Navy Chess Game Final Report

COMP 8046

Xinghua Wei – A00978597
10-5-2021

Table of Contents

1. Introduction

1.1. Student Background

I am a student enrolled in the BCIT CST Btech program major in the network security application development option. I graduated from the BCIT CST diploma program Database option.

During the BCIT CST diploma program, I had one year of programming experience with JAVA and some programming experience with C, C++, JavaScript, HTML5 and MySQL. I have accomplished a few website applications, such as a website to encourage users to save food and a website that supports selling products online. I also had an opportunity to create an Android mobile application that indicates crime in the current user area.

During the ongoing network security application development option, I learned some basic ideas and functionalities behind TCP/IP. Also, I learned how to create simple TCP/IP communication applications, a simple firewall using Shell Script. Now I am learning ideas and methods to do ethical hacking and hiding data while transmitting data. Project Description

Education

British Columbia Institute of Technology

Program: Computer Systems Technology	2016-2018

- Graduate from CST Database option

Program: Computer Network Security Application Development	Ongoing

Skills

- Network Security
  Understand basic ideas about network security, network applications using Python, Shell Script on the Linux platform. Understand basic socket, TCP/IP communication using Python to implement some IPv4 client/server applications.

- Website
  Implement and design websites with HTML5 and JavaScript. I have completed five team projects successfully. Also, I have programming experience using Bootstrap.

- Database

  I have half-year experience with Oracle, T-SQL, MySQL, Visual studio, and data mining with different software such as Hive and R studio.

1.2. Project Description

I propose to create a multiplayer chess game call Navy Chess, a two-player board game. There are no existing games on any platforms since it is an old-fashion board game to my current knowledge and research. I want to create a digital version of Navy Chess for Windows platforms where users can download it and connect to other players via the players' IP address and port using Python socket.

Besides the game itself, a live chat function with encrypt data transmit underneath that will provide a safer and more reliable gaming environment. To let the signal player enjoy the game, Two different AI levels for this game with Alpha-beta pruning and Minimax allow the signal player to start a match without waiting for another player. In terms of player connections, there will be a two-factor authentication applied to the game that whenever play

A wants to connect to player B, there will be a request to authenticate it via an external email system.

1.2.1. Essential Problems

For this project, I tried to digitalize a board chess game into a multiplayer game and there are some essential problems I try to solve:

- There are 50 chess units in the game. The variables of each chess object are essential since they need to move, attack, hide and be destroyed.
- I need to solve how the Alpha-beta pruning and Minimax algorithms can fit this chess game model.
- I need to set up a third-party SMTP service to build a two-factor authentication process based on that.
- I need to embed RSA encryption to the chat function.

1.2.2. Goals and Objectives

Overall, this project has four main goals:

- To develop a fully functional digitalized 2D Navy Chess game for the Windows platform using Python.
- To research and develop a single-player mode with Alpha-beta pruning and Minimax.
- To develop a secure live chat with a data transmitting function to enhance the gamer's gaming environment by using end-to-end encryption.
- Two-factor authentication function to secure player register, login and communicate.

2. Body

2.1. Background

Navy Chess is a two-player board game, and it was derived from another board game SanJun Chess. To my current knowledge and research, both games are no longer online on any platform since it is an old-fashion board game, and most people treat it as antique.

The main combat units include aircraft carriers, battleships, torpedoes (same as all types of ships), mine blasters (lightning strike ships), and submarines (the only ships that can defeat all types of ships are defeated by lightning strike ships). The secondary combat units include three missile boats (which can knock down Aircraft carriers), torpedo boats (which can knock down submarines), mines (to protect military flags at sea), and three minesweepers (which can knock down mines). In addition, because there are two anti-submarine aircraft groups (anti-submarine aircraft and bombers) walking on the chessboard with black dotted lines (courses), Navy chess has produced the concept of three-dimensional space combat because of the submarine, anti-submarine aircraft, and bomber group.

2.2. Project Statement

The problem is that the Navy Chess Game people can play their child games online without searching for such games in local stores. It has several modes suck as dark mode, normal multiplayer mode, and singal player mode, which solves the different needs of different players. The data encryption and the Two-factor authentication functions solve the potential user credential loss and could enlighten any children who could play the game with email verification.

2.3. Possible Alternative Solutions

There are some possible alternative solutions I can use for this project.

- I could use Java to build this program and GUI. Java is a high-level, object-oriented programing language, and it is generally faster compared to Python. Object-oriented means that it is suitable for a project that needs to create dozens of objects. Besides, Java can create GUI applications, but a little bit more complex than Python, in my opinion.
- I could use JavaScript to build this program and GUI. JavaScript is also a high-level, object-oriented language suitable for building applications that need a GUI.
- I could build this program by Android Studio. Android Studio is the official integrated development environment for Google's Android operating system. It is easy to create GUI by Android Studio.
- Instead of Minimax and the Alpha-beta pruning, I could build a database and use the method called Opening Move Database because many combinations of the first few initial moves on the board are known to establish a good position. The program can use this database at the beginning of play. Instead of performing the usual local search, it simply checks whether the moves performed so far fit into a given opening strategy, and then plays the appropriate response.
- I could also use machine learning to train the AI to play. Machine learning is the study of computer algorithms that can automatically improve through experience by using data.

- I could use DES, AES or other ciphers like substitution, transposition. Ciphers like substitution are easier to achieve. DES and AES are similar to RSA, and AES is the most commonly used encryption method.
- I could use Qt for Python GUI design. Qt for Python is another popular GUI toolkit. It is extracted from the library of Qt. PyQt is the product of the combination of the Python language and the Qt library. PyQt is coming with the Qt Builder. We will use it to get the python code from the Qt Creator. With the support of a qt designer, we can build a GUI, and then we can get python code for that GUI. PyQt supports all platforms, including Windows, macOS and UNIX. PyQt can be used to create stylish GUIs, a modern and portable python framework.
- I could use another third-party SMTP service like Google, Amazon. SMTP, Simple Mail Transfer Protocol, is the industry standard to send emails on the internet. It uses proper authentication, which increases the chances that your emails will actually get delivered to the users' inbox.

2.4. Chosen Solution

Although there are many alternatives I could use in terms of language choice and technologies, I still want to use the current setup.

- Python is an interpreted high-level programming language. It is designed to have more excellent readability, and it is also object-oriented. Python is easier to read and use than Java, and Python has tons of libraries that I do not need to start from scratches, such as cipher library and standard GUI library. JavaScript is not very suitable for build windows applications compared to Python. It works better if I want to build a web application. Android Studio is designed to build mobile applications, and my purpose is to build a desktop application. Besides, I am more familiar with Python than Java or other languages. Therefore, I choose Python to build this project.

- I choose to use Minimax and Alpha-beta pruning because Minimax is the most popular searching algorithm for chess games. Minimax is suitable for chess games like this project which does not need to search too deep. Opening Move Database requires a large database that better have hundreds or thousands of openings and following movement stored, and it does not worth the time to build such large database for this new game manually. Machine learning is actually a good alternative choice, but I have no experience building such a learning model for an AI, and there is a high possibility for me not able to finish this part of the program in this project. Therefore, Minimax is achievable, robust and feasible to use in a new chess game.
- For the data encryption, I choose to use RSA encryption. First of all, ciphers like substitution are too easy to decode. Also, I do not want to use symmetric encryption, which uses the same secret key to encrypt and decrypt information. Asymmetric encryption uses a pair of keys to encrypt information, and it is more secure than symmetric encryption. AES, Advanced Encryption Standard, is an asymmetric encryption algorithm that encrypts fixed blocks of data. Although AES is more popular, RSA works better when transmitting data between two separate endpoints, and it fits the game.
- I choose to use Tencent SMTP service mainly because I have no use of Tencent account, so I do not need to worry about account security when setting up the SMTP service.

Also, the Tencent SMTP service is easier to use than it only requires an account. Unlike Google SMTP, which requires allowing "less secure" apps access in order for my SMTP settings to work.

2.5. Details of Design and Development

2.5.1 Game Rules Setup

The chart below shows the settings of each chess unit of the game.

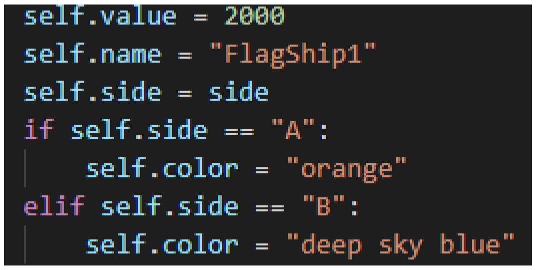
Defence	**Attack**									
	Aircraft carriers	**Battleship**	**cruiser**	**Destroyers**	**Strikers**	**Mine blasters**	**Submarines**	**Torpedo**	**Mine**	**Flagship**
Aircraft carriers	Equal	Defence Win	DefenceWin	DefenceWin	DefenceWin	DefenceWin	AttackWin	Equal	—	DefenceWin
Battleship	AttackWin	Equal	DefenceWin	DefenceWin	DefenceWin	DefenceWin	AttackWin	Equal	—	DefenceWin
cruiser	AttackWin	AttackWin	Equal	DefenceWin	DefenceWin	DefenceWin	AttackWin	Equal	—	DefenceWin
Destroyers	AttackWin	AttackWin	AttackWin	Equal	DefenceWin	DefenceWin	AttackWin	Equal	—	DefenceWin

Strikers	AttackWin	AttackWin	AttackWin	AttackWin	Equal	DefenceWin	DefenceWin	Equal	—	DefenceWin
Mine blasters	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	Equal	AttackWin	Equal	—	DefenceWin
Submarines	DefenceWin	DefenceWin	DefenceWin	DefenceWin	AttackWin	DefenceWin	Equal	Equal	—	DefenceWin
Torpedo	Equal	Equal	Equal	Equal	Equal	Equal	Equal	Equal	—	Equal
Mine	Equal	Equal	Equal	Equal	Equal	AttackWin	Equal	Equal	—	Equal
Flagship	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	Equal	—	Equal

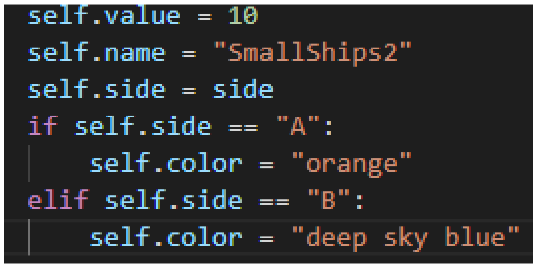
2.5.2 MiniMax Algorithm Design Idea

The miniMax algorithm will be used for this game. It is a search algorithm that allows the program to look ahead at possible future positions before deciding what move it wants to make in the current position. Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. It is widely used in two player turn-based games like chess games.

In Minimax, the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible, while the minimizer tries to do the opposite and get the lowest score possible. Every board state has a value associated with it. In a given state, if the maximizer has the upper hand then, the score of the board will tend to be some positive value. If the minimizer has the upper hand in that board state, it will tend to be negative. To determine the value of the board, I give different values to different types of chess units. For example, the Flagship has the highest value on board, and a small ship only has a value of 10. AI, the maximizer, will try to calculate the maximum onboard value based on chess values on board to decide the next movement. The image blows give an example of different chess units with different values. The "self.value" variable will be used to maximize board values by AI.



```
self.value = 2000
self.name = "FlagShip1"
self.side = side
if self.side == "A":
    self.color = "orange"
elif self.side == "B":
    self.color = "deep sky blue"
```



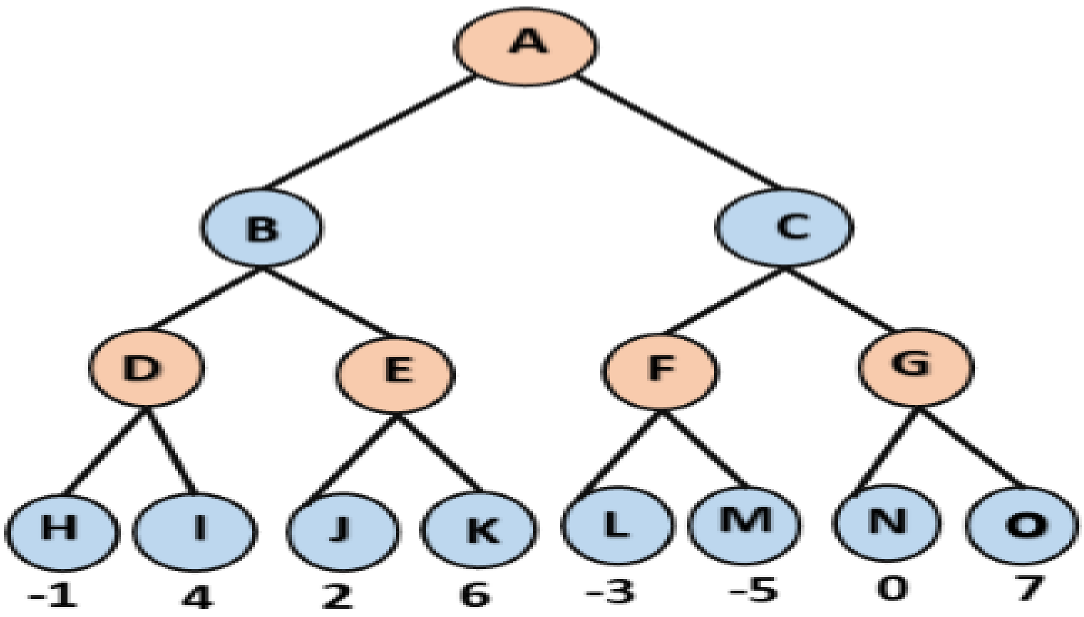
```
self.value = 10
self.name = "SmallShips2"
self.side = side
if self.side == "A":
    self.color = "orange"
elif self.side == "B":
    self.color = "deep sky blue"
```

But MiniMax itself is kind of slow. The searching and deciding process could be optimized by Alpha-beta pruning.

2.5.3 Alpha-Beta Pruning Design Idea

Alpha-beta pruning is a modified version of the miniMax algorithm. It is an optimization technique that will fast the miniMax process. With Alpha-beta pruning, the program will not need to check each node of the game to compute the correct decision. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree, which need not be searched because a better move is already available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely Alpha and Beta.

For example, there are two players that one is called Max, and the other is called Min. Each terminal node represents a chess unit, and each node has a value. The red node represents Max, and the blue node represents Min. Max will try to get the Maximum score, where Min will try to get the minimum possible score. Also, the Alpha will be added and means negative infinite. Beta means positive infinite.



Step 1:

Max will start the move from node A that Alpha is negative infinity, and Beta is positive infinity. The values will be passed down to nodes B and D.

At node D, the value of node H and I are compared. Max will get the maximum value:

- For D: max(-1:4) = 4

Step 2:

It is Min's turn that Min will compare the beta value with the current node B value at node B. Min will get the minimum value:

For B: min(4, positive infinite) = 4

Step 3:

At node E, Max will take its turn, and the value of Alpha will change. The current value of Alpha will be compared with 5,

- For alpha: max(-∞, 5) = 5,

Therefore, at node E, Alpha is 5 and Beta is 3, where Alpha is larger than Beta. So the right successor of node E will be pruned, and the algorithm will not traverse it, and the value at node E will be 5.

This is an example of how a node will be pruned. It will still use the chess unit value, such as that Flagships have a value of 2000 and a small ship has a value of 10. Then the Alpha-beta algorithm will apply the same methodology to the other nodes. In the real game, Alpha represents the highest value that the maximizer can presently promise at that level or higher, whereas Beta represents the best value that the minimizer can currently guarantee at that level or higher. The Alpha will be set to a big negative number, whereas the Beta will be set to a large positive value in this project. The AI will discover the highest score on the board, and if the following movement can create more scores than the highest score on the board, the AI will record it. The AI will then assign the Alpha the greatest value between the alpha value and the current best score on board. This node will be the optimum movement to act if the Alpha score has now surpassed the Beta score.

2.5.4 End-to-End Encrpytion

End to end encryption is the process in which encryption of data is being done at the end host. It is an implementation of asymmetric encryption.

There are two types of encryption, one is symmetric encryption, and the other is asymmetric encryption. I will use asymmetric encryption for this project. In Asymmetric encryption, two types of keys are used, one public key and one private key. The sender and the receiver both have the public key and the private key. The Public Keys are available to both sender and receiver to share public keys before communication starts. The sender uses the receiver's public key to encrypt the messages to be sent, and the receiver uses both its public and private keys to decrypt the messages. The private key to the receiver is available only with the receiver and no one else.

RSA algorithm will be used for this project. RSA algorithm is an asymmetric encryption algorithm. RSA requires a public key and a private key and works as follow:

1. Person A sends the public key to person B and requests some data
2. Person B encrypts the data using person A's public key and sends the encrypted data
3. Person A receives the data and decrypts it.

2.5.5 Two-Factor Authenticaiton

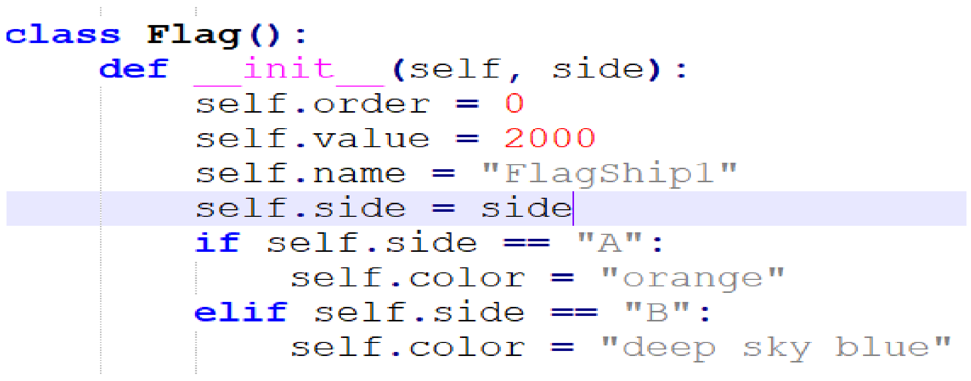
Two-factor authentication is an electronic authentication method in which a device user is granted access to a website or application only after successfully presenting two or more pieces of evidence to an authentication mechanism.

The client-side player will need to use a valid email account to connect to the server-side player for this project. First, the server will use a proper SMTP service provided by a third-party organization. Then, one or two players will go through a setup process that first configures his account to send his email a code by text. At the end of the setup, they will have a randomly generated authorization code to their email. Once done, the player will login with the registered email and receive an authentication code in the email when he wants to join the game.

2.6. Detail Designs

2.6.1 Chess Unit

There are 50 chess units in a single game. Each unit has its own purpose and value. Therefore, each chess unit will be created as different classes. Every chess unit will be assigned an "order" variable representing its rank value; a "value" variable representing how much score can one player earn if the player eliminates the unit; a "side" variable to different if a specific unit is at player A's side or player B's side. Same units on different sides will show in different colours. For example, the image below shows what is inside a Flag class:

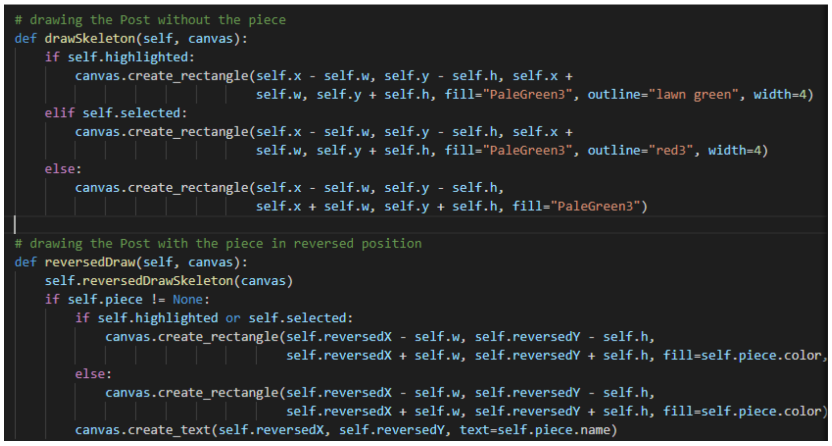


```
class Flag():
    def __init__(self, side):
        self.order = 0
        self.value = 2000
        self.name = "FlagShip1"
        self.side = side
        if self.side == "A":
            self.color = "orange"
        elif self.side == "B":
            self.color = "deep sky blue"
```

A flag is a target to win the game. The opponent will win the game if he destroys my flag. The flag cannot move or attack, so its order is 0. It can directly let a player win the game, and it has a maximum value among all the chess units, so its value is 2000. 2000 score value in a single game is unbeatable; therefore, the opponent will win. The sides variable shows that if a flag is at A's side, it will appear in orange colour. Otherwise, it will appear in blue at B's side.

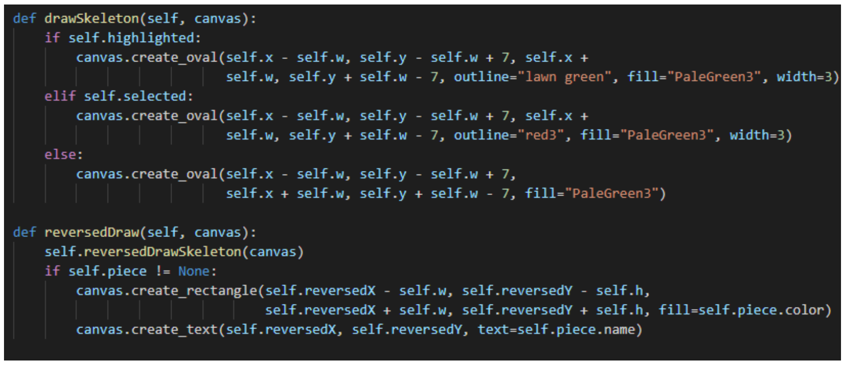
2.6.2 Map Objects

The game map overall is a 5 * 12 grid. It will be separated into two 6 * 5 grids for each player. Each side will contain 25 chess units and 5 camps. Chess units are designed to a class object, Post class, which is generally separated into two steps. For example, drawSkeleton function and reverseDraw will draw player A's side and Player's B side. That is for the convenience of drawing different groups of chess units on the board.



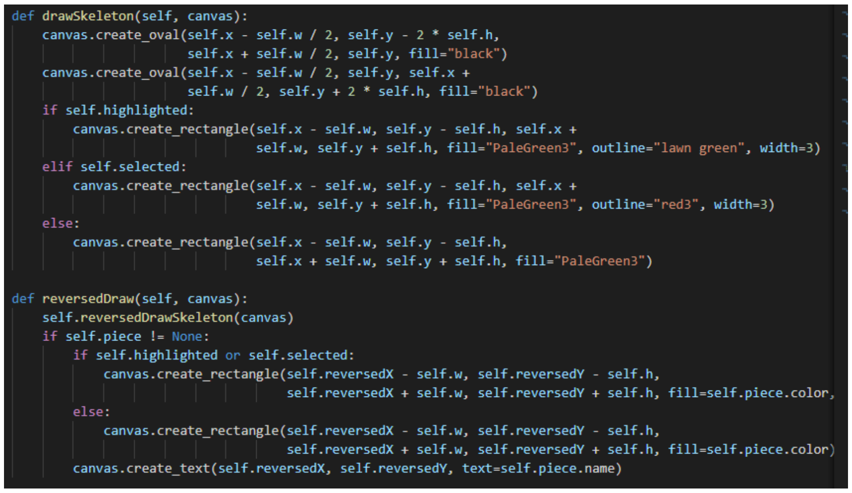
```python
# drawing the Post without the piece
def drawSkeleton(self, canvas):
    if self.highlighted:
        canvas.create_rectangle(self.x - self.w, self.y - self.h, self.x +
                                self.w, self.y + self.h, fill="PaleGreen3", outline="lawn green", width=4)
    elif self.selected:
        canvas.create_rectangle(self.x - self.w, self.y - self.h, self.x +
                                self.w, self.y + self.h, fill="PaleGreen3", outline="red3", width=4)
    else:
        canvas.create_rectangle(self.x - self.w, self.y - self.h,
                                self.x + self.w, self.y + self.h, fill="PaleGreen3")

# drawing the Post with the piece in reversed position
def reversedDraw(self, canvas):
    self.reversedDrawSkeleton(canvas)
    if self.piece != None:
        if self.highlighted or self.selected:
            canvas.create_rectangle(self.reversedX - self.w, self.reversedY - self.h,
                                    self.reversedX + self.w, self.reversedY + self.h, fill=self.piece.color,
        else:
            canvas.create_rectangle(self.reversedX - self.w, self.reversedY - self.h,
                                    self.reversedX + self.w, self.reversedY + self.h, fill=self.piece.color)
        canvas.create_text(self.reversedX, self.reversedY, text=self.piece.name)
```

Camps are designed for players to protect any movable chess units from being destroyed by the enemy. Camps are designed to a class object as well, and they inherit the chess units class, which is Post class. It will also draw all camps on players A's side and B's side separately.



```python
def drawSkeleton(self, canvas):
    if self.highlighted:
        canvas.create_oval(self.x - self.w, self.y - self.w + 7, self.x +
                           self.w, self.y + self.w - 7, outline="lawn green", fill="PaleGreen3", width=3)
    elif self.selected:
        canvas.create_oval(self.x - self.w, self.y - self.w + 7, self.x +
                           self.w, self.y + self.w - 7, outline="red3", fill="PaleGreen3", width=3)
    else:
        canvas.create_oval(self.x - self.w, self.y - self.w + 7,
                           self.x + self.w, self.y + self.w - 7, fill="PaleGreen3")

def reversedDraw(self, canvas):
    self.reversedDrawSkeleton(canvas)
    if self.piece != None:
        canvas.create_rectangle(self.reversedX - self.w, self.reversedY - self.h,
                                self.reversedX + self.w, self.reversedY + self.h, fill=self.piece.color)
        canvas.create_text(self.reversedX, self.reversedY, text=self.piece.name)
```

Also, the headquarters class is created for flagships to stay to represent a target to win the game. It also inherits the Post class and draws headquarters separately.

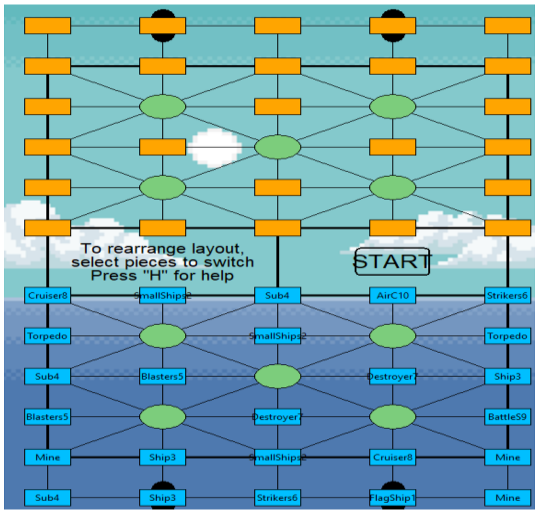


```python
def drawSkeleton(self, canvas):
    canvas.create_oval(self.x - self.w / 2, self.y - 2 * self.h,
                       self.x + self.w / 2, self.y, fill="black")
    canvas.create_oval(self.x - self.w / 2, self.y, self.x +
                       self.w / 2, self.y + 2 * self.h, fill="black")
    if self.highlighted:
        canvas.create_rectangle(self.x - self.w, self.y - self.h, self.x +
                                self.w, self.y + self.h, fill="PaleGreen3", outline="lawn green", width=3)
    elif self.selected:
        canvas.create_rectangle(self.x - self.w, self.y - self.h, self.x +
                                self.w, self.y + self.h, fill="PaleGreen3", outline="red3", width=3)
    else:
        canvas.create_rectangle(self.x - self.w, self.y - self.h,
                                self.x + self.w, self.y + self.h, fill="PaleGreen3")

def reversedDraw(self, canvas):
    self.reversedDrawSkeleton(canvas)
    if self.piece != None:
        if self.highlighted or self.selected:
            canvas.create_rectangle(self.reversedX - self.w, self.reversedY - self.h,
                                    self.reversedX + self.w, self.reversedY + self.h, fill=self.piece.color,
        else:
            canvas.create_rectangle(self.reversedX - self.w, self.reversedY - self.h,
                                    self.reversedX + self.w, self.reversedY + self.h, fill=self.piece.color)
        canvas.create_text(self.reversedX, self.reversedY, text=self.piece.name)
```

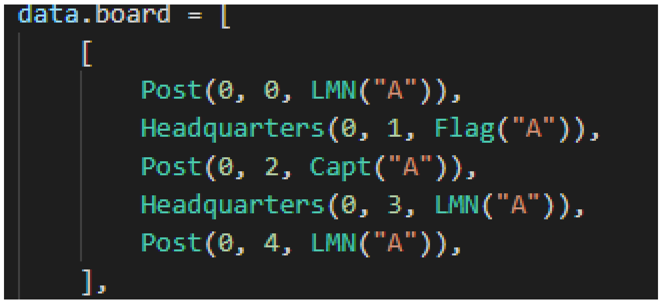
2.6.3 Map Board

To draw the map board that units can move through the board, I choose Python Tkinter to accomplish that. Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's standard GUI. Also, the Tkinter package is the standard Python interface to the Tk GUI toolkit, and it is available on most Unix platforms and Windows systems.

There are 5 vertical grids and 12 horizontal grids; therefore, a 5 * 12 grids in total, including 50 chess units, 10 camps and 4 headquarters. Headquarters will also be a location to put flagship and chess units at the beginning of the game, so, visually, 60 objects are on the map board. The image below shows the game board and 60 objects.

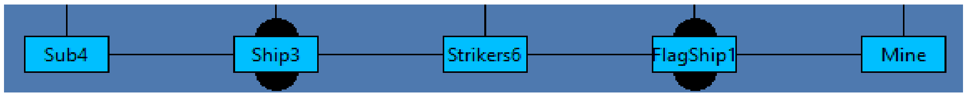


To draw all the objects, I will create all objects layer by layer. For example, I will draw the first layer at the bottom. Therefore, a Post object will be created first, then a Headquarters object, followed by another Post object, the second Headquarters object, and the third Post object. The image below shows that each layer contains 5 objects and will be store in a list.



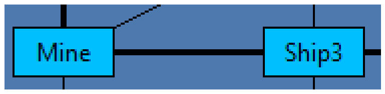
```
data.board = [
    [
        Post(0, 0, LMN("A")),
        Headquarters(0, 1, Flag("A")),
        Post(0, 2, Capt("A")),
        Headquarters(0, 3, LMN("A")),
        Post(0, 4, LMN("A")),
    ],
```

This means the first layer of players A's side is created, as in the image below.



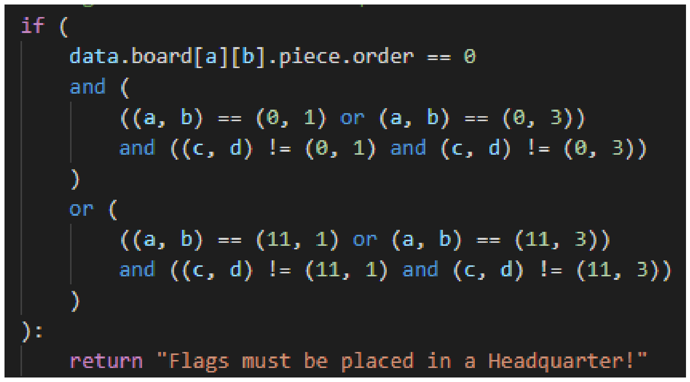
After drawing all the objects, objects need to be connected with lines. The lines are created with create_line function from the Tkinter library. It allows me to draw lines between two coordinates. For example,

`canvas.create_line(50, 110, 50, 690, width=3)` allows me to draw a wider line from location (50, 110) to location (50, 690). Overall, the board is designed to be a 600 * 800 space. After drawing the lines, objects will appear to be connected as in the image below.



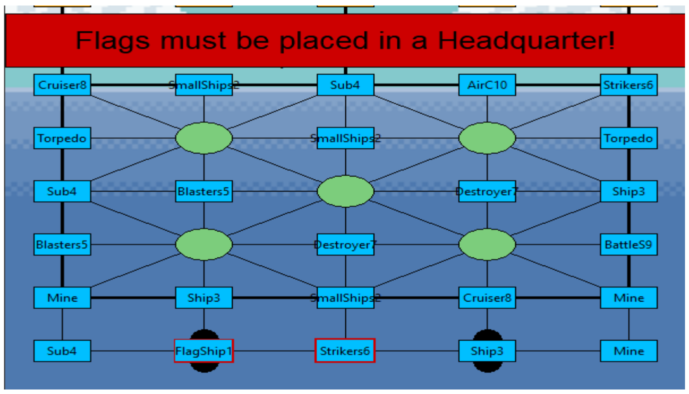
2.6.4 Units Functionalities

To make sure some chess units are in the proper location, I will create a check function to check if some chess units are in the correct location. For example, flagships must be put in one of the headquarters. Therefore, in this 5 * 12 grids board, I need to check the soundings of the headquarters and the flagships in its layer. The Flagship has an order of 0, so I will check the headquarters if they contain a rank order of 0. If not, an error message will be showed. The image below gives an example of flagships location checking.

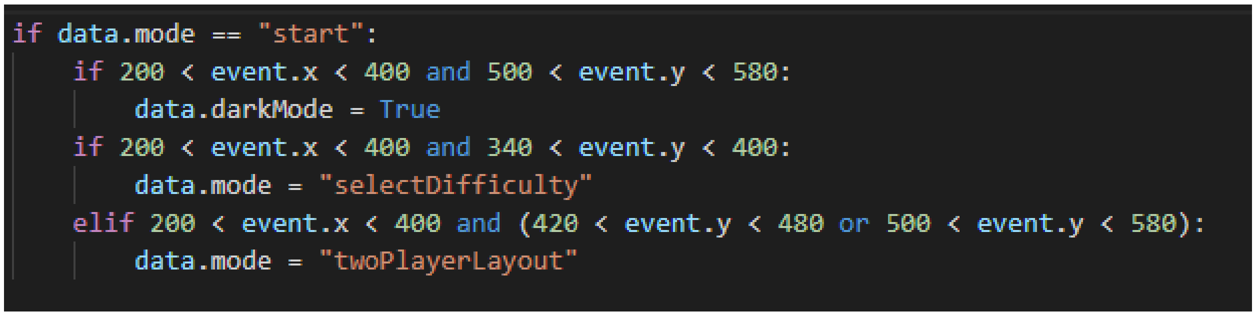


```
if (
    data.board[a][b].piece.order == 0
    and (
        ((a, b) == (0, 1) or (a, b) == (0, 3))
        and ((c, d) != (0, 1) and (c, d) != (0, 3))
    )
    or (
        ((a, b) == (11, 1) or (a, b) == (11, 3))
        and ((c, d) != (11, 1) and (c, d) != (11, 3))
    )
):
    return "Flags must be placed in a Headquarter!"
```

The image below shows that I was trying to put one of the flagships out of the headquarters (Movement show as red rectangles), and an error message appeared.

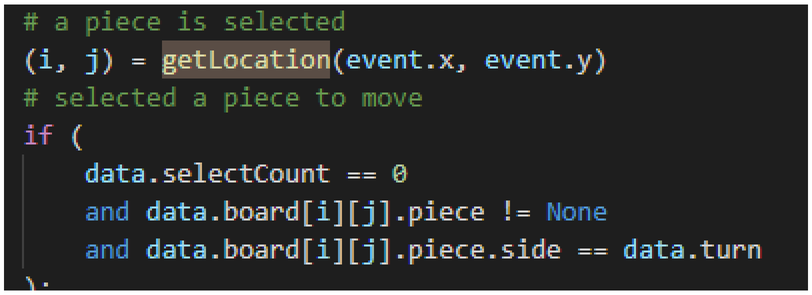


To track players' mouse click activity, I have a mousePressed function to react to different clicks. I will divide the click button into different 2D areas. For example, the landing page will show several different buttons. Each button will have a react area. The image below shows an example. If the click happens in the (200,500) and (400,580) area, which creates a rectangle area, a certain mode flag will be triggered.



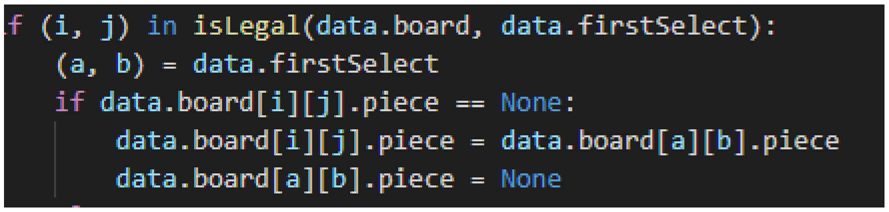
```
if data.mode == "start":
    if 200 < event.x < 400 and 500 < event.y < 580:
        data.darkMode = True
    if 200 < event.x < 400 and 340 < event.y < 400:
        data.mode = "selectDifficulty"
    elif 200 < event.x < 400 and (420 < event.y < 480 or 500 < event.y < 580):
        data.mode = "twoPlayerLayout"
```

To move a chess unit, players need to click a chess unit first. Once a player clicks a rectangle area of a chess unit, the chess will be highlighted in a green rectangle. Then the getLocation function will be called first to record its location on the game board. If the move does not violate the game rules, the player can select a second spot to move to. The second spot will also be recorded its location first; then, the first selected chess unit will replace the second selected chess unit location.



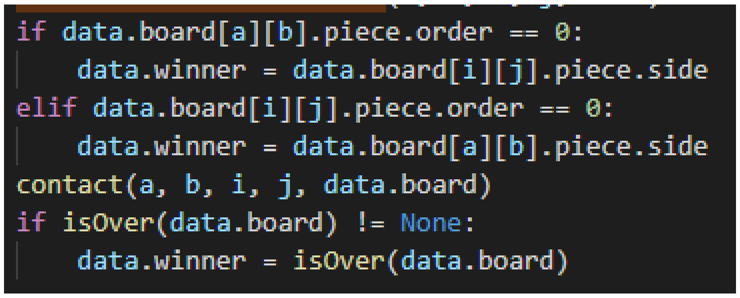
```
# a piece is selected
(i, j) = getLocation(event.x, event.y)
# selected a piece to move
if (
    data.selectCount == 0
    and data.board[i][j].piece != None
    and data.board[i][j].piece.side == data.turn
```

The image above shows that a chess unit is selected, and the select count will be set to 0. After selecting the second spot, the select count will be set to 1. If the movement passes the game rule check (pass the isLegal function check), the movement will take action.

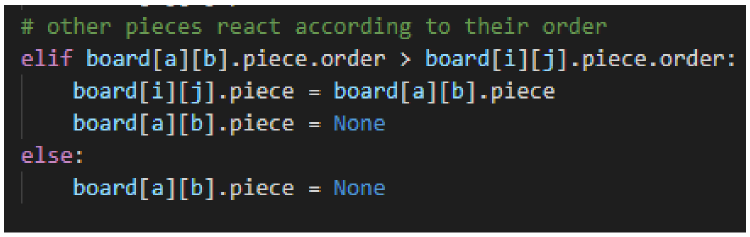


```
f (i, j) in isLegal(data.board, data.firstSelect):
   (a, b) = data.firstSelect
   if data.board[i][j].piece == None:
       data.board[i][j].piece = data.board[a][b].piece
       data.board[a][b].piece = None
```

If the chess unit attacks (move to) an opponent's chess unit, the program will check if the game-winning condition is met or not first. That is, if player A's chess unit is at the location of player B's flagship location, player A wins. The following image explains that if a chess unit appears at the location of the opponent's chess unit of the rank order of 0, that player wins. If the winning condition is not met, call the contact function for regular chess unit contact.



```
if data.board[a][b].piece.order == 0:
    data.winner = data.board[i][j].piece.side
elif data.board[i][j].piece.order == 0:
    data.winner = data.board[a][b].piece.side
contact(a, b, i, j, data.board)
if isOver(data.board) != None:
    data.winner = isOver(data.board)
```



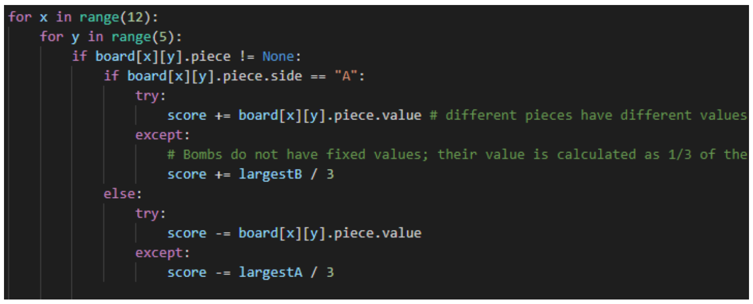
```
# other pieces react according to their order
elif board[a][b].piece.order > board[i][j].piece.order:
    board[i][j].piece = board[a][b].piece
    board[a][b].piece = None
else:
    board[a][b].piece = None
```

The image above shows that each chess unit has a rank order, and an order 9 unit will destroy an order 8 unit, as I mentioned before. This program checks the order of two contacted units and limits the lower order one, and the winning chess unit will replace the location of the lost one.

2.6.5 AI Design

For this project, the Minimax algorithm and the alpha-beta pruning will be used for AI to allow users to play single-player mode. The miniMax algorithm will be used for this game. It is a search algorithm that allows the program to look ahead at possible future positions before deciding what move it wants to make in the current position.

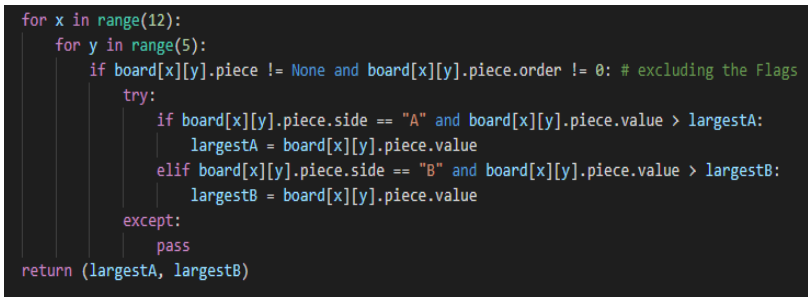
For example, there are two players that one is called Max, and the other is called Min. AI will seek to maximize the board score where player move seeks to minimize the board score. To achieve that, the program will need to calculate board scores. As I mentioned before, each chess unit has a value that the higher-ordered unit will have a higher score value.



```
for x in range(12):
    for y in range(5):
        if board[x][y].piece != None:
            if board[x][y].piece.side == "A":
                try:
                    score += board[x][y].piece.value # different pieces have different values
                except:
                    # Bombs do not have fixed values; their value is calculated as 1/3 of the
                    score += largestB / 3
            else:
                try:
                    score -= board[x][y].piece.value
                except:
                    score -= largestA / 3
```

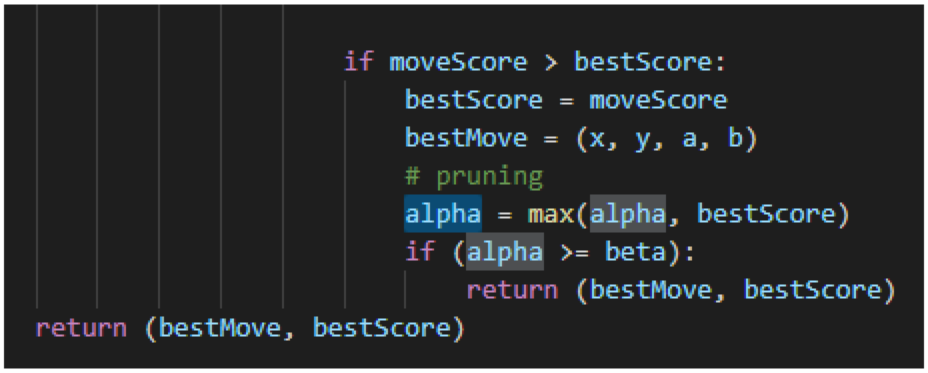
The above image shows that the board score will be added if a player has taken a unit. If the player lost a unit, the score would be minus according to the unit value.

Also, to use the algorithms, I need to get the largest unit value on the current board. The image below shows that the program will scan the entire board and find the largest existing unit value because that will be the top priority target for the AI.



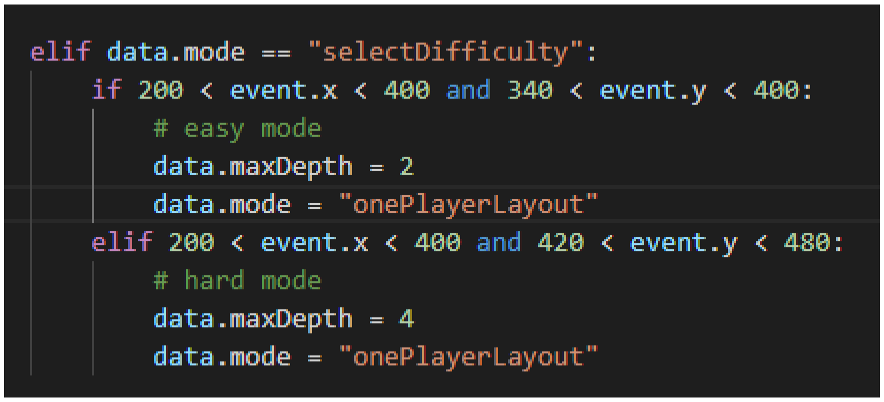
```
for x in range(12):
    for y in range(5):
        if board[x][y].piece != None and board[x][y].piece.order != 0: # excluding the Flags
            try:
                if board[x][y].piece.side == "A" and board[x][y].piece.value > largestA:
                    largestA = board[x][y].piece.value
                elif board[x][y].piece.side == "B" and board[x][y].piece.value > largestB:
                    largestB = board[x][y].piece.value
            except:
                pass
return (largestA, largestB)
```

Alpha-beta pruning is a modified version of the miniMax algorithm. It is an optimization technique that will fast the miniMax process. To use this algorithm, I need to let it find the more promising subtree on the board. Thus, the "more promising" subtree will be the node with the largest onboard value unit. I also need an alpha value and a beta value. Alpha is the best value that the maximizer currently can guarantee at that level or above, and Beta is the best value that the minimizer currently can guarantee at that level or above. In this project, the Alpha will be set to -100000, and the Beta will be set to 100000. The AI will find the best score on board; if the next movement can generate more scores than the current best score on board, AI will record this movement. Then the AI will find the largest value between the alpha value and the current best score on board and assign it to the Alpha. If the Alpha is now greater than the beta score, this node will be the best movement to act.



```
                    if moveScore > bestScore:
                        bestScore = moveScore
                        bestMove = (x, y, a, b)
                        # pruning
                        alpha = max(alpha, bestScore)
                        if (alpha >= beta):
                            return (bestMove, bestScore)
return (bestMove, bestScore)
```

There is also a depth value for the algorithm, and I use that to split the single-player mode into easy and hard modes. The difference is that the easy mode will set the max depth to 2, while the hard mode will be 4.

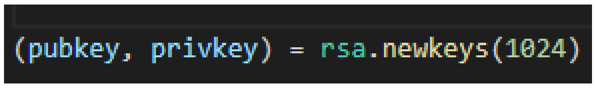


```
elif data.mode == "selectDifficulty":
    if 200 < event.x < 400 and 340 < event.y < 400:
        # easy mode
        data.maxDepth = 2
        data.mode = "onePlayerLayout"
    elif 200 < event.x < 400 and 420 < event.y < 480:
        # hard mode
        data.maxDepth = 4
        data.mode = "onePlayerLayout"
```

2.6.6 Chat and Encrpytion Funcition

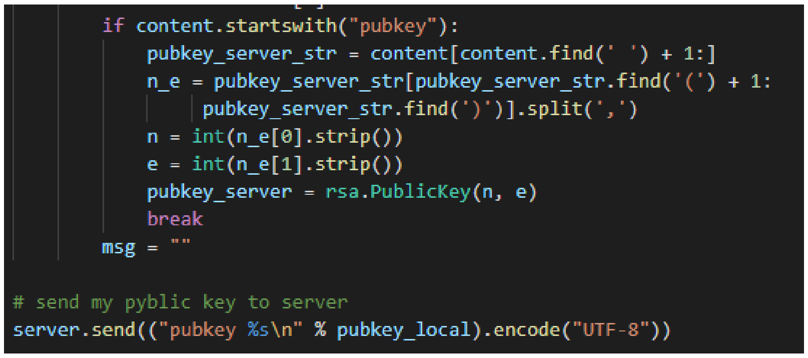
The chat function allows players to communicate with each other over a channel. The message will be encrypted with RSA encryption. RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys, i.e. Public Key and Private Key. As the name describes, the public key is given to everyone, and the private key is kept private.

A pair of RSA keys are generated using the standard RSA library in Python. In this project, it will generate a 1024 bits key.



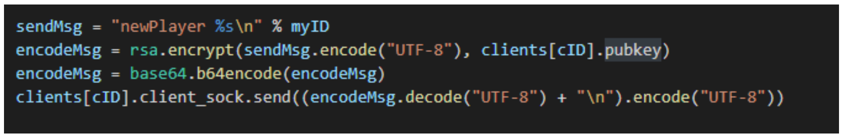
```python
(pubkey, privkey) = rsa.newkeys(1024)
```

Then the public key will be encoded in UTF-8 and send to the client side. If the client-side receives the message and finds out the message's first component is a public key, it will decode and verify it and send its public key to the server.



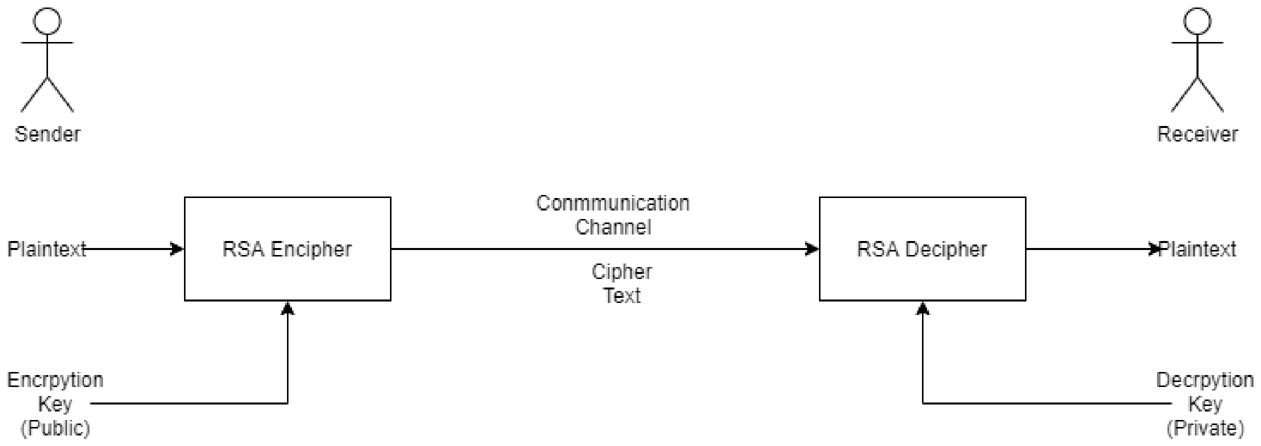
```python
        if content.startswith("pubkey"):
            pubkey_server_str = content[content.find(' ') + 1:]
            n_e = pubkey_server_str[pubkey_server_str.find('(') + 1:
                pubkey_server_str.find(')')].split(',')
            n = int(n_e[0].strip())
            e = int(n_e[1].strip())
            pubkey_server = rsa.PublicKey(n, e)
            break
        msg = ""

# send my pyblic key to server
server.send(("pubkey %s\n" % pubkey_local).encode("UTF-8"))
```

After the server gets the client's public key, it will decode and verify it and add the client's key to every message sent to the client side.



```python
sendMsg = "newPlayer %s\n" % myID
encodeMsg = rsa.encrypt(sendMsg.encode("UTF-8"), clients[cID].pubkey)
encodeMsg = base64.b64encode(encodeMsg)
clients[cID].client_sock.send((encodeMsg.decode("UTF-8") + "\n").encode("UTF-8"))
```

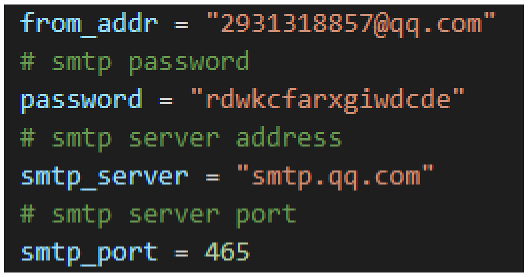
Below is the diagram of the encryption process.



2.6.7 SMTP Server

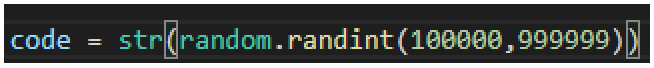
I choose to use a third-party SMTP service to achieve the functionality of the two-factor authentication. I choose to use the Tencent SMTP server since it is more convenient to send emails. I need to register a valid Tencent QQ account, a valid SMTP server address, and a valid SMTP port to set up the service.

The image below shows all elements to set up the SMTP server: a Tencent account and password, server address and server port.



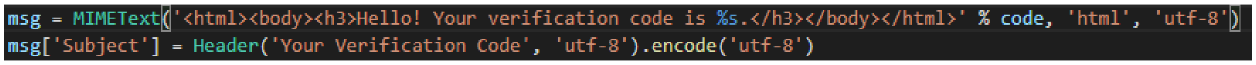
```
from_addr = "2931318857@qq.com"
# smtp password
password = "rdwkcfarxgiwdcde"
# smtp server address
smtp_server = "smtp.qq.com"
# smtp server port
smtp_port = 465
```

Then I need to create a random verification code for each request. I choose to use the Python Random library to generate a six-digit verification code for each request.



```
code = str(random.randint(100000,999999))
```

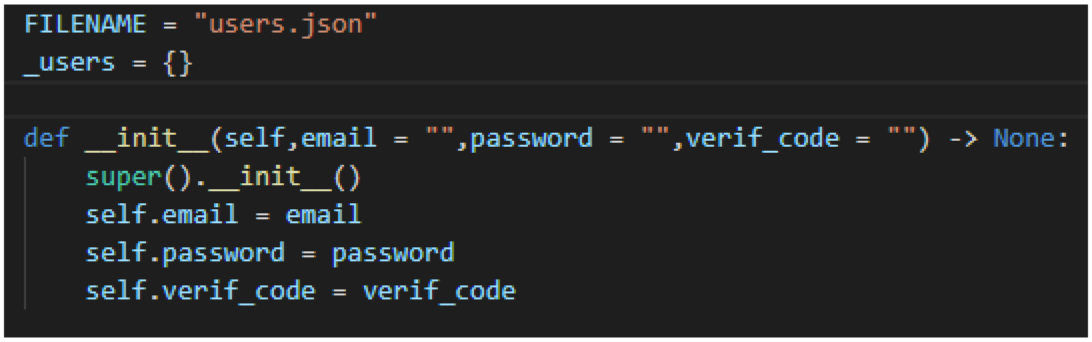
Then the verification code will be embedded into the body part of each email and send to the player's emails.



```
msg = MIMEText('<html><body><h3>Hello! Your verification code is %s.</h3></body></html>' % code, 'html', 'utf-8')
msg['Subject'] = Header('Your Verification Code', 'utf-8').encode('utf-8')
```

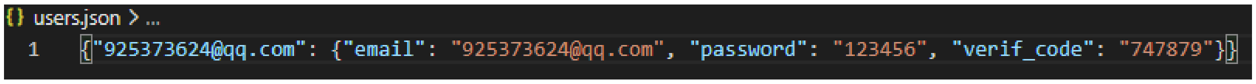
2.6.8 Player Register and Log in

Players can register and log in to the game via two-factor authentication. I created a JSON format file to store player login information to the local JSON file. As the image shown below, players will have a valid email address, passwords and verification code stored in the local file. If a player chooses to register in the first step, the SMTP server will send a verification email with a verification code inside.



```
FILENAME = "users.json"
_users = {}

def __init__(self,email = "",password = "",verif_code = "") -> None:
    super().__init__()
    self.email = email
    self.password = password
    self.verif_code = verif_code
```

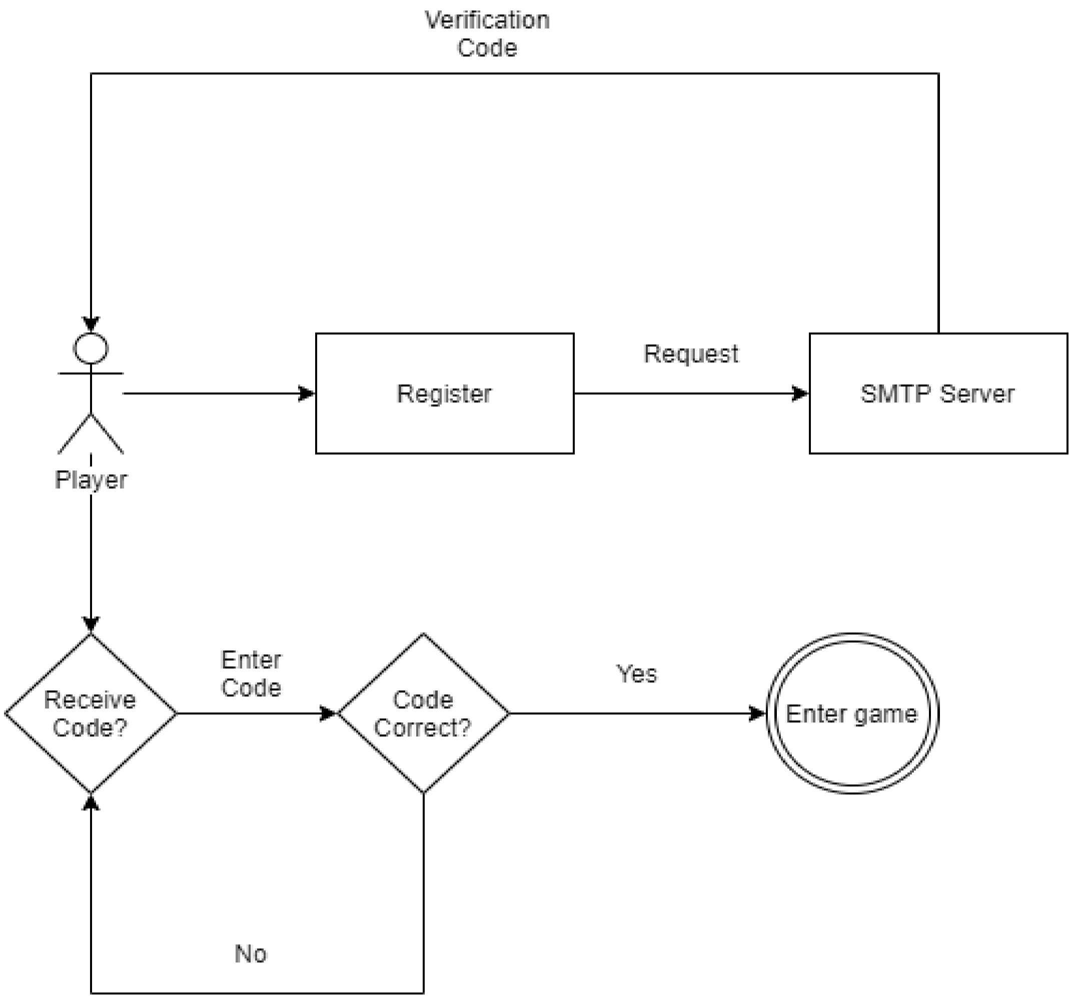
Once a player is registered, he will have a JSON file with his email and password for logging into the game. Every time the player tries to log in, the program will check whether the user entered password matches the email and password in the JSON file.



users.json > ...

```
{"925373624@qq.com": {"email": "925373624@qq.com", "password": "123456", "verif_code": "747879"}}
```

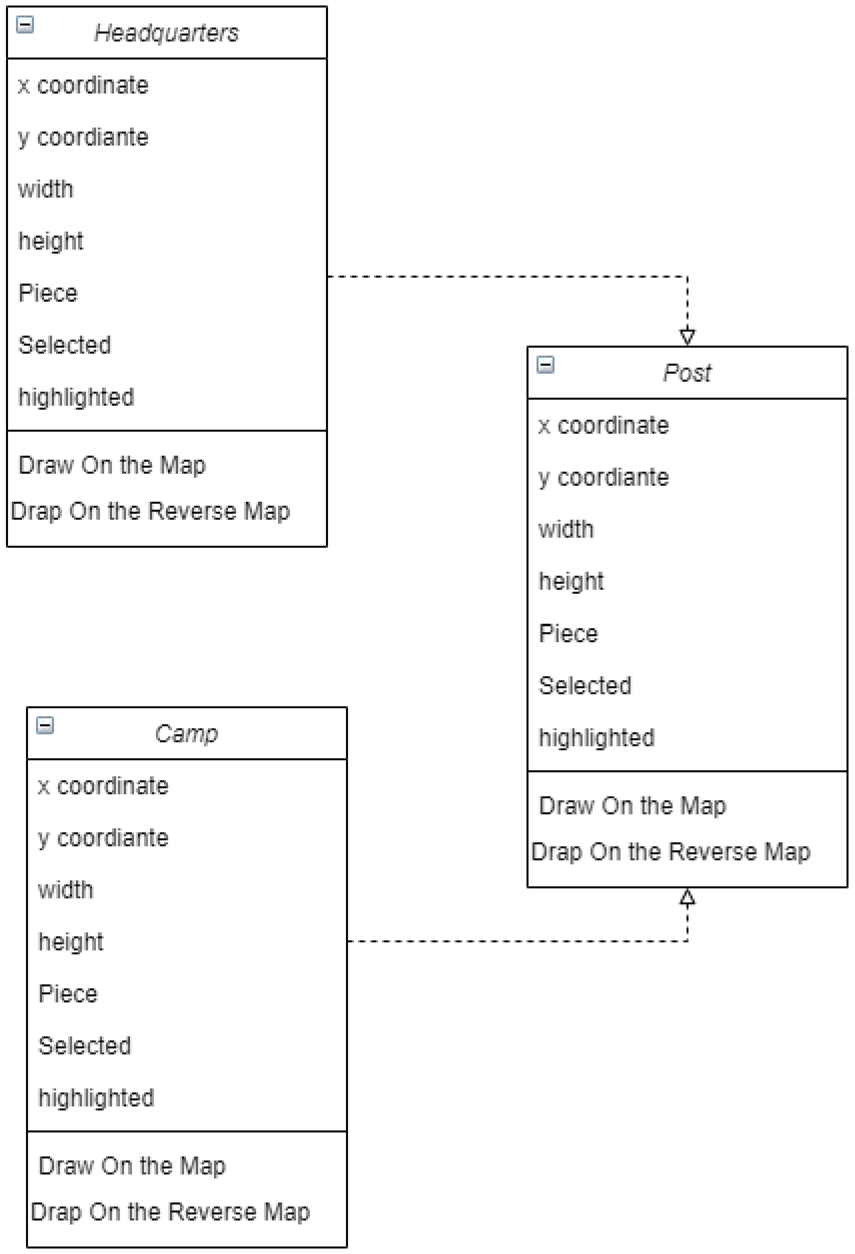
The image below explains the process.



2.7. Diagrams

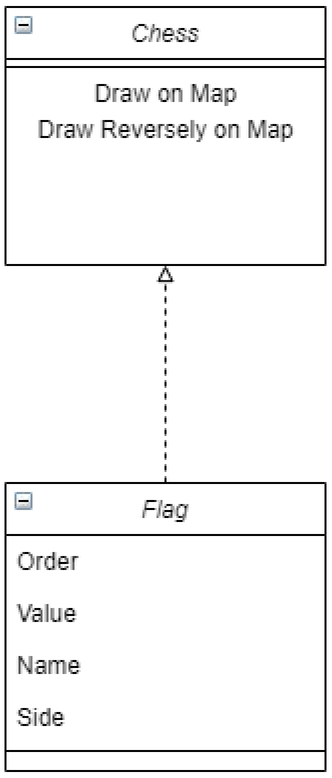
2.7.1 Map Objects Class Diagram

Class diagrams is the diagram in UML as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects.



2.7.2 Chess Units Class Diagram

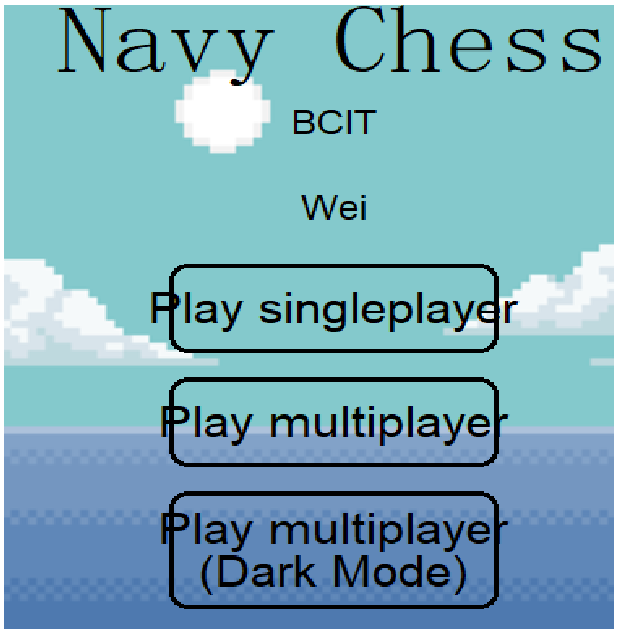
Below image will be a example of class diagram of a chess unit. Each chess unit will have same variable names but different values.



2.8. Testing Details and Results

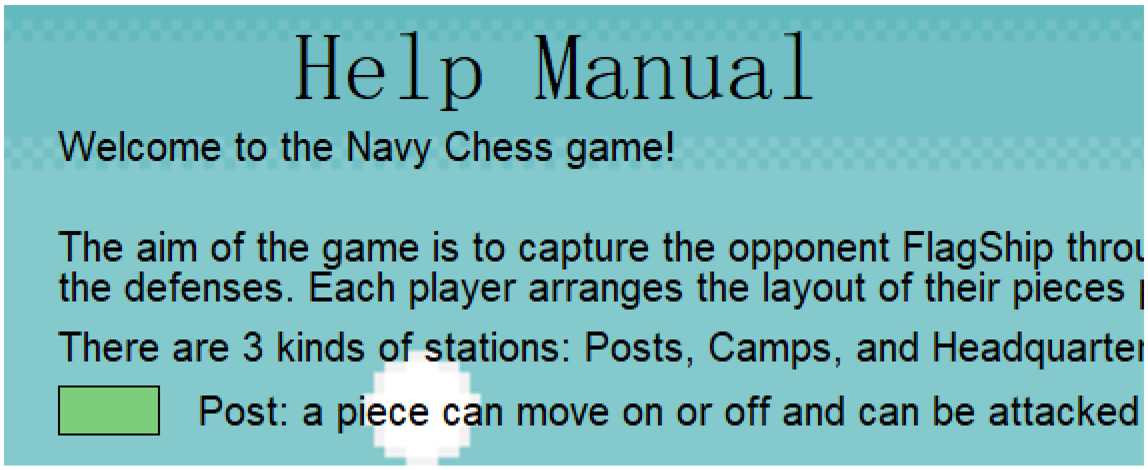
Test Case	**Description**	**Expect Result**	**Pass/Fail**
1	Verify landing page display properly	The landing page should display 3 buttons: Single Player, Multiplayer, Dark Mode	Pass

Result:



Test Case	Description	Expect Result	Pass/Fail
2	Verify landing page transfer to helper page	Players should enter and see the helper manual after pressing the "H" button on the landing page	Pass

Result:



Help Manual

Welcome to the Navy Chess game!

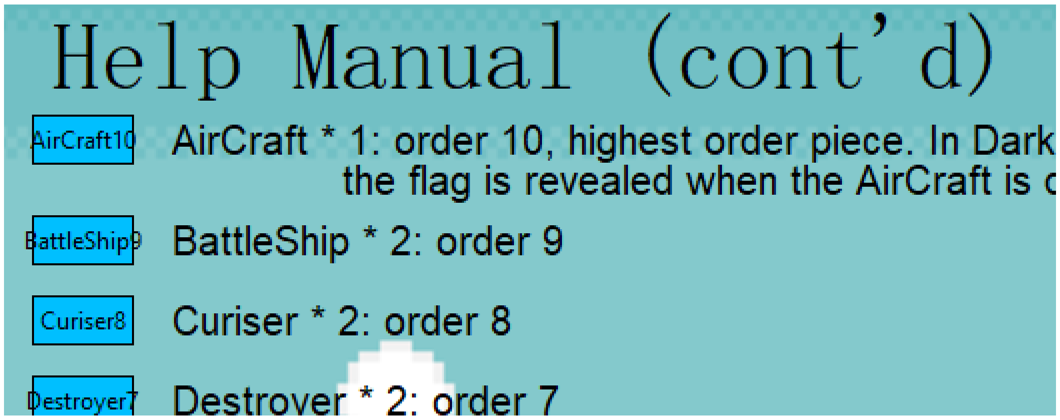
The aim of the game is to capture the opponent FlagShip throu
the defenses. Each player arranges the layout of their pieces

There are 3 kinds of stations: Posts, Camps, and Headquarte

🟩 Post: a piece can move on or off and can be attacked

Test Case	Description	Expect Result	Pass/Fail
3	Verify helper page transfer to the next helper page	Players should enter and see the second helper manual page after pressing the "N" button on the first	Pass

Result:

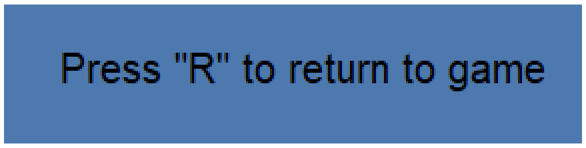


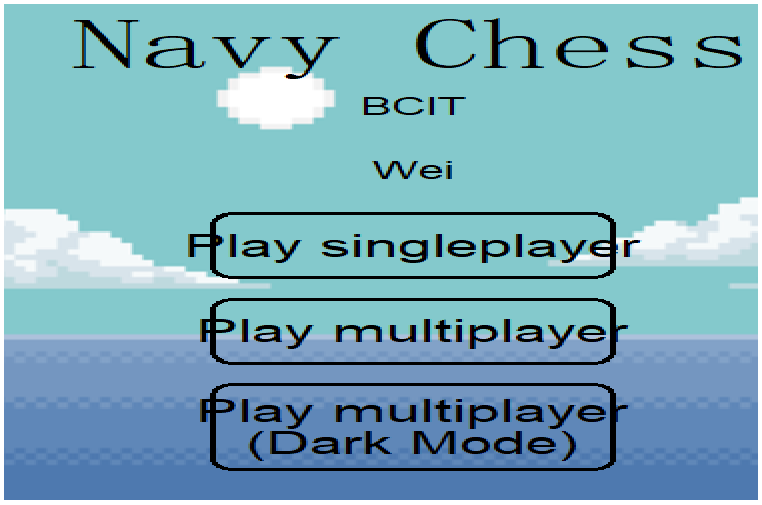
Help Manual (cont'd)

AirCraft10	AirCraft * 1: order 10, highest order piece. In Dark the flag is revealed when the AirCraft is c
BattleShip9	BattleShip * 2: order 9
Curiser8	Curiser * 2: order 8
Destroyer7	Destroyer * 2: order 7

Test Case	Description	Expect Result	Pass/Fail
4	Verify helper page transfer back to the landing page	Players should enter and see the landing page after pressing the "R" button in the second help manual.	Pass

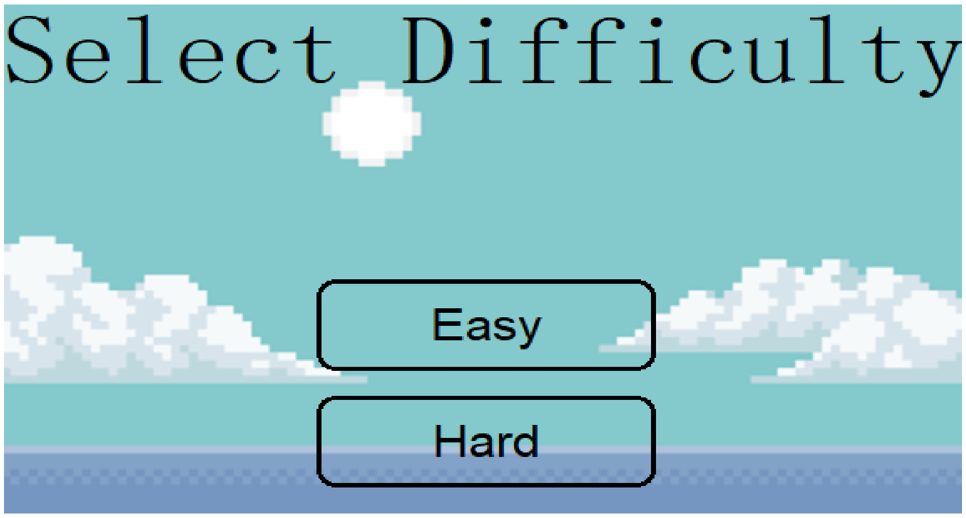
Result:





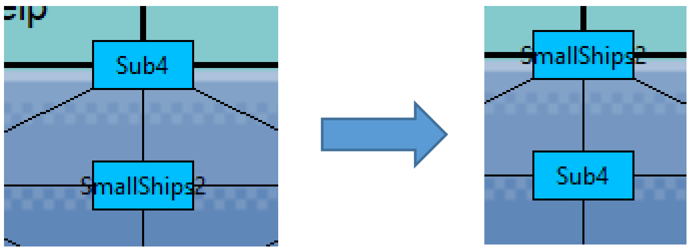
Test Case	Description	Expect Result	Pass/Fail
5	Verify transfer from landing page to single-player mode select page	Players should enter and see the single-player page and see easy and hard mode choices	Pass

Result:



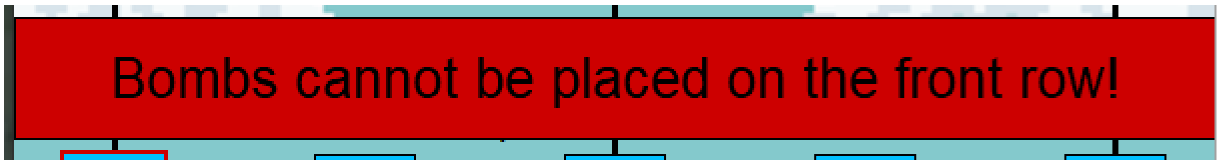
Test Case	Description	Expect Result	Pass/Fail
6	Verify single-player units set up	Players should see the game map, chess units and should be able to move chess units	Pass

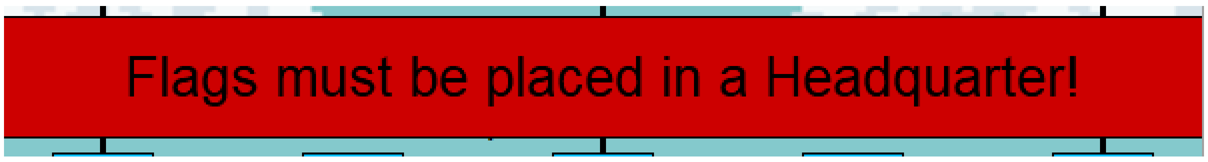
Result:



Test Case	Description	Expect Result	Pass/Fail
7	Verify single-player units set up rules	Players should see the game map, chess units and should be able to move chess units only to the allowed positions	Pass

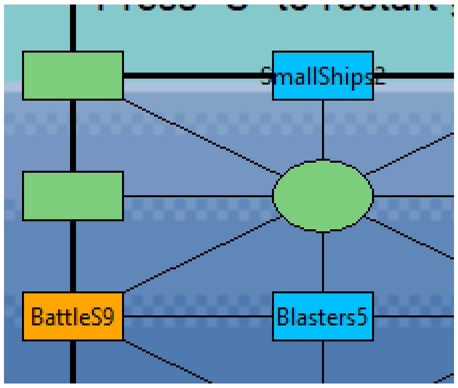
Result:


Bombs cannot be placed on the front row!


Flags must be placed in a Headquarter!

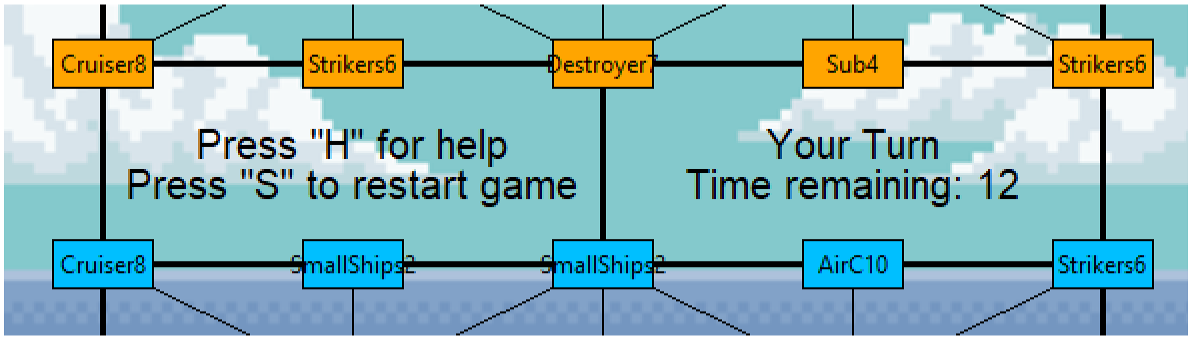
Test Case	Description	Expect Result	Pass/Fail
8	Verify single player start button	Players should see the game start and be able to start attacking opponent units	Pass

Result:



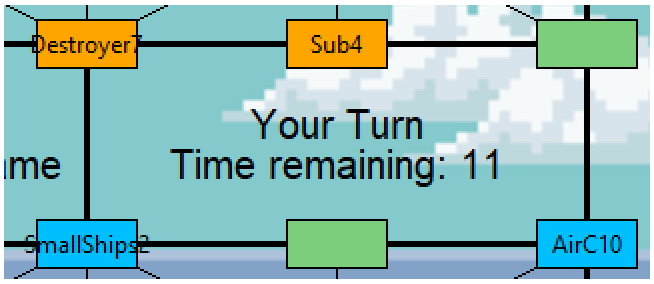
Test Case	Description	Expect Result	Pass/Fail
9	Verify single-player opponent setup	Players should see the opponent chess unit after starting the game	Pass

Result:



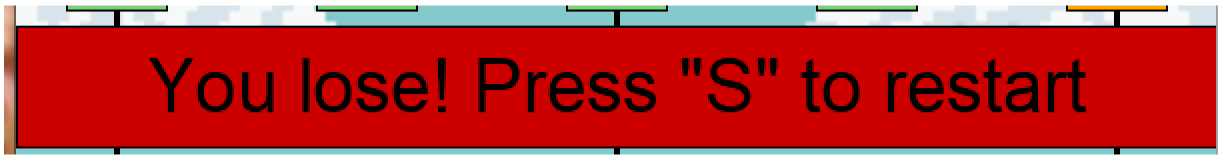
Test Case	Description	Expect Result	Pass/Fail
10	Verify single-player timer	Players should see the timer after starting the game	Pass

Result:



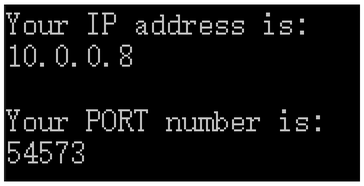
Test Case	Description	Expect Result	Pass/Fail
11	Verify single player that AI can win the game	Players should lose the game if the AI destroys their FlagShips.	Pass

Result:


You lose! Press "S" to restart

Test Case	Description	Expect Result	Pass/Fail
12	Verify single player server	User should see the server start and see server IP and port open	Pass

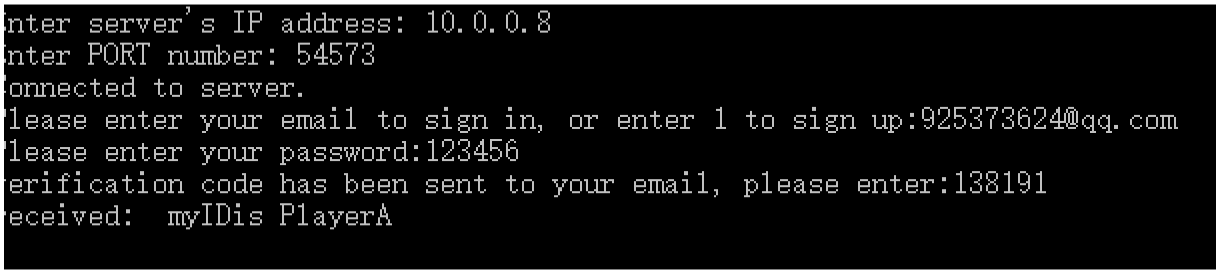
Result:



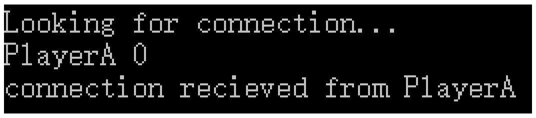
```
Your IP address is:
10.0.0.8

Your PORT number is:
54573
```

Test Case	Description	Expect Result	Pass/Fail
13	Verify single-player server-client connection	User should see the server start and connect to the server using the showing IP address and port	Pass

Result:



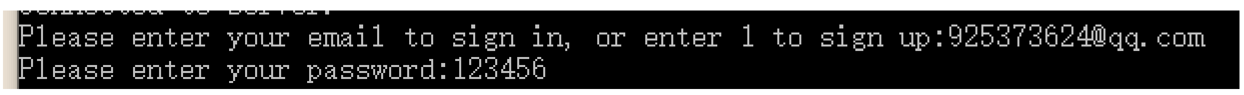
```
nter server's IP address: 10.0.0.8
nter PORT number: 54573
onnected to server.
lease enter your email to sign in, or enter 1 to sign up:925373624@qq.com
lease enter your password:123456
erification code has been sent to your email, please enter:138191
eceived:  myIDis PlayerA
```



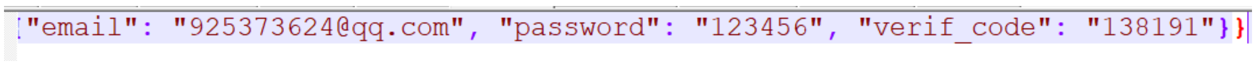
```
Looking for connection...
PlayerA 0
connection recieved from PlayerA
```

Test Case	Description	Expect Result	Pass/Fail
14	Verify single-player user login	The user should enter the correct registered password and email address to login	Pass

Result:



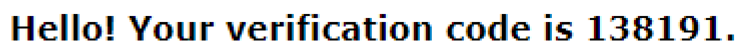
```
Please enter your email to sign in, or enter 1 to sign up:925373624@qq.com
Please enter your password:123456
```



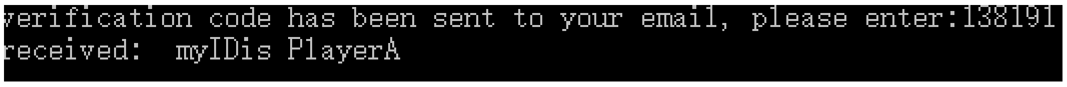
```
{"email": "925373624@qq.com", "password": "123456", "verif_code": "138191"}}
```

Test Case	Description	Expect Result	Pass/Fail
15	Verify single-player two-factor authentication code	The user should receive a verification code after entering the correct password and email address	Pass

Result:



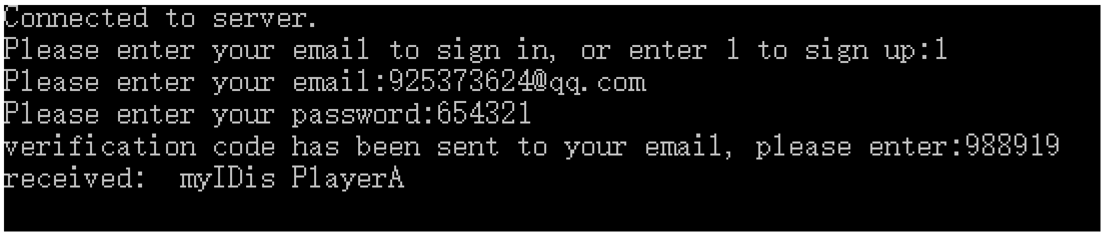
Hello! Your verification code is 138191.



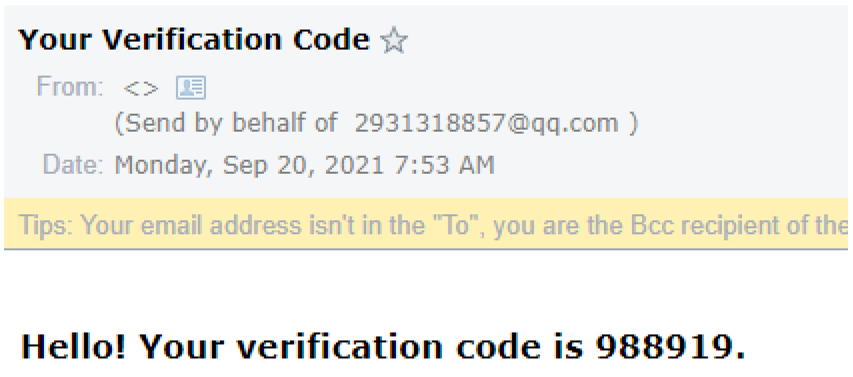
```
verification code has been sent to your email, please enter:138191
received:  myIDis PlayerA
```

Test Case	Description	Expect Result	Pass/Fail
16	Verify player register function	Players should be able to register accounts following the instructions showing on the terminal	Pass

Result:



```
Connected to server.
Please enter your email to sign in, or enter 1 to sign up:1
Please enter your email:925373624@qq.com
Please enter your password:654321
verification code has been sent to your email, please enter:988919
received:  myIDis PlayerA
```



Your Verification Code ☆

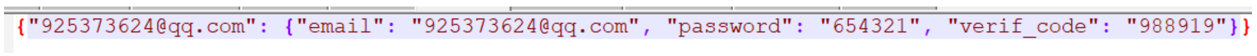
From: <>

(Send by behalf of 2931318857@qq.com)

Date: Monday, Sep 20, 2021 7:53 AM

Tips: Your email address isn't in the "To", you are the Bcc recipient of the

Hello! Your verification code is 988919.

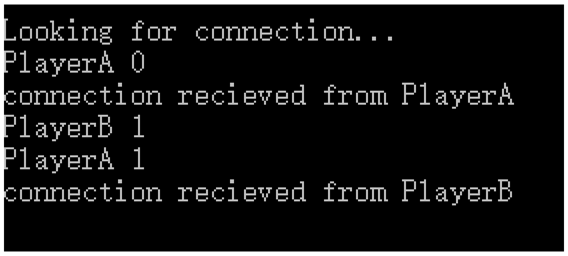


{"925373624@qq.com": {"email": "925373624@qq.com", "password": "654321", "verif_code": "988919"}}

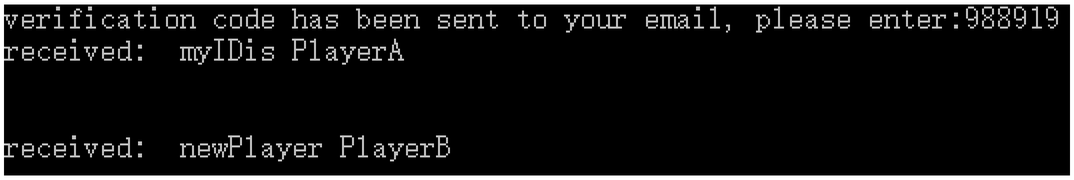
Test Case	Description	Expect Result	Pass/Fail

17	Verify multiplayer player connection	The second player should enter the multiplayer game	Pass

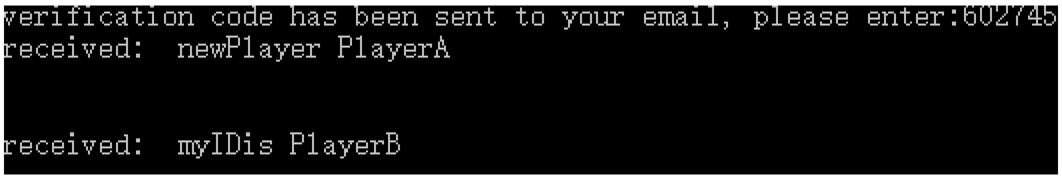
Result:



```
Looking for connection...
PlayerA 0
connection recieved from PlayerA
PlayerB 1
PlayerA 1
connection recieved from PlayerB
```



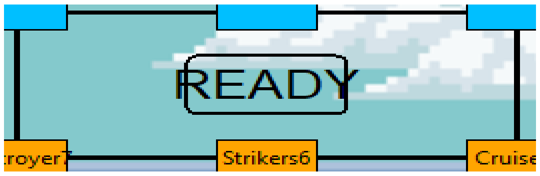
```
verification code has been sent to your email, please enter:988919
received:  myIDis PlayerA


received:  newPlayer PlayerB
```

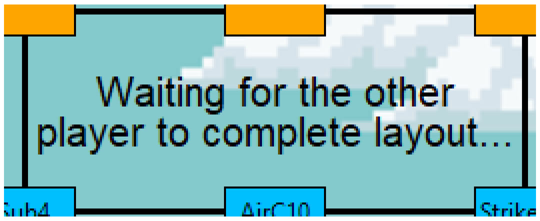


```
verification code has been sent to your email, please enter:602745
received:  newPlayer PlayerA


received:  myIDis PlayerB
```

Test Case	**Description**	**Expect Result**	**Pass/Fail**
18	Verify multiplayer player ready button	Both players should see the ready button and will only start the game when both players click the ready button	Pass

Result:

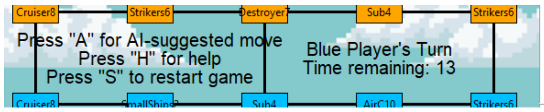




Waiting for the other player to complete layout...

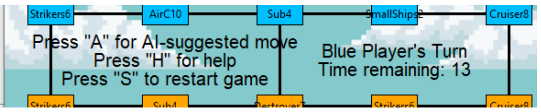
Test Case	Description	Expect Result	Pass/Fail
19	Verify multiplayer player timer	After starting the game, both players should see a timer	Pass

Result:



Press "A" for AI-suggested move
Press "H" for help
Press "S" to restart game

Blue Player's Turn
Time remaining: 13

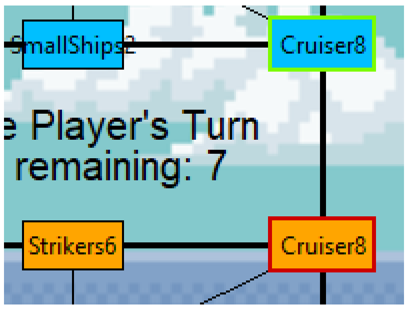


Press "A" for AI-suggested move
Press "H" for help
Press "S" to restart game

Blue Player's Turn
Time remaining: 13

Test Case	Description	Expect Result	Pass/Fail
20	Verify multiplayer AI suggestions	Both players should be able to see AI suggestions when pressing A button	Pass

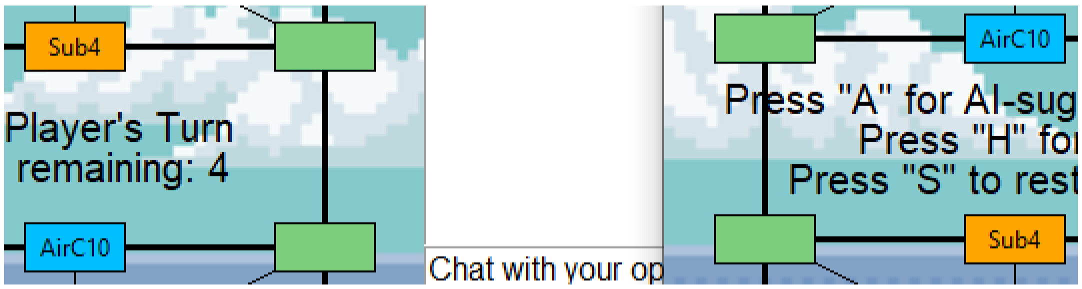
Result:



e Player's Turn
remaining: 7

Test Case	Description	Expect Result	Pass/Fail
21	Verify multiplayer player movements	Both players should be able to attack, move their units	Pass

Result:



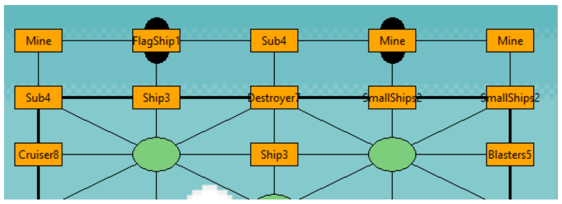
Player's Turn
remaining: 4

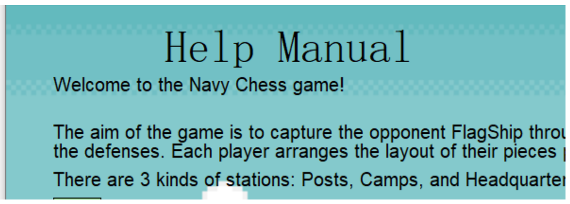
Chat with your op

Press "A" for AI-sug
Press "H" for
Press "S" to rest

Test Case	Description	Expect Result	Pass/Fail
22	Verify multiplayer player helper function	Both players should be able to check the help manual anytime after pressing the H button in the game	Pass

Result:



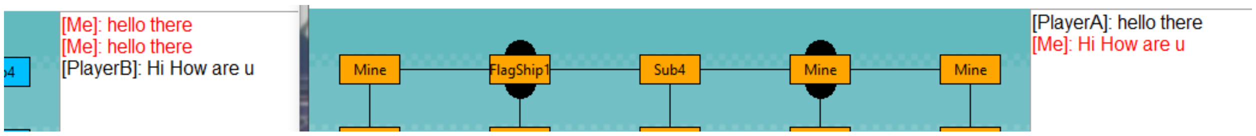

Help Manual

Welcome to the Navy Chess game!

The aim of the game is to capture the opponent FlagShip throu
the defenses. Each player arranges the layout of their pieces
There are 3 kinds of stations: Posts, Camps, and Headquarte

Test Case	Description	Expect Result	Pass/Fail
23	Verify multiplayer chat function	Both players should be able to communicate through the chat window on their right	Pass

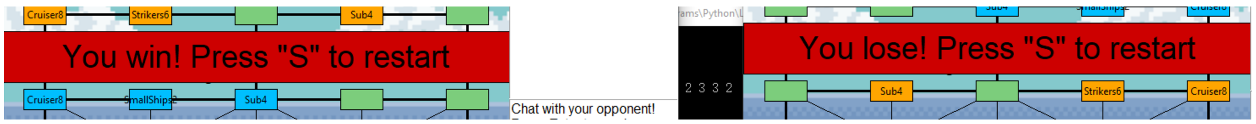
Result:


[Me]: hello there
[Me]: hello there
[PlayerB]: Hi How are u

[PlayerA]: hello there
[Me]: Hi How are u

Test Case	Description	Expect Result	Pass/Fail
24	Verify multiplayer win and lose	If one player wins the game, the other should lose the game	Pass

Result:


You win! Press "S" to restart

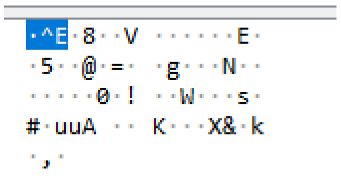
Chat with your opponent!

You lose! Press "S" to restart

Test Case	Description	Expect Result	Pass/Fail

25	Verify data encryption	Using wireshark monitoring data transfer, it should be encrypted messages instead of plaintext	Pass

Result:

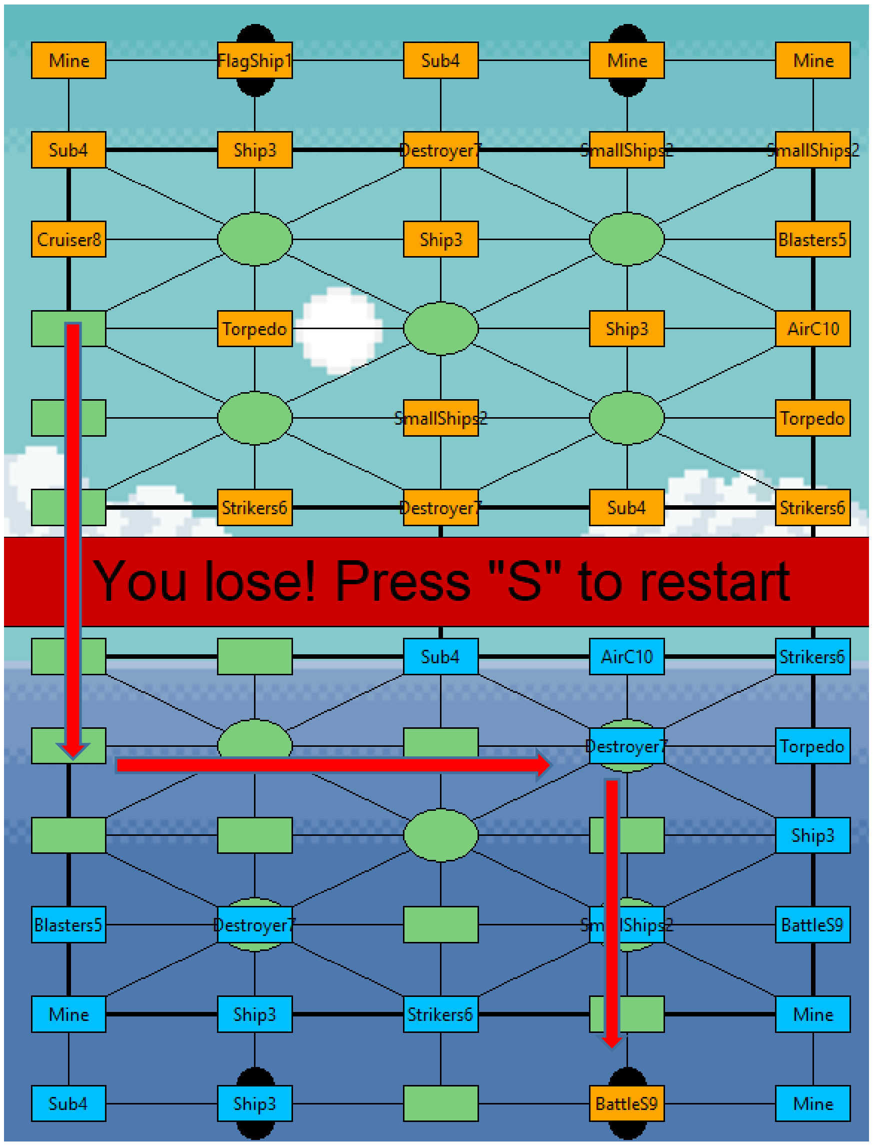


```
·^E·8··V ······E·
·5··@·=· ·g···N··
·····0·! ··W···s·
#·uuA ·· K···X&·k
·,·
```

2.9. Implications of Implementation

In this project, I digitalize a board chess game using Python. For the graphic part, I used Python Tkinter, which is a GUI toolkit that allows me to draw maps, buttons, chess units. For the client-server part of the project, I used the UDP and basic socket library in Python. This allows me to connect players to the server, and UDP ensures better performance in multiplayer game mode. By using UDP, the program can synchronize both players' chess board movements in less than 1.5 seconds.

As for the single-player part, AI is built by MiniaMax and Alpha-beta pruning algorithms. They are search algorithms that allow AI to find the best possible movement. For easy single-player mode, MiniMax with Alpha-beta pruning allows AI to make a move within 1.5 seconds. Considering the algorithm depth setting for easy mode is only 2, a player will sometimes see AI move its units back and forth in some rows even the players give AI a chance to win. Unlike easy mode, the algorithm depth setting for hard mode is 4. AI will make its next move in less than 2 seconds, but AI will always aim to the most important target if AI has a chance. For example, in easy mode, AI will take about 20 turns to win the game even it can win the game in 14 turns. AI will take only 14 to 15 turns in hard mode if AI can win the game in 14 turns. In the image below, it can be clearly seen that the AI's Battleship is aiming directly at my Flagships without any hesitation in hard mode.



The two-factor authentication is achieved by using the SMTP service provided by Tencent, which allows me to send and receive verification codes. Although I use Tencent's SMTP service, players can still get their verification code in less than 10 seconds.

2.10. Innovation

Play Mode: The current existing game in the Google Play store is four players mode. There will be two players modes for this game.

Chat: There will be a chat function that players can interact with their opponents during the game.

Encrypted data transfer: I will also add a live chat function for the game so players can communicate. But data transfer during the live chat will be encrypted by end-to-end encryption.

End-to-end encryption is the process in which encryption of data is being done at the end host. It is an implementation of asymmetric encryption.

RSA algorithm will be used for this project. RSA algorithm is an asymmetric encryption algorithm. RSA requires a public key and a private key and works as follow:

1. Person A sends the public key to person B and requests some data
2. Person B encrypts the data using person A's public key and sends the encrypted data
3. Person A receives the data and decrypts it.

AI: There will be a single-player mode for the game that players can match with AI. The AI algorithm will use miniMax with heuristics and alpha-beta pruning.

The MiniMax algorithm is a recursive algorithm usually used in decision-making games such as chess games. MiniMax uses recursion to search through the game tree. If two players play a game with the MiniMax algorithm, both the players will fight their opponent player to get the maximum benefit where their opponent gets the minimum benefit.

The heuristic algorithm is to find solutions among all possible solutions. However, it does not guarantee that the best solution will be found; therefore, it can be considered not accurate algorithms. The heuristic algorithm usually finds a solution close to the best one, and it finds it fast.

Alpha-beta pruning is an optimization technique for the minimax algorithm. Because there are 50 chess units for a game, alpha-beta pruning reduces the computation time. This allows the AI to search much faster and even dig deeper into the game tree that it also allows me to implement an easy AI mode and a hard AI mode. Alpha-beta will cut off branches in the game tree that will not be searched because there are better moves that exist.

Two-factor Authentication: Two-factor authentication is an electronic authentication method in which a device user is granted access to a website or application only after successfully presenting two or more pieces of evidence to an authentication mechanism.

For this project, the client-side player will need to use a valid Gmail account to connect to the server-side player. If one player enables Two-Factor Authentication for Gmail, the player goes through a setup process that first configures his account to send his phone a code by text or voice. At the end of the setup, he will have an opportunity to add Google Authenticator as an alternative code generator. He will use Google Authenticator to enter a 32-character secret string that he will use to add configuration to this account in the authenticator. Once done, the player will receive an authentication code in the Google Authenticator when he wants to join the game.

2.11. Complexity

Synchronize issue: this game needs to solve the synchronize problem. How the game can handle each command, such as move the unit, attack the opponent and sync all the steps so players can be in the same stage, instead of one player already wins the game and the other player is still moving his units. UDP is really fast, and if the program is not properly set up, there could be a situation that one player hasn't finished the movement, but the other player can start moving his units.

Algorithm: Because the game contains about 50 units per match, I need to create an algorithm that allows each unit to move, attack as players want. Also, an AI algorithm for the game requires me to research and study how to do AI for a game. It is hard to decide how the algorithm can fit the game.

Programming skills: I need to improve my Python skills more because BCIT never really taught us about this language. I would not use Unity for this game but building the game from scratch. So, I need to learn how to GUI programming based on Python. Also, I need to search for how to create links to make the player invitation function work.

Testing: Testing is also a big barrier I need to conquer. Because I have no experience testing a multiplayer game build by Python, and I need to test every part of the project.

Time: This game requires a student to have proper art design, socket, Python, Python GUI, AI algorithm skills to finish the game within 350 hours. Skills like art design, GUI, AI are fairly new to me. So, after researching and learning some of the skills, I may need to extend my schedule or limit my scope.

Encryption: End-to-end encryption is a new subject for me because my experience with encryption is to encrypt data with only one layer of protection, such as RSA encryption or AES encryption. End-to-end requires a secure channel with a combination of encryption methods, such as RSA with AES and SHA. I may need to spend time researching and understand the technology, which could cause me to extend the schedule.

2FA: Apply two-factor authentication to the game is a challenge. I need to combine the authentication with a TCP socket, Python and an external mobile application. I may need to spend time researching and understand the technology, which could cause me to extend the schedule.

2.12. Research in New Technologies

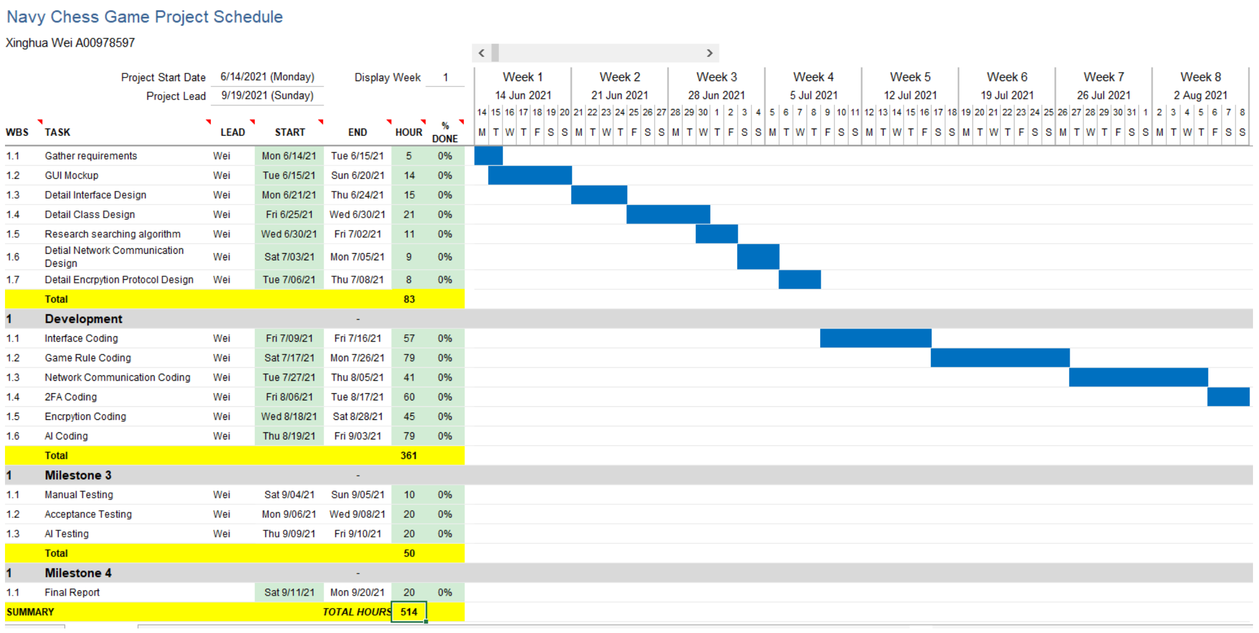
There are few technical challenges for this project. One of which is understanding how Alpha-beta pruning and Minimax can fit the game. Although there are many chess games that already use the algorithms, those games do not have the ability to suicide chess units on purpose. I treat the whole chess map as 12 times 5 list, and the program needs to calculate the highest onboard score every time on AI's turn. The second challenge is how to fit chess units variables into the searching algorithms. Since chess units can suicide in this game, I give every unit a rank order, a health value and a score value. The algorithms will calculate the best possible move best on the highest possible score value it can earn and use the highest possible rank order units to attack. Another challenge is how to add a two-factor authentication function to the game. I need to connect the two-factor authentication process to the client-server connections. Before doing that, I need to research which SMTP service is convenient and robust.

2.13. Future Enhancements

I have tested and verified that every function in this game can perform normally. However, the server is only designed to have maximum 2 people gaming at the same time. In the future, this game could have a game lobby to allow players to create their individual game room that dozens of people can game at the same time.

Second, players need to log in to their registered account every time they join the game. The game could have a function to remember user credentials if players do not want to enter email and passwords on their computer. Besides, the two-factor authentication process can be transformed to a third-party mobile application instead of using email to receive verification codes every time.

2.14. Timeline and Milestones



Navy Chess Game Project Schedule

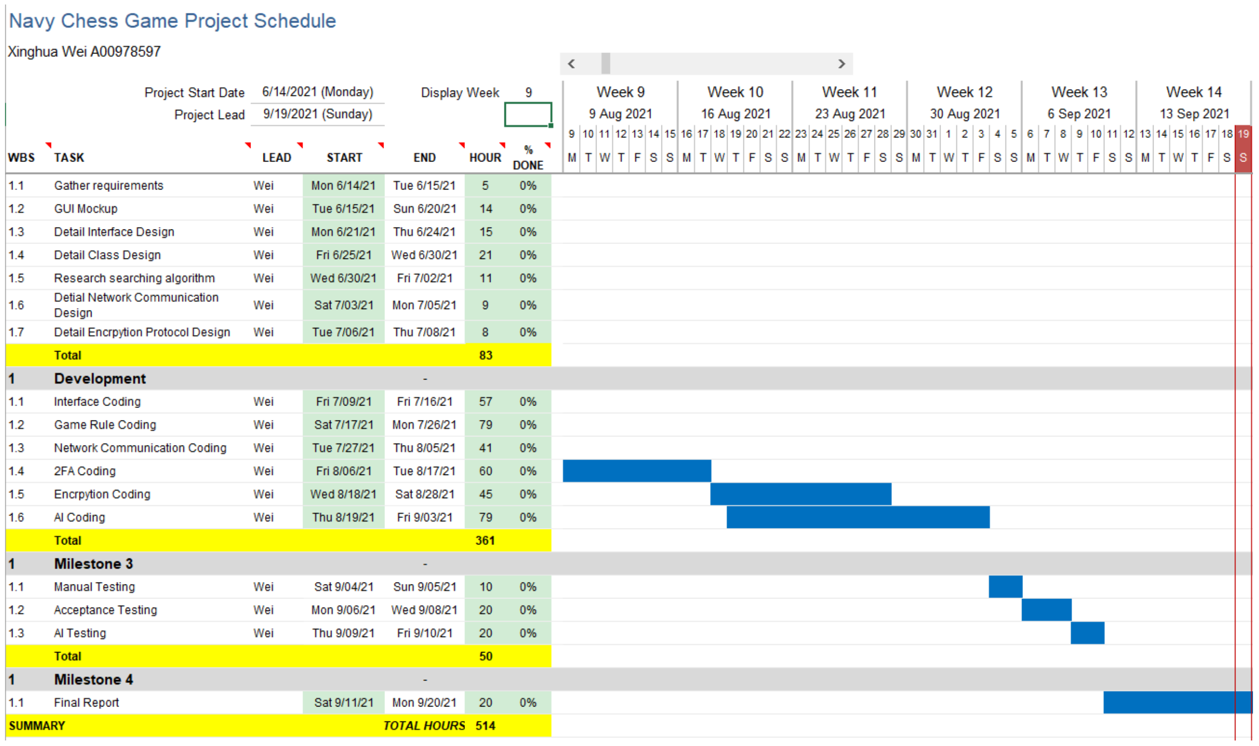
Xinghua Wei A00978597

Project Start Date 6/14/2021 (Monday) Display Week 1

Project Lead 9/19/2021 (Sunday)

Week 1 14 Jun 2021 · Week 2 21 Jun 2021 · Week 3 28 Jun 2021 · Week 4 5 Jul 2021 · Week 5 12 Jul 2021 · Week 6 19 Jul 2021 · Week 7 26 Jul 2021 · Week 8 2 Aug 2021

WBS	TASK	LEAD	START	END	HOUR	% DONE
1.1	Gather requirements	Wei	Mon 6/14/21	Tue 6/15/21	5	0%
1.2	GUI Mockup	Wei	Tue 6/15/21	Sun 6/20/21	14	0%
1.3	Detail Interface Design	Wei	Mon 6/21/21	Thu 6/24/21	15	0%
1.4	Detail Class Design	Wei	Fri 6/25/21	Wed 6/30/21	21	0%
1.5	Research searching algorithm	Wei	Wed 6/30/21	Fri 7/02/21	11	0%
1.6	Detial Network Communication Design	Wei	Sat 7/03/21	Mon 7/05/21	9	0%
1.7	Detail Encrpytion Protocol Design	Wei	Tue 7/06/21	Thu 7/08/21	8	0%
	Total				83	
1	**Development**			-		
1.1	Interface Coding	Wei	Fri 7/09/21	Fri 7/16/21	57	0%
1.2	Game Rule Coding	Wei	Sat 7/17/21	Mon 7/26/21	79	0%
1.3	Network Communication Coding	Wei	Tue 7/27/21	Thu 8/05/21	41	0%
1.4	2FA Coding	Wei	Fri 8/06/21	Tue 8/17/21	60	0%
1.5	Encrpytion Coding	Wei	Wed 8/18/21	Sat 8/28/21	45	0%
1.6	AI Coding	Wei	Thu 8/19/21	Fri 9/03/21	79	0%
	Total				361	
1	**Milestone 3**			-		
1.1	Manual Testing	Wei	Sat 9/04/21	Sun 9/05/21	10	0%
1.2	Acceptance Testing	Wei	Mon 9/06/21	Wed 9/08/21	20	0%
1.3	AI Testing	Wei	Thu 9/09/21	Fri 9/10/21	20	0%
	Total				50	
1	**Milestone 4**			-		
1.1	Final Report		Sat 9/11/21	Mon 9/20/21	20	0%
SUMMARY				TOTAL HOURS	514	



Navy Chess Game Project Schedule

Xinghua Wei A00978597

Project Start Date 6/14/2021 (Monday) Display Week 9

Project Lead 9/19/2021 (Sunday)

Week 9 9 Aug 2021 · Week 10 16 Aug 2021 · Week 11 23 Aug 2021 · Week 12 30 Aug 2021 · Week 13 6 Sep 2021 · Week 14 13 Sep 2021

WBS	TASK	LEAD	START	END	HOUR	% DONE
1.1	Gather requirements	Wei	Mon 6/14/21	Tue 6/15/21	5	0%
1.2	GUI Mockup	Wei	Tue 6/15/21	Sun 6/20/21	14	0%
1.3	Detail Interface Design	Wei	Mon 6/21/21	Thu 6/24/21	15	0%
1.4	Detail Class Design	Wei	Fri 6/25/21	Wed 6/30/21	21	0%
1.5	Research searching algorithm	Wei	Wed 6/30/21	Fri 7/02/21	11	0%
1.6	Detial Network Communication Design	Wei	Sat 7/03/21	Mon 7/05/21	9	0%
1.7	Detail Encrpytion Protocol Design	Wei	Tue 7/06/21	Thu 7/08/21	8	0%
	Total				83	
1	**Development**			-		
1.1	Interface Coding	Wei	Fri 7/09/21	Fri 7/16/21	57	0%
1.2	Game Rule Coding	Wei	Sat 7/17/21	Mon 7/26/21	79	0%
1.3	Network Communication Coding	Wei	Tue 7/27/21	Thu 8/05/21	41	0%
1.4	2FA Coding	Wei	Fri 8/06/21	Tue 8/17/21	60	0%
1.5	Encrpytion Coding	Wei	Wed 8/18/21	Sat 8/28/21	45	0%
1.6	AI Coding	Wei	Thu 8/19/21	Fri 9/03/21	79	0%
	Total				361	
1	**Milestone 3**			-		
1.1	Manual Testing	Wei	Sat 9/04/21	Sun 9/05/21	10	0%
1.2	Acceptance Testing	Wei	Mon 9/06/21	Wed 9/08/21	20	0%
1.3	AI Testing	Wei	Thu 9/09/21	Fri 9/10/21	20	0%
	Total				50	
1	**Milestone 4**			-		
1.1	Final Report		Sat 9/11/21	Mon 9/20/21	20	0%
SUMMARY				TOTAL HOURS	514	

3. Conclusion

In conclusion, the digitalized Navy Chess game is a huge challenge for me. I developed a fully functional digitalized 2D Navy Chess game for the Windows and Linux platforms. I researched and developed AI algorithms for the game so players can match with AI. Chat function and end-to-end encryption are essential for the game's security aspect to prevent potential data leaks and intercept during transmission. Also, players can send a link to their friends so that players can invite another player to his match room.

The Navy Chess game project helped me enhancing many of my skills in terms of software development:

- Software architecture design
- Software API design
- Software testing
- 2D art design
- Project time and schedule management
- Project scope control
- Python programming, Python GUI programming, Python Socket programming
- Networking programming, network security programming
- Decision-making algorithms

My overall skills in programming, algorithms, managing, networking and designing will be improved as I overcome the technical challenges and achieve the innovation part of the project.

3.1. Lessons Learned

The list below shows the knowledge I can get upon the completion of the project and how the acquired knowledge contributes to my expertise development.

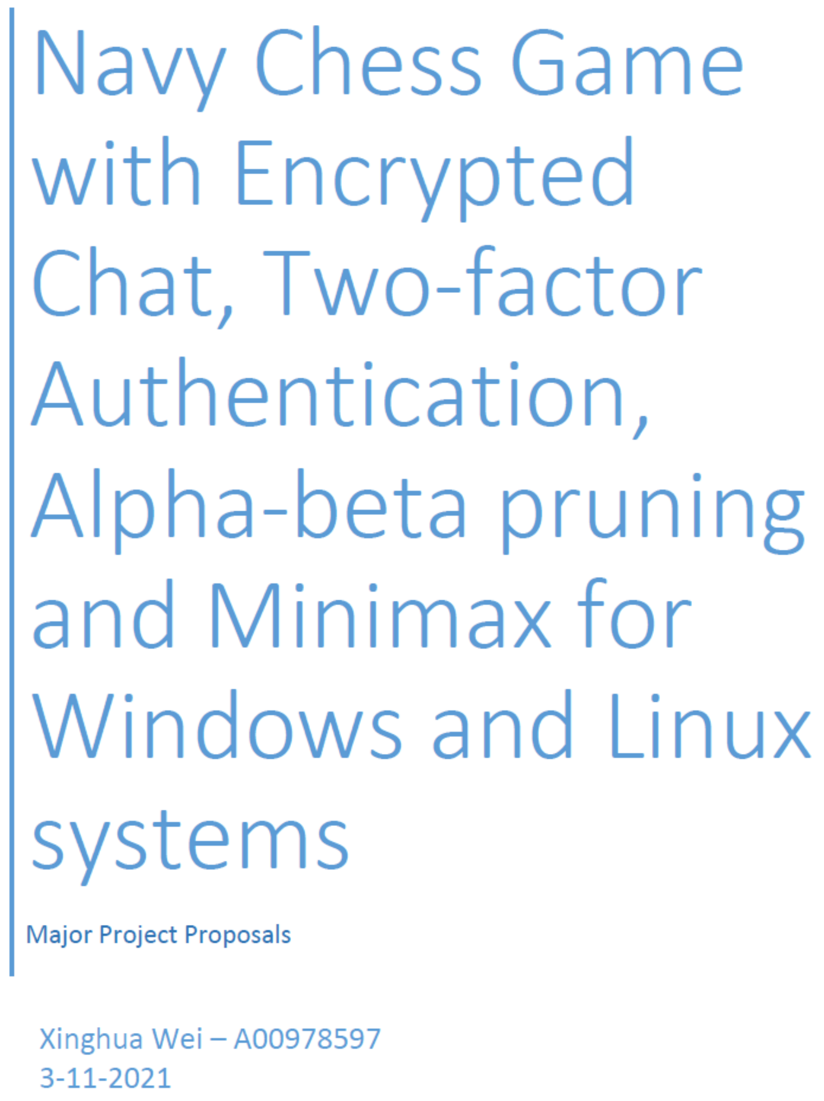
- I learned how to use a third-party SMTP service. Third-party SMTP service is very useful. It can accomplish two-factor authentication, but it can also help achieve functions like sending notification emails.
- I learned how to do GUI using Python Tkinter. Tkinter is a really useful Python GUI toolkit. Besides, Tkinter is Python's standard library, and it is easy to use. I will be able to build more Python applications with GUI.
- I learned how to embed encryption into UDP translations. UDP is generally fast but unreliable compare to TCP. But encrypting plaintext or data before sending them out can largely increase the security level of UDP transmission. As a network security option student, knowing more about encryption will be helpful to my further career since suitable encryption will avoid lots of data security problems like the data leak. Besides, one of the applications will be written in Python. As Python is a popular language for network security applications, it is helpful to build different kinds of network apps in the future if we expertise in Python. Moreover, we will use Python encryption libraries in this project. It will be a good practice because knowing these libraries will be useful for us to build our own encryption library.

3.2. Closing Remarks

[Include some concluding remarks regarding the project.]

4. Appendix

4.1. Approved Proposal



Navy Chess Game with Encrypted Chat, Two-factor Authentication, Alpha-beta pruning and Minimax for Windows and Linux systems

Major Project Proposals

Xinghua Wei – A00978597
3-11-2021

1. Student Background

I am a student enrolled in the BCIT CST Btech program major in the network security application development option. I graduated from the BCIT CST diploma program Database option.

During the BCIT CST diploma program, I had one-year programming experience with JAVA, some programming experience with C, C++, JavaScript, HTML5 and MySQL. I have accomplished a few website applications such as a website to encourage users to save food, website which support selling products online. Additionally, I also had an opportunity to create an Android mobile application that indicates crime in the current user area.

During the ongoing network security application development option, I learned some basic ideas and functionalities behind TCP/IP. Also, I learned how to create simple TCP/IP communication applications, a simple firewall using Shell Script. Now I am learning ideas and methods to do ethical hacking and hiding data while transmitting data. Project Description

1.1. Education

British Columbia Institute of Technology

Program: Computer Systems Technology 2016-2018

- Graduate from CST Database option

Program: Computer Network Security Application Development Ongoing

Skills

- Network Security
  Understand basic ideas about network security, network applications using Python, Shell Script on the Linux platform. Understand basic socket, TCP/IP communication using Python to implement some IPv4 client/server applications.

- Website

  Implement and design websites with HTML5 and JavaScript. I have completed five team projects successfully. Also, I have programming experience using Bootstrap.

- Database

  I have half-year experience with Oracle, T-SQL, MySQL, Visual studio, and data mining with different software such as Hive and R studio.

1.2. Work Experience

In 2019 summer, I interned at ScopeMedia, a studio that delivers products for e-commerce websites. My main job there was to help another developer testing his API calls.

2. Project Description

I propose to create a multiplayer chess game call Navy Chess, a two-player board game. There are no existing games on any platforms since it is an old-fashion board game to my current knowledge and research. I want to create a digital version of Navy Chess for Windows platforms where users can download it and connect to other players via the players' IP address and port using Python socket.

Besides the game itself, a live chat function with encrypt data transmit underneath that will provide a safer and more reliable gaming environment. To let the signal player enjoy the game, Two different AI levels for this game with Alpha-beta pruning and Minimax allow the signal player to start a match without waiting for another player. In terms of player connections, there will be a two-factor authentication applied to the game that whenever play A wants to connect to player B, there will be a request to authenticate it via an external 2FA application.

Overall, this project has four main goals:

- To develop a fully functional digitalized 2D Navy Chess game for the Windows platform using Python.

- To research and develop a single-player mode with Alpha-beta pruning and Minimax.
- To develop a secure live chat with a data transmitting function to enhance the gamer's gaming environment by using end-to-end encryption.
- To establish Python sockets so players can invite another player to his match room.
- The invitation will use two-factor authentication via an external application.

3. Problem Statement and Background

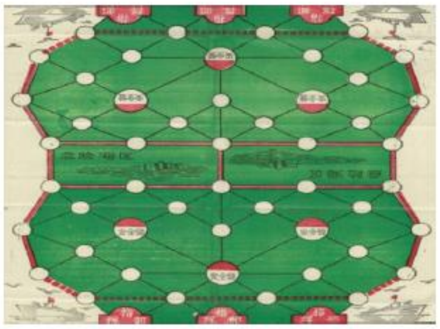
Navy Chess is a two-player board game, and it was derived from another board game SanJun Chess. To my current knowledge and research, both games are no longer online on any platform since it is an old-fashion board game, and most people treat it as antique.

The main combat units include aircraft carriers, Battleship, torpedoes (same as all types of ships), mine blasters (lightning strike ships), and submarines (the only ships that can defeat all types of ships are defeated by lightning strike ships). The secondary combat units include three missile boats (which can knock down Aircraft carriers), torpedo boats (which can knock down submarines), mines (to protect military flags at sea), and three minesweepers (which can knock down mines). In addition, because there are two anti-submarine aircraft groups (anti-submarine aircraft and bombers) walking on the chessboard with black dotted lines (courses), Navy chess has produced the concept of three-dimensional space combat because of the submarine, anti-submarine aircraft, and bomber group.

3.1 Existing APP

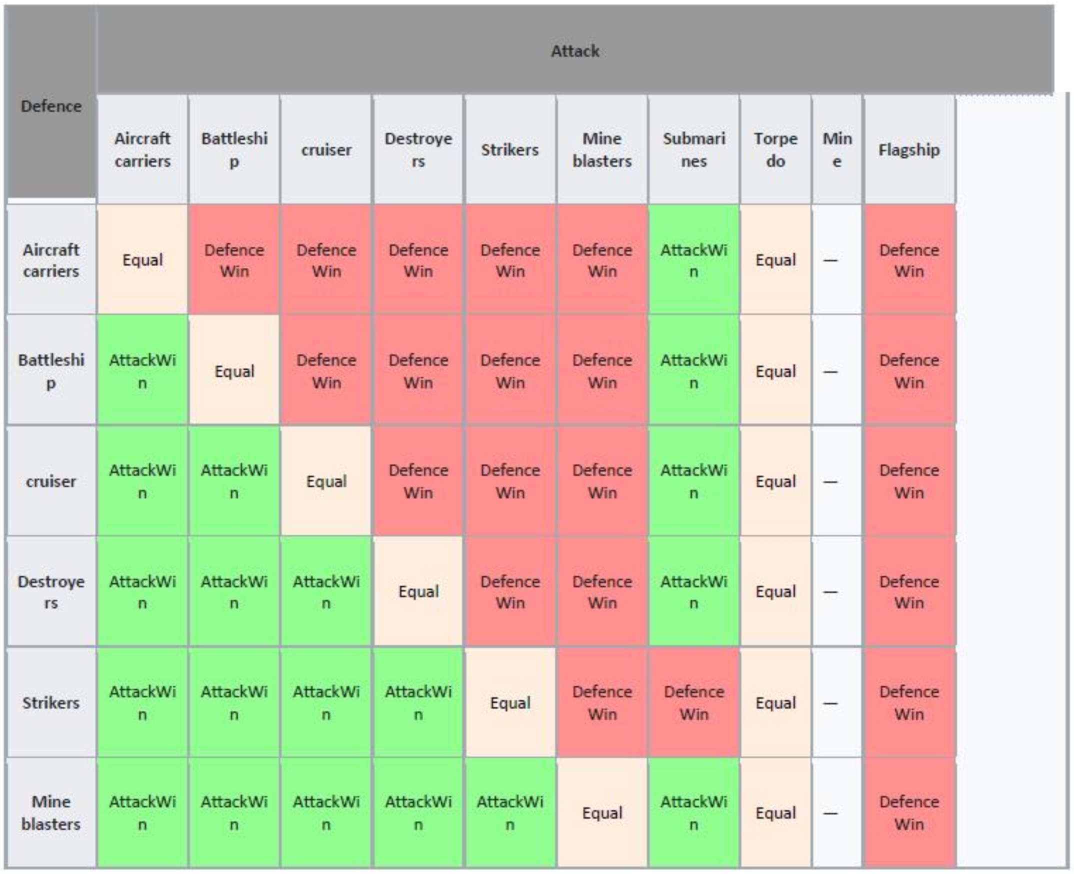
To my current knowledge and research, there are no existing games on any platform. The currently available information about the game only includes a map and the rules of the game.

The image below shows the paper version of the game map.



3.2 Navy Chess game

The chart below shows the settings of the game.



Defence	Attack									
	Aircraft carriers	Battleship	cruiser	Destroyers	Strikers	Mine blasters	Submarines	Torpedo	Mine	Flagship
Aircraft carriers	Equal	Defence Win	Defence Win	Defence Win	Defence Win	Defence Win	AttackWin	Equal	—	Defence Win
Battleship	AttackWin	Equal	Defence Win	Defence Win	Defence Win	Defence Win	AttackWin	Equal	—	Defence Win
cruiser	AttackWin	AttackWin	Equal	Defence Win	Defence Win	Defence Win	AttackWin	Equal	—	Defence Win
Destroyers	AttackWin	AttackWin	AttackWin	Equal	Defence Win	Defence Win	AttackWin	Equal	—	Defence Win
Strikers	AttackWin	AttackWin	AttackWin	AttackWin	Equal	Defence Win	Defence Win	Equal	—	Defence Win
Mine blasters	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	Equal	AttackWin	Equal	—	Defence Win

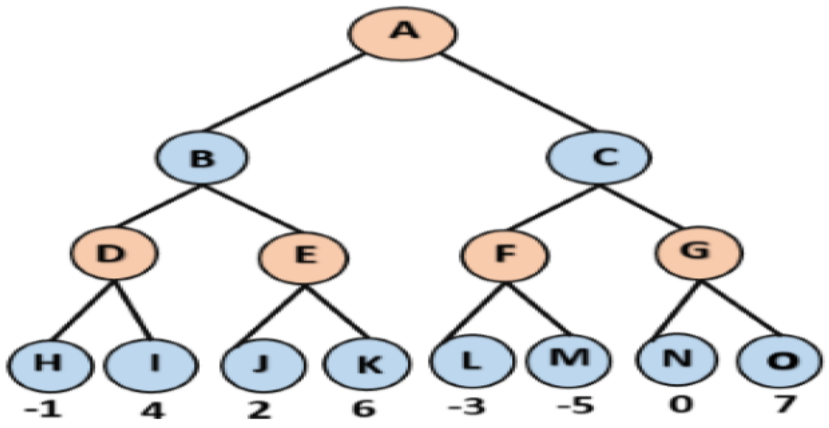
Submarines	Defence Win	Defence Win	Defence Win	Defence Win	AttackWin	Defence Win	Equal	Equal	—	Defence Win	
Torpedo	Equal	Equal	Equal	Equal	Equal	Equal	Equal	Equal	—	Equal	
Mine	Equal	Equal	Equal	Equal	Equal	AttackWin	Equal	Equal	—	Equal	
Flagship	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	AttackWin	Equal	—	Equal	

3.3 AI Algorithm Explain

3.3.1 MiniMax

The miniMax algorithm will be used for this game. It is a search algorithm that allows the program to look ahead at possible future positions before deciding what move it wants to make in the current position.

For example, there are two players that one is called Max, and the other is called Min. Each terminal node represents a chess unit, and each node has a value. The red node represents Max, and the blue node represents Min. Max will try to get the Maximum score, where Min will try to get the minimum possible score.



Step 1:

In the first step, the algorithm will generate the entire game tree and get each terminal node's values. Suppose Max takes the first turn. It will choose the best value from the bottom to the top for each node because the miniMax is a backtracking algorithm.

- For D: max(-1:4) = 4
- For E: max(2:6) =6
- For F: max(-3,-5) = -3
- For G: max(0,7) = 7

Step 2:

Then it is Min's turn. Min will try to get the minimum possible score.

- For B: min(4:6) = 4
- For C: min(-3,7) = -3

Step 3:

Now it is Max's turn. Max will again get the maximum possible score.

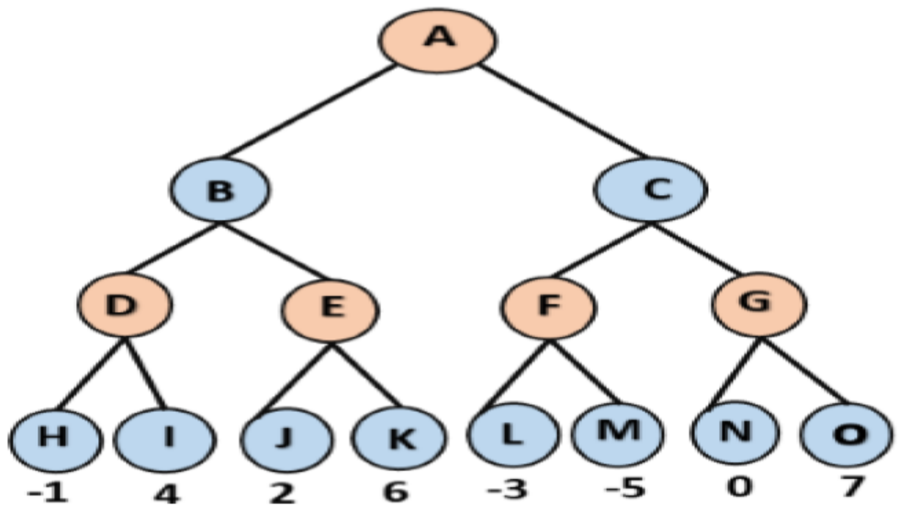
- For A: max(-3,4) = 4

Therefore, in this current position, Max will choose to get a value of 4, which means that Max will move accordingly.

3.3.2 Alpha-Beta Pruning

Alpha-beta pruning is a modified version of the miniMax algorithm. It is an optimization technique that will fast the miniMax process. With Alpha-beta pruning, the program will not need to check each node of the game to compute the correct decision.

For example, there are two players that one is called Max, and the other is called Min. Each terminal node represents a chess unit, and each node has a value. The red node represents Max, and the blue node represents Min. Max will try to get the Maximum score, where Min will try to get the minimum possible score. Also, the alpha will be added and means negative infinite. Beta means positive infinite.



Step 1:

Max will start the move from node A that alpha is negative infinity, and beta is positive infinity. The values will be passed down to nodes B and D.

At node D, the value of node H and I are compared. Max will get the maximum value:

- For D: max(-1:4) = 4

Step 2:

It is Min's turn that Min will compare the beta value with the current node B value at node B. Min will get the minimum value:

For B: min(4, positive infinite) = 4

Step 3:

At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5,

- For alpha: max(-∞, 5) = 5,

Therefore, at node E, alpha is 5 and beta is 3, where alpha is larger than beta. So the right successor of node E will be pruned, and the algorithm will not traverse it, and the value at node E will be 5.

This is an example of how a node will be pruned. The Alpha-beta algorithm will apply the same methodology to the other nodes.

There will be more than 4 layers and more than 2 nodes to compare for the real algorithm in the game. Each chess unit will be assigned a value according to its rank level. For example, a general has a value of 10, and a major have a value of 6. The AI will be the Max in the example, and it will try to get the maximum possible score around the board to make its move.

3.3.3 End-to-End encryption

End to end encryption is the process in which encryption of data is being done at the end host. It is an implementation of asymmetric encryption.

There are two types of encryption, one is symmetric encryption, and the other is asymmetric encryption. I will use asymmetric encryption for this project. In Asymmetric encryption, two types of keys are used, one public key and one private key. The sender and the receiver both have the public key and the private key. The Public Keys are available to both sender and receiver to share public keys before communication starts. The sender uses the receiver's public key to encrypt the messages to be sent, and the receiver uses both its public and private keys to decrypt the messages. The private key to the receiver is available only with the receiver and no one else.

RSA algorithm will be used for this project. RSA algorithm is an asymmetric encryption algorithm. RSA requires a public key and a private key and works as follow:

1. Person A sends the public key to person B and requests some data
2. Person B encrypts the data using person A's public key and sends the encrypted data
3. Person A receives the data and decrypts it.

3.3.4 Two-factor Authentication

Two-factor authentication is an electronic authentication method in which a device user is granted access to a website or application only after successfully presenting two or more pieces of evidence to an authentication mechanism.

For this project, the client-side player will need to use a valid Gmail account to connect to the server-side player. If one player enables Two-Factor Authentication for Gmail, the player goes through a setup process that first configures his account to send his phone a code by text or voice. At the end of the setup, he will have an opportunity to add Google Authenticator as an alternative code generator. He will use Google Authenticator to enter a 32-character secret string that he will use to add configuration to this account in the authenticator. Once done, the player will receive an authentication code in the Google Authenticator when he wants to join the game.

4. Scope and Depth

The list of activities the game is expected to have are:

User Interface

- Display a game background map for both players setting up their chess
- Display 50 chess units in total for both players setting up their strategy
- Display buttons for surrender, join, invite, exit, etc.
- Display a live chat panel on the side of the game map.

Functional Requirements

- Chess units are movable by the correct player on the right path of the map
- Higher-ranked chess units should destroy Lower-ranked chess units
- Players take turns to perform their moves
- Map space should be correctly allocated for each chess unit
- A player communicates with another player through live chat
- A player generate invitation links to friends to invite the second player to join the game

- AI can attack, move chess units and be able to win or lose the game.
- Players should be able to see the game rules by pressing the H button during a game.
- Player invitation use two-factor authentication via an external application.

Non-Functional Requirements

- In each turn, players should not exceed 30 seconds.
- A game introduction for new players to understand the rules of the game
- Communication between players should be encrypted except for two players
- Frequently used functions should be tested for stability and usability
- Each invitation links from a player lost invitation ability after 30 minutes
- The game will be developed by Python without any game engine
- The game should be available for both Windows platforms
- The game should be implemented with Python, Python socket and Python Tkinter.
- Players should be able to change AI mode. For example, choose an easy AI or a hard AI

Out of Scope

- Upload the game online to make it a website available game

5. Test Plan

Manual testings, including unit testing, will be done throughout the development of the application. Acceptance testing will also be done after finishing unit testing.

5.1 Manual Testing & Unit Testing

Manual Testing: Manual testing is the testing in which test cases are executed manually using any automated tool. The purpose of manual testing is to identify any existing bugs and defects in the application.

Manual Testing will be done for modules that are difficult to test, such as controller modules.

- List testing features in the modules
- List types of input the features can accept

- List expect result

Unit Testing: Unit testing will be done for a small, easy to test portion of the project and verifies its behaviour independently from other parts. For unit testing, the Python library "unittest" will be used.

- List codes I want to test
- Initialize a piece of code
- Apply unit testing code for the code
- Compare results

5.2 Acceptance Testing

Acceptance Testing will be done after unit testing of the project. Acceptance testing needs to verify that the application passes the manual tests list below. Additional tests may be conducted and added as need.

5.2.1 Players enter the match

Players enter a game match with correct IP addresses and ports.

- One player starts a match and opened a port
- The other player joins the match with the correct IP address and port
- Pass criteria:
    - One player should be able to create a game room and visually see that the game is running.
    - The second player should join the room with the correct IP address and port.

5.2.2 Display players' information

Players enter their players' names before joining a match

- Players join a match
- Players' names display on the correct side
- Pass criteria:

    - Both players should be able to see their player information on the game interface.

5.2.3 Players move their unit by turns

Players move their units to occupy a place on the map.

- Units can be moved correctly according to players' actions
- Units show up on the target position
- Pass criteria:
    - One player takes his or her turn to perform a move
    - Once the player finished his or her move, that player should not move any units until the other player completes his or her activity.

5.2.4 Players attack opponents' units

Players use their units to attack the opponent's units.

- Players move one of their units to the target unit's position
- The target unit is eliminated if the attacker's unit has a higher rank. Otherwise, the attacker's unit is eliminated.
- Pass criteria:
    - One player chooses a unit and moves the unit towards an enemy's unit
    - If the attacking unit's rank is higher than the victim's rank, the victim unit should be removed from the map. Otherwise, the attacking unit should be removed.

5.2.5 Players lose a game

Players lose a game if opponents' units eliminate their base.

- One of the opponents' units attacks and occupies a player' base
- The player loses the game
- Pass criteria:
    - The opponent's unit attacks one player's base. The player should lose the game.

5.2.6 Players win a game

Players win a game if they eliminate opponents' bases.

- One of the units attacks and occupies an opponent' base
- The player wins the game
- Pass criteria:
  - One player's unit attacks the opponent's base. The player should win the game.

5.2.7 Players communicate with each other

Players chat with their opponents using the chat function during a match.

- Players type and click send to send messages
- Opponents receive the messages
- Pass criteria:
  - One player types in the chat section and click send to send messages
  - The opponent should receive the messages and be able to send messages back

5.2.8 Players leave a game room

Players leave a game room before, during or after a match.

- Players click the leave the game room button
- Another window pops up to confirm players' actions
- Players leave the game room
- Pass criteria:
  - One player click the leave the game room button
  - The player clicks the confirm button when the confirm to leave alert shows up
  - The player should be able to leave the room

5.2.5 AI tests

The following tests are AI Accepting test cases:

Test Case	Test Description	Input

AI Moves a unit	AI should move a unit based on the game rules whenever the human player finishs his turn	Move Command
AI attacks with a unit	AI should move a unit towards the human player's unit to perform an attack	Attack Command
AI set up its unit	AI should be able to set up all its 50 units on the game map based on the game rules	AI set up commands
AI notifies the human player	AI should automatically notify the human player with a message to inform the human player that the player is playing against an AI through the chat section	A string message
AI wins the human player	AI should automatically notify the human player with a message to inform the human player that the AI beats the human player through the chat section	A string message
AI loses the human player	AI should automatically notify the human player with a message to inform the human player that the player beats the AI through the chat section	A string message

6. Methodology

6.1 Methodologies & Approach

The software development methodology for this project. Each task of this project will be executed serially because there is only one developer for this project. The waterfall methodology provides the capability to implement a minimum viable product quickly.

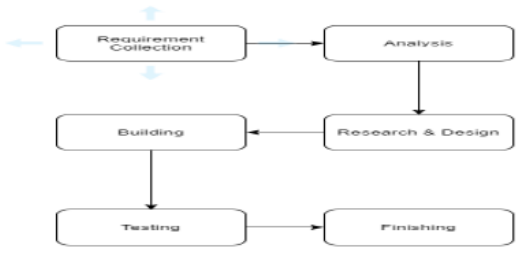


Figure 1Waterfall Methodology Lifecycle

Page **15** of **26**
Xinghua Wei – A00978597

- **First stage**: Requirements collection
    - Expectations and goals of the project will be collected in this stage.
- **Second stag**e: Design
    - Finalizing the core structure of the product
    - Prepare a schedule for the developer
- **Third stage**: Building
    - Developing the product
- **Fourth stage**: Testing
    - Manual Testing
    - Unit Testing
    - Acceptance Testing
- **Fifth stage**: Finishing
    - Validate the project whether the project meets the requirements
    - Create a final report of the project

Waterfall methodology is chosen because it is useful for projects that prioritize quality rather than the cost of time duration to better control the quality of the project. Also, the methodology's stability makes project management easier since there is only one developer who will monitor the entire project. I could actively monitor the progress at any stage of the development. The waterfall methodology reduces project complexity since all phases occur without any overlap. Additionally, it provides me with flexibility when I need to redesign any part of my project functionalities.

6.2 Technologies

For project management, the following tools and technologies:

- **Microsoft Project**: use to keep tracking the progress of each task, each stage and upcoming tasks and stages.
    - Microsoft Project is a project management software to help me develop a schedule, assign resources to tasks, and track progress.

For software design, the following tools and technologies:

- **Draw.io** to create diagrams like UML diagram, system diagram.
    - Draw.io is a free diagram editor.
- **Proto.io** to design UI for the project
    - Proto.io is an application to prototype apps for interfaces.

For software development, the following tools and technologies:

- **Python**: The whole project will be implemented based on Python 3.8. Python is an interpreted, high-level and general-purpose programming language. Because Python has many well-developed third-party libraries, a straightforward structure, it is suitable for this project.
- **Python Tkinter**: The Python Tkinter will be used to create user interfaces. Tkinter for Python is a GUI widget. Tkinter is a blend of Python programming language and the Tkinter library. It is suitable for developing graphical applications (Herrmann, 2020).
- **MiniMax**: MiniMax algorithm with Alpha-beta pruning will be used for the AI part of the game
- **Python unittest**: Unit testing will be done by using the Python unittest. The Python unittest unit testing framework is inspired by JUnit, and it supports test automation, unit testing.
- **Fedora**: The project will be implemented using the Fedora 32 system. Fedora is a Linux operating system with a complete set of tools for developing a project.
- **Github**: Github will be used for version control and source code management.
- **Google Authenticator**: Google Authenticator is a software-based authenticator by Google that implements two-step verification services using the Time-based One-time Password Algorithm and HMAC-based One-time Password algorithm for authenticating users of software applications

7. System/Software Architecture Diagram

7.1 Software UML Diagram

The following diagram is a software architecture diagram that visually represents how players interact with one another and how the software modules interact with one another.

The blocks in the diagram include:

- **Players**: Players take actions on playing the game, communicating with one another. A player versus player mode requires at least two players in a match.
- **Physical**: Data encryption will be done in the physical layer. In end-to-end encryption, the data is encrypted on the sender's system or device, and only the recipient is able to decrypt it. Nobody in the transport layer could read it or tamper with it.
    - The server would be the player who created a game room
    - The client would be the player who joins a game room
- **Application**: It will deal with network protocols, interact with them and present data change to players.
    - Player versus player: In this mode, players can match against another player based on the game rules.
    - Player versus Computer: In this mode, players can match against AI based on the game rules.

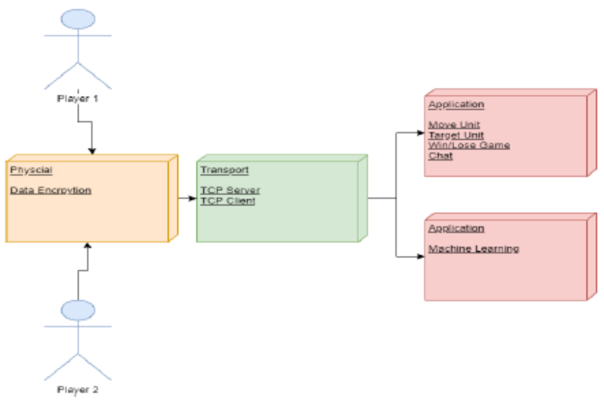


Figure 2 Software UML Diagram

7.2 Software Class Diagram

The following diagram is a software class diagram that visually represents how players interact with one another and how the software modules interact with one another.

- **UI Controller**: UI controller is also the central controller that integrates, controls, and calls every class function. It also generates UI/UX components:
  - Generate players' units
  - Generate a map
  - Generate action buttons
  - Generate chat panel
- **Players**: Player class contains functions that record players' names, IP addresses and ports.
- **Match**: Match class contains functions:
  - Win: Validate if one player wins the game
  - Lose: Validate if one player loses the game
  - Leave: Leave the game and the game room
  - Join: Join a game room and a match
- **Units**: Units class contains functions:
  - Move: Units could move to target positions
  - Attack: Units could attack an opponent's units
  - Be Destroyed: Units could be destroyed by opponents' based on game rules
  - Check Game Rules: Validate if a unit's current action is processable.
- **AI**: AI class contains the minimax algorithm that a computer could move units, attack units fast and accurate based on game rules.
  - Get the largest value of chess unit of the two sides of a unit
  - Calculate the score of the current board
  - AI seeks to maximize the board score
  - Player seeks to minimize the board score

- Move: Computer understands units could move to target positions
- Attack: Computer understands units could attack an opponent's units

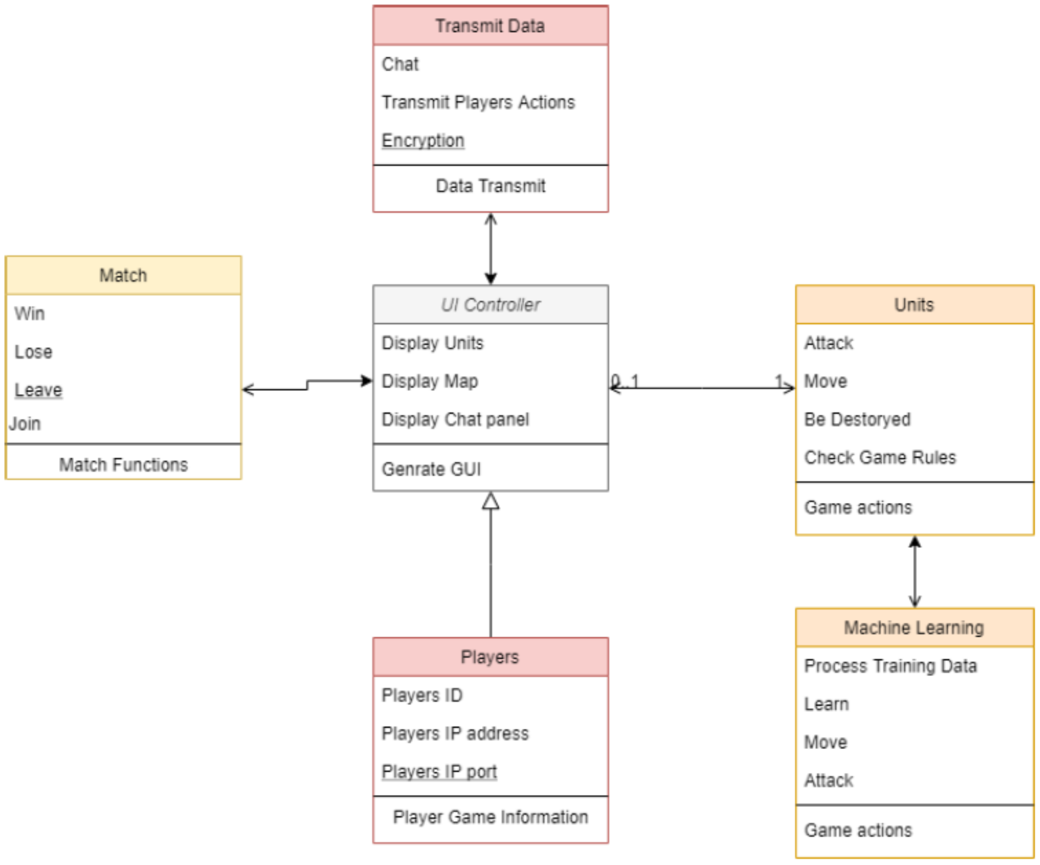


Figure 3 Software Class Diagram

8. Innovation

Play Mode: The current existing game in the Google Play store is four players mode. There will be two players mode for this game.

Chat: There will be a chat function that players can interact with their opponents during the game.

Encrypted data transfer: I will also add a live chat function for the game so players can communicate. But data transfer during the live chat will be encrypted by end-to-end encryption.

End to end encryption is the process in which encryption of data is being done at the end host. It is an implementation of asymmetric encryption.

RSA algorithm will be used for this project. RSA algorithm is an asymmetric encryption algorithm. RSA requires a public key and a private key and works as follow:

1. Person A sends the public key to person B and requests some data
2. Person B encrypts the data using person A's public key and sends the encrypted data
3. Person A receives the data and decrypts it.

AI: There will be a single-player mode for the game that players can match with AI. The AI algorithm will use miniMax with heuristics and alpha-beta pruning.

The MiniMax algorithm is a recursive algorithm usually used in decision-making games such as chess games. MiniMax uses recursion to search through the game tree. If two players play a game with the MiniMax algorithm, both the players will fight their opponent player to get the maximum benefit where their opponent gets the minimum benefit.

The heuristic algorithm is to find solutions among all possible solutions. However, it does not guarantee that the best solution will be found; therefore, it can be considered not accurate algorithms. The heuristic algorithm usually finds a solution close to the best one, and it finds it fast.

Alpha-beta pruning is an optimization technique for the minimax algorithm. Because there are 50 chess units for a game, alpha-beta pruning reduces the computation time. This allows the AI to search much faster and even dig deeper into the game tree that it also allows me to implement an easy AI mode and a hard AI mode. Alpha-beta will cut off branches in the game tree that will not be searched because there are better moves that exist.

Two-factor Authentication: Two-factor authentication is an electronic authentication method in which a device user is granted access to a website or application only after successfully presenting two or more pieces of evidence to an authentication mechanism.

For this project, the client-side player will need to use a valid Gmail account to connect to the server-side player. If one player enables Two-Factor Authentication for Gmail, the player goes through a setup process that first configures his account to send his phone a code by text or voice. At the end of the setup, he will have an opportunity to add Google Authenticator as an alternative code generator. He will use Google Authenticator to enter a 32-character secret string that he will use to add configuration to this account in the authenticator. Once done, the player will receive an authentication code in the Google Authenticator when he wants to join the game.

9. Complexity

Synchronize issue: this game needs to solve the synchronize problem. How the game can handle each command, such as move the unit, attack the opponent and sync all the steps so players can be in the same stage, instead of one player already wins the game and the other player is still moving his units.

Lagging: The game also needs to solve the lagging problem. For example, a player who is 50 pings should wait for a player with 100 pings.

Algorithm: Because the game contains about 50 units per match, I need to create an algorithm that allows each unit to move, attack as players want. Also, an AI algorithm for the game requires me to research and study how to do AI for a game.

Art design: I want each unit in the game to feel like that unit has a height instead of just a 2D image. so, the art design is somehow important for this game. This will make the game feels more interactive.

Overall, this game requires students to have proper art design, socket, Python, Python GUI, and algorithm skills to finish the game development. A diploma student should not be able to do such comprehensive work within 350 hours.

10. Technical Challenges

For this game, I would use TCP/IP communication skills, socket skills I learned from the Btech program to handle player connections and set up encrypted data transmission.

Programming skills: I need to improve my Python skills more because BCIT never really taught us about this language. I would not use Unity for this game but building the game from scratch. So, I need to learn how to GUI programming based on Python. Also, I need to search for how to create links to make the player invitation function work.

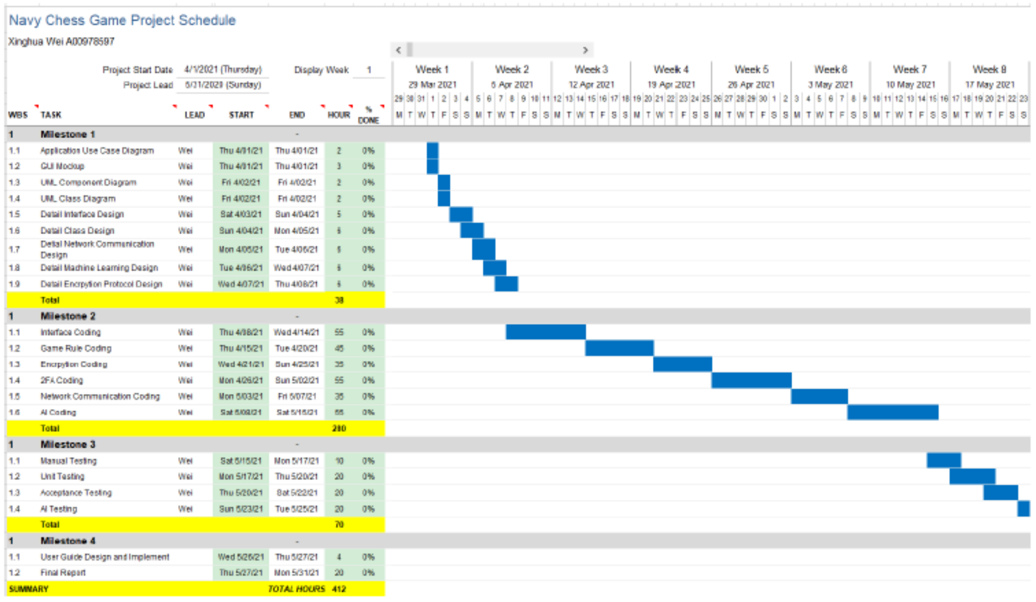
Testing: Testing is also a big barrier I need to conquer. Because I have no experience testing a multiplayer game build by Python. I need to research methods and tools to perform a proper test.

Time: This game requires a student to have proper art design, socket, Python, Python GUI, AI algorithm skills to finish the game within 350 hours. Skills like art design, GUI, AI are fairly new to me. So, after researching and learning some of the skills, I may need to extend my schedule or limit my scope.

Encryption: End-to-end encryption is a new subject for me because my experience with encryption is to encrypt data with only one layer of protection, such as RSA encryption or AES encryption. End-to-end requires a secure channel with a combination of encryption methods, such as RSA with AES and SHA. I may need to spend time researching and understand the technology, which could cause me to extend the schedule.

2FA: Apply two-factor authentication to the game is a challenge. I need to combine the authentication with a TCP socket, Python and an external mobile application. I may need to spend time researching and understand the technology, which could cause me to extend the schedule.

11. Development Schedule and Milestones



Navy Chess Game Project Schedule

Xinghua Wei A00978597

Project Start Date 4/1/2021 (Thursday) Display Week 1

Project Lead 5/31/2020 (Sunday)

WBS	TASK	LEAD	START	END	HOUR	% DONE
1	**Milestone 1**			-		
1.1	Application Use Case Diagram	Wei	Thu 4/01/21	Thu 4/01/21	2	0%
1.2	GUI Mockup	Wei	Thu 4/01/21	Thu 4/01/21	3	0%
1.3	UML Component Diagram	Wei	Fri 4/02/21	Fri 4/02/21	2	0%
1.4	UML Class Diagram	Wei	Fri 4/02/21	Fri 4/02/21	2	0%
1.5	Detail Interface Design	Wei	Sat 4/03/21	Sun 4/04/21	5	0%
1.6	Detail Class Design	Wei	Sun 4/04/21	Mon 4/05/21	6	0%
1.7	Detial Network Commenication Design	Wei	Mon 4/05/21	Tue 4/06/21	6	0%
1.8	Detail Machine Learning Design	Wei	Tue 4/06/21	Wed 4/07/21	6	0%
1.9	Detail Encrpytion Protocol Design	Wei	Wed 4/07/21	Thu 4/08/21	6	0%
	Total				38	
1	**Milestone 2**			-		
1.1	Interface Coding	Wei	Thu 4/08/21	Wed 4/14/21	55	0%
1.2	Game Rule Coding	Wei	Thu 4/15/21	Tue 4/20/21	45	0%
1.3	Encrpytion Coding	Wei	Wed 4/21/21	Sun 4/25/21	35	0%
1.4	2FA Coding	Wei	Mon 4/26/21	Sun 5/02/21	55	0%
1.5	Network Commenication Coding	Wei	Mon 5/03/21	Fri 5/07/21	35	0%
1.6	AI Coding	Wei	Sat 5/08/21	Sat 5/15/21	55	0%
	Total				280	
1	**Milestone 3**			-		
1.1	Manual Testing	Wei	Sat 5/15/21	Mon 5/17/21	10	0%
1.2	Unit Testing	Wei	Mon 5/17/21	Thu 5/20/21	20	0%
1.3	Acceptance Testing	Wei	Thu 5/20/21	Sat 5/22/21	20	0%
1.4	AI Testing	Wei	Sun 5/23/21	Tue 5/25/21	20	0%
	Total				70	
1	**Milestone 4**			-		
1.1	User Guide Design and Implement		Wed 5/26/21	Thu 5/27/21	4	0%
1.2	Final Report		Thu 5/27/21	Mon 5/31/21	20	0%
SUMMARY				*TOTAL HOURS*	412	

Each week is assumed to contain at least 50 work hours since I will be finishing this project when my other courses are finished, and the whole project will be finished in one year. Milestone one will be scheduled for 38 hours. Where milestone two will take most of the work hours, and it will be scheduled for 280 hours. Milestone three is the testing phase, which will take around 70 hours. Milestone four mainly is to wrap up the project and finish the project report.

Most of the estimates in the Gantt Chart above are estimates conservatively. More time may be allocated to each phase, especially for milestone two and milestone three.

Page **24** of **26**
Xinghua Wei – A00978597

12. Deliverables

The deliverables for this project are as follow:

- **Navy Chess Game Client**: A playable digitalized Navy Chess game
  - Game UI
  - Playable Game
  - Chat function with end-to-end encryption Communication
  - AI capability
- **Final Report**: Final report for the major project
- **User Guide**: User Guide to explain game rules and how to use the game client

13. Conclusion and Expertise Development

In conclusion, the digitalized Navy Chess game is a huge challenge for me. I will develop a fully functional digitalized 2D Navy Chess game for the Windows and Linux platforms. I need to research and develop AI algorithms for the game so players can match with AI. Chat function and end-to-end encryption are essential for the game's security aspect to prevent potential data leaks and intercept during transmission. Also, players can send a link to their friends so that players can invite another player to his match room.

The Navy Chess game project will help enhance many of my skills in terms of software development:

- Software architecture design
- Software API design
- Software testing
- 2D art design
- Project time and schedule management
- Project scope control

- Python programming, Python GUI programming, Python Socket programming
- Networking programming, network security programming
- Decision-making algorithms

My overall skills in programming, algorithms, managing, networking and designing will be improved as I overcome the technical challenges and achieve the innovation part of the project.

14. References

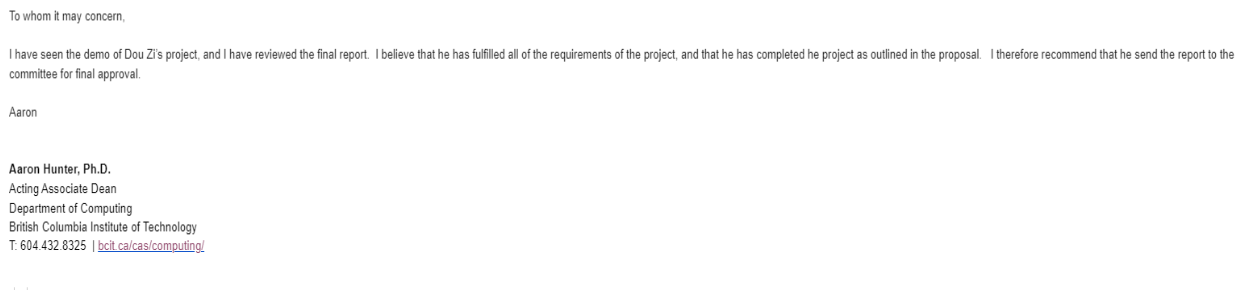
Herrmann, M. (n.d.). PyQt5 tutorial 2020: Create a GUI with Python and Qt. Retrieved October 28, 2020, from https://build-system.fman.io/pyqt5-tutorial

15. Change Log

Version 0.0:

- Nothing

4.2. Project Supervisor Approvals



To whom it may concern,

I have seen the demo of Dou Zi's project, and I have reviewed the final report. I believe that he has fulfilled all of the requirements of the project, and that he has completed he project as outlined in the proposal. I therefore recommend that he send the report to the committee for final approval.

Aaron

Aaron Hunter, Ph.D.
Acting Associate Dean
Department of Computing
British Columbia Institute of Technology
T: 604.432.8325 | bcit.ca/cas/computing/

5. References

Herrmann, M. (n.d.). PyQt5 tutorial 2020: Create a GUI with Python and Qt. Retrieved October 28, 2020, from https://build-system.fman.io/pyqt5-tutorial

6. Change Log

- [List all changes made from previous versions, referencing the page numbers where changes were made. Delete this section if not applicable.]