C:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Lost_badge.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Destroyer_True3.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Submarine_False3.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Cruiser_False3.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Aircraft_Carrier_False3.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Cruiser_False.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Destroyer_True.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Win_badge.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Submarine_True.pngC:\Users\Soham\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Aircraft_Carrier_False22.png

BATTLESHIP!!

Computer Science Project



**User Manual**

This project is a manifestation of the classic game, Battleship. Rules are the same as the original game, but for those playing for the first time, here is a basic set of rules:-

1. There will be two boards on the screen, yours is on the left, and the computers is on the right.

2. Select the ship tokens one by one and align them using arrow keys, on your board.

3. Click the box on your board to place the ship token.

4.Your goal is to sink all the ships of the computer, before it does yours!

**Let’s Ride the Seas, Admiral!**

CERTIFICATE

Amity International School

*This is to certify that ­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_, Roll number*

*\_\_\_\_\_\_\_\_ of Class \_\_\_ has successfully completed the \_\_\_\_\_\_\_\_\_\_\_\_\_\_ under my supervision according to the guidelines laid down by the CBSE.*

*Teacher Incharge*

Index

* **Certificate**
* **Acknowledgement**
* **About the project**
* **Full Source Code**
* **Screenshots of the project**
* **Classes and Modules used**
* **User Manual**
* **Future enhancement**
* **Bibliography**

**Acknowledgement**

I would like to thank my teacher Ms Kirti Tripathi for giving me the opportunity of working on this exciting project. It would not have been possible for me to complete it without her constant guidance.

I would also like to thank my teacher Ms Geetika Kapoor for clearing all my doubts regarding the code.

I would like to thank my parents for helping me gather the resources essential for completing the project.

**About the Project**

The project is based on the age old paper and pen game called battleship. In order to create it, we used many of the concepts we learnt in the past year like Object Oriented Programming. The toughest challenges however were not in implementing these but writing the algorithm for the gameplay. While it is fairly simple in real life (see where you’ve hit the opposition and then check the cells around it), the code to implement came as a challenge. Another major challenge faced was the implementation of the user interface and it was mutually agreed that we should all learn the pygame module in order to give a good presentation of the project.

**Full Source Code**

import pygame

import random

import time

from math import \*

pygame.init()

display\_x = 1040

display\_y = 680

aircraft\_carrier\_image\_false = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Aircraft\_Carrier\_False.png")

aircraft\_carrier\_image\_true = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Aircraft\_Carrier\_True.png")

cruiser\_image\_false = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Cruiser\_False.png")

cruiser\_image\_true = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Cruiser\_True.png")

submarine\_image\_false = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Submarine\_False.png")

submarine\_image\_true = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Submarine\_True.png")

destroyer\_image\_false = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Destroyer\_False.png")

destroyer\_image\_true = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Destroyer\_True.png")

aircraft\_carrier\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Aircraft\_Carrier\_button.png")

cruiser\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Cruiser\_button.png")

submarine\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Submarine\_button.png")

destroyer\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Destroyer\_button.png")

aircraft\_carrier\_button\_onclick = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Aircraft\_Carrier\_button\_onclick.png")

cruiser\_button\_onclick = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Cruiser\_button\_onclick.png")

submarine\_button\_onclick = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Submarine\_button\_onclick.png")

destroyer\_button\_onclick = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Destroyer\_button\_onclick.png")

miss\_cell = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Miss\_cell.png")

hit\_cell = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Hit\_cell.png")

orient\_true = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/orient\_true.png")

orient\_false = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/orient\_false.png")

orient\_none = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/orient\_none.png")

replay\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Replay\_button.png")

replay\_button\_onclick = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Replay\_button\_onclick.png")

quit\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Quit\_button.png")

quit\_button\_onclick = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Quit\_button\_onclick.png")

win\_badge = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Win\_badge.png")

lost\_badge = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Lost\_badge.png")

homepage = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Homepage.png")

play\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Play\_button\_1.png")

how\_to\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/How\_to\_play.png")

back\_to\_menu\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Back\_to\_menu.png")

quit\_2\_button\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Quit\_2.png")

turn\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Turn.png")

guest\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Guest.png")

computer\_image = pygame.image.load("C:/Users/Mehul/Personal/School material/Battleship/Computer.png")

x = display\_x / 26

y = display\_y / 17

black = (0, 0, 0)

back\_blue = (30, 149, 189)

white = (255, 255, 255)

sea\_blue = (89, 96, 112)

red = (255, 0, 0)

green = (0, 232, 0)

Display = pygame.display.set\_mode((display\_x, display\_y))

pygame.display.set\_caption("Battleship")

font = pygame.font.SysFont("Arial", 24)

user\_cells\_ref = ['0121', '0122', '0123', '0124', '0125', '0126', '0127', '0128', '0129', '0130', '0221', '0222', '0223', '0224', '0225', '0226', '0227', '0228', '0229', '0230', '0321', '0322', '0323', '0324', '0325', '0326', '0327', '0328', '0329', '0330', '0421', '0422', '0423', '0424', '0425', '0426', '0427', '0428', '0429', '0430', '0521', '0522', '0523', '0524', '0525', '0526', '0527', '0528', '0529', '0530', '0621', '0622', '0623', '0624', '0625', '0626', '0627', '0628', '0629', '0630', '0721', '0722', '0723', '0724', '0725', '0726', '0727', '0728', '0729', '0730', '0821', '0822', '0823', '0824', '0825', '0826', '0827', '0828', '0829', '0830', '0921', '0922', '0923', '0924', '0925', '0926', '0927', '0928', '0929', '0930', '1021', '1022', '1023', '1024', '1025', '1026', '1027', '1028', '1029', '1030']

AI\_cells\_ref = ['1121', '1122', '1123', '1124', '1125', '1126', '1127', '1128', '1129', '1130', '1221', '1222', '1223', '1224', '1225', '1226', '1227', '1228', '1229', '1230', '1321', '1322', '1323', '1324', '1325', '1326', '1327', '1328', '1329', '1330', '1421', '1422', '1423', '1424', '1425', '1426', '1427', '1428', '1429', '1430', '1521', '1522', '1523', '1524', '1525', '1526', '1527', '1528', '1529', '1530', '1621', '1622', '1623', '1624', '1625', '1626', '1627', '1628', '1629', '1630', '1721', '1722', '1723', '1724', '1725', '1726', '1727', '1728', '1729', '1730', '1821', '1822', '1823', '1824', '1825', '1826', '1827', '1828', '1829', '1830', '1921', '1922', '1923', '1924', '1925', '1926', '1927', '1928', '1929', '1930', '2021', '2022', '2023', '2024', '2025', '2026', '2027', '2028', '2029', '2030']

user\_unused\_cells = ['0121', '0122', '0123', '0124', '0125', '0126', '0127', '0128', '0129', '0130', '0221', '0222', '0223', '0224', '0225', '0226', '0227', '0228', '0229', '0230', '0321', '0322', '0323', '0324', '0325', '0326', '0327', '0328', '0329', '0330', '0421', '0422', '0423', '0424', '0425', '0426', '0427', '0428', '0429', '0430', '0521', '0522', '0523', '0524', '0525', '0526', '0527', '0528', '0529', '0530', '0621', '0622', '0623', '0624', '0625', '0626', '0627', '0628', '0629', '0630', '0721', '0722', '0723', '0724', '0725', '0726', '0727', '0728', '0729', '0730', '0821', '0822', '0823', '0824', '0825', '0826', '0827', '0828', '0829', '0830', '0921', '0922', '0923', '0924', '0925', '0926', '0927', '0928', '0929', '0930', '1021', '1022', '1023', '1024', '1025', '1026', '1027', '1028', '1029', '1030']

AI\_unused\_cells = ['1121', '1122', '1123', '1124', '1125', '1126', '1127', '1128', '1129', '1130', '1221', '1222', '1223', '1224', '1225', '1226', '1227', '1228', '1229', '1230', '1321', '1322', '1323', '1324', '1325', '1326', '1327', '1328', '1329', '1330', '1421', '1422', '1423', '1424', '1425', '1426', '1427', '1428', '1429', '1430', '1521', '1522', '1523', '1524', '1525', '1526', '1527', '1528', '1529', '1530', '1621', '1622', '1623', '1624', '1625', '1626', '1627', '1628', '1629', '1630', '1721', '1722', '1723', '1724', '1725', '1726', '1727', '1728', '1729', '1730', '1821', '1822', '1823', '1824', '1825', '1826', '1827', '1828', '1829', '1830', '1921', '1922', '1923', '1924', '1925', '1926', '1927', '1928', '1929', '1930', '2021', '2022', '2023', '2024', '2025', '2026', '2027', '2028', '2029', '2030']

user\_used\_cells = []

AI\_used\_cells = []

top\_edge = ["0221", "0321", "0421", "0521", "0621", "0721", "0821", "0921"]

right\_edge = ["1022", "1023", "1024", "1025", "1026", "1027", "1028", "1029"]

left\_edge = ["0122", "0123", "0124", "0125", "0126", "0127", "0128", "0129"]

bottom\_edge = ["0230", "0330", "0430", "0530", "0630", "0730", "0830", "0930"]

AI\_hit\_cells = []

user\_hit\_cells = []

AI\_missed\_cells = []

user\_missed\_cells = []

AI\_targeted = []

last\_click = ""

class board:

    def \_\_init\_\_(self, pos\_x, pos\_y, cells\_x, cells\_y, cell\_width\_x, cell\_width\_y, Display, board\_color, line\_color):

        self.pos\_x = pos\_x

        self.pos\_y = pos\_y

        self.cells\_x = cells\_x

        self.cells\_y = cells\_y

        self.cell\_width\_x = cell\_width\_x

        self.cell\_width\_y = cell\_width\_y

        self.board\_color = board\_color

        self.line\_color = line\_color

        self.display = Display

        pygame.draw.rect(self.display, self.board\_color, [self.pos\_x, self.pos\_y, self.cells\_x\*self.cell\_width\_x, self.cells\_y\*self.cell\_width\_y])

        for x in range(self.pos\_x, (self.pos\_x + (self.cells\_x\*self.cell\_width\_x)), self.cell\_width\_x):

            for y in range(self.pos\_y, (self.pos\_y + (self.cells\_y\*self.cell\_width\_y)), self.cell\_width\_y):

                pygame.draw.rect(self.display, self.line\_color, [x, y, self.cell\_width\_x, self.cell\_width\_y], 1)

        pygame.display.update()

    def draw\_lines\_again(self):

        for x in range(self.pos\_x, (self.pos\_x + (self.cells\_x\*self.cell\_width\_x)), self.cell\_width\_x):

            for y in range(self.pos\_y, (self.pos\_y + (self.cells\_y\*self.cell\_width\_y)), self.cell\_width\_y):

                pygame.draw.rect(self.display, self.line\_color, [x, y, self.cell\_width\_x, self.cell\_width\_y], 1)

        pygame.display.update()

class button:

    def \_\_init\_\_(self, pos\_x, pos\_y, size\_x, size\_y, onclickfunction, image, state = True):

        self.pos\_x = pos\_x

        self.pos\_y = pos\_y

        self.size\_x = size\_x

        self.size\_y = size\_y

        self.state = state

        self.onclick = onclickfunction

        self.image = image

        if self.state:

            Display.blit(self.image, (pos\_x, pos\_y))

    def on\_click(self):

        if not self.state:

            return -1

        else:

            self.onclickfunction()

    def set\_false\_state(self):

        self.state = False

        pygame.draw.rect(Display, back\_blue, [self.pos\_x, self.pos\_y, self.size\_x, self.size\_y])

    def set\_true\_state(self):

        self.state = True

        Display.blit(self.image, (self.pos\_x, self.pos\_y))

    def cell\_cover(self):

        l = []

        for i in range(self.pos\_x, self.pos\_x + self.size\_x, x):

            for j in range(self.pos\_y, self.pos\_y + self.size\_y, y):

                l.append((i, j))

        return l

class ship:

    def \_\_init\_\_(self, pos\_x, pos\_y, orient, length, user\_trigger = False):

        self.pos\_x = pos\_x

        self.pos\_y = pos\_y

        self.orient = orient

        self.length = length

        self.occupied = []

        if self.orient:

            for i in range(pos\_x, pos\_x + self.length):

                if len(str(i)) == 1:

                    i = "0" + str(i)

                else:

                    i = str(i)

                self.occupied.append(i + str(pos\_y))

        elif not self.orient:

            for j in range(pos\_y, pos\_y + self.length):

                if len(str(pos\_x)) == 1:

                    i = str(0) + str(pos\_x)

                else:

                    i = str(pos\_x)

                self.occupied.append(i + str(j))

        if not user\_trigger:

            for i in self.occupied:

                AI\_unused\_cells.remove(i)

                AI\_used\_cells.append(i)

        if user\_trigger:

            for i in self.occupied:

                user\_unused\_cells.remove(i)

                user\_used\_cells.append(i)

    def \_\_str\_\_(self):

        return str(self.orient) + " " + str(self.pos\_x) + " " + str(self.pos\_y)

    @staticmethod

    def calculate\_usable\_cells(orient, length):

        if orient == None:

            pass

        unusable = []

        usable = []

        if orient:

            for i in AI\_used\_cells:

                xc = int(i[0:2])

                yc = (i[2:4])

                for j in range(xc - length + 1, xc):

                    if len(str(j)) == 1:

                        j = "0" + str(j)

                    unusable.append(str(j) + yc)

        if not orient:

            for i in AI\_used\_cells:

                xc = i[0:2]

                yc = int(i[2:4])

                for j in range(yc - length + 1, yc):

                    unusable.append(xc + str(j))

        for i in AI\_cells\_ref:

            if i not in unusable and i not in AI\_used\_cells:

                if orient:

                    if int(i[1]) < 12 - length and int(i[0:2]) != 20:

                        usable.append(i)

                if not orient:

                    if int(i[3]) < 12 - length and int(i[2:4]) != 30:

                        usable.append(i)

        return usable

    @staticmethod

    def calculate\_usable\_cells\_user(orient, length):

        if orient == None:

            return -1

        usable = []

        unusable = []

        if orient:

            for i in user\_used\_cells:

                xc = int(i[0:2])

                yc = i[2:4]

                for j in range(xc - length + 1, xc):

                    if len(str(j)) == 1:

                        j = "0" + str(j)

                    unusable.append(str(j) + yc)

        if not orient:

            for i in user\_used\_cells:

                xc = i[0:2]

                yc = int(i[2:4])

                for j in range(yc - length + 1, yc):

                    unusable.append(xc + str(j))

        for i in user\_cells\_ref:

            if i not in unusable and i not in user\_used\_cells:

                if orient:

                    if int(i[1]) < 12 - length and int(i[0:2]) != 10:

                        usable.append(i)

                if not orient:

                    if int(i[3]) < 12 - length and int(i[2:4]) != 30:

                        usable.append(i)

        print "Usable", usable

        print "Unusable", unusable

        return usable

class aircraft\_carrier(ship):

    def \_\_init\_\_(self, pos\_x, pos\_y, orient, user\_trigger = False):

        if len(str(pos\_x)) == 1:

            pos\_x = "0" + str(pos\_x)

        if len(str(pos\_y)) == 1:

            pos\_y = "0" + str(pos\_y)

        pos\_x = int(pos\_x)

        pos\_y = int(pos\_y)

        ship.\_\_init\_\_(self, pos\_x, pos\_y, orient, 5, user\_trigger)

class cruiser(ship):

    def \_\_init\_\_(self, pos\_x, pos\_y, orient, user\_trigger = False):

        if len(str(pos\_x)) == 1:

            pos\_x = "0" + str(pos\_x)

        if len(str(pos\_y)) == 1:

            pos\_y = "0" + str(pos\_y)

        pos\_x = int(pos\_x)

        pos\_y = int(pos\_y)

        ship.\_\_init\_\_(self, pos\_x, pos\_y, orient, 4, user\_trigger)

class submarine(ship):

    def \_\_init\_\_(self, pos\_x, pos\_y, orient, user\_trigger = False):

        if len(str(pos\_x)) == 1:

            pos\_x = "0" + str(pos\_x)

        if len(str(pos\_y)) == 1:

            pos\_y = "0" + str(pos\_y)

        pos\_x = int(pos\_x)

        pos\_y = int(pos\_y)

        ship.\_\_init\_\_(self, pos\_x, pos\_y, orient, 3, user\_trigger)

class destroyer(ship):

    def \_\_init\_\_(self, pos\_x, pos\_y, orient, user\_trigger = False):

        if len(str(pos\_x)) == 1:

            pos\_x = "0" + str(pos\_x)

        if len(str(pos\_y)) == 1:

            pos\_y = "0" + str(pos\_y)

        pos\_x = int(pos\_x)

        pos\_y = int(pos\_y)

        ship.\_\_init\_\_(self, pos\_x, pos\_y, orient, 2, user\_trigger)

def qsequence():

    pygame.quit()

    quit()

def blank\_function():

    pass

def message\_to\_screen(x, y, msg, color):

    msg = str(msg)

    screen\_text = font.render(msg, True, color)

    Display.blit(screen\_text, (x, y))

def cell\_to\_x\_and\_y(cell):

    a = int(cell[0:2])

    b = int(cell[2:4])

    if a < 11:

        a = (a + 1)\*x

    else:

        a = (a + 3)\*x

    b = (b - 19)\*y

    return (a, b)

def x\_and\_y\_to\_cell(t):

    a = t[0]

    b = t[1]

    if floor(a/x) < 13:

        m = str(int(floor(a/x)) - 1)

        if len(m) == 1:

            m = "0" + m

    else:

        m = str(int(floor(a/x)) - 3)

    n = str(int(floor(b/y)) + 19)

    return (m + n)

def top\_corner(t):

    a = t[0]

    b = t[1]

    at = x \* (a/x)

    bt = y \* (b/y)

    return (at, bt)

def cell\_int\_to\_str(cell\_int):

    if len(cell\_int) == 3:

        return "0" + (cell\_int)

    return (cell\_int)

def aircraft\_carrier\_button\_click():

    global last\_click

    if aircraft\_carrier\_button.state == True:

        last\_click = "Aircraft Carrier"

        Display.blit(aircraft\_carrier\_button\_onclick, (3\*x, 13\*y))

    return 5

def cruiser\_button\_click():

    global last\_click

    if cruiser\_button.state == True:

        last\_click = "Cruiser"

        Display.blit(cruiser\_button\_onclick, (5\*x, 13\*y))

    return 4

def submarine\_button\_click():

    global last\_click

    if submarine\_button.state == True:

        last\_click = "Submarine"

        Display.blit(submarine\_button\_onclick, (8\*x, 13\*y))

    return 3

def destroyer\_button\_click():

    global last\_click

    if destroyer\_button.state == True:

        last\_click = "Destroyer"

        Display.blit(destroyer\_button\_onclick, (10\*x, 13\*y))

    return 2

def AI\_Ship\_Set():

    orient\_5 = random.choice([0, 1])

    usable = ship.calculate\_usable\_cells(orient\_5, 5)

    a = random.randrange(0, len(usable) - 1)

    ship\_5 = aircraft\_carrier(int(usable[a][0:2]), int(usable[a][2:4]), orient\_5)

    orient\_4 = random.choice([0, 1])

    usable = ship.calculate\_usable\_cells(orient\_4, 4)

    a = random.randrange(0, len(usable) - 1)

    ship\_4 = cruiser(int(usable[a][0:2]), int(usable[a][2:4]), orient\_4)

    orient\_3 = random.choice([0, 1])

    usable = ship.calculate\_usable\_cells(orient\_3, 3)

    a = random.randrange(0, len(usable) - 1)

    ship\_3 = submarine(int(usable[a][0:2]), int(usable[a][2:4]), orient\_3)

    orient\_2 = random.choice([0, 1])

    usable = ship.calculate\_usable\_cells(orient\_2, 2)

    a = random.randrange(0, len(usable) - 1)

    ship\_2 = destroyer(int(usable[a][0:2]), int(usable[a][2:4]), orient\_2)

def user\_ship\_set():

    global last\_click

    global buttons

    last\_click\_orient = None

    while True:

        for event in pygame.event.get():

            if last\_click\_orient == None:

                Display.blit(orient\_none, (6.5\*x, 14\*y))

            elif last\_click\_orient:

                Display.blit(orient\_true, (6.5\*x, 14\*y))

            elif not last\_click\_orient:

                Display.blit(orient\_false, (6.5\*x, 14\*y))

            if event.type == 2:

                if event.scancode == 72 or event.scancode == 80:

                    last\_click\_orient = False

                if event.scancode == 75 or event.scancode == 77:

                    last\_click\_orient = True

            if event.type == 5:

                if top\_corner(event.pos) in back\_to.cell\_cover():

                    back\_to.onclick()

                if event.button == 1:

                    if (x\_and\_y\_to\_cell(event.pos)) == "0232":

                        aircraft\_carrier\_button\_click()

                    if (x\_and\_y\_to\_cell(event.pos)) == "0432":

                        cruiser\_button\_click()

                    if (x\_and\_y\_to\_cell(event.pos)) == "0732":

                        submarine\_button\_click()

                    if (x\_and\_y\_to\_cell(event.pos)) == "0932":

                        destroyer\_button\_click()

                if last\_click == "Aircraft Carrier":

                    try:

                        if x\_and\_y\_to\_cell(event.pos) in ship.calculate\_usable\_cells\_user(last\_click\_orient, 5):

                            aircraft\_carrier(int(x\_and\_y\_to\_cell(event.pos)[0:2]), int(x\_and\_y\_to\_cell(event.pos)[2:4]), last\_click\_orient, True)

                            last\_click = None

                            aircraft\_carrier\_button.set\_false\_state()

                            if last\_click\_orient:

                                Display.blit(aircraft\_carrier\_image\_true, top\_corner(event.pos))

                            elif not last\_click\_orient:

                                Display.blit(aircraft\_carrier\_image\_false, top\_corner(event.pos))

                    except TypeError:

                        pass

                if last\_click == "Cruiser":

                    try:

                        if x\_and\_y\_to\_cell(event.pos) in ship.calculate\_usable\_cells\_user(last\_click\_orient, 4):

                            cruiser(int(x\_and\_y\_to\_cell(event.pos)[0:2]), int(x\_and\_y\_to\_cell(event.pos)[2:4]), last\_click\_orient, True)

                            last\_click = None

                            cruiser\_button.set\_false\_state()

                            if last\_click\_orient:

                                Display.blit(cruiser\_image\_true, top\_corner(event.pos))

                            if not last\_click\_orient:

                                Display.blit(cruiser\_image\_false, top\_corner(event.pos))

                    except TypeError:

                        pass

                if last\_click == "Submarine":

                    try:

                        if x\_and\_y\_to\_cell(event.pos) in ship.calculate\_usable\_cells\_user(last\_click\_orient, 3):

                            submarine(int(x\_and\_y\_to\_cell(event.pos)[0:2]), int(x\_and\_y\_to\_cell(event.pos)[2:4]), last\_click\_orient, True)

                            last\_click = None

                            submarine\_button.set\_false\_state()

                            if last\_click\_orient:

                                Display.blit(submarine\_image\_true, top\_corner(event.pos))

                            if not last\_click\_orient:

                                Display.blit(submarine\_image\_false, top\_corner(event.pos))

                    except TypeError:

                        pass

                if last\_click == "Destroyer":

                    try:

                        if x\_and\_y\_to\_cell(event.pos) in ship.calculate\_usable\_cells\_user(last\_click\_orient, 2):

                            destroyer(int(x\_and\_y\_to\_cell(event.pos)[0:2]), int(x\_and\_y\_to\_cell(event.pos)[2:4]), last\_click\_orient, True)

                            last\_click == None

                            destroyer\_button.set\_false\_state()

                            if last\_click\_orient:

                                Display.blit(destroyer\_image\_true, top\_corner(event.pos))

                            if not last\_click\_orient:

                                Display.blit(destroyer\_image\_false, top\_corner(event.pos))

                    except TypeError:

                        pass

            if event.type == 12:

                qsequence()

            temp = []

            for i in [destroyer\_button, aircraft\_carrier\_button, submarine\_button, cruiser\_button]:

                if i.state == False:

                    temp.append(1)

            if len(temp) == 4:

                pygame.display.update()

                return -1

            pygame.display.update()

def check\_cell(cell, user\_trigger = False):

    if user\_trigger:

        if cell in AI\_used\_cells:

            return 1

    elif not user\_trigger:

        if cell in user\_used\_cells:

            return 1

    return 0

def around\_cell\_str\_function(integer):

    if len(str(integer)) != 2:

        return "0" + str(integer)

    else:

        return str(integer)

def around\_cell(cell):

    if cell == "0121":

        around = ["0122", "0221"]

    if cell == "1021":

        around = ["1022", "0921"]

    if cell == "0130":

        around = ["0129", "0230"]

    if cell == "1030":

        around = ["1029", "0929"]

    x = int(cell[0:2])

    y = int(cell[2:4])

    a1 = x - 1

    a2 = x + 1

    b1 = y - 1

    b2 = y + 1

    x = around\_cell\_str\_function(x)

    y = around\_cell\_str\_function(y)

    a1 = around\_cell\_str\_function(a1)

    a2 = around\_cell\_str\_function(a2)

    b1 = around\_cell\_str\_function(b1)

    b2 = around\_cell\_str\_function(b2)

    if cell in top\_edge:

        around = [a1 + y, a2 + y, x + b2]

    if cell in bottom\_edge:

        around = [a1 + y, a2 + y, x + b1]

    if cell in left\_edge:

        around = [x + b1, x + b2, a2 + y]

    if cell in right\_edge:

        around = [x + b1, x + b2, a1 + y]

    else:

        around = [x + b1, x + b2, a1 + y, a2 + y]

    return around

def redef\_targets():

    global AI\_targeted

    AI\_targeted = []

    for i in AI\_hit\_cells:

        xc = int(i[0:2])

        yc = int(i[2:4])

        if cell\_int\_to\_str(str(xc - 1) + str(yc)) in AI\_hit\_cells and cell\_int\_to\_str(str(xc - 2) + str(yc)) not in AI\_hit\_cells and cell\_int\_to\_str(str(xc - 2) + str(yc)) not in AI\_missed\_cells:

            AI\_targeted.append(cell\_int\_to\_str(str(xc - 2) + str(yc)))

        if cell\_int\_to\_str(str(xc + 1) + str(yc)) in AI\_hit\_cells and cell\_int\_to\_str(str(xc + 2) + str(yc)) not in AI\_hit\_cells and cell\_int\_to\_str(str(xc + 2) + str(yc)) not in AI\_missed\_cells:

            AI\_targeted.append(cell\_int\_to\_str(str(xc + 2) + str(yc)))

        if cell\_int\_to\_str(str(xc) + str(yc - 1)) in AI\_hit\_cells and cell\_int\_to\_str(str(xc) + str(yc - 2)) not in AI\_hit\_cells and cell\_int\_to\_str(str(xc) + str(yc - 2)) not in AI\_missed\_cells:

            AI\_targeted.append(cell\_int\_to\_str(str(xc) + str(yc - 2)))

        if cell\_int\_to\_str(str(xc) + str(yc + 1)) in AI\_hit\_cells and cell\_int\_to\_str(str(xc) + str(yc + 2)) not in AI\_hit\_cells and cell\_int\_to\_str(str(xc) + str(yc + 2)) not in AI\_missed\_cells:

            AI\_targeted.append(cell\_int\_to\_str(str(xc) + str(yc + 2)))

def AI\_move():

    Display.blit(turn\_image, (23\*x, y))

    pygame.display.update()

    pygame.time.wait(1000)

    global AI\_targeted

    global user\_cells\_ref

    global AI\_hit\_cells

    global AI\_missed\_cells

    availible = []

    for i in user\_cells\_ref:

        if i not in AI\_hit\_cells and i not in AI\_missed\_cells:

            availible.append(i)

    if AI\_targeted == []:

        a = random.randrange(0, len(availible) - 1)

        if check\_cell(availible[a]):

            AI\_hit\_cells.append(availible[a])

            for i in around\_cell(availible[a]):

                if i in availible:

                    AI\_targeted.append(i)

            b = availible[a]

            Display.blit(hit\_cell, (cell\_to\_x\_and\_y(b)[0], cell\_to\_x\_and\_y(b)[1]))

            pygame.draw.rect(Display, back\_blue, [23\*x, y, x, y])

            return 1

        else:

            AI\_missed\_cells.append(availible[a])

            b = availible[a]

            pygame.draw.rect(Display, back\_blue, [23\*x, y, x, y])

            Display.blit(miss\_cell, (cell\_to\_x\_and\_y(b)[0], cell\_to\_x\_and\_y(b)[1]))

            return 0

    if AI\_targeted != []:

        t = AI\_targeted[0]

        if check\_cell(t):

            AI\_hit\_cells.append(t)

            redef\_targets()

            Display.blit(hit\_cell, (cell\_to\_x\_and\_y(t)[0], cell\_to\_x\_and\_y(t)[1]))

            pygame.draw.rect(Display, back\_blue, [23\*x, y, x, y])

            return 1

        else:

            AI\_missed\_cells.append(t)

            AI\_targeted.remove(t)

            Display.blit(miss\_cell, (cell\_to\_x\_and\_y(t)[0], cell\_to\_x\_and\_y(t)[1]))

            pygame.draw.rect(Display, back\_blue, [23\*x, y, x, y])

            return 0

def user\_move():

    global user\_missed\_cells

    global user\_hit\_cells

    global AI\_cells\_ref

    Display.blit(turn\_image, (11\*x, y))

    while True:

        for event in pygame.event.get():

            if event.type == 5:

                if top\_corner(event.pos) in back\_to.cell\_cover():

                    back\_to.onclick()

                if event.button == 1:

                    if x\_and\_y\_to\_cell(top\_corner(event.pos)) in AI\_cells\_ref and x\_and\_y\_to\_cell(top\_corner(event.pos)) not in user\_hit\_cells and x\_and\_y\_to\_cell(top\_corner(event.pos)) not in user\_missed\_cells:

                        if check\_cell(x\_and\_y\_to\_cell(event.pos), True):

                            Display.blit(hit\_cell, (top\_corner(event.pos)[0], top\_corner(event.pos)[1]))

                            user\_hit\_cells.append(x\_and\_y\_to\_cell(event.pos))

                            pygame.draw.rect(Display, back\_blue, [11\*x, y, x, y])

                            pygame.display.update()

                            return 0

                        else:

                            Display.blit(miss\_cell, (top\_corner(event.pos)[0], top\_corner(event.pos)[1]))

                            user\_missed\_cells.append(x\_and\_y\_to\_cell(event.pos))

                            pygame.draw.rect(Display, back\_blue, [11\*x, y, x, y])

                            pygame.display.update()

                            return 1

            if event.type == 12:

                qsequence()

            pygame.display.update()

def GameLoop():

    global aircraft\_carrier\_button

    global cruiser\_button

    global submarine\_button

    global destroyer\_button

    global AI\_hit\_cells

    global user\_hit\_cells

    global AI\_used\_cells

    global user\_used\_cells

    Display.fill(back\_blue)

    b1 = board(2\*x, 2\*y, 10, 10, x, y, Display, sea\_blue, black)

    b2 = board(14\*x, 2\*y, 10, 10, x, y, Display, sea\_blue, black)

    Display.blit(guest\_image, (5\*x, y))

    Display.blit(computer\_image, (17\*x, y))

    for i in buttons:

        if i in [destroyer\_button, aircraft\_carrier\_button, submarine\_button, cruiser\_button, back\_to]:

            i.set\_true\_state()

        else:

            i.set\_false\_state

    print aircraft\_carrier\_button.state

    GameExit = False

    GameOver = False

    AI\_Ship\_Set()

    user\_ship\_set()

    turn = random.randrange(0, 2, 1)

    while not GameExit:

        while GameOver:

            for event in pygame.event.get():

                if event.type == 5:

                    if x\_and\_y\_to\_cell(event.pos) in ["1232", "1332", "1432"]:

                        Display.blit(replay\_button\_onclick, (15\*x, 13\*y))

                    if x\_and\_y\_to\_cell(event.pos) in ["1732", "1832", "1932"]:

                        Display.blit(quit\_button\_onclick, (20\*x, 13\*y))

                    if top\_corner(event.pos) in back\_to.cell\_cover():

                        back\_to.onclick()

                if event.type == 6:

                    if x\_and\_y\_to\_cell(event.pos) in ["1232", "1332", "1432"]:

                        replay.onclick()

                    if x\_and\_y\_to\_cell(event.pos) in ["1732", "1832", "1932"]:

                        qbutton.onclick()

                if event.type == 12:

                    qsequence()

                pygame.display.update()

        AI\_hit\_cells.sort()

        user\_used\_cells.sort()

        user\_hit\_cells.sort()

        AI\_used\_cells.sort()

        if not turn:

            AI\_move()

            pygame.display.update()

            turn = 1

        if AI\_hit\_cells == user\_used\_cells:

            GameOver = True

            replay.set\_true\_state()

            qbutton.set\_true\_state()

            LW = False

            Display.blit(lost\_badge, (15\*x, 15\*y))

            pygame.display.update()

            continue

        if turn:

            user\_move()

            turn = 0

        AI\_hit\_cells.sort()

        user\_used\_cells.sort()

        user\_hit\_cells.sort()

        AI\_used\_cells.sort()

        if user\_hit\_cells == AI\_used\_cells:

            GameOver = True

            replay.set\_true\_state()

            qbutton.set\_true\_state()

            LW = True

            Display.blit(win\_badge, (15\*x, 15\*y))

            pygame.display.update()

            continue

        pygame.display.update()

def Home\_Screen():

    for i in buttons:

        if i not in [play1, how\_to, qbutton\_2]:

            i.set\_false\_state()

    Display.blit(homepage, (0, 0))

    play1.set\_true\_state()

    how\_to.set\_true\_state()

    qbutton\_2.set\_true\_state()

    while True:

        for event in pygame.event.get():

            if event.type == 5:

                if top\_corner(event.pos) in play1.cell\_cover():

                    play1.onclick()

                if top\_corner(event.pos) in how\_to.cell\_cover():

                    how\_to.onclick()

                if top\_corner(event.pos) in qbutton\_2.cell\_cover():

                    qbutton\_2.onclick()

            if event.type == 12:

                qsequence()

            pygame.display.update()

def How\_to\_play():

    for i in buttons:

        if i not in [play2, back\_to]:

            i.set\_false\_state()

    Display.blit(homepage, (0, 0))

    play2.set\_true\_state()

    back\_to.set\_true\_state()

    message\_to\_screen(100, 120, "1. The two boards displayed on the screen are yours and the computer (left to right respectively.)", white)

    message\_to\_screen(100, 170, "2. Select the ship tokens one by one and align them using arrow keys.", white)

    message\_to\_screen(100, 220, "3. Click the box on your board to place the ship token.", white)

    message\_to\_screen(100, 270, "4. Goal: To destroy all the ships of the AI by correctly guessing their location on its board.", white)

    message\_to\_screen(100, 370, "HAPPY BRAINSTORMING!", white)

    while True:

        for event in pygame.event.get():

            if event.type == 5:

                if top\_corner(event.pos) in play2.cell\_cover():

                    play2.onclick()

                if top\_corner(event.pos) in back\_to.cell\_cover():

                    back\_to.onclick()

            if event.type == 12:

                qsequence()

            pygame.display.update()

aircraft\_carrier\_button = button(3\*x, 13\*y, x, y, aircraft\_carrier\_button\_click, aircraft\_carrier\_button\_image, False)

cruiser\_button = button(5\*x, 13\*y, x, y, cruiser\_button\_click, cruiser\_button\_image, False)

submarine\_button = button(8\*x, 13\*y, x, y, submarine\_button\_click, submarine\_button\_image, False)

destroyer\_button = button(10\*x, 13\*y, x, y, destroyer\_button\_click, destroyer\_button\_image, False)

replay = button(15\*x, 13\*y, 3\*x, y, GameLoop, replay\_button\_image, False)

qbutton = button(20\*x, 13\*y, 3\*x, y, qsequence, quit\_button\_image, False)

play1 = button(12\*x, 3\*y, 4\*x, 4\*y, GameLoop, play\_button\_image, False)

how\_to = button(12\*x, 7\*y, 4\*x, 2\*y, How\_to\_play, how\_to\_button\_image, False)

qbutton\_2 = button(12\*x, 9\*y, 4\*x, 2\*y, qsequence, quit\_2\_button\_image, False)

play2 = button(22\*x, 13\*y, 4\*x, 4\*y, GameLoop, play\_button\_image, False)

back\_to = button(0, 15\*y, 4\*x, 2\*y, Home\_Screen, back\_to\_menu\_button\_image, False)

buttons = [aircraft\_carrier\_button, cruiser\_button, submarine\_button, destroyer\_button, how\_to, back\_to, play1, play2, replay, qbutton, qbutton\_2]

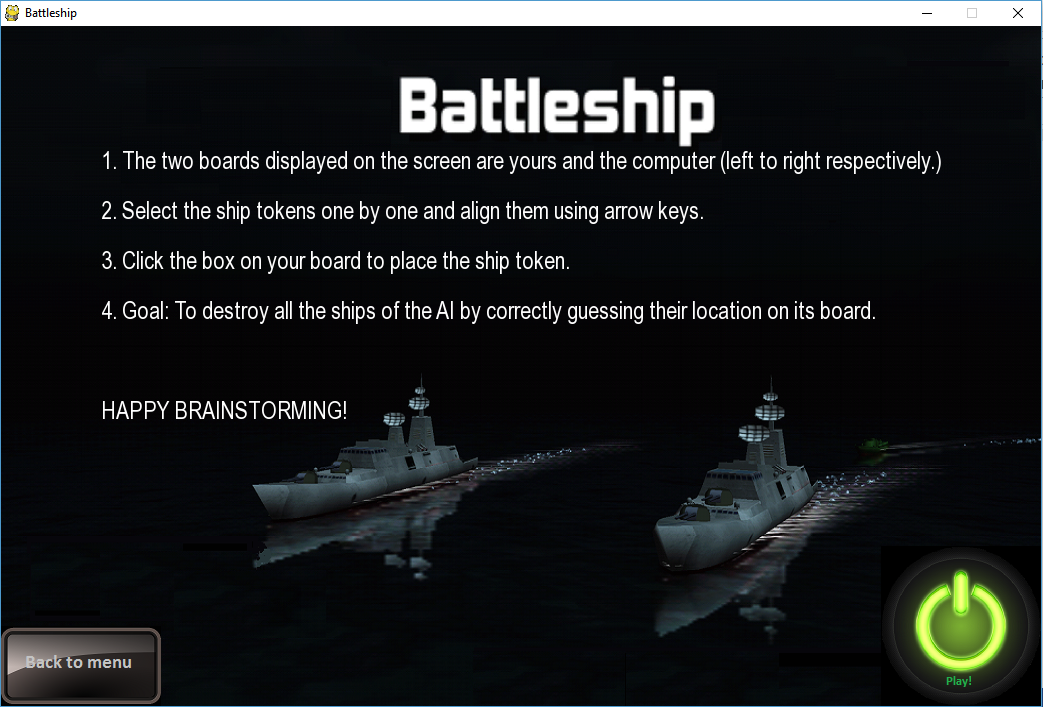
Home\_Screen()

**Screenshots of the project**

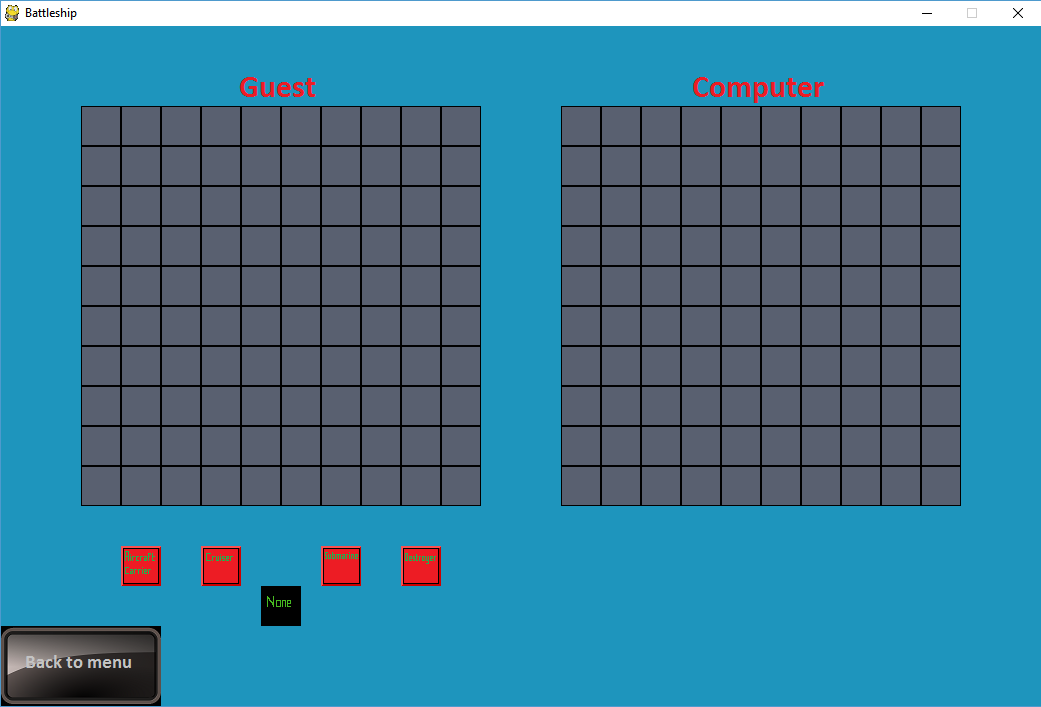
# Home Page:

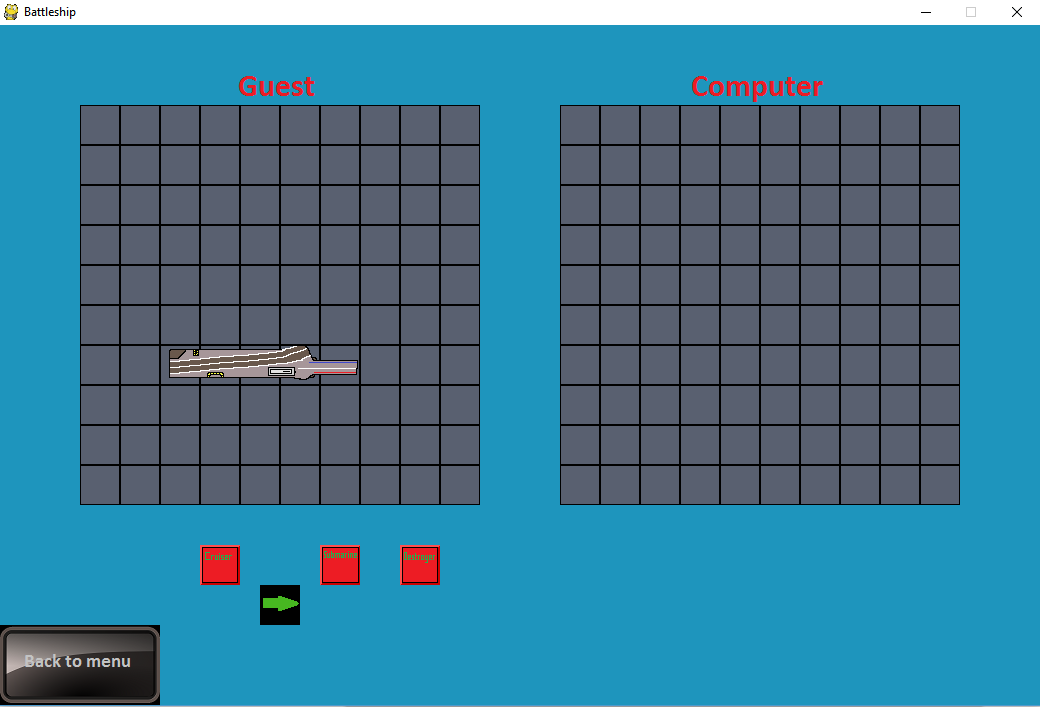


How to play Page:

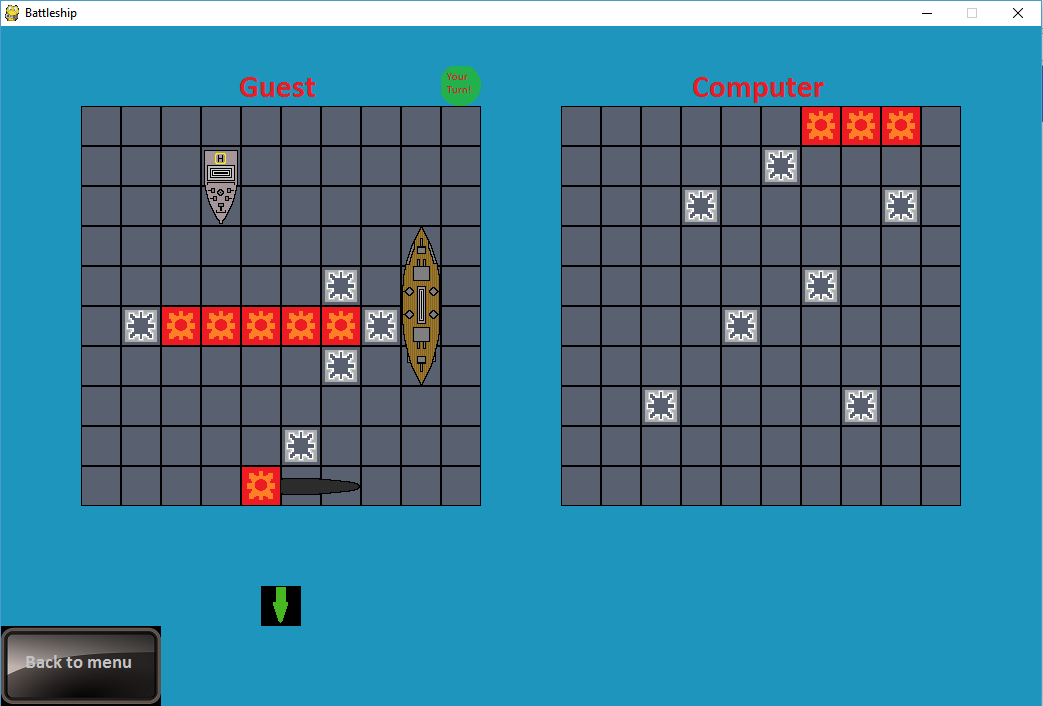


Starting the Game:

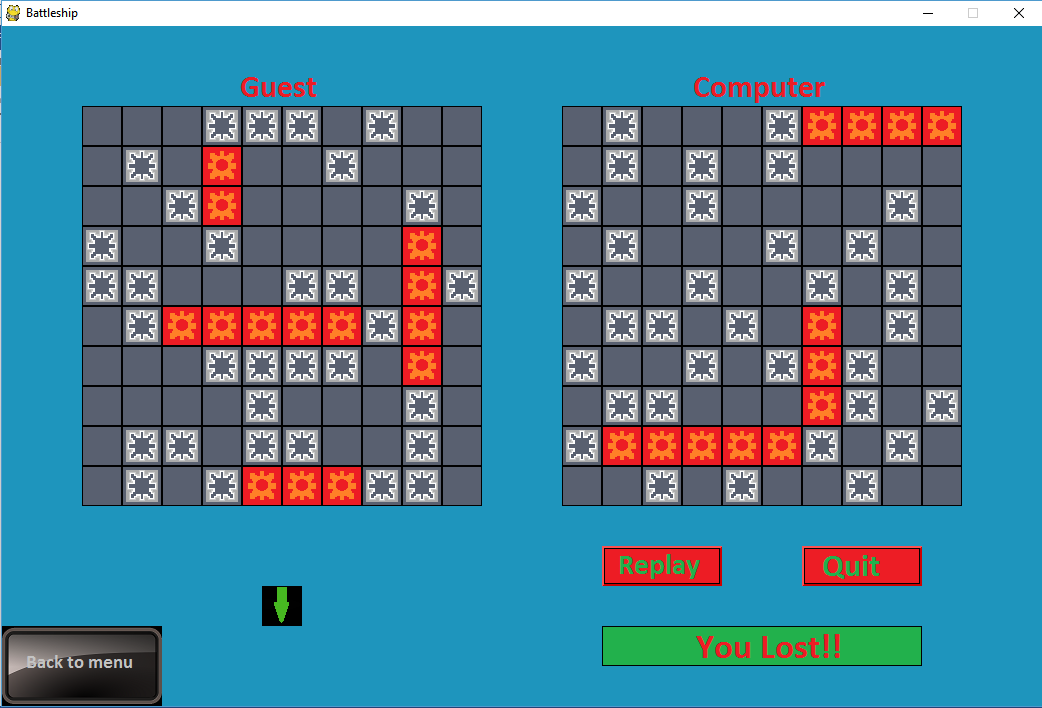


Setting your Ships:

During the Game:



End of Game:



**Classes created and Modules used**

**Modules:**1. Pygame Module: Used for designing user interface

2. Time Module: Used for pauses in gameplay, to give a more realistic experience to the player

3. Random Module: To create the algorithm for setting up of ships.

4. Math Module: Used for various mathematical operations over the course of the code

# Classes defined:

1. Button class

2. Ship class

2.1 Aircraft Carrier

2.2 Cruiser

2.3 Submarine

2.4 Destroyer

3. Board class

**FUTURE ENHANCEMENT**

There is a lot of scope for expanding this project. There are many tweaks and components we can add that can make it better equipped.

Some of them are:-

* Adding Multiplayer
* Enabling internet connectivity so that players sitting on two different systems can play together
* Better graphics for a more realistic experience
* Adding a few more games and making a collection, so that players can play according to their interests

