20

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и

технологий

**КУРСОВАЯ РАБОТА**

по дисциплине: “**Языки программирования**”

на тему: «**Разработка компьютерной игры 2048 на языке Python»**

Выполнил**:**

Студент группы 10702220 Самцевич А. И.

Руководитель:

ст.п кафедры ПОИСиТ Тетерюкова И.О.

Минск 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКА БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и

технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по дисциплине **«Языки программирования»**

на тему **«Разработка компьютерной игры 2048»**

Выполнил**:**

Студент группы 10702220 Самцевич А. И.

Руководитель:

ст.п кафедры ПОИСиТ Тетерюкова И.О.

Минск 2021

**ОГЛАВЛЕНИЕ**

[Введение………………………………………………………………………….....4](#_Toc27143195)

[1.Техническое задание на предмет разработки и/или сценарий игры….....5](#_Toc27143196)

[2.Обоснование используемых языков и технологий………………………](#_Toc27143197)6

[3.Моделирование и проектирование программного обеспечения](#_Toc27143198)….….…7

[4.Реализация программного обеспечения…………………..……………...](#_Toc27143199)8

[5.Развертывание и тестирование программного обеспечения………..…](#_Toc27143200)10

[Заключение ……………..….………………………………………………1](#_Toc27143203)1

[Список использованной литературы и источников……………................1](#_Toc27143204)2

[ПРИЛОЖЕНИЕ А ………………………………………………………….1](#_Toc27143205)3

**ВВЕДЕНИЕ**

Индустрия компьютерных игр - сектор экономики, связанный с разработкой, продвижением и продажей компьютерных игр. В неё входит большое количество специализаций, по которым работают тысячи специалистов по всему миру. Игровая индустрия тесно связана с производством центральных процессоров и других компонентов персональных компьютеров, так как игры зачастую требуют более высоких аппаратных мощностей, чем бизнес-приложения. В настоящее время игры являются движущей силой развития компьютерных технологий, которые впоследствии применяются и для других целей.

В современном мире игровая индустрия пользуется огромной популярностью. Ее пользователей становится все больше и больше. Каждый год появляются новые технологии и оборудование, что позволяют разработчикам делать продукцию более высокого качества, сложнее и многограннее.

С написанием простой игры может справиться любой начинающий программист. Для начала нужно разобраться какому языку лучше отдать предпочтение для написания игры. Затем выучить или повторить язык, а потом выучить нужные библиотеки. Новичку для начала подойдёт 2D игра, например: змейка, крестики нолики и т.д.

В своей курсовой работе, для написания игры-ремейка, я выбрала язык программирования Python. Библиотеку я использовала PyGame.

# **Техническое задание на предмет разработки и/или сценарий игры**

Головоломка  — название жанра компьютерных игр, целью которых является решение логических задач, требующих от игрока задействования логики, стратегии и интуиции.

Такие игры появились в начале 1990-х годов, когда компьютеры стали достаточно мощными, чтобы просчитывать поведение сталкивающихся объектов сложной формы, провисающих верёвок и т. д. Одна из первых игр такого типа — The Incredible Machine. Новое рождение жанр получил вместе с сенсорными телефонами. Наиболее известная физическая головоломка новой волны — Angry Birds. К физическим головоломкам иногда относят игры Portal, The Ball.

Игра 2048 в какое-то время занимала все топы рейтингов в App Store и Google Play. Моя версия имеет самый базовый функционал – саму игру.

# **Обоснование используемых языков и технологий**

Для написания игры я использовал язык Python, библиотеку PyGame и среду разработки Visual Studio Code.

В наше время Python хорошо известен и пользуется большой популярностью. Это современный язык программирования высокого уровня. У него большая производительность и легко читаемый код. У него также удобный синтаксис, что позволяет выучить его быстрее, чем остальные языки (C, C++, C#, Java). Python поддерживает динамическую типизацию, все значения являются объектами. Функции, модули, классы - они тоже являются объектами.

Можно отметить, что набор инструментов языка Python для разработки весьма разнообразен. При написании игры я использовал такую библиотеку, как PyGame. PyGame — это библиотека модулей для Python, используемая при разработке 2d игр. Сам Python позволяет писать короткий и ясный код, что является хорошим началом для изучения особенностей разработки игр. Преимущество PyGame в легком изучении и быстром написании. Плюсом также можно отметить достаточное количество примеров в интернете, что помогает лучшему пониманию. Из-за этого PyGame и была выбрана для написания кода игры.

# **Моделирование и проектирование программного обеспечения**

В программном коде мини-игры выделяют три основных логических блока:

• Отслеживание событий, производимых пользователем.

• Изменение состояний объектов, согласно произошедшим событиям.

• Отображение объектов на экране, согласно их текущим позициям.

Эти три этапа крутятся в цикле, где каждую итерацию необходимо перерисовывать (фон, квадраты с числами).

# **Реализация программного обеспечения**

Для начала, при реализации программного обеспечения, использовалось множество разнообразных методов библиотек PyGame.

Чтобы код работал в любой операционной системе, нужно использовать функцию os.path

Метод pygame.init (), используется для запуска PyGame, pygame.display.set\_mode в частности set.mode добавляет игровое окно и меню..

Важной частью являлся рендеринг, отрисовка на экране - это, фактически, вычисление необходимой структуры пикселей. При отрисовке текста за это отвечала функция font.render().

Перейдём к самим классам:

Py2048 – класс в котором и происходят все вычисления и обработка игры.

Py2048Logic – класс отвечающий за отображение всего происходящего в игровом окне в консоли.

При запуске игры у нас сразу открывается окно с игровым полем 4 на 4 и 2 заполенными рандомно ячейками.

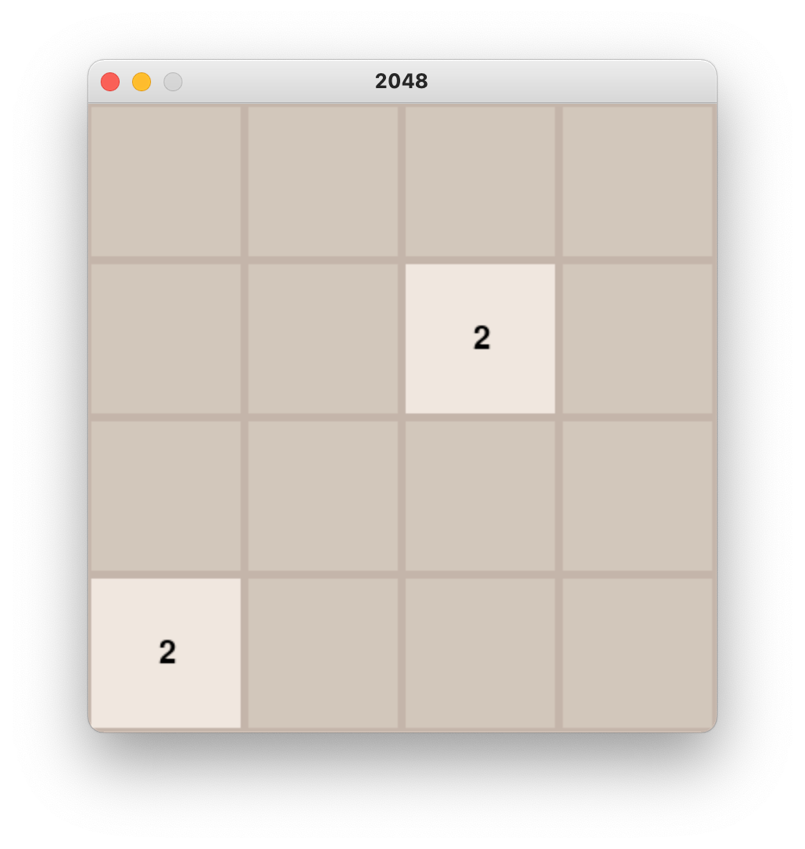


Рис. 1 – Игровое поле при запуске игры

****

Рис. 2 – игровое поле в процессе игры

# **Развертывание и тестирование программного обеспечения**

Игра выглядит так, как и ожидаешь от 2048. Максимально простой геймплей и реализация. Стоит учитывать что это первая моя игра на Python, исходя из этого можно считать что получилось довольно неплохо, но есть куда двигаться дальше.

Как минимум можно было бы добавить подсчет очков, но я не придумал как это реализовать именно для игры 2048. Так же не помешали бы на самом деле сейвы с именем игрока, но без очков в этом нет смысла по моему мнению.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была достигнута цель, заключающаяся в разработке игры 2048 на языке Python.

Для её достижения были выполнены следующие задачи:

1. Продуман и разработан геймплей игры.
2. Обосновано использование языка Python для написания кода.

[3. Осуществлено моделирование и проектирование программного обеспечения.](file:///C:\Users\razer\Downloads\Telegram%20Desktop\2_5314789606644782507%20(2).docx#_Toc27143198)

1. Реализовано программное обеспечение.
2. Протестировано программное обеспечения.

На практике были закреплены полученные знания, связанные с библиотекой Pygame на языке программирования Python, изучены новые методы работы, связанные с разработкой окон и шрифтов. Также изучены прорисовка объектов, создание игрового окна, подробно рассмотрены способы работы с клавиатурой, а также изучены методы разработки легко масштабируемого кода.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ**

1. Pygame и разработка игр:

<https://younglinux.info/pygame/pygame>

1. Библиотека Pygame / Часть 1. Введение:

<https://pythonru.com/uroki/biblioteka-pygame-chast-1-vvedenie>

1. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

# **ПРИЛОЖЕНИЕ А**

**Листинг исходных кодов**

**main.py**

import numpy as np

import random

import pygame

from pygame.locals import \*

from constants import CP

N = 4

class Py2048:

def \_\_init\_\_(self):

self.grid = np.zeros((N, N), dtype=int)

self.W = 400

self.H = self.W

self.SPACING = 5

pygame.init()

pygame.display.set\_caption("2048")

pygame.font.init()

self.myfont = pygame.font.SysFont('Comic Sans MS', 30)

self.screen = pygame.display.set\_mode((self.W, self.H))

def \_\_str\_\_(self):

return str(self.grid)

def new\_number(self, k=1):

free\_poss = list(zip(\*np.where(self.grid == 0)))

for pos in random.sample(free\_poss, k=k):

if random.random() < .1:

self.grid[pos] = 4

else:

self.grid[pos] = 2

@staticmethod

def \_get\_nums(this):

this\_n = this[this != 0]

this\_n\_sum = []

skip = False

for j in range(len(this\_n)):

if skip:

skip = False

continue

if j != len(this\_n) - 1 and this\_n[j] == this\_n[j + 1]:

new\_n = this\_n[j] \* 2

skip = True

else:

new\_n = this\_n[j]

this\_n\_sum.append(new\_n)

return np.array(this\_n\_sum)

def make\_move(self, move):

for i in range(N):

if move in 'lr':

this = self.grid[i, :]

else:

this = self.grid[:, i]

flipped = False

if move in 'rd':

flipped = True

this = this[::-1]

this\_n = self.\_get\_nums(this)

new\_this = np.zeros\_like(this)

new\_this[:len(this\_n)] = this\_n

if flipped:

new\_this = new\_this[::-1]

if move in 'lr':

self.grid[i, :] = new\_this

else:

self.grid[:, i] = new\_this

def draw\_game(self):

self.screen.fill(CP['back'])

for i in range(N):

for j in range(N):

n = self.grid[i][j]

rect\_x = j \* self.W // N + self.SPACING / 2

rect\_y = i \* self.H // N + self.SPACING / 2

rect\_w = self.W // N - self.SPACING

rect\_h = self.H // N - self.SPACING

pygame.draw.rect(self.screen,

CP[n],

pygame.Rect(rect\_x, rect\_y, rect\_w, rect\_h),

border\_radius=0)

if n == 0:

continue

text\_surface = self.myfont.render(f'{n}', True, (0, 0, 0))

text\_rect = text\_surface.get\_rect(center=(rect\_x + rect\_w / 2,

rect\_y + rect\_h / 2))

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

@staticmethod

def wait\_for\_key():

while True:

for event in pygame.event.get():

if event.type == QUIT:

return 'q'

if event.type == KEYDOWN:

if event.key == K\_UP:

return 'u'

elif event.key == K\_RIGHT:

return 'r'

elif event.key == K\_LEFT:

return 'l'

elif event.key == K\_DOWN:

return 'd'

elif event.key == K\_q or event.key == K\_ESCAPE:

return 'q'

def game\_over(self):

grid\_bu = self.grid.copy()

for move in 'lrud':

self.make\_move(move)

if not all((self.grid == grid\_bu).flatten()):

self.grid = grid\_bu

return False

return True

def play(self):

self.new\_number(k=2)

while True:

self.draw\_game()

pygame.display.flip()

cmd = self.wait\_for\_key()

if cmd == 'q':

break

old\_grid = self.grid.copy()

self.make\_move(cmd)

print(game.grid)

if self.game\_over():

print('GAME OVER!')

break

if not all((self.grid == old\_grid).flatten()):

self.new\_number()

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

game = Py2048()

game.play()

**test\_logic.py**

import numpy as np

import random

N = 4

class Py2048Logic:

def \_\_init\_\_(self):

self.grid = np.zeros((N, N), dtype=int)

def \_\_str\_\_(self):

return str(self.grid)

def new\_number(self, k=1):

free\_poss = list(zip(\*np.where(self.grid == 0)))

for pos in random.sample(free\_poss, k=k):

if random.random() < .1:

self.grid[pos] = 4

else:

self.grid[pos] = 2

@staticmethod

def \_get\_nums(this):

this\_n = this[this != 0]

this\_n\_sum = []

skip = False

for j in range(len(this\_n)):

if skip:

skip = False

continue

if j != len(this\_n) - 1 and this\_n[j] == this\_n[j+1]:

new\_n = this\_n[j] \* 2

skip = True

else:

new\_n = this\_n[j]

this\_n\_sum.append(new\_n)

return np.array(this\_n\_sum)

def make\_move(self, move):

for i in range(N):

if move in 'lr':

this = self.grid[i, :]

else:

this = self.grid[:, i]

flipped = False

if move in 'rd':

flipped = True

this = this[::-1]

this\_n = self.\_get\_nums(this)

new\_this = np.zeros\_like(this)

new\_this[:len(this\_n)] = this\_n

if flipped:

new\_this = new\_this[::-1]

if move in 'lr':

self.grid[i, :] = new\_this

else:

self.grid[:, i] = new\_this

def play(self):

self.new\_number(k=2)

while True:

print(self.grid)

cmd = input()

if cmd == 'q':

break

old\_grid = self.grid.copy()

self.make\_move(cmd)

if all((self.grid == old\_grid).flatten()):

continue

self.new\_number()

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

game = Py2048Logic()

game.play()

**test\_pygame.py**

import pygame

from pygame.locals import \*

import numpy as np

from constants import CP, TEST\_GRID

N = 4

W = 400

H = 600

SPACING = 10

def draw\_game(screen, grid, myfont):

screen.fill(CP['back'])

for i in range(N):

for j in range(N):

n = grid[i][j]

rect\_x = j \* W // N + SPACING

rect\_y = i \* H // N + SPACING

rect\_w = W // N - 2 \* SPACING

rect\_h = H // N - 2 \* SPACING

pygame.draw.rect(screen,

CP[n],

pygame.Rect(rect\_x, rect\_y, rect\_w, rect\_h),

border\_radius=0)

text\_surface = myfont.render(f'{n}', True, (0, 0, 0))

text\_rect = text\_surface.get\_rect(center=(rect\_x + rect\_w/2,

rect\_y + rect\_h/2))

screen.blit(text\_surface, text\_rect)

def wait\_for\_key():

while True:

for event in pygame.event.get():

if event.type == QUIT:

return 'q'

if event.type == KEYDOWN:

if event.key == K\_UP:

return 'u'

elif event.key == K\_RIGHT:

return 'r'

elif event.key == K\_LEFT:

return 'l'

elif event.key == K\_DOWN:

return 'd'

elif event.key == K\_q or event.key == K\_ESCAPE:

return 'q'

def main():

pygame.init()

pygame.display.set\_caption("2048")

pygame.font.init()

myfont = pygame.font.SysFont('Comic Sans MS', 30)

screen = pygame.display.set\_mode((W, H))

running = True

while running:

np.random.shuffle(TEST\_GRID)

draw\_game(screen, TEST\_GRID, myfont)

pygame.display.flip()

key = wait\_for\_key()

if key == 'q':

running = False

if \_\_name\_\_ == "\_\_main\_\_":

main()