Processing.org

@May 26, 2022

Documentation for Project:

In this video, I plan on introducing Processing, a software sketchbook and a language for learning how to code within the context of the visual arts. Processing is based on Java, so it is quite basic and quite familiar by all. However, this demonstration will dwell into how the IDE works, the built-in variables and functions that are famously used, the color pattern and effects. Most processing programs follow the flow of setup, draw and then additional required events, so I plan to introduce that along with the aforementioned ideas through demo code as attached below, which finally leads to finally creating a paddle ball game (which is also attached below).

Outline for Presentation:

15 minute video

Outline -

1. What is Processing? Where can you install Processing?
   Processing is based on Java.
2. What happens when we open processing, the **coordinate system** of the computer graphics screen and specify where the origin is.
3. Explain the toolbar - Run button (compile code and then execute) and how it is the most essential
4. What we use functions for and introduce built-in functions
5. RGB color pattern and other effects such as hue, saturation and brightness and transparency
6. Flow of code → setup() and draw()
   Command+T to auto indent code
7. Events ⇒ keyPressed(), mousePressed()
8. Declaration, initialization and usage of a variable and where to place them on processing code window
9. Conditionals : Boolean Expression and operators to be used in if conditions

Material -

Data → Variable, Array
Control → Conditional (If, Switch), Loops
Organization → Functions, Objects

Download Processing from processing.org

When we first open processing, a window opens, which is our Computer Graphics Screen. The origin of the CGS is on the top left. No negative pixels, there's no use for them.
default window size is 100 by 100, so one must set the window using size(x-limit,y-limit)

Explain the toolbar ⇒ Run button (compile code and then execute),

Syntax for code by explaining simple functions such as -
Functions: (function name - arguments that hold parameters)
line(x1,y1,x2,y2)
rect(x, y, width, height)
ellipse()
point()

Color Pattern -
stroke() and fill()
mix digital colors that follow the properties of light from red, green and blue (RGB colors) also with hue, saturation and brightness; values that range from 0 to 255.

SETUP - happens only once (where you mention size and background)
DRAW - keeps on repeating forever in a loop
make them into functions

Command+T to make indentation and line breaks proper

Blue - Function
Pink - built-in variable

Event :
keyPressed(), mousePressed()

User defined variables - declare, initialize and use

Data types and rules for name convention

println()
random() - uses int in both negative and positive

Conditionals :
Boolean Expressions - and '&&' vs or '||'

Script -

Body

Processing is a flexible software sketchbook and a language to code within the context of the visual arts. It is based on Java, but it has its own development environment and code editor. Today, we will dwell into how processing works, take a look at a few demo programs to understand the basics and will end by creating the famous paddle-ball game.

When we open the Processing IDE, we find ourselves with a raw sketchbook. On the toolbar we only have the run and the stop button. Run is used to both compile and execute the code written in the sketchbook.

-

1. The code written is either printed out on the console or gets printed on the pop-up window.
   We can use size() to change dimension (by default if not specified it is set to 100 by 100)
   The background function is used to set the background of the pop-up window.
   functions such as line(), rect() and ellipse()
   functions such as stroke(), fill() to change the colors and how RGB works as well as transparency
   explain code
2. There are two major functions called setup() and draw(), that is used to refine the previous we wrote.
   SETUP - happens only once (where you mention size and background)
   DRAW - keeps on repeating forever in a loop
   make them into functions
   explain code
   Something neat I use constantly is Command+T that auto-indents code.
3. Processing have a variety of in-built variables, one which is very useful is mouseX and mouseY. It is used in the parameters of x and y and replaces it to be the x and y- coordinates of the mouse.
   explain code
4. Processing also has in-built variables called pmouseX and pmouseY which have the previous coordinates of mouseX and mouseY. This can be easily used in code where we need both previous and current, for example a doodling sketch program using the line() function.
   What happens if we move the background function from draw() to setup()? This way the background is only changed once, at the beginning of the program, allowing all the outputs of every draw() function call visible, never having the slate empty.
5. Just like Java, we can also define our own variables. The three steps for user-defined variable are initialization, declaration, and usage. In accordance to our flow of setup() and draw(), we do the three steps as follows. The variable is declared outside both the functions (globally), initialized in setup() and is used in the draw() function.

We can also make use of the random() function for random number generation. One thing to keep in mind is that it generates a floating-point number, so keep in mind what type your variable is and cast accordingly.

6. After a while, we can no longer see the ball. In order to make the ball bounce back, we need to use our conditional statements. So, **if** the ball has x-values greater than the width of the window, we get the ball to reverse its direction. We need to make sure if the ball hits either edge, we reverse its direction, so in the condition we check for both the edges, and if either evaluates to true, we enter the block of code for the if statement.

-

Now that we have the basics down, we can get started on our code for the paddle ball game.

explain code

Resources Used -

- https://processing.org/

Personal Notes -

Processing describes itself as “a flexible software sketchbook and a language for learning how to code within the context of the visual arts”. It’s based loosely on Java, but it ships with its own development environment and code editor (the Processing “sketchbook”).

Code :

Demonstration of how processing works

```
// 1. code to show functions and how RGB colors work

size(640, 320);

background(255,255,0);

stroke(255,0,0);
line(350,150,200,250);

stroke(0,0,255);
fill(0,255,0);
ellipse(100,100,100,100);

stroke(0); // assumes same value for all three parameters
fill(255,0,0);
rect(450,100,100,200);
fill(0,200); // transparency
rect(400,200,100,100);
```

```
// 2. refined flow with setup and draw

//void setup() {
//  size(640, 320);
//}

//void draw() {
//  stroke(0);
//  fill(255, 0, 0);
//  rectMode(CENTER);
//  rect(470, 80, 100, 100);
//  fill(0, 200);
//  rect(400, 200, 50, 100);
//}

// 3. demonstrating mouseX and mouseY

//void setup() {
//  size(640, 320);
//}

//void draw() {
//  background(50); //what happens when we remove this line and just have it in the setup?
//  stroke(0);
//  rectMode(CENTER);
//  fill(255, 0, 0, 200);
//  rect(mouseX, mouseY, 70, 70);  //change parameter to say (height-mouseX)
//}

// 4. doodling by keeping the background in setup

//void setup() {
//  size(640, 320);
//  background(50);
//}

//void draw() {
//  stroke(255);
//  strokeWeight(mouseX/15);
//  line(pmouseX, pmouseY, mouseX, mouseY);
//}

//void mousePressed() {
//  background(50);
//}

// 5. intro to variables and random

//int x,y;  //declaration

//void setup() {
//  size(640, 320);
//  x = int(random(0,10));    //initialization
//  y = int(random(height));
//}

//void draw() {
//  background(50);
//  fill(255);
//  ellipse(x,y,25,25);
```

```
//  x = x + 1;    //usage
//}

// 6. conditionals

//int y;
//float x,t;

//void setup() {
//  size(640, 320);
//  x = 0;
//  y = int(random(height));
//  t = 6.0;
//}

//void draw() {
//  background(50);
//  fill(255);
//  ellipse(x,y,25,25);
//  x = x + t;

//  if(x > width+1 || x < 0){    //when circle reaches either edge, change direction
//    t = -1.5*t;    //increases speed each time by 50%
//  }

//}
```

Paddle Ball Game

```
float Xb, Yb, r, speedX, speedY;    // ball x-y location, size (radius), and speed for each component
float Xp, Yp, w, h;                 // paddle x-y location, width and height

boolean isGameOver = false;         // boolean expression used to end the game
int score = 0;                      // variable used to keep track of score

void setup() {
  size(400, 400);

  // initialize ball attributes
  Xb = random(r, width-r);
  Yb = 30;
  r = 15;
  speedX = int(random(2, 4));
  speedY = int(random(2, 4));

  // initialize paddle attributes
  w = 30;
  h = 8;
  Xp = width/2;
  Yp = height - h;

  // hide mouse cursor
  noCursor();
}

void draw() {
  background(0);                 // we don't want the window to show the previous positions of the ball
```

```
if (!isGameOver) {              // let it play as long as boolean expression is true

  // Draw game elements
  // draw Ball
  fill(255);
  noStroke();
  ellipse(Xb, Yb, 2*r, 2*r);
  // draw paddle
  stroke(0, 255, 0);
  strokeCap(ROUND);
  strokeWeight(h);
  line(Xp-w, Yp, Xp+w, Yp);
  // draw score
  fill(255, 0, 0);
  textAlign(LEFT);
  textSize(16);
  text("Score: " + score, 5, 15);

  // Move game elements
  // move Paddle
  Xp = mouseX;
  // move ball
  Xb += speedX;
  Yb += speedY;

  // check for collisions
  // bounce the ball off the two sides and the top edge
  if (Xb >= width - r || Xb <= 0 + r)
    speedX = -speedX;
  if (Yb <= 0 + r)
    speedY = -speedY;

  // check if ball lands on the paddle
  // if the ball is at the bottom edge
  //     if ball lands on paddle
  //         increment score, bounce ball up, and increase speed by 10%
  //     else
  //         set isGameOver to true;
  float d = dist(Xp, Yp, Xb, Yb);
  if (Yb > 370 && d<(w+r)) {
    score++;
    speedY = -speedY;
    speedY = 1.2*speedY;
  } else {
    if(Yb>400)
      isGameOver=true;
  }
}
else { // if game over
// putting the GameOver message and stoping the animation loop
background(0);
noStroke();
fill(255, 0, 0);
rect(50, 100, 300, 200);
fill(255, 255, 0);
textSize(30);
textAlign(CENTER);
text("Game Over! ", 200, 190);
text("Your score is "+score, 200, 230);
```

```
}
}
```