Game of Pong

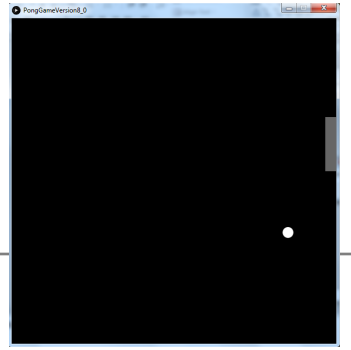
V2 Starting development

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)

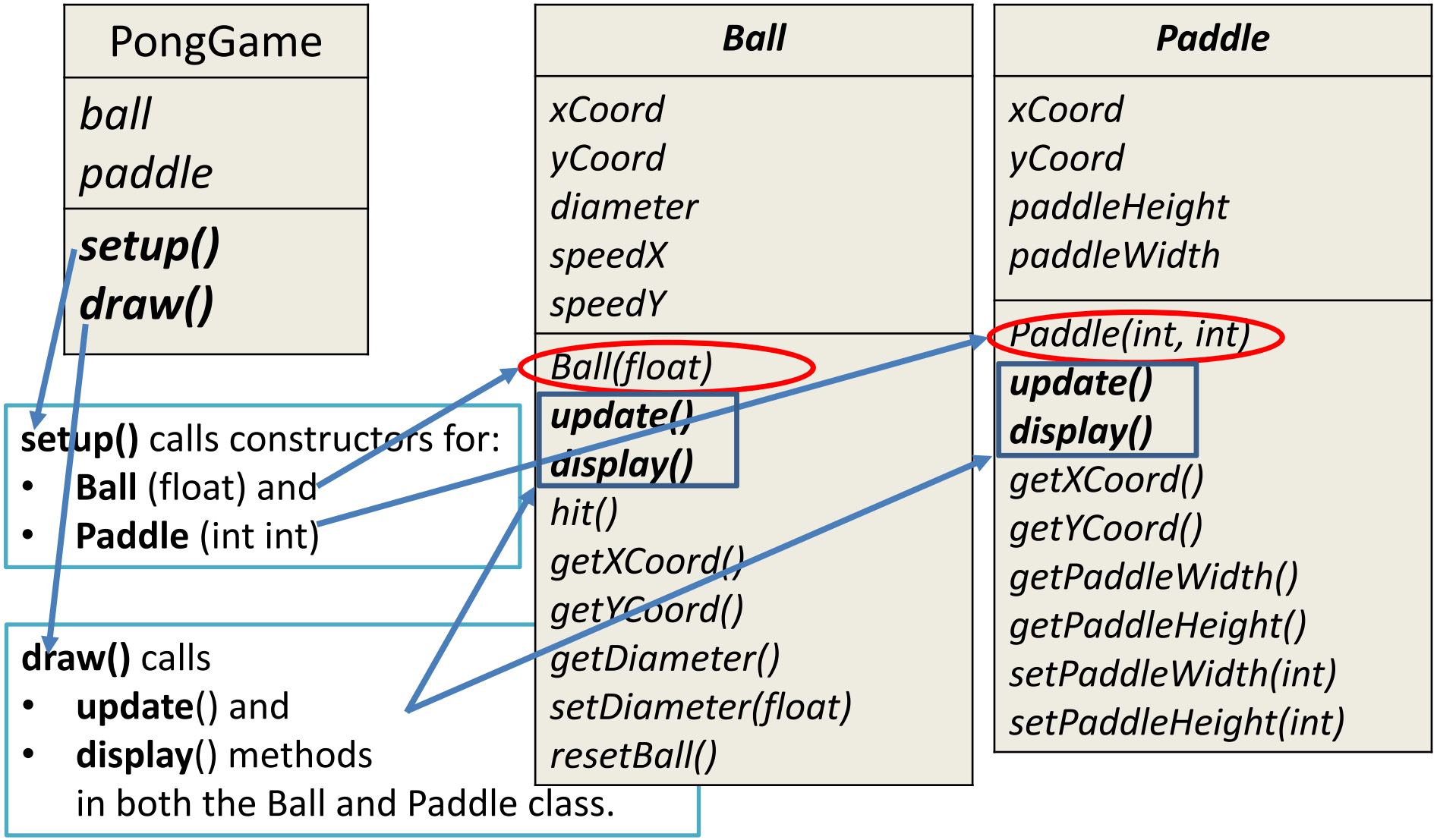
v7 - Player class using arrays (with **statistics** (Tournament Over - highest, lowest, average score))

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

v9 - alternative algorithm using **Pythagoras Theorem**

Demo of
Pong Game V2.0

Classes in the PongGameV2.0

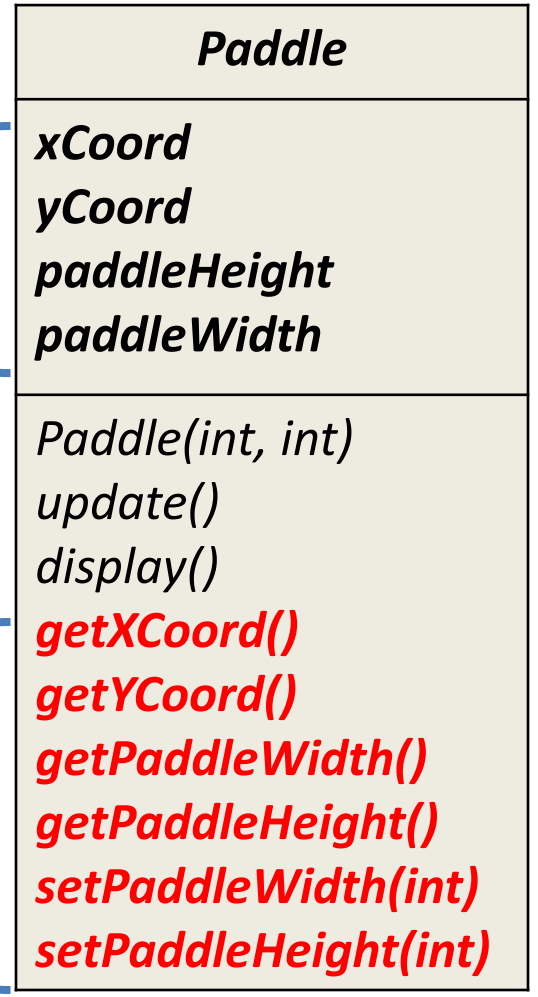


Paddle Class – instance **fields**

```
private int xCoord;          // X coordinate of the paddle
private int yCoord;          // Y coordinate of the paddle
private int paddleWidth;     // width of the paddle
private int paddleHeight;    // height of the paddle
```

Fields – made private

getters and **setters** for the private **fields**



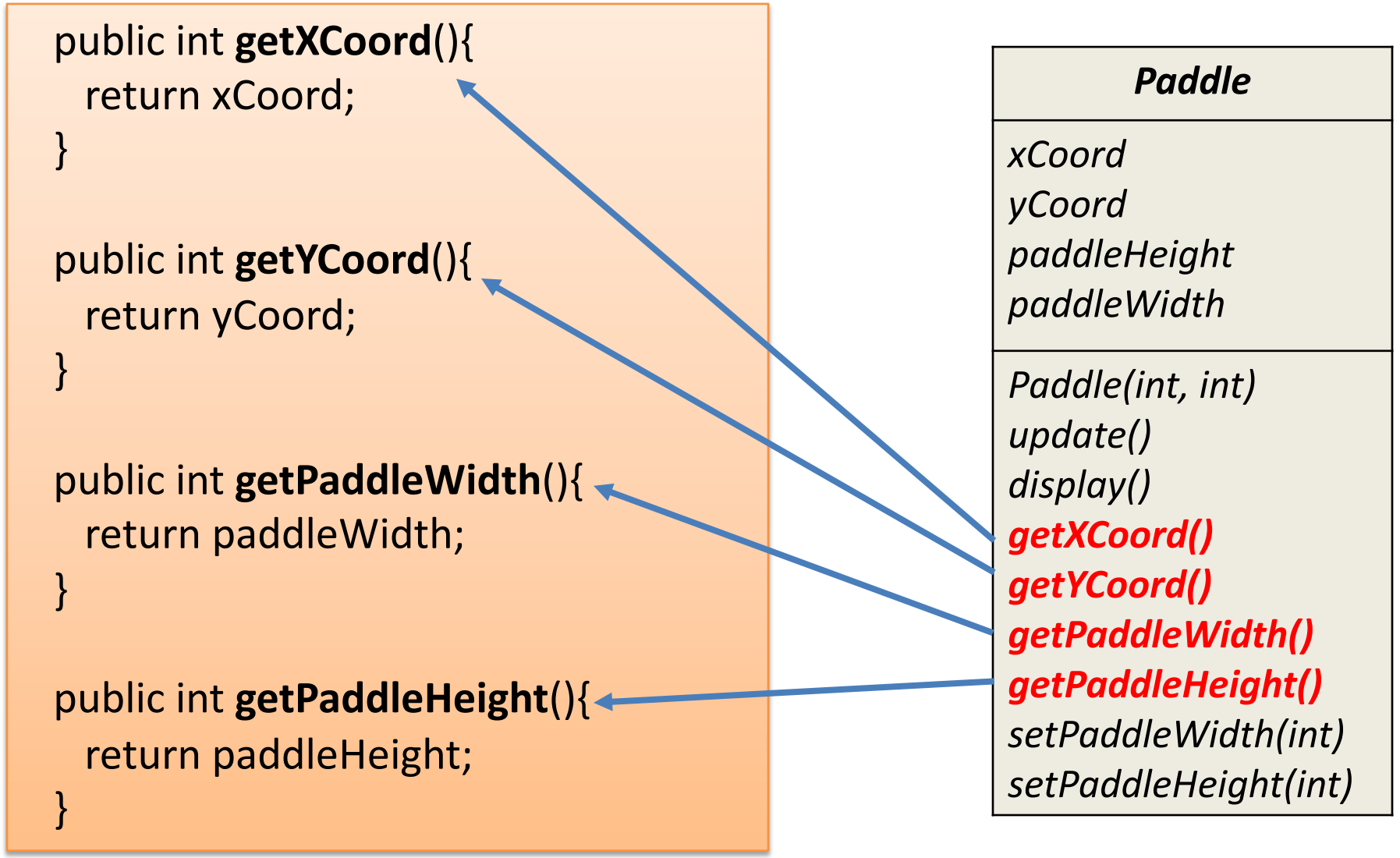
Paddle Class – **get**ters

```
public int getXCoord(){
   return xCoord;
}

public int getYCoord(){
   return yCoord;
}

public int getPaddleWidth(){
   return paddleWidth;
}

public int getPaddleHeight(){
   return paddleHeight;
}
```



Paddle Class – **set**ters

setPaddleWidth(int)

```
public void setPaddleWidth (int paddleWidth){
    //The paddle width must be
    // between 10 and width/2 (inclusive)
    if ((paddleWidth >= 20) && (paddleWidth <= width/2)){
        this.paddleWidth = paddleWidth;
    }
    else{
    // If an invalid width is passed as a parameter, a default
    // width of 20 is imposed.  With this animation, if we do
    // not supply a default value for the width, a paddle
    //may not be drawn on the display window.  Important
    // note: it is not always appropriate to provide a default
    // value at setter level; this will depend on your
    //design.
        this.paddleWidth = 20;
    }
}
```

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

Paddle Class – **set**ters

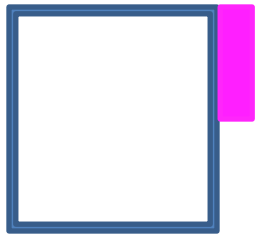
setPaddleHeight(int)

```
public void setPaddleHeight (int paddleHeight){
    // The paddle height must be
    // between 50 and height/2 (inclusive)
    if ((paddleHeight >= 50) && (paddleHeight <= height/2)){
        this.paddleHeight = paddleHeight;
    }
    else{
    // If an invalid height is passed as a parameter, a default
    // height of 50 is imposed.  With this animation, if we do
    // not supply a default value for the height, a paddle
    // may not be drawn on the display window.  Important
    // note: it is not always appropriate to provide a default
    // value at setter level; this will depend on your design.
        this.paddleHeight = 50;
    }
  }
```

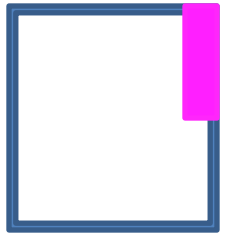
Paddle
xCoord *yCoord* *paddleHeight* *paddleWidth*
Paddle(int, int) *update()* *display()* *getXCoord()* *getYCoord()* *getPaddleWidth()* *getPaddleHeight()* *setPaddleWidth(int)* ***setPaddleHeight(int)***

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



Paddle constructor

```
public Paddle (int paddleWidth, int paddleHeight)
{
    setPaddleWidth (paddleWidth);
    setPaddleHeight (paddleHeight);

    // the xCoordinate variable is set here and it stays
    // this value for duration of the program.
    xCoord = width - this.paddleWidth;

    // the yCoordinate variable is set here and changes
    // later in the program as the mouse moves on the
    // vertical plane.
    yCoord = height/2;
 }
```

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

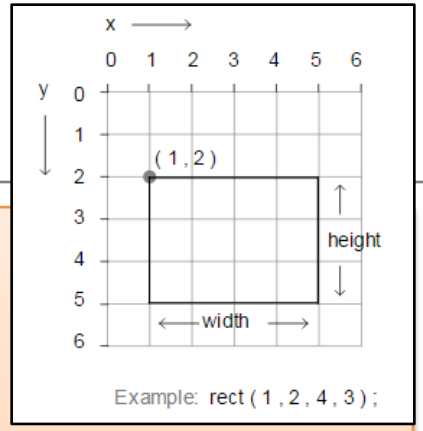
display() method

```
public void display() {
  fill(102);
  noStroke();
  rect(xCoord, yCoord, paddleWidth, paddleHeight);
}
```

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

Draws a gray paddle,
with no outline on the display window.

update() method




```
public void update()
{
   yCoord = mouseY - paddleHeight/2;

   //Reset yCoord if it's outside the window coordinates.
   if (yCoord < 0){
       yCoord = 0;
    }
    if (yCoord > (height - paddleHeight)){
       yCoord = height - paddleHeight;
    }
}
```

changes the vertical position of the paddle in line with the cursor.

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

PongGame V2.0

```
Ball ball;
Paddle paddle;

void setup(){
  size(600,600);
  noCursor();
  //setting up ball and paddle with hard-coded sizes.
  ball = new Ball(20.0);
  paddle = new Paddle(20,100);
}

void draw(){
  background(0);
  //Update the paddle location in line with the cursor
  paddle.update();
  paddle.display();
  //Update the ball position and display it.
  ball.update();
  ball.display();
}
```

PongGame
Ball *paddle*
setup() ***draw()***

Create Ball & Paddle objects.

Call their update() & display() methods in draw()

Questions?

BREAK

References

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