Technische Universität Braunschweig

ISF

Extensible Games

Marble Machine

Software-Entwicklungspraktikum (SEP)
Sommersemester 2016

Angebot

Auftraggeber
Technische Universität Braunschweig
Institut für Softwaretechnik und Fahrzeuginformatik
Prof. Dr.-Ing. Ina Schaefer
Mühlenpfordtstr. 23
D-38106 Braunschweig

Betreuer: M.Sc. Sven Schuster

Auftragnehmer:

Name	E-Mail-Adresse
Kamil Rosiak	k.rosiak@tu-bs.de
Thilo Krebs	t.krebs@tu-bs.de
Lukas Koppenhagen	l.koppenhagen@tu-bs.de
Thomas Graave	t.graave@tu-bs.de
Paul Maximilian Bittner	p.bittner@tu-bs.de
Mohammad Mahhouk	m.mahhouk@tu-bs.de
Niklas Lehnfeld	n.lehnfeld@tu-bs.de
Maximilian Homann	m.homann@tu-bs.de

Braunschweig, 17. April 2016

Bearbeiterübersicht

Kapitel	Autoren	Kommentare
1	Kamil Rosiak	-
1.1	Kamil Rosiak	In der Gruppe angepasst und abgesegnet
1.2	Kamil Rosiak	In der Gruppe angepasst und abgesegnet
2	Niklas Lehnfeld	In der Gruppe angepasst und abgesegnet
3	Mohammad Mahhouk und Thomas Graave	-
3.1	Mohammad Mahhouk und Thomas Graave	Anordnung der Meilensteine basierend auf Aufwandsabschätzung von Kamil Rosiak und Paul Maximilian Bittner
3.2	Mohammad Mahhouk und Thomas Graave	Anordnung der Meilensteine basierend auf Aufwandsabschätzung von Kamil Rosiak und Paul Maximilian Bittner
4	Thilo Krebs und Maximilian Homann	-
4.1	Maximilian Homann und Thilo Krebs	In der Gruppe angepasst und abgesegnet
4.2	Maximilian Homann und Thilo Krebs	Kostenplan basierend auf Aufwandsabschätzung von Kamil Rosiak und Paul Maximilian Bittner
4.3	Maximilian Homann und Thilo Krebs	In der Gruppe angepasst und abgesegnet
5	Paul Maximilian Bittner	-
5.1	Paul Maximilian Bittner	In der Gruppe angepasst und abgesegnet
5.2	Paul Maximilian Bittner	In der Gruppe angepasst und abgesegnet

5.3	Paul Maximilian Bittner	In der Gruppe angepasst und abgesegnet
6	Lukas Koppenhagen	-
6.1	Lukas Koppenhagen	In der Gruppe angepasst und abgesegnet
6.2	Lukas Koppenhagen	In der Gruppe angepasst und abgesegnet
6.3	Lukas Koppenhagen	In der Gruppe angepasst und abgesegnet
7	Alle	-

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Ziel

Das Projekt befasst sich mit der Entwicklung eines erweiterbaren Spiels. Die Erweiterbarkeit wird durch die Verwendung eines **Plugin-Frameworks** gewährleistet, um so alle Möglichkeiten offen zu lassen. Dabei wird „Marble Machine“ entstehen, ein auf den Gesetzmäßigkeiten der Physik basierendes Logik-Spiel für alle Altersklassen.

1.2 Motivation

Das Softwareentwicklungspraktikum bietet uns die Gelegenheit bisher erlangte Kenntnisse aus den Bereichen der Informatik sowie verwandten naturwissenschaftlichen Zweigen zusammenzuführen, um daraus ein Erlebnis für Jung und Alt zu schaffen.

2 Formale Grundlagen

Für die Realisierung des Projekts wird die Programmiersprache Java verwendet. Hierbei wird mit Hilfe der Physik-**Bibliothek** "JBox2D", um physikalisch korrektes Verhalten zu simulieren, und des gegebenen Plugin-Frameworks "Eleanor" entwickelt. Um den Aufwand des Zusammenführens von Komponenten zu minimieren, wird die Entwicklungsumgebung "Eclipse" verwendet. Ausgeliefert wird das komplette Projekt in Deutsch.

3 Projektablauf

3.1 Meilensteine

Folgend werden alle Meilensteine tabellarisch zusammen mit den abzugebenden Dokumenten und den Abgabeterminen aufgelistet.

Nummer	Meilenstein	Dokumente	Abgabetermin
1	Angebot	Angebot	20.04.2016
2	Haupt-Engine / Core	-	08.05.2016
3	Physik	-	15.05.2016
4	Pflichtenheft	Pflichtenheft	11.05.2016
5	Abnahmetest	Abnahmetestspezifikation	11.05.2016
6	Haupt-Logik	-	22.05.2016
7	Zwischenpräsentation	Prototyp	KW 21
8	Fachentwurf	Fachentwurf	01.06.2016
9	Erweiterte Engine / Core (+Physik)	-	08.06.2016
10	Erweiterte Logik	-	18.06.2016
11	Technischer Entwurf	Technischer Entwurf	29.06.2016
12	Grafik	-	06.07.2016
13	Testdokumentation	Testdokumentation	06.07.2016
14	Audio	-	13.07.2016
15	Tag der jungen Softwareentwickler	Marble Machine	14.07.2016

3.2 Geplanter Ablauf

Die Abbildung auf der nächsten Seite stellt den Zeitplan des Projektes als Gantt Diagramm dar.

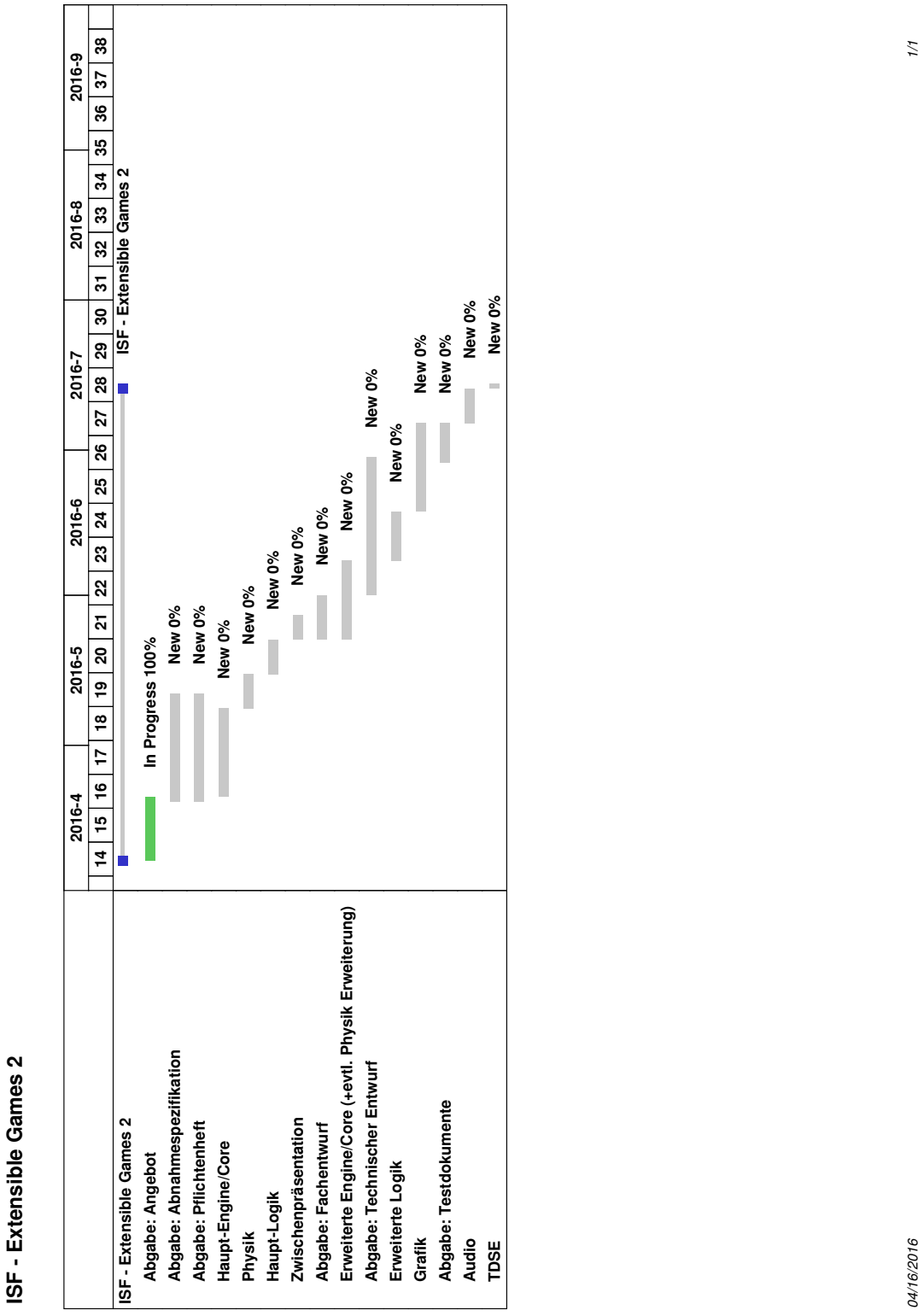

Abbildung 3.1: Projekt Gantt-Diagramm

4 Projektumfang

4.1 Lieferumfang

Im Lieferumfang enthalten ist:

- das ausführbare Spiel Marble Machine
- die Dokumentation/Anleitung zur Marble Machine
- der Quellcode des Spiels
- die Dokumente der Projektplanung und Ausführung

4.2 Kostenplan

Aufgrund der 7 Credit Points des SEP wird der Arbeitsaufwand pro Entwickler auf 210h geschätzt. Somit beträgt der durchschnittliche Arbeitsaufwand je Woche 16h pro Entwickler.
Am Projekt arbeiten 8 Entwickler und die Laufzeit des Projekts beträgt 13 Wochen.

Tabelle 4.1: Kostentabelle

	Stunden	**Preis**
Dokumente	335 Std	33.500 EUR
Engine	471 Std	47.100 EUR
Physik	134 Std	13.400 EUR
Grafik	270 Std	27.000 EUR
Logik	270 Std	27.000 EUR
Steuerung	93 Std	9.300 EUR
Audio	107 Std	10.700 EUR
	Gesamt	
	1680 Std	168.000 EUR

4.3 Funktionaler Umfang

Die Marble Maschine wird in der Lage sein, in verschiedenen Levels Murmeln auf vorgegebenen oder selbst hinzugefügten Bahnen nach grundlegenden physikalischen Gesetzen rollen zu lassen, um damit vordefinierte Ziele zu erreichen.

Es werden zwei Spielmodi zur Verfügung stehen:

- Die Kampagne, in der das Ziel der Level variiert, muss eine Murmel vom Startpunkt zum Zielpunkt gebracht werden. Dies kann durch die Platizierung neuer Bahnelemente als auch durch direkte Steuerung der Murmel geschehen. Ein Punktestand zeichnet die Aktionen des Spielers auf.
- Den Editor-Modus, in dem der Spieler sein eigenes Level erschaffen, Rahmenbedingungen verändern und Gewinnbedingungen definieren kann.

Letztens wird es die Möglichkeit geben eine Statistik des Spielers aufzurufen. Diese beinhaltet u.A. die Anzahl an Versuchen für jedes Level oder die Anzahl an platzierten Bahnelementen.

5 Entwicklungsrichtlinien

Da sich die unterschiedlichen Systeme der Spiele-Engine, wie beispielsweise Graphik, Audio und Physik, gut separieren lassen, werden die Verantwortlichkeiten für diese unter den Gruppenmitgliedern aufgeteilt. Dennoch soll jeder über den Gesamtfortschritt sowie die Arbeiten an den anderen Teilen informiert sein, um auch hier helfen und beraten zu können. Am Spiel selbst werden schließlich alle mitarbeiten.

5.1 Konfigurationsmanagement

Das Projekt wird über das **ISF**-SVN zentral versioniert. Hier liegt immer der aktuellste Stand der Software. Es werden nur lauffähige Versionen eingecheckt. Jede Version wird mit einem kurzen präzisen Kommentar versehen, der mit dem Vornamen der Person beginnt, die diese Änderung eincheckt. Es werden keine Änderungen an fremdem Code ohne Rücksprache vorgenommen.

Das SVN-Repository enthält zusätzlich einen Ordner für Entwürfe und Prototyp-Ressourcen, die noch nicht mit eingebunden sind.

5.2 Design- und Programmierrichtlinien

Syntax- und Semantikrichtlinien

Methoden und Klassen werden JavaDoc dokumentiert. Codefragmente, deren Lesbarkeit eingeschränkt ist, sollen zudem kommentiert werden. Alle Namen sind auf Englisch, Sprachen werden nicht gemischt!
Alle Namen haben eindeutig und nachvollziehbar zu sein.
Der Name ist im Bestfall schon ein ausreichender Kommentar.
Namen gehören grundsätzlich in Camel-Case Notation.
Klassen-, Interface- und Enumerationsnamen werden groß geschrieben.
Member-, Methoden- und Variablennamen werden klein geschrieben.
Alle Buchstaben von Konstantennamen sind groß. Hier ersetzt ein Unterstrich den Camel-Case.
Membervariablen beginnen mit einem Unterstrich. Interfacenamen beginnen mit “I“.

Getter für Booleans beginnen mit “is“.

Designrichtlinien

Logisch verschiedene Programmteile werden als einzelne Plug-Ins umgesetzt, um so deren Unabhängigkeit zu erzwingen und damit das Programm auch dann lauffähig zu halten, wenn einzelne Systeme ausfallen.
Static ist zu vermeiden, es sei denn es handelt sich um Werkzeugklassen wie java.lang.Math.
Objektorientierung ist Klassenorientierung vorzuziehen, sprich Objektkomposition kommt vor Vererbung.
Vererbungshierarchien sollen nicht tiefer als drei Klassen reichen.
Verantwortlichkeiten werden sinnvoll aufgeteilt:

- Nicht auf Objekten operieren, dafür gibt es Methoden.
- keine Gottklassen

Zugriffsmöglichkeiten auf Klassen, Member und Methoden so weit wie möglich einschränken.
Downcasten ist nur in Notfällen erlaubt.
Unterschiedliches Verhalten wird bestmöglich durch verschiedene Objekte statt if, else und switch erreicht.
Code wird nicht kopiert.
Hardcoden ist nur zu Testzwecken erlaubt.

5.3 Verwendete Software

Die Software wird mit der Programmiersprache Java (**JDK 8**) in Eclipse entwickelt.
Für die Dokumentation dienen **MikTeX** und **TeXstudio**. Diagramme zur Planung sollen in **Microsoft Office Visio** erstellt werden. Folgende Bibliotheken werden verwendet:

- Eleanor für Erweiterbarkeit über Plug-Ins
- **LWJGL** für Graphik, Anzeige und Eingabe
- **JBox2D** für die zweidimensionale Festkörperphysik-Simulation

6 Projektorganisation

6.1 Schnittstelle zum Auftraggeber

Auftraggeber ist:

Sven Schuster, M.Sc. Wissenschaftlicher Mitarbeiter

Technische Universität Braunschweig Institut für Softwaretechnik und Fahrzeuginformatik Mühlenpfordtstr. 23 D-38106 Braunschweig

Tel: +49-531-391-2294 E-Mail: s.schuster@tu-bs.de

Die Kommunikation mit dem Auftraggeber erfolgt mittels Telefon, E-Mail oder im persönlichen Gespräch.

6.2 Schnittstelle zu anderen Projekten

Bis auf die verwendeten **Bibliotheken** sind keine Schnittstellen zu anderen Projekten vorgesehen.

6.3 Interne Kommunikation

Zur Abstimmung innerhalb der Gruppe finden regelmäßige Treffen statt. Außerdem gibt es eine **Telegram**-Gruppe, sowie eine Mailing-Liste. Persönliche Gespräche und Telefonate zwischen den Gruppenmitgliedern dienen auch der Kommunikation.

7 Glossar

Bibliothek: Eine Sammlung von Klassen und Methoden, die in andere Programme eingebunden werden können.

IsF: Institut für Softwaretechnik und Fahrzeuginformatik an der Technischen Universität Braunschweig

JBox2D: Diese Bibliothek ist eine Portierung der Physik-Engines Box2D und LiquidFun. Sie bietet die Simulation zweidimensionaler Festkörperphysik an.

LWJGL: Die Lightweight Java Game Library ist eine API, die plattformunabhängigen Zugang zu OpenGL (Graphik), OpenAL (Audio) und OpenCL (Parallelität) gewährt. LWJGL sowie dessen Quellcode stehen (auch kommerziell) frei zur Verfügung.

Microsoft Office Visio: Ein Programm zur Erstellung von Diagrammen einer Vielzahl an Typen.

MikTeX: Eine TeX/LaTeX-Distribution für Microsofts Windows.

Plugin-Framework: Eine bestehende Software, die eine flexible Erweiterung ihrerselbst durch einzelne Module ermöglicht.

Library: siehe **Bibliothek**

Telegram: Messenger-Dienst für Smartphones, Tablets und den PC.

TeXstudio: Ein plattformunabhängiger Editor für die Erstellung von LaTeX-Dokumenten.