Game of Pong V9.0

Using Pythagoras Theorem for Collision Detection

Produced by: Mairead Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Simple Collision Detection Algorithm (introduced in PongGameV3_0)

Method signature:

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

Algorithm:

- Measure the magnitude of the gap between the paddle and the ball.
- If the ball is too far away from the Paddle on the X axis to have a collision → return false
- If the ball is too far away from the Paddle on the Y axis to have a collision → false
- Otherwise → return true.

Collision Detection Algorithm using Pythagoras Theorem (PongGameV9_0)

- Method signature:

  boolean hitPaddle(Paddle paddle, Ball ball)

- Two collision approaches:
  - The ball overlaps the paddle straight on, returns true.
  - The ball overlaps the corner of the paddle, returns true.
- If the ball does not overlap the paddle, false is returned.

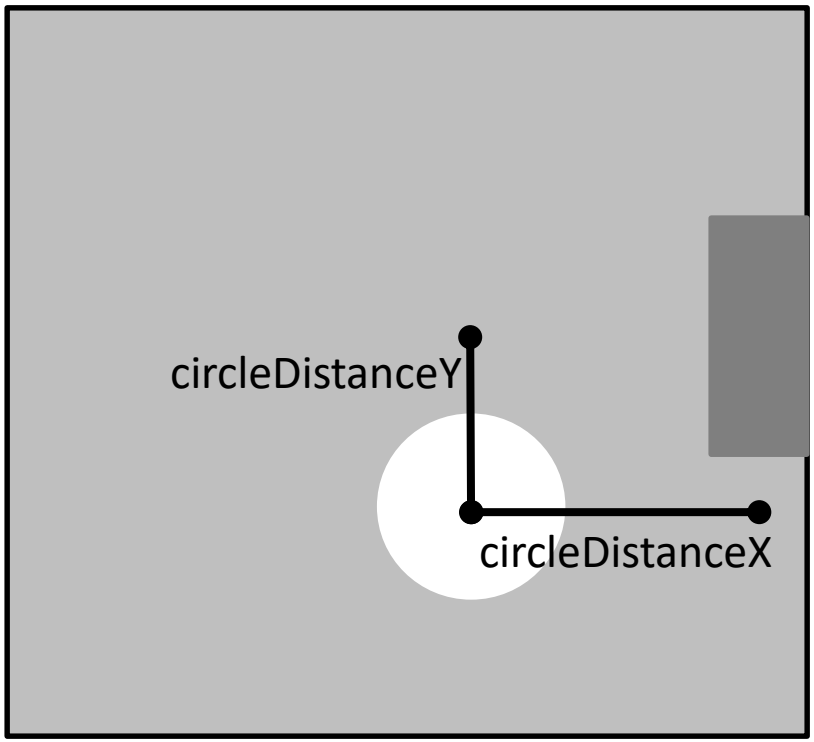
Revised Collision Detection Algorithm

```
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…
}
```

We will now look at the code when the ball overlaps straight on…

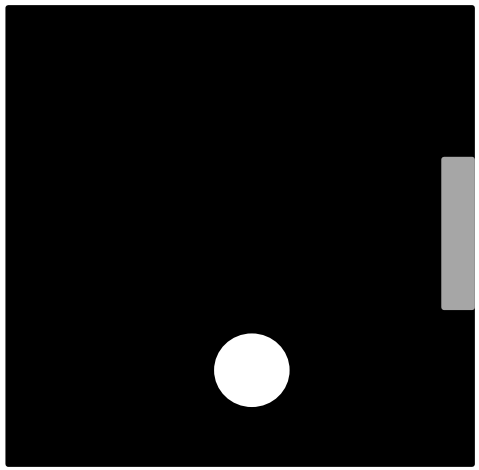
Revised Collision Detection Algorithm




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

Revised Collision Detection Algorithm

When Ball and Paddle **don't** overlap



circleDistanceX = abs(300 – 530 – 35) = 265
circleDistanceY = abs(450 - 200 - 100) = 150

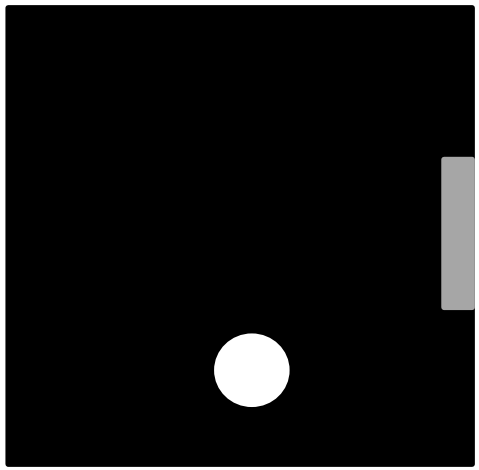
ball	
xCoord	300
yCoord	450
diameter	100

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);
```

Revised Collision Detection Algorithm

When Ball and Paddle **don't** overlap



circleDistanceX = 265
circleDistanceY = 150

If (265 > (35 + 50)) → returns from method with a **false** i.e. ball and paddle have not made contact

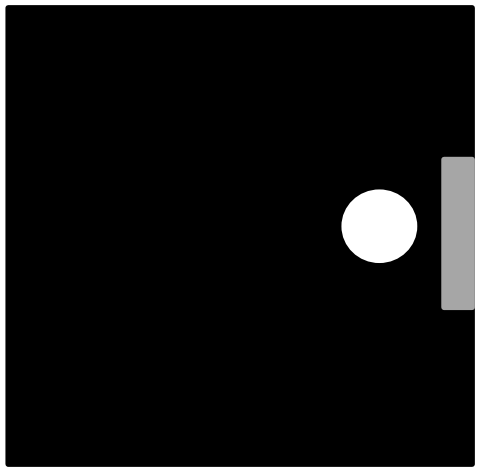
ball	
xCoord	300
yCoord	450
diameter	100

paddle	
xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

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

Revised Collision Detection Algorithm

When Ball and Paddle **are closer**



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

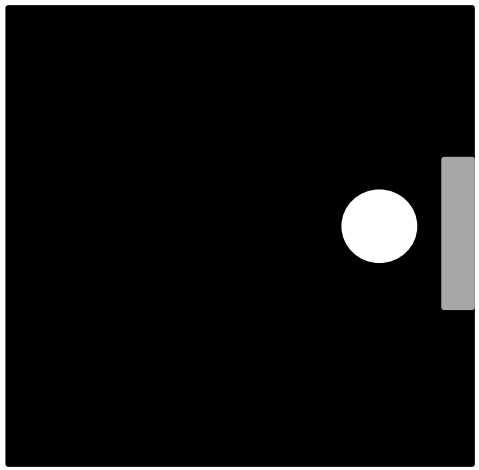
ball	
xCoord	450
yCoord	300
diameter	100

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);
```

Revised Collision Detection Algorithm

When Ball and Paddle **are closer**



circleDistanceX = 115
circleDistanceY = 0

If (115 > (35 + 50)) → returns from method with a **false** i.e. ball and paddle have not made contact.

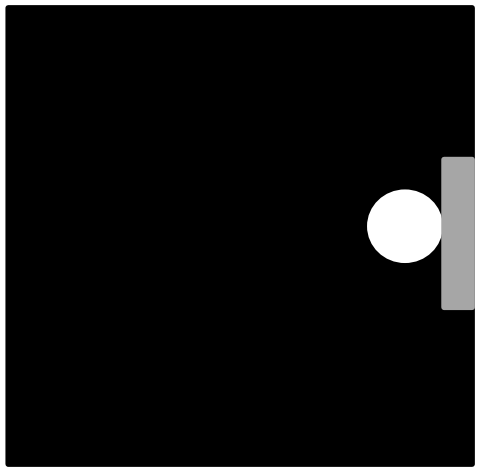
ball	
xCoord	450
yCoord	300
diameter	100

paddle	
xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

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

Revised Collision Detection Algorithm

When Ball and Paddle **overlap**



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

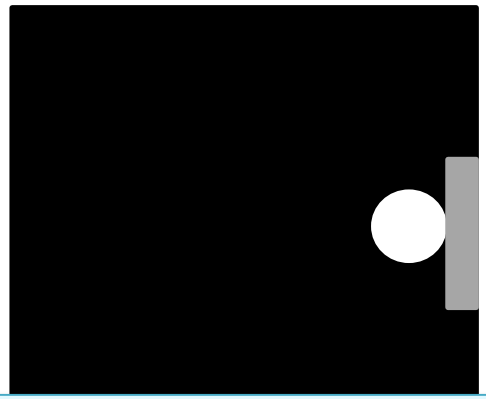
ball	
xCoord	481
yCoord	300
diameter	100

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);
```

Revised Collision Detection Algorithm

When Ball and Paddle **overlap**



circleDistanceX = 84
circleDistanceY = 0

ball	
xCoord	480
yCoord	300
diameter	100

paddle	
xCoord	530
yCoord	200
paddleHeight	200
paddleWidth	70

(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

```
(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;  }
```

Revised Collision Detection Algorithm

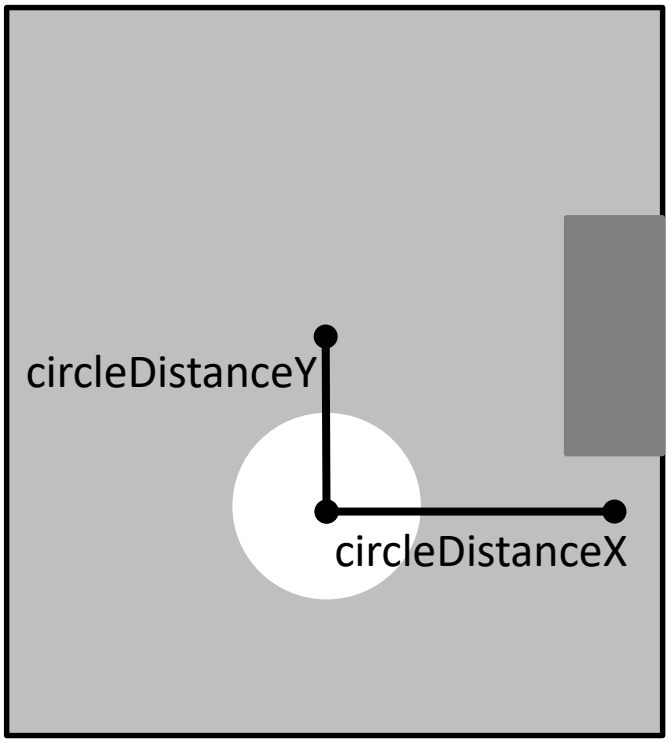
We will now look at the code when the ball hits a corner…

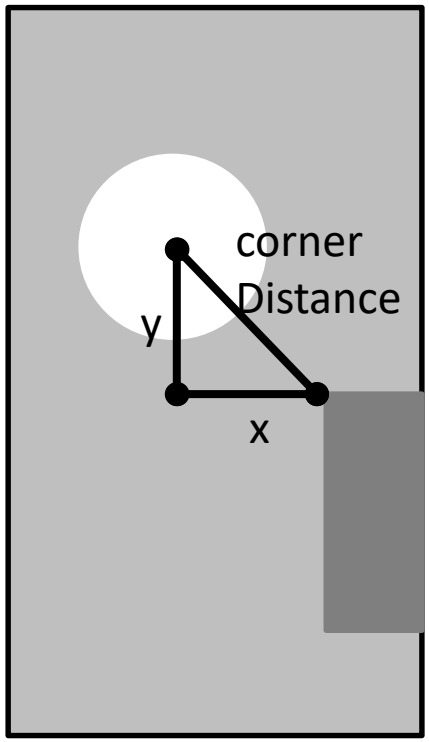
```
boolean hitPaddle(Paddle paddle, Ball ball)
{
  // code for ball and paddle overlapping straight on.
  // …

  // 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;
  }
```

Revised Collision Detection Algorithm





Pythagoras theorem:

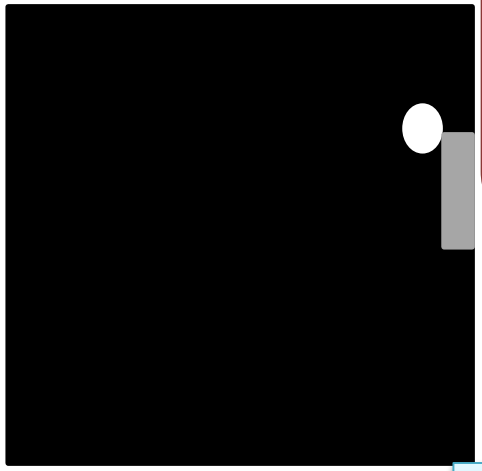
The square of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the other two sides.

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

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

Revised Collision Detection Algorithm

When Ball hits the Paddle **corner**



ball	
xCoord	575
yCoord	194
diameter	20

paddle	
xCoord	580
yCoord	200
paddleHeight	100
paddleWidth	20

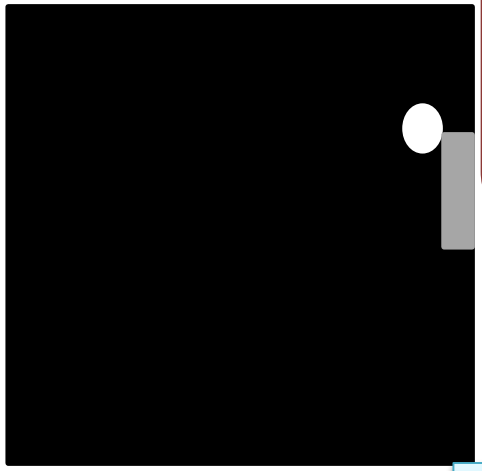
circleDistanceX = abs(575 – 580 – 10) = 15
circleDistanceY = abs(194 - 200 - 50) = 56

cornerDistance = pow ((15 – 10), 2) + pow ((56 – 50), 2)
= pow(5,2) + pow(6,2) = 25 + 36 = 61

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

Revised Collision Detection Algorithm

When Ball hits the Paddle **corner**



ball	
xCoord	575
yCoord	194
diameter	20

paddle	
xCoord	580
yCoord	200
paddleHeight	100
paddleWidth	20

circleDistanceX = abs(575 – 580 – 10) = 15
circleDistanceY = abs(194 - 200 - 50) = 56

cornerDistance = pow ((15 – 10), 2) + pow ((56 – 50), 2)
= pow(5,2) + pow(6,2) = 25 + 36 = 61

if (61 <= 100) → returns true

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

hitPaddle(paddle, ball) method

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

  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;  }

  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() class, the call to hit(ball, paddle) method has no changes to it e.g. :

```
//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();
  //Set variable to true if ball and paddle are overlapping, 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.

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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