CS1013 – Programming Project

Dr. Gavin Doherty
ORI LG.19
Gavin.Doherty@cs.tcd.ie

Trinity College Dublin

DSG
Distributed Systems Group

Last weeks exercise

- Two issues: two rectangles on visible area of window as rectangle goes over the edge of screen.
- “Resetting” the various counters.

Example

```
int x;
color white = color(255);
color yellowish = color(255, 204, 0);

// setup is called once when the program starts
void setup(){
  size(200, 200);
  noStroke();
  fill(yellowish);

  frameRate(30);
  x=0;
}
```

```
// This method will be called 30 times a second
void draw(){

  // set background colour to white
  background(white);
  //  draw a rectangle
  rect(x, 20, 50, 50);

  // if upper rectangle is overlapping edge of the window
  if(x>150){
    rect(x-200, 20, 50, 50);
  }

  // we've reached the edge of the window, start again.
  if(x++>=199)x=0;
}
```

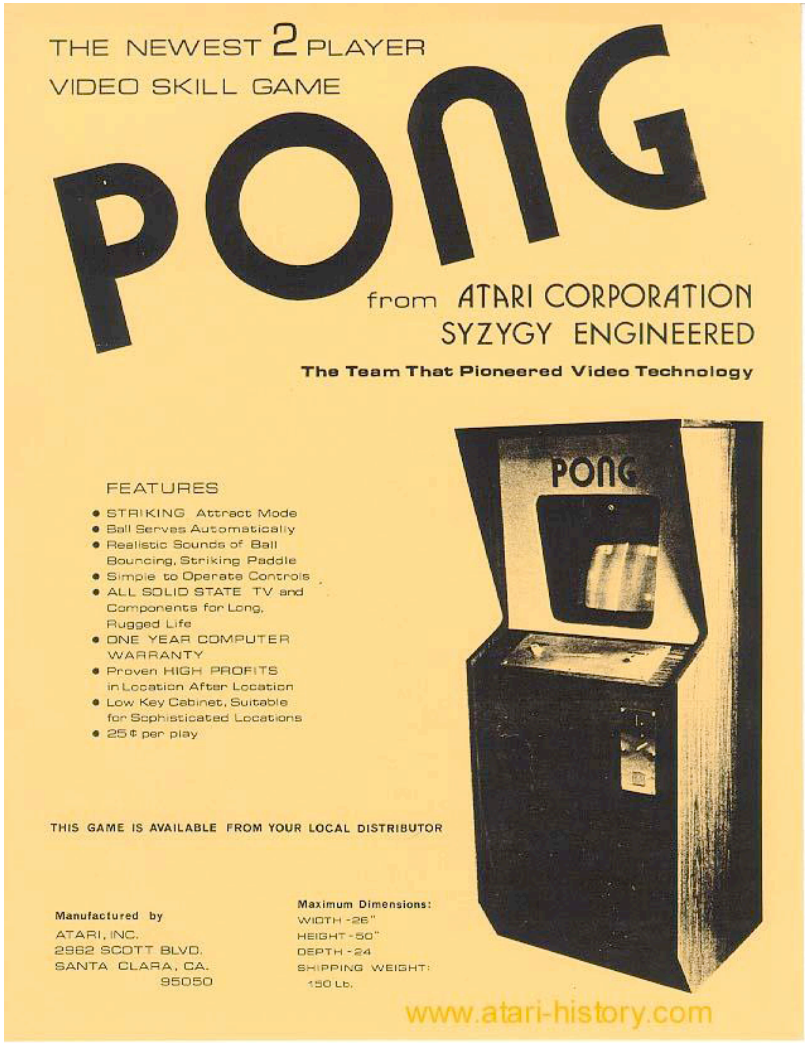
Is it correct?

- `x<150` - draw 1 square (ok).
- `x=150` – right pixel is 199 (ok).
- `x = 151` – right pixel is 200(not displayed)
- second **`rect`** call (x-200) = 151-200=-49, right edge of this will be pixel 0. (ok).
- x=199, draw two squares as before, set x to zero, doesn't get incremented after (ok).
- Can use **println** to investigate

```
void draw(){

  // set background colour to white
  background(white);
  //  draw a rectangle
  println("drawing first rectangle at "+x+" pixels");
  rect(x, 20, 50, 50);

  // if upper rectangle is overlapping edge of the window
  if(x>150){
    println("drawing second rectangle at "+(x-200)+"
   pixels");
    rect(x-200, 20, 50, 50);
  }

  // we've reached the edge of the window, start again.
  if(x++>=199)x=0;
}
```

Next topics

- Loops
- Input from user
  - `mouseX` and `mouseY`
- Displaying text
  - `class PFont`
  - `loadFont`
- Using classes
- Casting



Loops

- In a way, our processing programs already contain one (infinite) loop, as `draw()` is called over and over again.
- We can have loops inside `draw()`, but these must terminate or our programs will hang and never display anything.

Displaying text

- The PFont class stores the typefaces used for displaying text.
- First create the font in a format processing can use: got to **Tools->Create Font**.
- 18 point should be fine.
- Remember the name of the font (e.g. “Serif-18.vlw”)

```
size(600, 600);
PFont myFont = loadFont(“Impact-18.vlw”);
textFont(myFont);
text(“All your base are belong to us!”, 20, 20);
```

Input

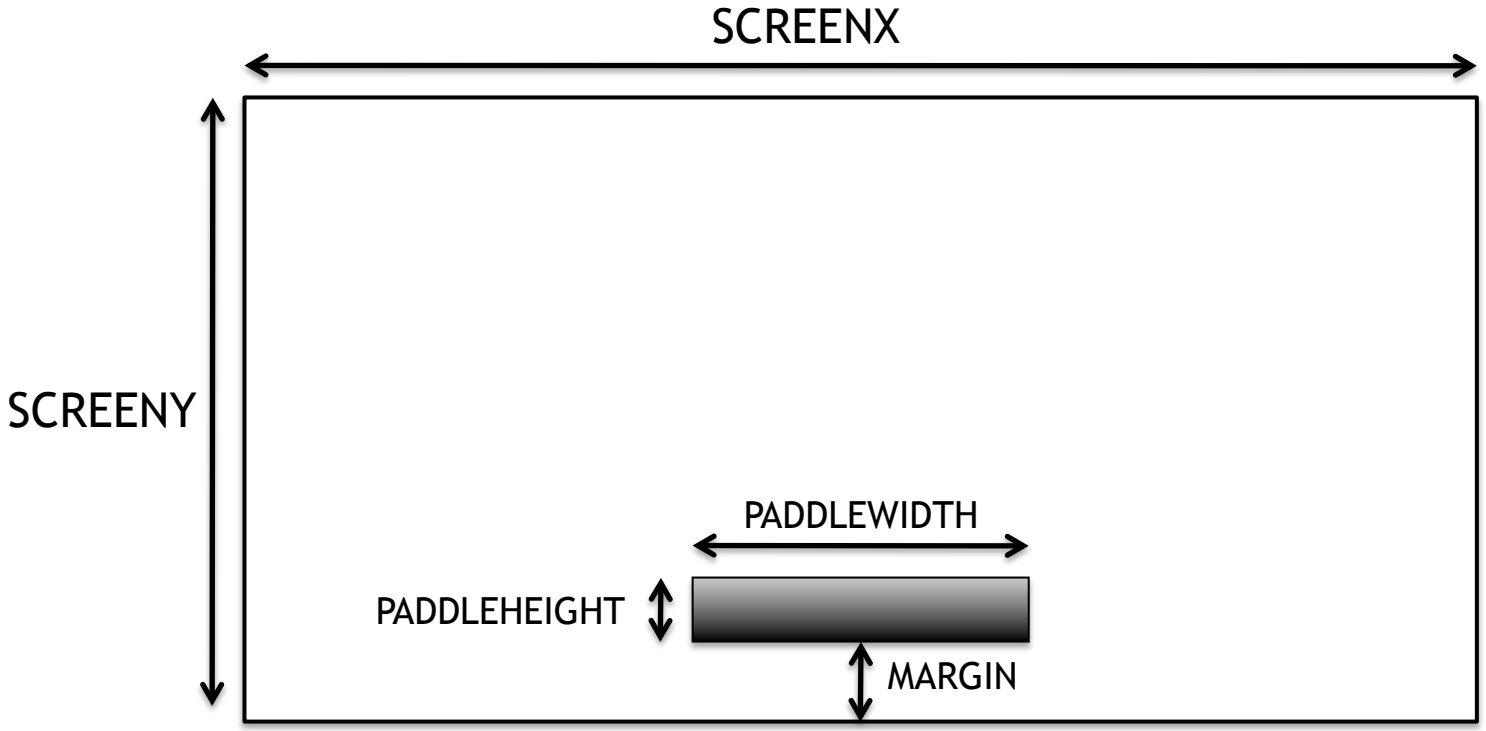
- One way of dealing with user input is to use the current value of the mouse.
- We can make an interactive program by using the **`mouseX`** value in **`draw`**()

```
void draw(){
   rect(mouseX, MARGIN,
      PADDLEWIDTH, PADDLEHEIGHT);
}
```

- Looks useful, bundle it up as a class.

Constants

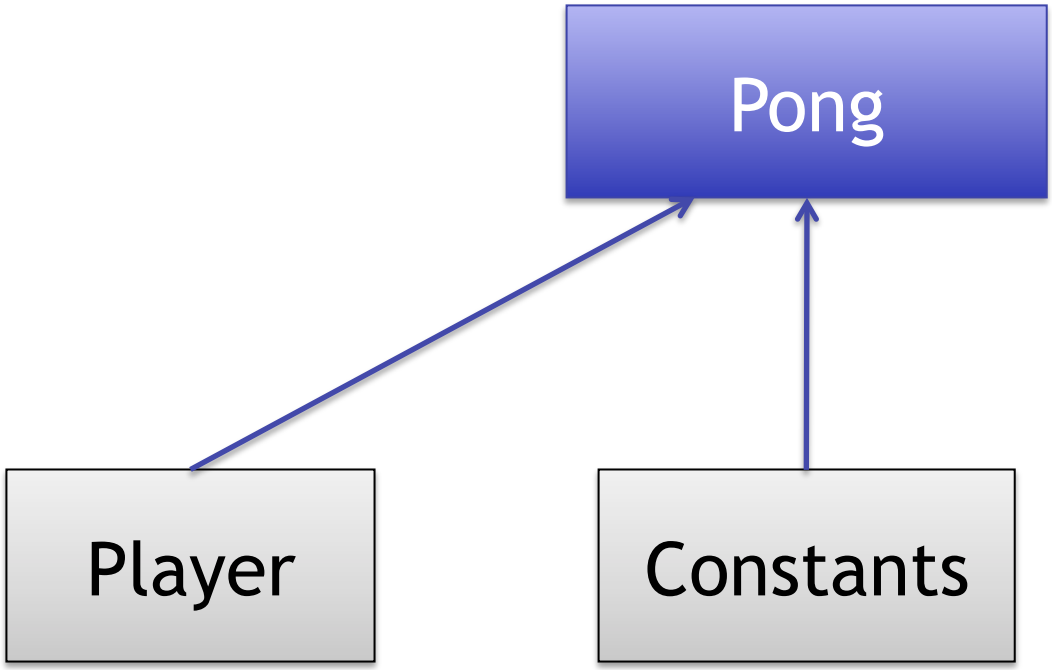
```
final int SCREENX = 320;
final int SCREENY = 240;
final int PADDLEHEIGHT = 15;
final int PADDLEWIDTH = 50;
final int MARGIN = 10;
```



Player

```
class Player {

  int xpos; int ypos;
  color paddlecolor = color(50);

  Player(int screen_y)
  {
    xpos=SCREENX/2;
    ypos=screen_y;
  }
  void move(int x){
    if(x>SCREENX-PADDLEWIDTH) xpos = SCREENX-PADDLEWIDTH;
    else xpos=x;
  }

void draw()
  {
    fill(paddlecolor);
    rect(xpos, ypos, PADDLEWIDTH, PADDLEHEIGHT);
  }
}
```

Program

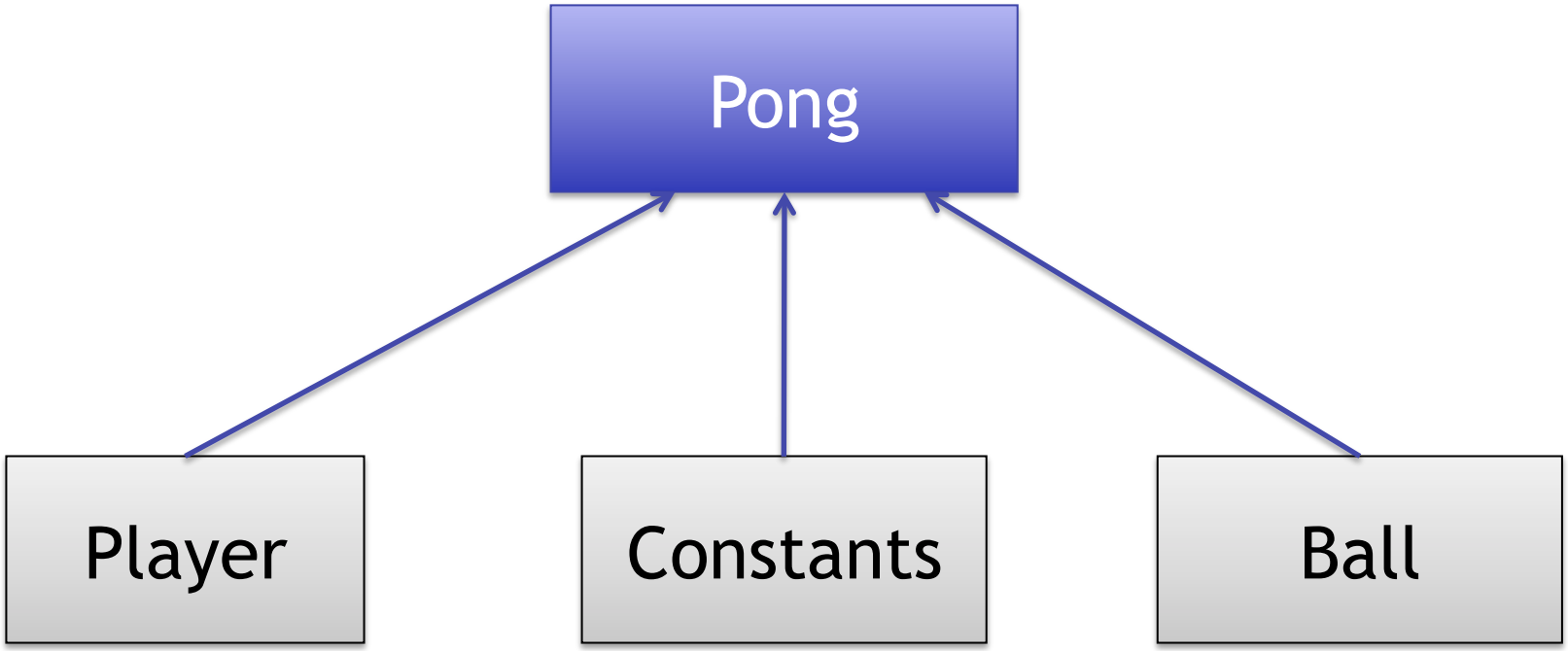


Main program

- In the main program, we create an instance of the Player class.

```
Player thePlayer;
void settings(){
  size(SCREENX, SCREENY);
}
void setup(){
  thePlayer = new Player(SCREENY-MARGIN-
  PADDLEHEIGHT);
}
void draw() {
  background(0);
  thePlayer.move(mouseX);
  thePlayer.draw();
}
```

Program



Ball

```
class Ball {
float x; float y;
float dx; float dy;
int radius;
color ballColor = color(200, 100, 50);

Ball(){
      x = random(SCREENX/4, SCREENX/2);
      y = random(SCREENY/4, SCREENY/2);
      dx = random(1, 2); dy = random(1, 2);
      radius=5;
  }
```

Ball

```
void move(){
  x = x+dx; y = y+dy;
}
void draw(){
  fill(ballColor);
  ellipse(int(x), int(y), radius,
  radius);
}
```

Collision detection

```
void collide(Player tp){
 if(y+radius >= tp.ypos &&
    y-radius<tp.ypos+PADDLEHEIGHT &&
    x >=tp.xpos &&
    x <= tp.xpos+PADDLEWIDTH){
      println("collided!");
      dy=-dy;
    }
}
```

Colliding with the walls

```
void collide(Player tp){
   if(x-radius <=0) dx=-dx;
   else if(x+radius>=SCREENX) dx=-dx;

   if(y+radius >= tp.ypos &&
         y-radius<tp.ypos+PADDLEHEIGHT &&
         x >=tp.xpos && x <= tp.xpos
 +PADDLEWIDTH){
     println("collided!");
     dy=-dy;
   }
 }
```

New main program

```
Player thePlayer;
Ball theBall;
void settings(){
  size(SCREENX, SCREENY);
}
void setup(){
   thePlayer = new Player(SCREENY-MARGIN-PADDLEHEIGHT);
   theBall = new Ball();
   ellipseMode(RADIUS);
}
void draw() {
   background(0);
   thePlayer.move(mouseX);
   theBall.move();
   theBall.collide(thePlayer);
   thePlayer.draw();
   theBall.draw()
}
```