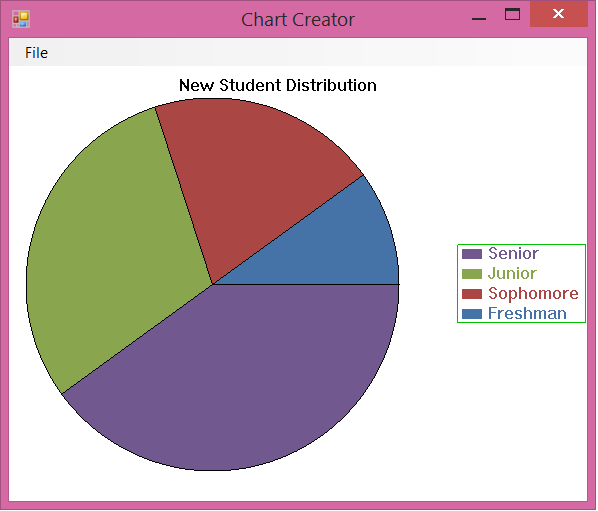
Your task for the next several assignments is to write a program named **Chart Creator** that displays either a pie chart or a line chart based on the data in an input file. The project will be broken up into several stages, but you should keep the ultimate goal in mind at every stage so your code is well-designed. There is a zipfile containing several sample data files as well as executables of solutions to each stage. It is available at the following URL (there is a link from the course webpage): <http://www.cs.hope.edu/~cusack/Notes/Notes/Graphics/Projects/Drawing%20Charts/WorkingSolutions.zip>

The final product will be an application that can read in and display data in two different formats: pie charts and line charts. When the program is first opened, the display area should be white. When a file of the appropriate format it opened, it will display the title of the chart centered at the top of the window. It will display a legend on the right part of the screen and the chart on the left part, with the chart taking up as much space as is available (while keeping the aspect ratio at 1 for pie charts). It will allow selection of a single element (a wedge of a pie chart or one of the lines of a line chart), will draw the selected element differently (details below) and allow the color used to draw that element to be changed by the user. Both the chart and the legend must be centered (vertically) within their respective portions of the window. When the window is resized, the chart should expand to continue to take up as much space as possible.

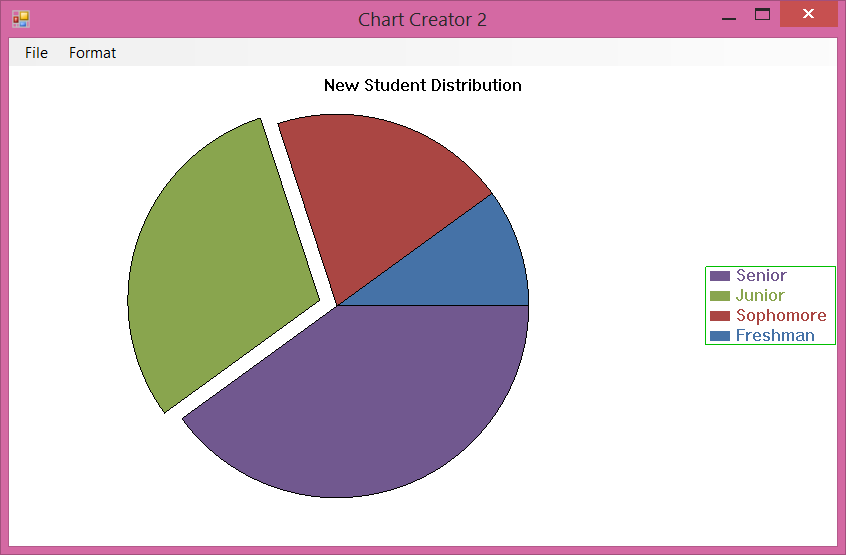
The file formats for both the pie and line charts are given toward the end of this document.

## Step 1: Pie chart

For this step you need to be able to read in the data for a pie chart, print the title, legend, and pie chart as specified. You also need to have a **File** menu with two options: **Open** and **Exit** that do as their names suggest. For now, the **Open** menu needs to only load charts in the pie-chart format. If an invalid file is selected, your program should display a dialog (Use a ***MessageBox***) with an appropriate error message.

## Step 2: Selecting wedges and changing colors

For this step you need to add the functionality that allows a wedge to be selected and its color to be changed. When a wedge is selected, it should be drawn so it is pulled out of the center of the circle (see picture below). In addition, there should be a **Format** menu with a menu item **Fill Color**. This should initially be disabled, and should only be enabled when an element of the chart is selected. Selecting this menu option should allow the user to pick a new color for the title using the standard color chooser dialog provided by C# (***ColorDialog***).  The color in the dialog should be set to the color of the selected slice when it is shown. Double-clicking on the wedge should accomplish the same thing. Changing the fill color should be reflected both in the color used to draw the pie slice and the legend entry. Clicking outside of any of the pie slices should clear the selection.

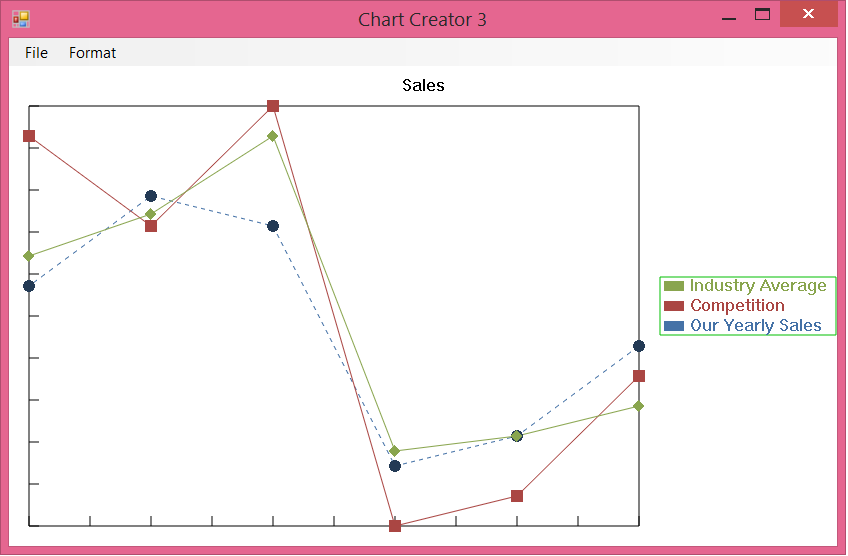


## Step 3: Line charts

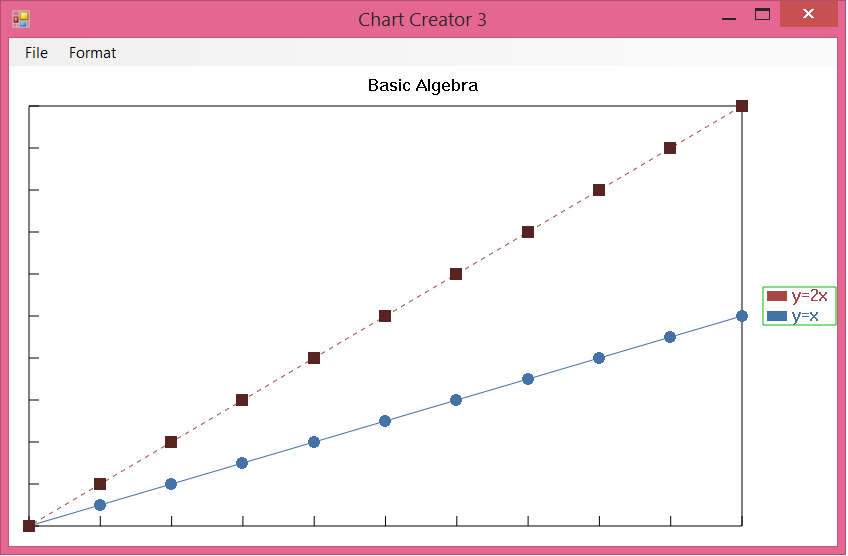
Line charts (or scatter plots) show several series of data. Each series should be drawn using lines between the points, and markers to represent the data points. You should use the following 6 types of markers:

* a solid square
* a solid circle
* a solid triangle
* a solid diamond
* a '+ sign
* a 'x' sign

You should cycle through these marker types for each data series, using the 8 colors defined for the pie slices in the pie chart.  You can again assume that there will be no more than 8 data series; the code selecting the colors and markers is the only portion of the code that should depend on the number of available colors. The color as well as the marker should be indicated in the legend (this is not represented in the sample solutions or the drawing below).

There should be a box drawn around the data corresponding exactly to the minimum and maximum values in the chart in both x and y. Each axis should be split into exactly ten sections by drawing hash marks as seen in the example.

In the example, the data series named *Our Yearly Sales* has been selected; this is indicated by the dashed line being used to draw the data series and the fact that the marker for the series is slightly darker.  As with pie charts, when a data series has been selected, the *Format -> Color*menu item should become enabled; selecting that item brings up a color chooser allowing the user to select the color for the data series. Also double-clicking on a line should bring up the color chooser. You should consider a line as being clicked on if the mouse coordinates are “close enough” to the line. The markers are of fixed size; that is, they are not dependent on the size of the window or viewport. This will require some thought since you will not be able to use the world coordinates to draw the markers (why not?).

The chart area should take up as much of the application window as possible, while leaving some margin between the left/bottom edges of the plot area and the window, as well as some space between the legend and the plot area.  Note that this means that the aspect ratio of the plot area will likely not match the aspect ratio of the viewport.  This can be seen in the screen shot below - note that the line y=x does not appear to be at a 45° angle; in fact, it is the line y=2x that is drawn this way.

## Chart Colors

You will not know in advance what the labels will be, nor how many lines will be in the input file. In order to simplify color management, you may assume that there will be no more than 8 unique labels in the file. Use the following colors, assigning them in this order to each new data item. The color management aspect of your program should be the only part that assumes there are no more than 8 unique labels. If more labels are present, you may reuse these colors.

|  |  |  |  |
| --- | --- | --- | --- |
| **Color** | **Red** | **Green** | **Blue** |
| 0 | 69 | 114 | 167 |
| 1 | 170 | 70 | 67 |
| 2 | 137 | 165 | 78 |
| 3 | 113 | 88 | 143 |
| 4 | 65 | 152 | 175 |
| 5 | 219 | 132 | 61 |
| 6 | 147 | 169 | 207 |
| 7 | 209 | 147 | 146 |

In Steps 2 and 3 users will be able to change the color of any of the elements, so make sure you take this into account as you design your code. (i.e. you should explicitly store what color each element is.)

## File Format

The data files for both pie and lines charts are text files. The first line of the file contains either the letter **p**(for pie chart) or **l** (for line chart). The second line specifies the title of the chart. Subsequent lines contain chart specific information to represent the data values; these lines are specific to the type of chart.

### Pie charts

Each remaining line for pie charts represent a data point that contains the name of one of the legend entries.  Here's an example that shows the data for the pie chart shown above (with the first two lines omitted).

Senior  
Senior  
Freshman  
Senior  
Junior  
Senior  
Junior  
Junior  
Sophomore

### Line Charts

A line chart consists of 1 or more data series.  Each data series begins with a line having the name of the series.

Next come the data points for the data series.  There can be an arbitrary number of points, in the format x,y.  The end of a data series is indicated by the presence of a -1 on a line by itself. See a sample data file at the end.

This file contains plots the functions **y=x** and **y=2x**, and is the file used to generate the line chart shown above.  You should not make any assumptions about the ordering of the points in the file; while the points above are shown in ascending order by their X coordinate, you should not assume that this is the case.

White space can be contained anywhere within the file, and is only significant in the chart title and data series names.

## Some Helpful Information

* There is a method in the ***DrawingTools*** class called **MeasureText** that facilitates measuring how wide a particular string of text will be.  The most useful way to do use this is to ask it to return pixels.  In addition to **MeasureText**, **DrawText** will draw the desired text on the screen.  Calling **MeasureText** will actually draw the text on the screen; you should therefore measure any strings you want measured and then clear the frame buffer to avoid having the text used for measuring show up on the screen. That means you need to measure everything you need to measure before you draw anything!
* To help you keep track of the occurrences of each unique value, C#, contains a class that implements a map-like data structure. In C# this data structure is the **Dictionary.** Like Java's **Map** interface, **Dictionary** is a generic class, parameterized by both the **key** type and the **value** type. **Dictionary** provides a **Keys** property which allows access to an enumerable collection of the key values within the Dictionary.  A helpful subclass of **Dictionary**is **SortedList**.
* A tutorial on designing menus with Visual Studio can be found at <http://www.techotopia.com/index.php/Creating_Top-Level_Menus_in_C_Sharp>

## Handing it in

Be sure to commit your changes to the SVN repository before the due date.

## Sample data file (Pie Chart)

p

New Student Distribution

Freshman

Sophomore

Sophomore

Junior

Junior

Junior

Senior

Senior

Senior

Senior

## Sample data file (Line Chart)

l  
Basic Algebra  
y=x   
0,0  
1,1  
2,2  
3,3  
4,4  
5,5  
6,6  
7,7  
8,8  
9,9  
10,10  
-1  
y=2x   
0,0  
1,2  
2,4  
3,6  
4,8  
5,10  
6,12  
7,14  
8,16  
9,18  
10,20  
-1