ОТЧЁТ ПО ПРОЕКТУ

Студентов группы № БКК2202

|  |
| --- |
| Написание курса по Pygame на примере создания Pacman |
| (название проекта) |

|  |
| --- |
| Кузин Антон Алексеевич |
| (Ф.И.О.) |
|  |

МТУСИ, 2024 г.

**Оглавление**

[Введение 3](#_Toc182511403)

[Цели и Задачи 4](#_Toc182511404)

[1. Теоретические материалы 5](#_Toc182511405)

[1.1. Введение в Pygame 5](#_Toc182511406)

[1.2. Создание Объектов 7](#_Toc182511407)

[1.3. Использование спрайтов 9](#_Toc182511408)

[2. Написание Пакмана 11](#_Toc182511409)

[2.1. Инициализация Pygame и переменных 11](#_Toc182511410)

[2.2. Доска и Обработчик Событий 12](#_Toc182511411)

[2.3. Пакман 15](#_Toc182511412)

[2.3.1 Реализация движения 16](#_Toc182511413)

[2.3.2 Реализация анимации 20](#_Toc182511414)

[2.4 Фишки 21](#_Toc182511415)

[2.5 Призраки 23](#_Toc182511416)

[3. Заключение 29](#_Toc182511417)

# Введение

Игры – это неотъемлемая часть человеческой истории, пронизывающая ее с самых ранних времен. Интерес человека к играм никогда не угасал, однако изменялись сами игры. Поначалу это были простые игры с камешками и костями, после человек придумал шахматы и го. Далее в эпоху средневековья получили свое развитие карточные игры. А сейчас, в эпоху бурного развития компьютеров, мы получили возможность создавать себе новые миры. Игры стали не только развлечением, но и инструментом обучения, тренировки и исследования. Они помогают развивать мышление, решать проблемы, улучшать координацию и реакцию.

Мы, вдохновившись многообразием компьютерных игр, решили обратиться к классике и повторить известнейшую игру – Пакман. А также написать мини курс, по которому каждый сможет написать простую игру!

# Цели и Задачи

Целью нашей работы является изучить базовые функции библиотеки Pygame и написать Пакман на Pygame.

Задачи:

1. Изучить основные функции библиотеки Pygame, а именно:
   1. Создание окна игры, а также создание игрового цикла;
   2. Создание объектов, а также изучение логики столкновения объектов друг с другом;
   3. Создание спрайтов и их соединения с игровыми объектами.
2. Продумать случаи проигрыша и выигрыша игрока.
3. Написать Pacman с функционалом увеличения очков, возможностью выигрыша за получение максимального количества очков и проигрыша за столкновение с приведением.

# 1. Теоретические материалы

## 1.1. Введение в Pygame

Наш путь разработчиков игр начинается с малого. Чтобы работать с библиотекой Pygame мы сначала должны ее установить, прописываем в командной строке:

pip install pygame

После установки создаем файл с названием main.py и в него прописываем:

import pygame

pygame.init()

Тем самым мы инициализируем pygame.

Следующим шагом является создание окна, в котором будет отображаться наша игра. Для этого мы используем следующую функцию.

screen = pygame.display.set\_mode((800, 600))

Сердцем каждой игры является игровой цикл. Он обеспечивает постоянную работу игры, обрабатывает события, обновляет состояние игры и отображает изменения на экране.

running = True

while running:

# Обработка событий

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Обновление состояния игры

# Отрисовка графики

# Обновление экрана

pygame.display.flip()

Этот цикл работает бесконечно, пока переменная running равна True. Внутри цикла мы выполняем следующие действия:

1. Обработка событий: Проверяем, не произошло ли каких-либо событий, например, нажатие клавиши, движение мыши, закрытие окна.
2. Обновление состояния игры: Изменяем состояние игры, например, перемещаем персонажа, проверяем столкновения, обновляем счет.
3. Отрисовка графики: Очищаем экран, рисуем фон, персонажей, объекты и другие элементы игры.
4. Обновление экрана: Перерисовываем весь экран, отображая все изменения, внесенные на предыдущих шагах.

Для рисования графики на экране мы используем функции из модуля pygame.draw:

screen.fill((255, 255, 255)) # Заливка экрана белым цветом

pygame.draw.rect(screen, (255, 0, 0), (100, 100, 50, 50)) # Рисование красного прямоугольника

pygame.draw.circle(screen, (0, 0, 255), (400, 300), 50) # Рисование синего круга

В данном коде:

* screen.fill(): Заполняет экран заданным цветом.
* pygame.draw.rect(): Рисует прямоугольник.
* pygame.draw.circle(): Рисует круг.

Также важно знать как в pygame задается цвет.

В Pygame цвет задается кортежем из трех чисел: (красный, зеленый, синий) в диапазоне от 0 до 255. Например, (255, 0, 0) соответствует красному цвету, (0, 255, 0) - зеленому, (0, 0, 255) - синему.

В Pygame события - это действия, которые происходят во время игры. Например:

pygame.QUIT: Пользователь закрыл окно.

pygame.KEYDOWN: Пользователь нажал клавишу.

pygame.KEYUP: Пользователь отпустил клавишу.

pygame.MOUSEBUTTONDOWN: Пользователь нажал кнопку мыши.

pygame.MOUSEBUTTONUP: Пользователь отпустил кнопку мыши.

Чтобы обрабатывать события, мы используем функцию pygame.event.get(), которая возвращает список всех произошедших событий.

for event in pygame.event.get():

  if event.type == pygame.QUIT:

    running = False

  elif event.type == pygame.KEYDOWN:

    if event.key == pygame.K\_LEFT:

      # Обработка нажатия клавиши влево

    elif event.key == pygame.K\_RIGHT:

      # Обработка нажатия клавиши вправо

В конце цикла игры нам нужно закрыть Pygame с помощью функции

pygame.quit()

## 1.2. Создание Объектов

Для создания персонажа мы можем использовать несколько подходов:

* Прямоугольник: Простейший вариант – использовать pygame.Rect():

import pygame

# ... (инициализация Pygame, создание окна)

player\_rect = pygame.Rect(100, 300, 50, 50) # Создаем прямоугольник для игрока

running = True

while running:

  # ... (обработка событий)

  screen.fill((255, 255, 255)) # Заливка экрана белым

  pygame.draw.rect(screen, (0, 255, 0), player\_rect) # Отрисовка прямоугольника игрока

  # ... (обновление экрана)

pygame.quit()

* Изображение: Мы можем загрузить изображение и использовать его как визуальное представление персонажа:

import pygame

# ... (инициализация Pygame, создание окна)

player\_image = pygame.image.load("player.png").convert\_alpha() # Загружаем изображение персонажа

player\_rect = player\_image.get\_rect(center=(100, 300)) # Создаем Rect с центром в (100, 300)

running = True

while running:

  # ... (обработка событий)

  screen.fill((255, 255, 255)) # Заливка экрана белым

  screen.blit(player\_image, player\_rect) # Отображение изображения персонажа

# ... (обновление экрана)

pygame.quit()

Чтобы сделать персонажа подвижным, мы будем обновлять его координаты в цикле игры:

import pygame

# ... (инициализация, создание окна, загрузка изображения персонажа)

player\_speed = 5 # Скорость движения

running = True

while running:

  # ... (обработка событий)

  keys = pygame.key.get\_pressed() # Получаем состояние всех клавиш

  if keys[pygame.K\_LEFT]:

    player\_rect.x -= player\_speed

  if keys[pygame.K\_RIGHT]:

    player\_rect.x += player\_speed

  # ... (очистка экрана, отрисовка, обновление экрана)

pygame.quit()

Процесс создания других объектов (например, препятствий, врагов, бонусов) очень похож на создание персонажа. Мы можем использовать прямоугольники, изображения или даже сочетание обоих методов.

Чтобы реализовать взаимодействие между объектами, нам необходимо уметь определять, столкнулись ли они. Pygame предоставляет метод colliderect(), который проверяет, пересекаются ли два прямоугольника:

if player\_rect.colliderect(enemy\_rect):

  # Персонаж столкнулся с врагом

## 1.3. Использование спрайтов

Спрайт - это отдельный графический объект, который можно размещать и анимировать на экране. Спрайты - это изображения (например, персонаж, враг, пуля, предмет), которые можно свободно перемещать, вращать, масштабировать и даже анимировать.

Чтобы создать спрайт в Pygame мы используем класс pygame.sprite.Sprite для создания спрайтов.

import pygame

class Player(pygame.sprite.Sprite):

  def \_\_init\_\_(self, image\_path, x, y):

    super().\_\_init\_\_()

    self.image = pygame.image.load(image\_path).convert\_alpha()

    self.rect = self.image.get\_rect(topleft=(x, y))

  def update(self):

    # Логика обновления спрайта, например, движение

    keys = pygame.key.get\_pressed()

    if keys[pygame.K\_LEFT]:

      self.rect.x -= 5

    if keys[pygame.K\_RIGHT]:

      self.rect.x += 5

    if keys[pygame.K\_UP]:

      self.rect.y -= 5

    if keys[pygame.K\_DOWN]:

      self.rect.y += 5

где

\_\_init\_\_ - Инициализирует спрайт:

* Загружает изображение.
* Создает rect - прямоугольник, который определяет позицию и размер спрайта.
* update: Метод, который вызывается в цикле игры для обновления спрайта. В этом методе мы можем реализовать логику движения, анимации или других изменений.

Чтобы анимировать спрайты, мы можем использовать несколько изображений, которые будут сменяться в определенном порядке.

* Создайте список изображений: images = [image1, image2, image3]
* Измените изображение в методе update:
* Используйте счетчик кадров, чтобы определить, какое изображение нужно отображать.
* Когда счетчик кадров достигает определенного значения, переключайтесь на следующее изображение в списке.

# 2. Написание Пакмана

Игру Пакман можно функционально разделить на несколько частей:

* Поле игры – Лабиринт;
* Фишки, разбросанные по полю игры, собирая которые можно выиграть.
* Главный герой – Пакман;
* Противники и логика их работы – Приведения;

## 2.1. Инициализация Pygame и переменных

В самом начале мы должны инициализировать библиотеки и переменные размера окна и счетчика событий, а также константы, которые мы будем использовать во всем проекте (коды кнопок WASD и RGD коды цветов).

import pygame

import random

BOARD\_WIDTH, BOARD\_HEIGHT = 19, 22 # размер в клетках

CELL\_SIZE = 25

WINDOW\_SIZE = WIDTH, HEIGHT = BOARD\_WIDTH \* CELL\_SIZE + 100, BOARD\_HEIGHT \* CELL\_SIZE # добавляем 100 для счетчика очков

TICK = pygame.USEREVENT + 1  # событие, нужно для отсчета одного момента

PACMAN\_MOTION = pygame.USEREVENT + 1  # событие для отсчета смены кадра

BLACK = (0, 0, 0)

BLUE = (0, 0, 128)

PINK = (252, 15, 192)

RIGHT\_KEY = 100

UP\_KEY = 119

LEFT\_KEY = 97

DOWN\_KEY = 115

KEYS = [RIGHT\_KEY, UP\_KEY, LEFT\_KEY, DOWN\_KEY]

pygame.init()

## 2.2. Доска и Обработчик Событий

На данном этапе стоит ввести поле игры, пакман довольно удобен в своем проектировании, поэтому мы можем создать список с вложенными списками и ориентироваться по полю с помощью board[i][j], где

i – координата по строке,

j – координата по столбцу

Код ниже является примером такой доски

board = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

        [1, 0, 0, 0, 3, 0, 0, 0, 3, 1],

        [1, 0, 1, 1, 0, 1, 1, 1, 0, 1],

        [1, 3, 0, 0, 3, 0, 0, 0, 3, 1],

        [1, 0, 1, 1, 0, 1, 1, 1, 3, 1],

        [1, 3, 3, 1, 3, 3, 3, 0, 0, 1],

        [1, 1, 0, 1, 0, 1, 0, 1, 1, 1],

        [1, 3, 3, 0, 3, 1, 3, 0, 3, 1],

        [1, 0, 1, 1, 1, 1, 1, 1, 0, 1],

        [1, 0, 0, 0, 0, 0, 0, 0, 3, 1],

        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]

Еще одной отличительной способностью такого подхода является гибкость самих клеток, так например на нашем поле есть разделения клеток:

0 - клетка, по которой можно ходить

1 – стена

2 - стенка выхода привидений

3 - узел, расхождение путей

Далее мы вводим класс Board, который рендерит поле игры

class Board:

    def \_\_init\_\_(self, screen):

        self.screen = screen

        BOARD\_WIDTH = 19  # ширина поля

        BOARD\_HEIGHT = 22  # высота поля

        CELL\_SIZE = 25  # размер клетки в пикселях

        self.render()

    def \_fill\_cell(self, x, y, color):

        pygame.draw.rect(

            self.screen,

            color,

            (

                x,

                y,

                CELL\_SIZE,

                CELL\_SIZE

            ),

            width=0)

    def render(self):

        # рендер поля

        for x in range(BOARD\_WIDTH):

            for y in range(BOARD\_HEIGHT):

                if board\_list[y][x] == 0 or board\_list[y][x] == 3:

                    self.\_fill\_cell(x \* CELL\_SIZE, y \* CELL\_SIZE, BLACK)

                if board\_list[y][x] == 1:

                    self.\_fill\_cell(x \* CELL\_SIZE, y \* CELL\_SIZE, BLUE)

                if board\_list[y][x] == 2:

                    self.\_fill\_cell(x \* CELL\_SIZE, y \* CELL\_SIZE, PINK)

А также вводим класс Game, который инициализирует класс Board и отвечает за обработку событий

class Game:

    def \_\_init\_\_(self):

        self.screen = pygame.display.set\_mode(WINDOW\_SIZE)

        board = Board(self.screen)

        pygame.time.set\_timer(TICK, 15)

        pygame.time.set\_timer(PACMAN\_MOTION, 10)

        pygame.display.flip()

        self.run()

    def run(self):

        running = True

        PacmanCurrentKey = ''

        while running:

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    running = False

            pygame.display.flip()

        pygame.quit()

На этом этапе очень важно рассмотреть таймеры.

pygame.time.set\_timer(TICK, 15)

pygame.time.set\_timer(PACMAN\_MOTION, 10)

В данный момент я ввожу 2 таймера:

1. Tick – Данный таймер будет отвечать за обработку нажатий, а также столкновения объектов
2. Pacman\_Motion – Данный таймер отвечает за перемещения всех объектов в игре (Призраков и Пакмана).

Важно, что числа после названия таймеров являются периодичностью срабатывания таймера (каждые 15 миллисекунд и каждые 10 миллисекунд соответственно).

Если мы на данном этапе запустим игру, то увидим пустой лабиринт Рис.1.

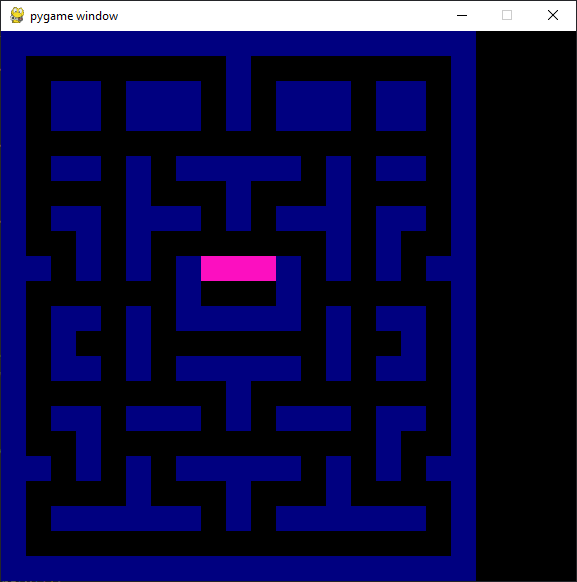


Рис. 1 Пустой Лабиринт

## Пакман

Вводим главного персонажа игры – Пакмана. У него есть определенный функционал:

1. Ходить по лабиринту и не проходить сквозь стены
2. Собирать фишки (которые будут добавлены позже)
3. Изменять направление своего взгляда в зависимости от направления движения

Для начала определим класс Pacman:

class Pacman(Board):

    def \_\_init\_\_(self, screen):

        super().\_\_init\_\_(screen)

        self.screen = screen

        self.retset = set()

        self.y = 12

self.x = 9

        self.PacmanCurrentPos = (225, 300)

        self.\_load\_sprite()

        self.pacman\_sprites = {

            "R": ['data/pcmn\_circ.png', 'data/pcmn\_right\_2.png', 'data/pcmn\_right\_3.png', 'data/pcmn\_right\_2.png'],

            "U": ['data/pcmn\_circ.png', 'data/pcmn\_up\_2.png', 'data/pcmn\_up\_3.png', 'data/pcmn\_up\_2.png'],

            "L": ['data/pcmn\_circ.png', 'data/pcmn\_left\_2.png', 'data/pcmn\_left\_3.png', 'data/pcmn\_left\_2.png'],

            "D": ['data/pcmn\_circ.png', 'data/pcmn\_down\_2.png', 'data/pcmn\_down\_3.png', 'data/pcmn\_down\_2.png'],

        }

        self.sprite\_direction = "L"

        self.currentkey = 0

        self.count = 0

    def \_load\_sprite(self):

        self.all\_sprites = pygame.sprite.Group()

        self.main\_pacman\_sprite = pygame.sprite.Sprite()

        self.main\_pacman\_sprite.image = pygame.image.load(

'data/pcmn\_left\_3.png'

)

        self.main\_pacman\_sprite.rect = self.main\_pacman\_sprite.image.get\_rect()

        self.main\_pacman\_sprite.add(self.all\_sprites)

Разберем переменные класса и их использование:

* self.screen = screen – определяем screen для возможности отображения спрайтов и обновления поля игры в классе Pacman;
* self.retset = set() – множество, которое используется для определения возможных путей, когда Пакман своим спрайтом заслоняет ровно одну клетку, и ему нужно идти дальше;
* self.y, self.x = 12, 9 – координаты Пакмана во вложенном списке, в дальнейшем изменяются от его местоположения;
* self.PacmanCurrentPos = (225, 300) – координаты Пакмана в пикселях на самом игровом поле;
* self.\_load\_sprite() – вызывает функцию, которая загружает спрайт Пакмана;
* self.pacman\_sprites = {

    "R": [...],

    "U": [...],

    "L": [...],

    "D": [...],

} – Пути в файловой системе до спрайтов Пакмана в различных положения и в различном раскрытии рта;

* self.sprite\_direction = "L" – сохраняет направление головы при повороте пакмана, связан со списком self.pacman\_sprites для большего удобства;
* self.currentkey = 0 – сохраняет код последней нажатой кнопки, для попиксельного смещения пакмана на поле. Хранит одно значения до момента, пока пакман не свернет или не упрется в стену;
* self.count = 0 – счетчик для смены спрайтов пакмана, нужен для анимации, иттерирует списки в self.pacman\_sprites;

### Реализация движения

Любое передвижения объектов в играх построено на изменении координат объектов, однако мы имеем некоторое преимущество, так как пакман движется в лабиринте. Схема передвижения пакмана выглядит вот так:



Данная схема проверок реализована в двух функциях класса Pacman, которые вызываются из цикла событий:

def pacman\_movement(self, key, cy, cx):

        y = cy // CELL\_SIZE

        x = cx // CELL\_SIZE

        self.retset = set()

        horkeycheck = (key == LEFT\_KEY or key == RIGHT\_KEY)

        verkeycheck = (key == UP\_KEY or key == DOWN\_KEY)

        ycellcheck = cy % CELL\_SIZE == 0

        xcellcheck = cx % CELL\_SIZE == 0

        if (horkeycheck and ycellcheck) or (verkeycheck and xcellcheck):

            if board\_list[y][(cx + 26) // CELL\_SIZE] not in [1, 2]:

                self.retset.add(RIGHT\_KEY)

            if board\_list[(cy + 26) // CELL\_SIZE][x] not in [1, 2]:

                self.retset.add(DOWN\_KEY)

            if board\_list[y][(cx - 1) // CELL\_SIZE] not in [1, 2]:

                self.retset.add(LEFT\_KEY)

            if board\_list[(cy - 1) // CELL\_SIZE][x] not in [1, 2]:

                self.retset.add(UP\_KEY)

            if key in self.retset:

                self.currentkey = key

                self.pacman\_move(key)

        else:

            self.pacman\_move(self.currentkey)

Данная функция вызывается из цикла событий и производит проверку – “стоит ли Пакман на ровной клетке?”, то есть совпадает ли положение спрайта пакмана с клеткой на которой он находится, так как спрайты пакмана и клетки равны 25 пикселям, мы можем легко проверить данное условие.

Если условие выполняется, то мы записываем клавишу на которую нажал игрок в отдельную переменную в цикле событий, чтобы потом поменять направление пакмана в нужную сторону.

Следующая функция непосредственно перемещает пакмана по полю в соответствии с нынешним направлением, которое записано в self.currentkey.

def pacman\_move(self, key):

        if key == LEFT\_KEY:

            x = (self.PacmanCurrentPos[0] - 1) // CELL\_SIZE

            y = (self.PacmanCurrentPos[1]) // CELL\_SIZE

            if board\_list[y][x] not in [1, 2]:

                self.\_fill\_cell(self.PacmanCurrentPos[0], self.PacmanCurrentPos[1], BLACK)

                self.PacmanCurrentPos = (self.PacmanCurrentPos[0] - 1, self.PacmanCurrentPos[1])

                self.sprite\_direction = "L"

        elif key == DOWN\_KEY:

            x = (self.PacmanCurrentPos[0]) // CELL\_SIZE

            y = (self.PacmanCurrentPos[1] + 1) // CELL\_SIZE

            if board\_list[y][x] not in [1, 2]:

                self.\_fill\_cell(self.PacmanCurrentPos[0], self.PacmanCurrentPos[1], BLACK)

                self.PacmanCurrentPos = (self.PacmanCurrentPos[0], self.PacmanCurrentPos[1] + 1)

                self.sprite\_direction = "D"

        elif key == RIGHT\_KEY:

            x = (self.PacmanCurrentPos[0] + 1) // CELL\_SIZE

            y = (self.PacmanCurrentPos[1]) // CELL\_SIZE

            if board\_list[y][x] not in [1, 2]:

                self.\_fill\_cell(self.PacmanCurrentPos[0], self.PacmanCurrentPos[1], BLACK)

                self.PacmanCurrentPos = (self.PacmanCurrentPos[0] + 1, self.PacmanCurrentPos[1])

                self.sprite\_direction = "R"

        elif key == UP\_KEY:

            x = (self.PacmanCurrentPos[0]) // CELL\_SIZE

            y = (self.PacmanCurrentPos[1] - 1) // CELL\_SIZE

            if board\_list[y][x] not in [1, 2]:

                self.\_fill\_cell(self.PacmanCurrentPos[0], self.PacmanCurrentPos[1], BLACK)

                self.PacmanCurrentPos = (self.PacmanCurrentPos[0], self.PacmanCurrentPos[1] - 1)

                self.sprite\_direction = "U"

        self.main\_pacman\_sprite.rect.x = self.PacmanCurrentPos[0]

        self.main\_pacman\_sprite.rect.y = self.PacmanCurrentPos[1]

Чтобы функционал движения заработал, нужно добавить выполнение функций в цикл событий, на данном этапе цикл событий выглядит вот так

running = True

PacmanCurrentKey = ''

while running:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            running = False

        if event.type == pygame.KEYDOWN:  # проверка по кнопкам ASDW

            if event.key == LEFT\_KEY:  # проверяем A

                PacmanCurrentKey = LEFT\_KEY

                self.pacman.pacman\_movement(

                    LEFT\_KEY,

                    self.pacman.PacmanCurrentPos[1],

                    self.pacman.PacmanCurrentPos[0]

                )

            if event.key == DOWN\_KEY:  # проверяем S

                PacmanCurrentKey = DOWN\_KEY

                self.pacman.pacman\_movement(

                    DOWN\_KEY,

                    self.pacman.PacmanCurrentPos[1],

                    self.pacman.PacmanCurrentPos[0]

                )

            if event.key == RIGHT\_KEY:  # проверяем D

                PacmanCurrentKey = RIGHT\_KEY

                self.pacman.pacman\_movement(

                    RIGHT\_KEY,

                    self.pacman.PacmanCurrentPos[1],

                    self.pacman.PacmanCurrentPos[0]

                )

            if event.key == UP\_KEY:  # проверяем W

                PacmanCurrentKey = UP\_KEY

                self.pacman.pacman\_movement(

                    UP\_KEY,

                    self.pacman.PacmanCurrentPos[1],

                    self.pacman.PacmanCurrentPos[0]

                )

        if event.type == TICK:

            self.pacman.pacman\_movement(

                PacmanCurrentKey,

                self.pacman.PacmanCurrentPos[1],

                self.pacman.PacmanCurrentPos[0]

            )

        if event.type == PACMAN\_MOTION:

            self.pacman.motion\_counting()

    pygame.display.flip()

pygame.quit()

### Реализация анимации

Анимации в игре являются важной частью отклика игры на действия пользователя. Чтобы добавить анимацию в Pygame нужно последовательно сменять спрайты на объекте. Данный функционал реализован с помощью функции motion\_counting.

def motion\_counting(self):

# счетчик для смены спрайта пакмана

    self.count = (self.count + 1) % 4

    self.main\_pacman\_sprite.image = pygame.image.load(

self.pacman\_sprites[self.sprite\_direction][self.count]

)

    self.all\_sprites.draw(self.screen)

    pygame.display.flip()

В данной функции мы добавляем к счетчику единицу и получаем остаток от деления на 4, чтобы не хранить в оперативной памяти миллионные значения счетчика для анимации. Вызов данной функции происходит каждый тик таймера PACMAN\_MOTION.

Смена спрайтов исходит из self.count, который принимает значения: 0, 1, 2, 3. Соответственно удобнее будет, если список, содержащий пути к картинкам спрайтов будет иметь длину 4. Поэтому мы вводим словарь self.pacman\_sprites, который содержит 4 ключа (соответствующие 4-м направлениям взора пакмана), а в каждом списке содержится 4 пути в таком порядке.



## Фишки

Выигрыш немаловажная часть каждой игры, нам нужно добавить фишки, которые при их съедении пакманом будут прибавлять очки победы.

Весь функционал фишек, а также их столкновения с пакманом прописан в классе Dots

class Dots(Pacman, pygame.sprite.Sprite):

    # класс точечек, которые ест пакман

    def \_\_init\_\_(self, screen):

        super().\_\_init\_\_(screen)

        self.screen = screen

        self.dots = pygame.sprite.Group()

        self.score = 0

В переменной score будет храниться количество съеденных пакманом точек.

Чтобы отрисовать точки, добавляем функцию render\_dots

def render\_dots(self):

# рендер точечек

for y in range(len(board\_list)):

for x in range(len(board\_list[y])):

if board\_list[y][x] == 0 or board\_list[y][x] == 3:

self.dot = pygame.sprite.Sprite()

self.dot.image = pygame.image.load('data/dot.png')

# спрайт точечек

self.dot.rect = self.dot.image.get\_rect()

self.dot.add(self.dots)

self.dot.rect.x = x \* CELL\_SIZE

self.dot.rect.y = y \* CELL\_SIZE

self.dots.draw(self.screen)

pygame.display.flip()

Данная функция будет отрисовывать точки во всех свободных клетках на поле, а также добавит все точки в группу спрайтов для более удобного вычисления столкновений с пакманом.

Нам также нужно вычислять счет, поэтому добавляем еще несколько функций:

def update\_dots(self):

    self.dots.draw(self.screen)

    pygame.display.flip()

def update(self, pos):

    self.main\_pacman\_sprite.rect.x = pos[0]

    self.main\_pacman\_sprite.rect.y = pos[1]

    if pygame.sprite.spritecollideany(self.main\_pacman\_sprite, self.dots):

        self.score += 1

        self.score\_calc()

    pygame.sprite.spritecollide(

self.main\_pacman\_sprite,

self.dots,

True

)

def score\_calc(self):

    # очки игрока

    font = pygame.font.Font(None, 25)

    text = font.render(f"score {self.score \* 10}", True, (255, 255, 0))

    place = text.get\_rect(

        center=(525, 20))

    text\_w = text.get\_width()

    text\_h = text.get\_height()

    pygame.draw.rect(self.screen, (0, 0, 0), (place[0], place[1],

                                            text\_w, text\_h), 0)

    self.screen.blit(text, place)

def score\_update(self):

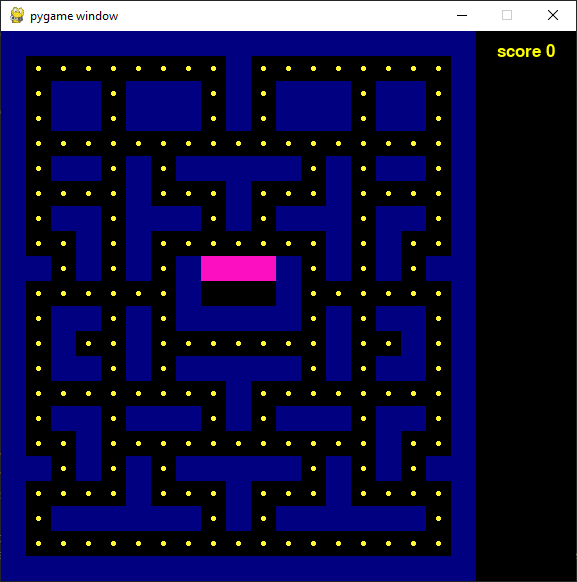
    # для главного игрового цикла

    return self.score \* 10

Каждая функция отвечает за свою логику:

* update – рассчитывает столкновения пакмана с точками каждый тик и в случае столкновения прибавляет к счету 10 очков.
* update\_dots – обновляет точки на поле, ведь если точка съедена, на поле она отображаться больше не должна.
* score\_calc – выводит на экран счетчик очков, который также обновляется каждый тик.
* score\_update – возвращает значение счетчика count умноженное на 10, ведь выводим мы не количество съеденных точек, а число очков.

Теперь поле выглядит вот так:



## Призраки

Собрать фишки чтобы выиграть это хорошо, но нет никакого смысла в игре без противников – добавляем призраков.

Весь функционал призраков будет прописан в классе Ghosts

class Ghosts(Pacman, pygame.sprite.Sprite):

    # класс привидений, где прописана вся их логика

    def \_\_init\_\_(self, screen):

        super().\_\_init\_\_(screen)

        self.screen = screen

        self.CianCurrentPos = (200, 250)

        self.RedCurrentPos = (225, 200)

        self.YellCurrentPos = (225, 250)

        self.PinkCurrentPos = (250, 250)

        # начальное положение привидений

        self.ghosts = pygame.sprite.Group()

        self.pink\_cell = [(250, 225), (225, 225), (200, 225)]

        self.ghostpos = []

        self.ghostsmoves = {}

Всего призраков будет 4, начальное положение каждого задано соответствующим кортежем для удобства. В переменной ghosts мы будем хранить спрайты всех привидений. В переменной ghotspos хранятся объекты спрайтов призраков, которые в свою очередь содержат информацию о расположении призраков на поле. В переменной ghostmoves содержится направление движения каждого призрака до момента следующей развилки.

Для инициализации привидений мы используем функцию render\_ghosts

def render\_ghosts(self):

        ghosts\_colors = [

            'data/ghostcian.png',

            'data/ghostred.png',

            'data/ghostyellow.png',

            'data/ghostpink.png'

        ]

        for g\_color in ghosts\_colors:

            self.ghost = pygame.sprite.Sprite()

            self.ghost.image = pygame.image.load(g\_color)

            self.ghost.rect = self.ghost.image.get\_rect()

            self.ghost.rect.x = self.CianCurrentPos[0]

            self.ghost.rect.y = self.CianCurrentPos[1]

            self.ghost.add(self.ghosts)

            self.ghostsmoves[self.ghost] = ''

            self.ghostpos.append(self.ghost)

        self.ghosts.draw(self.screen)

        pygame.display.flip()

Создание спрайтов происходит в цикле, так как с точки зрения объектов они ничем не отличаются, кроме самой картинки спрайта.

for g in *self*.ghostpos:

    x = g.rect.x

    y = g.rect.y

    cx = x // CELL\_SIZE

    cy = y // CELL\_SIZE

    if x % CELL\_SIZE == 0 and y % CELL\_SIZE == 0:

        oklist = {2, 3, 4}

        notokey = {1}

        goodmoves = []

        wflag = True

        aflag = True

        sflag = True

        dflag = True

        if board\_list[cy][cx] == 3 or board\_list[cy][cx] == 4:

            if board\_list[cy + 1][cx] == 2:

                oklist.remove(2)

                notokey.add(2)

            for i in range(1, 9):

                if (cx + i) <= 18 and (cx - i) >= 0:

                    if board\_list[cy][cx + i] in oklist and dflag:

                        goodmoves.append('d')

                        dflag = False

                    elif board\_list[cy][cx + i] in notokey:

                        dflag = False

                    if board\_list[cy][cx - i] in oklist and aflag:

                        goodmoves.append('a')

                        aflag = False

                    elif board\_list[cy][cx - i] in notokey:

                        aflag = False

                if (cy + i) <= 21 and (cy - i) >= 0:

                    if board\_list[cy + i][cx] in oklist and sflag:

                        goodmoves.append('s')

                        sflag = False

                    elif board\_list[cy + i][cx] in notokey:

                        sflag = False

                    if board\_list[cy - i][cx] in oklist and wflag:

                        goodmoves.append('w')

                        wflag = False

                    elif board\_list[cy - i][cx] in notokey:

                        wflag = False

        if len(goodmoves) >= 1:

            move = random.randint(0, len(goodmoves) - 1)

*self*.ghostsmoves[g] = goodmoves[move]

*self*.ghost\_move(g)

    else:  *#*

*self*.ghost\_move(g)

*self*.ghosts.draw(*self*.screen)

    pygame.display.update()

Движение призраков производится в коде выше. Чтобы призрак сходил мы должны рассчитать возможность того или иного хода, посредством изменения флагов возможности ходов в различных направлениях, чтобы в последствии случайно выбрать тот или иной ход. В конце выбора хода выполняется функция ghost\_move.

def ghost\_move(self, g):

*# перемещение привидений*

    pygame.draw.rect(*self*.screen, (0, 0, 0), (g.rect.x, g.rect.y,

                                                CELL\_SIZE, CELL\_SIZE), width=0)

    if *self*.ghostsmoves[g] == 'w':

        g.rect.y = g.rect.y - 1

    if *self*.ghostsmoves[g] == 'a':

        g.rect.x = g.rect.x - 1

    if *self*.ghostsmoves[g] == 's':

        g.rect.y = g.rect.y + 1

    if *self*.ghostsmoves[g] == 'd':

        g.rect.x = g.rect.x + 1

*self*.ghosts.draw(*self*.screen)

    pygame.display.flip()

Эта функция выполняет движение приведений посредством попиксельного изменения расположения призраков на поле, в соответствии с выбранным направлением.

На данном этапе призраки и пакман существуют на поле и передвигаются, но все еще не могут сталкиваться друг с другом, а мы не можем проверять столкнулись ли они. Эту проблему решает код ниже:

def collide\_pacman(self, pos):  # просчитываем столкновения с памананом

    # столкновение с пакманом

    self.main\_pacman\_sprite.rect.x = pos[0]

    self.main\_pacman\_sprite.rect.y = pos[1]

    if pygame.sprite.spritecollideany(self.main\_pacman\_sprite, self.ghosts):

        return True

    else:

        return False

Данное условие производит проверку столкнулся ли спрайт пакмана со спрайтом какого-либо приведения. После добавления условия выше мы должны дополнить код цикла событий следующими строками:

while running:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            running = False

        if event.type == pygame.KEYDOWN:

            ...

        if event.type == TICK:

            score = *self*.dot.score\_update()

*self*.ghosts.ghost\_calc()

*self*.dot.update(*self*.pacman.PacmanCurrentPos)

*self*.dot.update\_dots()

*self*.pacman.pacman\_movement(PacmanCurrentKey, *self*.pacman.PacmanCurrentPos[1], *self*.pacman.PacmanCurrentPos[0])

            if *self*.ghosts.collide\_pacman(*self*.pacman.PacmanCurrentPos):

*self*.lose()

            if score == 2050:

*self*.win()

        if event.type == PACMAN\_MOTION:

*self*.pacman.motion\_counting()

    pygame.display.flip()

pygame.quit()

С этого момента мы добавляем ходы для привидений в каждый тик, а также вводим условия выигрыша по очкам и проигрыша при столкновении с пакманом. Сами функции lose и win показаны ниже.

def win(self):

    pygame.time.set\_timer(TICK, 0)

    pygame.time.set\_timer(PACMAN\_MOTION, 0)

    font = pygame.font.Font(None, 50)

    text = font.render("You win!", True, (232, 72, 167))

    text\_x = 112

    text\_y = 150

    place = text.get\_rect(

        center=(237, 275))

    text\_w = text.get\_width()

    text\_h = text.get\_height()

    pygame.draw.rect(*self*.screen, (0, 0, 0), (place[0], place[1],

                                        text\_w, text\_h), 0)

*self*.screen.blit(text, place)

def lose(self):

    pygame.time.set\_timer(TICK, 0)

    pygame.time.set\_timer(PACMAN\_MOTION, 0)

    font = pygame.font.Font(None, 50)

    text = font.render("Game Over", True, (232, 72, 167))

    text\_x = 112

    text\_y = 150

    place = text.get\_rect(

        center=(237, 275))

    text\_w = text.get\_width()

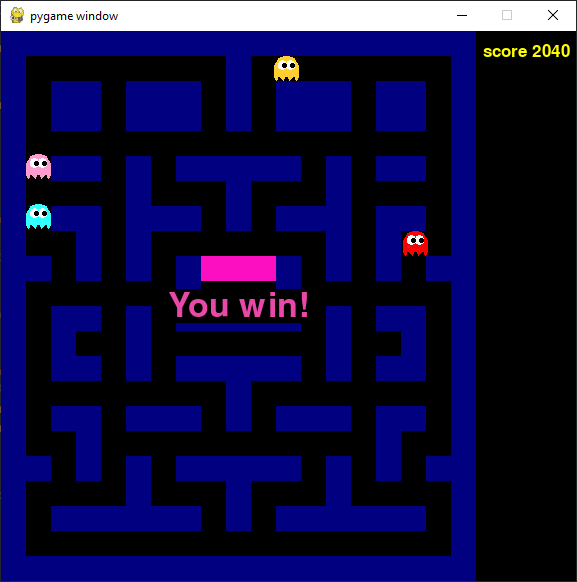
    text\_h = text.get\_height()

    pygame.draw.rect(*self*.screen, (0, 0, 0), (place[0], place[1],

                                        text\_w, text\_h), 0)

*self*.screen.blit(text, place)

Данные функции являются функциями рендера и делаеют ничего кроме вывода пользователю следующих сообщений.





# Заключение

В данном проекте мы достигли поставленных целей, а именно:

* Изучили библиотеку Pygame, а также ее возможности и функционал, а все знания оформили в мини-курс о котором пойдет речь ниже
* Написали мини курс с основами по pygame, в котором рассмотрели:
  + Инициализацию библиотеки
  + Игровой цикл событий
  + Возможности рендера на экране различных фигур
  + Создание спрайтов и анимаций
  + Написание пакмана со всеми навыками, изложенными выше

Данный курс мы оформили в markdown файлах, а сам курс расположен на гитхабе. Курс и все файлы проекта можно просмотреть по ссылке:

<https://github.com/AZOca/Pygame-course>

* Мы написали игру пакман, с подробным объяснением в отчете всего функционала и методов, используемых в коде, а также назначения всех функций в коде.