# PROJECT 3: “YES, BUT IS IT ART?”

## CSE100/INFO100 Fluency with Information Technology
## Spring 2001

## Introduction

Visual communication relies on the basic elements of color, line, shape, and form. Recall the color experiments by Josef Albers, the geometric experiments by Modrian, and the shape experiments by Rothko shown in class. While design may emphasize communicating a particular idea and art, to a greater degree, self-expression and aesthetics, both endeavors work with this same set of elements.

Historically, new technologies have presented new opportunities for visual communication and artistic expression. Photography is a case in point. Originally, rejected as an artistic medium, photography is now embraced as an accepted art form. Similarly, plastics have found their place as a medium in sculpture. And acrylics stand along side of oils as a viable painting medium. Each medium brings with it its own interactions with the basic elements of color, line, shape, and form, and poses its own challenges for expression. In this vein, computer technologies can be seen as yet another “new medium”.

Early work in computer-generated art pre-dated color monitors. For example, Harold Cohen created a program named AARON that directed a plotter to draw lines on a white piece of paper. Cohen programmed AARON to draw random lines within certain limitations – limitations that Cohen developed to reflect formal aesthetics principles. The results from AARON’s drawing are impressive. With the advent of higher density color monitors, much more is possible. In fact, the field of computer graphics developed in response to these new possibilities. However, like any medium, computer-generated art has certain qualities tied to the nature of the medium.

In this project, we invite you to explore these qualities in the service of your own self-expression. How creative can you be? How creative can you program the computer to be? Note that these are two very different questions and reflect two different approaches to exploring creativity with computation.

You many choose any visual design problem for your project, as long as it is in good taste and permits comfortable viewing by everyone in the course. Here are some possible ideas you may wish to incorporate:

- Design an aesthetic still image (that is, once it is drawn, it doesn’t change on the screen)
- Thinking along the lines of graphic design, design a logo for a group or product, or an advertisement, or a greeting card
- Consider using animation. An image which develops dynamically (by changing shapes) on the screen will appear to the user as if it contains motion. Similarly, changing colors can be used to give the appearance of motion.

## Objectives

1. To explore computation as a medium for expressing creativity
2. To experiment with color, line, shape and form on a computer monitor
3. To use a programming language to investigate the visual aspects of a computer display
4. To use procedures, parameters, and random numbers in a computer program
5. To reflect on the creativity and computation
6. Ultimately, to appreciate the possibility (and limitations) for computation to serve as a medium for artistic expression

## Part I: Due Friday, May 11, 12 PM Noon

Plan on constructing two separate designs, one for Part I of this assignment and one for Part II. The second can be an embellishment of the first, or it can be entirely different. One strategy might be to accomplish the technical requirements listed below for Part I and concentrate on aesthetics exclusively for Part II.

**What To Do:**

1. When your project begins executing, it should show an empty form to the user. The user then clicks on the form, causing the image to be drawn.
2. You will not be using controls extensively. All of the graphics will be produced by program code, written by you as procedures. Of course, since the image is produced when the user clicks on the form, the focus of your programming efforts will start with the `Form_Click()` event handler. The window can be closed to terminate its execution (that is, you do not need to stop it explicitly).
3. In developing this program, you should emphasize writing procedures. (One could easily imagine writing a dozen small procedures for this project.) First of all, writing procedures is the purpose of the project. Secondly, and more importantly, procedures allow you to organize the development, encapsulating effects that you can then compose into more complex images. Accordingly, the `Form_Click()`event procedure should not be a very long piece of code, but rather it should merely orchestrate the calling of procedures that do the work, probably by calling other procedures.
4. At a minimum for Part I, your program should contain the following technical elements:
   a. A procedure with two or more parameters
   b. A procedure that calls another procedure of yours
   c. A procedure that is called more than five times
   d. A Do While Loop
   e. The use of a random number

**Grading Criteria:**

1. A VB6 program that contains the technical elements in 4a – e.

## Part II: Due Friday, May 18, 12 PM Noon

**What to do:**

1. Create a VB6 program that is visually interesting. What you will program is up to you 100%. You many elaborate on what you programmed in Part I or you may do something entirely new. This may seem daunting, so an easy way to get started is to experiment with some of the different effects suggested in the Appendix.
2. Write a 2-3 paragraph discussion of your experience expressing creativity through computation. For example, you might compare your experience of visual creativity in writing this program with other sorts of creative activities and visual media (e.g., paint, pencil, photography) you are familiar with. Also consider the more general question of whether, in your opinion and based on your experience, computation could be a viable medium for artistic expression.

**Grading Criteria:**

1. A VB6 program that is visually interesting.
2. The thoughtfulness of your discussion on about creativity and computation.

## How to Turn in Your Project…

Turn in your VB6 code using eturnin (as directed on the Class Web page). Be sure to turn in both your VB6 project and form.

In addition, turn in a hardcopy of your project, as follows:

1. Print out the VB6 code
2. Print out a screen shot (at some point in the execution of your program)
3. Write or type at the top of your paper:
    a. Name
    b. Section Number
    c. Project Number
    d. Email Address
    e. Student ID
4. Staple all sheets together.

Turn in the paper copy to Dowell in the reception area of the Information School on the 3rd floor of Mary Gates Hall no later than 12 PM, noon, on the due date. Paper copies will be picked up at that time and no others will be accepted.

# APPENDIX TO PROJECT 3: VB6 RESOURCES

VB6 is not a graphics programming system, but there are many things that can be done with the basic primitives that it supplies. You can find relevant material in your VB6 text as follows: graphics (pp. 275-278), random numbers (272-275).

In an effort to provide some additional useful raw material for creating interesting images, consider the following resources.

a. The size of the window that you have to work with is given by the `ScaleWidth` and `ScaleHeight` properties.

b. It is possible to "splash" the window across the whole display, i.e. maximize it, by setting the `WindowState` property of the form to 2, e.g.

```
Form1.WindowState = 2
```

After this, `ScaleWidth` and `ScaleHeight` will be huge. Find out by running this version of the `Form_Click()` procedure from a normal size window:

```
Private Sub Form_Click()

MsgBox "The initial size is " & ScaleWidth & " x " & ScaleHeight

Form1.WindowState = 2

MsgBox "The new size is " & ScaleWidth & " x " & ScaleHeight

End Sub
```

c. It is possible to create, using a combination of red, green and blue, any color that can be displayed. As an example of producing a color by the combination of red, green and blue, use the `RGB(r,g,b)` function which has the three obvious parameters. The numbers given for each of the parameters have to be between 0 and 255. Try it by writing in the `Form_Click()` procedure:

```
Form1.BackColor = RGB(221,31,203)
```

which turns the background to "Husky purple". Of course, it is possible to change these values dynamically.

d. It is possible to develop the code discussed in the book to find the proper R-G-B values for your desired colors, but that kind of code is already running in many applications. So, to find the R-G-B values for a given color, go to some "More Colors" window from some other application, e.g. Word. (From Word go `Format` →`Background` → `More Colors` → `Custom` to get to one of the places where the color palette is available; there are many others, of course.) When you set the color you want, read off the R-G-B numbers that define for the color you want, and then use them in your program when calling the `RBG()` procedure.

e. To draw a box on `Form1`, use the line drawing facilities as follows:

```
FillColor = RGB(_,_,_) ‘ Say what color is inside the rectangle

Form1.Line (boxLeft, boxTop) – (boxLeft + boxWidth, boxTop + boxHeight),
RGB(_,_,_),B
```

where you must put everything after the first line on a single line in your program. Also, fill in the blank R-G-B values with numbers or variable values from 0 through 255. Remember the key points about this call, namely the facts that you are specifying two corners of a rectangle, the `"–"` is required between the two corners, and the `"B,"` which means "box," is required. If you don’t put in the second `RGB()`call shown, then the line around the rectangle is black.

Drawing a circle is even easier:

```
FillColor = RGB(_,_,_)
Form1.Circle (centerLeft, centerTop), radius, RGB(_,_,_)
```

where, if the RGB() is omitted, the line around the circle is black. Of course, (`centerLeft, centerTop`) is the center point of the circle, and `radius` is the radius.

f. An interesting technique is a splatter painting or background. Here’s one way to do this:

```
Private Sub Form_Click()
Dim count As Integer ‘ define an iteration variable
Dim nXCoord As Integer
Dim nYCoord As Integer
Dim nRed As Integer
Dim nGreen As Integer
Dim nBlue As Integer
Randomize ' Set up random number generator
count = 0
Do While count < 10000
   count = count + 1
   nXCoord = Int(Rnd(1) * Form1.ScaleWidth)
   nYCoord = Int(Rnd(1) * Form1.ScaleHeight)
   nRed = Int(Rnd(1) * 255)
   nGreen = Int(Rnd(1) * 255)
   nBlue = Int(Rnd(1) * 255)
   PSet (nXCoord, nYCoord), RGB(nRed, nGreen, nBlue)
Loop
End Sub
```

The form starts out with a white background. It enters a loop in which it generates on each cycle a random x, y coordinate and a random color as an RGB triple, and sets the pixel at that position to that color. The loop goes through 10,000 iterations. It is not possible to go through more than 32,767 using a standard `Integer`. So, to do more iterations of splattering, you will need to enclose the basic loop shown in another, outer loop.

A key idea in this example is random numbers. Computers cannot generate true random numbers, because they follow instructions faithfully. But, they can generate number sequences that "look" random and that can be verified scientifically to have the same properties as random numbers. To use random numbers, you need to do two things: First, you must put the `Randomize` command at the start of your program so the computer sets up the randomization. Second, when you want a random number, just write the function `Rnd(1).` The result is a random number between 0 and 1. So, the procedure above finds a random color value between 0 and 255 by the code:

```
Int(Rnd(1)*255)
```

Suppose the random number from `Rnd(1)` was 0.5 (which doesn't seem too random, I suppose!), then the product would yield 127.5 Since this must be converted into a whole number, the `Int` function is applied to the result to throw away the fractional digits, giving 127.

Note: The splatters will normally cover the form, including any letters you might already have on the form. If you want to avoid covering up black letters, change the form's drawing technique `DrawMode` to 9, Mask Pen. The `DrawMode`s give different affects, so you might want to experiment with them.

g. Recall in your first VB6 lab how we got the clock to "run" by redisplaying it whenever the time event went off? That same idea can be used for animations, too. Place a timer control on your form as we did before. (Don't worry about where you place it, because it will disappear when the program starts running.) Next, declare a global variable at the top of your form code:

```
Dim tickTime As Integer
Dim yDim As Integer
```

so that it is global to the timer routine. Next, initialize the `tickTime` variable in the `Form_Load()` event procedure:

```
Private Sub Form_Load()
tickTime = 0
End Sub
```

Finally, write some code to draw on each time tick:

```
Private Sub tmrMyClock_Timer()
FillStyle = 0 'Set style to opaque
FillColor = RGB(221, 31, 203) 'Husky purple
If tickTime * 200 > ScaleWidth Then
   yDim = yDim + 200
   tickTime = 0
End If
Line (200 + tickTime * 200, yDim)-(400 + tickTime * 200, yDim + 200),
RGB(221, 31, 203), B
tickTime = tickTime + 1
End Sub
```

Remember the line drawing command must be on a single line. Then, whenever the timer goes off, the box is drawn in the next position. Run it. Nothing happen? Did you set the `Interval` on the timer? Experiment with different "speeds", smaller numbers make the timer go off more frequently, speeding up the image drawn. How does it stop from running off the form?