REAL TIME 3D MODELLING IN SURVEILLANCE SYSTEMS

PROJECT REPORT

Submitted In Partial Fulfillment Of The Requirements For The Award Of The Degree Of Bachelor Of Engineering In Computer Science And Engineering Of Bharathiar University, Coimbatore.

Submitted by

Mr. Kishore Kumar J
Mr. Rama krishnan V
Ms. KN Seetha
Ms. Smitha Surendran

P. 676

Under the guidance of

Mr. VS. Shenoi
Scientist 'D'
NPOL
Cochin.

Mrs. S. Devaki, M.S
Assistant Professor
Kumraguru College Of Technology
Coimbatore.

Department of Computer Science & Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore - 641006

March 2002

CERTIFICATE

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore - 641006.

This is to certify that the project work entitled
"REAL TIME 3D MODELLING IN SURVEILLANCE SYSTEMS"
has been submitted by

Mr. Kishore Kumar J	**9827K0185**
Mr. Rama Krishnan V	**9827K0205**
Ms. KN Seetha	**9827K0213**
Ms. Smitha Surendran	**9827K0216**

In partial fulfillment for the award of the degree of
Bachelor of Engineering in Computer Science and Engineering
of Bharathiar University, Coimbatore
during the academic year 2001 - 2002

3/3/2002
Guide

12/3/02
Head of the Department

Certified that we examined the candidate in the Project Work Viva Voce Examination held on. ..... and the University Register Number was .....

Internal Examiner

External Examiner

ACKNOWLEDGEMENTS

This project report is an outcome of an enjoyed project, coordinated with the co-operation of various persons, views and ideas shared, warmth and affection showed by family and friends.

The credit goes to the faculty members and students and staff of the college especially our Principal **Dr. K.K. Padmanaban, B.Sc (Engg), MTech, PhD** who directed us with sincerity and dedication.

We place on record our sincere thanks to our **Head of the Department**, **Prof. Dr. S. Thangaswamy, PhD,** for his progressive outlook and valuable suggestions.

We express our sincere thanks to our internal project guide **Assistant Professor Mrs. S. Devaki, M.S,** for her thought provacating suggestions and encouragement.

We would like to express our sincere thanks to **Shri V Chander, Director NPOL** and **Dr. Unnikrishnan HRD, NPOL, Cochin** for giving us opportunity to carry out this project work.

We are greatly indebted to our guide and mentor **Mr. VS. Shenoi Scientist 'D', NPOL** for the relentless support, valuable suggestions and advice he gave throughout the project.

We are grateful to **Mr. Shoban, Project Manager, TBN, Coimbatore**, for his guidance.

We also extend our sincere thanks to **Mr. M.C. Surendran, MTech, Scientist 'E' CSIR, Chennai** for having helped us enter this project & work through it successfully.

We cherish the love and the moral support of our parents, which has helped in the successful completion of this project.

We thank our friends who deserve special mention for their magnanimity care and concern.

Above all we owe our gratitude to god almighty for showing abundant blessings on us.

SYNOPSIS

In today's world of nuclear warfare, the very existence depends upon the technological advancement of defense systems. Most of the warfare takes advantage of the vast seas and oceans, which implies a need for the development of highly equipped sea vessels. These vessels that can go underwater have employed surveillance systems with the ability to scan the terrains, for steering the vehicle by monitoring obstacles and targets on, above and below the surface.

The SONAR sensing technology is mainly used here in detecting targets that are not seen by bare eyes. The surveillance system picks up acoustic signals around the vessel covering 360 degrees. These signals undergo different processing and the processed data are finally presented pictorially.

This project "**REAL TIME 3D-MODELLING IN SURVELLANCE SYSTEMS**" is undertaken to simulate and study the various scenarios dealt by the surveillance system. The terrain simulated by software can be inferred to estimate the positions of the objects around the source.

The GUI for the software consists of a menu based interface, which can be easily used by the user. The required data is generated at the server end and passed to the client through IPC .The data received is used by the client to plot the 3D terrain. The position of the objects in the terrain can be inferred from the display.

CONTENTS

1. INTRODUCTION

The project **REAL TIME 3D MODELING IN SURVEILLANCE SYSTEM** is used to simulate the terrain around the surveillance system. In surveillance systems SONAR sensing technology is used to scan the ocean bed. Acoustic signals are sent 360 degree around the own vessel and the reflected signals are recepted using highly complicated equipments. The acoustic signals received, are converted to electrical signals, and after a series of transformations are finally presented pictorially. The terrain scenario thus generated will give lot of information regarding the objects around the source.

The equipments used in the surveillance system to generate the terrain scenario are highly complicated and expensive. Here in this project, we are simulating the 3D view of the terrain around the source vessel. A real time 3D presentation of the data would give more details with high degree of precision from that of the conventional systems.

1.1 EXISTING SYSTEM AND ITS LIMITATIONS

The conventional system generates pictorial output in the form of a discrete graph, history graph or waterfall charts. These models are just two-dimensional representations of the terrain. The inferences made from the 2D graph are limited and discrete intensity variations are not clearly visible. Apart from this it does not give the effect of the live scenario.

1.2 PROPOSED SYSTEM AND ITS ADVANTAGES

To overcome the problem associated with the existing system we can go for a real time three-dimensional presentation of the data that would give more details with high degree of precision from that of the conventional systems. This simulation will give a real effect to the terrain scenario. It is proposed to be user interactive and hence simple to handle. The simulation is done by making use of algorithms, which have been made as simple as possible.

1.3 OPERATING ENVIRONMENT

The software is developed on a Linux platform, for it's powerful features such as networking, portability and security. The GUI is developed using X Window System Programming (X11R6). The client/server application is implemented using UNIX network programming. The graphics algorithms are solved in C programming; the entire application is compiled using the GCC (GNU C) Compiler.

2. SYSTEM REQUIREMENTS

2.1 PRODUCT DEFINTION

2.1.1 Problem Statement:

The Product is concerned with the simulation of the terrain around the surveillance system. It should be able to simulate the scenario of multiple objects with the help of floating horizon algorithm.

2.1.2 Processing Environment

Hardware Specifications

Processor:	Pentium III 500 MHz
Floppy Disk Drive:	1.44 MB
Hard Disk:	10 G.B
Keyboard:	Intel 105 Keyboard
System RAM:	64 MB
Display adapter:	VGA card with on-board 4 MB VRAM supporting a resolution of 800 * 600 with 16-bit color depth running.
Monitor:	Color monitor with refresh rate of 85 Hz.

Software Specifications

Language Used:	X Window System Programming (X11R6), Unix Network Programming.
Tools Used:	Open Motif Toolkit
Operating System:	Red Hat Linux 7.1
Compiler:	gcc compiler version 2.96.
Debugger:	gdb

2.1.3 User Characteristics

The product is basically a simulation program to be used by scientists and other staff members monitoring the surveillance systems for further study and research. A user interactive interface is designed in which the user can visualize the terrain and the current data is displayed simultaneously. User's manual is prepared and available for the users.

2.1.4 Solution Strategy

The problem was dealt in a sequential manner. The first step was to study the relevant operations of the surveillance system. An analysis of the problem definitions was made based on the current system and the requirements of the proposed systems were compared. As a result of the requirements analysis, the project was split into three main modules.

2.1.5 Acceptance Criteria

The product has to accept some input values and do some processing. Finally the product should display the terrain in the window from which the relative position of objects can be studied.

2.2 PROJECT PLAN

2.2.1 Life Cycle Model

The Spiral Model is proposed to be the life cycle model for developing the software product. It provides the potential for the rapid development of incremental version of the software. The software is developed in the series of incremental releases. The spiral model has six task regions.

Task Region 1:

- Terminology: Customer Communication.
- Milestones: November 24th
- Work Product: The scientist defined the problem statement and suggested the books related to the project.

Task Region 2:

- Terminology: Planning
- Milestones: November 28th

Work Product: Analysis of the product definition. The function the product has to perform, the platform, programming languages and development tools to be used are decided in this stage.

Task Region 3:

- Terminology: Risk Analysis
- Milestones: December 11th
- Work Product:

Technical Risk: Preparation of design document is a crucial part of the project, which is time consuming. The entire coding depends on how the display is going to be given. The logics, GUI structure to be used are clearly designed.

Managerial Risk: The project modules have to be efficiently planned within the time limit specified. The modules under consideration are independent. Hence each module has to be completed in time.

Task Region 4

- Terminology: Engineering
- Milestones: SRS Document December 24th
- Work Product: Based on the needs of the scientists, the software requirements specification is prepared. SRS includes product Overview, Processing Environment, External interface and data flow, Functional specifications, performance requirements, Exception Condition and handling, early subsets, Foreseeable modifications, Acceptance critical design guidelines.

- Milestones: Design Document January 11th
- Work Product: The design document is prepared based on the Needs of the scientist. The Design document is a prominent part of the project and facilitates easy coding. It includes External design specification, Architectural design overview and detailed design specifications.

Task Region 5:

- Terminology: Construction and Release
- Milestones: February 15th - 20th
- Work Product: The product is designed according to the product definition with user friendly GUI.

Task Region 6:

- Terminology: Customer Evaluation.
- Milestones: March 2nd
- Work Product: The customer evaluated the present product by seeing the execution of the project and gave the feedback. The feedback was that the product works well and can be extended in future with more advanced features if time permits.

2.2.2 Team Structures

The software is used to simulate the terrain around the surveillance system. The scientists go for simulation since the line equipments are very expensive and complex. Hence the simulation is done first and later replaced by real equipments.

3. SOFTWARE REQUIREMENTS SPECIFICATION

3. 1 PRODUCT OVERVIEW AND SUMMARY

The product is mainly concerned with the plotting of the terrain, giving 3D view of the environment around the sea vessel. The terrain gets plotted every few seconds (i.e.) a predetermined interval is fixed.

Summary: The objective of the product designed is to generate the three dimensional view of the terrain, around the sea vessel. The inputs are received at real time from the server process console. After receiving the target details (as the input from the input module), the data are sent to the processing stage, in the server the computed values are sent to the client, through the established sockets. From there, sent to the display module. The display module plots the data.

3.2 DEVELOPMENT ENVIRONMENTS

Hardware specification

Processor:	Pentium III 500 MHz
Floppy Disk Drive:	1.44 MB
Hard Disk:	10 G.B
Keyboard:	Intel 105 Keyboard
System RAM:	64 MB
Display adapter:	VGA card with on-board 4 MB VRAM supporting a resolution of 800 * 600 with 16-bit color depth running.
Monitor:	Color monitor with refresh rate of 85 Hz.

Software Specifications

Language Used:	X Window System Programming (X11R6), Unix Network Programming.
Tools Used:	Open Motif Toolkit
Operating System:	Red Hat Linux 7.1
Compiler:	gcc compiler version 2.96.
Debugger:	gdb

3.3 EXTERNAL INTERFACE AND DATA FLOWS

3.3.1 User displays and report formats

This software is a user interactive product, where the user can select any desired option.

It contains a menu, form and drawing area.

Top-level Shell

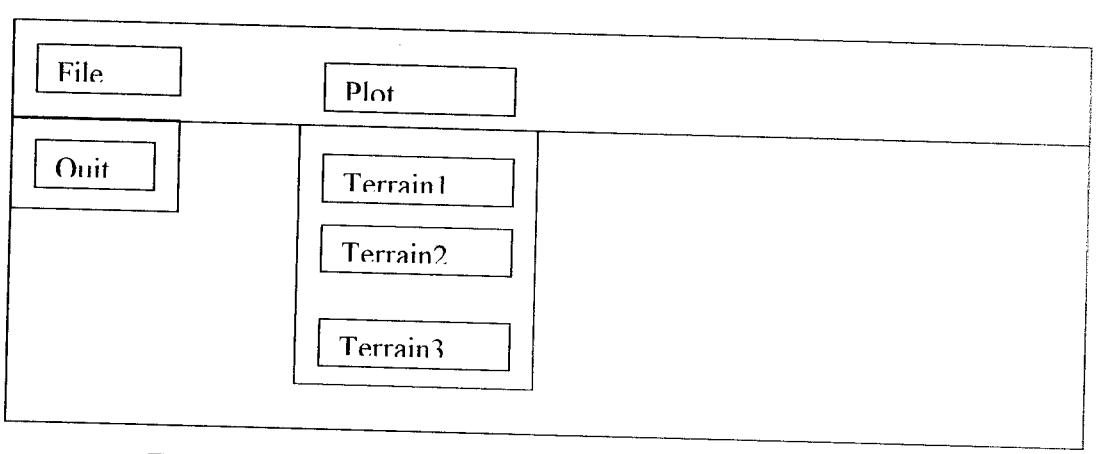
Main Widget

Menu

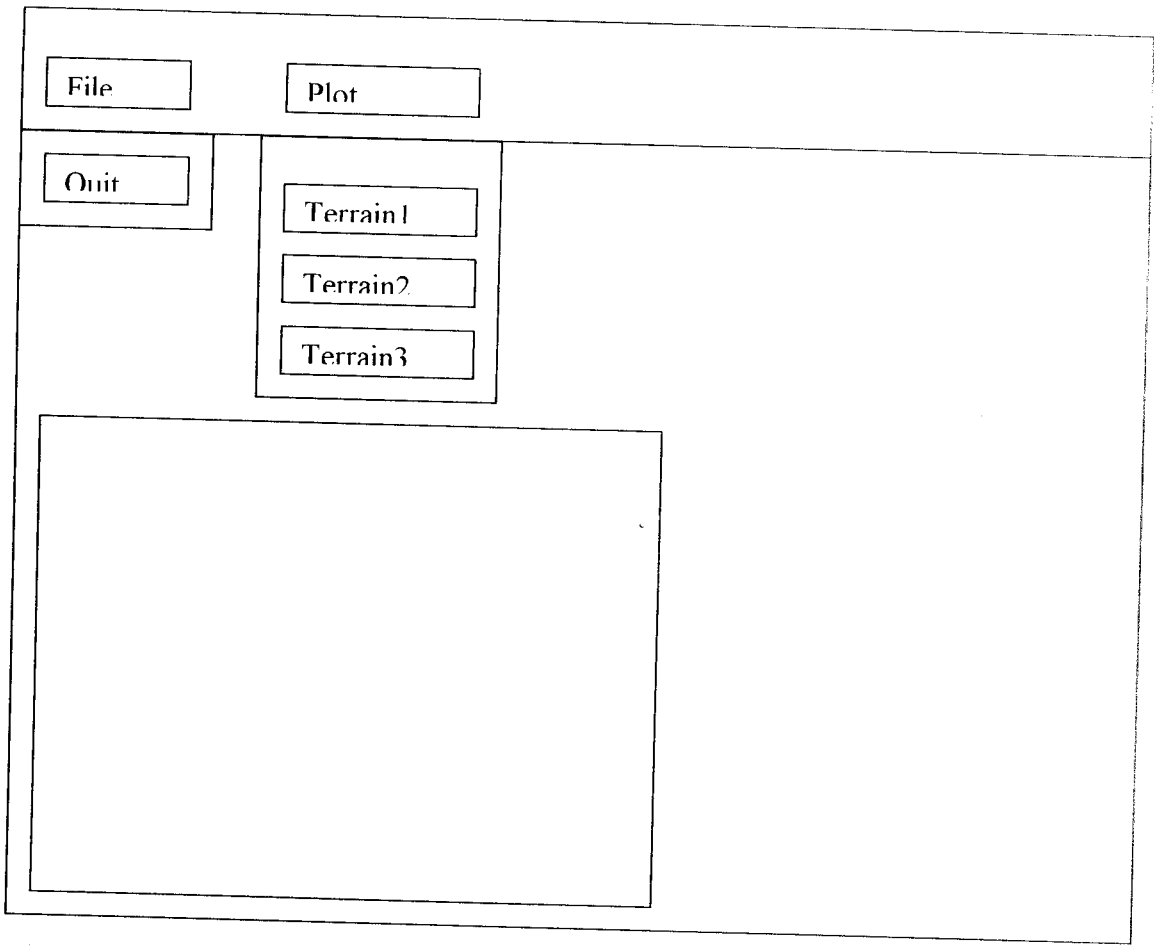
Form Widget

Drawing Area

The main menu consists of file and plot sub-menu items



The file menu item consists of quit option and the plot menu item consists of options to select different terrains to be drawn.



3.3.2 User Command Display

In order to be a user friendly software there are user commands available. With the help of the user commands the user can interact with this software. Interaction with the software is with the help of mouse.

1. File menu

With the help of left mouse button the user clicks the menu which inturn will drop down the pop up menu having the option quit.

2. Plot menu

With the help of the left mouse button the user clicks the plot menu which inturn will drop down the pop up menu having the option terrain1, terrain2, and terrain3.

2.1 Terrain 1

Clicking this generates a particular terrain scenario with the help of floating horizon algorithm.

2.2 Terrain 2

Clicking this generates a particular terrain scenario with the help of floating horizon algorithm.

2.3 Terrain 3

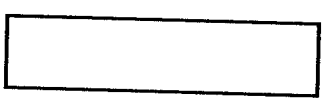
Clicking this generates a particular terrain scenario with the help of z-buffer algorithm.

3.3.3 High-level Data Flow Diagrams.

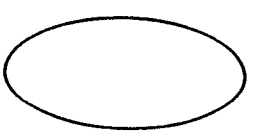
Data flow diagram is a graphical technique that depicts the information flow and the transforms that are applied as data move from the input to the output. DFD may be used to represent a system or software at any level of abstraction infact DFD's may be portioned into levels that represent increasing information flow and functional details.

Graphical notations used in DFD

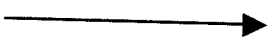
Inputs/Outputs to the Software



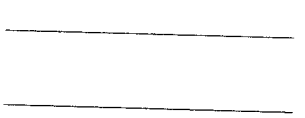
Processing the inputs



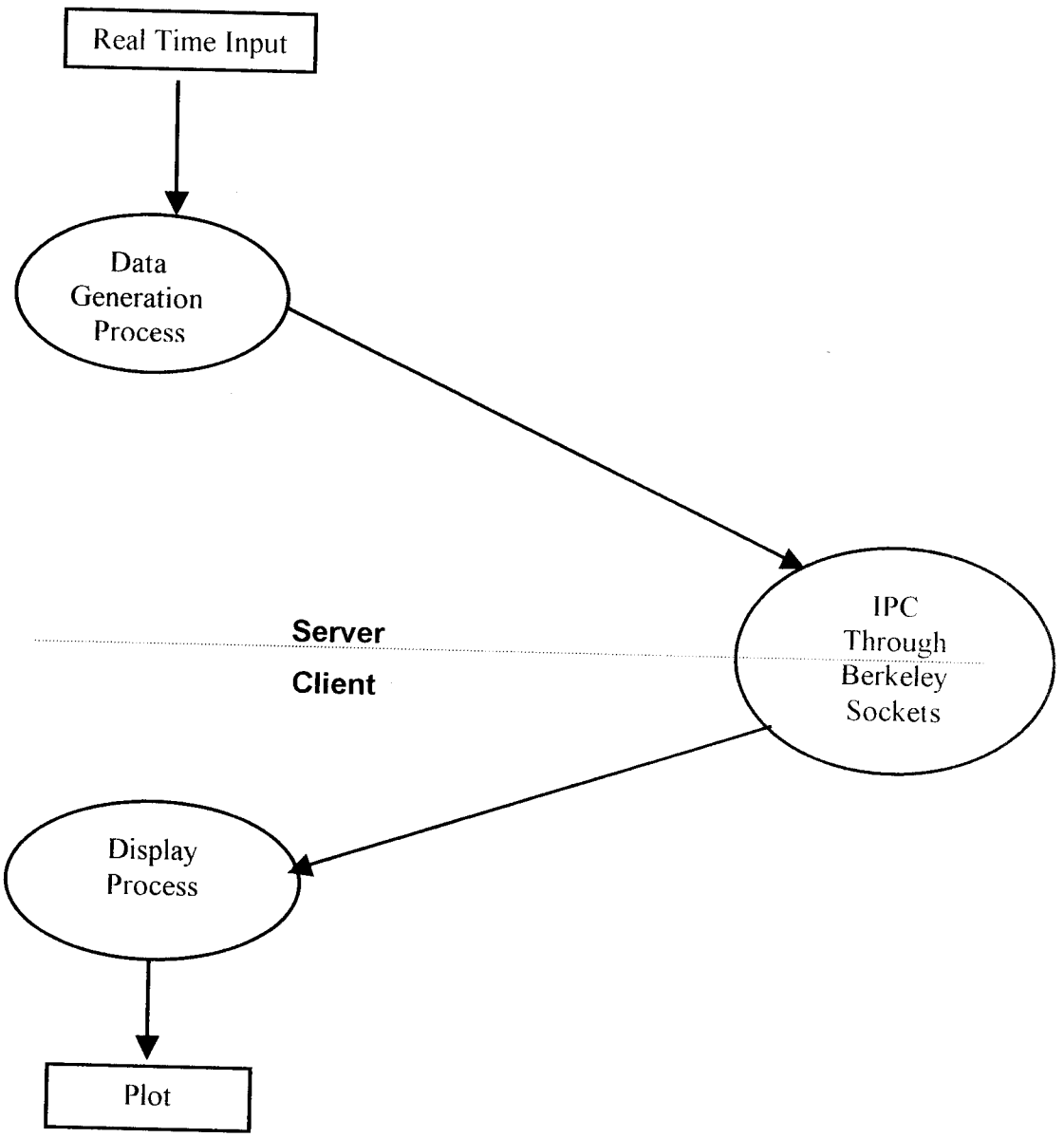
Data Flows



Data Stores



DATA FLOW DIAGRAM



3.4. FUNCTIONAL SPECIFICATIONS

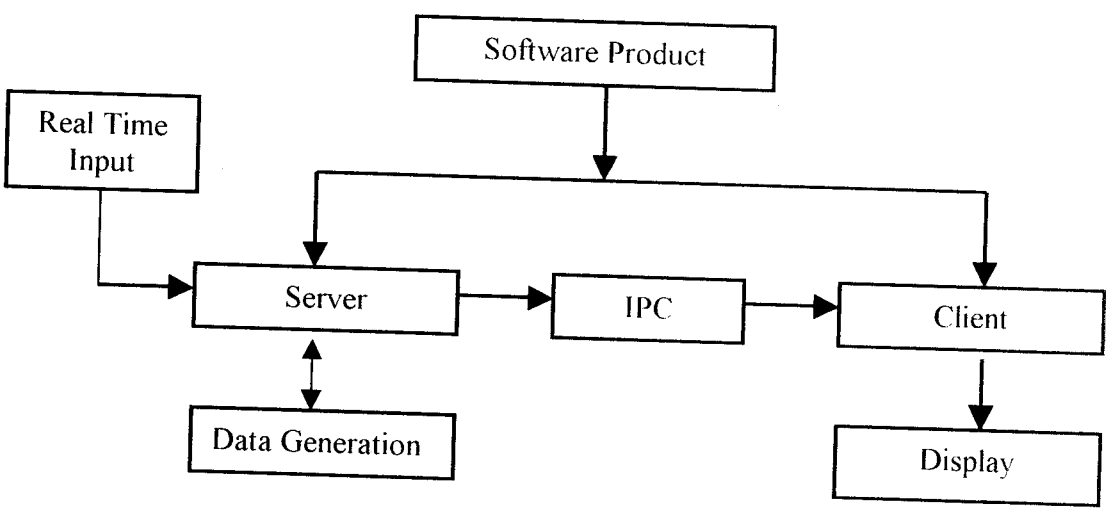
This software product comprises of two programs each consisting of two modules.

Server program

1. Data Generation
2. Communication process (IPC)

Client program

1. Communication process (IPC)
2. Display process



SERVER PROGRAM

1. Data Generation module

Real time input for the specific terrain is received and computed in accordance with the algorithm used and the data are generated accordingly.

Processing Narrative

This module generates the data based on two algorithms floating horizon algorithm and z-buffer algorithm, which are used for hidden line elimination of the terrain generated.

Floating horizon algorithm

The algorithm states that if at any given value of x, the y value of the curve in the current plane is larger than the maximum y value or smaller than the minimum y value for any previous curve at that x value then the curve is visible. Otherwise it is hidden.

Z-buffer algorithm

Z-buffer algorithm uses a depth buffer, which is used to store the z co-ordinate or depth of every visible pixel in image space. The depth or z value of a new pixel to be written to the frame is compared to the depth of that pixel stored in the z-buffer. If the comparison indicates that the new pixel in front of the pixel stored in the frame buffer then the new pixel is written to the frame buffer and the z-buffer updated with new z value. If not, no action is taken. Conceptually, the algorithm is a search over x, y for the largest value of $z = f(x, y)$.

2.Server Communication Module

This process waits for the client to request. On client request it communicates with data generation process for data and sends the data to the client. This is implemented using TCP Berkeley sockets protocol.

Processing Narrative:

The interprocess communication is established using Berkeley sockets. Berkley's Socket programming is also used to continuously poll the client request.

CLIENT PROGRAM

1.Client Communication Module

This program gets the data from the server. The communication is established using interprocess communication protocol.

Processing Narrative:

Data is generated in real time and sent through the socket to the client. The client side display program to generate the terrain uses this data.

2.Display Process Module

The display process reads the data sent through the socket and plots it. This module consists of developing a user interface in which the data is displayed.

Processing Narrative:

The data read is used to plot the terrain according to the menu selected. The terrain is generated at specific intervals of time. This is displayed in a separate drawing area.

3.5 PERFORMANCE REQUIREMENTS

The software displays the terrain periodically according to the inputs received. Since the product is used in Surveillance system, the time of response plays a very crucial role. The algorithms have been made very simple by reducing the complexity, which in turn increases the speed of execution.

3.6.ACCEPTANCE CRITERIA

3.6.1 Functional and Performance Tests

The scientists did Functional and Performance Tests and when the modules were finished, expected outcomes were tested and accepted.

3.6.2 Documentation standards

System Requirement document and the software requirement specification documents were done and accepted by the scientists.

4. DESIGN DOCUMENT

4.1 EXTERNAL DESIGN SPECIFICATION

4.1.1 User Displays and Report Format

The software has a user interactive interface, in which the user can easily simulate the output using click events.

It contains a menu, drawing area and a label.

Top-level Shell

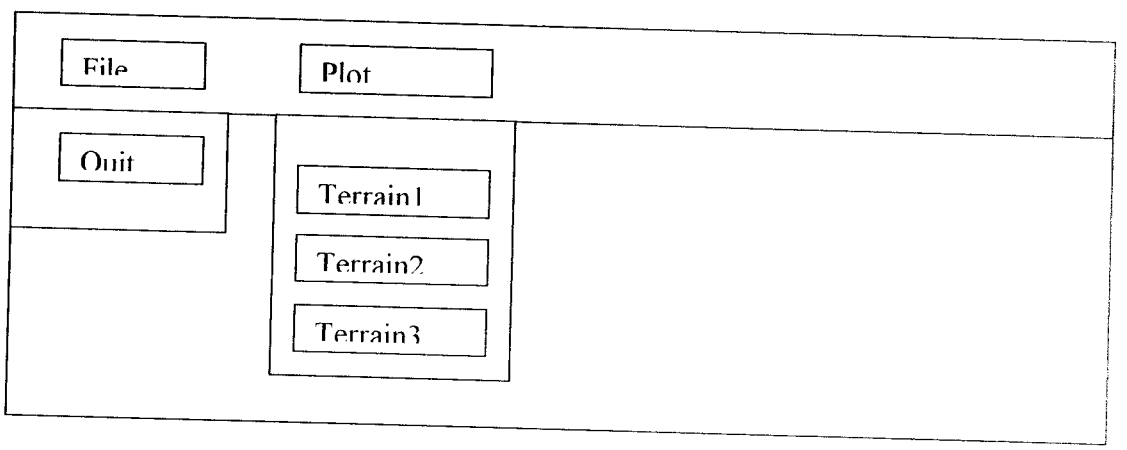
Main Widget

Menu

Form Widget

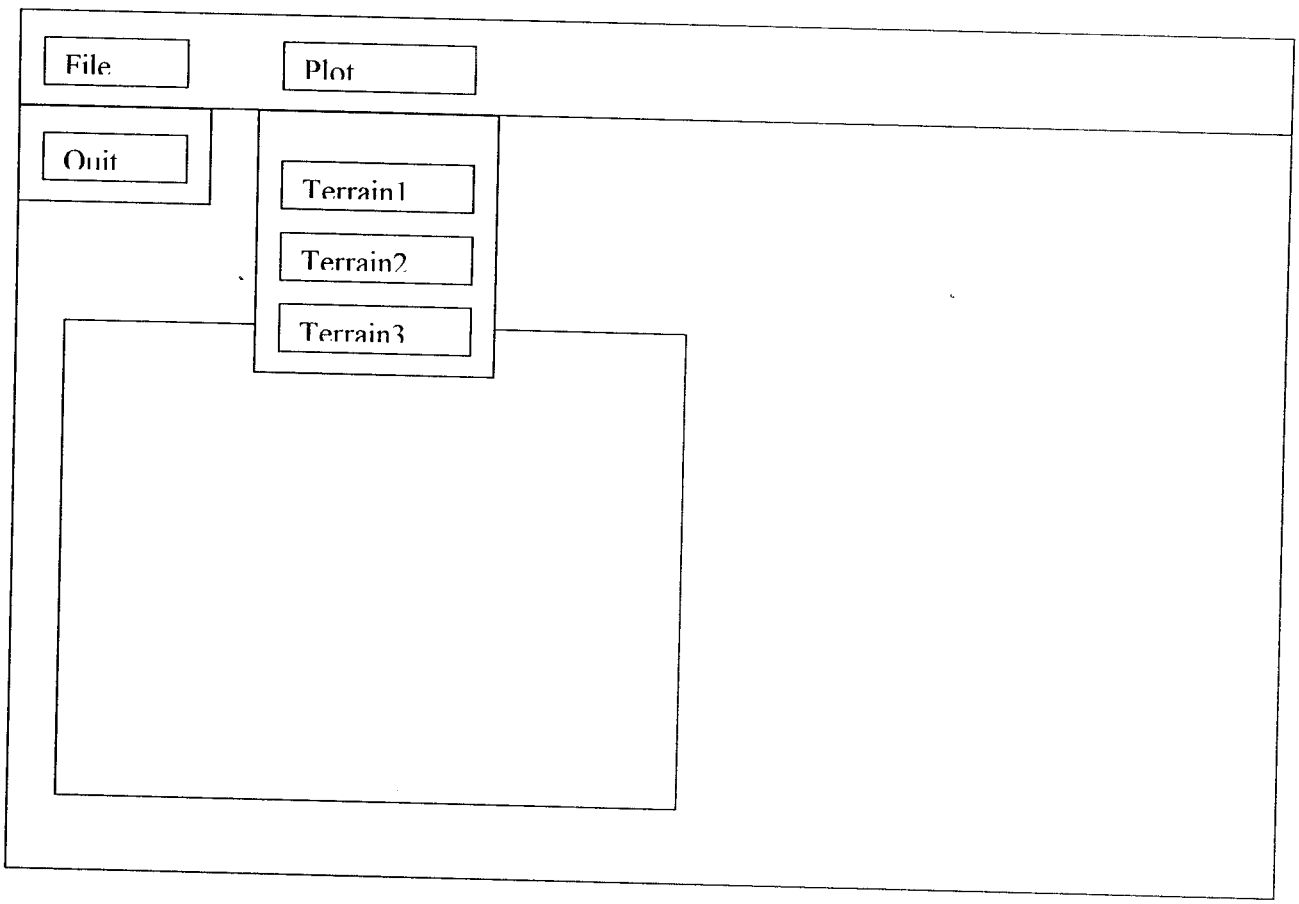
Drawing Area

The main menu consists of file and plot sub-menu items



The file menu item consists of quit option and the plot menu item consists of options to select different terrains to be drawn.

User Interface



In order to be a user friendly software there are user commands available. With the help of the user commands the user can interact with this software. Interaction with the software is with the help of mouse.

1. File menu

With the help of left mouse button the user clicks the menu which inturn will drop down the pop up menu having the option quit.

2. Plot menu

With the help of the left mouse button the user clicks the plot menu which inturn will drop down the pop up menu having the option terrain1, terrain2.

2.1 Terrain 1

Clicking this generates a particular terrain scenario with the help of floating horizon algorithm.

2.2 Terrain 2

Clicking this generates a particular terrain scenario with the help of floating horizon algorithm.

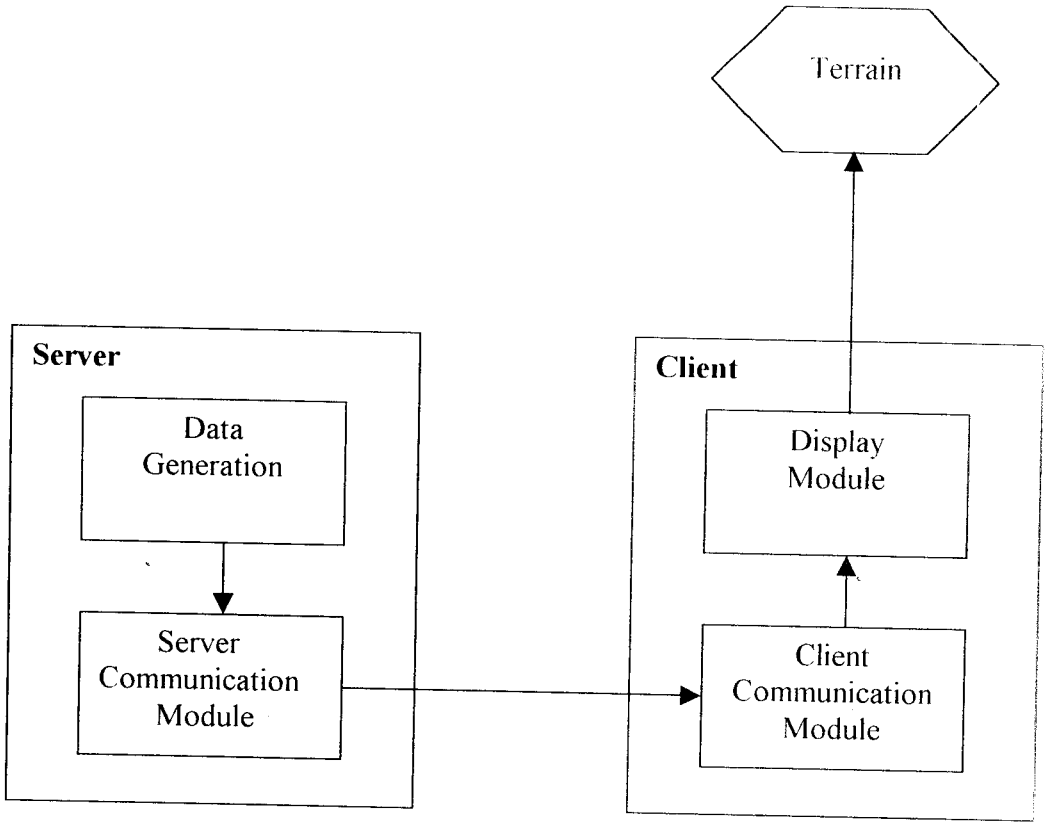
2.3 Terrain 3

Clicking this generates a particular terrain scenario with the help of z-buffer algorithm.

4.2 Design Specifications

4.2.1 Structure Diagrams

The product structure is explained in detail in Structure diagram. The Modules in the program and the input and outputs are mentioned here.



4.2.2 Functional Description

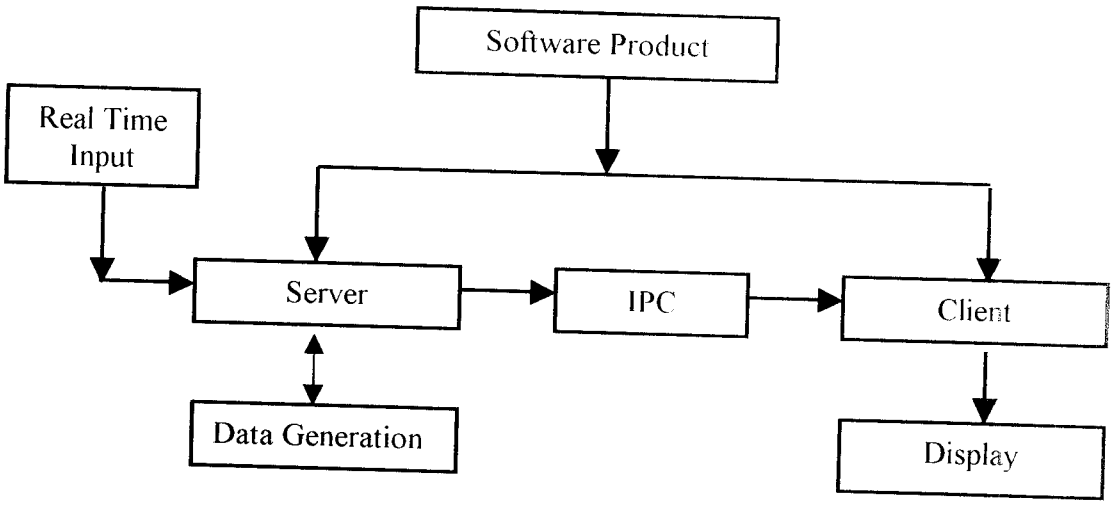
This software product comprises of two programs each consisting of two modules.

Server program

3. Data Generation
4. Communication process (IPC)

Client program

2. Communication process (IPC)
3. Display process



SERVER PROGRAM

1. Data Generation module

Real time input for the specific terrain is received and computed in accordance with the algorithm used and the data are generated accordingly.

Processing Narrative

This module generates the data based on two algorithms floating horizon algorithm and z-buffer algorithm, which are used for hidden line elimination of the terrain generated.

Floating horizon algorithm

The algorithm states that if at any given value of x, the y value of the curve in the current plane is larger than the maximum y value or smaller than the minimum y value for any previous curve at that x value then the curve is visible. Otherwise it is hidden.

Explanation:

Initialize upper and lower horizon arrays as follows:
Upper array for each x point = 0.0
Lower array for each x point = Vertical Screen

The fundamental idea behind the technique is to convert the three-dimensional problem to two-dimensions by intersecting the surface with a series of parallel cutting planes at constant values of z.

Then for each z = constant plane,
And for each x point on the curve in a z = constant plane,
Calculate the corresponding y value according to the function given.

If at any given value of x,
the y value of the curve in the current plane is greater than or equal to the lower horizon of that particular x or less than the upper horizon of that x,
then do nothing;

else plot that particular point and update the lower and upper horizons relatively.

PSEUDOCODE

Upper is the array containing the upper horizon values
Lower is the array containing the lower horizon values
Y is the current value of the function y = f (x, z) for
Z = constant

Xmin, Xmax are the minimum and maximum x
Coordinates for the function
Xinc is the increment between x values

Zmin, Zmax are the minimum and maximum z coordinates for the function

Zinc is the increment between z = constant planes

Initialize variables

Initialize the horizon arrays

Upper=0.0

Lower = Vertical Screen

Evaluate the function for each constant z plane

start with the closest plane, Zmax

start a loop for the x values

calculate the y value with the corresponding x and z values

convert x and y values to screen coordinate representations

compare the calculated y value with the upper and

lower horizons to satisfy the required conditions.

The relative steps are followed and the loop is repeated

until the conditions satisfied.

Z-buffer algorithm

Z-buffer algorithm uses a depth buffer which is used to store the z co-ordinate or depth of every visible pixel in image space. The depth or z value of a new pixel to be written to the frame is compared to the depth of that pixel stored in the z-buffer. If the comparison indicates that the new pixel in front of the pixel stored in the frame buffer then the new pixel is written to the frame buffer and the z-buffer updated with new z value. If not, no action is taken. Conceptually, the algorithm is a search over x, y for the largest value of z (x, y).

Pseudocode:

1. Initialize the depth buffer (Z-Buffer) and the refresh buffer, so that for all the buffer positions (x, y),

   Depth (x, y) = 0 ,

   Refresh (x, y) = intensity at the background.

2. For each position on each surface, compare depth (Z) values to previously stored values in the depth buffer to determine visibility .

   Calculate the depth Z for each (x, y) positions on the surface.

   If Z > depth (x, y), then set

   Depth (x, y) = Z

   Refresh (x, y) = intensity of the surface at x, y

After all surface have been processed, the depth buffer contains depth values
for the visible surfaces and the refresh buffer contains corresponding\g intensity values for those surface.

2.Server Communication Module

This process waits for the client to request. On client request it communicates with the data generation process and sends the data to the client. This is implemented using TCP Berkeley sockets protocol.

Processing Narrative:

The interprocess communication is established using Berkeley sockets. Berkley's Socket programming is also used to continuously poll the client request.

CLIENT PROGRAM

1. Client Communication Module

This program gets the data from the server .The communication is established using interprocess communication protocol.

Processing Narrative:

Data is generated in real time and sent through the socket to the client. The client side display program to generate the terrain uses this data.

2.Display Process Module

The display process reads the data sent through the socket and plots it. This module consists of developing a user interface in which the data is displayed.

Processing Narrative:

The data read is used to plot the terrain according to the menu selected. The terrain is generated at specific intervals of time. This is displayed in a separate drawing area.

Display module consists of plot function, which plots the terrain at frequent intervals of time. The position of the target can be relatively judged using the display simulated.

5. USER MANUAL

5. 1 Introduction

5. 1. 1 Product Rational and Overview

The product is mainly concerned with the simulation of the terrain scenario around the surveillance system. The data required to simulate the terrain is generated in the server using data generation process. The generated points are passed to the client via network using sockets. At the client side these points are received and passed on to the display process. The display process displays the corresponding terrain in the designed user interface.

5. 1. 2 Basic Features

The product is mainly developed for the simulation of a three dimensional terrain which is displayed in a user friendly GUI. The product has got it's own features like menu options and drawing area. The main menu options consist of File and Plot.

File menu consists of a sub menu quit. On selecting the quit option the user can exit the application.

The plot option consist of three sub option terrain1, terrain2 and terrain3, these three options are used for the display of different terrain scenarios in the drawing area

5. 1. 3 Summary of Display

The display consists of two types

Floating horizon:

The first two options under the plot menu simulate the terrain in which the hidden lines are eliminated using the floating horizon algorithm. This is an image space algorithm in which the curve representing the current data will be drawn after comparing with the previous curves and eliminating the hidden lines.

Z-buffer algorithm:

The third option under plot menu is used to simulate the three dimensional terrain which is displayed by eliminating the hidden lines using z-buffer algorithm. The pixels having the greatest depth value will be plotted

5.2 Getting Started

5. 2. 1 Menu

1. File

When the file menu option is selected it gives a sub menu quit

Quit submenu.

With the help of the left mouse button the user clicks the File Menu. This menu will in turn drop the pop up menu having quit submenu option. On clicking the quit submenu the whole software is exited.

and is rated for 415V. One 750 kVA is of Cummins make and the other is of Caterpillar m ake. They both are also rated for 415V. All gen-sets are provided with FRP cooling towers. At present, the mill is opting for captive power generation only during power failure.

Lighting:

There is a separate feeder for lighting which supplies power to all departments.

Metering at Power House:

Apart from electronic recorders in the EB side, the mill has its own metering arrangements as per following.

- Digital meters for measuring Voltage, Analog meters for measuring current and frequency are provided in the MV panels.

Location of Capacitors:

About 1041 kVAr capacitors are kept at the feeder end and no capacitor is provided at the powerhouse.

2. Plot

When the plot submenu is selected it gives three submenu options, terrain1, terrain2, terrain3.

Terrain1 submenu

This sub menu is used to display the terrain scenario simulated by the input generated in the server by eliminating the hidden line using Floating Horizon algorithm.

Terrain2 submenu

This sub menu is used to display the terrain scenario simulated by the input generated in the server by eliminating the hidden line using Floating Horizon algorithm.

Terrain3 submenu

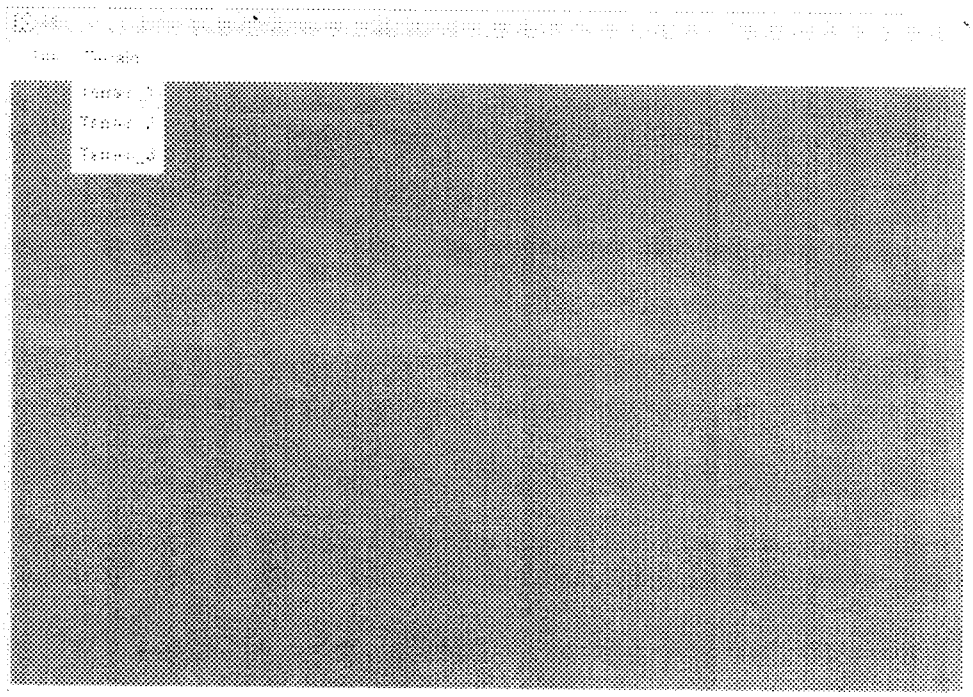
This sub menu is used to display the terrain scenario simulated by the input generated in the server by eliminating the hidden line using Z-buffer algorithm.

5.2.2 Sample Run

The following are the sample screen outputs from the software while being tested.

Screen 1

Main Menu- The user selects any option from the menu in this screen



6. PRODUCT TESTING

1. **Type of test: Functional Test**

Machine Configuration:

Processor:	Pentium III 500 MHz
Floppy Disk Drive:	1.44 MB
Hard Disk:	10 G.B
Operating System:	Red Hat Linux 7.1
Compiler:	gcc compiler 2.96 version.

Test assumption: communication between client and server.

Exact test Stimuli: To check the communication between the client and server under standard conditions.

Expected Outcome: The client application gets connected with the server properly and carries out it's functions.

1. **Type of test: Performance Test**

Machine Configuration:

Processor:	Pentium III 500 MHz
Floppy Disk Drive:	1.44 MB
Hard Disk:	10 G.B
Operating System:	Red Hat Linux 7.1
Compiler:	gcc compiler 2.96 version.

Test assumption: Correctness of Floating Horizon algorithm

Exact test Stimuli: To check the correctness of the floating horizon algorithm in removing the hidden lines for a given mathematical equation for surface plot.

Expected Outcome: The algorithm displays the series of curves for the given equation with all the hidden lines removed.

2. **Type of test: Performance Test**

Machine Configuration:

Processor:	Pentium III 500 MHz
Floppy Disk Drive:	1.44 MB
Hard Disk:	10 G.B
Operating System:	Red Hat Linux 7.1
Compiler:	gcc compiler 2.96 version.

Test assumption: Correctness of Z – buffer algorithm

Exact test Stimuli: To check the correctness of the z – buffer algorithm in removing the hidden lines for a given mathematical equation for surface plot.

Expected Outcome: The algorithm displays the series of curves for the given equation with all the hidden lines removed.

7.PROJECT LEGACY

7.1 Project Description

The product is mainly concerned with the plotting of the terrain, giving 3D view of the environment around the sea vessel. The terrain gets plotted every few seconds (i.e.) a predetermined interval is fixed.

7.2 Initial Expectations

The objective of the product designed is to generate the three dimensional view of the terrain, around the sea vessel. The inputs are received at real time from the server process console. After receiving the target details (as the input from the input module), the data are sent to the processing stage, in the server. Then the computed values are sent to the client, in response to its request. From there, sent to the display module. The display module plots the data.

7.3 Current Status of the project

This software performs operations like real time data generation, as a server process generates the data in real time and sends them for display to the client process, on its request .The client machine's process retrieves the data and sends it to its display process where in the terrain gets plotted, which is a periodical process. This can be further developed in future.

7.4 Activities/ Time logs

Time logs	Activities
Nov 24th	Product Definition
Nov 28th	Product Analysis
Dec 11th	Programming Language
Dec 17th	Risk Analysis
Dec 24th	SRS
Jan 11th	Design Documentation
Jan 25th	Coding for Server
Feb 5th	Coding for Client
Feb 10th	Coding for Floating Horizon Algorithm
Feb 15th	Coding for Z-buffer Algorithm
Feb 20th	Coding for GUI
Feb 24th	Prepare User Manual
Feb 28th	Test Plans
March 6th	Full Project Demo

7.5 Technical Lessons Learnt

Many Technical lessons were learnt while working on the project. The Software used such as the Xwindow, Open Motif Toolkit, were studied to implement the coding. Graphical concepts were learnt to implement the algorithms used. UNIX Network Programming with Berkley's Socket Programming was also learnt during the development of the software.

7.6 Managerial Lessons Learnt

During the course of the project, apart from the technical lessons certain Managerial aspects were also learnt .The systematic approach to a program, effective communication with the guide, and above all the team management helped in successful completion of the project .The time slots were set for each and every module and the time taken for each was noted as soon as they were finished

7.7 Recommendations for future

This project is mainly concerned with research and development in surveillance systems .The project can be further enhanced from the methods used to have more inferences to gather more information about the terrain scenario.

8. CONCLUSION

Real time generation of data and its simulated display gives a periodic update of the terrain scenario around the source vessel .It involves a lot of mathematical and logical calculations which can be developed in future. This project is designed in such a way that even a novice user can use the software efficiently, under the guidance of the User Manual, without causing any damage to the software.

The advantage of the project is that it implements a truly distributed system on a Linux Platform. The algorithms are made simple by reducing the complexity and it is mainly used for the speed of execution.

To conclude with the "Real Time 3D modeling In Surveillance Systems was an educative work, which was enjoyed by the team. The project also involves all the features for future enhancement.

Bibliography

1) **X Window System Programming** Second Edition, Nabajyoti Barkakati by Prentice Hall India

2) **Procedural Elements for Computer Graphics,** David.F.Rogers McGraw Hill, International Edition.

3) **UNIX Network Programming,** W. Richard Stevens by Prentice Hall India

4) **Linux COMPLETE,** Grant Taylor by BPB Publications

5) **Software Engineering,** Roger .S. Pressman McGraw Hill, International Edition.

6) http://www.openmotif.net

7) http://tronche.com/gui/x/xlib/

SOURCE CODE:

```
/*-------------------------------------------------------------------------
************************** T E R R A I N 3 D ******************************
*File: terrain.c
*Handles communication with Server and carries out display process
-------------------------------------------------------------------------*/
// Standard C Header Files
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <errno.h>

// Include files for X11 Library and Utility functions
#include <X11/Xlib.h>
#include <X11/Xutil.h>

// Include files for Motif Widgets
#include <Xm/Xm.h>
#include <Xm/RowColumn.h>
#include <Xm/MainW.h>
#include <Xm/DrawingA.h>
#include <Xm/Form.h>

// user defined Header files
#include "mathfunc.h" // math routine prototypes
#include "graph.h"    // graph routine prototypes
#include "xmutil.h"   // Make menu routine prototypes
#include "terrain.h"  // terrain routine prototypes
#include "genterrain.h"

// Definitions for TCP server program
```

```
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/uio.h>

// width and height for main window
#define WIDTH      800
#define HEIGHT     600

// structure to store points send from server
struct Points
{ int x[1000];
  int y[1000];
};

int XRes,YRes;
int PreCalcY[MaxY+1];

int CentreX,CentreY;
int Angl,Tilt;
float CosA,SinA;
float CosB,SinB;
float CosACosB,SinASinB ;
float CosASinB,SinACosB;

boolean PerspectivePlot;
float Mx,My,Mz,ds;

int Height[MaxRes+1][MaxRes+1];
int Scaling;
int MaxHeight;
int Res;

unsigned long bgpix;   // stores pixel value
```

```
GC   theGC;     /* GC for regular drawing */

// define widgets for window, form, menu and drawing area
Widget      top_level, main_window, menu_bar, drawing_area,drawing_area2,
          new_menu,form_win;

/*Equations*/

#define Span 5
float zf(float x,float y)
{
  float c;
  c=SqrFP(x)+SqrFP(y);
  return(75.0/(c+1.0));
}



XtAppContext app; // Application context for top level widget

/*********************************************************************
* Main function
*********************************************************************/
int main(argc, argv)
int  argc;
char **argv;
{
    Arg         args[MAXARGS]; // max arguments
    Cardinal    argcount;      // argument counter
    int         fg, bg, drawbgpix;
    XGCValues   xgcv;          // object to GC Values

 /* XtAppContext app;*/
```

```
    unsigned long drawfg = 5555;

/* Create the top-level shell widget and initialize the toolkit*/
    top_level = XtAppInitialize(&app, "Terrain3d", NULL, 0,
                                &argc, argv, NULL, NULL, 0);

/* Next, the main window widget */
    argcount = 0;
    XtSetArg(args[argcount], XmNwidth,WIDTH );   argcount++;
    XtSetArg(args[argcount], XmNheight, HEIGHT); argcount++;
    main_window = XmCreateMainWindow(top_level, "Main",
                                     args, argcount);
    XtManageChild(main_window);

/* Create the menu bar */
    menu_bar = XmCreateMenuBar(main_window, "Menubar", NULL, 0);
    XtManageChild(menu_bar);

/* Create the Form*/
    form_win=XtVaCreateManagedWidget("form",xmFormWidgetClass,
                        main_window,
                        NULL);
/* Create the "File" menu ----------------------------------*/
    new_menu = MakeMenuPane("File", menu_bar,
                        "Quit", quit_action, NULL,
                        NULL);
/* Create the "File" cascade button and attach new_menu to it */
    AttachToCascade(menu_bar, "File", new_menu);
/* Create the "Terrain" menu -------------------------------*/
    new_menu = MakeMenuPane("Terrain", menu_bar,
            "Terrain_1", gen_terrain1, NULL,
              "Terrain_2", gen_terrain2, NULL,
            "Terrain_3", gen_terrain, NULL,
            NULL);
```

```
/* Now create the cascade button and attach this sub menu to it */
   AttachToCascade(menu_bar, "Terrain", new_menu);

/* Create the drawing area */
   argcount = 0;

   XtSetArg(args[argcount], XmNresizePolicy, XmRESIZE_NONE);
argcount++;
   XtSetArg(args[argcount], XmNbackground, 0); argcount++;
   XtSetArg(args[argcount], XmNwidth, 640); argcount++;
   XtSetArg(args[argcount], XmNheight,400); argcount++;
   XtSetArg(args[argcount], XmNleftAttachment,
   XmATTACH_FORM);argcount++;
   XtSetArg(args[argcount], XmNleftOffset, 1); argcount++;
   XtSetArg(args[argcount], XmNtopOffset, 2); argcount++;
   XtSetArg(args[argcount], XmNtopPosition, 6); argcount++;
   XtSetArg(args[argcount], XmNtopWidget, menu_bar); argcount++;
   XtSetArg(args[argcount], XmNborderWidth, 12); argcount++;
   XtSetArg(args[argcount], XmNresizable, True); argcount++;

   drawing_area = XmCreateDrawingArea(form_win,
                  "drawing_area", args, argcount);
   XtManageChild(drawing_area);

/* Create drawing area 2*/
    argcount=0;
    XtSetArg(args[argcount], XmNresizePolicy, XmRESIZE_NONE);
argcount++;
    XtSetArg(args[argcount], XmNbackground, 0); argcount++;
    XtSetArg(args[argcount], XmNwidth, 20); argcount++;
    XtSetArg(args[argcount], XmNheight, 20); argcount++;
    XtSetArg(args[argcount], XmNtopOffset, 12); argcount++;
    XtSetArg(args[argcount], XmNtopWidget, drawing_area); argcount++;
    XtSetArg(args[argcount], XmNborderWidth, 12); argcount++;
```

```
    drawing_area2 = XmCreateDrawingArea(form_win,
                        "drawing_area2", args, argcount);
    XtManageChild(drawing_area2);

/* Attach the menu bar and the drawing area to main window */
/*  XmMainWindowSetAreas(main_window, menu_bar, NULL, NULL,
                NULL, drawing_area
                    );*/
/* Create the GCs. First retrieve the background and foreground
 * colors from the widget's resources.
 */
    argcount = 0;
    XtSetArg(args[argcount], XmNforeground, &fg); argcount++;
    XtSetArg(args[argcount], XmNbackground, &bg); argcount++;
    XtGetValues(drawing_area, args, argcount);

/* Define a GC with these colors */
    xgcv.foreground = fg;
    xgcv.background = bg;
    theGC = XtGetGC(drawing_area, GCForeground | GCBackground,
              &xgcv);
    XSetForeground(XtDisplay(drawing_area), theGC, drawfg);
    XSetForeground(XtDisplay(drawing_area2), theGC, 3333);

/* Add callback to handle expose events for the drawing area
    XtAddCallback(drawing_area, XmNexposeCallback, handle_expose,
            &drawing_area);
*/
/* Add event handlers for terrain to be drawn
    XtAddEventHandler(drawing_area, ButtonPressMask, False,
                draw_pixel, NULL);*/
/* Realize all widgets */
    XtRealizeWidget(top_level);
```

```
/* Start the main event-handling loop */
   XtAppMainLoop(app);
} // End of Main



/*--------------------------------------------------------------*/
/* q u i t _ a c t i o n
 *
 * This routine is called when the "Quit" item is selected from
 * the "File" menu.
 */
void quit_action(w, client_data, call_data)
Widget          w;
XtPointer       client_data;
XmAnyCallbackStruct *call_data;
{
   XtCloseDisplay(XtDisplay(w));
   exit(0);
}

/*--------------------------------------------------------------*/
/* g e n _ t e r r a i n 1
 *
 * Routine to plot terrain1
 */
static void gen_terrain1()
{
struct sockaddr_in ServAddr;
struct Points p;
int RSize,WSize;        // No. of bytes Read & Wrote
```

```
int i,z;
int ConnSoc,ConnID;

if( (ConnSoc=socket(AF_INET,SOCK_STREAM,0)) < 0 )
   perror("Cannot Create Manager Socket. Terminating..."), exit(0);

bzero(&ServAddr,sizeof(ServAddr));
ServAddr.sin_family=AF_INET;
ServAddr.sin_port=htons(2000);
ServAddr.sin_addr.s_addr=inet_addr("127.0.0.1");
ConnID=connect(ConnSoc,(struct sockaddr *)&ServAddr,sizeof(struct
sockaddr));
if(ConnID<0)
   perror("Cannot Connect Manager with Agent. Terminating..."), exit(0);

XClearWindow(XtDisplay(drawing_area), XtWindow(drawing_area));

for(z=320;z>=100;z=z-2)
 {
RSize = read(ConnSoc,&p,sizeof(p));
if(RSize<=0)
   printf("Cannot Read from the Agent.");
for(i=0;i<1000;i++)
  XDrawPoint(XtDisplay(drawing_area), XtWindow(drawing_area), theGC,
p.x[i],p.y[i]);
  sleep(1);
 //XtAddTimeOut(5000*10000,gen_terrain1,NULL);
}
 close(ConnID);
}
```

```
/*-------------------------------------------------------
 * g e n _ t e r r a i n 2
 *
 * Routine to plot terrain2
 */
static void gen_terrain2()
{
  int x,x2,z,i,j;
  double abc,y,y2,s,c,t,xyz;
  double hor[640],hor1[640];
  for(i=0;i<640;i++)
  {
          hor[i]=0.0;
          hor1[i]=479.0;
  }
  XClearWindow(XtDisplay(drawing_area), XtWindow(drawing_area));

  for(z=100;z<=150;z=z+2)
  {
   for(x=-319;x<=320;x++)
   {
    abc=((x-180)^2) + ((z-180)^2);
    s=x*3.14/180;
    c=z*3.14/180;
    xyz=sqrt((s*s)+(c*c));
 /*   y=(6*cos(s+c)+50);*/

  y=100*(sin(xyz)*cos(xyz));
  x2=x+319;
  y2=240-y;
```

```
 if(y2>=hor1[x2] && y2<hor[x2])
 {
 }
 else
 {
 XDrawPoint(XtDisplay(drawing_area), XtWindow(drawing_area),
theGC,x+319,240-y);

 if(y2<=hor1[x2])
        hor1[x2]=y2;
 else
        hor[x2]=y2;

 XtAddTimeOut(1*10, gen_terrain2, NULL);
 }
  }
 }
}


/* ----------------------------------------------------
 * g e n _ t e r r a i n
 *
 * Routine to plot terrain using z-buffer algorithm
 */
static void gen_terrain()
{

/*  TERRAIN terrain[];*/
  XClearWindow(XtDisplay(drawing_area), XtWindow(drawing_area));

  ClearHeightBuffer();
  Res=MaxRes;
  InitPerspective(false,0,0,600,600);
```

```
  InitPlotting(240,18);
  CreateEquationPlotHeightBuffer(-Span,Span,-Span,Span);

}

void HeightBufferScalingFactor()
{
  Scaling = 32767 / MaxHeight;
}

void ClearHeightBuffer()
{
int i, j;

for(i=0; i<=MaxRes; i++)
{
  for(j=0; j<=MaxRes; j++)
        Height[i][j] = 0;
  }
}

void PreCalc()
{
  Word j;

  for(j=0;j<=MaxY;j++)
  PreCalcY[j]=0;
  for(j=0;j<=MaxY;j++)
  PreCalcY[j]=XRes*j;

}

/*Three Dimensional Plotting Routines*/
```

```
void InitPlotting(Ang, Tlt)
int Ang;
int Tlt;
{
  CentreX=MaxX>>1;
  CentreY=MaxY>>1;
  Angl=Ang;
  Tilt=Tlt;
  CosA=CosD(Angl);
  SinA=SinD(Angl);
  CosB=CosD(Tilt);
  SinB=SinD(Tilt);
  CosACosB=CosA*CosB;
  SinASinB=SinA*SinB;
  CosASinB=CosA*SinB;
  SinACosB=SinA*CosB;
}

void InitPerspective(Perspective, x, y, z, m)
boolean Perspective;
float x, y, z, m;
{
  PerspectivePlot=Perspective;
  Mx=x;
  My=y;
  Mz=z;
  ds=m;
}

void MapCoordinates(float X, float Y, float Z, int *Xp, int *Yp)
{
  float Xt,Yt,Zt;
  Xt=(Mx+X*CosA-Y*SinA);
  Yt=(My+X*SinASinB+Y*CosACosB+Z*CosB);
```

```
  if(PerspectivePlot)
  {
    Zt=Mz+X*SinACosB+Y*CosACosB-Z*SinB;
    *Xp=CentreX+Round(ds*Xt/Zt);
    *Yp=CentreY-Round(ds*Yt/Zt);
  }

  else
  {
    *Xp=CentreX+Round(Xt);
    *Yp=CentreY-Round(Yt);
  }
}
void CartesianPlot3D(X, Y, Z)
float X;
float Y;
float Z;
{
  int  Xp,Yp;
  MapCoordinates(X,Y,Z,&Xp,&Yp);
  XDrawPoint(XtDisplay(drawing_area),XtWindow(drawing_area), theGC, Xp,
Yp);
//  XtAddTimeOut(1*1, gen_terrain, NULL);
}
void CreateEquationPlotHeightBuffer(Xlft, Xrgt, Ybot, Ytop)
float Xlft, Xrgt, Ybot, Ytop;
{
  int ix,iy,iz;
  float x,y;
  float dx,dy;
  MaxHeight=Res;
  HeightBufferScalingFactor();
  dx=(Xrgt-Xlft)/(Res-1);
  dy=(Ytop-Ybot)/(Res-1);
```

```
 for(ix=0;ix<Res;ix++)
 {
  x=Xlft+ix*dx;
  for(iy=0;iy<Res;iy++)
  {
   y=Ybot+iy*dy;
   iz=Round(zf(x,y));
   if(iz<0)
   {
       exit(1);
   }
   Height[ix][iy]=iz;
   CartesianPlot3D(ix-Res/2,iy-Res/2,iz);
  }
 }
}
```

```
/*-------------------------------------------------------------------------
* **************************** terrain.h ************************
-------------------------------------------------------------------------*/

/* Function prototypes */

void handle_expose(/* Widget w, XtPointer client_data,
                                XmDrawingAreaCallbackStruct *call_data */);

void quit_action(/* Widget w, XtPointer client_data,
                              XmAnyCallbackStruct *call_data */);

/* This function selects the terrain */
static void select_terrain(/* Widget w, XtPointer client_data,
                                          XmAnyCallbackStruct *call_data */);

/* This function draws the terrain */
static void draw_terrain(/* Display *d, Window w, GC gc, int terrains */);

static void gen_terrain();

static void gen_terrain1();

static void gen_terrain2();
```

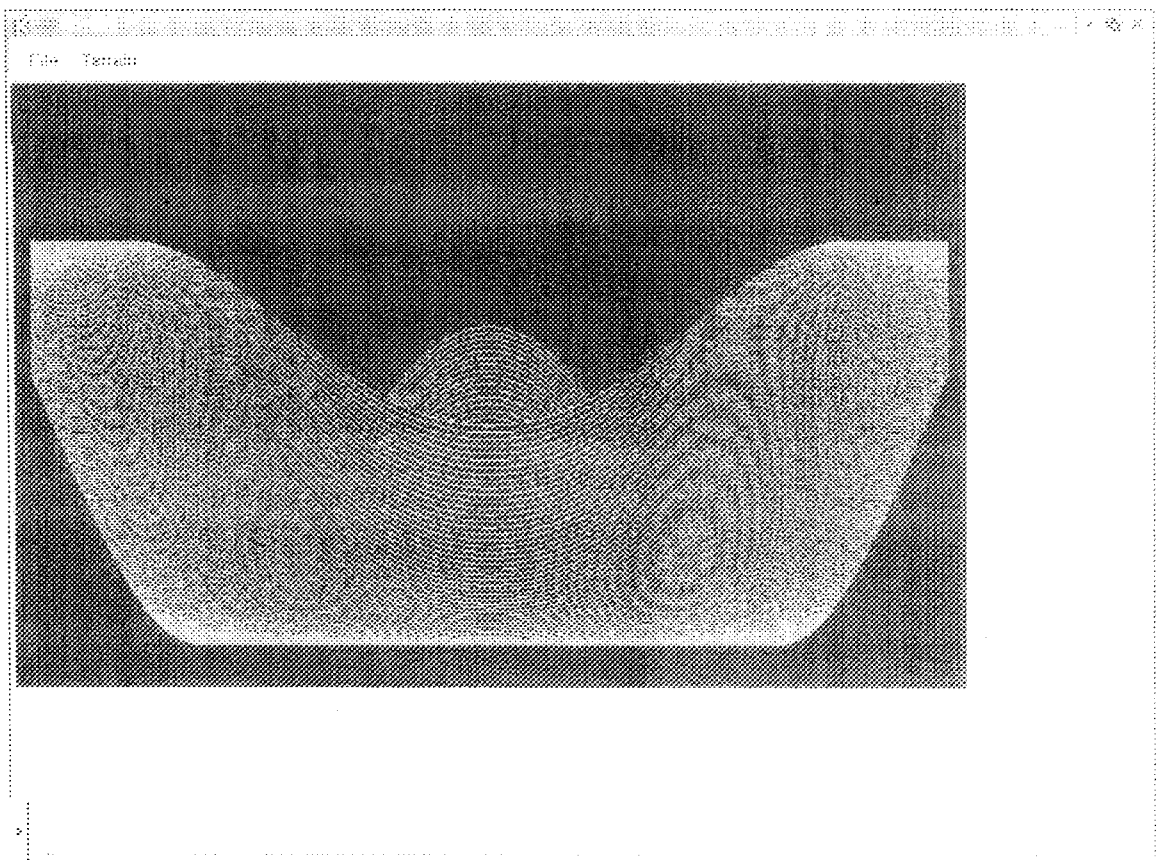
```
/*-------------------------------------------------------------------------
*************************** genterrain.h ***********************
*-------------------------------------------------------------------------*/

#ifndef __GENTERRAIN_H__
#define __GENTERRAIN_H__

void HeightBufferScalingFactor();

void ClearHeightBuffer();

void PreCalc();

/*Three Dimensional Plotting Routines*/

void InitPlotting(int, int);

void InitPerspective(boolean, float, float, float, float);

void MapCoordinates(float, float, float, int *, int *);

void CartesianPlot3D(float, float, float);

void CreateEquationPlotHeightBuffer(float, float, float, float);

#endif /* __GENTERRAIN_H__ */
```

```
/*-------------------------------------------------------------------------
* ***************************** mathfunc.h *********************
-------------------------------------------------------------------------*/

#ifndef __MATHFUNC_H__
#define __MATHFUNC_H__

/* Function returns (Rounded off)integer value of a floating point number */
int Round(float x);

/* Function returns square of a floating point number */
float SqrFP(float x);

/* Function returns square of a integer number */
int Sqr(int x);

/* Function returns angle in Radians */
float Radians(float Angle);

/* Function returns angle in Degrees */
float Degrees(float Angle);

/* Function returns cosine of angle in Degrees */
float CosD(float Angle);

/* Function returns sine of angle in Degrees */
float SinD(float Angle);

#endif /* __MATHFUNC_H__ */
```

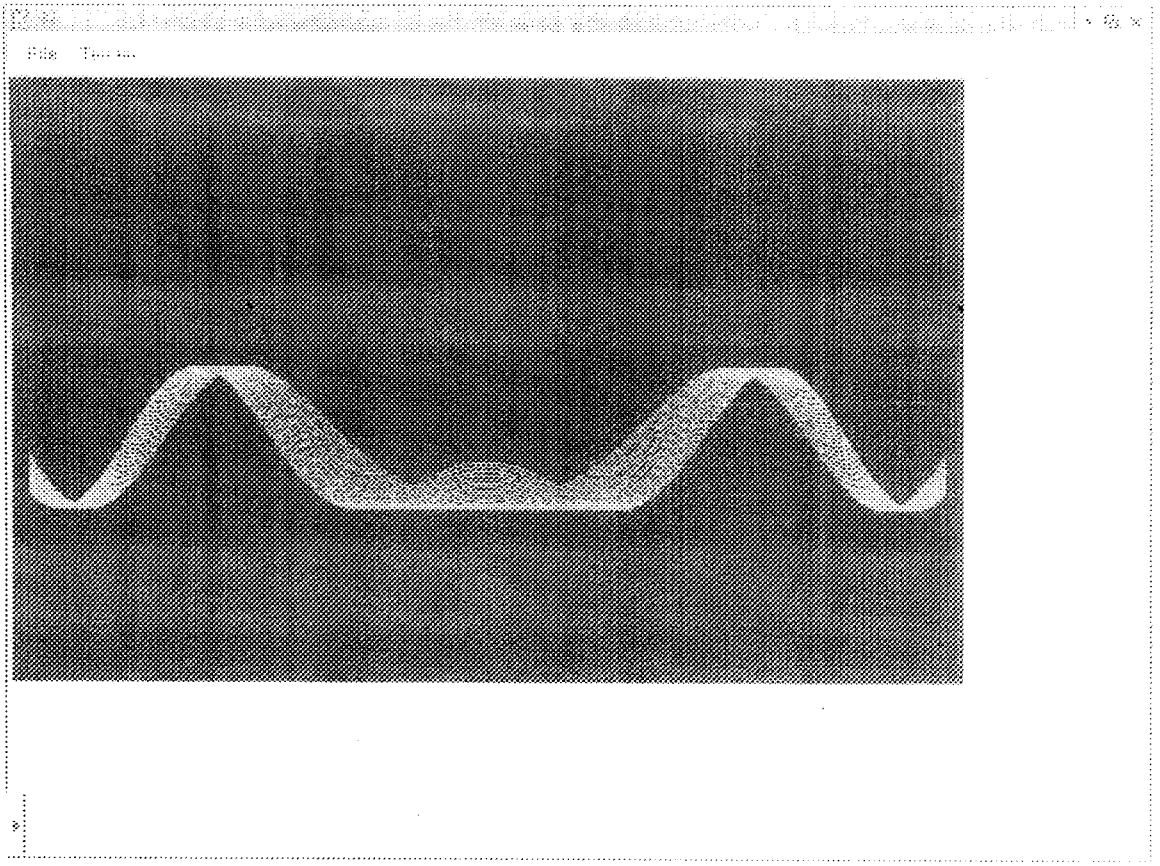
```
/*-------------------------------------------------------------------------
***************************** File: xmutil.h ********************
-------------------------------------------------------------------------*/

#ifndef __XMUTIL_H__
#define __XMUTIL_H__

/* This function, defined in file "xmutil.c" prepares menus */

Widget MakeMenuPane(/* char *name, Widget parent, ... */);


/* This function, also in xmutil.c, attaches a menu to a
 * cascade button
 */

void AttachToCascade(/* Widget parent, char *label,
                        Widget sub_menu */);

#endif /* __XMUTIL_H__ */
```

OUTPUT:

Terrain 1



Terrain 2



Terrain 3