

# Forum:[Hack Forums](#)

Thread: [Rain effect](#)

**Ensky** - July 28, 2017, 7:16 am

Hey, I'm relatively new to Java, and have never used graphics before in Java or Python.

I created a rain effect with awt, I'm trying to get the x-axis of the rain drops to change ever-so slightly depending on mouse location.

My code is below:

<https://github.com/bgr-github/Raindrops>

Here's what it looks like so far:

Cheers!

**JavaGuy** - July 28, 2017, 7:44 am

First off, your [RainDrop.java](#) class is messed up in Github. If you want your program to respond to mouse input (whether it be hovering or clicking), you will want to implement a `MouseListener` class. That will provide you with the methods needed to mess with the Mouse location on your applet.

**Vana** - July 28, 2017, 3:03 pm

I think the way you were doing it was getting messed up because of integer division.

Here's how I would do it:

```
Code public void setX() {
    int mX = Game.display.getMouseX();

    if (Random.nextInt(4) == 0) {
        if (this.x > mX) {
            this.x -= 1;
        } else if (this.x < mX) {
            this.x += 1;
        }
    }

    if (this.x < 0) {
        this.x += 400;
    } else if (this.x > 400) {
        this.x -= 400;
    }
}
```

Edit: Actually, I might have misunderstood what you wanted. Do you want all raindrops to slightly converge on the mouse X, or do you mean something like having them all slant in a direction based on where the X is relative to the center?

**matt** - July 28, 2017, 10:44 pm

For a new-commer to Java you've got all of the core materials down, your code is very well formatted and I see you're taking advantage very well over Java's OOP features. As for the x-axis of the rain particles shifting relative to the mouse coordinates I'd check out another Java base language's API called processing. There's a function called map() as explained here [https://processing.org/reference/map\\_.html](https://processing.org/reference/map_.html) (I wouldn't import the whole API that's too much overhead just decompile that class and see what it's doing and use that code)

You can map the x coordinate of the mouse to a number between -5 and 5 and then add that to the x location of the rain particle

Pseudo-code example: `float offset = map(mouse.xPos, 0, window.width, -5, 5);`

Code Name

map()

Examples

```
size(200, 200);
float value = 25;
float m = map(value, 0, 100, 0, width);
ellipse(m, 200, 10, 10);
float value = 110;
float m = map(value, 0, 100, -20, -10);
println(m); // Prints "-9.0"
void setup() {
  size(200, 200);
  noStroke();
}

void draw() {
  background(204);
  float x1 = map(mouseX, 0, width, 50, 150);
  ellipse(x1, 75, 50, 50);
  float x2 = map(mouseX, 0, width, 0, 200);
  ellipse(x2, 125, 50, 50);
}
```

Description Re-maps a number from one range to another.

In the first example above, the number 25 is converted from a value in the range of 0 to 100 into a value that ranges from the left edge of the window (0) to the right edge (width).

As shown in the second example, numbers outside of the range are not clamped to the minimum and maximum parameters values, because out-of-range values are often intentional and useful.

Syntax

`map(value, start1, stop1, start2, stop2)`

Parameters

value float: the incoming value to be converted  
start1 float: lower bound of the value's current range  
stop1 float: upper bound of the value's current range  
start2 float: lower bound of the value's target range  
stop2 float: upper bound of the value's target range

Returns float

Edit: I wanted to know how this works too so I went ahead and decompiled it for you. It's very simple but I'm not great at math at all so here it is.

You'll need to fix the warnings if you don't plan on using their API.

Code `public static final float map(float value, float start1, float stop1, float start2, float stop2)`

```

{
    float outgoing =
        start2 + (stop2 - start2) * ((value - start1) / (stop1 - start1));
    String badness = null;
    if (outgoing != outgoing) {
        badness = "NaN (not a number)";
    } else if ((outgoing == Float.NEGATIVE_INFINITY) ||
        (outgoing == Float.POSITIVE_INFINITY)) {
        badness = "infinity";
    }
    if (badness != null)
    {
        String msg =
            String.format("map(%s, %s, %s, %s, %s) called, which returns %s", new Object[] {
                nf(value), nf(start1), nf(stop1),
                nf(start2), nf(stop2), badness });
        PGraphics.showWarning(msg);
    }
    return outgoing;
}

```

I went and wrote a little functioning demo, excuse the code I wrote it as fast as I possibly could.

Example 1.

Example 2.