

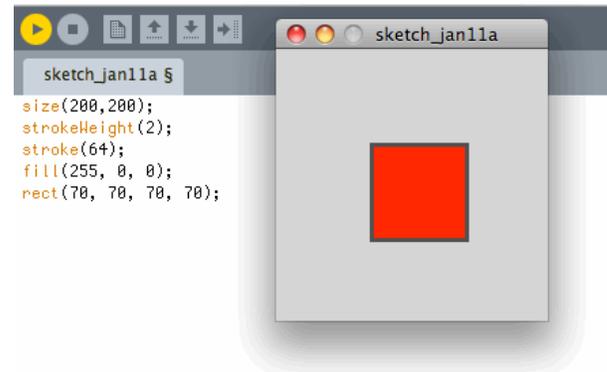


Resource: Making Processing Programs Active

The Processing language lets us to draw static, unmoving pictures that the software displays for us by drawing it once and stopping. Additionally, we can also program Processing so that the picture can change, and the software keeps drawing it each time it is updated. These are Active Processing Programs.

Static Programs

When writing static Processing programs, we simply start out and list the commands that create the image. For example, the program at right defines the size and a few properties, and then draws a rectangle. There is nothing more. There is no point in the software redrawing it because it will never be different. Drawing it once is all that is needed.



Active Programs

To make a program active, we need to structure it into multiple parts corresponding to the actions needed to redraw. These are called “blocks” in the Processing terminology. The main two are:

```
void setup( ) {  
  <set up instructions go here>  
}
```

The `setup()` block is run once to – pay attention closely – set up the initial information about the drawing. Notice that it begins with `VOID ...` this is always needed, and will be explained later in the term.

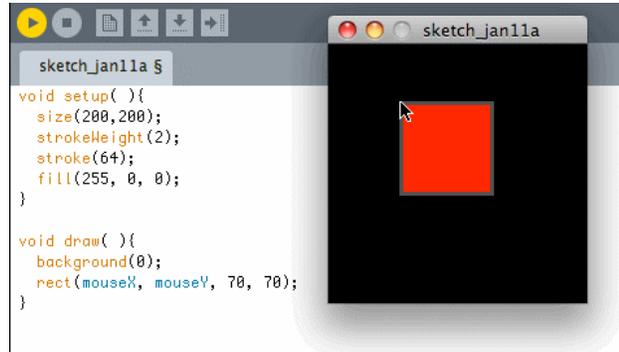
The other main block is

```
void draw( ) {  
  <drawing instructions go here>  
}
```

The `draw()` block is the part of the program that will be repeated over and over as things change. There are many ways to control how it changes and when it updates. We will introduce those later. For now, anything that should be performed multiple times should be defined in this block.

Notice where the void goes, where the parentheses go, and where the brackets go ... they **MUST** be placed in these positions. All will be explained eventually.

At right is an example of an Active Processing program. We see the two blocks, `setup()` and `draw()`, defined, and all punctuation is in its proper place. The program draws a 70x70 red square whose corner is wherever the mouse is located. So as the mouse moves around the screen, the square moves. The software updates whenever the mouse positions (given by `mouseX` and `mouseY`) change.



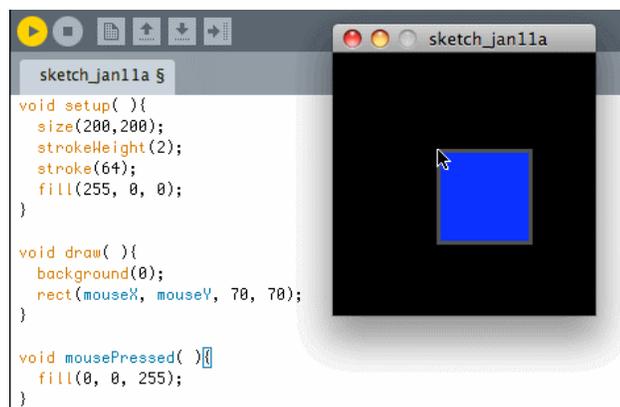
Notice that everything from the static program above is part of the `setup()` block except drawing the rectangle, because we want it to change as the mouse position changes. Also, we need to redraw the background each time so that the previously drawn rectangle is erased.

Mouse Clicks

If we want some change in the drawing as a result of a mouse click, we need to add another block. This one has the form

```
void mousePressed( ) {  
    <actions based on pressing instructions go here>  
}
```

This code is run each time the mouse is clicked. So, if we just make a single one-time change, then multiple clicks will just repeat the operation. So, the code at right fills a different color when the mouse is clicked; it's a one-time operation; once it's set to blue, it stays that way because it will only ever be set to blue by clicking the mouse.



Further Reading

Read more about these operations if you are interested at: <http://processing.org/reference/> These are "Structure" commands.