ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Collaborative Game Decision Analysis with Process Models

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Yasin Agirbas
yasin.agirbas@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Michael Winter

2019

Fassung 4. Dezember 2019

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Die Geschäftsprozessmodellierung hat sich in der heutigen Zeit stark weiterentwickelt und sich für Unternehmen erfolgreich bewährt. Die Prozessmodellierung hat daher einen wichtigen Standpunkt bei der Führung eines Unternehmens erreicht. Die Konzepte der Prozessmodellierung können anhand von realitätsnahen Spielen gelehrt werden. Durch diese Spiele können Spieler die Prozessmodellierung simulieren und durch Echtzeitrückmeldungen aus ihren Fehlern lernen. Durch den Einsatz von Spielelementen, wie zum Beispiel Punkte- und Rangsysteme, kann die Motivation der Mitarbeiter positiv beeinflusst werden.

In dieser Abschlussarbeit wird ein Spiel entworfen und entwickelt, in welchem der Spieler eigenständig geschäftliche Entscheidungen treffen soll. Die Besonderheit dieses Spiels ist die dynamische Generierung der Prozessmodelle anhand der Entscheidungen des Spielers. Diese Prozessmodelle ermöglichen dem Spieler seine eigenen Entscheidungen zu analysieren und dadurch seine Spielstrategie anzupassen und möglichst zu verbessern. Fehlentscheidungen können zur Niederlage im Spiel führen. Es haben sich bisher drei mögliche Spielstrategien durchgesetzt. Diese Spielstrategien unterscheiden sich hauptsächlich durch ihr notwendiges Risiko. Durch hohes Risiko kann der Spieler einen hohen Profit generieren, jedoch riskiert er dabei das Spiel vorzeitig zu verlieren. Während dessen bietet eine deterministische Spielstrategie finanzielle Sicherheit, aber dafür einen geringeren Maximalprofit.

Danksagung

Zuerst möchte ich mich bei meinem Gutachter Prof. Dr. Manfred Reichert, für die Ermöglichung dieser Abschlussarbeit bedanken.
Ein besonderer Dank gilt für meine fleißigen Spieltester Sina, Ali, Musab, Osman und Sebastian. Sina konnte mir Spielstrategien lehren, die mir nicht einmal als Entwickler bekannt waren und sie hält auch den bisherigen Spielrekord.
Ein großer Dank gilt ebenfalls für meinen Betreuer Michael Winter. Er war sehr hilfsbereit und hat mich stets beraten, wann immer es notwendig war.
Ebenfalls möchte ich mich bei meinen Brüdern und bei meinen Eltern bedanken. Sie haben mich bei meinem Studium stets motiviert und unterstützt.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen Zeit wird der Einsatz von Prozessmodellen immer wichtiger. Unternehmen müssen in der Lage sein, Geschäftsprozesse zu optimieren, damit sie weiterhin wettbewerbsfähig bleiben können [1]. Durch die Prozessmodellierung können Prozessausführungen leichter nachvollzogen werden und dadurch kann die Automatisierung von Teilaufgaben unterstützt werden. Virtuelle Spiele können genutzt werden, um Prinzipien der Geschäftsprozessmodellierung zu lehren [2]. Diese Spiele werden als *Serious Games* bezeichnet. Gamification ist die Anwendung von Spielelementen in einer Umgebung außerhalb eines Spiels [3]. Durch die Gamification kann die Produktivität und die Kreativität von Mitarbeitern gesteigert werden [4].
In dieser Arbeit soll nun ein Spiel entwickelt werden, in welchem der Spieler eigenständig geschäftliche Entscheidungen treffen soll. Bei jedem Spielablauf wird ein Prozessmodell mit allen Entscheidungen des Spielers dynamisch generiert. Diese Prozessmodelle sollen den Spieler dabei unterstützen, neue und bessere Spielstrategien zu finden.
Die Spielwelt beinhaltet sowohl deterministische als auch zufallsbedingte Interaktionsmöglichkeiten, die dem Spieler sichere oder risikoreiche Entscheidungen ermöglichen. Das Ziel des Spielers ist es möglichst viel Profit zu generieren. Um dies zu erreichen muss er seine vorherige Vorgehensweise anhand der Prozessmodelle analysieren und dadurch seine Strategie verbessern. Wenn der Spieler durch die Analyse seine Fehlentscheidungen erkennt, dann kann er diese Fehler in den zukünftigen Spielrunden besser vermeiden. Es ist auch denkbar, dass eine Kollektion aus den besten Prozessmodellen erstellt werden könnte. Die besten Prozessmodelle sind hierbei die Spieldurchläufe

mit denen die höchsten Profite erreicht wurden. Die Prozessmodelle können ebenfalls als Muster dienen, an denen sich unerfahrene Spieler orientieren können, um ohne weitreichende Spielerfahrung gute Resultate im Spiel zu erzielen.

1.2 Problemstellung und Zielsetzung

Die hauptsächliche Problemstellung bei dieser Arbeit ist das Entwerfen und das Implementieren einer adäquaten Spielwelt. Die Spielwelt muss dem Spieler einige geschäftliche Entscheidungsmöglichkeiten bieten können. Daher muss zuerst die Spielwelt mit allen Elementen entworfen und als Prototyp implementiert werden, um ermitteln zu können ob die Anzahl der verschiedenen Entscheidungsmöglichkeiten genügt. Dabei ist es notwendig die erforderlichen Kenntnisse über C# und der *Unity* Spiel-Engine zu erlernen. Ebenfalls soll eine intuitive Benutzeroberfläche implementiert werden, mit der das gesamte Spiel gesteuert wird. Der Spieler soll mit Gegenständen handeln können. Daher müssen einige Gegenstände mit unterschiedlichen Eigenschaften implementiert werden. Es wurde entschieden genau fünf NPCs (engl. für non-player character) mit unterschiedlichen Funktionalitäten zu entwickeln.
Zusätzlich muss ermittelt werden, wie die dynamische Generierung von Prozessmodellen implementiert werden soll. Dazu werden zuerst mögliche Ansätze auf Papier entworfen, um zu erkennen welche Informationen pro Entscheidung gespeichert werden müssen, damit die Prozessmodelle einen Sinn ergeben. Es sollen letztendlich die Entscheidungsnummer, die NPC-Bezeichnung, der Eventlog und die Geldmenge pro Entscheidung gespeichert und angezeigt werden. Der Eventlog bezieht sich hierbei auf eine textuelle Beschreibung der Entscheidung. Diese Beschreibung soll dabei helfen nachzuvollziehen was bei der Entscheidung letztendlich getan wird.
Eine weitere Zielsetzung ist das Finden von Spielstrategien durch den Einsatz der generierten Prozessmodelle. Es werden in dieser Arbeit drei mögliche Spielstrategien vorgestellt, die nicht alle möglichen Spielstrategien abdecken. Eine persönliche Zielsetzung ist es die Spielwelt in einem altmodischen Pixelstil zu entwerfen.

1.3 Struktur der Arbeit

Die Struktur der Ausarbeitung sieht wie folgt aus. In Kapitel 2 werden relevante Grundlagen erklärt, die für das Verständnis dieser Arbeit notwendig sind. Die Grundlagen beinhalten die Themenpunkte Prozessmodellierung und die *Unity* Spiel-Engine. Danach wird in Kapitel 4 das Spiel *MineTrade* eingeleitet und alle wichtigen Spielelemente beschrieben. Zusätzlich werden die Hintergedanken und die Umsetzung des Spieldesigns erläutert. In Kapitel 5 werden detailliert drei mögliche Spielstrategien festgelegt und angewandt. Am Ende dieses Kapitels befindet sich eine Diskussion zu diesen Spielstrategien. Kapitel 6 beinhaltet verwandte Arbeiten und das Kapitel 7 fasst die Arbeit zusammen. Zuletzt befindet sich in Kapitel 8 ein Ausblick mit Erweiterungsmöglichkeiten bezüglich *MineTrade*.

2

Grundlagen

Der Schwerpunkt dieser Ausarbeitung ist der Einsatz von Prozessmodellierung in einem virtuellem Spiel. Deshalb werden in diesem Kapitel die wichtigsten Grundlagen erläutert. Im Abschnitt 2.1 werden die Grundlagen der Prozessmodellierung zusammenfassend erklärt. Die *Unity* Spiel-Engine beinhaltet sämtliche Technologien die verwendet wurden und diese werden im Abschnitt 2.2 beschrieben.

2.1 Prozessmodellierung

In einem Unternehmen finden zahlreiche Geschäftsprozesse statt und diese Prozesse können modelliert werden. Die Prozessmodellierung soll die Wettbewerbsfähigkeit eines Unternehmens unterstützen [1]. Ein System aus mehreren Prozessmodellen wird als Prozessmanagementsystem (PMS) bezeichnet. Anhand des PMS sollen sämtliche Notwendigkeiten des Unternehmens erfüllt werden. Diese Notwendigkeiten beinhalten die aktive Prozessunterstützung anhand von Arbeitslisten, die Automatisierung von Teilaufgaben und die bessere Nachvollziehbarkeit einzelner Prozessausführungen. Es existieren verschiedene Notationen im Bereich der Prozessmodellierung, aber eine Notation die besonders häufig umgesetzt wird, ist die sogenannte *BPMN 2.0* Notation [5]. Die *BPMN 2.0* Notation beinhaltet drei Flussknoten aus denen Prozesse modelliert werden können. Diese Flussknoten werden in der Abbildung 2.1 dargestellt [5]. Diese drei Flussknoten haben folgende Eigenschaften. Ein *Activity*-Knoten beschreibt eine Aktivität die zuerst vollbracht sein muss, bevor der Kontrollfluss weiter verläuft. Zum Beispiel kann eine Information nicht verarbeitet werden, bevor sie überhaupt eingereicht wird.

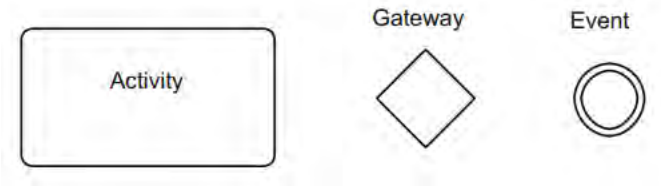

Abbildung 2.1: Flussknoten in *BPMN 2.0*

Mit einem *Gateway*-Knoten können Verzweigungen im Prozessmodell erstellt werden. Je nachdem ob eine bestimmte Bedingung erfüllt ist, wird ein Pfad nach der Verzweigung gefolgt. Dadurch kann der Kontrollfluss strukturiert werden.
Die *Event*-Knoten werden als Auslöser verwendet. Sobald der Kontrollfluss einen *Event*-Knoten erreicht, löst dieser *Event*-Knoten die nachfolgenden Knoten aus. Dabei ist der Endknoten jedoch eine Ausnahme, da dieser Knoten zum Terminieren des Kontrollflusses genutzt wird. In der Abbildung 2.2 befindet sich ein Beispiel für die Anwendung der *BPMN 2.0* Notation [5].

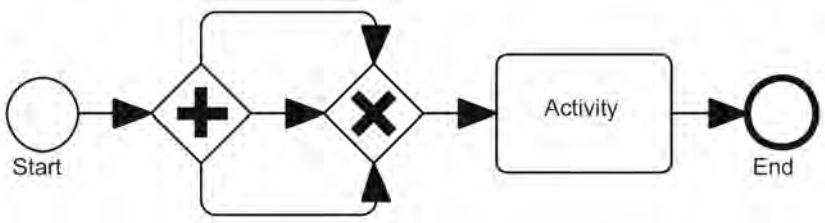

Abbildung 2.2: *BPMN 2.0* Beispiel mit Flussknoten

Dieses Prozessmodell beginnt mit einem Startknoten. Durch den *Gateway*-Knoten wird der Kontrollfluss auf drei Pfade verzweigt. Dadurch wird die Aktivität im *Activity*-Knoten genau dreimal ausgeführt. Die Ausführung kann hierbei parallel geschehen. Am Ende wird der Prozess durch ein *Event*-Knoten terminiert.

In der *BPMN 2.0* Notation sind die sogenannten *Token* ebenfalls sehr wichtig. Diese Tokens repräsentieren den zuvor genannten Kontrollfluss [5]. Sie werden grafisch durch schwarze Punkte dargestellt und sind in einem Beispiel auf der Abbildung 2.3 zu sehen.

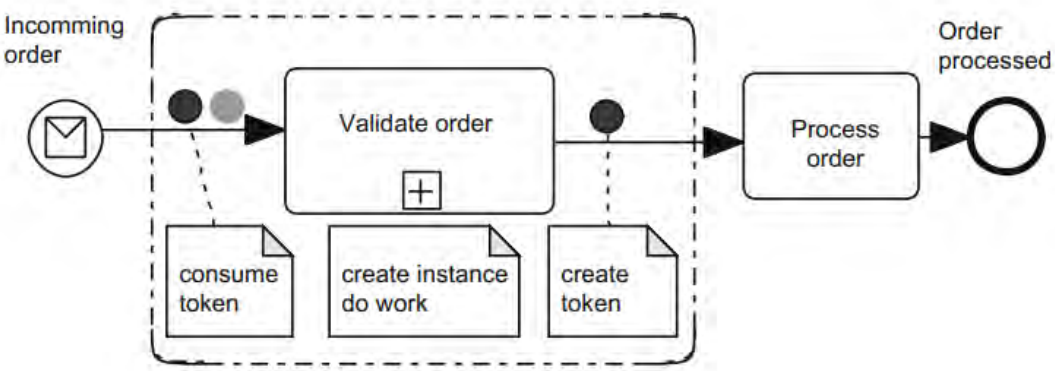

Abbildung 2.3: *BPMN 2.0* Beispiel mit Tokens

Der *Event*-Knoten *Incomming order* löst den *Activity*-Knoten *Validate order* mit den zwei Tokens aus. Die Tokens werden von dem *Activity*-Knoten nach der Ausführung der Aktivität konsumiert und erstellt danach einen neuen Token. Dieser Token wird dann zur Aktivität *Process order* weitergeleitet, damit diese Aktivität ausgeführt wird. Am Schluss ist ein *Event*-Knoten, welcher diesen Prozess terminiert.
In Bezug auf diese Ausarbeitung soll insbesondere die bessere Nachvollziehbarkeit durch Prozessmodelle genutzt werden. Denn die Prozessmodelle sollen dem Spieler bei seinen zukünftigen Entscheidungen unterstützen. Die funktionalen Anforderungen eines PMS sind abhängig von der Art der Anwendungsfunktionen. Diese Anwendungsfunktionen entsprechen in diesem Fall den Handlungsmöglichkeiten mit den sogenannten NPCs. Dies wird im Abschnitt 4.2 genauer erklärt. Prozessmodelle können normalerweise sequentiell und verzweigend verlaufen. Eine Verzweigung findet statt, wenn sich ein Prozess in zwei Prozesse aufteilt. Die Prozessmodelle in dieser Ausarbeitung sind stets sequentieller Art, da es in dem Spiel keine verzweigenden Prozesse gibt. Des Weiteren werden im Spiel Prozesse als Entscheidungen bezeichnet und gezählt, da der Spieler keine Vorgabe an Prozessabläufen erhält, sondern sich eigenständig im Rahmen

der Möglichkeiten entscheiden soll. Ebenfalls soll die Verwendung von sequentieller Prozessmodellierung die Überschaubarkeit hoch und die Komplexität niedrig halten. Das ist besonders wichtig, weil es im Spiel ermöglicht wird, mehrere Prozessmodelle nebeneinander anzeigen zu lassen, um sie miteinander vergleichen zu können.
Normalerweise werden Prozessmodelle vor der Ausführung der Geschäftsprozesse erstellt [1]. Im Spiel jedoch werden die Prozessmodelle dynamisch zur Spielzeit erstellt und zwar nach jeder getroffenen Entscheidung. Diese Prozessmodelle können dann zur Entscheidungsanalyse, wie in Kapitel 5, genutzt werden. Im Spiel hat jedes Prozessmodell einen Startknoten. Der Startknoten beinhaltet die Bezeichnung des jeweiligen Prozessmodells und die Gesamtanzahl der getroffenen Entscheidungen. In der Abbildung 2.4 ist ein Prozessmodell aus dem Spiel zu sehen. Dieses Beispiel besteht aus den ersten drei Entscheidungen die getroffen wurden. Jede Entscheidung wird in einem rechteckigen Block angezeigt. Diese Blöcke beinhalten alle wichtigen Informationen der Entscheidungen. Das Prozessmodell aus der Abbildung 2.4 kann in die Form der *BPMN 2.0* Notation umgeformt werden. Das Resultat wird in der Abbildung 2.5 angezeigt. Das Prozessmodell aus dem Spiel bietet im Vergleich eine bessere Übersicht über die genutzten NPCs und dem Geldstand bei jeder Entscheidung.

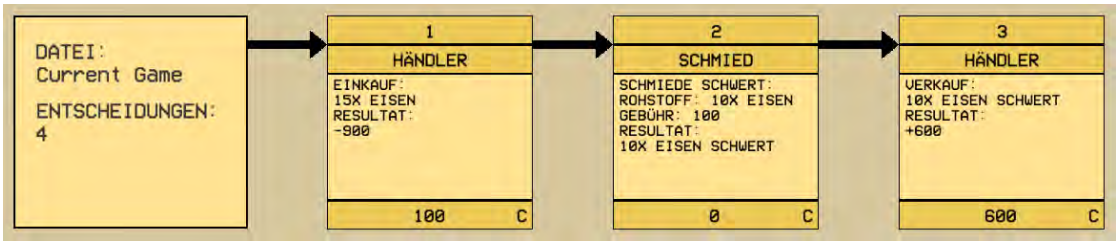

Abbildung 2.4: Prozessmodell aus drei Entscheidungen und einem Startknoten

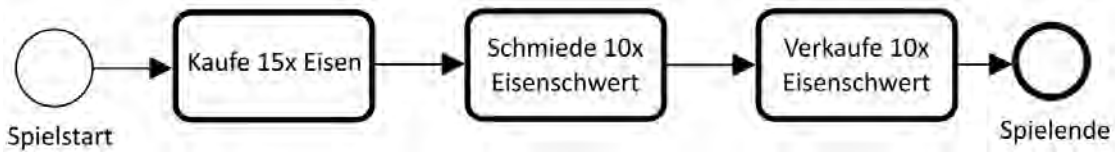

Abbildung 2.5: Prozessmodell umgeformt mit der *BPMN 2.0* Notation

2.2 Unity Spiel-Engine

Für die Entwicklung des Spiels wurde die *Unity* Spiel-Engine verwendet. Der Einsatz von Spiel-Engines ist sehr vorteilhaft, da sie Zeit sparen und häufig verwendete Funktionalitäten in benutzbarer Form anbieten [6]. Anstatt Funktionen in Eigenarbeit von Grund auf neu programmieren zu müssen, können Spieleentwickler direkt mit der Implementierung des Spiels beginnen. Mit *Unity* kann man dreidimensionale und auch zweidimensionale Spiele programmieren. Das für diese Ausarbeitung implementierte Spiel *MineTrade* ist ein zweidimensionales Spiel. Für die grafischen Elemente konnte das Sprite-System von *Unity* gut genutzt werden. Das Sprite-System ermöglicht es dem Entwickler anhand von Tilemaps seine grafischen Elemente zusammen zu legen und auch im nach hinein ohne Probleme zu verändern [6].
Neben den grafischen Funktionalitäten bietet *Unity* das sogenannte Prefab-Systen [6]. Prefabs sind Spielobjekte die erstellt werden können, um sie dann zur Laufzeit des Spiels instantiieren zu können. Das hat den Vorteil ohne Probleme mehrere Instanzen des gleichen Spielobjekts zu generieren. Des Weiteren können diese Prefabs mit sogenannten Scriptable-Objects verknüpft werden. Scriptable-Objects speichern alle Informationen die ein Prefab benötigt. Ein konkretes Beispiel hier für sind Items. Ein Item braucht eigene Informationen, wie zum Beispiel einen Namen, einen Verkaufswert und eine Seltenheit. Diese Informationen werden in den jeweiligen Scriptable-Objects festgelegt und anschließend zu einem Prefab zugeordnet. Allgemein werden Prefabs für die Grafik und Scriptable-Objects für die Daten genutzt [6]. Dieses System aus Prefabs und Scriptable-Objects unterstützen die Sauberkeit und Überschaubarkeit der Spielarchitektur, da sie sehr modular und einfach anpassbar sind [6]. Ebenfalls vorteilhaft ist ihre Wiederverwendbarkeit, denn anhand von Vererbung, bekannt aus der objektorientierten Programmierung, können aus einem abstrakten Item-Objekt neue alternative Items generiert werden. So wird beispielsweise aus einem abstrakten Schwertobjekt ein Goldschwert. Der Abschnitt 4.3 enthält mehr Informationen zu den implementierten Items. Der Einsatz von Scriptable-Objects spart Speicher, da alle Informationen bereits in einem einzigen Objekt gespeichert sind [6].
Unity bietet vorgefertigte Elemente für die Benutzeroberfläche an. So kann der Entwick-

ler per Drop-Down-Menü vorgefertigte Buttons, Textfelder und vieles mehr einfügen. Die Elemente müssen dann nur noch angepasst werden. Die Textfelder können so eingestellt werden, dass ihre Schriftgröße automatisch kleiner skaliert wird, falls die Flächen überschritten werden. Dynamische Funktionalitäten wie diese helfen dabei eine möglichst saubere Benutzeroberfläche zu erstellen [6].
Zum Speichern der relevanten Spieldaten wird die BinaryFormatter-Klasse genutzt. Diese Klasse bietet Methoden an, die die Serialisierung und die Deserialisierung von Spieldaten unterstützt. Dadurch kann der Entwickler zur Spiellaufzeit die Dateien generieren, speichern und laden. Diese Technik wird auch in *MineTrade* verwendet. Der verwendete Dateipfad wird in der Spielanleitung angezeigt.
Unity bietet zahlreiche Funktionalitäten an, jedoch muss der Entwickler in den meisten Fällen dennoch programmieren. Es werden hierfür die Programmiersprachen C# oder JavaScript genutzt. Beim Programmieren werden sogenannte Scripts erstellt. Diese Scripts können unterschiedlich eingesetzt werden. Einer der Einsatzmöglichkeiten sind die sogenannte MonoBehaviour-Scripts [6]. Diese Scripts vererben von der Klasse MonoBehaviour und erhalten dadurch Zugriff auf die Hauptmethoden Start() und Update(). Die Start()-Methode wird einmal bei der Instanziierung des Spiels und die Update()-Methode in Abhängigkeit zu einem zeitlichen Intervall ausgeführt. Dadurch kann der Entwickler Funktionen in diese Methoden geschickt einsetzen, um sie im richtigen Moment auszuführen. Die Start()-Methode wird häufig für die Initialisierung und Bereitstellung von Spielobjekten genutzt. Spielfunktionalitäten können in separate Skripts aufgeteilt werden. Dies unterstützt die Modularität der Spielarchitektur. Jedoch kann eine zu große Anzahl an Skripten die Überschaubarkeit der Spielarchitektur negativ beeinflussen. Daher sollten die Skripte stets kommentiert und sinnvoll bezeichnet werden. Der Einsatz von UML-Klassendiagrammen kann hierbei ebenfalls helfen.

3

Anforderungsanalyse

In diesem Kapitel werden die funktionalen und nicht-funktionalen Anforderung behandelt. Diese Anforderungen beziehen sich auf das Spiel, welches für diese Ausarbeitung entworfen und implementiert wurde. Der Abschnitt 3.1 befasst sich mit den funktionalen Anforderungen. Danach folgt der Abschnitt 3.2 mit den nicht-funktionalen Anforderungen.

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen beziehen sich auf die gewünschten Funktionalitäten die im Spiel verfügbar sein sollen. Alle Anforderungen wurden im Spiel implementiert und sie erfüllen ihre Funktion. Die meisten Anforderungen entsprechen in diesem Fall sämtliche Spielelemente, die den generellen Spielablauf ermöglichen.

FA 1: (Zielplattform) Das Spiel wird für das Betriebssystem Windows entwickelt und somit wird es auf dem PC spielbar sein.
FA 2: (Spielperspektive) Das Spiel ist zweidimensional und wird aus der Vogelperspektive gespielt. Diese Perspektive soll die Übersicht des Spielers unterstützen.
FA 3: (Interne Spielanleitung) Der Spieler erhält die Möglichkeit jederzeit eine umfassende Spielanleitung aufrufen zu können, die ihm beim Verstehen des Spiels helfen soll.
FA 4: (Spielmenü) In dem Spielmenü werden wichtige Spielfunktionen übersichtlich angezeigt.
FA 5: (Geld- und Entscheidungsanzeige) Es wird eine Geld- und Entscheidungsanzeige erstellt, die immer die aktuellen Werte anzeigen.
FA 6: (Korrektheit) Alle Interaktionen müssen korrekt ablaufen. Es darf insbesondere

beim Handeln keine Rechenfehler geben, da diese den Spielablauf manipulieren können.
FA 7: (Speicherfunktion) Der Spielablauf wird in Form von Prozessmodellen abgespeichert. Nach dem Ablauf einer Spielrunde kann der Spieler selbst entscheiden, ob das Prozessmodell archiviert werden soll.
FA 8: (Ladefunktion) Die Prozessmodelle sollen stets geladen und angezeigt werden können.
FA 9: (Fünf NPCs) Die fünf NPCs müssen alle implementiert werden, damit eine entsprechende Menge an Entscheidungsmöglichkeiten gewährleistet wird.
FA 10: (Robustheit) Das Spiel soll nicht abstürzen. Alle Interaktionen sind stabil implementiert. Bei allen Eingabefeldern wird die Eingabe korrekt verarbeitet. Ungültige Eingaben werden verhindert.
FA 11: (Inventar) Ein Inventar wird implementiert, damit der Spieler alle Items besitzen kann. Die Größe des Inventars reicht für den Besitz aller Items.
FA 12: (Tooltip) Ein Tooltip soll eingeblendet werden, wenn die Maus über ein Item positioniert wird. Das Tooltip zeigt dann alle relevanten Informationen zu diesem Item an. Dadurch erhält der Spieler einen sehr praktischen Informationszugang.
FA 13: (Fehlerfreiheit) Alle Daten, wie zum Beispiel die Ein- und Verkaufspreise der Items, sind korrekt eingetragen.
FA 14: (Neustartfunktion) Der Spieler ist in der Lage das Spiel neu zu starten, ohne die Applikation beenden zu müssen.

3.2 Nicht-Funktionale Anforderungen

Die nicht-funktionalen Anforderungen beziehen sich hauptsächlich auf die Bedienbarkeit, das Design und die Qualität des Spiels. Diese Anforderungen sollen insbesondere die Spielerfahrung positiv unterstützen und zusätzlich einen gewissen Spielspaß ermöglichen. Eine schlechte Bedienbarkeit und ein unverständliches Spieldesign kann einen negativen Einfluss auf die Spielerfahrung haben.

NFA 1: (Beschriftungen) Alle NPCs und ihre jeweiligen Benutzeroberflächen werden beschriftet, damit der Spieler sie zuordnen und einfacher verstehen kann.

NFA 2: (Hinweisfenster) Wenn ein Spieler beispielsweise eine Interaktion tätigt, aber eine bestimmte Bedingung nicht erfüllt, so wird ein Hinweisfenster mit der jeweiligen Begründung angezeigt.
NFA 3: (Übersichtlichkeit) Die Spielwelt soll übersichtlich aufgebaut werden.
NFA 4: (Vorteilhafte Programmfehler) Mögliche Programmfehler, die den Spieler auf eine unfaire Art unterstützen könnten, werden verhindert.
NFA 5: (Benutzerfreundlichkeit) Alle grafischen Benutzeroberflächen müssen möglichst intuitiv nutzbar sein.
NFA 6: (Klare Zielvorgabe) Das Ziel des Spiels soll präzise formuliert werden. Dabei muss kein genauer Spielweg genannt werden, da dies der Spieler selbstständig ermitteln muss.
NFA 7: (Authentische Spielwelt) Die Spielwelt soll möglichst authentisch gestaltet werden. Dabei sollen die NPCs mit Spieler textuell sprechen.
NFA 8: (Eventlogs) Spielinteraktionen die im Verborgenem ablaufen, wie zum Beispiel die Minen- und Banditenaufträge, sollen Eventlogs generieren, die den internen Ablauf textuell beschreiben.

4

Game Design und Implementierung von MineTrade

Dieses Kapitel behandelt das Spiel, das im Rahmen dieser Arbeit implementiert wurde. Das Spiel heißt inoffiziell *MineTrade* und wurde mit der *Unity* Spiel-Engine entwickelt. Bei diesem Spiel wurde es ermöglicht Prozessmodelle während dem Spielen zu generieren. Diese Prozessmodelle ermöglichen dem Spieler seine Entscheidungen zu analysieren. Da diese Arbeit den Einsatz von Prozessmodellen für die Entscheidungsanalyse behandelt, wurde bei der Entwicklung des Spielkonzepts ein Spiel entworfen, bei dem der Spieler einige Entscheidungen treffen muss. Des Weiteren sollte der Spieler möglichst freie Entscheidungsmöglichkeiten haben, anstatt einen vorgegebenen Spielweg zu erhalten. Daher wurden fünf NPCs (engl. für non-player character) entworfen, die unterschiedliche Handlungsfunktionalitäten bieten. Ebenfalls wichtig war es das Spielkonzept möglichst leicht verständlich zu halten.

4.1 Spielkonzept

MineTrade ist ein 2D Strategiespiel, bei dem der Spieler eigenständig handeln soll, um möglichst viel Profit zu machen. Der Spieler besitzt zu Spielbeginn ein festes Startkapital. Nach 30 Entscheidungen endet eine Spielrunde. Das Kapital am Ende einer Spielrunde gibt an wie gut die jeweilige Spielrunde ablief. Je höher das Kapital, desto besser ist die Spielrunde. Die Entscheidungen, die der Spieler trifft, sollen stets relevant und bedeutungsvoll sein [7]. Es gibt deterministische und zufallsabhängige Entscheidungen. Der Spieler kann während dem Spielen Entscheidungen aus vorherigen Spielrunden laden

und ansehen. Dies soll den Spieler dabei unterstützen möglichst gute Entscheidungen zu treffen. Die Entscheidungen werden im Form eines Prozessmodells dargestellt.
In der Abbildung 4.1 ist das Spiel im Startzustand zu sehen. Die Währung in *MineTrade* heißt Chronos und wird im Spiel durch einem C abgekürzt. In der rechten oberen Ecke der Abbildung 4.1 wird die aktuelle Menge der Chronos und die Anzahl der getroffenen Entscheidungen angezeigt. Des Weiteren sind die NPCs und ihre Standorte zu erkennen. Ebenfalls zu sehen ist das leere Inventar des Spielers. Der Spieler selbst ist im Spiel nicht abgebildet, sondern er steuert lediglich den Mauszeiger. In der ersten Version des Spielkonzepts sollte der Spieler sich im Dorf frei bewegen können. Diese Idee wurde jedoch verworfen, als sich das Laufen nach einer kurzen Spielzeit als mühsam und unnötig erwiesen hat.
Die Positionierung der NPCs wurde nach ihren Eigenschaften zusammengesetzt. So ist beispielsweise die Mine und der Händler mittig, da sie als Quelle für Rohstoffe dienen. Das Auktionshaus und die Banditen sind nebeneinander, weil beide zufallsbedingte Interaktionen besitzen.

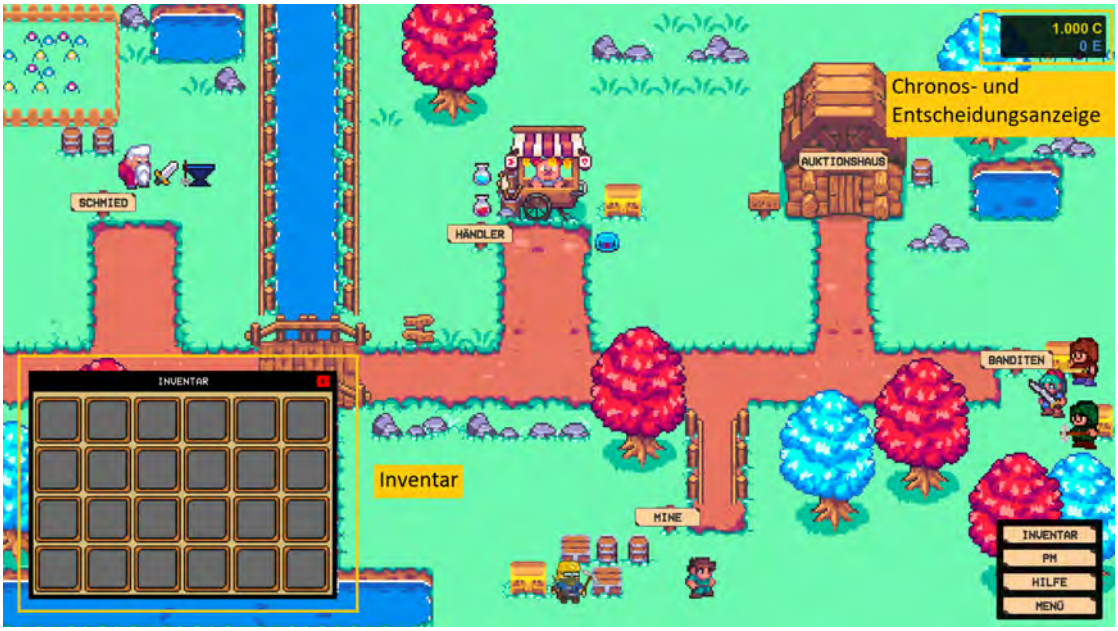

Abbildung 4.1: Ein Screenshot von *MineTrade*

4.2 Die fünf NPCs

Dieser Abschnitt setzt sich mit den fünf NPCs auseinander. Die NPCs sollen sich möglichst voneinander unterscheiden, damit sie alle durch ihre Einzigartigkeit eine wichtige Rolle im Spiel erhalten. Daraus folgt, dass jeder NPC eigene Funktionalitäten besitzt. So kann man beispielsweise beim Händler und bei der Mine Rohstoffe erwerben, aber bei der Mine ist der Erwerb zufallsbedingt. Die Komplexität des Spieles soll relativ gering gehalten werden, daher existieren fünf überschaubare NPCs. In der Abbildung 4.2 befindet sich eine zusammenfassende Übersicht der NPCs.

NPC - Bezeichnung	Funktionen und Eigenschaften
Schmied	➔ Kann aus Rohstoffen Schwerter und Rüstungen herstellen ➔ Schwerter und Rüstungen sind wertvoller als Rohstoffe ➔ Verlangt für jeden Auftrag eine bestimmte Gebühr ➔ Gebühr steigt mit der Seltenheit des Rohstoffs ➔ Keine zufallsbedingten Interaktionen
Händler	➔ Verkauft Rohstoffe für feste Preise ➔ Kauft Items von jeder Art und Seltenheit ab ➔ Verlangt keine Gebühren ➔ Ein- und Verkauf von mehreren Items gleichzeitig möglich ➔ Keine zufallsbedingten Interaktionen
Mine	➔ Dient als alternative Rohstoffquelle ➔ Ermöglicht eine günstige Beschaffung von sehr seltenen Rohstoffen ➔ Basiert jedoch auf einem Zufallssystem
Auktionshaus	➔ Dient als alternative Verkaufsmöglichkeit ➔ Jedoch ist die Bezahlung einer Gebühr notwendig ➔ Seltenheit der Gegenstände muss mindestens 4 sein ➔ Der Spieler bestimmt das Startgebot ➔ Auktion basiert auf einem Zufallssystem
Banditen	➔ Beschaffen seltene Schwerter und Rüstungen ➔ Gebühr ist relativ hoch ➔ Basieren auf einem Zufallssystem

Abbildung 4.2: Übersicht der NPCs

4.2.1 Der Schmied

Der Schmied befindet sich auf der Spielwelt in der linken oberen Ecke. Beim Schmied kann der Spieler aus Rohstoffen Schwerter und Rüstungen herstellen lassen. Für jeden Service verlangt der Schmied eine Gebühr. Diese Gebühr ist abhängig von dem benutzten Rohstoff. Je höher die Seltenheit des Rohstoffs, desto höher ist die Gebühr. Der Zweck des Schmieds ist es durch das Schmieden von Schwertern und Rüstungen Profit zu erwerben. Denn Schwerter und Rüstungen sind wertvoller als ihre Rohstoffform. In der Abbildung 4.3 wird die Benutzeroberfläche des Schmieds angezeigt.

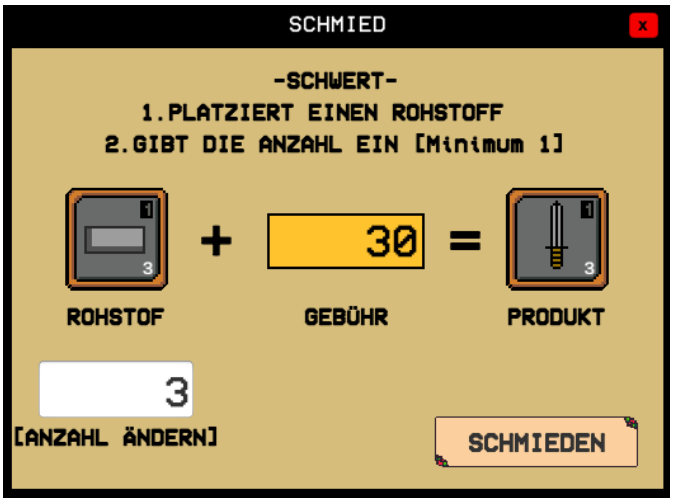

Abbildung 4.3: Benutzeroberfläche zum Schmieden von Schwertern

Es werden drei Eisen Rohstoffe verwendet und daraus ergeben sich drei Eisenschwerter. Die Gebühr beträgt hierbei 30 Chronos. Die Benutzeroberfläche für das Schmieden von Rüstungen ist auf die gleiche Art aufgebaut. Jedoch benötigt der Spieler für Rüstungen jeweils zwei Rohstoffe. Falls er versehentlich in das Inputfeld eine ungerade Zahl einträgt, so erhält er die ungenutzten Rohstoffe automatisch zurück.
Die Erfolgsrate des Schmieds beträgt stets 100%. Ursprünglich war die Überlegung die Erfolgsrate zufallsbedingt zu implementieren. Jedoch werden bereits zufallsbedingte Interaktionen bei den anderen NPCs verwendet und ein übermäßiger Gebrauch von Zufall würde den Determinismus im Spiel negativ beeinflussen. Des Weiteren kann der

Verlust von wertvollen Gegenständen für den Spieler frustrierend und demotivierend sein. Daher ist die Erfolgsrate beim Schmied stets gesichert. Es ist empfehlenswert Rohstoffe soweit möglich in Rüstungen und ansonsten in Schwerter zu verarbeiten, bevor sie verkauft oder versteigert werden. Denn Rüstungen haben den höchsten Basisverkaufspreis. Diese Information muss der Spieler eigenständig feststellen, jedoch wird diese Information nicht geheimgehalten. Denn der Spieler kann mit Hilfe des ToolTips aus Abschnitt 4.3 die Verkaufspreise der Items einsehen und sie dadurch miteinander vergleichen.
Die Implementierung des Schmieds ist in wenigen Schritten umsetzbar. Es werden lediglich Methoden für arithmetische Berechnungen erstellt. Anhand dieser Methoden kann beispielsweise die Gebühr abhängig von der Anzahl der Rohstoffe und ihrer Seltenheit berechnet werden. Bei der Methode des Schmieds werden die Rohstoffe des Spielers aus seinem Inventar gelöscht und die neuen Items hinzugefügt. Des Weiteren muss die Gebühr vom Spieler abgezogen und die Entscheidung gespeichert werden. Der Speichervorgang von Entscheidungen wird in Abschnitt 4.4 genauer beschrieben.

4.2.2 Der Händler

Der Händler bietet zwei Möglichkeiten der Interaktion. Die erste Interaktion ist die Einkaufsfunktion. Der Spieler kann nämlich für Chronos Rohstoffe beim Händler erwerben. Die zweite Interaktion ist die Verkaufsfunktion. Der Spieler kann jede Art von Item mit beliebiger Seltenheit beim Händler verkaufen. Die Abbildung 4.4 zeigt die jeweiligen Benutzeroberflächen für das Ein- und Verkaufen von Items.

Abbildung 4.4: Die Händlerbenutzeroberfläche zum Ein- und Verkaufen

Es wird dem Spieler ermöglicht drei verschiedene Items mit jeweils der maximalen Anzahl von 99 ein- und verkaufen zu können. Dadurch kann der Spieler durch einen geschickten Umgang Entscheidungen sparen, da dies in einer einzigen Interaktion geschieht und somit als eine einzige Entscheidung zählt. Diese Eigenschaft ist auch der Hauptvorteil des Händlers. Denn im Vergleich kann man im Auktionshaus nur ein

Item pro Entscheidung verkaufen. Ein weiterer Vorteil des Händlers ist, dass er keinerlei Gebühren anfordert und somit immer als Verkaufsoption zur Verfügung steht. Der Einkauf von Rohstoffen und der direkte Verkauf dieser Rohstoffe führt immer zu Verlust, da der Einkaufspreis höher als der Verkaufspreis ist. Daher sollten die Rohstoffe möglichst zu Rüstungen oder Schwerter geschmiedet werden.
Bei der Implementierung des Händlers ist es notwendig alle Preise zu den Items aufrufen zu können. Dies wurde durch den Einsatz von Scriptable-Objects ermöglicht, welche im Abschnitt 2.2 genauer erklärt wurden. Da es möglich ist die Ein- und Verkaufspreise der Items abrufen zu können, müssen lediglich nur noch Methoden für die Berechnung der Gesamtsumme implementiert werden. Bei dieser Methode wird die Anzahl der Items mit deren Ein- oder Verkaufspreis multipliziert und diese Werte miteinander summiert. Daraus ergibt sich dann die jeweilige Gesamtsumme. Auch beim Händler findet im Anschluss der Interaktion ein Itemaustausch statt. Beim Einkaufen wird dem Spieler die Gesamtsumme abgezogen und die eingekauften Items in seinem Inventar hinzugefügt. Beim Verkaufen werden ihm die Items abgenommen und die Gesamtsumme zu seinem aktuellen Chronosstand hinzu addiert. Im Anschluss wird dann die Entscheidung gezählt und gespeichert.

4.2.3 Die Mine

Die Mine dient als eine alternative Rohstoffquelle. Der Spieler kann hierbei die Rohstoffe nicht direkt erwerben. Stattdessen kann er eine von ihm ausgewählte Anzahl an Minenarbeitern bezahlen und sie dann zum Sammeln von Rohstoffen entsenden. Das Sammeln der Rohstoffe basiert auf einem Zufallssystem. Es wird für jeden Auftrag eine Mine instantiiert. Dabei wird im Programmcode eine feste Anzahl für jeden Rohstoff festgelegt. Neben den Rohstoffen existieren auch leere Felder. Ein leeres Feld entspricht dem Fehlschlag des Minenarbeiters. Findet ein Minenarbeiter ein leeres Feld, so geht er mit leeren Händen aus der Mine. Der Spieler kann bis zu 99 Minenarbeiter pro Auftrag einsetzen. In der Abbildung 4.5 wird die Benutzeroberfläche der Mine angezeigt. Der Spieler trägt die Anzahl der Arbeiter ein und daraus wird dann die Gebühr berechnet. Ein Minenarbeiter kostet 200 Chronos pro Auftrag.

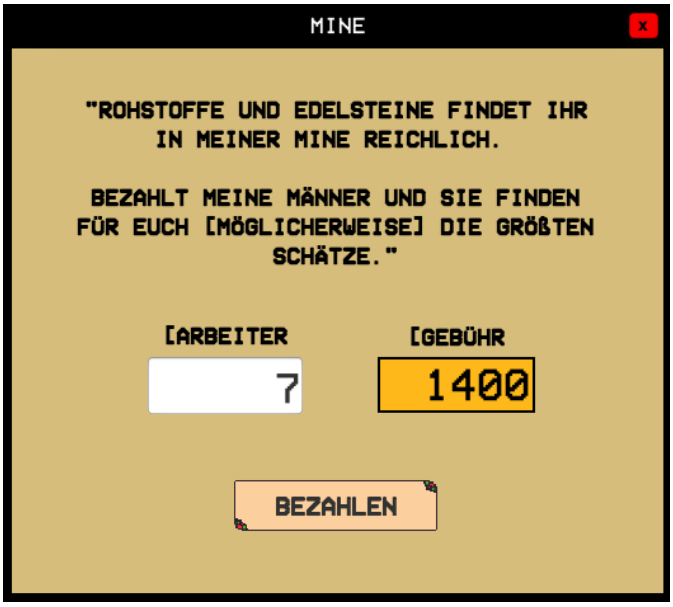

Abbildung 4.5: Die Benutzeroberfläche der Mine

Wenn der Spieler einen Minenauftrag bezahlt und gestartet hat, dann wird dynamisch ein Eventlog erstellt. Der Eventlog ist in der Abbildung 4.6 zu sehen.

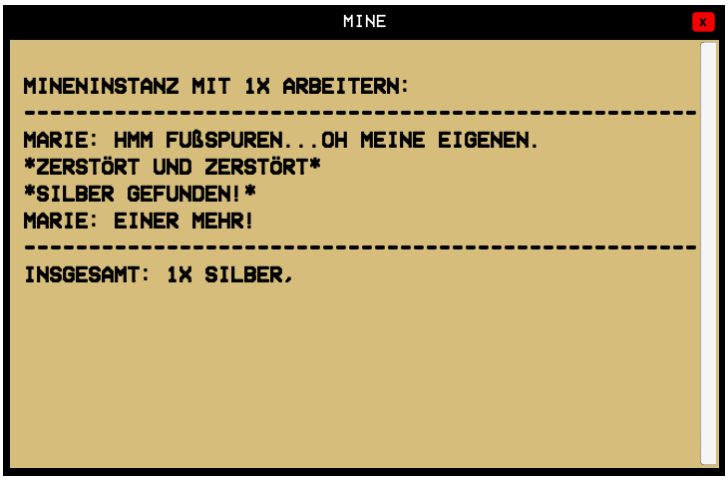

Abbildung 4.6: Eventlog der Mine

In diesem Eventlog wird in einer textuellen Form die Interaktion beschrieben. Dadurch erhält der Spieler wichtige Informationen zur Mineninstanz. Zusätzlich wird der Eventlog mit humorvollen Texten ausgeschmückt, die zufällig generiert werden.
Bei der Implementierung der Mine steht die Erstellung eines Zufallssystems im Vordergrund. Es wird ein Array generiert der alle Rohstoffe mit ihrer jeweiligen Anzahl beinhaltet. Bei jeder Mineninstanz ziehen die Minenarbeiter nacheinander eine zufällige Zahl. Die Zahl ist im Intervall aus null und der aktuellen Größe des Arrays. Diese zufällige Zahl wird dann als Index genutzt. Der jeweilige Minenarbeiter nimmt den Arrayinhalt des jeweiligen Index heraus und reduziert damit die Arraygröße um eins. Dadurch erhöht sich die Wahrscheinlichkeit für den nächsten Minenarbeiter eines der Rohstoffe zu erhalten, da die Gesamtanzahl kleiner wird. Daher lohnt es sich mehrere Minenarbeiter in einem Minenauftrag einzusetzen.
Der Eventlog wird durch die Verkettung mehrerer Strings generiert. Die zufälligen Texte entstehen durch die zufallsbedingte Auswahl von Zeichenketten. Dazu gehören die Namen der Minenarbeiter und die Aktionen, die sie tätigen. Am Ende des Eventlogs wird der Gesamtfund aufgezählt, um Übersicht für den Spieler zu schaffen. Ein Minenauftrag zählt als eine Entscheidung und wird nach der Ausführung gespeichert.

4.2.4 Das Auktionshaus

Das Auktionshaus dient als alternative Verkaufsmöglichkeit. Jedoch müssen beim Auktionshaus zwei Bedingungen erfüllt sein, um ein Item versteigern zu können. Die erste Bedingung ist die Seltenheit des Items. Die Seltenheit muss mindestens vier betragen, sonst kann die Auktion nicht gestartet werden. Die zweite Bedingung ist die Bezahlung einer Gebühr. Die Gebühr errechnet sich aus einem prozentualen Anteil des Verkaufspreises des jeweiligen Item. Daraus folgt, dass aus einem höherem Verkaufspreis eine höhere Gebühr bezahlt werden muss. Die Gebühr wurde eingeführt damit das Auktionshaus ausbalanciert wird. Denn wäre das Auktionshaus zu rentabel, so wäre der Händler nutzlos.

In der Abbildung 4.7 wird die Benutzeroberfläche des Auktionshauses angezeigt.

Abbildung 4.7: Die Benutzeroberfläche des Auktionshauses

Sobald die Auktion gestartet wird, entsteht ein Eventlog. Dieser Eventlog ist ähnlich aufgebaut wie bei der Mine. Der Eventlog befindet sich in der Abbildung 4.8.

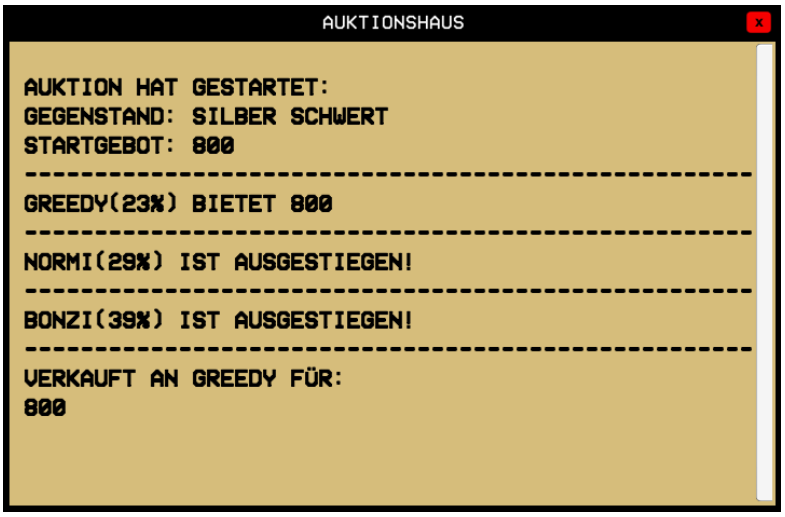

Abbildung 4.8: Eventlog des Auktionshauses

Die Auktion basiert auf einem Zufallssystem. Es wurden drei Auktionäre implementiert. Diese Auktionäre heißen Greedy, Normi und Bonzi. Ihre Namen beziehen sich hierbei auf ihre Basiswahrscheinlichkeit bei einer Auktion mit zu bieten. Greedy besitzt eine geringere Basiswahrscheinlichkeit als Normi und Bonzi besitzt die höchste Basiswahrscheinlichkeit. Das Prinzip des Auktionshaus beruht darauf, dass sich die Auktionäre gegenseitig überbieten. Daraus folgt, dass der Spieler von der hohen Basiswahrscheinlichkeit von Bonzi nur profitiert, wenn mindestens einer der beiden anderen Auktionäre mit bieten. Dieses System soll dem Spieler ermöglichen ein Item für einen sehr hohen Preis verkaufen zu können. Denn der Spieler kann das Startgebot selbst bestimmen. Dadurch kann er potentiell pro Item sehr viel Profit erhalten. Jedoch trägt er ebenfalls das Risiko, dass keiner der Auktionäre mit bietet. Wenn dies der Fall ist, so erhält der Spieler sein Auktionsitem zurück, aber er verliert die Gebühr.
Die Auktionäre wurden so implementiert, dass sie in einer festen Reihenfolge versuchen zu bieten. Es beginnt immer Greedy, danach folgt Normi und als letztes kommt Bonzi dran. Danach ist erneut Greedy an der Reihe. Je höher das Gebot wird desto geringer wird die Wahrscheinlichkeit, dass die Auktionäre weiter bieten. Dies wurde durch eine Methode ermöglicht, die durch die Differenz zwischen dem aktuellem Gebot und dem eigentlichen Verkaufswert des Items, die Basiswahrscheinlichkeiten reduziert. Jedoch ist es auch möglich ein Startgebot von null festzulegen. Wenn dies der Fall ist, wird die Basiswahrscheinlichkeit der Auktionäre erhöht, da keiner der Auktionäre sich ein Schnäppchen entgehen lassen will. Sobald das Gebot den eigentlichen Verkaufswert des Items erreicht oder übersteigt, beginnt die Reduzierung der Basiswahrscheinlichkeiten.

4.2.5 Die Banditen

Die Banditen dienen als Quelle für seltene Schwerter und Rüstungen. Sie bieten somit eine Möglichkeit an mit wenigen Entscheidungen hohen Profit zu generieren. Dabei besteht aber ein Verlustrisiko. Denn ein einzelner Bandit kostet 1500 Chronos. Die Banditen können beim Plündern fehlschlagen. Wenn das passiert, dann verliert der Spieler seine eingesetzten Chronos. Somit symbolisieren die Banditen eine schattige Geschäftsmöglichkeit. Die Banditen befinden sich rechts am Rande des Dorfes. Die Abbildung 4.9 zeigt die Benutzeroberfläche der Banditen. Ähnlich wie bei der Mine, trägt der Spieler die Anzahl der Banditen ein. Dabei wird dann die notwendige Gebühr berechnet. Ist der Spieler in der Lage diese Gebühr zu bezahlen, so startet die Banditeninstanz.

Abbildung 4.9: Die Benutzeroberfläche der Banditen

Bei der Banditeninstanz plündert jeder Bandit ein Dorf und hat dabei bestimmte Wahrscheinlichkeiten Schwerter und Rüstungen zu finden. In einem Eventlog (siehe Abbildung 4.10) wird das Ereignis dokumentiert, damit der Spieler einsehen kann was geschehen ist. Für die Banditen wurde ein neues Zufallssystem implementiert. Es wird

Abbildung 4.10: Eventlog der Banditen

in diesem Zufallssystem doppelt gewürfelt. Der erste Würfel gibt an ob der jeweilige Bandit ein Schwert, eine Rüstung oder beides finden kann. Der zweite Würfel gibt an welches Schwert oder welche Rüstung der Bandit findet. Dabei kann der Bandit scheitern und dadurch nichts erhalten. Die Wahrscheinlichkeiten kann der Spieler nicht einsehen. Er muss durch mehrfache Versuche selbstständig ermitteln, wie viele Banditen als lohnenswert erscheinen.

4.3 Inventar, Items und Tooltip

In der Abbildung 4.11 wird das Inventar und das Tooltip angezeigt.

Abbildung 4.11: Inventar und Tooltip

In *MineTrade* existieren insgesamt 24 Items. Dazu gehören acht Rohstoffe, acht Schwerter und acht Rüstungen. Die Items werden in der Abbildung 4.12 angezeigt. Das Inventar bietet genug Platz um alle Items enthalten zu können. Die Items wurden als Scriptable-Objects implementiert, damit zu jedem Item Informationen gespeichert werden können. Diese Informationen kann dann die Tooltip-Klasse abrufen, sodass sie im Spiel angezeigt werden können. Das Tooltip wird eingeblendet sobald der Spieler den Mauszeiger über ein Item positioniert. Das Konzept des Tooltip ist aus virtuellen Rollenspielen bekannt. Dazu gehören Spiele wie Path of Exile, World of Warcraft und Diablo. Der Vorteil des Tooltips ist der schnelle Zugang zu relevanten Informationen. In kürzester Zeit können alle notwendigen Informationen bezüglich Items eingeblendet werden. Das ist besonders wichtig in *MineTrade*, da der Spieler in der Lage sein muss den Verkaufswert eines Items vor einer Auktion überprüfen zu können. Das Tooltip wird in *Unity* durch eine vorgefertigte MouseEnter-Methode implementiert. Diese Methode wird ausgelöst sobald der Mauszeiger über einem Item positioniert wird. Es werden dann die Informationen des jeweiligen Item abgerufen und in das Tooltip eingesetzt.

Items wurden durch eine eigenständig erstellte Item-Klasse implementiert. Das erleichterte die Implementierung des Inventars, da das Inventar letztendlich eine Liste mit dem Typ Item ist. In diese Liste können dann beliebig Items eingefügt und wieder entfernt werden. Die Item-Klasse ist abstrakt und wird von den Rohstoff-, Schwert- und Rüstung-Klassen vererbt.

Erzliste			
Bezeichnung	**Kaufpreis (Händler)**	**Verkaufspreis (Händler)**	**Seltenheit**
Eisen	60	30	1
Stahl	100	50	2
Bronze	200	100	3
Silber	400	200	4
Gold	700	350	5
Smaragd	1400	700	6
Diamant	2000	1000	7
Rubin	4000	2000	8
Schwertliste			
Bezeichnung	**Verkaufspreis (Händler)**	**Seltenheit**	
Eisenschwert	60	1	
Stahlschwert	110	2	
Bronzeschwert	240	3	
Silberschwert	520	4	
Goldschwert	980	5	
Smaragdschwert	2100	6	
Diamantschwert	3500	7	
Rubinschwert	8000	8	
Rüstungliste			
Bezeichnung	**Verkaufspreis (Händler)**	**Seltenheit**	
Eisenrüstung	150	1	
Stahlrüstung	275	2	
Bronzerüstung	600	3	
Silberrüstung	1300	4	
Goldrüstung	2450	5	
Smaragdrüstung	5250	6	
Diamantrüstung	8750	7	
Rubinrüstung	12500	8	

Abbildung 4.12: Tabelle mit allen Items

4.4 Prozessmodellanalyse

In der Abbildung 4.13 wird die Benutzeroberfläche der Prozessmodellanalyse angezeigt.

Abbildung 4.13: Die Benutzeroberfläche der Prozessmodellanalyse

Links befindet sich eine Liste mit einer Scrollbar und zwei Buttons. Diese Liste beinhaltet immer die Datei mit den Prozessmodellen des aktuellen Spieldurchlaufs. Wenn der Spieler vorherige Prozessmodelle laden und anzeigen möchte, so kann er durch den linken Button die Liste aktualisieren. Dann erscheinen die vorherigen Dateien mit den Prozessmodellen. Anhand der jeweiligen Checkbox kann sich dann der Spieler die jeweiligen Dateien aussuchen und mit dem Button „Auswahl aktualisieren“ seine Auswahl anzeigen lassen. Die Prozessmodelle werden auf dem rechten Bereich angezeigt. Dieser Bereich kann mit der Maus bewegt werden, damit alle Elemente angesehen werden können. Falls das Layout sich ungewollt verschiebt, kann der Spieler mit dem Benutzen des entsprechenden Buttons das Layout neu ordnen. Dieser Button befindet sich rechts unten in der Ecke.

Der Zweck dieser Benutzeroberfläche ist die Darstellung der Prozessmodelle. Hier kann der Spieler seine Entscheidungen und deren Resultate miteinander vergleichen. Die Implementierung dieser Funktionalität besteht aus zwei Teilen.
Im ersten Teil müssen die Entscheidungen und alle notwendigen Informationen dieser Entscheidung direkt nach ihrer Ausführung gespeichert werden. Wichtig hierbei ist die Erstellung von Entscheidungskollektionen. Diese Kollektionen wurden im Programmcode als ActionUnitContainer bezeichnet. Eine ActionUnit entspricht einer einzigen Entscheidung und ein ActionUnitContainer beinhaltet mehrere ActionUnits. Durch dieses System wird automatisch die chronologische Folge der Entscheidungen gespeichert, da die ActionUnitContainer Listen sind. Eine Liste beinhaltet Elemente und jedes dieser Elemente erhält einen Index, der die Position in der Liste angibt. Demnach ist die ActionUnit mit dem Index null die aller erste Entscheidung und die ActionUnit mit dem Index eins die zweite Entscheidung und so weiter.
Der zweite Teil befasst sich mit der Serialisierung dieser ActionUnitContainer, damit sie als Dateien gespeichert werden können. Wie im Abschnitt 2.2 bereits erklärt bietet *Unity* bzw. C# die BinaryFormatter Klasse an. Anhand dieser Klasse kann man im Programmcode einen Dateipfad aussuchen, die Dateinamen festlegen und Spielinformationen serialisieren und deserialisieren. Es ist an dieser Stelle wichtig zu erwähnen, dass die Klasse die den BinaryFormatter benutzt als statische Klasse deklariert werden sollte, da diese Klasse keine Instanzen bilden soll, sondern einzigartig bleiben muss. Ebenso ist es empfehlenswert die jeweiligen Methoden als statisch zu deklarieren, damit sie in den anderen Klassen ohne Instanz verwendet werden können.
Die Darstellung der Prozessmodelle wurde durch den Einsatz von VerticalGridLayouts und HorizontalGridLayouts ermöglicht. Ein VerticalGridLayout positioniert die hinzugefügten Elemente untereinander und ein HorizontalGridLayout positioniert sie nebeneinander. In diesem Fall wurde ein VerticalGridLayout erstellt und dieses Layout enthält mehrere HorizontalGridLayouts. Die HorizontalGridLayouts enthalte alle Entscheidungen der Prozessmodelle. Dadurch werden Prozessmodelle untereinander und Entscheidungen nebeneinander angezeigt.

4.5 Anforderungsabgleich

In diesem Kapitel werden die Anforderungen aus dem Kapitel 3 abgeglichen. Beim Abgleich wird festgelegt, ob die jeweilige Anforderung erfüllt wurde. Dieser Abgleich umfasst sowohl die funktionalen, als auch die nicht-funktionalen Anforderungen.

4.5.1 Abgleich der funktionalen Anforderungen

Funktionale Anforderung	Erfüllt	Begründung
FA 1: Zielplattform	Ja	Das Spiel ist auf Computer mit Windows spielbar.
FA 2: Spielperspektive	Ja	Die Spielperspektive entspricht der gewünschten 2D-Vogelperspektive.
FA 3: Interne Spielanleitung	Ja	Eine interne Spielanleitung, mit allen wichtigen Informationen, ist vorhanden.
FA 4: Spielmenü	Ja	Ein Spielmenü ist im Spiel verfügbar.
FA 5: Geld- und Entscheidungsanzeige	Ja	Eine entsprechende Anzeige wurde implementiert.
FA 6: Korrektheit	Ja	Alle Interaktionen wurden gründlich auf Rechenfehler und Korrektheit geprüft.
FA 7: Speicherfunktion	Ja	Der Spieler kann Prozessmodelle speichern oder verwerfen.
FA 8: Ladefunktion	Ja	Der Spieler kann Prozessmodelle laden und anzeigen lassen.
FA 9: Fünf NPCs	Ja	Alle fünf NPCs und ihre Funktionalitäten wurden implementiert.
FA 10: Robustheit	Ja	Das Spiel ist selbst nach stundenlangen Spielphasen nicht abgestürzt. Die Eingabefelder sind gegen ungültige Eingaben abgesichert.
FA 11: Inventar	Ja	Das Inventar des Spielers wurde implementiert. Die Größe des Inventars umfasst alle existierende Items.
FA 12: Tooltip	Ja	Ein Tooltip für die Anzeige von Informationen wurde implementiert.
FA 13: Fehlerfreiheit	Ja	Alle Daten wurden nach der Implementierung auf Fehler kontrolliert.
FA 14: Neustartfunktion	Ja	Der Spieler kann das Spiel neu starten.

Abbildung 4.14: Abgleich der funktionalen Anforderungen

4.5.2 Abgleich der nicht-funktionalen Anforderungen

Nicht-funktionale Anforderung	Erfüllt	Begründung
NFA 1: Beschriftungen	Ja	Alle NPCs und ihre Benutzeroberflächen wurden mit hilfreichen Informationen beschriftet.
NFA 2: Hinweisfenster	Ja	Ein Hinweisfenster wird angezeigt, wenn eine bestimmte Interaktion nicht ausgeführt werden kann.
NFA 3: Übersichtlichkeit	Ja	Die Spielwelt wurde übersichtlich aufgebaut.
NFA 4: Vorteilhafte Programmfehler	Ja	Vorteilhafte Programmfehler wurden verhindert.
NFA 5: Benutzerfreundlichkeit	Ja	Die Benutzerfreundlichkeit wird durch eine verständliche Benutzeroberfläche unterstützt.
NFA 6: Klare Zielvorgabe	Ja	Die Zielvorgabe wird in einem spielinternen Handbuch erklärt.
NFA 7: Authentische Spielwelt	Ja	Die Spielwelt ist authentisch. Die NPCs sprechen mit dem Spieler.
NFA 8: Eventlogs	Ja	Die Eventlogs werden generiert und angezeigt.

Abbildung 4.15: Abgleich der nicht-funktionalen Anforderungen

5

Entscheidungsanalyse von drei Spielstrategien

In diesem Kapitel werden drei mögliche Spielstrategien vorgestellt und analysiert. Eine Spielrunde besteht aus 30 Entscheidungen, die der Spieler trifft. Bei der ersten Entscheidung aller Spielrunden muss der Spieler sich zwischen dem Händler und der Mine entscheiden. Alle anderen NPCs sind in der ersten Entscheidung nicht relevant, da der Spieler zu Beginn nicht genug Chronos bzw. Items besitzt. Ab der zweiten Entscheidung hat der Spieler, abhängig von dem Resultat seiner ersten Entscheidung, zahlreiche Entscheidungsmöglichkeiten. Die Kombinationsmöglichkeit der Entscheidungen erhöht sich daher mit dem Fortschreiten der Spielrunde. Aus einer kombinatorischen Perspektive betrachtet, gibt es somit zahlreiche Kombinationsmöglichkeiten aus Entscheidungen. Durch Empirie und Logik können gezielte Spielstrategien konstruiert werden. Die folgenden drei Spielstrategien dienen als Ansätze und sind nicht notwendigerweise optimiert. Generell basieren die Strategien auf empirischen Erkenntnissen und mathematischen Denkweisen, die während dem Spielen entstanden sind. Bei der Suche nach Spielstrategien hat der Einsatz der Prozessmodelle stark mitgeholfen, da die Resultate der Spielrunden miteinander verglichen werden können. Die ersten zwei Spielstrategien unterscheiden sich stark, wobei die dritte Spielstrategie eine Kombination aus den ersten beiden Spielstrategien darstellt.

5.1 Deterministische Spielstrategie

Im Kern besteht diese Strategie darin, nur deterministische Entscheidungen zu treffen. Es werden somit alle NPCs mit zufallsbedingten Handlungen vermieden. Deshalb interagiert man bei dieser Strategie nur mit dem Händler und dem Schmied. Die grundlegende Problemstellung dieser Strategie ist es den optimalen Handlungsvorgang zu finden. Der Ansatz für die Problemlösung basiert auf der Grundlage von sogenannten Greedy-Algorithmen [8]. Greedy-Algorithmen haben das Ziel Optimierungsprobleme zu lösen [8]. Sie finden anhand von verfügbaren Informationen sukzessiv eine optimale Lösung [8]. Das Greedy-Prinzip hat einen relativ geringen Aufwand und zusätzlich das Potential eine optimale Lösung zu finden [8]. Generell benutzt man beim Greedy-Prinzip eine Gewichtsfunktion. Diese Gewichtsfunktion hat einen Wert w und es soll eine Lösung konstruiert werden, die den Wert w maximiert [8]. Das Greedy-Prinzip kann auch bei *MineTrade* angewandt werden. Um es anwenden zu können, muss der Spieler zuerst verfügbare Informationen sammeln. Beim Händler kann der Spieler anhand des Tool-Tips alle Rohstoffkosten und deren Verkaufspreise ansehen. Der Spieler kann auch die Schwert- und Rüstungsinformationen beim Schmied einsehen, wenn er den jeweiligen Rohstoff einreicht.
Aus diesen Informationen kann der Spieler neues Wissen erlangen. Der Rüstungsverkaufspreis ist beispielsweise höher als der Schwertverkaufspreis. Daraus folgt, dass es tendenziell lohnenswerter ist Rüstungen statt Schwerter schmieden zu lassen. Ein weiteres Beispiel für den erweiterten Wissensstand durch die Informationssammlung, ist der steigende Wert der Rohstoffe und dessen Schwert- und Rüstungsprodukte.
In der Abbildung 5.1 kann man den Profitunterschied zwischen einem Silberschwert und einer Silberrüstung sehen. Im ersten Entscheidungspfad wurde eine Silberrüstung geschmiedet und verkauft. Der Profit beim ersten Entscheidungspfad beträgt 440 Chronos und beim zweiten Entscheidungspfad nur 90 Chronos. Anhand der gesammelten Informationen und dem Greedy-Prinzip ergibt sich nun die Spielstrategie. Es sind bei dieser Strategie stets drei Faktoren zu beachten. Der erste Faktor besagt, dass der Handel mit Rohstoffen der möglichst höchsten Seltenheit am rentabelsten ist. Daraus folgt, dass sofern möglich, die teuren Rohstoffe gekauft und geschmiedet werden sollten. Der zwei-

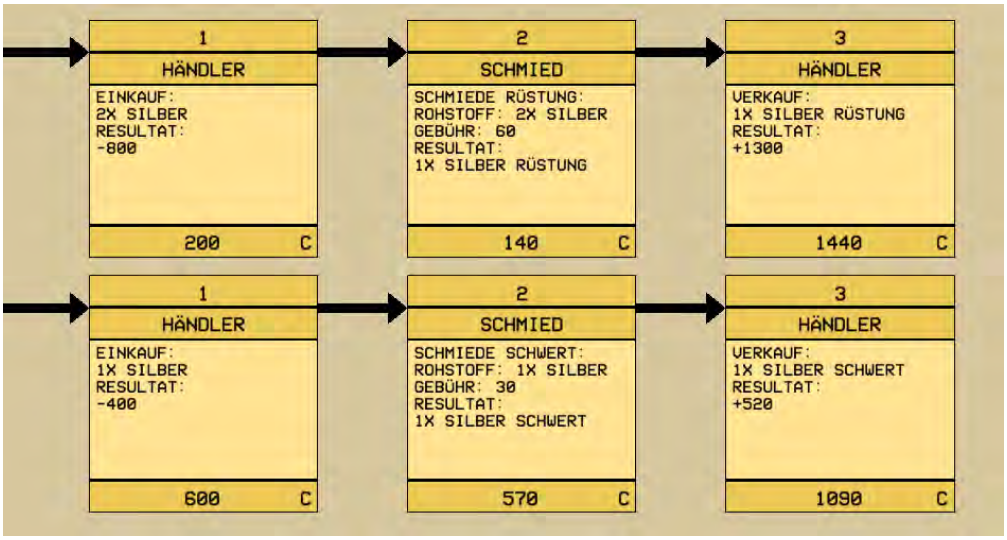

Abbildung 5.1: Profitunterschied zwischen Silberschwert und Silberrüstung

te Faktor besagt, dass der Handel mit Rüstungen rentabler ist als der mit Schwertern. Der dritte Faktor gibt an, dass es am sinnvollsten ist mit möglichst vielen Produkten zu handeln. Das bezieht sich auf den Einkauf, Verkauf und das Schmieden. Dieser Faktor beruht auf der Tatsache, dass jede Runde auf 30 Entscheidungen beschränkt ist. Der Profit soll mit jeder einzelnen Entscheidung möglichst maximiert werden.
Daraus folgt, dass der Spieler bei dieser Strategie stets maximal investieren und profitieren soll. Der Spieler muss vor dem Handeln stets ausrechnen, ob er sich den Handel leisten kann. Wenn der Spieler beispielsweise zu viel Chronos für die Rohstoffe investiert und deshalb im Anschluss nicht mehr die Schmiedegebühren leisten kann, dann scheitert diese Spielstrategie. Ein guter Spielablauf mit dieser Strategie sieht folgendermaßen aus. Zuerst kauft sich der Spieler für 800 Chronos zwei Silberrohstoffe. Diese schmiedet er danach zu einer Silberrüstung und verkauft diese dann beim Händler. Der Spieler wiederholt diesen Vorgang bis er sich zwei Goldrohstoffe und die Goldrüstungsgebühr leisten kann. Sobald dies der Fall ist, wird der Vorgang mit zwei Goldrohstoffe wiederholt bis er sich zwei Smaragde leisten kann. Es ist wichtig beim Einkauf möglichst viel, aber nicht zu viel zu investieren. Kann man sich beispielsweise vier Goldrohstoffe und zwei mal die Goldrüstungsgebühr leisten, so soll man vier Goldrohstoffe anstatt nur zwei kaufen. Der Spieler steigt um auf den nächst wertvolleren Rohstoff, sobald er diesen

Rohstoff zwei mal kaufen und sich die Schmiedegebühr leisten kann.
In der Abbildung 5.2 ist das Resultat dieser Spielstrategie zu sehen. Der Chronosstand nach der 30. Entscheidung beträgt 21880. Der Vorteil dieser Spielstrategie ist offensichtlich die deterministische Eigenschaft. Durch diesen Determinismus folgt, dass der Spieler bei der Anwendung dieser Strategie, das Spiel nicht verlieren kann, da der Bankrottfall nicht eintreten kann. Die Strategie ist vom Zufall unabhängig und kann somit hundertprozentig reproduziert werden. Die Strategie hat jedoch im Vergleich zu den anderen Strategien einen Nachteil. Der Profit kann zwar maximiert werden, jedoch ist dieses Maximum im Vergleich zu risikoreichen Strategien stärker begrenzt. Dies beruht darauf, dass die Interaktionen, die einen sehr viel höheren Gewinn abwerfen können, mit einem zufallsbasierten Risiko verbunden sind und diese Interaktionen in der deterministischen Spielstrategie konsequent vermieden werden.

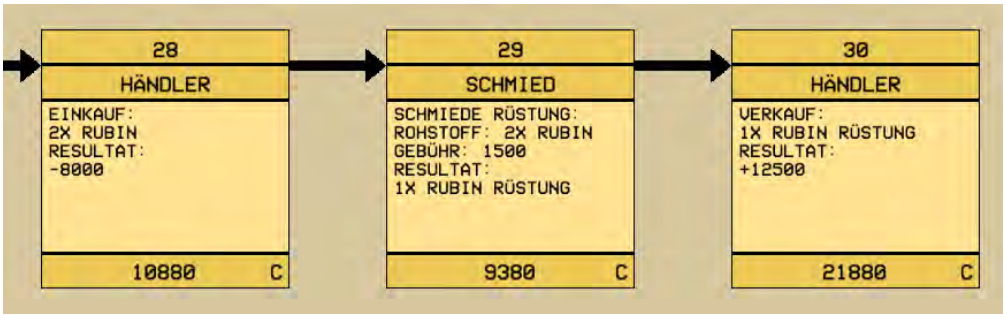

Abbildung 5.2: Resultat der deterministischen Spielstrategie

5.2 Risikoreiche Spielstrategie

Diese Spielstrategie basiert auf risikoreichen Interaktionen mit einem hohem Gewinnpotential. Da diese Spielstrategie vom Zufall sehr stark abhängt, kann das Resultat dieser Strategie stark variieren. Bei dieser Strategie interagiert man hauptsächlich mit der Mine, dem Auktionshaus und den Banditen. Mit dem Schmied wird dennoch gehandelt, da das Schmieden von Rohstoffen generell zu höherem Profit führt. Der Verzicht auf den Schmied ist ein zu großer Nachteil für den Spieler und ist daher nicht empfehlenswert.

Dies wurde bereits in der ersten Spielstrategie gezeigt. Sobald es nicht möglich ist die Auktionsgebühr zu bezahlen, wird der Händler als Verkaufsoption genutzt. Dies gilt auch wenn das Versteigern von Items, aufgrund ihrer geringen Seltenheit, nicht möglich ist. Zunächst werden bei dieser Spielstrategie bestimmte Verhaltensmuster und Grundprinzipien bei den jeweiligen NPCs festgelegt, an denen sich der Spieler orientieren kann. Angefangen bei der Mine muss sich der Spieler stets überlegen wie viele Minenarbeiter er für den jeweiligen Auftrag bezahlt. Das Grundprinzip ist es darauf zu achten, dass noch genug Chronos nach dem Minenauftrag vorhanden sind, sodass man die gewonnenen Rohstoffe schmieden lassen kann. Es müssen ebenfalls die Auktionsgebühren bedacht werden, um das Auktionshaus nutzen zu können. Der Spieler erhält die Information, dass es sich stets mehr lohnt viele Minenarbeiter gleichzeitig einzusetzen. Diese Tatsache beruht auf der Implementierungsweise der Mine. Es wird das Wahrscheinlichkeitsprinzip des Urnenmodell genutzt [9]. Beim Urnenmodell befinden sich Kugeln in einer Urne und diese Kugeln werden dann gezogen. Bei diesem Prinzip variieren die Art der Ziehung und die Unterscheidbarkeit der Kugeln. Die Kugeln können gleich sein oder sich voneinander unterscheiden. Die Arten der Ziehungen unterscheiden sich darin, ob die gezogene Kugel zurück in die Urne gelegt wird oder draußen bleibt [9]. Da sich nach der ersten Ziehung eine Kugel weniger in der Urne befindet, steigt die Wahrscheinlichkeit bei der zweiten Ziehung eine Kugel anderer Art zu ziehen [9].
Die Mine wurde nach dem Prinzip „Ziehen ohne Zurücklegen“ implementiert. Jeder Minenarbeiter zieht einen Rohstoff aus einer festen Menge der Rohstoffe. Dabei kann der Minenarbeiter auch einen leeren Rohstoff ziehen. Der leere Rohstoff repräsentiert das Fehlschlagen der Ziehung, d. h. der Spieler erhält dabei keinen Rohstoff. Stochastisch gesehen ist es somit sinnvoll mehrere Minenarbeiter in einem Auftrag einzusetzen. Die Wahrscheinlichkeit einen Rubin zu erhalten verhält sich nach den ersten drei Ziehungen folgendermaßen. Vor der ersten Ziehung existieren 130 Rohstoffe, einschließlich leere Rohstoffe. Es gibt fünf Rubine in diesen 130 Rohstoffen. Daraus ergibt sich die gerundete Wahrscheinlichkeit von 3,85% einen Rubin zu erhalten. Wenn der erste Minenarbeiter alles außer einen Rubin gefunden hat, so hat nun der zweite Minenarbeiter eine höhere Wahrscheinlichkeit einen Rubin zu finden, da die Mine nur noch 129 Rohstoffe

enthält. Dies erhöht die Wahrscheinlichkeit auf 3,88%. Weiter gerechnet hat der zehnte Minenarbeiter unter den selben Voraussetzungen eine Wahrscheinlichkeit von 4,17%. Diese Wahrscheinlichkeiten wirken auf dem ersten Blick sehr gering. Jedoch existieren ebenfalls die Chancen Rohstoffe wie Diamant, Smaragd und Gold zu erhalten und das Prinzip der steigenden Wahrscheinlichkeit betrifft auch diese Rohstoffe, sofern sie nicht schon vorher gefunden wurden. Nun wurde die Anzahl der Minenarbeiter auf die untere Grenze untersucht. Die obere Grenze kann folgender Maßen analysiert werden. Die maximale Anzahl an Minenarbeiter pro Auftrag beträgt 99. Dies kostet dem Spieler 19.800 Chronos und der Gesamtwert der Mine beträgt 30.250 Chronos. Der Gesamtwert wurde anhand der Verkaufspreise der Rohstoffe berechnet. Wenn die Rohstoffe aus der Mine zu Schwerter oder Rüstungen verarbeitet werden, dann erhöht sich der Gewinn noch weiter. Aus diesen Tatsachen ergibt sich somit die Strategie möglichst viele Minenarbeiter pro Auftrag einzusetzen. Der Spieler kennt die konkrete Anzahl der Rohstoffe nicht, dennoch kann er anhand der Prozessmodelle und zahlreicher Spieldurchläufe die Wahrscheinlichkeiten potentiell abschätzen. Ein weiterer Vorteil der sich beim Einsatz mehrerer Minenarbeiter entsteht, ist das Sparen von Entscheidungen. Wenn der Spieler zehn Minenarbeiter separat beauftragt, so würde dies zehn Entscheidungen kosten. Daher ist es sinnvoller bloß eine Entscheidung zu verbrauchen, indem man die zehn Minenarbeiter in einem Auftrag einsetzt.

Beim Auktionshaus gilt folgendes Grundprinzip. Zuerst muss der Spieler die Auktionsgebühr bezahlen. Diese Gebühr errechnet sich aus einem prozentualem Anteil des Verkaufspreises des jeweiligen Items. Danach trägt der Spieler ein Startgebot ein. Dieses Gebot muss mindestens null betragen. Empfehlenswert ist es jedoch einen sinnvolles Startgebot zu berechnen. Bei der Berechnung summiert man die Auktionshausgebühr mit dem Verkaufspreis des jeweiligen Items. Sobald der Spieler am Ende der Auktion diesen Betrag ausgezahlt bekommt, hat er für das Item genau so viel Profit erhalten, wie beim gewöhnlichem Verkauf beim Händler. Geht der Betrag jedoch darüber hinaus, so hat er seinen Profit gesteigert. Ist der Betrag geringer, so wurde das Item mit Verlust verkauft. Es soll somit pro Auktion mehr verdient werden, als beim Händler über den festen Verkaufspreis. Um diese Spielstrategie genauer zu untersuchen, muss die Implementierung des Auktionshauses betrachtet werden. Es gibt drei Auktionäre mit

unterschiedlichen Basiswahrscheinlichkeiten. Diese Basiswahrscheinlichkeiten beziehen sich auf die Bereitschaft weiter und höher zu bieten. Die Strategie beruht darauf, dass sich die Auktionäre stets gegenseitig überbieten. Je höher das Auktionsgebot wird, desto unwahrscheinlicher ist es, dass der jeweilige Auktionär höher bietet. Dennoch besteht das Potential, dass die Auktionäre unfassbar hohe Preise für die seltenen Items bieten. Dieses Potential wird bei dieser Spielstrategie genutzt. Der Spieler hat somit das Ziel, die optimalen Startgebote zu finden, denn wenn das Startgebot zu hoch ist so ist die Wahrscheinlichkeit ein Gebot zu erhalten niedrig. Wenn der Spieler kein Gebot erhält, so wird die Auktion unterbrochen und die gezahlte Auktionsgebühr wird zum Verlust. Es ist nicht empfehlenswert das Startgebot auf null zu setzen, da zwei der Auktionäre vorzeitig aussteigen können, obwohl dies sehr unwahrscheinlich ist.
Bei den Banditen kann der Spieler seine Strategie nicht stark variieren. Prinzipiell sollte er stets darauf achten mehrere Banditen in einem Auftrag einzusetzen, da es sonst unnötig viele Entscheidungen kosten könnte. Des weiteren sollte er die Auktionshausgebühr nicht vernachlässigen, die er beim Versteigern der gefundenen Gegenstände bezahlen muss. Bei den Banditen erhält der Spieler entweder nichts oder Schwerter und Rüstungen mit einer Seltenheit von mindestens vier. Pro Bandit besteht eine Wahrscheinlichkeit doppeltes Glück zu erhalten. Doppeltes Glück bedeutet, dass der Spieler die Chance hat mit einem Banditen ein Schwert und eine Rüstung zu erhalten. Bekommt er kein doppeltes Glück, so hat er bloß die Chance auf ein Schwert oder einer Rüstung.
Im Gegensatz zur deterministischen Strategie, ist es schwierig eine Mustervorgehensweise für die riskante Spielstrategie zu entwickeln. Dieses Problem beruht auf der Tatsache, dass die Resultate der zufallsbedingten Interaktionen stark variieren können. Der Hauptvorteil dieser Strategie ist das Potential auf sehr hohen Gewinn. Jedoch hat die Spielstrategie einen klaren Nachteil. Der Spieler könnte sehr viel Pech haben und sein gesamtes Chronos verlieren. Sobald er sich nichts mehr leisten kann und keine Items mehr besitzt ist das Spiel verloren. Diesem Problem wird in der kombinierten Spielstrategie entgegen gewirkt. In der Abbildung 5.3 sind die Resultate von drei Spielabläufen zu sehen, bei denen sich der Spieler an die risikoreiche Spielstrategie gehalten hat. Die Resultate unterscheiden sich sehr stark. Das höchste Resultat aus diesen drei Spielrunden beträgt 29.527 Chronos. Dieses Resultat übertrifft das Resultat

der deterministischen Strategie. Jedoch muss hierbei berücksichtigt werden, dass alle Spielrunden die vorzeitig verloren wurden nicht angezeigt werden. Eine Spielrunde ist erst ab 30 Runden gültig.

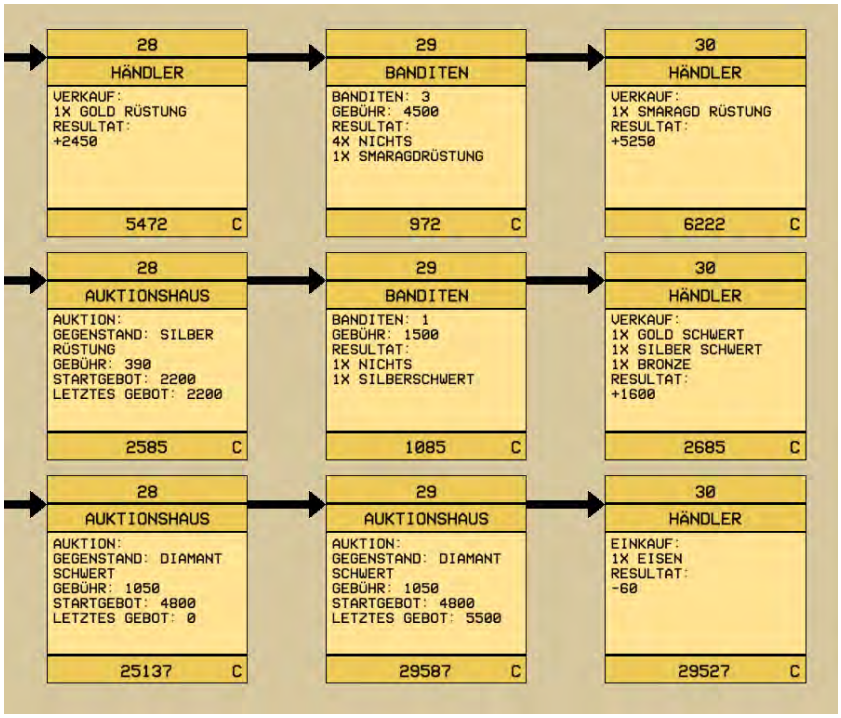

Abbildung 5.3: Drei Resultate der risikoreichen Spielstrategie

5.3 Kombinierte Spielstrategie

Bei der deterministischen Spielstrategie wurde eine sichere Vorgehensweise vorgestellt. Zwar kann der Profit maximiert werden, jedoch bleibt dieses Maximum durch den Verzicht auf Risiken kleiner als bei der risikoreichen Spielstrategie. Dennoch hat die risikoreiche Spielstrategie den Nachteil, das Spiel aufgrund eines Bankrotts vorzeitig zu verlieren. Nun sollen beide Spielstrategien kombiniert werden, um daraus eine neue Spielstrategie zu bilden. Diese Spielstrategie wird als kombinierte Spielstrategie bezeichnet. Bei dieser Spielstrategie soll die finanzielle Sicherheit des Spielers gewährleistet sein, während der Profit aus der deterministischen Spielstrategie möglichst hoch

überboten wird. Die Grundidee hierbei ist zuerst ein großes Kapital anzusammeln und danach dieses Kapital in riskante Interaktionen zu investieren. Bei der Ansammlung des Kapitals verwendet man die deterministische Strategie. Hierfür muss entschieden werden, wie viele Entscheidungen verbraucht werden sollen. Je mehr Entscheidungen man verbraucht, desto höher wird das Kapital. Nach zwölf Entscheidungen kann der Spieler mithilfe der deterministischen Spielstrategie c. a. 3.680 Chronos und nach 21 Entscheidungen c. a. 12.880 Chronos ansammeln. Der Spieler muss selbst entscheiden, nach wie vielen Entscheidungen er sein Kapital investieren möchte. Nach dem das Kapital aufgebaut wurde kann es beliebig eingesetzt werden. Empfehlenswert sind jedoch die Banditen, da sie Gegenstände der Seltenheit vier oder höher verschaffen. Diese Gegenstände können dann im Auktionshaus verkauft werden. Selbstverständlich besteht dennoch das Risiko bei den zufallsbedingten Interaktionen Verlust zu machen. Das vorher erarbeitete Kapital soll den Verlust mindern, da das übrige Kapital zum Wiederaufbau genutzt werden kann. Dabei muss der Spieler beachten nie sein gesamtes Kapital zu riskieren. In der Abbildung 5.4 sind zwei Spielabläufe zu sehen, bei denen die kombinierte Spielstrategie angewandt wurde. Bei dem ersten Ablauf wurden zwölf Entscheidungen und beim zweiten 21 Entscheidungen zum Kapitalaufbau genutzt. Nachdem Kapitalaufbau wurden 60 bis 80% des Kapitals für die Banditen genutzt. Auch hierbei gibt es Spielabläufe, die vorzeitig durch eine Niederlage geendet haben, da die Spielstrategie eine hohe aber keine hundertprozentige Sicherheit bietet.

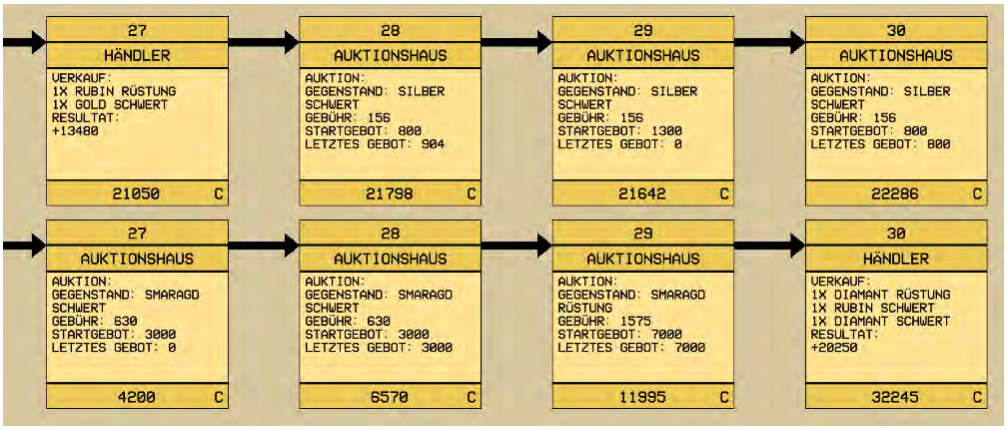

Abbildung 5.4: Zwei Resultate der kombinierten Spielstrategie

5.4 Diskussion

In diesem Kapitel wurden bisher drei mögliche Spielstrategien vorgestellt. In Abbildung 5.5 befindet sich eine Übersicht mit den drei Spielstrategien.

Bezeichnung	Vorgang	Vorteile	Nachteile
Deterministische Spielstrategie	➔ Benutzt nur Händler und Schmied ➔ Rüstungen schmieden und verkaufen ➔ Stets die möglichst seltenen Rohstoffe verwenden	➔ Risikofrei ➔ Niederlage nicht möglich ➔ Leicht anwendbar	➔ Maximalprofit geringer ➔ Spielerfahrung weniger spannungsvoll
Risikoreiche Spielstrategie	➔ Benutzt hauptsächlich zufallsbedingte Interaktionen ➔ Fokussiert auf Banditen, da hoher Profit ermöglicht wird	➔ Maximalprofit sehr hoch ➔ Spielerfahrung spannungsvoll	➔ Sehr risikobasiert ➔ Bei Verlust wird die Spielerfahrung frustrierend ➔ Niederlage sehr wahrscheinlich
Kombinierte Spielstrategie	➔ Kombination aus den beiden Spielstrategien ➔ Spieler baut sich mit der det. Spielstrategie ein Kapital auf ➔ Danach wird das Kapital risikoreich investiert	➔ Kapitalaufbau kann Niederlage entgegen wirken ➔ Hoher Kapital ermöglicht höhere Investitionen mit mehr Profit	➔ Niederlage nicht ausgeschlossen ➔ Mühsamer Kapitalaufbau ➔ Kapitalverlust kann frustrierend sein

Abbildung 5.5: Übersicht der drei Spielstrategien

Es gibt mit Sicherheit noch weitere Strategien, die durch Probieren und Berechnen erstellt werden können. Das Ziel von diesem Kapitel ist jedoch nicht die Vorstellung jeder möglichen Spielstrategie und auch nicht die perfekte Optimierung dieser Spielstrategien. Es geht hierbei eher um Ansätze die beim Spielen helfen können. Die Prozessmodelle haben bei der Entwicklung dieser Spielstrategien sehr mitgeholfen, da man sie gut Auswerten kann. Bezieht man die bisherigen Erkenntnisse auf die Realität, so ist es denkbar das Prinzip der dynamisch generierten Prozessmodelle auf einer Betriebsebene anzuwenden. In einem Betrieb werden zahlreiche Entscheidungen von zahlreichen Mitarbeitern getroffen. Auftretende Probleme können auf unterschiedliche Arten gelöst werden. Bei der Herstellung von Produkten könnte stets ein Optimierungsbedarf bestehen.

Genau bei diesen Vorkehrungen können Prozessmodelle helfen [10]. Wenn sich nun konsequent im Hintergrund Prozessmodelle dynamisch zum Entscheidungsverhalten der Mitarbeiter generieren würden, dann ist es denkbar, dass eine Auswertung dieser Prozessmodelle die Arbeitsweise optimieren könnte. Selbstverständlich soll hierbei die zusätzliche Komplexität der Realität nicht vernachlässigt werden. Die Komplexität in *MineTrade* wurde gering gehalten, da zu viel Komplexität in einem Spiel eine negative Auswirkung auf die Spielerfahrung erzeugt [7]. Die Komplexität bezieht sich hierbei beispielsweise auf die begrenzte Anzahl der Interaktionen. Die Interaktionen können zwar auf einer quantitativen Art variieren, zum Beispiel kauft man drei anstatt vier Items, aber dennoch bleibt die Struktur der Interaktion gleich. Diese Faktoren müssen bei der Anwendung in der Realität definitiv berücksichtigt werden.
Des Weiteren ist die Informationsbeschaffung in der Realität komplizierter. Informationen beziehen sich hierbei zum Beispiel auf die Kosten und die Beschaffung von Produkten. Ebenfalls zu beachten gilt die Verfügbarkeit von Mitarbeitern und deren Funktionen. In *MineTrade* hat keiner der NPCs einen Bedarf für Pausen, Krankheitstage oder Urlaub. Deshalb muss in einem realen Betrieb überprüft werden, ob die jeweilige Interaktion mit einem Mitarbeiter überhaupt verfügbar ist. Das Bezieht sich ebenso auf die Teamarbeit in einem Betrieb, denn alle Mitarbeiter in *MineTrade* sind programmiert. Die technische Umsetzung ist ebenfalls nicht so einfach, da man in der Realität jede Entscheidung maschinell oder per Hand protokollieren und in ein Kollektiv einordnen muss. Dies wiederum könnte bei einer schlechten Umsetzung einen zusätzlichen und enormen Zeitaufwand generieren [10]. Ein weiterer Faktor der bei der Anwendung in der Realität beachtet werden muss, ist die Weiterentwicklung eines Betriebs [10]. Wenn in *MineTrade* ein neuer NPC implementiert wird, dann ist der Aufwand diesen NPC mit der Prozessmodellgenerierung zu verknüpfen relativ einfach. In der Realität können Weiterentwicklungen viel komplexer sein.

6

Verwandte Arbeiten

Dieses Kapitel setzt sich mit wissenschaftliche Arbeiten aus ähnlichen Themenbereichen auseinander. Die Themen beschäftigen sich hauptsächlich mit Prozessmodellen, Serious Games und Gamification.
In [3] wird die Anwendung von Gamification-Prinzipien in einem realen Geschäftsprozessmanagement beschrieben. Gamification hat den Zweck anhand von Spielelementen die Teilnehmer beim Erreichen von Zielen zu motivieren. Dabei werden Punktesysteme, Herausforderungen und Bestenlisten verwendet. Die Teilnehmer erhalten Punkte für das Erledigen bestimmter Aufgaben. Hierbei ist es wichtig die Gamification-Aspekte in den alltäglichen Geschäftsablauf der Teilnehmer zu integrieren. Um dies zu Bewerkstelligen wurde ein System aus mehreren Abschnitten erstellt. Es wird bestätigt, dass sich die intrinsische Motivation der Teilnehmer durch das Gefühl der Selbstverwirklichung steigern lässt.
In [4] wurde anhand von Gamification der Einfluss der sozialen Distanz auf die Prozessmodellierung erforscht. Die soziale Distanz bezieht sich hierbei auf die Art der Informationsaufnahme der Versuchsgruppen. Es gibt nämlich zwei Versuchsgruppen. Beide Versuchsgruppen haben die Aufgabe ein Prozessmodell zu erstellen. Dabei soll der Bestellprozess in einem Warenhaus nach modelliert werden. Die erste Versuchsgruppe spielt den Bestellprozess in einer virtuellen 3D-Simulation nach. Die zweite Versuchsgruppe sieht lediglich ein Video auf dem dieser Bestellprozess abgespielt wird. Die These dieser Ausarbeitung wird nach der Auswertung des Experiments widerlegt. Ursprünglich bestand die Annahme darin, dass eine geringere soziale Distanz die Genauigkeit der nachträglich erstellten Prozessmodelle erhöhen sollte. Jedoch tritt hier das genaue Gegenteil auf, denn die zweite Versuchsgruppe mit der höheren sozialen

Distanz erstellt präzisere Prozessmodelle.
Die Konzepte der Geschäftsprozessmodellierung an der Universität zu lehren, hat sich als Herausforderung erwiesen [11]. Es wird vorgeschlagen die Konzepte anhand von Geschäftssimulationsspielen zu lehren, um den Studenten eine klarere Vorstellung der Modellierung von Geschäftsprozessen zu ermöglichen. Diese Geschäftssimulationsspiele helfen beim Experimentieren und unterstützen ein tieferes Lernen.
Die Lehre von Geschäftsprozessmodellierung kann mit Serious Games unterstützt werden [2]. Es wird in diesem Fall ein Spiel namens ImPROVE eingesetzt. Das Spiel ist eine Simulation und basiert auf einem realen Kontext. Dieser reale Kontext ist das sogenannte Manchester-Triage-System (MTS). Das MTS ist ein standardisiertes Verfahren die bei der Notaufnahme angewandt wird. Ziel des MTS ist es möglichst schnell die Behandlungsprioritäten eines Patienten festzulegen. Es wird schlussgefolgert, dass der Einsatz von Serious Games das Lernen von explizitem Wissen beschleunigen kann.
In [12] wurde ein innovatives Serious Game in einem Unternehmen, im Rahmen einer Feldstudie, eingesetzt. Die Angestellte haben das Ziel Prozesse zu interpretieren und dann daraus Prozessmodelle zu erstellen. Es wurde gezeigt, dass das Spieldesign und die Spielerfahrung einen positiven Einfluss auf die Erfüllung des Ziels haben. Das Spiel hat auch eine positive Wirkung auf die Gruppenarbeit der Mitarbeiter.
Ein weiterer Anwendungsbereich der Serious Games ist die Unterstützung der Gesundheit und Fitness[13]. Hierbei wird der Einsatz von Fitnessspielen (den sogenannten *Exergames*) erforscht. Die *Exergames* sollen die Motivation des Spielers unterstützen, damit der Spieler mehr Sport treibt. Das Resultat der Forschung hat gezeigt, dass die spielerische Art die Motivation des Spielers tatsächlich positiv beeinflusst.
In [14] wird der Einsatz der Gamification im Bereich des Tourismus erforscht. Es wird dabei die Motivation der Touristen ermittelt, Spiele zu spielen. Die Spiele sind mobil spielbar und teilen sich in zwei Kategorien. Die erste Kategorie sind die sogenannten *Social Games*. Die *Social Games* erfüllen hauptsächlich Marketingzwecke für Reiseunternehmen. Die zweite Kategorie sind *Location-based Games*. Die *Location-based Games* unterstützen die Beschäftigung der Touristen, durch zusätzliche Informationen zu örtlichen Sehenswürdigkeiten.

7

Zusammenfassung

Im Rahmen dieser Ausarbeitung wurde ein Spiel implementiert, in welchem die Entscheidungsanalyse des Spielers durch den Einsatz der Prozessmodellierung unterstützt wird. Diese Prozessmodelle werden dynamisch im Hintergrund zur Spiellaufzeit generiert und können jederzeit abgerufen werden. Es können auch Prozessmodelle aus vorherigen Spielrunden geladen und angezeigt werden. Das ermöglicht dem Spieler mehrere Prozessmodelle miteinander zu vergleichen. Es wurden alle Anforderungen des Spiels erfüllt und alle notwendigen Funktionalitäten implementiert. Zu diesen Funktionalitäten gehören die fünf NPCs, das Inventar, die Items, das Tooltip und die Speicherfunktion. Beim Game Design wurde viel recherchiert und vorausgeplant um die Umsetzung möglichst reibungslos zu verwirklichen. Das Spiel wurde vorzeitig fertiggestellt. In der Ausarbeitung wird zuerst die Motivation und die Problemstellungen näher erläutert. Danach folgen die Abschnitte mit den notwendigen Grundlagen für das Verständnis des Projekts und der Ausarbeitung. Diese Grundlagen behandeln die Prozessmodellierung und die *Unity* Spiel-Engine. Ebenfalls werden neben den Grundlagen auch der Zweck und die Vorteile der Prozessmodellierung und der Einsatz von der *Unity* Spiel-Engine erklärt. Unternehmen können durch den Einsatz von Prozessmodellierung ihre Geschäftsprozesse optimieren und dadurch ihre Wettbewerbsfähigkeit steigern. Die *Unity* Spiel-Engine bietet einige Funktionalitäten an, die den Entwickler bei der Implementierung unterstützen und dabei Zeit sparen.
In der Ausarbeitung werden ausführlich die Hintergedanken zum Game Design und der Implementierung des Spiels beschrieben. Ebenfalls werden verworfene Ideen erwähnt, wie zum Beispiel die Lauffunktionalität des Spielers zu entfernen und stattdessen den Mauszeiger zu benutzen. Es wurden für das Spiel fünf NPCs entworfen, damit mehrere

Entscheidungsmöglichkeiten für den Spieler existieren. Die NPCs erfüllen alle eine eigene Funktion im Spiel und sie unterscheiden sich voneinander. Es wurden sowohl deterministische als auch zufallsbedingte Spielmechanismen implementiert. Der Hauptgrund dafür ist die Ermöglichung von unterschiedlichen Spielstrategieansätzen. Der Zufall soll jedoch auch die Spielerfahrung spannender gestalten.
Es wurden drei unterschiedliche Spielstrategien erklärt. Die erste Spielstrategie benutzt die deterministischen Spielelemente. Das hat den Vorteil der finanziellen Sicherheit. Jedoch kann die zweite Spielstrategie, die risikoreich ist, den Profit der ersten Spielstrategie übertreffen. Die risikoreiche Spielstrategie hat jedoch den Nachteil, dass er vom Zufall abhängt und daher scheitern kann. Die dritte Spielstrategie kombiniert die ersten beiden Spielstrategien. Es wird zuerst ein Kapital aufgebaut und dieses Kapital wird dann investiert.
Des Weiteren wurden verwandte Arbeiten genannt und erläutert. Diese Arbeiten beschäftigen sich mit dem Einsatz der Gamification, der Geschäftsprozessmodellierung und Serious Games. Daher haben diese Arbeiten eine inhaltliche Schnittmenge mit dieser Ausarbeitung.

8

Ausblick

Im Hinblick auf denkbare und zukünftige Erweiterungen des Spiels *MineTrade*, können einige Möglichkeiten hervorgehoben werden. Die erste Erweiterungsmöglichkeit ist die Implementierung einer Multiplayer-Funktionalität. Das Spiel wird auf einem Server gehostet, so dass mehrere Spieler sich mit der Spielwelt verbinden und interagieren können. Neben der gewöhnlichen Interaktion mit den NPCs, soll es den Spielern ebenfalls ermöglicht werden untereinander zu handeln. Dadurch entstehen neue Faktoren, die die Spielerfahrung beeinflusst [15]. In den letzten Jahren ist die Beliebtheit von Multiplayer-Spielen angestiegen, insbesondere im Bereich der Rollenspiele [15]. Durch die Einführung von Berufsspezialisierungen wird das Spiel vielfältiger. Ein einfaches Beispiel hierfür ist die Interaktion von zwei Spielern, bei der sich der eine Spieler als Schmied und der andere Spieler als Minenarbeiter spezialisiert hat. So kann dann der Spieler mit der Spezialisierung als Minenarbeiter Rohstoffe für den anderen Spieler besorgen. Dadurch würden Abhängigkeiten zwischen den Spielern entstehen und jede Berufsspezialisierung seinen eigenen Wert erhalten. Ähnlich wie in der Realität würde in der Spielwelt ein Marktsystem entstehen, welches auf dem Prinzip Angebot und Nachfrage basiert [15].
Die zweite Erweiterungsmöglichkeit, unabhängig von der ersten, ist der Einsatz von Multiagent Learning [16]. In einem Multiagent-System gibt es mehrere Agenten [16]. Diese Agenten kooperieren miteinander oder konkurrieren gegeneinander, um ein bestimmtes Ziel zu erreichen [16]. In Bezug zu *MineTrade* kann man die Agenten gegeneinander konkurrieren lassen. Jeder Agent hat das Ziel durch möglichst wenige Entscheidungen einen bestimmten Geldbetrag zu erreichen. Sie sollen demnach die möglichst beste Strategie finden. Mit der Zeit würden die Agenten ihre Verhaltensweise anpassen und

es würden ständig neue Prozessmodelle entstehen [16]. Eine Analyse dieser Prozessmodelle ermöglicht einen Einblick in das Verhalten der Agenten und legt somit neue Strategieansätze für den Spieler offen. Natürlich bleiben hierbei die zufallsbedingten Spielhandlungen bestehen. Der Zufall gilt für den Spieler und auch für die Agenten. Dennoch würden die Agenten nach einigen Durchläufen beurteilen können, welche zufallsbedingten Handlungen als lohnenswert erscheinen und welche vermieden werden sollten [16].

Literaturverzeichnis

[1] Dadam, P., Reichert, M., Rinderle-Ma, S.: Prozessmanagementsysteme. Informatik-Spektrum **34** (2011) 364–376

[2] Ribeiro, C., Fernandes, J., Lourenço, A., Borbinha, J., Pereira, J.: Using Serious Games to Teach Business Process Modeling and Simulation. In: Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV). (2012) 1–7

[3] Herzberg, N., Kunze, M.: The Business Process Game. Proceedings of the 7th Central-European Workshop on Services and their Composition (2015) 26–32

[4] Winter, M.: Influence of Psychological Distance on Process Modeling: A Gamification Approach. Master's thesis, Ulm University (2015)

[5] Kossak, F., Illibauer, C., Geist, V., Kubovy, J., Natschläger, C., Ziebermayr, T., Theodorich, K., Freudenthaler, B., Schewe, K.D.: A Rigorous Semantics for BPMN 2.0 Process Diagrams. 1 edn. Springer (2014)

[6] Halpern, J.: Developing 2D Games with Unity. 1 edn. Apress (2019)

[7] Schell, J.: Die Kunst des Game Designs. 1 edn. mitp (2012)

[8] Schöning, U.: Algorithmen - kurz gefasst. 1 edn. Spektrum Akademischer Verlag (1997)

[9] Basler, H.: Grundbegriffe der Wahrscheinlichkeitsrechnung und Statistischen Methodenlehre. 11 edn. Physica-Verlag Heidelberg (1994)

[10] Mutschler, B., Reichert, M.: Understanding the Costs of Business Process Management Technology. In Glykas, M., ed.: Business Process Management - Theory and Applications. Number 444 in Studies in Computational Intelligence. Springer (2013) 157–194

[11] Vuksic, V.B., Bach, M.P.: Simulation Games in Business Process Management Education. In: International Journal of Business, Human and Social Sciences. Volume 6., World Academy of Science, Engineering and Technology (2012) 729–734

[12] Rosenthal, K., Strecker, S.: Business Process Modelling as Serious Game: Findings from a Field Study. In: ECIS. (2018) 165–180

[13] Göbel, S., Hardy, S., Wendel, V., Mehm, F., Steinmetz, R.: Serious Games for Health: Personalized Exergames. In: Proceedings of the 18th ACM international conference on Multimedia, ACM (2010) 1663–1666

[14] Xu, F., Tian, F., Buhalis, D., Weber, J., Zhang, H.: Tourists as Mobile Gamers: Gamification for Tourism Marketing. Journal of Travel & Tourism Marketing **33** (2016) 1124–1142

[15] Ducheneaut, N.: The Social Side of Gaming: A Study of Interaction Patterns in a Massively Multiplayer Online Game. In: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, CSCW 2004, Chicago, Illinois, USA, November 6-10, 2004, ResearchGate (2004)

[16] Tuyls, K., Weiss, G.: Multiagent learning: Basics, challenges, and prospects. AI Magazine **33** (2012)

Abbildungsverzeichnis

Name: Yasin Agirbas Matrikelnummer: 830155

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Yasin Agirbas