On The Use Of Deliveroo.js APIs

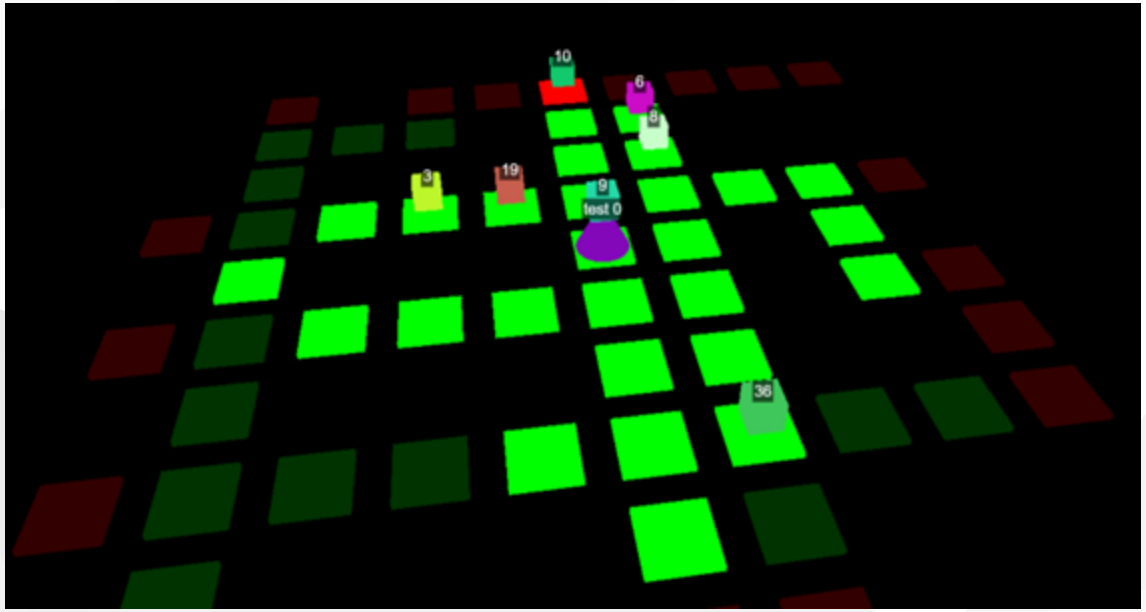
Autonomous Software Agents - Lab

Marco Robol - marco.robol@unitn.it

Contents

- **Deliveroo.js** (game server + built-in 3D WebApp)
- **Development Toolkit**
  - Client library
  - Example of a scripted player
- **APIs documentation**
  - Socket.IO events emitted and listened by the game

Deliveroo.js



> Deliveroo.js is simple parcel-deliverying game for educational purposes.

Play on the cloud: https://deliveroojs.onrender.com/
https://deliveroojs2.onrender.com/ https://deliveroojs3.onrender.com/

Running your own Deliveroo.js game server

- `$ git clone https://github.com/unitn-ASA/Deliveroo.js.git`
- `$ npm install`
- `$ node index.js level_1` or `level_2`, `level_3`
- Or `$ npm run dev` to load environment variables from .env file
- Go on http://localhost:8080, insert name, get your new token or use previous one stored in the browser cookies, play.

Setup environment variables (optional)

`.env`

```
# Passphrase used to generate jwt token
# SUPER_SECRET='default_token_private_key'

# Web server port; 8080 if not specified
# PORT='8080'
```

`index.js`

```
const PORT = process.env.PORT || 8080;
```

`src\deliveroo\Authentication.js`

```
const SUPER_SECRET = process.env.SUPER_SECRET || 'default_token_private_key';
jwt.sign( {name}, SUPER_SECRET ); jwt.verify( token, SUPER_SECRET );
```

Game configuration

`config.js`

Built-in 3D WebApp (client)

In addition to exposing APIs, the game serves a built-in 3D WebApp. See http://localhost:8080.

`Deliveroo.js\static\deliveroo.js` - Here is how the WebApp interact with the game.

```
socket.on( "connect", callback )
socket.on( "disconnect", callback )
socket.on( "token", callback )
socket.on( "tile", callback )
socket.on( "you", callback )
socket.on("agents sensing", callback )
socket.on("parcels sensing", callback )
socket.emit('pickup', callback )
socket.emit('putdown', callback )
socket.emit('move', 'up', callback )
```

Deliveroo.js Development Toolkit

`./packages/@unitn-asa/deliveroo-js-client`

Distributed as a npm package through the npmrc.

```
$ npm install @unitn-asa/deliveroo-js-client
```

Deliveroo.js Demo Agent Project

- `$ git clone https://github.com/unitn-ASA/DeliverooAgent.js`

This project contains an empty structure that you can use to start implementing your agent. In addition it includes examples.

To start developing your code, create a `\src` folder at the root of the project

Deliveroo.js dev-kit libraries are in main repo Deliveroo.js and here installed as dependencies.

Let's check out examples of scripted players

Clone repository `$ git clone https://github.com/unitn-ASA/DeliverooAgent.js` and install dependencies `$ npm install`. Then, setup **host** and **token** in `config.js`, get a valid token using the WebApp.

There are two implementation of a randomly moving agent, one implemented on top of raw socket.io messages and another implemented on top of `DeliverooApi.js` client.

```
$ node demo demo_agent_socket
$ node demo demo_agent_client
```

Open the browser and observe the scripted agent moving randomly.

APIs documentation

- Authentication
- Map
- The player
- Sensing parcels
- Sensing other agents
- Actions

APIs - Authentication - server side

Deliveroo.js\src\server.js

```
io.on('connection', (socket) => { const me = myAuthenticator.authenticate(socket); })
```

Deliveroo.js\src\deliveroo\Authentication.js

```
function authenticate (socket) {
    var token = socket.handshake.headers['x-token'];
    var name = socket.handshake.query.name;
    if ( !token || token=="" ) {  // Signup, no token provided, generate new one
        token = jwt.sign( {name}, SUPER_SECRET );
        socket.emit( 'token', token, name );
    } else {                      // Login, token provided, validate
        try { var decoded = jwt.verify( token, SUPER_SECRET );
        } catch(err) { socket.disconnect(); return; }
    }                             // ...create or retrieve agent given the token
    console.log( `Socket ${socket.id} login/signup as ${agent.name}
                  Token last 5 digits ${token.slice(-5)}`           );
    return agent;
}
```

APIs - Authentication - client side

`DeliverooAgent.js\src\DeliverooApi.js`

```
socket = io( config.host, {
    extraHeaders: {
        'x-token': config.token
    },
    // query: {
    //     name: "scripted",
    // }
});
```

`DeliverooAgent.js\config.js`

```
module.exports = {
    host: 'http://localhost:8080',
    token: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoic2NyaXB0ZWQiLCJpYXQiOjE2NzY3MjU5Mjl9.338MYxzhV6VLDuJ9YS1HhI3sUjUFZ7Pfhm9l31np5lc'
}
```

APIs - Game Configuration and Map

`./config.js` and `./levels/maps/default_map.js`

```
var myMap = [ [1, 1, 1],   // '0' are blocked tiles
              [0, 0, 0],   // '1' are walkable tiles
              [1, 1, 1] ]
```

`/src/ioServer.js`

```
for (const tile of myGrid.getTiles()) {
    if ( !tile.blocked )
        socket.emit( 'tile', tile.x, tile.y, tile.delivery )
}                           // '0' on borders are always 'delivery' tiles
```

`./packages/@unitn-asa/deliveroo-js-client/lib/DeliverooApi.js`

```
this.socket.on( "tile", ( x, y, delivery ) => { ... } );
```

APIs - The player

Server - `Deliveroo.js\src\server.js`

```
// Send every time I move or my score changes
me.on( 'agent', ({id, name, x, y, score}) => {
    socket.emit( 'you', {id, name, x, y, score} );
} );
// Send on initialization
socket.emit( 'you', {id, name, x, y, score} = me );
```

Client - `DeliverooAgent.js\src\DeliverooApi.js`

```
this.socket.on( "you", ( {id, name, x, y, score} ) => { ... } );
```

APIs - Sensing parcels

Server - `Deliveroo.js\src\server.js` - 'parcels sensing' is emitted every time the player move or parcels reward timer decades

```
me.on( 'parcels sensing', (parcels) =>  socket.emit('parcels sensing', parcels) );
me.emitParcelSensing(); // Trigger on initialization
```

```
// Deliveroo.js\src\deliveroo\Agent.js - Agent.emitParcelSensing:
var parcels = [];
for ( const parcel of this.#grid.getParcels() ) if ( Xy.distance(parcel, this) < 5 )
        parcels.push( { id: p.id, x: p.x, y: p.y, reward: p.reward,
            carriedBy: ( p.carriedBy ? p.carriedBy.id : null ) } )
this.emit( 'parcels sensing', parcels )
```

Client - `DeliverooAgent.js\src\DeliverooApi.js`

```
this.socket.on( "parcels sensing", parcels => { ... } ); // [ {id, x, y, carriedBy, reward} ]
```

APIs - Sensing other agents

Server - `Deliveroo.js\src\server.js` - 'agent sensing' is emitted every time players move or the player itself move

```
// Send agents every time I move or parcels reward timer decades
me.on( 'agents sensing', (agents) => socket.emit('agents sensing', agents) );
me.emitAgentSensing(); // Trigger on initialization
```

Client - `DeliverooAgent.js\src\DeliverooApi.js`

```
this.socket.on( "agents sensing", agents => { ... } ); // [ {id, name, x, y, score} ]
```

APIs - Actions

Server - `Deliveroo.js\src\server.js`

```
socket.on('move', async (direction, acknowledgementCallback) => {
    try { acknowledgementCallback( await me[direction]() ); } catch (err) {  }
});
socket.on('pickup', async (acknowledgementCallback) => {
    try { acknowledgementCallback( me.pickUp() ) } catch (err) {  }
}); // same for putdown
```

Client - `DeliverooAgent.js\src\DeliverooApi.js`

```
async move ( direction ) {
    return new Promise( (success, reject) => {
        this.socket.emit( 'move', direction, async (status) =>  ( status ? success() : reject() )
    } );
}
async pickup (  ) {
    return new Promise( (success) => {
        this.socket.emit( 'pickup', async ( picked ) => success(picked) );
    } );
} // same for putdown
```

Questions?

marco.robol@unitn.it

> Deliveroo.js: https://github.com/unitn-ASA/Deliveroo.js
> DeliverooAgent.js: https://github.com/unitn-ASA/DeliverooAgent.js