HOCHSCHULE FURTWANGEN UNIVERSITY HFU

Projektarbeit Informatik Workshop

im Studiengang

Allgemeine Informatik

InfintyRun

Jump 'n' Run Spiel

Referent	:	**Gabriela Mai**
Vorgelegt am	:	16. 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

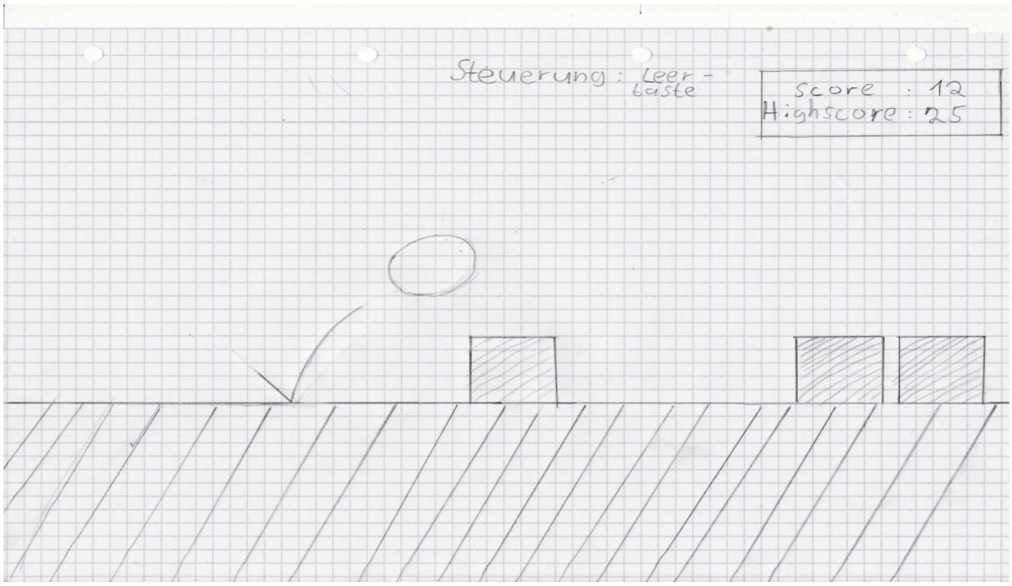

Abbildung 6: Entwurfsskizze

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.

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.

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.

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.

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 Code

2.2.1.1 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) {
    return Math.round(min + (Math.random() * (max - min)));
}

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


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

//------- Vector [Get/Set] Functions ---------

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


// 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() {
133     // Gravity
    this.velocityY += 1;
135     this.setPosition(this.x + this.velocityX, this.y + this.velocityY);

137     if (this.y > InfinityRun.height || this.x + this.width < 0) {
            this.x = 150;
139         this.y = 50;
            this.velocityX = 0;
141         this.velocityY = 0;
            InfinityRun.jumpCount = 0;
143         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;
        }

    };

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

    // ---------- Platforms ----------

    function Platform(options) {
        this.x = options.x;
        this.y = options.y;
        this.width = options.width;
        this.height = options.height;
        this.previousX = 0;
        this.previousY = 0;
        this.color = options.color;
    }

    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;
    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];
}

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);
```

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

329 switch (this.jumpCount) {
        case 10:
331         this.accelerationTweening = 1;
            this.platformManager.maxDistanceBetween =
                430;
333         //this.scoreColor = '#076C00';
            break;
335     case 25:
```

```
            this.accelerationTweening = 2;
337         this.platformManager.maxDistanceBetween =
               530;
            //this.scoreColor = '#0300A9';
339         break;
        case 40:
341         this.accelerationTweening = 3;
            this.platformManager.maxDistanceBetween =
               580;
343         //this.scoreColor = '#9F8F00';
            break;
345     }

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


349


351     for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
         if (this.player.intersects(this.platformManager.
            platforms[i])) {
353          this.collidedPlatform = this.platformManager.
                platforms[i];
             if (this.player.y < this.platformManager.
                platforms[i].y) {
355              this.player.y = this.platformManager.
                    platforms[i].y;

357              // Gravity after Collision with Platform
                 this.player.velocityY = 0;
359          }

361          this.player.x = this.player.previousX;
             this.player.y = this.player.previousY;
363
             this.particles[(this.particlesIndex++) % this
                .particlesMax] = new Particle({
365              x: this.player.x,
```

```
                y: this.player.y + this.player.height,
367             color: this.collidedPlatform.color
            });
369
            if (this.player.intersectsLeft(this.
               platformManager.platforms[i])) {
371             this.player.x = this.collidedPlatform.x -
                    64;
                for (i = 0; i < 10; i++) {
373                 // SpawnParticles @PlayerPostion with
                        intersecting Platform Color
                    this.particles[(this.particlesIndex
                       ++) % this.particlesMax] = new
                       Particle({
375                     x: this.player.x + this.player.
                           width,
                        y: random(this.player.y, this.
                           player.y + this.player.height)
                           ,
377                     velocityY: random(-30, 30),
                        color: randomChoice(['#181818', '
                           #181818', this.
                           collidedPlatform.color])
379                 });
                };
381
                // bounce player / push him away (effect)
383             this.player.velocityY = -10 + -(this.
                   acceleration * 4);
                this.player.velocityX = -20 + -(this.
                   acceleration * 4);
385             // this.jumpCount = 0;
                // this.acceleration = 0;
387             // this.accelerationTweening = 0;
                // this.scoreColor = '#181818';
389             // this.platformManager.
                   maxDistanceBetween = 350;
                // this.platformManager.updateWhenLose();
391
```

```
393             } else {

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

401                     if (this.jumpCount > this.
                            jumpCountRecord) {
                            this.jumpCountRecord = this.
                                jumpCount;
403                     }
                    }
405                                 /*if (keydown.keys.ESCAPE
                                        ) {
                                            //toggle;
407                                         InfinityRun.stop;
                                    }*/

409
                }
411
            }
413     };

415     for (i = 0; i < this.platformManager.platforms.length
            ; i++) {
            this.platformManager.update();
417     };

419     for (i = 0; i < this.particles.length; i++) {
            this.particles[i].update();
421     };
    }
423
    };
```

```
425
var selectedItem = 0;
427
InfinityRun.keydown = function() {
429     if (InfinityRun.keys.ESCAPE && GameState==State.
        Started) {
                InfinityRun.clear();
431             GameState = State.Menu;
         } else if (InfinityRun.keys.ESCAPE && GameState==
            State.Menu) {
433             GameState = State.Started;
                //InfinityRun.start();
435      }
         if (InfinityRun.keys.UP) {
437             //var prevSelected = this.selectedItem;
                selectedItem = (selectedItem + items.
                   length - 1) % items.length;
439      }
         if (InfinityRun.keys.DOWN) {
441             selectedItem = (selectedItem + 1) % items
                   .length;
         }
443
         if(InfinityRun.keys.ENTER) {
445             callback(selectedItem);
         }
447
}
449
Menu = function() {
451
         //this.backgroundCallback = null;
453 }


455
//---------- Draw ----------
457
InfinityRun.draw = function() {
459      if(GameState == State.Started) {
```

```
        this.player.draw();
461
        for (i = 0; i < this.platformManager.platforms.length
           ; i++) {
463         this.platformManager.platforms[i].draw();
        };
465
        //Draw particles
467     for (i = 0; i < this.particles.length; i++) {
            this.particles[i].draw();
469     };

471         //Draw menu --TODO prototype
            } else if (GameState == State.Menu) {
473
            this.title = "InfinityRun";
475         items = ["Play", "Settings", "Highscore"];

477         callback = function(numItem) { if (numItem == 0)
               GameState=State.Started };
            this.height = InfinityRun.height;
479         this.width = InfinityRun.width;
            this.size = 120;
481
            var lingrad = this.createLinearGradient(0,0,0,
               this.height);
483         lingrad.addColorStop(0, '#000');
            lingrad.addColorStop(1, '#023');
485         this.fillStyle = lingrad;
            this.fillRect(0,0,this.width, this.height)
487
            this.textAlign = "center";
489         this.fillStyle = "White";

491         var height = 150;

493         if (this.title) {
                    this.font = Math.floor(this.size*1.3).
                       toString() + "px Times New Roman";
```

```
495                 this.fillText(this.title, this.width/2,
                        height);
                    height+= height;
497             }

499             for (var i = 0; i < items.length; ++i)
                {
501                 var size = Math.floor(this.size*0.8);
                    if (i == selectedItem)
503                 {
                        //var v = Math.floor(127*Math.sin
                            (GameLoopManager.lastTime
                            *0.04) + 127);
505                     //this.fillStyle = "rgba
                            (255,255,"+v.toString()+",255)
                            ";
                        this.fillStyle = "#A9F5F2";
507                     size = this.size+5;
                    }
509                 this.font = size.toString() + "px Times
                        New Roman";
                    height += this.size;
511                 this.fillText(items[i], InfinityRun.width
                        /2, height);
                    this.fillStyle = "White";
513             }

515             }


517


519         //Debug
            if (debug) {
521             this.font = '12pt Arial';
                this.fillStyle = '#181818';
523             this.fillText('Record: ' + this.jumpCountRecord,
                    this.width - 150, 33);
                this.fillStyle = this.scoreColor;
```

```
525         //this.font = (12 + (this.acceleration * 3))+'pt Arial';
            this.fillText('Jumps: ' + this.jumpCount, this.width - 150, 50);
527                 //todo distance = velocity * time (date: passed time between frames)
            this.fillText('Distance: ' + 0/* -TODO- */, this.width - 150, 65);
529                 this.fillText('GameState: ' + GameState, this.width - 150, 80);
    }
531
};
533
InfinityRun.resize = function() {
535     /* todo Windowscale optimization
        *
537     *
        */
539 }
```

2.2.1.2 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 }
```

2.2.1.3 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>
```

2.3 Test

2.4 Dokumentation & Präsentation

3 Ausblick

4 Fazit

Literaturverzeichnis

[Sch11] Schlosser, Joachim: *Wissenschaftliche Arbeiten schreiben mit LATEX: Leitfaden für Einsteiger: Joachim Schlosser*. 4. Heidelberg and München and Landsberg and Frechen and Hamburg : mitp, 2011. – ISBN 9783826691027

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 16. November 2016 Florian Durli

FURTWANGEN, den 16. November 2016 Jannik Ivosevic

FURTWANGEN, den 16. November 2016 Johannes But

FURTWANGEN, den 16. November 2016 Marco Mayer

FURTWANGEN, den 16. November 2016 Koray Emtekin