CS25320 Coursework 2021: Canvas Game with Persistent Data

Julia Drozdz [jud28@aber.ac.uk]

Overview

BitHigher is a 2D simple platformer game inspired by Icy Tower. The aim of the game is to achieve the highest score possible. Score is obtained through surviving the game for as long as possible. What sets it apart from Icy Tower is the art and jumping mechanics. Further development of this game could include: collectibles, different types of platforms, etc.

Design

In this project there are several web pages: index, customize, game, scoreboard, about, readme and help. Readme contains information on how to set up the project and its contents. Scoreboard.html fetches all the users from the database and maps the data to the table using Javascript. I have decided to display the username, score and difficulty level they have used most recently. The help page provides some tips and tricks on how to play the

game which may not be apparent from the very beginning. The About page describes the game briefly and gives credits to all the resources for the game. In the main page (index.html) the user can choose their name, creating a new user in the database or using an existing account. This name is stored in the LocalStorage of the client's browser as it provides easy access to the correct user in the database for other pages. After logging in, the user is redirected to the customization page where they can choose the difficulty and the character they want to play as. The initial data for the form comes from the database, fetched by the name stored in the LocalStorage. After submitting the customization form the database is updated and the user is redirected to the game.

For this project I wanted to have a clear separation between the client and server. To communicate with the backend I am using the FetchAPI with async/await for handling asynchronous promises.

Project Structure

- **API directory**
  This directory contains all the endpoints for my database. When using the FetchAPI I am making distinct calls to each file for each action such as: fetching all users from the database, fetching a single user or updating a user. It also contains a file called queries which contains all the separate queries to the database for each endpoint.
- **Images**
  Contains all images for the websites. Images for the games are stored separately.
- **Src**
  Contains all the script files for all html files. It also has a subdirectory called game that holds all the game scripts and the game assets: images, sound.
- **Styles**
  Contains all the CSS files with one global CSS file which I am importing to each separate file. Each site has its own CSS file. This is a personal choice which makes it easier for me to work with a lot of CSS code.
- **Root**
  All the HTML files are stored in the root. I decided not to have them separately because the project is small enough for this way of storing to be relatively maintainable.

Data design

I decided to have two forms in this project. First is located on the main page of the game and the second is in customization.html. The model for the user consists of a name, difficulty, score, and hero. The data for the user that is logged in is fetched by the name stored in the LocalStorage. I decided not to use PHP sessions as that violates the separation of client and server structure I wanted to keep. For the customization form I am using form data sending it through a POST request to the updateUser.php endpoint in the api directory. Using the FetchAPI turned out to be quite tricky with PHP as the $_POST and $_GET variables were not

reading the payload (even though it was clearly there in the network tab). The solution to this problem was to read the raw body of the request using a PHP wrapper: `file_get_contents('php://input')`.

Username form

This simple form is used to set the username for the player. If the user exists in the database, it will use the existing record with its data and create a new user account otherwise. I have implemented some simple client-side data sanitization using regular expressions for this form. If the user entered a wrong value for the username, they will get the appropriate error message. There are three main cases I have covered: username contains any of the following characters: `<>'`"/?%;()&+-,` username is empty, or it exceeds 20 characters. After a successful logging it is possible to logout from the menu in the left-top corner of the web page.

User preferences form

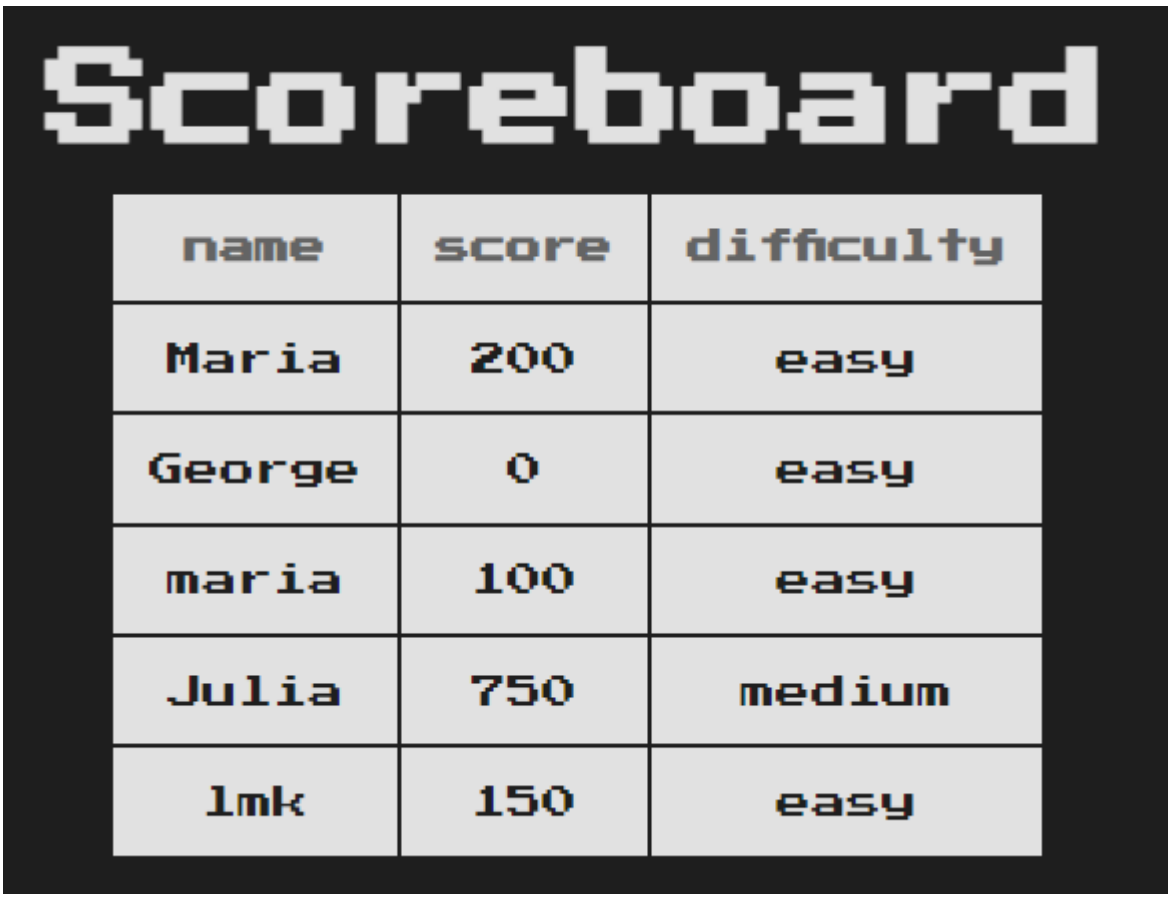
The second form in this project is the user's preferences form.

This form uses radio buttons with some advanced CSS styling to make it look more appealing. I have implemented this styling using the resources listed in the References section. The user can change the difficulty of the game and a hero sprite they want to use.

Updating the data

The data in the database is updated on two occasions: submitting the preferences form in customize.html and pressing G to play again. Submitting the preferences updates the difficulty and the hero for the username stored in the localStorage. Pressing G updates the score of the user (if it is a new highest score).

Leaderboard data in the database and the leaderboard as it appears in the game




```
 name   | score | difficulty | hero
--------+-------+------------+-------
 Maria  |   200 | easy       | goose
 George |     0 | easy       | goose
 maria  |   100 | easy       | goose
 Julia  |   750 | medium     | goose
 lmk    |   150 | easy       | goose
(5 rows)
```

After updating a user with the preferences form on customization.html (Julia)

```
 name   | score | difficulty | hero
--------+-------+------------+-------
 Maria  |   200 | easy       | goose
 George |     0 | easy       | goose
 maria  |   100 | easy       | goose
 lmk    |   150 | easy       | goose
 Julia  |   750 | easy       | cat
```

After establishing a new high score (and refreshing the page with G) - updating the score for the username stored in LocalStorage

name	score	difficulty	hero
George	0	easy	goose
maria	100	easy	goose
lmk	150	easy	goose
Julia	750	easy	cat
Maria	750	hard	chicken

(playing as Maria)

Reflection on the technical quality of the game

Use of ES6/7 classes and modules

When developing the game, one thing I immediately noticed is how unmaintainable it was. I decided to do some research on how I can improve this and I have come across the MVC architecture. I have not copied any code for this but the idea and the general structure can be seen in the youtube tutorial by Poth on Programming in the references.

I have tried to keep the code for the game highly modular, which improved maintainability significantly. The main.js file is the main file for the game, which provides a way of communication between all classes in the game. It contains global listeners for controlling buttons in the game.html file (muting sound, playing the game) the update, draw and game methods, as well as instances of the following classes: AssetsMaganer, Display, Controller and Game objects. The main idea behind this is that the classes do not know anything about other classes unless they absolutely need to, making it highly modular and maintainable.

Physics

One thing that I think could be improved a lot is the physics for this game, mainly jumping mechanics and collision detection. For the jumping mechanics it feels like the gravity is too strong, making the jump too abrupt. When it comes to collision detection I have decided to implement a collision detection sometimes referred to as “Tunneling”. This is due to the fact that when a player collides with a platform from the top, they are supposed to stop on top of it. In order to implement this behaviour the collision detection needs to “know” from which direction the player hit the platform. This cannot be achieved with a simple overlap detection since after triggering the overlap, there is no way of knowing the direction. This is why, with the help of a youtube tutorial from Poth on Programming’s channel, I have implemented tunneling collision detection. However, with this solution there might be a situation when a platform hits the player from the top while they are standing on the ground. In that particular instance the collision detection starts to fail and the player will fall through the ground. This happens due to the nature of the tunneling. Since we are checking the new values and then updating the old values, if many collisions happen on the screen they are not actually the same for the collision detection. A thorough explanation of this problem can be seen in the youtube video listed in the References section.

References

MVC

https://www.javacodegeeks.com/2012/02/building-games-using-mvc-pattern.html

Create a Platformer Game with JavaScript - Full Tutorial

PHP backend

For the backend part of this project I have used a number of resources to learn the concepts:
Brad Traversy:

PHP REST API From Scratch [1] - Database & Read

PHP PDO manual:
https://www.php.net/manual/en/book.pdo.php
PHP tutorial: https://www.phptutorial.net/php-pdo/

PHP wrapper

https://www.php.net/manual/en/wrappers.php.php

Redirecting web pages

https://stackoverflow.com/questions/17502071/transfer-data-from-one-html-file-to-another

Sounds

https://soundsbydane.itch.io/8-bit-sound-pack

The 8-bit-sound-pack is a copyrighted sound pack free to use. Credit to SoundsByDane.

Music

http://teknoaxe.com/Link_Code_3.php?q=1553

This work is licensed under the Creative Commons Attribution 4.0 International License

PHP wrapper

How to retrieve Request Payload
https://www.php.net/manual/en/wrappers.php.php

The customization form

For the customization form I have used the following resources:
https://stackoverflow.com/a/30663705/14638818
https://stackoverflow.com/a/17541916/14638818
https://markheath.net/post/customize-radio-button-css

Game loop

http://gameprogrammingpatterns.com/game-loop.html

Animation (Sprite class)

Pizza RPG Part 5 - Character Animations #pizzalegends

Mapping dynamic data to a table with JavaScript

https://stackoverflow.com/questions/29454620/map-a-javascript-array-to-html-table

Tunneling (2d Collision Detection)

https://www.youtube.com/watch?v=VpSWuywFIC8