Control Again (and Global Variables)

This is going to walk through the class code labeled set08. You should get that code from the Class Code link on the course web page.

Here is the first part of Jim's code in Demo 1:

```
// Set 08 Demo 1
// More Control

int diameter, diameterChangeAmount;

void setup( )
{
  size( 400, 400 );
  smooth( );
  noStroke( );
  diameter = 1;
  diameterChangeAmount = 1;

}

void draw( )
{
  fill( 0, 0, 255 );
  rect( 0, 0, width, height );

  changeCircle( .75 );  // max percentage of the width
  drawCircle( );
}
. . .
```

The first line of executable code is the global variable declaration for **diameter** *and* **diameterChangeAmount**. *Jim made these global for several reasons:*

- *There values are needed for the entire run of the program*
- *Their values are used in more than one function.*

If he makes them local variables, their "life" is only as long as the function is being executed. Once the function is finished, the variables are destroyed and

Copyright Jim Roberts May 2011 Pittsburgh, PA 15221 All Rights Reserved

their values are lost. The other reason is that one function cannot use another function's variables. One function can pass the values of its variables to another function using parameter/argument binding but the other function cannot actually change the original value. More on this another day.

Next we see the function **setup( )** *where these two global variables are initialized.*

Then we see the **draw( )** *function. After the previous frame is covered, we see two function calls. Jim has defined these functions below in the program.*

The parameter in the function call, **changeCircle(.75)** *represents the maximum percentage of the window's width to which the circle can expand.*

Let's look at more code:

```
void drawCircle( )
{
    if ( diameterChangeAmount > 0)
    {
      fill( 0, 255, 0 );
    }
    else if (diameterChangeAmount < 0 )
    {
      fill( 0, 0, 255 );
    }
    else
    {
      fill( 255, 255, 0 );
    }
    ellipse( width/2, height/2, diameter, diameter );
}
```

This function definition is very straightforward. The fill is :

- *set to green if the diameterDeltaAmount variable has a positive value*
- *set to blue if the diameterDeltaAmount variable has a negative value*

Copyright Jim Roberts May 2011 Pittsburgh, PA 15221 All Rights Reserved

- set to yellow if the diameterDeltaAmount variable has a zero value

and the **ellipse** is drawn at the middle of the window. The global variable **diameter** is used for the width and height of the ellipse. By making **diameter** a global variable, this function can use it as a parameter for the call of the function **ellipse**.

The next function uses cascaded sequence of **if/else** control syntax.

```
void changeCircle( float maxCircleSizePercent )
{
   diameter = diameter + diameterChangeAmount;
   if ( diameter > width*maxCircleSizePercent )
   {
     diameterChangeAmount = -diameterChangeAmount;
   }
   else if ( diameter <= 1 )
   {
     diameterChangeAmount = -diameterChangeAmount;
   }
}
```

The argument in the header (the first line of the function), **maxCircleSizePercent** receives its value from the parameter in the call. The value of **maxCircleSizePercent** is 0.75. This will be used to keep the circle smaller than 75% of the window's width.

The first line of code in the function definition (inside the brace) is NOT an algebraic expression of equality. It is an assignment statement. This code:

diameter = diameter + diameterChangeAmount;

is read as:

the new value of diameter is assigned the current value plus 0.75

This line of code causes the circle to grow or shrink on screen.

Now we see a slightly different form of the **if**. It is what we call a cascaded **if/else**. The **else** has its own **if**. This

Copyright Jim Roberts May 2011 Pittsburgh, PA 15221 All Rights Reserved

if *is not nested because it is not inside the braces of the* **else**.

```
if ( diameter > width*maxCircleSizePercent )
{
  diameterChangeAmount = -diameterChangeAmount;
}
else if ( diameter <= 1 )
{
  diameterChangeAmount = -diameterChangeAmount;
}
```

The first **if** *checks to see if the circle is too big. If the circle is too big, it must not be allowed to grow any further - it must be shrunk. To do this we alter the sign of the variable* **diameterChangeAmount** *to be negative. If the diameter is not too big, we then see if it is too small. The* **else** *is followed immediately by a second* **if**. *This* **if** *is not nested within the* **else** *because it is not within the braces of the* **else**. *This* **if** *is called a cascaded* **if**. *It is only executed if the first* **if**'*s test evaluates to false. This second* **if** *checks to see if the circle is too small. If it is, it cannot be allowed to get any smaller so we reverse the sign of* **diameterChangeAmount** *from negative to positive.*

You may be thinking that you should be able to do this with a single **if** *- you can. But we will do that another day…*

Finally, we added code to allow the user to speed up and slow down the rate of expansion and shrinking of the circle. Instead of mouse input, we used keyboard input.

Keyboard input from the user will cause Processing to call a **keyPressed( )** *function if we have it in our code. The use can press two types of keys: printing keys such as '***a***' and non-printing keys such as the* **return** *or the* **up-arrow**. *We are dealing only with the printing keys.*

Copyright Jim Roberts May 2011 Pittsburgh, PA 15221 All Rights Reserved

*If the user presses a printing key (e.g. '**a**', ' '[1], '**1**', '**$**'), the key that was pressed is stored in a new system variable named* **key**. *This new system variable is of the type,* **char**. *Variables of type* **char** *store one (and only one) character.*

```
void keyPressed ( )
{
  if (  key == 'f')
  {
     diameterChangeAmount = diameterChangeAmount * 2;
  }
  else if ( key == 's' )
  {
    diameterChangeAmount = diameterChangeAmount / 2;
  }
  else
  {
    diameterChangeAmount = 1;
  }
}
```

This was our code after two edits. The first **if** *looks to see if the user pressed the '***f***' key. When that happens, the variable that is controlling the speed of the growth or shrink rate of the circle is doubled.*

*If the user did not press the '***f***' key, the second* **if** *checks to see if the user pressed the '***s***' key. When that happens the variable that is controlling the speed of the growth or shrink rate of the circle is halved.*

Unfortunately, we hit a problem. When the value of **diameterChangeAmount** *was* **1** *and we divided it by* **2**, *we ended up with a* **diameterChangeAmount** *value of* **zero** *and nothing further happened. So we added a second* **else** *to respond to any other key press. This second* **else** *is executed only when both* **if**s *are* **false**.

[1] This is the char space. The char space can be assigned to a variable or tested for in an if by typing the following characters: **single-quote space-bar single-quote** -- it works.

Copyright Jim Roberts May 2011 Pittsburgh, PA 15221 All Rights Reserved

Global Variables Again:

The global variables **diameter** and **diameterChangeAmount** are used throughout the entire program. Their values are needed for every iteration of draw( ).

The global variable **diameter** is used by these functions: **setup( ), changeCircle( ), drawCircle( ).**

The global variable **diameterChangeAmount** is used by these functions: **setup( ), keyPressed( ), changeCircle( ).**

The fact that the values of the two variables are needed throughout the execution of the program and that they must be used by multiple functions requires that they be declared as global variables and not parameters/arguments or local variables.

Copyright Jim Roberts May 2011 Pittsburgh, PA 15221 All Rights Reserved