LIQUID GALAXY SPACE CHESS DOCS

DOCUMENTATION & USERGUIDE

Author: Pablo Sanchidrian
Mentors: Andreu Ibañez & Víctor Sánchez
Lleida Liquid Galaxy Support: Pau Francino

INDEX

CHAPTER I: General context

1. Introduction

1.1 Context

Liquid Galaxy has been selected this year, 2022, as an organization for the GSoC program, and is creating new project ideas, that contributors around the world are developing. One of them is LG Space Chess which has a strong relationship with space communications, the main collaborator is HydraSpace.

HydraSpace is dedicated to satellite IoT communications with full in-house technology, including satellites and ground-segment.

This year Liquid Galaxy and HydraSpace are joining forces to create the first chess in the world capable of playing against a satellite.

The main idea is to use the Liquid Galaxy cluster to visualize a world chess game that will happen with people worldwide and through satellite communications with the collaboration of HydraSpace & Hybridium.

There are two teams, the Earth (you) and the Space (a strong AI). Every day the Earth makes, at least, one move, the most common move among all the players. Once the Earth has made a move, wait for Space's turn, the satellite will send its move as soon as the connection with the ground station is available.

1.2 Objectives

Create the first chess in the world capable of playing against a satellite, never created before.

To do so, we must use satellite communications to exchange packages between the players and the satellite. We also need the controller, the one the users will use to play, and lastly, we need the screen visualization.

- Data exchange
- Controller
- Screen visualization

1.3 Definitions

Progressive Web Application (PWA): This is a type of application software delivered through the web, built using common web technologies including HTML, CSS, JavaScript, and WebAssembly. It is intended to work on any platform with a standards-compliant browser, including desktop and mobile devices.

NodeJS: This is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine (i.e. V8 engine) and executes JavaScript code outside a web browser, which was designed to build scalable network applications.

ThreeJS: Three.js is a cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser using WebGL.
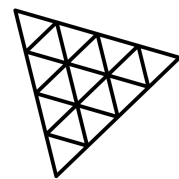
Web Socket: This is a computer communications protocol, that provides full-duplex communication channels over a single TCP connection.

Secure File Transfer Protocol (SFTP): This is a network protocol that provides file access, file transfer, and file management over any reliable data stream. It was designed by the Internet Engineering Task Force (IETF) as an extension of the Secure Shell protocol (SSH) version 2.0 to provide secure file transfer capabilities.

Google Cloud Platform (GCP): This is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products. Is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user.

Firebase: This is a platform developed by Google for creating mobile and web applications. Provide a NoSQL database, Web Hosting, Auth, and Cloud Functions, among others.

2. State of art

2.1 Current Situation

Remote chess is played in various forms, traditionally through the postal system or its updated version (email), less common methods are fax, homing pigeon, and phone. Today it is usually played throughout a dedicated server.

Chess has never been brought out to space, Liquid Galaxy has created a new way of playing chess with the collaboration of HydraSpace and Hybridium. The first chess in the world capable of playing against a satellite.

3. Plannification

3.1 Project Plannification

Preparation Phase: In this phase, it is analyzed what and how many functionalities/features will be developed concerning the time, important deadlines, and capabilities of the collaborators. We set the starting dates as well as an estimated finish date.

Design Phase: It is studied the compatibilities of the features that want to be added and make changes to them if needed. It is established the database structure, communication protocols, frameworks, and modules.

Development phase: The software development starts with all its modules and follows all the schemes already established

Deployment phase: All the built software will be deployed, in this specific phase, we will detect most of the problems that could affect the stream of development.

CHAPTER II: Development

4. Requirements Analysis

4.1 Actors

In the following lines will be presented every actor in this project:
- *Unauthenticated User*: This will be any user who has never entered the application. This type of user only has access to the login page.
- *Authenticated User*: This will be any user that has already signed up/logged in. This type of user has access to every single feature of the project
- *Time*: This will be the Scheduled cloud functions, that will execute every certain time.

4.2 Use Cases

Register/Login:
Description: Action when an unauthenticated user wishes to identify himself.
Actors: Unauthenticated user.
Precondition: The user has not authenticated.
Postcondition: The user becomes an authenticated user.
Main scenery:
1 - The system shows the login page to the user
2 - The user selects the option "Google auth"
3 - The system shows a google account selector
4 - The user selects the google account
5 - The system authenticates the user and redirects the user to the dashboard

Signout:
Description: The user signs out from the app
Actors: Authenticated user
Precondition: The user must be authenticated
Postcondition: The user will signout and become unauthenticated
Main scenery:
1 - The user selects the option "Signout"
2 - The system asks the user for confirmation
3 - The user selects the option "yes"
4 - The system redirects the user to the login page

Alternative cases:
3a - The user does not want to proceed
- The user selects the option "no"
- The system closes the confirmation modal and returns to the page

Move piece:
Description: The user moves a piece of the chessboard
Actors: Authenticated user
Precondition: The user must be authenticated
Postcondition: The piece will move to its new position
Main scenery:
1 - The user drags a piece to a square
2 - The system validates the move, and updates the chessboard status
3 - The system saves the vote

Alternative cases:
2a - The move is illegal
- The system notifies the user that the move is illegal and aborts the change
3a - The game mode is "Local Play"
- The system moves the user piece and answers by moving an opponent piece
3b - The game mode is "Top play"
- The system aborts the movement

Connect:
Description: Connect the controller to the screens
Actors: Authenticated user
Precondition: None
Postcondition: The controller ends up connected to the screens
Main scenery:
1 - The user selects the option "Connect"
2 - The System connects using the saved IP and changes the semaphore to "ON"

Alternative cases
2a The System takes the input field IP
- The system detects there is a new IP in the input field and saves it
- The system connects using the saved IP
2b There is no IP
- The system will notify that there is a failure while trying to connect

Disconnect:
Description: Disconnect the controller from the screens
Actors: Authenticated user
Precondition: The authenticated user must be connected
Postcondition: The controller ends up disconnected from the screens
Main scenery:
1 - The user selects the option "Disconnect"
2 - The System disconnects and changes the semaphore to "OFF"

Save IP:

Description: The user saves an IP in his account

Actors: Authenticated user

Precondition: None

Postcondition: The IP will be saved in the user profile

Main scenery:

1 - The user enters an IP in the input field and selects the option save

2 - The system validates if the Ip is correct and saves it

Alternative cases:

2a - The IP is invalid

- The System does not validate the IP, notifies the user, and abort

Hide/Show logos:

Description: Hide/Show the logos on the screens

Actors: Authenticated user

Precondition: The authenticated user must be connected

Postcondition: The logos will be shown or hidden from the screen

Main scenery:

1 - The user selects the option "Hide/Show" logos

2 - The system sends the order to the screens and they perform the corresponding action

Reboot:

Description: Reboot the screens

Actors: Authenticated user

Precondition: The authenticated user must be connected

Postcondition: The screens reboot

Main scenery:

1 - The user selects the option "reboot"

2 - The system sends the order to the screens and the screens restart

Poweroff:

Description: Poweroff the screens

Actors: Authenticated user

Precondition: The authenticated user must be connected

Postcondition: The screens shut down

Main scenery:

1 - The user selects the option "power off"

2 - The system sends the order to the screens and the screens shutdown

Change game mode:
Description: The game mode changes
Actors: Authenticated user
Precondition: None
Postcondition: The game mode of the app changes
Main scenery:
1 - The user selects the option "mode"
2 - The system asks the user the select one of the available game modes (Satellite, Local Play, and if available Top play)
3 - The user selects any of the three
4 - The system changes the game mode and updates the chessboard and GUI accordingly to the game mode selected

Resume Top play:
Description: Resume/continue the current top play
Actors: Authenticated user
Precondition: There is an already started "top play" and it is on pause
Postcondition: Resume the "top play" from the point it was stopped
Main scenery:
1 - The user selects the option "   "
2 - The system resumes the "top play"

Pause Top play:
Description: Pause/stand by the current top play
Actors: Authenticated user
Precondition: There is an already started "top play" and it is playing
Postcondition: The "top play" stop reproducing and stays in a fixed/static state
Main scenery:
1 - The user selects the option " ⏸ "
2 - The system stops the "top play"

Forward Top play:
Description: Move the next available movement
Actors: Authenticated user
Precondition: There is an already started "top play"
Postcondition: Move forward
Main scenery:
1 - The user selects the option "⏭"
2 - The system updates the chessboard with the next move

Alternative cases:
2a - No more movements available
   -   The system ends the "top play" and establishes the satellite game mode
2b - Top play is not paused
   -   The system now behaves like the pause button

Backward Top play:
Description: Undo the last movement
Actors: Authenticated user
Precondition: There is an already started "top play"
Postcondition: Undo the last reproduced movement
Main scenery:
1 - The user selects the option "◄◄"
2 - The system updates the chessboard with the previous move

Alternative cases
2a - Top play is not paused
- The system now behaves like the pause button
2b - No previous movement
- The system does not respond

Decrease speed x.5:
Description: Decrease the speed of the play
Actors: Authenticated user
Precondition: There is an already started "top play"
Postcondition: set new speed (slowest) to the reproduction system
Main scenery:
1 - The user selects the option 'x.5'
2 - The system set the slowest speed available

Restore default speed x1:
Description: Reset the speed of the play
Actors: Authenticated user
Precondition: There is an already started "top play"
Postcondition: reset the default speed to the reproduction system
Main scenery:
1 - The user selects the option 'x1'
2 - The system reset the default speed to the reproduction

Increase speed x2:
Description: Increase the speed of the play
Actors: Authenticated user
Precondition: There is an already started "top play"
Postcondition: set the fastest speed to the reproduction system
Main scenery:
1 - The user selects the option 'x2'
2 - The system set the fastest speed available

Fetch Chessboard Status:
Description: fetch the package that contains the response of the satellite
Actors: Time
Precondition: It is the satellite's turn
Postcondition: The database will be updated, so are the attempts per user, and change the turn to the players
Main scenery:
1 - The time will fetch the SATBOARD.dat through SFTP and update the chessboard status, restore attempts per player and change the turn to the players

Alternative cases:
1a - There is no SATBOARD.dat
- The time will abort the operation

Send Chessboard status:
Description: create and send the package that contains the players move
Actors: Time
Precondition: It is the player's turn
Postcondition: The created package will be sent to the ground station, SFTP
Main scenery:
1 - The time will fetch the most voted movement
2 - The time will create the package and send it to the ground station

Alternative cases:
1a - There are no votes
- The time will use a chess engine and vote automatically instead of the players
- Goto step 2

4.3 Functional Requirements

This section describes the functionalities that should appear in the system.
We will use cases to represent the interaction between all the actors described above and the system.

● The system will allow an unauthenticated user to register/log in on the platform.
● The system will allow the authenticated user to log out.
● The system will allow the authenticated user to change the game mode.
● The system will allow the authenticated user to vote for a movement.
● The system will allow the authenticated user to play against the local machine.
● The system will allow the authenticated user to reset the local play.
● The system will allow the authenticated user to reproduce a master chess play.
● The system will allow the authenticated user to interact with the player (play/pause, forward, backward, x.5,x1,x2).
● The system will allow the authenticated user to get the current satellite location.
● The system will allow the authenticated user to navigate to the about section.
● The system will allow the authenticated user to use the LGSetting (every action of it).
● The system will allow the authenticated user to restore the chessboard.

4.4 Non-Functional Requirements

- **Usability**: The tool should provide a simple interface so that any user can use our platform without problems. For this, the design of the interfaces will be the simplest as possible, and we will offer a user manual to facilitate the use of the tool.

- **Performance**: We must ensure that our system is as efficient as possible, preventing response time and lengthy communications

- **Safety**: The tool must protect the data in the database. All information taken from users will not be processed for any other purpose.

- **Stability and availability**: The tool must be available at any time for users to access the platform whenever they want. Any user who does not have access to an "LG Cluster" will be able to use the application anyway.

- **Portability**: The controller must adapt to all types of devices (Responsive) regardless of its shape and type.

- **Escalability**: Although our tool may initially have a low number of users, the app must work also if the number of users increases considerably. Also, the code that we develop must be easy to maintain.

5. System Design

5.1 Logical Archiquecture

- **Controller**: We use NextJS which follows the client-server architecture. In the context of web applications, the client refers to the browser on a user's device that sends a request to a server for your application code. It then turns the response it receives from the server into an interface the user can interact with. Server refers to the computer in a data center that stores your application code, receives requests from a client, does some computation, and sends back an appropriate response.

  The controller uses web sockets (SocketIO) to connect to the server

  LG Space Chess also uses Firebase to store all the user, chessboard, votes, and IP data.

- **Cloud Functions**: Serverless architecture. Functions are the unit of scale in a serverless architecture that abstracts the language runtime. We are not talking about CPU or RAM or any other resources that we need for a function to run. We talk about the time it takes to execute the function. We write our functions, publish them to the cloud, and pay as go.

- **Screens**: Follow the client-server architecture, We use ExpressJS. Web sockets to communicate the screens with the server (SocketIO) and also to communicate with the controller/web.

  Screens use Firebase to store the URL, the URL is generated by creating a tunnel between the localhost and the internet using SSH

5.2 Physical Architecture

Controller: Will be hosted on a remote server. Use the HTTP/HTTPS protocols to communicate the client with the server, these will be running on different machines. Android users will need at least android 6 to use the controller.

Cloud Functions: In serverless architecture, some physical aspects are unknown depending on the hoster. Use the HTTP/HTTPS protocols to communicate.

Screens: Will be hosted in the Liquid Galaxy Cluster, Use the HTTP/HTTPS protocols and web technologies. The server and the client will be running on the same machine.

5.3 User Interface

The interface should be as simple as possible, so every user will be able to interact with it without any previous understanding

The main structure is:
- - Top: Header or Navigation Bar
- - Body: The targeted area where the user is

5.3.1 Login Page

In this endpoint, the user will be shown the name of the app as well as the logo and the login/register button.

5.3.2 Dashboard / Board Page

Here the user finds the main purpose of the app, this is where he interacts with the game and screens if connected, where he will be shown all the available options (Vote, Local Play, Top Play Reproducer, Screen View Controller).

5.3.3 FindSat

The user will be shown the current location (real-time updated) of the satellite. The user will be able to Untrack the satellite in order to move freely through the map.
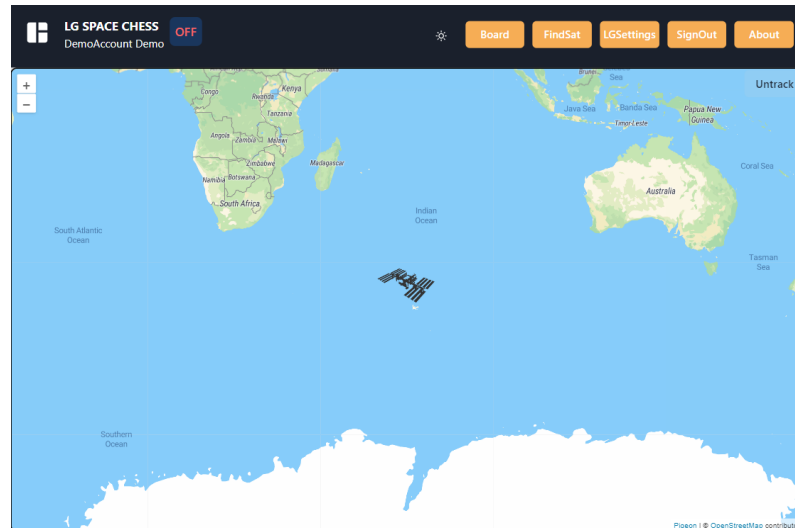
5.3.4 LGSettings

The user will be able to connect/disconnect to the rig, Hide/Show the screen logos if connected as well as Reboot or Poweroff the rig (must be connected).

5.3.5 About Page

There is info about the Author, Mentors, and Tester, also the logos will be shown as well as a brief description.
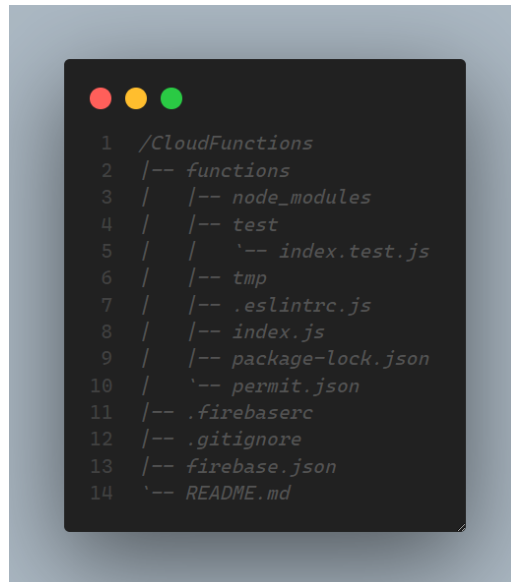

5.4 Data Design

In this section, we will define the scheme that we have designed to store the data of the users, as well as the board, votes, and tunnel URL. We have used Firestore (Firebase)
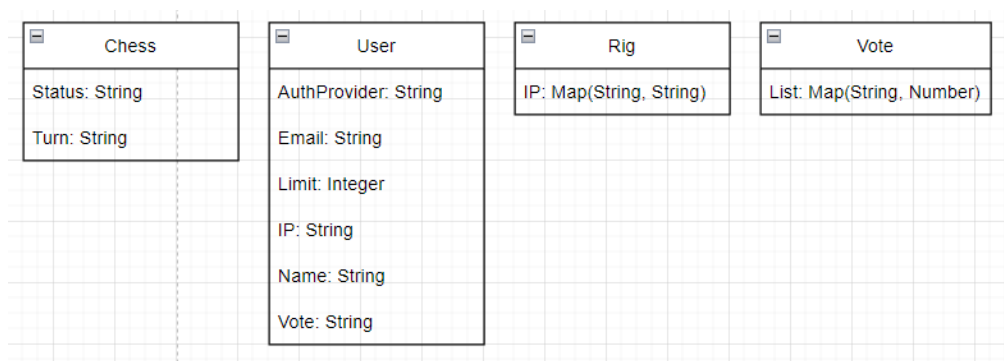
6 System Implementation

Explain the structure of the three subprojects

6.1 Cloud Function Structure

```
1   /CloudFunctions
2   |-- functions
3   |    |-- node_modules
4   |    |-- test
5   |    |    '-- index.test.js
6   |    |-- tmp
7   |    |-- .eslintrc.js
8   |    |-- index.js
9   |    |-- package-lock.json
10  |    '-- permit.json
11  |-- .firebaserc
12  |-- .gitignore
13  |-- firebase.json
14  '-- README.md
```

Description of the most important folder/files:
- *node_modules*: This folder stores all the modules that are being used, such as "js-chess-engine", "SFTP-Client"
- *test*: Contains the script to test all the functionalities
- *tmp*: Temporal folder, this is where we store temporary files to exchange with the ground station.
- *index.js*: Contain the code of the scheduled functions
- *permit.json*: Key to access the database

6.2 Screen Visualization Structure

```
 1  /Screen
 2  |-- Assest
 3  |   |-- Spl.png
 4  |   |-- controller.png
 5  |   `-- lgrigImage.png
 6  |
 7  |-- Public
 8  |   |-- bgImages
 9  |   |   |-- bground.png
10  |   |   `-- logos.png
11  |   |-- controller
12  |   |   |-- index.html
13  |   |   `-- main.js
14  |   |-- game
15  |   |   `-- demo.js
16  |   |-- models
17  |   |   |-- textures
18  |   |   |-- scene.bin
19  |   |   |-- scene.gltf
20  |   |   |-- Earth
21  |   |   |   |-- textures
22  |   |   |   |-- scene.bin
23  |   |   |   |-- scene.gltf
24  |   |   |-- sat
25  |   |   |   |-- textures
26  |   |   |   |-- scene.bin
27  |   |   |   |-- scene.gltf
```

```
28  |   |   `-- packet
29  |   |       |-- textures
30  |   |       |-- scene.bin
31  |   |       `-- scene.gltf
32  |   |
33  |   |-- GLTFLoader.js
34  |   |-- multiple.html
35  |   |-- multiple.js
36  |   |-- three.min.js
37  |   `-- tween.module.min.js
38  |
39  |-- .env.example
40  |-- Installation.md
41  |-- README.md
42  |-- index.js
43  |-- install.sh
44  |-- kill-chess.sh
45  |-- open-chess.sh
46  |-- package-lock.json
47  `-- package.json
```

Description of the most important folder/files:
- *Asset*: Contains images for the README.md doc.
- *Public*: Contains the client side such as the files to create the scene, 3d models, and background data. There is also a local test controller.
- .env.example: It is an example of the keys the user must have to use the LG Space Chess Screens.
- *Installation.md*: Document all the commands to install the project.
- *README.md*: Document with the explained installation and troubleshooting guide.
- *index.js*: Server implementation
- *install.sh*: Script to install the project
- *kill-chess.sh*: Script to stop the screen project
- *open-chess.sh*: Script to open the project on all the screens (LG Cluster)

6.3 PWA Structure

```
1   /Controller
2   |-- .well-known
3   |   '-- assetlinks.json
4   |-- components
5   |   |-- AboutBody.js
6   |   |-- DisplayChess.js
7   |   |-- DisplaySat.js
8   |   |-- Header.js
9   |   |-- Login.js
10  |   |-- Splash.js
11  |   '-- socketState.js
12  |-- pages
13  |   |-- api
14  |   |   '-- api.js
15  |   |-- 404.js
16  |   |-- _app.js
17  |   |-- _document.js
18  |   |-- _offline.js
19  |   |-- about.js
20  |   |-- findsat.js
21  |   '-- index.js

20  |-- public
21  |   |-- Spl.png
22  |   |-- Spl2.png
23  |   |-- Spl3.png
24  |   |-- favicon.ico
25  |   |-- icon-192x192.png
26  |   |-- icon-256x256.png
27  |   |-- icon-384x384.png
28  |   |-- icon-512x512.png
29  |   |-- logo.png
30  |   |-- logo2.png
31  |   |-- logos.png
32  |   |-- manifest.json
33  |   |-- noSpace.png
34  |   |-- satimage.png
35  |   '-- vercel.svg
36  |
37  |-- styles
38  |   |-- Home.module.css
39  |   '-- global.css
40  |-- utils
41  |   |-- requests.js
42  |   '-- theme.js
43  |

43  |
44  |-- .eslintrc.json
45  |-- .gitignore
46  |-- README.md
47  |-- firebase.js
48  |-- next.config.js
49  |-- package.json
50  '-- yarn.lock
```

Description of the most important folder/files:
- *.well-known*: Specially named folder that google play looks for.
- *assetinks.json*: File needed by google play to hide URL from the APK.
- *Components*: Contains all the components that the pages use.
- *Pages*: Endpoints of the controller project (build with components).
- *_offline.js*: This page shows when the user lost the internet connection
- *_document.js*: Configure web for PWA
- *404.js*: This page shows when trying to get an unknown endpoint
- *Public*: Public files such as images.
- *Public/manifest.json*: Important file for PWA conversion.
- *Styles*: Contains all the CSS styles files.
- *Utils*: Contains all the best play moves and theme configuration (dark mode config).
- *firebase.js*: Manage the connection between the server and the database.
- *next.config.js*: This contains the configuration of the NextJS framework.

CHAPTER III: Testing & Development

7. Test

This section will deal with all the tests carried out on our project. The target of these tests is to detect errors that may have occurred in the programming. The tests have been done by people outside of the programming part so that the tests are more reliable.

7.1 Unitary tests

Unit tests are those that check the functionalities of the different components of the system independently. The main target of these tests was to detect possible failures before communicating and building this machine-to-machine (M2M) system, So we can check if each module works well separately.
Mainly, these tests served to verify that the functionalities worked correctly.

Important checks were:
- Log in & protected routes (Controller).
- Real-time updates (Controller).
- SFTP Connection (Cloud function).
- Data division between screens; View offset (Screens).

Note: We have used mocha to test the majority of the functionalities

7.2 Integration tests

These tests verify that the union or assembly/connection of the components and projects analyzed in the unit tests are correct. We have put special attention to the controller's connection with the screens and the exchange of packages with the ground station.

7.3 System tests

System tests consist of testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements.
System testing takes, as its input, all of the integrated components that have passed integration testing.

7.4 Validation

In this phase, we have made sure that all the objectives have been achieved and that the final result is viable and works as it should so it could be uploaded into the Google Play Store and Huawei AppGallery.

CHAPTER IV: Installation & Troubleshooting

8. Installation

This section will detail all the steps to follow to install LG Space Chess Screens. We will also point out what are the necessary tools. This section will contain the installation manual for the screens since the controller does not need any configuration.

8.1 Requirements

1. Make sure the Liquid Galaxy Core is installed, Check out the git hub repository
2. Check your Node.js version, we recommend version 16, but the lowest version needed is 14. Use the following command to check your current version: `node -v`, the output should look something like `v16.*.*` (`v14.*.*`)
   a. Install Node: NodeJS (Windows), NodeJS (Linux)
3. Must have pm2 on the master machine
   a. Install pm2: `sudo npm i -g pm2`
4. Make sure Chromium Browser is installed on all machines

8.2 Installation guide

MASTER MACHINE ONLY:

1. Open a terminal and go to the default directory ('~'):
   a. `cd ~`
2. Clone the repository in the current directory:
   a. `git clone https://github.com/PabloSanchi/SpaceChessScreens`
3. Go to the SpaceChessScreens folder
   a. `cd SpaceChessScreens`
4. Execute the installation script (install.sh)
   a. `bash install.sh`
5. Reboot your machine if not done yet
6. Enter the following command in your machine (you must be in the SpaceChessScreen directory)
   a. `ssh -o TCPKeepAlive=yes -R 80:localhost:8120 nokey@localhost.run`
   b. Enter **yes** when asked '*Are you sure you want to continue connecting (yes/no)?*'
7. Ask the owners for the '**.env**' file, otherwise, the connection won't be available
8. Installation finished

8.3 Launch Space Chess Screens

MASTER MACHINE ONLY:

Note: Your current location must be '~/SpaceChessScreens'

1. Execute the launch script (open-chess.sh)
   a. `bash open-chess.sh NUMSCREEN`
   b. NUMSCREEN is the number of screens of your rig. The default value is 5. Make sure to set it properly according to your rig setup
2. Launching finished

CLOSE LG SPACE CHESS SCREENS
1. Close the opened terminal (Stop server)
2. Execute the kill script: **`bash kill-chess.sh`**

8.4 Troubleshooting

[1.0] Installation errors
[2.0] Launch errors
[3.0] Connecting issues

Solution

[1.0]
If something went wrong during the installation, the main cause is that you do not satisfy the requirements. Please make sure to check the requirement area.

[2.0]
If you are experiencing some errors while executing the **open-chess.sh** script, kill it, and restart it.

- Stop the server - Close all related running terminals
- Execute - `bash kill-chess.sh`
- `bash open-chess.sh NUMSCREEN`

Remember that your current location must be the ~/SpaceChessScreens folder

[3.0]
If you cannot connect using the rig modal on the controller, remember to ask the owners for the **.env** file.

Note: If none of the above fixes worked for you, send us an email.

CHAPTER V: User Manual

9. How to access

For the controller, you can use your web browser and enter the following URL:
https://lg-space-chess-pwa.vercel.app/
Or download the android application (APK):
https://play.google.com/store/apps/details?id=app.vercel.lg_space_chess_pwa.twa&hl=es

10 Interact with the environment

- Login:
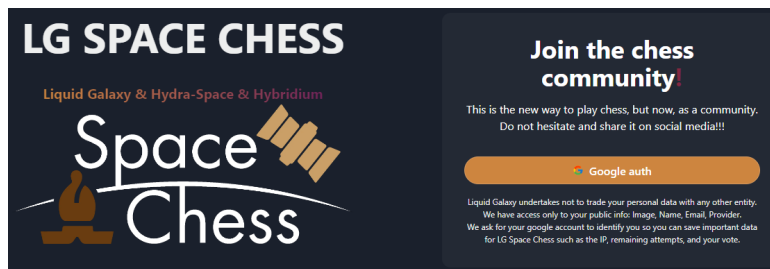
This is the first page that the unauthenticated user will see for the first time. Where he can see the name and logo of the project as well as the login/register button.

- Navigation Bar:

Once you logged in, you will find at the top of the app, a navigation bar or header.
Here you will be displayed some essential info such as your display name or connection status. You will also see the name of the app as well as the navigation buttons.

Buttons (from left to right):
- Toggle Dark mode: change the color theme (light or dark)
- Board: Show the dashboard (main page)
- FindSat: Display a map with the current satellite location (real-time updated)
- LGStetting: Display a modal with the connection properties
- SignOut: Signout from the application
- About: Display the about page

- Dashboard:

This is the main menu, the first thing the user will see once he logged in. The left buttons will be disabled so that the user won't use them.

If it is the user's turn he will be able to vote, by dragging any white piece to another square (the move must be legal).

If the user clicks the "votes" button, he will be displayed all the current votes by all users (if the user is connected, the votes will be displayed on the screens).

If the user clicks the "Top Plays" button, will appear a top play selector.
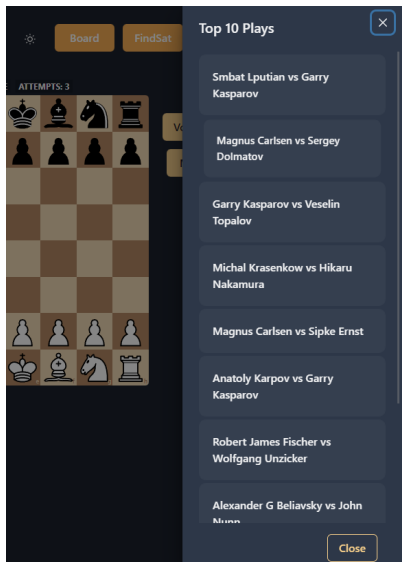
If the user selects any of the ten plays, he will be shown the "Top Play Reproducer", where he can pause it, play it again, go backward, go forward, change the speed and end the play.
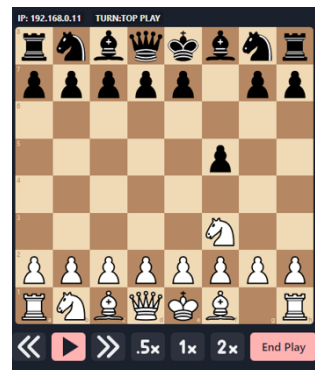
If the users decide to end the play, the satellite game mode will be set.
(Every state of the chessboard will be displayed on the screens if connected)

Back to the main photo of the dashboard.
If the user clicks the "Mode" button, he will be able to change it, to the local play or satellite. If he has already started a "top play" he will be able to continue it.
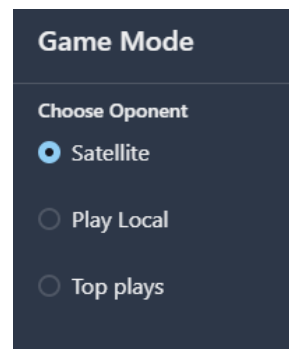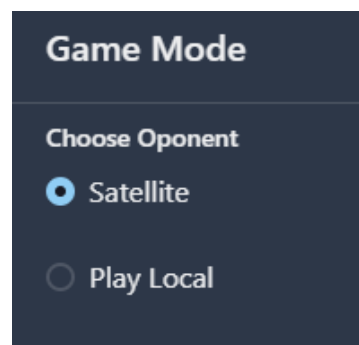
If the user selects "Play Local" he will be shown a new chessboard state (new or the last one if he already started a local play) and below the chessboard he will be displayed a difficulty selector (Easy, Medium, Hard).
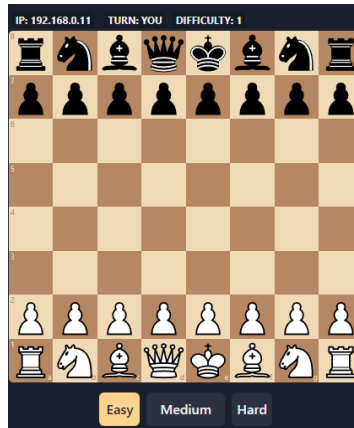
The "reset" button will be enabled and its function is to reset the local play status.

Once the user connects to the rig, the dashboard will look like the following:

Now the left buttons will be enabled, two joysticks will appear on the sides, two new buttons (Chess, Earth), and the RESTORE button.

These buttons will affect the view on the screens only.

♖ : Will set the perspective of the white pieces.

♜ : Will set the perspective of the black pieces.

Center: The button between the ♖ and ♜ , will restore the default position of the chessboard.
↻: Rotate the chessboard to the left.
↺: Rotate the chessboard to the right.
↑: reduces the inclination angle.
↓: increases the inclination angle.

Green Joystick: Move the scene horizontally.
Blue Joystick: Change the scene's depth.

Chess: Show the chess view, similar to the centering button.
Earth: Show a 3D illustration of how this game works in the background.

- **LGSettings:**

If the user hit "LGSettings" in the navigation bar, he will be shown a setting modal.

LGRig IP:
The default text is his current saved IP, if the user does not have any there will be an example "e.g. 192.168.0.15". If the user just wants to save an IP he must click the "save" button.
If the user wants to save & connect just hit "connect" directly without the need of saving first.

LG Logos:
If connected, the user will hide or show the logos on the cluster.
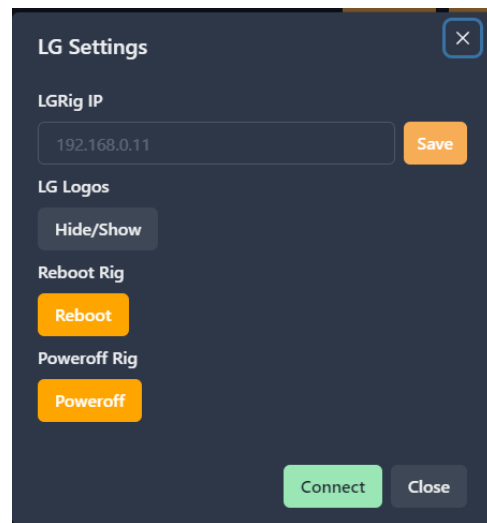
Reboot Rig:
If connected, the user will be able to reboot the rig (He will be asked for confirmation)
Poweroff Rig:
If connected, the user will be able to reboot the rig (He will be asked for confirmation)

Connect:
If the user already has an IP, LG Space Chess will try to connect to it if the user has entered an IP in the input field that IP will be saved and the app will try to connect to it.

Disconnect:
This button will be visible only if the user is connected, and its function is to disconnect from the screens.
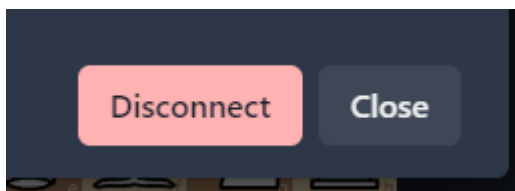
- **FindSat:**

   Here the user will visualize the current satellite location (real-time updated).

   On the top right corner, the user finds a blue button, that says "untrack" or "track" depending on if it has been already clicked.

Liquid Galaxy | Hydra Space | Hybridium

- About:

Here the user will find some info about the project, author, mentors, and tester/supporter, as well as the logos and a brief description.