**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**

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

surface.fill(color)

surface.get\_at((x, y)) -> Color

Returns a Color object representing the color of the pixel at (x, y)

surface.set\_at((x, y), color)

Sets the color of the pixel at (x, y)

surface.get\_size() -> (width, height)

surface.get\_width() -> width

surface.get\_height() -> height

**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.copy() -> Rect

rect.move() -> 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

rect.center

rect.size, rect.width, rect.height

**Common Draw Functions**

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

surface is a Surface object, often the global variable screen

rect must be a pygame.Rect object

width and border\_radius are 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), …]

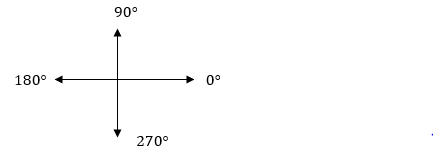
pygame.draw.circle(surface, color, center, radius, width=0)

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

rect is a Rect object that bounds the ellipse shape

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

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

pygame.draw.arc(surface, color, start\_angle, stop\_angle)

The arc is drawn in a counterclockwise direction from the

start\_angle to the stop\_angle, angle measures in radians.

**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.

**Keycode Constants**

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.

**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 Game Loop**

The following game loop example assumes that the coder has written custom update, draw, on\_mouse\_move(pos), on\_mouse\_down(pos, button), on\_key\_down(key), and on\_key\_up(key) functions.

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:

on\_mouse\_move(event.pos)

elif event.type == pygame.MOUSEBUTTONDOWN:

on\_mouse\_down(event.pos, event.button)

elif event.type == pygame.KEYDOWN:

on\_key\_down(event.key)

elif event.type == pygame.KEYUP:

on\_key\_up(event.key)

clock.tick(FPS)

pygame.init()

mainloop()