Game of Pong

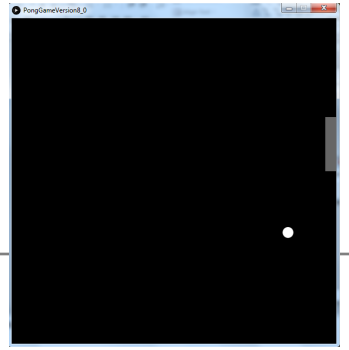
V3

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 V3.0

Classes in the PongGameV3.0

PongGame
ball *paddle*
setup() ***draw()*** ***hitPaddle* (paddle, ball)**

Paddle
Xcoord *yCoord* *paddleHeight* *paddleWidth*
Paddle(int, int) *update()* *display()* *getXCoord()* *getYCoord()* *getPaddleWidth()* *getPaddleHeight()* *setPaddleWidth(int)* *setPaddleHeight(int)*

Ball
xCoord *yCoord* *diameter* *speedX* *speedY*
Ball(float) *update()* *display()* *hit()* *getXCoord()* *getYCoord()* *getDiameter()* *setDiameter(float)* *resetBall()*

Ball and Paddle classes → no change

In PongGame, **draw()** is updated to call the new **hitPaddle()** method.

hitPaddle uses a *collision detection* algorithm

- if the paddle and ball are touching
  - returns true
- false otherwise.

Collision Detection Algorithm used in **hitPaddle**

Method signature:

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

Algorithm:

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

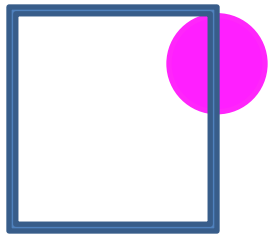
2) If the ball is too far away from the Paddle on the **X axis** to have a collision
→ return false

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

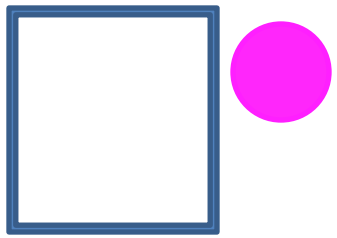
4) Otherwise
→ return true.

Recap – Drawing Modes: **ellipse**

- The default ellipse mode is **CENTER**
  - This means x & y positions for ellipse() specify the **center** of the ellipse
  - At the max width of the window, half the ellipse is seen

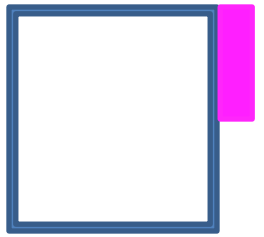


  - If we specify an x value > width + radius of the circle the circle has left the screen

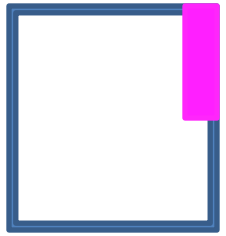


Recap – Drawing Modes: **rect**

- The default rect mode is **CORNER**
  - This means x & y positions for rect() specify the **top left CORNER** of the rectangle
  - At the max width of the window, the rectangle would be invisible



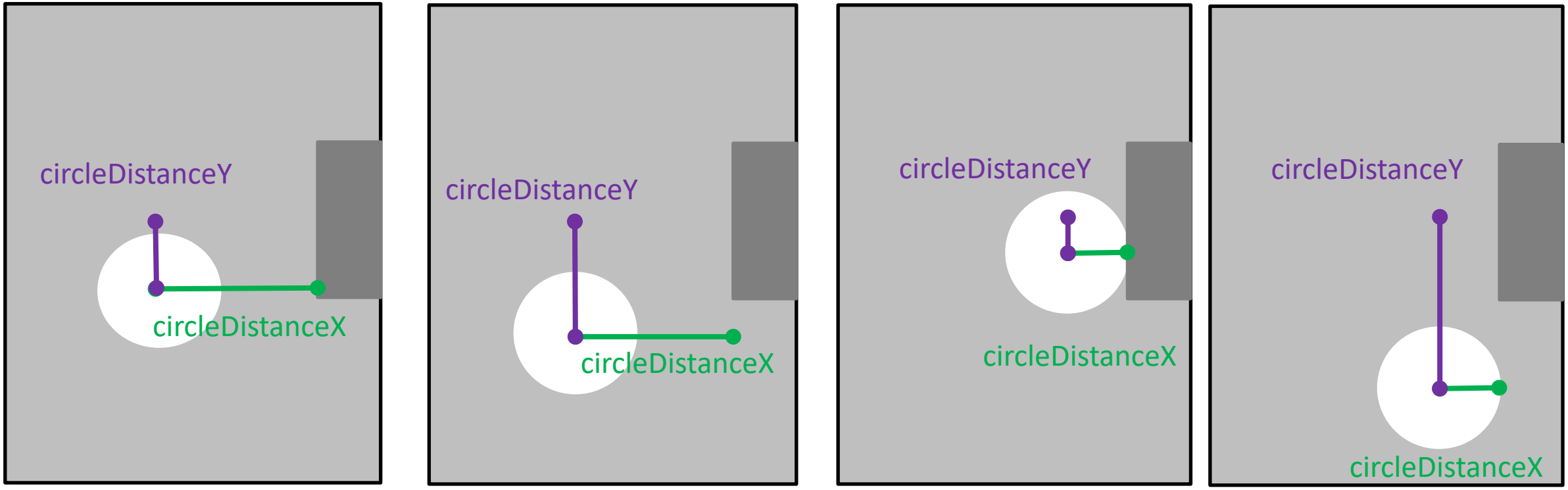
  - If we specify an x value which is the width of the screen – width of the rectangle it will be seen



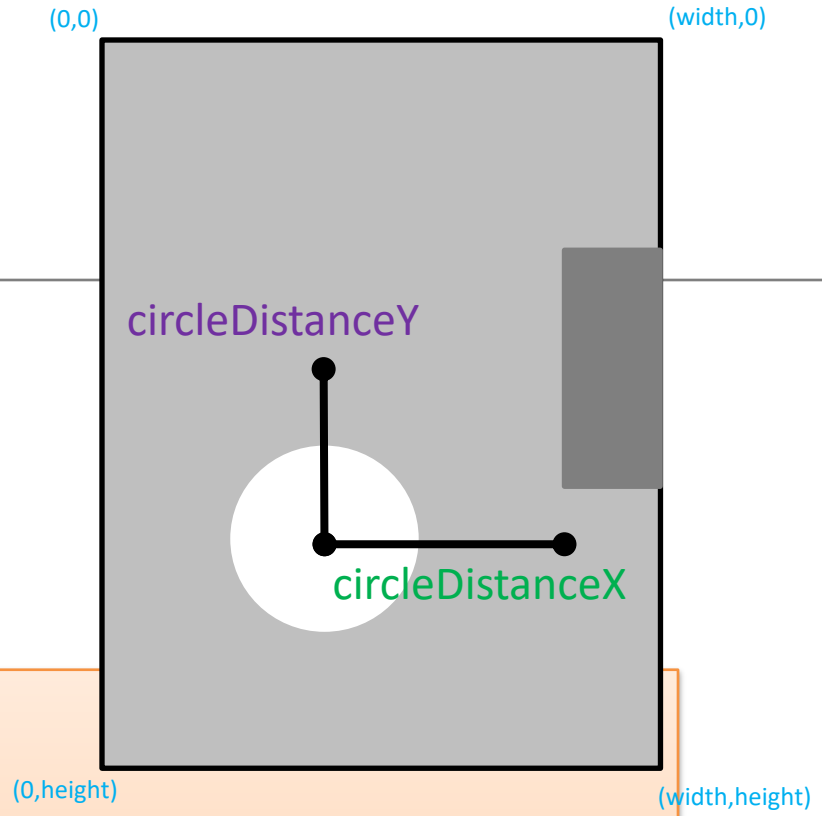
1) Measuring size of the **gap between the paddle and ball**.

We need to first calculate **how far** away the ball is from the paddle on both the **x and the y** axis e.g.:

circleDistanceY = distance from center of circle to **center** of paddle
circleDistanceX = distance from center of circle to **left** edge of paddle



1) Measuring size of the **gap between the paddle and ball.**




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

//These variables measure the magnitude of the gap between the paddle and ball.
 float circleDistanceX
          = abs(ball.getXCoord() - paddle.getXCoord());
 float circleDistanceY
          = abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);

}
```

Questions

Q1: What is the circleDistanceX
if the circle is at (200,400)
And the paddle is at (380,100)
with a paddle height of 100?

Q2: What is the circleDistanceY
if the circle is at (200,400)
And the paddle is at (380,100)
with a paddle height of 100?

Collision Detection Algorithm

Method signature:

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

Algorithm:

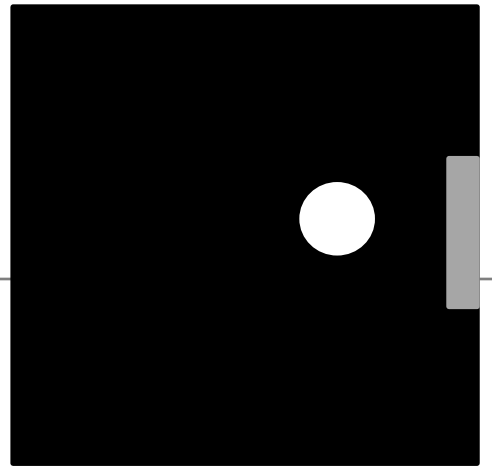
1) Measure the size of the gap between the paddle and the ball.

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.

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



```
//The Ball is too far away from the Paddle on the X axis
// to have a collision,
// so abandon collision detection

if (circleDistanceX > (ball.getDiameter()/2)) {
   return false;
}
```

If ball is too far away from the Paddle on the **X axis** → return false

Collision Detection Algorithm

Method signature:

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

Algorithm:

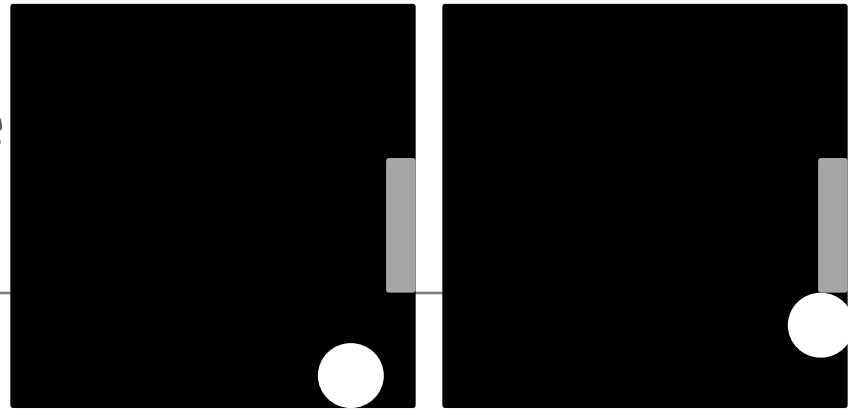
1) Measure the size of the gap between the paddle and the ball.

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.

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



return false return true

```
//The Ball is too far away from the Paddle on the Y axis to have a collision,
//so abandon collision detection

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

If ball is too far away from the Paddle on the **Y axis** → return false

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

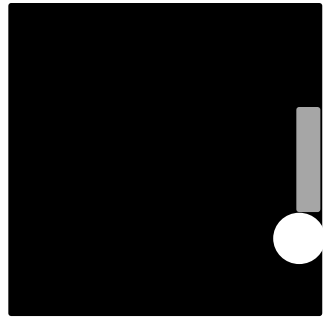
3) If the ball is too far away from the Paddle on the **Y axis** to have a collision
→ return false

4) Otherwise
→ return true.

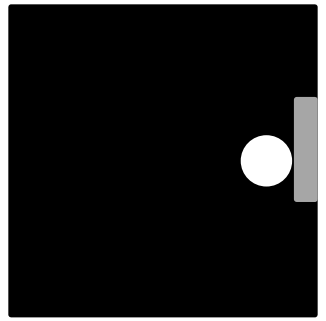
4) Otherwise return true

```
//We have a collision
  return true;
```

We have a collision



return true

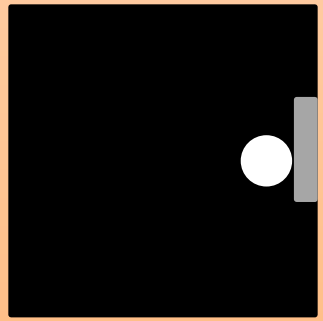


```
boolean hitPaddle (Paddle paddle, Ball ball)
{
1  //These variables measure the magnitude of the gap between the paddle and ball.
   float circleDistanceX
            = abs(ball.getXCoord() - paddle.getXCoord());
   float circleDistanceY
            = abs(ball.getYCoord() - paddle.getYCoord() - paddle.getPaddleHeight()/2);

2  //The Ball is too far away from the Paddle on the X axis to have a collision,
   //so abandon collision detection
   if (circleDistanceX > (ball.getDiameter()/2)) {
      return false;
   }

3  //The Ball is too far away from the Paddle on the Y axis to have a collision,
   //so abandon collision detection
   if (circleDistanceY > (paddle.getPaddleHeight()/2 + ball.getDiameter()/2)) {
       return false;
   }
4  //We have a collision
   return true;
}
```

hitPaddle()



hitPaddle (paddle, ball) method

- Call the **hitPaddle** (paddle,ball) method from the **draw()** method in our main PongGame class.
- Which in turn calls **ball.hit()** if true

```
void draw (){
  background(0);      //Clear the background
  paddle.update();    //Update the paddle location in line with the cursor
  paddle.display();   //Draw the paddle in this new location
  ball.update();      // update the ball position.
  ball.display();     //Draw the ball at its new location

  //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. reverse direction.
  }
}
```

Questions?

References

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