CSCU9N6 Assignment

Due by 4pm, Monday the 31st of March 2025

For this assignment, you are required to create a 2D platform game of your choice. The type of game you choose to implement is up to you but it should be suitable for implementation as a platform or maze type game where the underlying structure can be controlled using a tile map. The complete assignment is worth 100% of the overall mark for the CSCU9N6 module.

Specification

Your submission should be based around a single player, 2D game using the programming components and principles discussed in the lectures and practicals. Your game should include the following elements:

- Animations with suitable animation changes for player and controlled sprites
- One or more moving player controlled sprites
- Multiple moving computer controlled sprites that interact with the player and environment
- Correct collision detection and handling for all sprites and tile maps
- Multiple sounds that are appropriate to in game events
- Use of a novel sound filter
- Control of background theme music that uses a MIDI track
- Multiple keyboard and mouse events
- Interactive use of 2D tile maps with at least 2 different levels
- Parallax scrolling of the background to give an illusion of depth

In conjunction with a report, these elements contribute towards the overall assignment mark so please do not concentrate on fully implementing a small subset of them while ignoring the rest. The breakdown of marks are as follows: Animations & Sprites - 20%, Complexity & Collision Detection - 20%, Game World (tile maps/interactions/parallax) – 20%, Sounds - 20%, Report - 20%

Features

In order to achieve a good mark for each component, you must extend and expand upon the concepts discussed in the lectures. For example, when playing a sound filter, you should define one of your own and use this, rather than copy the sound filter code provided in lectures and practicals. When implementing collision detection, you should use multiple levels of collision analysis where appropriate and aim to be efficient when detecting and handling collisions. Interaction between the player character and the underlying tile map should be accurate and consistent with sprites capable of moving correctly around the map without getting stuck.

Your game world will need to be greater in size than the physical windowed game size. In this type of game, the world should appear to move around the player rather than the player move around the world (the player will appear stationary as the world moves underneath them). You should be able to use the offset values discussed in lectures and code to achieve this and this has been partially implemented for you in the starter code that is provided. The best approach to implement this is to keep the complete game world coordinate system consistent and then adjust the perspective only at the point you *draw* the player and world on the screen.

To achieve an excellent mark you will need to implement all of the above features effectively and efficiently. Your code should allow for easily expanding the entities and the number of levels in the

game without the need for significant design or coding changes. For example, you should use lists of computer controlled sprites rather than individually named variables for each sprite and the computer controlled sprites should interact with the environment rather than using hard-coded paths.

The game should have at least two playable levels and should demonstrate that you have learnt how to integrate the elements discussed in the lectures into a cohesive game. 20% of the marks will be awarded with respect to the challenge involved in writing the code that controls your game sprites and how they interact with the player and environment (for example, do they pursue or fire objects at the player) .

Your code should make good use of general purpose methods rather than just use hard coded objects declared in your classes. You should also aim to use inheritance in cases where you wish to add additional functionality to the classes given in the game2D library.

Please note that your submitted code should compile successfully - a penalty will be applied to any code that does not compile and only a minimal effort will be made to try and make it compile if there are problems. If you have code that attempts to implement a particular feature but it is causing problems, please just comment it out and make a note of what it was trying to do in your code and also in your report. This may still gain you some marks based on what you were trying to implement.

Your code should be properly commented and it should be clear what each class and method is aiming to do. You should use the standard Java Doc format when commenting your method headers and you should find that this will also make it easier for you to understand what each method does (see the code handout for examples of this commenting style and the level of commenting that is expected).

Assignment Code

The initial source code for the 2D assignment is available at the top of the assignment page on Canvas. This code is based on the GameCore library you have been using in your practicals with several additional useful features.

You must not use a different code library or game engine to build your game **or use any other library code that you did not write yourself. Any attempt to do this will result in a mark of 0 for your assignment and the module as a whole.** You can add further functionality to the supplied classes, extend them or override methods but you should not need to replace or substantially rewrite any of the supplied classes.

In addition to the game, you should submit a report of up to 3 sides of A4 that should critically evaluate your implementation of the game and point out where you think it was successful and where it needs improvement. You should also use this as an opportunity to highlight any features of your game that may not be readily obvious but that you think worked well. At the end of the report, please note how your game could be extended based on the code that you have already written.

Submission Process

Your assignment should be submitted in two parts:

1. Assignment Code
   Please *compress the complete IDE project folder* sufficient to compile and run your game into a ZIP file and upload this as a single file onto the Canvas assignment submission page.
2. Report
   Please submit this as a second file on the Canvas submission page. Feedback for the complete assignment including the code submission will be provided in response to the report submission on Canvas.

Ensure that you put your student ID number but not your name on the first page of the report and in the code that you submit.

Late Submission

If you cannot meet the assignment hand in deadline and have good cause, please see the module coordinator to explain your situation and discuss an extension as soon as possible after the problem becomes known. Coursework will be accepted up to seven days after the hand-in deadline (or expiry of any agreed extension) but the mark will be lowered by three marks per day or part thereof. After seven days the work will be deemed a non-submission and will receive an X for this assignment.

Plagiarism

Work which is submitted for assessment must be your own work. Plagiarism means presenting the work of others as though it were your own. The University takes a very serious view of plagiarism, and the penalties can be severe (ranging from a reduced mark in the assessment, through a fail mark for the module, to expulsion from the University for more serious or repeated offences). We check submissions carefully for evidence of plagiarism, and pursue those cases we find.