

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	15
2.4	Dokumentation & Präsentation	15
3	Ausblick	17
4	Fazit	19
	Literaturverzeichnis	21
	Eidesstattliche Erklärung	23
A	Anhang	25
A.1	game.js	25
A.2	game.css	38
A.3	index.html	38

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

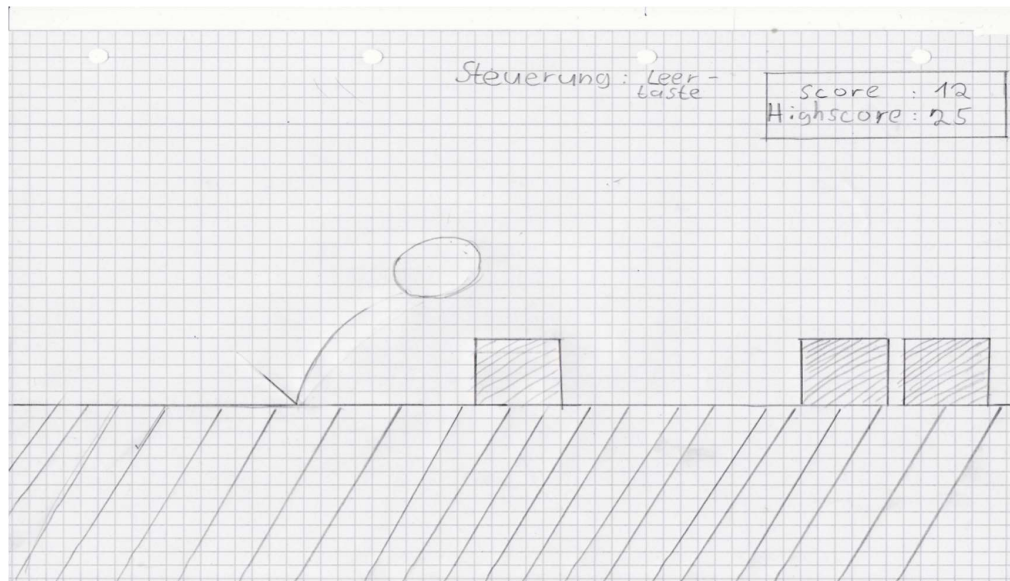


Abbildung 6.: Entwurfsskizze

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.

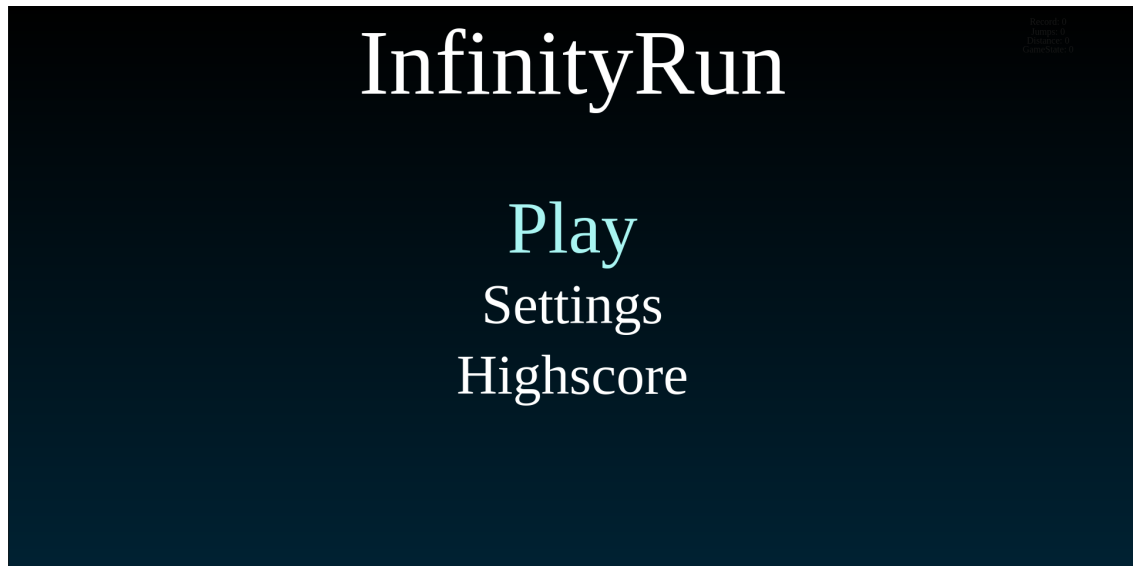


Abbildung 7.: Start Bildschirm



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

```
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

```
Player.prototype.update = function() {  
2 function PlatformManager()  
  {  
4  
      this.maxDistanceBetween = 300;
```

```
6      this.colors = ['#2ca8c2', '#98cb4a', '#f76d3c', '↵
      #f15f74', '#5481e6'];

8
9      //first 3 Platforms execept the Starter Platform
10     this.first = new Platform({
11         x: 300,
12         y: InfinityRun.width / 2,
13         width: 400,
14         height: 70
15     })
16
17     this.second = new Platform
18     ({
19         x: (this.first.x + this.first.width) + random(↵
20             this.maxDistanceBetween - 150, this.↵
21             maxDistanceBetween),
22         y: random(this.first.y - 128, InfinityRun.height ↵
23             - 80),
24         width: 400,
25         height: 70
26     })
27
28     this.third = new Platform
29     ({
30         x: (this.second.x + this.second.width) + random(↵
31             this.maxDistanceBetween - 150, this.↵
32             maxDistanceBetween),
33         y: random(this.second.y - 128, InfinityRun.height ↵
34             - 80),
35         width: 400,
36         height: 70
37     })
38
39     this.first.height = this.first.y + InfinityRun.↵
40         height;
41     this.second.height = this.second.y + InfinityRun.↵
42         height;
43     this.third.height = this.third.y + InfinityRun.↵
44         height;
```



```
36   this.first.color = randomChoice(this.colors);
37   this.second.color = randomChoice(this.colors);
38   this.third.color = randomChoice(this.colors);
39   this.colliding = false;
40   this.platforms = [this.first, this.second, this.third];
41 }
```

2.2.5.4. Update der Plattformen

```
PlatformManager.prototype.update = function()
2 {
3     this.first.x -= 3 + InfinityRun.acceleration;
4     if (this.first.x + this.first.width < 0)
5     {
6         this.first.width = random(450,
7             InfinityRun.width + 200);
8         this.first.x = (this.third.x + this.third
9             .width) + random(this.maxDistanceBetween - 150,
10             this.maxDistanceBetween);
11         this.first.y = random(this.third.y - 32,
12             InfinityRun.height - 80);
13         this.first.height = this.first.y +
14             InfinityRun.height + 10;
15         this.first.color = randomChoice(this.colors);
16     }
17
18     this.second.x -= 3 + InfinityRun.acceleration;
19     if (this.second.x + this.second.width < 0)
20     {
21         this.second.width = random(450,
22             InfinityRun.width + 200);
23         this.second.x = (this.first.x + this
24             .first.width) + random(this.maxDistanceBetween - 150,
25             this.maxDistanceBetween);
26     }
27 }
```

```

18         this.second.y = random(this.first.y - 32,
InfinityRun.height - 80);
20         this.second.height = this.second.y +
InfinityRun.height + 10;
22         this.second.color = randomChoice(this.
colors);
24     }

26     this.third.x -= 3 + InfinityRun.acceleration;
28     if (this.third.x + this.third.width < 0)
{
30         this.third.width = random(450,
InfinityRun.width + 200);
32         this.third.x = (this.second.x + this.
second.width) + random(this.
maxDistanceBetween - 150, this.
maxDistanceBetween);
34         this.third.y = random(this.second.y - 32,
InfinityRun.height - 80);
36         this.third.height = this.third.y +
InfinityRun.height + 10;
38         this.third.color = randomChoice(this.
colors);
40     }
42 };

```

2.2.5.5. Update der Plattformen

```

1     for (i = 0; i < this.platformManager.platforms.length;
2         i++)
3     {
4         if (this.player.intersects(this.
platformManager.platforms[i]))
5         {
6             this.collidedPlatform = this.
platformManager.platforms[i];
7             if (this.player.y < this.
platformManager.platforms[i].y)
8             {
9                 // ...
10            }
11        }
12    }

```

```
8      {
10          this.player.y = this.▷
            platformManager.▷
            platforms[i].y;
            // Gravity after ▷
            Collision with ▷
            Platform
10          this.player.velocityY = ▷
            0;
12      }
14      this.player.x = this.player.▷
            previousX;
14      this.player.y = this.player.▷
            previousY;
16      this.particles[(this.▷
            particlesIndex++) % this.▷
            particlesMax] = new Particle({
18      x: this.player.x,
            y: this.player.y + this.player.▷
            height ,
            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/>

[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. game.js

```

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

32 var State = { Menu:0, Started:1, Paused:2, Over:3 };
  var GameState = State.Menu;
34 var MainMenu;

36 var debug = true;

38 // randomizer
  function random(min, max) {

```

```

40     return Math.round(min + (Math.random() * (max - min))) >
        );
    }
42
    function randomChoice(array) {
44         return array[Math.round(random(0, array.length - 1))] >
            ];
    }
46

48 //initialize Sketch Framework
    var InfinityRun = Sketch.create({
50     fullscreen: true,
        width: 640,
52     height: 360,
        container: document.getElementById('container')
54 });

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

58 //Set X,Y,Width,Height
    function Vector2(x, y, width, height) {
60         this.x = x;
            this.y = y;
62         this.width = width;
            this.height = height;
64         this.previousX = 0;
            this.previousY = 0;
66     };

68
    // Set X,Y
70 Vector2.prototype.setPosition = function(x, y) {

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

74
        this.x = x;
76     this.y = y;

78 };
    // Set X
80 Vector2.prototype.setX = function(x) {

82     this.previousX = this.x;
        this.x = x;

84
    };

```

```
86 // Set Y
88 Vector2.prototype.setY = function(y) {
90     this.previousY = this.y;
91     this.y = y;
92 };
94 // Collision / Intersection Top
96 Vector2.prototype.intersects = function(obj) {
98     if (obj.x < this.x + this.width && obj.y < this.y + this.height &&
100         obj.x + obj.width > this.x && obj.y + obj.height > this.y) {
102         return true;
103     }
104     return false;
105 };
106 // Collision / Intersection Left
107 Vector2.prototype.intersectsLeft = function(obj) {
108     if (obj.x < this.x + this.width && obj.y < this.y + this.height) {
109         return true;
110     }
111     return false;
112 };
113 //————— Player—————
114 function Player(options) {
115     this.setPosition(options.x, options.y);
116     this.width = options.width;
117     this.height = options.height;
118     this.velocityX = 0;
119     this.velocityY = 0;
120     this.jumpSize = -13;
121     this.color = '#181818';
122 }
123 Player.prototype = new Vector2;
```

```

132 Player.prototype.update = function() {
    // Gravity
134     this.velocityY += 1;
    this.setPosition(this.x + this.velocityX, this.y +
        this.velocityY);
136
    if (this.y > InfinityRun.height || this.x + this.
        width < 0) {
138         this.x = 150;
        this.y = 50;
140         this.velocityX = 0;
        this.velocityY = 0;
142         InfinityRun.jumpCount = 0;
        InfinityRun.acceleration = 0;
144         InfinityRun.accelerationTweening = 0;
        InfinityRun.scoreColor = '#181818';
146         InfinityRun.platformManager.maxDistanceBetween =
            350;
        InfinityRun.platformManager.updateWhenLose();
148     }

150     if ((InfinityRun.keys.UP || InfinityRun.keys.SPACE ||
        InfinityRun.keys.W || InfinityRun.dragging) &&
        this.velocityY < -8) {
        this.velocityY += -0.75;
152     }

154 };

156 Player.prototype.draw = function() {
    InfinityRun.fillStyle = this.color;
158     InfinityRun.fillRect(this.x, this.y, this.width, this
        .height);
    };
160
    // ————— Platforms —————
162
    function Platform(options) {
164         this.x = options.x;
        this.y = options.y;
166         this.width = options.width;
        this.height = options.height;
168         this.previousX = 0;
        this.previousY = 0;
170         this.color = options.color;
    }
172

```



```

Platform.prototype = new Vector2;
174 Platform.prototype.draw = function() {
176     InfinityRun.fillStyle = this.color;
        InfinityRun.fillRect(this.x, this.y, this.width, this.
        .height);
178 };

180 // ————— Platform Manager —————

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

186
        //first 3 Platforms except the Starter Platform
188     this.first = new Platform({
        x: 300,
190     y: InfinityRun.width / 2,
        width: 400,
192     height: 70
        })
194     this.second = new Platform({
        x: (this.first.x + this.first.width) + random(
        this.maxDistanceBetween - 150, this.
        maxDistanceBetween),
196     y: random(this.first.y - 128, InfinityRun.height
        - 80),
        width: 400,
198     height: 70
        })
200     this.third = new Platform({
        x: (this.second.x + this.second.width) + random(
        this.maxDistanceBetween - 150, this.
        maxDistanceBetween),
202     y: random(this.second.y - 128, InfinityRun.height
        - 80),
        width: 400,
204     height: 70
        })

206     this.first.height = this.first.y + InfinityRun.height
        ;
208     this.second.height = this.second.y + InfinityRun.
        height;
        this.third.height = this.third.y + InfinityRun.height
        ;

```

```

210     this.first.color = randomChoice(this.colors);
211     this.second.color = randomChoice(this.colors);
212     this.third.color = randomChoice(this.colors);

214     this.colliding = false;

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

218 PlatformManager.prototype.update = function() {
220     this.first.x -= 3 + InfinityRun.acceleration;
222     if (this.first.x + this.first.width < 0) {
        this.first.width = random(450, InfinityRun.width +
            + 200);
224         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);
226         this.first.height = this.first.y + InfinityRun.
            height + 10;
        this.first.color = randomChoice(this.colors);
228     }

230     this.second.x -= 3 + InfinityRun.acceleration;
232     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);
234         this.second.y = random(this.first.y - 32,
            InfinityRun.height - 80);
        this.second.height = this.second.y + InfinityRun.
            height + 10;
236         this.second.color = randomChoice(this.colors);
    }

238     this.third.x -= 3 + InfinityRun.acceleration;
240     if (this.third.x + this.third.width < 0) {
        this.third.width = random(450, InfinityRun.width +
            + 200);
242         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);
244     this.third.height = this.third.y + InfinityRun.▷
        height + 10;
        this.third.color = randomChoice(this.colors);
246     }

248 };

250

252 // reset / new Game: set Starting Platform Parameters
PlatformManager.prototype.updateWhenLose = function() {
254     this.first.x = 300;
256     this.first.color = randomChoice(this.colors);
    this.first.y = InfinityRun.width / random(2, 3);
258     this.second.x = (this.first.x + this.first.width) + ▷
        random(this.maxDistanceBetween - 150, this.▷
        maxDistanceBetween);
    this.third.x = (this.second.x + this.second.width) + ▷
        random(this.maxDistanceBetween - 150, this.▷
        maxDistanceBetween);

260 };

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

264 function Particle(options) {
266     this.x = options.x;
    this.y = options.y;
268     this.size = 10;
    this.velocityX = options.velocityX || random(-(▷
        InfinityRun.acceleration * 3) + -8, -(InfinityRun.▷
        acceleration * 3));
270     this.velocityY = options.velocityY || random(-(▷
        InfinityRun.acceleration * 3) + -8, -(InfinityRun.▷
        acceleration * 3));
    this.color = options.color;
272 }

274 Particle.prototype.update = function() {
    this.x += this.velocityX;
276     this.y += this.velocityY;
    this.size *= 0.89;
278 };

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

```

```

282     InfinityRun.fillRect(this.x, this.y, this.size, this.size);
    };
284
    /**
286     InfinityRun.setup = function() {
288         this.jumpCount = 0;
290         this.acceleration = 0;
        this.accelerationTweening = 0;
292
        this.player = new Player({
294             x: 150,
            y: 30,
296             width: 32,
            height: 32
298         });

300         this.platformManager = new PlatformManager();

302         this.particles = [];
        this.particlesIndex = 0;
304         this.particlesMax = 20;
        this.collidedPlatform = null;
306         this.scoreColor = '#181818';
        this.jumpCountRecord = 0;
308
    };
310

312     InfinityRun.update = function() {
314         /*switch( GameState){
316             case State.Menu:
                //InfinityRun.stop();
318                 break;
            case State.Started:
320                 break;
            case State.Paused:
322                 break;
            case State.Over:
324                 break;
        }*/
326         if ( GameState == State.Started ) {
            this.player.update();
328

```

```

switch (this.jumpCount) {
330     case 10:
        this.accelerationTweening = 1;
332     this.platformManager.maxDistanceBetween = 430;
        //this.scoreColor = '#076C00';
334     break;
    case 25:
336     this.accelerationTweening = 2;
        this.platformManager.maxDistanceBetween = 530;
338     //this.scoreColor = '#0300A9';
        break;
340     case 40:
        this.accelerationTweening = 3;
342     this.platformManager.maxDistanceBetween = 580;
        //this.scoreColor = '#9F8F00';
344     break;
}

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

348

350 for (i = 0; i < this.platformManager.platforms.length; i++) {
352     if (this.player.intersects(this.platformManager.platforms[i])) {
        this.collidedPlatform = this.platformManager.platforms[i];
354     if (this.player.y < this.platformManager.platforms[i].y) {
        this.player.y = this.platformManager.platforms[i].y;

356         // Gravity after Collision with Platform
358         this.player.velocityY = 0;
    }

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

364     this.particles[(this.particlesIndex++) % this.particlesMax] = new Particle({
        x: this.player.x,
366         y: this.player.y + this.player.height,

```

```

        color: this.collidedPlatform.color
    });

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

    // bounce player / push him away (effect)
    this.player.velocityY = -10 + -(this.
    acceleration * 4);
    this.player.velocityX = -20 + -(this.
    acceleration * 4);
    // this.jumpCount = 0;
    // this.acceleration = 0;
    // this.accelerationTweening = 0;
    // this.scoreColor = '#181818';
    // this.platformManager.
    maxDistanceBetween = 350;
    // this.platformManager.updateWhenLose();

} 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++;
    }
}

```

```

400         if (this.jumpCount > this.jumpCountRecord) {
402             this.jumpCountRecord = this.jumpCount;
404         }
406         /*if (keydown.keys.ESCAPE) {
408             //toggle;
410             InfinityRun.stop;
412         }*/
414     };
416     for (i = 0; i < this.platformManager.platforms.length; i++) {
418         this.platformManager.update();
420     };
422     for (i = 0; i < this.particles.length; i++) {
424         this.particles[i].update();
426     };
428     var selectedItem = 0;
430     InfinityRun.keydown = function() {
432         if (InfinityRun.keys.ESCAPE && GameState === State.Started) {
434             InfinityRun.clear();
436             GameState = State.Menu;
438         } else if (InfinityRun.keys.ESCAPE && GameState === State.Menu) {
440             GameState = State.Started;
442             //InfinityRun.start();
444         }
446         if (InfinityRun.keys.UP) {
448             //var prevSelected = this.selectedItem;
450             selectedItem = (selectedItem + items.length - 1) % items.length;
452         }
454         if (InfinityRun.keys.DOWN) {

```

```

        selectedItem = (selectedItem + 1) % items.length;
442     }

444     if (InfinityRun.keys.ENTER) {
        callback(selectedItem);
446     }

448 }

450 Menu = function() {

452     //this.backgroundCallback = null;
    }
454

456 //————— Draw —————

458 InfinityRun.draw = function() {
    if (GameState == State.Started) {
460     this.player.draw();

462     for (i = 0; i < this.platformManager.platforms.length; i++) {
        this.platformManager.platforms[i].draw();
464     };

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

470

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

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

476

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

482     var lingrad = this.createLinearGradient(0,0,0,
        this.height);
    lingrad.addColorStop(0, '#000');
484     lingrad.addColorStop(1, '#023');

```



```

486     this.fillStyle = lingrad;
    this.fillRect(0,0,this.width, this.height)

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

490
    var height = 150;

492
    if (this.title) {
494         this.font = Math.floor(this.size*1.3).
            toString() + "px Times New Roman";
        this.fillText(this.title, this.width/2,
            height);
496         height+= height;
    }

498
    for (var i = 0; i < items.length; ++i)
500     {
        var size = Math.floor(this.size*0.8);
502         if (i == selectedItem)
        {
504             //var v = Math.floor(127*Math.sin
                (GameLoopManager.lastTime
                *0.04) + 127);
            //this.fillStyle = "rgba
                (255,255,"+v.toString()+",255)
                ";
506             this.fillStyle = "#A9F5F2";
            size = this.size+5;

508         }
        this.font = size.toString() + "px Times
            New Roman";
510         height += this.size;
        this.fillText(items[i], InfinityRun.width
            /2, height);
512         this.fillStyle = "White";
    }

514
    }

516

518
    //Debug
520    if (debug) {
        this.font = '12pt Arial';
522         this.fillStyle = '#181818';
        this.fillText('Record: ' + this.jumpCountRecord,
            this.width - 150, 33);

```

```

524     this.fillStyle = this.scoreColor;
        //this.font = (12 + (this.acceleration * 3))+'pt Arial';
526     this.fillText('Jumps:␣' + this.jumpCount, this.
        width - 150, 50);
        //todo distance = velocity * time (date:
        passed time between frames)
528     this.fillText('Distance:␣' + 0/* -TODO- */, this.
        width - 150, 65);
        this.fillText('GameState:␣' + GameState,
        this.width - 150, 80);
530 }
532 };

534 InfinityRun.resize = function() {
        /* todo Windowscale optimization
536     *
538     */
    }

```

A.2. 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.3. index.html

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD_XHTML_1.0_Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
        lang="en">
3 <head>
    <meta http-equiv="Content-Type" content="text/html;␣
        charset=utf-8">

```

```
5      <script type="text/javascript" src="js/sketch.min.js" charset="utf-8"></script>

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>
```