



Software
Engineering
School

Курсы программирования для
взрослых и детей

Делаем игры на Rugame

Чем мы сегодня займемся

- Познакомимся с понятием спрайта;
- Узнаем, как в Pygame загрузить картинки;
- Рассмотрим, как работать с клавиатурой;
- Нарисуем танк и научим его двигаться

Наша задача до конца модуля

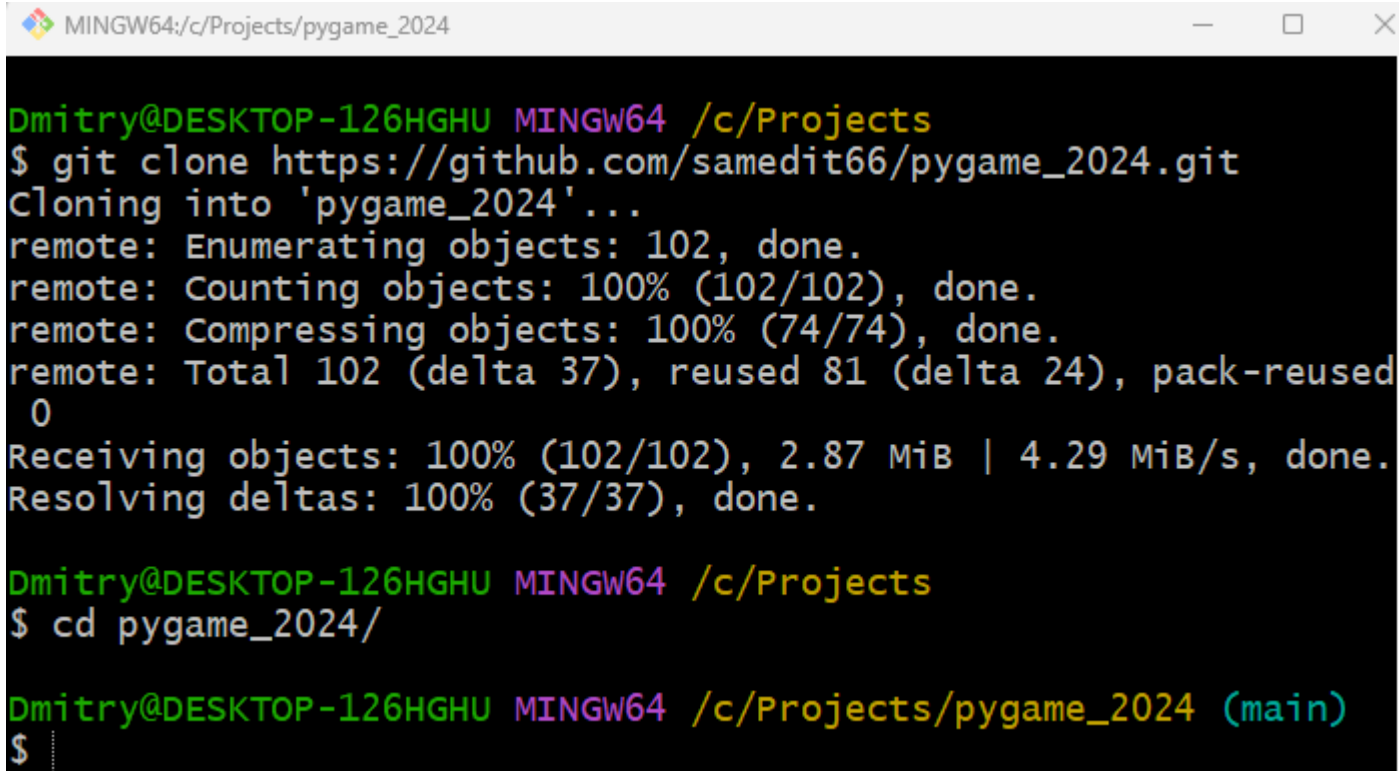
Разработать 2D танковый шутер



Прежде чем мы начнем

1. Переходим на свою почту и принимаем от меня приглашение
2. Заходим на гитхаб в наш репозиторий
https://github.com/samedit66/pygame_2024
3. Жмем зеленую кнопку Code и копируем ссылку
4. Открываем git bash, выбираем любую папку и клонируем туда репозиторий
5. Переходим в папку репозитория

Прежде чем мы начнем



A terminal window titled 'MINGW64:/c/Projects/pygame_2024' with standard window controls. The terminal shows the execution of 'git clone' and 'cd' commands. The output of 'git clone' shows the cloning progress, including object enumeration, counting, and compression. The 'cd' command successfully changes the directory to 'pygame_2024/'. The prompt then changes to reflect the new directory path.

```
MINGW64:/c/Projects/pygame_2024
Dmitry@DESKTOP-126HGHU MINGW64 /c/Projects
$ git clone https://github.com/samedit66/pygame_2024.git
Cloning into 'pygame_2024'...
remote: Enumerating objects: 102, done.
remote: Counting objects: 100% (102/102), done.
remote: Compressing objects: 100% (74/74), done.
remote: Total 102 (delta 37), reused 81 (delta 24), pack-reused
0
Receiving objects: 100% (102/102), 2.87 MiB | 4.29 MiB/s, done.
Resolving deltas: 100% (37/37), done.

Dmitry@DESKTOP-126HGHU MINGW64 /c/Projects
$ cd pygame_2024/

Dmitry@DESKTOP-126HGHU MINGW64 /c/Projects/pygame_2024 (main)
$
```

Что такое спрайт?

Спрайт – элемент компьютерной графики, представляющий объект на экране, который умеет двигаться. Их можно анимировать, заставлять взаимодействовать между собой, или передавать управление ими игроку. В нашей игре объект Танк это спрайт. Таким образом, спрайт - это любой объект, который умеет двигаться во время игры.

Что такое спрайт?

Чтобы использовать спрайты в Pygame, нам необходимо создать класс нашего спрайта и отнаследовать его от класса `pygame.sprite.Sprite` – базового класса для всех спрайтов в игре.

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
    def __init__(self):  
        pygame.sprite.Sprite.__init__(self) ←
```

Первой строчкой в конструкторе нашего спрайта всегда идет вызов родительского конструктора!!!

Добавляем картинки

Чтобы спрайт был виден на экране, ему нужна картинка. Для этого воспользуемся функцией `pygame.image.load()`, которая принимает имя файла с картинкой для спрайта:

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
    def __init__(self, image_file):  
        pygame.sprite.Sprite.__init__(self)  
        self.image = pygame.image.load(image_file).convert_alpha()
```

Функция `convert_alpha()` нужна, чтобы ускорить загрузку картинки танка в дальнейшем

Добавляем картинки

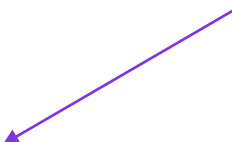
Помимо картинки спрайту необходимо знать прямоугольник, ограничивающий его на экране, чтобы отрисовать себя на экране. Для этого воспользуемся функций `get_rect()` возвращающей ограничивающий прямоугольник

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
    def __init__(self, image_file, position):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(image_file).convert_alpha()
        self.rect = self.image.get_rect(topleft=position)
```

Передаем
дополнительный аргумент
– координаты левого
верхнего угла
прямоугольника



Отображаем танк на экране

Сделаем по аналогии с игровым циклом метод `render()` отображающий танк на экране

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
    def __init__(self, image_file, position):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(image_file).convert_alpha()
        self.rect = self.image.get_rect(topleft=position)
```

```
    def render(self, window):
        window.blit(self.image, self.rect) ←
```

Чтобы нарисовать на главном окне, мы должны вызывать функцию `blit()` у окна – передать ей картинку и прямоугольник, в котором она будет отрисована

Игровой цикл

```
class Game():  
    def __init__(self):  
        pygame.init()  
  
        self.WINDOW_WIDTH = 800  
        self.WINDOW_HEIGHT = 800  
        self.FPS = 60  
  
        self.main_window = pygame.display.set_mode((self.WINDOW_WIDTH,  
                                                    self.WINDOW_HEIGHT))  
  
        self.clock = pygame.time.Clock()  
        self.running = True  
        self.tank = Tank("tanks_images/blue_tank.png", (0, 0))
```


Игровой цикл

```
class Game():
```

```
...
```

```
def process_input(self):  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            self.running = False  
        return
```


Добавим обработку выхода
по нажатию на крестик,
чтобы приложение не
зависало



```
def update_game_state(self):  
    pass
```

```
def render(self):  
    self.main_window.fill(pygame.color.THECOLORS["white"])  
    self.tank.render(self.main_window)  
    pygame.display.update()
```

Говорим танку отрисовать
себя, и передаем ему
главное окно – место, где
он себя отрисует



Игровой цикл

```
class Game:
```

```
...
```

```
def main_loop(self):  
    while self.running:  
        self.process_input()  
        self.update_game_state()  
        self.render()  
        self.clock.tick(self.FPS)  
pygame.quit()
```

```
game = Game()  
game.game_loop()
```

Обработка нажатий клавиш

Для обработки нажатий клавиш можно использовать стандартный способ – получение всех событий от пользователя через `pygame.event.get()`, однако такой способ нам не совсем подходит – нажав на клавишу и держа её определенное время, мы хотим, чтобы танк двигался, пока кнопка зажата, но так сделать нельзя, если мы используем `pygame.event.get()`. Нам нужен специальный модуль для работы с клавиатурой – `pygame.key` и его функция `pygame.key.get_pressed()`


Обработка нажатий клавиш

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
    def __init__(self, image_file, position):  
        pygame.sprite.Sprite.__init__(self)  
        self.image = pygame.image.load(image_file).convert_alpha()  
        self.rect = self.image.get_rect(topleft=position)  
        self.move_x = 0  
        self.move_y = 0
```

Насколько
сдвигается танк при
нажатии на клавиши



```
    def render(self, window):  
        window.blit(self.image, self.rect)
```

Обработка нажатий клавиш

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
...
```

```
def process_input(self):
```

```
    self.move_x = self.move_y = 0
```

```
    key = pygame.key.get_pressed()
```

```
    if key[pygame.K_LEFT] or key[pygame.K_a]:
```

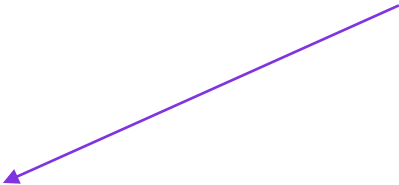
```
        self.move_x = 2
```

```
    elif key[pygame.K_UP] or key[pygame.K_w]:
```

```
        self.move_y = -2
```

```
    # код для обработки нажатий вниз и вправо, самостоятельно!
```

Всегда обнуляем
сдвиги, т.к.
обработка ввода
происходит
каждый кадр,, если
мы ничего не
сделали, сдвиги
должны быть
нулевые



Обработка нажатий клавиш

Функция `pygame.key.get_pressed()` возвращает список всех клавиш клавиатуры – они пронумерованы по индексам, т.е. эта функция возвращает список вида

`[True, False, True, False, False ...]`

И если сопоставить его со списком клавиш

`[W, A, S, D, Q...]`


То обратившись по номеру клавиши (например, пусть W – это нулевой индекс), мы получим логический флаг, нажата ли клавиша или нет: `if key[pygame.K_w] == True` если нажата клавиша W

Игровой цикл

Добавим обработку нажатия клавиш в класс Game:

```
class Game():  
  
    ...  
  
    def process_input(self):  
        for event in pygame.event.get():  
            if event.type == pygame.QUIT:  
                self.running = False  
                return  
        self.tank.process_input()
```

Все что нужно – добавить
вызов функции обработки
ввода у танка!



Обработка нажатий клавиш

Кажется, мы забыли добавить обновление состояние танка – сдвиг его координат, самое время это сделать

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
...
```

```
def update(self):  
    self.rect.x += self.move_x  
    self.rect.y += self.move_y
```

Игровой цикл

Не забудем также обновить класс Game:

```
class Game():
```

```
...
```

```
def update_game_state(self):  
    self.tank.update()
```

← Все что нужно – добавить
вызов функции обновления
состояния у танка!

Учим танк поворачиваться

Пусть, когда танк поворачивается, его картинка также поворачивается. Для этого нам понадобится функция `pygame.transform.rotate()`, которая принимает картинку и угол её поворота

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
    def __init__(self, image_file, position):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(image_file).convert_alpha()
        self.up_image = self.image
        self.left_image = pygame.transform.rotate(self.image, 90)
        self.rect = self.image.get_rect(topleft=position)
        self.move_x = 0
        self.move_y = 0
```

Изначально танк «смотрит» вверх (а точнее - его картинка), поэтому мы поворачиваем исходное изображение танка на 90 градусов влево, чтобы получить картинку танка, смотрящего влево

Учим танк поворачиваться

```
import pygame
```

```
class Tank(pygame.sprite.Sprite):
```

```
...
```

```
def process_input(self):
```

```
    self.move_x = self.move_y = 0
```

```
    key = pygame.key.get_pressed()
```

```
    if key[pygame.K_LEFT] or key[pygame.K_a]:
```

```
        self.move_x = 2
```

```
        self.image = self.left_image
```

```
    elif key[pygame.K_UP] or key[pygame.K_w]:
```

```
        self.move_y = -2
```

```
        self.image = self.up_image
```

```
    # код для обработки нажатий вниз и вправо, самостоятельно!
```

Задаем в качестве изображения для отрисовки в `render()` картинку смотрящего влево танка

И делаем точно также, когда танк смотрит вверх

Домашнее задание

Добавьте обработку случаев, когда танк смотрит вниз и вправо – по нажатию на WASD или стрелочки, танк должен не только двигаться, но и поворачиваться в эту же сторону

Наш репозиторий

https://github.com/samedit66/pygame_2024/tree/main