13006107

Python Project

61090030

Mr.Satheinpong Polachan

SE

IC

KMITL

Project Proposal

1.Project Developer

Student ID : 61090030

Name : Sathienpong Polachan

2.Project Title : Drowning Shark

3.Project Description and Functions

“Drowning Shark” is the game that play for fun and compete with friends. This game your shark character will jumping to a platform. Using spacebar to jump, you don’t need to press spacebar every time , just press it when you start, ← for move left and → for move right, but be careful if you don’t land on the

platform you will die and your score will be stored. After that the game will restart, you can view your high score on the right corner of the game window.

4.Project Requirements

This project use pygame

CODE:

```
from pygame import *
import random

window_x = 500
window_y = 550

init()
window = display.set_mode((window_x, window_y))
display.set_caption('Drowning Shark!')
clock = time.Clock()
background_image = image.load('gamebg2copy.jpg').convert_alpha()
bar_image = image.load('bar.png').convert_alpha()
bg2 = -550
start = image.load('start.png').convert_alpha()
music = mixer.music.load('bg.ogg')
mixer.music.play(-1)


class Shark:
    def __init__(self):
        self.fall = image.load('avatar.png').convert_alpha()
        self.jumping_right = image.load('avatar.png').convert_alpha()
        self.jumping_left = transform.flip(self.jumping_right, True, False)
        self.stand = image.load('avatar.png').convert_alpha()

        self.reset()

    def reset(self):
```

```
        self.speed_x = 0
        self.speed_y = 0
        self.max_speed_x = 5
        self.max_speed_y = 15
        self.x_acceleration = 0.5
        self.img = self.jumping_right
        self.jump_speed = 15
        self.mob_timer = 0

        scale = 7
        self.width, self.height = 7 * scale, 12 * scale
        self.scale = scale

        self.x = (window_x - self.width) / 2
        self.y = window_y - self.height


    def update(self,p):
        self.side_control()
        self.physics(p)
        self.move()
        self.show()


        self.x += self.speed_x
        self.y -= self.speed_y

        return (self.img, (self.x, self.y, self.width, self.height))

    def physics(self, p):

        on = False
```

```
    for colour, rect in p:
        x,y,w,h = rect

        #X range
        if self.x + self.width / 2 > x and self.x - self.width / 2 < x + w:
            #Y range
            if self.y + self.height >= y and self.y + self.height <= y + h:

                if self.speed_y < 0:
                    on = True

    if not on and not self.y >= window_y - self.height:
        self.speed_y -= 0.5
    elif on:
        self.speed_y = self.jump_speed
    else:
        self.y = window_y - self.height
        self.speed_x = 0
        self.speed_y = 0
        if self.x != (window_x - self.width) / 2:
            if self.x > (window_x - self.width) / 2:
                self.x = max((window_x - self.width) / 2, self.x - 6)
            else:
                self.x = min((window_x - self.width) / 2, self.x + 6)

        else:
            keys = key.get_pressed()
            if keys[K_SPACE]:
                self.speed_y = self.jump_speed



def side_control(self):
```

```
        if self.x + self.width < 0:
            self.x = window_x - self.scale
        if self.x > window_x:
            self.x = -self.width

    def show(self):
        if self.speed_y > 0:
            if self.speed_x > 0: self.img = self.jumping_right
            elif self.speed_x < 0: self.img = self.jumping_left
        else:
            self.img = self.fall



    def slow_character(self):
        if self.speed_x < 0:
            self.speed_x = min(0, self.speed_x + self.x_acceleration / 6)
        if self.speed_x > 0:
            self.speed_x = max(0, self.speed_x - self.x_acceleration / 6)

    def move(self):
        keys = key.get_pressed()

        if not self.y >= window_y - self.height:

            if keys[K_LEFT] and keys[K_RIGHT]: self.slow_character()
            elif keys[K_LEFT]: self.speed_x -= self.x_acceleration
            elif keys[K_RIGHT]: self.speed_x += self.x_acceleration
            else: self.slow_character()

            self.speed_x = max(-self.max_speed_x, min(self.max_speed_x,
self.speed_x))
            self.speed_y = max(-self.max_speed_y, min(self.max_speed_y,
self.speed_y))
```

```
platform_spacing = 125 #space between platform

class Platform_Manager:
    def __init__(self):
        self.platforms = []
        self.spawns = 0
        self.start_spawn = window_y

        scale = 3
        self.width, self.height = 24 * scale, 6 * scale

    def update(self):
        self.spawner()
        return self.manage()



    def spawner(self):
        if window_y - info['screen_y'] > self.spawns * platform_spacing:
            self.spawn()

    def spawn(self):
        y = self.start_spawn - self.spawns * platform_spacing
        x = random.randint(-self.width, window_x)

        self.platforms.append(Platform(x,y,random.choice([1,-1])))
        self.spawns += 1


    def manage(self):
        u = []
```

```
        b = []
        for i in self.platforms:
            i.move()
            i.change_direction()
            b.append(i.show())

            if i.on_screen():
                u.append(i)

        self.platforms = u
        return b




class Platform:
    def __init__(self,x,y,direction):
        self.x = x
        self.y = y
        self.direction = direction
        self.speed = 2

        scale = 3
        self.width, self.height = 24 * scale, 6 * scale

    def move(self):
        self.x += self.speed * self.direction
        self.change_direction()

    def change_direction(self):
        if self.x <= 0:
            self.direction = 1
```

```
        if self.x + self.width >= window_x:
            self.direction = -1

    def on_screen(self):
        if self.y > info['screen_y'] + window_y:
            return False
        return True

    def show(self):
        return ((0,0,0), (self.x, self.y, self.width, self.height))



def blit_images(x):
    for i in x:
        window.blit(transform.scale(i[0], (i[1][2],i[1][3])), (i[1][0], i[1][1] -
info['screen_y']))

def event_loop():
    for loop in event.get():
        if loop.type == KEYDOWN:
            if loop.key == K_ESCAPE:
                quit()
        if loop.type == QUIT:
            quit()

##def show_menu():
##    window.blit(start,(0,0))
##    display.flip()
##    waiting = True
##    while waiting:
##        clock.tick(60)
##        for event in event.get():
```

```
##          if event.type == KEYUP:
##              waiting = False


f = font.SysFont('', 55)
def show_score(score, pos):
    message = f.render(str(round(score)), True, (100,100,100))
    rect = message.get_rect()

    if pos == 0:
        x = window_x - rect.width - 10
    else:
        x = 10
    y = rect.height + 10

    window.blit(message, (x, y))


info = {
    'screen_y': 0,
    'score': 0,
    'high_score': 0
    }

stick_man = Shark()
platform_manager = Platform_Manager()
platform_image = image.load('bar.png')

start_screen = True
while True:
    #MATH THINGS
##    if start_screen:
##        show_menu()
```

```
##    start_screen = False

    event_loop()

    platform_blit = platform_manager.update()
    stick_blit = stick_man.update(platform_blit)
    info['screen_y'] = min(min(0,stick_blit[1][1] -
window_y*0.4),info['screen_y'])
    info['score'] = (-stick_blit[1][1] + 470)/50

    #print(stick_blit[1][1], info['screen_y'])
    if stick_blit[1][1] - 470 > info['screen_y']:
        info['score'] = 0
        info['screen_y'] = 0
        stick_man = Stick_Man()
        platform_manager = Platform_Manager()

    clock.tick(60)

    #DISPLAY THINGS
    scroll = bg2 % background_image.get_rect().height
    window.blit(background_image, [0,scroll -
background_image.get_rect().height ])
    if scroll < window_y:
        window.blit(background_image,(0,scroll))
    bg2 += 2

    for x in platform_blit:
        i = list(x)
        i[1] = list(i[1])
        i[1][1] -= info['screen_y']
        window.blit(platform_image, i[1])
    blit_images([stick_blit])
```

```
info['high_score'] = max(info['high_score'], info['score'])

show_score(info['score'],1)
show_score(info['high_score'],0)

display.update()
```

IN GAME PICTURE: