W3Chess
Web- and Turnbased Chess

Studienarbeit von
Tobias Müller

Betreut von
Prof. Dr. Rainer Parchmann

Mai 2003

Inhaltsverzeichnis

1 Einleitung

Das Schachspiel fasziniert die Menschen schon seit Jahrhunderten, es wurde bereits um das Jahr 531 in Persien[1] gespielt.
Vermutlich ist es der Mischung aus einfachen Regeln[2] gepaart mit einer immensen Komplexität zu verdanken, daß es bis heute nahezu unverändert gespielt wird. Nicht zuletzt den Informatiker reizt dieses Spiel als Paradebeispiel für die Unvollkommenheit aller Ansätze von künstlicher Intelligenz. So wurde zwar bereits 1950 von Alan Turing das erste Schachprogramm verfaßt, aber erst am 5. April 1997 gelang es IBM mit "Deep Blue"[3] (1,4 Tonnen pure Rechenleistung) den Schachweltmeister Garry Kasparow zu schlagen.
Und selbst wenn es den ein wenig vom Glanz des "Spiels der Könige" nimmt, liegt es nahe sich die Möglichkeiten des Internets zu Nutze zu machen und weltweit virtuelle Schachpartien auszutragen. Hierzu gibt es weltweit sogenannte ICS, also Internet Chess Server[4], über die sich zwei Spieler verbinden und eine Partie austragen können. Der Nachteil an dieser sicherlich sehr schönen Art weltweit Schach zu spielen ist vor allen Dingen der Zeitfaktor.
Ein normales Spiele dauert nicht unter einer Stunde. Diese Zeit muß man in der heutigen schnellebigen Zeit erst einmal aufbringen und außerdem muß man während der gesamten Partie im Internet sein. Dies bedeutet nicht zuletzt einen hohen Kostenfaktor.
Daher kam man schon sehr früh auch auf die Idee per EMail Schach zu spielen. Jeder Spieler hat bei sich ein Brett stehen und setzt jeweils den eigenen und den vom Gegner erhaltenen Zug.
Dieses Verfahren eliminiert zwar das Zeit- und das Kostenproblem, jedoch geschieht es recht häufig, daß die Bretter der Gegenspieler durch Unachtsamkeit oder Fehler inkonsistent werden.
Die Lösung ist also ein Web- und Mail basiertes Schach.
Die Gegner können per WWW an einem zentralen Ort das aktuelle Schachbrett einsehen, dort ihren Zug tätigen und werden per EMail informiert wenn der Gegner seinen Zug beendet hat. Die Daten werden zentral auf dem WWW-Server gespeichert und können somit nicht mehr inkonsistent werden.

[1] http://misc.traveller.com/chess/history/
[2] http://www.schachbund.de/fideregeln/Fideregeln.htm
[3] http://www.research.ibm.com/deepblue/
[4] siehe auch Seite 40

2 Einführung

2.1 Anforderungen an das Projekt

Davon ausgehend, daß möglichst jeder gegen jeden spielen können soll, muß gewährleistet sein, daß der Webschachserver auf möglichst vielen unterschiedlichen Systemen aufgesetzt werden kann. Dies erreicht man im einfachsten Fall durch Minimalismus. In diesem Fall also durch möglichst wenig Abhängigkeiten, etwa von Interpretern wie PHP oder von Datenbanken.
Aus diesem Grund wurde als Programmiersprache C gewählt, ohne Abhängigkeiten zu fremden Bibliotheken. Das Programm wurde mit dem freien GNU-C-Compiler[5] entwickelt, welcher für alle gängigen Plattformen erhältlich ist. Es sollte sich (eventuell nach kleinen Anpassungen) aber auch mit jedem anderen C-Compiler übersetzen lassen.
Da es sich nach der Übersetzung um ein native Binary handelt, benötigt der Server zudem wenig Systemressourcen und sollte auch auf älteren Rechnern problemlos funktionieren und sehr schnell laufen.
Die eingesparte Datenbank-Anbindung macht sich unwesentlich in etwas mehr Aufwand bei der Speicherung der Schachpartien bemerkbar. Ich verwende hierzu je eine Textdatei pro Spiel.

Auf der Client- also in diesem Fall auf der Browserseite ist die Vielzahl der sich am Markt befindlichen WebBrowser zu beachten. Für alle Tests wurde Mozilla[6] benutzt, aber der Spieler sollte theoretisch den Browser seiner Wahl benutzen können. Insbesondere dürfen also keine Funktionen wie Style-Sheets oder JavaScript zur Anwendung kommen. Dies macht sich im Spielverlauf zwar durch einige fehlende "Schönheiten" (wie das wirkliche "Ziehen" einer Figur per Drag and Drop) bemerkbar, aber dafür wurde W3Chess schon erfolgreich selbst in Textbrowsern und im Handy des Autors getestet. Einer Benutzung über einen PDA im WLan steht also nichts im Wege.
Die einzige Anforderung an den Browser bleibt die Darstellung von Tabellen und an Browser und WebServer gleichermaßen die Möglichkeit des Umgangs mit CGI-Anfragen und Formularen.
Insbesondere genügt damit für den WebServer eine Grundinstallation ohne Erweiterungen wie Scriptsprachen jedweder Art.
Ein einigermaßen moderner Browser und WebServer sollte damit als Grundvoraussetzung zur Benutzung von W3Chess völlig ausreichen.

Ein weiterer wesentlicher Aspekt ist die Sicherheit.
Hier betrachte ich Schach als das was es ist, ein Spiel. Wenn also jemand durch geschickte CGI-Anfragen den Datenbestand korumpiert, so ist dies als falsch spielen anzusehen und mehr nicht.
Daher wurde zwar bei der Programmierung großer Wert darauf gelegt, daß insbesondere keine Daten auf dem WebServer in Gefahr sind, jedoch ist nicht mit absoluter Sicher-

[5] http://gcc.gnu.org/
[6] http://www.mozilla.org/

heit die Manipulation eines einzelnen Spiels durch geschickte CGI-Anfragen vollkommen ausgeschlossen. Im Rahmen der Selbstkontrolle ist dies durchaus zulässig und nach den Schachregeln wird ein Fehlverhalten zum Abbruch des Spiels führen.

2.2 Die umgesetzten Schachregeln der Fide

Schon durch die Art des Spielens ist es nicht möglich alle Schachregeln[7] zu implementieren, als Beispiel sei darauf verwiesen, daß ein Zug mit "einer Hand alleine" ausgeführt werden muß , es also im übertragenen Sinne verboten ist die Maus mit beiden Händen zu benutzen oder aber die Hand zu wechseln.
Ich beschränke mich also auf die wesentlichen Schachregeln der Fide[8]. Diese wird W3Chess erkennen und verbotene Züge nicht zulassen.
Folgende Regeln werden von W3Chess umgesetzt:

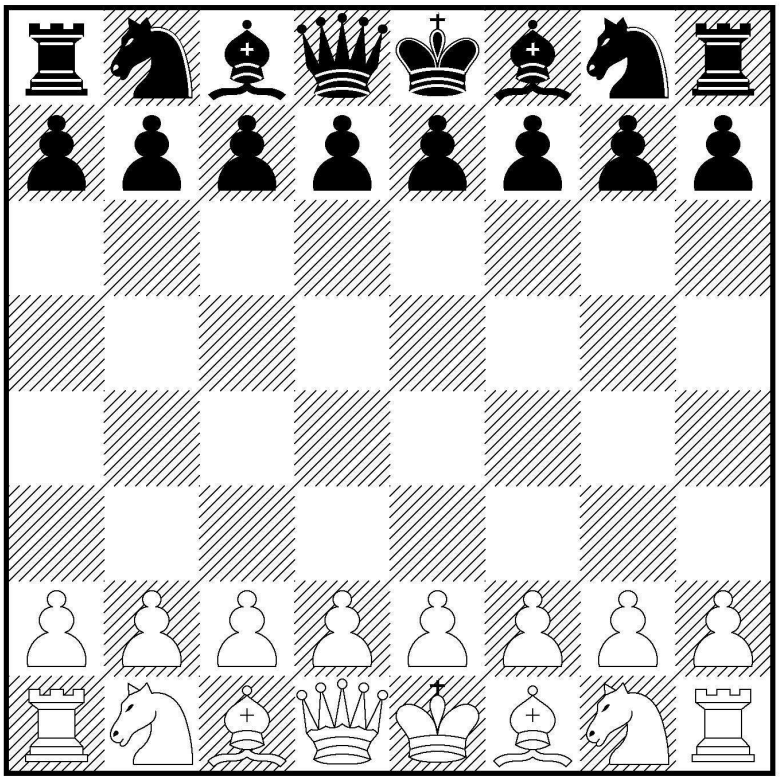
- Wesen und Ziele des Schachspiels
  - Das Schachspiel wird zwischen zwei Gegnern gespielt, die abwechselnd ihre Figuren auf einem quadratischen Spielbrett, "Schachbrett" genannt, ziehen. Der Spieler mit den weißen Figuren beginnt die Partie. Ein Spieler "ist am Zug", sobald der Zug seines Gegners ausgeführt worden ist.
  - Das Ziel eines jeden Spielers ist es, den gegnerischen König so "anzugreifen", daß der Gegner keinen regelgemäßen Zug zur Verfügung hat, der ein "Schlagen" des Königs im folgenden Zug vermeiden würde. Der Spieler, der dieses Ziel erreicht, hat den gegnerischen König "mattgesetzt" und das Spiel gewonnen. Der Gegner, dessen König mattgesetzt worden ist, hat das Spiel verloren.
- Die Anfangsstellung der Figuren auf dem Schachbrett
  - Das Schachbrett besteht aus einem 8 x 8 Gitter von 64 gleich großen Quadraten, die abwechselnd hell und dunkel sind (die "weißen" und die "schwarzen Felder"). Das Schachbrett wird so zwischen die beiden Spieler gelegt, daß auf der Seite vor einem Spieler das rechte Eckfeld weiß ist.

[7] http://www.schachbund.de/fideregeln/Fideregeln.htm
[8] Fédération Internationale des Échecs: Weltschachbund, http://www.fide.com/

– Zu Beginn der Partie hat der eine Spieler 16 helle (“weiße”), der andere 16 dunkle (“schwarze”) Figuren. Diese Figuren sind die folgenden:

 ein weißer König	 eine weiße Dame	 zwei weiße Türme	 zwei weiße Läufer
 zwei weiße Springer	 acht weiße Bauern	 ein schwarzer König	 eine schwarze Dame
 zwei schwarze Türme	 zwei schwarze Läufer	 zwei schwarze Springer	 acht schwarze Bauern

– Die Anfangsstellung der Figuren auf dem Schachbrett ist die folgende:



- Die acht senkrechten Spalten von Feldern heißen “Linien”, die acht waagerechten Zeilen von Feldern heißen “Reihen”. Eine geradlinige Folge von Feldern gleicher Farbe, die sich jeweils an den Ecken berühren, heißt “Diagonale”.

- Die Gangarten der Figuren

  - Es ist nicht gestattet, eine Figur auf ein Feld zu ziehen, das bereits von einer Figur der gleichen Farbe besetzt ist. Wenn eine Figur auf ein Feld zieht, das von einer gegnerischen Figur besetzt ist, wird letztere geschlagen und als Teil desselben Zuges vom Schachbrett entfernt. Eine Figur greift eine gegnerische Figur an, wenn sie auf jenem Feld schlagen könnte.
  - Der Läufer darf auf ein beliebiges anderes Feld entlang einer der Diagonalen ziehen, auf welchen er steht.
  - Der Turm darf auf ein beliebiges anderes Feld entlang der Linie oder der Reihe ziehen, auf welcher er steht.
  - Die Dame darf auf ein beliebiges anderes Feld entlang der Linie, der Reihe oder einer der Diagonalen ziehen, auf welchen sie steht.
  - Beim Ausführen dieser Züge dürfen Dame, Turm und Läufer nicht über dazwischenstehende Figuren hinwegziehen.
  - Der Springer darf auf eines der Felder ziehen, die seinem Standfeld am nächsten, aber nicht auf gleicher Linie, Reihe oder Diagonalen mit diesem liegen.
  - a) Der Bauer darf vorwärts auf das unbesetzte Feld direkt vor ihm auf derselben Linie ziehen oder
    b) in seinem ersten Zug entweder wie unter a) beschrieben ziehen oder um zwei Felder entlang derselben Linie vorrücken, vorausgesetzt, daß beide Felder frei sind, oder
    c) auf ein von einer gegnerischen Figur besetztes Feld diagonal vor ihm auf einer benachbarten Linie ziehen, indem er die Figur schlägt.
    d) Ein Bauer, der ein Feld angreift, das von einem gegnerischen Bauern überschritten worden ist, der von seinem Ursprungsfeld aus in einem Zug um zwei Felder vorgerückt ist, darf diesen gegnerischen Bauern so schlagen, als ob letzterer nur um ein Feld vorgerückt wäre. Dieses Schlagen darf nur in dem Zug geschehen, der auf ein solches Vorrücken folgt und wird “Schlagen en passant” genannt.
    e) Sobald ein Bauer diejenige Reihe erreicht hat, die am weitesten von seinem Ursprungsfeld entfernt ist, muss er als Teil desselben Zuges gegen eine Dame, einen Turm, einen Läufer oder einen Springer derselben Farbe ausgetauscht werden. Die Auswahl des Spielers ist nicht auf bereits geschlagene Figuren beschränkt. Dieser Austausch eines Bauern für eine andere Figur wird “Umwandlung” genannt, und die Wirkung der neuen Figur tritt sofort ein.

- Es gibt zwei verschiedene Arten den König zu ziehen:
  i) er zieht auf ein beliebiges angrenzendes Feld, das nicht von einer oder mehreren gegnerischen Figuren angegriffen wird.
  Von den gegnerischen Figuren gilt, daß sie ein Feld auch dann angreifen, wenn sie selbst nicht ziehen können.
  ii) er "rochiert". Die "Rochade" ist ein Zug des Königs und eines gleichfarbigen Turmes auf der gleichen Reihe. Sie gilt als ein Zug und wird folgendermaßen ausgeführt: Der König wird von seinem Ursprungsfeld um zwei Felder in Richtung des Turmes hin versetzt, dann wird dieser Turm auf das Feld gesetzt, das der König soeben überquert hat.
    1) Die Rochade ist regelwidrig:
    a) wenn der König bereits gezogen hat, oder
    b) mit einem Turm, der bereits gezogen hat.
    2) Die Rochade ist vorübergehend verhindert,
    a) wenn das Standfeld des Königs oder das Feld, das er überqueren muss, oder sein Zielfeld von einer oder mehreren gegenerischen Figuren angegriffen wird,
    b) wenn sich zwischen dem König und dem Turm, mit dem rochiert werden soll, irgendeine Figur befindet.
- Ein König "steht im Schach", wenn er von einer oder mehreren gegnerischen Figuren angegriffen wird, auch wenn diese selbst nicht ziehen können. Das Ansagen eines Schachgebotes ist nicht erforderlich.
- Keine Figur darf einen Zug machen, welcher den eigenen König einem Schachgebot aussetzt oder ihn in einem Schachgebot stehen läßt.

- Die Beendigung der Partie
  - Die Partie ist von dem Spieler gewonnen, der den gegnerischen König "matt" gesetzt hat. Damit ist die Partie sofort beendet, vorausgesetzt, daß der Zug, der die Mattstellung herbeigeführt hat, regelgemäß war.
  - Die Partie ist von dem Spieler gewonnen, dessen Gegner erklärt, daß er aufgebe. Damit ist die Partie sofort beendet.
  - Die Partie ist "remis", also unentschieden, durch eine von den beiden Spielern während der Partie getroffene Übereinkunft. Damit ist die Partie sofort beendet

2.3 Erweiterte Funktionen

Zusätzlich zu den Schachregeln muß ein WebSchach auch die folgenden Funktionen implementieren:

- Die Spielzustände “Schach” und “Schachmatt” sollten erkannt werden
- Wenn der Benutzer vergißt, welche Spiele er gerade spielt, muß ihm eine Liste aller seiner Spiele zugeschickt werden
- Sollte ein Benutzer eine Mail verlieren, so kann er sie sich erneut zusenden lassen
- Ein Benutzer sollte in der Lage sein seine EMail-Adresse zu ändern
- Ein Benutzer ohne Gegner sollte sich trotzdem eintragen und auf einen Gegner warten können
- Durch das Setzen einer Figur auf ihren aktuellen Standort sollte der Zug abgebrochen werden
- Damit ein Spieler nur dann einen Zug machen kann wenn er an der Reihe ist, sollte er für jeden Zug ein Paßwort zugesendet bekommen, durch welches seine Zugberechtigung sichergestellt wird.
- Alte Spiele müssen automatisch gelöscht werden.

3 Umsetzung

3.1 Funktionsweise von CGI-Skripten

CGI ist die Abkürzung für "Common Gateway Interface", also
"Allgemeine Vermittlungsrechner-Schnittstelle". Es stellt eine Möglichkeit dar, aus einer Web-Seite heraus ein nahezu beliebiges Programm auf dem Server-Rechner auszuführen. Dabei gibt das CGI-Script selbst die anzuzeigenden Daten aus.
Da bei dieser Form der Seitengenerierung jegliche Programmausführung auf dem WebServer geschieht und der Webbrowser nur die erzeugten Daten anzeigen muß , entstehen hieraus keinerlei spezielle Anforderungen an den Browser, wie es etwa bei JavaScript oder TclTk-Applets.

Technische funktioniert es folgender Maßen:
In eine beliebige (eventuell auch schon generierte) HTML-Seite werden spezielle Formular-Tags eingebunden, z.B.:

```
<HTML>
<FORM action="http://www.sirtobi.sir/cgi-bin/helloworld.cgi" method=post>
<INPUT type="hidden" name="GEHEIM" value="Secret">
<INPUT type="text" name="MESSAGE" value="Hello, world!">
<INPUT type="submit">
</FORM>
</HTML>
```

Dieses Beispiel würde zu einem solchen Formular führen:

Der Benutzer kann nun einen beliebigen Text eingeben (z.B. "Ich bin es") und diesen durch betätigen der Schaltfläche "abschicken".
In diesem Beispiel würde nun das Script
`http://www.sirtobi.sir/cgi-bin/helloworld.cgi` ausgeführt werden.

Die einzige Anforderung an dieses Script ist, daß es auf dem Webserver lauffähig ist, dabei spielt es keine Rolle ob es sich um ein Shell-Skript, ein Perl-Skript oder wie bei W3Chess um ein Binary handelt.

Damit es zu einer wirklichen Interaktion mit dem Benutzer kommen kann, müssen dem Skript selbstverständlich Parameter übergeben werden können. Dies geschieht je nach Aufruf im `FORM`-Tag über die Standardeingabe (`method=post`) oder durch das Setzen der Umgebungsvariable `QUERY_STRING` (`method=get`).
Das CGI-Script erhält einen String von "`NAME=WERT`"-Paaren, welche durch '&'-Zeichen voneinander getrennt sind. Der Name wird im HTML-Quellcode gesetzt (oben z.B. `GEHEIM` und `MESSAGE`), der Wert ist entweder der bei der Erstellung vordefinierte oder ein vom Benutzer eingestellter Wert. Zu Beachten ist, daß Sonderzeichen als ihr hexadezimaler zweistelliger Ascii-Wert mit vorangestelltem '%'-Zeichen dargestellt werden.

Im oberen Beispiel erhielte das Script `helloworld.cgi` also auf der Standardeingabe den String "`GEHEIM=Secret&MESSAGE=Ich%20bin%20es`" (beachte: 20 ist der hexadezimale Ascii-Wert des Leerzeichens).

Das Skript kann nun die Formulardaten interpretieren und daraus eine WebSeite erstellen.
Auf der Standardausgabe muß es hierzu zuerst den Mime-Type des erzeugten Dokuments in der Form "`Content-type: text/html`" (`text/html` kann je nach Anwendung auch ein beliebiger andere Mime-Type sein, z.B: auch `image/jpeg`) ausgegeben, danach eine Leerzeile und anschließend den eigentlichen Inhalt der generierten Seite.
Der WebServer reicht nun diese Seite wie eine "normale", also statische Datei an den Browser weiter.

Es gibt eine Vielzahl von möglichen Formularfeldern mit diversen Parametern. Eine genau Beschreibung kann man z.B. bei SelfHTML[9] erhalten.
W3Chess benutzt die folgenden Feldertypen:

`hidden`	Versteckte Angaben, also solche welche der Benutzer nicht sieht (z.B. `ACTION`, siehe unten)
`image`	Bilder, welche beim Anklicken wie "normale" links wirken (z.B. die Schachfiguren)
`password`	Textfelder bei denen man den eingegebenen Text nicht direkt sieht (für Paßworteingaben)
`text`	Texteingaben (z.B. für die Kurznachrichten)
`radio`	Radiobuttons, also eine Auswahl eines Elements aus einer Liste von Möglichkeiten (z.B. für die speziellen Aktionen wie "Aufgabe" oder "Logout")
`submit`	"Absende"-Schaltflächen

Will man einem Script direkt Daten übergeben ohne das ein Formular ausgefüllt werden muß , so kann dies auch geschehen indem das Script direkt aufgerufen und der Übergabeparameter mit einem '?' an den Dateinamen gehängt wird. Im obigen Beispiel also z.B.

```
http://www.sirtobi.sir/cgi-bin/
                        helloworld.cgi?GEHEIM=Secret&MESSAGE=Ich%20bin%20es
```

Hilfreich ist dies besonders dann, wenn man vordefinierte Funktionen des Scripts über einen "normalen" Link, also einen `<A HREF>`-Tag auslösen möchte.
In diesem Fall wird der Teilstring ab dem '?' direkt in der Umgebungsvariable `QUERY_STRING` übergeben.

[9] http://selfhtml.teamone.de/

Es wird ferner eine Reihe weiterer Umgebungsvariablen gesetzt auf die das Script zugreifen kann. Diese enthalten Informationen über den verwendeten Browser und den Rechner des Benutzers, den WebServer und vieles mehr.
In W3Chess werden die folgenden Variablen benutzt:

`SCRIPT_NAME`	Der relative Pfad des CGI-Scripts auf dem WebServer
`SERVER_NAME`	Der Name des Webservers
`CONTENT_LENGTH`	Die Anzahl der Daten in der Standardeingabe bei Verwendung von `method=post`
`QUERY_STRING`	Die Übergabeparameter bei Verwendung von `method=get`

3.2 Verwendung von CGI in W3Chess

In W3Chess werden CGI-Skripte neben der "normalen" Verwendung als Forumlarauswertung, vor allen Dingen verwendet um den eigentlichen Zug einzugeben.
Hierzu wird das Spielbrett in HTML als Tabelle innerhalb einer `FORM`-Umgebung definiert und jedes Feld, bzw. jede Figur ist als `<INPUT TYPE=image>`-Tag dargestellt.
In modernen Browsern kann man in diesem Tag, wie in allen Form-Elementen, ein "Name"- und einen "Value"-Argument übergeben. Gibt man also allen Elemente den gleichen Namen zusammen mit einem Wert welcher das Feld repräsentiert, z.B. A4, so könnte man in W3Chess nach der CGI-Variablen dieses Namens suchen und über den gesetzten Wert auf das ausgewählte Feld schließen.
Leider beherrschen einige ältere Browser, insbesondere auch der Netscape Navigator 4, welcher bei einigen Betriebssystemen bis heute ausgeliefert wird[10] das "Value"-Argument nicht.
Aus diesem Grund ist jedes Feld mit dem Namen "MOVE=XX" und einem Pseudo-Value benannt, also z.B. "`NAME=MOVE=A4`" und "`VALUE=TUX`". Durch die Codierung von Name-/Werte-Paaren im CGI-String (s. oben), also in diesem Fall MOVE=A4=TUX, kann nun auch bei älteren Browsern das Extrahieren des ausgewählten Feldes wie oben beschrieben geschehen.
Ferner ist zu beachten, daß Felder welche vom Benutzer nicht ausgewählt werden dürfen, also insbesondere Figuren welche nicht bewegt werden können, durch normale `<IMG>`- anstelle der `<INPUT>`-Tags dargestellt werden.
Auch Leerfelder werden mit einem leeren Feld belegt, da der Breiten- und Höhengaben von Tabellenspalten ebenfalls nicht von jedem Browser unterstützt werden.

Eine weitere Anwendung von der W3Chess in großem Umfang gebraucht macht, sind die `hidden`-Tags.
Im Prinzip könnte man das Programm als eine Sammlung einzelner CGI-Skripte erstellen. Hieraus würde ein funktioneller Overhead entstehen, da einige Funktionen wie z.B. `query2String` oder `readgamedata` in allen Skripten integriert werden müßten. Selbst bei Verwendung von dynamischen Libraries wäre aufgrund der geringen Größe von W3Chess schon die Größe des Headers eines jeden Binaries nicht mehr zu vernachlässigen. Ein gu-

[10] z.B. Solaris 9, http://www.sun.com/

tes Beispiel für die Richtigkeit dieser These ist z.B. das bekannte Programm BusyBox[11] aus der UniX-Welt.
W3Chess benutzt nun mit Hilfe von CGI einen ähnlichen Mechanismus.
Über den `hidden`-Tag "`ACTION`" wird mitgeteilt, wie sich das Programm "im Groben" verhalten soll und dieses Verhalten durch einen entsprechenden Aufruf in der Funktion `main` erzeugt:

ACTION=	**Funktion**	**Verhalten**
<leer>	start	Aufbau der Loginseite
NEW	newgame	Anlegen eines neuen Spiels
RESUME	resumegame	Ein Spiel fortsetzen
MOVED	piecemoved	Einen Zug beenden
CHMAIL	changemail	Die Emailadresse ändern
JOIN	joingame	An einem "offenen" Spiel teilnehmen
SENDGAMES	sendgames	Dem Spieler seine Spiele zusenden
<sonst>	start	Aufbau der Loginseite

Zu beachten ist dabei, daß in jedem Fall vor dem eigentlichen Aufruf der Funktion die Routine `header` und nach dem Verlassen der Funktion die Routine `footer` gestartet wird, um den einheitlichen Kopf und Fuß der generierten HTML-Seite auszugeben.

Will man auf die Abhängigkeit von externen Libraries verzichten, so muß die Auswertung der CGI-Parameter leider zu Fuß erfolgen. Dabei ist zu beachten das

- die Parameter entweder über den `stdin` (CGI-Post) oder als Umgebungsvariable `QUERY-STRING` (CGI-Get) übergeben werden.
- Sonderzeichen als ihr Ascii-Code mit einem vorangestellten '%'-Zeichen codiert werden.

Der Vorteil beim CGI-Post ist, daß man die Übergabevariablen nicht in der URL sehen kann. Dies ist in jedem Fall ästhetisch schöner und bringt auch Sicherheitsvorteile, da nach außen weniger über die Implementierung der Funktionen bekannt wird (zumindest nicht ohne etwas Aufwand).
Der Vorteil beim CGI-Get liegt darin, daß Parameter auch direkt als `<A HREF>`-Tag in HTML übergeben werden können. Je nach Anwendung benutzt W3Chess also beide Varianten und muß daher auch beide einlesen können.

Beim Starten wird also überprüft, ob die Umgebungsvariable `CONTENT_LENGTH` gesetzt und größer 0 ist. In diesem Fall liegt ein CGI-Post vor, der `stdin` wird in einen allocierten Speicherbereich der entsprechenden Länge kopiert und der globale Zeiger `query` auf diesen Speicherbereich gesetzt.
Im anderen Fall wird `query` auf die Umgebungsvariable gelegt. In beiden Fällen kann man nun also mit `query` direkt auf die CGI-Parameter zugreifen.

[11] http://www.busybox.net/

Nach dem Einlesen werden nun alle Sonderzeichen mittels `query2String` wieder zurückgewandelt und der String nach der Variable "ACTION" durchsucht, damit die entsprechende Funktion gestartet werden kann.

3.2.1 Das Bereinigen des CGI-Strings

Das Bereinigen des CGI-Strings geschieht direkt nach dem Einlesen in der Funktion `query2String`:

```
void query2String(char *string) {
  char *z1,*z2;
  /* The string could only get shorter, so we can need no extra space
      allocated */
  z1=string;
  while(*z1 != (char)NULL) {
    while((*z1 != '%') && (*z1 != (char)NULL) &&
          (*z1 != '+') && (*z1 != '<') && (*z1 != '>'))
      z1++;
    if(*z1 == '%') {
      *z1=hexDec(z1+1);
      z2=z1+3;
      while(*z2 != (char)NULL) {
        *(z2-2)=*z2;
        z2++;
      }
      *(z2-2)=(char)NULL;
    }
    if(*z1 == '+')
      *z1=' ';
    if(*z1 == '<')
      *z1='[';
    if(*z1 == '>')
      *z1=']';
  }
}
```

In dieser Funktion wird im durch `string` übergebenen Textfeld jedes Vorkommen von '%' gesucht.
Ist es gefunden, so werden die beiden folgenden Zeichen als Hexadezimaldarstellung eines Sonderzeichens interpretiert, anstelle des '%' in den Text eingefügt und der Rest des Feldes um zwei Zeichen nach links verschoben, z.B.: wird also "`Hallo%20Du`" erst zu "`Hallo 20Du`" und schließlich zu "`Hallo Du`".

Um zu vermeiden, daß ein Spieler in seiner Kurznachricht HTML-Tags einschleusen und damit unter Umständen den Aufbau der WebSeite verhindern kann (z.B. durch einen `</table>`-Tag), werden außerdem alle Vorkommen von '<' und '>' durch '[' und ']' ersetzt.
Ferner werden die zumeist durch '+' anstelle von '%20' maskierten Leerstellen durch solche ersetzt.

Da hierdurch der Text eventuell kürzer, jedoch nie länger wird, kann die Umsetzung direkt im übergebenen Text geschehen und auf ein temporäres Feld verzichtet werden. Die Verwendung eines zweiten Feldes würde zwar rein theoretisch Zeit sparen da man einen lineare Algorithmus verwenden könnte, jedoch ist dieser Zeitgewinn irrelevant, da die umzuwandelnden Textfelder sehr kurz sind.

3.2.2 Das Parsen des CGI-Strings

Das Parsen des CGI-Strings geschieht in jeder Funktion und für jede gesuchte CGI-Variable auf die folgende Weise:

```
  z1=strstr(query,"GEHEIM=");
  if(z1 == (char *)NULL) {
    errorwin(ERRORDEFAULT,"","");
    return;
  }
  z1+=7;
  z2=z1;
  while((*z2 != '&') && (*z2 != (char)NULL))
    z2++;
  uc1=*z2;
  *z2=(char)NULL;

/* An dieser Stelle steht der gesuchte Wert im
     durch NULL beendeten String an der Stelle z1 */

  *z2=uc1;
```

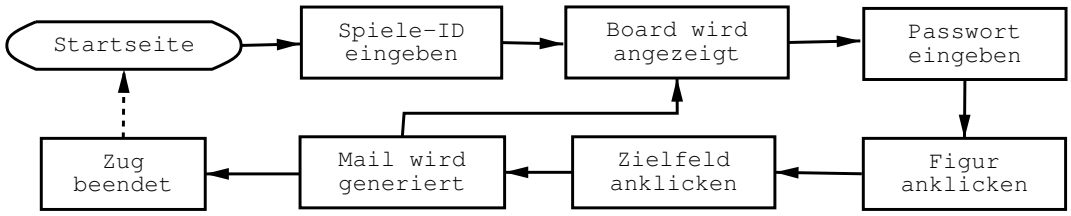
Die CGI-Strings werden wie im Abschnitt "Funktionsweise von CGI-Skripten"[12] beschrieben in der Form "`GEHEIM=Secret&MESSAGE=Ich bin es`" (nach Bereinigung!) übergeben.
Nun wird ein Zeiger auf das erste Vorkommen des gesuchten Variablennamens gesetzt (`z1=strstr(query,''GEHEIM='')`). Sollte die Variable nicht vorkommen aber für die Funktion zwingend erforderlich sein, so wird das Programm mit einer Fehlermeldung (`errorwin`) beendet, ansonsten der erhaltene Zeiger um die Länge der Variable + 1 weitergesetzt, also auf dem Anfang des Variablenwertes.

[12] siehe S.10

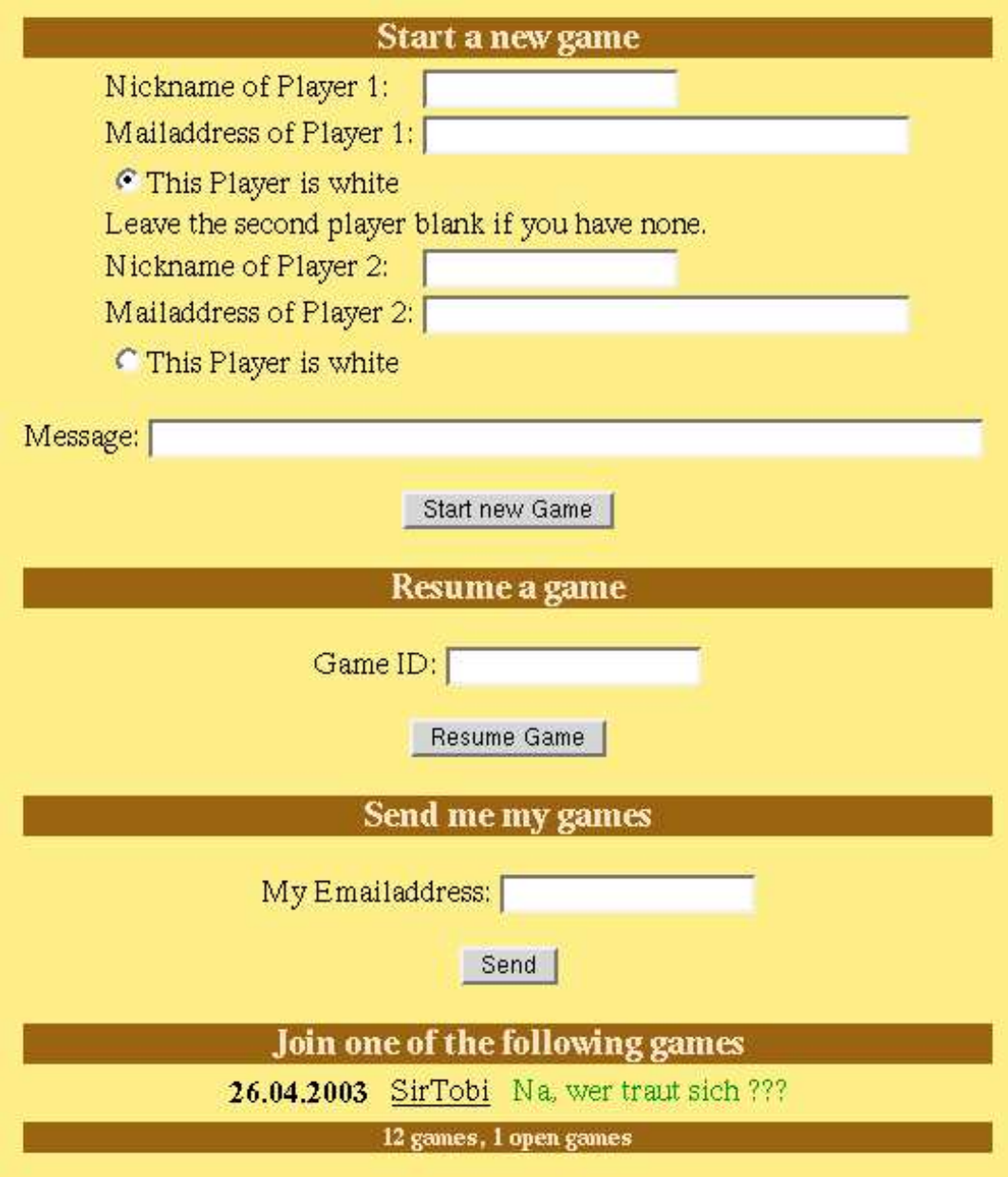
Aber hier wird ein weiterer Zeiger (`z2`) soweit vorwärts bewegt, bis er auf einen CGI-Trenner, also ein '&' oder die die Zeichenkette abschließende `NULL` zeigt. Dort wird das aktuelle Zeichen (also das '&' oder die `NULL`) gesichert und ein `NULL`-Zeichen eingefügt. Nun kann der Variablenwert an der Stelle `z1` als `NULL`-terminierte Zeichenkette eingelesen und z.B. in ein Array kopiert oder mit `atoi` in einen Zahlenwert umgewandelt werden. Nach der Bearbeitung muß der gesicherte Wert an die Stelle `z2` zurückgeschrieben werden, da sonst folgende Argumente verloren gehen könnten.

3.3 Spielablauf

Ein Zug verläuft im Groben immer folgender Maßen:



Startseite:



Beim Aufruf von W3Chess ohne CGI-Parameter wird zunächst ein Standard-Formular generiert mit der Möglichkeit ein neues Spiel anzumelden, ein altes Spiel fortzusetzen oder sich eine Liste von allen Spielen zusenden zu lassen, an denen man zur Zeit teilnimmt.
Gibt man bei der Anmeldung eines neuen Spiels keinen Gegner an, so wird ein "offenes" Spiel erzeugt, welches auf einer Liste ganz unten auf der Startseite erscheint. Hier kann sich ein fremder Spieler ein "offenes Spiel" aussuchen und daran teilnehmen. Durch die-

sen Mechanismus ist es leicht möglich einen fremden Gegner zu finden und mit diesem zu spielen.

Nachdem die Startseite erzeugt wurde, werden alle Spiele gezählt und eine kleine Statistik erstellt. Dabei weden automatisch solche Spiele gelöscht bei denen sich 30 Tage (der Wert ist in der `config.h`[13] durch die Variable `DELETEAFTER` frei änderbar) keine Veränderung gezeigt hat. Ausschlaggebend ist hierbei die Zeit der letzten Dateiänderung (modification time). Durch diesen einfachen Mechanismus ist gewährleistet, daß das System nicht durch eine Flut von beendeten oder verwaisten Dateien volläuft.
Wem ein automatisches Löschen jedoch als Risiko erscheint, der kann durch das Setzen der Variable `ALLOWREMOVE` auf "0" in der `config.h` diese Funktion unterdrücken.

Spiele-ID eingeben:
Jedes Spiel hat eine eindeutige ID welche aus dem aktuellen Datum und der ProzessID von W3Chess generiert wird.
Durch Eingabe dieser ID gelangt der Spieler an sein Board um es entweder noch einmal anzusehen oder den nächsten Zug zu machen.
Um es dem Benutzer leichter zu machen enthält jedoch auch jede Mail die ihm zugesendet wird eine direkte URL zum Brett. Je nach Mailreader kann er so auch direkt aus der Mail heraus "sein" Spielbrett aufrufen.

[13] siehe S.36

Board wird angezeigt:

Aus dem zur Spiele-ID gehörendem Datenfile wird das aktuelle Spielbrett und eine Liste aller vorangegangenen Züge inklusive der geschlagenen Figuren erzeugt.
Während der Generierung wird bestimmt auf welchem Feld eine Figur steht welche der Spieler bewegen kann. In diesem Fall wird das Feld mit einem Link hinterlegt. Auf diese Weise entsteht ein Spielbrett auf dem nur solche Figuren bewegt werden können bei denen es die Schachregeln zulassen.

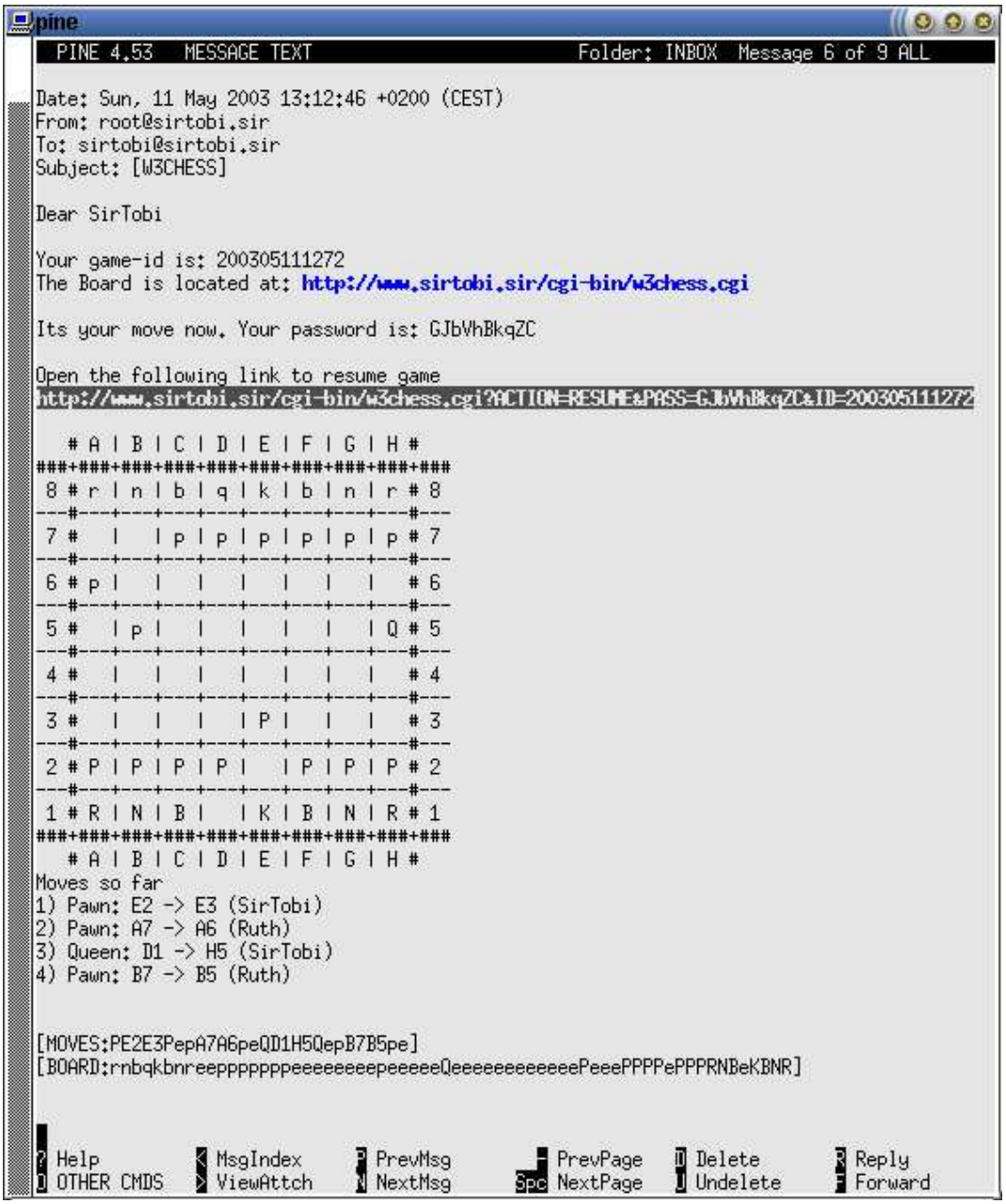
Unterhalb vom Brett befindet sich desweiteren eine Auswahlliste mit Zusatzfunktionen um sich das Paßwort (siehe unten) und das alte Spiel erneut zusenden zu lassen, dem Gegner ein "Remis" anzubieten, aufzugeben oder seine EMailadresse zu ändern. Dabei ist zu beachten das diese Aktionen sich nur von jenem Spieler ausführen lassen, welcher am Zug ist.

Figur anklicken:
Hat der Spieler eine Figur angeklickt, so wird das Brett erneut aufgebaut. Dabei werden dieses Mal solche Felder mit einem Link hinterlegt, auf welche der Benutzer nach den Schachregeln mit der gewählten Figur ziehen darf. Wiederum ist also ein falscher Zug nicht möglich.

Zielfeld anklicken:
Wenn der Spieler das Zielfeld anklickt, so wird zunächst überprüft, ob es sich um dasselbe Feld handelt wie das Startfeld. In diesem Fall wird der Zug abgebrochen.
Ansonsten wird als nächstes getestet ob der Spieler das korrekte Paßwort eingetragen hat und ob der Spezialfall eines Bauerntauschs vorliegt. In diesem Fall wird ein Dialog zur Auswahl der Austauschfigur erzeugt.
Hierbei ist eine Sache zu beachten: Wenn der Spieler noch ein älteres Brett vorliegen hat, so könnte er mit einem ihm zugeschickten neuen Paßwort nun unter unglücklichen Umständen ungültige Züge ausführen. Um dies zu vermeiden wird bei der Übergabe des Zuges die für das angezeigte Brett erfolgte Anzahl von vorangegangenen Zügen übermittelt und mit der momentan gespeicherten Anzahl verglichen. Stimmen diese nicht überein, so liegt ein veraltetes Board vor.

Paßwort eingeben:
Um sicherzustellen, daß der Spieler authorisiert ist den nächsten Zug zu vollziehen wird mit jeder Mail an die Mitspieler ein neues Paßwort generiert und dem betreffenden Spieler zugesandt. Diesen aus 10 alphanumerischen Zeichen bestehenden Code muß der Spieler vor Beendigung seines Zuges eingeben.
Von einem immer gleich bleibenden Paßwort für jeden Spieler wurde bei der Implementierung von W3Chess abgesehen, da die Gefahr besteht das dieses z.B. in den Logfiles des WebServers oder über einen einfach Proxy ausgespäht werden kann.
Sollte der Spieler über den in der generierten Mail enthaltenen Hyperlink zum Spielbrett gelangt sein, so ist das Paßwort automatisch gesetzt.

Mail wird generiert:



Bei erfolgreichem Abschluß eines Zuges wird nun das geänderte Board mit der erweiterten Liste der schon erfolgten Züge gespeichert, ein neues Paßwort wird generiert und die Mails an die Mitspieler abgeschickt.
Abschließend wird zu “Board wird angezeigt” verzweigt.

Zug beendet:
Der Zug ist beendet, die Mails sind geschrieben. Nun kann der Spieler sich ausloggen und warten bis er an der Reihe ist den nächsten Zug zu machen.

3.4 Implementierung der Regeln

Die Implementierung der Schachregeln geschieht zum Großteil in der Funktion `canmove`. Hier wird für ein gegebenes Feld versucht die dort stehende Figur regelkonform in eine beliebige (der Zugweise der Figur entsprechenden) Richtung zu bewegen.
Wurden zwei Felder übergeben, so wird versucht die Figur in einem Zug den Regeln entsprechend vom ersten zum zweiten Feld zu transferieren.
Besondere Beachtung gilt hierbei den Ausnahmeregelungen:

- Soll ein Bauer bewegt werden und steht dieser zwei Felder vor seiner Ausgangsposition so muß anhand der Liste der bereits abgeschlossenen Züge überprüft werden ob ein "en Passant" vorliegt, steht der Bauer noch in der Ausgangsposition so müssen Sprünge um ein oder zwei Felder erlaubt sein.
  Steht auf dem Startfeld ein Bauer und ist das Zielfeld eine gegnerische Figur so ist ferner ein schräger Zug erlaubt.

- Soll der König von seiner Ausgangsposition zwei Felder nach links oder nach rechts bewegt werden so könnte eine Rochade vorliegen.
  Hierfür muß anhand der Liste der bereits abgeschlossenen Züge getestet werden ob der König und der Turm noch nie bewegt wurden und außerdem muß bei jedem zwischen diesen Figuren liegendem Feld und beim König sichergestellt werden das diese zur Zeit nicht "im Schach" stehen.

- Beim Springer entfällt als einzige Figur der Test ob alle Felder zwischen Start und Endposition frei sind, da dieser auch Figuren überspringen darf.

- Nach dem Zug darf der eigene König nicht im Schach stehen. Deswegen wird der Zug "zur Probe" ausgeführt und getestet ob dann eine "Schach"-Situation vorliegt.

- Liegt ein Remis vor, so mußdies speziell gekennzeichnet werden. Dies geschieht durch einen `[REMIS]`-Eintrag anstelle des Paßworts. Dies ist möglich, da ein '[' im Paßwort nicht vorkommen kann.[14]

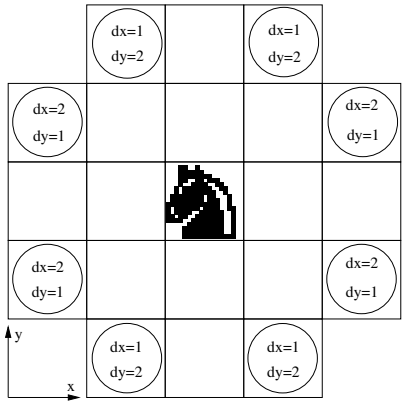
- Hat ein Spieler aufgegeben, so mußdies speziell gekennzeichnet werden. Dies geschieht durch einen `[GIVEUP]`-Eintrag anstelle des Paßworts.

Eine weitere Schwierigkeit liegt in der Implementation des Springers, da sich dieser nicht gerade, also weder senkrecht, noch wwgerecht oder diagonal bewegt.

[14] siehe S.20

Nach genauem Hinsehen erkannt man jedoch folgenden mathematischen Zusammenhang:
Sei dx der betragsmäßige Abstand des Ursprungsfeldes zum Zielfeld in waagerechter Richtung und dy der betragsmäßige Abstand des Ursprungsfeldes zum Zielfeld in senkrechter Richtung. Dann gilt für einen gültigen Zug des Springers:

$$dx + dy = 3 \text{ und } dx * dy = 2$$



3.4.1 Schach und Schachmatt

Ein Punkt an dem schon einige Schachprogramme gescheitert sind: Wie erkennt man das ein "Schach" oder ein "Schach-Matt" vorliegt ?
Der Test auf "Schach" ist recht aufwendig. Der König wird nacheinander als Springer, Läufer, Bauer u.s.w. behandelt. Kann er nun in einem Zug einen gegnerischen Stein dieser Art schlagen, so wird er in umgekehrter Richtung von dieser Figur attackiert und es liegt eine "Schach"-Situation vor.
Zu beachten ist dabei, daß die Bauern nur in eine Richtung gehen können.
Der Test auf "Schach-Matt" ist geschenkt.
Für den Aufbau des Spielfeldes wird die Funktion `canmove` benötigt, diese prüft zu einem gegebenen Feld ob die dort stehende Figur sich bewegen kann.
Dabei wird schon berücksichtig daß insbesondere der König nach dem durchgeführten Zug nicht im Schach stehen darf.
Ist kein solcher Zug möglich und der König steht bereits im Schach, so ist der Spieler Schachmatt.
Da es rein optisch wünschenswert ist das der Hinweis auf "Schach Matt" über dem Schachbrett steht, kann man jedoch leider diesen Test nicht direkt in die Funktion zum Feldaufbau integrieren. Aus diesem Grund wird bereits vor dem Aufbau des Feldes über die Funktion `isMate` die Anzahl aller beweglichen Figuren ausgezählt und bestimmt ob der König derzeit im Schach steht.

3.4.2 Speicherformat der Spiele

Pro Spiel wird eine Textdatei im Spieleverzeichnis angelegt. Der Name des Verzeichnisses kann in der `config.h`[15] durch die Variable `DATAPATH` festgelegt werden.
Zum schnellen Auffinden ist der Dateiname gleich der Game-ID und hat das Format `YYYYMMDDX...X`, mit:

YYYY	Jahr als 4-stellige Zahl
MM	Monat als 2-stellige Zahl
DD	Tag als 2-stellige Zahl
X...X	Die aktuelle Prozess-ID von W3Chess, variable Länge

, also z.B. 200304081108 für ein Spiel welches am 8. April 2003 vom W3Chess-Prozess mit der ProzessID 1108 initiiert wurde.
Durch die Maskierung von Datum und Prozess-ID sollte der Name eindeutig sein und außerdem ist zugleich bekannt wann das Spiel gestartet wurde.
Bei "offenen" Spielen wird ein 'X' vorangestellt.

Intern enthalten die Dateien zeilenweise alle wichtigen Informationen über das laufende Spiel, dabei ist die Reihenfolge der Zeilen relevant und Leer- oder Kommentarzeilen sind nicht gestattet.
Der Aufbau der Datei ist der folgende:

[*Nickname des sich am Zug befindlichen Spielers*]
[*EMail-Adresse des sich am Zug befindlichen Spielers*]
[*Nickname des Gegenspielers*]
[*EMail-Adresse des Gegenspielers*]
[*Passwort für den nächsten Zug*]
[*Anzahl der schon erfolgten Züge*]
[*Liste der schon erfolgten Züge in interner Notation*]
[*Aufbau des Boards in interner Notation*]
[*Besondere Flags*]@[*Kurznachricht an des Gegenspielers*]

Dabei werden derzeit die folgenden besonderen Flags erkannt:

`[REMIS?]`	Der Gegenspieler bietet ein "Remis" an
`[GIVEUP]`	Der Gegenspieler gibt auf

Man beachte das immer derjenige Spieler in der Datei an erster Stelle steht, der den nächsten Zug ausführen muß . Da "weiß " immer das Spiel beginnt kann anhand der Anzahl der bereits vorgenommenen Züge ermittelt werden welche Farbe der Spieler benutzt welcher sich am Zug befindet.

[15] siehe S.36

3.5 Die interne Notation der Züge

Die Züge werden nacheinander in einem langen String repräsentiert. Dabei besteht jeder Zug aus 7 Zeichen: ABCDEFG mit

A	Die Figur die bewegt wurde
BC	Das Startfeld, also z.B. A2
DE	Das Zielfeld, also z.B. A4
F	Die Figur die bewegt wurde oder bei der "Umwandlung" die Figur, gegen die der Bauer eingetauscht wurde
G	Die Figur die vorher auf dem Feld stand, 'e' falls es sich um ein leeres Feld gehandelt hat, 'R' bzw. 'r' für eine Rochade, 'O' für "En Passant"

Dabei wird die Figur durch eines der folgenden Zeichen repräsentiert:

R,r	Turm (Rook)
N,n	Springer (Knight)
B,b	Läufer (Bishop)
Q,q	Dame (Queen)
K,k	König (King)
P,p	Bauer (Pawn)

Die großen Buchstaben symbolisieren jeweils die weißen, die kleinen Buchstaben die schwarzen Figuren.
So bedeutet z.B. `PE2E4Pe`, daß der weiße Bauer von Feld `E2` auf das leere (`e`) Feld `E4` gezogen wurde und dort als Bauer stehen blieb. Dieses "ein Bauer bleiben" ist keinesfalls trivial, man bedenke die "Umwandlung"[16] !

3.6 Interne Darstellung des Boards und der Figuren

Das Board wird intern als 64 Zeichen langer String dargestellt. Jedes Zeichen symbolisiert ein Feld des Spielbretts, angefangen beim oberen linken und aufgehört beim unteren rechten Feld.
Jedes Zeichen steht dabei für eine Figur oder ein leeres Feld mit der gleichen Notation wie in der Darstellung der Züge.

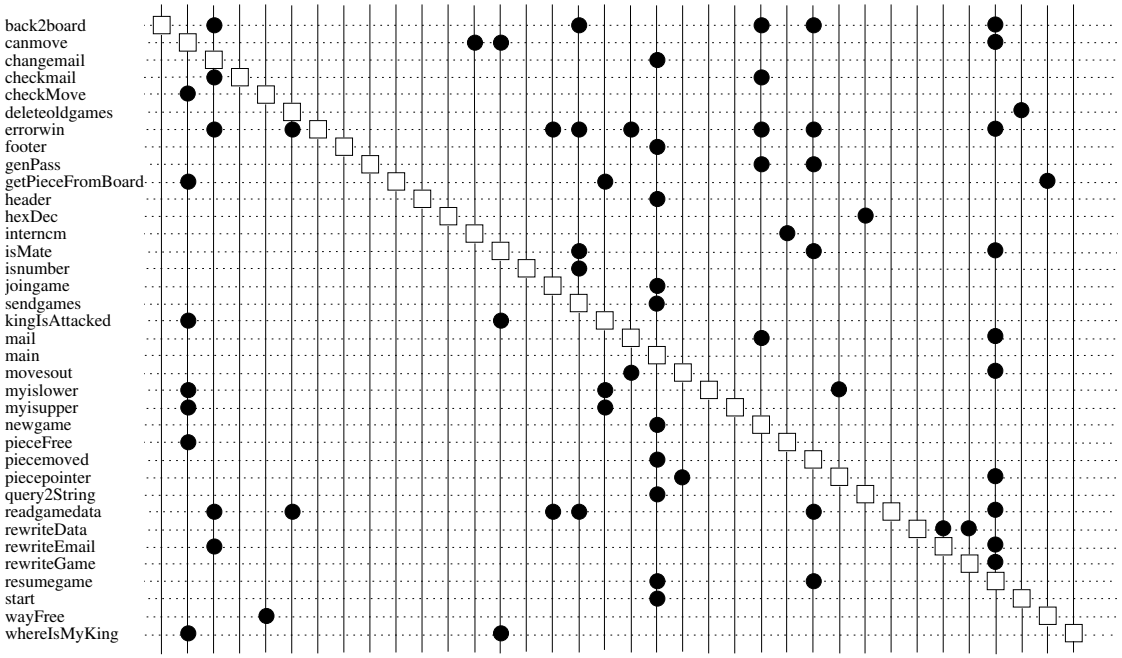
3.7 Darstellung der Kurznachrichten

Ein Spieler kann seinem Gegner bei jedem Zug eine Kurznachricht übermitteln, diese wird im Datenfile mit vorangestelltem '@'-Zeichen gespeichert. Vor dem '@' können noch spezielle Spielrelevante Flags[17] stehen, welche W3Chess z.B. veranlassen dem Benutzer die Anfrage für ein Remis zu übermitteln oder auch mitzuteilen wenn der Gegner aufgegeben hat.

[16] siehe S.7
[17] siehe "Speicherformat der Spiele", S.24

4 Die Funktionen von WebChess

4.1 Übersicht aller Funktionen



(Lesweise: Von oben nach unten, die Funtion mit dem □ ruft die mit dem • auf)

4.2 Beschreibung der Funktionen

Hier werden im Groben die einzelnen Funktionen in alphabetischer Reihenfolge des Programms beschrieben.
Zu beachten ist, das ein Großteil dieser Funktionen die folgenden globalen Variablen benützt:

Typ	Name	Beschreibung
char	GPLAYER1[ ]	Die Mailadresse des Spieler welcher am Zug ist
char	GPLAYER2[ ]	Die Mailadresse des Spielers welcher den letzten Zug gemacht hat
char	GNICK1[ ]	Der Spitzname des Spieler welcher am Zug ist
char	GNICK2[ ]	Der Spitzname des Spielers welcher den letzten Zug gemacht hat
char	GBOARD[ ]	Der momentane Stand des Spielfeldes
char	GPASS[ ]	Das aktuelle Paßwort für den nächsten Zug
char *	gmessage	Die Kurznachricht, die der andere Spieler hinterlassen hat und Informationen für Sonderfunktionen
char *	gmoves	Die bereits erfolgten Züge in interner Notation
int	gnummoves	Die Anzahl der bereits erfolgten Züge
char *	query	Die aktuelle CGI-Anfrage

Bei Prüf- und Testfunktionen ist der Rückgabewert, falls nicht anders angegeben “1” für “wahr” und “0” für “falsch”.

back2board

`void back2board(char *mesg, char *PASS, char *ID)`
zeigt eine allgemeine Meldung (`mesg`) mit ok-Button an. Wenn die `ID` nicht `NULL` ist, wird nach Bestätigung der Meldung das hierdurch angegebene Spielbrett angezeigt. Ein übergebenes Paßwort (`PASS`) wird durchgereicht.
Im Gegenteil zu `errorwin` wird hier nach einer angemessenen Wartezeit (diese kann durch `WAITTIME` in der `defaults.h` festgelegt werden) die Anzeige des Spielbretts automatisch ausgelöst.
Diese Funktion wird für kleine Infomeldungen an den Spieler benötigt, z.B. wenn er einen Zug abgeschlossen hat.

canmove

`int canmove(int fromx, int fromy, int x, int y)`
erwartet die Parameter (fromx,fromy) und (x,y) aus $\{'A', \ldots, 'H'\} \times \{1, ..., 8\}$ und $\{0, \ldots, 7\} \times \{0, ..., 7\}$. Sie testet ob ein Zug vom Feld (fromx, fromy) auf das Feld (x,y) erlaubt ist und gibt in diesem Fall eine 1 zurück, ansonsten eine 0.
Wenn (x,y) nicht auf (0,0) gesetzt ist, liegt der zweite Teil eines Zuges vor, also das Absetzen der Figur (`moveto=1`).
Nach einer Validierung der Parameter werden zunächst die allgemeinen Schachregeln überprüft, also z.B. ob man den eigenen Stein bewegen bzw. einen fremden Stein schlagen will und ob der Weg dorthin frei ist (`pieceFree`).
Befindet man sich bereits im Fall des Absetzens der Figur so wird nun der Zug testweise ausgeführt und getestet ob anschließend der eigene König verbotener Weise im Schach stehen würde (`kingIsAttacked`), danach werden die allgemeinen Zugregeln für den zu setzenden Stein überprüft.
Eingesetzt wird diese Funktionen zum einen von `isMate` um herauszufinden ob es noch “bewegliche” Steine gibt und zum anderen von `resumegame` beim Aufbau des Boards. Anhand von `canmove` wird dort entschieden ob eine Figur als `INPUT`-Tag oder als Bild implementiert wird[18], also ob man sie anklicken kann oder nicht.

[18] siehe S.11

changemail

`void changemail(void)`
speichert die vom Spieler geänderte EMailadresse im Spiel.
Die Spiele-ID und die neue Adresse werden direkt aus dem CGI-Query-String entnommen.
Diese Funktion wird aufgerufen wenn der Spieler seine EMailadresse geändert hat.

checkmail

`int checkmail(char *mail)`
prüft ob die durch `mail` übergebene EMail-Adresse gültig sein kann, also ob sie

- nicht leer ist
- ein @-Zeichen enthält
- einen . enthält
- der letzte . nicht vor dem @-Zeichen steht
- das @-Zeichen nicht direkt am Anfang auftaucht
- der Domainname mindestens 3 Zeichen lang ist
- der Toplevel-Domainname mindestens 2, jedoch maximal 3 Zeichen lang ist

Je nach Fehler wird einer der folgenden Werte zurückgegeben:

- 0, wenn die Adresse gültig ist
- -1, wenn `mail` ein `NULL`-Zeiger ist
- -2, wenn `mail` leer ist
- -3, wenn kein @-Zeichen vorkommt
- -4, wenn kein . vorkommt
- -5, wenn der letzte . vor dem @-Zeichen erscheint
- -6, wenn der Benutzername fehlt
- -7, wenn der Domainname zu kurz ist
- -8, wenn der Toplevel-Domainname ungültig ist

Man beachte: Bei einem Fehler ist der Rückgabewert immer <0, wenn also die Art des Fehlers irrelevant ist genügt eine Abfrage auf <0.
Wann immer der Spieler eine EMailadresse übermittelt, kommt diese Funktion zum Einsatz. Dabei geht es nicht darum "gefakte" EMailadressen zu verhindern, sondern vielmehr Schreibfehler bei der Eingabe des Paßwortes zu vermeiden.

checkMove

`int checkMove(int fromx, int fromy, int tox, int toy)`
prüft mit Hilfe der Funktion `wayFree` ob sich eine Figur von (`fromx`,`fromy`) nach (`tox`,`toy`) bewegen kann. Falls nicht wird eine '0' zurückgegeben, sonst die Richtung als numerischer Wert:

7	8	9
4		6
1	2	3

$\hat{=}$

↖	↑	↗
←		→
↙	↓	↘

Diese Funktion wird von `canmove` benutzt um zum Einen zu ermitteln ob z.B. ein Läufer wirklicher diagonal läuft und zum Anderen um herauszufinden ob eine Figur auf dem Weg eines Feldes zu einem anderen nicht durch eine Figur blockiert wird.

deleteoldgames

`void deleteoldgames(int html_out)`
löscht beim Aufruf der ersten Seite automatisch alle Spiele welche älter sind als 30 Tage. Dieser Wert kann über die Variable `DELETAFTER` in der `defaults.h`[19] verändert werden. Wird für `html_out` der Wert '1' übergeben, so wird automatisch eine Statistik über die Anzahl aller der momentan laufenden Spiele erstellt.
Durch dieses rigerose Löschen wird ein "Überlaufen" des Systems mit alten Spielen vermieden.
Ein Automatismus ist notwendig, da ein Spieler in der Regel ein Löschen des abgeschlossenen Spiels vergessen würde.
Hinweis: In der `config.h`[20] kann das Löschen der Spiele durch die Variable `ALLOWREMOVE` unter dem Gesichtspunkt der Sicherheit auch unterdrückt werden. In diesem Fall werden die Spiele aber dennoch gezählt.

errorwin

`void errorwin(char *mesg, char *pass, char *id)`
zeigt eine Fehlermeldung (`mesg`) mit ok-Button an. Wenn die `id` nicht `NULL` ist, wird nach Bestätigung der Meldung das hierdurch angegebene Spielbrett angezeigt. Ein übergebenes Paßwort (`pass`) wird durchgereicht.
Durch diese Funktion werden alle Fehlermeldungen dargestellt, also z.B. wenn das Paßwort falsch übergeben wurde.

footer

`void footer(void)`
fügt in jede Seite den HTML-Fuß ein. Dabei wird entweder ein Standardfuß ausgegeben, oder die in der `config.h`[21] durch `HTML_FOOTER` definierte Datei.

[19] siehe S.38
[20] siehe S.36
[21] siehe S.36

genPass

`void genPass(char *PASS)`
generiert ein 10 Zeichen + `NULL` langes Paßwort und schreibt es in `PASS`.[22]
Um SegFaults zu vermeiden ist dabei unbedingt darauf zu achten, das `PASS` mindestens 11 Zeichen lang ist !
Man beachte: `PASS` ist in diesem Fall ein beliebiger Zeiger, nicht zwingend die globale Variable `GPASS` !

getPieceFromBoard

`int getPieceFromBoard(int x, int y)`
liefert die Spielfigur an Position (x,y) in interner Darstellung zurück.[23]

header

`void header(void)`
fügt in jede Seite den HTML-Header ein. Dabei wird entweder ein Standardheader ausgegeben, oder die in der `config.h` durch `HTML_HEADER` definierte Datei.

hexDec

`int hexDec(char *hex)`
wandelt eine hexadezimale Zahl in eine Dezimalzahl um und gibt diese zurück.
Diese Funktion wird in `query2String` zur Umsetzung der Sonderzeichen im CGI-String benötigt.

interncm

`int interncm(int fromx, int fromy, int x, int y)`
ist ein Wrapper für `canmove` um bei internen Aufrufen die Feldbezeichner $\{0,...,7\}^2$ anstelle von $\{'A',...,'H'\} \times \{1,...,8\}$ verwenden zu können.
Außerdem wird durch sie bei internen Aufrufen das Brett gedreht, d.h. die Reihe mit der größten Nummer ist unten, nicht oben.

isMate

`int isMate(void)`
testet anhand der globalen Variablen `GBOARD` und `GPASS` mit Hilfe der Funktionen `whereIsMyKing` und `kingIsAttacked` den Status des aktuellen Spiels und gibt diesen zurück. Die Rückgabewerte sind dabei 0 (default), 1 (schachmatt), 2 (schach), 3 (remis), 4 (aufgegeben).
Die Funktionsweise wird im Abschnitt "Schach und Schachmatt" auf Seite 23 näher beschrieben.

[22] siehe S.20
[23] siehe S.25

isnumber

`int isnumber(char *string)`
prüft ob der übergebene `string` eine gütlige Spiel-ID ist und gibt in diesem Fall eine '1' zurück, sonst eine '0'.
In der Funktion `deleteoldgames` soll durch diese Erkennung das Problem des Löschens von Dateien eingedämmt werden welche keine Spiele sind. Insbesondere sollen keine Dateien akzeptiert werden welche ein '..' oder ein '/' enthalten, da diese auf einen typischen Hackerangriff oder eine Fehlkonfiguration hindeuten.

joingame

`void joingame(void)`
erstellt die Abfrage-Seite nach Benutzernamen und EMailadresse wenn ein Spieler an einem offenen Spiel teilnehmen möchte.

kingIsAttacked

`int kingIsAttacked(int x, int y, int king)`
prüft ob der durch `king` übergebene König (also 'k' oder 'K') auf dem Feld (x,y) bedroht ist (1) oder nicht (0).
Diese Funktion wird von `canmove` und `isMate` benötigt um zu testen ob der König im "Schach" steht.

mail

`void mail(char *id, char *an, char *from, char *annick, char *fromnick, char *pass, char *moves, char *board, char *message, int nummov, int mate)`
ist dafür zuständig, nach dem Abschluß eines Spielzuges (oder auch nach der Anmeldung) den aktuellen Spielstand an die Mitspieler zu mailen und sie zum nächsten Zug aufzufordern.[24]
Dabei werden die folgenden Parameter übergeben:

- `id`: Die ID des aktuellen Spiels
- `an`: Die Mailadresse des Empfängers
- `from`: Die Mailadresse des Absenders
- `annick`: Der Nickname des Empfängers
- `fromnick`: Der Nickname des Absenders
- `pass`: Das Paßwort, kann auch `NULL` sein
- `moves`: Die bereits erfolgten Züge in interner Notation
- `board`: Das Spielbrett in interner Notation
- `message`: Die Kurznachricht an den Mitspieler
- `nummov`: Die Anzahl der bereits erfolgten Züge
- `mate`: aktueller Spielstand: Matt, Remis, ... (siehe `isMate`)

[24] siehe S.21

Aus diesen Daten wird mit Hilfe von `movesout` die Mail an die Spieler generiert und an einen externen Mailer übergeben. Dieser kann während der Installation eingestellt werden.[25]

main

`int main(void)`
liest den CGI-Query-String ein, gibt den HTTP- und den HTML-Header aus (`header`), startet die im Query-String durch `ACTION` angeforderte Funktion und gibt anschließend den Fuß des HTML-Dokuments aus (`footer`).

movesout

`void movesout(FILE *wohin, char *moves, int nmoves)`
generiert die Liste der bereits abgeschlossenen Züge.
Dabei wird eine Liste der "`nmoves`" vielen bereits erfolgten Züge im internen Format durch `moves` übergeben.
Die Liste wird dann auf "`wohin`" ausgegeben, `wohin` ist dabei eine Pipe oder `stdout`.
Im Fall von `wohin=stdout` wird HTML-Code produziert, sonst Ascii-Code.
Benötigt wird diese Funktion zur Darstellung der vorangegangenen Züge unterhalb des Schachbretts und in der EMail an die Spieler.

myislower

`int myislower(int c)`
testet ob ein Buchstabe ein Kleinbuchstabe ist oder nicht. Im Gegensatz zu den Standard-C-Funktionen liefert sie definierte Werte zurück.

myisupper

`int myisupper(int c)`
testet ob ein Buchstabe ein Großbuchstabe ist oder nicht. Im Gegensatz zu den Standard-C-Funktionen liefert sie definierte Werte zurück.

newgame

`void newgame(void)`
wird direkt nach der Anmeldung eines neuen Spiels aufgerufen, oder wenn ein Benutzer sich einer Partie angeschlossen hat.
Sie speichert die Daten der Spieler und initiiert das Spielbrett.
Anschließend werden die Spieler per Mail benachrichtigt.

[25] siehe S.36

pieceFree

`int pieceFree(int piece,int x,int y)`
testet ob sich die Figur `piece` an der Position (x,y) bewegen kann oder nicht.
Die Figur wird dabei in interner Notation übergeben.[26] Diese Funktion wird von `canmove` benutzt um herauszufinden ob sich eine Figur bewegen kann oder blockiert ist.

piecemoved

`void piecemoved(void)`
schließt einen Zug ab. Speichert also das neue Spielbrett, generiert die Mails an die Mitspieler und initiiert gegebenenfalls den Bauerntausch. Außerdem findet in dieser Funktion ein letzter Test auf die Gültigkeit des Paßworts und des Zuges statt. Damit soll verhindert werden, daß durch geschickt gewählte CGI-Parameter ein ungültiger Zug vollzogen wird.

piecepointer

`int piecepointer(char piece,char **shortp, char **longp, char **image, char **small, char **stroken)`
setzt die übergebenen Zeiger auf die zu "`piece`" gehörenden Zeichenketten für den Kurznamen der Figur (`shortp`), den Langnamen der Figur (`longp`), das kleine (`small`), das durchgestrichene kleine (`stroken`) und das normale (`image`) Bild der Figur.
Beim Aufbau des Schachbretts, der Zugliste und der EMail an die Spieler wird diese Funktion wiederholt benötigt, sie ist vor allen Dingen nützlich um die verschiedenen Darstellungen der Figuren (kleine Symbole, große Symbole, durchgestrichene Symbole, Text, kurzer Text) an zentraler Stelle verwalten zu können.

query2String

`void query2String(char *string)`
wandelt in `string` jedes Vorkommen von '%AB' in das Zeichen mit dem Ascii-Code 'AB' um.
Mit dieser Funktion werden die durch den Browser codierten Sonderzeichen im übergebenen CGI-Query-String in die "normalen" Sonderzeichen zurückverwandelt.[27]
Da hierdurch `string` nur kürzer, aber nie länger wird, kann die Umwandlung direkt innerhalb der Zeichenkette erfolgen.

[26] siehe S.25
[27] siehe S.14

readgamedata

`int readgamedata(char *game)`
erwartet die ID einer Schachpartie, liest das zugehörige Datenfile ein und setzt mit diesen Werten die globalen Variablen.[28]
Diese Funktion wird von nahezu allen Routinen benötigt welche einen Zugriff auf die Spieldaten benötigen.

rewriteData

`void rewriteData(char *ID,char *PASS,char *MSG,char *MAIL)`
ändert im zu `ID` gehörenden Spiele das Paßwort (`PASS`), die Kurznachricht (`MSG`), oder die EMail-Adresse des sich am Zuge befindlichen Spielers (`MAIL`), sofern diese als nichtleere Zeichenketten übergeben werden.

rewriteEmail

`void rewriteEmail(char *ID, char *MAIL)`
ist ein Wrapper für `rewriteData`, mit dem Unterschied das immer nur die EMail-Adresse des sich am Zug befindlichen Spielers geändert werden kann.

rewriteGame

`void rewriteGame(char *ID,char *PASS,char *MSG)`
ist ein Wrapper für `rewriteData`, mit dem Unterschied das nur Paßwort und Kurznachricht geändert werden können.

resumegame

`void resumegame(int ischange)`
Dies ist die Hauptfunktion wenn es um den reinen Spielverlauf geht.
Sie ist zuständig für die Anzeige des Spielbretts und die Ausführung von Benutzerfunktionen wie die "Umwandlung" der Bauern am Ende des Spielbretts (`ischange=1`), das Ändern der Mailadresse, die Frage nach einem Remis, der Aufgabe und ähnlichem.
Die benötigten Informationen über den Aufbau des Spielbretts und für die Anzeige der bereits erfolgten Züge erhält die Funktion aus den globalen Variablen `GBOARD` und `gnummoves`, die Information über gewünschte Benutzerfunktionen direkt aus `query`.

[28] siehe S.26

sendgames

`void sendgames(void)`
schickt per Mail an den im CGI-Querystring durch `MAIL1` angegebenen Spieler eine Liste aller Spiele an denen er teilnimmt.
Diese Funktion wird ausschließlich benötigt wenn sich ein Spieler seine eigenen Spiele zusenden läßt.

start

`void start(void)`
erstellt die Seite welche beim Aufruf des Programms ohne Parameter erscheint. Im Normalfall kann man sich von hier aus in den Spielverlauf einloggen, sich seine aktuellen Spiele zusenden lassen, ein neues Spiel starten oder bei einem offenen Spiel mitmachen. Sollte in der `config.h`[29] die Variable `HTML_DEFPAGE` gesetzt sein, so wird diese Datei ausgegeben.

wayFree

`int wayFree(int fromx, int fromy, int tox, int toy,int xsgn, int ysgn)`
prüft ob der Weg von (`fromx`,`fromy`) nach (`tox`,`toy`) frei ist.
Dabei gibt (`xsgn`,`ysgn`) die beim Aufruf bereits bekannte Richtung durch $\texttt{xsgn}, \texttt{ysgn} \in \{-1, 0, 1\}$ an. Dies spart ein wenig Rechenzeit.
Diese Funktion wird ausschließlich von `checkMove` benötigt.

whereIsMyKing

`int whereIsMyKing(int color)`
gibt die Position des weißen (`color=0`) oder des schwarzen (`color=1`) Königs innerhalb des globalen Feldes `GBOARD` zurück, oder '-1', falls er nicht gefunden wurde.
Dieser Fall sollte allerdings nie eintreten, da dann der König geschlagen sein müßte.
Benötigt wird diese Funktion von `canmove` um die Postition des eigenen Königs für den Test auf "Schach" zu ermitteln.

[29] siehe S.36

5 Installation und Konfiguration

5.1 Erzeugung des Binaries

W3Chess verfügt über keine Möglichkeiten zur Konfiguration während der Laufzeit, alle Einstellungen müssen vor dem compilieren getroffen werden. Da diese Einstellungen sich während des Betriebs in der Regel nicht ändern, wird auf diese Weise der funktionelle Overhead vermieden den das Parsen eines Configfiles bei jedem Aufbau einer Seite mit sich bringen würde.

Als ersten Schritt sollte man sich für eine Sprache entscheiden, die vorhandenen Sprachen liegen in Dateien der Form `lang-xx.h` vor. Dabei steht `xx` für den zweistelligen Ländercode der entsprechenden Sprache, also z.B. `lang-en.h` für englisch.
Die zur Sprache passende Datei muß einfach nach `lang.h` kopiert werden.

Anschließend können in der Datei `config.h` die wichtigsten Parameter eingestellt werden:

`IMGURLPREFIX`	Die URL zum Verzeichnis in dem auf dem WebServer die für W3Chess benötigten Bilder später liegen werden. Können die Bilder mittels `http://www.abc.de/chessimg/yes.gif` erreicht werden, so wäre dies z.B. `/chessimg/`
`DATAPATH`	Das lokale Verzeichnis auf dem auf dem Webserver die Daten der Spiele gespeichert werden. Also z.B. `/var/lib/w3chess/`. Zu beachten ist das der Webserver in diesem Verzeichnis Schreibrechte besitzen muß !!! In diesem Verzeichnis sollten in keinem Fall andere Dateien als die der Spiele liegen !!!
`SENDMAIL`	Der vollständige Aufruf für ein Programm welches alle Daten die es auf dem `stdin` empfängt per Mail an den Empfänger sendet der als erster Parameter übergeben wird. Unter den meisten UniX-Systemen kann dies z.B. `sendmail` sein, auf anderen Systemen auch ein Perl-Skript oder ein beliebiges anderes Programm welches die geforderten Eigenschaften erfüllt.
`SUBJECT`	Der Betreff der Mails welche an den Benutzer geschickt werden, z.B. `[W3CHESS]`
`ALLOWREMOVE`	`1` wenn alte Spiele gelöscht werden sollen, `0` sonst.
`DELETEAFTER`	Die Anzahl der Tage nach denen ein unbenutztes Spiel gelöscht wird, z.B. `30`.

HTML_HEADER	Der Pfad zu einer lokalen Datei auf dem WebServer, z.B. `header`. Existiert diese, so wird sie anstelle des Default-Kopfes zu Beginn einer jeden generierten Seite ausgegeben.
HTML_FOOTER	Der Pfad zu einer lokalen Datei auf dem WebServer, z.B. `footer`. Existiert diese, so wird sie anstelle des Default-Fußes am Ende einer jeden generierten Seite ausgegeben.
HTML_DEFPAGE	Der Pfad zu einer lokalen Datei auf dem WebServer, z.B. `defaultpage`. Existiert diese, so wird sie anstelle der normalen Seite ausgegeben, wenn W3Chess ohne Parameter aufgerufen wird. Um die richtigen Formulardaten zu übertragen sollte man als Vorlage für eine solche Datei die Seite benutzen welche normalerweise ausgegeben wird.

Die letzten 3 Parameter ermöglichen es das Layout von W3Chess leicht an eigene WebDesigns anzupassen. Weitere Konfigurationsmöglichkeiten werden im Abschnitt 5.3 erläutert.

Anschließend kann das Programm `w3chess.c` mit einem C-Compiler übersetzt werden. Sollten auf dem System GNU-Make und GCC installiert sein so geschieht dies mit dem Befehl `make`, sollte nur der GCC vorhanden sein, so lautet der Aufruf
`gcc -o w3chess.cgi w3chess.c`.
Für andere Compilier muß dessen Beschreibung konsultiert werden.
Nach der Übersetzung sollte das erzeugte Binary `w3chess.cgi` heißen.

Anmerkung: Es ist dem Autor sehr wohl bewußt, daß diese Art der manuellen Konfiguration mittlerweile leider nicht mehr modern ist. Durch Verwendung von Hilfsmitteln wie `autoconf`, `automake` und `gettext` entstehen jedoch durch den geringen Konfigurationsaufwand nicht zu rechtfertigende Abhängigkeiten zu eben diesen Tools.

5.2 Installation des Binary

Vor der eigentlichen Installation muß verifiziert wo auf dem Web-Server CGI-Dateien ausgeführt werden dürfen, dort muß dann das Binary hinkopiert werden.
Unter vielen UniX-Systemen ist dies zum Beispiel das Verzeichnis `/var/www/cgi-bin/`.
Sollte es sich um den kostenlosen Apache-WebServer handeln[30], so kann auch mit der folgenden Zeile in der `httpd.conf` ein entsprechendes Verzeichnis definiert werden:
`ScriptAlias /w3chess/ ‘‘/var/www/w3chess/’’`
(In diesem Fall würde das lokale Verzeichnis `/var/www/w3chess/` von außerhalb als `http://rechnername/w3chess/` erreichbar sein.)
Ab jetzt wird die Installation einfach. Es müssen nur noch die Dateien aus dem Unterordner `figures/` in das Verzeichnis kopiert werden welches per Webbrowser wie in der `config.h`[31] durch die Variable `IMGURLPREFIX` definiert, erreicht werden kann.

[30] http://www.apache.org/
[31] siehe S.36

Bei der Installation ist auf jeden Fall auf richtig gesetzte Zugriffsrechte zu achten, besonders die Schreibrechte des Webservers auf das Datenverzeichnis werden erfahrungsgemäß oft vergessen.
Welche Rechte dies im Einzelnen sind hängt leider stark vom gewählten Betriebssystem ab, an dieser Stelle muß also einmal mehr auf die Dokumentationen des gewählten Systems und des WebServers verwiesen werden.

5.3 Anpassen des Designs

Im Normallfall sollte es reichen die Werte in der `config.h` anzupassen. Wem dies jedoch nicht genügt, der kann das Erscheinungsbild von W3Chess zusätzlich in der Datei `defaults.h` beeinflussen.
Die Datei ist nur für den erfahrenen Anwender bestimmt und sollte für einen solchen selbsterklärend sein, daher hier nur ein grober Überblick über die Möglichkeiten:

Debug-Funktionen	
`DEBUG`	Debug-Ausgaben erzeugen
`ALLCANMOVE`	Gültigkeitsprüfung für Züge unterdrücken
`NOMAIL`	Generierung von Mails unterdrücken
`NOPASS`	Jedes Paßwort akzeptieren
`REDFRAMES`	Mögliche Züge rot umranden

(Diese Funktionen werden durch ein `#define` anstelle des `#undef` aktiviert)

Array Größen	
`MAXSTRING`	Die Länge von allgemeinen Strings
`MAILLENGTH`	Die maximale Länge einer EMail-Adresse
`NICKLENGTH`	Die maximale Länge eines Nicknames

Farben	
`BOARDBLACK`	Farbe der “schwarzen” Felder
`BOARDWHITE`	Farbe der “weißen” Felder
`SELECTEDCOLOR`	Farbe des Rahmens um die ausgewählte Figur
`WARNCOLOR`	Farbe von Warnungen, z.B. die “Schach”-Ansage
`BOARDMARGBG`	Hintergrundfarbe der Randbeschriftung des Schachbretts und der Dialoge
`BOARDMARGFG`	Vordergrundfarbe der Randbeschriftung des Schachbretts und der Dialoge
`BOARDGRID`	Rahmenfarbe der Randbeschriftung des Schachbretts und der Dialoge
`MSGCOLOR`	Farbe von Kurzmitteilungen

Allgemeine Bilder	
BUTTONYES BUTTONNO	Die relativen Pfade zu den Bildern für die Dialogboxen.

Spielfiguren	
BPAWN, WPAWN BBISHOP, WBISHOP BKING, WKING BKNIGHT, WKNIGHT BQUEEN, WQUEEN BROOKE, WROOKE EMPTY	Die relativen Pfade zu den Bildern der Spielfiguren und des Leerfeldes. Das Leerfeld ist dabei leider unumgänglich da einige Webbrowser[32] mit Größenangaben in Tabellen nicht umgehen können.
SBPAWN, SWPAWN SBBISHOP, SWBISHOP SBKING, SWKING SBKNIGHT, SWKNIGHT SBQUEEN, SWQUEEN SBROOKE, SWROOKE	Die relativen Pfade zu den Bildern der kleinen Spielfiguren für die Anzeige der vorangegangenen Züge.
SSBPAWN, SSWPAWN SSBBISHOP, SSWBISHOP SSBKING, SSWKING SSBKNIGHT, SSWKNIGHT SSBQUEEN, SSWQUEEN SSBROOKE, SSWROOKE	Die relativen Pfade zu den Bildern der kleinen durchgestrichenen Spielfiguren für die Anzeige der geschlagenen Figuren in der Liste der vorangegangenen Züge.

Größen und Zeiten	
FIELDSIZE	Größe eines Schachfeldes
PIECESIZE	Größe einer Spielfigur
MESSAGEBOXLEN	Länge des Eingabefeldes für die Kurzmitteilungen
WAITTIME	Die Zeit nach welcher Warnungen automatisch bestätigt werden

6 Literaturliste und weiterführende Quellen und Links

Historisches

Zeitleiste des Schachspiels: *http://misc.traveller.com/chess/history/*

Mensch gegen Maschine

Näheres zum Match Kasparow gegen IBMs Großrechner:
http://www.heise.de/tp/deutsch/inhalt/lis/9305/1.html
http://www.research.ibm.com/deepblue/

Andere Web- und Zugbasierte Schachprogramme

WebChess und Jaquemate sind zwei zu W3Chess sehr ähnliche Programme, benötigen jedoch PHP und MySQL bzw. JavaScript:
http://webchess.sourceforge.net/
http://ambient.2y.net/wingo/projects/jaquemate/

ICS

Der einzige freie und leider nicht mehr weiterentwickelte freie Schachserver:
http://chessd.sourceforge.net/

Der freie Schachserver der Uni Kaiserslautern: *http://www.unix-ag.uni-kl.de/ chess/*
Ein freier Schachserver für weltweite Spiele: *http://www.freechess.org/*
Eines der bekanntesten ICS-Clienten und zeitgleich eines der besten freien Schachprogramme: *http://www.gnu.org/software/chess/chess.html*

HTML und CGI

Alles Informationen die für dieses Programm zum Thema HTML und CGI benötigt wurden stammen von SelfHTML: *http://selfhtml.teamone.de/*

Schachregeln

Die Schachregeln der FIDE: *http://www.schachbund.de/fideregeln/Fideregeln.htm*
Homepage des Weltschachverbandes: `http://www.fide.com/`

Andere Software

Der Webbrowser Mozilla: *http://www.mozilla.org/*
Der WebServer Apache: *http://www.apache.org/*
Der freie C-Compiler: *GCC: http://gcc.gnu.org/*
Das "schweizer Taschenmesser": `http://www.busybox.net/`

Unterhaltsames

Der Mystik des Schachspiels konnten sich auch Autoren nicht entziehen, daher gibt es viele Werke die Schach zum Hauptthema haben.
Genannt seien hier zur Abrundung der weiterführenden Quellen ein sehr guter Roman und ein guter Film:
Die Schachnovelle von Stefan Zweig, 2003, Fischer Taschenbuch, Frankfurt
Knight Moves, Thriller von Carl Schenkel, 1992

7 Danksagungen und Aussicht

Für zahlreiche Verbesserungsvorschläge, Anmerkungen, Bug Reports und stundenlange Tests möchte ich mich an dieser Stelle ganz herzlich bei Heike Opitz, Rainer Rose und Tabea Müller bedanken.
Ohne sie wären noch mehr Fehler unentdeckt geblieben und einige Funktionen würde es nicht geben.
Nicht zuletzt wurden auf diese Weise die Tests bisweilen sehr spannend....

W3Chess wird nach Ablauf der Studienarbeit als eigenständiges Projekt unter der GNU Public License weitergeführt, neuere Versionen werden unter
`http://www.freshmeat.net` erscheinen.
In den Status eines Produktionssystems wird das Spiel auf den Rechnern der UniX-AG der Uni Hannover verfügbar und vom Studierendenrechner aus erreichbar sein.