



Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent : **Gabriela Mai**
Vorgelegt am : 14. Dezember 2016
Vorgelegt von : Gruppe 4

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

Inhaltsverzeichnis

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

2.2.4	Bibliothek	10
2.2.5	Code	10
2.2.6	Nächste Ziele	16
2.3	Implementation - Endstand	16
2.3.1	Spielkonzept Änderungen	16
2.3.2	Funktionsdiagramm	16
2.3.3	Grafiken	18
2.3.4	Code Änderungen	19
2.3.5	Das Spiel - Endstand	23
2.3.6	Sounds	24
	Literaturverzeichnis	25
	Eidesstattliche Erklärung	27
A	Anhang	29
A.1	Github Changelog	29
A.2	game.js	32
A.3	game.css	60
A.4	index.html	60

Abbildungsverzeichnis

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

Tabellenverzeichnis

Tabelle 1: Rollenverteilung	2
Tabelle 2: Phase 1: Entwurf und Anforderungen	6
Tabelle 3: Phase 2: Implementierung	6
Tabelle 4: Phase 3: Test	6
Tabelle 5: Phase 4: Dokumentation und Präsentation	7
Tabelle 6: Releaseplan	7
Tabelle 7: Funktionsbeschreibung	18
Tabelle 8: Sound Links	24
Tabelle 9: Github Namen	29

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

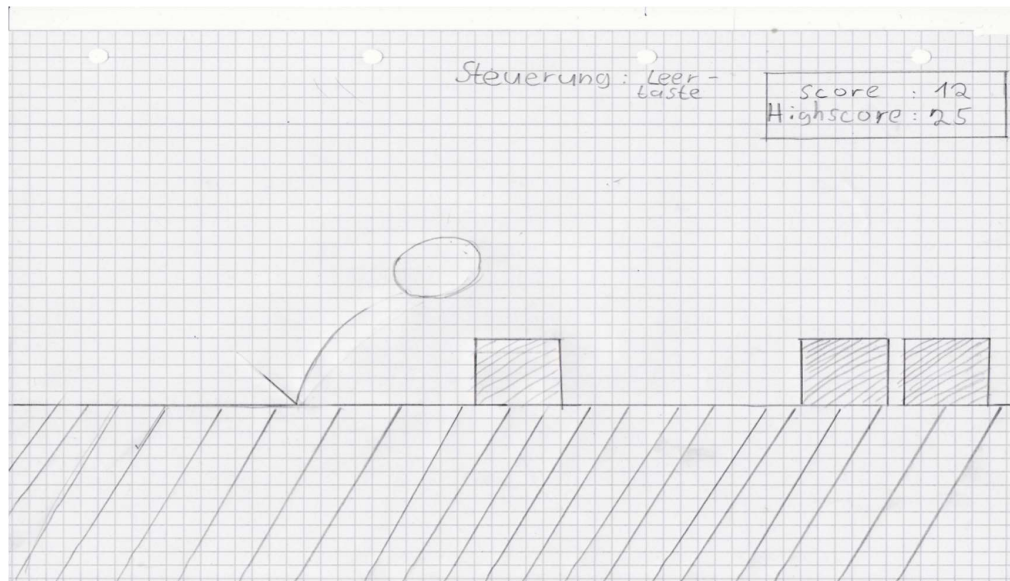


Abbildung 6.: Entwurfsskizze

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 Performance 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.

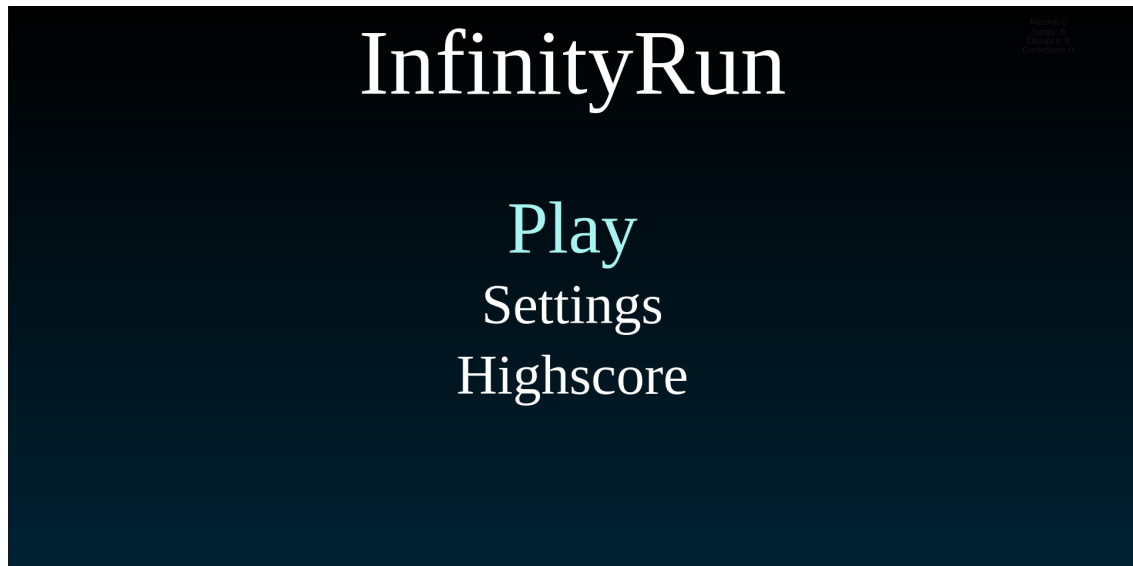


Abbildung 7.: Startbildschirm



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.
```



```

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

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

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

```

2.2.6. Nächste Ziele

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

2.3. Implementation - Endstand

2.3.1. Spielkonzept Änderungen

Folgende Spielkonzept Änderungen haben wir im laufe der Implementation vorgenommen:

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

2.3.2. Funktionsdiagramm

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.

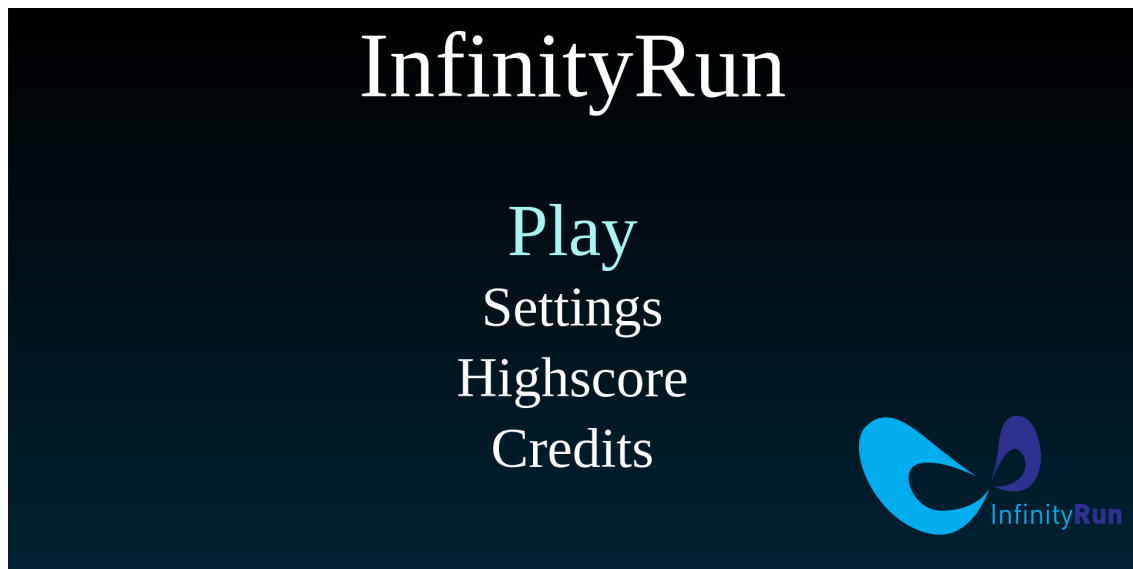


Abbildung 11.: Startbildschirm - Endstand



Abbildung 12.: Das Spiel - Endstand

2.3.6. Sounds

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

Folgende Sounds werden wir verwenden:

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

Tabelle 8.: Sound Links

Literaturverzeichnis

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

Eidesstattliche Erklärung

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

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

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

FURTWANGEN, den 14. Dezember 2016 Florian Durli

FURTWANGEN, den 14. Dezember 2016 Jannik Ivosevic

FURTWANGEN, den 14. Dezember 2016 Johannes But

FURTWANGEN, den 14. Dezember 2016 Marco Mayer

FURTWANGEN, den 14. Dezember 2016 Koray Emtekin

A. Anhang

A.1. Github Changelog

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

```
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 9.: Github Namen

Changelog:

```
1 2d7d953 butjo      2016-12-12 Favicon und Logo im Menü
3 2b8a139 ans77      2016-12-12 Logos hinzugefügt
5 92b7b28 Slay3r      2016-12-12 Code Cleanup
7 30b0c6f Slay3r      2016-12-12 Clean up
9 37f6e94 butjo      2016-12-12 Favicon hinzugefügt
11 35801ac butjo      2016-12-11 Hintergrund ist nun abhän
    nig von der Fenstergröße und reagiert auf den Player
    nicht mehr auf die Maus sowie noch die Dacharten
    erweitert und bugs behoben
13 8d06d8a butjo      2016-12-09 skyline eingefügt
15 4e52883 Slay3r      2016-12-07 Doku + Alte Dateien
17 2c3491b butjo      2016-12-05 Sounds hinzugefügt
    teilweise noch ein paar Bugs und dirty code
```

- 19 2e6dfbe butjo 2016-11-30 Dirty Code mit versuch ↵
den background zu implementieren files sind mit prefix ↵
back gekennzeichnet und liegen im Hauptverzeichnis.
- 21 dda171b Slay3r 2016-11-30 Changelog geändert
- 23 ffe660b Slay3r 2016-11-30 Ausblick/Fazit ↵
auskommentiert
- 25 01e309a Slay3r 2016-11-30 Sounds
- 27 3a05368 Slay3r 2016-11-30 Neue Dokumentation ↵
hinzufügen
- 29 7c3596c Slay3r 2016-11-30 Neue Dokumentation ↵
hinzufügen
- 31 bfdb05c Slay3r 2016-11-30 entfernen der ↵
Dokumentation
- 33 f944707 r4qtor 2016-11-25 Querlesung – Marco
- 35 8b6bdde Florian Durli 2016-11-25 Abgabe
- 37 bc8d933 Florian Durli 2016-11-25 Code Cleanup für Doku
- 39 b6d7a09 butjo 2016-11-24 Rechtschreibkorrekturen
- 41 1faa558 r4qtor 2016-11-24 Delete phasen.tex root/
- 43 51f4a79 r4qtor 2016-11-24 updated phasen.tex
- 45 62af4b8 Krusher999 2016-11-24 Letzten Codes ↵
geschrieben
- 47 93b8965 Florian Durli 2016-11-23 fix
- 49 9027301 Florian Durli 2016-11-23 Changelog finale Lösung
- 51 1392138 Florian Durli 2016-11-23 Jojos Description [in](#) ↵
LaTeX
- 53 25ff037 butjo 2016-11-23 Beschreibung der ↵
Codeteile [in](#) der phasen.tex von Johannes
- 55 1b2348c Florian Durli 2016-11-23 Changelog Additionally

57	cf467dc	Florian Durli	2016-11-23	Changelog	Additionally
59	6db1ad6	butjo	2016-11-23	Merge branch 'master' of	https://github.com/Slay3r/InfinityRun
61	b924713	Slay3r	2016-11-23	Generated	Changelog
63	69dd747	butjo	2016-11-23	Merge branch 'master' of	https://github.com/Slay3r/InfinityRun
65	203ae2e	Slay3r	2016-11-23	Bilder des Spiels	
67	d274cfc	Slay3r	2016-11-23	Anforderungen	
69	a0909ac	Slay3r	2016-11-23	Literaturverzeichnis	
71	19e2f3d	Slay3r	2016-11-23	Doku update	
73	21889f9	Florian Durli	2016-11-22	Add	Changelog
75	743c95a	butjo	2016-11-16	Formatierte	sketch.min.js
77	d64d254	butjo	2016-11-16	Endlose	Schwierigkeitserhöhung
79	3707b94	butjo	2016-11-16	Präsentation	Zwischenstand
81	f23c9be	Slay3r	2016-11-16	Präsentation ü	bearbeitet
83	53aa72e	butjo	2016-11-16	Präsentation und	test
85	b5cb978	Florian Durli	2016-11-16	Merge pull request	#1 from r4qtor/master
87	275bd69	r4qtor	2016-11-15	tiny cleanup & incl.	menu
89	11a5e55	Slay3r	2016-11-09	Basis implementation	
91	c783850	Slay3r	2016-11-09	Bilder hinzugefügt	
93	d88d0d2	Slay3r	2016-11-09	Dokumentations	Basis
95	39b4705	Florian	2016-10-19	Initial Struktur der	Ordner und files

97 093797e Florian Durli 2016-10-19 Delete Requirements
 99 56c4aae Florian 2016-10-19 doc Ordner
 101 b3b83fd Florian 2016-10-19 Requirements hinzugefügt
 103 b4d2627 Florian Durli 2016-10-19 Initial commit

A.2. game.js

```

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



```

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

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

52 //----->

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

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

```

```

80         {
            // Fail silently but show in F12 ↵
            // developer tools console
82         if(window.console && console.↳
            error("Error:" + e));

        }

84     }
    }

86 // randomizer
88 function random(min, max) {
    return Math.round(min + (Math.random() * (max - min)))↳
    };

90 }

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

94 }

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

104

106 //—————↵

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

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

```

```

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

```

```

    InfinityRun.lineTo(this.x + this.width, ↵
        this.y);
156  if (this.singleroofTopDirection)
    {
158      InfinityRun.lineTo(this.x + this.↵
          width, this.y - this.↵
          singleroofTopHeight);
    }
160  else
    {
162      InfinityRun.lineTo(this.x, this.y↵
          - this.singleroofTopHeight);
    }
164  InfinityRun.closePath();
    InfinityRun.fill();
166  InfinityRun.stroke();
}

168  if (this.normalTop)
170  {
    InfinityRun.beginPath();
    InfinityRun.moveTo(this.x, this.y);
172  InfinityRun.lineTo(this.x + this.width, ↵
        this.y);
174      InfinityRun.lineTo(this.x + (this↵
          .width/2), this.y-this.↵
          normalTopHeight);
    InfinityRun.closePath();
176  InfinityRun.fill();
    InfinityRun.stroke();
178      if (this.normalTopchimney)
    {
180          InfinityRun.beginPath();
          InfinityRun.moveTo(this.x↵
              +(this.width/5), this.↵
              y);
182          InfinityRun.lineTo(this.x↵
              +(this.width/5), this.↵
              y- 0.8*(this.↵
              normalTopHeight));
          InfinityRun.lineTo(this.x↵
              + (this.width/5)+(↵
              this.width/10), this.y↵
              - 0.8*(this.↵
              normalTopHeight));
184          InfinityRun.lineTo(this.x↵
              + (this.width/5)+(↵
              this.width/10), this.y↵

```

```

);
InfinityRun.closePath();
186 InfinityRun.fill();
InfinityRun.stroke();
188 }
}
190 if (this.coneTop)
{
192     InfinityRun.beginPath();
InfinityRun.moveTo(this.x, this.y);
194 InfinityRun.lineTo(this.x + (this.width -
this.coneTopWidth)/2, this.y - this.
coneTopHeight);
if (!this.coneTopeflat)
196 {
InfinityRun.lineTo(this.x +
+(this.width/2), this.y -
y - (this.coneTopHeight
*1.3));
198 }
InfinityRun.lineTo(this.x + ((
this.width - this.coneTopWidth)
/2) + this.coneTopWidth, this.y -
this.coneTopHeight);
200 InfinityRun.lineTo(this.x + this.
width, this.y);
InfinityRun.closePath();
202 InfinityRun.fill();
InfinityRun.stroke();
204 }
206 if (this.companyTop)
{
208     var ctc = 1;
while (ctc <= this.companyTopcount)
210 {
InfinityRun.beginPath();
212 InfinityRun.moveTo(this.x, this.
y);
InfinityRun.lineTo(this.x + ctc * (
this.width / this.
companyTopcount), this.y - this.
companyTopHeight);
214 InfinityRun.lineTo(this.x + ctc * (
this.width / this.
companyTopcount), this.y + this.
companyTopHeight);
InfinityRun.closePath();

```

```

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

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

240 };

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

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

256     this.speed = config.speed;

```

```

258     this.color = config.color;
        this.populate();
260     return this;
    };

262     Street.prototype.populate = function()
264     {
        var newHeight, newWidth, results, totalWidth;
266         totalWidth = 0;
        results = [];
268         while (totalWidth <= InfinityRun.width + (this.
            width.max * 2))
        {
270             newWidth = round(random(this.width.min,
                this.width.max));
            newHeight = round(random(this.height.min,
                this.height.max));
272             this.alltowers.push(new Tower({
                layer: this.layer,
274             x: this.alltowers.length == 0 ? 0 : this.
                alltowers[this.alltowers.length - 1].
                x + this.alltowers[this.alltowers.
                length - 1].width,
                y: InfinityRun.height - newHeight,
276             width: newWidth,
                height: newHeight,
278             color: this.color
            }));
280             results.push(totalWidth += newWidth);
        }
282         return results;
    };

284     Street.prototype.update = function()
286     {
        var firstTower, lastTower, newHeight, newWidth;
288         if (InfinityRun.accelerationTweening==0)
        {
290             this.x-=((150) * this.speed) * dt;
        }
292         else
        {
294             this.x -= ((InfinityRun.accelerationTweening*330) *
                this.speed) * dt;
        }

296         firstTower = this.alltowers[0];
298         if (firstTower.width + firstTower.x + this.x < 0)

```

```

    {
300         newWidth = round(random(this.width.min, ↵
            this.width.max));
            newHeight = round(random(this.height.min, ↵
            this.height.max));
302         lastTower = this.alltowers[this.alltowers ↵
            .length - 1];
            firstTower.reset({
304                 layer: this.layer,
                    x: lastTower.x + lastTower.width,
306                 y: InfinityRun.height - newHeight ↵
                    ,
                    width: newWidth,
308                 height: newHeight,
                    color: this.color
310             });
            return this.alltowers.push(this.alltowers.shift() ↵
            );
312     }
};

314 Street.prototype.render = function()
316 {
    var i;
318     i = this.alltowers.length;
    InfinityRun.save();
320     InfinityRun.translate(this.x, (InfinityRun.height ↵
        - (InfinityRun.height - (-jumpheight*0.5) - 400)) ↵
        / 20 * this.layer);
    while (i--)
322     {
        this.alltowers[i].render(i);
324     }
    return InfinityRun.restore();
326 };
//—————

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

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

```



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

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

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

358     this.previousX = this.x;
360     this.x = x;
362 };
364 // Set Y
366 Vector2.prototype.setY = function(y) {

368     this.previousY = this.y;
370     this.y = y;
372 };
374 // Collision / Intersection Top
376 Vector2.prototype.intersects = function(obj) {

378     if (obj.x < this.x + this.width && obj.y < this.y + this.height &&
380         obj.x + obj.width > this.x && obj.y + obj.height > this.y) {
382         return true;
384     }

386     return false;
388 };
390 // Collision / Intersection Left
392 Vector2.prototype.intersectsLeft = function(obj) {

394     if (obj.x < this.x + this.width && obj.y < this.y + this.height) {
396         return true;
398     }

400 }
```

```

384         return false;
386     };

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

390 function Player(options) {

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

400 }

402 Player.prototype = new Vector2;

404 Player.prototype.update = function() {
405     // Gravity
406     this.velocityY += 1;
407     //um bg zu ändern
408     jumpheight=(this.y);
409     this.setPosition(this.x + this.velocityX, this.y + this.velocityY);

410     if (this.y > InfinityRun.height || this.x + this.width < 0) {
411         this.x = 150;
412         this.y = 50;
413         this.velocityX = 0;
414         this.velocityY = 0;
415         InfinityRun.jumpCount = 0;
416         InfinityRun.acceleration = 0;
417         InfinityRun.accelerationTweening = 0;
418         InfinityRun.scoreColor = '#181818';
419         InfinityRun.platformManager.maxDistanceBetween = 350;

422         //InfinityRun.pause();

424

426         var j = 0;
427         var k = 1;
428         //var u = 0;

```

```
430         //update highscore TODO
432
434         for(var i = 0; i<9; i++) {
436             if (timePassed > highScore[i]) {
438
439                 // drag all scores down a
440                 value
441
442                 if(highScore[j] != 0) {
443                     highScore[k] =
444                     highScore[j];
445
446                     j++;
447                     k++;
448                 }
449             }
450         }
451
452         //highScore[0] =
453         timePassed;
454     }
455
456     InfinityRun.platformManager.updateWhenLose();
457     fxaudio.pause();
458     fxaudio.src = 'sounds/crash.wav';
459     fxaudio.load();
460     fxaudio.play();
461
462     date = new Date();
463     //bg ändern
464     //jumpheight=0;
465 }
466
467 if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
468     InfinityRun.keys.W || InfinityRun.dragging) &&
469     this.velocityY < -8) {
470     this.velocityY += -0.75;
471     //jumpheight+=1
472 }
```

```

472         if (InfinityRun.keys.DOWN) {
474             this.velocityY += 1;
476         }
478     };
479
480     Player.prototype.draw = function() {
481         InfinityRun.fillStyle = this.color;
482         InfinityRun.fillRect(this.x, this.y, this.width, this.height);
483     };
484
485     // ————— Platforms —————
486
487     function Platform(options) {
488         this.x = options.x;
489         this.y = options.y;
490         this.width = options.width;
491         this.height = options.height;
492         this.previousX = 0;
493         this.previousY = 0;
494         this.color = options.color;
495     }
496
497     Platform.prototype = new Vector2;
498
499     Platform.prototype.draw = function() {
500         InfinityRun.fillStyle = this.color;
501         InfinityRun.fillRect(this.x, this.y, this.width, this.height);
502     };
503
504     // ————— Platform Manager —————
505
506     function PlatformManager() {
507         this.maxDistanceBetween = 300;
508         //bunt
509         //this.colors = ['#2ca8c2', '#98cb4a', '#f76d3c', '#f15f74', '#5481e6'];
510         //bunt mit grau
511         //this.colors = ['#AB8888', '#8E8383', '#625B5B', '#9491B1', '#A68FA4'];
512         //grau gelb
513         //this.colors = ['#D8D6B1', '#B5B28C', '#71705E', '#A4A072', '#8E8A5E'];
514         //grau blau
515         this.colors = ['#BEC3E3', '#A6AAC6', '#9194AB', '#8E8A5E'];

```

```

    '#77798A', '#666876'];

516         //first 3 Platforms execept the Starter Platform
518     this.first = new Platform({
520         x: 300,
522         y: 600,
        width: 400,
        height: 70
    })
524     this.second = new Platform({
        x: (this.first.x + this.first.width) + random(▷
            this.maxDistanceBetween - 150, this.▷
            maxDistanceBetween),
526         y: random(this.first.y - 128, InfinityRun.height ▷
            - 80),
        width: 400,
528         height: 70
    })
530     this.third = new Platform({
        x: (this.second.x + this.second.width) + random(▷
            this.maxDistanceBetween - 150, this.▷
            maxDistanceBetween),
532         y: random(this.second.y - 128, InfinityRun.height ▷
            - 80),
        width: 400,
534         height: 70
    })
536     this.first.height = this.first.y + InfinityRun.height ▷
        ;
538     this.second.height = this.second.y + InfinityRun.▷
        height;
    this.third.height = this.third.y + InfinityRun.height ▷
        ;
540     this.first.color = randomChoice(this.colors);
    this.second.color = randomChoice(this.colors);
542     this.third.color = randomChoice(this.colors);

544     this.colliding = false;

546     this.platforms = [this.first, this.second, this.third ▷
        ];
    }
548 PlatformManager.prototype.update = function() {
550     this.first.x -= 3 + InfinityRun.acceleration;

```

```

552     if (this.first.x + this.first.width < 0) {
        this.first.width = random(450, 800);
554     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);
556     this.first.height = this.first.y + InfinityRun.>
        height + 10;
        this.first.color = randomChoice(this.colors);
558     }

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

568     this.third.x -= 3 + InfinityRun.acceleration;
    if (this.third.x + this.third.width < 0) {
570     this.third.width = random(450, 800);
572     this.third.x = (this.second.x + this.second.width >
        ) + random(this.maxDistanceBetween - 150, this.>
        .maxDistanceBetween);
        this.third.y = random(this.second.y - 32, >
        InfinityRun.height - 80);
574     this.third.height = this.third.y + InfinityRun.>
        height + 10;
        this.third.color = randomChoice(this.colors);
576     }

578 };

580

582 // reset / new Game: set Starting Platform Parameters
    PlatformManager.prototype.updateWhenLose = function() {
584
        this.first.x = 300;
586     this.first.color = randomChoice(this.colors);
        this.first.y = 500;

```

```

588 //this.first.y = InfinityRun.width / random(2, 3);
    this.second.x = (this.first.x + this.first.width) + ↵
        random(this.maxDistanceBetween - 150, this.↵
            maxDistanceBetween);
590 this.third.x = (this.second.x + this.second.width) + ↵
        random(this.maxDistanceBetween - 150, this.↵
            maxDistanceBetween);

592 };

594 // ————— Particle System ————— (Sketch Docs)

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

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

610 Particle.prototype.draw = function() {
612     InfinityRun.fillStyle = this.color;
        InfinityRun.fillRect(this.x, this.y, this.size, this.↵
            size);
614 };

616 /*****

618 InfinityRun.setup = function() {

620     this.jumpCount = 0;
        this.acceleration = 0;
622     this.accelerationTweening = 0;
        this.player = new Player({
624         x: 150,
            y: 30,
626         width: 32,

```

```

        height: 32
628    });
        bgaudio.pause();
630    bgaudio.src = 'sounds/menu.wav';
        bgaudio.load();
632    bgaudio.play();

634    this.platformManager = new PlatformManager();

636    this.particles = [];
    this.particlesIndex = 0;
638    this.particlesMax = 20;
    this.collidedPlatform = null;
640    this.scoreColor = '#181818';
    this.jumpCountRecord = 0;
642    //-----
    //bg add
644    var i, results;
    i = 3;
646    results = [];
    while (i--) {
648        results.push(Town.push(new Street({
            layer: i + 1,
650            width: {
                min: (i + 1) * 20,
652                max: (i + 1) * 50
            },
654            height: {
                min: InfinityRun.height - 200 - (i * round(↵
                    InfinityRun.height / 3)),
656                max: InfinityRun.height - 50 - (i * round(↵
                    InfinityRun.height / 3))
            },
658            speed: (i + 1) * .003,
            color: 'hsl( 200, ' + (((i + 1) * 1) + 10) + '%, ↵
                ' + (75 - (i * 13)) + '% )'
        })));
660    }
662    return results;
    //-----
664

666 };
    //-----
668 //clear func bg
    InfinityRun.clear = function() {
670     return InfinityRun.clearRect(0, 0, InfinityRun.width, ↵
        InfinityRun.height);

```



```

    };
672 //-----

674 Array.max = function( array ){
    return Math.max.apply( Math, array );
676 };

678 InfinityRun.update = function() {
    if (GameState == State.Started) {
680         //-----
        //clear func bg
        var i, results;
682         dt = InfinityRun.dt < .1 ? .1 : InfinityRun.dt / 16;
684         dt = dt > 5 ? 5 : dt;
        i = Town.length;
686         results = [];
        while (i--) {
688             results.push(Town[i].update(i));
        }
690         //return results;
        //-----

692         dateNow = new Date();
694         timePassed = (dateNow - date) / 10;

696         this.player.update();
        restartAudio();
698         switch (this.jumpCount) {
            case 0:
700             bgaudio.pause();
             bgaudio.src = 'sounds/main1.wav';
702             bgaudio.load();
             bgaudio.play();

704             break;
            case 10:
706             this.accelerationTweening = 1.3;
             this.platformManager.maxDistanceBetween = 430;
708             //this.scoreColor = '#076C00';
             bgaudio.pause();
710             bgaudio.src = 'sounds/main2.wav';
             bgaudio.load();
712             bgaudio.play();
             fxaudio.pause();
714             fxaudio.src = 'sounds/levelup.wav';
             fxaudio.load();
716             fxaudio.play();

```

```

        break;
718     case 25:
        this.accelerationTweening = 2.3;
720     this.platformManager.maxDistanceBetween = 530;
        //this.scoreColor = '#0300A9';
722         bgaudio.pause();
        bgaudio.src = 'sounds/main3.wav';
724         bgaudio.load();
        bgaudio.play();
726         fxaudio.pause();
        fxaudio.src = 'sounds/levelup.wav';
728         fxaudio.load();
        fxaudio.play();

730     break;
    case 40:
732     this.accelerationTweening = 3.5;
        this.platformManager.maxDistanceBetween = 580;
734     //this.scoreColor = '#9F8F00';
        bgaudio.pause();
736         bgaudio.src = 'sounds/main4.wav';
        bgaudio.load();
738         bgaudio.play();
        fxaudio.pause();
740         fxaudio.src = 'sounds/levelup.wav';
        fxaudio.load();
742         fxaudio.play();

        break;
744 }

746 this.acceleration += (this.accelerationTweening -
        this.acceleration) * 0.01;

748

750 for (i = 0; i < this.platformManager.platforms.length; i++) {
    if (this.player.intersects(this.platformManager.platforms[i])) {
752         this.collidedPlatform = this.platformManager.platforms[i];
        if (this.player.y < this.platformManager.platforms[i].y) {
754             this.player.y = this.platformManager.platforms[i].y;

```

```

756         // Gravity after Collision with Platform
              this.player.velocityY = 0;
758     }

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

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

768     if (this.player.intersectsLeft(this.
        platformManager.platforms[i])) {
770         this.player.x = this.collidedPlatform.x -
            64;
        for (i = 0; i < 10; i++) {
772             // SpawnParticles @PlayerPostion with
                intersecting Platform Color
            this.particles[(this.particlesIndex
                ++)% this.particlesMax] = new
                Particle({
774                 x: this.player.x + this.player.
                    width,
                    y: random(this.player.y, this.
                        player.y + this.player.height)
                    ,
776                 velocityY: random(-30, 30),
                    color: randomChoice(['#181818',
                        '#181818', this.
                            collidedPlatform.color])
                });
778        };

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

784

786

788         if (timePassed > this.
            jumpCountRecord) {

```

```

        this.jumpCountRecord = timePassed;
    }

    } else {

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

            //play jump_sound
            fxaudio.pause();
            fxaudio.src = '
                sounds/jump.
                wav';
            fxaudio.load();
            fxaudio.play();

        }
    }
};

for (i = 0; i < this.platformManager.platforms.length;
    i++) {
    this.platformManager.update();
};

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

//-----
//bg
return results;
//-----
}

};

```

```

832
834 var selectedItem = 0;
836 InfinityRun.keydown = function() {
    if (InfinityRun.keys.ESCAPE && GameState==State.▷
        Started) {
838         InfinityRun.clear();
            GameState = State.Menu;
840         bgaudio.pause();
            bgaudio.src = 'sounds/menu.wav';
842         bgaudio.load();
            bgaudio.play();
844
            } else if (InfinityRun.keys.ESCAPE && GameState==▷
                State.Menu && curMenuTab==MenuTab.Main) {
846                 GameState = State.Started;
            } else if (InfinityRun.keys.ESCAPE && GameState==▷
                State.Menu && curMenuTab==MenuTab.Settings) {
848                 curMenuTab = MenuTab.Main;
            } else if (InfinityRun.keys.ESCAPE && GameState==▷
                State.Menu && curMenuTab==MenuTab.Highscore) {
850                 curMenuTab = MenuTab.Main;
            }
852
            //main menu controls
854         if (InfinityRun.keys.UP) {
                selectedItem = (selectedItem + items.▷
                    length - 1) % items.length;
856         }
            if (InfinityRun.keys.DOWN) {
858                 selectedItem = (selectedItem + 1) % items▷
                    .length;
            }
860
            // settings audio change
862         if (InfinityRun.keys.LEFT && curMenuTab==MenuTab.▷
            Settings && selectedItem !=0) {
                selectedItem = (selectedItem + items.▷
                    length - 1) % items.length;
864         }
866
            if (InfinityRun.keys.RIGHT && curMenuTab==MenuTab▷
            .Settings && selectedItem !=9) {
                selectedItem = (selectedItem + 1) % items▷
                    .length;
868         }

```

```

870         if (InfinityRun.keys.ENTER) {
871             callback(selectedItem);
872         }
873     }
874 }
875
876 Menu = function() {
877
878     //this.backgroundCallback = null;
879 }
880
881 //----- Draw -----
882
883 InfinityRun.draw = function() {
884
885     if (GameState == State.Started) {
886         //-----
887
888         //bg draw
889         var i, results;
890         i = Town.length;
891         results = [];
892         while (i--) {
893             results.push(Town[i].render(i));
894         }
895         //-----
896
897         this.player.draw();
898
899         for (i = 0; i < this.platformManager.platforms.length; i++) {
900             this.platformManager.platforms[i].draw();
901         };
902
903         //Draw particles
904         for (i = 0; i < this.particles.length; i++) {
905             this.particles[i].draw();
906         };
907
908         /*
909         * Main Menu
910         */
911     } else if (GameState == State.Menu && curMenuTab >
912         == MenuTab.Main) {

```

```

    this.title = "InfinityRun";
914 items = ["Play", "Settings", "Highscore", "Credits"];

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

918 switch (numItem) {
    case 0:
920     GameState=State.Started;
        break;
922     case 1:
        curMenuTab=MenuTab.Settings;
924     break;
    case 2:
926     curMenuTab=MenuTab.Highscore;
        break;
928     case 3:
        curMenuTab=MenuTab.Credits;
930     break;

932 }

934

936 };
    this.height = InfinityRun.height;
938 this.width = InfinityRun.width;
    this.size = 120;

940
    var lingrad = this.createLinearGradient(0,0,0, this.height);
    lingrad.addColorStop(0, '#000');
942 lingrad.addColorStop(1, '#023');
944 this.fillStyle = lingrad;
    this.fillRect(0,0,this.width, this.height)

946
    this.textAlign = "center";
948 this.fillStyle = "White";

950
    var height = 150;
    //logo
952 this.drawImage(bglogo, this.width-500, this.height-300);
    //
954 if (this.title) {
        this.font = Math.floor(this.size*1.3).toString() + "px Times New Roman";

```

```

956         this.fillText(this.title , this.width/2, ↵
            height);
            height+= height;
958     }

960     for (var i = 0; i < items.length; ++i)
    {
962         var size = Math.floor(this.size*0.8);
        if (i == selectedItem)
964     {
            this.fillStyle = "#A9F5F2";
            size = this.size+5;
966     }
        this.font = size.toString() + "px Times ↵
            New Roman";
        height += this.size;
970        this.fillText(items[i], InfinityRun.width ↵
            /2, height);
        this.fillStyle = "White";
972    }
    //----->

974    //bg dd <— ??
    return results;
976    //----->

978    /*
    * Settings Tab
980    *
    */
982    } else if (GameState == State.Menu && curMenuTab ↵
        ==MenuTab.Settings){

984        this.title = "Settings";
        items = [10, 20, 30, 40, 50, 60, 70, 80, 90 , ↵
            100];
986

988

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

990

        switch (volume) {
992
        }
994

```



```
996         };
998
1000         this.height = InfinityRun.height;
1002         this.width = InfinityRun.width;
1004         var lingrad = this.createLinearGradient(0,0,0, ▷
            this.height);
1006         lingrad.addColorStop(0, '#000');
1008         lingrad.addColorStop(1, '#023');
1010         this.fillStyle = lingrad;
1012         this.fillRect(0,0,this.width, this.height, items[▷
            i]);
1014
1016         this.textAlign = "center";
1018         this.fillStyle = "White";
1020
1022         var width = 10;
1024         var height = 10;
1026         var posx = 130;
1028         var posy = 380;
1030         this.space = 15;
1032         this.heightincr = 4;
1034
1036         if (this.title) {
1038             this.font = Math.floor(this.size*1.3). ▷
                toString() + "px_Times_New_Roman";
1040             this.fillText(this.title, this.width/2, ▷
                150);
1042             height+= height;
1044         }
1046
1048         this.font = "70px_Times_New_Roman";
1050         //this.fillText('Volume', InfinityRun.Left, ▷
            InfinityRun.Top);
1052
1054
1056         this.fillText('Volume', 240, 300);
1058
1060
1062         for (var i = 0; i < items.length; ++i) {
1064             var size = Math.floor(this.size*0.8);
```

```

1040         if (i == selectedItem)
1041         {
1042             this.fillStyle = "#A9F5F2";
1043             size = this.size+5;
1044         }
1045         this.font = size.toString() + "px Times New Roman";
1046         posx += this.space;
1047         posy -= this.heightincr;
1048         height += this.heightincr;
1049
1050         items[i] = this.fillRect(posx, posy, width, height);
1051
1052         //this.fillText(items[i], InfinityRun.width/2, height);
1053         this.fillStyle = "White";
1054     }
1055
1056     /*
1057     * Highscore Tab
1058     */
1059
1060 } else if (GameState == State.Menu && curMenuTab == MenuTab.Highscore) {
1061
1062
1063
1064     this.title = "Highscore";
1065     items = highScore;
1066
1067
1068     callback = function(volume) { //if (numItem == 0)
1069         GameState=State.Started
1070
1071         switch (volume) {
1072
1073         }
1074
1075     };
1076     this.height = InfinityRun.height;
1077     this.width = InfinityRun.width;
1078
1079
1080     var lingrad = this.createLinearGradient(0,0,0,
        this.height);

```

```

1082     lingrad.addColorStop(0, '#000');
1083     lingrad.addColorStop(1, '#023');
1084     this.fillStyle = lingrad;
1085     this.fillRect(0,0,this.width, this.height, items[
        i]);

1086     this.textAlign = "center";
1087     this.fillStyle = "White";
1088
1089     var width = 10;
1090     var height = 150;

1091
1092     if (this.title) {
1093         this.font = Math.floor(this.size*1.3).
            toString() + "px Times New Roman";
1094         this.fillText(this.title, this.width/2,
            150);
1095         height+= height;
1096     }

1097
1098     var rank = 1;
1099
1100     for (var i = 0; i < items.length; ++i)
1101     {

1102
1103         var size = Math.floor(this.size*0.8);
1104         if (i == selectedItem)
1105         {
1106             this.fillStyle = "#A9F5F2";
1107             size = this.size+5;
1108         }
1109         this.font = 0.6*size.toString() + "px
            Times New Roman";
1110         height += 50;
1111         this.fillText(rank + ". " + items[i],
            InfinityRun.width/2, height);
1112         this.fillStyle = "White";

1113         rank++;
1114     }

1115 }

1116 }

1117 }

1118 }

1119
1120
1121
1122 //Debug
    if (debug) {

```

```

1124     this.font = '12pt Arial';
        this.fillStyle = '#181818';
1126     this.fillText('Record: ' + highScore[0]/*this.▷
        jumpCountRecord*/, this.width - 150, 33);
        this.fillStyle = this.scoreColor;
1128     this.fillText('Jumps: ' + this.jumpCount, this.▷
        width - 150, 50);
        this.fillText('Distance: ' + Math.round(▷
        timePassed) , this.width - 150, 65);
1130         this.fillText('mouse: ' + this.mouse.y , ▷
        this.width - 150, 100);
        this.fillText('GameState: ' + GameState, this.▷
        width - 150, 80);
1132     }

1134 };

1136 InfinityRun.resize = function() {
        /* todo Windowscale optimization
1138     *
        *
1140     */
    }

```

A.3. game.css

```

1 body{
    background: #e3e3e3;
3    overflow: hidden;
    margin: 0;
5    padding: 0;
    text-align: center;
7 }
#container{
9    /*margin-top: 10%;*/
    display: inline-block;
11 }
    canvas{
13     background: #cecece;
        border: 1px solid #181818;
15 }

```

A.4. index.html

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD_XHTML_1.0_Strict//EN" ▷
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" ▷
        lang="en">
3 <head>

```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5     <script type="text/javascript" src="js/sketch.min.js" charset="utf-8"></script>

7     <title>Infinity Run</title>

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