PYGAME.

ЦЕЛЬ: **Создать в игре цель, события выигрыша и проигрыша.**

ПРИМЕЧАНИЕ:

ПЛАНИРОВАНИЕ

1. Проигрыш.

Проигрыш в любой игре — это либо потеря персонажа, либо недостижение цели за определенное время.

Как вариант сделаем проигрышем – это потеря персонажа. Пускай у него будет определенный уровень жизни и каждый раз, когда на него падает огненный шар, уровень жизни уменьшается на 10%.

Для этого создадим в классе Player свойство – **health** в конструкторе

self.health = 100

Теперь будем уменьшать это свойство на 10 пунктов, когда происходит касание спрайта огненного шара и спрайта игрока. И сразу проверять: если уровень жизни стал ноль или меньше, то менять уровень игры.

```
if collideG(self.rect, GrFireBall):
        self.health -= 10
        if self.health <= 0:
            Stage = 2
```

Переменной, которая отвечает за уровень игры - Stage присваиваем 2, это соответствует событию проигрыша. Поэтому в игровом цикле нам необходимо создать блок, который будет отображать событие проигрыша.

Давайте сделаем так, чтобы в тот момент, когда жизни становятся равными нулю, игра останавливалась и появлялась надпись – **LOOSER**

И через 3 секунды игра переходила на уровень – 0, стартовый экран.

Новая информация

Чтобы в pygame выводить текст на экран, нам понадобиться модуль **font**

В модуле font есть класс – **Font**

Мы можем создать объект этого класса:

font = pygame.font.Font(None, 25)

Это мы создали объект – Шрифт, вместо none можно написать название шрифта, второй параметр – это размер.

И у объекта класса Font есть метод - render(ТЕКСТ',True, (252, 58, 58))

Где первый параметр э то текст, который необходимо вывести, второй определяет сглаживать или нет текст, а третий – это цвет текста в формате rgb.

Этот метод возвращает объект – Surface с изображенным на ней текстом в заданном формате.

И мы можем ее отображать в необходимых координатах на главной поверхности – **w**

Еще сделаем одну фишку, чтобы после проигрыша игра не сразу перепрыгивала на стартовый экран, а через 3 секунды. Игрок должен осознать, что он проиграл, успеть прочитать надпись – LOOSER.

Для этого познакомимся с новым методом - **get_time()** у класса **Clock**

clock.get_time()

Этот метод возвращает количество миллисекунд, прошедших с момента последнего вызова метода – **tick()**

Получается, мы можем создать глобальную переменную – **Timer**

И с помощью операции в игровом цикле

Timer += clock.get_time() мы можем считать, сколько прошло времени с момента какого-либо события.

Естественно, **Timer** необходимо каждый раз обнулять.

Итак добавим в игровой цикл следующую ветку - **elif**

```
elif Stage == 2:
    text = fontLooser.render('LOOSER',True,(252, 58, 58))
    w.blit(text, (int(ScreenWidth/2), int(ScreenHeight/2)))
    Timer += clock.get_time()
    if Timer > 3000:
        Timer = 0
        Stage = 0
```

2. ! Проблемная задача

Реализовать в программе следующее действие:

Когда в персонажа попадает огненный шар, персонаж обездвиживается на 2 секунды и в этот момент у него появляются «звездочки» над головой.

3. Рефлексия

- Сегодня мы создали событие проигрыша

- Познакомились с новым модулем и классом

- Познакомились с методом – get_time()