STRATEGIES FOR DRAWING DYNAMIC LINES ON MAP

_______________________

A Thesis
Presented to the
Faculty of
San Diego State University

_______________________

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

_______________________

by
Naitik Harshad Doshi
Fall 2015
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Naitik Harshad Doshi:

Strategies for Drawing Dynamic Lines on Map

Carl Eckberg, Chair
Department of Computer Science

Alan Riggins
Department of Computer Science

Mark Dunster
Department of Mathematics & Statistics

Oct 21, 2015
Approval Date
DEDICATION

This thesis is dedicated to my parents, elder brother and professor Carl Eckberg, for their continuous support and encouragement and special thanks to my friends and SDSU faculty without them this would not have been possible. I take this opportunity to express my gratitude towards all of them for their unwavering faith in me.
ABSTRACT OF THE THESIS

Strategies for Drawing Dynamic Lines on Map
by
Naitik Harshad Doshi
Master of Science in Computer Science
San Diego State University, 2015

The MOJO map integrated with a simple user interface and a scroll bar to the right for drawing dynamic lines is the key feature of this tool. The user has to simply enter the coordinates and observe the behavior on the map. The tool helps in drawing line dynamically, which can have user defined color, speed, and type of the line. The line can either be dotted or a solid line based on requirements of the end user. Users for such a tool might have an explanation, like voyage of Columbus, or the building of trans-continental railroad. The dynamic line tool can be coordinated with a slider bar, where advancing the slider bar triggers drawing a new line segment. Thus it is easy to use the tool for drawing a polygonal paths and the application developer can draw Magellan’s world tour with correct geography and rate of progress.

The application is developed for any machine and it is independent of the underlying operating system. The user interface also has a field for determining the speed at which the line has to be drawn. One of the biggest advantages of this feature is that the significance of events can be explained to an audience while the line is being drawn dynamically. The main motive behind developing such a tool is to have an interactive environment for the users, where the users can either belong to the GIS MOJO community or it can be any amateur users. The tool is developed keeping in mind that the end user may or may not have technical skills. The tool currently belongs to a generic category, but if required one can manipulate to make it entertaining and educative at the same time. Graphical display of the end output with the help of a user interface makes it easier to understand and retain any information. The users with more technical background can easily customize the features based on their requirements.
# TABLE OF CONTENTS

| ABSTRACT | ................................................................. | v |
| LIST OF TABLES | ................................................................. | viii |
| LIST OF FIGURES | ................................................................. | ix |
| ACKNOWLEDGEMENTS | ................................................................. | xi |
| CHAPTER |
| 1 INTRODUCTION | ................................................................. | 1 |
| 2 TECHNOLOGIES | ................................................................. | 3 |
| 2.1 Java | ................................................................. | 3 |
| 2.2 Java Swing | ................................................................. | 4 |
| 2.3 Map Objects Java Edition | ................................................................. | 6 |
| 2.4 Eclipse IDE | ................................................................. | 7 |
| 3 REQUIREMENTS | ................................................................. | 8 |
| 3.1 Platform Requirements | ................................................................. | 8 |
| 3.2 Data Requirements | ................................................................. | 9 |
| 3.3 Functional Requirements | ................................................................. | 9 |
| 4 PROTOTYPING | ................................................................. | 12 |
| 5 SOFTWARE DESIGN | ................................................................. | 14 |
| 5.1 High Level Architecture Diagram | ................................................................. | 14 |
| 5.2 Classes and Their Diagram | ................................................................. | 14 |
| 6 ECLIPSE AND MAP OBJECTS CONFIGURATION | ................................................................. | 16 |
| 7 MAPOBJECTS TOOLBAR | ................................................................. | 19 |
| 7.1 Custom | ................................................................. | 19 |
| 8 CUSTOM TOOLBAR | ................................................................. | 22 |
| 8.1 Print | ................................................................. | 22 |
LIST OF TABLES

Table 5.1. Importing Classes and Classes Used for Development .................................................15
Table 8.1. Custom Toolbar ......................................................................................................22
LIST OF FIGURES

PAGE

Figure 1.1. The Slider bar in the application with minimum and maximum values..............5
Figure 1.2. Ladder of Java Swing class. ........................................................................6
Figure 3.1. The User Interface to add different inputs.......................................................10
Figure 3.2. The line drawn using the tool – (x1,y1) – (27,-22) (x2,y2) – (98,54),
           Color-Red, Stroke –N, Speed – 1........................................................................10
Figure 3.3. Info messages being displayed with short description, learn more, see
           video Uri’s. ........................................................................................................11
Figure 4.1. Prototype model/SDLC. ...............................................................................13
Figure 5.1. High-level architecture design........................................................................15
Figure 6.1. The startup screen after installing eclipse. .....................................................17
Figure 6.2. Creation of a new java project.................................................................17
Figure 6.3. Adding JAR files. .......................................................................................18
Figure 6.4. ‘Strategies for drawing dynamic lines tool’ ..................................................18
Figure 7.1. Map Objects toolbar. ................................................................................19
Figure 7.2. Zoom pan toolbar. .....................................................................................20
Figure 7.3. Selection toolbar. .......................................................................................20
Figure 7.4. Java toolbar.............................................................................................21
Figure 8.1. Custom toolbar. .......................................................................................22
Figure 8.2. Print dialog window. ................................................................................23
Figure 8.3. Draw a Dynamic Line window..................................................................23
Figure 8.4. Add layer dialog window. .........................................................................24
Figure 8.5. Pointer icon.............................................................................................25
Figure 8.6. TimeLine tool with Hotlink window.........................................................25
Figure 8.7. Point shape file add dialogue box............................................................26
Figure 8.8. Distance tool .........................................................................................27
Figure 9.1. Selecting the shape files for the current application scope............................29
Figure 9.2. Adding specific shape file into the application. ....................................................29
Figure 9.3. Line drawn based on the slider bar. ...........................................................................30
Figure 9.4. Specifics for the pop up window. ..............................................................................31
Figure 9.5. User Interface with Input values. ..............................................................................31
Figure 9.6. Drawling the line based on inputs provide in previous step. .......................................32
Figure 9.7. Dynamic lines drawn simultaneously using UI tool and slider bar. .........................33
ACKNOWLEDGEMENTS

I would like to extend my heartfelt thanks towards my professor Carl Eckberg for giving me the opportunity to work on this thesis and for his immense support and encouragement at all times.

I would also like to thank my parents, my brother and friends for their belief in me.
CHAPTER 1

INTRODUCTION

The idea behind development of this thesis was to develop a simple user interface based tool to have visual understanding of some events, which occurred over time. The simplest example is to draw a line between two points at a speed chosen by the user who can choose other properties as well. If the application developer needs a polyline, i.e. a segment of lines, this can be hard coded in a loop or the user can be involved by giving the user a sliderbar, where when the slider advances to a given ‘time’, a new line segment is dynamically drawn. A window can be opened to discuss the ‘event’ represented by that segment, etc. As an example, one can trace the path at correct speeds of the travellers in Jules Verve’s Around the World in 80 days, which is better than a static map and could serve as visual Cliff notes.

There are various categories of intended application developer for this idea. By combining the dynamic line interface and the timeline interface, a sixth grader can create an application with

a. A slider bar to pass through a series of events at various times
b. A dynamically drawn line for each event
c. A pop up window for each event with image, text, web link and video

A student of CS537 at San Diego State University, for example, can use just the dynamic line feature to add that feature where needed. This user is much more sophisticated and wants to customize an application, with no software controls or limitations on what he can do. For this user, we might provide a class, and a constructor, with things like color, width and stroke as parameter.

This GIS tool has been developed using Java technologies and Map Objects Java Objects. The project was developed using the Eclipse IDE and the GIS libraries, which are open source libraries. The thesis report is organized into 12 chapters. The first two chapters
give an Introduction about the thesis and the technologies that were used in the development of the tool. Chapters 3 and 4 describe the requirements for this thesis and the prototype followed for it’s successful completion. Chapters 5 and 6 explain about the Software Design of the tool, and the process of setting up Eclipse IDE for development of Java based Map Objects classes. Chapters 7, 8, and 9 explain about the various toolbars used and the added functionalities in detail. Chapter 10 gives an insight to the entire project through screenshots. Chapter 11 and 12 discusses about the obstacles faced during the development cycle and a few possible future enhancements.
CHAPTER 2

TECHNOLOGIES

This chapter discusses different technologies used to accomplish this thesis project. Based on previous experience and careful consideration Dr. Carl Eckberg suggested that Java and MOJO (Map Objects Java Objects) would be the ideal candidates to accomplish the project. This thesis involves a simple user interface with various fields and labels. Java Swing is the ideal candidate in such a scenario. The second reason to use Java was that it doesn’t rely on the underlying and platform and is easily compatible with Map Objects Java Edition. Detailed discussion of these two technologies can be found further in this chapter.

2.1 JAVA

Java is a general-purpose programming language developed by Sun Microsystems. The key features of java are that it is concurrent, class-based and Object-Oriented. It is one of the most popular and widely used programming languages. [1]

It is intended to let application developers write once and run anywhere, fundamentally. Java applications are typically developed in a way that it compiles to byte code, which can run on any machine with any operating system having Java Virtual Machine (JVM) running on it. The main advantage of this feature is that once it is compiled on any machine, it does not require recompilation on a separate machine with a different operating system. Other features, which make Java so popular, are: [2]

**Simple:** Java is easy to write and develop which makes it popular. The Java features are concise and cohesive which makes it easier to learn and develop.

**Secure:** The Java applications run independently on the machine and usually does not bother any other application running on the operating system. It also provides a secure way to access web applications.

**Portable:** Portability is one of the key fundamentals, which were kept in mind while developing the language. It runs on any machine, which has a Java run time system (JVM) and must do
**Object-Oriented**: Java is a class-based language and it is a pure OOP language. It fulfills all the fundamentals of object oriented programming language like inheritance, polymorphism, and abstraction.

**Distributed**: Java was designed with the distributed environment in mind, and can be run over the Internet as well.

**Multithreaded**: Multithreaded is the capability for a program to perform several tasks simultaneously within a program. In Java, multithreaded programming has been smoothly integrated. Multithreading is a necessity in visual and network programming, as used as GIS with frequent refreshing of maps frames.

### 2.2 Java Swing

Java Swings is used to develop the user interface for the tool. Java Swings offers the most effective and flexible graphical user interface features. The Java Swing package offers basic UI component in the form of JComponents. Labels, buttons, text fields, scroll bars are a few simplistic features of Java Swings. These are the following elements of Java Swings along with its significance:

1. **JPanel**: JPanel can be considered as the Swing’s edition of the AWT class Panel. It has the same default plan and flow layout. Other components can be added to JPanel.
2. **JFrame**: The elements in the frame are managed by a contentPane. Adding any element to JFrame requires the use of the contentPane.
3. **JWindow**: This element is directly moved down from the Swing’s edition of Window. JWindow deploys BorderLayout.
4. **JDialog**: The JDialog uses BorderLayout as default. JDialog permits the layer manager as well as the panes of glass. Each of the dialogs is modal thereby referring to the existing thread that is jammed till the time interaction with the user has taken place completely.
5. **JLabel**: This element is used to come up with text labels. AbstractButton which is abstract provides the platform for different elements like JButton.
6. **JTextField**: This element is used to permit the modification of a single line of text. The new attributes comprises of the skill to prove the text lift, right, center etc.
7. **JPasswordField**: This is the element used to suppress the input display. It represents the number of characters with echo character. This actually permits to enter secret and confidential passwords. The echo character is usually the asterisk, *.
8. **JTextArea**: This is very similar to the JTextField, it allows multiple lines of text. It can be used with JScrollPane to have multiple lines of text.
9. JSlider: JSlider allows us to select a value from a list of values between two integers by sliding a knob along a track. JSlider has four important properties:
   a. An orientation
   b. A minimum value
   c. A maximum value
   d. A current value

The JSlider can be displayed horizontally as well as vertically and this can be determined through the orientation. SwingConstants HORIZONTAL and VERTICAL can be used for this purpose.

![Figure 1.1. The Slider bar in the application with minimum and maximum values.](image)

Figure 1.2 describes the hierarchy of the Java Swing Class and its different components.
2.3 

Map Objects Java Edition

Map Objects Java Objects (MOJO) is the core Java application-programming interface used for building GIS application. Map Objects with Java is a Java application, component architecture and also a GIS development tool. MOJO is software package developed and sold by ESRI. MOJO is written in Java and is bundled as a JAR file, which allows the user to extract the required objects. It is a developer’s toolkit; it includes a set of jar files containing pure java components that can be used to develop standalone GIS application.

Map Objects Java Edition is a powerful collection of client and server side components used to build custom, cross platform mapping and GIS applications. It has more precise control over the behavior of the application; a user can make his application from the bottom up, adding only those features that he requires; a GIS class at San Diego State University covers Map Objects in depth and has a published guide named Notes On Map Objects Java Edition by Dr. Carl Eckberg, which was one of the main criteria for choosing Map Objects [3].

Key Features as cited from the Map Objects Java Edition brochure are as follows [3, 4]:

- Includes the ability to combine multiple data sources (local, Internet, and Intranet) to create customized maps
- Has connectivity with ArcIMS (Arc Internet Mapping Service). Includes a wide range of GIS capabilities
• Compatible with many data sources including industry-standard shapefiles, ArcSDE layers, and a variety of image formats such as BMP, TIFF, PNG, JPEG, and GIF.
• Helps to create feature layers from their own custom data sources.

2.4 Eclipse IDE

Eclipse is an integrated development environment, which contains a base workspace and an extensible plugin for customizing the environment the way one want to. Eclipse itself is written in Java and is most popularly used for writing Java applications. An IDE normally consists of a source code editor; build automation tools and a debugger [5].

There are several other IDE’s, which are fairly popular among the Java developer communities. Eclipse, IntelliJ, Netbeans are a few such examples and the choice depends on the familiarity with the IDE based on previous experience.
CHAPTER 3

REQUIREMENTS

The ‘Strategies for Drawing Dynamic Lines’ can be used for learning about events that are spread over a period of time by using the slider bar functionality and have a visual display of how events occurred in the chronological order. The tool can be used through the user interface ‘Draw Dynamic Lines’ which provides a user interface where eventually a line with specific type, color and width would be drawn. The tool can be used for making an educative and interactive learning system for the users. The thesis/project has been developed under the guidance of Dr. Carl Eckberg from the San Diego State University Computer Science Department and he was the major contributor to determine the requirements for this thesis. The requirements gathered have been classified as follows:

- Platform requirements
- Data requirements
- Functional requirements

3.1 Platform Requirements

Platform requirements consist of mainly two things, the development language and the appropriate GIS solution. Dr. Carl Eckberg played a massive role in determining both of them. The project was developed entirely on MAC OSX, which has a powerful hardware and software capabilities along with other advantages.

Map Objects Java Edition helps build applications that perform a variety of geography-based display, query and data retrieval activities at the client, presentation, Web, and server tiers. Java was used as the development language with Map Objects Java Edition as the GIS solution.

Thus they allow representing the geographical data on maps, which make the tool more interactive and informative [3, 5].
Java as Programming Language
Java Swing for developing the User Interface
Map Objects Java Edition as the GIS API.

3.2 DATA REQUIREMENTS

The data requirements refer to the data that should be collected to be represented in the tool. The display of data is covered in the Functional Requirements. Dr. Carl Eckberg helped in identifying important information about TimeLine tools and points of interests.

All the data required for this thesis has been collected under the supervision of Dr. Carl Eckberg. The data includes all the shape files in the first place, the geographic coordinates of the places to be shown, the significance of that place or event, Images related to it, time when the event took place, related reason and consequences. The data requirements also involved a concrete example to show. The goal was to refine an earlier version of “mapping” the history of pi.

3.3 FUNCTIONAL REQUIREMENTS

Functional requirements of GIS tool for learning about ‘Strategies for drawing Dynamic line on Map’ were gathered from Prof. Carl Eckberg. The critical part of the thesis is to create an interface to allow people unskilled in Java and Map Objects to create timelines and display event information, using dynamically drawn lines.

The software comes up with an interactive screen so that the user can specify things like:

a. X and Y co-ordinates of two points to specify what line is to be drawn
b. Color of the line which would be drawn
c. Whether the user wants the line to be stroked or not
d. Width of the line
e. Description of the event, External links and Images
f. The software should come up with a selected map as the base map. The User Interface (Figure 3.1) should draw the line above that map (Figure 3.2).

Thus an “amateur” user can specify the line he wants and also information about what that line represents. (Figure 3.3)
Figure 3.1. The User Interface to add different inputs.

Figure 3.2. The line drawn using the tool – (x1,y1) – (27,-22) (x2,y2) – (98,54), Color-Red, Stroke –N, Speed – 1.
Figure 3.3. Info messages being displayed with short description, learn more, see video Uri’s.
CHAPTER 4

PROTOTYPING

The most important part in any application is to have the prototype. The prototype is the blueprint of the work that needs to be done, technically prototyping is done to understand the requirements instead of freezing the requirements before a design or coding can proceed. It describes the flow and phases of the work. It is essential because it gives a clear idea to the developer as well as the end user of how the application is going to be, and what exactly it is supposed to do. Also, if there are any changes at any point of time, it can be well adjusted in the prototype model. It basically follows an agile software development life cycle, which includes:

**Requirement Analysis** – This is the first step of the prototyping model where in we decide all the requirements for the entire thesis/project. It includes data requirement, functional requirement and platform requirement.

**Design & Architecture** – This step includes analyzing the design of the application in broader terms, which is like creating a initial draft about the features and operations in detail. It also includes the screen layouts.

**Development/Implementation** – It includes the actual coding part where in all the requirements are implemented and appropriate code is written to get the desired result.

**Verification/Review** – This step includes getting the application developed so far approved by the end user, in this case by Dr. Carl Eckberg to assure that all the requirements have been successfully met. In this step after getting the reviews, essential changes are made. These first four steps are iterated until the major players are happy with the application.

**Maintenance** – This step pertains to the help/ maintenance required once the application is deployed but in this case it is not of much importance as it is out of the scope of this thesis.

Figure 4.1 [5] of the software development lifecycle which shows the important steps included in it.
There were modifications to the initial prototype according to the feedback received by the thesis supervisor. All the required features and functionality were implemented and reviewed within the given time frame.
CHAPTER 5

SOFTWARE DESIGN

This chapter includes the high-level software design/architecture and the class diagrams along with it. The software design can be developed if and only if all the data and functional requirements are fulfilled. While developing the software design, keeping it easy and easy to install was the main goal.

5.1 HIGH LEVEL ARCHITECTURE DIAGRAM

Figure 5.1 shows a high-level architecture diagram for GIS tool for learning about TimeLine tool. It explains how the project looks at a higher level. The architecture is broadly divided into three different levels:

- User Interface: By clicking the tool icon from the tab a User Interface is displayed. The screen has text boxes for inputs of X and Y co ordinates between which the line has to be drawn.
- Map Objects and Java classes: The functionality ad the action of all the above text inputs, buttons and selection list along with custom tool bar and default GIS tool bars are defined in here.
- Resource files: The default main screen user interface uses the above classes for accessing the different resources files such as shape files, images and help documentation.

5.2 CLASSES AND THEIR DIAGRAM

The classes diagram helps in streamlining the overall code and its structure; it also provides the developer a stencil to use for code development. A class diagram shows the set of variable and functions present in the class. It also explains the interdependency of classes used for the development. Table 5.1 shows the base class and its imported classes.
Table 5.1. Importing Classes and Classes Used for Development

<table>
<thead>
<tr>
<th>Package</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import javax.swing</td>
<td>JButtons, JPanel, JSplitPane, JToolBar, ImageIcon, JMenuBar, JMenuItem, Abstract Table Model, Table Column</td>
</tr>
<tr>
<td>Import com.esri. Mo2.ui.tb</td>
<td>Zoom Pan Tool Bar, Selection Tool Bar, Project Tool Bar</td>
</tr>
<tr>
<td>Import com.esri. Mo2.file.shp</td>
<td>Shapefile folder, Shape file writer</td>
</tr>
<tr>
<td>Import com.esri. Mo2.map.dpy</td>
<td>Layerset, Featured Layer, Base Feature Layer</td>
</tr>
</tbody>
</table>
CHAPTER 6

ECLIPSE AND MAP OBJECTS CONFIGURATION

This project was developed in java platform, edited and compiled in Eclipse IDE. Eclipse IDE was chosen for the development of the GIS tool as it provides an ability to integrate with the Map Objects Java edition (Integrated Development Environment). Eclipse provides a rich Java development environment and plug-in system. Library JAR files are to be added to the project to run the Map Objects classes.

Steps to configure Eclipse IDE and Map Objects:

2. Install Eclipse IDE.
3. Install Map Objects Java Objects given by Dr. Carl Eckberg [3].
4. Install Java Software development kit.
5. Launch Eclipse IDE from Start → Programs → Eclipse by double clicking the eclipse icon. Figure 6.1 shows the process for step 4.
6. Create a new java project. Go to File → New → Java Project, it will open a window. See Figure 6.2.
7. Click on libraries tab, click on Add Libraries, now select User Libraries and click next. See Figure 6.3.
8. It will open a preference window click User Libraries and then click new and enter the name of the library. See Figure 6.4.
9. Now click on Add Jars button, navigate to the folder where MOJO is installed.
10. Select esri_mo20, esri_mo20res, and esri_mo20img the JAR files.
11. Make sure that the newly created library checkbox is selected.
12. Next step is to attach Javadocs to the jar files. Expand one of the JAR file and click on Javadoc location and then click edit.
13. In the Javadoc location path, navigate to the Javadoc folder in ESRI documentation directory and then click OK.
14. Now repeat step 13 for all other JAR files.
Figure 6.1. The startup screen after installing eclipse.

Figure 6.2. Creation of a new java project.
Figure 6.3. Adding JAR files.

Figure 6.4. ‘Strategies for drawing dynamic lines tool’.
CHAPTER 7

MAPOBJECTS TOOLBAR

Multiple beans are provided by Map Objects Java Edition, which the user can drag and drop onto eclipse workspace to provide basic functionality (see Figure 7.1). The existing toolbar includes functionalities like zoom in, zoom out, pan, and identify, etc. These tools are essential functionalities for visual display. This chapter explains the integration of this tool in the GIS tool and basic functionality of each tool.

Following toolbars will be discussed:

- Zoom pan toolbar
- Selection toolbar
- Java toolbar

![Figure 7.1. Map Objects toolbar.](image)

7.1 CUSTOM

The ZoomPanToolBar (Figure 7.2) provides user to change the map visual in a number of ways. It provides facilities like zooming, panning, etc. The actions performed by the various icons in the Toolbar are explained below:

- Previous Extent – Zooms to the map's previous extent that is stored in the history.
- Next Extent – Zooms to the map's next extent stored in the history.
- Zoom To Active Layer – Zooms the map to all selected features in the selected layer.
- Zoom To Full Extent – Zooms the map to full extent of all layers within the map.
- Zoom In – It provides a function to click or drag a rectangle on the map to zoom in (To view the selected area bigger).
- Zoom Out – It provides a function to click or drag a rectangle on the map to zoom out (To view the selected area smaller).
• Pan – Provides a tool for dragging the map to a new location without altering the zoom level.

![Image](image1.png)

**Figure 7.2. Zoom pan toolbar.**

• Pan In One Direction – Provides a tool for dragging the map in one direction (North, South, East, West).

• Identify – Identifies the selected layer and display the features of that layer in an attribute table.

### 7.2 COM.ESRI.MO2.UI.TB.SELECTION TOOLBAR

The Selection Toolbar (Figure 7.3) provides functions that perform future selection bases upon Attribute or query.

• Search – Opens a dialog for locating features based on a predefined “stored query”.

• Find – It opens a dialog for locating features whose attributes contain an end user provided string.

• Query Builder – Opens a dialog for locating features based on a query that an end user gives.

• Select Features – It provides a tool for selecting specified areas by different means like rectangle, polygon, line & circle in the map.

• Clear All Selection

• Buffer – It allows to construct a polygon around the currently selected features.

• Attributes – It displays the attributes of the layer selected on the TOC.

![Image](image2.png)

**Figure 7.3. Selection toolbar.**

### 7.3 JAVA TOOLBAR

The Project Toolbar (Figure 7.4) allows user to add a layer, print a map image to file, etc. The actions performed by the various options in the project toolbar are as follows [6]:

• Print Map – Use this option to print the map for the selected layer(s) in the TOC.

• Add Layer – Use this option to add layer in the TOC, which shows different map.

• Pointer – This is use to resume the cursor from previous selection.
• Distance Tool – It allows user to find the distance between two points on map in miles as well as kilometers.

![Java toolbar](image)

**Figure 7.4. Java toolbar.**

• XY Tool – It allows the user to add a layer of points using a .CSV file having the latitudes and longitudes.

• Hot Link – It provides functionality where in user can get more information about the locations visible on the map like the link to wiki, images, etc.

This is the critical toolbar, specific to this application. Considerable detail about this tool bar is provided in Chapter 8.
CHAPTER 8

CUSTOM TOOLBAR

Map Objects provides Zoom pan tool bar, Selection tool bar and Java Tool bar have been provided to the users. Along with these tool bars a custom tool bar is created to give more features to the map and more interaction to the users (see Figure 8.1). This tool bar is created in Java.

Figure 8.1. Custom toolbar.

The functionality of each icon in the custom tool bar is discussed below in the order of the icon occurrence in the custom tool bar (see Table 8.1).

Table 8.1. Custom Toolbar

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>🗿️</td>
<td>Print</td>
<td>Print com.esri.mo2.ui.bean.Print</td>
</tr>
<tr>
<td>🖌️</td>
<td>Draw Line</td>
<td>DrawLine.java</td>
</tr>
<tr>
<td>🚦</td>
<td>Add Layer</td>
<td>Com.esri.mo2.ui.tb.LayerToolBar</td>
</tr>
<tr>
<td>🔎</td>
<td>Pointer</td>
<td>Arrow.java</td>
</tr>
<tr>
<td>🔍</td>
<td>Distance Tool</td>
<td>DistanceTool.java</td>
</tr>
<tr>
<td>⚡</td>
<td>Hotlink</td>
<td>Hotlink.java</td>
</tr>
</tbody>
</table>

8.1 PRINT

The Print button provides the functionality to print the map. On Click this button opens a dialog where the printer and other specifications can be selected as shown in Figure 8.2.
8.2 DRAW DYNAMIC LINE

The draw dynamic line tool allows the user to draw a dynamic line based on the co-ordinates, color, speed and type (dotted or solid). Refer Figure 8.3
8.3 ADD LAYER

This tool allows the user to add a new layer to the table of contents. In Figure 8.4 the pictorial display of how the layer can be added to the map is shown. Once the user clicks on the Add Layer button a Dialog box pops up. The user can browse to the location where the shape file or image file or ARC file exists and select it to add to the current map.

- These are the steps to add layer in detail:
- Click on the plus sign from the custom toolbar, or “ADD LAYER” from the file menu.
- A new window will appear, browse to the location of file to be added.
- Select the shape file (.shp file).
- Click Ok.
- New Layer would be added to the existing map.
- Steps to remove layer:
- Click on the “REMOVE LAYER” from the file menu.
- By default it will delete the current selected layer.
- To delete a particular layer, first select the layer and then click on the remove

Figure 8.4. Add layer dialog window.
8.4 POINTER

Whenever the application is launched the default tool selected is the Pointer tool. The pointer tool is specially used in cases when the user is working on some other tool and wants to switch back to the default tool. The icon for getting back to the default tool is shown in Figure 8.5.

Figure 8.5. Pointer icon.

8.5 HOTLINK

Hotlink provides more detailed information about the selected point. The hotlink opens up a dialog box, which may contain information like description of the point selected, image associated with the point and external links. Hotlink can also be configured to open an html file. If desired the hotlink can add similar functionalities for line layers and polygon layers. The term "hotlink" was used in the other bolt image here, distributed ESRI products like Arc View. The image showing the functionality of hotlink is in Figure 8.6.

Figure 8.6. TimeLine tool with Hotlink window.
8.6 XY TOOL

It gives the user the ability to display new locations on the map by using a comma separated value file (.csv) or a text file. When the user clicks on this icon a pop up window will show up allowing the user to choose the text file or csv file having the latitudes and longitudes of the location to be displayed. It displays the data points immediately on the map. A potential use of this tool in team assignments by an instructor using this tool, where not tied by a time line (Figure 8.7).

![Image of XY Tool]

Figure 8.7. Point shape file add dialogue box.

8.7 DISTANCE TOOL

It gives the user the ability find out the distance between any two points on the map in miles as well as in kilometers. The display is at the bottom of the map in the status bar. See Figure 8.8.
Figure 8.8. Distance tool.
CHAPTER 9

SCREENSHOTS

This chapter contains the screenshots from the entire project. Screenshots seem naïve but are largely dependent on the target audience. The target audience for this tool can be technically empowered application developer with a background in Computer Science or a sixth grader with almost no technical background. This tool can be used widely in the Geography class to explain chronological events; such an audience would have very less knowledge about coding, integrated development environment, etc. Having a visual flow of how the tool works would prove essential for such an audience. Screenshots help simplify things and with the help of screenshots the end user can expect next step of the tool. We would be discussing all the different screens in the order in which the tool is expected to be used by the user. Figure 9.1 shows the default shape files (countries.shp, pi.shp) already loaded in the application. For demo purpose of the tool, we would be having screenshots for drawing the line, which demonstrates chronological events in which history of value pi (3.14) was developed

Figure 9.2 explains how the end user can have its own pre-defined points on the map on specific co-ordinates and the region on which the application would be used. Different types of shape files for different regions can be easily found online based on requirements. The pi.shp and country.shp in the demo case was covered in the requirements section.
Figure 9.1. Selecting the shape files for the current application scope.

Figure 9.2. Adding specific shape file into the application.
Once the shape files and the region are selected we can move onto actual drawing of the line, which would depend on the scrolling of the slider bar on the right hand side. The scrolling of mouse on right hand side of the screen signifies the events and helps draw line on the canvas. Figure 9.3 demonstrates the actual line drawn on the canvas.

The application also gives the ability to have a pop up window when the line begins. The pop up window can be used in different ways since it gives the application the ability to have a couple of links which can be redirected to any YouTube video, Wikipedia page, or any specific user defined page. The pop up window also has the ability to accumulate a picture along with a brief description of the event. Figure 9.4 demonstrates the above-explained feature.

The user can also use the tool to draw the line between specific points on the map. The tool is called ‘Mod’ with the light blue colored pencil icon and can be found on top of the screen. Clicking on the tool provides a Graphical User Interface for entering the X1, Y1, X2, Y2 co ordinates, the color, speed and type (dotted or solid) line on the map. Figure 9.5 shows different parameters for the user interface with input values.
Figure 9.4. Specifics for the pop up window.

Figure 9.5. User Interface with Input values.
Once all the specifics for drawing the line have been added on the User Interface is provided, and when the DrawLine button is clicked, the line is actually drawn on the canvas. Figure 9.6 depicts the line drawn on the basis of the input provided in the previous step.

![Figure 9.6. Drawing the line based on inputs provide in previous step.](image)

We can also have both the features incorporated together. This would have overlapping line drawn on the canvas and can be used to simultaneously understand two events. Figure 9.7 explains simultaneous drawing of lines on canvas.
Figure 9.7. Dynamic lines drawn simultaneously using UI tool and slider bar.
CHAPTER 10

CONCLUSION, OBSTACLES AND TESTING

Working on this project was an exhilarating and challenging task. The major obstacle was to develop the algorithm in a way that it takes the user defined values at run time and plot the dynamic lines based on those input values. For instance it was challenging to draw lines, which are stroked compared to the regular solid line. Another obstacle was to draw lines which at a particular speed. This feature of the tool was very crucial because it would be largely used in visual understanding of chronological events. The design of interfaces is always tricky since it is difficult to anticipate all desired applications and make sure they are feasible. Testing of the tool by the people in computer science, geography and other fields were helpful. In similar fashion it is hard to decide how much help information to provide in the tool, since users can have wide range of skills.
CHAPTER 11

FUTURE ENHANCEMENTS

The core reason behind developing this tool in Java was to make it platform independent and have all the functionalities in the tool. Extensibility criteria was always kept in mind while developing the tool, making the application executable on a smartphone can be the immediate requirement in some of the cases. Having same application work with similar user interface on a smartphone can be one of the enhancements for this tool. Another enhancement, which can be done, is having animations on the map while the line is being plotted on the map.

Making the slider bar work with dynamic inputs can prove to have stronger appeal to the application developers. One of the most important enhancements to this tool can be having the line draw based on the speed at which the mouse is scrolling on the slider bar. Making this feature dynamic and having the line drawn at the run time user defined pace would make it all the way more useful for the end user. One of the simplest use cases for such a tool can be the chronological order in which Olympic events occurred. Plotting all the countries as points through a shape file and then drawing the line based on occurrence of events. Using the ‘dynamic lines on map’ tool can be very informative for such sequence of events.
REFERENCES


