Бесплатная электронная книга

учусь pygame

Free unaffiliated eBook created from
Stack Overflow contributors.

#pygame

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pygame

It is an unofficial and free pygame ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pygame.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с pygame

замечания

Pygame - это оболочка Python для SDL - кросс-платформенная библиотека C для управления мультимедиа - написанная Питом Шиннером. Это означает, что с помощью pygame вы можете записывать видеоигры или другие мультимедийные приложения на Python, которые будут работать без изменений на любых поддерживаемых платформах SDL (Windows, Unix, Mac, beOS и других).

В этом разделе представлен обзор того, что такое pygame, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в pygame и ссылки на связанные темы. Поскольку Документация для pygame нова, вам может потребоваться создать начальные версии этих связанных тем.

Версии

Версия	==========> Дата выпуска
Pygame 1.9.0	========>	1 августа 2009
Pygame 1.8.1	========>	30 июля 2008
Pygame 1.8.0	========>	29 марта 2008
Pygame 1.7.1	========>	16 августа 2005
Pygame 1.6.2	========>	-
Pygame 1.6	=========>	23 октября 2003
Pygame 1.5	=========> 30 мая 2002
Pygame 1.4	=========>	30 января 2002
Pygame 1.3	=========>	19 декабря 2001
Pygame 1.2	=========>	4 сентября 2001
Pygame 1.1	=========>	23 июня 2001
Pygame 1.0	=========>	5 апреля 2001
Pygame 0.9	=========>	13 февраля 2001
Pygame 0.5	=========>	6 января 14, 2001
Pygame 0.4	=========>	14 декабря 2000
Pygame 0.3	=========>	20 ноября 2000
Pygame 0.2	=========> 3 ноября 2000
Pygame 0.1	=========>	28 октября 2000

Examples

Импорт и инициализация

Каждый модуль должен быть импортирован, а pygame не является исключением. Хотя нам нужно вызвать функцию `pygame.init()` чтобы все импортированные модули в pygame были правильно инициализированы. Если мы забудем это, некоторые модули не сработают. Функция также возвращает кортеж всех успешных и неудачных инициализаций (это не приведет к возникновению ошибки, если модуль не может инициализировать).

```
import pygame
successes, failures = pygame.init()
print("{0} successes and {1} failures".format(successes, failures))
```

Создание предметов первой необходимости

Нам также нужно создать дисплей. Pygame уже создал (скрытый) дисплей, поэтому все, что нам нужно сделать, это установить режим отображения (в этом примере мы только установили разрешение). Также рекомендуется создавать часы, чтобы убедиться, что наша программа обновляется с фиксированной скоростью (в противном случае она будет работать с разной скоростью в зависимости от скорости работы компьютера).

```
screen = pygame.display.set_mode((720, 480))  # Notice the tuple! It's not 2 arguments.
clock = pygame.time.Clock()
FPS = 60  # This variable will define how many frames we update per second.
```

Для более удобного чтения в нашем коде мы создадим две цветовые константы, которые представляют собой набор красного, зеленого и синего (RGB). Значения от 0 (без света) до 255 (полный свет).

```
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

В pygame мы обычно используем *Surface* для представления внешнего вида объекта и *Rect* (прямоугольник) для представления положения объекта. А *Поверхность* похожа на чистый лист бумаги, который содержит цвета или изображения. Если вы создаете класс, вы должны называть атрибуты *image* и *rect,* так как многие функции будут искать и использовать эти атрибуты. Такие классы выиграют, `pygame.sprite.Sprite` класс `pygame.sprite.Sprite` по причинам, которые вы можете прочитать здесь .

```
rect = pygame.Rect((0, 0), (32, 32))  # First tuple is position, second is size.
image = pygame.Surface((32, 32))  # The tuple represent size.
image.fill(WHITE)  # We fill our surface with a nice white color (by default black).
```

Игровой цикл

Теперь у нас есть все, что нужно для нашего игрового цикла. Это цикл, который будет запускаться для всей игры, где мы обрабатываем события и обновляем экран и позиции наших объектов.

Сначала мы убедимся, что наш цикл выполняется при заданном *FPS* . Мы определили *FPS* и создали наши часы в начале программы. Следующий код гарантирует, что наша программа будет спать достаточно времени, чтобы наш цикл повторял количество, которое мы определили для нашего *FPS* . В этом примере 60 раз в секунду.

```
clock.tick(FPS)
```

Затем мы будем обрабатывать события. Событие - это, в основном, действие пользователя, например, перемещение мыши или нажатие клавиши. Pygame зарегистрирует все эти события в очереди, которую мы получаем, вызывая `pygame.event.get()` . Мы можем перебрать это и проверить, есть ли какое-то событие, с которым мы хотели бы справиться. События имеют атрибут *типа,* который мы можем проверить против констант в модуле pygame, чтобы определить, какой тип события он имеет.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # The user pressed the close button in the top corner of
the window.
        quit()
        # Close the program. Other methods like 'raise SystemExit' or 'sys.exit()'.
        # Calling 'pygame.quit()' won't close the program! It will just uninitialize the
modules.
```

Мы также можем проверить `if event.type == pygame.KEYDOWN` чтобы увидеть, `if event.type == pygame.KEYDOWN` ли пользователь клавишу вниз. В этом случае событие имеет *ключ* атрибута, который мы можем проверить, чтобы увидеть, какой ключ он представляет.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        quit()
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_w:
            rect.move_ip(0, -2)  # Changes the rect's position.
        elif event.key == pygame.K_s:
            rect.move_ip(0, 2)
        elif event.key == pygame.K_a:
            rect.move_ip(-2, 0)
        elif event.key == pygame.K_d:
            rect.move_ip(2, 0)
```

Теперь нам нужно отобразить наш образ. Сначала мы можем очистить экран от предыдущего рендеринга. Мы делаем это, заполняя весь экран черным (удалите код, чтобы

узнать, почему мы хотим его очистить). Затем нам нужно *разжечь* наш *образ* на экране. Blitting по существу означает копирование *изображения* на другую поверхность (в нашем случае на экран). Наконец, мы *переворачиваем* или *обновляем* экран.

Когда мы бьемся, мы фактически ничего не показываем пользователю. Представьте это как компьютер с одной стороны, а пользователь - с другой. Компьютер рисует (*смеется*) на его стороне экрана, *переворачивает* его к пользователю, а затем повторяет.

```
screen.fill(BLACK)
screen.blit(image, rect)
pygame.display.update()  # Or 'pygame.display.flip()'.
```

Теперь у нас есть базовая игра! Довольно скучно, да, но основные вещи есть! Объединив это с вашим текущим знанием Python, вы сможете создать что-то потрясающее.

Полный код

```
import pygame
successes, failures = pygame.init()
print("{0} successes and {1} failures".format(successes, failures))


screen = pygame.display.set_mode((720, 480))
clock = pygame.time.Clock()
FPS = 60  # Frames per second.

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
# RED = (255, 0, 0), GREEN = (0, 255, 0), BLUE = (0, 0, 255).

rect = pygame.Rect((0, 0), (32, 32))
image = pygame.Surface((32, 32))
image .fill(WHITE)

while True:
    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                rect.move_ip(0, -2)
            elif event.key == pygame.K_s:
                rect.move_ip(0, 2)
            elif event.key == pygame.K_a:
                rect.move_ip(-2, 0)
            elif event.key == pygame.K_d:
                rect.move_ip(2, 0)

    screen.fill(BLACK)
    screen.blit(image, rect)
    pygame.display.update()  # Or pygame.display.flip()
```

Немного улучшенная игровая механика

Обратите внимание, что программа проверяет, когда мы нажимаем клавишу, а не когда удерживаем клавишу нажатой. Чтобы исправить это, мы могли бы ввести переменную *скорости* . Мы можем создать класс игрока, чтобы он был более организованным. Чтобы избежать зависания кадра (если мы изменим FPS на 30, объекты будут двигаться на половину скорости), мы вводим зависящее от времени движение, передавая время между тиками на наши подвижные объекты.

```
import pygame

successes, failures = pygame.init()
print("Initializing pygame: {0} successes and {1} failures.".format(successes, failures))

screen = pygame.display.set_mode((720, 480))
clock = pygame.time.Clock()
FPS = 60

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)


class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((32, 32))
        self.image.fill(WHITE)
        self.rect = self.image.get_rect()  # Get rect of some size as 'image'.
        self.velocity = [0, 0]

    def update(self):
        self.rect.move_ip(*self.velocity)


player = Player()
running = True
while running:
    dt = clock.tick(FPS) / 1000  # Returns milliseconds between each call to 'tick'. The
convert time to seconds.
    screen.fill(BLACK)  # Fill the screen with background color.

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                player.velocity[1] = -200 * dt  # 200 pixels per second
            elif event.key == pygame.K_s:
                player.velocity[1] = 200 * dt
            elif event.key == pygame.K_a:
                player.velocity[0] = -200 * dt
            elif event.key == pygame.K_d:
                player.velocity[0] = 200 * dt
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_w or event.key == pygame.K_s:
                player.velocity[1] = 0
            elif event.key == pygame.K_a or event.key == pygame.K_d:
```

```
                player.velocity[0] = 0

    player.update()

    screen.blit(player.image, player.rect)
    pygame.display.update()  # Or pygame.display.flip()

print("Exited the game loop. Game will quit...")
quit()  # Not actually necessary since the script will exit anyway.
```

Есть еще много вещей, которые следует улучшить об этом коде. Я бы рекомендовал вам прочитать учебник pygame и этот доклад Ричарда Джонса для более глубокого изучения.

Установка pygame

На окнах

1. Перейдите на страницу http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame - неофициальный сайт, содержащий двоичные файлы с открытым исходным кодом python для официального дистрибутива CPython от *Christoph Gohlke* .
2. Загрузите соответствующий файл pygame `.whl` соответствии с установленной версией python. (Файл называется чем-то вроде `pygame – <pygame version> – <python version> – win32.whl`)
3. Бежать

   ```
   pip install your-pygame-package.whl
   ```

   внутри вашего терминала, bash или consol.
   Примечание: если `pip` не найден в `PATH` попробуйте запустить `python -m pip install your-pygame-package.whl`
4. Проверьте, можно ли импортировать pygame в качестве модуля python

   ```
   import pygame
   ```

   Если вы не получили сообщение об ошибке, вы правильно установили pygame на свой компьютер :)

В Linux

1. Откройте терминал и запустите

   ```
   sudo apt-get install python-pygame
   ```

   Примечание. Это установит pygame для python2

2. Попробуйте импортировать pygame внутри

```
import pygame
```

Если вы не получили сообщение об ошибке, вы правильно установили pygame в своей Linux-системе :)

На macOS

Есть два способа установить его на mac:

Способ 1

Перейдите на страницу загрузки Pygame и загрузите программу установки mac. Запустите его, и он должен установить Pygame на ваш Mac.

Способ 2

Установить Homebrew :

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Затем используйте Homebrew для установки Python 2.7.12 и Pygame:

```
brew install python; brew install homebrew/python/pygame
```

Теперь запустите Python в вашем терминале и попробуйте `import pygame` . Если он ничего не говорит, он успешно установлен.

Импорт pygame и рисование на дисплее

Начиная

Для начала работы с Pygame вы должны сделать следующее:

```
import pygame
```

Это открывает окно размером 640,480 и сохраняет его в переменной, называемой экраном.

Настройка имени окна

Настройка имени окна pygame требует следующего синтаксиса:

```
pygame.display.set_caption('Name')
```

О экране

- Точка (0,0) находится в верхнем левом углу экрана.
- x координаты увеличиваются слева направо, y координаты увеличиваются сверху вниз. Это правильные координаты на карте декартовой плоскости положительны, а левая сторона отрицательна. Однако верхние боковые координаты на картезианской плоскости отрицательны сверху и положительны (**Примечание** : Это считается, если точки взяты из начала координат.)

Обновление экрана

Изменения, которые вы делаете на экране, например, заполняя его цветом или рисуя на нем, не отображаются сразу!
Итак, как это сделать?
Вы должны вызвать эту функцию:

```
pygame.display.update()
```

Цвета

Раскраска в pygame работает в режиме RGB.
Код для окраски:

```
color_Name = (r,g,b)
```

- R обозначает красный цвет.
- G означает зеленый цвет
- B обозначает синий цвет.
- Все три должны быть целыми числами от 0 до 255, причем 255 являются самыми яркими, а 0 - самыми темными

Рисование

1. Нарисовать линии

   ```
   pygame.draw.lines(screen, color, closed, pointlist, thickness)
   ```

2. Чтобы нарисовать прямоугольник

   ```
   pygame.draw.rect(screen, color, (x,y,width,height), thickness)
   ```

3. Нарисовать круг

```
pygame.draw.circle(screen, color, (x,y), radius, thickness)
```

Установка всего в цикл

Чтобы сделать цикл, используйте следующий код:

```
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Рисование прямоугольника в окне pygame (код)

```
import pygame
background_colour = (255,255,255) # White color
(width, height) = (300, 200) # Screen size
color=(0,0,0) #For retangle
screen = pygame.display.set_mode((width, height)) #Setting Screen
pygame.display.set_caption('Drawing') #Window Name
screen.fill(background_colour)#Fills white to screen
pygame.draw.rect(screen, color, (100,50,30,40), 1) #Drawing the rectangle
pygame.display.update()

#Loop
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Прочитайте Начало работы с pygame онлайн: https://riptutorial.com/ru/pygame/topic/3959/начало-работы-с-pygame

глава 2: Добавление фоновой музыки и звуковых эффектов

замечания

Попробуйте воспроизвести музыку в формате .wav вместо '.mp3'.In' .mp3 'музыка отстает.

Examples

Пример добавления музыки в pygame

```
import pygame
file = 'some.mp3'
pygame.init()
pygame.mixer.init()
pygame.mixer.music.load(file)
pygame.mixer.music.play(-1) # If the loops is -1 then the music will repeat indefinitely.
```

Пример добавления музыкального плейлиста в pygame

```
import pygame
import time

pygame.mixer.init()
pygame.display.init()

screen = pygame.display.set_mode ( ( 420 , 240 ) )

playlist = list()
playlist.append ( "music3.mp3" )
playlist.append ( "music2.mp3" )
playlist.append ( "music1.mp3" )

pygame.mixer.music.load ( playlist.pop() )  # Get the first track from the playlist
pygame.mixer.music.queue ( playlist.pop() ) # Queue the 2nd song
pygame.mixer.music.set_endevent ( pygame.USEREVENT )    # Setup the end track event
pygame.mixer.music.play()           # Play the music

running = True
while running:
   for event in pygame.event.get():
      if event.type == pygame.USEREVENT:    # A track has ended
         if len ( playlist ) > 0:       # If there are more tracks in the queue...
            pygame.mixer.music.queue ( playlist.pop() ) # Q
```

Прочитайте Добавление фоновой музыки и звуковых эффектов онлайн: https://riptutorial.com/ru/pygame/topic/7419/добавление-фоновой-музыки-и-звуковых-эффектов

глава 3: Обработка событий

Examples

Контур события

Pygame будет регистрировать все события от пользователя в очередь событий, которые могут быть получены с помощью кода `pygame.event.get()` . Каждый элемент в этой очереди является объектом `Event` и все они будут иметь `type` атрибута, который представляет собой целое число, представляющее, какое событие он имеет. В модуле pygame есть предопределенные целочисленные константы, представляющие тип. За исключением этого атрибута, события имеют разные атрибуты.

Постоянное имя	Атрибуты
УВОЛИТЬСЯ	никто
ActiveEvent	прибыль, состояние
KeyDown	unicode, key, mod
KeyUp	ключ, мода
MOUSEMOTION	pos, rel, кнопки
MOUSEBUTTONUP	pos, кнопка
MOUSEBUTTONDOWN	pos, кнопка
JOYAXISMOTION	радость, ось, значение
JOYBALLMOTION	радость, мяч, rel
JOYHATMOTION	радость, шляпа, ценность
JOYBUTTONUP	радость, кнопка
JOYBUTTONDOWN	радость, кнопка
VIDEORESIZE	размер, w, h
VIDEOEXPOSE	никто
USEREVENT	код

пример

Чтобы обрабатывать наши события, мы просто просматриваем очередь, проверяем, какой тип (с помощью предопределенных констант в модуле pygame), а затем выполняем некоторые действия. Этот код проверяет, нажал ли пользователь кнопку закрытия в верхнем углу дисплея, и если это так прекратит работу программы.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        # Close the program any way you want, or troll users who want to close your program.
        raise SystemExit
```

ВНИМАНИЕ : вы должны регулярно вызывать очередь событий при использовании pygame! Помимо выборки доступных событий, вызов очереди событий также является способом взаимодействия pygame с операционной системой. Если очередь событий не вызывается регулярно, ваша операционная система будет считать, что ваша программа больше не работает, и, возможно, она выглядит так, как если бы программа потерпела крах (в Windows это окно становится белым). Если вы не хотите ничего делать с событиями, вы должны вызвать `pygame.event.pump()` каждый цикл игры, чтобы pygame обрабатывал события внутри.

События на клавиатуре

В pygame есть два типа ключевых событий: `KEYDOWN` и `KEYUP` . Эти события имеют `key` атрибута, который является целым числом, представляющим ключ на клавиатуре. Модуль pygame имеет предопределенные целочисленные константы, представляющие все общие ключи. Константы называются с капиталом `K` , подчеркиванием и именем ключа. Например, `<-` называется `K_BACKSPACE` , `a` называется `K_a` а `F4` - `K_F4` .

пример

Этот код проверяет, нажал ли пользователь `w` , `a` , `s` или `d` .

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # Usually wise to be able to close your program.
        raise SystemExit
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_w:
            print("Player moved up!")
        elif event.key == pygame.K_a:
            print("Player moved left!")
        elif event.key == pygame.K_s:
            print("Player moved down!")
        elif event.key == pygame.K_d:
            print("Player moved right!")
```

Модификаторы

Для заглавных букв нет целочисленной константы. Вместо этого ключевые события имеют еще один атрибут `mod` , который является модификатором (`shift` , `ctrl` , `alt` и т. Д.) Одновременно нажатием клавиши. Атрибут `mod` - это целое число, представляющее нажатие модификатора. Каждое целое значение модификатора сохраняется в модуле pygame под именем `KMOD_` и их именем. Например, `сдвиг влево` называется `KMOD_LSHIFT` , `Tab` имеет имя `KMOD_TAB` а `Ctrl` - `KMOD_CTRL` .

пример

Этот код проверяет, нажал ли пользователь `a` , `сдвиг влево` + `a` или `Caps` + `a` .

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # It's still wise to be able to close your program.
        raise SystemExit
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_a:
            if event.mod == 0:  # No modifier.
                print("You pressed 'a'")
            elif event.mod == pygame.KMOD_LSHIFT or event.mod == pygame.KMOD_CAPS:
                print("You pressed 'A'")
            else:
                print("You pressed 'a' with another modifier than right shift or caps.")
```

События мыши

Существует три типа событий мыши в pygame `MOUSEMOTION` , `MOUSEBUTTONDOWN` и `MOUSEBUTTONUP` . Pygame зарегистрирует эти события, когда установлен режим отображения.

`MOUSEMOTION` принимается, когда пользователь перемещает свою или ее мышь на дисплей. Он имеет `buttons` атрибутов, `pos` и `rel` .

- *`buttons`* представляют собой кортеж, представляющий нажатие кнопок мыши (`left` , `mouse-wheel` , `right`) или нет.
- *`pos`* - абсолютное положение (`x` , `y`) курсора в пикселях.
- *`rel`* - позиция относительно предыдущей позиции (`rel_x` , `rel_y`) в пикселях.

`MOUSEBUTTONDOWN` и **`MOUSEBUTTONUP`** принимаются, когда пользователь нажимает или отпускает кнопку мыши. У них есть `button` атрибутов и `pos` .

- *`button`* представляет собой целое число, представляющее нажатую кнопку. `1` для левой кнопки, 2 для колесика мыши и `3` для правой кнопки.
- *`pos`* - это абсолютное положение мыши (`x` , `y`), когда пользователь нажал кнопку мыши.

пример

Вот краткий пример использования некоторых атрибутов каждого события мыши:

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # Close your program if the user wants to quit.
        raise SystemExit
    elif event.type == pygame.MOUSEMOTION:
        if event.rel[0] > 0:  # 'rel' is a tuple (x, y). 'rel[0]' is the x-value.
            print("You're moving the mouse to the right")
        elif event.rel[1] > 0:  # pygame start y=0 at the top of the display, so higher y-
values are further down.
            print("You're moving the mouse down")
    elif event.type == pygame.MOUSEBUTTONDOWN:
        if event.button == 1:
            print("You pressed the left mouse button")
        elif event.button == 3:
            print("You pressed the right mouse button")
    elif event.type == pygame.MOUSEBUTTONUP:
        print("You released the mouse button")
```

Поскольку в модуле pygame нет предопределенных констант для атрибута кнопки мыши, вот значения для каждого:

кнопка	Значение
Левую кнопку мыши	1
Кнопка мыши	2
Правая кнопка мыши	3
Колесо мыши прокручивается вверх	4
Колесо мыши прокручивается вниз	5

При прокрутке кнопки мыши будут генерироваться `pygame.MOUSEBUTTONDOWN` и `pygame.MOUSEBUTTONUP` .

Проверка состояния

Вы можете вызвать функции из модуля `pygame.key` и `pygame.mouse` для получения состояния клавиши и мыши. Тем не менее, это не рекомендуемый способ обработки событий в pygame, поскольку есть некоторые недостатки:

- Вы получите состояния при вызове функции, что означает, что вы можете пропустить события между вызовами, если пользователь быстро нажимает кнопки.
- Вы не можете определить порядок событий.

- Вам все равно нужно вызвать одну из функций события pygame для pygame, чтобы внутренне взаимодействовать с операционной системой, иначе он предупредит, что программа перестала отвечать на запросы. Функции, которые вы можете вызвать:
    - `pygame.event.get()` чтобы получить все события или типы событий (путем передачи типов в качестве аргумента) из очереди.
    - `pygame.event.poll()` чтобы получить одно событие из очереди.
    - `pygame.event.wait()` чтобы ждать одного события из очереди.
    - `pygame.event.clear()` чтобы очистить все события в очереди.
    - `pygame.event.pump()` чтобы позволить pygame обрабатывать внутренние действия (называется неявно указанными выше функциями).

События на клавиатуре

Ключевой модуль имеет функцию `pygame.key.get_pressed()` которая возвращает список состояний всех ключей. Список содержит `0` для всех клавиш, которые не нажаты, и `1` для всех нажатых клавиш. Его индекс в списке определяется константами в модуле pygame, все с префиксом `K_` и именем ключа.

```
pygame.event.pump()  # Allow pygame to handle internal actions.
key = pygame.key.get_pressed()
if key[pygame.K_a]:
    print("You pressed 'a'")
if key[pygame.K_F1]:
    print("You pressed 'F1'")
if key[pygame.K_LSHIFT]:
    print("You pressed 'left shift'")
if key[pygame.K_q]:  # Press 'q' to exit the program
    quit()
```

Если вы хотите проверить одно нажатие клавиши, а не удерживать клавишу, вы можете сохранить предыдущее состояние всех ключей во временной переменной и проверить, изменяется ли значение:

```
pygame.event.pump()  # Allow pygame to handle internal actions.
key = pygame.key.get_pressed()
if key[pygame.K_q] and not previous_key[pygame.K_q]:
    print("You pressed 'q'")
if key[pygame.K_p] and not previous_key[pygame.K_p]:
    print("You pressed 'p'")
previous_key = key
```

Оператор оценивает значение true только тогда, когда текущая клавиша нажата, а предыдущая клавиша не нажата. Чтобы проверить, освободил ли пользователь ключ, вам нужно переключить ключевое слово `not` (`if not key[pygame.K_q] and previous_key[pygame.K_q]`). Для правильной работы вам необходимо установить переменную `previous_key =`

`pygame.key.get_pressed()` перед игровым циклом, иначе вы получите `NameError` .

События мыши

Модуль мыши имеет функции, которые позволяют нам проверять и устанавливать положение мыши, а также проверять нажатые кнопки. Функция `pygame.mouse.get_pressed()` возвращает кортеж кортежа, представляющий, если `pygame.mouse.get_pressed()` кнопки мыши (левое, колесико мыши, справа) или нет.

```
pygame.event.pump()  # Allow pygame to handle internal actions.
mouse_pos = pygame.mouse.get_pos()
mouse_buttons = pygame.mouse.get_pressed()
if mouse_pos[0] > 100:
    pygame.mouse.set_pos(10, mouse_pos[1])  # Reset the mouse's x-position to 10.
    print("YOU SHALL NOT PASS!")
if mouse_buttons[2]:
    print("I'm right, right?")
if mouse_buttons[0]:  # Press left mouse button to exit.
    print("Program left")
    quit()
```

Прочитайте Обработка событий онлайн: https://riptutorial.com/ru/pygame/topic/5110/обработка-событий

глава 4: Основы

Examples

Рисование и базовая анимация

Эта программа рисует некоторые формы и « *привет мир!* 'и пусть изображение идет в каждый угол окна.

полный код:

```
import pygame,sys
from pygame.locals import *

pygame.init()

FPS = 30 #frames per second setting
fpsClock = pygame.time.Clock()

#set up the window
screen = pygame.display.set_mode((500,400), 0, 32)
pygame.display.set_caption('drawing')

#set up the colors
black = (  0,   0,   0)
white = (255, 255, 255)
red   = (255,   0,   0)
green = (  0, 255,   0)
blue  = (  0,   0, 255)

imageImg  = pygame.image.load('baddie.png')
imagex = 320
imagey = 220
direction = 'left'

fontObj = pygame.font.Font('freesansbold.ttf', 32)
text = fontObj.render('Hello World!', True, green, blue)
rect = text.get_rect()
rect.center = (200, 150)

# the main game loop
while True:
    screen.fill(white)

    # draw a green polygon onto the surface
    pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0,
106)))

    # draw some blue lines onto the surface
    pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
    pygame.draw.line(screen, blue, (120, 60), (60, 120))
    pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)
```

```
    # draw a blue circle onto the surface
    pygame.draw.circle(screen, blue, (300, 50), 100, 0)

    # draw a red ellipse onto the surface
    pygame.draw.ellipse(screen, red, (300, 250, 80,80), 1)

    # draw a red rectangle onto the surface
    pygame.draw.rect(screen,red, (200, 150, 100, 50))

    # draw the text onto the surface
    screen.blit(text, rect)

    if direction == 'right':
        imagex += 5
        if imagex == 320:
            direction = 'down'
    elif direction == 'down':
        imagey += 5
        if imagey == 220:
            direction = 'left'
    elif direction == 'left':
        imagex -= 5
        if imagex == 20:
           direction = 'up'
    elif direction == 'up':
        imagey -= 5
        if imagey == 20:
           direction = 'right'
    screen.blit(imageImg, (imagex, imagey))

for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()

pygame.display.update()
fpsClock.tick(FPS)
```

настройка pygame и окна:

```
import pygame,sys
from pygame.locals import *

pygame.init()

#set up the window
screen = pygame.display.set_mode((500,400), 0, 32)
pygame.display.set_caption('drawing')
```

рисуя белый фон:

В этой функции вы определяете цвет фона.

```
screen.fill(white)
```

рисунок зеленого полигона:

Здесь вы определяете поверхность дисплея, цвет и положение каждого угла многоугольника (координаты x и y), вы можете делать это по часовой стрелке и против часовой стрелки.

```
pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))
```

рисование синих линий:

В этой функции вы определяете поверхность дисплея, цвет, первую и последнюю точку и ширину линии (если вы не даете ширину, это всего лишь 1).

```
pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
pygame.draw.line(screen, blue, (120, 60), (60, 120))
pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)
```

рисуя синий круг:

В этой функции вы определяете поверхность дисплея, цвет, положение, радиус и ширину круга (если вы даете 0 для ширины, это простой круг).

```
pygame.draw.circle(screen, blue, (300, 50), 100, 0)
```

рисование эллипса:

В этой функции вы определяете поверхность дисплея, цвет, положение, горизонтальный размер и вертикальный размер и ширину.

```
pygame.draw.ellipse(screen, red, (300, 250, 80,80), 1)
```

рисование прямоугольника:

В этой функции вы определяете поверхность дисплея, цвет, положение и горизонтальный и вертикальный размер.

```
pygame.draw.rect(screen,red, (200, 150, 100, 50))
```

определение текста:

Сначала вы определяете тип и размер текста с помощью этой функции:

```
fontObj = pygame.font.Font('freesansbold.ttf', 32)
```

Затем вы определяете фактический текст, если текст выделен жирным шрифтом, цвет и, если хотите, цвет маркировки. Вы можете сделать это с помощью этой функции:

```
text = fontObj.render('Hello World!', True, green, blue)
```

Если вы хотите пометить свой текст, вы должны сказать pygame, что с помощью этой функции:

```
rect = text.get_rect()
```

И если вы хотите определить положение центра текста, вы можете сделать это с помощью этой функции:

```
rect.center = (200, 150)
```

составление текста:

Если вы определили маркировку и / или центр:

```
screen.blit(text, rect)
```

В противном случае вам необходимо определить положение текста, поэтому нарисуйте текст таким образом:

```
screen.blit(text, (100,50))
```

определение изображения:

Здесь вы определяете, какое изображение вы хотите использовать (если вы это сделаете, файл изображения должен находиться в том же каталоге, что и файл вашей программы), начальную позицию (x и y) и направление изображения.

```
image  = pygame.image.load('image.png')
baddiex = 320
baddiey = 220
direction = 'left'
```

анимация изображения:

С этой частью кода мы проверяем направление изображения, если оно достигло угла, если это так, измените направление, если нет, нарисуйте изображение на 5 пикселей дальше в том же направлении.

```
if direction == 'right':
    imagex += 5
    if imagex == 360:
        direction = 'down'
elif direction == 'down':
    imagey += 5
    if imagey == 260:
        direction = 'left'
elif direction == 'left':
    imagex -= 5
    if imagex == 20:
       direction = 'up'
elif direction == 'up':
    imagey -= 5
    if imagey == 20:
       direction = 'right'
screen.blit(imageImg, (imagex, imagey))
```

note: Мое изображение 20x20 пикселей, я использую `if imagex == 360:` **и** `if imagey == 260:` **потому что тогда мое изображение составляет 20 пикселей от края окна, как и другие 2 угла. Если ваше изображение имеет другой размер, вам, вероятно, придется изменить эти числа.**

проверка на выход:

Здесь мы проверяем, закрыли ли вы окно pygame, и если да, закройте окно, если вы не пишете это где-то в своей программе, вы, вероятно, не сможете закрыть окно.

```
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()
```

обновление экрана:

С помощью этой функции вы обновляете экран, чтобы все, что вы нарисовали, стало видимым.

```
pygame.display.update()
```

Настройка FPS:

С помощью этой функции вы скажете pygame, чтобы спать достаточно, чтобы ваша настройка FPS соблюдалась.

```
fpsClock.tick(FPS)
```

Использование с PIL

Когда вы должны использовать как PIL, так и Pygame, потому что отсутствуют функциональные возможности в обоих из них, вам нужен способ конвертировать между Pygame Surfaces и изображениями PIL, желательно, не записывая их на диск.

Для этого вы можете использовать функции «tostring» и «fromstring», предоставляемые в обеих библиотеках.

Преобразование из PIL в Pygame:

```
strFormat = 'RGBA'
raw_str = image.tostring("raw", strFormat)
surface = pygame.image.fromstring(raw_str, image.size, strFormat)
```

Преобразование из Pygame в PIL:

```
strFormat = 'RGBA'
raw_str = pygame.image.tostring(surface, strFormat, False)
image = Image.frombytes(strFormat, surface.get_size(), raw_str)
```

Прочитайте Основы онлайн: https://riptutorial.com/ru/pygame/topic/4196/основы

глава 5: Рисование на экране

Examples

рисование фигур, текста и изображений на экране с небольшой анимацией

Эта программа нарисует некоторые фигуры на дисплее, нарисует «привет мир!». в середине экрана, и пусть изображение направляется в каждый угол окна. Вы можете использовать любое изображение, которое вам нужно, но **вам нужно будет поместить файл изображения в тот же каталог, что и ваша программа.**

весь код:

```
import pygame, sys
from pygame.locals import *

pygame.init()

FPS = 30 #frames per second setting
fpsClock = pygame.time.Clock()

#set up the window
screen = pygame.display.set_mode((400, 300), 0, 32)
pygame.display.set_caption('animation')

#set up the colors
white = (255, 255, 255)
black = (  0,   0,   0)
green = (0, 255, 0)
blue = (0, 0, 180)
red   = (255,   0,   0)

image  = pygame.image.load('image.png')
imagex = 360
imagey = 260
direction = 'left'

# text setting
font_obj = pygame.font.Font('freesansbold.ttf', 32)
text_surface_obj = font_obj.render('Hello World!', True, GREEN, BLUE)
text_rect_obj = text_surface_obj.get_rect()
text_rectObj.center = (200, 150)

while True: # the main game loop
    screen.fill(WHITE)

    # draw a green polygon onto the surface
    pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0,
106)))
```

```
    # draw some blue lines onto the surface
    pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
    pygame.draw.line(screen, blue, (120, 60), (60, 120))
    pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)

    # draw a blue circle onto the surface
    pygame.draw.circle(screen, blue, (300, 50), 20, 0)

    # draw a red ellipse onto the surface
    pygame.draw.ellipse(screen, red, (100, 150, 40,80), 1)

    # draw a red rectangle onto the surface
    pygame.draw.rect(screen,red, (200, 150, 100, 50))

    # draw the text onto the surface
    screen.blit(text_surface_obj, text_rect_obj)


    #the animation of the image
    if direction == 'right':
        imagex += 5
        if imagex == 360:
            direction = 'down'
    elif direction == 'down':
        imagey += 5
        if imagey == 260:
            direction = 'left'
    elif direction == 'left':
        imagex -= 5
        if imagex == 20:
            direction = 'up'
    elif direction == 'up':
        imagey -= 5
        if imagey == 20:
            direction = 'right'
    screen.blit(image, (imagex, imagey))

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    pygame.display.update()
    fpsClock.tick(FPS)
```

рисуя белый фон:

```
screen.fill(white)
```

рисование полигона:

В этой функции вы определяете поверхность дисплея, цвет и положение каждого угла многоугольника, вы можете делать это по часовой стрелке и против часовой стрелки.

```
pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))
```

рисование линий:

Здесь вы определяете поверхность дисплея, цвет, первую и последнюю точку и ширину линии.

```
pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
pygame.draw.line(screen, blue, (120, 60), (60, 120))
pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)
```

рисование круга:

В этой функции вы определяете поверхность дисплея, цвет, положение, радиус и ширину круга (0 дает простой круг).

```
pygame.draw.circle(screen, blue, (300, 50), 20, 0)
```

рисование эллипса:

В этой функции вы определяете поверхность дисплея, цвет, положение, горизонтальный размер, вертикальный размер и ширину эллипса

```
pygame.draw.ellipse(screen, red, (100, 150, 40,80), 1)
```

рисование прямоугольника:

В этой функции вы определяете поверхность дисплея, цвет, положение и вертикальный и горизонтальный размер прямоугольника.

```
pygame.draw.rect(screen,red, (200, 150, 100, 50))
```

определение текста:

Сначала вы определяете тип и размер вашего текста, я использую базовый шрифт, который вы получаете с pygame.

```
font_obj = pygame.font.Font('freesansbold.ttf', 32)
```

Затем вы определяете фактический текст, если хотите, чтобы он полужирный или нет (True / False), цвет текста и, если вы хотите пометить свой текст, цвет маркировки.

```
text_surface_obj = font_obj.render('Hello World!', True, green, blue)
```

Если вы хотите пометить свой текст или хотите определить центр текста, вы должны сообщить pygame, что с помощью этой функции:

```
text_rect_obj = text_surface_obj.get_rect()
```

И после этого вы можете определить центр своего текста с помощью этой функции:

```
text_rect_obj.center = (200, 150)
```

составление текста:

Если вы отметили свой текст или определили центр, вы должны нарисовать текст следующим образом:

```
screen.blit(text_surface_obj, text_rectObj)
```

В противном случае вы рисуете текст, но вам нужно определить позицию, так что вы делаете так:

```
DISPLAYSURF.blit(textSurfaceObj, (100,50))
```

определение изображения:

Здесь вы определяете изображение, которое хотите использовать, начальную позицию (координаты x и y) и направление изображения.

```
image  = pygame.image.load('image.png')
imagex = 360
imagey = 260
direction = 'left'
```

анимация изображения:

Здесь вы проверяете направление изображения, если оно достигло угла, если это так измените направление, если нет, переместите его на 5 пикселей в одном направлении и снова нарисуйте изображение. Это то, что мы делаем с этой частью кода:

```
if direction == 'right':
    imagex += 5
    if imagex == 360:
        direction = 'down'
elif direction == 'down':
    imagey += 5
    if imagey == 260:
        direction = 'left'
elif direction == 'left':
    imagex -= 5
    if imagex == 20:
        direction = 'up'
elif direction == 'up':
    imagey -= 5
    if imagey == 20:
        direction = 'right'
screen.blit(image, (imagex, imagey))
```

примечание: мое изображение 20 на 20 пикселей, я использовал, `if imagex == 360` **и** `if imagey == 260`**: потому что тогда мое изображение будет 20 пикселей от края, если ваше изображение имеет другой размер, вам придется изменить числа ,**

если вы выходите из программы:

Здесь мы проверяем, закрыли ли вы окно своей программы.

```
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()
```

обновление дисплея:

Здесь вы сообщаете pygame, чтобы обновить дисплей, чтобы все, что вы нарисовали, появилось на экране.

```
pygame.display.update()
```

определение кадров в секунду:

Здесь вы можете сказать pygame, чтобы спать достаточно, чтобы обеспечить соответствие кадров в секунду.

```
fpsClock.tick(FPS)
```

Прочитайте Рисование на экране онлайн: https://riptutorial.com/ru/pygame/topic/6287/

рисование-на-экране

глава 6: Рисование на экране

Синтаксис

- pygame.draw.rect (Поверхность, цвет, Rect, width = 0)
- pygame.draw.polygon (Поверхность, цвет, список точек, ширина = 0)
- pygame.draw.circle (Поверхность, цвет, позиция, радиус, ширина = 0)
- pygame.draw.ellipse (Поверхность, цвет, Rect, width = 0)
- pygame.draw.arc (Поверхность, цвет, Rect, start_angle, stop_angle, width = 1)
- pygame.draw.line (Поверхность, цвет, start_pos, end_pos, width = 1)
- pygame.draw.lines (Поверхность, цвет, закрытая, pointlist, width = 1)
- pygame.draw.aaline (Поверхность, цвет, startpos, endpos, blend = 1)
- pygame.draw.aalines (Surface, color, closed, pointlist, blend = 1)

параметры

параметры	подробности
поверхность	Поверхность, чтобы нарисовать фигуру.
цвет	Целая последовательность 3 или 4, представляющая красный, зеленый и синий (и альфа), каждое значение находится в диапазоне от 0 до 255.
Rect	Прямоугольная область, на которую будет нарисована фигура.
ширина	Ширина линий. Форма будет заполнена, если ширина = 0.
PointList	Список произвольного количества точек / вершин в пикселях (x, y).
позиция	Положение центра круга в пикселях (x, y).
радиус	Радиус кругов в пикселях.
закрыто	Если true, линия между последней и первой точкой будет нарисована, закрывая фигуру.
смешаться = 1	Если true, оттенки будут смешиваться с существующими пиксельными оттенками, а не переписывать их.
start_angle	Начальный угол дуги в радианах.
stop_angle	Конечный угол дуги, в радианах.

параметры	подробности
start_pos	Исходное положение линии в пикселях.
end_pos	Конечное положение линии, в пикселях

Examples

Рисование с помощью модуля вытягивания

Pygame имеет модуль `pygame.draw` , который содержит функции, которые могут рисовать фигуры непосредственно на поверхности.

функция	Описание
pygame.draw.rect	нарисовать прямоугольную форму
pygame.draw.polygon	нарисуйте фигуру с любым количеством сторон
pygame.draw.circle	нарисуйте круг вокруг точки
pygame.draw.ellipse	нарисуйте круглую форму внутри прямоугольника
pygame.draw.arc	нарисуйте частичную часть эллипса
pygame.draw.line	нарисуйте отрезок прямой линии
pygame.draw.lines	нарисовать несколько смежных отрезков
pygame.draw.aaline	нарисовать тонкие сглаженные линии
pygame.draw.aalines	нарисовать связанную последовательность сглаженных линий

Как использовать модуль

Чтобы использовать модуль, вам сначала нужно правильно импортировать и инициализировать pygame и установить режим отображения. Удобно определять цветовые константы заранее, делая ваш код более читабельным и красивым. Все функции принимают поверх поверхности, аргумент цвета и позиции, который представляет собой либо pygame Rect, либо 2-элементную последовательность integer / float (`pygame.draw.circle` будет принимать только целые числа из-за неопределенных причин).

пример

В приведенном ниже коде будут показаны все различные функции, способы их использования и способы их просмотра. Мы инициализируем pygame и определяем некоторые константы перед примерами.

```
import pygame
from math import pi
pygame.init()

screen = pygame.display.set_mode((100, 100))
WHITE = pygame.Color(255, 255, 255)
RED = pygame.Color(255, 0, 0)
```

Черный цвет - это цвет по умолчанию для поверхности и представляет собой часть поверхности, на которую не нарисовано. Параметры каждой функции описаны ниже в **параметрах** .

Rect

```
size = (50, 50)

rect_border = pygame.Surface(size)  # Create a Surface to draw on.
pygame.draw.rect(rect_border, RED, rect_border.get_rect(), 10)  # Draw on it.

rect_filled = pygame.Surface(size)
pygame.draw.rect(rect_filled, RED, rect_filled.get_rect())
```

многоугольник

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]  # The corner points of the polygon.

polygon = pygame.Surface(size)
pygame.draw.polygon(polygon, RED, points, 10)

polygon_filled = pygame.Surface(size)
pygame.draw.polygon(polygon_filled, RED, points)
```

Круг

```
size = (50, 50)
radius = 25
```

```
circle = pygame.Surface(size)
pygame.draw.circle(circle, RED, (radius, radius), radius, 10)  # Position is the center of the
circle.

circle_filled = pygame.Surface(size)
pygame.draw.circle(circle_filled, RED, (radius, radius), radius)
```

Отверстия являются неудачным следствием алгоритма рисования pygame.

Эллипс

```
size = (50, 25)  # Minimize it's height so it doesn't look like a circle.

ellipse = pygame.Surface(size)
pygame.draw.ellipse(ellipse, RED, ellipse.get_rect(), 5)

ellipse_filled = pygame.Surface(size)
pygame.draw.ellipse(ellipse_filled, RED, ellipse.get_rect())
```

Отверстия являются неудачным следствием алгоритма рисования pygame.

дуга

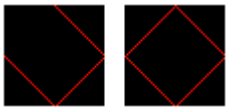
```
size = (50, 50)

arc = pygame.Surface(size)
pygame.draw.arc(arc, RED, arc.get_rect(), 0, pi)  # 0 to pi is 180° creating a half circle.
```

Линия

```
size = (50, 50)

line = pygame.Surface(size)
pygame.draw.line(line, RED, (0, 0), (50, 50))  # Start at topleft and ends at bottomright.
```

линии

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]

lines = pygame.Surface(size)
pygame.draw.lines(lines, RED, False, points)

lines_closed = pygame.Surface(size)
pygame.draw.lines(lines_closed, RED, True, points)
```

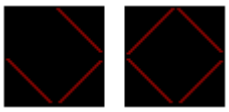


Сглаженная линия

```
size = (50, 50)

antialiased_line = pygame.Surface(size)
pygame.draw.aaline(antialiased_line, RED, (0, 0), (50, 50))
```

Сглаженные линии

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]

antialiased_lines = pygame.Surface(size)
pygame.draw.aalines(antialiased_lines, RED, False, points)

antialiased_lines_closed = pygame.Surface(size)
pygame.draw.aalines(antialiased_lines_closed, RED, True, points)
```



Попробуйте

Попробуйте сами: скопируйте один из фрагментов кода выше и код ниже в пустой файл, измените *изображение* имени на имя поверхности, которую хотите разбить и поэкспериментировать.

```
import pygame
from math import pi
```

```
pygame.init()

screen = pygame.display.set_mode((100, 100))
WHITE = pygame.Color(255, 255, 255)
RED = pygame.Color(255, 0, 0)

# But code snippet here

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()

    screen.blit(image, (25, 25))
    pygame.display.update()
```

Поверхности

В pygame вы обычно используете Surfaces для представления внешнего вида объектов и Rectangles для представления своих позиций. А Поверхность похожа на чистый лист бумаги, который содержит цвета или изображения. Существует два способа создания Поверхности: пустая с нуля или загрузка изображения.

Создание поверхности

Чтобы создать Поверхность, вам нужен минимум ее размер, который представляет собой 2-элементную целую последовательность ширины и высоты, представляющую размер в пикселях.

Вы также можете передавать дополнительные аргументы при создании Поверхности для управления глубиной, масками и дополнительными функциями в качестве пикселя альфа и / или создания изображения в видеопамяти. Однако это выходит за рамки этого примера.

```
size = width, height = (32, 32)
empty_surface = pygame.Surface(size)
```

Вы можете использовать модуль `pygame.draw` для рисования фигур на поверхности или заливки цветом, вызвав метод `fill(color)` поверхности `fill(color)` . *Цвет* аргумента представляет собой целую последовательность из 3 или 4 элементов или объект `pygame.Color` .

Загрузить изображение

Чаще всего вы хотели бы использовать свои собственные изображения в игре (называемые спрайтами). Создание поверхности с вашим изображением так же просто, как:

```
my_image = pygame.image.load(path_to_image)
```

Путь к изображению может быть относительным или абсолютным. Для повышения производительности обычно целесообразно преобразовать изображение в тот же формат пикселей, что и экран. Это можно сделать, вызвав метод Surface метода `convert()` , например:

```
my_image = pygame.image.load(path_to_image).convert()
```

Если ваше изображение содержит прозрачность (альфа-значения), вы просто вызываете метод `convert_alpha()` :

```
my_image = pygame.image.load(path_to_image).convert_alpha()
```

блиттинг

Для того, чтобы их можно было отображать, на экране должны быть блистерны. Blitting по существу означает копирование пикселей с одной поверхности на другую (экран также является поверхностью). Вам также необходимо передать позицию поверхности, которая должна быть 2-элементной целочисленной последовательностью или объектом Rect. Поверхность Поверхности будет помещена в позицию.

```
screen.blit(my_image, (0, 0))
pygame.display.update()  # or pygame.display.flip()
```

Можно смешать с другими поверхностями, чем экран. Чтобы показать, что было на экране, вам нужно вызвать `pygame.display.update()` или `pygame.display.flip()` .

прозрачность

В pygame есть виды 3 прозрачности: colorkeys, Surface alphas и per-pixel alphas.

Colorkeys

Делает цвет полностью прозрачным или более точно, делая цвет просто не blit. Если у вас есть изображение с черным прямоугольником внутри, вы можете установить colorkey, чтобы черный цвет не блистал.

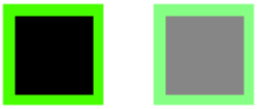
```
BLACK = (0, 0, 0)
my_image.set_colorkey(BLACK)  # Black colors will not be blit.
```

Поверхность может иметь только один colorkey. Установка другого colorkey перезапишет предыдущий. Colorkeys не может иметь разные значения альфа, он может только сделать цвет невидимым.

Поверхностные альфы

Делает всю поверхность прозрачной по альфа-значению. С помощью этого метода вы можете иметь разные значения альфа, но это повлияет на всю поверхность.

```
my_image.set_alpha(100)  # 0 is fully transparent and 255 fully opaque.
```



Пер-пиксельная альфа

Делает каждый пиксель в прозрачной поверхности индивидуальным альфа-значением. Это дает вам максимальную свободу и гибкость, но также является самым медленным методом. Этот метод также требует, чтобы поверхность создавалась как пиксельная альфа-поверхность, а цветовые аргументы должны содержать четвертое целое число.

```
size = width, height = (32, 32)
my_image = pygame.Surface(size, pygame.SRCALPHA)  # Creates an empty per-pixel alpha Surface.
```

Теперь поверхность будет рисовать прозрачность, если цвет содержит четвертое значение альфа.

```
BLUE = (0, 0, 255, 255)
pygame.draw.rect(my_image, BLUE, my_image.get_rect(), 10)
```

В отличие от других поверхностей, этот цвет по умолчанию не будет черным, но прозрачным. Вот почему черный прямоугольник посередине исчезает.

Объединить colorkey и Surface alpha

Colorkeys и Surface alphas можно комбинировать, но альфа-пиксель не может. Это может быть полезно, если вы не хотите более медленную производительность для пиксельной поверхности.

```
purple_image.set_colorkey(BLACK)
purple_image.set_alpha(50)
```

Полный код

Скопируйте это в пустой файл и запустите его. Нажмите клавиши 1, 2, 3 или 4, чтобы отобразить изображения. Нажмите 2, 3 или 4 раза, чтобы сделать их более непрозрачными.

```
import pygame
pygame.init()

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255, 50)  # This color contains an extra integer. It's the alpha value.
PURPLE = (255, 0, 255)

screen = pygame.display.set_mode((200, 325))
screen.fill(WHITE)  # Make the background white. Remember that the screen is a Surface!
clock = pygame.time.Clock()

size = (50, 50)
red_image = pygame.Surface(size)
green_image = pygame.Surface(size)
blue_image = pygame.Surface(size, pygame.SRCALPHA)  # Contains a flag telling pygame that the
Surface is per-pixel alpha
purple_image = pygame.Surface(size)

red_image.set_colorkey(BLACK)
green_image.set_alpha(50)
# For the 'blue_image' it's the alpha value of the color that's been drawn to each pixel that
determines transparency.
purple_image.set_colorkey(BLACK)
purple_image.set_alpha(50)

pygame.draw.rect(red_image, RED, red_image.get_rect(), 10)
pygame.draw.rect(green_image, GREEN, green_image.get_rect(), 10)
pygame.draw.rect(blue_image, BLUE, blue_image.get_rect(), 10)
pygame.draw.rect(purple_image, PURPLE, purple_image.get_rect(), 10)

while True:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_1:
                screen.blit(red_image, (75, 25))
            elif event.key == pygame.K_2:
                screen.blit(green_image, (75, 100))
            elif event.key == pygame.K_3:
                screen.blit(blue_image, (75, 175))
            elif event.key == pygame.K_4:
                screen.blit(purple_image, (75, 250))
```

```
    pygame.display.update()
```

Прочитайте Рисование на экране онлайн: https://riptutorial.com/ru/pygame/topic/7079/рисование-на-экране

глава 7: Создание окна pygame

замечания

Если вы хотите иметь другие цвета в качестве фона, то назовите новую переменную типа `red = (255,0,0)` и измените `display.fill(black)` на `display.fill(red)` . Вы можете создавать цвета, сохраняя их в переменной и проверяя их значения RGB из Интернета.

Examples

Создание окна pygame

```
import pygame

background_colour = (255,255,255) # For the background color of your window
(width, height) = (300, 200) # Dimension of the window

screen = pygame.display.set_mode((width, height)) # Making of the screen
pygame.display.set_caption('Tutorial 1') # Name for the window
screen.fill(background_colour) #This syntax fills the background colour

pygame.display.flip()

running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Прочитайте Создание окна pygame онлайн: https://riptutorial.com/ru/pygame/topic/6477/создание-окна-pygame

глава 8: Создание окна в pygame - pygame.display.set_mode ()

Синтаксис

- pygame.display.set_mode (разрешение = (0,0), flags = 0, depth = 0) # Возвращает pygame.Surface, представляющий окно на экране
- flags = pygame.FULLSCREEN | pygame.OPENGL # Флаги можно комбинировать с помощью "|" (побитовое ИЛИ или «труба»).

параметры

параметр	объяснений
разрешающая способность	пара чисел, представляющих ширину и высоту окна
флаги	дополнительные параметры, которые изменяют тип окна - см. «Примечания» для доступных флагов
глубина	количество бит, используемых для цвета

замечания

- Возможные значения аргументов `flag` :

флаг	описание
pygame.FULLSCREEN	окно полноэкранного
pygame.RESIZABLE	окно изменено по размеру
pygame.NOFRAME	Окно не имеет границ или элементов управления
pygame.DOUBLEBUF	использовать двойной буфер - рекомендуется для `HWSURFACE` или `OPENGL`
pygame.HWSURFACE	окно аппаратно ускорено, возможно только в сочетании с `FULLSCREEN`
pygame.OPENGL	окно визуализируется OpenGL

Другие замечания:

- Pygame может обрабатывать только одно окно одновременно. Создание второго окна путем вызова `pygame.display.set_mode((x,y))` второй раз закроет первое окно.
- Изменение аргумента `depths` почти никогда не требуется - pygame выберет лучший из них сам по себе. В случае установки глубины, не поддерживаемой системой, pygame будет эмулировать эту глубину, которая может быть очень медленной.
- Вещи, которые нарисованы на поверхности, возвращенные `pygame.display.set_mode()` , сразу не отображаются на экране - сначала нужно перевернуть `pygame.display.update()` используя `pygame.display.update()` или `pygame.display.flip()` .

Examples

Создать окно pygame

Это создает окно в полноэкранном режиме размером 500x500 пикселей:

```
pygame.init()
screen = pygame.display.set_mode((500, 500), pygame.FULLSCREEN)
```

`screen` теперь отображается в окне на экране; это объект pygame.Surface. Все, что должно быть видимым для пользователя, должно быть нарисовано на нем с помощью `screen.blit` .

Прочитайте Создание окна в pygame - pygame.display.set_mode () онлайн: https://riptutorial.com/ru/pygame/topic/6442/создание-окна-в-pygame---pygame-display-set-mode---

глава 9: Создание простого окна pygame

Examples

Полный код

```
import pygame

pygame.init()

WIDTH = 300
HEIGHT = 200
SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption('My Game')

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 255)

SCREEN.fill(RED)
pygame.display.flip()

is_running = True
while is_running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            is_running = False

pygame.quit()
```

Импорт и инициализация pygame

Как и в случае с любым модулем в python, нам нужно импортировать pygame:

```
import pygame
```

Затем мы инициализируем все импортированные модули pygame:

```
pygame.init()
```

Это используется для инициализации всех модулей pygame. Без этого модули не будут работать

Определение констант

Затем мы определяем некоторые константы:

```
WIDTH = 300
HEIGHT = 200
SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))
```

Константы `WIDTH` и `HEIGHT` используются для создания окна, которое имеет ширину 300 пикселей и высоту 200 пикселей. Функция, используемая в `SCREEN` , `pygame.display.set_mode((WIDTH, HEIGHT))` , установит режим отображения и вернет объект Surface . Обратите внимание, как параметры для этой функции являются константами `WIDTH` и `HEIGHT` определенными ранее.

Установка имени окна

Затем мы используем эту функцию для изменения имени окна в My Game:

```
pygame.display.set_caption('My Game')
```

Определение цветов

Затем мы определяем 6 цветов, которые можно использовать в нашем окне:

```
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 255)
```

При определении цветов мы помещаем в 3 значения, которые находятся в диапазоне от 0 до 255. Класс pygame.Color обычно проходит по этому формату:

```
COLOUR = (r, g, b, a)
```

Если параметр r задает красное значение цвета, параметр g задает зеленое значение цвета, а параметр b задает синее значение цвета. Параметр a задает альфа-значение цвета.

Затем мы приводим эту команду:

```
SCREEN.fill(RED)
```

Это функция pygame.Surface.fill, которая заполняет объект Surface, наш экран, красным цветом.

Использование pygame.display.flip ()

Затем мы используем эту функцию

```
pygame.display.flip()
```

Это в основном делает все, что мы нарисовали на экране. Поверхность становится видимой и обновляет содержимое всего дисплея. Без этой строки пользователь ничего не увидит на своем экране pygame.

Игровой цикл

Следующие несколько строк - это так называемый «игровой цикл».

Чтобы начать это, мы создаем переменную и делаем ее правдой:

```
is_running = True
```

Чтобы мы могли начать цикл while:

```
while is_running:
```

который будет работать на протяжении всей игры.

В самой базовой форме pygame имеет «события», которые принимают пользовательский ввод, например, нажатие кнопки или щелчок мышью. Pygame обрабатывает эти события через очередь событий. Мы можем получить эти события из очереди событий с помощью этого цикла:

```
for event in pygame.event.get():
```

Что в основном проходит через список событий, нашу очередь событий. Это следующие две строки:

```
if event.type == pygame.QUIT:
    is_running = False
```

Это сделает так, что когда пользователь нажмет кнопку выхода в верхнем углу, произойдет событие с типом `pygame.QUIT` .

Затем это завершает цикл while, так как `is_running` теперь `False` и сценарий переходит к последней строке:

```
pygame.quit()
```

Какой неинициализирует модули pygame.

Прочитайте Создание простого окна pygame онлайн:
https://riptutorial.com/ru/pygame/topic/6597/создание-простого-окна-pygame

кредиты

S. No	Главы	Contributors
1	Начало работы с pygame	Community, elegent, Inazuma, Nearoo, Ni., numbermaniac, svs, Ted Klein Bergman, White Shadow
2	Добавление фоновой музыки и звуковых эффектов	Hikaryu, White Shadow
3	Обработка событий	elegent, Mikhail V, Nearoo, Ted Klein Bergman
4	Основы	Ni., numbermaniac, svs, Ted Klein Bergman
5	Рисование на экране	svs
6	Создание окна pygame	Rishi Malhotra, White Shadow
7	Создание окна в pygame - pygame.display.set_mode ()	Nearoo
8	Создание простого окна pygame	ModoUnreal