PYGAME.

ЦЕЛЬ: Изучить принцип создания анимации в программировании игр.

ПРИМЕЧАНИЕ:

ПЛАНИРОВАНИЕ

1. Анимация

Уже многие знакомы с тем, как программируется анимация, но проговорим еще раз:

Анимация – это быстрая смена кадров одного изображения спрайта, в процессе чего создается реалистичность различных движений персонажа (спрайта) и его внешнего облика.

Как создавать в pygame изображения, мы знаем, нужно лишь запрограммировать быструю смену кадров.

Логично будет кадры одного спрайта отвечающие за одно движение хранить в специально созданном для этого списке (массиве).

Например, вот так:

ImgPlayerGo = [pygame.image.load(“Player1.png”), pygame.image.load(“Player1.png”), pygame.image.load(“Player1.png”), pygame.image.load(“Player1.png”), pygame.image.load(“Player1.png”)]

Мы создали список, элементами которого являются изображения-кадры.

Но, конечно, эти изображения должны быть заранее созданы и храниться в необходимой директории.

После этого нам понадобиться переменная, в которой мы будем хранить номер кадра.

Ncadr = 0

А дальше напрашивается в нашем игровом цикле **while** прописать следующий код:

```
while game:
        ……..


        w.fill((0, 0, 0))
        w.blit( ImgPlayerGo[ Ncadr ] , (Player.x, Player.y))     # выводим на экран элемент списка № Ncadr
        pygame.display.update()


        Ncadr += 1     # увеличиваем номер кадры на 1, чтобы в следующий раз вывелся следующий кадр
```

! Проблемная задача: найти ошибку в новых строчка кода.

Ошибка заключается в том, что в тот момент, когда переменная **Ncadr** достигнет значения 5, при выполнении команды:

w.blit(ImgPlayerGo[Ncadr] , (Player.x, Player.y))

произойдет ошибка, так как в списке **ImgPlayerGo** только 5 элементов и последний элемент имеет номер 4 (так как нумерация начинается с нуля).

Как обойти эту проблему?

Есть красивый математический способ. В Python и в других ЯП есть такая команда - **%**, которая обозначает остаток от деления.

Если мы запишем вместо:

w.blit(ImgPlayerGo[Ncadr] , (Player.x, Player.y))

вот так:

w.blit(ImgPlayerGo [Ncadr % 5] , (Player.x, Player.y))

То выражение **Ncadr % 5** , которое отвечает за номер элемента в списке с кадрами, будет последовательно принимать значения от 0 до 4, соответственно в таком же порядке будут изменяться изображения спрайта.

Все теперь анимация работает, но уж слишком быстро. Чтобы все было синхронизировано, нам необходимо каким-то образом определить сколько кадров в секунду будет выдавать наша игра, и в этом нам поможет модуль **time.**

А перед тем, как познакомимся с модулем time, письменная задачка:

Произвести вручную отладку* следующего кода:

```
N = 0
List = ['A', 'B', 'C', 'D']
While N < 21:
        print(List[N % 4]
        N += 1
```

*Отладка вручную, это когда берется лист бумаги, и представляется, что лист бумаги – это оперативная память, с которой работает наша программа, то есть в ней создаются все ячейки памяти – переменные, списки и другие объекты, и данные.
На листочек выписываются названия всех переменных и списков из программы.
Затем мы сами вручную построчно выполняем программу, за место компьютера и на листке отображаем значения переменных после каждой команды.

2. Модуль time и класс Clock

В библиотеке pygame есть такой модуль – **time,** он предназначен для отслеживания времени в игре.

В этом модуле основным объектом является объект – **Clock**. С английского переводится как часы. А у объекта – Clock есть метод – **tick(framerate),** где **framerate –** скорость, количество кадров в секунду. Иначе говоря, если в программе вызывается метод – **tick(50)** и он находится внутри цикла, то этот цикл будет выполняться 50 раз в секунду, либо меньше (в зависимости от мощности компьютера и сложности программы).

Поэтому, создадим объект класса **Clock**:

clock = pygame.time.Clock()

и в игровом цикле пропишем команду:

clock.tick(24)

вместо 24 можно взять любое другое количество кадров в секунду.

Теперь скорость анимации стала нормальной.

! Проблемная задача: сделать так, чтобы анимация была только во время движения и, чтобы спрайт смотрел в ту сторону, которую бежит.

Решение:

```
# animation for go to right
ImgPlayerGoR = [pygame.image.load('Images/GoAnim/b1.png'),
                pygame.image.load('Images/GoAnim/b2.png'),
                pygame.image.load('Images/GoAnim/b3.png'),
                pygame.image.load('Images/GoAnim/b4.png'),
                pygame.image.load('Images/GoAnim/b5.png')]


# animation for go to left
ImgPlayerGoL = []
for img in ImgPlayerGoR:
    ImgPlayerGoL.append(pygame.transform.flip(img, True, False))
```

В игровом цикле:

```
while game:
    for ev in pygame.event.get ():
        if ev.type == pygame.QUIT:
            game = False
    keys = pygame.key.get_pressed()
    if keys[pygame.K_RIGHT]:
        Player.x += Player.speed
        Player.image = ImgPlayerGoR[Ncadr % 5]
    elif keys[pygame.K_LEFT]:
        Player.x -= Player.speed
        Player.image = ImgPlayerGoL[Ncadr % 5]
```

```
    w.fill((0, 0, 0))
    w.blit(Player.image, (Player.x, Player.y))
    pygame.display.update()
    Ncadr += 1
```

3. Рефлексия

- Сегодня мы научились анимировать спрайта

- Познакомились с объектом – **Clock** и его методом – **tick()**

- Сделали анимацию бега в игре