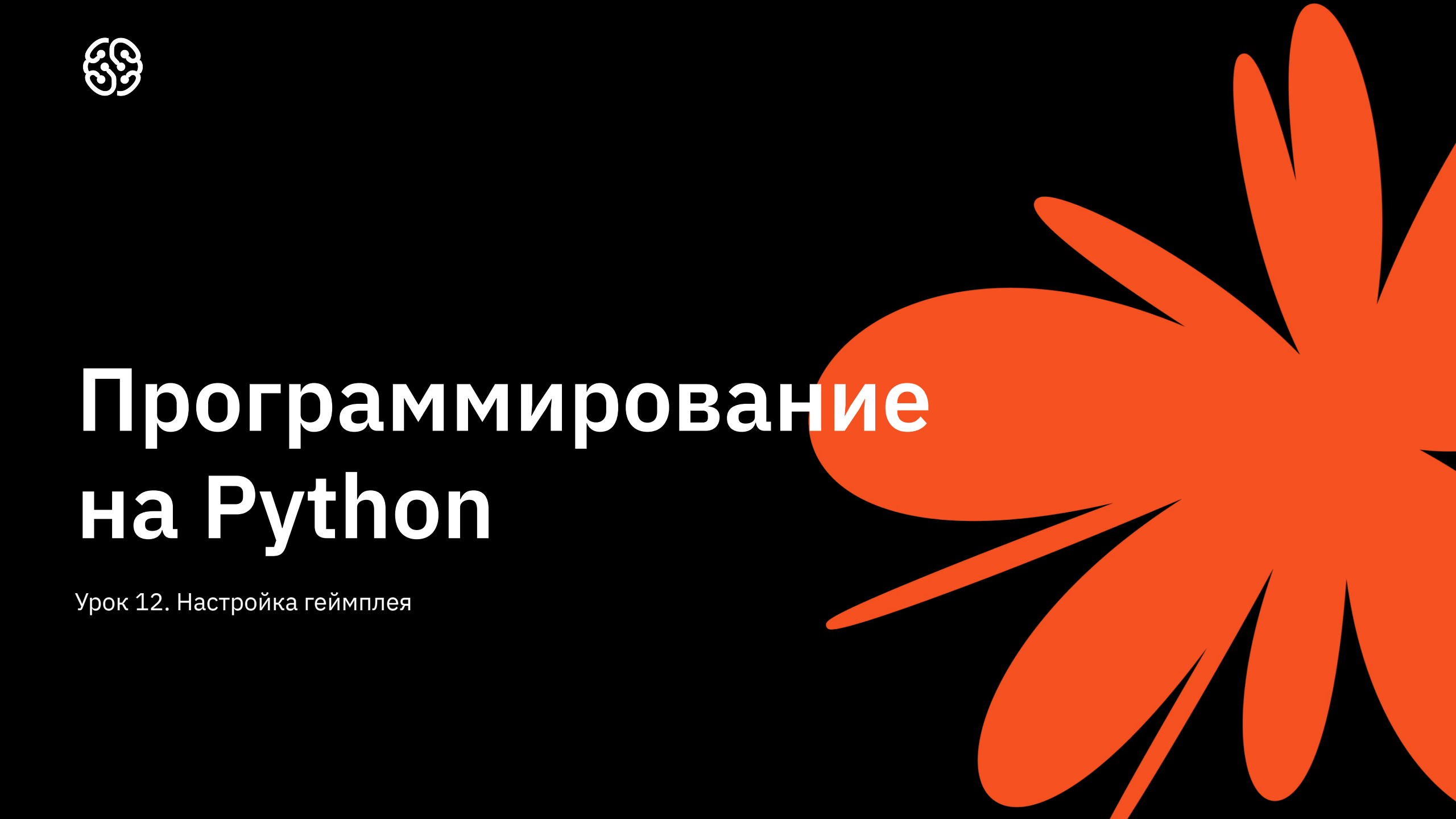


Программирование на Python

Урок 12. Настройка геймплея

Немного повторим прошлый урок

Что будет на уроке сегодня?

Добавим свойства здоровья игроку и мобам

Создадим и настроим класс со взрывом

Отобразим информацию о здоровье игрока на экране

Добавим взрывы в нужный момент

Здоровье игрока и мобов

Свойства здоровья

Первым делом нам необходимо задать количество жизней всем нашим объектам. Для этого перейдем в класс игрока и в каждый класс мобов и создадим внутри конструктора новые свойства, в которых будут храниться жизни. У игрока я сделаю 500 единиц здоровья, а у мобов по 100:

```
self.hp = 500            # Здоровье игрока
self.max_hp = self.hp    # Максимальное здоровье
```

```
self.hp = 100          # Здоровье моба
```

Разным мобам можно сделать разное количество здоровья

Повреждения и смерть игрока

Сначала давайте добавим событие смерти игрока. Находим событие столкновения групп игроков и мобов и добавляем в него код, который будет уменьшать количество жизней:

```
scratch = pygame.sprite.groupcollide(mobs_sprites, players_sprites, False, False)
if scratch:
    sprite = get_hit_sprite(scratch)
    sprite.snd_scratch.play()
    player.hp -= 1
```

Изменение количества жизней на 1 может показаться очень маленьким, но факт в том, что жизни будут уменьшаться всегда, пока моб касается игрока. Поэтому, ставить большие значения не рекомендуется.

Повреждения и смерть игрока

После этого нам необходимо завершать игру как только количество жизней игрока станет меньше 0. Поэтому ниже добавим еще одно условие:

```
    player.hp -= 1
    if player.hp <= 0:
        run = False
```

Теперь следует быть аккуратнее! Ведь игра может закончится, как только жизни упадут до 0.

Повреждения и смерть мобов

С мобами немножко все будет по-другому. Во первых мы должны отключить уничтожение моба при попадании пули:

```
shots = pygame.sprite.groupcollide(bullets_sprites, mobs_sprites, True, False)
```

Затем, как только получили спрайт, в который ударилась пуля, уменьшаем его количество жизней:

```
    sprite.hp -= 30
```

А вот звук взрыва и уничтожение моба мы будем делать только тогда, когда количество жизней моба станет меньше 0:

```
    if sprite.hp <= 0:
        sprite.snd_expl.play()          # Воспроизводим звук взрыва
        sprite.kill()                   # Уничтожаем спрайт
```

Повреждения и смерть мобов

Полный код события столкновения пули и моба теперь будет выглядеть так:

```
shots = pygame.sprite.groupcollide(bullets_sprites, mobs_sprites, True, False)
if shots:
    sprite = get_hit_sprite(shots)      # Получаем спрайт из второй группы
    sprite.hp -= 30
    if sprite.hp <= 0:
        sprite.snd_expl.play()          # Воспроизводим звук взрыва
        sprite.kill()
```

Вывод количества здоровья

Функция вывода здоровья на экран

Сейчас уже стало играть интереснее, но не имея представления о том сколько жизней осталось, сложно ориентироваться. Давайте добавим новую функцию draw_hp, чтобы можно было знать сколько жизней осталось у игрока. Создадим эту функцию до игрового цикла:

```
def draw_hp(screen, x, y, hp_width, hp_height, player):
    green = "#32CD32"                                    # Зеленый цвет
    white = "#FFFFFF"                                    # Белый цвет
    rect = pygame.Rect(x, y, hp_width, hp_height)        # Создаем рамку
    fill = (player.hp / player.max_hp) * hp_width        # Считаем ширину полосы hp
    fill_rect = pygame.Rect(x, y, fill, hp_height)       # Создаем полосу для hp
    pygame.draw.rect(screen, color, fill_rect)           # Рисуем полосу для hp
    pygame.draw.rect(screen, white, rect, 1)             # Рисуем рамку
```

Функция вывода здоровья на экран

А потом в игровом цикле вызовем её после отрисовки всех спрайтов, чтобы видеть количество жизней:

```
# Рендеринг (прорисовка)
screen.fill(CYAN)                       # Заливка заднего фона
all_sprites.draw(screen)                # Отрисовываем все спрайты
draw_hp(screen, 50, 50, 200, 20, player)
```

Дополнительно можно добавить функцию для вывода текста с количеством жизней на экран. Делается это аналогичным способом, как и в игре с арканоидом.

Перерыв 10 мин

Добавляем взрывы

Стартовый код класса взрыва

Такой же точно, как и для остальных спрайтов:

```
import pygame

snd_dir = 'media/snd/'  # Путь до папки со звуками
img_dir = 'media/img/'  # Путь до папки со спрайтами
width = 1366  # ширина игрового окна
height = 768  # высота игрового окна


# Создаем класс игрока
class Explosion(pygame.sprite.Sprite):
    def __init__(self, center):
        pygame.sprite.Sprite.__init__(self)
```

Точка появления взрыва

Нам важно создавать взрывы в том месте, где у нас умирает моб с гранатой, поэтому, мы добавим внутрь скобок рядом с self еще одну переменную, в которую будем записывать координаты того места, где взрыву надо появиться:

```
class Explosion(pygame.sprite.Sprite):
    def __init__(self, center):
        pygame.sprite.Sprite.__init__(self)
```

Необходимые свойства

Сначала укажем скорость с которой будет проигрываться анимация взрыва

```
self.anim_speed = 4         # Скорость анимации
```

Затем укажем номер стартового кадра анимации. Он будет равен нулю:

```
self.frame = 0              # Текущий кадр анимации
```

Потом создадим переменную, в которой будут храниться все кадры анимации. Воспользуемся специальной командой для заполнения:

```
self.anim = [pygame.transform.scale(
       pygame.image.load(img_dir + f'./explosion/{i}.png'),
       (100, 100)) for i in range(9)]
```

Здесь цифра 9 означает количество картинок для анимации

Необходимые свойства

Далее настраиваем картинку. Так как для анимации мы сжимали картинки, то возьмем первую картинку из тех что загрузили:

```
self.image = self.anim[0]          # Стартовая картинка спрайта
```

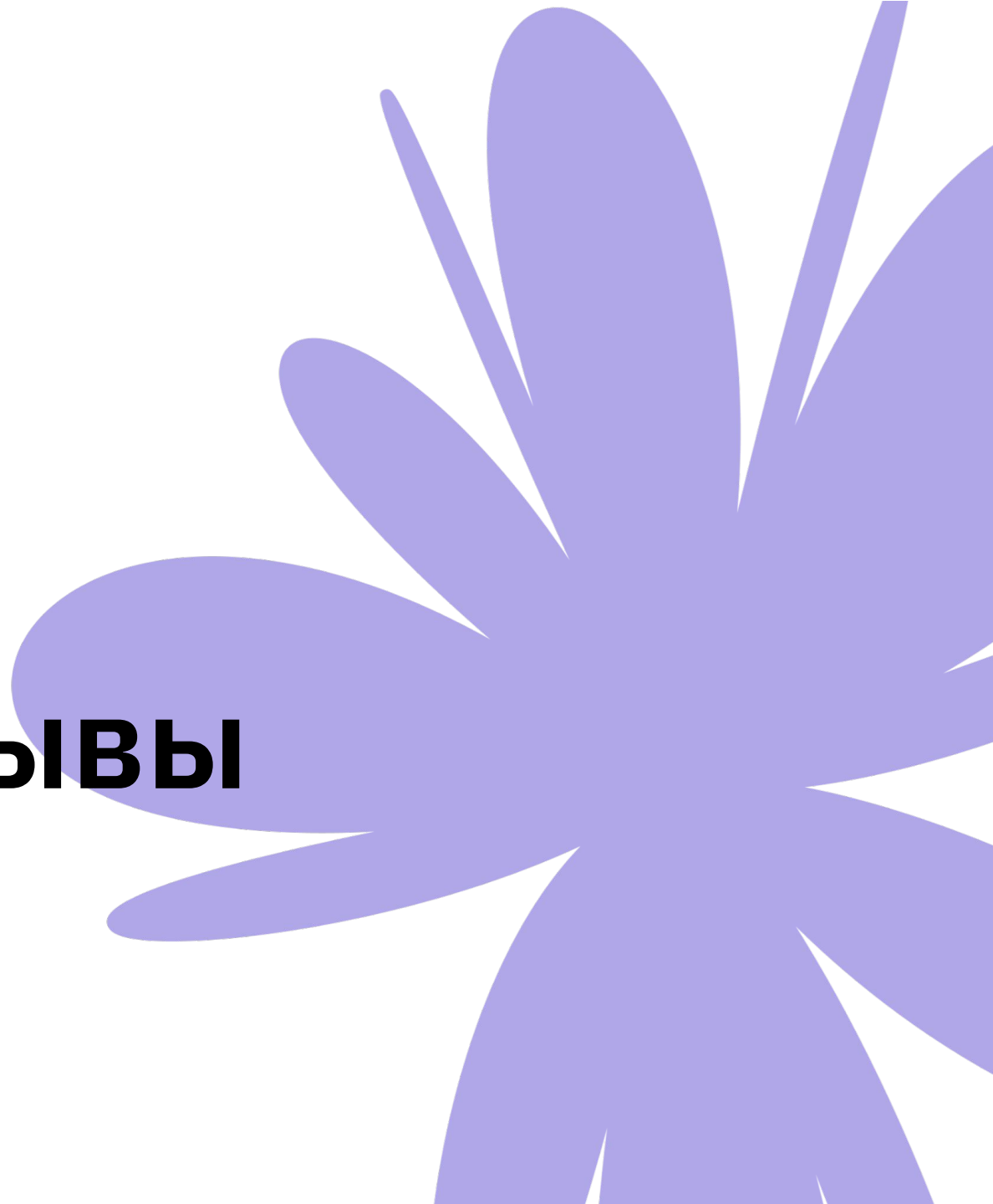
Далее, традиционно берем рамку:

```
self.rect = self.image.get_rect()
```

Смещаем центр в положение указанное при создании взрыва:

```
self.rect.center = center
```

Анимируем взрывы



Сменяем кадры

Единственное место, которое хранит картинку игрока - это свойство self.image. А значит, нам необходимо в него постоянно подгружать новые и новые картинки, чтобы создавалось впечатление анимации. Создадим метод `def update(self):` и в нем напишем формулу благодаря которой картинки будут меняться

```
def update(self):
    self.image = self.anim[self.frame // self.anim_speed] # следующая картинка
```

Далее увеличиваем номер кадра

```
    self.frame += 1
```

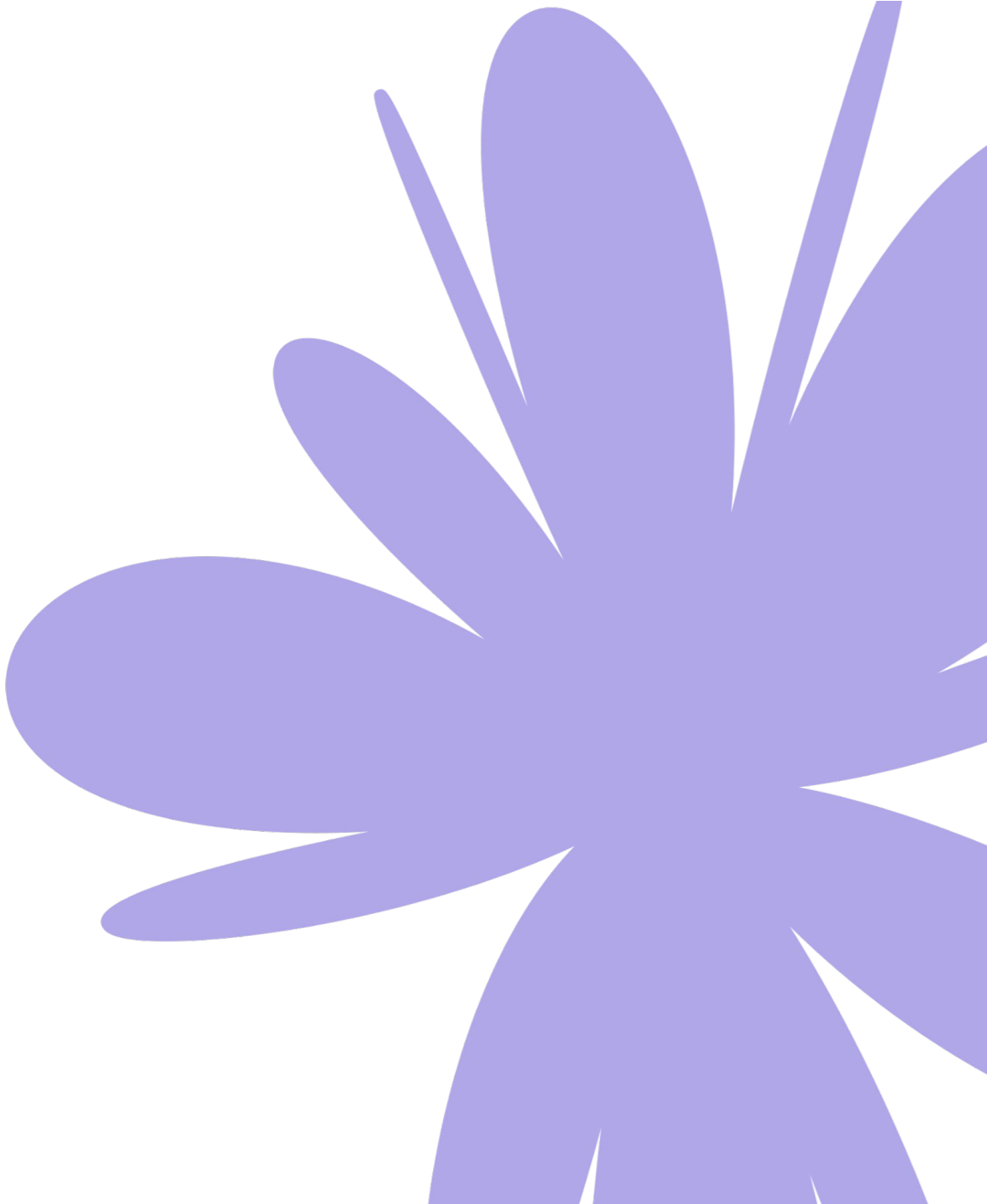
Последний кадр, конец анимации

Проверять последний кадр анимации очень важно, потому что если мы вдруг попросим наш список дать нам картинку с номером, которого не существует, то программа завершится с ошибкой. Поэтому проверяем:

```
if self.frame >= self.anim_speed*len(self.anim):
    self.kill()
```

Мы постоянно увеличиваем кадры во время анимации и в какой то момент номер кадра станет больше чем каров у нас есть. Поэтому просто сравниваем номер кадра с последним `self.anim_speed*len(self.anim)` И, как только данное равенство выполнится или даже вдруг станет больше, нам необходимо уничтожить спрайт. Поэтому вызываем уже знакомый нам метод `self.kill()`

Взрываем



Момент взрыва

Нам нужно создать спрайт взрыва и добавить его ко всем спрайтам тогда когда здоровье нужного моба уменьшится до 0:

```
shots = pygame.sprite.groupcollide(bullets_sprites, mobs_sprites, True, False)
if shots:
    sprite = get_hit_sprite(shots)  # Получаем спрайт из второй группы
    sprite.hp -= 30
    if sprite.hp <= 0:
        sprite.snd_expl.play()          # Воспроизводим звук взрыва
        expl = Explosion(sprite.rect.center)
        all_sprites.add(expl)
        sprite.kill()
```

Здесь `sprite.rect.center` - это центр спрайта, который должен быть уничтожен. Именно его координаты мы передаем внутрь взрыва, чтобы он создался в том же месте

Результат

Весь проект с готовыми файлами можно скачать здесь:

https://github.com/ronmount/gb_shooter/archive/refs/heads/lesson4.zip

Итоги

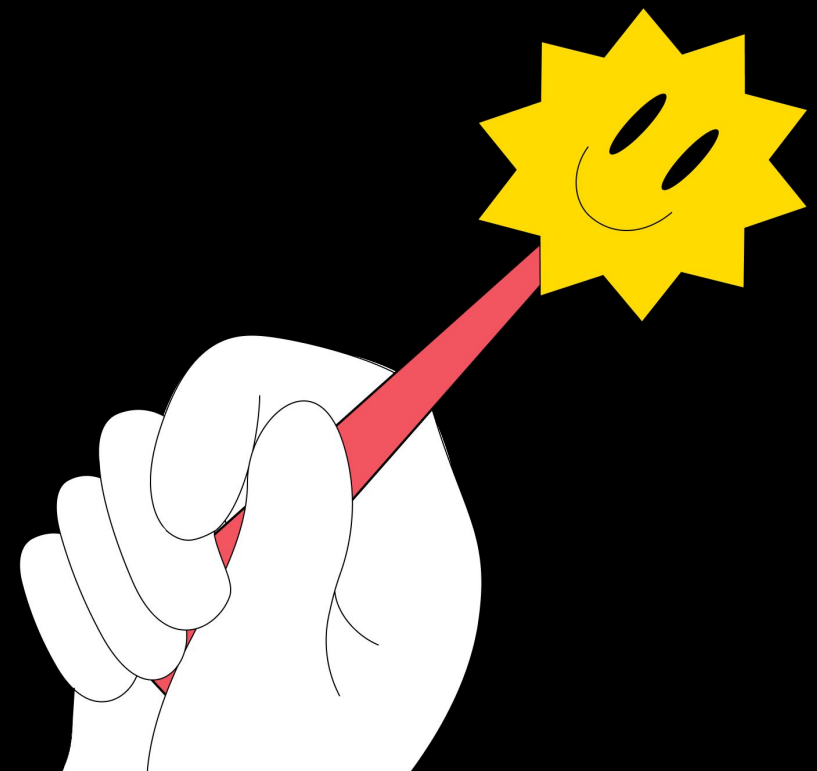
- ✔ Изучили способы контроля жизни объектов
- ✔ Повторили способ отображения жизней на экране
- ✔ Научились делать покадровую анимацию
- ✔ Добавили Жизни игровым объектам
- ✔ Отобразили здоровье игрока на экране
- ✔ Создали Взрывы в момент уничтожения мобов

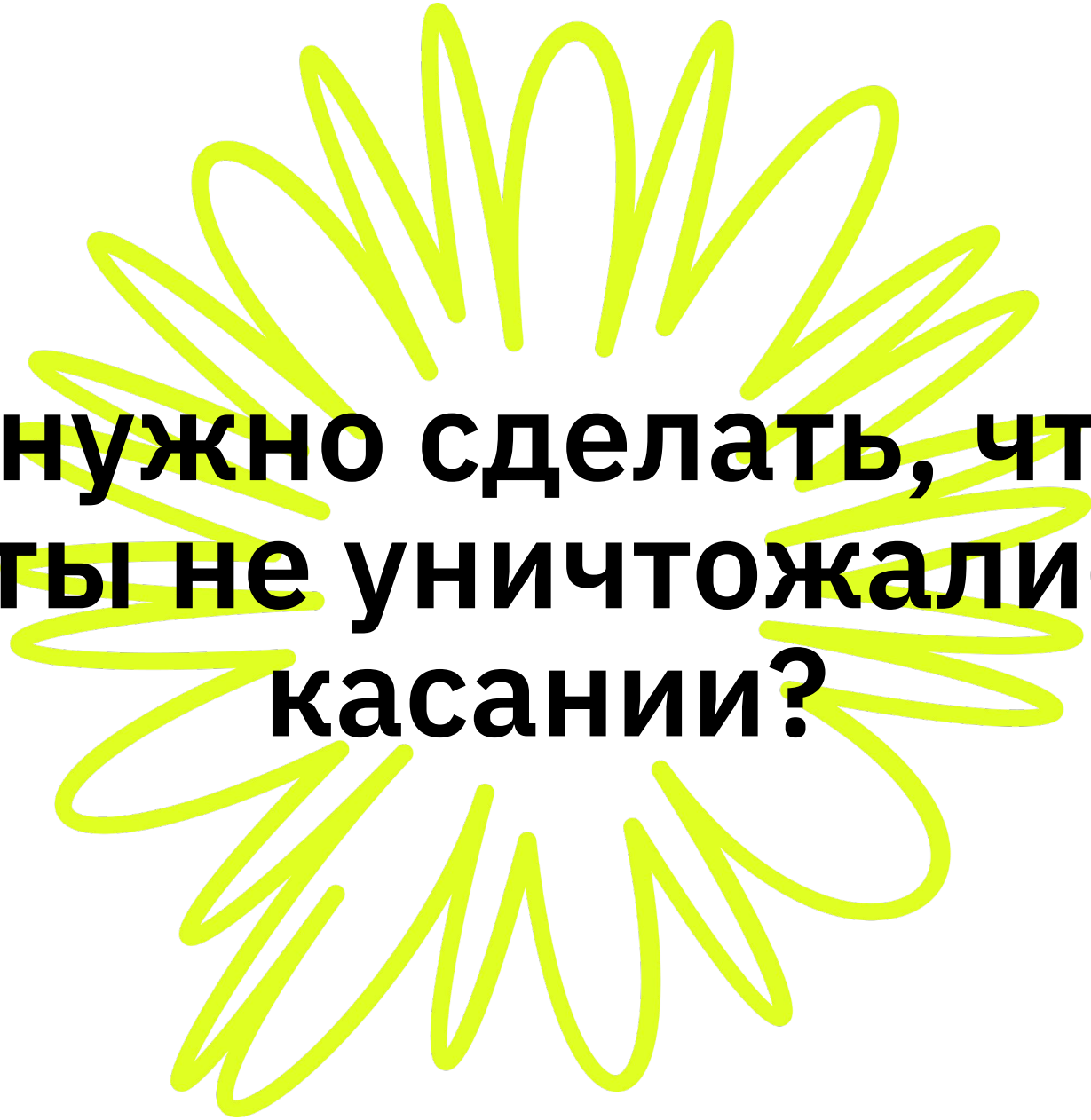
На следующем занятии:

Добавим меню

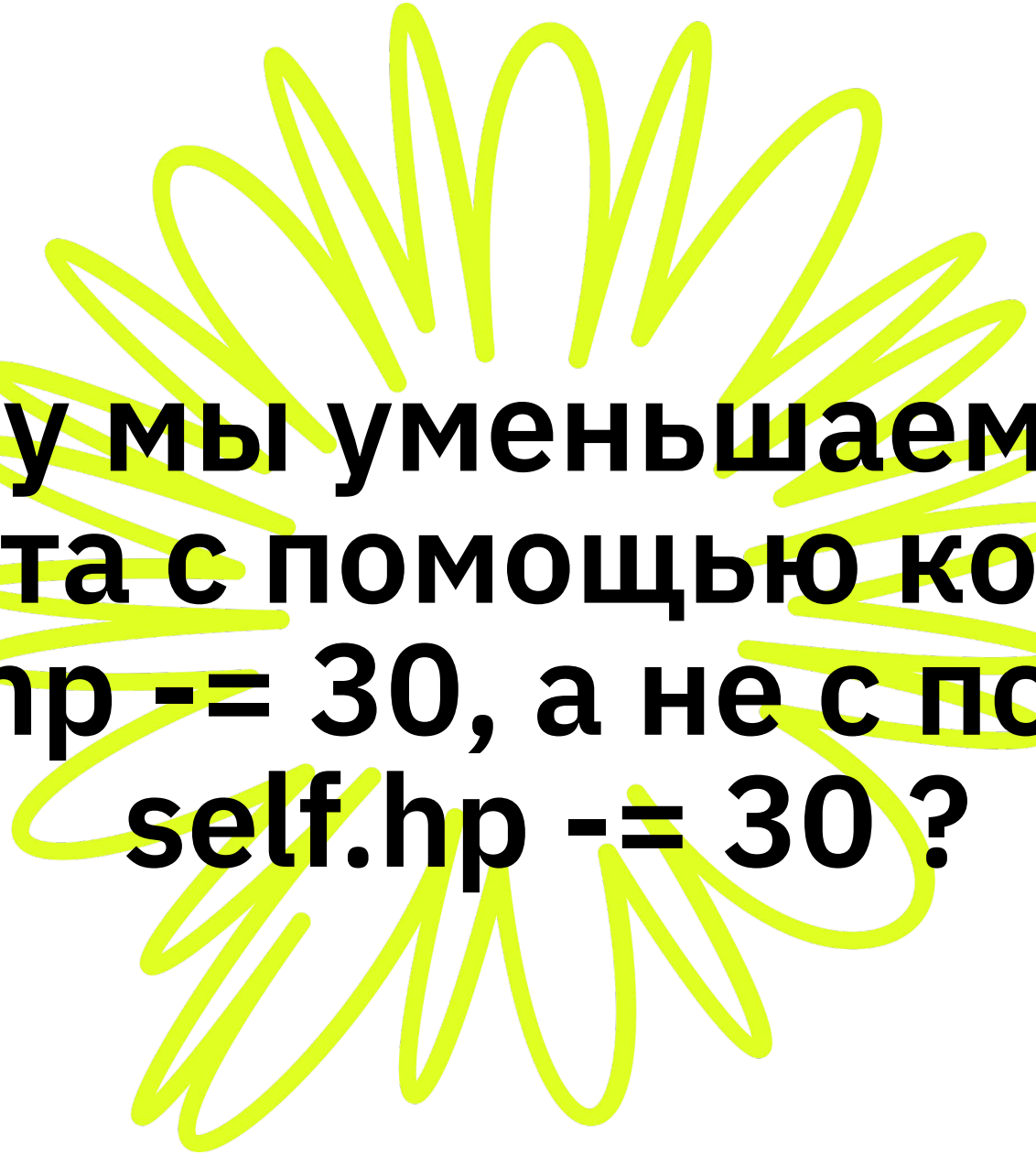
Добавим возможность перезапуска игры

Немного повторим

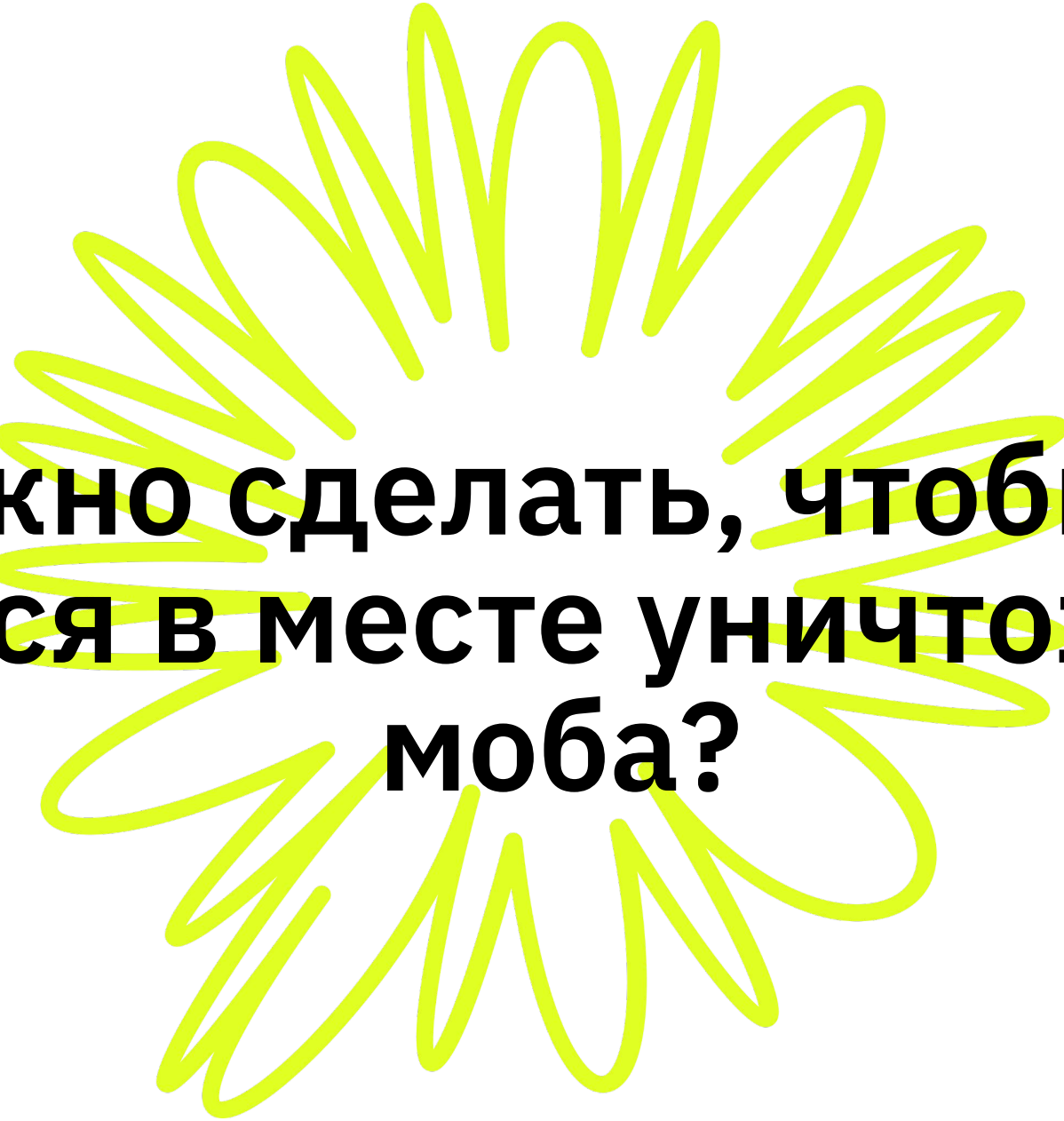




Что нужно сделать, чтобы спрайты не уничтожались при касании?

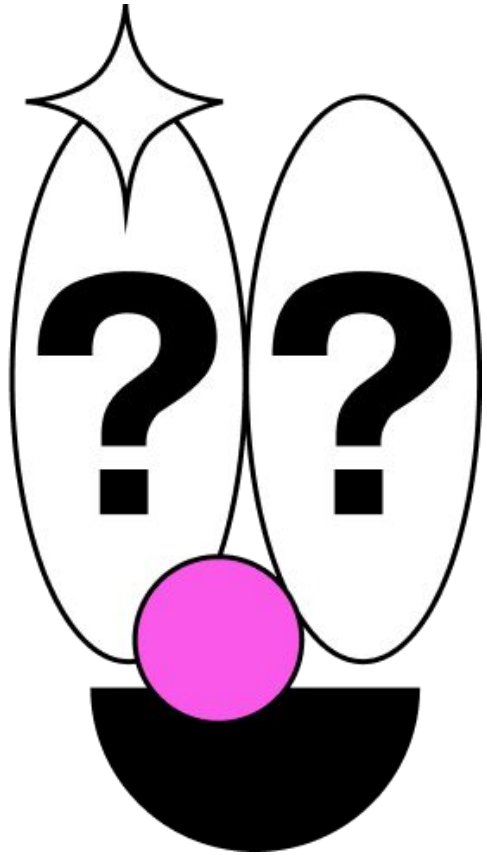


Почему мы уменьшаем жизни спрайта с помощью команды sprite.hp -= 30, а не с помощью self.hp -= 30 ?



Что нужно сделать, чтобы взрыв появился в месте уничтоженного моба?

Ваши вопросы



Спасибо за внимание

Домашнее задание

Заполни, пожалуйста, форму обратной связи по уроку

Напоминание для преподавателя

- **Проверить заполнение Журнала**
- **Заполнить форму Т22**