INSTRUCTION: HOW TO INSTALL PYGAME:

1. Go to your **TERMINAL (MAC)/ POWERSHELL (WINDOWS)**

   Keyboard Shortcut:

   † **Press Command + Space to open Spotlight, type 'Terminal', and press Return**

   † **Press Win + X, then press T**

2. **In your Terminal, type:**

   pip3 install pygame-ce

   # community edition of pygame, works better than traditional pygame

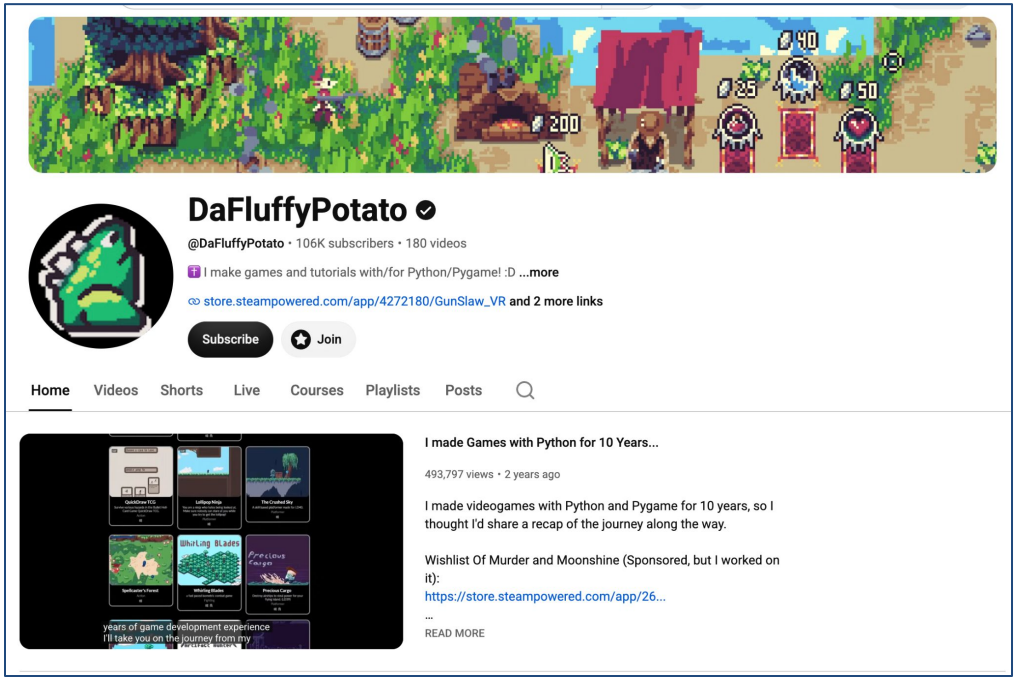
   If it is showing as below, then your installation is successful!

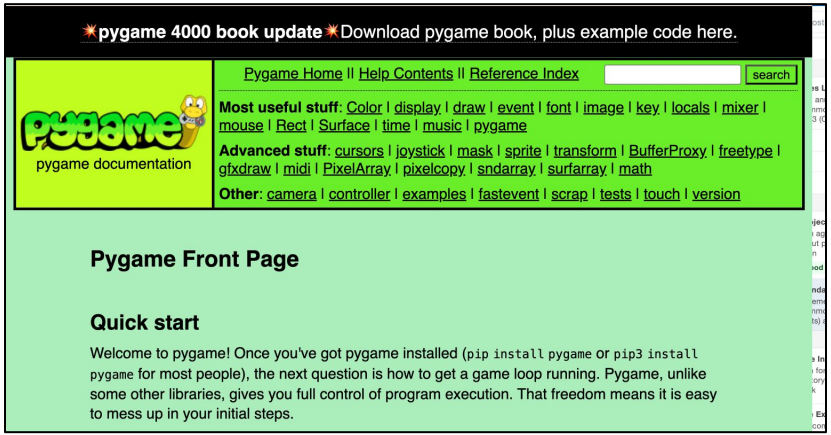
```
vamaridle-1802@Mac starterlab % pip3 install pygame-ce
Collecting pygame-ce
  Downloading pygame_ce-2.5.7-cp313-cp313-macosx_10_13_universal2.whl.metadata (11 kB)
Downloading pygame_ce-2.5.7-cp313-cp313-macosx_10_13_universal2.whl (17.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.2/17.2 MB 46.6 MB/s  0:00:00
Installing collected packages: pygame-ce
Successfully installed pygame-ce-2.5.7

[notice] A new release of pip is available: 26.0.1 -> 26.1.2
[notice] To update, run: pip3 install --upgrade pip
```

DOCUMENTATIONS | REFERENCES

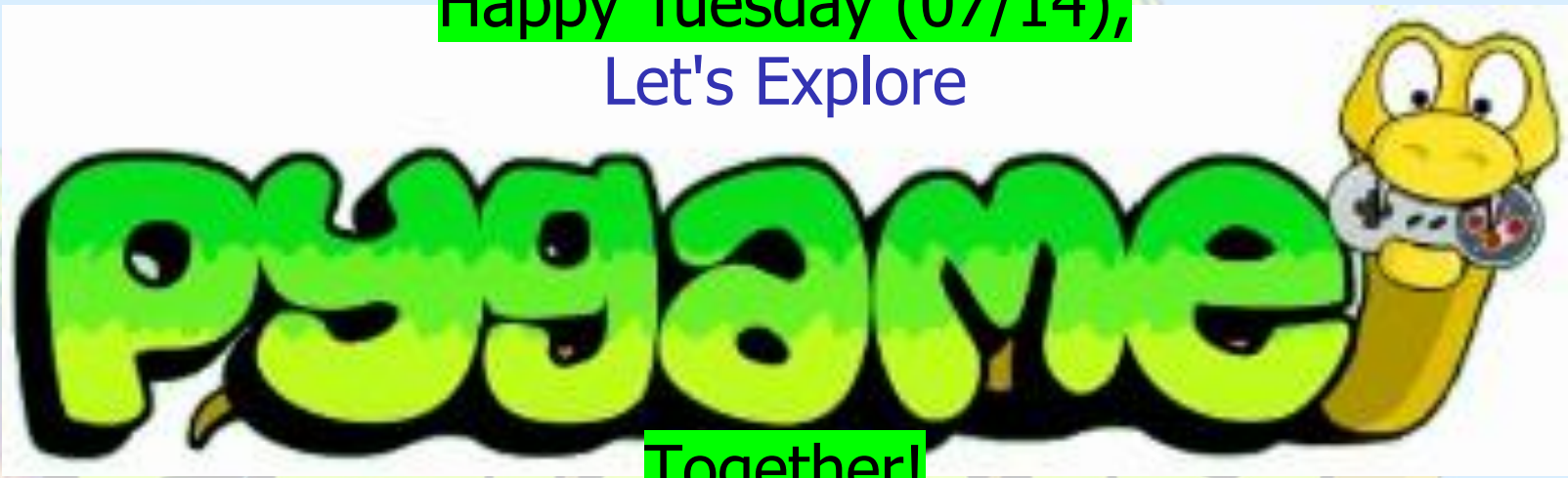
Pygame Documentation: https://www.pygame.org/docs/
Helpful Youtube channel: https://www.youtube.com/@DaFluffyPotato
Tutorials: https://youtu.be/blLLtdv4tvo?si=7xlgyedxp_HOPcHU





Happy Tuesday (07/14),

Let's Explore



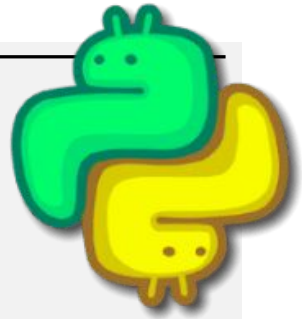
Together!

CMSC 19929 30

Creating Computer Games for Learning

https://youtube.com/shorts/LtJT6SzQM-A?si=9rMAkm1W8t-K7AhS

A simple Pygame example



```
1  import pygame
2
3  pygame.init()
4  screen = pygame.display.set_mode((640, 480))
5
6  color = [(0,0,0),(255,255,255)]
7  running = True
8
9  while running:
10   for event in pygame.event.get():
11      if event.type == pygame.QUIT:
12         running = False
13      if event.type == pygame.KEYDOWN:
14         color[0], color[1] = color[1],color[0]
15         screen.fill(color[0])
16         pygame.display.flip()
17
```

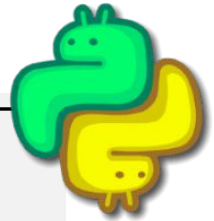
01-simple_pygame.py

What each element does: **Import** and **Initialize**

01-simple_pygame.py

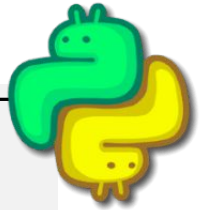
to **import** the Pygame module

to **set up** / initialize Pygame




```
1  import pygame
2
3  pygame.init()
4  screen = pygame.display.set_mode((640, 480))
5
6  color = [(0,0,0),(255,255,255)]
7  running = True
8
9  while running:
10   for event in pygame.event.get():
11      if event.type == pygame.QUIT:
12         running = False
13      if event.type == pygame.KEYDOWN:
14         color[0], color[1] = color[1],color[0]
15         screen.fill(color[0])
16         pygame.display.flip()
17
```

A simple Pygame example



```
1  import pygame
2
3  pygame.init()
4  screen = pygame.display.set_mode((640, 480))
5
6  color = [(0,0,0),(255,255,255)]
7  running = True
8
9  while running:
10   for event in pygame.event.get():
11      if event.type == pygame.QUIT:
12         running = False
13      if event.type == pygame.KEYDOWN:
14         color[0], color[1] = color[1],color[0]
15
16         screen.fill(color[0])
17         pygame.display.flip()
```

01-simple_pygame.py

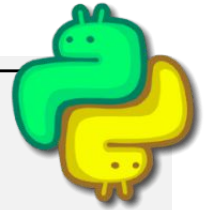
What each element does: **Setting Window & Screen**

```
screen = pygame.display.set_mode((640, 480))
```

initializes a **window** with dimensions 640 × 480 and returns the **screen object.**

Everything to be displayed needs to be drawn on the screen.

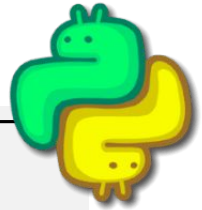
A simple Pygame example



```
1  import pygame
2
3  pygame.init()
4  screen = pygame.display.set_mode((640, 480))
5
6  color = [(0,0,0),(255,255,255)]
7  running = True
8
9  while running:
10   for event in pygame.event.get():
11      if event.type == pygame.QUIT:
12         running = False
13      if event.type == pygame.KEYDOWN:
14         color[0], color[1] = color[1],color[0]
15
16
17         screen.fill(color[0])
           pygame.display.flip()
```

01-simple_pygame.py

What each element does: **The Main Loop**



```
1  running = True while
2  running:
3      for event in pygame.event.get():
4          if event.type == pygame.QUIT:
5              running = False
6          if event.type == pygame.KEYDOWN:
7              color[0], color[1] = color[1],color[0]
8
9      screen.fill(color[0])
10     pygame.display.flip()
```

is the **main loop** of the game.

- **listen** to events (closing game, pressing keys) → **respond**
- **proceed** the game
- **draw** on the screen
- **stop** when done

Get Real DRAWINGS:)

DRAWING on the screen

Filling the screen with **a color**:

```
1 blue = (0,0,255)
2 screen.fill(blue)
3 pygame.display.flip()
```

After all drawing is done, call **display.flip()** to **update** the display.

Use pygame.draw to draw **geometric shapes**. A **circle**:

```
1 red = (255,0,0)
2 # position (320,240), radius = 50
3 pygame.draw.circle(screen, red, (320,240), 50)
```

DRAWING on the screen

Geometric shapes available for **pygame.draw**:

- **circle**(Surface, color, pos, radius, width=0)
- **polygon**(Surface, color, pointlist, width=0)
- **line**(Surface, color, start, end, width= 1)
- **rect**(Surface, color, Rect, width=0)
- **ellipse**(Surface, color, Rect, width=0)

Example:

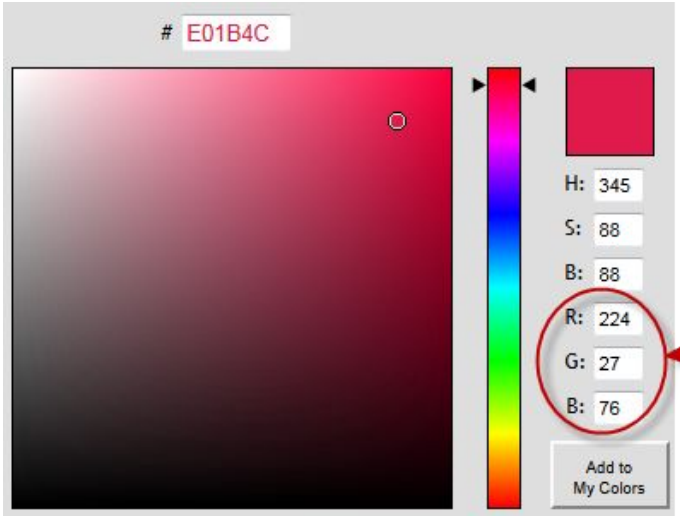
```
1  red = (255,0,0)
2  pygame.draw.line(screen, red, (10,50),(30,50),10)
```

DRAWING on the screen | **COLORS**

Defining a color:

```
gray = (200,  200 ,  200)
       #(red, green, blue)
```

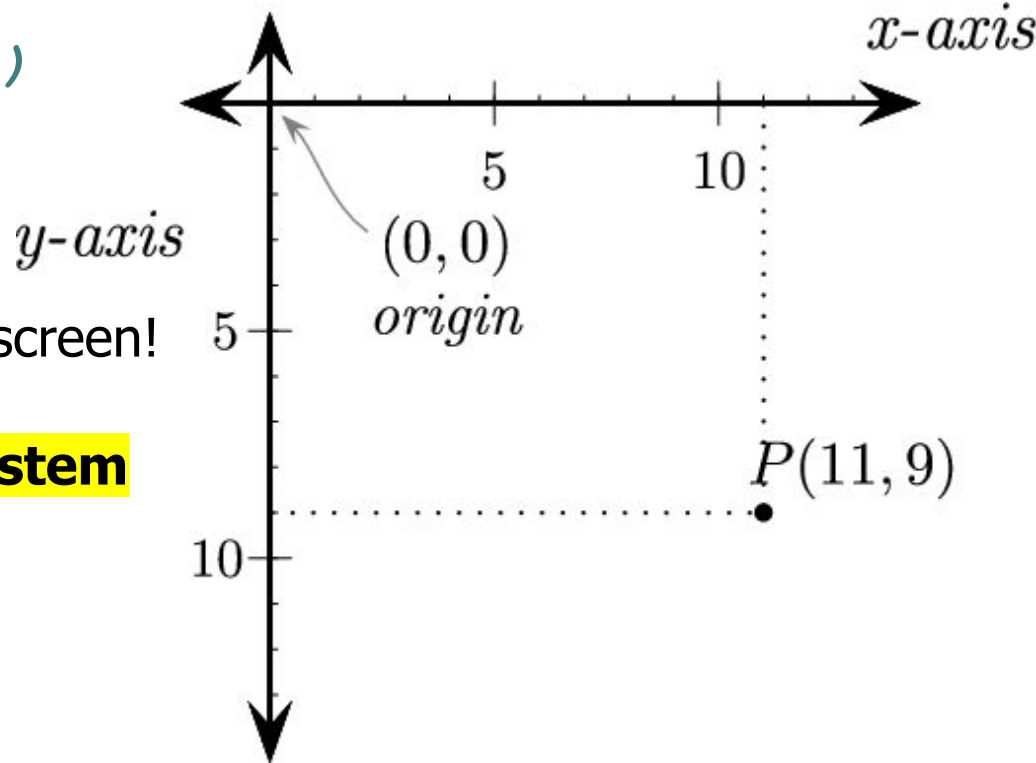
Use for example **colorpicker.com**:



DRAWING on the screen | **POSITIONS**

Defining a position:

```
P = (11,9)
#(x-axis, y-axis)
```



(0,0) starts at top left corner of screen!

To the reference **coordinate system**
x >> right
y >> down

DRAWING on the screen | **RECTS**

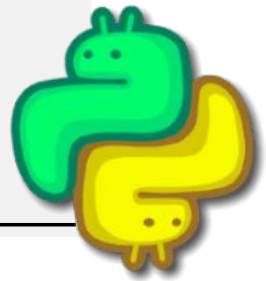
`pygame.Rect(left, top, width, height)`

to **create a Rect (i.e. rectangle)**

```
1  box = pygame.Rect(10, 10, 100, 40)
2  pygame.draw.rect(screen, blue, box)
3  #draws at (10,10) rectangle of width 100, height 40
```

A full drawing example

```
1  import pygame
2
3  pygame.init()
4  screen = pygame.display.set_mode((640, 480))
5
6  white = (255,255,255)
7  blue = (0,0,255)
8
9  running = True
10 while running:
11   for event in pygame.event.get():
12      if event.type == pygame.QUIT:
13         running = False
14
15      screen.fill(white)
16
17      pygame.draw.circle(screen, blue, (320,240), 100)
18      # position (320,240), radius = 100
19      pygame.display.flip()
```



04-drawing.py.

User Input

EVENTS | get()

get all events in **Pygame's event queue :**

`pygame.event.get()`

```
for event in pygame.event.get():
    if event.type == YourEvent:
        react_to_your_event()
```

EVENTS | Types

Some of the **most important** event types are:

```
pygame.QUIT
pygame.KEYDOWN
pygame.KEYUP
pygame.USEREVENT
```

EVENTS | KEYDOWN

```
while running:
    for event in pygame.event.get():
        if event.type == KEYDOWN:
            react_to_key()
```

React to KEYDOWN event:

Which key?
→ if event type is KEYDOWN or KEYUP event has attribute **key**.

```
for event in pygame.event.get():
    if event.type == KEYDOWN:
        if event.key == K_ESCAPE:
            running = False
```

PYGAME | KEYS

Some of the **most important keys** are:

```
K_RETURN K_SPACE
K_ESCAPE
K_UP, K_DOWN, K_LEFT, K_RIGHT
K_a, K_b, ...
K_0, K_1, ...
```

Full list of keys: http://www.pygame.org/docs/ref/key.html

pygame.key

pygame module to work with the keyboard

pygame.key.get_focused
true if the display is receiving keyboard input from the system

pygame.key.get_pressed
get the state of all keyboard buttons

pygame.key.get_mods
determine which modifier keys are being held

pygame.key.set_mods
temporarily set which modifier keys are pressed

GETTING **CONTINUOUS INPUT**

`KEYDOWN` is a **unique event**.

```
key = pygame.key.get_pressed()
```

to **get keys currently pressed**.

```
if key[pygame.K_UP]: move_up()
```

to **check and react** on a **specific key**.

A user input example

```
color = [0,0,0]

while running:
 for event in pygame.event.get():
    if event.type == pygame.QUIT:
       running = False
    if event.type == KEYDOWN and event.key == K_SPACE:
       color = [0,0,0]
  keys = pygame.key.get_pressed()
  if keys[K_UP]:
      color = [(rgb+1)%256 for rgb in color]
  screen.fill(color)
  pygame.display.flip()
```

05-user_input.py


```
243 void showNote(int mm){
244     FILE *fp;
245     int i = 0, isFound = 0;
246     system("cls");
247     fp = fopen("note.dat","rb");
248     if(fp == NULL){
249         printf("Error in opening the file");
250     }
251     while(fread(&R,sizeof(R),1,fp) == 1){
252         if(R.mm == mm){
253             gotoxy(10,5+i);
254             printf("Note %d Day = %d: %s", i+1, R.dd,  R.note);
255             isFound = 1;
256             i++;
257         }
258     }
259     if(isFound == 0){
260         gotoxy(10,5);
261         printf("This Month contains no note");
262     }
263     gotoxy(10,7+i);
264     printf("Press any key to back.......");
265     getch();
266
```

Pygame's Clock

Limiting the frames per second

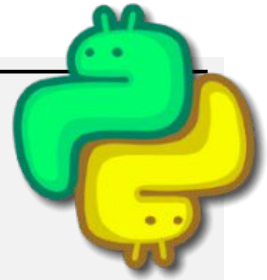
```
clock = pygame.time.Clock()
```

to **initialize the clock**.

In your **main loop call**

```
clock.tick(60) #limit to 60 fps
```

Clock example



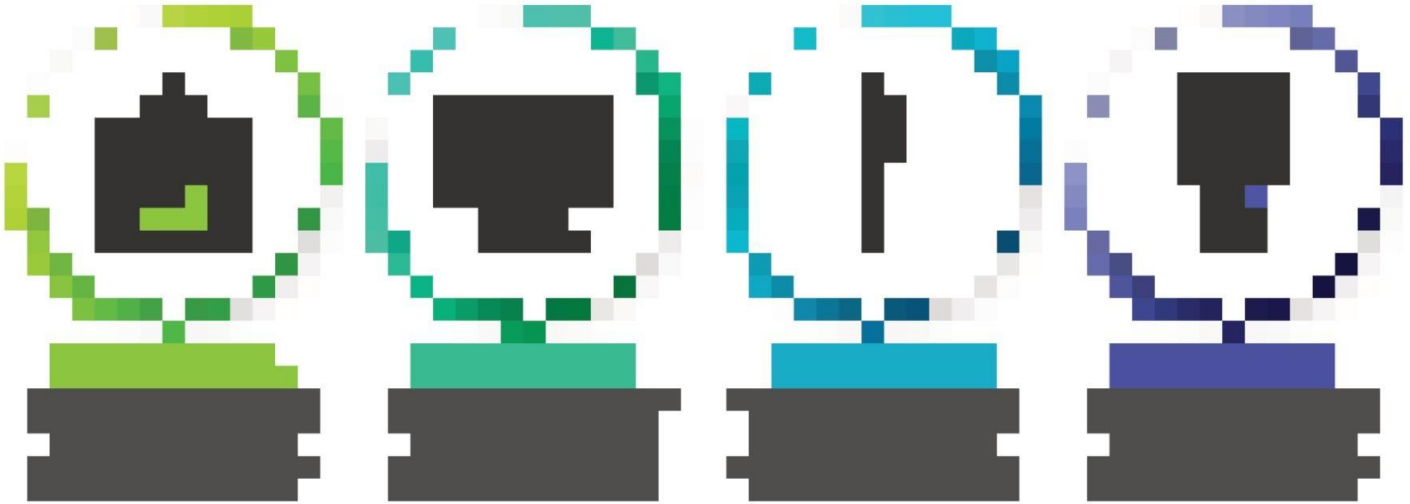
```
clock = pygame.time.Clock()

while running:
  for event in pygame.event.get():
     if event.type == pygame.QUIT:
        running = False
     if event.type == KEYDOWN and event.key == K_SPACE:
        color = [0,0,0]
     keys = pygame.key.get_pressed()
     if keys[K_UP]:
         color = [(rgb+1)%256 for rgb in color]
     screen.fill(color)
     pygame.display.flip()
     clock.tick(60)

```

06-clock.py

Game Events



Sprites

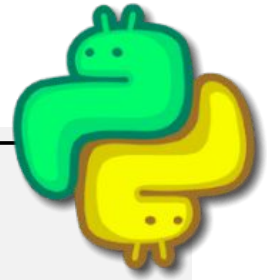
Pygame's **sprite class** gives **convenient options** for **handling interactive graphical objects**.

If you want to **draw** **and** **move (or manipulate)** and object, make it a **sprite**.

Sprites

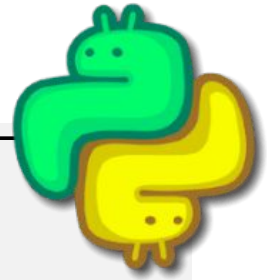
- can be grouped together
- are easily drawn to a surface (even as a group!)
- have an update method that can be modified
- have collision detection

Basic Sprite



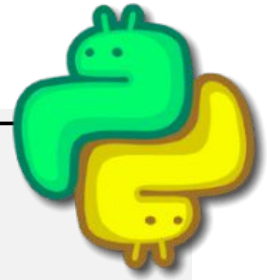
```
1  class SpriteExample(pygame.sprite.Sprite):
2
3      def __init__(self): pygame.sprite.Sprite.
4          init__(self)
5
6          self.image = #image
7          self.rect = #rect
8
9      def update(self): pass
10
```

Sprite from a local image



```
1  class SpriteExample(pygame.sprite.Sprite):
2
3      def __init__(self):
4          pygame.sprite.Sprite.__init__(self)
5          self.image = pygame.image.load('local_img.png')
6          self.rect = self.image.get_rect()
7          self.rect.topleft = (80,120)
8
9      def update(self): pass
10
11
```

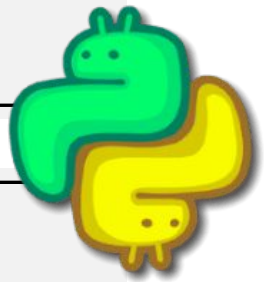
Self-drawn sprites: Rectangle



```
1  class Rectangle(pygame.sprite.Sprite):
2
3      def __init__(self):
4          pygame.sprite.Sprite.__init__(self)
5          self.image = pygame.Surface([200, 50])
6          self.image.fill(blue)
7          self.rect = self.image.get_rect() self.rect.top,
8          self.rect.left = 100, 100
9
10     def update(self): pass
11
```

Sprites: **A reminder about classes**

```
1  class Rectangle(pygame.sprite.Sprite):
2
3    def __init__(self, color):
4          pygame.sprite.Sprite.__init__(self)
5          self.image = pygame.Surface([200, 50])
6          self.image.fill(color)
7          self.rect = self.image.get_rect()
8          self.rect.top, self.rect.left = 100, 100
9
10     def update(self):
11         pass
```



```
1  white, blue = (255,255,255), (0,0,255)
2
3  white_rect = Rectangle(white)
4  blue_rect = Rectangle(blue)
```

Sprite groups

```
new_sprite_group = pygame.sprite.Group(sprite1)
```

to create a new sprite group containing sprite sprite1.

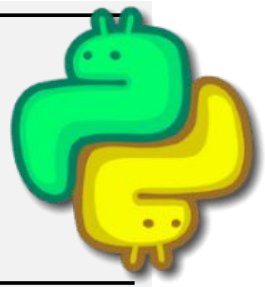
```
new_sprite_group.add(sprite2)
```

to add another sprite later.

Group updating and drawing:

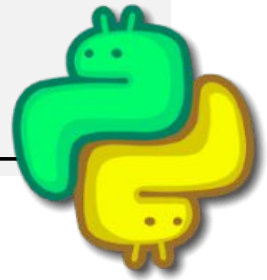
```
#inside main loop:

new_sprite_group.update() #game events
new_sprite_group.draw()   #drawing
```



Framework for working with sprite groups

```
1  class NewSprite(pygame.sprite.Sprite):
2      #defining a new sprite
3
4  newsprite = NewSprite()
5  sprites = pygame.sprite.Group()
6  sprites.add(newsprite)
7
8  while running:
9      for event in …
10        sprites.update() #game events
11        sprites.draw()
12        pygame.display.flip()
13
```



```
1  class Circle(pygame.sprite.Sprite):
2   def __init__(self):
3        pygame.sprite.Sprite.__init__(self)
4        self.image = pygame.Surface([100, 100])
5        pygame.draw.circle(self.image, blue, (50, 50), 50)
6        self.rect = self.image.get_rect()
7        self.rect.center = [320,240]
8
9   def draw(sprites):
10       screen.fill(white)
11       sprites.draw(screen)
12       pygame.display.flip()
13
14 circle = Circle()
15 sprites = pygame.sprite.Group(circle)
16
17 while running:
18   sprites.update()
19   draw(sprites)
```

Sprite groups: Working example

07-sprite-circle.py

Acknowledgement

Initial slides by Felix Hoffmann.

Modified by Minh Tran at University of Chicago and Han Dang at University of Illinois Chicago, for use in a summer pre-college course titled Creating Computer Games for Learning, Summer 2026.

https://youtu.be/blLLtdv4tvo?si=7xlgyedxp_HOPcHU