

POLYSTAR

A SERIOUS GAME ON TACTILE TABLE

By Lepage Tim and Sergeant Dimitri



INTRODUCTION

Polystar rescue is a revision game to help students acquire and apply key vocabulary and grammatical notions from chapters 1-6 from the 3rd year POLYTECH course booklet. This serious game is designed to help students revise before the final test. Our goal in this project is to create a multiplayer game by adapting the existing board game on the café tactile tables TACTUALITIES.

INTRODUCTION	1
Specifications	3
Technical difficulties	3
The game rules	3
The database	4
Combining Meteor and basic CSS	4
Work repartition	5
First period: groping creation	5
Second period: database, game and presentations	6
Third period: Bug correction, design and deployment	6
Fourth period: Report and finishes	7
Content implemented	8
The FirstPage	8
The Help page	9
The Authentication page	10
The Modify page	11
The Game page	14
Impressions	16
Conclusion	17

Specifications

In order to carry out this project we had to respect the specifications given by the professors. Ms A.L Finkel is the creator of the original board game, so she will be associated with the client in the following report. Here are the requirements given by the professors:

- Use of *Meteor* a free and open-source isomorphic JavaScript web framework used for rapid prototyping and production of cross-platform code. This defined the languages used in our application: JavaScript, HTML, CSS.
- The respect of the rules of the original game, available [here](#)
- The game has to be playable on the three tables at the same time.

These were the only constraints we had to respect. Otherwise we were pretty free to implement what we wanted to solve the technical difficulties we faced.

Technical difficulties

1. The game rules

In order to understand the choices of implementation we made, we have to understand the way the game is played. Following meetings with Ms A.L Finkel the game rules were well defined. Polystar is a multiplayer game (usually 6 players split into 3 teams) played with several cards subdivided in six categories plus a seventh containing special cards (chance cards). The players roll dices which give them the cards to take and read to the other teams. If the answering team answer is correct they win points and a token from the category of question they just answered. In order to win the game, a team has to collect at least 20 points and the six tokens of the six question categories.

The first technical difficulty we encountered is more of an ergonomic issue: the tactile tables are quite small and gathering 6 students around them would force them to be very close to one another. We talked about it to Ms Finkel and she agreed with us, thus allowing us to reduce the number of teams to two (4 players).

2. The database

As specified above we need to store a high number of cards (there are approximately 8 units each containing around 50 cards). We had to choose a way to store all those cards in a

database. As Meteor comes with MongoDB, a free and open-source cross-platform document-oriented database program and the possibility to access the database from the client side, the way to technical part was quickly chosen. We used MongoDB collections to store the different cards collection we needed. The application holds two database, one for the classic game cards and one for the chance cards. Each card is identified by the unit in which they belong, the type of question (I know, I bet, I guess, I see, I say), the question itself, the answer and the number of points they make earn.

```
CardsList = new Mongo.Collection('unitscollec');
```

The cards of type I know are a little bit particular since they do not belong to a specific unit and must be accessible for every units. We chose to put them in the unit 0 which doesn't really exists and is only used for the cards I know.

The operations carried on the database could only be done by an authenticated user: the English teacher. We had to make sure that this user is the only one able to affect the database in order to secure it.

3. Combining Meteor and basic CSS

Use CSS with Meteor was not an easy task. Because it was our first Web application we created, we did not know a lot about HTML+CSS. We had to documentate a lot and even do tutorials. Unfortunately, most of the CSS tutorials were about creating a website which is quite easy, but none of them were combined with Meteor technology. For example, the background-image is normally really easy to put, but with Meteor it's a complete different way to put it. So even if it was easy, we had to look for the solution during few hours because few people use Meteor.

Then we thought we could use another CSS technology to help us. We first tried Less, a library given by Meteor, but because of a lack of information, we moved on to something else. It appeared that Bootstrap was functional and allowed us to place the game as we wanted. It allowed us to place our components as we wanted and the application design became easier with it.

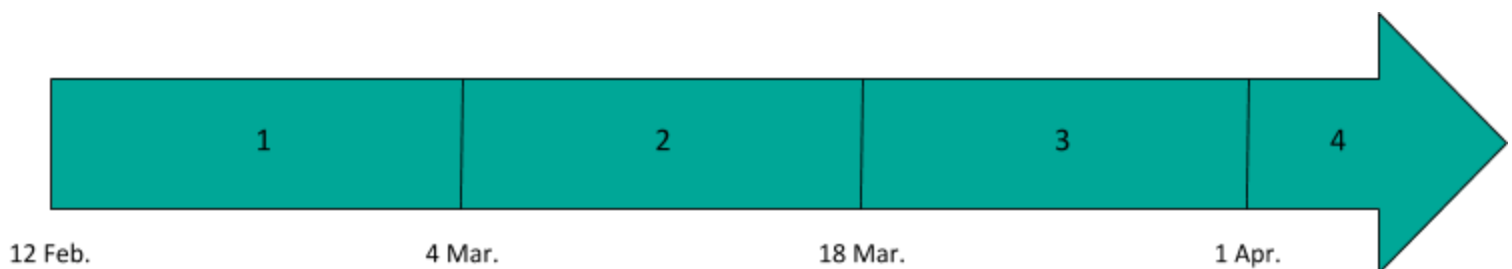
Work repartition

This project was carried out using agile methods such as the SCRUM. At every moment we were aware of what the other teammate was doing. We planned the project estimating the

approximate duration of each tasks and continuously informed on the current tasks advancement on a collaborative document.

Because we were only two in the group, it was easy to realise continuous integration. For each task realized, we merged our code and kept a version with the less bug possible.

All the documentation and diagrams beside the project where updated by both of us throughout the semester.



1. First period: groping creation

We started by gathering all the information we could about Meteor, JavaScript and HTML since it is the first web project we did. During this whole period we worked together. We started by creating the architecture of the project (the web pages). To navigate through the different pages we use iron:router which allows us to render the HTML template we want. We also started to implement the whole CSS background. (Image in background, styles of the buttons and the text etc.). The CSS was not completely working with Meteor, as said before, so we had to look for solutions during many weeks.

Once the pages were linked we started developing on a primordial point of the project: the authentication. Luckily meteor comes with a user account system allowing us to keep track of who is connected and thus preventing anyone from illegally interact with the future database. We also implemented a method to send a verification mail and the modifying page is only accessible for a verified user. The sending of the mail was completed at the end of the project because when the application is only available on localhost this method is unusable.

2. Second period: database, game and presentations

Now that the user can create an account and access to the page allowing him to modify the MongoDB database we needed to provide all the functions he needs as well as an intuitive interface for a tactile table. We created the methods allowing the user to create, delete or update a card as well as a way to display the cards and navigate through them.

Once the database was created and the cards usable, we started creating the game loop and mechanics. At this point there was very few graphic content and the second part of this period was more about presenting the advancement of the project. Before presenting the game, Tim finalized the game by adding the basic points system and winning conditions. We then had to prepare for the half-semester presentation and the demonstration during the event : 16 years of RICM. To do so we created a poster to represent our project in few lines.

At this point, the game was almost operational but with a poor design and few bugs to correct.

3. Third period: Bug correction, design and deployment

At this time the game was already playable, the interface was awful and some major bugs were remaining and we had to start a testing phase. To do so we first had to finish implementing the graphical part to facilitate the read of the game and to better locate the remaining bugs. We then spent a lot of time documenting about CSS to try to have an interface that really made think of a game with some animations for example.

Then came the time to upload our code on the tables to verify if everything was working as expected. Unfortunately, the tables being very rustic, we had to fully update them but still some of the problems we had are remaining to tis day. For example we did not manage to connect the tables with GitLab forcing us to upload the code we wrote using an USB key. At this point we were still running the application in the localhost and we needed a way to link the tables. Mr Donsez recommended us to use a switch and cables to link the tables since there was no server available for us, however since we were developing a website we found it too bad to not really deploy it. We asked our classmates owning a private server if they were willing to host our website and Aurélien Surier accepted and even took care of the deployment using meteor build.

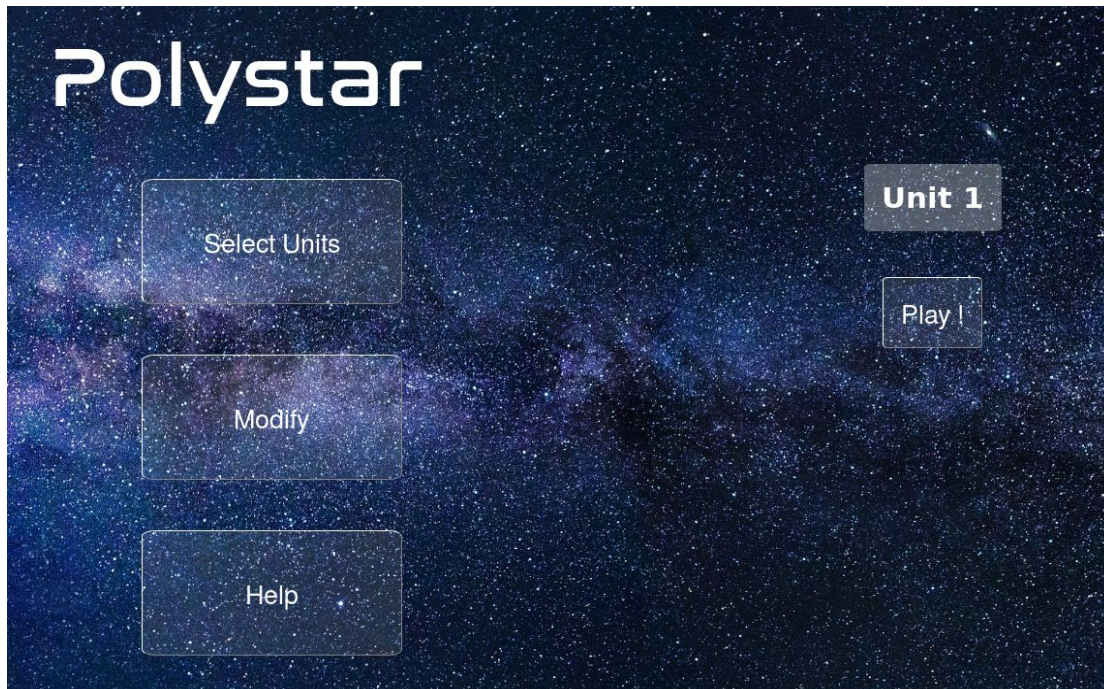
4. Fourth period: Report and finishes

Here is the last part of the semester, the application is working and deployed on asgdev.fr:3000. We finished the report that we started at the very end of the third period. We checked the diagrams we made and also cleaned a little bit the project because we already did it through the project.

Content implemented

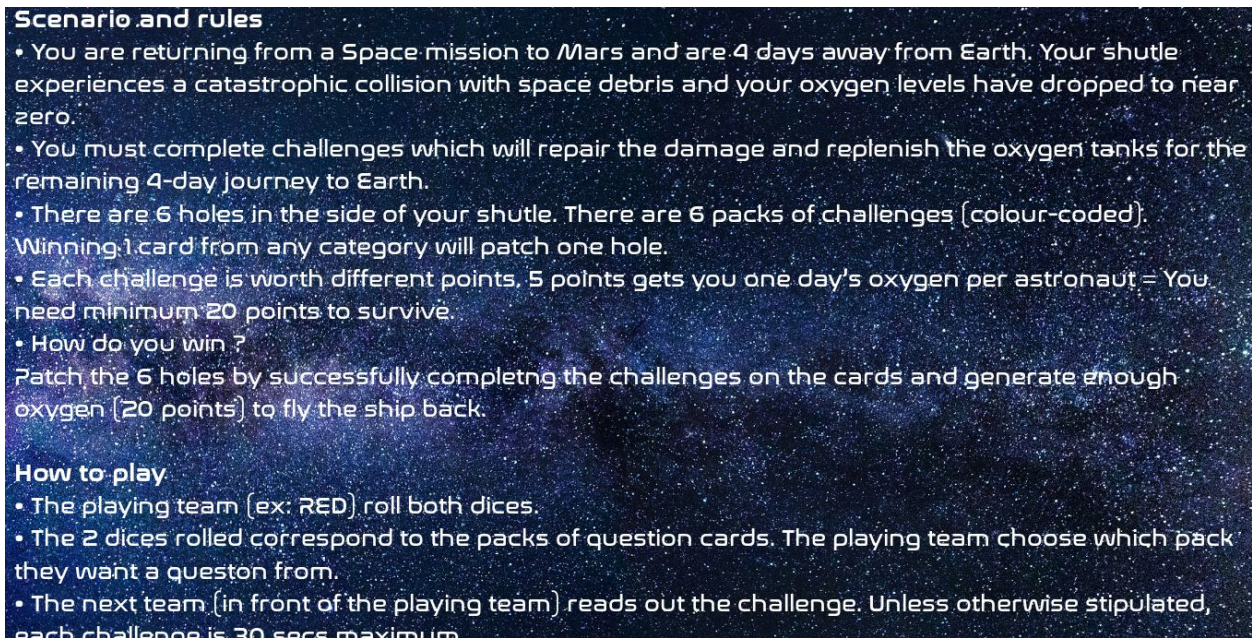
This project being a website we implemented 5 separated web pages.

1. The FirstPage



This page is the home page of our application. It lets the players choose the units to play with or see the help page. There is a link to the Authentication page too but this page is only intended for the teacher to modify the cards.

2. The Help page



Scenario and rules

- You are returning from a Space mission to Mars and are 4 days away from Earth. Your shuttle experiences a catastrophic collision with space debris and your oxygen levels have dropped to near zero.
- You must complete challenges which will repair the damage and replenish the oxygen tanks for the remaining 4-day journey to Earth.
- There are 6 holes in the side of your shuttle. There are 6 packs of challenges (colour-coded). Winning 1 card from any category will patch one hole.
- Each challenge is worth different points. 5 points gets you one day's oxygen per astronaut = You need minimum 20 points to survive.
- How do you win ? Patch the 6 holes by successfully completing the challenges on the cards and generate enough oxygen (20 points) to fly the ship back.

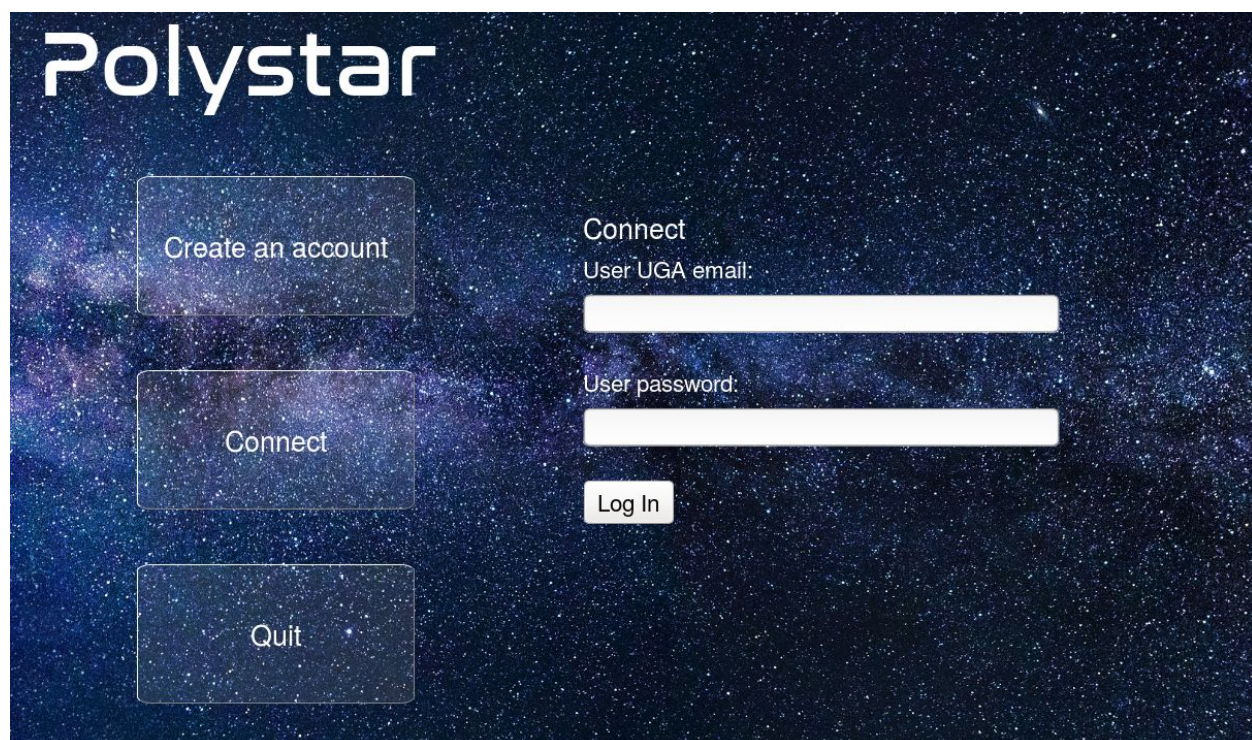
How to play

- The playing team (ex: RED) roll both dices.
- The 2 dices rolled correspond to the packs of question cards. The playing team choose which pack they want a question from.
- The next team (in front of the playing team) reads out the challenge. Unless otherwise stipulated, each challenge is 30 secs maximum.

Help page

This page is the simplest one of the five and its goal is to remind the game rules and scenario to the players.

3. The Authentication page



Polystar

Create an account

Connect

Quit

Connect

User UGA email:

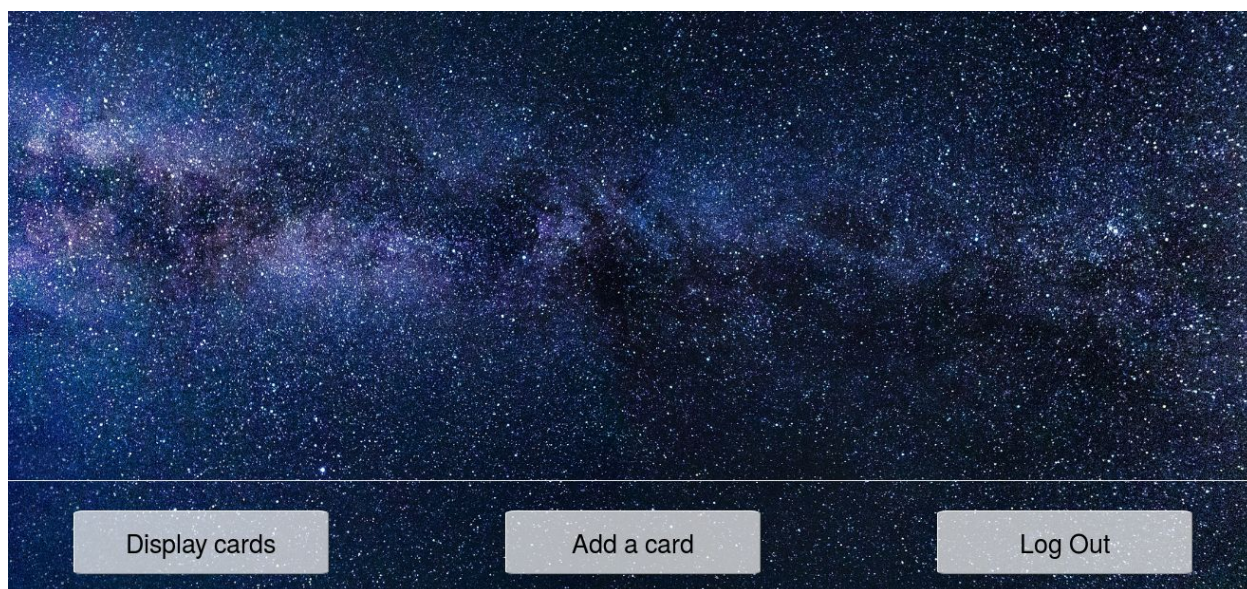
User password:

Log In

FirstPage

On this page the user (which is supposed to be the teacher) can create an account or log in. In order to create an account the user has to use an UGA teacher email address (ending with @univ-grenoble-alpes.fr). A confirmation email is then sent to validate his account and allow him to modify the cards. An account creation cannot be carried out if the user enters another email than an official UGA email address.

4. The Modify page



It is on this page that the teacher logged in with a verified account can modify the game content. This page is composed of two screens: on the first one the user has the possibility to add a new card. On the second one he can display the existing cards by navigating through them or selecting the unit and type to display, modify an existing card and delete a card.

Modify page: display cards

Modify page: add a card

To add a card you don't need to call a Server side method, this can be done in the client side thanks to this function we wrote:

```

export function createCard(unit, type, question, answer, points) {
  if (!UnitsAvailable.has(unit)) { //if the unit of the new card is not already existing
    UnitsAvailable.set(unit, 1); //We add the new unit to the list of unit containing
    at least one card
  } else {
    var old = UnitsAvailable.get(unit);
    UnitsAvailable.set(unit, old + 1); //If the unit already existed we increase the
    variable representing how many cards are in this unit
  }
  CardsList.insert({
    unit: unit,
    type: type,
    question: question,
    answer: answer,
    points: points
  });
}

```

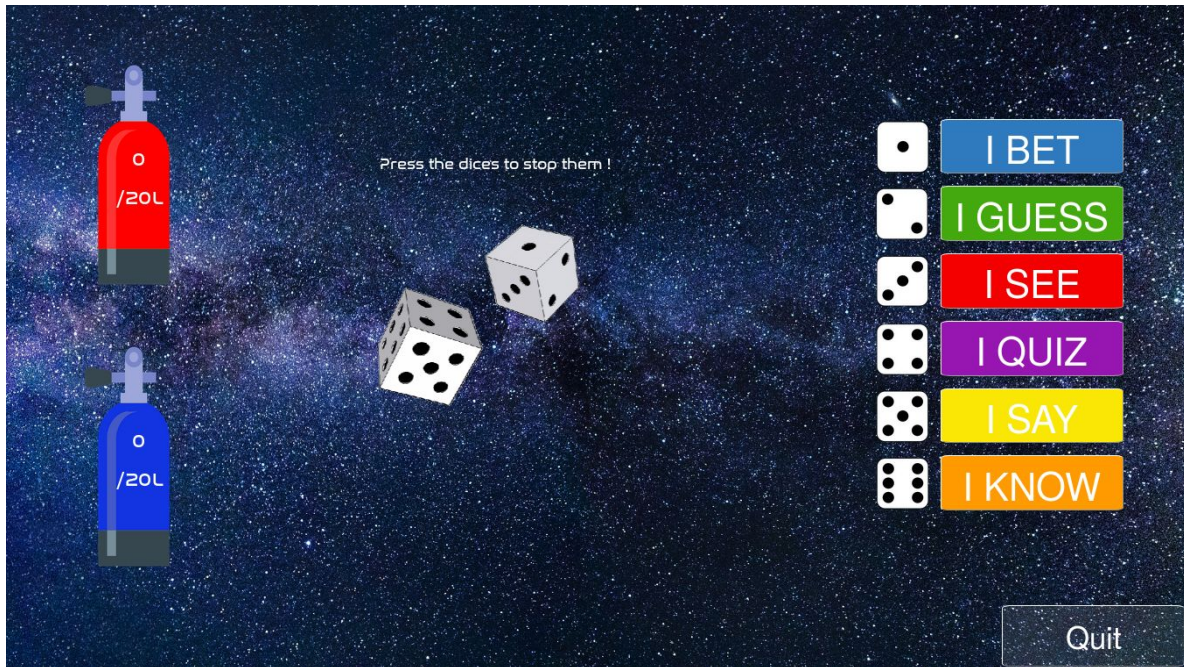
However to delete or modify a card the code cannot be placed on the Client side. That is why we wrote those two functions on the Server side:

```

deleteCard: function(e) { //e the card to remove
  CardsList.remove(e);
},
updateCard: function(e, unit, type, question, answer, points){ //e the card to update
  CardsList.update(e, {unit:unit, type:type, question:question, answer: answer,
  points:points});
},

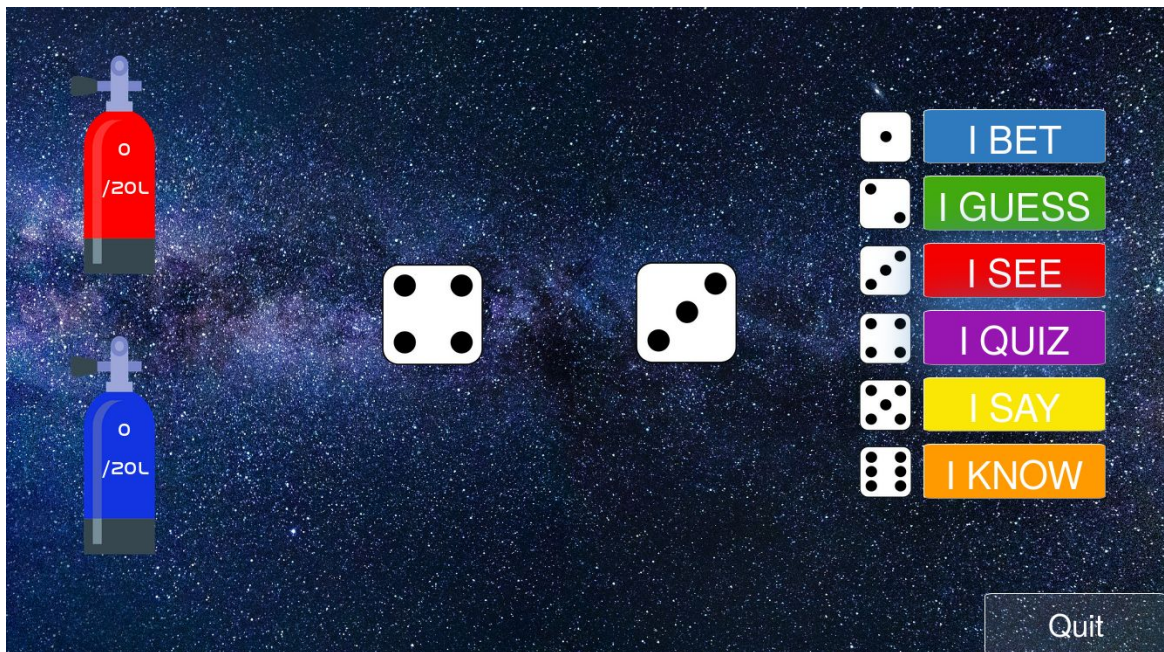
```


5. The Game page



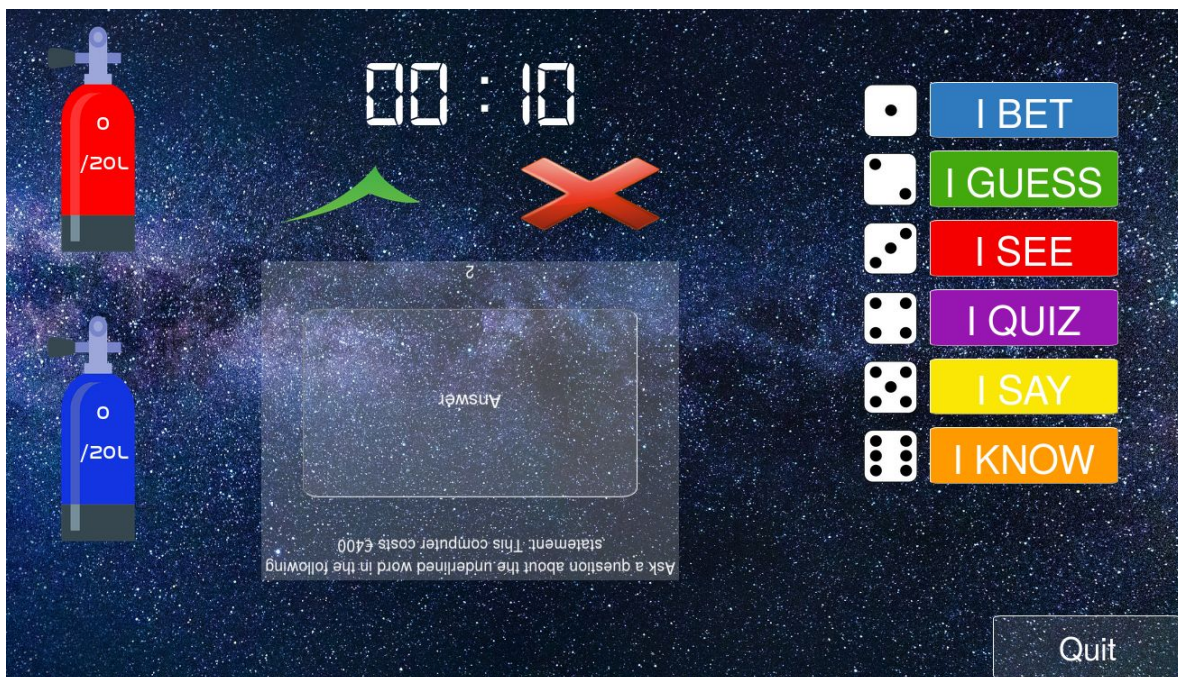
Game page: rolling the dices

This is the main page of our application: the game page. This is on this web page that all the game is going to be played. The designed is strongly inspired from the board game of A.L Finkel.



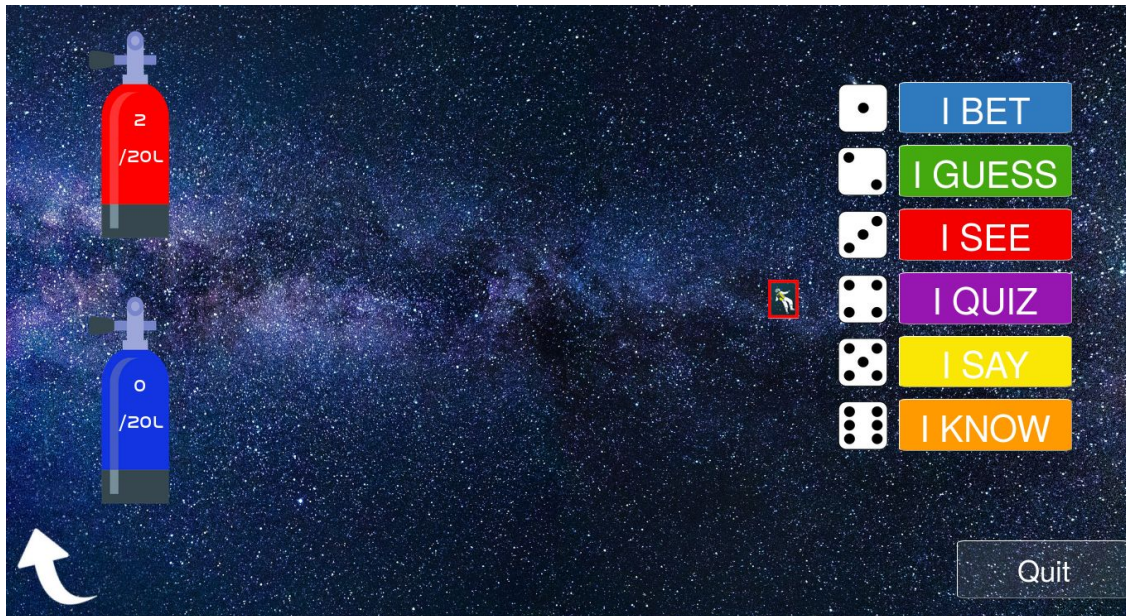
Game page: select the category

As you can see on the picture above, the selectable categories are highlighted thanks to a flashing css animation.



Game page: display of the question

The player select the category he wants and the other team reads out loud the displayed question and a timer starts. The non playing team then validates or invalidates the answer and the display updates itself.



Game page: update of the scores and end of turn

Impressions

This project has been presented twice to the public: once at the Polytech Grenoble open house day (3rd of March) and the RICM 16th anniversary (16th of March). During those two days we had the opportunity to show the project to a different type of public as young students and their parents as well as engineers came to us to see the game. The presence of the tactile tables brought us quite a lot of people since it is eye-catching and seems interesting and original. Unfortunately the project was not developed enough to be deployed on the tables and we had to show the advancement on the computers. However the people who we talked to seemed to be very interested in the game and surprisingly enough some of the older students or engineers were not able to answer some of the English questions of the game. This shows that this project has a real potential and could be used by anyone, not only the 3rd year students of Polytech Grenoble. We eventually showed the game to the client (Ms Finkel) and she was satisfied with it.

Conclusion

To conclude, we would like to say that it was interesting to do a useful project from scratch. It gave us many tools in Web programming that we didn't have at all. In terms of time, the semester we had to do this project was sufficient. It allowed us to discover the technologies that were new to us and to finish the project in time even if we had to work on our free time. Because it was interesting for us, it was not a problem to spend the week working on it. Another point for us that was really interesting is that we started to practice the techniques we learned in Génie Logiciel. Because we were only two in the group, it was easy to make continuous integration and to use Agile methods, but it still made an application of the theory we studied in lessons and that is important for our future.