University of Northampton

CSY 1018
Assignment 2: Web Development

Horse Racing

A BETTING GAME

Kaustuv Prajapati | UN ID: 17421493 | Date 30-JUL-2017

Summary

This report provides an analysis and development of a gaming web page/ website. The website is based on core HTML and CSS codes for its graphics, layout and animations. It is a Horse Race Betting game whose functionality is coded using JavaScript (a front-end web Development programming language). Coding is specific toward procedural programming rather than object oriented style. The report focuses on game design, program design and testing of the game.

Table of Content

Table of Contents

1. Introduction

Games' HTML and CSS codes were provided to us. In-order to write a JavaScript code I revised those HTML & CSS codes to develop an idea on implementing the best and efficient Jscript program on the game. As developing some initial codes and conducting requirement analysis based on the aims and objectives of the assignment I ought to change some Html and CSS codes to optimize the graphics and UI. The final code of this project is in appendix document.

2. Game Design

The code is managed using functions, each small task is compiled into functions and those functions are called on meeting certain condition on different part of the code where needed. For example, moving horse in the Right Direction is a task, thus a function is created as in the figure 2.0.1. This function takes some parameter and operate on them, which moves the required horse to move in the right direction.

The advantage of doing so is that this chunk of code will be used to move all the horse in the right direction on when required. This function is called at a number of places using different conditions and passing different parameters, thus code redundancy is removed.

Similarly, there are functions that moves the horse to left direction, up direction and down direction.

Some of the complex and challenging requirements are below, descriptions on how the solution works and what logic are used to do so.

2.1 TURNING HORSES AT THE RELEVANT POINTS WITHIN THE TRACK

Running horse on fixed direction could achieve using number of functions as in Figure 2.0.1. and calling different function under different conditions. To turn horses at the relevant points on the track, we will need to know the position of horse. In my case I've create number of variables that stores the width amount of track with respect to viewport (game window screen) as in the figure 2.1.1.

```
// variables that stores values related with current width and height of screen
// use in movement and turing of horses.
// distance from top and left side of screen to inner part of track
var leftLaneInner = window.innerWidth * (10/100); // stores 10% of total screen width ir
var topLaneInner = window.innerHeight * (15/100); // 15% of total height, indicates dist
var rightLaneInner = window.innerWidth * (70/100); // stores 70% of total screen width,
var bottomLaneInner = window.innerHeight * (65/100);

// distance from top and left side of screen to outer part of track.
var leftLaneOuter = window.innerWidth * (5/100); // stores 5% of total screen width, inc
var topLaneOuter = window.innerHeight * (5/100);
var rightLaneOuter = window.innerWidth * (80/100);
var bottomLaneOuter = window.innerHeight * (80/100);
```

Figure 2.1.1 width variables

Then to know if the horse has reached the turning point of the track; horse position relative to viewport is compare with the above width variables. When the condition is true then the horse movement direction is changed as in the figure 2.1.1. in this figure, you can see that for different condition the horse movement function (moveLeft, moveDown, etc) is different, this cause the turn. But the important part is all those IF-condition statements which detects the horse position and compare it to so that the

```
449
450     if((positionTop >= bottomLaneInner) && (positionLeft >= leftLaneOuter)){
451         runLeft(1);
452         moveLeft(1, rand(6));
453     }
454
455     if((positionLeft-100 >= rightLaneInner) && (positionTop <= bottomLaneOuter)){
456         runDown(1);
457         moveDown(1, rand(7));
458
459     } if((positionLeft <= leftLaneInner) && (positionTop >= topLaneOuter)){
460         runUp(1);
461         moveUp(1, rand(7));
462     }
463 }
```

Figure 2.1.2 condition checking and turning

direction function could be change under certain condition, thus that certain condition is a turning point on the track.

For different horse, the condition checking is different so that all horse won't turn at same point, which makes the game more realistic. In this way, all the horses are turned at relevant point. Testing for this code is at Topic Testing (coming on this report).

2.1.1 what happen at turning point?

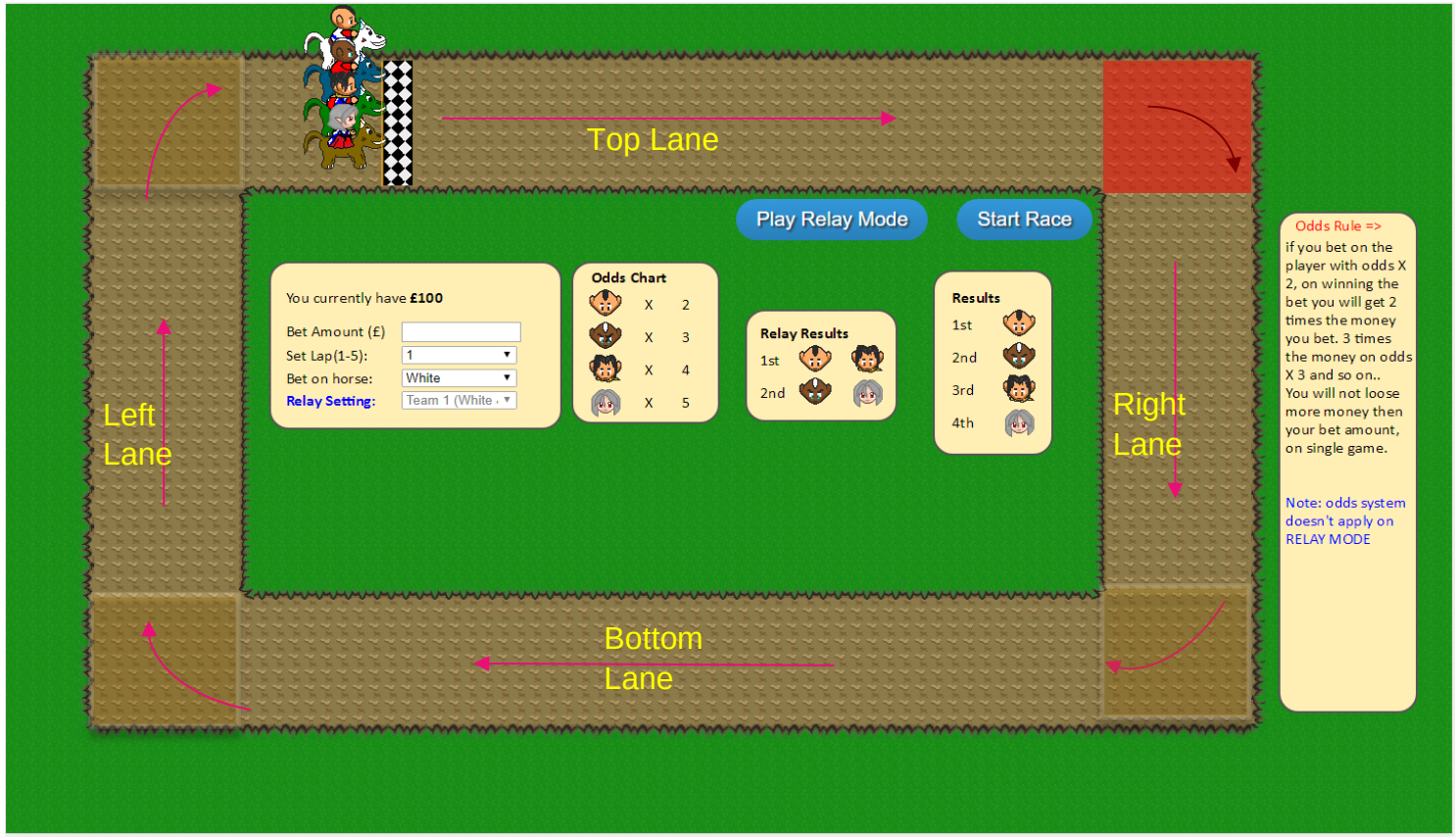

Figure 2.1.1.1 Track View

In game, the condition is applied such that the horses move to right direction when they are at top lane; they move down direction at Right Lane, left at bottom lane and up at Left lane.

But there are point where two different lane intersect, i.e. the Four corner of the track, shaded on the figure 2.1.1.1.

Let's say horses are moving on Top lane on Right direction and reached to the turning point on Right Lane (the red shaded region), in this case condition of being at **Top Lane** and **Right Lane** both are satisfied; thus, the horse will move Right and Down direction simultaneously, which leads to diagonal movement. Eventually the horse will be moving away from top lane and only satisfies the condition of Right Lane which makes the horse to move downward only. Same mechanism is working at all the turning points.

2.2 DIFFERENT HORSE WON EACH TIME

It is not like that different horse will won the race each time. The winning of horse is unpredictable, it's just random. And to do this the random number concept is used. In my movement function, some other functions are called within, to be specific it's the function rand() as in figure 2.2.1. rand() is a custom function that can be found in

```
375     */
376 function moveLeft(i,s){     /* make horses move LEFT */
377     var random = (rand(4) + (rand(0.1) * s));   // takes random number from c
378     var horses = document.getElementById(horse[i]); // selects specific horse
379     var positionLeft = horses.offsetLeft;
380     horses.style.left = positionLeft - random + 'px';
381 }
382
383 function moveUp(i,s){   /* make horses move UP */
384     var random = (rand(4) + (rand(0.1) * s));
385     var horses = document.getElementById(horse[i]);
386     var positionTop = horses.offsetTop;
387     horses.style.top = positionTop - random + 'px';
388 }
389
390 function moveRight(i,s){    /* make horses move RIGHT */
391     var random = (rand(4) + (rand(0.1) * s));
392     var horses = document.getElementById(horse[i]);
393     var positionLeft = horses.offsetLeft;
394
```

Figure 2.2.1 movement with random Value.

source code, which returns random value from 0 to the passed parameter value(argument). (*There are other custom function for random number like. 1.randRange(int var, int var) -> function that takes 2 parameters and returns random number that are between those range. 2.randFloor(int var), randCeil(int var); return random number, detail info on Code comment, all these function are used somewhere in code*).

2.2.1 How this random number works and determines the winner?

Whenever the movement function such as moveLeft() is called, the new random number is generated with rand() using parameters and other variables. This random number is stored on another variable named random. The statement on line 380 of Figure 2.2.1, is the main code to move the horse position; as the **random value** is use in that statement the horse movement will be random, this cause change in relative motion of horses. For each horse on each lane/direction, the random value is used and managed in such a way that the winner is unpredictable.

2.3 GRAPHIC IMPROVEMENT

Z-index is the main concern of this topic. The z-index value of horses <DIV> are fixed which means that the horse with high z-index value will always appear above the one with low z-index value, this setup disturbs the graphics of game, so I've set new function (*horseIndexing() function*) that checks the position of each horses and manages the z-index value as required, this improves graphical content of the game.

2.4 MAKING GAME INTERESTING

There are lot of factors that makes game interesting; behind the scene a code daemon does the magic and make it possible. In this topic, we will view some interesting facts of game and the logic that works behind.

2.4.1 Horse overtaking – Realistic Racing.

We have already discussed about random movements of horses' due to random numbers used to displace the horse in the game on topic **2.2 Different horse won each time.** Here those same random numbers and random movement cause horse to overtake one another. Also, the random number range varies for each horse on different lane, so it is possible that a horse leading on Top Lane may be left behind on other Lane, therefore making it unpredictable to know who will win by observing only starting of the game.

2.4.2 Odds system

Odds system is the feature of the game that is requirement of the project. As per requirement the initial odds value for any character is given randomly; here in my game there are some changes, the odd values are randomly distributed to all character but the number are fixed between (2-5). No horse can have more odds value than 5 and less than 2 at any part of the game. When the first lap/round ends the odds value arranges in descending order with respect to the winning rank of the character, i.e. if the certain character wins the game the odds value for that character on next round will be minimum (i.e. 2) and whoever comes last, its odds value will be maximum (i.e. 5), for this a bubble sorting technique is used. You can view the logic on source code in appendix document. (function name oddsManager()).

Figure 2.4.2.1 Odds chart

In game, the Odds Chart provides the current odds value of each character. *Fig 2.4.2.1*

Mechanism of Odds value

The odds value mechanism is same as per requirement. If we bet on horse with odds x 2, then on winning the bet we will receive 2 times the amount we have bet, similarly betting on horse with x 5 and winning the bet will earn us 5 times the bet amount.

There are no extra loss on single round more than bet amount.

Point to be Noted: odds system doesn't work on **Relay Mode** of game. (Relay Mode coming later this report).

2.4.3 Lap system

Lap system is a special function of the game. The game lets the user/player select the lap number range between 1 – 5, and conduct game in corresponding with what user have provided.

2.4.3.1 How is the lap completion detected?

Can you see the orange vertical line near the horse Head?

That line is a custom made which acts as a finish line of the race. Yes. It was possible to use the actual finish line provide in the game, but that finish line was positioned using margin-left in the CSS code provided, due to it the *offsetLeft* value of that finish line can't be used to check the horse position. Instead of changing original css code I use a custom made and set it postionLeft to required point without using margin attribute. Note: The custom is set invisible in final code.

Still how the lap completion is detected?

A counter variable is used to declare the lap completion. Whenever a horse position equals to that custom position the lap counter value is increased, and after all the horse crosses that line the actual lap value is increased and checked with the user input value for lap completion.

The horse movement is random so it's position might not get exactly equal to the vertical line's position; how can you say it's reliable?

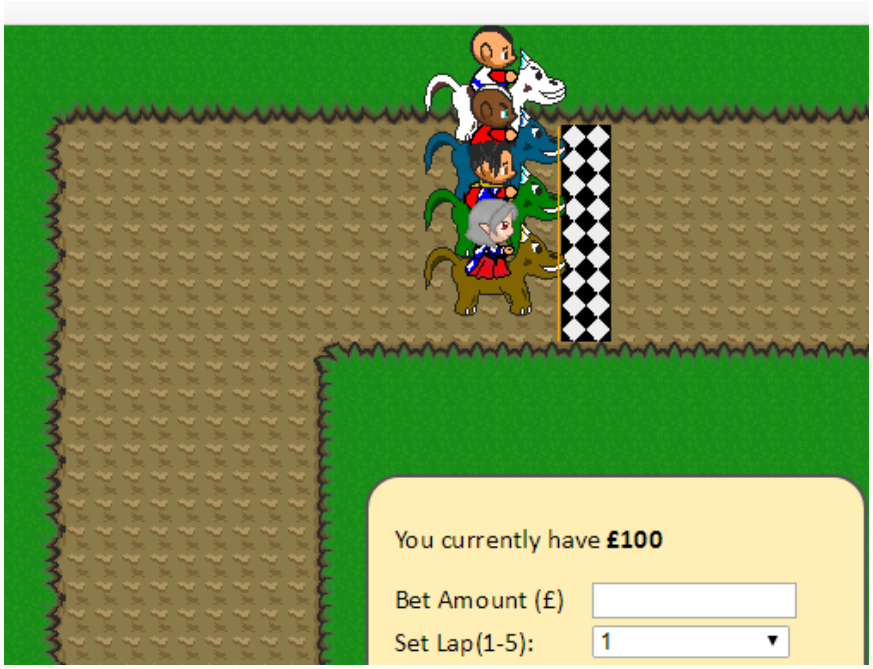

Figure 2.4.3.1.1 Lap Setting

Yes, the horse position might be random, but there is a special condition applied when horse is near to the finish line. When horse approach to finish line the condition gets activated and the horse movement is set to 1px for certain. This insures that the horse position gets equals to the line position only for 1 instance, this also prevent from incrementing the value of lap counter by multiple times in single contact. Also, it provides a slow-motion effect on horse movement when it touches the finish line and this could be added as extra feature of the game. (feature: *showing the movement of horse in slow motion when it approaches to finish line*).By using Jscript the current ongoing lap is also displayed on the game screen.

2.4.4 Relay Mode – Tag Team Racing

This is the Extra features of the game. Relay Mode disables the setting of choosing the horse for bet and enables the new setting of team selection. In game, there are 2 teams team1(white and green horse) and team2(blue and brown horse). As said already, the odds system does not apply to this **MODE**. It is set that If we win the bet we get double the amount we bet.

What happens in Relay Mode?

When race is started on Relay Mode at first White Horse and Blue Horse, horses from different team starts running. When white horse completes a lap then it stops at finish line and its team member (green horse in this case) start running, same happens when blue horse finishes the lap. Only after the 2nd horses from each team finishes the lap the actual 1 lap of the game is finished. Winner is declared on Relay Result Table on Game screen.

Figure 2.2.2.1 Result Tables

In Figure 2.4.4.1 there are 2 result tables. The Relay Result table and Results table.

The Relay Result table gives the result of Relay Game rank.

And Results table gives the result of Normal Game rank.

2.4.5 Sound Effects

There are 2 sound effect on the game.

1. Horse running sound
2. Audience/crowd cheering sound.

3. Program Design

My Jscript coding is completely based on functions and closure. I've used different kind of functions such as pure function (function that takes arguments and returns value) and other impure functions (which lacks the pure function attributes). Focus of program design is to reduce redundancy and make code readable and more user-friendly. My Jscript code is about 800 lines on the text editor and code this much large is very difficult to debug. Without proper program architectural design and commenting it's a nightmare to debug a code in java script.

In this topic, we will discuss on how the code is managed, how I used Jscript features to break the problem in small chunks and use the functions to improve the code efficiency.

3.1 MAIN CHALLENGE – CODE REDUNDANCY

Code Redundancy is a state where same code is typed (coded) at different part of program. Why is it bad? Code Redundancy increase the program size and makes it difficult to make changes in future.

In my code I have used functions, loops and closures to reduce Code Redundancy. Let's take an example of running horse to Right Direction, I have described earlier, the function moveRight() and its working. If I haven't created this function I had to write the code to move a horse to right direction for each horse. Similarly, for moving left I ought to write a code to move left 4 times for 4 horses, same goes to all other direction. But instead I wrote only 4 functions moveRight(), moveLeft(), moveUp() and moveDown(), and use these functions to move all the horses. Therefore, reducing lots of typing and program space.

Let's take another example of generating random numbers that I've mentioned earlier. The function rand() is used at number of places in the code which takes an argument and return random number from 0 to that argument. Without this function, I would have to write a code that gives random number at each place where needed.

The for-loop and while-loop are used to execute same code multiple times. These loops are used in declaring Rank, managing z-index, generating new random number, etc.

3.2 MAKING GAME INDEPENDENT OF SCREEN SIZE

The track of the game is relative to screen size. Increase in screen size cause increase in track length and width, but horses are not. Size of each horse if fixed with pixel value. Either if the screen size is big or small the ratio between horse size and track size is affected. It would be a problem if I had set horse to move after traveling certain distance because if the screen size is small then average then the horse would fly away from track and if the screen size is larger than average then the horse would turn even before reaching to turning point.

I have used percentage width of the view port as in Figure 2.1.1. so, if the screen size is larger than average, the width value of all variables is proportionally increased, same goes for the condition when screen size is small. When screen size is less, then all the component would get compact as components scaling is fixed. This game will work on any size devices but the appearance would not be good on too small or too large device. I suggest playing it on laptop with screen resolution ranging 1024px – 1440px for better graphics, though the game will work on any screen size.

If the screen size is too small, let's say the size of mobile then in that case due to fixed sizes of horses the horse might get stuck at turning point.

3.3 BREAKING UP THE PROBLEMS

It would be better to divide a huge problem into smaller individual parts, doing so not only it will be easier to solve that problem but also it will be easy to debug it. We can add extra features if we must make little bit changes on small chunk of code. In game, there were lots of problem/heavy tasks, like moving all horse in random motion around the track, generating odds value and change them depending on horse rank, using lap setting and different game mode, all these jobs are not done at instance but continuously working throughout the game, so just narrow approach toward the coding won't do any good on this. Let's discuss some of the important and complicated tasks in brief.

3.3.1 Running horses continuously until Lap ends

If we were to move the horse around the track for only 1 time then it would be efficient to just write a code that moves the horse in Right direction then meeting condition it turns and turns to make a round of the track, but in this case, we must move the horse around the track more than few times. What I did was we create running on certain direction function like *runLeft(), runRight()*, etc. which makes the individual horse run at particular direction. Then we create a condition to change these functions accordingly so that the horse turns at the turning point. At a same time, we keep a track on lap counter variable which increases as the horses passes the finish line. Then we shall stop the movement when the lap counter meets the amount of lap inserted by user. Here, lots of functions are working together and some functions are checking for certain condition to happen. Breaking the huge codes into small functions and implementing them where required is the key feature of Jscript. Also at the beginning of the game **CLOSURE** is used so that each time the start button is pressed, there won't be any error due to previous game set up or else.

3.3.2 Generating Random Odds and updating odds value

Game is designed such that we are greeted with lots of information even before the game has started. There is a custom Notice board at Right side of the screen which describes the game rules and other important stuffs. Most of user input values are pre-settled with default values as in the Figure 3.3.2.1.

The set Lap value, bet on horse value, and odds chart value are set default.

In my code a function named *oddsGenerator()*, generates the new random number that is set to the odds chart in Figure 3.3.2.1 as you can see all the numbers are unique ranging from 2 – 5. These values are used by a function named won() to calculate the winning amount of the user.

Figure 3.3.2.1 Game Display

Figure 3.3.2.2 Lap over

In Figure 3.3.2.2 Lap Over, it is the screen shot after a round is finished. As you can see in Result box blue horse character is First, followed by Green Horse character then white horse character and at last the brown horse character, as the odds for looser will be high and winner will be low for new round is seen at Odds Table. The odds value for winner is 2 and for looser it is maximum i.e. 5. For both ranking the horses and updating odds chart a simple array variable and **FOR** loop is used.

Extra: There is a current lap display screen at Right side of the track.

4. Testing

Testing is a the most important phase of programming. I have done lots of testing on my project. There are several approaches that I conduct the test my code such as, using console output, popping alerts to see if that function is working/ being called or not, use of certain variables to count number of iteration displaying on console, and also by directly viewing the game flow, etc.

Most of the testing process is conducted using **console** box and alert pop-ups, which can give output while running the game simultaneously. Bet calculations, horse animation, running certain chunk of code and to know it's output; these testing is done even before the game ends. Using such approach, it is easier to debug to code, as the console log also displays the line number where it is used. Jscript compiler also detect some syntax error which make work easier.

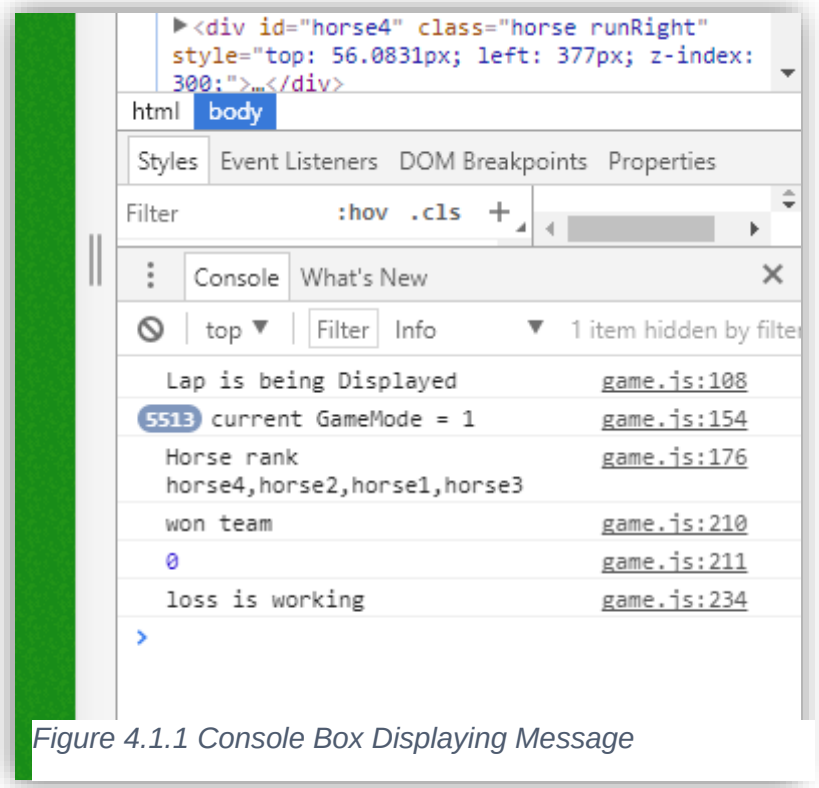

Figure 4.1.1 Console Box Displaying Message

There were some portion where I had to wait for a complete game to finish such as declaring winners, checking laps, updating odds value when game ends, etc. At first, I waited for an entire game to finish, but then I came to use the custom finish line to make the process faster. I displace that *div* near to the center of lane, when the horse crosses the center way of top lane then the game would declare finished and all the result would come sooner.

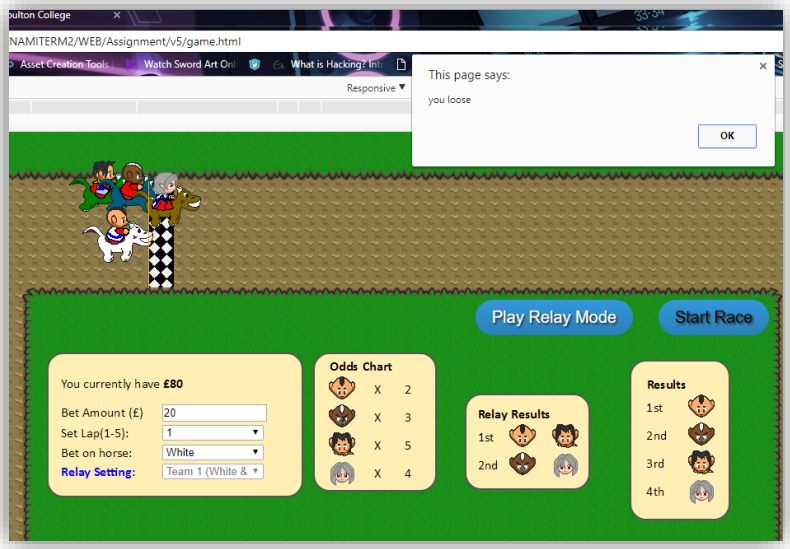

Figure 4.1.2 alert box popping you lose message

4.2 TESTING TABLE

Green Text: Extra Features Violet Text: Explanation Red Text: Minor Bugs

NO	Experiment Name	Expected Out Come	Output
1	Hose movement Direction and respective animation	If horse move to Right Direction then it's running animation should be on same direction, same to all other direction	Positive result, expected result is achieved.
2	Horse Turning effect	All horse should move within a track and it should move diagonally at turning point until turning is done	Positive result, Image distorts turning on left side corners.
3	Pressing 'Start Race' button	If user input in invalid then alert/prompt should pop-up. While game is running 'Start Button' should be disable. After that round is over Start button should be enabled again.	Positive result, expected result is achieved.
4	Odds chart	Random odds value should be given to all characters. When game is completed, the odds value should be changed for next round such that winner of previous match should have low odds value and looser of previous match should have high odds value	Positive result, expected result is achieved.
5	Lap Display	When game starts a small box should automatically appear on screen showing the current lap going as well as total laps. The current lap value should change when all the horses crosses first lap.	Lap is displayed. Lap value is increased as required. Positive result, expected result is achieved.
6	Bet Amount Entering	User shall able to enter the bet amount only greater than '0' and max of as much as the player currently have; violating this rule, game should generate message regarding invalid amount enter/input.	The user is getting prompt message when invalid amount value is used. (input type is number so only numeric character can be entered)
7	Game Over	Game over is the state when your credit amount goes down to '0'. When game is over, the game should generate message informing player that he/she no longer could play the game and provide facility to reset the game.	Positive result, expected result is achieved.
8	Screen size optimization	Game should run on all kind of screen	Game works perfectly on normal and larger screen. Due to fix horse image size, there is problem on mobile view.

9	Alert the result	After game ends, an alert box should pop-up informing player if he/she wins or lost the bet.	Yes, alert box appeared. Positive result, expected result is achieved.
10	Relay Button	Relay button should only be enabled while game is not in progress. Pressing Relay Button should disable choosing Horse for bet & enable to choose teams instead. Then the button should change to Normal Mode Button, which would bring the game to Normal mode.	Positive result, expected result is achieved.
11	Odds on Relay	Odds value should not affect Relay Game result. Odds value only works on Normal Game mode.	Odds values are still displayed, but it has no effect on Relay Mode game. Also, odds value doesn't update when Relay Game Ends.
12	Normal Mode Result	After a match ends in normal mode, the results Table should be updated according to the result of that game. The character Heads should be aligning from top to bottom in ascending order with respect to their rank on game.	Positive result, expected result is achieved. (initially heads are arranged in default order)
13	Relay Mode Result	When match in Relay Mode ends, the Relay Result box should be updated. Winner team's character's heads should be at top and looser team's heads should be below it.	Positive result, expected result is achieved. (initially heads are arranged in default order)
14	Single lap Duration	Each lap should not be more than 1 minute	Each laps ends before 1 minute.
15	DOM Content loaded	After DOM content loaded, all the graphical object should be present in the screen, and all default value should appear on their respective field (this includes head images, user input values). Only Lap Display screen should appear only after Game is started. Also, the rules Box should appear as well.	Positive result, expected result is achieved.
16	Sound effect	Sound should play only while game is started using 'Start Game' button & it should stop after game ends. Win and lose sound should play when game ends.	Positive result
17	Z-index value	Z-index value of horses should alter according to their position. Horse with highest Distance from top of screen should have highest Z-Index value & vice versa.	Positive Result. Detail on Topic **4.2.1 Z-Index testing**

4.2.1 Z-Index Testing

The Z-Index value is managed using Bubble sort Technique, function that manages this task is horseIndexing(). Figure 4.2.1 is screen shot showing a live game with google chrome developer mode. We can see all those purple highlighted lines are attributes of horses *div* that are being updated each second. Here div with id;

- horse1 is white horse div
- horse2 is blue horse div
- horse3 is green horse div
- horse4 is brown horse div

Here, in the game screen we can see that blue horse is farther from the top of the screen, followed by brown horse then green horse and at last the white horse. Thus, as expected out come in above table, we can see on those purple highlighted text, the z-index of horse2(blue horse) is higher followed by brown horse, then green and white respectively.

Why change in Z-Index value?

It makes graphical improvement when horses overlap each other.

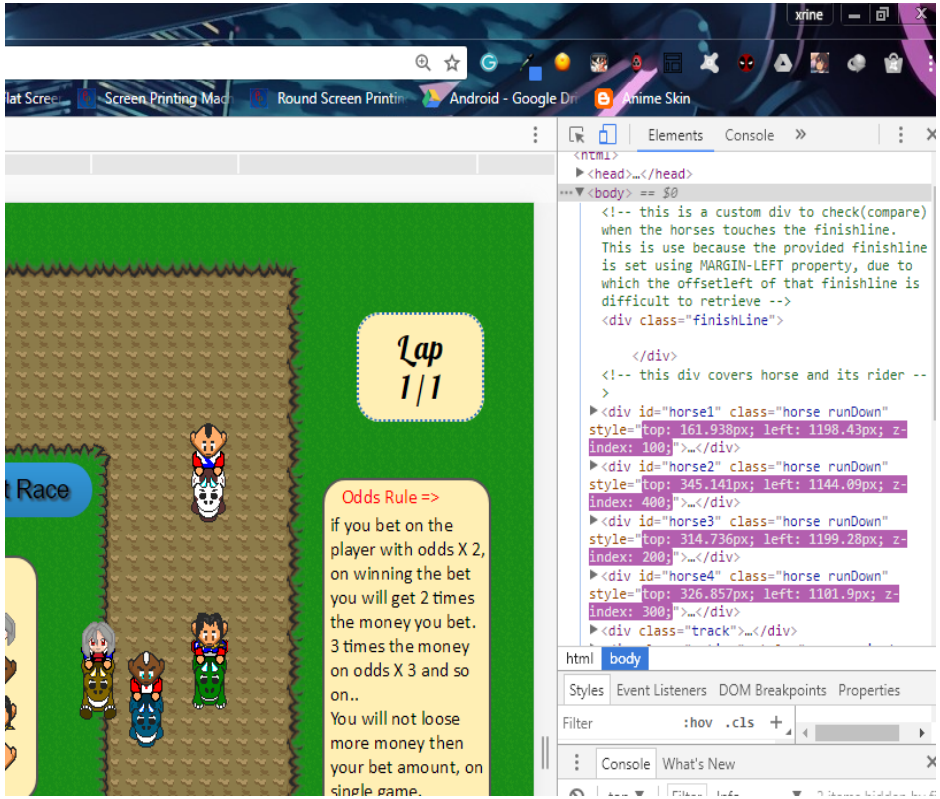

Figure 4.2.1 Z-index testing

5. Conclusion

The Game is fully functional, there are not any bug that would eventually crash the game Some minor bugs are due to pixel size of images. All the extra features of game such as Relay Mode, Lap Display, Notice board, Relay Result board, sound effects, etc. works well without any error. Most of the testing codes are commented in the final code so that the console won't flood with outputs while gaming.

A glitch;

- There is an image distort on horse image when horses turn at left side corners of the track. This glitch is not solved, because the actual problem could not be identified as the same code for Right Side corner of track works perfectly.

Also, as illustrated above in testing table, the game will work in any screen but if the screen is very small the horse might get stuck while turning due to fixed pixel size of horse div.

In a code, I had to write turning and movement code for each horse separately because of this the code is repeated, I tried to use closure function for this issue, but Relay Mode setting, Lap setting and Winner declaration management all depends on horse movement code. So using closure and managing all these function concurrently would generate lot of bug and complication. That's why the codes are written for each horse. This is one weakness of the code structure.

Apart for the above image glitch and small screen size problem, I don't find any bugs or distort. All other functions work well and efficient as exposed by testing table.

There could be lots of improvement in this game, as this is an assignment and I had limited time I could not add much of extra features than Relay Mode and sound effects.

Some of extra features and improvement that I would have done.

i. Better orientation of present graphic components.
ii. Use efficient sound clip.
iii. Introduce horse speed control as per user required.
iv. Betting while game is running.
v. Win and loss record history display screen.
vi. Custom made team on Relay Mode.
vii. Odds system on Relay game.
viii. Track changing.
ix. Random obstacles on track, that slows down horse.

If I had to build a similar game in the future, I would try to use closure function on horse movement and turning as well. This would make my code more efficient. Also, I would use a small sound clip on loop instead of using big sound files. This will reduce the game size and will be more efficient.

6. References

1 Stackoverflow.com. (2017). *Getting the z-index of a DIV in JavaScript?*. [online] Available at: https://stackoverflow.com/questions/1388007/getting-the-z-index-of-a-div-in-javascript [Accessed 16 Jul. 2017].

2 W3schools.com. (2017). *HTML DOM Audio play() Method*. [online] Available at: https://www.w3schools.com/jsref/met_audio_play.asp [Accessed 16 Jul. 2017].

3 YouTube. 2017. Horse Gallop Sound Effect freesound - YouTube. [ONLINE] Available at: https://www.youtube.com/watch?v=GRI_Ymc_8Xs. [Accessed 28 July 2017].