Entwicklerhandbuch
Praktikum FP - Projekt: Quoridor

X-Games (Christian Willner, Fynn Ittner, Alexander Rösler)

19. September 2021

1 Struktur des Clienten

Der Startpunkt für die Ausführung des Clienten ist die Datei "quoridor-client.rkt" . Diese benötigt im gleichen Verzeichnis die Textdateien "host-address.txt" und "client-settings.txt" . In der ersten Datei sind die IP-Adresse und der Port des Servers einzutragen und die zweite Datei kann genutzt werden um die Größe des Spielfelds zu verändern. Der gegebene Wert entspricht der Pixelgröße einer Kachel. HD-Displays können gut mit einem Wert von 45 genutzt werden, auf 2K-Monitoren sind auch 100 möglich.

Im "secondary" -Verzeichnis finden sich die Hilfsfunktionen für die eigentliche Logik, die Darstellung, die Kommunikation und die Interaktion. Die Client-Datei nutzt die htdp2/world Bibliothek um die Benutzeroberfläche zu genieren. Die wichtigsten Punkte der genutzten Funktionen sind unten beschrieben. Details zur genauen Funktionsweise sind im Quellcode ausführlich erläutert.

1.1 Clientseitige Logik

Der Client beinhaltet die von 2htdp/universe zur Verfügung gestellten big-bang welche den initialen Zustand erstellt und abhängig von den eintreffenden Informationen die Folgezustände verwaltet. Dabei reagiert big-bang auf Maus- und Tastatureingaben, die gerenderte Ausgabe, die Tickrate und die Kommunikation mit dem Server. Der Client prüft selber welche Aktionen in jedem Zustand valide sind, neben Rendering und Interaktion sind also auch Spiellogik und Einstellungen auf Seiten der Clienten vorhanden.

Das Programm verwendet 5 allem Zugrunde liegende Strukturen:

CELL: Eine Zelle besteht aus einer X- und einer Y-Koordinate und dient als Positionsangabe auf dem Spielfeld.

WALL: Eine Mauer besteht aus einer Zelle und einer Orientierung. Mit diesen beiden Angaben können die Mauern beliebig auf dem Spielfeld platziert werden. Graphisch gesehen dient die obere linke Ecke Die Zelle als Startpunkt und die Orientierung die Richtung in welche die Mauer in der Länge von Zwei Zellen gezeichnet wird.

PLAYER: Ein Spieler besteht aus einer Id, einer Zelle und einer Anzahl an Mauern die dieser Spieler in seinem Reservoire hat. Die Id ist eine Zahl welche den Spieler eindeutig Identifizieren kann.

SPECIAL: Ein special besteht aus X- und Y-Koordinate, einem Image und zwei Frame-Angaben. Ein Special wird für auf den Bildschirm projezierte Elemente und Nachrichten verwendet.

WS (Worldstate): Ein Worldstate besteht aus einer Liste von Players, einer Liste von Walls, einer Spieler-ID welche den aktuellen Spieler beinhaltet, einem Gamestate und einem Special. Der Gamestate wird für die Navigation zwischen verschiedenen Menüs außerhalb des Spiels benötigt, der Special für besondere Nachrichten. Jeder Mögliche Zustand den der Client erreichen kann ist durch einen Worldstate beschreibbar.

Alle Structures sind in der structures.rkt zu finden. In der helpers.rkt befinden sich implementation der Spielregeln und alle dafür benötigten Hilfsmethoden. In der interactions.rkt ist die Bedienung des Spiels definiert. Jegliche Maus und Tastatureingaben werden hier ausgelesen, die Spiellogik abgefragt und der neue Zustand in Form des nächsten Worldstates generiert. Die genauen Aufgaben jeder Funktion sind im Quellcode in den Kommentaren noch deutlich weiter Ausgeführt.

1.2 Rendering

Das Rendering wird durch eine zweistufige Steuerung realisiert. In der ersten Stufe wird die Nachricht des Servers verarbeitet und dadurch der Startzustand für die Hauptrenderfunktion (render-state) gesetzt. Nachgeschaltet wird durch die Tick-Updates der Big-Bang Funktion möglicherweise eine Reihe von weiteren Zwischen-Statusen eingeführt. Dazu sollte ein Bestandteil des Worldstate Struts "ws" beschrieben werden. Dieser speichert nicht nur die Position von Mauern, Spielern und dem aktuellen Status der Anwendung, sondern auch ein eigenes "special"-Struct. Hier können Grafiken, Positionen und Frames übergeben werden. Diese können genutzt werden um etwa Animationen an bestimmten Positionen des Bildschirms zu überlagern. Anhand dieser Frame-Information kann durch die Tick-Update-Funktion die vergangene Zeit geprüft werden und so ein zeitlich getakteter Übergang zwischen den Animationen (siehe unten) gewährleistet werden.

Das statische Spielfeld wird in verschiedenen Layern übereinander aufgebaut. Eine Übersicht ist in Abb. 1 zu finden.
Die unterste Ebene besteht aus dem Spielbrett, im statischen Fall werden darauf die möglichen Spiel-

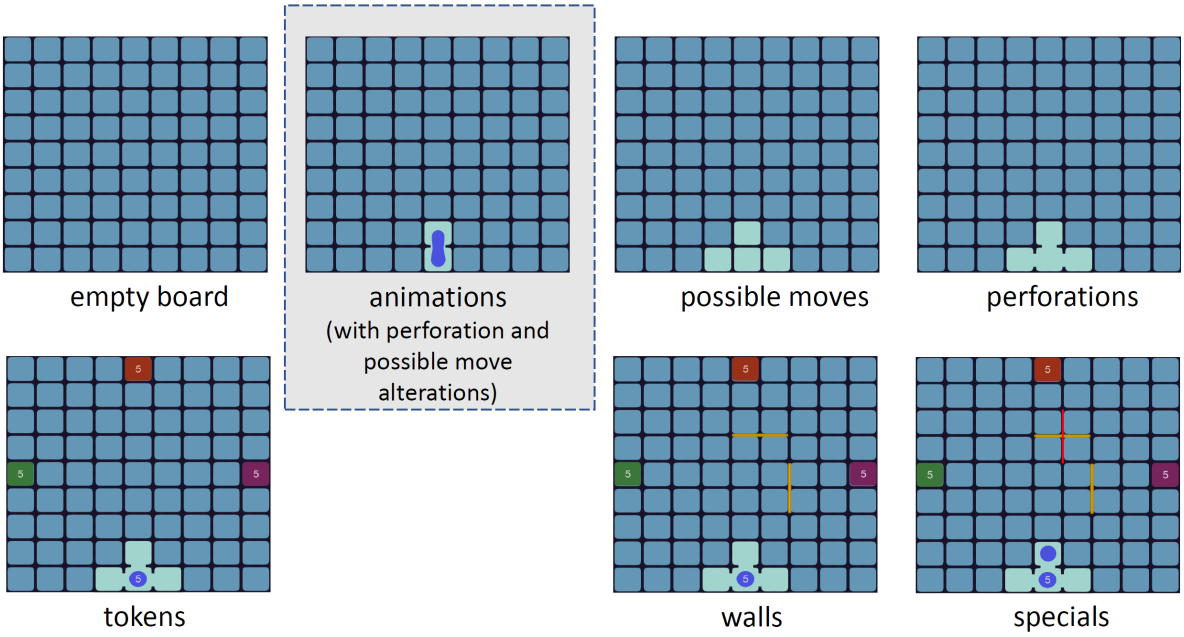

Abbildung 1: Schema der Ebenen für das Spielfeld, Zughinweise und die Figuren.

züge (bestehend aus possible moves und perforations), Spielsteine, Wände und Spezialbilder gerendert. Die Spezialbilder sind Hinweise für die gültigge Platzierung von Wänden und Spielsteinen. Die letzte Funktion wird durch die "on-mouse" Funktion gewährleistet. Abhängig von der Position der Maus, wird ein Bild und eine Position im Special Strukt des Worldstates hinterlegt, welche dann in der Renderfuntktion als obere Ebene gezeichnet wird.

Davon abweichend ist der Animationsfall für Bewegungen des Spielsteins, das Zeichnen der neuen Mauern und das Aktivieren bzw. Deaktiveren von Spielern zu Beginn jeder Runde. Da hier nur die betroffenen Zellen gezeichnet werden, übernimmt die Funktion zur Bewegung der Tokens auch das Rendern der erlaubten Zugkacheln und der Perforation zwischen diesen.

1.3 Animationen

Animationen treten an verschiedenen Stellen auf. Zum einen wird das Logo im Menü animiert, während der Spieler auf weitere Mitspieler wartet. Dadurch soll erkennbar sein, dass die Anwendung nicht abgestützt ist. Um dem Spieler anzuzeigen, welcher Spieler an der Reihe ist bzw. war und welche Spielzüge geschehen sind, werden diese ebenfalls animiert. Der grundsätzliche Ablauf der Animationsphasen während des Spiels ist in Abb. 2 zu sehen.

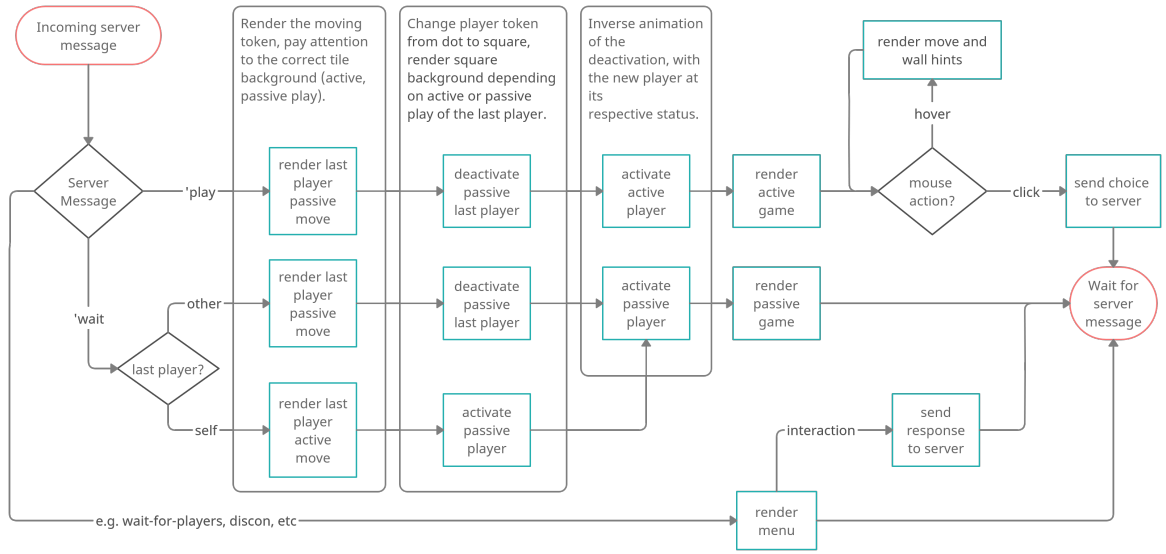

Abbildung 2: Animationsschema für Token-Bewegung und Status-Wechsel.

Einige der Pfade unterscheiden sich nur durch den Status des Spielers, welcher den Zug durchführt. Das kann entweder der konkrete Bediener sein (active), oder einer der anderen Spieler (passive). Dieser Statusunterschied wird während der Animationen durch verschiedene Farben der Kacheln für einen möglichen Zug sichtbar. Zum anderen wird dadurch das aktive, bzw. passive Spielfeld generiert. In der passiven Version bleibt dem aktuellen Spieler nur das Warten auf einen neuen Spielzug vom Server. Ist der Spieler selbst aktiv, wird die Position der Maus genutzt, um gültige Spielzüge und Platzierungen der Wände anzuzeigen. Bei einem Klick werden diese dann dem Server übermittelt. Der Client prüft hierbei, ob die Platzierungen bzw. Spielsteinbewegungen gültig sind, bevor sie abgeschickt werden können.

2 Struktur des Servers

2.1 Universum, Zustand und letzter Zug

Der Server basiert auf der Implementation der von 2htdp/universe vorgegebenen Funktionen. Dies sind eine Funktion, die auf Anmeldung verschiedener Welten antwortet (add-world), eine Funktion, die eingehende Nachrichten verarbeitet (handle-messages), und eine Funktion, die auf Verbindungsabbrüche reagiert (handle-disconnect). Außerdem nimmt der Server einen Port entgegen, über den er erreichbar sein soll und der in einer Textdatei *server-port.txt* manuell eingegeben werden kann. Das Universum selbst besitzt einen Zustand, der bei jedem Aufruf einer dieser Funktionen neu erzeugt wird. Der Zustand des Universums setzt sich aus drei Teilen zusammen: Einer Liste von Welten, einem Symbol für den Status des Universums und einer Liste mit dem letzten gültigen Zug. Die Liste mit den Welten sieht vor, dass Einträge Paare aus der iworld und einer ID sind. Diese Liste ist sortiert, das bedeutet, dass das erste Paar die aktive Welt beinhaltet, die anderen Welten sind in der entsprechenden Reihenfolge am Zug.
Der Status kann eines der folgenden Symbole sein:

- 'wait: Wartezustand des Servers
- 'finished: das Spiel ist zu Ende
- '2players: eine Zwei-Spieler-Partie
- '4players: eine Vier-Spieler-Partie.

Es gibt noch weitere Symbole, die für die Abstimmungsrunde bei zwei Spielern relevant sind:

- '2p?: Frage, ob die Spieler zu zweit Spielen wollen
- '2pw1a: Welt 1 akzeptiert auf vier Spieler zu warten (Welt 2 hat noch nicht geantwortet)
- '2pw2a: Welt 2 akzeptiert auf vier Spieler zu warten (Welt 1 hat noch nicht geantwortet).

Der letzte Teil des Zustandes beinhaltet den letzten Zug als Liste mit der ID des ausführenden Spielers, die Art des Zug (Spieler oder Mauer), Koordinaten des Zuges und der Orientierung der Mauer, falls er eine Mauer platziert hat.

Je nachdem, welche Anfrage der Server erhält wird dieser Zustand aktualisiert und die Teilnehmer über die Veränderung informiert. Wie genau die Kommunikation zwischen Client und Server abläuft ist beispielhaft im nächsten Abschnitt erklärt.

2.2 Hilfsfunktionen und Tests

Der Server greift auf eine Reihe von Hilsfunktionen zu, die in einer eignenen Datei zu finden sind. In dieser sind in erster Linie Funktionen für den Zugriff auf die einzelnen Teile des Zustandes oder von Nachrichten enthalten, aber auch Funktionen zur Erstellung von Nachrichten an den Clienten und immerhin auch ein Großteil der Message-Handler-Funktion. Eine wichtige Hilfsfunktion ist *winningMove*, die überpfüft, ob ein gegebener Zug das Spiel beendet. Außerdem gibt es noch eine Datei, die eine Reihe von Tests beinhaltet, die unter anderem prüfen, ob der neue Zustand des Universe auch immer korrekt erreicht wird, oder Nachrichten richtig erzeugt werden.

3 Kommunikation

3.1 Nachrichtenformate

Bei der Kommunikation wird zunächst zwischen Nachrichten vom Server an den Client und vom Client an der Server unterschieden. Diese haben nämlich einen leicht anderen Aufbau. Eine Mail vom Server an den Client (MailS2W) besteht aus einem Symbol (msgS2W), welche den Client beispielsweise darüber informiert, ob er am Zug ist, der ID des letzten aktiven Spielers, sowie einer Listendarstellung des letzten Zugs. Bei einer Mail vom Client an den Server (MailW2S) gibt es ebenfalls ein Symbol, welches den Server darüber informiert, ob ein Zug übermittelt, oder an einer Abstimmung teilgenommen wird, sowie eine Repräsentation des gewünschten, an alle zu übermittelnden, Zuges.

3.2 Anmeldung an den Server

Die Anmeldung an den Server ist etwas umständlich, denn es gibt eine Möglichkeit darüber abzustimmen, ob die Spieler zu zweit oder zu viert spielen wollen. Die forgende Grafik zeigt aus Sicht des Servers, wie so eine Anmeldung abläuft:

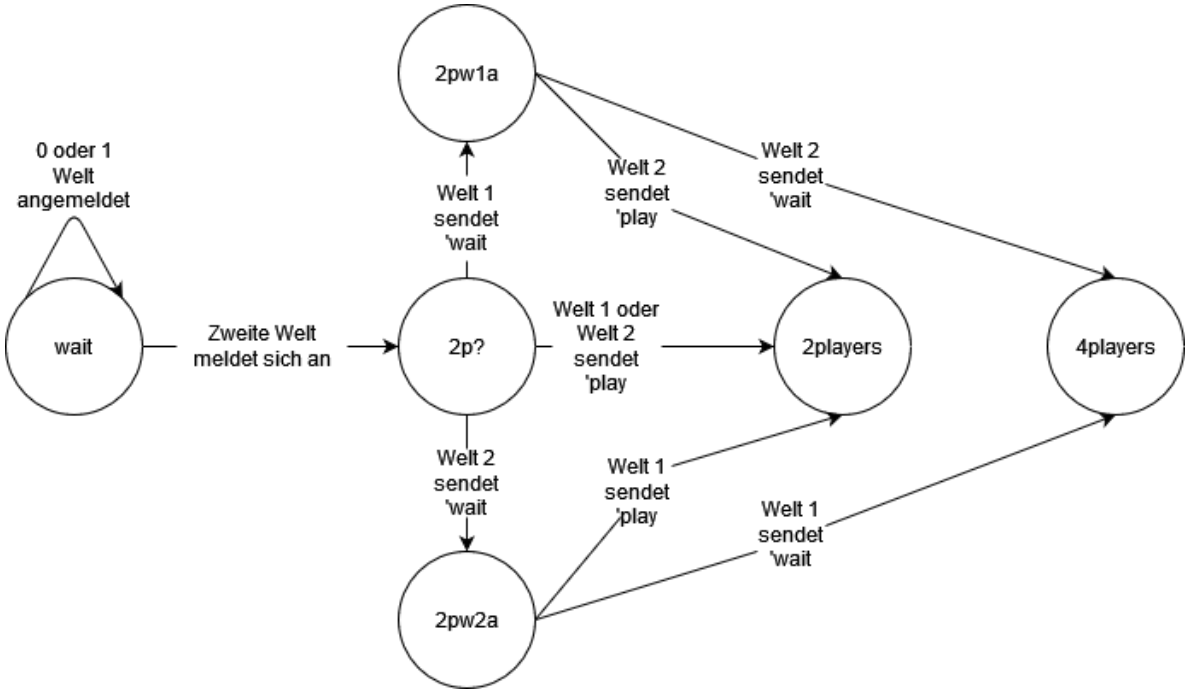

Abbildung 3: Serverstatus bei der Abstimmung zur Anzahl der Spieler

Das Wichitge hier ist, dass beide Spieler zustimmen müssen ein Vier-Spieler-Spiel zu starten; will einer der beiden Spieler nicht warten, wird sofort ein Zwei-Spieler-Spiel gestartet. Meldet sich während der Abstimmung eine weitere Welt an, so wird sie abgewiesen. Aus Kommunikationssicht sieht das dann so aus:

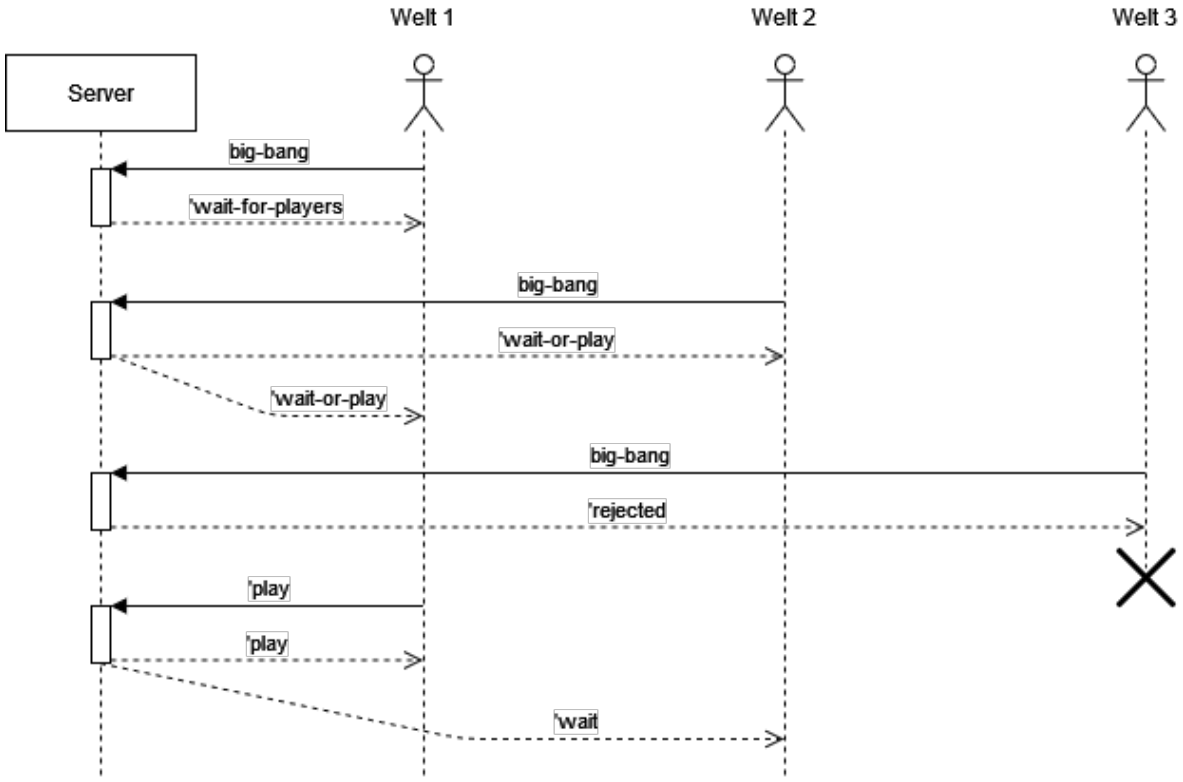

Abbildung 4: Client-Server-Kommunikation zum Spielbeginn

3.3 Übermittlung von Spielzügen

Eine Welt wird nur dann einen Zug an den Server senden, wenn sie gerade dran ist und wenn der gewünschte Zug gültig ist. Der Server überprüft nur, ob der Zug von einer Welt kommt, die auch dran ist, nicht aber, ob der Zug erlaubt ist. Dieser Zug wird dann an alle Mitspieler, auch an den, der den Zug gemacht hat, übermittelt, zusammen mit der Information, ob die eigene Welt jetzt am Zug ist und wer den letzten Zug gemacht hat. Dies könnte beispielsweise so ablaufen:

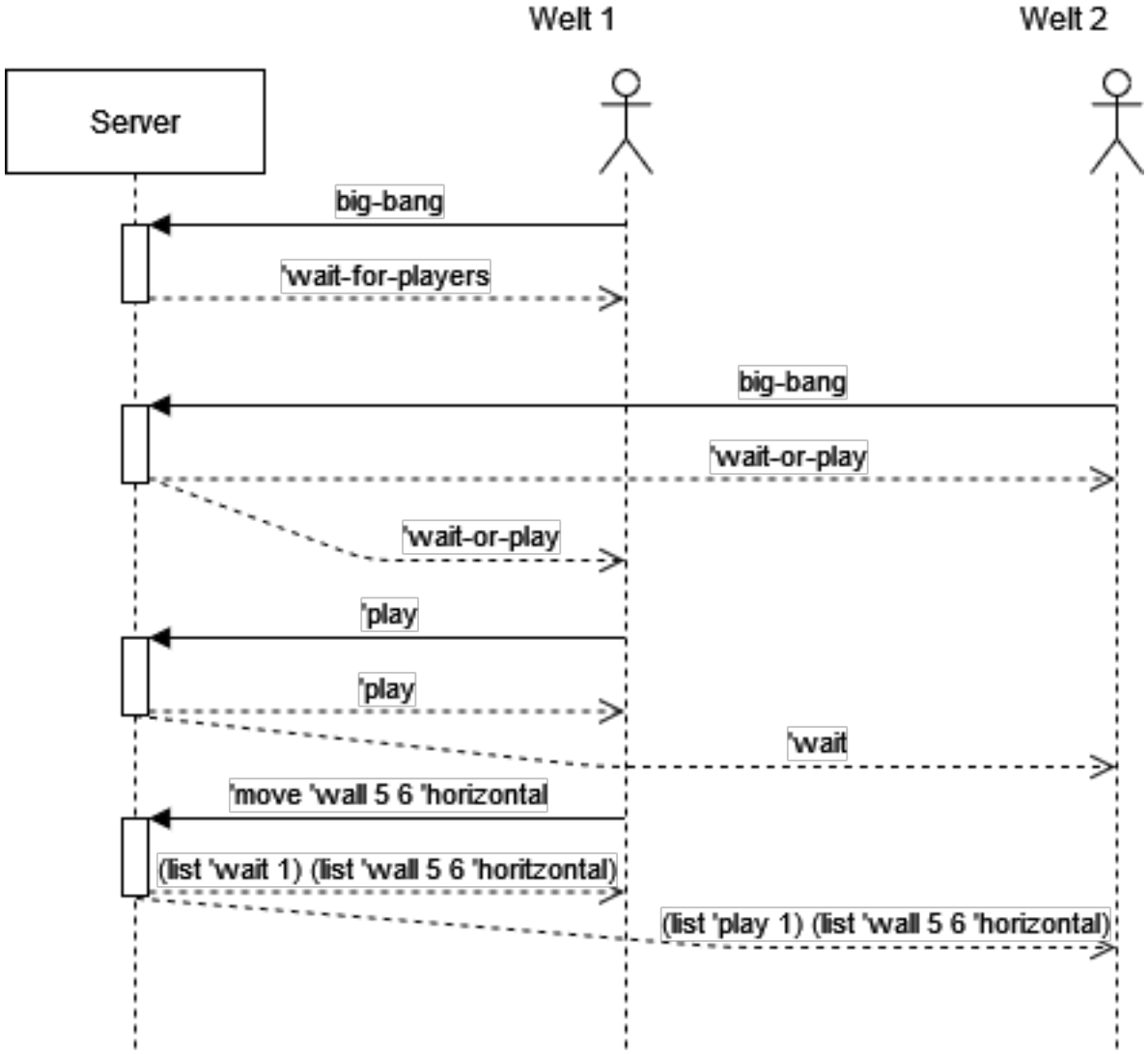

Abbildung 5: Client-Server-Kommunikation im Spiel

Es ist zu beachten, wie Welt 1, die gerade den Zug übermittelt hat, als Nachricht ein *'wait* erhält, während Welt 2 nun *'play* gesendet bekommt. Bringt ein gesendeter Zug das Spiel zum Ende, dann werden alle Teilnehmer darüber informiert, ob sie gewonnen oder verloren haben. Außerdem steht jedem Spieler dann die Möglichkeit zu, ein neues Spiel zu starten. Dies ist so implementiert, dass der Server nach beendetem Spiel im Status *'finished* verweilt und von dort aus durch ein eingehendes *'reset* von einem Spieler das Spiel für alle Welten neu startet. Bei einem Verbindungsabbruch wird das Spiel für alle Teilnehmer beendet und diese werden entsprechend durch ein *'disconnect* informiert.