On The Use Of Deliveroo.js APIs

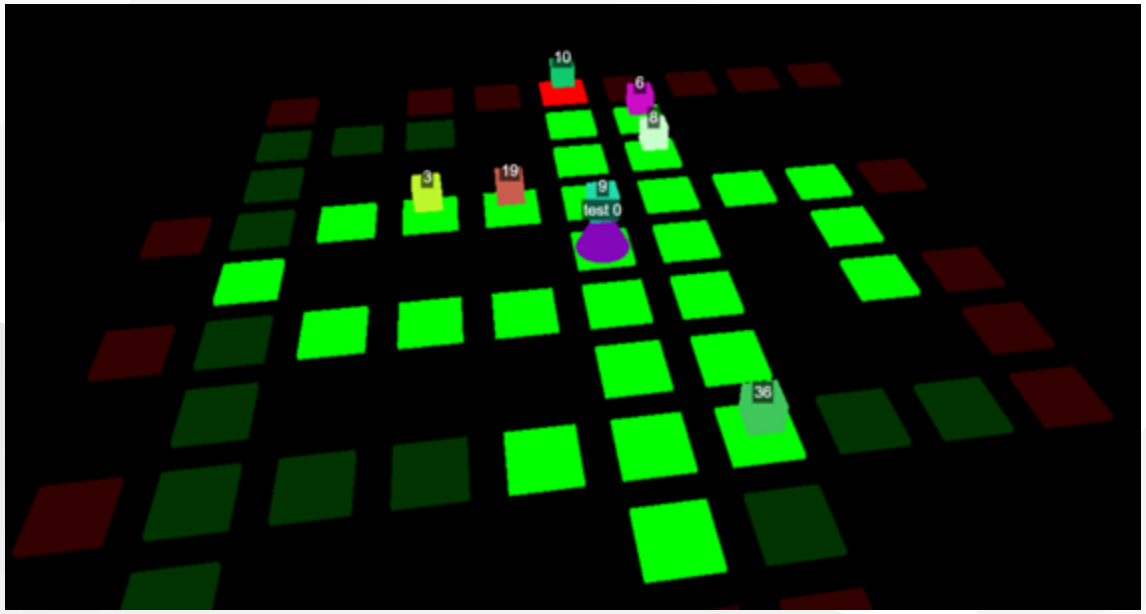
Autonomous Software Agents - Lab

Marco Robol - marco.robol@unitn.it

Contents

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

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



Play on the cloud at https://deliveroojs25.azurewebsites.net/

Or from within the unitn network at https://deliveroojs2.rtibdi.disi.unitn.it/

For that, you will need the unitn-vpn https://servicedesk.unitn.it/sd/it/service/vpn?id=unitrento_v2_service_card&sys_id=e5a898fec35bfd104cbb7055df013171

Running your own Deliveroo.js game server

- `$ git clone https://github.com/unitn-ASA/Deliveroo.js.git`
- `$ npm install` and `$ npm run build`
- `$ npm start -- -- -l ./levels/level_1.js` or any other config file in ./levels
  `$ npm start -- -- -h` for help on additional configuration
  Or set environment variables from `.env` file:

```
# Web server port; 8080 if not specified
PORT='8080'
# File in ./files to load a specific configuration
LEVEL='./levels/level_1.js'
```

- Go on http://localhost:8080 create your token and play

Game configuration `config.js`

```
const config = {
    MAP_FILE: 'default_map',              // files in levels/maps

    PARCELS_GENERATION_INTERVAL: '2s',  // '1s', '2s', '5s', '10s'
    PARCELS_MAX: 'infinite',              // number or 'infinite'

    MOVEMENT_STEPS: 1,                    // 1 intermediate position at 0.6
    MOVEMENT_DURATION: 50,                // time in ms
    AGENTS_OBSERVATION_DISTANCE: 5,     // number or 'infinite'
    PARCELS_OBSERVATION_DISTANCE: 5,    // number or 'infinite'
    AGENT_TIMEOUT: 10000,                 // time in ms

    PARCEL_REWARD_AVG: 30,                // number
    PARCEL_REWARD_VARIANCE: 10,           // number
    PARCEL_DECADING_INTERVAL: '1s',     // '1s', '2s', '5s', '10s', 'infinite'

    RANDOMLY_MOVING_AGENTS: 2,            // number
    RANDOM_AGENT_SPEED: '2s',             // '1s', '2s', '5s', '10s'

    CLOCK: 50                             // (50ms are 20frame/s)
}
LEVEL = process.argv[2] || process.env.LEVEL; // load config from file in ./levels
```

Game map configuration

`./levels/maps/default_map.js`

```
var myMap = [ [2, 3, 2],   // '0' are blocked tiles (empty or not_tile)
              [0, 3, 0],   // '1' are walkable spawning tiles
              [1, 3, 1] ]  // '2' are delivery tiles
                           // '3' are walkable non-spawning tiles
```

Deliveroo.js Development Toolkit

https://github.com/unitn-ASA/Deliveroo.js/tree/master/packages/%40unitn-asa/deliveroo-js-client

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

Distributed as a npm package through the npmrc.

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

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

```
socket.on( "connect", ()=>{...} )
socket.on( "disconnect", ()=>{...} )
socket.on( "token", (token)=>{...} )
socket.on( "config", (config)=>{...} )
socket.on( "map", (width, height, tiles)=>{} )
socket.on( "tile", ( {x, y, type} )=>{...} )
socket.on( "you", ({id, name, x, y, score})=>{...} )
socket.on("agents sensing", ([{id, name, x, y, score}])=>{} )
socket.on("parcels sensing", ([{id, x, y, carriedBy, reward}])=>{} )
socket.emit('pickup', acknowledgmentCallback )
socket.emit('putdown', acknowledgmentCallback )
socket.emit('move', 'up', acknowledgmentCallback )
```

Deliveroo.js Demo Agent Project

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

- `$ 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\ioServer.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: '...sUjUFZ7Pfhm9l31np5lc'
}
```

APIs - Game Configuration

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

```
/**
 * Listen to 'config' events from server
 * @type {function(config):void}
 */
onConfig ( callback ) {
    this.socket.on( "config", callback )
}
```

APIs - Map

`/src/ioServer.js`

```
for (const tile of myGrid.getTiles())
    if ( !tile.blocked )
        socket.emit( 'tile', tile.x, tile.y, tile.type )
socket.emit( 'map', width, height, myGrid.getTiles() )
```

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

```
onTile ( /** @type { function( x, y, type ) } */ callback ) {
    this.socket.on( "tile", callback );
}
onMap ( /** @type { function( width, height, [{x, y, type}] ) } */ callback ) {
    this.socket.on( "map", callback )
}
```

APIs - The player

Server - `Deliveroo.js\src\ioServer.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 - `DeliverooApi.js`

```
/**
 * Listen to 'you' events
 * @param {function({id:string, name:string, x:number, y:number, score:number})} callback
 */
onYou ( callback ) {
    this.socket.on( "you", callback )
}
```

APIs - Sensing parcels

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

```
// 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 - `DeliverooApi.js`

```
/**
 * @param { function( [ { id, x, y, carriedBy, reward } ] ) } callback
 */
onParcelsSensing ( callback ) {
    this.socket.on( "parcels sensing", callback )
}
```

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 - `DeliverooApi.js`

```
/**
 * @param { function( [ { id:string, name:string, x:number, y:number, score:number } ] ) } callback
 */
onAgentsSensing ( callback ) {
    this.socket.on( "agents sensing", callback )
}
```

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 emitMove ( direction ) {
    return new Promise( (success, reject) => {
        this.socket.emit( 'move', direction, async (status) =>  ( status ? success() : reject() )
    } );
}
async emitPickup (  ) {
    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