Ben Robinson

R020140g@student.staffs.ac.uk

Group Game Development Project and Work Based Simulation

Assessment Part 1

Games Engine Analysis

# Contents

[Contents 1](#_Toc42715393)

[1. Game Engine Overview 2](#_Toc42715394)

[2. Approach taken 2](#_Toc42715395)

[3. Text based tile demo 3](#_Toc42715396)

[4. Git Hub and how it assisted in the project 3](#_Toc42715397)

[Appendices 5](#_Toc42715398)

[Appendix 1: CSV’s in Excel 5](#_Toc42715399)

[Appendix 2: CSV’s in Text 6](#_Toc42715400)

[Appendix 3: Automatic integer reading 7](#_Toc42715401)

[Appendix 4: Automatic String Reading 8](#_Toc42715402)

[Appendix 5: Failure to read commas in excel 9](#_Toc42715403)

[Appendix 6: Git Hub Repositories 10](#_Toc42715404)

# Game Engine Overview

* 1. The game engine was created using OpenGL C++ with Simple DirectMedia Layer (SDL) and ImGui to assist certain aspects of implementation. The elements of the engine created described below enabled the other developers on the team to:
* Customise data, such as dialogue, enemies, players and attacks that the players could learn.
* Work on the tile based movement for the game engine, a basic grid loader was created that produced a set of tiles
  1. In addition a Git Hub organisation was created with five repositories. One for each member of the team and one for the game engine which the team worked on together to create.

# Approach taken

* 1. To create the element of the engine to customise data a set of Comma Separated Values (CSV) readers was created. CSV is a data structure commonly used within the software industry to move information between programs. However, they are also used in the games industry to easily customise important data within the game. This enabled the data within the engine to be customised, either through using an Excel spreadsheet, (see appendix 1) or notepad (see appendix: 2)
  2. The CSV reader will accept two types of variable: Integer and String. Integers can be read into the file easily, only requiring the number to be read in between two commas as a String, then having the read in value turned into an Integer using the “stoi()” function from the std library, see appendix 3. For Strings, punctuation marks such as commas or speech marks cannot normally be used. This is because they are both necessary for reading CSV files. To allow these punctuation marks within the CSV reader required it to start with the less-than symbol and to end with a more-than symbol. However, these symbols cannot be used within the CSV reader outside of declaring whether it is a String. The reading of these Strings are completed automatically using the functions set out in appendix 4.
  3. The CSV format was chosen as opposed to other file formats, such as the JSON format, because of the simplicity of the data being used within the program and the ease of access and implementation of the format. JSON files are used when it is necessary for the engine to read in larger and more complex data with differing data types. This was unnecessary for this engine due to all the areas only needing Strings and Integers. In addition, implementing a JSON file to a project requires the use of a JSON loader and editor, while CSV only needs Excel or Notepad.
  4. An issue with CSV is that some versions of Excel read them by using semi-colons instead of commas, see appendix 5. This can mean the CSV is unable to load into the program. To resolve this the prefix “sep=,” can be added to the top of the CSV in word to allow it to be read in Excel without any issues. The program then has the option to ignore this line while reading the CSV.

# Text based tile demo

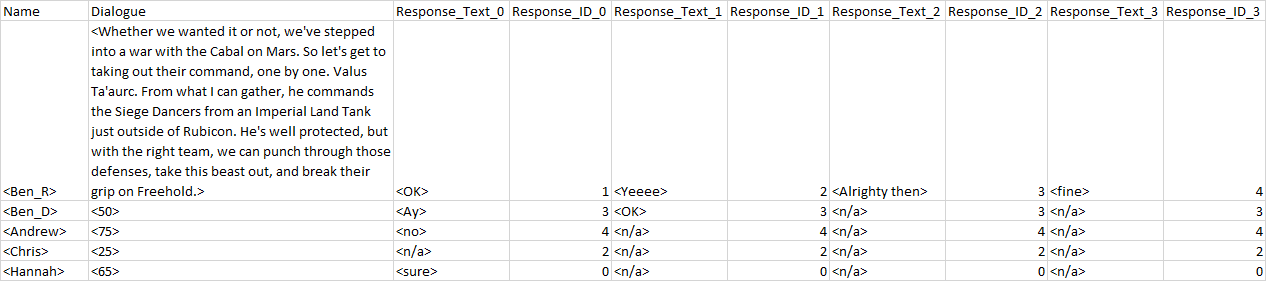
* 1. To assist a member of the team with work on the tile based movement for the game engine, a basic grid loader was created that produced a set of tiles. This reads out the details, including whether the player can move on the tile and if so whether the player is standing on it. This was done by creating a two by two vector array of tiles which contains its location within the grid and what type of tile is being stepped on. From here the player can move around the grid by inputting either the W, A, S or D key into the console. If the player tries to move outside the range of the grid, or onto a tile which cannot be moved to, the player will instead stay on the tile it currently occupies. This program was very basic and was used more to show how grid movement can be completed, rather than implemented into the engine. However, aspects of this prototype did end up in the final build of the engine.

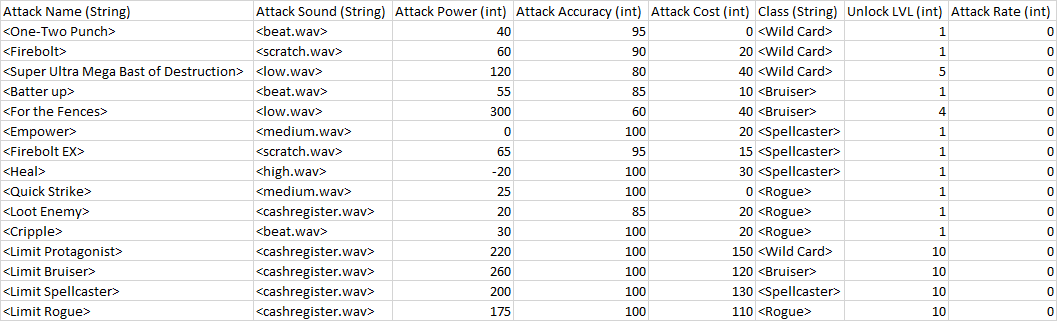
# Git Hub and how it assisted in the project

* 1. For the creation of the engine a Git Hub organisation was created with five repositories. One of each member of the team and one for the game engine. Programming and modification to documents were completed from Ben-Rs-repo and the feature/file\_io branch within the Engine repository, see appendix 6. The organisation assisted with development of the engine, enabling new work to be added to the engine without potentially overriding the work of other team members. If newer work was needed to be added to the engine, a pull request was made within the repository where it was be reviewed by all the team members before being merged into the engine.

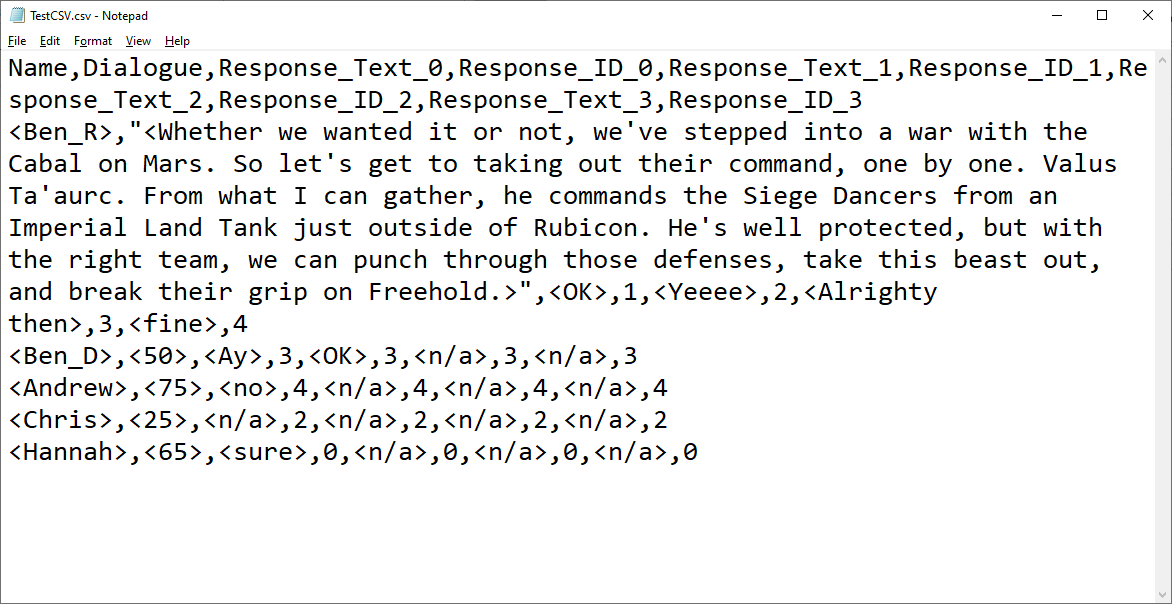
# Appendices

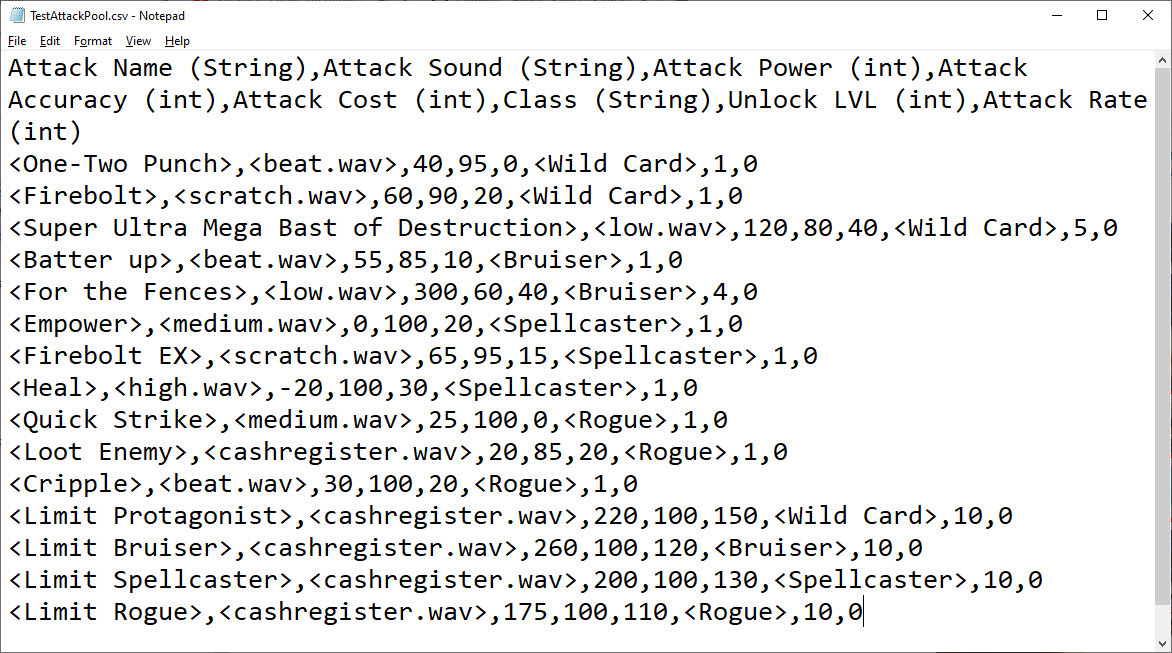
## Appendix 1: CSV’s in Excel



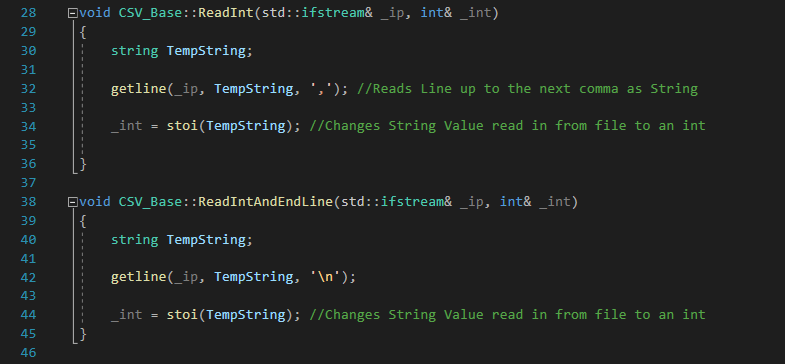


## Appendix 2: CSV’s in Text

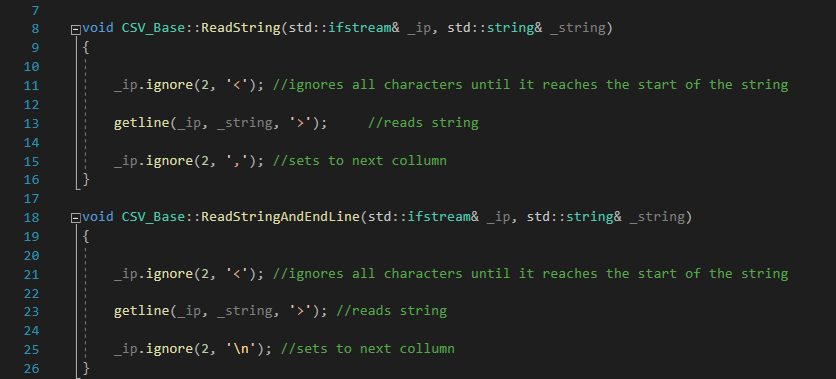




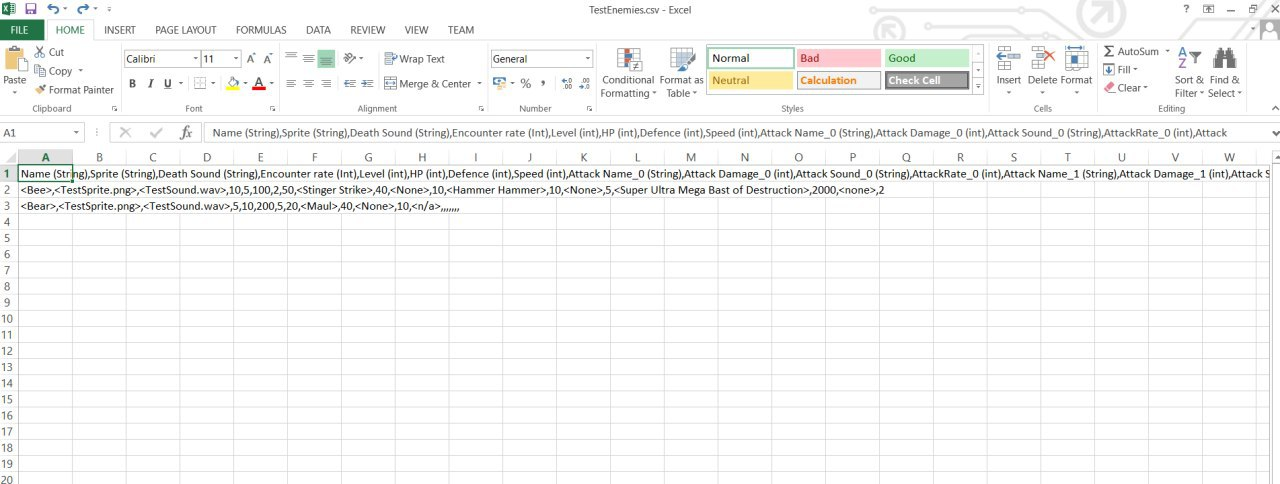
## Appendix 3: Automatic integer reading



## Appendix 4: Automatic String Reading



## Appendix 5: Failure to read commas in excel



## Appendix 6: Git Hub Repositories

