Generation and Animation of Perspective Views

Miss Korrapin Atichaichotikul
Miss Piyana Sukanghong

A Project Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi
Academic Year 2006

Generation and Animation of Perspective Views

Miss Korrapin Atichaichotikut
Miss Piyana Sukanghong

A Project Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi
Academic Year 2006

Project Title	Generation and Animation of Perspective Views
Project Credit	4 credits
Project Participant	Miss Korrapin Atichaichotikul
	Miss Piyana Sukanghong
Advisor	Sally E. Goldin, Ph.D.
Degree of Study	Bachelor's Degree
Department	Computer Engineering
Academic Year	2006

Abstract

The most common picture about terrain is 2D such as photographs, satellite images, and maps. A two dimensional representation has only a single viewpoint so it does not provide enough information for a user. User will find it hard to understand the detail of image such as height, curve, deepness, width, shape, etc.

Understanding the terrain landforms is important for many tasks such as planning the construction on any area or planning path. Thus, this project created a program that we call “3D Perspective Views” to solve these problems. This program presents the terrain model in 3 dimensions. It clearly shows the 3D structure, similar to real terrain by using DEM file to create 3D Model and satellite images to give detail of landform. User can rotate the model to make it possible to see this model in different views. Furthermore, user can adjust the display by adjusting the vertical scale, zooming in and zooming out. The purpose of this project is to provide more details of terrain to people who interested in the geology and terrain study.

This program is created by using C++ language in MingW environment and OpenGL library as a major development library. The user interface is developed by using Java programming language. This program can be used in both Windows and Linux operating system.

หัวข้อโครงงาน	การสร้างและการทำวัตถุเคลื่อนไหวในหลายมุมมอง
หน่วยกิตของโครงงาน	4 หน่วยกิต
จัดทำโดย	นางสาวกรพินธ์ อติชัยโชติกุล
	นางสาวปิยะณา สุกางโฮง
อาจารย์ที่ปรึกษา	Sally E. Goldin, Ph.D.
ระดับการศึกษา	วิศวกรรมศาสตรบัณฑิต
ภาควิชา	วิศวกรรมคอมพิวเตอร์
ปีการศึกษา	2549

บทคัดย่อ

ภาพโดยทั่วไปมักถูกนำเสนอในรูปแบบของภาพ 3 มิติ ซึ่งมันจำกัดมุมมองในการมองภาพของผู้ใช้งาน ทำให้ผู้ใช้งานไม่สามารถเข้าใจและรับรู้รายละเอียดของภาพได้อย่างเพียงพอ โดยเฉพาะอย่างยิ่งภาพแผนที่ทางภูมิศาสตร์ เมื่อถูกนำเสนอในรูปแบบของภาพ 2 มิติ มันถูกนำเสนอเพียงมุมมองเดียว ผู้ใช้งานจะทำความเข้าใจในรายละเอียดของภาพได้ยาก เช่น ความสูง ความลาดชัน ความลึก ความกว้าง ลักษณะพื้นที่โดยรอบและอื่นๆ อีกมากมาย ซึ่งการทำความเข้าใจเกี่ยวกับลักษณะทางภูมิศาสตร์มีความสำคัญมากในการทำกิจกรรมหลายๆ อย่าง อาทิเช่น การวางแผนสร้างสิ่งก่อสร้างบนพื้นที่ต่างๆ การวางแผนเส้นทางการเดินทาง เป็นต้น เราจึงสร้างโปรแกรมที่เรียกว่า 3D Perspective Views ขึ้นมา เพื่อแก้ปัญหาดังกล่าว ซึ่งโปรแกรมนี้สามารถนำเสนอภาพเป็น 3 มิติ ทำให้มองเห็นโครงสร้างของภาพได้อย่างชัดเจน มีการจำลองภาพเสมือนของจริง โดยการนำ DEM file มาสร้างเป็นแบบจำลอง 3 มิติ และใช้ภาพ Satellite image ในการให้รายละเอียดของลักษณะพื้นผิวโลก ผู้ใช้งานสามารถหมุนเพื่อดูวัตถุที่ถูกบดบังอยู่ สามารถมองเห็นภาพได้หลายมุมมอง และยังสามารถปรับแต่งการแสดงผลด้วยการขยายหรือย่อขนาด และปรับแต่งความสูงของแบบจำลองได้ โดยจุดมุ่งหมายของโปรแกรมนี้คือเพื่ออำนวยความสะดวกและสามารถให้รายละเอียดของลักษณะทางภูมิศาสตร์แก่ผู้ที่ต้องศึกษาพื้นที่ภูมิศาสตร์ให้มากที่สุดเพื่อนำไปใช้ประโยชน์ต่อไป

โครงงานนี้ถูกสร้างขึ้นโดยใช้ภาษา C++ และ OpenGL library เป็นหลักในการเขียนโปรแกรม ส่วนหน้าจอการใช้งานสร้างโดยใช้ภาษา Java และ Compile ด้วย MingW Compiler ทำให้โปรแกรมนี้สามารถใช้งานได้ทั้งในระบบปฏิบัติการ Windows และ UNIX

Acknowledgement

This project is successful since the help of many people who made our working easier with words of encouragement and more knowledge to expand theories and ideas.

The most important person are Sally E. Goldin, Ph.D. our advisor and Kurt T. Rudahl, M.Sc. who gave us many concepts, knowledge, support, suggestion, including check and edit all documents which have a lot of incorrect grammars, vocabulary, and strategy for writing academic documentation. We have a lot of thanks for both of them.

We would also like to gratefully acknowledge Mr.Pichayapong Tontikul and Mr.Wiroon Srisomburananont who gave us the consult and the suggestion for using OpenGL library in different parts of program. We can solve many problems by using his suggestion and help. Thank you.

Thanks persons who created many compiler, program and open-source library such as MingW compiler, EditPlus, Eclipse, OpenGL library, Java library and OpenDragon. These are facility to help our project accomplished.

Finally, very special thanks to our parent and family for give us a life, a good bring up, always take care, support and encouragement. We are really thanks for them.

Contents

Pages

Chapter 1 Introduction 1

1.1 Background 2

1.2 Objectives 2

1.3 Advantages 3

Chapter 2 Related Theories 4

2.1 Digital Elevation Model (DEM) 4

2.2 Texture Mapping 4

2.3 Animation 5

2.4 Satellite Image 6

2.4.1 Types of Satellite Images 6

2.4.2 Varying Resolution 8

2.4.3 Spectral Bands 8

2.5 Graphics Library 10

2.6 Algorithms 10

2.6.1 Coordinate Representation 10

2.6.2 Polygon Surface Representation 11

2.6.3 Texture Mapping Algorithm 15

Chapter 3 Tools 21

3.1 OpenGL 21

3.2 C and C++ 22

3.3 Java 22

3.4 MingW and MSYS 23

3.5 OpenDragon 24

3.6 The OpenDragon Toolkit Libraries (“libdtk”) 24

3.6.1 Reading Dragon Image 24

Pages

Chapter 4 Approaches 25

4.1 Requirements 25

4.2 Use Case Diagram 25

4.3 Use Case Narratives 26

4.4 Flow Chart 28

4.5 Detail Design 30

Chapter 5 Experiment 32

5.1 Create Simple 3D Model 32

5.2 Texture Mapping 33

5.3 Rotation of the 3D Model 36

5.4 Translation of the 3D Model 36

5.5 Zooming In and Zooming Out 37

5.6 Adjustment of Vertical Scale 39

5.7 User Interface 40

5.8 Connection between the User Interface and the Display 43

Chapter 6 Conclusions and Discussions 44

6.1 Conclusions 44

6.2 Discussion 44

6.3 Skills and Knowledge from the Senior Project 45

6.4 Problems 46

6.5 Suggestion 47

References 48

List of Figures

		Pages
Figure 1.1	General idea of Perspective View	1
Figure 2.1	Texture Mapping	5
Figure 2.2	Visible Satellite Image	7
Figure 2.3	Infrared Satellite Image	7
Figure 2.4	Microwave (Radar) Satellite Image	8
Figure 2.5	Polygon Surface Representations	11
Figure 2.6	The Vector Normal to the Surface of the Plane of the Polygon	13
Figure 2.7	A 3D Triangle Mesh	14
Figure 2.8	A Quadrilateral Mesh	15
Figure 2.9	Two-dimensional Texture-space Coordinates	17
Figure 2.10	Coordinate Reference Systems for 2D (s,t) Texture-space, Object-space, and Image-space	17
Figure 2.11	Texture Mapping by Projecting Pixel Areas to Texture-space	18
Figure 4.1	Use Case Diagram	25
Figure 4.2	Flow Chart	29
Figure 4.3	Module	31
Figure 5.1	AspenDem.img: Simple 3D Model	33
Figure 5.2	Texture Mapping in a True Color Mode	34
Figure 5.3	Texture Mapping in a Gray Scale Mode (Green Band)	35
Figure 5.4	Texture Mapping in a False Color Mode	35
Figure 5.5	Rotation of the 3D Model	36
Figure 5.6	Translation of the 3D Model	37
Figure 5.7	Zooming In	38
Figure 5.8	Zooming Out	39
Figure 5.9	Adjustment of Vertical Scale	40
Figure 5.10	User Interface	40

Pages

Figure 5.11	File Selection Box	41
Figure 5.12	Warning Message	42
Figure 5.13	Display Window	42

List of Table

		Pages
Table 2.1	Spectral Sensitivity of Landsat 7 Bands	9

Chapter 1

Introduction

The most common picture about terrain is 2D such as photographs, satellite images, and maps. A two dimensional representation has only a single viewpoint so it does not provide enough information for a user. We will present a 3D model instead 2D to provide more information by using techniques in the fields of computer graphics and computer animation.

The software developed in this project creates a 3D model starting by simulating a 3D model of the terrain from a digital elevation model (DEM). Then it uses texture mapping methods to map the pattern from satellite image to 3D model surface to show more information about terrain for the user. When the user gets a 3D model that has a realistic appearance, afterwards we can rotate it by using 3D animation to make it possible to see this model in different views. This result will show terrain as perspective views. User will get more understanding about this terrain such as in Figure 1.1 which shows general characteristics of terrain.

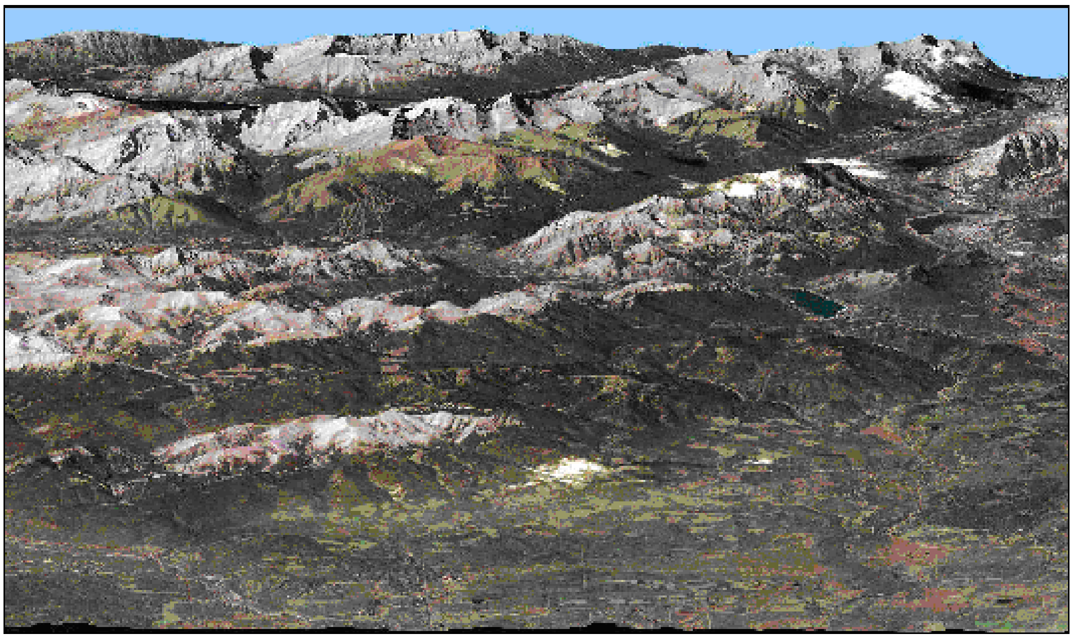


Figure 1.1 General Idea of Perspective View

[Source: http://www.photosat.ca]

1.1 Background

Understanding the terrain landforms is important for many kinds of planning and development tasks. For example, if you want to build a resort on a hill, you must understand and have enough information about this location where you desire to locate. You must know the geography of your location, distance for traveling, and dangers that might happen in this area such as landslide. In general the geographic images (such as maps, photographs or satellite images) can not give enough information because they are presented in two dimensions. These are represented as viewed from one direction. This limits what you can see in the image and makes it hard to understand the details of image, especially the general shape of the land.

Hence this project created a 3D model of the terrain to solve these problems by using OpenGL to generate this entire model and using MingW for compiling and running the program. This model can display properties of the terrain such as width, height, deepness, shape, curve, etc. Moreover, it can be rotated to bring more aspects of the image into view. This model can help people who would like to study the terrain to get more understanding than a normal image because it is like a simulation from the real landform.

1.2 Objectives

1. To create a simulated 3D model of the terrain from a digital elevation model (DEM) to improve visualization.
2. To use a satellite image to provide textural and color detail on top of the 3D model.
3. To animate the 3D model so that a user can rotate it in real time to display the area from different points of view.
4. To learn about 3D computer graphics and animation.
5. To learn about the basics of digital raster data: DEMs and satellite images.

1.3 Advantages

1. This 3D model will make it easier to understand the terrain than the normal picture, so it will be useful for people who want to study the terrain.
2. It looks more realistic than normal picture.
3. It can let the user see a perspective view.
4. It can be used to analyze the structure of area that will be useful for several activities such as planning the construction on the hill or planning paths.

Chapter 2

Related Theories

2.1 Digital Elevation Model (DEM)

A digital elevation model or DEM is represented in digital format, that is, numerical, ASCII or binary file. This digital format consists of detail about the elevation of the earth's surface from a known level or sea level. The DEM contains only spatial elevation data in a regular grid pattern in raster format. It can be applied for many applications such as analysis for creating a 3D model. DEM data are widely available for many locations on earth.

A DEM can be produced by many methods such as digitizing existing contour map, or stereo analysis of aerial photographs or satellite images. Most DEMs are created by remote sensing with satellites. However, a DEM is different from a satellite image. A DEM gives information about the elevation at each point/pixel/grid cell, while a satellite image gives information about ground cover such as a river, a lake, a mountain, a wood etc.

In this project, a DEM is used to generate a structure of 3D model. After that it will be a simple 3D model.

2.2 Texture Mapping

Texture mapping is the method for adding a pattern or texture to a simple 3D model surface. It is the same idea as to paint a picture onto a sketch. Texture mapping makes the simple shapes have a very realistic appearance. This allows a complicated pattern of the surface without adding complexity to the simple shape to represent a more details.

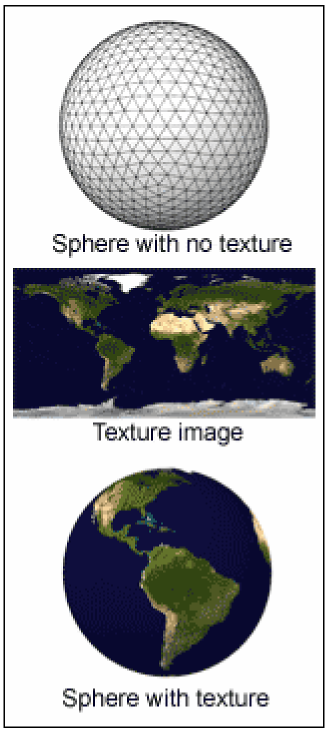


Figure 2.1 Texture Mapping

[Source: http://en.wikipedia.org/wiki/Image:TextureMapping.png]

This project uses texture mapping methods to map the pattern from satellite image to 3D model surface.

2.3 Animation

Computer animation is creating a moving image by using the computer. It relates to the theory of computer graphics and animation. Computer graphics uses mathematics to create the object that the user wants to display. Animating is moving something which can not move itself. Computer animation can use 3D computer graphic or 2D computer graphics, but in this project we concentrate on 3D computer graphics.

Computer animation uses different techniques to create animation. The animator fills the detail of the exact motion by using boolean operations on regular shapes; this technique is called "constructive solid geometry". Another technique uses mathematical algorithms to generate a motion. The animator chooses the appropriate algorithms and select necessary values such as initial values and boundary values. The motion of the object is controlled by the algorithms.

In this project we will use mathematical algorithms to control the motion for rotating the object. This allows the user to move the viewpoint to understand the data of 3D in different views and retrieve the information of the occluded objects.

2.4 Satellite Image

A satellite image is a photograph from a satellite that is a form of remote sensing. Remote sensing is defined as the science and technology by which characteristics of objects of interest that can be identified, measured or analyzed without direct contact with the object.

Generally remote sensing uses measurements of reflected or emitted electromagnetic radiation of the terrain of interest in a certain frequency domain (Visible Light, Infrared, Microwaves). Some systems use sound waves to collect the information in the same way, and others measure variations in gravitation and magnetic fields.

A satellite image can describe the details of geography such as a river, a lake, a mountain, a wood etc.

2.4.1 Types of Satellite Images

There are several types of satellite images.

2.4.1.1 Visible Satellite Image

A visible satellite image records visible light from the sun that reflects to the satellite. Visible light is roughly wavelengths between 400 and 700 nanometers. Some people may be able to perceive wavelengths from 380 to 780 nanometers. A visible satellite image can be represented as a gray scale image so that it can be seen with the eye. You can use a computer to change it to different shades of gray or different colors.

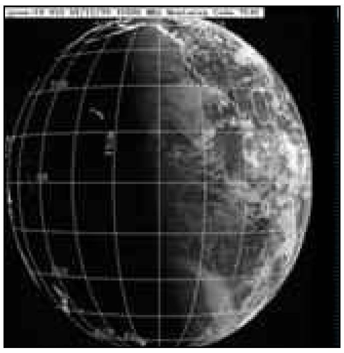


Figure 2.2 Visible Satellite Image

[Source: http://en.wikipedia.org/wiki/Main_Page]

2.4.1.2 Infrared Satellite Image

Infrared satellite image recorded by using invisible infrared radiation reflected or emitted from the earth and captured by a satellite. Infrared radiation is in the wavelength range from 710 nanometers to 1 millimeter. Many weather satellites measure infrared radiation. You can use a computer to change it to different shades of gray or different colors.

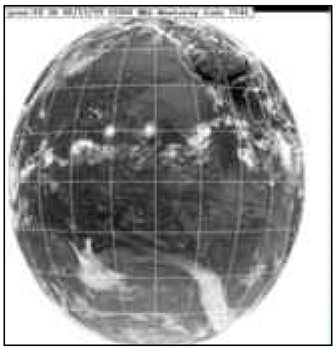


Figure 2.3 Infrared Satellite Image

[Source: http://en.wikipedia.org/wiki/Main_Page]

2.4.1.3 Microwave (Radar) Satellite Image

Microwave (Radar) satellite image recorded by using microwave region of the spectrum. Microwave satellites are usually active sensors; they send a beam of radiation toward the target (the earth) and measure the radiation reflected back.

Microwave radiation is in the wavelength range from 30 centimeters to 1 millimeter. Microwave sensors can get images at night and see though clouds.

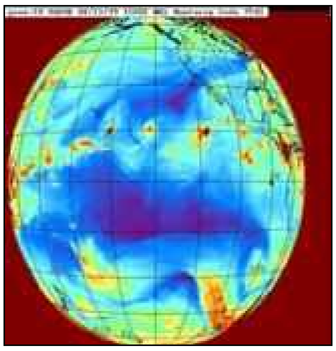


Figure 2.4 Microwave (Radar) Satellite Image
[Source: http://en.wikipedia.org/wiki/Main_Page]

2.4.2 Varying Resolution

The resolution of satellite images depends on the equipment for creating the image and the distance from the ground to the satellite's orbit. The resolution refers to the size of pixel on the ground. If an image has a resolution of x meters per pixel, you will usually not be able to see any objects smaller than x meters. For example, if image is 100 pixels wide and 100 pixels high and has resolution 20 meters per pixel, the real area shown in the image will be 2000 meters wide and 2000 meters high.

High resolution means the each pixel of image has a smaller size on the ground. Then, in one pixel you will see the detail more clearly than for low resolution, because in low resolution, one pixel has a bigger size on the ground.

Satellite images have less resolution than aerial photography, but they are cheaper to acquire than aerial photography. Modern satellite images vary in resolution from 1 kilometer per pixel to less than one meter per pixel.

2.4.3 Spectral Bands

Spectral bands are a collection of image data at different specific wavelengths. These data can be used to build an image. The satellite equipment is able to obtain many images of the same location, at the same time in different spectral bands. Each image is from a different wavelength range in the spectrum of reflected or radiated electromagnetic energy. For example, the Landsat 7 satellite can obtain an image using satellite instruments that collects seven images at once. Each image shows a specific section of wavelengths (called a band) at the same location, at the same time.

Table 2.1 Spectral Sensitivity of Landsat 7 Bands

Band Number	Wavelength Interval	Spectral Response
1	0.45-0.52 µm	Blue-Green
2	0.52-0.60 µm	Green
3	0.63-0.69 µm	Red
4	0.76-0.90 µm	Near IR
5	1.55-1.75 µm	Mid-IR
6	10.40-12.50 µm	Thermal IR
7	2.08-2.35 µm	Mid-IR

These seven images are not the same images. Light and dark area in the images appears in different area. This is because different objects on earth surface reflect different wavelengths. The different dark and bright area in each band can be analyzed to identify the area such as vegetation area, soil area, ocean and etc.

Traditionally, a single image in one wavelength range will be viewed as a gray scale image. However, a color image can be produced by combining three different black and white images and comparing wavelengths and RGB in computer to change it in different colors. This color is false color because the colors that appear are not normally the colors that exist on the ground. This color is represented by different wavelengths. A true color image can be produced by combining and mapping the true band with primary color (RGB), mapping Red band to red, Green band to green and Blue band to blue.

This project uses a satellite image to map the pattern by texture mapping method onto simple 3D model surface.

2.5 Graphics Library

A computer graphics library is a collection of functions designed to aid creating objects in computer graphics. This also provides optimized versions of functions that handle common rendering tasks. This can be operated in software and running on the CPU, common in embedded systems, or be hardware accelerated by a Graphics Processing Unit or GPU, more common in PCs. The library functions can process an image to be output for displaying on the monitor. This relieves the programmer of the task of creating and designing functions of graphics program and allows them to focus on building the graphics application. There are many graphics libraries available for programmers such as OpenGL, Direct3D, GD, GKS or MESA.

2.6 Algorithms

2.6.1 Coordinate Representation

General graphic packages specify scenes in a standard of right-handed Cartesian-coordinate reference frames. Several different Cartesian-coordinate reference frames are used in the process of constructing and displaying a scene.

For the individual object such as a bird or a tree, each object has the own coordinate reference frame at the origin (0, 0, 0).These references frames are called object coordinates, modeling coordinates, local coordinates or master coordinates.

The individual object will be taken to the appropriate location that maybe has another individual object in the same scene. This scene reference frame is called world coordinates. For example, a bird will be located in the sky and a tree will be located on the ground in the same scene. A bird and a tree will have a new position that is referenced by world coordinates.

To display the object, the world coordinates must be converted to viewing coordinates to show the desired view we want of the scene. World coordinates are based on the position and orientation for a view plane that corresponds to a camera film plane. The object locations are transformed to a 2D projection of the scene. The scene is stored in normalized coordinates (the graphic representation which is independent of the coordinate range for any specific output device).

To render the object on a device, the normalized coordinates that are referred to normalized device coordinates are converted to show on the display device. The coordinates systems for display devices are called device coordinates or screen coordinates.

2.6.2 Polygon Surface Representation

Polygon surface representation is the most commonly used method in computer graphic systems. It simplifies and speeds up the surface rendering and display of objects. It uses a boundary representation that is a set of surface polygons that enclose the object interior.

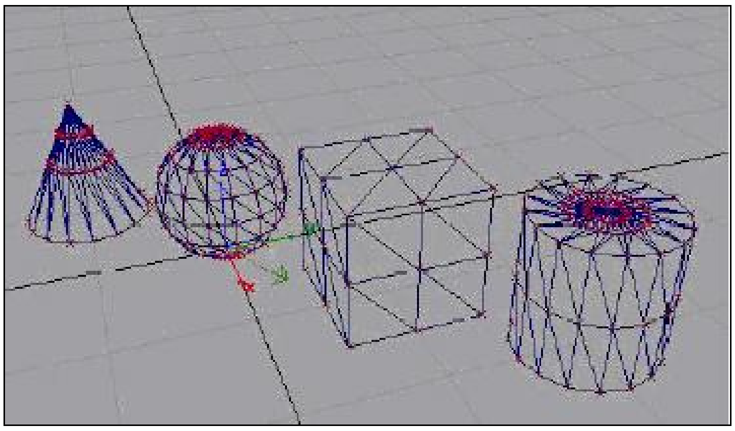


Figure 2.5 Polygon Surface Representations
[Source: http://escience.anu.edu.au/ lecture/ cg/surfaceModeling/polygon.en.html]

2.6.2.1 Polygon Table

A polygon surface is represented with a set of vertex coordinates and associated parameters. Each polygon is input. They are placed into a data structure that combines geometric tables, topological tables and attribute tables. Geometric tables contain vertex coordinates for each polygon. Topological tables contain the information to arrange polygons, that is, the relationships between polygons. Attribute tables contain the information specify the degree of transparency of the object and texture characteristics.

Geometric data is stored in three lists as a vertex table, an edge table and a polygon table. The coordinate values for each vertex are stored in a vertex table. An edge table contains information of edges for each polygon, defined with vertices that determine endpoint of each edge. This also points to the vertex table. A polygon table contains edges for each polygon that can point to edge table

From these three tables, we can check for errors by these conditions.

1. Every vertex must be a component of at least two edges.
2. Every edge must be a component of at least one polygon.
3. Every polygon must be closed.
4. Every polygon must have at least one edge that is shared with other polygon.
5. Every edge in the edge table must point to the polygon that uses it.

2.6.2.2 Plane Equation

A display of a three dimension object must consist of many processes such as transformation of the model and world-coordinates descriptions to viewing coordinates and device coordinates, identification of visible surfaces, hidden line or surface and surface rendering procedure such as determine the color or shading. For these processes, we need to know information about orientation of surface components that can be obtain from vertex coordinate values and equations that describe the plane of polygon.

The constant for the plane can be calculated from this equation.

$$Ax + By + Cz + D = 0$$

Where x,y,z is point on each plane, and the coefficients A,B,C,D are constant of the plane. We can find values of these coefficients from using the coordinate values for three noncollinear points in the planes as

(x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3).

We can calculate the constant for the planes from this equation

$$(A/D)x_k + (B/D)y_k (C/D)z_k = -1, k = 1,2,3$$

This equation can be solved by using Cramer's rule as

$$A = \begin{vmatrix} 1, y_1, z_1 \\ 1, y_2, z_2 \\ 1, y_3, z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1, 1, z_1 \\ x_2, 1, z_2 \\ x_3, 1, z_3 \end{vmatrix} \quad C = \begin{vmatrix} x_1, y_1, 1 \\ x_2, y_2, 1 \\ x_3, y_{3,} 1 \end{vmatrix} \quad D = -\begin{vmatrix} x_1, y_1, z_1 \\ x_2, y_2, z_2 \\ x_3, y_3, z_3 \end{vmatrix}$$

This equation can be rewritten as

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$
$$D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$$

The coefficients (A, B, C) can be used to define a vector normal to the plane of the polygon that can describe orientation of a plane surface in space. This vector is shown as in Figure 2.6.

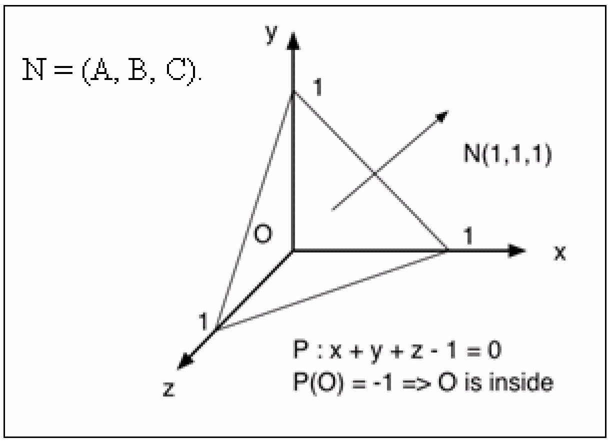


Figure 2.6 The Vector Normal to the Surface of the Plane of the Polygon [Source: http://escience.anu.edu.au/ lecture/ cg/surfaceModeling/polygon.en.html]

If the vertices of polygon are listed in a counterclockwise direction when viewed from outer side of the plane in a right-handed coordinate system (The direction to rotate the object follow by thumb points in the positive z axis direction and the remain finger bend from positive x axis to positive y axis), the direction of vector N will point from inside to outside.

It is possible to determine whether any point is inside or outside the surface of the polygon by using $Ax + By + Cz + D$ equation

If $Ax + By + Cz + D < 0$, the point (x,y,z) is inside the surface.

If $Ax + By + Cz + D > 0$, the point (x,y,z) is outside the surface.

2.6.2.3 Polygon Meshes

A polygon mesh is collection of vertices and polygons that specifies the shape of an object in 3D computer graphics. There are many types of polygon meshes such as triangle mesh, quadrilateral mesh or other simple polygonal mesh.

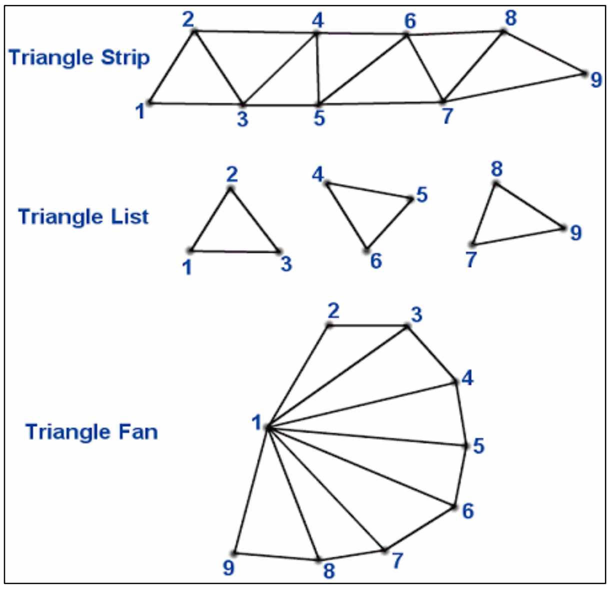


Figure 2.7 A 3D Triangle Mesh

[Source: http://escience.anu.edu.au/lecture/cg/surfaceModeling/polygon.en.html]

A 3D triangle mesh is a collection of triangles and vertices that has three vertices per polygon. They are connected by their edges. It can also be considered as a surface defined by a mesh of triangular facets. There are various ways to represent a 3D triangle mesh such as triangle strip, triangle fan and triangle list.

There will $N-2$ triangles in the triangle strip if N is the number of vertices. Each triangle shares an edge with the previous triangle. A triangles list will produce only $N/3$ triangles and does not have any edges are shared. A triangle fan provide $N-2$ triangles again with sharing the vertex for all triangles polygon

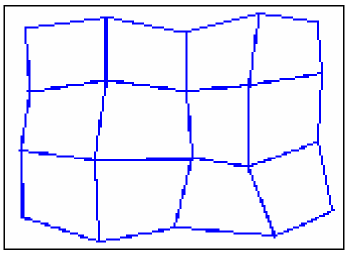


Figure 2.8 A Quadrilateral Mesh

[Source: http://escience.anu.edu.au/ lecture/ cg/surfaceModeling/polygon.en.html]

A quadrilateral mesh is a set of polygons with four sides and four vertices. They are connected by their edges. It provides $(N-1)\times(M-1)$quadrilaterals from $N\times M$ array of vertices

This project uses a triangle mesh because when a polygon has more than three vertices, all vertices may not lie in the same plane. This can cause confusion to calculate because the shape of the polygon may be changed from original polygon. Hence, using triangles mesh is safe and many hardware and software systems support it.

2.6.3 Texture Mapping Algorithm

Texture mapping is a common method to map a pattern onto a simple object.

2.6.3.1 Linear Texture Patterns

A 1D texture pattern is specified in a single-subscript array of color values that defines is a sequence of color in a linear texture space that is referenced with a single s-coordinate value. For example, consider a list of 32 RGB colors which references subscript values from 0 to 95. The first three elements of array store the RGB component of the first color, the next three elements of array store the RGB component of the second color, and so forth. This set of color can be used to form or a color pattern for mapping on a model.

To map a linear texture pattern into a scene it is necessary to assign an s-coordinate value to one spatial position and another s-coordinate value to a second spatial position. The color array in s-coordinate is used to generate a multicolored line between the two spatial positions. Texture mapping uses a linear function to calculate the array positions that are assigned to the pixels to a line segment. A simple color-mapping method is assigned the nearest array color to each pixel. If a pixel is mapped to a position between the starting array components for two colors, it can be computed by a linear combination of the nearest two color components in the array.

2.6.3.2 Surface Texture Patterns

A surface texture pattern is defined with rectangular color pattern and position of texture space that is referenced with 2D (s,t) coordinate values. Each color is stored in a 3-subscript array. For example, if a texture pattern is defined with 16 by 16 RGB colors, the array contain is 16*16*3 = 768 elements.

Figure 2.9 show the 2D (s,t) texture-space coordinates that reference positions in an array of color values containing *m* rows and *n* columns. Each position in an array can reference multiple color components. Values for both s and t vary from 0 to 1. The first row of the array has the color values across the bottom of the rectangular texture pattern and the last row of the array has the color values across the top of the rectangular texture pattern. The position of coordinate (0,0) references the first set of color components at the first position in the first row and the position of coordinate (1,1) references the last set of color components at the last position in the last row of the array.

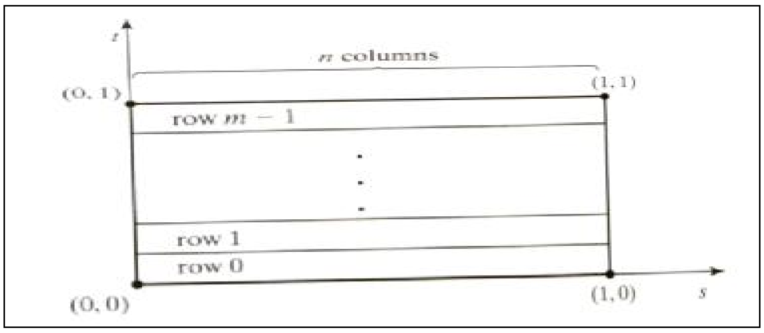


Figure 2.9 Two-dimensional Texture-space Coordinates
[Source: Computer Graphics with OpenGL Third Edition, Donald Hearn and M. Pauline Baker, page 630]

Surface positions on an object can be described with (u,v) object-space coordinates and projected pixel positions with (x,y) Cartesian coordinates.

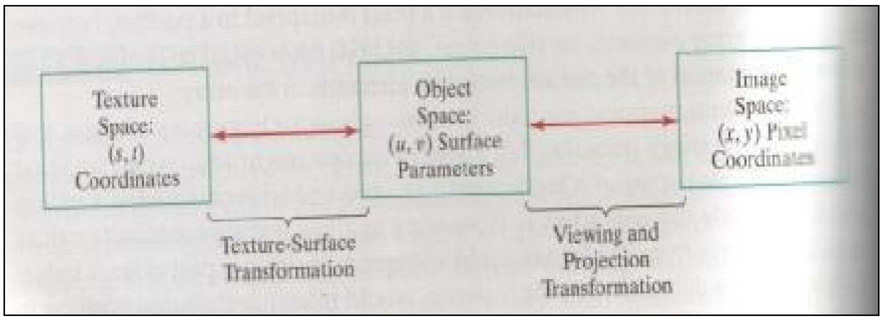

Texture Space: (s, t) Coordinates
Object Space: (u, v) Surface Parameters
Image Space: (x, y) Pixel Coordinates
Texture-Surface Transformation
Viewing and Projection Transformation

Figure 2.10 Coordinate Reference Systems for 2D (s, t) Texture-space, Object-space, and Image-space
[Source: Computer Graphics with OpenGL Third Edition, Donald Hearn and M. Pauline Baker, page 630]

The (s,t) texture-space coordinate for 4 corners of the texture pattern is assigned to 4 spatial positions on scene and a linear transformation is used to assign color value to the projection pixel positions. The parametric linear transformation provides a simple pattern for mapping position in texture space to object space.

$$u = u(s,t) = a_u s + b_u t + c_u$$

$$v = v(s,t) = a_v s + b_v t + c_v$$

$$p(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$$

Where u, v are surface parameters and a point **p** on the surface is a function of two parameter u and v.

The object-to-image-space transformation can be made by connecting of the viewing and projection transformations. A mapping from texture space to pixel space has disadvantages because pixel that we use to patch does not match up with the pixel boundary. It is necessary to calculate the new area of the pixel that covers the texture. So, texture mapping methods mostly use mapping from pixel space to texture space. To avoid pixel subdivision problem, it is necessary to calculate the inverse viewing-projection transformation M_{VP}^{-1} and the inverse texture-map transformation M_T^{-1}.

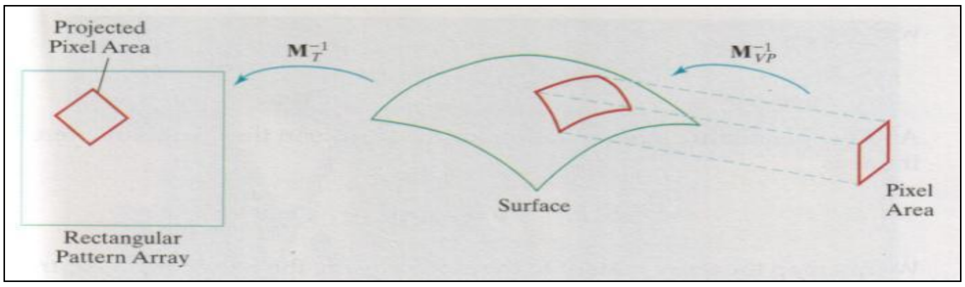


Figure 2.11 Texture Mapping by Projecting Pixel Areas to Texture-space [Source: Computer Graphics with OpenGL Third Edition, Donald Hearn and M. Pauline Baker, page 631]

2.6.3.3 Volume Texture Patterns

A volume texture patterns using 3D texture-space coordinates (s,t,r) and 3D texture space is defined in the unit cube that has texture coordinates ranging from 0 to 1.0. It can be stored in a 4-subscript array, which first subscript is a row position, second subscript is a column position, and next subscript is a depth position. In particular, fourth subscript is used to reference a component of color in the pattern. For example, assume that RGB volume texture pattern with 16 rows, 16 columns, and 16 depth planes. It can store in array is 16*16*16*3 = 12,288 elements.

To map pattern of entire texture space to 3D model, we have several way such as assign the coordinates for 8 corners of texture space to 8 spatial positions on scene of object or map plane of texture space (such as depth plane) to plane in scene.

A volume texture patterns allows internal view (such as cut-away display and cross-sectional slices) or in 3D models to be displayed with texture patterns such as a brick, a cinder block, or a wood materials can have the same texture patterns applied all the spatial area of the models.

2.6.3.4 Texture Reduction Patterns

A texture reduction pattern is used when the size of a texture object is reduced. Reducing the size of the object has the effect of making the texture pattern be applied to smaller area and leads to texture distortions. One way to avoid these problems is to use a set of texture reduction pattern such that each reduction pattern will be one half the size of the previous pattern. For example, if we have 2D 16 by 16 patterns, then we set up four patterns with reduction size at 8 by 8, 4 by 4, 2 by 2, and 1 by 1. For any view of the object, the reduction pattern can be applied to minimize distortions. These techniques are referred to as MIP maps or mip maps that it can translated as “much on a small object”.

Texture reduction maps are used in animation and in many other applications.

2.6.3.5 Procedural Texturing Methods

A procedural texturing method uses a procedural definition for color variations that are applied to the object. It uses calculation of variation for properties or characteristics of objects. This method can avoid transformation calculation in mapping array patterns to objects and decrease storage requirement that necessary when have large texture patterns is applied to scene. For example, wood-grain or marble patterns can be simulated using procedural texturing.

This project uses 2D texture mapping methods to map the pattern from satellite image to 3D model surface to show the detail of landform.

Chapter 3

Tools

There were many tools used in this project. These tools were very helpful. This chapter discusses the tools that were used and the function of each tool that necessary for our project.

3.1 OpenGL

OpenGL is a software interface to graphics hardware. It provides a large and efficient collection of device independent functions for creating computer graphic displays. It is designed to be hardware independent. This interface consists of about 150 distinct commands that are used to specify the objects and operations need to create two dimension and three dimension applications. OpenGL is available on most computer systems and is used widely.

OpenGL is designed for using a program written in C or C++ but it can also be used with other programming language such as Java, Delphi, Perl, Ada or FORTAN.

The general purpose of OpenGL is to provide users with a variety of function for creating polygon meshes and manipulating pictures. It can create primitives object such as points, lines segments or polygons and can manipulate many complex methods such as two dimension and three dimension transformation, viewing method, perspective projection, spline generation, mesh polygon generation, interactive mouse input, surface rendering, texture mapping or animation techniques.

Libraries of OpenGL include the OpenGL Utility Library (GLU), GLU contains basic commands to use in OpenGL. It exists in all OpenGL implementations. However, it is not possible to create the display window with the basic OpenGL functions. Window management depends on the computer used. The OpenGL Extension to the X window system (GLX) provides a set of window management for X window. The OpenGL Utility Toolkit (GLUT) provides a set of function for interacting with any screen window system and contains many commands to helpful creating graphic program by OpenGL. This project used GLUT so that the program would be device independent.

This project used functions of OpenGL for creating mesh polygon, texture mapping and animation.

3.2 C and C++

C and C++ are closely related programming languages developed for use on the UNIX operating system. They are flexible languages because they are independent of hardware. They have the ability of work to be equal to assembly language. They are often used to write system software and applications.

C is a general programming language. C was the C++ predecessor. C is a small compiler and lots of free compilers. The advantages C has over C++ are that the executables are generally smaller and the programs run slightly faster.

C++ is an object oriented programming language. There is a standard library of data classes Standard Template Library (STL) which provides all the standard data structures like linked lists, vectors etc. It has a few extra features. C++ allows the programmer to more easily manage and operate with objects, using an Object Oriented Programming (OOP) concept.

C++ applications are generally slower at runtime, and are much slower to compile than C programs. The low-level infrastructure for C++ binary execution is also larger.

This project used C and C++ languages to create the code for create perspective view.

3.3 Java

Java is an object oriented programming languages developed for use on both Windows and Linux.

Java language is a simple language for learning and using because the most syntax to be similar with C and C++ languages and it is not complicated. If the person has ever used C or C++ languages, they can easily understand the syntax of the Java language.

Java is a robust language because it provides exception handling to handle some mistakes of program while program is running. Moreover, Java has automatic garbage collection, so the programmer does not have to handle memory management.

Java incorporates both interpretation and compilation. The source text file program is compiled to the machine code, called bytecodes, for the Java Virtual Machine (JVM). The JVM simulates a processor that executes the bytecodes instructions. Then the JVM interprets the bytecodes. So, the bytecodes can be run on any platform on which a JVM has been developed.

This project uses the Java language to create the code for the user interface.

3.4 MingW and MSYS

MingW is a development environment that provides UNIX-like capabilities on Windows. In particular, it provides an implementation of the gcc (Gnu C Compiler) and also has commands to support Java language. This project used MingW to create software that can be compiled and run on both Windows and Linux.

MSYS is minimal system. MingW is often used together with MSYS. It provide POSIX/Bourne configure scripts that give the ability to execute and create a Makefile used by make.

MingW was used as the compiler in this project is it has characteristic of portability to other operating system which means it can compile and run on both Windows and Linux.

This is a command for the C language. It compiles hello.c by using OpenGL libraries.

“gcc –o hello.exe hello.c –IC:/mingw32/include /c/lib/libglut32.a /c/lib/libglu32.a /c/lib/libopengl32.a”

This is a command for the C++ language. It compiles hello.cpp by using OpenGL libraries.

“g++ hello.cpp –o hello.exe –IC:/mingw32/include /c/lib/libglut32.a /c/lib/libglu32.a /c/lib/libopengl32.a”

“make” This command useful for compiling programs that include several source files. For this command, it is necessary to have a makefile. Makefile is a collection of instructions that should be used to compile and link your program. The makefile can contain any operating system commands and be used to automate the creation of a complicated application. When you run make, it will look for a file named makefile in your directory and then execute it.

3.5 OpenDragon

OpenDragon is a remote sensing image processing package, which also includes a library of functions to allow users to write their own programs for use with OpenDragon. This project used the OpenDragon library to read and write image files and to display 2D images and DEMs. OpenDragon can combine satellite image in different bands and display the result that make user easier to understand the detail of satellite image and also can show the height value of each pixel on DEM images.

3.6 The OpenDragon Toolkit Libraries ("libdtk")

The OpenDragon Toolkit Libraries consist of several libraries of functions and tool which are used in creating Dragon, and are also useful for developer to write their own programs for use with OpenDragon.

The libdtk modules are written in C and C++. The developer can write own program in either C or C++. The methods to link your program to libdtk are necessary to include "libdtk.h" (libdtk header file) in source code and link to the suitable libraries for your source code at least should be "wfutl.a".

This project used libdtk in our project for module of creating simple 3D model and texture mapping. The input files for our program need to be DEMs and satellite images. Both should be OpenDragon images format (.img). The libdtk provides function that we need for reading OpenDragon image file.

3.6.1 Reading Dragon Image

The sequence for reading a Dragon input file is

1. Open the file and read the header (metadata) information consists of number of rows, number of column and Bit of image.
2. Based on the header information, create a one-line-long data buffer.
3. Cycle through the rows, reading and processing each in turn.
4. Close the input image. (Unnecessary if your program will be exiting immediately.)

Chapter 4

Approaches

4.1 Requirements

1. Platform independent: must be able to compile and run on either Windows or Linux.
2. Can work for any DEM and satellite images expressed in OpenDragon format images.
3. Animation must be smooth, so it looks as though viewpoint is gradually changing.
4. Adjustable vertical scale.

4.2 Use Case Diagram

This section describes use case diagram in our program.

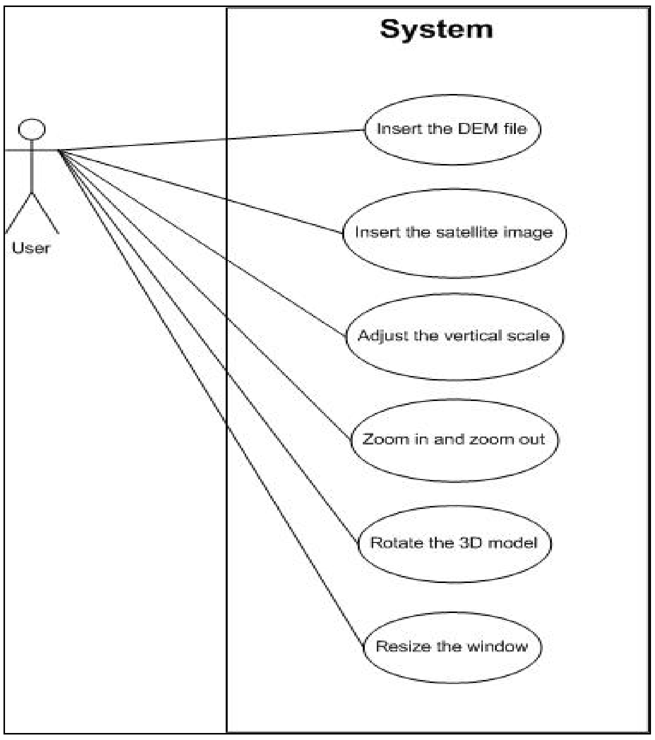


Figure 4.1 Use case Diagram

4.3 Use Case Narratives

This section describes use case narrative in each scenario that can be happened in our program. User need to start with the scenario 1, scenario2, or scenario 3 every time they would like to use this program.

4.3.1 Scenario 1: Open DEM File and Display Output

User would like to see the result from a DEM file only. Not need to select satellite images.

1. Open program
2. User specifies the DEM file which would like to display.
3. User click display button to see simple 3D model.
4. The display window shows simple 3D model.

4.3.2 Scenario 2: Open File and Display Output in Gray Scale Mode

User selects a DEM file and only one band of satellite image.

1. Open program
2. User specifies the DEM file which would like to display.
3. User select one band of satellite image file maybe Red band, Green band or Blue band and specifies a satellite image file from their desirable band.
4. User click display button to see 3D perspective view of terrain model at
5. The interface shows 3D perspective view of terrain model at a gray scale mode.

4.3.3 Scenario 3: Open File and Display Output in True Color Mode

User would like to see 3D model in true color mode. User need to select a DEM file and three bands of satellite images.

1. Open program
2. User specifies the DEM file which would like to display.
3. User specifies a Red band of satellite image file.
4. User specifies a Blue band of satellite image file.

5. User specifies a Green band of satellite image file.
6. User click display button to see 3D perspective view of terrain model at a true color mode.
7. The interface shows 3D perspective view of terrain model at a true color mode.

4.3.4 Scenario 4: Rotate the Terrain Model

After selecting the file to display, user would like to see the terrain model in a different direction.

1. User click mouse on the display window as the start point for rotation and drag a mouse to the direction that they would like to see the terrain. User can drag mouse in free direction. They can drag to the upper side, lower side, right side and the left side.
2. The interface shows the rotation of terrain model start at point of view at mouse clicked to mouse released in the stop point of rotation.

4.3.5 Scenario 5: Adjust Vertical Scale

After selecting the file to display, user would like to see the terrain model in a different vertical scale.

1. User changes a vertical scale by use a mouse to adjust the vertical scale. Dragging mouse to up direction is the increasing vertical scale and dragging mouse to down direction is the decreasing vertical scale.
2. The interface displays the terrain model which is changed the vertical scale.

4.3.6 Scenario 6: Zoom In the Terrain Model

After selecting the file to display, User would like to make a model look bigger.

1. User drags mouse to the right side on display window to control zooming in the terrain model.

2. The interface shows the terrain model at bigger size.

4.3.7 Scenario 7: Zoom Out the Terrain Model

After selecting the file to display, User would like to make a model look smaller.

1. User drags mouse to the left side on display window to control zooming out the terrain model.
2. The interface shows the terrain model at smaller size.
3. Close program by clicking close window button.

4.3.8 Scenario 8: Move the Terrain Model

After selecting the file to display, User would like to move a model to other area in the display window.

1. User drags mouse to the new position.
2. The interface shows the terrain model on the new position.
3. Close program by clicking close window button.

4.4 Flow Chart

This flowchart in Figure 4.2 shows the procedure of the perspective view program.

After receive input file (DEMs and satellite images) from user, the program reads the values from these input files. The value is obtained from DEMs file is used to create geometric mesh. Program continues to read satellite image and use this value to map satellite image to simple 3D model which is the result from creating geometric mesh. The result will be displayed in the display window. If user would like to control the display by zoom in, zoom out, rotating and modifying scale, program will recompute the output and redisplay it. But if user would like to exit program, program will close.

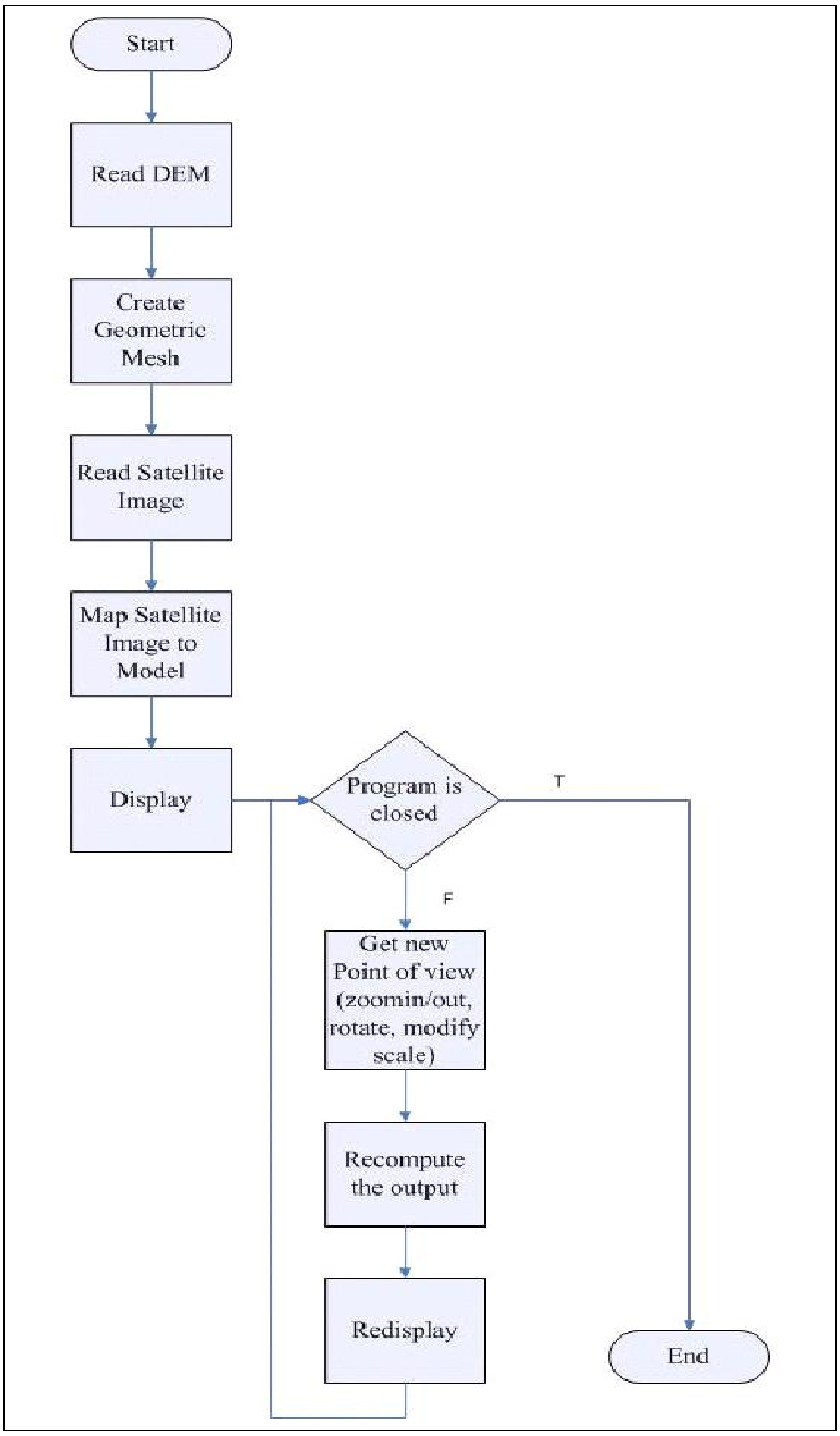


Figure 4.2 Flow Chart

4.5 Detail Design

This detail design shows the modules of the perspective view program. The program is divided into six modules, as follows:

1. Read the DEM and satellite image data is a module for reading and store information from DEM and Satellite images.
2. Create the 3D model is a module for creating the simple 3D model from the vertex data which is obtained by reading DEM data.
3. Texture mapping is module for mapping the satellite image data to simple 3D model. It make 3D model has a detail of landform.
4. Controller is module for handle data in this program. It responses data for adjustment the vertical scale and rotation the 3D model.
5. View is module response for display result from all user's interaction via user interface.
6. Interaction is module that sends value of interaction from user to module which responses for processing**.**

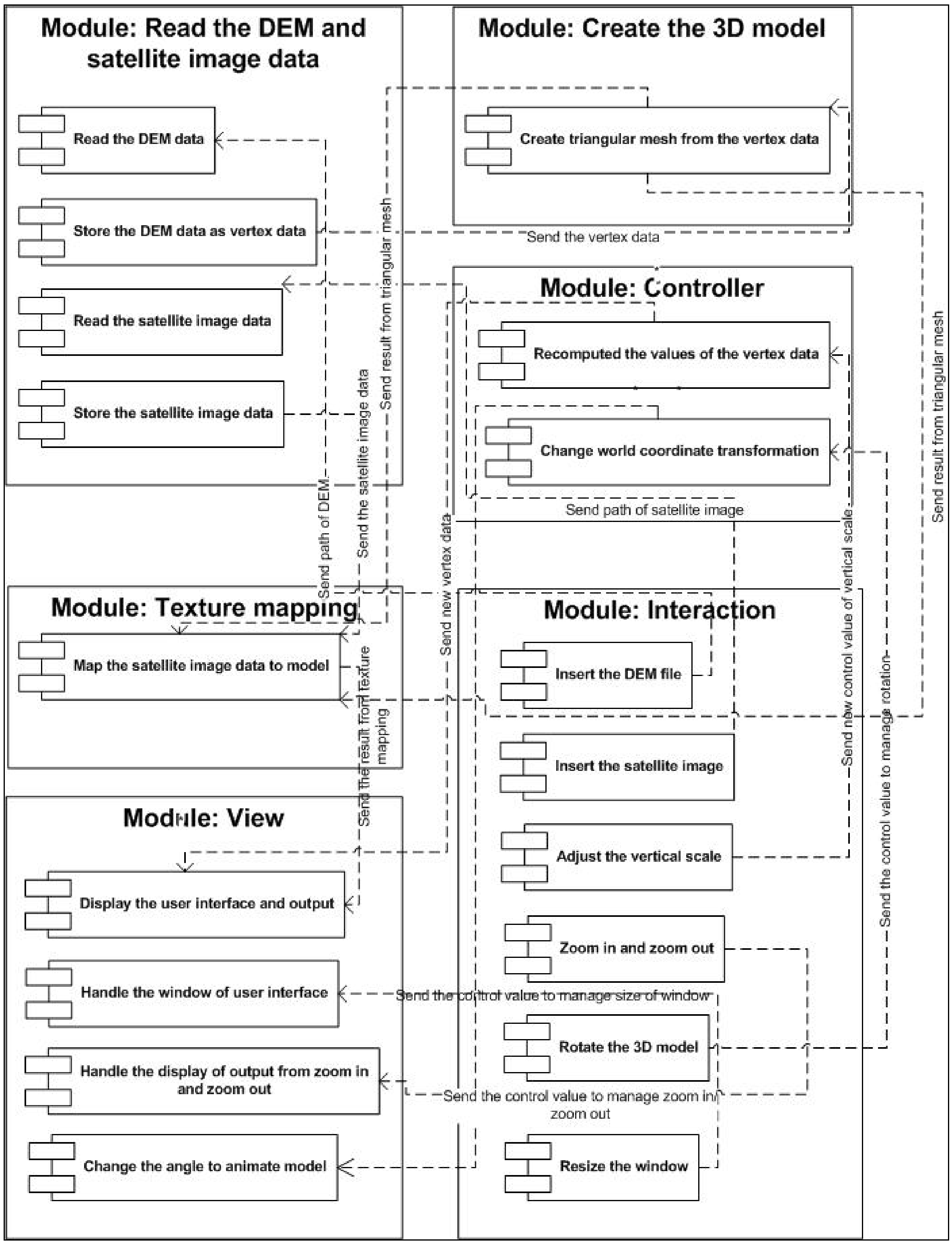

Module: Read the DEM and satellite image data
Read the DEM data
Store the DEM data as vertex data
Read the satellite image data
Store the satellite image data
Module: Create the 3D model
Create triangular mesh from the vertex data
Send the vertex data
Module: Controller
Recomputed the values of the vertex data
Change world coordinate transformation
Send path of satellite image
Module: Texture mapping
Map the satellite image data to model
Module: View
Display the user interface and output
Handle the window of user interface
Handle the display of output from zoom in and zoom out
Change the angle to animate model
Module: Interaction
Insert the DEM file
Insert the satellite image
Adjust the vertical scale
Zoom in and zoom out
Rotate the 3D model
Resize the window
Send path of DEM
Send the satellite image data
Send result from triangular mesh
Send new vertex data
Send the result from texture mapping
Send new control value of vertical scale
Send the control value to manage rotation
Send result from triangular mesh
Send the control value to manage size of window
Send the control value to manage zoom in/ zoom out

Figure 4.3 Module

Chapter 5

Experiment

5.1 Create Simple 3D Model

This part can read 8 bit per pixel and 16 bit per pixel Dragon images files and use value from this image that is value of row, value of column and height value to create Simple 3D model.

The procedure of this part is

1. User sends path of input file (DEM file) to this module.
2. Read input DEM file in Dragon image format (.img) by using libdtk. The libdtk function read information from header of dragon image. This provides the necessary values for the program such as the number of rows, the number of pixels and the data size of image.
3. Once the data size of image is known, the program considers that the DEM file is 8 bit image or 16 bit image and allocates memory for one-line-long data buffer.
4. In each turn, program read height values in all pixels from each row and collects these values to destination data buffer.

The procedure from 2-4 need to use libdtk to read and collect data from DEMs file.

5. The values in data buffer from all turns are collected in variable array of coordinate x, y and z. The value of pixel is x, the value of row is y and the height value is z.
6. These values in variable array of coordinate are used to simulate the simple 3D model. The module for creating simple 3D model uses these values to create triangular mesh. We used OpenGL function to create triangular mesh and represented this as triangle strip.
7. The result from create triangular mesh is displayed the simple 3D model in display window.

The procedure from 6-7 need to use OpenGL functions for creating triangular mesh and display window.

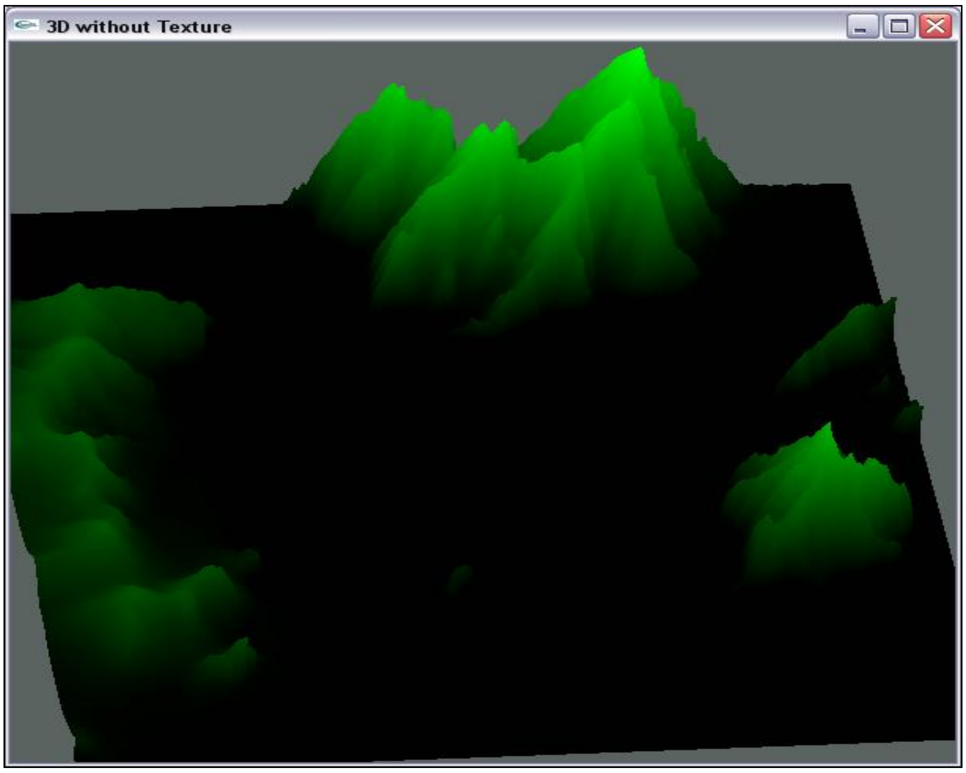


Figure 5.1 SanFranDemSub.img: Simple 3D Model

5.2 Texture Mapping

This part handles about satellite images to map the detail on a simple 3D model. The procedure of this part is

1. User sends path of input file (satellite images files) to read the information from header file and color values on each pixel. This procedure uses the same function as reading DEM file.
2. Satellite images which is used for texture mapping needs to have the same numbers of columns and row as DEM file. Both the numbers of columns and rows must be a power of two. Therefore we must check the number of columns and rows before mapping. If a number of columns and rows from satellite image is not a power of two, we must increase the size to be a power of two. But we do not use addition of image for mapping.
3. After reading values from satellite images and managing the size of image, we will store color values from each pixel in texture array. We use RGB color component that means each color of the texture pattern is

specified with three RGB values. Therefore the size of the array is three * width*height. Texture values from red band of satellite image are stored in R element, texture values from green band are stored in G element and texture values from blue band are stored in B element for each pixel. This gives the result which is displayed in a natural color. If a gathering of texture value from three bands does not follow this, the result will be displayed in a false color mode. However if only one band of the satellite image is input, the result will be displayed in a gray scale mode by giving the same texture value in R, G and B elements for each pixel.

4. Use the texture value in a texture array for mapping on the surface of simple 3D model. The texture value is applied by assigning texture coordinate values to vertex of simple 3D model.
5. The result from texture mapping is displayed in a display window.

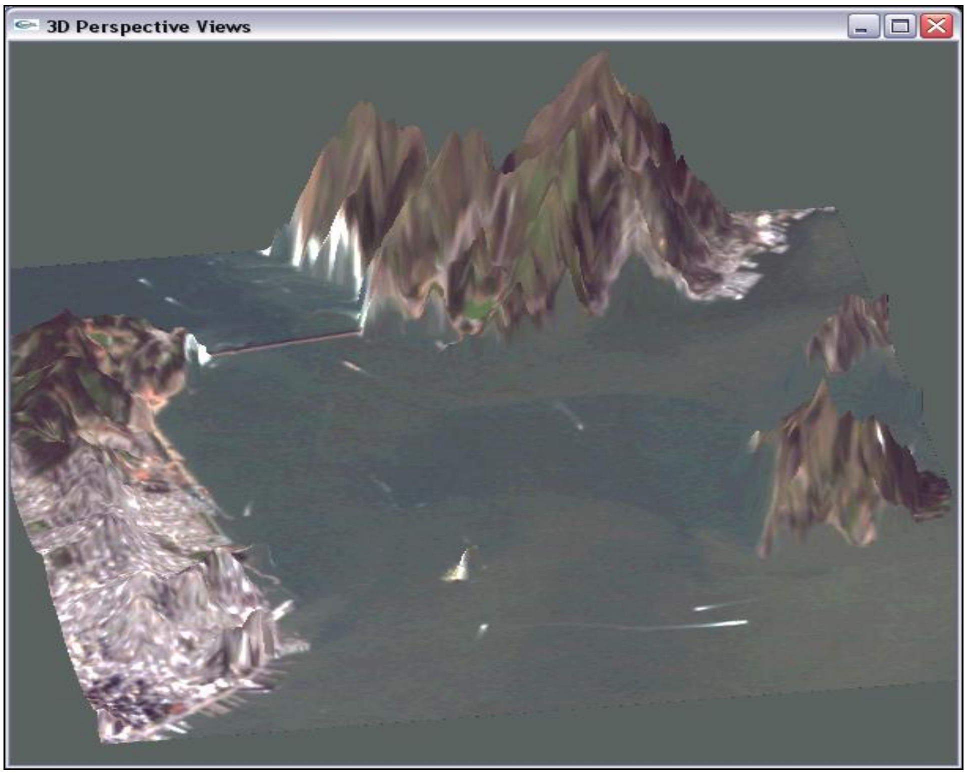


Figure 5.2 Texture Mapping in a True Color Mode

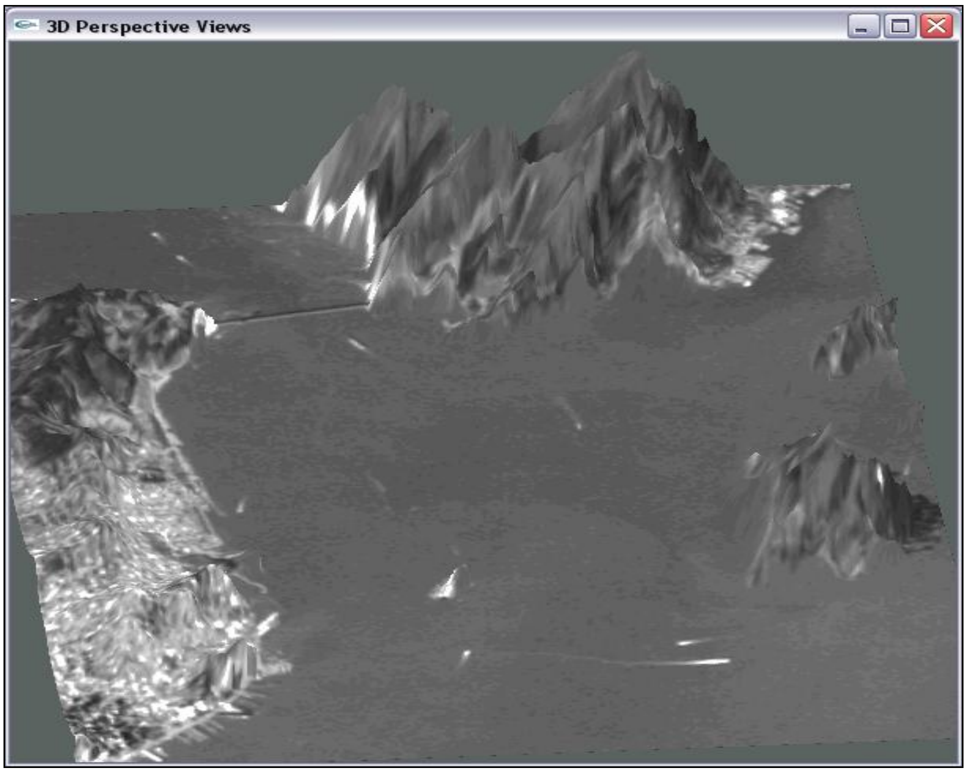


Figure 5.3 Texture Mapping in a Gray Scale Mode (Green Band)

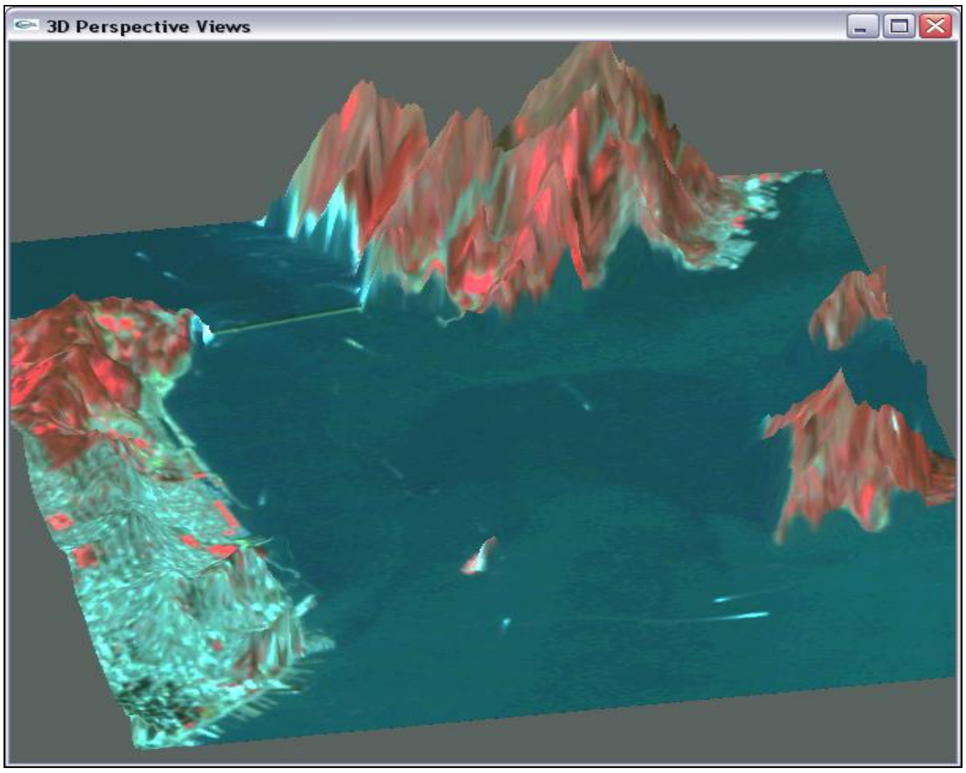


Figure 5.4 Texture Mapping in a False Color Mode

5.3 Rotation of the 3D Model

User can control the rotation of 3D model by mouse. These are steps to control the rotation

1. User must press key R on a keyboard before using the mouse to control the rotation.
2. User drags the mouse to control the rotation on X and Y axis. They can drag a mouse in free direction. Dragging mouse to the upper and lower side controls the rotation on Y axis. Dragging a mouse to the right and left side controls the rotation on X axis.

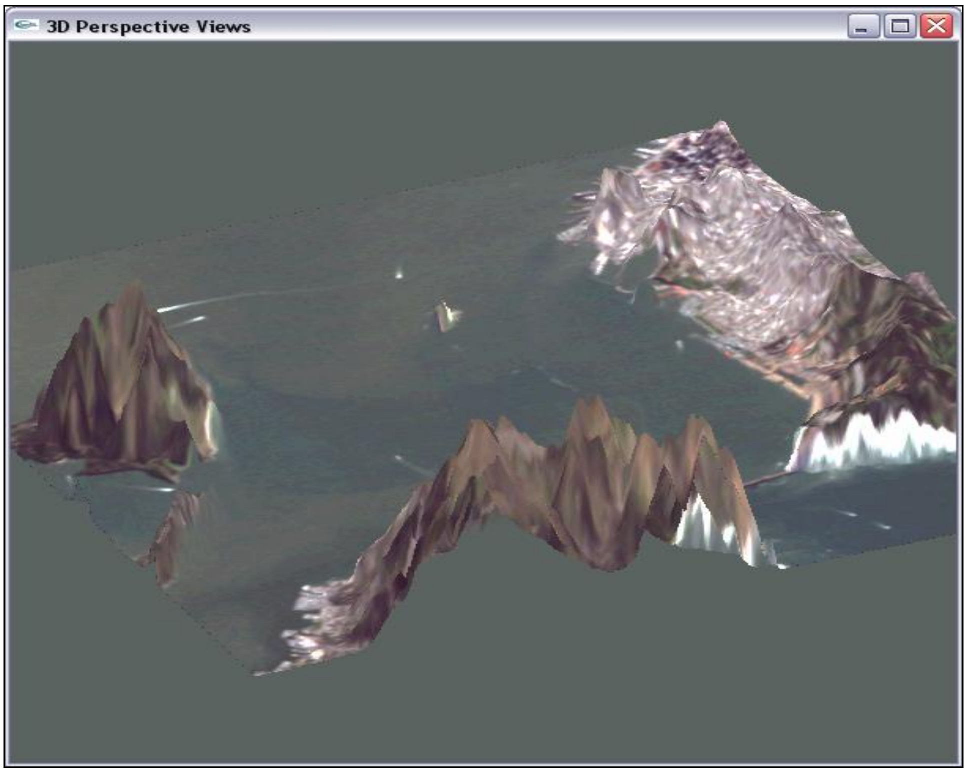


Figure 5.5 Rotation of the 3D Model

5.4 Translation of the 3D Model

User can control the translation of 3D model by mouse. These are steps to control the translation

1. User must press key M on a keyboard before use a mouse control the translation.

2. User drags the mouse to control the translation. They can drag a mouse in free direction to a new position.

The 3D model is moved to a new point because a viewport is changed to a new position. A viewport translate to the left side of window make a 3D model seem to be translate to the right side. A viewport translate to the right side of window make a 3D model seem to be translate to the left side.

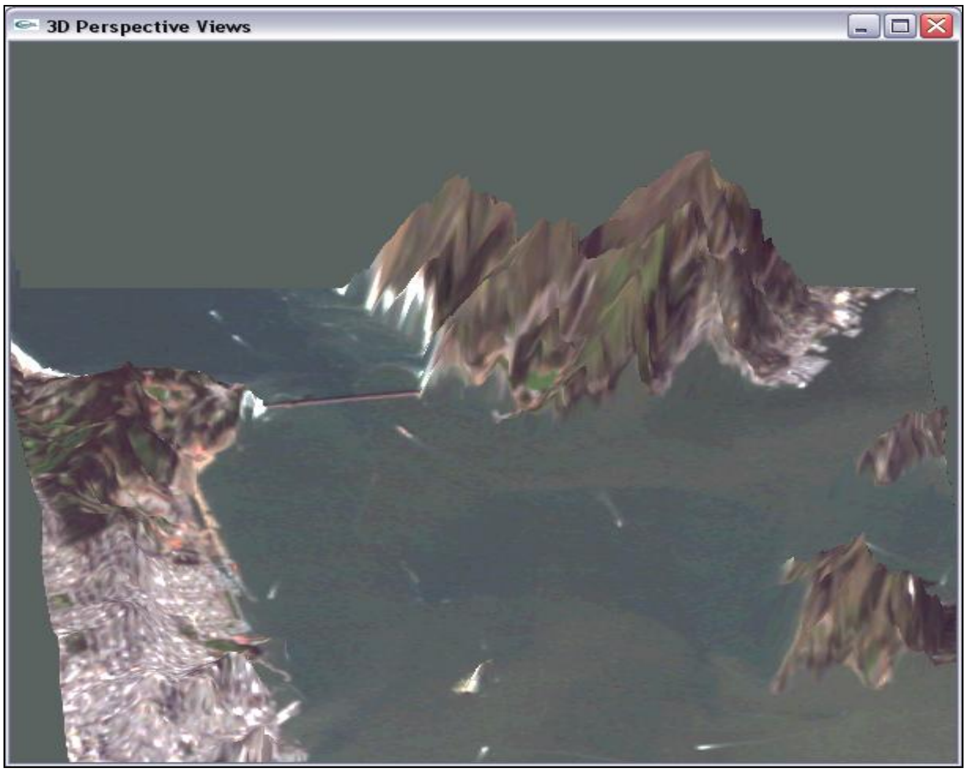


Figure 5.6 Translation of the 3D Model

5.5 Zooming In and Zooming Out

User can control zooming in and zooming out of 3D model by mouse. These are steps to control zooming in and zooming out.

1. User must press key S on a keyboard before using mouse control the zooming in and zooming out.
2. User drags a mouse to control zooming in and zooming out. They can drag a mouse to the left side on a display window to control zooming out the 3D model and drag a mouse to the right side on a display window to control zooming in the 3D model

The 3D model looks bigger or smaller because the viewport is changed a size. A bigger size of viewport make a 3D model look smaller and a smaller size of viewport make a 3D model look bigger.

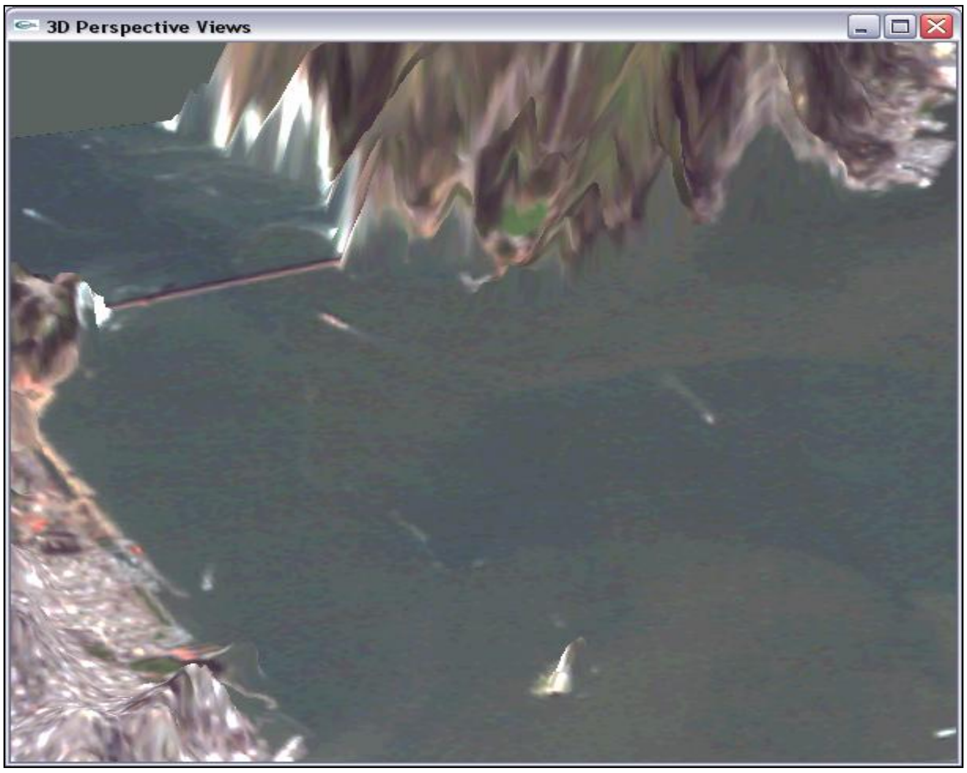


Figure 5.7 Zooming In

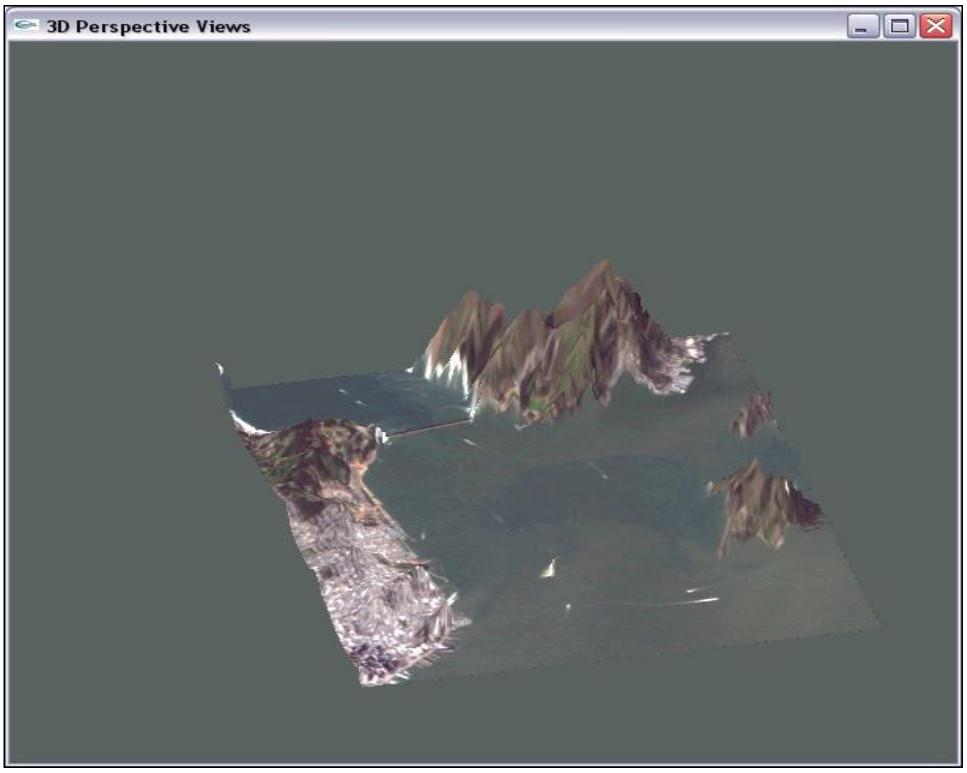


Figure 5.8 Zooming Out

5.6 Adjustment of Vertical Scale

User can adjust vertical scale of 3D model by a mouse. These are steps to adjust a vertical scale.

1. User must press key V on a keyboard before using the mouse adjust a vertical scale
2. User drags a mouse to adjust a vertical scale. They can drag a mouse to the upper side on a display window to adjust a vertical scale looks higher and can drag a mouse to the lower side on a display window to adjust a vertical scale looks shorter.

The vertical scale of 3D model looks higher or shorter because the viewport is changes the y-coordinates values range. A larger the y-coordinates values range makes a 3D model look shorter and a smaller y-coordinates values range makes a 3D model look higher.

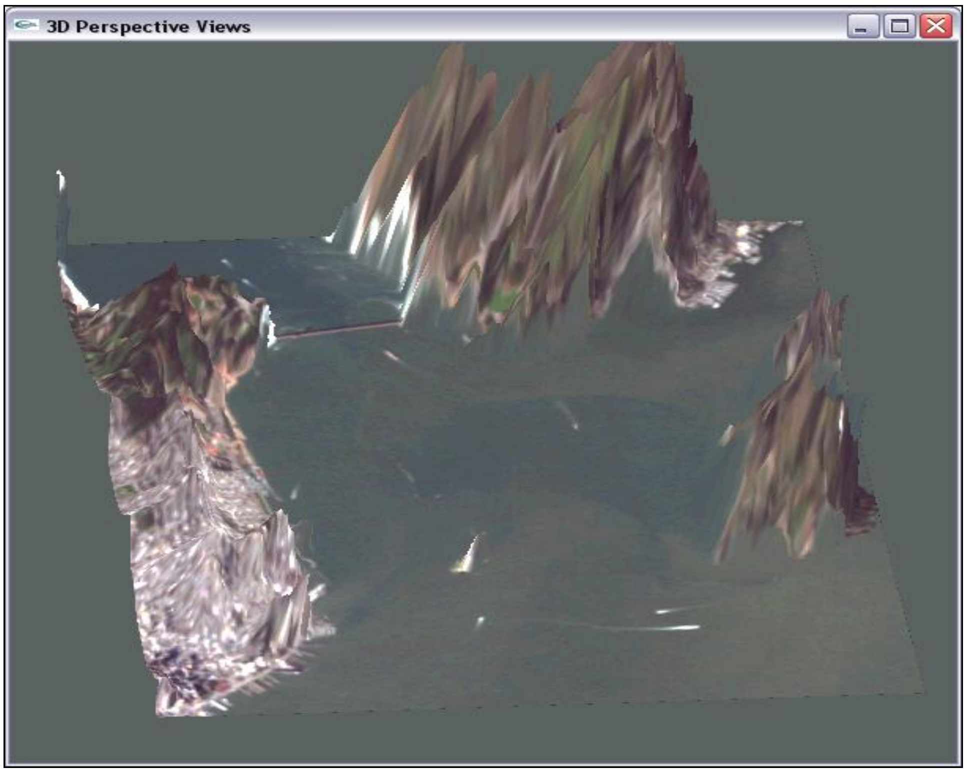


Figure 5.9 Adjustment of Vertical Scale

5.7 User Interface

Figure 5.2 shows the user interface for the program. The interface shows the capabilities of the program, which are described in this section.

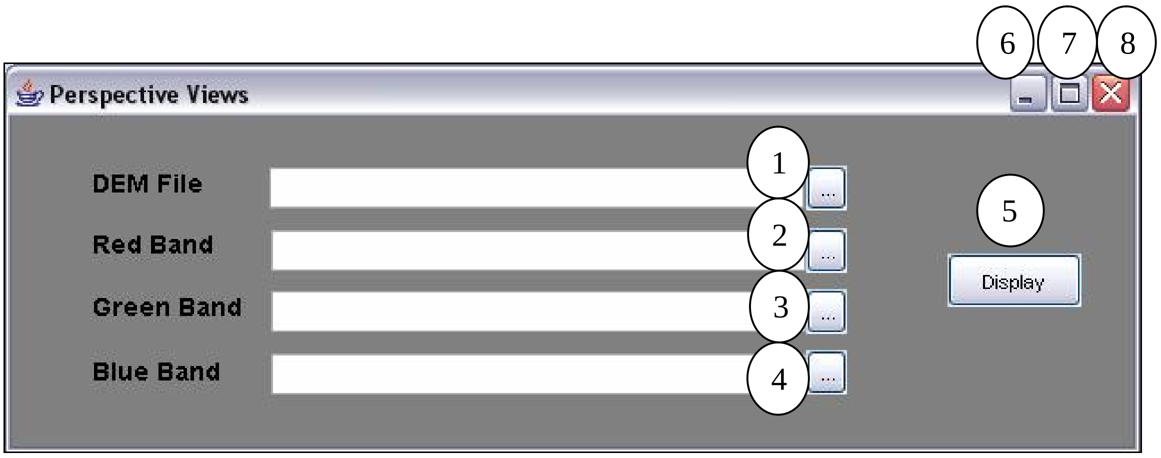


Figure 5.10 User Interface

1. DEM File: The button for insert the DEM file that used as input to display.
2. Satellite images (Red Band): The button for insert the Red Band of satellite images.
3. Satellite images (Green Band): The button for insert the Green Band of satellite images.
4. Satellite images (Blue Band): The button for insert the Blue Band of satellite images.
5. Display button: Click to compute the selected file in all text fields to display. If user selects only one band of satellite images from three bands, the output will display in gray scale mode, but if user selects three bands of satellite images, the output will display in true color mode.
6. Minimize window: Hide the window to the task bar.
7. Restore: Restore the window to the previous size.
8. Close: Close the window application.

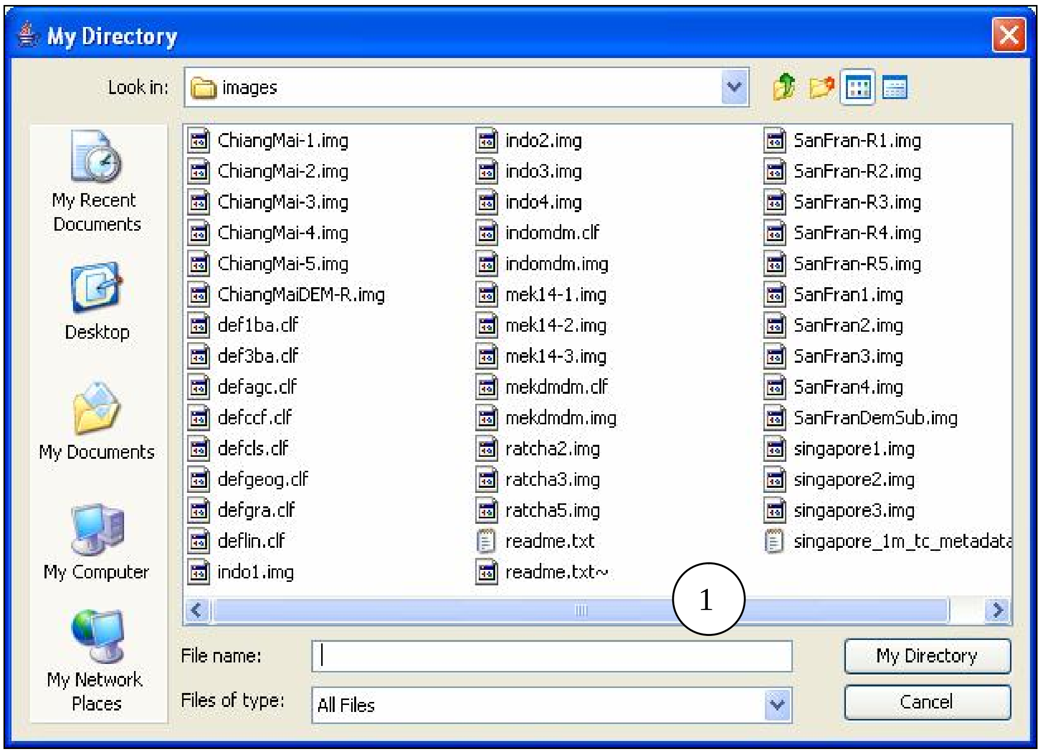
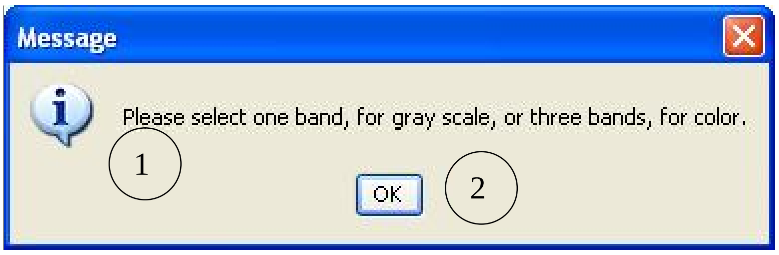


Figure 5.11 File Selection Box

1. File selection box: The dialog box for searching files that used as input to display.


Message

Please select one band, for gray scale, or three bands, for color.

Figure 5.12 Warning Message

1. Warning message: The dialog box will be shown when user chooses two bands of satellite image.
2. OK button: The button for return to the user interface for selecting the satellite image again.

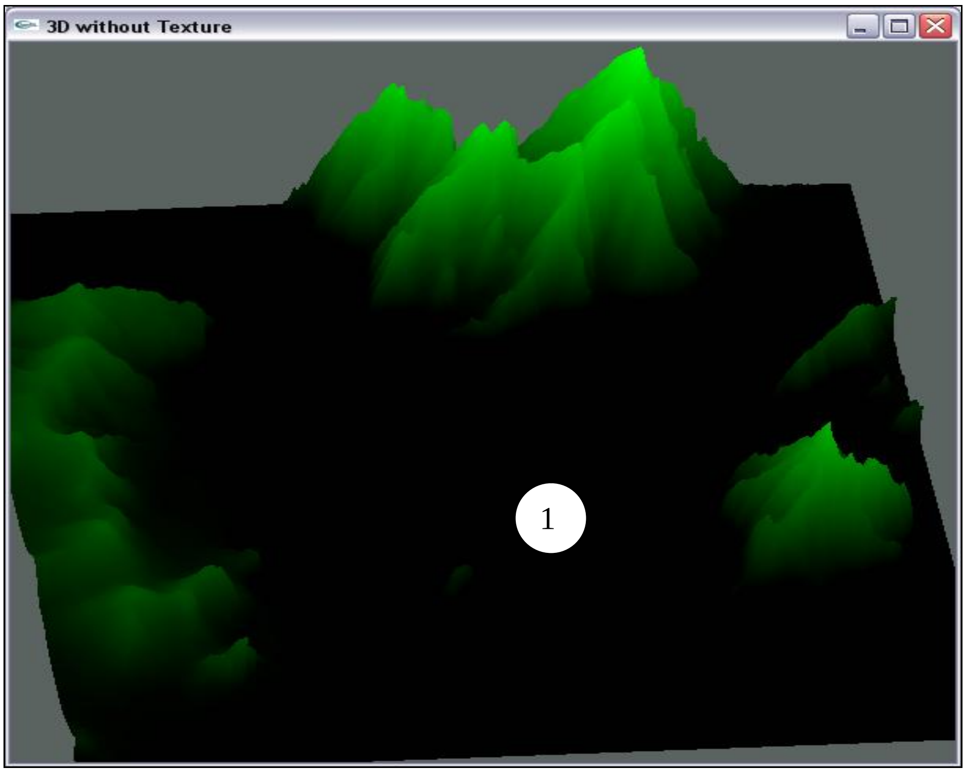


Figure 5.13 Display Window

1. Display window: Created from OpenGL function. It displays output of this program.

5.8 Connection between the User Interface and the Display

Connection between Java and C or C++ language of this program can connect by creating runtime object then create string variable to receive command line for running the C++ program follow input that user insert to the user interface afterward create process object to execute follow these command line.

Chapter 6

Conclusions and Discussions

6.1 Conclusions

This project is divided into three main parts:

1. The process to create a simple 3D model is the first process for this program. This process need the height value from DEMs files to create the simple 3D model. The result from this process will show the structure of the terrain that is of interest.
2. The process to map texture mapping is a process that need the result from the previous process for mapping details of landform on it. The details landform obtains from satellite images. Texture mapping with one band of satellite images displays the result in gray scale mode and three bands of satellite images displays in RGB mode. The combination of three bands gives a natural color result when the red band is used for the red image component, the green band for the green image component, and the blue band for the blue image component. If the arrangement of bands does not follow this pattern, the result will be a false color image. The result from this process shows a 3D model that has a realistic appearance because user can see the landform of this terrain 3D model.
3. The process to interact between the user and this program. This process is controlled via user interface. The user can choose input files via Java interface and can adjust display by zoom in, zoom out, adjust vertical scale and rotate model via the display window. The user can control these by using mouse.

6.2 Discussion

In implementation phase, we have changed some parts from design phase.

1. We have changed the facility to create user interface from WxWin to Java language. WxWin is a package that we have never learned or used.

We would need to spend the long time to learn it. Hence, we decided to use Java language instead of WxWin, because we have used Java language to create user interface. We thought it can help us to reduce the implementation time. Furthermore it has ability to create function of user interface coincident our requirement and also can connect to C++ language with creating runtime object then created string variable to receive command line for running the program follow input that user insert to the user interface afterward created process to execute follow these command line. The reason that we have not used JNI to link source code of Java language to C++ language because we try to use JNI but not success and our user interface has responsibility for getting input file only. So we can connect by using this method.

2. We have changed the user interface. Because we have changed the language from WxWin to Java, this affected the features of the user interface. We could not easily combine displaying window which was created by OpenGL function and C++ language to user interface which was created by Java language. We decided it would be better to separate the display window and the Java control panel. The Java control panel sends the control values from the user to the program and the display window only responds to display output.

6.3 Skills and Knowledge from the Senior Project

Doing this senior project provides many useful things for us that are development skill, new knowledge, concept and general skill. We can distribute by these.

1. We have developed our skill about programming in C, C++ and Java language. Especially, C++ language is programming language that we never used before. We have practiced programming with C++ in this project.
2. We have learned and used graphic library to create computer graphic program. In our project, we used OpenGL library to help us for creating model of perspective views.

3. We have learned many concepts and algorithms from our project that include polygon mesh, texture mapping, animation of 3D and the general concepts of computer graphics.
4. We have obtained knowledge and information about remote sensing image especially in DEM and satellite images which we used as the input to the program.
5. We have learned and practiced using MingW as compiler that is development environment that provides UNIX-like capabilities on Windows.
6. We have learned to use OpenDragon program that is a remote sensing image processing package and also practiced using OpenDragon toolkit for reading header of Open Dragon file.
7. We have learned about linking together code in different language. We have practiced creating runtime object in Java code to link source code of Java language to C++ language.
8. We have learned to work in a team, manage our time, and to have more responsibility.
9. We also learned to handle the problems which we have never faced before by searching information on internet, consulting with our advisor, and asking the people who know.
10. We have improved and developed our English skill in writing, speaking, reading and listening.

6.4 Problems

1. We need to collect data from two dimensions for using to create the three dimension but normal two dimension can not provides enough information for create three dimensions. Then, we use DEM data to create three dimensions because DEM data give information about the elevation of the earth’s surface and widely available for many locations.
2. Our program has several source codes modules. We used to compile our source code by use g++ command. This command will compile just only one source code per time. It makes problems that we must run this command many times to compile all source codes and we can not link all

source code together. We solved this problem by using makefile. Makefile help us to compile all source code automatically at one time when we use command make and link object that we need together.

3. We use Java language to create user interface and use C and C++ to create perspective views. It is a problem to connect module of user interface with module of perspective view. So we created runtime object then created string variable to receive command line to run program afterward created process to execute follow these command line to connect Java language to C++ language.
4. We can not embed a display window inside Java interface because a display window was created by OpenGL in part of C++ program. A display window by OpenGL has own window. It can not integrate with Java interface.

6.5 Suggestion

This program can be used in the real world coordinates and can be of benefit for people who are interested in geography. We would like this project continue to be developed for being the better program and has more function to respond the requirement of user.

For the person who is interested to continue on this project, we think this project should be add more function such as supporting other kind of DEM and satellite images file than .img file or displaying the value of the real world coordinates and the height value on each coordinate. This will be useful for people who would like to see these values for analyzing geographic data.

References

1. Donald Hearn and M. Pauline Baker, Computer Graphics C Version Second Edition, Prentice Hall, New Jersey, England, 1994.
2. Donald Hearn and M. Pauline Baker, Computer Graphics with OpenGL Third Edition, Prentice Hall, New Jersey, England, 2004.
3. Edward Angel, Interactive Computer Graphics A Top-Down Approach Using OpenGL Fourth Edition, Addison Wesley, USA, 2006.
4. Kurt T. Rudaul, M. Sc., OpenDragon's User Manual, Chapter 4 Classification Operation., King Mongkut's University of Technology Thonburi, 2006.
5. IRS Gallery of Sample Images [online], available: http://www.photosat.ca [2006, September 7].
6. Satellite Imagery [online], available: http://www.stormsurf.com/ page2/tutorials/satimagery.shtml [2006, September 12].
7. Chapter 1. Introduction to Computer Animation [online], available: http://www.siggraph.org/education/materials/HyperGraph/animation/rick _parent/Intr.html [2006, September 12].
8. Welcome to Wikipedia [online], available: http://en.wikipedia.org/ wiki/Main_Page [2006, September 12].What is Bank [online], available: http://www.bank.org/ [2002, November 10].
9. News [online], available: http://www.mesa3d.org/Mesa Home Page_files /news.htm [2006, September 8].
10. Polygon Surfaces [online], available: http://escience.anu.edu.au/ lecture/ cg/surfaceModeling/polygon.en.html [2006, September 14].
11. Working with Triangle Meshes [online], available: http://www.artofillusion.org/docs/trimeshtut/index [2006, September 4].
12. Computer Graphics [online], available: http://www.cs.su.ac.th/~rawitat/ teaching/graphics06/coursefiles/files/3_viewing.pdf [2006, September 19].

13. Texture Mapping [online], available: http://earthscape.org/t1/fre03/ fre03.11.htm [2006, September 12].

14. 2D and 3D Projections [online], available: http://www.cs.umu.se/ kurser/ TDBC07/HT05/handouts/HO-lecture4.pdf [2006, September 12].

15. Chapter 1 Fundamentals of Remote Sensing [online], available: http://www.profc.udec.cl/~gabriel/tutoriales/rsnote/cp1/cp1-1.htm [2006, September 11].

16. Satellite Image [online], available: http://landsat.gsfc.nasa.gov/ references/glossary.html [2006, September 12].

17. Articles and Resources [online], available: http://www.gamedev.net/ reference/ [2006, September 15].

18. Perspective Texture Mapping, Part 3: Endpoints and Mapping [online], available: http://www.d6.com/users/checker/misctech.htm [2006, September 18].

19. The Map Frame [online], available: http://www.mathworks.com/access/ helpdesk/help/toolbox/map/map.html?/access/helpdesk/help/toolbox/map /creati10.html [2006, September 18].

20. Introduction: Technical and History Perspectives of Remote Sensing [online], available: http://rst.gsfc.nasa.gov/Intro/Part2_1.html [2006, September 18].