Python Mini Project Theory

1)Getting Started

2)Snake Head

3)Snake Food

4)Snake Body

5)Border Collisions

6)Body Collisions

7)Scoring

**1.)Getting Started**

**Turtle Module**

-The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways.

-The TurtleScreen class defines graphics windows as a playground for the drawing turtles.

-Turtle” is a python feature like a drawing board, which lets you command a turtle to draw all over it.

Some of the turtle functions used in the project are:

a. Turtle Methods

1)Turtle Motion(Move and Draw):

-goto() #sets position

-setx() and sety() #sets the x and y coordinates

-speed() #speed of animation

2)Turtle’s State:

-xcor() and ycor() #tells the x and y coordinates

-distance() #tells us the distance between two turtles

3)Drawing State:

-penup() #no drawing when moving

4)Color Control:

-color() #turtle’s object color

5)Appearance

-shape() #turtle’s object shape

b. Methods of Turtle Screen

1)Window control

-bgcolor() # sets the background color

2)Animation Control

-tracer() #turns off the animation on screen/turns off the screen object

-update() #updates the screen every time

3)Using screen events:

-listen() #collects the key events

-onkeypress() #assigns the keys ta a function

-mainloop() #main loop function

4)Special Methods:

-turtles() #Return the list of turtles on the screen.

5)Methods specific to Screen:

-setup() #sets the dimensions of the screen

-title()#gives the title for the screen

c.Time Module

-Python has a module named time to handle time-related tasks.

Some of the time module functions used in the project are:

->Time Functions

1)time.sleep() #suspends the current running thread for the given number of seconds

d.Random Module

-This module implements pseudo-random number generators for various distributions.

Some of the random functions used in the project are:

-randint() #Returns a **random** integer between the specified integers.

2…………………………….Snake Head…………………………………

The snake head function does the following:

#head=turtle.Turtle #creating a turtle object for the snake head

head.speed(0) #this controls the speed of the animation

head.shape(“circle”) #gives the shape

head.color(“green”) #and color

head.goto(0,0) #this function will put the head at the middle

head.direction = “stop”

-move function

We also have a move function which will move the snake head

if the head.direction is up then we change the y coordinate, we do that with all four directions up, down, right, and left.

-key bindings

We also have binded the keys on the keyboard for the respective directions by using the

onkeypress().

3……………………………….Snake Food

The snake food function does the following:

food=turtle.Turtle #creating another turtle object for the food

The code is similar to that of the snake head.

We change the shape and color and also assign its position at (0,0).

To make the snake eat the food we use an if statement:

-if the distance between two turtles(ie the food and head) is less than 20 (which the length

and breadth of each pixel in turtle)

we put the food at random spots by using the randint() imported by the random module.

4…………………Snake body…………………………………..

Adding a snake body is nothing but adding segments whenever snake gets food i.e. adding multiple objects that will the snake body to grow.

1. At the starting of the game snake consist of just head and no segments will be added unless snake gets food.

Therefore we create an empty list which consist of segments.

i.e. segments= [].

1. In the main game loop:

#addition of the segment i.e. the growing body of the snake below head.

new\_segment = turtle.Turtle() #object

new\_segment.speed(0) # defines the animation speed.

new\_segment.shape(“circle”) #defines the shape of the segment.

new\_segment.color(“yellow”) #defines the colour of the segment.

new\_segment.penup() #segments should not draw any lines with its movement

segments.append(new\_segment) #adds new segment.

1. Move end segments in reverse order: following the above statements won’t help to add segments hence, in the main loop

for index in range(len(segments)-1,0,-1):

x = segments[index-1].xcor()

y = segments[index-1].ycor()

segments[index].goto(x, y)

1. Now we have to add these created segments at the back of the snake head ,

# Move segment 0 to where the head is.

if len(segments) > 0:

x = head.xcor()

y = head.ycor()

segments[0].goto(x,y)#segments gets added to the snake head.

5…………………………Border collisions………………….

1. In this part we have to check for a border collision i.e. if the snake hits the border of the window (i.e. screen) the game should reset and the segments added at the back of the head gets deleted.

if head.xcor()>290 or head.xcor()<-290 or head.ycor()>290 or head.ycor()<-290:#all the borders

time.sleep(1) #pauses the game for 1 second.

head.goto(0,0) the head of the snake goes to initial co-ordinates.

head.direction = "stop" #stops the head.

1. Now we have to hide the segments after the snake collides with the borders of the screen.

for segment in segments:

segment.goto(1000, 1000) #segments behind the head goes off the screen as python turtle module does not supports deleting the segment from the screen.

1. Clearing the segments list in order to get the head back at the initial position.

segments.clear()

6…………………………..Body collisions…………………

1. In this part we have to check if the snake collides with its own body then the game should reset i.e. the head goes to its initial co-ordinates and the segments behind the head should disappear.

# Check for head collision with the body segments

for segment in segments:

if segment.distance(head) < 20: #collision with the segment.

time.sleep(1) #time delay(pause)

head.goto(0,0) #head to the initial co-ordinates.

head.direction = "stop" #no movement of the head.

# Hide the segments

for segment in segments:

segment.goto(1000, 1000) #segments go off the screen

# Clear the segments list

segments.clear() #clearing the segments list from the code for head appearance.

7……Scoring…………

In this we will add the scoreboard in the main screen of the game. Whenever snake eats the food, the segments goes on increasing and simultaneously the score increases accordingly. After the snake collides with the border or with its own body the score should reset to the initial value and high score by the previous game played.

1. Defining the main scoreboard on the screen:

score=0

high\_score=0

pen = turtle.Turtle()

pen.speed(0) #animation speed.

pen.shape("square") #shape of the scoreboard.

pen.color("white") #colour of the scoreboard

pen.penup()

pen.hideturtle()

pen.goto(0, 260) #position of the scoreboard.

pen.write("Score: 0 High Score: 0", align="center", font=("Courier", 24, "normal")) #initializing the values.

1. In the main loop.

# Increase the score

score += 10 #increment of the score

if score > high\_score:

high\_score = score #saving high score of the previous game.

pen.clear()

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal")) #formatting

1. When body hits the border:

# Reset the score

score = 0

# Reset the delay

delay = 0.1 #resetting the speed of the snake

pen.clear() #clears the main initializing of the default score

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal"))

1. When the snake hits the segments(i.e. its own body): these statements are to be written at the loops of the border collisions and body collisions with proper indentation

# Reset the score

score = 0

# Reset the delay

delay = 0.1 #resetting the speed of the snake.

# Update the score display

pen.clear() #clears the main initializing of the default score

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal"))