HISTORY OF MUGHAL EMPIRE, GIS SUPPORT FOR WORLD

HISTORY

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
Harpal S. Randhawa
Spring 2013
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Harpal S Randhawa:

History of Mughal Empire, GIS Support for World History

____________________________
Carl Eckberg, Chair
Department of Computer Science

____________________________
William Root
Department of Computer Science

____________________________
Farid Mahdavi
Department of History

March 20, 2013
Approval Date
ABSTRACT OF THE THESIS

History of Mughal Empire, GIS Support for World History
by
Harpal S. Randhawa
Master of Science in Computer Science
San Diego State University, 2013

This thesis focuses on creating a GIS tool for the Department of History showcasing the history of the Mughal Empire and lifetime of Mughal rulers in Indian subcontinent. The users have the ability to click on particular events and see more details on respective web pages. This approach presents an effective learning tool for students by providing an interactive environment. Using a computer simulation is much more engaging and better aligned to the interests of modern students. Moreover, the students also have the ability to customize the application environment to suit their individual interests. By utilizing modern computer technology to present an important subject in an engaging and interesting fashion, this work could become an effective learning tool in the modern classroom not just for students, but also the instructors who can handle bigger class sizes with greater ease.

The application has been made in the Java programming language and utilizes the Map Object Java Edition (a.k.a MOJO) API, which is provided free by ESRI. Using MOJO, GIS related features can be easily incorporated into the application. A key design challenge in any learning tool is to have the interface as simple and intuitive as possible while not compromising on the feature set. Subsequently, every attempt has been made to present a rich depth of subject knowledge through a lucid yet powerful interface.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>iv</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 History of GIS</td>
<td>1</td>
</tr>
<tr>
<td>1.3 About the Application</td>
<td>2</td>
</tr>
<tr>
<td>2 A BRIEF REVIEW ON TECHNOLOGY</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Java</td>
<td>4</td>
</tr>
<tr>
<td>2.2 History of Java</td>
<td>4</td>
</tr>
<tr>
<td>2.1.2 Major Updates</td>
<td>5</td>
</tr>
<tr>
<td>2.2.2 Features of Java</td>
<td>6</td>
</tr>
<tr>
<td>2.3 MOJO</td>
<td>7</td>
</tr>
<tr>
<td>3 REQUIREMENTS</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Initial Requirements</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Intermediate Requirements</td>
<td>9</td>
</tr>
<tr>
<td>3.3 Final Functional Requirements</td>
<td>10</td>
</tr>
<tr>
<td>4 SOFTWARE DESIGN AND CODING</td>
<td>11</td>
</tr>
<tr>
<td>4.1 Basic Architecture</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Coding Snippets</td>
<td>12</td>
</tr>
<tr>
<td>5 NETBEANS IDE AND MOJO CONFIGURATION</td>
<td>17</td>
</tr>
<tr>
<td>5.1 Basic Information</td>
<td>17</td>
</tr>
<tr>
<td>5.2 Step by Step Project Creation</td>
<td>17</td>
</tr>
<tr>
<td>6 UNDERSTANDING MAPOBJECTS TOOLBAR</td>
<td>23</td>
</tr>
<tr>
<td>6.1 Toolbars</td>
<td>23</td>
</tr>
</tbody>
</table>
6.2 Zoom Pan Toolbar ..................................................................................................................23
6.3 Mojo Selection Toolbar ........................................................................................................23
7  HISTORY OF MUGHAL EMPIRE IN INDIA ............................................................................25
8  SCREEN SHOTS ....................................................................................................................28
9  FUTURE ENHANCEMENTS ..................................................................................................37
REFERENCES ................................................................................................................................38
LIST OF TABLES

PAGE

Table 6.1. Zoom Panel Toolbar .................................................................24
Table 6.2. Selection Tools ........................................................................24
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Application on launch</td>
<td>3</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Basic software architecture</td>
<td>11</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Select Java application from right pane and then click next button</td>
<td>18</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Give project a name</td>
<td>19</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Creating a new library</td>
<td>20</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Selecting classpath</td>
<td>21</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Selecting JAR files</td>
<td>22</td>
</tr>
<tr>
<td>Figure 6.1</td>
<td>Zoom panel toolbar</td>
<td>23</td>
</tr>
<tr>
<td>Figure 6.2</td>
<td>Selection toolbar</td>
<td>24</td>
</tr>
<tr>
<td>Figure 7.1</td>
<td>Mughal Empire map</td>
<td>25</td>
</tr>
<tr>
<td>Figure 7.2</td>
<td>Mughal Empire flag</td>
<td>26</td>
</tr>
<tr>
<td>Figure 8.1</td>
<td>Mughal Empire tool on launch</td>
<td>28</td>
</tr>
<tr>
<td>Figure 8.2</td>
<td>Adding a layer</td>
<td>29</td>
</tr>
<tr>
<td>Figure 8.3</td>
<td>India Shapefile added</td>
<td>30</td>
</tr>
<tr>
<td>Figure 8.4</td>
<td>Adding data to map</td>
<td>31</td>
</tr>
<tr>
<td>Figure 8.5</td>
<td>Data file is added</td>
<td>32</td>
</tr>
<tr>
<td>Figure 8.6</td>
<td>Editing and applying legend</td>
<td>33</td>
</tr>
<tr>
<td>Figure 8.7</td>
<td>Map with labels</td>
<td>34</td>
</tr>
<tr>
<td>Figure 8.8</td>
<td>Hotlinks on map</td>
<td>34</td>
</tr>
<tr>
<td>Figure 8.9</td>
<td>Map zoom to active layer</td>
<td>35</td>
</tr>
<tr>
<td>Figure 8.10</td>
<td>Attribute table</td>
<td>36</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to thank Dr. Carl Eckberg for his support, guidance and motivation from time to time during each phase of the project.

I would also like to thank Professor William A. Root and for being my committee members and for his help and co-operation.

Finally, I would like to thank Professor Farid Mahdavifar for giving me this opportunity to work on this project for the Department of History.
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Methods of learning have been changed and supplemented through the past decade, from book reading, to including electronic learning. With the boom in the internet and the increase in use of the computers, electronic learning has emerged as one of the most prominent and easy mediums for learning. Learning done at computers, usually connected through internet, given us the opportunity to learn and read anytime, anywhere is referred to as E-learning. This type of learning provides lot of benefits like improved efficiency, easy and fast access, convenience and flexibility to the students. Most higher education institutions offer online courses where students rarely attend face-to-face on campus educational facilities, because they study online.

It is defined as a planned teaching/learning experience that uses a wide spectrum of technologies, mainly internet or computer-based, to reach learners.

The goal of this thesis is to provide students better understanding of the history of Mughal empire. By utilizing modern computer technology to present an important subject in an engaging and interesting fashion, this work could become an effective learning tool in the modern classroom not just for students, but also the instructors who can handle bigger class sizes with greater ease. Students, who want to do more than read books, now will learn about the History of Mughal Empire in a multimedia manner which could stimulate more interest in learning.

1.2 HISTORY OF GIS

A geographic information system (GIS) integrates hardware, software, and data for capturing, managing, analyzing, and displaying all forms of geographical information. GIS allows us to view, understand, question, interpret, and visualize data in many ways that reveal relationships, patterns, and trends in the form of maps, globes, reports, and charts by merging cartography, statistical analysis and database technology.
The world’s first GIS based software was developed in Canada by Dr. Roger Tomlinson an English geographer. It was known as Canada Geographic Information System (CGIS) [1]. It could store, analyze and manipulate data for Canada Land Inventory (CLI). In 1964 Howard T. Fisher developed a laboratory and by 1970 developed some systems like ODYSSEY, GRID and SYMAP [1].

A detailed history of GIS is not well understood, because GIS technology evolved through multiple parallel but separate applications across numerous disciplines. The early 1980s saw the coming of ESRI and they developed a second generation approach to GIS software by incorporating data into database structures and attribute tables. With cross platform GIS software, interoperable applications could also be created.

Since its beginning, GIS technology has proven to be indispensable for a wide range of applications; to name a few, modern land administration, archaeology, urban planning, marketing, logistics, aviation, agriculture, forestry and geographic history all involve heavy use of GIS based systems.

In addition to localized operations, modern GIS systems are typically network enabled to cater to an increased demand for dynamic information, following the advent of the World Wide Web. Unfortunately, due to its complicated nature, there are only a few companies which develop GIS based system software; ESRI is one of them.

1.3 ABOUT THE APPLICATION

The GIS tool is designed for the Department of History. The main idea is to show the lifetimes of Mughal rulers and the history of the Mughal Empire on a timeline basis on a World Map. This tool is highly motivating for students as they will learn through pictures, videos and links to internet sources. It’s a modern way of learning by making use of the data represented by linking it to various geographical features on a map.

The application is based on Java and Map Objects Java Edition (MOJO, which is an ESRI product). MOJO is used for creating client side or server side mapping that can be used to make cross platform independent applications. The IDE used for developing this tool is NetBeans 6.7.1. MapObjects JAVA StandardEdition is an easy-to-use suite of JAVA developer components that you can use to build custom, cross-platform GIS applications or
applets. It provides a robust, JAVA-based API that lets you design applications to perform a wide variety of geographic-based display, query, and data retrieval activities.

Map Objects is a collection of GIS and mapping components developed by ESRI, a world leader in GIS technology. Map Objects supports shape files, a common ESRI format.

The rest of the thesis document will have more chapters which will go into further detailing about the project. The first few chapters introduce the thesis and the tools which have been used to develop this application, and the technology, the code that is used to build the application. Later chapters describe the requirements gathered from the customer, future enhancements and also some of the difficulties that have been faced during the building phase of the project. In the end it includes future enhancements that can be added using this tool to further attract students towards learning and education.

Figure 1.1 shows the initial window after starting the application.

Figure 1.1. Application on launch.
CHAPTER 2

A BRIEF REVIEW ON TECHNOLOGY

This chapter discusses about the programming language, the environment and the building blocks of the application. MapObjects JAVA edition and the JAVA programming language have been used to develop this project. Java can build platform independent applications better than most other computer languages and MapObjects comes with two flavors, one of which is designed to work with JAVA, the other with Visual Basic.

2.1 JAVA

JAVA is a high-level programming language which is developed by Sun Microsystems with a number of features that make the language well suited for use on the World Wide Web. It is an object-oriented language which is similar to C++, but it eliminates some language features that cause common programming errors.

JAVA source code files are compiled into an intermediate code called bytecode, which can then be executed by a JAVA interpreter. Compiled JAVA code can run on most computers because JAVA interpreters and runtime environments, known as JAVA Virtual Machines, exist for most operating systems like UNIX, the Macintosh OS, and Windows. Bytecode can also be converted directly into machine language instructions by a just-in-time compiler (JIT).

2.2 HISTORY OF JAVA

James Gosling initiated the JAVA language project in June 1991. The language, initially called Oak, also known as Green, ended up as JAVA. Gosling aimed to implement a virtual machine and a language that had a familiar C/C++ style of notation. Sun Microsystems released the first public implementation as JAVA 1.0 in 1995. It promised “write once, run anywhere” providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run JAVA applets within
web pages, and JAVA quickly became popular. With the advent of JAVA 2, new versions had multipleconfigurations built for different types of platforms like J2EE, J2ME and J2SE.

In 1997, Sun Microsystems tried to formalize JAVA, but it soon withdrew from the process. JAVA remains a de facto standard, controlled through the JAVA Community process. At one time, Sun made most of its JAVA implementations available without charge. Sun generated revenue from JAVA through the selling of licenses for specialized products such as the JAVA Enterprise System. Sun distinguishes between its Software DevelopmentKit (SDK) and Runtime Environment (JRE) (a subset of the SDK); the primary distinction involves the JREs lack of the compiler, utility programs, and header files.

On November 13, 2006, Sun released much of JAVA as open source software under the terms of the GNU General Public License. On May 8, 2007, Sun finished the process, making all of JAVAs core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright. More recently Oracle purchased Sun and thus Java.

2.1.2 Major Updates

The JAVA language had gone through several changes since JAVA Development Kit (JDK 1.0) was released in 1996, as well as numerous additions of classes and packages to the standard library.

- **JDK 1.1** was released on February 19, 1997. Major additions included an extensive retooling of the AWT event model, inner classes added to the language, JAVABeans and JDBC.

- **J2SE 1.2** (Codename Playground) was released on December 8, 1998. This and subsequent releases through J2SE 5.0 were rebranded JAVA 2 and the version name "J2SE" replaced JDK to distinguish the base platform from J2EE and J2ME. Major additions included reflection, a Collections framework, JAVA IDL, and the integration of the Swing graphical API into the core classes.

- **J2SE 1.3** (Codename Kestrel) was released on May 8, 2000. Notable changes included the bundling of the HotSpot JVM, JAVA Sound, JAVA Naming and Directory Interface (JNDI) and JAVA Platform Debugger Architecture (JPDA).

- **J2SE 1.4** (Codename Merlin) was released on February 6, 2002. This was the first release of the JAVA platform developed under the JAVA Community Process as JSR 59. Major changes included regular expressions modeled after Perl, exception chaining, an integrated XML parser, and JAVA Web Start.
• **J2SE 5.0** (Codename *Tiger*) was released on September 30, 2004. Originally numbered 1.5, which is still used as the internal version number. It has a number of significant new language features including the for-each loop, generics, and var-args.5

• **JAVA SE 6** (Codename *Mustang*) the current version released on December 11, 2006, is bundled with a database manager, facilitates the use of scripting languages with the JVM and has Visual Basic language support. As of this version, Sun replaced the name "J2SE" with **JAVA SE** and dropped the ".0" from the version number. Other major changes include support for pluggable annotations, lots of GUI improvements, including native UI enhancements to support the look and feel of Windows Vista, and improvements to the JVM Tool Interface for better monitoring and troubleshooting.

• **JAVA SE 7** (Codename *Dolphin*) The Dolphin Project started in August 2006, with release estimated in September 2010.

2.2.2 Features of Java

Java’s features can be summarized with its features as follows:

• *Object Oriented Programming Language*: Java is object oriented and its development model results in frequent releases of improved versions.

• *Platform Independent*: Java can build platform independent applications better than most of other computer languages.

• *Secure*: Java is used in many networking environments. Java can help to create virus-free applications.

• *Interpreted*: Code is compiled to bytecodes that are interpreted by the Java Virtual Machine (JVM). Java programs follow a write once run everywhere paradigm. All one needs is a JRE on the targeted machine.

• *Robust*: Much exception handling is required, there is strong type checking (that is, all data must be declared an explicit type), and local variables are initialized. All variables have a well-defined scope so that there is no misuse of them and these results in a robust application. Java follows a model which eliminates the possibility of overwriting memory and corrupting of data.

• *Better to use than C/C++ for highly interactive applications*: Java eliminates many programming related errors because there is no concept of pointers in java which are usually difficult to maintain and cause too many errors. Also it checks on array limit index checking which would otherwise cause an error if memory is referenced which has not been assigned to an array element.

• *Automatic Memory Management*: Automatic garbage collection – allocating and deallocating memory by JVM.
• **Dynamic Binding:** Automatically selecting appropriate methods at run time is called dynamic binding. Once libraries are compiled there is no need to recompile code that uses classes in those libraries.

• **Performance based:** Java is used in situations where higher performance is required. The bytecodes can be converted to machine code on run time for particular system application is running. Just in time compilation is used for higher performance.

• **Multi-Threading:** Lightweight processes called threads are used for real time behavior. It’s great for multimedia displays.

• **Good for Networking Applications:** Java can be used for designing protocol based networking applications.

• **Easy Deployment:** by executable jar files

### 2.3 MOJO

MOJO (Map Objects Java Objects) is a collection of client and server side software component provided by ESRI for creating GIS based java applications. It can be used to build applications that incorporate GIS which involves the display and analysis of location based data, maps being the most common example. ESRI’s two largest packages are ArcView 3.3 and ArcGIS 8.0/9.0. Both packages provide many features. MOJO helps you build applications that perform geography-based display, query & data retrieval activities at the client and server side. Anyone who creates a GIS application can use any of the provided features and can decide to not use others. There are complicated or fine features too for people who know or want to use extensive UI based application.

Key features of the MapObjects JAVA Edition as mentioned in an ESRI brochure are as follows [2]:

• **Ability to combine multiple data sources:** By using applications created with MapObjects JAVA Edition, users can combine local data with Internet and Intranet data to create their own customized maps and easily integrate these with existing application architectures. Some of the supported data formats are image formats (PNG, GIF, JPG), CAD, shapefiles, ArcSDE layers etc.

• **Wide range of GIS capabilities:** With applications built using MOJO, users can perform activities such as labeling map features, panning and zooming through multiple map layers, measuring distances, creating layouts etc.

• **Ability to deploy applications over the web:** You can easily distribute your applications over an Internet through browser-hosted applets or simplify Web delivery of your application with the use of JAVA Web Start technology, the industry-standard launching mechanism for distributing JAVA applications over the Web.
Helpful Tools for Building a User Interface: By using the rich swing components included with MOJO, you can quickly build applications that include functional toolbars; dynamic symbol control, query dialogs, overview and insert maps, and intelligent legends that make your custom applications easy to use and even easier to develop. Some of these GUI objects are ESRI created from JAVA swing objects.

The advantages of relatively small GIS packages like MOJO include lower cost, greater portability, simpler development, and much more opportunity for customization.

Also, Dr. Carl Eckberg’s book on MapObjects JAVA Edition is a great source to understand and write the application. This book discussed MOJO and all programming help you need to design viable GIS applications [3].
CHAPTER 3

REQUIREMENTS

The tool was designed for the Department of World history. The requirements were to make an interactive software tool for teaching the history of the Mughal Empire in India. Below is the list of requirements which were understood since the project’s initiation to the project’s final phase:

3.1 INITIAL REQUIREMENTS

Following are the requirements provided by the Department of World History in the initial phase of the project:

1. Collect the information about the History of Mughal Empire from different sources such as books in the library, online and faculty of the history department.

2. To display the information in the chronological order of the major events happened in the history of Mughal Empire such as gaining power by the new emperors and major events that led to the spread of the Mughal Empire.

3. Data should be represented in the form of the geographical history on maps.

4. User should be able to customize the application, so that users can make changes and addition to the information later, if needed.

5. Project should be made for the users who are instructors of the department of history and don’t have expertise or any experience using GIS tools.

3.2 INTERMEDIATE REQUIREMENTS

The following requirements are provided after the initiation of the project in the second phase of requirements gathering.

1. Tool should be created in such a way that new layers, photos, text, links related to Mughal Empire can be added easily when required by department faculty.

2. All graphical features and fonts should be neat and clear. There should be no broken links.

3. Tools should have rich multimedia capabilities like picture gallery, video and audio gallery.

4. All the labels used should be clearly visible and should not overlap when too many points are shown on map.
5. User should have capability to add CSV manually by a click on a button and providing the file’s path by browsing to the file location.

6. The Language used in the application should be easy to read.

3.3 FINAL FUNCTIONAL REQUIREMENTS

The following requirements are provided in the final phase of functional requirement gathering.

1. All dialog windows should have a cancel button so that user can close that the current window and go back to the previous window.

2. User should have the option to print the map with the data show on the map.

3. There should be a button to zoom in and zoom out the map. The selected area of map should zoom in and zoom out by the mouse clicks on the tool.

4. Pan functionality should be included in the map, which user can use to pan the map.

5. User should have the ability to identify the country or area occupied by the Mughal Empire by just clicking on the area.

6. User should be able to select and add the layers to the map as required.

7. There should be a tool to zoom to full extent.

8. Status bar should show the location of user cursor on the map in terms of the longitude.

9. Hot links: There should be hot links on the map and user should be able to click on the hot links on the map. When user clicks on the hot links a window should pop up with a description of the location where that hot link is located on the map. Window should display the information about the Mughal empire rule in that location on the map.

10. Selecting the layer: After selecting a layer, there should be a button. It should display the attributes from the layer.

11. Deselecting a layer: There should be a button to deselect the selected layer.

12. There should be help menu so that user can learn tool quickly by using the help documents. There should be a contact us information in this help menu to reach someone, if user have any questions which are not answered in the help.

13. JAVA: Tool should be developed in java which is simple and platform independent so that it will be easy to use in any environment.

14. Tool should be packed in JAR file, so that it will be easy to download and use on any system.
CHAPTER 4

SOFTWARE DESIGN AND CODING

This chapter focuses on the basic software architecture followed by code snippets which are critical to the application.

4.1 BASIC ARCHITECTURE

Figure 4.1 shows the basic software architecture that has been followed in this project. The application will use the MOJO APIs provided by ESRI. MOJO has access to all the shapefiles, dBASE files and CSV files which in turn would be used by the application to render data on display. Shapefile is a vector file containing points, lines and areas. A more detailed description of the shapefile will be covered in a separate chapter.

Figure 4.1. Basic software architecture.
4.2 CODING SNIPPETS

This is the code from some of the java files. These java files form the important part of the application from adding, selecting, promoting and demoting a layer and drawing points on the map etc.

1. Adding the layer to the map. This code was written to add a new layer which is a shapefile or image on the map.

   public class AddLyrDialog extends JDialog {
       Map map;
       ActionListener lis;
       JButton ok = new JButton("OK");
       JButton cancel = new JButton("Cancel");
       JPanel panel1 = new JPanel();
       com.esri.mo2.ui.bean.CustomDatasetEditor cus = new com.esri.mo2.ui.bean.
       CustomDatasetEditor();
       AddLyrDialog() throws IOException {
           setBounds(50,50,520,430);
           setTitle("Select a theme/layer");
           addWindowListener(new WindowAdapter() {
               public void windowClosing(WindowEvent e) {
                   setVisible(false);
               }
           });
           lis = new ActionListener() {
               public void actionPerformed(ActionEvent ae) {
                   Object source = ae.getSource();
                   if (source == cancel)
                       setVisible(false);
                   else {
                       try {
                           setVisible(false);
                           map.getLayerset().addLayer(cus.getLayer());
                       }
                   }
               }
           };
   }
map.redraw();
if (MughalEmpire.stb.getSelectedLayers() != null)
MughalEmpire.promoteitem.setEnabled(true);

} catch(IOException e){}

ok.addActionListener(lis);
cancel.addActionListener(lis);
getContentPane().add(cus, BorderLayout.CENTER);
panel1.add(ok);
panel1.add(cancel);
getContentPane().add(panel1, BorderLayout.SOUTH);
}

public void setMap(com.esri.mo2.ui.bean.Map map1){
    map = map1;
}

2. Getting the Points on Map and giving circle shape: Code below shows how to bring a map on display and add points on it. The points used in the application are circles but we can also draw squares or rectangles.

class DrawPoint extends Tool {
    Map map = MughalEmpire.map;
    SimpleMarkerSymbols ms = new SimpleMarkerSymbol();
    Point pt = new Point();
    AcetateLayer acetLayer = new AcetateLayer() {
        public void paintComponent(java.awt.Graphics g) {
            if (pt != null) {
                java.awt.Graphics2D g2d = (java.awt.Graphics2D)g;
                g2d.setTransform(
map.getWorldToPixelTransform().toAffine();
g2d.setClip(map.getExtent());
sms.draw(pt,g2d,"");
}
);

publicDrawPoint () {
    sms.setType(SimpleMarkerSymbol.CIRCLE_MARKER);
sms.setWidth(6);
sms.setSymbolColor(Color.red);
map.add(acetLayer);
}

public void mouseClicked(MouseEvent me) {
    pt = map.transformPixelToWorld(me.getX(),me.getY());
    acetLayer.repaint();
}

3. Promoting and Demoting a layer: The following code shows how the selected layer can be promoted or demoted amongst other layers.

layercontrollis = new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        String source = ae.getActionCommand();
        System.out.println("active index");
        if (source == "Promote Selected Layer")
            map.getLayerset().moveLayer(activeLayerIndex,++activeLayerIndex);
        else
            map.getLayerset().moveLayer(activeLayerIndex,--activeLayerIndex);
        enableDisableButtons();
        map.redraw();
    }
};

4. Browser Launch: The following code shows how a browser can be launched and how it handles all different kinds of browsers.

public class BrowserLaunch {
    private static final String errMsg = "Error attempting to launch web browser";
public static void openURL(String url) {
    String osName = System.getProperty("os.name");
    try {
        if (osName.startsWith("Mac OS")) {
            Class fileMgr = Class.forName("com.apple.eio.FileManager");
            Method openURL = fileMgr.getMethod("openURL", new Class[] {String.class});
            openURL.invoke(null, new Object[] {url});
        }
        else if (osName.startsWith("Windows"))
            Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + url);
        else {
            String[] browsers = {"firefox", "opera", "konqueror", "epiphany", "mozilla", "netscape"};
            String browser = null;
            for (int count = 0; count < browsers.length && browser == null; count++)
                if (Runtime.getRuntime().exec(new String[] {"hich", browsers[count]}).waitFor() == 0)
                    browser = browsers[count];
            if (browser == null)
                throw new Exception("Could not find web browser");
            else
                Runtime.getRuntime().exec(new String[] {browser, url});
        }
    }
    catch (Exception e) {
        JOptionPane.showMessageDialog(null, errMsg + ":\n" + e.getLocalizedMessage());
    }
}
5. Attribute Table: Following is the code for the attribute table and how rows and columns are manipulated.

```java
public class AttrTab extends JDialog {
    JPanel panel1 = new JPanel();
    com.esri.mo2.map.dpy.Layer layer = MughalEmpire.activeLayer;
    JTable jTable = new JTable(new MyTableModel());
    JScrollPane scroll = new JScrollPane(jTable);
    public AttrTab() throws IOException {
        setBounds(70, 70, 450, 350);
        setTitle("Attribute Table");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                setVisible(false);
            }
        });
        scroll.setHorizontalScrollBarPolicy(
                JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        jTable.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
        TableColumntc = null;
        int numCols = jTable.getColumnCount();
        for (int j = 0; j < numCols; j++) {
            tc = jTable.getColumnModel().getColumn(j);
            tc.setMinWidth(50);
        }
        getContentPane().add(scroll, BorderLayout.CENTER);
    }
    public int getColumnCount() {
        return jTable.getColumnCount();
    }
    public int getRowCount() {
        return jTable.getRowCount();
    }
    public String getColumnName(int colIndx) {
        return jTable.getColumnName(colIndx);
    }
    public Object getValueAt(int row, int col) {
        return jTable.getValueAt(row, col);
    }
```
CHAPTER 5

NETBEANS IDE AND MOJO CONFIGURATION

5.1 BASIC INFORMATION

NetBeans IDE (Integrated Development Environment) is an integrated development environment used to develop all sorts of applications, primarily Java ranging from desktop applications to web applications. This IDE can also be used to develop other applications, like PHP, C/C++ and HTML 5. This is a free open source IDE available online on the NetBeans.org website. NetBeans core IDE has following modules [4]:

1. NetBeans Profiler: This is a tool for the monitoring the Java applications. This tool does The code optimization can be done automatically with NetBeans profiler and various errors to which a program is prone to like memory leaks can be also checked automatically.

2. GUI design tool: Formerly known as project Matisse. The design tool can add a boost for UI based applications as it supports dragging and dropping various GUI components to prototype and design Swing GUIs.

3. NetBeans JavaScript Editor: This can be used for syntax highlighting, refactoring, code completion of all java code methods and class instances, browser compatibility checking, all basic editor features, all CSS editing features, etc. It Provides extended support for JavaScript, Ajax and CSS.

Following is what is required to create the History of Mughal Empire in India GIS Project:

1. Install Java (SE) Development kit 7.0 from java site [5].
2. Install NetBeans IDE 6.0.1 from the site [6].
3. MapObjects Java Edition Installation CD can be obtained from Dr. Carl Eckberg.
4. MOJO is integrated with NetBeans using MOJO Libraries.

5.2 STEP BY STEP PROJECT CREATION

The following steps are required to set up the project for development activities:

1. Create a Java Application first as seen in Figures 5.1 and 5.2.
2. Create a new Library (as seen in Figures 5.3 and 5.4):
   a. Right click on the Libraries folder and then click on add library.
3. Select Classpath and Add JAR files (as seen in Figures 5.4 and 5.5):
   a. Select Classpath Tab
   b. Click on Add JAR/Folder button
   c. Navigate to folder C:ESRI/MOJ23/lib to select the Jar files
Figure 5.2. Give project a name.
Figure 5.3. Creating a new library.
Figure 5.4. Selecting classpath.
Figure 5.5. Selecting JAR files.
CHAPTER 6

UNDERSTANDING MAPOBJECTS TOOLBAR

6.1 TOOLBARS

MOJO provides various toolbars to be used when the actual application runs and we can perform various tasks with them. The toolbars which are used for the Mughal Empire in India GIS tool Project provides functions like Pan, Pan to one direction, Identify, Hotlinks, Query Builder, Zooming in and out, select feature, deselecting the feature, find tool which can find anything in all the layers added, attribute tool, clear all tool and buffer. Now we will explain one by one all the Toolbars which are used in the application [3].

6.2 ZOOM PAN TOOLBAR

Figure 6.1 shows the zoom panel toolbar which provides capabilities to zoom in or zoom out the presented map. First icon is Previous extent tool, if you click on this tool it will take you to the previous extent of the map, second tool is Next extent tool which displays next extent of map, Zoom to active layer tool displays the active layer of the map, zoom to full extent display the full map, pan tool is use to move the displayed map in order to see the desire area, Next tool is a pan in one direction which on click gives option to pan north, south, east and west and the last tool is identity tool which helps to identify a specific area. Table 6.1 provides a brief description of the tools provided by the ZoomPan toolbar.

Figure 6.1. Zoom panel toolbar.

6.3 MOJO SELECTION TOOLBAR

Figure 6.2 illustrates how the toolbar helps the users to create queries on the selected features, selects and deselect features, attributes and adding the hotlinks layer. Table 6.2 is a brief description of the tools provided by the Selection toolbar.
### Table 6.1. Zoom Panel Toolbar

<table>
<thead>
<tr>
<th>Tool</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous Extent</td>
<td>Zooms to previous extent stored in extent history</td>
</tr>
<tr>
<td>Next Extent</td>
<td>Zooms to next extent stored in extent history</td>
</tr>
<tr>
<td>Zoom to Active Layer</td>
<td>Zooms the map to all selected features in selected layer</td>
</tr>
<tr>
<td>Zoom to Full Extent</td>
<td>Zooms to extent all layers within map</td>
</tr>
<tr>
<td>Zoom In</td>
<td>Helps in zooming in</td>
</tr>
<tr>
<td>Zoom Out</td>
<td>Helps in zooming out</td>
</tr>
<tr>
<td>Pan</td>
<td>Helps in moving to any direction without having to zoom</td>
</tr>
<tr>
<td>Pan One Direction</td>
<td>Pans to any of one direction- North, South, East or West</td>
</tr>
<tr>
<td>Identify</td>
<td>Identifies any area in active layer</td>
</tr>
</tbody>
</table>

![Selection toolbar](image)

**Figure 6.2. Selection toolbar.**

### Table 6.2. Selection Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>Opens a dialog for locating features whose attributes contain an end-user provided string</td>
</tr>
<tr>
<td>Search</td>
<td>Opens a dialog for locating features based on a predefined &quot;stored query&quot;</td>
</tr>
<tr>
<td>Query</td>
<td>Opens a dialog for locating features based on a query than an end user constructs</td>
</tr>
<tr>
<td>Select</td>
<td>Tool for selecting features by rubber banding a shape in the map</td>
</tr>
<tr>
<td>Deselect</td>
<td>Tool for deselecting features.</td>
</tr>
<tr>
<td>Buffer</td>
<td>Opens a dialog for constructing a buffer polygon around currently selected features</td>
</tr>
<tr>
<td>Attributes</td>
<td>Tool to display attributes of currently selected features</td>
</tr>
<tr>
<td>Add layer</td>
<td>XY tool is used to add the points or data layer to the map.</td>
</tr>
</tbody>
</table>
CHAPTER 7

HISTORY OF MUGHAL EMPIRE IN INDIA

This chapter is dedicated to the history of the Mughal Empire in India. The Mughal Empire in India lasted from 1526 to 1528. The Mughal Dynasty was established by able Muslim rulers who came from the present day Uzbekistan. Figure 7.1 [7] shows the map of the Mughal Empire and Figure 7.2 [7] shows the flag of Mughal Empire. The Mughal rule in India saw the country being united as one single nation and being administered under one single powerful ruler. During the Mughal period art and architecture flourished and many beautiful monuments like the Taj Mahal were constructed, which is still one of the seven wonders on the earth. The rulers of the Mughal Dynasty were skillful warriors and admirers of art as well.

![Image](Figure 7.1. Mughal Empire map. Source: WIKIPEDIA, Mughal Empire. Wikipedia, http://en.wikipedia.org/wiki/Mughal_Empire, accessed July 2010, n.d.)

Zahirud-din Muhammad Babur was the founder of Mughal Dynasty, he was born in Feb 23 1483 in Fergana, which is now in Uzbekistan. He was driven out of Fargana after his
father died and wandered around for two years. In 1504 he came to Afghanistan and occupied Kabul and became the Padshah of Kabul; later he invaded India five times and eventually occupied certain parts of it and started Mughal Dynasty in India. After the death of Babur Humayun became the Emperor but he was less effective ruler and his rule ended in 1540. He restored his rule again in 1555 and ruled India until 1556.

Jalaluddin Mohammed Akbar was the most effective and greatest ruler of the Mughal Dynasty. Akbar greatly expanded the empire and is regarded as the most illustrious ruler of the Mughal dynasty as he set up the empire’s various institutions. He was a founder of a syncretic religion based on Hinduism and Islam and he married a Rajput princess who was Hindu. Akbar constructed Lahore Fort which is still a famous monument in Pakistan. Akbar’s rule ended in 1605. Akbar’s son Nuruddin Mohammed Shah Jahangir ruled the Mughal Empire from 1605 to 1627. After the Jahangir came, time for when Mughal art and Architecture reached their zenith under the rule of Shahabuddin Mohammed Shah Jahan. He constructed the Taj Mahal a symbol of love in the memory of her beloved wife Mumtaz Mahal after her death in 1631. He was later imprisoned by his own son Mohiuddin Mohammed Aurangzeb Alamgir in 1658.

Aurangzeb was the sixth Mughal Emperor of the Mughal Empire. He extended the Mughal Empire to its farthest boundaries, but his reign was harsh and marked by revolts.
He was more conservative in behavior which led to more revolts and he brought back the tax on people who didn’t follow Islam. His conquests expanded the empire to its greatest extent, incorporating much of modern India. He wrote the Quran twice in his own handwriting. He stretched the Empire to the extent that it faced a lot of challenges after his death, and the decline of Mughal Empire started. He died in Feb 2012 and Bahadur Shah took the rule. Bahadur shah was the first of the Mughal Emperors to preside over a steady and severe decline in the territories under the empire’s control and military power. After him emperors became a progressively insignificant figurehead and the end of the Mughal Dynasty started in India. The last known Mughal Dynasty major figure was Bahadur Shah Zafar, who was deposed by the British and died in 1862.
CHAPTER 8

SCREEN SHOTS

This chapter shows and demonstrates the capabilities of the tool. This chapter contains the screenshots for different capabilities of the tool and demonstration of those capabilities. Capabilities which demonstrated here include adding a layer, adding CSV files, using different tools in the toolbar in Figures 8.1 through 8.10.

Figure 8.1. Mughal Empire tool on launch.

1. Initializing the Mughal Empire GIS tool
   a. Click on the Mughal_Empire.jar file.
   b. This will open the Mughal Empire GIS tool GUI, it’s displaying the map of south Asia which includes India, Pakistan and Afghanistan.
2. Adding a layer
   a. You can add more shapfiles in this map.
   b. Click on the File and then Add layer, it will open a browser where you can add Shapefiles, imagefiles, ArchIMS and ArcSDE.
   c. Navigate to the folder Shapfiles and select the shape file you want to add and click open.
   d. India states shapefile is added.

3. Add a point file with hotlinks to data and web pages.
   a. Click on the XY tool, it will open the browser.
   b. Add Introduction_Mughal_Empire.txt to see point and data for Mughal Empire, add the Mughal_Emperors.txt file to see the points and data related to Mughal emperors in India, and Battles_of_Mughal_Empire.txt to see the points and data for battles fought by Mughal Empire.
   c. Mughal_Emperors.txt points file is added in the screenshot shown in Figure 8.5.
4. Editing and applying the legend
   a. Select the Points layer, Click on the File and then Legend Editor
   b. You can change the color, shape of the points and add labels to the points or the shapes in the selected layers.
   c. Edited with legend editor

5. Hotlinks
   a. Select the Mughal_XY layer and click the lightning bolt in order to activate the hotlinks.
   b. Click on the any point on the map, it will open a webpage which will contain the information or facts about Mughal empire or emperors.
   c. You can click on the links on the webpage for more information about the Mughal emperor or empire.

6. Zoom to active layer tool
   a. Click on this tool, it will zoom the map to the active layer.
Figure 8.4. Adding data to map.

7. Open Attribute Table
   a. If you open this menu icon from Theme Menu by clicking on Toc
   b. First click on any layer on the Toc which contains two. If you click on Mughal layer and open Theme menu and select open attribute table you will get the entire information stored about those points
Figure 8.5. Data file is added.
Figure 8.6. Editing and applying legend.
Figure 8.7. Map with labels.

Figure 8.8. Hotlinks on map.
Figure 8.9. Map zoom to active layer.
Figure 8.10. Attribute table.
CHAPTER 9

FUTURE ENHANCEMENTS

The requirements envisioned in the beginning of the project have been accomplished and verified. The requirements were the immediate need of the world history students of the Department of History to provide information regarding the history of the Mughal Empire in a descriptive geographical context. This project provides further opportunities to refine the learning experience, to name a few:

1. Create an iPhone app for this tool, or an android app
2. More topics related to History of other Dynasties or Empire can be added to this tool. This tool can be made to study the history of Indian subcontinent in more details.
3. Localization of this project in Hindi or other language can be done in order to serve more students.
4. Music or games can be added in the project to make it more fun for the students.
5. Feedback section can be added in the future in order to improve the application functionality according to users need.
REFERENCES


