Introduction to Visual Media Programming

: Interactive Drawing & Animation

Student No	20221593
Name	Lee Seo Yeon
Date	2022.01.09

I. Import

1) Import libraries

```
import pygame
import random
import os
import numpy as np
```

II. Upload Files

1) Upload images

```
player_img = pygame.image.load(os.path.join(assets_path, 'player.png'))
angry_img = pygame.image.load(os.path.join(assets_path, 'angry.png'))
happy_img = pygame.image.load(os.path.join(assets_path, 'happy.png'))
sad_img = pygame.image.load(os.path.join(assets_path, 'sad.png'))
background = pygame.image.load(os.path.join(assets_path, 'background.png'))
background_rect = background.get_rect()


ghost_item_img = pygame.image.load(os.path.join(assets_path, 'ghost_item.png'))
heart_item_img = pygame.image.load(os.path.join(assets_path, 'heart_item.png'))
umbrella_item_img = pygame.image.load(os.path.join(assets_path, 'umbrella_item.png'))
power_item_img = pygame.image.load(os.path.join(assets_path, 'power.png'))
present_img = pygame.image.load(os.path.join(assets_path, 'present.png'))

bomb_img = pygame.image.load(os.path.join(assets_path, 'bomb.png'))
bombskull_img = pygame.image.load(os.path.join(assets_path, 'bomb_skull.png'))
skull_img = pygame.image.load(os.path.join(assets_path, 'skull.png'))
meteor_img = pygame.image.load(os.path.join(assets_path, 'meteor.png'))
tornado_img = pygame.image.load(os.path.join(assets_path, 'tornado.png'))
wind_img = pygame.image.load(os.path.join(assets_path, 'wind.png'))
gimg = pygame.image.load(os.path.join(assets_path, 'ghost.png'))
ghost_img = pygame.transform.scale(gimg, (300,300))

heart_img= pygame.image.load(os.path.join(assets_path, 'heart.png'))
bullet_img = pygame.image.load(os.path.join(assets_path, 'bullet.png'))
umbrella_img = pygame.image.load(os.path.join(assets_path, 'umbrella.png'))
explosion_img = pygame.image.load(os.path.join(assets_path, 'explosion.png'))
```

2) Upload sounds

```
game_over_sound = pygame.mixer.Sound(os.path.join(assets_path, 'gameoversound.wav'))
explosion_sound = pygame.mixer.Sound(os.path.join(assets_path, 'explosionsound.wav'))
hit_sound =pygame.mixer.Sound(os.path.join(assets_path, 'hitsound.wav'))
mob_sound = pygame.mixer.Sound(os.path.join(assets_path, 'mobsound.wav'))
item_sound = pygame.mixer.Sound(os.path.join(assets_path, 'itemsound.wav'))
ghost_sound = pygame.mixer.Sound(os.path.join(assets_path, 'ghostsound.mp3'))
```

III. Functions

1) DrawText()

```
font_name = pygame.font.match_font('arial')
def DrawText(text, size, x, y, color):
    font = pygame.font.SysFont('FixedSys', size, True, False)
    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect()
    text_rect.midtop = (x, y)
    screen.blit(text_surface, text_rect)
```

Show text on the screen.

2) DrawHearts()

```
def DrawHearts():
    x = 0
    y = 0
    hearts = player.lives
    for i in range(hearts):
        img_rect = heart_img.get_rect()
        img_rect.x = x + 30 * i
        img_rect.y = y
        screen.blit(heart_img, img_rect)
```

Draw hearts on the screen.

3) Mobs Functions

```
def Mob_Tornado(pos):
    mob = Tornado(pos)
    wind1 = Wind(mob.rect.center, -45)
    wind2 = Wind(mob.rect.center, 0)
    wind3 = Wind(mob.rect.center, 45)
    AddMobs(mob)
    AddMobs(wind1)
    AddMobs(wind2)
    AddMobs(wind3)

def Mob_BombSkull(pos):
    mob = BombSkull(pos)
    bomb = Bomb(mob.rect.center)
    AddMobs(mob)
    all_sprites.add(bomb)
    bombs.add(bomb)

def Mob_Skull(pos):
    mob = Skull(pos)
    AddMobs(mob)

def Mob_Meteor():
    mob = Meteor()
    AddMobs(mob)
```

Create mob objects and add them to the sprite groups.

4) Item Functions

```
def Item_Present(pos):
    new = Present(pos)
    all_sprites.add(new)
    presents.add(new)

def Drop_item(pos):
    item = Item(pos)
    all_sprites.add(item)
    items.add(item)
```

Create present objects and item objects.

5) Ghost()

```
def Ghost():
    ghostpos=[]
    for i in range(3):
        x = random.randint(0, WINDOW_WIDTH-300)
        y = random.randint(0, WINDOW_HEIGHT-300)
        ghostpos.append([x,y])
    return ghostpos
```

Set the position of the ghost.

6) GetStart()

```
def GetStart():
    screen.blit(background, background_rect)
    DrawText("Emoji Word!", 64, WINDOW_WIDTH / 2, WINDOW_HEIGHT / 4, BLACK)
    DrawText("Press a key to begin", 18, WINDOW_WIDTH / 2, WINDOW_HEIGHT * 3 / 4, BLACK)
    pygame.display.flip()
    waiting = True
    while waiting:
        clock.tick(60)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
            if event.type == pygame.KEYUP:
                waiting = False
```

Show the start screen.

IV. Classes

1) Player

```
class Player(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = player_img
        self.rect = self.image.get_rect()
        self.rect.center = (WINDOW_WIDTH/2, WINDOW_HEIGHT-50)
        self.speedx = 0
        self.shoot_delay = 400
        self.last_shot = pygame.time.get_ticks()
        self.lives = 3
        self.power = 1
        self.powerup_time = pygame.time.get_ticks()
```

```
    def update(self):
        now = pygame.time.get_ticks()
        if self.power>=2 and now - self.powerup_time > POWERUP_TIME:
            self.power -= 1
            self.powerup_time = pygame.time.get_ticks()

        self.speedx = 0
        keystate = pygame.key.get_pressed()
        if keystate[pygame.K_LEFT]:
            self.speedx = -8
        if keystate[pygame.K_RIGHT]:
            self.speedx = 8

        self.rect.x += self.speedx

        if self.rect.right > WINDOW_WIDTH:
            self.rect.right = WINDOW_WIDTH
        if self.rect.left < 0:
            self.rect.left = 0
        self.shoot()
```

```
    def shoot(self):
        now = pygame.time.get_ticks()
        bulletpos_x = [self.rect.centerx, self.rect.centerx-30, self.rect.centerx-40]
        if now - self.last_shot > self.shoot_delay:
            for i in range(self.power):
                bullet = Bullet(bulletpos_x[self.power-1]+120/self.power*i, self.rect.top)
                bullets.add(bullet)
            self.last_shot = pygame.time.get_ticks()

    def powerup(self):
        if self.power<3:
            self.power +=1
        self.powerup_time = pygame.time.get_ticks()
```

Make the player move when the arrow key is pressed.
Shoot bullets constantly and regularly.

If player get the powerup item, the number of bullets increases.

```
class Bullet(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = bullet_img
        self.rect = self.image.get_rect()
        self.rect.bottom = y
        self.rect.centerx = x
        self.speed = -10

    def update(self):
        self.rect.y += self.speed
        if self.rect.bottom < 0:
            self.kill()

class Emotion(pygame.sprite.Sprite):
    def __init__(self, img):
        pygame.sprite.Sprite.__init__(self)
        self.image = img
        self.rect = self.image.get_rect()
        self.rect.center = player.rect.center - pygame.Vector2((0, 10))
    def update(self):
        self.rect.center = player.rect.center - pygame.Vector2((0, 10))
```

Bullet class and emotion class.

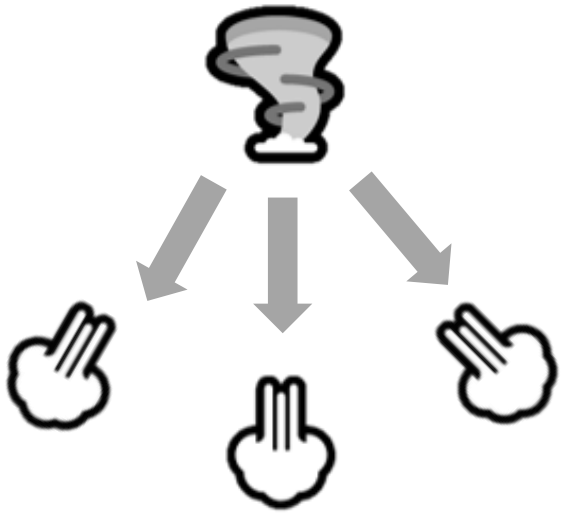
2) Mobs

```
class BombSkull(pygame.sprite.Sprite):
    def __init__(self, pos):
        pygame.sprite.Sprite.__init__(self)
        self.image = bombskull_img
        self.rect = self.image.get_rect()
        self.rect.left = pos
        self.rect.centery = -100

    def update(self):
        self.rect.y += falling_speed
        if self.rect.top>WINDOW_HEIGHT:
            self.kill()
```

```
class Bomb(pygame.sprite.Sprite):
    def __init__(self, pos):
        pygame.sprite.Sprite.__init__(self)
        self.image = bomb_img
        self.rect = self.image.get_rect()
        self.rect.center = pos
        self.bomb = random.randint(300,500)
        self.exploded = False
        self.speed = random.randint(5, 8)
    def update(self):
        if self.rect.bottom < WINDOW_HEIGHT:
            self.rect.y += self.speed
            self.clock = pygame.time.get_ticks()
        else:
            self.explode()
    def explode(self):
        if not self.exploded:
            now = pygame.time.get_ticks()
            if now - self.clock > self.bomb:
                self.image = explosion_img
                self.exploded = True
                self.clock = pygame.time.get_ticks()
        else:
            now = pygame.time.get_ticks()
            if now - self.clock > 300:
                explosion_sound.play()
                self.kill()
```

Bombskull drops the bomb.

```python
class Tornado(pygame.sprite.Sprite):
    def __init__(self, pos):
        pygame.sprite.Sprite.__init__(self)
        self.image = tornado_img
        self.rect = self.image.get_rect()
        self.rect.left = pos
        self.rect.centery = -100

    def update(self):
        self.rect.y += falling_speed
        if self.rect.top>WINDOW_HEIGHT:
            self.kill()

class Wind(pygame.sprite.Sprite):
    def __init__(self, pos, degree):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.transform.rotate(wind_img, degree)
        self.rect = self.image.get_rect()
        self.rect.center = pos
        if degree > 0 :
            self.xspeed = 1
        elif degree < 0 :
            self.xspeed = -1
        else:
            self.xspeed = 0
    def update(self):
        self.rect.x += self.xspeed
        self.rect.y += 3
        if self.rect.top>WINDOW_HEIGHT:
            self.kill()
```

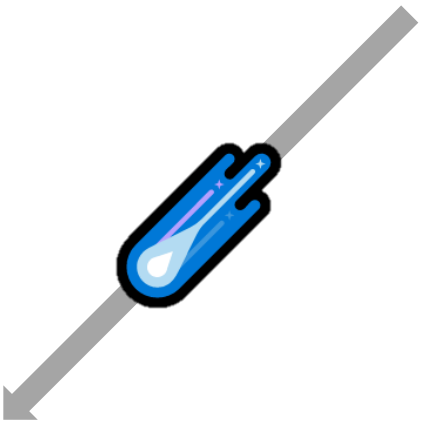


Tornado generates the wind in three directions.

```python
class Skull(pygame.sprite.Sprite):
    def __init__(self, pos):
        pygame.sprite.Sprite.__init__(self)
        self.image = skull_img
        self.rect = self.image.get_rect()
        self.rect.left = pos
        self.rect.centery = -100

    def update(self):
        self.rect.y += falling_speed
        if self.rect.top>WINDOW_HEIGHT:
            self.kill()
```

Shull.

```python
class Meteor(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = meteor_img
        self.rect = self.image.get_rect()
        self.rect.center = (WINDOW_WIDTH+100, -100)
        self.speed = random.randint(2,5)
    def update(self):
        self.rect.y += self.speed
        self.rect.x -= 2
        if self.rect.top>WINDOW_HEIGHT:
            self.kill()
```

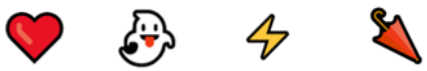


Meteor falls.

3) Items

```
class Item(pygame.sprite.Sprite):
    def __init__(self, pos):
        pygame.sprite.Sprite.__init__(self)
        type = random.randint(0, 4)
        if type==0:
            self.image = heart_item_img
            self.type = 'heart'
        elif type==1:
            self.image = ghost_item_img
            self.type = 'ghost'
        elif type==2:
            self.image = power_item_img
            self.type = 'powerup'
        else:
            self.image = umbrella_item_img
            self.type = 'shield'
        self.rect = self.image.get_rect()
        self.rect.center = pos

    def update(self):
        self.rect.y += 4
        if self.rect.top>WINDOW_HEIGHT:
            self.kill()
```

There are four types of items. Heart, Ghost, Powerup, Shield.



```
class Umbrella(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = umbrella_img
        self.rect = self.image.get_rect()
        self.rect.center = player.rect.center - pygame.Vector2((0, -50))
        self.shield_delay = 3000
        self.clock = pygame.time.get_ticks()
    def update(self):
        self.rect.center = player.rect.center - pygame.Vector2((0, 60))
        now = pygame.time.get_ticks()
        if now - self.clock > self.shield_delay:
            self.kill()
```

Heart increases the lives. Powerup increases the power. Shield protects the player for a moment.

V. Game Loop

```
falling_speed = 2
delay = 1500
ghost_delay = 1500
sad_delay = 2000

pygame.mixer.music.load(os.path.join(assets_path, 'background_music.mp3'))
pygame.mixer.music.play(loops=-1)

game_over = True
done = False
```

```
game_over = True
done = False
while not done:
    if game_over:
        GetStart()
        game_over=False
        player = Player()
        bullets = pygame.sprite.Group()
        emotion = Emotion(happy_img)
        all_sprites = pygame.sprite.Group()
        all_sprites.add(player, emotion)
        mobs = pygame.sprite.Group()
        bombs = pygame.sprite.Group()
        presents = pygame.sprite.Group()
        shields = pygame.sprite.Group()
        items = pygame.sprite.Group()
        score = 0
        last_update= pygame.time.get_ticks()
        shield = False
        ghost = False
        ghostpos = []
        sad = False

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
```

If game over, it initializes.

```
now = pygame.time.get_ticks()
if now - last_update > delay:
    Generate_Mobs()
    last_update= pygame.time.get_ticks()
    delay = random.choice([1000, 1500, 2000, 3000])

all_sprites.update()
bullets.update()
```

Mobs and presents are created at particular intervals.

```
    hits = pygame.sprite.groupcollide(mobs, bullets, True, True)
    for hit in hits:
        score += 50
        mob_sound.play()

    hits = pygame.sprite.groupcollide(bullets, presents, True, True)
    for hit in hits:
        Drop_item(hit.rect.center)


    hits = pygame.sprite.spritecollide(player, items, True)
    for hit in hits:
        score += 10
        item_sound.play()
        if hit.type == 'heart':
            player.lives +=1
        elif hit.type == 'ghost':
            ghost_clock = pygame.time.get_ticks()
            ghost = True
            ghostpos = Ghost()
            ghost_sound.play()
        elif hit.type == 'powerup':
            player.powerup()
        elif hit.type == 'shield':
            umbrella = Umbrella()
            all_sprites.add(umbrella)
            shields.add(umbrella)
            shield = True
```

When the mobs and bullets collide, the score increases.

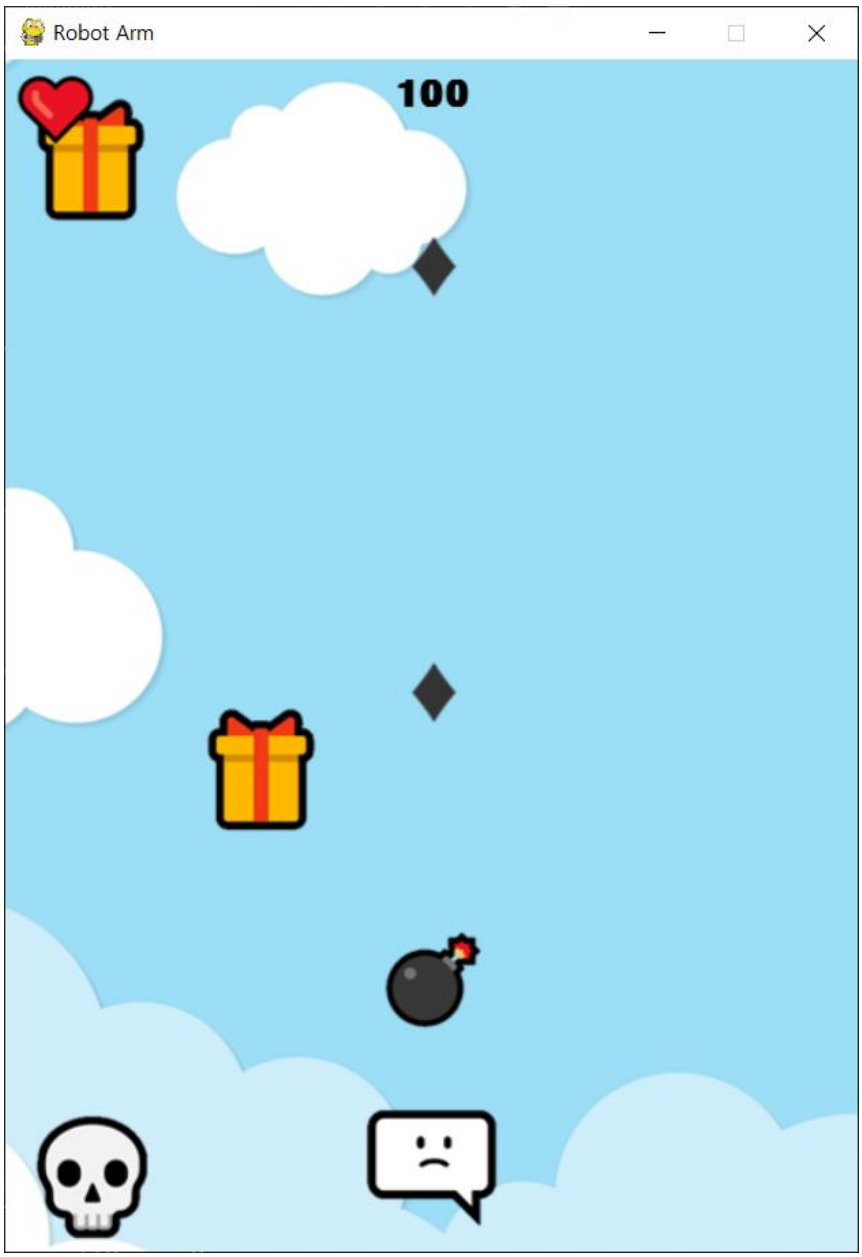
When the player and items collide, particular events occur according to the item type.

```
    if shield:
        hits = pygame.sprite.groupcollide(shields, mobs, True, True)
        for hit in hits:
            shield = False
    else:
        hits = pygame.sprite.spritecollide(player, mobs, True, pygame.sprite.collide_circle)
        for hit in hits:
            hit_sound.play()
            player.lives -= 1
            sad = True
            sad_clock = pygame.time.get_ticks()

    hits = pygame.sprite.spritecollide(player, bombs, True, pygame.sprite.collide_circle)
    for hit in hits:
        player.lives -= 1
        sad = True
        sad_clock = pygame.time.get_ticks()
```

When the mobs and player collide, the player loses the life.

```
    screen.fill(WHITE)
    screen.blit(background, background_rect)
    all_sprites.draw(screen)
    bullets.draw(screen)
    if ghost:
        if now - ghost_clock < ghost_delay :
            for i in ghostpos:
                screen.blit(ghost_img, i)
        else:
            ghost = False
    if sad:
        if now - sad_clock < sad_delay :
            emotion.image = sad_img
        else:
            sad = False
            emotion.image = happy_img
            if player.lives <= 0:
                game_over_sound.play()
                game_over = True
    DrawText(str(score), 32, WINDOW_WIDTH / 2, 10, BLACK)
    DrawHearts()
    pygame.display.flip()
    clock.tick(60)


pygame.quit()
```

V. Run



https://github.com/tjdus/Interactive-Drawing-Animation.git