Introduction to Computer Science

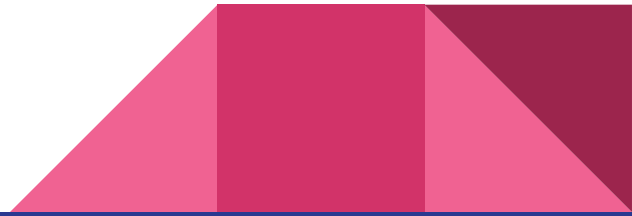
Jinyu Zhao, Wei Jie Chua

Game Development

- Introduction
- Live Showcase
- Analysis of Design Choices

Introduction

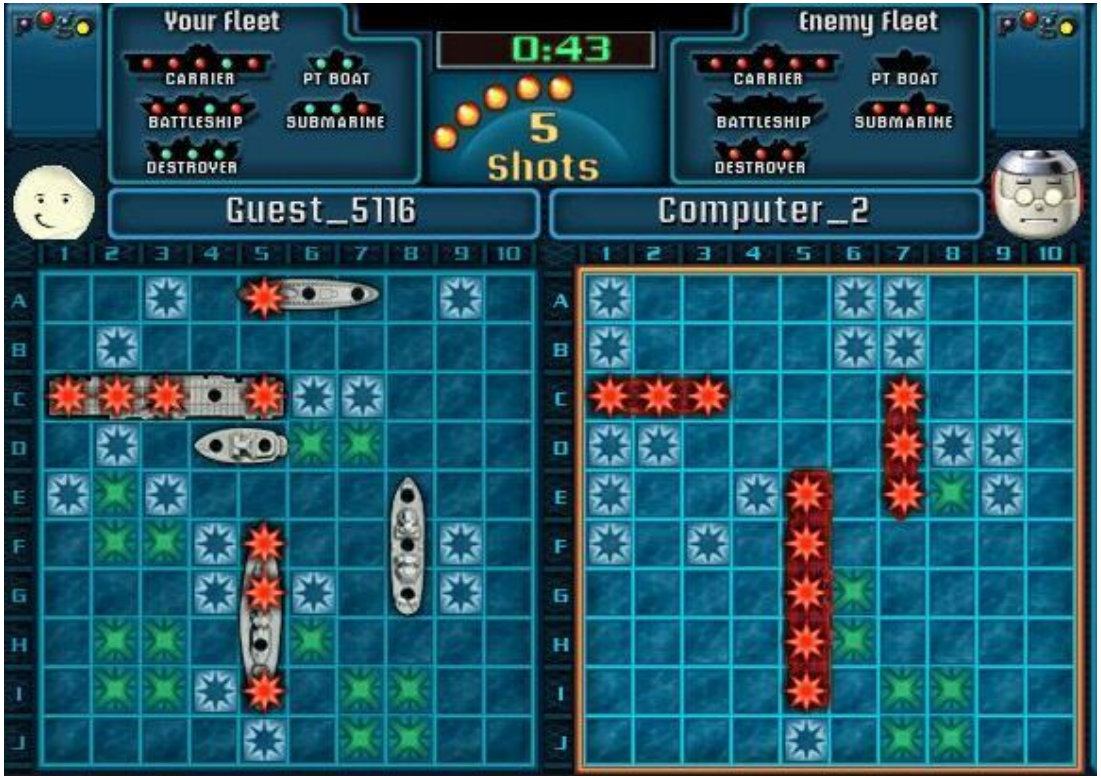
- Game development
- Both of us are gamers




HEROES
OF THE STORM™

Idea

- Battleship
  - First player to hit all the ship wins
  - And FUN!
  - (Not so compared to Heroes of the Storm)



Introduction

- Characteristics of the game
  - Utilize the chat system
  - Two-players
  - Take turns - Each player makes a move, and the following move will be determined by opponent.
  - BUT TO MAKE IT FUN, WE DECIDE TO HAVE IT REAL TIME SHOOTING - You can fire at will, don’t have to wait for your opponent

Live Demo

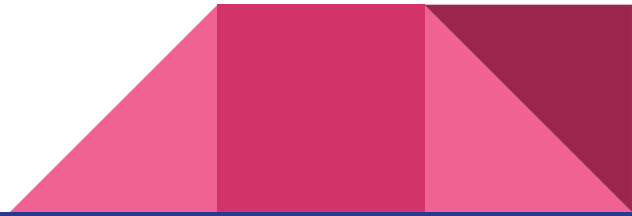
Discussion

- There are two main parts:
  - Gaming class
    - Class of Battlefield
    - Class of Ship
  - Incorporating into chat system
    - chat_server.py
    - client_state_machine.py

Gaming Class

Gaming Class

- Two classes:
  - Ship()
  - Battlefield()



Class Ship() - Method

```
71 class Ship:
72     def __init__(self, length, direction):
73         self.length = length
74         self.direction = direction
75         self.actual_location = []
```

- Creation of ship on the battlefield

Class Ship() - Method

```
81      def set_ship(self, location_x, location_y):
82          if self.direction == "U":
83              for i in range(self.length):
84                  the_boat = [location_x, location_y + i]
85                  self.actual_location.append(the_boat)
86                  #self.battlefield[location_y + length][location_x] = "x"
87          else:
88              for i in range(self.length):
89                  the_boat = [location_x + i, location_y]
90                  self.actual_location.append(the_boat)
91                  #self.battlefield[location_y][location_x + length] = "x"
```

- Recording the actual location of the boat in a list to avoid showing directly on the battlefield

Class Battlefield() - Attributes

```
 8 class Battlefield:
 9     def __init__(self, width = 8, length = 8):
10         self.battlefield = [['o' for i in range(width + 1)] for j in range(length + 1)]
11         for i in range(width + 1):
12             self.battlefield[0][i] = str(i)
13         for j in range(length + 1):
14             self.battlefield[j][0] = str(j)
15         self.width = width
16         self.length = length
17         self.strbattlefield = ''
```

- Attributes: Creating a list of list with width, length, and strbattlefield
- Notes: strbattlefield (For transmission through mysend and myrecv - they only accept string)

Class Battlefield() - Method

```
20      def update_battlefield(self, x_coordinate, y_coordinate, Ship):
21          the_point = [x_coordinate, y_coordinate]
22          location = []
23          for ship in Ship:
24              location += ship.actual_location
25          for actual_boat in location:
26              print (actual_boat)
27              if the_point == actual_boat:
28                  self.battlefield[y_coordinate][x_coordinate] = 'x'
29                  break
30          if the_point not in location:
31              self.battlefield[y_coordinate][x_coordinate] = ' '
```

- If hit the ship, the coordinate will change from ‘o’ to ‘x’
- Otherwise, it will change to ‘ ‘ to indicates nothing is hit

Class Battlefield() - Method

```
34      def check_coordinate(self, x_coordinate, y_coordinate):
35          if self.battlefield[y_coordinate][x_coordinate] == 'x':
36              return False
37          elif self.battlefield[y_coordinate][x_coordinate] == ' ':
38              return False
39          else:
40              return True
```

- Preventive measure: To avoid choosing a coordinate that was chosen before

Class Battlefield() - Method

```
42      def init_my_battlefield(self, Ship):
43          location = []
44          for ship in Ship:
45              location += ship.actual_location
46          for actual_boat in location:
47              self.battlefield[actual_boat[1]][actual_boat[0]] = '@'
48
```

- Two battlefield: Opponent and myself
- Use for myself - Making own boat = ‘@’

Class Battlefield() - Method

```
57      def win_game(self):
58          count = 0
59          for i in range(self.length):
60              for j in range(self.width):
61                  if self.battlefield[i][j] == 'x':
62                      count += 1
63          if count == 9:
64              return True
65          else:
66              return False
```

- One player win when all points of the ships are hit

Incorporation into Chat System

Client -- Server

```
elif my_msg[0] == 'g':
    peer = my_msg[1:].strip()
    if self.game_to(peer) == True:
        self.state = S_GAMING
        self.out_msg += 'Connect to ' + peer + '. Game away!\n\n'
        self.out_msg += '-----------------------------------\n'
    else:
        self.out_msg += 'Connection unsuccessful\n'
else:
    self.out_msg += menu
```

```
def game_to(self, peer):
    msg = M_GAME + peer
    mysend(self.s, msg)
    response = myrecv(self.s)
    if response == (M_GAME+'ok'):
        self.peer = peer
        self.out_msg += 'You are connected with '+ self.peer + '\n'
        return (True)
    elif response == (M_GAME + 'busy'):
        self.out_msg += 'User is busy. Please try again later\n'
    elif response == (M_GAME + 'hey you'):
        self.out_msg += 'Cannot game with yourself (sick)\n'
    else:
        self.out_msg += 'User is not online, try again later\n'
    return(False)
```

```
if code == M_GAME:
    print('YES')
    #self.check = 1
    to_name = msg[1:]
    from_name = self.logged_sock2name[from_sock]
    if to_name == from_name:
        msg = M_GAME + 'hey you'
    # connect to the peer
    elif self.group.is_member(to_name):
        #to_sock = self.logged_name2sock[to_name]
        to_sock = self.logged_name2sock[to_name]
        self.group.connect(from_name, to_name)
        the_guys = self.group.list_me(from_name)
        msg = M_GAME + 'ok'
        mysend(to_sock, M_GAME + from_name)
    else:
        msg = M_GAME + 'no user'
        #to_sock = self.logged_name2sock[to_name]
    mysend(from_sock, msg)
```

Client -- Server

Three sub-states under game mode

```
G_INIT = 6
G_START = 7
G_HOLDING = 8
```

```
self.gstate = G_INIT
```

Client -- Server

```
def game_init(self, msg):
    mysend(self.s, M_GINIT + msg)
    response = myrecv(self.s)
    #print(response)
    if response[0] == (M_GSTART):
        self.out_msg += response[1:]
        self.out_msg += 'GAME START\n'
        self.gstate = G_START
        return (True)
    if response[0] == M_GHOLDING:
        #print('fuck')
        #print(self.out_msg+'fuck1\n')
        self.out_msg += response[1:]
        self.out_msg += 'Wait for your oppenent...\n'
        self.gstate = G_HOLDING
        #.print(self.out_msg+'fuck2\n')
        return (False)
    if response == (M_GINIT + 'invalid'):
        self.out_msg += 'Please enter valid location and direction.
    return (False)
```

```
if code == M_GINIT:
    his_battlefield = bf.Battlefield(9, 9)
    my_battlefield = bf.Battlefield(9, 9)
    para = msg[1:].split(',')
    s_msg = ''
    from_name = self.logged_sock2name[from_sock]
    the_guys = self.group.list_me(from_name)
    to_name = the_guys[1]
    to_sock = self.logged_name2sock[to_name]
    if para[0] in self.valid_para and para[1] in self.valid_para and para[3] in self.
        if self.location == {}:
            s_msg += M_GHOLDING
        else:
            s_msg = M_GSTART
            t_msg = M_GSTART + 'Choose your target>> \n'
            mysend(to_sock, t_msg)
        # initate ship
        ship1 = bf.Ship(3, para[2])
        ship2 = bf.Ship(3, para[5])
        ship = [ship1, ship2]
        ship1.set_ship(int(para[0]), int(para[1]))
        ship2.set_ship(int(para[3]), int(para[4]))
        # store ship&battlefield information
        self.location[self.logged_sock2name[from_sock]] = ship
        #print(self.location)
        my_battlefield.init_my_battlefield(ship)
        self.my_battlefield[self.logged_sock2name[from_sock]] = my_battlefield
        self.his_battlefield[self.logged_sock2name[from_sock]] = his_battlefield
        my_battlefield.battlefield2string()
        # handle_msg
        s_msg += 'My battlefield:\n'
        s_msg += my_battlefield.strbattlefield
        #print(s_msg)
        mysend(from_sock, s_msg)

    else:
        s_msg = M_GINIT + 'invalid'
        mysend(from_sock, s_msg)
```

Client -- Server >> G_START

```
elif self.gstate == G_START:
    if len(my_msg) > 0:
        if my_msg == 'gg':
            self.disconnect()
            self.state = S_LOGGEDIN
            self.peer = ''
        elif self.check_coordinates(my_msg):
            self.ggstate == G_HOLDING1
        else:
            self.out_msg += 'Target invalid, please try again... \
    if len(peer_msg) > 0:    # peer's stuff, coming in
            self.out_msg += peer_msg
            if peer_msg[-1] != 'G':
                self.out_msg += 'Your turn ... Choose a target'
            self.ggstate = 9
```

```
def hold2start(self,msg):
    mysend(self.s, M_GHOLDING + msg)
    response = myrecv(self.s)
    if response == 'ok':
        return (True)
    else:
        return (False)

def check_coordinates(self, msg):
    self.out_msg = ''
    mysend(self.s, M_GSTART + msg)
    response = myrecv(self.s)
    if response == (M_GSTART + 'invalid'):
        self.out_msg += 'Please enter valid target...\n'
        return (False)
    else:
        self.out_msg += response[1:]
        return (True)
```

```
elif code == M_GSTART:
    s_msg = ''
    # divide information
    from_name = self.logged_sock2name[from_sock]
    the_guys = self.group.list_me(from_name)
    print(the_guys)
    to_name = the_guys[1]
    to_sock = self.logged_name2sock[to_name]
    my_ship = self.location[from_name]
    his_ship = self.location[to_name]
    his_my_battlefield = self.my_battlefield[to_name]
    my_his_battlefield = self.his_battlefield[to_name]
    X = int(msg[1])
    Y = int(msg[3])
    if my_his_battlefield.check_coordinate(X, Y) == False:
        s_msg = M_GSTART + 'invalid'
        mysend(from_sock, s_msg)
    else:
        msg1 = M_GSTART
        msg2 = M_GSTART
        my_his_battlefield.update_battlefield(X, Y, his_ship)
        his_my_battlefield.update_battlefield(X, Y, his_ship)
        my_his_battlefield.battlefield2string()
        his_my_battlefield.battlefield2string()
        msg1 += "----BOOM----\nYour opponent's battlefield: \n"
        msg1 += my_his_battlefield.strbattlefield
        msg1 += 'Wait for your opponent ...\n'
        if my_his_battlefield.win_game():
            msg1 += "Congratualations, you win ..."
        mysend(from_sock, msg1)
        msg2 += "----BOOM----\nYour battlefield: \n"
        msg2 += his_my_battlefield.strbattlefield
        msg2 += 'Wait for your opponent ...\n'
        if his_my_battlefield.win_game():
            msg2 += 'Sorry, you lose ... GG'
        mysend(to_sock, msg2)
```

Thank you!