Defence Science Journal, Vol. 52, No. 3, July 2002, pp. 277–284

Search and Render Algorithm for Three-dimensional Terrain Visualisation of Large Dataset

Sudhir Porwal, B.K. Pradhan, J.K. Sharma and S.C. Jain

Defence Electronics Applications Laboratory, Dehradun – 248 001

ABSTRACT

There exists a wide class of algorithm for 3-D modelling and visualisation of terrain. These techniques are not very useful for flight simulation application. A very large terrain has to be modelled for flight simulators and primitive techniques do not support large data handling. The terrain keeps on changing rapidly, so in flight simulator application, rendering of each frame must be very fast. This paper describes an algorithm which handles very large dataset and can generate 3-D frames quickly. A flight simulator application can be designed using this technique.

Keywords: Modelling, simulation, flight simulation, terrain evaluation, 3-D terrain visualisation, terrain modelling, remote senssing, dataset, search and render algorithm

1. INTRODUCTION

The satellite-based remote sensing technology has made possible the acquisition of volumetric dataset of the earth's atmosphere. Because of technological progress, high-resolution data of earth is available which results in large storage in our databases. The 3-D modelling of the satellite imagery using the digital elevation data is one of the major activities nowadays. The 3-D modelling requires terrain elevation data, which is irregular in nature and cannot be defined by mathematical equations. It can best be described by triangular irregular network[1]. Triangular irregular network is used to generate the digital elevation data. The polygon approximation technique is used for terrain modelling[2], medical imaging[3], scientific data modelling[4], etc. This technique is the simplest and useful for small data rendering. As the data size grows up, the frame computation time shoots up exponentially. The other technique, B-spline curve method[5] is also computationally costly. This paper presents a technique that reduces the frame computation time up to an acceptable limit and with frame rate almost independent of data size. This provides a smooth visualisation and a flight simulator[6] can be built using this technique.

Large dataset visualisation contains a lot of problems. Consider a case when user wishes to see the full data in one frame. It is possible if and only if the user is viewing the entire data from a high altitude. System has to draw complete 3-D model of the terrain in a single frame but that can lead to a very poor frame rate. If the user is looking at the terrain from very low altitude, a small portion of the image only will be visible, but a high-resolution data will be used for creating such a view. A complex situation occurs when the user is near the ground and looking out towards the horizon. A very large fraction of the terrain will be visible in the distance because of perspective. Also, as the user turns his/her head or moves quickly over the terrain, the terrain changes rapidly.

Revised 24 December 2001

This paper discusses a search and render technique that calculates the visible objects depending on the viewing direction and the viewpoint parameters, and it draws only visible objects in the 3-D frame. As soon as the viewing parameters change, it recalculates the visible objects and draws into the new frame. This technique provides a consistent frame rendering performance independent of the size of dataset.

2. GENERATION OF DATABASE

The large dataset of the terrain must be stored in such a data structure that provides easy retrieval and rendering of the terrain data. It should also support search and render algorithm. The visualisation of the terrain from different viewpoints and altitudes requires the multiresolution data. The portion of the terrain near to the eye must be rendered using high-resolution imagery but in contrast, the farthest portion of the terrain can be rendered using low-resolution imagery. The selection of the proper resolution data for rendering depends on the level of details of the terrain, but this topic is beyond the scope of the paper. This technique is implemented using openGL graphics library[6]. OpenGL provides a mip-mapping technique that automatically selects the proper resolution image data for rendering. The other algorithm[7] for level of details selection can also be used. A multiresolution representation of terrain, divided into equal-size elements called tiles will be created, i.e., only those tiles of the resolution required for the given viewpoint need to be retrieved from the database.

A landform can be represented by two functions: (i) the elevation function $Z(x, y)$, and (ii) image function $I(x, y)$. Here, $Z(x, y)$ gives the height at (x, y) coordinates on the plane and $I(x, y)$ gives the R, G, B intensity at (x, y) point. The landform will be divided into small equal size tiles as shown in Fig. 1 and a multiresolution image pyramid will be generated to store multiresolution imagery. A multiresolution pyramid can also be used for elevation function but for simplicity, only image pyramid is discussed.

An image pyramid is shown in Fig. 2 that contains the original image at the bottom level

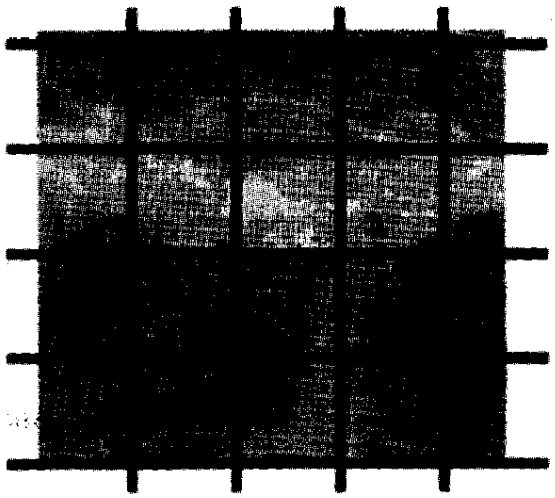

Figure 1. Tiling of landform

($L = 0$) and down-sampled images as the level increases. If the tile size at level $L = 0$ is 256, then it is 64 at $L = 1$, 32 at $L = 2$, and so on. Number of tiles is same at each level. The level of the pyramid can be increased up to any level depending on the memory available on the system. Figure 3 shows how the multiresolution images are used to create a scene. The distance of each tile from viewer or eye location decides the resolution of imagery for rendering the scene.

Let it be assumed that the landform is situated in the first coordinant of the Cartesian coordinate system as shown in Fig. 4. The bottom-left corner

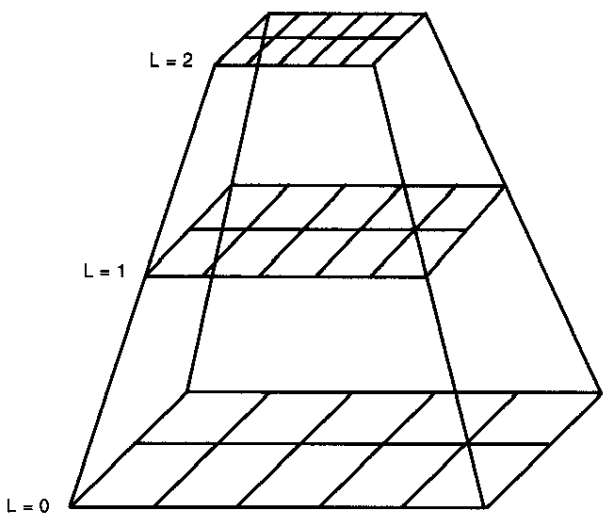

Figure 2. Image pyramid

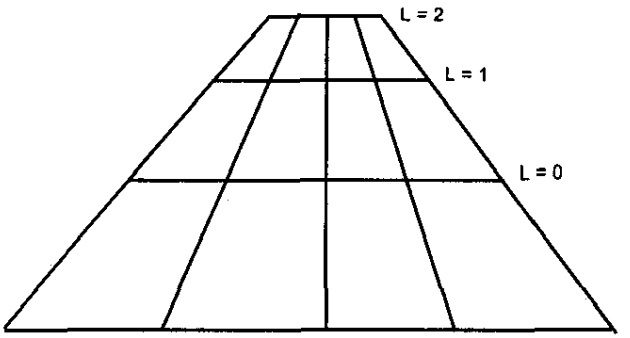

Figure 3. Use of image pyramid

of the landform is over the origin. If the size of the tile is *S*, the width and the height of the landform is *X* and *Y*, respectively.

The number of tiles in a row will be

$$Tr = X/S$$

The number of tiles in a column will be

$$Tc = Y/S$$

Total number of tiles

$$T = Tr*Tc$$

Now, a number will be given to each tile. The bottom-left tile will be the 1st tile, and then number will be given to each tile row-wise as shown in Fig. 4, i.e., each tile will have a unique number. This tile identification number will be used by search and render algorithm to fetch the correct data from the database.

The coordinate position of the bottom-left corner of each tile will be stored in a data structure

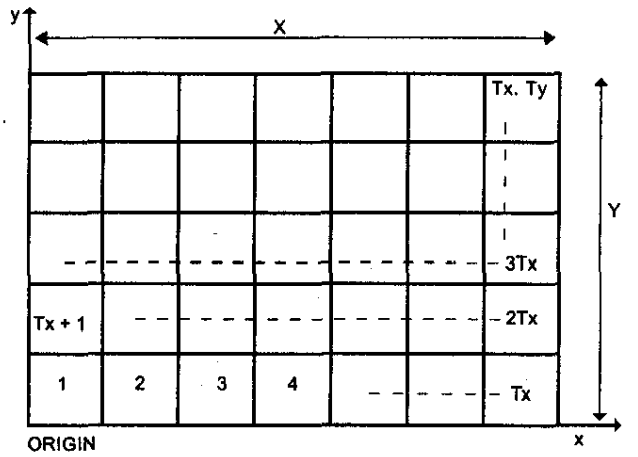

Figure 4. Numbering of titles

to fix the location of each tile. It could be a simple 2-D array, as

$$\text{Array } [T][2] = \{(x_1, y_1), (x_2, y_2), \ldots\ldots (x_T, y_T)\}$$

where $(x_1, y_1), (x_2, y_2)\ldots\ldots (x_T, y_T)$ are bottom-left corner coordinate of each tile. The identification numbers will be the same for down-sampled images in image pyramid. OpenGL will automatically select which image sample to use for rendering.

3. SETTING UP VIEWING VOLUME

Generally, two types of projections[8] are used for 3-D rendering, i.e., orthogonal and perspective projections. Orthogonal projection is best suited for engineering drawing applications, but for real-world object rendering, animation and simulation, perspective projection is used. The characteristic of perspective projection is foreshortening; the farther an object is from the camera, the smaller it appears in the scene. This occurs because the viewing volume for a perspective projection is a frustum of pyramid. The object that falls inside the frustum is visible. The frustum is shown in Fig. 5.

Few parameters have to be set for a frustum. These are:

FoV	Field of view
Z_{near}	Near clipping plane
Z_{far}	Far clipping plane
Aspect	Ratio of *b/a*

Setting the value of FoV is similar to selecting lens for a camera. One can choose a wide-angle lens, normal lens, and telephoto lens by just varying the value of FoV. If the FoV is very large, the number of visible objects will increase and rendering mechanism has to draw more objects. This can lead to poor performance. The FoV can be set according to the requirement. It is a trade-off between the quality of the scene and the frame rate. Let the FoV for this application is *F*. Z_{near} and Z_{far} are the distances of near and far clipping planes, respectively from the viewpoint. If an object is within the frustum but lies beyond the $(Z_{near} - Z_{far})$

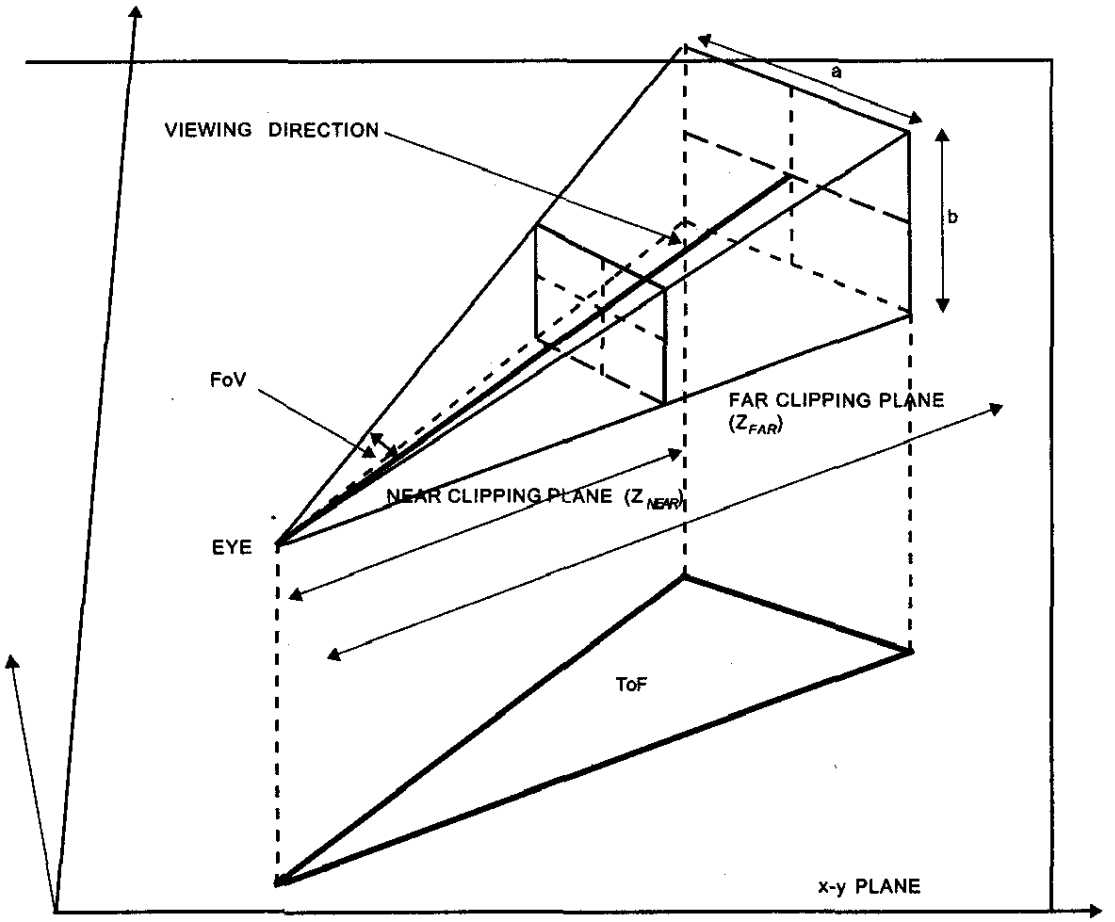

Figure 5. Frustum of perspective projection

range, the object will be discarded for rendering. This range gives the flexibility to render a limited number of objects. If the viewer is parallel to the ground and looking towards the horizon, it is not necessary to render a very large landform. The system will render only those objects that come in the clipping range (Z_{near}– Z_{far}). Let it be assumed that the Z_{near} is 1 and Z_{far} is Z.

Aspect is nothing but the ratio of y dimension and x dimension of the frustum. It should be 1 to maintain the actual shape of the rendered object. The mathematical modelling[8] can be done for this perspective projection.

4. SEARCH & RENDER TECHNIQUE

The parameters of the viewing volume have been fixed in previous section. The viewer will navigate the landform using this frustum only. The viewer can move forward, backward, turn right, turn left, move up, and move down. Every time, a new location of the viewer will change the position of frustum and visible objects have to be re-calculated for new scene rendering. If the viewer or eye location (point E) in 3-D space is given by (E_x, E_y, E_z) and the point location (point P), where the viewer is looking at, is given by (P_x, P_y, P_z), then the line between these two points (viewing direction) will decide the location of the frustum. Generally, two situations occur; (i) when the frustum is more or less parallel to the horizon and (ii) when the frustum is perpendicular to the landform and the viewer is looking vertically down at the landform. In this algorithm, second situation has not been discussed because there is no point of looking at a 3-D model of landform from the top at high altitude. This requires a lot of processing time to calculate the visible portion of landform. As the

altitude increases, more and more objects are visible and the system has to render a large number of objects in a single frame, which is generally not required. Only the first situation in this technique is considered.

If viewing direction is parallel to the landform, the projection of the frustum on landform will be a triangle (Fig. 5). Let this triangle be called triangle of frustum (ToF). Objects inside the triangle will be visible to viewer and other objects can be discarded. Let it be presumed that the point P is fixed and E_x, E_y (components of point E) are also fixed. Let E_z (altitude of the viewer) be varied by a considerable amount; then it can be seen that as E_z changes, projection of the frustum changes on ground but lies within the ToF. The ToF will be the upper bound of all the projections of frustum and rendering of ToF will be sufficient for every frame. The vertices of the ToF can be calculated easily.

The vertices of the ToF (Fig. 6) can be easily calculated using simple mathematics. Once the Z_{far} and FoV are fixed, d is always fixed. It can be calculated and stored permanently. As and when the viewer location or the looked at point P is changing, the new vertices of the ToF are calculated. A mathematical technique can be evolved to check

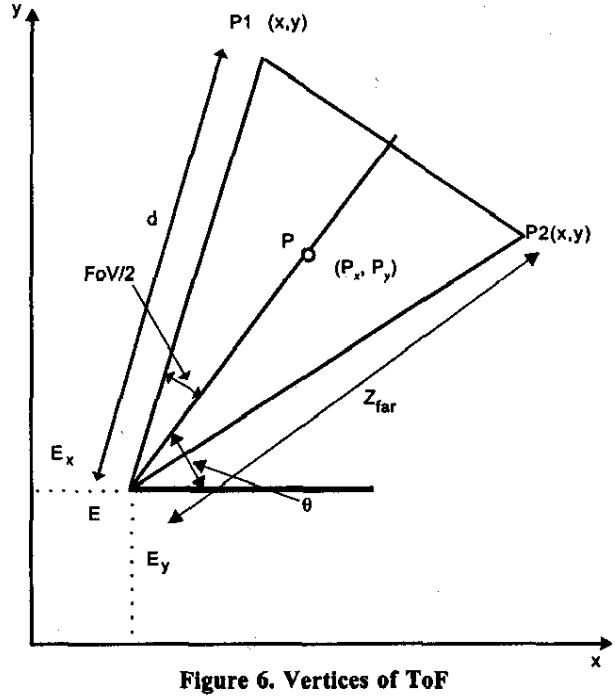

Figure 6. Vertices of ToF

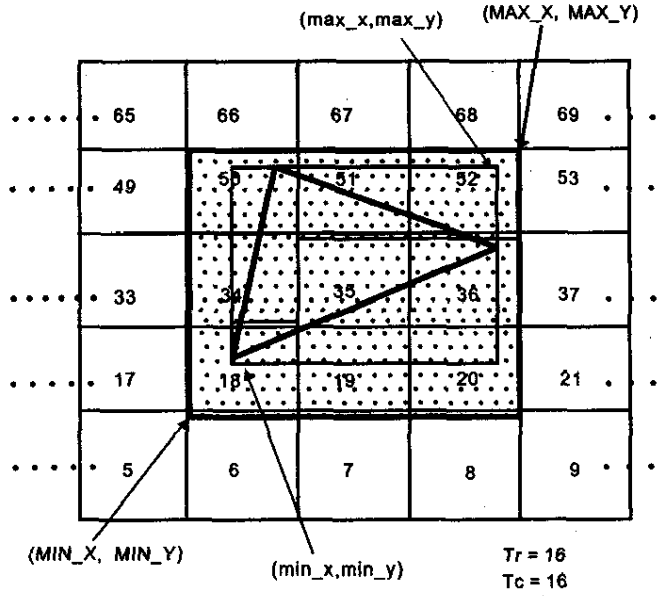

Figure 7. Enclosed rectangle of ToF

the object location, i.e., weather the object is inside or outside the triangle. In this case, the objects are nothing but the tiles. If each tile is checked for containment within the triangle, the approach will become data size-dependent. As the data size increases, number of tiles also increase, and visible tile identification takes more time. It can lead to poor rendering response.

To overcome this problem, it is proposed to use enclosed rectangle of the ToF to get a probable list of the visible tiles shown in Fig. 7. The minimum of E_x, $P1\ (x)$, $P2(x)$ and E_y, $Pl(y)$, $P2(y)$ will give the bottom-left corner (min_x, min_y) of the enclosed rectangle and the maximum of E_x, $Pl(x)$, $P2(x)$ and E_y, $Pl(y)$, $P2(y)$ will give the upper-right corner (max_x, max_y) of the enclosed rectangle.

The enclosed rectangle of ToF contains few complete tiles and few partial tiles. This enclosed rectangle is stretched diagonally in such a way that the new enclosed rectangle covers the full tiles. The diagonal coordinate of new enclosed rectangle becomes (MIN_X, MIN_Y) and (MAX_X, MAX_Y). This new diagonal coordinate determines the number of tiles inside the enclosed rectangle. In the case of ToF (Fig. 7), algorithm generates a list of tiles as given below:

18, 19, 20, 34, 35, 36, 50, 51, 52

Table 1. Time complexity in CPU seconds

No. of samples	Polygon Approx. technique	Search & render technique
32 × 32	0.010	0.010
64 × 64	0.010	0.010
128 × 128	0.040	0.010
256 × 256	0.110	0.040
512 × 512	0.271	0.160
768 × 768	0.861	0.161
1024 × 1024	1.680	0.171
3072 × 3072	–	0.172

These tiles have to be rendered for appropriate scene generation. This method does not depend on the size of the dataset. The calculation overheads are more or less the same for every frame rendering. This technique provides a constant frame rate and can visualise a large dataset very easily. The time complexity of the search and render algorithm is shown in Table 1.

It is compared with the polygon approximation technique which is the simplest among other techniques (so a comparison with other techniques is implied). One can easily see the huge difference in the time as the data size increases. The time taken by the search and render algorithm to draw a frame is nearly independent of data size. This is the important feature of this algorithm. This comparison data is collected on a pentium III 550 MHz computer.

This technique renders some of the tiles that are totally or partially outside the ToF. Checking the intersection of each tile with the boundaries of ToF can reduce it but it will be a very complex mathematical calculation. This calculation can take more time than rendering the tiles that are totally or partially outside the ToF. So, it is better to render each tile present in the enclosed rectangle. Any rendering approach[9,10] can be used for each tile rendering. This technique has been implemented in C using OpenGL graphics library. The output is shown in Fig. 8.

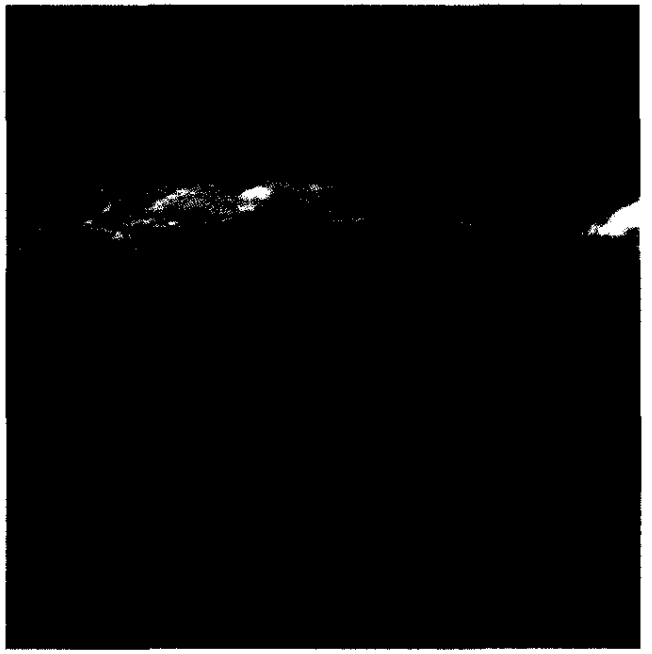

Figure 8. Output of software

5. CONCLUSIONS & FUTURE WORK

This study has shown that it is possible to visualise the 3-D model of a large dataset with consistent frame rate. It is also possible for the viewer to efficiently navigate across the landform. This study can be extended in several ways. The digital map features like roads, communication lines, rivers, point themes, etc. can also be overlaid on the 3-D model to make it more informative. This technique does not cover the case of high altitude rendering. It requires more efforts to handle the high altitude view without compromising the rendering speed.

ACKNOWLEDGEMENTS

The authors are thankful to Dr A.S. Bains, Director, Defence Electronics Applications Laboratory (DEAL) for his cooperation. They are also thankful to their colleagues at the Image Analysis Centre, DEAL for providing the required dataset.

REFERENCES

1. Mirante, Anthony & Weingarten, Nicholas H. The radial sweep algorithm for constructing triangulated irregular network. *IEEE CG&A*, May 1982, 11–21.

2. Anupam. Delaunay triagulation-based surface modelling and three-dimensional visualization of landform. *IETE Techn. Review*, November-December 1998, **15**(6), 425–33.

3. Cook, Larry T.; Dwyer III, Samual J.; Batnitzky, Soloman & Lee, Kyo Rak. A three-dimensional display system for diagnostic imaging applications. *IEEE CG&A*, August 1983, **3**(5), 13–20.

4. Grotch, Stanley L. Three-dimensional and stereoscopic graphics for scientific data display and analysis. *IEEE CG&A*, November 1983, **3**(8), 31–44.

5. Tiller, Wayne. Rational B-splines for curves and surface representation. *IEEE CG&A*, September 1983, **3**(6), 61–69.

6. Woo, Mason.; Neider, Jackie.; Davis, Tom & Shreiner, Dave. OpenGL programming guide. Ed. 3. Version 1.2. Addison-Wesley Publishing Co., December 1999.

7. P. Lindstorm, *et al.* Real-time continuous level of detail rendering of height fields. *In* Proceedings Siggraph 92. Addison Wesley, Logman, Reading, Mass., 1996. pp. 109–18.

8. Watt, Alan. Fundamental of three-dimensional computer graphics. Addison Wesley Publishing Co., 1989.

9. Klasky, Ronald S. Computer animation for visualization terrain data. *IEEE CG&A*, May 1989, 12–13.

10. Upson, Craig; Faulhaber, Thomas Jr.; Kamins, David; Laidlaw, David; Senlegel, David; Vroom, Jeffrey; Gurwitz, Robert & Vam Dam, Andries. The application visualization system: A computational environment for scientific visualization. *IEEE CG&A*, July 1989, 30–42.

Contrbutors

Mr Sudhir Porwal received his BE (Computer Science and Engg) from the G.B. Pant Engineering College, Garhwal, in 1996. Presently, he is working in the Image Processing Group of the Defence Electronics Applications Laboratory (DEAL), Dehradun. He has developed software related to mission planning, terrain modelling and measurements, image processing activities in Unix and NT environment. Areas of his research include: Artificial intelligence, three-dimensional modelling, computer vision, object-oriented programming, and design techniques.

Mr BK Pradhan received his BE (Electrical Engg) from the University of Bengal in 1969. He has served the Indian Armed Forces for 21 years. Presently, he is working as Scientist in the Image Processing Group at the DEAL, Dehradun. Areas of his research include: Image processing, geographical information system, and cartography.