Google Summer of Code

Idea Proposal

ECLIPSE FOUNDATION

Project: **SWTChart**

Mentor: Philip Weing

- **Extending SWTChart to include Simple and Multi-Level PieCharts.**
- **Correcting the Issue of Slider Orientation in SWT.**

Name: **Himanshu Bhushan Balasamanta**

Identification: Electrical Engineering sophomore at IIT BHU Varanasi

Contact : 9326993986

Email : himanshubb.eee18@itbhu.ac.in

Github: Himanshu-Balasamanta

Project Proposal

Extending SWTChart to include Simple and Multi-Level Pie Charts.

Viewing data distribution in terms of its subcomponents distribution is of great value to analysts. A PieChart is an elegant way of viewing such data.

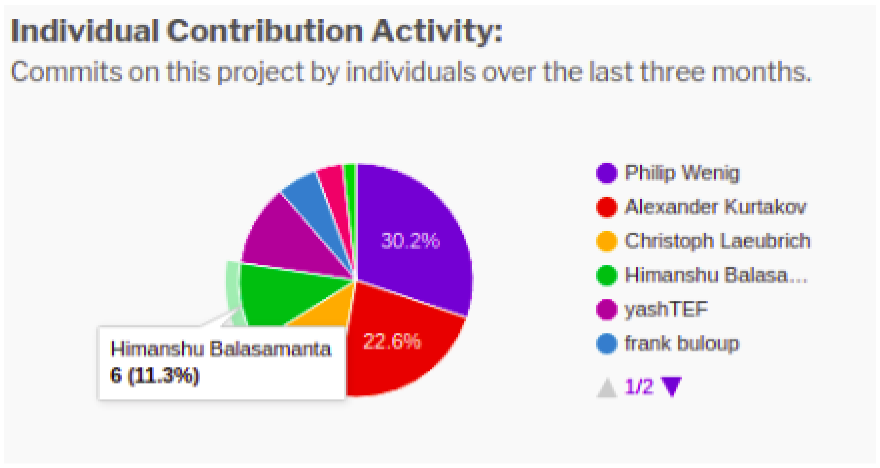
SWTChart is a charting library that is built on SWT. It has a flexible structure that can help in extending the project to include different types of charts.

Currently the SWTChart contains 3 different types of charts.

- Bar Chart (helps in comparing data of different categories).
- Line Chart (helps in analysing data variation with time).
- Scatter Chart (helps in plotting data locations on 2D plane).

I propose to add to the list

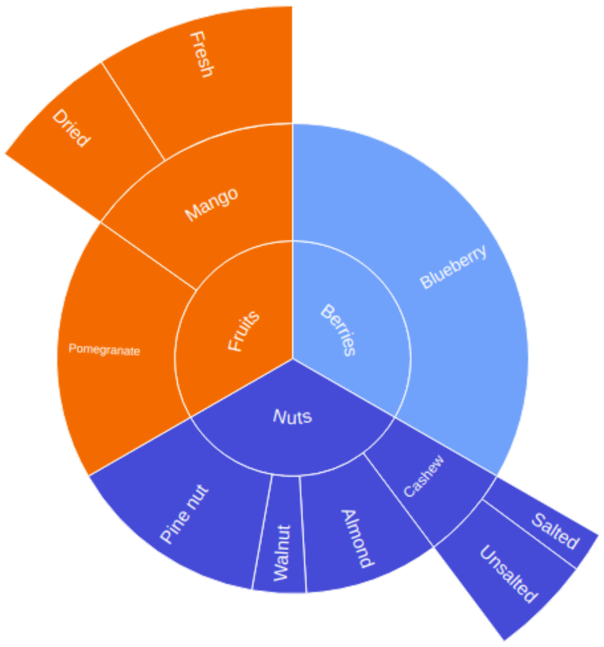
- PieChart (helps in viewing the distribution of data into its components)
- Multi-Level PieChart (helps in visualising data distribution hierarchy).

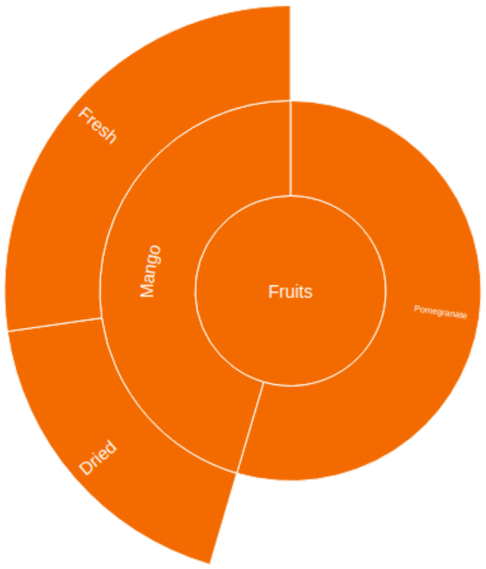


Above Fig.(Pie Chart taken from eclipse swtchart webpage displaying the % distribution of contributions to the projects in the last 3 months.)

Addition of Multi-Level Pie Charts will

Help in viewing sub series of a particular Series together with the series, which gives a very detailed outlook of the data distribution.



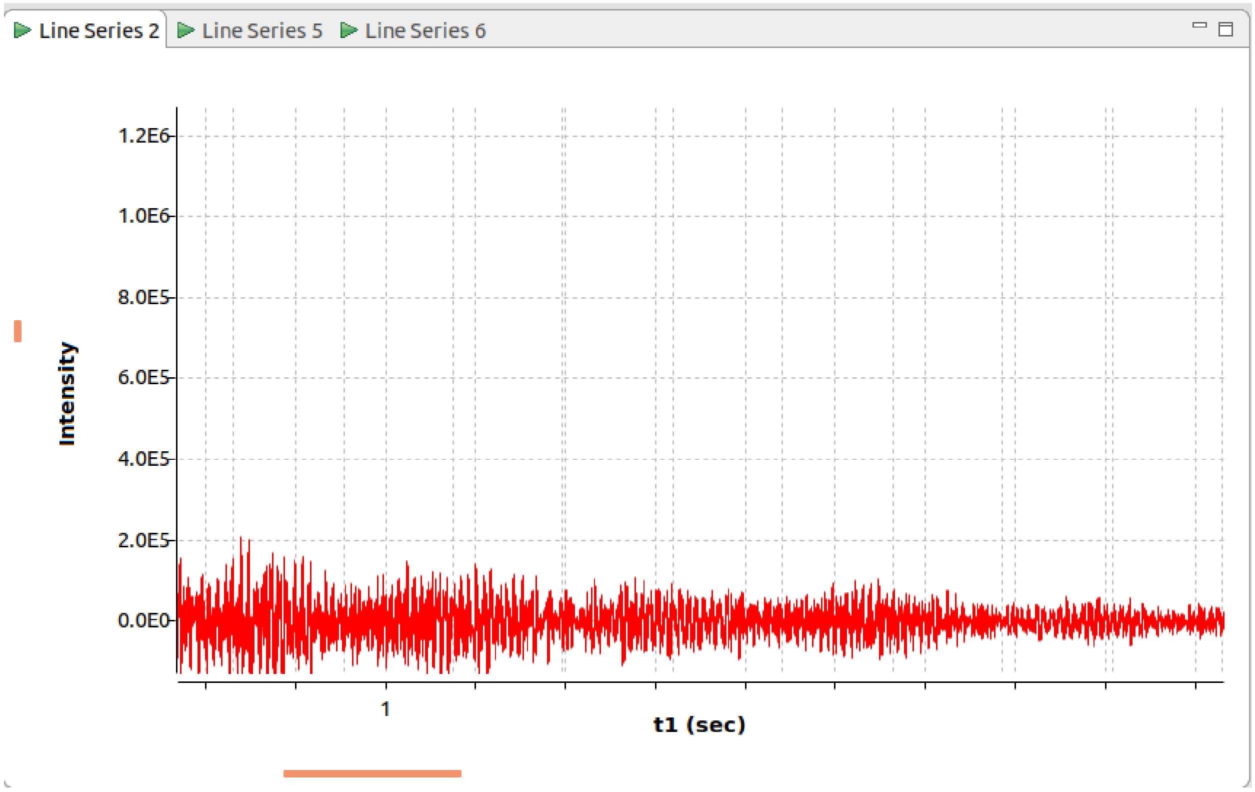


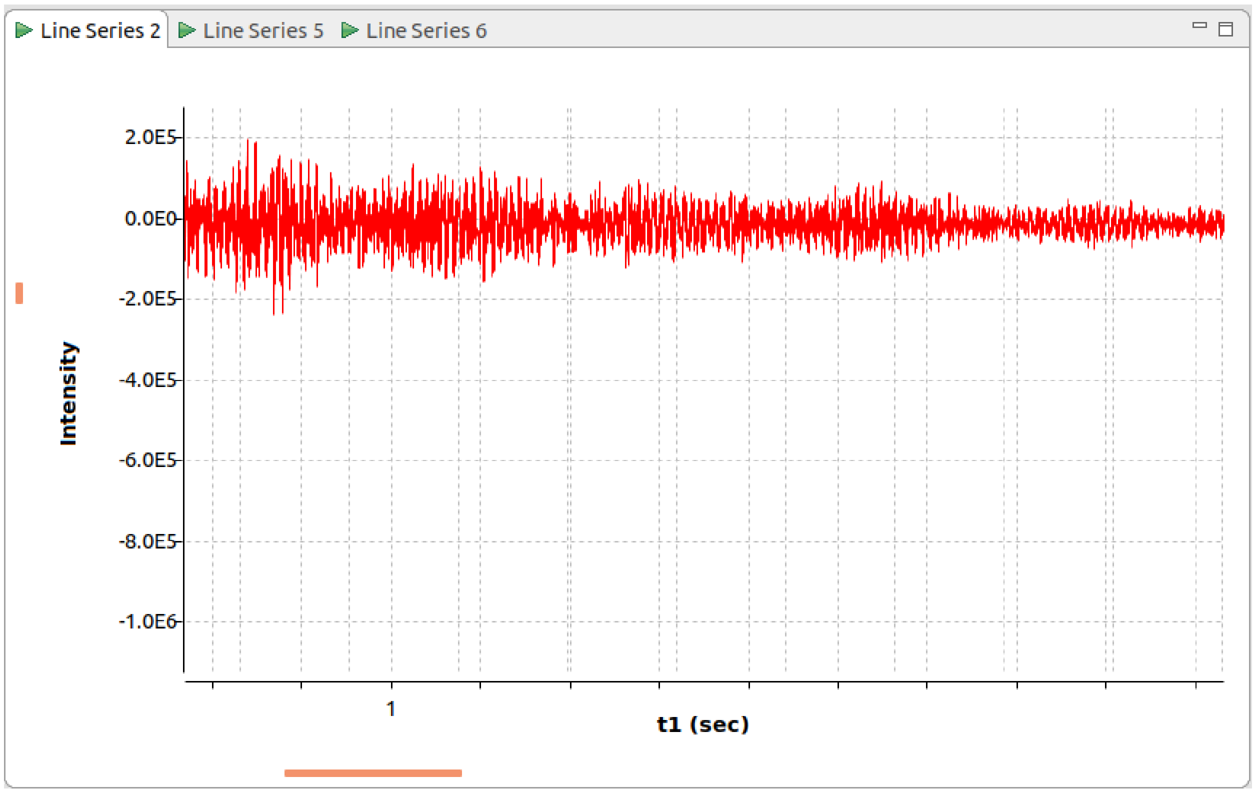
Correcting the issue of Slider Orientation in SWT

The default direction of increase of scroll values in a widget is from top to bottom. However while creating Charts,the direction of increase in y axis is primarily increasing from bottom to top. The Slider class of SWT offers a method to set the Orientation of a Slider, but due to a bug in SWT, it does not change the default orientation of Vertical Slider.

The issue for linux has been solved by me and is ready to be included into SWT. With some help, I believe I can set it for the other OS as well.

The following picture shows the issue. As the slider is dragged above, the chart also moves upward. This setting is not reversed by calling the setOrientation method of Control Class.


Line Series 2
Line Series 5
Line Series 6
Intensity
1.2E6
1.0E6
8.0E5
6.0E5
4.0E5
2.0E5
0.0E0
1
t1 (sec)


Line Series 2
Line Series 5
Line Series 6
Intensity
2.0E5
0.0E0
-2.0E5
-4.0E5
-6.0E5
-8.0E5
-1.0E6
1
t1 (sec)

Project Goals (Required)

1. Add API to create Simple PieChart.
2. Add API to create MultiLevel PieChart
    a. Allowing the user to create different Elements with the same Labels if their parents are not the same.
    b. Allow option to set Visibility of elements.
3. Allow runtime change of chart drawn in Multi-Level PieChart (example shown in above picture).
4. Solve the Vertical Slider issue of SWTChart for Linux.
5. Solve the Vertical Slider Issue of SWTChart for other OS, if given proper support.

Project Goals (Optional)

- ❑ Setting different color relationships between parent data and its subcategories.
- ❑ Option to restrict the number of elements shown in Legend.
- ❑ Ability to draw more than one Pie Series in the same chart.
- ❑ Make a new style for axes set, named as central, that sets the origin at the centre of the plot area.
- ❑ Add two new style constants to SWT, and use them to improve the flexibility of setting widget orientations.

Plan Of Action

Underlying Data Model Structure

The flexibility in the swtchart is that though the current charts included only use the Cartesian Double Array Data Model to store its data, the architecture provides the chance of creation of other Data Models, and use them for creation of different charts.

The Multi-Level PieChart shall be based on a tree with each node being an object of Class Node. Each node of the tree shall contain the information required to identify the element, draw the element and data to be displayed as tooltip or in legend.

Each Node shall contain the following data

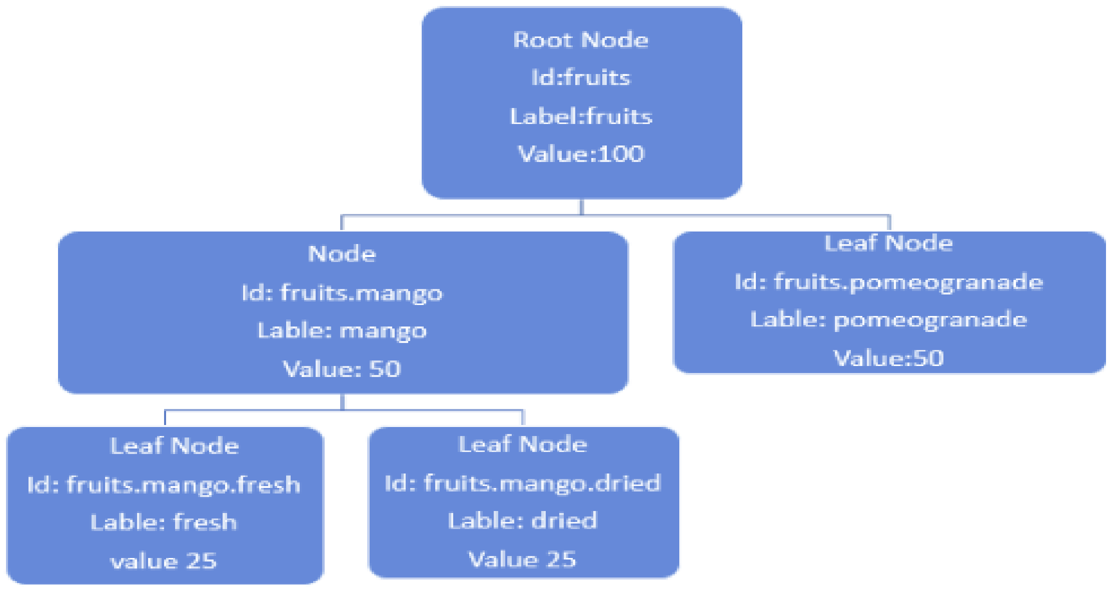
1. Label: This is what the user identifies the node with.
2. Id : An unique Identity that is formed by concatenating it's label with the id of its parent. Id of the root is the same as its Label.
3. Value: This is a double.
4. PercentageRoot: This is the percentage of value with respect to the root.
5. NumId : This is a number which shall be used to assign color to the node during call to Graph Coloring Algorithm.
6. Color : This shall be decided by the output of the Graph Coloring Algorithm.

7. Start and Stop Angles: Its value is set by the DFS algorithm.
8. Depth: The distance of node from root. Will be used in fixing the radius.

The IdNodeDataModel class is instantiated only when a new Multi-Level Pie Series is created. Unlike the currently used data model, the IdNodeDataModel is dynamic, and assigns and modifies most of the node data at fly time.

The IdNodeDataModel contains

1. A map of String, Node that maps the id to the node.
2. A vector of vectors of Strings, where each vector stores ids of all nodes having the same depth.
3. A pointer to the root node of the tree.
4. The depth of the tree.



User API

The Final API Shall be made available to be implemented as:

To create the primary series, for example

```
    static public Chart createChart(Composite parent) {

        final double[] values = {0.0, 0.38, 0.71};
        final String[] idSeries = {"fruits","berries","nuts"};
        final double[] subValues = {0.92, 1.0};
        final String[] subSeries = {"Mango","Pomeogranade"};

        Chart chart = new Chart(parent, SWT.NONE);

        SeriesSet seriesSet = (SeriesSet)chart.getSeriesSet();

        PieSeries pieSeries = (PieSeries)seriesSet.createSeries(SeriesType.PIE, "pie series");

        pieSeries.createSeries(idSeries,values);

        return chart;
    }
```

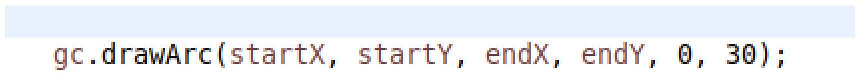
And for subsequent subSeries

```
final double[] subValues = {0.92, 1.0};
final String[] subSeries = {"Mango","Pomeogranade"};
/**id of parent element, String[] of children identities, their values*/
pieSeries.createSubSeries("fruits" ,subSeries , subValues);
```

Underlying Processes

Currently the series of SWTChart draw themselves on the basis of axis through an interface of range settings of axes. However, in PieCharts there are no axes. To utilise the current model of SWTChart however, we shall not dispose the axes. The Axis and Grid shall be set the same color as the background to make them invisible. The axes shall help us in drawing the arcs required for Pie Charts.

The drawArc method of GC takes as parameters the diagonally opposite pixel Coordinates to paint.



```
gc.drawArc(startX, startY, endX, endY, 0, 30);
```

An axis can provide an easier way to set parameters for the drawArc method, if its range is set to twice the depth of the tree.

After every modification in the series, a dfs(depth first search) check will be called to check and update the data of the child to inherit required settings from parents.

To make the nodes distinguishable, another necessity that will need to be addressed is preventing adjacent elements from having similar colors. This shall be solved by implementing the Graph Coloring Algorithm. The SWT class defines 11 colors, which we shall use. This is also to be called after every series modification made by the user.

Graph Coloring Algorithm

Every node shall store the value of its distance from the root node (its depth). All nodes at equal distances from the root will add their id to a vector and the vector is sorted. The sorted vector now shall have the ids of the elements in an order that circularly adjacent elements are connected to each other and each element shall be connected with it’s parent and children. The nodes are then numbered uniquely and an adjacency matrix is created. The colors are numbered 1 till 11 and the data is fed to the Algorithm to get an array of colors as output.

Incorporating Into SWTChart Architecture

SWTChart has its main codes at org.eclipse.swtchart and org.eclipse.swtchart.extensions bundles.

The org.eclipse.swtchart has the code which draws the chart, setting the entire series to be drawn into the chart area, the settings of axes and series and their interface (axis range) and other settings that are related to such.

The extensions project creates an interface between user and SWTChart that enables additional settings that registers more events that enables the interface to be more user friendly. Also it provides numerous templates with predefined settings to make chart plotting very scientific.

The code to create Pie Charts and Multi Level Pie Charts shall be placed in SWTChart project, and the settings that enable redraw with different nodes as root in extensions project.

The work on the project has been started at https://github.com/Himanshu-Balasamanta/swtchart/tree/pie-chart

Issues

Due to the potential presence of a large number of elements that need different marking, a decision to label all elements within their plotted area or to leave it just to tooltips or to fix the legend to display only the primary series nodes.

SWT: Solving the issue in Slider

About SWT

SWT is the widget toolkit used by Eclipse that gives performant access to the platform's native tools in a portable manner. SWT delegates the drawing to the underlying operating system, thus letting the OS do the heavy work and minimising the time spent in the JVM.

In linux, the SWT uses GTK to offload it's work.

SWT is infact the Layer between PlatformUI and underlying GTK2 or GTK3 of the system.GTK (I prefer saying GTK+) is a widget toolkit that is written in C. The communication between SWT(Java) and GTK(C) is handled by the calls to the methods of the OS class in SWT.

The Issue

The Slider Widget of the org.eclipse.swt.widgets package extends the Control Class. In a call to set it's orientation by the Control.setOrientaion(int style) method, it takes a style int as the parameter. The SWT class has styles only for RIGHT_TO_LEFT and LEFT_TO_RIGHT. So the call given to set the orientation of a Vertical slider does not have any proper style that can customise the direction of increase in the value.

Proposed Solution

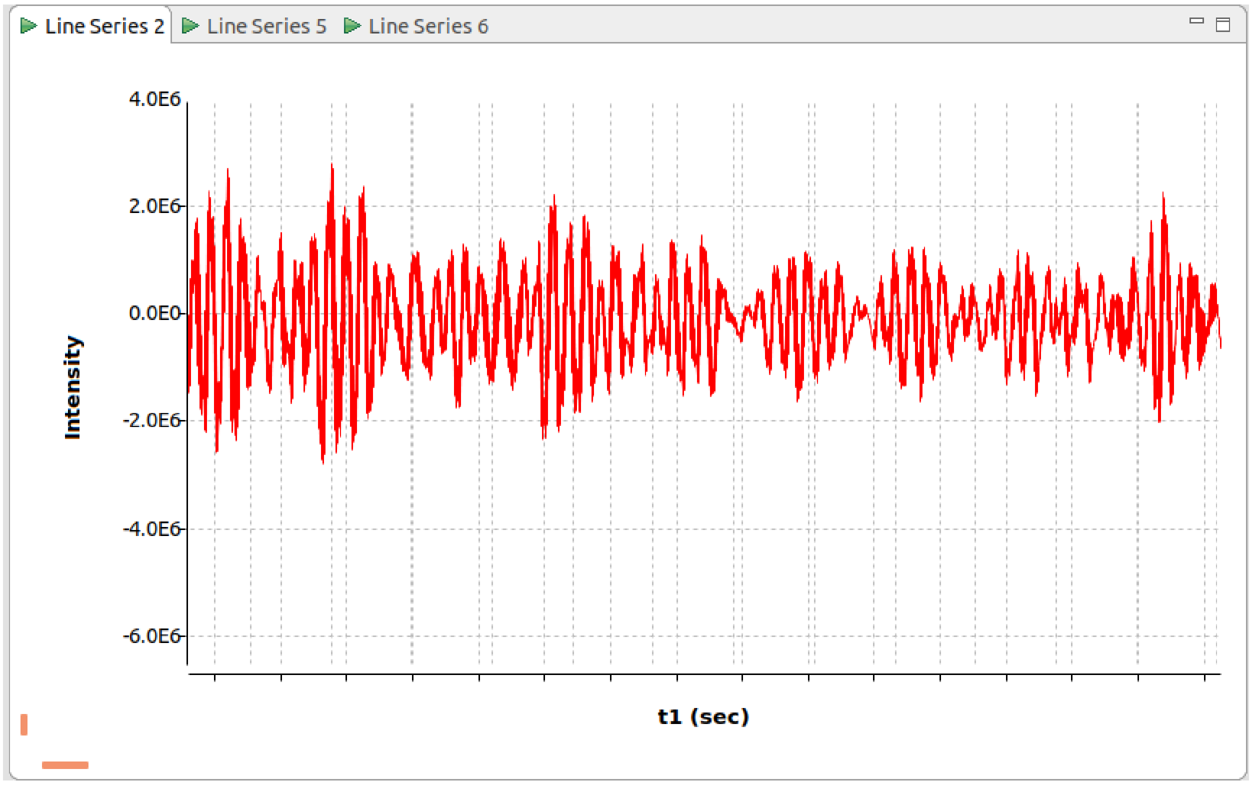
The GTK+ function that enables one to set the Bottom to Top or the Right to Left as increasing is gtk_range_set_inverted (GtkRange* range, gboolean setting); The function has a return type of void (as almost all gtk functions do).

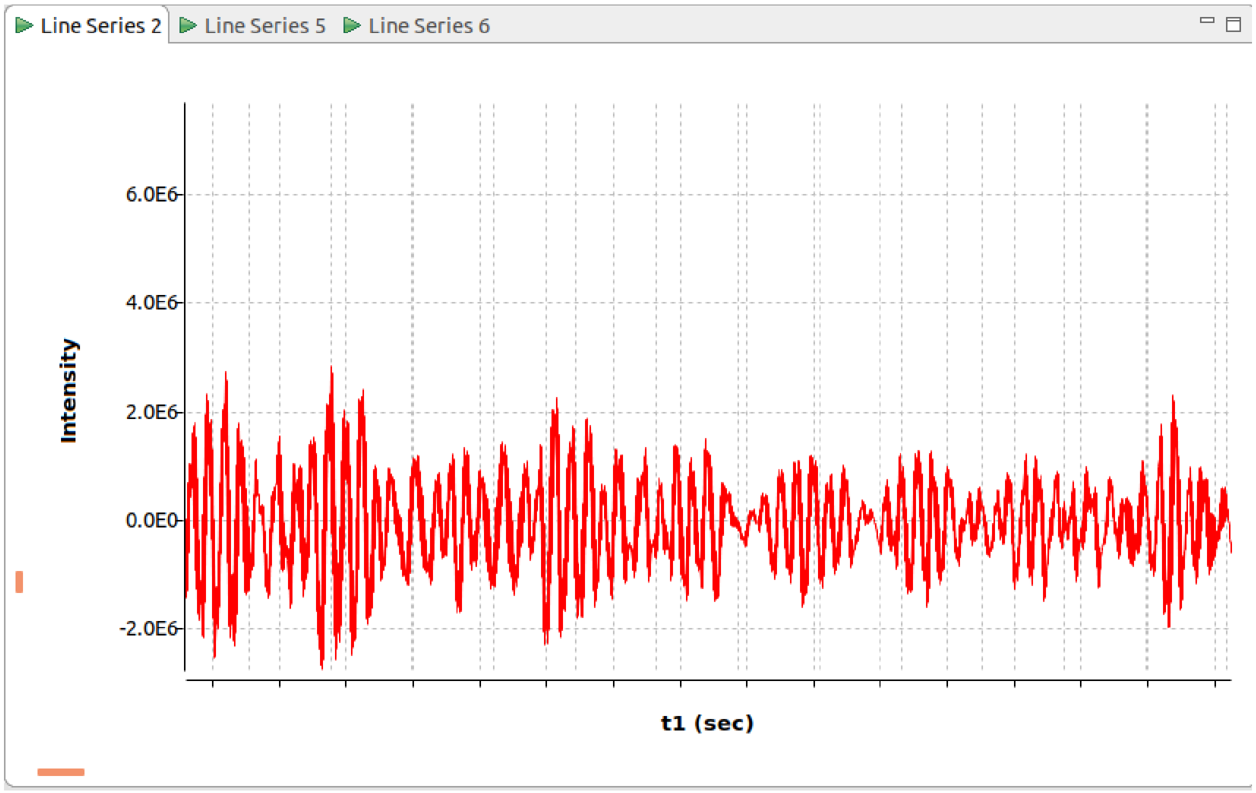
However in SWT, the call to this function is delegated by GTK.gtk_range_set_inverted(long handle, boolean invert); The handle is an identifier that represents a GTKHandle. GTK handles are GTK+ pointers to GTK+ widgets. They are defined in Widget class as long int.

The function is called by Slider class only when the style parameter is set to be HORIZONTAL. The invert parameter passed into the function is set true for RIGHT_TO_LEFT, false for LEFT_TO_RIGHT.

Currently SWT does not have a BOTTOM_TO_TOP and TOP_TO_BOTTOM style constant. Introducing this into SWT Class and modifying the Slider.setOrientation(boolean invert); method to consider this style parameter in the calling of gtk_range_set_inverted() function from the GTK class is what I propose.

The Changes I propose have been tested in the swtchart project and they work in linux. I shall work to include them into Mac and Windows. The following screenshots show the corrected chart in linux.


Line Series 2
Line Series 5
Line Series 6
4.0E6
2.0E6
0.0E0
-2.0E6
-4.0E6
-6.0E6
Intensity
t1 (sec)


Line Series 2
Line Series 5
Line Series 6
6.0E6
4.0E6
2.0E6
0.0E0
-2.0E6
Intensity
t1 (sec)

TimeLine

Community Bonding Period	May 4 - May 31	Bond with all members of Eclipse and know what all projects are under development.Get in touch with experts and learn more about SWT and how it implements the native OS functions.
Coding Phase 1	June 1- June 29	Work on the SWTChart project shall be completed.
	June 1 - June 7	.Complete the Integration of architecture of PieChart into SWTChart.
	June 8 - June 14	Create the required data model and Element Class, start creating and implementing abstract methods to access and modify them.
	June 15 - June 21	Add the dfs algorithm to validate the model and create a base for graph coloring algorithm.
	June 22-June 28	Implement Graph Coloring Algorithm, and the drawing methods to draw the Chart on GC.
Coding Phase 2	July 3 - July 27	Fix bugs if any, and complete Work on Extensions Project.
	July 3- July 5	Add settings to extensions and route them to swtchart.
	July 6- July 19	Add listener to Mouse click and add code to carry out the change in root element pointer.
	July 20 - July 23	Additional settings to be added.
	July 24- July 27	Debugging… and adding example parts.
Coding Phase 3	August 1 - August 23	Solving the Slider Orientation issue
	August 1 - August 4	Debug my work on SWTChart, and extensions.
	August 5 - August 20	Solve the Slider issue for SWTChart and if possible for the Slider as well.
	August 21 - August 23	Debug and remove remaining minor issues in my work. Document and submit code for final evaluation.

Final Evaluation	August 24 - august 31	Submitting the final evaluation of the mentor and saying thanks.

Why Eclipse?

Prior to being a Java developer, I was a web developer. However, I was unable to see the value of the codes I made in the software world. I wanted to work on IDEs or text editors.

It has been about 8 months since I started studying eclipse and learning more about it primarily by reading from “Eclipse Plug-in Development Beginner’s Guide” by Dr. Alex Blewitt and adding breakpoints in the widget and display class of SWT and tracking down the code right from the Main Class. A wonderful tool that the IDE is, now seems as an extension of my thinking.

I never had any other organisation in my mind.

Additional Contributions

Before the ideas list was out, I was working on those issues that I had faced in Eclipse, without any mentor. I shall continue with them post GSoC.

1) Allow the creation of Folders and simple files into the workspace and import them.
2) Eclipse wizard to have the option to create parent folders before creation of a file.
3) Allow deletion of multiple projects from workspace together.

Also in SWTChart

1) Fix the bar series to display bars whose top parts do not lie in the range of the axis.
2) Add histograms as a bar series to the chart.
3) Include the ability to plot simple graphs on the plot area.

I shall try to fix any issue that comes my way.

Contributions to Eclipse

1) Adding the option to the charts to set only required ticks. (https://github.com/eclipse/swtchart/pull/131)
2) Removing redundant code from swtchart. (https://github.com/eclipse/swtchart/pull/130)

About Myself

Myself Himanshu Balasamanta, I am a student of the IIT Varanasi. I am a meditator and a volunteer at the Art Of Living. I take interest in things that stand for something large.I like thinking out of the box and I am interested in research.

Achievements

- *Ranked an all India rank of 2503 in IIT JEE 2018 examination.*
- *Awarded the Prestigious NTSE (National Talent Search Examination Scholarship) in the 10th grade.*
- *Was awarded a National level scholarship in the 4th grade.*
- *Been recognised by the governor of my state for performing numerous Talent Examinations.*
- *Won numerous Scholarships and Talent Searches at State Level.*