

Game of Pong

V3, V4 and V5

Produced Ms. Mairead Meagher
by: Dr. Siobhán Drohan

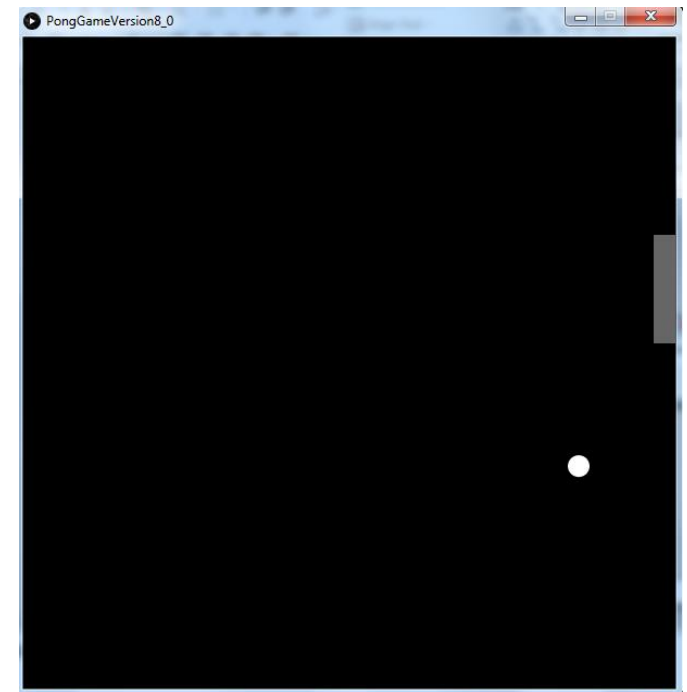


Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topics list - PONG

- Overview of PongGame
- Developing PongGame
 - 9 versions (iterations) described with 4 sets of slides:
 - Set 1
 - V1.0 (Ball class)
 - V2.0 (Paddle class)
 - Set 2
 - V3.0 (**Collision detection**)
 - V4.0 (Lives lost, lives per game, score)
 - V5.0 (Tournament functionality)
 - Set 3
 - V6.0 (Player class – array, no statistics)
 - V7.0 (Player class – array, with statistics)
 - V8.0 (JOptionPane for I/O)
 - Set 4
 - V9.0 (Advanced Collision Detection)



Idea is based on Reas and Fry (2014) example

Demo of Pong Game V3.0

Classes in the PongGameV3.0

PongGame	Paddle	Ball
<i>ball</i> <i>paddle</i>	<i>Xcoord</i> <i>yCoord</i> <i>paddleHeight</i> <i>paddleWidth</i>	<i>xCoord</i> <i>yCoord</i> <i>diameter</i> <i>speedX</i> <i>speedY</i>
<i>setup()</i> <i>draw()</i> <i>hitPaddle (paddle, ball)</i>	<i>Paddle(int, int)</i> <i>update()</i> <i>display()</i> <i>getXCoord()</i> <i>getYCoord()</i> <i>getPaddleWidth()</i> <i>getPaddleHeight()</i> <i>setPaddleWidth(int)</i> <i>setPaddleHeight(int)</i>	<i>Ball(float)</i> <i>update()</i> <i>display()</i> <i>hit()</i> <i>getXCoord()</i> <i>getYCoord()</i> <i>getDiameter()</i> <i>setDiameter(float)</i> <i>resetBall()</i>

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

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.

Collision Detection Algorithm

Method signature:

boolean hitPaddle (Paddle paddle, Ball ball)

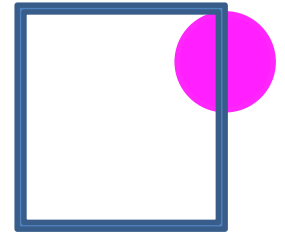
Algorithm:

- 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.

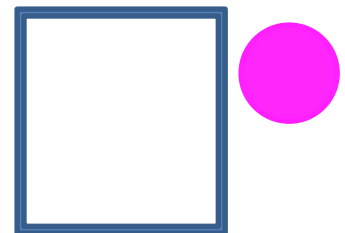
Recap – Drawing Modes: **ellipse**

- 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

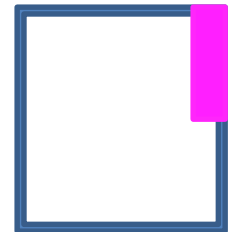
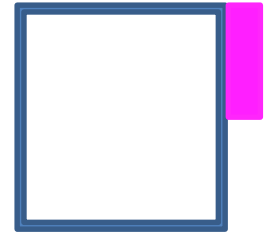


- If we specify an x value $>$ width + radius of the circle
the circle has left the screen



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

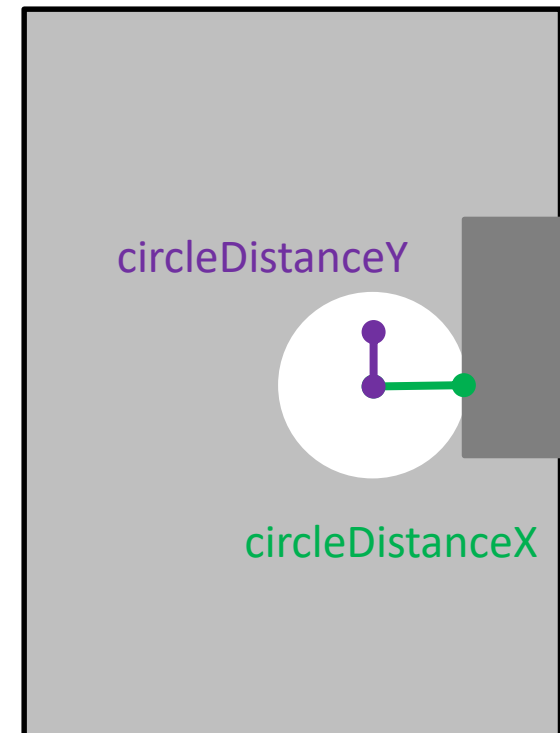
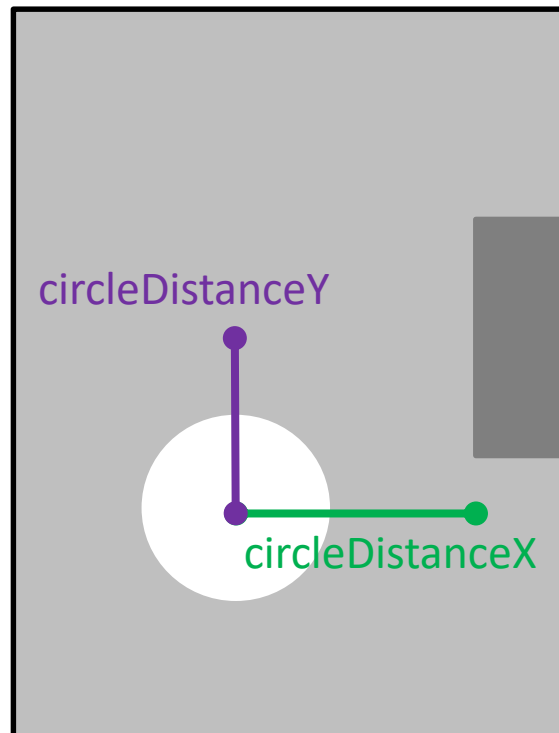
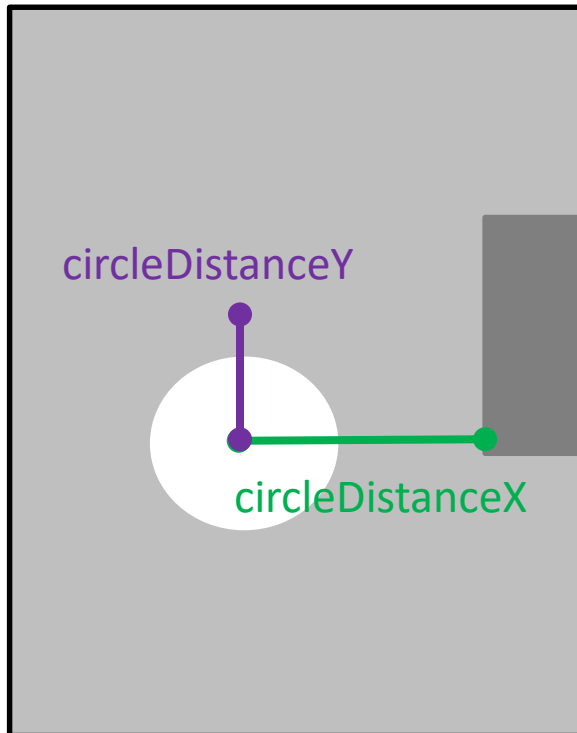


1) Measuring size of the **gap** between the paddle and ball.

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

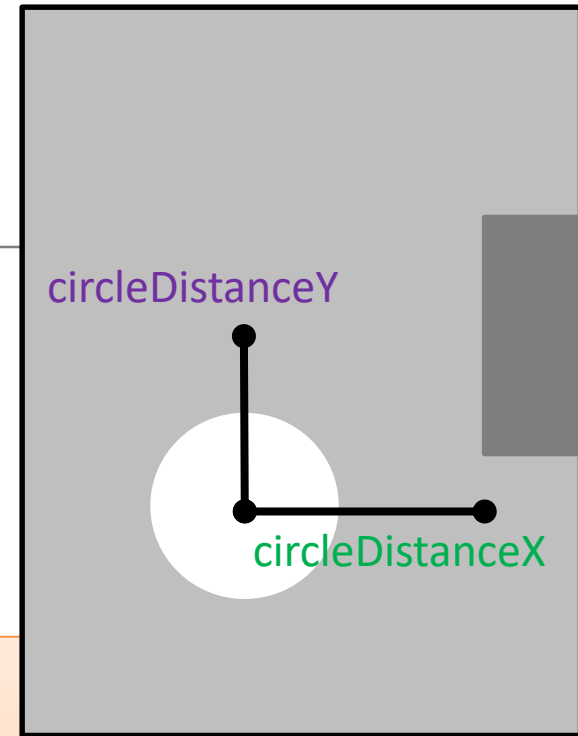
circleDistanceY = distance from center of circle to **center** of paddle

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



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

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



```
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);
```

```
}
```

Collision Detection Algorithm

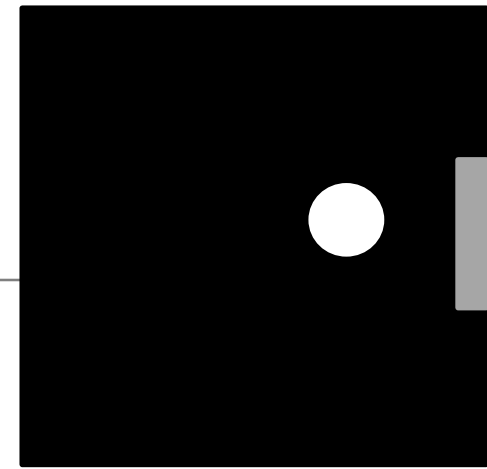
Method signature:

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;  
}
```

Collision Detection Algorithm

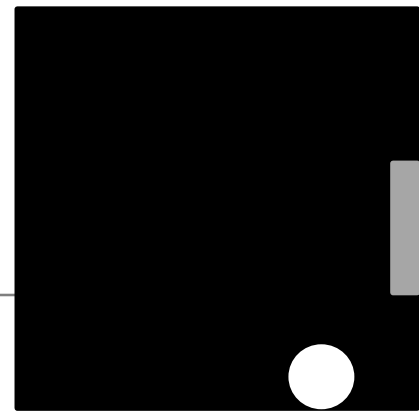
Method signature:

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



```
//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;  
}
```

Collision Detection Algorithm

Method signature:

boolean hitPaddle (Paddle paddle, Ball ball)

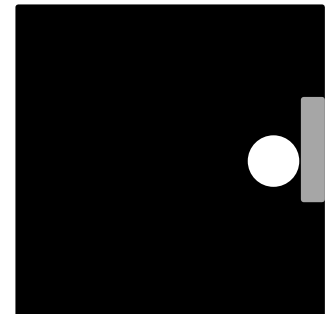
Algorithm:

- 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 false

//We have a collision
return true;

We have a collision



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;

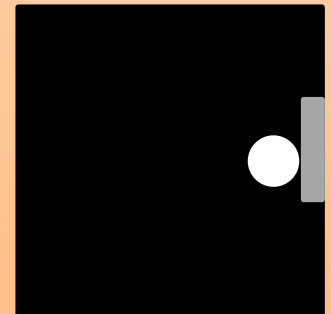
}

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

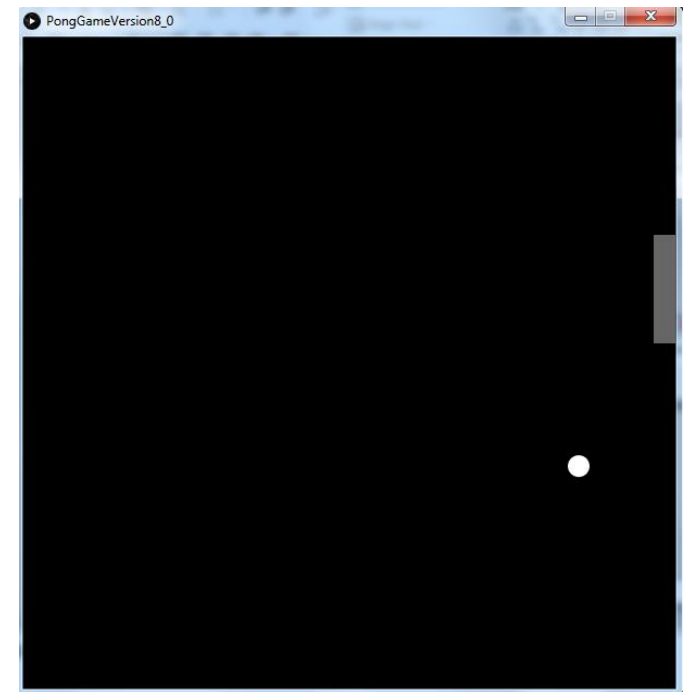
```
void draw (){  
    background(0);    //Clear the background  
    paddle.update();  //Update the paddle location in line with the cursor  
    paddle.display(); //Draw the paddle in this new location  
    ball.update();    // update the ball position.  
    ball.display();   //Draw the ball at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

Topics list - PONG

- Overview of PongGame
- Developing PongGame
 - 9 versions (iterations) described with 4 sets of slides:
 - Set 1
 - V1.0 (Ball class)
 - V2.0 (Paddle class)
 - Set 2
 - V3.0 (Collision detection)
 - V4.0 (**Lives lost, lives per game, score**)
 - V5.0 (Tournament functionality)
 - Set 3
 - V6.0 (Player class – array, no statistics)
 - V7.0 (Player class – array, with statistics)
 - V8.0 (JOptionPane for I/O)
 - Set 4
 - V9.0 (Advanced Collision Detection)



Idea is based on Reas and Fry (2014) example

Demo of Pong Game V4.0

PongGameV4.0

- This version **stores game information**:
 - The number of **lives lost**
 - The **maximum lives** allowed per game
 - The **score** of the game
- Game Over
 - when user loses the number of lives allowed per game.
- Changes
 - None in the Ball and Paddle class
 - All changes in PongGameV4.0 class.

Classes in the PongGameV4.0

PongGame
<i>ball</i> <i>Paddle</i> <i>livesLost</i> <i>score</i> <i>maxLivesPerGame</i>
<i>setup()</i> <i>draw()</i> <i>hitPaddle(paddle, ball)</i>

Paddle
<i>Xcoord</i> <i>yCoord</i> <i>paddleHeight</i> <i>paddleWidth</i>
<i>Paddle(int, int)</i> <i>update()</i> <i>display()</i> <i>getXCoord()</i> <i>getYCoord()</i> <i>getPaddleWidth()</i> <i>getPaddleHeight()</i> <i>setPaddleWidth(int)</i> <i>setPaddleHeight(int)</i>

Ball
<i>xCoord</i> <i>yCoord</i> <i>diameter</i> <i>speedX</i> <i>speedY</i>
<i>Ball(float)</i> <i>update()</i> <i>display()</i> <i>hit()</i> <i>getXCoord()</i> <i>getYCoord()</i> <i>getDiameter()</i> <i>setDiameter(float)</i> <i>resetBall()</i>

PongGameV4.0 class – global fields

//Current game data

```
int livesLost = 0;           //keeps track of number of lives lost in current game
int score = 0;              //high score of the current game
int maxLivesPerGame = 3;    //maximum number of lives that can be lost
                           //before the game ends
```

PongGameV4.0 class – draw()

Version 3.0

```
// Update the ball position.  
ball.update();
```



Version 4.0

```
// Update the ball position. If true is returned, the ball has left the display window  
// i.e. a life is lost  
if (ball.update() == true){  
    livesLost++;  
    println("Lives lost: " + livesLost);  
}
```


PongGameV4.0 class – draw()

Version 3.0

```
//Draw the ball at its new location and check for a collision with the paddle  
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.  
}
```

PongGameV4.0 class – draw()

Version 4.0

```
//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 score in the current game by 1, if the player hit the ball.  
        println("Score: " + score);  
    }  
}  
  
//The player has no lives left so the game ends  
else{  
    println("Game Over!");  
    println("You have lost all of your lives: " + livesLost);  
    println("Your final score is: " + score);  
    exit();  
}
```

```
Lives lost: 1  
Score: 1  
Score: 2  
Score: 3  
Score: 4  
Lives lost: 2  
Lives lost: 3
```

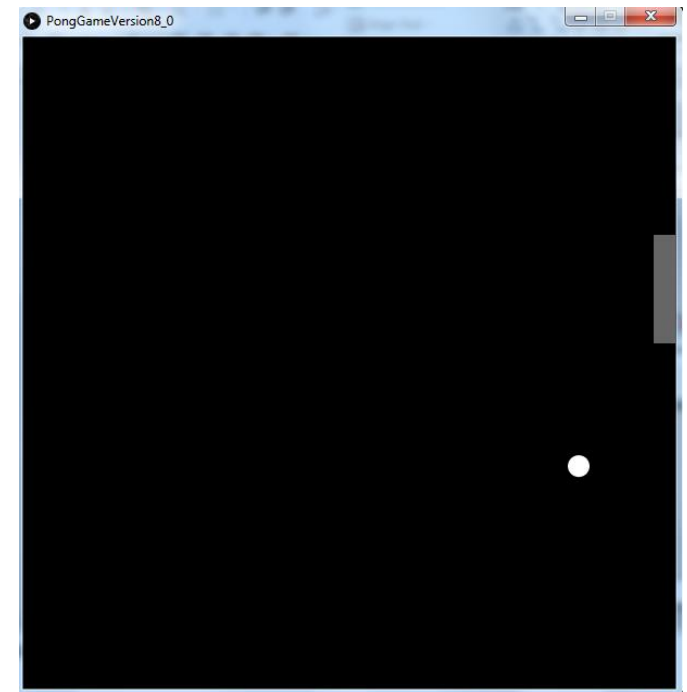
```
Lives lost: 1  
Score: 1  
Score: 2  
Score: 3  
Score: 4  
Lives lost: 2  
Lives lost: 3  
Game Over!  
You have lost all of your lives: 3  
Your final score is: 4
```

PongGameV4.0 – sample output

```
Lives lost: 1  
Score: 1  
Score: 2  
Score: 3  
Score: 4  
Lives lost: 2  
Lives lost: 3  
Game Over!  
You have lost all of your lives: 3  
Your final score is: 4
```

Topics list - PONG

- Overview of PongGame
- Developing PongGame
 - 9 versions (iterations) described with 4 sets of slides:
 - Set 1
 - V1.0 (Ball class)
 - V2.0 (Paddle class)
 - Set 2
 - V3.0 (Collision detection)
 - V4.0 (Lives lost, lives per game, score)
 - V5.0 (**Tournament functionality**)
 - Set 3
 - V6.0 (Player class – array, no statistics)
 - V7.0 (Player class – array, with statistics)
 - V8.0 (JOptionPane for I/O)
 - Set 4
 - V9.0 (Advanced Collision Detection)



Idea is based on Reas and Fry (2014) example

Demo of Pong Game V5.0

PongGameV5.0

- This version **stores tournament information**:
 - The number of **games in a tournament**.
 - The number of **games played** so far.
- If the number of games in the tournament is over, **end the program**.
- Changes
 - None in the Ball and Paddle class
 - All changes in PongGameV5.0 class.

Classes in the PongGameV5.0

PongGame
<i>ball</i> <i>Paddle</i> <i>livesLost</i> <i>score</i> <i>maxLivesPerGame</i> <i>maxNumberOfGames</i> <i>numberOfGamesPlayed</i>
<i>setup()</i> <i>draw()</i> <i>resetGame()</i> <i>tournamentOver()</i> <i>hitPaddle(paddle, ball)</i>

Paddle
<i>Xcoord</i> <i>yCoord</i> <i>paddleHeight</i> <i>paddleWidth</i>
<i>Paddle(int, int)</i> <i>update()</i> <i>display()</i> <i>getXCoord()</i> <i>getYCoord()</i> <i>getPaddleWidth()</i> <i>getPaddleHeight()</i> <i>setPaddleWidth(int)</i> <i>setPaddleHeight(int)</i>

Ball
<i>xCoord</i> <i>yCoord</i> <i>diameter</i> <i>speedX</i> <i>speedY</i>
<i>Ball(float)</i> <i>update()</i> <i>display()</i> <i>hit()</i> <i>getXCoord()</i> <i>getYCoord()</i> <i>getDiameter()</i> <i>setDiameter(float)</i> <i>resetBall()</i>

PongGameV5.0 class – global fields

//Tournament data

```
int maxNumberOfGames = 5;    //maximum number of games in a tournament  
int numberOfGamesPlayed = 0; //num of games played, so far, in a tournament
```


PongGameV5.0 class – draw

Version 4.0

```
//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){  
    //displays the ball code  
    //if the ball and paddle are overlapping, hit the ball and increase the score by 1  
}  
//The player has no lives left so the game ends  
else{  
    println("Game Over!");  
    println("You have lost all of your lives: " + livesLost);  
    println("Your final score is: " + score);  
    exit();  
}
```

PongGameV5.0 class – draw

Version 5.0

```
//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){  
    //displays the ball code  
    //if the ball and paddle are overlapping, hit the ball and increase the score by 1  
}  
//The player has no lives left so the game ends  
else{  
    numberOfGamesPlayed++;  
    //If the player has more games left in the tournament,  
    //display their score and ask them if they want to continue with tournament.  
    if (numberOfGamesPlayed < maxNumberOfGames)  
        resetGame();  
    else  
        //the player has no more games left in the tournament  
        tournamentOver();  
}
```

PongGameV5.0 class – **resetGame()**

// method prepares for the next game by resetting the variables // that store the current game information.

```
void resetGame()
{
    println("Game Over!");
    println("Starting a new game...");
    livesLost = 0;    //resets the lives lost in the current game to zero
    score = 0;        //resets the score of the current game to zero
}
```

PongGameV5.0 class – **tournamentOver ()**

```
// method displays the player information, before exiting  
// the program.
```

```
void tournamentOver ()  
{  
    println("Game Over!");  
    println("Tournament Over!");  
    exit();  
}
```

PongGameV5.0 – sample output

Score: 1
Score: 2
♥ Lives lost: 1
Score: 3
♥ Lives lost: 2
Score: 4
♥ Lives lost: 3
→ Game Over!
Starting a new game...
♥ Lives lost: 1
♥ Lives lost: 2
♥ Lives lost: 3
→ Game Over!

Starting a new game...
Score: 1
Score: 2
♥ Lives lost: 1
Score: 3
♥ Lives lost: 2
♥ Lives lost: 3
→ Game Over!
Starting a new game...
Score: 1
♥ Lives lost: 1
Score: 2
♥ Lives lost: 2
♥ Lives lost: 3
→ Game Over!

Starting a new game...
♥ Lives lost: 1
Score: 1
Score: 2
♥ Lives lost: 2
♥ Lives lost: 3
→ Game Over!

Tournament Over!

5 games in tournament
3 lives in a game

Questions?



References

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