



Программирование на Python

Урок 13. Игровое меню и события



Немного повторим прошлый урок



Урок 13. Игровое меню и события

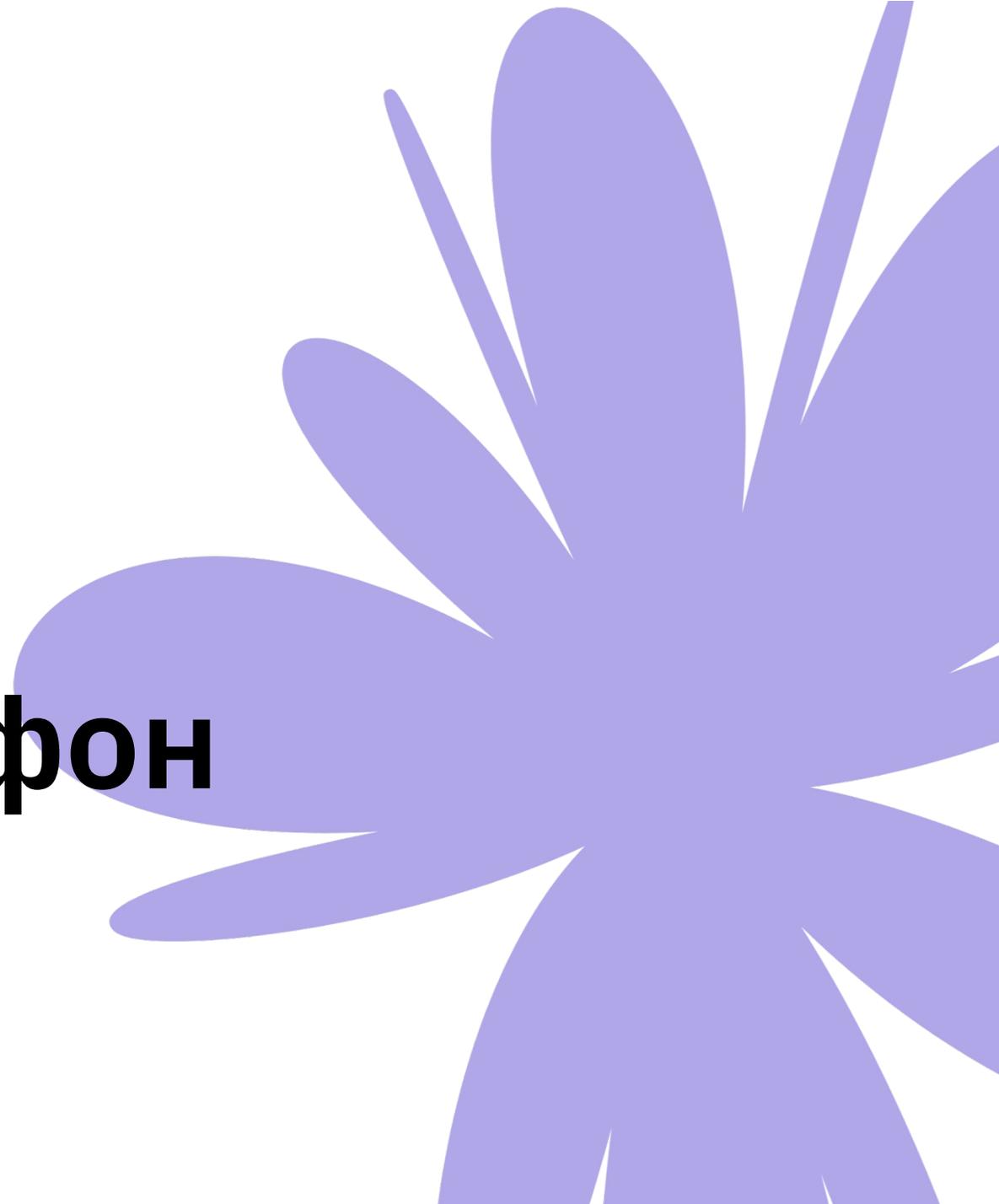
Что будет на уроке сегодня?

Добавим игровое меню

Настроим событие смерти игрока ведущее
в игровое меню

Добавим вращающийся фон

Вращающийся фон



Класс с фоном

Мы уже с вами создавали классы несколько раз. И здесь будет всё то же самое. В свойства необходимо добавить всё нужное для вращения:

```
import pygame
from pygame.math import Vector2

snd_dir = 'media/snd/' # Путь до папки со звуками
img_dir = 'media/img/' # Путь до папки со спрайтами
width = 1280           # ширина игрового окна
height = 720          # высота игрового окна

# Создаем класс игрока
class Bg(pygame.sprite.Sprite):
    def __init__(self): # Специальная функция, где указываем что будет у игрока
        pygame.sprite.Sprite.__init__(self) # Игрок - спрайт

        self.image = pygame.image.load(img_dir + 'bg.jpg')
        self.rect = self.image.get_rect()
        self.rect.center = [width/2, height/2]
        self.copy = self.image
        self.position = Vector2(self.rect.center)
        self.direction = Vector2(0, -1)
        self.angle = 0
```


Урок 13. Игровое меню и события

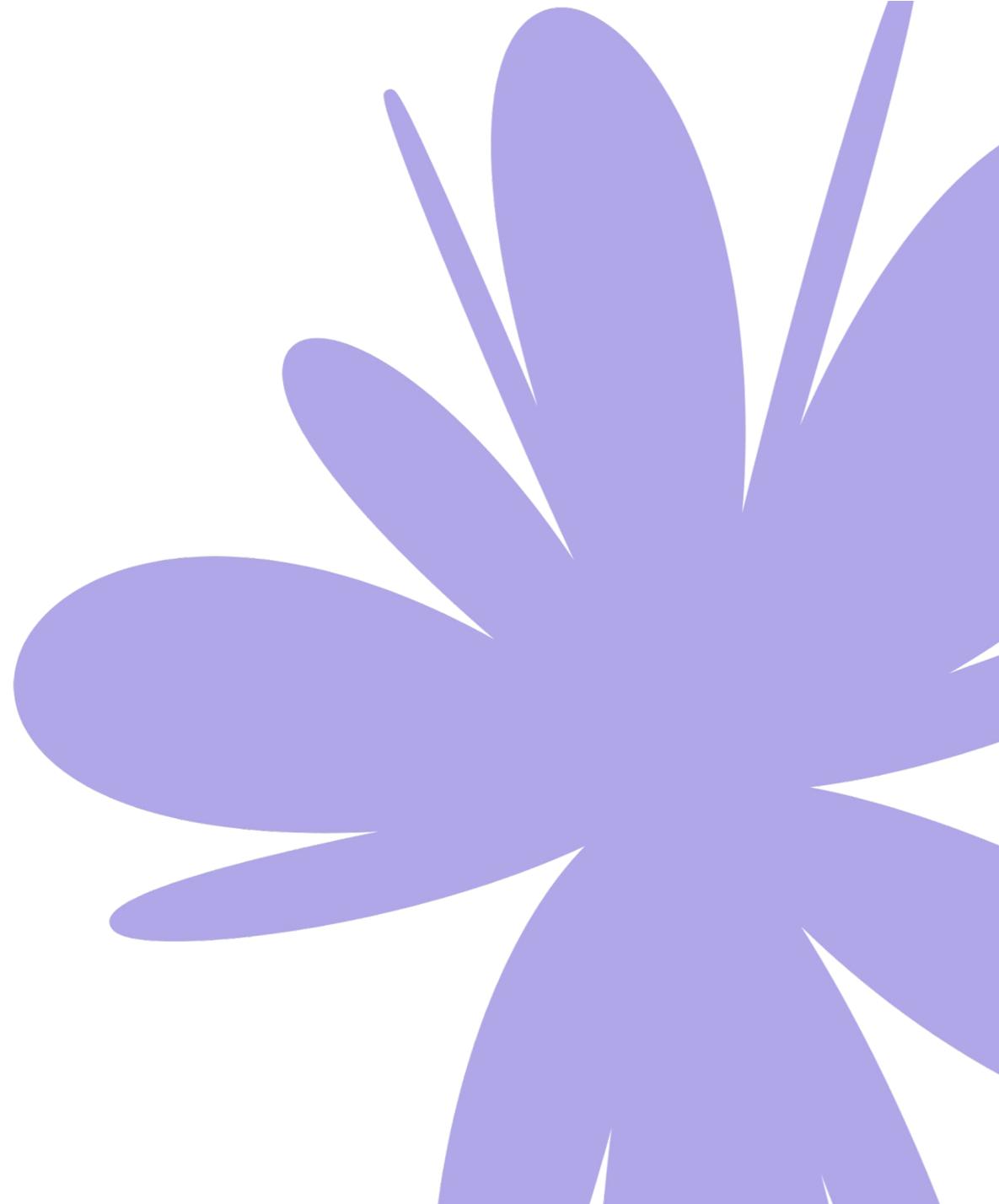
Класс с фоном

Не забудьте добавить ко всем спрайтам в основном файле игры:

```
player = Player() # Создаем игрока класса Player
bg = Bg()         # Создаем класс заднего фона

all_sprites.add(bg)      # Сначала добавляем фон
all_sprites.add(player)  # Добавляем игрока в группу спрайтов
```

Игровое меню



Функция вывода текста на экран

Это точно такая же функция, как и в прошлой игре, поэтому просто добавим её в наш основной файл игры:

```
def draw_text(screen, text, size, x, y, color):  
    font_name = pygame.font.match_font('arial') # Выбираем тип шрифта для текста  
    font = pygame.font.Font(font_name, size) # Шрифт выбранного типа и размера  
    text_image = font.render(text, True, color) # Превращаем текст в картинку  
    text_rect = text_image.get_rect() # Задаем рамку картинке с текстом  
    text_rect.center = (x, y) # Переносим текст в координаты  
    screen.blit(text_image, text_rect) # Рисуем текст на экране
```

Функция игрового меню

Функция игрового меню по сути представляет из себя набор вызовов функции `draw_text` с отображением определенного текста в нужных частях экрана. Создадим данную функцию прямо перед игровым циклом и добавим в неё сначала вывод заднего фона, а потом вывод нужного нам текста:

```
def menu():
    screen.blit(bg.image, bg.rect)           # Включаем задний фон
    draw_text(screen, game_name, 128, width / 2, height / 4, WHITE)
    draw_text(screen, "Arrows for move, space - fire", 44,
                width / 2, height / 2, WHITE)
    draw_text(screen, "Press any key to start", 36, width / 2, height * 3 / 4, WHITE)
    pygame.display.flip()                   # Отображаем содержимое на экране
```

Ожидание нажатия клавиши

Чтобы наше меню отображалось постоянно и исчезло только в момент нажатия клавиши, нам придется сделать бесконечный цикл внутри, который будет отслеживать нажатие клавиши, а также нажатие на крестик окна (вдруг захотим закрыть игру так и не начав играть). Организация цикла ожидания очень похожа на организацию игрового цикла:

```
draw_text(screen, "Press any key to start", 36, width / 2, height * 3 / 4, WHITE)
pygame.display.flip() # Отображаем содержимое на экране
run = True
while run:
    timer.tick(fps) # Тикаем игровой таймер
    for event in pygame.event.get():
        if event.type == pygame.QUIT: # Событие закрытия окна
            pygame.quit()
        if event.type == pygame.KEYUP: # Событие нажатия любой клавиши
            run = False
```

Перерыв 10 мин



Включение игры и рестарт



Признак конца игры

Теперь нам необходимо немного изменить наш игровой цикл, так чтобы он в самом начале отображал наше игровое меню. А также возвращался всякий раз, когда игрок умирал.

Для этого до игрового цикла мы заведем переменную которая будет определять конец игры. Сделать это нужно до игрового цикла. И так как наша игра еще не началась, то запишем в неё значение True:

```
game_over = True
```

А внутри игрового цикла мы проверим её значение. И если она будет равна True, то просто включим наше игровое меню, а состояние конца игры переключим в False:

```
while run: # Начинаем бесконечный цикл
    if game_over:
        game_over = False
        menu()
    timer.tick(fps) # Контроль времени (обновление игры)
```

Событие смерти игрока

Теперь осталось только найти событие, когда игрок умирает и поменять переменную `gun`, которая отвечает за игровой цикл на переменную `game_over`, которая отвечает за конец игры:

```
if scratch:
    sprite = get_hit_sprite(scratch)
    sprite.snd_scratch.play()
    player.hp -= 1
    if player.hp <= 0:
        game_over = True
```

Воскрешение игрока и рестарт

Всё работает отлично, мы действительно попадаем с вами обратно в игровое меню. Правда есть одна проблема - заново запустить игру по второму разу не получится. А всё потому, что у нас значение здоровья игрока как было 0, так и осталось 0. Нужно это изменить, дописав несколько команд внутри проверки на конец игры.

Просто пересоздадим игрока, просто вызвав у него функцию, которая запишет все его свойства заново:

```
while run: # Начинаем бесконечный цикл
    if game_over:
        player.__init__()
```

Респаун мобов

Игрок перерождается с нуля, а вот мобы остаются в прежнем состоянии. Чтобы этого избежать и к тому же добавить еще возможность появления новых мобов, когда закончатся старые, давайте напишем отдельную функцию, в которой мы будем просто создавать этих мобов и добавлять в соответствующие группы. Причем функция будет создавать указанное количество мобов, передаваемое через параметр. **До игрового цикла** напишем нашу функцию.

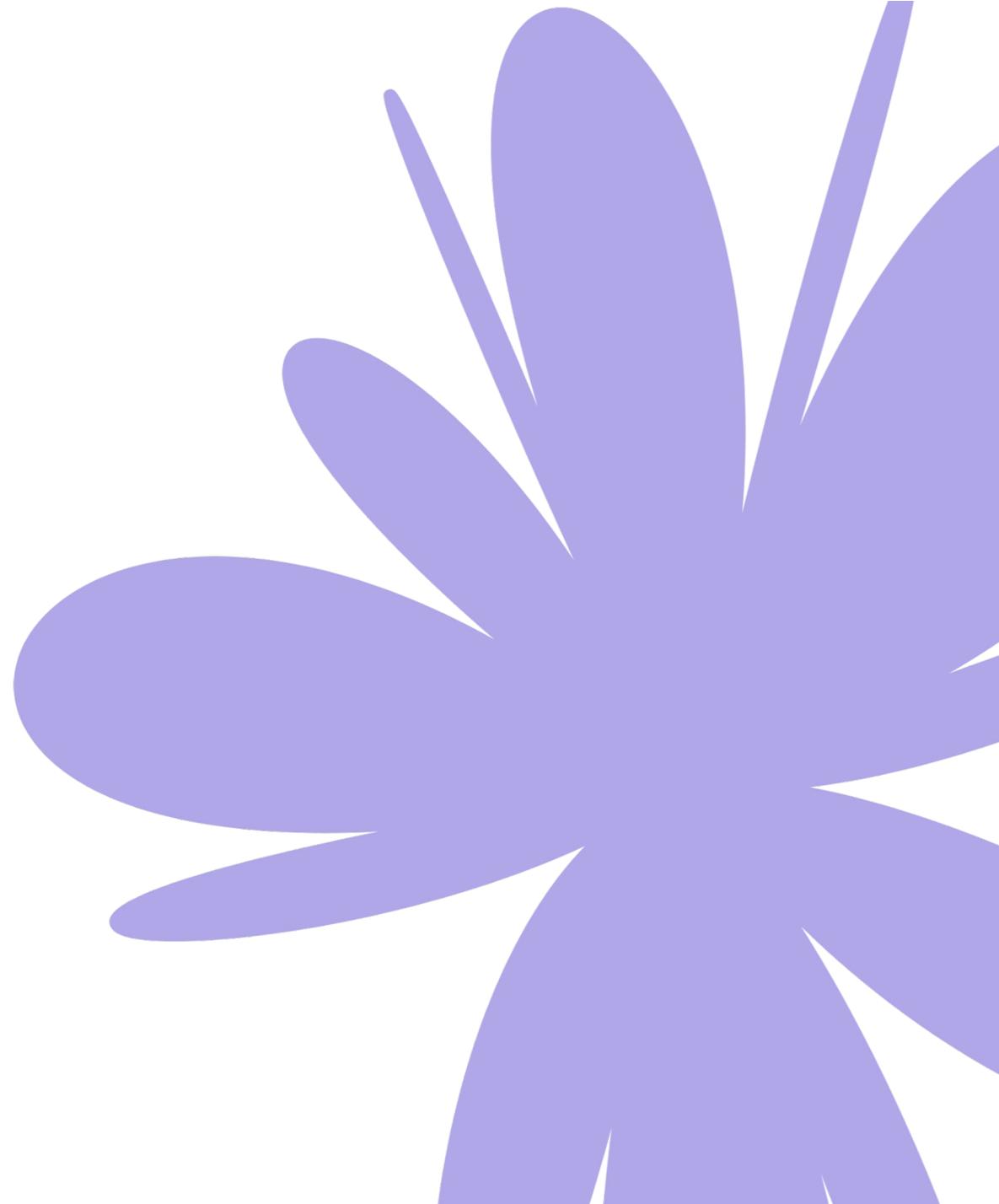
```
def new_mobs(count):  
    for i in range(count):  
        el = EnemyLeft()  
        er = EnemyRight()  
        et = EnemyTop()  
        eb = EnemyBottom()  
        all_sprites.add([el, er, et, eb])  
        mobs_sprites.add([el, er, et, eb])
```

Респаун мобов

А создавать мы их будем именно в момент отображения игрового меню. Правда сначала нам необходимо очистить группу от старых спрайтов на случай, если они там останутся после проигрыша игрока:

```
while run: # Начинаем бесконечный цикл
    if game_over:
        player.__init__() # Пересоздаем игрока
        for sprite in mobs_sprites: # Очищаем группу спрайтов
            sprite.kill()
        new_mobs(1) # Создаем мобов каждого по 1 штуке
        game_over = False
        menu()
```

Уровни в игре



Урок 13. Игровое меню и события

Переменная с уровнями

Создается как обычная переменная. До игрового цикла. Начальный уровень равен 1:

```
level = 1
```

Генерация мобов в зависимости от уровня

Давайте изменим вызов нашей функции генерации мобов так, чтобы с каждым уровнем, мобов появлялось все больше. Для этого мы просто будем передавать номер уровня в момент вызова функции:

```
while run: # Начинаем бесконечный цикл
    if game_over:
        player.__init__() # Пересоздаем игрока
        for sprite in mobs_sprites: # Очищаем группу спрайтов
            sprite.kill()
        new_mobs(level) # Создаем мобов по уровню
        game_over = False
        menu()
```

Управление уровнем

Должно происходить, когда мобов не останется. Идем в игровой цикл и сразу под событием где фиксируется попадание пули в моба пишем проверку:

```
if len(mobs_sprites) == 0: # Если мобов в группе не осталось
    level += 1             # Увеличиваем уровень
    new_mobs(level)       # Создаем новых мобов
```

Также нам необходимо сбросить уровень до 1, когда игрок умрет и код перейдет в наше меню. Поэтому добавим еще там строчку кода:

```
if game_over:
    level = 1
    player.__init__()
```

Урок 13. Игровое меню и события

Результат

Весь проект с готовыми файлами можно скачать здесь:

https://github.com/ronmount/gb_shooter/archive/refs/heads/lesson5.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

Урок 13. Игровое меню и события

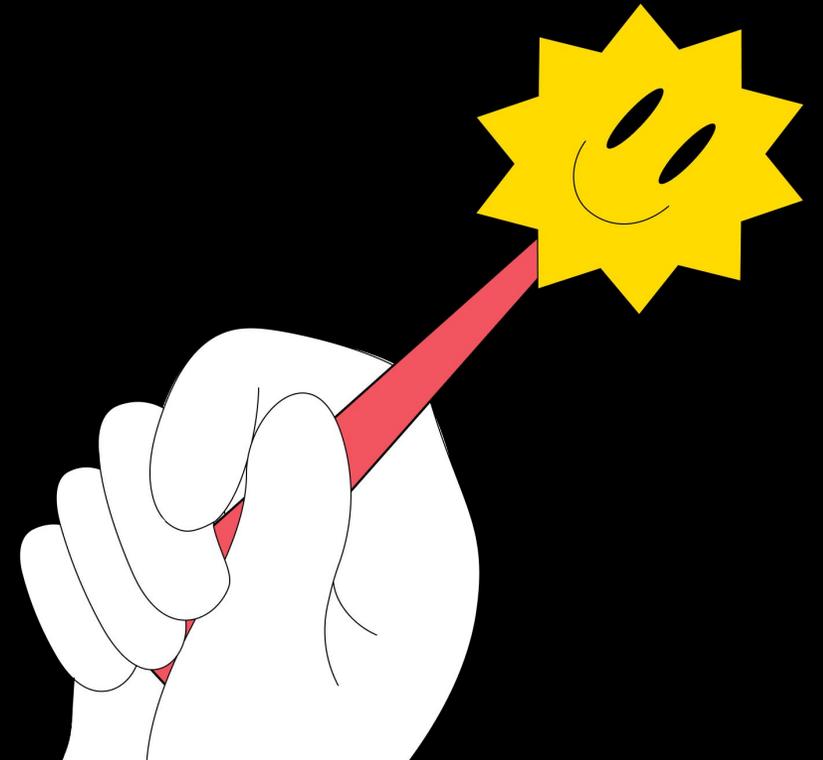
На следующем занятии:

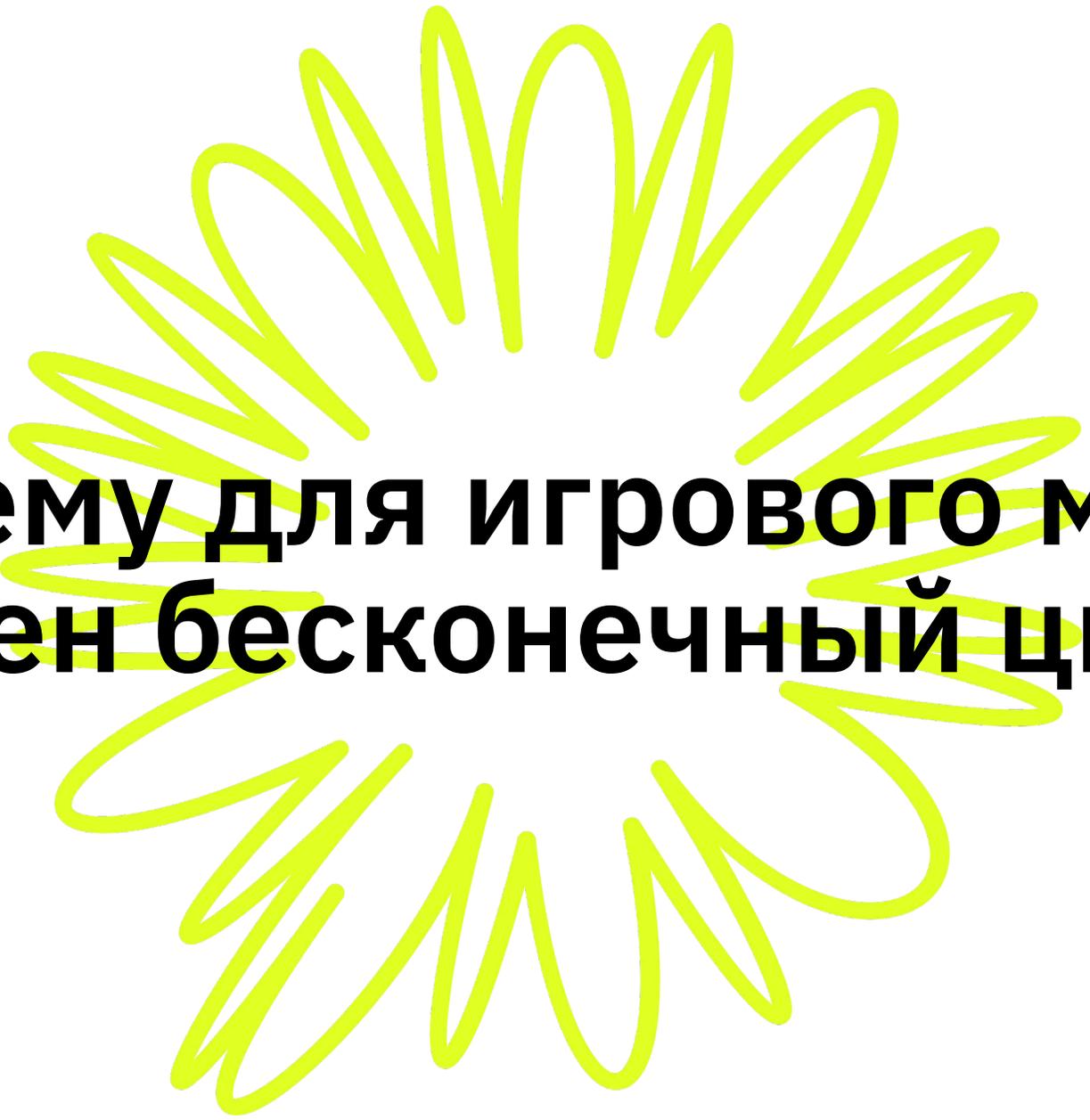
Разделимся на команды

Начнем групповую разработку над новой игрой



Немного повторим

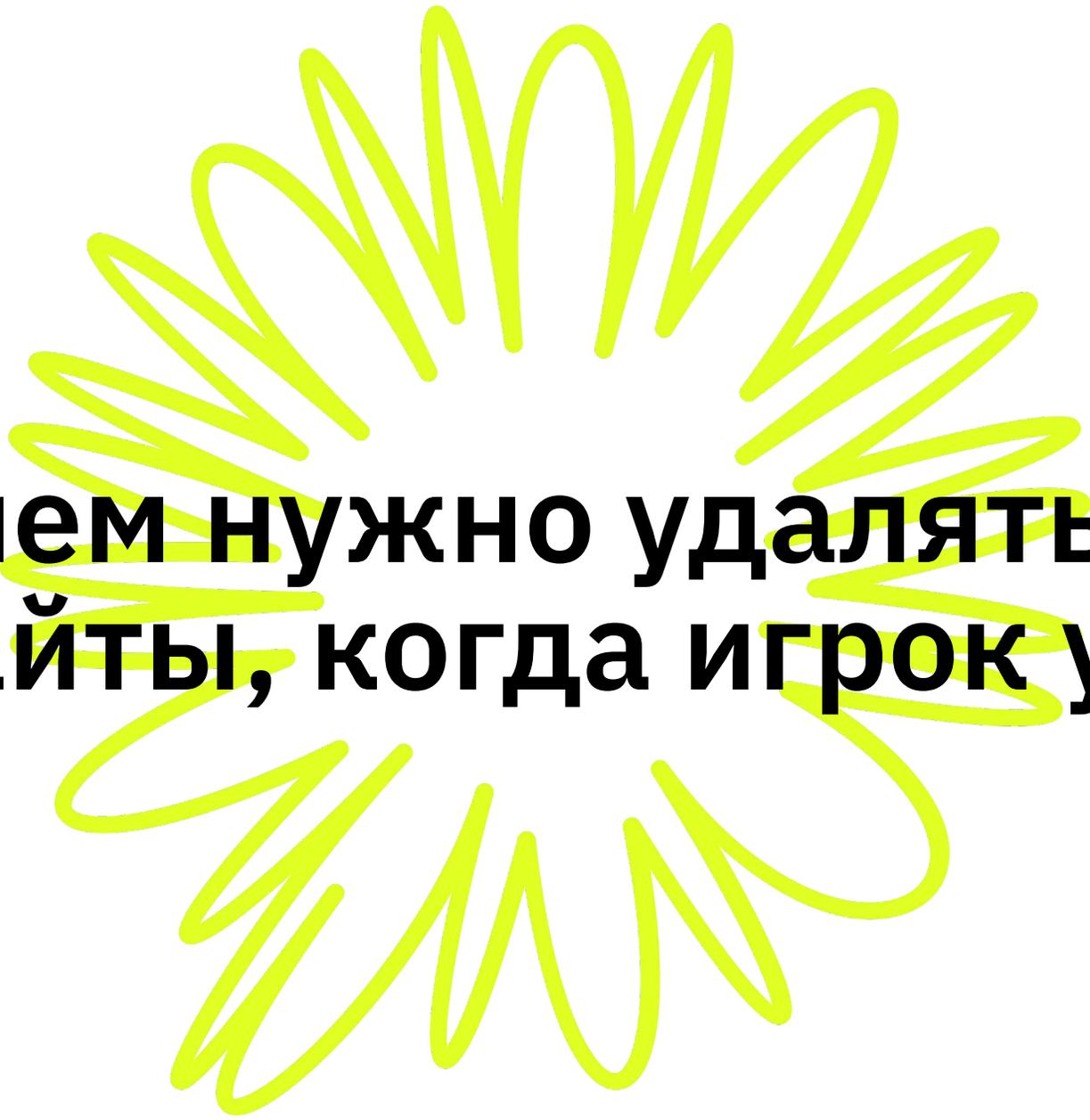




**Почему для игрового меню
нужен бесконечный цикл?**



Можно ли менять положение и надписи игрового меню?

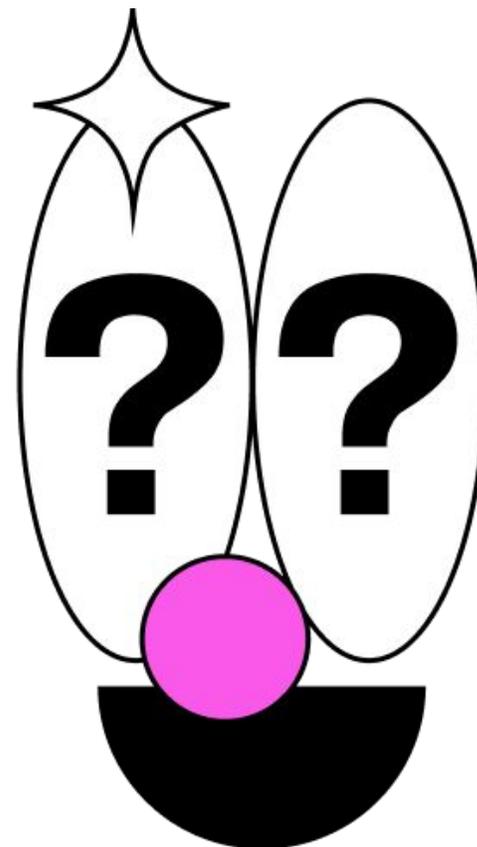


**Зачем нужно удалять все
спрайты, когда игрок умер?**



**Как ругате понимает, что нужно
включить следующий уровень?**

Ваши вопросы

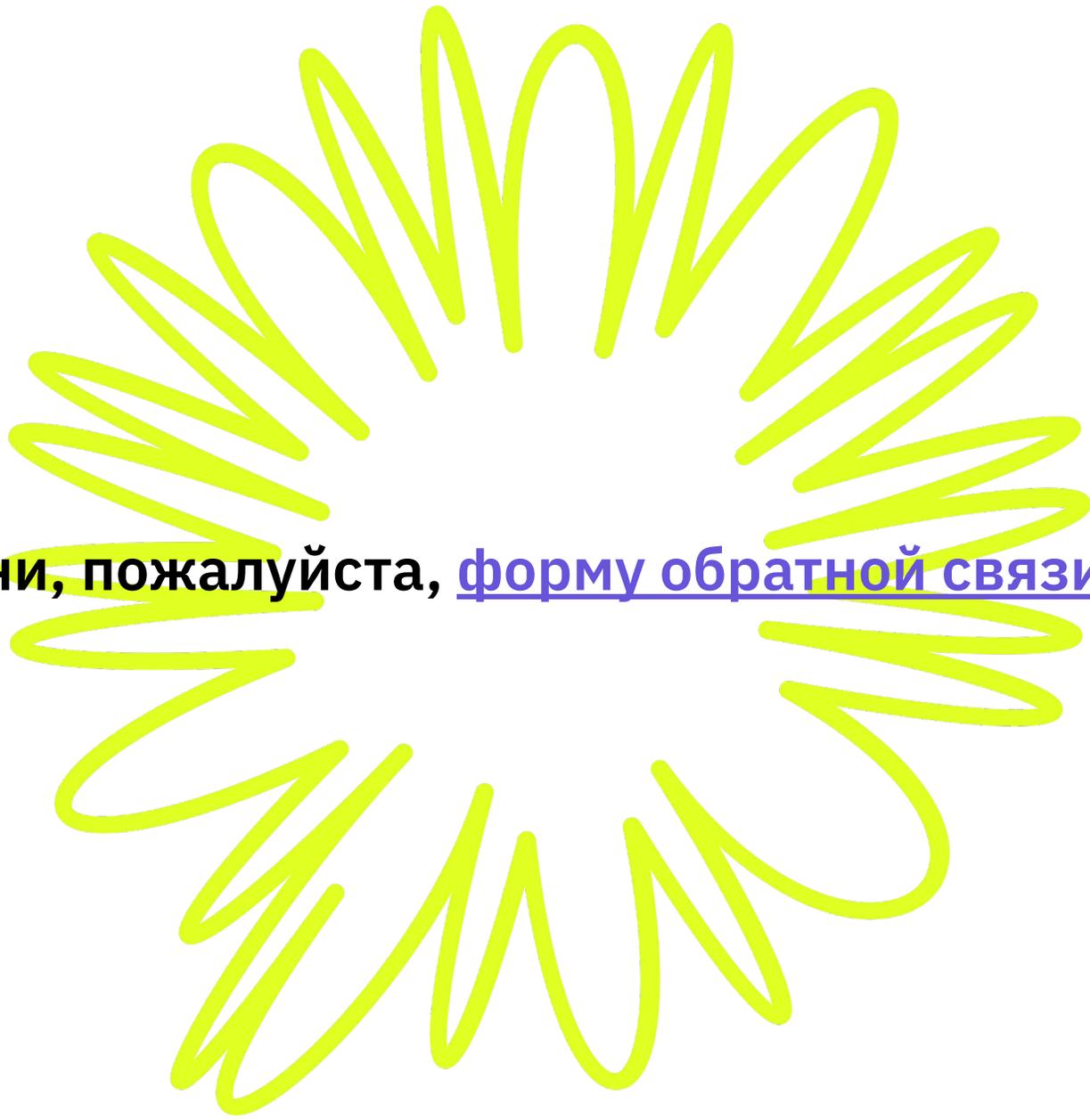




Спасибо 
за внимание



Домашнее задание



Заполни, пожалуйста, форму обратной связи по уроку

Напоминание для преподавателя

- Проверить заполнение Журнала
- Заполнить форму T22