Defence Science Journal, Vol. 57, No. 1, January 2007, pp. 149-162

Real-time Photorealistic Visualisation of Large-scale Multiresolution Terrain Models

Anupam Agrawal[1], R.C. Joshi[2], and M. Radhakrishna[1]

[1]Indian Institute of Information Technology, Allahabad–211 011
[2]Indian Institute of Technology, Roorkee–247 667

ABSTRACT

Height field terrain rendering is an important aspect of GIS, outdoor virtual reality applications such as flight simulation, 3-D games, etc. A polygonal model of very large terrain data requires a large number of triangles. So, even most high-performance graphics workstations have great difficulty to display even moderately sized height fields at interactive frame rates. To bring photorealism in visualisation, it is required to drape corresponding high-resolution satellite or aerial phototexture over 3-D digital terrain and also to place multiple collections of point-location-based static objects such as buildings, trees, etc and to overlay polyline vector objects such as roads on top of the terrain surface. It further complicates the requirement of interactive frame rates while navigation over the terrain. This paper describes a novel approach for objects and terrain visualisation by combination of two algorithms, one for terrain data and the other for objects. The terrain rendering is accomplished by an efficient dynamic multiresolution view-dependent level-of-detail mesh simplification algorithm. It is augmented with out-of-core visualisation of large-height geometry and phototexture terrain data populated with 3-D/2-D static objects as well as vector overlays without extensive memory load. The proposed methodology provides interactive frame rates on a general-purpose desktop PC with OpenGL-enabled graphics hardware. The software TREND has been successfully tested on different real-world height maps and satellite phototextures of sizes up to 16K*16K coupled with thousands of static objects and polyline vector overlays.

Keywords: Digital terrain models, level-of-detail management, multiresolution modelling, real-time rendering, photorealistic visualisation, terrain-rendering algorithm

1. INTRODUCTION

Computer-generated perspective imagery of mission-critical terrain has been recognised as a valuable tool for acquiring strategic and tactical insight. Currently available high quality digital representation of earth features is facilitating visualisation with greater detail and informational content. Photorealistic terrain visualisation is achieved by combining terrain topography with terrain image or phototexture. Terrain topography is represented in the form of elevation samples and is also known as height field or digital elevation model (DEM). There are varieties of sources to acquire DEM data of the area of interest, which includes digitised contour data from topographic maps, satellite or aerial optical stereo image pair, microwave synthetic aperture radar (SAR) images (using interferometry or radargrammetry), etc.

The image or terrain phototexture is obtained from a satellite or by any other suitable mechanism

Revised 16 March 2006

with near-vertical view angle. Such imagery is widely available from commercial remote sensing satellites such as SPOT, LANDSAT, IRS, IKONOS and QuickBird. Apart from the above two sets of data, the visualisation may have multiple collection of point-location-based static objects such as buildings, trees, etc and polyline vector objects such as roads, district boundaries, etc on top of the terrain surface. Such high quality visualisation along with real-time navigation over digital terrain has significant applications in many defence, civil, and training-related areas.

Many factors determine the quality of a real-time 3-D visual simulation. Two of these are the fidelity of the scenes rendered and the frame rate of the simulation, factors that are at odds with each other[1]. High fidelity scenes provide a greater degree of realism while high frame rates, in excess of 30 frames per second, provide smooth motion. The challenge is to add many additional features that enhance realism of the user application while keeping in view the desired fidelity and frame rate.

On the face of it, rendering of terrains is simple. The basic input is a height field, which can be drawn on the screen as a network of triangles. However, simple calculations show that this is not practical on hardware available on the desktop PCs today. Consider real-world digital terrain data set covering 160 km*160 km area at a 10 m resolution. This makes a total of 256 million points. Assuming that each point is stored as an 8-bit integer, the height field would require a total of 256 MB RAM. The same size coloured image with Red-Green-Blue bands would require a total of 768 MB memory. This excludes the memory required to represent objects that populate the terrain. The rendering of the height field would involve the drawing of around 512 million triangles. The high-end desktop graphics card, Nvidia's GeForce4 MX 460 claims a theoretical maximum of 38 million triangles per second only. Not to mention that transferring total 1024 MB of data across the graphics pipeline minimum 30 times a second is far beyond bandwidth capacity of today's buses. This shows the complexity of the real-time rendering of large terrain data sets.

Restricting visualisation to small terrains or using low-resolution terrains[2,3] was suggested, but this has limited practical value. To render the large terrain data sets in real-time, it is necessary to reduce the complexity of the scene while maintaining a high image quality. Level of detail (LOD) modelling provides efficient mechanism to represent and manipulate complex object details by optimising the trade-off between complexity and accuracy of representation[4]. Multiresolution polygonal mesh simplification algorithm have been used to generate multiple surface models at varying levels of detail, and appropriate techniques to select and render the appropriate LOD model have been employed.

This paper discusses the methodology and implementation aspects to improve the quality and speed of rendering of large terrains on general-purpose desktop PCs. The proposed LOD algorithm uses a compact and efficient multiresolution grid representation of height fields and employs a variable screen-space threshold to limit the maximum error in the projected image. The method is different from the individual triangle-based LOD algorithm and is optimised for modern, consumer 3-D graphics cards and minimises CPU usage during rendering. It is augmented with out-of-core visualisation of large height geometry and texture terrain data. To display large collection of point-location-based static objects over the terrain while maintaining the real-time frame rate, an efficient object handling method has been proposed using paging technique and object instantiation. User is allowed to control the objects locations, scales and orientations. Display of polyline vector data over multiresolution 3-D terrain has been accomplished using an efficient geometry-based mapping approach.

2. RELATED WORK

Image-based modelling and rendering (IBMR) and many polygonal mesh simplification (PMS) techniques for terrain height fields have been developed for terrain visualisation. The IBMR techniques model a scene by combining 3-D geometry and 2-D image sprites; these create a texture map from an object represented at a high resolution, which is used to texture the same object represented at a lower resolution–reintroducing details contained in the texture[5,6].

One of the PMS approaches employ regular hierarchical structures to represent the terrain, whereas the second PMS approach is characterised by the use of less constrained triangulations or triangulated irregular networks (TIN). The most established methods of the first class make use of triangle bin-trees/quadtrees[7-9], and edge bisections[10]. These structures facilitate compact storage due to their regularity, as topology and geometry information is implicitly defined. Approaches of the second class include data structures like multi-triangulations[11], adaptive merge trees[12], hyper-triangulations[13] and the adaptation of progressive meshes to view-dependent terrain rendering[14]. Triangulated irregular networks are able to reduce number of necessary triangles since these are much better adopted to high-frequency variations. However, to capture irregular refinement or simplification operations and connectivity, a complex data structure is needed. Also, the algorithm are complex and CPU-intensive for dynamic view-dependent simplification.

Regular hierarchical structure were chosen to represent terrain (stored as height map) as it allows fast collision detection between the moving camera or viewer position and the terrain. It also supports use of efficient hierarchical data structures for fast and easy view frustum culling. Most of the LOD-based terrain-rendering algorithm attempt to generate triangulations, which optimally adapt to terrain given as a height map, and hence are CPU-intensive. This definition of optimality specifies as few triangles as possible for a given quality criteria. Today, the absolute number of triangles is not as important. Today's graphic hardwares make it possible to render lots of triangles quickly (of the order of 40 million triangles per second or more). However, to overcome the problem of limited-memory bandwidth, it is necessary to supply the geometry to the card in a specific manner, which usually means the creation of long triangle strips. Block-based view-dependent dynamic LOD terrain-rendering algorithm[15,16], considers the above facts using 3-D rendering hardware and minimises the CPU overhead. The algorithm cleverly avoids T-junctions and cracks in the multiresolution surface while generating long triangle strips with significant gain in rendering performance[17]. Image draping over 3-D mesh geometry is performed using texture mipmapping using OpenGL18 3-D API.

Relatively less work has been reported in literature on object management over multiresolution terrain. Szenberg[19], *et al*. describe a method of terrain visualisation with point-location-based objects such as houses, transmission poles, etc and overlay of polyline vector, objects such as transmission lines. The visualisation scheme for terrain height field is not based on multiresolution modelling but combines the Z-buffer with the floating-horizon algorithm. Also, results are shown on limited-sized terrain data (512*512 size) only. Douglass[20], *et al*. describe a bottom-up LOD height field rendering scheme by placing building objects over the terrain. In contrast to a top-down LOD approach, a bottom-up approach necessitates the entire model being available at the first step, and therefore, has higher memory and computational demands[4]. Zachary[21], *et al*. extend the approach of Douglass[20], *et al*. to overlay polyline vector data over multiresolution 3-D terrain. Kersting[22], *et al*. describe a texture-based rendering of polyline vector data onto the LOD terrain geometry.

In this paper, a new object management approach coupled with block-based multiresolution LOD terrain modelling approach has been proposed. It employs an efficient object-paging scheme and multiresolution modelling of polyline vector data, which smoothly adapt with tile-based organisation of geometry and texture data for out-of-core data management.

3. TERRAIN DATA GENERATION

For generation of terrain data pertaining to height maps, two sources of data, namely Survey of India supplied topographic maps containing contours depicting surface relief and National Remote Sensing Agency (NRSA) supplied IRS-1C/1D stereo images for the area of interest are used. Dehradun and surrounding hilly regions in northern India were selected for this purpose. An efficient Delaunay triangulation-based surface-reconstruction scheme has been developed to generate the height map from digitised contour data[2,23]. An attempt was also made to generate the height map from the satellite stereo image pair using PCI Geomatica software. The raw satellite images are required to be preprocessed for sensor-error correction, geometric correction, reduction of atmospheric effects, mosaicing, image enhancements, etc as per the need. The modified registered image (PAN or LISS-3) has been used for image draping over the 3-D views of terrain. An efficient algorithm to fuse the two sets of images such that the resulting image contains high spectral contents of the LISS-3 image and high structural contents of PAN images[24] has been developed.

Although the use of remote sensing imagery allows very realistic landscape visualisation, the vertical sides of objects, such as building's fronts, individual trees, etc are not recorded by the sensor. AutoCAD 3-D modelling package and billboarding techniques are used to represent complex and natural objects respectively. Further more, facility is provided for the user to interactively digitise vector data such as roads from the scanned topographic map to overlay and display over 3-D LOD views of the terrain.

4. RENDERING LARGE TERRAIN GEOMETRY AND TEXTURE IN REAL TIME

The main objective of multiresolution terrain rendering research is to create visual simulations of very large terrains on inexpensive desktop PCs. The rendering must be done in real-time ensuring frame rates above 30 frames per second.

4.1 Data Preprocessing

4.1.1 Terrain Geometry Data

Since data sets are very large, these will not fit into main memory in its entirety. To address this problem for terrain navigation application, it is proposed to organise the digital height map data in tiles of size $(2^n+1) * (2^n+1)$ pixels with $n = 8$. One pixel overlap is kept between adjacent geometry tiles to ensure proper stitching of tiles. Only those terrain tiles that need to be rendered or are at least near the camera, need to be loaded from disk and stored in main memory. A group of 3 x 3 tiles, which are active (loaded in main memory) at a given time has been considered. The viewer is assumed to be inside the centre tile. An efficient indexing scheme has been developed to organise the digital terrain data as collection of tiles on secondary storage[16].

A quadtree was constructed for each active tile of the terrain height map where size of each leaf block or patch of the quadtree was 17*17 (the size decided after experimentation). The data held by each node of the quadtree include the minimum x and z and maximum x and z coordinates (bounding box) of the terrain represented by this node. The root node of the quadtree stores the values of the minimum and maximum coordinates of the whole tile. Each leaf node has the index of the block it represents. This terrain block layout is chosen such that the block can be optimised for rendering, using one draw-primitive call for the entire block and, even better, using indexing to get rid of multiple transformations of vertices[17]. Multiresolution pyramid representation was used to define each leaf block of size 17*17.

4.1.2 Satellite Texture Images

The combination of geospecific textures with digital elevation models (height maps) allows construction of photorealistic visual simulation of landforms. For real-world terrain models, large, high-resolution textures need to be processed which do not usually fit into graphics texture memory. Previously, the whole satellite image texture was used for image draping using OpenGL API mipmapping[18]. In such a case, due to the limitation on texture buffer size on Pentium IV machine, the data size was restricted to 4K*4K. Current techniques developed[15] enable the use of large texture data in tiles of size 256*256 pixels similar to geometry tiles, and establishes a one-to-one correspondence between these. Also during rendering, anisotropic filtering has been used instead of conventional mipmapping scheme supported by OpenGL API. This enhances the rendering quality and also improves locality because the level of filtering is chosen by the maximum partial derivative.

In the direct texture tiling approach, visible seams appear at the edges of texture tiles when interpolation is used to smooth textures. This occurs where no information is available for neighbouring textures. It can also be exacerbated by the wraparound behaviour of interpolation, where the first pixel colour is used to influence the colour of the last pixel in a tile row. The texture-clamping feature of the OpenGL was used and also modification to the texture tile generation process, which uses a power of 2 tile size with a single pixel of redundancy on all sides of the texture. Further, to ensure that landcover features in texture image fall at appropriate places over height map, texture image was to be reprojected to adjust pixel size.

4.2 Run-time

4.2.1 Reducing Polygon Flow in Rendering Pipeline

In the proposed framework, dynamic terrain paging and view-frustum culling techniques were used to control substantial amount of polygons in the rendering pipeline. Initially, the viewer was assumed to be standing in the middle of centre tile. As the viewer walkthroughs near the edge of the centre tile, three new tiles in the direction of movement were paged-in and three old tiles in the opposite direction were paged-out. Terrain paging has been used for the out-of-core management of large terrain data.

To avoid excessive time spent on rendering polygons that are not within the field-of-view, the terrain was intersected with the view frustum and renders only those blocks at quadtree leaves that are part of this intersection (Fig. 1). For a field-of-view of 90 degree, the culling stage generally reduces the number of polygons to less than half. The quadtree representation of tile data enables very fast view-frustum culling[16].

4.2.2 Adaptive View-dependent Geometry Refinement

After dynamic terrain paging and view-frustum culling, one is left with a set of terrain blocks or patches (size 17*17), which can be immediately sent to the graphic pipeline. Because terrain complexity (surface roughness) can be quite high, one will not get the desirable frame rate unless the terrain is very small; above-mentioned methods will not suffice for high complexity terrain. In fact, terrain blocks that are far away from the camera do not need to be rendered with the same detail as terrain blocks that are near the camera. These can be approximated by a lower-resolution version; thereby drastically decreasing triangle count and increasing render speed. This leads to LOD rendering of height maps.

Each block (size 17*17) of the terrain data in multiresolution hierarchy was organised using pyramid data structure. The pyramid level 0 represents the default (highest) resolution terrain block, and pyramid level N [$N \in (1,\rightarrow)$] represents each successive lower resolution version. Figure 2 shows the four pyramid levels of the height map block of size 17 * 17. To simplify the multiresolution method, simply skipping samples were opted to build a lower-resolution version of a block. Therefore, lower-resolution blocks do not have to be pre-generated. One can precalculate the resolution level and camera distance relationship for all the

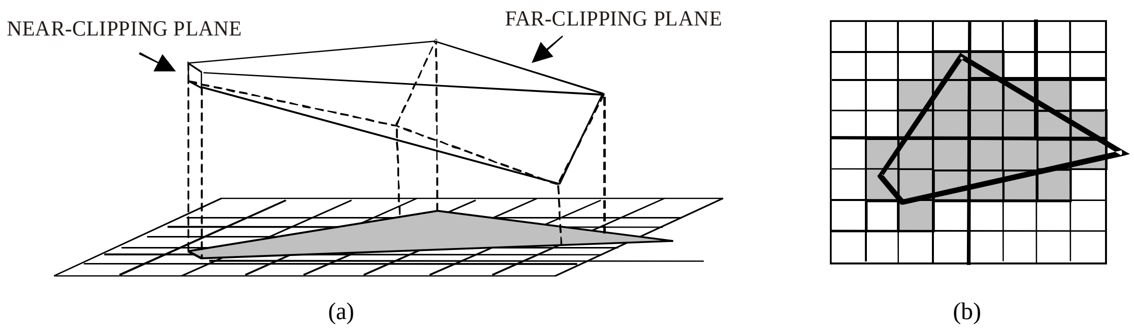


Figure 1. View-frustum culling – top and bottom planes are not being used: (a) 3-D view-frustum and its projection over 2-D plane, (b) visible and partially visible quad cell blocks.

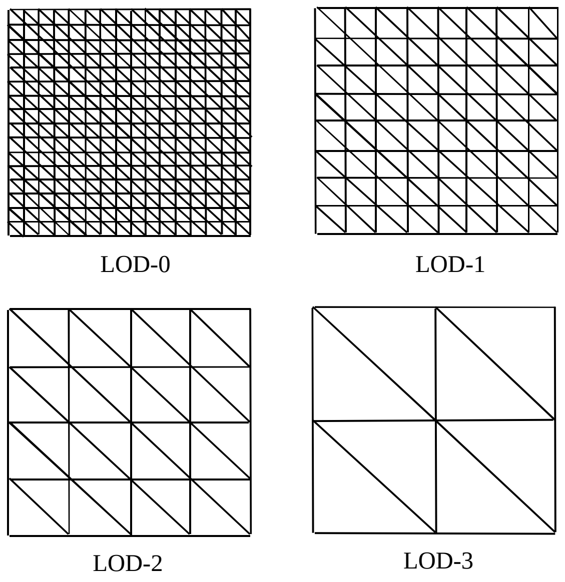


Figure 2. Multiresolution modelling of height map.

blocks of the active tiles at terrain-tiles load time. Calculating the maximum geometric error between the current level of the block and a lower-resolution block (for given distance from the camera to the centre of the terrain block) generates a screen space error (∈) in pixels. If this error is smaller than a user-defined threshold (τ), then the algorithm will render the block with the lower resolution.

To speed up the dynamic level of detail selection, one precalculates same for each visible terrain block and prepares a Look-up Table to decide the tessellation level of the block based on position of the camera from the block[16]. To calculate the same, one treats the camera's direction vector as being permanently horizontal (worst case which may lead to more triangles in some cases). The graphics hardware takes care of these extra triangles, and hence, the above scheme brings down CPU overhead to a minimum.

4.2.3 *Removing Terrain Artifacts in Multiresolution Rendering*

It is important to note that in a view-dependent framework, the resolution of adjacent patches might change at every frame. Hence, cracks occur on borders of adjacent patches of different levels of detail. In Fig. 3 (a), the circle shows the position of crack in tessellation with level difference one

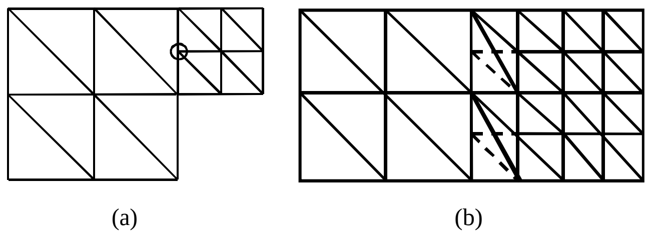


Figure 3. Removing cracks between adjacent patches: (a) before crack removal (b) after crack removal.

(right side patch is shown partially). Crack-filling methods usually involve creating additional triangles to fill in the gaps between patches, and/or modifying the geometry of one or other of the patches to produce a crack-free join. Figure 3(b) shows the modified geometry to remove the cracks where the dashed edges are excluded in triangulation and the bold edges are included. Similar procedure is followed to eliminate cracks when level difference between adjacent patches is two or three. Figure 4 shows a snapshot of the LOD wiremesh view with adjacent patches of different resolutions.

4.2.4 *Optimising Rendering Speed using Long Triangle Strips*

Current graphics hardware is now capable of rendering vast number of triangles, but graphics throughput to the hardware remains a bottleneck. Triangle strips are the most efficient primitives on today's video cards as these save CPU-to-card bandwidth by sending fewer than three vertices per triangle to the graphics pipeline.

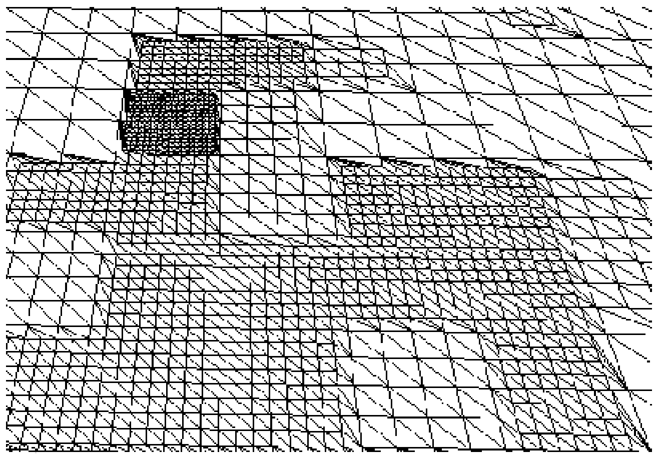


Figure 4. LOD wiremesh view of terrain geometry.

Triangle strips have been used extensively for static mesh representation. However, using such triangle strip representation and generation techniques is not practical for a multiresolution triangle mesh. In a view-dependent framework, the underlying mesh topology might change at every frame, thus making it difficult to use triangle strips. The triangle strip generation scheme for view-dependent dynamic multiresolution terrain shows significant improvement in rendering speed as compared to individual triangle-based and triangle-fan-based rendering schemes[17].

5. OBJECT MANAGEMENT OVER DIGITAL TERRAIN

There may be thousands of objects placed over the terrain. These objects may include man-made objects such as various kinds of buildings, lampposts, signposts, as well as roads, district boundaries, etc and also natural objects such as various kinds of trees, etc. Here, roads or district boundaries are polyline-based vector objects whereas the others are point-location-based objects. One may further divide the point-location objects into two categories. The first one is simple objects, those having simple geometry and can be drawn with the help of OpenGL primitive functions. These objects do not need to be loaded into memory. Examples of simple objects include sky-scrapper buildings drawn using elongated cube with texture mapping over its exposed faces and also objects created using billboarding technique. The second category is of complex objects having complex geometry and large number of triangles. These objects are required to be loaded into main memory containing their vertices and topology information. Examples of such objects include complex 3-D geometrical models of buildings, trees, etc.

The software has the provision to import such complex structures from .dxf format and convert into native .mesh extension files obtained after selecting relevant-only information. Complex objects may consume substantial memory as well as drawing time and may severely affect the rendering performance. So one usually prefers to use simple objects to populate the terrain. To cater for the requirement of populating the terrain using multiple copies of the same object, a methodology has been proposed for efficient memory management and maintaining real-time rendering performance. Further, an efficient geometry-based mapping algorithm has been proposed to render polyline vector data over 3-D multiresolution terrain.

5.1 Rendering Point-location-based Objects

While walkthrough over the 3-D terrain, user cannot see all the objects at once. Thus there is no need of keeping all of them in memory and render them. The objective is to deal with a large number of objects over the terrain. The number of objects is not constant throughout the navigation process; also more objects can be edited, added, or even deleted from the objects list.

5.1.1 Placing Multiple Instances of Objects Over Terrain

The first task is the deployment of various buildings on the terrain. The various types of buildings and houses being rendered on the terrain would give a look of a good human settlement. The buildings, which can be used over here, can be of the following types: like designed in Opengl or a DXF- 3DFACE building designed using AutoCAD package. The software helps the user to place a building anywhere on the terrain by showing the scanned georeferenced map or image in the background inside a 2-D window. As soon as the user clicks on the map (after choosing the kind of building (s)he wants to deploy), the building is placed at the appropriate position over the 3-D terrain.

The user can change various parameters associated with different buildings such as width, height or depth and then save the changes appropriately. An object may have its multiple instances with possibly different scaling factors. If an object is already been loaded into memory, on its subsequent occurrences, simply the pointer of object memory is returned for further processing and its counter is incremented by one. Thus the multiple loading of the same objects can be avoided.

For many objects, the technique of billboarding has been used (which is sometimes of great use as the loss in frame rate is quite negligible) and it shows the 2-D images just like 3-D objects.

Billboarding is a technique that adjusts an object's orientation so that it faces some target, usually the camera. Billboarding can be used to cutback on the number of polygons required to model a scene by replacing geometry with an impostor texture. Billboarding guarantees that the texture is always facing the camera, therefore the user never realises that the tree is in fact a flat texture quad. This particular technique can be used in the cases when finer details are not required about the object. In this study small size .bmp or .tga images has been used as textures, which takes very less space and hence helps in achieving a better frame rate, i.e., faster rendering of the terrain with objects.

5.1.2 Paging and Display of Objects Data

When user opens the model layer, a message is sent to ‘Objects’ class to open the particular file, followed by loading of names and locations of all the objects those are there in the file. These all objects (i.e., their geometry and texture) are not loaded into memory, but the objects of same tiles are grouped together, so that objects of current nine tiles can be loaded into memory. The software internally manages a dynamic data structure to store tile-wise objects details (without their geometry and topology information). The geometry and topology information of only those objects are kept in main memory which are inside current active nine tiles.

When new tiles are loaded and old are deleted, 'AddModel' function is called to add new models and "RemoveModel" is called to remove old models of corresponding tiles. The 'AddModel' function is called prior to 'RemoveModel' function to optimise the time by avoiding loading and unloading overheads. Objects paging helps in out-of-core management of large objects data on secondary storage.

5.2 Rendering Polyline Vector Data

Polyline vector data represents one major category of geographic information and defines geometry as lists of 2-D coordinates. Narrow linear features such as roads, railway lines, etc are usually not visible on a satellite remote sensing image used in creating 3-D phototextured views of the terrain. Other vector features such as state or country boundaries, property lines, etc, usually used for logical demarcation, are also required to be overlaid on top of the 3-D views with different type, shape, and display properties. These vector features are separately digitised from corresponding topographic map.

There are two options to render polyline vector data on a 3-D mesh. One option is to convert the polyline data to a texture image layer and combine this polyline image layer with the primary terrain texture image layer (e.g., from a satellite/aerial photograph). The second option is to render the polyline data as separate 3-D geometric primitives. One may call these two approaches as polyline-as-texture solution[22] and polyline-as-geometry solution[21] respectively. The second approach was preferred as it supports interactive enabling and disabling of the display of different subsets of polyline data and interactive adjustment of line styles such as line colour, width, and stipple patterns to distinguish and highlight different geographic data. The approach suggested by Zachary[21], *et al*. is not directly usable in the present case because multiresolution LOD mesh simplification scheme is different.

Displaying 2-D polyline data on top of 3-D terrain becomes challenging in the terrain visualisation system for several reasons. First, the display of vector layer should be limited to current nine raster tiles active at a time. Second, in the polyline-as-geometry solution, the 2-D polyline data should be treated independently from the raster data, and therefore, should be rendered as separate geometry by the graphics pipeline. This presents a challenge because terrain artifacts are likely to occur unless vector-data is mapped consistently and exactly to current LOD of terrain geometry.

5.2.1 Interactive Capture of Vector Data

The software helps a user to digitise linear vector features to overlay on top of the 3-D views. It displays the 2-D scanned topographic map or raster image in the background. A user may select the vector digitisation option and capture a linear feature as collection of 2-D points on a polyline.

The user may choose varying spacing between points based on the curvature of the vector layer. It automatically computes the intermediate points between two successive points. The height of terrain on all the points over the polyline vector feature can be retrieved from the DEM quadtree during run-time. After digitisation of different vector features, the vector layer may be saved in a file.

5.2.2 *Data Storage and Display of Vector Data*

Internally the digitised vector data is stored as per tile layout of corresponding geometry and texture image data. Hence a vector feature, which is extending between two tiles, will be stored as two vector segments. Whenever the user opens a vector data file, a linked-list is created in primary memory to store vector segments for all the tiles. Tile-wise vector segments information helps to display selective vector information based on current active nine tiles. A user may add new vector features in the existing vector file. In the multiresolution LOD rendering framework, the underlying mesh geometry (due to patch resolution) is changing at nearly every frame. Therefore for the polyline data to appear smoothly overlaid on the 3-D mesh, the rendered polyline geometry (i.e., height values at points) should therefore also change at each frame. The algorithm of multiresolution modelling of vector data allows the system to adapt the visual mapping without rendering artifacts to the context and user needs while maintaining interactive frame rates[25].

6. SOFTWARE ARCHITECTURE, USER INTERFACE, AND NAVIGATION

6.1 Software Architecture

The software TREND (acronym for **T**errain **Rend**ering) is an object-oriented 3-D application that can be used to model and interactively visualise real-world environments on a desktop PC. Architecture of the system is shown in Fig. 5.

To achieve real-time performance even on very large data size, TREND Data Modeller organises the geographic raster and vector data in tiles of size 256*256 grid-cells. User can selectively display different types of data as per the requirement. For

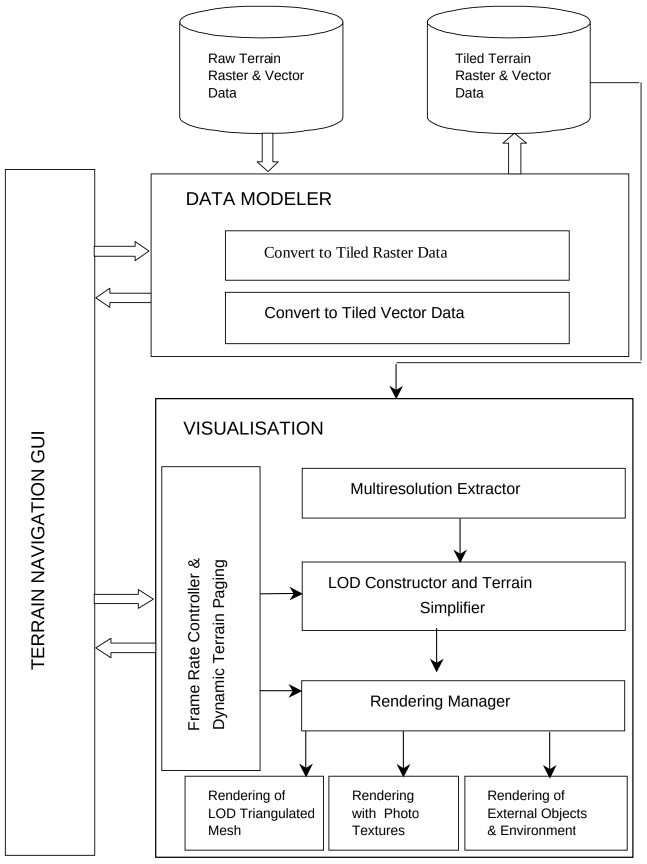

Raw Terrain Raster & Vector Data
Tiled Terrain Raster & Vector Data
DATA MODELER
Convert to Tiled Raster Data
Convert to Tiled Vector Data
TERRAIN NAVIGATION GUI
VISUALISATION
Frame Rate Controller & Dynamic Terrain Paging
Multiresolution Extractor
LOD Constructor and Terrain Simplifier
Rendering Manager
Rendering of LOD Triangulated Mesh
Rendering with Photo Textures
Rendering of External Objects & Environment

Figure 5. System architecture diagram.

example, user can visualise the terrain height map data in triangulated wireframe form and then drape satellite phototexture over it, if required. Similarly, user can switch-on and switch-off object layer (containing various objects such as buildings, trees, lampposts, etc) as well as vector layer (representing polyline features such as roads, district boundaries, etc) over the level of detail mesh. Subsequently, a user may enable or disable various special effects such as fog, multitexturing etc. Multiresolution Extractor module helps to decide the resolution level of a terrain patch based on its distance from the camera and user-specified image quality metric (τ). LOD Constructor and Terrain Simplifier module generates LOD representation of the mesh under view-frustum and removes possible artifacts such as cracks between different resolution patches and also constructs long triangle strips of the LOD mesh. Rendering Manager module displays various types of views of the terrain as explained above. The Frame Rate Controller and Dynamic Terrain

]

Paging modules help Rendering Manager to control frame rate based on user-specified image quality metric (τ) and maintain required nine tiles in main memory based on user (or camera) position respectively.

6.2 User Interface and Navigation

The software supports different user-interaction modes for terrain navigation through keyboard, mouse and also through voice.

7. RESULTS AND ALGORITHM PERFORMANCE ANALYSIS

The software TREND is developed in the Microsoft Visual C++ using the OpenGL 3-D API libraries for a Win32 environment. Microsoft speech SDK 5.1 is used for designing voice interface to the software. The software has been tested with 4K*2K terrain raster dataset of Grand Canyon and 16K*16K terrain data set of Puget Sound area obtained from Georgia Institute of Technology website. The height map of Dehradun (India) area has also been generated using digitised contours on Survey of India supplied topographic map. The corresponding geo-referenced IRS-1D FCC satellite imagery has been used for image draping.

The images in Figs 6 and 7 show the height map (DEM) of Grand Canyon area and corresponding false colour composite satellite imagery respectively. Figures 8(a) and 8(b) show the overlay of polyline vector data on top of 3-D level of detail wireframe display of terrain mesh geometry and corresponding phototextured terrain view respectively. Without multiresolution modelling of the polyline vector data, the visual artifacts are visible in the vector data display. Figures 9(a) and 9(b) show the views obtained after the geometry-based mapping of polyline vector data over multiresolution 3-D terrain as discussed in Section 5.2. The visual appearance of the displayed vector data is now much improved.

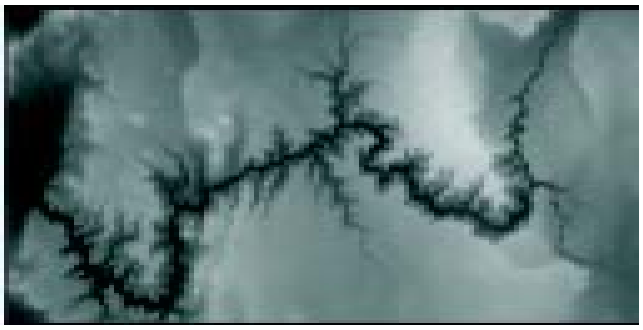


Figure 6. DEM (height map) of Grand Canyon area (size 4K*2K).

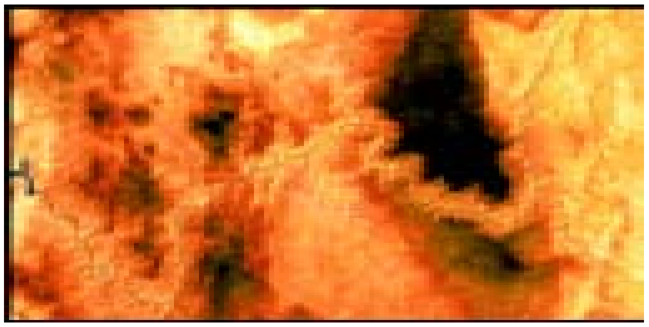
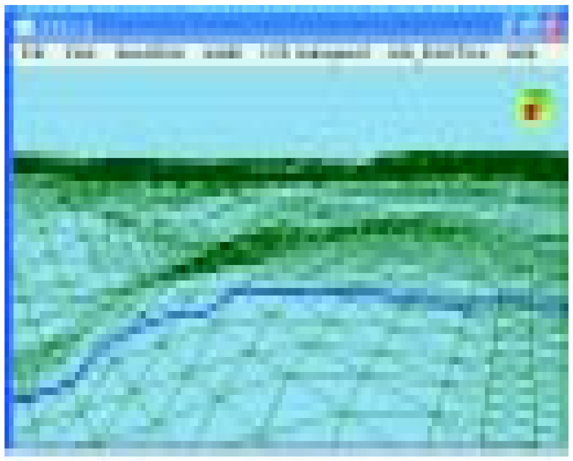


Figure 7. FCC satellite imagery of Grand Canyon area (size 4K*2K).


(a)

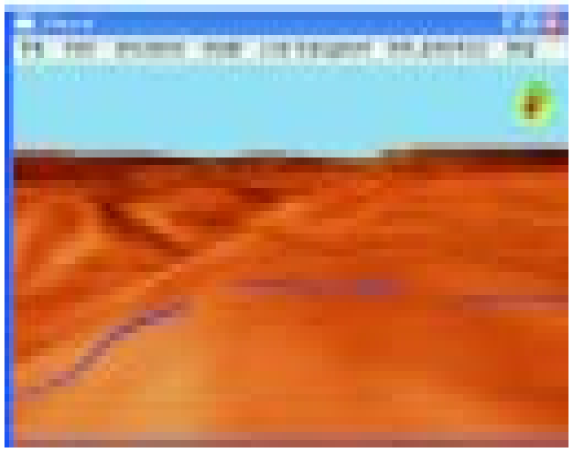
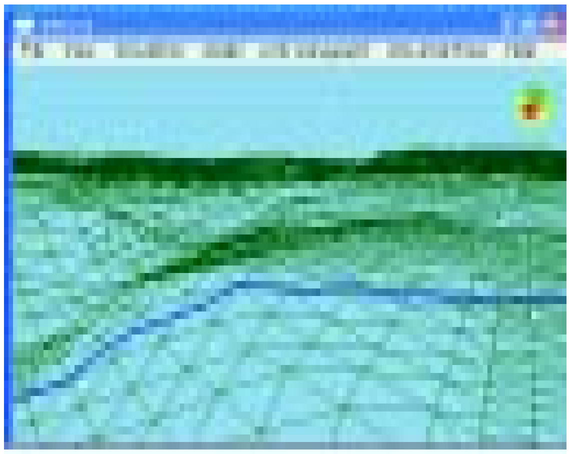
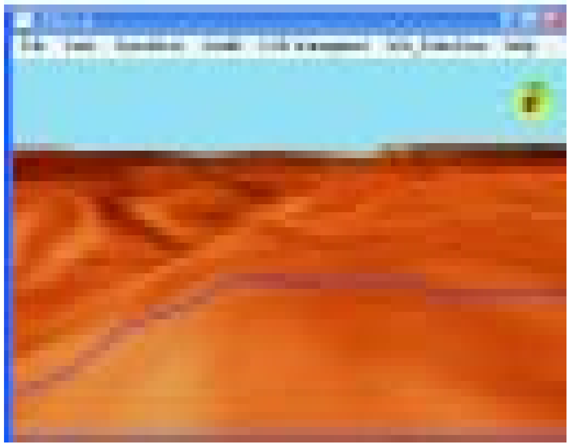

(b)

Figure 8. Display of vector data over LOD 3-D terrain (without multiresolution modelling).



(a)

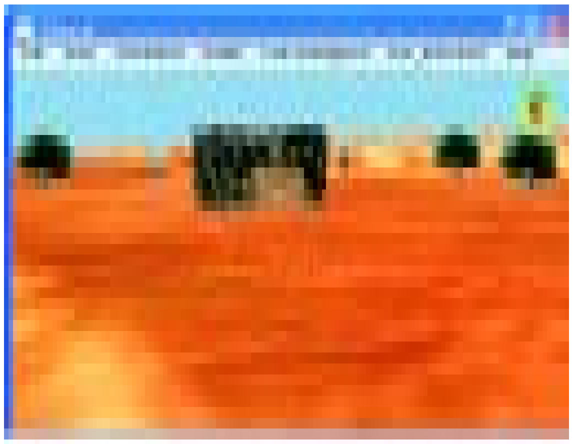


(b)

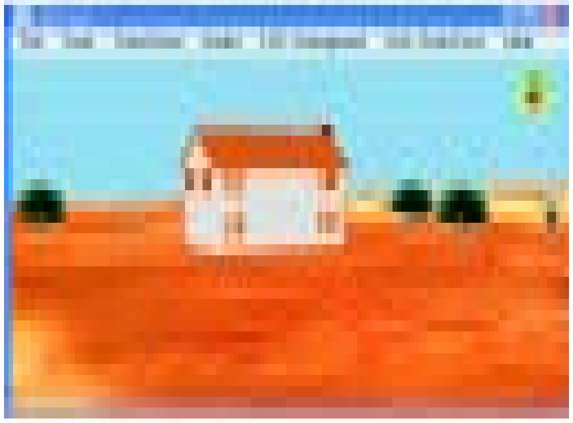
Figure 9. Display of vector data over LOD 3-D terrain (with multiresolution modelling).

Figure 10 (a) shows a view with point-location-based simple objects (OpenGL and billboard-drawn objects). A complex object (a building designed using AutoCAD) is shown over the terrain in Fig. 10 (b). Figure 10(c) shows the effect of fog and multitexturing to increase photorealism in the scene.

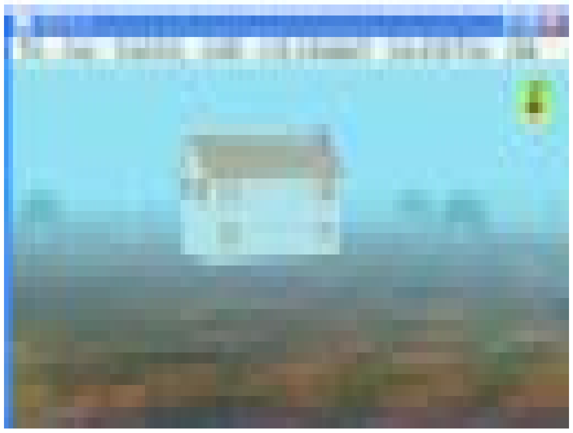
The performance of the software has been evaluated on a Pentium IV 2.4 GHz computer with 512MB RAM and Intel 82865G onboard Graphics Controller on 865GL motherboard. The performance of the algorithm on raster data is independent of size of terrain data as with the tiles indexing scheme, the algorithm only keeps nine tiles active in the main memory. The organisation of the terrain data in tiles of defined size is required to be done only once on the same data set. For raster data, the number of frames rendered per second mainly depends on the complexity of the terrain (roughness) under the view-frustum and the user-defined image quality metric (τ). Table 1 shows performance analysis of the adaptive LOD algorithm for 3-D visualisation of the raster data set.



(a)



(b)



(c)

Figure 10. Display of point-location-based objects: (a) with simple objects, (b) with complex objects, and (c) with special effects.

Table 1. Performance analysis of the adaptive LOD algorithm (without objects and vector data)

Terrain rendering without objects	Avg. no. of triangles	Avg. frames per second
Full resolution (considering 9 tiles only)	1327104.00	1.72
View-frustum culled surface	268120.70	7.18
Adaptive LOD algorithm (τ=4)		
1. Using triangle list	20368.45	57.23
2. Using triangle fan	20368.44	74.11
3. Using triangle strip (without using indexed vertex array)	23655.06	114.33
4. Using triangle strip (with using indexed vertex array)	23733.79	130.79

The results of testing the same adaptive LOD algorithm using triangle strip (with indexed vertex array) in conjunction with object management algorithm are shown in Table 2. Vector overlay has been performed using the proposed geometry-based mapping of polyline vector data over the 3-D LOD mesh. The polyline vector display algorithm maps the vector data consistently and exactly to the current of terrain geometry, and thus minimises the rendering artifacts. The software provides the facility to the user to populate the terrain with various types of point-location based objects and to visualize these in 3-D. To increase photorealism in the scene, various special effects such as fog, multitexturing etc. have been included. As a next step to further improve the rendering performance and quality of visualization, we are currently investigating rendering using state-of-the-art programmable GPU cards through vertex and fragment programs. Complex 3D objects such as buildings and trees with large number of polygons, severely affect the rendering performance. Discrete multiresolution representation of these objects and their run-time selection may further increase the rendering speed.

Table 2. Performance analysis of the adaptive LOD algorithm (with objects and vector data)

Terrain rendering with objects	Avg. frames per second
1. Using total 758 objects (Opengl: 39, Billboards: 719)	117.17
2. Using total 763 objects (Complex: 5, Opengl: 39, Billboards: 719)	111.32
3. Using multiresolution vector polyline (total 573 points) and 758 objects (OpenGL: 39, Billboards: 719)	85.16

8. CONCLUSIONS AND FUTURE WORK

The proposed methodology and algorithm produce real-time frame rates during terrain navigation showing 3-D phototextured views of the surface on large terrain data sets. User can make trade-off between image quality and rendering time by specifying quality metric τ (maximum error permissible in terms of number of screen pixels) for terrain raster data. Tile-based approach is used to organise terrain raster and vector data, and dynamic paging scheme is used for out-of-core management of data between secondary storage and main memory. This allows smooth interactive terrain visualisation on desktop PCs even with massive terrain data sets. In contrast to conventional algorithms, in the present algorithm the LOD computations are not performed on a per-triangle basis, but rather on larger chunks of data. Hence, the amount of work the CPU must perform is greatly reduced. The problem of limited memory bandwidth is handled by supplying the geometry to the graphic card in the form of long triangle strips.

ACKNOWLEDGEMENTS

The research reported here has been partially supported by the Ministry of Human Resource Development (MHRD), Govt. of India under contract F.26-4/2002.TS.V (R&D Scheme). The authors express their sincere gratitude to Dr M.D. Tiwari, Director, Indian Institute of Infomation Technology, Allahabad, for his encouragement and support throughout the course of this study.

REFERENCES

1. Akenine-Moller, T. & Haines, E. Real-time rendering, Ed. 2. A.K. Peters, 2002.
2. Anupam, Delaunay. Triangulation-based surface modelling and three-dimensional visualisation of landforms. *IETE Tech. Rev.*, 1998, **15**(6), pp. 425-33.
3. Anupam. Application of three-dimensional computer graphics in terrain visualisation. *In* Computer Science Section of 84th Indian Science Congress Symposium, 3-8 January 1997, Delhi University, Delhi.
4. Luebke, D., *et al*. Level of detail for 3-D graphics. Morgan Kaufmann Pub, 2003.
5. Cohen, J., *et al.* Appearance-preserving simplification. *In* Proceedings of SIGGRAPH. 1998. pp. 59-66.
6. Chen, B., *et al*. LOD-sprite technique for accelerated terrain rendering. *In* Proceedings of IEEE Visualisation, 1999. pp. 291-98.
7. Lindstrom, P., *et al*. Real-time continuous level of detail rendering of height fields. *In* Proceedings of ACM SIGGRAPH, August 1996, pp. 109-18.
8. Duchaineau, M., *et al*. ROAMing terrain: real-time optimally adapting meshes. *In* Proceedings of IEEE Visualisation, 1997. pp. 81-88.
9. Pajarola R. Large scale terrain visualisation using the restricted quadtree triangulation. *In* Proceedings of IEEE Visualisation, 1998. pp. 19-26.
10. Lindstrom, P. & Pascucci, V. Terrain simplification simplified: A general framework for view-dependent out-of-core visualisation. *IEEE Trans. V&CG*, 2002, **8**(3), 239-54.
11. Puppo, E. Variable resolution terrain surfaces. *In* Proceedings of 8th Canadian Conference on Computational Geometry, 1996. pp. 202-10.
12. Xia, J.C. & Varshney, A. Dynamic view-dependent simplification for polygonal models. *In* Proceedings of IEEE Visualisation, 1996. pp. 327-34.
13. Cignoni, P., *et al*. Representation and visualisation of terrain surfaces at variable resolution. *The Visual Computer*, 1997, **13**(5), 199-217.
14. Hoppe, H. Smooth view-dependent level-of-detail control and its application to terrain rendering. *In* Proceedings of IEEE Visualisation, 1998. pp. 35-42.
15. Agrawal, Anupam, *et al*. Dynamic multiresolution level-of-detail mesh simplification for real-time rendering of large digital terrain models. *In* Proceedings of IEEE *INDICON*-2004, 20-22 December 2004, IIT Kharagpur. pp. 278-82.
16. Agrawal, Anupam, *et al*. TREND: Adaptive real-time view-dependent level-of-detail-based terrain rendering. *In* Proceedings of IT++: The Next Generation. 39h Annual National Convention of CSI, 1-4 December 2004, Mumbai. pp. 146-57.
17. Agrawal, Anupam, *et al*. An approach to improve rendering performance of large multiresolution phototextured terrain models using efficient triangle strip generation. *In* IEEE IGARSS-2005 held in Seoul, Korea, during July 25-29, 2005. pp. 4984-4987.
18. Mason, Woo, *et al*. OpenGL programming guide: The official guide to learning OpenGL, Ver 1.2, Ed. 3. Addison-Wesley, 2000.
19. Szenberg, Flávio, *et al*. An algorithm for the visualisation of a terrain with objects. http://www.tecgraf.puc-rio.br/~szenberg/artigo_sib97/artigo_sib97.html.
20. Douglass, D., *et al*. Real-time visualisation of scalably large collections of heterogeneous objects. *In* Proceedings of IEEE Visualisation, 1999. pp. 437-40.
21. Zachary, W., *et al*. Rendering vector data over global, multiresolution 3-D terrain. *In* Proceedings of Joint EUROGRAPHICS- IEEE TCV Symposium on Visualisation, 2003. pp. 213-22.
22. Kersting, O. & Dollner, Jurgen. Interactive 3-D visualisation of vector data in GIS. *In* Proceedings of the 10th ACM International Symposium on

Advances in Geographic Information Systems, November 2002. pp. 107-12.

23. Anupam, *et al*. Representation techniques for topographic surfaces: An overview and an efficient TIN algorithm. *In* Proceedings of the International Conference on Geoinformatics for Natural Resource Assessment, Monitoring and Management, 9-11 March 1999, IIRS, Dehradun, India. pp. 490-97.

24. Agrawal, Anupam *et al*. Additive wavelet decomposition-based resolution merge for remote sensing images. *In* Proceedings of International Symposium on Information Technology: Emerging Trends, 19-21 September 2003, Indian Institute of Information Technology, Allahabad, India, 2003. pp. 156-67.

25. Agrawal, Anupam, *et al*. Geometry-based mapping and rendering of vector data over level-of-detail phototextured terrain models. *In* Proceedings of WSCG-2006, 14th International Conference in Central Europe on Computer Graphics, Visualisation and Computer Vision, Plzen, Czech republic, Jan 30-Feb 03, 2006 pp.1-8

Contributors

Mr Anupam Agrawal received his MS(Computer Science) from the J.K. Institute of Applied Physics and Technology, University of Allahabad in 1988 and MTech (Computer Sc & Engg) from the IIT Madras, Chennai in 1995. He is presently working as Assistant Professor at the Indian Institute of Information Technology, Allahabad. Earlier, he was working as Scientist D at the DEAL, Dehradun. His research interests include: real-time 3-D graphics, computer vision, artificial intelligence and soft computing, data mining, GIS and remote sensing image processing. He has more than 30 research papers to his credit. He is a member of the IEEE Computer Society, USA, CSI, Mumbai, IETE, New Delhi, IE(I), Kolkata, and ISTE, New Delhi.

Mr R.C. Joshi received his ME and PhD (Electronics and Computer Engg) from the University of Roorkee, [now Indian Institute of Technology, Roorkee (IITR)] in 1970 and 1980, respectively. He joined E&CE Dept., IITR as Lecturer in 1970. His research interests include: Computer graphics and image processing, parallel and distributed processing, artificial intelligence, databases and bioinformatics.

Mr M. Radhakrishna received his MSc (Nuclear Physics) from the Andhra University in 1962. Currently, he is Advisor and Professor at the Indian Institute of Information Technology, Allahabad. He is also Technology Advisor to Aptech, Mumbai. His research interests include: Artificial intelligence, automation, cognitive sciences, computer graphics and image processing, modelling and simulation, computer networks. He has published more than 60 papers.