**Acknowledgements**

I would like to express my gratitude to the principal, ***Ms. Shantha Chandran***, and the school for giving me the opportunity to create this project. I would also like to thank our Computer Science teachers, ***Ms. Manjula*** and ***Ms. Suguna*** for guiding me at every step of the project. Their insights have been extremely valuable and helped me build an efficient, robust and fault tolerant program.

Also, the completion of this project would not have been possible without the immense support and encouragement given to me by parents and my sister.

**Index**

|  |  |  |
| --- | --- | --- |
| **Serial no.** | **Title** | **Page no.** |
| 1. | Introduction |  |
| 2. | Requirements |  |
| 3. | Modules used |  |
| 4. | Functions |  |
| 5. | Variables |  |
| 6. | Algorithm |  |
| 7. | Source Code |  |
| 8. | Screen shots |  |
| 9. | Bibliography |  |

**Introduction**

**System Requirements**

**Requirements to run Python file:**

1. Python 2.7.x 32 bit Version
2. Python IDLE or any other Python Interpreter
3. Pygame Module: Version 1.9.1 or higher

**Basic Algorithm**

Step 1: Start.

Step 2: Load images, sounds and music.

Step 3: Create superman object.

Step 4: Create platform objects.

Step 5: Create coin objects.

Step 6: Initialize fonts and variables.

Step 7: Start main loop of game.

Step 8: Check whether a platform is present under superman.

If yes, proceed to Step 9a.

Else, proceed to Step 9b.

Step 9a: Continue as per Step 8.

Step 9b: Make superman go down and continue as per step 8.

Step 10: Check whether superman touches a coin.

If yes, increment coin counter and score.

Step 11: Check user input and make superman fly if possible.

Step 12: Check if user exits or if superman falls into lava.

If yes, then go to step 13.

Else, go to step 8.

Step 13: Then exit main loop of game and go to high scores.

Step 14: Check if user exits, then go to Step 15.

Step 15: Stop.

**Modules and Library Functions**

|  |  |  |
| --- | --- | --- |
| **Module** | **Library Functions** | **Usage** |
| pygame/pygame. locals | .font | To generate fonts used |
| .render | To render the font |
| .draw | To draw a given shape |
| .blit | Adds object to pygame screen |
| .display.update | Refreshes pygame screen |
| .time.clock | Initializes the clock |
| .display.set\_mode | Initializes display |
| .event.get | Returns list of events |
| .mixer | Controls music |
| .quit | Quits pygame |
| sys | .exit | Exits from the terminal window |
| random | .randint | Generating random integer |
| .choice | Selects a random element from a sequence |
| .gauss | Generates a Gaussian distribution centred around mean with a given Standard Deviation |
| time | .time | Return the current time in seconds since the Epoch |
| .sleep | Delay execution of code |

**User Defined Functions**

|  |  |
| --- | --- |
| **Function name** | **Task** |
| Platgen | Generates the platforms |
| Bi | Converts decimal to binary |
| Hscore | Checks whether score should be stored |
| Dispscore | Displays the top 5 scores |
| Getname | Retrieves the name from the pygame screen |
| Pixelator | Generates random pixels for intro screen |
| Logo | Displays the logo |
| Menu | Generates the menu and waits for user input |
| Quitter | Handles final quitting by player |

**Variables**

|  |  |
| --- | --- |
| **Data Type** | Variable name |
| String | up, centre, down, nstr, |
| Dictionary | plist, pyabs, endcoor, height\_flag |
| List | Platlist, Splist, Clist, menu\_list |
| Integer/Floating Point | x, y, dx, dy, scrwidth, scrheight, yabs |
| Objects | Superman, Coins, Platforms |
| Boolean | alive, deathsoundstate, newcoin, flyflag |
| Tuple | BLACK, RED, GREEN, YELLOW, WHITE |

**Source Code**

The Python source code for the program is as follows:

“””

import pygame,sys

import time, random

from pygame.locals import \*

pygame.init()

"""Initial definitions of global varibales for screen setup and images of each object"""

scrwidth = 640

scrheight = 360

scr = pygame.display.set\_mode((scrwidth,scrheight),0,32)

pwidth = 150 #sizes of the platform images

pheight = 40

cwidth = 20 #sizes of the coin images

cheight = 20

skwidth = 80

skheight = 36

#Colours

BLACK = (0,0,0)

GREEN = (0,255,0)

RED = (255,0,0)

YELLOW = (255,255,0)

WHITE = (255,255,255)

#better to have pics which require high resolution in jpg

#pics which are small in size/res can be in png instead

back = "backt.jpg"

platimg = "platform2t.png"

supermanimg = "superman2t.png"

supermanupimg = "supermanupt.png"

supermandownimg = "supermandownt.png"

inst = "Instructions.png"

sto = "Story.png"

coin1 = "coin1.png"

coin10 = "coin10.png"

coin5 = "coin5.png"

lavaimg = "lava.png"

#supermanslantimg = "superman2.jpg" # for 'jumping'

#must convert images to use in pygame

spimg = pygame.image.load(supermanimg).convert\_alpha() # Normal

spUpimg = pygame.image.load(supermanimg).convert\_alpha() # Move Up

spDownimg = pygame.image.load(supermanimg).convert\_alpha() # Move Down

corner = spimg.get\_at((0, 0))

spimg.set\_colorkey(corner, RLEACCEL)

angle = 5

spUpimg = pygame.transform.rotate(spUpimg, angle)

spDownimg = pygame.transform.rotate(spDownimg, -angle)

corner = spUpimg.get\_at((0, 0))

spUpimg.set\_colorkey(corner, RLEACCEL)

corner = spDownimg.get\_at((0, 0))

spDownimg.set\_colorkey(corner, RLEACCEL)

coin1 = pygame.image.load(coin1).convert\_alpha()

coin10 = pygame.image.load(coin10).convert\_alpha()

coin5 = pygame.image.load(coin5).convert\_alpha()

corner = coin1.get\_at((0, 0))

coin1.set\_colorkey(corner, RLEACCEL)

corner = coin10.get\_at((0, 0))

coin10.set\_colorkey(corner, RLEACCEL)

corner = coin5.get\_at((0, 0))

coin5.set\_colorkey(corner, RLEACCEL)

instr = pygame.image.load(inst).convert()

story = pygame.image.load(sto).convert()

background = pygame.image.load(back).convert()

platimg = pygame.image.load(platimg).convert\_alpha()

lava = pygame.image.load(lavaimg).convert\_alpha()

#sp2img = pygame.image.load(supermanslantimg).convert()

#Font Defintions

up = 'up'

centre = 'centre'

down = 'down'

font1 = pygame.font.SysFont('Arial', 18)

font2 = pygame.font.SysFont('Arial', 20)

font2.set\_bold(True)

font3 = pygame.font.SysFont('Arial', 40)

font4 = pygame.font.SysFont('Arial', 28)

font5 = pygame.font.SysFont('Arial', 18)

font6 = pygame.font.SysFont('Castellar', 32)

font7 = pygame.font.SysFont('Rockwell', 32)

font8 = pygame.font.SysFont('Monaco', 30)

font8.set\_bold(True)

font9 = pygame.font.SysFont('Monaco', 22)

font10 = pygame.font.SysFont('Arial', 28)

font10.set\_bold(True)

studio\_red, studio\_green, studio\_blue = 50,125,225

game\_red, game\_green, game\_blue = 175, 50, 175

#Class Definitions

class MainChar(pygame.sprite.Sprite):

""" Class MainChar determines the behavior and properties of Superman

Retrieves Image for Superman and derives from Sprite Class to check for collisions"""

flightstate = False

alive = True

def \_\_init\_\_(self):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image = spimg

self.rect = self.image.get\_rect()

self.rect.x = 100 # Initially it was 30

self.rect.y = ((scrheight + 20 )/2) - skheight

self.true\_y = 0

def update(self,dsy):

self.true\_y += dsy

def x\_y(self,sx, sy):

self.rect.x = sx

self.rect.y = sy

def dx\_dy(self,dsx, dsy):

self.rect.x += dsx

self.rect.x += dsy

def default(self):

self.image = spimg

self.rect = self.image.get\_rect()

self.rect.x = 100 # Initially it was 30

self.rect.y = ((scrheight + 20 )/2) - skheight

self.true\_y = 0

class Coins(pygame.sprite.Sprite):

"""Class Coins for coin objects and the functions, movement

and values of coins in the game. Derived from Sprite Class to detect collisions"""

clist = []

sound = pygame.mixer.Sound('coin1.wav')

ctime = time.time()

rt = 8 # time for second coin after first

def \_\_init\_\_(self,px,py,n):

pygame.sprite.Sprite.\_\_init\_\_(self)

if n == 1:

self.image = coin1

self.type = 1

elif n == 5:

self.image = coin5

self.type = 5

elif n == 10:

self.image = coin10

self.type = 10

pygame.sprite.Sprite.\_\_init\_\_(self)

self.rect = self.image.get\_rect()

self.rect.x = px

self.rect.y = py

def update(self,dpx,dpy):

self.rect.x += dpx

self.rect.y += dpy

if self in Coins.clist and self.rect.x < -(cwidth) :

Coins.clist.remove(self)

splist.remove(self)

del self

@staticmethod

def generate():

r1 = [1,2,2,3,3,3,4,4,4,5,5,5,5,6,6,6,5,5,8,8,8,8,10,12,12]

r2 = [1]\*4+[5]

rt = random.choice(r1)

Coins.rt = random.choice(r1)

Coins.ctime = time.time()

gen = False

while not(gen):

ycoin = yabs + random.randint(-100,100)

c = Coins(scrwidth+30, ycoin, random.choice(r2))

gen = Coins.checkcollide(c)

Coins.clist += [c]

@staticmethod

def checkcollide(c):

splist.add(c)

for i in Plat.plist:

for j in Plat.plist[i]:

if pygame.sprite.collide\_rect(c, j):

splist.remove(c)

return False

return True

class Plat(pygame.sprite.Sprite):

""" Class Plat for all platform objects and controls movement and image blitted"""

pos = [up, centre, down]

plist = {up : [], centre : [], down : []}

pyabs = {up : -200, centre : 0, down : 200}

endcoor = {up : 0, centre : 0, down : 0} # The ending x-coordinate of the last platform

height\_flag = {up : False, centre : False, down : False}

# This determines whether the blocks added continuously

# are added at the same height

def \_\_init\_\_(self,px,py,position):

pygame.sprite.Sprite.\_\_init\_\_(self)

self.image = platimg

self.rect = self.image.get\_rect()

self.rect.x = px

self.rect.y = py

Plat.plist[position] += [self]

Plat.endcoor[position] = px + pwidth # This gives the ending

# x-coordinate of the last added platform

def update(self,dpx,dpy):

self.rect.x += dpx

self.rect.y += dpy

for position in Plat.plist:

if self in Plat.plist[position] and self.rect.x < -(pwidth) :

Plat.plist[position].remove(self)

splist.remove(self)

del self

break

def \_\_del\_\_(self):

pass

############################################################################

#User Defined Functions

def Platgen(yabs, sigma,position):

"""Generates the each of the platforms at one position and accepts location variables"""

global Plat

# plat\_num is a list that contins the values - 1,2,3 with

# different frequencies.

num = random.choice(plat\_num) # num determines the number of plates

# that will be added continuously in one go

if random.choice(existlist):

# Here, the plates are added continuously at the same height

fixed\_height = random.gauss( yabs + Plat.pyabs[position] - 30, sigma)

for i in range(num):

platform = Plat( scrwidth + i\*pwidth , fixed\_height, position)

splist.add(platform)

Plat.height\_flag[position] = True

# Here, height\_flag is True. Thus, more plates will be added

# after leaving some space. Otherwise it becomes inconvenient.

else:

# Here, the plates are added continuously at different heights

for i in range(num):

platform = Plat( scrwidth + i\*pwidth , random.gauss( yabs + Plat.pyabs[position]- 30, sigma), position)

splist.add(platform)

Plat.height\_flag[position] = False

# Here, height\_flag is False. Thus, more plates can be added

# immediately.

def quitter(score):

"""Function to handle the quitting of game and calls for highscore and displaying it"""

global alive

alive = False

txt1 = font3.render('YOU DIED!', True, WHITE)

txt2 = font1.render('Click to continue', True, WHITE)

txt3 = font1.render('Your score is: ' + str(int(score)), True, WHITE)

scr.blit(txt1, ((scrwidth-txt1.get\_width())/2, int(scrheight/2.5)))

scr.blit(txt2, ((scrwidth-txt2.get\_width())/2, int(scrheight/2.5)+70))

scr.blit(txt3, ((scrwidth-txt3.get\_width())/2, int(scrheight/2.5+145)))

flag = True

while flag:

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

elif event.type == MOUSEBUTTONDOWN:

scr.blit(background,(0,0))

hscore(score)

dispscore()

flag = False

break

elif event.type == KEYDOWN:

if event.key == K\_RETURN:

scr.blit(background,(0,0))

hscore(score)

dispscore()

flag = False

break

pygame.display.update()

def bi(n):

"""Converts decimal number n to binary equivalent"""

b,p=0,1

while n>0:

m=n%2

b+=m\*p

p\*=10

n//=2

return b

def hscore(n):

"""Checks if new highscore is applicable, writes over

Checks for Errors and tampering by external sources as well"""

n = int(n)

while True:

try:

b = db = vc = hsr = 0

f = open("HScore.bin","rb")

d = f.read()

f.close()

if d == "#\#001":

b = hsr =001

db = 1

else:

dec,last = 0,0

for i in range(len(d)):

if d[i:i+3] == "#\#":

dec += 1

last = i

for i in range(len(d)):

if 47<ord(d[i])<58 and db == 1 and vc == 1:

hsr = hsr\*10+int(d[i])

if 47<ord(d[i])<58 and db == dec and vc == 1:

b = b\*10+int(d[i])

elif d[i:i+3] == "#\#":

db += 1

vc = 0

elif d[i:i+2]=="AF":

vc = 1

if (int(str(b//100),2)%9999999967) != 0 or (int(str(hsr//100),2)%9999999967)!=0:

raise ValueError("Tampered file!")

s = (int(str(b//100),2)/9999999967)

if (n >= s or dec < 5) and n!=0:

nam = getname()

tnam = "F/"

for i in nam:

tnam = tnam+str(bi(ord(i)\*137))+"/"

nam = tnam+"AF"

if db == 1 and s == 0:

f = open("HScore.bin","wb")

f.write("#\#"+nam+"//"+str(bi(n\*9999999967)\*100+len(str(n))))

f.close()

else:

f = open("HScore.bin","rb")

d = f.read()

f.close()

j = flag = i = 0

while i <len(d):

if d[i:i+3] == "#\#":

j = i

dec = ""

if d[i:i+4] == "AF//":

i += 4

while (d[i:i+3]!="#\#" and i<len(d)):

dec += d[i]

i += 1

dec = int(dec)

i -= 1

if (int(str(dec//100),2)/9999999967)<n:

flag = 1

break

i += 1

if db == 5:

d = d[:last]

f = open('HScore.bin',"wb")

if flag:

f.write(d[:j]+"#\#"+nam+"//"+str(bi(n\*9999999967)\*100+len(str(n)))+d[j:])

else:

f.write(d+"#\#"+nam+"//"+str(bi(n\*9999999967)\*100+len(str(n))))

f.close()

else:

scr.blit(font4.render('High score:' + str(int(str(hsr//100),2)/9999999967), True, (255,255,255)), (scrwidth//2.5-50, int(scrheight/2.5)+55))

break

except:

f = open("HScore.bin","wb")

f.write("#\#001")

f.close()

def dispscore1():

"""Function for simple viewing of highscores only

Also checks for tampering from external sources"""

while True:

try:

f = open('HScore.bin',"rb")

d = f.read()

f.close()

break

except:

f = open("HScore.bin","wb")

f.write("#\#001")

f.close

i = 0

scr.blit(background,(0,0))

sl=0

linespace = 40

scrwidth = 640

scrheight = 360

if d == "#\#001":

scr.blit(font4.render("No Scores Yet On This Device", True, (255,255,255)), (scrwidth//2.5-80, int(scrheight/2.5)))

sl = 3

else:

l = []

size = sznstr = 0

while i < len(d):

if d[i] == "#":

sl += 1

nstr = b = ""

i += 5

while d[i] != "A":

while d[i].isdigit():

b += d[i]

i += 1

nstr += chr(int(b,2)/137)

i += 1

b = ''

i += 4

while i < len(d):

b += d[i]

if i+1==len(d) or d[i] == "#":

b = b[:-3]

i-=2

break

i+=1

b = int(b,2)/9999999967

if i == len(d) -3:

b \*= 2

l += [[sl , nstr , b]]

txt1 = font4.render(str(sl)+nstr + str(b), True, (255,255,255))

if size < sznstr:

sznstr = font4.render(nstr, True, (255,255,255)).get\_width()

size = sznstr

txt2 = font4.render(str(sl)+nstr + str(b), True, (255,255,255))

if size < txt1.get\_width():

size = txt1.get\_width()+3

sznstr = font4.render(nstr, True, (255,255,255)).get\_width()

centre = scrwidth//2

if i == len(d) -3:

for j in l:

if j[0] == 1:

txt1 = font4.render("SL No.", True, (255,255,255))

txt2 = font4.render("NAME", True, (255,255,255))

txt3 = font4.render("SCORE", True, (255,255,255))

half = sznstr//2

nsx = (centre - half)+(sznstr-txt2.get\_width())/2

scr.blit(txt1, (nsx-90, int(scrheight/4.5)-20))

scr.blit(txt2, (((nsx+10, int(scrheight/4.5)-20))))

scr.blit(txt3, (nsx+half+60, int(scrheight/4.5)-20))

txt4 = font4.render(str(j[0]), True, (255,255,255))

txt5 = font4.render(j[1], True, (255,255,255))

txt6 = font4.render(str(j[2]), True, (255,255,255))

half = (txt5.get\_width())//2

nsx = (centre - half)+(sznstr-txt5.get\_width())/2

scr.blit(txt4, (nsx-58, int(scrheight/4.5)+j[0]\*linespace-15))

scr.blit(txt5, (nsx, int(scrheight/4.5)+j[0]\*linespace-15))

scr.blit(txt6, (nsx+half+80, int(scrheight/4.5)+j[0]\*linespace-15))

i += 1

scr.blit(font1.render('Click to continue', True, WHITE), (scrwidth//2.5+23, int(scrheight/4.5)+(sl+1)\*(linespace)))

pygame.display.update()

flag = True

while flag:

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

elif event.type == MOUSEBUTTONDOWN:

flag = False

break

elif event.type == KEYDOWN:

if event.key == K\_RETURN:

flag = False

break

def dispscore():

"""Function for simple viewing of highscores only

Also checks for tampering from external sources"""

while True:

try:

f = open('HScore.bin',"rb")

d = f.read()

f.close()

break

except:

f = open("HScore.bin","wb")

f.write("#\#001")

f.close

i = 0

scr.blit(background,(0,0))

sl=0

linespace = 40

scrwidth = 640

scrheight = 360

if d == "#\#001":

scr.blit(font4.render("No Scores Yet On This Device", True, (255,255,255)), (scrwidth//2.5-80, int(scrheight/2.5)))

sl = 3

else:

l1,l2 = ["NAME"], ["SCORE"]

l = [["SL No.","NAME","SCORE"]]

size = sznstr = 0

while i < len(d):

if d[i] == "#":

sl += 1

nstr = b = ""

i += 5

while d[i] != "A":

while d[i].isdigit():

b += d[i]

i += 1

nstr += chr(int(b,2)/137)

i += 1

b = ''

i += 4

while i < len(d):

b += d[i]

if i+1==len(d) or d[i] == "#":

b = b[:-3]

i-=2

break

i+=1

b = int(b,2)/9999999967

if i == len(d) -3:

b \*= 2

l += [[sl , nstr , b]]

l1 += [nstr]

l2 += [str(b)]

i += 1

centre = scrwidth//2

bsx = bnx = 0

for i in range(len(l1)):

if font4.render(l2[i], True, (255,255,255)).get\_width() > bsx:

bsx = font4.render(l2[i], True, (255,255,255)).get\_width()

if font4.render(l1[i], True, (255,255,255)).get\_width() > bnx:

bnx = font4.render(l1[i], True, (255,255,255)).get\_width()

halfn = bnx/2

halfs = bsx/2

for j in l:

if j[0] == "SL No.":

txt1 = font10.render("SL No.", True, (255,255,255))

txt2 = font10.render("NAME", True, (255,255,255))

txt3 = font10.render("SCORE", True, (255,255,255))

nsx = centre - halfn

cn = (bnx-txt2.get\_width())/2

nnx = centre +halfn

cs = (bsx-txt3.get\_width())/2

scr.blit(txt1, (nsx-110, int(scrheight/4.5)-20))

scr.blit(txt2, (((nsx + cn, int(scrheight/4.5)-20))))

scr.blit(txt3, (nnx+cs+50, int(scrheight/4.5)-20))

continue

txt4 = font4.render(str(j[0])+".", True, (255,255,255))

txt5 = font4.render(j[1], True, (255,255,255))

txt6 = font4.render(str(j[2]), True, (255,255,255))

nsx = centre - halfn

cn = (bnx-txt5.get\_width())/2

nnx = centre + halfn

cs = (bsx-txt6.get\_width())/2

scr.blit(txt4, (nsx-80, int(scrheight/4.5)+j[0]\*linespace-15))

scr.blit(txt5, (nsx+cn, int(scrheight/4.5)+j[0]\*linespace-15))

scr.blit(txt6, (nnx+cs+53, int(scrheight/4.5)+j[0]\*linespace-15))

scr.blit(font1.render('Click to continue', True, WHITE), (scrwidth//2.5+23, int(scrheight/4.5)+(sl+1)\*(linespace)))

pygame.display.update()

flag = True

while flag:

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

elif event.type == MOUSEBUTTONDOWN:

flag = False

break

elif event.type == KEYDOWN:

if event.key == K\_RETURN:

flag = False

break

def getname():

"""Function to retreive name from pygame window with input inside a box"""

nstr = key = ''

blink = ["|"," "]

bl = 0

blt = time.time()

while True:

txt = font4.render(nstr + blink[bl], 1, (255,255,255))

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

elif event.type == KEYDOWN:

key = event.key

if key == K\_RETURN:

return nstr

elif key == K\_BACKSPACE:

nstr = nstr[:-1]

elif key<=127:

twidth = txt.get\_width()

if twidth < 224:

nstr += chr(key)

else:

nstr = nstr[:-1] + chr(key)

if time.time() - blt > 0.4 :

bl += 1

bl %= 2

blt = time.time()

txt = font4.render(nstr + blink[bl], 1, (255,255,255))

scr.blit(background,(0,0))

scr.blit(font4.render('You got a new high score!', True, WHITE), (scrwidth//2.5-70, int(scrheight/3.5)))

scr.blit(font4.render("Enter your Name:", 1, WHITE),((scrwidth / 2) - 240, (scrheight / 2)-13))

pygame.draw.rect(scr, (0,0,0),((scrwidth / 2) - 50,(scrheight / 2) - 10,230,32), 0)

pygame.draw.rect(scr, (255,255,255),((scrwidth / 2) - 50,(scrheight/ 2) - 12,230,34), 1)

nstr = nstr.title()

#Blits the flashing line

scr.blit(txt,((scrwidth / 2) - 47, (scrheight / 2) - 14))

pygame.display.update()

def pause():

""" Function to handle the game when paused by user"""

scr.blit(background,(0,0))

pygame.draw.polygon(scr, BLACK, ((fl\_x, fl\_y),

(fl\_x + flwidth, fl\_y), (fl\_x + flwidth, fl\_y + flheight),

(fl\_x, fl\_y + flheight)))

pygame.draw.polygon(scr, barcolour, ((fl\_x + 2, fl\_y + 1),

(fl\_x + greenwidth + 2, fl\_y), (fl\_x + greenwidth + 2, fl\_y + flheight - 1),

(fl\_x + 2, fl\_y + flheight - 1)))

scr.blit(font2.render('STAMINA', True, (255,255,255)), (fl\_x+flwidth/4, fl\_y))

scr.blit(font2.render('SCORE: ', True, (255,0,0)), (scrwidth - 150, 20))

scr.blit(font1.render(str(int(score)), True, (255,0,0)), (scrwidth - 150 + 80, 22)) ###

scr.blit(lava, (xlava, ylava))

txt1 = font3.render('GAME PAUSED', True, WHITE)

txt2 = font1.render('Click to continue', True, WHITE)

scr.blit(txt1, ((scrwidth-txt1.get\_width())/2, int(scrheight/2.5)))

scr.blit(txt2, ((scrwidth-txt2.get\_width())/2, int(scrheight/2.5)+70))

pygame.display.update()

paused = True

while paused:

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == KEYDOWN:

if event.key in [K\_q,K\_ESCAPE]:

scr.blit(background,(0,0))

quitter(score)

elif event.key in [K\_p]:

paused = False

elif event.key == K\_RETURN:

paused = False

if event.type == MOUSEBUTTONDOWN:

paused = False

intern

def list\_gen():

"""Generates the list with random length"""

global length\_list

notyet = []

for i in xrange(0,scrwidth, box\_width):

for j in xrange(0,scrheight, box\_width):

notyet.append([i,j])

while notyet:

j = random.choice(notyet)

notyet.remove(j)

length\_list -= 1

yield j

def pixelator():

"""Function for initial animation of pixels appearing"""

clock = pygame.time.Clock()

fill\_num = 2

pix = list\_gen()

prev\_length = length\_list

esc\_flag = False

try:

fps = 500

while True and not esc\_flag:

if length\_list <= int(prev\_length\*0.25):

fill\_num += 1

prev\_length = length\_list

for i in xrange(fill\_num):

j = pix.next()

a, b = j[0], j[1]

red = random.randint(0,255)

green = random.randint(0,255)

blue = random.randint(0,255)

pygame.draw.polygon(scr, (red, green, blue),

((a,b),

(a + box\_width -1, b), (a + box\_width -1, b + box\_width -1),

(a, b + box\_width -1)))

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

elif event.type == KEYDOWN:

if event.key == [K\_ESCAPE, K\_SPACE]:

esc\_flag = True

if event.type == MOUSEBUTTONDOWN:

esc\_flag = True

pygame.display.update()

clock.tick(fps)

except StopIteration:

pass

fps = 50

i = 0

while i<= scrwidth/2 and not esc\_flag:

left, right = scrwidth/2 - i, scrwidth/2 + i

up = int(left \* float(scrheight)/scrwidth)

down = int(right \* float(scrheight)/scrwidth)

pygame.draw.polygon(scr, (0, 0, 0),

((left,up), (right, up), (right, down), (left, down)))

i += 3

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == KEYDOWN:

if event.key in [K\_ESCAPE, K\_SPACE]:

esc\_flag = True

if event.type == MOUSEBUTTONDOWN:

esc\_flag = True

pygame.display.update()

clock.tick(fps)

pygame.draw.polygon(scr, (0,0,0),((0,0),(scrwidth, 0),

(scrwidth, scrheight),(0, scrheight)))

pygame.display.update()

def logo():

"""Displays GA Studios Logo in different colours"""

red, green, blue = studio\_red, studio\_green, studio\_blue

steps = 15.0

txt = font6.render('GA Studios'+' '+chr(169), True,

(0, 0, 0))

txt\_width = txt.get\_width()

txt\_height = txt.get\_height()

del txt

clock = pygame.time.Clock()

fps = 10

rcount = gcount = bcount = 0

esc\_flag = False

while True and not esc\_flag:

pygame.draw.polygon(scr, (0,0,0), ((int(scrwidth\*0.3), int(scrheight\*0.4)),

(int(scrwidth\*0.7), int(scrheight\*0.4)),

(int(scrwidth\*0.7), int(scrheight\*0.6)),

(int(scrwidth\*0.3), int(scrheight\*0.6))))

pygame.display.update()

if rcount <= int(steps):

scr.blit(font6.render('GA Studios'+' '+chr(169), True,

(int(red \* rcount/steps), 0, 0)),

((scrwidth - txt\_width)/2, (scrheight - txt\_height)/2))

pygame.display.update()

rcount += 1

elif gcount <= int(steps):

scr.blit(font6.render('GA Studios'+' '+chr(169), True,

(red, int(green \* gcount/steps), 0)),

((scrwidth - txt\_width)/2, (scrheight - txt\_height)/2))

pygame.display.update()

gcount += 1

elif bcount <= int(steps):

scr.blit(font6.render('GA Studios'+' '+chr(169), True,

(red, green, int(blue \* bcount/steps))),

((scrwidth - txt\_width)/2, (scrheight - txt\_height)/2))

pygame.display.update()

bcount += 1

else:

break

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == KEYDOWN:

if event.key in [K\_ESCAPE, K\_SPACE]:

esc\_flag = True

if event.type == MOUSEBUTTONDOWN:

esc\_flag = True

clock.tick(fps)

pygame.draw.polygon(scr, (0,0,0),((0,0),(scrwidth, 0),

(scrwidth, scrheight),(0, scrheight)))

scr.blit(font6.render('GA Studios'+' '+chr(169), True,

(red, green, blue)),

((scrwidth - txt\_width)/2, (scrheight - txt\_height)/2))

pygame.display.update()

time.sleep(0.5)

def instructions():

"Displays the intructions and the story"

scr.blit(story, (0,0))

pygame.display.update()

esc\_flag = False

clock = pygame.time.Clock()

fps = 60

while not esc\_flag:

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == KEYDOWN:

if event.key in [K\_ESCAPE, K\_SPACE, K\_RETURN]:

esc\_flag = True

if event.type == MOUSEBUTTONDOWN:

pass

clock.tick(fps)

scr.blit(instr, (0,0))

pygame.display.update()

esc\_flag = False

while not esc\_flag:

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == KEYDOWN:

if event.key in [K\_ESCAPE, K\_SPACE, K\_RETURN]:

esc\_flag = True

if event.type == MOUSEBUTTONDOWN:

pass

def menu():

"""Main function to handle menu and subsequent input from user"""

pygame.draw.polygon(scr, (0,0,0),((0,0),(scrwidth, 0),

(scrwidth, scrheight),(0, scrheight)))

txt = font6.render('GA Studios'+' '+chr(169), True,

(studio\_red, studio\_green, studio\_blue))

txt\_width = txt.get\_width()

txt\_height = txt.get\_height()

scr.blit(txt, ((scrwidth - txt\_width)/2, (scrheight - txt\_height)/2 + 100))

game\_txt = font7.render('Superbot', True,

(game\_red, game\_green, game\_blue))

game\_width, game\_height = game\_txt.get\_width(), game\_txt.get\_height()

scr.blit(game\_txt, ((scrwidth - game\_width)/2,

(scrheight - 150 - game\_height)/2))

pygame.display.update()

time.sleep(1)

menu\_list = ['Play', 'Instructions', 'High scores', 'Exit']

menu\_state = prev\_state= 0

st\_left = ' '\*12

st\_right = menu\_list[1]

st\_current = menu\_list[0]

superman.x\_y(-skwidth, scrheight/2 - 150)

dsx, dsy = 4,0

clock = pygame.time.Clock()

fps = 60

col\_1 = (200,100,100)

col\_2 = (220,170,120)

start\_play = False

txt1 = font1.render('Use SPACE or ENTER key to select', True, (175,175,100))

while True:

pygame.draw.polygon(scr, (0,0,0),((0,0),(scrwidth, 0),

(scrwidth, scrheight),(0, scrheight)))

if start\_play:

break

scr.blit(superman.image, (superman.rect.x,superman.rect.y))

scr.blit(txt, ((scrwidth - txt\_width)/2,

(scrheight - txt\_height)/2 + 100))

scr.blit(game\_txt, ((scrwidth - game\_width)/2,

(scrheight - 150 - game\_height)/2))

scr.blit(txt1, (20,330))

if menu\_state != prev\_state:

prev\_state = menu\_state

st\_current = menu\_list[menu\_state]

if menu\_state != 0:

st\_left = menu\_list[menu\_state-1]

else:

st\_left = ' '\*12

try:

st\_right = menu\_list[menu\_state+1]

except IndexError:

st\_right = ' '\*12

txt\_left = font9.render(st\_left, True, col\_1)

scr.blit(txt\_left, (scrwidth/5 - txt\_left.get\_width()/2,

(scrheight - txt\_left.get\_height())/2))

txt\_right = font9.render(st\_right, True, col\_1)

scr.blit(txt\_right, ((scrwidth\*4)/5 - txt\_right.get\_width()/2,

(scrheight - txt\_right.get\_height())/2))

txt\_current = font8.render(st\_current, True, col\_2)

scr.blit(txt\_current, ((scrwidth - txt\_current.get\_width())/2,

(scrheight - txt\_current.get\_height())/2))

if superman.rect.x > scrwidth:

superman.x\_y(-skwidth, scrheight/2 - 150)

superman.dx\_dy(dsx, dsy)

pygame.display.update()

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == KEYDOWN:

if event.key == K\_RIGHT:

if menu\_state < 3:

menu\_state += 1

if event.key == K\_LEFT:

if menu\_state > 0:

menu\_state -= 1

if event.key in [K\_RETURN, K\_SPACE]:

if menu\_state == 0:

start\_play = True

superman.default()

elif menu\_state == 1:

instructions()

elif menu\_state == 2:

dispscore()

elif menu\_state == 3:

pygame.quit()

sys.exit()

if event.type == MOUSEBUTTONDOWN:

pass

clock.tick(fps)

############################################################################

"""Data Definitions at the start of the program"""

superman = MainChar()

splist = pygame.sprite.Group()

splist.add(superman)

box\_width = 5

length\_list = 0

fl\_x = fl\_y = 20

flwidth = 200

flheight = 20

maxflytime = 2.0 # Fly Key is f

flydivconst = 3.0 # The flybar will get charged at 1/flydivconst times the

# speed with fly gets used up

clock = pygame.time.Clock()

yabs = skheight+superman.rect.y-1 #starting height of platforms (mean of Gaussian Distribution)

sigma = 20 #Standard deviation for the Gaussian distribution of platforms

# sigma was initially 150

pixelator()

logo()

ingame = True

"""Loop to run entire game begins"""

while ingame:

#Data Definitions

superman = MainChar()

splist = pygame.sprite.Group()

splist.add(superman)

Coins.clist = []

fps = 60

dx = -5

dy = fy = 0

y\_coord = ((scrheight + 20 )/2) - skheight

xlava = 0

ylava = scrheight+220 #200 pixels below bottom of screen

minylava = 290 # lava rises to this height

deathsound = pygame.mixer.Sound('lava.wav')

deathsoundstate = True

for i in range(10):

for position in Plat.pos:

platform = Plat(i\*pwidth, yabs, position)

splist.add(platform)

st = time.time()

fsj = False

#Platform variables

existlist = [0 for i in range(13)] + [1 for i in range(3)] # This is used to

#determine whether a plate should be added or not

gap\_length = {up : 0, centre : 0, down : 0}

# if a plate is not added, then gap\_length is added

# to endcoor (ending coordinate), so, there will be some

# space before the next plate is added

plat\_num = [1 for i in range(11)] + [2 for i in range(5)] + [3 for i in range(2)]

# This has the values for the number of plates that should be added

# continuously in one go

last\_platform = {} # Stores the last platform of each level

#Flying

flybar = 0.0 # Show how much fly you have left

flyflag = False # To check whether superman is flying

prevflytime = time.time()

jump = 0

score = 0

dist = 0

coincount = 0

newcoin = False

coinscore = 500

alive = True

menu()

c1 = Coins(10\*pwidth-10,yabs-35, 1)

Coins.clist += [c1]

Coins.ctime = time.time()

splist.add(c1)

superman.default()

pygame.sprite.spritecollide

"""Loop for inividual gameplay begins"""

while alive:

scr.blit(background,(0,0))

splist.draw(scr)

scr.blit(superman.image, (superman.rect.x,superman.rect.y))

dist += abs(dx)

if newcoin:

score += (abs(dx))\*\*1.2 \* (coincount)\*coinscore

coincount = 0

newcoin = False

else:

score += (abs(dx))\*\*1.2

for i in Coins.clist:

if pygame.sprite.collide\_rect(superman, i):

Coins.sound.play()

newcoin = True

coincount = i.type

Coins.clist.remove(i)

splist.remove(i)

del i

if time.time() - Coins.ctime > Coins.rt:

Coins.generate()

# Fly Bar Code Starts

# Checking how long superman can fly

flybarfraction = flybar/maxflytime

if flybarfraction > 1:

flybarfraction = 1

elif flybarfraction < 0:

flybarfraction = 0

pygame.draw.polygon(scr, BLACK, ((fl\_x, fl\_y),

(fl\_x + flwidth, fl\_y), (fl\_x + flwidth, fl\_y + flheight),

(fl\_x, fl\_y + flheight)))

greenwidth = int((flwidth-4) \* flybarfraction)

if flybar > 1.25:

barcolour = GREEN

elif 1.25 > flybar > 0.45:

barcolour = YELLOW

else:

barcolour = RED

pygame.draw.polygon(scr, barcolour, ((fl\_x + 2, fl\_y + 1),

(fl\_x + greenwidth + 2, fl\_y), (fl\_x + greenwidth + 2, fl\_y + flheight - 1),

(fl\_x + 2, fl\_y + flheight - 1)))

scr.blit(font2.render('STAMINA', True, (255,255,255)), (fl\_x+flwidth/4, fl\_y))

scr.blit(font2.render('SCORE: ', True, (255,0,0)), (scrwidth - 150, 20))

scr.blit(font1.render(str(int(score)), True, (255,0,0)), (scrwidth - 150 + 80, 22)) ###

scr.blit(lava, (xlava, ylava))

pygame.display.update()

y\_coord += dy

ylava += dy

if y\_coord < -100:

superman.rect.y += 5

if deathsoundstate and superman.rect.y > (scrheight-130):

deathsound.play()

deathsoundstate = False

elif superman.rect.y > (scrheight+10):

#code to die

quitter(score)

#alive = False

break

if not flyflag:

if flybar < maxflytime:

flybar += 1.0/(fps \* flydivconst)

else:

flybar = maxflytime

if flyflag:

if jump == 0:

jump = 1

if flybar <= 0:

flybar = 0

flyflag = False

dy = -2

prevflytime = time.time()

else:

flybar -= 1.0/(fps)

#Fly Bar Code Ends Here

if dy == 2 and time.time()-uptime >= 0.8:

superman.image = spDownimg

dy = -2

uptime = 0

if dy != 2 and not(flyflag):

dy = -2

superman.image = spDownimg

for a in Plat.plist:

for i in Plat.plist[a]:

if i.rect.x-skwidth-6 <= superman.rect.x <= (i.rect.x+pwidth+6) and -3<i.rect.y -(superman.rect.y+skheight)<3 :

dy = 0

jump = 0

flyflag = False

fsj = False

superman.image = spimg # To change images while moving straight, moving up and moving down

break

yabs += dy

for i in Plat.plist:

for j in Plat.plist[i]:

j.update(dx,dy)

for i in Coins.clist:

i.update(dx,dy)

superman.update(dy)

#Generating new platforms

for position in Plat.pos:

#last\_platform[position] = Plat.plist[position][-1]

last\_platform = Plat.plist[position][-1]

Plat.endcoor[position] = last\_platform.rect.x + pwidth + gap\_length[position]

# gap\_length is added if a gap should be inserted

if Plat.endcoor[position] - scrwidth <= 0:

if random.choice(existlist) and not Plat.height\_flag[position]:

# In this case, platform(s) will be added

Platgen(yabs, sigma, position)

gap\_length[position] = 0

else:

# In this case, gap(s) will be added

gap\_length[position] = pwidth \* random.choice(plat\_num)

if Plat.height\_flag[position] == True:

Plat.height\_flag[position] = False

#Retrieving user input from Keypress and Mouse Actions

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

if event.type == KEYDOWN and deathsoundstate:

if event.key == K\_SPACE and deathsoundstate:

if (dy == 0 or fsj) and jump < 2:

dy = 2

jump += 1

uptime = time.time()

fsj = not(fsj)

superman.image = spUpimg # Moving Up

elif (dy == -2 or fsj) and jump < 2:

dy = 2

jump = 2

uptime = time.time()

fsj = not(fsj)

superman.image = spUpimg # Moving Up

elif event.key in [K\_q,K\_ESCAPE]:

quitter(score)

elif event.key in [K\_p]:

pause()

elif event.key in [K\_w]:

while True:

pass

elif event.key in (K\_f,) and flybar > 0: # I thought (K\_f, K\_F)

dy = 0

flyflag = True

startflytime = time.time()

superman.image = spimg # Changing it back to the normal image

if event.type == KEYUP and deathsoundstate:

if flyflag:

dy = -2

flyflag = False

superman.image = spDownimg

if event.type == MOUSEBUTTONDOWN and deathsoundstate:

if dy == 0 and jump < 2:

dy = 2

jump += 1

uptime = time.time()

superman.image = spUpimg

elif dy == -2 and jump < 2:

dy = 2

jump = 2

uptime = time.time()

superman.image = spUpimg

#Ensures the screen updates at fps frames per second

clock.tick(fps) “””

***Screenshots***

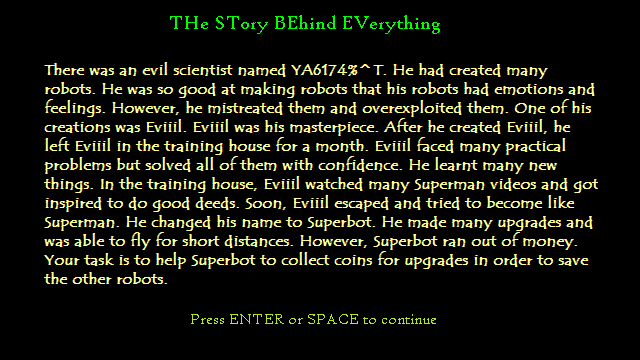
Start-up Screen:



Menu:



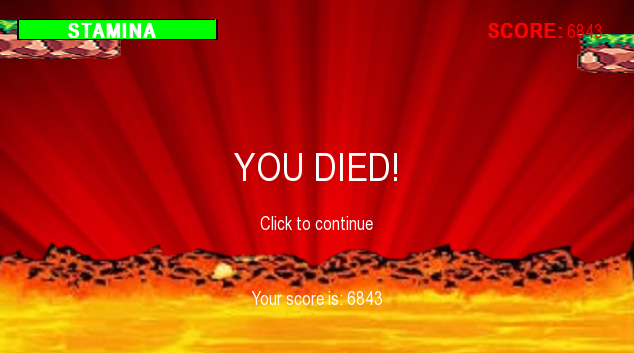
Instructions and Storyline:



Main Loop of Game:



Screen after loss:



HighScore List:



Bibliography

Websites:

1. Google.com
2. Python.org
3. Pygame.org
4. Inventwithpython.com
5. Stackoverflow.com
6. Editphotosforfree.com
7. Twistedwave.com

Books and PDFs:

1. Invent Your Own Computer Games With Python 3rd Edition ; Authored by Al Sweigart
2. Making Games with Python & Pygame ; Authored by Al Sweigart