MNM
TEAM
Munich Network Management Team

Ludwig-Maximilians-Universität München

Prof. Dr. D. Kranzlmüller
Dr. N. gentschen Felde

Willkommen in der Gruppenphase des Systempraktikums. Ihre Aufgabe in der Projektphase ist es, einen Client für das Brettspiel *Mühle* in der Programmiersprache `C` zu entwickeln. Die Übungsblätter werden Sie schrittweise zu diesem Ziel führen.

Der Lehrstuhl stellt im Rahmen des Systempraktikums einen Gameserver und ein Webinterface bereit. Der Gameserver implementiert den Spielablauf, Ihr Client die Spielelogik. Der Gameerver ist gewissermaßen der Spielleiter und somit insbesondere für die Regeleinhaltung verantwortlich, wohingegen Ihr Client in die Rolle eines Spielservers schlüpft. Über das Webinterface unter `http://sysprak.priv.lab.nm.ifi.lmu.de`[1] können Spiele verwaltet werden.

Um das Testen Ihres Clients zu erleichtern, bietet das Webinterface zudem die Möglichkeit als menschlicher Spieler an einem Spiel teilzunehmen. Sie können jetzt testweise ein neues Spiel erstellen und zum Einstieg gegen ihre Kommilitonen oder sich selbst spielen. Sie werden einen Einblick gewinnen, wie das Spiel abläuft.

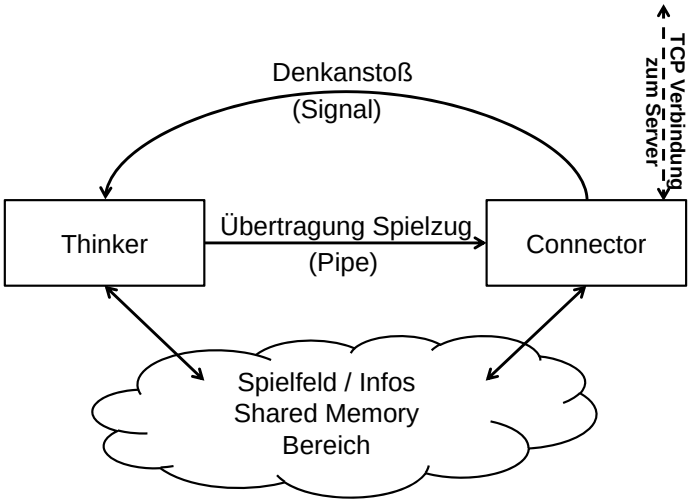

Abbildung 1: Übersicht über den Client

Der von Ihnen zu entwickelnde Client wird im Verlaufe des Systempraktikums schrittweise entwickelt. Maßgeblich besteht er aus zwei Prozessen, dem *Thinker* und dem *Connector*. Der *Connector* übernimmt die Kommunikation mit dem Gameserver und der *Thinker* berechnet den nächsten Spielzug.

Wird der Connector vom Gameserver zum nächsten Spielzug aufgefordert, so legt er alle Informationen, die er vom Gameserver bekommen hat um den nächsten Spielzug zu berechnen, in einen *geteilten Speicherbereich*. Danach sendet der Connector ein *Signal* an den Thinker. Der dadurch „aufgeweckte“ Thinker fängt nun an, mithilfe der aus dem *geteilten Speicherbereich* gelesenen Informationen, den nächsten Spielzug zu berechnen. Das Ergebnis der Berechnungen, den Spielzug, sendet der Thinker über eine *Pipe* an den Connector zurück, welcher den Spielzug an den Gameserver sendet. Abbildung 1 gibt Ihnen einen groben Überblick über den Aufbau des Clients.

[1] Der Gameserver und das Webinterface sind nur aus dem MWN erreichbar.
Informationen zum MWN: `http://www.lrz.de/services/netz/`

1 Mühle

Knapp 1000 Jahre nachdem das Brettspiel Mühle seine Hochzeit in Europa hatte, wird es nun Ihre Aufgabe es in moderner Form zu Implementieren. Hier müssen sie jedoch das Spiel nicht in Stein meißeln oder einen verzierten Holztisch schnitzen, sondern es genügt, wenn sie es in C implementieren.

Hinweis: Ein paar Jahre nach der Hochzeit in Europa – im Jahr 1993 – konnten Informatiker der ETH Zürich nachweisen, dass das Mühlespiel gelöst ist. Das bedeutet, es kommt immer zu einem unentschieden, wenn beide Spieler „perfekt“ spielen.

1.1 Spielregeln

Von dem Mühlespiel gibt es diverse Variationen. Wir werden hier die bekannte Version der *Neunermühle* (engl. *Nine Men's Morris*) verwenden. Die Spielregeln sind an die des *Welt Mühle Dachverbandes* angelehnt.

Das Spielbrett besteht aus 24 Knoten. Diese sind, wie in Abbildung 2 dargestellt, mit Kanten verbunden. Jeder der zwei Spieler bekommt zu Beginn neun Spielsteine. Die Spielsteine liegen zu Beginn des Spiels noch neben dem Spielfeld und werden in einer ersten Spielphase nach und nach auf dem Spielbrett platziert. Der Spieler mit den weißen Spielsteinen eröffnet das Spiel, im Anschluss wird abwechselnd gezogen. Welcher Zug als gültig erachtet wird, hängt davon ab in welcher der drei Spielphasen sich der Spieler befindet.

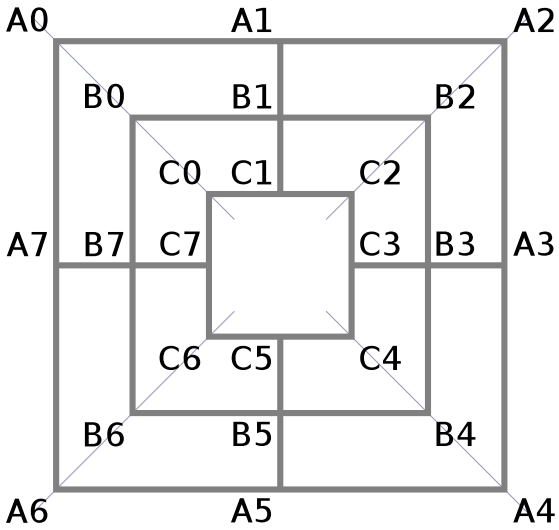

Abbildung 2: Neunermühle-Spielbrett mit Feldbezeichnungen

1. Solange ein Spieler noch nicht alle seine Steine auf dem Brett platziert hat, darf er einen dieser Steine auf ein beliebiges freies Feld legen.
2. Sind alle Steine platziert und es sind noch mehr als drei der eigenen Steine auf dem Spielfeld, so darf man mit einem beliebigen eigenen Stein auf einen seiner unmittelbaren freien Nachbarknoten ziehen.
3. Sind alle Steine platziert und es sind nur noch drei der eigenen Steine auf dem Spielfeld, so darf man mit einem beliebigen eigenen Stein auf ein beliebiges anderes freies Feld springen.

Um zu gewinnen, müssen Mühlen gelegt werden. Eine Mühle sind drei unmittelbar benachbarte Steine der gleichen Farbe auf einer geraden Linie. Z. B. ist A0-A1-A2 oder A1-B1-C1 eine Mühle, wohingegen A1-A2-A3 oder B1-C1-C5 keine ist. Wer mit einem Zug eine Mühle schließt darf einen beliebigen gesetzten Stein des Gegners entfernen (schlagen). Wer mit einem Zug zwei Mühlen schließt darf zwei beliebige gesetzte Steine des Gegners schlagen. Es ist dabei unerheblich, ob die Steine des Gegners teil einer Mühle sind oder nicht.

Hinweis: Gewiefte Spieler versuchen eine Zwickmühle zu bauen. Dabei schließt man mit jedem seiner Züge eine Mühle und öffnet dafür eine andere.

Verloren hat, wem der siebte Stein vom Spielfeld genommen wird oder wer am Zug ist, aber keinen gültigen Zug machen kann. Das Spiel endet unentschieden nach dem 50ten Zug ohne schlag.

2 Protokolldefinition des Gameservers

Ihr zu implementierender Client kann mit dem Gameserver über das hier beschriebenen zeilenorientierte Protokoll kommunizieren. Zeilen werden, wie unter Unix üblich, mit einem „newline character“ (`'\n'`) abgeschlossen. Der MNM-Gameserver ist wie folgt zu erreichen:

- *Hostname:* `sysprak.priv.lab.nm.ifi.lmu.de`
- *Port:* `1357` (TCP)

Wenn der Gameserver eine Zeile mit einem `+` als erstes Zeichen schickt, ist dies eine positive Antwort. Im Folgenden ist nur der positive Verlauf einer Kommunikation angegeben. An jedem Schritt kann eine Negativantwort auftreten, diese ist erkennbar an dem `-` als erstes Zeichen der Zeile. Ein `-` ist stets gefolgt von einer aussagekräftigen Fehlermeldung. Im Anschluss an die Fehlermeldung wird die Verbindung getrennt.

Wenn nicht innerhalb von im Gameserver festgelegten Zeitgrenzen auf Befehle geantwortet wird, oder eine zu lange „Denkzeit“ benötigt wird (s. u.), schickt der Gameserver:
S: - TIMEOUT ⟨⟨ *Begründung* ⟩⟩

Die folgende Protokolldefinition kürzt eine Zeile, welche vom Client an den Gameserver geschickt wird, mit **C:** für *Client* ab. Eine Zeile, welche vom Gameserver an den Client übermittelt wird, wird mit **S:** für *Server* abgekürzt.

In ⟨⟨ ⟩⟩ eingeschlossene Werte werden obligatorisch durch die ihnen entsprechenden Werte ersetzt. Werte, die in *[[ ]]* eingeschlossen sind, geben optionale Werte an, d. h. sie können auch weggelassen werden.

Das Protokoll gliedert sich in folgenden drei Phasen:

1. *Prolog* – hier wird dem Spiel beigetreten und Informationen über das Spiel ausgetauscht
2. *Spielverlauf* – hier wird gewartet bis man an der Reihe ist, bzw. das Spiel beendet wird
3. *Spielzug* – hier übermittelt der Gameserver ein Spielfeld und erwartet einen Spielzug

Diese Phasen werden in den folgenden Unterkapiteln ausführlich beschrieben. Das Zustandsdiagramm in Abbildung 3 gibt eine grobe Übersicht über den Ablauf der drei Protokollphasen.

2.1 Protokollphase Prolog

⟨⟨ *Aufbau der TCP-Verbindung durch Client* ⟩⟩
S: `+ MNM Gameserver` ⟨⟨ *Gameserver Version* ⟩⟩ `accepting connections`
C: `VERSION` ⟨⟨ *Client Version* ⟩⟩
S: `+ Client version accepted - please send Game-ID to join`
C: `ID` ⟨⟨ *Game-ID* ⟩⟩
S: `+ PLAYING` ⟨⟨ *Gamekind-Name* ⟩⟩
S: `+` ⟨⟨ *Game-Name* ⟩⟩
C: `PLAYER` *[[Gewünschte Spielernummer]]*
S: `+ YOU` ⟨⟨ *Spielernummer* ⟩⟩ ⟨⟨ *Spielername* ⟩⟩
S: `+ TOTAL` ⟨⟨ *Spieleranzahl* ⟩⟩

Nun kommt für jeden der anderen Spieler die Zeile:
S: `+` ⟨⟨ *Spielernummer* ⟩⟩ ⟨⟨ *Spielername* ⟩⟩ ⟨⟨ *Bereit* ⟩⟩

S: `+ ENDPLAYERS`

Der Gameserver akzeptiert den Client, wenn die Major-Versionsnummern (links vom Punkt) von ⟨⟨ *Gameserver Version* ⟩⟩ und ⟨⟨ *Client Version* ⟩⟩ gleich sind. Der Gameserver setzt jeweils ein `v` voran. Die Major-Versionsnummer des Gameserver ist dieses Semester immer 1, die Minor-Versionsnummer kann sich jedoch ändern. Kompatibel sind z. B. ein Gameserver mit der Version `v1.1` und ein Client mit der Version `1.42`. Nicht kompatibel wären hingegen `v1.1` und `2.1` oder `v1.1` und `v1.1` (Client hat ein `v` Präfix).

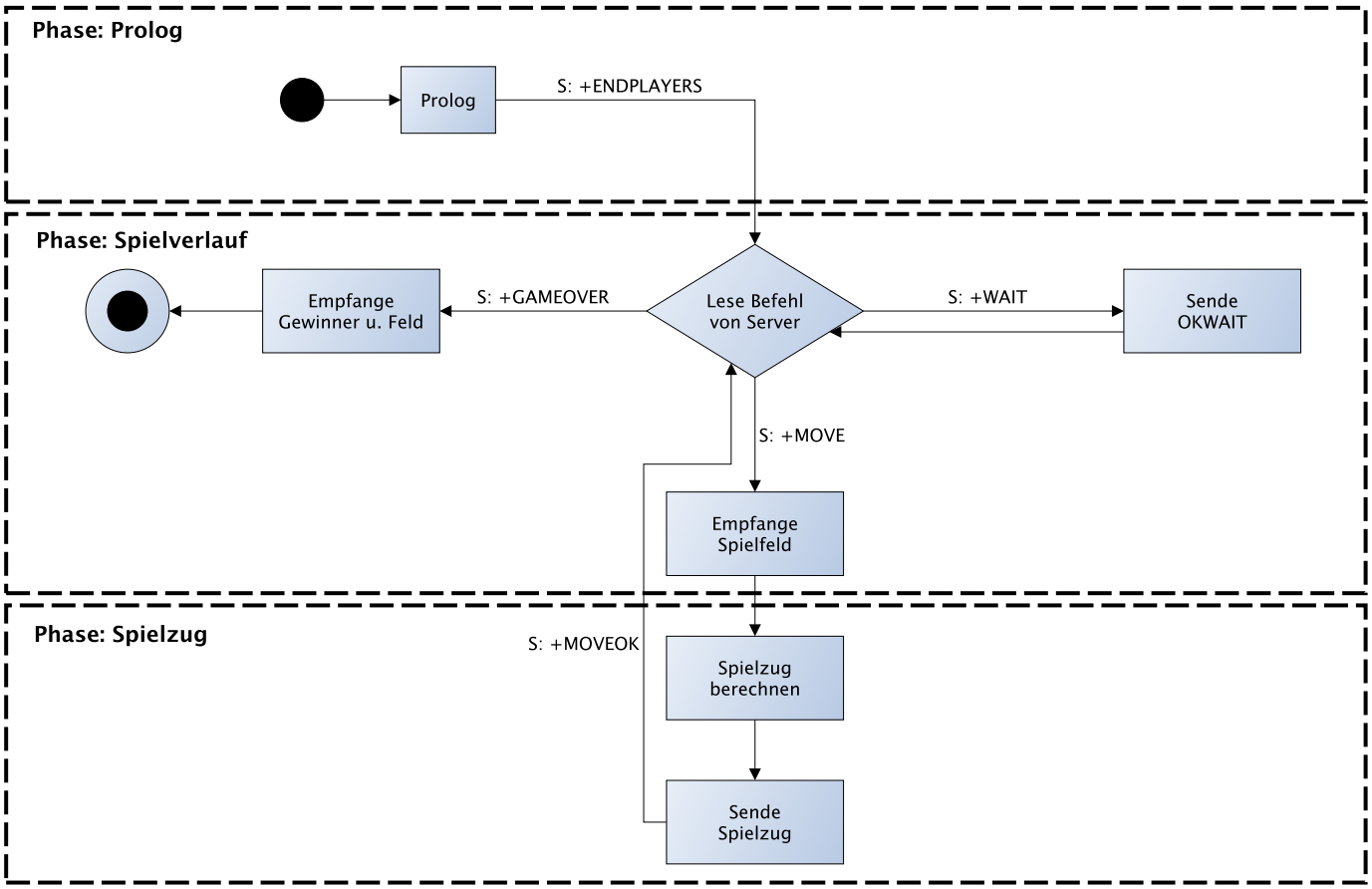

Abbildung 3: Protokollübersicht

Da der Gameserver nicht nur Mühle spielen kann, teilt ⟨⟨ *Gamekind-Name* ⟩⟩ die Spielart mit. Ist diese anders als `NMMorris` (verkürzte vorm von *Nine Men's Morris*), muss der Client sich mit einer Meldung, die auf dieses Problem hinweist, beenden.

⟨⟨ *Game-Name* ⟩⟩ und ⟨⟨ *Spielername* ⟩⟩ können im Webinterface beim Erstellen eines Spiel angegeben werden, bzw. enthalten Standartwerte, wenn die Angabe ausbleibt.

Wenn man einen Spieler mit einer bestimmten Nummer übernehmen möchte, kann dieser Wunsch mit *[[Gewünschte Spielernummer]]* angegeben werden. Lässt man die Spielernummer und das Leerzeichen nach `PLAYER` weg, so bekommt man einen freien Computerspieler vom Gameserver zugeteilt. Es ist zu beachten, dass der Spieler nicht sofort nach dem Abbruch der Verbindung (z. B. nach Absturz der Client-Software) wieder zur Verfügung steht. Es muss ein paar Sekunden gewartet werden, bevor der Gameserver den Computerspieler frei gibt.

Die ⟨⟨ *Spieleranzahl* ⟩⟩ wird bei einem Mühle-Spiel immer `2` sein.

Ist ein Spieler bereits verbunden, so ist ⟨⟨ *Bereit* ⟩⟩ `1`, ansonsten `0`. Auch wenn nicht alle Spieler bereit sein sollten, so wird mit der nächsten Protokollphase fortgefahren.

2.2 Protokollphase *Spielverlauf*

In dieser Phase können eine *Idle*, *Move* sowie eine *Game over* Befehlssequenz vorkommen. Wurde das erste Mal in die Spielverlaufsphase gewechselt oder eine der Befehlssequenzen beendet, so entscheidet die nächste Zeile (die im Diagramm an den Kanten annotiert ist) über die nächste abzuarbeitende Befehlssequenz.

2.2.1 *Idle* Befehlssequenz

S: `+ WAIT`
C: `OKWAIT`

WAIT-Befehle kommen in regelmäßigen Abständen und müssen rechtzeitig mit einem OKWAIT quittiert werden. Ansonsten beendet der Gameserver die Verbindung und das Spiel gilt im Turnier als verloren. Wie dem Diagramm in Abbildung 3 zu entnehmen ist, bleiben die WAIT-Befehle z. B. in der Protokollphase *Spielzug* oder während der *Move* Befehlssequenzen aus.

2.2.2 *Move* Befehlssequenz

S: + MOVE ⟨⟨ *Maximale Zugzeit* ⟩⟩
S: + CAPTURE ⟨⟨ *Anzahl zu schlagender Steine* ⟩⟩
S: + PIECELIST ⟨⟨ *Anzahl Spieler* ⟩⟩ , ⟨⟨ *Anzahl Steine* ⟩⟩

Die folgende Zeile wird nun für jeden Spielstein eines jeden Spielers geschickt:
S: + PIECE ⟨⟨ *Spielernummer* ⟩⟩ . ⟨⟨ *Steinnummer* ⟩⟩ ⟨⟨ *Postition* ⟩⟩

S: + ENDPIECELIST
C: THINKING
S: + OKTHINK

Der MOVE-Befehl fordert zum Zug auf. Nach der in Millisekunden angegebenen ⟨⟨ *Maximale Zugzeit* ⟩⟩ muss die anschließende Protokollphase „Spielzug“ abgeschlossen sein. Bitte berücksichtigen Sie bei Ihrer Implementierung die Latenzzeiten der Verbindung. Beachten Sie, dass zwischen ⟨⟨ *Anzahl Spieler* ⟩⟩ und ⟨⟨ *Anzahl Steine* ⟩⟩ kein Leerzeichen ist. Die ⟨⟨ *Spielernummer* ⟩⟩ und ⟨⟨ *Steinnummer* ⟩⟩ beginnen bei 0. Beachten Sie, dass zwischen PIECE, ⟨⟨ *Spielernummer* ⟩⟩ und ⟨⟨ *Steinnummer* ⟩⟩ keine Leerzeichen sind. Die ⟨⟨ *Postition* ⟩⟩ der Spielsteine kann folgende Werte annehmen:

- A – der Stein ist noch nicht gesetzt worden (available)
- A0 ... A7, B0 ... B7, C0 ... C7 – der Stein befindet sich auf dem Feld auf der Position, die Abbildung 2 zu entnehmen ist.
- C – der Stein wurde geschlagen (captured)

Der Client muss direkt nach der Übermittlung des Spielfeldes THINKING schicken und hat hierfür wenig Zeit. Nach der Gameserver-Antwort OKTHINK befinden wir uns in der Protokollphase „Spielzug“.

2.2.3 *Game over* Befehlssequenz

S: + GAMEOVER *[[* ⟨⟨ Spielernummer des Gewinners ⟩⟩ ⟨⟨ Spielername des Gewinners ⟩⟩ *]]*
S: + CAPTURE ⟨⟨ *Anzahl zu schlagender Steine* ⟩⟩
S: + PIECELIST ⟨⟨ *Anzahl Spieler* ⟩⟩ , ⟨⟨ *Anzahl Steine* ⟩⟩

Die folgende Zeile wird nun für jeden Spielstein eines jeden Spielers geschickt:
S: + PIECE ⟨⟨ *Spielernummer* ⟩⟩ . ⟨⟨ *Steinnummer* ⟩⟩ ⟨⟨ *Postition* ⟩⟩

S: + ENDPIECELIST
S: + QUIT
⟨⟨ *Abbau der TCP-Verbindung durch Gameserver* ⟩⟩

Bei einem Unentschieden wird kein Gewinner angegeben. Die weiteren Zeilen geben den Spielzustand wieder, mit dem das Spielende erreicht wurde. Dieser wird an alle Spieler geschickt. Nach QUIT beendet der Server die Verbindung.

2.3 Protokollphase *Spielzug*

C: PLAY ⟨⟨ *Spielzug* ⟩⟩
S: + MOVEOK

Befindet sich der Client in der ersten Phase des Spiels, so besteht ein ⟨⟨ *Spielzug* ⟩⟩ aus dem Namen des Feldes, an dem der Stein gesetzt werden soll. Wurde der Spieler mittels `CAPTURE` aufgefordert einen Stein des Mitspielers zu schlagen, so gibt er den Namen des Feldes an, auf dem der zu schlagende Stein liegt. In den verbleibenden Fällen gibt der Spieler den Namen des Feldes an, von dem mit seinem Stein auf ein anderes gezogen oder gesprungen werden soll. Der Name des Zielfeldes wird mit `:` getrennt angehängt. Um z. B. von A1 auf B4 zu springen lautet der ⟨⟨ *Spielzug* ⟩⟩ `A1:B4`. Sollte der übermittelte Zug nicht gültig sein, wird eine entsprechende Fehlermeldung übermittelt und die Verbindung getrennt. Ist absehbar, dass der Spieler erneut zum Zug aufgefordert wird (z. B. nach dem schließen einer Mühle) so können mehrere Spielzüge mit `;` getrennt aneinandergehängt werden.