Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent	:	**Gabriela Mai**
Vorgelegt am	:	13. Dezember 2016
Vorgelegt von	:	Gruppe 4

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

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Team

Abbildung 1.: Florian Durli

Abbildung 2.: Jannik Ivosevic

Abbildung 3.: Johannes But

Abbildung 4.: Marco Mayer

Abbildung 5.: Koray Ektekin

1.2. Rollenverteilung

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

Tabelle 1.: Rollenverteilung

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

1.3. Spielidee

1.3.1. Spielkonzept

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

1.3.2. Entwurfsskizze

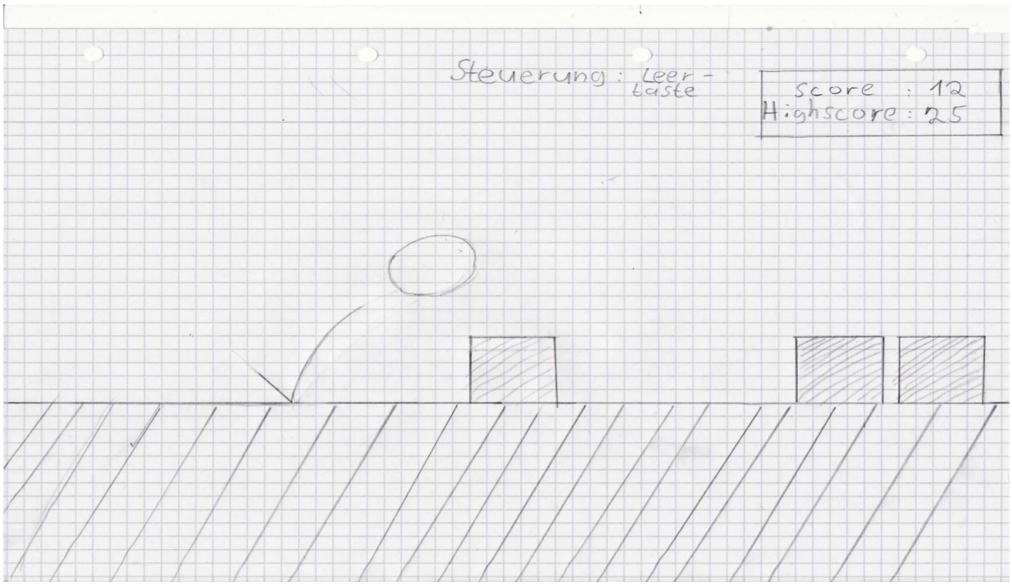

Abbildung 6.: Entwurfsskizze

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

1.3.3. Erforderliche Software

1.3.3.1. Notepad++

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

1.3.3.2. Chrome

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

1.3.3.3. Gimp

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

1.3.3.4. Git/Github

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

2. Phasen

2.1. Entwurf und Anforderungen

2.1.1. Funktionale Anforderungen

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

2.1.2. Nicht funktionale Anforderungen

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

2.1.3. Projektplan

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

Tabelle 2.: Phase 1: Entwurf und Anforderungen

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

Tabelle 3.: Phase 2: Implementierung

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

Tabelle 4.: Phase 3: Test

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

Tabelle 5.: Phase 4: Dokumentation und Präsentation

2.1.4. Releaseplan

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

Tabelle 6.: Releaseplan

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

2.2. Implementation - Zwischenstand

2.2.1. Erfüllte Anforderungen

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

2.2.2. Nicht erfüllte Anforderungen

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

2.2.3. Das Spiel

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

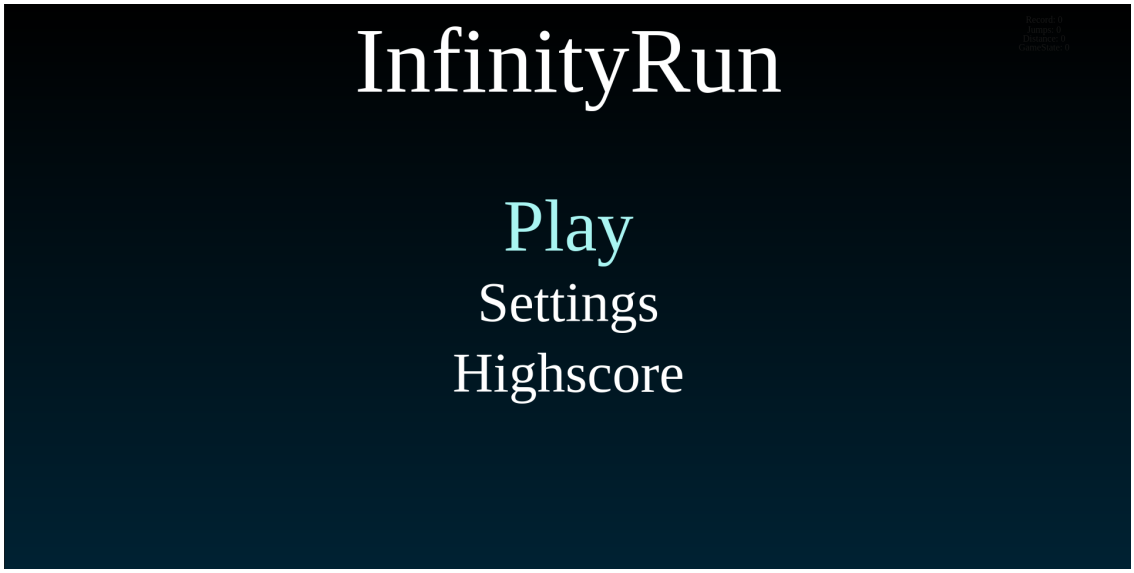

Abbildung 7.: Startbildschirm

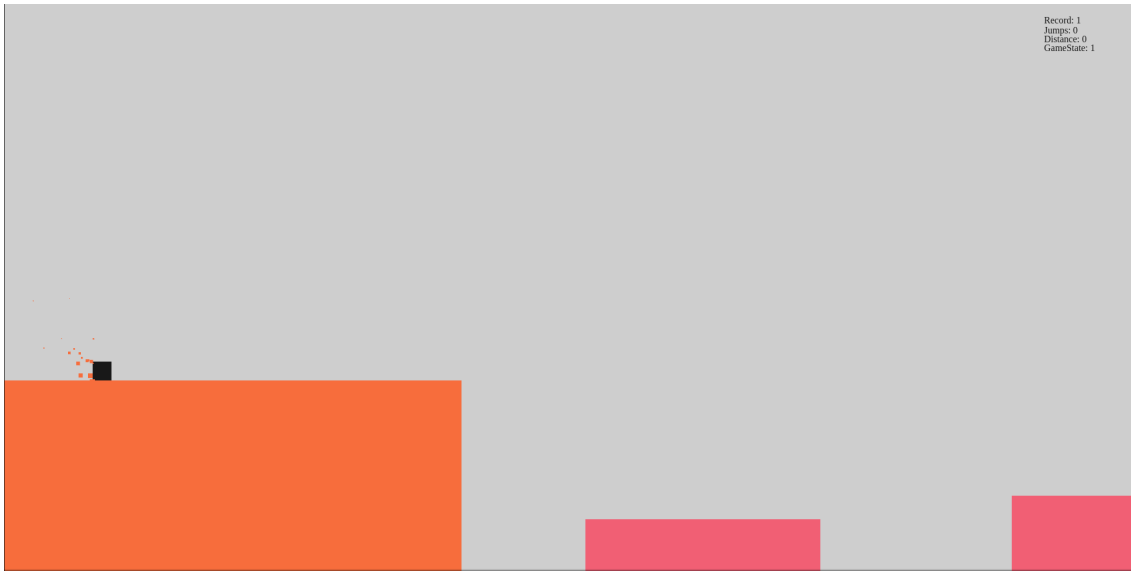

Abbildung 8.: Das Spiel

2.2.4. Bibliothek

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

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

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

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

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

Quelle: [sou]

2.2.5. Code

2.2.5.1. Framework initialisieren

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

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

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

2.2.5.2. Spieler initialisieren

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

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

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

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

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

2.2.5.3. Erstellen der Spielebene

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

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

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

2.2.5.4. Update der Plattformen

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

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

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

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

2.2.5.5. Update der Plattformen

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

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

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

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

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

2.2.6. Nächste Ziele

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

2.3. Implementation - Endstand

2.3.1. Spielkonzept Änderungen

Folgende Spielkonzept Äbnderungen haben wir im laufe der Implementation vorgenommen:

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

2.3.2. Funktionsdiagramm

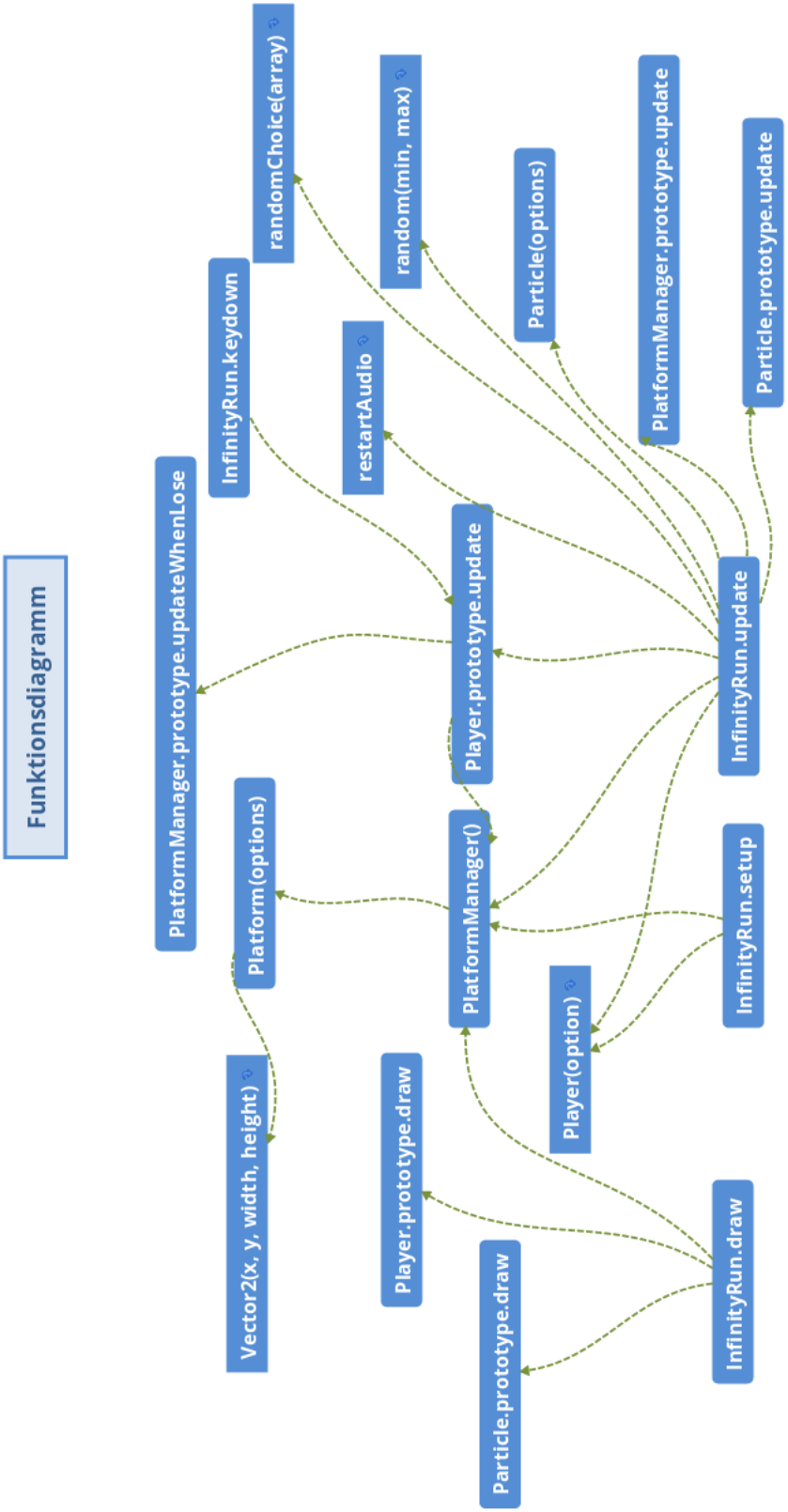

Abbildung 9.: Funktionsdiagramm

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

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

2.3.4. Das Spiel - Endstand

Hier werden zwei Screenshots des derzeitigen Spiels dargestellt. In der Abbildung 10 zu sehen, ist der endgültige Startbildschirm des Spiels. Hier gibt es verschiedene Auswahlmöglichkeiten, die das Spielerlebnis ergänzen. In der Abbildung 11 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.

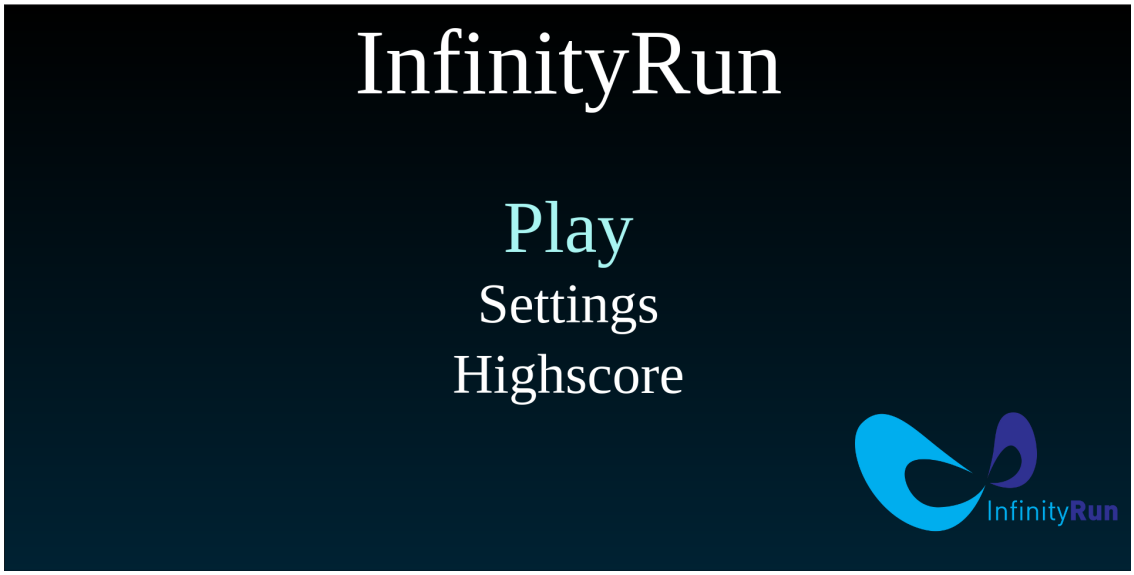

Abbildung 10.: Startbildschirm - Endstand

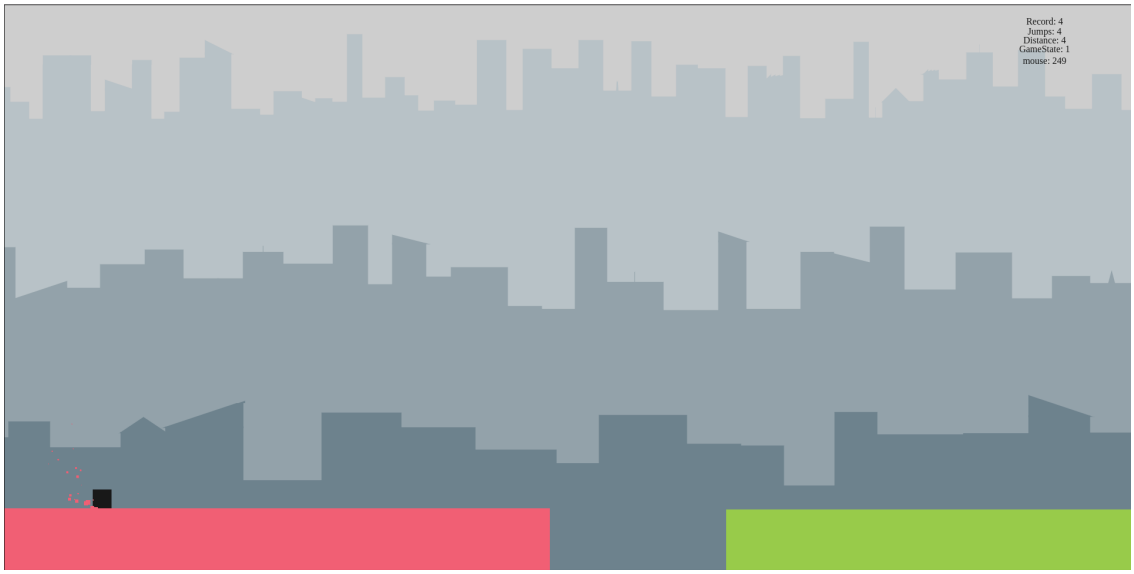

Abbildung 11.: Das Spiel - Endstand

2.3.5. 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 8.: Sound Links

Literaturverzeichnis

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

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

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

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

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

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

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

Eidesstattliche Erklärung

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

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

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

FURTWANGEN, den 13. Dezember 2016 Florian Durli

FURTWANGEN, den 13. Dezember 2016 Jannik Ivosevic

FURTWANGEN, den 13. Dezember 2016 Johannes But

FURTWANGEN, den 13. Dezember 2016 Marco Mayer

FURTWANGEN, den 13. Dezember 2016 Koray Emtekin

A. Anhang

A.1. Github Changelog

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

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

Quelle: [Gru]

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

Tabelle 9.: Github Namen

Changelog:

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

2b8a139 ans77        2016-12-12 Logos hinzugefügt

92b7b28 Slay3r       2016-12-12 Code Cleanup

30b0c6f Slay3r       2016-12-12 Clean up

37f6e94 butjo        2016-12-12 Favicon hinzugefügt

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

8d06d8a butjo        2016-12-09 skyline eingefügt

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

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

```
19 2e6dfbe butjo          2016-11-30 Dirty Code mit versuch ↩
       den background zu implementieren files sind mit prefix↩
        back gekennzeichnetund liegen im Hauptverzeichnis.

21 dda171b Slay3r         2016-11-30 Changelog geändert

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

25 01e309a Slay3r         2016-11-30 Sounds

27 3a05368 Slay3r         2016-11-30 Neue Dokumentation ↩
       hinzufügen

29 7c3596c Slay3r         2016-11-30 Neue Dokumentation ↩
       hinzufügen

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

33 f944707 r4qtor         2016-11-25 Querlesung - Marco

35 8b6bdde Florian Durli  2016-11-25 Abgabe

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

39 b6d7a09 butjo          2016-11-24 Rechtschreibkorrekturen

41 1faa558 r4qtor         2016-11-24 Delete phasen.tex root/

43 51f4a79 r4qtor         2016-11-24 updated phasen.tex

45 62af4b8 Krusher999     2016-11-24 Letzten Codes ↩
       geschrieben

47 93b8965 Florian Durli  2016-11-23 fix

49 9027301 Florian Durli  2016-11-23 Changelog finale Lösung

51 1392138 Florian Durli  2016-11-23 Jojos Description in ↩
       LaTeX

53 25ff037 butjo          2016-11-23 Beschreibung der ↩
       Codeteile in der phasen.tex von Johannes

55 1b2348c Florian Durli  2016-11-23 Changelog Additionally
```

```
57 cf467dc Florian Durli 2016-11-23 Changelog Additionally

59 6db1ad6 butjo         2016-11-23 Merge branch 'master' of
       https://github.com/Slay3r/InfinityRun

61 b924713 Slay3r        2016-11-23 Generated Changelog

63 69dd747 butjo         2016-11-23 Merge branch 'master' of
       https://github.com/Slay3r/InfinityRun

65 203ae2e Slay3r        2016-11-23 Bilder des Spiels

67 d274cfc Slay3r        2016-11-23 Anforderungen

69 a0909ac Slay3r        2016-11-23 Literaturverzichniss

71 19e2f3d Slay3r        2016-11-23 Doku update

73 21889f9 Florian Durli 2016-11-22 Add Changelog

75 743c95a butjo         2016-11-16 Formatierte sketch.min.
       js

77 d64d254 butjo         2016-11-16 Endlose
       Schwierigkeitserhöhung

79 3707b94 butjo         2016-11-16 Präsentation
       Zwischenstand

81 f23c9be Slay3r        2016-11-16 Präsentation ü
       berarbeitet

83 53aa72e butjo         2016-11-16 Präsentation und test

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

87 275bd69 r4qtor        2016-11-15 tiny cleanup & incl.
       menu

89 11a5e55 Slay3r        2016-11-09 Basis implementation

91 c783850 Slay3r        2016-11-09 Bilder hinzugefügt

93 d88d0d2 Slay3r        2016-11-09 Dokumentations Basis

95 39b4705 Florian       2016-10-19 Initial Struktur der
       Ordner und files
```

```
097 093797e Florian Durli 2016-10-19 Delete Requirements

099 56c4aae Florian       2016-10-19 doc Ordner

101 b3b83fd Florian       2016-10-19 Requirements hinzugefügt

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

```
36
  var debug = true;
38 var bgaudio = document.getElementById('backgroundmusic');
  var fxaudio = document.getElementById('fxaudio');
40 var backgroundaudio = "sounds/main1.wav";
  //-------------------------------------------------------------------->

42 //vars background
  var Building, Skyline, dt, skylines;
44
  skylines = [];
46
  dt = 1;
48  var jumpheight = 0
   //logoimage
50 var bglogo = new Image();
  bglogo.src = 'image/logo.png';
52 //-------------------------------------------------------------------->

  function restartAudio()
54 {
        // Check for audio element support.
56      if (window.HTMLAudioElement)
        {
58              try
                {
60                      // Tests the paused attribute and
                           set state.
                        if (bgaudio.ended)
62                      {
                                bgaudio.currentTime = 0;
64                              bgaudio.play();
                        }
66              }
                catch (e)
68              {
                        // Fail silently but show in F12
                           developer tools console
70                      if(window.console && console.
                           error("Error:" + e));
                }
72      }
  }
74
  // randomizer
76 function random(min, max)
  {
78      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')
});

//Mountainintain mouse init
InfinityRun.mouse.x = InfinityRun.width / 10;
InfinityRun.mouse.y = InfinityRun.height;

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


//Mountainfunc
Building = function(config)
{
        return this.reset(config);
};

Building.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.slantedTop = floor(random(0, 10)) === 0;
        this.slantedTopHeight = this.width / random(2, 4);
        this.slantedTopDirection = round(random(0, 1)) === 0;
        this.normalTop = !this.slantedTop && floor(random(0, 10)) === 0;
        this.normalTopHeight = this.width / random(2, 4);
        this.normalTopwindow = round(random(1, 2)) === 0;
```

```
        this.companyTop = !this.slantedTop && !this.
            spireTop && !this.antennaTop && !this.
            normalTop && floor(random(0, 10)) === 0;
122     this.companyTopHeight = this.width / random(4, 6)
            ;
        this.companyTopcount = 4;//round(random(3, 6));
124     this.spireTop = floor(random(0, 15)) === 0;
        this.spireTopWidth = random(this.width * .01,
            this.width * .07);
126     this.spireTopHeight = random(10, 20);
        this.antennaTop = !this.spireTop && floor(random
            (0, 10)) === 0;
128     this.antennaTopWidth = this.layer / 2;
        return this.antennaTopHeight = random(5, 20);
130 };

132 Building.prototype.render = function()
    {
134     InfinityRun.fillStyle = InfinityRun.strokeStyle =
            this.color;
        InfinityRun.lineWidth = 2;
136     InfinityRun.beginPath();
        InfinityRun.rect(this.x, this.y, this.width, this
            .height);
138     InfinityRun.fill();
        InfinityRun.stroke();
140     if (this.slantedTop)
        {
142             InfinityRun.beginPath();
                InfinityRun.moveTo(this.x, this.y);
144             InfinityRun.lineTo(this.x + this.width,
                    this.y);
                if (this.slantedTopDirection)
146             {
                        InfinityRun.lineTo(this.x + this.
                            width, this.y - this.
                            slantedTopHeight);
148             }
                else
150             {
                        InfinityRun.lineTo(this.x, this.y
                            - this.slantedTopHeight);
152             }
                InfinityRun.closePath();
154             InfinityRun.fill();
                InfinityRun.stroke();
156     }
```

```
158         if (this.normalTop)
            {
160                 InfinityRun.beginPath();
                    InfinityRun.moveTo(this.x, this.y);
162                 InfinityRun.lineTo(this.x + this.width,
                       this.y);
                    InfinityRun.lineTo(this.x + (this.width
                       /2), this.y-this.normalTopHeight);
164                 InfinityRun.closePath();
                    InfinityRun.fill();
166                 InfinityRun.stroke();
            }
168
            if (this.companyTop)
170         {
                    var ctc = 1;
172                 while (ctc<=this.companyTopcount)
                    {
174                         InfinityRun.beginPath();
                            InfinityRun.moveTo(this.x , this.
                               y);
176                         InfinityRun.lineTo(this.x + ctc*(
                               this.width/this.
                               companyTopcount), this.y-this.
                               companyTopHeight);
                            InfinityRun.lineTo(this.x + ctc*(
                               this.width/this.
                               companyTopcount), this.y+this.
                               companyTopHeight);
178                         InfinityRun.closePath();
                            InfinityRun.fill();
180                         InfinityRun.stroke();
                            ctc++;
182                 }
            }
184         if (this.spireTop)
            {
186                 InfinityRun.beginPath();
                    InfinityRun.moveTo(this.x + (this.width /
                        2), this.y - this.spireTopHeight);
188                 InfinityRun.lineTo(this.x + (this.width /
                        2) + this.spireTopWidth, this.y);
                    InfinityRun.lineTo(this.x + (this.width /
                        2) - this.spireTopWidth, this.y);
190                 InfinityRun.closePath();
                    InfinityRun.fill();
192                 InfinityRun.stroke();
            }
```

```
194
            if (this.antennaTop)
196         {
                    InfinityRun.beginPath();
                    InfinityRun.moveTo(this.x + (this.width /
198                     2), this.y - this.antennaTopHeight);
                    InfinityRun.lineTo(this.x + (this.width /
                        2), this.y);
200                 InfinityRun.lineWidth = this.
                        antennaTopWidth;
                    return InfinityRun.stroke();
202         }
    };
204
    Skyline = function(config)
206 {
            this.x = 0;
208         this.buildings = [];
            this.layer = config.layer;
210         this.width = {
            min: config.width.min,
212         max: config.width.max
    };
214
    this.height = {
216         min: config.height.min,
            max: config.height.max
218 };

220     this.speed = config.speed;
        this.color = config.color;
222     this.populate();
        return this;
224 };

226 Skyline.prototype.populate = function()
    {
228         var newHeight, newWidth, results, totalWidth;
            totalWidth = 0;
230         results = [];
            while (totalWidth <= InfinityRun.width + (this.
                width.max * 2))
232         {
                    newWidth = round(random(this.width.min,
                        this.width.max));
234                 newHeight = round(random(this.height.min,
                        this.height.max));
                    this.buildings.push(new Building({
```

```
236                 layer: this.layer,
                    x: this.buildings.length === 0 ? 0 : this
                       .buildings[this.buildings.length - 1].
                       x + this.buildings[this.buildings.
                       length - 1].width,
238                 y: InfinityRun.height - newHeight,
                    width: newWidth,
240                 height: newHeight,
                    color: this.color
242                 }));
                    results.push(totalWidth += newWidth);
244         }
            return results;
246 };

248 Skyline.prototype.update = function()
    {
250         var firstBuilding, lastBuilding, newHeight,
               newWidth;
            if (InfinityRun.accelerationTweening==0)
252         {
                    this.x-=((150) * this.speed) * dt;
254         }
            else
256         {
                    this.x -= ((InfinityRun.
                       accelerationTweening*330) * this.speed
                       ) * dt;
258         }

260         firstBuilding = this.buildings[0];
            if (firstBuilding.width + firstBuilding.x + this.
               x < 0)
262         {
                    newWidth = round(random(this.width.min,
                       this.width.max));
264                 newHeight = round(random(this.height.min,
                        this.height.max));
                    lastBuilding = this.buildings[this.
                       buildings.length - 1];
266                 firstBuilding.reset({
                            layer: this.layer,
268                         x: lastBuilding.x + lastBuilding.
                               width,
                            y: InfinityRun.height - newHeight
                               ,
270                         width: newWidth,
                            height: newHeight,
```

```
272                             color: this.color
                        });
274             return this.buildings.push(this.buildings.shift()
                    );
                }
276 };

278 Skyline.prototype.render = function()
    {
280             var i;
                i = this.buildings.length;
282             InfinityRun.save();
                InfinityRun.translate(this.x, (InfinityRun.height
                    - (InfinityRun.height-(-jumpheight*0.5)-400))
                    / 20 * this.layer);
284             while (i--)
                {
286                     this.buildings[i].render(i);
                }
288             return InfinityRun.restore();
    };
290 //-------------------------------------------------------------

    //-------- Vector [Get/Set] Functions ----------
292
    //Set X,Y,Width,Height
294 function Vector2(x, y, width, height)
    {
296             this.x = x;
                this.y = y;
298             this.width = width;
                this.height = height;
300             this.previousX = 0;
                this.previousY = 0;
302 };


304
    // Set X,Y
306 Vector2.prototype.setPosition = function(x, y)
    {
308
                this.previousX = this.x;
310             this.previousY = this.y;
                this.x = x;
312             this.y = y;
    };
314
    // Set X
```

```
316 Vector2.prototype.setX = function(x)
    {
318         this.previousX = this.x;
            this.x = x;
320 };

322 // Set Y
    Vector2.prototype.setY = function(y)
324 {
            this.previousY = this.y;
326         this.y = y;
    };
328
    // Collision / Intersection Top
330 Vector2.prototype.intersects = function(obj)
    {
332     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)
334         {
                return true;
336         }
        return false;
338 };

340 // Collision / Intersection Left
    Vector2.prototype.intersectsLeft = function(obj)
342 {
            if (obj.x < this.x + this.width && obj.y < this.y↩
                + this.height)
344         {
                return true;
346         }
            return false;
348 };

350 //---------- Player ----------

352 function Player(options)
    {
354         this.setPosition(options.x, options.y);
            this.width = options.width;
356         this.height = options.height;
            this.velocityX = 0;
358         this.velocityY = 0;
            this.jumpSize = -13;
360         this.color = '#181818';
```

```
}

Player.prototype = new Vector2;

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

        if (this.y > InfinityRun.height || this.x + this.
           width < 0)
        {
                this.x = 150;
                this.y = 50;
                this.velocityX = 0;
                this.velocityY = 0;
                InfinityRun.jumpCount = 0;
                InfinityRun.acceleration = 0;
                InfinityRun.accelerationTweening = 0;
                InfinityRun.scoreColor = '#181818';
                InfinityRun.platformManager.
                   maxDistanceBetween = 350;
                InfinityRun.platformManager.
                   updateWhenLose();
                fxaudio.pause();
                fxaudio.src = 'sounds/crash.wav';
                fxaudio.load();
                fxaudio.play();
        }

        if ((InfinityRun.keys.UP || InfinityRun.keys.
           SPACE || InfinityRun.keys.W || InfinityRun.
           dragging) && this.velocityY < -8)
        {
                this.velocityY += -0.75;
        }
        if ((InfinityRun.keys.UP || InfinityRun.keys.
           SPACE || InfinityRun.keys.W || InfinityRun.
           dragging) && this.velocityY >0)
        {
                fxaudio.pause();
                fxaudio.src = 'sounds/jump.wav';
                fxaudio.load();
                fxaudio.play();
        }
```

```
};

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

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

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

Platform.prototype = new Vector2;

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

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

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


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

```
            this.maxDistanceBetween - 150, this.
            maxDistanceBetween),
        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: 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;
    if (this.first.x + this.first.width < 0) {
        this.first.width = random(450, InfinityRun.width
            + 200);
        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);
        this.first.height = this.first.y + InfinityRun.
            height + 10;
        this.first.color = randomChoice(this.colors);
    }
```

```
480     this.second.x -= 3 + InfinityRun.acceleration;
        if (this.second.x + this.second.width < 0) {
482         this.second.width = random(450, InfinityRun.width
                + 200);
            this.second.x = (this.first.x + this.first.width)
                + random(this.maxDistanceBetween - 150, this.
               maxDistanceBetween);
484         this.second.y = random(this.first.y - 32,
               InfinityRun.height - 80);
            this.second.height = this.second.y + InfinityRun.
               height + 10;
486         this.second.color = randomChoice(this.colors);
        }
488
        this.third.x -= 3 + InfinityRun.acceleration;
490     if (this.third.x + this.third.width < 0) {
            this.third.width = random(450, InfinityRun.width
               + 200);
492         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);
494         this.third.height = this.third.y + InfinityRun.
               height + 10;
            this.third.color = randomChoice(this.colors);
496     }

498 };


500

502 // reset / new Game: set Starting Platform Parameters
    PlatformManager.prototype.updateWhenLose = function() {
504
        this.first.x = 300;
506     this.first.color = randomChoice(this.colors);
        this.first.y = InfinityRun.width / random(2, 3);
508     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);

510
    };
512
```

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

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

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

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

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

InfinityRun.setup = function() {

    this.jumpCount = 0;
    this.acceleration = 0;
    this.accelerationTweening = 0;
    this.player = new Player({
        x: 150,
        y: 30,
        width: 32,
        height: 32
    });
        bgaudio.pause();
        bgaudio.src = 'sounds/menu.wav';
        bgaudio.load();
        bgaudio.play();

    this.platformManager = new PlatformManager();

    this.particles = [];
```

```
556     this.particlesIndex = 0;
        this.particlesMax = 20;
558     this.collidedPlatform = null;
        this.scoreColor = '#181818';
560     this.jumpCountRecord = 0;
            //-----------------------------------------
562         //bg add
        var i, results;
564     i = 3;
        results = [];
566     while (i--) {
          results.push(skylines.push(new Skyline({
568         layer: i + 1,
            width: {
570           min: (i + 1) * 20,
              max: (i + 1) * 50
572         },
            height: {
574           min: InfinityRun.height-200 - (i * round(
                InfinityRun.height/3)),
              max: InfinityRun.height-50 - (i * round(
                InfinityRun.height/3))
576         },
            speed: (i + 1) * .003,
578         color: 'hsl( 200, ' + (((i + 1) * 1) + 10) + '%,
                ' + (75 - (i * 13)) + '% )'
          })));
580     }
            return results;
582         //-----------------------------------------


584
    };
586 //-------------------------------------------------
    //clear func bg
588 InfinityRun.clear = function() {
        return InfinityRun.clearRect(0, 0, InfinityRun.width,
            InfinityRun.height);
590   };
    //-------------------------------------------------
592


594
    InfinityRun.update = function() {
596         if (GameState == State.Started) {
            //-----------------------------------------
598         //clear func bg
            var i, results;
```

```
600     dt = InfinityRun.dt < .1 ? .1 : InfinityRun.dt / 16;
        dt = dt > 5 ? 5 : dt;
602     i = skylines.length;
        results = [];
604     while (i--) {
          results.push(skylines[i].update(i));
606     }
        //return results;
608         //-----------------------------------------------------
        this.player.update();
610     restartAudio();
        switch (this.jumpCount) {
612                 case 0:
                            bgaudio.pause();
614                         bgaudio.src = 'sounds/main1.wav';
                            bgaudio.load();
616                         bgaudio.play();
                break;
618         case 10:
                this.accelerationTweening = 1;
620             this.platformManager.maxDistanceBetween =
                   430;
                //this.scoreColor = '#076C00';
622                         bgaudio.pause();
                            bgaudio.src = 'sounds/main2.wav';
624                         bgaudio.load();
                            bgaudio.play();
626                         fxaudio.pause();
                            fxaudio.src = 'sounds/levelup.wav
                               ';
628                         fxaudio.load();
                            fxaudio.play();
630             break;
            case 25:
632             this.accelerationTweening = 2;
                this.platformManager.maxDistanceBetween =
                   530;
634             //this.scoreColor = '#0300A9';
                            bgaudio.pause();
636                         bgaudio.src = 'sounds/main3.wav';
                            bgaudio.load();
638                         bgaudio.play();
                            fxaudio.pause();
640                         fxaudio.src = 'sounds/levelup.wav
                               ';
                            fxaudio.load();
642                         fxaudio.play();
                break;
```

```
644         case 40:
                this.accelerationTweening = 3;
646             this.platformManager.maxDistanceBetween =
                    580;
                //this.scoreColor = '#9F8F00';
648                         bgaudio.pause();
                            bgaudio.src = 'sounds/main4.wav';
650                         bgaudio.load();
                            bgaudio.play();
652                         fxaudio.pause();
                            fxaudio.src = 'sounds/levelup.wav
                                ';
654                         fxaudio.load();
                            fxaudio.play();
656             break;
    }
658
    this.acceleration += (this.accelerationTweening -
        this.acceleration) * 0.01;
660


662
    for (i = 0; i < this.platformManager.platforms.length
        ; i++) {
664     if (this.player.intersects(this.platformManager.
            platforms[i])) {
            this.collidedPlatform = this.platformManager.
                platforms[i];
666         if (this.player.y < this.platformManager.
                platforms[i].y) {
                this.player.y = this.platformManager.
                    platforms[i].y;
668
                // Gravity after Collision with Platform
670             this.player.velocityY = 0;
            }
672
            this.player.x = this.player.previousX;
674         this.player.y = this.player.previousY;

676         this.particles[(this.particlesIndex++) % this
                .particlesMax] = new Particle({
                x: this.player.x,
678             y: this.player.y + this.player.height,
                color: this.collidedPlatform.color
680         });

682         if (this.player.intersectsLeft(this.
```

```
                platformManager.platforms[i])) {
                 this.player.x = this.collidedPlatform.x -
                     64;
684              for (i = 0; i < 10; i++) {
                     // SpawnParticles @PlayerPostion with
                         intersecting Platform Color
686                  this.particles[(this.particlesIndex
                        ++) % this.particlesMax] = new
                        Particle({
                         x: this.player.x + this.player.
                            width,
688                      y: random(this.player.y, this.
                            player.y + this.player.height)
                            ,
                         velocityY: random(-30, 30),
690                      color: randomChoice(['#181818',
                            '#181818', this.
                            collidedPlatform.color])
                     });
692              };

694              // bounce player / push him away (effect)
                 this.player.velocityY = -10 + -(this.
                    acceleration * 4);
696              this.player.velocityX = -20 + -(this.
                    acceleration * 4);
             } else {
698
                 // ---------- Controller ----------
700              // dragging: Mouse click & touch support
                 if (this.dragging || this.keys.SPACE ||
                    this.keys.UP || this.keys.W) {
702                  this.player.velocityY = this.player.
                        jumpSize;
                     this.jumpCount++;
704
                     if (this.jumpCount > this.
                        jumpCountRecord) {
706                      this.jumpCountRecord = this.
                            jumpCount;
                     }
708              }
             }
710      }
    };
712
    for (i = 0; i < this.platformManager.platforms.length
        ; i++) {
```

```
714             this.platformManager.update();
        };
716
        for (i = 0; i < this.particles.length; i++) {
718             this.particles[i].update();
        };
720             //-------------------------------------------------------
                //bg
722             return results;
                //-------------------------------------------------------
724 }

726 };

728 var selectedItem = 0;

730 InfinityRun.keydown = function() {
        if (InfinityRun.keys.ESCAPE && GameState==State.
           Started) {
732                     InfinityRun.clear();
                        GameState = State.Menu;
734                     bgaudio.pause();
                    bgaudio.src = 'sounds/menu.wav';
736                 bgaudio.load();
                        bgaudio.play();
738
            } else if (InfinityRun.keys.ESCAPE && GameState==
               State.Menu) {
740                     GameState = State.Started;
            }
742         if (InfinityRun.keys.UP) {
                        selectedItem = (selectedItem + items.
                           length - 1) % items.length;
744         }
            if (InfinityRun.keys.DOWN) {
746                     selectedItem = (selectedItem + 1) % items
                           .length;
            }
748
            if(InfinityRun.keys.ENTER) {
750                     callback(selectedItem);
            }
752
    }
754
    Menu = function() {
756
            //this.backgroundCallback = null;
```

```
758 }

760
//---------- Draw ----------
762
InfinityRun.draw = function() {
764         if(GameState == State.Started) {
            //-----------------------------------------------------------------------------

766         //bg draw
            var i, results;
768     i = skylines.length;
        results = [];
770     while (i--) {
          results.push(skylines[i].render(i));
772     }
            //-----------------------------------------------------------------------------

774     this.player.draw();

776     for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
            this.platformManager.platforms[i].draw();
778     };

780     //Draw particles
        for (i = 0; i < this.particles.length; i++) {
782         this.particles[i].draw();
        };
784
            //Draw menu --TODO prototype
786         } else if (GameState == State.Menu) {

788         this.title = "InfinityRun";
            items = ["Play", "Settings", "Highscore"];
790
            callback = function(numItem) { if (numItem == 0)
               GameState=State.Started };
792         this.height = InfinityRun.height;
            this.width = InfinityRun.width;
794         this.size = 120;

796         var lingrad = this.createLinearGradient(0,0,0,
               this.height);
            lingrad.addColorStop(0, '#000');
798         lingrad.addColorStop(1, '#023');
            this.fillStyle = lingrad;
800         this.fillRect(0,0,this.width, this.height)
```

```
802         this.textAlign = "center";
            this.fillStyle = "White";
804
            var height = 150;
806         //logo
            this.drawImage(bglogo,this.width-500,this.height
               -300);
808         //-------------------------------------
            if (this.title) {
810                 this.font = Math.floor(this.size*1.3).
                       toString() + "px Times New Roman";
                    this.fillText(this.title, this.width/2,
                       height);
812                 height+= height;
            }
814
            for (var i = 0; i < items.length; ++i)
816         {
                    var size = Math.floor(this.size*0.8);
818                 if (i == selectedItem)
                    {
820                         this.fillStyle = "#A9F5F2";
                            size = this.size+5;
822                 }
                    this.font = size.toString() + "px Times
                       New Roman";
824                 height += this.size;
                    this.fillText(items[i], InfinityRun.width
                       /2, height);
826                 this.fillStyle = "White";
            }
828         //-------------------------------------------------------------

            //bg dd
830         return results;
            //-------------------------------------------------------------


832
            }
834


836
    //Debug
838 if (debug) {
        this.font = '12pt Arial';
840     this.fillStyle = '#181818';
        this.fillText('Record: ' + this.jumpCountRecord,
```

```
            this.width - 150, 33);
842         this.fillStyle = this.scoreColor;
            this.fillText('Jumps: ' + this.jumpCount, this.
               width - 150, 50);
844         this.fillText('Distance: ' + random (3,6) , this.
               width - 150, 65);
                  this.fillText('mouse: ' + this.mouse.y , 
                     this.width - 150, 100);
846         this.fillText('GameState: ' + GameState, this.
               width - 150, 80);
        }
848
    };
850
    InfinityRun.resize = function() {
852     /* todo Windowscale optimization
         *
854      *
         */
856 }
```

A.3. game.css

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

A.4. index.html

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

```
5         <script type="text/javascript" src="js/sketch.min.js" charset="utf-8"></script>

7     <title>Infinity Run</title>

9     <link href="css/game.css" rel="stylesheet" type="text/css">
          <link rel="shortcut icon" type="image/x-icon" href="image/favicon.png">
11 </head>
  <body>
13 <!-- Game div -->
  <div id="container">
15
  </div>
17 <audio id="backgroundmusic" ></audio>
  <audio id="fxaudio" ></audio>
19 <script type="text/javascript" src="js/game.js" charset="utf-8"></script>
  </body>
21 </html>
```