Pygame Reference Sheet

YoungWonks

EMPTY WINDOW

```
import pygame

#initializes pygame
pygame.init()
#creates the pygame window
screen = pygame.display.set_mode((width,height))
#sets the name of the pygame window
pygame.display.set_caption("Name of window")
while True:
  #continuously updates the window
  pygame.display.update()
```

EVENT LOOP & QUIT EVENT

```
#gets a list of events from the window
#each event type is handled with an if
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        exit()
```

COLORS

```
#color = (R, G, B)
#tuple of 3 integers
#between 0 - 255
red = (255, 0, 0)
blue = (0, 0, 255)
green = (0, 255, 0)
```

COLORS

```
pink = (175, 0, 175)
orange = (240, 100, 0)
yellow = (235, 225, 0)
black = (0, 0, 0)
white= (255, 255, 255)
```

BACKGROUND COLOR

```
#fills background color
screen.fill(bg_color)
```

DRAW TEXT

```
#display text on window
def show_text(msg,x,y,color):
    fontobj= pygame.font.SysFont("freesans",32)
    msgobj = fontobj.render(msg,False,color)
    screen.blit(msgobj,(x,y))

show_text ("Hello",x,y,color)
```

DRAWING SHAPES

```
#rectangle
pygame.draw.rect(screen,color,(x,y,width,length),thickness)
#circle
pygame.draw.circle(screen,color,(x,y),radius,thickness)
#line
pygame.draw.line(screen,color,(x1,y1),(x2,y2),thickness)
#polygon
pygame.draw.polygon(screen,color,((x1, y1),(x2, y2),(x3, y3),..),thickness)
```

DRAW IMAGES

```
#load an image
imagevariable = pygame.image.load(path)
#show image on window
screen.blit(imagevariable,(x,y))
```

GAME CLOCK AND SPEED

```
#update the clock
clock=pygame.time.Clock()
clock.tick(frames_per_sec)
# pause the program
pygame.time.delay(milliseconds)
pygame.time.wait()
```

IMAGE TRANSFORMATION

```
#rotate image
pygame.transform.rotate(imagevariable, angle)
#resize image
pygame.transform.scale(imagevariable, (width, height))
#flip image
pygame.transform.flip(imagevariable)
#scale and rotate image
pygame.transform.rotozoom(Screen, angle, scale)
```

ANIMATION

```
#load images into list
animation = []
for i in range(num_of_images):
    image = pygame.image.load('image'+str(i)+'.png')
    animation.append(image)
#stepping through the images
screen.blit(animation[count], (x, y))
count +=1
if count == len(animation):
    count = 0
```

KEYBOARD EVENTS

```
#placed inside the event loop
#for a list of keys,
#visit pygame.org/docs/ref/key.html
if event.type == pygame.KEYDOWN:
    #up arrow key
    if event.key == pygame.K_UP:
        print('up key pressed',event.key)
    #down arrow key
    if event.key == pygame.K_DOWN:
        print('down key pressed')
    #letter a
    if event.key == K_a:
        print('a key pressed',chr(event.key))
if event.type == pygame.KEYUP:
    #up arrow key
    if event.key == pygame.K_UP:
        print('up key released',event.key)
    #down arrow key
    if event.key == pygame.K_DOWN:
        print('down key released')
    #letter a
    if event.key == K_a:
        print('a key released',chr(event.key))
```

MOUSE EVENTS

```
#placed inside the event loop
if event.type == pygame.MOUSEMOTION:
    #triggered for every movement of the mouse
    print(event.pos)
if event.type == pygame.MOUSEBUTTONDOWN:
    #left button
    if event.button == 1:
        print('left button pressed',event.pos)
    #middle button
    if event.button == 2:
        print('middle button pressed',event.pos)
    #right button
    if event.button == 3:
        print('right button pressed',event.pos)
if event.type == pygame.MOUSEBUTTONUP:
    #left button
    if event.button == 1:
        print('left button released',event.pos)
    #middle button
    if event.button == 2:
        print('middle button released',event.pos)
    #right button
    if event.button == 3:
        print('right button released',event.pos)
```

Pygame Reference Sheet

YoungWonks

RECT OBJECTS

```
#a rect object the rectangular area around a shape
#a rect object is very useful to detect collisions

#creating a rect object
rect = pygame.Rect(x, y, width, height)
print(rect)

#drawing a shape return rect objects
#works for all shapes
rect = pygame.draw.rect(screen, color, (x, y, width, height))

#getting the rect area of an image
image = pygame.image.load('path_to_image')
rect = image.get_rect()

#blitting also returns a rect object
rect = screen.blit(image,(x,y))
```

SURFACES

```
#it is a reactangular surface to draw on
#it helps us create surfaces in advace
#they can just be drawn when needed.
surface = pygame.Surface((width, height))
surface.fill(color)
pyagme.draw.circle(surface,color,(x, y), radius)
rect = screen.blit(surface, (x, y))
```

COLLISIONS

```
#checks if rect2 is contained within rect1
#returns a boolen value
rect1.contains(rect2)

#checks if rect1 and rect2 are colliding
#returns boolean value
rect1.colliderect(rect2)

#checks is a point is inside a rectangle
#useful to check if a rectable is clicked
rect.collidepoint((x, y))

#checks is a point is inside a rectangle
#useful to check a bullet hits any character
rect1.collidelist(list_of_rects)
```

SOUNDS AND MUSIC

```
#initialize sound in pygame
pygame.mixer.pre_init()

##Background sound/music
#loads music to pygame
pygame.mixer.music.load(filename)

#start background music
#repeat : -1 is indefinitely ,0 is no repeat
pygame.mixer.music.play(repeat)
#stop background music
pygame.mixer.music.stop()
#pause background music
pygame.mixer.music.pause()
#unpause background music
pygame.mixer.music.unpause()

##Event sounds
#loads sound
sound = pygame.mixer.Sound(filename)

#play sound
sound.play(repeat, maxtime)
#stop sound
sound.stop()
```

CHECKERBOARD

```
WIDTH = 640
HEIGHT = 480
def checkerboard(rows, columns):
    colors = [(255,0,0),(0,0,0)]
    r_h = HEIGHT/rows
    c_w = WIDTH/columns
    for col in range(columns):
        for row in range(rows):
            pygame.draw.rect(screen, colors[0],
                             (col*c_w, row*r_h,c_w, r_h))
            colors.reverse()
        if columns%2==0:
            colors.reverse()

checkerboard(10, 20)
```

TYPICAL USAGE

```
import pygame
#initializes pygame
pygame.init()

#creates the pygame window
screen = pygame.display.set_mode((WIDTH, HEIGHT))

#sets the name of the pygame window
pygame.display.set_caption("Name of window")

#color = (R, G, B) : tuple of 3 integers between 0 - 255
red = (255, 0, 0)
blue = (0, 0, 255)
green = (0, 255, 0)
black = (0, 0, 0)
white= (255, 255, 255)

#Assets are initialized here
#Variables are created here
#functions are defined here
while True:
    screen.fill((0,0,0))

    #Game logic goes here

    #Event Loop
    for event in pygame.event.get():
        #Conditions to handle events
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        #Keyboard Events
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                print('up key pressed',event.key)
            if event.key == K_a:
                print('a key pressed',chr(event.key))
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_UP:
                print('up key released',event.key)
            if event.key == K_a:
                print('a key released',chr(event.key))
        #Mouse Events
        if event.type == pygame.MOUSEMOTION:
            print(event.pos)
        if event.type == pygame.MOUSEBUTTONDOWN:
            if event.button == 1:
                print('left button pressed',event.pos)
            if event.button == 3:
                print('right button pressed',event.pos)
        if event.type == pygame.MOUSEBUTTONUP:
            if event.button == 1:
                print('left button released',event.pos)
            if event.button == 3:
                print('right button released',event.pos)

    pygame.display.update() #continuously updates the window
```