Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №4
“Морской бой”

Выполнил:
студент группы ИУ5-35Б:
Купцов С.Р.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.В.
Подпись и дата:

Москва, 2024 г.

Задание

Реализовать Морской Бой на языке python

Код программы

Constants.py

```
SIZE_BUTTON = [40, 40]
max_ships = [4, 3, 2, 1]
COUNT = sum(max_ships)
players = -1
```

Main.py

```
# Купцов Святослав 2020-2024

import time

from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import sys
from functools import partial
from constants import *
from random import randint
from threading import Thread

board_1 = [[0 for _ in range(COUNT)] for _ in range(COUNT)]
board_2 = [[0 for _ in range(COUNT)] for _ in range(COUNT)]
ships1 = [0, 0, 0, 0]
ships2 = [0, 0, 0, 0]


def len_ship(i, j, board, l=1):
    len_row = 1
    row = 1
    if 0 <= j + 1 < COUNT and board[i][j + 1] == l:
        while 0 <= j + row < COUNT and board[i][j + row] == l:
            row += 1
        len_row += row - 1
    row = 1
    if 0 <= j - 1 < COUNT and board[i][j - 1] == l:
        while 0 <= j - row < COUNT and board[i][j - row] == l:
            row += 1
        len_row += row - 1
    row = 1
    if 0 <= i + 1 < COUNT and board[i + 1][j] == l:
        while 0 <= i + row < COUNT and board[i + row][j] == l:
            row += 1
        len_row += row - 1
    row = 1
    if 0 <= i - 1 < COUNT and board[i - 1][j] == l:
        while 0 <= i - row < COUNT and board[i - row][j] == l:
            row += 1
        len_row += row - 1
    return len_row


def inactive_ship(i, j, board):
    row = 1
```

```
    if j + 1 < COUNT and 0 <= j - 1 and i + 1 < COUNT and 0 <= i - 1 and
board[i][j+1] in [0, -1] and board[i][j-1] in [0, -1] and board[i+1][j] in
[0, -1] and board[i-1][j] == 0 in [0, -1]:
        return True
    if 0 <= j + 1 < COUNT and board[i][j + 1] in [-2, 1]:
        while 0 <= j + row < COUNT and board[i][j + row] in [-2, 1]:
            if 0 <= j + row < COUNT and board[i][j + row] == 1:
                return False
            row += 1
    row = 1
    if 0 <= j - 1 < COUNT and board[i][j - 1] in [-2, 1]:
        while 0 <= j - row < COUNT and board[i][j - row] in [-2, 1]:
            if 0 <= j - row < COUNT and board[i][j - row] == 1:
                return False
            row += 1
    row = 1
    if 0 <= i + 1 < COUNT and board[i + 1][j] in [-2, 1]:
        while 0 <= i + row < COUNT and board[i + row][j] in [-2, 1]:
            if 0 <= i + row < COUNT and board[i + row][j] == 1:
                return False
            row += 1
    row = 1
    if 0 <= i - 1 < COUNT and board[i - 1][j] in [-2, 1]:
        while 0 <= i - row < COUNT and board[i - row][j] in [-2, 1]:
            if 0 <= i - row < COUNT and board[i - row][j] == 1:
                return False
            row += 1
    return True


class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(QDesktopWidget().screenGeometry().width() // 2 -
175, QDesktopWidget().screenGeometry().height() // 2 - 175, 350, 350)
        self.setWindowIcon(QIcon('data\\icon.png'))
        self.setWindowTitle('Морской бой')
        self.setFixedSize(350, 350)
        self.but1 = QPushButton('Одиночная игра', self)
        self.but1.setFont(QFont('Times', 15))
        self.but1.resize(250, 50)
        self.but1.move(50, 50)
        self.but1.clicked.connect(self.one_player)
        self.but2 = QPushButton('1 на 1', self)
        self.but2.setFont(QFont('Times', 15))
        self.but2.resize(250, 50)
        self.but2.move(50, 150)
        self.but2.clicked.connect(self.two_player)
        self.show()

    def one_player(self):
        global players
        w.text.setText('Расположите ваши корабли')
        w.show()
        main.text_pole1.setText('Ваше поле:')
        main.text_pole2.setText('Поле вашего противника:')
        main.text.setText('')
        self.hide()
        players = 1
        w.auto_stay(board_2, ships2)

    def two_player(self):
        global players
        w.show()
```

```
        self.hide()
        players = 2


class Victory(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(QDesktopWidget().screenGeometry().width() // 2 -
200, QDesktopWidget().screenGeometry().height() // 2 - 50, 400, 100)
        self.setWindowIcon(QIcon('data\\icon.png'))
        self.setWindowTitle('Морской бой (Результат)')
        self.setFixedSize(400, 100)
        self.text = QLabel('', self)
        self.text.resize(400, 100)
        self.text.setFont(QFont('Times', 17))

    def set(self, text):
        self.text.setText(text)
        self.show()


class GameWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(QDesktopWidget().screenGeometry().width() // 2 -
425, QDesktopWidget().screenGeometry().height() // 2 - 250, 850, 500)
        self.setWindowIcon(QIcon('data\\icon.png'))
        self.setWindowTitle('Морской бой (Битва)')
        self.setFixedSize(850, 500)
        self.text = QLabel('Ходит 1 игрок', self)
        self.text.setFont(QFont("Times", 17))
        self.text.move(0, 0)
        self.text.show()
        self.text_pole1 = QLabel('Поле 1 игрока:', self)
        self.text_pole1.setFont(QFont("Times", 15))
        self.text_pole1.move(0, 70)
        self.text_pole2 = QLabel('Поле 2 игрока:', self)
        self.text_pole2.setFont(QFont("Times", 15))
        self.text_pole2.move(SIZE_BUTTON[0] * COUNT + 51, 70)
        self.flag = 0

        self.point = QIcon('data\\point.png')
        self.killed = QIcon('data\\killed.png')

        self.board1 = [[QPushButton(self) for _ in range(COUNT)] for _ in
range(COUNT)]
        self.board2 = [[QPushButton(self) for _ in range(COUNT)] for _ in
range(COUNT)]

        self.weights = [[1 for _ in range(COUNT)] for _ in range(COUNT)]
        for i in range(COUNT):
            for j in range(COUNT):
                self.board1[i][j].setStyleSheet('QPushButton {background-
color: white;}')
                self.board1[i][j].resize(*SIZE_BUTTON)
                self.board1[i][j].clicked.connect(partial(self.click, j, i,
board_1, self.board1, ships1, 1))
                self.board1[i][j].move(SIZE_BUTTON[0] * i, SIZE_BUTTON[1] * j
+ 100)
                self.board2[i][j].setStyleSheet('QPushButton {background-
color: white;}')
                self.board2[i][j].clicked.connect(partial(self.click, j, i,
board_2, self.board2, ships2, 2))
                self.board2[i][j].resize(*SIZE_BUTTON)
```

```
                self.board2[i][j].move(SIZE_BUTTON[0] * i + SIZE_BUTTON[0] *
COUNT + 51, SIZE_BUTTON[1] * j + 100)

    def click(self, i, j, board, b, ship, num):
        if players == 2:
            if num == 1 and self.flag % 2 == 0:
                return
            elif num == 2 and self.flag % 2 == 1:
                return
            if board[i][j] in [-2, -1]:
                return
            if board[i][j] == 1:
                board[i][j] = -2
                b[j][i].setIcon(self.killed)
                self.flag += 1
                if inactive_ship(i, j, board):
                    self.dectroy(board, i, j, b)
                    ship[len_ship(i, j, board, -2) - 1] -= 1
            if board[i][j] == 0:
                board[i][j] = -1
                b[j][i].setIcon(self.point)
                self.text.setText(f'Ходит {num % 2} игрок')
            self.flag += 1
            if ship == [0, 0, 0, 0]:
                self.hide()
                v.set(f'Игрок {num % 2} победил!\nОтличная игра!)')
        elif players == 1:
            if board == board_1:
                return
            if board[i][j] in [-2, -1] or self.flag:
                return
            if board[i][j] == 1:
                board[i][j] = -2
                b[j][i].setIcon(self.killed)
                self.flag = 0
                if inactive_ship(i, j, board):
                    self.dectroy(board, i, j, b)
                    ship[len_ship(i, j, board, -2) - 1] -= 1
            if board[i][j] == 0:
                board[i][j] = -1
                b[j][i].setIcon(self.point)

                # Ходы ИИ
                th = Thread(target=self.randomcheck)
                th.start()
            if ship == [0, 0, 0, 0]:
                self.hide()
                v.set(f'Вы победили!\nОтличная игра!)')
            self.text.setText(f'Ходит {self.flag + 1} игрок')

    def randomcheck(self):
        self.flag = 1
        self.text.setText(f'Ходит {self.flag + 1} игрок')
        for i in range(COUNT):
            for j in range(COUNT):
                if self.weights[i][j] == 9:
                    if self.attack(i, j):
                        self.attack(i, j)
                        if ships1 == [0, 0, 0, 0]:
                            self.hide()
                            v.set('Зная тупость алгоритма, если он вас\n
обыграл, то Восстание машин \nнеминеумо...')
                        self.flag = 0
                        return 0
```

```
                    else:
                        self.weights[i][j] = 0
        while True:
            if self.attack(randint(0, COUNT - 1), randint(0, COUNT - 1)):
                for i in self.weights:
                    print(*i)
                print('-' * 40)
                break
            for i in self.weights:
                print(*i)
            print('-' * 40)
        if ships1 == [0, 0, 0, 0]:
            self.hide()
            v.set('Зная тупость алгоритма, если он вас\nобыграл, то Восстание
машин \nнеминеумо...')
        self.flag = 0
        self.text.setText(f'Ходит {self.flag + 1} игрок')

    def attack(self, i, j):
        if board_1[i][j] in [-1, -2]:
            return 0
        if board_1[i][j] == 1:
            self.board1[j][i].setIcon(QIcon('data\\killed.png'))
            for _i in [1, -1]:
                for _j in [1, -1]:
                    if 0 <= i + _i < COUNT and 0 <= j + _j < COUNT and
board_1[i + _i][j + _j] != 1:
                        self.weights[i + _i][j + _j] = 0
            for _i, _j in [[0, 1], [0, -1], [1, 0], [-1, 0]]:
                if 0 <= i + _i < COUNT and 0 <= j + _j < COUNT:
                    if self.weights[i + _i][j + _j] != 0:
                        self.weights[i + _i][j + _j] = 9
            board_1[i][j] = -2
        else:
            self.board1[j][i].setIcon(QIcon('data\\point.png'))
            board_1[i][j] = -1
        self.weights[i][j] = 0
        if board_1[i][j] == -2 and inactive_ship(i, j, board_1):
            self.dectroy(board_1, i, j, self.board1)
            ships1[len_ship(i, j, board_1, -2) - 1] -= 1
            for _i, _j in [[0, 1], [0, -1], [1, 0], [-1, 0]]:
                if 0 <= i + _i < COUNT and 0 <= j + _j < COUNT and board_1[i
+ _i][j + _j] != 1:
                    self.weights[i + _i][j + _j] = 0
        if board_1[i][j] == -2:
            time.sleep(1)
            self.randomcheck()
        return 1

    def dectroy(self, board, i, j, b):
        for k in [-1, 0, 1]:
            for l in [-1, 0, 1]:
                if 0 <= i + k < COUNT and 0 <= j + l < COUNT and board[i +
k][j + l] == 0:
                    b[j + l][i + k].setIcon(self.point)
                    board[i + k][j + l] = -1
                    #self.weights[i + k][j + l] = 0
        row = 1
        if 0 <= j + 1 < COUNT and board[i][j + 1] == -2:
            while 0 <= j + row < COUNT and board[i][j + row] == -2:
                for k in [-1, 0, 1]:
                    for l in [-1, 0, 1]:
                        if 0 <= i + k < COUNT and 0 <= j + row + l < COUNT
and board[i + k][j + row + l] == 0:
```

```
                            b[j + row + l][i + k].setIcon(self.point)
                            board[i + k][j + row + l] = -1
                            #self.weights[i + k][j + row + l] = 0
                    row += 1
            row = 1
            if 0 <= j - 1 < COUNT and board[i][j - 1] == -2:
                while 0 <= j - row < COUNT and board[i][j - row] == -2:
                    for k in [-1, 0, 1]:
                        for l in [-1, 0, 1]:
                            if 0 <= i + k < COUNT and 0 <= j - row + l < COUNT
and board[i + k][j - row + l] == 0:
                                b[j - row + l][i + k].setIcon(self.point)
                                board[i + k][j - row + l] = -1
                                #self.weights[i + k][j - row + l] = 0
                    row += 1
            row = 1
            if 0 <= i + 1 < COUNT and board[i + 1][j] == -2:
                while 0 <= i + row < COUNT and board[i + row][j] == -2:
                    for k in [-1, 0, 1]:
                        for l in [-1, 0, 1]:
                            if 0 <= i + row + k < COUNT and 0 <= j + l < COUNT
and board[i + row + k][j + l] == 0:
                                b[j + l][i + row + k].setIcon(self.point)
                                board[i + row + k][j + l] = -1
                                #self.weights[i + row + k][j + l] = 0
                    row += 1
            row = 1
            if 0 <= i - 1 < COUNT and board[i - 1][j] == -2:
                while 0 <= i - row < COUNT and board[i - row][j] == -2:
                    for k in [-1, 0, 1]:
                        for l in [-1, 0, 1]:
                            if 0 <= i - row + k < COUNT and 0 <= j + l < COUNT
and board[i - row + k][j + l] == 0:
                                b[j + l][i - row + k].setIcon(self.point)
                                board[i - row + k][j + l] = -1
                                #self.weights[i - row + k][j + l] = 0
                    row += 1


class StayShips(QWidget):
    def __init__(self):
        super().__init__()
        self.setGeometry(QDesktopWidget().screenGeometry().width() // 2 -
300, QDesktopWidget().screenGeometry().height() // 2 - 250, 600, 500)
        self.setWindowIcon(QIcon('data\\icon.png'))
        self.setWindowTitle('Морской бой (расстановка кораблей)')
        self.setFixedSize(600, 500)
        self.hide()
        self.text = QLabel('Расположение кораблей 1 игрока', self)
        self.count_ship = QLabel('■  ■  ■  ■\n■  ■  ■\n■  ■\n■', self)
        self.counter1 = QLabel(' 0     0     0    0', self)
        self.counter2 = QLabel(' 0     0     0    0', self)
        self.error = QPlainTextEdit('Пока всё норм\nНо я хз', self)
        self.start = QPushButton('Принять', self)
        self.start.resize((SIZE_BUTTON[0] * COUNT + 2), SIZE_BUTTON[1])
        self.error.resize(595 - SIZE_BUTTON[0] * COUNT, 50)
        self.error.move(SIZE_BUTTON[0] * COUNT + 5, 490 - SIZE_BUTTON[1])
        self.count_ship.move(420, 80)
        self.counter1.move(420, 50)
        self.counter2.move(-3000, 0)
        self.start.move(0, SIZE_BUTTON[1] * COUNT + 57)
        self.count_ship.setFont(QFont('Times', 17))
        self.counter1.setFont(QFont('Times', 17))
        self.counter2.setFont(QFont('Times', 17))
```

```
        self.text.setFont(QFont("Times", 17))
        self.start.clicked.connect(self.check_start)
        self.error.setReadOnly(True)
        self.board1 = [[QPushButton(self) for _ in range(COUNT)] for _ in
range(COUNT)]
        self.board2 = [[QPushButton(self) for _ in range(COUNT)] for _ in
range(COUNT)]

        self.flag = 0
        for i in range(COUNT):
            for j in range(COUNT):
                self.board1[i][j].setStyleSheet('QPushButton {background-
color: white;}')
                self.board1[i][j].clicked.connect(partial(self.click, j, i,
board_1, self.board1, self.counter1, ships1))
                self.board1[i][j].resize(*SIZE_BUTTON)
                self.board1[i][j].setIconSize(QSize(40, 40))
                self.board1[i][j].move(SIZE_BUTTON[0] * i, SIZE_BUTTON[1] * j
+ 50)
                self.board2[i][j].setStyleSheet('QPushButton {background-
color: white;}')
                self.board2[i][j].clicked.connect(partial(self.click, j, i,
board_2, self.board2, self.counter2, ships2))
                self.board2[i][j].resize(*SIZE_BUTTON)
                self.board2[i][j].move(0, -3000)
                self.board2[i][j].setIconSize(QSize(40, 40))

    def auto_stay(self, board, ship):
        try:
            from random import randint, choice
            for l_ship in range(1, len(max_ships) + 1):
                for _ in range(max_ships[l_ship - 1]):
                    while True:
                        i, j = randint(0, COUNT - 1), randint(0, COUNT - 1)
                        d_i, d_j = choice([[0, 1], [0, -1], [1, 0], [-1, 0]])
                        count = 0
                        if 0 <= i + d_i * (l_ship - 1) < COUNT and 0 <= j +
d_j * (l_ship - 1) < COUNT:
                            for row in range(l_ship):
                                if self.check(i + d_i * row, j + d_j * row,
board):
                                    count += 1
                            if count == l_ship:
                                for row in range(l_ship):
                                    board[i + d_i * row][j + d_j * row] = 1
                                ship[l_ship - 1] += 1
                                break
        except Exception as es:
            print(es)
        for k in board:
            print(*k)
        print('ships =', ship)

    def check(self, i, j, board):
        for _i in [1, 0, -1]:
            for _j in [1, 0, -1]:
                if 0 <= i + _i < COUNT and 0 <= j + _j < COUNT and board[_i +
i][_j + j] == 1:
                    return 0
        return 1

    def checkShip(self, i, j, board1, b, ships):
        try:
            if 0 <= i+1 < COUNT and 0 <= j+1 < COUNT and board1[i+1][j+1] ==
```

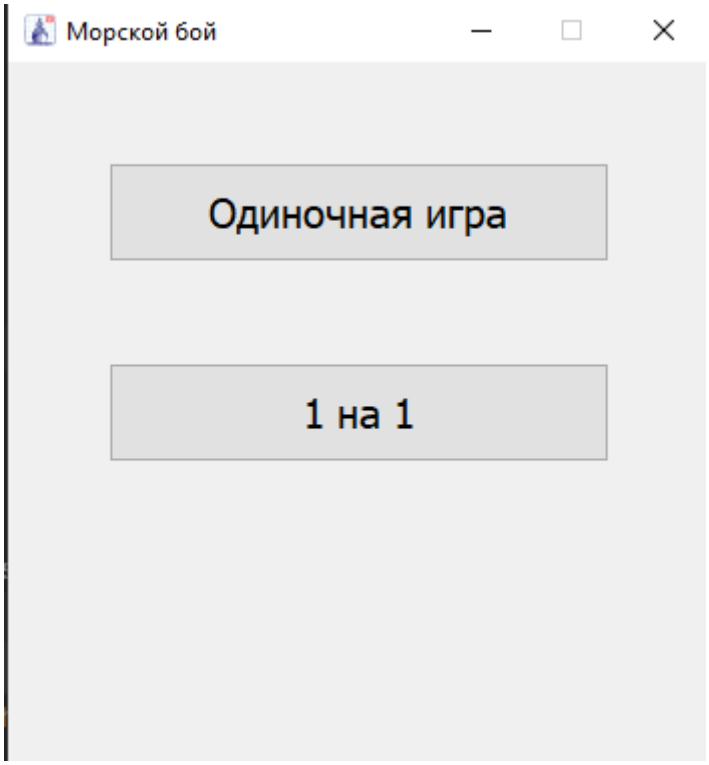
```
1 and board1[i][j] == 1:
                b[j][i].setIcon(QIcon('data\\killed.png'))
                board1[i][j] = -1
            elif 0 <= i-1 < COUNT and 0 <= j+1 < COUNT and board1[i-1][j+1]
== 1 and board1[i][j] == 1:
                b[j][i].setIcon(QIcon('data\\killed.png'))
                board1[i][j] = -1
            elif 0 <= i-1 < COUNT and 0 <= j-1 < COUNT and board1[i-1][j-1]
== 1 and board1[i][j] == 1:
                b[j][i].setIcon(QIcon('data\\killed.png'))
                board1[i][j] = -1
            elif 0 <= i+1 < COUNT and 0 <= j-1 < COUNT and board1[i+1][j-1]
== 1 and board1[i][j] == 1:
                b[j][i].setIcon(QIcon('data\\killed.png'))
                board1[i][j] = -1
            else:
                len_row = len_ship(i, j, board1)
                if len_row > 4 or ships[len_row - 1] + 1 > max_ships[len_row
- 1]:
                    b[j][i].setIcon(QIcon('data\\killed.png'))
                    board1[i][j] = -1
                else:
                    b[j][i].setIcon(QIcon('data\\live1.png'))
                    k = self.func(i, j, board1, b)
                    if k == 2:
                        if 0 <= j - 1 and j + 1 < COUNT and board1[i][j + 1]
== 1 or board1[i][j - 1]:
                            b[j][i].setIcon(QIcon('data\\live3.png'))
                        else:

b[j][i].setIcon(QIcon(QPixmap(QImage('data\\live3.png')).transformed(QTransfo
rm().rotate(90))))
                        if len_row == 4:
                            ships[3] += 1
                            ships[0] -= 1
                            ships[1] -= 1
                        elif len_row == 3:
                            ships[2] += 1
                            ships[0] -= 2
                    else:
                        len_row -= 1
                        if len_row > 0:
                            ships[len_row - 1] -= 1
                        if len_row < 4:
                            ships[len_row] += 1
        except IndexError:
            print('lol')

    def func(self, i, j, board1, b):
        k = 0
        if 0 <= j + 1 < COUNT and board1[i][j + 1] == 1:
            k += 1
            if 0 <= j + 2 < COUNT and board1[i][j + 2] == 1:
                b[j + 1][i].setIcon(QIcon('data\\live3.png'))
            else:
                b[j +
1][i].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransform
().rotate(180))))

b[j][i].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransfo
rm().rotate(0))))
        if 0 <= j - 1 < COUNT and board1[i][j - 1] == 1:
            k += 1
            if 0 <= j - 2 < COUNT and board1[i][j - 2] == 1:
```

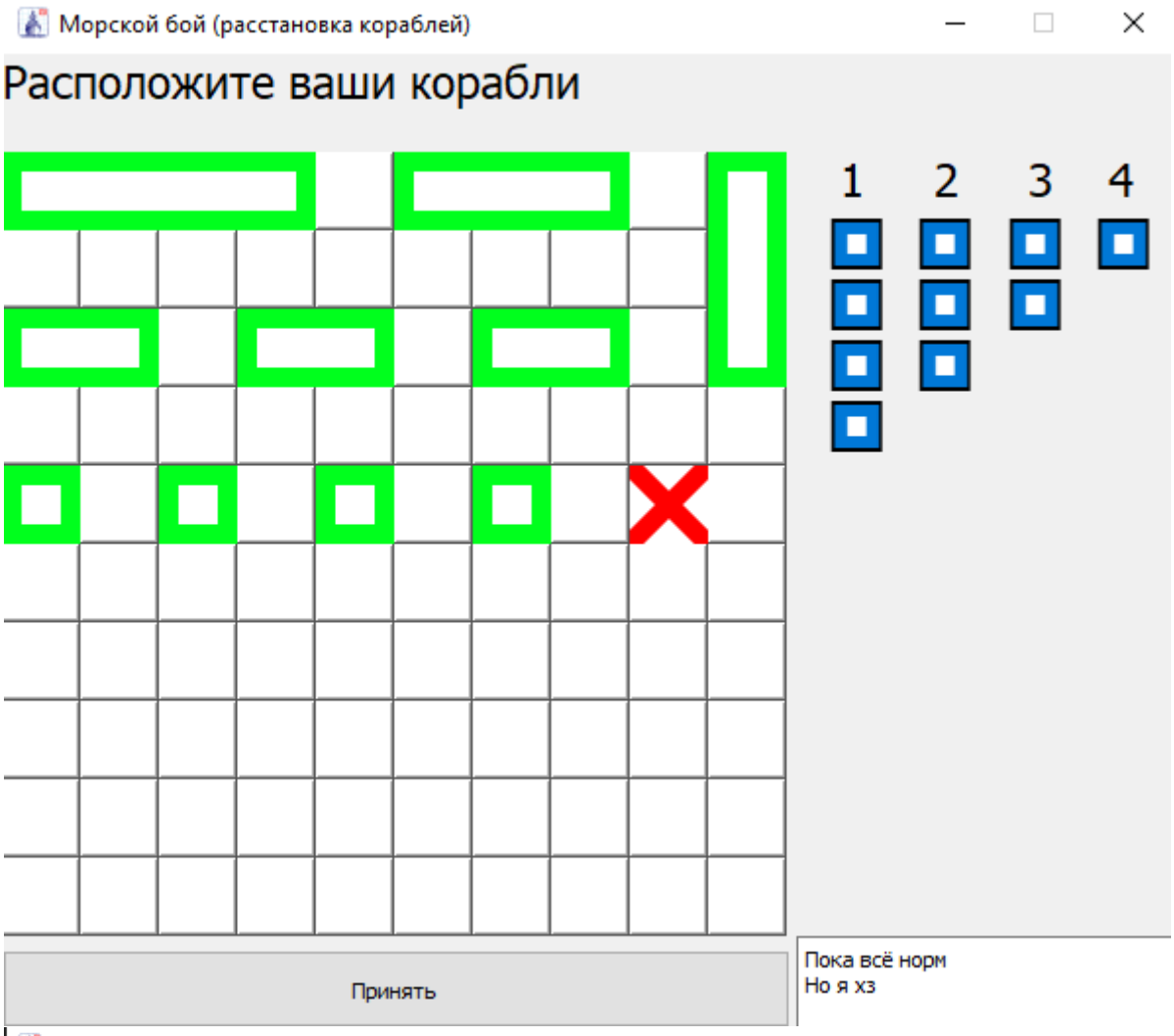
```
                b[j - 1][i].setIcon(QIcon('data\\live3.png'))
            else:
                b[j -
1][i].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransform
().rotate(0))))

b[j][i].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransfo
rm().rotate(180))))
        if 0 <= i + 1 < COUNT and board1[i + 1][j] == 1:
            k += 1
            if 0 <= i + 2 < COUNT and board1[i + 2][j] == 1:
                b[j][i +
1].setIcon(QIcon(QPixmap(QImage('data\\live3.png')).transformed(QTransform().
rotate(90))))
            else:
                b[j][i +
1].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransform().
rotate(-90))))

b[j][i].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransfo
rm().rotate(90))))
        if 0 <= i - 1 < COUNT and board1[i - 1][j] == 1:
            k += 1
            if 0 <= i - 2 < COUNT and board1[i - 2][j] == 1:
                b[j][i -
1].setIcon(QIcon(QPixmap(QImage('data\\live3.png')).transformed(QTransform().
rotate(90))))
            else:
                b[j][i -
1].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransform().
rotate(90))))

b[j][i].setIcon(QIcon(QPixmap(QImage('data\\live2.png')).transformed(QTransfo
rm().rotate(-90))))
        return k

    def check_start(self):
        if self.flag == 0:
            if ships1 == max_ships:
                count = [1 for k in board_1 if -1 not in k]
                if sum(count) == COUNT and players == 2:
                    self.error.setPlainText('всё ок!')
                    self.text.setText('Расположение кораблей 2 игрока')
                    self.counter2.move(420, 50)
                    self.counter1.move(-3000, 0)
                    for i in range(COUNT):
                        for j in range(COUNT):
                            self.board1[i][j].move(-3000, 0)
                            self.board2[i][j].move(SIZE_BUTTON[0] * i,
SIZE_BUTTON[1] * j + 50)
                    self.flag = 1
                elif sum(count) == COUNT and players == 1:
                    self.error.setPlainText('всё ок')
                    self.hide()
                    main.show()
                else:
                    self.error.setPlainText('Уберите все ваши ошибки!')
            else:
                self.error.setPlainText('Не все корабли стоят!')
        elif self.flag == 1:
            if ships2 == max_ships:
                count = [1 for k in board_2 if -1 not in k]
                if sum(count) == COUNT:
                    self.hide()
```

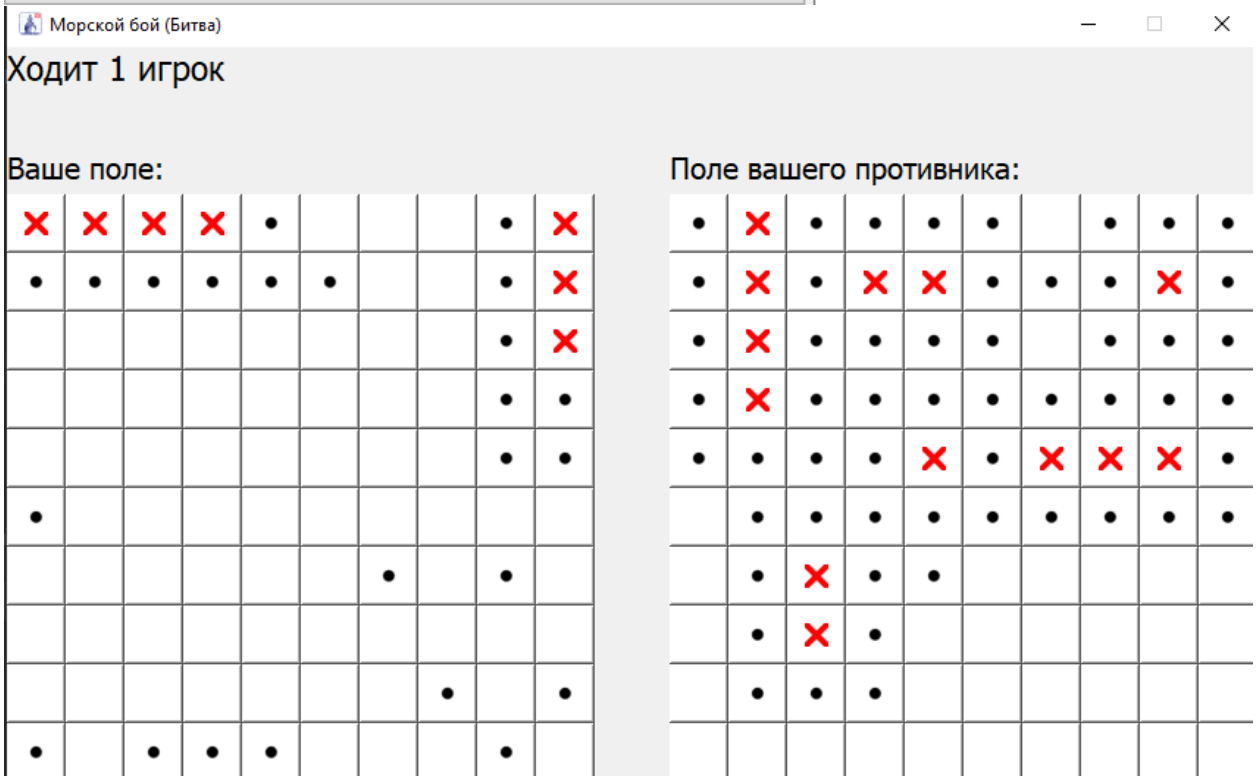
```
                    main.show()
                else:
                    self.error.setPlainText('Уберите все ваши ошибки!')
            else:
                self.error.setPlainText('Не все корабли стоят!')

    def click(self, i, j, board1, b, count, ships):
        board1[i][j] = (board1[i][j] + 1) % 2 if board1[i][j] >= 0 else -1
        if board1[i][j] == 1:
            self.checkShip(i, j, board1, b, ships)
        elif board1[i][j] == -1:
            b[j][i].setIcon(QIcon())
            board1[i][j] = 0
        else:
            b[j][i].setIcon(QIcon())
            l = len_ship(i, j, board1) - 1
            k = 0
            if 0 <= j + 1 < COUNT and board1[i][j + 1] == 1:
                k += 1
                self.func(i, j + 1, board1, b)
                if 0 <= j + 2 < COUNT and board1[i][j + 2] != 1 or not (0 <=
j + 2 < COUNT):
                    b[j + 1][i].setIcon(QIcon('data\\live1.png'))
            if 0 <= j - 1 < COUNT and board1[i][j - 1] == 1:
                k += 1
                self.func(i, j - 1, board1, b)
                if 0 <= j - 2 < COUNT and board1[i][j - 2] != 1 or not (0 <=
j - 2 < COUNT):
                    b[j - 1][i].setIcon(QIcon('data\\live1.png'))
            if 0 <= i + 1 < COUNT and board1[i + 1][j] == 1:
                k += 1
                self.func(i + 1, j, board1, b)
                if 0 <= i + 2 < COUNT and board1[i + 2][j] != 1 or not (0 <=
i + 2 < COUNT):
                    b[j][i + 1].setIcon(QIcon('data\\live1.png'))
            if 0 <= i - 1 < COUNT and board1[i - 1][j] == 1:
                k += 1
                self.func(i - 1, j, board1, b)
                if 0 <= i - 2 < COUNT and board1[i - 2][j] != 1 or not (0 <=
i - 2 < COUNT):
                    b[j][i - 1].setIcon(QIcon('data\\live1.png'))
            if k == 2:
                board1[i][j] = 1
                l = len_ship(i, j, board1)
                if l == 4:
                    ships[3] -= 1
                    ships[1] += 1
                    ships[0] += 1
                elif l == 3:
                    ships[2] -= 1
                    ships[0] += 2
                board1[i][j] = 0
            else:
                if l < 4:
                    ships[l] -= 1
                if l > 0:
                    ships[l - 1] += 1
        count.setText(f' {ships[3]}     {ships[2]}     {ships[1]}
{ships[0]}')


if __name__ == "__main__":
    app = QApplication(sys.argv)
    w = StayShips()
```

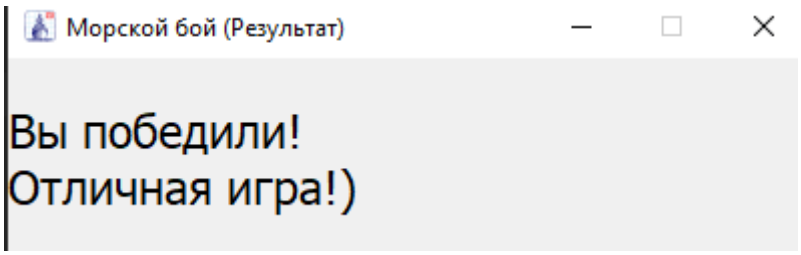
```
    window = MainWindow()
    main = GameWindow()
    v = Victory()
    sys.exit(app.exec())
```

Экранные формы с примерами выполнения программы




Морской бой (расстановка кораблей)
Расположите ваши корабли
1
2
3
4
Принять
Пока всё норм
Но я хз


Морской бой (Битва)
Ходит 1 игрок
Ваше поле:
Поле вашего противника:


Морской бой (Результат)
Вы победили!
Отличная игра!)