DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ARCHAEOLOGICAL IMAGING AND VISUALIZATION

by
Özcan KINALI

October, 2009
İZMİR

ARCHAEOLOGICAL IMAGING AND VISUALIZATION

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering, Computer Engineering Program

by
Özcan KINALI

October, 2009
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**ARCHAEOLOGICAL IMAGING AND VISUALIZATION**" completed by **ÖZCAN KINALI** under supervision of **ASSIST. PROF. DR. ADİL ALPKOÇAK** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Assist. Prof. Dr. Adil ALPKOÇAK
Supervisor

.....

(Jury Member)

.....

(Jury Member)

Prof.Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to thank my supervisor Assist. Prof. Dr. Adil ALPKOÇAK for his kind suggestions, guidance and all the other valuable supports.

I would also thank to Prof. Dr. Mahmut DRAHOR and Res. Assist. Meriç BERGE for their comments, suggestions and feedback.

I have special thanks to my family for all their support, patience, comments, guidance and everything rest than these during my thesis study; especially to my mother Huri, my father Hikmet and my brother Orhan.

Özcan KINALI

ARCHAEOLOGICAL IMAGING AND VISUALIZATION

ABSTRACT

The main goal of this thesis is to develop a sophisticated visualization application that can be better used by geophysicists, geologists and archaeologists to explore and understand the spatial visualization of data collected from the historical site. Application has functionalities such as rotating, zooming and cutting the 3D visualization of data from the historical site. Additionally, it is also possible to make some geophysical values transparent for getting a better understanding about the historical site. Application also has capability such as taking snap-shot and recording video abilities.

Visualization process can be repeated by using different interpolation and colormap settings. Maximum geophysical data limit value can be defined and changed in application, so users can adjust the insignificant data values and get start the visualization process. System was developed by using Java programming language in Eclipse Platform and VTK (Visualization ToolKit) is used to generate three dimensional graphics.

Keywords: Imaging, Visualization, Archaeology, VTK, Interpolation, Histogram Equalization, 3D Graphics, IDW.

ARKEOLOJİK GÖRÜNTÜLEME VE GÖRSELLEŞTİRME

ÖZ

Bu tezin ana amacı, gelişmiş bir görüntüleme uygulaması geliştirerek, jeofizikçilerin, jeologların ve arkeologların tarihsel alanları daha iyi araştırmasını ve anlamasını sağlamaktır. Uygulama, üç boyutlu olarak görselleştirilen tarihi alanlardan elde edilen verinin döndürülmesi, yakınlaştırılması ve bazı alanların ilgili bölgelerinin kesilmesi gibi işlevselliklere sahiptir. Bunun yanında, tarihi alanı daha iyi anlayabilmek için bazı jeofizik değerlerinin saydam yapılması da sağlanabilmektedir. Uygulama ayrıca, snap-shot alma ve video kaydetme gibi yeteneklere de sahiptir.

Görselleştirme süreci farklı interpolasyon ve colormap ayarları kullanarak tekrarlanabilir. Uygulamada, maksimum jeofiziksel veri limiti tanımlanabilir ve değiştirilebilir, böylece kullanıcılar önemsiz veri değerlerini belirleyerek, görselleştirme sürecini başlatabilir. Sistem, Eclipse platformunda Java programlama dili kullanılarak geliştirilmiştir ve üç boyutlu grafikler elde edebilmek için de VTK (Visualization ToolKit) kullanılmıştır.

Anahtar Sözcükler: Görüntüleme, Görselleştirme, Arkeoloji, VTK, Interpolasyon, Histogram Eşitleme, 3D Grafikler, IDW.

CONTENTS

Page

CHAPTER ONE

INTRODUCTION

Visualization is a part of our everyday life. From weather maps to the exciting 3D graphics of the entertainment industry, examples of visualization abound. Visualization engages the primary human sensory apparatus, vision, as well as the processing power of the human mind. The result is a simple, effective and high-bandwidth medium for communicating complex information. Different terminology is used to describe visualization.

Scientific visualization is the formal name given to the field in computer science that encompasses user interface, data representation and processing algorithms, visual representations and other sensory presentation such as sound or touch. Data visualization is another phrase used to describe visualization. Although rigorous definitions have not been accepted data visualization generally connotes application of statistical methods outside the realm of visualization. On the other hand, data visualization implies applications beyond the sciences and engineering. In this thesis, the term data visualization is used instead of the more specific scientific visualization.

Computer imaging techniques have become an important diagnostic tool in the practice of archeology and modern medicine. These techniques includes such as XRAY Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). In archeology, these techniques use a sampling or data acquisition process to capture information about internal structure. The information is in the form of slice planes or cross sectional images of the structure.

Archaeology is a destructive process in which accurate and detailed recording of a historical site is imperative. As a historical site is exposed, documentation is required in order to recreate and understand the site in context. Archaeologists use various

kinds of written documentation, sketches, diagrams, and photographs to document the physical state of a historical site while it is being excavated. While there are many standards or guidelines for recording the state of the historical site during excavation, their main focus is to record and archive the data, rather than visualize it.

A 3D imaging and visualization application is developed by conclusion of this M.Sc thesis project. The application can assist archaeologists by building rich, geometrically and photometrically accurate 3D visualization of data collected from a historical site. The modeling effort begins with data acquisition (images, range scans, coordinate and geophysical values).

This thesis concentrated on developing a useful visualization application that can be better used by researchers to explore and understand the historical site. The application is developed on Eclipse platform by using Java programming language. Graphical User Interfaces (GUI's) are implemented by using Java's AWT and SWING libraries. Docking windows are builded by using InfoNode Docking Windows (IDW) tool. IDW is a Java Swing framework for building docking windows. Imaging and Visualization processes are created by using Visualization Toolkit (VTK).

This thesis is divided into 5 chapters. Next chapter, Chapter Two, explains the studies and the algorithms used in the thesis. Chapter Three introduces the packages (IDW, VTK) used in the application VTK, explains same basic terms about VTK and gives information about the VTK Visualization Pipeline Architecture. Chapter Four explains the application and gives some information about the used datasets, diagrams and screen shots about the application. Finally, the last chapter presents the conclusion of the thesis and gives information about possible future studies.

CHAPTER TWO

BACKGROUND AND PRELIMINARIES

This project focuses the visualization process, but before beginning of the visualization process, some operations are followed in two processes. These two processes are, Histogram Equalization and Interpolation processes.

Source file of the historical site includes unrefined data. Source file consists coordinate and geophysical data values. If the histogram of these geophysical values is generated, it can be seen that some geophysical values are quite much bigger or smaller than the common such values. This situation is not prefered because when the visaulization process is completed and the historical site is shown to the user in the screen, colors of the historical site might be quite different from the expected colors. To eliminate these values and to contrast the histogram of whole data values, histogram equalization method is used.

After equalized geophysical values are generated, Interpolation Process begins. In this process, new points are generated by using the known data points. Interpolation can be succeeded in different ways. After the interpolation process, new data points are obtained. These points are used in the visaulization process.

2.1 Histogram equalization

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. Histogram equalization usually increases the local contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast without affecting the global contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

For example, if a simple image is analyzed, some definitions like below can be approached:

Let n_i be the number of occurrences of color or data level i. The probability of an occurrence of a point value of level i in the image is:

$$p(x_i) = \frac{n_i}{n}, i \in 0, ..., L-1 \quad (1)$$

L is the total number of geophysical data levels in the image, n is the total number of points in the image, and p is the image's histogram, normalized to [0,1].

If the *cumulative distribution function* (cdf) defined as "c", corresponding to p, the definition in below can be written:

$$c(i) = \sum_{j=0}^{i} p(x_j) \quad (2)$$

Cdf also known as the image's accumulated normalized histogram.

If a transformation of the form

$$y = T(x) \quad (3)$$

is created, that will produce a level y for each level x in the original image, such that the *cdf* of y will be linearized across the value range. The transformation can be defined by:

$$y_i = T(x_i) = c(i) \quad (4)$$

T maps the levels into the domain of 0..1. In order to map the values back into their original domain, the following simple transformation needs to be applied on the result:

$$y_i' = y_i \cdot (\max - \min) + \min \quad (5)$$

2.1.1 A Simple 8x8 2D Image

Figure 2.1 shows a 2D image. It has 64 geophysical (or electrical) data values.

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Figure 2.1 A simple 8x8 2D image

The histogram for the image in Figure 2.1 is shown in Table 2.1. Data or electrical resistivity values that have a zero count are excluded for the sake of brevity.

Table 2.1 Histogram of the image

Value	Count	Value	Count	Value	Count	Value	Count	Value	Count
52	1	64	2	72	1	85	2	113	1
55	3	65	3	73	2	87	1	122	1
58	2	66	2	75	1	88	1	126	1
59	3	67	1	76	1	90	1	144	1
60	1	68	5	77	1	94	1	154	1
61	4	69	3	78	1	104	2		
62	1	70	4	79	2	106	1		
63	2	71	2	83	1	109	1		

The cdf is shown in Table 2.2. Again, the data that do not contribute to an increase the cdf are excluded for brevity.

Table 2.2 Cdf of the image

Value	cdf	Value	cdf	Value	cdf	Value	cdf	Value	cdf
52	1	64	19	72	40	85	51	113	60
55	4	65	22	73	42	87	52	122	61
58	6	66	24	75	43	88	53	126	62
59	9	67	25	76	44	90	54	144	63
60	10	68	30	77	45	94	55	154	64
61	14	69	33	78	46	104	57		
62	15	70	37	79	48	106	58		
63	17	71	39	83	49	109	59		

This *cdf* shows that the minimum value in the subimage is 52 and the maximum value is 154. The *cdf* of 64 for value 154 coincides with the number of points in the image. The cdf must be normalized to [0,255]. The general histogram equalization formula is:

$$cdf(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right) \quad (6)$$

Where cdf_{min} is the minimum value of the cumulative distribution function (in this case 1), *MxN* gives the image's number of points (for the example above 64, where *M* is width and *N* the height) and *L* is the number of geophysical data levels used (in

most cases, like this one, 256). The equalization formula for this particular example is:

$$cdf(v) = \text{round}\left(\frac{cdf(v) - 1}{63} \times 255\right) \tag{7}$$

For example, the cdf of 78 is 46. (The value of 78 is used in the bottom row of the 7th column.) The normalized value becomes:

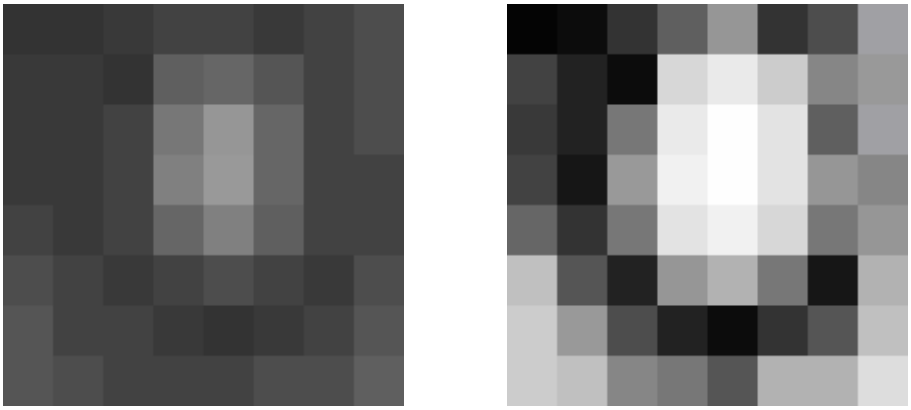
$$cdf(78) = \text{round}\left(\frac{46 - 1}{63} \times 255\right) = \text{round}\left(0.714286 \times 255\right) = 182 \tag{8}$$

Once this is done then the values of the equalized image are directly taken from the normalized *cdf* to yield the equalized values. Figure 2.2 shows the equalized image.

$$\begin{bmatrix} 0 & 12 & 53 & 93 & 146 & 53 & 73 & 166 \\ 65 & 32 & 12 & 215 & 235 & 202 & 130 & 158 \\ 57 & 32 & 117 & 239 & 251 & 227 & 93 & 166 \\ 65 & 20 & 154 & 243 & 255 & 231 & 146 & 130 \\ 97 & 53 & 117 & 227 & 247 & 210 & 117 & 146 \\ 190 & 85 & 36 & 146 & 178 & 117 & 20 & 170 \\ 202 & 154 & 73 & 32 & 12 & 53 & 85 & 194 \\ 206 & 190 & 130 & 117 & 85 & 174 & 182 & 219 \end{bmatrix}$$

Figure 2.2 Equalized image

It is important to pay attention that the minimum value (52) is now 0 and the maximum value (154) is now 255.



Original Image Equalized Image

Figure 2.3 Comparision of original and equalized images

Figure 2.3 shows the differences of the original and the equalized images. The contrast of the lighter and darker colors can be seen easily.

2.2 Interpolation

In the mathematical subfield of numerical analysis, interpolation is a method of constructing new data points within the range of a discrete set of known data points. In engineering and science one often has a number of data points, as obtained by sampling or experimentation, and tries to construct a function which closely fits those data points. This is called curve fitting or regression analysis. Interpolation is a specific case of curve fitting, in which the function must go exactly through the data points.

Interpolation can be used by many applications. In archaeological applications, interpolation is as quite relevant as the biomedical applications. Interpolation can be used in these applications to modify the sampling rate of pixels (picture elements) or voxels (volume elements). This operation, named rescaling, is desirable when an acquisition device or a scanner, has a nonhomogeneous resolution, typically a fine within slice resolution and a coarse across slice resolution. In this case, the purpose is to change the aspect ratio of voxels in such a way that they correspond to geometric

cubes in the physical space. Often, the across slice resolution is modified to match the within slice resolution, which is left unchanged. This results in a volumetric representation that is easy to handle (e.g., to visualize or to rotate) because it enjoys homogenous resolution. A related operation is reslicing. Suppose again that some volume has a higher within slice than across slice resolution. In this case, it seems natural to display the volume as a set of images oriented parallel to the slices, which offers its most detailed presentation. Physicians may however be sometimes interested in other views of the same data; for simplicity, they often request that the volume is also displayed as set of images oriented perpendicular to the slices.

In this project four kinds of interpolation is used:

- Trilinear Interpolation
- Cosine Interpolation
- Cubic Interpolation
- Nearest Neighbor Interpolation

2.2.1 Trilinear Interpolation

Linear interpolation takes two data points and intends to find the interpolated value by using distance proportion approach. The points are simply joined by straight line segments. Each segment (bounded by two data points) can be interpolated independently.

```
public static double linearInterpolate(double a1, double
a2, int n, int m)
{
      return a1+(a2-a1)*n/m;
}
```

Figure 2.4 Trilinear Interpolation Java code

Figure 2.4 shows Trilinear Interpolation Java Code. *m* defines the count of new points to be interpolated. The parameter *n* defines where to estimate the value, on the interpolated line. It is 0 at the first point and 1 at the last point. Interpolated point is

the nth value from the beginning. Figure 2.5 shows the graph of changing the result value in Linear Interpolation method according to *a*. *a* is the parameter which is used in the Java code in Figure 2.4.

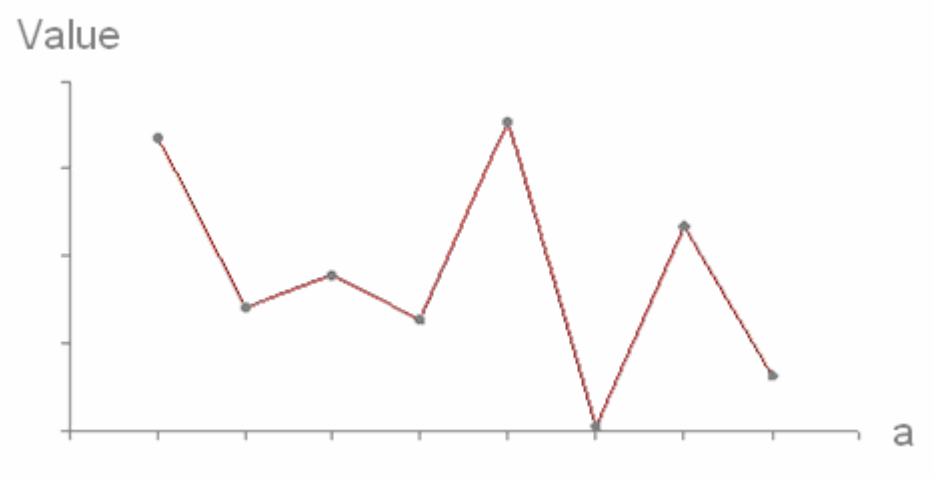


Figure 2.5 Linear Interpolation graph

Trilinear interpolation is a method of multivariate interpolation on a three dimensional regular grid. It approximates the value of an intermediate point (x,y,z) within the local axial rectangular prism linearly, using data on the lattice points. For an arbitrary, unstructured mesh (as used in finite element analysis), other methods of interpolation must be used; if all the mesh elements are tetrahedra (3D simplices), then barycentric coordinates provide a straightforward procedure. Trilinear interpolation is frequently used in numerical analysis, data analysis, and computer graphics.

On a periodic and cubic lattice with spacing 1, let x_d, y_d, and z_d be the differences to the largest integer smaller than each of x, y, z, that is:

$$x_d = x - \lfloor x \rfloor \tag{9}$$

$$y_d = y - \lfloor y \rfloor \tag{10}$$

$$z_d = z - \lfloor z \rfloor \tag{11}$$

First it is interpolated along z, giving:

$$i_1 = v[\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor] \times (1 - z_d) + v[\lfloor x \rfloor, \lfloor y \rfloor, \lceil z \rceil] \times z_d \quad (12)$$

$$i_2 = v[\lfloor x \rfloor, \lceil y \rceil, \lfloor z \rfloor] \times (1 - z_d) + v[\lfloor x \rfloor, \lceil y \rceil, \lceil z \rceil] \times z_d \quad (13)$$

$$j_1 = v[\lceil x \rceil, \lfloor y \rfloor, \lfloor z \rfloor] \times (1 - z_d) + v[\lceil x \rceil, \lfloor y \rfloor, \lceil z \rceil] \times z_d \quad (14)$$

$$j_2 = v[\lceil x \rceil, \lceil y \rceil, \lfloor z \rfloor] \times (1 - z_d) + v[\lceil x \rceil, \lceil y \rceil, \lceil z \rceil] \times z_d. \quad (15)$$

Then these values are interpolated along y, giving:

$$w_1 = i_1(1 - y_d) + i_2 y_d \quad (16)$$

$$w_2 = j_1(1 - y_d) + j_2 y_d \quad (17)$$

Finally these values are interpolated along x:

$$IV = w_1(1 - x_d) + w_2 x_d. \quad (18)$$

These equations give a predicted value for the point.

The result of trilinear interpolation is independent of the order of the interpolation steps along the three axes: any other order, for instance along x, then along y, and finally along z, produces the same value.

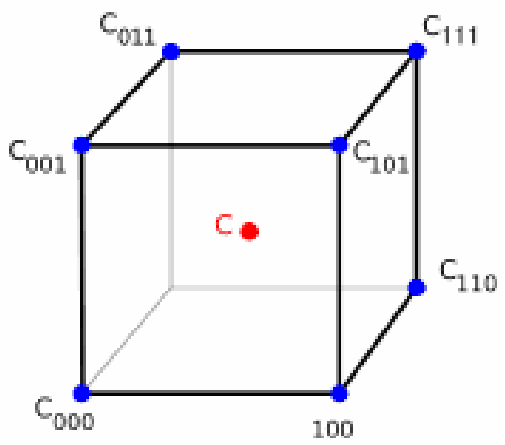


Figure 2.6 A cube with eight points

Figure 2.6 shows a cube with eight points which Trilinear Interpolation will be performed on. The above operations can be visualized as follows: First the eight corners of the cube are assessed. These corners have the values C_{000}, C_{100}, C_{010}, C_{110}, C_{001}, C_{101}, C_{011}, C_{111}.

Next, linear interpolation is performed between C_{000} and C_{100} to find C_{00}, C_{001} and C_{101} to find C_{01}, C_{011} and C_{111} to find C_{11}, C_{010} and C_{110} to find C_{10}.

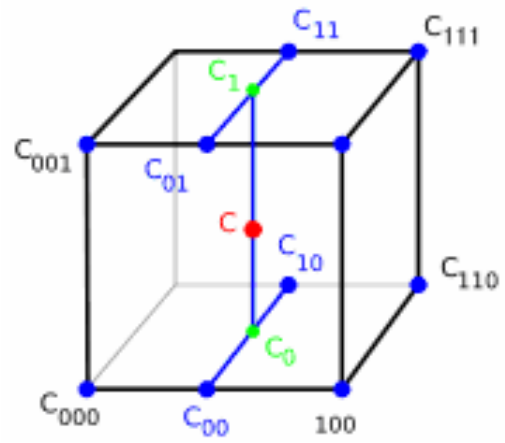


Figure 2.7 Trilinear Interpolation

Then interpolation is done between C_{00} and C_{10} to find C_0, C_{01} and C_{11} to find C_1. Finally, the value C is calculated via linear interpolation of C_0 and C_1. Figure 2.7 shows the operations which are used to find the value C.

2.2.2 Cosine Interpolation

Cosine Interpolation is a smoother function than the linear interpolation. A suitable orientated piece of a cosine function serves to provide a smooth transition between adjacent segments.

```
public static double cosineInterpolate(double a1, double
a2, int n, int m)
{
      double mu=(1-Math.cos(Math.PI*n/m))/2;
      return a1*(1-mu)+a2*mu;
}
```

Figure 2.8 Cosine Interpolation Java code

Figure 2.8 shows Cosine Interpolation Java Code. *m* defines the count of new points to be interpolated. The parameter *n* defines where to estimate the value, on the interpolated line. Figure 2.9 shows the graph of changing the result value in Cosine Interpolation method according to *a*.

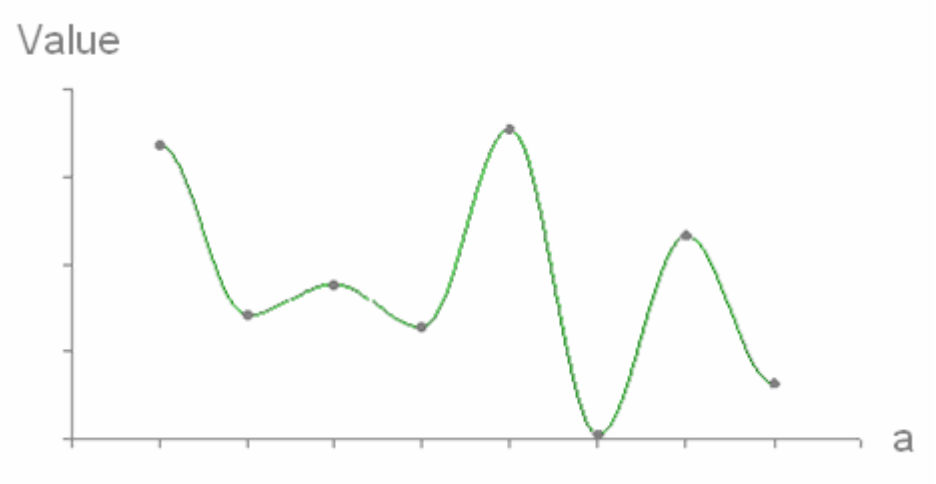


Figure 2.9 Cosine Interpolation graph

2.2.3 Cubic Interpolation

Cubic interpolation is the simplest method that offers true continuity between the segments. As such it requires more than just the two endpoints of the segment but also the two points on either side of them. So the function requires 4 points in all labeled *y0*, *y1*, *y2*, and *y3*, in the Java Code in Figure 2.10.

```
public static double cubicInterpolate(double y0, double
y1, double y2, double y3, int n, int m)
{
      double mu, mu2, a0, a1, a2, a3;

      mu  = (double)n/m;
      mu2 = mu*mu;
      a0  = y3-y2-y0+y1;
      a1  = y0-y1-a0;
      a2  = y2-y0;
      a3  = y1;
      return a0*mu*mu2 + a1*mu2 + a2*mu + a3;
}
```

Figure 2.10 Cubic Interpolation Java code

mu behaves for interpolating between the segment *y1* to *y2*. This does raise issues for how to interpolate between the first and last segments. A common solution is the dream up two extra points at the start and at the end of the sequence, the new points are created so that they have a slope equal to the slope of the start or end segment. Figure 2.11 shows the graph of changing the result value in Cubic Interpolation method according to *a*.

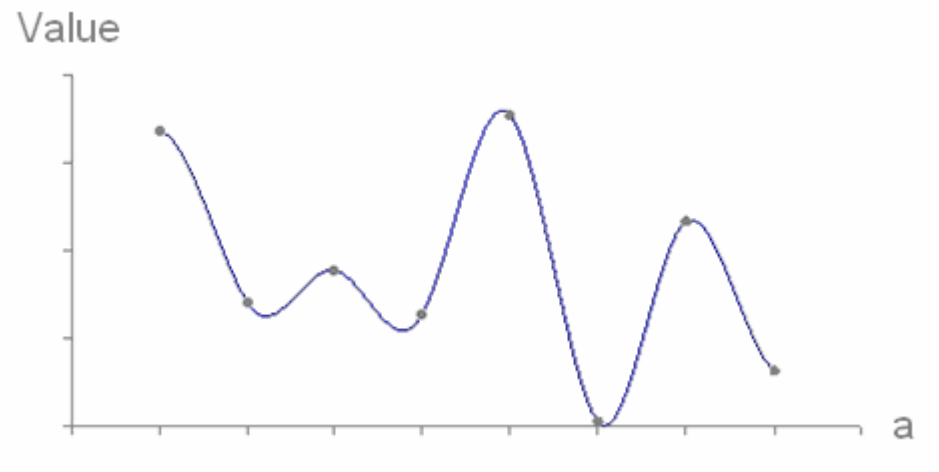


Figure 2.11 Cubic Interpolation graph

2.2.4 Nearest Neighbor Interpolation

The simplest interpolation method is to locate the nearest data value, and assign the same value. In one dimension, there are seldom good reasons to choose this one over linear interpolation, which is almost as cheap, but in higher dimensions, in

multivariate interpolation, this can be a favorable choice for its speed and simplicity. Figure 2.12 shows the graph of changing the result value in Nearest Neighbor Interpolation method according to *a*.

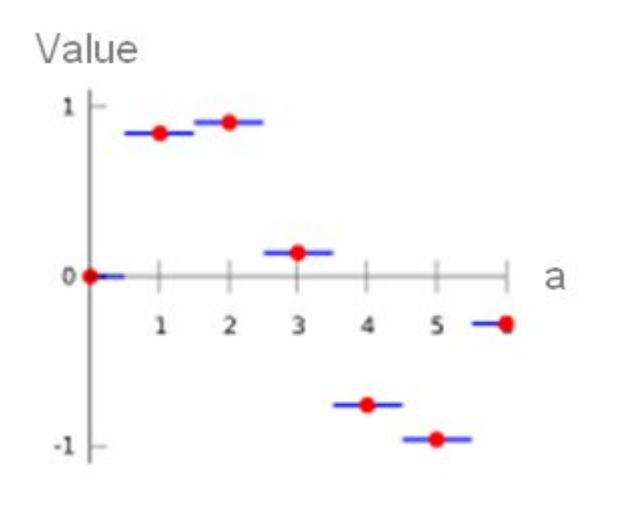


Figure 2.12 Nearest Neighbor Interpolation

CHAPTER THREE

VTK AND IDW

Visualization process of the application in this project is developed by using VTK. VTK is an open-source, object-oriented software system for 3D computer graphics, visualization and image processing. VTK is implemented in C language, but VTK also supports Java, Tcl and Java language bindings, permitting complex applications, rapid application prototyping, and simple scripts. So java is choosen for use in this project.

Although VTK doesn't provide any user interface components, it can be integrated with existing widget sets, such as in this project it is integrated with Java SWING and Java AWT Libraries. VTK provides a variety of data representations including unorganized point sets, polygonal data, images, volumes and structured, rectilinear, and unstructured grids. VTK comes with readers/importers and writers/exporters to exchange data with other applications. Hundreds of data processing filters are available to operate on these data, ranging from image convolution to Delaunay triangulation. VTK's rendering model supports 2D, polygonal and volumetric approaches that can be used in any combination.

In this application portable windows are used. This ability is provided by using IDW (InfoNode Docking Windows) tool. The architecture in IDW allows the user to move, minimize and maximize the independent dockable windows. Windows can be also closed and restored independently from the other dockable windows.

3.1 VTK Architecture

VTK consists of two major pieces; a compiled core (implemented in Ç) and an automatically generated interpreted layer. The interpreted layer supports Java, Tcl and Python.

3.1.1 VTK Core

Data structures, algorithms, and time-critical system functions are implemented in the core. Common design patterns such as object factories and virtual functions insure portability and extensibility. Since VTK is independent of any graphical user interface (GUI), it doesn't depend on the windowing system. Hooks into the window ID and event loop let developers plug VTK into their own applications. An abstract graphics model achieves graphics portability.

3.1.2 VTK Interpreted Layer

While the compiled core provides speed and efficiency, the interpreted layer offers flexibility and extensibility. For example, using GUI prototyping tools such as Java AWT and Java SWING, Tcl/Tk or Python/Tk permits building professional applications rapidly. These popular programming languages come with other packages such as Python's numerical library NumPy.

VTK is a large, complicated and powerful toolkit for 3D imaging and visualization. It is an object-oriented library. Many operations in VTK are performed using a pipeline architecture where multiple elements are attached together to perform a complex task. A typical pipeline takes the form shown in the figure below. This is broken into two parts. The first part of the VTK Visualization Pipeline Architecture is shown in Figure 3.1 below.

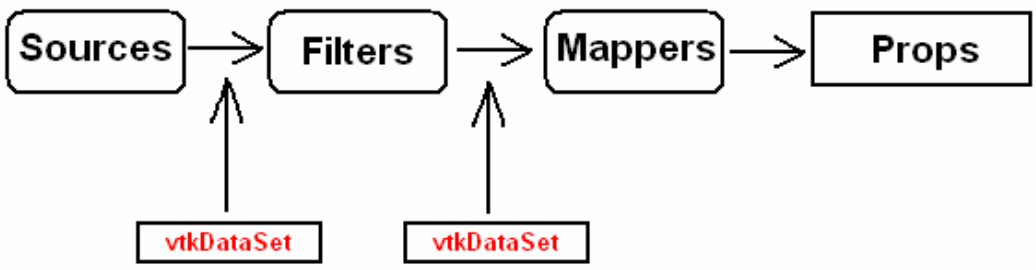


Figure 3.1 First part of the VTK Visualization Pipeline Architecture

- Sources; these classes produce data. For example, vtkImageData is a source which consists the image data.
- Filters; these operate on some data to produce a modified version. For example, vtkVolume is used to represent a volumetric entity in a rendering scene. The volume maintains a reference to the volumetric data (i.e. the volume mapper).
- Mappers; these define the interface between data (e.g. images) and graphics primitives or software rendering techniques. A special kind of "mapper" like class are the writers which output the data to files (e.g. vtkVolumeMapper). Multiple mappers may share the same input, but render it in different ways.

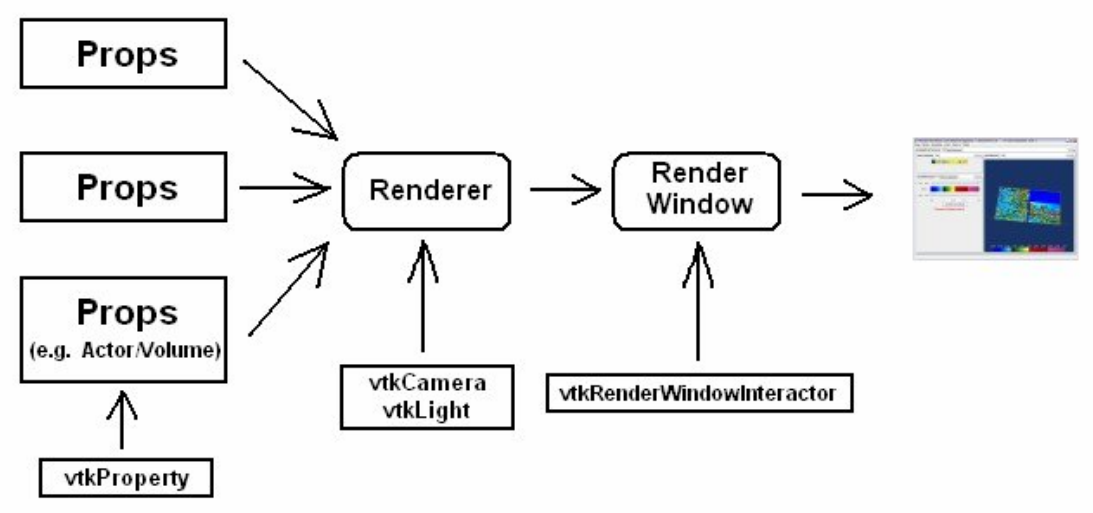


Figure 3.2 Second part of the VTK Visualization Pipeline Architecture

The second part of the VTK Visualization Pipeline Architecture is shown in Figure 3.2. This part consists of the elements that make up the virtual 3D world, namely:

- Props/Actors; these take as input the output of a mapper and know how to generate the visible representation of data. The type of rendering produced is governed by an auxiliary data structure known as a property (e.g. color, showing a surface as a wire frame vs a full surface etc.). Props take as their input the output of a mapper. Mappers should not be shared among props.

Volumes – these are special kinds of props that are used to display volume rendered images.

- Renderer; Renderers are the classes that generate a 2D image from a 3D scene. They have attached actors as well as lights and cameras. Renderer is an object that controls the rendering process for objects. Rendering is the process of converting geometry, a specification for lights, and a camera view into an image. vtkRenderer also performs coordinate transformation between world coordinates, view coordinates (the computer graphics rendering coordinate system), and display coordinates (the actual screen coordinates on the display device). Certain advanced rendering features such as two-sided lighting can also be controlled.
- Render Window; the Render Window is the piece of screen real estate in which the virtual camera image is displayed. An important auxiliary item attached to a render window is an interactor which handles mouse/keyboard input to the window.

3.2 IDW

InfoNode Docking Windows (IDW) is a Java Swing framework for building docking windows. Docking windows are internal application windows which the user can rearrange by dragging them. Unlike common windows GUIs in Java, the docking windows are docked to each other and can not be moved around freely (with some exceptions).

Docking windows are commonly used in IDEs like Eclipse, Netbeans and the Microsoft developer tools. However, their usage is not in any way limited or connected to IDEs, many GUI applications can benefit form using a docking windows framework that allows the user to customize the layout of the application to his or her needs and preferences. Using IDW is similar to creating an application layout using JSplitPanes, JTabbedPanes and layout managers, but the GUI will become much more flexible and customizable. Some of the features of IDW are:

- Unlimited depth of nested split and tab windows. For example, two windows can be put in a split pane that is located in a tab of a tab pane. There is practically no limitation on the window layout.
- Tabs can be placed on any side of the tab window. The text and icon of the tabs can be rotated in any direction.
- Windows can be minimized to any edge of any application. They can be dragged to and from the edges.
- Windows can be docked to floating windows that can be moved anywhere on the desktop.
- A tab window can be maximized to cover the entire window space (except the window bars). It can be later be restored to its original location.
- A window or window tree can be undocked to a floating window.
- Easily save and load the window layout using streams.
- Theme support.
- Flexible properties system which allows the user to customize the look and behaviour of ID to suit the user's application.

CHAPTER FOUR

APPLICATION DESIGN AND DEVELOPMENT

The application in this thesis is coded under Java Platform with VTK support. It can be accepted that the application is developed with arrangement of some processes and steps as it is shown in Figure 4.1.

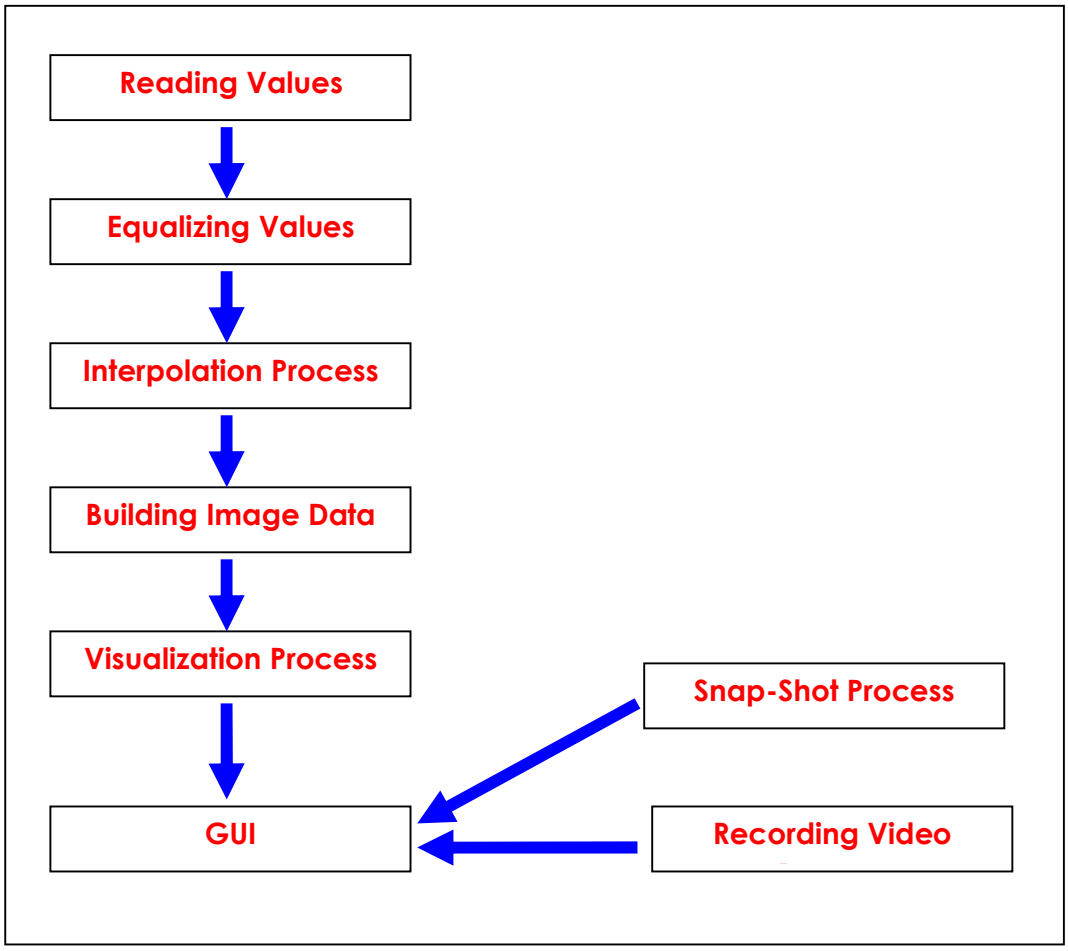


Figure 4.1 General architecture of the application

First, source file is read and value equalizing step starts. Then interpolation process begins and image data is builded to be used later in visualization process. Visualization Process is coded by using VTK in Java. Like the general architecture of the application seen in Figure 4.1, Visualization Process consists of some sub-

steps. When this process is finished, 3D visualization of data collected from the historical site is drawn in GUI frame by the application.

Snap-Shot and Recording Video processes are secondary processes in the application. These processes starts when user triggers them by main GUI form.

4.1 Reading and Equalizing the Values

Source file includes the coordinate and data values of the historical site. First, the source file is opened and the coordinate and geophysical values are read, then frequency arrays are made for each axes (Veriler.orj_x_degerler, Veriler.orj_y_degerler, Veriler.orj_z_degerler).

Source file must be prepared as tab separated file as in "x\ty\t\z\tr" form as it is shown in Figure 4.2. This means; x, y, z and geophysical data must be written in the same sentence by putting \t (tab character) between the values.

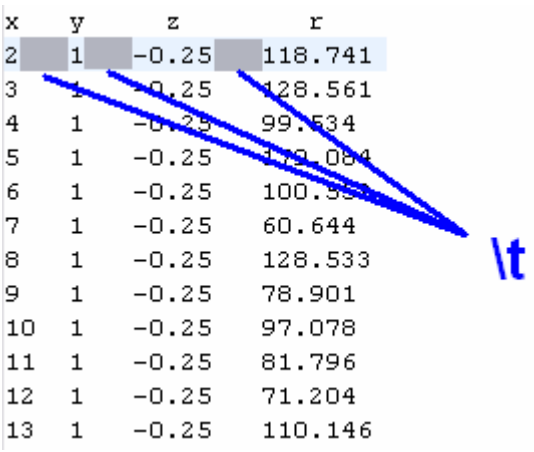


Figure 4.2 Structure of the source file

The frequency arrays are used for determining base xyz points and the array sizes of the coordinate values. To adjust the colors and to eliminate the unneeded data, "Histogram Equalisation" technique is used, as explained in Chapter TWO.

To adjust the data, the geophysical values are put into their place in the cumulative distribution function (cdf) array. Related element of the cdf array's value is incremented by 1. This also means that, it's frequency value is incremented. Thus, this provides to determine which geophysical values are most common used.

```
for(int i=1;i<101;i++)
{
      value=Veriler.cum_dist_func[i]+Veriler.cum_dist_func[i-1];
      Veriler.cum_dist_func[i]=value;
}
```

Figure 4.3 Building Cumulative Distribution Function (cdf) Java code

After reading all data and incrementing the related values, the geophysical data histogram is formed. Then cdf array is updated by adding each element's value to its next element's value as it is shown in Figure 4.3. This provides to find which values are mostly contributed to the coordinate array.

The situated cdf value is divided into the count of the geophysical values and multiplied by the highest data value (please take a look at equation 6). The result values are the new values of the equalized array.

4.2 Color Mapping Process

Color values (which will be calculated from the geophysical values) are calculated by considering the color map array and then values are balanced to 0-100 limits. Each color value is proportioned to the color map array, and so rgb values are found. After this operation, calculated color values are put into the point array.

Two classes are used to apply color mapping process. ColorMapPenceresi is a GUI class to select, open, change or create a Color Map file. When a Color Map file is selected by user, ColorMap class is used for loading the color map array. Color map files have ".clr" extension. File's first column is used for determining the limit percentage of rgb values. Following three values are expressing red, blue and green tones as it is shown in Figure 4.4.

```
ColorMap 1 1
    0.000000   0   0 255
    8.866995   0   0 255
    8.866995   0   0 255
   21.674877   0 255 255
   22.167488 204 204 204
   27.586207   0   0   0
   28.078818   0  51  51
   43.349754 153 255   0
   43.842365 255 255   0
   49.261084 255 255 102
   49.261084 220   0   0
   77.832512 153   0   0
   77.832512 255  51 153
  100.000000 153  51 102
```

Figure 4.4 ColorMap source file structure

ColorMap class has the "double[] rgbDondur(double value)" method. This method takes the geophysical value, looks the ColorMap array to find its equivalent red, green, blue values. The geophysical value is proportioned between the distance of two percentage value to find the color values. The Java code of this operations are shown in Figure 4.5.

```
public static double[] rgbDondur(double value)
{
      double rgb[]=new double[3];
      double red,green,blue,agirlik;
      red=0;
      green=0;
      blue=0;

      for(int i=0;i<Veriler.colorMapArray.length;i++)
      {
         if(value<Veriler.colorMapArray[i][0])
         {
            agirlik=(value-Veriler.colorMapArray[i-1][0]) /
(Veriler.colorMapArray[i][0]-Veriler.colorMapArray[i-1][0]);

            red=Veriler.colorMapArray[i-1][1] + agirlik *
(Veriler.colorMapArray[i][1]-Veriler.colorMapArray[i-1][1]);

            green=Veriler.colorMapArray[i-1][2] + agirlik *
(Veriler.colorMapArray[i][2]-Veriler.colorMapArray[i-1][2]);

            blue=Veriler.colorMapArray[i-1][3] + agirlik *
(Veriler.colorMapArray[i][3]-Veriler.colorMapArray[i-1][3]);

            break;
         }
         else if(value==Veriler.colorMapArray[i][0])
         {
            red=Veriler.colorMapArray[i][1];
            green=Veriler.colorMapArray[i][2];
            blue=Veriler.colorMapArray[i][3];
            break;
         }
      }
      rgb[0]=red;
      rgb[1]=green;
      rgb[2]=blue;

      return rgb;
}
```

Figure 4.5 Geophysical value to RGB color conversion Java code

If an example is given with considering the Figure 4.4, the ColorMap process can be better understood.

Let the value has the percentage of 27.8, then it can be seen that the value must be between 27.586207-28.078818 values. If the value is proportioned to these two values, the rgb values can be easily found:

value percentage:	27.8
red value:	0
green value:	22
blue value:	22

4.3 Building the Point Array

After geophysical values are converted to RGB colors, they are put into the point array with their coordinate values. The point array holds the instances of Nokta class objects. The point array's variable name is "Veriler.orjinal". Nokta class holds the RGB colors, coordinate and geophysical value of a point. Veriler.orjinal is a Three-Dimensional array which holds the objects derived from the class Nokta.

4.4 Interpolation Process

Interpolation is used to increase the sampling rate of the voxels to get better understanding, as interpolation is explained in CHAPTER TWO. User chooses two interpolation parameters; the interpolation method and the interpolation level. After that choices, the interpolation process begins. Because interpolation increases the sampling rate of the voxels, a new array is builded by the interpolation process (Veriler.noktalar). New array's point count is bigger than the original array (Veriler.orjinal) according to the interpolation level.

4.5 Building the Image Data and the Visualization Process

Building the point array leads to determine the image dimensions and the image spacing. The vtkImageData structure is defined by dimensions, spacing and origin. Dimensions are the number of voxels or pixels along each of the major axes. Origin is the starting point whose x,y,z values are the lowest than the other points. Spacing is the distance between pixels along each of the three major axes.

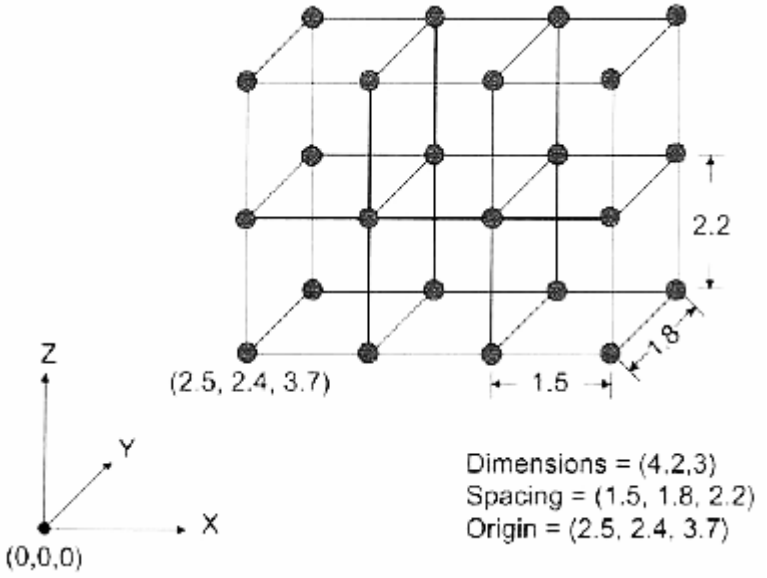


Figure 4.6 vtkImageData structure

Every point has its own geophysical color value. In Figure 4.6, the image's point count is, multiplication of the dimension values:

4*2*3=24

Lowest point (2.5, 2.4, 3.7) is regarded as the origin, which is also the image's first point.

The spaces in the figure, between lines are:

1.5 between x lines,

1.8 between y lines,

2.2 between z lines.

In this process, an archaeological image is created, processed, mapped, visualized and finally showed to the user. These operations are followed in a specified order, as it is shown in the Figure 4.7.

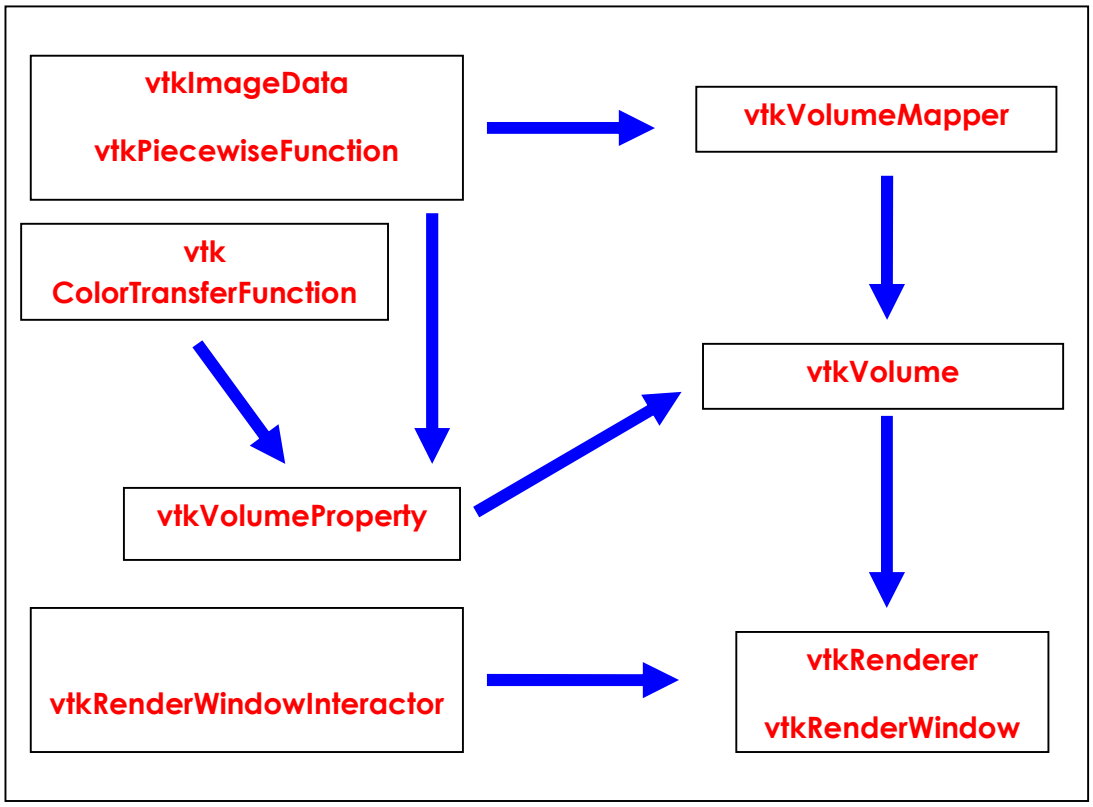


Figure 4.7 Architecture of the Imaging and Visualizing Process

The purposes of these concepts are explained in sub-sections in below.

4.5.1 vtkVolume

A vtkVolume is intended for use in volume rendering. A volume mapper and volume property must be specifed to describe the rendering technique and rendering parameters, respectively. vtkVolume holds the transformation information such as position, orientation, scale and pointers to the mapper and property for the volume. After building the image data, the functions that map scalar value into opacity and color, which are used in the vtkVolumeProperty, are defined.

The vtkVolume class accepts objects the vtkVolumeMapper as input to setMapper() and accepts a vtkVolumeProperty objects as input to SetProperty(). vtkVolume is a class that enforces the different types of the mappers and properties.

4.5.2 vtkPiecewiseFunction

In order to control the appearance of a three dimensional volume of scalar values, three mappings or transfer functions must be defined. The first transfer function, known as the "scalar opacity transfer function", maps the scalar value into an opacity or an opacity per unit length value. The second transfer function, referred to simply as the "color transfer function", maps the scalar value into a color.

Two important method exists in vtkPiecewiseFunction. Those add information to the mapping, and those that clear out information from the mapping. When information is added to a mapping, it is considered to be a point sample of the mapping with linear interpolation used to determine values between the specified ones. For example, the following code in Figure 4.8 produces the transfer function draw on the below in Figure 4.9.

```
vtkPiecewiseFunction pf;
pf =new vtkPiecewiseFunction();
pf.AddPoint(50,0.2);
pf.AddPoint(200,1.0);
```

Figure 4.8 vtkPiecewiseFunction Producing Code (I)

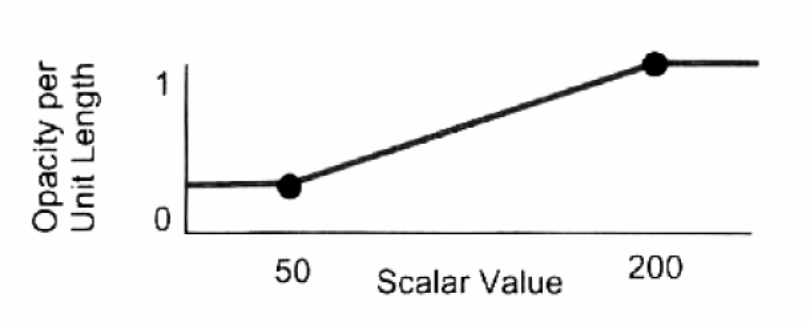


Figure 4.9 Changing the opacity (I)

If user doesn't want to see some data then values are mapped to an opacity of 0.0 to eliminating them from contributing to the image. The other values are mapped to an opacity of 1.0.

The value of the mapping for the scalar values of 50 and 200 are given as 0.2 and 1.0 respectively and all other mapping values can be obtained by interpolating between these two values.

Points can be added to the mapping at any time. For example user can change the data limits to be shown in the visualizaion GUI form. By this time, a new mapping is redefined. If a mapping is redefined, it replaces the existing mapping. In addition to adding a single point, a segment can be added which will define two mapping points and clear any existing points between the two. As an example, by considering the following two modification steps and the corresponding pictorial representations of the transfer functions:

```
pf.RemovePoint(50);
pf.AddPoint(50,0.0);
pf.AddSegment(100,0.8,150,0.2);
```

Figure 4.10 vtkPiecewiseFunction producing code (II)

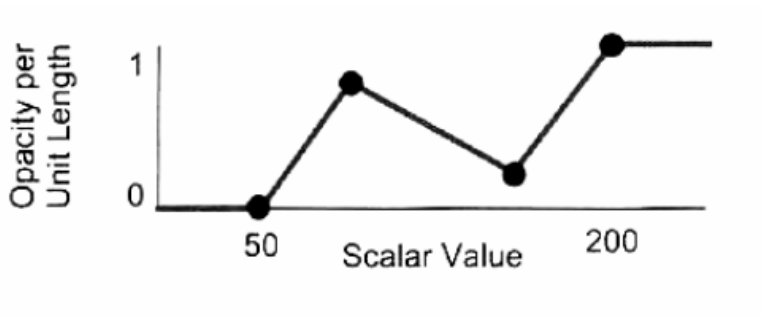


Figure 4.11 Changing the opacity (II)

```
pf.AddPoint(50,0.2);
pf.AddSegment(60,0.4,190,0.8);
pf.clampingOff();
```

Figure 4.12 vtkPiecewiseFunction producing code (III)

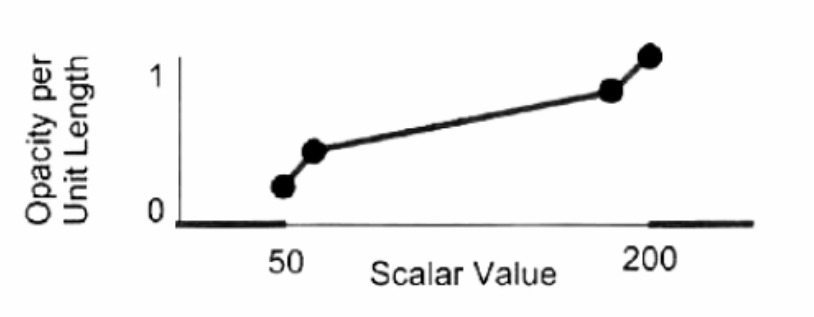


Figure 4.13 Changing the opacity (III)

In the first step in Figure 4.11, by applying the code in Figure 4.10, the mapping of scalar value 50 is changed by removing the point and then again the value and a segment added. In the second step in Figure 4.13, by applying the code in Figure 4.12, the mapping of scalar value 50 is changed by simply adding a new mapping without first removing the old one. A new segment is also added which eliminates the mappings for 100 and 150 since they lie within the new segment.

4.5.3 vtkColorTransferFunction

A vtkColorTransferFunction can be used to specify a mapping of scalar value to color using either an RGB or HSV color space. The methods available are similar to those provided by vtkPiecewiseFunction, but tend to come in two flavors. For example, AddRGBPoint() and AddHSVPoint() both add a point into the transfer function with one accepting an RGB value as input and the other accepting an HSV value as input. In this project RGB methods are used to define the color functions.

The following Java code shown in Figure 4.14, explains how to specify a transfer function from red to green to blue with RGB interpolation performed for values in between those specified:

```
vtkColorTransferFunction ctf = new vtkColorTransferFunction();
ctf.AddRGBPoint(i,red,green,blue);  // "i" is the order of the
point in image data
```

Figure 4.14 Specifying a Transfer Function Code from Red to Green to Blue

4.5.4 vtkVolumeProperty

Defining the transfer functions is the hardest and longest part of achieving an effective volume visualization because a classification operation is performed to understand the meaning of the underlying xyz and geophysical data values. Volume visualization uses a vtkVolumeProperty to contain the information for how to convert data values to a color and an opacity (i.e., the transfer function) and what lighting parameters to use. The vtkPiecewiseFunction and vtkColorTransferFunction are used to specify the details of the transfer functions. The transfer functions which are releated to vtkVolumeProperty class are, the vtkPiecewiseFunction and the vtkColorTransferFunction. The SetColor() method accepts a vtkColorTransferFunction. The SetScalarOpacity() method accepts a vtkPiecewiseFunction to define the scalar opacity transfer function. Figure 4.15 shows how vtkVolumeProperty lies the transfer functions.

```
vtkVolumeProperty prop = new vtkVolumeProperty();
prop.SetColor(ctf);
prop.SetScalarOpacity(pf);
```

Figure 4.15 Setting up the Color and Opacity Transfer Functions code

4.5.5 vtkVolumeMapper

vtkVolumeMapper has the setInput() method with an argument pointer to the vtkImageData object. vtkVolumeMapper has the SetCroppingRegionPlanes() method which can be used to slice the image data by x,y or z planes. vtkVolumeMapper is the bridge between the image data and the volume, as shown in the Figure 4.7.

4.5.6 Finishing the Visualization Process, Rendering Concepts

vtkRenderer and vtkRenderWindow are used to manage the interface between the graphics engine and the Java Panel (JPanel) in main window. The render window is

the window in the JPanel that the renderer draws into. The region that the renderer draws into is visualization area.

One the drawing process is finished, the object vtkRenderWindowInteractor is used for manipulating the camera, picking the historical site.

Keypress r, resets the camera view along the current view direction and fits and centers the historical site on the panel. And also provides all actors to visible.

4.6 Snap-Shot Process

After the completion of the visualization process, user can interact with the drawing in the screen. User can rotate the historical site, cut the site and change the transparency of some data values. And user may want to take a picture from one scene. In this case, this process is applied for doing this.

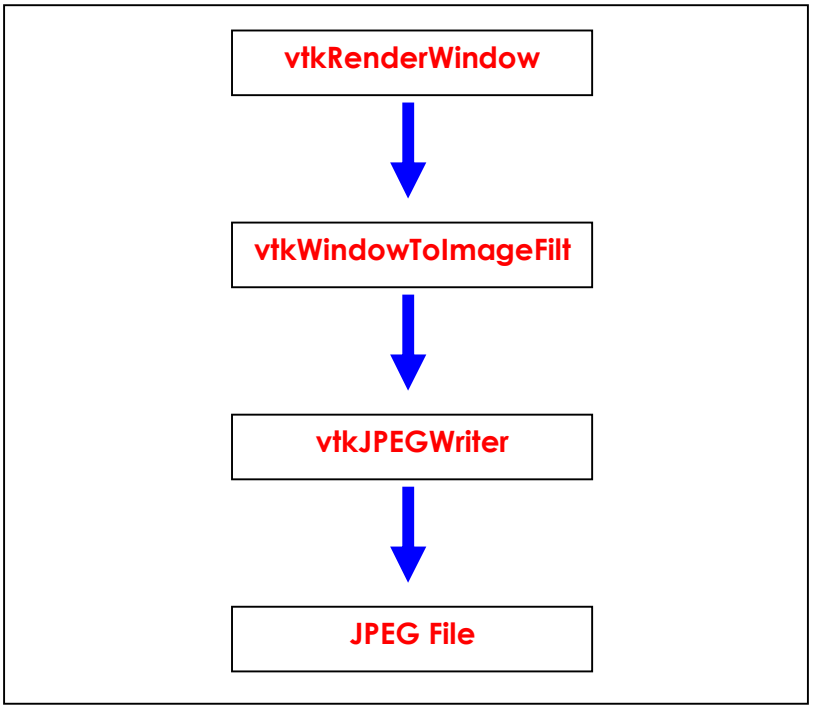


Figure 4.16 Pipeline of Snap-Shot Process

In above, Figure 4.16 shows the pipeline of this process. vtkRenderWindow keeps the image data of the drawing information shown to the user. vtkWindowToImageFilter takes the kept image data in the render window and sends it to vtkJPEGWriter. Finally, vtkJPEGWriter writes the image data to a JPEG file, which is selected by user before.

4.7 RecordingVideo Process

Recording Video Process is similar to the Snap-Shot Process that is shown in Figure 4.17, but some differences are existing in this process.

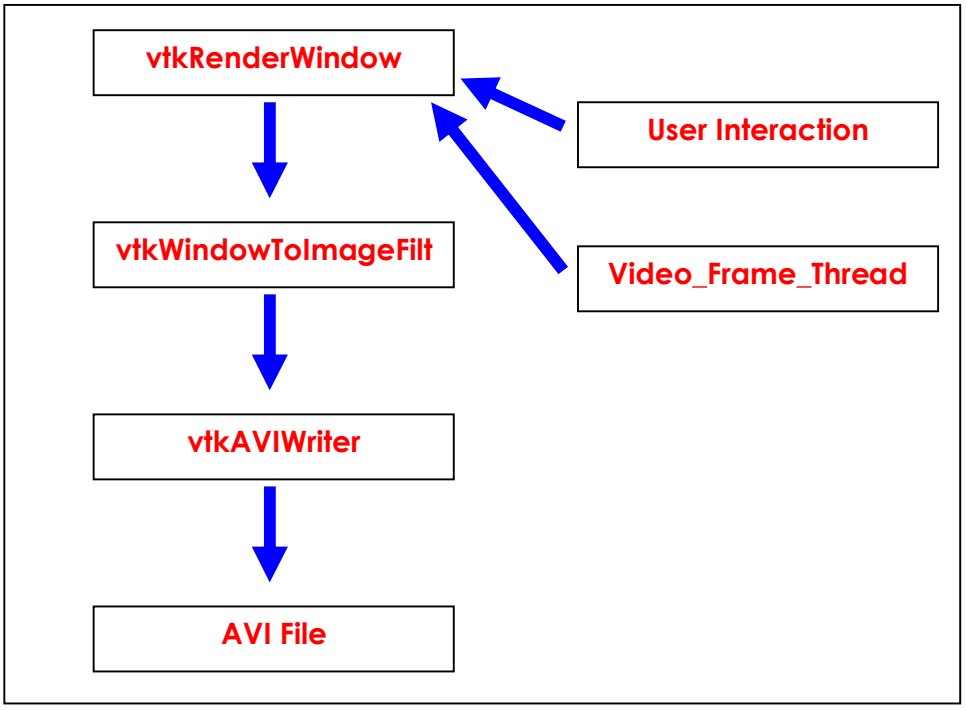


Figure 4.17 Pipeline of Recording Video Process

In Snap-Shot Process, vtkWindowToImageFilter takes the image data from the render window once only. But in this process, vtkWindowToImageFilter takes the image data in every user interaction. Or it takes the image data 15 times in a second if an user interaction does not trigger. This event is being done by a Java Thread class whose name is Video_Frame_Thread.

Video_Frame_Thread observes the user interactions and coordinates them. The product video of this process has 15 frames/second. This means that, this thread divides a second to 15 unit time. If an user interaction doesn't exist in a specific unit time, the thread uses the previous unit time's image data.

After the end of the process, vtkAVIWriter writes the image datas to an AVI file, which is selected by user before.

4.8 GUI Forms and Some Explanation About the Application

At this sub-section, some explanation and important information will be given about the application.

4.8.1 Starting the Application

If the application is started, then a screen at Figure 4.18 will be appeared. In this screen, there are three buttons. First button changes the language of the application from Turkish to English or From English to Turkish.

Figure 4.18 Starting frame of the application

Second button, “Son Projeyi Aç”, opens the last project which user worked on before. In this option the historical site will be shown to the user by previous cuttings and camera settings. Namely, it can be said that, all the operations lastly made by the user, will be shown in the visualization frame.

Third button, Yeni Proje, opens a new project. This options gives the chance to the user to select interpolation method, interpolation level, color map file, maximum geophysical data limit and the historical site source file. These options can be selected in New Project Screen shown in Figure 4.19.

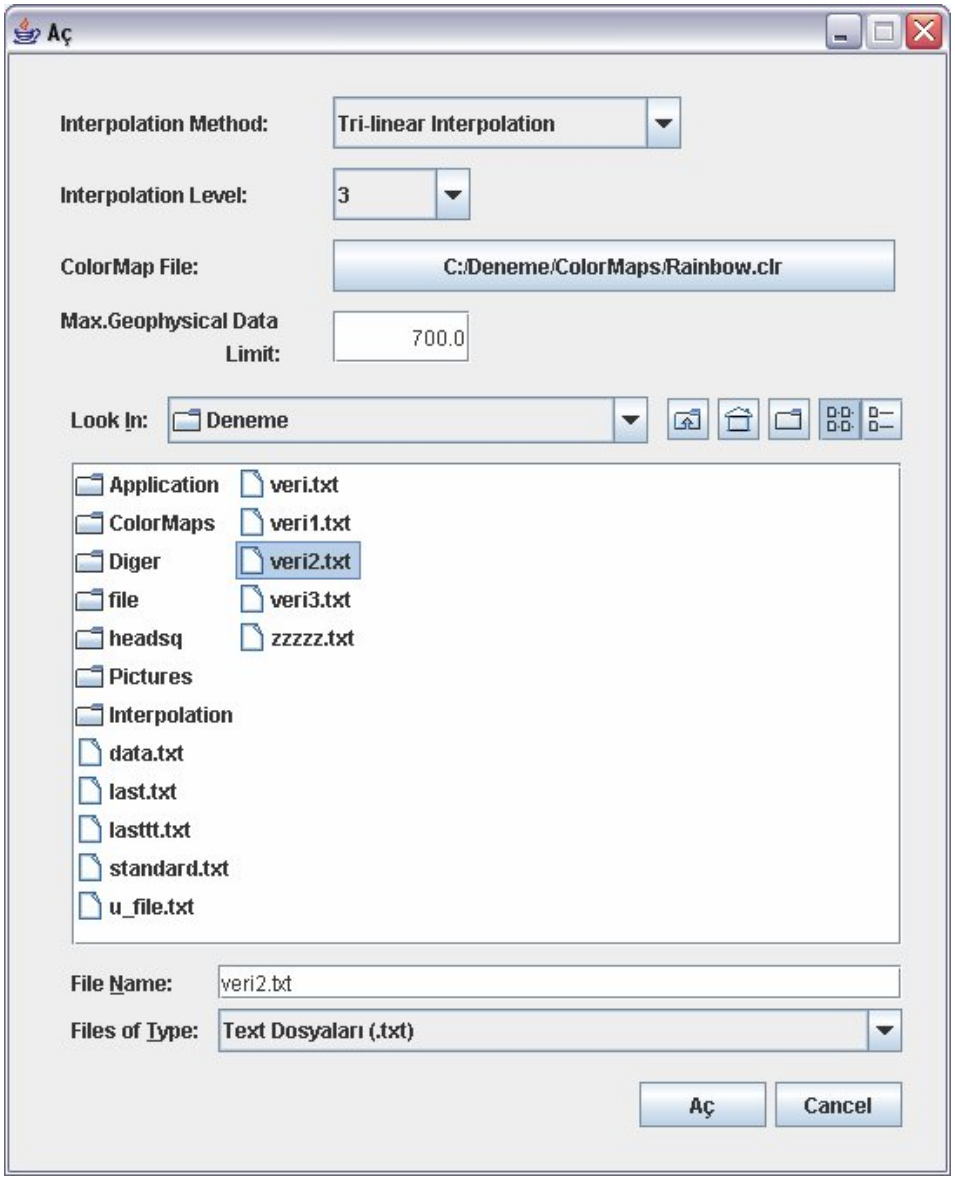


Figure 4.19 New Project screen

Max. Geophysical Data Limit determines, to which geophysical data are regarded for the visualization process. Upper values are accepted as the Max. Geophysical Data Limit value.

For interpolation options are existing in the application. Detailed explanation is given in CHAPTER TWO about the interpolation methods.

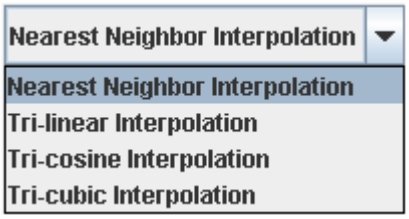


Figure 4.20 Interpolation Options

After the interpolation method is choosen as it is shown in Figure 4.20, the interpolation level must be selected. Bigger numbers increases the number of points in the image data and creates smoother and more understandable visualization. But it takes much time to finish the visualization process.

Table 4.1 shows that, when interpolation level increases, avarage visualization time increases too. If user wants more quality visualization, then he or she must choose a bigger interpolation level and wait much time than little interpolation levels.

The interpolation level selection is and important function in geophysics. Sometimes, the bigger interpolation level is resulted more smoothed data, and this situation can remove the small data levels, which can be important in data interpolation.

Table 4.1 Interpolation Level – Average Visualization Time

Point Count	Interpolation Level	Average Visaulization Time (sec)
7080	1	1.5
	2	2.0
	3	2.5
	4	3.1
	5	3.5
	6	4.0
	7	5.0
	8	7.0
	9	8.0
	10	9.1
19470	1	3.0
	2	5.0
	3	8.0
	4	11.5

4.8.2 ColorMap Operations

Before visualization process, user must choose the color map file, as it is shown in Figure 4.21. “Rainbow.clr” file is the default file at starting position. But user can change the color map file.

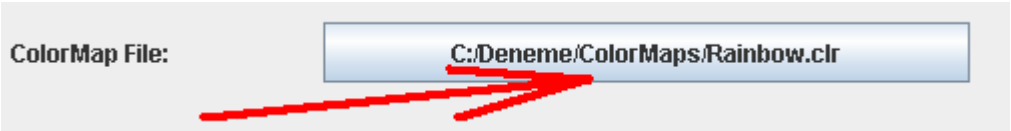


Figure 4.21 Select Color Map button

If the button in In Figure 4.21 is clicked, then the ColorMap Window will be opened. This window is used for selecting a color map file from a folder or creating a new color map file or editing a created color map file.

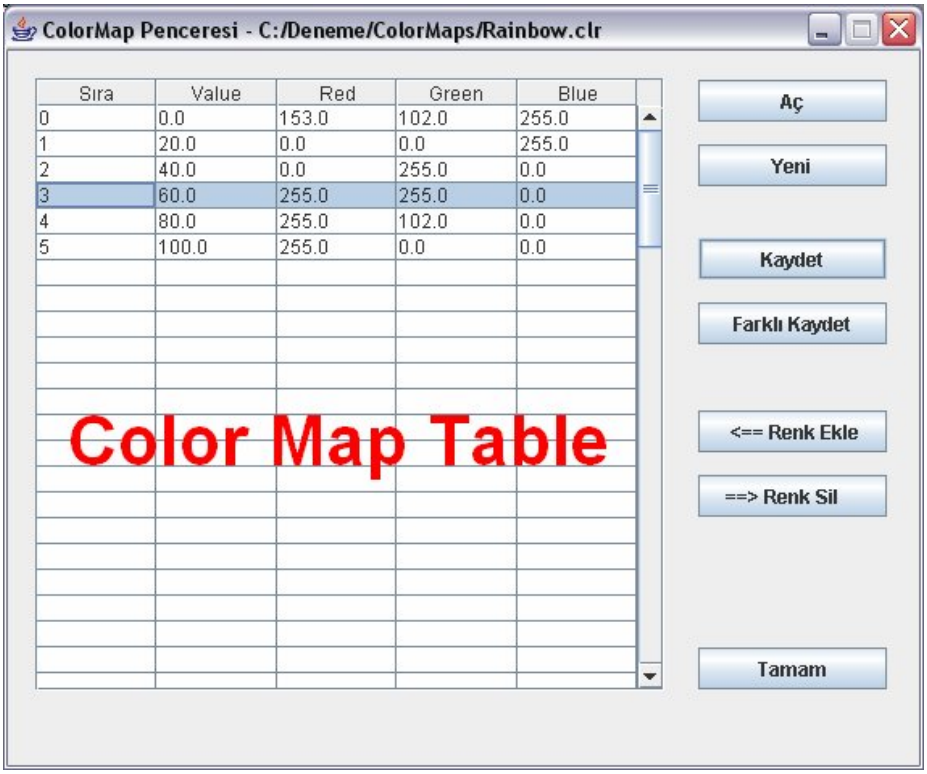


Figure 4.22 Color Map window

The color map window is shown is Figure 4.22. The table, in the color map window, shows the RGB color values and their percentage values. The functions of buttons in window are explained in below:

Aç: If another color map file is wanted to be chosen, user have to click this button, and select the new file from file chooser panel.

Yeni: Opens a new blank color map file

Kaydet: If a new color map file is created or an old color map file is edited, a file chooser panel is shown to the user and then user can save the color map file by the file chooser panel.

Farklı Kaydet: When a color map file is edited, if this button is clicked, user can save the color map file with a different name.

<== Renk Ekle: This button adds a new line after the selected line in the color map table.

==> Renk Sil: When this button is clicked, the selected line will be deleted, and next lines will be shifted upward.

Tamam: This button is used to accept the selected color map file and return to previous screen.

4.8.3 Main Visualization Window

Visualization window is the main window of the application. Main window is shown in Figure 4.23. As it can be seen in the figure, there are five panel windows are existing in the main window:

- Visaulization Window (Görselleştirme)
- Cutting Operations Window (Kesme İşlemleri)
- Choose Geophysical Data Window (Jeofiziksel Veri Seçimi)
- Operation Shortcuts Window (İşlem Kısayolları)
- Data Window (Veri)

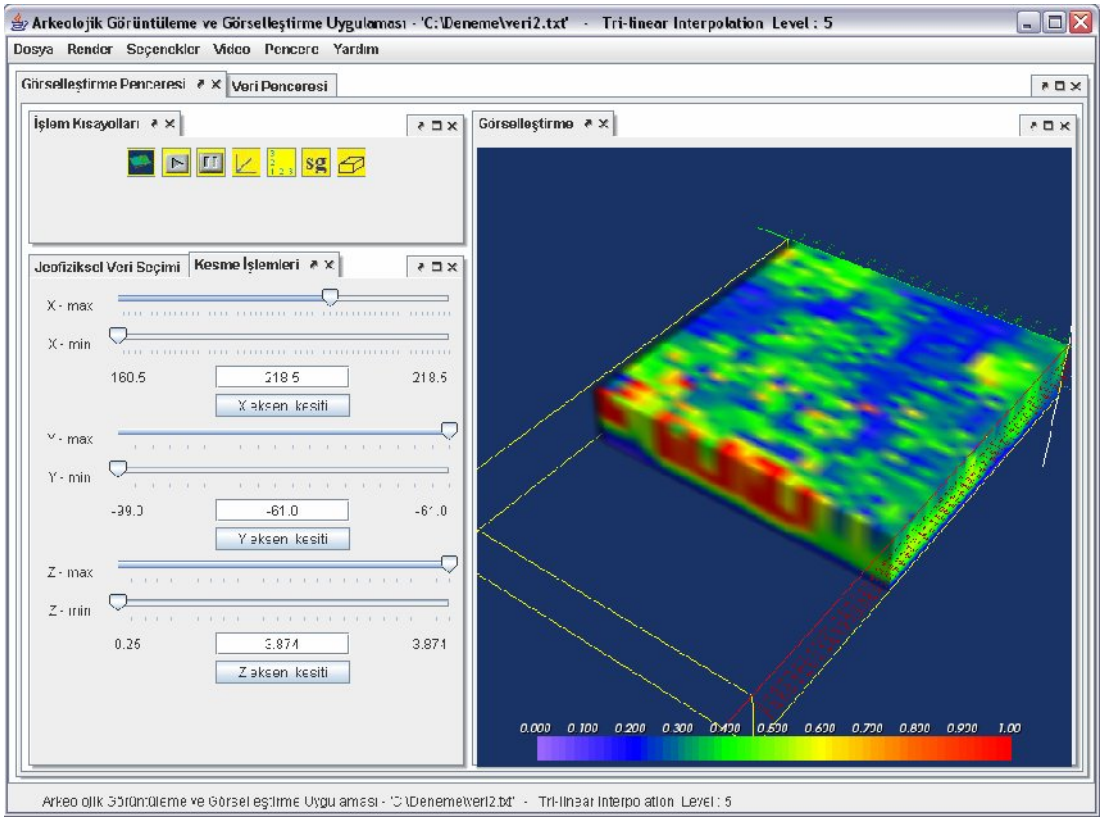


Figure 4.23 Main Visualization window

These panels are developed in tabbed structure. The structure is builded by IDW (Infonode Docking Windows) tool. These windows can be minimized, maximized, restored, closed, docked and undocked freely form the main window. The usage of the windows are explained below sections.

4.8.3.1 Visualization Window

This window is the visualization area where rendering and drawing processes are executed. The window is shown in Figure 4.24.

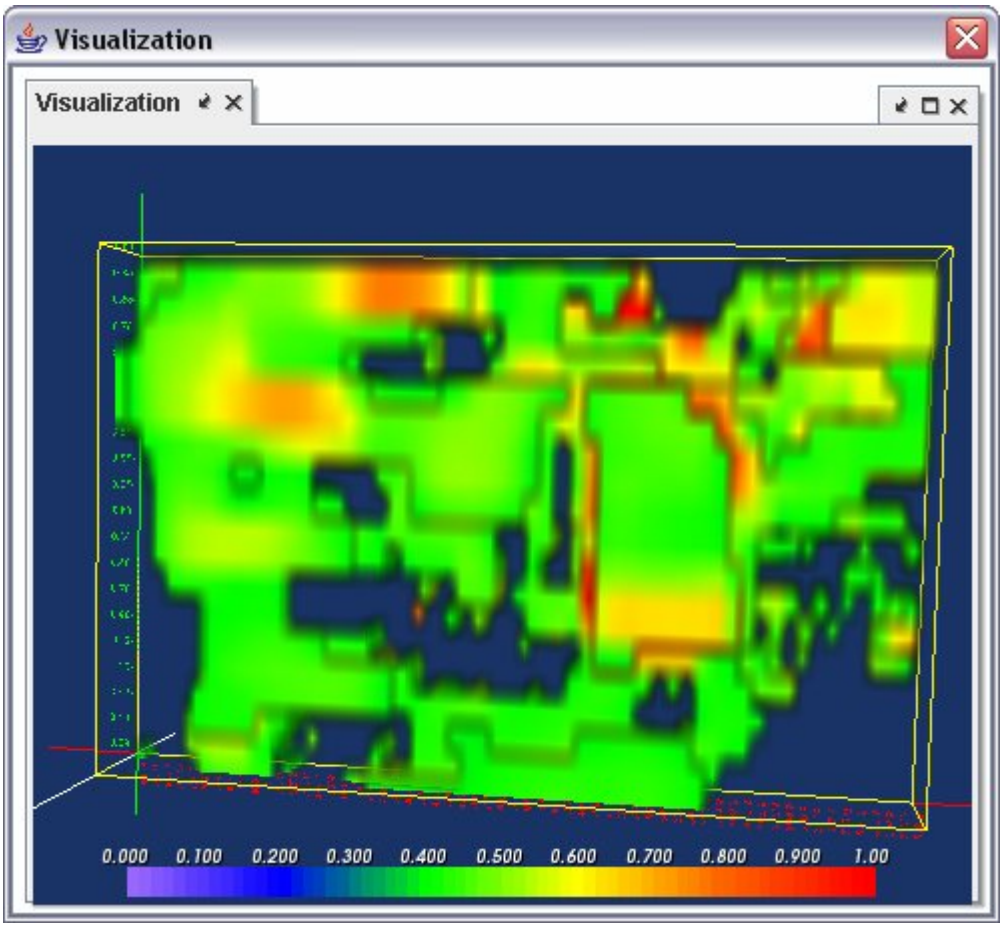


Figure 4.24 Visualization window

In this window user can see the data collected from the historical site, site borders, scalar bar, coordinate numbers and xyz axis. User can interact with the data of the historical site, rotate and zoom it.

4.8.3.2 Cutting Operations Window

This window is shown in Figure 4.25. Cutting operations can be executed in this window. Cutting operations can be done in both x, y, z axis.

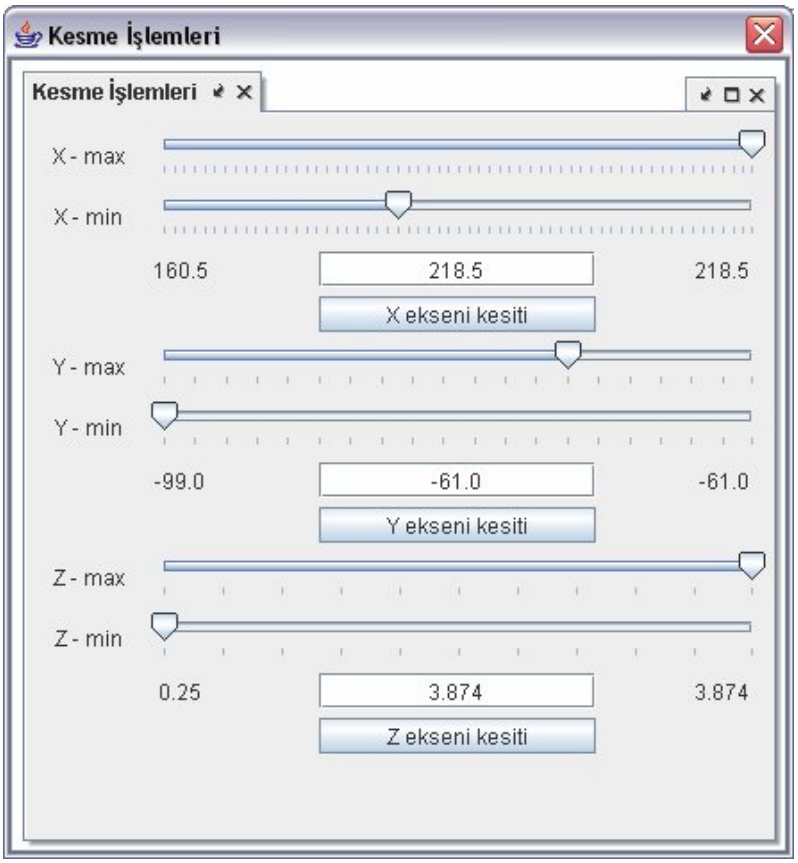


Figure 4.25 Cutting Operations window

The area between the maximum and minimum values are shown and outside of this area is not shown. If a cross section is wanted to be shown, then the buttons, “X ekseni kesiti, Y ekseni kesiti, Z ekseni kesiti” can be used. These button are used to focus the input value of the textfields only.

4.8.3.3 Choose Geophysical Data Window

In Choose Geophysical Data Window, scalar bar and miniumum-maximum visualization color values are shown according the color map file. This window is used to eliminate some color (geophysical data) values to get a better understanding about the historical site. Maximum and minimum input values can be entered by the textfields or moving the slide bars. This window is shown in Figure 4.26.

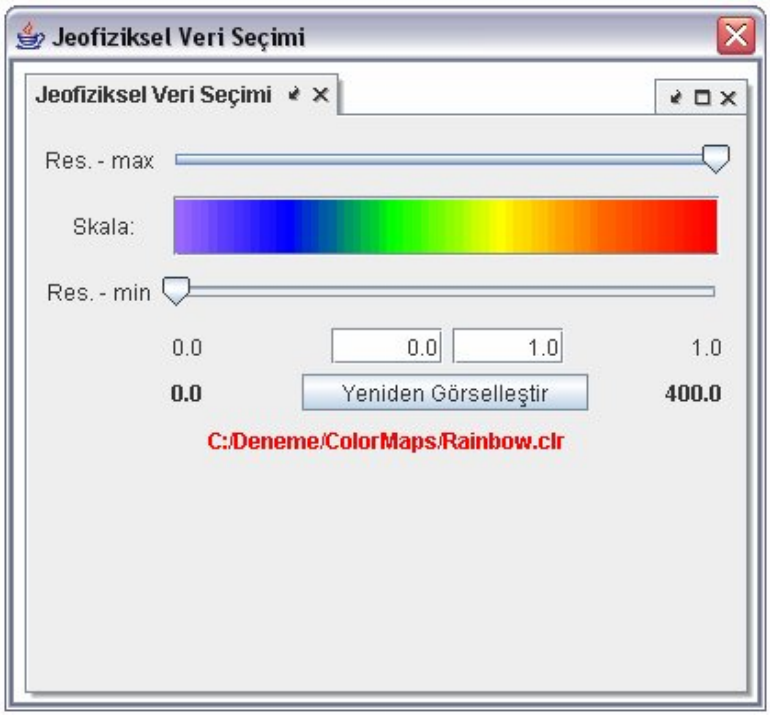


Figure 4.26 Choose Geophysical Data window

After minimum and maximum color values are entered or selected, then the ReVisualize button (Yeniden Görselleştir) must be clicked. When this button is clicked, the visaulization process begins again and so it takes a little time to eliminate the unwanted colors.

4.8.3.4 Operation Shortcuts Window

This window gives user some facilities. Operation Shortcuts Window is shown in Figure 4.27.

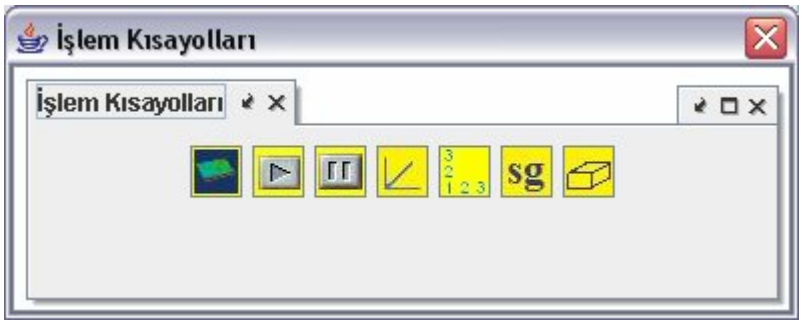


Figure 4.27 Operation Shortcuts window

The shorcuts from left to right are:

- Snap-Shot (Resim Çek); takes a picture at that moment.
- Start Recording Video Process (Video Çekimi Başlat); starts the Recording Video Process which monitors the user's interactions and operations on the Visualization Window and makes a video.
- Finish Recording Video Process (Video Çekimi Bitir); finishes the Recording Video Process.
- Coordinate Lines (Koordinat Çizgileri); enables or disables showing the coordinate lines.
- Coordinate Numbers (Koordinat Sayıları); enables or disables showing the coordinate numbers.
- Standard View (Standart Görünüm); If a Standard view is defined before, this shortcut positions the camera according to that view.
- Site Borders (Site Sınırları); enables or disables showing the site bordes.

4.8.3.5 Data Window

Data Window shows the coordinate and data values of the historical site, as it is shown in Figure 4.28.

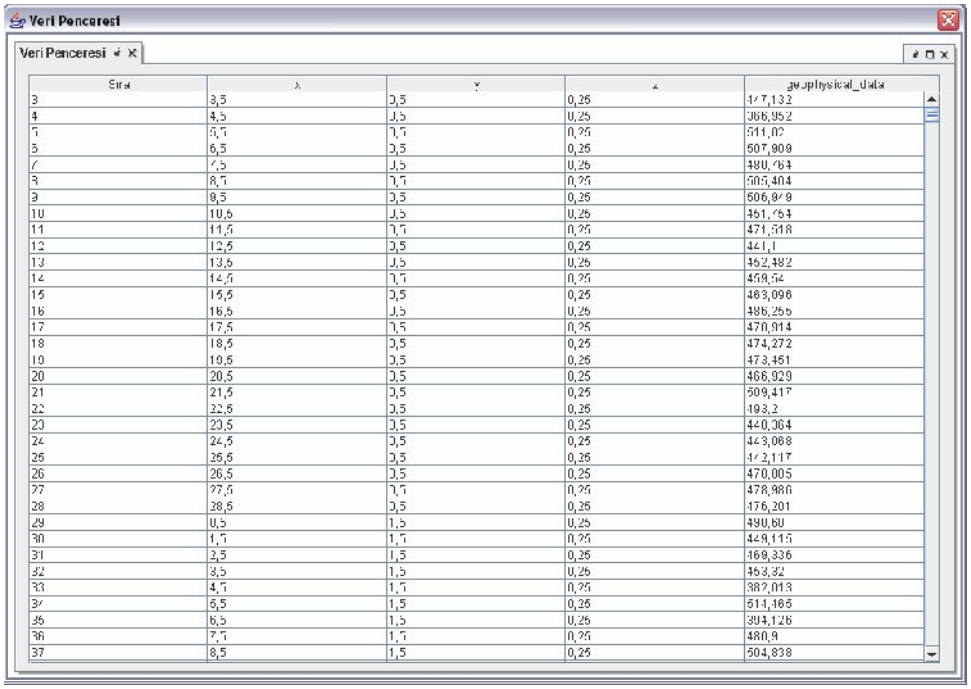


Figure 4.28 Data window

4.8.4 Main Visualization Window Menu

The menu elements of the Main Visaulization Window are shown in Figure 4.29. Menu elements will be explained in detail below sections. Menu elements will be explained in detail below sections.

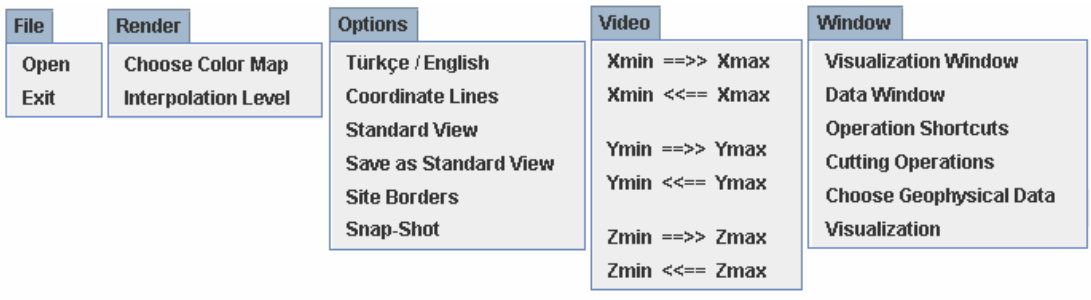


Figure 4.29 Main Visualization Window menu

4.8.4.1 File Menu

This menu provides to open a new historical site source and to close the application.

- Open; this option shows the New Project Screen and gives chance the user to start a new visualization.
- Exit; closes the application and stores the camera and visualization settings in a file to continue the project later.

4.8.4.2 Render Menu

Render Menu provides the user to change the render options; interpolation level and the color map file.

- Choose Color Map; this options shows the Color Map Window and so supports to change color map.
- Interpolation Level; opens the Interpolation Level Window, as it is shown in Figure 4.30. This option provides the user to change visualization level and revisualize the source data.

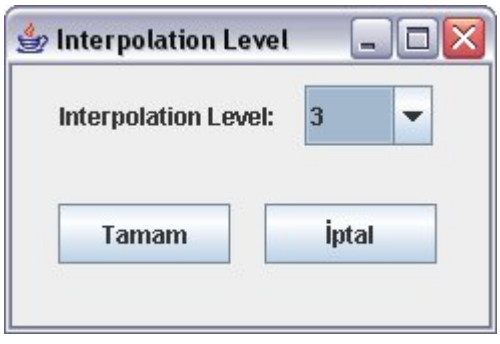


Figure 4.30 Changing Interpolation Level window

4.8.4.3 Options Menu

Options Menu has some choosable options that can be wanted or unwanted by user.

- Türkçe / English; this option changes application language, from Turkish to English or from English to Turkish.
- Coordinate Lines; enables or disables showing the coordinate lines.
- Standard View, if a Standard view is defined before, this shortcut positions the camera according to that view.
- Save as Standard View; if the camera settings are wanted to be kept as standard view, this option must be used.
- Site Borders; enables or disables showing the site bordes.
- Snap-Shot; takes a picture at that moment;

4.8.4.4 Video Menu

Video Menu, give chance to user to use some default recording video options.

- Xmin ==>> Xmax; this option prepares a video by showing all cross sections of x axis, from minimum x value to maximum x value.
- Xmin <<== Xmax; this option prepares a video by showing all cross sections of x axis, from maximum x value to minimum x value.
- Ymin ==>> Ymax; this option prepares a video by showing all cross sections of y axis, from minimum y value to maximum y value.
- Ymin <<== Ymax; this option prepares a video by showing all cross sections of y axis, from maximum y value to minimum y value.
- Zmin ==>> Zmax; this option prepares a video by showing all cross sections of z axis, from minimum z value to maximum z value.
- Zmin <<== Zmax; this option prepares a video by showing all cross sections of z axis, from maximum z value to minimum z value.

4.8.4.5 Window Menu

Window menu provides to enable and show the clicked window on top even if it is closed or minimized before. The elements of this menu are:

- Visualization Window
- Data Window
- Operation shortcuts
- Cutting Operations
- Choose Geophysical Data
- Visualization

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

In this thesis, a 3D imaging and visualization application is developed. This application reads the source file of the historical site and visualize the site in the main visualization window of the application. Before the visualization process, user choose the interpolation and colormap settings and define the maximum geophysical data limit value.

When the visualization process is finished and 3D graphics are drawn in screen, user can interact with the historical site; rotate, zoom and cut it. User can prepare a video or user can take a picture at a preferred time.

If the application is closed before and user wants to continue his or her last project, user can do that, because application saves the settings before it is closed. A Standard camera view can be defined to use, while working with a historical site if a view is important to user. So if camera is moved, user can turn back to the saved view point.

In future, some additions can be made to this application to make the application more user friendly. Maybe, the application can prepare a graphic chart and thus user can see the frequency of the geophysical values of the historical site. Or a route can be defined for the camera, and user can move the camera according to that route in order to see the historical site's inner areas.

REFERENCES

Hansen, C.D., & Johnson, C.R. (2005). *The Visualization Handbook*. Elsevier Butterworth–Heinemann Publications.

Getting Started: Java Wrapping of VTK / ITK and Programming in Eclipse under Windows XP. (n.d.). Retrieved March 2008, from http://user.cs.tu-berlin.de/~marionsc/data/vtkitk/GS-Java_Wrapping_in_ITK_VTK-v2.2.pdf.

Handbook of Medical Imaging: Processing and Analysis. (n.d.). Retrieved 2000, from http://books.google.com.tr/books.

Histogram equalization. (n.d.). Retrieved May 2008, from http://en.wikipedia.org/wiki/Histogram_equalization.

How to Use VTK with Java in Eclipse. (n.d.). Retrieved March 2008, from http://dev.artenum.com/projects/cassandra/forum/how-to-use-eclipse-and-vtk.

InfoNode Docking Windows Developer's Guide Revision 1.3 for IDW Version 1.6.1. (n.d.). Retrieved January 2009, from http://www.infonode.net/documentation/idw/guide/IDW%20Developer%27s%20Guide%201.3.pdf.

Interpolation. (n.d.). Retrieved May 2008, from http://en.wikipedia.org/wiki/Interpolation.

Interpolation Methods. (n.d.). Retrieved May 2008, from http://idlastro.gsfc.nasa.gov/idl_html_help/Interpolation_Methods.html.

Interpolation Methods. (n.d.). Retrieved May 2008, from http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/interpolation/.

Avila, L.S., Barre,S., Geveci, B., Henderson, A., Hoffman, W.A., Law, C.C., Martin, K.M., Schroeder, W.J., & Wanzelek, A. (2001). *The VTK User's Guide*, Kitware, Prentince Hall.

Nabble – VTK forum. (n.d.). Retrieved March 2008, from http://www.nabble.com/VTK-f14272.html.

VTK - The Visualization Toolkit. (n.d.). Retrieved March 2008, from http://www.vtk.org/.

Schroeder, W., Martin, K., & Lorensen, B. (2002). *The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics*, Kitware, Prentince Hall.