**Аннотация**

*На этом занятии мы начинаем блок, посвященный созданию игр c помощью библиотеки pygame.*

**Введение**

В этом модуле вам предстоит написать игру.

С одной стороны, игры — дело несерьезное. Конференции и конкурсы программных проектов очень редко рассматривают такие работы.

С другой, разработка игр — это сложная задача, включающая в себя много процессов и требующая использовать различные технологии.

Можно утверждать, что любая **игра — это Объекты + Графика + Физика + ...**

Для того чтобы было удобно программировать все это многообразие, используют фреймворки. Одним из самых известных и популярных фреймворков для разработки игр на Python является .

**Pygame — самая популярная библиотека для разработки игр**

Разработка pygame началась в 2000 году Питером Шиннерсом. Тогда он активно программировал на Си и познакомился с Python (версии 1.5.2) и библиотекой [SDL](http://www.libsdl.org/). В то время она была очень популярна, на ней разрабатывались сотни игр, в том числе коммерческих, и идея подключить ее к Python показалась ему очень интересной.



Начиная примерно с 2004-2005 года, pygame поддерживается и развивается сообществом свободного программного обеспечения.

Pygame включает в себя все, что необходимо для разработки игр: удобную работу с графикой (например, поддержку спрайтов), c методами-детекторами столкновений, звук и многое другое.

**Установка pygame**

Благодаря утилите pip, установить pygame очень просто:

pip install pygame

**Hello, Pygame**

Для начала работы нужно подключить модуль pygame, вызвать функцию init() и создать холст.

Например:

import pygame

if \_\_name\_\_ == '\_\_main\_\_':

pygame.init()

size = width, height = 800, 600

screen = pygame.display.set\_mode(size)

И теперь можно свободно рисовать, вернее, формировать кадр на холсте с именем screen.

Что же всегда волновало авторов компьютерных игр? Правильно, их очень заботила проблема сделать движения героев, смены кадров и прочие анимационные вещи плавными, то есть создать эффект анимации.

Для того чтобы движение объектов было плавным, pygame использует двойную буферизацию. Пока пользователь смотрит на один кадр, следующий уже строится в памяти, при этом у пользователя ничего не меняется. Как только новый кадр нарисован, он помещается на видимый экран. Это происходит очень быстро, и пользователь не видит процесса рисования, который на большой скорости обновления выглядит как мерцание.

Для пользователя смена изображений происходит мгновенно.

Функция flip() как раз и выполняет эту смену. Поэтому после отрисовки кадра обязательно требуется команда:

pygame.display.flip()

Давайте дождёмся того момента, когда пользователь закроет окно. Сделаем это так:

while pygame.event.wait().type != pygame.QUIT:

pass

Мы подробно разберем условие этого цикла на следующем занятии.

После этого, в конце работы программы нужно вызвать функцию quit() модуля pygame.

Таким образом, самая простая корректная программа на pygame выглядит так:

import pygame

if \_\_name\_\_ == '\_\_main\_\_':

# инициализация Pygame:

pygame.init()

# размеры окна:

size = width, height = 800, 600

# screen — холст, на котором нужно рисовать:

screen = pygame.display.set\_mode(size)

# формирование кадра:

# команды рисования на холсте

# ...

# ...

# смена (отрисовка) кадра:

pygame.display.flip()

# ожидание закрытия окна:

while pygame.event.wait().type != pygame.QUIT:

pass

# завершение работы:

pygame.quit()

Теперь вынесем рисование в отдельную функцию:

def draw(screen):

screen.fill((0, 0, 0))

font = pygame.font.Font(None, 50)

text = font.render("Hello, Pygame!", True, (100, 255, 100))

text\_x = width // 2 - text.get\_width() // 2

text\_y = height // 2 - text.get\_height() // 2

text\_w = text.get\_width()

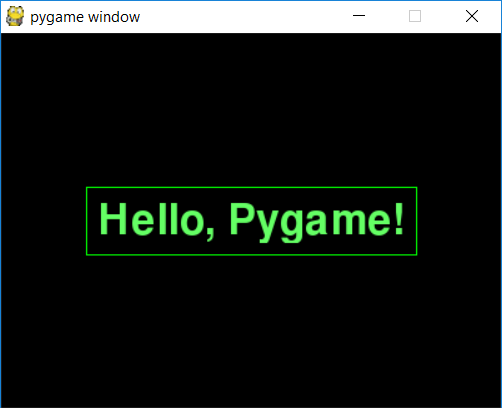
text\_h = text.get\_height()

screen.blit(text, (text\_x, text\_y))

pygame.draw.rect(screen, (0, 255, 0), (text\_x - 10, text\_y - 10,

text\_w + 20, text\_h + 20), 1)

Теперь, если добавить вызов этой функции в основной цикл, мы получим результат:



**Тренировочное задание.** Попробуйте воспроизвести самостоятельно приведенный пример и напишите программу Hello, pygame.

Сама функция draw() содержит несколько новых команд, мы их рассмотрим в следующих разделах.

**Рисование**

Конечно, при оценке игры в первую очередь смотрят на ее графику. Обычно в играх используют готовые изображения, которые загружаются из дополнительных файлов. Но на этом занятии мы принципиально не будем использовать изображения. Наша основная задача «привыкнуть» к экранным координатам и изучить возможности модуля [draw](https://www.pygame.org/docs/ref/draw.html#comment_pygame_draw_rect).

**Цвета**

Давайте вспомним, как устроены цвета в компьютерном мире.

Рисование на экране в конечном итоге — это появление **пикселей** (точек) разного цвета. Практически все команды рисования принимают цвет в качестве параметра.

Цвета каждого пикселя на мониторах формируется тремя световыми источниками:

* Красным
* Синим
* Зеленым

Каждая компонента-источник может иметь 256 градаций интенсивности (0 — совсем не горит, 255 — горит очень ярко).

Например,

**(0, 0, 0)** — это черный цвет,

**(255, 255, 255)** — белый цвет.

Вообще, все цвета, имеющие одинаковые значения компонент — серые разной интенсивности.

**(255, 0, 0)** — красный,

**(255, 100, 100)** — светло-красный.

Для цветов в pygame используется отдельный модуль [Color](https://www.pygame.org/docs/ref/color.html).

Цвета в pygame — это объекты типа Color. Их могут создавать конструкторы:

Color(name) # строкой, например, "yellow"

Color(r, g, b, a) # красный, зеленый, синий и прозрачность

Color(rgbvalue)

Последний конструктор может принимать как число, соответствующее цвету, так и строки в HTML-формате (#rrggbbaa или #rrggbb).

Таким образом, для создания цветов можно использовать такой код:

lightred = pygame.Color(255, 255, 100)

darkgreen = pygame.Color('#008000')

yellow = pygame.Color('yellow')

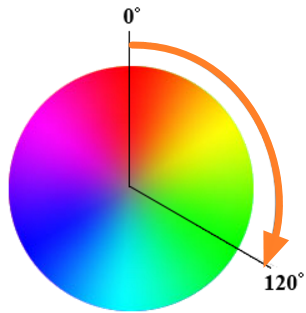
В Интернете есть много [инструментов по RGB-представлению цветов](https://yandex.ru/search/?text=RGB%20online&lr=213).

Полезно знать, что существуют и другие форматы представления цвета (не только RGB).

Очень важным является HSV формат (Hue, Saturation, Value — тон, насыщенность, значение) или HSB (Hue, Saturation, Brightness — тон, насыщенность, яркость).

Возможные значения в диапазонах: H = [0, 360], S = [0, 100], V = [0, 100].

Яркость и насыщенность задаются в процентах, оттенок — положением на цветовом круге:



Использование такого формата позволяет удобно менять «освещенность», не меняя оттенка. Например, квадрат с «тенью»



можно нарисовать так:

def draw\_square(screen):

color = pygame.Color(50, 150, 50)

# рисуем "тень"

pygame.draw.rect(screen, color,

(20, 20, 100, 100), 0)

hsv = color.hsva

# увеличиваем параметр Value, который влияет на яркость

color.hsva = (hsv[0], hsv[1], hsv[2] + 30, hsv[3])

# рисуем сам объект

pygame.draw.rect(screen, color, (10, 10, 100, 100), 0)

Но если сложные преобразования цветов не нужны, можно обойтись и триплетами.

Функции модуля draw принимают кортежи в качестве цвета, например:

# нарисуем красный квадрат

pygame.draw.rect(screen, (255, 0, 0), (10, 10, 100, 100), 0)

**Тренировочное задание.** Выведите надпись «Hello, Pygame» в программе из предыдущего раздела [цветом Яндекса](https://yandex.ru/search/?text=%D1%86%D0%B2%D0%B5%D1%82%20%D0%AF%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%B0&lr=213).

Несмотря на то, что цвета можно создать с полупрозрачностью, рисование фигур модулем draw не поддерживает прозрачность.

Конечно, прозрачность в Pygame использовать возможно, но мы подробно поговорим об этом позже.

**Система координат**

Экран pygame устроен обычно для компьютерных графических систем (и необычно для математиков). Точка (0;0) — в верхнем левом углу, а ось Y направлена вниз.

**Surface, Rect**

В pygame рисование происходит на Surface. Хоть это слово дословно переводится как **Поверхность**, мы чаще будем использовать термин **Холст**. Объект именно этого класса возвращается методом set\_mode() модуля display.

Мы будем достаточно редко вызывать методы этого объекта, чаще передавать его в функции рисования модуля draw, но один метод используется чаще других.

При помощи метода fill() можно залить весь холст. Ему передается цвет заливки. Например,

screen.fill((255, 255, 255))

зальет весь холст белым.

В этот метод можно передать и прямоугольник, тогда будет залита только область, заданная прямоугольником.

Для представления прямоугольников в pygame используется класс Rect. Объекты этого класса можно создать при помощи конструкторов:

Rect(left, top, width, height)

Rect((left, top), (width, height))

Rect(rect)

Последний конструктор принимает в себя другой прямоугольник. При этом создается копия передаваемого прямоугольника.

Таким образом, для рисования красного квадрата на зеленом фоне можно написать:

screen.fill((0, 255, 0))

screen.fill(pygame.Color('red'), pygame.Rect(10, 10, 60, 60))

При программировании компьютерной графики прямоугольники встречаются очень часто, поэтому в pygame все функции, принимающие прямоугольники в качестве параметров, могут принимать и кортеж из четырех чисел. При этом прямоугольник будет создан «на лету».

Например, последнюю строчку можно упростить:

screen.fill(pygame.Color('red'), (10, 10, 60, 60))

В pygame не предусмотрена специальная команда рисования пикселя, но ее можно сэмулировать методом fill().

Например, код

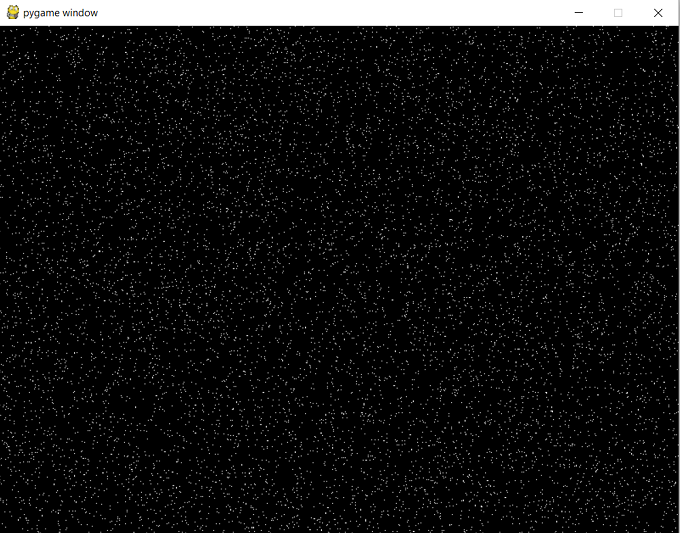
for i in range(10000):

screen.fill(pygame.Color('white'),

(random.random() \* width,

random.random() \* height, 1, 1))

выведет на черный экран 10 тысяч случайных белых точек.



**Функции рисования**

Основные функции рисования фигур расположены в модуле [draw](https://www.pygame.org/docs/ref/draw.html#comment_pygame_draw_rect).

Функции очень похожи друг на друга. Практически всем функциям передается Surface-объект, цвет (или кортеж значений, его задающих) и параметры.

Для того чтобы нарисовать **линию**, применяют функцию

line(Surface, color, start\_pos, end\_pos, width=1)

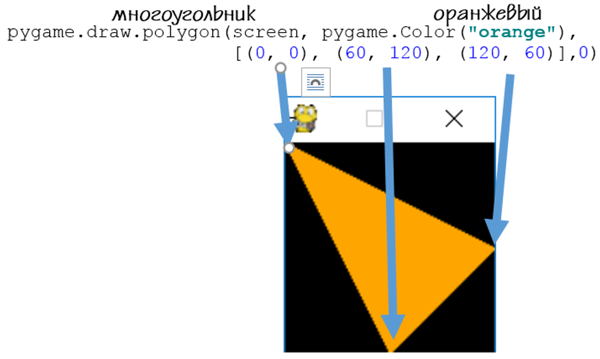
Она принимает два кортежа, содержащие координаты начальной и конечной точек. Последний параметр задает толщину линии.

Для рисования **прямоугольника**, стороны которого параллельны осям координат, используют функцию

rect(Surface, color, Rect, width=0)

Если последний параметр не задан (тогда он по умолчанию равен нулю), рисуется залитый прямоугольник, иначе — только рамка. Заметим, что закрашенный прямоугольник быстрее рисовать методом [fill()](https://www.pygame.org/docs/ref/surface.html#pygame.Surface.fill) холста. Вторым параметром метод fill принимает прямоугольник (Rect) (или кортеж, которым он задается).

Для рисования **многоугольника (полигона)** функции polygon() необходимо передать список его вершин.



Разберитесь по [документации](https://www.pygame.org/docs/ref/draw.html) с функциями рисования окружности, эллипса и дуг самостоятельно.

Давайте перейдем к практике.