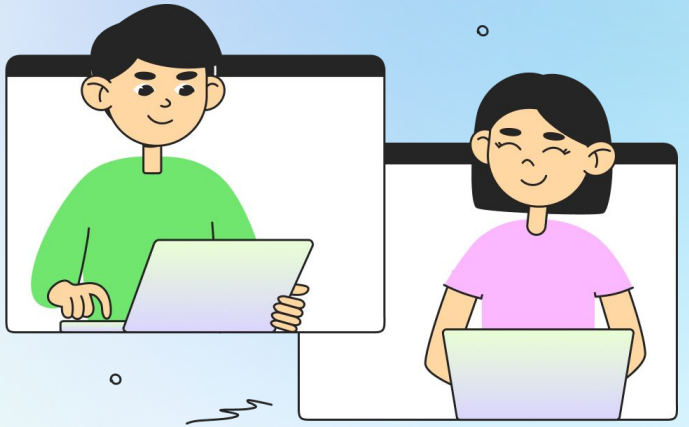
GeekSchool

Программирование на Python

Урок 8

Создаем gameplay

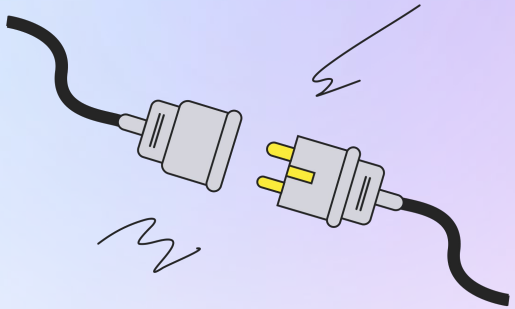


Немного повторим прошлый урок

Что будет на уроке сегодня?

- Добавим события проигрыша
- Снабдим мячик жизнями и выведем их на экран
- Реализуем логику потери жизней
- Добавим звуки и музыку
- Добавим задний фон и анимируем его

Событие проигрыша



Убираем нижний отскок

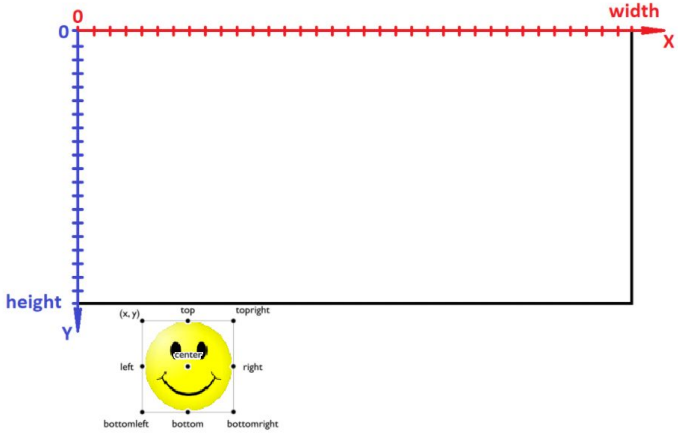
Для начала давайте найдем это событие. Важно не перепутать! Когда точка bottom рамки спрайта становится больше чем максимальное значение Y (height):

```
if pic_rect.bottom > height:                # Если достигли низа экрана
    pass                                    # Пустая команда, ничего не делает
```

📌 **pass** — это команда заглушка позволяющая запускать код с незавершенными блоками, но при этом данная команда ничего не делает.

Событие конца игры

Чтобы выглядело всё более реалистично, сделаем так, чтобы игровой цикл заканчивался тогда, когда весь наш смайлик спрячется за нижней частью экрана. А это значит, что координата его верхней точки рамки должна стать больше высоты экрана.



Переделаем наш код:

```
if pic_rect.top > height:                    # Если зашли за нижнюю границу экрана
    run = False                              # Завершаем игровой цикл
    print('Game Over')                       # Выводим надпись на экран
```

Добавляем жизни игроку

Добавляем новый параметр

Так как жизни игрока — это его характеристика, такая же как и скорость,
то вынесем это за пределы игрового цикла:

```
speedx = 10
speedy = 10
lives = 3                                   # Жизни игрока

racket = pygame.image.load('racket.png')  # Загружаем спрайт игрока
racket_rect = racket.get_rect()           # Получаем рамку спрайта игрока
```

Логика потери жизни

Как только мы теряем смайлик из виду, нам необходимо уменьшить количество жизней.
Переходим в игровой цикл и вместо остановки игрового цикла уменьшаем количество жизней:

```
if pic_rect.top > height:                    # Если зашли за нижнюю границу экрана
    lives -= 1
    # run = False                            # Завершаем игровой цикл
    # print('Game Over')                     # Выводим надпись на экран
```

Обратите внимание, что мы пока закомментировали строки, в которых отключали игровой цикл и выводили сообщение о конце игры. Так часто делают, когда хотят на время отключить работу какого-либо кода.

Начало раунда заново

Жизни мы уменьшили, а раунд не начали заново. Наш смайлик будет продолжать двигаться вниз, а жизни его будут продолжать уменьшаться с каждым тактом игрового цикла.

Чтобы этого избежать нам просто необходимо перенести наш смайлик снова в верхнюю позицию. То есть координата Y (или top) рамки должна быть приравнена к 0. А в координату X давайте для большего интереса запишем случайное значение.

```
if pic_rect.top > height:                    # Если зашли за нижнюю границу экрана
    lives -= 1                               # Уменьшаем количество жизней на 1
    pic_rect.y = 0                           # Поднимаем смайлик вверх
    pic_rect.x = random.randint(0, width)    # Случайное положение по горизонтали
    # run = False
    # print('Game Over')
```

Исправленный конец игры

Ну и наконец, переделаем логику конца игры так, чтобы она кончалась только тогда, когда количество жизней будет равно 0. А в этом нам поможет проверка if:

```
if pic_rect.top > height:                    # Если зашли за нижнюю границу экрана
    lives -= 1                               # Уменьшаем количество жизней на 1
    pic_rect.y = 0                           # Поднимаем смайлик вверх
    pic_rect.x = random.randint(0, width)    # Случайное положение по горизонтали
    if lives == 0:                           # Если жизней не осталось
        run = False                          # Завершаем игровой цикл
        print('Game Over')                   # Выводим надпись на экран
```

Перерыв

10 мин

Графика

Функция для вывода текста

Для того, чтобы игра была информативнее, текущее положение дел игрока обычно выносят на интерфейс. Давайте вынесем на интерфейс количество жизней. Для этого воспользуемся готовой функцией. Прямо копируем её в наш код до игрового цикла:

```
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 картинки с текстом
    text_rect.center = (x,y)                       # Переносим текст в координаты
    screen.blit(text_image, text_rect)             # Рисуем текст на экране
```

Функция для вывода текста

```
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 картинки с текстом
    text_rect.center = (x,y)                       # Переносим текст в координаты
    screen.blit(text_image, text_rect)             # Рисуем текст на экране
```

Функция принимает сразу 6 параметров! В них:

- screen — это наш экран, на котором всё и выводится
- text — текст, который хотим вывести
- size — размер текста
- x,y — координаты положения текста на экране
- Color — цвет текста

Функция для вывода текста

Так как кадр обновляется каждый такт игрового цикла, то текст мы должны выводить внутри игрового цикла сразу после того, как заполнили экран цветом. Вызовем эту функцию в нужном месте:

```
screen.fill(CYAN)                                   # Заливка заднего фона
screen.blit(pic, pic_rect)                          # Отрисовываем смайлик
screen.blit(racket, racket_rect)                    # Отрисовываем ракетку
draw_text(screen, str(lives), 30, width//2, 30, WHITE)   # Отрисовываем текст
```

Важно! Для того, чтобы выводить цифры на экран, их нужно преобразовать к тексту с помощью конвертирующей функции `str()`. Можно соединить со строкой**: «Lives» + str(lives)**

Задний фон

Загружать картинку мы уже с вами умеем. Но для движущегося заднего фона нужна особенная бесшовная картинка. Возьмем её из наших ресурсов, скопируем в папку с проектом и подключим в коде:

```
pic = pygame.image.load('smile.png')     # Загружаем спрайт
pic_rect = pic.get_rect()                # Получаем рамку спрайта

bg = pygame.image.load('bg.jpg')         # Загружаем спрайт
bg_rect = bg.get_rect()                  # Получаем рамку спрайта
```

Задний фон

Для того, чтобы её отобразить, нам нужно вместо заливки добавить отрисовку этой картинки. Отрисовывается задний фон точно также как и другие объекты. Только важно разместить отрисовку этой картинки самой первой, потому что сверху на ней будут уже рисоваться наши объекты:

```
# screen.fill(CYAN)  # Заливка заднего фона (закоментировали, больше не нужна)
screen.blit(bg,bg_rect)     # Отрисовываем картинку для заднего фона
```

Анимируем задний фон

На самом деле размеры экрана и картинки подобраны так, что мы можем перемещать картинку в левую сторону ровно до середины. А когда достигнет середины, перенести вправо, чтобы снова начать перемещать в левую сторону. Но игроку будет казаться, что задний фон плывет бесконечно влево, потому что эти переходы заметны не будут.

Сначала добавим постоянное движение влево в игровом цикле:

```
pic_rect.x += speedx
pic_rect.y += speedy

bg_rect.x -= 2                          # Фон плывет влево
```

Анимируем задний фон

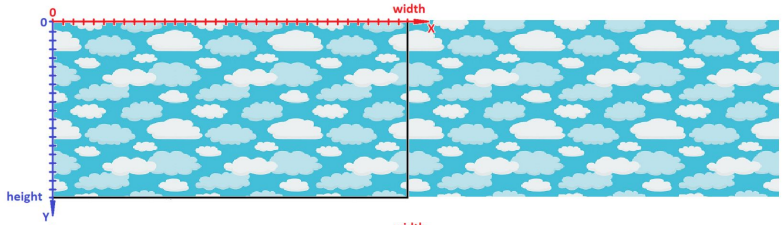
Затем, когда он сдвинется влево ровно на ширину экрана, мы перенесем его координату X снова в левый нижний угол (то есть приравняем к 0):

```
bg_rect.x -= 2              # Фон плывет влево
if bg_rect.x <= -width:     # Как только сдвинулись влево на ширину экрана
    bg_rect.x = 0
```

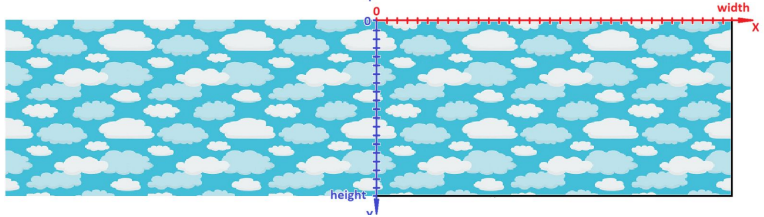
Анимируем задний фон

Как это происходит в действительности:

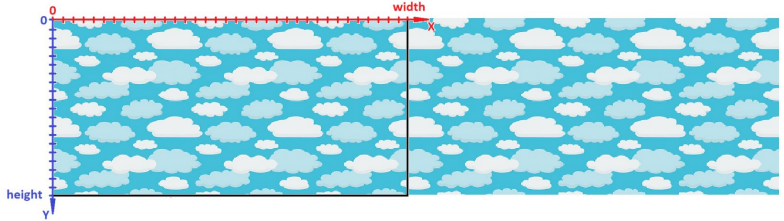
Начало движения



Достигли центра



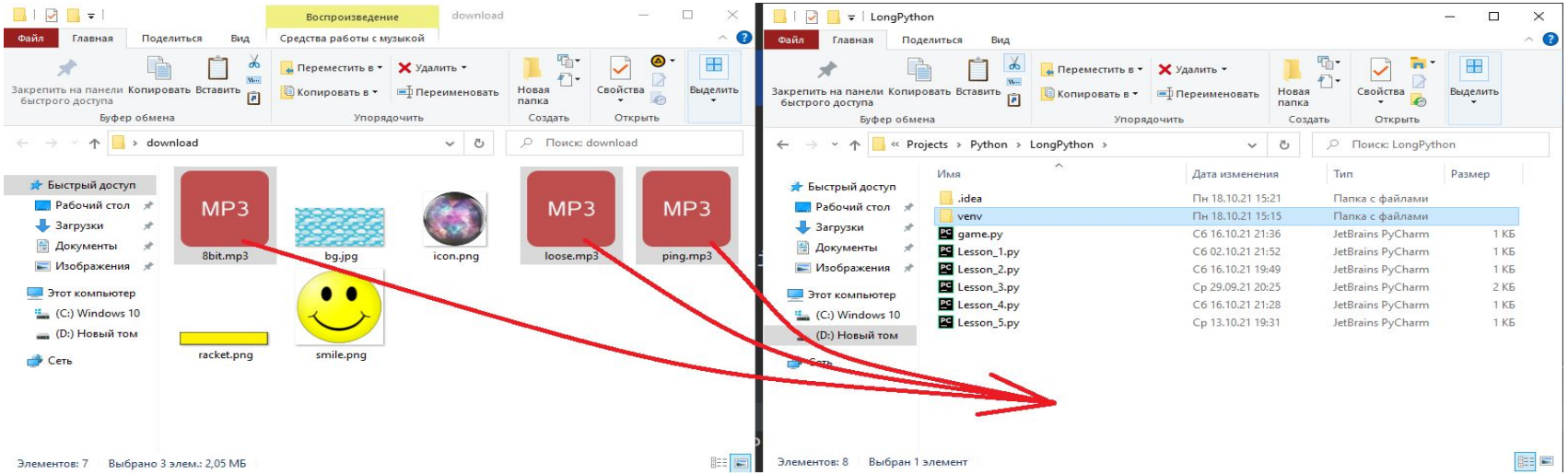
Перенесли в стартовое положение



Звуки и музыка

После, добавляем в код. Для этого нам нужно уже обратиться к другому разделу pygame. **pygame.mixer** Ниже представлен код, который сохранит звуки в разных переменных:

```
ping = pygame.mixer.Sound('ping.mp3')     # Звук отскока
loose = pygame.mixer.Sound('loose.mp3')   # Звук проигрыша
```



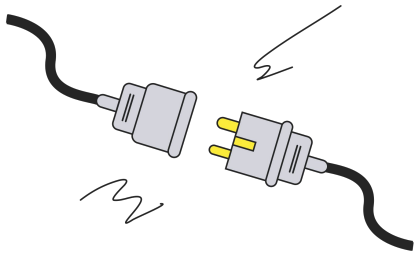
Добавляем звук отскока

Теперь, когда звуки хранятся в отдельных переменных, мы можем запросто их воспроизвести, вызвав команду **.play()** у любого звука. Главное это сделать в нужном месте! Давайте добавим соответствующую строчку кода в тех местах, где у нас шарик отскакивает от стенки:

```
if pic_rect.right > width or pic_rect.left < 0:  # Если прав. или лев. граница
    speedx = -speedx
    pi_rect.top < 0                              # Если верх. граница
    speedy = -speedy
    ping.play()
if pic_rect.colliderect(racket_rect):            # Если ракетка
    speedy = -speedy
    ping.play()ng.play()
if pic
```

Добавляем звук потери жизни

```
if pic_rect.top > height:                    # Если зашли за нижнюю границу экрана
    lives -= 1                               # Уменьшаем количество жизней на 1
    loose.play()                             # Звук проигрыша
    pic_rect.y = 0                           # Поднимаем смайлик вверх
    pic_rect.x = random.randint(0, width)    # Случайное положение по горизонтали
```



Добавляем фоновую музыку

С фоновой музыкой всё попроще. Мы её добавляем всего один раз, причем делаем это до игрового цикла. А потом просто устанавливаем параметр, благодаря которому она играет постоянно. Можно кстати также регулировать её громкость. До игрового цикла добавьте следующие команды:

```
pygame.mixer.music.load('8bit.mp3')      # Загружаем музыку
pygame.mixer.music.set_volume(0.1)       # Громкость 10%
pygame.mixer.music.play(-1)              # Бесконечный повтор
```

Можно теперь запустить и проверить результат! Наш шаблон игры в арканоид готов!

Полный код программы

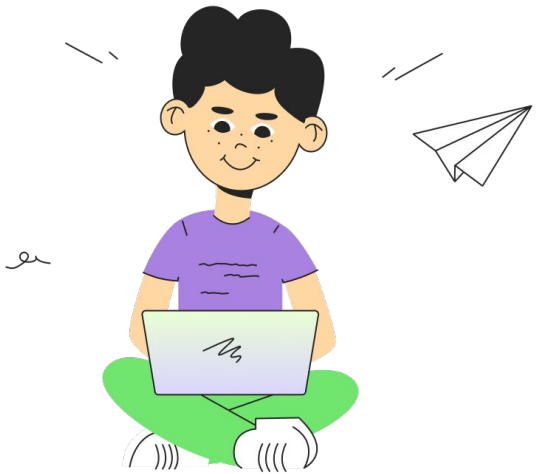
Сюда не влазит, поэтому посмотреть можно по ссылке :)

Итоги

- Познакомились с реализацией конца игры
- Научились контролировать жизни игрока
- Реализовали вывод жизней на интерфейс
- Добавили задний фон и анимировали его
- Добавили звуки и музыку в игру

На следующем занятии:

- Начнем создавать новую игру
- Познакомимся с новым подходом в программировании — объектно-ориентированным
- Добавим класс игрока
- Реализуем движение игрока в любую сторону



Немного повторим

Что нужно, чтобы остановить игровой цикл?

Для чего нужна команда pass?

В какую сторону будет двигаться спрайт, если выполним следующую команду:

```
bg_rect.x -= 2
```

В чем отличие во включении музыки и звуков в pygame?

Ваши вопросы

Спасибо
за внимание

Домашнее задание

Заполни, пожалуйста,
форму обратной связи по уроку

Напоминание для преподавателя

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