**Common Display Functions**

pygame.display.set\_mode((WIDTH, HEIGHT)) -> Surface

Returns a Surface object representing the screen with given WIDTH and HEIGHT.

Usually this is assigned to a global variable called screen.

pygame.display.update()

Updates the display.

**Common Surface Methods**

pygame.Surface((WIDTH, HEIGHT)) -> Surface

Assume a Surface object has been stored in the variable surface.

surface.fill(color)

surface.blit(image, rect)

Paints an image, technically a Surface, at the location defined by the rect.

**Rect Methods and Properties**

pygame.Rect(left, top, width, height) -> Rect

Assume a Rect object has been stored in the variable rect. The following are commonly used methods and attributes of the rect.

rect.contains(rect) -> bool

rect.collidepoint((x, y)) -> bool

rect.colliderect(rect) -> bool

rect.collidelist(list) -> index

rect.x, rect.y

rect.top, rect.left, rect.bottom, rect.right

rect.topleft, rect.bottomleft, rect.topright, rect.bottomright

rect.centerx, rect.centery

(x+width, y)

(right, top)

topright

rect.center

(x, y)

(left, top)

topleft

rect.size, rect.width, rect.height

(x, y+height)

(left, bottom)

bottomleft

(x+width, y+height)

(right,bottom)

bottomright

(centerx, centery)

center

**Common Draw Functions**

Each draw function requires a Surface object to draw upon this can be the global variable screen or another Surface you created such as one for a sprite.image.

pygame.draw.rect(surface, color, rect, width=0)

rect must be a pygame.Rect object

width is an *optional* *arguments.* width=0 fills rect.

pygame.draw.polygon(surface, color, points, width=0)

points is a list of coordinates such as [(x1, y1), (x2, y2), …]

pygame.draw.circle(surface, color, center, radius, width=0)

pygame.draw.line(surface, color, start\_pos, end\_pos, width=1)

start\_pos and end\_pos are tuples (x, y).

Advanced drawing functions and options available at https://www.pygame.org/docs/

**Sprite Basics**

pygame.sprite.Sprite () -> Sprite

In order to work properly, the following custom properties should be added to your sprites. Note, these properties do not exist until you assign them. Assume an instance of Sprite is called sprite.

sprite.image

A Surface either created and drawn by you or loaded with pygame.image.load

sprite.rect

A Rect object which holds the information on where the sprite image is located on the screen.

**Common Mouse Functions**

pygame.mouse.get\_pos() -> (x, y) gets the mouse position

pygame.mouse.get\_rel() -> (dx, dy) gets the amount of movement since last call.

pygame.mouse.set\_visible(bool) True to make visible, False to hide

**Common Key Functions and Codes**

pygame.key.get\_pressed() -> [bools]

Returns a list of all the keys with the value at a keycode's index True if the key is being

pressed. (see keycode constants)

pygame.key.set\_repeat(delay)

delay is the amount of time before a held down key triggers an additional key event.

**Common Events Functions and Codes**

pygame.event.get() -> [events]

Returns a list of all the events that have occurred since the last call.

Commonly included in an event loop such as for event in pygame.event.get():

All event objects have a type attribute. Depending upon the type of the event, the object will have these additional attributes. This list includes some of the commonly used attributes.

**Event Types Attributes**

QUIT none

KEYDOWN key

KEYUP key

MOUSEMOTION pos, rel, buttons

MOUSEBUTTONUP pos, button

MOUSEBUTTONDOWN pos, button

JOYAXISMOTION axis, value

JOYBUTTONUP button

JOYBUTTONDOWN button

**Sprite Class Requirements**

Classes used to create Sprite objects must extend pygame.sprite.Sprite. For example,

class MySprite(pygame.sprite.Sprite)

Suppose sprite is an object made by a class extending pygame.sprite.Sprite. The following methods and attributes must be defined in the class.

sprite.update()

This method is called by the group.update() and should update the sprites position

and properties.

sprite.rect, sprite.image are required attributes.

sprite.rect is used to position the sprite

sprite.image is used to draw the sprite to a surface at the location of its rect

**Common Sprite Group Methods**

pygame.sprite.Group(sprite1, sprite2, …) -> Group

Creates a sprite group containing all of the sprites, or empty is no arguments.

Suppose group is a pygame.sprite.Group object. The following are commonly used methods and attributes of the group.

group.add(sprite1, sprite2, …)

group.remove(sprite1, sprite2, …)

group.update()

calls sprite.update() on each sprite in the group.

Arguments can be passed to group.update and then will be passed to each sprite's

update.

group.draw(surface)

Draws each sprite in the group on the surface (often screen) at the location of the

sprites rect attribute.

group.sprites() -> sprite\_list

group.has(sprite) -> bool

**Basic main.py File**

import pygame

from settings import \*

screen = pygame.display.set\_mode( (WIDTH, HEIGHT) )

pygame.display.set\_caption(TITLE)

def update():

pass # put code to automate sprite movement if desired

def draw():

screen.fill(BGCOLOR)

pygame.display.update() # update the screen

def onMouseDown(x, y):

pass

**Keycode Constants for** key **parameter**

pygame.K\_UP up arrow

pygame.K\_DOWN down arrow

pygame.K\_RIGHT right arrow

pygame.K\_LEFT left arrow

pygame.K\_SPACE space key

pygame.K\_RETURN return/enter key

pygame.K\_a a key

pygame.K\_b b key

etc.

def onMouseMove(x, y):

pass

def onKeyDown(key):

pass

def mainloop():

running = True

clock = pygame.time.Clock()

while running:

update()

draw()

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

pygame.quit()

elif event.type == pygame.MOUSEMOTION:

onMouseMove(event.pos[0], event.pos[1])

elif event.type == pygame.MOUSEBUTTONDOWN:

onMousePress(event.pos[0], event.pos[1])

elif event.type == pygame.KEYDOWN:

onKeyDown(event.key)

clock.tick(FPS)

pygame.init()

mainloop()

**Basic settings.py File**

from colors import \*

FPS = 60

WIDTH = 800

HEIGHT = 600

TITLE = 'BASIC GAME'

BGCOLOR = SIENNA

**Basic colors.py File**

# COLORS R G B

BLACK = ( 0, 0, 0)

BLUE = ( 0, 0,255)

BROWN = (165, 42, 42)

CORNSILK = (255,248,220)

CRIMSON = (220, 20, 60)

CYAN = ( 0,255,255)

DARKGREEN = ( 0,100, 0)

FIREBRICK = (178, 34, 34)

GOLD = (255,215, 0)

GRAY = (100,100,100)

GREEN = ( 0,128, 0)

INDIGO = ( 75, 0,130)

MAGENTA = (255, 0,255)

MAROON = (128, 0, 0)

MIDNIGHTBLUE= ( 25, 25,112)

NAVY = ( 0, 0,128)

LIME = ( 0,255, 0)

OLIVE = (128,128,128)

ORANGE = (255,128, 0)

PURPLE = (128, 0,128)

RED = (255, 0, 0)

ROYALBLUE = ( 65,105,225)

SADDLEBROWN = (139, 69, 19)

SIENNA = (160, 82, 45)

SILVER = (192,192,192)

VIOLET = (238,130,238)

WHEAT = (245,222,179)

WHITE = (255,255,255)

YELLOW = (255,255, 0)