Palim Palim

Palim-Palim is a peer-to-peer video game for children and the elderly. It allows two players to play pretend store in a virtual environment. The client side is built using Three.js to provide an interactive 3D environment. It features a WebRTC video chat to promote communication between players. The project uses a node server running socket.io to provide WebRTC signaling. The multiplayer functionality is implemented using WebRTC peer-to-peer datachannels instead of an authorative server.

Project Structure

```
build - Directory for built and compressed files from the npm build script
src - Directory for all dev files
├── css - Contains all SCSS files, that are compiled to `src/public/css`
├── js - All the Three.js app files, with `app.js` as entry point. Compiled to `src/public/js` with webpack
|   ├── game
|   |   ├── components - Various compontents used by the game
|   |   └── config - Different scene configurations to be loaded
|   ├── utils - Helpers and vendor classes
|   └── videoChat - WebRTC peer connection for video chat and other data channels
└── public - Used by webpack-dev-server to serve content. Webpack builds local dev files here.
    └── assets - Is copied over to build folder with build command. Place external asset files here.
```

Getting started

Install node.js (version 14 LTS) and install the dependencies with npm:

```
npm install
```

Then build the production files with:

```
npm run build
```

This cleans existing build folder while linting js folder and copies over the public assets folder from src. Then sets environment to production and compiles js and css into build.

```
npm start
```

Spins up the node.js server at localhost:8080 and uses the current prod build from the build folder. Open a second tab to chat and play the game with yourself.

Running App in dev mode (wihout Server/VideoChat)

After installing the dependencies, run:

```
npm run dev
```

This will spin up a webpack dev server at localhost:8080 and keeps track of all js and sass changes to files. Useful for developing the Three.js-scene but does not start the node.js server, so the video chat can not be tested (for this you have to build it and then start the node.js server locally - as described above). Press H to show or hide the dat.GUI menu.

All NPM Scripts

You can run any of these individually if you'd like with the `npm run` command: * `build` - Cleans build folder, lints js code, copies assets, sets env to production, compiles everything into build * `start` - Starts `server.js` at localhost:8080 serving the current prod build from the build folder * `test` - Runs the Mocha tests defined in `test/test.js` * `prebuild` - Cleans up build folder and lints `src/js` * `clean` - Cleans build folder * `lint` - Runs lint on the `src/js` folder and uses the `.eslintrc` file in root for linting rules * `webpack-server` - Starts up a webpack-dev-server with hot-module-replacement * `webpack-watch` - Runs webpack in dev environment with watch * `dev:js` - Runs webpack in dev environment without watch * `build:dir` - Copies files and folders from `src/public` to `build` * `build:js` - Runs webpack in production environment

Deployment

The project can be built with webpack and deployed in any node.js environment. For convenience, Heroku was used as a deployment service during development. Deployment on Heroku is pretty well documented, and normally Heroku will recognise the app as a node.js application and use the proper buildpack for the deployment. If you encounter any problems, try following steps: * Set up Heroku CLI * Ensure the Heroku application is using the `heroku/nodejs` buildpack by running the `heroku buildpacks -a <your-heroku-app-name>` command. More infos * If it is not set to `heroku/nodejs`, set the buildpack with the command `heroku buildpacks:set heroku/nodejs -a <your-heroku-app-name>`. More infos * With the proper buildpack set up, Heroku will automatically install all dependencies and will start the server using `npm start` when deploying the app.

Setting up a TURN server

While running and deploying the project locally works just fine, it will need a dedicated TURN server as fallback, if you want to use WebRTC over the internet. This means to deploy the project for production a TURN server must be setup and provided. To host your own TURN server, a Linux server with a public IP address is needed. The TURN server used in Palim-Palim uses Coturn, which is an open source implementation of the TURN protocol. Specifications of the server are (not mandatory): * 8 GB RAM * 8 VCPU * Memory: 20 GB * Operating system: Linux Ubuntu * Depending on the environment on which the server is set up, the ports of the TURN server must be opened. With the TURN configuration below, we had to open the following ports on our SwitchEngines Server: * TCP and UDP, in and out: 10000 to 20000 * TCP, in: 3478

Instructions for the Palim-Palim TURN server based on Gabriel Tanner's instructions:

1. Update the linux server
   `sudo apt-get update -y`
2. Install Coturn
   `sudo apt-get install coturn`
3. Define Coturn as an automatic service, so that it automatically starts up again after a server restart
   Comment in `TURNSERVER_ENABLED=1` in the file at `/etc/default/coturn`
4. Start Coturn service
   `systemctl start coturn`
5. The turn server configuration is located at `/etc/turnserver.conf` and must be set as follows:
   See example file from this project `turnserver.config`
6. Restart coturn service
   `sudo service coturn restart`
7. The turn server functionality can be tested with the Trickle ICE examples page. To do this, the following parameters must be specified:
   STUN or TURN URI `URI from turnserver.config` , for example `turn:86.119.43.130:3478`
   TURN username `username from turnserver.config`
   TURN password `password from turnserver.config`
8. Now your TURN-Server is ready, and can be passed along as a parameter while setting up the RTCPeerConnection. See file `peerConnection.js` as a reference how to configure the RTCPeerConnection.

Credits

Project team:
Severin Peyer
Daniel Obrist

Supervision:
Marco Soldati
Tabea Iseli

Fachhochschule Nordwestschweiz FHNW

This project was built using boilerplate code from:
https://github.com/paulmg/ThreeJS-Webpack-ES6-Boilerplate/
https://github.com/AidanNelson/threejs-webrtc