HOCHSCHULE FURTWANGEN UNIVERSITY HFU

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

Abbildungsverzeichnis

Tabellenverzeichnis

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

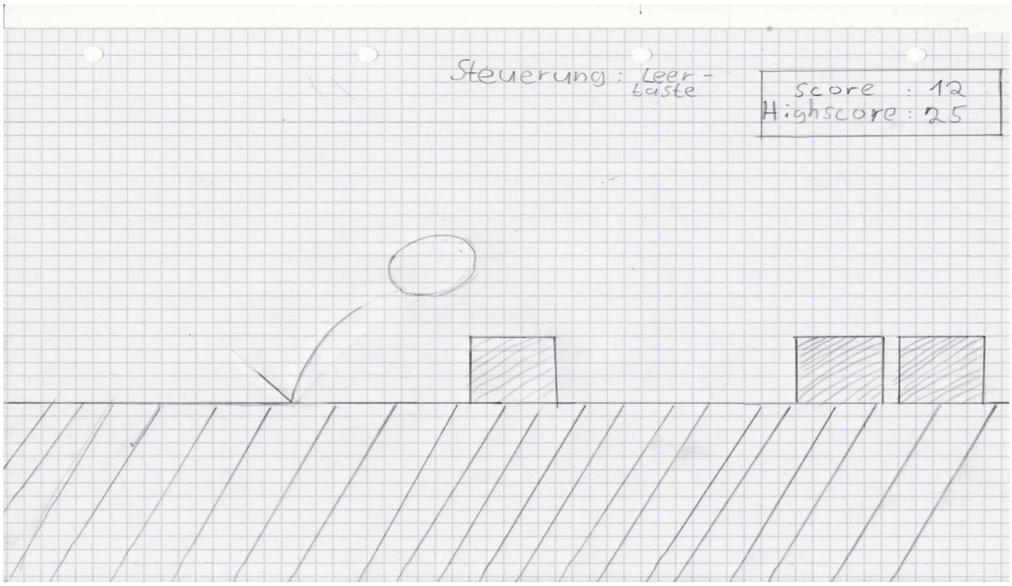

Abbildung 6.: Entwurfsskizze

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 geniert. 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 Perfomarnce 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.

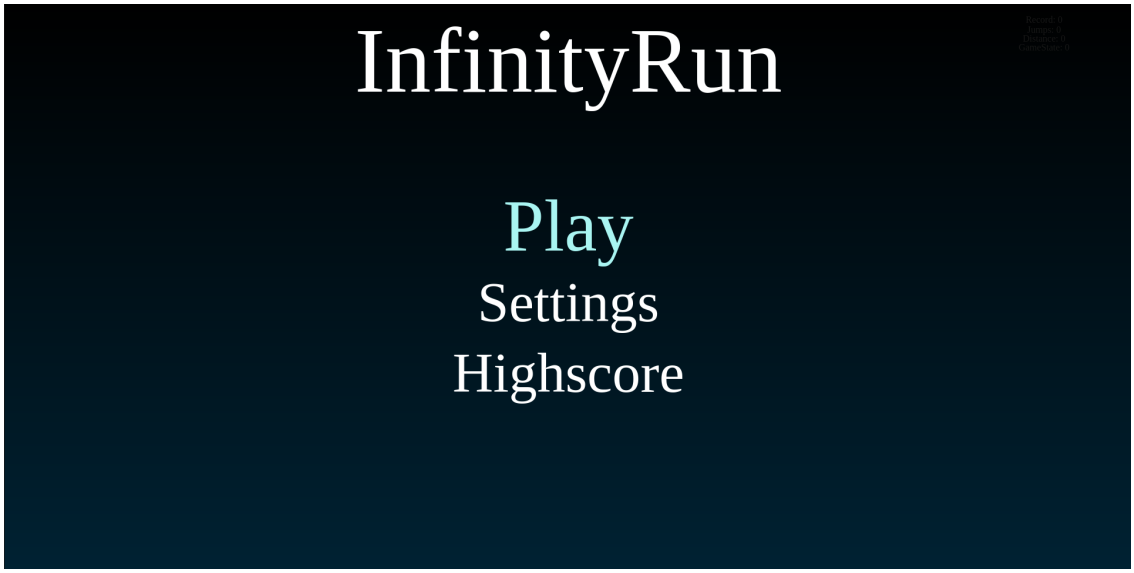

Abbildung 7.: Start Bildschirm

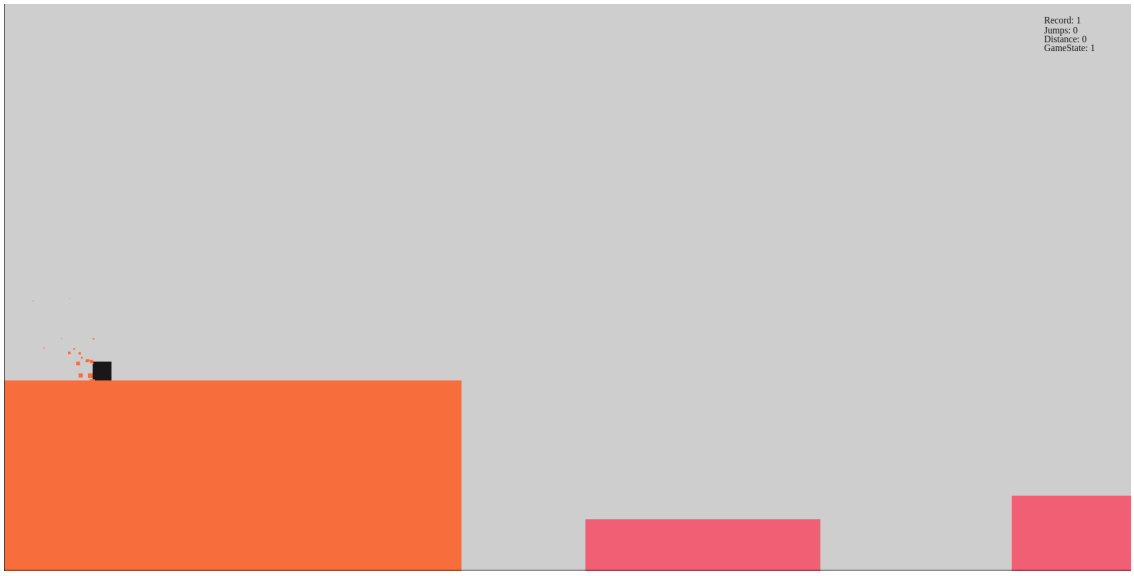

Abbildung 8.: Das Spiel

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:

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

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

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

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

Quelle: [sou]

2.2.5. Code

2.2.5.1. Framework initialisieren

```
var InfinityRun = Sketch.create({
fullscreen: true,
width: 640,
height: 360,
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() {
// Gravity
this.velocityY += 1;
this.setPosition(this.x + this.velocityX, this.y + this.velocityY);

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

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

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() {
function PlatformManager()
{

        this.maxDistanceBetween = 300;

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

//first 3 Platforms execept the Starter Platform
        this.first = new Platform({
        x: 300,
        y: InfinityRun.width / 2,
        width: 400,
        height: 70
})

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,
        height: 70
})

this.third = new Platform
({
```

```
            x: (this.second.x + this.second.width) + random(
                this.maxDistanceBetween - 150, this.
                maxDistanceBetween),
            y: random(this.second.y - 128, InfinityRun.height
                - 80),
            width: 400,
            height: 70
    })
            this.first.height = this.first.y + InfinityRun.
                height;
            this.second.height = this.second.y + InfinityRun.
                height;
            this.third.height = this.third.y + InfinityRun.
                height;
            this.first.color = randomChoice(this.colors);
            this.second.color = randomChoice(this.colors);
            this.third.color = randomChoice(this.colors);
            this.colliding = false;
            this.platforms = [this.first, this.second, this.
                third];
    }
```

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()
{
        this.first.x -= 3 + InfinityRun.acceleration;
        if (this.first.x + this.first.width < 0)
        {
                this.first.width = random(450,
                    InfinityRun.width + 200);
                this.first.x = (this.third.x + this.third
                    .width) + random(this.
```

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

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

        this.third.x -= 3 + InfinityRun.acceleration;
        if (this.third.x + this.third.width < 0)
        {
            this.third.width = random(450,
                InfinityRun.width + 200);
            this.third.x = (this.second.x + this.
                second.width) + random(this.
                maxDistanceBetween - 150, this.
                maxDistanceBetween);
            this.third.y = random(this.second.y - 32,
                 InfinityRun.height - 80);
            this.third.height = this.third.y +
```

```
                InfinityRun.height + 10;
            this.third.color = randomChoice(this.
                colors);
        }
};
```

2.2.5.5. Update der Plattformen

```
    for (i = 0; i < this.platformManager.platforms.length
        ; i++)
    {
            if (this.player.intersects(this.
                platformManager.platforms[i]))
            {
                this.collidedPlatform = this.
                    platformManager.platforms[i];
                if (this.player.y < this.
                    platformManager.platforms[i].y
                    )
                {
                    this.player.y = this.
                        platformManager.
                        platforms[i].y;
                    // Gravity after
                        Collision with
                        Platform
                    this.player.velocityY =
                        0;
                }

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

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

```
18                                  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),
 *
 * Put static values / vars into initialization function
 *
 * ---------------------------------
 * Design / Graphics
 *
 * Parallax Background?
 *
 * ---------------------------------
 * Menu
 * Menu draw in Input & draw prototypes
 * Handle / Manage CSS or HTML variables from JavaScript (Fullscreen,...)
 * ---------------------------------
 *
 * Platform Schematic? - Schematic files?
 * Different Themes depending on Progress?
 *
 * ---------------------------------
 * Test-Phase
 *
 * Controller: 'dragging' test Touch support
 * Browsertesting tools
 * eg.:
 * http://browserling.com/
 * http://browsershots.org/
 * https://crossbrowsertesting.com/
 * https://www.browserstack.com/
 */
var i = 0;

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

var debug = true;

// 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;
       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 &&
           obj.x + obj.width > this.x && obj.y + obj.height ↩
             > this.y) {
100        return true;
       }
102
       return false;
104 };

106 // Collision / Intersection Left
   Vector2.prototype.intersectsLeft = function(obj) {
108
       if (obj.x < this.x + this.width && obj.y < this.y + ↩
          this.height) {
110        return true;
       }
112
       return false;
114 };

116 //---------- Player ----------

118 function Player(options) {

120    this.setPosition(options.x, options.y);
       this.width = options.width;
122    this.height = options.height;
       this.velocityX = 0;
124    this.velocityY = 0;
       this.jumpSize = -13;
126    this.color = '#181818';

128 }

130 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;

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

// ———————— Platform Manager ————————

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


        //first 3 Platforms execept the Starter Platform
    this.first = new Platform({
        x: 300,
        y: InfinityRun.width / 2,
        width: 400,
        height: 70
    })
    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,
        height: 70
    })
    this.third = new Platform({
        x: (this.second.x + this.second.width) + random(this.maxDistanceBetween - 150, this.maxDistanceBetween),
        y: random(this.second.y - 128, InfinityRun.height - 80),
        width: 400,
        height: 70
    })

    this.first.height = this.first.y + InfinityRun.height;
    this.second.height = this.second.y + InfinityRun.height;
    this.third.height = this.third.y + InfinityRun.height;
```

```
210     this.first.color = randomChoice(this.colors);
        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;
        if (this.second.x + this.second.width < 0) {
232         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) {
328     this.player.update();
```

```
330         //endless increasing difficulty

332         /*
        this.accelerationTweening = 0.2 * this.jumpCount;
334         if (this.jumpCount>5) {
            this.platformManager.maxDistanceBetween = 300 + ↩
               2* this.jumpCount;
336         }
            */
338
        switch (this.jumpCount) {
340         case 10:
                this.accelerationTweening = 1;
342             this.platformManager.maxDistanceBetween = ↩
                   430;
                //this.scoreColor = '#076C00';
344             break;
            case 25:
346             this.accelerationTweening = 2;
                this.platformManager.maxDistanceBetween = ↩
                   530;
348             //this.scoreColor = '#0300A9';
                break;
350         case 40:
                this.accelerationTweening = 3;
352             this.platformManager.maxDistanceBetween = ↩
                   580;
                //this.scoreColor = '#9F8F00';
354             break;
        }
356
        this.acceleration += (this.accelerationTweening - ↩
           this.acceleration) * 0.01;
358


360
        for (i = 0; i < this.platformManager.platforms.length↩
           ; i++) {
362         if (this.player.intersects(this.platformManager.↩
               platforms[i])) {
                this.collidedPlatform = this.platformManager.↩
                   platforms[i];
364             if (this.player.y < this.platformManager.↩
                   platforms[i].y) {
                    this.player.y = this.platformManager.↩
                       platforms[i].y;
366
```

```
                // Gravity after Collision with Platform
368             this.player.velocityY = 0;
            }
370
            this.player.x = this.player.previousX;
372         this.player.y = this.player.previousY;

374         this.particles[(this.particlesIndex++) % this
               .particlesMax] = new Particle({
                x: this.player.x,
376             y: this.player.y + this.player.height,
                color: this.collidedPlatform.color
378         });

380         if (this.player.intersectsLeft(this.
               platformManager.platforms[i])) {
                this.player.x = this.collidedPlatform.x -
                    64;
382             for (i = 0; i < 10; i++) {
                    // SpawnParticles @PlayerPostion with
                        intersecting Platform Color
384                 this.particles[(this.particlesIndex
                       ++) % this.particlesMax] = new
                       Particle({
                        x: this.player.x + this.player.
                           width,
386                     y: random(this.player.y, this.
                           player.y + this.player.height)
                           ,
                        velocityY: random(-30, 30),
388                     color: randomChoice(['#181818', '
                           #181818', this.
                           collidedPlatform.color])
                    });
390             };

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

```
402
                } else {
404
                    // ---------- Controller ----------
406                 // dragging: Mouse click & touch support
                    if (this.dragging || this.keys.SPACE ||
                       this.keys.UP || this.keys.W) {
408                     this.player.velocityY = this.player.
                           jumpSize;
                        this.jumpCount++;
410
                        if (this.jumpCount > this.
                           jumpCountRecord) {
412                         this.jumpCountRecord = this.
                               jumpCount;
                        }
414                 }
                                /*if (keydown.keys.ESCAPE
                                   ) {
416                                 //toggle;
                                    InfinityRun.stop;
418                             }*/

420             }

422         }
        };
424
        for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
426         this.platformManager.update();
        };
428
        for (i = 0; i < this.particles.length; i++) {
430         this.particles[i].update();
        };
432 }

434 };

436 var selectedItem = 0;

438 InfinityRun.keydown = function() {
        if (InfinityRun.keys.ESCAPE && GameState==State.
           Started) {
440                 InfinityRun.clear();
                    GameState = State.Menu;
```

```
442         } else if (InfinityRun.keys.ESCAPE && GameState==
               State.Menu) {
                    GameState = State.Started;
444                 //InfinityRun.start();
            }
446         if (InfinityRun.keys.UP) {
                    //var prevSelected = this.selectedItem;
448                 selectedItem = (selectedItem + items.
                       length - 1) % items.length;
            }
450         if (InfinityRun.keys.DOWN) {
                    selectedItem = (selectedItem + 1) % items
                       .length;
452         }

454         if(InfinityRun.keys.ENTER) {
                    callback(selectedItem);
456         }

458 }

460 Menu = function() {

462         //this.backgroundCallback = null;
    }
464

466 //---------- Draw ----------

468 InfinityRun.draw = function() {
            if(GameState == State.Started) {
470     this.player.draw();

472     for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
            this.platformManager.platforms[i].draw();
474     };

476     //Draw particles
        for (i = 0; i < this.particles.length; i++) {
478         this.particles[i].draw();
        };
480
            //Draw menu --TODO prototype
482         } else if (GameState == State.Menu) {

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

```
486
        callback = function(numItem) { if (numItem == 0) GameState=State.Started };
488     this.height = InfinityRun.height;
        this.width = InfinityRun.width;
490     this.size = 120;

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

498     this.textAlign = "center";
        this.fillStyle = "White";
500
        var height = 150;
502
        if (this.title) {
504             this.font = Math.floor(this.size*1.3).toString() + "px Times New Roman";
                this.fillText(this.title, this.width/2, height);
506             height+= height;
        }
508
        for (var i = 0; i < items.length; ++i)
510     {
                var size = Math.floor(this.size*0.8);
512             if (i == selectedItem)
                {
514                     //var v = Math.floor(127*Math.sin(GameLoopManager.lastTime
                        *0.04) + 127);
                        //this.fillStyle = "rgba(255,255,"+v.toString()+",255)";
516                     this.fillStyle = "#A9F5F2";
                        size = this.size+5;
518             }
                this.font = size.toString() + "px Times New Roman";
520             height += this.size;
                this.fillText(items[i], InfinityRun.width/2, height);
522             this.fillStyle = "White";
        }
```

```
524
            }
526


528
        //Debug
530     if (debug) {
            this.font = '12pt Arial';
532         this.fillStyle = '#181818';
            this.fillText('Record: ' + this.jumpCountRecord,
                this.width - 150, 33);
534         this.fillStyle = this.scoreColor;
            //this.font = (12 + (this.acceleration * 3))+'pt
                Arial';
536         this.fillText('Jumps: ' + this.jumpCount, this.
                width - 150, 50);
                    //todo distance = velocity * time (date:
                        passed time between frames)
538         this.fillText('Distance: ' + 0/* -TODO- */, this.
                width - 150, 65);
                    this.fillText('GameState: ' + GameState,
                        this.width - 150, 80);
540     }

542 };

544 InfinityRun.resize = function() {
        /* todo Windowscale optimization
546      *
         *
548      */
    }
```

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;
```

```
}
```

A.3. index.html

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

    <title>Infinity Run</title>

    <link href="css/game.css" rel="stylesheet" type="text
        /css">
</head>
<body>
<!--test github johannes-->

<!-- Game div -->
<div id="container">

</div>
<!---------------->


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

</body>
</html>
```