Aufgabenblatt 5 – Analyse „Swarmation“

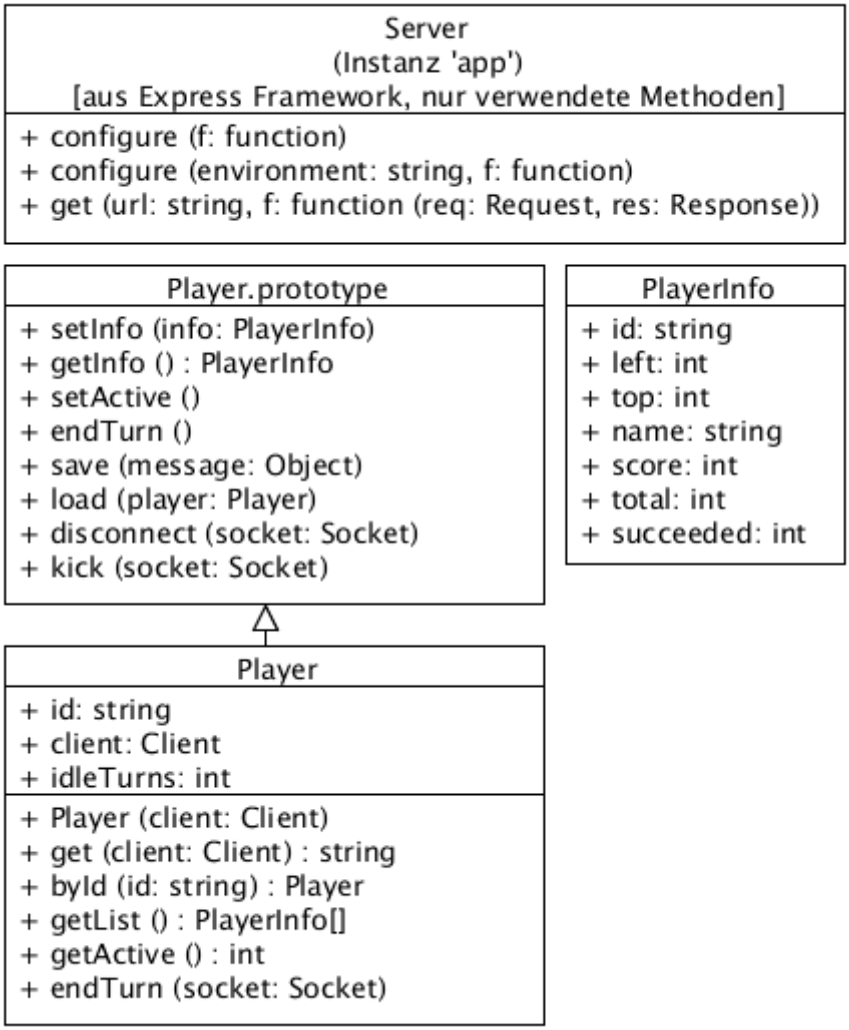
Projekt: Keynode 11
Gruppe: swp11-5
Projektleiter: Florian Golemo
Analyse: Enrico Reich & Florian Golemo

Inhaltsverzeichnis

1 Klassendiagramme

Es hat uns größte Schwierigkeiten bereitet, den JavaScript-Code in Klassen zu zerlegen, weil die Definition dafür in der Aufgabenstellung mehr als schwammig ist und die Implementierung halbherzig, was die Objektorientierung angeht. Es wurde nach besten Möglichkeiten versucht, innere Klassen sowie Beziehungen direkt darzustellen. jQuery-Eventlistener, also Objektname.bind(„Eventname“, …); sind als eigene Funktionen erfasst mit Name: „onEventname(Parameter)“ - aus Gründen der Übersichtlichkeit.

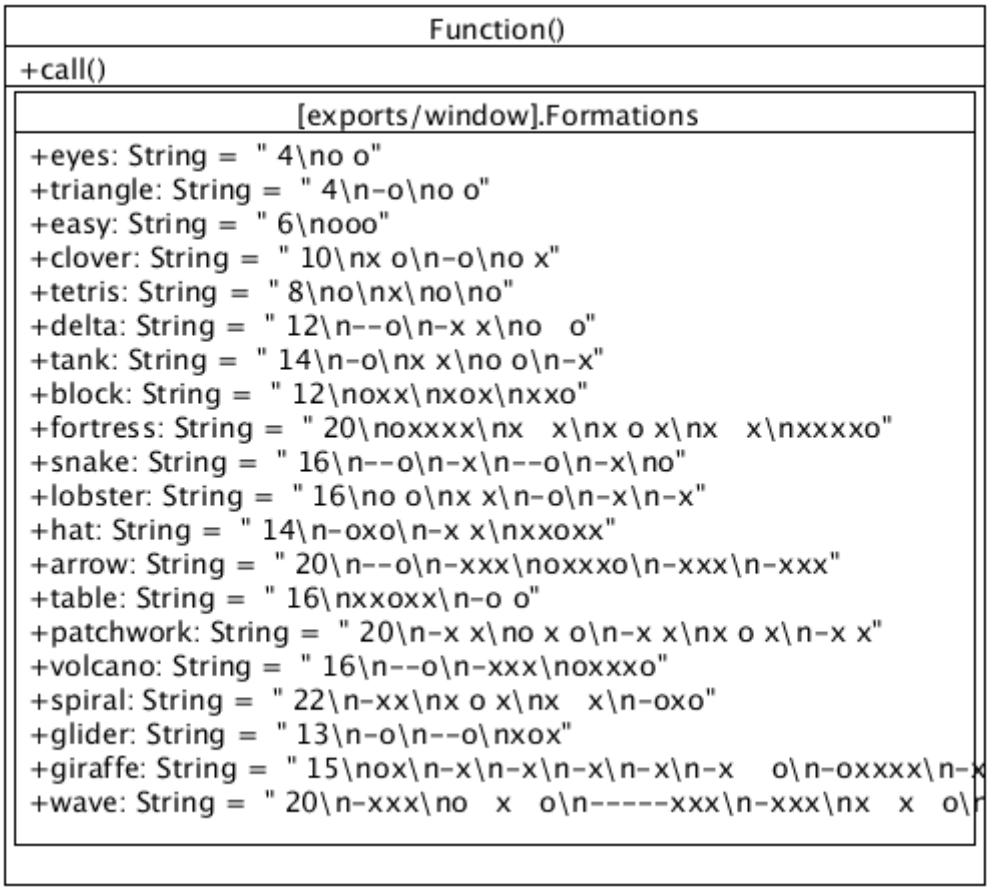
Server:
(server.js+
players.js)



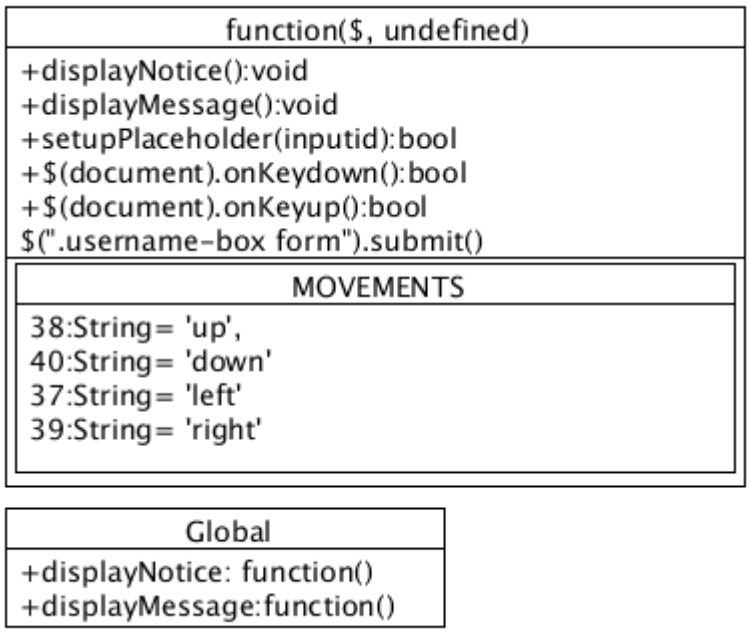
Client: (public/js/players.js)

Und folgende Diagramme wurden nur der Vollständigkeit halber erstellt und hier hinterlegt. Sie sind Helfer der beiden vorangegangenen Klassen.

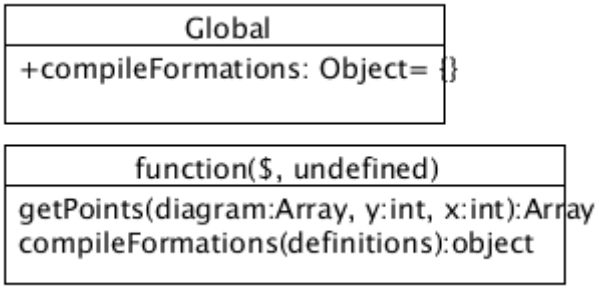
Formations: (public/js/formations.js)



Page: (public/js/page.js)



Forms: (public/js/forms.js)



2 Repräsentation der Clients im Server

Der Server erstellt zu Beginn ein leeres assoziatives Array namens „PLAYERS“, in das dann die Clients gespeichert werden: Für jeden Client wird in dem Array als Schlüssel die Session-ID des Websockets und als Wert ein Objekt der Klasse Player vom Server. Kurz, beispielhaft:

PLAYERS[sessionId] = Player();

3 Format des Nachrichtenaustauschs

Es muss zwischen zwei Arten des Nachrichtenaustauschs unterschieden werden: der Kommunikation mit dem Swarmation-Datenbank-Server (CouchDB) und der des Spiel-Servers mit den Spielern.

Ersterer Datenverkehr findet statt, um intern Statistiken zu generieren, denn es werden in jedem Browser Cookies angelegt, die ermöglichen, die Anzahl der distinkten Spieler und deren Spiele und Erfolge zu tracken. Wenn ein Cookie in dem Browser existiert, wird zu Beginn eines jeden Spiels (auch in mehreren Tabs, jedes mal neu) der letzte Stand geladen, dann fortgeführt und aller x Spiele wieder in die Datenbank gespeichert. Hinterlegt pro Spielinstanz (also pro Tab in einem Browser):

- Eindeutige Spieler-ID (pro Browser einzigartig, mittels Cookie ermittelt)
- Instanz-ID (pro Spielinstanz, wird mit jedem neuen Tab neu generiert)
- der eingegebene Spielername (sofern vorhanden, ansonsten einen zufälligen)
- Gesamtspielzahl
- gewonnene Spiele
- erreichte Punktzahl

Diese Daten werden als JSON-Objekt übertragen. Sie haben jedoch für den Spieler und den Server keinerlei Bedeutung, werden derzeit nirgendwo verwendet, weshalb wir denken, dass dies nur der Statistik für die Entwickler dient.

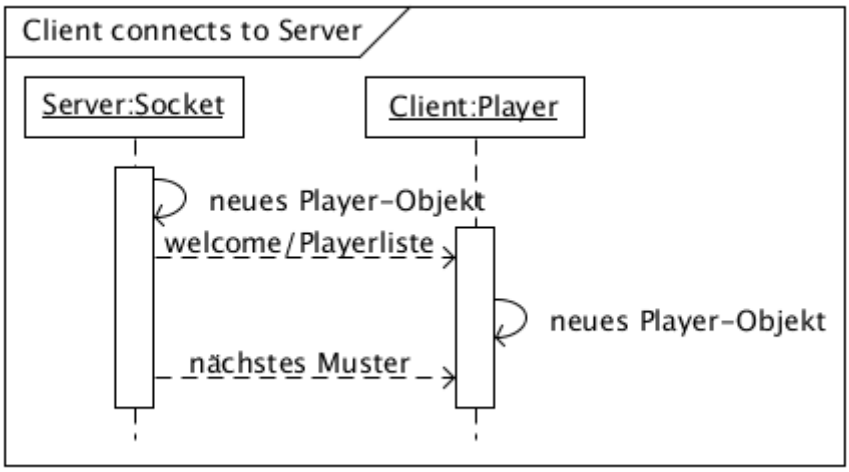
Der wichtige Datenverkehr findet zwischen dem node.js-Spiel-Server und den Client-Browsern statt: Nachrichten werden prinzipiell als JSON-Objekte gesendet. Diese Objekte enthalten immer die Eigenschaft „type“, die es ermöglicht, zu kategorisieren, um welche Art Nachricht es sich

handelt. Der Rest des Objektes ist nicht eindeutig, kann ein oder mehrere Eigenschaften sein, nie jedoch Methoden. Die Behandlung des JSON-Nachrichten-Objektes verlässt sich auf Client- und Serverseite nur darauf, dass der „type“ richtig ist und sucht dementsprechend nach Eigenschaften, die er durch den Typ im Objekt vermutet.

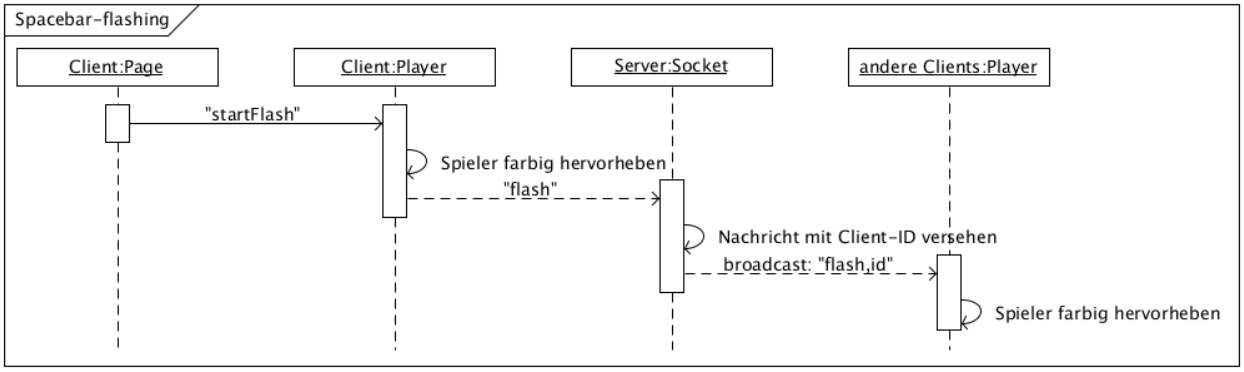
Diese Übertragungsweise JSON-Objekt allgemein hat den Vorteil, dass man sehr komfortabel auf die übertragenen Daten zugreifen kann, bei akzeptablem Sicherheitsrisiko gegenüber dem „klassischen“ Weg der Übermittlung als String mit anschließender manueller Zerlegung. Die Methode, nicht eine feste Anzahl von Eigenschaften und Eigenschaften mit unterschiedlichen Namen mit zu schicken erachten wir allerdings nicht für sinnvoll und sind der Ansicht, das die Nachricht, die ausgetauscht wird, ein universelles Format haben sollte, da dies mehr Sicherheit mit sich bringt und homogene Verarbeitung ermöglichen würde. (z.B. ist die save(message)-Methode auf dem Server in der Klasse Player.protoype nicht sicher gegen eventuelle Überladung des Objektes und so könnte die Datenbank-Verbindung angegriffen werden).

4 Nachrichtenaustausch – Beispiele:

4.1 Neuer Client verbindet sich mit Server

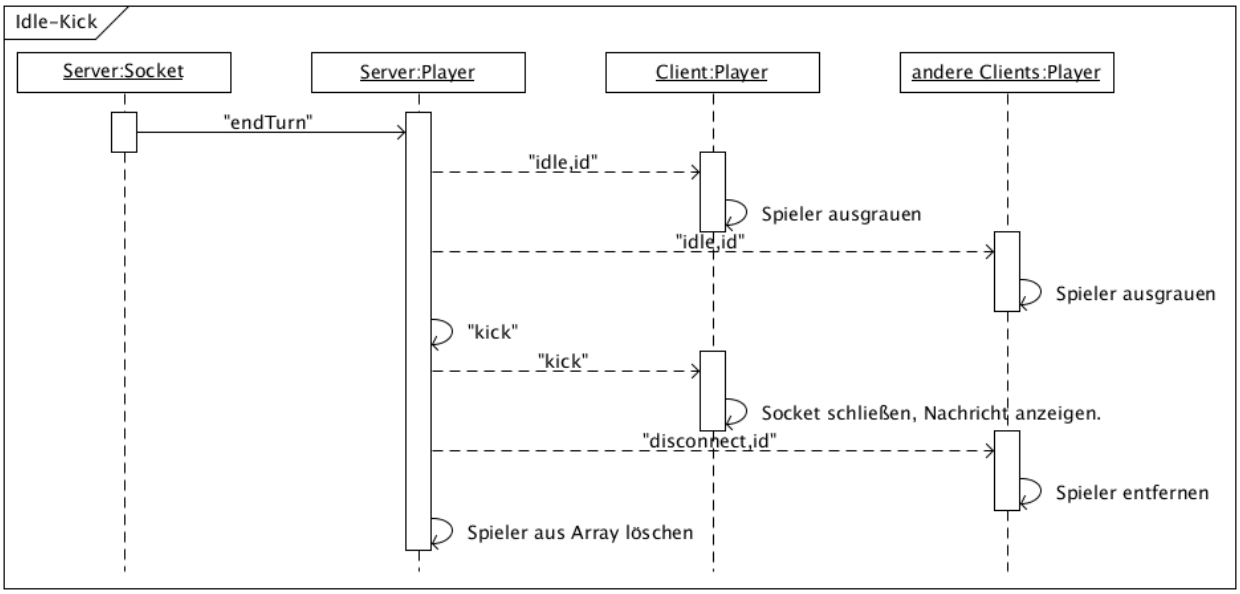


4.2 Leertaste gedrückt auf einem Client bis Aufleuchten des entspr. Spielers bei den anderen Clients



„startFlash“ wird von einem Eventlistener am Keyboard aufgerufen.

4.3 Client reagiert einige Runden nicht und wird gekickt



„endTurn“ wird aller x Sekunden (nach jeder Runde) automatisch aufgerufen und der ganze Block dient dem regelmäßigen Wechsel der Muster und der Bepunktung.