Final Group Project

Title

Space Invaders

Course

CIS/CSC-17B

Section

40502

Date

June 8th, 2020

Authors

Brandon Sanchez / Matthew Borja

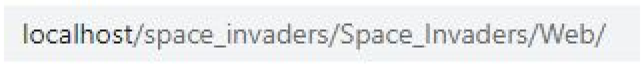
Christopher Alexman / Martin Perina

Introduction

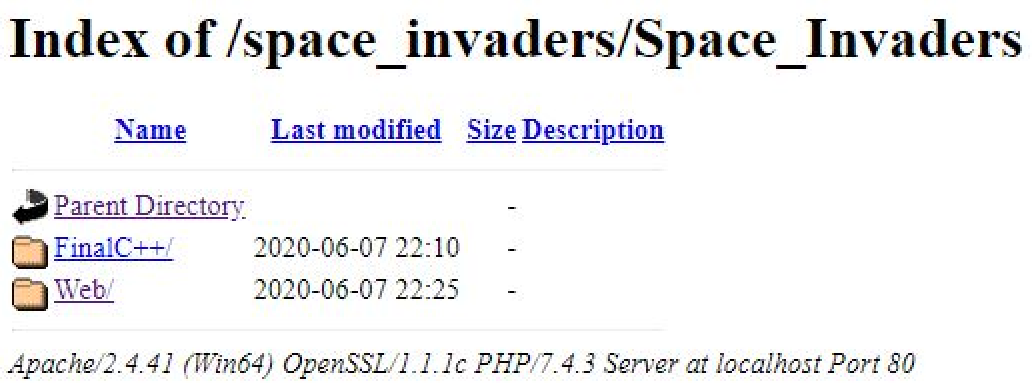
Space Invaders was an extremely popular arcade game when it was released in 1978. The game was originally created by the Japanese video game developer Tomohiro Nishikado. Even though it is an old game, many different ports have been made to play on modern PC's. The player controls a spaceship at the bottom of the screen that can be moved left and right. To win, the player must kill all of the enemies on the screen with their laser. A player loses if they get hit by enemy lasers three times during the playthrough. We decided to do a modern take on this classic game through the internet browser.

How to Run It

We mainly used the text editor VSCode to code this project and interact with changes on GitHub. In order to run it you need to put the entire folder in the htdocs in xampp and navigate to the Web directory within. There is an index.php file so it will start automatically when there.



```
localhost/space_invaders/Space_Invaders/Web/
```

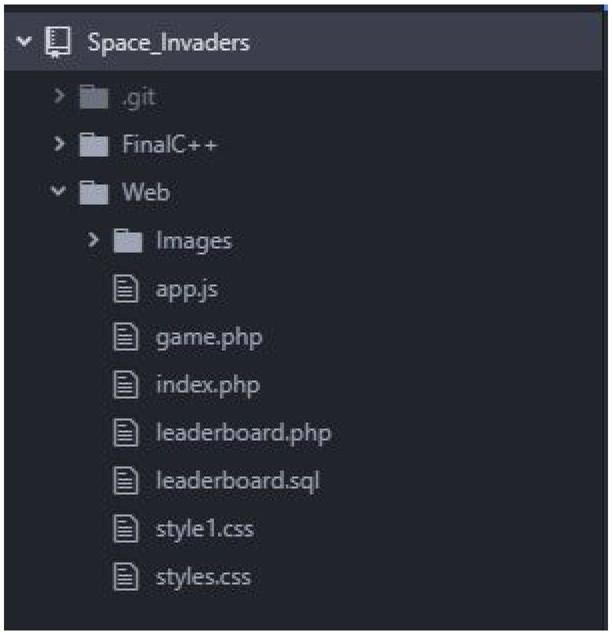


```
Index of /space_invaders/Space_Invaders

      Name          Last modified     Size Description

   Parent Directory                    -
   FinalC++/        2020-06-07 22:10   -
   Web/             2020-06-07 22:25   -

Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.4.3 Server at localhost Port 80
```

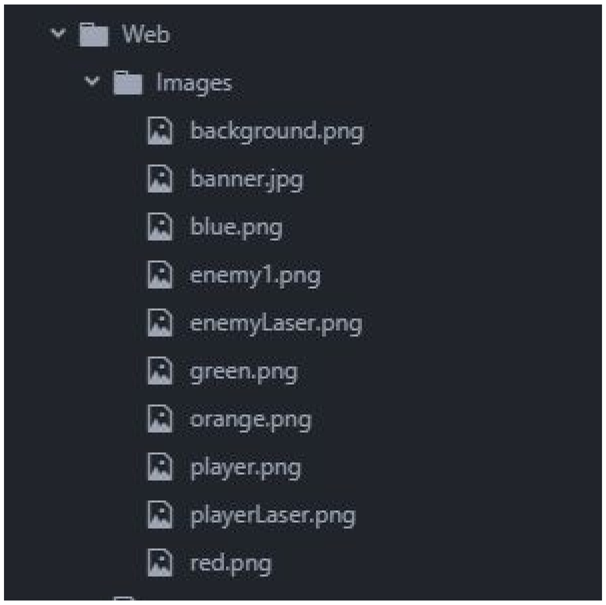
Development Summary

Lines of Code: 946

Here is an overview of what the project looks like in a folder:



```
Space_Invaders
  .git
  FinalC++
  Web
    Images
    app.js
    game.php
    index.php
    leaderboard.php
    leaderboard.sql
    style1.css
    styles.css
```



```
Web
  Images
    background.png
    banner.jpg
    blue.png
    enemy1.png
    enemyLaser.png
    green.png
    orange.png
    player.png
    playerLaser.png
    red.png
```

Chris

What you worked on:

For this project I worked on a few different components. My main part was the leaderboard and database side. I created the tables and query connection for the project and put it on the remote server at RCC. I tested it local first before putting it there. I also created a few forms related to the leaderboard where buttons on the win/loss screen navigated the user to the leaderboard page. I also added in all of the necessary sessions to pass information between .php files for the user input like their name, ship color choice, and high score. I had a hand in the debugging process as well as problems arose in development.

Compare C++ to JS/PHP:

This project of recreating Space Invaders with code is definitely easier done with JS/PHP than with C++. Visually there is a lot more to do with formatting and aesthetic but the capability with forms, cookies, sessions, etc makes it an easier time in my opinion.

Favorite part of project:

I had a ton of fun with this project. My favorite part was being able to interact with a remote database and get a glimpse of what is going on in the background of the internet and how it's all connected.

What you learned:

I learned a ton in this project. We mainly used VSCode as our text editor to code in. I learned a lot more about pushing/pulling/merging with GitHub with that. I also learned a lot about HTML, PHP, CSS, JavaScript, MySQL and Git. I was surprised looking back just how much I learned.

Brandon

What you worked on: I created the majority of the actual game part of the program that was written in Javascript. I implemented the animations using CSS transformations and windows request animation frame function. For Space Invaders, Javascript was much better to deal with since the animations were smooth and we could place images on the screen. My favorite part of the project was implementing the lasers and enemy movement. The enemy movement uses circular movement from the sine and cosine functions. I learned a lot about git merge since it was my first time working on a major project with others. There were a lot of challenges and one time I deleted about 3 hours of my work debugging the lives system.

Martin

-What you worked on:

I worked on the style and format of each page.

-Compare C++ to JS/PHP:

Javascript and PHP are much easier to program with than C++, especially when interacting with a web page. Moreover, programming with HTML and CSS are much easier to use than C++ when making an organized and good looking user interface(mainly because that's what HTML and CSS are for).

-Favorite part of project:

I really enjoyed being part of a web development project. It was nice to program a functional application that wasn't contained to a command line.

-What you learned:

I learned a lot about front-end and back-end web development.

Matthew

What you worked on:

In this stage of development on the project, I began with translating my segments of the C++ code over into Javascript, mainly the score counter and lives. As my work dominated my schedule, I played a supporting role in a few different aspects of the code thereafter. I contributed to the creation of the form element of Index.php and the customization of the player's ships along with finding the sprites to give variety. Lastly, I created a very basic cookie for the project.

Compare C++ to JS/PHP:

C++ holds a special place in my heart, as it is a language I'm more comfortable using and understanding, however I was surprised with the capability and flashiness JS/PHP brought to the table. The C++ version of our project seemed underwhelming, but translating it to JS/PHP, I appreciated the work we all put in, much more. I definitely look forward to learning more about JS/PHP web development and becoming better acquainted with it all.

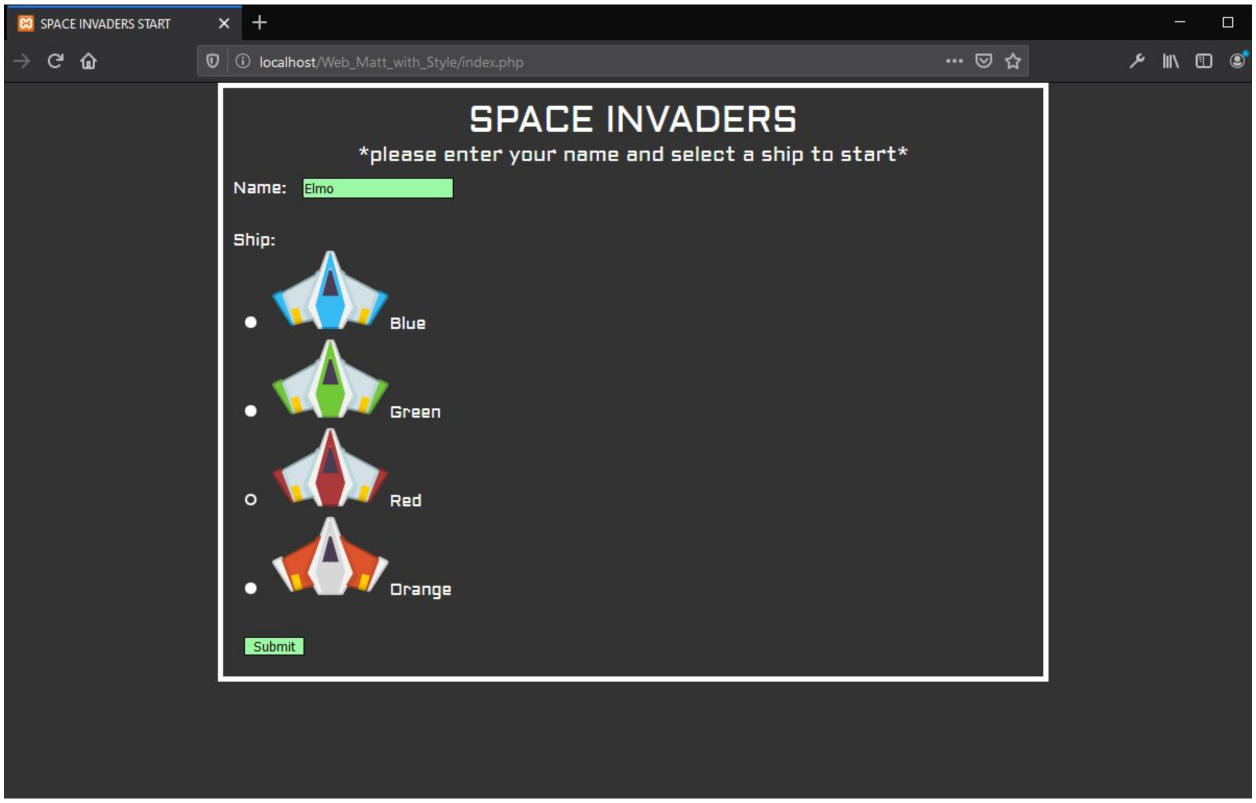
Favorite part of project:

My favorite part was watching our code come to life in a manner that would strike me as professionally done. Working as a team, I appreciated what we could achieve together and look forward to a future career in software development.

What you learned:

Through this experience, I learned PHP and JavaScript incorporate what seems to me, to be such a vast library of functionality, something you could only achieve in C++ by including a plethora of libraries. I got first-hand experience debugging and learning these various new functions. Last but not least, I gained invaluable experience working in a group on a large project.

Gameplay With Sample IO



SPACE INVADERS START

localhost/Web_Matt_with_Style/index.php

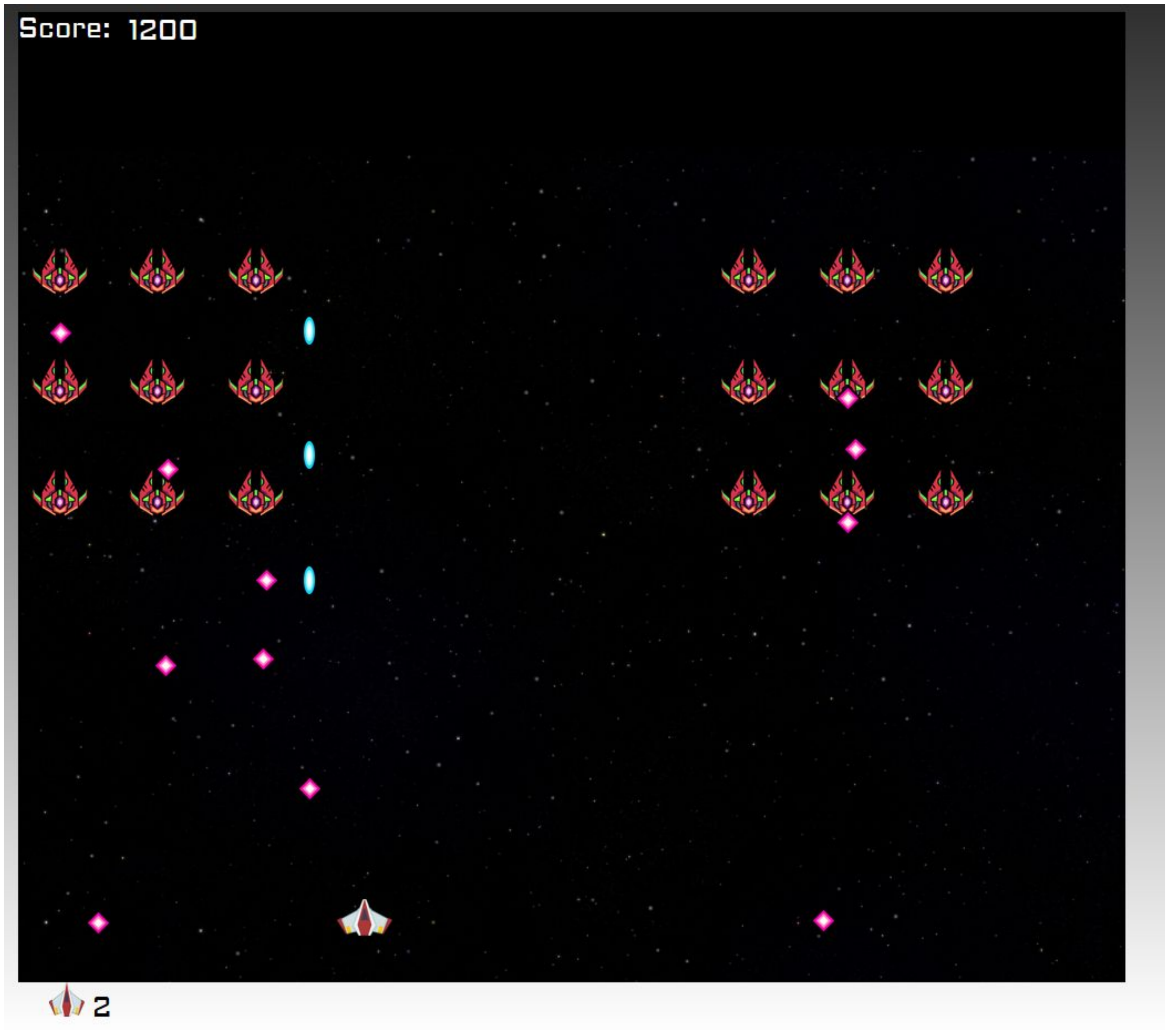
SPACE INVADERS

please enter your name and select a ship to start

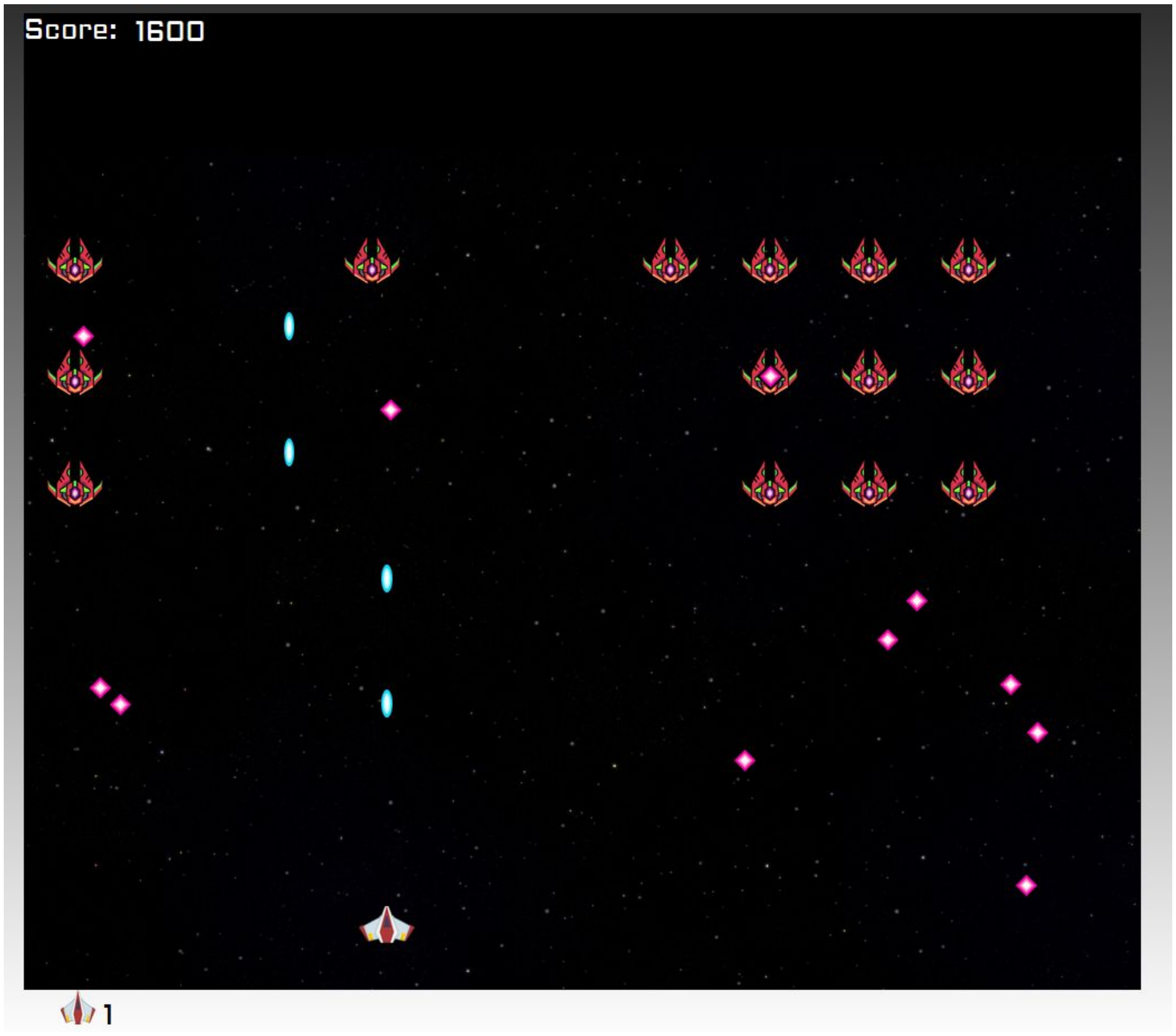
Name: Elmo

Ship:

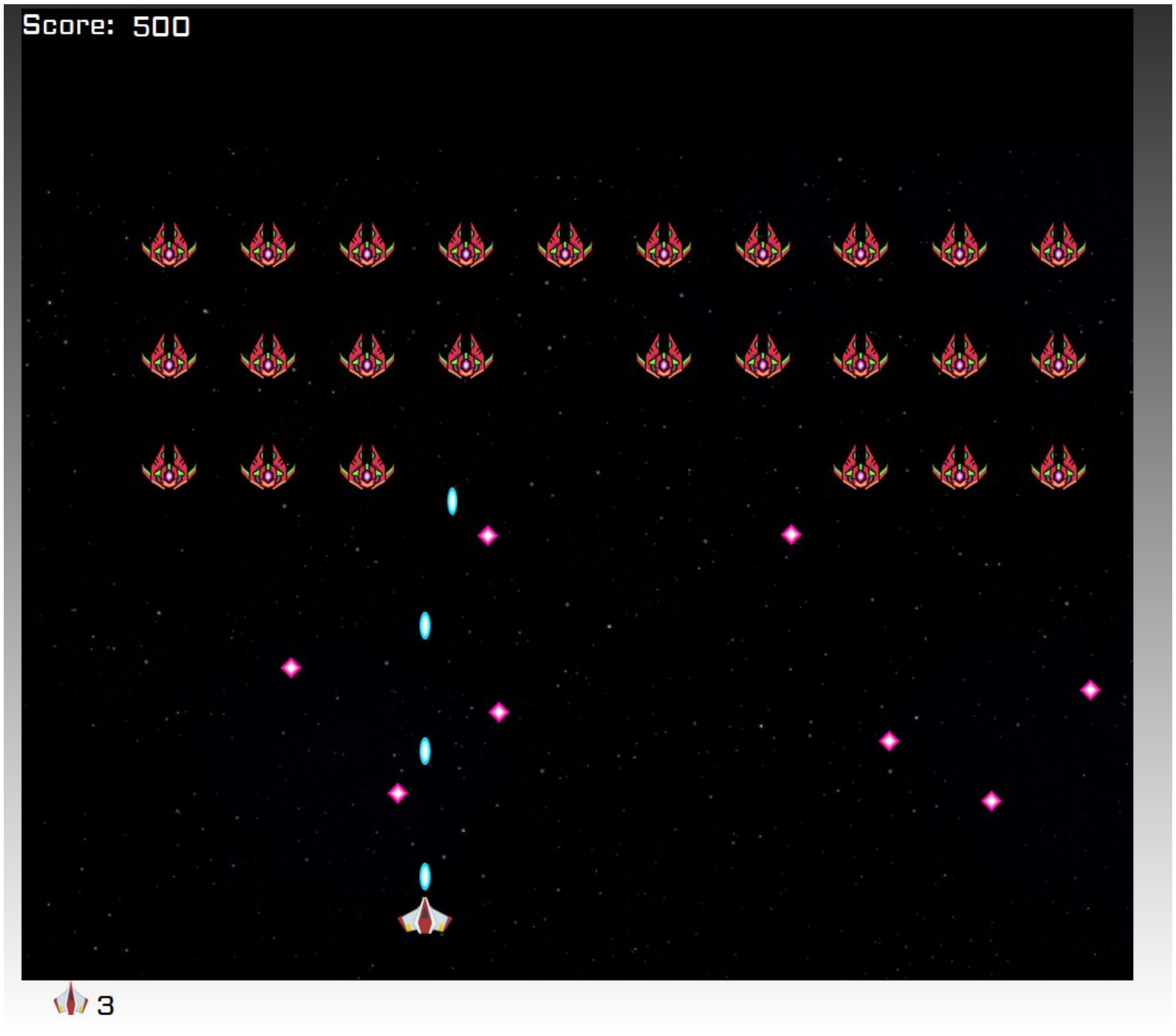
Index.php



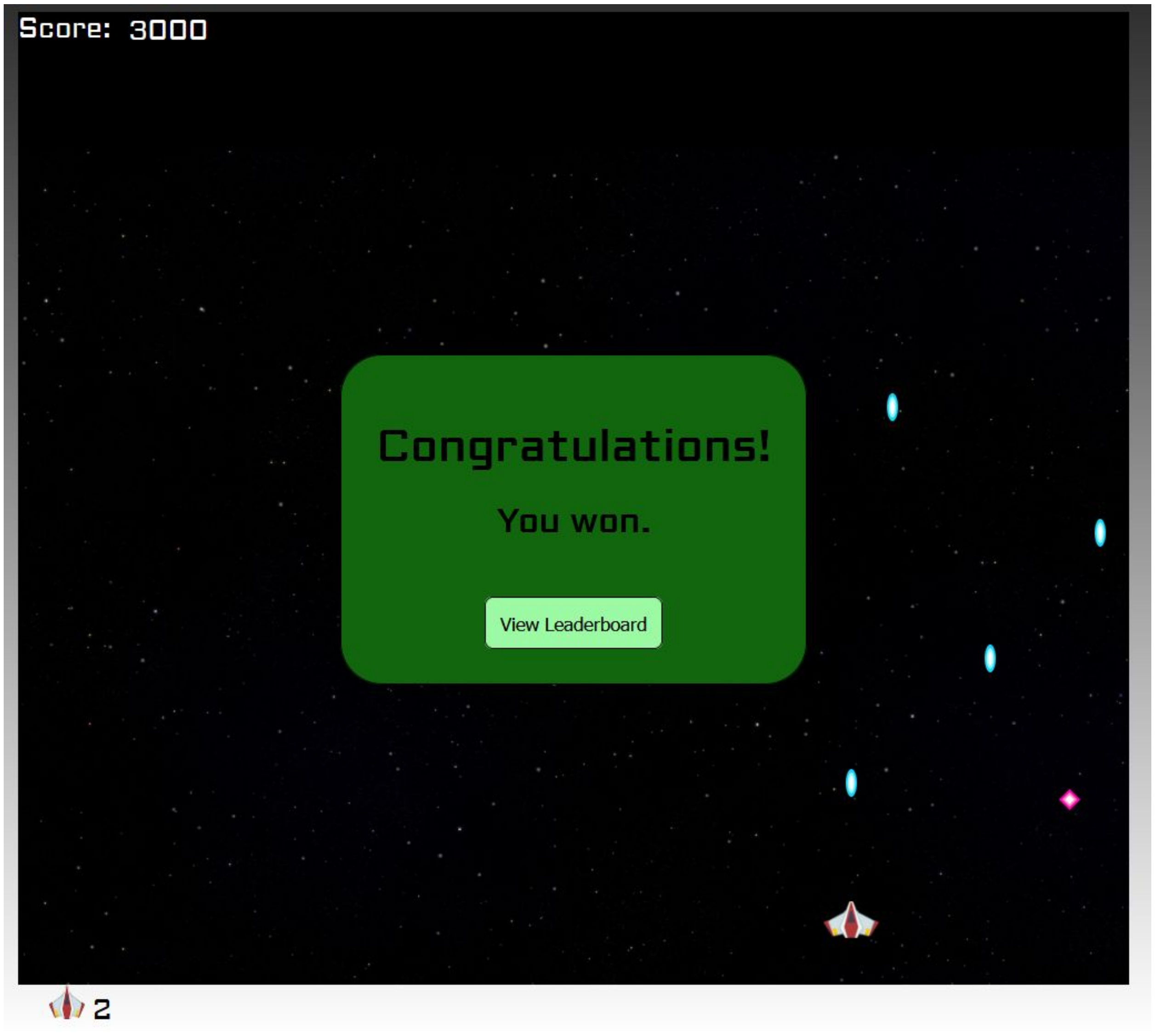
Gameplay 1



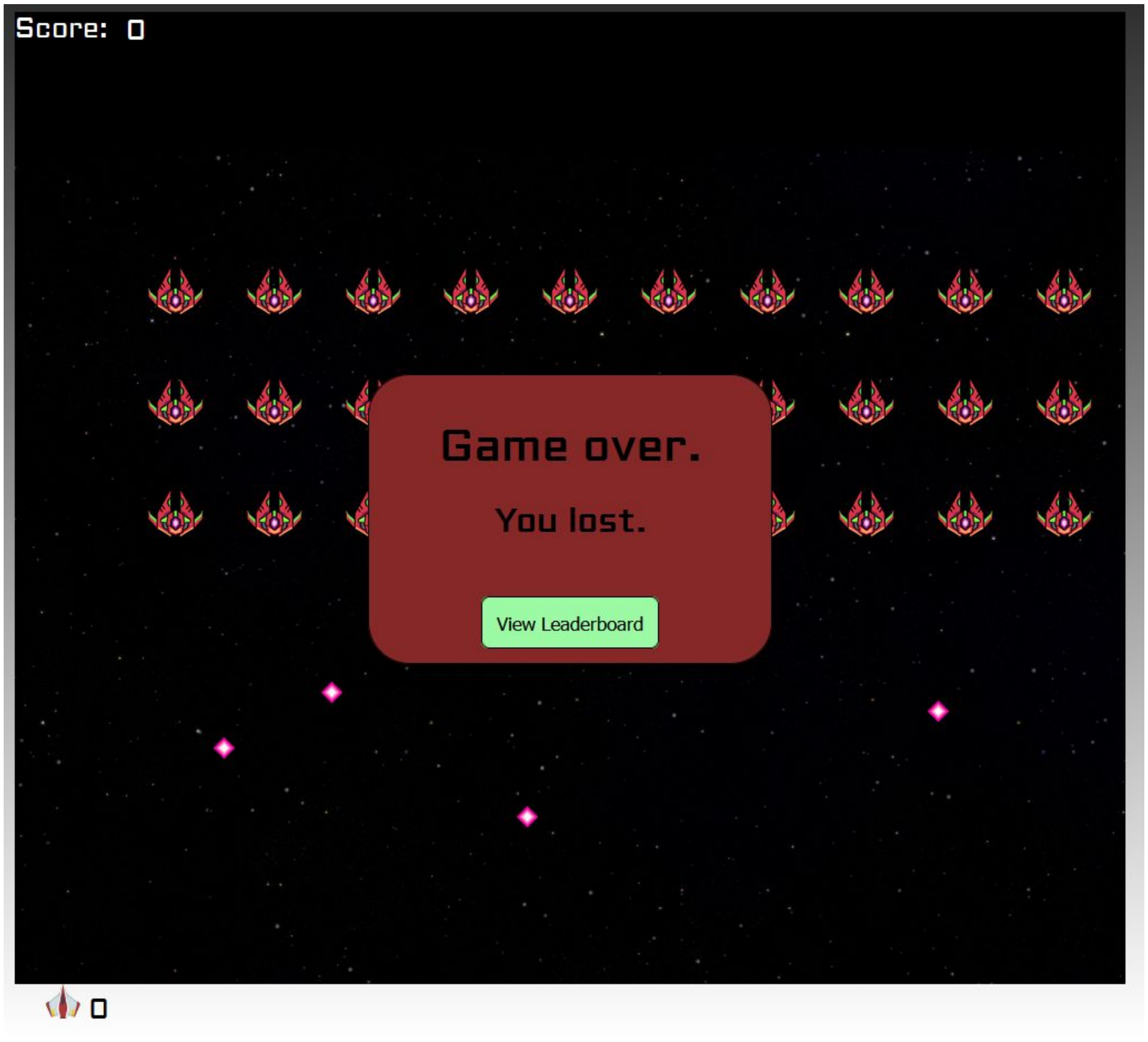
Gameplay 2



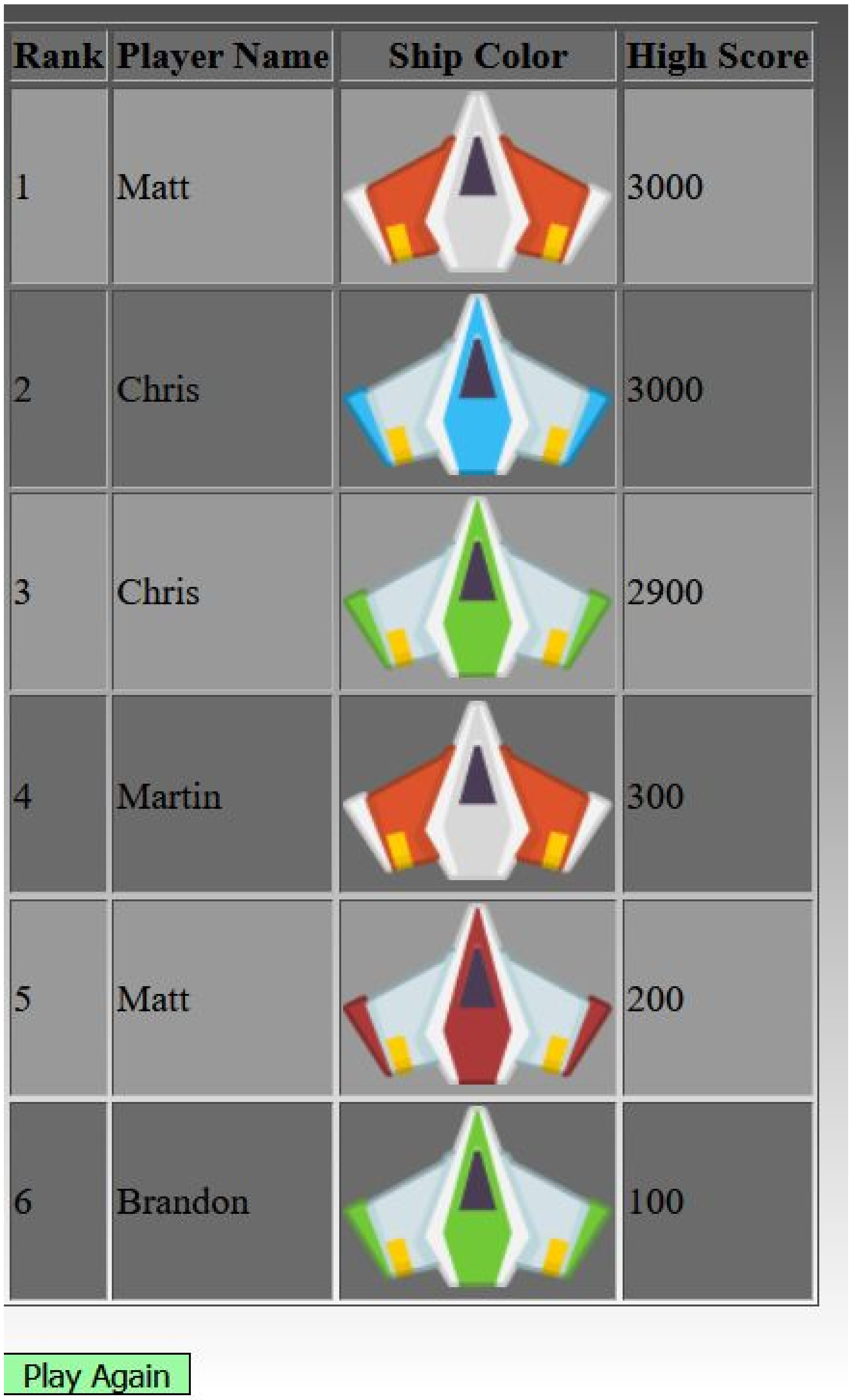
Gameplay 3



Win



Lose

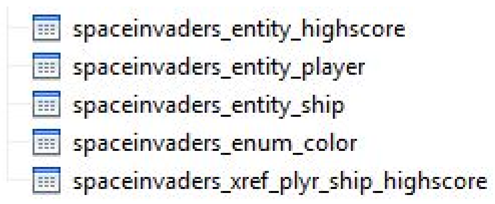


Rank	Player Name	Ship Color	High Score
1	Matt	🟧	3000
2	Chris	🟦	3000
3	Chris	🟩	2900
4	Martin	🟧	300
5	Matt	🟥	200
6	Brandon	🟩	100

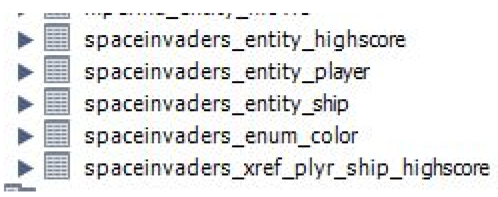
Leaderboard.php

Database

For the RCC remote database here are the tables specifically for the Space Invaders group. The screenshots are from OpenOffice(left) and MySQL(right):

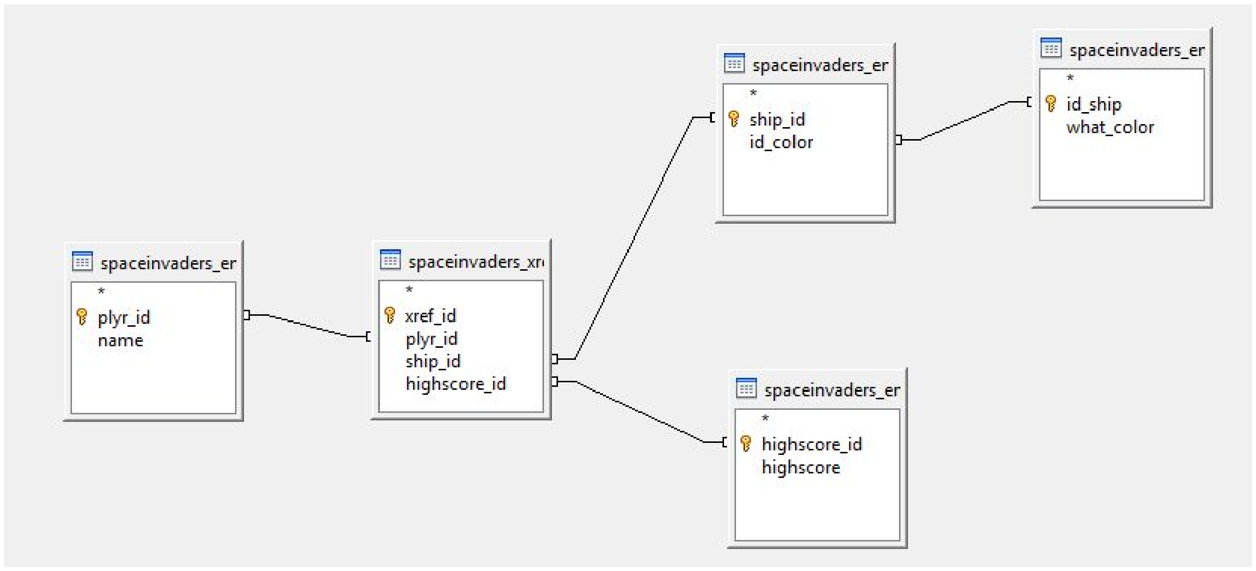


spaceinvaders_entity_highscore
spaceinvaders_entity_player
spaceinvaders_entity_ship
spaceinvaders_enum_color
spaceinvaders_xref_plyr_ship_highscore



spaceinvaders_entity_highscore
spaceinvaders_entity_player
spaceinvaders_entity_ship
spaceinvaders_enum_color
spaceinvaders_xref_plyr_ship_highscore

Here is essentially the flowchart or Query Design for the tables in Space Invaders:



There are 3 entity tables:

Player, Ship, HighScore

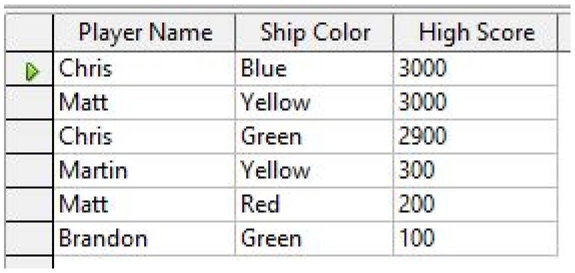
1 enum table:

Ship Color

1 xref table:

Player to Ship to HighScore

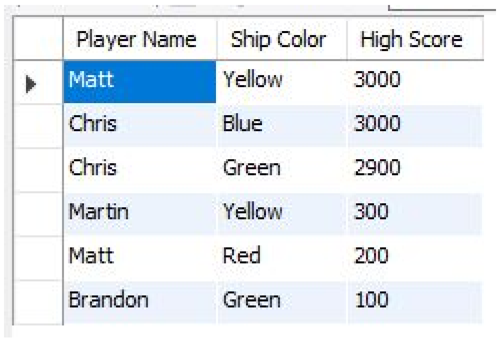
Here is also an example of a table that I captured in OpenOffice during the development process:



Player Name	Ship Color	High Score
Chris	Blue	3000
Matt	Yellow	3000
Chris	Green	2900
Martin	Yellow	300
Matt	Red	200
Brandon	Green	100

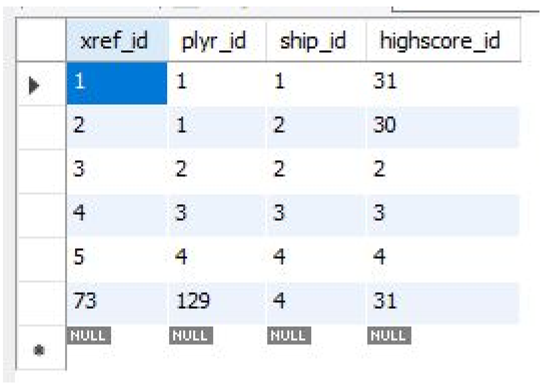
The usual way that we order the chart is by descending high scores. We also allow players to have different high scores with different ships since we felt this was more realistic to the game. This translates later into the actual leaderboard later in the documentation.

Here is the same table run as a query in MySQL:



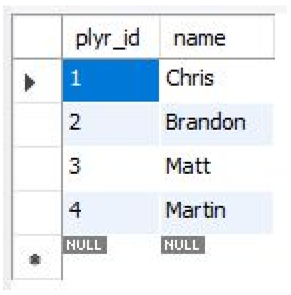
Player Name	Ship Color	High Score
Matt	Yellow	3000
Chris	Blue	3000
Chris	Green	2900
Martin	Yellow	300
Matt	Red	200
Brandon	Green	100

Here is also the xref for the board above that links up the player, ship, and highscore. Notice the number of times we've tested the game by the number of players entered with the plyr_id variable:

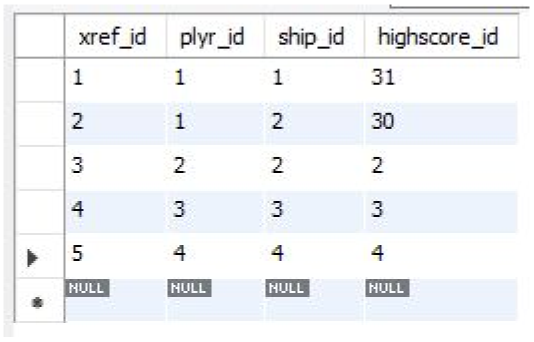


	xref_id	plyr_id	ship_id	highscore_id
▶	1	1	1	31
	2	1	2	30
	3	2	2	2
	4	3	3	3
	5	4	4	4
	73	129	4	31
*	NULL	NULL	NULL	NULL

I’m going to reset the tables now, this is what they look like:

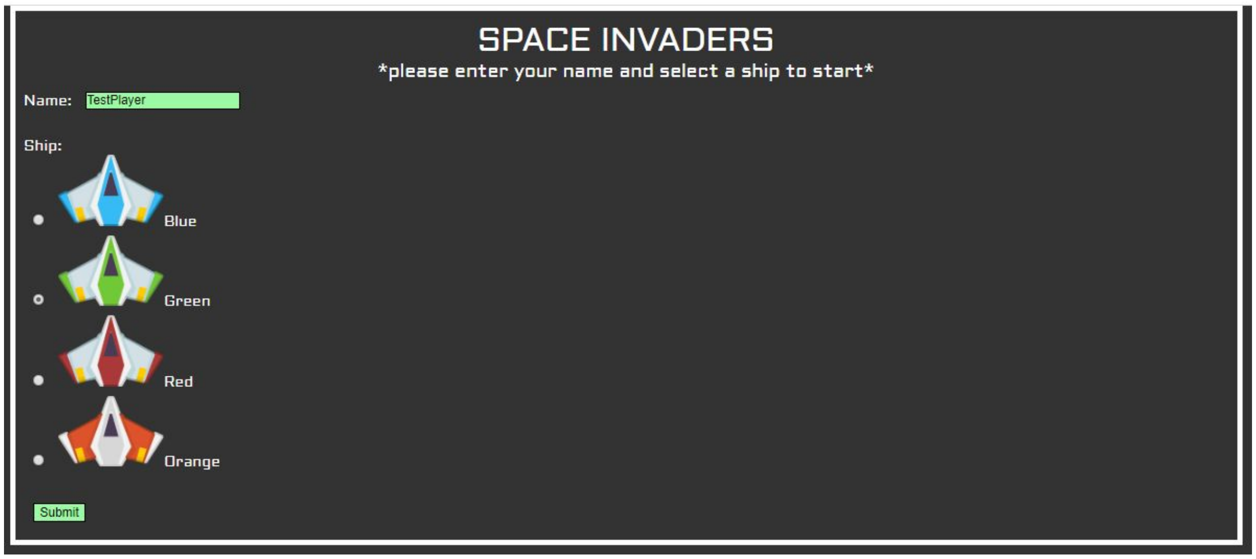


	plyr_id	name
▶	1	Chris
	2	Brandon
	3	Matt
	4	Martin
*	NULL	NULL



	xref_id	plyr_id	ship_id	highscore_id
	1	1	1	31
	2	1	2	30
	3	2	2	2
	4	3	3	3
▶	5	4	4	4
*	NULL	NULL	NULL	NULL

I am going to go to the game and have a player of ‘TestPlayer’ with a green ship:



SPACE INVADERS

please enter your name and select a ship to start

Name: TestPlayer

Ship:

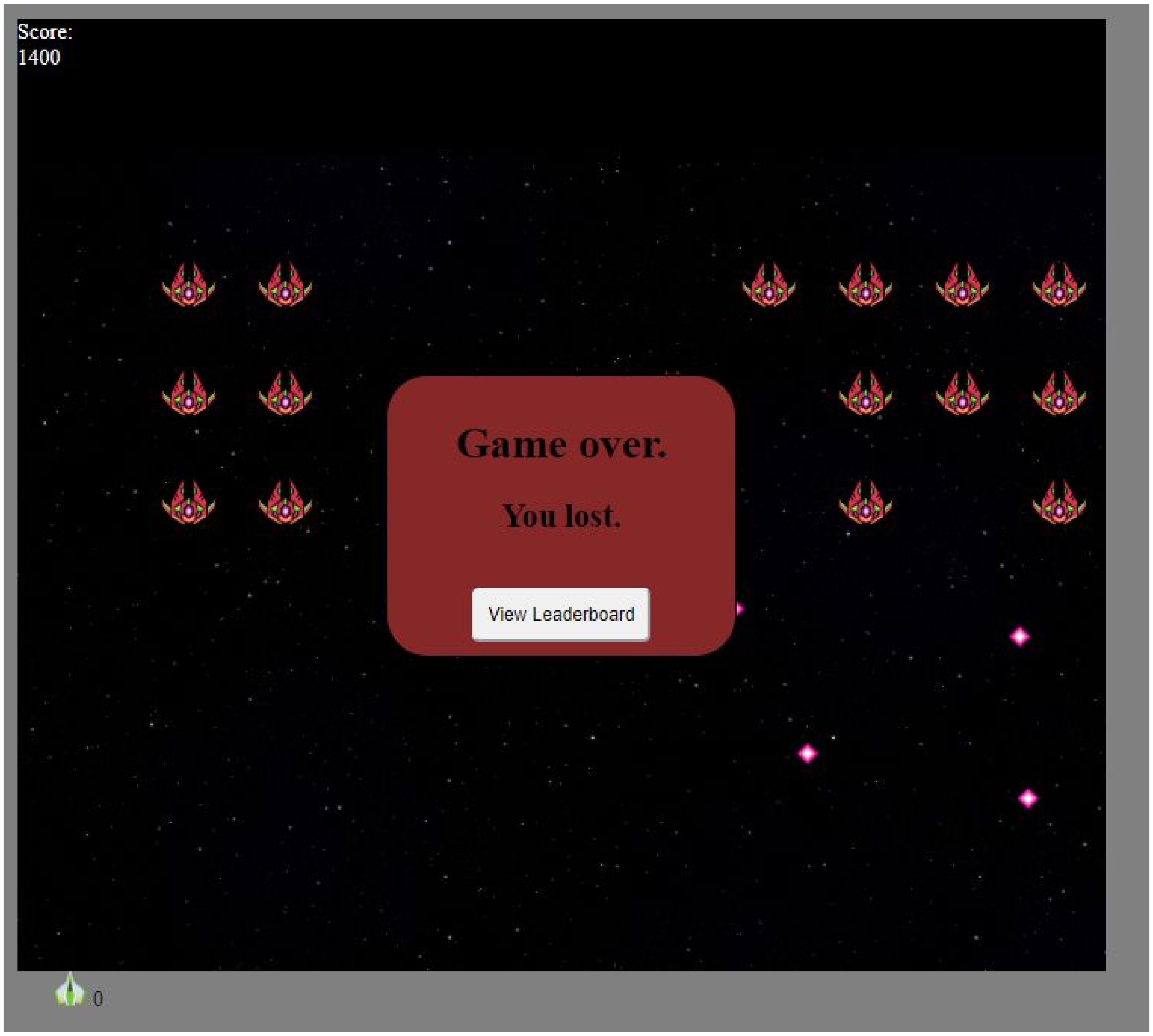
Blue

[✓ Green]

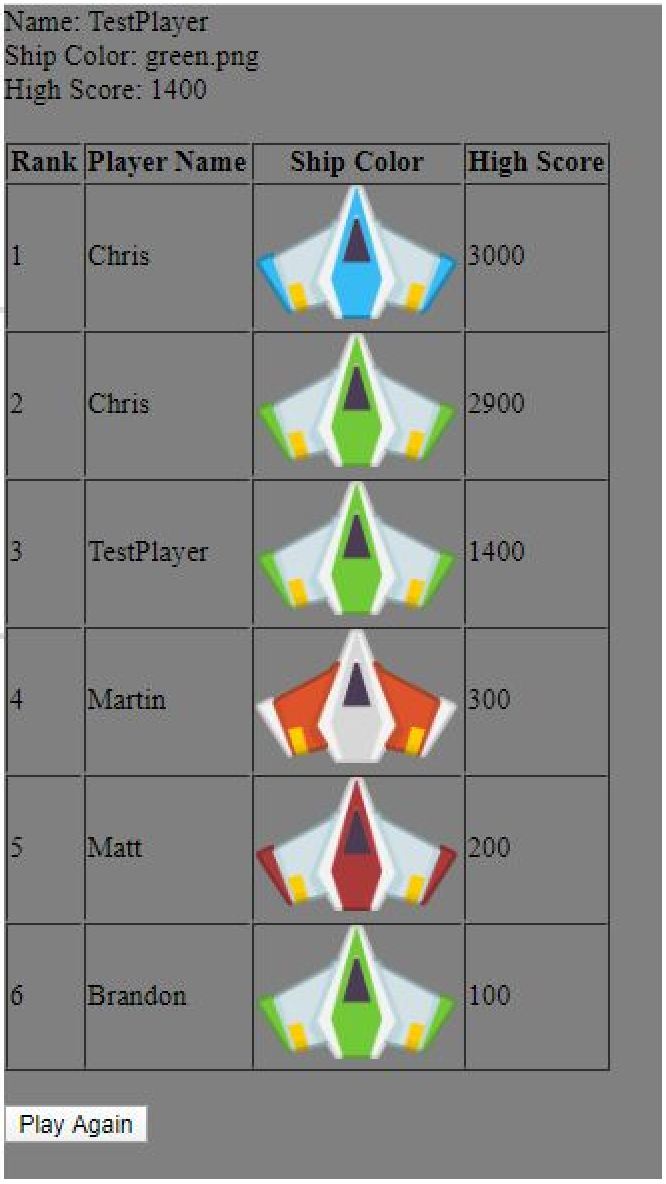
Red

Orange

I got a score of 1400 before losing:



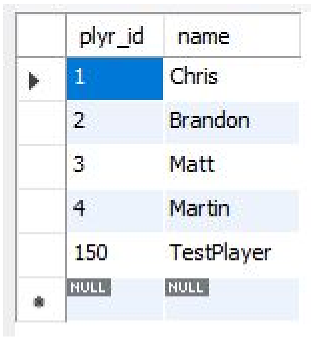
Here is the resulting leaderboard too. It gives the data on top of the name, ship color, and highscore earned. Then below it is shown that I am in 3rd place on the leaderboard with a score of 1400.



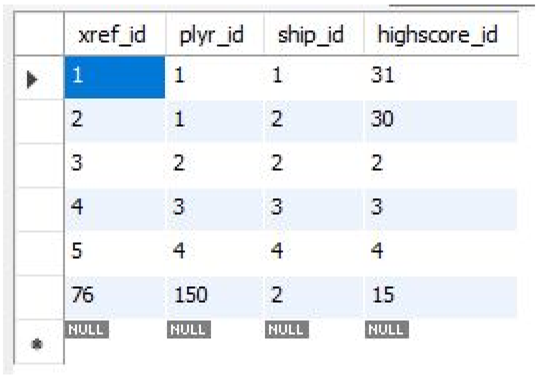
Name: TestPlayer
Ship Color: green.png
High Score: 1400

Rank	Player Name	Ship Color	High Score
1	Chris		3000
2	Chris		2900
3	TestPlayer		1400
4	Martin		300
5	Matt		200
6	Brandon		100

Then let's go back and look at the tables in the RCC remote database. It’s updated with the new data:

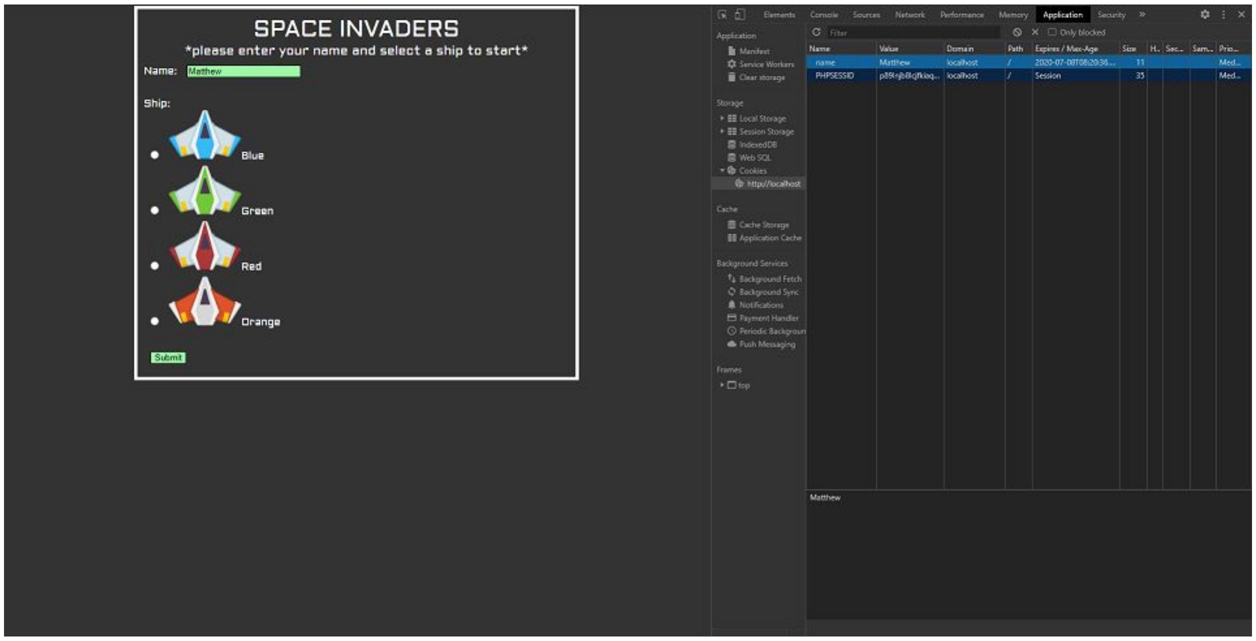


plyr_id	name
1	Chris
2	Brandon
3	Matt
4	Martin
150	TestPlayer
NULL	NULL



xref_id	plyr_id	ship_id	highscore_id
1	1	1	31
2	1	2	30
3	2	2	2
4	3	3	3
5	4	4	4
76	150	2	15
NULL	NULL	NULL	NULL

Cookies - Sessions

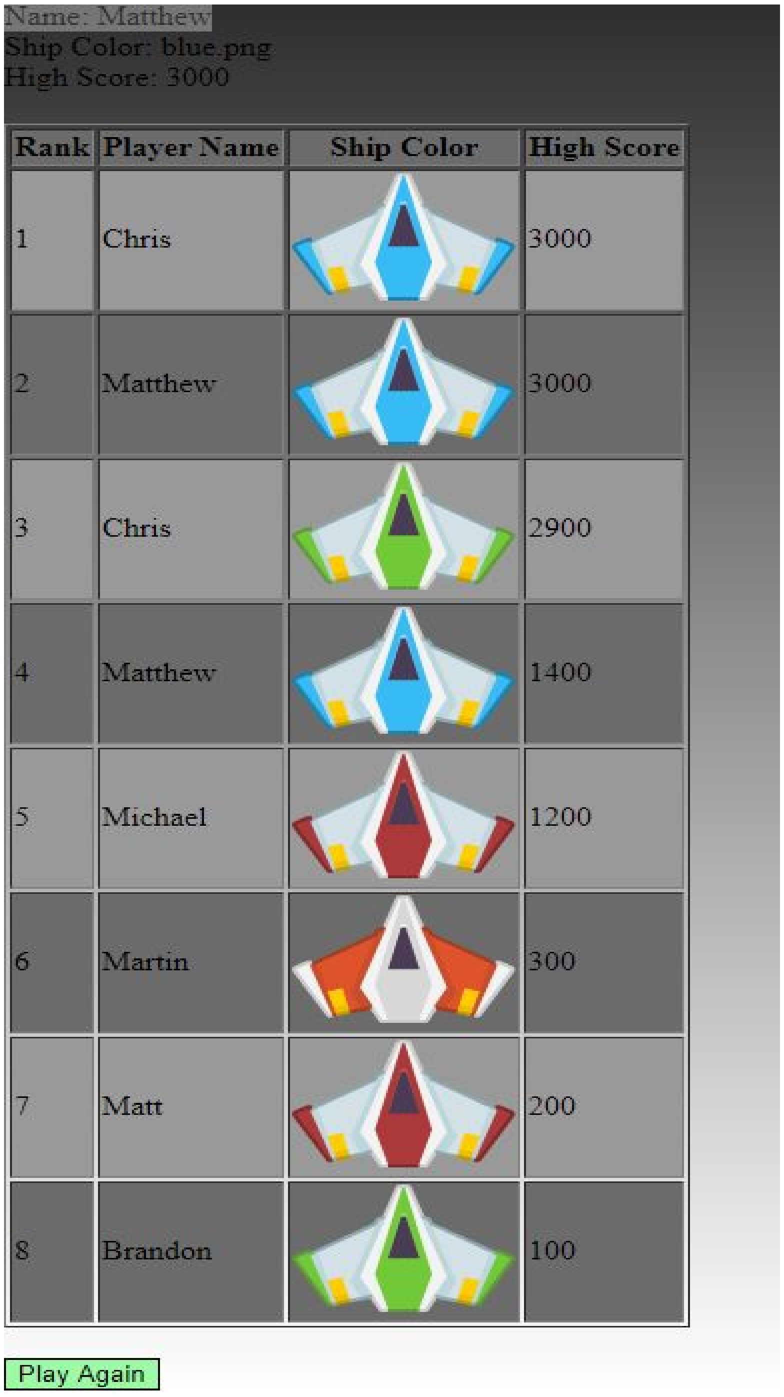


Name: Matthew

Name	Value	Domain	Path	Expires / Max-Age	Size	H...	Sec...	Sam...	Prio...
name	Matthew	localhost	/	2020-07-08T08:20:36...	11				Med...
PHPSESSID	p89lnjb8lqfkiaq...	localhost	/	Session	35				Med...

Matthew

In the Index.php file, we have a cookie to remember the user's name. When the user wants to play again, an if statement will determine if a cookie is set and if there is, the name input will be auto-filled with the cookie's data.



Name: Matthew
Ship Color: blue.png
High Score: 3000

Rank	Player Name	Ship Color	High Score
1	Chris	🟦	3000
2	Matthew	🟦	3000
3	Chris	🟩	2900
4	Matthew	🟦	1400
5	Michael	🟥	1200
6	Martin	🟧	300
7	Matt	🟥	200
8	Brandon	🟩	100

And finally, at the leaderboard page, the cookie is the preference for retrieving and displaying the name, if a cookie is not set, the else statement pulls from $_SESSION.

Pseudocode

```
Index.php
    Use form to get username and ship color from user.
    Send data to game.php.
Game.php
    Set a cookie with form data to hold the user's name.
    Set foundation for game board, dialog boxes, and images.
    While game is not won or lost:
    App.js
        Create board object, leaderboard object, player object, enemy object and array.
        Run scripts to loop enemy movement and attack.
            Create and delete enemy laser objects.
        Accept user input for movement and attack.
            Create and delete user laser objects.
        Check if the game is won or lost.
    Win or lose: Display dialog box alerting user of win/loss.
    Take the user to leaderboard.php.
```

Leaderboard.php

Start session to hold username, ship type and color.

Calculate score.

Query MySQL database to retrieve past user’s scores.

Display scores with ranks, names, ship color picked, and high scores.

If high score, insert into the database the score.

Offer the user the option to play again.