1. УСТАНОВКА БИБЛИОТЕКИ PYGAME. ЗНАКОМСТВО

ЦЕЛЬ: установить библиотеку pygame. Создать первую программу. Познакомиться с первыми основными объектами библиотеки.

ПРИМЕЧАНИЕ: предполагается, что на компьютерах уже установлена программа – среда разработки Python. В случае, если она не установлена, необходимо установить ее с официального сайта – python.org.

Среду Python можно установить с официального сайта python.org.

ПЛАНИРОВАНИЕ

1. Вводный этап. Установка Pygame.

Чтобы установить библиотеку (модуль) **pygame** для Python, необходимо запустить командную строку и в ней выполнить команду:

python3 pip install pygame

Если эта команда не выполнилась, то в таком случае, необходимо открыть папку, куда установлен **Python**, там найти папку **Scripts**, зайти в эту папку и скопировать весь путь к ней.

Пример пути:

C:\Users\Asus\AppData\Local\Programs\Python\Python37-32\Scripts

Затем открыть командную строку и выполнить команду: **cd ваш путь**

Пример:

cd C:\Users\Asus\AppData\Local\Programs\Python\Python37-32\Scripts

После этой команды вы перейдете в директорию **Scripts.**

Далее выполните команду:

pip install pygame

Запустится установка и по завершению выйдет сообщение:

Successfully installed pygame

Теперь, если запустить новый проект на **python** и выполнить команду: import pygame

Программа выдаст сообщение:

pygame 1.9.6

Hello from the pygame community. **https://www.pygame.org/contribute.html**

Можно поздравить учеников с установкой модуля **Pygame**

2. Первая программа

Итак, настало время познакомиться с первыми командами из библиотеки pygame.

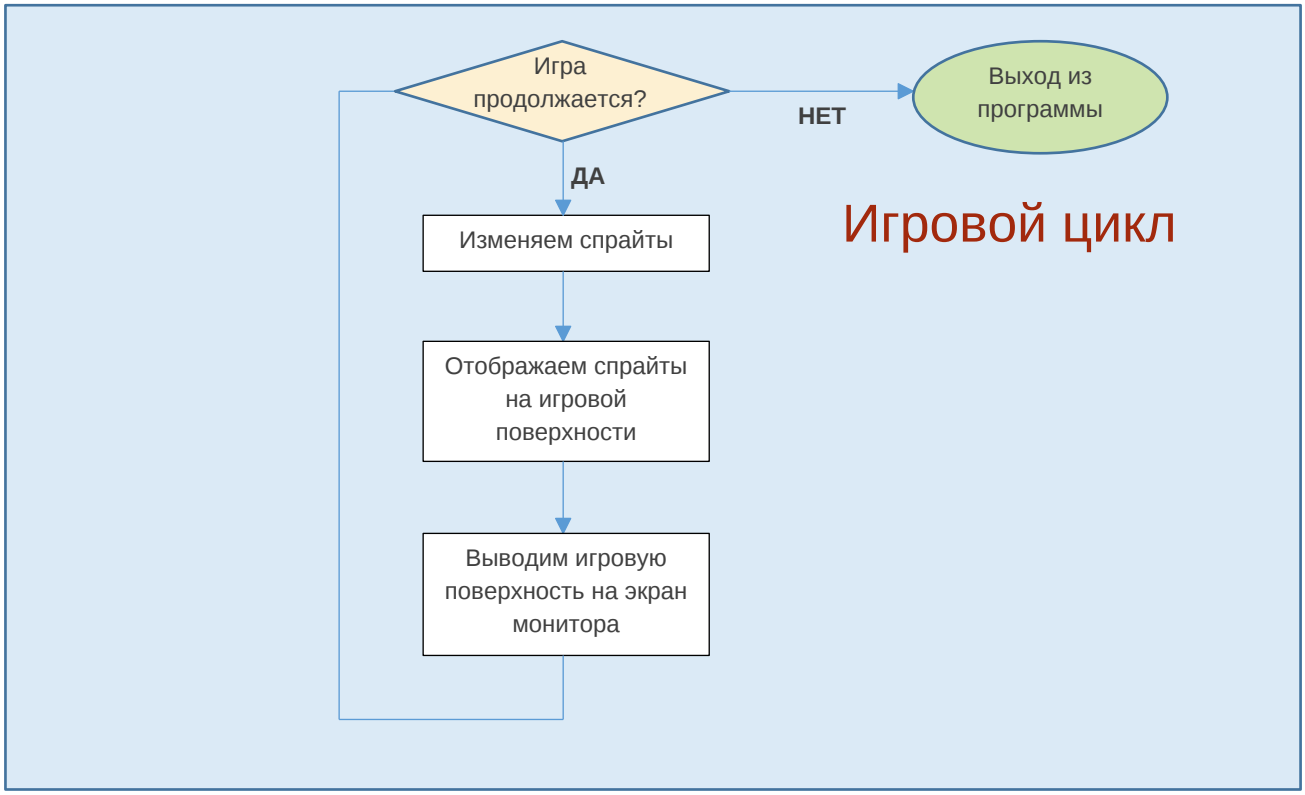
Открываем среду Python и создаем новый проект.

Чтобы подключить библиотеку первая команда в программе должна быть:

import pygame

После этой команды мы можем пользоваться всеми инструментами библиотеки.

Принцип создания игр с использованием таких средств, как pygame, заключается в том, что сначала создается поверхность, на которой рисуются объекты игры, а затем создается игровой цикл, внутри которого рисуются кадры игры и перед каждым кадром происходит программная обработка игровых событий, изменение положений спрайтов, изменение самих спрайтов.



Как вы уже знаете, в профессиональном программировании во всем мире используется объектно-ориентированный подход. И поверхность, на которой рисуются игровые объекты – тоже является объектом. В pygame можно создавать хоть сколько поверхностей, но одна поверхность является самой главной и именно она выводиться на экран монитора. Чтобы ее создать, необходимо выполнить метод – set_mode () модуля **display**, внутрь скобок этого метода необходимо передать кортеж из двух целых чисел, являющихся шириной и высотой поверхности.

Пример:

w = pygame.display.set_mode((1280, 700))

Метод set_mode() возвращает саму эту поверхность как объект, поэтому его необходимо присвоить какому либо имени. В примере мы создаем поверхность шириной 1280 пикселей и высотой 700 пикселей. (Далее пиксели будем обозначать – **px**).

Если мы сейчас запустим программу, то увидим, как откроется окно с заданными размерами и черным фоном. А теперь предложим ученикам закрыть его)) Ничего не выйдет, программа как будто бы зависла!

Можно задать вопрос ученикам: как вы думаете почему?

А фишка заключается в том, что **pygame** – это уже серьёзный профессиональный инструмент, в котором все необходимо программировать самостоятельно и даже само закрытие программы!

Для выхода из программы в модуле pygame существует метод – quit(). Допишем ее в программу:

pygame.quit()

Запустим – теперь открывается наше окно и сразу закрывается. Уже не зависает, но хотелось бы, чтобы программа, как нормальная, закрывалась при нажатии на крестик в правом верхнем углу.

Как же это сделать?

В pygame есть инструменты, с помощью которых можно узнавать какие происходят события во время работы программы, и программировать их обработку.

Это модуль – **event.** У него есть такой метод – **get(),** который возвращает список всех событий, произошедших в данный момент.

Мы можем получить этот список:

Listevent = **pygame.event.get ()**

И потом пробежаться по нему циклом **for** и проверить есть ли в нем необходимое нам событие:

```
for ev in Listevent:
    if ev.type == pygame.QUIT:
```

QUIT – это код события нажатия крестика и сочетания клавиш **Alt – f4**

type – свойство объекта-событие

Давайте запрограммируем событие – когда мы нажимаем крестик закрытия окна программы. Но для этого уже потребуется создать **игровой цикл (ИЦ).**

ИЦ – это цикл с условием (while)? Который постоянно выполняется, пока работает программа (идет игра), и внутри которого обрабатываются все игровые события, рисуются спрайты и т.д.

Создадим его и сразу запрограммируем выход из этого цикла если нажат крестик закрытия окна.

Вот какая программа у нас должна получиться:

```
import pygame
w = pygame.display.set_mode ((1279, 700))
game = True
while game:
    for ev in pygame.event.get ():
        if ev.type == pygame.QUIT:
            game = False
pygame.quit ()
```

Итак, что мы сделали, создали логическую переменную – Game. Пока она – True, игра идет, цикл выполняется. Внутри цикла создали еще один цикл – for, который перебирает все элементы в списке, который возвращает метод – get() объекта event. Элементами данного списка являются объекты – события, которые успели произойти за “микровремя” прошедшее с момента предыдущей итерации цикла. Цикл for, перебирая эти события присваивает их по очереди имени – **ev**, а уже внутри цикла мы проверяем тип этих событий:

if ev.type == pygame.QUIT

И если тип события равен QUIT, то меняем переменную game на False и цикл while завершается, выполняется следующая команда после него – pygame.quit(). Программа закрывается.

3. Рефлексия

- Сегодня мы научились устанавливать дополнительные библиотеки.

- Познакомились с библиотекой – pygame.

- Создали нашу первую программу.

- Узнали, что такое игровой цикл.

- Узнали, как получить список событий

- Познакомились с событием QUIT