**Introduction**

We set out to make our own rendition of the classical arcade game, ‘Snake’. In our game, we made use of the fundamentals of Python taught to us over the course of the last two years. We have incorporated classes, functions, type binding, while, for loops, exception handling et cetera. To make our game look modern, we made use of the popular module Pygame that does not come pre-installed on Python. To learn, Pygame, we made use of a series of YouTube videos and different websites.  
  
In the actual game, the objective is to control the snake using arrow keys and to eat the apple that appears at random coordinates on the screen and to beat the high score previously set by one of the earlier users or developers. If a user has beaten the high score, their name and score is saved securely onto a text file.

**Requirements**

**System Requirements**1. Pentium Processor or higher with 2GB of RAM.  
2. Mouse  
3. Keyboard  
4. Colour Monitor  
  
**Software Requirements**1. Operating system - Windows XP or higher.  
2. Python 2.7.x only.

**Modules and Functions:**

|  |  |
| --- | --- |
| **Module** | **Function** |
| pygame | pygame.init() , pygame.font.SysFont() ,pygame.image.load() ,  pygame.mixer.music.load() , pygame.mixer.music.play(), pygame.transform.rotate(), pygame.draw.rect(), pygame.display.set\_mode(), pygame.display.set\_caption(), pygame.time.Clock(), pygame.display.update(), pygame.event.get() |
| time | time.sleep() |
| random | random.randint() |

**Functions and Tasks:**

|  |  |
| --- | --- |
| **User Defined Function** | **Task** |
| \_\_init\_\_(self) | To initialise variables of a class |
| snake(self) | When the direction of the snake is changed using an arrow key, this function will rotate the read of the snake which is an image and will change the direction of the entire snake, which is basically a list of coordinates of blocks |
| score\_show(self) | This function will display the score that the user has obtained on the top left of the screen during game play in white font. The score is incremented each time the snake eats the apple |
| message\_to\_screen() | This is an extremely helpful function. Anytime something has to be printed onto the screen, this function is used. (what has to be printed, colour of the message) are passed as parameters |
| gameLoop() | Function that has the main Game loop. It contains all the code for the gameplay itself and the logic used |

|  |  |
| --- | --- |
| **Datatype** | **Name** |
| string | user\_name, x, direction,msg,tmp, tempMes, e |
| integer | block\_size,display\_width,display\_height, BlockSize, self.snakeLegnth, self.lead\_x, self.lead\_y, FPS, lead\_x\_change, lead\_y\_change |
| List | self.snakeList, g , tempL,pygame.event.get(), snakeHead,eachSegment, |
| Key events | pygame.QUIT, pygame.KEYDOWN, pygame.K\_q, pygame.K\_c, pygame.K\_LEFT, pygame.K\_RIGHT,pygame.K\_UP, pygame.K\_DOWN, |
| tuple | white,black,red,green |
| Boolean | gameExit, gameOver |
| file object | f |
| Clock\* | clock |
| Object of a Class | sc, |
| pygame.surface | logo, img, head, text, screen\_text,gameDisplay, apple |
| pygame.font.Font | font |

**Algorithm**

1. Import all the required modules

2. Define all required variables like colors, block sizes, etc

3. Accept user’s name and display game instructions

4. Load the required music for the game

5. Define a class Snake having variables – snakeList, snakeLength, lead\_x and  
lead\_y and function – snake

6. Define a class Score having variables – score, snakeLength and function –  
score\_show

7. Define a function message\_to\_screen that displays a message on the screen

8. Define a surface object gameDisplay with required height and width. Set its  
caption.

9. Display the game logo with a caption

10. Define the main function gameLoop()

11. Run gameLoop() with appropriate exception handling

Working of gameLoop():     

1. Play music
2. Define variables that control snake position changes, game exit, loops, score, snakelength, position of  
   apples, etc
3. Run a loop that is designed to keep running until the game is over (gameExit = True) – this is the main loop of the program. All subsequent steps are within this loop
4. Run a loop that runs when the game is over (gameOver = True). In this loop, display the score, update the high-score using file handling, and offer the user choice to play again or quit
5. Run a loop that monitors all user events (clicking, typing, etc). Based on the event, alter necessary variables.  
   (eg: if the user presses “left”, control the x coordinate of the snake)
6. Check if the snake head is within the defined boundary. If it is not, set gameOver = True
7. Color the screen black, blit the apple onto the screen
8. Define a list snakeHead that has two elements, x and y coordinate of each snake segment
9. Append this list to snakeList
10. To eliminate extra segments, if the length of snakeList is greater than length of snake, remove the first element  
    of the list
11. If the snakeHead value is the same as any another segment on the list, set gameOver to True.
12. Call appropriate functions to display the snake and score. Update the display.
13. Use if statements to determine if coordinates of the snakeHead are within the range of coordinates for the apple, generate new coordinates for the apple, increment snakeLength, score

**Source Code**

import pygame

import time

import random

pygame.init()

#Defining all required colours

white = (255,255,255)

black = (0,0,0)

red = (255,0,0)

green = (0,255,0)

#Defining the dimensions of the display

display\_width = 800

display\_height = 600

#defining each block's size

block\_size = 10

#Defining the font object from pygame

font = pygame.font.SysFont(None, 25)

img = pygame.image.load('snakehead.png')

#Interactive menu

user\_name = raw\_input("Enter your name->")

print "Hi", user\_name, "!"

print """This is our rendition of the classic game 'Snake'.

The objective of the game is to eat the apple using the snake.

The instructions are fairly simple.

The user is required to control the snake using the arrow keys and eat the apple by

moving the snake's head over the apple.

\*\*RULES\*\*

->If you cross the boundary of the screen, you lose.

->If you eat yourself, you lose.

->If you try to move in the opposite direction, you lose.

\*\*IMPORTANT\*\*

Using the mouse,click the window that pops up after starting the game to take control.

"""

#Checking if music exists and if it does, playing it

try:

pygame.mixer.music.load("Music.mp3")

pygame.mixer.music.play(-1)

except:

pass

x = raw\_input("Enter any key to start the game.")

user\_score = 0

#Making the 'snake' a class with necessary attributes and methods

class Snake:

BlockSize = 10

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

self.snakeList = []

self.snakeLength = 0

self.lead\_x = display\_width/2

self.lead\_y = display\_height/2

def snake(self):

#rotating the head of snake according to which direction the snake is moving

if direction == "right":

head = pygame.transform.rotate(img,270)

if direction == "left":

head = pygame.transform.rotate(img,90)

if direction == "up":

head = img

if direction == "down":

head = pygame.transform.rotate(img,180)

#Displaying the snake onto the screen

gameDisplay.blit(head, (self.snakeList[-1][0],self.snakeList[-1][1]))

for XnY in self.snakeList[:-1]:

pygame.draw.rect(gameDisplay, green, [XnY[0],XnY[1],Snake.BlockSize,Snake.BlockSize])

#Creating a score class. Each object of the class controls the score

class Score:

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

self.score = 0

self.snakeLength = 0

def score\_show(self):

text = font.render("Score: "+str(self.score), True, white)

gameDisplay.blit(text,[0,0])

#Function to display message to the screen

def message\_to\_screen(msg,colour):

screen\_text = font.render(msg, True, colour)

gameDisplay.blit(screen\_text, [display\_width/2-300, display\_height/2])

#Defining the surface object with caption

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

pygame.display.set\_caption('The Snake Game')

clock = pygame.time.Clock()

logo = pygame.image.load("logo.png")

gameDisplay.blit(logo, (275,75))

msg = """Welcome to the snake game!"""

screen\_text = font.render(msg, True, white)

gameDisplay.blit(screen\_text, [display\_width/2 - 100, display\_height/2+50])

pygame.display.update()

time.sleep(3)

def gameLoop():

#Main loop of the gam

global user\_name

global user\_score

apple = pygame.image.load("apple.png")

global direction

FPS = 25

gameExit = False

gameOver = False

#lead\_change variables control the amount by which the snake head's coordinates change

lead\_x\_change = 10

lead\_y\_change = 0

direction = "right"

SNAKE = Snake()

SNAKE.snakeLength = 1

sc = Score()

#Generating random coordinates for the apple

randAppleX = round(random.randint(0,display\_width-block\_size)/10.0)\*10

randAppleY = round(random.randint(0,display\_height-block\_size)/10.0)\*10

#This loop runs as long as the user decides to continue

while not gameExit:

#This loop runs when game over condition is satisfied

while gameOver == True:

#obtaining old high score from external file

f= open("highscore.txt",'r')

g = f.readlines()

#checking if new high score has been set

if sc.score > int(g[1]):

tmp = "CONGRATULATIONS, YOU BEAT "+str(g[0])+" WITH A NEW RECORD OF "+str(sc.score)

message\_to\_screen(tmp,red)

pygame.display.update()

#slowing down the display time

time.sleep(3)

gameDisplay.fill(black)

tmp = "Saving...."

message\_to\_screen(tmp,red)

pygame.display.update()

gameDisplay.fill(black)

time.sleep(1)

tempL = [str(user\_name)+"\n",str(sc.score)]

f.close()

#if highscore has been beaten, writing it into the external file

f= open("highscore.txt",'w')

f.writelines(tempL)

f.close()

tmp = "Saved."

message\_to\_screen(tmp,red)

pygame.display.update()

gameDisplay.fill(black)

#giving the user the option to either play again or quit the game

tmp = "PRESS C TO PLAY AGAIN OR Q TO QUIT"

message\_to\_screen(tmp,red)

else:

#Screen to be displayed if high score is not obtained

gameDisplay.fill(black)

tempMes = "GAME OVER Score " + str(sc.score) + " PRESS C TO PLAY AGAIN OR Q TO QUIT"

message\_to\_screen(tempMes, red)

pygame.display.update()

#This loops checks the user's input

for event in pygame.event.get():

if event.type == pygame.QUIT:

gameExit = True

gameOver = False

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_q:

gameExit = True

gameOver = False

if event.key == pygame.K\_c:

gameLoop()

#This loop monitors user events and inputs and makes neccessary changes to variables

for event in pygame.event.get():

if event.type == pygame.QUIT:

gameExit = True

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

lead\_x\_change = -10

lead\_y\_change = 0

direction = "left"

elif event.key == pygame.K\_RIGHT:

lead\_x\_change = 10

lead\_y\_change = 0

direction = "right"

elif event.key == pygame.K\_UP:

lead\_y\_change = -10

lead\_x\_change = 0

direction = "up"

elif event.key == pygame.K\_DOWN:

lead\_y\_change = 10

lead\_x\_change = 0

direction = "down"

#If the snake crosses the boundary of the window, gameOver is set to True

if SNAKE.lead\_x >= display\_width or SNAKE.lead\_x < 0 or SNAKE.lead\_y >= display\_height or SNAKE.lead\_y < 0:

gameOver = True

SNAKE.lead\_x += lead\_x\_change

SNAKE.lead\_y += lead\_y\_change

gameDisplay.fill(black)

#pygame.draw.rect(gameDisplay, red, [randAppleX,randAppleY,block\_size,block\_size])

gameDisplay.blit(apple, (randAppleX,randAppleY))#generates the apple

#Appending coordinates of all snake segments to its list

snakeHead = []

snakeHead.append(SNAKE.lead\_x)

snakeHead.append(SNAKE.lead\_y)

SNAKE.snakeList.append(snakeHead)

#Deleting redundant segments

if len(SNAKE.snakeList) > SNAKE.snakeLength:

del SNAKE.snakeList[0]

#Checking if the snake is eating itself

for eachSegment in SNAKE.snakeList[:-1]:

if eachSegment == snakeHead:

gameOver = True

#Displaying the snake, score and updating the screen

SNAKE.snake()

sc.score\_show()

pygame.display.update()

#Checking if the snake has crossed or eaten the apple

if SNAKE.lead\_x >= randAppleX and SNAKE.lead\_x <= randAppleX + 10 or SNAKE.lead\_x + 10 <= randAppleX and SNAKE.lead\_x + 10 >= randAppleX:

if SNAKE.lead\_y >= randAppleY and SNAKE.lead\_y <= randAppleY + 10 or SNAKE.lead\_y + 10<= randAppleY and SNAKE.lead\_y + 10 >= randAppleY:

randAppleX = round(random.randint(0,display\_width-block\_size)/10.0)\*10

randAppleY = round(random.randint(0,display\_height-block\_size)/10.0)\*10

SNAKE.snakeLength += 1

sc.score += 1

FPS +=2

clock.tick(FPS)

pygame.quit()

quit()

#gameLoop()

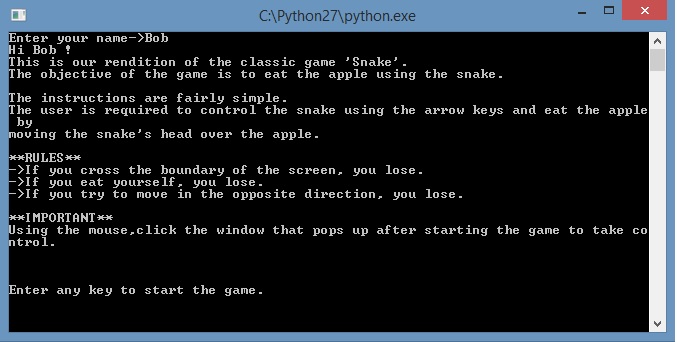
try:

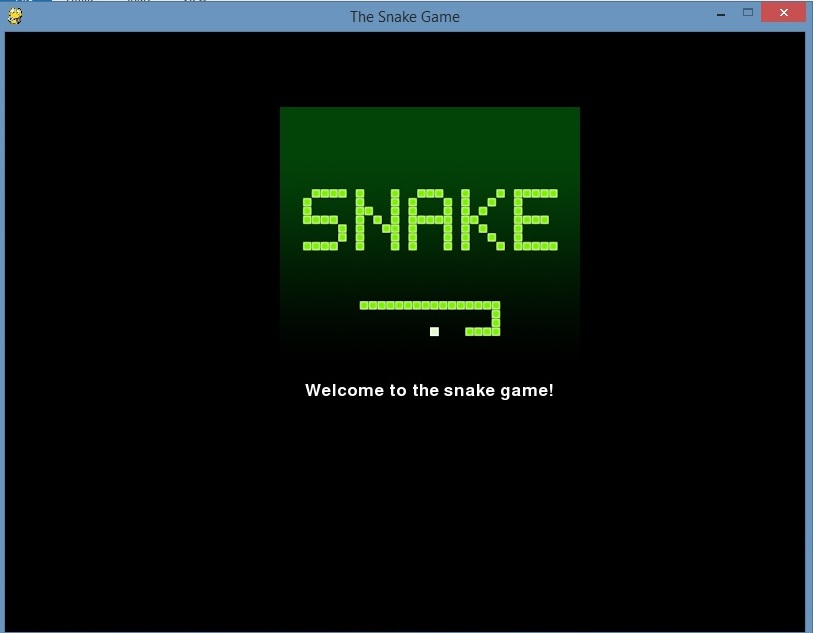
gameLoop()

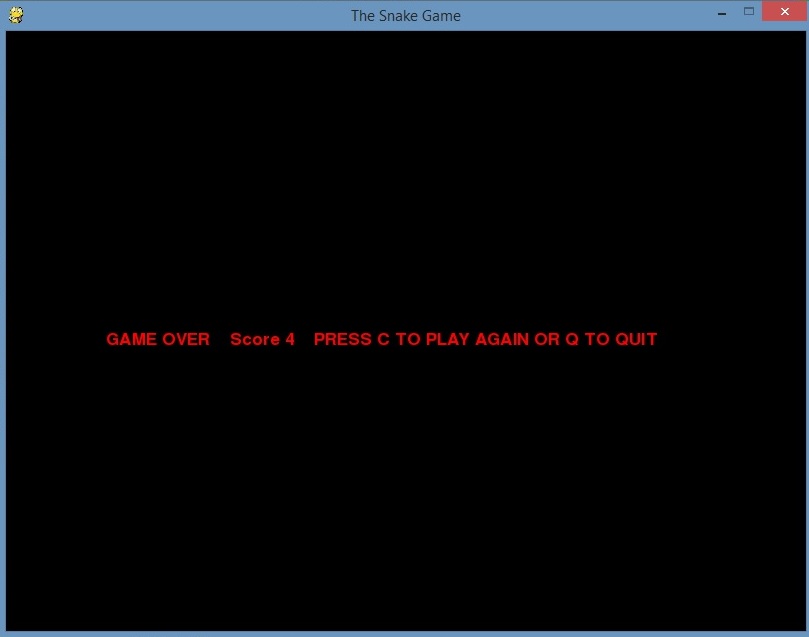
except Exception, e:

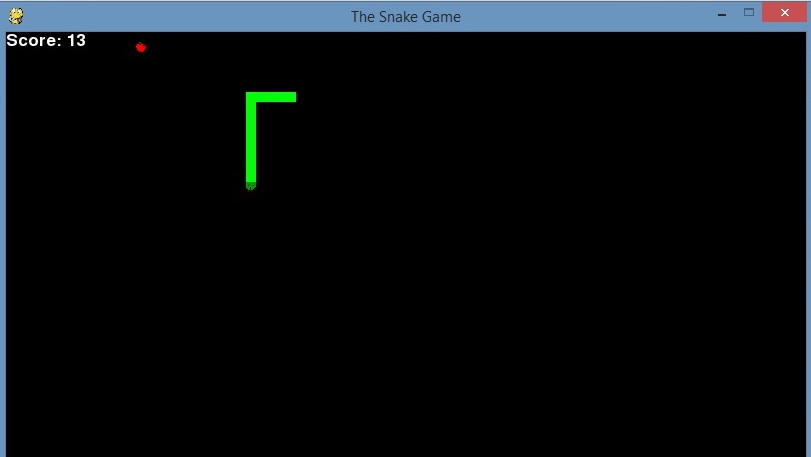
print e.message

**Screenshots**

****

****

****

****

**Bibliography:**

* <https://www.youtube.com/user/thenewboston>
* <https://www.pygame.org/docs/>