

Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent : **Gabriela Mai**
Vorgelegt am : 23. November 2016
Vorgelegt von : Gruppe 4

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

Abstract

Ziel ist es ein Browsergame mittels Javascript zu programmieren. Dieses Spiel wird mittels Notepad++ als Editor, Chrome als ausführenden Browser, Gimp als Bearbeitungsprogramm und Github als Softwareverwaltung realisiert. Stilistische Elemente werden mittels HTML und CSS eingebunden. Unser Spiel namens „InfinityRun“ wird ein Endlos-Spiel sein bei der 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. Spiel begleitend wird ein Soundtrack das Spielerlebnis abrunden. Bei Aufprall auf ein Hindernis, besteht die Möglichkeit das Spiel neu zu starten.

Inhaltsverzeichnis

Inhaltsverzeichnis	iv
Abbildungsverzeichnis	v
Tabellenverzeichnis	vii
Abkürzungsverzeichnis	ix
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	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
2.3	Test	16
2.4	Dokumentation & Präsentation	16
3	Ausblick	17
4	Fazit	19
	Literaturverzeichnis	21
	Eidesstattliche Erklärung	23
A	Anhang	25
A.1	Github Changelog	25
A.2	game.js	29
A.3	game.css	42
A.4	index.html	42

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: Start Bildschirm	9
Abbildung 8: Das Spiel	9

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

Abkürzungsverzeichnis

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 der 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. Spiel begleitend wird ein Soundtrack das Spielerlebnis abrunden. Bei Aufprall auf ein Hindernis, besteht die Möglichkeit das Spiel neu zu starten.

1.3.2. Entwurfsskizze

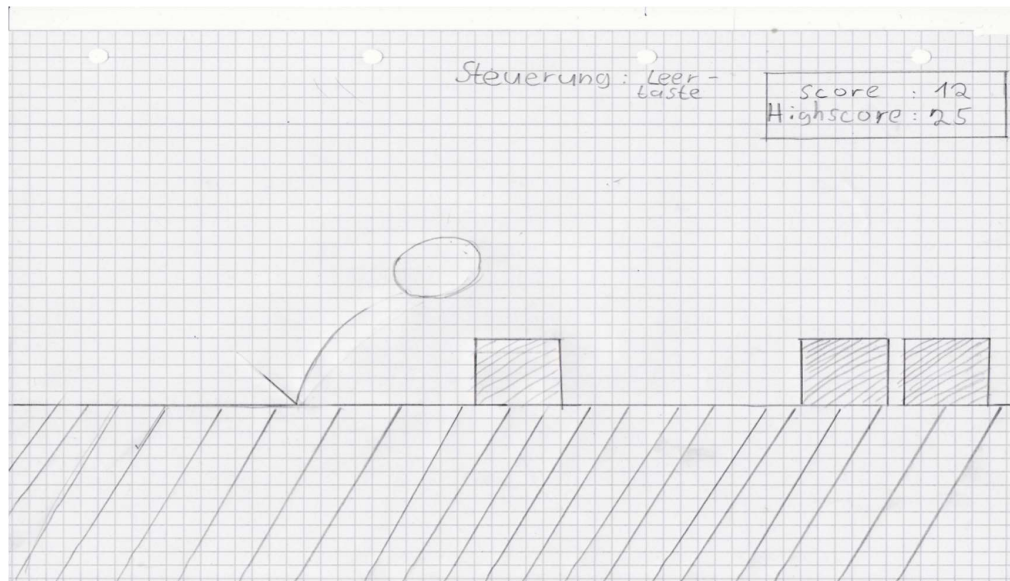


Abbildung 6.: Entwurfsskizze

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 werden Zufällig von rechts in den Bildschirm generiert. Es können verschieden Kombinationen z.B. ein Block, zwei Blöcke oder drei Blöcke generiert werden. Außerdem kann man oben im 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. Daneben soll noch ein Pausebutton sichtbar sein, womit man das Spiel pausieren kann. Dieser Pausebutton wird mit der Taste P hinterlegt. 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 von Linus Torvalds entstand. Github ist eine Open Source Plattform die dieses Konzept nutzt. Somit können wir parallel an dem Projekt arbeiten. 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. Zu jeweiligen Version haben wir ein Fertigstellungsdatum festgelegt und den dann erforderlichen Inhalt festgelegt.

2.2. Implementation

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 Start Bildschirm des Spiels. Hier gibt es verschiedene Auswahl Möglichkeiten. In der Abbildung 8 zu sehen ist der derzeitige Stand des Spiels.

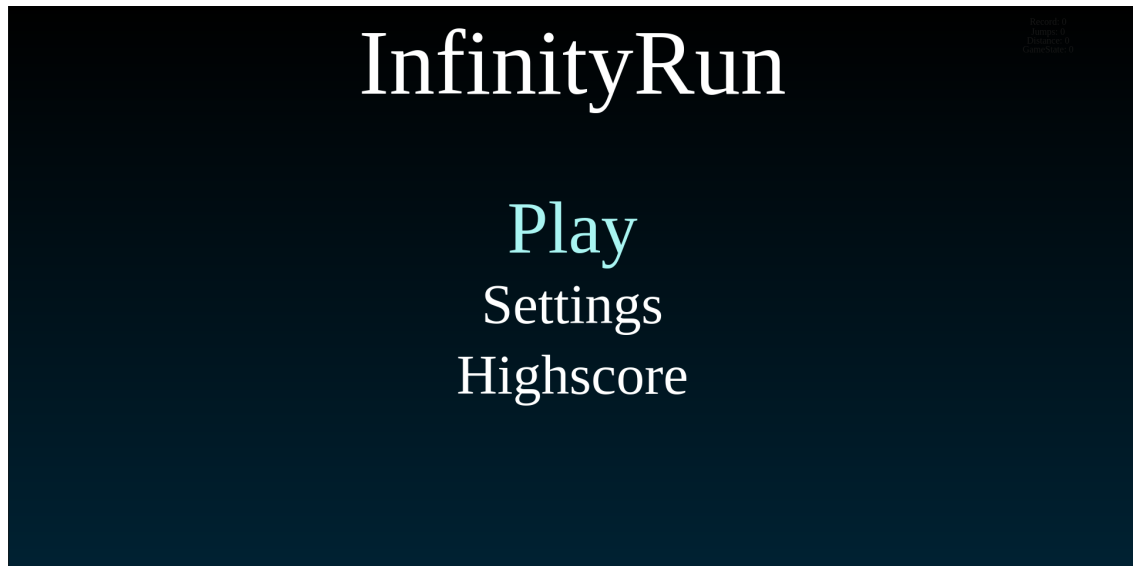


Abbildung 7.: Start Bildschirm



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

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

```

Player.prototype.update = function() {
2 function PlatformManager()
{
4
    this.maxDistanceBetween = 300;
6
    this.colors = [ '#2ca8c2 ', '#98cb4a ', '#f76d3c ', '
    #f15f74 ', '#5481e6 ' ];
8
    //first 3 Platforms execept the Starter Platform
10    this.first = new Platform({
        x: 300,
12        y: InfinityRun.width / 2,
        width: 400,
14        height: 70
    })
16
    this.second = new Platform
18 ({
        x: (this.first.x + this.first.width) + random(
        this.maxDistanceBetween - 150, this.
        maxDistanceBetween),
20        y: random(this.first.y - 128, InfinityRun.height
        - 80),
        width: 400,
22        height: 70
    })
24
    this.third = new Platform
26 ({

```



```

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

```

2.2.5.4. Update der Plattformen

Die Plattformupdatefunktion 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, sämtliche werte so zufällig neu gesetzt, dass sie wieder von rechts ins Bild laufen kann. Dies wird für alle 3 Plattformen gleich durchgeführt.

```

PlatformManager.prototype.update = function()
2 {
    this.first.x -= 3 + InfinityRun.acceleration;
4 if (this.first.x + this.first.width < 0)
    {
6         this.first.width = random(450, ▷
            InfinityRun.width + 200);
        this.first.x = (this.third.x + this.third▷
            .width) + random(this.▷

```

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

```
for (i = 0; i < this.platformManager.platforms.length;
    ; i++)
2   {
      if (this.player.intersects(this.
        platformManager.platforms[i]))
4       {
          this.collidedPlatform = this.
            platformManager.platforms[i];
6          if (this.player.y < this.
            platformManager.platforms[i].y
              )
            {
7              this.player.y = this.
                platformManager.
                platforms[i].y;
                // Gravity after
                Collision with
                Platform
10             this.player.velocityY =
                0;
            }
12
          this.player.x = this.player.
            previousX;
14         this.player.y = this.player.
            previousY;

16         this.particles[(this.
            particlesIndex++) % this.
            particlesMax] = new Particle({
x: this.player.x,
```

```
18         y: this.player.y + this.player.height ,  
19         color: this.collidedPlatform.color  
20     });
```

2.3. Test

2.4. Dokumentation & Präsentation

3. Ausblick

4. Fazit

Literaturverzeichnis

- [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 23. November 2016 Florian Durli

FURTWANGEN, den 23. November 2016 Jannik Ivosevic

FURTWANGEN, den 23. November 2016 Johannes But

FURTWANGEN, den 23. November 2016 Marco Mayer

FURTWANGEN, den 23. November 2016 Koray Emtekin

A. Anhang

A.1. Github Changelog

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

```
$ git log > CHANGELOG.md
```

Quelle: [Gru]

Changelog:

```

1 commit 1392138abdf3404d8c7815a59455d922811c37b5
  Author: Florian Durli <florian.durli@yahoo.de>
3 Date:   Wed Nov 23 22:44:10 2016 +0100

5     Jojos Description in LaTeX

7 commit 25ff037d002168fa55525271059ad9add29ed616
  Author: butjo <johannes.but@gmail.com>
9 Date:   Wed Nov 23 21:59:41 2016 +0100

11    Beschreibung der Codeteile in der phasen.tex von ↵
      Johannes

13 commit 1b2348c013bfe18d574bc041d7dbc6049b64f7d8
  Author: Florian Durli <florian.durli@yahoo.de>
15 Date:   Wed Nov 23 17:06:56 2016 +0100

17    Changelog Additionally

19 commit cf467dcb47436f7b9637544e603115d548931192
  Author: Florian Durli <florian.durli@yahoo.de>
21 Date:   Wed Nov 23 17:05:30 2016 +0100

23    Changelog Additionally

25 commit 6db1ad68850073222e10bfa65d1dbb9029a2466c
  Merge: 69dd747 b924713
27 Author: butjo <johannes.but@gmail.com>
  Date:   Wed Nov 23 10:48:34 2016 +0100

29    Merge branch 'master' of https://github.com/Slay3r/↵
      InfinityRun
  
```

```
31 commit b924713e96128d8baee3806dcda4f73e858f83fb
33 Author: Slay3r <florian.durli@yahoo.de>
   Date:   Wed Nov 23 10:41:49 2016 +0100

35     Generated Changelog

37 commit 69dd7479384cc9c649a4e239feeb802842c077cb
39 Merge: 743c95a 203ae2e
   Author: butjo <johannes.but@gmail.com>
41 Date:   Wed Nov 23 10:31:50 2016 +0100

43     Merge branch 'master' of https://github.com/Slay3r/∞
       InfinityRun

45 commit 203ae2e60637b23024065ab718fee024912acafb
   Author: Slay3r <florian.durli@yahoo.de>
47 Date:   Wed Nov 23 09:53:47 2016 +0100

49     Bilder des Spiels

51 commit d274cfc72b3b221bfad21f8b5b4c6c12eea77794
   Author: Slay3r <florian.durli@yahoo.de>
53 Date:   Wed Nov 23 09:09:44 2016 +0100

55     Anforderungen

57 commit a0909ac15cc946db0d4c2f692077c9213e6f0e1c
   Author: Slay3r <florian.durli@yahoo.de>
59 Date:   Wed Nov 23 08:56:53 2016 +0100

61     Literaturverzeichnis

63 commit 19e2f3de440e911ca68986a2d614f43aeb154fb9
   Author: Slay3r <florian.durli@yahoo.de>
65 Date:   Wed Nov 23 08:29:52 2016 +0100

67     Doku update

69 commit 21889f9ba5664cc42f61515c17dbeba18813df5c
   Author: Florian Durli <florian.durli@yahoo.de>
71 Date:   Tue Nov 22 20:41:15 2016 +0100

73     Add Changelog

75 commit 743c95acc76492ba7513f12d5661fc1cbc25d4bd
   Author: butjo <johannes.but@gmail.com>
77 Date:   Wed Nov 16 14:51:03 2016 +0100
```

```
79      Formatierte sketch.min.js

81      Ich hab ein paar Zeilen umbrüche und leerzeichen ↵
        eingefügt so ,dass der
        Code lesbar wird, ist noch nicht perfekt jedoch mal ↵
        ein Anfang

83      commit d64d2548992071cd7c270bdd10260866e7c7f895
85      Author: butjo <johannes.but@gmail.com>
        Date:   Wed Nov 16 13:46:27 2016 +0100

87      Endlose Schwierigkeitserhöhung

89      Vorschlag zum ersetzten der switch Anweisung ist ↵
        allerdings noch
91      ziemlich unbalanced vllt ist die switch Anweisung ↵
        die bessere
        Alternative

93      commit 3707b94db557746f67bc9a5304f15585ef4ed844
95      Author: butjo <johannes.but@gmail.com>
        Date:   Wed Nov 16 13:43:56 2016 +0100

97      Präsentation Zwischenstand

99      commit f23c9beb29c6d1aaa4dad726f92e56358e39db5
101     Author: Slay3r <florian.durli@yahoo.de>
        Date:   Wed Nov 16 11:00:40 2016 +0100

103     Präsentation überarbeitet

105     commit 53aa72e67df34194cc284825962a50e71ab7817a
107     Author: butjo <johannes.but@gmail.com>
        Date:   Wed Nov 16 08:13:14 2016 +0100

109     Präsentation und test

111     commit b5cb9783d57cbcd840d0dcc0d209c99a88ae8668
113     Merge: 11a5e55 275bd69
        Author: Florian Durli <florian.durli@gmail.com>
115     Date:   Wed Nov 16 07:53:21 2016 +0100

117     Merge pull request #1 from r4qtor/master

119     tiny cleanup & incl. menu

121     commit 275bd697f1c89e2aefb702594e16df897f8e134f
```

```
Author: r4qtor <r4qtor@gmail.com>
123 Date: Tue Nov 15 23:51:06 2016 +0100

125     tiny cleanup & incl. menu

127 commit 11a5e55daa5210eae69714aefdf20627b1ff34db
Author: Slay3r <florian.durli@yahoo.de>
129 Date: Wed Nov 9 15:53:22 2016 +0100

131     Basis implementation

133 commit c783850311ea1a406c57c9732d6851285e82f49e
Author: Slay3r <florian.durli@yahoo.de>
135 Date: Wed Nov 9 10:54:05 2016 +0100

137     Bilder hinzugefügt

139 commit d88d0d2c2077143f88ed75a82c6faa4cc9b91bfd
Author: Slay3r <florian.durli@yahoo.de>
141 Date: Wed Nov 9 09:42:35 2016 +0100

143     Dokumentations Basis

145 commit 39b4705b056ebfcf2067ba4e6d052919c994c983
Author: Florian <florian.durli@yahoo.de>
147 Date: Wed Oct 19 13:47:16 2016 +0200

149     Initial Struktur der Ordner und files

151 commit 093797e0b08692b71de636618261919b26918cde
Author: Florian Durli <florian.durli@gmail.com>
153 Date: Wed Oct 19 13:27:05 2016 +0200

155     Delete Requirements

157 commit 56c4aae71ca61dc46376e53e63353a2ac28926ca
Author: Florian <florian.durli@yahoo.de>
159 Date: Wed Oct 19 10:36:28 2016 +0200

161     doc Ordner

163 commit b3b83fd1b017e5f79a47b4ecc06073d3851ac871
Author: Florian <florian.durli@yahoo.de>
165 Date: Wed Oct 19 10:32:25 2016 +0200

167     Requirements hinzugefügt

169 commit b4d2627efaa5e5410cf89d9dccaed088fa75323
```


Author: Florian Durli <florian.durli@gmail.com>

171 Date: Wed Oct 19 10:28:46 2016 +0200

173 Initial commit

A.2. game.js

```

1  /* todo: cleanup (dirty code),
   *
3  * Put static values / vars into initialization function
   *
5  * _____
   * Design / Graphics
7  *
   * Parallax Background?
9  *
   * _____
11 * Menu
   * Menu draw in Input & draw prototypes
13 * Handle / Manage CSS or HTML variables from JavaScript ↵
   * (Fullscreen ,...)
   * _____
15 *
   * Platform Schematic? – Schematic files?
17 * Different Themes depending on Progress?
   *
19 * _____
   * Test-Phase
21 *
   * Controller: 'dragging' test Touch support
23 * Browsertesting tools
   * eg.:
25 * http://browserling.com/
   * http://browsershots.org/
27 * https://crossbrowsertesting.com/
   * https://www.browserstack.com/
29 */
   var i = 0;
31
   var State = { Menu:0, Started:1, Paused:2, Over:3 };
33 var GameState = State.Menu;
   var MainMenu;
35
   var debug = true;
37
   // randomizer
39 function random(min, max) {

```

```

        return Math.round(min + (Math.random() * (max - min))) >
    );
41 }

43 function randomChoice(array) {
    return array[Math.round(random(0, array.length - 1))] >
    ];
45 }

47
49 //initialize Sketch Framework
var InfinityRun = Sketch.create({
    fullscreen: true,
51     width: 640,
    height: 360,
53     container: document.getElementById('container')
});
55
57 //----- Vector [Get/Set] Functions -----
//Set X,Y,Width,Height
59 function Vector2(x, y, width, height) {
    this.x = x;
61     this.y = y;
    this.width = width;
63     this.height = height;
    this.previousX = 0;
65     this.previousY = 0;
    };
67

69 // Set X,Y
Vector2.prototype.setPosition = function(x, y) {
71
    this.previousX = this.x;
73     this.previousY = this.y;

75     this.x = x;
    this.y = y;
77
    };
79 // Set X
Vector2.prototype.setX = function(x) {
81
    this.previousX = this.x;
83     this.x = x;

85 };

```

```

87 // Set Y
   Vector2.prototype.setY = function(y) {
89
   this.previousY = this.y;
91   this.y = y;

93 };

95 // Collision / Intersection Top
   Vector2.prototype.intersects = function(obj) {
97
   if (obj.x < this.x + this.width && obj.y < this.y + ↵
       this.height &&
99       obj.x + obj.width > this.x && obj.y + obj.height ↵
       > this.y) {
       return true;
101   }

103   return false;
   };

105 // Collision / Intersection Left
107 Vector2.prototype.intersectsLeft = function(obj) {

109   if (obj.x < this.x + this.width && obj.y < this.y + ↵
       this.height) {
       return true;
111   }

113   return false;
   };

115 //————— Player—————
117 function Player(options) {
119
   this.setPosition(options.x, options.y);
121   this.width = options.width;
   this.height = options.height;
123   this.velocityX = 0;
   this.velocityY = 0;
125   this.jumpSize = -13;
   this.color = '#181818';

127 }

129 Player.prototype = new Vector2;

```

```

131 Player.prototype.update = function() {
132     // Gravity
133     this.velocityY += 1;
134     this.setPosition(this.x + this.velocityX, this.y +
135         this.velocityY);
136
137     if (this.y > InfinityRun.height || this.x + this.
138         width < 0) {
139         this.x = 150;
140         this.y = 50;
141         this.velocityX = 0;
142         this.velocityY = 0;
143         InfinityRun.jumpCount = 0;
144         InfinityRun.acceleration = 0;
145         InfinityRun.accelerationTweening = 0;
146         InfinityRun.scoreColor = '#181818';
147         InfinityRun.platformManager.maxDistanceBetween =
148             350;
149         InfinityRun.platformManager.updateWhenLose();
150     }
151
152     if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
153         InfinityRun.keys.W || InfinityRun.dragging) &&
154         this.velocityY < -8) {
155         this.velocityY += -0.75;
156     }
157
158 };
159
160 Player.prototype.draw = function() {
161     InfinityRun.fillStyle = this.color;
162     InfinityRun.fillRect(this.x, this.y, this.width, this.
163         height);
164 };
165
166 // ----- Platforms -----
167
168 function Platform(options) {
169     this.x = options.x;
170     this.y = options.y;
171     this.width = options.width;
172     this.height = options.height;
173     this.previousX = 0;
174     this.previousY = 0;
175     this.color = options.color;
176 }

```

```

173 Platform.prototype = new Vector2;

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

179 // ————— Platform Manager —————

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

185

187     //first 3 Platforms execept the Starter Platform
    this.first = new Platform({
189         x: 300,
        y: InfinityRun.width / 2,
191         width: 400,
        height: 70
193     })
    this.second = new Platform({
195         x: (this.first.x + this.first.width) + random(
            this.maxDistanceBetween - 150, this.
                maxDistanceBetween),
        y: random(this.first.y - 128, InfinityRun.height
            - 80),
197         width: 400,
        height: 70
199     })
    this.third = new Platform({
201         x: (this.second.x + this.second.width) + random(
            this.maxDistanceBetween - 150, this.
                maxDistanceBetween),
        y: random(this.second.y - 128, InfinityRun.height
            - 80),
203         width: 400,
        height: 70
205     })

207     this.first.height = this.first.y + InfinityRun.height
        ;
    this.second.height = this.second.y + InfinityRun.
        height;
209     this.third.height = this.third.y + InfinityRun.height
        ;

```

```

    this.first.color = randomChoice(this.colors);
211  this.second.color = randomChoice(this.colors);
    this.third.color = randomChoice(this.colors);
213
    this.colliding = false;
215
    this.platforms = [this.first, this.second, this.third];
217 }

219 PlatformManager.prototype.update = function() {

221     this.first.x -= 3 + InfinityRun.acceleration;
    if (this.first.x + this.first.width < 0) {
223         this.first.width = random(450, InfinityRun.width +
            + 200);
        this.first.x = (this.third.x + this.third.width) +
            + random(this.maxDistanceBetween - 150, this.
            maxDistanceBetween);
225         this.first.y = random(this.third.y - 32,
            InfinityRun.height - 80);
        this.first.height = this.first.y + InfinityRun.
            height + 10;
227         this.first.color = randomChoice(this.colors);
    }

229
    this.second.x -= 3 + InfinityRun.acceleration;
231  if (this.second.x + this.second.width < 0) {
        this.second.width = random(450, InfinityRun.width +
            + 200);
233         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);
235         this.second.height = this.second.y + InfinityRun.
            height + 10;
        this.second.color = randomChoice(this.colors);
237     }

239
    this.third.x -= 3 + InfinityRun.acceleration;
    if (this.third.x + this.third.width < 0) {
241         this.third.width = random(450, InfinityRun.width +
            + 200);
        this.third.x = (this.second.x + this.second.width) +
            + random(this.maxDistanceBetween - 150, this.
            maxDistanceBetween);
243         this.third.y = random(this.second.y - 32,

```

```

        InfinityRun.height - 80);
        this.third.height = this.third.y + InfinityRun.▷
        height + 10;
245         this.third.color = randomChoice(this.colors);
    }
247 };
249

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

255     this.first.x = 300;
        this.first.color = randomChoice(this.colors);
257     this.first.y = InfinityRun.width / random(2, 3);
        this.second.x = (this.first.x + this.first.width) + ▷
        random(this.maxDistanceBetween - 150, this.▷
        maxDistanceBetween);
259     this.third.x = (this.second.x + this.second.width) + ▷
        random(this.maxDistanceBetween - 150, this.▷
        maxDistanceBetween);

261 };

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

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

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

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

```

```

        InfinityRun.fillRect(this.x, this.y, this.size, this.size);
283 };

285 /*****

287 InfinityRun.setup = function() {

289     this.jumpCount = 0;
        this.acceleration = 0;
291     this.accelerationTweening = 0;

293     this.player = new Player({
        x: 150,
295         y: 30,
        width: 32,
297         height: 32
    });

299     this.platformManager = new PlatformManager();

301     this.particles = [];
303     this.particlesIndex = 0;
        this.particlesMax = 20;
305     this.collidedPlatform = null;
        this.scoreColor = '#181818';
307     this.jumpCountRecord = 0;

309 };

311

313 InfinityRun.update = function() {

315     /*switch(GameState){
        case State.Menu:
317         //InfinityRun.stop();
        break;
319         case State.Started:
        break;
321         case State.Paused:
        break;
323         case State.Over:
        break;

325     }*/

327     if (GameState == State.Started) {
        this.player.update();
    }
}

```



```
329         //endless increasing difficulty
331     /*
333     this.accelerationTweening = 0.2 * this.jumpCount;
335     if (this.jumpCount>5) {
337         this.platformManager.maxDistanceBetween = 300 + ↵
            2* this.jumpCount;
339     }
341     */

343     switch (this.jumpCount) {
345     case 10:
347         this.accelerationTweening = 1;
349         this.platformManager.maxDistanceBetween = ↵
            430;
351         //this.scoreColor = '#076C00';
353         break;
355     case 25:
357         this.accelerationTweening = 2;
359         this.platformManager.maxDistanceBetween = ↵
            530;
361         //this.scoreColor = '#0300A9';
363         break;
365     case 40:
367         this.accelerationTweening = 3;
369         this.platformManager.maxDistanceBetween = ↵
            580;
371         //this.scoreColor = '#9F8F00';
373         break;
375     }

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

379

381     for (i = 0; i < this.platformManager.platforms.length ↵
        ; i++) {
383         if (this.player.intersects(this.platformManager.↵
            platforms[i])) {
385             this.collidedPlatform = this.platformManager.↵
                platforms[i];
387             if (this.player.y < this.platformManager.↵
                platforms[i].y) {
389                 this.player.y = this.platformManager.↵
                    platforms[i].y;
391             }
393         }
395     }
397 }
```

```

367         // Gravity after Collision with Platform
           this.player.velocityY = 0;
369     }

371     this.player.x = this.player.previousX;
           this.player.y = this.player.previousY;
373
           this.particles[(this.particlesIndex++) % this.particlesMax] = new Particle({
375         x: this.player.x,
           y: this.player.y + this.player.height,
377         color: this.collidedPlatform.color
           });
379
           if (this.player.intersectsLeft(this.platformManager.platforms[i])) {
381         this.player.x = this.collidedPlatform.x - 64;
           for (i = 0; i < 10; i++) {
383         // SpawnParticles @PlayerPosition with intersecting Platform Color
           this.particles[(this.particlesIndex++) % this.particlesMax] = new Particle({
385         x: this.player.x + this.player.width,
           y: random(this.player.y, this.player.y + this.player.height),
387         velocityY: random(-30, 30),
           color: randomChoice(['#181818', '#181818', this.collidedPlatform.color])
389     });
           };
391
           // bounce player / push him away (effect)
393     this.player.velocityY = -10 + -(this.acceleration * 4);
           this.player.velocityX = -20 + -(this.acceleration * 4);
395     // this.jumpCount = 0;
           // this.acceleration = 0;
397     // this.accelerationTweening = 0;
           // this.scoreColor = '#181818';
399     // this.platformManager.maxDistanceBetween = 350;
           // this.platformManager.updateWhenLose();

```

```

401
403         } else {
405             // ----- Controller -----
406             // dragging: Mouse click & touch support
407             if (this.dragging || this.keys.SPACE || ↵
                this.keys.UP || this.keys.W) {
                this.player.velocityY = this.player.↵
                    jumpSize;
409                 this.jumpCount++;
411
                if (this.jumpCount > this.↵
                    jumpCountRecord) {
                    this.jumpCountRecord = this.↵
                        jumpCount;
413                }
                }
415
                /*if (keydown.keys.ESCAPE↵
                    ) {
                    //toggle;
                    InfinityRun.stop;
417
                }*/
419
            }
421
        }
423    };
425
    for (i = 0; i < this.platformManager.platforms.length ↵
        ; i++) {
        this.platformManager.update();
427    };
429
    for (i = 0; i < this.particles.length; i++) {
        this.particles[i].update();
431    };
    }
433
    };
435
    var selectedItem = 0;
437
    InfinityRun.keydown = function() {
439         if (InfinityRun.keys.ESCAPE && GameState==State.↵
            Started) {
                InfinityRun.clear();
441                GameState = State.Menu;

```

```

    } else if (InfinityRun.keys.ESCAPE && GameState === State.Menu) {
        GameState = State.Started;
        //InfinityRun.start();
    }
    if (InfinityRun.keys.UP) {
        //var prevSelected = this.selectedItem;
        selectedItem = (selectedItem + items.length - 1) % items.length;
    }
    if (InfinityRun.keys.DOWN) {
        selectedItem = (selectedItem + 1) % items.length;
    }

    if (InfinityRun.keys.ENTER) {
        callback(selectedItem);
    }
}

Menu = function() {
    //this.backgroundCallback = null;
}

//----- Draw -----
InfinityRun.draw = function() {
    if (GameState === State.Started) {
        this.player.draw();

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

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

        //Draw menu —TODO prototype
    } else if (GameState === State.Menu) {

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

```

```

487     callback = function(numItem) { if (numItem == 0) {
        GameState=State.Started };
        this.height = InfinityRun.height;
489     this.width = InfinityRun.width;
        this.size = 120;

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

497     this.textAlign = "center";
499     this.fillStyle = "White";

501     var height = 150;

503     if (this.title) {
        this.font = Math.floor(this.size*1.3).
            toString() + "px Times New Roman";
505     this.fillText(this.title, this.width/2,
        height);
        height+= height;

507     }

509     for (var i = 0; i < items.length; ++i)
    {
511         var size = Math.floor(this.size*0.8);
        if (i == selectedItem)
513         {
            //var v = Math.floor(127*Math.sin
            (GameLoopManager.lastTime
            *0.04) + 127);
            //this.fillStyle = "rgba(255,255,
            "+v.toString()+",255)";
            this.fillStyle = "#A9F5F2";
517             size = this.size+5;
        }
        this.font = size.toString() + "px Times
        New Roman";
        height += this.size;
521     this.fillText(items[i], InfinityRun.width
        /2, height);
        this.fillStyle = "White";

523     }

```

```

525     }

527

529     //Debug
    if (debug) {
531         this.font = '12pt Arial';
        this.fillStyle = '#181818';
533         this.fillText('Record: ' + this.jumpCountRecord,
            this.width - 150, 33);
        this.fillStyle = this.scoreColor;
535         //this.font = (12 + (this.acceleration * 3))+ 'pt
            Arial';
        this.fillText('Jumps: ' + this.jumpCount, this.
            width - 150, 50);
537         //todo distance = velocity * time (date:
            passed time between frames)
        this.fillText('Distance: ' + 0/* -TODO- */, this.
            width - 150, 65);
539         this.fillText('GameState: ' + GameState,
            this.width - 150, 80);
    }

541 };

543 InfinityRun.resize = function() {
545     /* todo Windowscale optimization
        *
547     *
        */
549 }

```

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">
    </head>
11 <body>
    <!--test github johannes-->
13    <!-- Game div -->
15 <div id="container">

17 </div>
    <!-->
19

21 <script type="text/javascript" src="js/game.js" charset=" ␣
    utf-8"></script>
    <script type="text/javascript" src="js/menu.js" charset=" ␣
    utf-8"></script>
23 <!--<script type="text/javascript" src="js/game_flat.js" ␣
    charset="utf-8"></script>-->

25 </body>
    </html>

```