Fakultät Elektrotechnik Feinwerktechnik Informationstechnik
Projekt „ReMasterBlaster“

Prüfungsstudienarbeit von
Maximilian Seyfert
2125266
Bachelor Media Engineering 6. Semester

ReMasterBlaster

Sommersemester 2012

Bestätigung gemäß § 35 (7) RaPO

Maximilian, Seyfert

Ich bestätige, dass ich die Prüfungsstudienarbeit mit dem Titel:

ReMasterBlaster

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die gegebenen Quellen oder Hilfsmittel benützt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 1.08.12

Unterschrift:

“Abstract

Gemäß Paragraph 21, Absatz Eins der Rahmenprüfungsordnung, kurz RaPo, sind Prüfungsstudienarbeiten Prüfungsleistungen mit überwiegend zeichnerischem, gestalterischem oder sonstigem komplexen Inhalt und offenem Lösungsweg zum Nachweis kreativer Fähigkeiten, die sich wegen der umfassenden Aufgabenstellung und der Art der Aufführung in der Regel über einen längeren Zeitraum erstrecken.

Die einzelnen Prüfungsstudienarbeiten des Projektteams, bestehend aus Sebastian Adam, Sergej Bjakow, Michael Kao, Pavlina Pavlova und Maximilian Seyfert, welches sich mit der Realisierung eines Remakes des AMIGA Klassikers MasterBlaster mit neuen Webtechnologien (HTML5, CSS, JavaScript) befasst hat, sollen einen individuellen Einblick auf die jeweiligen Anforderung, die es mit Lösungsstrategien zu bewerkstelligen galt, gewährleisten.

Diese, meine eigene Studienprüfungsarbeit, will Auskunft über die Bedeutung der Rolle meiner Person in der gemeinsamen Teamarbeit verdeutlichen.“

Projekt-Roadmap

Tätigkeit	Dokument	Beteiligte
Erstellung des Lastenhefts	02_documents/02_requirements/ Lastenheft1_1.pdf	Sebastian Adam
Erstellung des Pflichtenhefts	02_documents/02_requirements/ Pflichtenheft_v1_1.pdf	Sebastian Adam
Erstellung des Zeitplans	02_documents/03_timeline/ Remasterblaster.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_01.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_02.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_03.pdf	Sergej Bjakow
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_04.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_05.pdf	Maximilian Seyfert
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_06.pdf	Michael Kao
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_07.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_08.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_09.pdf	Maximilian Seyfert
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_10.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_11.pdf	Sergej Bjakow
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_12.pdf	Sebastian Adam
Erstellung des Protokolls	02_documents/01_protocols/ ...Protokoll_13.pdf	Sebastian Adam
Erstellung der Testbench mit der Sprite – Engine	01_development/01_testbench/*	Pavlina Pavlova
Erstellung der ersten Testumgebung	01_development/02_testenvironment/*	Michael Kao Pavlina Pavlova
Programmierung der Spielelogik	01_development/03_final/game.js	Michael Kao
Erstellung der Sprite Zuordnung	01_development/03_final/sprites.js	Michael Kao
Programmierung des Graphical User Interface	01_development/03_final/gui.js	Pavlina Pavlova
Erstellung der Sprites – Quelldatei	04_material/01_rawfiles/ 01_sprites_psd/*	Michael Kao Sergej Bjakow

Modifikation der Sounds zu mp3 Dateien	01_development/03_final/sounds/*	Sergej Bjakow Maximilian Seyfert
Capturing der einzelnen Elemente (Sprites)	01_development/03_final/img/*	Sergej Bjakow
Zusammenfügen der Sprites	04_material/01_rawfiles/ 01_sprites_ psd/*	Michael Kao
Vorbereiten der einzelnen Szenenelemente	01_development/03_final/img/*	Pavlina Pavlova Sergej Bjakow Michael Kao
Erstellung der Zwischenpräsentation	03_presentations/ 2012-05-09_ReMasterBlaster.pdf	Sebastian Adam
Erstellung der Abschlusspräsentation	03_presentations/ 2012-07-20_ReMasterBlaster.pdf	Sebastian Adam Michael Kao Pavlina Pavlova
Erstellung eines Plakats für die Abschluss-präsentation	04_material/04_poster/ extras_plakat.pdf	Sebastian Adam
Erstellung eines Plakats für die Abschluss-präsentationen	04_material/04_poster/ project_presentation.pdf	Michael Kao
Erstellung des Wikis	http://schorsch.efi.fh-nuernberg.de/mewiki/index.php/Project-BME-2012-07-RemasterBlaster/Project-BME-2012-07-RemasterBlaster	Sebastian Adam
Erstellung der Gesamtdokumentation	02_documents/04_documentation/ 02_overalldocumentation/ Gesamtdokumentation.pdf	Sebastian Adam

Die mit blau markierten Zeilen stellen die Arbeitsbereiche mit eigener Arbeitsleistung dar.

Arbeitsleistungen wurden jedoch auch in anderen Gebieten erbracht, die aber aufgrund ihrer unzureichenden Größe nicht in der Tabelle berücksichtigt wurden.

Inhaltsverzeichnis:

1. Ausgangssituation:

Wir, Sebastian Adam, Sergej Biakov, Michael Kao, Pavlina Pavlova und ich, haben uns entschieden den AMIGA – Klassiker Masterblaster von Alexander Ivanof, der 1994 Erschienen ist im Webbrowser als Neuauflage lauffähig zu machen. Dazu wollten wir die aktuellen Web – Technologien HTML 5, CSS und JavaScript verwenden. Da es im Internet kaum brauchbares Bild oder Video – Material zu besagtem Klassiker gibt, trafen wir uns am 3. April zum Projektmeeting, wo Prof. Dr. Hopf uns seinen eigenen AMIGA inklusive dem Spiel mitbrachte. Wir spielten ein paar Partien um einen Eindruck von Masterblaster selber zu bekommen und dessen Gameplay feeling hautnah zu erleben. Außerdem erklärte uns Prof. Dr. Hopf die Spielmechanik und die im Spiel vorkommenden Goodies. Ausserdem erhielten wir noch ein Rom des Orginals. Dieses lässt sich in einem Emulator ausführen wie unter anderen der UAE Amiga Emulator.

Nun zur Erkärung des Orginals:

Master Blaster ist ein multiplayer Spiel und unterstützt 1-5 Spieler mittels Tastatur und Joysticks. Das Hauptziel des Spiels ist es, der letzte Überlebende in einer Runde zu sein. Eine Runde wird auf einem Komplett auf dem Bildschirm zu sehenden Spielfeld ausgetragen. Die Spieler starten in den Ecken, mit Ausnahme das der fünfte in der Mitte startet. Die Spieler sind zu beginn des Spiels durch Wände getrennt, die sie im laufe der Runde durch legen einer Bombe weg sprengen können. Hinter den Wänden können sich sog. Goodies verbergen, die dem Spieler unterschiedliche Hilfestellungen geben. Außerdem kann es auch passieren, dass sich Geldmünzen dahinter verbergen, für die sich der Spieler in einem Shop Goodies für die nächste Runde Kaufen kann. Die anderen Spieler können auch mit Hilfe der Bomben Gesprengt werden und sind somit tot. Das Spiel ist beendet, wenn ein Spieler die zuvor eingestellte Anzahl an Runden gewonnen hat.

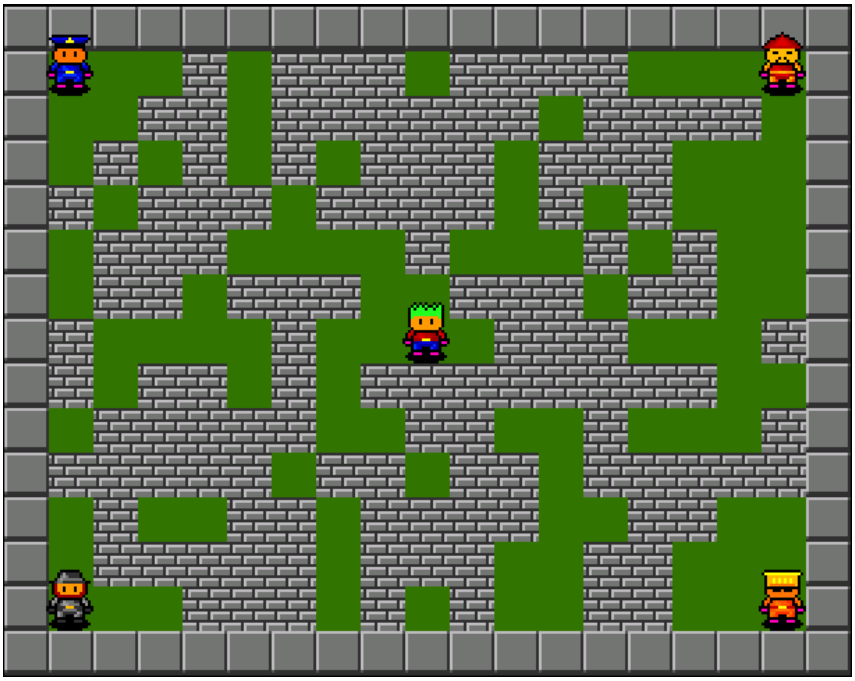

Abbildung 1: Screenshot des Master Blaster Orginals

2. Projektorganisation:

Zur Projektorganisation wurden wöchentliche Projektmeetings und Besprechungen gehalten. Ab und zu trafen sich auch Personen aus der Projektgruppe, die zu zweit oder zu tritt eine Aufgabe zu bewältigen hatten. Für kurzfristige Absprachen erstellten wir uns eine geschlossene Facebook Gruppe. Außerdem wurde die Goggle Calendar Web-App zur Verwaltung der Termine benutzt. Zur Projektplanung und dem Ressourcenmanagement verwendeten wir GanttProject, eine in Java geschriebene Anwendung, die unter der GPL Lizenz frei verfügbar ist.

Unser Projekt setzten wir in einem öffentlich verfügbaren, Github repository um. Github basiert, wie der Name schon erahnen lässt auf dem Content Management- und Versionsverwaltungs-System Git. Es wurde ursprünglich für die Verwaltung des Linux-Kernels entwickelt. Git ist heut zu Tage erste Wahl, wenn es um das verwalten von Software-Projekten mit mehreren Entwicklern geht und verdrängt somit stetig Subversioning.

Dieser Umstand rührt vor allem daher das Git sehr viele Funktionen und Eigenschaften mitbringt, die so kein anderes dieser Systeme umsetzt.
Ein paar dieser Eigenschaften sind zum Beispiel das jeder Benutzer eine lokale Kopie des gesamten Repositorys, inklusive der History besitzt. Dadurch können die meisten Aktionen lokal und ohne Netzwerkzugriff zu benötigen ausgeführt werden. Ein weiterer Bonus von Git ist die Möglichkeit der nicht linearen Entwicklung. Es können unterschiedliche Branches erstellt werden, die dann durch das sog. Merging zusammengeführt werden können.

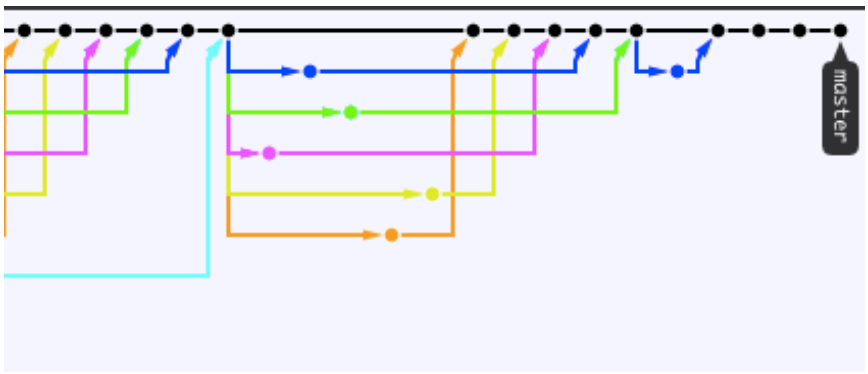

Abbildung 2: Branches Graph von Github

3. Sprite Engine:

Um uns die Arbeit etwas leichter zu machen, begaben wir uns zu Beginn unseres Projektes auf die suche nach einer Sprite Engine. Eine Sprite Engine ist eine Bibliothek, die vorgefertigte Funktionen bereit stellt, die es den Entwicklern ermöglicht Sprites einfacher in Spielen zu verwenden. Unsere Kriterien an die Sprite Engine waren, das sie HTML5/JavaScript kompatible sein sollte, möglichst performant, um zu gewährleisten dass, das Endprodukt auf möglichst vielen Systemen lauffähig ist. Außerdem musste die Sprite Engine, aufgrund der Vorgaben durch das Original Master Blaster eine Möglichkeit bieten, die Sprites übereinander zu legen was so viel heißt wie, sie muss einen parameterisierbaren Z-Wert für Sprites besitzen. Was für uns auch noch sehr wichtig war, das es für die Engine eine gute Dokumentation so wie vorhandene Tutorials gibt. Aufgrund der Kriterien grenzten wir die

Auswahl auf ein Paar wenige ein. Um den besten Kandidaten für unser Projekt zu finden schrieben wir eine Performance Test, dessen Auswertung uns gleichzeitig die Target Plattform lieferte, in unserem Fall der Browser Google Chrome, so wie die Sprite Engine Crafty. Die neben oben genannten Kriterien auch noch zusätzliche Features liefert, wie die Einfache Möglichkeit Spielelemente ohne Vererbung zu organisieren. Sie unterstützt sowohl Canvas wie auch DOM. Außerdem stellt sie Custom Events zur Verfügung, die an jeder Stelle zu jedem Zeitpunkt ausgelöst werden können. Die Community von Crafty ist sehr aktiv und das Forum wird ständig durch neue Fragen und Antworten bereichert.

Abbildung 3: Testbench zu Performancezwecken

4. Reverse Engineering

Wenn man, wie bei Unserem Projekt ReMasterBlaster versucht einen Nachbau einer bereits vorhandenen Software zu erstellen, der möglichst originalgetreu sein soll kommt man nicht um das so genannte „Reverse Engineering“ herum. Der Begriff wird auf Deutsch auch als „Nachkonstruktion“ bezeichnet und meint ein bereits bestehendes System zu Analysieren und bestimmte Konstruktionselemente zu extrahieren. Außerdem analysiert man bestimmte Verhaltensweisen die das Original System aufweist, um dann eine möglichst eins zu eins Kopie dessen zu bekommen.

In Unserem Fall verwendeten wir das uns zu Verfügung gestellte Rom-Image und den UAE Amiga Emulator in der Windows Version.

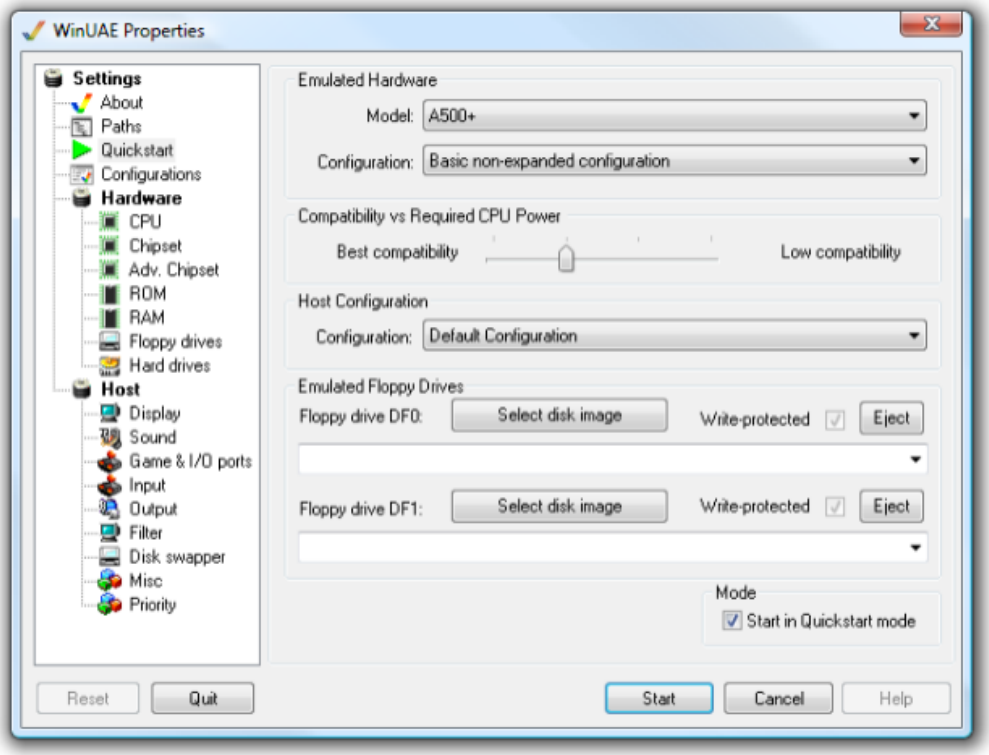

Abbildung 4: WinUAE Property Pane

Dieser bietet wie oben auf der Abbildung vier zu sehen schon Funktionen um Daten von der aktuell geladenen Rom zu extrahieren. Dies kann als Video, Audio oder Bild File gespeichert werden. Um die Sounds zu Extrahieren nahmen wir verschiedene Videos auf und suchten uns die Teile heraus die wir benötigten. Anschließend nahmen wir die Files und bearbeiteten sie mit

dem Open Source Programm Audacity, um wirklich exakt den Audioschnitt hinzubekommen. Ein weiteres Problem entstand dann bei der Exportierung der Dateien, weil nicht jeder Browser jeden Audio Standard unterstützt.

Browser	Ogg Vorbis	MP3	WAV
FireFox 3.6+	✓		✓
Safari 5+		✓	✓
Chrome 6	✓	✓	
Opera 10.5+	✓		✓
Internet Explorer 9 (beta)		✓	✓

Abbildung 5: Browser Audio Support

Wir entschieden uns dann dazu die Dateien im MP3 Format zu exportieren, da wie auf Abbildung 5 zu sehen dieses von unserem Target Browser Google Chrome unterstützt wird. Eine andere Fragestellung offenbarte sich uns wie wir die Hintergrundmusik extrahieren sollten. Die Hintergrundmusik im Original ist ein immer schneller werdender Loop des immer gleichen Sound-Intervalls. Nun währe hier die eleganteste Lösung gewesen die Sounddatei wie im Original mit steigender Sample Rate zu loopen. Dies gelang uns allerdings nicht, da es uns mit den zu Verfügung gestellten Werkzeugen nicht ohne viel aufwand gelang den Loop ohne kurze Pause laufen zu lassen. Folglich mussten wir die Ganze Hintergrundmusik verwenden, was leider wesentlich mehr Speicherplatz und Bandbreite benötigt.

Die Einzelnen Sprites wurden durch Screenshots extrahiert. In Photoshop dann wurden die gewünschten ausgeschnitten und in eine Sprite Datei gespeichert. Auf diese Sprite Datei kann dann die Sprite Engine zugreifen und sich die gerade benötigten Teile herausnehmen. Dies funktioniert deshalb, weil in der Sprite Datei die einzelnen Sprites immer gleich angeordnet und in selber Größe vorhanden sind.

Abbildung 6: Sprites, die für das Spiel erstellt wurden

Außer den Audio und Video Files mussten auch noch die genauen Zeiten für Aktionen bestimmt werden. Zum Beispiel, wie lang dauert es bis eine Bombe explodiert oder wie schnell bewegen sich die Charaktere auf dem Spielfeld. Für die Menüs und den Anfangsbildschirm extrahierten wir nicht die einzelnen Buchstaben sondern sind im Internet auf eine Schriftart gestoßen, die der im Spiel verwendeten sehr nahe kommt. Sie nennt sich Amiga4ever und ist frei verfügbar. Dazu wurden wiederum Videos mit dem UAE-Emulator extrahiert, die dann von uns untersucht wurden und dadurch die Zeiten daraus ermittelt wurden.

Um ein möglichst originalgetreues Spielgefühl beim Nachbau zu erzeugen ist es auch sehr wichtig gewisse Umstände zu hinterfragen. Einer dieser Umstände war zum Beispiel der genaue Bewegungsablauf der Charaktere und wie sie bei Tastendruck genau reagieren. Dazu mehr im nächsten Inhaltspunkt.

5. Coding:

Da wir uns bei unserem Projekt auf die Technologien HTML, CSS, JavaScritpt beschränkten, mussten wir uns auf Grund mangelnder Kenntnis dieser erst einmal Einarbeiten. Dies geschah meinerseits mit Hilfe von im Web verfügbarer Tutorials und Youtube Videos. Außerdem war die Vorlesung von Prof. Dr. Matthias Hopf eine große Hilfe auf diesem Gebiet. Nun begaben wir uns daran die Crafty Sprite Engine zum laufen zu bekommen. Dies taten wir anfangs mit den auf der Crafty Seite verfügbaren Tutorials. In die Einarbeitung in Crafty verwendete ich außerdem noch folgendes Tutorial: http://cykod.github.com/Presentations/HTML5/Crafty/

Zuerst braucht man ein HTML Dokument mit den üblichen Tags wie <html> <head> <body>, dann verknüpft man diese Seite mit der Crafty.js Datei.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <script src="crafty.js" type="text/javascript"></script>
5          <script src="game.js" type="text/javascript"></script>
6      </head>
7      <body>
8      </body>
9  </html>
```

Abbildung 7: Einfache Game Startseite

Zusätzlich wurde noch die Java Script Bibliothek Jquery verwendet, die wir in diesem Semester bei Prof. Dr. Hopf kennengelernt haben und auf die gleiche Art wie crafty.js importiert wird. Im game.js wird dann der eigentliche Game Code geschrieben. Zuerst muss man Crafty starten dies geschied mit der Funktion crafty.init(608, 480); Nun beginnt man mit dem Laden der Sprites, was mit der Funktion loadSprites(„Sprite-Name“); passiert, die sich bei uns in einem extra JavaScript File wiederfindet.

Hier wird durch Crafty.sprite die Größe des Sprites festgelegt(In ReMasterBlaster 32 Pixel bei nicht Player Sprites) und dann das entsprechende File spezifiziert aus dem man sich die Sprites holt. Dann werden Crafty die Positionen der für das jeweilige Teil aus der Sprite Datei mitgeteilt. Dies kann folgendermaßen aussehen:

```
Crafty.sprite(32, "img/sprites.png", {
    wall: [0, 0],
    brick: [0, 1],
    brick_cracked_1: [1, 1],
    brick_cracked_2: [2, 1],
    bomb: [0, 2],
    fire: [0, 3],
    burning_brick: [0, 4],
    speed_up: [0, 5],
    bombs_up: [1, 5],
    fire_up: [2, 5],
    time_fuze: [3, 5],
    disease: [4, 5],
    invincible: [0, 6],
    death_skull: [0, 7],
    wall_appear: [0, 8],
    money: [1, 7],
    trophy: [2, 7],
    empty: [0, 8]
});
```

Abbildung 8: Festlegung der Environment Sprites

Die Audio-Files habe ich verschiedenen Variablen zugeordnet. Was folgendermaßen aussehen kann.

```
var bgmusic = new Audio("sounds/bgmusic.mp3");
var exp = new Audio("sounds/explode.mp3");
var scream = new Audio("sounds/scream.mp3");
var itemSpeedUp = new Audio("sounds/go.mp3");
var bingo = new Audio("sounds/bingo.mp3");
var ohlala = new Audio("sounds/ohlala.mp3");
var warp = new Audio("sounds/warp.mp3");
var cash = new Audio("sounds/cash.mp3");
var alarm = new Audio("sounds/alarmloop.mp3");
```

Abbildung 9: Initialiseren der Audiofiles

Abgespielt werden die Dateien dann mit der Funktion variable-name.play();

Nun haben wir alle externen Daten zusammen, die wir während der Programmierung der Spielelogik brauchen.

Nun legen wir mit Hilfe von Arrays die Position der Blöcke fest.

Ein weiteres Problem was sich durch den möglichst originalgetreuen Nachbau zeigte, war das Bewegungsverhalten der Charaktere, die sich im Original wie auf einem unsichtbaren Gitternetz bewegen und bei Wechsel der Position und gleichzeitig Wechsel der Orientierung (Vertikal/Horiziontal) wieder über die Ecken des anliegenden Blocks auf die Gitternetzstruktur zurückbewegt. Micha und ich versuchten hier eine Lösung zu finden, die dem Original am nächsten kommt. Dazu versuchten wir ein Gitternetz zu realisieren, das genau in der Mitte der Blöcke verläuft. Unsere Version sollte per Modulo ermitteln an welchem Gitterteil sich der Charakter am nächsten befindet. Und ihn durch eine If-Else Condition wieder zurück auf das unsichtbare Gitternetz zu lenken.

Reflexion über die Eigene Tätigkeit innerhalb des Projektes.

Ich habe mich währen des kompletten Projektzeitraumes mit viel Engagement, Freude und Interesse um meine Tätigkeiten in dem Projekt ReMasterBlaster gekümmert. Außerdem hab ich viel über das mir bis Dato neue Verfahren des Re-engineering gelernt. Im Zuge dessen hab ich im Internet gelesen, das dieses Verfahren, in unterschiedlichen Formen sehr häufig in der Softwareindustrie verwendet wird, und somit ein sehr gebräuchliches Werkzeug der aktuellen Software Entwicklung ist. Meine in diesem Semester bei Prof. Dr. Matthias Hopf erlernten Client seitigen Internetprogrammierungskenntnisse konnte ich in unserem Projekt erfolgreich optimieren. Diese umfassen vor allem HTML,CSS und JavaScript auch die Verwendung von jQuery fällt mir nun leichter. Bei der Konzeption der Bewegungsabläufe frischte ich meine Mathematik Kenntnisse aus den vorherigen Semestern auf. Natürlich ist in jeder Gruppe, die gemeinsam an einem Projekt arbeitet auch der soziale Aspekt von großer Wichtigkeit. Ich gehe davon aus das mir die Projektarbeit auf diesem gebiet auch wichtige Erfahrungen zugetragen hat, die ich in Zukunft sicher zu nutzen weis.

Für die Zukunft ist zu sagen, dass ich viel gelernt habe und hoffe meine erworbenen Kenntnisse später einmal einsetzen zu können.

6. Anhang:

Muss - Kriterien	• Spielbares Produkt • 2 Steuerbare Spieler mit Tastatur • Bomben legen und Wände sprengen • Gegenspieler bei Treffer entfernen	
Soll - Kriterien	• Einbindung der Extras (Goodies) • Konfigurationsmenü • Szenenabfolge • Hall of Fame	
Kann - Kriterien	• **Alle** Goodies implementieren	
Kann - Kriterien:	• mehrere Spieler • zusätzliche Eingabegeräte • Shop für Extras • Sprite-Auswahl für die Spieler