Projektdokumentation

im

Software-Praktikum 2002

(Dozent: Dr. Ulrich Kortenkamp)

Umsetzung des Brettspiels Torres in Java

Tutoriumsgruppe: SWP2

Semester: Sommersemester 2002

Die Hängenden Gärten von Semiramis –

Umsetzung des Brettspiels Torres in Java

Software-Praktikum 2002

Revision 2.29

Impressum

Printed in Berlin. Documentation (Layout, Review, Editing) by Christian Dürrhauer, Berlin.

1. Inhaltsverzeichnis

EINLEITUNG

2. Einleitung

2.1 Softwarepraktikum

Im Sommersemester 2002 veranstaltete Dr. Ulrich Kortenkamp vom Fachbereich Informatik der Freien Universität Berlin ein Softwarepraktikum. Das Softwarepraktikum ist eine Pflichtveranstaltung für Studierende der Informatik sowie Studierende anderer Fachrichtungen. Insgesamt beteiligten sich mehrere Dutzend Studierende in von 7 verschiedenen Tutoren betreuten Projektgruppen daran, das bekannte Brettspiel Torres in jeweils eine auf Computern lauffähige Version umzusetzen.

Aufgrund seiner weiten Verbreitung sowie seiner Plattformunabhängigkeit sollte Java die Sprache sein, in der die Umsetzung geschrieben werden sollte.

Die genaue Aufgabenstellung bestand darin, für eine von den Tutoren geschriebene Server-Applikation, die das Spiel grundsätzlich verwaltet, eine gemäß der bekanntgegebenen Server-API funktionierende Client-Applikation zu schreiben, die das Brettspiel umsetzt und einen Computerspieler anbietet, der selbstständig – automatisiert – über das Netzwerk an einem Turnier teilnehmen kann.

Leitidee der Veranstaltung war, zwischen den Projektgruppen Wettbewerb zu induzieren. Dies wurde erreicht, indem Turniere Jeder gegen Jeden ausgetragen wurden, deren Ergebnisse zu einer Liga führten. Die Punkte wurden nach dem bewährten Schema der Formel-1 vergeben. Der Ligaerste sollte dann ausgezeichnet werden.

Zum gegenwärtigen Stand - während diese Dokumentation geschrieben wird - ist die Tutoriumsgruppe SWP2 die führende Mannschaft. Nur theoretisch kann noch Gleichstand hergestellt werden.

2.2 Unser Tutorium

Unser Tutorium wurde von Herrn Dr. Ulrich Kortenkamp betreut. Die Teilnehmer des Tutoriums waren (in alphabetischer Reihenfolge): Jakob Bilger, Evelyn Dittmer, Tobias Dobberphul, Christian Dürrhauer, Marc Georgi, Carsten Gräser, Martin Hense, Patrik Marschalik, Juergen Schütz, Daria Schymura, Bogdan Catalin-Stanca, Bernhard Weber, Andrea Wiese und Jenny Wöß.

Jeden Mittwoch zwischen 14 und 18 Uhr – sehr oft auch außerhalb dieser Regelhaftigkeit und bis weit in den nächsten Morgen hinein – trafen sich die meisten der Gruppe, teils alternierend, meist im festen Kern, um Bugs zu beheben, neue Funktionen zu implementieren, wichtige Meilensteine einzuhalten oder auch, um einfach nur Spaß miteinander zu haben.

Die festen Termine mittwochs wurden in den ersten Wochen als Gesprächs- und Besprechungstermine zum Aufstellen der Planungen genutzt (siehe Protokolle), zusätzlich gesellten sich anfangs auch noch die Dienstage für Präsentationen von unbedingt für die Entwicklung notwendigen Tools und IDEs sowie Projektmanagementstrategien dazu.

Der Tutor steuerte in der Anfangsphase bewußt. Zunächst war bis auf zwei Ausnahmen fast allen Tutanden Projektarbeit – und im Besonderen Softwareentwicklung – nicht vertraut. Dies bedeutete eine große Gefahr für das Scheitern des angestrebten Ziels. Durch die bewußte Intensivierung von Arbeit am Anfang wurden Erfolgserlebnisse geschaffen, eine Gemeinschaft aufgebaut, die Studierenden motiviert und die Grundlagen für die erfolgreiche Weiterarbeit geschaffen. Indem z.B. Debugging nicht wie aus den Basisvorlesung Informatik A und B bekannt per `System.out.println` etwas unsystematisch vorgenommen wurde, sondern Einarbeitungszeit für das komfortable Debugging mit log4j und das Erstellen von Testroutinen für Junit in Kauf genommen wurde, konnten spätere Probleme relativ schnell lokalisiert werden.

Es ist erwähnenswert, daß sich die Tutoriumsgruppe SWP2 fast ausschließlich aus Studierenden anderer Fachbereiche (Mathematik, Deutsche Philologie, Geographie) zusammensetzte und angehende Informatiker die Minderheit bildeten. Obwohl der Leiter der Veranstaltung, Herr Dr. Kortenkamp, unsere Gruppe betreute, wäre es ein Vorurteil, zu sagen, daß er uns mehr als andere Tutoren besonders geholfen hätte, und würde den Erfolg der Gruppe unpassend verringert er-

scheinen. Natürlich ist auch seine Leistung nicht zu schmälern: Es ist sein Verdienst, in kritischen Phasen steuernd eingegriffen und stets – auch nachts – mit Rat und Tat zur Seite gestanden zu haben.

Als ich mir am Anfang des Semesters die Aufgabenstellung verinnerlichte, hielt ich die Bewältigung nicht für möglich. Im Nachhinein bin ich deshalb freudig überrascht, was für ein schönes, zuverlässiges Produkt wir – die Gruppe SWP2 – innerhalb von nur knapp 3 Monaten auf die Beine gestellt haben.

Herrn Dr. Ulrich Kortenkamp sei an dieser Stelle für die Betreuung und Hilfestellung recht herzlich gedankt.

Dem geneigten Leser oder interessierten Spieler wünschen wir nun viel Spaß beim Lesen oder Spielen von oder gar gegen Semiramis und bedanken uns für die uns entgegengebrachte Aufmerksamkeit.

Berlin, Sonntag, 28. Juli 2002

i.A. Christian Dürrhauer

SPIELANLEITUNG

3. Spielanleitung

3.1.1. Systemvoraussetzungen

Sie benötigen (alternativ) eine/einen

- Windows®-PC mit Windows® 95 oder höher (98, Me, NT 4.0SP3, 2000,XP),
- SUN® Workstation,
- Apple® PowerPC® ab System 8, oder
- Linux oder FreeBSD.

Dies alles sind Plattformen, für die das absolut notwendige Java Runtime 1.3 erhältlich ist. Dieses muß zum Spielen von Semiramis installiert sein.
Für eine annehmbare Spielgeschwindigkeit sollte Ihr intel®-PC mindestens über 350MHz und 32MB RAM (NT und höher: 64MB RAM) verfügen. Andere Plattformen stellen eventuell andere Anforderungen.
Sie benötigen einen Netzwerkstack über TCP/IP, der heutzutage in allen Betriebssystemen vorhanden ist. Gggfls. müssen Sie ihn nachinstallieren, falls Sie über keinen Netzwerkanschluß verfügen, nötigenfalls über einen Loopback-Adapter.

3.1.2. Zum Lesen der Dokumentation

Semiramis ist eine Anwendung, die in JAVA geschrieben ist. Leider sind deshalb zum Starten des Programms auf einer Kommandozeile einige Texteingaben erforderlich, danach ist Semiramis allerdings benutzerfreundlich mit der Maus bedienbar. Haben Sie deshalb keine Angst!
Um die nötigen Schritte zu beschreiben, die erforderlich sind, Semiramis installieren (d.h. auf einem Computer nutzen zu können) und starten zu können, ist diese Dokumentation in Hinsicht auf Benutzereingaben stark formalisiert, d.h. sie sehen genau, was Sie erwartet und was Sie tun sollen. Sie sollten also vor keine Probleme gestellt werden.

3.1.2.1.Begriffsdefinition: Kommandozeile

Mit einer Kommandozeile können Sie dem Computer per reinem Text Befehle erteilen. Sicherlich kennen Sie die Möglichkeit, dem Computer per sogenanntem „Mausklick“ zu etwas zu veranlassen, so z.B. wenn Sie auf einem MS® Windows®-PC die Textverarbeitung MS® Word® starten möchten oder auch auf anderen Systemen, die eine grafische Benutzeroberfläche (GUI) haben (wie Apple®, SUN® oder Linux).

Zunächst mag diese Möglichkeit der Kommunikation umständlich oder gar archaisch wirken, dennoch hat sie ihre Existenzberechtigung, vor allem, wenn es um die auf hohe Geschwindigkeit ausgelegte Darstellung von Informationen geht oder um die automatische Verarbeitung vieler Daten oder Befehle. Es gibt gar Betriebssysteme, die zunächst nur diese Form der Befehlseingabe ermöglichen.
Die Kommandozeile wird auf verschiedenen Systemen anders genannt. Da JAVA weitestgehend plattformunabhängig ist, stellt sich daher die Problematik, für die für die Ausführung von JAVA-Bytecode bereitstehenden Systeme die Eingabemöglichkeiten zu benennen:

- auf Windows®-PCs: „Eingabeaufforderung“
- auf SUN®-Workstations, Linux-PCs oder Apple® MacOS® X: „Terminal“
- weiterhin gebräuchliche Namen: Kommandozeile, Konsole, DOS-Fenster oder ähnliches

Wenn Sie im Kontext dieser Bedienungsanleitung also mit **Eingabeaufforderung** aufgefordert werden, eine Eingabeaufforderung zu öffnen, so benutzen Sie bitte die entsprechende Variante Ihres Betriebssystems.

3.1.2.2.allgemeine Benutzerinteraktion

Aufforderungen an Sie, den Benutzer, werden in dieser Dokumentation grundsätzlich in einer anderen Schriftart und zwar in rot und fett geschrieben. Macht das Geschriebene sprachlich keinen großen Sinn für Sie, ist davon auszugehen, daß Sie es eintippen müssen (und zwar so wie angeben, inkl. Groß- und Kleinschreibung sowie Zeichen und Leerzeichen).
Beispiel:

Lesen Sie jetzt bitte weiter!

3.1.2.3.spezielle Tasten

Eine spezielle Taste, die Sie immer wieder benötigen werden, ist „Return“. Die Taste wird durch Return dargestellt. Grundsätzlich bedeuten eng eingekästete Zeichenfolgen, daß Sie die entsprechende Taste auf Ihrer Tastatur drücken müssen.
Beispiel:
F1 öffnet die Hilfe zu dem Programm, indem Sie dies hier gerade lesen.

3.1.2.4.Verweise

An einigen Stellen werden Verweise auftreten. Diese erscheinen grundsätzlich schräggedruckt.
Beispiel: Für interessante Ausführungen zur Projektarbeit lesen Sie bitte *Kapitel 4.1.1*.

Für weiterführende Informationen, z.B. Literatur, beachten Sie bitte die Fußnoten[1]. Diese finden sich am Ende der jeweiligen Seite.

3.1.2.5.Bildschirmanzeigen

Die Anzeigen, die auf Ihrem Bildschirm stehen (stehen sollten), werden grundsätzlich normal in einer anderen Schriftart in einem abgetrennten, weit abgekästeten Absatz dargestellt.

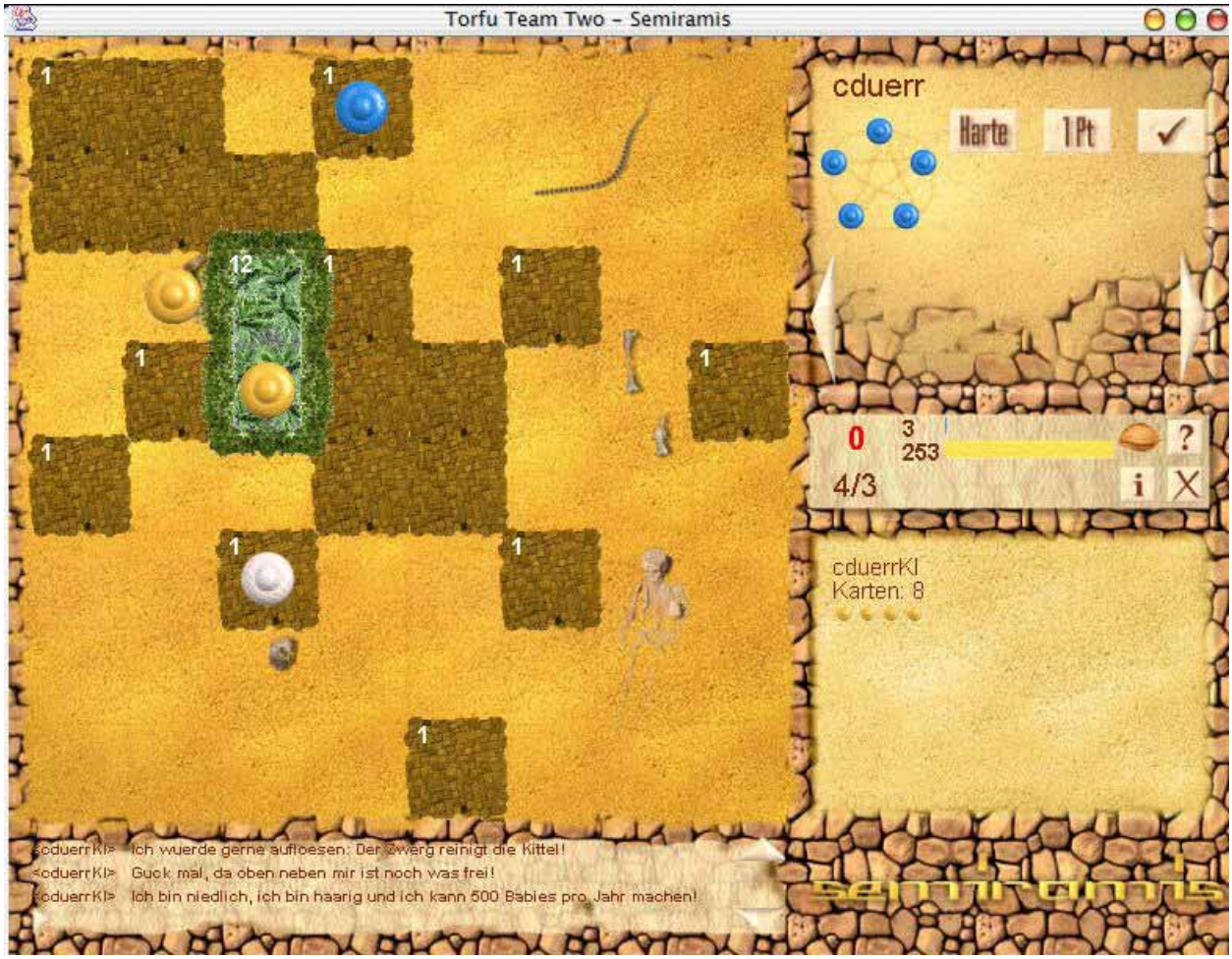
Beispiel:

```
C:\>_
```

Sollten graphische Aktionen näher erläutert werden, so wird die entsprechende Graphik im Text dargestellt und mit näheren visuellen Hinweisen versehen.

[1] Fußnoten sehen so aus und befinden sich am Ende der jeweiligen Seite.

Beispiel:



3.1.2.6.Für Fortgeschrittene: Programm-Code-Dokumentation

Sollten Sie Interesse am Lesen von in dieser Dokumentation gezeigtem Programm-Code haben, dann finden Sie ihn in folgender Form in dem Kapitel zur Entwicklung (*Kapitel 4*):

Beispiel:

```
void dialogEingabe_keyPressed(KeyEvent e) {
    if (dialogEingabe.getText().equals(new String(""))) return;
    if (e.getKeyCode() == 10) {
        String spielerName = GUI.myName;
        String nachricht = dialogEingabe.getText();
        dialogEingabe.setText("");
        ChatEreignis ce = new ChatEreignis(GUI.myId, nachricht);
```

```
        try{
          ra.fuehreAktionAus(ce);
        }catch (Exception ex){
            JOptionPane.showInternalMessageDialog(this.getParent(),
                new String(ex.getMessage()) ,
                "EXCEPTION", JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
```

Wir hoffen, mit dieser konsistenten Systematik zum Verständnis des ohnehin sehr einfach gehaltenen Oberfläche des Programms beizutragen und wünschen viel Spaß!

3.2 Das Spiel

3.2.1. Hintergrundgeschichte

584 v. Chr. buhlen die Söhne Nabopolassars [der älteste war Nebukadnezar] im reichen Babylonien um die Gunst der schönen persischen Prinzessin Semiramis.

Der alte König entscheidet nun einen fairen Wettstreit unter den Anwärtern auf die Verlobung, indem er sie mit der Errichtung einer großartigen Gartenanlage als Verlobungsgeschenk für die Prinzessin betraut. Wer es vermag, im Laufe von drei Jahren [Phasen] die eindrucksvollsten [höchsten und größten] Gärten zu bauen und ihre schönsten Stellen für die Prinzessin zu besetzen vermag, soll ihre Hand erhalten.

Sie schlüpfen in die Rolle eines dieser Anwärter. Sie möchten die Hand der wunderschönen und begehrenswerten Prinzessin Semiramis erobern!

3.2.2. Ziel des Spiels

Der Spielplan zeigt Babylon als ein Gebiet von 8 x 8 Feldern, auf denen im Laufe des Spiels die Gärten entstehen. Am äußeren Rand des Spielplans zeigt die Wertungsleiste den aktuellen Spielstand an sowie verschieden Informationen, die Ihnen zur Beurteilung der Spielsituation weiterhelfen.
Die Spieler erhalten bei jeder der drei Wertungen Erfolgspunkte für die Gärten, auf denen sie mit ihren Prinzen vertreten sind. Dort wird jeweils die Grundfläche der Burg mit der Ebene, auf der der Prinz steht, multipliziert. Natürlich kann ein Prinz weitere Hilfe von anderen Mitstreitern erhalten, die er selbst bestimmt und die ihn in anderen Gärten als er selbst vertreten. Wer Plätze in der Gartenanlage bereithält, in der sich die Prinzessin zur Zeit am liebsten aufzuhalten geruht, rückt in ihrer Gunst zusätzlich nach oben!

Am Ende jeden Jahres wird der König durch das Land reisen, um den Erfolg der Prinzen zu begutachten. Nach der dritten Reise wird er seine Entscheidung fällen. Gewinnen wird dann derjenige Spieler, dessen Prinz die meisten Punkte auf der Wertungsleiste auf sich vereinigen kann.

3.2.3. Regeln

a) Vorbereitung/Spielbeginn

- Beginnend mit dem jüngsten Mitspieler stellt reihum jeder Spieler **einen** seiner verliebten Prinzen auf einen beliebigen, freien Gartenbauplatz.
- Der Spieler, der als Letzter seinen Prinzen plaziert hat, setzt die Prinzessin auf einen **freien** Garten und macht ihn damit zum Lieblingsgarten der Prinzessin Semiramis.

Was Sie noch wissen sollten:

Der Begriff der **"Ebenen"** ist wichtig für die Ermittlung der Punkte bei den Wertungen. Mit "Ebene" bezeichnet man die Höhe, in der sich ein Prinz befindet. Ein Prinz, der direkt auf dem Spielplan steht, befindet sich auf Ebene 0. Ein Prinz, der auf einem Baustein steht, befindet sich auf Ebene 1, ein Prinz auf zwei Bausteinen auf Ebene 2 usw.

b) Spielverlauf

"Semiramis" läuft in drei Phasen ab. Jede Phase endet mit einer Wertung. Die erste Phase besteht aus vier Runden, die zweite und dritte Phase jeweils aus drei Runden (außer beim Spiel zu zweit, dort sind es immer vier Runden). Eine Runde ist beendet, wenn jeder Spieler einmal an der Reihe war.

c) Verteilung von Bausteinen

Vor jeder Phase erhalten die Spieler vom allgemeinen Vorrat Bausteine, die Terrassen eines Gartens darstellen. Die Anzahl der ausgeteilten Bausteine unterscheidet sich nach der Anzahl der Kontrahenten. Jeder Spieler bekommt die Bausteine in einem 5er-Türmchen vor sich auf den Tisch gestellt (rechts oben auf der Wertungsleiste). Dies sind die eigenen Vorratstürme für diese Phase.

Startspieler

Der jüngste Spieler beginnt. Nach ihm bestimmt der Computer automatisch, wer folgt. Diese Reihenfolge wird bis zum Spielende eingehalten..

d) Spielzug

Wer an der Reihe ist, hat 5 Aktionspunkte (AP) zur Verfügung, die er in beliebiger Reihenfolge auf die folgenden Aktionen aufteilen kann.

Prinz einsetzen (= 2 Aktionspunkte pro Prinz)

Ein neuer Prinz muß beim Einsetzen immer direkt neben einem eigenen, bereits eingesetzten Prinzen auf ein freies Feld gestellt werden. Dieses freie Feld, auf das ein Prinz eingesetzt wird,

- muß horizontal oder vertikal zum vorhandenen Prinz liegen,
- muß sich auf gleicher Ebene oder einer beliebig tieferen Ebene befinden, nicht aber auf einer höheren.

Jeder Prinz hat also maximal vier Einsetzfelder um sich herum.

Ziehen eines Prinzen (=1 Aktionspunkt pro Feld)

Sie können in Ihrem Zug beliebig viele Prinzen bewegen. Ein Prinz kann (außer, wenn eine Aktionskarte anderes erlaubt)

- nur waagerecht oder senkrecht gezogen werden, nie diagonal,
- pro Feld maximal eine Ebene nach oben steigen. Nach unten kann er beliebig viele Ebenen überwinden,
- nur auf oder über freie Felder bzw. Bausteine ziehen. Eigene und fremde Prinz sowie die Prinzessin sind Hindernisse, die nicht überwunden werden können.

Die Bausteine haben an allen vier Seiten Tore, wie Arkaden. **Prinzen können durch diese Tore gehen!** Dies bedeutet: Ein Prinz kann in ein Tor hineingehen und aus einem beliebigen Tor dieses Gartens wieder herauskommen. Dies kostet nur einen Aktionspunkt. Der Prinz muß dabei entweder auf derselben Ebene bleiben oder beliebig viele Ebenen herabsteigen (außer, wenn eine Aktionskarte anderes erlaubt). In einem Zug kann ein Prinz für entsprechend viele AP durch beliebig viele Gärten nacheinander hindurchgehen.

Bausteine einsetzen (=1 Aktionspunkt pro Baustein)

Die Bausteine, die vor Ihnen stehen, geben an, wie viele Runden noch bis zur nächsten Wertung gespielt werden. In Ihrem Spielzug können Sie nur die Bausteine **des** vor Ihnen stehenden Turmes verbauen, bis dieser abgebaut ist. Sie können in einer Runde nicht zwei verschiedene Bausteintürmchen abbauen anfangen. Welchen Baustein Sie wählen, entscheiden Sie selbst. Sollten Sie darüber hinaus noch Bausteine übrig haben, werden diese zum allgemeinen Vorrat zurückge-

legt. Bausteine, die Sie nach einer Wertung noch übrig haben, werden ebenfalls zurück zum allgemeinen Vorrat gelegt.
Beim Einsetzen von Bausteinen gelten folgende Regeln (außer, wenn auf Aktionskarten anderes zugelassen ist):

- Sie können in Ihrem Zug an mehreren, beliebigen Gärten anbauen.
- Sie müssen immer an einem bestehenden Garten anbauen. Entweder auf einer bereits vorhandenen, freien Terrasse oder auf Ebene 0 an einen Baustein (die Grundfläche des Gartens wird größer).
- Wer in die Höhe baut, muß dabei beachten, daß ein Garten nie höher sein darf als ihre Grundfläche. Wenn die Grundfläche z. B. aus 3 Bausteinen besteht, darf dieser Garten maximal 3 Ebenen haben.
- Wer die Grundfläche eines Garten erweitert, darf dabei keine zwei Gärten miteinander verbinden. Gärten dürfen sich nur diagonal berühren.

Aktionskarten kaufen (=1 Aktionspunkt pro Karte)
Pro Zug kann ein Spieler maximal zwei Karten kaufen. Die gekauften Karten dürfen **nicht im selben Zug** eingesetzt werden. Sie können sie aber bis zum Ende des Spiels verwenden.

Aktionskarte einsetzen (=0 Aktionspunkte)
Pro Zug können Sie nur eine Aktionskarte einsetzen - sie darf aber nicht im selben Zug gekauft worden sein. Sie legen die Karte zu einem beliebigen Zeitpunkt Ihres Zuges ab und führen die Aktion kostenlos aus, die auf der Karte angegeben ist. Die Karte geht aus dem Spiel. Am Ende der Anleitung werden die einzelnen Karten ausführlich erklärt.

Ziehen auf der Wertungsleiste (=1 Aktionspunkt pro Feld)
Haben Sie von Ihren 5 AP noch Punkte übrig und möchten diese nicht anderweitig verwenden, so dürfen Sie Ihren Wertungsstein auch außerhalb einer Wertung vorrücken. Sie erhöhen Ihre Wertungspunkte pro AP um 1. Es kann immer nur ein Spieler die gleiche Anzahl an Wertungspunkten haben. Wenn Sie die gleichen Wertungspunkte wie ein anderer Spieler haben würden, werden Ihre Wertungspunkte automatisch soweit erhöht, bis Sie einen einzigartigen Punktestand haben.

e) Die Wertungen

Jeweils am Ende einer Phase kommt es zu einer Wertung. Sie erhalten nur Punkte für die Gärten, auf denen Sie mit mindestens einem Prinz vertreten sind. Jeden Garten können Sie nur einmal werten, auch wenn dort mehrere Prinzen von Ihnen stehen, wenn also z. B. zwei Ihrer Prinzen auf einem Garten stehen, erhält nur der höher stehende Prinz Erfolgspunkte. Der andere geht leer aus.

Pro Garten erhalten Sie Erfolgspunkte nach folgender Formel: Ebene, auf der Ihr Prinz steht, multipliziert mit der Grundfläche des Gartens.

Beispiel: Ihr Prinz steht auf Ebene 3. Der Garten hat eine Grundfläche von 5 Bausteinen. Sie erhalten für diesen Garten 3 x 5 = 15 Punkte und können Ihren Wertungsstein auf der Wertungsleiste 15 Felder vorziehen.

Wenn Prinzen verschiedener Farben auf einem Garten stehen, erhält **jeder** Spieler, der dort mit einem Prinz vertreten ist, die entsprechenden Punkte, die sich für seinen Prinz errechnen.

Es wird automatisch gewertet, d. h. der Startspieler als Erstes, dann in der bekannten Reihenfolge, in der auch die Spielzüge vorgenommen werden. Es gilt die Regel, daß die Wertungspunkte in ihrem Wert einzigartig sein müssen (siehe *Ziehen auf der Wertungsleiste*).

Der Lieblingsgarten der Prinzessin

Der **Lieblingsgarten der Prinzessin** wird besonders gewertet. Zunächst erhält jeder Spieler, der dort mit einem Prinz vertreten ist, die Erfolgspunkte wie auf jedem anderen Garten auch. Zusätzlich werden aber noch **Sonderpunkte** verteilt (und zwar unabhängig davon, auf welcher Ebene die Prinzessin steht):

- **Bei der ersten Wertung:**
- 5 Sonderpunkte, wenn Ihr Prinz dort auf Ebene 1 steht.
- **Bei der zweiten Wertung:**
- 10 Sonderpunkte, wenn Ihr Prinz dort auf Ebene 2 steht.
- **Bei der dritten Wertung:**
- 15 Sonderpunkte, wenn Ihr Prinz dort auf Ebene 3 steht.

Steht einer Ihrer Prinzen auf dem **Lieblingsgarten der Prinzessin** und erfüllt diese Bedingung, kassieren Sie für ihn die normalen Erfolgspunkte und zusätzlich die Sonderpunkte. Erfüllen Prinzen von verschiedenen Spielern diese Bedingung, erhalten **alle** die entsprechenden Erfolgspunkte und Sonderpunkte.

Stehen **zwei Prinzen einer Farbe** auf dem **Lieblingsgarten der Prinzessin**, von denen einer die Bedingung erfüllt, so erhält **nur** dieser die Sonderpunkte. Falls der zweite Prinz auf einer höheren Ebene steht, so erhält dieser die Erfolgspunkte, andernfalls gehen diese auch an den Prinz, der die Bedingung erfüllt. Erfüllen beide die Bedingung, kann nur ein Prinz gewertet werden (Sonderpunkte Erfolgspunkte).

Nach einer Wertung

Sobald eine Wertung beendet ist, beginnt eine neue Phase. Die Spieler erhalten neue Bausteine und das Spiel wird fortgesetzt.

Die Prinzessin verschieben

Der Spieler, der nach einer Wertung auf der Wertungsleiste die wenigsten Punkte hat, kann den Prinzessin auf einen anderen Garten versetzen. Er darf sie vor Beginn der nächsten Phase auf einen beliebigen, freien Baustein einer beliebigen Ebene stellen. Ansonsten wird die Prinzessin nicht bewegt.

f) SPIELENDE

Das Spiel endet nach der dritten Wertung. Wer die meisten Wertungspunkte hat, ist der erfolgreichste Anwärter auf die Verlobung und gewinnt neben dem Herzen der Prinzessin auch das Spiel!

g) DIE AKTIONSKARTEN

(Kartentext in **fett**)

Sie können einmalig zwei Ebenen nach oben auf ein freies Feld klettern, z. B. von Ebene 1 auf Ebene 3.
Das neue Feld muß ein benachbartes Feld sein.

Setzen Sie zusätzlich einen Baustein vom allgemeinen Vorrat ein.
Es gelten die normalen Regeln für das Einsetzen von Bausteinen.

Setzen Sie unter einen Ihrer Prinzen einen Baustein vom eigenen Vorrat.

Nehmen Sie den Baustein. Ihr Prinz kann auf einer beliebigen Ebene stehen. Kein Garten darf durch diese Aktion höher werden als ihre Grundfläche groß ist. Sollte Ihr Prinz auf der Ebene 0 stehen, können Sie mit dieser Aktion sogar einen neuen Garten gründen, wenn das Feld des Prinzen keine andere Gärten berührt (diagonal ist zulässig).

Gehen Sie mit Ihrem Prinzen in ein benachbartes Tor und kommen Sie aus einem beliebigen Tor dieses Gartens auf ein freies Feld wieder heraus.

Sie können dabei beliebig viele Ebenen überwinden, solange Sie wieder aus einem Tor heraustreten und das neue Feld frei ist. Sie dürfen Ihren Prinz nicht auf ein beliebiges Feld des Gartens stellen, sondern müssen aus einem Tor heraustreten, das sich neben dem Baustein befindet, auf den Sie ziehen möchten.

Es stehen Ihnen in diesem Spielzug insgesamt 6 Aktionspunkte zur Verfügung.

Sie können über einen zusätzlichen Aktionspunkt in diesem Zug verfügen.

Es stehen Ihnen in diesem Spielzug insgesamt 7 Aktionspunkte zur Verfügung.

Sie können über zwei zusätzliche Aktionspunkte in diesem Zug verfügen.

Gehen Sie in Ihrem Zug 1 Feld diagonal. Sie können dabei auch 1 Ebene überwinden.

Sie können mit diesem Zug auch von einem Garten auf einen anderen wechseln, wenn sich die beiden Gärten an den Ecken berühren und Ihr Prinz auf einer dieser Ecken steht.

Versetzen Sie einen beliebigen, freien Baustein auf dem Spielplan an einen anderen freien Platz. Ein Garten darf dabei nicht geteilt werden.

Wenn ein Garten aus nur einem Baustein besteht, können Sie auch diesen Baustein versetzen und auf oder an einen bestehenden Garten bauen, so daß es nach Ihrem Zug einen Garten weniger gibt. Es müssen aber immer mindestens 6 Gärten vorhanden sein. Außerdem darf kein Garten höher sein als seine Grundfläche groß ist. Sie können durch diese Aktion aber auch eine neue Garten gründen, wenn Sie den Baustein bei einer Garten wegnehmen und ihn auf ein freies Feld setzen, so daß er keine andere Garten berührt (diagonal ist zulässig).

Sie können einen eigenen Prinzen vom Spielplan zurücknehmen und auf ein anderes Feld - gemäß den Einsetzregeln - kostenlos wieder einsetzen.

Sie können einen fremden Prinz überspringen. Die Ebene des übersprungenen Prinzs ist beliebig. Das Feld, auf dem Sie landen, darf auf gleicher Ebene, 1 Ebene höher oder beliebig tiefer sein.

Sie können den Prinz nur in gerader Richtung überspringen.

3.3 Bedienungsanleitung

3.3.1. Installation von Semiramis

Die Installation von Semiramis setzt sich zusammen aus einem Teil von Hilfsprogrammen, die aufgrund von Lizenzbestimmungen nicht direkt Teil der Distribution sein können, sondern einzeln zu installieren sind, und dem eigentlichen Semiramis-Programm.

Für Windows-Anwender ist eine Batch-Datei enthalten, die dies alles automatisch für Sie vornimmt. Bitte beachten Sie: Semiramis ist ein Java-Programm und setzt voraus, daß Sie zumindest die Java 1.3 Runtime Edition installiert haben[2].

Wechseln Sie bitte auf Ihr CD-Laufwerk[3].

Wechseln Sie auf das Verzeichnis „*Semiramis*“.

Doppelklicken Sie die Datei „*setsem.bat*“.

Andere Anwender müssen bitte die in dem Verzeichnis „*Semiramis*“ auf der beigelegten CD enthaltenen Dateien gemäß der jeweils beigefügten Installationsanleitungen entpacken und ggfls. installieren. Denken Sie bitte daran, die Pfade zu den notwendigen Dateien von Junit, log4j, Java-Classfiles etc. gemäß Anleitung der einzelnen Pakete zu setzen. Bitte merken Sie sich den Pfad, in dem Sie Semiramis installiert haben.

3.3.2. Starten

Zum Starten von Semiramis ist ein Server erforderlich. Dieser verwaltet das Spiel und ermöglicht zudem, daß Sie auch über ein Netzwerk mit Freunden oder gegen einen Computerspieler antreten können.

Dieser Server kann entweder direkt über eine Eingabeaufforderung oder – komfortabler – über die unten näher beschriebene ServerGUI gestartet werden. Das Anmelden eines auf dem Server verwalteten Spiels (jeder Server kann auch mehrere Spiele verwalten) kann zum einen per Telnet oder zum anderen über einen weitgehend automatisierter Ablauf initiiert werden. Möchten Sie nur reinen Spielspaß, dann ersparen Sie sich das Hantieren mit der Eingabeaufforderung und wechseln Sie direkt zu *Kapitel 3.3*.

[2] www.java.sun.com

[3] Bitte beachten Sie die Anweisungen Ihres Betriebssystems, wie Sie dies tun können.

3.3.2.1.Manueller Start über eine Eingabeaufforderung

In der DOS-Eingabeaufforderung nach dem Wechseln auf das Verzeichnis, welches die ausführbare Jar-Datei des Programms Semiramis enthält, kann der Server gestartet werden.

Server-Start

Wechseln Sie bitte auf das Semiramis-Verzeichnis, in welchem Sie Semiramis installiert haben[4].

Geben Sie bitte ein: java –jar torfuServer.jar

Anmelden eines Spiels

In der DOS-Eingabeaufforderung nach dem Wechseln auf das Verzeichnis, welches die ausführbare Jar-Datei des Programms Semiramis enthält, kann ein Spiel eingerichtet werden. Sie können dies per Telnet textorientiert[5] oder sehr viel benutzerfreundlicher über die graphische Oberfläche von Semiramis erledigen. Dafür müssen Sie Semiramis starten.

Starten von Semiramis

Wechseln Sie bitte auf das Semiramis-Verzeichnis, in welchem Sie Semiramis installiert haben[6].

Geben Sie bitte ein: java –jar swp2.jar

Nach einer Weile erscheint nun die graphische Oberfläche von Semiramis, zunächst mit dem Teil, der Ihnen das Einrichten eines Spiels auf dem Server sowie das Anmelden hieran ermöglicht. Diesen Teil nennen wir ServerGUI.

[4] Bitte beachten Sie die Anweisungen Ihres Betriebssystems, wie Sie dies tun können.

[5] Lesen Sie dafür bitte *Kapitel 6.1*

[6] Bitte beachten Sie die Anweisungen Ihres Betriebssystems, wie Sie dies tun können.

3.3.3. ServerGui

Die ServerGUI enthält Knöpfe, um

- ein Spiel auf einem Server einzurichten
- einen Spieler an einem Spiel auf einem Server anzumelden
- einen Zuschauer an einem Spiel auf einem Server anzumelden
- einen Server ordentlich zu beenden.

Spielserver auf localhost starten

Klicken Sie hierauf, wenn Sie den notwendigen Spielserver noch nicht gestartet haben sollten und dies komfortabel graphisch tun wollen.

Geben Sie eine Portnummer an (Standard: 14195). Verwenden Sie am besten keine gebräuchlichen Ports (z.B. 110 für POP3, 25 für SMTP, 80 für http-Webserver).

Spielserver auf localhost beenden

Hiermit beenden Sie den auf Ihrer Maschine gestarteten Spielserver, damit aller von ihm verwendeter Speicher wieder freigegeben wird.
Zur Zeit ist diese Funktion deaktiviert.

Spielserveradresse angeben

Wählen Sie dies aus, wenn Sie Semiramis mit Freunden über das Netzwerk spielen möchten. Sie müssen wissen, wie der Rechner heißt, auf dem der Spieleserver läuft und auf welchem Port er „lauscht“.

Hinweis: Ähnlich, wie die Straße, in der Sie wohnen einen Namen hat, hat auch jeder Computer im Internet eine eindeutige Adresse (IP-Nummer), der im Normalfall ein für uns Menschen besser verständlicher Name zugeordnet ist. Die Portnummern kann man bei dieser Analogie dann als Hausnummern in dieser Straße betrachten.

Spiel erstellen

Hiermit melden Sie auf dem im Serverfenster mitgeteilten Server ein Spiel an. Dieses dient wiederum als Container für die Teilnehmer eines Spiels.

Wählen Sie den Namen des Spiels, die Anzahl der Teilnehmer und ein Timeout, nachdem ein Zug abgegeben sein muß, bevor der Spieler wechselt. Sie können auch ein Paßwort vergeben.

Beachten Sie bitte: Sie müssen dieses Fenster immer ausfüllen, selbst, wenn Sie mit Freunden über ein Netzwerk spielen möchten. Ggfls. müssen Sie sich die notwendigen Daten inkl. Rechneradresse und Portnummer von Ihren Freunden geben lassen.

Computergegner starten

Möchten Sie alleine gegen einen Computergegner spielen oder einfach einen bzw. Computergegner haben, um eine gesellige Runde zu erhalten, dann wählen Sie diesen Button durch Klick mit der Maus oder Tastatur aus. Das auszufüllende Fenster entspricht dem aus „*An gewähltem Spiel teilnehmen*“.

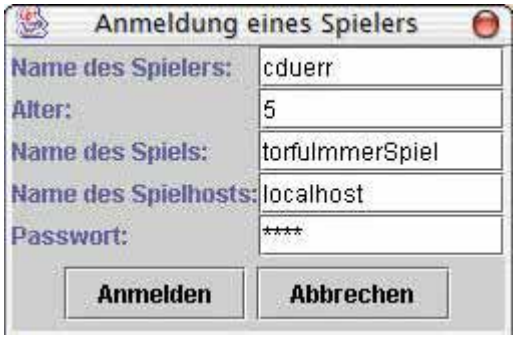
Bitte beachten Sie:
Damit Sie diesen Knopf drücken können ohne einen Fehler zu erzeugen müssen Sie zunächst den entsprechenden Server in der Server-Auswahlliste auswählen und das Spiel, an dem der Computergegner angemeldet werden soll, auswählen. Ggfls. ist ein Server hinzuzufügen bzw. zu starten oder ein neues Spiel zu erzeugen.

Hilfe

Für eine Online-Hilfe, die die entsprechenden Dialoge und Möglichkeiten von Semiramis erklärt, klicken Sie bitte hier. Analog können Sie die Taste F1 verwenden.
Zur Zeit ist die Hilfe nicht verfügbar.

An gewähltem Spiel teilnehmen

Unter diesem Menüpunkt verbirgt sich das Anmeldefenster, um an einem Spiel teilzunehmen.



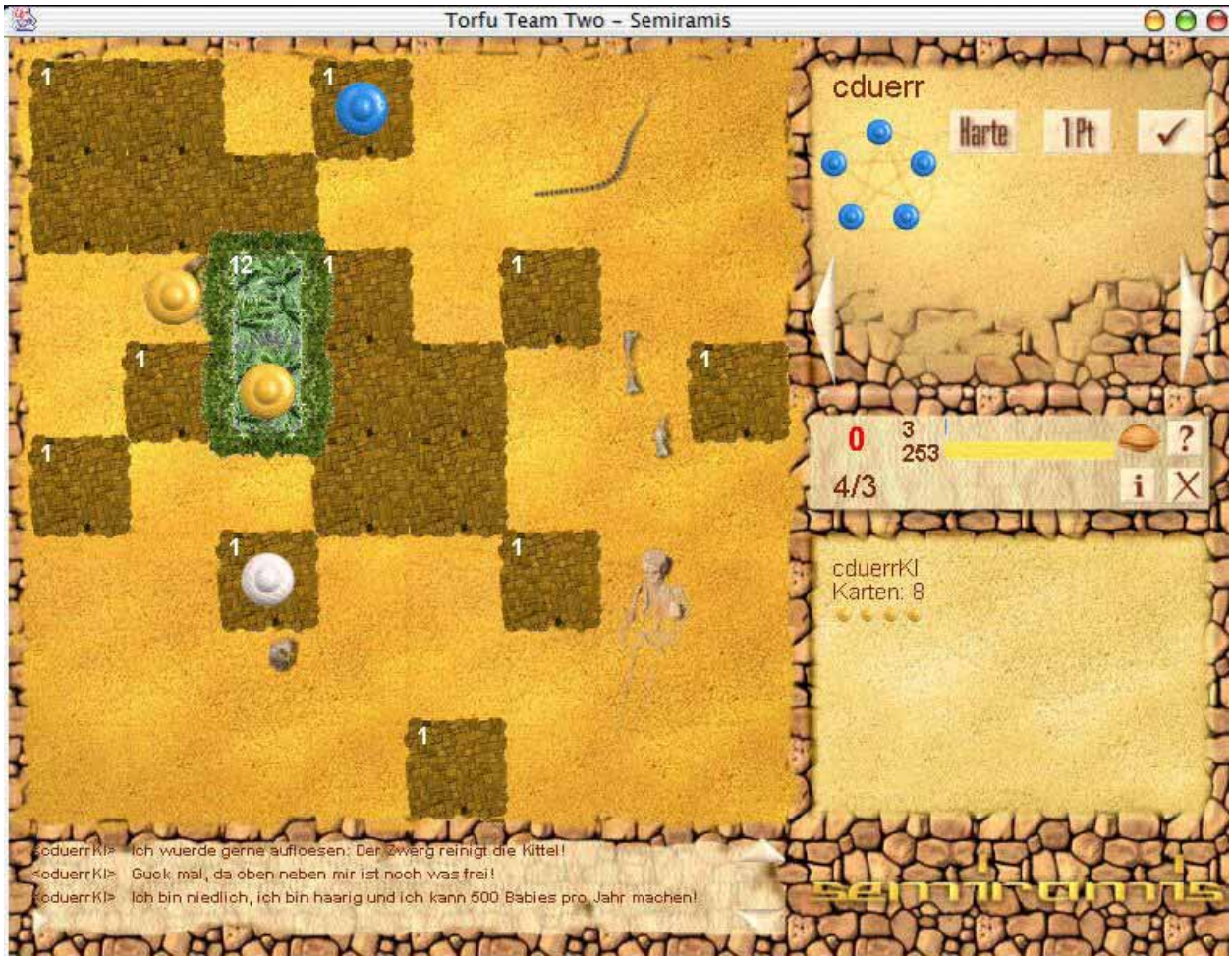
Um sich an einem existierenden Spiel anzumelden, müssen die unteren drei Werte exakt so eingegeben werden, wie Sie sie erhalten haben. Haben Sie das Spiel aus der Serverauswahlliste und der Spielauswahlliste markiert, so sind die entsprechenden Felder bereits so eingetragen, daß Sie nichts mehr weiter tun müssen, als Namen und Alter (frei) zu wählen.

Bitte beachten Sie:
Damit Sie diesen Knopf drücken können ohne einen Fehler zu erzeugen müssen Sie zunächst den entsprechenden Server in der Server-Auswahlliste auswählen und das Spiel, an dem der Computergegner angemeldet werden soll, auswählen. Ggfls. ist ein Server hinzuzufügen bzw. zu starten oder ein neues Spiel zu erzeugen.

Bei gewähltem Spiel zuschauen

Sie können an einem Spiel teilnehmen, ohne spielberechtigt zu sein. Klicken Sie hier, um das Spiel und alle Nachrichten zu verfolgen.

3.3.4. GUI



Die Darstellung des Spielfeldes ist untergliedert in die 8*8-Kästchen-Darstellung vom reichen Babylon (links), einem Chat-Feld darunter sowie zwei Informations- und Vorratsfeldern rechts oben und rechts unten mit dazwischen liegender Statusinformation bezüglich Restzugdauer (rote Null) und ein paar Bedienelementen, deren Zweck sich intuitiv erschließt (Info, Beenden, Hilfe). Verhaart man auf einem Bildelement eine zeitlang mit dem Mauszeiger, so klappt ein sogenanntes Tool-Tip hervor, welches die Bedeutung des Elements zeigt.

Beispiel für Tooltips:

3.3.4.1. Allgemeines

Chat

Ganz unten links befindet sich das Chat-Fenster, in dem alle eingehenden Nachrichten automatisch untereinander angezeigt werden. Möchten Sie etwas ältere Nachrichten sehen, oder wird eine Nachricht nicht vollständig dargestellt, so benutzen Sie bitte die Pfeile sinngemäß, um nach oben oder unten zu „scrollen“.

Möchten Sie eine Nachricht an einen Mitspieler abgeben, so klicken Sie kurz in den Bereich, der rechts neben dem nach rechts zeigenden untersten Pfeil liegt. Es erscheint dann ein Cursor, der angibt, daß der Computer bereit ist, Ihre Nachricht über die Tastatur einzulesen. Schließen Sie bitte die Eingabe mit Return ab.

Beachten Sie bitte:
Während Sie Ihre Nachricht eingeben, läuft ggfls. Ihre Spielzeit weiter.

Spielerwechsel

Das Spielfeld des Spielers, der gerade an der Reihe ist, wird beleuchtet und somit graphisch hervorgehoben.

Spieler 1

Ganz links oben sehen Sie den Namen des Spielers – dies ist gleich für alle Spieler. Rechts oben erscheint die Anzahl der noch für Ihren Zug verbleibenden Aktionspunkte. Darunter (von links nach rechts) sind aufgereiht:

- Prinzen
- Karte kaufen
- Wertungspunkt kaufen
- Zug vorzeitig für beendet erklären.

Darunter sind Ihre Bausteintürmchen aufgelistet, darunter Ihre Aktionskarten, so Sie denn eine gekauft haben.

Computergegner

Die Angaben für den Computergegner haben mehr informativen Charakter. Weitere Spieler erhalten wie ein möglicher Computergegner einen Bereich in dem Fenster.

Statistik

Links oben sehen Sie mit der roten Null die normalerweise rückwärtslaufende Restzugdauer eines Zuges. Daneben finden Sie die Anzahl der Wertungspunkte, wie sie sich aktuell darstellen (Spieler 1 oben, darunter Spieler 2-4) und ganz rechts außen können Sie ? für Hilfe, i für info und X für Beenden aktivieren.

3.3.4.2.2D-Modus

Zur Zeit ist nur der 2D-Modus von Semiramis voll funktionsfähig.
Die Darstellung ist auf Effizienz hin ausgerichtet und ermöglicht es den Spielern, auf einen Blick alle erforderlichen Daten zu erfassen und bleibt dennoch übersichtlich.

Die Funktionsweise der graphischen Oberfläche ist schnell und leicht erklärt: Nahezu alles, was es zu tun gibt, erledigt man am besten über das sogenannte „Drag'n'Drop". Dazu positioniert man seinen Mauszeiger über einem der Spielobjekte wie Bausteinen, Prinzen oder Prinzessin, drückt eine Maustaste, läßt diese gedrückt und zeiht bei gedrückter Maustaste das entsprechende Objekt an die Zielposition.

Beispiel:

Ist ein Zug dann möglich, sieht der Mauspfeil wie nebenstehend aus.

Ist ein Zug dann nicht möglich, beinhaltet der Mauspfeil ein „Verboten"-Symbol.

Andere Bedienelemente reagieren durch Einfach-Klick (z.B. Aktionskarten).

ENTWICKLUNG

4. Entwicklung

4.1 Einleitende Beschreibung der Entwicklung

Die Entwicklung der Applikation Semiramis wurde von der Tutoriumsgruppe SWP2 vorgenommen.

4.1.1. Projektorganisation

Grundsätzlich war die Entwicklung in Projektform organisiert. Die Form des Projektmanagements, wie sie von der Tutoriumsgruppe SWP2 angewendet wurde, wird auch als Extreme Programming bezeichnet, die allerdings nicht ganz konservativ verstanden wurde.
Grundsätzlich hatte jeder Teilnehmer der Tutoriumsgruppe zwei Aufgabenstämme, damit er in zwei Gruppen mitarbeiten konnte.
Aufgeteilt wurde:

- Serverkommunikation
- Look&Feel
- GUI
- Logik
- Künstliche Intelligenz
- Rechtsanwalt

Zusätzlich stand am Anfang noch an, sich mit verschiedenen, später dann sowohl arbeitserleichternden als auch das Programmieren dieser umfangreichen Anwendung überhaupt erst ermöglichenden Hilfsmittel wie Debugger, IDEs, Logging-Mechanismus, Projektmanagement und ähnliches, auseinanderzusetzen.
Dies setzte voraus, daß gerade am Anfang die Arbeitsbelastung etwas intensiviert werden mußte, da auch z.B. die Serveranbindung als erstes fehlerfrei arbeiten können mußte. Wie sich herausgestellt hat, stimmt die Aussage nicht im Umkehrschluß: Weniger Arbeit ist es nicht geworden...

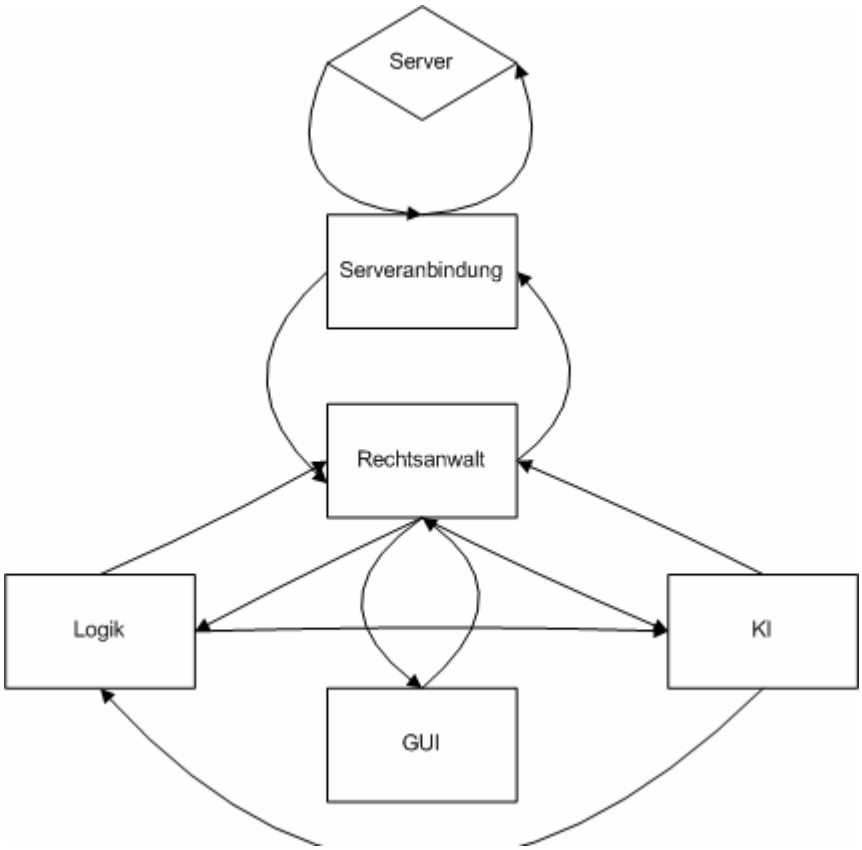
4.1.2. Verwendete IDEs

Viele arbeiteten mit Borland JBuilder Version 3, vor allem, da er in den am häufigsten verwendeteten Räumen des Mathematik-pi-Gebäudes installiert war. Zusätzlich gab es noch einen Vortrag über netBeans, doch – gehe ich von mir aus – habe ich intelliJ idea besonders liebgewonnen. In absehbarer Zeit werde ich mir die Software kaufen und kann sie nur jedem empfehlen.

4.2 Allgemeine Programmstruktur (Module und Aufgaben, Kommunikation)

4.2.1. Grundidee

Wie schon aus der Verteilung der Projektaufgaben ersichtlich, gab es ein Konzept, wie das Programm aufgebaut werden sollte. Prinzipiell sollen Aktionen ausgelöst werden, wo auch immer etwas passiert und weitergegeben werden. Wie das im Einzelnen abläuft, ist im *Kapitel 4.3 Logik* beschrieben.



4.2.2. Quellcode

Der Quellcode ist auf der beiliegenden CD als Zip-Datei enthalten[7].

4.2.3. Dokumentation

Die JavaDoc ist auf der beiliegenden CD als Zip-Datei enthalten[8].

4.3 Logik

Es sollte zum Verständnis des Codes vorausgeschickt werden, daß Burg gemäß Ursprungsspiel für Garten steht und der König im Ursprungsspiel die Prinzessin ersetzt (außer, wenn der König zur Bewertung herumreist).

4.3.1. Spielverwaltung

Logik.java
In Logik.java wird die Spiellogik verwaltet. Die 64 Felder des Spielfelds werden in einem Array von Feldobjekten gespeichert. Die Spieler werden durch einer Liste von Spielerobjekten dargestellt. Außerdem gibt es eine Liste von Burgobjekten, die die Arbeit mit der Spielfelddarstellung erleichtern.

Feld.java
Jedes Feld kennt seine Höhe, Position und weiß, ob eine Spielfigur drauf steht.
In Feld stehen Methoden, um zu bestimmen, ob das Feld bebaubar ist und ob ein weiteres Feld von diesem Feld aus in einem Schritt erreichbar ist.

Burg.java
Burg hat Referenzen auf die zur Burg gehörenden Feldobjekte. Außerdem wird die Maximalhöhe gespeichert. Es gibt eine Methode zur Bestimmung der benachbarten Burgfelder.

Spieler.java
Der Spielstand jedes Spielers, also Siegpunkte, Prinzpositionen etc. wird in einem Spielerobjekt gespeichert.

[7] Verzeichnis *Source*

[8] Verzeichnis *Source*

4.3.2. Ereignisklassen

Um jegliche Art von Geschehnissen verarbeiten zu können, werden Ereignisobjekte erzeugt. Diese Ereignisobjekte beinhalten all das, was ausgeführt werden soll.
Aktionsobjekte sind spezielle Ereignisobjekte, die Geschehnisse auf dem Spielfeld widerspiegeln.

Ereignis.java
Ereignis ist eine abstrakte Klasse mit folgenden Methoden:

```
public abstract void doAktion(Logik l);
```
führt die Ereignisse auf der angegebenen Logik aus

```
public abstract boolean checkAktion(Logik l);
```
prüft, ob ein Ereignis zulässig ist
Bei Aktionen werden zum Beispiel die Spielregeln überprüft.

```
public abstract void anzeigen(UI i);
```
aktualisiert das UI

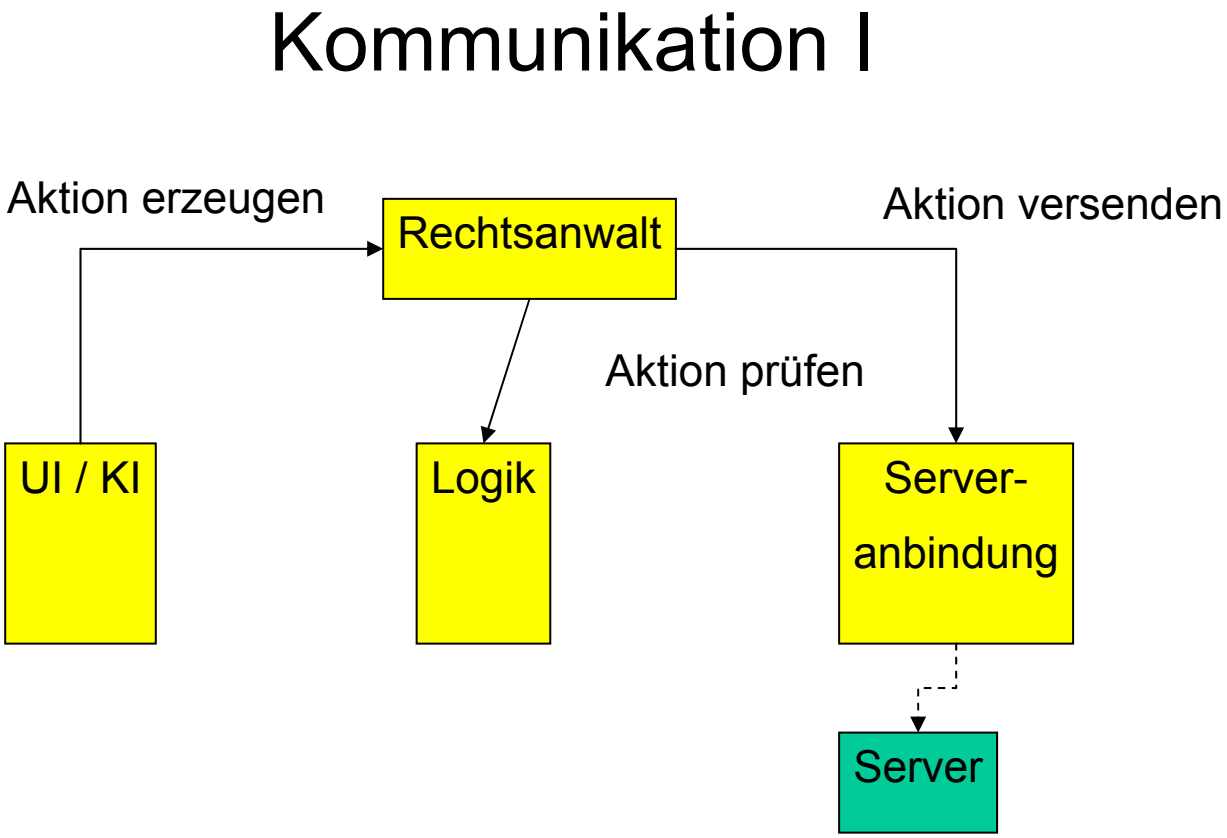
```
public abstract void benachrichtigeKI(KI ki);
```
benachrichtigt die KI

```
public abstract void verarbeiteEreignis(ServerAnbindung s);
```
schickt das Ereignis zum Server

```
public void ereignisVersenden(ServerAnbindung s);
```
schickt das Ereignis an den Rechtsanwalt

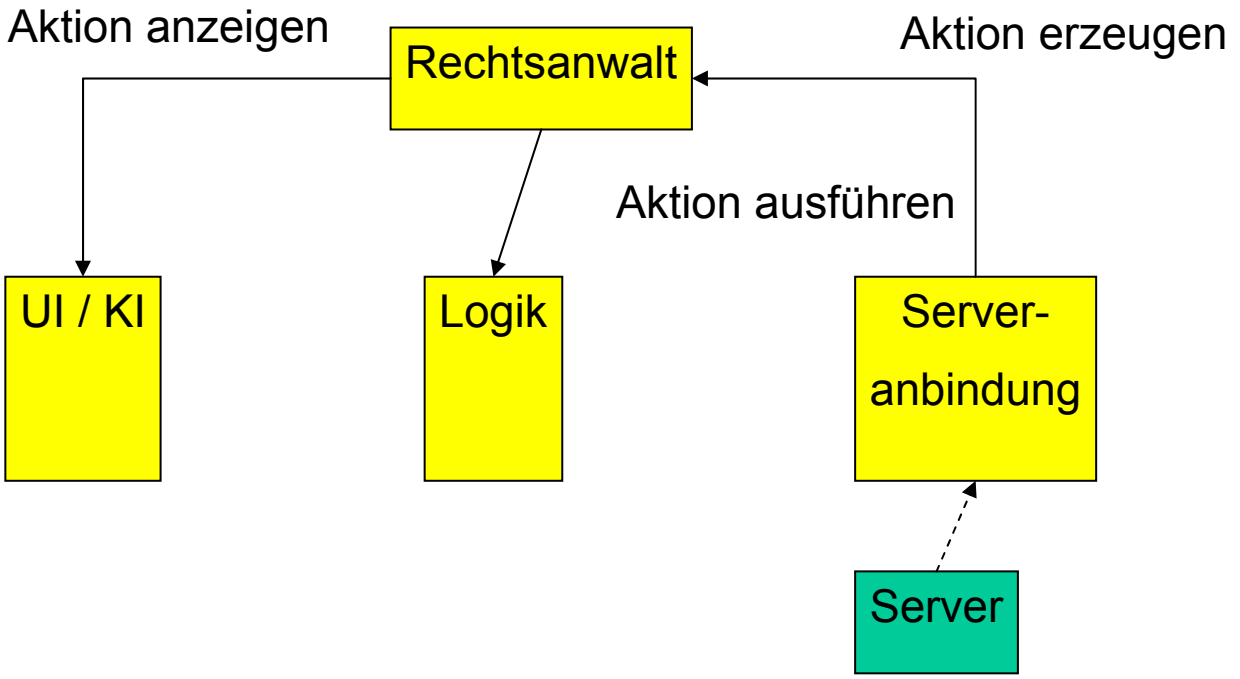
4.3.3. Kommunikationswege/Rechtsanwalt

Der Rechtsanwalt ist die Kommunikationszentrale unseres Programms. Er verteilt sämtliche Informationen, genauer Ereignisobjekte, an die einzelnen Komponenten.
Der Rechtsanwalt stellt die folgenden beiden Kommunikationswege zur Verfügung:



Das UI / die KI erzeugen ein Ereignisobjekt und schicken es an den Rechtsanwalt. Der Rechtsanwalt fragt bei der Logik an, ob das Ereignis zulässig ist, das heißt, die Methode checkAktion des Ereignisses wird aufgerufen. Schließt die Prüfung positiv ab, wird das Ereignis an die Serveranbindung übergeben, das heißt, die Methode verarbeiteEreignis dieses Ereignisses wird aufgerufen. Damit ist das Ereignis beim Server angekommen.

Kommunikation II



Kommt nun irgendeine Meldung vom Server erzeugt die Serveranbindung ein Ereignis und schickt es an den Rechtsanwalt, das heißt, die Methode ereignisVersenden des Ereignisses wird aufgerufen. Dann gibt der Rechtsanwalt das Ereignis an Logik, UI und KI weiter, das heißt die Methoden doAktion, anzeigen und benachrichtigeKI des Ereignisses werden aufgerufen.

4.4 KI

4.4.1. Vorgehensweise (Mustersuche, Strategien, Ziele usw.)

Grundsätzlich arbeitet die KI als künstlicher Spieler. Die KI hat Strategien implementiert, nach denen sie Bewertungen von möglichen Zielen vornimmt. Ziele sind dabei definiert als mögliche Endresultate von mehreren Zügen.

4.4.1.1.Allgemeine Struktur

Die Künstliche Intelligenz berechnet ihre Züge jeweils im Hinblick auf die aktuelle Spielsituation, indem sie auf dem Spielbrett nach Mustern sucht, für die ihr (das heißt uns, den Programmierern) Herangehensweisen bekannt sind. Wird sie zur Abgabe eines Zuges aufgefordert, überprüft sie zunächst, welche Strategien es sich überhaupt zu verfolgen lohnt.
(Beispiel: Existieren auf dem Spielfeld nur Burgen mit sehr geringer Grundfläche, brauchen keine eventuell möglichen Ziele zum Besetzen von neuen Burgen berechnet werden.)

In Abhängigkeit der Strategien werden dann jeweils sogenannte Ziele erzeugt.
(Beispiel: Je ein Ziel für eine für Besetzung in Frage kommende Burg.)

Bei der Erzeugung eines Ziels werden die Aktionen berechnet, die zum Erreichen dieses Ziels nötig sind. Bei mehreren Möglichkeiten wird die beste gewählt, also jene mit dem größeren Punktzuwachs - die Länge des Weges (d. h. die Kosten für die nötigen Aktionen) wird hierbei außer Acht gelassen.
Alle durch die Strategien erzeugten Ziele werden nun in Bezug auf den aktuellen Punktzuwachs bewertet und das höchsten bewertete Ziel ausgeführt. Bleiben noch Aktionspunkte übrig bzw. können noch Karten verwendet werden, werden aufgrund der nun weiterentwickelten Spielsituation neue Ziele erzeugt und so fort.

4.4.1.2.Code-Struktur

Die KI implementiert Runnable und wird vom Rechtsanwalt erzeugt. Im Konstruktor werden die Strategien erzeugt. Das sind im wesentlichen die BesetzeBurgenStrategie, StatischerBurgausbauStrategie, KaufeKarteStrategie, NehmePunktStrategie und zusätzlich für jede Karte eine Strategie.

Dann ruft der Rechtsanwalt die Methode anmelden() mit den Kommandozeilenargumenten bei der KI auf. Nachdem sie sich bei dem Spiel angemeldet hat, wird wait() aufgerufen und der Main-Thread der RechtsAnwalt-Klasse wartet bis zum Spielende.

Wenn der Spieler KI am Zug ist, wird von der Serveranbindung ein ZugBeginn-Ergeignis erzeugt. Die Ereignis-Objekte implementieren die Methode benachichtigeKi(). In dieser Methode wird bei ZugBeginn ein neuer Thread mit der Runnable KI erzeugt. In der run() der KI selbst wird nur macheZug() aufgerufen; nach dem Zug endet der Thread wieder.

4.4.1.3.Züge ausführen

In der macheZug()-Methode werden für jede Strategie die entsprechenden Ziele erzeugt, die sich aus einzelnen Aktionen zusammensetzen. [Jedes Ziel hat seine eigene Bewertungsfunktion. Die Aktionen des Zieles mit der höchsten Bewertung werden ausgeführt, wenn dann noch Aktionspunkte übrig sind, werden erneut Ziele erzeugt und wieder das beste davon ausgeführt, usw...]
Wenn beim Ausführen eines Zieles etwas schiefgegangen ist, werden für die restlichen Aktionspunkte nur noch Punkte gekauft.

4.4.1.4.Ausführen eines Zieles

Die Aktionen werden der Reihe nach ausgeführt, indem beim Rechtsanwalt fuehreAktionAus() aufgerufen wird. Nach diesem Aufruf wartete der KI-Thread so lange, bis er von dem Aktionsobjekt, das vom Server als Bestätigung kommt, aufgeweckt wird (in den Aktionsobjekten passiert genau das in der benachrichtigeKI()).

4.4.1.5.Strategien

NehmePunktStrategie

Diese einfachste und grundlegende Strategie erzeugt immer ein NehmePunktZiel, welches ggf. einen Siegpunkt für einen Aktionspunkt kauft. Dieses Ziel ist standardmäßig am niedrigsten gewertet, kann aber durchaus höher bewertet werden, wenn sich durch das Kaufen eines Siegpunktes andere Spieler auf der Wertungsleiste überspringen lassen.

StatischerBurgausbauStrategie

Diese Strategie erzeugt (je Prinz) ein entsprechendes Ziel, wenn die KI noch einen Baustein setzen kann. Im Ziel wird geprüft, ob neben dem Prinz ein bebaubares Feld existiert, welches auf gleicher Höhe bzw. um eine Ebene niedriger ist. Hierbei werden höherwertige Felder vorgezogen. Bei Bedarf wird dieses Feld mit einem Stein bebaut. Bewertet wird das Ziel im Allgemeinen aus der Grundfläche der neuen Burg (sie bestimmt den Punktzuwachs zwischen aktueller und zukünftiger Standhöhe). Erweitert wurde dieses Ziel um den Anbau an nichtbesetzte Burgen in direkter Zusammenarbeit mit einem BesetzeBurgZiel (s. u.).
(Anekdote: Der Name dieser Strategie ist aus einer ursprünglich vorgesehenen Strategie entstanden, die nach einem bestimmten, streng vorgegeben Muster eine Burg ausbauen sollte.)

BesetzeBurgenStrategie

Diese Strategie erzeugt zugehörige Ziele zum Besetzen jeweils einer anderen Burg, wenn diese Burg einer minimalen Grundfläche entspricht. 'Burg besetzen' zielt dabei nicht nur auf unbesetzte Burgen, sondern vor allem auch auf den Aufstieg in bereits besetzen Burgen. Die Ziele errechnen Aktionen zum Erreichen des werthöchsten freien und erreichbaren Feldes auf einer Burg. Dabei wird beachtet, ob es sich um Königsburgen bzw. -Ebenen handelt und ob sich bereits andere eigene Prinz auf der Burg befinden. Der Wert eines Feldes (und somit auch des Zieles) wird dabei bestimmt aus der Differenz zwischen Zielfeld- und Startfeldwert des zu ziehenden Prinzen. Bei Bedarf wird hier auch gleich ein neuer (wertloser) Prinz vor dem Ziehen eingesetzt. Um zu gewährleisten, daß zum Einsatz von Sonderzugkarten oder zum Anbau an neue Burgen auch vermeintlich weniger wertvolle Felder (beispielsweise solche auf Ebene 0) angesteuert werden, werden diese Felder bei Bedarf so gewertet, als ob die erst im nächsten Zug folgenden Aktionen bereits ausgeführt werden.

KaufeKarteStrategie

Karten werden grundsätzlich nur in den ersten beiden Runden gekauft. Die Bewertung ist ein Punkt höher als die des NehmePunktZieles. Karten werden also nur gekauft, wenn nichts sinnvolleres gemacht werden kann.

Kartenstrategien

KarteVersetzeBaustein, Karte6AP, Karte7AP,KarteZusatzbausteinStrategie rufen die anderen Strategien auf (die Zusatzbausteinkarten nur StatischerBurgausbauStrategie, die Punktekarten zusätzlich noch BesetzeBurgenStrategie) und übergeben die Zusatzaktionspunkte, bzw. den Zusatzbaustein als Parameter.
Die übrigen Karten suchen nach allen Möglichkeiten sich einzusetzen und bewerten diese mit dem Punktezuwachs.

4.4.1.6.Bewertung

Jedes Ziel bringt seine eigene Bewertungsfunktion mit. Diese bewertet im wesentlichen den Punktezuwachs. Nur die Ziele NehmePunkt und KaufeKarte haben feste Werte (Nehmepunkt 1 + überholte Spieler, KaufeKarte 2 in den ersten beiden und 0 in der dritten Runde).

4.4.2. Das swp-KI-Turnier

Wie auch schon während des Softwarepraktikums, in dem Turniere durchgeführt wurden, läßt sich dies auch zuhause bewerkstelligen.
Dazu muß das swp2.jar mit den Parametern für den Turniermodus gestartet warden.
Starten Sie eine Eingabeaufforderung.
Wechseln Sie dort in das Semiramis-Verzeichnis.
Geben Sie bitte ein (und wechseln Sie die Variablen aus):
Java –jar swp2.jar hostname portnummer Spielname Spielpaßwort Spielername Alter

Bitte beachten Sie: Für den Turniermodus muß bereits ein Server mit entsprechend eingerichtetem Spiel laufen und erreichbar sein.

4.4.3. Re-engeneering

Wünschenswert ist das Ausmerzen einiger kleiner Bugs sowie das Sicherstellen der Lauffähigkeit auf allen unterstützten Plattformen. Als Ziel bis zum Ende des Softwarepraktikums ist beabsichtigt, noch eine 3D-GUI zu integrieren. Auch die Dokumentation weist noch leichte Verbesserungs- und Ergänzungsmöglichkeiten auf.

CREDITS UND PERSÖNLICHES

5. Persönliches

5.1 Persönliche Erfahrungen von Kommilitonen/Kommilitoninnen

5.1.1. Jenny Wöß:

Meine ersten Programmiererfahrungen in Java machte ich, als mir die Überarbeitung eines großen Java-Programmes aufgegeben wurde. Dort arbeitete ich alleine an einem riesigen, schlecht dokumentierten Code, dessen Semantik ich bei der Überarbeitung erst lernte.
Klar, daß das zu Frust und einer Ablehnung dem Programmieren gegenüber geführt hatte. Das Softwarepraktikum hat mir den Spaß am Coden zurückgegeben, denn dort arbeiteten wir als großes Team an dem Code.
Besonders motivierend fand ich:

- die gemeinschaftliche Arbeit am Code,
- daß jeder jeden IMMER um Hilfe bitten konnte,
- daß man sich recht schnell dem Ziel zu nähern schien,
- daß man nie das Gefühl hatte etwas sinnloses zu tun,
- und auch, wenn ich selber nicht direkt daran beteiligt war: die wöchentlichen Turniere.

Was ich nicht so toll fand, war das zu Ende des Semesters, die gemeinschaftlichen Programmierzeiten immer weiter in die Nacht rutschten.

Vielen Dank an alle SWP2ler und Ulli, es hat viel Spaß mit Euch gemacht.

Jenny

5.1.2. Martin Hense:

Das Softwarepraktikum war eine großartige Erfahrung in vielerlei Hinsicht.
Zum einen konnte ich endlich prüfen, ob mein bisher während des Studiums gesammeltes Wissen zu etwas taugt. Die Antwort darauf scheint zu sein: Ja, das tut es, aber die Hälfte wäre ebenso ausreichend gewesen.

Da habe ich mich wie die anderen Informatiker die letzten Semester mit rekursiven Lösungen in Haskell und Beweisführungen von Algorithmen herumgeschlagen, mit B*-Bäumen und dem Lambda-Kalkül, und die lieben Mathematiker mögen das beste Zeugnis dafür zu sein, daß es auch ohne diese Umwege geht. Gut, Rekursionen und B-Bäume scheinen bei den reinen Informatiker-Gruppen auch ihre Anwendung gefunden zu haben, wie wir an der Laufzeit ihrer KIs sehen können *hehe*.

Zum anderen war es unheimlich spannend mitanzusehen, wie aus vielen einzelnen Teilen plötzlich ein großes Ganzes wurde, und das große Ganze macht einen jetzt um so stolzer, auch gerade weil man 'nur' an ein paar kleinen aber feinen Einzelteilen mitgewirkt hat (Anmerkung der Redaktion: Welch eine Untertreibung, Herr Hense!).

Vom informationstechnischen Standpunkt aus haben sich viele schon gelernte Dinge (wie Objektorientierung) nochmals vertieft; etliche anderen Techniken habe ich zusätzlich erlernt oder zumindest einen Einblick in sie bekommen. Es hat sich gezeigt, daß mir nun ein Fundament zur Verfügung steht, auf dem sich wirklich umfangreich aufbauen läßt - eine schöne Erfahrung zum Ende des Grundstudiums.

Die wohl größte Errungenschaft des Softwarepraktikums liegt für mich aber im zwischenmenschlichen Bereich (und ich vermute, das Softwarepraktikum ist in Wahrheit auch nur der Deckname für ein Gruppenseminar, bei dem die psychologischen und sozialen Fähigkeiten der Studenten unter Extrembedingungen getestet werden sollen). Nicht zuletzt konnte ich hier meine Vorurteile gegenüber anderen Informatikern (bzw. Programmierern) bestätigen oder verwerfen (Mathematiker können auch Spaß empfinden), sondern viel mehr noch meine eigenen wissenschaftlichen und sozialen Fähigkeiten prüfen. Teamwork, Zuverlässigkeit, Umgänglichkeit, pädagogisches Feingefühl, Zeitmanagement, kritische Argumentation, Ehrgeizigkeit, Offenheit, Rücksichtnahme, Spaß und Wein waren hier unverzichtbar. Wer hiervon für sich und andere nichts in einen zukünftigen Job mitnehmen kann, wird es später schwer haben.

Alles in allem war es eine echte Plackerei, hat aber auch unglaublich viel Spaß gemacht. War schön, Euch kennengelernt zu haben!

Martin

5.2 Credits

An dieser Stelle ist es Zeit, noch einmal Dank zu sagen. Zu allererst natürlich gebührt größter Dank an Herrn Dr. Ulrich Kortenkamp für seine Hilfe.

Vielen Dank auch an alle Teilnehmer des SWP2, die die Arbeit sehr angenehm gestaltet haben. Es war wirklich ein gutes Arbeiten mit Euch!

Besonderer Dank an:

- die Programmierer der KI für eine superschnelle und ziemlich erfolgreiche KI
- die Programmierer der GUI und die Designer des Look&Feel für ein Produkt, dessen Aussehen sich nicht vor professionellen Spielen verstecken muß
- die Programmierer der Serveranbindung für eine sauber arbeitende Kommunikation mit dem Server, denen Fehler aufgefallen sind, die andere aufgrund von Abstürzen nicht ermitteln konnten
- die Programmierer der Logik, die die Grundlagen für eine vernünftig laufende KI geschaffen haben

Nochmals Danke!

ANHANG

6. Anhang

6.1 Serverkommunikation

Server Kommunikation $Revision: 1.16 $

Syntax:

<...> Variable

[...] Optionale Felder

{a|b|c|...} Entweder a oder b oder c oder ...

Alles andere sind Terminalzeichen

Das was jetzt dasteht wird gesendet bzw. empfangen. Mit Ausnahme der Fehler

6.1.1. Anmeldung:

6.1.1.1.Spielerzeugung:

Zu Senden:

NEU <Spieler Anzahl>

NAME <Spielname>

PASSWD <Passwort>

TIMEOUT <Timeout in Millisekunden>

Bestätigung:

SPIEL_ANGEMELDET(Disconnect)

Fehler:

FEHLER_NAME_VERGEBEN(Disconnect)

6.1.1.2.Spielauflistung:

Zu Senden:

LIST

Bestätigung:

LISTE <anzahlSpiele>

<Name> <Angemeldete Spieler> <Spieleranzahl> {WARTET| GESTARTET| BEENDET}

<Name> <Angemeldete Spieler> <Spieleranzahl> {WARTET| GESTARTET| BEENDET}

...

ENDE

(Disconnect)

Fehler:

Keine(Disconnect)

6.1.1.3.Spielteilnahme:

Zu Senden:

JOIN <Spielname>

PASSWD <Passwort>

NAME <Spielername>

ALTER <Alter>

Bestätigung (an Alle):

SPIELER_ANGEMELDET <Spielername> <Alter>

Fehler:

FEHLER_NAME_VERGEBEN(Disconnect)

6.1.1.4.Spielzuschauen:

Zu Senden:

WATCH <Spielname>

NAME <Zuschauername>

Bestätigung (an Alle):

ZUSCHAUER_ANGEMELDET <Zuschauername>

Fehler:

Keine (Disconnect)

6.1.1.5.Zuschauer meldet sich ab (oder ist einfach weg):

Wenn ein Zuschauer sich abmeldet oder dessen Netzwerkverbindung weg ist, bekommen alle Spieler und Zuschauer diese Nachricht.

Gesendet:

ZUSCHAUER_ABGEMELDET <Zuschauername>

6.1.1.6.Netzgeschwindigkeit testen:

Zu Senden:

PING

Bestätigung:

PONG

Fehler:

Keine (Disconnect)

6.1.1.7.Spielstatus erfragen:

Zu Senden:

STATUS <Spielname>

Bestätigung:

Ein Statusbericht für das Spiel, siehe unten.

Fehler:

ERROR 1029: Es gibt kein Spiel mit diesem Namen. (Disconnect)

6.1.2. Vom Server kommende Nachrichten

6.1.2.1.Ersten Ritter setzen:

Beschreibung: Wird ganz am Anfang gesendet damit jeder Spieler seinen Ritter platziert

Gesendet:

RITTER_SETZEN

Reaktion:

AKTION SETZE_RITTER_ERSTESMAL

X <x>

Y <y>

Bestätigung : Standard Aktionsbestätigung

Fehler: s.u.

6.1.2.2.König setzen:

Beschreibung: Wird vor jeder Phase einem Spieler gesendet

Gesendet:

KOENIG_SETZEN

Reaktion:

AKTION SETZE_KOENIG

FARBE <f>

X <x>

Y <y>

Bestätigung : Standard Aktionsbestätigung

Fehler: s.u.

6.1.2.3.Spieler <Spielerfarbe> erhält die Zugaufforderung:

Alle Spieler haben in dieser Runde ihre Züge abgegeben.

Gesendet:

ZUG_BEGINNT <Spielerfarbe>

6.1.2.4.Spieler <Spielerfarbe> beendet seinen Zug:

Alle Spieler haben in dieser Runde ihre Züge abgegeben.

Gesendet:

ZUG_BEENDET <Spielerfarbe>

6.1.2.5.Runde beendet:

Alle Spieler haben in dieser Runde ihre Züge abgegeben.

Gesendet:

RUNDE_BEENDET

6.1.2.6.Spiel beendet:

Das Spiel ist zu Ende.

Gesendet:

SPIEL_BEENDET

6.1.2.7.Zugabgabe:

Beschreibung: Sobald ein Spieler einen Zug abgeben soll bekommt er diese Nachricht. *Für die Reaktion gilt: X ist horizontal, Y vertikal. Wie es eben üblich ist ...*

Gesendet:

ZUG_ABGEBEN

Reaktion:

ZUG
AKTION <Aktion>
[KARTE <Kartennummer>] -- Die komplette zweistellige Nummer
[X <x1> -- Wenn eine Positon benötigt wird
Y <y1>
[X <x2> -- Wenn eine zweite Position benötigt wird
Y <y2>]]
AKTION <Aktion>
.... -- Weitere Aktionen
ZUGENDE

Bestätigung :

Standardaktionsbestätigung für jede einzelne Aktion, zusätzlich Bestätigung des Zugendes (siehe einzelne Aktionen für Details).

Fehler:

Zugfehler (Siehe einzelne Aktionen)

FEHLER_NICHT_VORHANDEN -- Die Aktion gibt es nicht

FEHLER_NICHT_AM_ZUG

FEHLER_TIMEOUT_UEBERSCHRITTEN -- kommt sobald er abgelaufen ist

FEHLER_ZUG_UNMOEGLICH -- Wenn undefinierter Zugfehler auftritt

6.1.3. Aktionen die kommen und gehen

Syntax:

Aktion(Aktionskarte,Spielerfarbe,x1,y1,x2,y2) entspricht:

```
AKTION <aktuelle Aktion>
KARTE <Aktionskarte>     -- komplette zweistellige Nummer
FARBE <Spielerfarbe>
X <x1>
Y <y1>
X <x2>
Y <y2>
```

6.1.3.1. NEHME_PUNKT

Beschreibung: Verwende Aktionspunkt um den Punktestand zu erhöhen

Zu Senden: Aktion()

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe)

Fehler:

FEHLER_ZU_WENIG_AKTIONSPUNKTE

6.1.3.2.SETZE_RITTER

Beschreibung: Setze einen neuen Ritter ein

Zu Senden: Aktion(x,y);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1);

Fehler:

FEHLER_ZU_WENIG_AKTIONSPUNKTE

FEHLER_ZU_HOCH

FEHLER_FIGUR_VORHANDEN

FEHLER_UNBENACHBART

6.1.3.3.SETZE_BAUSTEIN

Beschreibung: Setze einen Baustein ein

Zu Senden: Aktion(x,y);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x,y);

Fehler:

FEHLER_ZU_WENIG_AKTIONSPUNKTE

FEHLER_HAT_ZUWENIG_BAUSTEINE

FEHLER_BURGEIGENSCHAFT_VERLETZT

FEHLER_BURG_NEUBAU

6.1.3.4.SETZE_KOENIG

Beschreibung: Versetze den König (nur nach Aufforderung)

Zu Senden: Aktion(x,y);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x,y);

Fehler:

FEHLER_FIGUR_VORHANDEN

6.1.3.5.ZIEHE_RITTER

Beschreibung: Ziehe einen Ritter für einen Aktionspunkt

Zu Senden: Aktion(x1,y1,x2,y2);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1,x2,y2);

Fehler:

FEHLER_ZU_WENIG_AKTIONSPUNKTE

FEHLER_ZU_HOCH

FEHLER_FIGUR_VORHANDEN

FEHLER_KEINE_FIGUR

FEHLER_ZUG_NICHT_GERADEAUS

FEHLER_ZUG_ZU_LANG

FEHLER_ZUG_DURCH_BURG_UNMOEGLICH

6.1.3.6.ZIEHE_KARTE

Beschreibung: Ziehe eine Aktionskarte

Zu Senden: Aktion();

Extra Bestätigung: KARTE <Kartennummer>

Bestätigung an Alle: Aktion(Spielerfarbe);

Fehler:

FEHLER_ZU_WENIG_AKTIONSPUNKTE

FEHLER_KEINE_KARTE_VORHANDEN

FEHLER_NUR_ZWEI_KARTEN

6.1.3.7.VERSETZE_BAUSTEIN

Beschreibung: Versetze einen Baustein (AK)

Zu Senden: Aktion(Karte,x1,y1,x2,y2);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1,x2,y2);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_BURGEIGENSCHAFT_VERLETZT

FEHLER_BURG_WIRD_GETRENNT

FEHLER_BURGEN_VERBUNDEN

FEHLER_BURG_NEUBAU

FEHLER_ZU_WENIG_BURGEN

6.1.3.8.ZIEHE_DIAGONAL

Beschreibung: Ziehe einen Ritter Diagonal (AK)

Zu Senden: Aktion(Karte,x1,y1,x2,y2);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1,x2,y2);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_ZU_HOCH

FEHLER_FIGUR_VORHANDEN

FEHLER_KEINE_FIGUR

FEHLER_ZUG_ZU_LANG

6.1.3.9.ZIEHE_DURCHTOR

Beschreibung: Ziehe in einer Burg auch aufwärts (AK)

Zu Senden: Aktion(Karte,x1,y1,x2,y2);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1,x2,y2);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_FIGUR_VORHANDEN

FEHLER_KEINE_FIGUR

FEHLER_ZUG_NICHT_GERADEAUS

FEHLER_ZUG_ZU_LANG

FEHLER_ZUG_DURCH_BURG_UNMOEGLICH

6.1.3.10. UNTERSCHIEBE_BAUSTEIN

Beschreibung: Setze einen Baustein unter einen Ritter (AK)

Zu Senden: Aktion(Karte,x,y);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x,y);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_HAT_ZUWENIG_BAUSTEINE

FEHLER_ZUG_UNMOEGLICH

FEHLER_BURGEIGENSCHAFT_VERLETZT

6.1.3.11. VERSETZE_RITTER

Beschreibung: Versetze einen eigenen Ritter an eine andere Position (AK)

Zu Senden: Aktion(Karte,x1,y1,x2,y2);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1,x2,y2);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_FIGUR_VORHANDEN

FEHLER_KEINE_FIGUR

6.1.3.12. ZIEHE_ZWEIEBENEN

Beschreibung: Ziehe mit einem Ritter zwei Stufen nach oben (AK)

Zu Senden: Aktion(Karte,x1,y1,x2,y2);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1,x2,y2);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_ZU_HOCH

FEHLER_FIGUR_VORHANDEN

FEHLER_KEINE_FIGUR

FEHLER_ZUG_NICHT_GERADEAUS

FEHLER_ZUG_ZU_LANG

6.1.3.13. ZIEHE_UEBER_RITTER

Beschreibung: Springe mit einem Ritter über ein Feld auf dem sich ein anderer befindet (AK)

Zu Senden: Aktion(Karte,x1,y1,x2,y2);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x1,y1,x2,y2);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_ZU_HOCH

FEHLER_FIGUR_VORHANDEN

FEHLER_KEINE_FIGUR

FEHLER_ZUG_NICHT_GERADEAUS

FEHLER_ZUG_ZU_LANG

6.1.3.14. SETZE_BAUSTEIN_ALLGEMEIN

Beschreibung: Setze Baustein aus allgemeinen Vorrat (AK)

Zu Senden: Aktion(Karte,x,y);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x,y);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

FEHLER_FIGUR_VORHANDEN

FEHLER_BURGEIGENSCHAFT_VERLETZT

FEHLER_BURG_NEUBAU

6.1.3.15. SETZE_RITTER_ERSTESMAL

Beschreibung: Erstemal den Ritter setzen (Nur nach Aufforderung bei Spielbeginn)

Zu Senden: Aktion(x,y);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe,x,y);

Fehler:

FEHLER_FIGUR_VORHANDEN

6.1.3.16. SECHS_AKTIONSPUNKTE

Beschreibung: Spiele die Karte die dir diese Runde Sechs Aktionspunkte gibt (AK)

Zu Senden: Aktion(Karte);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

6.1.3.17. SIEBEN_AKTIONSPUNKTE

Beschreibung: Spiele die Karte die dir diese Runde Sieben Aktionspunkte gibt (AK)

Zu Senden: Aktion(Karte);

Extra Bestätigung: Keine

Bestätigung an Alle: Aktion(Spielerfarbe);

Fehler:

FEHLER_HAT_AKTIONSKARTE_NICHT

6.1.4. Konstanten für Aktionskarten :

KARTE_VERSETZE_BAUSTEIN = 0

KARTE_ZIEHE_DIAGONAL = 1

KARTE_UNTERSCHIEBE_BAUSTEIN = 2

KARTE_VERSETZE_RITTER = 3

KARTE_ZIEHE_ZWEIEBENEN = 4

KARTE_ZIEHE_UEBER_RITTER = 5

KARTE_SETZE_BAUSTEIN_ALLGEMEIN = 6

KARTE_ZIEHE_DURCHTOR = 7

KARTE_SECHS_AKTIONSPUNKTE = 8

KARTE_SIEBEN_AKTIONSPUNKTE = 9

Die Kartennummer bekommt dann noch die Farbe, so dass sich Nummern zwischen 0 und 39 ergeben.

6.1.5. Weitere Nachrichten:

6.1.5.1.Nachrichten:

Diese beschreiben beliebige Textnachrichten zum Chatten, aber auch damit der Server gewisse Informationen verbreiten kann, wie zum Beispiel die Anmeldung eines neuen Spielers oder ähnliches.

Zu Senden:

NACHRICHT <Text>

Bestätigung an Alle:

NACHRICHT <Spielerfarbe> <Text>

Fehler: Keine

6.1.5.2.Statusabfrage:

Dient dazu den aktuellen Spielstatus zu erfahren.

Zu senden:

STATUS

Bestätigung:

STATUS <Spielname> {WARTET|GESTARTET|BEENDET}

PHASE <aktuelle Phase>

RUNDE <aktuelle Runde>

ZUG <aktueller Zug>

ACTIVE <Spielername der am Zug ist>

TIMEOUT <Länge des Timeouts>

SPIELER <Anzahl Mitspieler>

<Spielernummer> <Spielername> <Spielerpunkte> <Spieleralter> <Burgen für Zug>

<Spielernummer> <Spielername> <Spielerpunkte> <Spieleralter> <Burgen für Zug>

...

ZUSCHAUER

<ZuschauerName>

...

STATUSENDE

Fehler: Keine

6.1.5.3.Spielplanabfrage:

Dient dazu das aktuelle Spielfeld anzuzeigen.

Zu Senden:

PLAN

Bestätigung:

Die inneren Zahlen stehen für die Höhe, steht daneben ein Buchstabe, dann sagt dieser aus, was für ein Ritter (Orange, Blau,...) oder König draufsteht.

PLAN <Breite der quadratischen Matrix>

	0	1	2	3	4	5	6	7
0	0	0	0	0	1	0	0	0
1	0	0	1K	0	0	0	0	0
2	0	0	1B	0	0	1O	1	0
3	1O	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	1
5	0	0	1	0	0	1	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0

PLANENDE

Fehler: FEHLER_SPIEL_LAEUFT_NICHT

6.1.6. Konstanten für Fehler:

FEHLER_ZU_WENIG_AKTIONSPUNKTE	1000
FEHLER_HAT_AKTIONSKARTE_NICHT	1001
FEHLER_HAT_ZUWENIG_BAUSTEINE	1002
FEHLER_ZUG_UNMOEGLICH	1003
FEHLER_NICHT_IMPLEMENTIERT	1004
FEHLER_ZU_HOCH	1005
FEHLER_FIGUR_VORHANDEN	1006
FEHLER_UNBENACHBART	1007
FEHLER_KEINE_FIGUR	1008
FEHLER_BURGEIGENSCHAFT_VERLETZT	1009
FEHLER_ZUG_NICHT_GERADEAUS	1010
FEHLER_ZUG_ZU_LANG	1011
FEHLER_BURG_WIRD_GETRENNT	1012
FEHLER_BURG_NEUBAU	1013
FEHLER_ZU_WENIG_BURGEN	1014
FEHLER_KEINE_KARTE_VORHANDEN	1015
FEHLER_NICHT_AM_ZUG	1016
FEHLER_NUR_ZWEI_KARTEN	1017
FEHLER_ZUG_DURCH_BURG_UNMOEGLICH	1018
FEHLER_NICHT_VORHANDEN	1019
FEHLER_NAME_VERGEBEN	1020
FEHLER_TIMEOUT_UEBERSCHRITTEN	1021
FEHLER_UNGUELTIGE_NACHRICHT	1022
FEHLER_ANZAHL_SPIELER	1023
FEHLER_IN_ANMELDUNG	1024
FEHLER_HAT_KEINE_RITTER_MEHR	1025
FEHLER_SPIEL_LAEUFT_NICHT	1026
FEHLER_KARTE_SCHON_GESPIELT	1027

FEHLER_MAX_SPIELANZAHL_ERREICHT	1028
FEHLER_SPIEL_NICHT_VORHANDEN	1029
FEHLER_FALSCHES_PASSWORT	1030
FEHLER_KARTE_PASST_NICHT	1031
FEHLER_BURGEN_VERBUNDEN	1032

6.2 Lizenz

Sämtliche Software, Dokumentation und andere Bestandteile, die die Tutoriengruppe SWP2 im Rahmen des Softwarepraktikums 2002 bei Herrn Dr. Ulrich Kortenkamp erzeugt hat, stehen unter der unten genannten Apache-License. Durch Benutzen der Software oder der Dokumentation werden die darin enthaltenen Lizenzbestimmungen anerkannt.

6.3 Quellenangaben und weitere Verweise

6.3.1. Webseiten

Sun Java Development Kit 1.3: www.java.sun.com

Java ist eine Programmiersprache, die ebenso wie die darin geschriebenen Applikationen auf vielen Plattformen lauffähig ist.

Ant http://jakarta.apache.org/ant

Ant ist ein Tool ähnlich wie Makefiles in C/C++ speziell für Java.

CVS http://www.cvshome.org

Mit dem "Concurrent Versions System" (CVS) kann Sourcecode für Softwareprojekte verwaltet werden.

Junit http://www.junit.org

JUnit ist ein Testframework für Java-Applikationen.

Log4j http://jakarta.apache.org/log4j

Log4J stellt einen ausgefeilten Logging-Mechanismus für Java bereit.

intelliJ http://www.intellij.com/idea/

IntelliJ Idea ist **die** Entwicklungsumgebung für Java.

DEJA.com http://www.deja.com

Deja.com ist das Newsgruppenarchiv von Google. Sinnvolle Lösungen zu Problemen lassen sich hier finden, wenn sie schon einmal aufgetreten sind und e seine bekannte Lösung dafür gibt (Merke. Warum das Rad zweimal erfinden?)

6.3.2. CD