CSY1018: Web Development

Due for Issue (week commencing):		**Last Date for Submission:**	

Aspect (& weighting)	Excellent A	Good B	Satisfactory C	Needs some more work D	Needs much more work F
Functionality (60%)					
Evaluation (10%)					
Testing (10%)					
Code Quality and Efficiency (15%)					
Video Demonstration (5%)					

Submission Guidelines – Please read carefully

1. The University of Northampton's Policy on Plagiarism & Mitigating Circumstances will be strictly implemented.

2. This is not a group project, by submitting this assignment you are asserting that this submission is entirely your own individual work. You may discuss the assignment with other students but any code written should be your own. **Sharing your work with another student, or submitting code that was written by someone else may be deemed academic misconduct.**

3. If you have used any code that you did not write you must:
   i. Correctly reference the code in the report (use Harvard referencing)
   ii. In your report, clearly document which lines of code you used, where you used them and what they are used for.

4. You must supply **all four** items of assessment and **upload them to the correct submission points**. All work must be uploaded to turnitin
   i. Report word document (uploaded to turnitin)
   ii. Source code (zip file). The marker must be able to download and run your code. *Do not include your video in your zip file*
   iii. Source code word document (uploaded to turnitin)
   iv. Video demonstration

Please make sure you double check all submissions. It is your responsibility as a student to ensure these guidelines have been met.

Failure to follow the submission guidelines may result in a capped grade of F-.

Assignment 2 (Javascript)

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 as2.zip provides the start of a game with existing functionality for:

- Walking your character around using the arrow keys
- Preventing the player walking above the horizon

The aim of the game is that the player should dodge falling bombs (see sample video). You should randomly create bombs at the top of the sky that fall down towards the player. When the bomb hits the ground it should explode.

Two CSS classes are provided for you for bombs:

- bomb (display a bomb)
- explosion (larger area, explosion)

When the bomb explodes, if the bomb touches the player they should lose a life (top right of the screen). When the player has lost all three lives the game should say "Game over" with an option to restart.

Requirements

For a **bare pass** (D- - D+) you must implement the following:

1) When the start button is pressed the game should begin and the start button should no longer be visible

2) Bombs get created at a random position at the top of the screen

3) Bombs fall down the screen towards the player

4) When the bomb hits the ground (green grass area) it explodes

5) If the player is in the radius of the bomb, print "game over" and set the player's animation to dead (css class "character dead")

For a **good pass** (C- - B-) you must improve the game

1) The bombs should explode at random points on the grass, not all at the same height.

3) The player will have three lives signified at the top right

4) Each time the player gets hit by a bomb the player loses a life and display the "hit" animation (css class, e.g. "character hit left"). The life should be removed from the panel in the top right.

5) If the player loses all three lives, display the *character dead* animation and print "Game over" to the screen

6) Display a "play again?" message to allow the player to restart the game if they lose

For a **very good pass** (B – A-) you must implement *two* of the following:

1) Implement a scoring system with high scores. Count the number of bombs the player managed to avoid and when they die have them enter their name. When the game ends, ask their name and log the score using Local Storage. When the game ends display the high scores.

2) Make the game more difficult by randomizing the bomb's speed and frequency at which they are dropped.

3) Make the bombs fall at different angles rather than straight down.

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

1) Add levels of increasing difficulty. For example, 10 slow bombs in level 1, 15 bombs in level 2, etc. Extra marks are available if there are an infinite number of levels that gradually increase in difficulty.

2) There is an "arrow" class and a "fire" animation for the player. Make it so the player can fire up and destroy the bombs before they hit the ground by pressing the space key. When the key is pressed:

- The player should be given the class "character stand up fire"
- The player should not be able to move for 0.5s while the arrow is being fired
- The arrow should fire up from the player's position and if it hits a bomb, both the arrow and the bomb should be destroyed
- The player should only be able to fire one arrow every 0.5s

Technical Details

The following CSS classes are available for you to use. You can chain CSS classes by choosing one of each of the following. For example, the CSS class list:

character up stand fire

Displays the player looking left and standing still while firing the bow. Animations with times (in brackets) will need to be removed after the animation has finished

Base class	**Directions**	**Primary Actions**	**Secondary Actions**
character	left right	walk stand	fire (0.5s) dead hit
arrow	left right up down		
bomb			
explosion			

Deliverables

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

1. **Checklist of completed functionality**
    1. A checklist of features implemented from the requirements section

2. **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 the correct condition to be met?
    3. What tests did you carry out and what were the outcomes?
    4. What bugs did you discover during testing?

3. **Evaluation**
    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. How easy would it be to extend the game to add more functionality?
    6. 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 checklist

1. Your technical report as a word document
2. All HTML, CSS, Images and Javascript code in a zip and uploaded on NILE. **The marker must be able to download, extract and run your code or your submission will be classed as incomplete.**
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
5. Your Javascript code must be your own work. 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.**
6. Excluding the supplied HTML/CSS and lecture notes, any code you submit which you did not write must be referenced and you must make it clear in your report which sections of the code you didn't write and reference it correctly.

Failure to comply with the submission procedure, e.g. not uploading a source code appendix or video may 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 (60%)	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
Evaluation (10%)	All questions in the assessment brief have been answered with a good level of detail. Possible structural enhancements have been noted.	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	Does not answer all questions asked in brief	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.. Adding more enemies would require very minimal changes to the java script file	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. winning/losing	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