Charles Sturt University

**Assessment item 2**

**Forest Drawing**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Student ID: 11706271

Student Name: Alina Maksumova

Lecturer Name: Dr Thair Al-Dala'in

Subject Name: Programming Principles

Subject Code: ITC558

Sydney 2020

**Contents**

[**Task 1** 3](#_Toc39784270)

[**Flow chart of handle\_click() function** 3](#_Toc39784271)

[**Task 2** 4](#_Toc39784272)

[**Testing Data** 4](#_Toc39784273)

[**Task 3** 8](#_Toc39784274)

[**Python files:** 8](#_Toc39784275)

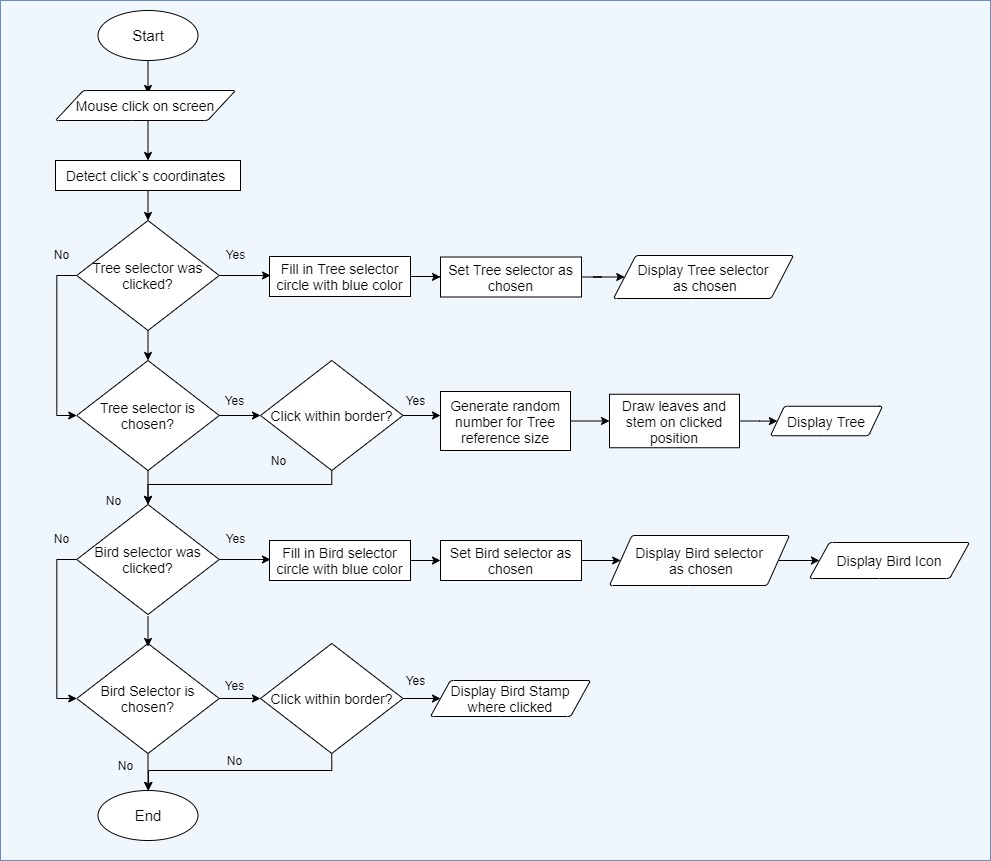
[**Source code – Utilities.py:** 9](#_Toc39784276)

[**Source code – Forest.py:** 14](#_Toc39784277)

[**Output:** 15](#_Toc39784278)

# **Task 1**

## **Flow chart of handle\_click() function**

****

# **Task 2**

## **Testing Data**

| **Test Data Table** | | | | |
| --- | --- | --- | --- | --- |
| **Test data type** | **Test data** | **The reason it was selected** | **The output expected due to the use of the test data** | **The screenshot of actual output when the test data is used** |
| Normal | Mouse clicks on Tree and Bird selectors | To check whether selectors and interactive | When Tree selector is clicked – it appears as chosen one, same with the Bird selector |  |
| Normal | Mouse clicks within drawing borders and during Tree mode | To check whether drawings of different size Trees are able during Tree mode | Trees of different sizes (from 0.7x to 1.3x of a reference size) |  |
| Normal | Mouse clicks within drawing borders and during Bird mode | To check whether drawings of birds are able during Bird mode | Birds appear in clicked positions |  |
| Normal | Key “Right” and “Left” presses during Bird mode; mouse clicks within drawing borders | To check whether bird icon and birds stamps change their angles in accordance with key presses | Bird icon shows current bird angle, birds stamps appear with different set angles |  |
| Abnormal | Mouse clicks outside the drawing borders | To check whether drawings will be implemented if clicks are done outside drawing boundary | Clicks have no effect, for exception of choosing mode or clicking on bird icon. |  |
| Abnormal | Left key press during Tree mode | To check whether bird icon will change its tilt during Tree mode | Bird icon tilt can be changed during Tree mode. It will appear once to show new tilt and will disappear after next click. It will not influence on drawings. |  |
| Abnormal | Press key “Up” and “Down” | To check whether bird`s icon changes somehow when up or down bottoms are clicked | Up or down bottoms do not influence bird`s icon. |  |

# **Task 3**

## **Python files:**





## **Source code – Utilities.py:**

Utilities.py

import turtle #import turtle to implement graphics

import math #to find height of triangle

import random #to generate random size of trees

#set window size and color

w\_width = 800

w\_height = 800

bg\_color = 'aliceblue' #background color simulating sky

turtle.setup(w\_width, w\_height)

turtle.bgcolor(bg\_color)

turtle.speed(0)

#set global variables

x = -400 #global x coordinates, invisible before click within the border

y = -400 #global y coordinates, invisible before click within the border

click = 0 #declaring mode(tree/bird) global

tilt\_angle = 0 #tilt angle value

#set border for interactive window

def set\_border(point\_x ,point\_y, window\_size, pen\_size, angle):

turtle.penup()

turtle.goto(point\_x,point\_y)

turtle.pendown()

turtle.pensize(pen\_size)

#draw a square using for loop

for i in range (4):

turtle.forward(window\_size)

turtle.left(angle)

turtle.hideturtle()#make turtle invisible

#write text for selectors

def write\_word (word, start\_x,start\_y):

turtle.penup()

turtle.goto(start\_x, start\_y)

turtle.pendown()

turtle.write(word, align='center', font=('Times New Roman', 12, 'bold'))

#create function to draw selectors

def draw\_circle(centre\_x, centre\_y, radius, pen\_color, fill\_color):

#set pen up if turtle is down

turtle.pensize(1)

if turtle.isdown():

turtle.penup()

#go to identified location

turtle.pencolor(pen\_color)

turtle.fillcolor(fill\_color)

turtle.begin\_fill()

turtle.goto(centre\_x, centre\_y)

turtle.pendown()

turtle.circle(radius) #draw circle

turtle.end\_fill()

#create function to handle mouse click

def handle\_click(x, y):

global click #call global click

print('detected a click at', x, y)

turtle.hideturtle() #hide turtle

turtle.penup()#hide turtle`s pen

turtle.goto(x,y) #move turtle to click`s coordinates

turtle.speed(0) #set animation on fastest speed

if (-10 <= x <= 10 and 370 <= y <= 390): #check whether Tree selector was clicked

draw\_circle(0, 370, 10, 'black', 'royalblue') #mark Tree selector as chosen

draw\_circle(100, 370, 10, 'black', 'aliceblue') #mark Bird selector as disable

click=1 #set Tree mode

return click ##return mode

#enable Tree drawings for Tree mode and default mode

if click==0 or click==1:

if (-350<x<350 and -350<y<350):#check if click`s within drawing border

turtle.penup() #hide turtle`s pen

turtle.goto(x,y) #move turtle to click`s coordinates

turtle.pendown() #set turtle pen down

r = random.uniform(0.7,1.3) #generate random number within from 0.7 to 1.3

draw\_triangle(r\*100,'green', 'green') #draw tree leaves

draw\_rectangle(r\*15, r\*80,'sienna','sienna') #draw tree stem

if(90 <= x <=110 and 370 <= y <= 390): #check whether Bird selector was clicked

draw\_circle(100, 370, 10, 'black', 'royalblue') #mark Bird selector as chosen

draw\_circle(0, 370, 10, 'black', 'aliceblue')#mark Tree selector as disable

bird\_icon() #display current bird icon

click=2 #set Bird mode

return click #return mode

#enable Bird drawings for Bird mode

if click==2:

if (-350<x<350 and -350<y<350):#check if click`s within drawing border

draw\_bird(x, y, 'indigo', 'indigo')#make Bird`s stamp

bird\_icon() #display current tilt of turtle(Bird), no stamp

#draw triangles for tree leaves

def draw\_triangle(width, pen\_color, fill\_color):

#set pen up if turtle is down

turtle.hideturtle()

height =((width\*math.sqrt(3))/2)#math formula to find height of regular triangle

#set colors

turtle.pencolor(pen\_color)

turtle.fillcolor(fill\_color)

turtle.begin\_fill()

for i in range (3): #start drawing 3 triangles in a row

#draw regular triangle

turtle.forward(width/2) #start from base centre

turtle.left(120) #make angle of 60 degdrees

turtle.forward(width)

turtle.left(120)

turtle.forward(width)

turtle.left(120)

turtle.forward(width/2)#finish with base centre

for j in range (1): #exclude 3rd triangle

turtle.left(90)

turtle.penup()

turtle.forward(height\*0.4)#this is the point of a new triangle base (40% overlapping)

turtle.right(90)

turtle.end\_fill()

turtle.penup()#hide turtle`s pen

turtle.right(90)

turtle.forward((height\*0.4)\*3)#returns turtle to first triangle base`s centre

#create funtion to draw tree stem

def draw\_rectangle(width, height, pen\_color, fill\_color):

#set colors

turtle.pencolor(pen\_color)

turtle.fillcolor(fill\_color)

turtle.begin\_fill()

#draw rectangle

turtle.forward(height)#goes to rectangle`s base centre

turtle.left(90)

turtle.pendown()

turtle.forward(width/2)

turtle.left(90)

turtle.forward(height)

turtle.left(90)

turtle.forward(width)

turtle.left(90)

turtle.forward(height)

turtle.left(90)

turtle.forward(width/2)#finishes with rectangle`s base centre

turtle.end\_fill()

#create function to handle left key press

def left\_keypress():

global tilt\_angle #calls global tilt

tilt\_angle += 20 #increases tilt angle on 20 degress

turtle.tiltangle(tilt\_angle) #turns tilt left on 20 degress

bird\_icon() #displays new position of bird

return tilt\_angle # returns new tilt

#create function to handle right key press

def right\_keypress():

global tilt\_angle #calls global tilt

tilt\_angle -= 20 #decreases tilt angle on 20 degress

turtle.tiltangle(tilt\_angle) #turns tilt right on 20 degress

bird\_icon()#displays new position of bird

return tilt\_angle # returns new tilt

#set turtles icon as a bird to leave stamps

def draw\_bird (bird\_x, bird\_y, fill\_color, pen\_color):

turtle.penup()

turtle.pencolor(pen\_color)

turtle.fillcolor(fill\_color)

turtle.begin\_fill()

turtle.goto(bird\_x, bird\_y)

turtle.pendown()

turtle.register\_shape('bird', ((-22,-39),(-20,-7),(-7,3),(-11,7),(-12,9),(-11,10),(-9,10),(-3,7),

(10,24),(30,16),(13,18),(4,0),(14,-6),(6,-13),(0,-4),(-14,-13),(-22,-39)))

turtle.shape('bird')

turtle.showturtle()

turtle.end\_fill()

turtle.stamp() #saves bird on the screen

#set turtles icon as a bird

def bird\_icon():

pen\_color='indigo'

fill\_color = 'indigo'

turtle.penup()

turtle.pencolor(pen\_color)

turtle.fillcolor(fill\_color)

turtle.begin\_fill()

turtle.goto(-340,370)

turtle.pendown()

turtle.register\_shape('bird', ((-22,-39),(-20,-7),(-7,3),(-11,7),(-12,9),(-11,10),(-9,10),(-3,7),

(10,24),(30,16),(13,18),(4,0),(14,-6),(6,-13),(0,-4),(-14,-13),(-22,-39)))

turtle.shape('bird')

turtle.showturtle()

turtle.end\_fill()

## **Source code – Forest.py:**

Forest.py

import turtle #import turtle module

import utilities #import utilities module to implement its functions

turtle.title("Wonderful Forest") #name the window

def main ():

#set border

utilities.set\_border(-350,-350,700,3,90)

#draw circles for tree and bird selectors

utilities.draw\_circle(0,370,10,'black', 'royalblue')

utilities.draw\_circle(100,370,10,'black','aliceblue')

#write texts for tree and bird selectors

utilities.write\_word('Tree', 40,370)

utilities.write\_word('Bird', 140,370)

#track mouse click

handle\_click = utilities.handle\_click

turtle.listen()

turtle.onscreenclick(handle\_click, 1)#assign function name as parametr

##track key presses:

#track left key press

left\_keypress = utilities.left\_keypress

turtle.listen()

turtle.onkey(left\_keypress, 'Left')

#track right key press

right\_keypress = utilities.right\_keypress

turtle.listen()

turtle.onkey(right\_keypress, 'Right')

main()

## **Output:**

