**Introduction**

In today's world, we all use a computer in some form. Computers help us with many tasks such as helping us with our academic studies, playing video games, sharing Facebook post and the list will keep growing. Due to our software dependence, understanding how computer programs are made is a valuable skill. Well, what does this have to do with math? Computer programs complete these daily activities, which are built with math computer languages. I know! Math is a four-letter word for most people, even for people who create computer programs. Along with the difficulty of mathematical theories, I think that many math instructors don't give practical examples that most people could relate to. Computer languages are a new medium for graphic designers to generate complex designs that have better accuracy and precision. So, I'm going to show visual artist and math students with no programming experience, the basics of computer programming and its relationship with math and geometry.

**Instructing the Computer**

What is Computer Programming? According to BusinessDictionary.com, it is “the process of developing and implementing various sets of instructions to enable a computer to do a certain task.” What kind of instructions? More importantly, what type of task? We all know that we don’t have to be a computer programmer to give instructions to the computer. In Photoshop, you give instructions to paint or draw by clicking on buttons in the toolbar. The buttons hide the code instructions that the programmer created, which make Photoshop easier to use. Programming has been a tool in other disciplines for years and has changed the world the way we interact with common applications. There are dozens of computer languages and learning them can be intimidating like a math or a foreign language course. Trust me! I know how you feel. However, learning a computer language is like learning how to speak and write. Do you remember mastering your ABC’s? Overtime, you then learned how to spell words, write sentences and create stories. This tutorial will follow that same process.

[Back to Top](file:///Users/dan/Documents/Project%201/index.html#TOP)

**Start Programming with Processing**

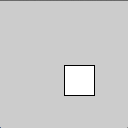
In order to program your computer, you first need a specialized program called an Integrated Development Environment (IDE) that will translate your ideas in a language the computer understands. We will be using an IDE called "Processing" that allows us to create images and interactive animation with a computer. Once you completed the installation steps, go ahead and open the Processing IDE program. Also, because the name of the IDE is “Processing”, which is a generic term. I will abbreviate the name as "PDE" (Processing Development Environment).

When you start PDE, you will see a blank window with a blinking cursor just like you would see in a Microsoft Office Word document. There is also a toolbar at the top of the window. The two buttons on the left look like media playback controls. Lets start writing a simple line of code instruction in order to see what the buttons do. Type in the following below or Copy & Paste into the PDE window:

**Copy & Paste Code 1**

**rect(50,50,30,30);**

Click on the button that looks like the Play media symbol, which is the “Run” command in the upper-left corner of the PDE window. Another window should appear with a square like the one below.



The window above is called a “sketch” in the PDE. To close the sketch, click on the “Stop” button which looks like a stop playback control

[Back to Top](file:///Users/dan/Documents/Project%201/index.html#TOP)

**Understanding Functions in Processing**

Notice that Code Example 1 is in a certain format. Before clicking the play button and without any coding experience, you probably had no idea what that code would do. **Copy & Paste Code 1** is an example of many ***Functions*** in PDE. Functions allow us to create graphics like the rectangle, as long as they are typed in a specific format.

**Rectangle Code Key**



1. *Function Name*: A function name like **rect** tells you what type of command the PDE is performing whether that is a shape, color, math operation, etc.

2. *Left Parenthesis*:"**(**" Parenthesis are used in many topics, like writing and math. A left Parenthesis tells the PDE to open the function so you could enter information.

3. *Parameters*: Parameters are the numbers inside the function. Each function has its own set of parameters and allows you to adjust the size of the shape. The amount of parameters can very. **rect(50,50,30,30);** and **point(45, 23);** have different parameters.

4. *Right Parenthesis*: The right parenthesis ")" closes the function to tell the PDE that no more parameters are needed within a function.

5. *Semicolon*: Since the PDE (and other programming languages) give the computer step-by-step instructions to output information, a semicolon is needed to tell the computer to move on to the next step.

For readability, follow this format for multiple functions in below.

**Copy & Paste Code 2**

**ellipse(50,50,15,15);  
triangle(20, 75, 10, 20, 86, 75);  
line(50,50,70,20);**

[Back to Top](file:///Users/dan/Documents/Project%201/index.html#TOP)

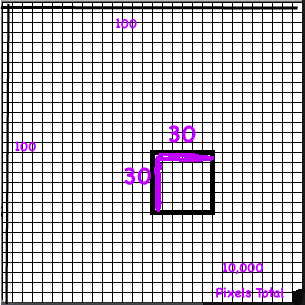
[Back to Understanding Functions](file:///Users/dan/Documents/Project%201/index.html#Functions)

**Coordinate System of Pixels**

Seeing a graph is the best way to understand the numbers in the **rect** functions for instance. Understanding the coordinate system of computer graphics is pretty straightforward. If you have taken algebra, you will notice that this system is slightly different from the Cartesian Coordinate System. The good news is that there is only 1 quadrant to worry about. The system is just two lines meeting at a point in the upper-left corner. The line in the West-East direction is called the **X-Axis** and the one in the North-South direction is the **Y-Axis**. The x-axis and y-axis meet at a special position called the **Origin**. Both x and y-axis are divided into tiny colorful blocks called **Pixels**, which make up everything you see on the screen. Without pixels, computer graphics would be very difficult to create. Because the parameters within a function literally tell you the amount of pixels needed, you will see the process of how computer graphics are made.

**The Sketch Window**

In order to draw the rectangle, the computer needs to create a **Sketch**. The computer chooses a starting point within the sketch that will be the origin. Then, counts straight to the right on the x-axis and down on the y-axis to draw the sketch window. By default, the computer creates a window that is 100 pixels wide and 100 pixels high with an invisible border. Since the PDE was programmed to know geometry ahead of time, it knows that multiplying 100 X 100 will give us an area of 10,000 pixels to draw on (later, we will make the window bigger), big enough to draw the rectangle. The computer sees the code from **Copy & Paste Code 1** and does the same thing, but starts at the center of the sketch window, which is 50 pixels X 50 pixels. Then, draws 30 pixels to the left and 30 pixels down We now have a rectangle like **t**he graph below, which shows an approximation on where the pixels would be on the screen.

****

[Back to Coordinate System and Pixels](file:///Users/dan/Documents/Project%201/index.html#Coordinate)

[Back to Top](file:///Users/dan/Documents/Project%201/index.html#TOP)

**Creating Other Shapes**

For many aspiring math students, the names of these functions should be familiar. If not, fear not. Here is an example of the basic shapes. Each sketch window is 100 X 100 pixels.

* A **Point** is the simplest shape and are made with pixels. The intersections of each pixel on the x and y-axis. To create a Point, you need an x and y coordinate. Because pixels are tiny, you have to zoom in to see them. Points in the PDE are the building blocks for the basic shapes.

|  |  |  |
| --- | --- | --- |
| **Copy & Paste Code 3** | **Point Code Key** | **Sketch** |
| **point(50,50);** | Macintosh HD:Users:dan:Documents:Project 1:pointCode.png | Macintosh HD:Users:dan:Documents:Project 1:drawPoint.png |

* A **Line** is a series of back-to-back points and needs at least 2 points, a pair of x and y coordinates separated by a certain distance, which are 2 edges of the line, also called endpoints. The PDE will automatically draw more points between the 2 endpoints you created to draw the line.

|  |  |  |
| --- | --- | --- |
| **Copy & Paste Code 4** | **Line Code Key** | **Sketch** |
| **line(50,50,70,20);** | Macintosh HD:Users:dan:Documents:Project 1:lineCode.png | Macintosh HD:Users:dan:Documents:Project 1:DrawLine.png |

* **A Triangle** is made with 3 separate Points, forming 3 lines connected at their endpoints, requiring 3 pairs of x and y coordinates.

|  |  |  |
| --- | --- | --- |
| **Copy & Paste Code 5** | **Triangle Code Key** | **Sktch** |
| **triangle(20, 75, 10, 20, 86, 75);** | Macintosh HD:Users:dan:Documents:Project 1:triangleCode.png | Macintosh HD:Users:dan:Documents:Project 1:DrawTriangle.png |

* **An Ellipse** is a round object that has a x and y point at its center and a certain width and height. Notice that the PDE starts drawing from the center instead of the upper-left corner in the rectangle example. The ellipse is drawn at all 4 corners from the center, forming a cross. If the width and height are equal, the ellipse is a circle.

|  |  |  |
| --- | --- | --- |
| **Copy & Paste Code 6** | **Ellipse Code Key** | **Sketch** |
| **ellipse(50,50,70,20);** | Macintosh HD:Users:dan:Documents:Project 1:ellipseCode.png | Macintosh HD:Users:dan:Documents:Project 1:drawEllipse.png |

* That's the end of our first lesson. Believe it or not, these examples were little programs. Next time I’ll be showing how to add shades of colors and borders to the shapes. This will also include Variables. A very important skill to know when learning computer programming
  + [Back to Top](file:///Users/dan/Documents/Project%201/index.html#TOP)

**My Web design Choices**

For the color scheme, I choose a combination of purple, beige, dark turquoise and grey shades. I choose purple it has a peaceful feeling, welcoming everyone to Learn how to program. The color scheme in the body is the color in the Processing syntax and grey is the default background color in a sketch I chose beige because it was traditionally the color for computer cases years ago. It represents the conservatives of accurate computer coding but blends in the purple and turquoise, mixing old with new concepts. I designed the logo to mimic pixels forming basic geometric shapes, giving the reader a clue on what to expect. Also, pixels are fundamental concept in Processing.

**External Resources**

* + http://hello.processing.org/ - shows an interactive way on how to get started with Processing and how its used in other subjects. [Creative Coding](http://www.youtube.com/watch?v=eBV14-3LT-g)- Shows how artist are using code. A 5 min video

[Here is my Blog[](http://tangledwebtwo.blogspot.com/)](http://tangledwebtwo.blogspot.com/)

Copyright Computing Geometry 2014 Contact: [dan06982@gmail.com](mailto:dan06982@gmail.com)