TICKET TO RIDE

SERVER-CLIENT SCHNITTSTELLE

Client Befehle an den Server

Client Anfrage	Server Effekt
enterLobby:[PlayerName] enterLobby:Player1	Betritt die Lobby unter diesen [unique] Namen. Wenn vergeben, wird enterLobby:null zurück gesendet Muss immer erster Befehl sein Spieler wird am Server angelegt
createGame:[gameName] createGame: game1	Spiel mit [unique] Namen wird angelegt. Wenn vergeben, wird createGame:null zurück gesendet Spieler tritt dem Spiel bei
exitGame	Verlässt das Spiel, wenn beigetreten. Wenn spiel gestartet, Punkte = 0 Wenn keine Spieler mehr im Spiel wird Spiel nach 2min gelöscht Wenn in keinem Spiel, wird exitGame:null zurück gesendet
startGame	Wenn Spieler Session-Leader (hat Spiel erstellt) und mindestens 2 spieler im spiel, wird spiel gestartet. Wenn fehlschlägt, wird startGame:null zurück gesendet.
joinGame:[gameName] joinGame:game1	Tritt dem Spiel mit [unique] Namen bei. Wenn Spiel voll, Spieler bereits in einem Spiel oder spiel bereits läuft wird joinGame:null zurück gesendet.
leave	verlässt das Spiel, die Lobby, den Server, beendet Kommunikation
sonst spielzüge -> siehe unten	

Server Satus abfragen

Jeder Client kann zu jedem Zeitunkt jeden relevanten Status des Servers abfragen.

Server Antwort: jede Zeile ein Beispiel [Variable] ausgetauscht

Allgemeine abfragen

Client Anfrage	Server Antwort
listGames	listGames:empty listGames: [nameGame1]: ... :[nameGameN] listGames:Spiel1

	listGames:Spiel1:Spiel2:Spiel3
listPlayersLobby	listPlayersLobby:null listPlayersLobby: [name1]: ... :[nameN] listPlayersLobby:player1:player2:player3 listPlayersLobby:player1
listPlayersGame:[GameName]	listPlayersLobby:[name1]: ... :[nameN] listPlayersLobby:player1:player2:player3 listPlayersLobby:player1 wenn kein Spiel dieses Names listPlayersGame:null
getGameState:[gameName]	gameState:[gameState] gameState:waitingForPlayers gameState:gaming gameState:over gameState:null (wenn kein solches Spiel)

InGame Abfragen

Client Anfrage	Server Antwort
getHandCards	getHandCards:[card1]: ... :[cardN] getHandCards:green:green:red:train getHandCards:null
getOpenCards	getOpenCards:[card1]: ... :[card5] getOpenCards:green:green:red:black:train
getMap	[RailRoadID1]:[OwnerName]; ... ;[RailRoadIDN]:[OwnerName] [optional] wenn doubleRail: [RailRoadID1]:[Owner1] :[Owner2]; 1:null;2:null;3:player1;4:player2; .....
getPoints	getPoints:[Name1]:[numPoints]: ... : [NameN]:[numPoints] getPoints:Fritzi:10:Cindy:50:Chantalle:666
getColors	getPoints:[Name1]:[color]: ... : [NameN]:[color] getPoints:Fritzi:green:Cindy:blue:Chantalle:red

Server to Client SYNC

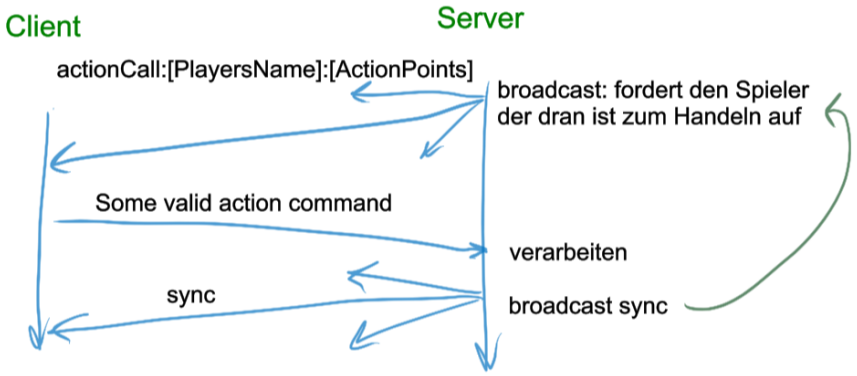
Sync wird als Broadcast gesendet um alle Clients zu informieren sich upzudaten.

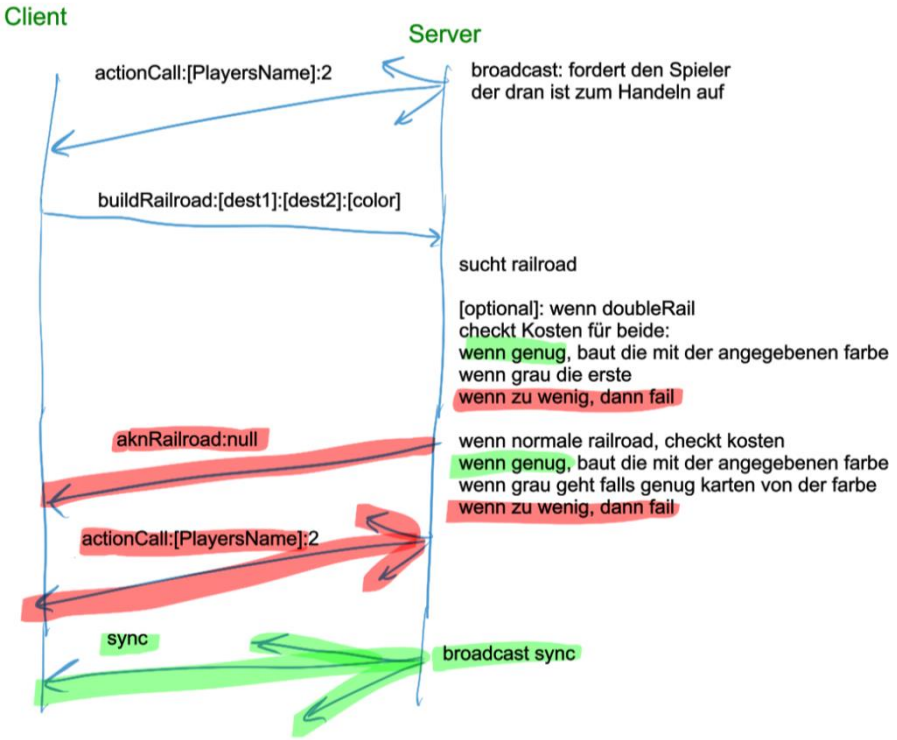
Spielzug:

Das ist die vereinfachte Game-loop.

Der Spieler kann verschiedene Aktionnen durchführen, die unterschiedliche Kosten haben.

Ein Spieler ist so lange dran, bis er keine Aktion mehr durchführen kann.



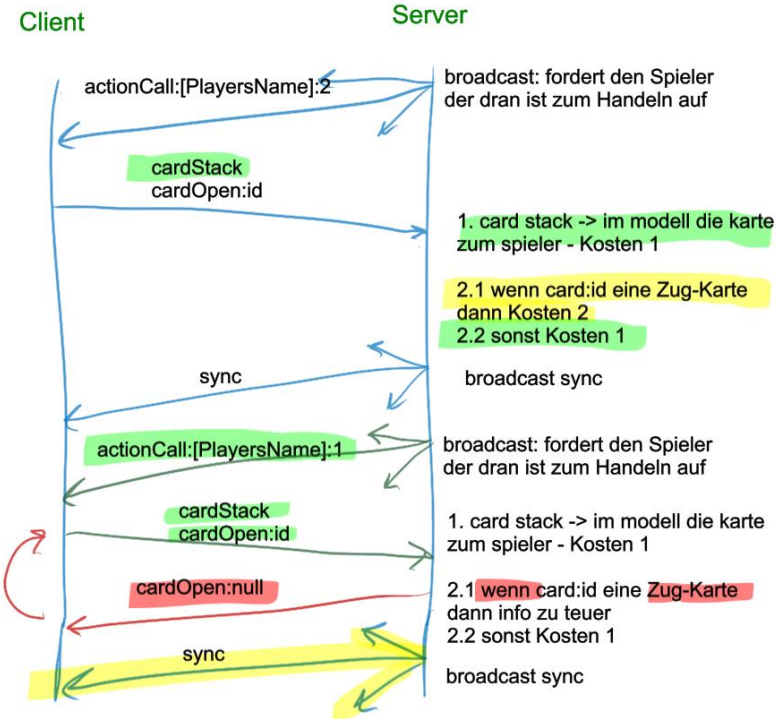


Get Card - Kosten 1 oder 2

Wenn der Spieler eine Zug karte vom offenen Stapel zieht, ist sein Zug vorbei.

Sonst kann er zwei Karten ziehen, egal ob offen oder vom Stapel

Versucht der Spieler eine Zug Karte zu ziehen, wenn er schon eine andere Karte gezogen hatte, wird "cardOpen:null" geantwortet und der Spieler muss nochmal ziehen.



Build RailRoad Ablauf - Kosten 2:

Entscheidet sich der Spieler, eine Strecke zu bauen, kann er danach keine Aktionen mehr durchführen.

Illegaler Spielzug

Spieler kann Spielzug nicht mehr leisten (wenn eine Karte gezogen wurde können keine Strecken mehr gebaut, keine Missionen gezogen und keine Zugkarten von den offenen Karten genommen werden), wird das so zurückgegeben:

[kommando]:null

Bsp client sendet	`buildRailroad:SaltLakeCity:SanFrancisco`	obwohl nicht dran
Server antwortet (nur dem Client)	`buildRailroad:null`	