|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2.7** | **Use advanced programming techniques to develop a computer program** | **6 credits** | **Internal** | **Version 1** |
| **2.8** | **Use advanced processes to develop a digital technologies outcome** | **6 credits** | **Internal** | **Version 1** |

**Programming Project NCEA Level 2**

**Project outline:**

**Game Requirements:** You need to plan and create a HTML Canvas game that contains and or does the following.

An advanced computer program:

● uses variables storing at least two types of data (e.g. numeric, text, Boolean)

● uses sequence, selection and iteration control structures (if, else, for loops, forEach loops)

● takes input from a user, sensors, or other external source (keypress, mouse move/ click)

● produces output (game play)

Examples of advanced programming techniques include:

● modifying data stored in collections (arrays)

● storing multidimensional data in collections (2d array aka array of arrays/objects)

● creating methods, functions, or procedures that use parameters and/or return values

● responding to events generated by a graphical user interface (GUI)

Examples of ways of making a program flexible and robust include:

● using actions, conditions, control structures and methods, functions, or procedures effectively (breaking program into functions, functions have for or forEach loops, and if and else conditions)

● using constants, variables and derived values in place of literals (literal = 400, derived = canvas.width)

● checking input data for validity (testing log for evidence)

● correctly handling expected, boundary and invalid cases (testing logs)

**Recommended games**

* Avoid falling object/ catch falling objects
* Flappy bird remake
* Road crossing
* Don’t get pushed out of the bottom of the screen
* Shoot the enemies before they go out of the back of the screen (1 bullet that moves fast)
* Jet pack/ rocket landing game with fuel

**Phase 1 (planning)**

* Select the type of game you will make, general description of how the game will function and rules (in GitHub wiki)
* break down the game into parts. (in GitHub wiki)
* These parts will be broken into functions with described behaviours (in GitHub wiki)
* Transfer the game parts and functions into project section of GitHub as tickets

**Phase 2 (build + trailing/ testing)**

* Rapidly create your program focused on getting a rough working program going
* Trial and test program function while improving an polishing the program (going from blocks to images to animated sprite sheets)
* Use of 2d Array for either username + scores or character animation with sprite sheets

**Phase 3 (documentation) – this is a large part of the 6 credit standard**

* Create an excel document that is uploaded to GitHub
* The excel testing document will have what you are testing, expected result and the actual result
* Testing expected or informed users that don’t do the wrong thing
* Testing user inputs and key presses or mouse input
* 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.
* Need to write about 2 relevant implications (describe in context what they are)

● usability ● functionality ● aesthetics ● sustainability and future proofing ● end-user consideration

* *Final Evaluation on how beneficial the use of the processes to create the program was in GitHub Wiki*

**Phase 4 (game assembly & creation) standard**

* Build your completed game
* 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)
* Test your game to ensure you remove bugs
* ***For each ticket or 2 commit your code with supportive comments***

**AS91896**  Use advanced programming techniques to develop a computer program

6 credits internal Version 1

Name: Bradley Cade Grade: Yes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Achieved** | **Description** | **Grade** | **Resub** | **Comments** |
| writing code for a program that performs a specified task   * Your game is complete * User name and difficulty * End of game screen win/loss and score display | -Y  -Y  -Y |  |  |
| using advanced techniques in a suitable programming language   * Uses variables storing at least two types of data (e.g. numeric, text, Boolean) * Uses sequence, selection and iteration control structures **(loops, ifs)** * Takes input from a user, sensors, or other external sources **(key or mouse presses)** * modifying data stored in collections **(arrays)** * Storing multidimensional data in collections **(2d arrays, array of arrays/ objects)** * Creating methods, functions, or procedures that use parameters and/or return values * Responding to events generated by a graphical user interface **(GUI)** | -Y  -Y  -Y  -Y  -Y  -Y  -Y |  |  |
| 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 | -Y  -Y  -Y |  |  |
| 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 | -Y  -Y |  |  |
|  | | | | |
| **Merit** | documenting the program with appropriate names and comments that describe code function and behaviour | -Y |  |  |
| following common conventions for the chosen programming language   * All caps for constants * camelCase for variables * code indented * array names are pluralised | -Y  -Y  -Y  -Y |  |  |
| testing and debugging the program effectively 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 | -Y  -Y  -Y |  |  |
|  | | | | |
| **Excellence** | ensuring that the program is a well-structured, logical response to the specified 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 | -Y  -Y  -Y  -Y |  |  |
| 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 | -Y  -Y |  |  |
| 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 | -Y  -Y  -Y |  |  |

**AS91897**  Use advanced processes to develop a digital technologies outcome

6 credits internal Version 1

Name: Bradley Cade Grade: Yes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Achieved** | **Description** | **Grade** | **Resub** | **Comments** |
| using appropriate project management tools and techniques to plan the development of a digital technologies outcome   * Use of GitHub wiki to plan with statement of general game play & rules * Program broken down into parts in wiki * Broken down parts contain a description of functions behaviours * Variables, constants, array and functions named in wiki | -Y  -Y  -Y  -Y |  |  |
| decomposing the 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) | -Y |  |  |
| trialling the components of the digital technologies outcome   * Some parts of the program is uploaded as branches as you are testing 2 or more methods/ approaches * Parts of your game will be reworked e.g. blocks changes to images | -Y  -Y |  |  |
| testing that the digital technologies outcome functions as intended   * Testing excel log and project tickets move through testing stage * Game functions without gameplay bugs | -Y  -Y |  |  |
| explaining relevant implications   * In wiki describe 2 relevant implications that apply to your program **(but not how your project sorted this)** | -Y |  |  |
|  | | | | |
| **Merit** | effectively using project management and version control tools and techniques to manage the development of a digital technologies outcome   * Consistent use of projects section in GitHub * Regular code commits * Bug logged in project and notes made around how the bug was resolved | -Y  -Y  -Y |  |  |
| trialling multiple components and/or techniques and selecting those which are most suitable   * Evidence that a part of the game that was functioning has been improved through reworking/ modifying * Committing an upload branch as it was selected as the best choice for the program | -Y  -Y |  |  |
| using information appropriately from testing and trialling to improve the functionality of the digital technologies outcome   * Notes in testing tickets that show changes or improvements were made to program | -Y |  |  |
| addressing relevant implications   * Program shows consideration for relevant implication through its design and function | -Y |  |  |
|  | | | | |
| **Excellence** | discussing how the information from planning, testing and trialling of components assisted in the development of a high-quality outcome.   * Wiki final evaluation how the correct project process helped produce your program that help for time management and creation of a robust result | -Y |  |  |