# Rendering

Multimedia productions like movies, music videos, and television commercials can be large productions that take a lot of people working together to produce the media and a huge amount of computing power to put on the screen. To save time and money, the entire process can be broken down into two (or more) phases. In the first phase, the artists and designers create media files using quicker, less expensive “rough draft” quality equipment to create the media components and store them in files. In the second phase, the components are “rendered” - produced in their final form from the files.

## Goals and outcomes:

Practice:

* reading data from a file into a program
* creating and accessing lists
* using functions

## General description:

Create a program which reads the locations of a collection of points from a file. These are the anchor points for a string art display. Using these points, it renders the string art display, drawing it on the turtle screen.

## Specific tasks:

Data files for this program will contain numbers that represent points on the graphics screen. Each line will have representations of the x and y coordinates of a single point, and the two numbers will be separated by one blank. Remember that everything in a file is read in as a value in the string (**str**) data type. You can assume that all the numbers are correct, and that there are at least **10 points** in the file.

Your program should ask the user for the name of an input file to use. Then it will read the file, converting each line into a tuple containing two floating-point numbers, then store the tuples in a list. The tuples in the list should be in the same order as the corresponding values in the file. Make sure to close the fie when you are done using it. Read A LINE STORE IT IN A TUPLE

Break this line into two numbers.

Once the list of points has been created, then use the points to create 5-pixel dots at each of the points.

Then connect each point to five others with colored lines as follows:

* Connect each point to the point 9 positions after it in the list (point 0 to point 9, point 1 to point 10, and so on) with a red line.
* Connect each point to the point 19 positions after it in the list (point 0 to point 19, point 1 to point 20, and so on) with an orange line.
* Connect each point to the point 29 positions after it in the list (point 0 to point 29, point 1 to point 30, and so on) with a green line.
* Connect each point to the point 39 positions after it in the list (point 0 to point 39, point 1 to point 40, and so on) with a blue line.
* Connect each point to the point 49 positions after it in the list (point 0 to point 49, point 1 to point 50, and so on) with a purple line.

Consider the list of points to be ‘circular’, so the point after the last point in the list is point 0. Therefore, the point that is 9 after the last point in the list is point 8. The mod function ‘**%**’ and the **len()** function will be useful here.

The order in which you make the connections is up to you – all the connections for one point then all the connections to the next, or all the red lines then all the green lines, or some other order – just make all the connections to the right points and In the right colors.

## Notes:

That’s a lot of drawing, so make sure you hide the turtle and set the speed to 0.

To grade your program, I will test it by running it first with the sample files provided. I will then run it with valid sample files that you have not seen. If is crashes on one of the provided sample files, it will not get many points. If it crashes on any valid input file, it can get no more than 50%. Your program must use at least one list of tuples (20%) and at least one user-defined function (20%). Your program must close the input file properly before ending (5%)