



Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent : **Gabriela Mai**
Vorgelegt am : 13. Januar 2017
Vorgelegt von : Gruppe 4

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

Inhaltsverzeichnis

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

2.2.4	Bibliothek	10
2.2.5	Code	10
2.2.6	Nächste Ziele	16
2.3	Implementation - Endstand	16
2.3.1	Spielkonzept Änderungen	16
2.3.2	Funktionsdiagramm	16
2.3.3	Grafiken	18
2.3.4	Code Änderungen	19
2.3.5	Das Spiel - Endstand	23
2.3.6	Sounds	25
2.4	Test	25
2.4.1	Umfang	25
2.4.2	Testplan	25
2.4.3	Testberichte	26
2.4.4	Testberichte analysieren	31
2.4.5	Bug Beschreibung	31
2.4.6	Browsertest	32
2.4.7	Bug Behebung	32
3	Github - Changelog	33
4	Ausblick	39
5	Fazit	41
	Literaturverzeichnis	43
	Eidesstattliche Erklärung	45
A	Anhang	47
A.1	game.js	47
A.2	game.css	81

A.3	index.html	82
-----	----------------------	----

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
Abbildung 13: QR-Code zum Webspace	24

Tabellenverzeichnis

Tabelle 1: Rollenverteilung	2
Tabelle 2: Phase 1: Entwurf und Anforderungen	6
Tabelle 3: Phase 2: Implementierung	6
Tabelle 4: Phase 3: Test	6
Tabelle 5: Phase 4: Dokumentation und Präsentation	7
Tabelle 6: Releaseplan	7
Tabelle 7: Funktionsbeschreibung	18
Tabelle 8: Menü-Steuerung	24
Tabelle 9: Spiel-Steuerung	24
Tabelle 10: Sound Links	25
Tabelle 11: Bugs	31
Tabelle 12: Github Namen	33

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

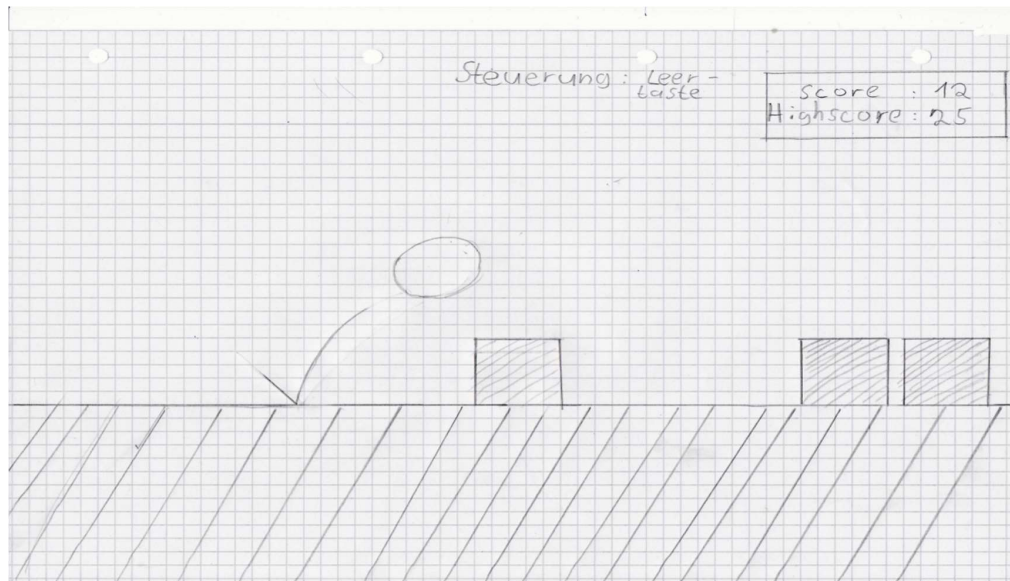


Abbildung 6.: Entwurfsskizze

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.

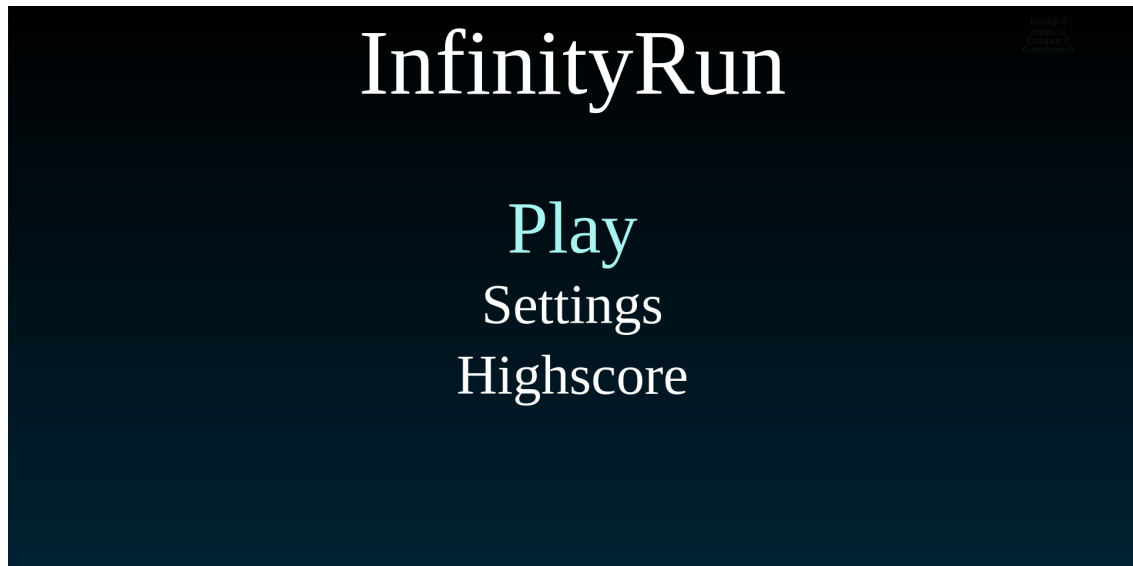


Abbildung 7.: Startbildschirm



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
    })
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)

```

```
{
    this.first.width = random(450, >
        InfinityRun.width + 200);
    this.first.x = (this.third.x + this.third >
        .width) + random(this.>
        maxDistanceBetween - 150, this.>
        maxDistanceBetween);
    this.first.y = random(this.third.y - 32, >
        InfinityRun.height - 80);
    this.first.height = this.first.y + >
        InfinityRun.height + 10;
    this.first.color = randomChoice(this.>
        colors);
}

this.second.x -= 3 + InfinityRun.acceleration;
if (this.second.x + this.second.width < 0)
{
    this.second.width = random(450, >
        InfinityRun.width + 200);
    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);
    this.second.height = this.second.y + >
        InfinityRun.height + 10;
    this.second.color = randomChoice(this.>
        colors);
}

this.third.x -= 3 + InfinityRun.acceleration;
if (this.third.x + this.third.width < 0)
{
    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
- Spielfigur ändert sich nicht nach Aufprall

2.3.2. 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)  
            {  
                12 bgaudio.currentTime = 0;  
                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.

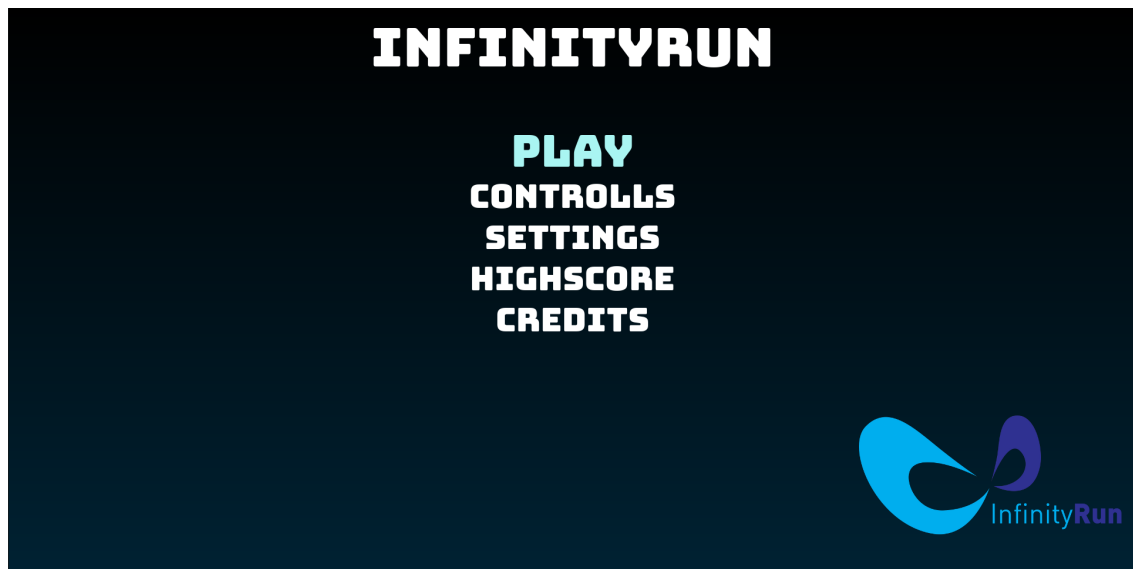


Abbildung 11.: Startbildschirm - Endstand



Abbildung 12.: Das Spiel - Endstand

2.3.5.1. Steuerung

Menü-Steuerung:

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

Tabelle 8.: Menü-Steuerung

Spiel-Steuerung:

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

Tabelle 9.: Spiel-Steuerung

2.3.5.2. Webserver und Smartphone

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

Spiel: <http://infinityrun.square7.ch/>



Abbildung 13.: QR-Code zum Webspace

2.3.6. Sounds

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

Folgende Sounds werden wir verwenden:

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

Tabelle 10.: Sound Links

2.4. Test

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

2.4.1. Umfang

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

2.4.2. Testplan

2.4.2.1. Testumgebung

- Browser
- Betriebssystem
- System
- Auflösung

2.4.2.2. Modultest

- Steuerung
- Sound

- Spieloberfläche
- Spielverlauf

2.4.2.3. Systemtest

- Performance

2.4.3. Testberichte

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

Testbericht

Name: Felix Duffner, Beruf: Zimmermann

Datum: 28.12.2016

1 Testumgebung

Browser	Chrome
Betriebssystem	Windows 7
System	Laptop
Auflösung	1280x720

2 Modultest

2.1 Steuerung

	Erfüllt Ja/Nein?	Anmerkungen
Funktioniert	Ja	
Intuitiv?	Ja	

2.2 Sound

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

2.3 Spieloberfläche

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

2.4 Spielverlauf

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

3 Systemtest

Performance: Ab und zu kleinere Ruckler. Sonst gut.

Testbericht

Name: Maurice Ketterer, Beruf: Verwaltungs Azubi

Datum: 30.12.2016

1 Testumgebung

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

2 Modultest

2.1 Steuerung

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

2.2 Sound

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

2.3 Spieloberfläche

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

2.4 Spielverlauf

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

3 Systemtest

Performance: Läuft gut.

Testbericht

Name: Tomas Müller, Beruf: Mechatroniker

Datum: 03.01.2017

1 Testumgebung

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

2 Modultest

2.1 Steuerung

	Erfüllt Ja/Nein?	Anmerkungen
Funktioniert	Ja	
Intuitiv?	Ja	

2.2 Sound

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

2.3 Spieloberfläche

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

2.4 Spielverlauf

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

3 Systemtest

Performance: Kaum Spielbar durch ruckeln.

Testbericht

Name: Gruppe 4, Beruf: Studenten

Datum: 21.12.2016

1 Testumgebung

Browser	Chrome
Betriebssystem	Linux
System	Laptop
Auflösung	Full-HD

2 Modultest

2.1 Steuerung

	Erfüllt Ja/Nein?	Anmerkungen
Funktioniert	Ja	
Intuitiv?	Ja	

2.2 Sound

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

2.3 Spieloberfläche

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

2.4 Spielverlauf

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

3 Systemtest

Performance: Nicht durchgehend Flüssig Spielbar.

2.4.4. Testberichte analysieren

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

2.4.4.1. Bugliste

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

Tabelle 11.: Bugs

Legende:

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

2.4.5. Bug Beschreibung

2.4.5.1. Performance - Priorität 1

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

2.4.5.2. Hintergrundmusik - Priorität 2

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

2.4.5.3. Effektlautstärke - Priorität 3

Die Effektlautstärke ist deutlich zu laut und sollte angepasst werden.

2.4.5.4. Menüsound - Priorität 3

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

2.4.6. Browsertest

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

2.4.7. Bug Behebung

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

2.4.7.1. Performance - Priorität 1

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

2.4.7.2. Hintergrundmusik - Priorität 2

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

2.4.7.3. Effektlautstärke - Priorität 3

Die Effektlautstärke wurde von uns nun perfekt zum Spiel angepasst.

2.4.7.4. Menüsound - Priorität 3

Diesen Fehler konnten bei der Behebung der Hintergrundmusik zusätzlich lösen.

3. Github - Changelog

Unser Github finden Sie unter <https://github.com/Slay3r/InfinityRun/>.

Unsere Versionen die nach Release Plan erfolgt sind finden Sie unter <https://github.com/Slay3r/InfinityRun/releases>. Aus unserem Changelog ist auch zu entnehmen, wann welcher Fehler behoben wurde. Der Changelog wird aus unseren Github Commits per Befehl exportiert.

Befehl:

```
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 12.: Github Namen

Changelog:

```
1 487056f Florian Durli 2017-01-13 Changelog

3 f5f3c09 butjo          2017-01-12 Dokumentation angepasst , >
   QR-Code hinzugefügt

5 8727fbd butjo          2017-01-11 Präsentation Test

7 18d2ff9 Slay3r          2017-01-11 Bug Beschreibung

9 e090c8d Slay3r          2017-01-11 Doku Testphase , Code >
   Cleanup
```

11	3983a15	r4qtor	2017-01-10	Add files via upload
13	2447123	r4qtor	2017-01-10	Add files via upload
15	dbc5093	r4qtor	2017-01-10	Add files via upload
17	0c43c83	r4qtor	2017-01-10	Soundssystem , bugfixes , audio tweaks
19	6e383de	butjo	2017-01-09	Filtersettings hinzugefügt noch nicht komplett implementiert
21	4340101	butjo	2017-01-09	Graphiksettings implementiert
23	ba0c690	butjo	2017-01-09	Graphische Oberfläche zum Settingsändern implementiert
25	ff0c36e	Florian Durli	2017-01-02	Browsertest
27	0af9b6b	Slay3r	2016-12-21	Delete Logos
29	4376420	Slay3r	2016-12-21	Testplan
31	14c229b	Slay3r	2016-12-16	fix
33	86c56e1	Slay3r	2016-12-16	Doku final
35	c4876c7	Florian Durli	2016-12-16	Controlls
37	9ce6d99	Florian Durli	2016-12-16	Doku Server , Steuerung
39	e1c7138	r4qtor	2016-12-15	acceleration fix
41	188bcd8	r4qtor	2016-12-15	Delete sketch.minlesbar.js (RE: unbenutzt)
43	55b3162	r4qtor	2016-12-15	bug fixes , 'Farbänderung' , kleines 'Hindernis '

45	0f5b9cc butjo	2016-12-15 HighScore gefixed ↵
	Plattformen erhöht	
47	414cbc9 butjo	2016-12-14 Creditsverbesserungen
49	fc96b76 butjo	2016-12-14 Credits eingefügt
51	3b990da butjo	2016-12-14 Audioeinstellungen ↵
	implementiert	
53	6fb84f9 Slay3r	2016-12-14 Doku Quellen angepasst
55	02cecd1 butjo	2016-12-14 Merge branch 'master' of ↵
	https://github.com/Slay3r/InfinityRun	
57	682c0d2 butjo	2016-12-14 Farbschema geändert
59	449cf21 Slay3r	2016-12-14 Doku
61	288bb25 butjo	2016-12-14 Merge
63	2003a79 butjo	2016-12-13 Dacharten erweitert
65	42e9a8f butjo	2016-12-13 Code verbessert/↵
	Optimiert	
67	67cc215 Slay3r	2016-12-13 Doku logo
69	2d7d953 butjo	2016-12-12 Favicon und Logo im Menü
71	2b8a139 ans77	2016-12-12 Logos hinzugefügt
73	92b7b28 Slay3r	2016-12-12 Code Cleanup
75	30b0c6f Slay3r	2016-12-12 Clean up
77	37f6e94 butjo	2016-12-12 Favicon hinzugefügt

79	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
81	8d06d8a butjo	2016-12-09 skyline eingefügt
83	4e52883 Slay3r	2016-12-07 Doku + Alte Dateien
85	2c3491b butjo	2016-12-05 Sounds hinzugefügt teilweise noch ein paar Bugs und dirty code
87	2e6dfbe butjo	2016-11-30 Dirty Code mit versuch den background zu implementieren files sind mit prefix back gekennzeichnet und liegen im Hauptverzeichnis.
89	dda171b Slay3r	2016-11-30 Changelog geändert
91	ffe660b Slay3r	2016-11-30 Ausblick/Fazit auskommentiert
93	01e309a Slay3r	2016-11-30 Sounds
95	3a05368 Slay3r	2016-11-30 Neue Dokumentation hinzufügen
97	7c3596c Slay3r	2016-11-30 Neue Dokumentation hinzufügen
99	bfdb05c Slay3r	2016-11-30 entfernen der Dokumentation
101	f944707 r4qtor	2016-11-25 Querlesung – Marco
103	8b6bdde Florian Durli	2016-11-25 Abgabe
105	bc8d933 Florian Durli	2016-11-25 Code Cleanup für Doku
107	b6d7a09 butjo	2016-11-24 Rechtschreibkorrekturen

109	1faa558	r4qtor	2016-11-24	Delete phasen.tex root/
111	51f4a79	r4qtor	2016-11-24	updated phasen.tex
113	62af4b8	Krusher999	2016-11-24	Letzten Codes ↗ geschrieben
115	93b8965	Florian Durli	2016-11-23	fix
117	9027301	Florian Durli	2016-11-23	Changelog finale Lösung
119	1392138	Florian Durli	2016-11-23	Jojos Description in ↗ LaTeX
121	25ff037	butjo	2016-11-23	Beschreibung der ↗ Codeteile in der phasen.tex von Johannes
123	1b2348c	Florian Durli	2016-11-23	Changelog Additionally
125	cf467dc	Florian Durli	2016-11-23	Changelog Additionally
127	6db1ad6	butjo	2016-11-23	Merge branch 'master' of ↗ https://github.com/Slay3r/InfinityRun
129	b924713	Slay3r	2016-11-23	Generated Changelog
131	69dd747	butjo	2016-11-23	Merge branch 'master' of ↗ https://github.com/Slay3r/InfinityRun
133	203ae2e	Slay3r	2016-11-23	Bilder des Spiels
135	d274cfc	Slay3r	2016-11-23	Anforderungen
137	a0909ac	Slay3r	2016-11-23	Literaturverzeichnis
139	19e2f3d	Slay3r	2016-11-23	Doku update
141	21889f9	Florian Durli	2016-11-22	Add Changelog

143	743c95a	butjo	2016-11-16	Formatierte sketch.min.js ↗
145	d64d254	butjo	2016-11-16	Endlose Schwierigkeitserhöhung ↗
147	3707b94	butjo	2016-11-16	Präsentation Zwischenstand ↗
149	f23c9be	Slay3r	2016-11-16	Präsentation überarbeitet ↗
151	53aa72e	butjo	2016-11-16	Präsentation und test
153	b5cb978	Florian Durli	2016-11-16	Merge pull request #1 from r4qtor/master ↗
155	275bd69	r4qtor	2016-11-15	tiny cleanup & incl. menu ↗
157	11a5e55	Slay3r	2016-11-09	Basis implementation
159	c783850	Slay3r	2016-11-09	Bilder hinzugefügt
161	d88d0d2	Slay3r	2016-11-09	Dokumentations Basis
163	39b4705	Florian	2016-10-19	Initial Struktur der Ordner und files ↗
165	093797e	Florian Durli	2016-10-19	Delete Requirements
167	56c4aae	Florian	2016-10-19	doc Ordner
169	b3b83fd	Florian	2016-10-19	Requirements hinzugefügt
171	b4d2627	Florian Durli	2016-10-19	Initial commit

4. Ausblick

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

5. Fazit

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

Literaturverzeichnis

- [bro] BROWERSHOTS.ORG: *Browsershots* <http://browsershots.org/>
- [dis] DISSIMULATE: *Skyline* <https://codepen.io/dissimulate/pen/CAzlt>
- [Fre] FREESOUND: *Freesound.org* <https://www.freesound.org/>
- [Git] GITHUB: *Softwareverwaltung* <https://github.com/>
- [Goo] GOOGLE: *Google Chrome* <https://www.google.com/chrome/>
- [Gru] GRUPPE4: *Changelog* <https://github.com/Slay3r/InfinityRun/commits/master>
- [Ho] HO, Don: *Notepad++* <https://notepad-plus-plus.org/>
- [sou] SOULWIRE: *Sketch Bibliothek* <https://github.com/soulwire/sketch.js>
- [Sqa] SQUARE 7 HOST SERVICE PROVIDER: *Kostenloses Webhosting* <http://www.square7.ch/>
- [Tea] TEAM, The G.: *Bildbearbeitungssoftware* <https://www.gimp.org/>

Eidesstattliche Erklärung

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

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

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

FURTWANGEN, den 13. Januar 2017 Florian Durli

FURTWANGEN, den 13. Januar 2017 Jannik Ivosevic

FURTWANGEN, den 13. Januar 2017 Johannes But

FURTWANGEN, den 13. Januar 2017 Marco Mayer

FURTWANGEN, den 13. Januar 2017 Koray Emtekin

A. Anhang

A.1. 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 var debug = false;

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

```

```

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

42
   //timer
44 var s = 0,
   ms = 0,
46 playTimer = false;

48 var highScore = new Array(10);
   highScore = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
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 //----->

   var bgFX;
66 var sfx;

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

72
   function playbgFX (soundID) {
74         bgFX = createjs.Sound.play(soundID, {loop:-1});
       }

76

78 function playSFX (soundID) {
       sfx = createjs.Sound.play(soundID);
80 }

```

```

82 // randomizer
   function random(min, max) {
84     return Math.round(min + (Math.random() * (max - min)))
       };
   }
86
   function randomChoice(array) {
88     return array[Math.round(random(0, array.length - 1))]
       ];
   }
90

92 //initialize Sketch Framework
   var InfinityRun = Sketch.create({
94     fullscreen: true,
       width: 640,
96     height: 360,
       container: document.getElementById('container')
98 });
   var qs = document.querySelector('canvas');
100
   //bg func
102 Tower = function(config)
   {
104     return this.reset(config);
   };
106
   Tower.prototype.reset = function(config)
108 {
       this.layer = config.layer;
110     this.x = config.x;
       this.y = config.y;
112     this.width = config.width;
       this.height = config.height;
114     this.color = config.color;
       this.summitTop = floor(random(0, 15)) ==
         0;
116     this.summitTopWidth = random(this.width * .01,
       this.width * .07);
       this.summitTopHeight = random(10, 20);
118     this.singleroofTop = floor(random(0, 10)) == 0;
       this.singleroofTopHeight = this.width / random(2,
         4);
120     this.singleroofTopDirection = round(random(0, 1))
       == 0;
       this.normalTop = !this.singleroofTop &&
         floor(random(0, 10)) == 0;
122     this.normalTopHeight = this.width / random(2, 4);

```

```

    this.normalTopchimney = round(random(0, 1)) == 0;
124     this.coneTop = !this.singleroofTop && !this.normalTop && floor(random(0, 10)) == 0;
    this.coneTopHeight = this.width / random(3, 4);
126     this.coneTopWidth = this.width / random(1, 2);
    this.coneTopeflat = round(random(0, 1)) == 0;
128     this.companyTop = !this.singleroofTop && !this.summitTop && !this.radioTop && !this.normalTop && floor(random(0, 10)) == 0;
    this.companyTopHeight = this.width / random(4, 6);
    ;
130     this.companyTopcount = 4;//round(random(3, 6));
    this.radioTop = !this.summitTop && floor(random(0, 10)) == 0;
132     this.radioTopWidth = this.layer / 2;
    return this.radioTopHeight = random(6, 30);
134 };

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

144     if (vgaquality > 0){ //graphics higher then low
146         if (this.singleroofTop)
        {
148             InfinityRun.beginPath();
            InfinityRun.moveTo(this.x, this.y);
150             InfinityRun.lineTo(this.x + this.width, this.y);
            if (this.singleroofTopDirection)
152             {
                InfinityRun.lineTo(this.x + this.width, this.y - this.singleroofTopHeight);
154             }
            else
156             {

```

```

InfinityRun.lineTo(this.x, this.y,
    - this.singleroofTopHeight);
158     }
    InfinityRun.closePath();
160     InfinityRun.fill();
    InfinityRun.stroke();
162 }

164 if (this.normalTop)
{
166     InfinityRun.beginPath();
    InfinityRun.moveTo(this.x, this.y);
168     InfinityRun.lineTo(this.x + this.width,
        this.y);
        InfinityRun.lineTo(this.x + (this.
            width/2), this.y-this.
                normalTopHeight);
170     InfinityRun.closePath();
    InfinityRun.fill();
172     InfinityRun.stroke();
        if (this.normalTopchimney)
174         {
            InfinityRun.beginPath();
176             InfinityRun.moveTo(this.x,
                +(this.width/5), this.
                    y);
            InfinityRun.lineTo(this.x,
                +(this.width/5), this.
                    y- 0.8*(this.
                        normalTopHeight));
178             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
                        );
180             InfinityRun.closePath();
            InfinityRun.fill();
182             InfinityRun.stroke();
        }
184 }
186 if (vgaquality > 1 && this.coneTop)
{
    InfinityRun.beginPath();
188     InfinityRun.moveTo(this.x, this.y);

```

```

InfinityRun.lineTo(this.x + (this.width -
    this.coneTopWidth)/2, this.y - this.
    coneTopHeight);
190     if (!this.coneTopeflat)
        {
192             InfinityRun.lineTo(this.x +
                +(this.width/2), this.
                y - (this.coneTopHeight
                * 1.3));
        }
194     InfinityRun.lineTo(this.x + ((
        this.width - this.coneTopWidth)
        / 2) + this.coneTopWidth, this.y -
        this.coneTopHeight);
        InfinityRun.lineTo(this.x + this.
        width, this.y);
196     InfinityRun.closePath();
        InfinityRun.fill();
198     InfinityRun.stroke();

200 }
    if (vgaquality > 1 && this.companyTop)
202 {
        var ctc = 1;
204         while (ctc <= this.companyTopcount)
            {
206                 InfinityRun.beginPath();
                InfinityRun.moveTo(this.x , this.
                    y);
208                 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);
210                 InfinityRun.closePath();
                InfinityRun.fill();
212                 InfinityRun.stroke();
                ctc++;
214             }
        }
216     if (vgaquality > 1 && this.summitTop)
        {
218             InfinityRun.beginPath();
            InfinityRun.moveTo(this.x + (this.width /
                2), this.y - this.summitTopHeight);

```

```

220         InfinityRun.lineTo(this.x + (this.width / 2) + this.summitTopWidth, this.y);
        InfinityRun.lineTo(this.x + (this.width / 2) - this.summitTopWidth, this.y);
222         InfinityRun.closePath();
        InfinityRun.fill();
224         InfinityRun.stroke();
    }

226     if (vgaquality > 1 && this.radioTop)
228     {
        InfinityRun.beginPath();
230         InfinityRun.moveTo(this.x + (this.width / 2), this.y - this.radioTopHeight);
        InfinityRun.lineTo(this.x + (this.width / 2), this.y);
232         InfinityRun.lineWidth = this.radioTopWidth;
        return InfinityRun.stroke();
234     }
    }
236 };

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

248         this.height = {
            min: config.height.min,
250             max: config.height.max
        };

252         this.speed = config.speed;
254         this.color = config.color;
        this.populate();
256         return this;
    };

258 Street.prototype.populate = function()
260 {
    var newHeight, newWidth, results, totalWidth;
262     totalWidth = 0;

```

```

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

```



```

302         x: lastTower.x + lastTower.width,
        y: InfinityRun.height - newHeight,
        width: newWidth,
304         height: newHeight,
        color: this.color
306     });
    return this.alltowers.push(this.alltowers.shift());
308 }
};

310 Street.prototype.render = function()
312 {
    var i;
314     i = this.alltowers.length;
    InfinityRun.save();
316     InfinityRun.translate(this.x, (
        InfinityRun.height - (InfinityRun.
        height - (-jumpheight * 0.5) - 400)) / 20 *
        this.layer);

318     while (i--) {
        this.alltowers[i].render(i);
320     }
    return InfinityRun.restore();
322 };

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

336 // Set X,Y
    Vector2.prototype.setPosition = function(x, y) {
338     this.previousX = this.x;
340     this.previousY = this.y;

342     this.x = x;
    this.y = y;

```

```
344     };
346 // Set X
    Vector2.prototype.setX = function(x) {
348         this.previousX = this.x;
350         this.x = x;

352     };

354 // Set Y
    Vector2.prototype.setY = function(y) {
356         this.previousY = this.y;
358         this.y = y;

360     };

362 // Collision / Intersection Top
    Vector2.prototype.intersects = function(obj) {
364         if (obj.x < this.x + this.width && obj.y < this.y + ↵
            this.height &&
366             obj.x + obj.width > this.x && obj.y + obj.height ↵
                > this.y) {
            return true;
368         }

370         return false;
    };

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

380         return false;
    };

382 //----- Player -----
384 function Player(options) {

386     this.setPosition(options.x, options.y);
    this.width = options.width;
388     this.height = options.height;
```

```

    this.velocityX = 0;
390    this.velocityY = 0;
    this.jumpSize = -13;
392    this.color = '#181818';

394 }

396 Player.prototype = new Vector2;

398 Player.prototype.update = function() {
    // Gravity
400    this.velocityY += 1;
    //um bg zu ändern
402    jumpheight=(this.y);
    this.setPosition(this.x + this.velocityX, this.y +
        this.velocityY);

404    if (this.y > InfinityRun.height || this.x + this.
        width < 0) {
406        this.x = 150;
        this.y = 50;
408        this.velocityX = 0;
        this.velocityY = 0;
410        InfinityRun.jumpCount = 0;
        InfinityRun.acceleration = 0;
412        InfinityRun.accelerationTweening = 0;
        InfinityRun.scoreColor = '#181818';
414        InfinityRun.platformManager.maxDistanceBetween =
            350;

416        //InfinityRun.pause();

418        //highscore update
        if (timePassed>highScore[0]){
420            var help =highScore[0];
            var help2=highScore[1];
422            highScore[0]=timePassed;
            for(i=1; i<=9;i++){
424                help2 = highScore[i];
                highScore[i]=help;
426                help=help2;

428            }

430        }
        InfinityRun.platformManager.updateWhenLose();

432        playSFX("Crash");

```

```

434         bgFX.stop();
           difficulty = 0;
436         ms = 0;
           }
438
           if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE || ↵
               InfinityRun.keys.W || InfinityRun.dragging) && ↵
               this.velocityY < -8) {
440             this.velocityY += -0.75;
           }
442
           }
444
           if (InfinityRun.keys.DOWN) {
446             this.velocityY += 1;
           }
448
           };
450
           Player.prototype.draw = function() {
452             InfinityRun.fillStyle = this.color;
               InfinityRun.fillRect(this.x, this.y, this.width, this ↵
               .height);
454           };
456 // ————— Platforms —————
458 function Platform(options) {
           this.x = options.x;
460           this.y = options.y;
           this.width = options.width;
462           this.height = options.height;
           this.previousX = 0;
464           this.previousY = 0;
           this.color = options.color;
466       }
468 Platform.prototype = new Vector2;
470 Platform.prototype.draw = function() {
           InfinityRun.fillStyle = this.color;
472           InfinityRun.fillRect(this.x, this.y, this.width, this ↵
               .height);
           };
474
           // ————— Platform Manager —————
476 function PlatformManager() {
           this.maxDistanceBetween = 300;

```

```

478         this.colors = ['#3D494F'];

480         //first 3 Platforms execept the Starter Platform
this.first = new Platform({
482             x: 300,
             y: 600,
484             width: 400,
             height: 70
486         })
this.second = new Platform({
488             x: (this.first.x + this.first.width) + random(▷
                this.maxDistanceBetween - 150, this.▷
                maxDistanceBetween),
             y: 570, //y: random(this.first.y - 128, ▷
                InfinityRun.height - 80),
490             width: 400,
             height: 70
492         })
this.third = new Platform({
494             x: (this.second.x + this.second.width) + random(▷
                this.maxDistanceBetween - 150, this.▷
                maxDistanceBetween),
             y: 540, //y: random(this.second.y - 128, ▷
                InfinityRun.height - 80),
496             width: 400,
             height: 70
498         })

500     this.first.height = this.first.y + InfinityRun.height ▷
        ;
    this.second.height = this.second.y + InfinityRun.▷
        height;
502     this.third.height = this.third.y + InfinityRun.height ▷
        ;
    this.first.color = randomChoice(this.colors);
504     this.second.color = randomChoice(this.colors);
    this.third.color = randomChoice(this.colors);

506
    this.colliding = false;

508
    this.platforms = [this.first, this.second, this.third ▷
        ];
510 }

512 PlatformManager.prototype.update = function() {

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

```

```

516     this.first.width = random(450, 800);
        this.first.x = (this.third.x + this.third.width) >
            + random(this.maxDistanceBetween - 150, this.>
                maxDistanceBetween);
518     //this.first.y = random(this.third.y - 32, >
        InfinityRun.height - 80);
            this.first.y = random(this.third.y - 32, >
                InfinityRun.height - 200);
520     this.first.height = this.first.y + InfinityRun.>
        height + 10;
        this.first.color = randomChoice(this.colors);
522 }

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

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

544 };

546

548 // reset

```

```

PlatformManager.prototype.updateWhenLose = function() {
550     this.first.x = 300;
552     this.first.color = randomChoice(this.colors);
        this.first.y = 700;
554     //this.first.y = InfinityRun.width / random(2, 3);
        this.second.y = 650;
556     this.third.y = 600;
        this.second.x = (this.first.x + this.first.width) +
        random(this.maxDistanceBetween - 150, this.
        maxDistanceBetween);
558     this.third.x = (this.second.x + this.second.width) +
        random(this.maxDistanceBetween - 150, this.
        maxDistanceBetween);

560 };

562 // ————— Particle System ————— (Sketch Docs)
    function Particle(options) {
564         this.x = options.x;
        this.y = options.y;
566         this.size = 10;
        this.velocityX = options.velocityX || random(-(
        InfinityRun.acceleration * 3) + -8, -(InfinityRun.
        acceleration * 3));
568         this.velocityY = options.velocityY || random(-(
        InfinityRun.acceleration * 3) + -8, -(InfinityRun.
        acceleration * 3));
        this.color = options.color;
570     }

572 Particle.prototype.update = function() {
        this.x += this.velocityX;
574         this.y += this.velocityY;
        this.size *= 0.89;
576     };

578 Particle.prototype.draw = function() {
        InfinityRun.fillStyle = this.color;
580         InfinityRun.fillRect(this.x, this.y, this.size, this.
        size);
    };
582
    /*****
584 InfinityRun.setup = function() {

586     this.jumpCount = 0;
        this.acceleration = 0;

```

```

588     this.accelerationTweening = 0;
        this.player = new Player({
590         x: 150,
           y: 30,
592         width: 32,
           height: 32
594     });

596         setTimeout(function (){
            playMenuFX("MainMenu");
598
            }, 200);

600

602     this.platformManager = new PlatformManager();

604     this.particles = [];
        this.particlesIndex = 0;
606     this.particlesMax = 20;
        this.collidedPlatform = null;
608     this.scoreColor = '#181818';
        this.jumpCountRecord = 0;
610     //-----
        var i, results;
612     i = 3;
        results = [];
614     while (i--) {
        results.push(Town.push(new Street({
616         layer: i + 1,
        width: {
618         min: (i + 1) * 20,
            max: (i + 1) * 50
620         },
        height: {
622         min: InfinityRun.height - 200 - (i * round(▷
            InfinityRun.height/3)),
            max: InfinityRun.height - 50 - (i * round(▷
            InfinityRun.height/3))
624         },
        speed: (i + 1) * .003,
626         color: 'hsl( 200, ' + (((i + 1) * 1) + 10) + '%', ▷
            ' + (75 - (i * 13)) + '% )'
        ))));
628     }

        return results;
630     //-----
632

```



```

};
634 InfinityRun.clear = function() {
636     return InfinityRun.clearRect(0, 0, InfinityRun.width,
        InfinityRun.height);
    };
638 Array.max = function( array ){
640     return Math.max.apply( Math, array );
    };
642 var sc = 0;
644 var sx = 0;
    var sy = 0;
646 var sz = 0;
    var invertRunning = false;
648 var sunsetRunning = false;
    timer = setInterval(function() {
650     if (!playTimer) return;
        ms += 1;
652         sc += 1;
            sy += 1;
654             sz += 1;
                if (sc == 99) {
656                     s+=1;
                        sx+=1;
658                         sc = 0;
                }
660         updateTimer();
662     }, 1);
664 function randomIntFromInterval(min,max)
666 {
        var milliseconds = new Date().getMilliseconds();
668     return Math.floor(Math.random()*(max-min+1)+min);
    }
670
672 var rng = random(115,124);
    var rng2 = random(13,16)
674 function updateTimer() {
    if (s==rng) {
676         if(!invertRunning) {
            invertRunning = true;
678             rng = random(30,50);
            if(setFilters) //toggle filters

```

```

680         {
681             qs.classList.toggle('invertFilter ');
682         }
683     }
684     s=0;
685 }
686 if (sx==rng2) {
687     if(!sunsetRunning) {
688         sunsetRunning = true;
689         rng2 = random(2,5);
690         if(setFilters) //toggle filters
691         {
692             qs.classList.toggle('sunsetFilter ');
693         }
694     }
695     sx=0;
696 }
697 if (sz==70) {
698     invertRunning = false;
699     sz = 0;
700 }
701 if (sy==70) {
702     invertRunning = false;
703     sy = 0;
704 }
705
706     timePassed = ms;
707 }
708
709 function toggleTimer() {
710     if (!playTimer) {
711         //s = 0, ms = 0;
712
713         updateTimer();
714     }
715     playTimer = !playTimer;
716 }
717
718 var difficulty = 0;
719
720 InfinityRun.update = function() {
721     if (GameState == State.Started) {
722         //clear func bg
723         var i, results;
724         dt = InfinityRun.dt < .1 ? .1 : InfinityRun.dt / 16;
725         dt = dt > 5 ? 5 : dt;
726         i = Town.length;

```

```
728     results = [];  
729     while (i--) {  
730         results.push(Town[i].update(i));  
731     }  
732  
733     if(document.hasFocus()) {  
734         toggleTimer();  
735     } else {  
736         toggleTimer();  
737     }  
738  
739  
740     this.player.update();  
741     if(difficulty == 0) {  
742  
743         playbgFX("Main1");  
744         difficulty = 1;  
745     } else if (timePassed > 1000 && timePassed < 5000 &&  
746         difficulty == 1) {  
747         this.accelerationTweening = 1.5;  
748         this.platformManager.maxDistanceBetween = 430;  
749  
750         bgFX.stop();  
751         playbgFX("Main2");  
752         playSFX("LevelUP");  
753  
754         difficulty = 2;  
755     } else if (timePassed > 5000 && timePassed < 10000 &&  
756         difficulty == 2) {  
757         this.accelerationTweening = 2.7;  
758         this.platformManager.maxDistanceBetween = 530;  
759  
760         bgFX.stop();  
761         playbgFX("Main3");  
762         playSFX("LevelUP");  
763  
764         difficulty = 3;  
765     } else if (timePassed > 10000 && timePassed < 15000 &&  
766         difficulty == 3) {  
767         this.accelerationTweening = 3.8;  
768         this.platformManager.maxDistanceBetween = 580;  
769  
770         bgFX.stop();  
771         playbgFX("Main4");  
772         playSFX("LevelUP");  
773  
774         difficulty = 4;  
775     } else if (timePassed > 15000 && timePassed < 20000 &&
```

```

    && difficulty == 4) {
        this.accelerationTweening = 4.4;
774         this.PlatformManager.maxDistanceBetween = 610;

776         playSFX("LevelUP");

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

782

784         playSFX("LevelUP");

786         difficulty = 6;
    }
788     this.acceleration += (this.accelerationTweening -
        this.acceleration) * 0.01;

790     for (i = 0; i < this.platformManager.platforms.length;
        ; i++) {
        if (this.player.intersects(this.platformManager.
            platforms[i])) {
792             this.collidedPlatform = this.platformManager.
                platforms[i];
            if (this.player.y < this.platformManager.
                platforms[i].y) {
794                 this.player.y = this.platformManager.
                    platforms[i].y;

796                 // Gravity after Collision with Platform
                this.player.velocityY = 0;

798             }

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

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

808             if (this.player.intersectsLeft(this.
                platformManager.platforms[i])) {

```

```

810         this.player.x = this.collidedPlatform.x ->
            64;
            for (i = 0; i < 10; i++) {
812             // SpawnParticles @PlayerPostion with >
                intersecting Platform Color
                this.particles[(this.particlesIndex >
                    ++)% this.particlesMax] = new >
                    Particle({
814                     x: this.player.x + this.player.>
                        width,
                        y: random(this.player.y, this.>
                            player.y + this.player.height)>
                        ,
816                     velocityY: random(-30, 30),
                        color: randomChoice(['#181818', >
                            '#181818', this.>
                                collidedPlatform.color])
                    });
818             });
            };

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

824
                if (timePassed > this.>
                    jumpCountRecord) {
826                     this.jumpCountRecord = timePassed >
                        ;
                    }
            } else {

828
                // ----- Controller -----
                // dragging: Mouse click & touch support
832                 if (this.dragging || this.keys.SPACE || >
                    this.keys.UP || this.keys.W) {
                    this.player.velocityY = this.player.>
                        jumpSize;
834                     this.jumpCount++;

                        playSFX("Jump");
836
                    }
838                 }
            }
840         };

842         for (i = 0; i < this.platformManager.platforms.length >

```

```

        ; i++) {
            this.platformManager.update();
844    };

846    for (i = 0; i < this.particles.length; i++) {
        this.particles[i].update();
848    };

850    //bg
    return results;
852 }

854 };

856

858

860 var selectedItem = 0;
var audioItem = 10;

862 InfinityRun.keydown = function() {
    if (InfinityRun.keys.ESCAPE && GameState==State.↳
        Started) {
864         InfinityRun.clear();
        GameState = State.Menu;

866         bgFX.setPaused(true);
        //playMenuFX("MainMenu");
868         menuFX.setPaused(false);
        toggleTimer();
870

872         freshStart = 1;

874     } else if (InfinityRun.keys.ESCAPE && GameState==↳
        State.Menu && curMenuTab==MenuTab.Main && ↳
        freshStart == 1) {
        GameState = State.Started;
876         toggleTimer();
        //menuFX.stop();
878         bgFX.setPaused(false);
        menuFX.setPaused(true);

880     } else if (InfinityRun.keys.ESCAPE && GameState==↳
        State.Menu && curMenuTab==MenuTab.Controlls) {
882         curMenuTab = MenuTab.Main;
    } else if (InfinityRun.keys.ESCAPE && GameState==State.↳
        .Menu && curMenuTab==MenuTab.Settings) {
884         curMenuTab = MenuTab.Main;

```

```

    } else if (InfinityRun.keys.ESCAPE && GameState==>
        State.Menu && curMenuTab==MenuTab.Highscore) {
886         curMenuTab = MenuTab.Main;
    } else if (InfinityRun.keys.ESCAPE && GameState==>
        State.Menu && curMenuTab==MenuTab.Credits) {
888         curMenuTab = MenuTab.Main;
    }

890
    //main menu controls
892    if (InfinityRun.keys.UP && GameState == State.>
        Menu) {
        selectedItem = (selectedItem + items.>
            length - 1) % items.length;
894    }
    if (InfinityRun.keys.DOWN && GameState == State.>
        Menu) {
896        selectedItem = (selectedItem + 1) % items.>
            length;
    }
898    //general settings choose
    if (InfinityRun.keys.UP && curMenuTab==MenuTab.>
        Settings && settingsItem!=0) {
900        settingsItem -=1;
    }
902    if (InfinityRun.keys.DOWN && curMenuTab==MenuTab.>
        Settings && settingsItem!=2) {
        settingsItem +=1;
904    }
    // settings audio change
906    if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.>
        Settings && audioltem !=0 && settingsItem ==0)>
    {
        audioltem = (audioltem + items.length - >
            1) % items.length;
908        createjs.Sound.volume -= 0.1;
    }
910
    if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab.>
        .Settings && audioltem !=10 && settingsItem >
        ==0) {
912        audioltem = (audioltem + 1) % items.>
            length;
        createjs.Sound.volume += 0.1;
914    }
    //graphic settings change
916    if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.>
        Settings && vgaquality!=0 && settingsItem ==1)>
    {

```

```

        vgaquality -=1;
918     }

920     if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab.
        .Settings && vgaquality!=2 && settingsItem >
        ==1) {
        vgaquality+=1;
922     }
    //filter settings change
924     if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.
        Settings && !setFilters && settingsItem ==2) {
        setFilters=true;
926     }

928     if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab.
        .Settings && setFilters && settingsItem ==2) {
        setFilters=false;
930         if (invertRunning) {
            qs.classList.toggle('invertFilter >
                ');
932         }
        if (sunsetRunning) {
934             qs.classList.toggle('sunsetFilter >
                ');
        }
936     }
    if (InfinityRun.keys.ENTER && GameState == State.>
        Menu) {
938         callback(selectedItem);
    }

940 }
942 Menu = function() {
944 }

946 InfinityRun.draw = function() {

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

956     }
    }
}

```



```

958     if (vgaquality==1)
959     {
960         i-=1; //only upper towers will be drawn
961         while (i-->0) {
962             results.push(Town[i+1].render(i+1));
963
964         }
965     }
966     if (vgaquality<1)
967     {
968         i-=2; //only one street will be drawn
969         while (i-->0) {
970             results.push(Town[i+1].render(i+1));
971
972         }
973     }
974
975     this.player.draw();
976
977     for (i = 0; i < this.platformManager.platforms.length; i++) {
978         this.platformManager.platforms[i].draw();
979     };
980
981     //Draw particles
982     for (i = 0; i < this.particles.length; i++) {
983         this.particles[i].draw();
984     };
985
986     //draw score
987     this.font = '30pt Bungee';
988     this.textAlign="left";
989     this.fillStyle = '#494949';
990     this.fillText('Score: ', this.width - 330, 65);
991     this.fillText(timePassed, this.width - 170, 65);
992
993     /*
994     * Main Menu
995     */
996     } else if (GameState == State.Menu && curMenuTab == MenuTab.Main) {
997
998         this.title = "InfinityRun";
999         items = ["Play", "Controls", "Settings", "
1000             Highscore", "Credits"];
1001
1002

```

```

callback = function(numItem) { //if (numItem == 0)
    GameState=State.Started
1004
switch (numItem) {
1006     case 0:
        GameState=State.Started;
1008         toggleTimer();
        //bgFX.stop();
1010         menuFX.setPaused(true);
        bgFX.setPaused(false);
1012         break;
        case 1:
1014             curMenuTab=MenuTab.Controls;
            break;
1016         case 2:
            curMenuTab=MenuTab.Settings;
1018             break;
        case 3:
1020             curMenuTab=MenuTab.Highscore;
            break;
1022         case 4:
            curMenuTab=MenuTab.Credits;
1024             break;
1026     }

1028 };
this.height = InfinityRun.height;
1030 this.width = InfinityRun.width;
this.size = 70;
1032

var lingrad = this.createLinearGradient(0,0,0,
    this.height);
1034 lingrad.addColorStop(0, '#000');
lingrad.addColorStop(1, '#023');
1036 this.fillStyle = lingrad;
this.fillRect(0,0,this.width, this.height)

1038
this.textAlign = "center";
1040 this.fillStyle = "White";

1042
var height = 100;
//logo
1044 this.drawImage(bglogo, this.width-500, this.height-
    -300);
//
1046 if (this.title) {
        this.font = Math.floor(this.size*1.3).

```

```

1048         toString() + "px_Bungee";
        this.fillText(this.title, this.width/2, height);
        height+= height;
1050     }

1052     for (var i = 0; i < items.length; ++i)
    {
1054         var size = Math.floor(this.size*0.8);
        if (i == selectedItem)
1056         {
            this.fillStyle = "#A9F5F2";
            size = this.size+5;
1058         }
        this.font = size.toString() + "px_Bungee";
        ;
        height += this.size;
1062         this.fillText(items[i], InfinityRun.width/2, height);
        this.fillStyle = "White";
1064     }
    return results;
1066

    /*
1068     * Settings Tab
    *
1070     */
    }else if (GameState == State.Menu && curMenuTab==
        MenuTab.Controlls){
1072         this.title = "Controlls";
        items = highScore;
1074

1076         callback = function(volume) { //if (numItem == 0)
            GameState=State.Started

1078         switch (volume) {

1080         }

1082

1084     };
    this.height = InfinityRun.height;
1086     this.width = InfinityRun.width;

1088     var lingrad = this.createLinearGradient(0,0,0,
        this.height);

```

```

1090    lingrad.addColorStop(0, '#000');
1091    lingrad.addColorStop(1, '#023');
1092    this.fillStyle = lingrad;
1093    this.fillRect(0,0,this.width, this.height, items[
        i]);

1094    this.textAlign = "center";
1095    this.fillStyle = "White";

1096
1097    var width = 10;
1098    var height = 150;

1099
1100    if (this.title) {
1101        this.font = Math.floor(this.size*1.3).
            toString() + "px_Bungee";
1102        this.fillText(this.title, this.width/2,
            150);
1103        height+= height;
1104    }
1105    var distanceText = 50
1106    this.font = Math.floor(40).toString() + "px_
        Bungee";
1107    this.textAlign = "left";
1108    //Names
1109    this.fillText("Menu:", this.width/5, 300);
1110    this.font = Math.floor(20).toString() + "px_
        Bungee";
1111    this.fillText("[ESC]_Menu", this.width/5, 300+
        distanceText);
1112    this.fillText("[Arrow_up/down]_To_navigate",
        this.width/5, 300+2*distanceText);
1113    this.fillText("[Enter]_Accept", this.width/5,
        300+3*distanceText);

1114
1115    this.font = Math.floor(40).toString() + "px_
        Bungee";
1116    this.textAlign = "left";
1117    //Names
1118    this.fillText("Game:", this.width/5, 300+5*
        distanceText);
1119    this.font = Math.floor(20).toString() + "px_
        Bungee";
1120    this.fillText("[ESC]_Menu", this.width/5,
        300+6*distanceText);
1121    this.fillText("[W],_ [Arrow_up],_ [Leertaste]_
        Jump", this.width/5, 300+7*distanceText);
1122    this.fillText("[Arrow_down]_increase_falling_
        speed", this.width/5, 300+8*distanceText);

```

```

    }
1124     else if (GameState == State.Menu && curMenuTab==>
           MenuTab.Settings){

1126         this.title = "Settings";
         items = [0,10, 20, 30, 40, 50, 60, 70, 80, 90 , >
           100];

1128         callback = function(volume) { //if (numItem == 0)>
           GameState=State.Started

1130         switch (volume) {

1132         }

1134

1136

1138         };

1140         this.height = InfinityRun.height;
         this.width = InfinityRun.width;

1142         var lingrad = this.createLinearGradient(0,0,0,>
           this.height);
         lingrad.addColorStop(0, '#000');
1144         lingrad.addColorStop(1, '#023');
         this.fillStyle = lingrad;
1146         this.fillRect(0,0,this.width, this.height, items[>
           i]);

1148         this.textAlign = "center";
         this.fillStyle = "White";

1150

1152         var width = 10;
         var height = 10;
         var posx = 130;
1154         var posy = 380;
         this.space = 15;
1156         this.heightincr = 4;

1158         if (this.title) {
           this.font = Math.floor(this.size*1.3).>
             toString() + "px_Bungee";
1160           this.fillText(this.title, this.width/2,>
             150);
           height+= height;

1162         }

```

```

1164     this.font = "55px_Bungee";
1165     if (settingsItem==0) {
1166         this.fillStyle = "#A9F5F2";
1167     }
1168     this.fillText('Volume', 240, 300);

1170     this.fillStyle = "White";

1172
1173     for (var i = 0; i < items.length; ++i) {
1174         var size = Math.floor(this.size*0.8);
1175         if (i == audioltem && settingsItem==0)
1176         {
1177             this.fillStyle = "#A9F5F2";
1178             //size = this.size+5;
1179         }
1180         this.font = size.toString() + "px_Bungee" >
1181             ;
1182         posx += this.space;
1183         posy -= this.heightincr;
1184         height += this.heightincr;

1185         items[i] = this.fillRect(posx, posy, width, >
1186             height);

1187         //this.fillText(items[i], InfinityRun.>
1188             width/2, height);
1189         this.fillStyle = "White";

1190     }

1191
1192     //Graphic Settings
1193     this.fillStyle = "White";
1194     if (settingsItem==1) {
1195         this.fillStyle = "#A9F5F2";
1196     }
1197     this.fillText('Graphics', 240, 500);
1198     this.fillStyle = "White";
1199     switch (vgaquality) {
1200         //Low
1201         case 0:
1202             this.fillText('Mid', 500,600 );
1203             this.fillText('High', 700, 600);
1204             if (settingsItem == 1){
1205                 this.fillStyle = "#A9F5F2";
1206             }
1207             this.fillText('Low', 240, 600);
1208             break;

```

```

//mid
1210     case 1:

1212         this.fillText('Low', 240, 600);
1213         this.fillText('High', 700, 600);
1214         if (settingsItem == 1){
1215             this.fillStyle = "#A9F5F2";
1216         }
1217         this.fillText('Mid', 500,600 );
1218
1219         break;
//high
1220     case 2:
1221         this.fillText('Mid', 500,600 );
1222         this.fillText('Low', 240, 600);
1223         if (settingsItem == 1){
1224             this.fillStyle = "#A9F5F2";
1225         }
1226         this.fillText('High', 700, 600);
1227
1228         break;
1229
1230 }
1231 //Filter settings
1232 this.fillStyle = "White";
1233 if (settingsItem==2) {
1234     this.fillStyle = "#A9F5F2";
1235 }
1236 this.fillText('Filters ', InfinityRun.width-300, ↵
    300);
1237 this.fillStyle = "White";
1238 if(setFilters)
1239 {
1240     this.fillText('Off ', InfinityRun.width-200, ↵
        400);
1241     if (settingsItem==2)
1242     {
1243         this.fillStyle = "#A9F5F2";
1244     }
1245 this.fillText('On', InfinityRun.width-350, 400);
1246 }
1247 else
1248 {
1249     this.fillText('On', InfinityRun.width-350, ↵
        400);
1250 if (settingsItem==2)
1251 {
1252     this.fillStyle = "#A9F5F2";

```

```

1254     }
    this.fillText('Off', InfinityRun.width-200, 400);
1256 }

1258 //-----

    /*
1260     * Highscore Tab
    *
1262     */

1264 } else if (GameState == State.Menu && curMenuTab >
    == MenuTab.Highscore) {

1266     this.title = "Highscore";
1268     items = highScore;

1270     callback = function(volume) { //if (numItem == 0) >
        GameState=State.Started

1272

1274     switch (volume) {

1276     }

1278 };
1280 this.height = InfinityRun.height;
    this.width = InfinityRun.width;

1282     var lingrad = this.createLinearGradient(0,0,0,>
        this.height);
1284     lingrad.addColorStop(0, '#000');
    lingrad.addColorStop(1, '#023');
1286     this.fillStyle = lingrad;
    this.fillRect(0,0,this.width, this.height, items[>
        i]);

1288

1290     this.textAlign = "center";
    this.fillStyle = "White";

1292     var width = 10;
    var height = 100;

1294

1296     if (this.title) {

```



```

        this.font = Math.floor(this.size*1.3).toString() + "px Bungee";
1298     this.fillText(this.title, this.width/2, height+50);
        height+= height;
1300     }

1302     var rank = 1;

1304     for (var i = 0; i < items.length; ++i)
    {
1306         var size = Math.floor(this.size*0.8);
1308         if (i == selectedItem)
        {
1310             this.fillStyle = "#A9F5F2";
            size = this.size+5;
1312         }
        this.font = 0.6*size.toString() + "px Bungee";
1314         height += 50;
        this.fillText(rank + ". " + items[i], InfinityRun.width/2, height);
1316         this.fillStyle = "White";

        rank++;
1318     }
1320 }
1322 // Credits Menu
    else if (GameState == State.Menu && curMenuTab == MenuTab.Credits) {
1324
        this.title = "Credits";
1326         items = highScore;

1328         callback = function(volume) { //if (numItem == 0) GameState=State.Started

1330             switch (volume) {

1332             }

1334

1336         };
        this.height = InfinityRun.height;
1338         this.width = InfinityRun.width;

```

```

1340     var lingrad = this.createLinearGradient(0,0,0, ▷
        this.height);
    lingrad.addColorStop(0, '#000');
1342     lingrad.addColorStop(1, '#023');
    this.fillStyle = lingrad;
1344     this.fillRect(0,0,this.width, this.height, items[▷
        i]);

1346     this.textAlign = "center";
    this.fillStyle = "White";

1348

    var width = 10;
1350     var height = 150;

1352

    if (this.title) {
1354         this.font = Math.floor(this.size*1.3). ▷
            toString() + "px_Bungee";
        this.fillText(this.title, this.width/2, ▷
            150);
1356         height+= height;
    }

1358     var distanceText = 50
    this.font = Math.floor(50).toString() + "px_▷
        Bungee";
1360     this.textAlign = "left";
    //Names
1362     this.fillText("Group_Members:", this.width/5, ▷
        300);
    this.font = Math.floor(40).toString() + "px_▷
        Bungee";
1364     this.fillText("__Florian_Durli", this.width/5, ▷
        300+distanceText);
    this.fillText("__Koray_Emtekin", this.width/5, ▷
        300+2*distanceText);
1366     this.fillText("__Jannik_Ivosevic", this.width/5, ▷
        300+3*distanceText);
    this.fillText("__Marco_Mayer", this.width/5, ▷
        300+4*distanceText);
1368     this.fillText("__Johannes_But", this.width/5, ▷
        300+5*distanceText);
    //bottom info
1370     this.font = Math.floor(15).toString() + "px_▷
        _New_Roman";
    this.textAlign = "center";
1372     distanceText = 20;
    this.fillText("InfinityRun_is_a_nonprofit_▷

```

```

        students_project_at_"Hochschule Furtwangen"
        /\ "Furtwangen University.\ "Special_thanks_to_
        \ "Soulwire\ "for_his_Sketch.js_Minimal_
        JavaScript_Creative_Coding_Framework", this.
        width/2, this.height-2.2*distanceText);
1374 this.fillText("Sounds: freesounds.org Special_
        thanks_to_Jack_Rugil_for_his_Parrallax_Skyline_
        ", this.width/2, this.height-distanceText-5);
        this.fillText("2016", this.width/2, this.height-8);
        ;
1376     }

1378 //Debug
    if (debug) {
1380         this.font = '16pt Arial';
        this.fillStyle = '#181818';
1382         this.fillText('Record: ' + s + " " + sc/*this.
            jumpCountRecord*/, this.width - 150, 33);
        this.fillStyle = this.scoreColor;
1384         this.fillText('Jumps: ' + this.jumpCount, this.
            width - 150, 50);
    }

1386 };

1388 InfinityRun.resize = function() {
1390     /* todo Windowscale optimization
        *
1392     *
        */
1394 }

```

A.2. game.css

```

body{
2   background: #e3e3e3;
    overflow: hidden;
4   margin: 0;
    padding: 0;
6   text-align: center;
}
8 #container{
    /*margin-top: 10%;*/
10    display: inline-block;
}

12 canvas{
14    font-family: 'Bungee', cursive;
    background: #cecece;

```

```

16  border: 1px solid #181818;
17  }
18
19  canvas.sunsetFilter {
20    -webkit-animation: sunset-animation 70s;
21  }
22
23  canvas.invertFilter {
24    -webkit-animation: invert-animation 20s;
25  }
26
27
28  @-webkit-keyframes sunset-animation {
29    0% {
30      -webkit-filter: sepia(0) saturate(2);
31    }
32
33    50% {
34      -webkit-filter: sepia(1) saturate(15);
35    }
36
37    100% {
38      -webkit-filter: sepia(0) saturate(2);
39    }
40  }
41
42
43
44  @-webkit-keyframes invert-animation {
45    0% {
46      -webkit-filter: invert(0);
47    }
48
49    50% {
50      -webkit-filter: invert(.8);
51    }
52
53    100% {
54      -webkit-filter: invert(.0);
55    }
56  }

```

A.3. 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>
      <script type="text/javascript" src="js/soundjs␣
        -0.6.2.min.js" charset="utf-8"></script>
7      <link href="https://fonts.googleapis.com/css?␣
        family=Bungee" rel="stylesheet">

9      <title>Infinity Run</title>

11      <link href="css/game.css" rel="stylesheet" type="text␣
        /css">
        <link rel="shortcut␣icon" type="image/x-icon" ␣
          href="image/favicon.png">
13 </head>
    <body onload="loadSound();">
15      <script>
17 // internal load sound files (called in onload (body))
    function loadSound() {
19     if (!createjs.Sound.initializeDefaultPlugins()) {␣
        return;}

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

33     ];

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

41 </div>
    <audio id="backgroundmusic" ></audio>
43 <audio id="fxaudio" ></audio>

```

```
<script type="text/javascript" src="js/game.js" charset="utf-8"></script>
45 </body>
</html>
```