Holiday Image

TBA - To Be Announced? Nope - Time-Based Art! In this HotSheet you will learn how to group shapes together and get them to move over time and react to your mouse!

PROCESSING
SKILL REQUIREMENTS
DIFFICULTY 1-5

- ELECTRICAL PROTOTYPING
- ROBOTICS
- SOLDERING
- PROGRAMMING
- DIY

MATERIALS LIST

- Computer
- Processing
- Graph Paper
- Colored Pencils
- Super fine-tip Sharpie

STEP 1: Draw Something!

We highly recommend that you spend some time coming up with an analog (paper and pencil) version of what you plan to create in Processing. Sometimes it's hard enough to come up with something when you are just drawing it, let alone drawing it in code. You can then note origin points of shapes and their colors, widths and heights. This step is not required, but highly recommended. If you use graph paper it will allow you to transfer coordinates quickly and efficiently, and will all around make your life easier.

STEP 2: Shapes

The next step from the rectangles we used to create our pixel art is creating other shapes. Below you can find the basic geometry functions for Processing and their required parameters (the needed numbers for the function to work). Before you really start digging into your holiday image, just spend some time playing and drawing different shapes.

```
void setup()
{
  size(200,200);
}

void draw()
{
  background(150);
 rect(50,50,100,100);
 ellipse(150,150,50,50);
 triangle(100,100,125,75,150,100);
 quad(0,50,50,0,100,0,75,100);

}
```

`rect(x,y,w,h );` - *Rectangle with an origin at the upper right hand corner (x,y) and a width and height of W, H*

`ellipse(x,y,w,h);` - *Ellipse with a center origin at X,Y and a width and height of W, H*

`triangle(x1,y1,x2,y2,x3,y3);` - *A triangle with the three points at the given X,Y points*

`quad(x1,y1,x2,y2,x3,y3,x4,y4);` - *A quad with the four points at the given X,Y points*

`arc( x,y,w,h,Start, Stop);` - *An arc with a center X,Y with a width and height radius of W,H and a start* and stop point

Once you have played around with the different geometry functions, break your analog image down into basic shapes to be translated into Processing. See the code and given sketch for an example of each shape.

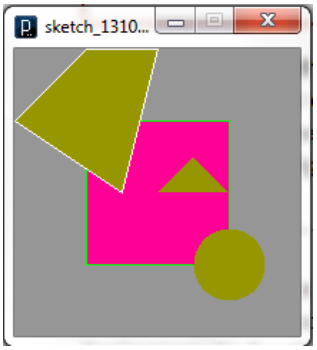
STEP 3: Lines, Weight and Color

You know from project 0 that you can draw lines in Processing. You can also modify those lines using a number of functions that precede the line function. Here are a few "modifiers" for lines and strokes

`stroke(R,G,B);` - *Sets a line or stroke color in RGB*

`strokeWeight(#);` - *Changes the thickness of a line in pixels*

`noStroke();` - *Removes all lines and outlines*




```
void draw()
{
  background(150);

 fill(255,0,150);
 stroke(0,255,0);
 rect(50,50,100,100);

 noStroke();
 fill(150,150,0);
 ellipse(150,150,50,50);
 triangle(100,100,125,75,150,100);

 stroke(255);
 quad(0,50,50,0,100,0,75,100);

}
```

STEP 4: Layers

You have probably noticed by now that shapes overlap in Processing. How do you know which one is going to be on top and which one is going to be on the bottom? The answer is simple to remember: The shape that is coded first is going to be on the bottom. Think of Processing as digital collage; if you want something on top of another shape move its place down in the code. Make sure that you are moving any modifiers that go with it as well!

STEP 5: Changing Color

By now you should have an awesome holiday image! How can you improve it? One way to make it better would be to have it change color. I think of Christmas lights that twinkle, or a poor snowman that may turn yellow after a series of unfortunate events.

So far, all of the numbers that you are using are static; they don't change. If you were to introduce a ***variable*** you could make things change. Try using a few of these built-in variables as position points, RGB values, or in any other place that has numbers.

`mouseX` - The X coordinate of your mouse
`mouseY` - The Y coordinate of your mouse
`second()` - The number of seconds since you hit play
`minute()` - The minutes on your computer's clock

STEP 8: Share your work!

OpenProcessing

If you would like to share your work (which you do), you can sign up at openprocessing.org. Open Processing allows you to share your code and view the drawing online, and allows others to fork or borrow your code to manipulate and change it while still keeping your sketch intact for others to check out.

TAKING IT FURTHER

- Try using math and variables to make interaction more intricate.
- Make a snowman wave!
- Play around with `text("text",x,y);`