

PROG1300
Project
Documentation
For
Catch the Red,
Avoid the Black!

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

Table of Contents

1. Project Overview	1
2. Project Requirements	1
3.1 Use Case Diagram.....	2
3.1.1 Use Case #1	3
3.1.2 Use Case #2.....	3
3.1.3 Use Case #3.....	4
3.1.4 Use Case #4.....	4
3.2 Activity Diagrams.....	5
4 Design Plans	6
4.1 Sequence Diagrams.....	6
5 Implementation.....	7
5.1 Screen Snippet	7
5.2.1 Code Snippet #1	9
5.2.2 Code Snippet #2	9
6 Testing Paragraphs	10
6.1 Unit Testing	10
7 Integration Testing.....	23
Appendix A. Requirements Traceability Matrix (<i>Catch the Red, Avoid the Black-2D pong game</i>)	24

List of Tables

Table 1: Code #1 – Win game determination	9
Table 2: Code #2 – Lose game determination	9

List of Figures

Figure 1: Use cases diagram - Catch the Red, Avoid the Black	2
Figure 2: Activity diagram - Catch the Red, Avoid the Black.....	2
Figure 3: Sequence diagram - Catch the Red, Avoid the Black	6
Figure 4: Implementation 1-How to win and end game	7
Figure 5: Implementation 2 – How to lose and end game	8
Figure 6: Unit test – Collide condition	10
Figure 7: Unit test #1 – Coding Snippet	11
Figure 8: Unit test #2 – Coding Snippet	12
Figure 9: Unit test #3– Coding Snippet	13
Figure 10: Unit test #4– Coding Snippet	14
Figure 11: Unit test #5– Coding Snippet	15
Figure 12: Unit test #6– Coding Snippet	16
Figure 13: Unit test #7– Coding Snippet	17
Figure 14: Unit test #8– Coding Snippet	18
Figure 15: Unit test #9– Coding Snippet	19
Figure 16: Unit test #10 – Coding Snippet	20
Figure 17: Unit test #11 – Coding Snippet	21
Figure 18: Unit test #12 – Coding Snippet	22
Figure 19: Integration test - Outcome Snippet.....	23

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.

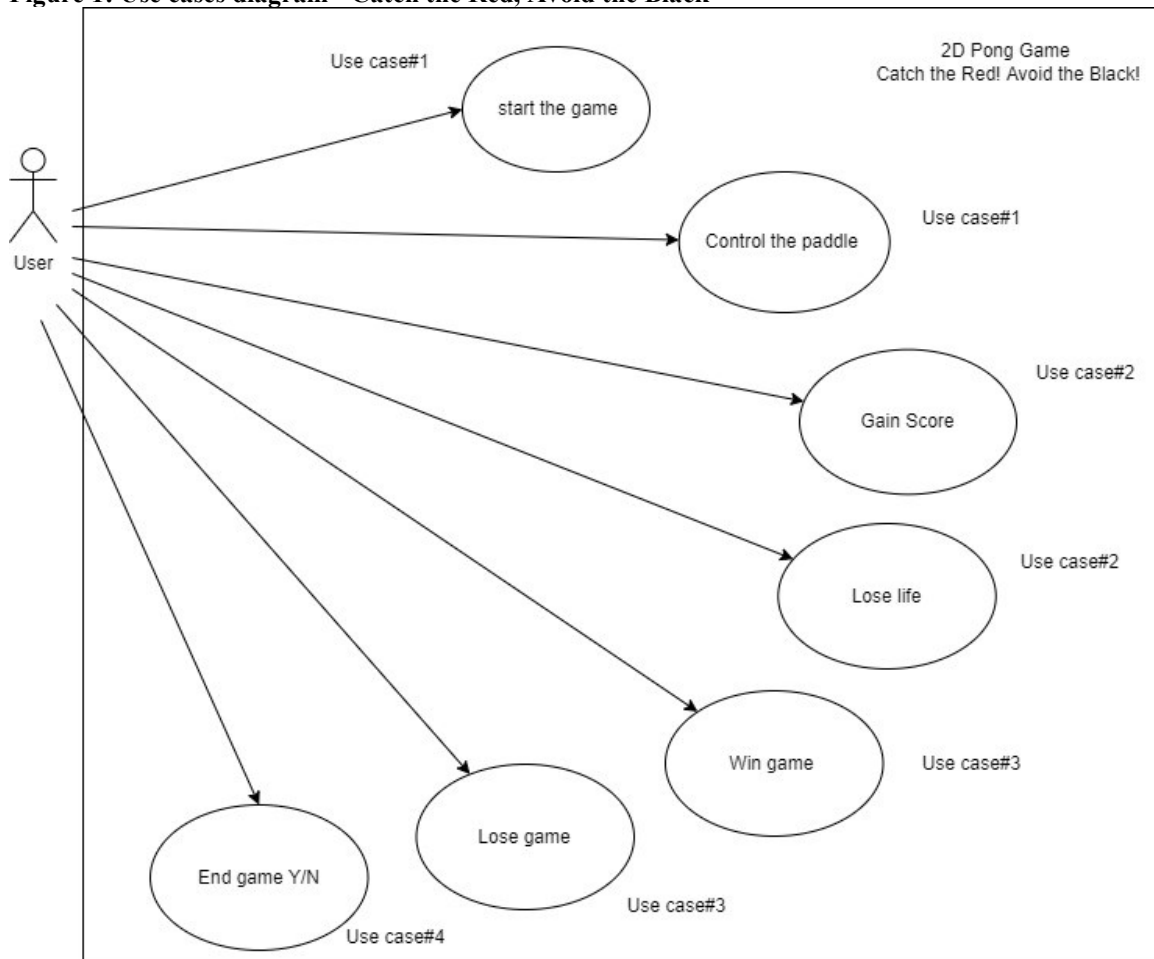
2. Project Requirements

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

<i>Use Case 1</i>	<i>User start the game, use the keyboard to play</i>
<i>Pre-conditions</i>	<i>User can run and start to play the game</i>
<i>Steps</i>	<ol style="list-style-type: none">1. <i>Open the game file</i>2. <i>Run the code in PyCharm</i>3. <i>Every element in the game can be run and act successfully</i>4. <i>Use the keyboard to control the paddle: AD or left right</i>
<i>Post-conditions</i>	<i>Run the game successfully and start to control the paddle with keyboard</i>

3.1.2 Use Case #2

<i>Use Case 2</i>	<i>Gain Score / Lose Life</i>
<i>Pre-conditions</i>	<i>User can control the paddle to gain score or lose life</i>
<i>Steps</i>	<ol style="list-style-type: none">1. <i>User Control the paddle</i>2. <i>To move the paddle left and right, and try to catch the bouncing red circle and avoid hitting by the black circle</i>3. <i>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</i>4. <i>System will display the score and the lifetime on the canvas</i>
<i>Post-conditions</i>	<i>User catch the red circle to get one point or user being hit by the black circle and decreases one lifetime</i>

3.1.3 Use Case #3

<i>Use Case 3</i>	<i>Win / Lose</i>
<i>Pre-conditions</i>	<i>User and win or lose game</i>
<i>Steps</i>	<ol style="list-style-type: none"> 1. <i>Users catch all the circle</i> 2. <i>Red circle will decrease on the canvas when being caught</i> 3. <i>When a user catches all the red circle, this action will let the user win the game,</i> 4. <i>Tkinter canvas will quit when game ends</i> 5. <i>When user being hit by the black circle over the three lifetimes limited, this action will make the user lose the game</i> 6. <i>Tkinter canvas will quit, and the game will end</i>
<i>Post-conditions</i>	<i>User ends the game by win or lose</i>

3.1.4 Use Case #4

<i>Use Case 4</i>	<i>Users continue to play the game or end game</i>
<i>Pre-conditions</i>	<i>Users can run again the python game.py to play the game or choose not to run the game</i>
<i>Steps</i>	<ol style="list-style-type: none"> 1. <i>User can play again after win or lose</i> 2. <i>If the user does not want to play again, the game ends and the game window quits.</i>
<i>Post-conditions</i>	<i>User can choice to play again after both win and lose, or choice the end game</i>

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:

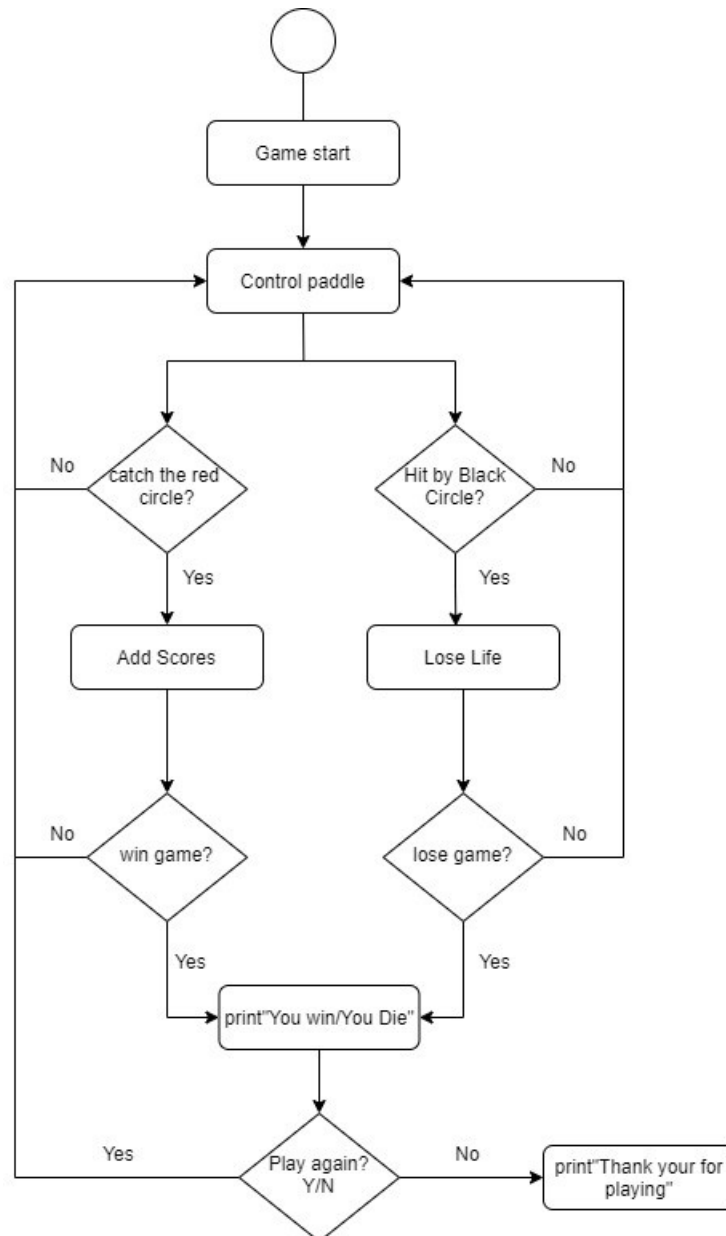
a) Win(condition): Catch all the red circle in the canvas without being hit by the black circle over 3 times

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

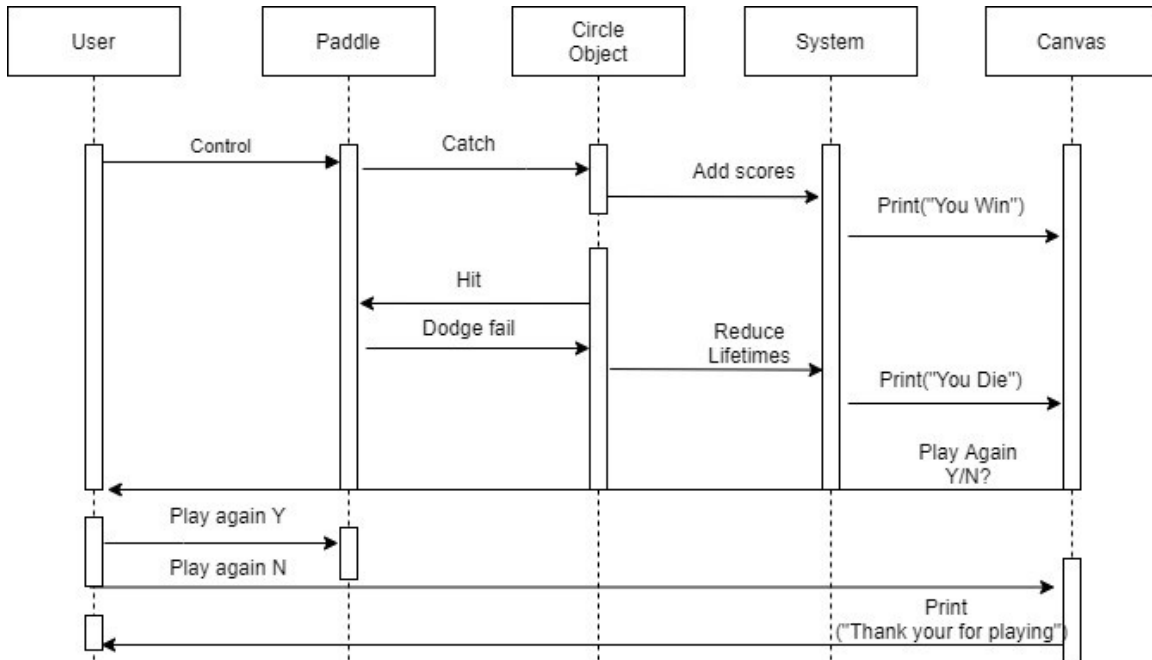
Figure 2: Activity diagram - Catch the Red, Avoid the Black



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

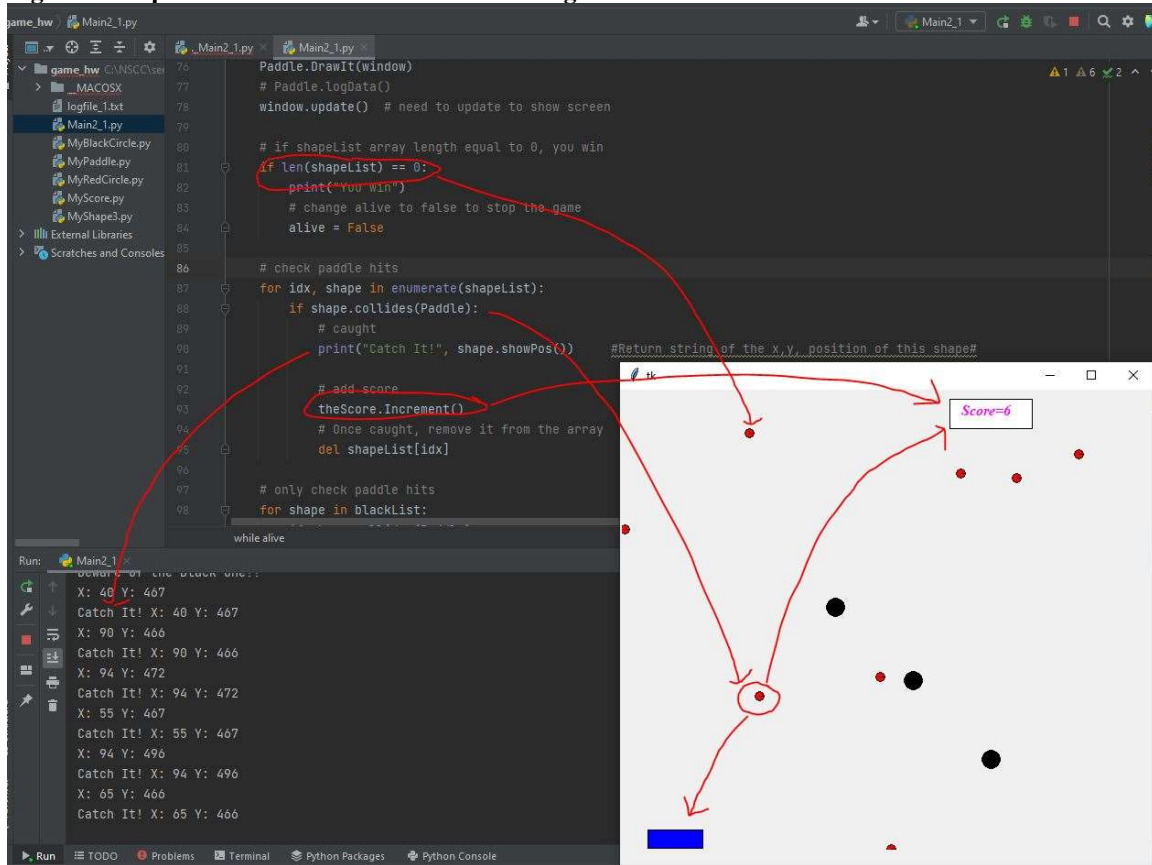
Important implementations: End game condition – You Win or You died

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 become 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

```

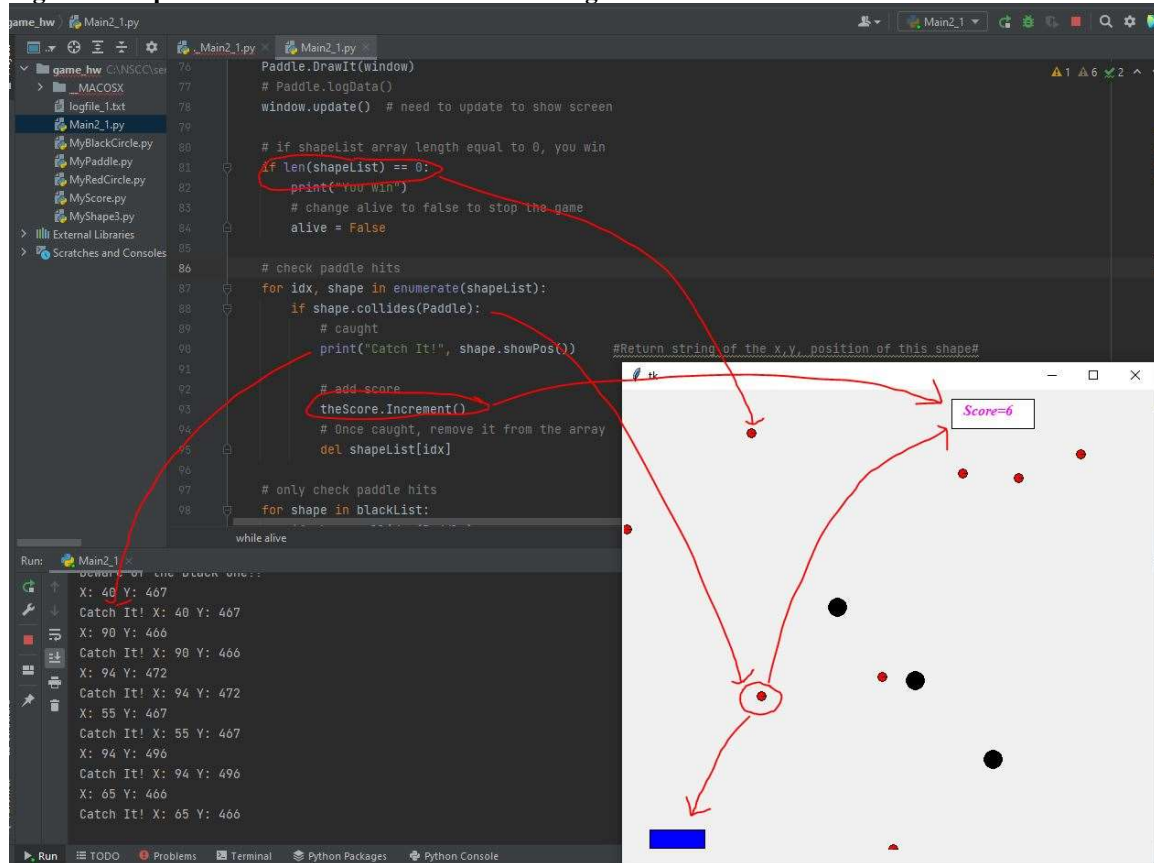
PROG1300 Project Documentation – Catch the red, avoid the black!

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



You Died!

Process finished with exit code 0

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)
```

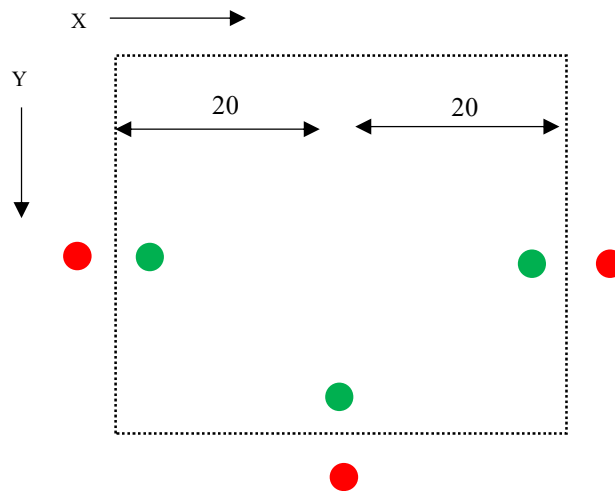


Figure 6: Unit test – Collide condition

- 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)

PROG1300 Project Documentation – Catch the red, avoid the black!

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

```

1  # sample unit text to show how the rest of the code need not be running
2  # only the unit test driver and the code being unit tested.
3
4
5
6
7  from MyRedCircle import MyRedCircle # pull in the MyCircle class from MyCircle.py
8  from MyPaddle import MyPaddle
9  # don't insert random
10
11
12 # Unit Test 1 # Circle collides with Paddle
13 # Two objects are close, should collide
14 print("Unit test #1 Red circle collides on Paddle")
15 print("Distance close, should collides")
16 shape = MyRedCircle(x=100, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
17 Paddle = MyPaddle(x=100, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
18 expected_results = True
19 actual_results = shape.collides(Paddle)
20 print("expected_results=", expected_results, " actual_results=", actual_results, " ==>")
21 if (expected_results==actual_results):
22     print("Unit Test #1 PASS")
23 else:
24     print("Unit Test #1 FAIL")
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

MyRedCircle > collides()

Unit test #1 Red circle collides on Paddle
Distance close, should collides
expected_results= True actual_results= True ==> Unit Test #1 PASS
Process finished with exit code 0

PROG1300 Project Documentation – Catch the red, avoid the black!

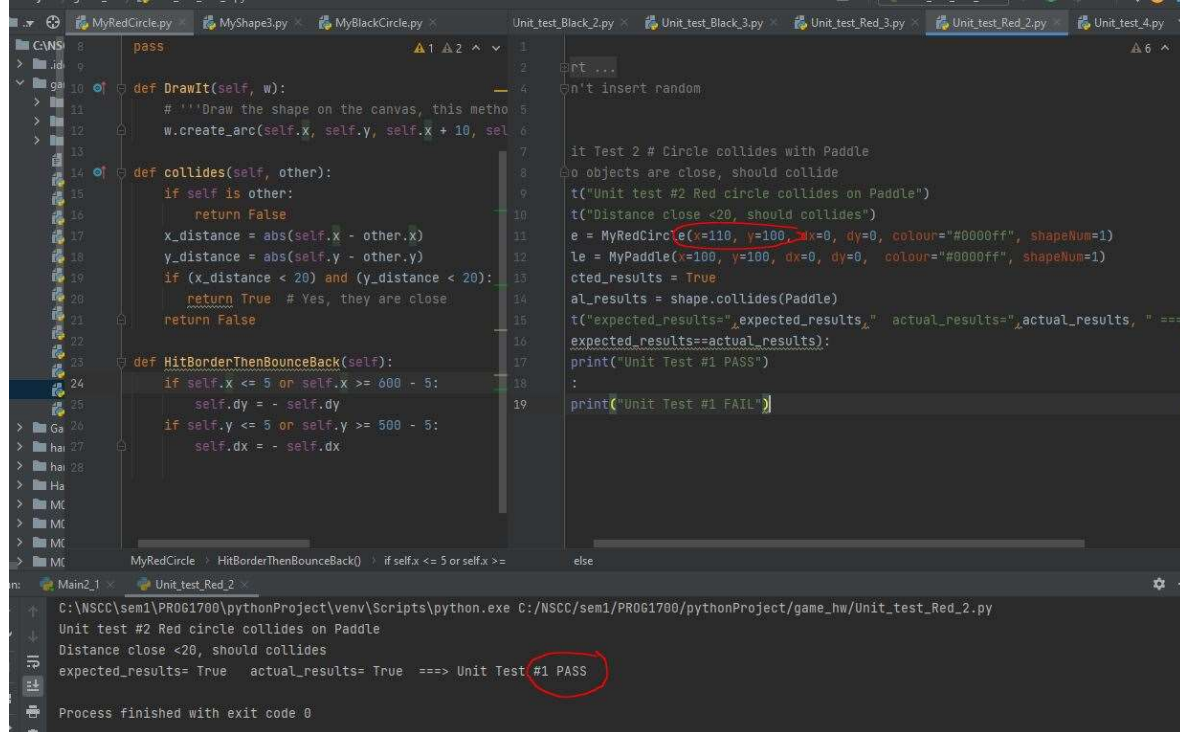
Unit Test#2 – Circle collides with Paddle – True (110, 100)

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



```
pass
def DrawIt(self, w):
    #'''Draw the shape on the canvas, this metho
    w.create_arc(self.x, self.y, self.x + 10, sel
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 < 20):
        return True # Yes, they are close
    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

Unit test ...
n't insert random

if Test 2 # Circle collides with Paddle
o objects are close, should collide
t("Unit test #2 Red circle collides on Paddle")
t("Distance close <20, should collides")
e = MyRedCircle(x=110, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
le = MyPaddle(x=100, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
cted_results = True
al_results = shape.collides(Paddle)
t("expected_results=", expected_results, " actual_results=", actual_results, " ==
expected_results==actual_results):
print("Unit Test #1 PASS")
:
print("Unit Test #1 FAIL")

C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:\NSCC\sem1\PROG1700\pythonProject\game_hw\Unit_test_Red_2.py
Unit test #2 Red circle collides on Paddle
Distance close <20, should collides
expected_results= True actual_results= True ==> Unit Test #1 PASS
Process finished with exit code 0
```

PROG1300 Project Documentation – Catch the red, avoid the black!

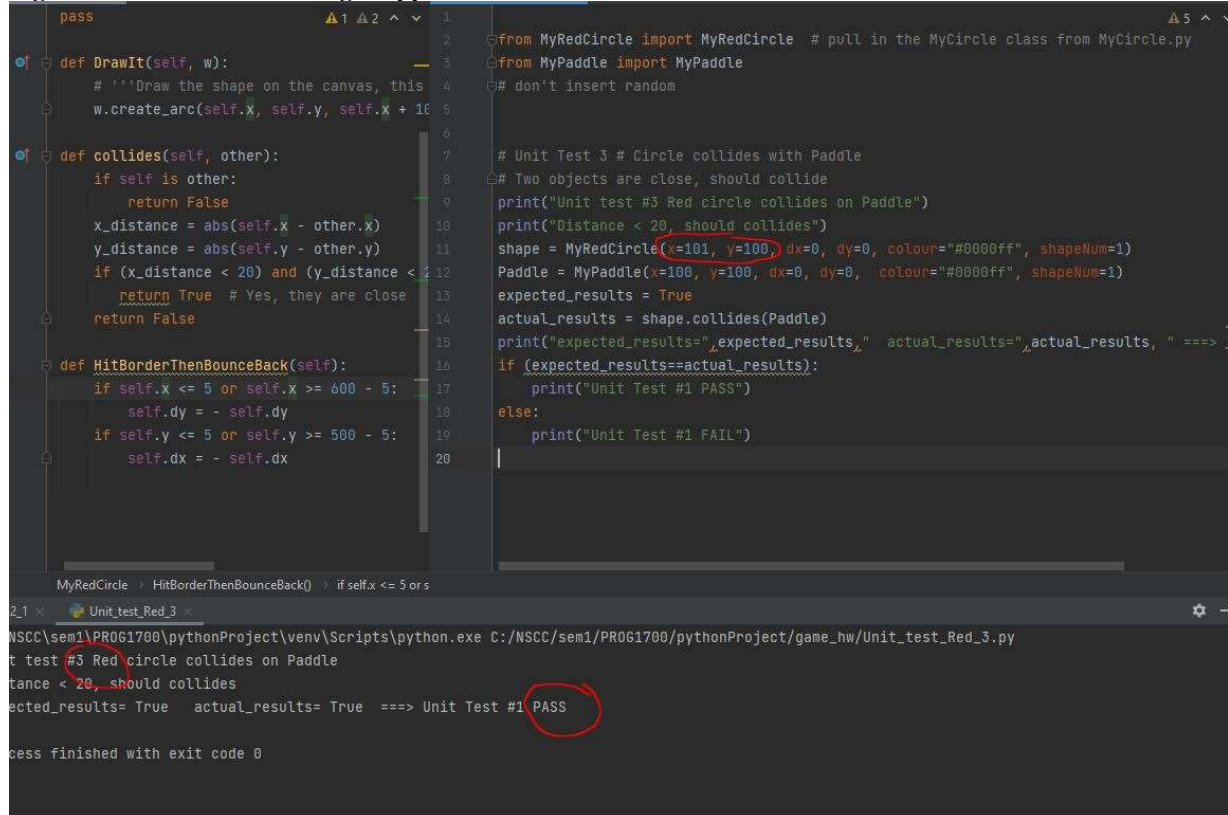
Unit Test#3 – Circle collides with Paddle – True (101, 100)

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



```
pass
def DrawIt(self, w):
    # '''Draw the shape on the canvas, this
    w.create_arc(self.x, self.y, self.x + 100, self.y + 100, fill='red')
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 < 20):
        return True # Yes, they are close
    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 #3 Red circle collides on Paddle")
print("Distance < 20, should collides")
shape = MyRedCircle(x=101, 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 = True
actual_results = shape.collides(Paddle)
print("expected_results=", expected_results, " actual_results=", actual_results, " ==> ")
if (expected_results==actual_results):
    print("Unit Test #1 PASS")
else:
    print("Unit Test #1 FAIL")

MyRedCircle > HitBorderThenBounceBack() > if self.x <= 5 or s
2.1 x Unit_test_Red_3 x
NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:/NSCC/sem1/PROG1700/pythonProject/game_hw/Unit_test_Red_3.py
Unit test #3 Red circle collides on Paddle
Distance < 20, should collides
expected_results= True actual_results= True ==> Unit Test #1 PASS
Process finished with exit code 0
```


PROG1300 Project Documentation – Catch the red, avoid the black!

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

The screenshot displays a code editor with two Python files open:

- MyRedCircle.py**: Contains methods for drawing and collision detection.
 - `DrawIt(self, w):` Draws the shape on the canvas.
 - `collides(self, other):` Checks if two objects are close enough to collide based on distance and radius.
 - `HitBorderThenBounceBack(self):` Handles boundary collisions by reversing velocity components.
- Unit_test_Red_4.py**: Contains unit tests for the circle class.
 - Imports `MyRedCircle` from `MyCircle` and `MyPaddle`.
 - Creates a `MyRedCircle` object with `x=100, y=100`.
 - Tests the `collides` method against a `Paddle` object.

A red circle highlights the coordinates `x=100, y=100` in both files, indicating they refer to the same point in space.

The terminal output at the bottom shows the execution of the test script:

```
C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:\NSCC\sem1\PROG1700\pythonProject\game_hw\Unit_test_Red_4.py
Unit test #4 Red circle collides on Paddle
Distance < 20, should collides
expected_results= True actual_results= True ==> Unit Test #1 PASS
Process finished with exit code 0
```


PROG1300 Project Documentation – Catch the red, avoid the black!

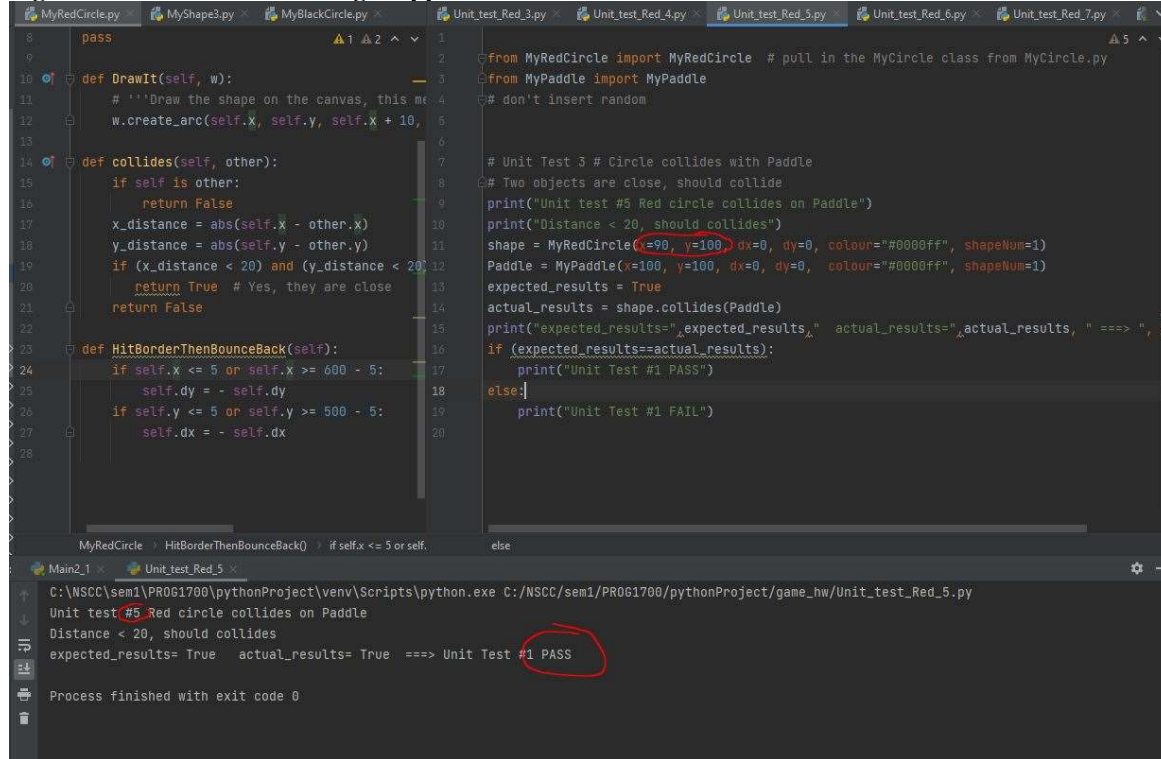
Unit Test#5 – Circle collides with Paddle – True (90, 100)

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



```
8 pass
9
10 def DrawIt(self, w):
11     # '''Draw the shape on the canvas, this me
12     w.create_arc(self.x, self.y, self.x + 10,
13
14 def collides(self, other):
15     if self is other:
16         return False
17     x_distance = abs(self.x - other.x)
18     y_distance = abs(self.y - other.y)
19     if (x_distance < 20) and (y_distance < 20):
20         return True # Yes, they are close
21     return False
22
23 def HitBorderThenBounceBack(self):
24     if self.x <= 5 or self.x >= 600 - 5:
25         self.dy = - self.dy
26     if self.y <= 5 or self.y >= 500 - 5:
27         self.dx = - self.dx
28
1 from MyRedCircle import MyRedCircle # pull in the MyCircle class from MyCircle.py
2 from MyPaddle import MyPaddle
3 # don't insert random
4
5 # Unit Test 3 # Circle collides with Paddle
6 # Two objects are close, should collide
7 print("Unit test #5 Red circle collides on Paddle")
8 print("Distance < 20, should collides")
9 shape = MyRedCircle(x=90, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
10 Paddle = MyPaddle(x=100, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
11 expected_results = True
12 actual_results = shape.collides(Paddle)
13 print("expected_results=", expected_results, " actual_results=", actual_results, " ==> ",
14 if (expected_results==actual_results):
15     print("Unit Test #1 PASS")
16 else:
17     print("Unit Test #1 FAIL")
18
C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:/NSCC/sem1/PROG1700/pythonProject/game_hw/Unit_test_Red_5.py
Unit test #5 Red circle collides on Paddle
Distance < 20, should collides
expected_results= True  actual_results= True ==> Unit Test #1 PASS
Process finished with exit code 0
```

PROG1300 Project Documentation – Catch the red, avoid the black!

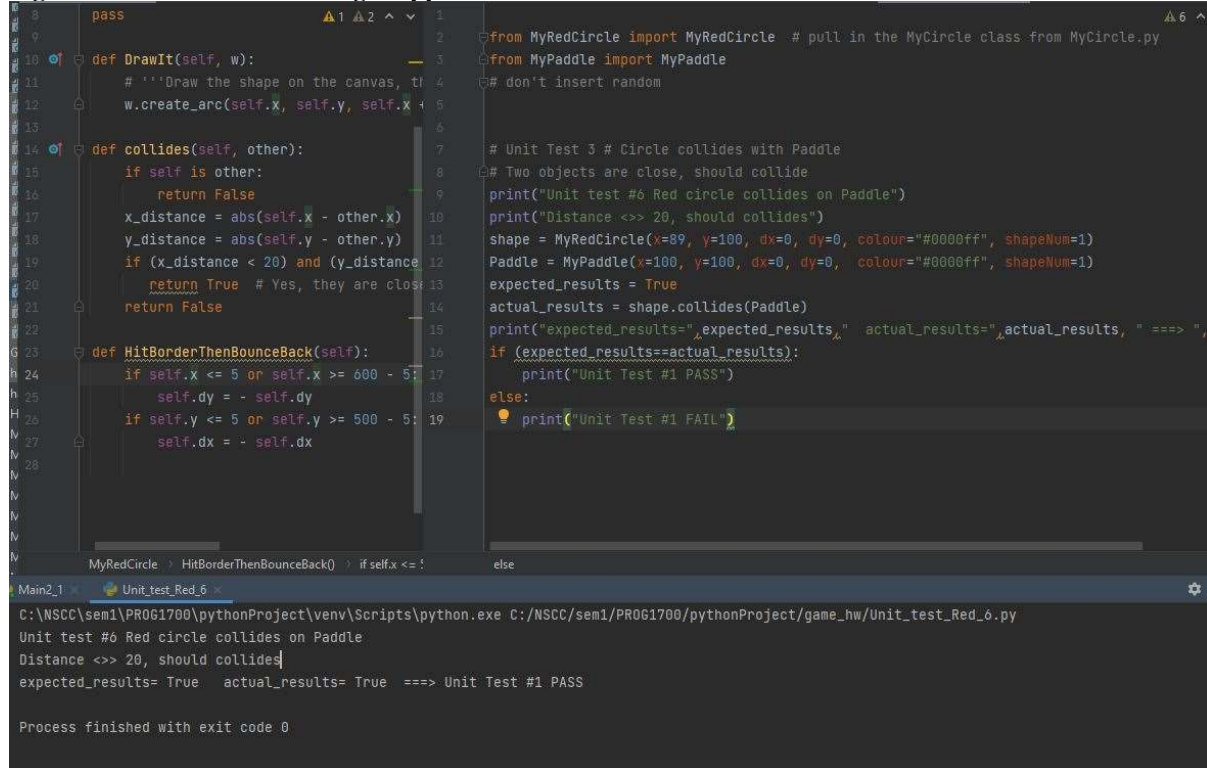
Unit Test#6 – Circle collides with Paddle – True (89, 100)

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



```
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 close
    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 #6 Red circle collides on Paddle")
print("Distance <>> 20, should collides")
shape = MyRedCircle(x=89, 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 = True
actual_results = shape.collides(Paddle)
print("expected_results=", expected_results, " actual_results=", actual_results, " ==> ",
if (expected_results==actual_results):
    print("Unit Test #1 PASS")
else:
    print("Unit Test #1 FAIL")

MyRedCircle HitBorderThenBounceBack() if self.x <= 5 else

Main2_1 Unit_test_Red_6
C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:\NSCC\sem1\PROG1700\pythonProject\game_hw\Unit_test_Red_6.py
Unit test #6 Red circle collides on Paddle
Distance <>> 20, should collides
expected_results= True actual_results= True ==> Unit Test #1 PASS

Process finished with exit code 0
```

PROG1300 Project Documentation – Catch the red, avoid the black!

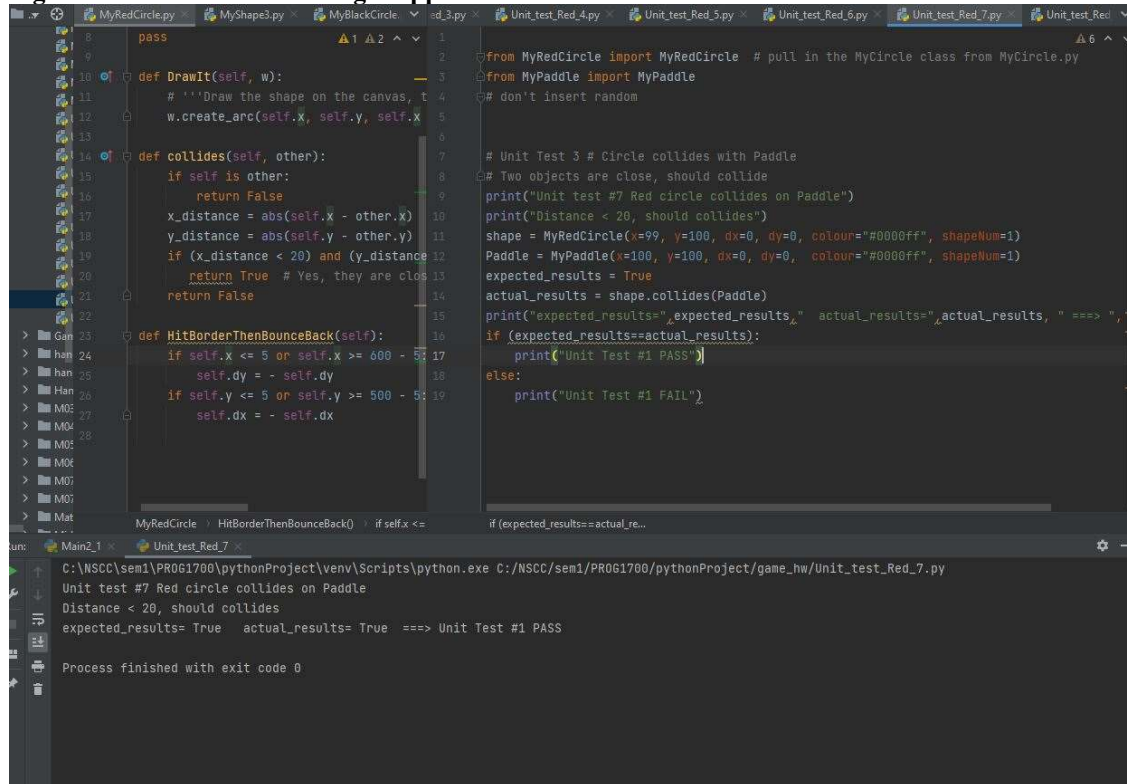
Unit Test#7 – Circle collides with Paddle – True (99, 100)

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



```
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 #7 Red circle collides on Paddle")
print("Distance < 20, should collides")
shape = MyRedCircle(x=99, 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 = True
actual_results = shape.collides(Paddle)
print("expected_results=", expected_results, " actual_results=", actual_results, " ==> ",
if (expected_results==actual_results):
    print("Unit Test #1 PASS")
else:
    print("Unit Test #1 FAIL")

C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:/NSCC/sem1/PROG1700/pythonProject/game_hw/Unit_test_Red_7.py
Unit test #7 Red circle collides on Paddle
Distance < 20, should collides
expected_results= True  actual_results= True  ==> Unit Test #1 PASS
Process finished with exit code 0
```

PROG1300 Project Documentation – Catch the red, avoid the black!

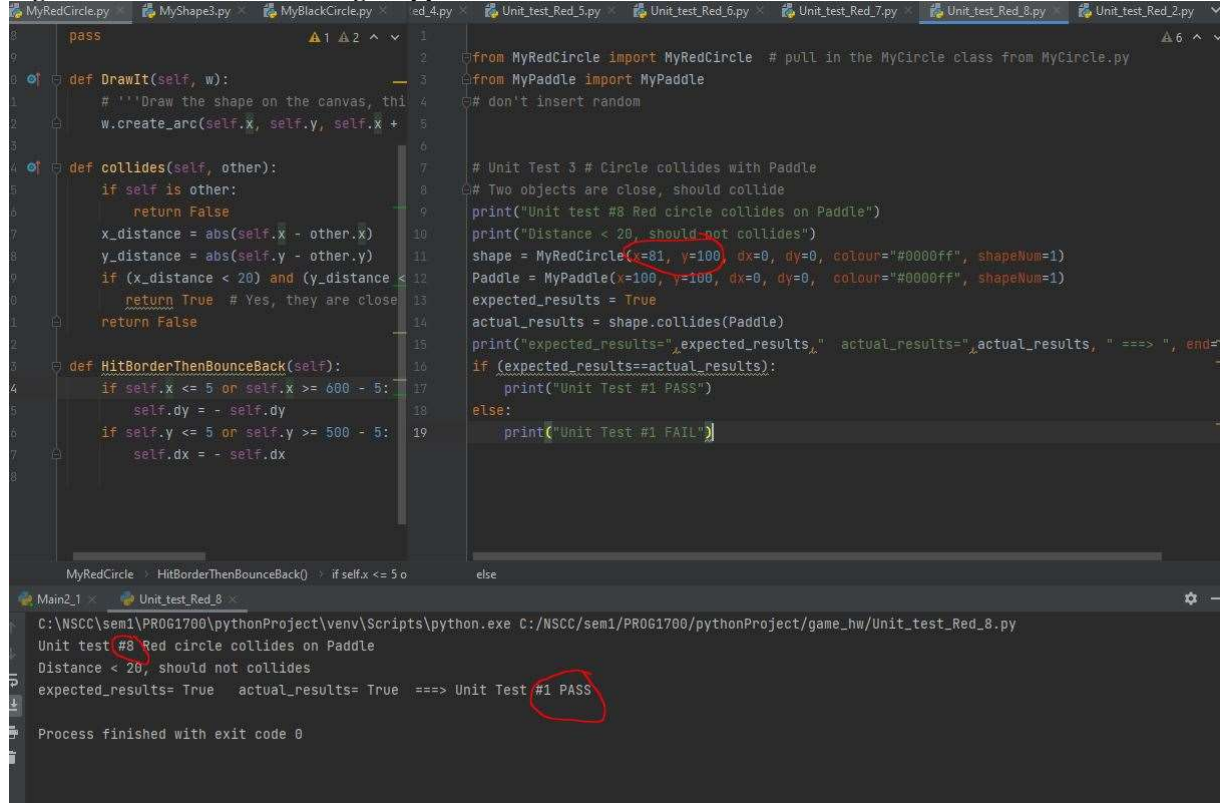
Unit Test#8 – Circle collides with Paddle – True (81, 100)

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



```
pass
def DrawIt(self, w):
    # '''Draw the shape on the canvas, this is a circle'''
    w.create_arc(self.x, self.y, self.x + 20, self.y + 20, style=ARC_CHORD, fill='red', outline='red')
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 < 20):
        return True # Yes, they are close
    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 #8 Red circle collides on Paddle")
print("Distance < 20, should not collides")
shape = MyRedCircle(x=81, 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 = True
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")
```

```
C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:\NSCC\sem1\PROG1700\pythonProject\game_hw\Unit_test_Red_8.py
Unit test #8 Red circle collides on Paddle
Distance < 20, should not collides
expected_results= True  actual_results= True  ==> Unit Test #1 PASS

Process finished with exit code 0
```

PROG1300 Project Documentation – Catch the red, avoid the black!

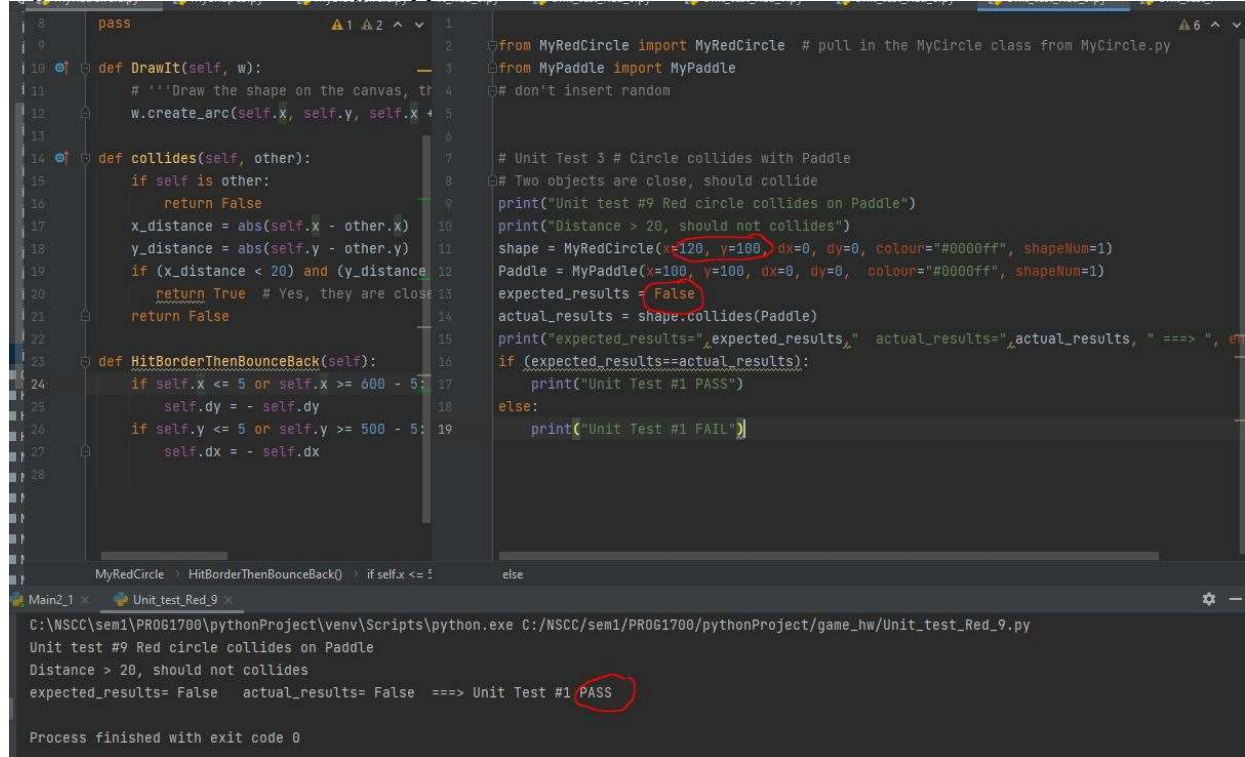
Unit Test#9 – Circle collides with Paddle – False (120, 100)

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



```
8 pass
9
10 def DrawIt(self, w):
11     # '''Draw the shape on the canvas, tr
12     w.create_arc(self.x, self.y, self.x +
13
14 def collides(self, other):
15     if self is other:
16         return False
17     x_distance = abs(self.x - other.x)
18     y_distance = abs(self.y - other.y)
19     if (x_distance < 20) and (y_distance
20         return True # Yes, they are close
21     return False
22
23 def HitBorderThenBounceBack(self):
24     if self.x <= 5 or self.x >= 600 - 5:
25         self.dy = - self.dy
26     if self.y <= 5 or self.y >= 500 - 5:
27         self.dx = - self.dx
28
1 from MyRedCircle import MyRedCircle # pull in the MyCircle class from MyCircle.py
2 from MyPaddle import MyPaddle
3 # don't insert random
4
5 # Unit Test 3 # Circle collides with Paddle
6 # Two objects are close, should collide
7 print("Unit test #9 Red circle collides on Paddle")
8 print("Distance > 20, should not collides")
9 shape = MyRedCircle(x=120, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
10 Paddle = MyPaddle(x=100, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
11 expected_results = False
12 actual_results = shape.collides(Paddle)
13 print("expected_results=", expected_results, " actual_results=", actual_results, " ==> ", end="")
14 if (expected_results==actual_results):
15     print("Unit Test #1 PASS")
16 else:
17     print("Unit Test #1 FAIL")
18
C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:/NSCC/sem1/PROG1700/pythonProject/game_hw/Unit_test_Red_9.py
Unit test #9 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
```

PROG1300 Project Documentation – Catch the red, avoid the black!

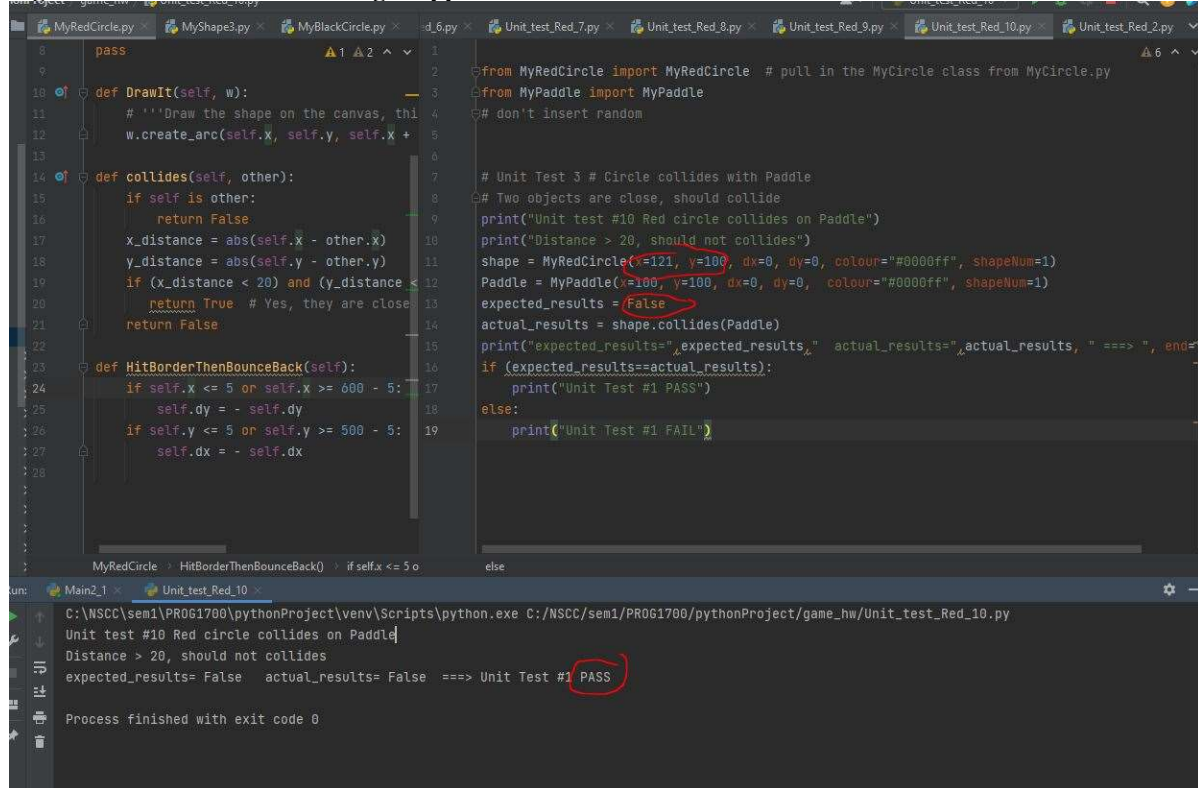
Unit Test#10 – Circle collides with Paddle – False (121, 100)

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



```
pass
def DrawIt(self, w):
    # 'Draw the shape on the canvas, this is a circle'
    w.create_arc(self.x, self.y, self.x + 10, self.y + 10, fill="red", outline="red", width=1)
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 < 20):
        return True # Yes, they are close
    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 #10 Red circle collides on Paddle")
print("Distance > 20, should not collides")
shape = MyRedCircle(x=121, 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")

C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:\NSCC\sem1\PROG1700\pythonProject\game_hw\Unit_test_Red_10.py
Unit test #10 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
```


PROG1300 Project Documentation – Catch the red, avoid the black!

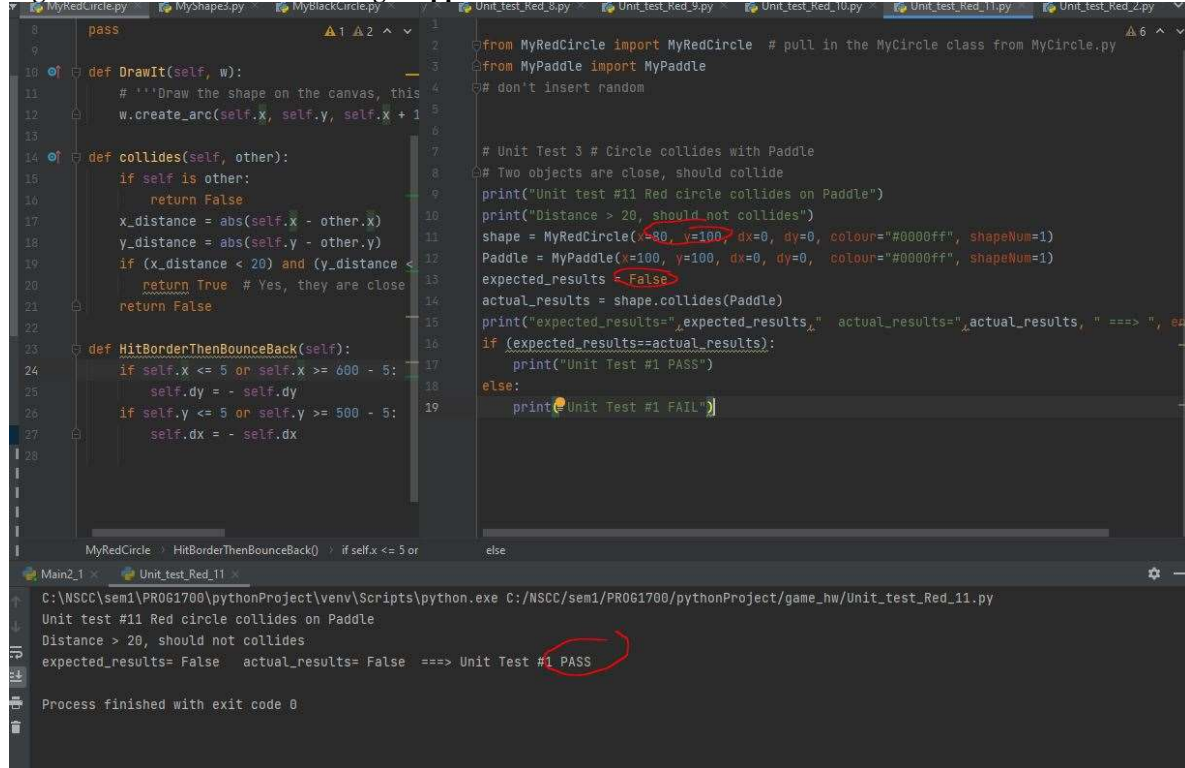
Unit Test#11 – Circle collides with Paddle – False (80, 100)

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



```
pass
def DrawIt(self, w):
    # ''Draw the shape on the canvas, this
    w.create_arc(self.x, self.y, self.x + 1
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 close
    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 #11 Red circle collides on Paddle")
print("Distance > 20, should not collides")
shape = MyRedCircle(x=80, 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")

C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:/NSCC/sem1/PROG1700/pythonProject/game_hw/Unit_test_Red_11.py
Unit test #11 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
```

PROG1300 Project Documentation – Catch the red, avoid the black!

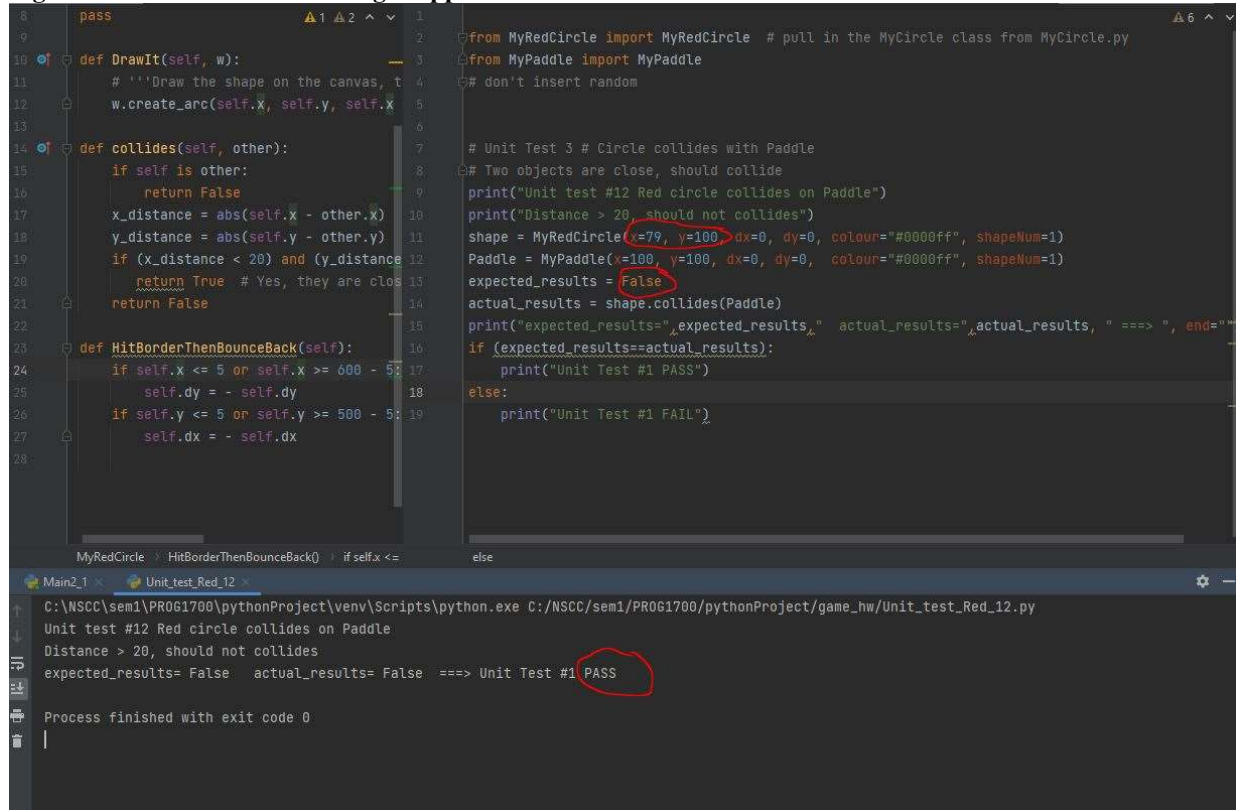
Unit Test#12 – Circle collides with Paddle – False (79, 100)

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



```
8 pass
9
10 def DrawIt(self, w):
11     # '''Draw the shape on the canvas, t
12     w.create_arc(self.x, self.y, self.x
13
14 def collides(self, other):
15     if self is other:
16         return False
17     x_distance = abs(self.x - other.x)
18     y_distance = abs(self.y - other.y)
19     if (x_distance < 20) and (y_distance
20         return True # Yes, they are clos
21     return False
22
23 def HitBorderThenBounceBack(self):
24     if self.x <= 5 or self.x >= 600 - 5:
25         self.dy = - self.dy
26     if self.y <= 5 or self.y >= 500 - 5:
27         self.dx = - self.dx
28
29 from MyRedCircle import MyRedCircle # pull in the MyCircle class from MyCircle.py
30 from MyPaddle import MyPaddle
31 # don't insert random
32
33 # Unit Test 3 # Circle collides with Paddle
34 # Two objects are close, should collide
35 print("Unit test #12 Red circle collides on Paddle")
36 print("Distance > 20, should not collides")
37 shape = MyRedCircle(x=79, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
38 Paddle = MyPaddle(x=100, y=100, dx=0, dy=0, colour="#0000ff", shapeNum=1)
39 expected_results = False
40 actual_results = shape.collides(Paddle)
41 print("expected_results=", expected_results, " actual_results=", actual_results, " ==> ", end="")
42 if (expected_results==actual_results):
43     print("Unit Test #1 PASS")
44 else:
45     print("Unit Test #1 FAIL")
46
47 C:\NSCC\sem1\PROG1700\pythonProject\venv\Scripts\python.exe C:/NSCC/sem1/PROG1700/pythonProject/game_hw/Unit_test_Red_12.py
48 Unit test #12 Red circle collides on Paddle
49 Distance > 20, should not collides
50 expected_results= False  actual_results= False ==> Unit Test #1 PASS
51
52 Process finished with exit code 0
```

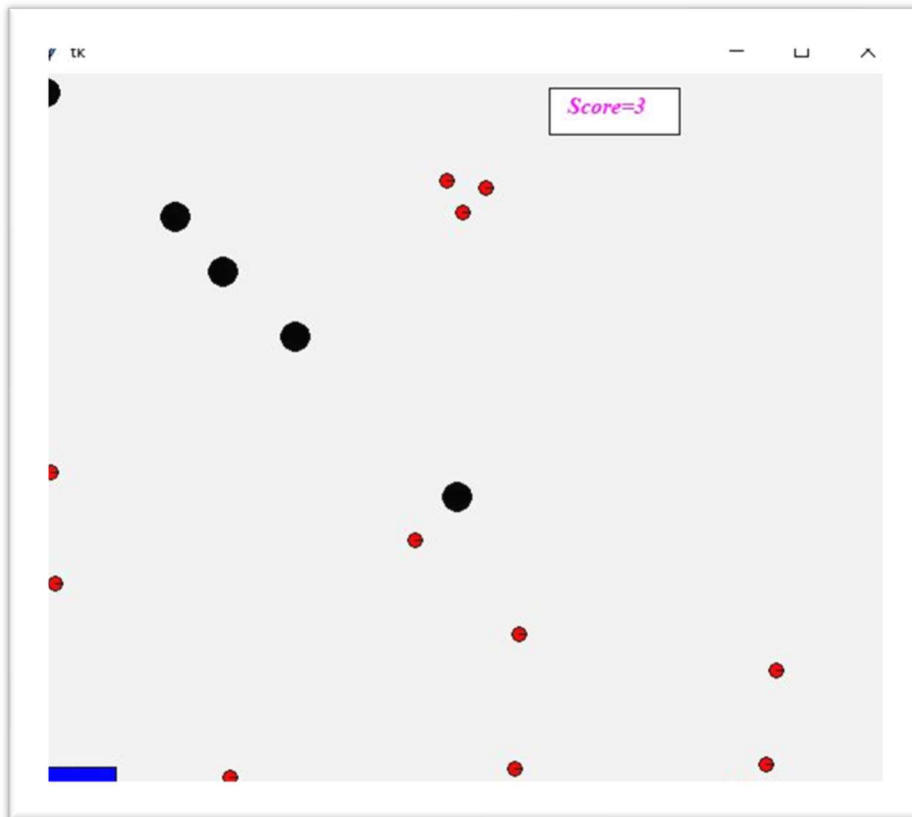

7 Integration Testing

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