Brändi Dog 2.0 Protokoll

Generelles Konzept des Protokolls

In Brändi Dog wird zur Netzwerkkommunikation ein String von Client zu Server (oder von Server zu Client) übermittelt. Aufgrund der Server/Client Architektur des Spiels werden alle Kommunikationen von Clients als Anfragen behandelt und alle Kommunikationen vom Server als Befehle.

Beispiel

Ein Spieler will eine Figur bewegen. Der Client fragt:

- „Kann ich Figur X auf Feld Y bewegen?“

Der Server überprüft die Legalität des Spielzuges:

- Ist der Spielzug illegal, antwortet er dem Client: „Nein.“
- Ist der Spielzug legal, antwortet er allen Clients im Spiel: „Bewege Figur X auf Feld Y.“

Ist der Spielzug legal, führen die Clients die Anweisung des Servers aus.

Format des zu übertragenden Strings

Die Strings, die zu Kommunikation verwendet werden, sind natürlich speziell formatiert. Es handelt sich um Character Separated Values (CSV). Als Separator dient das gute alte Semikolon (;). Der erste Wert im String ist jeweils der Befehl, und sämtlichen weiteren Werte sind die Argumente, die für den Befehl benötigt werden. Der übermittelte String hat also die folgende Form:

Befehl;Argument 1;Argument 2; (...) ;Argument n

Beispiel (Fortsetzung)

Der im obigen Beispiel genannte Befehl „Bewege Figur X auf Feld Y.“ (sowie die gleichgerichtete Anfrage) sehen, nach obiger Regel formatiert, so aus: „Bewege;Figur X;Feld Y“. Die Negativantwort wäre entsprechend einfach „Nein“, ganz ohne Semikolon und somit ohne Argumente.

Das Protokoll konkret

Implementation als Enumerator

Das Protokoll wird im Spiel mit Enumeratoren (enum) implementiert. Alle legalen Befehle werden im Enumerator aufgelistet, so kann das Programm einfach parsen, was genau übermittelt wird.

Trennung von Chat und Spiel

In der Implementation des Protokolls in unserem Programm haben wir uns dazu entschieden, den Chat vom Spiel zu trennen. Konkret bedeutet dies, das zwei Enumeratoren existieren: einer für den Chat, einer fürs Spiel. Da der Chat entkoppelt vom Spiel funktionieren soll, schien es eine logische Entscheidung zu sein, dies auch fürs Protokoll zu tun.

Die Befehle im Detail:

Das Chat-Protolkoll

Das Chat-Protokoll ist als ChatProtocol Enumerator in der Klasse Protocol implementiert. Es verfügt über die folgenden Befehle:

- **EXIT**
  EXIT wird genutzt, um den Chat zu verlassen. Der Befehl hat keine Argumente.
- **PING**
  PING sendet ein einfaches Ping-Signal zur überpüfung der Client/Server-Verbindung.
- **PONG**
  PONG ist die antwort auf ein PING-Request.
- **SHOW;[information to be shown]**
  SHOW zeigt informationen im Chat-Fenster an. Zum Beispiel eine Liste aller User. Als Argument wird gewählt, welche Information gezeigt werden soll. Sind mehrere Informationen gewünscht, müssen mehrere Show-Anfragen gesendet werden.
- **NICK;[new nickname]**
  NICK ändert den Namen im Chat. Als Argument wird der neue Name mitgeliefert.
- **CHAT;[recipient];[chat message]**
  Mit chat wird eine Chat-Nachricht gesendet. Das erste Argument ist der gewünschte Empfänger (Alle (> all)oder, für Flüstern, nur eine Person) und das zweite Argument ist die Nachricht.
- **JOIN;[nickname]**
  Mit JOIN wird einem spezifischen Chatroom beigetreten. Argument ist der Name im Chat (welcher später auch geändert werden kann).
- **ENTER;[Lobby Name]**
  Mit Enter wird einer Lobby beigetreten
- **READY**
  Mit ready wird dem Server gesagt, dass der Client zum Spielstart bereit ist.

Das Spiel-Protokoll

Das Spiel-Protokoll ist als GameProtocol Enumerator in der Klasse Protocol implementiert. Es verfügt über die folgenden Befehle:

- **START;[color]**
  Mit START schickt der Server das Startsignal und die eigene Spielerfarbe
- **EXIT**
  Mit EXIT wird das laufende Spiel verlassen. Es gibt keine Argumente.
- **SET;**
  SET ist eine Setter Methode. Als solche gibt es mehrere Varianten davon:
  SET;CARD;[Card]
  Die zuletzt gespielte Karte wird aktualisiert.
  SET;CARD;REMOVE;[Card]
  Die Karte [Card] wird aus der Hand entfernt.
  SET;HAND;EMPTY
  Die eigene Hand wird geleert.
  SET;TURN;[Farbe];[Name]
  Als nächstes zieht der Spieler mit Name [Name] und Farbe [Farbe].

- **GET**
  Mit GET wird eine Karte vom Server angefordert
- **FINISH**
  Mit FINISH werden die Spieler darüber informiert, dass das Spiel zuende ist.
- **PLAY;[Color];[Card];[Token];[Suffix]**
  Mit PLAY wird ein zug gespielt. Argumente sind: Die eigene Farbe, die zu spielenden Karte, die zu spielende Murmel, und, im Falle von besonderen Karten, ein Suffix, der die gewählte Variante der Karte übergibt.
- **TURN;[Player]**
  TURN sagt den Clients, welcher Spieler dran ist.
- **WIN;[winning player]**
  Mit WIN wird der Spieler ausgewählt, welcher das Spiel gewonnen hat. Als Argument wird der Spieler mitgegeben, der gewonnen hat.