



GEORG-SIMON-OHM
HOCHSCHULE NÜRNBERG

Fakultät Elektrotechnik Feinwerktechnik Informationstechnik
Projekt „ReMasterBlaster“

Prüfungsstudienarbeit von
Pavlina Pavlova
Matr. – Nr.: 2192523
Bachelor Media Engineering 6. Semester

Prüfungsstudienarbeit
für das Projekt „ReMasterBlaster“
im Rahmen der Projektarbeit im 6. Semester
des Studiengangs Media Engineering



Sommersemester 2012

Bestätigung gemäß § 35 (7) RaPO

Pavlova, Pavlina

Ich bestätige, dass ich die Prüfungsstudienarbeit mit dem Titel:

Prüfungsstudienarbeit
für das Projekt „ReMasterBlaster“
im Rahmen der Projektarbeit im 6. Semester
des Studiengangs Media Engineering

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

Datum: 03.08.12

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

Erstellung des Lastenhefts	02_Dokumente/04_Lasten- und Pflichtenheft/Lastenheft_v1_1	Sebastian Adam
Erstellung des Pflichtenhefts	02_Dokumente/04_Lasten- und Pflichtenheft/Pflichtenheft_v1_1	Sebastian Adam
Erstellung des Zeitplans	02_Dokumente/03_Projektplan/Remasterblaster.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_01.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_02.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_03.pdf	Sergej Bjakow
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_04.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_05.pdf	Maximilian Seyfert
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_06.pdf	Michael Kao
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_07.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_08.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_09.pdf	Maximilian Seyfert
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_10.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_11.pdf	Sergej Bjakow
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_12.pdf	Sebastian Adam
Erstellung des Protokolls	02_Dokumente/01_Protokolle/...Protokoll_13.pdf	Sebastian Adam
Erstellung der Testbench mit der Sprite – Engine	04_Quellcode/TestBench/*	Pavlina Pavlova
Erstellung der ersten Testumgebung	04_Quellcode/TestEnvironment/	Michael Kao
Programmierung der Spielelogik	04_Quellcode/finalGame/game.js	Michael Kao
Erstellung der Sprite Zuordnung	04_Quellcode/finalGame/sprites.js	Michael Kao
Programmierung des Graphical User Interface	04_Quellcode/finalGame/gui.js	Pavlina Pavlova
Erstellung der Sprites – Quelldatei	04_Quellcode/finalGame/sprite_players.psd	Michael Kao Sergej Bjakow
Modifikation der Sounds	04_Quellcode/finalGame/sounds/*	Sergej Bjakow

zu mp3 Dateien		Maximilian Seyfert
Capturing der einzelnen Elemente (Sprites)	05_Materialien/Sprites/*	Sergej Bjakow
Zusammenfügen der Sprites	04_Quellcode/finalGame/sprites.psd	Michael Kao
Vorbereiten der einzelnen Szenenelemente	04_Quellcode/finalGame/img/*	Pavlina Pavlova Sergej Bjakow Michael Kao
Erstellung der Zwischenpräsentation	02_Dokumente/05_Präsentationen/2012_05_09_ReMasterBlaster.pdf	Sebastian Adam
Erstellung der Abschlusspräsentation	02_Dokumente/05_Präsentationen/2012_07_20_ReMasterBlaster.pdf	Sebastian Adam Michael Kao Pavlina Pavlova
Erstellung eines Plakats für die Abschlusspräsentation	05_Materialien/Extras_Platat.pdf	Sebastian Adam
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	03_Abschluss/Gesamtdokumentation	Sebastian Adam

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

Inhalt

BESTÄTIGUNG GEMÄß § 35 (7) RAPO	2
ABSTRACT	3
PROJEKT-ROADMAP	4
GRUPPENFINDUNG	7
ENGINE AUSWAHL	7
ERSTE TESTS	8
GIT HUB.....	9
ROLLEN UND AUFGABENVERTEILUNG	9
ENTWICKLUNGSPHASE	9
Konfiguration	9
GUI erstellen	10
PARAMETRISIEREN UND FINAL TOUCHES.....	13
REFLEKTION:	14
SOLL UND IST ZUSTAND:.....	15

Gruppenfindung

Am Anfang des Semesters hatte ich keine Projektarbeit-Gruppe. Bei der Einführungsveranstaltung von BME4 habe ich Herrn Hopf gefragt, wie die Projektvergabe abläuft und wie die Gruppen zustande kommen. Er hat mir erklärt, dass sich die Gruppen langsam schon zusammengefunden haben, aber dass er noch ein Thema hätte, das für vier bis fünf Leute geeignet ist. Da ich leider fast keinen aus dem 6ten Semester kannte, habe ich die Idee aufgegeben eine Projektarbeit in dem Semester zu machen. Bis ich von Sergej Bjakow angesprochen wurde, ob ich mit ihm und Maximilian Seyfert in einer Projektgruppe sein will, da sie noch Leute suchen. Es stand also fest! Wir haben Herrn Hopf kontaktiert und er hat uns ein bisschen mehr Information über das "Remake des Spiels MasterBlaster" erzählt. Da die Projektgruppen aus mindestens vier Leute bestehen müssen, war unser Hauptziel noch weitere Person zu finden.

Bei der ersten Besprechung mit allen Professoren, in dem auch offiziell die Themen verteilt wurden sind Michael Kao und Sebastian Adam zu der Gruppe gekommen, die sich bereits seit dem Ende des vorigen Semester für Thema interessierten. Sebastian hat vorgeschlagen die Projektleitung zu übernehmen, und da niemand etwas einzuwenden hatte - stand unsere kleine Gruppe fest.

Engine Auswahl

Nachdem jeder von uns sich das Spiel auf einem Emulator angeschaut hatte, war es an der Zeit, dass wir mit der Planung anfangen. Schon vor der ersten Vorbesprechung haben wir von Prof. Hopf eine Tabelleⁱ mit einer Übersicht sämtlicher JavaScript Game-Engines bekommen.

Ein "Remake von MasterBlaster mit aktuellen Technologien" bedeutete für uns, dass wir eine passende Spiele-Engine brauchen. Herr Hopf hat uns drei Kriterien genannt, auf die wir achten sollten:

- eine Sprite Engine, um Spielgrafik anzuzeigen.
- schnelle Animation, damit viele Animationen gleichzeitig ohne Ruckeln ablaufen können (dass es möglich ist einen "Teppich aus Bomben, die gleichzeitig explodieren, zu legen")
- z-Tiefenwert unterstützt.

Uns, als Gruppe, war es wichtig, dass die Engine aktuell, gut dokumentiert und vollständig implementiert ist.

Eine meiner Aufgaben war es, diese Game-Engines anzuschauen und eine engere Auswahl zu treffen, welche für uns in Frage kommen.

Aus über 50 Game-Engines habe ich eine persönliche Rangliste von 7 Favoriten erstellt. Es ist ziemlich schwierig, 50 Engines gleich gut zu überprüfen, wenn man sich nur wenig auskennt, aber ich habe da eine sehr einfache Regel befolgt - "dress to impress". Wenn die Webseite einer Engine nicht "seriös" genug wirkte, wurde sie durchgestrichen. Dies hat mir eine Menge Ärger und Zeit gespart, weil so sind solche Engines weggefallen, die nicht mehr unterstützt und gepflegt werden, auch solche die nicht vollständig implementiert sind.

Als diese Rangliste stand, war meine Aufgabe eine Testbench für unsere Favoriten zu schreiben, damit wir die Schnelligkeit der Animation testen konnten. Als Favorit aller Teammitglieder stellte sich die Engine Crafty heraus.

Die Testbench bestand aus einem Feld mit 25x21 Blöcken. Jeder Block enthielt eine Figur, bestehend aus 16 Pixel, die sich in einer Endlosschleife drehte.

Erste Tests

Als die Testbench das bestätigt hatte, was wir uns erhofft hatten - dass Crafty die richtige Engine für uns ist, war es meine Aufgabe die erste Spielumgebung damit aufzubauen.

Da hat mir die ausführliche Dokumentation auf der Crafty Webseiteⁱⁱ sehr geholfen. Die gut dokumentierte Seite und ihre Tutorials waren eine der Hauptgründen, wieso uns diese Engine so gut gefallen hat. Die einfachen Tutorials haben mir bei der Erstellung der Umgebung geholfen, so dass nach kurzer Zeit die erste bewegbare Figur in einem 15x19 Feld stand. Alles wurde mit Crafty-eigenen Funktionen realisiert, die die Steuerung und Animation erleichtern.

Git Hub

Da wir unser Projekt "ReMasterBlaster" unter der GPL Lizenz veröffentlichen wollten, war es sehr wichtig, dass unser Code online frei verfügbar ist. Damit wir alle gleichzeitig und auch versioniert arbeiten können, haben wir uns für das git-basierte Online-System GitHubⁱⁱⁱ entschieden.

Was für mich bedeutete, dass ich mich intensiv mit dem System befassen musste, da ich dieses noch nicht kannte. Anfangs dachte ich, dass ich Schwierigkeiten haben werde, aber mit der Hilfe von Prof. Hopf und den Git-Tutorials war der Einstieg einfach und übersichtlich.

Rollen und Aufgabenverteilung

Schon nach den ersten paar Projekttreffen und nachdem ich die Testbench erstellt hatte, war es für mich klar, dass ich mich fast ausschließlich mit dem Programmieren beschäftigen möchte, was in dem Plan von Sebastian ziemlich gut gepasst hat. Meine Hauptaufgaben waren Konfigurationsmöglichkeiten für unser Spiel zu erstellen, so zu sagen die Architektur des Spiels schaffen, und die GUI zu erstellen in der das Spiel laufen soll.

Entwicklungsphase

Konfiguration

Eine wichtige Aufgabe, die ich hatte, war die Konfigurationsmöglichkeiten des Spiels zu untersuchen und eine passende Architektur zu entwickeln. Nutzer aller Arten von Videospielen sind es gewöhnt, dass man seine persönlichen Präferenzen speichern kann, damit man sie nicht jedes Mal erneut eingeben muss. Auch ist es bei vielen Spielen möglich, eigenen Content zu dem Spiel hinzuzufügen, wie zum Beispiel eigene Spielfiguren. Da unser Spiel in einem Browser laufen sollte, wäre das Speichern der Konfiguration in Form von Cookies und das laden von externe Sprite-Dateien möglich.

Meine Aufgabe bestand darin mich zu informieren, wie wir unser Programm aufbauen müssen, damit später eine Konfiguration via Cookies und externe Konfigurationsdateien möglich sind. Es hat sich schnell herausgestellt, dass sich JSON-Objekte gut dafür eignen.

Die JSON Syntax war ziemlich schnell verstanden. Eine der größeren Schwierigkeiten, die ich hatte, bestand darin die Funktionen, die jQuery liefert und jQuery selber, zu verstehen, da wir in der Vorlesung Multimedia Applikationen noch nicht soweit waren.

So entstand die Basis-Konfigurationsvariable *config*, die als JSON-Objekt aufgebaut ist. Als dieses Konzept stand, war ziemlich deutlich wie der Rest der Architektur aufzubauen ist. Es bestand eine index.html, aus der alle anderen Funktionen aufgerufen wurden, GUI und Spiellogik als externe Dateien.

GUI erstellen

Meine Hauptaufgabe war die GUI mit allen Szenen und Abläufen so zu definieren, dass die Spiellogik mit allen nötigen Parameter aufgerufen werden kann.

Die GUI besteht aus Szenen, die per Tastendruck der Reihe nach gewechselt werden. Man kann Spieler- und Gewinnanzahl konfigurieren.

Ich habe versucht, mich an den Originalszenen zu halten, habe aber ein paar Veränderungen und Anpassungen vorgenommen. Wir haben eine Schriftart gefunden, die der original Amigaschriftart ähnelte, die ich in allen Szenen, um das Feeling beizubehalten, angebunden habe. Anbei einen Überblick aller Szenen mit kurzer Beschreibung.

Startscreen - es wird das Logo des Spiels angezeigt. In unseren Fall das Logo, das Michael für uns entwickelt hat.

Credits - ähnlich wie im Originalspiel, habe ich die Credits so konstruiert, dass sie unsere Namen, das Originalspiel und ein kleines "Danke schön" an den Schöpfer des Spiels enthalten.

Konfigurationsmenü - Das ist das Hauptkonfigurationsmenü. Hier kann man einstellen wie viele Spieler spielen werden und wie viele Gewinne benötigt werden um Sieger zu sein.

Um dem Originalspiel treu zu bleiben habe ich alle Konfigurationsmöglichkeiten beibehalten, es sind aber momentan nur einige veränderbar.

Mit den Tasten nach oben und unten wechselt man zwischen den unterschiedlichen Konfigurationsmöglichkeiten und mit den Tasten für links und rechts wird der Wert erniedrigt oder erhöht.

Ich habe eine Verbesserung des Konfigurationsmenüs vorgenommen. Ich fand dass es eine gute Idee wäre zwischen Maximal- und Minimalwert nur mit einem Tastendruck zu wechseln, so dass wenn man die maximale Anzahl an Spieler (z.B.) erreicht hat, wieder mit nur einem Tastendruck nach rechts die minimale Spieleranzahl erreicht. Dies habe ich

eingebaut, da man aus anderen Spielen gewöhnt ist einen solchen Verhalten zu erwarten und das wiederholte Drücken einer Taste ohne Rückmeldung als störend empfunden werden kann.

Für die Werte YES/NO, ON/OFF ist ein Toggle eingebaut, so dass auch diese einfacher zu ändern sind.

Spiellerauswahl - Gleich nach der Konfiguration kommt die Möglichkeit für jeden Spieler eine Figur auszuwählen, ohne dass die selbe Figur ein zweites Mal ausgewählt werden kann.

Da es ursprünglich geplant war nur zwei Spieler zu implementieren, dachten wir uns dass es etwas langweilig wäre, wenn man immer mit den gleichen zwei Spielerfiguren spielt. Meine Idee war eine Art Auswahlmenü einzubauen. Dieses kleine Menü war nicht geplant und stand nicht als Ziel bei den Kann-Kriterien, sondern ist komplett als "Goodie" gedacht. Mit ein paar Anpassungen der Variable zum Speichern von Spielerinformationen, war dies nun auch erledigt.

Ab jetzt ist jeder Spieler nicht nur per Spielernummer identifizierbar, sondern hat auch einen Namen - den Namen seiner Spielfigur.

All diese Szenen werden nur einmal am Anfang, wenn das Spiel gestartet wird, ausgeführt. Alle folgenden Szenen werden wiederholt aufgerufen, so oft bis einer der Spieler die benötigte Anzahl an Siege erreicht hat.

Shop - Wenn ein Spieler, während des Spiels Münzen aufgesammelt hat, erscheint nach der Spielrunde der Shop, in dem der Spieler einkaufen darf. Ein Spieler darf solange er Geld hat einkaufen, oder bis er auf "Exit" gedrückt hat.


Ich habe den Shop bei der Implementierung etwas erweitert und angepasst. Der Button EXIT wird farblich markiert, was eine bessere Übersicht darstellt und es ist zu jedem Zeitpunkt sichtbar welcher Spieler grade den Shop betreten hat - im Originalspiel stand nur eine Nummer, bei uns ist jeder Spieler durch seinen Namen identifizierbar.

Countdown - nach dem alle Spieler ihre Shop-Auswahl bestätigt haben startet ein kleiner Countdown - 3,2,1 - und das Spiel kann beginnen!


Spiel - hier wird die Funktion `startgame()` von `game.js` aufgerufen, in der das Spiel stattfindet.

Hall of Fame - nach dem das Spiel zu Ende ist, wird ein Objekt zurückgegeben, welches alle Spielerinformationen enthält - die Gewinne und das gesammelte Geld. Die "Halle der

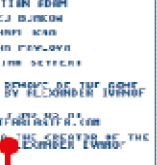
Startscreen:



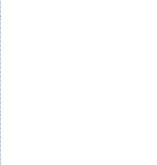
Credits:




Konfiguration:




Spielauswahl:




Countdown:




Shop:



Hall of Fame:



Spiel:



wenn Anzahl nötiger Gewinne erreicht

wenn Anzahl nötiger Gewinne noch nicht erreicht

Die Spiellogik von Michael und meine GUI haben unterschiedliche Crafty Versionen verwendet, was dazu geführt hat, dass alle Key-Anbindungen, Crafty-Animationen usw. nicht dargestellt wurden.

Ursprünglich hatten wir uns darauf geeinigt, dass wir soweit wie möglich alles mit Crafty implementieren. Dies hat zu Problemen geführt. Da die Engine eine Sprite Engine ist und darauf ausgelegt ist, dass alle Animationen und dynamische Inhalte in einer Main-Szene passieren, so dass die Szene nicht neu aufgebaut werden soll.

Pavlina Pavlova

zu können, müsste bei jeder Änderung die Szene neu gezeichnet werden. Crafty implementiert seine Szenen intern auf eine aufwändige Art und Weise, was dazu geführt hat, dass der Übergang nicht flüssig war und es sichtbar war, wenn eine Szene neu geladen wird.

Deswegen war es naheliegend, dass wir Crafty nur für die Funktionen, die von der Engine vorgesehen sind und für alles andere reine JavaScript oder jQuery-Funktionen nehmen.

Dies bedeutete für mich, dass ich meinen Code erneut umschreiben musste und alle Szenen von Crafty nach JS Canvas umziehen.

Als die Verbindung zwischen GUI und Spiel stand, war es einfach den kompletten Ablauf zu testen. Erst dann sind einige Logikfehler aufgetreten. Die GUI übergibt der Spiellogik eine Objektvariable und bekommt als Rückgabe auch ein Objekt zurück. Das bedeutet, dass immer in die gleiche Variable geschrieben wurde, da in JavaScript Objekte und Arrays per Referenz und nicht per Value übergeben werden. Dieses Problem habe ich gelöst indem ich vor dem Aufruf die Variable kopiert habe. Es sind oft so kleine Probleme aufgetreten, die aber schnell behoben waren, wenn man die Theorie dazu gelernt hat.

Parametrisieren und Final Touches

Als das Spiel voll funktionsfähig war, haben wir die Möglichkeit gefunden mehrere Game-Controller anzubinden. Ich habe die Spielersteuerung in der Konfiguration parametrisiert und für alle Spielaktionen immer die Steuerung des ersten Spielers übernommen, wie in dem Originalspiel.

In den Shop-Szenen habe ich die Sounds angebunden, so dass es deutlich ist, dass man etwas eingekauft hat. Unser Spiel näherte sich so gut wie möglich an dem Original.

Reflektion:

Für mich war das Projekt ReMasterBlaster ein Erfolg, da wir alle Ziele, die wir uns gesetzt hatten auch erreicht und sogar etwas erweitert und neue Ziele gesetzt haben. Ich habe vieles gelernt und fand es sehr hilfreich, dass ich neben der Projektarbeit eine Vorlesung besucht habe, die sich mit ähnlichen Themen befasst hat. Es hat geholfen, etwas über typische Fehler zu lernen, diese zu machen aber auch zu beheben.

Da das für mich die erste Projektarbeit war, habe ich vieles über Gruppenarbeit gelernt. Die Gruppenbildungsphasen wie - Forming, Storming, Norming, Performing, die ich nur aus der Theorie kannte, konnte ich in der Praxis erleben und daraus lernen. Die gute Koordination aller Projektmitglieder durch den Projektleiter Sebastian Adam hat vieles in der Hinsicht erleichtert.

Insgesamt fand ich es eine sehr positive Erfahrung mit Leuten zusammen zu arbeiten, die so viel Spaß an ihre Arbeit haben und extra Zeit und Aufwand in ihre Tätigkeiten stecken. Oft war das Beheben kleiner Fehler in dem JavaScript-Code eine kleine Herausforderung, die ich und Michael Kao durch Extreme Programming schnell überwältigt haben. Ich konnte mein Wissen in vielen Gebieten, wie Software- und Projektmanagement und Software Entwicklung, vertiefen und neues Wissen aneignen, wie Game Engines, Git und JavaScript.

Für mich war das auch das erste Open-Source Projekt, bei dem ich mitgewirkt habe. Da mein Nebenjob in der Industrie eher Software Entwicklung betreibt, das Geld in einer Firma einbringt, war es eine sehr schöne und wichtige Erfahrung für mich an einem Open-Source Projekt teilzunehmen.

Das Reverse Engineering, was bei unserem Projekt die Hauptrolle gespielt hat, ist etwas was mir eine neue Sichtweise auf Spiele, die ich selber spiele, erlaubt hat. Ich sehe die Spiele nicht mehr als reine Unterhaltung, sondern frage mich öfter wie es gemacht wurde und achte mehr auf Details. Dieses Projekt hat mich auf jeden Fall motiviert zu versuchen, einen anderen Konsolenklassiker mit den neuen Technologien umzusetzen.

Insgesamt finde ich, dass ich kein besseres Thema und Team für das erste Projekt in dem Studium hätte haben können.

Soll und Ist Zustand:

Muss - Kriterien	<ul style="list-style-type: none">• <u>Spielbares Produkt</u>• 2 Steuerbare Spieler mit Tastatur• Bomben legen und <u>Wände sprengen</u>• Gegenspieler bei Treffer entfernen	
Soll - Kriterien	<ul style="list-style-type: none">• Einbindung der Extras (Goodies)• Konfigurationsmenü• <u>Szenenabfolge</u>• Hall of Fame	
Kann - Kriterien	<ul style="list-style-type: none">• Alle Goodies implementieren	
Kann - Kriterien:	<ul style="list-style-type: none">• mehrere Spieler• zusätzliche Eingabegeräte• Shop für Extras• Sprite-Auswahl für die Spieler	

i <https://github.com/bebraw/jswiki/wiki/Game-Engines>

ii <http://www.craftyjs.com/>

iii <https://github.com/>