

PYTHON PROJECT

Topic : Battleship using Graphic User Interface(GUI)

Group Members: Shobhit Behl (IMT2016024)

Gautami Gupta (IMT2016069)

Siddarth Reddy Desu (IMT2016037)

Explanation of the game: Battleship is game that can be played by a single player or by two players. It is played on a 10*10 grid. There are five ships to be set. The condition of setting the ships is that minimum length of a ship can be two i.e. a player can select minimum two adjacent blocks in order to place a ship. The maximum length of a ship can be five i.e. a player can select maximum five adjacent blocks in order to place a ship. If it is a 2 player game then initially, player 1 places his five ships by selecting blocks and same goes with player 2. Now, the screen is reset and the ships placed are hidden. The game starts and the each player gets a chance to open a single block of opposite partner. This way each player opens each others blocks and aims at opening/blasting every ship of the opponent. The player who first blasts all the ships of the opponent is declared as a winner.

Features of our game:

- Our code runs for both single as well as for two players. User will get an option to choose how many player game he/she wants to play.
- In a two player game, both the players will get the chance to place their ships as per choice by selecting the blocks on the

screen by a simple mouse click. Any wrong way of selecting the blocks for ships will lead to an error.

- Each player can open only one block at a time and will get a chance alternately. The player who destroys all the ships of the opponent first will be declared as winner.

Concepts used in our game code:

- Graphical User Interface(GUI) through pygame and tkinter ,
- User defined classes and objects(Object Oriented Programming),
- Functions, lists, tuples, dictionaries, iterative loops like while and for and basic use of variables.

Use of GUI: To make Battleship, we are using Pygame and Tkinter.

- Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.
- Whereas, Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk. Tkinter is not the only GUI Programming toolkit for Python. It is however the most commonly used one.
- We are using pygame to set up the battleship game screen/board/layout where the game will be played. Also we are using pygame for certain operations required in the game like making the grid or selecting blocks in a grid. We are also using Tkinter so as display certain messages we wish to deliver to the player while playing the game.

Use of Object Oriented Programming: Our aim to use this concept was to make the understanding of the program easier and also since by

the use of classes and objects we can certain small portions of our program rather than making a big program, therefore debugging will be easier. We have defined certain classes and have imported the classes and its methods where ever required.

Execution of the code(DESIGN):

Single player game:










- Display screen:
 - ✚ We have imported pygame module and have used it to set up a screen as per our design.
- Grid:
 - ✚ We have set up a single 10x10 grid on the screen using pygame.
- Infinite while loop:
 - ✚ In order to make the screen display continuously on the computer screen we have to run an infinite while loop which is the main loop containing the major part of our code. This loop will terminate only when the game has ended.
- Setting the ships:
 - ✚ We will be importing random and randint module to set up the ships randomly for the player to play.
 - ✚ The blocks selected randomly will be stored in a list in order to check in future, whether the block hit is missed or attacked.

- **Game play:**
 - ✚ In single player game the number of blocks a player can open is fixed to 30 moves. This will make the game even more interesting.
 - ✚ The game starts with the player selecting one block at a time.
 - ✚ If the block contains the ship it will be opened and marked and stored in a list.
 - ✚ If a player selects the block that has already being selected then no change will take place.
 - ✚ If a player selects a block that does not contains a ship it will be marked as missed and its coordinates will be stored in a separate list.
- **Game exit:**
 - ✚ In the given limit of game i.e. 30, if the player is able to complete the game i.e. if he is able to explore all the ships then he wins.
 - ✚ If the player is not able to find all the ships in 30 moves then the player loses the game.

Two player game:

- **Display screen:**
 - ✚ We have imported the pygame module and have used it features to design a display screen where the game will be played.
 - ✚ We have set an adequate size of the screen, together with certain features specifying player 1 area and player 2 area with good use different colours.

- ✚ Players can place their ships by just clicking on the block they want to and the selected block will take up different image indicating that the block is now selected.
- Infinite while loop:
 - ✚ In order to make the screen display continuously on the computer screen we have to run an infinite while loop which is the main loop containing the major part of our code. This loop will terminate only when the game has ended.
- Grid:
 - ✚ A 10x10 grid will be set up using pygame modules. If user selects a single player game then only one grid will be visible.
 - ✚ If a user selects 2 player game then two 10x10 grids will set up with the indication given above, which grid belongs to which player.
- Message Boxes using Tkinter:
 - ✚ A class is defined for such message boxes which contains methods with different messages in it.
 - ✚ All through this while we have used pygame but now here we import a tkinter module so as to display certain messages and instruction regarding the game which we want to convey. Like if its player 1's chance to select his/her particular ships will be shown as pop-up and the player has to follow the instructions accordingly.
- Selecting The ships:

-  We have defined a module ships which returns the ship which is presently being selected by the player among the five ships, they are destroyer, submarine, battleship, cruiser and carrier.
 -  We have imported the ships module in our main code.
 -  Now each player selects his/her ships following the rules by just clicking on the box he wishes to select.
 -  Once selected it cannot be undone. Each player gets a chance to select his/her five ships with the condition that minimum length of the ship can be 2 adjacent blocks and maximum can be 5 adjacent blocks.
 -  One cannot select diagonal blocks. One has to select only adjacent blocks so as to indicate a single ship.
- Checks module:
 -  We have a user defined module named 'checks' which checks certain conditions of selecting the blocks in order to place the five ships. i.e. `check_if_in_box()`
 -  This module checks the situation in which if a player selects the diagonal block it will be not be treated as a valid choice. i.e. `check_if_diagonal()`
 -  This module also checks the adjacent blocks of the selected block so as to make a valid selection. i.e. `check_if_valid()`
 -  This module also checks the orientation of the ships placed, i.e. whether it is horizontally placed or vertically. i.e. `check_if_vertical()` , `check_if_horizontal()`

- **Error if blocks not selected properly:**
 - ✚ If a player does not follow the rules of selecting blocks as mentioned before then errors will occur. Eg. if a player selects more than 5 adjacent blocks then the error will be shown that “ship entry exceeds its limit” , and the player has to select that particular ship again.
- **Storing the blocks containing ships:**
 - ✚ The blocks a player has selected is stored in a list. Starting coordinates of each block are stored in a list for future reference.
- **Game Starts:**
 - ✚ We have defined a new class for this purpose which is imported in the main program.
 - ✚ The screen is reset and now the players can target each other's area to find the ships.
 - ✚ If the player clicks on the block which does not occupy the ship which will be checked by checking its top left coordinate with that present in the list which stores the coordinates of the blocks occupying the ships. They will be marked with another colour which indicates the open and missed block.
 - ✚ If a player clicks on the block having the ship by checking it in the list then that block is coloured with different colour indicating open and attacked.
 - ✚ The blocks opened will be again stored in a list and this list is checked every time a block is clicked so that clicking on this block does not make any difference.

- **Player's chance:**

- ✚ Players play alternately chance by chance and they are allowed to open only one block at a time which is checked by a counter variable.
- ✚ If the counter variable is one then that player won't get another chance and the chance will drift towards another player.

- **Checks module:**

- ✚ The checks module defined above also contains functions which checks whether the block selected while playing is a attacked block or a missed block and does task according to it. i.e. `check_if_attacked()`
- ✚ Also, this module checks the condition when the game is over i.e. when all the ships of a player sinks. When the game ends the screen is closed. i.e. `check_if_win()`

- **Game result:**

- ✚ Every time a player selects a block and if that block contains a ship then that particular coordinate is removed from the list containing the blocks with ship coordinated and a `[0,0]` sub list is appended. This list is checked every time a player selects a block.
- ✚ In case of 2 players, there are 2 such lists which stores the ship coordinates of both the player.

- ✚ As the opponent keeps on selecting blocks containing ships , that particular coordinate keeps on getting removed and [0,0] sub lists are appended.
- ✚ Once the first element of the list is [0,0] we stop the game and the player whose first element is not [0,0] i.e player still has certain ships which are not blasted.

*****END*****