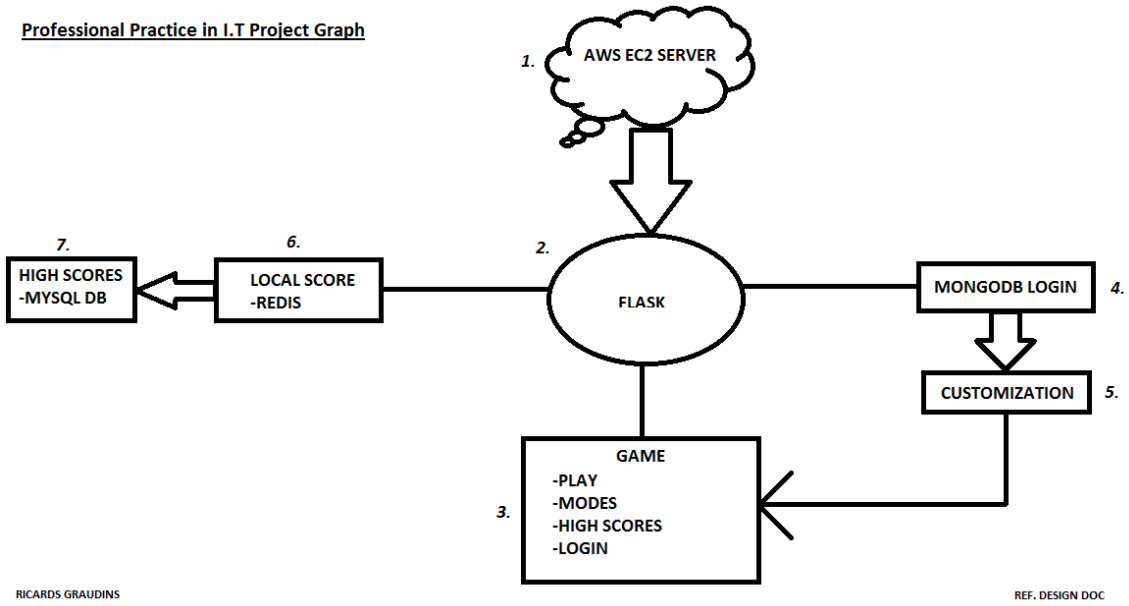
Galaxian

Professional Practice in I.T Project

Ricards Graudins | Software Development | 24/4/2017

INTRODUCTION

The goal of this project is to create my own variation of the classic arcade game known as Galaxian that was released in 1979. Initially I had considered developing the game in C# for windows store using visual studio. After taking into consideration that the game needs an internet connection to connect to a database which would then require a privacy policy to get certified, and the fact that it is already a tedious process to get applications certified in the first place with windows store, I had decided to develop the game in JavaScript and host it online.



The above image displays a simple illustration of my design plan for the project. The game is to be developed using the flask framework to host the game locally and provide templates for user registration, login, high scores, customization, profile, and the game template. User information is to be stored using a MongoDB database stored at mlab.com which is a cloud database service that hosts MongoDB databases. Successful user authentication allows access to the customization template which allows the user to modify certain parts of the game, primarily the player's spaceship. Redis is to be used to store player's high score and the user has the option of saving that high score which would then be stored on a MYSQL database and then displayed on the high score template. Once the project is complete it will be hosted on an AWS EC2 Server which will allow users remote access to the game.

TECHNOLOGY USED AND WHY

1. Flask
2. MongoDB
3. Redis
4. MYSQL
5. Canvas
6. AWS
7. PyMongo
8. Bcrypt

Flask was chosen for several reasons but for the most part it's because it is a microframework for python that gets the job done for this project. It allows me to host the application locally, provide templates and it matches well with MongoDB. Flask provides PyMongo which allows the use of Flask's normal mechanisms to configure and connect to MongoDB and Bcrypt which unlike MD5 and SHA1 is intentionally structured to be a slow hashing algorithm that provides high security.

MongoDB was my database of choice because it is a schema less database which allows me to jump right in and start storing data without having to worry about creating a RDBMS. If necessary it is easier to scale out and you can choose what level of consistency you want depending on the value of the data e.g. faster performance – fire and forget inserts to MongoDB, slower performance – wait until insert has been replicated to multiple nodes before returning.

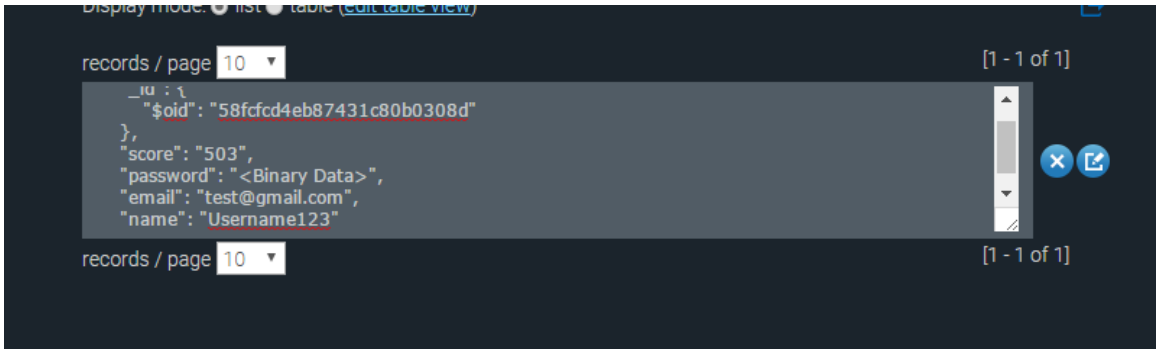
Initially I had chosen Redis because I was eager to test it out and I thought that it could be useful for storing high scores for one reason and that is because it is an in-memory database. Meaning the data is volatile and is not stored for a long time unless the user decides to store the data which I would insert into a MYSQL database. The plan was to store the scores and the usernames of only the top 10 high scores inside the MYSQL database to keep it very small and have it locally so that it can quickly display the top 10 scores inside the high score template. However, as the deadline was fast approaching there simply wasn't enough time to implement Redis and MYSQL and I had started to think that idea was over complicating things and thus I decided to store the score inside the MongoDB database.

I went with the HTML5 canvas for two reasons. I wanted to build on the basics I had learned from the module graphics programming and to see if I could build an actual game using canvas. The alternatives for canvas that I had considered are SVG and flash. After doing research on flash I have concluded that canvas is

the better option and many reviews have stated that canvas is the way to go and will be the norm for future browser games. SVG was ruled out because it is not suited for game applications and focuses more on drawing complex shapes.

I decided to host the game remotely using AWS. The AWS EC2 Server provides more than enough to host the game lag free and saves me the trouble of getting the game up on a website.

Screenshot of the data stored inside MongoDB:



As you can see there isn't a lot of data being stored per account only what's necessary and the email isn't currently utilized but can be used in the future to notify users that an update has come out. Only one score is stored and it is overwritten every time the user wants to update the score, there is absolutely no need to store multiple scores and they would just take up unnecessary space.

FEATURES IMPLEMENTED

Reference my GitHub repository at https://github.com/RicardsGraudins/Professional-Practice-in-I.T-2017-Project for full documentation of features inside README.md and the code files for more technical information. The README.md also contains a commit summary which shows the order features were implemented in, the installation and user guides are also included in the README.

LIMITATIONS

- Canvas size – The game is currently optimized for 1920 x 1080 resolution meaning any smaller resolution does not display the entire game. To get around this limitation use chrome, press f12 to access the developer tools and press the toggle device toolbar button which is the second top left icon to automatically scale down the canvas so the entire game is visible. The canvas size can further be altered by pulling on the edge of the toolbar or by using input fields at the top to change the size. The game hasn't been tested on a resolution above 1920 x 1080.
- The game is constantly reusing objects instead of creating new ones and as a result runs at a steady frame rate. However, it may have latency issues on older devices as is with all games.
- .wav is not supported on internet explorer and since all the sound effects excluding music are .wav will not be played on internet explorer.
- Accounts can be created using any input e.g. the username can just be the letter 'a' and same goes for password and email.
- The game uses arrow keys and space bar which makes it unplayable on devices that aren't connected to a keyboard.
- The AWS EC2 Server hosting the game has low specs and as a result the game takes a while to load and when it does load it may not play the sound effects on time including several other things. To overcome this limitation the game should be hosted on a better server and the game should not be playable before all resources are loaded.

KNOWN BUGS

- There is a 1 second delay that can occur at any point in time during the first time the game is played. This is most likely because of not pre-loading resources.
- Explosion audio of enemy ships is sometimes skipped when the ships are destroyed consecutively which should not really be happening because the sound effect only lasts 1 second and the time it takes for the player's bullet to hit an enemy is about the same. This is not a big issue and may even be beneficial as hearing the sound effect repeatedly may get tedious.

RECOMMENDATIONS FOR FUTURE DEVELOPMENT

- There are similar functions throughout player.js and several objects that take in the same number of parameters. Although an abstract function was used to cut back on the lines of code for mid, left and right enemies, there could have been more abstract functions for other features.
- Throughout the project, I had to go back and forth between a lot of code and I came to the conclusion that having relatable code next to each other would make it easier to identify where and what needs to be changed while also cutting back on the time needed to locate parts of code.
- A considerable amount of time was lost trying to implement a functioning object pool that was to be used for several features that had to be scraped for the time being in order to get other features implemented on time. For future development of this project I recommend getting the object pool implemented first as it can be used for many features such as spawning enemies and bullets.

EXPANDING THE GAME

There are many things that can be done to make the game better and more complete although when compared to the original game that was released in 1979 it can be considered finished. Nowadays games are expected to have more depth and replayability and here are some of the things I would like to add to the game:

- Customization, I initially intended to allow the player to customize the way the player's spaceship looks by letting the player upload an image to the database and then using that data to replace the sprite inside the game. There are several things to keep in mind for doing this and one of them is having the image be an appropriate size while keeping the hitbox rational.
- More enemy types, it goes without saying that having more enemies with different shooting patterns and mechanics will make the game more interesting.
- Animations, so far the sprites remain stationary as they move across the canvas, the game would be more visually appealing if they had animations as they moved such as spinning or exploding when they get hit.
- Levels and bosses, so far the game only contains one boss and the same types of enemies that spawn endlessly. The game could be made longer and more difficult by adding multiple levels with bosses at the end of each level.
- More power-ups, adding more power ups such as allowing the player to shoot several bullets simultaneously for a limited time or penetration rounds that go through enemies can make the game a lot more enjoyable.
- Movement using vectors, instead of using x and y, vectors can be used to make the enemy movement unpredictable and less linear.
- The first boss fight can be made more difficult by adding more mechanics such as allowing the twins to shoot in several directions as they move or having more enemies spawned during the leviathan phase.

CONCLUSIONS

There are two main conclusions I arrived at while working on this project. The first one being it is possible to create a decent smooth running game using the HTML5 canvas which is what I initially set out to do. Several JavaScript libraries could have been used to speed up the development which I came across rather late in the project to make use of and I should have definitely researched those before starting the project. The second conclusion was that I should have had better time management and although it was a busy semester for me I should have planned ahead and tackled the project bit by bit instead of all at once towards the end.