Level 3 Programming Assessment

91906v2: Use complex programming techniques to develop a computer program (6 credits)

91907v2: Use complex processes to develop a digital technologies outcome (6 credits)

Introduction

This assessment activity requires you to plan, develop and create a complex computer program.

You will be assessed on

- how effectively you use project management techniques to plan and manage the development of a digital outcome
- how effectively you decompose the problem into smaller components, and test and refine your outcome so that it is a high-quality response to the task
- how well you test and debug your program
- how well you have addressed relevant implications
- how well you synthesise information from the planning, testing, and trialling of components to develop a high-quality response to the task (e.g., well-structured, logical, flexible, robust and comprehensively tested program)
- discuss how this information assisted in the development of a high-quality outcome.

Choose one of the following tasks:

Option 1: A quiz program for testing knowledge of Te Reo Māori.

The questions can be multiple choice or one word / one number answers. A record of correct/incorrect answers is kept, and statistics gathered about the responses. There could be an option to export the statistics and record of questions & answers to a text file.

Students may wish to consider quizzes based on…

- Tikanga Māori
- Aotearoa New Zealand place names
- Months, days, colours, or numbers

Option 2: A single player 2D car racing game.

Some images have been provided in the Assessment Resources folder.

Your program needs to include at least the following (you can add additional features/functionality if you wish):

- It should include the player's car and at least four other cars.
- The player should be able to control the direction of their car.
- The other cars should move at different (random) speeds.
- If the player's car hits any of the other cars, the game is over.
- Scoring should be based on the number of cars the player car passes before the game is over (i.e., until the player car hits any of the other cars). The score should be displayed on the screen throughout the game.
- You need to record and retain the high score for the game. Save a high score to a file, so that when a new game is started, the high score is retained.

You need to use a third party/non-core API, library, or framework (like Pygame) to create the interface for the game.

Program requirements

Writing code that performs a specified task and uses complex programming techniques. You should be able to show evidence of at least **TWO** complex programming techniques. Examples of complex programming techniques include writing code that:

- creates a graphical user interface (GUI)
- reads from, or writes to, files or other persistent storage
- defines class(es) and creates objects
- defines and uses custom type(s)
- uses third party or non-core API, library, or framework
- uses complex data structures (e.g., stacks, queues, trees).

Programming code should be set out clearly. Document the program with appropriate variable/module names and organised comments that describe code function and behaviour. Use appropriate variable/module names and follow conventions for your chosen programming language.

- Show comprehensive testing and debugging of the program. This should be carried out in an organised way to ensure that it works on expected cases and boundary cases.
- Ensure that the program is a well-structured, logical response to the task.
- Ensure that the program is flexible and robust.

Planning, Development & Testing Requirements

Your program needs to be developed using complex processes and you will need to show evidence of this process.

- Use recognised project management techniques, GitHub and Trello, to help you plan and the development of your program.
- Decompose your outcome into components.
- You should trial multiple components or techniques to determine which one will be best for the overall quality of your outcome.

> **Important Note - for the 3.8 standard (Use complex processes):**
> You MUST trial multiple components and/or techniques and select the most suitable. This means trialling different ways to solve the same sub-problem and selecting the best. Even to get Achieved, you MUST have evidence of how you trialled AT LEAST TWO of your components and then gave reasons for choosing one over the other.
> NOTE that testing is about making sure something works. Testing and Trialling are NOT the same thing. Trialling is about trying out different ways of doing the same thing.
> For example, you might trial the use of pop-up menus, text boxes or radio-buttons to get the same information from the user in a GUI. You might assess their quality in terms of functionality, usability, flexibility, interface aesthetic, etc. for your outcome.

- You need to provide evidence of effectively using the information from testing and trialling to improve the functionality of the outcome. Use your project management techniques to record and gather evidence of testing, trialling, and feedback. You should take screen captures, recorded in a simple table showing dates, images and a brief statement identifying the stage of your process. Be sure to annotate/discuss the changes you have made and why.
- Provide evidence of comprehensively testing your final outcome.
- Address any relevant implications such as usability, functionality, legal/ethical requirements.
- Synthesise the information from the planning, testing, and trialling of components to develop a high-quality outcome. This should include evidence showing how user testing has been incorporated / used to shape the final outcome.
- Discuss how this information led to the development of a high-quality outcome.
  - This should include evidence of how the outcome addresses relevant implications.