OCR GCE A  
COMPUTER SCIENCE  
PROJECT  
  
Name: Blake Horwill  
Candidate Number: 1908

Wyedean:   
Project Title: FarmGame

PROJECT CONTENTS

[Analysis: 3](#__RefHeading___Toc2029_912071140)

[Design 6](#__RefHeading___Toc2031_912071140)

# Analysis:

Problem identification:  
The problem that is faced by many students is an overstressed and overstimulated environment. This is coupled with a disturbing rise in computer illiteracy expressed by the younger generations.

To remedy this I have been tasked to design and build a game that increases computor literacy, introducing students to file navigation, editing text documents, image manipulation, writing code or writing markup with proper formatting, including varying granularity to the edit ability inside of the codebase, as well as providing a tool that can be useful in de-stressing during a lunch or break period, on top of that the UI will be graphical but will have aspects reminiscent of the terminal to acclimatise students to using text inputs over graphical interfaces as well as not to bore them.

The game style chosen will be farming-SIM due to its calming, distressing effect as-well as its easy implementation, allowing for easy editing and comprehension. This will decrease the iterative process of coding giving a greater feedback faster, allowing for the perception of faster progress and a greater retention of attention among students using it.

I have chosen to use computational methods to achieve my project as I believe the best way to understand and better utilise computors is to use them. I think it is considerably harder to implement learned ideas without a testing bed to experiment on, I can provide that by programming this game in a non compiled (interpreted) easily editable language.

Stakeholders:

This game is going to be an open source Farming-SIM, created for the purpose of teaching coding to students.

Because of this our stakeholders will be primarily students and teacher, two groups of people who will be using the software the most. For these two groups I need to have very-well documented and notated code as well as making it an easily approachable program utilising clean design and well as making an easy program to install and utalise.

Target Platform:  
For our target platform I would ideally make the program able to run on the pc with little consumer effort. A release on pc would be ideal due to the distribution among schools with most schools having at least one pc, another benefit is due to the nature of a pc, allowing for easy file management as well as being an ideal machine to edit code or use text/image editors on, being the machine of choice for offices and developers. Another aspect to consider is reliability as due to the variation of ecosystems among phone OS’s there is a vast way to develop for them, the same goes for consoles and other considerably harder / closed environments. This is in stark contrast to the PC with several languages designed to work on the widest range of OS’s possible, making development considerably easier and standardisation a wider spread norm. For our targeted OS ideally we would plan to build and test the computor program on both Linux and Windows as those are the two largest operating systems and lend themselves to computor development and programming. It would also be beneficial as Linux has a wide range of uses, being used on servers as well as raspberry PI’s making ideal for a learning experience due to its versatile nature.

Research:

For my research I have looked over many farming sims and educational projects to aid me in my development. I think that these would be the best to look into due to there relevance in my project with its crux being an open source editable farming sim that can be used to teach differing programming paradigms, or to teach general computer skills.

FarmVille:

Questionnaire:

If you do, what are the problems you face with technological illiteracy among youth?1

Do you see an increase of stress in the class room2

Do you believe that a project like the one propped would help in the action of “de-stressing” the student populous?3

1:(Teacher stakeholder):

QI:  
 Among my student I have notated a steep decrease in technical ability from my students, this fills me with dread as a computor science teacher.

Q2:

2:(Teacher stakeholder):

Q1:  
 I have not noticed a large decrease in

3:(Student stakeholder):

O1:

I have experienced a great deal of technical shortcoming with using desktop computors.

Q2:

I have generally seen an increase in stress among my peers especially recently.

O3:

I believe that a publicly available game like this would greatly aid in “de-stressing” me and my peers.

Feasibility Study:

Economic Feasibility:

This project is economical feasible due to it not using propitiatory software that is necessary to rent or own the rights too, meaning that it should be economically profitable relying only on donations.

Time Feasibility:

This project should be feasible to create within the given time period. However implementing a wide range of plant types and variations could be more time expensive than I can afford. Along with thorough documentation. However these features can be introduced later on through updates to the project and are not necessary initially.

Technical Feasibility:

This would be technically feasible due to the small amount of resources needed to create the project, only needing a simple IDE to code it in and only requiring a small complement of ram and processing power to run due to the light weight nature of the codebase.

Legal Feasibility:  
This project uses the GNU General Public License allowing for the studying and sharing of the software, this makes it ideal for teaching software, having the ability to be changed based on current needs.

Strengths&Weaknesses:  
For our strengths we have a clear idea and plan for the project as well as a wide stakeholder base to draw from, this will allow for easy iteration and feedback as well as a large user base if we decide to continue support on this project. However one downside I an for see is the financial incentive of the project being lacking due to the project relying on donations. A solution to this could be to add a pay wall, however I would be considerate on the addition of a form of transaction like this, due to it potentially alienating a large section of the people that we want to include (that being students).

Essential Features:  
Game UI:

the ability to see your farm and see it updated as well as to show contextual UI feedback, suck as .

Saving:  
The ability to save, to allow for multiple instances of multiple save files and the ability to manually change save data using a text editor or other means.

Limitations:

justification:

A wide variation of plants:

A wide base of plants would be useful, adding to the depth of the game and showing a wider range of examples for the crop tiles.

Limiting factor:

The limiting factor for this would be time constraints, heavily limiting the amount of assets that I can produce in the given time with each tile taking a long time to make and each plant requiring multiple tiles.

Justification:

An updating system to allow for updates of the code base:

An easy way to insert new code or tiles into the code base without requiring reinstalling the entire project over.

Limiting factor:

I do not have the technical ability to create an updating system to remotely update the files of the user-base.

Solution Requirements:

Development language:  
For the development language I have chosen to use python due to its flexibility, its ability to run off of a large amount of devices. It also has several other benefits, one being that python is an interpreted language, this allows for fast iteration on code and aids in the goal of this project being used as a teaching tool. Another aspect is that python is an easy language to learn, making it ideal for teaching coding concepts like OOP and functional programming to students.

Deployment:

For the deployment of my program I will create a public repository hosted on a file sharing website such as GitHub, I will not be hosting the project from private servers due to the cost and time investment in building up the infrastructure around a server system, as well as my lack of knowledge in server development.

Success Criteria:  
Text input:  
A text input that allows for commands to be executed, editing states inside of the game.

Justification:

This allows for the editing of the farm state and the several arrays there within.

Reference:  
Terminal

GUI:

A graphical user-face to display the current game state and to show relevant contextual updates such as whether the game is being saved or whether the farm is being watered.

Justification:

This allows for the user to not only get text feedback upon an action, but visual feedback too up engagement with the project and to generate a more ergonomic experience, being inherently more intuitive to a person coming from contemporary gaming experiences.

Reference:  
FarmVille

Saving:  
The game should have the ability to save and dump its data to an external file that can be edited or duplicated.

Justification:  
This allows for a greater user experience and also shows how data can be saved and moved/read from in an intuitive manor.

Adoption rate:  
The program should at least have a 1/3 adoption rate for students or a 1/3 approval rate from teachers among our beta testers.

Justification:  
This shows that the product has a place among our consumers and would work well in the public market, we can also teak our product with feedback provided by the beta testers to increase our adoption rate to meet the threshold.

# Design

Decomposition:

GUI:

When the game starts tiles are called and displayed on the screen based on the users save data, these tiles are updated every frame with there reference being stored inside of a multi dimensional array.

Saving/Dumping save\_data:  
When the save data is dumped/saved it is converted to/from the json format, saving into a file specified by the settings file.

Options:  
For the options (settings) the settings will be read during the start of the program from an external settings file labelled “settings.json”. This will then provide information such as the location and name of the save file and the tick speed.

Game-cycle:

The game will run on a main loop where when ever the cycle reaches that of the specified tick update it will update all of the crops.

Commands:  
There will be a switch statement to call subfunctions when certain commands and arguments are inputted into the text input box.

Game Start:

Upon starting the game all of the assets utilised during game play will be called and initialised. After this the main loop will be called allowing for the game to play indefinitely.

Quit:

This will be a statement checked every frame for when the quit button is pressed, once it is pressed the main loop will cancelled and the initialised assets will be destroyed.

Structure:

Start game

load

objects

Quit

End

TickTrue

Update

Farm

command