Game of Pong **V9.0**

V9 Using Pythagoras Theorem for Collision Detection

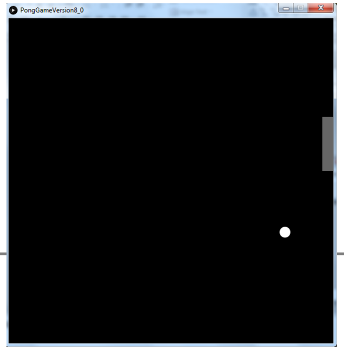
Produced by:

Dr. Siobhán Drohan
Mr. Colm Dunphy
Mr. Diarmuid O’Connor

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Pong Versions - introduction



v1 - **Ball moving** from left to right of screen. Can bounce off top or bottom

v2 - **Mouse controlling the Paddle**

v3 - **Collision detection** (ball bounces back). Changes made only to PongGame

v4 - **Game Over** (when 3 lives gone), Score (lives Lost). Output to Console. Changes made only to PongGame.

v5 - **Tournament** (no of games per tournament default is 5). Changes made only to PongGame.

v6 - new **Player class using arrays** (no statistics)

v7 - Player class using arrays (with **statistics** (Tournament Over - highest, lowest, average score))

v8 - **JOptionPane for I/O** instead of console

v9 - alternative algorithm using **Pythagoras Theorem**

Demo of
Pong Game V9.0
Same as V8.0

We introduced a
‘**Simple**’ **Collision Detection** Algorithm
in PongGameV3_0.

Now we will look
at a more complex, versatile algorithm,
using **Pythagoras Theorem**!

‘Simple’ Collision Detection Algorithm

Method signature:

boolean **hitPaddle** (Paddle paddle, Ball ball)

Algorithm:

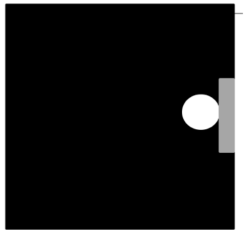
```
1) Measure the size of the gap between the paddle and the ball.

2) If the ball is too far away from the Paddle on the X axis to have a collision
        → return false

3) If the ball is too far away from the Paddle on the Y axis to have a collision
        → return false

4) Otherwise
        → return true.
```

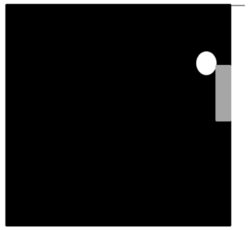
‘Pythagoras’ **Collision Detection** Algorithm

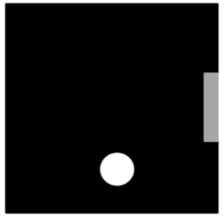
Method signature:

boolean **hitPaddle** (Paddle paddle, Ball ball)

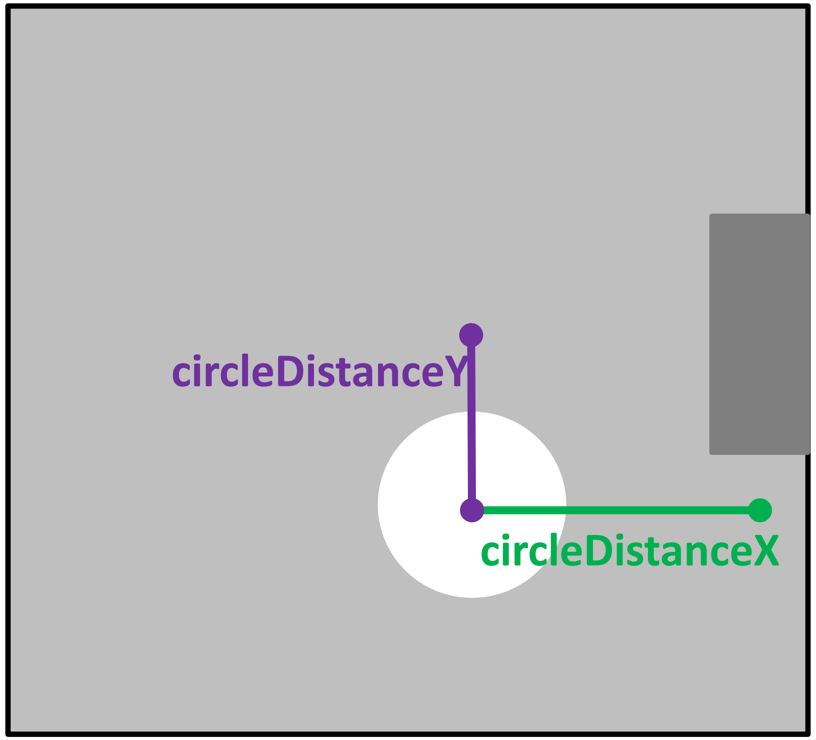
- Two collision approaches:
  1. The ball overlaps the paddle **straight on**,
     → returns true.
  2. The ball overlaps the **corner** of the paddle,
     → returns true.
- Non collision
  - If the ball **does not overlap** the paddle,
    → return false







‘Pythagoras’ Collision Detection Algorithm



First we work out the distances

```
float circleDistanceX
            = abs (ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);

float circleDistanceY
            =  abs (ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);
```

e.g. abs (-5) = 5

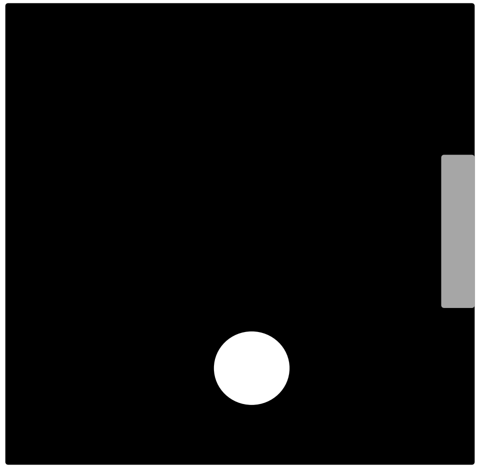
‘Pythagoras’ Collision Detection Algorithm

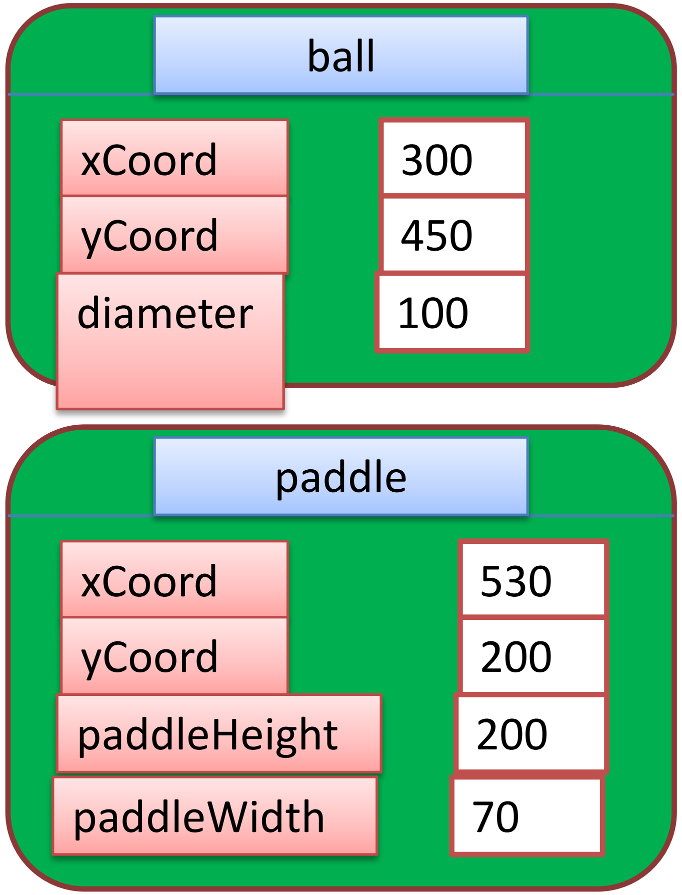
… the same code inside hitPaddle()

```
boolean hitPaddle (Paddle paddle, Ball ball)
{
  // These variables measure the magnitude of the gap
  // between the paddle and the ball.

   float circleDistanceX =
           abs(ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);

   float circleDistanceY =
           abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);

  // code omitted…
}
```

1) COLLISIONS - STRAIGHT ON

‘Pythagoras’ Collision Detection Algorithm - Ball & Paddle not overlapping





ball	
xCoord	300
yCoord	450
diameter	100

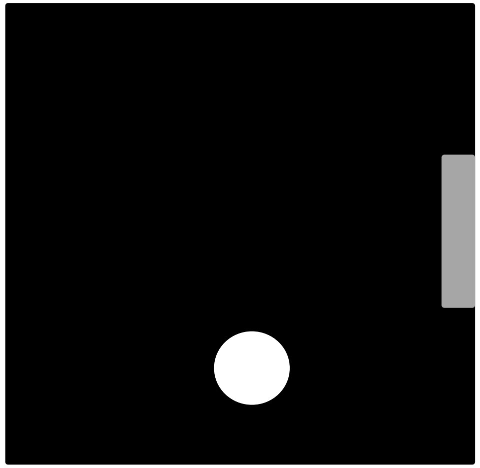
paddle	
xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

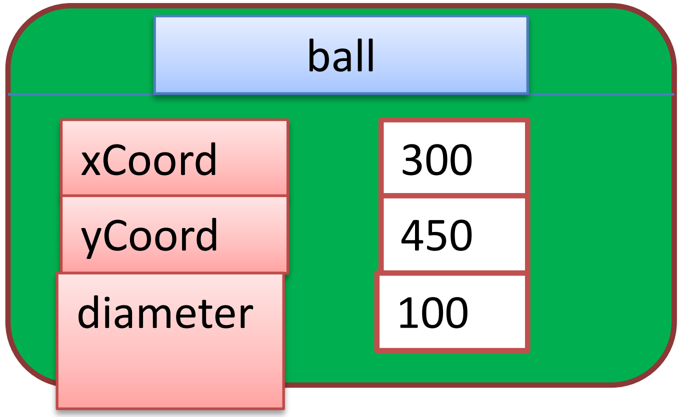
circleDistanceX = abs(300 – 530 – 35) **= 265**

circleDistanceY = abs(450 - 200 - 100) **= 150**

```
float circleDistanceX = abs(ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);
float circleDistanceY =  abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);
```

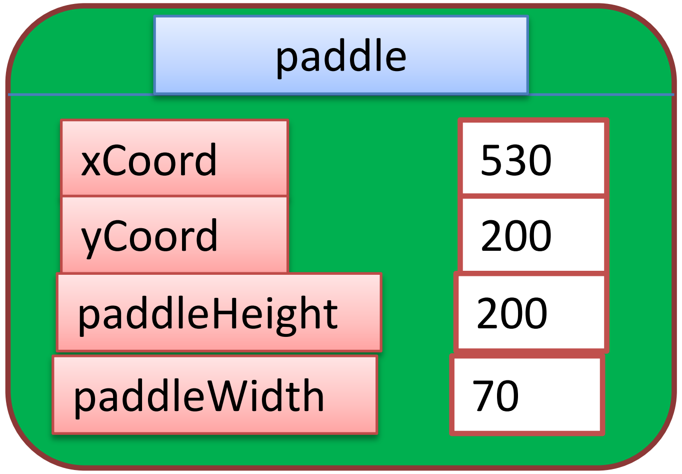
‘Pythagoras’ Collision Detection Algorithm
- Ball & Paddle not overlapping





ball

xCoord	300
yCoord	450
diameter	100



paddle

xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

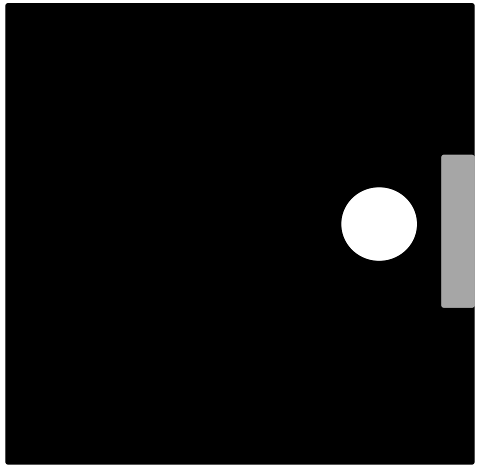
circleDistanceX = 265
circleDistanceY = 150

If (265 > (35 + 50))
→ returns from method with a **false**

i.e. ball and paddle have not made contact

```
if (circleDistanceX > (paddle.getPaddleWidth()/2  + ball.getDiameter()/2)) { return false; }
if (circleDistanceY > (paddle.getPaddleHeight()/2 + ball.getDiameter()/2)) { return false; }
```

‘Pythagoras’ Collision Detection Algorithm
- Ball & Paddle closer



circleDistanceX = abs(450 – 530 – 35) **= 115**
circleDistanceY = abs(300 - 200 - 100) **= 0**

ball

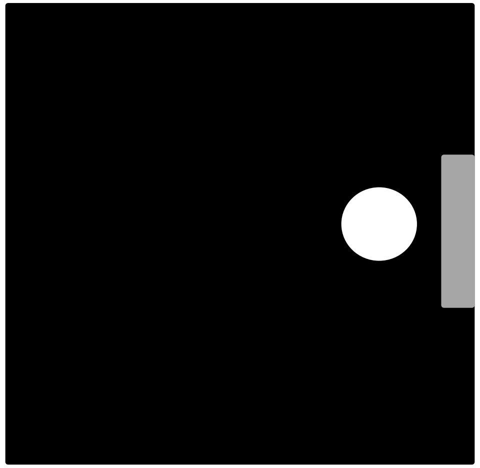
xCoord	450
yCoord	300
diameter	100

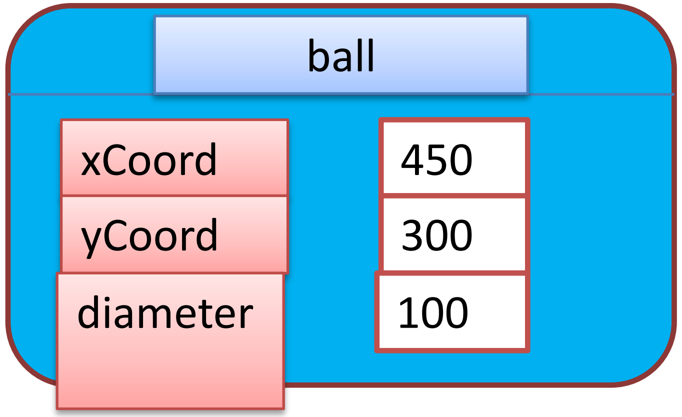
paddle

xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

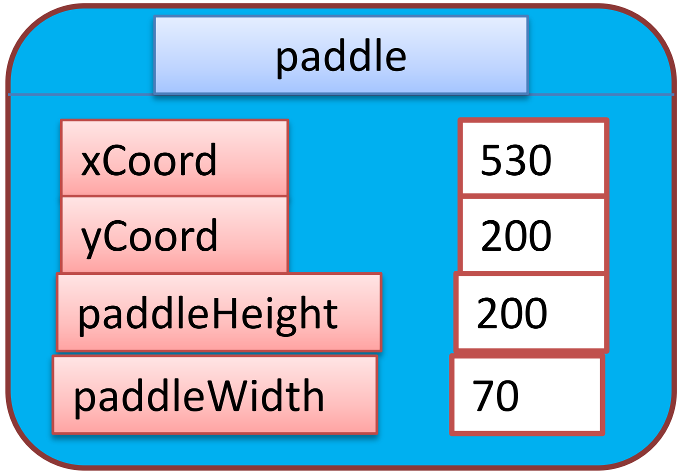
```
float circleDistanceX = abs(ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);
float circleDistanceY =  abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);
```

‘Pythagoras’ Collision Detection Algorithm
- Ball & Paddle closer





ball	
xCoord	450
yCoord	300
diameter	100



paddle	
xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

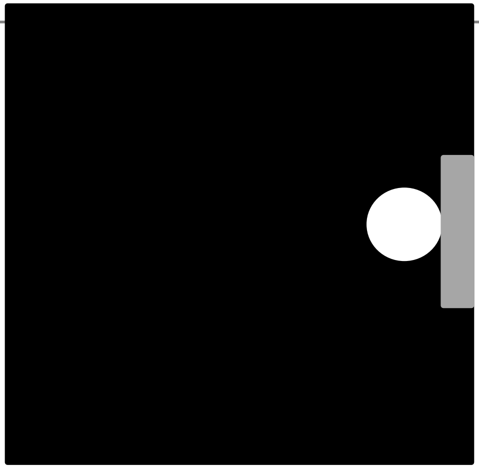
circleDistanceX = 115
circleDistanceY = 0

If (115 > (35 + 50))
→ returns from method with a **false**

i.e. ball and paddle have not made contact.

```
if (circleDistanceX > (paddle.getPaddleWidth()/2  + ball.getDiameter()/2)) { return false; }
if (circleDistanceY > (paddle.getPaddleHeight()/2 + ball.getDiameter()/2)) { return false; }
```

‘Pythagoras’ Collision Detection Algorithm
- Ball & Paddle **overlapping**



circleDistanceX = abs(481 – 530 – 35) = 84
circleDistanceY = abs(300 - 200 - 100) = 0

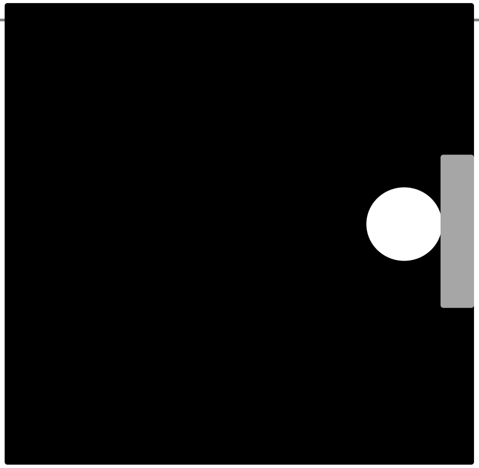
ball	
xCoord	481
yCoord	300
diameter	100

paddle	
xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

```
float circleDistanceX = abs(ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);
float circleDistanceY =  abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);
```

‘Pythagoras’ Collision Detection Algorithm

- Ball & Paddle **overlapping**



circleDistanceX = 84
circleDistanceY = 0

(1) if (84 > (35 + 50)) → boolean condition is false
(2) if (0 > (100 + 50)) → boolean condition is false
(3) if (84 <= (35)) → boolean condition is false
(4) If (0 <= 100)) → returns **true**

ball	
xCoord	480
yCoord	300
diameter	100

paddle	
xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

```
(1)  if (circleDistanceX > (paddle.getPaddleWidth()/2  + ball.getDiameter()/2)) { return false; }
(2)  if (circleDistanceY > (paddle.getPaddleHeight()/2 + ball.getDiameter()/2)) { return false; }
(3)  if (circleDistanceX <= (paddle.getPaddleWidth()/2))  {  return true;  }
(4)  if (circleDistanceY <= (paddle.getPaddleHeight()/2)) {   return true;  }
```

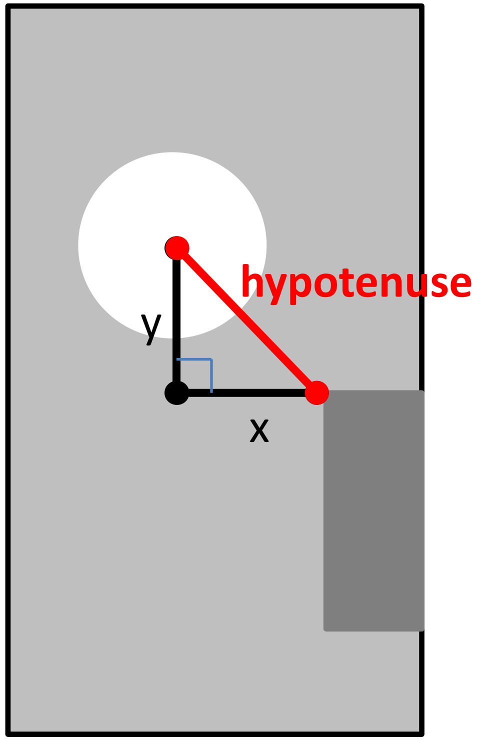
2) COLLISIONS - CORNERS

‘Pythagoras’ Collision Detection Algorithm

We will now look at the code when the ball hits a corner…

```
boolean hitPaddle (Paddle paddle, Ball ball)
{
  // code for ball and paddle overlapping straight on.
  // …

  // Code for ball hitting the corner of the paddle.
  float cornerDistance =
                        pow(circleDistanceX - paddle.getPaddleWidth()/2,  2) +
                        pow(circleDistanceY - paddle.getPaddleHeight()/2, 2);

  if (cornerDistance <= pow(ball.getDiameter()/2, 2)){
     return true;
  }
  else{
     return false;
  }
```

Pythagoras Theorem



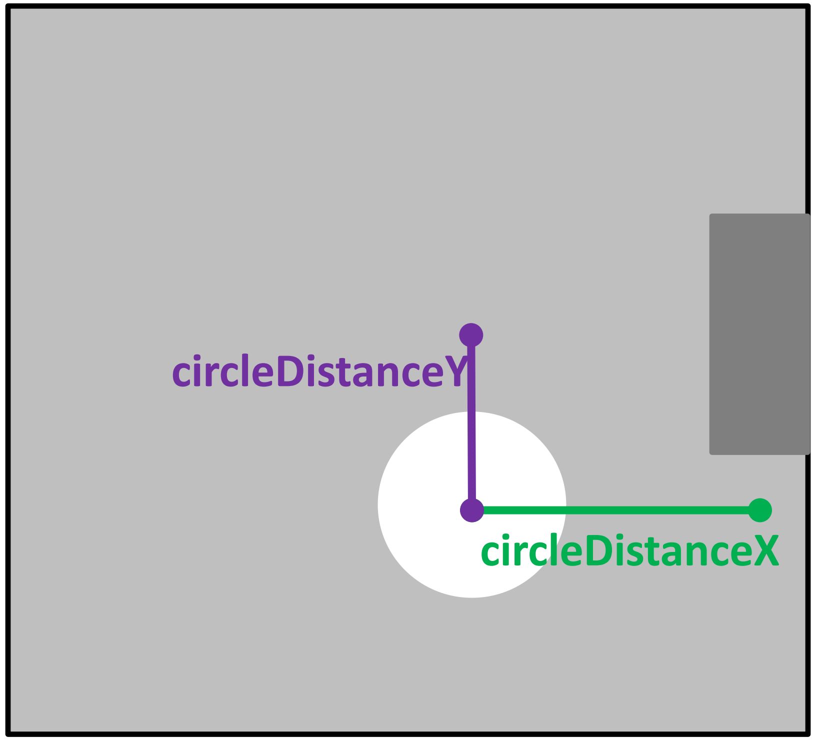
Pythagoras theorem:

The square of the **hypotenuse**
(the side opposite the right angle)

is equal to the sum of the squares
of the other two sides
(in this case x and y).

$$\text{hypotenuse}^2 = x^2 + y^2$$

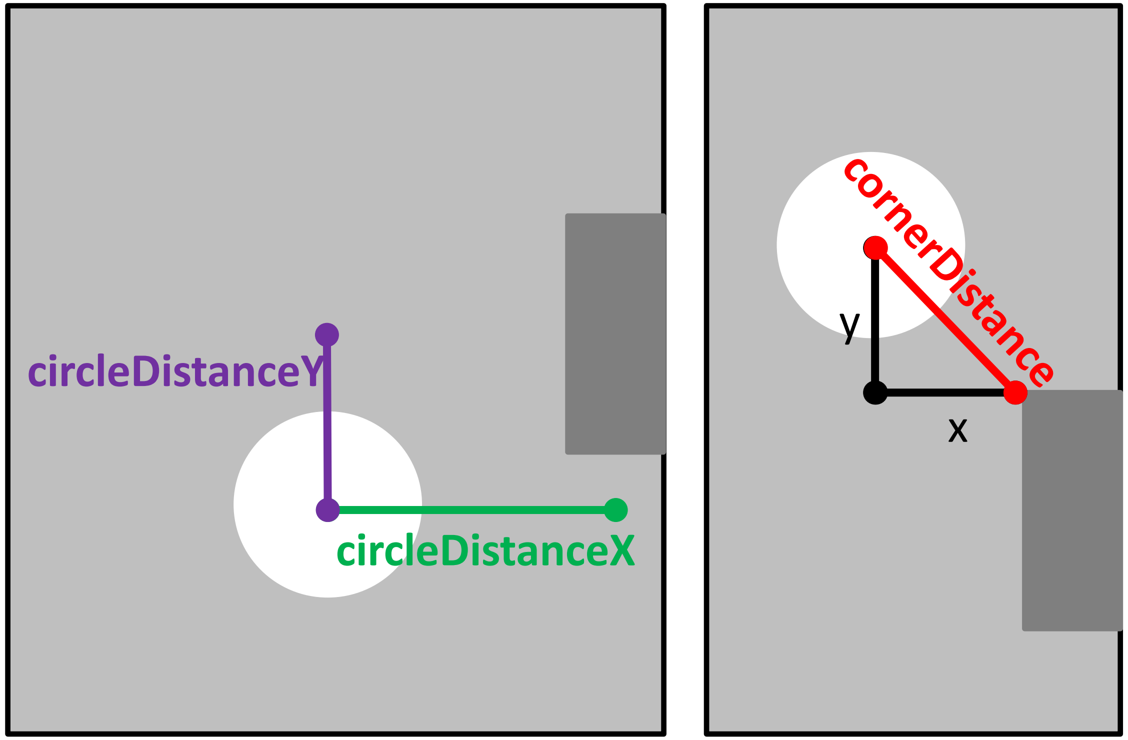
‘Pythagoras’ Collision Detection Algorithm



As before we work out the distances

```
float circleDistanceX
            = abs(ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);

float circleDistanceY
            =  abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);
```

‘Pythagoras’ Collision Detection Algorithm



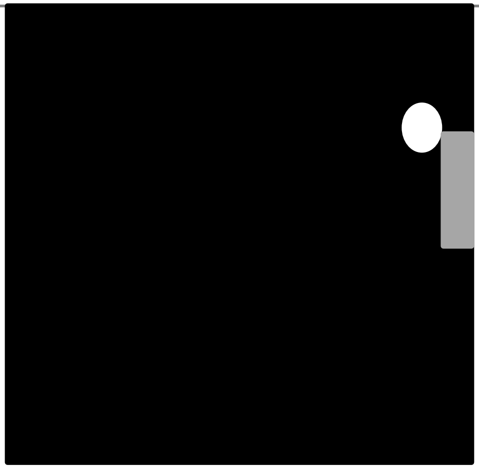
cornerDistance

is the square of the distance from the centre of the circle to the corner of the paddle.

```
float cornerDistance
                = pow (circleDistanceX - paddle.getPaddleWidth()/2,  2) +
                  pow (circleDistanceY - paddle.getPaddleHeight()/2, 2);
```

pow (num, toThePowerOf)   e.g. 5 squared = **pow** (5,2) = 25

‘Pythagoras’ Collision Detection Algorithm - Ball hits the Paddle corner

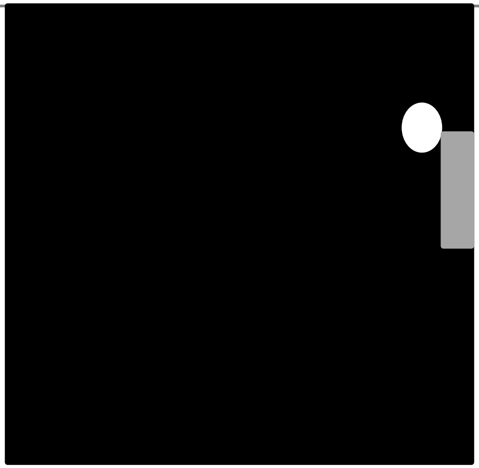


ball	
xCoord	575
yCoord	194
diameter	20

paddle	
xCoord	580
yCoord	200
paddleHeight	100
paddleWidth	20

```
float circleDistanceX
                575                         - 580                       - 20 / 2
          = abs (ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);
          = 15


float circleDistanceY
                194                         - 200                       - 100 / 2
          =  abs (ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);
          = 56

float cornerDistance
                15                                - 20/2
          = pow (circleDistanceX - paddle.getPaddleWidth()/2,  2) +
                56                               - 100/2
            pow (circleDistanceY - paddle.getPaddleHeight()/2, 2);

          = pow (5,2) + pow(6,2)  = 25 + 36 = 61
```

‘Pythagoras’ Collision Detection Algorithm
- Ball hits the Paddle **corner**



ball	
xCoord	575
yCoord	194
diameter	20

paddle	
xCoord	580
yCoord	200
paddleHeight	100
paddleWidth	20

```
              61            pow (                    20/2 , 2)
if (cornerDistance  <=  pow (ball.getDiameter()/2, 2)){
              61        <= 100
    return true;
}
else{
    return false;
}
```

hitPaddle (paddle, ball) method

```
boolean hitPaddle (Paddle paddle, Ball ball)
{
// 1. Work out circleDistanceX and circleDistanceY
   float circleDistanceX = abs(ball.getXCoord() - paddle.getXCoord() - paddle.getPaddleWidth()/2);
   float circleDistanceY = abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);

// 2. Four straight on tests
   if (circleDistanceX > (paddle.getPaddleWidth()/2  + ball.getDiameter()/2)) { return false; }
   if (circleDistanceY > (paddle.getPaddleHeight()/2 + ball.getDiameter()/2)) { return false; }

   if (circleDistanceX <= (paddle.getPaddleWidth()/2))  {  return true;   }
   if (circleDistanceY <= (paddle.getPaddleHeight()/2)) {   return true;  }

// 3. Corner calculation & test
   float cornerDistance = pow(circleDistanceX - paddle.getPaddleWidth()/2,  2) +
                          pow(circleDistanceY - paddle.getPaddleHeight()/2, 2);

   if (cornerDistance <= pow(ball.getDiameter()/2, 2))
     return true;
   else
     return false;
}
```

hitPaddle (paddle, ball) method

- In the **draw()** method,
  the call to **hitPaddle**(ball, paddle) method
  has **no changes** to it i.e. :

```
  //If the player still has a life left in the current game,
  //draw the ball at its new location and check for a collision with the paddle
  if (livesLost < maxLivesPerGame){
    ball.display();
    //if ball and paddle are overlapping, Set variable to true, false if not
    boolean collision = hitPaddle(paddle, ball);
    if (collision == true){
      ball.hit();       //the ball is hit i.e. reverses direction.
      score++;           //increase the score in the current game by 1, if the player hit the
ball.
    }
  }
```

Questions?

References

- Reas, C. & Fry, B. (2014) Processing – A Programming Handbook for Visual Designers and Artists, 2nd Edition, MIT Press, London.