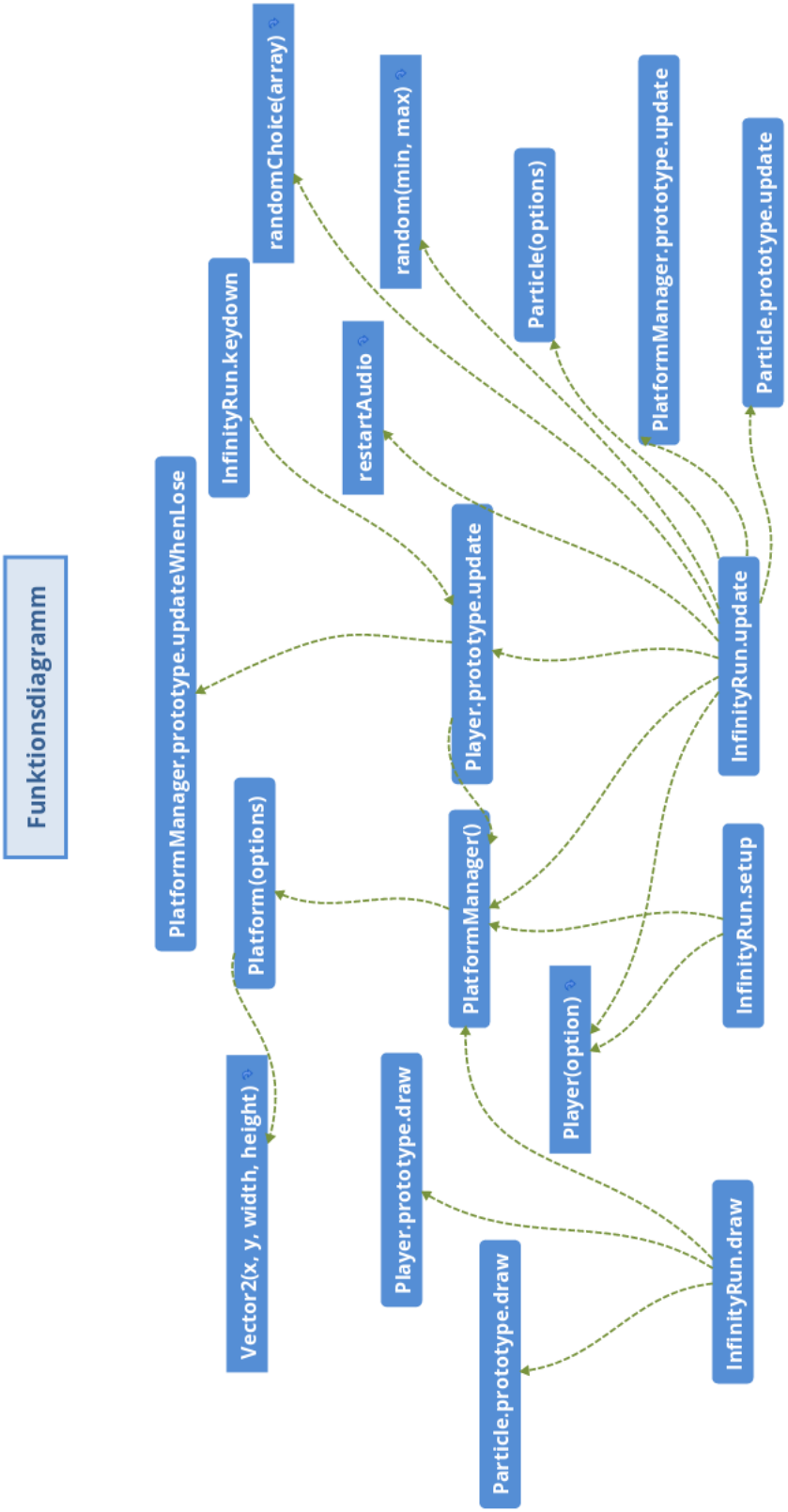

Abbildung 9.: Funktionsdiagramm

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.

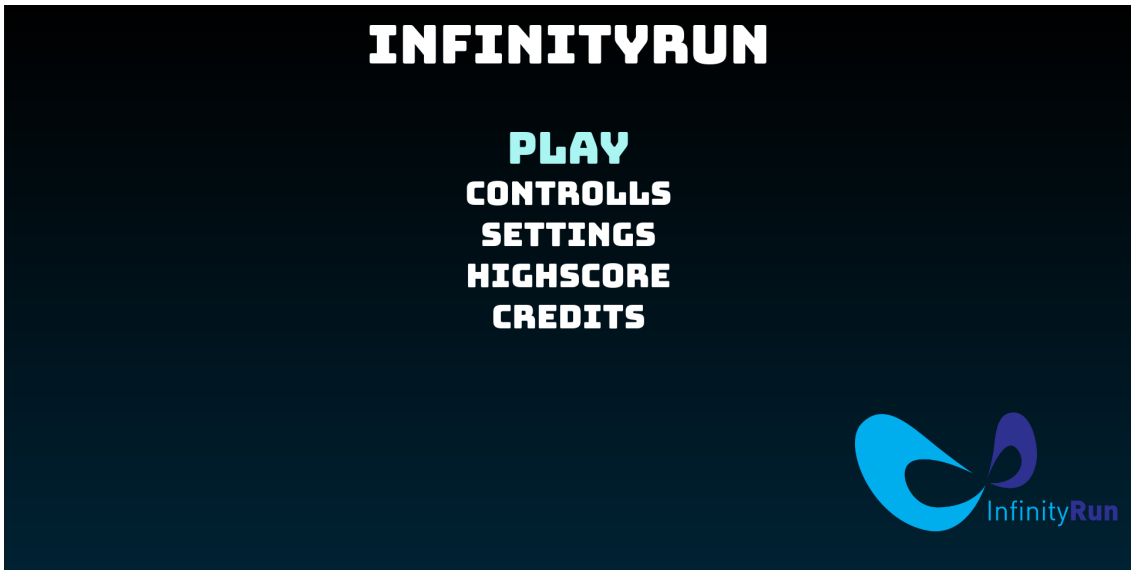

Abbildung 11.: Startbildschirm - Endstand

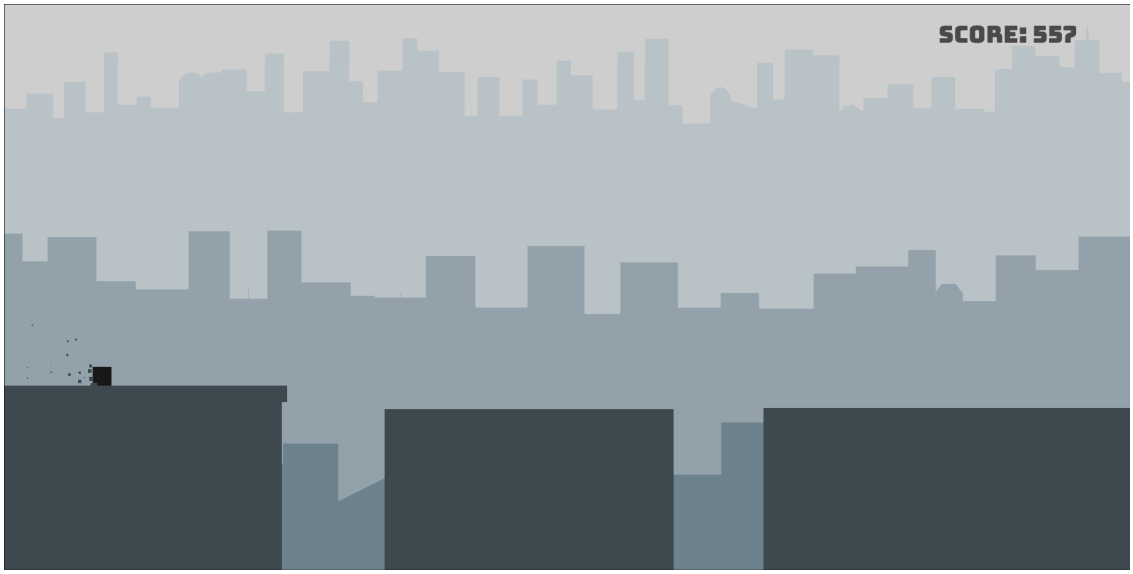

Abbildung 12.: Das Spiel - Endstand

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 haben wir das Spiel nun auf einem freien Webserver(Quelle:[Sqa]) 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 Spielspaß auch unterwegs zu genießen.
Spiel: `http://infinityrun.square7.ch/`

Abbildung 13.: QR-Code zum Webspace

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. Unseren 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. Bug Beschreibung

2.4.5.1. Performance - Priorität 1

Je nach Auflösung und Leistung des Rechners, kann es zu Performance Einbrüchen kommen.

2.4.5.2. Hintergrundmusik - Priorität 2

Das Abspielen der Musik funktioniert in manchen Fällen, aber in anderen nicht, dies scheint an dem HTML-Audioelement zu liegen.

2.4.5.3. Effektlautstärke - Priorität 3

Die Effektlautstärke ist deutlich zu laut und sollte angepasst werden.

2.4.5.4. Menüsound - Priorität 3

Beim Zurückkehren von dem Spiel ins Menü wird die Menümusik nicht mehr abgespielt. Auch dieser Bug kann ein Fehler der Implementation des HTML-Audioelements sein.

2.4.6. 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.

2.4.7. Bugbehebung

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

2.4.7.1. 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. Ebenfalls wurde an gleicher Stelle eine Option zum Abstellen der Filter implementiert.

2.4.7.2. 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.

2.4.7.3. Effektlautstärke - Priorität 3

Die Effektlautstärke wurde von uns nun perfekt zum Spiel angepasst.

2.4.7.4. 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, einen umgekehrten Spielverlauf und einen globalen Score, der bei Neustart verfügbar ist, besitzen. 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.

4. Fazit

Der Informatik Workshop hat uns gezeigt, dass Teamwork eine der wichtigsten Eigenschaften einer solchen Projektarbeit ist. Wir haben gelernt gemeinsam an einem Strang zu ziehen und hatten keinerlei Probleme unseren Projektplan einzuhalten. Trotz Problemen bei der Umsetzung des Spiel konnten wir wie geplant alle Features implementieren. Das Verwenden 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

[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

[Sqa] SQARE 7 HOST SERVICE PROVIDER: *Kostenloses Webhosting* http://www.square7.ch/

[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 20. Januar 2017 Florian Durli

FURTWANGEN, den 20. Januar 2017 Jannik Ivosevic

FURTWANGEN, den 20. Januar 2017 Johannes But

FURTWANGEN, den 20. Januar 2017 Marco Mayer

FURTWANGEN, den 20. Januar 2017 Koray Emtekin

A. Anhang

A.1. 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/
 */
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;
        this.singleroofTopHeight = this.width / random(2, 4);
        this.singleroofTopDirection = round(random(0, 1)) === 0;
                this.normalTop = !this.singleroofTop &&
```

```
                floor(random(0, 10)) === 0;
        this.normalTopHeight = this.width / random(2, 4);
        this.normalTopchimney = round(random(0, 1)) === 0;
                this.coneTop = !this.singleroofTop && !this.normalTop && floor(random(0, 10)) === 0;
        this.coneTopHeight = this.width / random(3, 4);
                this.coneTopWidth = this.width / random(1, 2);
        this.coneTopeflat = round(random(0, 1)) === 0;
                this.companyTop = !this.singleroofTop && !this.summitTop && !this.radioTop && !this.normalTop && floor(random(0, 10)) === 0;
        this.companyTopHeight = this.width / random(4, 6);
        this.companyTopcount = 4;//round(random(3, 6));
        this.radioTop = !this.summitTop && floor(random(0, 10)) === 0;
        this.radioTopWidth = this.layer / 2;
        return this.radioTopHeight = random(6, 30);
};

Tower.prototype.render = function()
{
        InfinityRun.fillStyle = InfinityRun.strokeStyle = this.color;
        InfinityRun.lineWidth = 2;
        InfinityRun.beginPath();
        InfinityRun.rect(this.x, this.y, this.width, this.height);
        InfinityRun.fill();
        InfinityRun.stroke();

if (vgaquality > 0){ //graphics higher then low
        if (this.singleroofTop)
        {
                InfinityRun.beginPath();
                InfinityRun.moveTo(this.x, this.y);
                InfinityRun.lineTo(this.x + this.width, this.y);
                if (this.singleroofTopDirection)
                {
                        InfinityRun.lineTo(this.x + this.width, this.y - this.singleroofTopHeight);
                }
```

```
155                     else
                        {
157                             InfinityRun.lineTo(this.x, this.y
                                    - this.singleroofTopHeight);
                        }
159                     InfinityRun.closePath();
                        InfinityRun.fill();
161                     InfinityRun.stroke();
                }
163
                if (this.normalTop)
165             {
                        InfinityRun.beginPath();
167                     InfinityRun.moveTo(this.x, this.y);
                        InfinityRun.lineTo(this.x + this.width,
                           this.y);
169                             InfinityRun.lineTo(this.x + (this
                                   .width/2), this.y-this.
                                   normalTopHeight);
                        InfinityRun.closePath();
171                     InfinityRun.fill();
                        InfinityRun.stroke();
173                             if(this.normalTopchimney)
                                {
175                                     InfinityRun.beginPath();
                                        InfinityRun.moveTo(this.x
                                           +(this.width/5), this.
                                           y);
177                                     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));
179                                     InfinityRun.lineTo(this.x
                                            + (this.width/5)+(
                                           this.width/10), this.y
                                           );
                                        InfinityRun.closePath();
181                                     InfinityRun.fill();
                                        InfinityRun.stroke();
183                             }
                }
185             if (vgaquality > 1 && this.coneTop)
                {
```

```
187                 InfinityRun.beginPath();
                    InfinityRun.moveTo(this.x, this.y);
189                 InfinityRun.lineTo(this.x + (this.width-
                       this.coneTopWidth)/2, this.y-this.
                       coneTopHeight);
                            if(!this.coneTopeflat)
191                         {
                                    InfinityRun.lineTo(this.x
                                       +(this.width/2), this.
                                       y-(this.coneTopHeight
                                       *1.3));
193                         }
                            InfinityRun.lineTo(this.x + ((
                               this.width-this.coneTopWidth)
                               /2)+this.coneTopWidth, this.y-
                               this.coneTopHeight);
195                         InfinityRun.lineTo(this.x + this.
                               width, this.y);
                    InfinityRun.closePath();
197                 InfinityRun.fill();
                    InfinityRun.stroke();
199
            }
201         if (vgaquality > 1 && this.companyTop)
            {
203                 var ctc = 1;
                    while (ctc<=this.companyTopcount)
205                 {
                            InfinityRun.beginPath();
207                         InfinityRun.moveTo(this.x , this.
                               y);
                            InfinityRun.lineTo(this.x + ctc*(
                               this.width/this.
                               companyTopcount), this.y-this.
                               companyTopHeight);
209                         InfinityRun.lineTo(this.x + ctc*(
                               this.width/this.
                               companyTopcount), this.y+this.
                               companyTopHeight);
                            InfinityRun.closePath();
211                         InfinityRun.fill();
                            InfinityRun.stroke();
213                         ctc++;
                    }
215         }
            if (vgaquality > 1 && this.summitTop)
217         {
                    InfinityRun.beginPath();
```

```
219                     InfinityRun.moveTo(this.x + (this.width /
                            2), this.y - this.summitTopHeight);
                        InfinityRun.lineTo(this.x + (this.width /
                            2) + this.summitTopWidth, this.y);
221                     InfinityRun.lineTo(this.x + (this.width /
                            2) - this.summitTopWidth, this.y);
                        InfinityRun.closePath();
223                     InfinityRun.fill();
                        InfinityRun.stroke();
225             }

227             if (vgaquality > 1 && this.radioTop)
                {
229                     InfinityRun.beginPath();
                        InfinityRun.moveTo(this.x + (this.width /
                            2), this.y - this.radioTopHeight);
231                     InfinityRun.lineTo(this.x + (this.width /
                            2), this.y);
                        InfinityRun.lineWidth = this.
                           radioTopWidth;
233                     return InfinityRun.stroke();
                }
235 }
    };
237
    Street = function(config)
239 {
                this.x = 0;
241             this.alltowers = [];
                this.layer = config.layer;
243             this.width = {
                min: config.width.min,
245             max: config.width.max
    };
247
    this.height = {
249             min: config.height.min,
                max: config.height.max
251 };

253         this.speed = config.speed;
            this.color = config.color;
255         this.populate();
            return this;
257 };

259 Street.prototype.populate = function()
    {
```

```
261         var newHeight, newWidth, results, totalWidth;
            totalWidth = 0;
263         results = [];
            while (totalWidth <= InfinityRun.width + (this.
               width.max * 2))
265         {
                    newWidth = round(random(this.width.min,
                       this.width.max));
267                 newHeight = round(random(this.height.min,
                        this.height.max));
                    this.alltowers.push(new Tower({
269                 layer: this.layer,
                    x: this.alltowers.length === 0 ? 0 : this
                       .alltowers[this.alltowers.length - 1].
                       x + this.alltowers[this.alltowers.
                       length - 1].width,
271                 y: InfinityRun.height - newHeight,
                    width: newWidth,
273                 height: newHeight,
                    color: this.color
275                 }));
                    results.push(totalWidth += newWidth);
277         }
            return results;
279 };

281 Street.prototype.update = function()
    {
283         var firstTower, lastTower, newHeight, newWidth;
            if (InfinityRun.accelerationTweening==0)
285         {
                    this.x-=((150) * this.speed) * dt;
287         }
            else
289         {
        this.x -= ((InfinityRun.accelerationTweening*330) *
           this.speed) * dt;
291         }

293         firstTower = this.alltowers[0];
            if (firstTower.width + firstTower.x + this.x < 0)
295         {
                    newWidth = round(random(this.width.min,
                       this.width.max));
297                 newHeight = round(random(this.height.min,
                        this.height.max));
                    lastTower = this.alltowers[this.alltowers
                       .length - 1];
```

```
299                 firstTower.reset({
                            layer: this.layer,
301                         x: lastTower.x + lastTower.width,
                            y: InfinityRun.height - newHeight
                               ,
303                         width: newWidth,
                            height: newHeight,
305                         color: this.color
                    });
307         return this.alltowers.push(this.alltowers.shift()
               );
            }
309 };

311 Street.prototype.render = function()
    {
313         var i;
            i = this.alltowers.length;
315         InfinityRun.save();
                    InfinityRun.translate(this.x, (
                       InfinityRun.height - (InfinityRun.
                       height-(-jumpheight*0.5)-400)) / 20 *
                       this.layer);
317
                    while (i--) {
319                 this.alltowers[i].render(i);
            }
321         return InfinityRun.restore();
    };
323
    //------- Vector [Get/Set] Functions ----------
325 //Set X,Y,Width,Height
    function Vector2(x, y, width, height) {
327     this.x = x;
        this.y = y;
329     this.width = width;
        this.height = height;
331     this.previousX = 0;
        this.previousY = 0;
333 };


335
    // Set X,Y
337 Vector2.prototype.setPosition = function(x, y) {

339     this.previousX = this.x;
        this.previousY = this.y;
341
```

```
        this.x = x;
343     this.y = y;

345 };
    // Set X
347 Vector2.prototype.setX = function(x) {

349     this.previousX = this.x;
        this.x = x;
351
    };
353
    // Set Y
355 Vector2.prototype.setY = function(y) {

357     this.previousY = this.y;
        this.y = y;
359
    };
361
    // Collision / Intersection Top
363 Vector2.prototype.intersects = function(obj) {

365     if (obj.x < this.x + this.width && obj.y < this.y + ↩
           this.height &&
            obj.x + obj.width > this.x && obj.y + obj.height ↩
              > this.y) {
367         return true;
        }
369
        return false;
371 };

373 // Collision / Intersection Left
    Vector2.prototype.intersectsLeft = function(obj) {
375
        if (obj.x < this.x + this.width && obj.y < this.y + ↩
           this.height) {
377         return true;
        }
379
        return false;
381 };

383 //---------- Player ----------
    function Player(options) {
385
        this.setPosition(options.x, options.y);
```

```
387     this.width = options.width;
        this.height = options.height;
389     this.velocityX = 0;
        this.velocityY = 0;
391     this.jumpSize = -13;
        this.color = '#181818';
393
}
395
Player.prototype = new Vector2;
397
Player.prototype.update = function() {
399     // Gravity
        this.velocityY += 1;
401         //um bg zu ändern
            jumpheight=(this.y);
403     this.setPosition(this.x + this.velocityX, this.y + ↩
           this.velocityY);

405     if (this.y > InfinityRun.height || this.x + this.↩
           width < 0) {
            this.x = 150;
407         this.y = 50;
            this.velocityX = 0;
409         this.velocityY = 0;
            InfinityRun.jumpCount = 0;
411         InfinityRun.acceleration = 0;
            InfinityRun.accelerationTweening = 0;
413         InfinityRun.scoreColor = '#181818';
            InfinityRun.platformManager.maxDistanceBetween = ↩
               350;
415
                    //InfinityRun.pause();
417
                    //highscore update
419                 if (timePassed>highScore[0]){
                            var help =highScore[0];
421                         var help2=highScore[1];
                            highScore[0]=timePassed;
423                         for(i=1; i<=9;i++){
                            help2 = highScore[i];
425                         highScore[i]=help;
                            help=help2;
427
                            }
429
                    }
431         InfinityRun.platformManager.updateWhenLose();
```

```
433                     playSFX("Crash");
                        bgFX.stop();
435                     difficulty = 0;
                        ms = 0;
437         }

439         if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
                InfinityRun.keys.W || InfinityRun.dragging) &&
               this.velocityY < -8) {
                this.velocityY += -0.75;
441
        }
443

445         if (InfinityRun.keys.DOWN) {
                this.velocityY += 1;
447         }

449 };

451 Player.prototype.draw = function() {
        InfinityRun.fillStyle = this.color;
453     InfinityRun.fillRect(this.x, this.y, this.width, this
           .height);
    };
455
    // ---------- Platforms ----------
457
    function Platform(options) {
459     this.x = options.x;
        this.y = options.y;
461     this.width = options.width;
        this.height = options.height;
463     this.previousX = 0;
        this.previousY = 0;
465     this.color = options.color;
    }
467
    Platform.prototype = new Vector2;
469
    Platform.prototype.draw = function() {
471     InfinityRun.fillStyle = this.color;
        InfinityRun.fillRect(this.x, this.y, this.width, this
           .height);
473 };

475 // ---------- 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);
    this.second.color = randomChoice(this.colors);
    this.third.color = randomChoice(this.colors);

    this.colliding = false;

    this.platforms = [this.first, this.second, this.third];
}

PlatformManager.prototype.update = function() {

```

```
        this.first.x -= 3 + InfinityRun.acceleration;
515     if (this.first.x + this.first.width < 0) {
            this.first.width = random(450, 800);
517         this.first.x = (this.third.x + this.third.width)
               + random(this.maxDistanceBetween - 150, this.
               maxDistanceBetween);
            //this.first.y = random(this.third.y - 32,
               InfinityRun.height - 80);
519                 this.first.y = random(this.third.y - 32,
                       InfinityRun.height - 200);
            this.first.height = this.first.y + InfinityRun.
               height + 10;
521         this.first.color = randomChoice(this.colors);
        }
523
        this.second.x -= 3 + InfinityRun.acceleration;
525     if (this.second.x + this.second.width < 0) {
            this.second.width = random(450, 800);
527         this.second.x = (this.first.x + this.first.width)
                + random(this.maxDistanceBetween - 150, this.
               maxDistanceBetween);
                    //this.first.y = random(this.third.y -
                       32, InfinityRun.height - 80);
529         this.second.y = random(this.first.y - 32,
               InfinityRun.height - 200);
            this.second.height = this.second.y + InfinityRun.
               height + 10;
531         this.second.color = randomChoice(this.colors);
        }
533
        this.third.x -= 3 + InfinityRun.acceleration;
535     if (this.third.x + this.third.width < 0) {
            this.third.width = random(450, 800);
537         this.third.x = (this.second.x + this.second.width
               ) + random(this.maxDistanceBetween - 150, this
               .maxDistanceBetween);
                    //this.first.y = random(this.third.y -
                       32, InfinityRun.height - 80);
539         this.third.y = random(this.second.y - 32,
               InfinityRun.height - 200);
            this.third.height = this.third.y + InfinityRun.
               height + 10;
541         this.third.color = randomChoice(this.colors);
        }
543
    };
545
```

```
547
// reset
549 PlatformManager.prototype.updateWhenLose = function() {

551     this.first.x = 300;
    this.first.color = randomChoice(this.colors);
553         this.first.y = 700;
    //this.first.y = InfinityRun.width / random(2, 3);
555         this.second.y = 650;
        this.third.y = 600;
557     this.second.x = (this.first.x + this.first.width) +
        random(this.maxDistanceBetween - 150, this.
        maxDistanceBetween);
    this.third.x = (this.second.x + this.second.width) +
        random(this.maxDistanceBetween - 150, this.
        maxDistanceBetween);

559
};
561
// ---------- Particle System ---------- (Sketch Docs)
563 function Particle(options) {
    this.x = options.x;
565     this.y = options.y;
    this.size = 10;
567     this.velocityX = options.velocityX || random(-(
        InfinityRun.acceleration * 3) + -8, -(InfinityRun.
        acceleration * 3));
    this.velocityY = options.velocityY || random(-(
        InfinityRun.acceleration * 3) + -8, -(InfinityRun.
        acceleration * 3));
569     this.color = options.color;
}
571
Particle.prototype.update = function() {
573     this.x += this.velocityX;
    this.y += this.velocityY;
575     this.size *= 0.89;
};
577
Particle.prototype.draw = function() {
579     InfinityRun.fillStyle = this.color;
    InfinityRun.fillRect(this.x, this.y, this.size, this.
        size);
581 };

583 /*********************************************/
InfinityRun.setup = function() {
585
```

```
    this.jumpCount = 0;
587 this.acceleration = 0;
    this.accelerationTweening = 0;
589 this.player = new Player({
        x: 150,
591     y: 30,
        width: 32,
593     height: 32
    });
595
        setTimeout(function (){
597     playMenuFX("MainMenu");

599     }, 200);


601
    this.platformManager = new PlatformManager();
603
    this.particles = [];
605 this.particlesIndex = 0;
    this.particlesMax = 20;
607 this.collidedPlatform = null;
    this.scoreColor = '#181818';
609 this.jumpCountRecord = 0;
        //-----------------------------------------
611 var i, results;
    i = 3;
613 results = [];
    while (i--) {
615   results.push(Town.push(new Street({
        layer: i + 1,
617     width: {
          min: (i + 1) * 20,
619       max: (i + 1) * 50
        },
621     height: {
          min: InfinityRun.height-200 - (i * round(↩
             InfinityRun.height/3)),
623       max: InfinityRun.height-50 - (i * round(↩
             InfinityRun.height/3))
        },
625     speed: (i + 1) * .003,
        color: 'hsl(␣200,␣' + (((i + 1) * 1) + 10) + '%,␣↩
           ' + (75 - (i * 13)) + '%␣)'
627   })));
    }
629     return results;
        //-----------------------------------------
```

```
631
633 };
635 InfinityRun.clear = function() {
        return InfinityRun.clearRect(0, 0, InfinityRun.width,
            InfinityRun.height);
637   };
639 Array.max = function( array ){
        return Math.max.apply( Math, array );
641 };
643 var sc = 0;
    var sx = 0;
645 var sy = 0;
    var sz = 0;
647 var invertRunning = false;
    var sunsetRunning = false;
649 timer = setInterval(function() {
      if (!playTimer) return;
651     ms += 1;
            sc += 1;
653         sy += 1;
            sz += 1;
655         if (sc == 99) {
                    s+=1;
657                 sx+=1;
                    sc = 0;
659         }

661         updateTimer();

663 }, 1);
665 function randomIntFromInterval(min,max)
    {
667         var milliseconds = new Date().getMilliseconds();
        return Math.floor(Math.random()*(max-min+1)+min);
669 }

671
    var rng = random(115,124);
673 var rng2 = random(13,16)
    function updateTimer() {
675   if (s==rng) {
              if(!invertRunning) {
677           invertRunning = true;
```

```
            rng = random(30,50);
679         if(setFilters) //toggle filters
            {
681             qs.classList.toggle('invertFilter');
            }
683         }
            s=0;
685 }
    if (sx==rng2) {
687         if(!sunsetRunning) {
            sunsetRunning = true;
689         rng2 = random(2,5);
            if(setFilters) //toggle filters
691         {
                qs.classList.toggle('sunsetFilter');
693         }
            }
695      sx=0;
    }
697 if (sz==70) {
            invertRunning = false;
699         sz = 0;
    }
701 if (sy==70) {
            invertRunning = false;
703         sy = 0;
    }
705


707       timePassed = ms;
  }
709
  function toggleTimer() {
711   if (!playTimer) {
      //s = 0, ms = 0;
713
      updateTimer();
715   }
    playTimer = !playTimer;
717 }

719 var difficulty = 0;

721 InfinityRun.update = function() {
          if (GameState == State.Started) {
723       //clear func bg
          var i, results;
725     dt = InfinityRun.dt < .1 ? .1 : InfinityRun.dt / 16;
```

```
    dt = dt > 5 ? 5 : dt;
727 i = Town.length;
    results = [];
729 while (i--) {
      results.push(Town[i].update(i));
731 }

733     if(document.hasFocus()) {
                toggleTimer();
735     } else {
                toggleTimer();
737     }

739
    this.player.update();
741     if(difficulty ==0) {

743             playbgFX("Main1");
                difficulty = 1;
745     } else if (timePassed>1000 && timePassed < 5000 ↩
          && difficulty == 1) {
                this.accelerationTweening = 1.5;
747     this.platformManager.maxDistanceBetween = 430;

749             bgFX.stop();
                playbgFX("Main2");
751             playSFX("LevelUP");

753             difficulty = 2;
        } else if (timePassed>5000 && timePassed < 10000 ↩
          && difficulty == 2) {
755             this.accelerationTweening = 2.7;
        this.platformManager.maxDistanceBetween = 530;
757
                bgFX.stop();
759             playbgFX("Main3");
                playSFX("LevelUP");
761
                difficulty = 3;
763     } else if (timePassed>10000 && timePassed < 15000 ↩
           && difficulty == 3) {
                this.accelerationTweening = 3.8;
765     this.platformManager.maxDistanceBetween = 580;

767             bgFX.stop();
                playbgFX("Main4");
769             playSFX("LevelUP");
```

```
771                 difficulty = 4;
            } else if (timePassed>15000 && timePassed < 20000
                && difficulty == 4) {
773                 this.accelerationTweening = 4.4;
                    this.PlatformManager.maxDistanceBetween =
                        610;
775
                    playSFX("LevelUP");
777
                    difficulty = 5;
779         } else if (timePassed>20000 && difficulty == 5) {
                    this.accelerationTweening = 5;
781                 this.PlatformManager.maxDistanceBetween =
                        620;


783
                    playSFX("LevelUP");
785
                    difficulty = 6;
787         }
        this.acceleration += (this.accelerationTweening -
            this.acceleration) * 0.01;
789
        for (i = 0; i < this.platformManager.platforms.length
            ; i++) {
791          if (this.player.intersects(this.platformManager.
                platforms[i])) {
              this.collidedPlatform = this.platformManager.
                  platforms[i];
793           if (this.player.y < this.platformManager.
                  platforms[i].y) {
                this.player.y = this.platformManager.
                    platforms[i].y;
795
                // Gravity after Collision with Platform
797             this.player.velocityY = 0;
              }
799
              this.player.x = this.player.previousX;
801           this.player.y = this.player.previousY;

803           this.particles[(this.particlesIndex++) % this
                  .particlesMax] = new Particle({
                x: this.player.x,
805             y: this.player.y + this.player.height,
                color: this.collidedPlatform.color
807           });
```

```
809             if (this.player.intersectsLeft(this.
                    platformManager.platforms[i])) {
                    this.player.x = this.collidedPlatform.x -
                        64;
811                 for (i = 0; i < 10; i++) {
                        // SpawnParticles @PlayerPostion with
                            intersecting Platform Color
813                     this.particles[(this.particlesIndex
                            ++) % this.particlesMax] = new
                            Particle({
                            x: this.player.x + this.player.
                                width,
815                         y: random(this.player.y, this.
                                player.y + this.player.height)
                                ,
                            velocityY: random(-30, 30),
817                         color: randomChoice(['#181818', '
                                #181818', this.
                                collidedPlatform.color])
                        });
819                 };

821                 // bounce player / push him away (effect)
                    this.player.velocityY = -10 + -(this.
                        acceleration * 4);
823                 this.player.velocityX = -20 + -(this.
                        acceleration * 4);

825                             if (timePassed > this.
                                    jumpCountRecord) {
                        this.jumpCountRecord = timePassed
                            ;
827                 }
                } else {
829
                    // ---------- Controller ----------
831                 // dragging: Mouse click & touch support
                    if (this.dragging || this.keys.SPACE ||
                        this.keys.UP || this.keys.W) {
833                     this.player.velocityY = this.player.
                            jumpSize;
                        this.jumpCount++;
835
                                                playSFX("Jump");
837                 }
                }
839         }
        };
```

```
841
        for (i = 0; i < this.platformManager.platforms.length
            ; i++) {
843         this.platformManager.update();
        };
845
        for (i = 0; i < this.particles.length; i++) {
847         this.particles[i].update();
        };
849
            //bg
851         return results;
    }
853
    };
855


857


859 var selectedItem = 0;
    var audioItem = 10;
861
    InfinityRun.keydown = function() {
863     if (InfinityRun.keys.ESCAPE && GameState==State.
            Started) {
                    InfinityRun.clear();
865                 GameState = State.Menu;

867                 bgFX.setPaused(true);
                    //playMenuFX("MainMenu");
869                 menuFX.setPaused(false);
                    toggleTimer();
871
                    freshStart = 1;
873
            } else if (InfinityRun.keys.ESCAPE && GameState==
                State.Menu && curMenuTab==MenuTab.Main &&
                freshStart == 1) {
875                 GameState = State.Started;
                    toggleTimer();
877                 //menuFX.stop();
                    bgFX.setPaused(false);
879                 menuFX.setPaused(true);

881         }else if (InfinityRun.keys.ESCAPE && GameState==
                State.Menu && curMenuTab==MenuTab.Controlls) {
                    curMenuTab = MenuTab.Main;
883     }else if (InfinityRun.keys.ESCAPE && GameState==State
```

```
.Menu && curMenuTab==MenuTab.Settings) {
            curMenuTab = MenuTab.Main;
885     } else if (InfinityRun.keys.ESCAPE && GameState==
        State.Menu && curMenuTab==MenuTab.Highscore) {
            curMenuTab = MenuTab.Main;
887     } else if (InfinityRun.keys.ESCAPE && GameState==
        State.Menu && curMenuTab==MenuTab.Credits) {
            curMenuTab = MenuTab.Main;
889     }

891     //main menu controls
        if (InfinityRun.keys.UP && GameState == State.
        Menu) {
893             selectedItem = (selectedItem + items.
                length - 1) % items.length;
        }
895     if (InfinityRun.keys.DOWN && GameState == State.
        Menu) {
            selectedItem = (selectedItem + 1) % items
                .length;
897     }
        //general settings choose
899     if (InfinityRun.keys.UP && curMenuTab==MenuTab.
        Settings && settingsItem!=0) {
            settingsItem-=1;
901     }
        if (InfinityRun.keys.DOWN && curMenuTab==MenuTab.
        Settings && settingsItem!=2) {
903             settingsItem+=1;
        }
905     // settings audio change
        if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.
        Settings && audioItem !=0 && settingsItem ==0)
         {
907             audioItem = (audioItem + items.length -
                1) % items.length;
            createjs.Sound.volume -= 0.1;
909     }

911     if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab
        .Settings && audioItem !=10 && settingsItem
        ==0) {
            audioItem = (audioItem + 1) % items.
                length;
913             createjs.Sound.volume += 0.1;
        }
915     //graphic settings change
        if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.
```

```
            Settings && vgaquality!=0 && settingsItem ==1)
             {
                vgaquality-=1;
        }

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

        if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab
            .Settings && setFilters && settingsItem ==2) {
                setFilters=false;
                if (invertRunning) {
                        qs.classList.toggle('invertFilter
                            ');
                }
                if (sunsetRunning) {
                        qs.classList.toggle('sunsetFilter
                            ');
                }
        }
        if(InfinityRun.keys.ENTER && GameState == State.
            Menu) {
                callback(selectedItem);
        }

}

Menu = function() {
}

InfinityRun.draw = function() {

        if(GameState == State.Started) {
        var i, results;
    i = Town.length;
    results = [];
        if(vgaquality>1){
    while (i--) {
             results.push(Town[i].render(i));

```

```
        }
957         }
        if (vgaquality==1)
959         {
        i-=1; //only upper towers will be drawn
961         while (i--) {
                results.push(Town[i+1].render(i+1));
963
        }
965         }
        if (vgaquality<1)
967         {
        i-=2; //only one street will be drawn
969         while (i--) {
                results.push(Town[i+1].render(i+1));
971
        }
973
            }
975
        this.player.draw();
977
        for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
979         this.platformManager.platforms[i].draw();
        };
981
        //Draw particles
983     for (i = 0; i < this.particles.length; i++) {
            this.particles[i].draw();
985     };

987         //draw score
            this.font = '30pt Bungee';
989         this.textAlign="left";
            this.fillStyle = '#494949';
991         this.fillText('Score: ', this.width - 330, 65);
            this.fillText(timePassed , this.width - 170, 65);
993
            /*
995          * Main Menu
             *
997          */
            } else if (GameState === State.Menu && curMenuTab
               ==MenuTab.Main) {
999
            this.title = "InfinityRun";
1001        items = ["Play","Controls", "Settings", "
```

```
    Highscore", "Credits"];

1003        callback = function(numItem) { //if (numItem ==
    0) GameState=State.Started

1005        switch (numItem) {
              case 0:
1007                GameState=State.Started;
                    toggleTimer();
1009                //bgFX.stop();
                    menuFX.setPaused(true);
1011                bgFX.setPaused(false);
                    break;
1013          case 1:
                curMenuTab=MenuTab.Controlls;
1015            break;
              case 2:
1017                curMenuTab=MenuTab.Settings;
                    break;
1019          case 3:
                curMenuTab=MenuTab.Highscore;
1021                break;
              case 4:
1023            curMenuTab=MenuTab.Credits;
                    break;
1025

            }
1027

            };
1029        this.height = InfinityRun.height;
            this.width = InfinityRun.width;
1031        this.size = 70;

1033        var lingrad = this.createLinearGradient(0,0,0,
    this.height);
            lingrad.addColorStop(0, '#000');
1035        lingrad.addColorStop(1, '#023');
            this.fillStyle = lingrad;
1037        this.fillRect(0,0,this.width, this.height)

1039        this.textAlign = "center";
            this.fillStyle = "White";
1041

            var height = 100;
1043        //logo
            this.drawImage(bglogo,this.width-500,this.height
    -300);
1045        //-----------------------------------
```

```
        if (this.title) {
1047            this.font = Math.floor(this.size*1.3).toString() + "px Bungee";
                this.fillText(this.title, this.width/2, height);
1049            height+= height;
        }
1051
        for (var i = 0; i < items.length; ++i)
1053    {
            var size = Math.floor(this.size*0.8);
1055        if (i == selectedItem)
            {
1057            this.fillStyle = "#A9F5F2";
                size = this.size+5;
1059        }
            this.font = size.toString() + "px Bungee";
1061        height += this.size;
            this.fillText(items[i], InfinityRun.width/2, height);
1063        this.fillStyle = "White";
        }
1065    return results;

1067    /*
         * Settings Tab
1069     *
         */
1071    }else if (GameState == State.Menu && curMenuTab==MenuTab.Controlls){
            this.title = "Controlls";
1073    items = highScore;

1075
        callback = function(volume) { //if (numItem == 0) GameState=State.Started
1077
        switch (volume) {
1079
        }
1081



1083
        };
1085    this.height = InfinityRun.height;
        this.width = InfinityRun.width;
1087
```

```
        var lingrad = this.createLinearGradient(0,0,0,
            this.height);
1089    lingrad.addColorStop(0, '#000');
        lingrad.addColorStop(1, '#023');
1091    this.fillStyle = lingrad;
        this.fillRect(0,0,this.width, this.height, items[
            i]);
1093
        this.textAlign = "center";
1095    this.fillStyle = "White";

1097    var width = 10;
        var height = 150;
1099
        if (this.title) {
1101            this.font = Math.floor(this.size*1.3).
                    toString() + "px Bungee";
                this.fillText(this.title, this.width/2,
                    150);
1103            height+= height;
        }
1105    var distanceText = 50
        this.font = Math.floor(40).toString() + "px
            Bungee";
1107    this.textAlign = "left";
        //Names
1109    this.fillText("Menu:", this.width/5, 300);
        this.font = Math.floor(20).toString() + "px
            Bungee";
1111    this.fillText("[ESC] - Menu", this.width/5, 300+
            distanceText);
        this.fillText("[Arrow up/down] - To navigate",
            this.width/5, 300+2*distanceText);
1113    this.fillText("[Enter] - Accept", this.width/5,
            300+3*distanceText);

1115    this.font = Math.floor(40).toString() + "px
            Bungee";
        this.textAlign = "left";
1117    //Names
        this.fillText("Game:", this.width/5, 300+5*
            distanceText);
1119    this.font = Math.floor(20).toString() + "px
            Bungee";
        this.fillText("[ESC] - Menu", this.width/5,
            300+6*distanceText);
1121    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);
1123        }
            else if (GameState == State.Menu && curMenuTab==
                MenuTab.Settings){
1125
            this.title = "Settings";
1127        items = [0,10, 20, 30, 40, 50, 60, 70, 80, 90 ,
                100];

1129        callback = function(volume) { //if (numItem == 0)
                GameState=State.Started

1131        switch (volume) {

1133        }


1135


1137        };

1139        this.height = InfinityRun.height;
            this.width = InfinityRun.width;
1141
            var lingrad = this.createLinearGradient(0,0,0,
                this.height);
1143        lingrad.addColorStop(0, '#000');
            lingrad.addColorStop(1, '#023');
1145        this.fillStyle = lingrad;
            this.fillRect(0,0,this.width, this.height, items[
                i]);
1147
            this.textAlign = "center";
1149        this.fillStyle = "White";

1151        var width = 10;
            var height = 10;
1153        var posx = 130;
            var posy = 380;
1155        this.space = 15;
            this.heightincr = 4;
1157
            if (this.title) {
1159                this.font = Math.floor(this.size*1.3).
                        toString() + "px Bungee";
                    this.fillText(this.title, this.width/2,
                        150);
1161                height+= height;
```

```
        }
1163
        this.font = "55px Bungee";
1165    if (settingsItem==0) {
        this.fillStyle = "#A9F5F2";
1167    }
        this.fillText('Volume', 240, 300);
1169
        this.fillStyle = "White";
1171

1173    for (var i = 0; i < items.length; ++i) {
                var size = Math.floor(this.size*0.8);
1175            if (i == audioItem && settingsItem==0)
                {
1177                    this.fillStyle = "#A9F5F2";
                        //size = this.size+5;
1179            }
                this.font = size.toString() + "px Bungee";
1181            posx += this.space;
                posy -= this.heightincr;
1183            height += this.heightincr;

1185            items[i] = this.fillRect(posx,posy,width,height);

1187            //this.fillText(items[i], InfinityRun.width/2, height);
                this.fillStyle = "White";
1189
        }
1191
        //Graphic Settings
1193    this.fillStyle = "White";
        if (settingsItem==1) {
1195    this.fillStyle = "#A9F5F2";
        }
1197    this.fillText('Graphics', 240, 500);
        this.fillStyle = "White";
1199    switch (vgaquality) {
                //Low
1201      case 0:
            this.fillText('Mid', 500,600 );
1203            this.fillText('High', 700, 600);
                if (settingsItem == 1){
1205            this.fillStyle = "#A9F5F2";
                }
```

```
1207                this.fillText('Low', 240, 600);
                    break;
1209                //mid
            case 1:

1211
                this.fillText('Low', 240, 600);
1213            this.fillText('High', 700, 600);
                if (settingsItem == 1){
1215            this.fillStyle = "#A9F5F2";
                }
1217            this.fillText('Mid', 500,600 );

1219          break;
                //high
1221        case 2:
              this.fillText('Mid', 500,600 );
1223            this.fillText('Low', 240, 600);
                if (settingsItem == 1){
1225            this.fillStyle = "#A9F5F2";
                }
1227            this.fillText('High', 700, 600);

1229            break;

1231      }
          //Filter settings
1233      this.fillStyle = "White";
          if (settingsItem==2) {
1235      this.fillStyle = "#A9F5F2";
          }
1237      this.fillText('Filters', InfinityRun.width-300,
             300);
          this.fillStyle = "White";
1239      if(setFilters)
          {
1241          this.fillText('Off', InfinityRun.width-200,
                 400);
              if (settingsItem==2)
1243          {
              this.fillStyle = "#A9F5F2";
1245          }
        this.fillText('On', InfinityRun.width-350, 400);
1247      }
          else
1249      {
              this.fillText('On', InfinityRun.width-350,
                 400);
1251      if (settingsItem==2)
```

```
            {
1253        this.fillStyle = "#A9F5F2";
            }
1255    this.fillText('Off', InfinityRun.width-200, 400);
        }
1257
        //-----------------------------------------------------------------------------

1259    /*
         * Highscore Tab
1261     *
         */
1263
        } else if (GameState == State.Menu && curMenuTab == MenuTab.Highscore) {
1265

1267    this.title = "Highscore";
        items = highScore;
1269

1271    callback = function(volume) { //if (numItem == 0) GameState=State.Started

1273    switch (volume) {

1275    }


1277


1279    };
        this.height = InfinityRun.height;
1281    this.width = InfinityRun.width;

1283    var lingrad = this.createLinearGradient(0,0,0, this.height);
        lingrad.addColorStop(0, '#000');
1285    lingrad.addColorStop(1, '#023');
        this.fillStyle = lingrad;
1287    this.fillRect(0,0,this.width, this.height, items[i]);

1289    this.textAlign = "center";
        this.fillStyle = "White";
1291
        var width = 10;
1293    var height = 100;
```

```
1295
        if (this.title) {
1297            this.font = Math.floor(this.size*1.3).
                    toString() + "px Bungee";
                this.fillText(this.title, this.width/2,
                    150);
1299            height+= height;
        }
1301
        var rank = 1;
1303
        for (var i = 0; i < items.length; ++i)
1305    {

1307            var size = Math.floor(this.size*0.8);
                if (i == selectedItem)
1309            {
                        this.fillStyle = "#A9F5F2";
1311                    size = this.size+5;
                }
1313            this.font = 0.6*size.toString() + "px
                    Bungee";
                height += 50;
1315            this.fillText(rank + ". " + items[i],
                    InfinityRun.width/2, height);
                this.fillStyle = "White";
1317
                rank++;
1319    }

1321    }
    // Credits Menu
1323    else if (GameState == State.Menu && curMenuTab ==
            MenuTab.Credits) {

1325    this.title = "Credits";
        items = highScore;
1327
        callback = function(volume) { //if (numItem == 0)
            GameState=State.Started
1329
        switch (volume) {
1331
        }
1333


1335
```

```
        };
1337    this.height = InfinityRun.height;
        this.width = InfinityRun.width;
1339
        var lingrad = this.createLinearGradient(0,0,0,
           this.height);
1341    lingrad.addColorStop(0, '#000');
        lingrad.addColorStop(1, '#023');
1343    this.fillStyle = lingrad;
        this.fillRect(0,0,this.width, this.height, items[
           i]);
1345
        this.textAlign = "center";
1347    this.fillStyle = "White";

1349    var width = 10;
        var height = 150;
1351


1353    if (this.title) {
              this.font = Math.floor(this.size*1.3).
                 toString() + "px Bungee";
1355          this.fillText(this.title, this.width/2,
                 150);
              height+= height;
1357    }
        var distanceText = 50
1359    this.font = Math.floor(50).toString() + "px
           Bungee";
        this.textAlign = "left";
1361    //Names
        this.fillText("Group Members:", this.width/5,
           300);
1363    this.font = Math.floor(40).toString() + "px
           Bungee";
        this.fillText("  Florian Durli", this.width/5,
           300+distanceText);
1365    this.fillText("  Koray Emtekin", this.width/5,
           300+2*distanceText);
        this.fillText("  Jannik Ivosevic", this.width/5,
           300+3*distanceText);
1367    this.fillText("  Marco Mayer", this.width/5,
           300+4*distanceText);
        this.fillText("  Johannes But", this.width/5,
           300+5*distanceText);
1369    //bottom info
        this.font = Math.floor(15).toString() + "px Times
           New Roman";
```

```
1371            this.textAlign = "center";
                distanceText = 20;
1373            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);
                this.fillText("Sounds: freesounds.org Special thanks to Jack Rugil for his Parrallax Skyline ",this.width/2, this.height-distanceText-5);
1375            this.fillText("2016",this.width/2, this.height-8);
                }
1377
        //Debug
1379    if (debug) {
            this.font = '16pt Arial';
1381        this.fillStyle = '#181818';
            this.fillText('Record: ' + s + " " + sc/*this.jumpCountRecord*/, this.width - 150, 33);
1383        this.fillStyle = this.scoreColor;
            this.fillText('Jumps: ' + this.jumpCount, this.width - 150, 50);
1385    }

1387 };

1389 InfinityRun.resize = function() {
        /* todo Windowscale optimization
1391     *
         *
1393     */
    }
```

A.2. 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{
  font-family: 'Bungee', cursive;
  background: #cecece;
  border: 1px solid #181818;
}

canvas.sunsetFilter {
  -webkit-animation: sunset-animation 70s;
}

canvas.invertFilter {
  -webkit-animation: invert-animation 20s;
}



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

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

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



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

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

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

A.3. 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>
```