Client - Server Kommunikation:

Benutzerhandbuch: nur clientseitig

Welcome – Verbinden:
Client → Server: Möchte Verbindung herstellen
Server → Client: ok, ist Möglich
Client → Server: Anfrage für Verbindung mit Username senden.
Server: Checkt username in Database, falls doppelt, hänge kleinste Zahl hinten dran (nochmal testen bis nicht vorhanden), schreibe User mit IP in Datenbank.
Server → Client: Send Back chosen Username und gibt Bestätigung für Verbindung.

Ping-Pong:
?

Dem Dokument der Software-Anforderungen nach, wird der Benutzername beim Eintreten in die Lobby festgelegt und er kann nur verändert werden, wenn man die Lobby verlässt und wieder betritt. Deshalb ist folgender Abschnitt evtl. überflüssig.

Lobby - Username wechseln:
Client → Server: Username
Server: check username in Database, falls doppelt, hänge kleinste Zahl hinten dran (nochmal testen bis nicht vorhanden), suche User anhand von IP / ID und speichere neuen Username
Server → Client: Send Back chosen Username
Server hat letztes Wort bei Vergabe des Usernamens (?), User kann zwar Username nochmals wechseln, wir aber nicht gefragt ob er hinten eine Zahl dran will (?)

Lobby - Lobby verlassen:
Client → Server : Möchte Lobby schliessen, Username

Folgendes steht nicht im Dokument Software-Anforderungen.

Lobby - Gruppe Info:
Client → Server: gib Gruppeninfo zu Gruppe1
Server: sucht Gruppe anhand von Spiel- ID oder Gruppename ??? siehe Gruppenname check (duplicates)
Server → Client: sende Liste der Mitglieder der Gruppe

Lobby - reload:
Client → Server: ask for reload, wird regelmässig (in Abstand von 5 Sekunden) gemacht.
Server: Holt Liste aller offenen Spiele, Anzahl der beigetretenen Spieler und Anzahl der möglichen Spieler, und Liste aller sich in der Lobby befindenden Spieler.
Server → Client: sende die Informationen
Client: Wenn er auf Tasten “Liste begonnener Spiele” oder “Liste beendeter Spiele” drückt

Client → Server: ask for data, Username //Muss ab jetzt der Client stets seinen username senden, damit der Server weiss, wer mit ihm spricht?
Server: Holt entsprechende Daten
Server → Client: Schickt die Daten

Erstellen:
Client → Server: Gruppenname, Anzahl Spieler, Username
Server: erstellt Spiel
Server → Client: ok, Name des Spiels
Client: weiter zu Phase “Vor Beginn”
Jede Sekunde informiert der Server den Client mit den aktuellen Spielern
Server → Client: Aktuelle Spielerliste

Lobby, Spielphasen - Spiel beitreten
Client: User klickt auf Spiel
Client → Server: sende Anfrage zum Beitreten, Spielname und Username
Server verbindet Client mit Spiel
Server → Client: ok, Name des Spiels
Jede Sekunde informiert der Server den Client mit den aktuellen Spielern
Server → Client: Aktuelle Spielerliste

Lobby, Spiel - Chat
Client möchte jemanden schreiben:
Client → Server: Wem schreiben, was schreiben.
Server → Client: ok
Jemand schreibt dem Client etwas:
Server → Client: Nachricht, wer es geschickt hat
Client → Server: ok

Von “Vor Spielbeginn” zu “Vorbereitung”
Wenn genug Leute im Chat sind, schickt der Server dem Client ein ok mit der Liste der Spielpersonen
Server → Client: Spiel beginnt, Liste der Usernames
Client → Server: ok

Spiel: Vorbereitung
Der Server schickt dem Client wichtige Informationen.
Server → Client: Farbe, Radkarten, Lang- und Kurzstrecken-Zielkarten schicken
Client → Server: ok
Client gibt Server bescheid, welche Zielkarten ausgewählt wurden:
Client → Server: ausgewählte Zielkarten
Server → Client: ok

Spiel: Während des Spiels: Spieler ist dran
Server → Client: Du bist dran
Client → Server: ok

Spiel: Während des Spiels: Radstrecke bauen

User will Radstrecke bauen (und wenn Radstrecke grau ist, wählt User Farben aus)
Client → Server: will Radstrecke xy bauen (+ Farbe, wenn Strecke grau ist)
Zuerst wird gecheckt, ob das bzgl. doppelter Radstrecke geht. Falls nicht: Fehler 1. Art

Server → Client: Geht nicht, zweite Radstrecke zu bauen, weil ihr zu wenig Spieler seid / Geht nicht zweite, Radstrecke zu bauen, weil du bereits eines davon bebaut hast.
Client → Server: ok
Dann wird gecheckt, ob genug Räder vorhanden sind oder ob genug Karten vorhanden sind.
Wenn nicht: Fehler 2. und/oder 3. Art
Server → Geht nicht, Radstrecke zu bauen, weil du nicht genug Räder und/oder nicht genug Radkarten der Farbe xy hast.
Wenn alles klappt:
Server → allen Clients: Client z baut Radstrecke xy, hat u Räder weniger, v Karten weniger und Skala erhöht sich um p Punkte
Server → spielender Client: Du verlierst v Karten der Farbe f
Sollte eine Zielkarte bei diesem Zug erreicht werden:
Server → Client: Du hast Zielkarte xy erreicht.
Client → Server: ok

Spiel: Während des Spiels: Radkarten sammeln

Spiel: Während des Spiels: Zielkarten sammeln

Spiel: Während des Spiels: Weiter zur Endphase
Sollte ein Spieler in einem Zug weniger kleiner-gleich 2 Räder haben, kommt beim Beginn des nächsten Spielzuges des folgenden Spielers:
Server → Client: Letzter Spielzug
Client → Server: ok

Spiel: Endphase:
Server → Client: Alle Daten senden
Client → Server: ok

Client-Server-Protokoll

Folgende Liste dient nur als Überblick des Protokolls. In diesem Abschnitt muss aber das Protokoll stehen.

- Spielverlauf- und Zustand (regelkonforme Züge etc.) von Server kontrolliert
- Verwaltung meherer Spielinstanzen durch Server
- Verwaltung von Liste offener Spiele durch Server
- Server sendet Client Ping-Signal und Client sendet Server Pong-Signal, um Verbindungsverluste zu erkennen und handzuhaben.
- Broadcast hat eigenen Namen
- eindeutige Definition jedes Befehls
- Protokoll erweiterbar
- Seite 9 auf den Slides

Ideen für Klassen

Player / Client:
- interne ID
- nickname
- IP
- Farbe

Chat:
- inwiefern werden Chatnachrichten in der DB gespeichert
- nach Spiel nicht mehr wichtig
- Empfang sicherstellen ... blabla

Spielzug:
- Text der im Feld "Aktivitäten" angezeigt wird
- Aktivitätsart (Zielkarte gezogen, Radkarte gezogen, Strecke gebaut) --> entsprechen Klassen mit wichtigen Infos

Strecke
- Player
- Farbe

Game
- PlayerList
- Anzahl Spieler (wie viele Spieler definiert)
- Gruppenname
- Status (open, ongoing, finished)