МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

Московский технический университет связи и информатики

**Проект**

**По дисциплине “Python-программирование”**

На тему: “Написание курса по Pygame на примере создания Pacman”

Выполнили ст. гр. БКК2202

Проверил: Кузин А. А.

Москва, 2024

# 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.

# Теоретические материалы

# 3.1. 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()

# 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

**3.3 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.

# Написание Пакмана

Игру Пакман можно функционально разделить на несколько частей:

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

# Инициализация Pygame и переменных

В самом начале мы должны инициализировать библиотеки и переменные размера окна и счетчика событий.

import pygame

pygame.init()

WINDOW\_SIZE = WIDTH, HEIGHT = 575, 550  # размер поля (19, 22), размер клетки 25

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

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

# Доска и Пакман

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 создаю класс пакмана, в дальнейшем в этом классе будут функции, которые будут участвовать в логике работы пакмана. В данном классе мы определяем множество кнопок управления, начальную позицию, спрайты самого пакмана, а также счетчик очков.

class Pacman(pygame.sprite.Sprite):

    # основной класс, где прописан рендер поля и логика движения пакмана

    def \_\_init\_\_(self, screen):

        super().\_\_init\_\_()

        self.retset = set()  # множество для записи допустимых кнопок

        self.width = 19  # ширина поля

        self.height = 22  # высота поля

        self.screen = screen  # поверхность, на которой все выводим

        self.y = 12  # координаты пакмана во вложенном списке

        self.x = 9  # нужен для метода create\_pacman в классе Pacman

        self.PacmanCurrentPos = (225, 300)

        # сохраняет позицию пакмана, (225, 300) - позиция в начале игры в пикселях

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

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

        self.main\_pacman\_sprite.image = pygame.image.load('data/pacmanleft.png')

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

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

        # спрайт пакмана

        pygame.display.flip()

        self.currentkey = 0

        # для проверки возможности хода

        self.count = 0  # счетчик для смены спарйта пакмана

        self.left = 0  # отступ с левого верхнего края по оси x

        self.top = 0  # отступ с левого верхнего края по оси y

        self.cell\_size = 25  # размер клетки в пикселях

# Рендер и движения

При создании пакмана мы должны отрисовывать несколько событий:

1. Первоначальную отрисовку доски
2. Движение пакмана и приведений
3. Съедение пакманом фишек – обновление доски

Для первоначальной отрисовки поля мы проходимся вложенным циклом по списку board и заполняем поле попиксельно квадратами разных цветов:

* + Синего для обозначения стен.
  + Черного для путей
  + Розовым для «дома» привидений

def render(self):

# рендер поля

for x in range(self.width):

for y in range(self.height):

if self.board[y][x] == 0 or self.board[y][x] == 3:

pygame.draw.rect(self.screen, (0, 0, 0), (x \* self.cell\_size + self.left,

y \* self.cell\_size + self.top,

self.cell\_size, self.cell\_size), width=0)

if self.board[y][x] == 1:

pygame.draw.rect(self.screen, (0, 0, 128), (x \* self.cell\_size + self.left,

y \* self.cell\_size + self.top,

self.cell\_size, self.cell\_size), width=0)

if self.board[y][x] == 2:

pygame.draw.rect(self.screen, (252, 15, 192), (x \* self.cell\_size + self.left,

y \* self.cell\_size + self.top,

self.cell\_size, self.cell\_size), width=0)

Для отслеживания передвижений Пакмана (отслеживания нажатий WASD) мы добавляем обработку нажатий каждой кнопки исходя из ее кода на клавиатуре

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

y = (cy - self.top) // self.cell\_size

x = (cx - self.left) // self.cell\_size

self.retset = set()

horkeycheck = (key == 97 or key == 100)

verkeycheck = (key == 119 or key == 115)

# вертикальные и горизонтальные ходы

ycellcheck = (cy - self.top) % self.cell\_size == 0

xcellcheck = (cx - self.top) % self.cell\_size == 0

# координаты клеток

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

if (self.board[y][(cx - self.left + 26) // self.cell\_size] != 1 and

self.board[y][(cx - self.left + 26) // self.cell\_size] != 2):

self.retset.add(100)

if (self.board[(cy - self.left + 26) // self.cell\_size][x] != 1 and

self.board[(cy - self.left + 26) // self.cell\_size][x] != 2):

self.retset.add(115)

if (self.board[y][(cx - self.left - 1) // self.cell\_size] != 1 and

self.board[y][(cx - self.left - 1) // self.cell\_size] != 2):

self.retset.add(97)

if (self.board[(cy - self.left - 1) // self.cell\_size][x] != 1 and

self.board[(cy - self.left - 1) // self.cell\_size][x] != 2):

self.retset.add(119)

if key in self.retset: # проверяем допустим ли наш ход WASD

self.currentkey = key

pacman.pacman\_move(key)

else:

pacman.pacman\_move(self.currentkey)