Projektvertrag: Entwicklung eines 2D-Spiels

§ 1 Vertragsparteien

Zwischen

- *Jan Kunzmann, G3C der AKSA*
- *Gabriel Gavrilas, G3C der AKSA*
- *Patrick Eigensatz, G3C der AKSA*

und

- *Roger Sax, Mathematiklehrer der AKSA*

wird folgendes vereinbart:

§ 2 Thema der Projektarbeit

Das Ziel des Projektes ist es, ein funktionierendes, lustiges 2D-Computerspiel zu entwickeln. Natürlich ist nicht nur das Endprodukt, also das Spiel selber, sondern auch der ganze Entwicklungsprozess davor, von grosser Bedeutung. Es soll kein Kampf oder Fluchtspiel sein, sondern ein Spiel, bei dem man nicht entdeckt werden, also möglichst verdeckt agieren soll. Der Spielspass entsteht nicht durch exorbitante Grafiken, sondern durch einfaches Design und knifflige Situationen im Spiel.

(2) Es geht nicht darum, einfach ein Spiel zu programmieren, sondern die Programmierung eines Computerspieles in allen Facetten zu lernen.

§ 3 Wissensstand und mögliche Quellen

Patrick verfügt über fortgeschrittene Erfahrungen in der Programmiersprache C++. Jan und Gabriel besitzen grundlegende Kenntnisse der Programmiersprache C++.

(2) In der Projektvorbereitung haben wir uns sehr oberflächlich mit der Entwicklungsumgebung Code::Blocks beschäftigt, und mit dem Open Source Multimedia Framework SFML eine Lösung für das Anzeigen und Abspielen von Grafiken und anderen Multimediaelementen gefunden.

(3) Im Umgang mit Gamedesign bringt keiner von uns Erfahrungen mit.

(4) Unser Wissen beziehen wir hauptsächlich aus Büchern, offiziellen Dokumentationen und dem Internet. Im Anhang sind die genauen Quellen in unserem Rechercheprotokoll angegeben.

§ 4 Begriffsbestimmungen

Multimedia Framework: Eine Bibliothek mit Funktionen, die sich in C++-Programme einbinden lassen. Damit kann man verschiedene Aufgaben plattformübergreifend bewältigen und muss nicht alle Details selber programmieren.

OpenSource Lizenz: Eine Lizenz, die das Einsehen, das Verändern und das Weiterverteilen des Quellcodes erlaubt.

OpenSource-Software (OSS): Software, die unter einer OpenSource-Lizenz verfügbar ist.

Versionskontrollsystem: Ein Programm, das Veränderungen an Dateien aufzeichnet und speichert, sodass mehrere Entwickler den Code gleichzeitig ohne Redundanz verändern können.

GitHub: Für OSS-Entwickler von OpenSource kostenlose Plattform, um die über das Versionskontrollsystem git aufgezeichneten Veränderungen übersichtlich zugänglich zu machen.

Travis CI: Für OSS-Entwickler kostenlose Plattform zur Qualitätssicherung: Kompiliert das Projekt automatisch nach jeder Änderungen und führt Tests durch, um zu sicherzustellen, dass das Programm einwandfrei funktioniert.

Coverity: Für OSS-Entwickler kostenlose Plattform: Führt eine tiefgreifende statische Quellcodeanalyse durch, und hilft so Programmierfehler zu vermeiden und zu finden.

§ 5 Projektziele / Hypothesen / Fragestellung

Ziel unseres Projektes ist ein 2D-Einbrecherspiel zu entwickeln. Der Spieler soll eine Figur spielen, die in Häuser einbrechen muss und dort unentdeckt Gegenstände entwenden soll.

(2) Unser persönliches Ziel ist es, durch dieses Projekt, die Anwendung der Techniken der Programmiersprache C++ zu lernen und zu vertiefen. Dies wollen wir durch das Prinzip Learning-by-Doing erreichen. Jeder in der Gruppe soll am Ende mittelgrosse C++-Programme selber programmieren können.

(3) Wir setzen uns zum Ziel, die Konzepte hinter Frameworks (namentlich SFML) zu verstehen und die programmiertechnischen Abläufe hinter dem Spiel zu definieren und umzusetzen.

(4) Wir wollen unser Spiel komplett im Team entwickeln und mit dem Versionskontrollsystem git vertraut werden. Das alles soll uns später für eigene Projekte und unsere Maturaarbeit helfen.

(5) Das Spiel muss/soll unter eine OpenSource Lizenz gestellt werden.

§ 6 Methode / Vorgehen

Um unser erstes Ziel zu erreichen, haben Gabriel und Jan das Programmieren in C++ erlernt. Unser Grobplan beim Erlernen von C++ war:

- Grundstruktur eines Programmes
- Die Basisdatentypen string, int, float kennen
- Kontrollstrukturen if-else/for/while
- Funktionen, Rückgabewerte, void Typen
- Zeiger, Referenzen
- Namespaces
- Klassen, Objekte
- STL-Templates
- SFML

(2) Das Spiel entwickeln wir aufbauend. Wir wenden uns laufend den verschiedenen Aspekten des Programmierens eines Spieles zu. Zum Beispiel: Strukturierung, Grafik und Spielprinzip. Das Spiel soll auf den Zielbetriebssystemen (siehe unten) spielbar sein. Da wir selber verschiedene Betriebssysteme haben, können wir das Spiel problemlos auf unseren eigenen PCs und Laptops testen.

§ 7 Ressourcen / Voraussetzungen

Wir wollen unser 2D-Spiel nur für den Computer entwickeln. Portierungen auf Konsolen oder andere Plattformen sind nicht geplant.

(2) Zielbetriebssysteme sind gängige Linuxdistributionen und Microsoft Windows.

(3) Zum Entwickeln der Software verwenden wir nur OpenSource Software, und die für OpenSource Entwickler frei zugänglichen Plattformen GitHub, Travis CI, Coverity.

§ 8 Mögliche Erweiterungen des Themas

Da wir die Gameidee gut erweiterbar gewählt haben, könnte man das Spiel noch detailreicher und realitätsnäher gestalten. Man könnte sich auch überlegen die Gameidee in ein 3D Game umzusetzen oder für weiter Konsolen erhältlich machen.

(2) Das angeeignete Wissen und die Erfahrung können auch für eine neue, aufwändigere Gameidee gebraucht werden. Ausserdem könnte man sich auch vertieft in die grafische Gestaltung eines Spieles einarbeiten.

§ 9 Sprache

Unsere Arbeit werden wir in Deutsch schreiben. Aufgrund unserer Thematik werden oft englische Ausdrücke vorkommen, die man im technischen Jargon gar nicht übersetzt, oder für die keine korrekte deutsche Übersetzung existiert. (Zum Beispiel bei Namen und Bezeichnungen: ”pointer” oder ”stack overflow”)

§ 10 Aufbau der Arbeit

Gemäss den Vorgaben im Handbuch für den Projektunterricht. Das heisst es werden folgende Dokumente geschrieben:

- Eine Dokumentation
- Ein Arbeitsprotokoll
- Ein Lernbericht
- Das entstandene Spiel

§ 11 Zeitplan / Besprechungstermine

Wichtige Termine Während der Arbeit ist mindestens ein Fixpunktgespräch mit unserem Betreuer vorgesehen. Obwohl die Softwareentwicklung etwas sehr dynamisches, unkalkulierbares hat, versuchen wir uns so gut wie möglich an die zeitlichen Abmachungen verschiedener, untereinander abgemachten Projektmeilensteine zu halten. Unsere Zeit, sowie die Aufgaben teilen wir auf GitHub ein. Die bereits festgelegten Fixpunkte sind:

Abgabe Lernbericht:	10.12.14
Fixpunktgespräch:	17.12.14
Abgabe Projektarbeit:	21.01.15

Recherchearbeiten Unsere Quellen haben wir in der Vorbereitung in unserem Rechercheprotokoll festgelegt. (Siehe Anhang) Die Recherche wird in der Dokumentation nachzulesen sein. Im Verlauf der Arbeit kann es zu weiteren Recherchen kommen, um unerwarteten Probleme während der Arbeit zu lösen.

Datenerhebungen Für unser Projekt benötigen wir keine besonderen Daten, wie beispielsweise aus einer Umfrage. Solche Arbeiten enfallen folglich.

Auswertung Am Ende der Arbeit, liegen eine Dokumentation, welche die Motivation, das Herangehen und die Entwicklung beschreibt, ein Arbeitsprotokoll, welches Arbeitsvorgänge und Termine beschreibt (Auszug aus git) und ein Lernprotokoll, welches eine Auflistung von Daten mit den dazugehörigen Aktivitäten und Besprechungen enthält, vor.

Wir werten die Protokolle im Hinblick aus auf: Spezifische Herausforderungen beim Programmierung, Zusammenarbeit, Versionierung, Termineinhaltung, Rollenverteilung im Projekt und deren Lösung.

Zusammenstellung / Gestaltung Die Arbeit liegt am Schluss in digitaler Form vor. Die Dokumentation und die obligatorischen Protokolle werden ausgedruckt abgegeben.

Abgabetermin Der Abgabetermin ist, wie oben genannt, der 21. Januar 2015.

§ 12 Themenspezifische Beurteilungskriterien

Die Bewertungskriterien befinden sich im Anhang.

§ 13 Darstellung von Zitaten, Literatur- und Quellenangaben

Literatur und Zitate werden wir gemäss der Variante 1 im Handbuch erwähnen und zitieren. Wir benützen BibTeX zum Bibliographieren.

.....
Ort, Datum

....
Unterschrift der Gruppe Unterschrift des Betreuers