



HOCHSCHULE
FURTWANGEN
UNIVERSITY



Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent : **Gabriela Mai**

Vorgelegt am : 16. Dezember 2016

Vorgelegt von : Gruppe 4

Florian Durli : 254791

Jannik Ivosevic : 255028

Johannes But : 254053

Marco Mayer : 254795

Koray Emtekin : 254816

Inhaltsverzeichnis

Inhaltsverzeichnis	ii
Abbildungsverzeichnis	iii
Tabellenverzeichnis	v
1 Einleitung	1
1.1 Team	1
1.2 Rollenverteilung	2
1.3 Spielidee	2
1.3.1 Spielkonzept	2
1.3.2 Entwurfsskizze	3
1.3.3 Erforderliche Software	4
2 Phasen	5
2.1 Entwurf und Anforderungen	5
2.1.1 Funktionale Anforderungen	5
2.1.2 Nicht funktionale Anforderungen	6
2.1.3 Projektplan	6
2.1.4 Releaseplan	7
2.2 Implementation - Zwischenstand	8
2.2.1 Erfüllte Anforderungen	8
2.2.2 Nicht erfüllte Anforderungen	8
2.2.3 Das Spiel	9

2.2.4	Bibliothek	10
2.2.5	Code	10
2.2.6	Nächste Ziele	16
2.3	Implementation - Endstand	16
2.3.1	Spielkonzept Änderungen	16
2.3.2	Funktionsdiagramm	16
2.3.3	Grafiken	18
2.3.4	Code Änderungen	19
2.3.5	Das Spiel - Endstand	23
2.3.6	Sounds	25
	Literaturverzeichnis	27
	Eidesstattliche Erklärung	29
A	Anhang	31
A.1	Github Changelog	31
A.2	game.js	34
A.3	game.css	66
A.4	index.html	68

Abbildungsverzeichnis

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

Tabellenverzeichnis

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

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

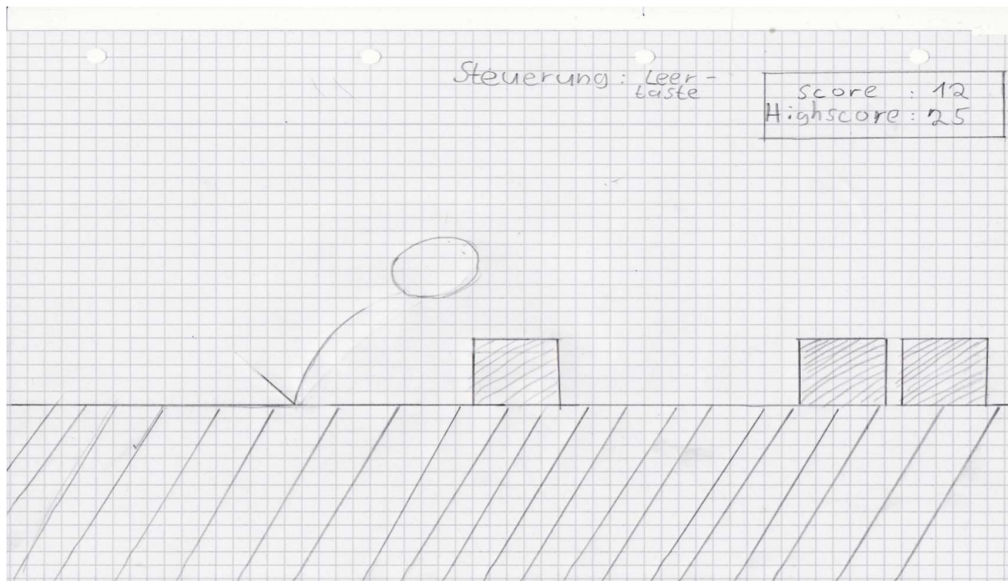


Abbildung 6.: Entwurfsskizze

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 Performarncce 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.

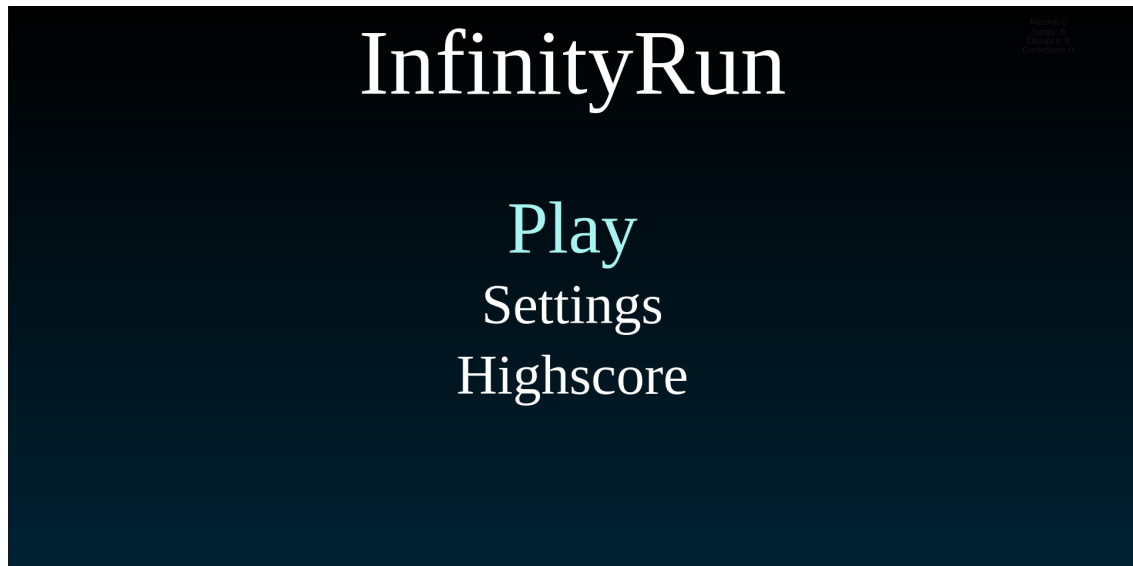


Abbildung 7.: Startbildschirm



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:

```
1 function start()
  {
3     context.now = +new Date();
    context.running = true;
5  }

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

11 function toggle()
13 {
    ( context.running ? stop : start )();
15 }

17 function clear()
  {
19     if ( is2D )
        context.clearRect( 0, 0, context.width, context.height );
21 }
```

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.

```
1 var InfinityRun = Sketch.create({
  fullscreen: true,
3 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
6     this.maxDistanceBetween = 300;
8
10    this.colors = [ '#2ca8c2 ', '#98cb4a ', '#f76d3c ', '
        #f15f74 ', '#5481e6 ' ];
12
14    //first 3 Platforms execept the Starter Platform
16    this.first = new Platform({
18        x: 300,
        y: InfinityRun.width / 2,
        width: 400,
        height: 70
    })
20    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,

```

```

22         height: 70
23     })
24
25     this.third = new Platform
26     ({
27         x: (this.second.x + this.second.width) + random(▷
28             this.maxDistanceBetween - 150, this.▷
29             maxDistanceBetween),
30         y: random(this.second.y - 128, InfinityRun.height▷
31             - 80),
32         width: 400,
33         height: 70
34     })
35
36     this.first.height = this.first.y + InfinityRun.▷
37         height;
38     this.second.height = this.second.y + InfinityRun.▷
39         height;
40     this.third.height = this.third.y + InfinityRun.▷
41         height;
42     this.first.color = randomChoice(this.colors);
43     this.second.color = randomChoice(this.colors);
44     this.third.color = randomChoice(this.colors);
45     this.colliding = false;
46     this.platforms = [this.first, this.second, this.▷
47         third];
48 }

```

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 {
3     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);
28     this.third.y = random(this.second.y - 32,
        InfinityRun.height - 80);
    this.third.height = this.third.y +
        InfinityRun.height + 10;
30     this.third.color = randomChoice(this.
        colors);
    }
32 };

```

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 Plattformen annehmen.

```

1  for (i = 0; i < this.platformManager.platforms.length;
2      ; i++)
3  {
4      if (this.player.intersects(this.platformManager.platforms[i]))
5      {
6          this.collidedPlatform = this.platformManager.platforms[i];
7          if (this.player.y < this.platformManager.platforms[i].y)
8          {
9              this.player.y = this.platformManager.platforms[i].y;
10             // 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 Änderungen haben wir im laufe der Implementation vorgenommen:

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

2.3.2. Funktionsdiagramm

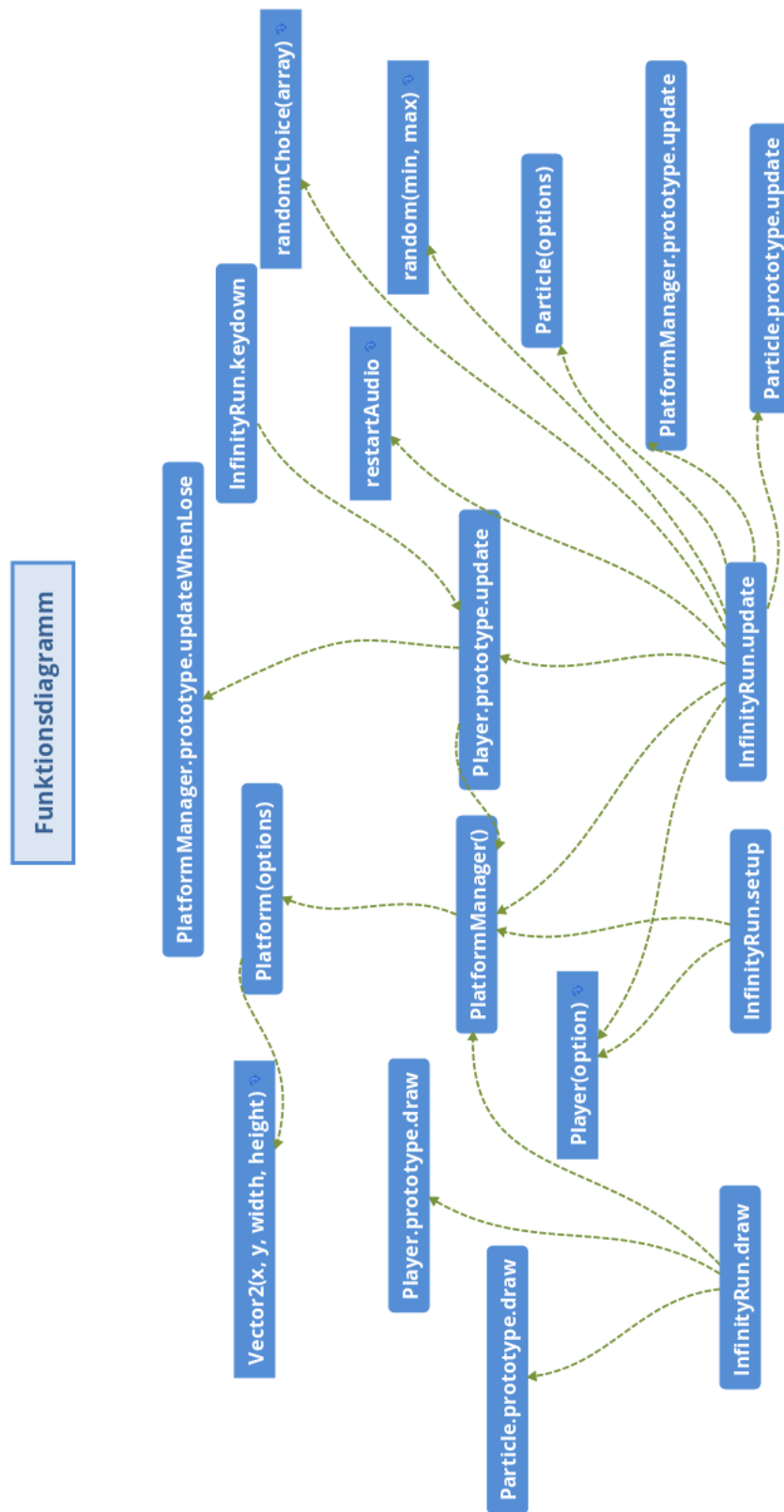


Abbildung 9.: Funktionsdiagramm

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');  
2 var fxaudio = document.getElementById('fxaudio');
```

Zugriff auf Element per Funktion zum Neustarten der Musik:

```
function restartAudio()  
2 {  
    // Check for audio element support.  
    4 if (window.HTMLAudioElement)  
    {  
        6 try  
        {  
            8 // Tests the paused attribute and  
            // set state.  
            10 if (bgaudio.ended)  
            {  
                bgaudio.currentTime = 0;  
                12 bgaudio.play();  
            }  
            14 }  
            16 catch (e)  
            {  
                18 // Fail silently but show in F12  
                // developer tools console  
                20 if(window.console && console.  
                error("Error:" + e));  
            }  
        }  
    }  
}
```

Beispiel für die Audio Wiedergabe:

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

```
{  
3     this.player.velocityY = this.player.jumpSize;  
     this.jumpCount++;  
5     fxaudio.pause();  
     fxaudio.src = 'sounds/jump.wav';  
7     fxaudio.load();  
     fxaudio.play();  
9 }
```

2.3.4.2. Hintergrund

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

```

1 Street.prototype.populate = function()
  {
3     var newHeight, newWidth, results, totalWidth;
      totalWidth = 0;
5     results = [];
      while (totalWidth <= InfinityRun.width + (this.
        width.max * 2))
7     {
          newWidth = round(random(this.width.min,
            this.width.max));
9         newHeight = round(random(this.height.min,
            this.height.max));
        this.alltowers.push(new Tower({
11             layer: this.layer,
            x: this.alltowers.length == 0 ?
              0 : this.alltowers[this.
                alltowers.length - 1].x + this.
                  alltowers[this.alltowers.
                    length - 1].width,
13             y: InfinityRun.height - newHeight
              ,
            width: newWidth,
15             height: newHeight,
            color: this.color
17         }));
          results.push(totalWidth += newWidth);
19     }
      return results;
21 };

```

Aktualisieren der Hochhäuser, für neues erscheinen am rechten Spielrand:

```

1 Street.prototype.update = function()
  {

```

```
3      var firstTower, lastTower, newHeight, newWidth;
4      if (InfinityRun.accelerationTweening==0)
5      {
6          this.x -= ((150) * this.speed) * dt;
7      }
8      else
9      {
10         this.x -= ((InfinityRun.▷
11             accelerationTweening*330) * this.speed▷
12             ) * dt;
13     }
14
15     firstTower = this.alltowers[0];
16     if (firstTower.width + firstTower.x + this.x < 0)
17     {
18         newWidth = round(random(this.width.min, ▷
19             this.width.max));
20         newHeight = round(random(this.height.min, ▷
21             this.height.max));
22         lastTower = this.alltowers[this.alltowers▷
23             .length - 1];
24         firstTower.reset({
25             layer: this.layer,
26             x: lastTower.x + lastTower.width,
27             y: InfinityRun.height - newHeight▷
28             ,
29             width: newWidth,
30             height: newHeight,
31             color: this.color
32         });
33     }
34     return this.alltowers.push(this.alltowers.shift()▷
35         );
36 }
37
38 };
```

2.3.5. Das Spiel - Endstand

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

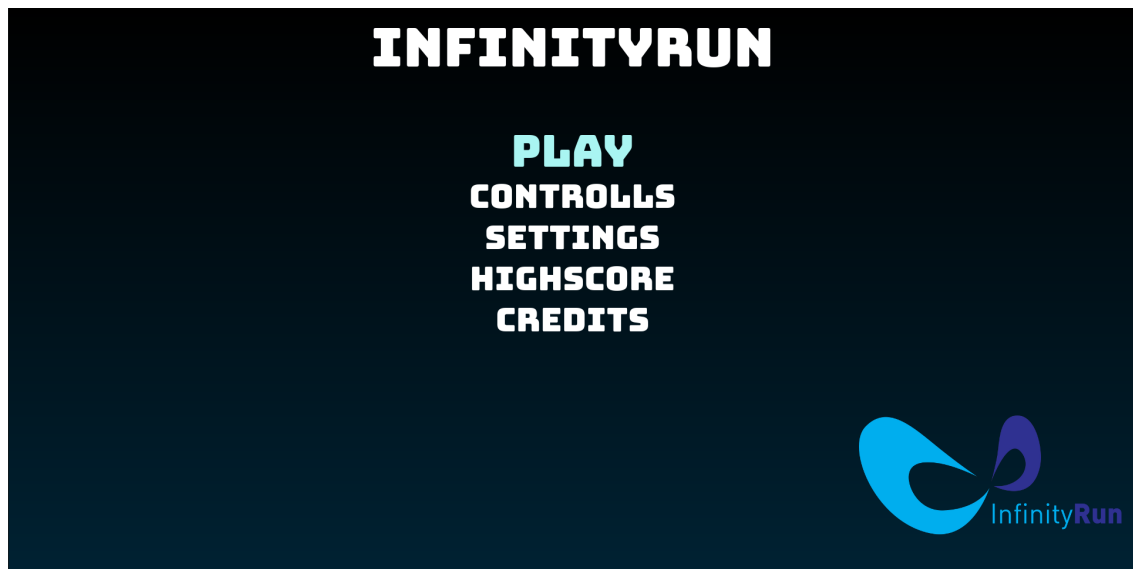


Abbildung 11.: Startbildschirm - Endstand

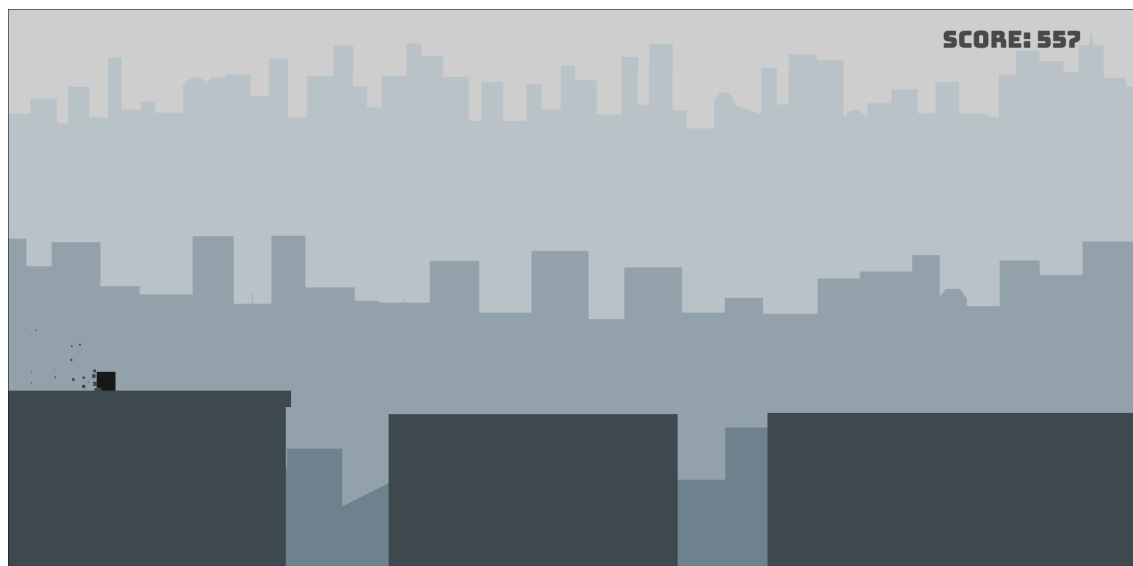


Abbildung 12.: Das Spiel - Endstand

2.3.5.1. Steuerung

Menü-Steuerung:

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

Tabelle 8.: Menü-Steuerung

Spiel-Steuerung:

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

Tabelle 9.: Spiel-Steuerung

2.3.5.2. Webserver und Smartphone

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

Spiel: <http://infinityrun.freevar.com/>

2.3.6. Sounds

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

Folgende Sounds werden wir verwenden:

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

Tabelle 10.: Sound Links

Literaturverzeichnis

- [Are] AREA, Free Web H.: *Free Hosting* <http://freevar.com/>
- [dis] DISSIMULATE: *Skyline* <https://codepen.io/dissimulate/pen/CAzlt>
- [Fre] FREESOUND: *Freesound.org* <https://www.freesound.org/>
- [Git] GITHUB: *Softwareverwaltung* <https://github.com/>
- [Goo] GOOGLE: *Google Chrome* <https://www.google.com/chrome/>
- [Gru] GRUPPE4: *Changelog* <https://github.com/Slay3r/InfinityRun/commits/master>
- [Ho] HO, Don: *Notepad++* <https://notepad-plus-plus.org/>
- [sou] SOULWIRE: *Sketch Bibliothek* <https://github.com/soulwire/sketch.js>
- [Tea] TEAM, The G.: *Bildbearbeitungssoftware* <https://www.gimp.org/>

Eidesstattliche Erklärung

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

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

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

FURTWANGEN, den 16. Dezember 2016 Florian Durli

FURTWANGEN, den 16. Dezember 2016 Jannik Ivosevic

FURTWANGEN, den 16. Dezember 2016 Johannes But

FURTWANGEN, den 16. Dezember 2016 Marco Mayer

FURTWANGEN, den 16. 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.

```
1 $ 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 11.: Github Namen

Changelog:

```
1 c4876c7 Florian Durli 2016-12-16 Controlls
3 9ce6d99 Florian Durli 2016-12-16 Doku Server ,Steuerung
5 e1c7138 r4qtor          2016-12-15 acceleration fix
7 188bcd b r4qtor          2016-12-15 Delete sketch.minlesbar.
   js (RE: unbenutzt)
9 55b3162 r4qtor          2016-12-15 bug fixes , 'Farbänderung
   ', kleines 'Hindernis '
11 0f5b9cc butjo          2016-12-15 HighScore gefixed
   Plattformen erhöht
13 414cbc9 butjo          2016-12-14 Creditsverbesserungen
15 fc96b76 butjo          2016-12-14 Credits eingefügt
17 3b990da butjo          2016-12-14 Audioeinstellungen
   implementiert
```

19	6fb84f9	Slay3r	2016-12-14	Doku Quellen angepasst
21	02cecd1	butjo	2016-12-14	Merge branch 'master' of https://github.com/Slay3r/InfinityRun
23	682c0d2	butjo	2016-12-14	Farbschema geändert
25	449cf21	Slay3r	2016-12-14	Doku
27	288bb25	butjo	2016-12-14	Merge
29	2003a79	butjo	2016-12-13	Dacharten erweitert
31	42e9a8f	butjo	2016-12-13	Code verbessert / Optiemiert
33	67cc215	Slay3r	2016-12-13	Doku logo
35	2d7d953	butjo	2016-12-12	Favicon und Logo im Menü
37	2b8a139	ans77	2016-12-12	Logos hinzugefügt
39	92b7b28	Slay3r	2016-12-12	Code Cleanup
41	30b0c6f	Slay3r	2016-12-12	Clean up
43	37f6e94	butjo	2016-12-12	Favicon hinzugefügt
45	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
47	8d06d8a	butjo	2016-12-09	skyline eingefügt
49	4e52883	Slay3r	2016-12-07	Doku + Alte Dateien
51	2c3491b	butjo	2016-12-05	Sounds hinzugefügt teilweise noch ein paar Bugs und dirty code
53	2e6dfbe	butjo	2016-11-30	Dirty Code mit versuch den background zu implementieren files sind mit prefix back gekennzeichnet und liegen im Hauptverzeichnis.
55	dda171b	Slay3r	2016-11-30	Changelog geändert
57	ffe660b	Slay3r	2016-11-30	Ausblick/Fazit

auskommentiert

59	01e309a	Slay3r	2016-11-30	Sounds	
61	3a05368	Slay3r hinzufügen	2016-11-30	Neue Dokumentation	↗
63	7c3596c	Slay3r hinzufügen	2016-11-30	Neue Dokumentation	↗
65	bfdb05c	Slay3r Dokumentation	2016-11-30	entfernen der	↗
67	f944707	r4qtor	2016-11-25	Querlesung – Marco	
69	8b6bdde	Florian Durli	2016-11-25	Abgabe	
71	bc8d933	Florian Durli	2016-11-25	Code Cleanup für Doku	
73	b6d7a09	butjo	2016-11-24	Rechtschreibkorrekturen	
75	1faa558	r4qtor	2016-11-24	Delete phasen.tex root/	
77	51f4a79	r4qtor	2016-11-24	updated phasen.tex	
79	62af4b8	Krusher999 geschrieben	2016-11-24	Letzten Codes	↗
81	93b8965	Florian Durli	2016-11-23	fix	
83	9027301	Florian Durli	2016-11-23	Changelog finale Lösung	
85	1392138	Florian Durli	2016-11-23	Jojos Description in LaTeX	↗
87	25ff037	butjo	2016-11-23	Beschreibung der Codeteile in der phasen.tex von Johannes	↗
89	1b2348c	Florian Durli	2016-11-23	Changelog Additionally	
91	cf467dc	Florian Durli	2016-11-23	Changelog Additionally	
93	6db1ad6	butjo	2016-11-23	Merge branch 'master' of https://github.com/Slay3r/InfinityRun	↗
95	b924713	Slay3r	2016-11-23	Generated Changelog	
97	69dd747	butjo	2016-11-23	Merge branch 'master' of	↗

<https://github.com/Slay3r/InfinityRun>

99	203ae2e	Slay3r	2016-11-23	Bilder des Spiels
101	d274cfc	Slay3r	2016-11-23	Anforderungen
103	a0909ac	Slay3r	2016-11-23	Literaturverzeichnis
105	19e2f3d	Slay3r	2016-11-23	Doku update
107	21889f9	Florian Durli	2016-11-22	Add Changelog
109	743c95a	butjo	2016-11-16	Formatierte sketch.min.js ↗
111	d64d254	butjo	2016-11-16	Endlose Schwierigkeitserhöhung ↗
113	3707b94	butjo	2016-11-16	Präsentation Zwischenstand ↗
115	f23c9be	Slay3r	2016-11-16	Präsentation überarbeitet ↗
117	53aa72e	butjo	2016-11-16	Präsentation und test
119	b5cb978	Florian Durli	2016-11-16	Merge pull request #1 from r4qtor/master ↗
121	275bd69	r4qtor	2016-11-15	tiny cleanup & incl. menu ↗
123	11a5e55	Slay3r	2016-11-09	Basis implementation
125	c783850	Slay3r	2016-11-09	Bilder hinzugefügt
127	d88d0d2	Slay3r	2016-11-09	Dokumentations Basis
129	39b4705	Florian	2016-10-19	Initial Struktur der Ordner und files ↗
131	093797e	Florian Durli	2016-10-19	Delete Requirements
133	56c4aae	Florian	2016-10-19	doc Ordner
135	b3b83fd	Florian	2016-10-19	Requirements hinzugefügt
137	b4d2627	Florian Durli	2016-10-19	Initial commit

A.2. game.js

```

/* todo: cleanup (dirty code),
2  *
  * Put static values / vars into initialization function
4  *
  * _____
6  * Design / Graphics
  *
8  * Parallax Background?
  *
10 * _____
  * Menu
12 * Menu draw in Input & draw prototypes
  * Handle / Manage CSS or HTML variables from JavaScript ↵
  (Fullscreen , ...)
14 * _____
  *
16 * Platform Schematic? – Schematic files?
  * Different Themes depending on Progress?
18 *
  * _____
20 * Test-Phase
  *
22 * Controller: 'dragging' test Touch support
  * Browsertesting tools
24 * eg.:
  * http://browserling.com/
26 * http://browsershots.org/
  * https://crossbrowsertesting.com/
28 * https://www.browserstack.com/
  */
30 //testweise rausgenommen verändert nix
  //var i = 0;
32
  var debug = false;
34
  var State = { Menu:0, Controlls:1, Started:2, Paused:3, ↵
    Over:4 };
36 var GameState = State.Menu;
  var MainMenu;
38 var MenuTab = {Main:0, Controlls:1, Settings:2, Highscore ↵
    :3, Credits:4};
  var curMenuTab = MenuTab.Main;
40
  //timer
42 var s = 0,
  ms = 0,

```

```

44 playTimer = false;

46 var highScore = new Array(10);
  highScore = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
48

50 var bgaudio = document.getElementById('backgroundmusic');
  var fxaudio = document.getElementById('fxaudio');
52 var backgroundaudio = "sounds/main1.wav";

54
  //----->

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

60 Town = [];

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

68 function restartAudio()
{
70     // Check for audio element support.
    if (window.HTMLAudioElement)
72     {
        try
74         {
            // Tests the paused attribute and
            // set state.
76             if (bgaudio.ended)
            {
78                 bgaudio.currentTime = 0;
                bgaudio.play();
80             }
        }
        catch (e)
82        {
            // Fail silently but show in F12
            // developer tools console
84            if(window.console && console.log)
                error("Error: " + e);

86        }
    }
}

```

```

    }
88 }

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

94 function randomChoice(array) {
96     return array[Math.round(random(0, array.length - 1))]
    };
    }
98

100 //initialize Sketch Framework
    var InfinityRun = Sketch.create({
102     fullscreen: true,
        width: 640,
104     height: 360,
        container: document.getElementById('container')
106 });
    var qs = document.querySelector('canvas');
108
    //----->

110 //bg func
112 Tower = function(config)
    {
114     return this.reset(config);
    };
116
    Tower.prototype.reset = function(config)
118 {
        this.layer = config.layer;
120     this.x = config.x;
        this.y = config.y;
122     this.width = config.width;
        this.height = config.height;
124     this.color = config.color;
        this.summitTop = floor(random(0, 15)) ==>
            0;
126     this.summitTopWidth = random(this.width * .01,
        this.width * .07);
        this.summitTopHeight = random(10, 20);
128     this.singleroofTop = floor(random(0, 10)) == 0;
        this.singleroofTopHeight = this.width / random(2,

```

```

    4);
130   this.singleroofTopDirection = round(random(0, 1)) >
      == 0;
      this.normalTop = !this.singleroofTop && >
        floor(random(0, 10)) == 0;
132   this.normalTopHeight = this.width / random(2, 4);
      this.normalTopchimney = round(random(0, 1)) == >
        0;
134       this.coneTop = !this.singleroofTop && ! >
        this.normalTop && floor(random(0, 10)) >
          == 0;
      this.coneTopHeight = this.width / random(3, 4);
136       this.coneTopWidth = this.width / random >
        (1, 2);
      this.coneTopeflat = round(random(0, 1)) == 0;
138       this.companyTop = !this.singleroofTop && >
        !this.summitTop && !this.radioTop && ! >
        this.normalTop && floor(random(0, 10)) >
          == 0;
      this.companyTopHeight = this.width / random(4, 6) >
        ;
140   this.companyTopcount = 4;//round(random(3, 6));
      this.radioTop = !this.summitTop && floor(random >
        (0, 10)) == 0;
142   this.radioTopWidth = this.layer / 2;
      return this.radioTopHeight = random(6, 30);
144 };

146 Tower.prototype.render = function()
    {
148     InfinityRun.fillStyle = InfinityRun.strokeStyle = >
        this.color;
      InfinityRun.lineWidth = 2;
150     InfinityRun.beginPath();
      InfinityRun.rect(this.x, this.y, this.width, this >
        .height);
152     InfinityRun.fill();
      InfinityRun.stroke();
154     if (this.singleroofTop)
        {
156         InfinityRun.beginPath();
          InfinityRun.moveTo(this.x, this.y);
158         InfinityRun.lineTo(this.x + this.width, >
          this.y);
          if (this.singleroofTopDirection)
160         {
            InfinityRun.lineTo(this.x + this.>
              width, this.y - this.>

```

```

                                singleroofTopHeight);
162         }
        else
164         {
                                InfinityRun.lineTo(this.x, this.y -
                                this.singleroofTopHeight);
166         }
        InfinityRun.closePath();
168         InfinityRun.fill();
        InfinityRun.stroke();
170     }

    if (this.normalTop)
    {
174         InfinityRun.beginPath();
        InfinityRun.moveTo(this.x, this.y);
176         InfinityRun.lineTo(this.x + this.width,
                                this.y);
                                InfinityRun.lineTo(this.x + (this
                                .width/2), this.y - this.
                                normalTopHeight);
178         InfinityRun.closePath();
        InfinityRun.fill();
180         InfinityRun.stroke();
        if (this.normalTopchimney)
182         {
                                InfinityRun.beginPath();
184                                InfinityRun.moveTo(this.x +
                                +(this.width/5), this.y);
                                InfinityRun.lineTo(this.x +
                                +(this.width/5), this.y -
                                0.8*(this.
                                normalTopHeight));
186                                InfinityRun.lineTo(this.x +
                                + (this.width/5) + (
                                this.width/10), this.y -
                                0.8*(this.
                                normalTopHeight));
                                InfinityRun.lineTo(this.x +
                                + (this.width/5) + (
                                this.width/10), this.y
                                );
188                                InfinityRun.closePath();
                                InfinityRun.fill();
190                                InfinityRun.stroke();
                                }
192     }

```

```

194         if (this.coneTop)
195         {
196             InfinityRun.beginPath();
197             InfinityRun.moveTo(this.x, this.y);
198             InfinityRun.lineTo(this.x + (this.width -
199                 this.coneTopWidth)/2, this.y - this.
200                 coneTopHeight);
201             if (!this.coneTopeflat)
202             {
203                 InfinityRun.lineTo(this.x +
204                     +(this.width/2), this.
205                     y - (this.coneTopHeight
206                         * 1.3));
207             }
208             InfinityRun.lineTo(this.x + ((
209                 this.width - this.coneTopWidth)
210                 / 2) + this.coneTopWidth, this.y -
211                 this.coneTopHeight);
212             InfinityRun.lineTo(this.x + this.
213                 width, this.y);
214             InfinityRun.closePath();
215             InfinityRun.fill();
216             InfinityRun.stroke();
217         }
218         if (this.companyTop)
219         {
220             var ctc = 1;
221             while (ctc <= this.companyTopcount)
222             {
223                 InfinityRun.beginPath();
224                 InfinityRun.moveTo(this.x, this.
225                     y);
226                 InfinityRun.lineTo(this.x + ctc * (
227                     this.width / this.
228                     companyTopcount), this.y - this.
229                     companyTopHeight);
230                 InfinityRun.lineTo(this.x + ctc * (
231                     this.width / this.
232                     companyTopcount), this.y + this.
233                     companyTopHeight);
234                 InfinityRun.closePath();
235                 InfinityRun.fill();
236                 InfinityRun.stroke();
237                 ctc++;
238             }
239         }
240         if (this.summitTop)

```



```

    {
226         InfinityRun.beginPath();
        InfinityRun.moveTo(this.x + (this.width / 2), this.y - this.summitTopHeight);
228         InfinityRun.lineTo(this.x + (this.width / 2) + this.summitTopWidth, this.y);
        InfinityRun.lineTo(this.x + (this.width / 2) - this.summitTopWidth, this.y);
230         InfinityRun.closePath();
        InfinityRun.fill();
232         InfinityRun.stroke();
    }

234     if (this.radioTop)
236     {
        InfinityRun.beginPath();
238         InfinityRun.moveTo(this.x + (this.width / 2), this.y - this.radioTopHeight);
        InfinityRun.lineTo(this.x + (this.width / 2), this.y);
240         InfinityRun.lineWidth = this.radioTopWidth;
        return InfinityRun.stroke();
242     }
    };

244     Street = function(config)
246     {
        this.x = 0;
248         this.alltowers = [];
        this.layer = config.layer;
250         this.width = {
            min: config.width.min,
252             max: config.width.max
        };
254         this.height = {
256             min: config.height.min,
            max: config.height.max
258         };

260         this.speed = config.speed;
        this.color = config.color;
262         this.populate();
        return this;
264     };

266     Street.prototype.populate = function()

```

```

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

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

300     firstTower = this.alltowers[0];
        if (firstTower.width + firstTower.x + this.x < 0)
302     {
            newWidth = round(random(this.width.min, ▷
                this.width.max));
304     newHeight = round(random(this.height.min, ▷
                this.height.max));
            lastTower = this.alltowers[this.alltowers▷

```

```

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

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

    //----- Vector [Get/Set] Functions -----
332
    //Set X,Y,Width,Height
334 function Vector2(x, y, width, height) {
        this.x = x;
336         this.y = y;
        this.width = width;
338         this.height = height;
        this.previousX = 0;
340         this.previousY = 0;
    };
342

344 // Set X,Y
    Vector2.prototype.setPosition = function(x, y) {
346

```

```

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

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

360    };

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

368    };

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

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

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

388        return false;
    };
390
    //----- Player -----

```

```

392 function Player(options) {
394     this.setPosition(options.x, options.y);
396     this.width = options.width;
398     this.height = options.height;
400     this.velocityX = 0;
402     this.velocityY = 0;
404     this.jumpSize = -13;
406     this.color = '#181818';
408 }
410
412 Player.prototype = new Vector2;
414
416 Player.prototype.update = function() {
418     // Gravity
420     this.velocityY += 1;
422     //um bg zu ändern
424     jumpheight=(this.y);
426     this.setPosition(this.x + this.velocityX, this.y +
428         this.velocityY);
430
432     if (this.y > InfinityRun.height || this.x + this.
434         width < 0) {
436         this.x = 150;
438         this.y = 50;
440         this.velocityX = 0;
442         this.velocityY = 0;
444         InfinityRun.jumpCount = 0;
446         InfinityRun.acceleration = 0;
448         InfinityRun.accelerationTweening = 0;
450         InfinityRun.scoreColor = '#181818';
452         InfinityRun.platformManager.maxDistanceBetween =
454             350;
456
458         //InfinityRun.pause();
460
462         //highscore update
464
466         if (timePassed>highScore[0]){
468             var help =highScore[0];
470             var help2=highScore[1];
472             highScore[0]=timePassed;
474             for(i=1; i<=9;i++){
476                 help2 = highScore[i];
478                 highScore[i]=help;

```

```

                                help=help2;
436
                                }
438
                                }
440        InfinityRun.platformManager.updateWhenLose();
                                fxaudio.pause();
442        fxaudio.src = 'sounds/crash.wav';
                                fxaudio.load();
444        fxaudio.play();

                                ms = 0;
446    }
448
    if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
        InfinityRun.keys.W || InfinityRun.dragging) &&
        this.velocityY < -8) {
450        this.velocityY += -0.75;

452    }

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

458 };
460
Player.prototype.draw = function() {
462    InfinityRun.fillStyle = this.color;
    InfinityRun.fillRect(this.x, this.y, this.width, this
        .height);
464 };

466 // ----- Platforms -----

468 function Platform(options) {
    this.x = options.x;
470    this.y = options.y;
    this.width = options.width;
472    this.height = options.height;
    this.previousX = 0;
474    this.previousY = 0;
    this.color = options.color;
476 }

478 Platform.prototype = new Vector2;

```

```
480 Platform.prototype.draw = function() {
    InfinityRun.fillStyle = this.color;
482    InfinityRun.fillRect(this.x, this.y, this.width, this
        .height);
    };
484 // ----- Platform Manager -----
486 function PlatformManager() {
488     this.maxDistanceBetween = 300;
        this.colors = ['#3D494F'];
490
        //first 3 Platforms execept the Starter Platform
492     this.first = new Platform({
        x: 300,
494         y: 600,
        width: 400,
496         height: 70
    })
498     this.second = new Platform({
        x: (this.first.x + this.first.width) + random(
            this.maxDistanceBetween - 150, this.
                maxDistanceBetween),
500         y: 570, //y: random(this.first.y - 128,
            InfinityRun.height - 80),
        width: 400,
502         height: 70
    })
504     this.third = new Platform({
        x: (this.second.x + this.second.width) + random(
            this.maxDistanceBetween - 150, this.
                maxDistanceBetween),
506         y: 540, //y: random(this.second.y - 128,
            InfinityRun.height - 80),
        width: 400,
508         height: 70
    })
510
        this.first.height = this.first.y + InfinityRun.height
        ;
512     this.second.height = this.second.y + InfinityRun.
        height;
        this.third.height = this.third.y + InfinityRun.height
        ;
514     this.first.color = randomChoice(this.colors);
        this.second.color = randomChoice(this.colors);
516     this.third.color = randomChoice(this.colors);
```

```

518     this.colliding = false;

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

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

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

544     this.third.x -= 3 + InfinityRun.acceleration;
546     if (this.third.x + this.third.width < 0) {
        this.third.width = random(450, 800);
548         this.third.x = (this.second.x + this.second.width) +
            random(this.maxDistanceBetween - 150, this.maxDistanceBetween);
        //this.first.y = random(this.third.y - 32,
            InfinityRun.height - 80);
550         this.third.y = random(this.second.y - 32,
            InfinityRun.height - 200);

```



```

        this.third.height = this.third.y + InfinityRun.▷
            height + 10;
552     this.third.color = randomChoice(this.colors);
        }
554     };
556
558     // reset / new Game: set Starting Platform Parameters
560     PlatformManager.prototype.updateWhenLose = function() {

562         this.first.x = 300;
        this.first.color = randomChoice(this.colors);
564         this.first.y = 500;
        //this.first.y = InfinityRun.width / random(2, 3);
566         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);

568     };
570     // ————— Particle System ————— (Sketch Docs)
572     function Particle(options) {
574         this.x = options.x;
        this.y = options.y;
576         this.size = 10;
        this.velocityX = options.velocityX || random(-(▷
            InfinityRun.acceleration * 3) + -8, -(InfinityRun.▷
                acceleration * 3));
578         this.velocityY = options.velocityY || random(-(▷
            InfinityRun.acceleration * 3) + -8, -(InfinityRun.▷
                acceleration * 3));
        this.color = options.color;
580     }

582     Particle.prototype.update = function() {
        this.x += this.velocityX;
584         this.y += this.velocityY;
        this.size *= 0.89;
586     };

588     Particle.prototype.draw = function() {
        InfinityRun.fillStyle = this.color;

```

```

590     InfinityRun.fillRect(this.x, this.y, this.size, this.size);
    };
592
    /*****
594     InfinityRun.setup = function() {
596         this.jumpCount = 0;
598         this.acceleration = 0;
        this.accelerationTweening = 0;
600         this.player = new Player({
            x: 150,
602             y: 30,
            width: 32,
604             height: 32
        });
606         bgaudio.pause();
        bgaudio.src = 'sounds/menu.wav';
608         bgaudio.load();
        bgaudio.play();
610
        this.platformManager = new PlatformManager();
612
        this.particles = [];
614         this.particlesIndex = 0;
        this.particlesMax = 20;
616         this.collidedPlatform = null;
        this.scoreColor = '#181818';
618         this.jumpCountRecord = 0;
        //-----
620         var i, results;
        i = 3;
622         results = [];
        while (i--) {
624             results.push(Town.push(new Street({
                layer: i + 1,
626                 width: {
                    min: (i + 1) * 20,
628                     max: (i + 1) * 50
                },
                height: {
630                     min: InfinityRun.height - 200 - (i * round(
                        InfinityRun.height/3)),
                    max: InfinityRun.height - 50 - (i * round(
632                         InfinityRun.height/3))
                },
                speed: (i + 1) * .003,
634

```

```

        color: 'hsl( 200, ' + (((i + 1) * 1) + 10) + '%, >
            ' + (75 - (i * 13)) + '% )'
636     }));
    }
638     return results;
        //_____

640

642 };
    //_____
644 InfinityRun.clear = function() {
        return InfinityRun.clearRect(0, 0, InfinityRun.width, >
            InfinityRun.height);
646 };
    //_____

648 Array.max = function( array ){
650     return Math.max.apply( Math, array );
    };
652
    var sc = 0;
654 var sx = 0;
    var sy = 0;
656 var sz = 0;
    var invertRunning = false;
658 var sunsetRunning = false;
    timer = setInterval(function() {
660     if (!playTimer) return;
        ms += 1;
662         sc += 1;
            sy += 1;
664             sz += 1;
                if (sc == 99) {
666                     s+=1;
                        sx+=1;
668                             sc = 0;
                }

670         updateTimer();

672     }, 1);
674
    function randomIntFromInterval(min,max)
676 {
        var milliseconds = new Date().getMilliseconds();
678     return Math.floor(Math.random()*(max-min+1)+min);
    }
680

```

```

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

710     timePassed = ms;
712 }

714 function toggleTimer() {
    if (!playTimer) {
716         //s = 0, ms = 0;

718         updateTimer();
    }
    playTimer = !playTimer;
720 }
722

724 InfinityRun.update = function() {
726     if (GameState == State.Started) {
        //_____
728         //clear func bg

```

```

    var i, results;
730 dt = InfinityRun.dt < .1 ? .1 : InfinityRun.dt / 16;
    dt = dt > 5 ? 5 : dt;
732 i = Town.length;
    results = [];
734 while (i--) {
        results.push(Town[i].update(i));
736 }
    //-----

738     if(document.hasFocus()) {
740         toggleTimer();
    } else {
742         toggleTimer();
    }

744

746 this.player.update();
    restartAudio();
748     if(timePassed==0) {
        bgaudio.pause();
750         bgaudio.src = 'sounds/main1.wav';
        bgaudio.load();
752         bgaudio.play();
    } else if (timePassed>1000 && timePassed < 5000) {
        {
754             this.accelerationTweening = 1.5;
            this.platformManager.maxDistanceBetween = 430;
756             //this.scoreColor = '#076C00';
            bgaudio.pause();
758             bgaudio.src = 'sounds/main2.wav';
            bgaudio.load();
760             bgaudio.play();
            fxaudio.pause();
762             fxaudio.src = 'sounds/levelup.wav';
            fxaudio.load();
764             fxaudio.play();
        } else if (timePassed>500000 && timePassed < 10000) {
766             this.accelerationTweening = 2.7;
            this.platformManager.maxDistanceBetween = 530;
768             //this.scoreColor = '#0300A9';
            bgaudio.pause();
770             bgaudio.src = 'sounds/main3.wav';
            bgaudio.load();
772             bgaudio.play();
            fxaudio.pause();
774             fxaudio.src = 'sounds/levelup.wav';

```

```

fxaudio.load();
fxaudio.play();
776 } else if (timePassed > 10000 && timePassed < 15000) {
778     this.accelerationTweening = 3.8;
this.platformManager.maxDistanceBetween = 580;
780 //this.scoreColor = '#9F8F00';
    bgaudio.pause();
782     bgaudio.src = 'sounds/main4.wav';
    bgaudio.load();
784     bgaudio.play();
    fxaudio.pause();
786     fxaudio.src = 'sounds/levelup.wav';
    fxaudio.load();
788     fxaudio.play();
} else if (timePassed > 15000 && timePassed < 20000) {
790     this.accelerationTweening = 4.4;
    this.PlatformManager.maxDistanceBetween = 610;
792     fxaudio.pause();
    fxaudio.src = 'sounds/levelup.wav';
794     fxaudio.load();
    fxaudio.play();
796 } else if (timePassed > 20000) {
    this.accelerationTweening = 5;
798     this.PlatformManager.maxDistanceBetween = 620;
    fxaudio.pause();
800     fxaudio.src = 'sounds/levelup.wav';
    fxaudio.load();
802     fxaudio.play();
}
804 this.acceleration += (this.accelerationTweening - this.acceleration) * 0.01;

806 for (i = 0; i < this.platformManager.platforms.length; i++) {
    if (this.player.intersects(this.platformManager.platforms[i])) {
808         this.collidedPlatform = this.platformManager.platforms[i];
        if (this.player.y < this.platformManager.platforms[i].y) {
810             this.player.y = this.platformManager.platforms[i].y;

812             // Gravity after Collision with Platform

```

```

    this.player.velocityY = 0;
814 }

816 this.player.x = this.player.previousX;
    this.player.y = this.player.previousY;
818

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

824

    if (this.player.intersectsLeft(this.platformManager.platforms[i])) {
826     this.player.x = this.collidedPlatform.x - 64;
        for (i = 0; i < 10; i++) {
828         // SpawnParticles @PlayerPostion with intersecting Platform Color
            this.particles[(this.particlesIndex++) % this.particlesMax] = new Particle({
830             x: this.player.x + this.player.width,
                y: random(this.player.y, this.player.y + this.player.height),
                velocityY: random(-30, 30),
832             color: randomChoice(['#181818', '#181818', this.collidedPlatform.color])
            });
834         };
    };

836

    // bounce player / push him away (effect)
838     this.player.velocityY = -10 + -(this.acceleration * 4);
        this.player.velocityX = -20 + -(this.acceleration * 4);
840

            if (timePassed > this.jumpCountRecord) {
                this.jumpCountRecord = timePassed;
            }
842
        } else {
844

```

```

846         // ----- Controller -----
847         // dragging: Mouse click & touch support
848         if (this.dragging || this.keys.SPACE || ↵
            this.keys.UP || this.keys.W) {
            this.player.velocityY = this.player.↵
                jumpSize;
850             this.jumpCount++;
            //play jump_sound
852             fxaudio.pause();
            fxaudio.src = '↵
                sounds/jump.↵
                    wav';
854             fxaudio.load();
            fxaudio.play();
856         }
            }
858     }
};
860
for (i = 0; i < this.platformManager.platforms.length ↵
    ; i++) {
862     this.platformManager.update();
};
864
for (i = 0; i < this.particles.length; i++) {
866     this.particles[i].update();
};
868     //-----
    //bg
870     return results;
    //-----
872 }
874 };
876
878 var selectedItem = 0;
880 var audioItem = 10;
882
InfinityRun.keydown = function() {
884     if (InfinityRun.keys.ESCAPE && GameState==State.↵
        Started) {
        InfinityRun.clear();
886         GameState = State.Menu;
        bgaudio.pause();

```



```

888         bgaudio.src = 'sounds/menu.wav';
          bgaudio.load();
890         bgaudio.play();

892         toggleTimer();

894     } else if (InfinityRun.keys.ESCAPE && GameState==>
        State.Menu && curMenuTab==MenuTab.Main) {
        GameState = State.Started;
896         toggleTimer();
    } else if (InfinityRun.keys.ESCAPE && GameState==>
        State.Menu && curMenuTab==MenuTab.Controls) {
898         curMenuTab = MenuTab.Main;
    } else if (InfinityRun.keys.ESCAPE && GameState==State.>
        .Menu && curMenuTab==MenuTab.Settings) {
900         curMenuTab = MenuTab.Main;
    } else if (InfinityRun.keys.ESCAPE && GameState==>
        State.Menu && curMenuTab==MenuTab.Highscore) {
902         curMenuTab = MenuTab.Main;
    } else if (InfinityRun.keys.ESCAPE && GameState==>
        State.Menu && curMenuTab==MenuTab.Credits) {
904         curMenuTab = MenuTab.Main;
    }

906    //main menu controls
908    if (InfinityRun.keys.UP && GameState == State.>
        Menu) {
        selectedItem = (selectedItem + items.>
            length - 1) % items.length;
910    }
    if (InfinityRun.keys.DOWN && GameState == State.>
        Menu) {
912        selectedItem = (selectedItem + 1) % items.>
            .length;
    }

914    // settings audio change
916    if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.>
        Settings && audioltem !=0) {
        audioltem = (audioltem + items.length - >
            1) % items.length;
918        if (bgaudio.volume>=0)
            bgaudio.volume-=0.1;
920        fxaudio.volume-=0.1;
    }

922    if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab.>
        .Settings && audioltem !=10) {

```

```

924         audioItem = (audioItem + 1) % items.␣
            length;
            if (bgaudio.volume < 1.0)
926         bgaudio.volume += 0.1;
            fxaudio.volume += 0.1;
928     }

930     if (InfinityRun.keys.ENTER && GameState == State.␣
        Menu) {
        callback(selectedItem);
932     }

934 }

936 Menu = function() {
    }
938

940 //----- Draw -----

942 InfinityRun.draw = function() {
944     if (GameState == State.Started) {
946         //-----
        //bg draw
948         var i, results;
950         i = Town.length;
        results = [];
952         while (i--) {
            results.push(Town[i].render(i));
954         }
        //-----

956         this.player.draw();
958         for (i = 0; i < this.platformManager.platforms.length;
            i++) {
960             this.platformManager.platforms[i].draw();
        };
962         //Draw particles
964         for (i = 0; i < this.particles.length; i++) {
            this.particles[i].draw();
966         };

```

```
968         //draw score
          this.font = '30pt Bungee';
970         this.textAlign="left";
          this.fillStyle = '#494949';
972         this.fillText('Score: ', this.width - 330, 65);
          this.fillText(timePassed , this.width - 170, 65);
974
          /*
976         * Main Menu
          *
978         */
          } else if (GameState == State.Menu && curMenuTab >
                    ==MenuTab.Main) {
980
          this.title = "InfinityRun";
982         items = ["Play","Controlls", "Settings", "Highscore", "Credits"];
984
          callback = function(numItem) { //if (numItem == 0) GameState=State.Started
986
          switch (numItem) {
            case 0:
988                 GameState=State.Started;
                 toggleTimer();
990                 break;
            case 1:
992                 curMenuTab=MenuTab.Controlls;
                 break;
994             case 2:
                 curMenuTab=MenuTab.Settings;
996                 break;
            case 3:
998                 curMenuTab=MenuTab.Highscore;
                 break;
1000             case 4:
                 curMenuTab=MenuTab.Credits;
1002                 break;
1004         }
1006
1008     };
          this.height = InfinityRun.height;
1010         this.width = InfinityRun.width;
          this.size = 70;
```

```

1012     var lingrad = this.createLinearGradient(0,0,0,↵
        this.height);
1014     lingrad.addColorStop(0, '#000');
        lingrad.addColorStop(1, '#023');
1016     this.fillStyle = lingrad;
        this.fillRect(0,0,this.width, this.height)

1018
        this.textAlign = "center";
1020     this.fillStyle = "White";

1022     var height = 100;
        //logo
1024     this.drawImage(bglogo, this.width-500, this.height ↵
        -300);
        //—————
1026     if (this.title) {
        this.font = Math.floor(this.size*1.3). ↵
            toString() + "px Bungee";
1028         this.fillText(this.title, this.width/2, ↵
            height);
            height+= height;

1030     }

1032     for (var i = 0; i < items.length; ++i)
    {
1034         var size = Math.floor(this.size*0.8);
        if (i == selectedItem)
1036         {
            this.fillStyle = "#A9F5F2";
            size = this.size+5;
1038         }
        this.font = size.toString() + "px Bungee" ↵
            ;
        height += this.size;
1042         this.fillText(items[i], InfinityRun.width ↵
            /2, height);
            this.fillStyle = "White";

1044     }
        //—————↵

1046     //bg dd <— ??
        return results;
1048     //—————↵

1050     /*
        * Settings Tab

```

```

1052         *
1053         */
1054     }else if (GameState == State.Menu && curMenuTab==
        MenuTab.Controlls){
            this.title = "Controlls";
1056     items = highScore;

1058

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

1060

        switch (volume) {

1062

        }

1064

1066

        };
1068     this.height = InfinityRun.height;
        this.width = InfinityRun.width;

1070

        var lingrad = this.createLinearGradient(0,0,0,
            this.height);
1072     lingrad.addColorStop(0, '#000');
        lingrad.addColorStop(1, '#023');
1074     this.fillStyle = lingrad;
        this.fillRect(0,0,this.width, this.height, items[
            i]);

1076

        this.textAlign = "center";
1078     this.fillStyle = "White";

1080

        var width = 10;
        var height = 150;

1082

        if (this.title) {
1084             this.font = Math.floor(this.size*1.3).
                toString() + "px␣Bungee";
            this.fillText(this.title, this.width/2,
                150);
1086             height+= height;
        }
1088     var distanceText = 50
        this.font = Math.floor(40).toString() + "px␣
            Bungee";
1090     this.textAlign = "left";
        //Names
1092     this.fillText("Menu:", this.width/5, 300);

```

```

    this.font = Math.floor(20).toString() + "px␣↵
        Bungee";
1094    this.fillText("[ESC]␣↵Menu", this.width/5, 300+↵
        distanceText);
    this.fillText("[Arrow␣↵up/down]␣↵To␣↵navigate", ↵
        this.width/5, 300+2*distanceText);
1096    this.fillText("[Enter]␣↵Accept", this.width/5, ↵
        300+3*distanceText);

1098    this.font = Math.floor(40).toString() + "px␣↵
        Bungee";
    this.textAlign = "left";
1100    //Names
    this.fillText("Game:", this.width/5, 300+5*↵
        distanceText);
1102    this.font = Math.floor(20).toString() + "px␣↵
        Bungee";
    this.fillText("[ESC]␣↵Menu", this.width/5, ↵
        300+6*distanceText);
1104    this.fillText("[W],␣↵[Arrow␣↵up],␣↵[Leertaste]␣↵↵
        Jump", this.width/5, 300+7*distanceText);
    this.fillText("[Arrow␣↵down]␣↵↵increase␣↵falling␣↵↵
        speed", this.width/5, 300+8*distanceText);
1106 }
    else if (GameState == State.Menu && curMenuTab==↵
        MenuTab.Settings){

1108
        this.title = "Settings";
1110        items = [0,10, 20, 30, 40, 50, 60, 70, 80, 90 , ↵
            100];

1112        callback = function(volume) { //if (numItem == 0)↵
            GameState=State.Started

1114        switch (volume) {

1116        }

1118

1120    };

1122    this.height = InfinityRun.height;
    this.width = InfinityRun.width;

1124    var lingrad = this.createLinearGradient(0,0,0,↵
        this.height);
1126    lingrad.addColorStop(0, '#000');

```

```

1128     lingrad.addColorStop(1, '#023');
        this.fillStyle = lingrad;
        this.fillRect(0,0,this.width, this.height, items[
            i]);

1130
        this.textAlign = "center";
1132     this.fillStyle = "White";

1134     var width = 10;
        var height = 10;
1136     var posx = 130;
        var posy = 380;
1138     this.space = 15;
        this.heightincr = 4;

1140
        if (this.title) {
1142             this.font = Math.floor(this.size*1.3).
                toString() + "px_Bungee";
            this.fillText(this.title, this.width/2,
                150);
1144             height+= height;
        }

1146
        this.font = "55px_Bungee";
1148     this.fillText('Volume', 240, 300);

1150

1152
        for (var i = 0; i < items.length; ++i) {
1154             var size = Math.floor(this.size*0.8);
            if (i == audioltem)
1156             {
                this.fillStyle = "#A9F5F2";
1158                 size = this.size+5;
            }
            this.font = size.toString() + "px_Bungee"
                ;
            posx += this.space;
1162             posy -= this.heightincr;
            height += this.heightincr;

1164
            items[i] = this.fillRect(posx, posy, width,
                height);

1166
            //this.fillText(items[i], InfinityRun.
                width/2, height);
1168             this.fillStyle = "White";

```

```

1170     }

1172     /*
1173     * Highscore Tab
1174     *
1175     */
1176 } else if (GameState == State.Menu && curMenuTab >
    == MenuTab.Highscore) {

1178

1180     this.title = "Highscore";
1181     items = highScore;

1182

1183     callback = function(volume) { //if (numItem == 0) >
        GameState=State.Started

1184

1185     switch (volume) {

1186

1187     }

1188

1189

1190

1191     };
1192     this.height = InfinityRun.height;
1193     this.width = InfinityRun.width;

1194

1195     var lingrad = this.createLinearGradient(0,0,0,>
        this.height);
1196     lingrad.addColorStop(0, '#000');
1197     lingrad.addColorStop(1, '#023');
1198     this.fillStyle = lingrad;
1199     this.fillRect(0,0,this.width, this.height, items[>
        i]);

1200

1201     this.textAlign = "center";
1202     this.fillStyle = "White";

1203

1204     var width = 10;
1205     var height = 150;

1206

1207

1208     if (this.title) {
1209         this.font = Math.floor(this.size*1.3).>
            toString() + "px␣Bungee";
1210         this.fillText(this.title, this.width/2, >

```



```

1212         150);
           height+= height;
1214     }
1216     var rank = 1;
1218     for (var i = 0; i < items.length; ++i)
1220     {
1222         var size = Math.floor(this.size*0.8);
           if (i == selectedItem)
1224         {
           this.fillStyle = "#A9F5F2";
           size = this.size+5;
1226         }
           this.font = 0.6*size.toString() + "px";
           Bungee";
           height += 50;
1228           this.fillText(rank + ". " + items[i],
             InfinityRun.width/2, height);
           this.fillStyle = "White";
1230
           rank++;
1232     }
1234 }
// Credits Menu

1236 else if (GameState == State.Menu && curMenuTab ==
  MenuTab.Credits) {
1238     this.title = "Credits";
     items = highScore;
1240
     callback = function(volume) { //if (numItem == 0)
       GameState=State.Started
1242
     switch (volume) {
1244
     }
1246
1248
1250 };
     this.height = InfinityRun.height;
     this.width = InfinityRun.width;
1252

```

```

var lingrad = this.createLinearGradient(0,0,0,▷
    this.height);
1254 lingrad.addColorStop(0, '#000');
lingrad.addColorStop(1, '#023');
1256 this.fillStyle = lingrad;
this.fillRect(0,0,this.width, this.height, items[▷
    i]);

1258
this.textAlign = "center";
1260 this.fillStyle = "White";

1262 var width = 10;
var height = 150;
1264

1266 if (this.title) {
    this.font = Math.floor(this.size*1.3).▷
        toString() + "px_Bungee";
1268 this.fillText(this.title, this.width/2, ▷
        150);
    height+= height;
1270 }
var distanceText = 50
1272 this.font = Math.floor(50).toString() + "px_▷
    Bungee";
this.textAlign = "left";
1274 //Names
this.fillText("Group_Members:", this.width/5, ▷
    300);
1276 this.font = Math.floor(40).toString() + "px_▷
    Bungee";
this.fillText("__Florian_Durli", this.width/5, ▷
    300+distanceText);
1278 this.fillText("__Koray_Emtekin", this.width/5, ▷
    300+2*distanceText);
this.fillText("__Jannik_Ivosevic", this.width/5, ▷
    300+3*distanceText);
1280 this.fillText("__Marco_Mayer", this.width/5, ▷
    300+4*distanceText);
this.fillText("__Johannes_But", this.width/5, ▷
    300+5*distanceText);
1282 //bottom info
this.font = Math.floor(15).toString() + "px_▷
    Times_▷
    _New_Roman";
1284 this.textAlign = "center";
distanceText = 20;
1286 this.fillText("InfinityRun_is_a_nonprofit_▷
    students_project_at_\"Hochschule Furtwangen\"▷

```

```

        /\ "Furtwangen University.\ "Special thanks to
        \ "Soulwire\ "for his Sketch.js Minimal
        JavaScript Creative Coding Framework", this.
        width/2, this.height - 2.2*distanceText);
        this.fillText("Sounds: freesounds.org Special
        thanks to Jack Rugil for his Parrallax Skyline
        ", this.width/2, this.height - distanceText - 5);
1288      this.fillText("2016", this.width/2, this.height - 8);
        ;

1290
    }

1292
    //Debug
1294    if (debug) {
        this.font = '16pt Arial';
1296        this.fillStyle = '#181818';
        this.fillText('Record: ' + s + " " + sc/*this.
        jumpCountRecord*/, this.width - 150, 33);
1298        this.fillStyle = this.scoreColor;
        this.fillText('Jumps: ' + this.jumpCount, this.
        width - 150, 50);
1300    }

1302 };

1304 InfinityRun.resize = function() {
    /* todo Windowscale optimization
1306     *
1307     *
1308     */
    }

```

A.3. game.css

```

1 body{
    background: #e3e3e3;
3    overflow: hidden;
    margin: 0;
5    padding: 0;
    text-align: center;
7 }
    #container{
9     /*margin-top: 10%;*/
    display: inline-block;
11 }

13 canvas{
    font-family: 'Bungee', cursive;

```

```

15  background: #cecece;
    border: 1px solid #181818;
17 }

19 canvas.sunsetFilter {
    -webkit-animation: sunset-animation 70s;
21 }

23 canvas.invertFilter {
    -webkit-animation: invert-animation 20s;
25 }

27

29 @-webkit-keyframes sunset-animation {
    0% {
31     -webkit-filter: sepia(0) saturate(2);
    }
33
    50% {
35     -webkit-filter: sepia(1) saturate(15);
    }
37
    100% {
39     -webkit-filter: sepia(0) saturate(2);
    }
41 }

43

45 @-webkit-keyframes invert-animation {
    0% {
47     -webkit-filter: invert(0);
    }
49
    50% {
51     -webkit-filter: invert(.8);
    }
53
    100% {
55     -webkit-filter: invert(.0);
    }
57 }

```

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>
    <link href="https://fonts.googleapis.com/css?family=Bungee" rel="stylesheet">
7
    <title>Infinity Run</title>
9
    <link href="css/game.css" rel="stylesheet" type="text/css">
11    <link rel="shortcut icon" type="image/x-icon" href="image/favicon.png">
    </head>
13 <body>
    <!-- Game div -->
15 <div id="container">

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