Silesian
University
of Technology

Faculty of Automatic Control,
Electronics and Computer Science

Internet Technologies – project work

Cyberventure game

Authors:

Mateusz Siedliski

Radosław Tchórzewski

Year 2022/2023, semester 5, group 6, section 9:

Project supervisor: mgr inż. Oliwia Krauze

Gliwice 2023

Contents

Chapter 1

Introduction

When "Flappy Bird" came out in 2013, it has brought a lot of attention. There were reports of people breaking their devices, thanks to the rage caused by difficult gameplay. The game concept was simple. Try to avoid obstacles moving forward by tapping the screen to go up while gravity constantly pulls you down.

We've decided to create a new and improved version of this game. During our project we've explored new fitting and interesting graphic designs. We've used many languages and technologies we haven't used before, which was made possible thanks to the web developer bootcamp we took [1]. Our game contains features such as adding additional pickups, laser meant to be avoided and gravity switching. We were aiming to create a feeling of fair gameplay with options that reminded us of flash games hosted on various websites, and we think we've managed to achieve that. Gaining knowledge about which tools we have to use was vastly improved thanks to the YouTube's tutorials [2] [3] [4].

Our game consists of two distinct parts: client side website and server with MySQL database responsible for providing global leaderboard. A player can see top 5 scores of the best players, as well as the highest score from the current session and current score for the round. A JavaScript structure based on p5.js library is responsible for handling rendering assets and preparing each frame. We needed jQuery library to communicate between client and server side. The game is hosted on free external hosting.

Chapter 2

Aim and scope of the project

Although many versions of "Flappy Bird" were created, none of them seemed to change the concept enough. We've created an experience that is unique thanks to the graphic design and challenging in a fair way without encouraging anger. We wanted to give players an additional sense of challenge by implementing a global leaderboard.

We've decided to use p5.js library as our main starting point. Furthermore, we made this decision knowing that this library contains powerful tools for creating looping graphics and rendering images on website correctly. A second library called jQuery was chosen for its event handling, animation and Ajax handling.

MySQL was chosen as our database system thanks to it being free and having easy to use interface. To prevent overfilling the database we've used a job scheduler called cron.

Chapter 3

Schedule

3.1 Schedule approved at the beginning

- Learning HTML and creating 1st version of homepage
- Learning CSS and creating styles for the webpage
- Learning JavaScript and creating first sketch of the game
- Creating game physics (collision)
- Learning PHP and creating simple communication between website and the database for storing high scores
- Adding Leaderboard and accounts system
- Adding collectibles/game speed mechanic
- Implementing dynamic background (changed by earning more points, constantly moving behind player to give impression of speed)
- Creating final graphics, sprites and finalizing project

3.2 Schedule reflecting actual work

- Learning HTML and creating 1st version of homepage
- Learning CSS and experimenting with styles (made temporary ones)
- Learning JavaScript and creating first sketch of the game
- Creating game physics (collision) and implementing temporary sprites
- Reworking background rendering and collision to optimize code
- Learning PHP and creating communication between website and the database for storing players scores. Adding user input after game is ended.
- Implementing CSS animations for various elements. Finalizing buttons design. Reworking structure of CSS files.
- Major changes to background rendering system. Finalizing Leaderboard. Optimization of loading.
- Finalizing graphics with use of Midjourney AI. Adding sounds. Finalizing code structure.

Chapter 4

Software implementation

We have managed to create a game, providing each user with semi unique background rendering and smooth gameplay. Safeguards for overflowing database with empty or repeated user inputs were implemented. Our code structure is divided to different interactable windows, which allows us to expand this project easily, thanks to its compartmentalization.

We've decided to use an object-oriented programming approach and treat each gameplay element on the client side as a separate class.

4.1 Client side

Upon loading the page, the client side receives all the required files. Calculating correct coordinates of elements is handled on the user's machine to get faster results and to reduce the load on the server.

4.1.1 Main gameplay loop file

Game.js is a file responsible for handling correct loading of assets and keeping the game running. This file provides many functions to control gameplay elements and communicate with the database. After opening the page, this file waits for all the assets to load. After correct loading process, it provides a loop based on p5.js function in which all the other objects interact.

The block diagram presented in Fig.4.1 shows the order of function calls. All three are based on p5.js library. Preload() is used to load assets used on the page. After all required files are loaded, a flag is raised to communicate with the Draw() function. Setup() is called directly after Preload() and initializes all the classes, as well as sets the frame rate and distance between obstacles.

Draw() function waits for the flag from Preload() and Setup() completion. After both conditions are met, it calls custom-made functions which handle gameplay and window changes.

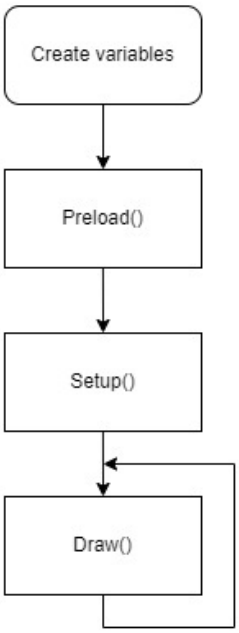

Figure 4.1. Block diagram representing loading and servicing page.

4.1.2 Collision Object

Collision object class handles obstacles stored in an array. It consists of variables responsible for current state and position of pylons and pickups if this option is turned on.

This class provides methods for checking collision with player, as well as for moving and rendering obstacles in the current frame. Those methods are called from Game.js for each object in the array, creating a way to keep many obstacles on screen at the same time. It is also responsible for deleting objects which are outside the screen to prevent memory leaks.

Height and position of passable part of an obstacle are generated randomly with rand() function. It is customized to have a different range, depending on the size of the user's screen.

Additional pickups are also handled by this class. A special boolean variable is used to determine if the object should contain bonus points. Collision and rendering is handled by methods described above.

Fig.4.2 represents variables which are used to render and move obstacles. Computations are based on three presented values and screen size to accommodate different window sizes. If the option responsible for additional points is turned on, then a pickup is placed based on hole center value. Each object contains two boolean values responsible for giving points to the player and generating pickups.

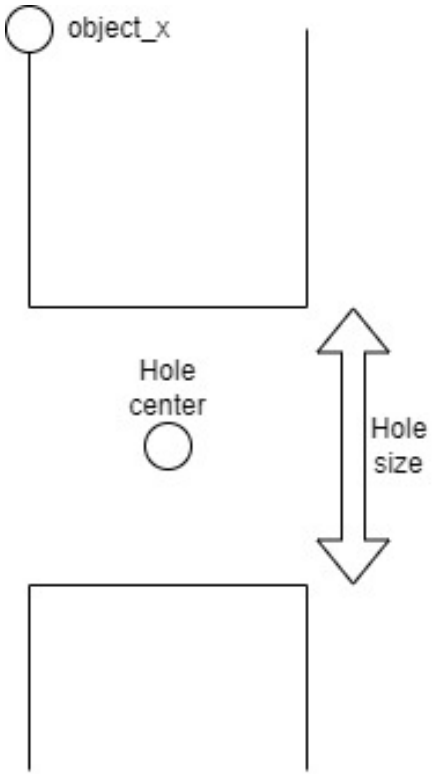

Figure 4.2. Representation of Collision object class coordinates.

4.1.3 Handling switching between options

To reduce usage of additional libraries, option handling was made with just JavaScript. Each window is represented by a div containing specific buttons, inputs and additional information. A file called div_master.js manages the structure of those divs by changing the CSS property responsible for element visibility and style. After switching windows, a special function is called, hiding all divs which aren't required. Values such as user input are reset as well.

Expanding this structure to handle more divs requires a future developer to edit a function called div_create_all() and add a new one responsible for handling the newly created div. Creation of this new element should be handled in the same file as well. A representation of current div structure can be seen in Fig.4.3.

Controlling sound volume and gameplay customization options is handled by functions in div_create_all() as well. Sounds can have two states, muted (default) and unmuted. When a user mutes a soundtrack, the song isn't stopped, but silenced.

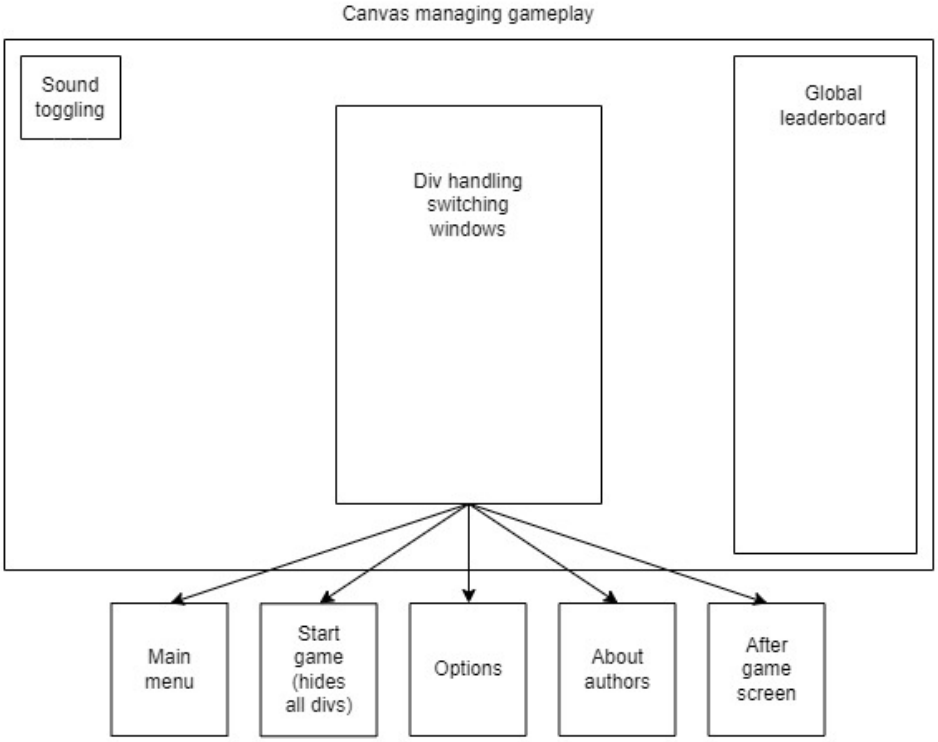

Figure 4.3. Representation of divs connection.

4.1.4 Laser obstacle

The concept of laser element was overhauled few times during development. The final design decision was to make the laser remove one point for each frame the player touches it. The lowest score is capped at 0 to prevent negative values. Animation of firing is made based on font awesome icons and CSS "-webkit-mask-image" property to add overlay on the base laser.

Future changes of laser style could be made by changing laser.css file, which contains gradient property describing laser appearance. By changing parameters in laser.js file, the future developer could edit animations speed and the time the laser is present on the screen. Fig.4.4 shows the algorithm used to render the laser firing animation.

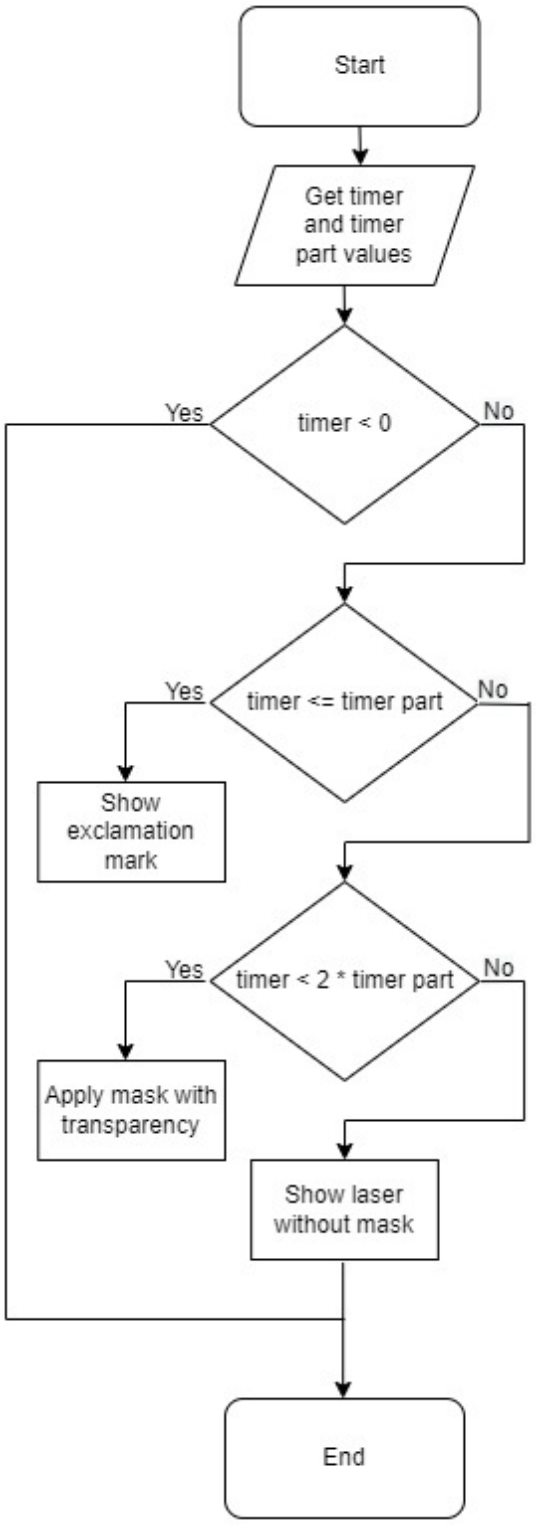

Figure 4.4. Block diagram representing process of laser rendering.

4.1.5 Image rendering

Achieving parallax effect with background assets was crucial. To achieve this, two classes were made. Image_rendering.js receives four parameters. Three of them are assets to produce the effect of randomly generated background. Fourth one is speed value. Background is divided into four main parts, all of them have different speed, which creates the desired effect.

Image_render.js handles changing coordinates for the asset it contains. To simplify code in Image_rendering.js it creates three entities of Image_render.js and manages them.

Player character, pickups and obstacles images are handles in separate files. Each asset is characterized by a set of coordinates which changes each frame. Some of them were created using free packages shared online [5] [6].

4.1.6 CSS options

To keep files managing clear, each distinct CSS option is handled in a separate file. For editing style of specific element, developer would have to find respective file and change properties of main element. As inspiration for part of our designs, we've used YouTube tutorials showcasing "Cyberpunk" themed creations [7] [8].

Some animations use root variables, which are changed through JavaScript. They are placed in a specific root.css file.

4.2 Server side

Main computations and user experience are handled on the client side. Our server is responsible for managing proper hosting and communication by Ajax requests which control players' scores in our MySQL database.

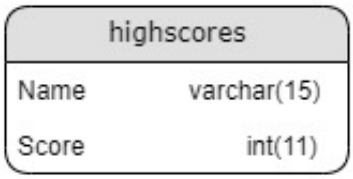

Figure 4.5. Table responsible for handling inserted scores.

We use database to handle only top scores of players. This meant that our database structure didn't have to be complex. Primary key was avoided as a deliberate choice, since only the best five scores are shown. Fig. 4.5 represents the table we've created.

4.3 Problems during development

During development, we've encountered several problems. Firstly, we've had to rewrite the physics system after implementing more complex images. This was caused by change in coordinates inputs for new functions and methods we use. Secondly, we've had to optimize generating obstacles and use a few tricks to make sure each resolution is served correctly. When using XAMPP as our temporary hosting on localhost, we've had few issues with page not refreshing correctly after implementing changes. This forced us to remove cookies and fully restart the page, which added some unwanted time during development.

Chapter 5

Summary

We are very satisfied with the result of our work. The application fully meets our design expectations. Graphic and sound design fits well with the gameplay loop we created. Using version control proved to be very useful during debugging and optimization stage. Creating a proper README.md for our GitHub page was an additional task we didn't foresee, however it allowed us to manage progress more clearly. To prove usage of version control system, we submit a link to our GitHub page, which can be found here.

Possible future project expansions are:

- adding additional gameplay mechanics such as increasing obstacle speed based on players score
- expanding leaderboard to contain separate tables for different configurations of customization options and changing global leaderboard from permanent top score to one being shown only for a specific amount of time

To summarize, during our project we've gained valuable experience in handling multiple programming tools at the same time, which is sure to be useful in our future learning. It can also help us get a job as it is a valuable portfolio piece. We've also experienced working as a team and improved our abilities to divide and manage tasks properly.

Bibliography

[1] K. Czekalski. Bootcamp programistyczny 2020 (pl): Web dev full stack. Course studied during 2022. [Online]. Available: https://www.udemy.com/course/bootcamp-programistyczny/

[2] T. C. Train. Tutorials introducing p5.js. [Online]. Available: https://www.youtube.com/c/TheCodingTrain

[3] M. Dane. Php programming language tutorial - full course. [Online]. Available: https://www.youtube.com/watch?v=OK_JCtrrv-c&ab_channel=freeCodeCamp.org

[4] P. with Mosh. Mysql tutorial for beginners [full course]. [Online]. Available: https://www.youtube.com/watch?v=7S_tz1z_5bA&ab_channel=ProgrammingwithMosh

[5] ambientCG. Facade 009 - part of background design. [Online]. Available: https://ambientcg.com/view?id=Facade009

[6] KITBASH3D. Assets used in project. [Online]. Available: https://kitbash3d.com/products/mini-kit-neo-city

[7] W. D. Michael. Cyberpunk 2077 inspired menu in css only! [Online]. Available: https://www.youtube.com/watch?v=Wmb-V87tmqI&ab_channel=WebDevMichael

[8] K. Powell. Animated glitch text effect with css only. [Online]. Available: https://www.youtube.com/watch?v=7Xyg8Ja7dyY&ab_channel=KevinPowell

Appendix A

Early development and graphic designs

Figure A.1. First stage of development with working gameloop

Figure A.2. Final player and obstacle designs

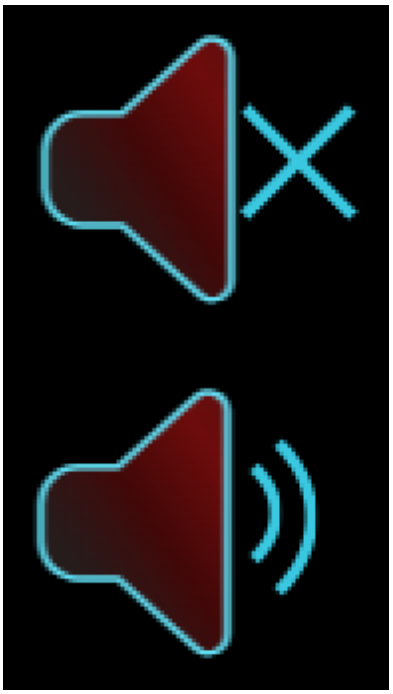

Figure A.3. Two states of volume icon, animation was added to toggle between them

Figure A.4. Assets used as part of background layering