**Name: Session:**

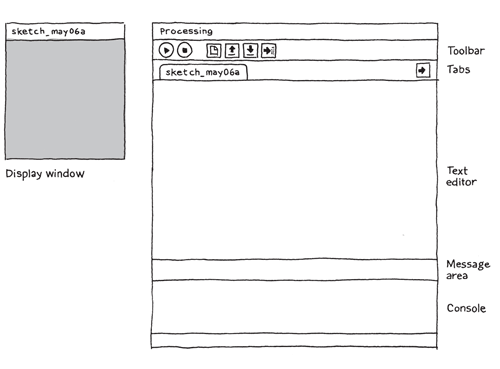
**Programming I**

**Lab Exercise 9.17.2024**

**Programming in Processing**

Processing is a simple programming language that fuses Art and Computer Science. In this activity, you will create some simple graphics.

**Run Button**



A program in Processing is referred to as a sketch. You can just type in commands into Processing and it will execute them. It is best to start your program with the following code. This code contains two functions; setup and draw. Setup is used to initialize your drawing surface and whatever you put in draw will be placed on the drawing surface.

void **setup**() {

//Place code here

}

void **draw**() {

//Place code here

}

You can think of each function as a container for code. The container contains **curly braces** to show where blocks of code start and stop. One other thing you need to remember when writing code in this language (and many others) is that each instruction must end in a semi-colon (;).

One of the first things we will always put in setup is what size the window is the command:

size(1024, 768)

which will set the size of your window as such:

void **setup**() {

size(1024, 768);

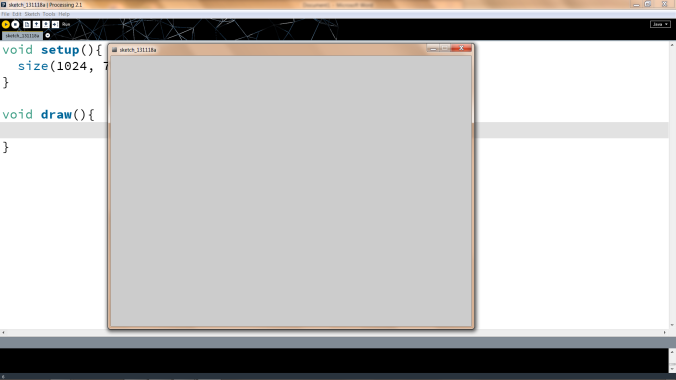
}

void **draw**() {

}

The program will run when you click on the Run button on the toolbar.

Run the program and you will get a blank window.



**This is not very interesting so let’s add some shapes.**

void setup() {

size(1024, 768);

}

void draw() {

background(100, 200, 0);

fill(255, 0, 0);

rect(500, 500, 250, 250);

fill(0, 255, 0);

ellipse(200, 200, 200, 200);

fill(0, 0, 255);

triangle(100, 100, 400, 400, 100, 700);

fill(255, 0, 255);

arc(300, 300, 300, 300, 0, .5);

noFill();

stroke(255, 0, 255);

arc(800, 300, 200, 200, 0, 2);

}

Now let us make something a bit more interesting such as random dots on our screen.

**Here is a program that will draw 100 little white circles over and over again.**

void setup(){

size(1024, 768);

}

void draw(){

for(int i = 0; i < 100; i++)

{

fill(255, 255, 255);

float x = random(1024);

float y = random(768);

ellipse(x, y, 5, 5);

}

}

**Now let us modify this to add some random colors.**

void setup(){

size(1024, 768);

}

void draw(){

for(int i = 0; i < 100; i++)

{

float r = random(255);

float g = random(255);

float b = random(255);

fill(r, g, b);

float x = random(1024);

float y = random(768);

ellipse(x, y, 10, 10);

}

}

Here is some information to help you understand what this code means:

* The x and y mean the x and y coordinates where the shape will be drawn
* The w and h refer to the width and height of the shape
* The start and stop in the arc function mean the start and stop angle of the arc (in radians).
* The r, g, and b in Coloring functions refer to the amount of red, green, and blue (0 – 255).
* Thickness refers to the number of pixels thick a line is.

|  |  |
| --- | --- |
| **Shapes** | |
| * [rect(x, y, w, h)](https://www.khanacademy.org/cs/rectx-y-w-h/839496660) Draw a rectangle | * [ellipse(x, y, w, h)](https://www.khanacademy.org/cs/ellipsex-y-w-h/839435680) Draw an ellipse |
| * [line(x1, y1, x2, y2)](https://www.khanacademy.org/cs/linex1-y1-x2-y2/827916099) Draw a line | * [triangle(x1, y1, x2, y2, x3, y3)](https://www.khanacademy.org/cs/trianglex1-y1-x2-y2-x3-y3/839546599) Draw a triangle |
| * [point(x, y)](https://www.khanacademy.org/cs/pointx-y/827809834) Draw a point | * [quad(x1, y1, x2, y2, x3, y3, x4, y4)](https://www.khanacademy.org/cs/quadx1-y1-x2-y2-x3-y3-x4-y4/1907244018) Draw any quadrilateral |
| * [image(image, x, y)](https://www.khanacademy.org/cs/imageimage-x-y/937672662) Display an image | * [arc(x, y, w, h, start, stop)](https://www.khanacademy.org/cs/arcx-y-w-h-start-stop/1903619297) Draw an arc |
| **Coloring** | |
| * [background(r, g, b)](https://www.khanacademy.org/cs/backgroundr-g-b/839653892) Set the background color | * [fill(r, g, b)](https://www.khanacademy.org/cs/fillr-g-b/839774957) Set the fill color for shapes |
| * [noFill()](https://www.khanacademy.org/cs/nofill/877946290) Turn off fill for shapes | * [stroke(r, g, b)](https://www.khanacademy.org/cs/stroker-g-b/839545910) Set the outline color for shapes |
| * [noStroke()](https://www.khanacademy.org/cs/nostroke/839859412) Turn off outlines for shapes | [strokeWeight(thickness)](https://www.khanacademy.org/cs/strokeweightthickness/877859744) Change the thickness of lines and outlines |
| * [color(r, g, b)](https://www.khanacademy.org/cs/colorr-g-b/957020020) Store a color in a variable |  |
| **Text** | |
| * [text(text, x, y)](https://www.khanacademy.org/cs/texttext-x-y/937624625) Draw some text | * [textFont(font, size)](https://www.khanacademy.org/cs/textfontfont-size/940030209) Changes the font of text |
| * [textSize(size)](https://www.khanacademy.org/cs/textsizesize/937728198) Change the size of text |  |
| **Mouse** | |
| * [mouseX, mouseY](https://www.khanacademy.org/cs/mousex-mousey/5538427537719296) The coordinates of the mouse | * [mousePressed](https://www.khanacademy.org/cs/mouseispressed/939933053) True if mouse is being pressed, false otherwise |

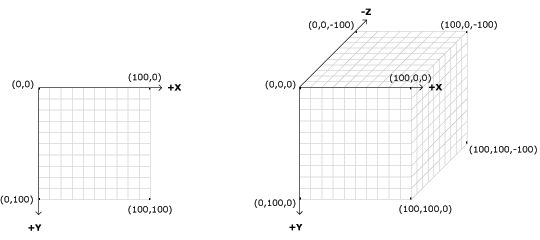


Figure 1 Processing 2D Coordinate System

**Now let’s do something with the mouse. Type in the following code in a new sketch.**

void setup(){

size(1024, 768);

}

void draw(){

textSize(48);

fill(255,0,0);

if (mousePressed){

text("Hello World", mouseX, mouseY);

}

else{

background(0,0,0);

}

}

**When you run the program it will “paint” Hello World where ever the mouse cursor is located if the left mouse button is pressed. As soon as you release the mouse cursor, the screen is painted black.**

**Now let’s paint with Circles (create a new sketch):**

void setup(){

size(1024, 768);

}

void draw(){

float r = random(255);

float g = random(255);

float b = random(255);

fill(r, g, b);

if (mousePressed)

ellipse(mouseX, mouseY, 100, 100);

}

**I know that people always tell you to not make waves, but let’s make some anyway. Create a new sketch and type in the following code.**

float y = 0;

float angle = 0;

float amplitude = 12;

float waveGap = 14;

float frequency = 3;

void setup( ){

size(400, 400);

background(255);

stroke(255,0,0);

frameRate(30);

}

void draw( ){

if (y < height){

float py = 0;

for (int i = 0; i < width; i++){

py = y + sin(radians(angle)) \* amplitude;

point(i, py);

angle += frequency;

}

y += waveGap;

}

}

**Now let’s draw a spiral by creating a new sketch and typing in the code:**

size(500,300);

background(255);

strokeWeight(3);

smooth();

float radius = 100;

int centX = 250;

int centY = 150;

stroke(255,0,0);

radius = 10;

float x, y;

float lastx = -999;

float lasty = -999;

for (float ang = 0; ang <= 1440; ang += 5) {

radius += 0.5;

float rad = radians(ang);

x = centX + (radius \* cos(rad));

y = centY + (radius \* sin(rad));

if (lastx > -999) {

line(x,y,lastx,lasty);

}

lastx = x;

lasty = y;

}

**Now let’s paint a gradient by starting a new sketch and typing in the following code:**

int dim;

void setup() {

size(640, 360);

dim = width/2;

background(0);

colorMode(HSB, 360, 100, 100);

noStroke();

ellipseMode(RADIUS);

frameRate(10);

}

void draw() {

background(0);

drawGradient(width/2, height/2);

}

void drawGradient(float x, float y) {

int radius = dim/2;

float h = random(0, 360);

for (int r = radius; r > 0; --r) {

fill(h, 90, 90);

ellipse(x, y, r, r);

h = (h + 1) % 360;

}

}

**Now let’s make a bouncing ball by starting a new sketch and typing in the following code:**

int xspeed, yspeed;

int xpos, ypos, w, h;

void setup(){

size(400, 400);

background(0);

xspeed = 3;

yspeed = 6;

w = 10;

h = 10;

noStroke();

xpos = width/2;

ypos = height/2;

frameRate(30);

}

void draw(){

background(0);

ellipse(xpos, ypos, w, h);

xpos += xspeed;

ypos += yspeed;

if (xpos > width - w/2 || xpos < w/2)

xspeed \*= -1;

if (ypos > height - h/2 || ypos < h/2)

yspeed \*= -1;

}