**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:

