Abschlussarbeit Sommer 2016

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Charaktersteuerung

Programmierung einer intuitiven Charaktersteuerung im Projekt „Sisyfox“

Abgabetermin: 26.04.2016

Prüfungsbewerber:
Marvin Jung
Weddinger Str. 23
38690 Immenrode

Ausbildungsbetrieb:
ckc ag
Am Alten Bahnhof 13
38122 Braunschweig

Inhaltsverzeichnis

Seite

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Die ckc group ist ein von Christian Krentel im Jahr 1989 gegründeter IT- und Business-Consulting-Anbieter in Braunschweig. Sie gehört in dem Bereich zu den führenden Unternehmen Deutschlands. In den Standorten Braunschweig, Berlin, Darmstadt, Hamburg, Dortmund und München beschäftigt sie rund 400 Mitarbeiter[1].

Hauptbranchen des Unternehmens sind die Automobilindustrie samt Zulieferer sowie Banken, Versicherungen, Luft- und Raumfahrt, Retail, Transport und Logistik. Die Kernkompetenz von ckc liegt im Bereich der IT nicht nur auf der Softwareentwicklung, sondern zu Teilen zusätzlich auf der Managementberatung.

1.1. Projektbeschreibung

Das "Sisyfox" Projekt wird entworfen, um für firmeninterne Events (z.B. Seminare, Konferenzen, Sommerfeste, etc.) eine Attraktion zu bieten, die nicht nur mit Spaß verbunden wird, sondern aus der sich auch ein Lernwert ergibt. Das Spiel stellt in einer vereinfachten Form den Mythos des Sisyphos und sein immerwährendes Scheitern gamifiziert[2] dar. Den Mitarbeitern soll dabei der Umgang mit dem Erlebnis des Scheiterns näher gebracht werden. Außerdem soll diese Gamifizierung zu einer Motivationssteigerung führen.

Der Charakter im Spiel wird mit einem Aufbau, auf dem ein 1.20m großer Ball als Steuerkugel[3] befestigt ist, vom Spieler gelenkt. Dafür steht der Aufbau vor dem Spieler auf dem Boden. Die Kugel kann durch den Aufbau mit den Händen auf der Stelle bewegt werden. Die Sensordaten des Aufbaues werden direkt an den Computer übermittelt und müssen in der Steuerung umgesetzt werden.

Die Aufgabe in dem Projekt besteht darin, die Steuerung für den Charakter im Spiel zu programmieren. Diese wird in der Programmiersprache C# geschrieben, damit sie in der Spieleentwicklungsumgebung "Unity" als Skript in das vorhandene Projekt eingebunden werden kann. Hierbei muss auf eine intuitive Steuerung geachtet werden, da die Attraktion einer großen Zielgruppe zur Verfügung gestellt werden soll. Die Intuitivität[4] soll durch eine leicht erlernbare Steuerung, sowie eine deutliches visuelles Feedback realisiert werden. Im Projekt vorhanden sind bereits sämtliche Modelle, Animationen, sowie Sounds und Hintergrundmusik, aber auch schon einzelne, von mir früher programmierte Skripte, die sich z.B. um das Benutzerinterface kümmern. Die benötigten Daten für die Steuerung des Charakters werden von einer Trackballmaus zum Computer übertragen und müssen dort in die Bewegung umgesetzt werden. Außerdem muss im Rahmen der Projektarbeit bestimmt werden, wie sich der Charakter in der virtuellen

[1] *Unternehmensporträt der ckc.* [2016] [1]
[2] *Gamifizierung Definition.* [2016] [2]
[3] Wie in einem Kugellager.
[4] Je leichter es ist sich die Bedienung der Steuerung anzueignen, desto höher ist die Intuitivität.

Welt verhält, sowie in welchen Situationen der Spieler das Spiel vorzeitig verlieren kann und ab welchem Zeitpunkt das Spiel gewonnen ist.

1.2. Projektziel

Mit dem Projekt soll eine gewisse Hartnäckigkeit, gefolgt von gesteigerter Motivation durch Erfolgserlebnisse bei den Mitarbeitern, sowie Besuchern von Firmenfesten und anderen Events erzielt werden. Der Benutzer wird mit jeder gespielten Runde vor die Aufgabe gestellt, eine scheinbar unerreichbare Hürde zu überwinden. Er wird viele Versuche brauchen, um den Gipfel zu erreichen. Der spielerische Aspekt sorgt unweigerlich dafür, dass der Benutzer leichter mit häufigen Niederlagen umgehen kann, um anschließend den anfangs weit entfernten Berg erneut herauszufordern. Sobald der Spieler am Gipfel angekommen ist, gibt es auditive und visuelle Erfolgsmeldungen. Dazu erscheint eine Anzeige, wie lange der Weg zur Bergspitze gedauert hat, damit der Spieler sich mit anderen messen kann und ein Wettkampfgefühl entsteht.

1.3. Projektabgrenzung

Im Projekt sind bereits ein Spielehauptcontroller, sowie eine Animations- und Bewegungssteuerung vorhanden. Der Hauptcontroller sorgt für die Verwaltung der Grundstatus[5] im Spiel. Außerdem speichert er die wichtigsten Variabeln, wie unter anderem Spieldauer oder die Höhe abweichend vom Fuß des Berges. Es wird lediglich ein Verweis via Assoziation in der Charaktersteuerung eingebunden, damit z.B. die Höhendaten kommuniziert werden können.

[5] Die vier Status sind in aktiv, spielend, gewonnen und verloren eingeteilt.

2. Projektplanung

2.1. Projektphasen

Für die Umsetzung des Teils des Projekts stehen insgesamt 70 Stunden zur Verfügung. Die Entwicklung wurde vor Beginn des Projekts auf die verschiedene Projektphasen aufgeteilt. In der folgenden Tabelle [1] lässt sich die grobe Zeitplanung ablesen.

Projektphase	**Geplante Zeit**
Ist-Analyse	6 Std
Soll-Konzept	11 Std
Implementierung	24 Std
Integration in das Unityprojekt	8 Std
Testen und Fehlerbehebung	11 Std
Dokumentation	10 Std
Gesamt	**70 Std**

Tabelle 1.: Grobe Zeitplanung

2.2. Ressourcenplanung

Es werden alle verwendeten Ressourcen, wie Hard- und Software in einer Übersicht im Anhang in Abbildung A.1 auf Seite i aufgeführt. Um möglichen Aufwand für Software so gering wie möglich zu halten, wurde darauf geachtet kostenfreie, sowie Open Source Software zu benutzen.

3. Analysephase

3.1. Ist-Analyse

In dem Spiel gibt es viele kleine Details, welche über das Einbinden von Scripten und das Platzieren von 3D-Modellen realisiert werden. Dazu gehört zunächst die Spielkarte auf der sich der Charakter bewegt. Diese wurde von anderen Teammitgliedern erstellt, mit 3D-Modellen wie z.B. Bäumen, Sträuchern oder Gesteinsbrocken bestückt und in das Projekt implementiert. Zusätzlich wurden 3D-Modelle von Wolken und Vögeln erstellt, die sich im Kreis um den Hauptberg bewegen[1]. Die Spielfigur namens „Sisyfox“ gehört zu den schon erstellten 3D-Modellen und wurde ebenfalls schon mit einer vollständigen Animation versehen. Diese Animation hat jedoch nichts mit der eigentlichen Bewegung des Charakters zu tun. Sie zeigt lediglich visuell eine Laufbewegung an, die aber nicht für eine Positionsverschiebung sorgt. Wie in Abschnitt 1.3 Projektabgrenzung bereits erwähnt, ist der Hauptcontroller, als eines von vielen Scripten, schon vorhanden. Weitere wichtige Scripte sind die Kamerasteuerung, sämtliche Animationsscripte, eine Höhenmeteranzeige und ein Arduinokommunikationsscript, welches eine Verbindung zu einer Hardwarekonstruktion namens „Arduino[2]“ aufbaut. Letztes sorgt dafür, dass sich relativ zur Höhe auf der sich der Charakter befindet, ein angeschlossener Ventilator stärker dreht und eine Nebelmaschine kurz vor Erreichen des Gipfels anspringt.

3.2. Anforderungsanalyse

Um die Positionsverschiebung des Spielecharakters zu implementieren, muss eine Steuerung geschrieben werden. Diese soll von dem Nutzer kontrolliert werden. Das bedeutet, dass der Nutzer Eingaben tätigt und die Steuerung dafür sorgt, dass es ein visuelles Feedback gibt. Hierbei muss innerhalb der Steuerung auf vorher festgelegte Verlier- und Gewinnbedingungen reagiert und zusätzlich eine möglichst realistische und damit intuitive Bewegung ermöglicht werden. In der Entwurfsphase wird das genauere Vorgehen einschließlich einer Programmarchitekturauswahl hinsichtlich der gestellten Anforderungen erläutert.

[1] Die Bewegung wird von einem Script gesteuert.
[2] *Arduino Beschreibung.* [2016] [3]

4. Entwurfsphase

4.1. Zielplattform

Zur Auswahl standen die IDEs[1] „Unity 3D Engine“ und „Unreal Engine 4“. Gewählt wurde die Unity Engine, da es bei Verwendung der Unreal Engine mangels grundlegender Vorkenntnisse zu einer deutlich längeren Entwicklungszeit kommen würde. Es müsste zunächst ein großer Teil der Zeit für Einarbeitung aufgewandt werden. Ein weiterer Grund ist das sehr einfache und intuitive Einbinden von Gameobjekten[2] und deren angebundenen Scripten.

4.2. Architekturdesign

Für den Architekturaufbau standen das „Model-View-Controller-“ (MVC) und das „Presentation-Abstraction-Control-Muster“ (PAC) zur Auswahl.

Das MVC Muster (Siehe Schema Abbildung A.2 auf Seite i) besteht grundsätzlich aus drei Strukturen, die unterschiedliche Ebenen im Programmaufbau darstellen. Diese unterteilen sich in „Modell-“, „View-“ und „Controller-Ebene“. Die Unterteilung dient hierbei dem Zweck, dass jede Ebene flexibel und unabhängig von den Anderen verändert, ersetzt oder erweitert werden kann. Das Modell beinhaltet die Daten, die für die Anzeige nötig sind. Über die View-Ebene werden die Daten aus dem Modell dem Benutzer angezeigt und des Weiteren mögliche Benutzereingaben entgegengenommen. Wenn sich Daten im Modell ändern wird im Regelfall die View-Ebene mithilfe eines Beobachters[3] benachrichtigt, sodass sich die View aktualisieren kann. Die entgegengenommen Benutzereingaben werden von dem Controller übernommen und verarbeitet. In dem objektorientierten Ansatz führt der Controller Methoden im Modell aus, damit die Aktionen des Benutzers wirksam werden. Außerdem kann der Controller die View direkt manipulieren, um z.B. die angezeigten Eingabemöglichkeiten zu verändern.

Für den alternativen Ansatz, in dem man ein System in verschiedenen Einzelteilen erstellt, benötigt man ein Muster, dass jede Aufgabe in einem anderen Teil des System abbildet. Dadurch wird hohe Flexibilität ermöglicht, die sich auch im Wartungsaufwand widerspiegelt. Im PAC Muster (Siehe Schema Abbildung A.3 auf Seite ii) teilt man das System in zwei Richtungen auf, zunächst in drei Einheiten, die grafische Oberfläche, einer Kommunikationsschnittstelle und einem Datenmodell. Diese Einheiten ähneln dem MVC-Muster.

Darüber hinaus teilt sich das Architekturmuster hierarchisch auf sogenannte „Agenten“ auf. Diese Agenten werden auf drei Schichten verteilt, auf *Top-Level-*, *Intermediate-Level-* und *Bottom-Level-*Agenten. Den *Top-Level-*Agenten gibt es nur einmal im System, er übernimmt die globalen

[1] eng: integrated development environment, de: Integrierte Entwicklungsumgebung
[2] „Gameobjekte“ sind die Grundbausteine von jedem Unity Spiel.
[3] Der Beobachter meldet sich bei der View-Ebene, sobald sich Daten im Modell ändern.

Aufgaben. Für die *Bottom-Level*-Agenten ist eine möglichst in sich abgeschlossene Aufgabe vorgesehen. Die *Bottom-Level*-Agenten können über implementierte Methoden in den *Intermediate-Level*-Agenten mit dem *Top-Level* kommuniziert.

In dem Projekt wird sich für eine einfache Variante des PAC-Musters entschieden. Der schon vorhandenen Hauptcontroller wird als Top-Level-Agent eingestuft und die Charaktersteuerung als eine Mischung aus Intermediate- und Bottom-Level-Agent. Die Steuerung wird über eine einfache Assoziation direkt mit dem Hauptcontroller kommunizieren können. Im nächsten Abschnitt wird der Entwurf eines Klassendiagrammes erläutert, in dem diese Herangehensweise umgesetzt wird.

4.3. Entwurf der Benutzeroberfläche

An der Benutzeroberfläche wird in dem Projekt nichts verändert. Es werden jedoch einige Daten von der Charakter Steuerung an den Hauptcontroller übermittelt, welche auf dem Bildschirm angezeigt werden. Im Anhang befindet sich eine Skizze (Siehe Anhang Abbildung A.4 auf Seite ii), sowie ein Anzeigebild aus dem laufendem Spiel (Siehe Anhang Abbildung A.5 auf Seite iii), in dem die wichtigsten Anzeigen eingetragen sind. Auf der Skizze befindet sich Punkt „A“ oben links auf dem Bildschirm lokalisiert. An der Position wird eine weiße Bergsilhouette gezeigt, die sich relativ zur Charakterhöhe am Berg füllt. Die Charakterhöhe wird von der Charaktersteuerung ermittelt und an den Hauptcontroller für die Anzeige der Höhe und der Bergfüllanzeige übergeben. Gleich rechts daneben - Punkt „B“ - zeigt die vergangene Zeit für den bisherigen Versuch an. In der Mitte des Bildschirms ist eine Höhenanzeige in Metern eingebaut (siehe Punkt „C“). Diese erscheint alle 150 Höhenmeter und gibt den derzeitigen Zwischenstand an. Anschließend verschwindet sie wieder. Als letzten Punkt auf der Skizze ist die Charakterposition eingezeichnet.

In der Abbildung A.5 auf Seite iii sind alle auftretenden Benutzerinterface Anzeigen im Spiel dargestellt.

4.4. Geschäftslogik

Zunächst werden die Anforderungen an die Charaktersteuerung festgelegt, damit daraus ein Klassendiagramm entstehen kann. Das Klassendiagramm kann im Anhang in Abbildung A.6 auf Seite iii eingesehen werden. Die Steuerung muss als Hauptaufgabe die Mausinformationen, sowie deren Bewegungsrichtung ermitteln und in eine Bewegung der virtuellen Kugel übersetzen. Um das zu realisieren, muss die aktuelle Mausposition gespeichert werden und im anschließenden Frame[4] der Unterschied zum vorherigen Frame berechnet werden. Aus dieser Bewegung kann ein Richtungsvektor ermittelt werden. Dieser Vektor beinhaltet die genaue Richtung, sowie die Strecke die sich die Maus bewegt hat. Durch die Strecke kann die Steuerung eine Beschleunigung

[4]Ein Frame ist ein angezeigtes Bild während das Spiel läuft. Das Spiel läuft mit ca. 60 Frames pro Sekunde.

der Kugel berechnen. Ist das geschehen, so kann die Beschleunigung auf die Kugel angewandt werden, damit sie sich in eine Richtung bewegt.

Zusätzlich sollen vorher festgelegte Mechanismen überprüfen, in welchen Situationen der Spieler das Spiel gewinnt oder verliert. Dieser Zustand oder auch Status des Spiels wird vom Hauptcontroller verwaltet (Siehe Statemachine Anhang Abbildung A.7 auf Seite iv). Der Status geht vom *inaktiv*[5] Status über in den *laufend* Status, welcher je nach Spielweise in *gewinnen* oder *verlieren* endet. Gewinnen wird der Spieler, wenn er am Gipfel des Berges angelangt ist und dort in eine Triggerbox[6] (Siehe auch Abbildung A.8 auf Seite iv) hineinläuft und somit den *gewonnen* Status erreicht. Der Niederlage-Mechanismus funktioniert ähnlich. Hierfür wurde festgelegt, dass bei Berührung eines Baumes (Siehe auch Abbildung A.9 auf Seite v) oder Felsbrockens das Spiel verloren sein soll. Dafür werden diese ebenfalls mit einem Trigger versehen, der den *verloren* Status auslöst.

Zusätzlich soll der Kugel eine möglichst realistische Physik gegeben werden. Das heißt, dass der Charakter nicht in der Lage sein soll, die Kugel seitlich an einem Abhang entlang zu rollen, ohne dass sie ihm zur Seite weg rollt und er die Runde verliert. Der Benutzer kann als Reaktion versuchen den Charakter unter die Kugel zu manövrieren, sodass die Steigung vor ihm liegt und nicht seitlich. Dann kann der Charakter die Kugel festhalten. Bei einem seitlichen Steigungswinkel von über 65°verliert der Spieler sofort das Spiel. Liegt der Winkel zwischen 40°und 65°, dann hat der Spieler 2 Sekunden, um seine Spielfigur zu richten. Auf einem Hand mit einem Neigungswinkel zwischen 20°und 40°hat der Spieler abhängig vom Winkel vier bis zwei Sekunden Zeit die Spielfigur zu richten. Es liegt im Anhang die Abbildung A.10 auf Seite v vor, die diese Funktionsweise verbildlicht. Wie angesprochen kann die Spielfigur die Kugel an beliebig steilen Steigungen festhalten, solange sie sich unter[7] der Kugel befindet. Befindet sich der Charakter auf der anderen Seite („über“ der Kugel), dann ist die Kugel ebenfalls bei einem Winkel von 65°verloren.

Als weitere Funktion der Steuerung soll der Charakter, wenn er durch einen Busch läuft, verlangsamt werden. Hierzu muss für jeden Busch ein weiterer Trigger (Siehe auch Abbildung A.11 auf Seite vi) hinzugefügt werden.

4.5. Maßnahmen zu Qualitätssicherung

[5] Im inaktiv Status verweilt das Spiel solange, wie es keine Benutzerinteraktion gibt.
[6] Ein Trigger ist in dem Fall eine unsichtbare Box mit Kollisionserkennung an den Außenseiten, die ein Kollisionsevent erzeugt. Das Event löst eine Funktion beim kollidierenden Objekt aus.
[7] „Unter“ bedeutet, dass der Spieler gerade auf die Steigung zuläuft.

5. Implementierungsphase

5.1. Implementierung der Geschäftslogik

5.2. Implementierung der Benutzeroberfläche

6. Abnahme- und Einführungsphase

6.1. Abnahme

6.2. Einführung

6.3. Dokumentation

7. Fazit

7.1. Soll-/Ist-Vergleich

7.2. Lessons Learned

7.3. Ausblick

Literaturverzeichnis

[1] *Unternehmensporträt der ckc.* 2016. URL: http://www.ckc-group.de/index.php?id=709.

[2] *Gamifizierung ist die Übertragung von spieltypischen Elementen und Vorgängen in spielfremde Zusammenhänge mit dem Ziel der Verhaltensänderung und Motivationssteigerung bei Anwenderinnen und Anwendern.* 2016. URL: http://wirtschaftslexikon.gabler.de/Definition/gamification.html.

[3] *Die Arduino Hardware ist ein Entwicklerboard zum Basteln von eigenen Schaltkreisen und zum Ausführen von Steuerungsprogrammen für die Verwaltung von angeschlossener Elektronik.* 2016. URL: https://www.arduino.cc/en/Guide/Introduction.

A. Anhang

Abbildung A.1.: Verwendete Ressourcen

Hardware:

- Büroarbeitsplatz mit Standrechner

Software:

- Windows 7 Professional Service Pack 1 - Betriebssystem
- Unity 3D Version 5.3.3f1 - 3D Entwicklungsumgebung
- Visual Studio 2015 - Code Entwicklungsumgebung
- MiKTEX- Distribution des Textsatzsystems TEX
- TEXMaker - LATEXSchreibprogramm
- Dia Version 0.97.2 - Anwendung zum Zeichnen strukturierter Diagramme
- Entwickler - Implementierung der Scripte / Realisierung
- Anwendungsentwickler - Code Begutachtung

Abbildung A.2.: Model View Controller

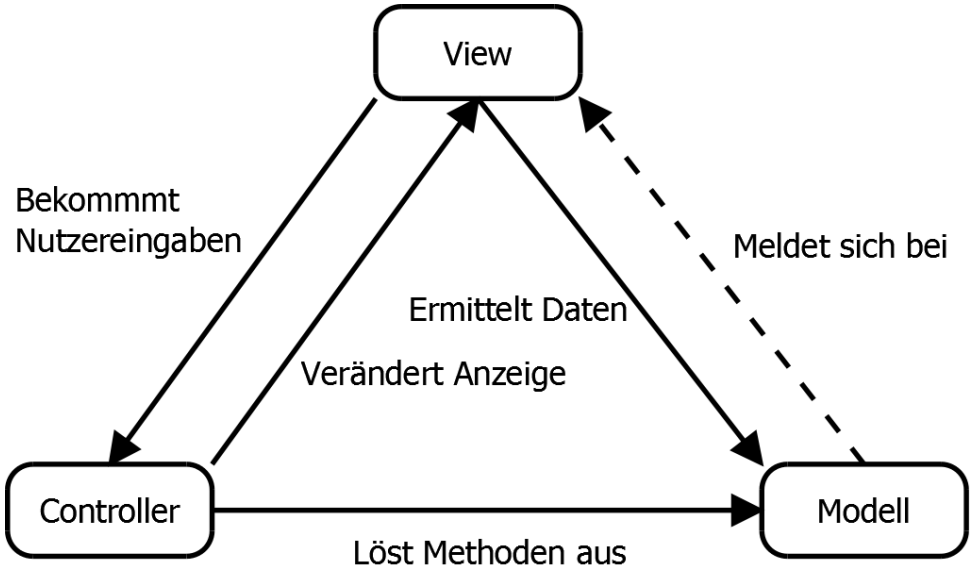

Abbildung A.3.: Presentation Abstraction Control

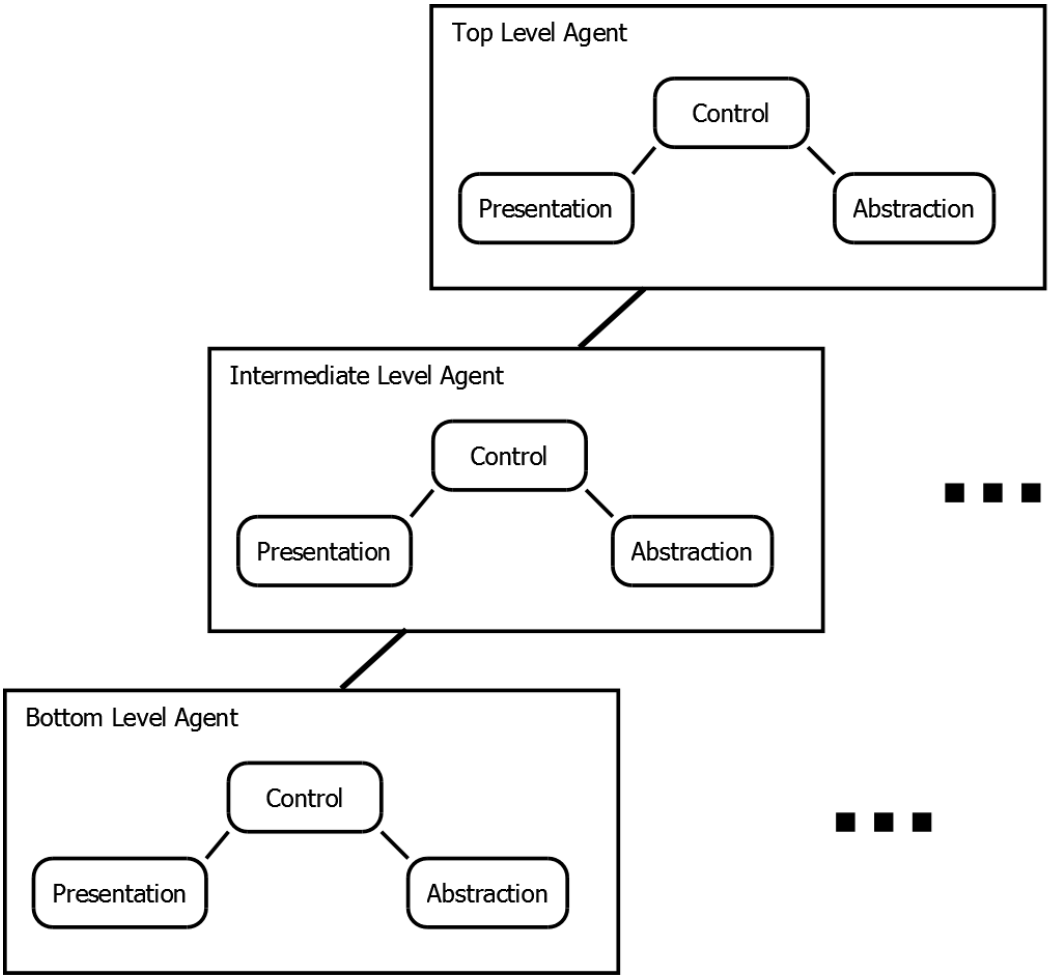

Abbildung A.4.: Benutzer Oberfläche Skizze

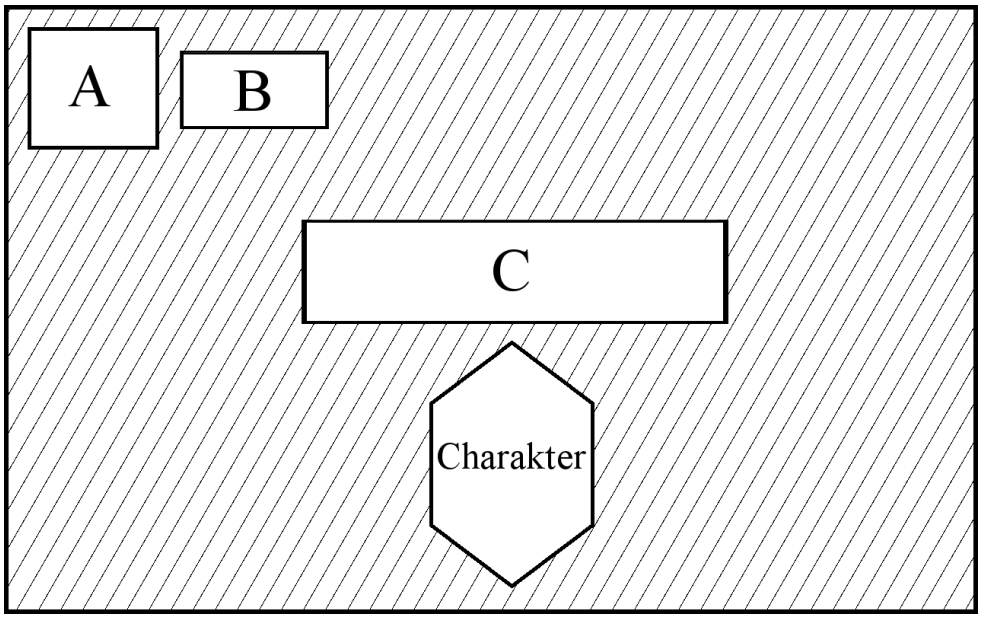

Abbildung A.5.: Benutzer Oberfläche Im Spiel

Abbildung A.6.: Klassendiagramm Charaktersteuerung mit Sphere

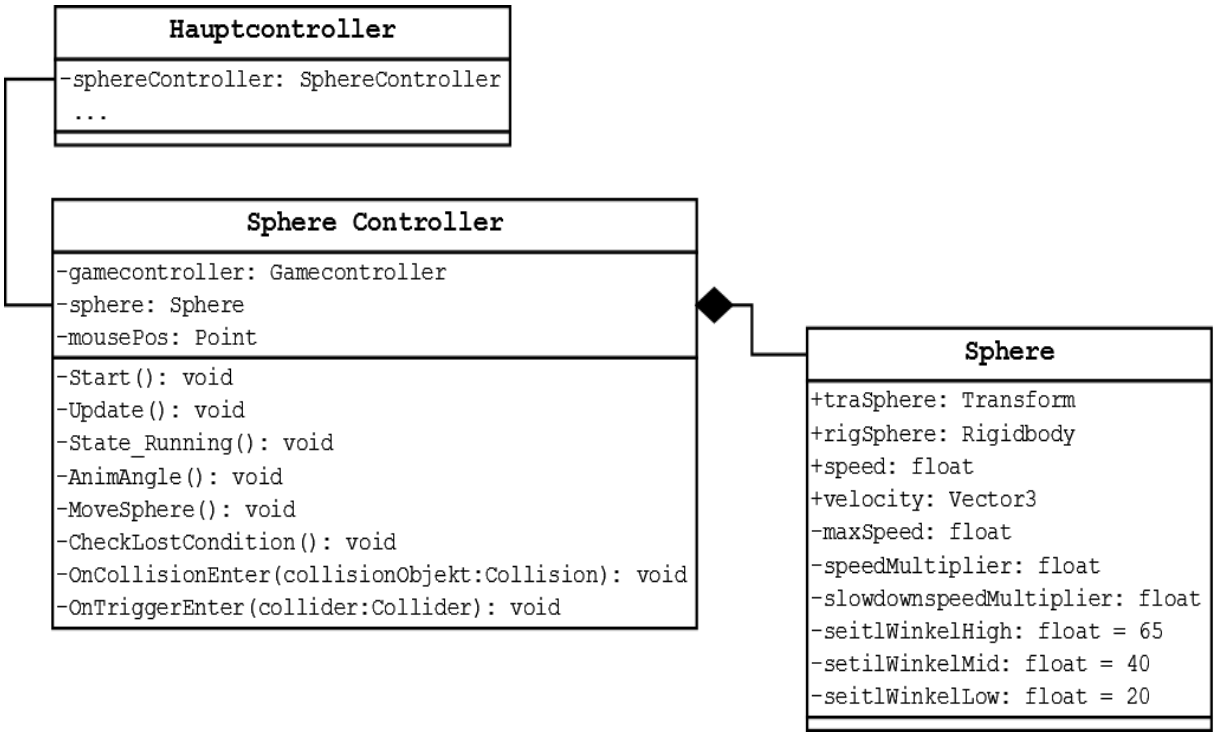

Abbildung A.7.: Zustandsdiagramm Statusverwaltung

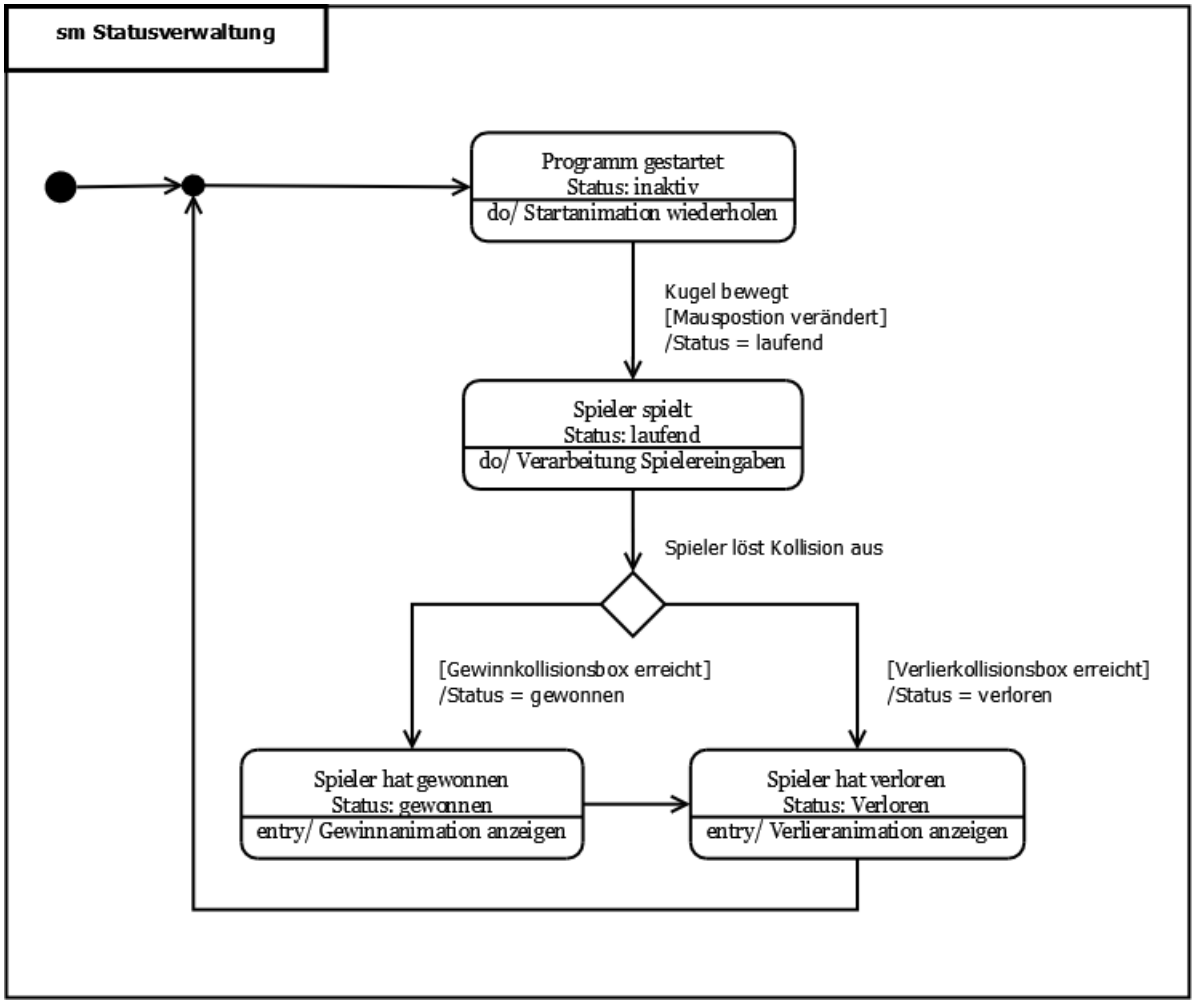

Abbildung A.8.: Gewinn Triggerbox am Gipfel des Berges

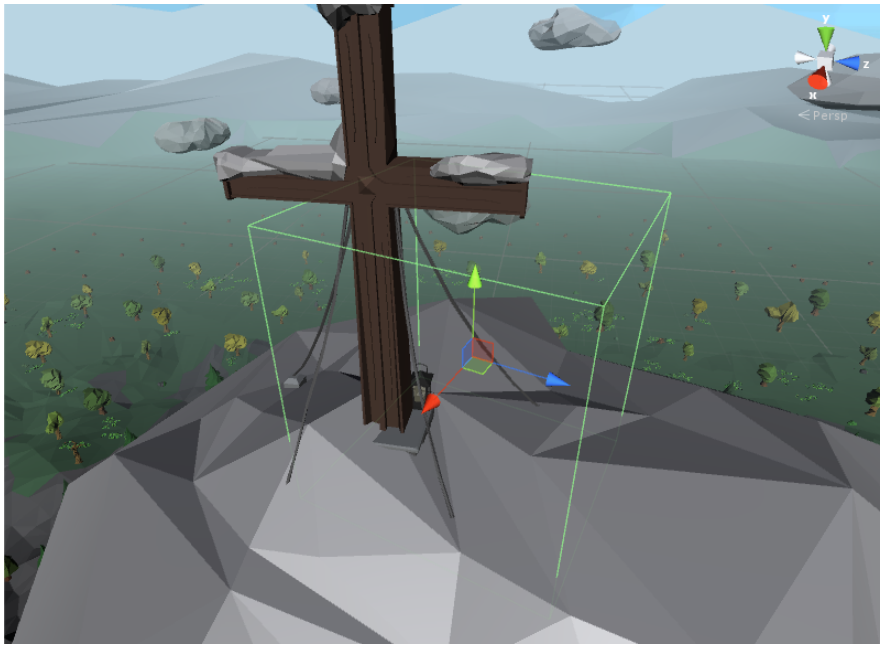

Abbildung A.9.: Baum Triggerbox

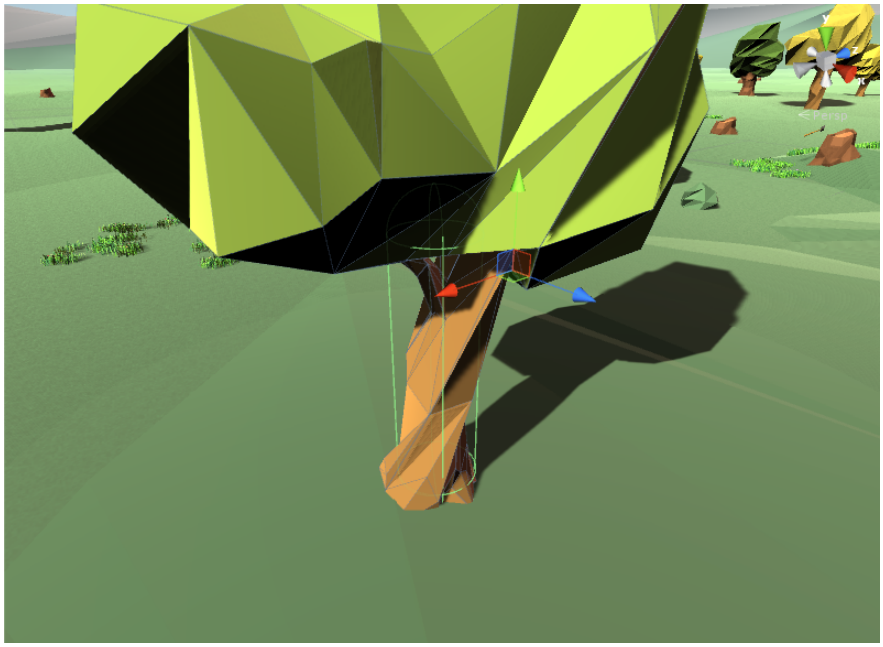

Abbildung A.10.: Verloren Status bei seitlichem Steigungswinkel

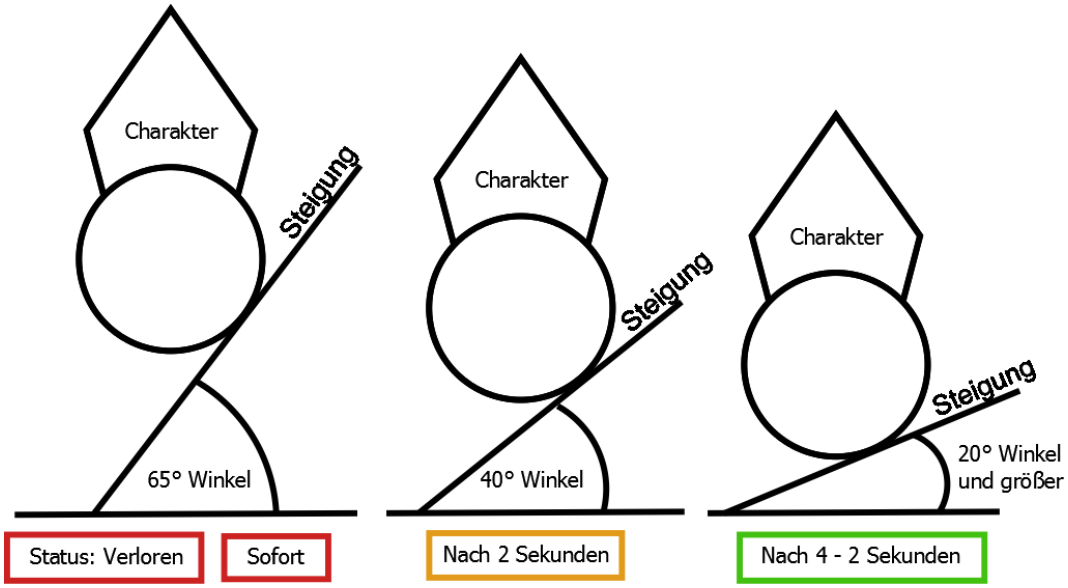

Abbildung A.11.: Busch Triggerbox am Rand des Berges