Project Documentation For Catch the Red, Avoid the Black!

Author: Ka Tin June Man
Date: Dec 7, 2021

Table of Contents

List of Tables

List of Figures

1. Project Overview

Catch the Red, Avoid the Black is a 2D pong game built by the python tkinter. This game is used to train the user's body reaction. This game is played by a single user, users can control the paddle in this game, the mission of the user is catching all the red circles by controlling the paddle and to avoid being hit by the black circle.

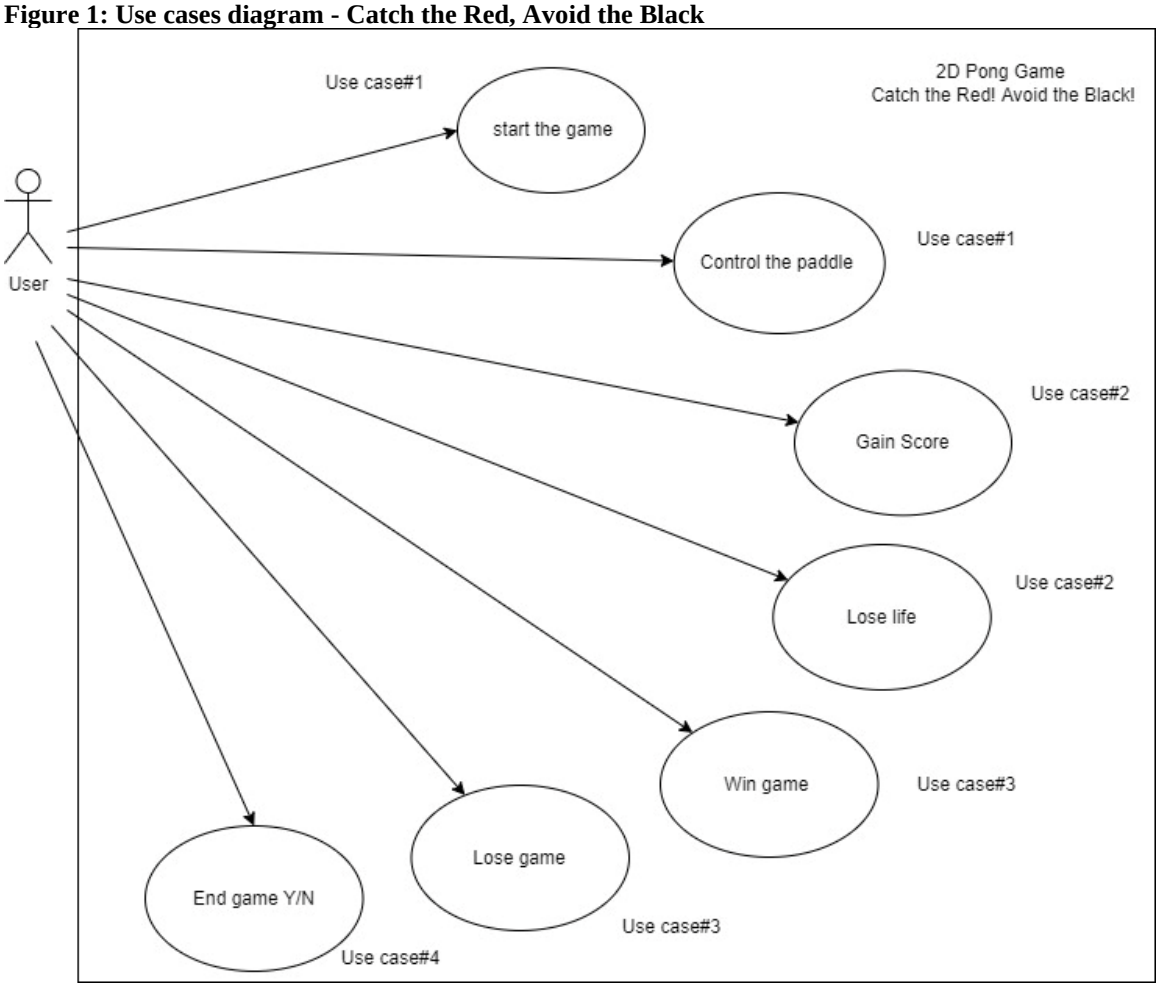
2. Project Requirements

This 2D pong game Catch the Red, Avoid the Black is a single user game where the user can:

1. *Start the game, user will see the two different colors of circle, they are all bouncing in the tkinter canvas at the same time.*
2. *User able to control the paddle (at the bottom of the canvas) to move left and right to catch the red circles*
3. *User also able to control the paddle by keyboard to avoid being hit by the black circle*
4. *If the user's paddle being hit by the black circle, the user will lose one life*
5. *If the user's paddle catches the red circle successfully, it will add one point of score*
6. *If the user's paddle catches all the red circle without being hit by the black one, the user will win this game and the canvas will close*
7. *If the user's paddle being hit by the black circle over the 3 times limited, the user will lose the game and the canvas will shut down immediately*
8. *If the user can not catch all the red circle within the time limited, user will lose the game also*
9. *User controls the paddle and fulfills the condition to win the game or lose the game*

3.1 Use Case Diagram

Figure 1: Use cases diagram - Catch the Red, Avoid the Black



3.1.1 Use Case #1

Use Case 1	*User start the game, use the keyboard to play*
Pre-conditions	*User can run and start to play the game*
Steps	1. *Open the game file* 2. *Run the code in PyCharm* 3. *Every element in the game can be run and act successfully* 4. *Use the keyboard to control the paddle: AD or left right*
Post-conditions	*Run the game successfully and start to control the paddle with keyboard*

3.1.2 Use Case #2

Use Case 2	*Gain Score / Lose Life*
Pre-conditions	*User can control the paddle to gain score or lose life*
Steps	1. *User Control the paddle* 2. *To move the paddle left and right, and try to catch the bouncing red circle and avoid hitting by the black circle* 3. *When a user catches one red circle successfully, they will get one point. If user being hit by a black circle, that will decrease one lifetime* 4. *System will display the score and the lifetime on the canvas*
Post-conditions	*User catch the red circle to get one point or user being hit by the black circle and decreases one lifetime*

3.1.3 Use Case #3

Use Case 3	*Win / Lose*
Pre-conditions	*User and win or lose game*
Steps	1. *Users catch all the circle* 2. *Red circle will decrease on the canvas when being caught* 3. *When a user catches all the red circle, this action will let the user win the game,* 4. *Tkinter canvas will quit when game ends* 5. *When user being hit by the black circle over the three lifetimes limited, this action will make the user lose the game* 6. *Tkinter canvas will quit, and the game will end*
Post-conditions	*User ends the game by win or lose*

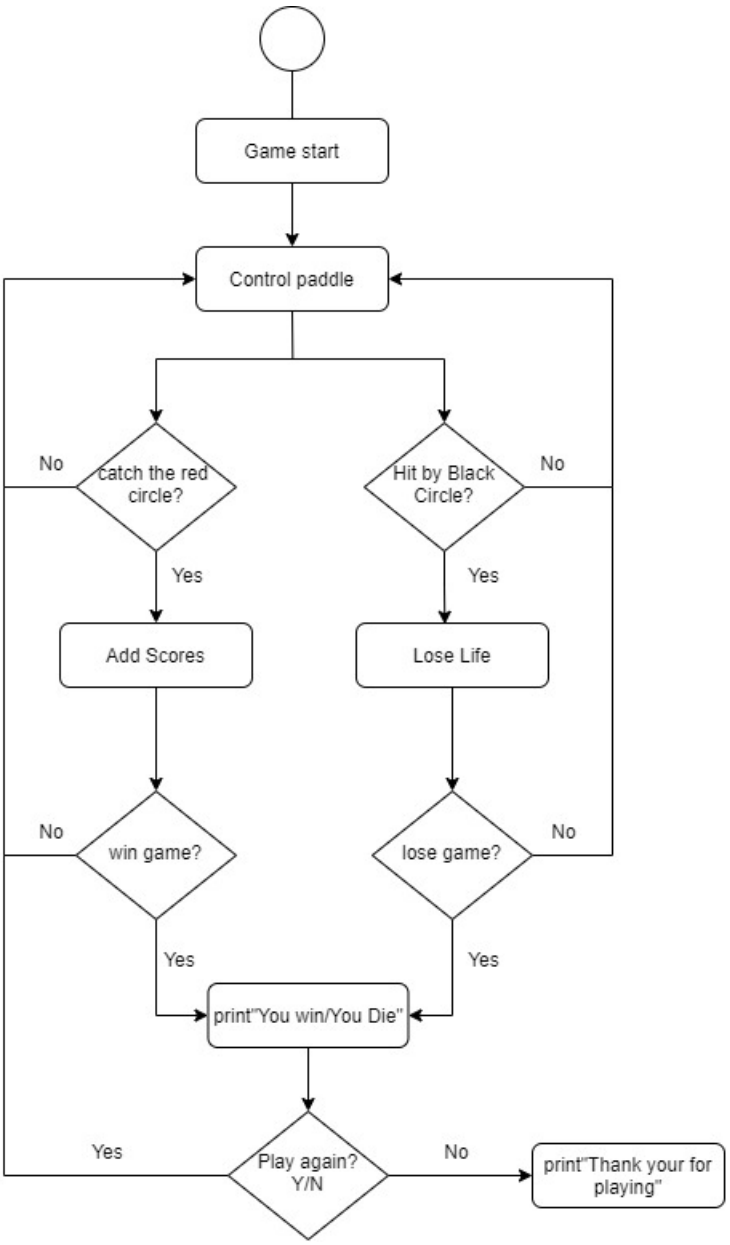
3.1.4 Use Case #4

Use Case 4	*Users continue to play the game or end game*
Pre-conditions	*Users can run again the python game.py to play the game or choose not to run the game*
Steps	1. *User can play again after win or lose* 2. *If the user does not want to play again, the game ends and the game window quits.*
Post-conditions	*User can choice to play again after both win and lose, or choice the end game*

3.2 Activity Diagrams

When game started, user need to use keyboard to control the paddle to get score or lose life, user can win or lose game by fulfil different condition:
a) Win(condition): Catch all the red circle in the canvas without being hit by the black circle over 3 times
b) Lose(condition): Being hit by the black circle over three times
After You Win/You Die, the syster will ask user play again or not Y/N
Y=Restart the game / N=Quit canvas

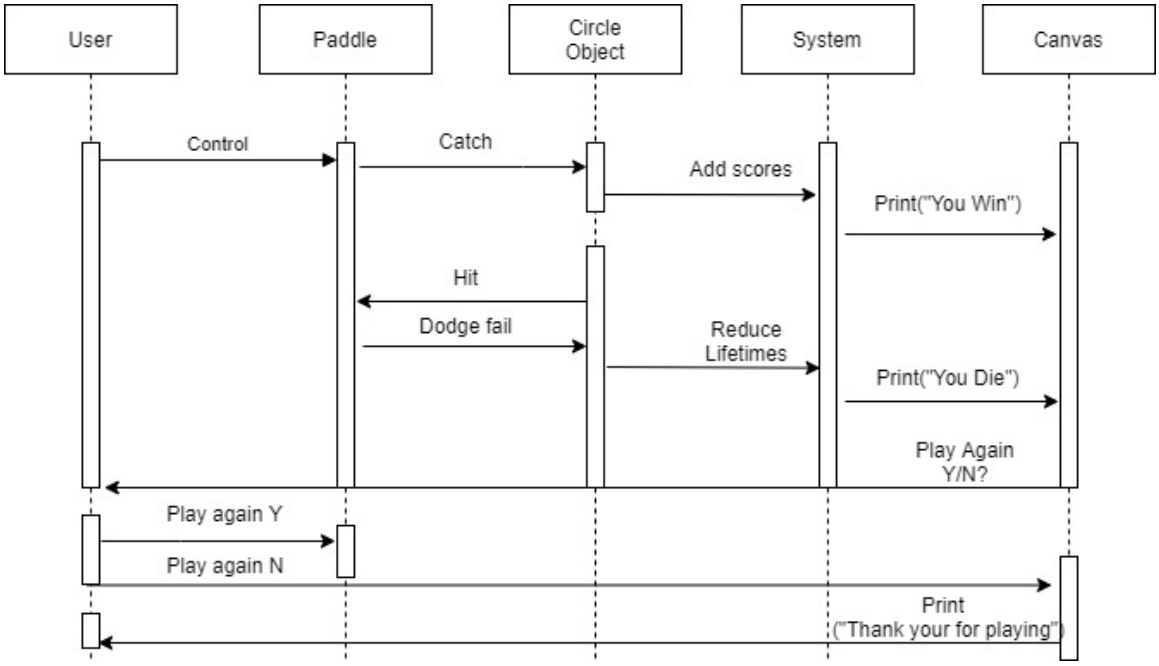
Figure 2: Activity diagram - Catch the Red, Avoid the Black



4 Design Plans

4.1 Sequence Diagrams

Figure 3: Sequence diagram - Catch the Red, Avoid the Black



1. User started to control the paddle
2. The paddle will catch the red circle or being hit by the black circle
3. When catch the red circle, system will add one score,
4. The amount of the red circle will decrease after user caught them
5. System will determine the user win the game until all the red circles disappear on the screen, then print" You win" on the canvas.
6. The black circle will not decrease even user paddle being hit by them
7. When the user's paddle being hit by black circle, the lifetime of the user will decrease one
8. System will determine the user lose the game after the lifetime turn to zero, then print" You Die" on the canvas
9. After the Win/Lose result, canvas will print" Play again Y/N"
   If user input Y, game will be restarted
   If user input N, canvas will quit automatically

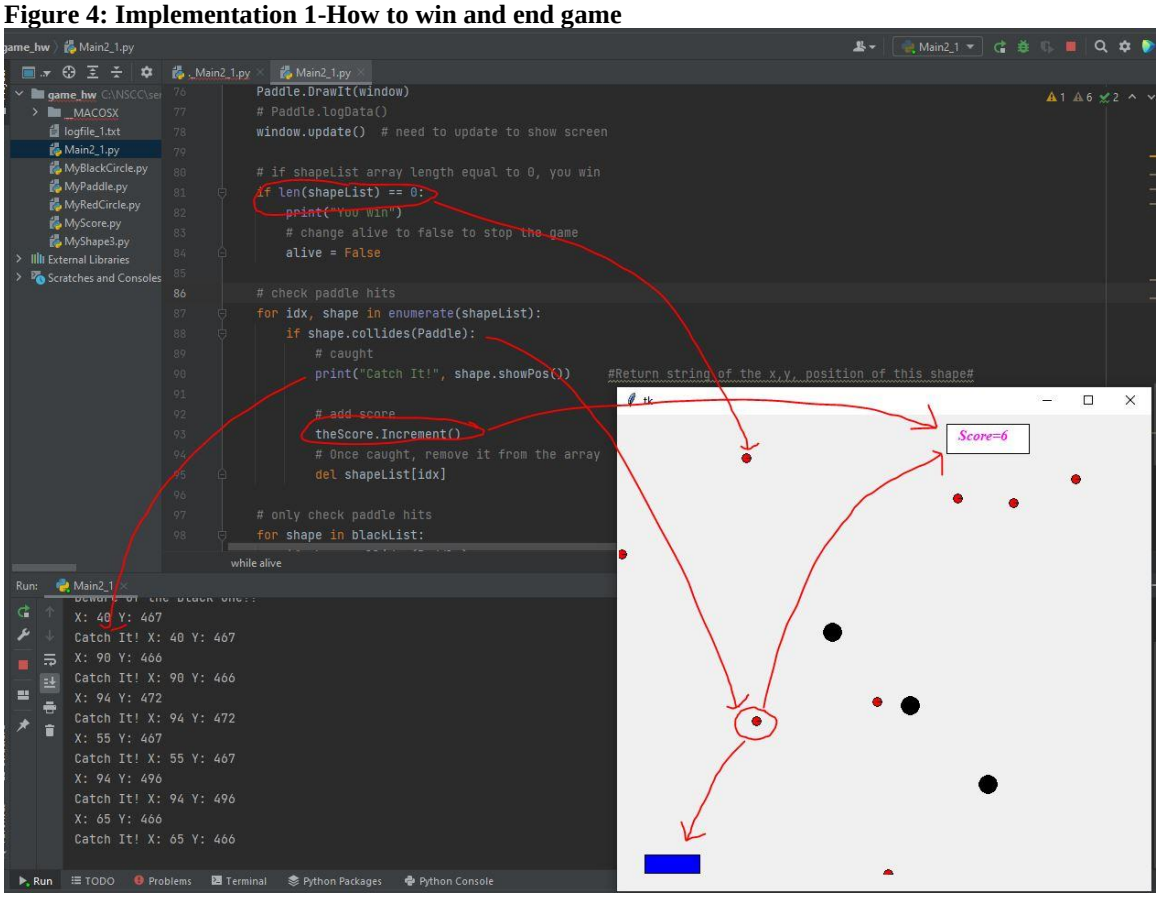
5 Implementation

5.1 Screen Snippet

Important implementations: End game condition – You Win or You died

How to win and end the game
Conditions: Make the red circle become zero on the canvas
In (Fig.4), paddle collides with the red circle, then print ”Catch It” on the console window. After collides, score will increase one and the red circle in the shape list will remove after collides with the paddles. Until all the red circles because zero on the canvas. System will determinate the user wins the game, then the canvas will close and game end.

Figure 4: Implementation 1-How to win and end game



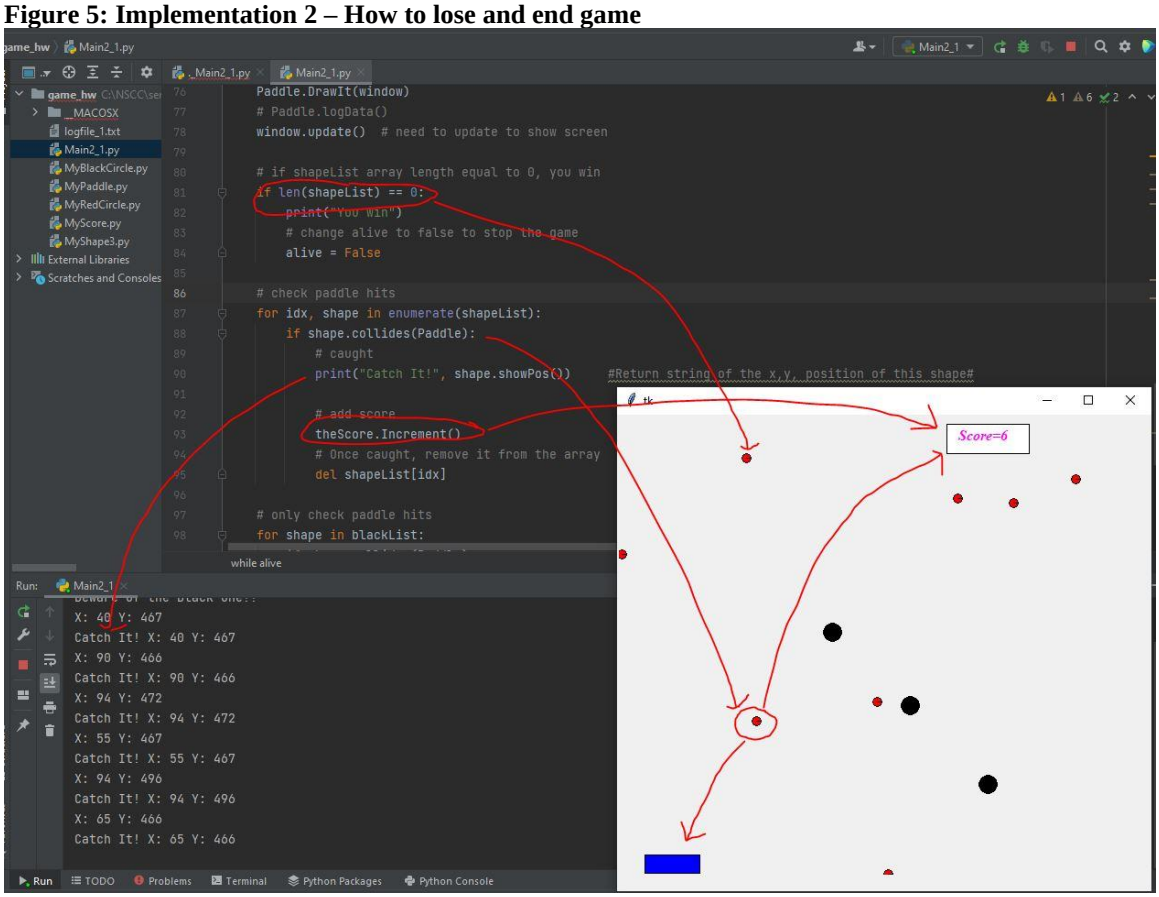

```
Catch It! X: 449 Y: 487
You win

Process finished with exit code 0
```

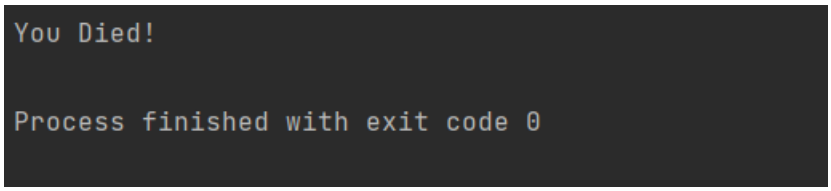
How to lose and end the game

Conditions: Paddle collides with the black circle

In (Fig.5), paddle collides with the black circle, then print" You Died" on the console window. After collides, system will determinate the user lose the game, then the canvas will shut down and game end

Figure 5: Implementation 2 – How to lose and end game





5.2.1 Code Snippet #1

Table 1 – Python script in game, determinate the win game conditions

This code is one of the importation parts in this game, it was used to determinate how to end this game.

Table 1: Code #1 – Win game determination

```
# if shapeList array length equal to 0, you win
    if len(shapeList) == 0:
        print("You win")
        # change alive to false to stop the game
        alive = False

    # check paddle hits
    for idx, shape in enumerate(shapeList):
        if shape.collides(Paddle):
            # caught
            print("Catch It!", shape.showPos())
          #Return string of the x,y, position of this shape#

            # add score
            theScore.Increment()
            # Once caught, remove it from the array
            del shapeList[idx]
```

5.2.2 Code Snippet #2

Table 2 - Python script in game, determinate the lose game conditions

This code is other importation point in this game, it was used to determinate another way to end this game.

Table 2: Code #2 – Lose game determination

```
# only check paddle hits
    for shape in blackList:
        if shape.collides(Paddle):
            print("You Died!")
            # change alive to false to stop the game
            alive = False
```

6 Testing Paragraphs

6.1 Unit Testing

Expected result of Unit Test - Circle collides with Paddle
In the circle, the collides distance set < 20, if the x, y distance < 20, two object should be collided

Class MyShape
def __init__(self, x=100, y=100, dx=0, dy=0)

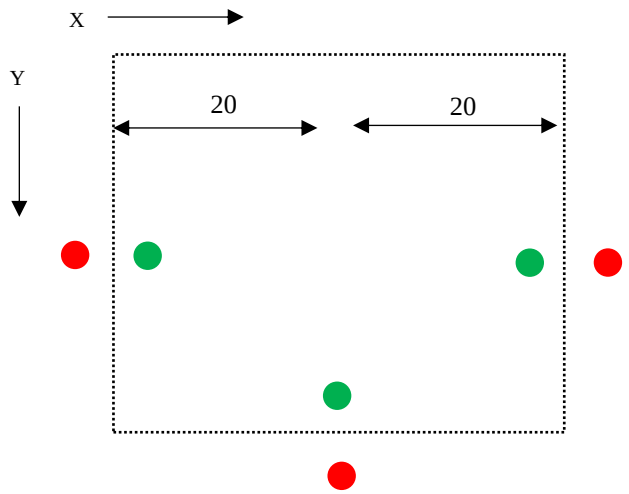


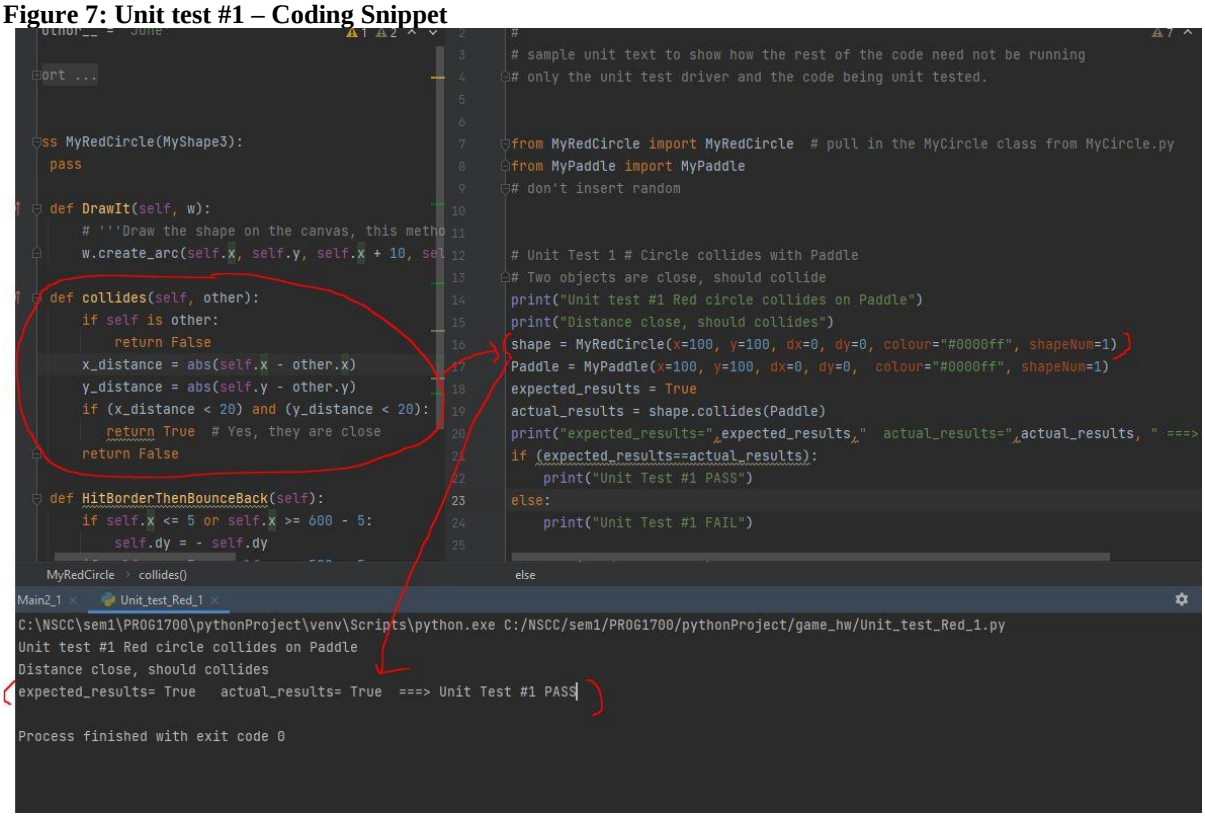
Figure 6: Unit test – Collide condition

Green = Hit (100, 100)
Hit (110,100)
Hit (101,100)
Hit (119,100)
Hit (90, 100)
Hit (89, 100)
Hit (99, 100)
Hit (81, 100)

Red = Miss (120, 100)
Miss (121, 100)
Miss (80, 100)
Miss (79, 100)

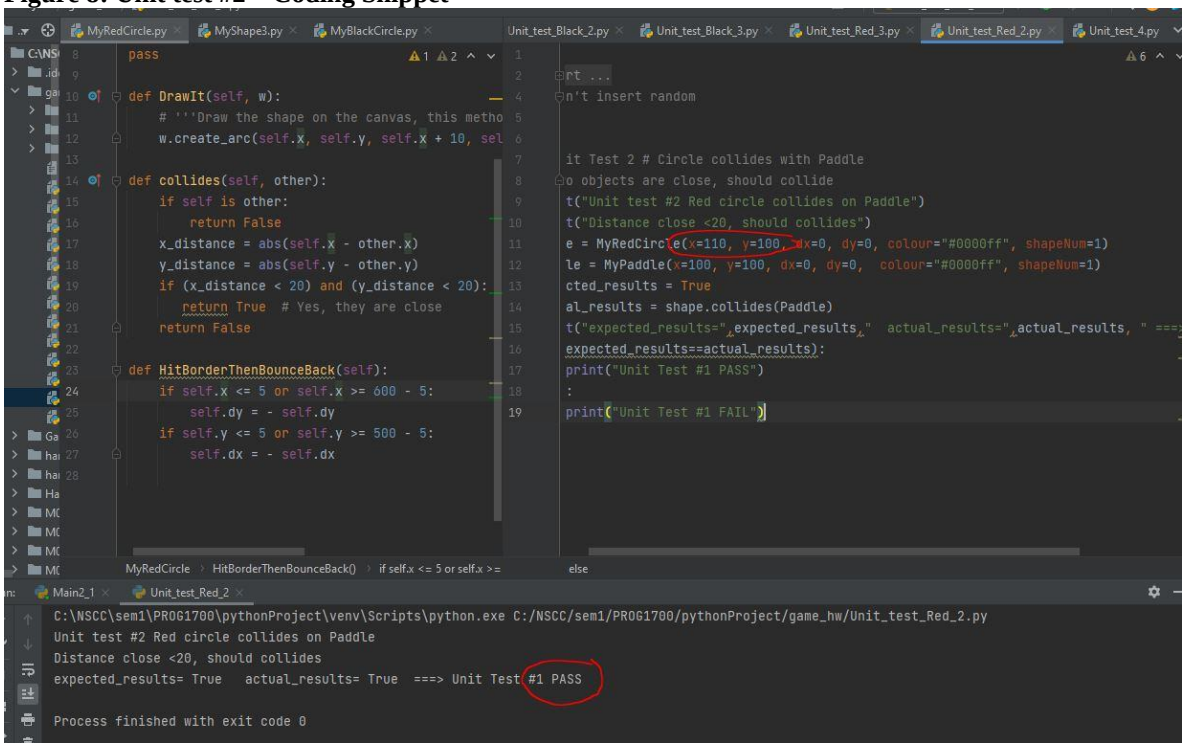
***Unit Test#1 –Circle collides with Paddle – True* (100, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 7: Unit test #1 – Coding Snippet



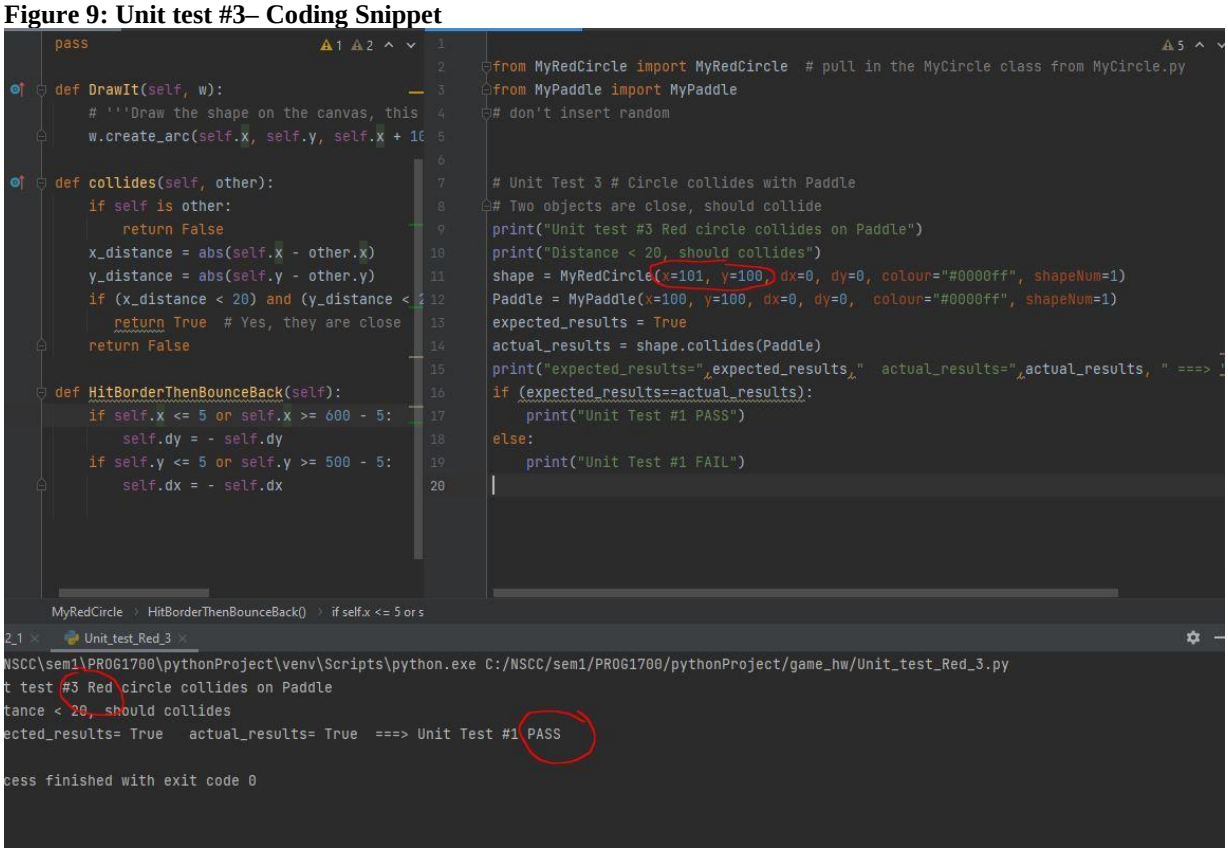
Unit Test#2 –Circle collides with Paddle – True **(110, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 8: Unit test #2 – Coding Snippet



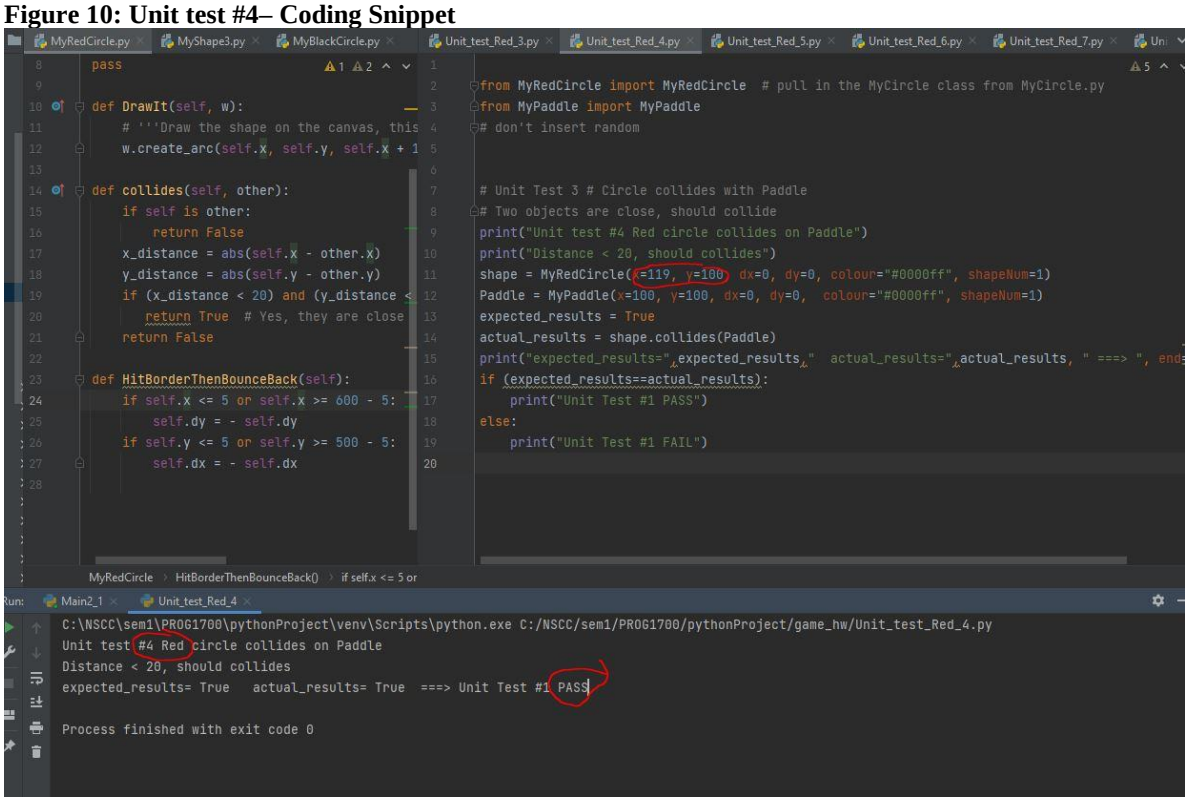
***Unit Test#3 –Circle collides with Paddle – True* (101, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 9: Unit test #3– Coding Snippet



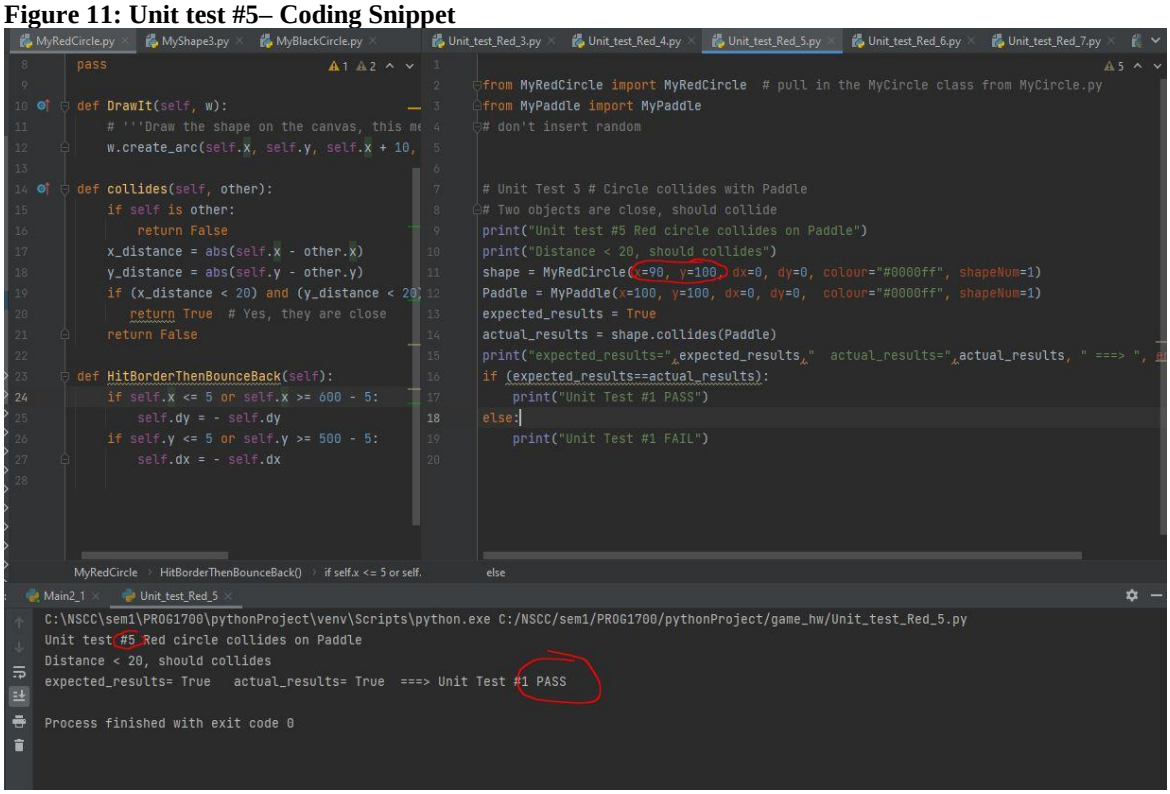
***Unit Test#4 –Circle collides with Paddle – True* (119, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 10: Unit test #4– Coding Snippet



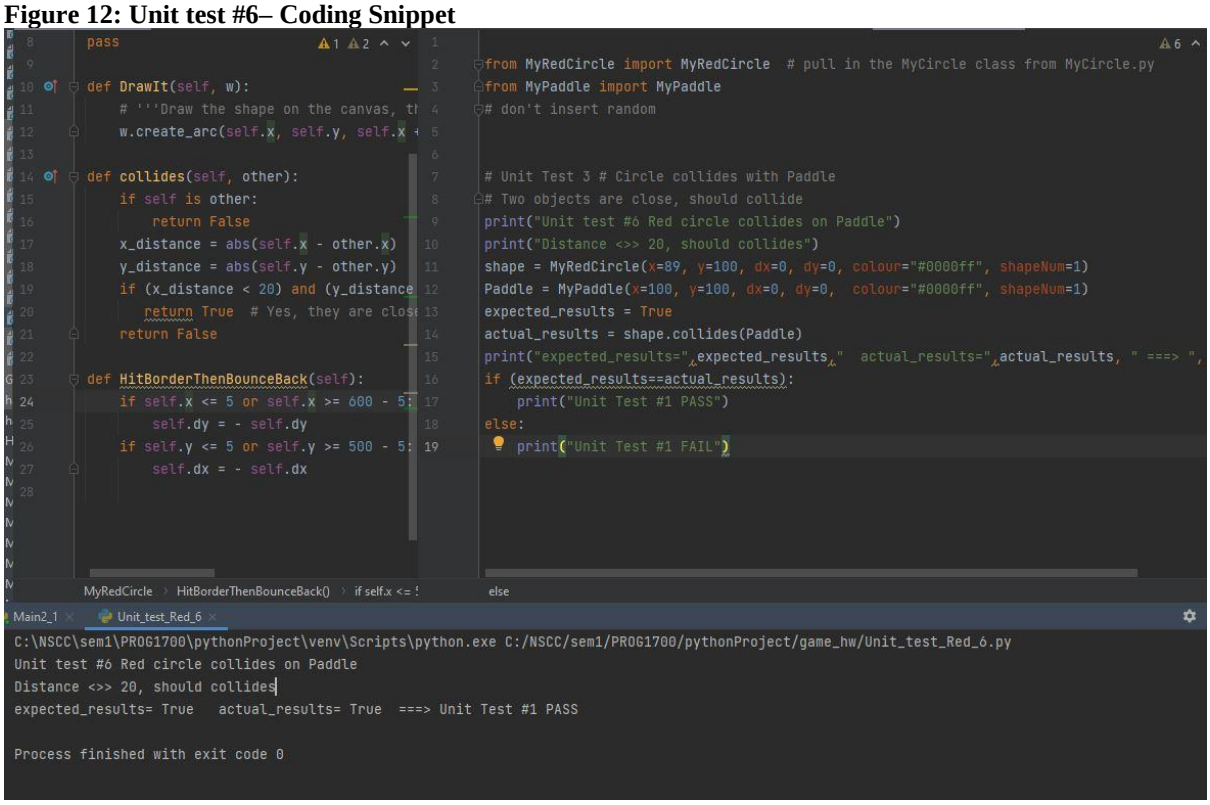
***Unit Test#5 –Circle collides with Paddle – True* (90, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 11: Unit test #5– Coding Snippet



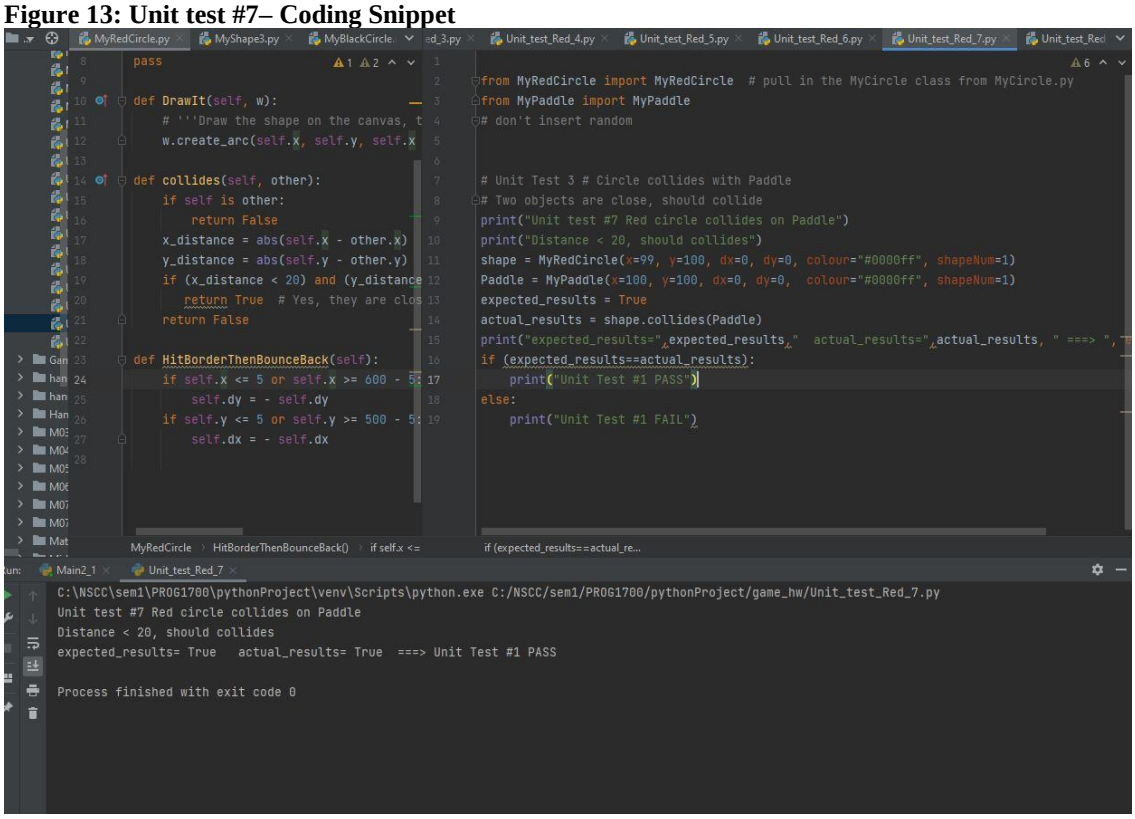
***Unit Test#6 –Circle collides with Paddle – True* (89, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 12: Unit test #6– Coding Snippet



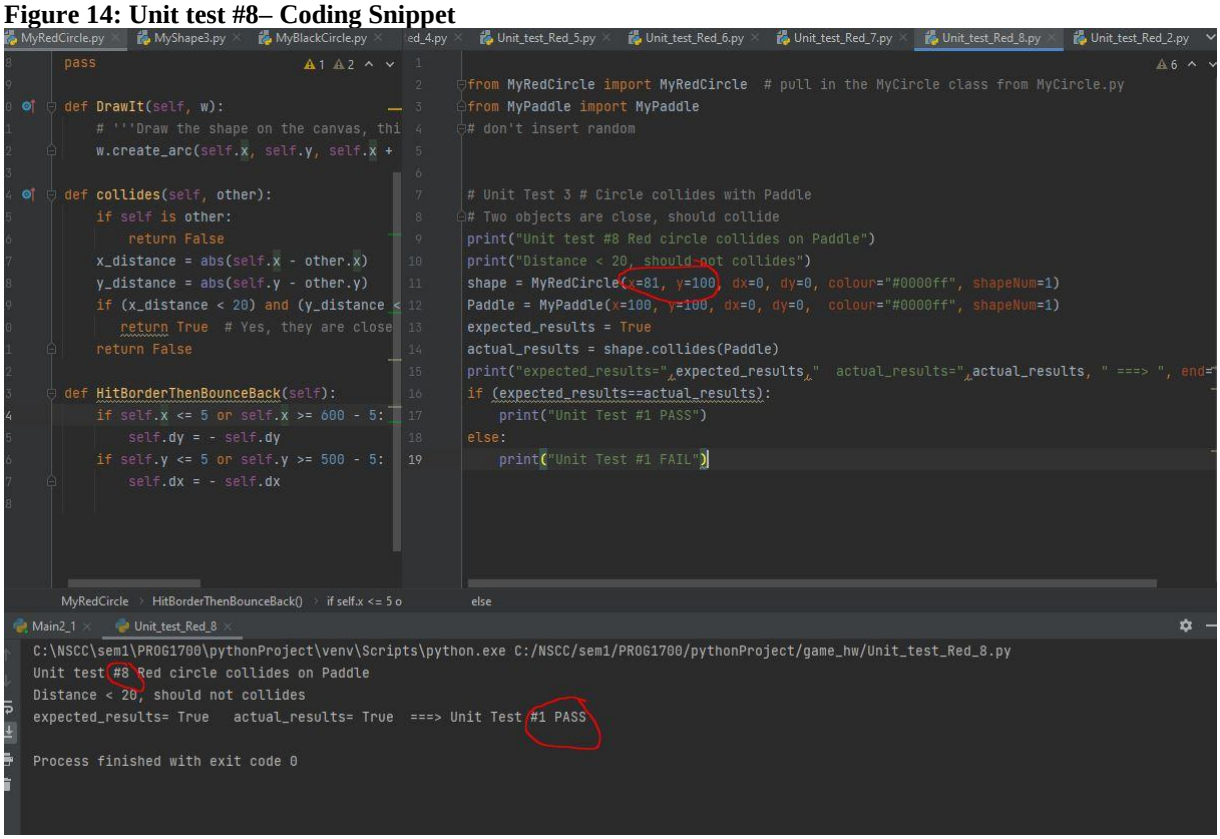
***Unit Test#7 –Circle collides with Paddle – True* (99, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 13: Unit test #7– Coding Snippet



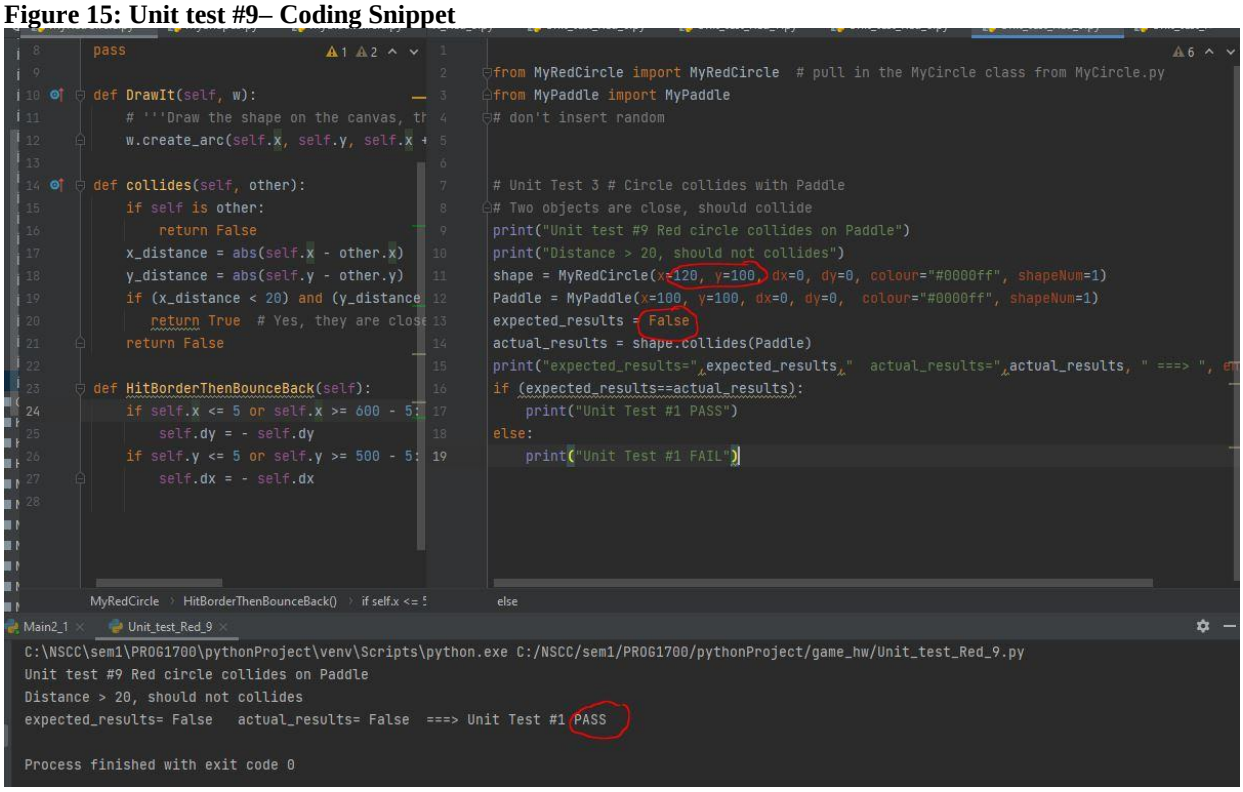
***Unit Test#8 –Circle collides with Paddle – True* (81, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance < 20, should be collided, unit test#1.1=PASS

Figure 14: Unit test #8– Coding Snippet



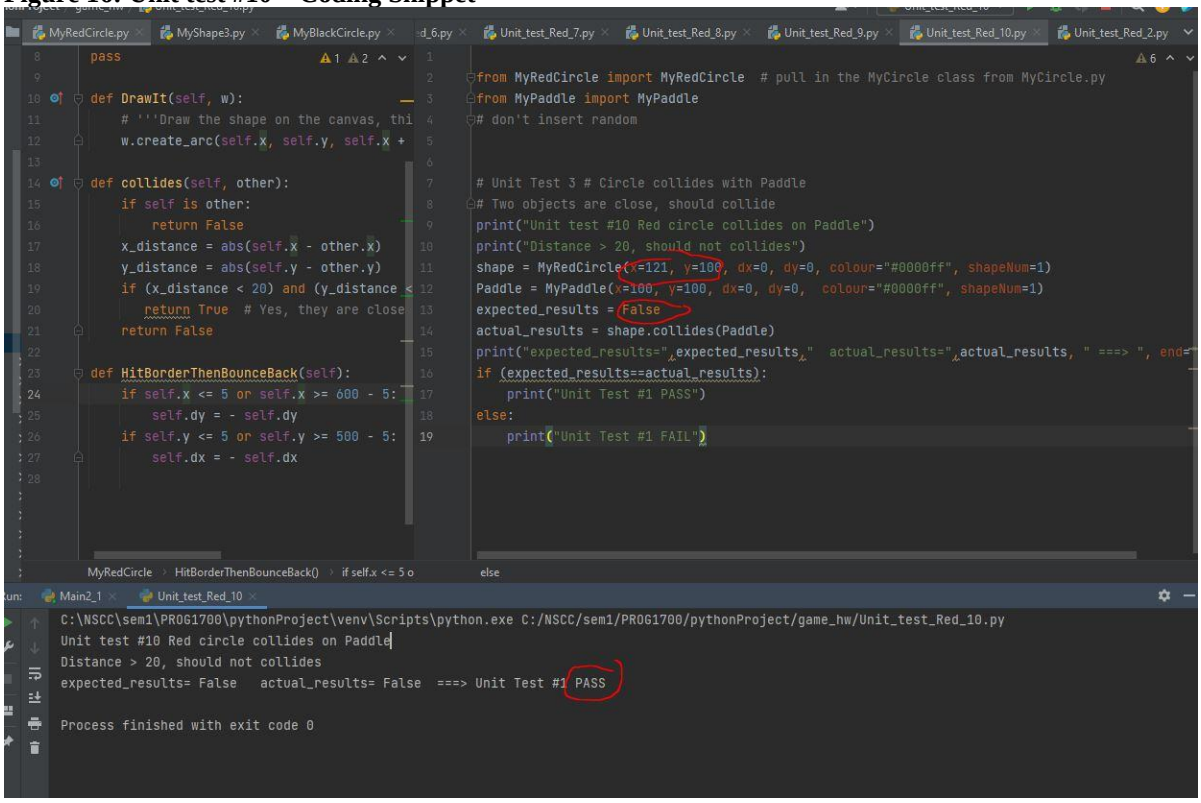
***Unit Test#9 –Circle collides with Paddle – False* (120, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance > 20, should be collided, unit test#1.1=PASS

Figure 15: Unit test #9– Coding Snippet



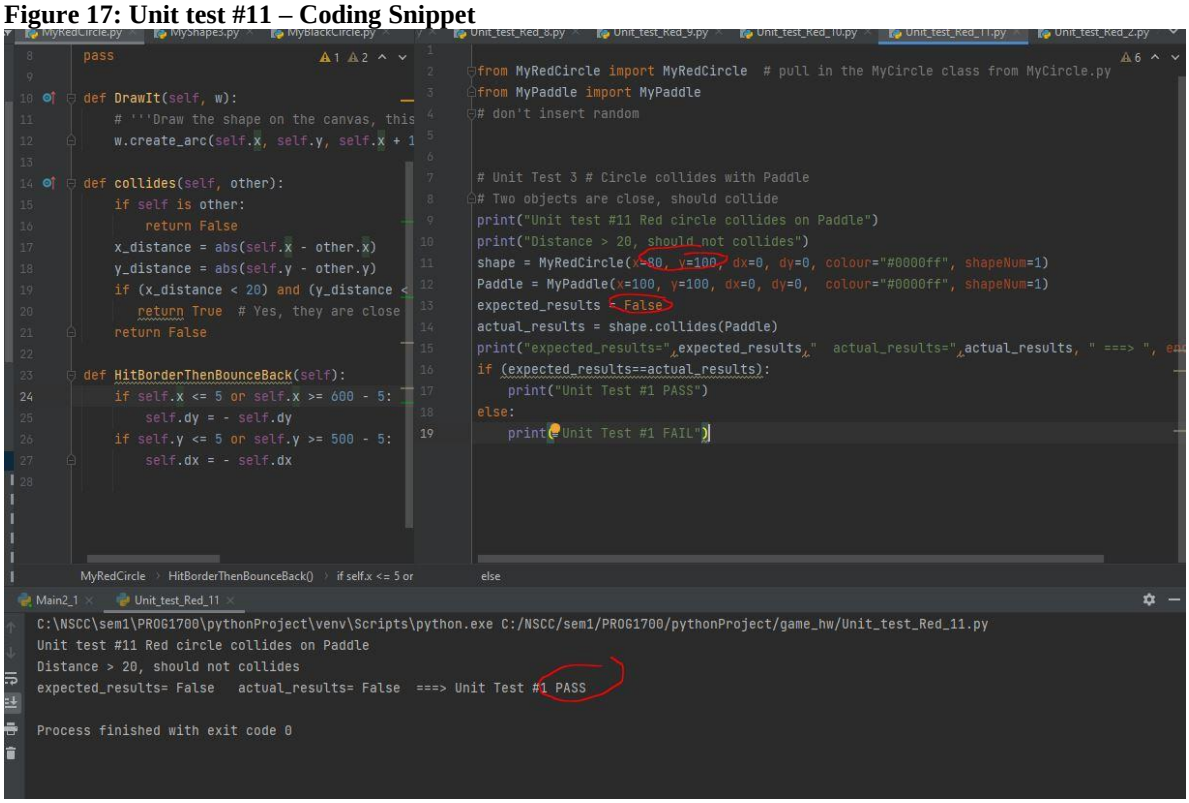
***Unit Test#10 –Circle collides with Paddle – False* (121, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance > 20, should be collided, unit test#1.1=PASS

Figure 16: Unit test #10 – Coding Snippet



***Unit Test#11 –Circle collides with Paddle – False* (80, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance > 20, should be collided, unit test#1.1=PASS

Figure 17: Unit test #11 – Coding Snippet



***Unit Test#12 –Circle collides with Paddle – False* (79, 100)**
In Left-hand side is the collides function in python
In the right-hand side is the unit testing code
Two objects distance > 20, should be collided, unit test#1.1=PASS

Figure 18: Unit test #12 – Coding Snippet

```
        pass

    def DrawIt(self, w):
        # '''Draw the shape on the canvas, t
        w.create_arc(self.x, self.y, self.x

    def collides(self, other):
        if self is other:
            return False
        x_distance = abs(self.x - other.x)
        y_distance = abs(self.y - other.y)
        if (x_distance < 20) and (y_distance
            return True  # Yes, they are clos
        return False

    def HitBorderThenBounceBack(self):
        if self.x <= 5 or self.x >= 600 - 5:
            self.dy = - self.dy
        if self.y <= 5 or self.y >= 500 - 5:
            self.dx = - self.dx
```

```
from MyRedCircle import MyRedCircle  # pull in the MyCircle class from MyCircle.py
from MyPaddle import MyPaddle
# don't insert random


# Unit Test 3 # Circle collides with Paddle
# Two objects are close, should collide
print("Unit test #12 Red circle collides on Paddle")
print("Distance > 20, should not collides")
shape = MyRedCircle(x=79, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
Paddle = MyPaddle(x=100, y=100, dx=0, dy=0,  colour="#0000ff", shapeNum=1)
expected_results = False
actual_results = shape.collides(Paddle)
print("expected_results=",expected_results,"  actual_results=",actual_results, " ===> ", end="")
if (expected_results==actual_results):
    print("Unit Test #1 PASS")
else:
    print("Unit Test #1 FAIL")
```

```
MyRedCircle › HitBorderThenBounceBack() › if self.x <=        else
Main2_1 × Unit_test_Red_12 ×
C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:/NSCC/sem1/PROG1700/pythonProject/game_hw/Unit_test_Red_12.py
Unit test #12 Red circle collides on Paddle
Distance > 20, should not collides
expected_results= False   actual_results= False  ===> Unit Test #1 PASS

Process finished with exit code 0
```

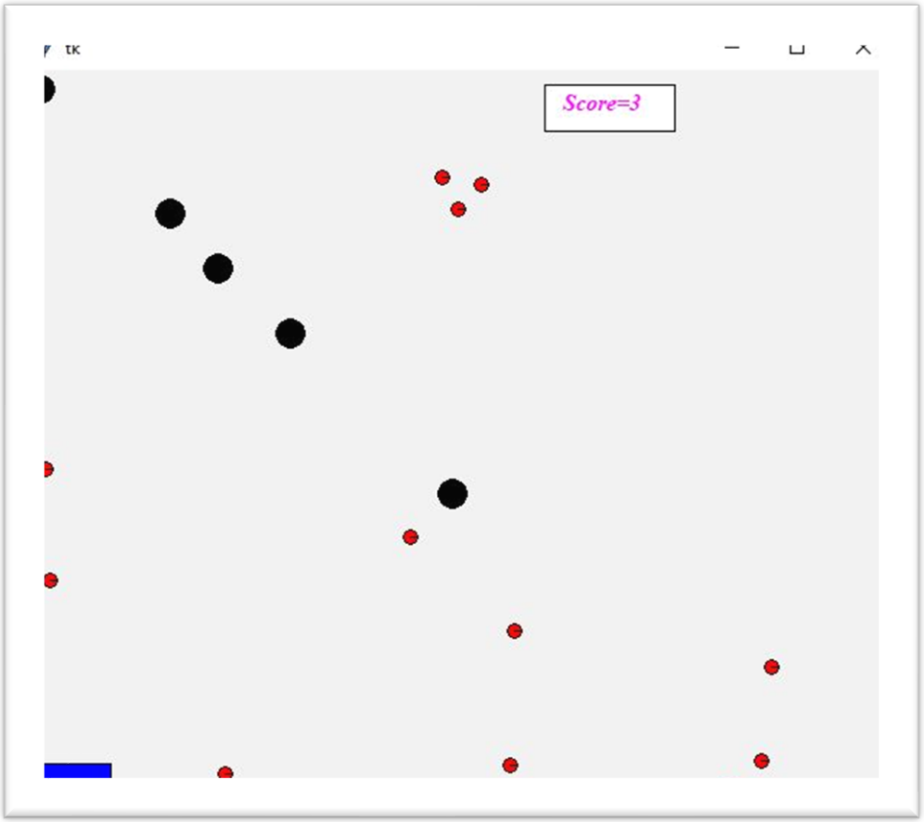
7 Integration Testing

Shape Circle in both red and black colour start to move and bounce in the canvas, shape paddle stay at the bottom of the canvas, user can use keyboard left/right/up/down or W/A/S/D to control move left and move right in the canvas.

For the result of collision, when red circle and shape paddle collided, score will increase one start from zero. The red circle will delete on canvas after collided with paddle, the canvas will shut down until all the red circle deleted from the canvas.

When the black circle collided with the paddle, the canvas will shut down immediately.

Figure 19: Integration test - Outcome Snippet



Appendix A. Requirements Traceability Matrix

(*Catch the Red, Avoid the Black-2D pong game*)

Paragraph Defined	Requirements Text	Paragraph Tested
3.1.1	User can run the file in python to start the game	7
3.1.2	User control the paddle – move left and right by using the keyboard	7
3.1.2	User can collide the red circle by using the controlled paddle	6.1
3.1.2	User can control the paddle to escape from the black circle	7
3.1.2	User controlled paddle can being hit by the black circle	7
3.1.2	Red circle can be collided with the user-controlled paddle	6.1
3.1.2	circle will be decrease after collided with the user controlled paddle	7
3.1.2	Score will increase one point after paddle collided with the red circle	7
3.1.2	Black circle can hit the paddle	6.1
3.1.2	Black will not be decrease after hit the paddle	7
3.1.3	End game situation 1 – User win the game Catching all the red circle	7
3.1.3	End game situation 2 – User lose the game Being hit by the black circle	7
3.1.4	Quit tkinter – User satisfied the end game situation	7