**Code Page**

*'''*

*Created on May 7, 2014*

**@author:** *brewsterc*

*'''*

# Conner Joseph Brewster, Devon S. Chao, Jonathan V. Nguyen

# Friday, May 23rd, 2014 @ 3:11:57 PM Central Standard Time (Daylight Savings Time)

# Pre-AP Computer Science

# Annette Walter - 7th Period

# Speed Snail

# Final Project

# Base code built off of the Dodger and Escape the Dragon programs.

#import needed modules

import pygame, random, sys, time

from pygame.locals import \*

#initialize constant variables

window\_width = 1000

window\_height = 600

text\_color = (0, 0, 0)

background\_color = (0, 0, 0)

frames\_per\_second = 120

missile\_minimum\_size = 10

missile\_maximum\_size = 40

missile\_minimum\_speed = 1

missile\_maximum\_speed = 4

level = 1

def **terminate**():

*""" Create a way to exit the program. """*

pygame.quit()

sys.exit()

def **waitForPlayerToPressKey**():

*""" Waits for the player to press a keyboard key. """*

while True:

for event in pygame.event.get():

if event.type == QUIT:

terminate()

if event.type == KEYDOWN:

if event.key == K\_ESCAPE: #pressing escape quits

terminate()

elif event.key == K\_SPACE:

return

def **playerHasHitMissile**(snailRect, missiles):

*""" Detects whether or not a missile has hit Speed Snail. """*

for m in missiles:

if snailRect.colliderect(m[*'rect'*]):

return True

return False

def **playerHasHitSpeedPowerUp**(snailRect, speed\_power\_ups):

*""" Detects whether or not Speed Snail collides with a speed power-up. """*

if snailRect.colliderect(speed\_power\_upRect):

return True

return False

def **playerHasHitPointsPowerUp**(snailRect, points\_power\_ups):

*""" Detects whether or not Speed Snail collides with a points power-up. """*

if snailRect.colliderect(points\_power\_upRect):

return True

return False

def **drawText**(text, font, surface, x, y):

*""" Renders all variants of text. """*

textobj = font.render(text, 1, text\_color)

textrect = textobj.get\_rect()

textrect.topleft = (x, y)

surface.blit(textobj, textrect)

#format the in-game sprites.

snailImage = pygame.image.load(*'speed\_snail.png'*)

snailRect = snailImage.get\_rect()

missileImage = pygame.image.load(*'salt.png'*)

speed\_power\_upImage = pygame.image.load(*"speed\_power\_up\_orb.png"*)

speed\_power\_upRect = speed\_power\_upImage.get\_rect()

points\_power\_upImage = pygame.image.load(*"points\_power\_up\_orb.png"*)

points\_power\_upRect = points\_power\_upImage.get\_rect()

#setup the pygame basics.

pygame.init()

mainClock = pygame.time.Clock()

windowSurface = pygame.display.set\_mode((window\_width, window\_height))

pygame.display.set\_caption(*'Speed Snail'*)

pygame.mouse.set\_visible(False)

# set up sounds

power\_up\_sound = pygame.mixer.Sound(*'orb\_collect.wav'*)

collision\_sound = pygame.mixer.Sound(*'missile\_collision.wav'*)

pygame.mixer.music.load(*'background\_music.mp3'*)

#allocate the font.

font = pygame.font.SysFont(None, 48)

#load the background image.

bg = pygame.image.load(*"background.png"*).convert\_alpha()

begin\_bg = pygame.image.load(*"beginning\_screen.png"*).convert\_alpha()

start\_bg = pygame.image.load(*"starting\_screen.png"*).convert\_alpha()

instruct\_bg = pygame.image.load(*"instructions\_screen.png"*).convert\_alpha()

#create the starting screen sequence.

windowSurface.blit(begin\_bg, (0,0))

pygame.display.update()

waitForPlayerToPressKey()

windowSurface.blit(instruct\_bg, (0,0))

pygame.display.update()

waitForPlayerToPressKey()

windowSurface.blit(start\_bg, (0,0))

pygame.display.update()

waitForPlayerToPressKey()

#set high score to zero at game launch.

hi\_score = 0

while True:

#set up the start of the game.

facing = *"right"*

missiles = []

if level == 1:

windowSurface.blit(bg, (0,0))

elif level == 2:

windowSurface.blit(bg, (0,0))

if level == 1:

score = 0

elif level == 2:

score = 10000

score\_multiplyer = 1

speed\_snail\_speed = 2

points\_power\_up\_value = 500

missile\_spawning\_rate = 50

death = False

endless\_mode = False

snailRect.topleft = (400, 400)

speed\_power\_upRect.topleft = (random.randint(200,950), random.randint(0,400))

points\_power\_upRect.topleft = (random.randint(200,950), random.randint(0,400))

moveLeft = moveRight = moveUp = moveDown = False

missile\_add\_counter = 0

pygame.mixer.music.play(-1, 0.0)

while True: #the game loop runs while the game part is playing.

score += score\_multiplyer

for event in pygame.event.get():

if event.type == QUIT:

terminate()

if event.type == KEYDOWN:

#set parameters for when a keyboard key is pressed.

if event.key == K\_LEFT or event.key == ord(*'a'*):

facing = *"left"*

snailImage = pygame.image.load(*'speed\_snail\_left.png'*)

moveRight = False

moveLeft = True

if event.key == K\_RIGHT or event.key == ord(*'d'*):

facing = *"right"*

snailImage = pygame.image.load(*'speed\_snail.png'*)

moveLeft = False

moveRight = True

if event.key == K\_UP or event.key == ord(*'w'*):

if facing == *"right"*:

snailImage = pygame.image.load(*'speed\_snail\_up.png'*)

else:

snailImage = pygame.image.load(*'speed\_snail\_up\_left.png'*)

moveDown = False

moveUp = True

if event.key == K\_DOWN or event.key == ord(*'s'*):

if facing == *"right"*:

snailImage = pygame.image.load(*'speed\_snail\_down.png'*)

else:

snailImage = pygame.image.load(*'speed\_snail\_down\_left.png'*)

moveUp = False

moveDown = True

if event.type == KEYUP:

#set parameters for when a keyboard key is released.

if event.key == K\_ESCAPE:

terminate()

if event.key == K\_LEFT or event.key == ord(*'a'*):

snailImage = pygame.image.load(*'speed\_snail\_left.png'*)

moveLeft = False

if event.key == K\_RIGHT or event.key == ord(*'d'*):

snailImage = pygame.image.load(*'speed\_snail.png'*)

moveRight = False

if event.key == K\_UP or event.key == ord(*'w'*):

moveUp = False

if event.key == K\_DOWN or event.key == ord(*'s'*):

moveDown = False

if moveLeft == False and moveRight == False and moveUp == False and moveDown == False:

if facing == *"right"*:

snailImage = pygame.image.load(*'speed\_snail.png'*)

else:

snailImage = pygame.image.load(*'speed\_snail\_left.png'*)

#add new missiles at the top of the screen, if needed.

missile\_add\_counter += 1

if missile\_add\_counter == missile\_spawning\_rate:

missile\_add\_counter = 0

missilesize = random.randint(missile\_minimum\_size, missile\_maximum\_size)

newMissile = {*'rect'*: pygame.Rect(0 - missilesize, random.randint(0, 500-missilesize), missilesize, missilesize),*'speed'*: random.randint(missile\_minimum\_speed, missile\_maximum\_speed),*'surface'*:pygame.transform.scale(missileImage, (missilesize, missilesize)),}

missiles.append(newMissile)

#move the player around.

if moveLeft and snailRect.left > 0:

snailRect.move\_ip(-1 \* speed\_snail\_speed, 0)

if moveRight and snailRect.right < window\_width:

snailRect.move\_ip(speed\_snail\_speed, 0)

if moveUp and snailRect.top > 0:

snailRect.move\_ip(0, -1 \* speed\_snail\_speed)

if moveDown and snailRect.bottom < 490:

snailRect.move\_ip(0, speed\_snail\_speed)

#move the missiles down.

for m in missiles:

m[*'rect'*].move\_ip(m[*'speed'*], 0)

#delete missiles that have gone past the screen.

for m in missiles[:]:

if m[*'rect'*].left > window\_width:

missiles.remove(m)

#draw the background.

if level == 1:

windowSurface.blit(bg, (0,0))

elif level == 2:

windowSurface.blit(bg, (0,0))

#draw the score and top score.

drawText(*'SCORE: %s'* % (score), font, windowSurface, 10, 550)

drawText(*'HI-SCORE: %s'* % (hi\_score), font, windowSurface, 290, 550)

if endless\_mode == True:

drawText(*'Endless Mode: Unlocked'*, font, windowSurface, 600, 550)

#draw the player's rectangle.

windowSurface.blit(snailImage, snailRect)

#draw the speed power-up's rectangle.

windowSurface.blit(speed\_power\_upImage, speed\_power\_upRect)

#draw the points power-up's rectangle.

windowSurface.blit(points\_power\_upImage, points\_power\_upRect)

#draw each missile.

for m in missiles:

windowSurface.blit(m[*'surface'*], m[*'rect'*])

pygame.display.update()

#check if any of the missiles have hit the player.

if playerHasHitMissile(snailRect, missiles):

if score > hi\_score:

hi\_score = score # set new high score.

collision\_sound.play()

death = True

break

#check if Speed Snail collects a speed power-up.

if playerHasHitSpeedPowerUp(snailRect, speed\_power\_upRect):

speed\_power\_up\_status = True

power\_up\_sound.play()

speed\_snail\_speed += 1

score\_multiplyer += 1

speed\_power\_upRect.topleft = (random.randint(200,1000), random.randint(0,400))

#check if Speed Snail collects a points power-up.

if playerHasHitPointsPowerUp(snailRect, points\_power\_upRect):

power\_up\_sound.play()

score += points\_power\_up\_value

points\_power\_up\_value += 500

points\_power\_upRect.topleft = (random.randint(200,1000), random.randint(0,400))

#check for score necessary to move on to level two.

if level == 1:

if score > 9999:

break

else:

if score > 24999:

endless\_mode = True

#setup frames per second.

mainClock.tick(frames\_per\_second)

#finalize game over screen.

if death == True:

drawText(*'You died!'*, font, windowSurface, (window\_width / 3), (window\_height / 3))

drawText(*'Press SPACEBAR to try again.'*, font, windowSurface, (window\_width / 3) - 80, (window\_height / 3) + 50)

if level == 2:

score = 10000

pygame.display.update()

waitForPlayerToPressKey()

elif level == 1:

drawText(*'Entering level two...'*, font, windowSurface, (window\_width / 3), (window\_height / 3))

drawText(*'Press SPACEBAR to continue.'*, font, windowSurface, (window\_width / 3) - 80, (window\_height / 3) + 50)

bg = pygame.image.load(*"background2.png"*).convert\_alpha()

missileImage = pygame.image.load(*'flame.png'*)

level = 2

pygame.display.update()

waitForPlayerToPressKey()