CSY1018: Web Programming

Due for Issue (week commencing):	**Sunday, 5th February, 2017**	**Last Date for Submission:**	**Sunday 30th April 2017 23:59:59**
Agreed Date for late submission:		Module Tutor:	Thomas Butler
Student ID:			

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

Specific aspects of the assignment that the 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.

CSY1018 Web Development
Javascript Assignment

Aims & Objective

The purpose of this assignment is to assess your ability to create an interactive web page using Javascript, CSS and HTML

Brief

The zip file Assignment2.zip provides some HTML and CSS for a horse racing game. Your task is to build a web based horse racing/betting game by adding the relevant Javascript code. You must use the existing HTML/CSS as a basis and build onto it. There are four horses which start on the start line and after pressing the start button should run one lap around the track.

You can set class names on the horse elements for running/standing animations and each of the four directions up/down/left/right. You will need to examine the CSS file to see which CSS classes are available to use.

Basic requirements:

For a **bare pass** (D- - D+) you must make the horses perform one lap of the track

1) Each horse should move to the top right corner of the track, then down the right hand side of the track, along the bottom and back up to the top turning while ensuring they stay on the track.
2) The horses should continue following the track until they reach the start/finish line.
3) As horses change direction and start/stop their animation should change. The animations are provided for you as CSS classes. The horses should always face the direction they are travelling.
4) As each horse reaches the start/finish line it completes the lap and should stop racing (the horses don't have to stop exactly on the line, but must go over it)
5) Pressing the start button again should reset the horses and start another lap

The speed the horses run at is up to you, but a lap should not take more than one minute.

Note: **The track fills the entire screen! Do not assume that everyone's screen will be the same size, you will need to calculate where the horses turn/stop based on the screen size. The track's width is 10% of the window's size.**

For a **good pass** (C- - B-) you must make the race randomised

1) Each horse should run at a different speed around the track
2) To make it so you can't see who will win by identifying who's fastest at the start of the race, the speed should vary as the horse goes around the track (e.g. one horse starts slow but later overtakes another horse by speeding up)
3) Don't have each horse turn at exactly the same point on the track
4) Detect the winner (the first horse to complete a lap) and display the results by listing the position and the name of the horse.
5) The start button should be disabled while the race is in progress.

For a **very good pass** (B – A-) you must implement betting:

1) The user starts with £100 and can bet on a horse by entering the amount they wish to bet
2) If they win, they get double their money back
3) They can then bet on another horse with their new funds and run the race again.

For an **excellent pass** (A) you must implement *one* of the following:

1) Randomly generate the odds for each horse at the beginning of the game. E.g. one horse might be 2 to 1 (you get twice your money back if you win). Another might be 100 to 1 (you get 100 times your bet back if you win). Once a horse wins, it's odds should decrease. E.g. if it wins, on 100 to 1 odds, its odds should be reduced for the next race, if a horse comes second its odds should decrease. This should stack, for example, if a horse has lost 5 times in a row it should have high odds.

2) Allow the user to specify a *number of laps* around the track that the horses will run over the course of the race before a winner is declared.

3) Adjust the shape of the track so it's more interesting than just a square, e.g. a figure of eight. You will need to amend the HTML/CSS and paths that the horses run.

4) You may also implement other features you feel would be interesting to add, the level of difficultly will need to be in line with the other options in this section.

Deliverables

Along with your code, you will need to provide a report which includes the following sections:

1. **Game design** - How did you come up with a solution to meet the brief?
    1. What did you need to do to work out how to get the horses to stop/turn at the relevant points on the track?
    2. How did you design the game so that a different horse won each time?
    3. How did you design the game to make it interesting? E.g. so overtaking happens

2. **Program design**
    1. How did you use language tools to reduce repeated code?
    2. What did you have to do to account for different screen sizes?
    3. How did you break the problem up into different tasks and;
    4. How did you use the features of Javascript to do this?
    5. Did you consider any alternative approaches? If so, why did you choose the approach you decided to use

3. **Testing**
    1. How did you test that your code worked?
    2. Could you test certain aspects of the code without running the entire game and waiting for it to finish?
    3. What tests did you carry out and what were the outcomes?
    4. What bugs did you discover during testing?

4. **Conclusion**
    1. A list of known bugs/weaknesses in the game
    2. What works well?
    3. What improvements could can be made?
    4. What else would you have done if you had more time?
    5. If you had to build a similar game in the future, what would you do differently and why?

In Addition to the report you must provide a video demonstration of your assignment working. The demo should be 5-10 minutes (No longer than 15 minutes). The video should demonstrate at least three rounds of the game and any additional features you have implemented.

Submission procedure

When you submit your work you will need to provide:

1. Your technical report as a word document
2. All HTML, CSS, Images and Javascript code in a zip and uploaded on NILE
3. All HTML, CSS and Javascript code copied to a word document (You must paste the code, not screenshots from your editor. It doesn't matter if you lose syntax colouring)
4. Video demonstration uploaded to NILE via Kaltura. Note: Kaltura uses your @my.northampton.ac.uk account. Instructions for using Kaltura can be found at: https://nile.northampton.ac.uk/bbcswebdav/orgs/Help/KalturaMediaspace/MediaSpace%20Student%20Guide%202015%20-%20Version%2013.pdf

Failure to comply with the submission procedure, e.g. not uploading a source code appendix or video will result in a capped grade of an F.

Marking Criteria

Aspect (& weighting)	Excellent A	Good B	Satisfactory C	Needs some more work D	Needs much more work F
Functionality (40%)	Contains all functionality required for a B grade along with the functionality listed under the "For a very good pass" section of the assessment brief	Contains all functionality required for a C grade along with all the functionality listed under the "For a good pass" section of the assessment brief	Contains all functionality required for a D grade along with some functionality listed under the "For a good pass" section of the assessment brief	Contains all functionality listed under the "For a bare pass" section of the assessment brief	Does not meet requirements for a bare pass
Game Design (15%)	All questions in the assessment brief have been answered with a good level of detail. Clear thought has been put into different approaches which could have been used and the relative strengths/weaknesses of the different approaches has been discussed	All questions in the assessment brief have been answered with a good level of detail. Some thought has been put into different approaches which could have been used.	All questions in the assessment brief have been answered with a satisfactory level of detail	Some documentation of the thought process behind designing the game. Most questions asked in the assessment brief have minimal answers	Nothing of value has been presented to document the thought process behind implementing the game's concepts
Program Design (15%)	All questions in the assessment brief have been answered with a good level of detail. Clear thought has been put into different approaches which could have been used and the relative strengths/weaknesses of the different approaches has been discussed	All questions in the assessment brief have been answered with a good level of detail. Some thought has been put into different ways the code could be structured.	All questions in the assessment brief have been answered with a satisfactory level of detail	Some documentation of the thought process behind the code structure. Most questions asked in the assessment brief have minimal answers	Nothing of value has been presented to document the thought process behind the structure of the code
Testing (10%)	Everything for B but also has a strategy that makes testing easier. E.g. being able to speed up the race so you don't have to wait for the entire race to complete each time.	Tests have been provided for most of the functionality. Any bugs found are listed and fixed or an evaluation of what needs to be done to fix the bug	Tests have been provided for most of the functionality. Any bugs found are listed.	Some tests have been documented for some of the functionality	Little to no testing strategy has been provided

Code Quality and Efficiency (15%)	Use of functions, loops, arrays and other language tools to prevent repeated code.	Everything for C but some attempt has been made at reducing repeated code.	Program meets basic requirements, minimal use of loops, functions to reduce repeated code.	Program meets basic requirements but code is repeated throughout and/or some bugs remain	Program does not work or contains errors which prevent the basic requirements being met.
Video Demonstration (5%)	Demonstrates multiple outcomes e.g. different horses winning, placing a winning bet and and placing a losing bet.	Demonstrates known bugs and several different iterations of the game being played.	Shows basic functionality but does not demonstrate known bugs	Shows basic functionality but does not demonstrate known bugs	Does not show the website meeting the basic functionality