Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent	:	**Gabriela Mai**
Vorgelegt am	:	14. 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

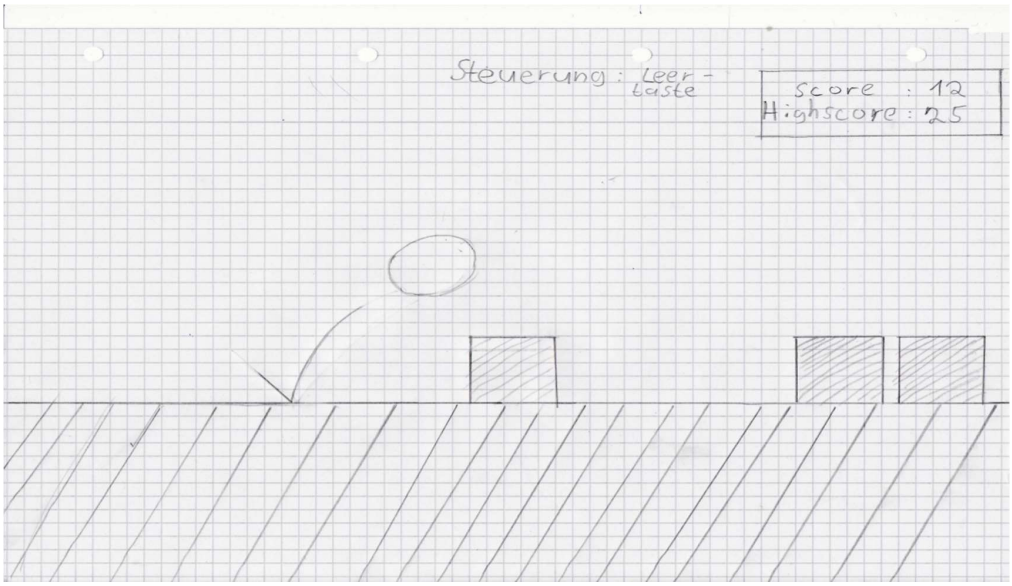

Abbildung 6.: Entwurfsskizze

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.

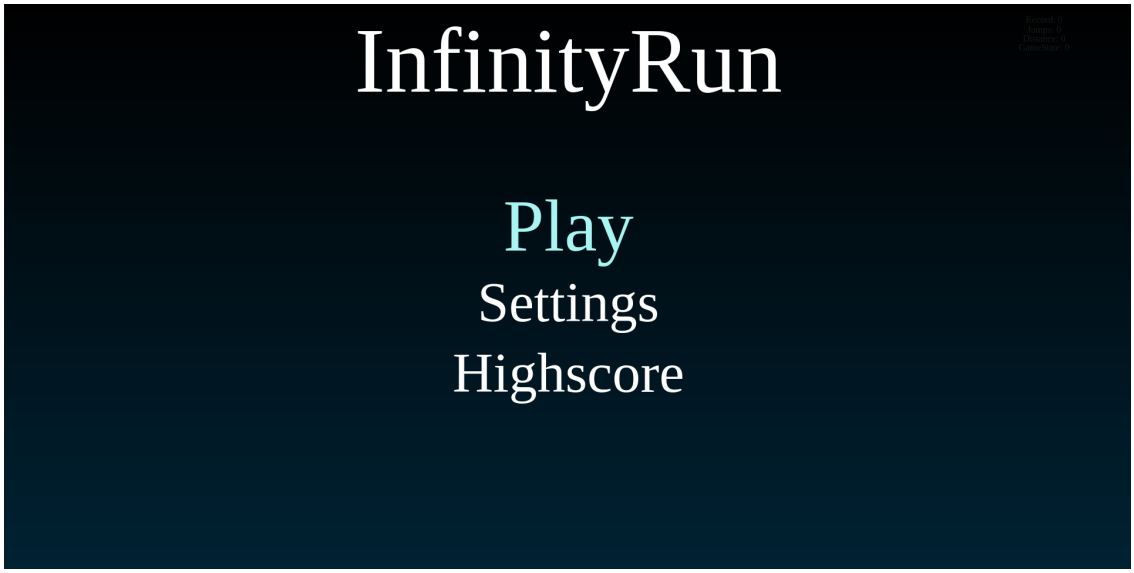

Abbildung 7.: Startbildschirm

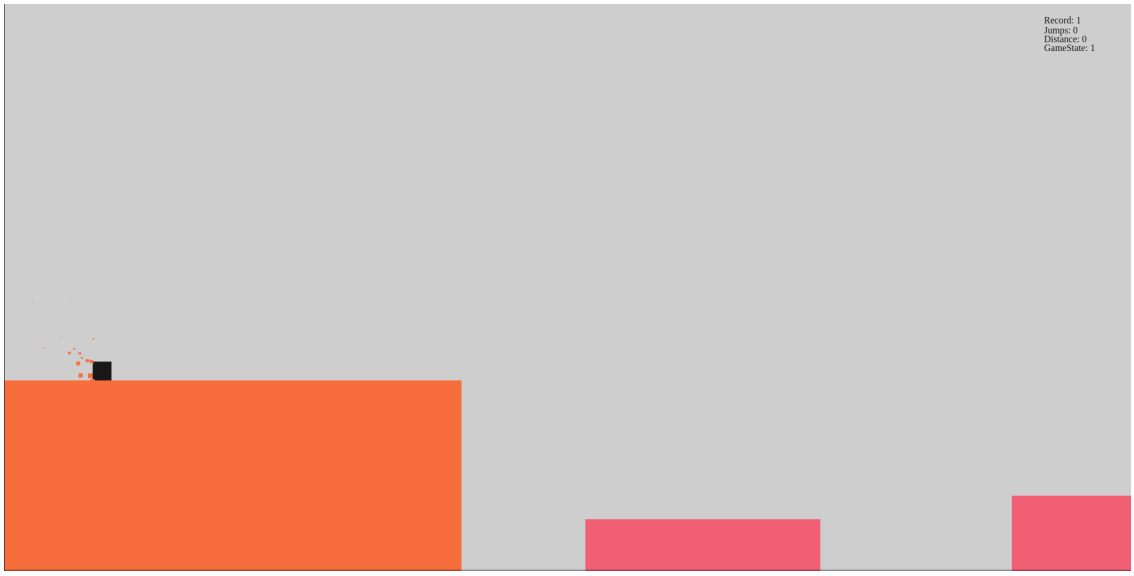

Abbildung 8.: Das Spiel

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

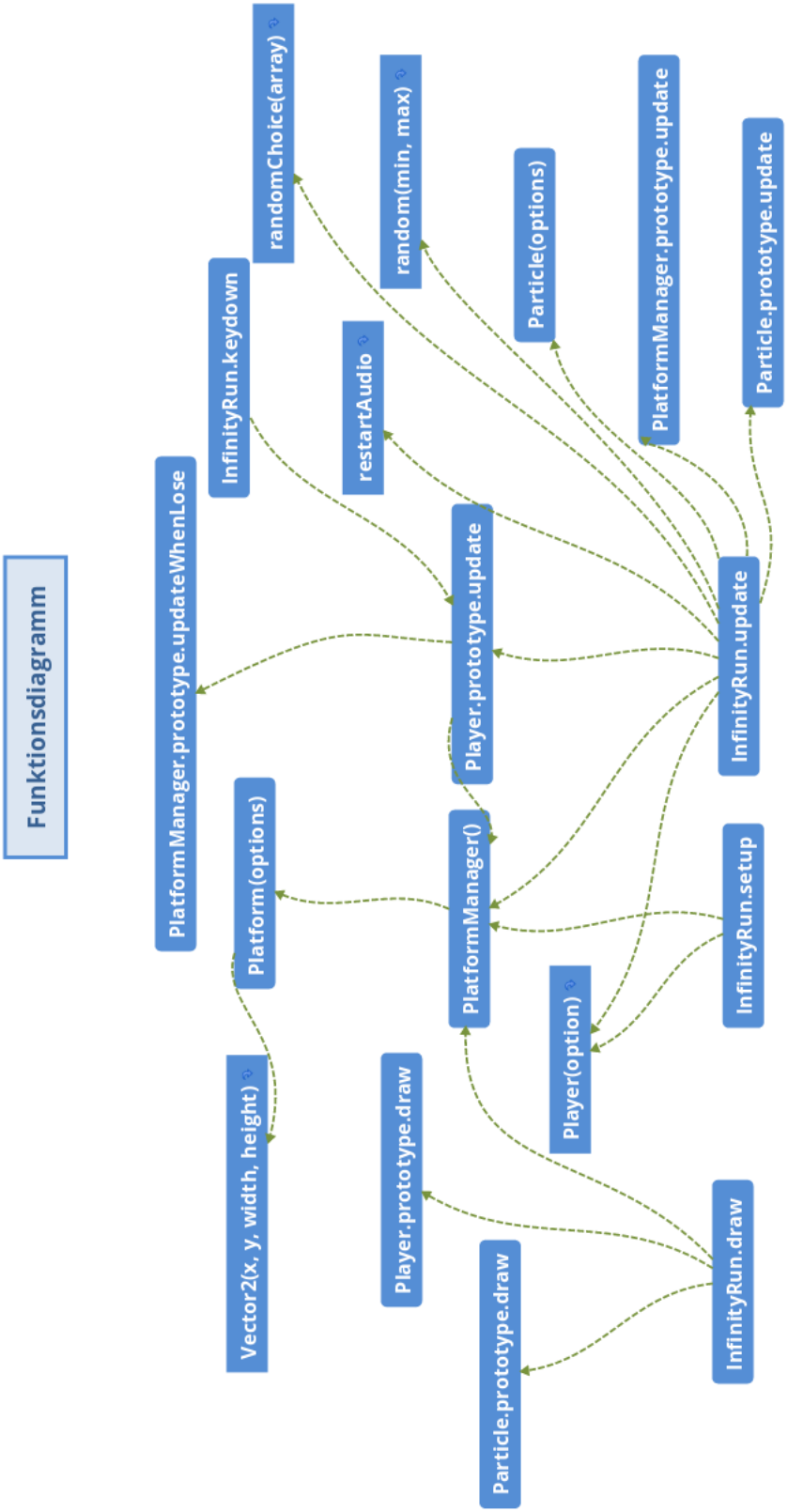

Abbildung 9.: Funktionsdiagramm

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

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. 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()
{
        var firstTower, lastTower, newHeight, newWidth;
```

```
        if (InfinityRun.accelerationTweening==0)
        {
                this.x-=((150) * this.speed) * dt;
        }
        else
        {
                this.x -= ((InfinityRun.
                   accelerationTweening*330) * this.speed
                   ) * dt;
        }

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

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.

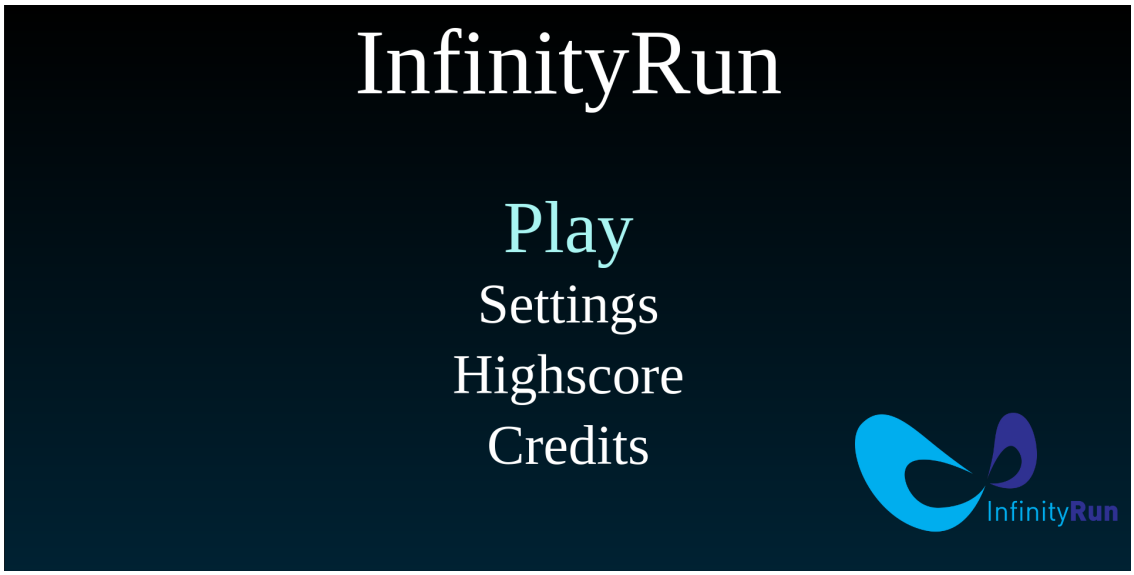

Abbildung 11.: Startbildschirm - Endstand

Abbildung 12.: Das Spiel - Endstand

2.3.6. 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 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 14. Dezember 2016 Florian Durli

FURTWANGEN, den 14. Dezember 2016 Jannik Ivosevic

FURTWANGEN, den 14. Dezember 2016 Johannes But

FURTWANGEN, den 14. Dezember 2016 Marco Mayer

FURTWANGEN, den 14. 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 abhängig 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 MenuTab = {Main:0, Settings:1, Highscore:2, Credits
   :3};
   var curMenuTab = MenuTab.Main;
38
   var dateNow;
40 var date = new Date();
   var timePassed;
42
   var highScore = new Array(10);
44 highScore = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

46 var debug = true;
   var bgaudio = document.getElementById('backgroundmusic');
48 var fxaudio = document.getElementById('fxaudio');
   var backgroundaudio = "sounds/main1.wav";
50


52 //-------------------------------------------------------------

   //vars background
54 var Tower, Street, dt, Town;
   //Building
56
   Town = [];
58
   dt = 1;
60  var jumpheight = 0
    //logoimage
62 var bglogo = new Image();
   bglogo.src = 'image/logo.png';
64 //-------------------------------------------------------------

   function restartAudio()
66 {
        // Check for audio element support.
68      if (window.HTMLAudioElement)
        {
70              try
                {
72                      // Tests the paused attribute and
                            set state.
                        if (bgaudio.ended)
74                      {
                                bgaudio.currentTime = 0;
76                              bgaudio.play();
                        }
78              }
                catch (e)
```

```
80                  {
                            // Fail silently but show in F12 developer tools console
82                          if(window.console && console.error("Error:" + e));
                    }
84          }
}
86
// randomizer
88 function random(min, max) {
    return Math.round(min + (Math.random() * (max - min)));
90 }

92 function randomChoice(array) {
    return array[Math.round(random(0, array.length - 1))];
94 }


96
//initialize Sketch Framework
98 var InfinityRun = Sketch.create({
    fullscreen: true,
100 width: 640,
    height: 360,
102 container: document.getElementById('container')
});
104

106 //-------------------------------------------------------------------------


108 //bg func
Tower = function(config)
110 {
        return this.reset(config);
112 };

114 Tower.prototype.reset = function(config)
{
116         this.layer = config.layer;
        this.x = config.x;
118         this.y = config.y;
        this.width = config.width;
120         this.height = config.height;
        this.color = config.color;
122                 this.summitTop = floor(random(0, 15)) ===
```

```
                0;
        this.summitTopWidth = random(this.width * .01,
            this.width * .07);
124     this.summitTopHeight = random(10, 20);
        this.singleroofTop = floor(random(0, 10)) === 0;
126     this.singleroofTopHeight = this.width / random(2,
            4);
        this.singleroofTopDirection = round(random(0, 1))
            === 0;
128         this.normalTop = !this.singleroofTop &&
                floor(random(0, 10)) === 0;
        this.normalTopHeight = this.width / random(2, 4);
130     this.normalTopchimney = round(random(0, 1)) ===
            0;
            this.coneTop = !this.singleroofTop && !
                this.normalTop && floor(random(0, 10))
                === 0;
132     this.coneTopHeight = this.width / random(3, 4);
            this.coneTopWidth = this.width / random
                (1, 2);
134     this.coneTopeflat = round(random(0, 1)) === 0;
            this.companyTop = !this.singleroofTop &&
                !this.summitTop && !this.radioTop && !
                this.normalTop && floor(random(0, 10))
                === 0;
136     this.companyTopHeight = this.width / random(4, 6)
            ;
        this.companyTopcount = 4;//round(random(3, 6));
138     this.radioTop = !this.summitTop && floor(random
            (0, 10)) === 0;
        this.radioTopWidth = this.layer / 2;
140     return this.radioTopHeight = random(6, 30);
};
142
Tower.prototype.render = function()
144 {
        InfinityRun.fillStyle = InfinityRun.strokeStyle =
            this.color;
146     InfinityRun.lineWidth = 2;
        InfinityRun.beginPath();
148     InfinityRun.rect(this.x, this.y, this.width, this
            .height);
        InfinityRun.fill();
150     InfinityRun.stroke();
        if (this.singleroofTop)
152     {
            InfinityRun.beginPath();
154         InfinityRun.moveTo(this.x, this.y);
```

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

```
                                        );
                                    InfinityRun.closePath();
                                    InfinityRun.fill();
                                    InfinityRun.stroke();
                            }
                    }
                    if (this.coneTop)
                    {
                            InfinityRun.beginPath();
                            InfinityRun.moveTo(this.x, this.y);
                            InfinityRun.lineTo(this.x + (this.width-
                               this.coneTopWidth)/2, this.y-this.
                               coneTopHeight);
                                    if(!this.coneTopeflat)
                                    {
                                            InfinityRun.lineTo(this.x
                                               +(this.width/2), this.
                                               y-(this.coneTopHeight
                                               *1.3));
                                    }
                                    InfinityRun.lineTo(this.x + ((
                                       this.width-this.coneTopWidth)
                                       /2)+this.coneTopWidth, this.y-
                                       this.coneTopHeight);
                                    InfinityRun.lineTo(this.x + this.
                                       width, this.y);
                            InfinityRun.closePath();
                            InfinityRun.fill();
                            InfinityRun.stroke();

                    }
                    if (this.companyTop)
                    {
                            var ctc = 1;
                            while (ctc<=this.companyTopcount)
                            {
                                    InfinityRun.beginPath();
                                    InfinityRun.moveTo(this.x , this.
                                       y);
                                    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);
                                    InfinityRun.closePath();
```

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

232         if (this.radioTop)
            {
234                 InfinityRun.beginPath();
                    InfinityRun.moveTo(this.x + (this.width /
                        2), this.y - this.radioTopHeight);
236                 InfinityRun.lineTo(this.x + (this.width /
                        2), this.y);
                    InfinityRun.lineWidth = this.
                       radioTopWidth;
238                 return InfinityRun.stroke();
            }
240 };

242 Street = function(config)
    {
244         this.x = 0;
            this.alltowers = [];
246         this.layer = config.layer;
            this.width = {
248         min: config.width.min,
            max: config.width.max
250 };

252 this.height = {
            min: config.height.min,
254         max: config.height.max
    };
256

        this.speed = config.speed;
```

```
258     this.color = config.color;
        this.populate();
260     return this;
};
262
Street.prototype.populate = function()
264 {
        var newHeight, newWidth, results, totalWidth;
266     totalWidth = 0;
        results = [];
268     while (totalWidth <= InfinityRun.width + (this.
           width.max * 2))
        {
270             newWidth = round(random(this.width.min,
                   this.width.max));
                newHeight = round(random(this.height.min,
                    this.height.max));
272             this.alltowers.push(new Tower({
                layer: this.layer,
274             x: this.alltowers.length === 0 ? 0 : this
                   .alltowers[this.alltowers.length - 1].
                   x + this.alltowers[this.alltowers.
                   length - 1].width,
                y: InfinityRun.height - newHeight,
276             width: newWidth,
                height: newHeight,
278             color: this.color
                }));
280             results.push(totalWidth += newWidth);
        }
282     return results;
};
284
Street.prototype.update = function()
286 {
        var firstTower, lastTower, newHeight, newWidth;
288     if (InfinityRun.accelerationTweening==0)
        {
290             this.x-=((150) * this.speed) * dt;
        }
292     else
        {
294 this.x -= ((InfinityRun.accelerationTweening*330) *
           this.speed) * dt;
        }
296
        firstTower = this.alltowers[0];
298     if (firstTower.width + firstTower.x + this.x < 0)
```

```
        {
300             newWidth = round(random(this.width.min,
                    this.width.max));
                newHeight = round(random(this.height.min,
                     this.height.max));
302             lastTower = this.alltowers[this.alltowers
                    .length - 1];
                firstTower.reset({
304                     layer: this.layer,
                        x: lastTower.x + lastTower.width,
306                     y: InfinityRun.height - newHeight
                            ,
                        width: newWidth,
308                     height: newHeight,
                        color: this.color
310             });
        return this.alltowers.push(this.alltowers.shift()
            );
312     }
};
314
Street.prototype.render = function()
316 {
        var i;
318     i = this.alltowers.length;
        InfinityRun.save();
320     InfinityRun.translate(this.x, (InfinityRun.height
            - (InfinityRun.height-(-jumpheight*0.5)-400))
            / 20 * this.layer);
        while (i--)
322     {
            this.alltowers[i].render(i);
324     }
        return InfinityRun.restore();
326 };
//-------------------------------------------------------------

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

330 //Set X,Y,Width,Height
function Vector2(x, y, width, height) {
332     this.x = x;
    this.y = y;
334     this.width = width;
    this.height = height;
336     this.previousX = 0;
    this.previousY = 0;
338 };
```

```
340
   // Set X,Y
342 Vector2.prototype.setPosition = function(x, y) {

344     this.previousX = this.x;
        this.previousY = this.y;
346
        this.x = x;
348     this.y = y;

350 };
   // Set X
352 Vector2.prototype.setX = function(x) {

354     this.previousX = this.x;
        this.x = x;
356
   };
358
   // Set Y
360 Vector2.prototype.setY = function(y) {

362     this.previousY = this.y;
        this.y = y;
364
   };
366
   // Collision / Intersection Top
368 Vector2.prototype.intersects = function(obj) {

370     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) {
372         return true;
        }
374
        return false;
376 };

378 // Collision / Intersection Left
   Vector2.prototype.intersectsLeft = function(obj) {
380
        if (obj.x < this.x + this.width && obj.y < this.y + ↩
           this.height) {
382         return true;
        }
```

```
384
        return false;
386 };

388 //---------- Player ----------

390 function Player(options) {

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

400 }

402 Player.prototype = new Vector2;

404 Player.prototype.update = function() {
        // Gravity
406     this.velocityY += 1;
          //um bg zu ändern
408       jumpheight=(this.y);
        this.setPosition(this.x + this.velocityX, this.y + ↩
           this.velocityY);
410
        if (this.y > InfinityRun.height || this.x + this.↩
           width < 0) {
412       this.x = 150;
          this.y = 50;
414       this.velocityX = 0;
          this.velocityY = 0;
416       InfinityRun.jumpCount = 0;
          InfinityRun.acceleration = 0;
418       InfinityRun.accelerationTweening = 0;
          InfinityRun.scoreColor = '#181818';
420       InfinityRun.platformManager.maxDistanceBetween = ↩
             350;

422               //InfinityRun.pause();


424


426               var j = 0;
                  var k = 1;
428
```

```
                //update highscore TODO
430             for(var i = 0; i<9; i++) {
                        if (timePassed > highScore[i])  {
432

434

                                // drag all scores down a
                                    value
436

                                if(highScore[j] != 0)  {
438                                     highScore[k] =
                                            highScore[j];
440
                                        j++;
442                                     k++;

444                             }

446                     }
                                        //highScore[0] =
                                            timePassed;
448
                }
450
        InfinityRun.platformManager.updateWhenLose();
452             fxaudio.pause();
            fxaudio.src = 'sounds/crash.wav';
454         fxaudio.load();
            fxaudio.play();
456
                date = new Date();
458             //bg ändern
                //jumpheight=0;
460
    }
462
    if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
         InfinityRun.keys.W || InfinityRun.dragging) &&
       this.velocityY < -8) {
464     this.velocityY += -0.75;
            //jumpheight+=1
466 }

468
    if (InfinityRun.keys.DOWN) {
470     this.velocityY += 1;
    }
```

```
472
};
474
Player.prototype.draw = function() {
476     InfinityRun.fillStyle = this.color;
        InfinityRun.fillRect(this.x, this.y, this.width, this
           .height);
478 };

480 // ---------- Platforms ----------

482 function Platform(options) {
        this.x = options.x;
484     this.y = options.y;
        this.width = options.width;
486     this.height = options.height;
        this.previousX = 0;
488     this.previousY = 0;
        this.color = options.color;
490 }

492 Platform.prototype = new Vector2;

494 Platform.prototype.draw = function() {
        InfinityRun.fillStyle = this.color;
496     InfinityRun.fillRect(this.x, this.y, this.width, this
           .height);
    };
498
    // ---------- Platform Manager ----------
500
    function PlatformManager() {
502     this.maxDistanceBetween = 300;
        this.colors = ['#2ca8c2', '#98cb4a', '#f76d3c', '#
           f15f74', '#5481e6'];
504


506         //first 3 Platforms execept the Starter Platform
        this.first = new Platform({
508         x: 300,
            y: 600,
510         width: 400,
            height: 70
512     })
        this.second = new Platform({
514         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, 800);
            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);
        }

        this.second.x -= 3 + InfinityRun.acceleration;
        if (this.second.x + this.second.width < 0) {
```

```
        this.second.width = random(450, 800);
552     this.second.x = (this.first.x + this.first.width)
            + random(this.maxDistanceBetween - 150, this.
            maxDistanceBetween);
        this.second.y = random(this.first.y - 32,
            InfinityRun.height - 80);
554     this.second.height = this.second.y + InfinityRun.
            height + 10;
        this.second.color = randomChoice(this.colors);
556 }

558 this.third.x -= 3 + InfinityRun.acceleration;
    if (this.third.x + this.third.width < 0) {
560     this.third.width = random(450, 800);
        this.third.x = (this.second.x + this.second.width
            ) + random(this.maxDistanceBetween - 150, this
            .maxDistanceBetween);
562     this.third.y = random(this.second.y - 32,
            InfinityRun.height - 80);
        this.third.height = this.third.y + InfinityRun.
            height + 10;
564     this.third.color = randomChoice(this.colors);
    }
566

};
568


570
// reset / new Game: set Starting Platform Parameters
572 PlatformManager.prototype.updateWhenLose = function() {

574 this.first.x = 300;
    this.first.color = randomChoice(this.colors);
576     this.first.y = 500;
    //this.first.y = InfinityRun.width / random(2, 3);
578 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);

580

};
582
// ---------- Particle System ---------- (Sketch Docs)
584

function Particle(options) {
586 this.x = options.x;
```

```
        this.y = options.y;
588     this.size = 10;
        this.velocityX = options.velocityX || random(-(
            InfinityRun.acceleration * 3) + -8, -(InfinityRun.
            acceleration * 3));
590     this.velocityY = options.velocityY || random(-(
            InfinityRun.acceleration * 3) + -8, -(InfinityRun.
            acceleration * 3));
        this.color = options.color;
592 }

594 Particle.prototype.update = function() {
        this.x += this.velocityX;
596     this.y += this.velocityY;
        this.size *= 0.89;
598 };

600 Particle.prototype.draw = function() {
        InfinityRun.fillStyle = this.color;
602     InfinityRun.fillRect(this.x, this.y, this.size, this.
            size);
    };
604
    /*********************************************/
606
    InfinityRun.setup = function() {
608
        this.jumpCount = 0;
610     this.acceleration = 0;
        this.accelerationTweening = 0;
612     this.player = new Player({
            x: 150,
614         y: 30,
            width: 32,
616         height: 32
        });
618         bgaudio.pause();
            bgaudio.src = 'sounds/menu.wav';
620         bgaudio.load();
            bgaudio.play();
622
        this.platformManager = new PlatformManager();
624
        this.particles = [];
626     this.particlesIndex = 0;
        this.particlesMax = 20;
628     this.collidedPlatform = null;
        this.scoreColor = '#181818';
```

```
630     this.jumpCountRecord = 0;
        //-----------------------------------------
632         //bg add
        var i, results;
634     i = 3;
        results = [];
636     while (i--) {
          results.push(Town.push(new Street({
638         layer: i + 1,
            width: {
640           min: (i + 1) * 20,
              max: (i + 1) * 50
642         },
            height: {
644           min: InfinityRun.height-200 - (i * round(
                InfinityRun.height/3)),
              max: InfinityRun.height-50 - (i * round(
                InfinityRun.height/3))
646         },
            speed: (i + 1) * .003,
648         color: 'hsl( 200, ' + (((i + 1) * 1) + 10) + '%,
                ' + (75 - (i * 13)) + '% )'
          })));
650     }
            return results;
652         //-----------------------------------------


654
    };
656 //-------------------------------------------------
    //clear func bg
658 InfinityRun.clear = function() {
        return InfinityRun.clearRect(0, 0, InfinityRun.width,
            InfinityRun.height);
660   };
    //-------------------------------------------------
662
    Array.max = function( array ){
664     return Math.max.apply( Math, array );
    };
666
    InfinityRun.update = function() {
668         if (GameState == State.Started) {
            //-----------------------------------------
670         //clear func bg
            var i, results;
672     dt = InfinityRun.dt < .1 ? .1 : InfinityRun.dt / 16;
        dt = dt > 5 ? 5 : dt;
```

```
674     i = Town.length;
        results = [];
676     while (i--) {
          results.push(Town[i].update(i));
678     }
        //return results;
680         //-----------------------------------------------------------

682         dateNow = new Date();
            timePassed = (dateNow - date) / 10;
684
        this.player.update();
686     restartAudio();
        switch (this.jumpCount) {
688                 case 0:
                            bgaudio.pause();
690                         bgaudio.src = 'sounds/main1.wav';
                            bgaudio.load();
692                         bgaudio.play();
                break;
694         case 10:
                this.accelerationTweening = 1.3;
696             this.platformManager.maxDistanceBetween =
                   430;
                //this.scoreColor = '#076C00';
698                         bgaudio.pause();
                            bgaudio.src = 'sounds/main2.wav';
700                         bgaudio.load();
                            bgaudio.play();
702                         fxaudio.pause();
                            fxaudio.src = 'sounds/levelup.wav
                               ';
704                         fxaudio.load();
                            fxaudio.play();
706             break;
        case 25:
708             this.accelerationTweening = 2.3;
                this.platformManager.maxDistanceBetween =
                   530;
710             //this.scoreColor = '#0300A9';
                            bgaudio.pause();
712                         bgaudio.src = 'sounds/main3.wav';
                            bgaudio.load();
714                         bgaudio.play();
                            fxaudio.pause();
716                         fxaudio.src = 'sounds/levelup.wav
                               ';
                            fxaudio.load();
```

```
718                         fxaudio.play();
                    break;
720             case 40:
                    this.accelerationTweening = 3.5;
722                 this.platformManager.maxDistanceBetween =
                        580;
                    //this.scoreColor = '#9F8F00';
724                         bgaudio.pause();
                            bgaudio.src = 'sounds/main4.wav';
726                         bgaudio.load();
                            bgaudio.play();
728                         fxaudio.pause();
                            fxaudio.src = 'sounds/levelup.wav
                                ';
730                         fxaudio.load();
                            fxaudio.play();
732                 break;
            }
734
            this.acceleration += (this.accelerationTweening -
                this.acceleration) * 0.01;
736


738
            for (i = 0; i < this.platformManager.platforms.length
                ; i++) {
740              if (this.player.intersects(this.platformManager.
                    platforms[i])) {
                    this.collidedPlatform = this.platformManager.
                        platforms[i];
742                 if (this.player.y < this.platformManager.
                        platforms[i].y) {
                        this.player.y = this.platformManager.
                            platforms[i].y;
744
                        // Gravity after Collision with Platform
746                     this.player.velocityY = 0;
                    }
748
                    this.player.x = this.player.previousX;
750                 this.player.y = this.player.previousY;

752                 this.particles[(this.particlesIndex++) % this.
                        particlesMax] = new Particle({
                        x: this.player.x,
754                     y: this.player.y + this.player.height,
                        color: this.collidedPlatform.color
756                 });
```

```
758             if (this.player.intersectsLeft(this.
                    platformManager.platforms[i])) {
                    this.player.x = this.collidedPlatform.x -
                        64;
760                 for (i = 0; i < 10; i++) {
                        // SpawnParticles @PlayerPostion with
                            intersecting Platform Color
762                     this.particles[(this.particlesIndex
                            ++) % this.particlesMax] = new
                            Particle({
                            x: this.player.x + this.player.
                                width,
764                         y: random(this.player.y, this.
                                player.y + this.player.height)
                                ,
                            velocityY: random(-30, 30),
766                         color: randomChoice(['#181818',
                                '#181818', this.
                                collidedPlatform.color])
                        });
768                 };

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


774


776                                 if (timePassed > this.
                                        jumpCountRecord) {
778                     this.jumpCountRecord = timePassed
                            ;
                    }
780


782


784             } else {

786                 // ---------- Controller ----------
                    // dragging: Mouse click & touch support
788                 if (this.dragging || this.keys.SPACE ||
                        this.keys.UP || this.keys.W) {
                        this.player.velocityY = this.player.
```

```
                        jumpSize;
790                     this.jumpCount++;
                                            //play jump_sound
792                                         fxaudio.pause();
                                            fxaudio.src = '
                                               sounds/jump.
                                               wav';
794                                         fxaudio.load();
                                            fxaudio.play();
796


798


800                 }
                }
802         }
        };
804
        for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
806         this.platformManager.update();
        };
808
        for (i = 0; i < this.particles.length; i++) {
810         this.particles[i].update();
        };
812         //------------------------------------------------
            //bg
814         return results;
            //------------------------------------------------
816 }

818 };


820


822
    var selectedItem = 0;
824
    InfinityRun.keydown = function() {
826     if (InfinityRun.keys.ESCAPE && GameState==State.
           Started) {
                    InfinityRun.clear();
828                 GameState = State.Menu;
                    bgaudio.pause();
830             bgaudio.src = 'sounds/menu.wav';
                bgaudio.load();
832                 bgaudio.play();
```

```
834         } else if (InfinityRun.keys.ESCAPE && GameState==
                State.Menu && curMenuTab==MenuTab.Main) {
                    GameState = State.Started;
836         } else if (InfinityRun.keys.ESCAPE && GameState==
                State.Menu && curMenuTab==MenuTab.Settings) {
                    curMenuTab = MenuTab.Main;
838         } else if (InfinityRun.keys.ESCAPE && GameState==
                State.Menu && curMenuTab==MenuTab.Highscore) {
                    curMenuTab = MenuTab.Main;
840         }

842         //main menu controls
            if (InfinityRun.keys.UP) {
844                 selectedItem = (selectedItem + items.
                        length - 1) % items.length;
            }
846         if (InfinityRun.keys.DOWN) {
                    selectedItem = (selectedItem + 1) % items
                        .length;
848         }

850         // settings audio change
            if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.
                Settings && selectedItem !=0) {
852                 selectedItem = (selectedItem + items.
                        length - 1) % items.length;
            }
854
            if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab
                .Settings && selectedItem !=9) {
856                 selectedItem = (selectedItem + 1) % items
                        .length;
            }
858
            if(InfinityRun.keys.ENTER) {
860                 callback(selectedItem);
            }
862
    }
864
    Menu = function() {
866
            //this.backgroundCallback = null;
868 }

870
    //---------- Draw ----------
```

```
872
InfinityRun.draw = function() {
874
        if(GameState == State.Started) {
876     //------------------------------------------------------------------------

        //bg draw
878     var i, results;
    i = Town.length;
880 results = [];
    while (i--) {
882   results.push(Town[i].render(i));
    }
884     //------------------------------------------------------------------------

    this.player.draw();
886
    for (i = 0; i < this.platformManager.platforms.length
       ; i++) {
888     this.platformManager.platforms[i].draw();
    };
890
    //Draw particles
892 for (i = 0; i < this.particles.length; i++) {
        this.particles[i].draw();
894 };

896     /*
         * Main Menu
898      *
         */
900     } else if (GameState == State.Menu && curMenuTab
          ==MenuTab.Main) {

902     this.title = "InfinityRun";
        items = ["Play", "Settings", "Highscore", "
          Credits"];
904
        callback = function(numItem) { //if (numItem ==
          0) GameState=State.Started
906
        switch (numItem) {
908       case 0:
              GameState=State.Started;
910           break;
          case 1:
912           curMenuTab=MenuTab.Settings;
              break;
```

```
914             case 2:
                  curMenuTab=MenuTab.Highscore;
916                   break;
                case 3:
918               curMenuTab=MenuTab.Credits;
                      break;
920
            }
922


924
            };
926         this.height = InfinityRun.height;
            this.width = InfinityRun.width;
928         this.size = 120;

930         var lingrad = this.createLinearGradient(0,0,0,
               this.height);
            lingrad.addColorStop(0, '#000');
932         lingrad.addColorStop(1, '#023');
            this.fillStyle = lingrad;
934         this.fillRect(0,0,this.width, this.height)

936         this.textAlign = "center";
            this.fillStyle = "White";
938
            var height = 150;
940         //logo
            this.drawImage(bglogo,this.width-500,this.height
               -300);
942         //-----------------------------------
            if (this.title) {
944               this.font = Math.floor(this.size*1.3).
                     toString() + "px Times New Roman";
                  this.fillText(this.title, this.width/2,
                     height);
946               height+= height;
            }
948
            for (var i = 0; i < items.length; ++i)
950         {
                  var size = Math.floor(this.size*0.8);
952               if (i == selectedItem)
                  {
954                     this.fillStyle = "#A9F5F2";
                        size = this.size+5;
956               }
                  this.font = size.toString() + "px Times
```

```
                New Roman";
958             height += this.size;
                this.fillText(items[i], InfinityRun.width
                    /2, height);
960             this.fillStyle = "White";
            }
962         //-------------------------------------------------------------------

            //bg dd <-- ??
964         return results;
            //-------------------------------------------------------------------

966

            /*
968          * Settings Tab
             *
970          */
            } else if (GameState == State.Menu && curMenuTab
                ==MenuTab.Settings){
972

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

976


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

980         switch (volume) {

982         }


984


986         };


988


990         this.height = InfinityRun.height;
            this.width = InfinityRun.width;
992

            var lingrad = this.createLinearGradient(0,0,0,
                this.height);
994         lingrad.addColorStop(0, '#000');
            lingrad.addColorStop(1, '#023');
996         this.fillStyle = lingrad;
            this.fillRect(0,0,this.width, this.height, items[
```

```
            i]);
998
        this.textAlign = "center";
1000    this.fillStyle = "White";

1002    var width = 10;
        var height = 10;
1004    var posx = 130;
        var posy = 380;
1006    this.space = 15;
        this.heightincr = 4;
1008


1010
        if (this.title) {
1012            this.font = Math.floor(this.size*1.3).
                   toString() + "px Times New Roman";
                this.fillText(this.title, this.width/2,
                   150);
1014            height+= height;
        }
1016
        this.font = "70px Times New Roman";
1018    //this.fillText('Volume', InfinityRun.Left,
           InfinityRun.Top);


1020
        this.fillText('Volume', 240, 300);
1022



1024


1026    for (var i = 0; i < items.length; ++i) {
            var size = Math.floor(this.size*0.8);
1028        if (i == selectedItem)
            {
1030                this.fillStyle = "#A9F5F2";
                    size = this.size+5;
1032        }
            this.font = size.toString() + "px Times
               New Roman";
1034        posx += this.space;
            posy -= this.heightincr;
1036        height += this.heightincr;

1038        items[i] = this.fillRect(posx,posy,width,
               height);
```

```
1040                //this.fillText(items[i], InfinityRun.width/2, height);
                    this.fillStyle = "White";
1042
            }
1044
            /*
1046         * Highscore Tab
             *
1048         */

1050        } else if (GameState == State.Menu && curMenuTab == MenuTab.Highscore) {


1052
            this.title = "Highscore";
1054        items = highScore;


1056
            callback = function(volume) { //if (numItem == 0) GameState=State.Started
1058
            switch (volume) {
1060
            }
1062



1064
            };
1066        this.height = InfinityRun.height;
            this.width = InfinityRun.width;
1068
            var lingrad = this.createLinearGradient(0,0,0, this.height);
1070        lingrad.addColorStop(0, '#000');
            lingrad.addColorStop(1, '#023');
1072        this.fillStyle = lingrad;
            this.fillRect(0,0,this.width, this.height, items[i]);
1074
            this.textAlign = "center";
1076        this.fillStyle = "White";

1078        var width = 10;
            var height = 150;
1080

1082        if (this.title) {
```

```
                this.font = Math.floor(this.size*1.3).toString() + "px Times New Roman";
1084            this.fillText(this.title, this.width/2, 150);
                height+= height;
1086        }

1088        var rank = 1;

1090        for (var i = 0; i < items.length; ++i)
            {
1092
                var size = Math.floor(this.size*0.8);
1094            if (i == selectedItem)
                {
1096                this.fillStyle = "#A9F5F2";
                    size = this.size+5;
1098            }
                this.font = 0.6*size.toString() + "px Times New Roman";
1100            height += 50;
                this.fillText(rank + ". " + items[i], InfinityRun.width/2, height);
1102            this.fillStyle = "White";

1104            rank++;
            }
1106
            }
1108


1110
    //Debug
1112    if (debug) {
        this.font = '12pt Arial';
1114    this.fillStyle = '#181818';
        this.fillText('Record: ' + highScore[0]/*this.jumpCountRecord*/, this.width - 150, 33);
1116    this.fillStyle = this.scoreColor;
        this.fillText('Jumps: ' + this.jumpCount, this.width - 150, 50);
1118    this.fillText('Distance: ' + Math.round(timePassed) , this.width - 150, 65);
                this.fillText('mouse: ' + this.mouse.y , this.width - 150, 100);
1120    this.fillText('GameState: ' + GameState, this.width - 150, 80);
    }
```

```
1122
};
1124
InfinityRun.resize = function() {
1126    /* todo Windowscale optimization
        *
1128    *
        */
1130 }
```

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>
```