**Использование класса Sprite библиотеки Pygame при изучении основ объектно – ориентированного программирования**

Pygame — это «игровая библиотека», набор инструментов, помогающих программистам создавать игры. В играх используются множество подвижных объектов, с которыми взаимодействует пользователь. Для того чтобы использовать несколько объектов в игре используется класс Sprite.

Спрайт — это элемент компьютерной графики, представляющий объект на экране, который может двигаться. В двухмерной игре все, что вы видите на экране, является спрайтами. Спрайты можно анимировать, заставлять их взаимодействовать между собой или передавать управление ими игроку.

Для загрузки и отрисовки спрайтов нужно их добавить в игровой цикл. Несложно представить, что если в игре много спрайтов, то цикл довольно быстро станет большим и запутанным. В Pygame для этого есть решение: группировка спрайтов.

Набор спрайтов — это коллекция спрайтов, которые могут отображаться одновременно.

Рассмотрим пример игры «Стрелялка» где космический корабль будет лететь через метеориты и стрелять по ним освобождая себе путь. Для этого нам понадобятся объекты «Космический корабль», «Метеорит» и «Лазерный луч». Поэтому мы создадим 3 класса с названиями «Player», «Mob», «Bullet» соответсвенно.

 class Player(pygame.sprite.Sprite):

Здесь определяется новый объект, который будет спрайтом игрока. Его тип pygame.sprite.Sprite. Это значит, что он является подклассом класса Sprite.

 У каждого спрайта в Pygame обязательно должно быть два свойства: image и rect. Свойство image хранит картинку спрайта, свойство rect используется для размещения спрайта в любом месте а так же определения его координат. Нам нужно через метод  \_\_init\_\_() инициализировать эти свойства.

class Player(pygame.sprite.Sprite):
 def \_\_init\_\_(self):
 pygame.sprite.Sprite.\_\_init\_\_(self)
 self.image = pygame.transform.scale(player\_img, (50, 38))
 self.image.set\_colorkey(BLACK)
 self.rect = self.image.get\_rect()
 self.rect.centerx = WIDTH / 2
 self.rect.bottom = HEIGHT - 10
 self.speedx = 0

Команда get\_rect() высчитывает прямоугольную область, которую занимает image.

Так же используем свойство speedx для вычисления координат спрайта по оси Х.

После определения спрайта Player, нужно создать экземпляр класса Player. Также нужно обязательно добавить спрайт в группу all\_sprites.

all\_sprites = pygame.sprite.Group()

player = Player()

all\_sprites.add(player)

В игровом цикле используется функция all\_sprites.update() для движения спрайтов. Это значит, что для каждого спрайта в группе Pygame ищет функцию update() и запускает ее. Поэтому необходимо переопределить метод update(), который является методом суперкласса Sprite. Чтобы спрайт двигался при нажатии стрелок, нужно проверить, какая стрелка нажата, а после менять координаты объекта. Когда объект достигнет границ окна, останавливаем его.

def update(self):
 self.speedx = 0
 keystate = pygame.key.get\_pressed()
 if keystate[pygame.K\_LEFT]:
 self.speedx = -8
 if keystate[pygame.K\_RIGHT]:
 self.speedx = 8
 self.rect.x += self.speedx
 if self.rect.right > WIDTH:
 self.rect.right = WIDTH
 if self.rect.left < 0:
 self.rect.left = 0

Метод update() спрайта запускается в каждом кадре. Он будет перемещать спрайт с конкретной скоростью.

Точно так же создаем класс для метеоритов «Mob».

class Mob(pygame.sprite.Sprite):

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

 pygame.sprite.Sprite.\_\_init\_\_(self)

 self.image = meteor\_img

 self.image.set\_colorkey(BLACK)

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

 self.rect.x = random.randrange(WIDTH - self.rect.width)

 self.rect.y = random.randrange(-100, -40)

 self.speedy = random.randrange(1, 8)

 self.speedx = random.randrange(-3, 3)

 self.radius = int(self.rect.width \* .85 / 2)

 def update(self):

 self.rect.x += self.speedx

 self.rect.y += self.speedy

 if self.rect.top > HEIGHT + 10 or self.rect.left < -25 or self.rect.right > WIDTH + 20:

 self.rect.x = random.randrange(WIDTH - self.rect.width)

 self.rect.y = random.randrange(-100, -40)

 self.speedy = random.randrange(1, 8)

В классе «Mob» мы берем случайные значения для переменных self.rect.x, self.rect.y, self.speedy, self.speedx. Чтобы наши метеориты появлялись из различных точек и имели различные скорости. Так же добавили переменную self.radius для определения радиуса метеорита для определения столкновении с лазером.

Переопределяем метод update() для класса Mob, чтобы задать скорость движения метеоритов. Если метеориты уходят за пределы окна меняем их координаты х и у.

Так же создаем класс «Bullet» для отображения лазерного луча.

class Bullet(pygame.sprite.Sprite):

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

 pygame.sprite.Sprite.\_\_init\_\_(self)

 self.image = bullet\_img

 self.image.set\_colorkey(BLACK)

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

 self.rect.bottom = y

 self.rect.centerx = x

 self.speedy = -10

 def update(self):

 self.rect.y += self.speedy

 # убить, если он заходит за верхнюю часть экрана

 if self.rect.bottom < 0:

 self.kill()

Лазеры как и корабль находятся внизу, поэтому координаты х и у будут в самом низу.

self.rect.bottom = y

self.rect.centerx = x

Так как движение лазеров идет с низу в верх то и значение скорости будет уменьшаться self.speedy = -10. В методе update() создается движение по координате у. Если лазер уходит за пределы верхней границы экрана удаляем его.

Таким образом используя возможности библиотеки Pygame мы можем показать принципы ООП: наследования от суперкласса Sprite и полиморфизм метода update().

Полный код игры:

import pygame

import random

from os import path

img\_dir = path.join(path.dirname(\_\_file\_\_), 'img')

WIDTH = 480

HEIGHT = 600

FPS = 60

# Задаем цвета

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLUE = (0, 0, 255)

YELLOW = (255, 255, 0)

# Создаем игру и окно

pygame.init()

pygame.mixer.init()

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Shmup!")

clock = pygame.time.Clock()

font\_name = pygame.font.match\_font('arial')

#вывод текста в начальной странице

def draw\_text(surf, text, size, x, y):

 font = pygame.font.Font(font\_name, size)

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

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

 text\_rect.midtop = (x, y)

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

# метод создания метеоритов

def newmob():

 m = Mob()

 all\_sprites.add(m)

 mobs.add(m)

# метод вывода начального экрана

def show\_go\_screen():

 screen.blit(background, background\_rect)

 draw\_text(screen, "Стрелялка!", 64, WIDTH / 2, HEIGHT / 4)

 draw\_text(screen, "Управление стрелками, Стрелять - пробел", 22,

 WIDTH / 2, HEIGHT / 2)

 draw\_text(screen, "нажмите любую клавишу для начала игры", 18, WIDTH / 2, HEIGHT \* 3 / 4)

 pygame.display.flip()

 waiting = True

 while waiting:

 clock.tick(FPS)

 for event in pygame.event.get():

 if event.type == pygame.QUIT:

 pygame.quit()

 if event.type == pygame.KEYUP:

 waiting = False

# метод вывода экрана об окончании игры

def show\_end\_screen():

 screen.blit(background, background\_rect)

 draw\_text(screen, "Игра окончена!", 64, WIDTH / 2, HEIGHT / 4)

 draw\_text(screen, f"Ваш счет {score}", 22,

 WIDTH / 2, HEIGHT / 2)

 draw\_text(screen, "нажмите любую клавишу для начала игры", 18, WIDTH / 2, HEIGHT \* 3 / 4)

 pygame.display.flip()

 waiting = True

 while waiting:

 clock.tick(FPS)

 for event in pygame.event.get():

 if event.type == pygame.QUIT:

 pygame.quit()

 if event.type == pygame.KEYUP:

 waiting = False

class Player(pygame.sprite.Sprite):

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

 pygame.sprite.Sprite.\_\_init\_\_(self)

 self.image = pygame.transform.scale(player\_img, (50, 38))

 self.image.set\_colorkey(BLACK)

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

 self.rect.centerx = WIDTH / 2

 self.rect.bottom = HEIGHT - 10

 self.speedx = 0

 def update(self):

 self.speedx = 0

 keystate = pygame.key.get\_pressed()

 if keystate[pygame.K\_LEFT]:

 self.speedx = -8

 if keystate[pygame.K\_RIGHT]:

 self.speedx = 8

 self.rect.x += self.speedx

 if self.rect.right > WIDTH:

 self.rect.right = WIDTH

 if self.rect.left < 0:

 self.rect.left = 0

 def shoot(self):

 bullet = Bullet(self.rect.centerx, self.rect.top)

 all\_sprites.add(bullet)

 bullets.add(bullet)

class Mob(pygame.sprite.Sprite):

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

 pygame.sprite.Sprite.\_\_init\_\_(self)

 self.image = meteor\_img

 self.image.set\_colorkey(BLACK)

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

 self.rect.x = random.randrange(WIDTH - self.rect.width)

 self.rect.y = random.randrange(-100, -40)

 self.speedy = random.randrange(1, 8)

 self.speedx = random.randrange(-3, 3)

 self.radius = int(self.rect.width \* .85 / 2)

 def update(self):

 self.rect.x += self.speedx

 self.rect.y += self.speedy

 if self.rect.top > HEIGHT + 10 or self.rect.left < -25 or self.rect.right > WIDTH + 20:

 self.rect.x = random.randrange(WIDTH - self.rect.width)

 self.rect.y = random.randrange(-100, -40)

 self.speedy = random.randrange(1, 8)

class Bullet(pygame.sprite.Sprite):

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

 pygame.sprite.Sprite.\_\_init\_\_(self)

 self.image = bullet\_img

 self.image.set\_colorkey(BLACK)

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

 self.rect.bottom = y

 self.rect.centerx = x

 self.speedy = -10

 def update(self):

 self.rect.y += self.speedy

 # убить, если он заходит за верхнюю часть экрана

 if self.rect.bottom < 0:

 self.kill()

# Загрузка всей игровой графики

background = pygame.image.load(path.join(img\_dir, "look.com.ua-98569.jpg")).convert()

background\_rect = background.get\_rect()

player\_img = pygame.image.load(path.join(img\_dir, "playerShip3\_blue.png")).convert()

meteor\_img = pygame.image.load(path.join(img\_dir, "meteorGrey\_med1.png ")).convert()

bullet\_img = pygame.image.load(path.join(img\_dir, "laserBlue16.png")).convert()

# Цикл игры

running = True

game\_over = True

show\_go\_screen()

while running:

 if game\_over:

 game\_over = False

 all\_sprites = pygame.sprite.Group()

 mobs = pygame.sprite.Group()

 bullets = pygame.sprite.Group()

 powerups = pygame.sprite.Group()

 player = Player()

 all\_sprites.add(player)

 for i in range(8):

 newmob()

 score = 0

 # Держим цикл на правильной скорости

 clock.tick(FPS)

 # Ввод процесса (события)

 for event in pygame.event.get():

 # проверка для закрытия окна

 if event.type == pygame.QUIT:

 running = False

 elif event.type == pygame.KEYDOWN:

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

 player.shoot()

 # Обновление

 all\_sprites.update()

 # проверка, попала ли пуля в моб

 hits = pygame.sprite.groupcollide(mobs, bullets, True, True)

 for hit in hits:

 score += 50 - hit.radius

 m = Mob()

 all\_sprites.add(m)

 mobs.add(m)

 # Проверка, не ударил ли моб игрока

 hits = pygame.sprite.spritecollide(player, mobs, False)

 if hits:

 game\_over = True

 show\_end\_screen()

 # Рендеринг

 screen.fill(BLACK)

 screen.blit(background, background\_rect)

 all\_sprites.draw(screen)

 draw\_text(screen, str(score), 18, WIDTH / 2, 10)

 # После отрисовки всего, переворачиваем экран

 pygame.display.flip()

pygame.quit()