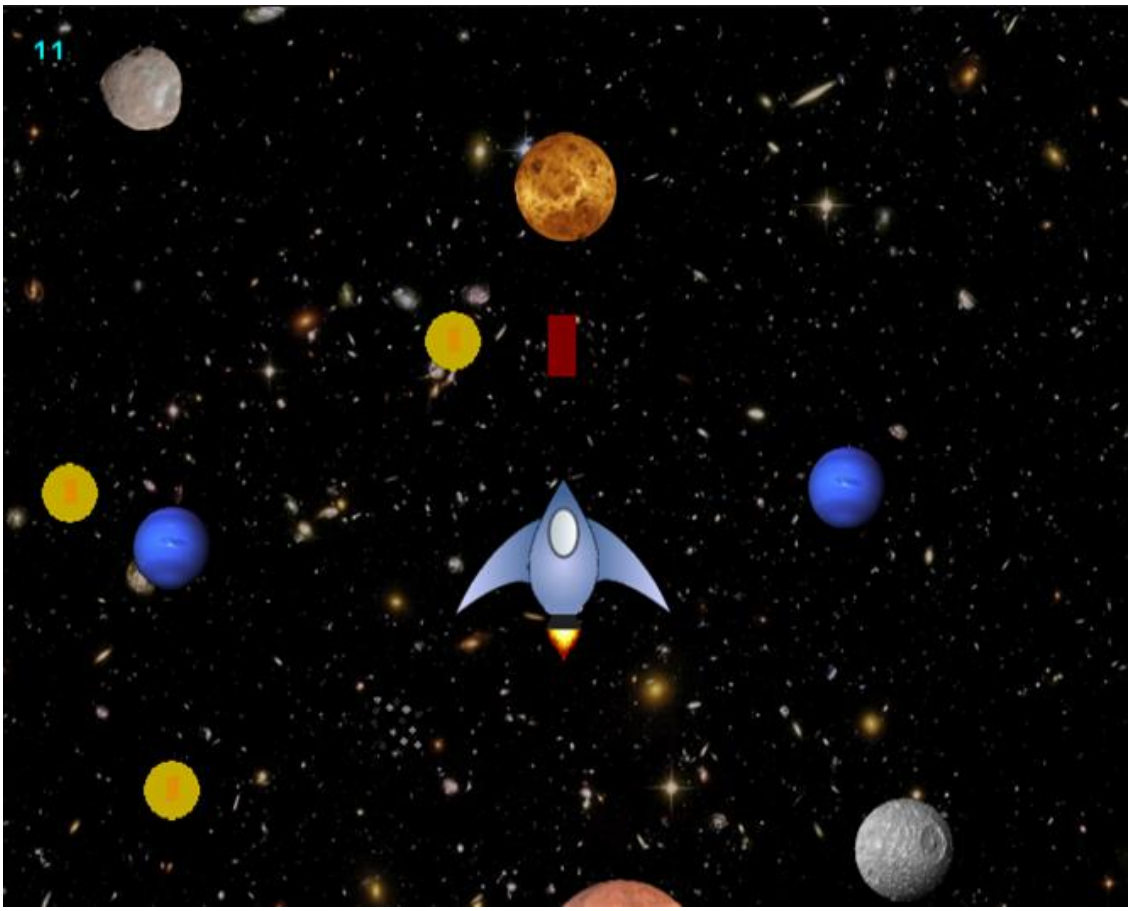
DIE ENTWICKLUNG EINES EIGENEN COMPUTERSPIELS



Maturaarbeit
2013/14
Semian Detreköy
Fach: Informatik
Betreuer: Dr. Rainer Steiger
Kantonsschule Schaffhausen

Inhaltsverzeichnis

Titebild: Ausschnitt aus dem ersten Level des Spiels

1. Einleitung

Schon zu Beginn der Kantonsschule habe ich mir Gedanken über ein mögliches Thema zur Maturaarbeit gemacht. Jedoch kam ich nie auf eine wirklich geniale Idee. Die Themen waren entweder zu umfangreich oder ich konnte mir nicht vorstellen längerfristig daran zu arbeiten. Doch dann kam im 3. Jahr der Moment, wo ich mich auf ein Thema festlegen musste. So überlegte ich mir, welche meine Lieblingsfächer sind und versuchte auf diese Weise den Themenbereich einzugrenzen. Da mich die Chemie sehr interessiert, versuchte ich in diesem Fach eine Arbeit zu finden. Dass Chemie oftmals sehr praktisch orientiert ist, hat mich gereizt. Trotz dessen konnte ich kein Thema finden, über das ich eine ganze Arbeit hätte schreiben wollen. So fragte ich in meiner Familie nach weiteren Ideen. Jedoch waren die meisten Vorschläge nicht wirklich nach meinem Interesse. Doch da kam der Vorschlag ein Computerspiel zu programmieren. Ich war zuerst eher abgeneigt gegenüber dieser Idee, da ich noch nie in meinem Leben programmiert hatte und ein solches Projekt einen sehr grossen Zeitaufwand mit sich bringen würde. Da die Zeit ein Thema auszusuchen inzwischen sehr knapp wurde und ich kein anderes mögliches Thema gefunden hatte, beschloss ich Samuel Vonäsch, einen Game Designer[1], den mein Bruder kennt, zu fragen, was er von einem solchen Maturaarbeitsthema halten würde. Dieser fand diese Idee sehr gut und meint auch, dass es für einen Anfänger gut möglich sei, ein einfaches Spiel zu kreieren. So beschloss ich für meine Maturaarbeit ein Computerspiel zu entwickeln.

Jedoch musste ich zuerst einen Betreuer finden. Als ich Kontakt mit den Informatikern der Kantonsschule aufnahm, stellte sich Rainer Steiger als mein Betreuer zur Verfügung.

Ich beschloss mein Spiel mit dem „Game Maker“ zu programmieren. Der Game Maker ist ein Programm, welches das Entwickeln von Spielen erleichtert. Es vereinfacht das Programmieren und gewährleistet somit einen sanften Einstieg in die Welt des Programmierens. Ausserdem wird durch dieses Programm, im Vergleich zum grundlegenden Programmieren nur mit Codes[2], viel Zeit gespart.

Zuerst werde ich in meiner Arbeit die Geschichte der Spieleentwicklung erläutern. Ausserdem werde ich darauf eingehen, wie Spiele heutzutage entstehen. Dies ermöglicht einen genaueren Einblick in die Welt der Spieleentwicklung. Damit man besser verstehen kann, worum es sich bei meiner Arbeit genau handelt, möchte ich anschliessend zeigen, was der Game Maker genau ist und wie man mit ihm arbeitet. Danach werde ich auf mein Projekt, ein Raumschiff-Geschicklichkeitsspiel, von der grundlegenden Vorbereitung bis zur finalen Optimierung eingehen. Im letzten Abschnitt werde ich noch ein Fazit aus meiner Arbeit ziehen.

[1] Gamedesign ist eine Tätigkeit, die sich im Bereich der Spieleentwicklung mit der theoretischen Konzeption des Spiels befasst.
[2] Codes sind schriftlich verfasste Befehle, die das Programm ausführt

2. Geschichte und Hintergrund der Spieleentwicklung[3]

2.1. Spieleentwicklung im Allgemeinen

Die Spieleentwicklung ist ein laufender Prozess von Veränderungen. Was in den 70er Jahren mit Arcade-Spieleautomaten[4] begonnen hat und zur heutigen Zeit mit Touch-Display[5] weitergetragen wird, hat eins gemein, die Massenunterhaltung durch Spiele. Massgeblich beeinflusst wurde die Veränderung durch den Wandel der Technik. Nicht nur 2D wurde zu 3D, sondern auch weitgehend in allen Spieleentwicklungsprozessen bieten die neuen Technologien mehr Möglichkeiten. Jedoch hat sich der grundlegende Prozess der Spieleentwicklung nicht verändert.

Der Entstehungsprozess ist hauptsächlich in 3 verschiedene Abschnitte zu unterteilen: Erstens das Game Design. Hier wird die eigentliche Spielidee entwickelt. Fragen wie „was ist das Ziel des Spieles“ oder „welche Gegner begegnet man im Spiel“ werden hier beantwortet. Die Game Designer legen den Grundstein der Spieleentwicklung. Ihre Ideen und Vorschläge werden dann im nächsten Arbeitsschritt von den Grafikern und Programmierern umgesetzt. Die Grafiker gestalten die ganzen Level und Charaktere im Spiel. Sie kümmern sich auch um die Musik. Sie erstellen sozusagen das Ambiente. Die Programmierer setzten die ganze Spielmechanik in Codes um. Sie sind dafür verantwortlich, dass das Spiel so läuft, wie es laufen soll.

Nun gibt es schon ein paar Veränderungen im Verlauf der Entwicklungsgeschichte. Beispielsweise waren früher alle Prozesse der Spieleentwicklung in einer Person vereinigt. Heutzutage wird jeder einzelne Arbeitsschritt unter Personen aufgeteilt. Natürlich trifft dies vorwiegend für grössere Spiele zu, da besonders im Bereich von Indie-Games[6] es auch vorkommen kann, dass nur eine Person am Spiel beteiligt ist. Genau genommen ist auch das hier entwickelte Spiel ein Indie-Game. Oftmals sind solche Spiele gratis und werden nur aus Freude an der Spieleentwicklung gemacht.

[3] Im Wesentlichen gestützt auf Internetquelle 2: Die Geschichte der Spieleentwicklung, Internetquelle 3: Spieleentwicklung von heute – YouTube – Bubble Universe Staffel 3

[4] Automaten bei denen durch Münzeinwurf Spiele gespielt werden können

[5] Touch Displays oder auch Touchscreens sind Bildoberflächen mit denen man durch Berührungen mit dem Gerät interagieren kann

[6] Indie-Games sind Spiele von unabhängigen Publishern

2.2. Gamedesign

Um jedoch genauer aufzeigen zu können, was sich in den letzten Jahren alles verändert hat, soll im Folgenden auf die einzelnen Teilbereiche der Spieleentwicklung eingegangen werden. Im Bereich des Gamedesigns hat sich einiges verändert. Grundlegend sind die Dimensionen, in denen man arbeitet, grösser geworden. Während man das Spielkonzept eines kleinen Spiels in einer Stunde ausarbeiten kann, gehören zu einem grossen Spiel viel mehr Faktoren, die beachtet werden müssen. Je umfangreicher ein Spiel ist, desto mehr Entscheidungen müssen getroffen werden. Beispielsweise kann man, bei einem einfachen Spiel wie Tetris, in drei Sätzen die ganze Spielmechanik definieren, weil nicht sehr viele Faktoren mitspielen. Bei komplexen Spielen jedoch ist jeder einzelne Spielabschnitt individuell und muss eigenständig definiert werden. Beispielsweise muss bei einem Rollenspiel[7] jede einzelne Quest[8] geplant werden und dies können schnell mehrere hundert Quests sein. Auch ein massiver Unterschied ist die Anzahl an Möglichkeiten, die der Game Designer hat, das Spiel zu definieren. Beispielsweise gibt es heutzutage unglaublich viele Arten, wie man ein Spiel steuern kann. Nicht nur Kontroller und Tastatur kann man miteinbeziehen, sondern auch Touch Screen oder Joystick[9]. Und da man auch in weiteren Bereichen der Spieleentwicklung, wie beim Art Work[10] oder Programmieren, mehr Mittel zur Verfügung hat, kann der Game Designer viel genauer die Ideen umsetzten, die er im Kopf hat und das Spiel vielfältiger gestalten. Jedoch ist die Arbeit an sich nicht anders geworden. Das Gamedesign ist trotz der starken Veränderungen eine Ideenfindungsarbeit geblieben.

2.3. Art Work

Im Bereich der Grafik und des Tons haben sich auch viele neue Optionen aufgetan. Bewegungen können mit Sensoren vom Menschen auf den Computer übertragen werde, um ziemlich realitätsgetreue Spielcharaktere zu schaffen. Damit können sich Figuren fast wie echte Menschen bewegen. Und durch die besseren Grafikprogramme kann eine detailgetreuere Umgebung geschaffen werden, falls dies beabsichtigt wird, denn nicht jedes Spiel setzt auf möglichst realistische Grafik. Ausserdem kann mittels professionellen Tonstudios fast jeder beliebige Sound produziert werden. Dies bietet die Möglichkeit die In-game[11] Welt so zu schaffen, wie man sie haben möchte. Egal ob realistisch, fantasiehaft oder im Comic Stil.

[7] Rollenspiele sind Spiele in denen der Spieler die Rolle einer fiktiven Figur übernimmt und mit dieser Abendteuer erlebt
[8] Aufgaben in Spielen werden oft Quest genannt
[9] Ein Joystick ist ein Art Steuerknüppel, der oft für Flugzeugsimulationen verwendet wird
[10] Art Work wird der grafische und soundbetreffende Prozess der Spieleentwicklung genannt
[11] In-game steht für „im Spiel“

2.4. Programmieren

Im Bereich des Programmierens hat sich die Arbeitsumgebung verändert. Statt das Spiel direkt zu programmieren, erstellt man sogenannte Tools. Dies sind programmierte Arbeitsoberflächen, die alle Elemente des Spiels, wie in einem Baukasten, enthalten. Diese Tools werden darauf von allen anderen an der Entwicklung beteiligten Arbeitsgruppen verwendet. Dies hat den Vorteil, dass alle Prozesse am gleichen „Ort“ gemacht werden können und somit der Datenaustausch vereinfacht wird. Ausserdem können die Grafiker und Level Designer selbständig arbeiten, ohne Absprachen mit den Programmierern. Sie können in diesem Tool selbst die Werte der Charaktere verändern und sie frei in der Spielewelt platzieren. Ausserdem können diese Tools oftmals auch für nachfolgende Spiele verwendet werden, falls diese ein ähnliches Spielprinzip und eine ähnliche Grafik verwenden. Es müssen nur kleine Korrekturen vorgenommen werden.

3. Einführung in den Game Maker

3.1. Grundlagen

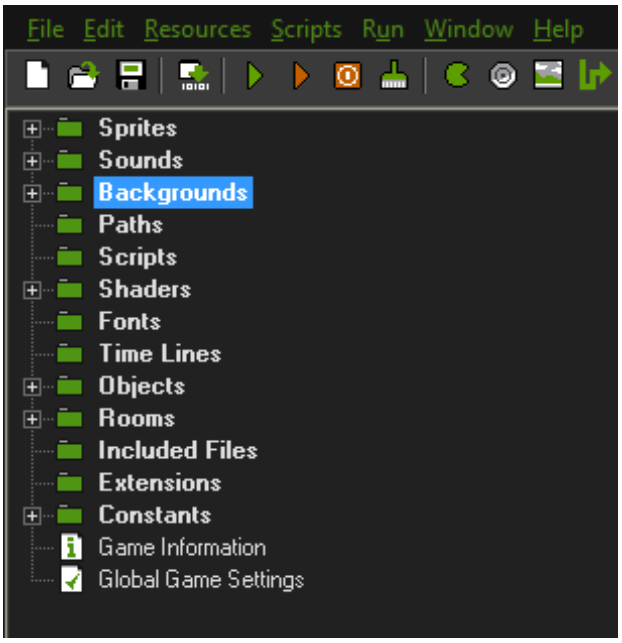


Abbildung 1: Hauptmenu des Game Makers

Um einen genaueren Einblick in diese Arbeit zu bekommen, ist es wichtig zu verstehen, was der Game Maker ist und wie man mit ihm arbeitet. Der Game Maker benutzt eine grafische Oberfläche, die die Arbeit nicht nur erleichtert, sondern auch sehr viel Zeit spart.

Über die Startseite gelangt man in alle Bereiche des Game Makers. Hier sind auch die bereits erstellten Daten aufgelistet. Im Game Maker kann das Spiel direkt getestet werden, ohne es zuvor installiert zu haben. Angenehm am Game Maker ist, dass man mit ihm nicht nur Spiele für Windows machen kann, sondern auch für Mac oder Smartphones.

3.2. Grafiken

3.2.1. Sprites

Die Grundlage des Spieles sind die grafischen Bilder. Diese werden Sprites genannt. Alle sichtbaren Objekte im Spiel brauchen ein Aussehen. Dieses wird ihnen im Sprite Menu gegeben, das hier abgebildet ist. Für einen Sprite können Bilder importiert[12] oder auch selbst gezeichnet werden. Das Bild rechts zeigt den Sprite des Portals, das in diesem Spiel den Anfang eines Levels und dessen Ende markiert.

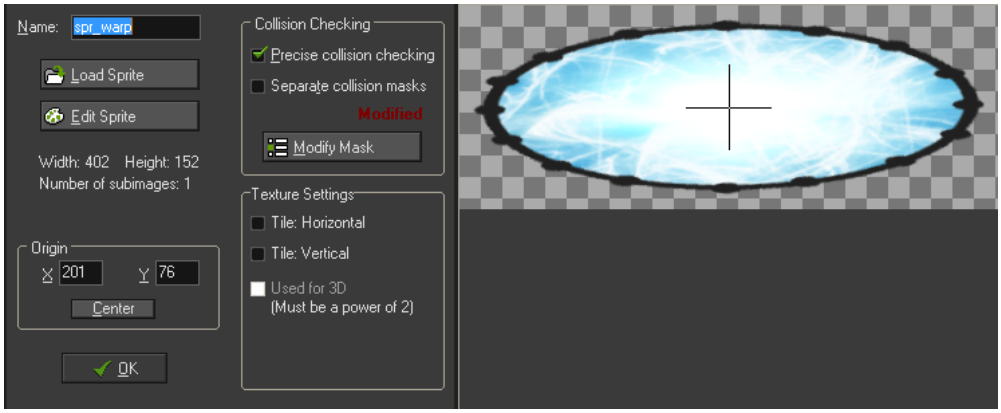


Abbildung 2: Spritemenu des Portals

[12] Importieren bedeutet hier Daten vom Computer ins Spiel einzuführen

3.2.2. Grafik Editor

Wenn man ein Bild selbst zeichnen oder ein gegebenes Bild editieren[13] möchte, leistet der Game Maker auch hier Hilfestellung. Mit dem eingebauten Editor lassen sich einfache Bilder selbst gestalten. Er ist zu vergleichen mit Bildbearbeitungsprogrammen, wie Photoshop oder Paint.net. Es bleibt anzumerken, dass der Editor nur sehr beschränkte Möglichkeiten bietet und daher ein externes grafisches Programm zu bevorzugen ist, falls man grafisch hochwertige Bilder designen möchte. Hier ist eine selbstgezeichnete einfache Münze abgebildet, die im Spiel den Punktestand um 2 Punkte erhöht.

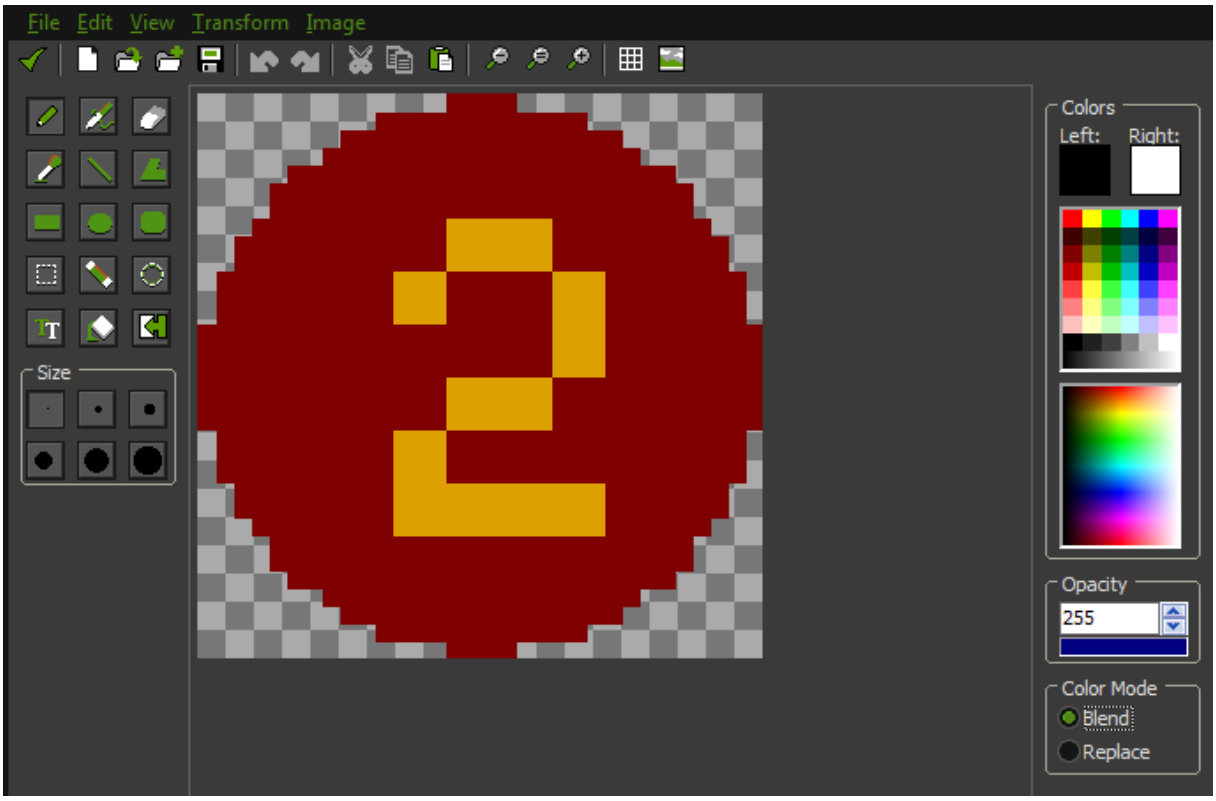


Abbildung 3: Grafik Editor mit einer zwei Punkte Münze

[13] Editieren bedeutet Bearbeiten

3.2.3. Collision Mask

Von enormer Wichtigkeit ist bei den Sprites die „collision mask“. Diese gibt an, ab wann im Spiel eine Berührung mit diesem Sprite[14] Auswirkungen hat. Zwei Objekte berühren sich im Spiel erst dann, wenn sich deren „collision masks“ berühren. Im Normalfall entspricht die „collison mask“ genau dem optischen Bild und in diesem Spiel ist dies auch immer der Fall. Es muss jedoch nicht immer so sein. Deshalb hat der Game Maker diese Bearbeitungsfunktion eingebaut. Hier sieht man die „collision mask“ von dem Portal. Die collision mask ist gelb markiert.

Abbildung 4: collision mask vom Portal

[14] Eigentlich ist hier das Objekt mit dem Aussehen des Sprites gemeint, darüber später jedoch mehr

3.3. Objekte

3.3.1. Grundlagen

Wenn man nun einen Sprite erstellt hat, muss man daraus ein Objekt machen, damit man es im Spiel platzieren kann. Ein Objekt besteht grundsätzlich aus 3 Teilen:

(1) Einem Sprite, der das Aussehen bestimmt. Dieser Teil fällt weg, falls das Objekt kein Aussehen hat, sondern nur eine Funktion. Dies kann beispielsweise der Fall sein, wenn man ein Objekt hat, das nur die Lautstärke reguliert, aber nicht visuell am Spiel teilnimmt.

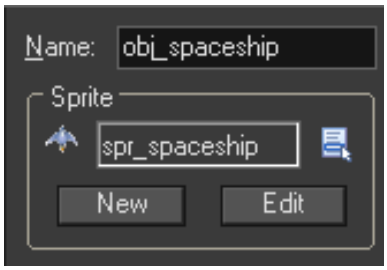


Abbildung 5: Sprite Auswahl des Raumschiffes

(2) Ein Event (Voraussetzung) bei dem etwas geschehen soll. Beispielsweise könnte dies ein Mausklick oder eine Berührung mit einem andern Objekt sein. Nur falls ein solches Event im Spiel aktiviert wird, geschieht auch etwas.

Abbildung 6: Events des Raumschiffes

(3) Und einer Aktion, die dem Objekt sagt, was geschehen soll, falls das entsprechende Event aktiviert wurde.

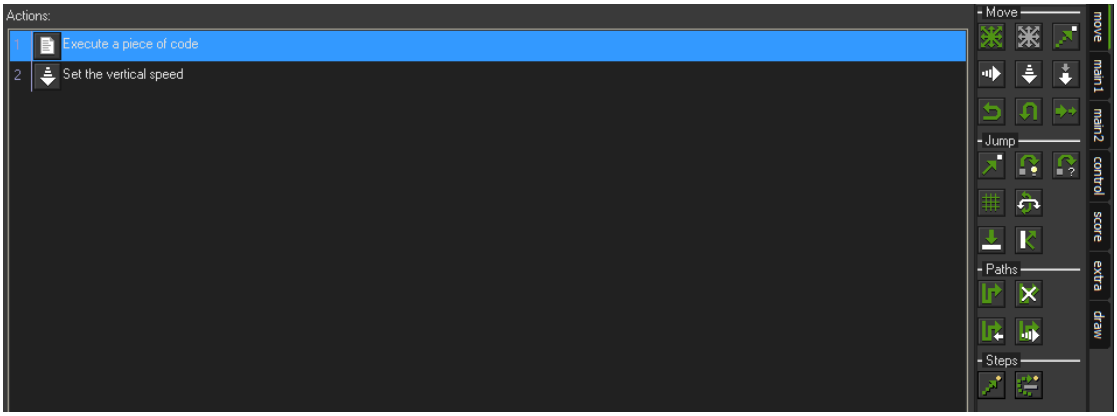


Abbildung 7: Aktionen des Raumschiffes

Um dies genauer zu erläutern, ist hier das Raumschiff, also ein sehr zentrales Objekt, abgebildet. Auf der linken Seite sind das Aussehen (Sprite) und die Zustände des Raumschiffes definiert. Zustände sind zum Beispiel, ob das Raumschiff sichtbar ist oder ob Gravitation auf das Raumschiff wirken soll. In der Mitte sind alle Events aufgelistet, bei denen das Raumschiff etwas machen soll. Auf der rechten Seite sind die Folgen (Aktionen), die aus diesen Events resultieren. Diese gelten nicht nur für das Raumschiff, sondern können jedes beliebige Objekt betreffen. Hierbei unterscheidet man zwischen 2 Aktionen. Den Drag n` Drop (D&D)[15] Aktionen und den Codes.

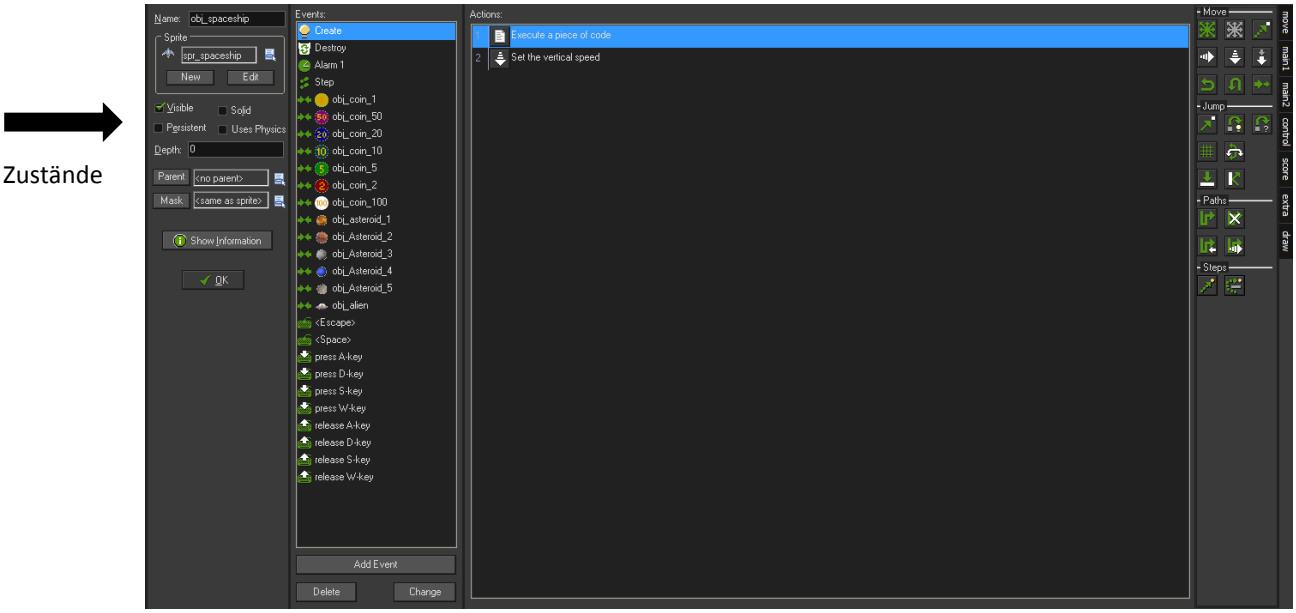


Abbildung 8: Objektmenu des Raumschiffes

[15] Drag n` Drop oder kurz D&D bedient sich vorgefertigten Codes, die man wie Bauklötze miteinander verbinden kann um einfach Programme zu erstellen

3.3.2. Aktionen

3.3.2.1. Drag n`Drop Aktionen

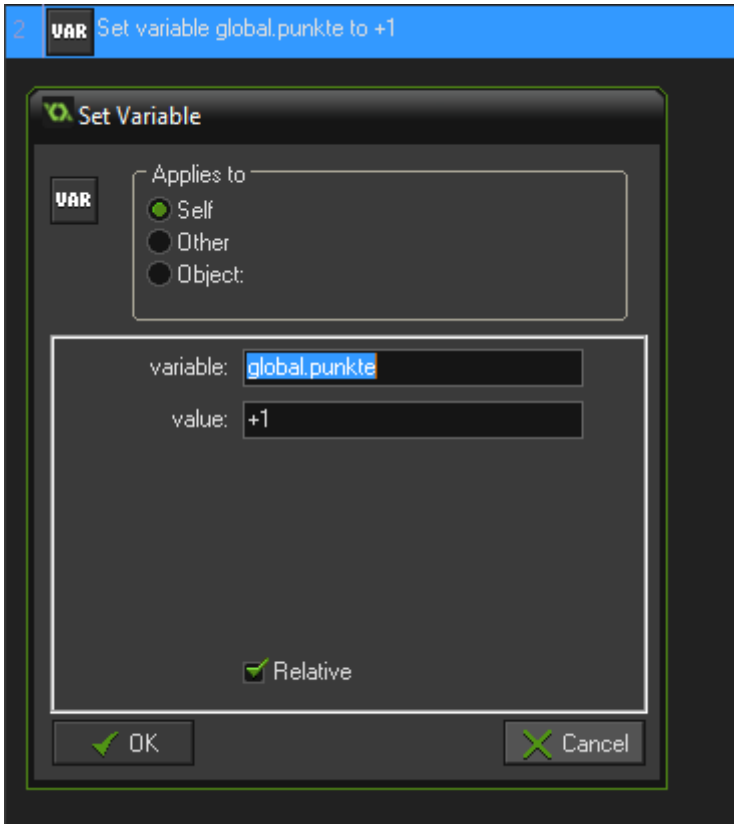


Abbildung 9: D&D Variable der Punkte

Die D&D Aktionen sind schon vorgefertigte Aktionen. Hier ist ein Beispiel einer solchen D&D Aktion zusehen. Diese Aktion kann den Wert einer Variablen[16] verändern. Man kann eingeben, welche Variable verändert werden soll. Darunter steht, um welchen Wert diese verändert werden soll. Zuletzt kann man bestimmen, ob dies ein fixer Wert sein soll oder ob dieser Wert zum bisherigen dazugerechnet werden soll. Diese D&D Aktion wird in diesem Spiel verwendet, wenn das Raumschiff mit einer Münze des Wertes 1 in Berührung kommt. Bei der Berührung wird der Punktestand um 1 erhöht.

Es könnte auch als Code geschrieben werden. Jedoch geht es auf diese Weise schneller und Veränderungen sind leichter zu machen, da eine D&D Aktion meist übersichtlicher ist. Bei diesem Beispiel wäre der Code jedoch sehr simpel, wie sie hier sehen können.

Abbildung 10: Code Variable der Punkte

[16] Eine Variable ist ein Wort oder Buchstabe, der eine Zahl zugeordnet wird

3.3.2.2. Codes als Aktionen

In diesem Spiel werden die Codes für komplexere Aktionen verwendet, die nicht mehr zureichend durch die D&D Funktion definiert werden können. Ein Beispiel für eine solche Aktion ist die „Schuss Funktion“ des Raumschiffes. Diese Aktion tritt in Kraft, wenn das Event „Leertaste drücken“ aktiviert wird. Die zweite und dritte Zeile enthält die eigentliche Aktion. Nämlich wird das Objekt „Schuss“ an der aktuellen Position des Raumschiffes kreiert und fliegt nach vorne mit einer Geschwindigkeit von 6. Die 90 in der dritten Zeile steht für die Flugrichtung, die hier nach oben ist. Die anderen Zeilen sind dafür da, dass man nur alle 1,5 Sekunden schiessen kann und sie stehen in Verbindung mit anderen Events. Die Aktion wird nur ausgeführt, wenn die „Variable“ Schuss aktiviert wird. Dies ist zu Beginn des Levels der Fall, denn eine andere Aktion, die sich zu Beginn jedes Levels abspielt, aktiviert den Schuss. Nach einem abgegebenen Schuss, wird dieser zuerst deaktiviert, danach wird ein Timer gestellt, der auf 45 steps, das entspricht 1,5 Sekunden, gestellt wird. Bei Ablauf des Timers wird der Alarm[1] ausgelöst, dies ist eine andere Aktion, die den Schuss wieder aktivert.

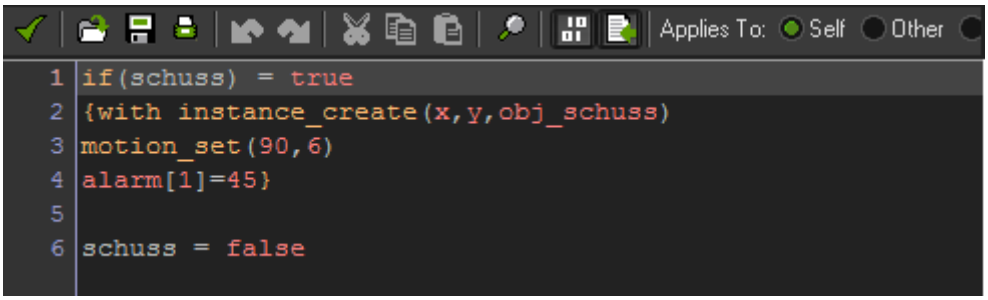


Abbildung 11: Code vom Schuss

3.4. Raum

Wenn man nun alle Objekte hat, kann man diese in einen Raum einfügen. Im Raum können Einstellungen betreffend dem Hintergrund, der Sicht des Spielers im Spiel selbst, den Objekten, der Grösse des Raumes und vieles mehr gemacht werden. Jedoch ist wohl der wichtigste Teil das Platzieren der Objekte. Durch einfaches Auswählen des gewünschten Objektes und hineinziehen in den Raum wird dieses platziert.

Der Game Maker bietet noch viel mehr Möglichkeiten für das Gestalten eines Spieles, wie beispielsweise das Hinzufügen von Musik. Dies erlaubt es schon allein durch die erwähnten Grundlagen ein beeindruckendes Spiel entwickeln.

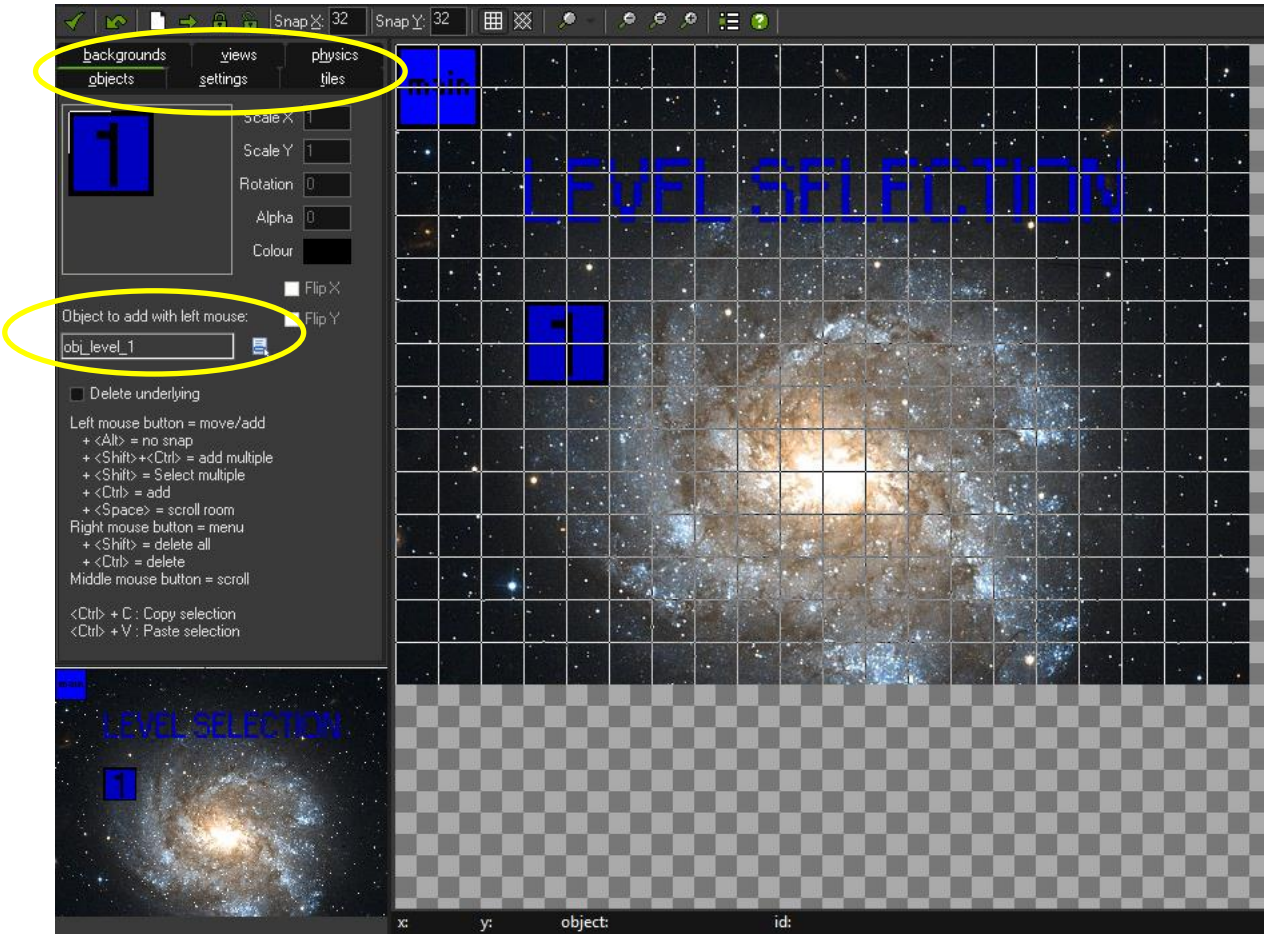


Abbildung 12: Raum des Level Selects

4. Die Entwicklung eines eigenen Spiels

4.1. Vorbereitung

Als Erstes musste die Durchführbarkeit eines solchen Projektes abgeklärt werden. Dazu wurde Kontakt mit einem Game Designer aufgenommen. Es galt abzuklären, ob man ein solches Projekt ohne Vorkenntnisse überhaupt bewältigen kann. Doch dieser war begeistert von der Idee und äusserte sich positiv zur Absicht ein Computerspiel zu programmieren. Er meinte, es sei zeitlich und leistungstechnisch gut machbar ein kleines Flashgame[17] im Rahmen einer Maturaarbeit zu entwickeln.

4.1.1. Programmiersprache

Das Projekt, ein eigens Spiel zu designen, stand nun fest. Als Erstes musste eine Programmieroberfläche ausgewählt werden. Auf Anfrage schlug der Game Designer unter anderem Java, c++, Game Maker und ActionScript vor. Nun mussten die Vor- und Nachteile der verschiedenen Programmiersprachen und Plattformen abgewogen werden.

(1) Java ist eine Programmsprache, die sehr einsteigerfreundlich ist und bevorzugt bei kleinen Games eingesetzt wird. Sie ist sehr leistungsverbrauchend und somit ungeeignet für grössere Projekte. Da das geplante Spiel nicht umfangreich werden sollte, wäre dies kein Problem.

(2) C++ ist eine sehr komplexe und die meist verwendete Programmiersprache der Welt. Sie wird so häufig wegen ihrer Effizienz und Kompaktheit verwendet, was eine geringere Computerleistung von Nöten macht.

(3) ActionScript[18] ist eine Programmiersprache von Adobe System, die für Adobe Flash, Flex und AIR[19] entwickelt wurde. Wegen Unkenntnissen über diese Programmiersprache und keinen bedeutenden Vorteilen, wie Einsteigerfreundlichkeit oder Vielfältigkeit, war klar, dass sie für dieses Spiel nicht in Frage kommen würde.

(4) Der Game Maker ist eine Programmierumgebung und keine Programmiersprache. Sie verwendet als Programmiersprache Game Maker Language, die an bekannte Programmiersprachen, wie Java und C angelehnt ist.

[17] Flashgames sind Spiele die mit Adobe Flash programmiert wurden. Es steht jedoch oft auch als Synonym von kleinen, einfachen Spielen

[18] Informationen aus Internetquelle 1: Wikipedia: ActionScript /Adobe Flash / Flex / AIR

[19] Adobe Flash und Flex dienen der Erstellung von Programmen, die auf Adobe basieren. AIR ermöglicht die Erstellung von Desktop-Anwendungen, die auf Adobe basieren

Eine Programmierumgebung ist eine grafische Oberfläche, die den Prozess des Programmierens erleichtert. Im Game Maker ist es möglich, selbst Codes zu schreiben oder per D&D zu programmieren. Das D&D System erlaubt es schnell und anfängerfreundlich zu arbeiten, doch auf Kosten der Vielseitigkeit.

Da bisher noch keine Erfahrungen im Programmieren bestanden, wurde beschlossen, das anfängerfreundliche Java oder den Game Maker zu verwenden. Zu Beginn bestand die Meinung, dass D&D kein „wahres“ Programmieren sei. Deshalb sollte Java verwendet werden. Es wurde jedoch festgestellt, dass die D&D Funktion gar nicht verwenden werden muss. Trotzdem bestände die Möglichkeit, auf diese Funktion zurückzugreifen, falls Probleme mit dem „normalen“ Programmieren bestehen würden. So wurde beschlossen den Game Maker zu verwenden. Später wurde bemerkt, dass der essenzielle Teil des Programmierens nicht das Codeschreiben selbst ist. Vielmehr sind es die Überlegungen, die hinter dem Code stecken, wie man ein Problem auf eine grundlegende Art in einen Code fassen kann. Dieser Prozess ist zu vergleichen mit dem, wie man ein Objekt im Mathematik- oder Physikunterricht mit Zahlen und Zeichen aufs Genauste beschreibt. Da man weniger tief in die Materie eintaucht, macht das Programmieren mit D&D trotzdem weniger Spass, als das Programmieren mit Codes. Es ist, als ob man bei D&D Moleküle betrachtet, aber beim Programmieren Atome. Man versteht viel genauer im Detail, was vor sich geht, wenn man etwas selbst programmiert hat. Dem Umgang mit Codes wird zusätzlich ein persönlicher Vorzug gegeben, im Gegensatz zu grafischen Funktionsblöcken, die man zusammen bastelt. Dies ist schwierig zu erklären, jedoch kann man sich das ähnlich vorstellen, wie dass man Mathematik, als Sprache, der literarischen Sprache vorzieht. Trotz der wachsenden Begeisterung für das Programmieren, wurde vieles mit der D&D Methode programmiert, da eine vollständige Programmierung mit Codes zu zeitaufwändig gewesen wäre. Somit wurden die Codes nur dann verwendet, wenn man mit der vereinfachten D&D Methode nicht mehr weiter kommen konnte.

4.1.2. Spieloberfläche

Es war nicht von Anfang an klar, ob das Spiel für den Computer oder das Smartphone programmiert werden sollte, da der Game Maker beide Varianten unterstützt. Es wurde beschlossen das Spiel für den Computer zu programmieren, da dazumal noch kein Smartphone besessen wurde und somit der Computer naheliegender war. Falls noch Zeit übrig bliebe, wäre es immer noch möglich das Spiel für das Smartphone zu programmieren. Es war damals noch schwierig den Umfang einer solchen Arbeit einzuschätzen, weil sich alles in einem unbekannten Umfeld abspielte. Deshalb wurde entschieden die Dimensionen des Spiels noch nicht definitiv festzulegen.

4.1.3. Tutorial

Vor den Überlegungen, was man für ein Spiel programmieren könnte, musste zuerst der Umgang mit dem Game Maker gelernt werden. Um dies zu erreichen, wurde beschlossen zuerst ein Tutorial, das ist eine Einführung zu diesem Programm, zu machen. In dieser Einführung wird einem Schritt für Schritt erklärt, wie man ein sehr einfaches Spiel mit D&D programmiert. In diesem Spiel bewegt sich ein Clown in einem abgesperrten Raum und prallt wie eine Billardkugel von den Wänden ab. Ziel des Spiels ist es mit der Maus auf den Clown zu klicken, der sich bei Erfolg schneller zu bewegen beginnt. Es gab nur wenige Probleme mit dieser Einführung und das Gefühl für die grundlegenden Funktionen des Programms war schnell da.

4.1.4. Spielidee

Nun folgten die Überlegungen, was für ein Spiel es werden sollte. Es kamen Ideen auf, ein Rollenspiel in 2D zu programmieren. Es war schon von Anfang an klar, dass 3D ein zu schwieriges Unterfangen wäre. Diese Art von Spiel macht Freude. Auch waren schon sehr konkrete Ideen da, wie das Spiel aussehen sollte. Die ganze Karte wäre in Quadrate aufgeteilt, auf denen man sich bewegt. Ziel des Spiels sollte sein, dass man mit einem Charakter eine Geschichte durchlebt, um stärker zu werden. Es hätte in einer Mittelalter- oder Fantasiewelt gespielt. Der Kampfmodus wäre ähnlich dem von dem bekannten Spiel Pokémon gewesen. Rundenbasierend wären abwechselnd Spieler und Gegner am Zug gewesen, um ihre gewünschte Handlung auszuwählen. Jedoch entstanden Zweifel an der Durchführbarkeit eines solchen Projekts, da nicht nur das ganze Spiel programmiert, sondern noch ein Kampfsystem entwickelt und eine Story geschrieben werden müsste. Ein solches Spiel wäre nur attraktiv, wenn es einen grossen Umfang hätte, da eine lange gute Geschichte von einem Rollenspiel erwartet wird. So wurde beschlossen weiter zu überlegen. Bei einem Abendessen entstand dann eine überzeugende Idee für ein Spiel. Es wurde beschlossen ein Weltraumspiel zu programmieren, indem das Ziel wäre, möglichst lange zu überleben, während Asteroiden und andere Gefahren auf einen zufliegen. Mit der Zeit sollte das Spiel immer schwieriger werden. Das bedeutet, je länger man überlebt, desto mehr Asteroiden sollten erscheinen. Die Punktzahl wäre die Zeit gewesen, wie lang man durchgehalten hätte ohne zu sterben.

4.2. Erster Spielansatz

Mit den neu erlangten Kenntnissen entstand der Entscheid, dieses Spiel so zu programmieren, dass sich das Raumschiff nur nach links oder rechts bewegen kann. Grundlegende Objekte, wie ein Raumschiff und Asteroiden, wurden programmiert. Jedoch kamen einige markante Probleme auf. Die Asteroiden erschienen nicht zufällig irgendwo am oberen Ende des Spiels, wie es geplant war, sondern zufällig im ganzen Raum verteilt. Um dieses Problem zu lösen, hätte ein Algorithmus programmiert werden müssen, der all Asteroiden gleichzeitig steuert. Dies war mit den bekannten Kenntnissen nicht möglich. Das nächste Problem war, dass sich die Asteroiden nur konstant in eine Richtung bewegten. Weil das Asteroidenproblem nicht gelöst werden konnte, wandte man sich vorerst anderen Themen zu. Der Beschluss, das Raumschiff nur in 2 Richtungen fliegen zu lassen, wurde revidiert, um die Komplexität des Spiels zu erhöhen. Somit wurde das Raumschiff so programmiert, dass es in alle vier Himmelsrichtungen fliegen konnte. Unglücklicherweise konnte es nicht schnell die Richtung ändern, sondern brauchte eine Sekunde dazu, da sich zwei Funktionen überschnitten. Einerseits soll sich das Raumschiff beim Loslassen der einen Taste aufhören zu bewegen und andererseits soll es beim Drücken einer anderen Taste in eine neue Richtung fliegen. Somit musste das Raumschiff zuerst stillstehen, damit man die Richtung ändern konnte. Um den Richtungswechsel visuell darzustellen, mussten zuerst Erfahrungen mit der Bildbearbeitungsfunktion des Game Makers gemacht werden. Einfache Funktionen halfen das Bild zu drehen, zu vergrössern oder zu verkleinern. Damit das Raumschiff sich nicht aus dem Spiel heraus bewegt, wurde eine Wand am Rande des Raumes platziert, die das Raumschiff bei Berührung zerstörte. Dies erfüllte seinen Zweck, jedoch sah die Wand visuell nicht sehr schön aus. Ausserdem wurde ein Knopf eingefügt, der beim Drücken das Spiel neu startete. Das schon genannte Problem mit den Asteroiden konnte jedoch noch immer nicht gelöst werden.

4.3. Zweiter Spielansatz

Da man zu keiner Lösung kam, gab es Probleme mit der Arbeit voranzuschreiten. Beim Versuch die Situation im Ganzen zu überblicken, entstand die Frage, ob vielleicht zu fixiert mit einer Vorgehensweise gearbeitet wurde. Es wurde in Betracht gezogen das Spiel grundlegend anders zu gestalten, sodass das Problem mit den Asteroiden umgangen werden kann und das Spiel so gestaltet wird, dass es besser auf den Game Maker und seine Möglichkeiten abgestimmt ist. Man begann mit dem Versuch das Spiel so aufzubauen, dass die Asteroiden fix sind und sich nur das Raumschiff bewegt. Dieses sollte sich jedoch ständig bewegen, sodass man es nur beschleunigen oder abbremsen kann.

Zu diesem Zeitpunkt kam die Frage auf, wie das Spiel von anderen Spielen abgegrenzt werden soll, um attraktiver zu wirken. Es entstand die Idee Wurmlöcher ins Spiel einzubauen, die das Raumschiff teleportieren würden. Eine andere Idee war, Level mit mehreren Etagen zu machen, die miteinander verbunden sind. Jedoch wurde erkannt, dass ein solches Projekt nicht im Bereich der Möglichkeiten liegen würde. Ausserdem bestand keine Absicht ein Spiel zu kreieren, das völlig anders sein würde, als alle bisherigen. Dies wäre schlicht und einfach nicht möglich, da es beinahe jeden Spieltyp bereits gibt und dieses Spiel nicht dem Zweck dient, kommerzialisiert zu werden. So blieb es bei der ursprünglichen Idee.

4.4. Art Work

Schon früh wurde, beim Versuch das neue Spielkonzept umzusetzen, bemerkt, dass man auf gutem Wege war. Es wurde sogar damit begonnen eigene Sprites selbst zu gestalten. Für die neue Spielidee war es von enormer Wichtigkeit, dass die Kamera dem Raumschiff folgt, denn dieses ist ständig in Bewegung. Darum wurde das Spiel so programmiert, dass man immer ein gewisses Sichtfeld um das Raumschiff herum hat und sich dieses mit dem Raumschiff mitbewegt. Kaum war das Problem gelöst, wurde eine andere Schwierigkeit angepackt. Man musste sich mit dem Urheberschutz befassen. Egal ob Bild oder Ton, man kann nicht irgendwelche Quellen verwenden. Die meisten Bilder sind geschützt, aber das Spiel kann nicht auf sie verzichten. Somit wurde im Web nach einer Möglichkeit gesucht, einfach an ungeschütztes Material heranzukommen. Durch die Suche wurde eine Suchmaschine gefunden, die genau dies bezweckt. Sie stellt Bilder und Musik zur Verfügung, die nicht urheberrechtlich geschützt sind oder besser gesagt, sie verlinkt auf Seiten mit Medien, die nicht urheberrechtlich geschützt sind. Durch die neu gewonnene Suchmethode wurden die bisher kreierten Objekte durch legal nutzbare ersetzt. Viele davon wurden verändert, wie beispielsweise das Raumschiff, das fast komplett selbst gestaltet wurde. So wurde auch die Düse des Raumschiffes selbst in kleiner Pixelarbeit gezeichnet. Auch alle anderen später dazukommenden Medien konnten durch diese Suchmaschine ausfindig gemacht werden.

4.5. Programmierung

4.5.1. Hauptmenu

Das Startmenu ist bei einem Spiel von enormer Wichtigkeit. Dieses war der erste definitive Spielinhalt, der programmiert wurde. In diesem Menu kann man ein neues Spiel starten, ein Spiel speichern oder laden. Falls man keine Lust mehr hat weiterzuspielen, kann man das Spiel dort beenden. Die Funktion, wie man das Spiel speichert und lädt, ist im Game Maker leider fehlerhaft. Deshalb musste eine eigene Idee entwickelt werden. Man kam zum Schluss, dass man nur zwei Daten speichern muss: Zum einen der Punktestand und zum anderen das momentane Level. Es wurde beschlossen eine Funktion zu machen, die ein INI-File[20,21] erstellt und die Daten danach in diesem speichert, wenn man auf den Speichern-Knopf drückt. Wenn man nun den Laden-Knopf drückt, wird das entsprechende INI-File geöffnet und die Daten werden ins Spiel übertragen.

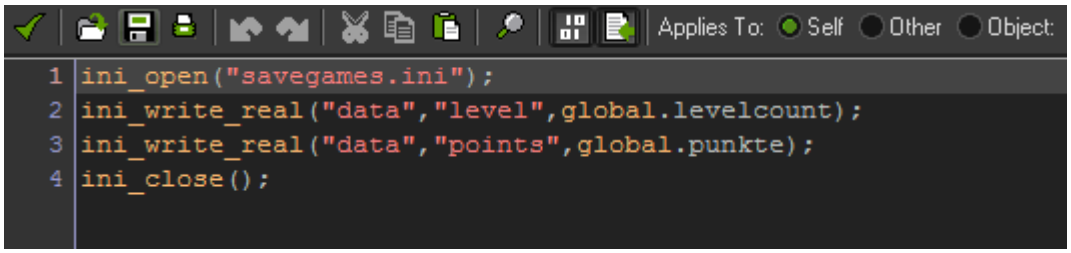


Abbildung 14: Code für das Speichern

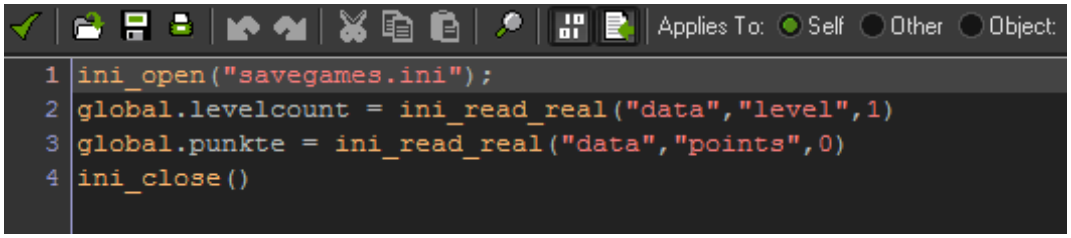


Abbildung 13: Code für das Laden

[20] Ein INI-File ist eine Art Textdokument

[21] Informationen zur Verwendung von INI-Files im Game Maker aus Internetquelle 4: Game Maker – INI Files – YoutTube

4.5.2. Level Auswahl

Beim Konkretisieren der Ideen, nahm das Spiel nun klare Formen an, wie beispielsweise, dass das Spiel 10 Level haben soll. Man kann ein Level beliebig oft wiederholen, jedoch wird das nächste Level erst freigeschaltet, wenn das vorherige erfolgreich beendet wurde. Es erlaubt auch später wieder zurückzugehen und die ersten Level erneut zu spielen. Diese Funktion, wurde „Level Select“ genannt und war nicht einfach zu programmieren. Es galt zuerst herauszufinden, wie man verhindern kann, dass man bereits zu Beginn Level 10 spielen kann, sondern erst nach dem Durchspielen von Level 9. Um dieses Problem zu lösen, wurde eine Variable eingeführt, die zu Beginn des Spieles den Wert 1 hat. Wenn man nun zum Level Select Menu geht, erscheint nur Level 1 zur Auswahl. Wird nun das Ziel von Level 1 erreicht, wird diese Variable um eins erhöht. Jedoch musste sichergestellt werden, dass diese Variable nur beim ersten Mal durchspielen des Level 1 erhöht wird. Aus diesem Grund wurde hinzugefügt, dass der Wert nur um 1 erhöht wird, wenn die Variable den Wert 1 besitzt. Dadurch hat die Variable nach abschliessen von Level 1 den Wert 2. Im Level Select stehen nun Level 1 und Level 2 zur Auswahl. Und so geht das weiter bis Level 10.

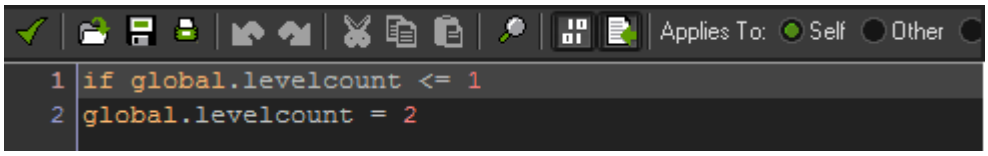


Abbildung 15: Code für Level Select

4.5.3. Punkte

Um das Spiel interessanter zu gestalten, wurden Münzen ins Spiel eingeführt, die beim Einsammeln die Punktzahl erhöhen. Die Punktzahl sollte oben links angezeigt werden. Anhand dieser Münzen kann anschaulich gezeigt werden, dass der Prozess, eine Funktion oder ein Objekt einzufügen, sehr aufwändig ist und viele Rückschläge beinhaltet. Zur Vereinfachung wird in dieser Arbeit nicht jeder einzelne Ansatz von jedem Objekt behandelt, sondern nur der endgültige Code. Um jedoch einmal zu zeigen, wie dieser Prozess aussieht, wird dies anhand des Punktesystems gezeigt.

Zu Beginn hat man eine grobe Idee, wie das Objekt aussehen soll. Hier ist dies die oben genannte Beschreibung. Danach wurde die Idee konkretisiert. Dies bedeutet, dass es verschiedene Münzen mit verschiedenen Werten gibt. Durch die Berührung mit dem Raumschiff wird eine Variable um den entsprechenden Wert der Münze erhöht. Bei der Berührung mit dem Raumschiff müssen diese Münzen dann verschwinden. Die Punktzahl gibt den Wert der Variable an und der Betrag ist zu Beginn Null. Die Münzen müssen zufällig irgendwo im Raum erscheinen. Beim Programmieren konnte alles ausser der Punkteanzeige direkt gemacht werden, da nur die Punkteanzeige eine neue Problemstellung war. Deshalb wurde das Game Maker Wiki nach entsprechenden Befehlen durchsucht. Doch diese funktionierten nicht, denn nur die Kamera bewegte sich, die Anzeige aber nicht. Somit musste nach einem Code gesucht werden, der die X und Y Koordinate relativ zur Kamera angibt. Dies gelang, jedoch war die Anzeige schwarz und im Weltall sieht man dies nur sehr schlecht. Also musste noch nach einem Code gesucht werden, um die Anzeigefarbe zu verändern. Diesen fand ich im Game Maker Wiki. Schliesslich war es geschafft, die Münzen und ein Punktesystem in das Spiel einzuführen. Es ist meist so, dass nicht direkt der finale Code beim ersten Versuch zustande kommt. Durch Testen stellt sich heraus, was nicht wie gewünscht funktioniert. Der Fehler wird gesucht und behoben, was sehr lange dauern kann. Es wird erneut getestet und die Fehler behoben, bis man die gewünschte Funktion erreicht hat. Somit bringt jeder einzelne Rückschlag einem näher ans Ziel.

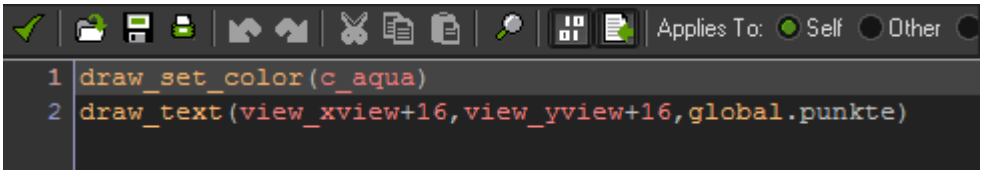


Abbildung 16: Code für die Punkteanzeige

4.5.4. Planeten und Aliens

Zu dieser Zeit war das Spiel noch relativ langweilig, denn es gab nur eine Art von Hindernissen. Dies sollte geändert werden. Somit wurde begonnen eine Vielzahl verschiedener Planeten und sogar Aliens einzuführen. Einige Planeten waren gross, einige klein und manche bewegten sich. Das „coolste“ an den neuen Planeten und Aliens war jedoch, dass manche davon zerstörbar sein sollten. Um dies zu erreichen, musste man zuerst einen Schiessmechanismus im Raumschiff einbauen. Dies funktionierte ziemlich gut. Das Raumschiff gibt bei Drücken der Leertaste einen Schuss ab, der einen Planeten bei Berührung zerstört und verschwinden lässt. Sogar eine Explosion und ein Trümmerfeld wurden einprogrammiert. Jedoch gab es ein Problem. Beim Halten der Leertaste gab das Raumschiff kontinuierlich Schüsse ab, was nicht gewollt war. Darauf wurde eine Art Stoppuhr in den Schussmechanismus eingebaut, der immer nach drücken der Leertaste aktiviert wird. Während die Stoppuhr läuft, wird der Schussmechanismus deaktiviert. Sobald die Zeit abgelaufen ist, also etwa nach einer Sekunde, wird der Mechanismus wieder aktiviert. Der genaue Code ist im Bereich Einführung in den Game Maker unter „Codes als Aktionen“ abgebildet.

Eine andere Möglichkeit an Codes zu kommen, als aus dem Wiki, ist durch Mathematik. Dies kann gut an dem Beispiel der sich bewegenden Planeten veranschaulicht werden. Dieser Planetentyp soll sich im Kreis bewegen. Um den Code für diesen Planten zu schreiben, wurde einfach die Parameterfunktion eines Kreises genommen. Eine Parameterfunktion ist eine Funktion, die die x Koordinate und die Y-Koordinate getrennt berechnet. Man gibt nun in diese Funktion viele Zahlen (t) ein, die von 0 an langsam grösser werden und erhält so eine grosse Anzahl Punkte. Bei genügend kleiner Veränderung in der (t) steigt erhält man eine Linie, die der Funktion entspricht. Diese Linie ist bei uns die Kreisbahn des Planeten. Wenn man nun (t) in jeder Zeiteinheit des Spiels, einem step[22], erhöht und dem Raumschiff den daraus resultierenden Punkt als Position zuweist, folgt das Raumschiff der Kreisbahn. So kommt man mit einfachster Geometrie auf die Lösung des Problems.

```
1
2 winkel = random_range(0, 5);
3 radius = 6;
4 t = round(random(1));
```

```
1
2
3 if t = 0 then
4 {winkel += 0.1;
5 x += cos(winkel) * radius
6 y += sin(winkel) * radius}
7
8 if t = 1 then
9 {winkel -= 0.1;
10 x += cos(winkel) * radius
11 y += sin(winkel) * radius}
```

Abbildung 17: Code für bewegende Asteroiden

[22] 30 Steps entsprechen einer Sekunde

4.5.5. Raumschiff

Nun ging es darum, die Bewegung des Raumschiffes zu optimieren. Es beherrschte die Beschleunigung, das Abbremsen und das sich nach links oder rechts Bewegen. Zu diesem Zeitpunkt konnte das Rumschiff die Geschwindigkeit und die Richtung nicht gleichzeitig verändern. Denn es konnten keine diagonalen Richtungen programmiert werden. Nach reiflicher Überlegung wurde beschlossen, die Bewegung in 2 Richtungen aufzuteilen: horizontal und vertikal. Beide dieser Richtungen haben nun einen Wert, der mit entsprechendem Drücken oder Loslassen der verschiedenen Tasten verändert werden kann. Also statt die Bewegung mit einer direkten Richtung anzugeben, wurden die Bewegungen als Summe zweier Richtungen definiert, um die Geschwindigkeit vom Richtungswechsel zu trennen. Und deshalb konnte das Raumschiff danach diagonal fliegen.

4.5.6. Feinschliff

Das Spiel musste nun nur noch einen Anfang und ein Ende haben. Dies liess sich durch ein selbst gezeichnetes Portal anzeigen.

Um den Einstieg in dieses Spiel möglichst sanft zu gestalten, wurde aus dem ersten Level ein Tutorial[23] Level gemacht. In diesem wird man durch Texte auf die verschiedenen Hindernisse und die Steuerung im Spiel hingewiesen.

Jedoch konnte man das Spiel noch sehr leicht austricksen, da es erlaubt war ohne negative Konsequenzen sich links oder rechts aus dem Spielfeld hinaus zu bewegen. Auf diese Weise konnte man allen Hindernissen ausweichen und einfach am Ende des Levels wieder hineinfliegen. Es gab verschiedene Optionen für das weitere Vorgehen. Das Raumschiff hätte zerstört werden können, wenn man aus dem Raum hinausfliegt. Eine andere Möglichkeit wäre eine Wand am Rande des Raumes gewesen, die nicht passierbar wäre, wie es bereits beim ersten Spielansatz der Fall war. Es wurde jedoch entschieden dem Raumschiff möglich zu machen, auf der rechten, respektive linken Seite hinauszugfliegen und dann auf der jeweils gegenüberliegenden Seite wieder neu zu erscheinen. Das Event wurde so programmiert, dass wenn das Raumschiff zu mehr als der Hälfte nicht mehr auf dem Spielfeld ist, es auf die andere Seite teleportiert wird. Dort erscheint es auch wieder zur Hälfte. Ein fliessender Übergang, das bedeutet, dass das Raumschiff gleichzeitig auf einer Seite verschwindet und auf der anderen Seite erscheint, wurde verhindert. Bei einem solchen Übergang wäre das Raumschiff halbiert, eine Hälfte wäre links am Rande des Raumes und eine rechts. Dies wäre nicht möglich.

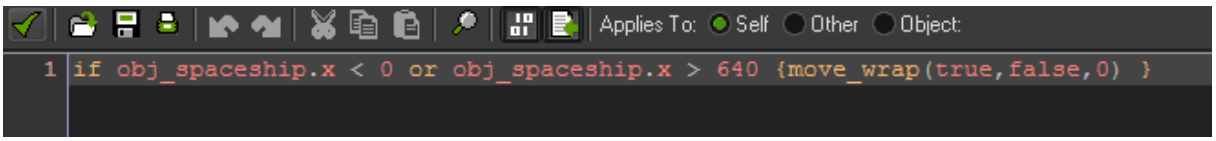


Abbildung 18: Code für Raumschiff ausserhalb

[23] Ein Tutorial Level ist ein Übungslevel, um das Spiel kennen zu lernen

4.6. Level Design

Nun war das grobe Spiel programmiert. Es war aber noch nicht sehr attraktiv gestaltet. Deshalb wurden verschiedene Hintergründe und Musiklieder eingefügt, um eine spannende Atmosphäre zu schaffen.

Bevor die verschiedenen Level zusammenstellt wurden, mussten sie zuerst geplant werden. Durch das Festlegen der Anzahl und der Art der Hindernisse, konnte der Schwierigkeitsgrad festgelegt werden. Die Level sollten weder alle gleich, noch zu einfach, noch zu schwierig sein. Dieser Teil des Spiels ist äusserst anspruchsvoll, denn der Schwierigkeitsgrad hat einen wesentlichen Einfluss darauf, wie lange ein Spieler an einem Spiel interesseiert ist. Ist es zu einfach, hat man es in kurzer Zeit durchgespielt oder man findet es langweilig. Ist das Spiel zu schwierig, ist der Spieler schnell frustriert. Nach sorgfältiger Planung machte ich mich daran, die 10 Level zu gestalten. Nur das erste Level hatte ich als Tutorial bereits gestaltet.

4.7. Testphase

Es ist klar, dass ein Spiel nicht gleich zu Beginn fehlerfrei funktioniert. Deshalb sind das Testen, die Fehlerbehebung und das Verbessern sehr wichtige Bestandteile der Spieleentwicklung. Zuerst soll erklärt werden, wie beim Testen vorgegangen wurde. Um nicht erst am Ende des Spiels das ganz Spiel zu testen, wurde ein kleines Testlevel gemacht, indem die einzelnen Planeten, Menus, Ereignisse etc. getestet wurden. Fast alle programmierten Objekte funktionierten beim ersten Mal nicht wie gewünscht. Somit musste man sich überlegen, ob es ein Programmierungsfehler oder ein Überlegungsfehler war. Programmierungsfehler sind sehr einfach zu beheben, aber schwierig zu finden, da allein schon eine fehlende Klammer oder Buchstabe den ganzen Code unbrauchbar machen kann. Überlegungsfehler dagegen sind schwieriger zu beheben, da man in einigen Fällen eine ganz neue Herangehensweise herausfinden muss. Manchmal muss man sogar feststellen, dass man auf die gewünschte Weise nicht vorgehen kann und seine Idee nicht verwirklichbar ist. Bei diesem Spiel war dies beispielsweise der Fall, als ein Pausenmenu eingebaut werden sollte. Mit den vorhandenen Kenntnissen war es schlicht und einfach nicht möglich dies zu programmieren.

Wenn nun das Spiel soweit fertig programmiert ist, ist der Prozess der Spieleentwicklung noch nicht fertig. Zuerst muss kontrolliert werden, ob alle Level zusammenhängend funktionieren. Um beim Testen nicht das ganze Spiel durchspielen zu müssen, wurde die Level-Freischaltfunktion ausser Kraft gesetzt. Danach wurde das Spiel extern, das heisst ausserhalb vom Game Maker, getestet. Leider stellte sich heraus, dass es sehr massive Fehler im Spiel gab. So war unter anderem das Beenden eines Levels nicht möglich. Zuerst wurden diese extremen Fehler behoben. Danach wurde das Spiel von neuem getestet. Dieses Mal funktionierte alles reibungslos. Einige wenige Makel wurden noch bemerkt, die jedoch nicht den Spielfluss behinderten. Diese wurden zuletzt auch noch beseitigt. Nun musste noch die Level Schwierigkeit getestet werden. Dabei stellte sich heraus, dass die höheren Level nicht auf anhin zu schaffen waren. Schlussendlich konnte jedoch mit Erfolg bestätigt werden, dass alle Level machbar sind und einen „guten“ Schwierigkeitsgrad besitzen.

5. Fazit

5.1. Allgemeine Betrachtung

Das Spiel wurde von einigen Menschen getestet, um ein objektives Bild zu bekommen. Die Meinungen über das Spiel waren sehr gespalten. Einige fanden es ausnahmslos gut, andere einfach nur nervig. Besonders viele Tester waren erstaunt, dass man ein solch ausgearbeitetes Spiel als Unerfahrener entwickeln kann. Der grösste Kritikpunkt des Spiels war die eher langsame Steuerung. Viele Tester hätten sich ein hektischeres Spiel gewünscht. Auch die Musik hat manche gestört, da man die Lautstärke nicht regulieren kann. Einig Tester störten sich an der Breite des Raumschiffes. Wahrscheinlich ist das Verhältnis zwischen Raumschifffflügel und –rumpf nicht optimal.

Ein Spiel ist nie wirklich fertig, da man es immer noch weiter entwickeln könnte. Deshalb möchte ich auf mögliche Erweiterungen des Spiels eingehen.

- Ein Möglicher Zusatz wären mehr Level, die neue Arten von Hindernissen enthalten. Beispielsweise könnte man Aliens einfügen, die auf den Spieler schiessen. Oder man könnte Wurmlöcher einfügen, die den Spieler an einen anderen Ort teleportieren.
- Eine Idee wäre, dass man Level machen würde, die mehrere Etagen besitzen, zwischen denen man hin und her wechseln kann. Der Übergang sollte mit Portale oder Wurmlöcher dargestellt werden.
- Eine andere mögliche Erweiterung wäre ein Boss-Level, wo man gegen ein grosses Gegnermonster kämpfen müsste, um weiter zu kommen.
- Besonders Wünschenswert wäre ein Shop, wo man mit den eingesammelten Münzen sein Raumschiff verbessern könnte, indem man es kleiner und wendiger macht. Ausserdem könnte die Anzahl Schuss pro Sekunde erhöht oder sogar neue Waffen gekauft werden. Dies würde neue, schwierigere Level ermöglichen, die ohne diese Verbesserungen unüberwindbar wären.
- Ein Pausenmenu, wäre eine weiter mögliche Erweiterung. Es könnte Funktionen beinhalten wie „zurück zum Level Select“.
- Ein Optionsmenu wäre auch möglich. In diesem könnte beispielsweise die Lautstärke reguliert werden.
- Von einem Tester wurde gewünscht, dass bei erfolgreichem Abschliessen des letzten Levels, dem Spieler gratuliert wird oder eine Videosequenz gezeigt wird.
- Ein weiterer möglicher Zusatz wäre ein „Easter Egg[24]“ oder Cheats[25].

Es gäbe noch viele weitere mögliche Verbesserungen, jedoch hätten die erwähnten Erweiterungen Priorität.

[24] Ein Easter Egg ist ein versteckter Abschnitt eines Spiels, der bei gewöhnlichem Spielen nicht gefunden wird und eine Besonderheit. Es hat meist keinen bestimmten Nutzen

[25] Ein Cheat ist ein Code, den man im Spiel eingeben kann, der dem Spieler einen Vorteil verschafft

5.2. Persönliche Stellungnahme

Ich bin ziemlich stolz darauf, dass ich ein funktionierendes Spiel entwickeln konnte. Die Maturaarbeit hat nicht immer Freude bereitet und ich würde eine solche Arbeit nicht freiwillig wiederholen. Jedoch hat mir besonders der praktische Teil der Arbeit gefallen. Ich bin positiv überrascht, dass die Arbeit so verlief, wie ich es erwartet hatte. Es gibt jedoch zwei Dinge, die ich anders machen würde, wenn ich eine solche Arbeit nochmals machen würde. Ich würde versuchen schon beim groben Schreiben der Arbeit genauer auf Rechtschreibung, Grammatik und Formulierungen zu achten, um im späteren Verlauf der Arbeit Zeit zu sparen. Ausserdem würde ich schon zu Beginn der Arbeit meinen Betreuer fragen, ob man bestimmte, formale Wünsche beachten muss. Denn ich musste den ganzen Text vom Aktiv in den Passiv übersetzen und somit die ganze Arbeit grundlegend umschreiben.

Die Maturaarbeit hat mir persönlich sehr viel gebracht. Ich habe nicht nur gelernt, wie man eine solch grosse Arbeit bewältigt und umsetzt, ich habe ausserdem Freude am Game Design gefunden. Ich könnte mir sogar vorstellen dies zu studieren. So habe ich mich bereits entschieden ein weiteres Spiel zu programmieren. Dieses soll ein 2D Rollenspiel sein, mit dem im Kapitel Spielidee erwähnten Spielkonzept.

6. Danksagung

Ich möchte allen Menschen, die mich bei dieser Arbeit unterstützt haben, danken. Ohne sie wäre meine Maturaarbeit nicht dieselbe.

Mein Dank gebührt Rainer Steiger, der mich bei dieser Arbeit betreut hat und besonders in den Bereichen Planung und Struktur behilflich war.

Eine grosse Hilfe war Samuel Vonäsch, studierender Game Designer, der mich besonders in der Anfangsphase unterstützt hat. Er bestärkte mich in meinem Entscheid, dieses Maturaarbeitsthema zu wählen und gab mir gute Einsteigertipps im Bereich Spieleentwicklung.

Vielen Dank an alle, die mein Spiel getestet haben. Durch ihre konstruktive Kritik weiss ich nun, wie das Spiel bei anderen ankommt.

Ich möchte meinem Bruder Niculin Detreköy danken, dass er meine Arbeit genau durchgelesen hat. Er half mir viele versteckte Fehler zu finden und die Sätze schöner zu formulieren.

Meine Mutter Karin Detreköy leistete mir während meiner ganzen Arbeit Beistand. Sie motivierte mich zum Arbeiten und half mir beim Überarbeiten der kompletten Arbeit. Ich bin ihr sehr dankbar.

An alle möchte ich an dieser Stelle ein grosses Dankeschön aussprechen.

7. Bibliografie

7.1. Internetquellen

- Internetquelle 1: Wikipedia: ActionScript /Adobe Flash / Flex / AIR
  URL: http://tinyurl.com/onmg375
  URL: http://tinyurl.com/32aaru
  URL: http://tinyurl.com/26khpw3
  URL: http://tinyurl.com/m4f5gm [11.10.2013]

- Internetquelle 2: Geschichte der Spieleentwicklung
  URL: http://tinyurl.com/kn6kkg [20.10.2013]

- Internetquelle 3: Spieleentwicklung von heute – YouTube – Bubble Universe Staffel 3
  URL: http://tinyurl.com/k8h2we7 (fortfolgende) [19./20.10.2013]

- Internetquelle 4: Game Maker – INI Files – YouTube
  URL: http://tinyurl.com/ml8nkt9 [03.11.2013]

7.2. Abbildungsverzeichnis

8. Anhang

8.1. Bildquellen

- Bildquelle 1: Raumschiff
  URL: http://tinyurl.com/pf93czz [06.08.2013]
- Bildquelle 2: Alien
  URL: http://tinyurl.com/oomzwwc [08.08.2013]
- Bildquelle 3: Portal
  URL: http://tinyurl.com/ncwoqkr [08.08.2013]
- Bildquelle 4: Planet 1
  URL: http://tinyurl.com/oxdgzvu [08.08.2013]
- Bildquelle 5: Planet 2
  URL: http://tinyurl.com/o6g7xlq [08.08.2013]
- Bildquelle 6: Planet 3
  URL: http://tinyurl.com/nfx25k3 [08.08.2013]
- Bildquelle 7: Planet 4
  URL: http://tinyurl.com/nn9yunn [08.08.2013]
- Bildquelle 8: Planet 5
  URL: http://tinyurl.com/pyzkesq [08.08.2013]
- Bildquelle 9: Hintergrund Hauptmenu
  URL: http://tinyurl.com/ojmxcov [06.08.2013]
- Bildquelle 10: Hintergrund Level Auswahl
  URL: http://tinyurl.com/q6whvc8 [06.08.2013]
- Bildquelle 11: Hintergrund Level 1
  URL: http://tinyurl.com/nnr62or [08.08.2013]
- Bildquelle 12: Hintergrund Level 2
  URL: http://tinyurl.com/oyqn4e8 [05.10.2013]
- Bildquelle 13: Hintergrund Level 3
  URL: http://tinyurl.com/ogg7pgs [08.08.2013]
- Bildquelle 14: Hintergrund Level 4
  URL: http://tinyurl.com/ppc78g7 [05.10.2013]
- Bildquelle 15: Hintergrund Level 5
  URL: http://tinyurl.com/pfwotjk [08.08.2013]
- Bildquelle 16: Hintergrund Level 6
  URL: http://tinyurl.com/nemrqaa [05.10.2013]

- Bildquelle 17: Hintergrund Level 7
  URL: http://tinyurl.com/p5z2tyk [08.08.2013]
- Bildquelle 18: Hintergrund Level 8
  URL: http://tinyurl.com/oyqn4e8 [05.10.2013]
- Bildquelle 19: Hintergrund Level 9
  URL: http://tinyurl.com/o6y45hb
  [08.08.2013]
- Bildquelle 20: Hintergrund Level 10
  URL: http://tinyurl.com/nt3w8y4
  [05.10.2013]

8.2. Musikquellen

- Musikquelle 1: Hintergrund Level Auswahl
  URL: http://tinyurl.com/o39yqu7 [06.08.2013]
- Musikquelle 2: Hintergrund Level 1
  URL: http://tinyurl.com/odua9ug [06.08.2013]
- Musikquelle 3: Hintergrund Level 2
  URL: http://tinyurl.com/nwlybyb [06.08.2013]
- Musikquelle 4: Hintergrund Level 3
  URL: http://tinyurl.com/njt5h9u [10.10.2013]
- Musikquelle 5: Hintergrund Level 4
  URL: http://tinyurl.com/onfvtwa [10.10.2013]
- Musikquelle 6: Hintergrund Level 5
  URL: http://tinyurl.com/poxvwc8 [10.10.2013]
- Musikquelle 7: Hintergrund Level 6
  URL: http://tinyurl.com/nmuvrjc [05.10.2013]
- Musikquelle 8: Hintergrund Level 7
  URL: http://tinyurl.com/pbky88r [06.08.2013]
- Musikquelle 9: Hintergrund Level 8
  URL: http://tinyurl.com/p758m7q [10.10.2013]
- Musikquelle 10: Hintergrund Level 9
  URL: http://tinyurl.com/ohdjdc2 [05.10.2013]
- Musikquelle 11: Hintergrund Level 10
  URL: http://tinyurl.com/o26c6ms [06.08.2013]