**Министерство науки и высшего образования Российской Федерации**

ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт цифровых технологий физики и электроники

Кафедра вычислительной техники и электроники

**Отчёт по технологической (проектно-технологической) практике**

**по теме:**

**игра «Морской бой»**

Выполнил студент 595 гр.:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Осипенко Д.В.

Проверил: к.т.н., доцент каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Скурыдин Ю.Г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_ г.

Барнаул 202\_

**Задание по проектно-технологической практике**

Создать простую графическую игру «Морской бой». Игровой процесс должен проходить по правилам вышеупомянутой игры. В игре принимает участие 1 игрок. Процесс игры должен отображаться на экране. Приложение должно иметь удобную систему меню.

***Перечень представляемых к защите материалов:***

Пояснительная записка к курсовой работе

Программный продукт

# Оглавление

[Оглавление 3](#_Toc61020769)

[Введение 4](#_Toc61020770)

[1. Постановка задачи 5](#_Toc61020771)

[2. Техническое задание 6](#_Toc61020772)

[3. Проектирование программного продукта 8](#_Toc61020773)

[4. Обоснования выбора используемого языка программирования и среды разработки 11](#_Toc61020774)

[5. Схема алгоритма 12](#_Toc61020775)

[6. Текст программного кода 17](#_Toc61020776)

[7. Отладка и тестирование работоспособности 25](#_Toc61020777)

[8. Список литературы 26](#_Toc61020778)

# Введение

Графические приложения – одни из самых массовых в пользовательском сегменте. К настоящему времени разработано огромное кол-во графических приложений, направленных на различные области деятельности человека, начиная от узкоспециализированных до развлекательных программных продуктов. Наиболее выделяющимися за последние годы и в свете недавних событий являются компьютерные игры, в которые к середине 2020 года, по некоторым исследованиям[1], играют 3.1 миллиарда человек.

В данной курсовой работе стоит задача разработать графический программный продукт: игра «Морской бой», которая основана на оригинальных правилах ее настольного аналога.

Для разработки данного приложения была выбрана среда программирования Visual Studio Code, предоставляющая широкие возможности для своей настройки, удобство и широкую поддержку пользовательских расширений. В качестве основного языка программирования был выбран Python 3.8, язык программирования 3.5 поколения, за его простоту и высокую скорость разработки. В качестве графического фреймворка был выбран PyGame, также за его простоту и удобную документацию.

# Постановка задачи

Разработать программный продукт – простую графическую игру «Морской бой». Данная игра должна иметь меню, игровое окно с двумя игровыми полями (1ое для игрока, 2ое для бота), возможность с помощью устройства ввода мышь расставлять корабли, взаимодействовать с пунктами меню и вражеским полем, с помощью устройства ввода клавиатура поворачивать перед расстановкой корабли и выхода в игровое меню. Также игра должна соответствовать некоторым правилам:

1. Игра проходит на двух полях размером10 на 10 клеток.
2. На каждом поле размещаются 8 кораблей разного типа (1 – 4х палубник, 2 – 3х палубник, 3 – 2х палубник, 2 – 1х палубник)
3. Корабли могут стоять как вертикально, так и горизонтально,но нельзя ставить впритык друг к другу
4. Каждая клетка может игрового поля может принимать четыре состояния (0 – клетка пуста, 1 – клетка содержит палубу, 2 – клетка содержит попадание по палубе, 3 – клетка содержит попадания в пустую клетку)
5. Каждый игрок поочередно указывает координаты выстрела по клетке соперника
6. Победителем становиться тот, кто потопит весь флот оппонента

# Техническое задание

1. Наименование программного продукта
   1. “Lord of the Sea”
   2. Условное наименование: МБ
2. Заказчик и исполнитель работы
   1. Заказчик: кафедра ВТиЭ АлтГУ
   2. Исполнитель: студент 595 гр. Осипенко Данил Владимирович
3. Цель разработки и функциональное назначение программного продукта.
   1. Целью разработки является создание простой игры МБ по правилам оригинальной игры «Морской бой»
   2. Программный продукт предназначен для увеселительного время препровождения, развития тактического и стратегического мышлений
4. Источники разработки
   1. Формулировка задания
   2. Постановка задания
   3. Критический анализ аналогичных разработок
5. Системные и технические требования к программному продукту
   * 1. Комплектность
        + Готовый программный продукт
     2. Условия эксплуатации
        + Программный продукт должен работать на персональном компьютере типовой комплектации
        + Программный продукт должен запускаться под любой системой, поддерживающей Python 3.8 и PyGame
        + Основные требования к МБ
     3. МБ должен обладать удобным интерфейсом и управлением
     4. МБ должен работать на ПК типовой комплектации
   1. Требования к программе
      1. МБ выполняется в виде исходного файла с помощью интерпретатора Python 3.8
      2. МБ удовлетворяет желаниям заказчика
   2. Требования к операционным системам и инструментарию
      1. Допускается применение продукта на любой операционной системе, которая поддерживается Python 3.8 и имеет предустановленные PyGame, Numpy
6. Требования к маркировке, транспортировке и распространению программного продукта
   1. МБ маркируется и распространяется по лицензии MIT(X11)
7. Прочие требования
   1. В техническое задание могут вносится изменения и дополнения по согласованию между Заказчиком и Исполнителем
8. Стадии разработки
   1. Разработка ТЗ
   2. Разработка эскиза проекта
   3. Реализация проекта
   4. Тестирование проекта
   5. Подготовка проектной документации и пояснительной записки
   6. Сдача продукта в эксплуатацию
9. Порядок сдачи и приемки программного продукта
   1. Проверка соответствия продукта ТЗ
   2. Проверка соответствия документа ГОСТам
   3. Тестирование МБ
   4. Защита пояснительной записки

# Проектирование программного продукта

В настоящий момент существует огромное множество игр по морскому бою, обладающих различным функционалом, внешним видом, графической составляющей. Многие из них представляют собой мобильные игры, например «Морской бой 2», и никак не представлены на ПК. Те же что имеются это либо игры командной строки[2], невзрачные и неинтересные, либо сделанные на различных мощных игровых движках как Unity[3], либо наипростейшие и неприглядные проекты[4].

Из вышесказанного следует, что создаваемый программный продукт должен обладать простым и удобным графическим интерфейсом, простым игровым процессом, приятной глазу палитрой цветов, иметь малый размер и нераздражающий внешний вид. Также работать на ПК.

Разрабатываемая игра должна иметь следующие функции:

1. вращение корабля перед его расположением
2. выбор позиции для корабля
3. выбор клетки, куда будет совершен выстрел
4. завершение работы

Пользовательский интерфейс МБ должен иметь окно главного меню с 3мя кнопками: играть, настройки, выход. Это окно появляется сразу после запуска программы. Первая кнопка создает сцену с самим игровым процессом, вторая кнопка вызывает настройки (на данный момент не реализовано), третья кнопка осуществляет выход из программы с закрытием всех окон (Рис. 1).



Рис. 1. Главное меню

После нажатия кнопки Играть, появляется игровое окно (Рис. 2), состоящие из надписи направления расставляемых кораблей, белого поля, на котором, собственно, находятся те самые корабли, выделенные синим цветом. Расстановка кораблей осуществляется с помощью мыши, а изменение направления с помощью клавиши «r».

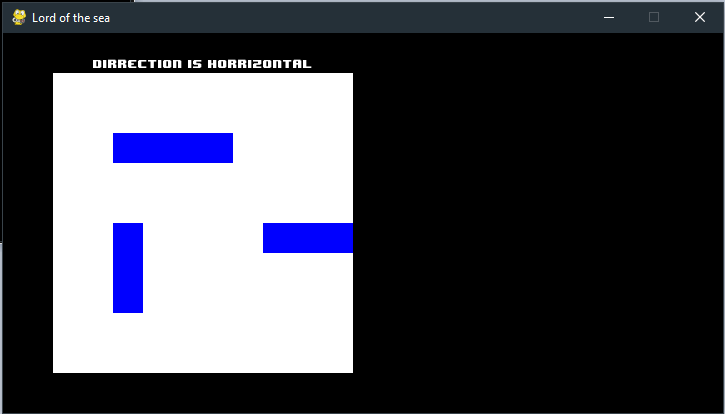


Рис. 2. Подготовка к игре

По окончанию расстановки надпись с направлением пропадает. Во второй части окна появляется вражеское игровое поле (Рис. 3), с которым игрок может взаимодействовать, кликая на нужный участок, тем самым обозначая свой выстрел, после этого бот выбирает случайную (ранее не задействованную) клетку на поле игрока. В случае попадания, клетка окрашивается в красный цвет, в ином случае(промахе) – в зеленый.

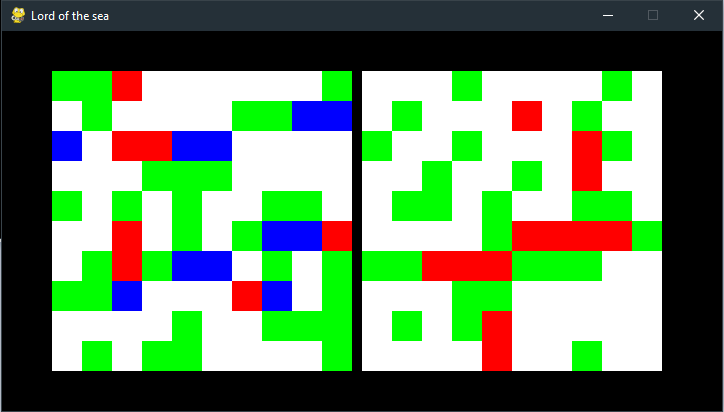


Рис. 3. Окно в момент игры

По окончанию игры появляется окно с результатом, зависящим от исхода матча, и красной кнопкой для перехода в меню. В случае победы окно выглядит как Рис.4(а), в случае поражения как Рис.4(б)

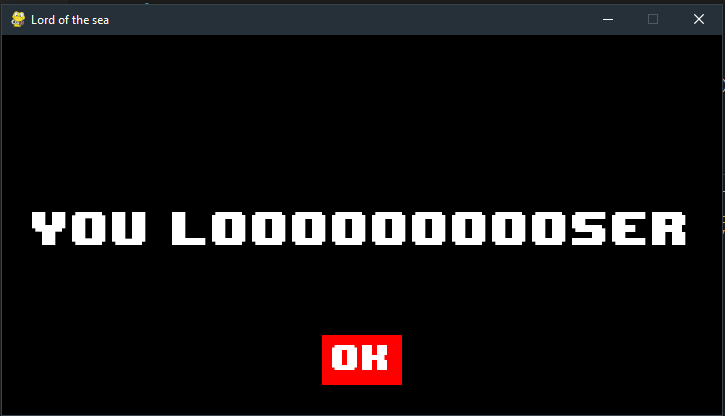
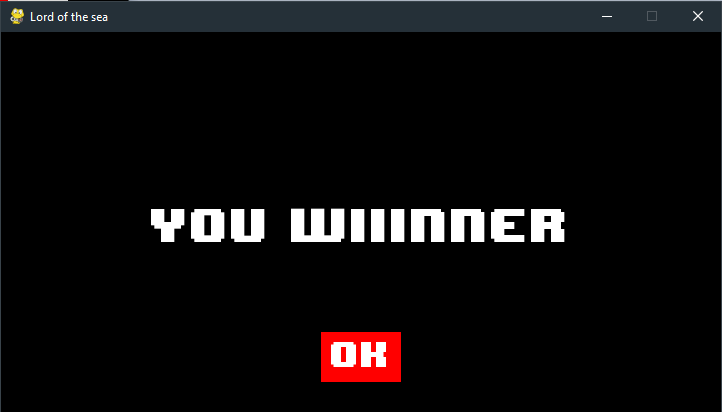


Рис. 4. Окно результата игры (а) (б)

# Обоснования выбора используемого языка программирования и среды разработки

Язык Python 3.8 является одним из популярных языков программирования с большим сообществом, множеством разнообразных библиотек. Кроме того, это простой и понятный язык программирования, позволяющий быстро создавать программный продукт. Главным недостатком является скорость выполнения, но ввиду простоты данной программы этот недостаток нивелируется. Помимо Python также рассматривался C++. C++ высокоуровневый язык, позволяющий создавать высокопроизводительные приложения, но платить за это приходится сложностью и большим количеством строчек кода. Ввиду предполагаемой простоты программы, скорость работы не является ключевым фактором, а затраченное время на разработку может оказаться значительным, поэтому выбор был сделан в пользу Python.

Средой разработки выбран Visual Studio Code, предоставляющий возможности для своей настройки, удобство и широкую поддержку пользовательских расширений. Помимо этого, рассматривался Sublime Text 3, но ввиду сложности настройки и платного распространения, выбор лег на VSC.

В качестве графического фреймворка был выбран PyGame за его простоту и удобную документацию. Также рассматривался Panda3D, движок, разработанный компанией Дисней, но ввиду простоты предполагаемой программы необходимости в таком мощном инструменте не было.

# Схема алгоритма

Main

Начало

1. Создание объекта g класса MainMenu
2. Вызов метода g.menuLoop()

Конец

Метод g.menuLoop()

Начало

1. Проверка списка событий методом checkEvents(),

Если нажат крестик на окне или кнопку “exit” осуществить выход;

Если осуществлено нажатие на кнопку “play” создать объект g класса Game и вызвать метод g.gameLoop();

1. Отрисовать элементы меню методом g.showMenu()
2. Обновить экран окна, перейти к пункту 1

Конец

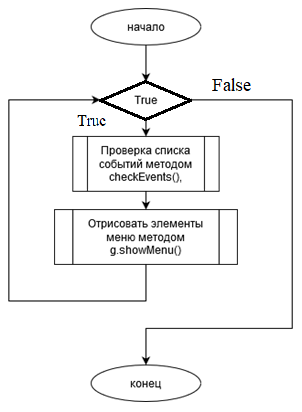
 

Рис.5 схема main() Рис. 6. схема g.menuLoop()

Метод g. gameLoop ()

Начало

1. Проверка списка событий методом checkEvents(),

Если нажат крестик на окне осуществить выход;

Если нажата клавиша “r”, сменить направление корабля;

Если нажата клавиша “q”, вернуться в метод g.menuLoop();

Если игра начата, при клике получить позицию мыши игрока, проверить на попадание и передать ход боту,

Иначе если игра завершена проверить нажатие кнопки ok, если осуществлено нажатие вернуться в метод g.menuLoop(),

Иначе при клике мышью добавить корабль на поле игрока;

1. Если игра окончена вызвать метод g.over(), создающий итоговое окно и возвращающийся в метод g.menuLoop(); перейти к пункту 5
2. Иначе если игра продолжается вызвать метод g.game(), рисующий и обновляющий поля бота и игрока, также проверяющий, остались ли живые корабли у кого-нибудь; перейти к пункту 5
3. Иначе вызвать метод g.pregame(), с помощью которого отрисовывается только одно поле игрока, на котором происходит расстановка кораблей
4. Обновить экран окна, перейти к пункту 1

Конец

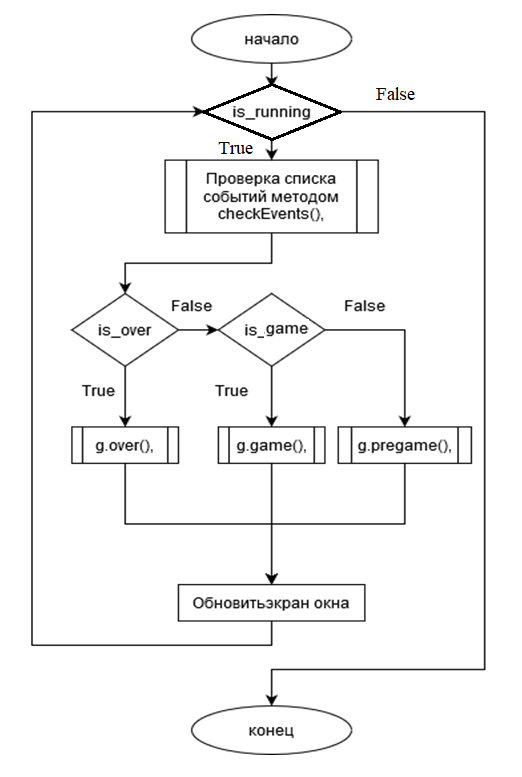


Рис. 7. Схема g.gameLoop()

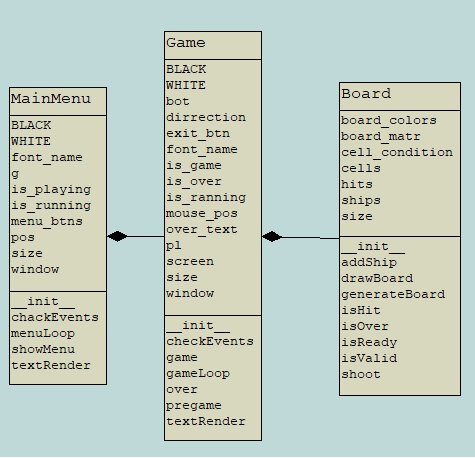


Рис. 8. Диаграмма классов

Диаграмма 5 демонстрирует отношение классов. Из рисунка следует, что все три класса являются независимыми, класс Game содержит два объекта класса Board, а класс MainMenu содержит объект класса Game.

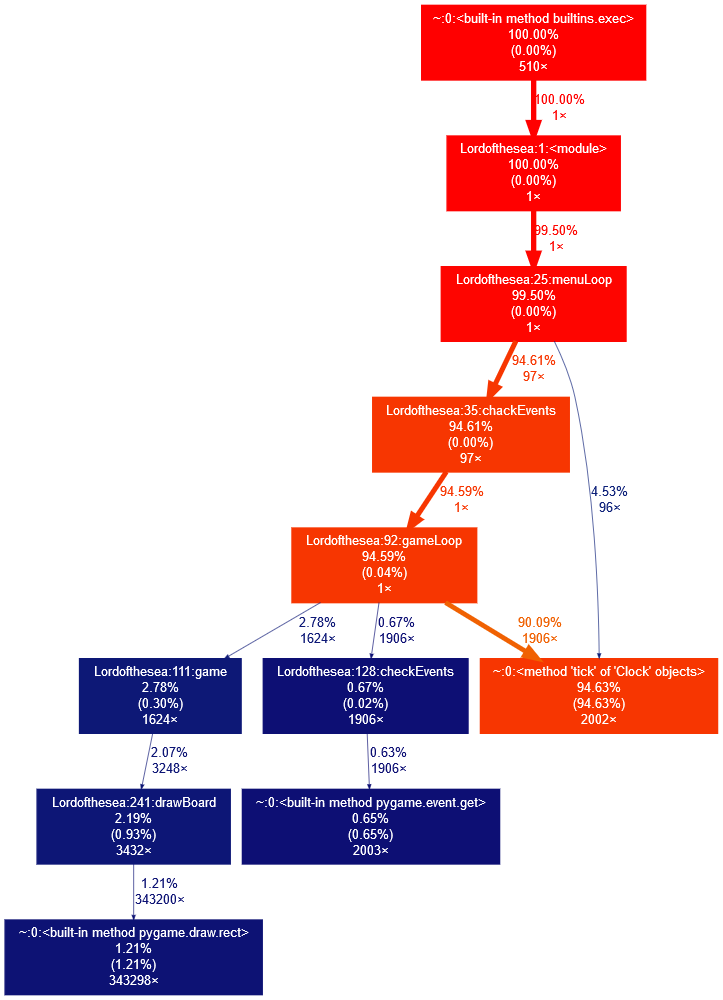


Рис 9. Краткая диаграмма вызовов, сформированная профайлером

# Текст программного кода

import pygame

import random

import sys

import os

import numpy as np

class MainMenu:

“””

Класс, который выводит окно главного меню и создает различные средства для взаимодействия с программой(кнопки)

”””

def \_\_init\_\_(self,size=(720,380)):

“””

Конструктор, инициализирующий основные параметры класса

”””

self.size = size

self.window = pygame.display.set\_mode(self.size)

self.WHITE = pygame.Color(255,255,255)

self.BLACK = pygame.Color(0,0,0)

self.font\_name = '8BIT WONDER.otf'

self.is\_running = False

self.is\_playing = False

pygame.display.set\_caption('Lord of the sea')

self.g = Game()

def menuLoop(self):

“””

Петля, в рамках которой работает программа

”””

self.is\_running = True

while self.is\_running:

self.pos = (0,0)

self.chackEvents()

self.showMenu()

pygame.display.update()

pygame.time.Clock().tick(30)

sys.exit()

def chackEvents(self):

“””

Метод, проверяющий наличие различных событий

”””

for event in pygame.event.get():

if event.type == pygame.QUIT:

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN:

if pygame.mouse.get\_pressed()[0]:

self.pos = pygame.mouse.get\_pos()

if self.menu\_btns[0].collidepoint(self.pos):

g = Game()

g.gameLoop()

if self.menu\_btns[1].collidepoint(self.pos):

print('settings')

if self.menu\_btns[2].collidepoint(self.pos):

sys.exit()

def showMenu(self):

“””

Метод, генерирующий объекты главного меню

”””

screen\_rect = self.window.get\_rect()

self.window.fill(self.BLACK)

self.textRender('Lord of The Sea',40,(screen\_rect.centerx,screen\_rect.centery/3))

self.menu\_btns = [

self.textRender('play',30,(screen\_rect.centerx,screen\_rect.centery/1.4)),

self.textRender('settings',30,(screen\_rect.centerx,screen\_rect.centery)),

self.textRender('exit',30,(screen\_rect.centerx,screen\_rect.centery\*1.3))

]

def textRender(self,text:str,text\_size:int,place:tuple):

“””

Метод, размещающий текст на экране

”””

font = pygame.font.Font(self.font\_name,text\_size)

font\_surface = font.render(text,True,self.WHITE)

font\_rect = font\_surface.get\_rect()

font\_rect.center = place

self.window.blit(font\_surface,font\_rect)

return font\_rect

class Game:

“””

Класс, в котором происходят основные вычисления и события связанные с игровым процессом

”””

def \_\_init\_\_(self,size=(720,380)):

“””

Конструктор, инициализирующий основные параметры класса

”””

self.size = size

self.window = pygame.display.set\_mode(self.size)

self.screen = self.window.subsurface((0,0,self.size[0],self.size[1]))

self.WHITE = pygame.Color(255,255,255)

self.BLACK = pygame.Color(0,0,0)

self.is\_game = False

self.is\_over = False

self.is\_ranning = True

self.font\_name = '8BIT WONDER.otf'

pygame.display.set\_caption('Lord of the sea')

self.pl = Board()

self.bot = Board()

self.bot.board\_colors[1] = pygame.Color(255,255,255)

self.bot.generateBoard()

self.dirrection = True

def gameLoop(self):

“””

Петля игрового окна

”””

while self.is\_ranning:

self.checkEvents()

self.screen.fill(self.BLACK)

if self.is\_over:

self.over()

elif self.is\_game:

self.game()

else:

self.pregame()

pygame.display.flip()

pygame.time.Clock().tick(30)

def pregame(self):

“””

Метод, осуществляющий основные действия, необходимые до начала игры

”””

self.textRender('dirrection is {}'.format('vertical' if self.dirrection else 'horrizontal'),10,(200,30))

self.screen.blit(self.pl.drawBoard(),(50,40))

#self.screen.blit(self.bot.drawBoard(),(self.pl.size\*10 + 60,40))

self.is\_game = self.pl.isReady()

def game(self):

“””

Метод, осуществляющий основные действия во время игрового процесса

”””

self.screen.blit(self.pl.drawBoard(),(50,40))

self.screen.blit(self.bot.drawBoard(),(self.pl.size\*10 + 60,40))

if self.pl.isOver():

self.is\_over = True

self.over\_text = 'You LOOOOOOOOOSER'

elif self.bot.isOver():

self.is\_over = True

self.over\_text = 'You WIIINNER'

def over(self):

“””

Метод создающий необходимый экран по окончанию игры

”””

self.textRender(self.over\_text,40,(360,190))

self.exit\_btn = pygame.draw.rect(self.screen,(255,0,0),[320,300,80,50])

self.textRender('OK',30,(360,320))

self.is\_game = False

def checkEvents(self):

“””

Метод, осуществляющий проверку событий

”””

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_r:

self.dirrection = not self.dirrection

if event.key == pygame.K\_q:

self.is\_ranning = False

if event.type == pygame.MOUSEBUTTONDOWN:

if pygame.mouse.get\_pressed()[0]:

self.mouse\_pos = pygame.mouse.get\_pos()

if self.is\_game:

x,y = self.mouse\_pos

self.bot.isHit((x-(self.pl.size\*10 + 60),y - 40))

x,y = self.bot.shoot()

self.pl.isHit((x\*self.pl.size + 15,y\*self.pl.size + 15))

elif self.is\_over:

if self.exit\_btn.collidepoint(self.mouse\_pos):

self.is\_ranning = False

else:

#print(self.mouse\_pos)

self.pl.addShip(self.mouse\_pos,self.dirrection)

def textRender(self,text:str,text\_size:int,place:tuple):

“””

Метод, осуществляющий подстановку текста на экран

”””

font = pygame.font.Font(self.font\_name,text\_size)

font\_surface = font.render(text,True,self.WHITE)

font\_rect = font\_surface.get\_rect()

font\_rect.center = place

self.screen.blit(font\_surface,font\_rect)

return font\_rect

class Board:

“””

Класс, осуществляющий основные методы для игровой доски(поля)

”””

def \_\_init\_\_(self,size = 30):

“””

Конструктор, инициализирующий основные параметры класса

”””

self.size =size

self.board\_matr = np.zeros((10,10),dtype='int32')

self.cell\_condition = [0,1,2,3]

self.ships = [1,1,2,2,2,3,3,4]

self.board\_colors = [

pygame.Color(255,255,255),

pygame.Color(0,0,255),

pygame.Color(255,0,0),

pygame.Color(0,255,0)

]

self.hits = np.zeros((10,10))

def isValid(self,pos:tuple,d:int,l:int):

“””

Метод, осуществляющий проверку на корректность постановки корабля на поле

”””

y,x = pos

l = l - 1

if (x+l > 9 and d == 0) or (y + l > 9 and d == 1):

#print('1')

return False

try:

for i in range(l+1):

if (self.board\_matr[y,x+i] == 1 and d == 0):

#print('2')

return False

elif (self.board\_matr[y+i,x] == 1 and d == 1):

#print('3')

return False

elif (((self.board\_matr[y+i,x+1] == 1) or (self.board\_matr[y+i,x-1] == 1)) and (d == 1)) or (((self.board\_matr[y+1,x+i] == 1) or (self.board\_matr[y-1,x+i] == 1)) and (d == 0)):

#print('4')

return False

if ((self.board\_matr[y+l+1,x] == 1 or self.board\_matr[y-1,x] == 1) and d == 1) or ((self.board\_matr[y,x+l+1] == 1 or self.board\_matr[y,x-1] == 1) and d == 0):

#print('5')

return False

except IndexError as e:

pass#print(e)

return True

def addShip(self,mouse\_pos:tuple,dirrection):

“””

Метод, добавляющий корабль на поле

”””

mouse\_pos = mouse\_pos[0] - 50, mouse\_pos[1] - 40

for i in range(10):

for j in range(10):

if self.cells[i][j].collidepoint(mouse\_pos):

index = (i,j)

if self.ships == []:

return False

if not self.isValid(index,dirrection,self.ships[-1]):# or self.board\_matr[i,j] == 1:

return False

self.board\_matr[i,j] = 1

l = self.ships.pop()

for k in range(l):

if dirrection == 0:

self.board\_matr[i,j+k] = 1

if dirrection == 1:

self.board\_matr[i+k,j] = 1

#print(self.board\_matr)

def generateBoard(self):

“””

Метод, генерирующий случайным способом игровое поле с кораблями для бота(возможно и для игрока)

”””

while True:

if self.ships == []:

return

pos = (random.randint(0,9),random.randint(0,9))

dirrection = random.randint(0,1)

if self.isValid(pos,dirrection,self.ships[-1]):

#print(self.ships)

l = self.ships.pop()

for k in range(l):

if dirrection == 0:

self.board\_matr[pos[0],pos[1]+k] = 1

if dirrection == 1:

self.board\_matr[pos[0]+k,pos[1]] = 1

#print(self.board\_matr)

def drawBoard(self,pos = (0,0)):

“””

Метод, отрисовывающий доску на экране

”””

screen = pygame.Surface((self.size\*10,self.size\*10))

x, y = pos

self.cells = [[]]

for i in range(10):

for j in range(10):

self.cells[i].append(pygame.draw.rect(screen,self.board\_colors[self.board\_matr[i,j]],(x + j\*self.size,y + i\*self.size,self.size,self.size)))

self.cells.append([])

return screen

def shoot(self):

while True:

pos = (random.randint(0,9),random.randint(0,9))

#print(pos)

if self.hits[pos[0],pos[1]] == 0:

self.hits[pos[0],pos[1]] = 1

return pos

def isHit(self,pos:tuple):

“””

Метод, регистрирующий попадание

”””

for i in range(10):

for j in range(10):

if self.cells[i][j].collidepoint(pos):

if self.board\_matr[i,j] == 0:

self.board\_matr[i,j] = 3

elif self.board\_matr[i,j] == 1:

self.board\_matr[i,j] = 2

def isReady(self):

“””

Метод, проверяющий возможность начать игру

”””

if self.ships == []:

#print(self.board\_matr.tolist())

return True

return False

def isOver(self):

“””

Метод, проверяющий возможность завершить игру

”””

count = 0

for i in range(10):

for j in range(10):

if self.board\_matr[i][j] == 1:

count += 1

if count == 0:

print('over')

return True

return False

if \_\_name\_\_ == '\_\_main\_\_':

pygame.init()# инициализаци основных модулей фреймворка pygame

g = MainMenu()

g.menuLoop()

# Отладка и тестирование работоспособности

Использовался метод ручного тестирования - самый простой и естественный способ. При обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которым была обнаружена ошибка.

Для установления корректности работы программы были проведены некоторые тесты:

1.Все кнопки работают по своему назначению – True

2.Корабли расставляются исключительно на игровом поле, по правилам морского боя – не выходя за границы поля и не находясь в притык друг к другу – True

3.Игрок может стрелять только по полю противника, нажатия вне его не учитываются как ход – True

4.Бот генерирует свое игровое поле по правилам морского боя – True

5.Бот стреляет только по полю игрока, при этом не попадая в уже выстрелянные клетки – True

6.Бой корректно завершается и верно определяется победитель – True

# Список литературы

1. DFC Intelligence [Электронный ресурс]: Мировое количество потребителей видеоигр превышает 3 миллиарда URL:<https://www.dfcint.com/dossier/global-video-game-consumer-population/> (дата обращения 10.12.2020)
2. Source Code & Projects [Электронный ресурс]: Морской бой на C ++ URL:<https://code-projects.org/battleship-game-in-c-with-source-code-2/> (дата обращения 10.12.2020)
3. UnityList. Games and Projects [Электронный ресурс]: Демо-версия игры Seabattle Unity. URL:<https://unitylist.com/p/123p/Sea-Battle-3D> (дата обращения 10.12.2020)
4. YouTube [Электронный ресурс]: Морской бой в Python - финальный проект CS10. URL:https://www.youtube.com/watch?v=5MuZ0L0x2WQ (дата обращения 10.12.2020)
5. Python 3.8 documentation [Электронный ресурс]: официальная документация языка программирования Python 3.8. URL:<https://docs.python.org/3/> (дата обращения 07.12.2020)
6. Pygame Front Page – pygame v2.0.1.dev1 documentation [Электронный ресурс]: официальная документация фреймворка pygame. URL:<https://www.pygame.org/docs/> ((дата обращения 07.12.2020)