UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

COMPUTERVISUALISTIK

Entwicklung eines interaktiven Spiels unter besonderer Berücksichtigung des Gamedesigns

Bachelorarbeit

zur Erlangung des Grades einer Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von

Scarlett Grenzemann

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Anna Katharina Hebborn, M.Sc. (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im November 2013

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....
(Ort, Datum) (Unterschrift)

Summary

This paper is about how to create a little interactive video game, concerning the principles of game design. This contains the development of a concept, its implementation and the application of the most important game design rules. It also presents the problems that occur in programming and how to solve these. The result is the prototype of a working game.

Inhaltsverzeichnis

1 Einleitung

Heutzutage sind Videospiele so sehr in der Gesellschaft etabliert, dass sie ein Teil unserer Kultur geworden sind. Die Spieleindustrie hat sogar das Niveau der Filmindustrie erreicht und macht Umsätze in Millionenhöhe. Die meisten Spiele sind groß angelegte Projekte mit riesigen Teams bestehend aus Programmierern, Animatoren, Künstlern, Musikern, Autoren, teilweise sogar Schauspielern. Es steckt viel Arbeit und Aufwand in einem guten Spiel, und ich möchte mit dieser Arbeit einen Schritt in die Welt der Spieleentwicklung wagen.

1.1 Motivation

Videospiele haben mich mein Leben lang begleitet und geprägt und sind auch der Grund, warum ich Computervisualistik studiere. Nach meinem Studium möchte ich in der Spieleindustrie Fuß fassen, weshalb ich mithilfe dieser Bachelorarbeit die Chance nutzen möchte erste Erfahrungen im Programmieren von Videospielen und dem Thema Game Design zu machen. Dieses Wissen möchte ich während meiner Masterarbeit erweitern, sodass ich gute Referenzen und Erfahrungen aufweisen kann, wenn ich in die Berufswelt eintrete.

1.2 Entwicklung eines interaktiven Spiels

Es erfordert zweierlei Dinge, um ein Spiel zu entwickeln; ein Konzept und eine geeignete Programmiersprache. Dazu stellen sich automatisch folgende Fragen: Welche Erfahrung soll der Spieler machen? Was macht das Spiel spielenswert? Welches Erscheinungsbild eignet sich für das Spiel? Womit kann ich mein Spiel realisieren? Bevor man also überhaupt anfangen kann zu programmieren, muss man zuvor ein paar grundlegende Entscheidungen treffen.

1.3 Auswahl der Programmiersprache/ Entwicklungsumgebung

Die Auswahl an Programmiersprachen ist groß, deshalb ist es vorerst wichtig zu wissen, was man will und was man dementsprechend braucht, um seine Ziele umzusetzen. Ich wollte ein Spiel in klassischer 2D Grafik, mit Sprites und dem typischen Pixel-Design. Also suchte ich nach einer Entwicklungsumgebung, die dies realisieren kann und stieß auf LÖVE und Game Maker. Zu beiden werde ich mich später genauer äußern und auch erklären, warum ich mich für Game Maker entschied.

1.4 Auswahl der Literatur zum Thema Game Design

Natürlich gibt es zu einem Thema wie Game Design sehr viel Literatur. Auf der Suche nach der für mich geeigneten fand ich drei Werke. Game Design von Bob Bates, Fundamentals of Game Design von Ernest Adams und The Art of Game Design von Jesse Schell. Jesse Schells hat mir besonders gefallen. Der Autor hat eine besondere Herangehensweise an dieses umfassende Thema, die nie überfordernd wirkt und immer nachvollziehbar bleibt. Er benutzt auch gerne Anekdoten aus seinem Erfahrungsschatz um ein bestimmtes Problem nahezulegen. Meiner Arbeit liegen zum Thema Game Design diese drei Bücher zugrunde.

1.5 Zielsetzung

Ziel meiner Arbeit ist es, ein eigenes Spiel von Grund auf zu konzipieren und im Rahmen meiner Fähigkeiten umzusetzen. Dabei möchte ich besonders auf das Game Design eingehen und versuchen, die wichtigsten Regeln in mein Spiel einfließen zu lassen. Meine Lernerfolge möchte ich dem Leser verständlich darlegen und diese auch am eigenen Spiel anwenden und präsentieren. Die Arbeit soll Schritt für Schritt zeigen, wie man an die Umsetzung eines Spiels herangeht und auch die Probleme behandeln, die dabei entstehen. Das Ergebnis soll ein funktionierendes Spiel sein, welches in einem gewissen Umfang den Game Design Regeln entspricht.

2 Grundlagen

In diesem Kapitel soll grundlegendes Wissen vermittelt werden, das zum besseren Verständnis dieser Bachelorarbeit dient. Ich selbst habe mich in diese Themen einarbeiten müssen, weshalb es wichtig ist einen kurzen Überblick über die Begrifflichkeiten und die Entwicklungsumgebung zu haben.

2.1 Game Maker

Game Maker ist eine integrierte Entwicklungsumgebung (IDE) von YoYo Games, mit der man Videospiele per Drag & Drop Icons oder mithilfe einer Skriptsprache namens GML (Game Maker Language) programmieren kann. Ich habe die Version Game Maker 8.1 Lite verwendet, die kostenfrei ist und natürlich in ihren Fähigkeiten begrenzt; beispielsweise ist das Partikelsystem nicht nutzbar. Die Benutzeroberfläche ist so konzipiert, dass man jedes Element im Spiel in einem eigenen Ordner bearbeiten kann. So gibt es zum Beispiel einen Ordner für Sprites, in dem man per Mausklick ein neues Sprite erstellt. Hierfür können verschiedenste Einstellungen vorgenommen werden, wie die Größe, das Koordinatensystem, den Namen des Sprites. Wenn man einen Sprite erstellen und nicht laden will, gibt es dafür auch ein Tool. Darin sind sogar Animationen möglich. Insgesamt gibt es folgende Ordner: Sprites, Sounds, Backgrounds, Paths, Scripts, Fonts, Time Lines, Objects und Rooms. Paths dient dazu einer spielbaren Figur in einem bestimmten Moment im Spiel einen Weg vorzugeben, den sie ablaufen soll. Das ist recht gebräuchlich in 2D Videospielen, wenn zum Beispiel der Spieler an einen bestimmten Ort gehen soll, weil dort ein Ereignis stattfindet. In Scripts hat man die Möglichkeit, alle Funktionen des Spiels in GML zu programmieren. Time Lines sind sehr nützlich, wenn Ereignisse in einem bestimmten Moment im Spiel eintreten sollen. Wie schon erwähnt, kann man in Game Maker nicht nur die Skriptsprache GML zum programmieren nutzen, sondern auch die Drag & Drop Icons. Die Ordnerhierachie befindet sich wie gewohnt links auf der Benutzeroberfläche, während die Icons auf der rechten Seite liegen, wenn man mit Objekten arbeitet. Objekte besitzen Events und Actions. Man kann verschiedene Events wählen, beispielsweise Create, Destroy, Keyboard oder Collision. Sobald also ein Event eintritt, werden Actions ausgeführt. Dafür nutzt man schließlich die Icons. Die Icons sind durch Tabs in verschiedene Themenbereiche eingeteilt: move, main1, main2, control, score, extra, draw. Da diese Begriffe nicht viel über den Inhalt verraten, werde ich kurz auf jeden Einzelnen eingehen. In move werden die Bewegungen des Objektes festgelegt. Das heißt konstante Bewegungen in x oder y Richtung (Move); Springen (Jump); Pfade, die abgelaufen werden sollen (Paths); oder Schritte auf ein Objekt zu oder um eines herum (Steps). main1 beinhaltet die Kategorien Objects, Sprite, So-

unds und Rooms. Hier können beispielsweise Regeln für das Erstellen von Instanzen, das Verändern von Sprites oder das Einspielen von Geräuschen festgelegt werden. Man kann auch Ereignisse für die verschiedenen Räume wählen, in denen sich die Figur bewegt; zum Beispiel, dass der Raum neu gestartet oder in einen anderen gewechselt werden soll. main2 hat folgende Abschnitte: Timing, Info, Game, Resources. Timing behandelt die Time Lines; mit Info kann man innerhalb des Spiels Nachrichten hinterlassen; Game hat Funktionen wie Restart oder End Game; in Resources kann man Sprites oder Sounds ersetzen (Resources funktioniert nur in der Standard Edition). control wird wohl am häufigsten gebraucht. Hier hat man die Kategorien Questions, Other, Code und Variables. Questions ersetzt if-Anweisungen oder Ähnliches, die man beim Programmieren nutzen würde. Zum Beispiel Check Empty oder Test Expression. Other dient dazu, die Questions in Blöcke einzubinden oder einen Else-Zweig anzuhängen. Code gibt einem die Möglichkeit, etwas in GML zu programmieren. Variables behandelt Variablen auf verschiedene Arten, man kann zum Beispiel einer Variable einen Wert zuweisen oder eine Vergleichsoperation ausführen. score bietet die Möglichkeit, Score, Lives und Health ins Spiel einzubauen. extra ist der Standard Edition vorbehalten, hier gibt es ein Partikelsystem (Particles) und die Möglichkeit, Einstellungen für das Abspielen einer CD (CD) und für einen Cursor einzustellen (Other). Der letzte Tab, draw, hat die Funktionen Drawing, Settings und Other. Drawing und Settings kann man nutzen, um Dinge wie Sprites oder geometrische Formen zu zeichnen und ihnen zum Beispiel eine Farbe zuzuweisen. Other ist nur in der Standard Edition verfügbar und ermöglicht Schnappschüsse und Effekte. Damit wären alle wichtigen Funktionen von Game Maker abgedeckt. Wie man sieht, ist die IDE sehr umfangreich und bietet eine gute Grundlage, um selbst ein Spiel zu programmieren.

2.2 Game Design

Game Design ist ein breitgefächertes Gebiet, welches ich so allumfassend nicht in meiner Bachelorarbeit behandeln kann und möchte. Ich konzentriere mich nur auf die Kernpunkte, die die Autoren in ihren Werken zu Anfang darlegen und die ich für wichtig erachte. Da jeder Autor eine andere Herangehensweise hat, möchte ich zu jedem einzelnen dessen Game Design-Konzept erklären.

2.2.1 Fundamentals of Game Design, Second Edition von Ernest Adams

The essential elements of a game are play, pretending, a goal and rules.

"Die essentiellen Elemente eines Spiels sind Spielen, Vortäuschen, ein Ziel und Regeln." Im Grunde bedeutet dies lediglich folgendes: Ein Spiel benötigt Regeln, die der Spieler einhalten muss. Ein Spiel benötigt Ziele, damit der Spieler einen Ansporn hat. Mit einem Spiel muss man spielen können, das heißt der Spieler interagiert mit dem Spiel und hat Einfluss darauf. Und ein Spiel soll uns eine Welt vortäuschen, es muss eine Immersion entstehen; der Spieler tut so, als sei er Teil einer fiktionalen Welt sobald er in ein Spiel einsteigt.

Design Rules:
You can't please everyone.
Gameplay comes first.
Aesthetics are important, too.
Strive for harmony.
Risks need rewards.

Ernest Adams stellt außerdem ein paar Design-Regeln auf. Erstens: Du kannst es nicht jedem Recht machen. Das bedeutet, versuche nicht ein Spiel zu erschaffen, dass jedem gefällt. Das ist unmöglich, also verschwende nicht deine Zeit damit. Zweitens: Gameplay steht an erster Stelle. So schön und aufregend eine gute Grafik oder eine tolle Story sein können, das Gameplay ist der Kern des Spiels und sollte die meiste Aufmerksamkeit bekommen. Drittens: Ästhetik ist auch wichtig. Nach dem Gameplay kommt die Ästhetik, denn auch davon lebt ein gutes Spiel. Wenn der Spieler sich dauerhaft unansehnliche Grafiken ansehen muss, verliert er auch dadurch den Spaß. Es geht hier nicht um realistische Darstellungen oder Ähnliches, sondern rein um schönes Design. Dies kann auch 2D Grafik à la Super Mario sein. Viertens: Streben nach Harmonie. Gib dem Spieler das Gefühl, dass alles irgendwie zusammen gehört. Von der Benutzeroberfläche bis zur Storyline. Der rote Faden muss erkennbar sein. Fünftens: Risiken erfordern Belohnungen. Lass den Spieler nicht etwas im Spiel riskieren, wenn er dafür keine Belohnung kriegt. Nichts ist ärgerlicher als das.

Games provide gameplay, that is, challenges and actions that entertain.

Kommen wir noch zum wichtigsten Element eines Spiels, dem Gameplay. Das Wort Gameplay ist schwer zu definieren, deshalb hat Adams es in zwei Bereiche aufgeteilt: Herausforderungen und Handlungen. Ein gutes Gameplay besteht also aus diesen beiden, was bedeutet das? Ein Spiel sollte den Spieler herausfordern. Er soll denken: Das klingt schwer, aber

ich beweise mir selbst, dass ich das schaffe. Der Spieler wird gerne herausgefordert, solange er in Erwartung eines Erfolgs ist. Und das Spiel soll dem Spieler die Möglichkeit geben, zu handeln, Einfluss zu haben, zu interagieren. Der Spieler hat Spaß daran ein Flugzeug zu steuern, ein Schloss zu bauen oder zu singen und zu tanzen. Was Spiele interessant macht ist, dass sie uns die Möglichkeit geben Dinge zu tun, die wir in der Realität nicht machen können.

An action game is one in which the majority of challenges presented are tests of the player's physical skills and coordination.

Adams hält für jedes Spielgenre ein eigenes umfassendes Kapitel bereit. So kam ich nicht umhin, mein Spiel in eines dieser Kategorien einordnen zu wollen. Ein Action Game scheint meinem Spiel am ehesten zu entsprechen. Action Games sind beispielsweise Spiele wie Tetris oder Super Mario Bros. Sie bestehen zum größten Teil aus Herausforderungen, die die physischen Fähigkeiten und die Koordination des Spielers testen. Da mein Spiel ein Geschicklichkeitsspiel ist, bei dem man schnell handeln und zwei Figuren parallel steuern muss (dazu später mehr), entspricht es deshalb ziemlich genau der Definition eines Action Games.

Noch eine kleine Anmerkung am Schluss: Ernest Adams enpfiehlt Studenten, die zum ersten Mal ein Spiel programmieren sollen, die Engine Game Maker. Er ist der Meinung, dass das ein sehr guter Einstieg in das Thema Game Design und der Entwicklung eines eigenen Spiels mit Hilfe der Game Design Regeln ist. Ich war sehr überrascht und fühlte mich in meiner Entscheidung, ein 2D Spiel zu programmieren, bestätigt.

Nachdem ich nun einige Erfahrungen mit Game Maker machen konnte, kann ich sagen, dass man auch mit dieser Engine auf jeden Fall Verständnis und Übung für das Programmieren braucht. Sonst bleibt es bei einem wirklich sehr minimalistischen Spiel, das man eventuell mit Hilfe eines Tutorials in Game Maker lernt.[1]

[1] Die behandelten Themen umfassen die Kapitel 1 Games and Videogames, Kapitel 9 Gameplay und Kapitel 13 Action Games. [REF01]

2.2.2 Game Design, Second Edition von Bob Bates

Principles of Game Design:

Player empathy
Feedback
Grounding the player
The moment-to-moment experience
Immersion
Writing
Design within limits
Removing impediments (disc swapping, load times, game interruptions, saving the game, housekeeping, bugs)
Interface design
The start-up screen
Customizable controls
Cheat codes
Tutorial or practice mode
Structure and progression
Taking care of the player (Dead Man Walking, protect newbies,"Play it again, Sam", give the player the information he needs, reduce player paranoia, offer levels of difficulty)

Bob Bates behandelt das Thema Game Design anders als Ernest Adams. Er beschäftigt sich hauptsächlich mit dem Spieler und seiner Erfahrung mit dem Spiel.

Prinzipien des Game Designs:

- Spieler-Empathie

  Der Designer muss sich in den Spieler hineinversetzen. Er muss sich fragen, was der Spieler denkt, wenn er das Spiel spielt.

- Feedback

  Der Spieler interagiert mit dem Spiel. Deshalb möchte er auch Feedback erfahren. Er möchte nicht etwas tun und anschließend keine Antwort bekommen. Egal ob negatives oder positives Feedback, es muss Feedback geben für jede Handlung des Spielers.

- Erdung des Spielers

  Der Spieler muss immer wissen, wo er sich befindet. In der Welt und in der Geschichte. Er muss sein Ziel kennen und wer er ist. Er muss den Verlauf des Spiels nachvollziehen können und wissen, warum er tut was er tut.

- Die Moment-zu-Moment Erfahrung

  Der Spieler könnte jederzeit das Spiel beenden und etwas Anderes tun. Deshalb soll ein Spiel die Aufmerksamkeit des Spielers auf sich ziehen und beibehalten, und ihn in jedem Moment des Spiels unterhalten können. Der Spieler muss immer eine interessante Aufgabe haben. Das Schlimmste, was passieren kann, ist, dass er sich langweilt. Man muss also dem Spieler Entscheidungen gewähren, die signifikante Folgen im Spiel habe. Deshalb ist es nützlich, sich die Handlungen des Spielers als Verben vorzustellen. Je mehr Verben der Spieler zur Verfügung hat (beispielsweise bewegen, schießen, klettern), umso mehr Aufgaben wird er erhalten. Es sind die Handlungen die den Kern guten Gameplays ausmachen und eine positive Moment-zu-Moment Erfahrung ermöglichen. Um diese Erfahrung nicht zu zerstören sollte Folgendes vermieden werden: das Wiederholen von Handlungen; langes Wandern durch die Spielewelt (anstatt ihn hin- und her laufen zu lassen, sollte er eine schnellere Alternative haben); lange Dialoge oder Filmsequenzen, die nicht übersprungen werden können; lange Einarbeitungszeiten, in denen der Spieler erst einmal seine Figur erstellen muss bevor er spielen darf. Ein Spiel soll unterhaltsam sein, also muss es interessant bleiben. Der Spieler soll viel zu tun haben, allerdings sollten diese Dinge auch Spaß machen.

- Immersion

  Der Spieler soll in die Welt eintauchen können, die der Designer erschaffen hat. Er soll sich als Teil der Welt fühlen, in der er handelt und interagiert.

- Geschichten schreiben

  Jedes Spiel erzählt eine Geschichte. Der Designer muss die Fähigkeit besitzen, die Geschichte, die er erzählen möchte, dem Spieler glaubwürdig und spannend zu präsentieren. Nicht nur das, die Geschichte muss in das Szenario passen, das er gewählt hat und muss einen interessanten Verlauf haben. Für diese Aufgabe werden meistens professionelle Autoren herangezogen.

- Design innerhalb der Grenzen

  Das Design eines Spiels muss sich in einem gewissen Rahmen aufhalten. Es sind nur bestimmte Technologien gegeben und dementsprechend wird das Design gestaltet.

- Entfernen von Hindernissen

  Das Austauschen von CDs; lange Ladezeiten; Spielunterbrechungen durch Game Over; keine oder begrenzte Speichermöglichkeiten;

schlechte Organisation des Spiels (Pause, Speichern, Hilfe nicht vorhanden); Bugs (Fehler im Spiel). All dies führt dazu, dass der Spielspaß verloren geht. Der Spieler möchte jederzeit pausieren und speichern können. Der Spieler möchte nicht, wenn er mitten in der Geschichte ist, darauf hin fiebernd was passiert, eine CD wechseln müssen um weiterspielen zu können. Er möchte nicht ewig auf den Ladebalken warten. Die Immersion geht dadurch verloren und der Spielspaß sinkt.

- Benutzeroberflächen-Design

  Das Benutzeroberflächen-Design, also das Interface-Design, muss angemessen gestaltet werden. Es soll übersichtlich und leicht verständlich sein.

- Startbildschirm

  Der Startbildschirm muss dem Spieler die Möglichkeit geben, ein neues Spiel zu beginnen, einen Spielstand zu laden, in einem Tutorial üben zu können und verschiedene Einstellungen an der Steuerung oder dem Ton vornehmen zu dürfen.

- Anpassbare Steuerung

  Der Spieler muss die Möglichkeit haben, die Steuerung seinen Wünschen anzupassen.

- Cheat Codes

  Jedes Spiel braucht Cheat Codes, die den Spieler unbesiegbar werden lassen. Es ist egal, wenn die Herausforderung dadurch verloren geht, lass ihm diese Möglichkeit.

- Tutorial- oder Übungsmodus

  Es muss jederzeit die Möglichkeit geben, in einem Tutorial oder einer Übung die Steuerung auszuprobieren, ohne dass es Konsequenzen im Spiel selbst hat.

- Struktur und Fortschritt

  Ein Spiel sollte einfach zu erlernen, aber schwer zu meistern sein. Das bedeutet ganz einfach, dass das Spiel einen leichten Einstieg hat und mit fortschreitendem Verlauf schwieriger wird. Erst lernt er die wichtigsten Handlungen und probiert diese aus. Schritt für Schritt, wenn er mehr Erfahrung hat, erkennt er, wie er es besser machen könnte und probiert wieder neue Dinge aus. Er muss erkennen, dass es einen Fortschritt im Spiel gibt und eine gewisse Struktur, also dass er erst mit der Zeit schwierigere Aufgaben bekommt. Das Interesse des Spielers ist sehr fragil: Entweder das Spiel wird mit der Zeit

zu einfach oder es ist zu Beginn zu schwer. Man muss zu Anfang des Spiels die Aufmerksamkeit des Spielers gewinnen und dabei auf einem einfachen Niveau bleiben, damit er weiterspielen möchte. Es ist wie in Kinofilmen: Die ersten Minuten müssen den Zuschauer für den Film begeistern können, ansonsten verliert er das Interesse. Wenn man den Spieler zu Beginn nicht in das Spiel involvieren kann, wird er es nie wieder spielen wollen. Deswegen ist es so wichtig, ein Spiel von unabhängigen Personen testen zu lassen. Der Game Designer selbst kennt alle Tricks, alle Stärken und Schwächen des Spiels, und kann deshalb nicht beurteilen, ob ein Spiel zu schwer ist.

- Sich um den Spieler kümmern

  Das sogenannte "Dead Man Walking" sollte konsequent vermieden werden. Es kann sehr ärgerlich sein zu erfahren, dass man zwei Level zuvor einen Gegenstand verpasst hat, den man braucht um in genau diesem Moment voran zu kommen. Der Spieler irrt herum und weiß nicht, was er falsch gemacht hat, und das ist sehr frustrierend. Neulinge im Spiel, zum Beispiel in Online-Spielen, sollten nicht damit bestraft werden, dass sie eben Neulinge sind. Es macht keinen Spaß, wenn man weiß, dass man bereits verloren hat sobald ein paar stärkere Spieler einem begegnen. Deswegen sollten neue Mitspieler geschützt werden. Spiele sollten den Spieler nicht dazu zwingen, etwas wieder und wieder zu tun. Es ist frustrierend und macht keinen Spaß, wenn man immer nur ein kleines Stück voran kommt, um dann erneut zu verlieren und von Neuem zu beginnen. Der Spieler sollte zu jeder Zeit die Informationen bekommen, die er braucht, um voran zu kommen. Wenn er nicht weiß, was er tun soll, spielt er es nicht weiter. Spieler wissen oft nicht, ob sie das Richtige tun. Das Spiel sollte ihnen also regelmäßig die Bestätigung geben, dass sie auf dem richtigen Weg sind oder ob sie vom Pfad abkommen. Um die Motivation des Spielers zu steigern, ist es gut, verschiedene Schwierigkeitsgrade einzubringen. Der Spieler soll entscheiden können in welchem Modus er spielen möchte.

How to design:
Create an integrated whole
Economy of design
Where do you get your ideas

Wie man entwirft:

- Ein integriertes Ganzes erschaffen

  Das Spiel muss einen roten Faden aufweisen. Ein gutes Design weiß das zu vermitteln. Die verschiedenen Elemente müssen eine Einheit bilden.

- Sparsamkeit des Designs

  Gutes Design benötigt nur das Wichtigste. Der Designer muss sich vor Augen führen, was er braucht und setzt nur diese Elemente ein. Überflüssiges wird ausgelassen.

- Woher bekommt man die Ideen?

  Der Designer sollte sich mit einem Thema beschäftigen, das ihn persönlich interessiert. Dann kommen die Ideen ganz von allein. Und nur so entsteht auch ein gutes Spiel. Es bringt nichts, sich in einem Genre aufzuhalten, mit dem man nichts am Hut hat. Man sollte auch (Spiel-) Erfahrung mit Spielen in dem Genre haben, mit dem man sich beschäftigt.[2]

2.2.3 The Art of Game Design von Jesse Schell

The designer creates an experience
The game is not the experience
The lens of essential experience

Jesse Schell setzt den Fokus auf den Game Designer. Alles beginnt mit ihm und er benötigt bestimmte Fähigkeiten in verschiedensten Bereichen; wie Animation, Architektur, kreatives Schreiben, Geschichte, Mathematik, Musik. Eine einzelne Person kann natürlich nicht all diese Fähigkeiten vereinen, weshalb es wichtig ist in Teams zu arbeiten. Die wichtigste Fähigkeit jedoch, die er haben muss, ist das Zuhören. Es gibt fünf Arten des Zuhörens: Man muss seinem Team zuhören; man muss seinem Publikum zuhören; man muss seinem Spiel zuhören (der Game Designer muss es in- und auswendig kennen); man muss dem Kunden zuhören; und man muss sich selbst zuhören (was für die Meisten die schwierigste Aufgabe ist). Es ist also wichtig, wirklich zu wissen was gefordert wird von jeder Einzelnen dieser Gruppen.

[2] Die behandelten Themen umfassen das Kapitel 2 Principles of Game Design. [REF02]

- Der Designer erschafft eine Erfahrung

  Wofür nutzt der Game Designer nun seine Fähigkeiten? Man könnte sagen: Um ein Spiel zu erschaffen. Aber das ist falsch. Das Spiel ist nur ein Medium, ein Artefakt. Spiele sind wertlos, solange sie nicht gespielt werden. Der Game Designer erschafft also kein Spiel, sondern eine Erfahrug. Das ist es, was ihn beschäftigt.

- Das Spiel ist nicht die Erfahrung

  Das Spiel ermöglicht die Erfahrung, aber das Spiel selbst ist nicht die Erfahrung. Darüber muss man sich im Klaren sein. Game Designer beschäftigen sich nur damit, was zu existieren scheint. Denn nur der Spieler und das Spiel sind real, die Erfahrung ist imaginär. Diese Erfahrung ist der Grund, warum Menschen Spiele spielen. Weil jeder Mensch diese Erfahrung anders erlebt und sie nicht mit anderen geteilt werden kann, ist es so wichtig, dass der Game Designer auf einer tieferen Ebene zuhören kann.

- Die Linse der essentiellen Erfahrung

  Schell benutzt in seinem Buch das Wort Linse als Merksatz. In jedem Kapitel gibt es weitere Linsen, die das Wichtigste zusamenfassen. Die erste Linse, die er einführt, ist die Linse der essentiellen Erfahrung. Dies bedeutet Folgendes: Der Game Designer muss sich fragen, 1. welche Erfahrung er für den Spieler erschaffen will, 2. was die Essenz dieser Erfahrung ist und 3. wie das Spiel diese Essenz einfangen kann.

The experience rises out of a game:
The lens of surprise
The lens of fun
The lens of curiousity
The lens of endogenous value
The lens of problem solving

Die Erfahrung geht aus dem Spiel hervor:

- Die Linse der Überraschung

  Schell's Linsen stellen immer Fragen an den Game Designer. Die Linse der Überraschung enthält folgende: 1. Was wird die Spieler überraschen wenn sie das Spiel spielen? 2. Hat das Spiel überraschende Momente (in der Geschichte), Spielregeln, Grafiken, Technologien? 3. Ermöglichen die Spielregeln den Spielern, sich gegenseitig zu überraschen? 4. Oder sich selbst?

- Die Linse des Spaßes

  Hier muss sich der Game Designer fragen, 1. welche Teile des Spiels Spaß machen und warum und 2. welche Teile mehr Spaß machen müssen.

- Die Linse der Neugierde

  Der Game Designer muss sich überlegen, 1. welche Fragen im Spiel beim Spieler aufkommen, 2. was er tun muss damit dem Spieler diese Fragen wichtig sind und 3. was er tun kann damit der Spieler noch mehr Fragen stellt.

- Die Linse des unentbehrlichen Wertes

  Es muss klar sein, 1. was für die Spieler im Spiel von Wert ist, 2. wie man diese Dinge noch wertvoller machen kann und 3. was die Verbindung zwischen den Werten im Spiel und der Motivation des Spielers ist.

- Die Linse des Problem-Lösens

  Der Game Designer fragt sich 1. welche Probleme der Spieler im Spiel lösen muss, 2. ob die versteckten zu lösenden Probleme als Teil des Gameplays erscheinen und 3. wie das Spiel neue Probleme generieren kann, sodass der Spieler immer zurück kommt.

Dies sind also die Erfahrungen, die der Game Designer beim Spieler erzeugen möchte. Es gilt alle Linsen zu beachten, um die bestmögliche Erfahrung zu erschaffen. Schell legt außerdem vier Elemente fest, aus denen ein Spiel besteht.

The four basic elements:

Mechanics
Story
Aesthetics
Technology

Die vier grundlegenden Elemente:

- Mechanik

  Die Mechanik besteht aus den Prozessen und Regeln, die das Spiel ausmachen. Sie beschreiben das Ziel des Spiels und was die Spieler versuchen können (oder nicht versuchen sollten), um jenes zu erreichen und was passiert, wenn sie es versuchen.

- Geschichte

  Die Geschichte ist die Sequenz von Ereignissen, die sich im Spiel entfaltet. Sie kann linear und fest vorgeschrieben sein oder verzweigt mit verschiedenen Verläufen.

- Ästhetik

  Die Ästhetik des Spiels ist ein wichtiger Bestandteil der Erfahrung, die der Spieler macht. Die Ästhetik entsteht aus dem Aussehen und dem Klang des Spiels, und dem Gefühl beim Spielen selbst.

- Technologie

  Die Technologie ist das Medium, durch das die Mechanik stattfindet, die Geschichte erzählt wird und die Ästhetik entsteht. Technologie umfasst alles, womit das Spiel möglich gemacht wird; egal ob mit Stift und Papier, einem Brettspiel oder einer hoch entwickelten Software.[3]

2.2.4 Zusammenfassung

Fassen wir also die wichtigsten Punkte zusammen. Ein Spiel hat Regeln, die eingehalten werden müssen und Ziele, die erreicht werden wollen. Ein Spiel soll eine Immersion erschaffen, der Spieler soll sich als Teil einer fiktionalen Welt fühlen. Das Wichtigste in einem Spiel ist das Gameplay, der Spieler möchte mit Hilfe von bestimmten Fähigkeiten handeln und interagieren können um die Herausforderungen und Probleme zu meistern, denen er sich stellen muss und will. Abgesehen vom Gameplay spielt die Ästhetik eine wichtige Rolle. Wie ein Spiel aussieht, klingt, sich anfühlt beeinflusst ebenso den Spielspaß wie das Gameplay. Außerdem muss ein Spiel harmonisch und der rote Faden erkennbar sein, um ein integriertes Ganzes zu ergeben. Der Spieler muss immer wissen, wo in der Welt er sich befindet und was seine Aufgabe ist. Wenn er etwas riskiert, möchte er belohnt werden und für jede Handlung ein Feedback erhalten. Ein Spiel erzeugt eine Erfahrung, und um diese soll es beim Spielen schlussendlich gehen. Jedes Spiel erzählt eine Geschichte, die passend gewählt wird und sich während des Spielens entfaltet. Zu guter Letzt hat ein Spiel außerdem die Eigenschaft, durch verschiedene Schwierigkeitsgrade und einen hohen Wiederspielwert den Spieler herauszufordern und dazu zu bringen, es immer wieder aufs Neue spielen zu wollen, ohne dabei von Anfang an zu schwierig zu sein.

[3] Die behandelten Themen umfassen die Kapitel 1 In The Beginning, There Is the Designer, Kapitel 2 The Designer Creates an Experience, Kapitel 3 The Experience Rises Out of a Game und Kapitel 4 The Game Consists of Elements. [REF03]

3 Konzeption und Implementierung

Ich wollte im Rahmen meiner Bachelorarbeit ein eigenes Spielkonzept entwickeln und umsetzen, und mir dafür die notwendigen Kenntnisse in der erforderten Programmiersprache und zum Thema Game Design aneignen. Ich überlegte also, welches Konzept interessant genug sein könnte, das gleichzeitig die Regeln des Game Designs erfüllt und im Rahmen meiner Fähigkeiten liegt. Ich hatte die Idee ein Spiel zu programmieren, in dem zwei Figuren gleichzeitig gesteuert werden. Das sagte mir sehr zu und schien ein spannender Ansatz zu sein, da die Herausforderung des Multi-Taskings ein gutes Gameplay und einen hohen Wiederspielwert versprachen. Konkret funktioniert es folgendermaßen: Zwei unabhängig voneinander spielbare Figuren müssen zwei Pfade entlanglaufen. Das Spiel soll von der Vogelperspektive betrachtet werden. Die Herausforderung besteht darin, nicht vom Pfad herunterzufallen, während man zwei Charaktere gleichzeitig steuert und den Gefahren auf diesem Weg zu entkommen versucht (in Form von Gegnern oder Hindernissen auf dem Weg). Natürlich kann man die Figuren auch abwechselnd steuern, allerdings muss man dann darauf achten, dass die andere Figur nicht in dem Moment von einem Gegner attackiert wird. Sollte eine der Figuren herunterfallen, wird das Level neu gestartet. Dieses Spielkonzept wollte ich so gut wie möglich umsetzen und zusätzlich noch ein paar Dinge einbringen, die es spannender machen könnten. Wie zum Beispiel einen Countdown oder eine gewisse Anzahl an Versuchen vor dem Game Over, sowie Hindernisse auf dem Weg der Figur oder Gegner die auf ihn zu fliegen. Oder ein Punktesystem, für das man Items einsammeln kann. Dazu wollte ich mir noch eine interessante und sinnvolle Geschichte ausdenken, die die Motivation des Spiels unterstützt. In diesem Kapitel werde ich getrennt voneinander die Implementierung und die Konzeption behandeln, da ich nicht alles umsetzen konnte was ich mir vorgestellt habe.

3.1 Konzeption und Game Design

Als ich mich in das Thema Game Design einarbeitete, habe ich versucht mein Spielkonzept so detailliert wie möglich aufzuschreiben. Zur Orientierung nahm ich einen Artikel von der Website making games, der einen recht guten Überblick darüber gibt, was ein Game Design Konzept beinhalten muss. Im Folgenden zeige ich, welche Ideen ich damals zu dem Spiel hatte. Natürlich ist dieses Spielkonzept nicht mehr aktuell, ich habe beispielsweise andere Spielfiguren gewählt und bereits neue Ideen für den Spielverlauf entwickelt. Trotz alledem möchte ich einen Überblick darüber geben, was die anfänglichen Überlegungen für das Spiel waren.[4]

[4] Aufbau des Spielkonzepts nach dem Vorbild eines Artikels der Website making games [REF04]

3.2 Anfängliches Konzept und erste Ideen

3.2.1 Beschreibung der Features

- Zwei Pfade, die sich vor den Figuren in Form von Affen schrittweise aufbauen.
- Bananenstauden, die Auslöser oder Savepoints darstellen und den weiteren Weg freigeben oder den bereits gegangenen speichern.
- Hindernisse in Form von Bananenschalen und Kokosnüssen.
- Lianen, die dem Spieler einen Vorsprung verschaffen.
- Ein Vogel, der dem Spieler seinen Fortschritt anzeigt oder versucht, den Spieler abzulenken.
- Bananen zum Einsammeln für ein weiteres Leben.
- Das Spiel startet bei 3 Leben; Game Over wird ausgelöst bei 0 Leben, danach gibt es einen Neustart mit 3 Leben; dies kann 3 Mal wiederholt werden, danach muss man komplett neu beginnen.

3.2.2 Zusammenhänge der Features

- Die Pfade bauen sich parallel zueinander auf, es dauert immer ein paar Sekunden bis das nächste Feld sichtbar wird.
- In der Eigenschaft eines Auslösers sorgt die Bananenstaude dafür, dass sich der Pfad weiter aufbaut, wenn sie von einer Figur aufgesammelt wird. Als Savepoint baut sich der Pfad erst weiter auf, wenn die Bananenstaude von beiden Figuren aufgesammelt wurde; hierbei wird der bisher begangene Pfad gespeichert, sodass man dorthin zurückgelangt sollte man vom Pfad herunterfallen. Savepoints befinden sich nur auf Blöcken. Auslöser hingegen auf losen und festen Platten.
- Die Figur kann über die Bananenschalen springen; sollte sie sie berühren, rutscht sie aus (und landet auf einem Feld in der unmittelbaren Umgebung); der Spieler kann herunterfallende Kokosnüsse erahnen, wenn ein Schatten zu sehen ist und muss rechtzeitig ausweichen, ansonsten wird er für einige Sekunden bewegungsunfähig.
- Ab und zu tauchen Lianen auf dem Pfad auf; wenn der Spieler diese berührt kann er einen Teil des Pfades überspringen (beide Figuren müssen die Liane erreicht haben, sodass sie zusammen vorankommen).
- Der Vogel erscheint an den Savepoints oder zwischendurch, um dem Spieler anzuzeigen wie weit er bereits ist oder um ihn abzulenken.

- Die Bananen liegen auf dem Pfad; wenn die Figur sie einsammelt, kann sie Punkte sammeln; ab einer bestimmten Bananenanzahl erhält man ein Leben.

3.2.3 Liste der zu entwickelnden Grafiken

- Figuren - Affe, Vogel
- Objekte - Bananenstaude, Banane, Bananenschale, Kokosnuss, Liane
- Pfad - Platten oder Baumstämme und Äste; Start-/Endpunkt ist eine große Platte oder eine Baumkrone; zwischendurch gibt es auch feste Blöcke und Pfade, die nicht verschwinden können
- innere Welt - ein Dschungel, der entweder nur mit einer grünen Textur um den Pfad herum dargestellt wird, oder ein richtiger Untergrund aus anderen Bäumen, Pflanzen, Ästen (Vogelperspektive, also sieht man den Dschungel von oben)
- äußere Welt - ein Dschungel auf einer Insel aus der Vogelperspektive in Miniaturansich,t um das nächste Level auszusuchen
- Score - Zeit, Bananenanzahl, Leben in Form eines Affenkopfes
- Animationen - die Bewegungen des Affen; das Einsammeln der Bananenstaude/ Bananen; das Ausrutschen auf der Bananenschale; das Aufblinken der Figur, wenn sie von einer Kokosnuss getroffen wird; das Schwingen mit der Liane; das Erscheinen der Liane/ Kokosnuss/ des Vogels; das Aufbauen des Pfades vor der Figur; das Bewegen von Level zu Level in der äußeren Welt; der Zeitablauf im Score; Erhöhung der Bananenanzahl im Score; Erhöhung der Leben im Score; das Herunterfallen der Figur vom Pfad; das Verschwinden von Objekten (Kokosnuss, Bananenschale)

3.2.4 Klarheit über die benötigte Technologie

Eventuell sind C++ oder Java gut geeignet, da die Grafik einfach bleiben soll. Vielleicht wäre 2D-Pixel-Grafik passend, wie in den alten Spieleklassikern. Falls nötig, eine Game Engine die dafür konzipiert ist; wie zum Beispiel Game Maker oder LÖVE.

3.2.5 Spielgenre

Ein Mix aus Jump'n'Run und Geschicklichkeitsspiel.

3.2.6 Plot

Zwei Affen wollen ein Rennen zum Rand des Dschungels machen und wieder zurück sein bevor die Sonne untergeht. Der Jüngere ist klein und wendig, aber nicht so schnell. Der Ältere ist schneller, aber weniger geschickt. Beide bringen durch ihre Fähigkeiten Vor- und Nachteile ins Spiel mit ein. Der Vogel ist quasi ein Schiedsrichter. Er beobachtet den Stand des Rennens oder gibt Kommentare ab. Die Affen müssen einander helfen, um an den Rand des Dschungels zu gelangen und so können nur beide zusammen das Ziel erreichen. Wenn man ein bestimmtes Level erreicht hat, tritt die Dämmerung ein und man erreicht bald das Ziel. In den Leveln danach geht es dann darum, zurück zu sein bevor die Sonne untergegangen ist. Während der einzelnen Level läuft auch die Zeit zur Orientierung rückwärts ab (im Score).

3.2.7 Spielmechanismus

Der Spieler startet mit beiden Figuren links und rechts am Startblock. Sobald er den Auslöser (die Bananenstaude) einsammelt, baut sich der Weg vor ihm auf und das Level startet. Am Anfang sollen es zwei Felder sein, sobald der Spieler fortschreitet wird der Weg im Sekundentakt um ein Feld erweitert. Ein Feld bedeutet hier eine Zeile. Diese Zeile kann aus 1-5 Platten bestehen. Am Startblock bestehen die ersten 2 Zeilen aus 3 Platten. Zusätzlich verschwinden die Felder hinter den Figuren zeilenweise. Jedes Feld soll für eine gewisse Anzahl an Sekundentakten existieren. Es gibt allerdings auch fest eingebaute Platten, die dauerhaft vorhanden sind. Auch Blöcke wie am Amfang und Ende des Levels kommen zwischendurch vor. Jedes Level hat einen eigenen vorgegebenen Weg, der Schwierigkeitsgrad erhöht sich um jedes neue Level. Der Spieler steuert beide Figuren gleichzeitig. Er muss sowohl die Forwärtsbewegung als auch das Ausweichen nach links und rechts steuern. Außerdem muss er über Hindernisse springen, die ihm begegnen (Bananenschalen). Er kann natürlich auch wieder zurücklaufen. Eine mögliche Tastenbelegung wäre z.B. W-A-S-D und die Pfeiltasten, um die Figuren zu bewegen und die Leertaste um zu springen. Um den Spieler nicht zu überfordern springen bei Druck auf die Leertaste beide Charaktere gleichzeitig. Es gäbe auch die Möglichkeit, das Spiel mit einem Controller zu spielen, der zwei bewegliche Analogsticks besitzt. Um zu springen, genüge ein Druck auf den Stick. So könnten beide Figuren eigenständig bleiben. Auf dem Weg können den Figuren einige Objekte begegnen. Die Bananen können eingesammelt werden, um ein neues Leben, also einen weiteren Spielversuch, zu erhalten. Bei eine bestimmten Anzahl Bananen gibt es ein Leben. Lianen werden an einigen Stellen im Level eingebaut, um dem Spieler einen Vorsprung zu verschaffen. Er muss aber mit beiden Figuren auf beiden Seiten die Liane ergreifen. So gelangt er

mit beiden Figuren einige Felder vorwärts. An gewissen Punkten auf den Wegen begegnet der Spieler auch Hindernissen wie Bananenschalen und Kokosnüssen. Die Bananenschalen liegen fest auf dem Weg und können mit Forwärtsbewegung+Leertaste übersprungen werden. Sollte der Spieler sie berühren, rutscht die Figur auf ihr aus; das bedeutet, sie landet auf einem Feld in der näheren Umgebung. Die Entscheidung, auf welchem Feld er landet, wird zufällig berechnet. Auch die Umgebung außerhalb des Pfades wird berücksichtigt, sodass die Figur im schlimmsten Fall hinunterfallen kann. Die Kokosnüsse sind ebenfalls fest eingebaut. Sobald der Spieler einen bestimmten Punkt erreicht, fällt einige Felder vor ihm die Kokosnuss herunter. Er kann mit Geschick ausweichen (wenn es der Weg zulässt) oder rechtzeitig anhalten und warten bis sie verschwunden ist. Wird der Spieler von ihr getroffen, blinkt die Figur eine gewisse Zeitspanne lang und ist unbeweglich. Auf dem gesamten Pfad gibt es eine bestimmte Anzahl an Auslösern und Savepoints in Form der Bananenstauden. Diese befinden sich entweder auf dem normalen Weg, auf festen Platten oder auf festen Blöcken. Je nach Schwierigkeitsgrad ändert sich die Anzahl. Die Savepoints auf den festen Blöcken speichern den bereits zurückgelegten Pfad, sodass der Spieler (mit beiden Figuren) an diesen Punkt zurück gelangt, sobald er vom Weg abgekommen ist. Wenn man so einen Block erreicht, baut sich der Weg erst weiter auf wenn der Savepoint berührt wurde. Die Auslöser haben die Eigenschaft, den Weg des Partners freizulegen (auf losen und festen Platten). In diesem Fall kommt der Partner nicht voran (da kein Auslöser vorhanden ist) und muss warten bis die andere Figur den Auslöser berührt hat. An den Savepoints erscheint auch der Vogel, der dem Spieler anzeigt, wie weit es noch bis zum Ziel ist (in Form eines Satzes in einer Sprechblase). Er erscheint aber nicht nur dort. Per Zufall taucht er an einigen Stellen im Spiel auf um den Spieler abzulenken. Die Kommentare des Vogels sind vielfältig und werden zufällig ausgewählt, oder eventuell an den Verlauf der Story angepasst (wenn es zum Beispiel dämmert und der Spieler sich beeilen muss). Die vorgegebenen Pfade können auch einander überschneiden. Das bedeutet, dass einer der Pfade höher gelegen ist und die Sicht auf den unteren versperren kann. Es gibt keine räumlichen Begrenzungen für die einzelnen Pfade. Der Score hat die Aufgabe, bestimmte Dinge am Rande des Bildschirms anzuzeigen. Zum Beispiel die verbleibende Zeit, die man für ein Level zur Verfügung hat. Außerdem werden die Anzahl der gesammelten Bananen und direkt daneben die übrigen Leben angezeigt. Kommt man schließlich an das Ende des Levels, ist dort ein Savepoint auf dem Zielblock, der nur noch berührt werden muss, um das Level zu beenden. Man kehrt zurück in die äußere Welt und kann das nächste Level starten, das durch Beenden des vorigen freigeschaltet wurde.

3.3 Finales Konzept

Der Kern meines Spiels ist das besondere Gameplay. Es ist herausfordernd und erfordert Übung, Disziplin und Geschicklichkeit. Die Aufgabe des Multi-Taskings ist nicht einfach, und sollte den Spieler dazu bringen, es immer wieder versuchen zu wollen; so lange, bis er es schafft. Das entspricht gänzlich dem Gedanken hinter der Definition des Gameplays von Ernest Adams, in dem es heißt, dass der Spieler denken soll: Diese Aufgabe klingt schwer, aber ich will beweisen, dass ich es schaffe. Wie schon erwähnt, haben sich ein paar Dinge geändert. Es tauchten einige Probleme auf, die ich lösen musste. Zum Beispiel konnte ich nicht herausfinden, wie ich den Pfad erstellen kann, der sich auf- und abbaut. Egal wie ich es versuchte, es klappte einfach nicht. Außerdem musste ich aus Zeitgründen viele Details, wie Hindernisse, über die man springen kann, streichen. Trotz dieser Schwierigkeiten wollte ich dasselbe Konzept wie zuvor umsetzen. Also überlegte ich mir, wie man die fehlenden Elemente ersetzen kann. Statt den Pfad als Begrenzung zu wählen, wollte ich Gegner einbauen, die quasi die Geschwindigkeit des Spielers kontrollieren und ihn zwingen, sich einerseits zu beeilen und andererseits nicht zu weit voraus zu laufen. Mittlerweile steuert man nicht mehr Affen durch die Level, sondern Katzen. Und sie klettern auch nicht durch einen Dschungel, sondern laufen über die Dächer einer Stadt. Im Nachhinein fand ich die Idee mit den Katzen über den Dächern ansprechender und passender für das Konzept des Gameplays, da das Gameplay eine geeignete Geschichte benötigt, wie wir in 2.2.1 Streben nach Harmonie von Ernest Adams und 2.2.2 Geschichten schreiben von Bob Bates gelernt haben. Ich habe mir dementsprechend neue Gedanken gemacht, um das Spiel interessant und fordernd zu gestalten. Zum Beispiel, dass Gegner in Form von Vögeln die Figuren behindern und so den Spieler irritieren sollen. Durch den Zeitmangel musste ich meine vielen Ideen leider auf ein Minimum reduzieren, deshalb möchte ich kurz erklären wie das finale Konzept aussieht.

3.3.1 Beschreibung der Features

- Zwei Pfade für jede Figur in Form von Katzen, die räumlich begrenzt sind und sich nicht überschneiden.
- Gegner in Form von Vögeln, die auf die Figuren zufliegen und Ratten, die von hinten auf die Figuren zulaufen.
- Fische zum Einsammeln, um den Score und die Anzahl an Leben zu erhöhen.
- Die Katzen haben eine bestimmte Anzahl an Leben und es gibt einen Countdown. Sie werden gleichzeitig gesteuert.

3.3.2 Zusammenhänge der Features

- Die Pfade sind jederzeit komplett sichtbar. Sollten sie verlassen werden, startet das Level neu. Berührt eine Katze das Dach, verliert man ein Leben und das Level startet ebenfalls neu.
- Kollidieren Katzen und Vögel, wird ein Leben abgezogen. Sollte die Anzahl auf Null sinken, startet das Spiel neu. Kollidieren Katzen und Ratten, startet das Spiel sofort neu.
- Fische liegen verteilt in den Leveln auf den Pfaden; die Figur erhält Punkte und Leben für das Einsammeln der Fische.

3.3.3 Liste der zu entwickelnden Grafiken

- Figuren - Katze, Vogel, Ratte
- Objekte - Fisch
- Pfad - Regenrinnen, Dachziegel
- innere Welt - Dächer einer Stadt, dazwischen Wiese
- äußere Welt - Startbildschirm und Tutorial
- Score - Zeit, Fischanzahl, Leben
- Animationen - Bewegungen der Katzen, Vögel, Ratten, Fische

3.3.4 Klarheit über die benötigte Technologie

Game Maker 8.1 Lite, Sprache: Game Maker Language.

3.3.5 Spielgenre

Ein Action Game, welches einen Mix aus Jump'n'Run und Geschicklichkeitsspiel darstellt.

3.3.6 Plot

Um die Geschichte zu erzählen, die ich mir zum Spiel überlegt habe, habe ich mir ein passendes Gedicht ausgedacht, um es auf eine interessante Art und Weise dem Spieler zu präsentieren.

Two cats want to go for a walk
On a shiny summer day.
But the tins of the roofs are too hot,
So they must find another way.
And please do not flutter
Or utter a cry,
If on the gutter
A big rat goes by.
So hurry and do not fall down
Or you will have another try.

Es geht also darum, dass zwei Katzen einen Spaziergang über die Dächer einer Stadt machen wollen. Da es aber ein sonniger Tag ist, ist das Blech auf den Dächern zu heiß um darauf zu laufen. Deshalb nutzen sie die Regenrinnen um voran zu kommen. Auf diesen Regenrinnen halten sich aber große Ratten auf, vor denen sie flüchten müssen. Dabei müssen sie darauf achten, nicht vom Dach zu fallen.

3.3.7 Spielmechanismus

Das Spiel startet mit einem Titelbilschirm, auf dem mittig der Name des Spiels zu sehen ist (Cats on a hot tin roof) und unten rechts ein Hinweis, dass durch Drücken der Enter Taste das Spiel beginnt. Daraufhin folgt ein Tutorial. Der Spieler kann frei auf dem Bildschirm herum laufen, ohne dass es Konsequenzen hat. Dabei kann er das Gedicht freischalten, das die Geschichte erzählt. Unten ist erneut ein Hinweis, dass es mit Enter weitergeht. Der Spieler startet mit beiden Figuren links und rechts unten auf dem Bildschirm. Die Spielzeit wird rückwärts runtergezählt und oben rechts angezeigt. Man hat für alle drei Level insgesamt 42 Sekundentakte Zeit. Auf derselben Seite ist auch die Fischanzahl zu sehen, die natürlich bei 0 beginnt. Oben links befindet sich die Lebensanzahl in Form von Herzen, zu Beginn sind es 3; diese Zahl kann nicht überschritten werden. Sollte eine der Figuren vom Pfad, also der Regenrinne, abweichen, wird das Level neu gestartet. Berührt eine von ihnen das Dach, verliert man ein Leben und das Level wird ebenfalls neu gestartet. Die vorherige Anzahl der Herzen, der Fische und die Restzeit bleiben nach dem Neustart des Levels erhalten. Kollidieren die Katzen mit vorbeifliegenden Vögeln, verliert man ein Leben und der Vogel verschwindet. Sammelt man einen Fisch ein, erhält man ein

neues Leben und einen Punkt im Score; der Fisch verschwindet nach dem Einsammeln. Ein Game Over tritt ein, wenn man alle drei Leben verloren hat, eine der Ratten eine Katze berührt oder die Zeit abgelaufen ist. Dann muss man von vorne beginnen. Das Spiel wird erfolgreich abgeschlossen, wenn man alle Level durchläuft. Es wird außerdem ein Highscore am Ende angezeigt. Das Ergebnis setzt sich zusammen aus der Addition von der Anzahl der übrigen Herzen, der Anzahl der Fische und der Restzeit. Die Ergebnisse werden gespeichert und in einer Top Ten List präsentiert. Je mehr Herzen und Fische man also hat und je schneller man die Level gemeistert hat, umso höher ist die Punktzahl.

3.4 Implementierung

Zunächst einmal vorweg: Während meiner Einarbeitungsphase habe ich viel Zeit damit verbracht, mich für eine Programmiersprache zu entscheiden und mich in diese einzuarbeiten. Meine erste Entscheidung fiel auf die Skriptsprache Lua, die lediglich einen Editor wie Notepad++ benötigt. Nach vielen Tutorien und kleinen Erfolgserlebnissen merkte ich leider, dass die Zeit nicht reicht um mir genügend Wissen für mein Spielkonzept anzueignen. In Lua muss alles von Grund auf selbst implementiert werden. Deshalb wagte ich den Schritt, mich für einen anderen Weg zu entscheiden. Ich wollte mit der Engine Game Maker arbeiten und lernte schnell die wichtigsten Funktionen kennen. Nach kürzester Zeit wusste ich, wie eine Figur animiert wird, wie Kollisionen funktionieren, und auch wie man gegnerische Figuren erstellt, die sich quer durch das Fenster bewegen, oder wie die Kamera gesteuert werden kann. Durch den Zeitverlust war es mir nicht möglich, mich mehr mit der Gestaltung meines Spiels zu beschäftigen, sodass nur die wichtigsten Elemente vorhanden sind. Natürlich hätte ich gerne noch viel mehr in das Spiel eingebaut, um den Spielspaß zu erhöhen. Das Kapitel Implementierung gewährt einen kleinen Einblick in die Funtkionalität von LÖVE und Game Maker.

3.4.1 Lua/ LÖVE

Wie schon im Titel erwähnt, habe ich die 2D Game Engine LÖVE benutzt um in der Skriptsprache Lua zu schreiben. Es ist keine Engine in dem Sinne, dass man eine Benutzeroberfläche hat oder Ähnliches. Man kann aber sein Programm mit Hilfe von LÖVE ausführen, indem man die Textdatei oder den Ordner mit seinem Code auf das LÖVE Icon zieht. So wird das Spiel gestartet, es werden sogar Fehlermeldungen ausgegeben. In der Regel benötigt man ein main.lua Datei, die das Hauptprogramm enthält, Libraries, Bilder, und eventuell weitere lua-Dateien für eine bessere Organisation des Codes. LÖVE hat also minimalste Eigenschaften einer Entwicklungsumgebung, meiner Meinung nach trotzdem ausreichend. Hilfe bekommt

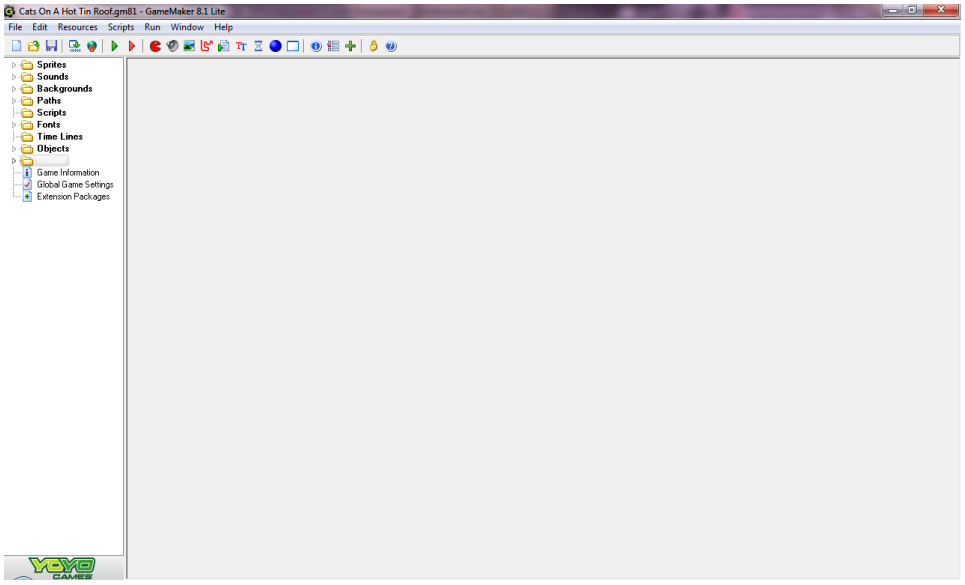

Abbildung 1: Benutzeroberfläche von Game Maker 8.1 Lite

man aus zahlreichen Wikis und Tutorien, die Online zur Verfügung stehen. Ich habe mir viele Tutorials angesehen und gelernt, wie man eine Map erstellt und die Kamera über der Map steuern kann. Außerdem habe ich versucht, eine Art Pfad aus Vierecken zu erstellen, der sich schrittweise aufbaut. Das ist mir nur teilweise gelungen. Der Pfad wurde angezeigt, aber nicht schrittweise. Ich konnte auch eine Figur steuern und rotieren lassen und dafür sorgen, dass die Figur den Bildschirm nicht verlässt. Ich habe mit Hilfe eines Tutorials sogar einen kleinen 2D Action Shooter programmiert, in dem man Gegner mit Schüssen besiegen kann. Allerdings musste ich dann feststellen, dass ich mit meiner Spielidee nicht voran kam und auch noch nicht wusste, wie ich sie umsetzen kann. Der Aufwand war einfach viel zu groß für so eine kurze Zeit. Deshalb beendete ich die Arbeit mit Lua/ LÖVE und wechselte zu Game Maker.

3.4.2 GML/ Game Maker

Dieses Kapitel dient dazu, die Funktionalität und den Aufbau von Game Maker in den Grundlagen zu verstehen und zu erfahren, wie ein Spiel aufgebaut ist. Ich werde hier nicht auf alle Details eingehen und auch nicht jede Funktionalität erklären. Ich behandle nur die grundlegenden Funktionen und Herangehensweisen, die man im Umgang mit Game Maker zu Beginn lernt. Ich werde auch nicht jede Codezeile meines Spiels erklären, dies würde sonst den Rahmen sprengen.

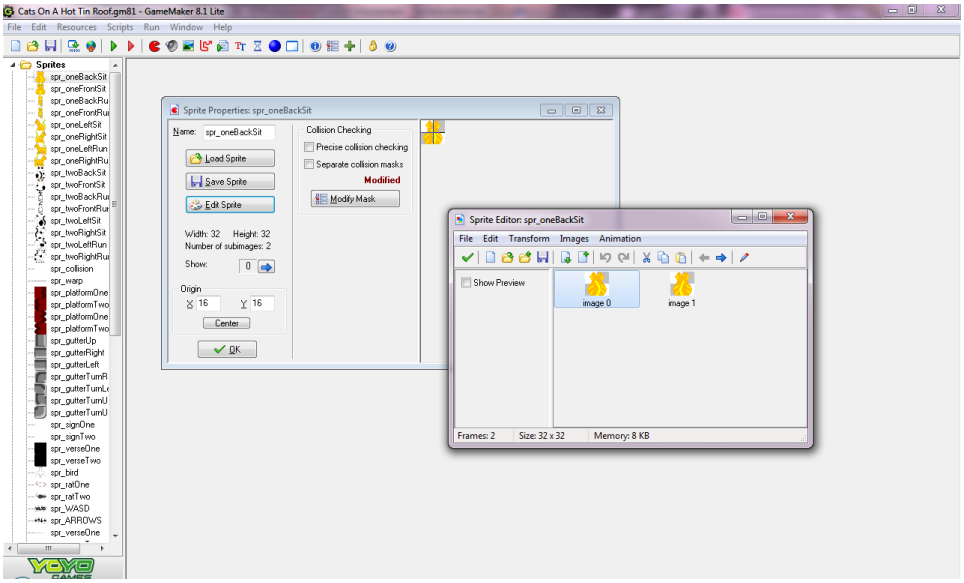

Abbildung 2: Sprites

Zunächst einmal erstellt man ein neues Spiel oben links unter File->New oder mit dem Icon Create a new game. Normalerweise beginnt man damit Sprites zu erstellen. Dazu genügt ein Rechtsklick auf den Ordner Sprites um Create Sprite auszuwählen. Ein Fenster auf der Benutzeroberfläche öffnet sich und in der Ordnerhierarchie erscheint ein neues Sprite. Das Fenster mit dem Namen Sprite Properties bietet nun die Möglichkeit ein paar Einstellungen vorzunehmen. Man wählt zuerst einen Namen; am Besten beginnt dieser mit spr_ als Abkürzung für Sprite und im Anschluss folgt die Beschreibung der Figur, wie zum Beispiel oneBackSit (Figur Eins, Rückseite, sitzend), sodass es später nicht zu Verwechslungen kommen kann. Unter dem Namen stehen nun folgende Buttons zur Verfügung: Load Sprite, Save Sprite, Edit Sprite. Für gewöhnlich erstellt man hier einen Sprite mit Edit Sprite, wenn man nicht ein anderes Programm dafür nutzen möchte um es dann mit Load Sprite einzufügen. Mit Edit Sprite öffnet man also den Sprite Editor. Mit Create a new sprite öffnet sich ein kleines Fenster, in dem man die Breite und Höhe des Sprites festlegen kann. Im Sprite Editor befindet sich nun ein Bild mit dem Namen image 0. Mit einem Doppelklick auf das Bild oder dem Icon Edit the image oben rechts öffnet sich das Fenster Image Editor. Das Bild ist sehr klein, deshalb kann man mit Zoom in hineinzoomen. Der Image Editor ist ein kleines Malprogramm, das die typischen Funktionen eines solchen aufweist. Man kann mit einem Stift zeichnen und geometrische Formen oder auch Text einfügen. Es gibt außerdem die Möglichkeit, die Opazität, also die Lichtundurchlässigkeit einer Farbe einzustellen. Wenn das Bild fertig ist, klickt man oben links auf das grüne Häkchen um zu speichern und den Image Editor zu schließen.

Zurück im Sprite Editor kann man entweder weitere Bilder mit dem Icon Add an empty image at the end anlegen und somit Animationen erstellen, oder man beendet ihn auch hier mit dem grünen Häkchen oben links. Eine Animation ist also nur eine indizierte Abfolge der Bilder, die hier angelegt werden. Sprite Properties besitzt noch weitere Funktionen. Es zeigt Breite und Höhe, die Anzahl der Subimages, die Bilder selbst (Show) und den Koordinatenurspung (Origin) des Bildes/ der Bilder an. Letzterer ist standardmäßig auf (0,0) eingestellt und kann auch manuell für das jeweilige Sprite verändert werden (mit dem Cursor auf dem Bild selbst oder per Eingabe), oder das Zentrum des Bildes wird per Button (Center) als Ursprung festgelegt. Das Koordinatensystem in Game Maker ist ein Linkssystem, wobei der Ursprung (0,0) immer oben links ist. Bei der typischen Bildgröße von 32 x 32 wäre das Zentrum (16,16). Üblicherweise wählt man das Zentrum als Ursprung, da dies das Programmmieren mit den Objekten vereinfacht. Um die Art von Kollision mit dem Sprite bestimmen zu können gibt es die Funktion Collision Checking, bei der man die Wahl zwischen Precise collision checking (Standardeinstellung) und Separate collision masks hat, wofür man mit Modify Mask die Maske verändern kann. Öffnet man Modify Mask, kann in Mask Properties Bounding Box, Shape und Alpha Tolerance nach Belieben eingestellt werden. Somit wären alle möglichen Eigenschaften eines Sprites abgedeckt.

Als nächstes erstellt man Objects, denen man dann die zuvor bearbeiteten Sprites zuweist. Ein Rechtsklick auf Objects und die Auswahl von Create Object öffnen das Fenster Object Properties und erstellen ein neues Object in der Ordnerhierarchie. Wie auch in Sprites gibt man dem Object einen sinnvollen Namen, der mit obj_ beginnen sollte; beispielsweise obj_playerOne. Dementsprechend erhält dieses Object auch das Sprite spr_oneBackSit (da man die Figur beim Starten des Spiels von hinten sehen soll), welches man direkt unter der Namenseingabe auswählen kann. Manchmal haben Objekte aber auch keinen Sprite, wenn man zum Beispiel nur einen Text im Raum anzeigen lassen will. Klickt man auf den Button New im Abschnitt Sprite kann ein neues Sprite erstellt werden und es öffnet sich das Fenster Sprite Properties. Man kann dem Object die Eigenschaft Visible und Solid zuweisen, und mit Depth die z-Koordinate im Raum bestimmen. Soll sich ein Object zum Beispiel tiefer im Raum befinden als andere, setzt man für Depth einen positiven Wert ein. Soll es weiter vorne positioniert sein, ist der Wert von Depth ein negativer. Wenn man dem Object die Eigenschaft Persistent gibt, so werden Veränderungen dauerhaft gespeichert; also auch wenn man in einen neuen Raum wechselt, bleiben die Veränderungen erhalten. Dies ist zum Beispiel sinnvoll, wenn Objekte in einem Raum verändert werden und dieser Zustand erhalten werden soll. Ein Object kann einen Parent besitzen, der auch ein Object ist und diesem seine Eigenschaften vererbt. Es kann auch die Maske (Mask)

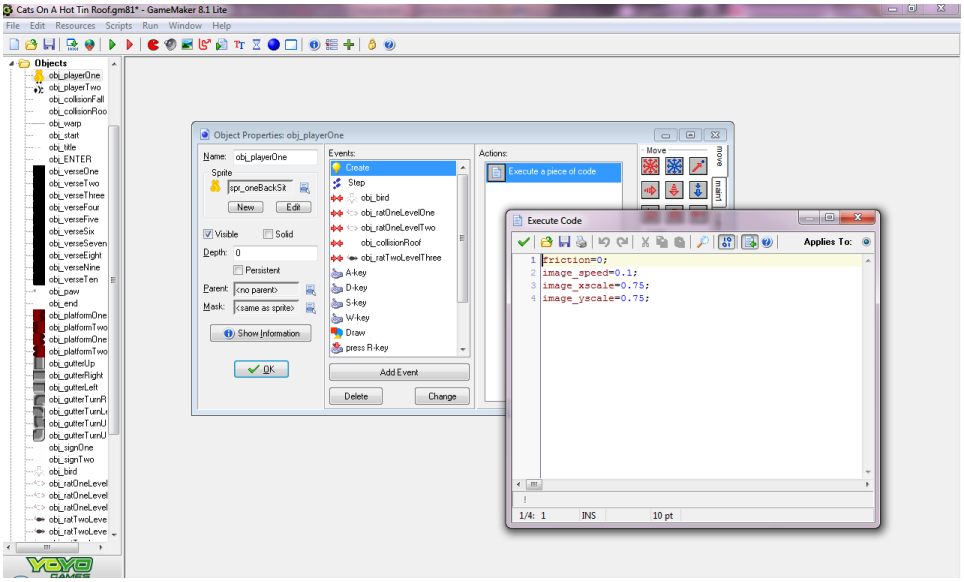

Abbildung 3: Objekte

verändert werden, wenn es nicht dieselbe wie das zugewiesene Sprite haben soll. Mit dem Button Show information öffnet sich ein Fenster mit allen Daten zu einem Object, die man auch als Datei außerhalb von Game Maker abspeichern kann. Dies ist übrigens mit Sprites und selbst geschriebenem Code ebenfalls möglich.

Wie schon im Kapitel 2 Grundlagen im Abschnitt 2.1 Game Maker erklärt worden ist, werden Objekten Events und dazugehörige Actions zugewiesen. Ich möchte hier beispielhaft erklären, wie ich die Bewegungen und Animationen meiner Spielfigur programmiert habe.

Das erste Event heißt Create. Hier stehen alle Eigenschaften, die das Objekt besitzen soll wenn es im Raum erstellt wird. In diesem Fall folgendes Code-Fragment:

```
friction=0;
image_speed=0.1;
```

Friction reduziert die Geschwindigkeit eines Objektes (einer Instanz). In jedem Schritt (Step) wird die Zahl, die man hier angibt, von der Geschwindigkeit subtrahiert bis diese gleich Null ist. In diesem Fall wird friction initialisiert mit 0.

image_speed legt die Geschwindigkeit der Animationen fest, also wie schnell von einem zum anderen image gewechselt beziehungsweise wie lange das jeweilige image angezeigt werden soll. Ich habe sie sehr klein gewählt, damit die Animationen gut zu sehen sind und eine angenehme Geschwindigkeit für den Betrachter haben.

Das nächste wichtige Event ist das Step Event. Das Step Event wird bei jedem Schritt im Spiel ausgeführt. Hier werden Actions gesetzt, die kontinuierlich geschehen sollen.

```
    x=min(x,room_width-16);
x=max(x,0+16);
y=min(y,room_height-16);
y=max(y,0+16);
```

Die min-/ max-Funktion, die hier aufgerufen wird, ist sehr simpel. Der x- und y-Koordinate der Instanz wird immer der Wert zugewiesen, der momentan der Größte (max) oder der Kleinste (min) ist. Hier dient dies dazu, zu verhindern, dass eine Instanz den Raum verlassen kann. Sollte also x einen Wert erreichen der größer als room_width-16 ist, ist letzteres das Minimum und wird x zugewiesen. Im Spiel wäre dies am rechten Rand des Raumes. Und sollte x kleiner als 16 werden, ist letzteres das Maximum und wird x zugewiesen. Das wäre der linke Rand des Raumes. Wäre hier der Koordinatenursprung der Instanz bei (0,0), würde man room_width als Minimum und 0 als Maximum wählen. Die Zuweisungen für y sind von der Funktionsweise her identisch, allerdings benutzt man hier logischerweise room_height-16. Die Instanz kann also auch nicht über den oberen und unteren Rand des Raumes hinaus bewegt werden.

Nun möchte ich die Keyboard Events erklären. Die Figur soll mit der Tastenbelegung W-A-S-D (Keys) bewegt werden, dementsprechend gibt es Keyboard Events für jeden Key. Der W-Key ist folgendermaßen programmiert:

```
    y-=1;
sprite_index=spr_oneBackRun;
```

Diese zwei Zeilen haben eine einfache Bedeutung. Wenn sich die Figur nach oben bewegt (entlang der y-Achse), soll man sie von hinten laufend sehen können (spr_oneBackRun). y-=1 dekrementiert den y-Wert der Instanz um 1, dies ist eine Bewegung nach oben. Mit sprite_index wird der Instanz ein neues Sprite zugewiesen. Wenn man den Namen eines Sprites angibt, wird automatisch das image mit dem Index 0 angezeigt. Wenn dann mehrere Bilder existieren, wird eine Animation ausgeführt. Man könnte

hier auch nur eine Zahl für sprite_index angeben. Dann würde das image des Sprites mit dem jeweiligen Index angezeigt werden, das der Instanz zugewiesen wurde und gegebenenfalls die darauf folgenden Bilder. Bei Drücken des Keys würde dann immer zu dem image mit dem angegebenen Index gesprungen werden und erst danach folgen die anderen Bilder. In diesem Fall wechselt man also beim Event W-Key zum image0 von spr_oneBackRun und die Figur läuft nach oben. Gleichermaßen funktionieren die Keys A, S und D. Bei A läuft man nach links (x-=1) und spr_oneLeftRun wird gezeichnet. Mit S läuft die Figur nach unten (y+=1) und wird durch spr_oneFrontRun animiert. D schließlich bewegt die Figur nach rechts (x+=1) und führt spr_oneRightRun aus.

Zusätzlich habe ich das Event Key Release für die Keys W, A, S und D eingebaut. Dort wird lediglich folgender Code ausgeführt (in diesem Fall für W):

sprite_index=spr_oneBackSit;

Gleichermaßen ist der sprite_index von A mit spr_oneLeftSit, von S mit spr_oneFrontSit und von D mit spr_oneRightSit belegt. Sobald also der jeweilige Key losgelassen wird (release), wird die vorgegebene Animation ausgeführt. Ich wollte, dass die Figur entsprechend ihrer Laufrichtung sitzt (mit dem Gesicht in diese Richtung), sobald sie nicht mehr läuft.

Als kleinen Zusatz habe ich ein Key Press Event eingefügt, mit dem man mit Hilfe von game_restart() das Spiel neu starten kann, sobald der Key R gedrückt wird (press). Das ist recht nützlich beim Testen des Spiels.

Da ich zwei Figuren brauche, habe ich ein entsprechendes Object obj_playerTwo angelegt, das die gleichen Eigenschaften besitzt wie obj_playerOne, mit dem Unterschied, das andere Sprites und die Pfeiltasten zum Bewegen benutzt werden.

Sobald man also Sprites und Objects hat, kann man seinen ersten Room erstellen. Dazu reicht wieder ein Rechtsklick auf den Ordner Rooms um Create Room auszuwählen. Das Fenster Room Properties öffnet sich und in der Ordnerhierarchie erscheint ein neuer Room. Room Properties hat mehrere Tabs, welche da sind: settings, objects, tiles, backgrounds, views. Zuerst wird unter settings der Name festgelegt. Wie gewohnt beginnt dieser mit einer passenden Abkürzung, hier rm_, und endet mit einer Beschreibung wie levelOne. Außerdem bestimmt man Caption, Width, Height und Speed. Man kann festlegen ob der Room Persistent sein soll. Dies ist sinnvoll wenn man während des Spiels Veränderungen im Room hervorruft. Mit Caption kann man festlegen, was oben im Fenster des Spiels stehen

soll, wie zum Beispiel der Name des Spiels. Speed ist standardmäßig auf 30 eingestellt und legt die Länge eines Steps im Raum fest, also wie lang eine Zeiteinheit ist. Wenn Speed also 30 ist, dauert ein Step 30 Millisekunden. Mit dem Button Creation code kann ein Programmcode für den Raum angelegt werden. Im Tab objects kann ein beliebiges Objekt ausgewählt und im Raum platziert werden. Mit einem Rechtsklick auf das platzierte Objekt können an diesem Einstellungen vorgenommen werden, wie zum Beispiel eine neue Position, das Löschen des Objektes oder ein Creation code, um etwas in GML zu programmieren. Mit dem Tab tiles können statt eines Hintergrunds oder Objekten Tiles platziert werden. Tiles sind Platten, die meist zur Gestaltung eines Raums benutzt werden und sonst keine Funktion besitzen. In einem Room kann man im Tab backgrounds entweder eine Hintergrundfarbe wählen und zeichnen lassen, oder einen Hintergrund aus dem Ordner Backgrounds verwenden. Man kann bis zu 8 Backgrounds für einen Raum anlegen, jeder für sich muss auf Visible when room starts gesetzt werden um ihn zu aktivieren. Für jeden einzelnen können Einstellungen ausgewählt werden, wie zum Beispiel ob ein Hintergrund im Vordergrund gezeichnet werden soll. Wenn die horizontale oder vertikale Geschwindigkeit größer als 0 gewählt wird, bewegt sich der Hintergrund während des Spiels mit der entsprechenden Geschwindigkeit horizontal und/ oder vertikal. Zu guter Letzt kann im Tab views die Kamera eingestellt werden. Wie in backgrounds gibt es bis zu 8 Views, die aktiviert und sichtbar gemacht werden müssen. Clear Background with Window Colour sollte auch aktiviert sein. Zusätzlich wird die Position im Raum sowieso Höhe und Breite des Views und des Ports festgelegt. Außerdem kann die Kamera mit einer angegebenen Geschwindigkeit einem ausgewählten Objekt in einem festgelegten Abstand folgen.

Die Tabs befinden sich im Fenster Room Properties links und der Room selbst wird rechts angezeigt. Dort werden dann per Mausklick die Objekte platziert, die man haben möchte; wie beispielsweise die Objekte für die Regenrinnen, die Ratten oder die Katzen. Nun hat man schließlich auch einen Raum in dem die Spielfiguren sich bewegen können. Sprites, Objects und Rooms sind die Grundlagen für ein Spiel. Alle anderen Funktionen, wie Backgrounds, Sounds, Paths kommen erst im Verlauf der Spieleentwicklung hinzu, sind aber nicht nötig um ein kleines Spiel zu starten.

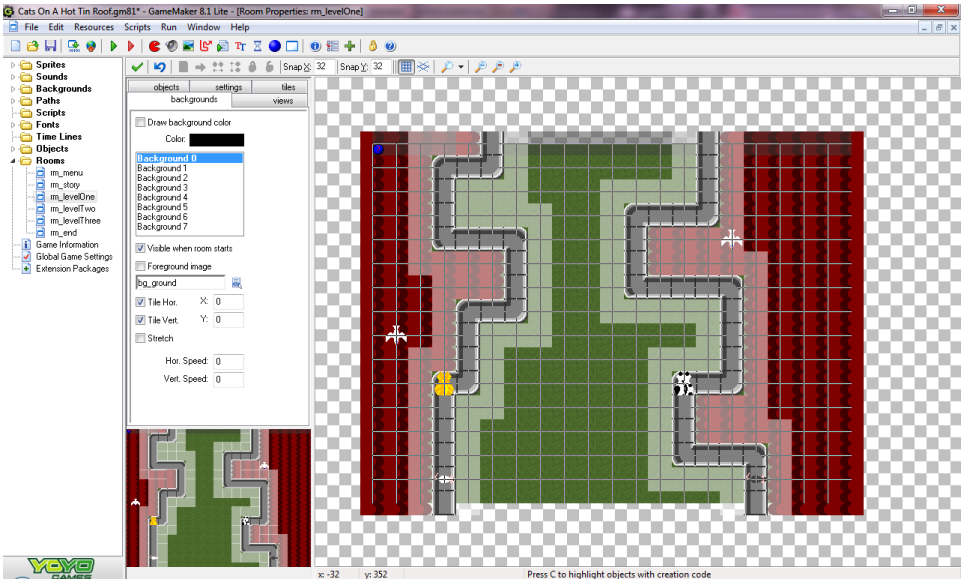

Abbildung 4: Räume, Level Eins in Game Maker

4 Ergebnisse

In diesem Kapitel möchte ich erklären, wie ich das Spielkonzept umgesetzt habe und welche Probleme dabei entstanden sind. Außerdem habe ich mein Spiel testen lassen. Die Ergebnisse daraus möchte ich auswerten und das fertige Spiel den wichtigsten bekannten Game Design Regeln gegenüberstellen.

4.1 Das endgültige Spiel

4.1.1 Titelbildschirm

Beginnen wir mit dem Titelbildschirm. Er ist ein einfacher Raum mit einem schwarzen Hintergrund. Dort platziert sind zwei Objekte, die nur die Funktion haben einen Sprite anzuzeigen. In diesem Fall den Namen des Spiels und die Aufforderung, Enter zu drücken um zu beginnen. Damit das funktioniert ist im Raum auch ein Objekt (ohne Sprite), das bei dem Event release Enter zum nächsten Raum wechselt. Außerdem spielt es durch ein Create Event die Titelmelodie ab, die während des ganzen Spiels zu hören ist.

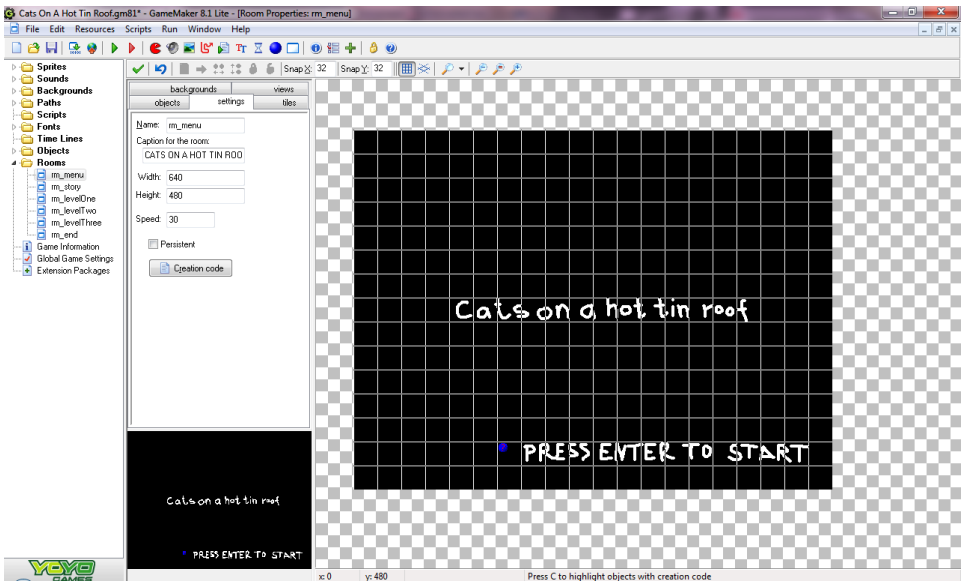

Abbildung 5: Titelbildschirm in Game Maker

Abbildung 6: Titelbildschirm aus der In-Game Sicht

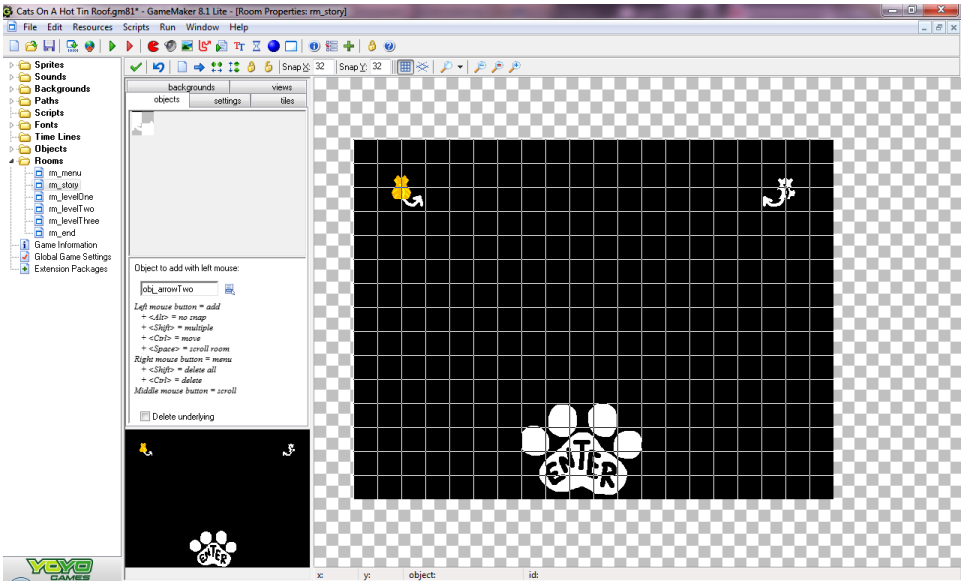

Abbildung 7: Tutorial in Game Maker

4.1.2 Tutorial und Geschichte

Sobald also Enter losgelassen wird, wechselt man zum nächsten Raum. Dieser Raum hat ebenfalls einen schwarzen Hintergrund. Dort sind zwei Objekte für die Katzen links und rechts angelegt, damit der Spieler die Steuerung kennenlernt. Dort, wo die Katzen sind, ist ein Objekt, das einen Sprite anzeigt. Dieser Sprite ist eine Sprechblase, in der die Tastenbelegung für die jeweilige Katze angezeigt wird. Sobald die Katzen den Ort verlassen, verschwinden die Sprechblasen. Direkt unter den Katzen erstellen zwei Objekte Pfeile, die in die Mitte zeigen. Ich wollte, dass der Spieler erkennt, in welche Richtung er zuerst laufen soll. Sobald er also in die angezeigte Richtung läuft, kollidieren die Katzen mit Objekten, die in Form von Sprites die Zeilen des Gedichts darstellen. Die Kollision löst diese Aktion aus. Diese Objekte sind von oben nach unten über den Bildschirm verteilt und geben Zeile für Zeile das Gedicht frei, wenn der Spieler umher läuft. Ganz unten ist wieder ein Hinweis mit dem Wort Enter hinterlegt, der so ausgelöst wird wie im ersten Raum.

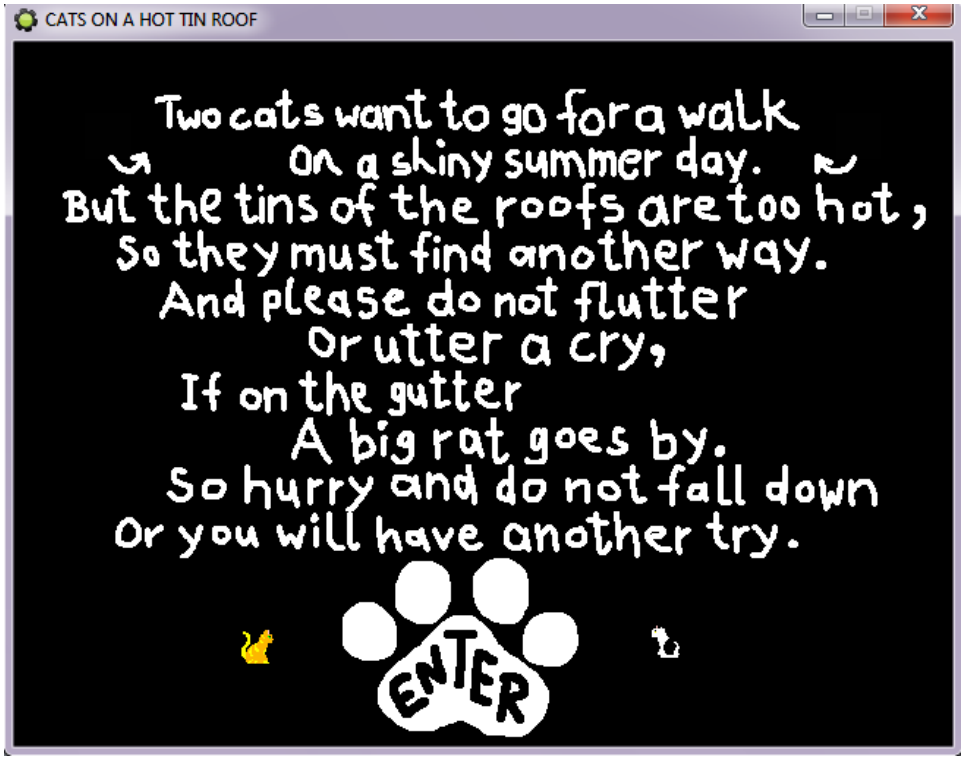

Abbildung 8: Tutorial aus der In-Game Sicht (nach dem Freischalten aller Zeilen des Gedichts, vorher nur schwarz und das Enter-Symbol sowie die Pfeile)

4.1.3 Erstes Level

Dieses Level präsentiert zum ersten Mal die Welt, in der der Spieler sich befindet. Der Hintergrund sieht aus wie eine grüne Wiese. Objekte, die Katzen, Regenrinnen, Dachziegel, Vögel, Ratten darstellen, sind dort zu sehen. Auch hier erscheinen wieder die Sprechblasen an dem Platz, auf dem die Katzen sind. Als ich das Level erstellte, platzierte ich zuerst Dachziegel und Regenrinnen, um danach einen Path für die Ratten zu erstellen. Diesen Path laufen die Ratten immer wieder entlang, so lange man sich in dem Level aufhält. Die Vögel haben einen eigenen festgelegten Winkel im Creation code, um den sie rotiert werden, damit sie in die richtige Richtung schauen. Das hätte man allerdings auch mit dem Befehl image_angle=direction lösen können, wie ich später herausfand, als ich nach einer Lösung für die Ratten suchte. Außerdem steht im Creation code der Vögel ihre Geschwindigkeit, damit ich jederzeit individuell entscheiden kann, wie schnell der Vogel sich über den Bildschirm bewegen soll. Damit der Spieler auf seinem Pfad bleibt, habe ich unsichtbare Kollisionsobjekte erstellt. Die Kollisonsobjekte auf den Dächern starten das Level neu und verringern die Lebensanzahl um 1. Die anderen verursachen nur einen Neustart. Für das erste Level habe ich ein Objekt erstellt, das keinen Sprite hat, aber die Leben, die Fischanzahl und die Zeit erstellt und anzeigt. Hier muss festgelegt werden, welche Startwerte alle drei besitzen sollen, welche Farbe und welchen Font die Schrift haben soll und welche Sprites in der Lebensanzeige benutzt werden sollen. Die Zeit wird in einem gewissen, festgelegten Takt runtergezählt und löst bei 0 einen Neustart des Spiels aus. Um dem Spieler ein gewisses Feedback zu geben, habe ich Klänge und Nachrichten eingebaut. Bei einem Neustart des Levels wird die Meldung “TRY AGAIN KITTY CAT” angezeigt, die mit einem OK-Button weggeklickt wird. Dabei ertönt auch ein Geräusch. Bei einem Neustart des Spiels erscheint die Meldung “GAME OVER”, bei Kollision mit den Objekten auf dem Dach “YOUR PAWS GET BURNT”, bei Kollision mit den anderen ist die Nachricht “YOU FALL DOWN” zu lesen. Bei jeder Meldung habe ich ein Geräusch eingebaut, das ertönt, sobald das entsprechende Event wie No more lives ausgelöst wird. Mit den verschiedenen Nachrichten wollte ich dem Spieler zu verstehen geben, was gerade passiert ist und durch den Klang etwas negatives für den Spieler suggerieren. So fällt ihm eher auf, dass gerade etwas passiert ist, was nicht passieren sollte.

Abbildung 9: Level Eins aus der In-Game Sicht

Um das Spiel zu beginnen, habe ich entschieden, im ersten Level nur zwei Vögel und keine Fische einzubauen. So soll der Spieler lernen, dass er den Vögeln ausweichen muss. Der Abstand zu den Ratten ist zu Beginn des Levels im Vergleich zu den anderen am Größten, sodass der Spieler nicht sofort erwischt wird. All die Funktionen, die dieses Objekt inne hat, gelten bis zum Ende des Spiels. Allerdings werden die verschiedenen Werte nur ein einziges Mal bei Erstellung des Objekts im ersten Level festgelegt. Um von Level zu Level zu gelangen, habe ich außerdem unsichtbare Objekte erstellt, die bei Kollision mit den Katzen einen Raumwechsel ausführen.

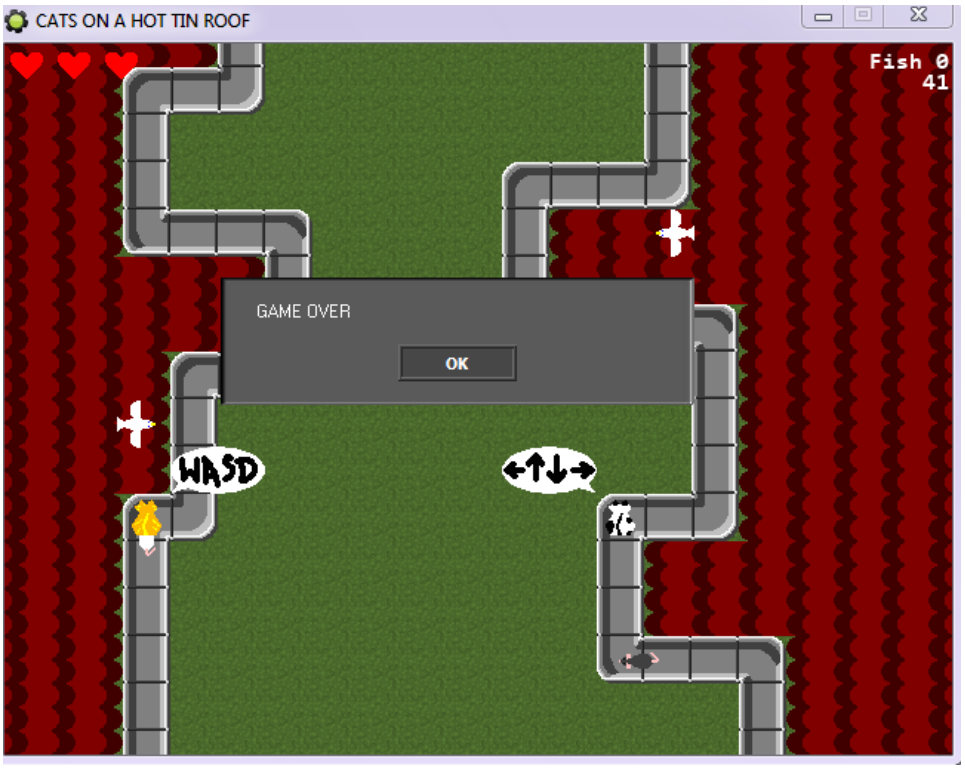

Abbildung 10: Game Over aus der In-Game Sicht

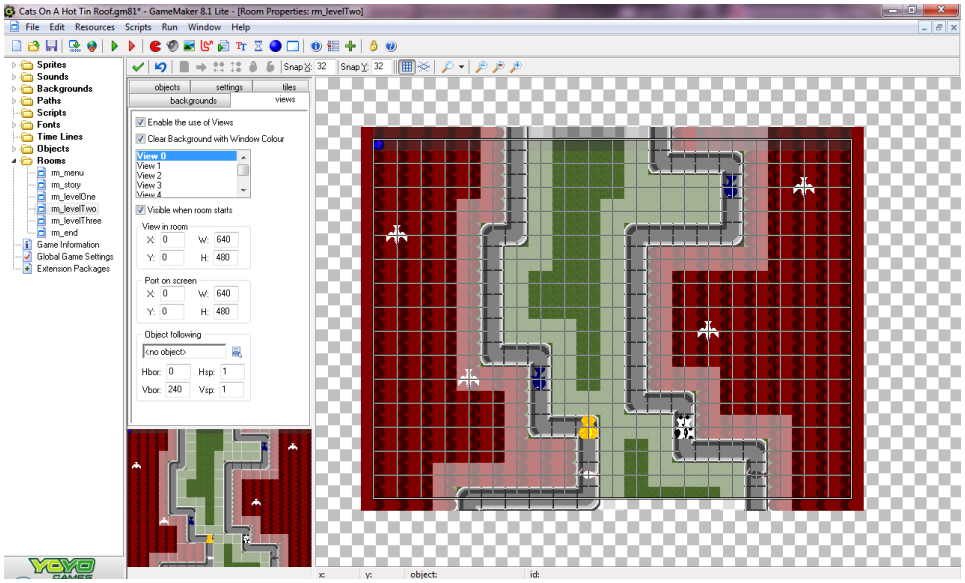

Abbildung 11: Level Zwei in Game Maker

4.1.4 Zweites Level

Neu im zweiten Level ist, dass es nun vier Vögel gibt und zwei Fische auf dem Weg der Katzen, um den Schwierigkeitsgrad zu erhöhen und dem Spieler zu zeigen, dass er durch das Einsammeln der Fische Leben und Punkte erlangt. Außerdem gibt es hier ein Objekt, das nur dafür zuständig ist, die Lebensanzahl, die Fischanzahl und die Zeit zu zeichnen. Es verändert also nicht die im Level zuvor gesetzten Werte. Der Abstand der Ratten zu den Katzen ist geringer, allerdings ist auch die Geschwindigkeit der Ratten hier etwas geringer als zuvor.

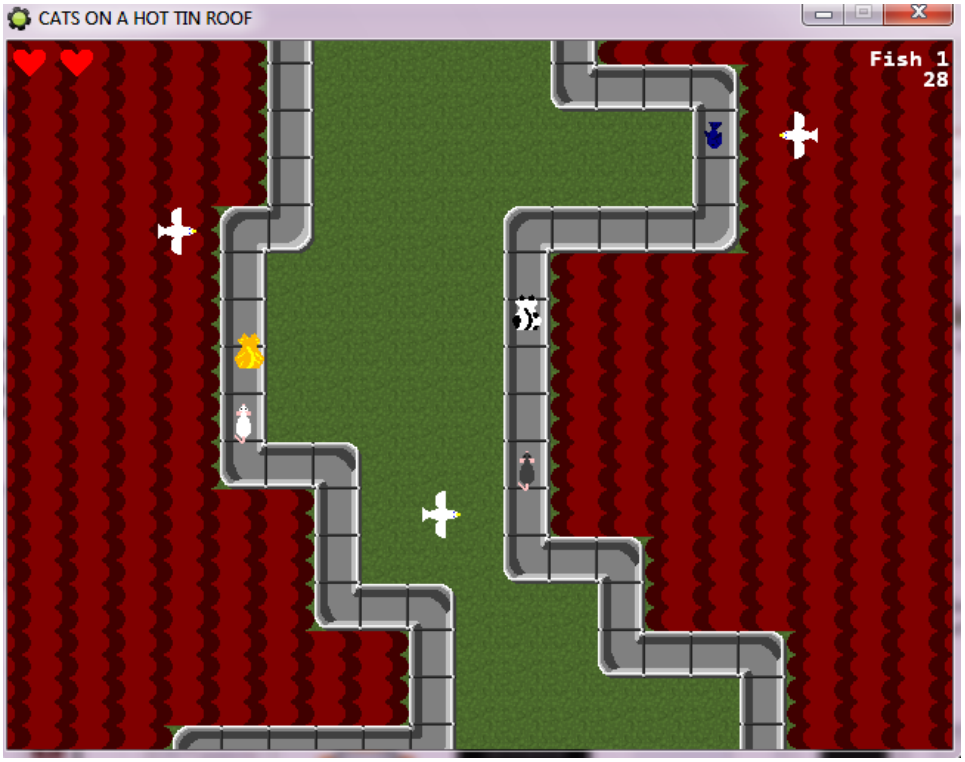

Abbildung 12: Level Zwei aus der In-Game Sicht

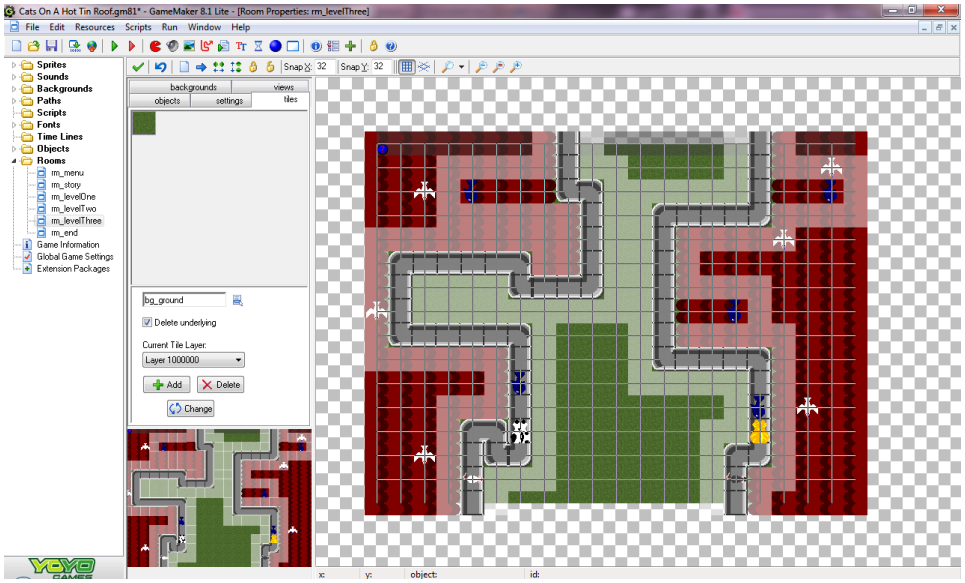

Abbildung 13: Level Drei in Game Maker

4.1.5 Drittes und letztes Level

Da ich mein Spiel testen lassen wollte, habe ich es kurz gehalten. Deshalb ist das dritte Level gleichzeitig das letzte. Der Schwierigkeitsgrad ist hier der höchste. Ich habe die beiden Katzen vertauscht platziert, sodass sich der Spieler der Herausforderung stellen muss, mit der linken Hand die rechte Figur zu steuern und mit der rechten Hand die linke Figur. Die Geschwindigkeit der Ratten ist wieder etwas geringer als zuvor. Es gibt sechs Vögel und fünf Fische in diesem Raum. Davon sind drei auf den Dächern platziert. Ich wollte, dass der Spieler sich fragt, wie er an die Fische auf den Dächern herankommt und versucht, dies herauszufinden. Natürlich gibt es einen Weg zu den Fischen, aber keinen sichtbaren. Ich habe die Kollisionsobjekte um die Fische herum gebaut, sodass es nur einen Weg zu jedem Fisch gibt. So kann der Spieler sogar den Ratten ausweichen. So wie im zweiten Level gibt es auch hier wieder ein Objekt, das die Anzeigen oben im Fenster zeichnet, aber nicht verändert.

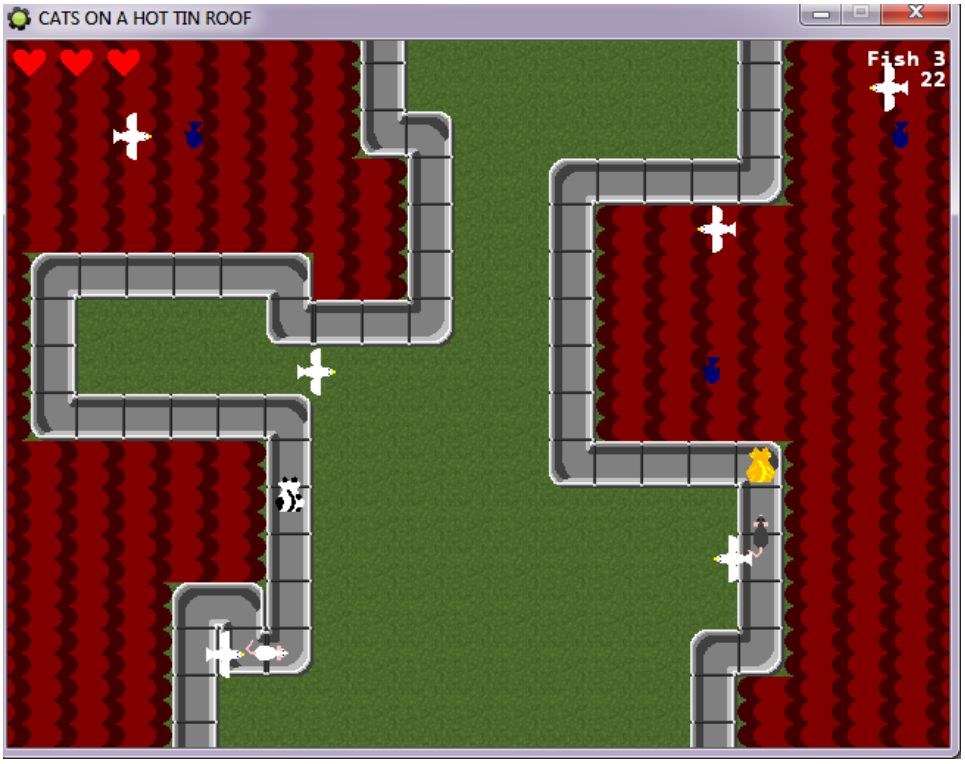

Abbildung 14: Level Drei aus der In-Game Sicht

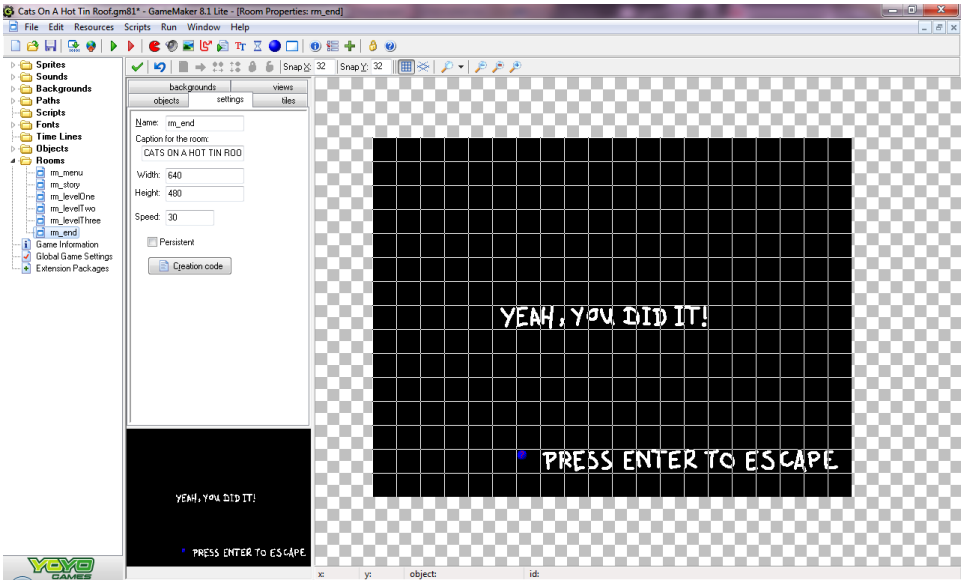

Abbildung 15: Das Ende des Spiels in Game Maker

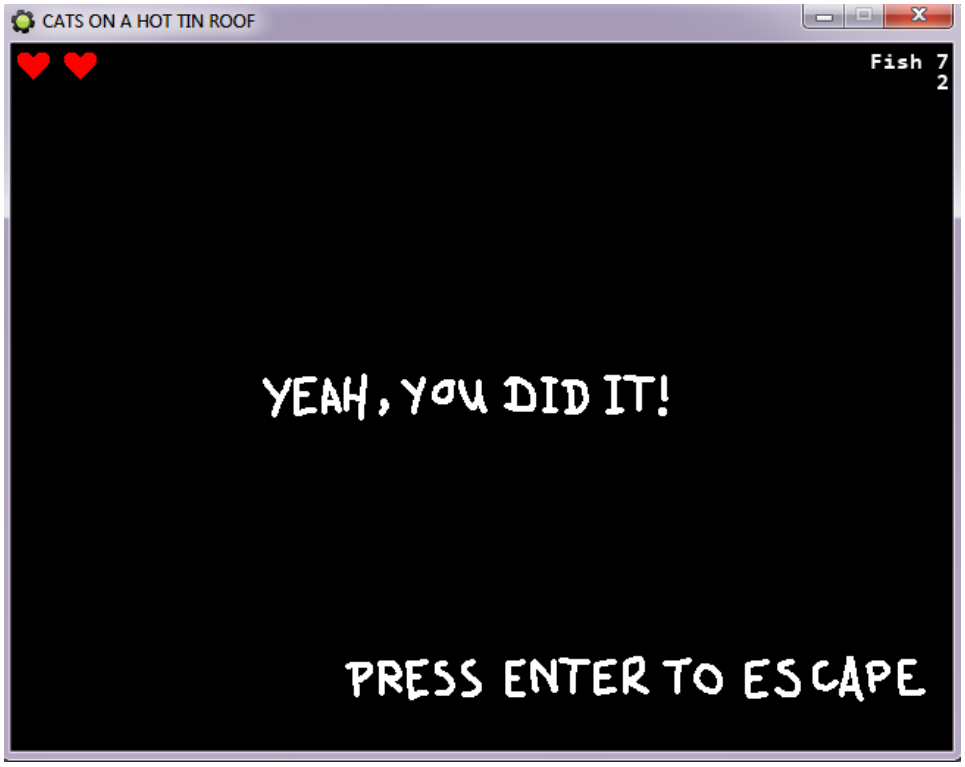

Abbildung 16: Das Ende des Spiels aus der In-Game Sicht

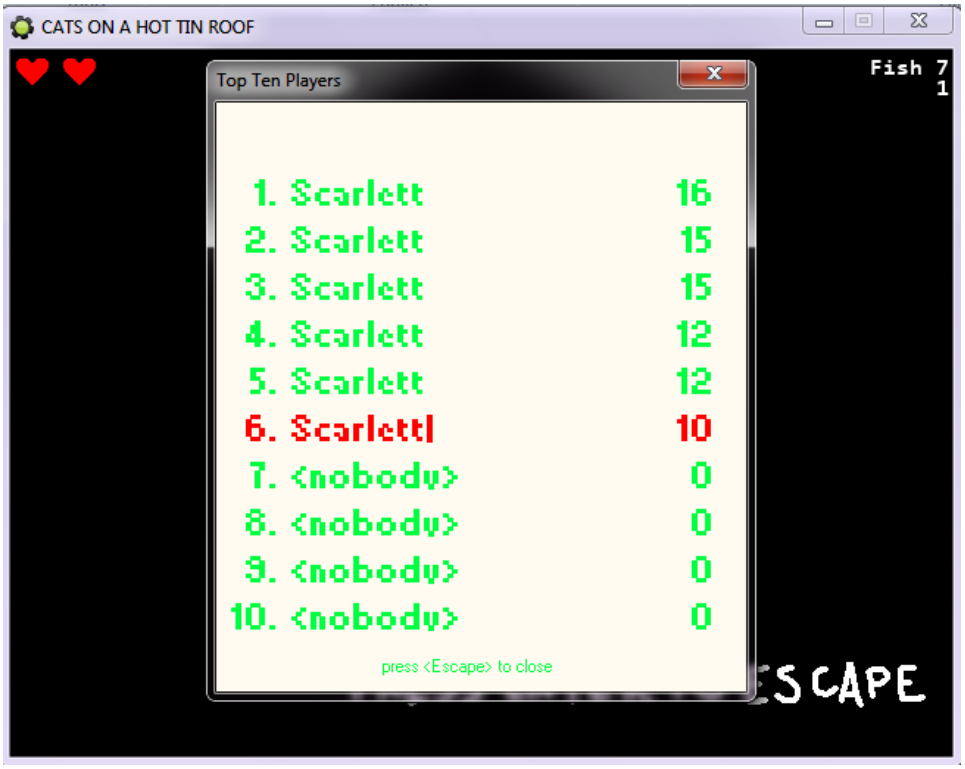

Abbildung 17: Highscore aus der In-Game Sicht

4.1.6 Das Ende und der Highscore

Hat der Spieler das dritte Level verlassen, erscheint der letzte Raum, der wie der Titelbildschirm schwarz ist, und eine Melodie ertönt. Die Lebensanzahl, die Fischanzahl und die Zeit sind hier weiterhin zu sehen. Ich habe keine Lösung gefunden, um sie verschwinden zu lassen oder die Zeit im letzten Raum zu stoppen. Im Raum werden zwei Sprites anzegezeigt. Einer, um dem Spieler zu gratulieren und einer, um anzuzeigen, dass er Enter drücken soll, um das Spiel zu beenden. Das Objekt, das dieses Event behandelt, ist auch für die Melodie beim Eintreten verantwortlich. Drückt der Spieler also Enter, erscheint ein Fenster mit den Top Ten Players und eine weitere Melodie ist zu hören. Dort kann er seinen Namen eintragen und erneut Enter drücken, um das Spiel zu beenden. Der Highscore soll den Spieler für seine Leistung belohnen und ihn anspornen, es noch einmal zu versuchen, um ein besseres Ergebnis zu erzielen.

4.2 Evaluation

Ich habe das fertige Spiel von 14 Personen testen und einen Fragebogen ausfüllen lassen, um herauszufinden, welche Game Design Regeln es erfüllt und welche nicht. Die Noten 1 bis 5 stehen für sehr gut bis mangelhaft, können aber bei einigen Fragen etwas andere Bedeutungen haben; dann steht 1 weiterhin für einen hohen Wert und 5 für einen niedrigen. Anhand der Ergebnisse lässt sich Folgendes sagen: die Regeln, das Ziel, die Steuerung, und die Welt wurden größtenteils verstanden und erkannt. Den Meisten hat das Spiel insgesamt gefallen, obwohl Ästhetik und Harmonie eher im Mittelfeld liegen. Ein besonders gutes Resultat kam bei der Frage 6 auf; die Spieler fanden, dass das Gameplay besonders hervorsticht. Das Belohnungsystem wurde sehr unterschiedlich bewertet; ich vermute das liegt an dem zu hohen Schwierigkeitsgrad. Die meisten Spieler hatten nicht die Möglichkeit, die Items einzusammeln. Das Spiel ist für einen Großteil der Testpersonen eine große Herausforderung und sie fühlen sich in ihrer Handlungsfreiheit stark eingeschränkt. Die Definition eines Action Games scheint dem Spiel nahe zu kommen, zumindest sind fast alle annähernd dieser Meinung. Das Feedback kam unterschiedlich an, was aber wieder durch den zu hohen Schwierigkeitsgrad verursacht worden sein könnte. Fast niemand hat im Tutorial das versteckte Gedicht entdeckt, weshalb nicht alle die Geschichte verstanden haben. Die, die über Level Eins hinaus kamen, haben bemerkt, dass das Spiel schwieriger wird. Alle anderen waren sich nicht sicher, da sie nicht so weit gekommen sind. Das schlechte Rollenverständnis ist ebenfalls auf das versteckte, nicht entdeckte Gedicht zurückzuführen. Dies gilt ebenso für die Frage, ob die Geschichte ins Szenario passt. Die Anzahl der Personen, die das Tutorial bemerkt haben ist nur um 1 größer als derer, die es nicht gesehen haben. Allerdings

haben die, die es ausprobiert haben, das Gedicht trotzdem nicht entdeckt. Der Sinn hinter den Handlungen konnte ebenfalls nicht verstanden werden, wenn das Gedicht nicht gefunden oder die Welt nicht erkannt wurde. Fast jede Testperson fühlte sich überfordert von der Aufgabe des Multi-Taskings und dem Schwierigkeitsgrad. Die Meisten würden es noch einmal spielen wollen, was ein positives Resultat ist. Außerdem waren sich die meisten einigermaßen sicher, was sie zu tun hatten. Der Überraschungsmoment des Gameplays ist unerwarteterweise sehr unterschiedlich bewertet worden. Das könnte aber daran liegen, dass ein paar der Tester eine grobe Idee meines Spiels hatten und sie nicht bis zu Level Drei gelangt sind. Da das Spiel für fast alle Tester zu schwierig war, ist auch die Benotung des Spielspaßes eher mittelmäßig bis schlecht ausgefallen. Weil nur wenige die Items einsammeln konnten, ist auch die Bewertung bezüglich ihres Wertes für den Spieler schlecht ausgefallen. Und da kaum jemand das dritte Level erreichen konnte, ist auch die Benotung für die Neugierde an den schwer erreichbaren Items insgesamt schlecht. Das Lösen der Problemstellungen wurde ebenfalls negativ bewertet, was wieder daran liegt, dass das Spiel für die Tester zu schwierig war.

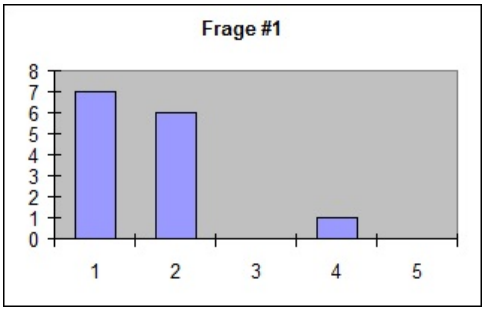

Abbildung 18: Wie verständlich findest du die Spielregeln?

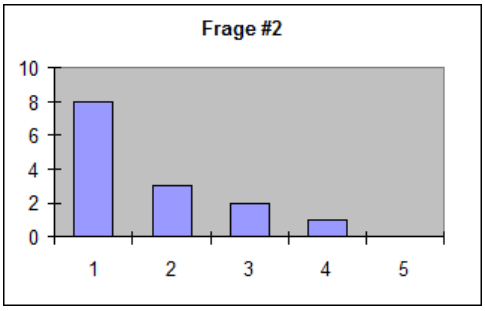

Abbildung 19: Hast du das Ziel des Spiels verstanden?

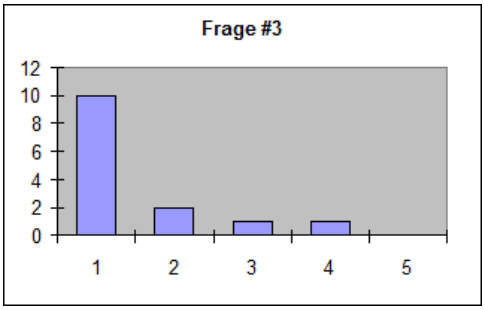

Abbildung 20: Hast du die Steuerung des Spiels verstanden?

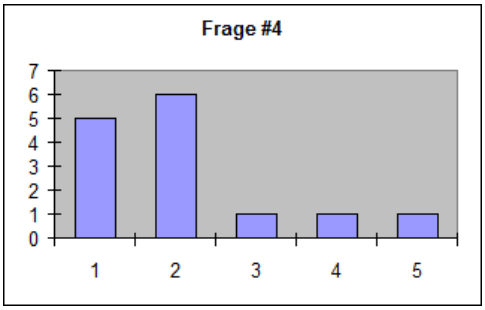

Abbildung 21: Hast du die Welt, in der du dich befindest, erkannt?

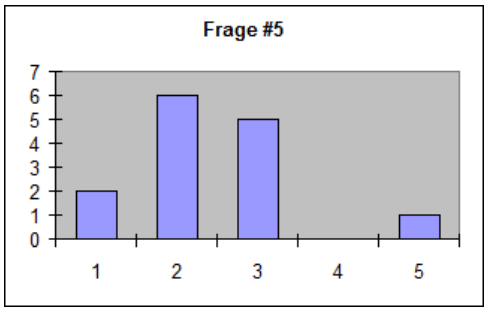

Abbildung 22: Gefällt dir das Spiel?

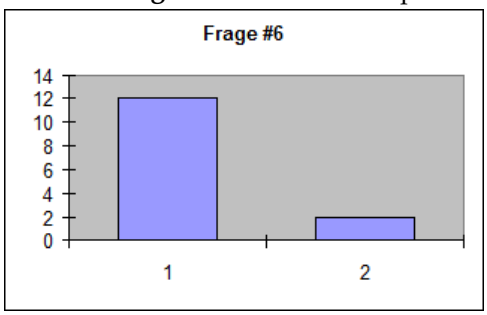

Abbildung 23: Was tritt für dich besonders in Erscheinung: Gameplay, Story oder Grafik? - Die Testperson sollte eines davon angeben, 1 = Gameplay, 2 = keine Antwort.

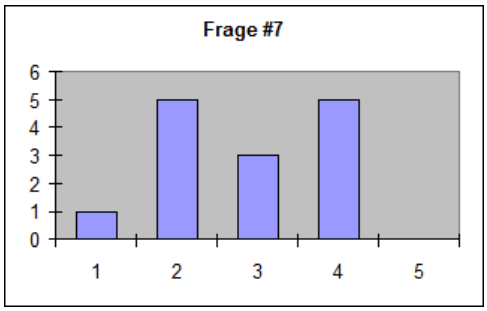

Abbildung 24: Wie ästhetisch ansprechend findest du das Spiel?

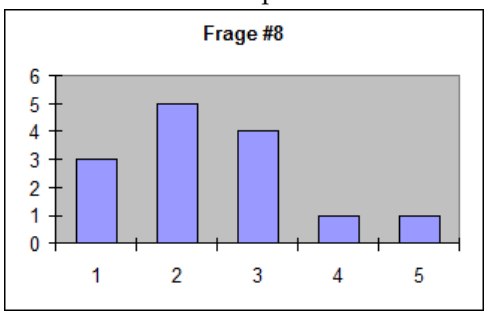

Abbildung 25: Wie harmonisch wirkt das Spiel im Ganzen auf dich?

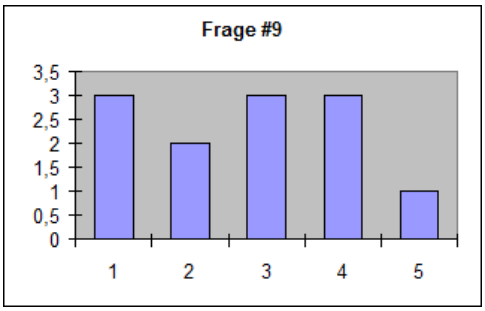

Abbildung 26: Wie zufrieden warst du mit den Belohnungen im Spiel?

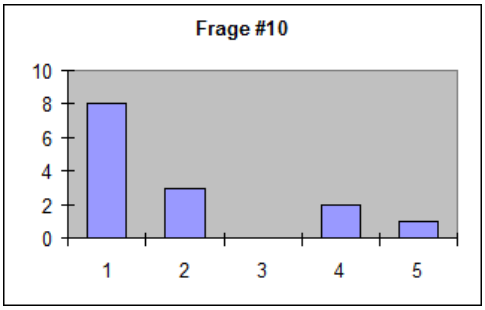

Abbildung 27: Wie herausfordernd ist das Spiel für dich?

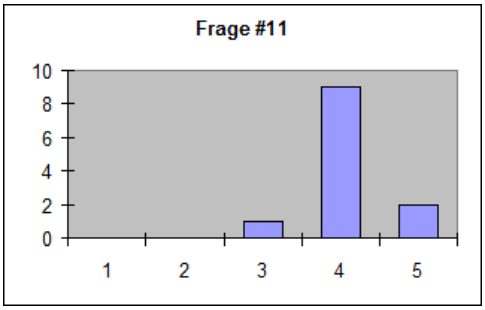

Abbildung 28: Wie hoch empfindest du die Handlungsfreiheit?

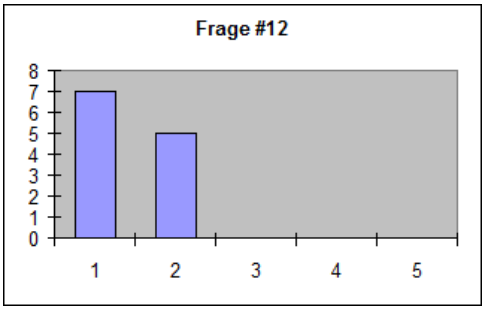

Abbildung 29: Action Games testen die physischen Fähigkeiten und die Koordination des Spielers. Wie sehr erfüllt das Spiel diese Eigenschaften?

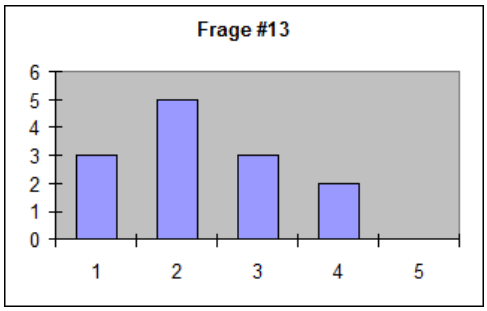

Abbildung 30: Wie zufrieden bist du mit dem Feedback des Spiels auf deine Handlungen? (Rückmeldungen, Reaktionen)

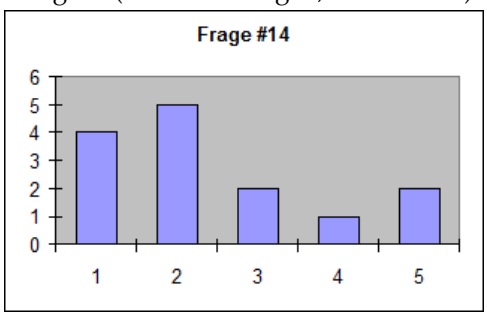

Abbildung 31: Wie verständlich ist die Geschichte des Spiels?

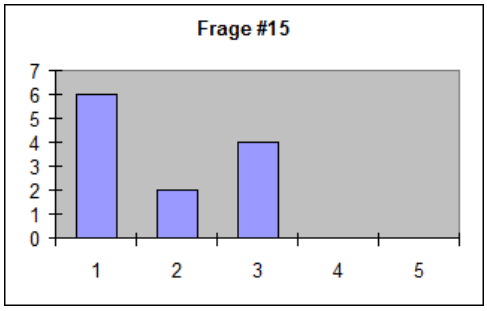

Abbildung 32: Wird das Spiel mit der Zeit schwieriger? (Schwierigkeitsgrad: 1 für stark steigend bis 5 für stark fallend)

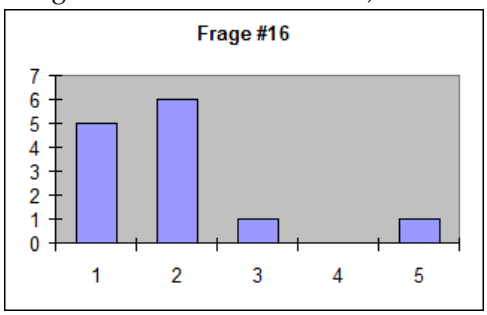

Abbildung 33: Hast du deine Rolle im Spiel verstanden? (Die Note gibt an, wie sehr du sie nachvollziehen kannst!)

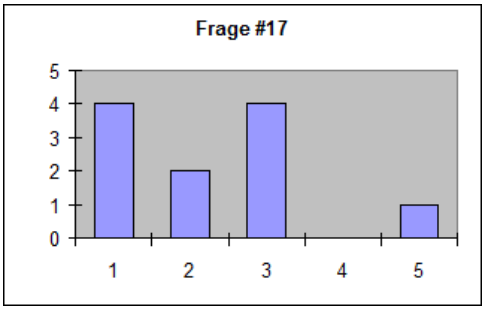

Abbildung 34: Wie sehr passt die Geschichte zum Spiel?

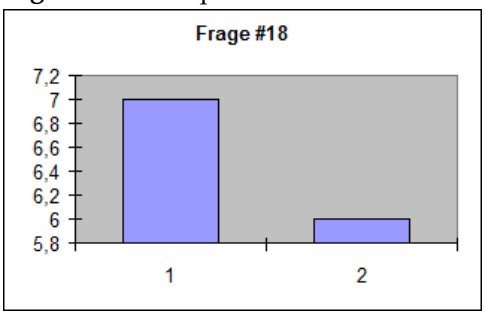

Abbildung 35: Gibt es ein Tutorial? - 1 = ja, 2 = nein

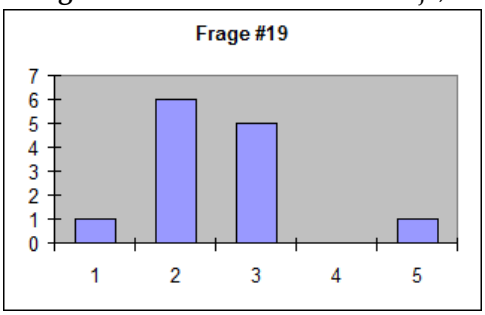

Abbildung 36: Wie sinnvoll findest du die Handlungen im Spiel?

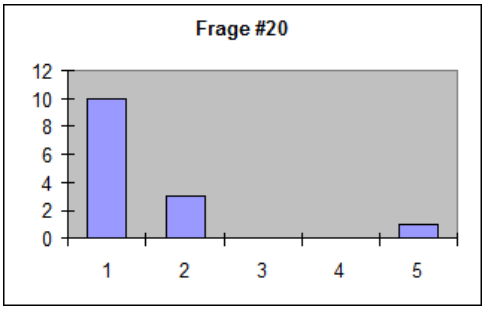

Abbildung 37: Wie hoch war die Überforderung beim ersten Spielen?

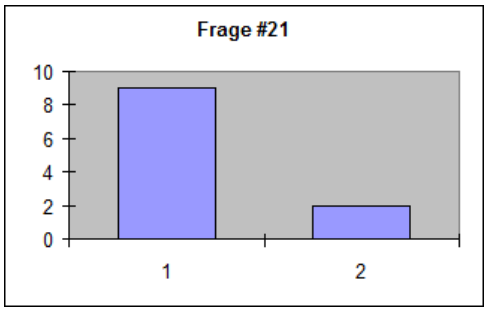

Abbildung 38: Würdest du das Spiel noch einmal spielen wollen, nachdem du es durchgespielt hast? - 1 = ja, 2 = nein

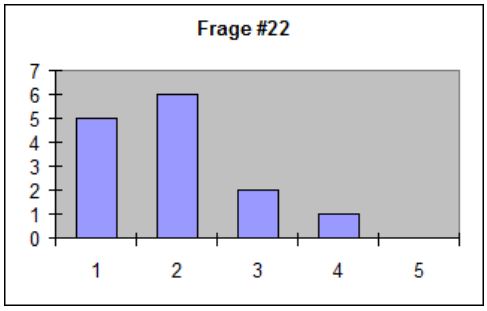

Abbildung 39: Wie sicher warst du damit, was zu tun ist?

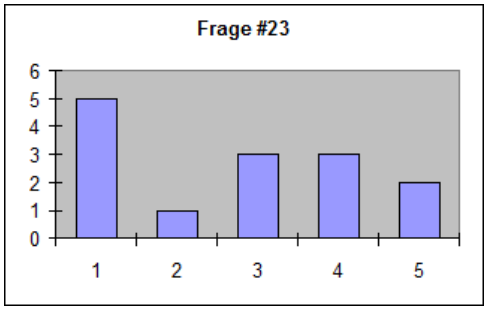

Abbildung 40: Warst du überrascht vom Gameplay? Wenn ja, wie sehr?

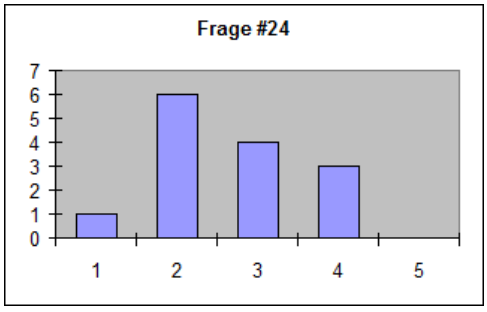

Abbildung 41: Wie hoch schätzt du den Spaß ein, den du hattest?

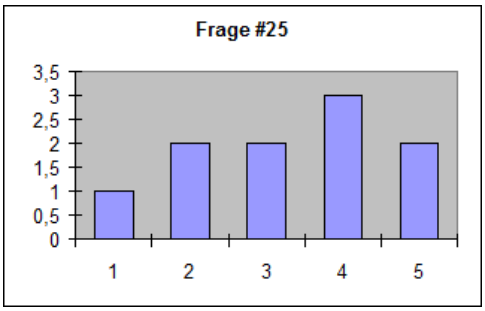

Abbildung 42: Wie neugierig warst du, zu erfahren, wie du an alle Items gelangst?

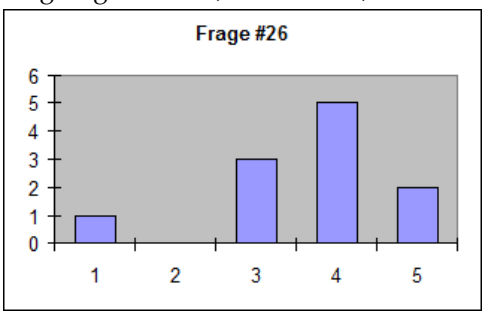

Abbildung 43: Wie wichtig waren dir die Items im Spiel?

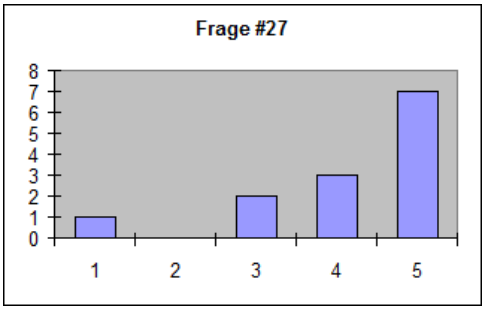

Abbildung 44: Denkst du, du konntest alle Problemstellungen im Spiel lösen?

5 Fazit und Ausblick

Fazit meiner Bachelorarbeit ist, dass ich gelernt habe, was Game Design bedeutet und wie Spieleentwicklung in den Grundlagen funktioniert. Ich fühle mich jetzt in der Lage, ein Spiel in Game Maker umsetzen zu können und habe Grundkenntnisse über die wichtigsten Elemente des Game Designs erhalten. Ich habe außerdem versucht, die Regeln guten Game Designs in meinem Spiel anzuwenden, indem ich mir ein interessantes Gameplay und eine kleine Geschichte ausdachte, die dazu passt. Ich habe durch das Design der Sprites und durch den Einsatz von Melodien und Geräuschen versucht, das Spiel ästhetisch und harmonisch wirken zu lassen. Ich habe mir Mühe gegeben, dem Spieler in einem Tutorial die Möglichkeit zu geben, die Steuerung zu üben. Ich habe es geschafft, mein Konzept umzusetzen und musste dabei einige Probleme umgehen und habe dies auch erfolgreich geschafft. Mein Ziel war es, mit einem neuen, ungewöhnlichen Gameplay den Spieler herauszufordern. Das ist mir auch gelungen. Allerdings musste ich auch erfahren, dass, wenn ich mein eigenes Spiel durchspielen kann, dies nicht bedeutet, dass das auch Andere können. Leider war mein Spiel für fast jeden Spieler zu schwer, sodass sie schnell aufgaben. Dieses Wissen werde ich in Zukunft nutzen, um zu vermeiden, dass mein Spiel den Spieler schon zu Beginn frustriert und er die Lust daran verliert. Insgesamt bin ich trotzdem mit dem Ergebnis zufrieden, da ich ein funktionierendes Spiel entwickeln konnte, das meinen ursprünglichen Vorstellungen entspricht.

6 Literatur

REF01 Ernest Adams, The Fundamentals of Game Design, Second Edition, 2008

REF02 Bob Bates, Game Design, Second Edition, 2010

REF03 Jesse Schell, The Art of Game Design, A Book of Lenses, 2010

REF04 http://www.makinggames.de/index.php/magazin/630_von_der_idee_zum_konzept