[Посилання на офіційний сайт Pygame](https://www.pygame.org/docs/)

[Посилання на редактор коду VSC в онлайн форматі, тільки написання коду і коментарів](https://vscode.dev/)

1. Тут починаємо писати конспект.

pygame.display

◦Этот модуль предлагает контроль над отображением pygame. Pygame имеет одну поверхность отображения, которая либо содержится в окне, либо работает в полноэкранном режиме.

◦Создав дисплей, вы относитесь к нему как к обычной поверхности. Изменения не сразу видны на экране; вы должны выбрать одну из двух функций перелистывания, чтобы обновить фактический дисплей.

◦Начало отображения, где x = 0 и y = 0, находится в верхнем левом углу экрана. Обе оси положительно увеличиваются к нижней правой части экрана.

1. pygame.display.set\_caption()

◦Если дисплей имеет заголовок окна, эта функция изменит название окна. Некоторые системы поддерживают альтернативный более короткий заголовок, который можно использовать для свернутого отображения.

1. pygame.display.set\_icon()

◦Устанавливает значок времени выполнения, который система будет использовать для представления окна дисплея. Все окна по умолчанию используют простой логотип pygame для значка окна.

1. pygame.display.set\_mode()

Эта функция создаст поверхность дисплея. Передаваемые аргументы являются запросами на тип отображения. Фактическое созданное отображение будет наилучшим возможным соответствием, поддерживаемым системой.

Обратите внимание, что вызов этой функции неявно инициализирует pygame.display, если она не была инициализирована ранее.

1. pygame.display.flip()

Эта функция похожа на оптимизированную версию pygame.display.flip()для программных дисплеев. Это позволяет обновлять только часть экрана, а не всю область. Если аргумент не передан, он обновляет всю область Surface, например pygame.display.flip().

1. pygame.image

◦Модуль изображения содержит функции для загрузки и сохранения изображений, а также перевода поверхностей в форматы, используемые другими пакетами.

◦Обратите внимание, что класса Image нет; изображение загружается как объект Surface. Класс Surface позволяет манипулировать (рисовать линии, устанавливать пиксели, захватывать области и т. д.).

◦Модуль изображения является обязательной зависимостью pygame, но он лишь опционально поддерживает любые расширенные форматы файлов. При построении с полной поддержкой изображений pygame.image.load() функция может поддерживать следующие форматы.

1. pygame.image.load()

Загрузите изображение из источника файла. Вы можете передать имя файла, файловый объект Python или pathlib.Path.

Pygame автоматически определит тип изображения (например, GIF растровое) и создаст из данных новый объект Surface. В некоторых случаях необходимо знать расширение файла (например, GIF изображения должны заканчиваться на «.gif»). Если вы передаете необработанный файло подобный объект, вы также можете передать исходное имя файла в качестве аргумента подсказки имени.

Возвращенная поверхность будет содержать тот же цветовой формат, цветовой ключ и альфа-прозрачность, что и файл, из которого она была получена. Вы часто будете хотеть вызывать Surface.convert()без аргументов, чтобы создать копию, которая будет быстрее рисовать на экране.

Для альфа-прозрачности, как и в изображениях .png, используйте convert\_alpha() метод после загрузки, чтобы изображение имело прозрачность на пиксель.

pygame не всегда поддерживает все форматы изображений. Как минимум, он будет поддерживать несжатые файлы BMP. Если pygame.image.get\_extended() возвращает «True», вы сможете загружать большинство изображений (включая PNG, JPG и GIF).

Вы должны использовать os.path.join()для совместимости 1.

1. pygame.transform

◦Преобразование поверхности — это операция, которая перемещает или изменяет размер пикселей. Все эти функции принимают поверхность для работы и возвращают новую поверхность с результатами.

◦Некоторые преобразования считаются разрушительными. Это означает, что каждый раз, когда они выполняются, они теряют данные пикселей. Типичными примерами этого являются изменение размера и вращение. По этой причине лучше повторно преобразовать исходную поверхность, чем многократно преобразовывать изображение.

1. pygame.transform.scale()

◦Изменяет размер поверхности до нового размера, заданного как (ширина, высота). Это операция быстрого масштабирования, которая не выполняет выборку результатов.

◦Можно использовать дополнительную целевую поверхность, а не создавать новую. Это быстрее, если вы хотите многократно масштабировать что-то. Однако целевая поверхность должна быть того же размера, что и переданный размер (ширина, высота). Также целевая поверхность должна быть того же формата.

1. pygame.surface.blit()

Рисует множество поверхностей на этой поверхности. В качестве входных данных он принимает последовательность, в которой каждый из элементов соответствует элементам . Ему нужна как минимум последовательность (источник, пункт назначения).blit()

1. pygame.surface.fill()

Залейте поверхность сплошным цветом. Если аргумент rect не указан, вся поверхность будет заполнена. Аргумент rect ограничит заливку определенной областью. Заливка также будет содержаться в области обрезки поверхности.

Аргументом цвета может быть либо RGB последовательность, либо RGBA последовательность, либо отображаемый индекс цвета. Если используется RGBA, альфа (часть RGBA) игнорируется, если только поверхность не использует альфу для каждого пикселя (поверхность имеет SRCALPHA флаг).

1. pygame.rect

◦Pygame использует объекты Rect для хранения прямоугольных областей и управления ими. Rect может быть создан из комбинации значений left, top, width и height. Rect также можно создавать из объектов python, которые уже являются Rect или имеют атрибут с именем «rect».

◦Любая функция pygame, которой требуется аргумент Rect, также принимает любое из этих значений для создания Rect. Это упрощает создание Rect на лету в качестве аргументов функций.

◦Функции Rect, которые изменяют положение или размер Rect, возвращают новую копию Rect с затронутыми изменениями. Оригинальный Rect не изменен. У некоторых методов есть альтернативная версия «на месте», которая возвращает None, но затрагивает исходный Rect. Эти методы «на месте» обозначаются суффиксом «ip».

1. pygame.rect.collidepoint()

◦**Возвращает true, если заданная точка находится внутри прямоугольника. Точка вдоль правого или нижнего края не считается находящейся внутри прямоугольника.**

1. **pygame.draw**

**◦Нарисуйте несколько простых фигур на поверхности. Эти функции будут работать для рендеринга в любой формат поверхности. Рендеринг на аппаратные поверхности будет медленнее, чем на обычные программные поверхности.**

**◦Большинство функций принимают аргумент ширины для представления размера обводки (толщины) вокруг края фигуры. Если ширина равна 0, форма будет заполнена (сплошной).**

**◦Все функции рисования учитывают область обрезки поверхности и будут ограничены этой областью. Функции возвращают прямоугольник, представляющий ограничивающую область измененных пикселей. Этот ограничивающий прямоугольник представляет собой «минимальную» ограничивающую рамку, которая окружает затронутую область.**

1. **pygame.draw.rect()**

**◦Рисует прямоугольник на заданной поверхности. Это всё что про него можно сказать.**

1. **pygame.font**

**Модуль шрифта позволяет отображать шрифты TrueType в новый объект Surface. Он принимает любой символ UCS-2 (от «u0001» до «uFFFF»). Этот модуль является необязательным и требует SDL\_ttf в качестве зависимости. Вы должны проверить это pygame.font модуль pygame для загрузки и рендеринга шрифтов доступен и инициализируется перед попыткой использования модуля.**

1. **pygame.font.Font.render()**

**Это создает новую поверхность с указанным текстом, отображаемым на ней. pygame не позволяет напрямую рисовать текст на существующей поверхности: вместо этого вы должны использовать Font.render()для создания изображения (поверхности) текста, а затем перенести это изображение на другую поверхность.**

1. **pygame.time**

**Время в pygame представлено в миллисекундах (1/1000 секунды). Большинство платформ имеют ограниченное временное разрешение около 10 миллисекунд. Это разрешение в миллисекундах задается TIMER\_RESOLUTION константой.**

1. **pygame.time.tick()**

**Этот метод следует вызывать один раз за кадр. Он подсчитает, сколько миллисекунд прошло с момента предыдущего вызова.**

**Если вы передадите необязательный аргумент частоты кадров, функция сделает задержку, чтобы игра работала медленнее, чем заданное число тиков в секунду. Это можно использовать для ограничения скорости выполнения игры. При вызове Clock.tick(40)один раз за кадр программа никогда не будет работать со скоростью более 40 кадров в секунду.**

**Обратите внимание, что эта функция использует функцию SDL\_Delay, которая не является точной на каждой платформе, но не использует много ресурсов ЦП. Используйте отметку \_busy\_loop , если вам нужен точный таймер, и вы не возражаете против пережевывания процессора.**

1. **pygame**

**набор модулей языка программирования Python, предназначенный для написания компьютерных игр и мультимедиа-приложений. Pygame базируется на мультимедийной библиотеке SDL. Изначально Pygame был написан Питом Шиннерсом.**