
**Freie Universität Berlin**
**Institut für Informatik**

**Diplomarbeit**

# Java Settlers

# Intelligente agentenbasierte Spielsysteme für intuitive Multi-Touch-Umgebungen

Miao Wang

mwang@mi.fu-berlin.de

Betreuer: Prof. Dr. Raúl Rojas

Marco Block

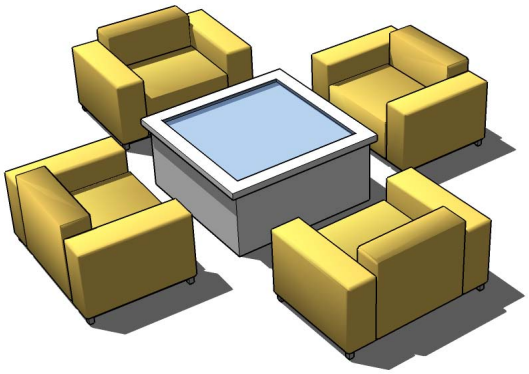
## Kurzfassung

## Java Settlers
## Intelligente agentenbasierte Spielsysteme für intuitive Multi-Touch-Umgebungen

Computersysteme sind heutzutage kaum wegzudenken und weisen intelligente Algorithmen zur Lösung komplexer Probleme vor. Besonders in der Unterhaltungselektronik haben Computerspiele in den letzten Jahren einen großen Fortschritt erlangt, indem lernende künstliche Intelligenzen die besten menschlichen Spieler auf der Welt herausfordern [1, 2, 3]. Dabei dient die agentenbasierte Spieltheorie als ein Mittel, um Menschen ähnliche oder überlegene Computergegner zu generieren [4]. In einem Suchverfahren wird für jede Situation die beste Aktion ermittelt und durchgeführt. Die Herausforderung dabei ist es, die Berechnungen in einer für den Menschen akzeptablen Zeit zu beenden.

Außerdem zeigen heutzutage verschiedene Computersysteme unterschiedliche Möglichkeiten zur Interaktion. Während sich beim Standard-PC die Tastatur und Maus durchgesetzt hat, gibt es im mobilen Bereich vermehrt die Steuerung über Fingerberührungen [5, 6, 7]. Multi-Touch ist hierbei eine intuitive Art der Mensch-Computer-Interaktion, die versucht herkömmliche Eingabeelemente wie Maus und Tastatur zu ersetzen [8]. Dabei werden optische oder akustische Systeme eingesetzt, um mehrere Berührungspunkte einer Hand zu orten und in Gesten für Anwendungsprogramme zu übersetzen.

Heutige Computerspiele sind gute Beispiele dafür, künstliche Intelligenz mit intuitiver Interaktion zu verbinden und in Einklang zu bringen. In dieser Diplomarbeit soll anhand einer Umsetzung des Brettspiels „Siedler von Catan“ ein intelligentes Spielsystem entwickelt werden, welches über eine Multi-Touch-Benutzerschnittstelle zu bedienen ist. Für die unterliegende künstliche Intelligenz wird das generische Framework jGameAI verwendet [9, 10]. Diese Arbeit soll einen Ausblick von natürlichen menschlichen Interaktionen mit einem intelligenten und intuitiven Computersystem demonstrieren. Zudem soll es als Basis für weitere Projekte in dieser Richtung dienen.



## Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Diplomarbeit mit dem Titel „Java Settlers - Intelligente agentenbasierte Spielsysteme für intuitive Multi-Touch-Umgebungen“ von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben. Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den

(Miao Wang)

# Danksagungen

Ich möchte mich herzlichst bedanken bei Prof. Dr. Raúl Rojas, der mir dieses Thema als Diplomarbeit bewilligt hat. Als Leiter der Arbeitsgruppe *Künstliche Intelligenz* hat er mich mit seinen Vorträgen und Vorlesungen gefördert, mein Wissen in vielen Bereichen zu erweitern und wissenschaftliche Herangehensweisen anzueignen. Zudem stand er jederzeit für Diskussionen und Fragen zur Verfügung.

Auch meinem Betreuer Marco Block gebührt großer Dank, da er stets für Fragen offen war und mich immer wieder auf neue Ideen gebracht hat. Ich bedanke mich auch für konstruktive Kritik und entscheidende Tipps.

Ein großes Dankeschön geht an Maro Bader und Philipp Holzschneider, ohne die beiden diese Diplomarbeit nicht möglich gewesen wäre. Maro danke ich für die allzeitige und hilfsbereite Unterstützung mit seinem Framework jGameAI. Philipp danke ich für die ständige und konstruktive Hilfe beim Thema Multi-Touch.

Ein weiteres Lob geht an alle Studenten, die mich in der Vorlesung Künstliche Intelligenz im Sommersemester 2008 tatkräftig unterstützt haben, indem sie alternative künstliche Spieler entwickelt haben. Gleiches gilt für die vielen anderen Teilnehmer, die auch bei den Experimenten geholfen haben.

Der Arbeitsgruppe *Künstliche Intelligenz* und der Arbeitsgruppe *Spieleprogrammierung* möchte ich als ganzes danken für ein familiäres Umfeld und grenzenlose Forschungsfreiheit. Des Weiteren möchte ich mich bei all den Korrekturlesern bedanken, die noch etliche Fehler entdeckt haben.

Ein offizielles Dankeschön geht an Tim Roth für Schemabilder und Mika Vaaraniemi für ingame-Texturen.

Letztlich möchte ich noch meinen herzlichen Dank an meine Familie aussprechen, die immer für mich da war.

Miao Wang

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Überblick

Computerbasierte Systeme sind heutzutage fast allgegenwärtig vorzufinden. Sie spielen in unserem modernen Leben eine große Rolle und sind zudem kaum wegzudenken. Durch ihre Vielseitigkeit unterstützen sie uns bei komplexen Verarbeitungen und Simulationen oder helfen uns bei alltäglichen Aufgaben. Softwareentwickler verwenden sie zur Erstellung neuartiger Anwendungsprogramme und Algorithmen. Ein Designer nutzt sie zum Entwurf und zur Gestaltung digitaler Medien. In der Medizin unterstützen Expertensysteme bei der Diagnose [11]. Viele Jugendliche nutzen sie als Multimediaplattform oder spielen fast fotorealistische Spiele auf ihnen. Einerseits werden die Systeme immer intelligenter und können kompliziertere Probleme bewältigen. Andererseits benötigen viele Systeme stets noch den menschlichen Anwender zur zuverlässigen Steuerung und Überwachung. Für die Kommunikation zwischen Mensch und Maschine gibt es verschiedene Möglichkeiten zur Interaktion. Die Maus und die Tastatur haben sich sicherlich über Jahre hinweg bewährt, aber es gibt noch unzählige andere Techniken wie z.B. Touchscreens, Augmented Reality, Gestenerkennung oder Spracherkennung [5, 7].
Diese Diplomarbeit soll im Hinblick auf intelligente und intuitiv interaktive Systeme versuchen, drei Aspekte miteinander zu vereinen. Erstens soll das bekannte Brettspiel *„Die Siedler von Catan“* in eine Computerversion mit durchdachten Computerspielern überführt werden. Zweitens wird für dieses Vorhaben jGameAI eingesetzt, ein generisches Framework zum Erstellen von intelligenten Spielen [9]. Es wird verwendet, um nutzenbasierte Agenten für das Brettspiel zu entwickeln. Drittens wird das Ganze mit einer intuitiven Ansteuerung über Multi-Touch ausgestattet. Dafür soll eine Schnittstelle konzipiert werden, die mehrere gleichzeitige Fingerberührungen eines Tischsystems erkennen und verarbeiten kann. Der Bau des Tischsystem, welches das Brettspiel darstellen und Eingaben entgegennehmen soll, wird skizziert. Dieses Vorhaben wurde in der Programmiersprache Java realisiert und trägt somit als Arbeit den Namen *Java Settlers*. Teile dieser Arbeit sollen dokumentiert und online zur Verfügung gestellt werden [12].
Im folgenden Kapitel sollen die Beweggründe und die Motivation dargelegt werden, warum dieses Thema als Diplomarbeit ausgewählt wurde und was es mit den einzelnen Aspekten auf sich hat. Um die einzelnen Punkte dieser Arbeit gänzlich zu verstehen, bedarf es einigem Hintergrundwissen. Dies betrifft beispielsweise die Regeln und die verschiedenen Strategien des Brettspiels, die im Kapitel 2 tiefergehend vorgestellt werden. Im selben Kapitel werden auch einige Bezüge aus der Spieltheorie definiert und das jGameAI-Framework erläutert. Verschiedene Applikationstypen und Projekte mit Multi-Touch sowie die Techniken, die dahinter stehen, werden im Anschluss daran präsentiert. Das Konzept und der mögliche technische Aufbau eines Tischsystems werden in Kapitel 3 erläutert. Im 4. Kapitel wird näher auf die Software und ihre Benutzeroberfläche eingegangen, gemeinsam mit den Schnittstellen zu Multi-Touch und dem

jGameAI Framework. Die Intelligenz der Software soll schließlich in Kapitel 5 näher beleuchtet werden, indem die Spiellogik nochmal im Detail erklärt wird. In Kapitel 6 werden Experimente erläutert, die unternommen wurden sind, um die Intuitivität und die Intelligenz des Systems zu bewerten, und deren Ergebnisse präsentiert. Dieses Projekt stellt eine Basis für weitere Projekte dar und bietet Möglichkeiten für zukünftige Arbeiten, die im letzten Kapitel 7 vorgestellt werden.
Aus Platzgründen und zwecks Übersichtlichkeit werden große Bilder und Skizzen im Anhang A aufgeführt. Dort finden sich Schemazeichnungen für verschiedene Multi-Touch-Systeme, Screenshots, diverse Codefragmente und grafische Auswertungen zur Umfrage, die für die Experimente eingesetzt wurde.
In dieser Diplomarbeit werden Eigennamen und Begriffe bei der ersten Nennung *kursiv* dargestellt. Wichtige Begriffe werden zudem **fett** hervorgehoben, während wichtige Markennamen in Anführungszeichen („“) gesetzt werden. In einigen Beispielen und bei manchen Begriffen wie „der Agent“ oder „der Spieler“ wird die männliche Form verwendet, jedoch gilt hierfür natürlich genauso die weibliche Form. Bei manchen Fachbegriffen wird der englische Terminus verwendet, wenn kein äquivalenter deutscher Begriff existiert.

## 1.2 Motivation

Das deutsche Brettspiel „Die Siedler von Catan“, welches 1995 von Klaus Teuber entwickelt wurde und im Kosmos-Verlag erschienen ist, hat sich bis heute über 15 Mio. mal weltweit verkauft und ist in über 20 verschiedenen Sprachen in mehr als 40 Ländern erschienen [13]. Dabei wurde es sowohl national als auch international mit diversen Preisen ausgezeichnet, insbesondere mit der im deutschsprachigen Raum bedeutendsten Spieleauszeichnung *„Spiel des Jahres“* [14]. Der hohe Bekanntheitsgrad hat dazu beigetragen, dass dieses Spiel für diese Arbeit gewählt wurde, denn durch die Beliebtheit ist ein ganzes Universum bezüglich „Die Siedler von Catan“ entstanden, in dem rege Strategien und Tipps diskutiert und ausgetauscht werden.
Mit vielen neuen frischen Spielideen und einfachen Regeln hebt sich „Die Siedler von Catan“ von den üblichen Brettspielen ab. In einer simulierten Spielbrettwelt müssen die Spieler Siedlungen, Städte und Straßen errichten, um mit denen an Rohstofferträgen zu gelangen, die weitere Baumöglichkeiten zu eröffnen. Dafür werden Siegpunkte vergeben. Wer zuerst eine gewisse Anzahl an Siegpunkten erreicht, hat gewonnen (siehe Regeln in Kapitel 2.1).
Charakteristisch sind die kooperativen Möglichkeiten der Spieler untereinander Rohstoffe zu handeln. Damit eröffnen sich eine große Menge an Handlungsmöglichkeiten und somit erfolgreichen Spielstrategien, die zum Sieg führen könnten. Zudem wird das Spiel deutlich lebhafter durch das gegenseitige Handeln und Feilschen. Aufgrund der Popularität sind im Catan-Universum zum Basisspiel noch viele weitere Erweiterungen erschienen als auch andere Spielvarianten wie Karten-, Würfel, Computer- oder Konsolenspiele. Bei letzteren beiden werden

als Ansatz für die Intelligenz lokale Entscheidungen mit Hilfe von einfachen Heuristiken getroffen, da der Suchraum durch die vielen Handlungsmöglichkeiten enorm groß ist. Aus diesem Grund stellt „Die Siedler von Catan“ eine Herausforderung für die Künstliche Intelligenz dar. In der Spieltheorie wurden üblicherweise Spiele wie Tic-Tac-Toe [15], Schach [16, 1], Dame [2], Backgammon [3], Go [17], usw. zu wissenschaftlichen Betrachtungen herangezogen, die ich für diese Arbeit als *„traditionelle Spiele“* deklarieren will. Im Vergleich zu den traditionellen Spielen weist „Die Siedler von Catan“ eine Reihe von Unterschieden auf (siehe Kapitel 5.1), was es für spieltheoretische Untersuchungen reizvoll macht.
Ich selber spiele bereits seit Jahren begeistert „Die Siedler von Catan“. Zu dritt oder zu viert gemeinsam mit Freunden oder Bekannten können lustige Nachmittage oder Abende mit dem Spiel verbracht werden. Der Spielaufbau ist durch das Spielprinzip jedes Mal anders und auch der Spielablauf ist nicht detailliert vorhersehbar. Während einer Partie, die mehrere Stunden dauern kann, werden mit den Mitspielern Allianzen geschlossen oder Handel verweigert. Damit wird das Spiel heiter lustig, teilweise sehr emotional und auch temperamentvoll. Meine eigenen Erfahrungen vom Spiel flossen auch mit in die Computerversion ein und trugen sicherlich dazu bei, dieses Spiel für ein Diplomthema zu wählen.
Teil dieser Diplomarbeit ist es, das Brettspiel in eine Computerversion zu überführen. Mit einer schriftlichen Genehmigung der Catan GmbH in Deutschland wurde es mir erlaubt das Spielprinzip von „Die Siedler von Catan“ zur wissenschaftlichen Nutzung im Rahmen dieser Diplomarbeit zu verwenden.
Für die Umsetzung sollte das Spiel nicht von Grund auf neu geschrieben werden. In der Spieltheorie besitzen die traditionellen Spiele teilweise viele gemeinsame Prinzipien. Es liegt nahe ein Framework zu erstellen, welches diese Gemeinsamkeiten überdeckt und für eine Hilfestellung zur Erstellung solcher Spiele sorgt. jGameAI ist genau solch ein generisches Framework zur Erstellung von einer Vielzahl an intelligenten Spielen, welches an der Freien Universität Berlin entwickelt wurde [9, 10]. Mit dem Framework lassen sich für neue Spiele neue Regeln und ein Zuggenerator definieren, der gültige Züge generieren kann. Zudem muss eine grafische Benutzeroberfläche (GUI) und eine Bewertungsfunktion erstellt werden, die später die Intelligenz des Spieles widerspiegelt. Die abstrakten Vorgänge wie die Suche nach einem besten Zug werden vom Framework übernommen. Mittels jGameAI soll also nun ein intelligenter Computerspieler für „Die Siedler von Catan“ entwickelt werden, der mit Hilfe von spieltheoretischen Suchalgorithmen stets den „besten“ Zug durchführt.
Neben computergesteuerten Spielern sollen auch menschliche Spieler am Spiel teilnehmen dürfen, um sich gegen andere menschliche oder computergesteuerte Spieler zu messen. Die Bedienung des Spiels kann zwar an einem herkömmlichen PC mittels Maus und Tastatur erfolgen, diese Variante soll aber nicht die bevorzugte Interaktionsart sein. Stattdessen soll das Computerspiel ebenso als Brettspiel auf einem Tisch präsentiert werden. Dafür soll ein Tisch konzipiert werden, dessen Oberfläche ein Display repräsentiert und die Anwendung darstellt. Für die In-

teraktion mit dem Tisch sollen die Finger der einzelnen Spieler benutzt werden, wie es bei einem üblichen Brettspiel der Fall ist. Dafür soll die Oberfläche Multi-Touch-sensitiv gemacht werden, um mehrere Aktionen der Spieler gleichzeitig zu interpretieren. Multi-Touch ist hierbei ein modernes Bedienkonzept für grafische Benutzeroberflächen, welches in der Praxis stark im Kommen ist und in Zukunft für gewisse Anwendungsbereiche die übliche Bedienung mit Maus und Tastatur ersetzen wird [8, 18]. Mit Hilfe von Multi-Touch können nun intuitive und interaktive Benutzeroberflächen erschaffen werden, wie zum Beispiel bedienbare Tische oder ganze Tafelwände. Mit Hand, Finger oder Objekten können diese gesteuert werden und bieten somit neuartige Konzepte der Gebrauchstauglichkeit (engl. Usability). Das Ziel ist es, die Computerversion möglichst realistisch nah am Original zu halten, aber trotzdem durch intelligente Agenten oder Animationen einen Mehrwert zu erschaffen.

Im Zuge dieser Diplomarbeit sollen diese Aspekte kombiniert werden, um die Grundlage zu schaffen mit intuitiven Eingaben gegen intelligente Computergegner oder menschlicher Mitspieler eine Partie von „Die Siedler von Catan“ zu spielen [19]. Der Entwurf des Tisches ist dabei nicht auf diese bestimmte Anwendung ausgerichtet. Neben Java Settlers soll sie noch viele andere Anwendungsprogramme unterstützen, die über Multi-Touch-Eingaben gesteuert werden können. Dies soll realisiert werden, indem klare Schnittstellen zwischen Software und Hardware geschaffen werden. Damit stellt der Tisch eine Basis für weitere Arbeiten in diesem Gebiet dar. Zudem ist das Ganze eine Machbarkeitsstudie für Multi-Touch-Systeme im privaten Alltag, die Anstoß geben soll für weitere großflächige Projekte in anderen Anwendungsgebieten.

# 2 Verwandte Arbeiten

In diesem Kapitel sollen einige verwandte Themen und Arbeiten besprochen werden, die für das Verständnis dieser Diplomarbeit hilfreich sind. Im Folgenden wird das Spiel „Die Siedler von Catan“ detailliert anhand den offiziellen Regeln vorgestellt. Zudem werden die wichtigsten Themen der Spieltheorie zusammengefasst, die die Basis für eine KI eines intelligenten Agenten stellen. Danach wird das Framework JGameAI erläutert und letztlich werden verschiedene Techniken hinter dem Konzept von Multi-Touch vorgestellt.

## 2.1 Das Brettspiel „Die Siedler von Catan“

Im Folgenden wird der Spielaufbau, der Spielbeginn, der generelle Spielablauf nach Beginn des Spiels, das Spielende sowie einige Sonderregeln anhand der offiziellen Regeln erklärt. Danach werden generelle spieltypische Strategien vorgestellt.

### 2.1.1 Spielaufbau

„Die Siedler von Catan“ ist ein Brettspiel ohne festes Spielbrett. Stattdessen wird das Spielbrett für jedes Spiel neu aus insgesamt 37 sechseckigen Feldern (im nachfolgenden Hex genannt) zusammengebaut. Die Hexes in Abbildung 1 symbolisieren dabei verschiedene Landschaften, die bestimmte Erträge an Rohstoffen abwerfen:

- Wald wirft den Rohstoff Holz ab
- Weideland wirft den Rohstoff Wolle ab
- Ackerland wirft den Rohstoff Getreide ab
- Hügelland wirft den Rohstoff Lehm ab
- Gebirge wirft den Rohstoff Erz ab
- Wüste wirft keinen Rohstoff ab
- Meer wirft keinen Rohstoff ab
- Hafenfeld wirft keinen Rohstoff ab (kann aber für einen Handel mit dem Hafen eingesetzt werden)

Zudem befinden sich auf den Hexes, die Rohstoffe abwerfen können, jeweils ein Zahlenchip nummeriert von 2 bis 12. Auf der Wüste befindet sich kein Zahlenchip, stattdessen steht zum Spielanfang eine schwarze Räuberfigur auf dem Feld.
Im klassischen Spiel werden mit den Hexes eine Insel erstellt, die von Meer und Häfen umgeben ist, wie in Abbildung 2 gezeigt. Alternativ kann aber auch ein eigenständiges Spielfeld genutzt werden mit z.B. mehreren kleinen Inseln.

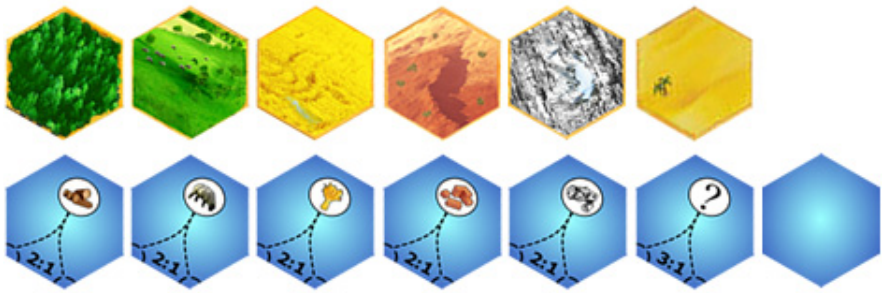


Abbildung 1: Hexes in Java Settlers (o.l. nach u.r.): Holz, Wolle, Getreide, Lehm, Erz, Wüste, Hafen Holz, Hafen Wolle, Hafen Getreide, Hafen Lehm, Hafen Erz, Hafen Normal, Meer

Abbildung 2: Beispielaufbau vom Brettspiel „Die Siedler von Catan“: Es wurde eine große Insel erstellt, umgeben von Meer und Häfen. Siedlungen, Städte und Straßen wurden bereits errichtet.

Ziel des Spieles ist es 10 Siegpunkte er erlangen. Dabei gibt es verschiedene Möglichkeiten Siegpunkte zu erreichen:

- Für den Bau einer Siedlung gibt es einen Siegpunkt
- Für den Bau einer Stadt gibt es zwei Siegpunkte (eine Stadt muss eine Siedlung ersetzen)
- Pro ausgespielter Entwicklungskarte, die einen Siegpunkt repräsentiert, gibt es einen Siegpunkt
- Für die längste zusammenhängende Handelsstraße (ab 5 Straßen) gibt es die Sonderkarte *Längste Handelsstraße* und zwei Siegpunkte
- Für die größte Anzahl an Ritterkarten (ab 3 Ritterkarten) gibt es die Sonderkarte *Größte Rittermacht* und zwei Siegpunkte

Zum Errichten von Straßen, Siedlungen und Städten sowie zum Erwerb von Entwicklungskarten benötigt es nun die bereits erwähnten Rohstoffe:

- Straße: 1xHolz + 1xLehm
- Siedlung: 1xHolz + 1xWolle + 1xGetreide + 1xLehm
- Stadt: 2xGetreide + 3xErz
- Entwicklungskarte: 1xGetreide + 1xWolle + 1xErz

Eine Straße kann nur an Kanten (im nachfolgenden Edge genannt) von zwei benachbarten Hexes gebaut werden, während Siedlungen und Städte nur an Kreuzungen (im nachfolgenden Node genannt) von zwei oder drei benachbarten Hexes gebaut werden können.
In jeder Runde wird mit 2 Würfeln entschieden, welche Hexes Rohstoffe abwerfen. Wird z.B. eine 9 gewürfelt, so werfen alle Hexes mit dem Zahlenchip 9 ihren jeweiligen Rohstoff ab. Die Rohstoffe erhalten aber nur diejenigen Spieler, die an dem Hex eine Siedlung oder eine Stadt errichtet haben. Die folgende Abbildung 3 aus Java Settlers zeigt ein Beispiel für die Verteilung der Rohstoffe bei einem Würfelwurf.


Abbildung 3: Beispiel zur Rohstoffverteilung zu einem Würfelwurf:
Spieler Blau hat eine Siedlung an einem Hafenstandort errichtet. Blau hat Zugang zu den Rohstoffen Getreide bei einer gewürfelten 3 oder Wolle bei einer 9. Spieler Weiß hat etwas südlich davon eine Siedlung, wofür er bei einer 6 an Erz gelangt oder ebenfalls mit einer 9 Wolle erhält. Bei einer gewürfelten 6 erhält Spieler Weiß einmal Erz und Spieler Blau geht leer aus. Bei einer gewürfelten 9 erhalten beide Spieler je einmal Wolle.

Zusätzlich kann ein Spieler stets mit anderen Mitspielern Rohstoffe handeln und somit Rohstoffe untereinander tauschen. Mit den Rohstoffen müssen neue Gebäude errichtet werden, um somit mehr Siegpunkte zu erlangen und an weitere Rohstoffe zu gelangen. Diese Gebäude werden (unter bestimmten Regeln) auf dem öffentlichen Spielbrett platziert. Die Rohstoffkarten jedes Spielers werden verdeckt in der Hand gehalten.

### 2.1.2 Spielbeginn

Nachdem das Spiel aufgebaut wurde, wählt jeder Spieler eine Farbe (weiß, rot, blau, grün), wovon er sich die Straßen, Siedlungen und Städte vor sich liegen hat. Jeder Spieler erhält eine Baukostenkarte, die Informationen bereitstellt, mit welchen Rohstoffen ein Spieler welche Gebäude bauen kann. Die Sonderkarten Längste Handelsstraße und Größte Rittermacht werden neben dem Spielplan bereitgelegt, ebenso die Würfel. Die Entwicklungskarten werden gemischt und als verdeckter Stapel bereitgelegt.
Jeder Spieler würfelt nun einmal mit den beiden Würfeln - der Spieler A mit der höchsten Augenzahl darf beginnen. Spieler A darf nun eine Siedlung an einem beliebigen Node auf dem Spielbrett platzieren. Zudem darf er eine benachbarte Straße an eine Edge platzieren, die an diesem Node liegt. Danach gibt der Spieler A weiter an den nächsten Spieler B im Uhrzeigersinn. Ist der letzte Spieler D an der Reihe und hat seine Siedlung und Straße platziert, so darf er dies erneut und gibt dann entgegen dem Uhrzeigersinn wieder zurück. Schlussendlich müssten alle Spieler je 2 Siedlungen mit je einer anhängenden Straße auf dem Spielbrett besitzen. Von der zuletzt platzierten Siedlung erhält jeder Spieler nun die Rohstoffe der benachbarten Hexes. Jeder Spieler hat somit bereits zum Spielbeginn 2 Siegpunkte und benötigt noch weitere 8 zum Sieg. Nun ist Spieler A an der Reihe und darf seine Runde beginnen (siehe Kapitel 2.1.3).

### 2.1.3 Spielablauf

Ist ein Spieler an der Reihe hat er in der angegebenen Reihenfolge von Phasen unterschiedliche Aktionsmöglichkeiten: Würfelphase, Handelsphase und Bauphase. Dabei darf von einer späteren Phase nicht mehr zurück in eine frühere Phase gesprungen werden.

#### Würfelphase

Der Spieler beginnt seinen Zug, indem er beide Würfel wirft und somit die Rohstofferträge dieses Zuges ermittelt. Jeder Spieler, der eine Siedlung errichtet hat, die an eines der ausgewürfelten Hexes grenzt, nimmt sich dafür eine Rohstoffkarte dieses Feldes von der Bank. Hat ein Spieler eine Stadt errichtet, so darf er sich gleich zwei Rohstoffkarten nehmen.

#### Handelsphase

Der aktuelle Spieler darf nun handeln, um Rohstoffkarten zu tauschen. Er hat die Möglichkeit mit allen Spielern zu tauschen und kann mitteilen, welche Rohstoffe er benötigt und was er bereit ist, dafür abzugeben. Er kann sich aber auch die Vorschläge seiner Mitspieler anhören und Gegenangebote machen. Es ist zu beachten, dass nur der Spieler, der an der Reihe ist, handeln darf. Andere Spieler dürfen nicht untereinander tauschen.

Alternativ gibt es für den aktuellen Spieler auch die Möglichkeit mit der Bank zu tauschen. Grundsätzlich kann ein Spieler immer im Verhältnis 4:1 tauschen, indem vier gleiche Rohstoffkarten gegen eine Rohstoffkarte seiner Wahl mit der Bank getauscht werden. Hat der Spieler eine Siedlung oder eine Stadt an einem Hafenstandort (zwei ausgewählte Node an einem Hafenfeld) errichtet, so kann er im Verhältnis 3:1 oder bei Spezialhäfen mit bestimmten Rohstoffen 2:1 tauschen.

### Bauphase

Schließlich kann der aktuelle Spieler Gebäude bauen, um die Anzahl seiner Siegpunkte zu erhöhen.

Hat der Spieler noch freie Straßen in der Hand und die erforderlichen Rohstoffkarten zum Bau einer Straße, so kann eine neue Straße an eine bestehende eigene Straße, Siedlung oder Stadt angelegt werden. Sobald ein Spieler eine durchgehende Straße aus mindestens fünf Einzelstraßen besitzt, erhält er die Sonderkarte Längste Handelsstraße. Gelingt es einem anderen Spieler, eine längere Straße als der jeweilige Besitzer zu bauen, wechselt die Sonderkarte.

Hat der Spieler noch freie Siedlungen in der Hand und die erforderlichen Rohstoffkarten zum Bau einer Siedlung, so kann eine neue Siedlung an einem Node errichtet werden, wenn eine eigene Straße zu dem Node führt und wenn die drei angrenzenden Nodes nicht von Siedlungen oder Städten besetzt sind. Jede Siedlung zählt einen Siegpunkt und gibt dem Besitzer die Möglichkeit, an neue Rohstofferträge aus angrenzenden Landfeldern zu gelangen.

Hat der Spieler noch freie Städte in der Hand und die erforderlichen Rohstoffkarten zum Bau einer Stadt, so kann eine eigene Siedlung zu einer Stadt ausgebaut werden. Dafür geht die Siedlung zurück in die Hand und wird ersetzt durch eine Stadt. Jede Stadt zählt zwei Siegpunkte und gibt dem Besitzer die Möglichkeit, doppelt so viele Rohstofferträge aus angrenzenden Landfeldern zu erhalten als mit der Siedlung.

Sind noch Entwicklungskarten vom Stapel vorhanden und hat der Spieler die erforderlichen Rohstoffkarten zum Erwerb einer Entwicklungskarte, so kann die oberste Entwicklungskarte vom Stapel gekauft werden. Gekaufte Entwicklungskarten hält ein Spieler bis zur Verwendung geheim, so dass kein Mitspieler irgendwelche Schlüsse daraus ziehen kann.

Die Trennung von Handels- und Bauphase dient der klaren Strukturierung des Spiels für Neulinge. Im Allgemeinen wird die Handelsphase mit der Bauphase kombiniert, so dass in beliebiger Reihenfolge gehandelt und gebaut werden kann.

Falls der Spieler, der an der Reihe ist, keine weiteren Züge mehr machen kann oder will, so wird an den linken Nachbarn weitergegeben.

### 2.1.4 Sonderregeln

Zu den erwähnten elementaren Regeln, treten zusätzlich einige Sonderregeln in bestimmten Situationen ein.

#### Sieben gewürfelt - Räuber wird aktiv

Würfelt ein Spieler eine 7, erhält keiner der Spieler Rohstofferträge (denn es gibt auch keinen Zahlenchip 7). Stattdessen müssen alle Spieler, die mehr als 7 Rohstoffkarten auf der Hand besitzen, die Hälfte (abgerundet) davon an die Bank abgeben. Der Spieler, der an der Reihe ist, muss nun den Räuber auf den Zahlenchip eines beliebigen anderen Hexes versetzen. Danach entnimmt er einem Mitspieler seiner Wahl, der eine Siedlung oder Stadt an diesem Hex hat, eine Rohstoffkarte. Der Spieler, der beraubt wird, hält dabei seine Rohstoffkarten verdeckt in der Hand, während der andere Spieler ihm eine Rohstoffkarte entzieht. Wird im späteren Verlauf die Zahl gewürfelt, auf dem der Räuber steht, erhalten die Besitzer der angrenzenden Siedlungen und Städte von dem Räuberfeld keine Rohstofferträge. In diesem Fall verhindert der Räuber die Rohstofferträge und bleibt auf dem Feld solange stehen, bis er wieder versetzt wird. Da die Wahrscheinlichkeit eine 7 zu würfeln am größten ist, soll diese Regeln verhindern, dass Spieler versuchen viele Rohstoffe lange in ihren Händen zu behalten.

#### Entwicklungskarte spielen

Es gibt drei verschiedene Arten von Entwicklungskarten mit unterschiedlichen Auswirkungen: Ritter, Fortschritts- und Siegpunktkarten. Der Spieler, der an der Reihe ist, darf zu einem beliebigen Zeitpunkt eine Entwicklungskarte ausspielen. Eine Karte, die der Spieler erst in diesem Zug gekauft hat, ist jedoch nicht erlaubt. Zudem ist es verboten in einem Zug mehr als eine Ritter- oder Fortschrittskarte auszuspielen.
Deckt der aktuelle Spieler eine Ritterkarte auf, so muss der Spieler, ähnlich wie bei der gewürfelten 7, den Räuber versetzen und einem Mitspieler eine Rohstoffkarte rauben. Die ausgespielten Ritterkarten bleiben offen vor dem Besitzer liegen. Wer zuerst drei Ritter vor sich liegen hat, erhält die Sonderkarte Größte Rittermacht, die zwei Siegpunkte wert ist. Schafft es ein anderer Spieler, mehr Ritterkarten vor sich zu sammeln als der jeweilige Besitzer, wechselt die Sonderkarte den Besitzer.
Spielt der Spieler eine Fortschrittskarte aus, so führt er die Anweisung auf der Karte aus. Bei einer Straßenbaukarte kann der Spieler sofort und ohne Rohstoffkosten zwei neue Straßen nach den üblichen Regeln auf dem Spielbrett platzieren. Ist es eine Erfindungskarte, darf der Spieler sich zwei beliebige Rohstoffkarten von der Bank nehmen. Bei einer Monopolkarte wählt der Spieler einen Rohstoff aus und alle Mitspieler müssen ihm alle Rohstoffkarten dieser Sorte geben. Eine Fortschrittskarte ist nach dem Ausspielen aus dem Spiel zu nehmen.

Karten mit Siegpunkten werden erst am Ende des Spieles aufgedeckt, wenn ein Spieler damit 10 oder mehr Siegpunkte erreicht und damit das Spiel beendet. Dabei dürfen beliebig viele Siegpunktkarten auf einmal aufgedeckt werden. Kauft ein Spieler eine Siegpunktkarte, mit der er seinen 10. Siegpunkt erreicht, darf er diese Karte auch sofort ausspielen.

#### Unterbrechung von Handelsstraßen

Eine Handelsstraße eines Spielers kann durch einen Mitspieler unterbrochen werden, indem er eine Siedlung auf einem freien Node der Handelsstraße errichtet. Die lange Handelsstraße zerfällt in diesem Fall in zwei kleinere, wie das Beispiel in Abbildung 4 zeigt. Haben nach der Unterbrechung einer Handelsstraße mehrere Spieler gleich lange Handelsstraßen, so wird die Sonderkarte Längste Handelsstraße weggelegt. Die Sonderkarte wird ebenfalls weggelegt, wenn kein Spieler mehr eine längste Handelsstraße besitzt. Sie kommt erst wieder ins Spiel, wenn ein Spieler alleine die längste Handelsstraße besitzt.


Abbildung 4: Beispiel einer Unterbrechung von Handelsstraßen:
Spieler Blau hat eine durchgängige Handelsstraße von 5 Straßen von seiner westlichen Stadt bis zur seiner östlichen Siedlung. Damit würde Spieler Blau die Längste Handelsstraße besitzen. Jedoch hat es Spieler Weiß geschafft, eine Siedlung auf dem Weg dieser Handelsstraße zu platzieren. Somit zerfällt die lange Handelsstraße von Spieler Blau in zwei kleinere Handelsstraßen mit Länge 2 bzw. 3. Spieler Blau muss die Sonderkarte Längste Handelsstraße abgeben.

### 2.1.5 Spielende

Ist ein Spieler an der Reihe und erreicht 10 oder mehr Siegpunkte, ist das Spiel sofort beendet und der Spieler hat gewonnen. Für schnellere Spiele kann die zu erreichende Anzahl an Siegpunkten auch herunter gesetzt werden.

### 2.1.6 Generelle Strategien

Es gibt einige vielversprechende Strategien für „Die Siedler von Catan“, jedoch keine allgemein beste Strategie. Durch die Vielfalt der Möglichkeiten und Faktoren, die außerhalb der Kontrolle der Spieler liegen, kann es keine allgemeine Formel für einen Sieg in jeder Situation geben. Stattdessen ist es oft hilfreich, das Spiel flexibel anzugehen, Vor- und Nachteile einer eigenen Strategie zu erkennen und die Strategie im Verlaufe des Spiels gegebenenfalls anzupassen. In diesem Kapitel soll kein vollständiger Überblick für Strategien gegeben werden, sondern nur die bekanntesten und wichtigsten Strategien kurz zusammengefasst werden.

Bevor eine Strategie ausgewählt wird, sollten die wichtigsten Faktoren im Spiel analysiert werden: Siegpunkte, Rohstoffe und Rohstoffertrag.

#### Siegpunkte und Rohstoffe

Die Anzahl der **Siegpunkte** ist der wichtigste Indikator, wie gut ein Spieler im Spiel steht. Zwar spielen noch andere Faktoren eine Rolle, aber ein Spieler kann das Spiel als ein Rennen bis zu 10 Siegpunkten ansehen. Wer mehr Siegpunkte als die anderen hat, ist näher am Ziel. Zwar wird der Sieg anhand der Siegpunkte entschieden, die Voraussetzung für Siegpunkte sind aber die **Rohstofferträge**. Somit bilden die Rohstoffe, die ein Spieler erhalten kann, das wichtigste Element zum Voranschreiten im Spiel und bestimmen somit die Geschwindigkeit, mit der sich der Spieler gen Ziel bewegt.

#### Exponentieller Rohstoffertrag

Die Anfangsphase ist die wichtigste Phase im Spiel. Die Begründung dafür ist, dass der Rohstoffertrag als eine **exponentielle** und nicht als eine lineare Funktion verstanden werden muss. Bei einem linearen Ertrag würde ein Spieler pro Runde im Durchschnitt gleich viele Rohstoffe erhalten. Bei Siedler können die Rohstoffe in Siedlungen oder Städte reinvestiert werden, die wiederrum Rohstoffe produzieren. Der Gewinn ist also exponentiell, sodass ein kleiner Vorteil zum Beginn des Spiels zu einem großen Vorteil gegen Ende des Spiels führen kann. Dies zeigt sich häufig in vielen Spielen, in denen die führenden Spieler mit der Zeit immer mehr die anderen Spieler hinter sich lassen. Das führt dazu, dass die **Produktivität** ein extrem entscheidender Indikator dafür ist, wo ein Spieler zu Beginn seine Siedlungen platzieren sollte. Die Längste Handelsstraße oder die Größte Rittermacht ist zu Beginn zu vernachlässigen.

Aus dieser Erkenntnis heraus ergeben sich vier oft gespielte Strategien: die Erz-Getreide-Strategie, die Holz-Lehm-Strategie, die Monopol-Strategie und die Produktivitäts-Strategie.

#### Erz-Getreide-Strategie

Diese Strategie versucht sich frühzeitig viel Erz und Getreide zu sichern, um Städte zu errichten, die den Rohstoffertrag verdoppeln. Dabei sollte darauf geachtet werden, sich mehr dem Erz zu widmen, da dieses dreifach benötigt wird (Getreide nur zweifach) und Erz in der klassischen Spielvariante seltener auf dem Spielbrett vorkommt. Im weiteren Verlauf des Spiels gilt es den Überschuss an Erz und Getreide in andere Rohstoffe umzutauschen, um entweder zu einem geeigneten Hafen zu expandieren oder um Entwicklungskarten zu erwerben. In den meisten Fällen zielt ein Spieler darauf ab auch die größte Rittermacht zu ergattern. Dies ist in zweierlei Hinsicht wichtig, da der Spieler einerseits bereits zwei Rohstoffarten für Entwicklungskarten besitzt und andererseits durch den hohen Rohstoffertrag oftmals Opfer vom Räuber wird. Durch die Ritterkarten ist ein Spieler in der Lage den Räuber wieder wegzuschicken.

#### Holz-Lehm-Strategie

Eine andere Strategie versucht möglichst viel Holz und Lehm zu sichern, um schneller Straßen und weitere Siedlungen zu errichten. Um an die anderen drei Rohstoffe zu gelangen, ist ein Hafen meist sehr hilfreich. Ein Spieler sollte beachten möglichst Richtung Erz oder Getreide zu expandieren, da der Spieler sonst gegen Ende des Spiels zu lange auf Rohstoffe für Städte warten muss. In den meisten Fällen führt diese Strategie auch zur Längsten Handelsstraße. Durch den hohen Expansionsgrad ist ein Holz-Lehm-Spieler in der Lage andere Spieler einzubauen oder deren Expansionen einzudämmen. Problematisch wird es jedoch, wenn zwei oder mehrere Holz-Lehm-Spieler aufeinandertreffen und sich gegenseitig blockieren.

#### Monopol-Strategie

Mit der Monopol-Strategie wählt ein Spieler sich eine oder einige Rohstoffe aus, über die der Spieler den exklusiven Zugang kontrolliert. Diese Strategie bietet sich meist an, wenn derjenige Spieler nicht einer der ersten Spieler ist, die die Anfangssiedlungen platzieren. Denn so kann sich der Spieler ein etwaiges Bild machen, was die anderen Spieler versuchen und sich in Ruhe einen Monopol-Rohstoff bestimmen. Der Monopolist steht sehr oft in Verhandlungen mit anderen Spielern, die gerne den Monopol-Rohstoff haben wollen. In der Regel kann durch die verbesserte Handelsposition ein vorteilhafterer Handel entstehen. Andererseits ist der Spieler oft die Zielscheibe für den Räuber. Genau wie bei der Erz-Getreide Strategie ist es wichtig, durch viele Ritterkarten den Räuber von einem selber fern zu halten.

### Produktivitäts-Strategie

Die einfachste Strategie ist es, in alle fünf Rohstoffe zu investieren und einzig auf die Produktivität der Zahlenchips zu achten. Die 6 und die 8 die häufig auftreten sind besonders beliebt. Mit dieser Strategie ist ein Spieler meist unabhängig und nicht so sehr auf den Handel mit Mitspielern angewiesen. Zudem ist es relativ einfach, die Strategie im Verlauf des Spiels an eine der oberen drei Strategien anzupassen.

Neben der gewählten Strategie sollte ein Spieler jedoch auch andere Faktoren berücksichtigen:

### Würfelphänomene

Die Zahlen, die gewürfelt werden sind zwar theoretisch normalverteilt (siehe Tabelle 1), jedoch nicht in ihrer Wirkung. Über die Spieldauer hinweg wird nicht oft genug gewürfelt, sodass jede Zahl entsprechend ihrer Wahrscheinlichkeit dran kommt. Gemäß dem Gesetz der kleinen Zahlen [20] als Konsequenz aus der Poisson-Verteilung, führt die Gleichwahrscheinlichkeit nicht zu einer Gleichverteilung bei einer kleinen Menge an Stichproben. Zudem sind aufgrund der exponentiellen Ertragsrate Zahlen zu Beginn des Spiels deutlich wichtiger als zum Ende hin. Auch ein Räuber stellt zu Beginn des Spiels eine größere Bedrohung dar, als gegen Ende des Spiels. Weniger Rohstoffe zu Beginn werfen einen Spieler stärker nach hinten, da keine Rohstoffe zum Reinvestieren bereit stehen.

| Würfelzahl | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Wahrscheinlichkeit | $\frac{1}{36}$ | $\frac{2}{36}$ | $\frac{3}{36}$ | $\frac{4}{36}$ | $\frac{5}{36}$ | $\frac{6}{36}$ | $\frac{5}{36}$ | $\frac{4}{36}$ | $\frac{3}{36}$ | $\frac{2}{36}$ | $\frac{1}{36}$ |

Tabelle 1: Wahrscheinlichkeiten der Würfelwürfe

Manchmal erscheint es einem Spieler, dass eine gewisse Würfelzahl oftmals hintereinander vorkommt und dann lange Zeit gar nicht. Dies passiert in der Regel einige Male im Spiel, obwohl es einem Spieler unwahrscheinlich erscheint. Dies mag darauf zurückzuführen sein, dass der Mensch zu sehr der Wahrscheinlichkeitsverteilung vertraut, wie folgendes Experiment zeigen kann: Eine Person erhält die Aufgabe 100-mal hintereinander eine Münze zu werfen und die Resultate aufzuschreiben. Eine andere Person soll eine zufällige Folge von 100 Münzwürfen aufschreiben. In über 90% der Fälle erhält der münzwerfende Spieler eine Reihe von 7 oder mehr aufeinanderfolgende Kopf oder Zahl der Münze. Der Spieler, der sich die Reihe ausgedacht hat, wird so etwas selten in seiner Liste finden, da er dies als sehr unwahrscheinlich empfindet. Für „Die Siedler von Catan“ bedeutet es auf solche Phänomene zu achten oder sich dagegen abzusichern.

### Zahlenverteilung

Die Zahlenchips werden in der Regel zufällig auf die Hexes verteilt, wobei eine 6 oder eine 8 nicht neben einer weiteren 6 oder 8 liegt. Gleiches gilt für 2 und 12. Zwar ergeben zwei Hexes mit je einer 5 dieselbe Wahrscheinlichkeit wie ein Hex mit 4 und eins mit 6, jedoch ist die Verteilung der Rohstoffe anders. Bei mehreren gleichen Zahlen erhält ein Spieler in einigen Runden vermehrt Rohstoffe und in vielen anderen kaum etwas. Bei vielen verschiedenen Zahlen verteilen sich die Rohstofferträge mehr über die Runden hinweg. Dies kann Vor- als auch Nachteile mit sich bringen, die ein Spieler abwägen sollte.

### Straßenbau

Eine ganz spezielle Entscheidung ist die Frage, wohin die Straßen gebaut werden sollen. Meistens wird die Straße in Richtung einer produktiven Hex platziert, wobei sich viele Spieler dann dort um die Plätze streiten. Baut ein Spieler in das „Landesinnere“, so sollte der Spieler vor Augen halten, dass es dort etwas enger zugeht, sodass die meisten Spieler eher nach außen bauen, um einen Hafen zu ergattern.

### Längste Handelsstraße oder Größte Rittermacht

Es ist sehr schwierig ohne eine dieser Sonderkarten zu gewinnen. Je nach gewählter Strategie (Erz-Getreide oder Holz-Lehm) legt ein Spieler sich sowieso auf einer dieser beiden Ziele fest. Bei der Längste Handelsstraße sollte tunlichst darauf geachtet werden, nicht in Sackgassen oder dicht besiedelte Bereichen zu bauen. Zudem sollte der Spieler darauf achten, dass die Mitspieler nicht die Gelegenheit erhalten, einem die Längste Handelsstraße zu unterbrechen. Sollten viele Mitspieler um die Sonderkarte Konkurrenz machen, kann es auch hilfreich sein, zu der anderen Sonderkarte zu wechseln, auch wenn der Spieler nicht die richtige Strategie dafür hat.

### Häfen

Häfen spielen zu Beginn eine untergeordnete Rolle, aber werden immer wichtiger im Verlauf des Spiels. Ein Hafen gibt dem Spieler mehr Handelsmöglichkeiten und stellt somit einen Katalysator für die Rohstofferträge dar. Ein Spieler muss den richtigen Zeitpunkt im Spiel erwischen, um zu einem Hafen zu expandieren. Expandiert der Spieler zu früh, so verspielt er sich die Möglichkeiten für bessere Produktivität zu Spielbeginn und wird häufig zum Opfer vom Räuber. Ist der Spieler aber zu spät, amortisiert sich der Gewinn der Handelsmöglichkeiten eventuell nicht mehr. Da die Häfen alle an Meerfeldern liegen, sind sie leicht zu reservieren, indem jemand sie mit Hilfe von Straßen geschickt abblockt.

### Handel

Der Handel ist ein sehr wichtiger Faktor im Spiel, und wer geschickt handeln kann, verschafft sich einen großen Vorteil. Beim Handel gibt es für den Spieler drei Sachen zu betrachten: mit wem er handelt, über welche Rohstoffe er handelt und zu welchem Zeitpunkt er handelt. Ein Spieler kann stets nur mit dem Spieler handeln, der an der Reihe ist. Dabei ist ein Handel mit Spielern, die keine Bedrohung für einen darstellen, selbstverständlich. Bei den Rohstoffen muss abgewägt werden, welche ein Spieler gerne abgeben möchte und welche die Mitspieler wohl erhalten und was sie damit anstellen wollen. In dieser Hinsicht kann ein Mitspieler fast als ein 1:1 Hafen angesehen werden, aber auch ein 2:1 Handel ist manchmal vorteilhaft. Der Zeitpunkt des Handels ist entscheidend und wird häufig nicht beachtet. Aufgrund des exponentiellen Rohstoffertrags empfiehlt es sich frühzeitig zu handeln. Im weiteren Verlauf sollte ein Spieler bevorzugt handeln, wenn der Spieler an der Reihe ist oder bald an der Reihe sein wird. Dies erhöht die Chance, die gehandelten Rohstoffen sofort zu benutzen und nicht durch eine gewürfelte 7 oder ausgespielte Monopolkarte die Rohstoffe wieder zu verlieren.

### Entwicklungskarten

Zu den Entwicklungskarten ist zu wissen, dass es insgesamt 25 davon gibt, bestehend aus 14 Ritterkarten, 2 Straßenbaukarten, 2 Erfindungskarten, 2 Monopolkarten und 5 Siegpunktkarten. Es ist also recht wahrscheinlich, eine Ritterkarte zu erhalten, die ein Spieler dringend benötigt, um den Räuber von einem wegzuschicken. Eine Entwicklungskarte kann einen Spieler weit voran bringen. Der Spieler sollte aber nicht allzu viele zu Beginn erwerben, denn sie bringen kaum Produktivität. Die Straßenbaukarte kann sehr überraschend wirken für die Längste Handelsstraße. Es bietet sich an, diese auch zum Ende zu spielen, um damit das Spiel zu gewinnen. Gleiches gilt natürlich für die Siegpunktkarten.

### Räuber

Der Räuber kann eine frustrierende Rolle im Spiel spielen. Einerseits muss beachtet werden, wo er sich häufig aufhält und andererseits, wohin er hin platziert werden sollte. Hexes mit Zahlenchips 6 oder 8 sind genauso beliebt wie Hexes, die von vielen Spieler besetzt sind. Ein Spieler sollte also achtgeben, ob er sich selber zu diesen Hexes gesellt. Den Räuber selber sollte ein Spieler auf Rohstoffe platzieren, die er selber besitzt, denn dies erhöht die Seltenheit und somit den Wert des Rohstoffes. Genauso selbstverständlich ist es, den Räuber auf ein Hex zu platzieren, um den führenden Spieler zu schaden. Bei Spielern mit wenigen Entwicklungskarten ist die Wahrscheinlichkeit umso größer, dass der Räuber dort länger stehen bleibt. Neben dem Räuber sollte ein Spieler stets auf seine eigene Anzahl an Rohstoffen achten, damit bei einer gewürfelten 7 nicht die Hälfte davon an die Bank abgeben werden muss.

## 2.2 Verwandte Implementierungen

Wie bereits erwähnt, gibt es im Siedler-Universum bereits einige Versuche das Brettspiel in eine Computerversion zu überführen. Zunächst wäre da „Catan: The Computer Game“ zu nennen, welches von „Castle Hill Studios“ entwickelt und vom Verlag „Big Fish Games“ herausgebracht wurde [21]. Es war das erste offizielle Computerspiel bezüglich „Die Siedler von Catan“ und wies wahlweise 2 oder 3 Computerspieler auf, die in 3 verschiedenen Stufen gegen den Menschen antreten konnten. Jedoch war selbst die schwierigste Stufe leicht zu durchschauen, da sie recht lokale Entscheidungen getroffen hat. Das Spiel wurde später von Microsoft aufgekauft und kann noch heute über MSN Games bezogen werden [22]. Die Entwicklung wurde jedoch eingestellt.
Microsoft entwickelte stattdessen an einer Online-Version des Spiels, welches als „Catan Online“ im August 2005 veröffentlicht wurde [23]. Dieses erforderte eine monatliche Gebühr zum Spielen gegen andere Gegner rund um die Welt. Ein ähnliches Spiel wurde später auch für Microsoft X-Box und Nokia N-Gage herausgebracht.
Erste wissenschaftliche Auseinandersetzungen mit dem Spiel gab es 2002 von Robert S. Thomas, der in seiner Dissertation zum Ph.D. an der Northwestern University eine Künstliche Intelligenz (KI) für eine Java-Version entwickelte [24, 25]. Später wurde das Spiel ebenfalls namens „Java Settlers“ online gestellt, um empirische Daten von vielen Spielern zu erheben, heute sind die meisten Server aber wieder offline. Die KI spielte insgesamt akzeptabel, zeigte aber in vielen Situationen Schwächen. Dies mag darauf zurückzuführen sein, dass keine globale Suchstrategie verwendet wurde, sondern lediglich lokale Entscheidungen durch planbasierte Heuristiken herangezogen wurden.
Es gab zwischenzeitlich noch andere Implementierungen hobbymäßiger Art, wovon viele heute noch über das Webportal SourceForge zu finden sind [26]. Die meisten Versuche kamen aber nicht über einfache Spielintelligenz hinaus, in denen ein menschlicher Spieler auf einfache Weise den Computerspieler überwinden konnte. Jedoch haben diese Open-Source-Implementierungen dabei geholfen, Ideen für diese Arbeit umzusetzen. So wurden mit Erlaubnis des Projektadministrators aus dem Projekt „GL Catan“ viele Grafiken für diese Arbeit übernommen [27].

## 2.3 Spieltheoretische Grundlagen

Die Spieltheorie ist ein mathematischer Wissenschaftszweig, der „Probleme analysiert, die auftreten, wenn wir versuchen, in einer Welt vorauszuplanen, in der andere Agenten gegen uns planen“ [4]. In einer *Arbeitsumgebung* interagieren mehrere *Agenten* kooperierend oder konkurrierend zueinander und treffen *rationale Entscheidungen* bzw. *Aktionen* basierend auf ihren Beobachtungen. Für die Aktionen eines Agenten wird eine *Leistungsbewertung* ausgeführt, die die Güte der Aktionen angibt. Die Spieltheorie findet Anwendungen in der Wirtschafts- und

Politikwissenschaften, der Biologie, der Soziologie, der Psychologie oder der Künstlichen Intelligenz, um Modelle für intelligenten Agenten zu definieren. In der Künstlichen Intelligenz wurden viele traditionelle Spiele wie Schach, Dame, Backgammon, usw. erfolgreich als Agentensysteme modelliert.

### 2.3.1 Die Arbeitsumgebung

Folgende Definitionen für Arbeitsumgebungen sind informell und entstammen dem Buch Künstliche Intelligenz von Russell, Norvig [4]. Sie werden benutzt, um die Arbeitsumgebung für „Die Siedler von Catan“ zu beschreiben und Unterschiede zu den traditionellen Spielen aufzuweisen:

- Eine Umgebung mit nur einem Agenten wird als **Einzelagent-Umgebung** bezeichnet. Sind zwei Agenten beteiligt so gilt es als eine **Zwei-Agenten-Umgebung** oder bei mehreren Agenten eine **Multiagent-Umgebung** [4].
- In einer Zwei-Agenten-Umgebung kommt es häufig vor, dass ein Spieler seine Leistungsbewertung maximiert während die des anderen Spielers minimiert wird. Dies ist eine **konkurrierende** Umgebung, die typisch ist für sogenannte *Nullsummenspiele*. Wird durch eine Aktion jedoch die Leistungsbewertung mehrerer Agenten gleichzeitig maximiert, so handelt es sich um eine (vollständige oder partielle) **kooperative** Umgebung [4].
- Die Arbeitsumgebung wird **vollständig beobachtbar** genannt, wenn der Agent zu jedem beliebigen Zeitpunkt Zugriff auf den vollständigen Zustand der Umgebung besitzt. Sind jedoch Störungen, ungenaue Daten oder unvollständige Daten im Spiel, so ist die Umgebung **teilweise beobachtbar**. In diesem Kontext wird auch von **vollständiger** oder **unvollständiger Information** [4] gesprochen.
- Eine Arbeitsumgebung ist **deterministisch**, wenn der nächste Zustand der Umgebung vollständig durch den aktuellen Zustand und der vom Agenten durchgeführten Aktionen bestimmt werden kann, andernfalls ist sie **stochastisch**. Wenn jedoch die Umgebung bis auf die Aktionen anderer Agenten deterministisch ist, so wird sie als **strategisch** bezeichnet [4].
- Ist die Erfahrung des Agenten in atomare Episoden aufgeteilt, wo eine Aktion in einer Episode nur von dieser Episode abhängig ist und unabhängig von bisherigen Entscheidungen, so kennzeichnet dies eine **episodische** Arbeitsumgebung. Andernfalls gilt die Umgebung als **sequentiell**, wo ein vorausschauender Agent auch die Historie der Entscheidungen berücksichtigen muss [4].
- Ist die Umgebung in der Lage, sich während einer Entscheidungsfindung zu ändern, so ist sie für den Agenten **dynamisch**; andernfalls ist sie **statisch**. Wenn sich jedoch die Umgebung im Laufe der Zeit nicht ändert, aber die Leistungsbewertung des Agenten, so haben wie eine **semidynamische** Umgebung [4].

- Zusätzlich wird zwischen **diskreten** und **stetigen** Umgebungen unterschieden. Wird die Zeit in der Entscheidungen gefunden und Aktionen ausgeführt werden als diskret behandelt oder ist die Menge der möglichen Zustände, Wahrnehmungen und Aktionen endlich, so ist diese Umgebung diskret. Ist die Zeit oder sind die Zustände, Wahrnehmungen und Aktionen aber stetig, so wird auch die Umgebung als stetig bezeichnet [4].

Bei „Die Siedler von Catan“ haben wir ein Mehrspielerspiel und somit eine **Multiagenten-Umgebung**. Im Gegensatz zu Schach können wahlweise 3 oder 4 Personen teilnehmen, somit ist die Entscheidung von Sieg und Niederlage nicht mehr zweidimensional, sondern mehrdimensional. Dies ist ein Unterschied zu den traditionellen Nullsummenspielen, die in einer konkurrierenden Zwei-Agenten-Umgebung sich befinden. „Die Siedler von Catan“ ist also diesbezüglich kein klassisches Nullsummenspiel. Das Spiel zeigt eine **partiell kooperative** Umgebung, in der Aktionen (z.B. Handelszüge) einer Teilmenge von Spielern, aber nicht allen Spielern helfen. Für einen einzelnen Agenten sind zwar alle Informationen bezüglich sich selbst und dem öffentlichen Spielbrett erhältlich, jedoch sind die Informationen bezüglich der anderen Spieler verborgen. Somit muss der Agent Entscheidungen auf **unvollständigen Informationen** treffen und befindet sich somit in einer **teilweise beobachtbaren** Umgebung. Durch die Würfelzüge und andere wahrscheinlichkeitsbasierte Züge, wie das Ziehen einer Rohstoffkarte beim Räuberzug oder dem Erwerb von Entwicklungskarten, sind die Zustandsübergänge als **stochastisch** einzuordnen. Gleichermaßen ist es schwierig zu entscheiden, wie sehr eine unwahrscheinliche Aktion in die Leistungsbewertung mit einfließen soll. Die Entscheidungen des Agenten haben Auswirkungen auf den restlichen Spielverlauf, von daher ist die Umgebung **sequenziell**. Aufgrund des Rundenprinzips kann sich während einer Entscheidungsfindung die Spielsituation nicht ändern, jedoch die Leistungsbewertung schon, weshalb unsere Umgebung **semidynamisch** ist. Ebenfalls aufgrund des Rundenprinzips gibt es diskrete Zustände, womit die Umgebung auch **diskret** ist. Eine Zusammenfassung der Eigenschaften einer Arbeitsumgebung für „Die Siedler von Catan“ ist in der Tabelle 2 gegeben.

| Agenten? | Multiagent-Umgebung |
|---|---|
| Konkurrierend? | nein, partiell kooperativ |
| Beobachtbar? | teilweise beobachtbar |
| Deterministisch? | nein, stochastisch |
| Episodisch? | nein, sequenziell |
| Statisch? | nein, semidynamisch |
| Diskret? | ja |

Tabelle 2: Charakterisierung der Arbeitsumgebung für „Die Siedler von Catan“

### 2.3.2 Der Agent

> *„An agent is a computational system, situated in some environment, that is capable of intelligent, autonomous action in order to meet its design objectives.“ – Nicholas Jennings (Intelligent Agents, 1995) [28]*

Wenn von dem Agent die Rede ist, wird meist das Verhalten eines Agentenprogramms betrachtet, welches auf eine Anfrage eine Aktion berechnet. Dabei erhält der Agent die aktuelle Wahrnehmung von Sensoren als Eingabe und gibt eine Aktion als Ausgabe aus. Eine sehr einfache Form eines Agenten wäre alle möglichen Eingabe-Ausgabe-Paare in eine Tabelle abzuspeichern und die benötigte Aktion nachzuschlagen. Für die meisten relevanten Probleme ist dies aber nicht praktikabel, da der Speicherplatz für die Tabelle nicht realisierbar wäre und nicht immer alle Paare ermittelt werden können. Beim Schach zum Beispiel müssten bei 20 Züge pro Stellung, einer Halbzugtiefe von 16 und ohne Optimierungen ca. 600.000.000.000.000.000.000 mögliche Varianten betrachtet werden.
Ein etwas verbesserter, aber immer noch einfacher Agent, ist der **Reflex-Agent**. Diese Agenten bestimmen aufgrund der aktuellen Wahrnehmung und aus einer Menge einfacher Regeln eine durchzuführende Aktion [4]. Diese Agenten sind einfach und schnell, aber nur begrenzt intelligent, da sie weder die Vergangenheit noch die Zukunft in Betracht ziehen und sich nur lokal aufgrund einer kleinen Menge an Regeln entscheiden.
Ein verbesserter Reflex-Agent führt einen internen Zustand mit sich. Der sogenannte **Modellbasierte Reflex-Agent** versucht also sich die Welt aufgrund der Wahrnehmungshistorie zu modellieren und hat nun die Möglichkeit Entwicklungen und Auswirkungen auf das Modell zu übertragen und vom Modell basierend neue Aktionen zu ermitteln [4].
Damit auch Auswirkungen auf die Zukunft in Erwägung gezogen werden, benötigt es **Zielbasierte Agenten**, die zusätzlich noch Zielinformationen mit sich führen [4]. Damit ist es möglich, zielgerecht nach einer Aktion zu suchen, die den Agenten näher zum Ziel bringt.
Um zwischen Zuständen vergleichen zu können, die sich auf dem Weg zum Ziel befinden, wird eine Leistungsbewertung benötigt, die den Nutzen ermittelt. Solche **Nutzenbasierte Agenten** können durch Heuristiken schneller zum Ziel gelangen [4]. Dafür führen sie temporäre Aktionen auf ihrem internen Zustand aus, bewerten den neuen Zustand und ziehen die Aktion zurück. Somit können verschiedene Aktionsfolgen in Hinsicht auf die Leistungsbewertung simuliert werden und der beste ausgewählt werden. Zusätzlich können bei mehreren Zielen abgeschätzt werden, wie groß der Nutzen im Verhältnis zur Wahrscheinlichkeit ist oder bei in Konflikt stehenden Zielen abgewogen werden, welcher geeigneter ist.
Für Java Settlers soll ein Nutzenbasierter Agent entwickelt werden, der durch geeignete Heuristiken und einer Bewertungsfunktion das Ziel erreichen soll, das Spiel zu gewinnen. Dabei soll mit einer adversarialen Suche der bestmögliche Zug in einer vertretbaren Zeit ermittelt werden. Im Folgenden soll nun näher auf die Suche eingegangen werden.

### 2.3.3 Die Suche

Das Finden einer optimalen Aktion $a^*$ vom Agenten wird als Suche in einem Suchbaum formalisiert. Wir bezeichnen das Modell eines Zustandes oder einer Stellung als $S$. Ein Zugegenerator $A$ kann nun eine Menge aller möglichen Aktionen $\{a_1, a_2, \cdots a_k\}$ auf diesem Zustand generieren, wobei $k$ die Anzahl der legalen Aktionen darstellt. Jede Aktion $a_i$ definiert eine Übergangsfunktion zu einem Folgezustand $S_i$.
Gegeben ein Startzustand können nun Folgezustände für mögliche Spielverläufe bei der Suche generiert werden, was einen *impliziten Suchbaum* aufspannt (Abbildung 5). In diesem Suchbaum führen manche Pfade zu gleichen Zuständen, so dass eigentlich ein Graph vorliegt. Für einfache Spiele kann die Suche bis zu den Endzuständen vordringen, in denen fest steht, ob das Spiel gewonnen, verloren oder remis gespielt wird.
Für komplexere Spiele ist der Suchraum jedoch zu groß um in vertretbarer Zeit zu den Blattknoten zu stoßen. Von daher wird in der Regel nur bis zu einer bestimmten Suchtiefe $t$ vorgedrungen um die dortigen Zustände als Endzustände zu behandeln. Eine nutzenbasierte Bewertungsfunktion $f(S_t) = p$ bewertet eine terminale Stellung $S_t$ mit einem Nutzenwert $p$. Auf diese Weise können für $t = 2, 3, 4, ...$ Züge im voraus verschiedene Zustände betrachtet und miteinander verglichen werden.

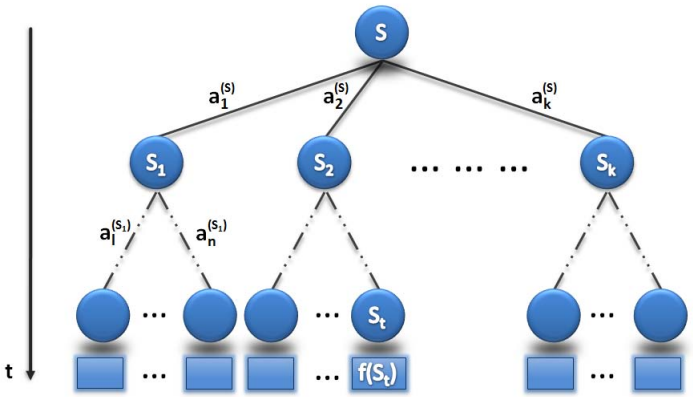



Abbildung 5: Allgemeiner Suchbaum

Ein Vergleich mit Schach illustriert erneut die Dimension des allgemeinen Suchbaums: Die Shannon-Zahl $10^{120}$ gibt laut der Analyse von Shannon eine untere Grenze für unterschiedliche Spielvariationen bei Schach an. Dabei treten ca. $10^{43}$ unterschiedlichen Zustände auf [29]. Victor Allis präzisierte die Schätzung später auf $10^{123}$ Spielvariationen und $5x10^{52}$ Zustände [30]. Der übliche Vergleich mit der Anzahl an Atomen im Universum ($10^{81}$) veranschaulicht die Komplexität. Für „Die Siedler von Catan“ existieren keine Schätzungen, jedoch kann aufgrund des höheren durchschnittlichen Verzweigungsgrades durch verdeckte Informationen, Zufallszüge und Handelsmöglichkeiten die Komplexität ähnlich oder höher wie Schach eingeschätzt

werden.

Für Zwei-Agenten-Umgebungen, insbesondere für Nullsummenspielen, ist der aus der Literatur bekannte **Minimax-Algorithmus** von John v. Neumann ein Verfahren für die Suche [31, 32]. Es wird ein Spielbaum erstellt, in dem abwechselnd für jede Ebene versucht wird, den maximalen bzw. den minimalen Nutzen der Kinderknoten nach oben weiterzureichen. Für den Spielbaum wird somit eine vollständige Tiefensuche durchgeführt, wobei die Zeitkomplexität bei einer Tiefe $m$ und $b$ erlaubten Zügen pro Knoten gleich $O(b^m)$ ist. Liegt ein Nullsummenspiel vor, ergibt sich immer ein sogenanntes *Nash-Gleichgewicht* [33, 34] in dem Spielbaum. Jeder Zug eines Agenten, der vom Nash-Gleichgewicht abweicht, kann den Agenten nur verschlechtern und nicht verbessern, wie das Beispiel in Abbildung 6 illustrieren soll.

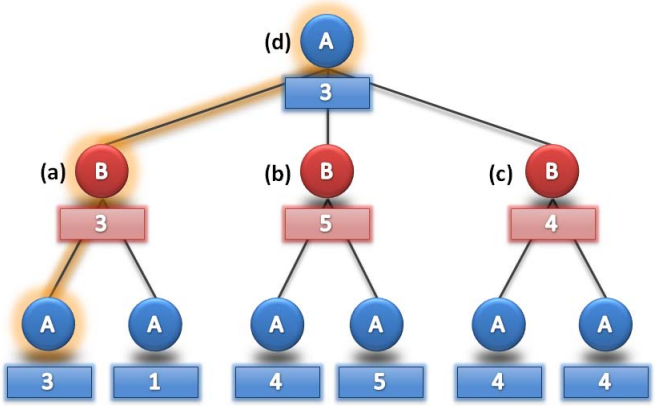



Abbildung 6: Beispiel für den Minimax-Algorithmus:
In diesem kleinen Suchbaum treten Spieler A und Spieler B gegeneinander an. Dabei versucht Spieler A den Nutzenwert zu minimieren, während Spieler B versucht ihn zu maximieren. Der Minimax-Algorithmus durchläuft den Suchbaum in einer Tiefensuche und evaluiert die Blattknoten. Der Spieler B wählt aus seinen Kinderknoten stets die Aktion mit dem maximalen Wert (a,b,c), während Spieler A die Aktion des Minimum auswählt (d). Es bildet sich ein Nash-Gleichgewicht für die Hauptvariante, die in orange hervorgehoben ist. Weicht Spieler A von dieser Hauptvariante ab, so kann sich die Stellung aus seiner Sicht nur verschlechtern und einen höheren Wert als 3 liefern. Weicht Spieler B von der Hauptvariante ab, so sinkt der Wert auf jeden Fall unter 3.

Wird Minimax auf mehrere Spieler übertragen, so wird der Nutzenwert in einen absoluten Nutzenvektor überführt. Die Nutzenfunktion $f(S_t) = \vec{p}$ bewertet einen Zustand $S_t$ durch einen Vektor $\vec{p} = <p_1, p_2, \cdots, p_n>$, wobei $n$ die Spieleranzahl darstellt und $p_i$ der Nutzenwert vom Spieler $i$ ist. Eine Unterscheidung zwischen Minimum und Maximum ist nicht mehr möglich. Stattdessen, wird nun in einer **max$^n$-Variante** versucht, für jeden Spieler seinen Nutzenwert zu maximieren [35, 36]. Ein Nash-Gleichgewicht muss jedoch nicht mehr unbedingt eintreten. Ein Beispiel in Abbildung 7 erklärt diesen Vorgang.

Dadurch, dass Siedler kein Nullsummenspiel und zudem kooperativ ist, verkompliziert sich die Suche umso mehr. Durch Zusammenschlüsse (Allianzen) können sich mehrere Spieler gegen-

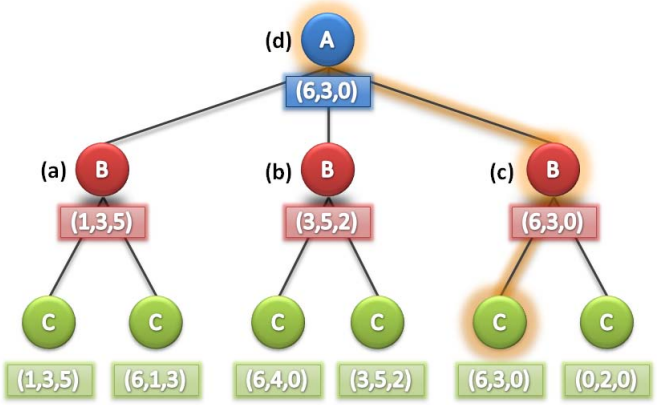



Abbildung 7: Beispiel für den max$^n$-Algorithmus:
In diesem Suchbaum gibt es drei Akteure: Spieler A, B, und C. Jeder Spieler versucht seinen eigenen Wert im Nutzenvektor zu maximieren. Spieler B wählt bei (a) die 3 gegenüber der 1, bei (b) die 5 gegenüber der 4 und bei (c) die 3 gegenüber der 2. Spieler A wird bei (d) die 6 wählen, da dies der höchste ist. Es entsteht wieder eine Hauptvariante, die in orange markiert ist. Ein Nash-Gleichgewicht besteht jedoch nicht mehr, denn je nach Spiel kann der Nutzenvektor (0,2,0) für Spieler B besser sein als (6,3,0). Für Spieler C wäre es auf jeden Fall besser.

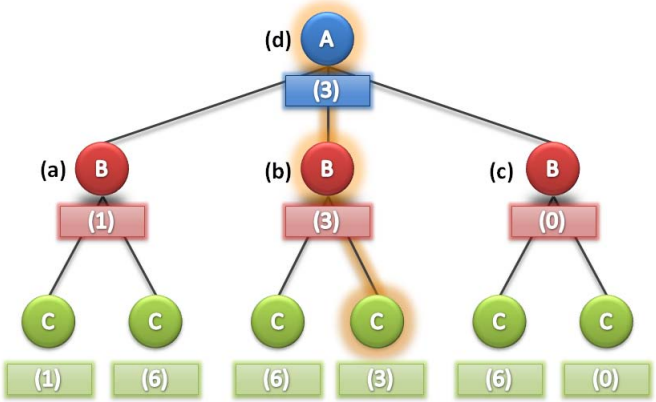



Abbildung 8: Beispiel für den paranoiden Algorithmus:
Für die paranoide Version des Suchbaums wird der Baum aus Abbildung 7 wiederverwendet. Spieler A verhält sich hier paranoid und nimmt an, dass Spieler B und C ihren Nutzenwert ignorieren und gemeinsam versuchen den Nutzenwert von Spieler A zu minimieren. Aus diesem Grund sind nur die Nutzenwerde von Spieler A eingetragen. Spieler B verhält sich bei (a) und (b) genauso wie vorhin, aber bei (c) wird der rechte Ast gewählt. Spieler A wird bei (d) nun den mittleren Ast mit Nutzenwert 3 wählen, denn dieser ist der beste, den er erhalten kann.

über einem anderen Spieler in einer stärkeren Position verbünden. Dieses Allianzen werden dynamisch im Laufe des Spiels gebildet und wieder aufgelöst. Die Frage ist jetzt, wie sich ein Spieler im Verhältnis zu anderen Spieler sieht, wofür es mehrere Möglichkeiten gibt. Ein **„paranoider“** Spieler geht immer davon aus, dass er alleine gegen die restlichen Spieler als Allianz spielt und somit alle gegnerischen Aktionen ihn benachteiligen [35, 37]. Damit wird das Mehrspielerproblem auf ein Zweispielerproblem reduziert. Die Abbildung 8 illustriert einen paranoiden Spieler.

Diese Variante ist für Java Settlers jedoch zu drastisch, denn tatsächlich wird mehr miteinander kooperiert, anstatt dass konkurrierend agiert wird. In einer Art und Weise repräsentiert der $\max^n$-Algorithmus eine komplett optimistische Sicht auf die Kooperationen, wo ein Spieler nicht auf die Gegner achtet, sondern nur auf sich selber schaut. Die paranoide Version dagegen zeigt eine komplett pessimistische Ansicht, wo der suchende Spieler alle Gegner als einen einzelnen sieht, der versucht die eigene Bewertung zu minimieren. Beide Varianten vergleichen Zustände miteinander, indem nur die absoluten Nutzenwerte des jeweiligen Spielers in Betracht gezogen werden. Stattdessen ist es sinnvoller den Nutzen relativ zu den Gegenspieler zu sehen. Dafür muss eine Umwandlung des absoluten Nutzenvektors in einen relativen Nutzenvektor erfolgen. Einige Lösungen dazu werden in Kapitel 5 vorgestellt.

Für Java Settlers wäre eine Zwischenlösung optimal, die den Nutzenwert des Spielers mit dem Durchschnitt der anderen Spieler vergleicht. Ziel des Spielers ist es also, sich möglichst weit abzusetzen gegenüber dem durchschnittlichen Rest, was auch die Interpretation des Spiels als Rennen widerspiegelt. Der Durchschnitt jedoch ist anfällig für Ausreißer und somit nicht gänzlich geeignet. Abhilfe schafft hier noch ein zusätzliches Entropie-Maß in Java Settlers, welches ebenfalls in Kapitel 5 erklärt wird.

Um die Laufzeit des Minimax-Algorithmus zu verbessern, kann der Spielbaum an geeigneten Stellen beschnitten werden. Das bekannte **Alpha-Beta-Pruning** berechnet während der Suche einen $\alpha$-Wert mit der bestmöglichen und einen $\beta$-Wert mit der schlechtmöglichsten Alternative [38]. Stellt sich bei einem Knoten heraus, dass die Kinder keine bessere bzw. schlechtere Bewertung mehr liefern können, als anhand des $\alpha$-Wert bzw. $\beta$-Wert, so kann der Ast abgeschnitten werden und benötigt nicht mehr berechnet zu werden. Abbildung 9 zeigt den Verlauf von einer Alpha-Beta-Kürzung. Bei einer theoretisch perfekten Sortierung der Knoten schafft Alpha-Beta die Laufzeit auf $O(b^{m/2})$ zu drücken [38]. Leider ist bis heute keine äquivalente Lösung für Mehrspieler bekannt, daher ist in Java Settlers keine Alpha-Beta-Kürzung vorgesehen.

Weitere Optimierungen bieten *Transpositionstabellen*, *Ruhesuche*, *Datenbanken* usw., auf die hier nicht näher eingegangen und stattdessen auf die Literatur verwiesen wird [39, 40, 41].

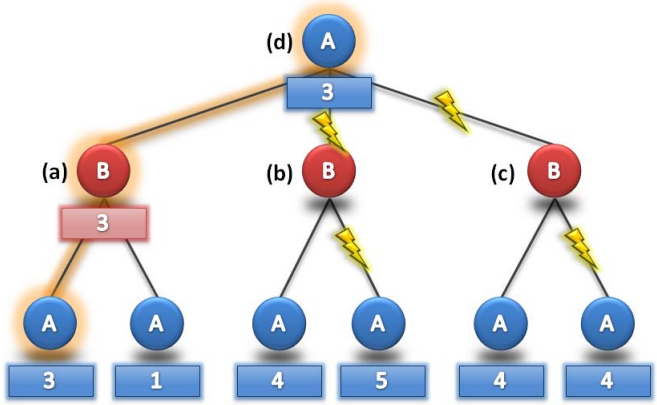



Abbildung 9: Beispiel für den Alpha-Beta-Algorithmus:
Wir verwenden erneut den Suchbaum aus Abbildung 6. Der Minimax-Algorithmus läuft erneut in Form einer Tiefensuche durch den Baum und ermittelt für Knoten (a) den maximalen Wert 3. Bei Knoten (b) würde zunächst ein Wert von 4 vom Kindknoten übernommen. Da aber bereits feststeht, dass Spieler A bei (a) keinen Wert größer als 3 annehmen wird, ist eine weitere Untersuchung von Knoten (b) überflüssig, denn Knoten (b) wird am Ende einen Wert von 4 oder größer enthalten. In diesem Fall findet ein alpha-Cut statt und der gesamte mittlere Ast kann abgeschnitten werden, denn er benötigt keine weiteren Untersuchungen. Analoges passiert bei Knoten (c).

### 2.3.4 Das Lernen

Im Jahr 1950 bedauerte Alan M. Turing, wie viel Arbeit es bedeutete, intelligente Maschinen manuell zu programmieren und wünschte sich eine schnellere Methode, lernende Maschinen zu erstellen und diese dann zu lehren [42]. Er leitete damit viele Forschungen im Bereich des sogenannten *maschinellen Lernens* ein [43, 4, 44]. Eine klassische Unterteilung des maschinellen Lernens ist in **überwachte**, **unüberwachte** und **verstärkende** Methoden [4].

Zu den überwachten Lernmethoden gehört stets ein „Lehrer“, der dem System Klassifikationen anhand von gegebenen Ein- und Ausgaben-Paaren lehrt. Dieser Lehrer kann eine menschliche Person sein, aber auch eine Bewertungsfunktion oder eine andere externe Klassifikationsmethode. Klassische Beispiele für überwachtes Lernen wären der *ID3-Algorithmus* zur Erstellung von *Entscheidungsbäumen* oder der *Backpropagation-Algorithmus* in *Neuronalen Netzen* [45, 46].

Beim unüberwachten Lernen versucht das System stattdessen selbstständig zu klassifizieren, in dem es ein Modell für Vorhersagen erzeugt. Verschiedene *Clustering-Verfahren* wie z.B. der *k-means-Algorithmus* sind passende Beispiele für unüberwachte Lernmethoden [47]. Interessanterweise stellt die Naturwissenschaft selber eine Art unüberwachtes Lernen dar, denn sie kann auf keinen Lehrer zurückgreifen. Stattdessen werden Hypothesen für Beobachtungen aufgestellt, diese bewertet und durch Experimente getestet [44].

Die Klasse der verstärkende Lernverfahren (Reinforcement-Lernen) lernt durch Belohnung oder Bestrafung einer gewählten Strategie für eine gegebene Situation. Ein nutzenbasierter Agent

z.B. würde stets versuchen seinen Nutzen zu maximieren. Eine Methode dies zu verwirklichen ist das Lernen durch Temporale-Differenz-Methoden [39, 44]. Reinforcement-Lernen ahmt in einer Art und Weise die Lernweise des Menschen nach, der mit seiner Umwelt interagiert und durch Rückmeldungen Rückschlüsse ziehen kann. Für Spiele wäre Reinforcement-Lernen sehr geeignet, da ein nutzenbasierte Agent meist bereits existiert.

## 2.4 Das generische Framework jGameAI

jGameAI ist ein an der Freien Universität Berlin entstandenes generisches Framework zur Erstellung von einer Vielzahl an intelligenten Spielen, welches dabei viele Aufgaben selbstständig erledigt [9, 10]. So wird der Suchalgorithmus automatisch angestoßen, sobald ein neuer bester Zug von einem Computerspieler angefordert wird. Ebenso gibt es Events und Benachrichtigen, die bei bestimmten Ereignissen ausgelöst werden, auf die der Anwender gesondert reagieren kann. Eine weitere Funktion von jGameAI ist es, beliebige Turniere oder Ligen zu erstellen, in der Computergegner gegeneinander antreten können und in Tausenden von Spielen sich langsam maschinell verbessern können. Dies kann genutzt werden, um verschiedene Implementierungen gegeneinander zu testen oder auch evolutionär die stärksten herauszufiltern. Der Computerspieler soll nach jeder gespielten Partie versuchen, sich zu verbessern, indem aus dem Ergebnis mit Hilfe von Reinforcement Learning maschinell gelernt wird. Dabei soll auch das Verhalten der Gegner mitgelernt werden, sodass ein Agent beim erneuten Aufeinandertreffen mit Hilfe eines *Opponent Models* auf erlernte Informationen zurückgreifen kann [48].
Das Framework unterstützt sowohl rundenbasierte als auch Echtzeitspiele. Um ein neues rundenbasiertes Spiel mit Hilfe dieses Framework zu implementieren, bedarf es lediglich weniger Bestandteile, die in Abbildung 10 abgebildet sind.

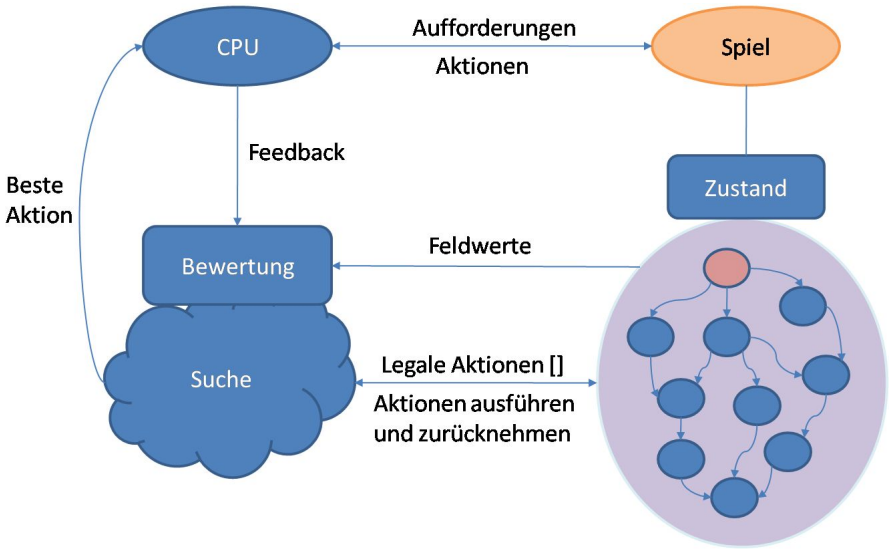



Abbildung 10: Übersichtsdiagramm für das jGameAI Framework [9, 10]

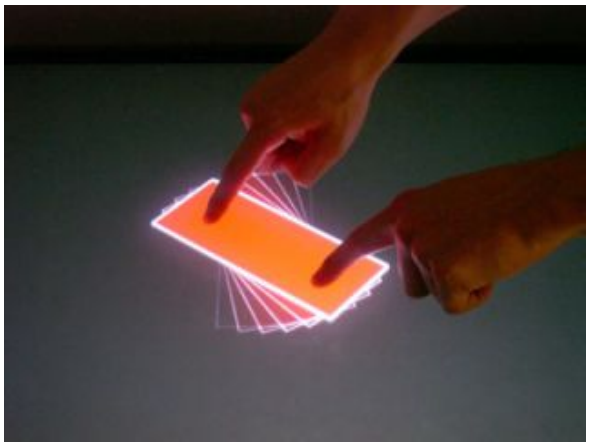
Zunächst muss eine Repräsentation für den Spielzustand definiert werden, die bei Java Settlers eine abstrakte Repräsentation des Spielbretts und der Spielerrohstoffe ist. Dieser Zustand entspricht dem Modell des modellbasierten Agenten und dient als Basis für die Berechnungen der Suche. Daneben ist ein Zugegenerator zu schreiben, der von einem Zustand aus eine Menge an gültigen Zügen generieren kann. Diese legalen Züge entsprechen den Zustandsübergängen zu neuen Spielzuständen. Erreicht die Suche einen Blattknoten, so wird eine Bewertung der aktuellen Stellung durchgeführt. Dies wird realisiert, indem eine Bewertungsfunktion spezifiziert wird, die in der Regel aus einer gewichteten Ansammlung vieler kleiner Bewertungsfunktionen besteht. Die errechneten Bewertungen werden von der Suche zum Vergleich verschiedener Stellungen herangezogen. Letztendlich muss nur noch eine GUI für das Spiel geschrieben werden, die den Spielverlauf graphisch visualisieren kann.
Eine ganze Menge typischer Spiele, wie z.B. Tic-Tac-Toe, Schach, Dame, Vier gewinnt, Asteroids, usw. wurde bereits mit dem jGameAI Framework erstellt [49]. Java Settlers erweitert diese Liste um ein weiteres Spiel, das sich mit komplexen Strategien, wahrscheinlichkeitsbasierten Spielverlauf und einem dynamischen Spielaufbau von den anderen unterscheidet.

## 2.5 Multi-Touch Techniken

Multi-Touch-Systeme sind Interaktionssysteme, die mehrere Druckpunkte, sogenannte Touches, von einem oder mehreren Benutzern als Eingabe entgegennehmen. Manche Systeme ermöglichen es sogar, die verschiedenen Benutzer zu unterscheiden und ermöglichen so neue Interaktionsmöglichkeiten. Im Gegensatz zu herkömmliche Eingabegeräte wie Maus und Tastatur, Stylus oder Touchscreens, die stets nur einen Druckpunkt erkennen können, bietet Multi-Touch wesentlich intuitivere Mensch-Maschine-Interaktion. Die meisten Systeme basieren auf einer großen Bildschirmoberfläche, die mit Fingern und Gesten gesteuert wird [50]. Dies ermöglicht auch virtuelle Bildschirmobjekte zu manipulieren, verschieben, skalieren oder zu rotieren. Manche Systeme können reale Bedienobjekte, sogenannte „fiducials“ erkennen und durch diese gesteuert werden [51]. So kann das System Handys erkennen oder mit einfachen Spielsteinen geleitet werden. Durch die Konstruktion einfach und intuitiv zu bedienender Systeme soll eine breite Benutzerschicht der Umgang mit dem System ohne lange Einarbeitungszeit ermöglicht werden. Natürlich werden auch weiterhin Systeme koexistieren, die aufgrund ihrer Komplexität nicht komplett durch Multi-Touch-Steuerung ersetzt werden können.
Dennoch lässt sich die Berechtigung von Multi-Touch im Vergleich zu Maus und Tastatur folgendermaßen illustrieren: 1984 gab es den ersten durchbrechenden Markterfolg für die Maus, als Apple den Macintosh eingeführt hatte [7]. Vorher war die Industrie auf spezielle Tastaturen angewiesen, die für das jeweilige Anwendungsprogramm verschiedene Tasten bereit stellten. Mit der Maus kam die Ära der graphischen Bediensysteme, wo die Tasten bildlich gesehen in die Software verfrachtet wurden und die Maus den Finger simulierte. Seitdem hat sich am

Tandem Maus und Tastatur kaum etwas verändert, während der Rechner selbst und die Netzinfrastrukturen stetig verbessert wurden. In der Tat wurde in der Computer-Computer-Interaktion ständig weiterentwickelt; die Mensch-Computer-Interaktion aber vernachlässigt. Heutzutage arbeiten wir mit Prozessoren im GHz-Bereich und surfen im Internet mit MBit/s, aber kommunizieren immer noch über ein dünnes Kabel von der Maus mit dem Rechner. Es hat sich in Experimenten gezeigt, dass der Mensch bevorzugt über denselben Kanal rückkommuniziert über den er angesprochen wurde [6]. Wenn der Benutzer über Signale auf dem Bildschirm informiert wird, ist es intuitiv mit den Fingern auf dem Bildschirm Rückmeldungen zu geben. Für den Menschen ist der Sehsinn der wichtigste Sinn, während die Hände und Finger die wichtigsten Aktoren sind [6].

(a) Erkennung mehrerer Berührungspunkte [51]



(b) Rotation von Bildschirmobjekten [51]

Abbildung 11: Multi-Touch-Systeme

Die Technik, die verwendet wird, um Systeme Multi-Touch-fähig zu gestalten, sind vielfältig. Grundsätzlich müssen sie mehrere Touches unterscheiden können und sollten dabei wenige falsch positive und falsch negative Touches erkennen. Zudem ist der Wartungsaufwand der Systeme zu berücksichtigen, ob sie rekalibriert oder gar Teile ausgetauscht werden müssen. Diesbezüglich sind also Qualität, Geschwindigkeit und Stabilität der Erkennung wesentlich. Wünschenswert, aber nicht notwendig, sind Fähigkeiten für Gesten, Drag'n'Drop-Aktionen, der Einsatz von fiducials oder die Unterscheidung der Touches zwischen mehreren Benutzern.

### 2.5.1 Optische Systeme

Multi-Touch-Systeme basierend auf optischer Sensorik ist die bevorzugte Art großflächige interaktive Arbeitsoberfläche zu konstruieren. Mit Hilfe von einer oder mehrerer Kameras wird kontinuierlich ein Bild von der Oberfläche aufgenommen, welches dann in Echtzeit verarbeitet wird, um die einzelnen Druckpunkte zu identifizieren.

### Lichtzeiger

Eine einfache Möglichkeit auch ohne Finger das System zu steuern könnte über die Verwendung von Lichtzeiger, wie herkömmlichen Laserpointer, realisiert werden. Dabei strahlt ein Laserpointer in einer bestimmten Farbe (rot, grün, blau, o.a.) auf die Oberfläche, während eine Kamera im Kamerabild nach dieser Farbmarkierung sucht. Dies ist eine sehr kostengünstige Variante und ermöglicht die Steuerung des Systems bis zu einer gewissen Entfernung. Problematisch ist jedoch die Erkennung, wenn die Farbe des Laserpointer im Bild selbst auftritt und nicht zu unterscheiden ist. Dies kann gelöst werden, indem auf menschlich nicht sichtbare Farben gewechselt wird: Ultraviolett oder besser Infrarot im Wellenbereich von 780 nm bis 1400 nm, welches weniger schädlich für den Menschen ist.

### Frustrated Total Internal Reflection (FTIR)

Die New York University hat 2005 unter Jeff Han das Prinzip der Lichtbrechung genutzt, um eine kostengünstige Multi-Touch-Erkennung zu installieren [52, 50]. Das unter *frustrated total internal reflection (FTIR)* bekannte Prinzip wird bereits häufig in der Biometrie zur Fingerabdruckerkennung verwendet: Lichtstrahlen werden gebrochen, wenn sie auf eine Grenze zwischen zwei Medien treffen, wobei das Medium, in das eingetreten wird, einen geringeren Brechungsindex aufweist als das Medium, aus dem es kommt. Als Beispiel dient die Lichtbrechung an der Wasseroberfläche. Trifft der Lichtstrahl aber unter einem sogenannten kritischen Winkel, so wird er komplett reflektiert und bleibt im ursprünglichem Medium (*(total internal reflection (TIR)*).

Han wählte als Medien Plexiglas und herkömmliche Luft. Wird nun seitlich in das Glas mit Infrarot-LEDs gestrahlt, dann bleibt das Licht im Glas „gefangen“. Berührt ein Finger die Glasoberfläche, so besteht zwischen Finger und Glas ein anderes Brechungsverhältnis, was dazu führt, dass die Lichtstrahlen gebrochen werden und zur Rückseite der Glasfläche austreten. An der Rückseite angebrachte Infrarotempfindliche Kameras erkennen die Finger nun als helle Flecken, sogenannte *blobs*. Das Computerbild kann derweil von einem Projektor an die Oberfläche projiziert werden. In Abbildung 12 ist der Aufbau nochmal als Schema skizziert.

Insgesamt jedoch bietet FTIR eine exzellente Erkennung von Touches, weil durch die kurze Entfernung von der Lichtquelle zu den Fingern wenig Helligkeit verloren geht. Die blobs erscheinen daraufhin klar und hell. Damit existiert eine sehr kostengünstige Variante um Multi-Touch zu verwirklichen, auch wenn die Erstellung der seitlichen LED-Leisten ein wenig Bastelarbeit verlangt. Diese Technik ist zudem beliebig skalierbar und somit auch für sehr große Oberflächen nutzbar, jedoch durch den sperrigen Aufbau aufgrund der Entfernung zur Kamera nicht für kleine mobile Geräte möglich. Ein weiterer Nachteil an dem System ist, dass Drag’n’Drop-Aktionen nicht klar zu erkennen sind. Durch die Bewegung der Finger auf der Oberfläche lässt

der Druck nach und es bilden sich Luftpolster zwischen Finger und Glas, woraufhin die blobs nicht mehr so hell erscheinen. Experimente mit nachgiebigen Oberflächen, z.B. einem dünnen Aufstrich aus Silikon auf die Glasoberfläche, zeigen leichte Verbesserungen der Erkennung (siehe Abbildung 42 in Anhang A.1).

### Diffused Illumination (DI)

Die Platzierung der LEDs an der Glasseite kann für große Flächen aufwendig sein. Eine alternative Variante wäre mittels Infrarot-Scheinwerfern die gesamte Glasoberfläche von der Rückseite aus gleichmäßig anzustrahlen. Die Lichtstrahlen durchdringen die Glasfläche und strahlen ins Unendliche. Ein Finger auf der Glasfläche würde nun direkt angestrahlt werden und das Licht hell zurück reflektieren. Die so erstellten blobs können erneut für die weitere Verarbeitung genommen werden. Dieser Aufbau ist nochmal als Schema in Abbildung 14 skizziert.

Ein DI-System ist einfacher und billiger zu realisieren, denn es benötigt keine speziellen Materialien für die Glasoberfläche. Die blobs sind meist nicht so scharf wie bei FTIR, aber deutlich genug und bei Drag'n'Drop-Aktionen sehr stabil. Zudem kann ein DI-Aufbau viel besser fiducials erkennen. Wichtig ist es, die Scheinwerfer so auszurichten, dass die Oberfläche möglichst gleichmäßig bestrahlt wird. Wie bei den anderen Infrarotbasierten Systemen auch, muss darauf geachtet werden, dass das umgebende Sonnenlicht, welches auch Infrarot mit sich bringt, keine allzu große Einwirkungen auf das System haben sollte.

Ein Beispiel für DI in der Praxis kommt demnächst von Microsoft namens „Surface“ auf dem Markt [51]. Es soll neben Fingern auch fiducials wie Handys gut erkennen, wenn sie auf die Glasoberfläche gelegt werden. Die angehängte Abbildung 43 zeigt einen wahrscheinlichen inneren Aufbau von „Surface“.

Das DI-Verfahren verhindert nicht den Einsatz von FTIR, so dass eine Kombination durchaus vorteilhaft sein kann. Dabei muss beachtet werden, die Glasoberfläche nicht zu überleuchten, damit die Finger noch gut von der Umgebung zu unterscheiden sind.

### Angulation

Angulation ist eine alternative Variante, bei der mehrere kleine Kameras in die Ecken der Projektionsfläche eingebaut werden [55]. Diese schauen parallel zur Oberfläche und filmen direkt die Hände der Benutzer ab. Aus den Kamerabildern werden nun die Handpositionen trianguliert und verarbeitet. Da bei vielen Händen Verdeckungen stattfinden, skaliert dieses System zwar in Bezug auf die Größe, aber nicht in Bezug auf die Benutzerzahl. Im Gegensatz zu den anderen optischen Varianten hat sich diese Technik nicht sonderlich durchgesetzt.

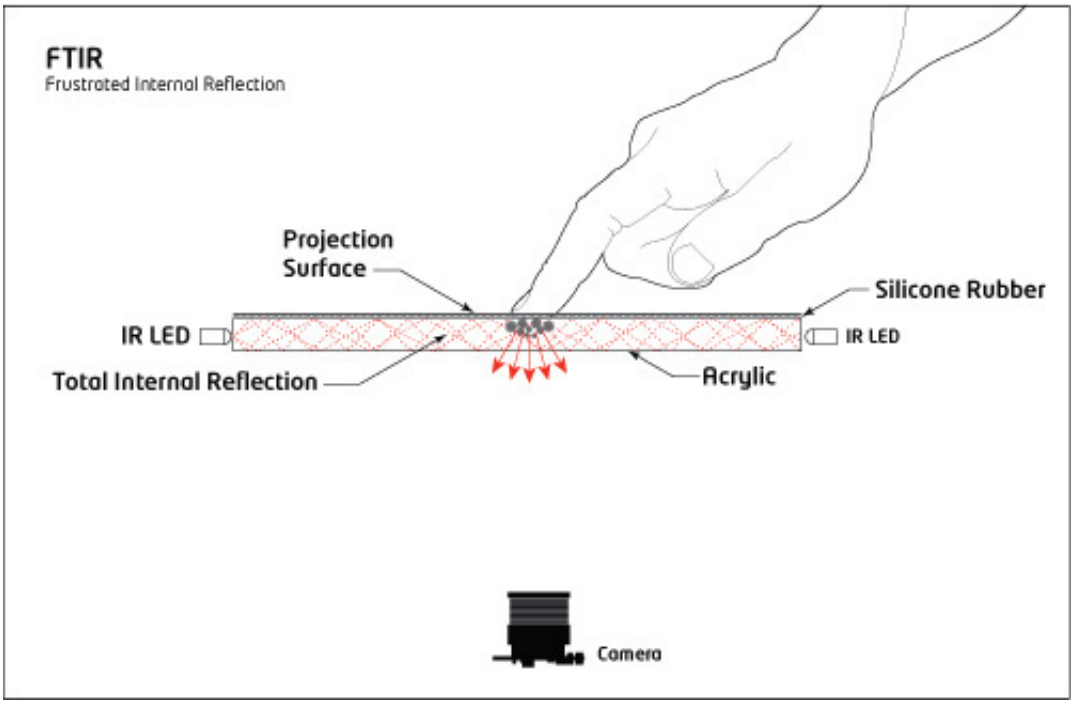



Abbildung 12: Aufbau für Frustrated Total Internal Reflection, Abbildung aus [53]

(a) Ein Multi-Touch-Wandsystem von Jeff Han [54]

(b) Kamerabild der Finger-Blobs einer Hand [54]

Abbildung 13: Multi-Touch-Systeme mit FTIR

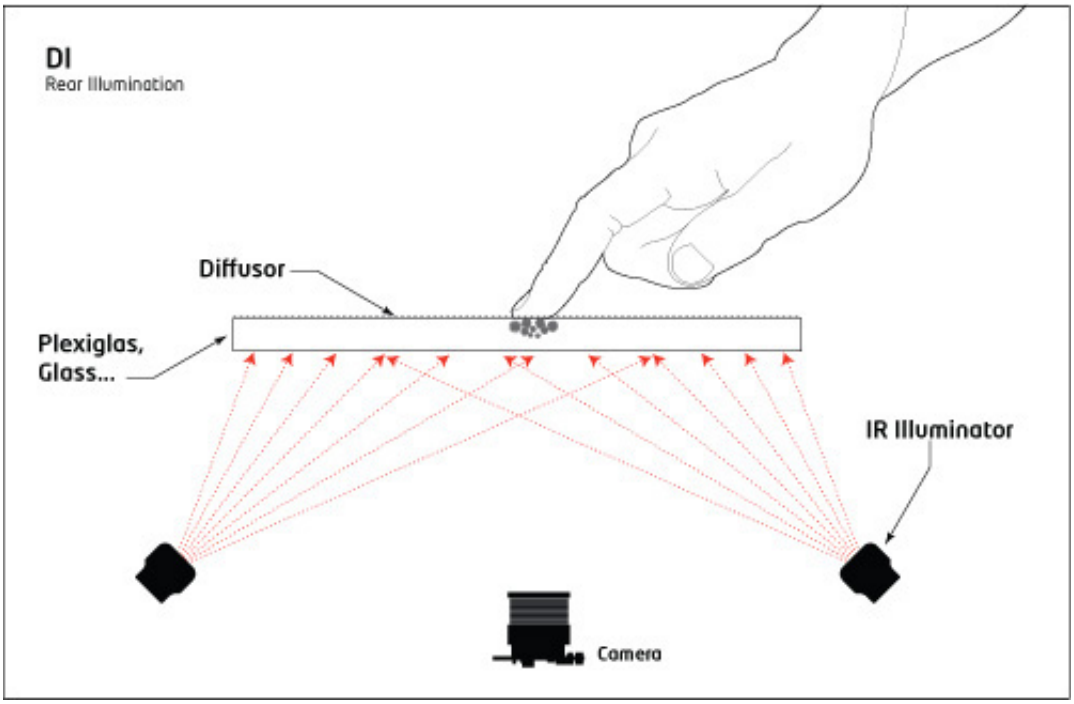



Abbildung 14: Aufbau für Diffused Illumination, Abbildung aus [53]

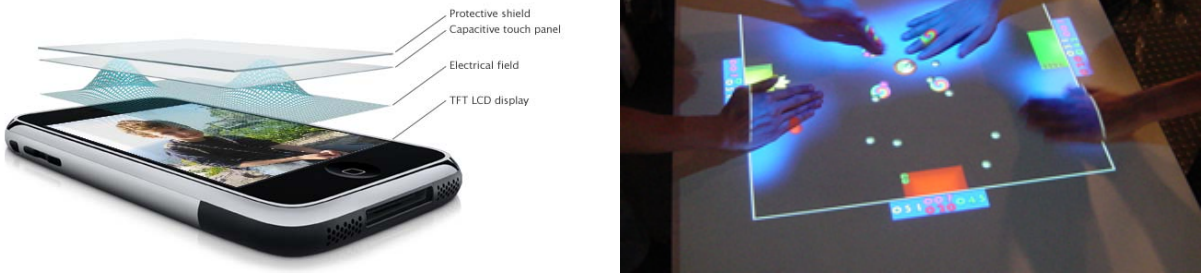
(a) Microsoft Surface [51]

(b) Erkennung von Weingläser als fiducials [51]

Abbildung 15: Multi-Touch-Systeme mit DI

### 2.5.2 Kapazitive Systeme

Mit kapazitiver Sensorik können auch kleine bis mittelgroße Multi-Touch-Systeme konstruiert werden, die sogar fähig sind Berührungen von mehreren Benutzern zu unterscheiden. Die Idee basiert auf der kapazitiven Kopplung zwischen einer Senderantenne und einer Empfängerantenne. Je nach Intensität der Kopplung kann eine Spannung gemessen werden, welche als Touch klassifiziert werden kann. In der Regel werden ganze Antennen-Grids eingesetzt, um eine dichte Abdeckung zu erreichen. Aufgrund der Kosten rentiert sich dies jedoch nicht für großflächige Installationen.
Das 2007 erschienene iPhone von Apple [56] bedient sich eines solchen kapazitiven Grids auf einem Flüssigkristall-Feld, um über elektrische Felder die Fingerpositionen zu erkennen. Mit dem iPhone ist auch Multi-Touch zum ersten Male breit der Öffentlichkeit verfügbar gemacht worden, obgleich das iPhone in der Regel nur bis zu zwei Fingerberührungen verarbeiten kann. Weitere Beispiele für kapazitive Systeme sind Sony SmartSkin und DiamondTouch [57, 58].




(a) Apple iPhone [59] (b) Sony SmartSkin [57]

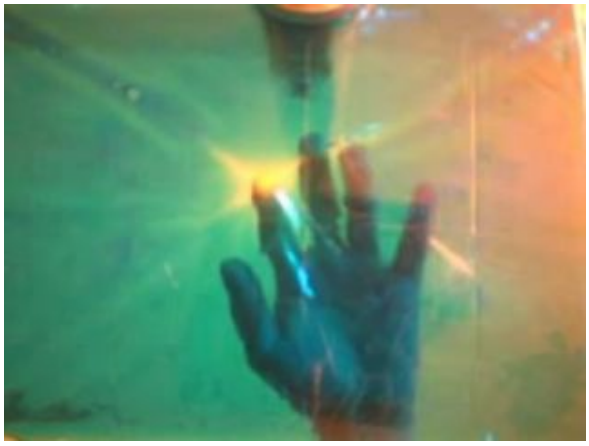
Abbildung 16: Multi-Touch-Systeme mit kapazitiver Sensorik

### 2.5.3 Weitere Ansätze für Multi-Touch

Im Gegensatz zu optischen oder kapazitiven Verfahren setzt werden bei akustischen Systemen Mikrofone eingesetzt, die Berührungsgeräusche aufzeichnen [60]. An den Seiten oder Ecken der Oberfläche angebrachte Mikrofone können durch die Laufzeitunterschiede die Position der Berührung triangulieren. Durch die Überbestimmung des Signals mit Hilfe mehrerer Mikrofone kann das Rauschen kompensiert werden. Die Interaktion mit dem System geschieht aber weniger mit Berührungen, sondern mit leichtem Klopfen. Dadurch fallen aber Drag'n'Drop-Funktionalitäten und eine Menge an intuitiven Gesten weg.
Es gibt noch viele weitere Verfahren, die sich aber kaum durchgesetzt haben. So gibt es die

Möglichkeit, eine in dünne Folie eingeschweißte Farbflüssigkeit zu benutzen, die an den Berührungspunkten zur Seite weicht. Eine Kamera kann anhand der hellen „Löcher“ im Farbbild die Positionen der Fingerberührungen identifizieren.

(a) Reactable zur Erstellung elektronischer Musik mit Musik-Steinen als fiducials [61]



(b) Multi-Touch über in Folie eingeschweißte Farbflüssigkeit [62]

Abbildung 17: weitere Multi-Touch-Systeme

## 2.6 Multi-Touch Applikationen

Basierend auf der Technik zur Erkennung der Fingerberührungspunkte müssen durch verschiedene Bildverarbeitungsschritte zweidimensionale Koordinaten gewonnen werden, die die Position und Stärke der Berührungen beinhalten. Diese Koordinaten können dann vom Rechner zur Steuerung von Applikationen entgegen genommen werden. Zwar können die Touches auch in Mausbewegungen und Mausklicks übersetzt werden, um somit alle bisher gängigen Applikationen zu steuern, jedoch verliert der Anwender die Möglichkeit der Steuerung durch mehrere Punkte gleichzeitig. Ein solcher Multi-Touch-zu Maus-Treiber ist lediglich eine Bequemlichkeit, um bisherige Applikationen zu steuern. Um aber die vollen Möglichkeiten der neuen Technologie auszuspielen, bedarf es neuartig entworfene Anwendungsprogramme für diesen Zweck. Letztendlich muss für das neuartige Bedienkonzept eine angepasste Usability entwickelt werden, die sowohl einfach verständlich ist als auch die vollen Möglichkeiten der Interaktion ausschöpft. Komplizierte Befehle zum Rotieren oder Skalieren von Objekten müssen nicht mehr in hierarchische Menüs eingebaut werden, sondern können direkt genutzt werden. Eine neue Welt an Interaktionsmöglichkeit eröffnet sich für den Benutzer zum Erforschen und für den Entwickler diese auszureizen. Im Folgenden sollen einige unterschiedliche Arten von Applikationen für Multi-Touch vorgestellt werden.

### 2.6.1 Transformation rigider Körper

Eine Gruppe von Applikationen widmet sich der Transformation von *rigiden Körpern* (engl.: rigid body), wie z.B. Bilder, Videos, Dokumente oder anderen abstrakten starren Objekten (siehe Abbildung 18). Diese haben alle die Eigenschaft sich nicht zu verformen und können mit den Fingern verschoben, skaliert, rotiert und auch umgedreht werden. Durch Multi-Touch können gleichzeitig mehrere Objekte bearbeitet werden und auch von mehreren Personen bearbeitet werden. Es ist leicht vorstellbar, dass mehrere Personen an einem Multi-Touch-System sitzen, um damit ein technisches Modell gemeinsam zu konzipieren oder strategisch Ereignisse auf einer Landkarte zu planen.

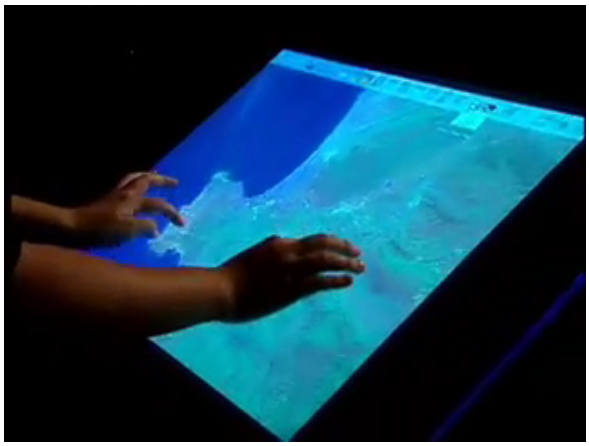
### 2.6.2 Transformation nicht rigider Körper

Während die Transformation rigider Körper auch mit herkömmlichen Eingabemöglichkeiten zu lösen wäre, sind Applikationen, die *nicht starre Objekte* transformieren, an sich nur mit Multi-Touch möglich (siehe Abbildung 19). Gemeint sind hier verformbare Objekte wie simulierte Flüssigkeiten oder aufgeweichte Modelliermassen, die durch Fingerberührungen verteilt, gepresst, verformt oder gefestigt werden können. Auch hier steht der Einsatz einer Modellierungssoftware für ein gesamtes Team im Raum, bei der jede Person seine Ideen einbringen kann.

### 2.6.3 Schreiben und Malen

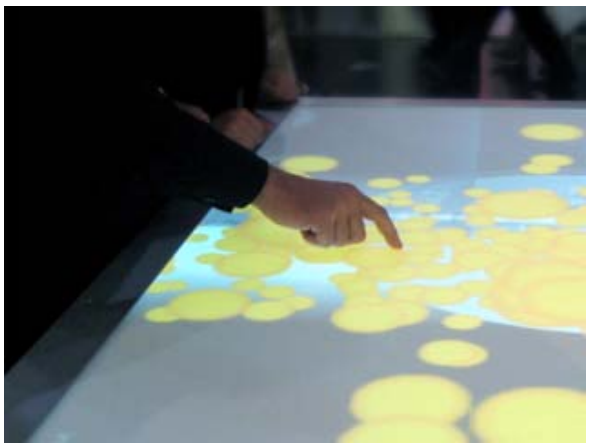
Andere Applikationen wären das direkte Schreiben oder Malen auf die Oberfläche wie bei gängigen Touchscreens (siehe Abbildung 20). Eine Optische Zeichenerkennung könnte zusätzlich Rechtschreibfehler korrigieren. Der Vorteil bei Multi-Touch liegt in der intuitiven Eingabe direkt mit mehreren Fingern, die ebenfalls genutzt werden könnte, um etwas gemaltes wieder zu verwischen oder zu vermischen. Die FTIR-Technik hat hier den Vorteil einer genaueren Erkennung. Andererseits ermöglicht ein DI-Ansatz eine bessere Möglichkeit für den Einsatz von Stiften oder einen Pinseln. Bei ausreichender Auflösung der Kamera könnten nur die einzelnen Pinselhaare zum Zeichnen beitragen und die Druckstärke würde der Dicke entsprechen, mit der gemalt wird.

(a) Bewegen, Rotieren und Skalieren von Fotos [51]



(b) Freies Bewegen einer 3D Welt [54]

Abbildung 18: Transformation rigider Körper



(a) Simulation einer Lavalampe [54]

(b) Simulation von Wasser [51]

Abbildung 19: Transformation nicht rigider Körper


(a) Schreiben mit mehreren Fingern [51]

(b) Malen mit mehreren Fingern [51]

Abbildung 20: Schreiben und Malen

# 3 Hardware

Im Folgenden soll ein möglicher Tischaufbau für Java Settlers und weitere Multi-Touch-Projekte konzipiert werden. Für zukünftige Arbeiten sollen auch weitere Überlegungen für eine Erweiterung des Systems erklärt und eine Vision für zukünftige Systeme vorgestellt werden.

## 3.1 Konzeptueller Tischaufbau

Die Idee ist es, einen Tisch so zu entwerfen, auf dessen Oberfläche das Spielbrett zu sehen ist und alle möglichen Aktionen über Fingerbewegungen durchzuführen sind. Ein Rechner im Inneren des Tisches ist beauftragt die Fingererkennung als auch die Spielberechnungen durchzuführen. Diese beiden Aufgaben können auch getrennt auf zwei separaten Rechner ausgeführt werden, was aber mehr Platz im Tischinneren erforderlich macht. Das darzustellende Bild soll über einen Projektor an die Oberfläche geworfen werden, während die Erkennung der Touches durch eines der erwähnten optischen Techniken geschehen sollte. Die teilnehmenden Spieler können wie bei üblichen Brettspielen an den Seiten einen Platz einnehmen wie die Konzeptzeichnung in Abbildung 21 aufzeigt.

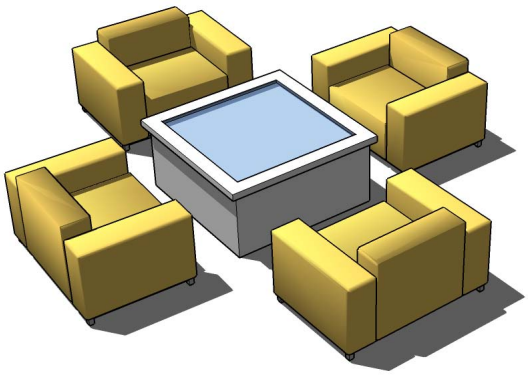


Abbildung 21: Konzeptzeichnung für den Einsatz eines Multi-Touch -Tisches für Java Settlers

Der erwähnte DI-Ansatz ist ideal für einen Multi-Touch-Tisch in der Höhe eines üblichen Arbeitstisches, weil er einfach zu realisieren ist und eine relativ stabile Fingererkennung ermöglicht. Der sperrige Aufbau, den der DI-Ansatz mit sich bringt, kann geschickt im Inneren des Tisches versteckt werden. Um die Finger auf der Oberfläche sichtbar zu machen, wird mit Hilfe von Infrarotstrahlern die gesamte Oberfläche gleichmäßig ausgehellt. Eine Kamera filmt derweil eine Glasoberfläche ab und leitet das Bild an den Hauptrechner weiter. Schafft es eine Kamera von der Optik her nicht die gesamte Oberfläche abzufilmen, so müssen mehrere Kameras eingesetzt werden und das Gesamtbild zusammengesetzt werden. Ein Projektor wirft das Bild direkt oder, wenn die Projektionsfläche nicht ausreicht, indirekt über einen Spiegel an die

Oberfläche zurück. Alternativ kann ein Short-Throw Projektor eingesetzt werden. Die Abbildung 22 stellt einen möglichen Aufbau in einer Schemazeichnung dar.

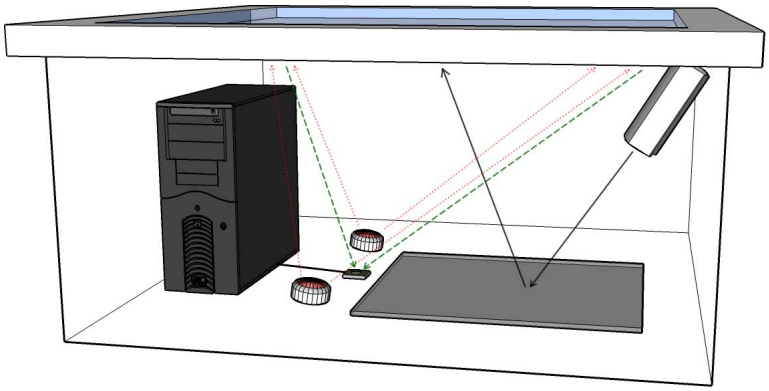


Abbildung 22: Schemazeichnung eines Multi-Touch-Tisches

Ein weiterer Entwurf in Form eines Rednerpultes berücksichtigt den Einsatz eines Multi-Touch-Systems für Präsentationszwecke (siehe Abbildung 23). Der innere Aufbau entspricht in etwa dem vorherigen, wobei die Komponenten sich den neuen Maßen anpassen müssten. Besonders in der Lehre ist der Einsatz eines solchen mobilen Pultes äußerst sinnvoll. Es kann zu diversen Lokationen transportiert und an einen externen Projektor zwecks großflächiger Projektion angeschlossen werden. Eine entsprechende Software vorausgesetzt kann das direkte Schreiben auf der Oberfläche die übliche Tafel ersetzen. Hier würde der zusätzliche Einsatz von FTIR sinnvoll erscheinen, um eine präzisere Erkennung zu ermöglichen.

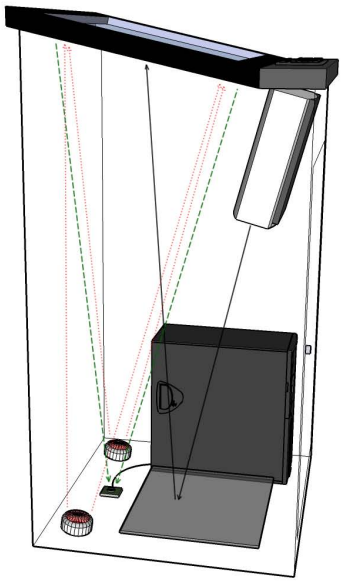


Abbildung 23: Schemazeichnung eines Multi-Touch-Rednerpultes

Unabhängig von der Form des Systems müssen einige generelle Überlegungen bezüglich der Wärmebildung und Lüftung getroffen werden. Der Rechner und vor allem der Projektor benötigen eine gute Belüftung um langfristig eingesetzt werden zu können. Dafür müssen Lüftungsöffnungen und eventuell zusätzliche Lüfter platziert werden. Dies jedoch hat wiederum den Nachteil, dass die Lautstärke des Systems nach oben steigt und als störend empfunden werden kann. Noch gravierender ist der Umstand, dass Umgebungslicht in das geschlossene System fallen kann, was die Erkennungsleistung beeinträchtigt. Eine eindeutige generelle Lösung gibt es für dieses Problem leider nicht. Je nach eingesetzten Komponenten und Aufbau müssen mehr oder weniger Freiraum für Lüftung gewährt werden. Diese sollten so platziert oder verdeckt werden, dass die Oberfläche und die Kameralinse vor Umgebungslicht geschützt sind.

## 3.2 Weitere Überlegungen und Visionen

Die Glasscheibe als Tischoberfläche stellt ein Display da, welches die notwendigen Informationen des Spiels für alle Spieler gleichzeitig anzeigt. Nun gibt es auch geheime Informationen im Spiel (z.B. Entwicklungskarten, Rohstoffe, usw.), die nicht jedem Spieler zugänglich sein sollen, sondern nur einem speziellen Spieler. Dies ist ein Problem, welches aus Kostengründen softwaretechnisch gelöst wurde, wie es im nächsten Kapitel noch näher erläutert wird. Konzeptuell schöner, aber auch teurer, wäre es, dieses Problem hardwareseitig zu lösen. Die Vision hierbei ist es, die Anzeige der spielerbezogenen Informationen nicht für alle Spieler auf dem Display zu platzieren, sondern jedem Spieler ein portables Display zur Verfügung zu stellen. Dies könnte ein mobiles Smartphone oder eine Handheld-Konsole sein, welches in der Lage ist mit dem Tisch zu kommunizieren, um aktuelle Informationen abzurufen, anzuzeigen und Befehle an den Tisch zu senden.
Eine andere Variante wäre es in Richtung Augmented oder Virtual Reality zu gehen und jedem Spieler eine virtuelle Shutterbrille oder ein Head-Mounted Display zur Verfügung zu stellen [63, 64]. Diese können neben dem Tischdisplay weitere Informationen und Animationen anzeigen. Mit Hilfe eines Datenhandschuhes könnten Interaktionen zurück an den Tisch gesendet werden, um eine Multi-Touch-Aktion anzustoßen. Diese Variante könnte sogar soweit gehen, dass der gesamte Tischaufbau durch ein verteiltes Netz an leistungsfähigen Brillen ersetzt wird, um damit das Spielgeschehen zu koordinieren.
Der Tisch an sich soll ein Prototyp darstellen für neuartige Computersysteme und -formen, die von Benutzern gesteuert werden können. Die Vision ist es, den Tisch weiterzuentwickeln und auch andere Systeme zu entwerfen für den intuitiven Einsatz im Alltag. Obwohl sich der Tisch sehr eignet, um einige Runden Java Settlers zu spielen, gibt es noch wenige Punkte zu bemängeln. So ist der platzeinehmende Aufbau des Systems hinderlich für qualitativ hochwertige und kleine Systeme. Zwar lässt sich der Großteil im Tischinneren verbergen, aber je flacher und kompakter der Aufbau gestaltet wird, umso robuster und attraktiver wird das System. Die Ursa-

che dafür ist sicherlich der verwendete DI-Ansatz. Mit einem anderen technischen Ansatz oder einer teuren kompakten Verkleinerung in ein Display könnte jemand dies in Zukunft umgehen, was aber noch einige Jahre Vorlauf in Forschung und Entwicklung benötigt.
Sollte dies eines Tages verwirklicht werden, steht Multi-Touch nichts mehr im Wege im täglichen Alltag praktische Anwendung zu finden. Es bieten sich gleich unzählig viele Visionen vom großflächigen Einsatz an öffentlichen Plätzen [65], in Messen, Ausstellungen, Hotels, Bars, Spielhallen oder an Litfaßsäulen bis hin zu kleinen Displays an Fahrkartenautomaten und mobilen Geräten. Höchstwahrscheinlich bietet sich auch der Einsatz im eigenen Haus an: Ein Multi-Touch-Display am Kühlschrank, im Flur zur Wohnungssteuerung oder als intelligenter Spiegel lassen die Idee vom intelligenten Haus wahr werden [66].

# 4 Software

In den nächsten beiden Kapiteln soll ein Überblick über die Software-Architektur von Java Settlers gegeben werden (Abbildung 24). Diese Architektur kann grob in drei große Module eingeteilt werden: die Benutzeroberfläche, die Schnittstelle zu Multi-Touch und die Schnittstelle zu jGameAI.

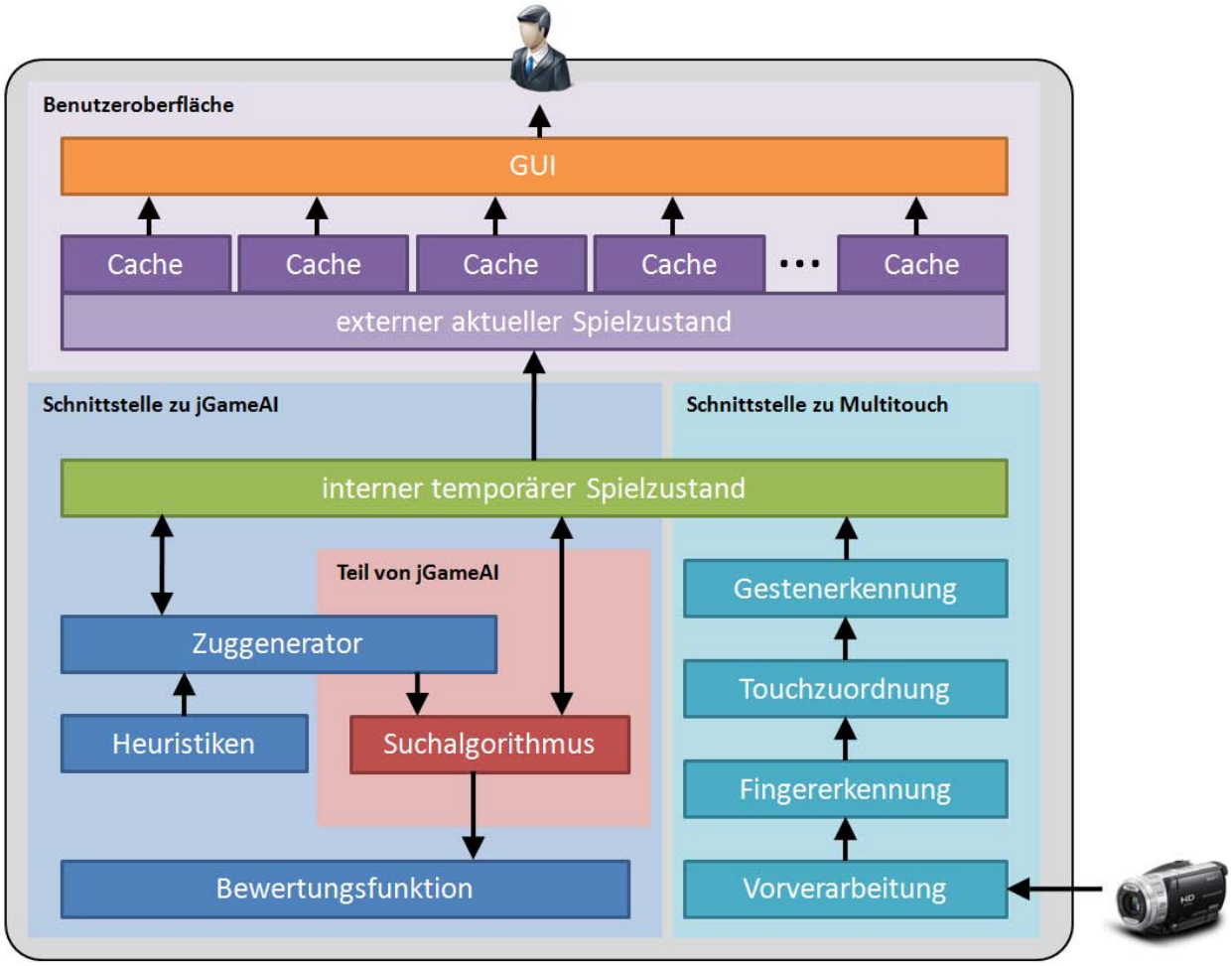



Abbildung 24: Architektur von Java Settlers. Eine grafische Repräsentation der Architektur von Java Settlers

Java Settlers verwaltet den aktuellen Spielzustand zweifach: Einmal als internen temporären Zustand für Berechnungen in die Zukunft und ein zweites Mal als externen aktuellen Zustand für die Darstellung durch die Benutzeroberfläche. Wird eine Aktion vom Benutzer oder vom Computerspieler ausgeführt, so werden beide Zustände synchronisiert. Der externe Zustand holt sich benötigte Informationen vom internen Zustand und speichert sie in Caches ab zur weiteren Darstellung. Die Schnittstelle zu jGameAI verbindet Java Settlers mit den Funktionalitäten vom jGameAI-Framework. Ein allgemeiner Suchalgorithmus wird vom Framework bereit gestellt, während ein Zuggenerator, Heuristiken und Bewertungsfunktionen Einfluss auf die Suche neh-

men können. Für die Steuerung wird eine Schnittstelle zu Multi-Touch bereit gestellt, die die Kamerabilder in einer Verarbeitungskette zu Gesten verarbeiten kann.

## 4.1 Benutzeroberfläche

„Die Siedler von Catan“ kann wahlweise mit 3 bis 4 Spielern gespielt werden, mit zusätzlichen Erweiterungen sind auch bis zu 6 Spieler möglich. Für die PC-Umsetzung Java Settlers wurde die Spieleranzahl auf maximal 4 Spieler festgesetzt.

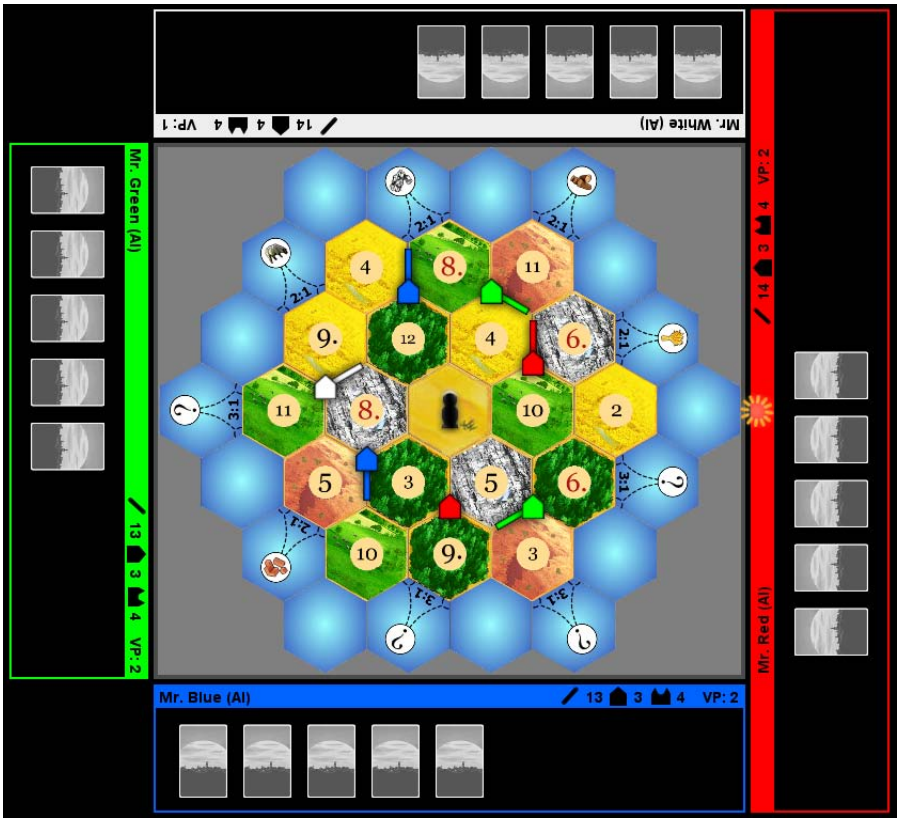



Abbildung 25: Screenshot von Java Settlers. Weitere Screenshots sind im Anhang A.2 zu finden.

Auf einem quadratischen Bildschirmausschnitt wird das virtuelle Spielbrett angezeigt und zu jeder Seite jeweils ein Bedienungsfeld für jeden Spieler. Dabei sind die Bedienelemente in Orientierung zu jedem Spieler entsprechend gedreht (siehe Abbildung 25). Das Spielbrett zeigt die aktuelle Brettstellung an sowie Animationen und Menüs, die zum Spielfluss dazu gehören. Mit Hilfe einer Geste von zwei Fingern kann der Benutzer in das Spielbrett zoomen: Zieht er die Finger auseinander, so zoomt er hinein, während er wieder heraus zoomt, wenn er beide Finger zueinander bewegt. Diese Geste ist analog zu vorhandenen Zoom-Gesten im iPhone [56]. Im gezoomten Zustand kann der Benutzer mit einem Finger den Spielbrettausschnitt verschieben. Somit ist es jederzeit möglich, eine Übersicht über das gesamte Spielgeschehen zu erlangen, als auch einen detaillierten Ausschnitt eines Brettbereichs. Zu dem Spielbrett werden die einzelnen

Spielfiguren (Straßen, Siedlungen, Städte) an die jeweiligen Positionen platziert. Halbdurchlässige Spielfiguren markieren für einen Spieler potentielle Stellen um neue Straßen, Siedlungen oder Städte zu platzieren.
Zu gewissen Anlässen im Spiel wird eine Animation angezeigt, so wird z.B. der Würfelwurf animiert dargestellt (siehe Abbildung 26). Diese Animationen verschönern nicht nur das Spielerlebnis, sondern bieten auch Möglichkeiten zur Spielgestaltung virtuelle Effekte anzuzeigen, die bei einem Brettspiel nicht möglich sind. Weitere Animationen zeigen die Verteilung der Rohstoffe nach einem Würfelwurf oder das Ausspielen einer Entwicklungskarte (siehe Abbildungen 27, 28).
Während das Platzieren von Spielfiguren direkt auf dem Spielbrett möglich ist, kann sich ein Spieler verschiedene Menüs anzeigen lassen, um Aktionen auszuführen. Ist ein Spieler z.B. an der Reihe, kann er Menüoptionen aufrufen, um gewisse Handel abzuschließen (siehe Abbildung 29). Jede Schaltfläche kann einfach mit dem Finger angetippt werden, um sie auszuwählen. Zu jedem Spieler gibt es ein Bedienfeld, welches die aktuellen Ressourcen und Entwicklungskarten anzeigt. Zudem gibt es in einer Leiste Informationen über noch verfügbare unbenutzte Straßen, Siedlungen und Städte sowie die Anzahl der Siegpunkte. Auch die Sonderpunkte wie die längste Handelsstraße oder die größte Rittermacht werden hier angezeigt.
Weitere Screenshots sind im Anhang A.2 zu finden: Sie zeigen z.B. die Aktionen des Räubers oder Einstellungen im spielübergreifenden Hauptmenü.
Nun gibt es nur eine Anzeige für alle vier Spieler und jede Aktion eines Spielers ist für die anderen Spieler ebenfalls sichtbar. Bei den meisten Aktionen hat dies keine Auswirkungen, bei geheimen Informationen jedoch gibt es ein Konflikt mit der Spielprinzip. So dürfen andere Mitspieler nicht mitbekommen, welche Entwicklungskarten ein Spieler besitzt oder wie viele Rohstoffe er in der Hand hält. Da ein Benutzer die virtuellen Karten nicht mehr verdeckt zu sich nehmen kann, müssen andere Interaktionen dafür sorgen, dass diese Informationen nur einem Spieler zugänglich sind. Eine Geste in Java Settlers ist, dass der Benutzer mit nach vornegleiten eines Fingers die betroffenen Karten kurz aufdeckt und beim zurückgleiten diese wieder verdeckt. Dies ist eine intuitive Bewegung, die dem Aufrollen der Karte entspricht. Der Spieler selber muss natürlich dabei beachten, die sichtbaren Informationen gegenüber den anderen Mitspielern abzudecken, indem er z.B. eine Hand davor hält. Alternativ kann über das Hauptmenü auch eingestellt werden, dass die Rohstoffe jederzeit sichtbar bleiben.
Die GUI wird komplett in OpenGL mit Java gerendert, sodass die meiste Arbeit von einem Grafikprozessor übernommen wird, der die CPU entlasten kann. Da bei OpenGL die Darstellung in einem Renderloop erzeugt wird, müssen Zustandsänderungen von außen als Variablen im Programm abgelegt werden, damit der Renderprozess diese in der nächsten Iteration rendern kann. Dies wirkt sich problematisch aus auf den Suchalgorithmus, den die künstliche Intelligenz verwendet: Dadurch, dass der Suchalgorithmus temporäre Züge in die Zukunft simuliert, dürfen diese temporären Zustände nicht an die GUI weitergegeben werden. Faktisch wird ein

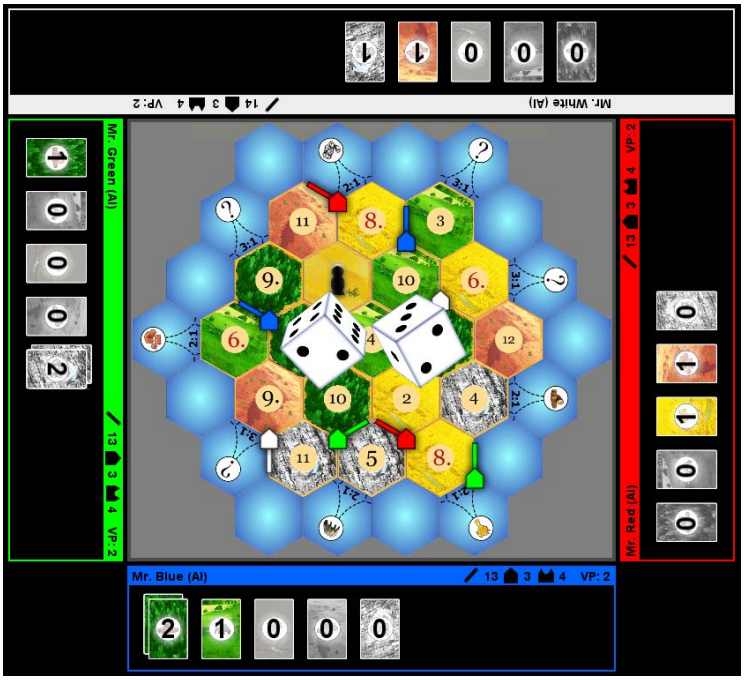



Abbildung 26: Screenshot von Java Settlers: Würfelwurf

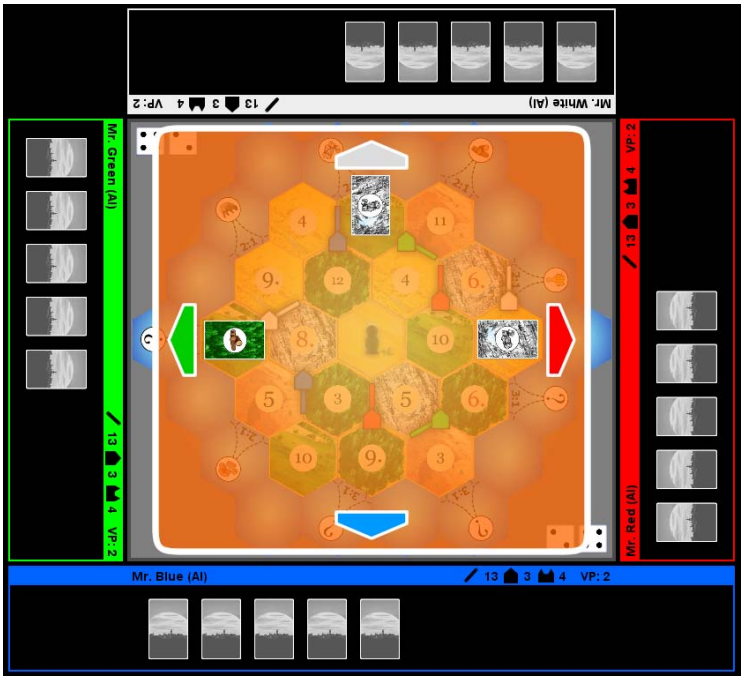



Abbildung 27: Screenshot von Java Settlers: Rohstoffverteilung nach Würfelwurf

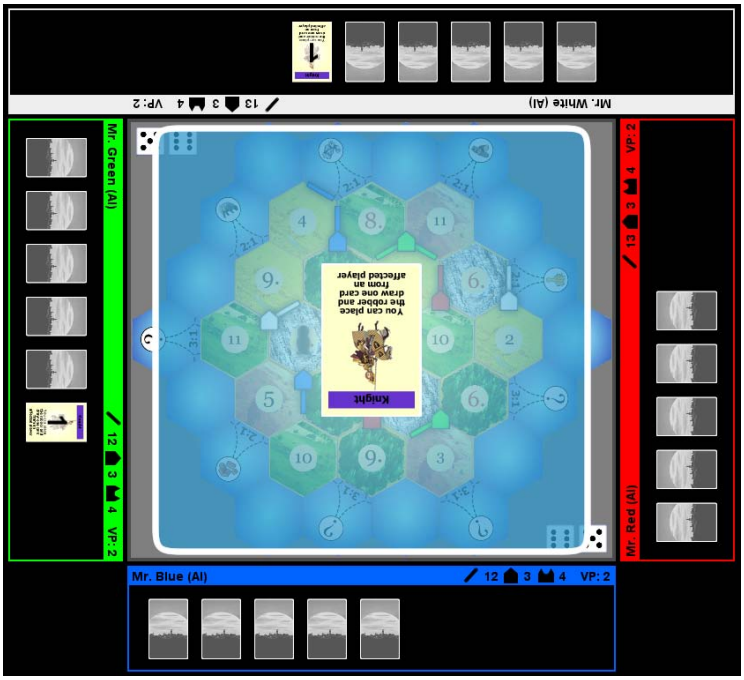



Abbildung 28: Screenshot von Java Settlers: Ausspielen einer Entwicklungskarte

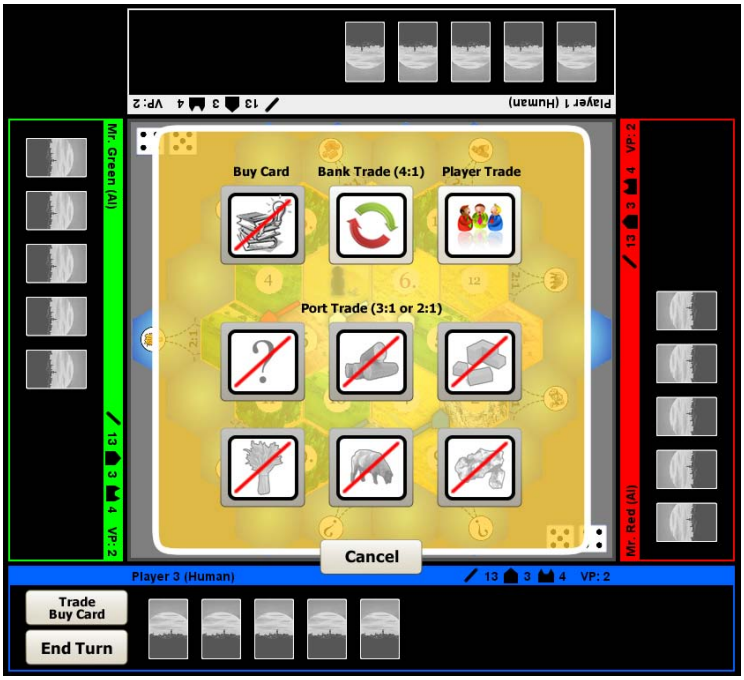



Abbildung 29: Screenshot von Java Settlers: Handelsmenü

interner Zustand für Berechnungen geführt und ein externer für die Darstellung. Beide Zustände müssen synchronisiert werden, wenn die Suche einen neuen Zug ermittelt hat. Jedoch erfordern einige Aktionen eines Spielers (z.B. die Anzeige der Ressourcenkarten wie oben erwähnt) eine sofortige Synchronisation der jeweiligen Daten unabhängig von den Daten anderer Spieler. Die GUI muss also für jeden Spieler und für jede Information einen eigenen Cache führen. Diese Caches lassen sich in OpenGL geschickt als Display Lists realisieren.

| | Spielbrett | Rohstoffe | Entwicklungskarten |
|---|---|---|---|
| Allgemein | ja, für Räuber | nein | nein |
| Spieler 1 | ja, für Spielfiguren | ja | ja |
| Spieler 2 | ja, für Spielfiguren | ja | ja |
| Spieler 3 | ja, für Spielfiguren | ja | ja |
| Spieler 4 | ja, für Spielfiguren | ja | ja |

Tabelle 3: Benötigte Caches für die Benutzeroberfläche

Neben OpenGL wurde noch OpenAL verwendet um dem Spiel eine Soundkulisse zu verpassen. In einer Schleife wird im Hintergrund Musik abgespielt, die über das Hauptmenü abgestellt werden kann. Gewinnt ein Spieler ertönt zusätzlich ein Sieg-Sound.

## 4.2 Schnittstelle zwischen Java Settlers und Multi-Touch

Die Anbindung an die Fingererkennung ist über eine Schnittstelle klar abgekapselt. Die Erkennung- und Verarbeitungsprozesse arbeiten parallel zur Spielapplikation und nehmen kalibrierte Bilder von der Kamera entgegen und liefern Touch-Objekte für die Applikation. Die Kamerabilder werden zunächst gemäß ihrer radialen Verzerrung entzerrt und dann in Grauwertbilder transformiert, um die Erkennung zu vereinfachen. Werden mehr als eine Kamera verwendet, so müssen die einzelnen Kamerabilder zu einem großen zusammengelegt werden. Einige Bildausschnitte überlappen sich zudem und müssen zusammengenäht werden, weshalb dieser Vorgang auch als *„stitching“* bezeichnet wird. Das somit entstandene große Kamerabild kann in der Verarbeitungskette weitergereicht werden.
Im nächsten Schritt müssen die Fingerkuppen von der restlichen Hand und eventuell dem Arm isoliert werden. Zunächst wird eine gewisse Hintergrundhelligkeit vom gesamten Bild abgezogen, sodass ein guter Kontrast zwischen Fingerkuppen und Hintergrund zu sehen ist. Danach wird das Bild mit einem Gauß-Kernel defokusiert und vom Originalbild abgezogen. Das defokusierte Bild entspricht einer verschwommenen Version, die erreicht wird, wenn sich die Finger etwas weiter weg von der Glasoberfläche bewegen. Durch die Subtraktion, einem *„Difference of Gaussian“*-Vorgang, werden die weiter von der Glasfläche befindlichen Körperteile aus dem

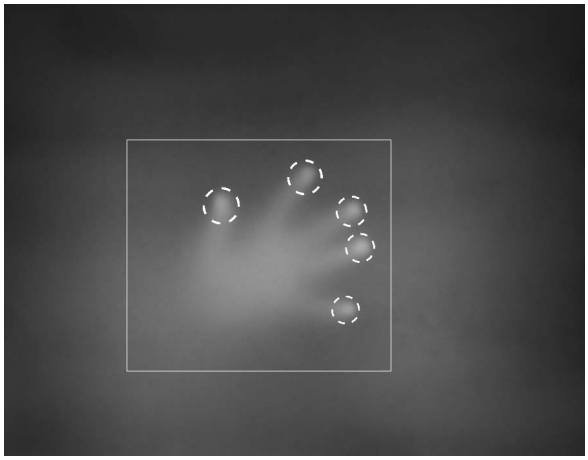
Bild abgezogen, sodass nur noch die Körperteile übrig bleiben, die am nächsten an der Glasoberfläche zu finden sind. Die ganze wirkt wie ein Hochpassfilter auf das Bild.

Um die Fingerkuppen noch eindeutiger zu identifizieren, können Algorithmen verwendet werden, die fingerkuppenähnliche Segmente im Bild erhalten und nicht ähnliche verwerfeb. Das segmentierte Bild der Fingerkuppen kann in der Intensität mit einer Intensitätserhöhung oder -senkung normalisiert werden. Das nächste Ziel wäre es nun die Fingerzentren zu finden, um dessen x- und y-Koordinaten als Touches an die Spielapplikation zu übermitteln. Dafür wird über die Intensitäten im Bild ein Gradientenabstiegsverfahren benutzt, um in diese Zentren mit höchster Intensität zu rutschen. Die Koordinaten dieser gefundenen Zentren können nun als Touch-Objekte in ein Netzwerk gesendet werden. Neben den Koordinaten ist es auch hilfreich mitzugeben, wie stark die Berührung war. Dies wird ermittelt, indem die Intensitäten zum Zentrum hin auf akkumuliert werden. Jede Applikation, z.B. unsere Spielapplikation, kann diese Touches nun aus dem Netzwerk abhören und auf diese reagieren. Beispielbilder dieser Verarbeitungskette werden in Abbildung 30 gezeigt.

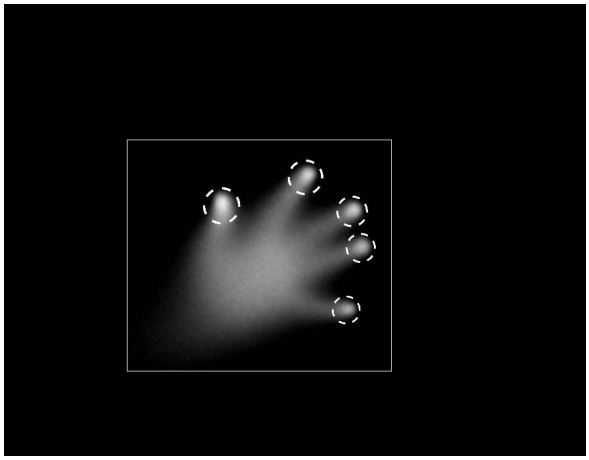
Für die Applikation ist es auch wichtig zu wissen, welcher Touch welchem Touch vorangegangen ist, sodass der Anwender eine ganze Reihe von Touches über die Zeit verfolgen kann, um somit Wischbewegungen eines Fingers nach zu verfolgen. Es wird also eine Touchzuordnung über die Zeit hinweg benötigt. Dies kann auf einfache Art und Weise lokal und *greedy* von statten gehen, ist aber nicht perfekt. In der Regel ist es so, dass das Finden der richtigen Zuordnung oder einer Zuordnung, die „gut genug“ ist, ziemlich rechenintensiv sein kann.

Die Spielapplikation auf der anderen Seite lauscht im Netzwerk nach diesen Touch-Informationen und kann auf neue Touches reagieren. Die Menge der neu eingetroffenen Touches wird für jedes Frame eingesammelt. Alle Touches unter einer entsprechenden Intensität werden raus gefiltert, sodass nur saubere klare Touches überleben. Für einfache Schaltflächen ist die Existenz eines Touches ausreichend, um die Schaltfläche zu aktivieren. Für Drag-and-Drop-Aktionen ist es nötig, die Zuordnung des Touches in die Vergangenheit zu überprüfen, um herauszufinden ob dieser Touch eine bestehende Drag-and-Drop-Aktion fortsetzt oder eine neue startet. Auch bei mehreren Touches ist es notwendig, die Zuordnung zu befragen, welche Touches Fortsetzungen zu welchen bereits abgearbeiteten sind.

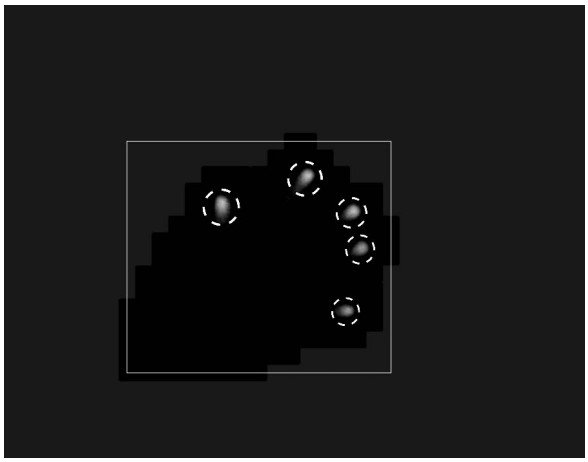
Java Settlers reagiert auf die Touches gemäß ihrer Anzahl, da in der Regel nur ein Spieler Aktionen ausführt. Ist es nur ein Touch, der registriert wurde, wird der mit Schaltflächen oder bestehenden Drag-and-Drop-Aktion assoziiert. Dies ist die häufigste Form der Touches. Sind es jedoch zwei Touches, könnten es zwei separate sein oder zwei die eine Zoom-Aktion ausführen. Diese Unterscheidung wird an Hand der Distanz und Position der beiden Touches bestimmt. Sind mehr als zwei Touches im Spiel, so sind das verschiedene Drag-and-Drop-Aktionen.



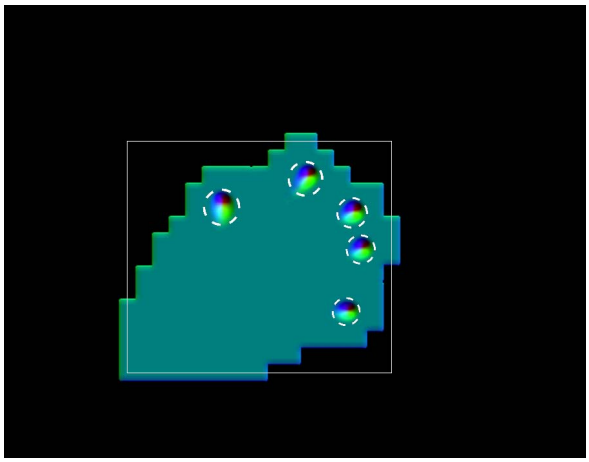
(a) Entzerrtes Bild



(b) Hintergrundsubtraktion



(c) Hochpassfilter



(d) Clustering

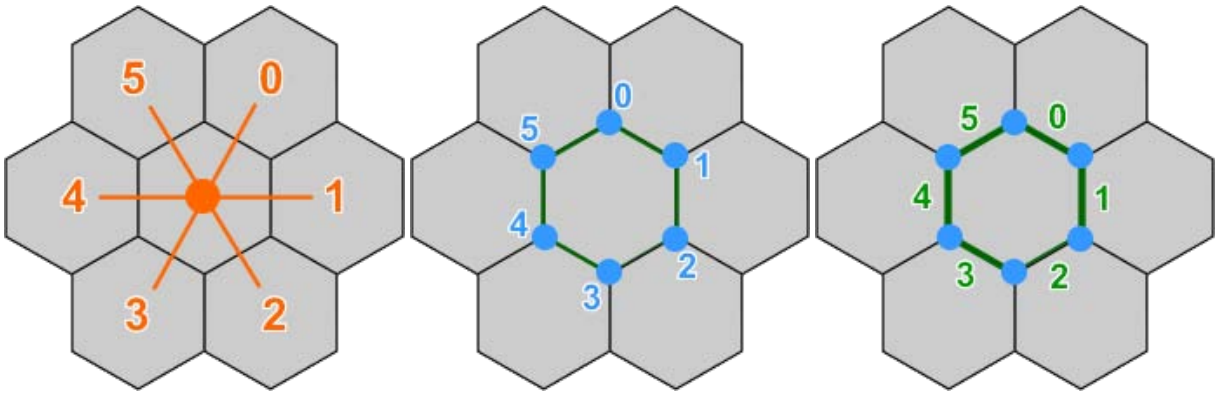
Abbildung 30: Multi-Touch Verarbeitungskette

## 4.3 Schnittstelle zwischen Java Settlers und jGameAI

Um das jGameAI Framework anzusprechen, bedarf es der Implementierung einer Repräsentation für einen Spielzustand und dem Zuggenerator, die nun vorgestellt werden sollen. Der Suchalgorithmus und die verwendeten Heuristiken und Bewertungsfunktion werden im nächsten Kapitel näher durchleuchtet.

### 4.3.1 Implementierung des Spielzustandes

Der Zustand entspricht in Java Settlers dem öffentlichen Spielbrett und den privaten Spielerinformationen. Die Spielerinformationen werden dabei soweit es geht zu den Spielerobjekten hinzu gespeichert. Für das Spielbrett wurde jedoch eine neue Datenstruktur entworfen, um die Informationen zu halten und Operationen auf diesen effizient durchzuführen. Das Spielbrett besteht aus hexagonalen Elementen, daher ist eine Modellierung als Matrixdarstellung mit Adressierung durch zwei Koordinaten nicht sinnvoll. Stattdessen wird es als Graph modelliert, was auch eine einfache Navigation innerhalb der Struktur ermöglicht. Dabei gibt es drei Adjazenzlisten, die unterschiedliche Graphen aufspannen. Die Abbildung 31 illustriert diese drei Graphstrukturen.




(a) HexGraph (b) NodeGraph (c) EdgeGraph

Abbildung 31: Repräsentation des Spielbrettes über drei Graphenstrukturen

Der *HexGraph* ist eine ungeordnete Liste an Hexes, wobei ein Hex für ein hexagonales Landschaftsfeld des Bretts steht. Für einen Hex können die benachbarten Hexes mittels einer Orientierung identifiziert werden. Außerdem kann für ein Hex und der gleichen Orientierung die zugehörigen Nodes und Edges ermittelt werden. Systemintern wird für jeden Hex eine eindeutige ID vergeben, mit der auf Un-/Gleichheit geprüft wird. Zudem kann der Rohstoff und der Token (die zu erwürfelnde Zahl) abgefragt werden und zusätzlich überprüft werden, ob der Räuber diesen Hex besetzt.
Der *NodeGraph* ist eine ungeordnete Liste an Nodes, wobei ein Node für einen Kreuzpunkt des

Bretts steht, auf der Siedlungen oder Städte platziert werden können. Jeder Node hat mindestens zwei und maximal drei benachbarte Nodes und somit mindestens zwei und maximal drei ausgehende Edges. Der Anwender kann diese Nachbarnodes bzw. -edges von jedem Node-Objekt abrufen, wobei nicht vorhandene Nodes / Edges ein null-Objekt zurückliefern. Ein Node gehört zu mindestens einem Hex und zu maximal drei Hexes, die ebenfalls ermittelt werden können. Auch die Nodes werden mit einer ID identifiziert. Zudem trägt jeder Node einen Wert, der angibt, ob er frei ist oder von einer Siedlung oder Stadt eines bestimmten Spielers besetzt ist.
Der *EdgeGraph* schließlich ist eine ungeordnete Liste an Edges, wobei ein Edge eine Kante im Brett darstellt, auf der Straßen platziert werden können. Jede Edge besteht aus zwei Nodes und hat bis zu vier benachbarte Edges. Auch hier wird mit einer ID identifiziert und jede Edge besitzt einen Wert, der angibt, ob er frei ist oder von einer Straße eines bestimmten Spielers besetzt ist.
Mit Hilfe dieser drei Listen ist es möglich beliebig im Graphen zu traversieren und viele benötigte Anfragen lösen, z.B. welche Spieler etwas an einem Hex gebaut haben? Oder welche Spieler Zugang zu Weizen haben? Oder welcher Spieler die Längste Handelsstraße besitzt? Um z.B. die letztgenannte Anfrage zu lösen, wurde über die Graphenstruktur eine Variante der Tiefensuche implementiert, die im Codefragment 1 im Anhang A.3 wiederzufinden ist.

### 4.3.2 Implementierung des Zuggenerators

Der Zuggenerator ist direkt an den Zustand gebunden, denn nur aus einem bestimmten Zustand heraus lassen sich die legalen Züge erzeugen. Daher entspricht jeder Zustand einer Klassifizierung, die charakteristisch dafür ist, welche Züge erlaubt werden. Für Java Settlers gibt es lediglich drei Klassen: die *PREPHASE*, die *ROBBERPLACEPHASE* und die *BUILDTRADEPHASE*. Jede dieser Klassen definiert eine Menge an Zügen, die erlaubt sind, wobei die Züge wiederum in Zugsorten unterteilt werden. Die PREPHASE ist die Anfangsphase vor dem Würfeln, wo jeder Spieler reihum seine ersten Siedlungen und Straßen auf das Spielbrett platziert. Hier gibt es nur die Möglichkeit, einer der Züge aus den Zugsorten Siedlungsbau oder Straßenbau zu wählen. Die ROBBERPLACEPHASE wird angestoßen, wenn nach einer gewürfelten sieben der Spieler den Räuber platzieren muss. Hier ist nur ein Zug aus der Sorte „Räuber platzieren“ möglich. Letztendlich kann ein Spieler in der BUILDTRADEPHASE sein, nachdem er seinen Würfelwurf getätigt hat. Hier kann der Spieler aus verschiedenen Zugsorten einen Zug wählen, z.B. Städtebau, Siedlungsbau, Straßenbau, Entwicklungskarte kaufen, Entwicklungskarte spielen, Handel tätigen oder Runde beenden.
Zudem gibt es noch eine spezielle Klasse für Zufallszüge, die nur zu bestimmten Zeitpunkte zum Einsatz kommen: Dies ist der Erhalt einer zufälligen Entwicklungskarte vom Stapel nachdem ein Spieler eine Entwicklungskarte kauft, der Erhalt einer Rohstoffkarte nachdem ein Spieler bei einem Gegner über den Räuber eine Karte zieht oder der Würfelwurf als einzig möglicher

erster Zug, wenn ein Spieler an die Reihe kommt. Durch diese endliche Anzahl an möglichen Zuständen kann der Zuggenerator stets eine Liste an legalen Zügen definieren, woraus sich der menschliche Spieler oder Computerspieler den passenden Zug wählt.
Was im Spiel bisher fehlt, ist die Fähigkeit des Computerspielers, automatisch Handelsvorschläge zu generieren um somit kooperative Handel abzuschließen. Davon abgesehen, sind nochmal alle möglichen Zugsorten in der Tabelle 4 aufgeführt.

| Klassifikation | Zugsorte | Erklärung |
|---|---|---|
| PREPHASE | BUILD_PREPHASE_ROAD | Bau einer Straße in der Anfangsphase |
| PREPHASE | BUILD_PREPHASE_SETTLEMENT | Bau einer Siedlung in der Anfangsphase |
| ROBEBRPLACEPHASE | ROBBER_PLACE | Platziere den Räuber und ziehe eine Karte von einem betroffenen Mitspieler |
| BUILDTRADEPHASE | BUILD_ROAD | Bau einer Straße |
| BUILDTRADEPHASE | BUILD_CITY | Bau einer Siedlung |
| BUILDTRADEPHASE | BUILD_SETTLEMENT | Bau einer Stadt |
| BUILDTRADEPHASE | BUY_DEVCARD | Kauf einer Entwicklungskarte |
| BUILDTRADEPHASE | BANK_TRADE | Ein Handel mit der Bank |
| BUILDTRADEPHASE | PLAY_DEVCARD_DISCOVERY | Das Ausspielen einer Entwicklungskarte Entdeckung (Erhalt zwei beliebiger Rohstoffe) |
| BUILDTRADEPHASE | PLAY_DEVCARD_KNIGHT | Das Ausspielen einer Ritterkarte |
| BUILDTRADEPHASE | PLAY_DEVCARD_MONOPOLY | Das Ausspielen einer Entwicklungskarte Monopol (Erhalt aller Rohstoffe eines Typs) |
| BUILDTRADEPHASE | PLAY_DEVCARD_ROADBUILDING | Das Ausspielen einer Entwicklungskarte Straßenbau (Kostenloser Bau von zwei Straßen) |
| BUILDTRADEPHASE | PLAY_DEVCARD_VICTORYPOINT | Das Ausspielen einer Siegpunktkarte |
| BUILDTRADEPHASE | PLAYER_TRADE | Ein Handel zwischen Spieler |
| BUILDTRADEPHASE | END_TURN | Beendet die Runde |
| Zufallszug | ROLL_DICE | Würfelwurf |
| Zufallszug | RECEIVE_DEVCARD | Erhalt einer zufälligen Entwicklungskarte |
| Zufallszug | ROBBER_STEAL | Erhalt eines zufälligen Rohstoffs von einem anderen Spieler |

Tabelle 4: Zugsorten für Java Settlers

# 5 Spiellogik

*„A strange game. The only winning move is not to play. How about a nice game of chess?“ - Joshua (Wargames 1983) [67]*

Einen noch offenen Punkt dieser Arbeit bildet die Künstliche Intelligenz (KI) im Spiel. Wie ist diese realisiert worden und wie gut verhält sie sich? Im Folgenden soll eine spieltheoretische Sicht auf Java Settlers betrachtet werden, um die Herausforderungen einer künstlichen Intelligenz zu identifizieren. Demnach wird der verwendete Suchalgorithmus erläutert sowie generelle Strategien und spezielle Heuristiken zur Reduktion des Suchbereichs präsentiert. Die Bewertungsfunktion zur Evaluation und Vergleich verschiedener Stellungen soll nicht unerwähnt bleiben, denn sie ist essentiell für die Qualität und Wettbewerbsfähigkeit der künstlichen Intelligenz.

## 5.1 Eine spieltheoretische Sicht auf Java Settlers

Betrachtet wir „Die Siedler von Catan“ oder speziell Java Settlers aus einer spieltheoretischen Sicht, fallen gleich mehrere Unterschiede gegenüber traditionellen Spielen ins Auge. Zunächst ist Java Settlers nach der Klassifizierung der Arbeitsumgebung aus Kapitel 2.3.1 eine partiell kooperative Multiagenten-Umgebung und kein Zwei-Personen-Spiel. Zusätzlich handelt es sich nicht um ein Nullsummenspiel, d.h. ein Zug meines Gegners muss meine Stellung im Spiel nicht wahlweise verschlechtern. So könnte es durch Kooperationen Züge (z.B. Handel) geben, die mehrere Parteien verbessern.
Außerdem haben wir es mit einer stochastisch und teilweise beobachtbaren Umgebung zu tun, in der Wahrscheinlichkeiten und somit unvollständige Informationen den Suchbaum in die Breite ziehen. Für jeden möglichen Würfelwurf (2-12) müssen alle Alternativen berechnet und gewichtet kombiniert werden. Gleichermaßen ist es schwierig zu entscheiden, wie sehr ein unwahrscheinlicher Zug in die Bewertung mit einfließt.
Während ein Spieler an der Reihe ist, kann der Spieler mehrere Züge hintereinander ausführen und dann entscheiden an den nächsten Spieler weiterzugeben, wenn er keine weiteren Züge mehr ausführen kann oder will. Im Gegensatz zu den traditionellen Spielen wechselt somit der aktuelle Spieler nicht nach jedem Zug, sondern nach einer unregelmäßigen Anzahl an Zügen. Damit wird der generierte Suchbaum nicht mehr regulär, was eine Suche in eine gewisse Spielertiefe berücksichtigen muss.
Insgesamt wird ein riesiger Suchbaum aufgebaut, in dem es ähnlich wie bei Schach nicht effizient ist, tief zu suchen. Durch die vielen möglichen Strategien, Handelszüge und Würfelwahrscheinlichkeiten wächst der Suchbaum umso mehr in die Breite. Ein Ziel muss es also sein, die Suche stets möglichst klein zu halten und innerhalb einer vertretbaren Zeit eine vernünftigen Zug zu errechnen.

## 5.2 Verwendeter Suchalgorithmus

Der Suchbaum, der erstellt wird, entspricht einem üblichen Spielbaum. Die Wurzel kennzeichnet die aktuelle Spielstellung, während mögliche Aktionen als Kanten zu neuen Spielstellungen als Kinderknoten eine Ebene tiefer führen können. Der Baum wird bis zu einer gewissen Spielertiefe für jeden Suchschritt online erstellt.
Der verwendete Suchalgorithmus entspricht einer Minimax-Suche für mehrere Spieler. Die Suchtiefe orientiert sich dabei nach Spielerwechsel und nicht nach Anzahl der Züge, da ein Spieler mehrere Züge hintereinander ausführen könnte. Aus diesem Grund entspricht die Suchtiefe nicht der Tiefe des Suchbaums, stattdessen wurde ein künstlicher Zug eingeführt, um eine Runde zu beenden und an den nächsten Spieler weiterzugegeben. Wird solch ein Zug in der Suche durchlaufen, erhöht sich die Suchtiefe um Eins, da ein Spielerwechsel stattgefunden hat. Ist die maximale Spielertiefe erreicht, endet die Suche für diesen Ast und bewertet den aktuellen Blattknoten als Nutzenvektor für jeden Spieler mit Hilfe der jeweiligen Bewertungsfunktionen.
Beim Traversieren des Suchbaums müssen Zufallszüge gesondert behandelt werden, denn sie können mit verschiedenen Wahrscheinlichkeiten andere Effekte auslösen. Daher werden bei einem Zufallszug alle möglichen Folgezüge als Kinderknoten generiert und die Kanten mit der Wahrscheinlichkeit versehen. Die Bewertungen der Kinderknoten werden nach oben gereicht, indem jeder mit der jeweiligen Wahrscheinlichkeit multipliziert wird und alle zusammen addiert werden. Damit wird der erwarteten Nutzenwert ermittelt.
Die ermittelten Nutzenvektoren der evaluierten Stellungen müssen vom suchenden Spieler miteinander verglichen werden, um die beste Alternative auszuwählen. Dabei werden jedoch weder die erwähnten max$^n$- oder die „paranoide“ Variante gewählt, denn beide berücksichtigen nicht die Entwicklung der Gegenspieler. Dies kann an einem Beispiel gezeigt werden: Es werden ein Nutzenvektor $\vec{a} = <a_1, a_2, a_3>$ für drei Spieler mit einem Nutzenvektor $\vec{b} = <b_1, b_2, b_3>$ verglichen. Sei $\vec{a} = <3, 1, 2>$ und $\vec{b} = <4, 8, 7>$. Spieler 1 würde sowohl bei der max$^n$- als auch bei der „paranoiden“ Variante $\vec{b}$ aufgrund des Maximalisieren bevorzugen. Jedoch befindet der Spieler sich bei $\vec{a}$ in der Führungsposition und ist bei $\vec{b}$ als Schlusslicht positioniert.
Anstatt direkt die absoluten Nutzenvektoren zu vergleichen, sollten sie zunächst normiert und in einen relativen Nutzenvektoren überführt werden. Dabei sollte der eigene Nutzenwert mit dem der Gegenspieler verglichen werden. Eine Möglichkeit, dies zu realisieren, wäre es den Abstand zum führenden Gegenspieler zu messen. Ein relativer Nutzenvektor $\vec{q} = <q_1, \ldots, q_n>$ wird aus einem absoluten Nutzenvektor $\vec{p} = <p_1, \ldots, p_n>$ folgendermaßen gewonnen:

$$q_i = p_i - \max_{j \neq i}(p_j) \tag{1}$$

Die Formel (1) orientiert sich jedoch nur am führenden Gegenspieler und lässt andere Gegenspieler außer Acht. Bei $\vec{a} = <1, 2, 7>$ und $\vec{b} = <1, 6, 7>$ würde Spieler 1 beide Situationen

gleich bewerten, weil der Abstand zum führenden Spieler 3 in beiden Fällen 6 beträgt. Jedoch wäre die erstere Situation zu favorisieren, da der Abstand zu Spieler 2 geringer ist. Eine bessere Variante wäre es, auf den Durchschnitt zurückzugreifen und einen relativer Nutzenvektor über den Mittelwert zu berechnen:

$$\begin{aligned} q_i &= \frac{1}{n-1}\sum_{j=1}^{n}(p_i - p_j) \\ &= p_i - \frac{1}{n-1}\sum_{j\neq i} p_j \end{aligned} \tag{2}$$

Der Durchschnitt durch Formel (2) ist jedoch empfindlich für Ausreißer. Zudem werden die Vektoren $\vec{a} = <3, 1, 7>$ und $\vec{b} = <3, 4, 4>$ für Spieler 1 gleich bewertet, weil in beiden Fällen der Abstand zum Mittelwert der Gegner 1 beträgt. Die zweite Situation wäre jedoch zu bevorzugen, da alle Spieler fast homogene Nutzenwerte aufweisen und Spieler 1 eine größere Chance hat in Führung zu gehen. Eine geeignetere Vergleichsmethode wäre, die Unterschiede bzw. die Ordnung zwischen den Gegenspieler mit einzubeziehen. Gemeinsam mit Maro Bader und Marco Block aus der AG Spieleprogrammierung an der Freien Universität Berlin [49] wurde eine Entropie-Nutzenfunktion geschaffen, die eine gute Generalisierungsfähigkeit für viele Spiele besitzt:

$$q_i = H(p_i) \cdot (p_i - \mu) \tag{3}$$

mit

$$H(p_i) = \frac{\sum_{j\neq i} p_j \cdot \log_2(p_j)}{\sum_{j\neq i} \log_2(p_j)} \tag{4}$$

und

$$\mu = \frac{1}{n}\sum_{j=1}^{n} p_j$$

Die Entropiefunktion (4) berechnet die normalisierte Entropie, die in der Informationstechnik den Erwartungswert des Informationsgehaltes eines Alphabetes repräsentiert [68]. Anders interpretiert gibt die Entropie ein Maß für die Wahrscheinlichkeit gleicher Zeichen aus dem Alphabet oder ein Maß der Unordnung. Die Entropie bildet gemeinsam mit dem Abstand zum Mittelwert eine Entropie-Nutzenfunktion (3), die sich an der eigenen Bewertung, dem Durchschnitt der Bewertungen und der Ordnung der Gegner orientiert. Somit bildet (3) auch eine gute Vergleichsmethode für Java Settlers und wurde später auch in das jGameAI-Framework eingebaut.

## 5.3 Verwendete Heuristiken

Der Suchraum weist sich selbst bei einer Spielertiefe von vier als zu groß auf, um ihn in einer akzeptablen Zeit komplett zu durchsuchen. Es bedarf also Heuristiken, die den Suchraum in einer Art beschneiden, sodass das Ergebnis nicht minder schlecht wird. Dafür werden aus einer möglichen Zugsorte nur die $n$ besten für die Erstellung des Suchbaums gewählt. Somit sind über die einzelnen $n$-Werte der Zugsorten ein Kompromiss einstellbar zwischen einer qualitativ guten Suche und einer schnellen Suche.
Die Heuristiken basieren gemäß den in Kapitel 2.1.6 vorgestellten Strategien. Dementsprechend werden Siegpunkte und Rohstofferträge als wichtigste Indikatoren für Bewertungen angesehen. Durch den exponentiellen Rohstoffertrag erweist es sich für die Anfangsphase als sehr wichtig, eine produktive Initialstellung zu besitzen. Zu Beginn eines Spiels wird also eine Produktivitäts-Strategie gefahren, während im Verlauf des Spiels je nach Situation zu einer Erz-Getreide-Strategie, einer Holz-Lehm-Strategie oder einer Monopol-Strategie gewechselt wird.
Um die Produktivität für eine Stellung zu ermitteln, wird eine abgewandelte Form des *Estimated Time to Build*-Wertes (ETB-Wert) berechnet, der bereits in vorhergehenden Arbeiten eingesetzt wurde [25]. Dieser Wert gibt für eine Stellung die erwartete Anzahl an Runden an, die ein Spieler warten muss, bis er genug Rohstoffe besitzt, um eine Aktion durchzuführen. So berechnet der *ETB-Siedlungswert* die zu erwartete Rundenanzahl und somit die zu erwartete Anzahl an Würfelwürfen, um die Rohstoffe zu erhalten, die zum Bau einer Siedlung benötigt werden. Analog dazu berechnet der *ETB-Stadtwert* dasselbe für eine Stadt, der *ETB-Straßenwert* für eine Straße und der *ETB-Entwicklungskartenwert* für eine Entwicklungskarte. Der *ETB-Gesamtwert* bildet eine gewichtete Summe aus dem ETB-Siedlungswert, dem ETB-Stadtwert, dem ETB-Straßenwert und dem ETB-Entwicklungskartenwert und ist somit eine gute Repräsentation für die Produktivität eines Spielers. Bei der Berechnung gehen Rohstofferträge bei Würfelwürfen ein, aber auch Handelsmöglichkeiten mit der Bank. Der Handel mit Mitspielern kann aufgrund der Unvorhersehbarkeit nicht prognostiziert werden.
Um einen spezifischen ETB-Wert für einen Spieler zu berechnen, bedarf es zwei Schritte. Zunächst wird eine Frequenztabelle errechnet, in der für jeden Rohstoff eingetragen ist, in wie vielen Runden der Spieler im Schnitt den Rohstoff zu erwarten hat. Mit Hilfe der Frequenztabelle können wir danach nun einen spezifischen ETB-Wert iterativ ausrechnen.

Betrachten wir für die Berechnung von ETB-Werten folgende beispielhafte Stellung in Abbildung 32.

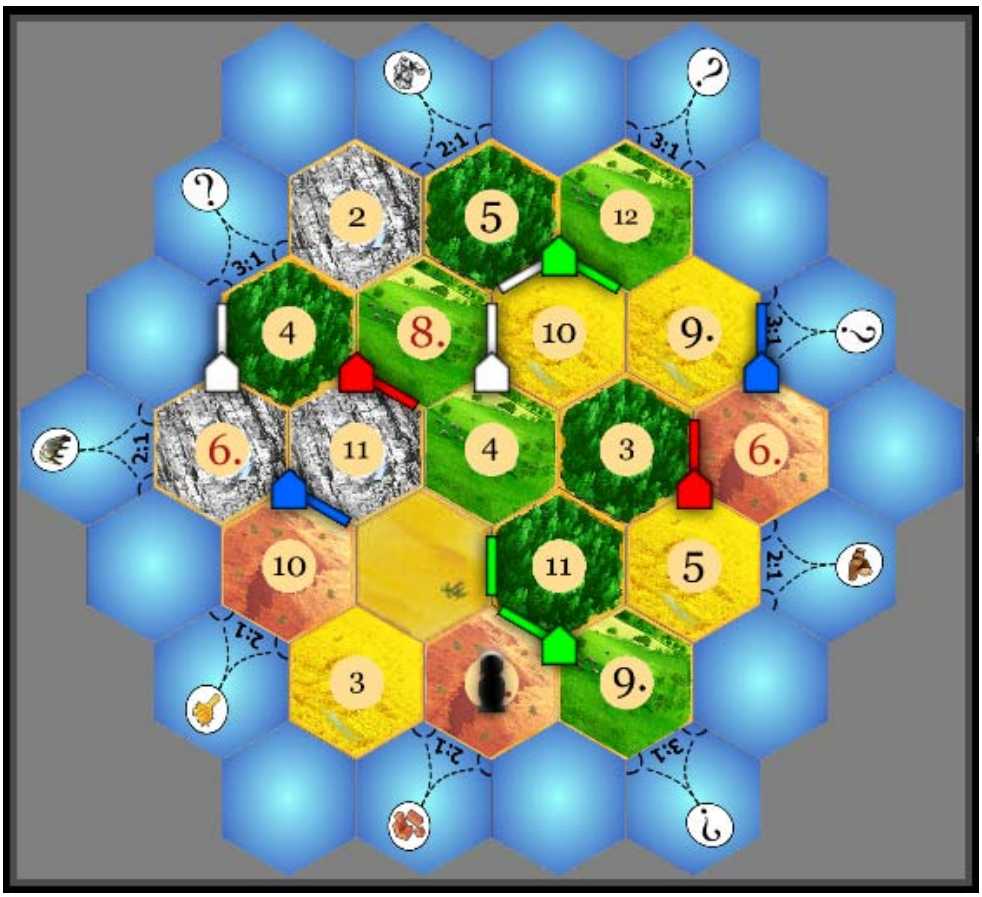



Abbildung 32: Eine beispielhafte Stellung für die ETB-Berechnung

Das Spielbrett zeigt eine Stellung im Spiel. Für uns sind in diesem Beispiel nur Spieler Weiß und Spieler Blau interessant. Spieler Weiß hat eine Siedlung an der westlichen Küste (angrenzend an Erz mit 6 und Holz mit 4) und noch eine zweite Siedlung (angrenzend an Wolle mit 4 oder 8 und Getreide mit 10). Spieler Blau hat eine Siedlung an der östlichen Küste (angrenzend an Lehm mit 6 und Getreide mit 9) und eine zweite Siedlung (angrenzend an Lehm mit 10 und Erz mit 6 oder 11).

Betrachten wir dir Berechnung des ETB-Siedlungswerts für Spieler Weiß. Spieler Weiß besitzt zwei Siedlungen, wobei er bei einer gewürfelten 5 oder 11 einmal Holz als Rohstoff erhalten würde. Die Wahrscheinlichkeit laut Tabelle 1 für eine 5 liegt bei $\frac{4}{36}$ und für 11 bei $\frac{2}{36}$ und somit in der Summe bei $\frac{6}{36}$. Die Frequenz entspricht dem Kehrwert 6 ist ein ungefähres Maß für die durchschnittliche Anzahl an Runden, die auf den Rohstoff gewartet werden muss. Beträgt die Wahrscheinlichkeit 0, so wird die Frequenz mit $-1$ notiert, was angibt, dass dieser Rohstoff durch Würfelwurfe alleine nicht zu erhalten ist. Für Städte wird die Wahrscheinlichkeit verdoppelt, da Städte doppelt so viele Rohstoffe erwirtschaften wie Siedlungen. Die Tabelle 5 zeigt die

Berechnung der Frequenztabelle für Spieler Weiß.

| | Holz | Wolle | Getreide | Lehm | Erz |
|---|---|---|---|---|---|
| Würfelzahlen | 5,11 | 9,12 | 10 | 6 | - |
| Wahrscheinlichkeiten | $\frac{4}{36}, \frac{2}{36}$ | $\frac{4}{36}, \frac{1}{36}$ | $\frac{3}{36}$ | $\frac{5}{36}$ | 0 |
| summierte Wahrscheinlichkeit | $\frac{6}{36}$ | $\frac{5}{36}$ | $\frac{3}{36}$ | $\frac{5}{36}$ | 0 |
| Frequenz | 6 | 7 | 12 | 7 | $-1$ |

Tabelle 5: Beispiel zur Berechnung der Frequenz für Spieler Weiß anhand des Beispiels

Sind wir z.B. am ETB-Siedlungswert von einem Spieler interessiert, so können wir die benötigten Ressourcen in einem Vektor $(-1,-1,-1,-1,0)$ für $(Holz, Wolle, Getreide, Lehm, Erz)$ notieren. Wir zählen nun die zu erwartenden Runden von 0 hoch und addieren neue Rohstoffe gemäß der Frequenztabelle. Ist die Rundenanzahl modulo der Frequenz gleich 0, erhalten wir den entsprechenden Rohstoff. Besteht unser Vektor nur noch aus nicht negativen Werten, so haben wir die benötigten Rohstoffe und geben die erreichte Rundenanzahl als ETB-Wert zurück.

| Runde | Holz | Wolle | Getreide | Lehm | Erz |
|---|---|---|---|---|---|
| 0 | -1 | -1 | -1 | -1 | 0 |
| 1 | -1 | -1 | -1 | -1 | 0 |
| 2 | -1 | -1 | -1 | -1 | 0 |
| 3 | -1 | -1 | -1 | -1 | 0 |
| 4 | -1 | -1 | -1 | -1 | 0 |
| 5 | -1 | -1 | -1 | -1 | 0 |
| 6 | 0 | -1 | -1 | -1 | 0 |
| 7 | 0 | 0 | -1 | 0 | 0 |
| 8 | 0 | 0 | -1 | 0 | 0 |
| 9 | 0 | 0 | -1 | 0 | 0 |
| 10 | 0 | 0 | -1 | 0 | 0 |
| 11 | 0 | 0 | -1 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |

Tabelle 6: Beispiel zur Berechnung des ETB-Siedlungwert anhand des Beispiels

Wir erwarten also, dass wir in 12 Runden eine Siedlung erbauen können und hätten sogar noch ein Holz an Überschuss (welcher aber ignoriert wird).
Manchmal reicht es nicht aus, Rohstoffe über Würfelwürfe zu erhalten, sondern es ist zusätzlich noch nötig mit der Bank zu handeln. Dabei kann ein Spieler entweder 4 zu 1 handeln oder wenn ein entsprechender Hafen besetzt ist, auch 3 zu 1 oder 2 zu 1. Betrachten wir dafür als Beispiel

den ETB-Straßenwert von Spieler Blau. Spieler Blau besitzt eine Siedlung an einem 3:1-Hafen und kann somit 3 gleiche Rohstoffe gegen einen beliebigen Rohstoff von der Bank tauschen. Wir berechnen erneut die Frequenztabelle für Spieler Blau:

| | Holz | Wolle | Getreide | Lehm | Erz |
|---|---|---|---|---|---|
| Würfelzahlen | - | - | 9 | 6,10 | 6,11 |
| Wahrscheinlichkeiten | 0 | 0 | $\frac{4}{36}$ | $\frac{5}{36}, \frac{3}{36}$ | $\frac{5}{36}, \frac{2}{36}$ |
| summierte Wahrscheinlichkeit | 0 | 0 | $\frac{4}{36}$ | $\frac{8}{36}$ | $\frac{7}{36}$ |
| Frequenz | $-1$ | $-1$ | 9 | 4 | 5 |

Tabelle 7: Beispiel zur Berechnung der Frequenz für Spieler Blau anhand des Beispiels

Bei der Berechnung des ETB-Straßenwerts erhalten wir nie den Rohstoff Holz aus Würfelwürfen. Stattdessen müssen wir mit der Bank 3:1 handeln, um den Rohstoff Holz zu erhalten. Wann und wie gehandelt werden soll bei der Berechnung, läuft greedy ab: Es wird so früh wie möglich der am meist benötigte Rohstoff umgetauscht. Für den ETB-Straßenwert erhalten wir in diesem Beispiel 15 Runden, nachdem wir 3 Erz gegen ein Holz getauscht haben.

| Runde | Holz | Wolle | Getreide | Lehm | Erz |
|---|---|---|---|---|---|
| 0 | -1 | 0 | 0 | -1 | 0 |
| 1 | -1 | 0 | 0 | -1 | 0 |
| 2 | -1 | 0 | 0 | -1 | 0 |
| 3 | -1 | 0 | 0 | -1 | 0 |
| 4 | -1 | 0 | 0 | 0 | 0 |
| 5 | -1 | 0 | 0 | 0 | 1 |
| 6 | -1 | 0 | 0 | 0 | 1 |
| 7 | -1 | 0 | 0 | 0 | 1 |
| 8 | -1 | 0 | 0 | 1 | 1 |
| 9 | -1 | 0 | 1 | 1 | 1 |
| 10 | -1 | 0 | 1 | 1 | 2 |
| 11 | -1 | 0 | 1 | 1 | 2 |
| 12 | -1 | 0 | 1 | 2 | 2 |
| 13 | -1 | 0 | 1 | 2 | 2 |
| 14 | -1 | 0 | 1 | 2 | 2 |
| 15 | -1 | 0 | 1 | 2 | 3 |
| 15a | 0 | 0 | 1 | 2 | 0 |

Tabelle 8: Beispiel zur Berechnung des ETB-Straßenwert anhand des Beispiels

Der ETB-Wert lässt sich also relativ leicht iterativ für jeden Spieler berechnen. Der passende Code aus Java Settlers ist im Codefragment 2 aufgeführt. Mit dieser Methode können für viele Zugsorten die *n* besten herausgefiltert und somit der Suchbaum beschnitten werden.

Für die Anfangsphase ist es aber auch notwendig zu entscheiden, ob ein Spieler aus einer Initialposition gut expandieren kann. Dafür kann der ETB-Wert ein wenig modifiziert werden, indem die Frequenzberechnung nicht nur die Rohstofffelder einbezieht, zu denen direkter Zugang besteht, sondern zusätzlich noch jene Rohstofffelder einkalkuliert, zu denen nach einer Expansion Zugang bestehen würde. Dieser *ETBExpandingOptions*-Wert wird vor allem für die Straßenbauzüge gewählt.

Für Züge, die den Räuber platzieren sollen, ist der ETB-Wert kein geeignetes Maß, da der Räuber in der Berechnung nicht einbezogen wird. Hierfür wurde eine sehr einfache Räuber-Heuristik entwickelt, die versucht den Räuber möglichst fern von einem selber zu halten und zu einem Spieler zu stellen, dem dies schadet oder es uns hilft. So kann es der Spieler mit den meisten Siegpunkten sein, den wir auf dem Weg zum Sieg hindern wollen. Andererseits könnte auch der zu blockierende Rohstoff auschlaggebend sein, um anderen Spielern den Zugang zum Rohstoff zu verweigern. Letztlich muss noch ein Opfer gewählt werden, von dem der Täter eine Rohstoffkarte entnehmen darf. Dies orientiert sich nach der Wahrscheinlichkeit, den Rohstoff zu erhalten, der für einen zu dem Zeitpunkt am wichtigsten erscheint.

Die Tabelle 9 fasst nochmal die eingesetzten Heuristiken zur Filterung für die verschiedenen Zugsorten zusammen.

| Zugsorte | Verwendete Heuristik |
|---|---|
| BUILD_PREPHASE_ROAD | ETBExpandingOptions |
| BUILD_PREPHASE_SETTLEMENT | ETB-Gesamtwert |
| ROBBER_PLACE | Räuber-Heuristik |
| BUILD_ROAD | ETBExpandingOptions |
| BUILD_CITY | ETB-Gesamtwert |
| BUILD_SETTLEMENT | ETB-Gesamtwert |
| BUY_DEVCARD | ETB-Gesamtwert |
| BANK_TRADE | ETB-Gesamtwert |
| PLAY_DEVCARD_DISCOVERY | ETB-Gesamtwert |
| PLAY_DEVCARD_KNIGHT | Räuber-Heuristik |
| PLAY_DEVCARD_MONOPOLY | ETB-Gesamtwert |
| PLAY_DEVCARD_ROADBUILDING | ETBExpandingOptions |
| PLAY_DEVCARD_VICTORYPOINT | keine Filterung |
| PLAYER_TRADE | keine Filterung |
| END_TURN | keine Filterung |

Tabelle 9: Verwendete Heuristiken zur Filterung der verschiedenen Zugsorten

## 5.4 Verwendete Bewertungsfunktion

Wird ein Kindknoten gemäß der Suchtiefe bei der Suche erreicht, so wird diese Stellung $S_t$ ausgewertet, indem mittels einer Bewertungsfunktion $f$ ein jeweiliger Nutzenwert für jeden Spieler ermittelt. Dieser differenziert zwischen schlechten bis hin zu verlorenen Stellungen (negativer Nutzenwert) und guten bis hin zu gewonnenen Stellungen (positiver Nutzenwert). Je detaillier die Bewertungsfunktion zwischen verschiedenen Stellungen unterscheiden kann, umso akkurater ist die Beurteilung der Spielsituation und dementsprechend besser die ermittelten Züge der Suche. Dabei besteht die Bewertungsfunktion $f$ aus einer linear gewichteten Summe kleiner Bewertungsfunktionen $g_i$ für $n$ verschiedene Kriterien im Spiel, ausgedrückt in der Formel 5. Durch diesen linearen Zusammenhang können die Gewichte durch Reinforcement Learning-Techniken trainiert werden [39, 43].

$$f(S_t) = \sum_{i=1}^{n} w_i f_i \tag{5}$$

Java Settlers benutzt eine Vielzahl an kleinen Bewertungsfunktionen, die sich an den erwähnten spieltypischen Strategien orientieren. Die wichtigste orientiert sich an der Anzahl an Siegpunkten, denn diese entscheiden bekanntlich über Sieg oder Niederlage. Dabei unterscheidet Java Settlers zwischen der öffentlich sichtbaren und der privaten geheimen Siegpunktanzahl. So zählen zu den öffentlichen Siegpunkten all jene, die von allen Spielern einsehbar sind (z.B. Siedlungen, Städte, Sonderkarten, bereits ausgespielte Siegpunktkarten). Zu den privaten zählen zusätzlich noch die Siegpunkte, die ein Spieler versteckt hält, aber jederzeit ausspielen könnte (z.B. noch nicht ausgespielte Siegpunktkarten). Für den eigenen Spieler ist es sinnvoll, die private Siegpunktanzahl mit der öffentlichen Siegpunktanzahl der Gegenspieler zu vergleichen. Eine weitere Bewertung, die in die Bewertung einbezogen wurde, ist die Rohstoffproduktivität einer Stellung. Dafür eignet sich der ETB-Gesamtwert, der bereits für die Heuristiken ermittelt wurde. Dieser wird für den Spieler ausgerechnet, für den die Stellung zu bewerten ist. Da der ETB-Wert die zu erwartende Rundenanzahl angibt, ist der Nutzen für größer werdende ETB-Werte absteigend. Daher wird der ETB-Wert von einer maximalen Rundenzahl abgezogen, bevor er gewichtet in die Gesamtbewertung mit einfließt. Somit unterscheiden sich produktive von weniger produktiven Stellungen durch einen höheren Nutzenwert. Genauso können wir das gleiche Verfahren für die Expansionsfähigkeit eines Spielers verwenden, indem wir statt dem ETB-Gesamtwert die ETBExpandingOptions verwenden.

Die Position des Räubers ist ebenfalls in die Stellungsbewertung aufzunehmen. Hierfür werden Werte definiert, die eine Bewertung runterziehen, wenn der Räuber bei einem selber steht bzw. die Bewertung erhöhen, wenn der Räuber beim Gegner steht. Die Größe des Werts hängt zusätzlich von der Siegpunktanzahl der betroffenen Spieler und vom blockierten Rohstoff ab.

Neben einigen weiteren Kriterien werden auch die Rohstoff- und Entwicklungskarten auf der Hand mit einbezogen. Bei den Rohstoffen wird die Gesamtanzahl betrachtet und wie viele Kombinationen wie Holz-Lehm oder Getreide-Erz existieren. Dafür existiert ein Wichtigkeitsmaß für die jeweiligen Rohstoffe: Rohstoffe, die dringend benötigt werden, erhalten ein höheres Gewicht als Rohstoffe, von denen ein Spieler mehr als genug hat. Dieses Wichtigkeitsmaß wird für jede Runde neu ermittelt, da die benötigten Rohstoffe sich während des Spielverlaufs ändern können. Bei den Entwicklungskarten werden die bereits ausgespielten ignoriert, von den anderen wird eine Bewertung durchgeführt. Offene Siegpunktkarten werden aufgrund ihrer Wichtigkeit besonders positiv gewertet. Die Tabelle 10 fasst alle eingesetzten kleinen Bewertungsfunktionen zusammen.

| Bewertung | Erklärung |
|---|---|
| Anzahl der Siegpunkte | Ermittlung und Vergleich der privaten und öffentlichen Siegpunktanzahl |
| Rohstoffproduktivität | Bewertung der Produktivität über den ETB-Gesamtwert |
| Expansionsfähigkeit | Bewertung der Expansionsfähigkeit über den ETBExpandingOptions |
| Räuberposition | Bewertung ob Räuber bei einem selber oder beim Gegner steht, Siegpunktanzahl der betroffenen Spieler und blockierter Rohstoff |
| Anzahl Rohstoffe auf der Hand | Gesamtanzahl an Rohstoffen auf der Hand |
| Anzahl Holz-Lehm Kombinationen | Gesamtanzahl an Kombinationen von Holz-Lehm |
| Anzahl Getreide-Erz Kombinationen | Gesamtanzahl an Kombinationen von Getreide-Erz |
| Bewertung der Entwicklungskarten | Bewertung von nicht ausgespielten Entwicklungskarten auf der Hand |
| Besitz der größten Rittermacht | Bewertung wer die größte Rittermacht besitzt |
| Besitz der längsten Handelsstraße | Bewertung wer die längste Handelsstraße besitzt |

Tabelle 10: Verwendete kleine Bewertungsfunktionen, die gewichtet in die Gesamtbewertungsfunktion eingehen.

Ein offener Punkt ist eine dynamische Stellungsbewertung, die auf den Spielverlauf reagiert. So könnte aus einer Bewertungsdatenbank Informationen für verschiedene Spieleröffnungen einbezogen werden. Im späteren Spielverlauf würde eine veränderte Bewertung stattfinden, die sich mehr auf das Mittelspiel oder das Endspiel konzentriert. Diese Idee entstammt der Schachprogrammierung, die solch eine Stellungsklassifikation durchführt. In der aktuellen Version von Java Settlers ist so eine Klassifikation jedoch nicht integriert und bietet hier eine Möglichkeit für weitere Arbeiten.

# 6 Experimente und Ergebnisse

## 6.1 Zielsetzung der Arbeit

Diese Arbeit hat sich zum Ziel gesetzt, eine intelligente Brettspielsimulation mit intuitiver Eingabemöglichkeit zu erstellen. Dabei sollte die künstliche Intelligenz im Computerspieler so klug und flexibel sein, dass es Spaß macht gegen einen herausfordernden Gegner zu spielen. Gleichermaßen sollte es vom Schwierigkeitsgrad keinen Spieler überfordern. Die Bedienung dabei soll einfach und selbstverständlich sein, sodass Benutzer ohne komplizierte Erklärungen sofort mit dem Spiel interagieren können. Im Großen und Ganzen soll Java Settlers das Brettspiel „Die Siedler von Catan“ in einer neuartigen Art und Weise den Benutzern präsentieren.
Ob dies gelungen ist, soll durch Untersuchungen gemessen werden. Dafür wurde als Zielsetzung definiert, die Intuitivität und die Intelligenz von Java Settlers zu charakterisieren und zu bewerten. Einerseits soll hinterfragt werden, ob die eingesetzten Heuristiken und Bewertungskriterien denen entsprechen, die andere Entwickler oder erfahrene Siedlerspieler einsetzen würden und ob die erstellte KI anderen KIs überlegen sein würde. Andererseits soll ermittelt werden, wie ein Benutzer die Intuitivität und die Intelligenz des Systems ansieht.

## 6.2 Einsatz von Java Settlers in der Lehre

Um den ersten Punkt zu durchleuchten, wollten wir Antworten auf folgende Fragestellungen finden:

- Welche Heuristiken und Bewertungskriterien würden andere Entwickler einsetzen?
- Welche Heuristiken und Bewertungskriterien würden erfahrene Siedlerspieler einsetzen?
- Wie erfolgreich agiert unser Computerspieler gegen Computerspieler andere Entwickler?

Dafür wurde Java Settlers in der Lehre bei 40 Studenten der Freien Universität Berlin eingesetzt. In der Übung zur Vorlesung „Künstliche Intelligenz“ im Sommersemester 2008 wurde den Studenten zunächst beauftragt, sich zwei Wochen lang intensiv mit dem Brettspiel „Die Siedler von Catan“ zu beschäftigen, die Regeln zu erlernen und bereits Strategien zu erdenken. Danach wurde ihnen eine Version von Java Settlers übergeben, für die sie einen KI-Spieler implementieren sollten. Dafür mussten sie Heuristiken entwerfen und die Bewertungsfunktion erstellen, wofür sie erneut zwei Wochen Zeit hatten. Im Anschluss wurde mit Hilfe von JGameAI ein Turnier gestartet mit allen erschaffenen KI-Implementierungen, um die besten Computerspieler zu ermitteln.
Die drei besten Computerspieler wurden herangezogen, um mit der eingebauten KI von Java Settlers in einem Ligamodus anzutreten. In dieser Liga wurden insgesamt über 100 Spiele absolviert, wobei die KI von Java Settlers etwas mehr als die Hälfte für sich entscheiden konnte

(siehe Tabelle 11). Dies ist ein exzellenter Wert, der deutlich über dem erwartenden Wert von 25% liegt, wenn alle Spieler gleich stark wären.

| KI | Java Settlers KI | KI A | KI B | KI C | Gesamt |
|---|---|---|---|---|---|
| Gewonnen (absolut) | 71 | 18 | 26 | 21 | 136 |
| Gewonnen (prozentual) | 52,2% | 13,2% | 19,1% | 15,4% | 100% |
| Siegpunkte gesamt | 1095 | 770 | 923 | 813 | 3601 |
| Siegpunkte im Schnitt | 8.05 | 5.66 | 6.79 | 5.98 | 26.48 |

Tabelle 11: Liga-Ergebnisse zwischen Java Settlers KI und anderer KIs

Aus der Dokumentation und dem Quellcode der einzelnen Computerspieler wurden die eingesetzten Heuristiken und Bewertungskriterien notiert. Diese entsprachen in der Regel den gleichen Heuristiken und Kriterien, die in den Kapitel 5.3 und Kapitel 5.4 vorgestellt wurden. Einige neue Ideen, wie die Betrachtung verschiedene Rohstoffkombinationen wie Holz-Lehm oder Getreide-Erz, haben sich als effektiv herausgestellt und wurden nach der Untersuchung in die Java Settlers KI integriert.
Um die Kenntnisse der erfahrenen Siedlerspieler mit einzubringen, wurde im Forum von siedlen.de [69], der größten Fanseite von „Die Siedler von Catan“ mit dutzenden Turnieren und tausenden von Siedlerexperten, die Spieler befragt, welche Kriterien im Spiel am wichtigsten anzusehen sind. Die meisten Antworten bestätigen die Punkte, die im Kapitel 2.1.6 vorgestellt und in Java Settlers umgesetzt worden sind. Dies bestätigt, dass in Java Settlers bereits die wichtigsten Spielkriterien berücksichtigt worden sind, die auch von erfahrenen Siedlerspielern gewählt worden wären.

## 6.3 Ergebnisse bezüglich der Intuitivität der Bedienung

Die intuitive Benutzung eines Multi-Touch-Systems sollte in der Praxis evaluiert werden. Dafür wurde neben Java Settlers einige kleine Multi-Touch-Applikationen geschrieben: z.B. Air-Hockey, Ausrichten von Graphen zu planaren Graphen, Skalieren und Rotieren von Bildern, freies Bewegen einer 3D Welt. Diese wurden zur „Langen Nacht der Wissenschaft“ 2008 an der Freien Universität Berlin den Besuchern präsentiert [70]. Dahingehend wurde das existierende großflächige E-Chalk-System des Institut für Informatik erweitert mit Multi-Touch-Funktionalität [71]. Nach einer kurzen Einführung in die technischen Grundlagen von Multi-Touch wurden die Applikationen vorgeführt, danach durfte das Publikum die Applikationen selber ausprobieren. In einigen persönlichen Gesprächen wurden die Personen gebeten, ihre Eindrücke zur Interaktion zu beschreiben und zu bewerten.

Im Speziellen wurden viele Fragen zur Intuitivität der Benutzung gestellt:

- Wie intuitiv ist das System zu bedienen?
- Was gefällt Ihnen an der Bedienung und was nicht?
- Können Sie sich weitere solcher Systeme vorstellen?
- Können Sie sich weitere solcher Systeme in Ihrem Alltag vorstellen?
- Was für Möglichkeiten sehen Sie in solch einer Technik?
- Welche Vor- oder Nachteile sehen Sie gegenüber herkömmlichen Systemen mit Maus und Tastatur?

Der Großteil der Besucher fand das System überaus intuitiv zu bedienen. Ohne Erklärung welche Handbewegungen für welche Operationen notwendig sind, erkannten die Leute schnell, dass mit einem Finger Objekte verschoben werden, während mit zwei oder mehreren Fingern Objekte skaliert und rotiert werden. Besonders Kinder fanden sich im System sehr schnell zu recht, aber auch ältere Personen schienen das System schneller zu verstehen als bei einer Erklärung für herkömmliche Software.
Fast alle waren vom großflächigen Aufbau beeindruckt und können sich vorstellen, in Zukunft solche Systeme an publikumsreichen öffentlichen Plätzen wiederzufinden. Auch am Einsatz in den eigenen vier Wänden waren die meisten überzeugt, sofern es von den Kosten akzeptabel wäre. Bezüglich der Möglichkeiten, die diese Technik mit sich bringen könnte, fanden auf der einen Seite die jugendlichen Besucher die Gebrauchstauglichkeit (engl. Usability) mit mehreren Fingern und Händen gleichzeitig interessant und hatten viele Ideen für weitere effektvolle Unterhaltungssoftware. Das etwas ältere Publikum auf der anderen Seite sah im System ein Potential für eine einfache Erlernbarkeit, die Ihnen deutlich verständlicher ist als z.B. der Umgang mit Maus und Tastatur.
Dennoch ergaben sich auch Stimmen die das System kritisch beäugt und es noch nicht reif für den Massenmarkt empfunden haben. So warnten sie vor der riesigen Anzahl an Programmen, die bereits existieren und für Maus und Tastatur optimiert sind. Eine Umstellung würde Zeit und Geld benötigen sowie eine gute Programmierschnittstelle und die Nachfrage des Marktes. Einige Aufgaben, wie Tabellenkalkulation, seien selbst bei Verfügbarkeit von Multi-Touch einfacher mit Maus und Tastatur zu lösen.

## 6.4 Ergebnisse bezüglich der Intelligenz der Computerspieler

Schließlich sollte noch die Intelligenz bei Java Settlers selber untersucht werden. Für dieses Vorhaben wurde die fertige Version von Java Settlers einigen Versuchspersonen gegeben, die mehrere Partien gegen den eingebauten Computerspieler absolvieren sollten. Danach wurde gebeten eine Umfrage auszufüllen und die eigenen Erfahrungen mit der KI zu schildern. Die

Umfrage, die im Anhang A.4 komplett vorzufinden ist, sollte Rückschlüsse auf Fragen zulassen wie z.B.:

- Wie intelligent kam Ihnen der Computerspieler vor?
- Fanden Sie es eher spannend oder frustrierend gegen den Computerspieler zu spielen?
- Empfanden Sie den Computerspieler in der Zugfindung zu langsam?
- Hat der Computerspieler Ihnen geholfen, neue Strategien zu entwickeln?

Die Resultate von 26 Teilnehmern waren zufriedenstellend und sind ebenfalls im Anhang A.4 grafisch aufbereitet. Bei den Teilnehmern waren über 80% gut informiert über das Brettspiel „Die Siedler von Catan“, haben es bereits einige Male gespielt und schätzen sich in dem Gebiet als Fortgeschrittener oder Profi ein (Abbildung 33, 34, 35). Um eine genaue Einschätzung der KI von Java Settlers zu erhalten, musste jede/r Teilnehmer/in das Spiel 10 mal spielen - knapp die Hälfte spielte sogar öfters (Abbildung 36).
Die Qualität der KI soll einmal objektiv gemessen werden an der Anzahl der Spiele, die an den Computerspieler verloren gingen, andererseits auch subjektiv vom Empfinden der Testpersonen. Es wird für ein optimales Ergebnis erwartet, dass der menschliche Spieler 25% der Spiele gewinnt, da er stets alleine gegen drei Computerspieler antritt. Die Umfrageergebnisse in Abbildung 37 zeigen jedoch, dass die menschlichen Spieler deutlich mehr Spiele gewonnen haben. Dies deutet darauf hin, dass die KI noch weiteres Potential nach oben hat, jedoch bei einer 50%-Siegrate bereits recht wettbewerbsfähig ist. Auf einer imaginären Skala von 1 bis 10 sollten die Personen die Intelligenz des Systems beurteilen (1 = gar nicht intelligent, 10 = sehr intelligent). Lediglich eine Person schätzte die KI als wenig intelligent (1-3), während 5 Personen sie als sehr intelligent (8-10) sahen (Abbildung 38). Der Großteil stimmte aber für eine mittelgute Intelligenz (4-7), der Mittelwert lag hier bei 6,15. Auch hier zeigt sich, dass die KI nicht gänzlich als intelligent eingestuft wurde, aber im Großen und Ganzen eine akzeptable Intelligenz vorweisen konnte.
Für eine qualitative Untersuchung wurde mit Freitextfeldern erfragt, ob die KI wiederkehrende Strategien benutzt oder nicht nachvollziehbare Aktionen durchgeführt hat. Beides wurde in der Mehrheit verneint, es gab aber jedoch einige Bemerkungen, dass einige Straßenbauten unsinnig erschien. Tatsächlich scheint die KI in manchen Situationen willkürlich Straßen zu bauen. Dies kann damit erklärt werden, dass der Computerspieler entweder versucht seine längste Handelsstraße auszubauen oder zu besseren Feldern expandieren wollte (siehe ETBExpandingOptions). Die richtige Bewertung von Straßenbauzügen zählt ohnehin zu dem komplexesten Bewertungen im Spiel, die zu bewältigen sind. Andere Bemerkungen, dass die KI gezielt auf Sonderkarten wie die Längste Handelsstraße oder die Größte Rittermacht hin entwickelt, sind zwar als wiederkehrende Strategien zu verstehen, aber in der Regel auch notwendig für einen Sieg.

Nachdem bisher nur allgemein über die Intelligenz gefragt wurde, sollten in Prüffragen die KI auch aus anderen Blickwinkeln bewertet werden. So war herauszufinden, wie die gegnerische Intelligenz sich auf die Motivation des Spielers ausgewirkt hat. Eine „zu gute“ oder „zu schlechte“ KI kann für einen Spieler frustrierend bis langweilig wirken, eine herausfordernde KI erzeugt jedoch spannende und packende Spiele. Erneut auf einer Skala von 1 bis 10 sollte angegeben werden wie frustrierend (=1) oder spannend (=10) die Spiele waren. Im Ergebnis fanden mehr als die Hälfte der Teilnehmer das Spiel als eher spannend (>5), der Mittelwert lag hier bei 6,35 (Abbildung 39).
Eine gute KI zeichnet sich auch durch Flexibilität aus, die sich gegebenen Situationen anpassen kann. Die Testperson hat auf einer Skala von 1 bis 10 zu beurteilen, ob die KI eher stur (=1) oder eher kreativ (=10) agiert hat. Abbildung 40 zeigt die Ergebnisse zu dieser Frage, die recht gleichverteilt liegen. 15 Personen empfanden die KI als eher kreativ (>5), während 11 Personen sie als eher stur (<=5) bezeichnen.
Zuletzt wollten wir noch erfahren, ob die Suchdauer der Computerspieler angemessen war oder den Teilnehmern als zu langatmig galt. Auf die Frage „Empfanden Sie den Computerspieler in der Zugfindung zu langsam?“ antworteten 8 mit „ja, viel zu langsam“, 17 mit „nein, das war in Ordnung“ und eine Person mit „nein, er hätte ruhig mehr Zeit in Anspruch nehmen können“ (Abbildung 41). Somit ist die Suchdauer der KI für die meisten Teilnehmer akzeptabel, könnte aber nach Wunsch etwas geringer ausfallen. Auf die letzte Frage, ob der Computerspieler geholfen hat neue Strategien zu entwickeln, gab es leider keine aufschlussreichen Antworten.

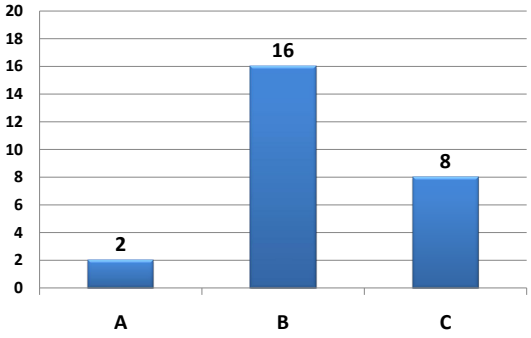



Abbildung 33: Grafische Auswertung zu Frage 1 der Umfrage: Wie lange sind Sie mit dem Brettspiel „Die Siedler von Catan“ vertraut? A = weniger als 1 Woche, B = bis zu 1 Jahr, C = länger als 1 Jahr

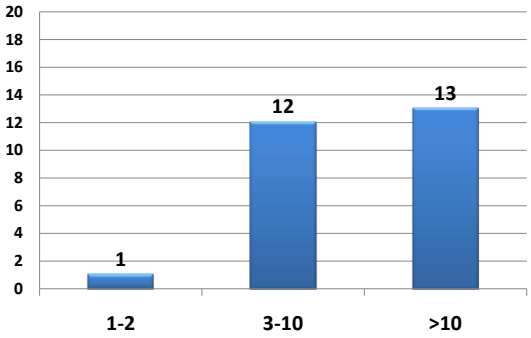



Abbildung 34: Grafische Auswertung zu Frage 2 der Umfrage: Wie oft haben Sie schätzungsweise das Brettspiel „Die Siedler von Catan“ gespielt?

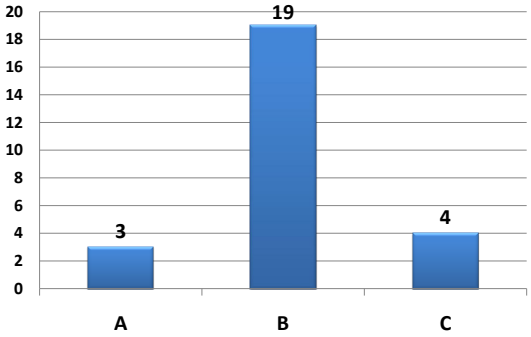



Abbildung 35: Grafische Auswertung zu Frage 3 der Umfrage: Wie stark würden Sie ihre Fähigkeiten im Brettspiel „Die Siedler von Catan“ einschätzen? A = Anfänger, B = Fortgeschrittener, C = Profi

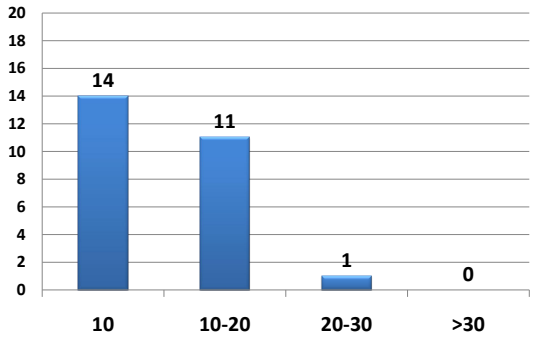



Abbildung 36: Grafische Auswertung zu Frage 4 der Umfrage: Wie viele Spiele haben Sie mit Java Settlers gespielt?

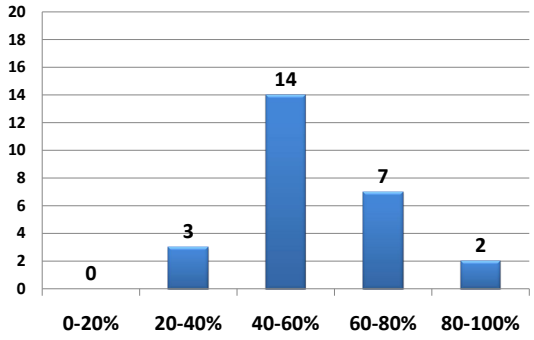



Abbildung 37: Grafische Auswertung zu Frage 5 der Umfrage: Wie viele Spiele haben Sie davon schätzungsweise gewonnen?

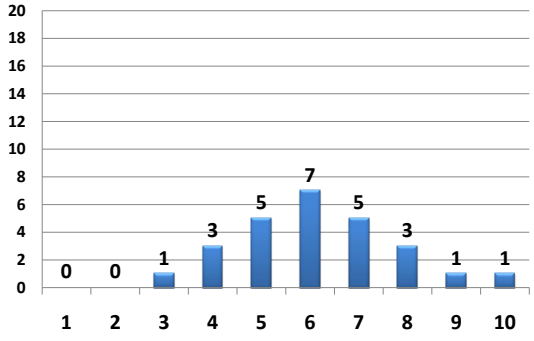



Abbildung 38: Grafische Auswertung zu Frage 6 der Umfrage: Wie intelligent kam Ihnen der Computerspieler auf einer Skala von 1 bis 10 vor? 1 = gar nicht intelligent, 10 = sehr intelligent

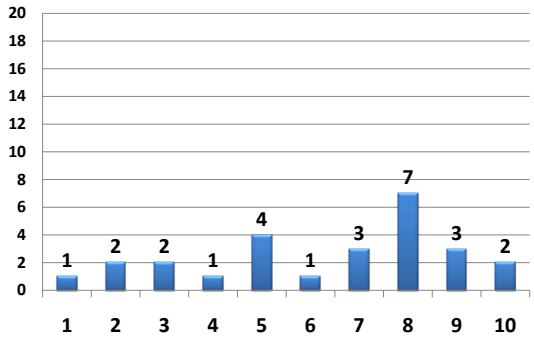



Abbildung 39: Grafische Auswertung zu Frage 9 der Umfrage: Fanden Sie es eher frustrierend oder spannend gegen den Computerspieler zu spielen? 1 = eher frustrierend, 10 = eher spannend

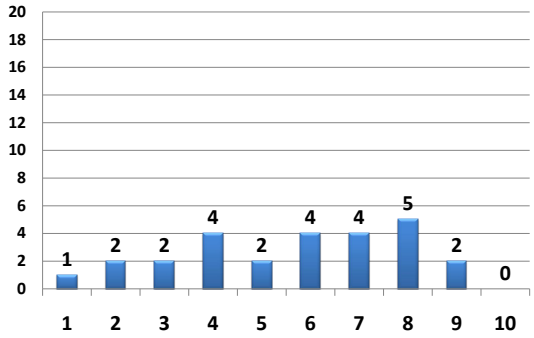



Abbildung 40: Grafische Auswertung zu Frage 10 der Umfrage: Empfanden Sie den Computerspieler in seinen Aktionen eher stur oder kreativ? 1 = eher stur, 10 = eher kreativ

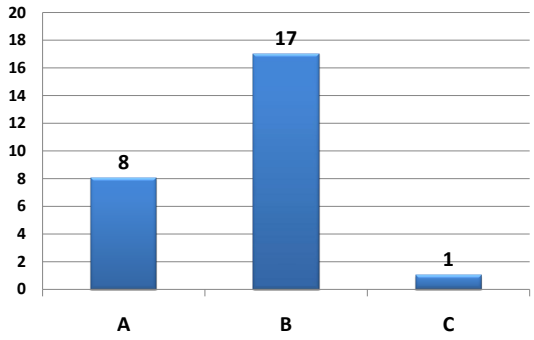



Abbildung 41: Grafische Auswertung zu Frage 11 der Umfrage: Empfanden Sie den Computerspieler in der Zugfindung zu langsam? A = ja, viel zu langsam, B = nein, das war in Ordnung, C = nein, er hätte ruhig mehr Zeit in Anspruch nehmen können

# 7 Fazit, Ausblick und zukünftige Arbeiten

## 7.1 Fazit

Das Projekt Java Settlers zeigt vielversprechende Ergebnisse bezüglich eines intelligenten, nutzenbasierten Agentensystems für eine Computerversion des Brettspiels „Siedler von Catan“ und einer intuitiven Ansteuerung über Multi-Touch. „Siedler von Catan“ hat sich als Forschungsbereich als sehr interessant erwiesen, da die Arbeitsumgebung markante Unterschiede zu traditionellen Spielen aufweist [19]. Der eingebaute Agent versteht die Regeln des Spiels und verkörpert die Anwendung genereller Spielstrategien. Dabei arbeitet das System auf einem Zustandsmodell und führt eine tiefenbeschränkte Suche durch. Zum Beschneiden des Suchbaumes werden verschiedene Heuristiken eingesetzt, die denen entsprechen, die erfahrene Spieler einsetzen würden. Die Bewertungsfunktion orientiert sich an den wichtigsten Kriterien des Spiels um einen Sieg zu erreichen. Aufgrund der kooperativen Natur des Spiels wird eine eigens entwickelte Entropie-Nutzenfunktion eingesetzt, die den absoluten Nutzenvektor in einen relativen überführt und somit verschiedene Nutzenvektoren vergleichbar macht.
Die Software wurde mit Hilfe des Frameworks jGameAI erstellt. Diesbezüglich demonstriert Java Settlers den einfachen Einsatz von jGameAI zur Erstellung neuer Spielimplementierungen. Dabei bestand die hauptsächliche Arbeit in der Modellierung eines Spielzustands und des Zuggenerators, sowie der Erstellung einer Benutzeroberfläche und einer geeigneten Bewertungsfunktion. Neue Entwicklungen, die bei anderen Spielen ebenfalls hilfreich sind, wurden in das Framework eingebaut.
Für die Steuerung des Systems wurde Multi-Touch als Interaktionsform gewählt. Die Interaktion über mehrere Finger ermöglicht ein natürliches und komfortableres Spielerlebnis. Der konzeptuelle Aufbau orientiert sich an einem DI-Ansatz, wo die Berührungspunkte durch infrarotes, reflektiertes Licht von den Fingern identifiziert werden können. Ein praktisches Experiment ergab positive, qualitative Ergebnisse in Bezug auf die Bedienung. Damit weist Multi-Touch Vorteile gegenüber herkömmlichen Interaktionsmethoden auf und empfiehlt sich für weitere Forschungsprojekte.
Java Settlers wurde als begleitende Aufgabe in einer Lehrveranstaltung eingesetzt und hat als Basis für die Entwicklung künstlicher Spieler gedient. Dies zeigt einen weiteren potentiellen zukünftigen Einsatz von diesem Projekt als Framework für Aufgaben oder Übungen in den Bereichen Spieltheorie, Spieleprogrammierung und künstlicher Intelligenz im Allgemeinen.

## 7.2 Ausblick

Die Spieltheorie ist ein relativ junger Wissenschaftszweig, um rationale Entscheidungsverhalten mathematisch herzuleiten. Anwendung findet sie heute in den Wirtschaftswissenschaften für

ökonomische Analysen und Unternehmensforschung, aber auch in Rechts- und Politikwissenschaft, Soziologie, Psychologie und Biologie [4]. In der Informatik wird sie häufig benutzt, um spielähnliche Probleme mit Hilfe von Suchstrategien und Heuristiken zu lösen [4]. Computerspiele weisen heutzutage eine sehr starke künstliche Intelligenz auf und zeigen einen gewaltigen Fortschritt im Vergleich zu zehn Jahren früher [1, 2, 3]. Es ist abzusehen, dass kommende Spiele sich in der Zukunft nicht nur grafisch immer näher dem Fotorealismus annähern, sondern auch immer mehr intelligentes natürliches Verhalten aufweisen.
Das Ziel wird es weiterhin bleiben, wettbewerbsfähige und herausfordernde bis hin zu perfekten Computerspieler zu entwerfen, die jederzeit immer die beste Aktion durchführen und bei der Suche zu dieser in einem akzeptablen Zeitrahmen bleiben. Die Entwicklung wird vereinfacht durch Frameworks wie JGameAI, die ein Bündel an allgemeinen Aufgaben für einen übernehmen. Durch zukünftige Arbeiten an diesem Framework, können Reinforcement Lernmethoden entwickelt werden, die die Intelligenz der künstlichen Spieler weiter anheben [39, 43].
Auch die Computer-Mensch-Interaktion wird in naher Zukunft einige Änderungen erfahren und sich deutlich vielseitiger präsentieren. Heutzutage greifen wir auf herkömmliche Eingabegeräte wie Maus, Tastatur, Trackball, Tablets und Joysticks zurück, die in Zukunft durch weitere erweitert werden. Bald könnten viele Systeme durch Head-Mounted Displays, Datenhandschuhe oder Geräte mit Bewegungssensoren gesteuert werden [5, 6]. Auch gänzlich ohne Eingabegeräte gibt es bereits Bestrebungen Anwendungen durch Sprachsteuerung oder gar Gehirnwellen zu steuern [72, 73]. Multi-Touch wird sicher eine zentrale Rolle in diesen Bestrebungen spielen, denn sie erlaubt intuitive komplexe Aktionen durch mehrere Berührungspunkte.
Dabei ist die Multi-Touch-Technik, die in dieser Arbeit vorgestellt wurde, ein Ausblick auf das, was in einigen Jahren möglich sein wird. Die heutzutage eingesetzten Techniken sind jedoch eher eine Übergangslösung und möglicherweise bald veraltet. Moderne flache Displays mit eingebauter nativer Unterstützung für Multi-Touch würden einen großräumigen Aufbau unwirtschaftlich machen [74, 75]. Zudem könnte eine einheitliche API für Multi-Touch-Eingaben helfen, bessere Anwendungen zu erstellen. Es ist denkbar in naher Zukunft sowohl großflächige Projektionen als auch mobile Displays mit Multi-Touch ausgestattet zu sehen, sei es an öffentlichen Orten [65] oder im Privaten [66].

## 7.3 Zukünftige Arbeiten

Die aktuelle Version von Java Settlers bietet eine Reihe an Punkten, wo nach nachgebessert werden kann. Wie in dieser Arbeit bereits erwähnt, fehlt die Fähigkeit des Computerspielers, aktiv kooperative Handel zu schließen. Dies würde zu mehr Handel zwischen Computer und Mensch als auch zwischen Computer untereinander führen. Auch die Heuristiken und Bewertungskriterien sind noch nicht perfekt und könnten mehr Feinschliff vertragen. Eine Weiterentwicklung hier könnte die Qualität der künstlichen Intelligenz weiter verbessern oder auch die Suchdauer

für einen neuen Zug verringern. Bei der Bewertung könnte noch zusätzlich differenziert werden zwischen Eröffnungs-, Mittelspiel- und Endspielstellungen.
Was noch wünschenswert gewesen wäre, aber aus Zeitgründen nicht realisierbar, wäre eine Anbindung an einen globalen Game-Server über das Internet. Die Idee ist es über eine Verbindung ins Internet mit anderen Spielern rund um den Globus spielen zu können. Eventuell könnte eine globale Rangliste der besten Spieler geführt werden, die von Zeit zu Zeit kleine Turniere veranstalten. Solch eine Arbeit würde die Implementierung des Game-Servers beanspruchen sowie eine Verbindung der bestehenden Java Settlers Version mit dem Server.
Eine ganze Menge an neuen Arbeiten sind denkbar rund um das Thema Multi-Touch. Der Multi-Touch-Tisch bietet eine Plattform für weitere Spiele sowie primitive oder komplexe Multi-Touch-Anwendungen. Für den Softwareentwurf oder für großflächige Planungen bietet sich ein Tisch in etwas größerer Form an, an dem mehrere Personen gleichzeitig ihre Ideen visualisieren können. Das kollaborative Entwerfen auf einer interaktiven Oberfläche erhöht die Kommunikation und den Ideenaustausch. Ein Einsatz ist in Firmen oder Unternehmen, aber auch im akademischen Umfeld vorstellbar.
Im Kapitel 3.2 wurde bereits die Idee eines intelligenten Hauses erwähnt. In solch einer Umgebung können verschiedenartige Multi-Touch-Geräte unterschiedliche Funktionen übernehmen. Ein kleines mobiles Gerät oder ein in der Wand eingebautes Display dient der Steuerung und Kontrolle wichtiger Ereignisse, wie Temperatur, Strom, Licht, Gas etc. Zusätzlich können hier andere Geräte ein- bzw. ausgeschaltet, das Home Cinema System gesteuert oder das Hausüberwachungssystem kontrolliert werden. Weitere Displays im Wohnzimmer, Küche oder Bad könnten spezifische Funktionen wie Unterhaltung, Information oder Warnungen übernehmen.
Ebenfalls erwähnt wurde der Einsatz in der Lehre für großflächige Projektionen. An einer Multi-Touch-Wand, ähnlich dem E-Chalk Projekt an der Freien Universität Berlin [71], könnte Dozenten interaktive Vorlesungen präsentieren und komplexe Sachverhalte intuitiv veranschaulichen. Dafür muss eine geeignete Präsentationsapplikation entworfen werden, die sich über Fingerberührungen bedienen lässt. Etwas mobiler und einfacher hat es ein Dozent, wenn die Multi-Touch-Funktionalität in ein mobiles Rednerpult umgesetzt wird, das zu verschiedenen Orten transportiert werden kann. Dabei wird das Bild über ein Bildsignal nach außen geführt, damit es an einen externen Projektor angeschlossen werden kann.
Dies soll nur einige Beispiele von möglichen neuen Arbeiten im Bereich Multi-Touch vorstellen. Durch die breite Vielseitigkeit sind sicherlich noch etliche weitere spannende Projekte möglich. Die Zukunft wird zeigen, was aus dieser Technik möglich ist und was möglich gemacht wird.

# A Anhang

## A.1 Multi-Touch Schemazeichnungen

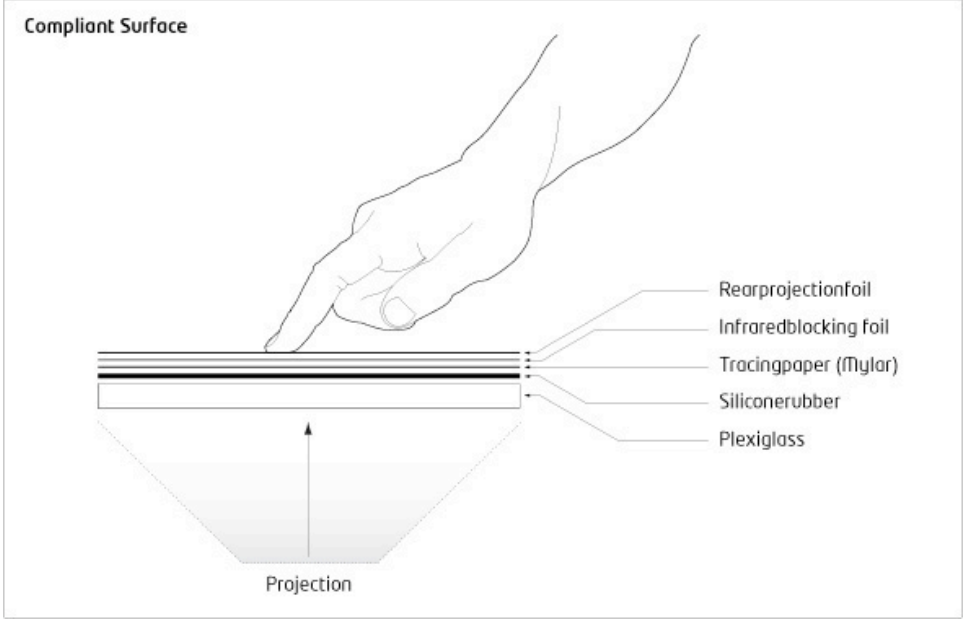



Abbildung 42: Nachgiebige Oberfläche mit Silikon für FTIR. Abbildung aus [76]

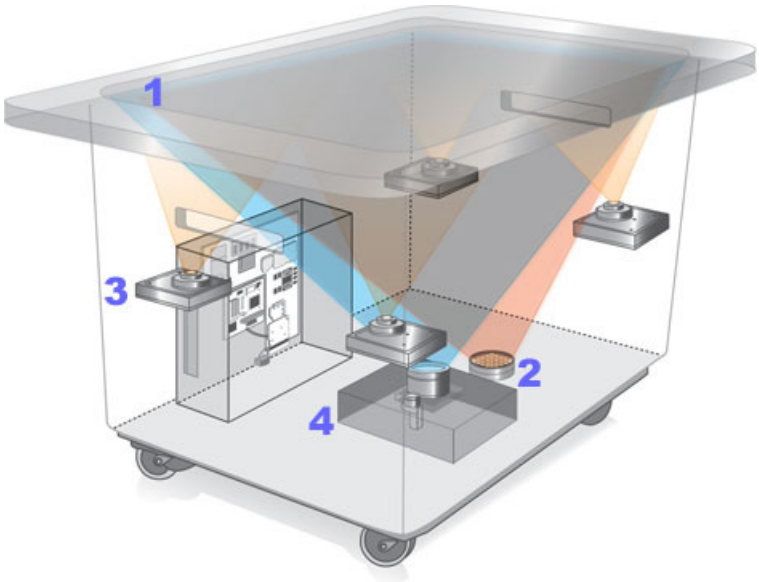



Abbildung 43: Aufbau von Microsoft Surface (Abbildung aus [77]):
1 Glasfläche, 2 Infrarot-Scheinwerfer, 3 Infrarot-Kameras, 4 Projektor

## A.2 Weitere Screenshots

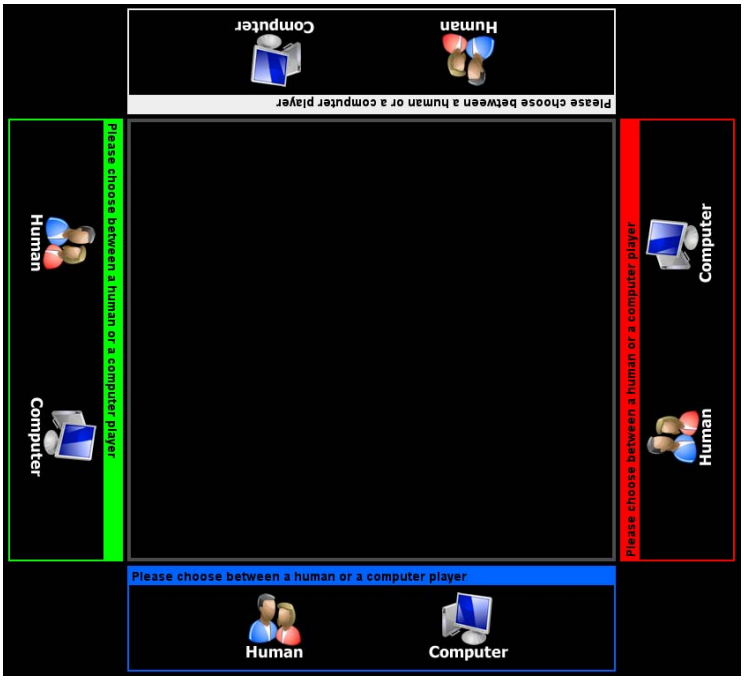



Abbildung 44: Screenshot von Java Settlers: Auswahl der Spieler

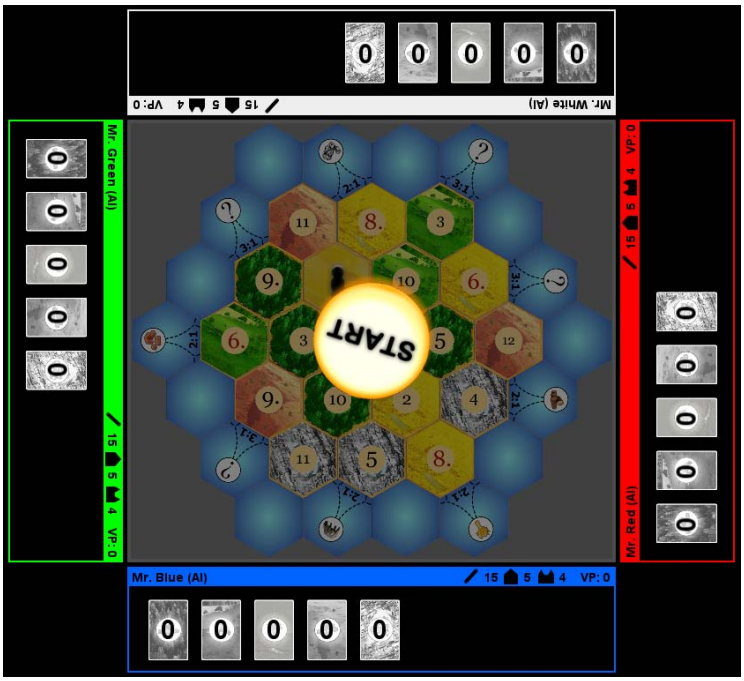



Abbildung 45: Screenshot von Java Settlers: Vor dem Spielstart

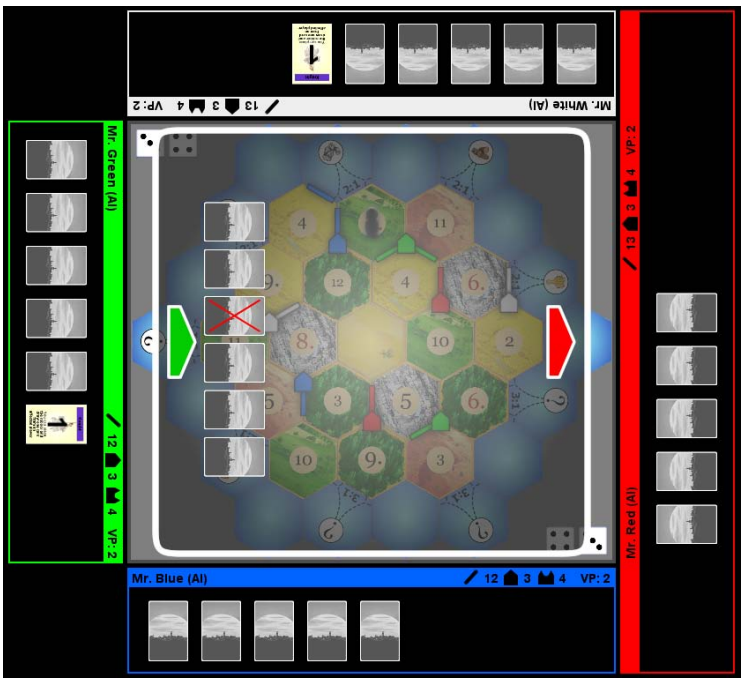



Abbildung 46: Screenshot von Java Settlers: Ziehen einer Rohstoffkarte (bei verdeckten Rohstoffen)

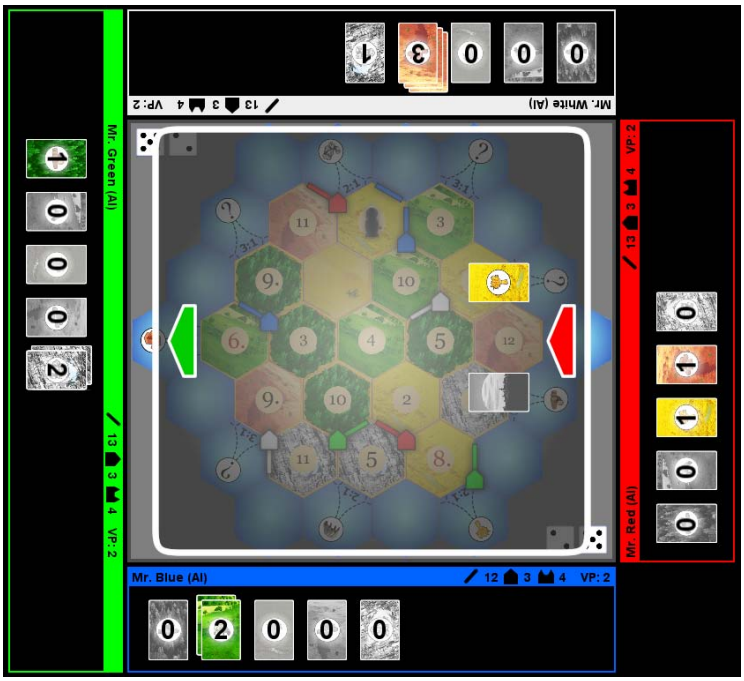



Abbildung 47: Screenshot von Java Settlers: Ziehen einer Rohstoffkarte (bei sichtbaren Rohstoffen)


Abbildung 48: Screenshot von Java Settlers: Hauptmenü

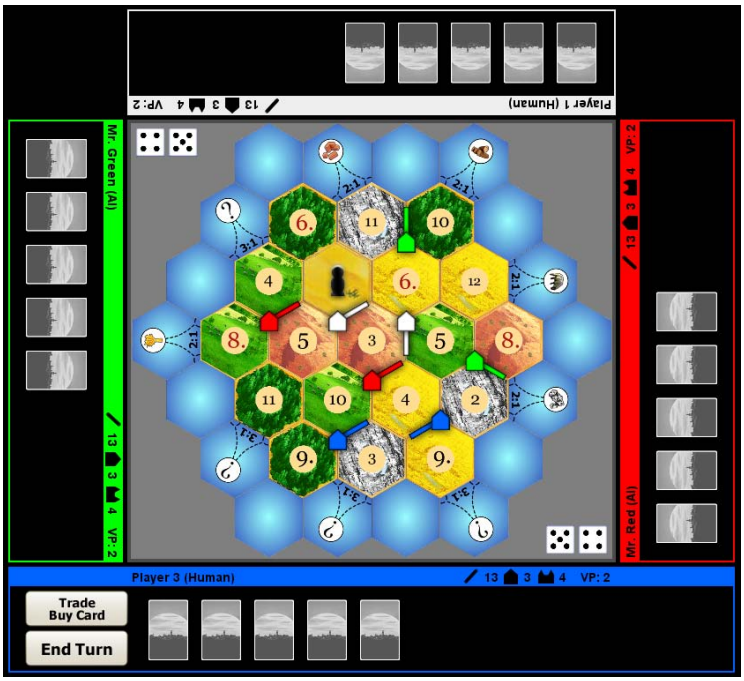



Abbildung 49: Screenshot von Java Settlers: Spielermenü


Abbildung 50: Screenshot von Java Settlers: Spielende

## A.3 Quellcodefragmente

Der folgende Quellcode ist ein Ausschnitt aus dem Quellcode von Java Settlers. Sie zeigt die Berechnung der längsten Handelsstraße für einen Spieler:

```
1  /**
2   * calculate the longest road length for a player
3   * @param player the player
4   * @return the length of the longest road of player
5   */
6  private int calcLongestRoad(int player) {
7      int longest = 0;
8
9      //potential start nodes for dfs
10     Set<Node> roadNodes = new HashSet<Node>();
11
12     synchronized(edgeGraph) {
13         //fill in potential start nodes
14         for(Edge e : edgeGraph) {
15             if(e.belongsTo(player)) {
16                 roadNodes.add(e.getNode1());
17                 roadNodes.add(e.getNode2());
18             }
19         }
20
21         //map of visited edges
22         edgeVisited = new HashMap<Integer, Integer>();
23
24         //recursive dfs
25         for(Node n : roadNodes) {
26             edgeVisited.clear();
27             dfs(n, player, 1);
28             for(int i : edgeVisited.values()) {
29                 if(i > longest) longest = i;
30             }
31         }
32     }
33     return longest;
34 }
35
36 /**
37  * depth first search to retrieve longest road length
38  * @param n the current node
39  * @param player the player for whom to check
40  * @param curLength the current length of the longest road
41  */
42 private void dfs(Node n, int player, int curLength) {
43     for(Edge e : n.getEdges()) {
44         if(e != null && e.belongsTo(player) && !edgeVisited.containsKey(e.getID())) {
45             //edge hasn't been visited yet
46             if(e.getNode1() == n) {
47                 edgeVisited.put(e.getID(), curLength);
48                 if(e.getNode2().belongsTo(player) || e.getNode2().isFree())
49                     dfs(e.getNode2(), player, curLength+1);
50             } else if(e.getNode2() == n) {
51                 edgeVisited.put(e.getID(), curLength);
```

```
52                  if(e.getNode1().belongsTo(player) || e.getNode1().isFree())
53                      dfs(e.getNode1(), player, curLength+1);
54              }
55          }
56      }
57  }
```

Quellcodefragment 1: Berechnung der längsten Handelsstraße für jeden Spieler

Der folgende Quellcodeausschnitt aus Java Settlers zeigt die iterative Berechnung des ETB-Gesamtwerts:

```
1   private float evalETBAll(JavaSettlersBoard jsBoard, int forPlayer) {
2       float value = 0;
3
4       List<Node> buildingNodes = new ArrayList<Node>();
5
6       //fill in all building nodes
7       synchronized(jsBoard.getNodeGraph()) {
8           for(Node n : jsBoard.getNodeGraph()) {
9               if(n.belongsTo(forPlayer)) {
10                  buildingNodes.add(n);
11              }
12          }
13      }
14
15      //calculate frequency table
16      int[] frequency = calcFrequency(buildingNodes, forPlayer);
17
18      //subtract each ETB-value from maximal number of rolls and add to total value
19      value += ETB_MAXROLLS - evalETBRoad(jsBoard, forPlayer, frequency);
20      value += ETB_MAXROLLS - evalETBSettlement(jsBoard, forPlayer, frequency);
21      value += ETB_MAXROLLS - evalETBCity(jsBoard, forPlayer, frequency);
22      value += ETB_MAXROLLS - evalETBDevCard(jsBoard, forPlayer, frequency);
23
24      return value;
25  }
26
27  private int[] calcFrequency(Collection<Node> buildingNodes, int forPlayer) {
28      float[] resourceProbs = new float[5];
29
30      //calculate resource probabilities
31      for(Node n : buildingNodes) {
32          for(Hex h : n.getHexes()) {
33              if(h != null && h.isLand() && !h.isDesert()) {
34                  if(n.isSettlement()) resourceProbs[h.getResource()-RESOURCE_BASE] += bo[
                        currentClassIndex][BO_BUILDED] * tokenProbs[h.getToken()];
35                  else if(n.isCity()) resourceProbs[h.getResource()-RESOURCE_BASE] += 2*bo[
                        currentClassIndex][BO_BUILDED]*tokenProbs[h.getToken()];
36                  else if(n.canBuildSettlement(forPlayer)) resourceProbs[h.getResource()-
                        RESOURCE_BASE] += bo[currentClassIndex][BO_CAN_BUILD]*tokenProbs[h.
                        getToken()];
37                  else resourceProbs[h.getResource()-RESOURCE_BASE] += bo[currentClassIndex][
                        BO_BUILD_WITH_ROAD]*tokenProbs[h.getToken()];
38              }
39          }
```

```
40      }
41
42      //convert to frequency
43      int[] frequency = new int[5];
44      for(int i = 0; i < resourceProbs.length; i++) {
45          if(resourceProbs[i] == 0) frequency[i] = -1;
46          else frequency[i] = Math.round(1.0f/resourceProbs[i]);
47      }
48
49      return frequency;
50  }
51
52  private int evalETBRoad(JavaSettlersBoard jsBoard, int playerId, int[] frequency) {
53      //WOOD - BRICK
54      int[] neededResources = new int[] {-1, 0, 0, -1, 0};
55      return evalETB(jsBoard, playerId, frequency, neededResources);
56  }
57
58  private int evalETBSettlement(JavaSettlersBoard jsBoard, int playerId, int[] frequency) {
59      //WOOD - WOOL - WHEAT - BRICK
60      int[] neededResources = new int[] {-1, -1, -1, -1, 0};
61      return evalETB(jsBoard, playerId, frequency, neededResources);
62  }
63
64  private int evalETBCity(JavaSettlersBoard jsBoard, int playerId, int[] frequency) {
65      //WHEAT - WHEAT - ORE - ORE - ORE
66      int[] neededResources = new int[] {0, 0, -2, 0, -3};
67      return evalETB(jsBoard, playerId, frequency, neededResources);
68  }
69
70  private int evalETBDevCard(JavaSettlersBoard jsBoard, int playerId, int[] frequency) {
71      //WOOL - WHEAT - ORE
72      int[] neededResources = new int[] {0, -1, -1, 0, -1};
73      return evalETB(jsBoard, playerId, frequency, neededResources);
74  }
75
76  private int evalETB(JavaSettlersBoard jsBoard, int playerId, int[] frequency, int[]
        neededResources) {
77      int rolls = 0;
78
79      //build resource table iteratively
80      while(!Util.allPositive(neededResources) && rolls < ETB_MAXROLLS) {
81          rolls++;
82          for(int i=0; i < neededResources.length; i++) {
83              if(frequency[i] != -1 && rolls % frequency[i] == 0) {
84                  neededResources[i]++;
85              }
86          }
87
88          //check most needed resource to trade
89          int mostNeeded = Util.minIndex(neededResources);
90
91          //check if we have 2:1 port
92          if(mostNeeded != -1) {
93              for(int i=0; i < neededResources.length; i++) {
94                  if(jsBoard.hasPortTrade(playerId, i+PORT_BASE) && neededResources[i] >= 2) {
95                      neededResources[i] -= 2;
```

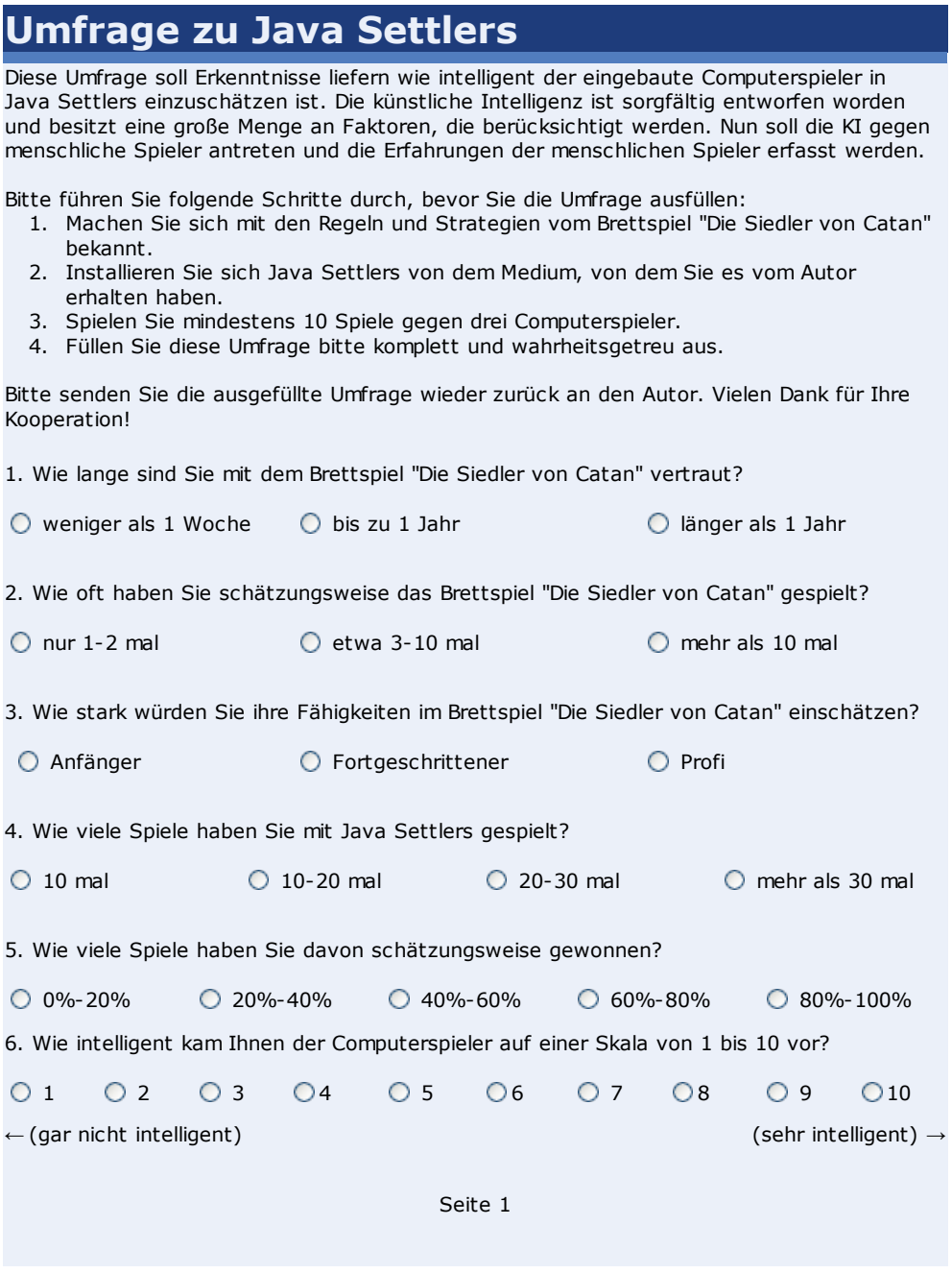
```
96                      neededResources[mostNeeded] += 1;
97                      mostNeeded = Util.minIndex(neededResources);
98                  }
99              }
100         }
101
102         //check if we have 3:1 port
103         if(mostNeeded != -1) {
104             for(int i=0; i < neededResources.length; i++) {
105                 if(jsBoard.hasPortTrade(playerId, PORT_NORMAL) && neededResources[i] >= 3) {
106                     neededResources[i] -= 3;
107                     neededResources[mostNeeded] += 1;
108                     mostNeeded = Util.minIndex(neededResources);
109                 }
110             }
111         }
112
113         //else take 4:1 trade
114         if(mostNeeded != -1) {
115             for(int i=0; i < neededResources.length; i++) {
116                 if(neededResources[i] >= 4) {
117                     neededResources[i] -= 4;
118                     neededResources[mostNeeded] += 1;
119                     mostNeeded = Util.minIndex(neededResources);
120                 }
121             }
122         }
123     }
124
125     return rolls;
126 }
```

Quellcodefragment 2: Berechnung des ETB-Gesamtwertes

## A.4 Java Settlers Umfrage

### A.4.1 Umfragebogen

Folgender Umfragebogen wurde für eine Umfrage eingesetzt, um quantitative und qualitative Daten bezüglich der Intelligenz des Systems zu ermitteln. Die Resultate dieser Umfrage werden in Kapitel 6.4 vorgestellt.



**Umfrage zu Java Settlers**

Diese Umfrage soll Erkenntnisse liefern wie intelligent der eingebaute Computerspieler in Java Settlers einzuschätzen ist. Die künstliche Intelligenz ist sorgfältig entworfen worden und besitzt eine große Menge an Faktoren, die berücksichtigt werden. Nun soll die KI gegen menschliche Spieler antreten und die Erfahrungen der menschlichen Spieler erfasst werden.

Bitte führen Sie folgende Schritte durch, bevor Sie die Umfrage ausfüllen:

1. Machen Sie sich mit den Regeln und Strategien vom Brettspiel "Die Siedler von Catan" bekannt.
2. Installieren Sie sich Java Settlers von dem Medium, von dem Sie es vom Autor erhalten haben.
3. Spielen Sie mindestens 10 Spiele gegen drei Computerspieler.
4. Füllen Sie diese Umfrage bitte komplett und wahrheitsgetreu aus.

Bitte senden Sie die ausgefüllte Umfrage wieder zurück an den Autor. Vielen Dank für Ihre Kooperation!

1. Wie lange sind Sie mit dem Brettspiel "Die Siedler von Catan" vertraut?

○ weniger als 1 Woche ○ bis zu 1 Jahr ○ länger als 1 Jahr

2. Wie oft haben Sie schätzungsweise das Brettspiel "Die Siedler von Catan" gespielt?

○ nur 1-2 mal ○ etwa 3-10 mal ○ mehr als 10 mal

3. Wie stark würden Sie ihre Fähigkeiten im Brettspiel "Die Siedler von Catan" einschätzen?

○ Anfänger ○ Fortgeschrittener ○ Profi

4. Wie viele Spiele haben Sie mit Java Settlers gespielt?

○ 10 mal ○ 10-20 mal ○ 20-30 mal ○ mehr als 30 mal

5. Wie viele Spiele haben Sie davon schätzungsweise gewonnen?

○ 0%-20% ○ 20%-40% ○ 40%-60% ○ 60%-80% ○ 80%-100%

6. Wie intelligent kam Ihnen der Computerspieler auf einer Skala von 1 bis 10 vor?

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10

← (gar nicht intelligent) (sehr intelligent) →

Seite 1

Abbildung 51: Seite 1 des Umfragebogens

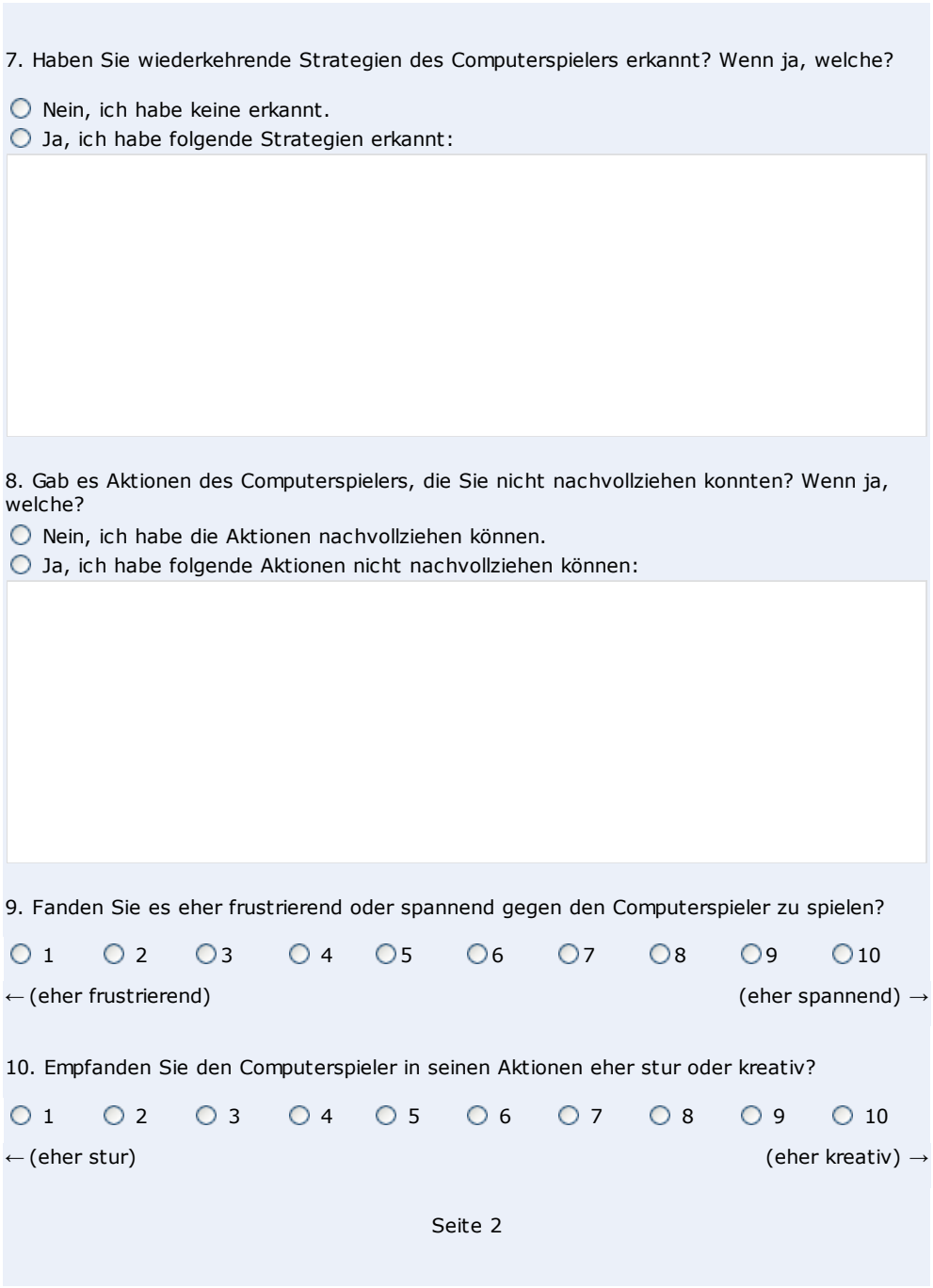
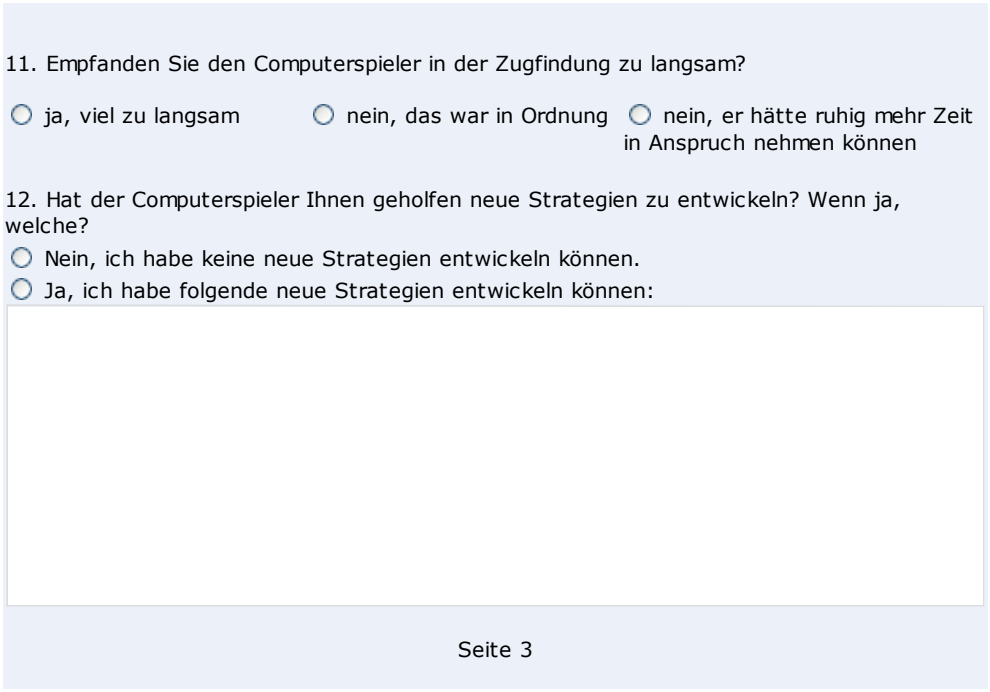



Abbildung 52: Seite 2 des Umfragebogens



11. Empfanden Sie den Computerspieler in der Zugfindung zu langsam?

○ ja, viel zu langsam ○ nein, das war in Ordnung ○ nein, er hätte ruhig mehr Zeit in Anspruch nehmen können

12. Hat der Computerspieler Ihnen geholfen neue Strategien zu entwickeln? Wenn ja, welche?

○ Nein, ich habe keine neue Strategien entwickeln können.
○ Ja, ich habe folgende neue Strategien entwickeln können:

Seite 3

Abbildung 53: Seite 3 des Umfragebogens

# Literaturverzeichnis


[1] C. Donninger and U. Lorenz, *The Chess Monster Hydra*, vol. 3203/2004. Springer Berlin / Heidelberg, 2004. ISBN-13: 978-3-54022-989-6.

[2] J. Schaeffer, *One Jump Ahead: Challenging Human Supremacy in Checkers*. Springer-Verlag, 1997. ISBN-13: 978-0-38794-930-7.

[3] G. Tesauro, "TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play," *Neural Computation*, vol. 6, no. 2, pp. 215–219, 1994. ISSN: 0899-7667.

[4] S. J. Russell and P. Norvig, *Künstliche Intelligenz: Ein moderner Ansatz*. Pearson Studium, 2004. ISBN-13: 978-3-82737-089-1.

[5] A. Sears and J. A. Jacko, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. CRC, 2007. ISBN-13: 978-0-80585-870-9.

[6] M. Dahm, *Grundlagen der Mensch-Computer-Interaktion*. Pearson Studium, 2005. ISBN-13: 978-3-82737-175-1.

[7] B. A. Myers, "A Brief History of Human Computer Interaction Technology," *ACM interactions*, vol. 5, no. 2, pp. 44–54, 1998. ISSN: 1072-5520.

[8] "B. Buxton, Multi-Touch Systems that I Have Known and Loved." Webseite, 2008. `http://www.billbuxton.com/multitouch0verview.html`.

[9] M. Bader, "Eine allgemeine selbstlernende Strategie für nicht-kooperative Spiele," *Diplomarbeit, Freie Universität Berlin*, 2008.

[10] "jGameAI." Webseite, 2008. `http://sourceforge.net/projects/jgameai/`.

[11] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, 1984. ISBN-13: 978-0-20110-172-0.

[12] "JavaSettlers." Webseite, 2008. `http://www.javasettlers.de/`.

[13] "Siedler von Catan." Webseite, 2008. `http://www.catan.com/`.

[14] "Spiel des Jahres e.V. - Ausgezeichnete Spiele 1995." Webseite, 2008. `http://www.spiel-des-jahres.com/`.

[15] A. Gordon, "A general algorithm for tic-tac-toe board evaluation," *Journal of Computing Sciences in Colleges*, vol. 21, pp. 42–46, 2006. ISSN: 1937-4771.

[16] G. Tesauro, *Comparison training of chess evaluation functions, Machines that learn to play games*. Nova Science Publishers, Inc., Commack, NY, 2001. ISBN-10: 1-59033-021-8.

[17] T. Huang, “The game of go: an ideal environment for capstone and undergraduate research projects,” *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, vol. 35, no. 1, pp. 84–88, 2003. ISSN: 0097-8418.

[18] J. Y. Han, “Multi-touch interaction research,” *International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2006 Computer animation festival*, no. 244, 2006. ISBN-10: 1-59593-364-6.

[19] M. Wang, M. Bader, M. Block, and R. Rojas, “Intuitive Spielsysteme für Multi-Touch-Umgebungen,” *4th Microsoft Academic Days 2008*, 2008.

[20] M. P. Quine and E. Seneta, “Bortkiewicz’s data and the law of small numbers,” *International statistical review*, vol. 55, no. 2, pp. 173–181, 1987. ISSN: 0306-7734.

[21] “Catan Online.” Webseite, 2008.
`http://www.catanonline.com/`.

[22] “MSN Games: Catan - The Computer Game.” Webseite, 2008.
`http://zone.msn.com/en/root/deluxe.htm?code=110252700`.

[23] “MSN Games: Catan Online.” Webseite, 2008.
`http://zone.msn.com/en/catanonline/`.

[24] R. S. Thomas and K. Hammond, “Java settlers: a research environment for studying multi-agent negotiation,” *Proceedings of the 7th international conference on Intelligent user interfaces*, p. 240, 2002. ISBN-10: 1-58113-459-2.

[25] R. S. Thomas, *Real-time Decision Making for Adversarial Environments using a Plan-based Heuristic*. PhD thesis, Northwestern University, 2003.

[26] “SourceForge.” Webseite, 2008.
`http://www.sourceforge.net`.

[27] “GL Catan.” Webseite, 2008.
`http://sourceforge.net/projects/glcatan/`.

[28] M. Wooldridge and N. R. Jennings, “Intelligent agents: Theory and practice,” *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[29] C. Shannon, “Programming a Computer for Playing Chess,” *Philosophical Magazine*, vol. 41, p. 314, 1950.

[30] V. Allis, *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, University of Limburg, Maastricht, The Netherlands, 1994.

[31] J. von Neumann, “Zur Theorie der Gesellschaftsspiele,” *Mathematische Annalen*, vol. 100, no. 1, pp. 295–320, 1928.

[32] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton University Press, 2007. ISBN-13: 978-0-69113-061-3.

[33] J. Nash, “Equilibrium Points in N-Person Games,” *Proceedings of the National Academy of Sciences*, vol. 36, pp. 48–49, 1950.

[34] J. Nash, "Non-Cooperative Games," *The Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[35] N. R. Sturtevant, *A Comparison of Algorithms for Multi-player Games*, vol. 0302/2003. Springer Berlin / Heidelberg, 2003. ISBN-13: 978-3-540-20545-6.

[36] C. Luckhardt and K. Irani, "An algorithmic solution of N-person games," *Fifth National Conference of the American Association for Artificial Intelligence (AAAI-86)*, vol. 1986, pp. 158–162, 1986.

[37] N. R. Sturtevant and R. E. Korf, "On Pruning Techniques for Multi-Player Games," *Sixteenth National Conference of the American Association for Artificial Intelligence (AAAI-00)*, pp. 201–207, 2000.

[38] D. E. Knuth and R. W. Moore, "An Analysis of Alpha-Beta Pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[39] M. Block, M. Bader, E. Tapia, M.Ramírez, K. Gunnarsson, E.Cuevas, D.Zaldivar, and R. Rojas, "Using Reinforcement Learning in Chess Engines," *Concibe Science 2008, Guadalajara/Mexico*, 2008.

[40] E. A. Heinz, *Scalable Search in Computer Chess*. PhD thesis, Universität Karlsruhe, 2000.

[41] D. Steinwender and F. Friedel, *Schach am PC*. Markt und Technik, 1995. ISBN-10: 3-87791-522-1.

[42] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[43] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007. ISBN-13: 978-0-38731-073-2.

[44] G. F. Luger, *Künstliche Intelligenz . Strategien zur Lösung komplexer Probleme*. Pearson Studium, 2001. ISBN-13: 978-3-82737-002-0.

[45] R. Rojas, *Neural Networks: A Systematic Introduction*. Springer, 1996. ISBN-13: 978-3-54060-505-8.

[46] R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986. ISSN: 0885-6125.

[47] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[48] A. J. Lockett, C. L. Chen, and R. Miikkulainen, "Evolving explicit opponent models in game playing," *Proceedings of the 9th annual conference on genetic and evolutionary computation*, pp. 2106–2113, 2007. ISBN-13: 978-1-59593-697-4.

[49] "Freie Universität Berlin, AI-Game Programming Group." Webseite, 2008. `http://page.mi.fu-berlin.de/block/gameinggroup/startseite.html`.

[50] J. Y. Han, "Multi-touch interaction wall," *International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2006 Emerging technologies*, vol. 25, 2006. ISBN-10: 1-59593-364-6.

[51] "Microsoft Surface Website." Webseite, 2008. http://www.microsoft.com/surface/.

[52] J. Y. Han, "Low-cost multi-touch sensing through frustrated total internal reflection," *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pp. 115–118, 2005. ISBN-10: 1-59593-271-2.

[53] "T. Roth, Interaction Design Projects at Zurich University of the Arts (Zhdk)." Webseite, 2008. http://iad.projects.zhdk.ch/multitouch/.

[54] "Homepage of Jeff Han." Webseite, 2008. http://cs.nyu.edu/~jhan/.

[55] S. Boring, O. Hilliges, and A. Butz, "A Wall-Sized Focus Plus Context Display," *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications, PERCOM*, pp. 161–170, 2007. ISBN-10: 0-7695-2787-6.

[56] "Apple iPhone." Webseite, 2008. http://www.apple.com/iphone/.

[57] J. Rekimoto, "SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces," *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 113–120, 2002. ISBN-10: 1-58113-453-3.

[58] P. Dietz and D. Leigh, "DiamondTouch: a multi-user touch technology," *Proceedings of the 14th annual ACM symposium on User interface software and technology, Symposium on User Interface Software and Technology*, pp. 219–226, 2001. ISBN-10: 1-58113-438-X.

[59] "Techno Babble: Inside Multi-Touch." Webseite, 2008. http://www.kf12.com/blogs/techno/2007/10/inside-multi-touch/.

[60] J. A. Paradiso, "Tracking contact and free gesture across large interactive surfaces," *Communications of the ACM*, vol. 46, no. 7, pp. 62–69, 2003. ISSN: 0001-0782.

[61] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner, "The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces," *Proceedings of the first international conference on Tangible and Embedded Interaction (TEI07)*, pp. 139–146, 2007. ISBN-13: 978-1-59593-619-6.

[62] "dyeSight $2 Multi-Touch Pad." Webseite, 2008. http://blog.medallia.com/2007/06/dyesight.html.

[63] O. Oda, L. J. Lister, S. White, and S. Feiner, "Developing an augmented reality racing game," *Proceedings of the 2nd international conference on INtelligent TEchnologies for interactive enterTAINment*, no. 2, 2008. ISBN-13: 978-9-63979-913-4.

[64] T. L. Andersen, S. Kristensen, B. W. Nielsen, and K. Grønbæk, "Designing an augmented reality board game with children: the battleboard 3D experience," *Proceedings of the 2004 conference on Interactive Design And Children*, pp. 137–138, 2004. ISBN-10: 1-58113-791-5.

[65] P. Peltonen, E. Kurvinen, A. Salovaara, G. Jacucci, T. Ilmonen, J. Evans, A. Oulasvirta, and P. Saarikko, "It's Mine, Don't Touch!: interactions at a large multi-touch display in a city centre," *Proceeding of the 26th annual SIGCHI Conference on Human Factors in Computing Systems*, pp. 1285–1294, 2008. ISBN-13: 978-1-60558-011-1.

[66] A. Greenfield, *Everyware: The Dawning Age of Ubiquitous Computing*. New Riders Publishing, 2006. ISBN-13: 978-0-32138-401-0.

[67] "WarGames (Movie)." Film, 1983.
http://www.imdb.com/title/tt0086567/.

[68] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. University of Illinois Press, 1963. ISBN-13: 978-0-25272-548-7.

[69] "Siedeln.de - Die Seite für Siedler von Catan." Webseite, 2008.
http://www.siedeln.de/.

[70] "Lange Nacht der Wissenschaft." Webseite, 2008.
http://www.langenachtderwissenschaften.de/.

[71] "Freie Universität Berlin, E-Chalk." Webseite, 2008.
http://www.echalk.de.

[72] R. Cole, J. Mariani, H. Uszkoreit, G. B. Varile, A. Zaenen, and A. Zampolli, *Survey of the State of the Art in Human Language Technology (Studies in Natural Language Processing)*. Cambridge University Press, 1998. ISBN-13: 978-0-52159-277-2.

[73] J. R. Wolpaw, "Brain-computer interfaces (BCIs) for communication and control," *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, pp. 1–2, 2007. ISBN-13: 978-1-59593-573-1.

[74] "N-Trig." Webseite, 2008.
http://www.n-trig.com/.

[75] S. Hodges, S. Izadi, A. Butler, A. Rrustemi, and B. Buxton, "ThinSight: versatile multi-touch sensing for thin form-factor displays," *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pp. 259–268, 2007. ISBN-13: 978-1-59593-679-2.

[76] "T. Roth, 180 - A multitouch application for consulting situations." Webseite, 2008.
http://timroth.de/180/.

[77] "Techno Babble: Microsoft Surface." Webseite, 2008.
http://www.kf12.com/blogs/techno/2007/05/microsoft-surface/.

Sofern nicht anders angegeben, waren alle URLs gültig am 14. November 2008.

# Abbildungsverzeichnis

# Tabellenverzeichnis



# Quellcodeverzeichnis