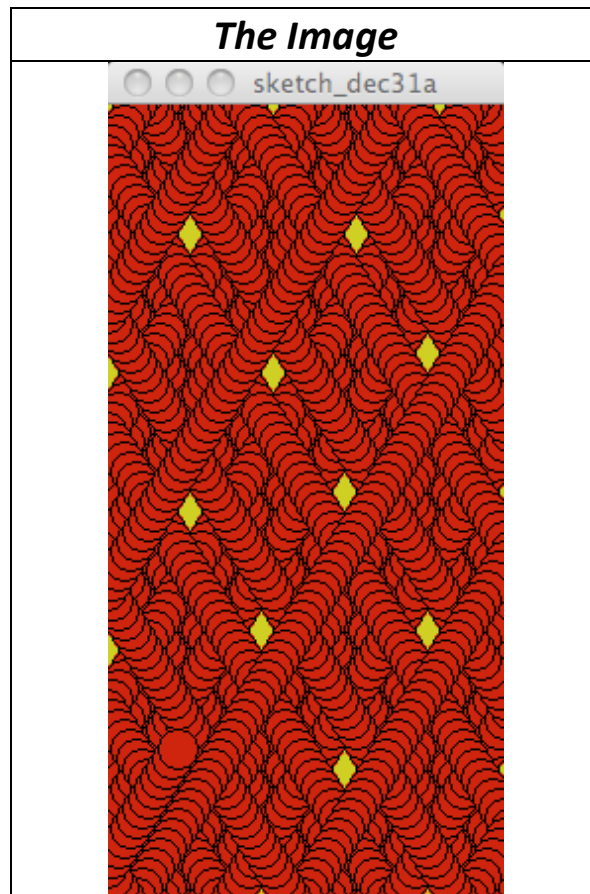
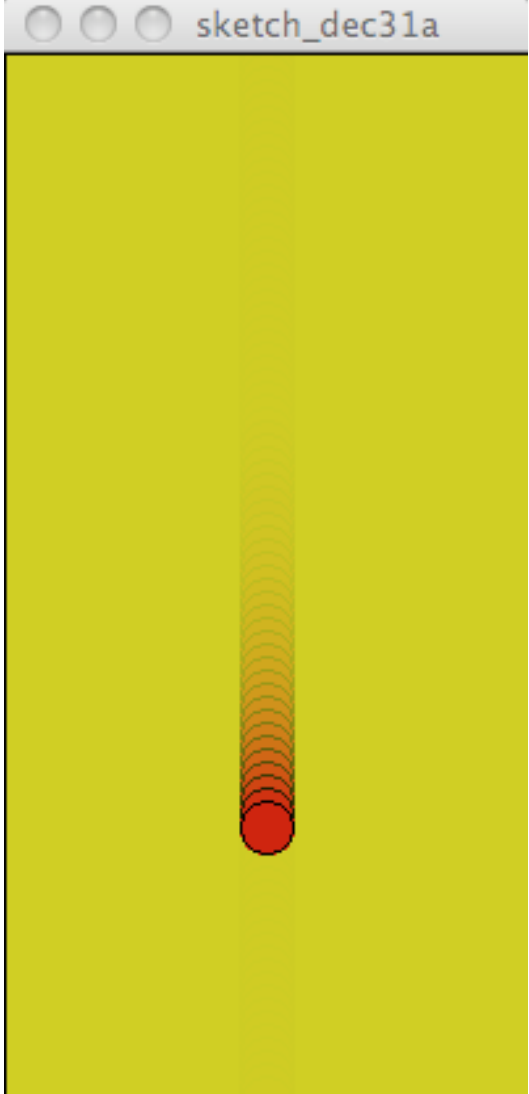


# ***The Junior Woodchuck Manuel of Processing Programming for Android Devices***



# Chapter 5

## One More Creek to Wade...

The Image	The Code
	<pre> float y;  void setup( ) {   size( 200, 400 );   y = 0; }  void draw( ) {   // draw the background   fill( 200, 200, 0, 30 );   rect( 0, 0, 200, 400 );    // draw animated circle   fill( 200, 0, 0 );   ellipse( 100, y, 20, 20 );    // move circle   y = y + 5;    // see if circle has gone too far   if ( y &gt; 400 )   {     y = 0;   } } </pre>

### First, A Brief Review...

*Wow! You have done a lot in only three days of class. We hope you have had some fun doing this stuff. If you are not having fun, talk to us today. Maybe we can help you find some.*

*In only three classes and on your own time you have written programs, worked with functions and variables, done programming arithmetic, learned how to ask questions in your code using stuff called relational operators. The result of this has been some animation you can see on your screen. Today we want to add to your animation abilities.*

*The code below is the code that is on page 2 of this exciting chapter. Let's walk through it for the review.*

Code	Its Purpose
<pre> float y;  void setup( ) {   size( 200, 400 );   y = 0; }  void draw( ) {   // draw the background   fill( 200, 200, 0, 30 );    rect( 0, 0, 200, 400 );    // draw animated circle   fill( 200, 0, 0 );   ellipse( 100, y, 20, 20 );    // move circle   y = y + 5;    // see if circle has gone too far   if ( y &gt; 400 )   {     y = 0;   } } </pre>	<p><i>This is our variable that we will use to control the vertical position of the circle.</i></p> <p><i>This is our setup function.</i></p> <p><i>This sets the size of the window.</i> <i>This assigns y the value of zero.</i></p> <p><i>This is our draw function that Processing will run 60 times a second.</i></p> <p><i>This sets the color of the background to yell but it will be slightly transparent so we can see the older frames for a while.</i></p> <p><i>This draw a slightly transparent rectangle that covers the entire window.</i></p> <p><i>This sets the fill to red.</i> <i>This draws the circle y pixels from the top edge.</i></p> <p><i>This adds 5 to the current value of y.</i></p> <p><i>This asks if y is bigger than 400.</i> <i>If the answer is true, then the value of y is changed to zero.</i></p>

## Now, Something New...

The review code above and the code we worked with last time gives our program the illusion of movement of the circle that we call **wrapping**. Last week we wrapped the circle from the right edge back to the left edge. The code in the review wraps the circle from the bottom edge back to the top edge.

Today, we want to figure out how to **bounce** the circle back and fourth. We will work with the code in the review so if you do not understand the code in the review, talk, be sure to one of us during the work session.

Before we begin to code, let's think a minute. We have perfectly good code for wrapping. Can we use this code and just modify it? Let's list the code below and mark what we already have that we can use:

Code	Its Purpose
<code>float y;</code>	<i>We need this to move the circle</i>
<code>void setup( )</code> { size( 200, 400 ); y = 0; }	<i>We need this,</i>  <i>and this,</i> <i>and this, too.</i>
<code>void draw( )</code> { // draw the background fill( 200, 200, 0, 30 ); rect( 0, 0, 200, 400 );  // draw animated circle fill( 200, 0, 0 ); ellipse( 100, y, 20, 20 );  // move circle y = y + 5;  // see if circle has gone too far if ( y > 400 ) { y = 0; } }	<i>We need this,</i>  <i>and we need this,</i> <i>and this.</i>  <i>We need this,</i> <i>this,</i>  <i>and this.</i>  <i>We need this.</i>

So it looks like we can use the entire program and just modify parts of the code. This is one reason to put comments into your code so you do not have to remember what you did yesterday or last week.

Ok - Great!!! How do we use this code? The first thing we need to do is to find the code that actually moves the circle. The reason is that this code moves the circle in only down from top to bottom. We have to find it and figure out a way to change it to be able to move the circle both down and up!

Go back and find the code that actually moves the circle before looking at the next page.

**GO ON** - Look for it before you peek.

Did you really  
look first???

We are watching  
you...

Is this the code you picked?

```
ellipse( 100, y, 20, 20 );
```

Think again if you did. Is this really moving anything? Nope. It is just drawing the ellipse 100 pixels to the right of the left edge and y pixels down from the top edge. The result is a moved circle but it did not actually move the circle.

What moves the circle?

What tells Processing where to draw the circle?

The answer is the **first two arguments**:

```
ellipse( 100, y, 20, 20 );
```

What actually moves the circle is the code that changes the value of the variable, **y**.

So the code that actually moves the circle is this:

```
y = y + 5;
```

In fact, the comment tells you that this is the code. This code is similar to the fill( ) function. When we use fill( ) we do not see anything change right then. We see the result of using fill( ) when we draw a circle or rectangle. The fill( ) function changes stuff sorta' behind the scenes. The code **y = y + 5;** works the same way. Without functions like fill( ) and code like **y = y + 5;** nothing would ever change - boring...

The reason the circle moves down from top to bottom is that we are always adding 5 to the value of y. This is also why the answer to **if ( y > 400 )** is eventually true and we reset the value of y back to zero.

What we have to do is to figure out how to change the blue part of this code:

```
y = y + 5 ;
```

If we add 5, the circle goes down.

But if we can subtract 5, the circle will go up.

How can we do this???

Let's step back and remember why we used the variable, `y` and not a constant like 100 or 42 in the original code.

We used `y` so we could change the location of the circle or "vary" it - hence the name variable. By using the variable we can vary its value and change the circle's vertical position.

To be able to bounce the circle off of the edge, we have to be able to vary the value that we add to `y` so it can move down sometimes and up sometimes. Since we want to "vary" the value we add to `y`, maybe we need a variable...

If we use a variable instead of 5, we can set its value to 5 to move it down. What value would cause the circle to move up. What would we change it to? If the value, 5 causes the circle to move down, then the value -5 should move the circle up. Think about it. . .

Let's run an experiment. Let's move the circle up instead of down. What would we have to change in our program to test this idea? Here is the code - mark it up before looking at the next page.

```
float y;
void setup( )
{
  size( 200, 400 );
  noFill( );
  y = 0;
  background( 200, 200, 0 );
}
void draw( )
{
  fill( 200, 200, 0, 30 );
  rect( 0, 0, 200, 400 );

  fill( 200, 0, 0 );
  ellipse( 100, y, 20, 20 );

  y = y + 5;
  if ( y > 400 )
  {
    y = 0;
  }
}
```



**Did you really  
mark up th code  
first???**

**Remember, we are  
watching you...**

Here is our idea:

Code	Its Purpose
<pre>float y;  void setup( ) {   size( 200, 400 );   y = 400; }  void draw( ) {   // draw the background   fill( 200, 200, 0, 30 );   rect( 0, 0, 200, 400 );    // draw animated circle   fill( 200, 0, 0 );   ellipse( 100, y, 20, 20 );    // move circle   y = y - 5;    // see if circle has gone too far   if ( y &lt; 0 )   {     y = 400;   } }</pre>	<p><i>Let's start at the bottom.</i></p> <p><i>Let's subtract 5 instead of adding 5.</i></p> <p><i>We need to ask if the circle is too high. If this is true, we need to set the circle back at the bottom.</i></p>

Code both your ideas and see if they work. Then ours and see what happens.

Ok - we have tested our idea. How do we put both ideas together to get the circle to move down until it hits the bottom and then move up towards the top?

First we need to replace the 5 and now the -5 with a variable. Variables need to have names that make sense to you and to other programmers. Picking a variable name is important. What name would you choose?

The name of your dog or cat would not be a good name:

`y = y + spot;` or `if ( whiskers < 0 )`  
might be cute but not very good.

Since the variable will be the distance the circle moves, we could call it **distance**.

Since the variable will be the change in the location of the circle, we could call it **change**.

If we want to get really silly we could use the name:

**amountOfDistanceTheCircleChangesEachFrame**

Notice how these three names are written. The first letter is small. Programmers usually follow a set of "unwritten" rules that are not required but are very helpful. One of these rules is that all variables begin with small letters and not capital letters. This tells other programmer that this is a variable. This rule is also used for function names.

Notice also that when a variable name or function name is made up of two or more words, the new words begin with a capital letter so they are easier to read and say. Compare this:

**amountOfDistanceTheCircleChangesEachFrame**  
to this:  
**amountofdistancethecirclechangeseachframe**

We are not suggesting that you use names like this silly name, **amountOfDistanceTheCircleChangesEachFrame** but we are suggesting that you use names that make sense to others.

For our code let's use **yChange**.

Our first edits are on the next page:

<pre>float y; float yChange;  void setup( ) {   size( 200, 400 );   noFill( );   y = 400;   yChange = -5;   background( 200, 200, 0 ); //   yellow }  void draw( ) {   fill( 200, 200, 0, 30 );   rect( 0, 0, 200, 400 );    fill( 200, 0, 0 );   ellipse( 100, y, 20, 20 );   y = y + yChange;   if ( y &lt; 0 )   {     yChange = 5;   } }</pre>	<p>The new variable goes here.</p> <p>Set it to -5 so the circle moves up.</p> <p>Add the variable to y. Remember that adding a negative number is the same as subtracting a positive number.</p> <p>When the circle gets to the top, change the value of the variable to move the circle down.</p>
--	---

Code this and run it and see if it works.

It almost works. The circle moves up, and then down. AND, it disappears past the bottom of the window... We have no code to send it back up.

What can we do?

Lets' do the same thing we did to solve the problem of going past the top of the window. Let's ask if the circle is too far down. If it is, we can change the value of the variable back to -5

We can add this code:

```
if( y > 400)
{
    yChange = -5;
}
```

Here is our completed code:

```
float y;
float yChange;
void setup( )
{
    size( 200, 400 );
    noFill( );
    y = 400;
    yChange = -5;
    background( 200, 200, 0 ); // yellow
}
void draw( )
{
    fill( 200, 200, 0, 30 );
    rect( 0, 0, 200, 400 );

    fill( 200, 0, 0 );
    ellipse( 100, y, 20, 20 );
    y = y + yChange;
    if ( y < 0 )
    {
        yChange = 5;
    }
    if( y > 400)
    {
        yChange = -5;
    }
}
```

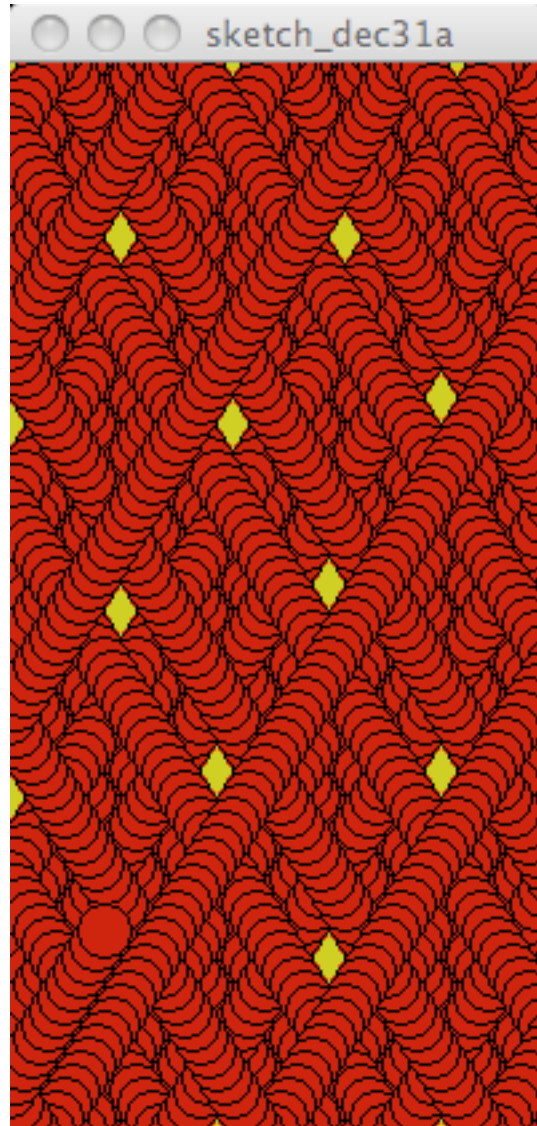
Try it and see if it works.

NOW – add code to bounce the circle horizontally as well as vertically.

If you get this working, give the change variables different values.

Comment out the line of code that draws the rectangle and see the results.

Here is a challenge for you. Below is the image on page 1 of this exciting chapter. If you figure out how to bounce the circle both horizontally and vertically, then you can modify your program to draw an image very similar to this one with only one or two edits or changes to your code.



If you get this working, try other shapes, values for your variables, and multiple shapes. This is a chance for your artistic self to peek out. We are anxious to see what you come up with.

Good Programming...