CSY1018 WEB PROGRAMMING
ASSIGNMENT REPORT
TERM II

HORSE RACING GAME JAVASCRIPT

18406547

Diwas Lamsal

2018

CSY1018: Web Programming

Due for Issue (week commencing):	**Sunday, 25th February, 2018**	**Last Date for Submission:**	**Sunday 22nd April 2018 23:59:59**
Agreed Date for late submission:		Module Tutor:	Thomas Butler
Student ID:	18406547 - Diwas Lamsal		

Aspect (& weighting)	Excellent A	Good B	Satisfactory C	Needs some more work D	Needs much more work F
Functionality (40%)					
Design (20%)					
Evaluation (10%)					
Testing (10%)					
Code Quality and Efficiency (15%)					
Video Demonstration (5%)					

Specific aspects of the :he assignment that marker likes:	Specific aspects of the assignment that need more work:

Tutor's Signature:		Date:		Grade:	

The University of Northampton Policy on Plagiarism & Mitigating Circumstances will be strictly implemented. By submitting this assignment you are asserting that this submission is entirely your own/individual work.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION:

1.1 Introduction to the module and the assignment:

JavaScript – one of the most commonly used languages as of now, is used to make interactive webpages. It is the widely used programming language across the web. Lately, with the availability of different libraries, this useful language has gotten further simpler, some libraries even allowing the language to be a server-side programming language.

The second term of the CSY1018 Web Development (Web Programming – Term II) module deals with web scripting and adding behaviors to webpages made using HTML and CSS. JavaScript has been taught throughout the term and the given assignment assesses the level of understanding and implementation abilities among the students for the language. Given is a scenario where four different horses(racers) exist. Using JavaScript knowledge, an interactive horse racing game is to be designed.

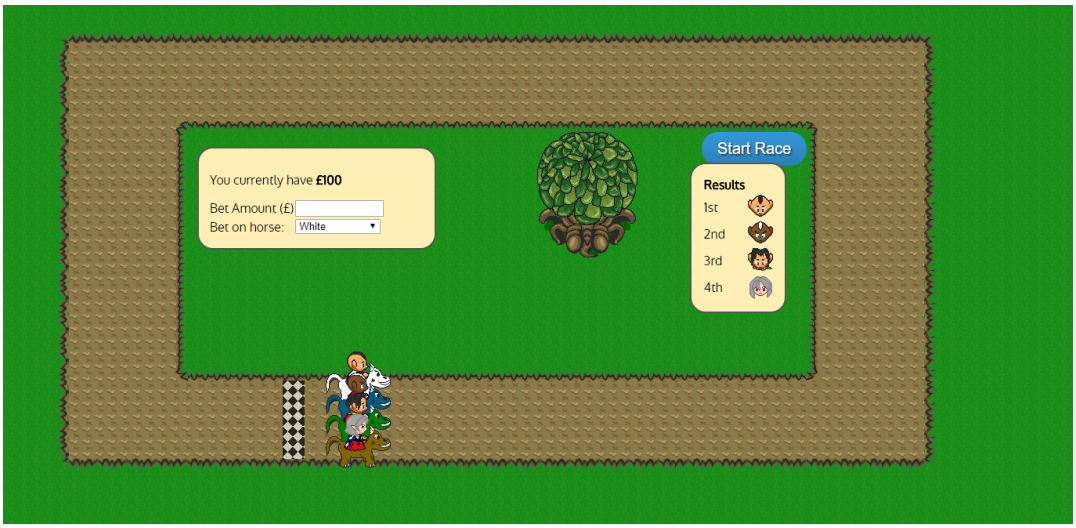

Figure 1.1 – Given Scenario

1.2 Introduction to the Report:

This report is to include the description of how different aims and objectives of the assignment have been dealt with. This report will enlist and describe the task requirements, the various functionalities added and implemented to complete the task, the methods used to make the program more efficient (for example: - reducing redundancy), the test methodologies utilized, the test results and feedbacks about the assignment. These individual portions will be thoroughly discussed under their respective headings in the upcoming topics.

1.3 Provided requirements:

In the assignment brief, different levels of tasks have been provided which include the requirements for assignment completion.

i. Basic:

The provided basic requirements state that the horses should move across the path (the track) until they reach the end of the race (the finish line). Applying different CSS classes on the horses would add animations to the racers while they race. This would make the horse seem like running towards different directions or standing facing a certain direction. The horses should follow the track as they race and pressing the start race button again after the race completes, the race should start once again.

ii. Intermediate:

The intermediate level difficulty of the assignment has some more advanced features enlisted. This portion asks to set different speeds for the horses as they race which would change at certain intervals during the race itself. The speeds would be random and so would the winner. The other feature asked by this difficulty level asks for different turning points for different horses in the track. The winner should be detected and the results be displayed as 1^{st}, 2^{nd}, 3^{rd} or 4^{th} in the tabular results box. The last feature enlisted asks to disable the start button while the race is going on.

iii. Advanced:

This can be referred to as the hard difficulty of the assignment. £100 should be the starting bet provided to the player which would increase and decrease as the game progresses. If the selected horse wins, the player would get double the bet money back whereas they would lose the bet money if the horse loses the race. As the race completes, the players should be able to bet on a new race with the existing available funds.

iv. Further Advanced:

The final level of difficulty of the assignment asks to perform any of the few features enlisted. One of the features must be completed while additional features would make the game more interactive and complete. The features implemented in this portion include the ability to specify the number of laps for each race and some additional features.

1.4 Structure of the Report:

The report is structured to have different parts enlisted which document the solution in a descriptive manner. This section contains the information about each heading and their contents in brief. This can be used to find the required contents within the report.

- **Introduction:** The introduction section provides the introduction to the assignment, the solution and the report itself. It describes what the brief asks for in the solution.

- **Game Design:** The game design section is about how the game works as a whole to meet the brief. This section contains detailed information about the features of the game. Additionally, some character information and a user manual has been provided.

- **Program Design:** This section is about how the codes were written and organized. This explains how the code was reduced and reused with the help of language tools, also showing how the breakdown of the problem into smaller problems was done. Flowcharts have been added to express the logic behind program codes.

- **Testing:** This contains the information on what aspects of the game were tested and the verified results along with screenshots. The testing section also contains the information about how the tests were performed for different purposes, what aspects of the program could be tested without having to finish the race and bugs discovered during the test.

- **Evaluation:** The evaluation section contains the bugs and weaknesses of the solution, the strength of the solution, what works well, what improvements could have been made and what different approach would have been made if the author were to design a similar game.

- **Conclusion**: The conclusion section is the conclusion of the report, the solution and the Module Term-II.

2. GAME DESIGN:

2.1 Game Description:

The game is a horse racing betting game made using HTML, CSS and JavaScript. As the HTML and CSS were already provided, with slight modifications and additions to these files, JavaScript was written from scratch in order to make the game interactive and complete.

Figure 2.1 – The Game Design

As in the Figure 2.1, the game scenario is a horse race where four horses race along a track. Specifying the bet amount, the horse to bet on and the number of laps, and then finally clicking the Start Race button would start the race (Refer to the User Manual to know how to play the game). The four horses run towards the right corner of the track and when they reach their respective turning points (Figure 2.2), they turn upwards and continue to run. They repeat these steps as they reach towards each corners of the tracks, till they finally cross the finish line. If the set number of laps is more than one, the horses repeat this process until they complete the number of laps.

Figure 2.2 – Turning Points

2.1.1 Game Features:

In addition to the requirements, many other features have been added and implemented in the game. Some of these include input validation, sound effects, volume controls, number of laps, random speeds and winner, and some extra features. Each of the added features will be described in this section.

i. Horses run along the track:

When the race begins, the horses continue to run along the track before they finish the number of laps and cross the finish line. Correct CSS class names are given initially, and at each turning points so that the horses face the correct direction and correct animations are shown. The track **turning points** were determined using viewport width and viewport height as the pixel alternatives. The initial placements of the horses were taken from the CSS file. From here on, according to the width of the track and the initial horse positioning, this left position was increased at regular intervals to show that the horses are moving towards the right. Values of 'vw' and 'vh' (Viewport Width and Viewport Height) were tested to determine the turning point of the horses. According to the horse number, each horse would turn at a different point. The turning points are set so that the race would be fair and each horse would have almost same length to run.

Using vw and vh instead of px would allow a similar result across smaller screen devices as well. Although the result is not the same, and some devices might show the horses running off-track, the results are close to the expected one, i.e. the horses do not go far off the track which would have been the case using px. However, these results are the current limitations of the game.

ii. Random speeds and winners:

The race is setup so that the speeds for each horse are random and keep changing every second. For example: - If the white horse is leading the race, it could be overtaken by another horse at any point of the race. This other horse could then be overtaken by any of the horses as well. This continues to happen until the horses cross the finishing line, where the winner is declared. The leading horse is declared as the winner.

iii. Disabled Start Button:

The start button is disabled as the race begins – this meaning the background of the start button is greyed out and clicking the button during this period does not bring any change in the game. A race should complete before the button is enabled once again and this repeats every time a race starts.

iv. Betting:

The player gets an initial amount of £100 and can bet throughout the game until he has money left. It is set so that whenever the bet horse wins the race, the player gets

double the money back whereas the money is lost upon losing the race. The user cannot play after using up all the money and shall reload the game.

v. Number of Laps:

The horse races are lap configurable meaning that the player can define the number of laps they want the horses to run before the winner gets declared or the race is over.

vi. Input validation

The other one (Figure 2.3) is the validation for different values entered as bet amount and the number of laps. It would not be normal for a bet to have negative value, or a value greater than the available amount. Similarly, for the laps, there must be at least one lap which would otherwise not make sense in a race.

The rules set up for this include:

- A numeric value must be entered as the bet amount.
- The value for bet amount should be greater than 0 but not more than the total available amount.
- The lap number must be a numeric value more than 0.

The player is notified of this with the help of messages being displayed showing relevant error (Figure 2.3). These errors would also prevent the race from starting, so, the player first needs to correct these errors in order to begin playing.

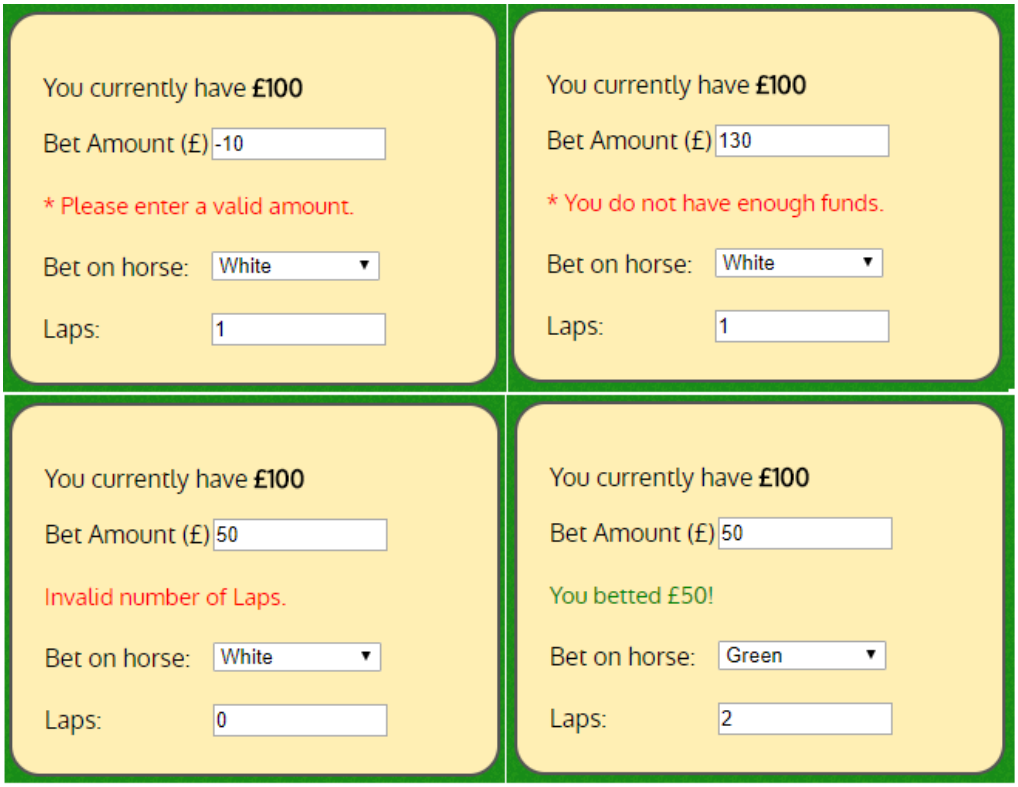

Figure 2.3 – Input Validation

vii. Start Game Menu:

A start game menu appears when the game is started (when the html file is opened in the browser). This prompts the player to click the Start Game button before making the game layout visible and the game playable. The menu was found necessary as some browser policies do not allow sound autoplay feature on page load which would not allow the background music in this game to play. Clicking the Start Game button to make the game playable and turning on the background music in this occasion, would however allow the background music to be played. In addition, this is also a trend across multiple games to have a Start Game menu.

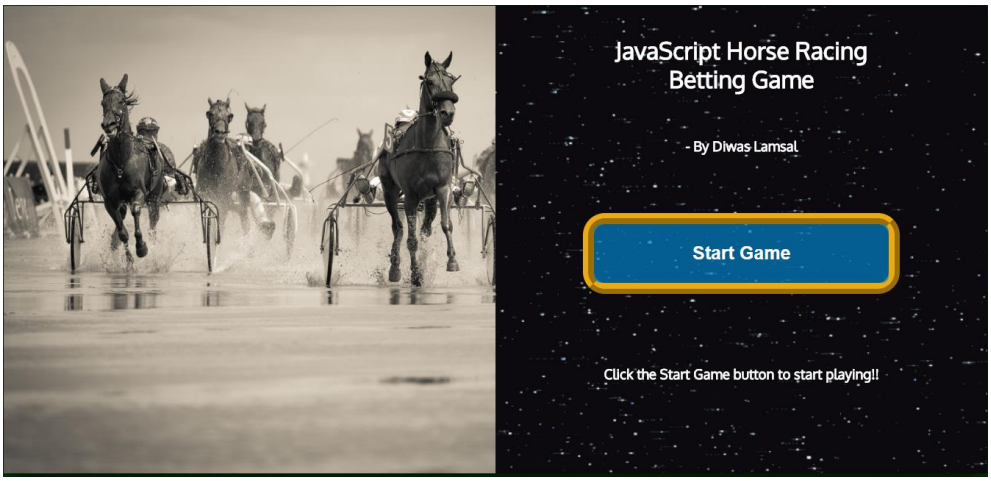

Figure 2.4 – Start game menu

viii. Disabled input areas when game begins:

Once the game starts, the number of laps, horse betted on and the sum entered for betting are not editable. This is done to make the game fair. Otherwise, being able to manipulate bet amounts during the race would allow the player to increase or decrease their bet amount looking at which racer is about to win the race. This can rather be considered as reducing the bugs of the game and improving the gameplay experience.

ix. Countdown timer for the race:

The race does not immediately begin after being triggered by the start race button, but when the countdown timer finishes to execute. With the click of the start race button, if the input validation goes right, a countdown timer appears in the middle of the screen. The horses start to run only after this countdown timer vanishes after counting from 3 to 1 and displaying the “Go!” message.

x. Sound effects for different events:

A ticking sound effect [Soundbible (2012)] is added to the countdown timer which plays every second during the countdown interval. Similarly, two different kinds of sound effects have been added which are played according to the condition of winning or losing the game. Winning the game would play a Ta da Sound [Soundbible

(2009)] which would resemble a win in video games whereas losing the game would trigger the sad trombone [Soundbible (2011)] sound effect.

xi. Background music along with audio control widget:

Clicking the Start Game button from the menu would start a background music for the game. The music is typically a kind of retro music suitable for retro games. The music is played at a default specified volume which can be changed through the provided audio control widget in the top right corner of the screen. The audio can have states of paused, played or muted and the volume levels can be raised or lowered using the plus or minus icons. Figure 2.5 shows different stages of the audio control widget. From left to right: - Paused, Playing, Muted, High Volume and Low Volume.

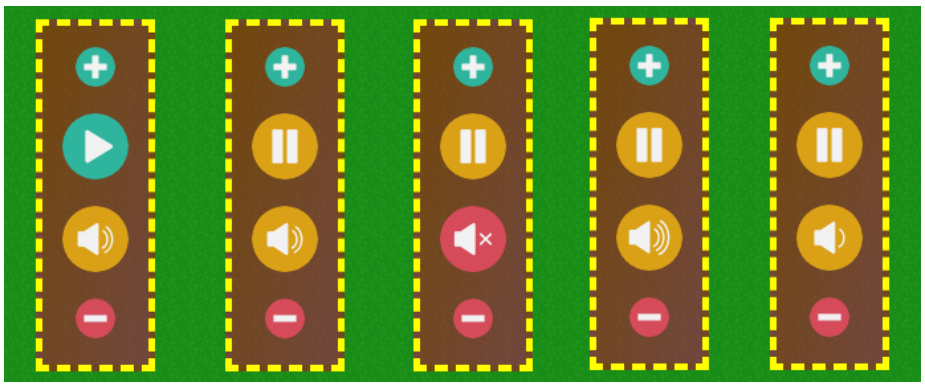

Figure 2.5 – Audio control widget

xii. TV Screen (Display Screen)

A display screen has been added inside the track area. This screen displays horse race based ads throughout the race and afterwards, but at the end of the race, if the player wins the bet, a winner GIF is displayed on the screen for six seconds. Similarly, on losing conditions, a loser GIF is displayed which would appear to be mocking the player for losing the game. The ads continue to display after the set six seconds timeout. (Figure 2.6 shows the different types of images shown on the screen)

Figure 2.6 – Display screen

2.1.1.1 Characters:

The game is designed for a single player who can bet on four different racers which are the characters or the participants in the horse race. Whenever the betted horse wins, the player gets double the money back whereas loses the money otherwise.

Figure 2.7 – Racers

2.1.2 User Manual:

The game is for a single player and they can follow these instructions to play the game:

- Clicking the Start Game button at the beginning of the game would skip the game menu and let the player continue into the game.
- The initial amount is £100. The available amount can be used to bet on a specific horse. Note: Winning the bet gives double the money back whereas the player loses the money otherwise. The player should set a valid amount to bet on any of the four horses.
- Number of laps (By default: 1) can be altered by the player before starting the race.
- Setting up everything and clicking on the Start Race button would start the race.
- Before the horses start to run, a countdown timer appears in the middle of the screen, when it vanishes, the horses begin to run. The user shall wait until the race is complete.
- After a race completes, the winners and amount are updated, and another race can be started with the available amount.
- To change the audio properties, the audio toolbar can be used which is available at the top right corner of the game window.

2.2 Game Design Summary:

All the listed features together make for a complete solution to meet the brief. Some extra features have been added to improvise the given scenario. While the assignment was being worked on, some or more portions of HTML and CSS were edited to add these extra features. The end result is an interactive horse racing betting game.

3. PROGRAM DESIGN:

3.1 Program Design Overview:

The code was designed to be efficient, less redundant and on point on what it does. The Object-Oriented Programming principles have been implemented which has helped reduce the code at least to one fourth. Using these objects, along with the usage of different kinds of loops, conditional statements, inbuilt functions and user-defined functions, the coding phase was completed.

3.2 Reduced Redundancy:

In any programming language, redundant or repeated codes are considered a bad practice. Unnecessary declarations and codes would occupy memory space, which might not be as noticeable in a smaller program, however, when chunks of memory spaces are used up like this in a larger program, it effects the program in a significant way by using up a lot of resources. Similarly, writing up the same lines of codes over and over again in a single program is not an ideal method of programming at a time where programming has evolved to allow using similar blocks of codes countless times with just a few lines of codes. Having codes divided according to their features and usage, the codes can be called or used anywhere instead of rewriting. This not only saves time and makes the coding efficient, but also makes debugging a lot easy.

Excessive usage of functions, loops and arrays have been done in order to meet the brief's requirements. Codes that together perform certain actions have been grouped inside functions and called whenever necessary or with the use of intervals and timeouts. Likewise, codes that need to be executed more than once have been written within loops. The Object-Oriented Programming principles have been used for creating four different objects as the racers. These racers perform different actions according to the functions available to them. This will be discussed in the **3.5 The Features of JavaScript and Alternative Approaches** section.

3.3 Different Sized Screens:

As mentioned earlier in the **2.1.1 Game Features - i. Horses run along the track** section**,** the game is designed to have a similar experience across different sized devices. The horses run and turn at the exact same point of the track across any devices. This has been made possible with the use of 'vw' and 'vh' to change their locations instead of 'px'. The 'vw' and 'vh' use viewport width and height to get the percentage values of the screen, which would make the horses turn exactly at the same point, no matter the size of screen they are running on.

3.4 Breaking up the Problems:

3.4.1 Racer Objects and their Methods:

Much of the problems have been dealt with using the OOP principles. Four racers exist in the solution which have different sets of data and methods, which define their behaviors. For example: - When the runUp function is called for racer1, the racer1 would run towards the upward direction (Figure 3.1). The racer objects have different sets of data and methods like the one mentioned, and these together, solve the various aspects of the task. This approach greatly breaks up the task into simpler and functional units, which not only reduces the code, but helps in making use of each of these features efficiently.

More about this in the **3.5 The Features of JavaScript and Alternative Approaches** section.

```
var racer1 = new racer('horse1', 30, 68, 1, laps);
var racer2 = new racer('horse2', 30, 72, 2, laps);
var racer3 = new racer('horse3', 30, 76, 3, laps);
var racer4 = new racer('horse4', 30, 80, 4, laps);
```

```
this.runUp = function(){
    var racers = this;
    clearInterval(racers.interval);
    racers.element.className = 'horse runUp';
    racers.speed = 10 + Math.ceil(Math.random()*5);
    racers.interval = setInterval(movingUp, racers.speed);

    clearInterval(racers.speedInterval);
    racers.speedInterval = setInterval(function(){
        clearInterval(racers.interval);
        racers.speed = 10 + Math.ceil(Math.random()*5);
        racers.interval = setInterval(movingUp, racers.speed);
    }, 1000);
        function movingUp(){
            racers.top -= 0.2;
            if(racers.top <= 3 + racers.number*2.8){
                racers.runLeft();
            }
            racers.element.style.top = racers.top + 'vh';

        }
}
```

Figure 3.1 – Run Method Example

3.4.2 Problems Broken Down and Flowcharts

i. The Race

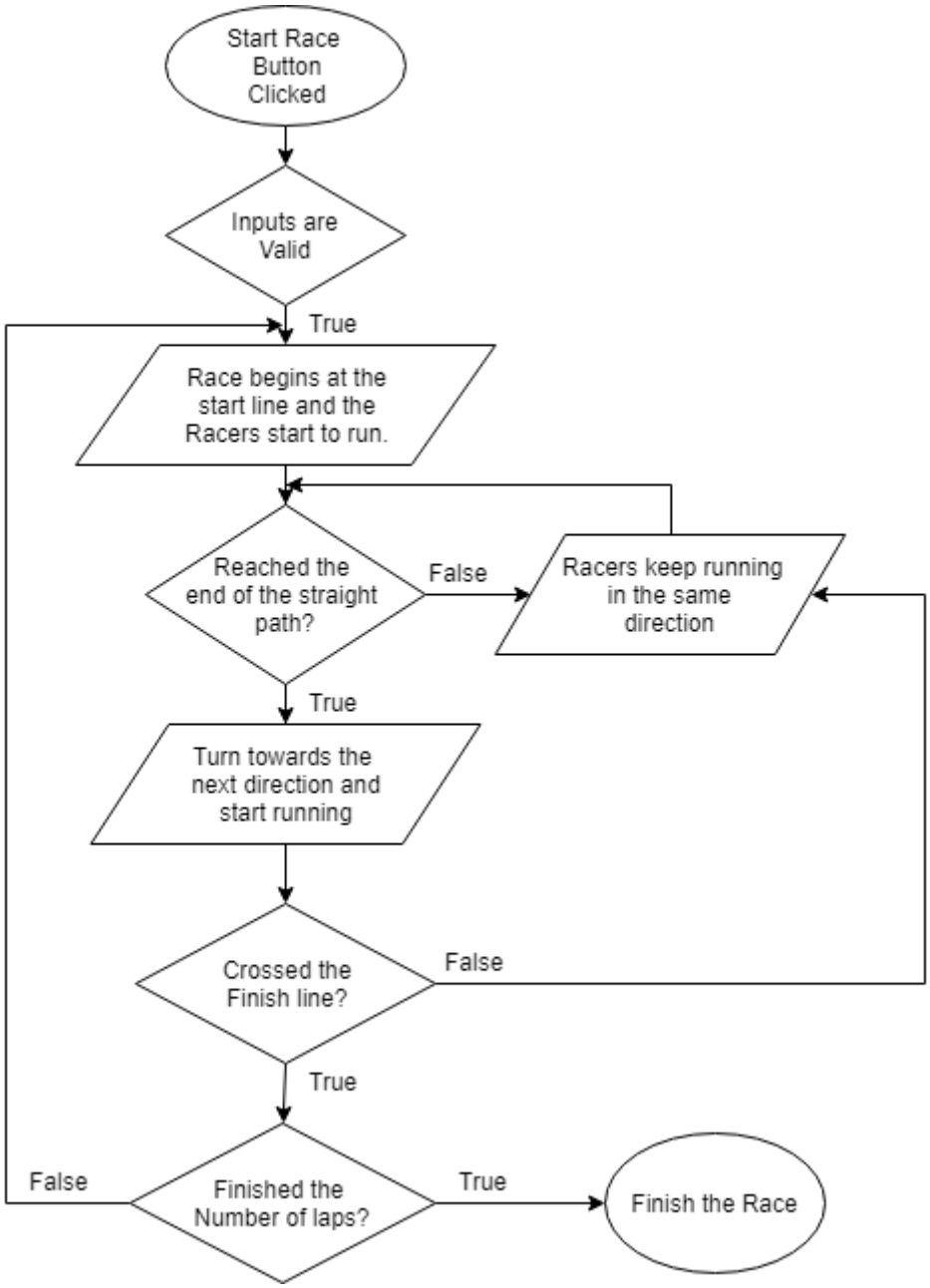

Figure 3.2 – The Race Flowchart

The race itself follows a simplified logic where the racers begin to run once the Start Race button is clicked. Along the path, if they reach the end of the line, the racers turn and start running. When they cross the finish line, the number of laps is checked and if the set number of laps is completed, the race ends.

ii. The Run Method

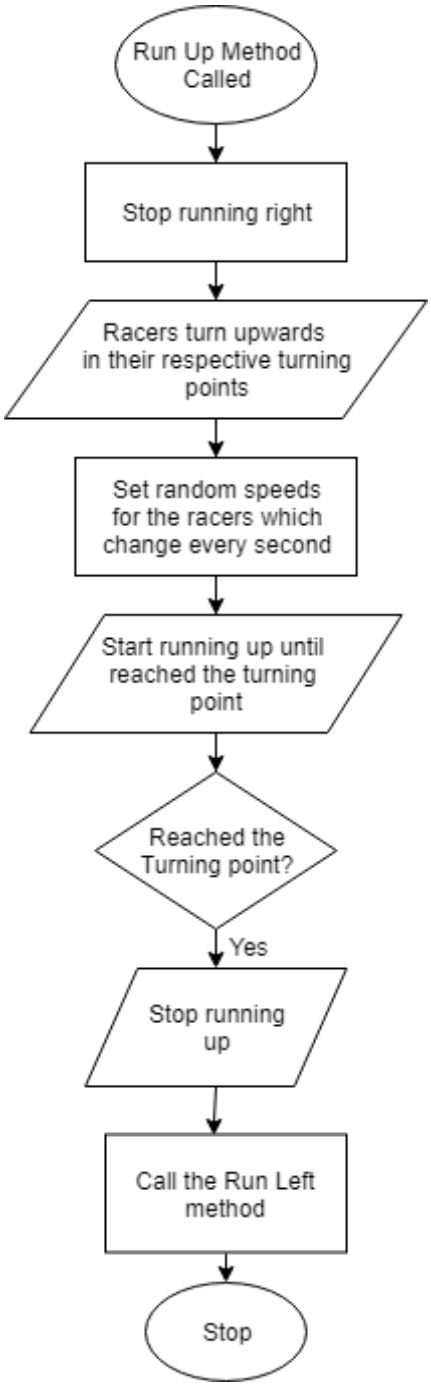

Figure 3.3 – The Run Method Flowchart

The run methods are the functions called during the race to make the racers run towards the respective directions and detect when to change the direction. These methods keep changing the speeds of the racers every second. Whenever the racer reaches their turning point, another method which makes them run towards a different direction is called, and the horses start running towards that direction.

iii. Start Race Button and the inputs

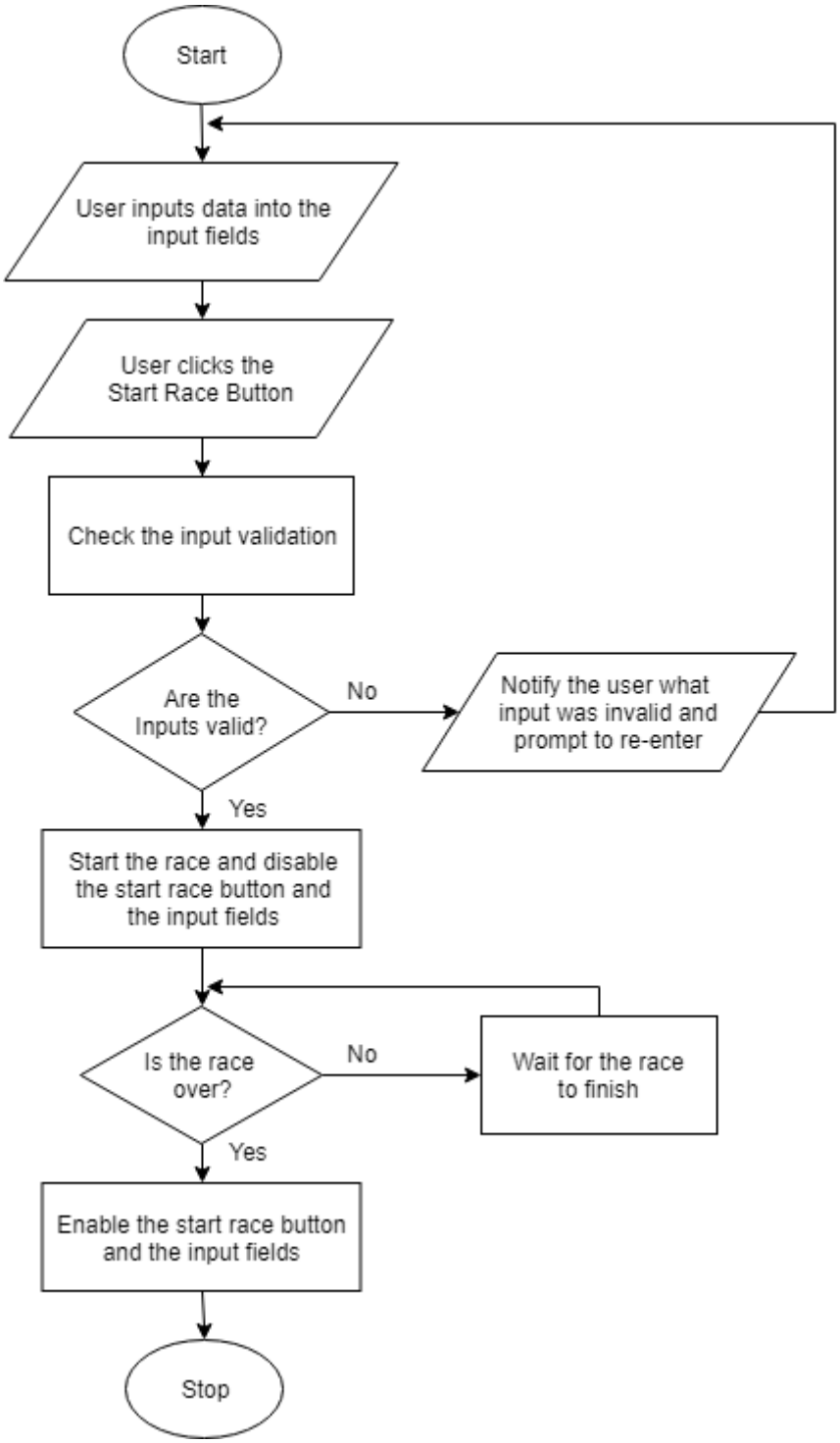

Figure 3.4 – Start Race Button and Inputs Flowchart

The start race button is disabled as the race goes on and is enabled once again as the race is completed. The logic mainly functions to start the race after the betting and lap inputs are validated and the start race button is clicked.

iv. Winner Selection

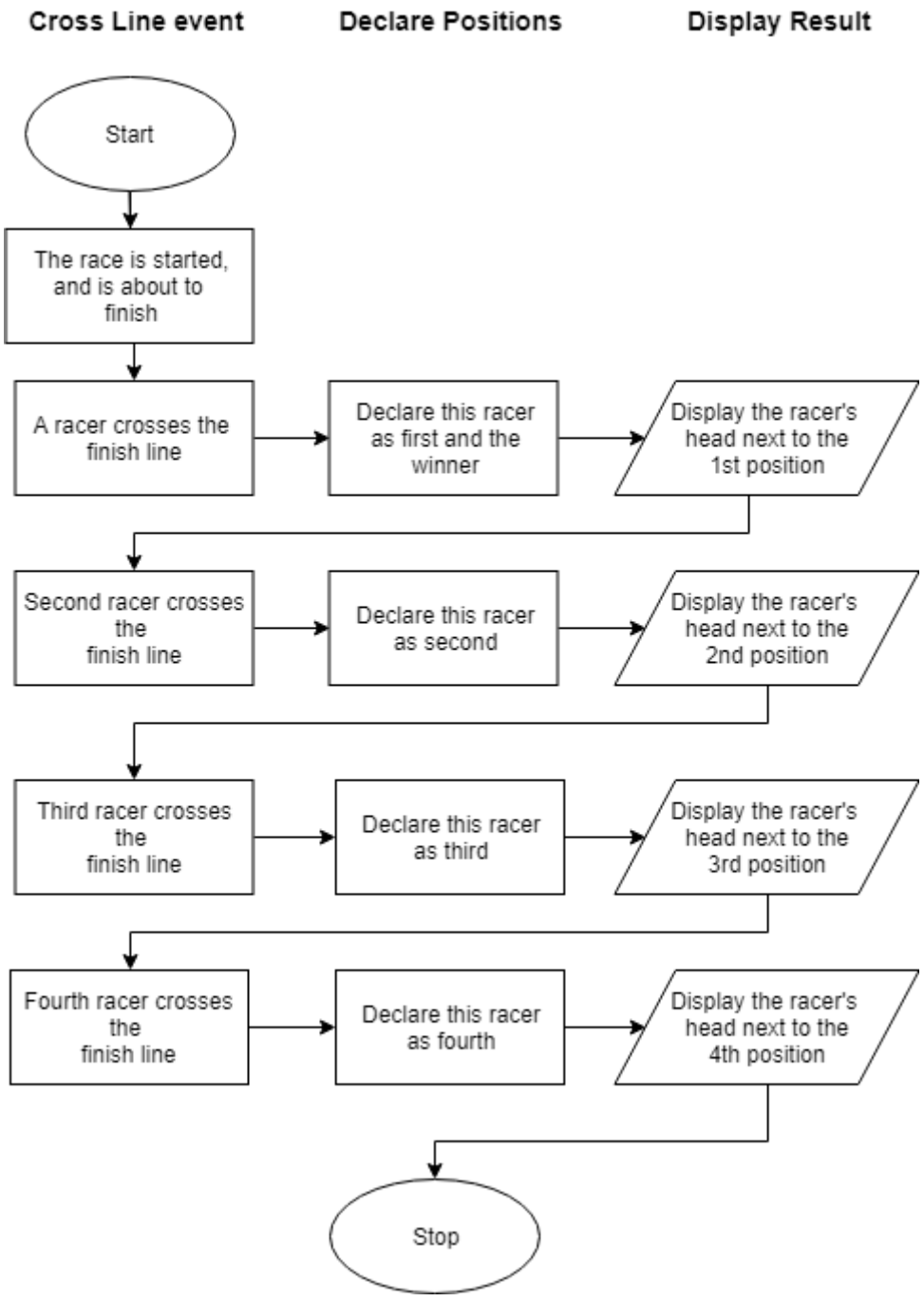

Figure 3.5 – The Winner Selection Flowchart

The winner selection logic just follows the basic principles of a race. Whoever crosses the finish line first is declared as the winner and displayed next to the first position. Similarly, the following racers are given their respective positions and displayed in the table for the user.

v. Betting, sound effects and the GIF screen

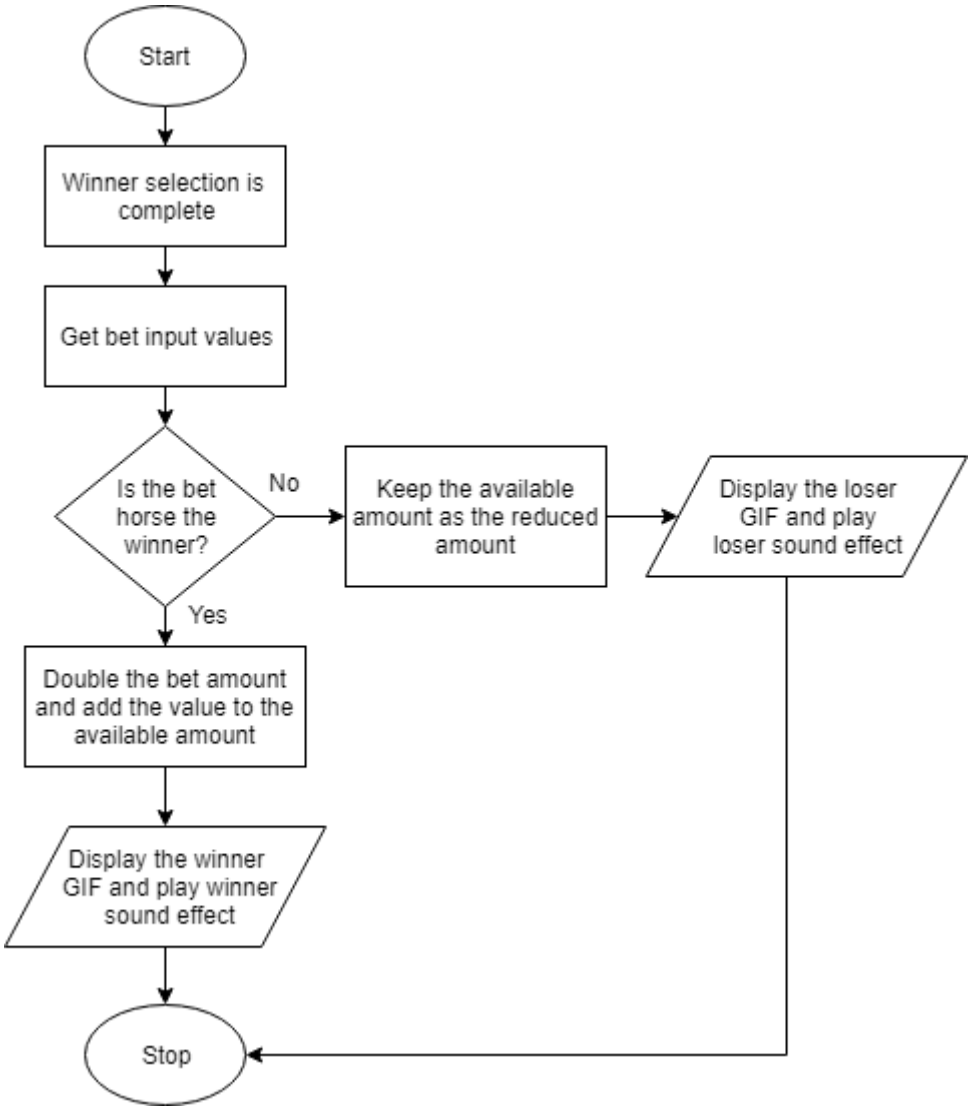

Figure 3.6 – Betting, sound effects and the GIF screen Flowchart

3.5 The Features of JavaScript and Alternative Approaches

JavaScript provides a lot of features such as arrays, loops, functions, conditional statements and creation of objects. All of these have been utilized - some to more extent and some less. The major feature used could refer to as the DOM (Document Object Model) manipulation itself. Changing class names, editing CSS, inner HTML contents, all off these have been utilized to come up with the solution.

Working on the solution, some alternative approaches were also looked at, which was not included in this term's syllabus. This approach mainly being the OOP concept. Figure 3.7 shows the creation of objects and the class used for the objects.

```
function startRace(){
    var laps = parseInt(document.getElementById('lapNumber').value);

    var racer1 = new racer('horse1', 30, 68, 1, laps);
    var racer2 = new racer('horse2', 30, 72, 2, laps);
    var racer3 = new racer('horse3', 30, 76, 3, laps);
    var racer4 = new racer('horse4', 30, 80, 4, laps);
```

```
// The racer class which creates racer objects
    function racer(horseId, left, top, number, laps){

// The data for the racers objects.

        this.element = document.getElementById(horseId);
        this.left = left;
        this.top = top;
        this.number = number;
        this.interval = 0;
        this.positions;
        this.laps = laps;
        this.speedInterval = 0;
        horseBetOn = document.getElementById('bethorse').value;
// Different functions for the racers objects. These define the direction and spe
```

Figure 3.7 – Usage of OOP Features

The four racer objects are the four different horses and racers in the game. The racer objects have different functions and variables defined which make them a complete game character for this game. From making the racers run to deciding the winner and making the bet decisions, everything is based on these principles. Instead of running through loops and defining everything individually for different horses, using this approach saves a lot of time and is the correct way to deal with problems such as this.

4. TESTING:

In every programming project, whether documented or not, testing is a crucial step to find out if the written program works. Various tests have been performed throughout the writing stage before finally reaching the end of the solution. Even so, further testing was required to find out what works well and what does not. During the code writing phase, the tests mainly included the direct effects on the browser across different events and browser console results. For example: - If a block of code was not running or functioning correctly, a console message would be added in the same block of code to find out where the debugging was required. Also, the variable values could be displayed in the console to know how they are being changed across multiple events. Besides these, the console also displays the errors generated during the runtime.

Some aspects of the game could be tested without having to run and finish the entire game. These include:

- The turning points in the tracks
- The speeds of the horses
- The CSS class names and respective animations
- The start race button and the aspects of the game it triggers
- The TV Screen
- The countdown timer
- Input Validation
- The background music and the audio widgets

The other simple way of testing the written code would be running the game in the browser and trying out the different features added. For example: - the plus button on the audio widget should increase the volume. This aspect can be tested by clicking the button and verifying the result. All of such tests can be included in a tabular form where the expected and actual outcomes are presented along with the tests performed.

Test Plan:

	Test	Expected Result	Actual Result
1.	Entering a bet amount of 50 and clicking the start race button (Test for horses running)	The horses start running along the track and turn at set turning points for each of them	Same as the expected result. Figure 4.1.
2.	Checking in multiple races if the winners are same and if the horses run with same speeds. Two races to find out whether the results are random.	The winners might be different across different races but every horse has equal chance. The speeds of horses are random and change every second.	First run: White horse is the winner, horses run at random speeds which frequently changes. Figure 4.2. Second run: Blue horse is the winner, horses run at random speeds which frequently changes. Figure 4.3.
3.	Start race button after the race is started.	The start race button should be disabled - greyed out and shall have no effect during this period. The button should be enabled after the race finishes.	Same as expected result. The button is greyed out and has no effect when clicked during the race. The button is enabled after the race finishes. Figure 4.4.

4.	Betting 50 and verifying the result on winning and losing cases	The amount should be doubled and updated on winning condition whereas should be reduced by the bet amount on losing condition.	Same as expected result. Figure 4.5.
5.	Setting number of laps to 4	The horses should run across the track four times before the race finishes. The laps counter at the bottom right of the screen should decrease as each lap is completed.	Same as expected result. The lap screenshots were taken after every two laps. Figure 4.7.
6.	Clicking the start race button by entering a negative value as bet amount.	The race should not start and a red colored text message should appear inside the input box informing that the input bet amount is invalid.	Same as expected result. Figure 4.6.
7.	Clicking the start race button by entering zero as lap number.	The race should not start and a red colored text message should appear inside the input box informing that the input lap number is invalid.	Same as expected result. Figure 4.6.
8.	Clicking the start race button by entering higher amount than the available amount.	The race should not start and a red colored text message should appear inside the input box informing that the input bet amount is higher than the actual amount.	Same as expected result. Figure 4.6.
9.	Clicking the start race button by entering a value within the available amount range and setting the lap number to a positive integer.	The race should start and a green colored text should appear inside the input box informing how much amount was bet.	Same as expected result. Figure 4.6.
10.	Clicking the start race button without adding bet amount	Input validation prevents the race from starting and the user is prompted to enter bet amount.	Same as expected result.

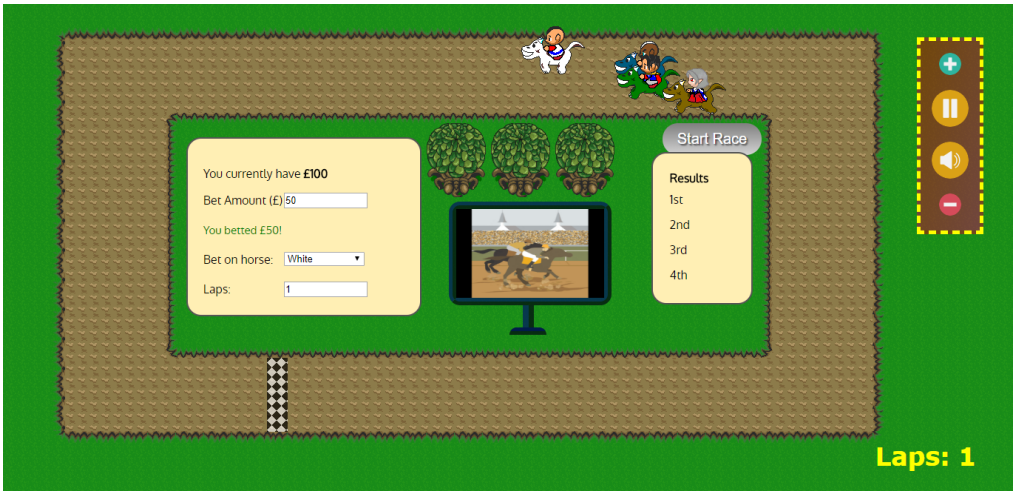

Figure 4.1 – Horses Running

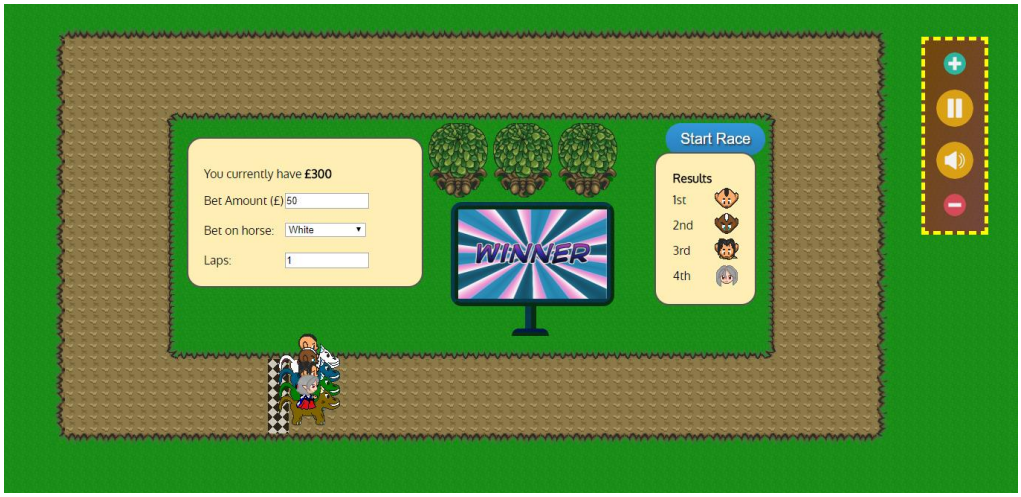

Figure 4.2 – Random Winner First Run

Figure 4.3 – Random Winner Second Run

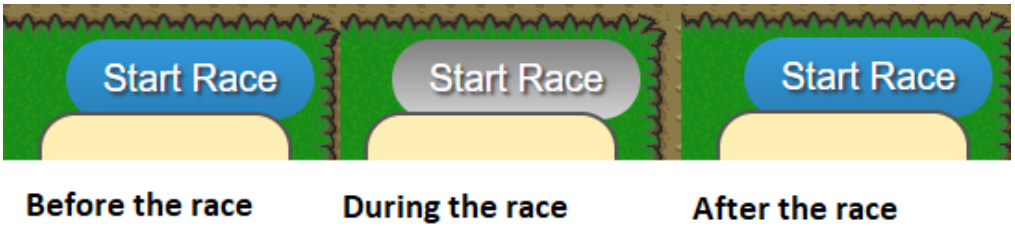

Figure 4.4 – Start Race Button Different Stages

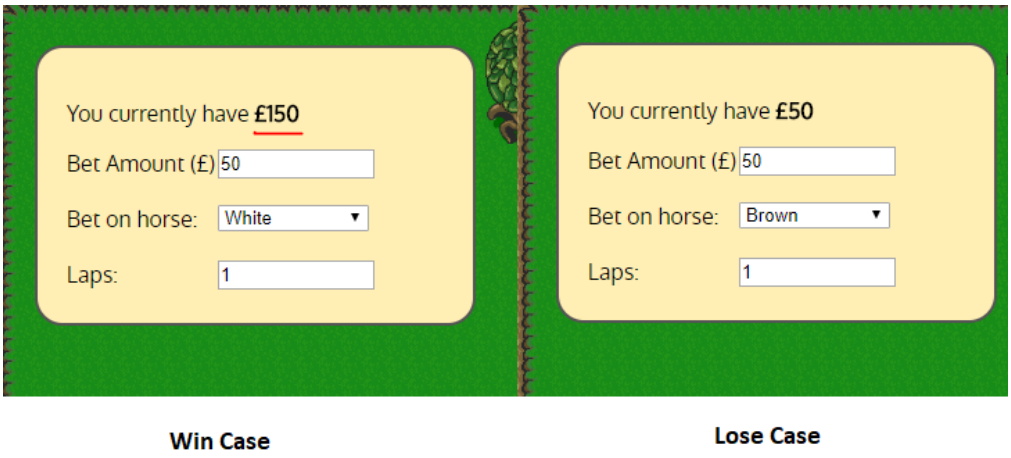

Figure 4.5 – Betting win and lose cases

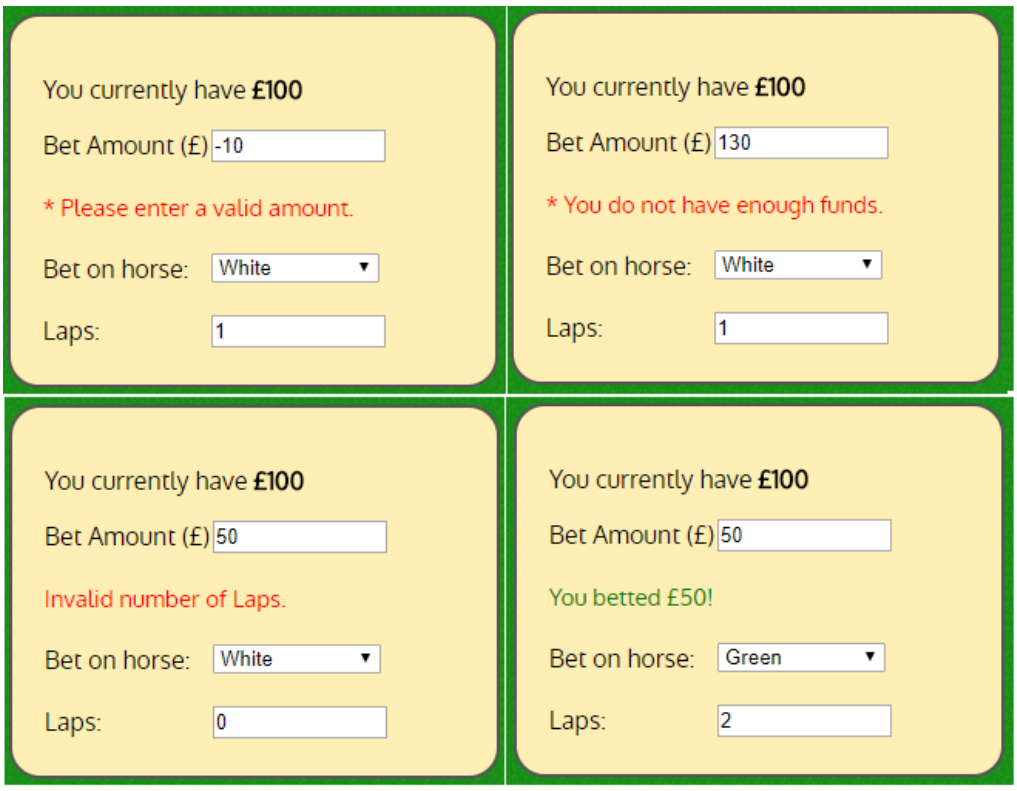

Figure 4.6 – Input Validation

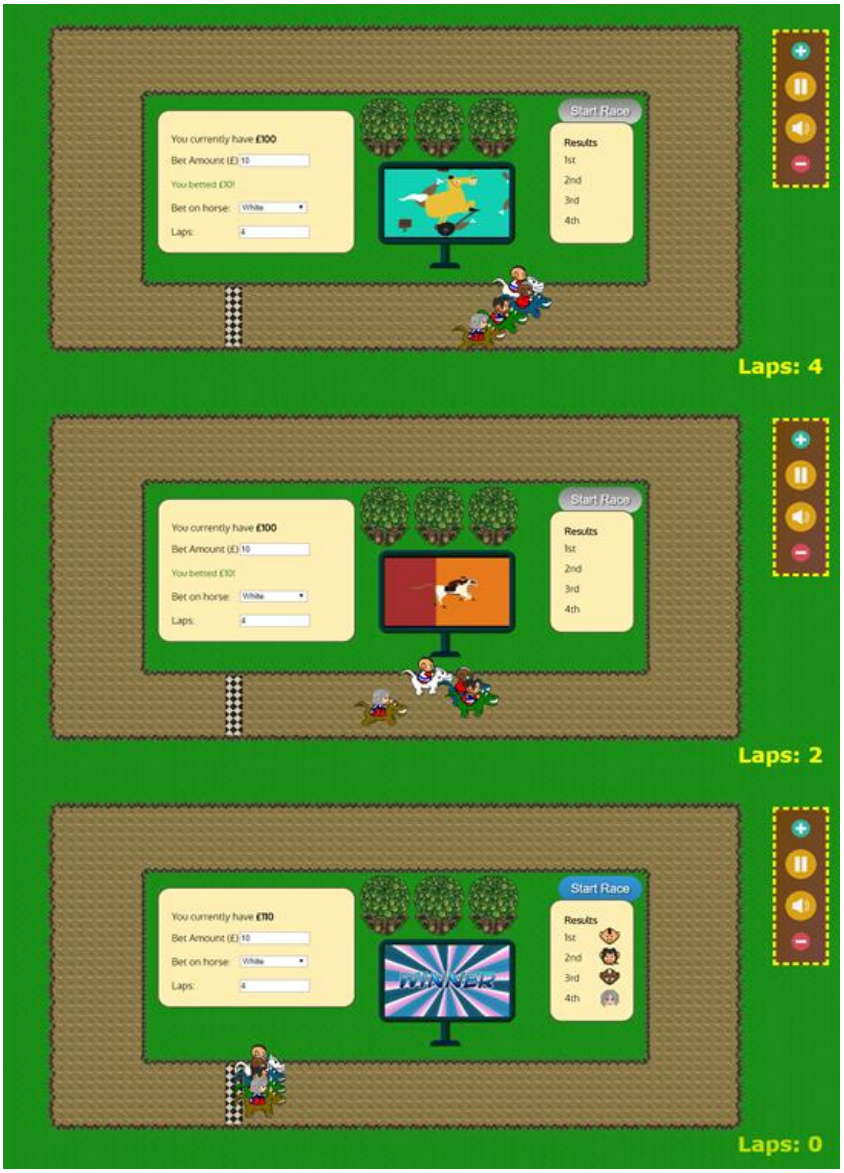

Figure 4.7 – Number of laps

Figure 4.8 – Reduced Track Size

Notable Bugs:

Although the code was designed to have the same turning points across any device sizes, and while this is working to a certain extent, the results were not fully as expected as some of the racers (mainly being the white and brown horses) sometimes go off the track when they turn (Figure 4.8). This is the only bug in the game among the included features and the result that was not as expected.

5. EVALUATION:

With the usage of most features within the language tools, different sources as study materials and basic knowledge about games through experience, the solution was completed mixing up these combinations. The game has turned up to be an interactive horse racing betting game for a single player which does have some or more weaknesses and room for improvements. A single bug exists within the game which effects the gameplay partially and could be solved if more time was available.

5.1 Bugs and Weaknesses:

The Bug that exists:

As mentioned in the **5. TESTING – Notable Bugs** section, the game has a bug (Figure 4.8). The racers appear to be moving out of track when they turn. This happens only on smaller screen devices. The bug could be fixed by finding the correct value in vw and vh as the turning points for each horse.

Some weaknesses in the game:

- Whenever the race is started for a second time, there is a slight delay for when the horses run after the countdown timer fades.
- The lap number decrement is not done exactly at the same point but set by giving it a timeout. The result is the lap number display shows the reduced number of laps upon a lap completion when all the racers cross the start line, when one of them crosses the start line or when none or a few of them cross the start line. This being random and not static.
- In some cases, more arrays and loops could have been used which would have reduced the code repetition by a greater margin.
- Extensive usage of if…else statements which could have been replaced by using switch case.

5.2 Strengths of the game and what works well:

The strengths and features of the game have already been mentioned in the **4. GAME DESIGN** section in detail. The features that were described include:

- The horses run around the track and appear to be running in the set direction with the help of correct CSS class names. These horses do not turn exactly at the same point.

- The horses run at different speeds which also change every second. The winner is not always a single horse and could be any one out of the four.
- The start race button is disabled once the race starts and is only enabled after the end of the race (After all the horses cross the finish line after completion of all the laps).
- Betting has been enabled and the player can bet on any of the four horses with the available amount.
- The available amount after the first round of bet can be used to play the game again.
- Number of laps can be defined before the race starts. The horses complete the number of laps before the race ends.
- The bet amount and lap inputs are validated before the game begins. The amount needs to be a positive number and cannot be higher than the available amount. The lap needs to be a positive integer. The validation errors are shown upon invalid inputs and the race cannot start without correcting these.
- A start game menu appears upon opening the game in the browser. The Start Game button should be clicked before the game's layout is visible and being able to play the game.
- When the game starts, it is set so that the input fields are disabled and cannot be edited. These are enabled once again after the completion of the race.
- A countdown timer appears before the race begins. The racers start to run when the timer fades.
- Sound effects are added across multiple events. The ticking sound for countdown timer, winning sound for winning the bet and losing sound for losing.
- A background music starts to play once the Start Game button is clicked on the Start Game menu. The audio properties such as volume and play/pause states can be configured using the provided audio widget.
- An ad-display screen is added to the middle of the game screen which displays ads and winner/loser based GIFs upon winning or losing the game.

5.3 Improvements or Extra Features:

The game could have some improvements or extra features to make it more enjoyable and interactive with the availability of more time. Some of these could be:

- An eight-shaped track – The game could be edited to have an eight-shape track where the horses run at a certain pattern.
- A timer could be added to get the total time taken for each horse, which could also be used to determine the winner. This could even be used to get the highest score across multiple races.
- Audiences could be added from the provided images and the CSS files from the topic-6 game content. This was thought of but not actually implemented in the solution.

- Traffic lights or some kinds of roadblocks would make the race appear more interesting. These would display green light or let the horses pass as they reach near them.
- The game could be turned to a multiplayer game where not just one player, but many different players could place bets on different or same horses. The winner could be selected after certain number of races to be the one with the highest amount in their balance.

5.4 Adding a Fifth Horse:

As this solution follows the OOP principles, adding a fifth horse would not make much difference, as, the only tasks that would be added are creating a new racer object and editing some of the values that determine the end of the race. Hence, provided the CSS and HTML, it would not be a tough task to add a fifth horse to the race.

5.5 Building a Similar Game:

If the author had to build a similar kind of game, he would have done things differently. As the author would prefer allowing the player to have more interaction in the game, he would allow the player to control one of the horses. Instead of the betting system in this solution, the game would be a horse controlling racing game. The other three horses would be bots and move at set random speeds while the player controls one of the horses he chooses and plays the race. The speeds of bots would increase with increase in levels and the path pattern would also be changed. Although the difficulty of coming up with such a solution would be tougher, it would provide better experience to the player and would be a more challenging game.

6. CONCLUSION:

As JavaScript is a programming language, and an Object-Oriented Programming language at that (which follows prototype based object principles), it provides a lot of features that could not be possible with plain HTML and CSS. The manipulation of DOM objects is the main usage of JavaScript. Without reloading the page, different aspects of the page can be changed by using JavaScript. Not only the vanilla JS – the pure JavaScript we use, but some popular JavaScript libraries such as jQuery can be used which makes writing JavaScript easier and these kinds of libraries also have some features that could not have been possible using only the vanilla JS. Some other popular JavaScript libraries include react, angular, vue, etc. The second term of the CSY 1018 module has taught about the most important concepts and features of JavaScript. This can be used as a base to improve efficiency in writing more JavaScript and being able to solve more complex tasks. This knowledge can also be used to learn different available libraries as well as making it a pathway to learning PHP – a server-side programming language.

Appendix 1: REFERENCES

File sources:

Play Button Flaticon

Flaticon (2018), Play free icon. Available from: https://www.flaticon.com/free-icon/play-button_189638 [Accessed: 05/31/2018]

Pause Button Flaticon

Flaticon (2018), Pause free icon. Available from: https://www.flaticon.com/free-icon/pause_189639 [Accessed: 05/31/2018]

Minus Button Flaticon

Flaticon (2018), Remove free icon. Available from: https://www.flaticon.com/free-icon/remove_189690 [Accessed: 05/31/2018]

Plus Button Flaticon

Flaticon (2018), Add free icon. Available from: https://www.flaticon.com/free-icon/add_189689 [Accessed: 05/31/2018]

Low volume Flaticon

Flaticon (2018), Speaker free icon. Available from: https://www.flaticon.com/free-icon/speaker_189652 [Accessed: 06/02/2018]

Mid-level volume icon Flaticon

Flaticon (2018), Speaker free icon. Available from: https://www.flaticon.com/free-icon/speaker_189651 [Accessed: 06/02/2018]

High-level volume icon Flaticon

Flaticon (2018), Speaker free icon. Available from: https://www.flaticon.com/free-icon/speaker_189650 [Accessed: 06/02/2018]

Mute icon Flaticon

Flaticon (2018), Mute free icon. Available from: https://www.flaticon.com/free-icon/mute_189653 [Accessed: 06/02/2018]

SoundBible Tick Sound

Soundbible (2012), Tick Sound. Tick sound of a tick-tock clean and quick tick-DeepFrozenApps. Available from: http://soundbible.com/2044-Tick.html [Accessed: 06/01/2018]

SoundBible Winning Sound

Soundbible (2009), Ta Da Sound. Ta Da - Mike Koenig. Available from: http://soundbible.com/1003-Ta-Da.html [Accessed: 06/02/2018]

SoundBible Sad Trombone Sound

Soundbible (2011), Sad Trombone Sound. Sad Trombone - Joe Lamb. Available from: http://soundbible.com/1830-Sad-Trombone.html [Accessed: 06/02/2018]

All the GIFs were added from GIPHY - https://giphy.com/

Pexels Horse Picture

Pexels (2018), Horse Grayscale Photo - Martin Damboldt. Available from: https://www.pexels.com/photo/grayscale-photo-of-group-of-horse-with-carriage-running-on-body-of-water-802861/ [Accessed: 06/03/2018]

Study Resources:

W3Schools JavaScript Tutorials

https://www.w3schools.com/js/default.asp

Mozilla – Web technology for Developers - JavaScript

https://developer.mozilla.org/bm/docs/Web/JavaScript

JavaScript.info – The Modern JavaScript Tutorial

https://javascript.info/