API of websockets for gameinformation

Version	Datum	Verfasser	Änderungen
0.1	29.06.2015	Jana Klemp	initial
0.2	05.06.2015	FL	United Channels added
0.21	08.06.2015	FL	Change of Channel QuestEvent -> Return contains an array under content, instead of a object

Server andPort right now:

http://giv-mgl.uni-muenster.de:3030

Entweder:

ID required and additional optional,

or all objects required, old plus changes, → Whole object is sent

API for this channels:

- Player
- MapItem
- InventarItem
- Quest
- QuestEvent

Websocket Channel: ‘Player’ **(do not use now)**

Client -> Server Messages:

Update position of a player

```
{
    x: double,
    y: double
}
```

send coordinates from client to “position” object with x and y

Server -> Client Messages:

JSON object:

```
{
"_id": Integer,  //needed
“roleName”: String,
"position": GEOJSON (Point),
“buffer”: Double,
```

"properties":[{"playerIcon":"URL","type":"player"}] }
enabled/slot?

Websocket Channel: ‘MapItem’

send MapItem from Server to the client of the active quest

send delete command from server to client

[

{

“operation”: “visible” | “add” | “remove”,

item: { //MapItem like DBSchema

"_id":Integer, //needed

“name”: String,

"visible": Boolean,

"position": GEOJSON (Point),

“buffer”: Double,

“icon”: String, (URL)

"property":[{"text": String}]

  }, ...

]

icons(URL)? visible?

Websocket Channel: ‘InventarItem’

send from client to Server “used” with ItemID

[

{

"_id": Integer, //needed

“operation”:[new, exists]

“name”: String,

"exists": Boolean,

“icon”: String(URL),

“actions”: [see action],

"property":[{"text": String}] add

},
{
"_id": Integer,
"exists": Boolean
}
]
Actions?
action: {
type: String,
timeAktion: { “ wait”: Boolean, countdown: Integer, “startTime”: Boolean, “stopTime”: Boolean},
progressAction: {“start”: Boolean, “unlock”: Boolean, “finish”: Boolean, “update”: Boolean, “questName”: String},
objectAction:{add: Integer, ressourceName: String, decreaseResource: Integer, itemName: String, addItem: Boolean, removeItem: Boolean, placeItemOnMap: Boolean},
groupAction:{groupName: String, setVisibility: Boolean }
}
groupName in datenbank

[
{"ID":12345,
"visible":true,
"property":{"name":"Ware","quantity":20,"type": "player","icon":"URL", "Käse":4, "Brot":1} },
]

Websocket Channel: ‘Quest’

send ID, recieve quest with title, description, finished, available

Returns quests as a list.

[
{
"_id": Integer,
“operation”:[available,started,finished],
"title": String,
"description": {“name”: String, “URL”: String, type: "image" | "video" | "sound" | "text", html: String}
},
{"ID":2334,

"available":false

}

]

Websocket Channel: ‘QuestEvent’

(was “Sequences” before)

[

{"_id": Integer,

“title”: String,

sequences: [{“name”: String, “URL”: String, type: "image" | "video" | "sound" | "text",

html: String}]

}

]

actions? (don’t send)

Summary:
As headings the channels of the sockets were choosen
S→C from Server to Client
S←C from Client to Server

authenticate

The first thing to do after a connect is to authenticate. This should be done by using the chanel “authenticate”.

S←C

{

access_token: String,

gameSessionID: String

}

Quest

S→C

Message if quest is finished

{

operation: “finished”,

quest: questID

}

S→C

Message if quest gets available

{

operation: “available”

quest: {_id: questID, description: CONTENT}

}

S←C
{
operation: “active”,
quest: questID
}

MapItem

S→C
{
operation: “add”,
item: GeoJSON+attributes from MapItem
}

S→C
{
operation: “remove”,
item: itemID
}

Item

S→C
{
operation: “removeFromInventar”,
item: itemID,
amount: int
}

S→C
{
operation: “addToInventar”,
item: itemID,
amount: int
}

C←S
{
operation: “use”,
item:itemID
}

Property

C←S

{
operation: “getProperty”,
property: propertyID
}

QuestEvent

S→C
{
content: [CONTENT]
}
Note: this returns an array of contents. Because the model defines that there can be multiple content entries for a questevent.
This entries should follow an given order (e.g. simulate an dialog between two character)