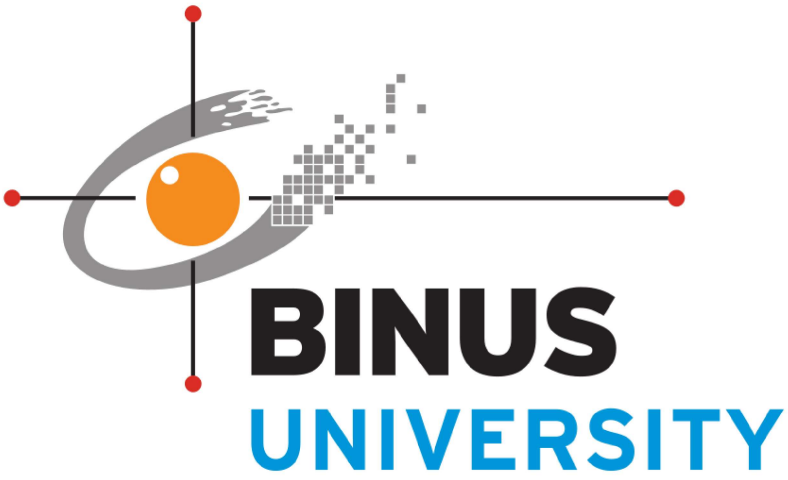
Cover Page



Making a Text Based Game Using Python

Lecturer:
D4017 - Jude Joseph Lamug Martinez, MCS

Course:
COMP6047001 - Algorithm & Programming

Arranged by:
2702364963 - Registan

Table of Content

Chapter 1: Brief Project Description

Introduction

This project uses the Python programming language, the project is based on the classic game “Battleship”. Since it is text-based it does not use Pygame or any other libraries that are required to render graphics, instead the graphics are made using letters and spaces.

Initial Idea Inspiration

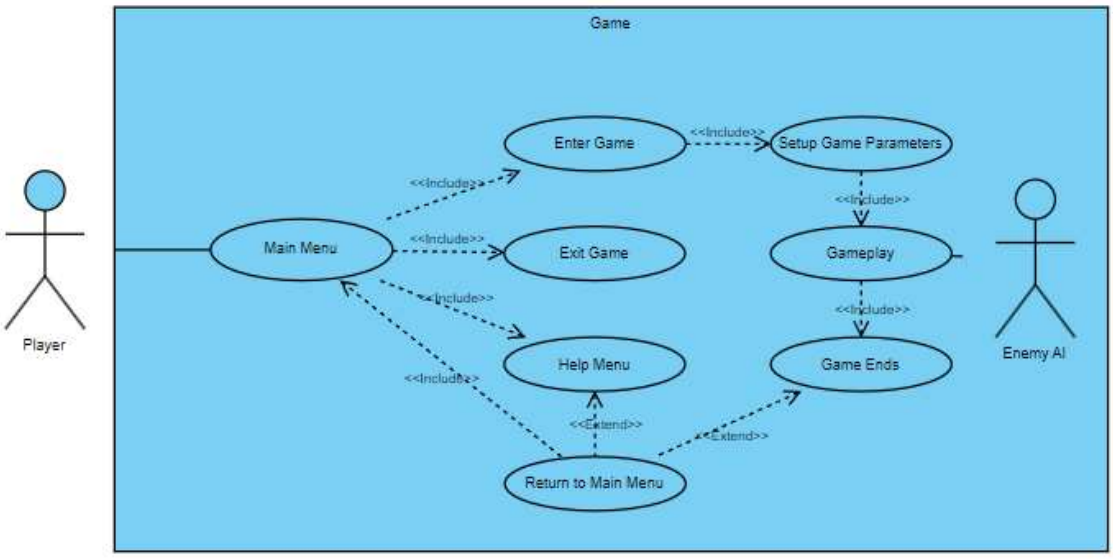
At first I was trying to make a 2d platform using the Pygame library, but it seemed like adding any new features would break previous features and I did not understand how to fix them. So I decided that I would do a text-based game since it was simpler and I did not have to use Pygame. The only 2 ideas that came to mind was a text adventure and battleship but using text, I chose battleship since it seemed simpler and I would not have to make up a story. I also changed up the idea a bit, instead of just being the same as battleship I would call it “Artillery Commander” and instead of having ships I would have artillery pieces which are the same as ships but they take singular grids instead of multiple.

How It Works

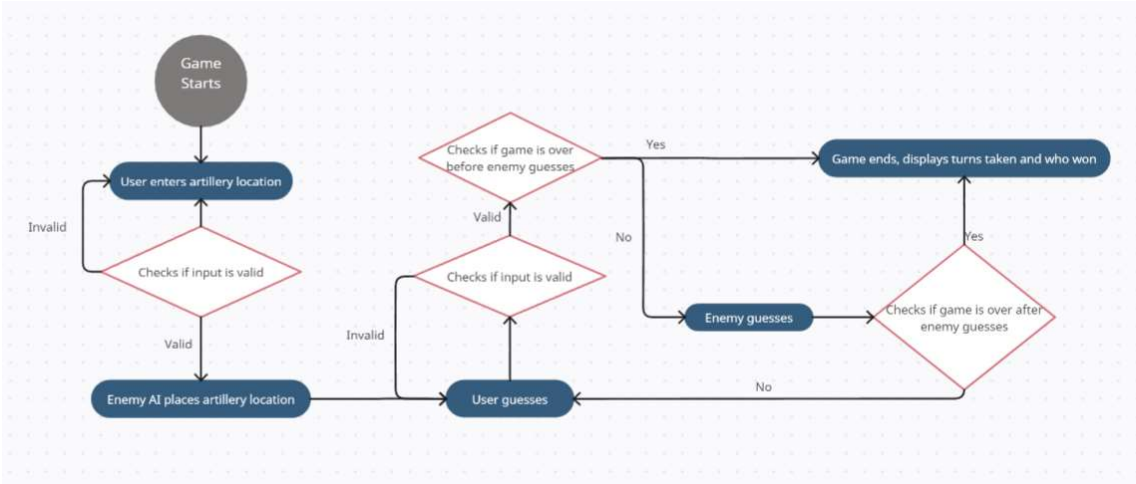
This project heavily utilises the print() and random.randinit function, print() critical as it is used to print the menus and game board, the game board are 2 different coordinate grids (one for the player and the other for the enemy). These coordinate grids are square and can be 1x1 to 9x9 grids, the size can be changed during the game parameter setup (happens when “1” is entered in the main menu) where it asks for a grid size.The random.randint function is only used for enemy

player, since implementing a proper enemy AI would be hard. It is used during the placement phase (after the game parameter setup) the enemy player chooses random values that are within the grid coordinate range to place artillery pieces and guess the location of the player's artillery pieces.

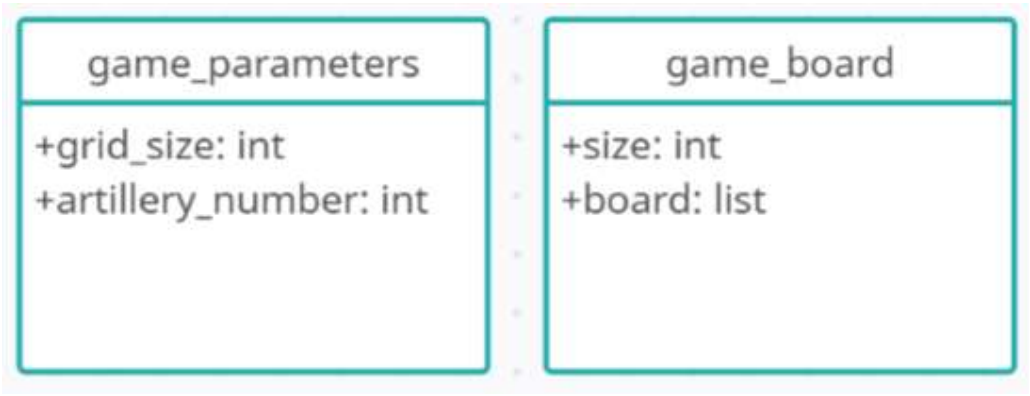
Chapter 2: Use Case Diagram



Chapter 3: Activity Diagram



Chapter 4: Class Diagram



Two classes was used in this project:

1. game_parameters
    - Used to contain the current game parameters that were setup by the player.
    - Has the attribute grid_size and artillery_number, they are both integer values.The grid_size attribute contains the length of one side of the square grid, while artillery_number contains the number of artillery the user and enemy has to place down after the game parameter setup phase.
2. game_board
    - Stores the data that involves the game board grid.
    - Has 2 attributes; size and board. The size attribute just like the grid_size attribute in game_parameter contains the length of one side of the square grid. The board attribute contains a list with the rows of the square grid, these rows are strings containing the letter “O”.

Chapter 5: Modules Used

The module used in this project is shown below:

```
1   #Imports
2   import random
```

Only the “**random**” module is used in this project, “**random**” as the name says, contains functions that generate random numbers or choices. This project only uses the function “**randinit**” to generate a random number within range, this module is used for the enemy AI which chooses random spots in the grid to place artillery pieces after the player has set up the game parameters. It also uses the “**randinit**” function to guess the locations of the player.

Chapter 6: Essential Algorithms

What is the Essential Algorithm

The essential algorithm in this Python project is the "**game_logic**" algorithm; it contains a lot of the gameplay logic. This is the function that is called when the game is started (After game parameter setup), it utilises many functions to simplify the gameplay loop and streamline it.

What it does

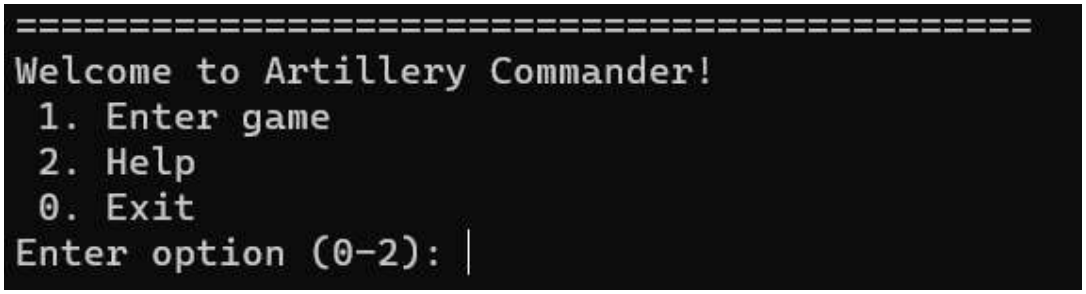
When the "**game_logic**" function is called, it assigns 2 variables the "**game_board**" class; player_board and enemy_board. These contain the grid size attribute and the board attribute. After this it calls the "**place_artillery**" and "**enemy_place_artillery**" functions, the first function (place_artillery) prompts the player to choose a grid coordinate to place their artillery pieces. The second function chooses random grid coordinates to place the artillery pieces for the enemy. It then assigns the integer value "**0**" to a variable called "**match_turns**".

After that, it starts the main gameplay loop where at the start of every loop, the "**match_turns**" variable is added by 1. It then prints the player and enemy boards, after that it asks the player for a coordinate guess on the enemy's board with the "**player_guess**" function and proceeds to check it with the "**check_guess**" function. If the check guess returns "True", a conditional statement subtracts an artillery piece from the enemy and tells the player that they have destroyed an enemy artillery piece. If "False" is returned, then the conditional statement shows the enemy artillery pieces remaining. After this, it calls the "**is_game_over**" function to check whether the game is over or not. If the game is over a conditional statement will tell who won, the turns taken from the "**match_turns**" variable and return to the main menu. After this it is the enemy's turn to guess, it chooses a random grid coordinate and uses the "**check_guess**" function again. After this it checks if the game is over again, this is done twice so that it can be checked at every turn since the enemy could still win even though its correct guess was afterwards. If the game is not over after this, it will return to the start of the loop until the game is over.

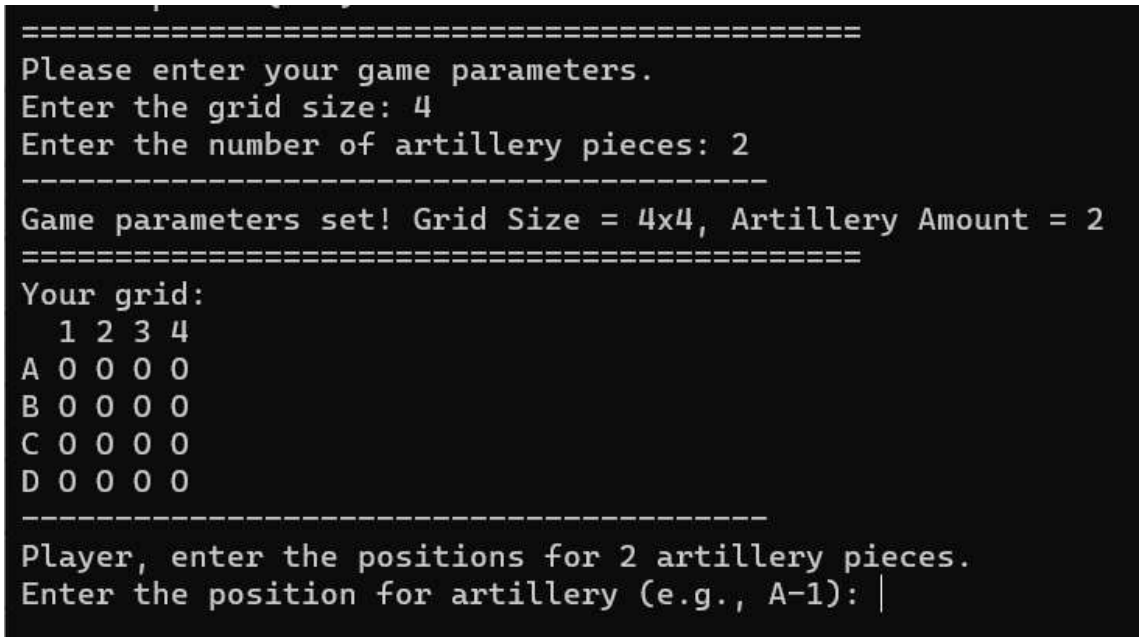
Chapter 7: Screenshots of The Project

---

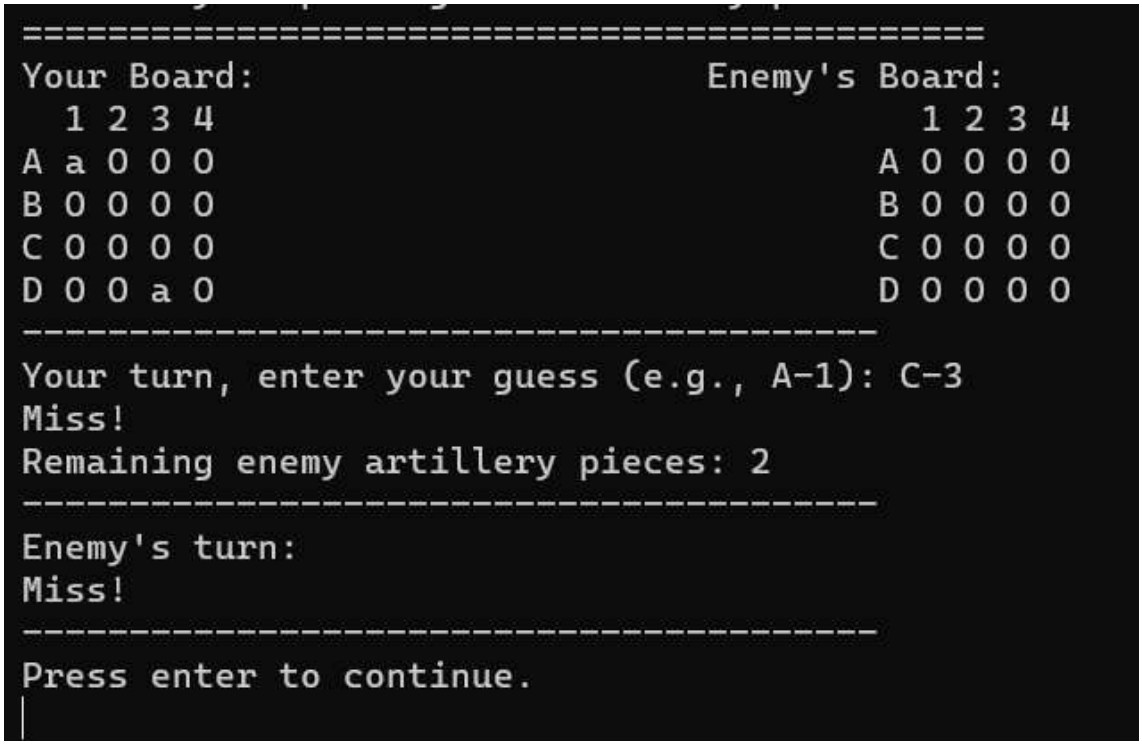
1. The main menu of the game



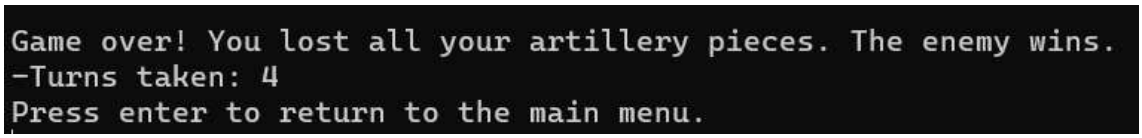
2. The setup and placement menu for the game



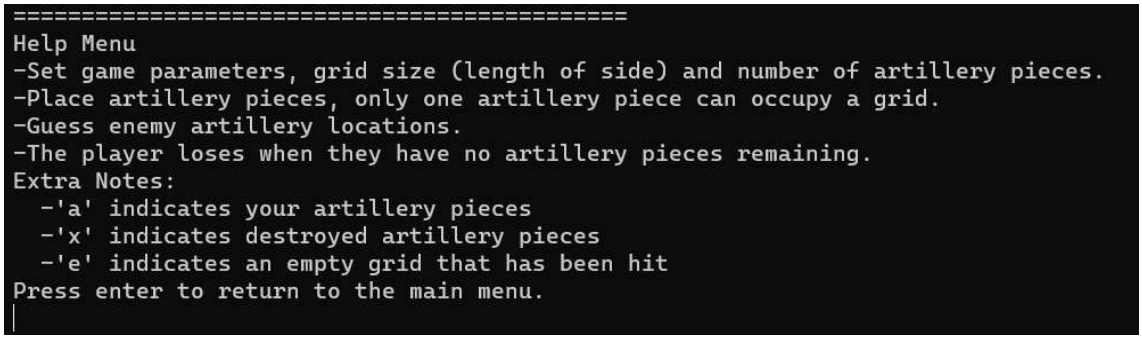
3. The main game loop



4. The game over prompt



5. The help menu



Chapter 8: Things Learnt

After working on this project, I learnt a few new things in Python:

- The utilisation of the random module: I found out that the random module can be very useful to make a simple enemy AI, even though it just chooses random numbers it can be convincing especially in this situation as all the player is doing is just guessing random squares with no real strategy.

- The usage of classes: I learnt that using classes is important even for making a simple project like mine, it can come in very useful especially for storing some data to make it simpler and less visible variables needed.

Reflection

After looking back at the making of the project, I found myself quite content with what I have made, although I could have made something that is not text-based that uses Pygame or some other library to render graphics, I feel like the concept of a text-based game is neat. I found out about the limitations of text-based game, for example; real-time games are not possible since you cannot just press a button but you need to type something then input it, this adds a delay which ruins the experience of a real-time game.

Chapter 9: Source Code and Example Video

Github Repo Link: https://github.com/Regis-tan/A-P-Final-Project
Example Video Link: