|  |  |
| --- | --- |
| ****Subject Name and CRN:**** | INFT562 – Software Development 2  CRN - 83109 |
| **Assessment Type:** | Project |
| **Due Date:** | September 10th, 11:59PM AEST |
| **Weight:** | 50% |
| **Marks:** | 100 |
| **Learning Outcomes:** | 2, 3, 4 & 5 |
| **Canvas:** | <https://aie.instructure.com/courses/188/assignments/553> |
| **Topic:** | Tool Project |

# Project Brief

You are tasked with creating an application to aid in the development of games in the form of a tool, specifically using visual design elements to create controls and interfaces. It is highly recommended that you create a tool to aid in the creation of animated 2-dimensional sprites, or 2-dimensional map creation, but you are able to create any tool that has a graphical interface. You must first provide a proposal to your assessor that must be approved for development, before you begin.

# Project Features

Based off suggestions from your assessor, you are to write a proposal for a tool to aid in game development. This proposal should be a document at least 1 page or more, and must include visual mock-ups of the proposed tool’s graphical interface.

This proposal must be submitted for approval **before development begins** on the tool.

Once the proposal has been approved you are to develop the tool. The tool must have the following features:

* The tool saves and loads data to a file format of your choosing.
* The tool has custom visual elements that may include buttons, text boxes, sliders, control box, drop-downs and forms.

**Part 3** of this document lists potential example applications that you may develop.

To meet the minimum requirements, you must submit:

* Tool proposal submitted for approval
* Working stand-alone application (runs outside of an IDE)
* Source code for the application
* An example saved file created by the tool that can be loaded

The following pages contain references for example applications that you might consider creating.

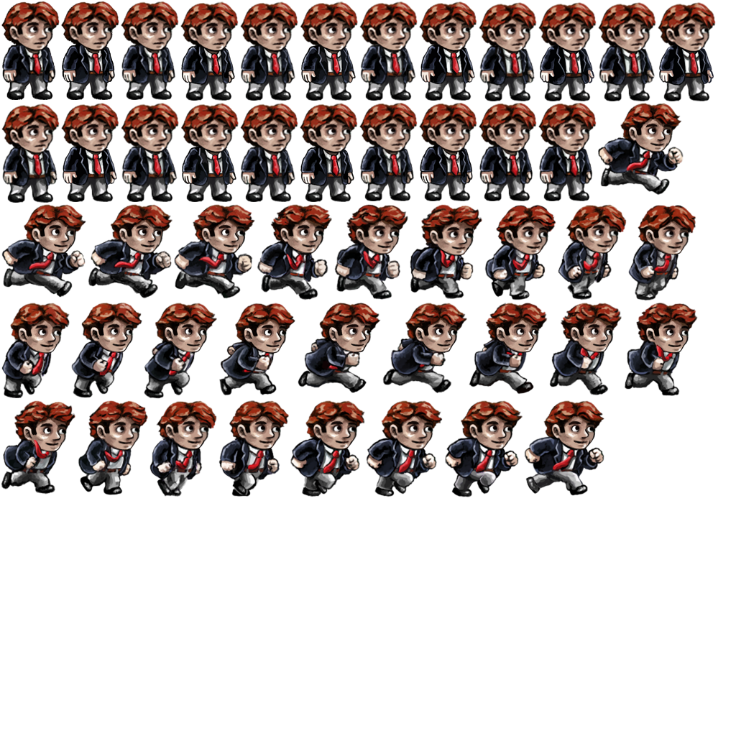
# Example Applications

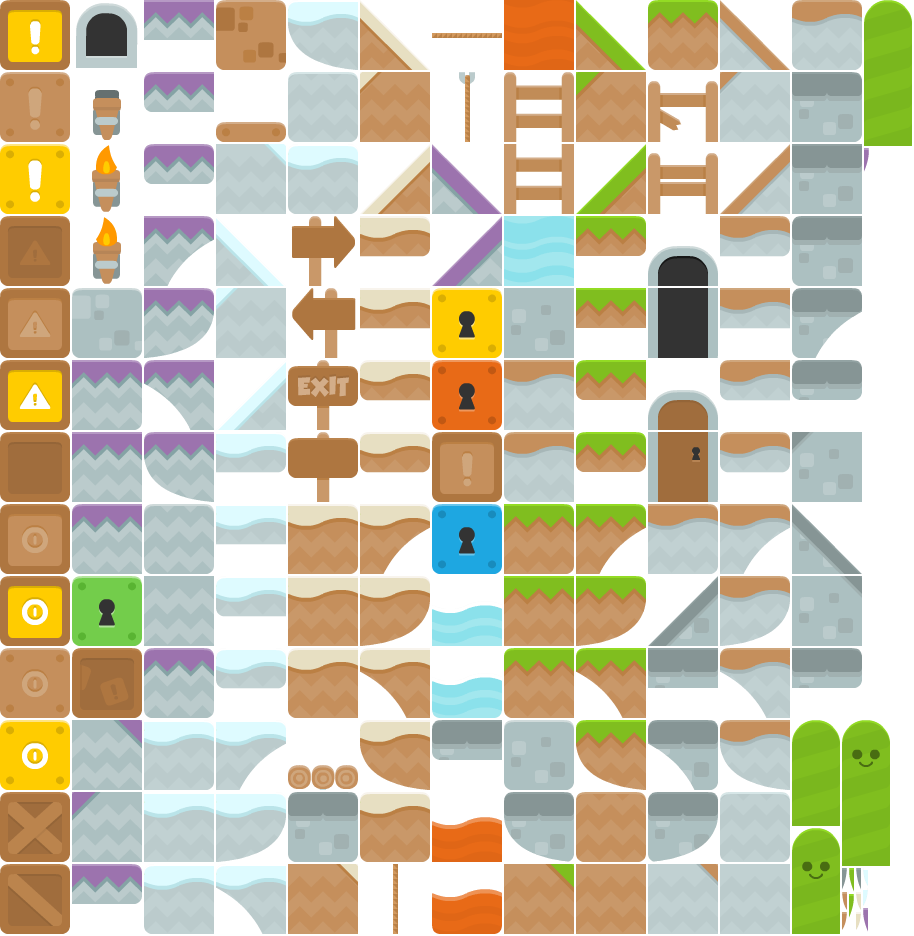
It is highly recommended that you use an API such as WinForms with C# to create an application as it will aid in the creation of the graphical interface.

Potential applications of the tool may include:

* Animated Sprite Editor
* Texture Atlas Editor
* GUI Sprite Sheet Editor
* 2-dimensional Map Editor
* Sprite Font Editor
* Artificial Intelligence Editor, such as:
  + Finite State Machine Editor
  + Behaviour Tree Editor

A 2-dimensional sprite sheet (also called a sprite map, tile maps or a sprite / gui atlas) is an image that contains the images of several smaller individual images used in 2-dimensional games for animation purposes or as images used for 2-dimensional overlays such as menus and heads-up-displays (HUDs). For examples, see Figure 1 and Figure 2.

Figure 1. Character Sprite Sheet

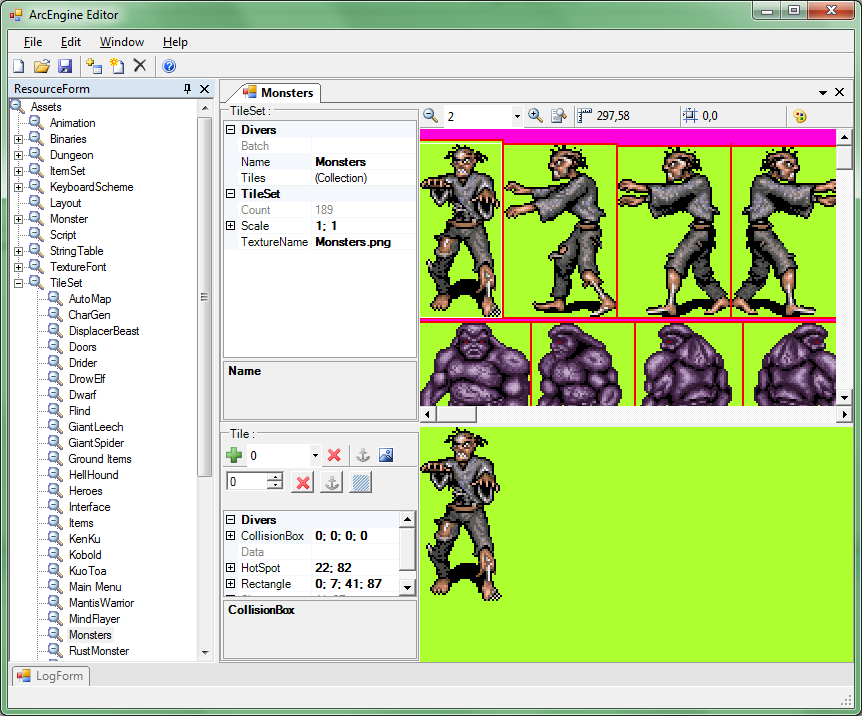
Figure 2. Level Sprite Sheet

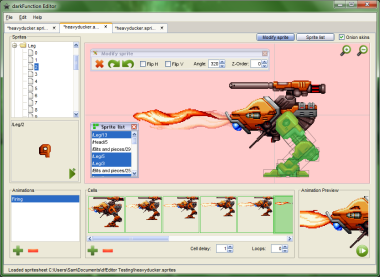
As an example, for a **Texture Atlas Editor** an image that your tool loads may contain all of the various icons for a HUD, and your tool must allow a user of the tool to select the portion of the image that represents the “**Health**” icon for the game. This region is assigned a name, an integer identifier, and the coordinates of the texture that the region represents. The tool then saves a file that lists all of the different regions that the user has created so that a game could load the original image that contained all the regions, and the file that specifies which sections of the image represent which element of the HUD.

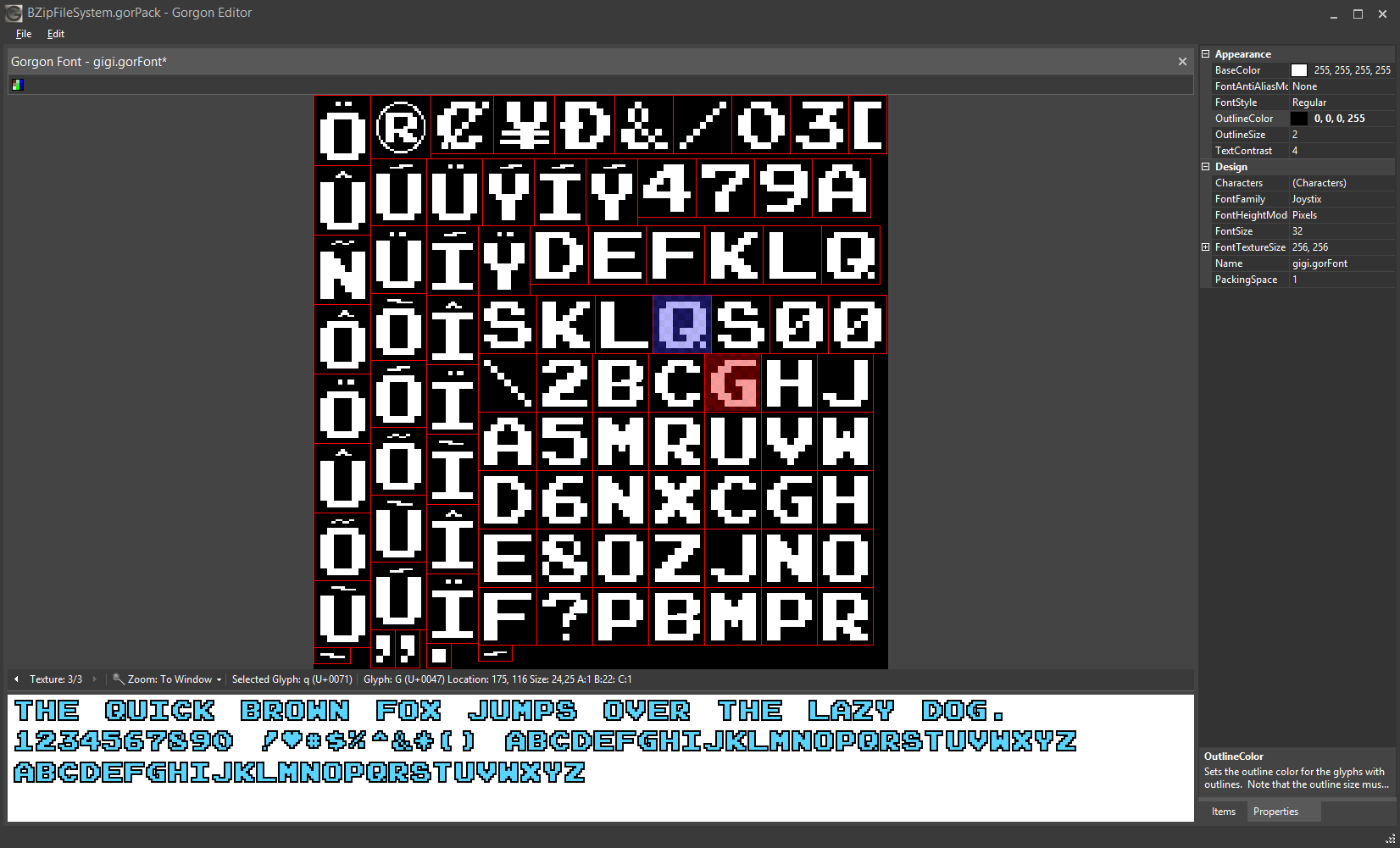
The same idea would work for an **Animated Sprite Editor**, which involves selecting regions from an image to represent individual frames within an animation, then creating an animation that lists which frames to show in a certain order, how long to show each frame, and how long the animation plays for and if it should loop or not. This animation data would then be saved.

A **Sprite Font Editor** is another example that works in a similar manner, where each regions represents a different character to be displayed. When a user then wants to display text in a game they would simply look up the font sprite to find the sections of the image that represent each character.

Refer to figures 3 to 7 for examples.

Figure 3. Sprite Animation Editor

Figure 4. Sprite Animation Editor

Figure 5. Sprite Font Editor

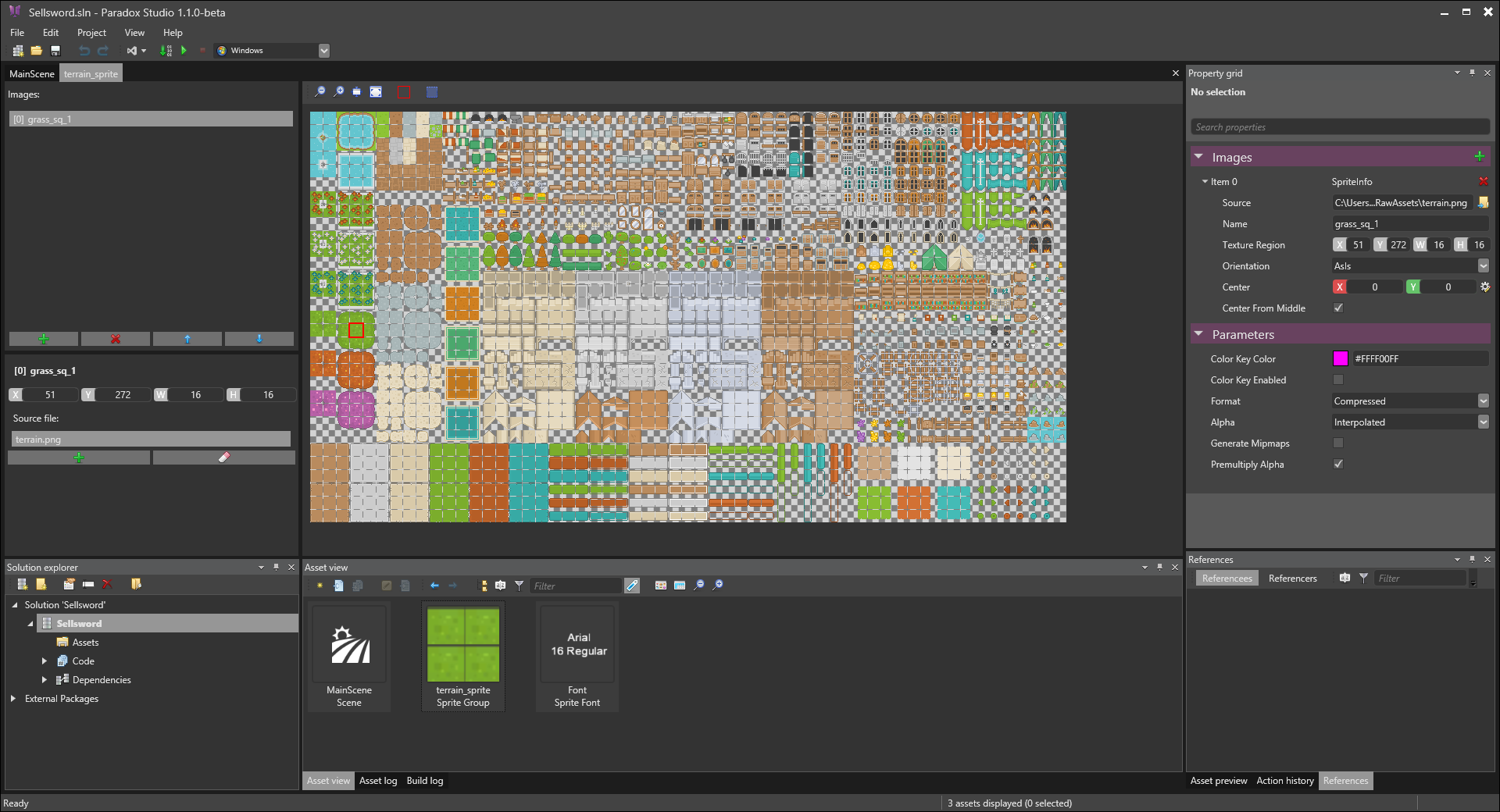
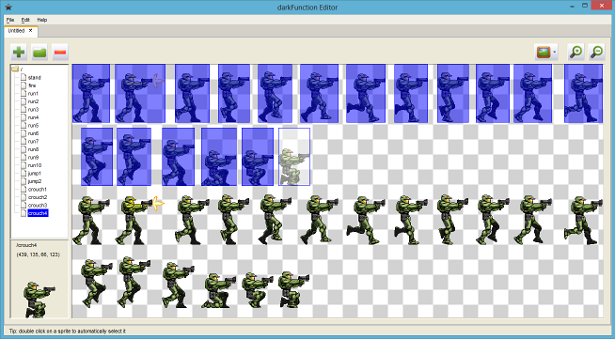
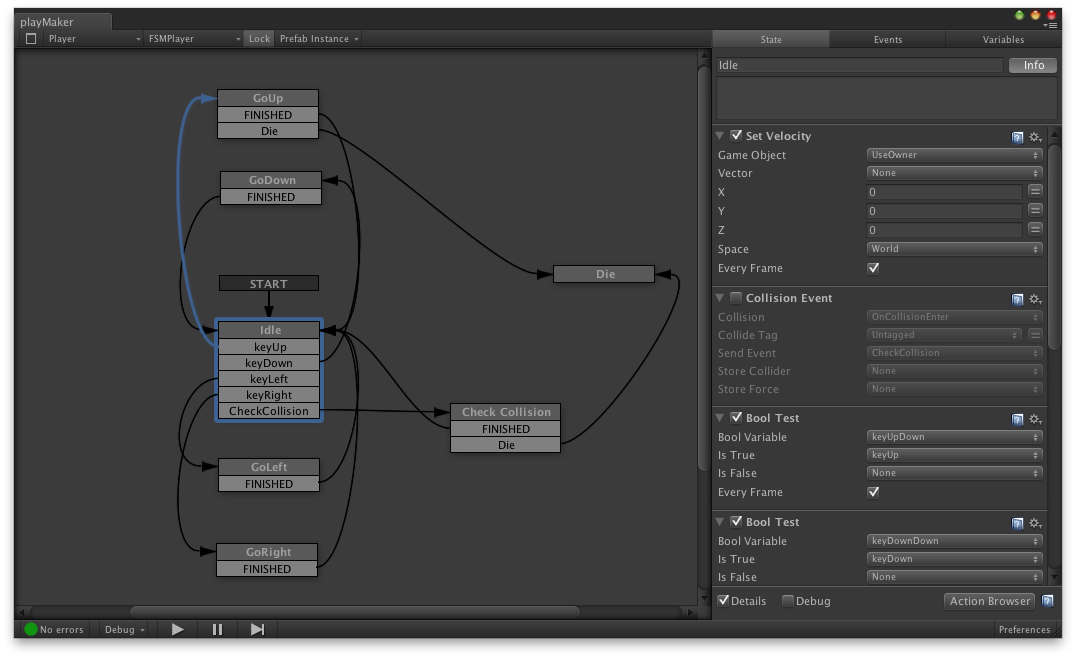


Figure 6. Texture Atlas Tool

Figure 7. Sprite Animation Editor

Another example may be a tool that allows the user to design the behaviour of artificial intelligence via a graph editor. For example a **Finite State Machine** could be created as a series of 2-d rectangles to represent each state, with lines connecting the states to each other representing transitions. A similar tool could be created for a **Behaviour Tree Editor**. Each would save the graph and the links between nodes / states within the graph.

See Figures 8 and 9 for examples.



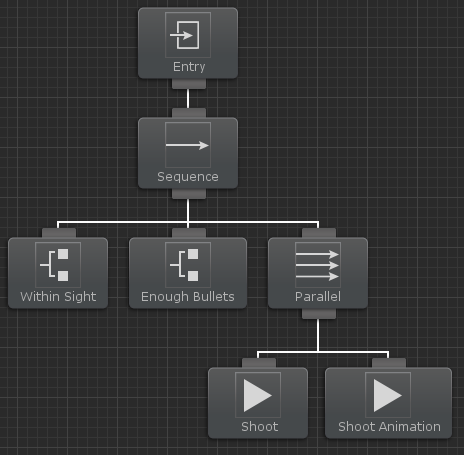
Figure 8. Finite State Machine editor.

Figure 9. Behaviour Tree Editor.

# Submission Process

All submissions must be made by the due date and time listed.

Students submitting an assessment after the due date and **without** an approved extension of time will lose 10% of the maximum available mark on the assessment per day it is late. In addition, assessments submitted late will only be awarded a maximum of a **pass** grade. This means that an assessment submitted 2 days late can achieve a mark of 80%, which contributes to the subject’s overall mark, but the assessment item itself will only receive a pass mark.

You are to submit a single zipped file via the Canvas link, or via an alternative approved method, that contains:

* A folder containing an executable run-time of your application that can be run external to any IDE. This folder must not contain more than is required for executing the runtime and must contain all required resources and assets.
* A folder containing the project source, **excluding temporary build files**, which can be successfully compiled by an assessor, without errors.

# Grading Rubric

Your assessment will be graded based on the grading rubric below and feedback will be given to the student as required.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Mark** | **High Distinction** | **Distinction** | **Credit** | **Pass** | **Fail** |
| **Proposal Document** | 20 | Multiple page proposal including multiple image mock-ups with clearly explained elements, and program UML diagrams or user guide. (**20-17**) | Multiple page proposal including multiple image mock-ups with design elements clearly explained. (**17-15**) | Multiple page proposal including 1 mock-up image with design elements clearly explained. (**15-13**) | 1 page proposal submitted briefly detailing application, including 1 mock-up image concept. (**13-10**) | Proposal does not meet minimum requirements. (**10-9**) |
| **Application** | 50 | Application uses advanced features of the API to implement a complex tool that exceeds requirements, including drag-n-drop features. (**50-42.5**) | Application meets minimum requirements and includes drag-n-drop features. Multiple example files included. (**42.5-37.5**) | Application runs as expected for minimum requirements. Source builds with no compile or link errors. Multiple example files included. (**38.5-32.5**) | Application runs as expected for minimum requirements. Source builds with no compile or link errors. (**32.5-25**) | Application does not fulfil the required features and may crash. Source may not build. (**25-0**) |
| **Source Code Quality** | 30 | Code written to a high standard and is fully modular, able to be reused in future projects, and is fully commented when explanation is needed. Code adheres to a coherent standard throughout the project. (**30-25.5**) | Code written to a high standard and is fully commented when explanation is needed. Code adheres to a coherent standard throughout the project. (**25.5-22.5**) | Code is neat and has few comments for obscure sections of code. Code mostly adheres to a standard. (**22.5-19.5**) | Code has few comments for obscure code sections. Code is neat and readable, but does not consistently follow a standard. (**19.5-15**) | Code is poorly written and follows no coding standard. Little to no commenting used. Obscure code has no comments describing functionality. (**15-0**) |