**Logo

Description automatically generated**

Gaming Programming Project 2022

3.7 Use complex programming techniques to develop a computer program **6 credits** internal (version 1)

3.8 Use complex processes to develop a digital technologies outcome **6 credits** internal (version 1)

**Programming Project NCEA Level 1**

**Project outline:**

**Game Requirements**

* Can win / lose the game (or have a game end and replay for improvement like endless runner games / flappy bird)
* Uses player input from mouse or keyboard
* Have 3 difficulty options or more (speed, amount of enemies, lives ect will change for difficulty)
* Takes username and display at the game over (win / lose screen) along with their score
* Displays a scoring unit during gameplay and at the end screen (score is just a way for the user to gauge progress like distance or time)
* Use multiple enemies / objects this is coded as an array of objects (use of images)
* Use of animation or arrays colliding with arrays & external javaScript files and use of 1 class or more

**GitHub Wiki Component**

* + Game overview of the scenario how to win/ lose, user controls, sprites behaviour
  + Feedback section, what was the feedback and what you did in response
  + Game code planning (variables, arrays, functions (their logic)
  + Need to write about 2 relevant implications (describe in context what they are)

**GitHub Project/tickets and commits Component**

* Program is broken down into small steps / tasks (30-40 for completed project)
* For each completed ticket code should be committed / uploaded
* Every time you run into bugs convert the ticket into an issue and when fixed resolve the issue and commit code
* For each new area, stage or sprite when building your code create a new branch for your commits (this is like patch updates when it is completed to a stage that the new feature is useable not done merge the branch back to the main)

**Testing**

* Complete the excel testing template
* All inputs and expected results (key presses, username, difficulty)
* Testing boundary cases like collision and health becoming 1 and 0
* Testing unexpected cases, user trying to do the wrong thing for entry e.g. cancelling a prompt.

**Code requirements**

* Comment groups of vars, constants, end of functions and what difficult does (Collisions)
* Comment your code (where relevant describe what the code does and why this approach eg. this ensures the game doesn’t begin until the user finishes putting in their name)
* Use of private vars, code indentation and no use of literals (400 vs canvas.width / 2)

**AS91906**  Use complex programming techniques to develop a computer program

**Developed by Daniel Greenwood Head of Digital Technologies 2022**

6 credits internal Version 1

Name: Grade:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Achieved** | **Description** | **Grade** | **Resub** | **Comments** |
| writing code for a program that performs a specified task   * Your game is complete * UserName and difficulty * End of game screen win/loss and score display | A  A  A |  |  |
| Using complex techniques in a suitable programming language   * Use of classes * Storing multidimensional data in collections **(2d arrays, array of arrays/ objects)** * Responding to events generated by a graphical user interface **(GUI)** | A  A  A |  |  |
| setting out the program code clearly and documenting the program with comments   * Comments that state groups of vars * States where functions end * Few comments on what the complex part of the program does * Splitting the classes into separate external JavaScript files | A  A  A  A |  |  |
| testing and debugging the program to ensure that it works on a sample of expected cases   * Excel sheet documenting all test cases of expected inputs (doing what a user should) * What testing, expected result and tested result | A  A |  |  |
|  | | | | |
| **Merit** | documenting the program with appropriate variable/module names and organised comments that describe code function and behaviour   * Comments describe how complete parts of the program behave | M |  |  |
| following common conventions for the chosen programming language   * All caps for constants * camelCase for variables * code indented * array names are pluralised * program split into separate external JavaScript files | M  M  M  M  M |  |  |
| testing and debugging the program in an organised way to ensure that it works on a sample of both expected cases and relevant boundary cases   * excel log has the boundary cases e.g. collisions, state changes (health 1 and health 0) * has expected outcomes for those cases * has tested outcomes for those cases | M  M  M |  |  |
|  | | | | |
| **Excellence** | ensuring that the program is a well-structured, logical response to the task   * no use of literals (400 vs canvas.width) * program runs without bugs * program broken down into clear functions * use of constants and private variables | E  E  E  E |  |  |
| making the program flexible and robust   * use of for loop(s) or forEach loop(s) with conditional statements * conditions use greater or less than to ensure all boundary cases are caught | E  E |  |  |
| comprehensively testing and debugging the program   * Testing for unexpected entries e.g. when asked to choose a number and the user types the number as a word * User hits cancel or null entry. * Excel log shows test case, expected outcome and actual outcome | E  E  E |  |  |

**AS91907**  Use a complex processes to create a digital technologies outcome

6 credits internal Version 1

Name: Grade:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Achieved** | **Description** | **Grade** | **Resub** | **Comments** |
| using recognised and appropriate project management tools and techniques to plan the development of a digital technologies outcome   * GitHub wiki has an outline of general game play, rules and sprite behaviours * Vars, consts, array(s), javaScript files, classes & functions named in wiki * Functions and classes broken down stating what they will do | A |  |  |
| decomposing the digital technologies outcome into smaller components   * In projects area have created tickets that has broken down all steps into small tasks (some functions will be more than 1 task) 20 plus tickets | A |  |  |
| trialling the components of the outcome   * Program is uploaded to 3+ branches before merging to main project for new features and updates (branch is finalised before merging to main) * Branch, tickets and commit done from feedback changes | A  A |  |  |
| testing that the digital outcome functions as intended   * Testing excel log and project tickets move through testing stage * Game functions without gameplay bugs | A  A |  |  |
| addressing relevant implications   * Stating in wiki how relevant implications were considered and resolved |  |  |  |
|  | | | | |
|  | effectively using project management tools and techniques to manage development, feedback and/or collaborative processes   * Commits 20+, tickets 20+ and wiki notes on at least 1 area of feedback you received from testing and what improvements you made as a result. * 3 branches made for stepped improvements and merged |  |  |  |
| effectively trialling multiple components and/or techniques   * 1-2 branches for game improvements with a few commits and clearly selecting the best method when merging the branch. | M |  |  |
| effectively using information from testing and trialling to improve the functionality of the digital technologies outcome   * Feedback improvement section in wiki outlines what areas were changed from feedback and how it improved the outcome |  |  |  |
|  | | | | |
| **Excellence** | synthesising information gained from the planning, testing and trialling of components   * The program displays a robust outcome that works well, submitted on time and is clear to play for a new user. | E |  |  |
| discussing how this information led to the development of a high-quality digital technologies outcome  Wiki final evaluation how the correct project process helped produce your program that help for time management and creation of a robust result |  |  |  |

**GitHub Link:**