Use of 3D on the Web: Fundamental Issues and Emerging Technologies

Dinesh Shikhare
National Centre for Software Technology, Juhu, Mumbai.
November 1999.

Abstract

The 3D rendering capabilities of the desktop have improved immensely in the recent years and on the other hand, the availability and the use of WWW have also increased. This has created a large number of opportunities for development and deployment of 3D content for the Internet. This report surveys the following aspects the subject:

1. Classes of online 3D Applications that can be deployed,
2. Content creation issues,
3. Content distribution of issues,
4. Rendering and interaction techniques, and
5. Security and copyright protection technologies.

We focus on the data management and algorithmic aspects of these issues and cite the representative work that has been done already and also the current research issues.

1 Introduction

Many applications that essentially deal with 3D data need effective visualization and interaction systems for access and manipulation. The data is sometimes generated in one site and is accessed by many people who are geographically separated. Hence it is most convenient to share that data across the machines in an intranet or across the Internet. Below, we present only some of the representative scenarios where 3D data is shared:

1. A car company may wish to give a preview of some of its new models to the prospective buyers by placing the 3D digital models of their products on the web. The interested people can then view the 3D model from various viewpoints inside and outside the car model. It is also possible to interactively choose various colours for the car and also try different accessories and uphostery for the model. Extending this idea further, the user can even order the car with the various options he/she has selected.

2. Architectural models or CAD models may be worked upon by multiple architects or engineers who visualise the shared models, interact with each other by accessing the model. Here, various parts of the same model may be stored in various locations, but visualised in an integrated fashion. The degree of integration may also be controlled by individuals inspecting the models.

3. Digital documentation of heritage sites can been placed on the web along with the 3D models of the monuments and statues for interactive viewing.

4. Medical data such MRI and CT-scan volumetric images can be made available to experts residing at remote locations for their opinions and diagnosis. Such 3D data must be rendered at remote location for interactive inspection from various viewpoints.

5. Educational material that requires 3D visualization for improved understanding of the subject being studied can be presented with appropriate 3D animations of models.

In all the above cases, the fundamental technical issues are acquition of data, formats for storage and sharing of information, transmission of data, visualization, access control and copyright protection. The past few years have seen an exponential growth in the complexity of 3D data being used on desktop class machines and also in sophistication of the modeling tools for the creation and editing of the content. This is an output of persistent work of researcher and developers in the area of computer graphics. The technical issues in efficiently and effectively using 3D on the web posed many new and challenging problems. Consequently, a large pool of new computational techniques, data-structures and algorithms has emerged. This work continues to grow at a rapid pace. In this report we present the state of the art in deployment and the use of 3D models on the web.

1.1 How this report is organised.

Rest of the report is organised as follows. Section 2 defines the basic elements and the types of 3D data and also describes the various prevalent processes used for the acquisition of the models. Section 3 discusses VRML97 file format which is currently the standard for exchange of virtual worlds across the web. Section 4 surveys the simplification techniques use to reduce redundant data in the 3D models. Section 5 describes the issues invoved in compression of geometric models represented as triangle meshes. Section 6 motivates and describes the concept of progressive transmission of large models. Section 7 outlines the issues in handling copyright protection of polygon mesh models.

2 Acquision of 3D data

We first identify the various kinds of 3D data that can be of interest for visualization over the web and then briefly touch upon the tools used to acquire the data.

2.1 Data Elements

The different forms of 3D data currently used for exchange over the web are:

1. **Polygon mesh:** This is most general form of modeling boundary representation of objects. The popular choice of polygons in such representations is triangles. The

meshes are represented in terms of a shared list of vertices, a list of triangles – with each triangle defined with a triple of indices into the shared vertex list. The vertices are defined minimally by their (x, y, z) coordinates. Additionally, the vertices may have other attributes like normals, colour value and texture coordinates. The meshes also have material properties defining their light reflectance behaviour when illuminated. The textures to be mapped are represented as images in some format like JPEG, GIF, etc.

2. **Smooth surface definition:** B-spline curves and surfaces provide a mechanism of defining objects with smooth features. These modeling schemes provide a compact representation of complex models that would require large number of polygons to capture the smoothness. This representation is the most popular for modeling CAD data of engineering models.

3. **Voxel models:** Voxel-based modeling scheme is another generic scheme for modeling 3D objects. This data consists of three dimensional array of some scalar or vector values. These values are interpreted by visual rendering software in different ways depending on the application.

Of these different representations, triangle mesh based representation is the most popular. This popularity is partly due to the hardware for rendering 3D graphics is mostly a triangle rendering engine.

2.2 Modes of Acquision

Various application areas generate 3D data via different processes and different tools. In fact, any modeling tool that generates 3D models can be used. This provides a large spectrum of data acquisition tools ranging from the traditional CAD packages to 3D scanning systems to motion tracking systems to MRI scanners and so on.

CAD packages: The most traditional technique that continues to be used for creating 3D content is to use a CAD package. These packages have numerous specialised tools for creating various aspects of models: geometry, visual attributes and temporal behaviours such as animation and morphing. Geometry is modelled by creating free-form shapes by extrusion, sweep, fleshing about a skeleton, editing of control points and so on. The visual attributes such as colour and material properties are determined at a granularity of a mesh or a surface. Other attributes such as vertex normals and texture coordinates are computed automatically by the package. These are generated by mapping the given object to the closest cononical shape like sphere, cylinder, cube, tetrahedron, etc. Examples of some popular packages are 3D Studio MAX, Alias Wavefront and Maya.

3D scanning: Sometimes modellers create clay models of the objects. At other times, digital models of existing physical objects are needed. These objects are then scanned by using 3D scanners. These scanners sample a large number of points on the model from all directions and construct a dense 3D model. Some scanners also scan the colour values along of the coordinates of the points scanned. The software that drives the scanner

connects the points scanned in a regular pattern into a triangle mesh or a quadrilateral mesh.

3D digitizers: Before the automatic scanners appeared in the market, manual digitizers were in use. The model of usage of these digitizers is as follows: (a) the user must mark points on the clay model in a regular pattern of a quadrilateral mesh, (b) the digitizer is then calibrated to obtain a frame of reference for digitising the points, (c) the stylus of the digitizer is then placed on each point marked on the model in a sequence such that the measured points can then be used to create a mesh in a trivial manner. These scenners are inexpensive and usable, and hence continue to be used in the animation industry. The number of points that can be manually digitized is limited. To refine the resolution of distribution of points on the model, mathematical techniques such as Coon's patch fitting are used. To modify the shape definition the users can edit the scanned points and other parameters at these points like tangents and normals. The surface fitting is modified to reflect the changes in the local parameters.

MRI and CT scanning: Magnetic resonance imaging and computed tomograph based scanners generate a large amount of voxel data of 3D objects. These data consist of a three dimensional array of scalar. Most common sources of such data are medical imaging centres and industrial non-destructive visual testing machines.

A few other specialised tools exist for the creation of 3D data. These are volume discretization tools (also called grid generation packages), CFD analysis packages, etc.

2.3 Format conversion and healing of models

The models are acquired using various processes and stored in forms the are different from those required for the end use. The data needs to be converted to a suitable file format and the model also needs to be transformed so that it is in a usable state. Here we list the issues related to the file formats and transformation of models for the end application.

Issues in conversion of file formats:

1. Information captured in the source file format may not be sufficient to construct the details necessary to build the model in the other file format. In such cases, it may be necessary to infer this information.

2. The conversion between file formats may be a lossy process where the target file format does not need the excess data in the source file format.

3. The data types used in the source file format and the destination file format may not be the same. Hence some approximation of model may have to be accomodated.

Other issues adapting the captured source data for an end application are:

1. The source data may have a very high resolution which may not be suitable for the end application and needs to be reduced. While reducing the data, the information being communicated (from the viewpoint of the application) must be minimally affected. This can be achieved by adaptive sampling rather than trivial uniform sampling.

2. The source data sometimes has errors incurred during recording or modeling of objects, either due to the instrumentation errors or human errors. These errors could be missing polygons, gaps in the geometric description, incorect textures mapped onto the geometry. These artifacts must be corrected before using the models for the application. Some of the errors can be healed automatically, where as some need to be manually fixed by loading the model in some editor.

3. To achieve faster transmission and progressive loading of the models, the data must be transformed into a suitable format such that the information is differently represented. Many model compression and progressive transmission schemes have been proposed by researchers. In this report we survey some such representative techniques.

3 File Formats

In this section, we shall largely concentrate on the polygon mesh representation of surfaces for modeling 3D worlds or scenes. The basic elements of 3D data to be modelled are:

1. Scene definition: A moderate sized scene usually has many meshes and a few light sources defining the illumination conditions in the scene.
   (a) List of meshes
   (b) List of lights
2. Triangle mesh definition: Individual meshes in the scene form the largest part of a scene definition. The mesh definition provides the geometry and visual attributes.
   (a) Shared vertex list: This list is shared among the triangles in the mesh. Sharing the list saves the duplication of vertex details when a vertex is shared across multiple triangles.
      i. vertex coordinates,
      ii. vertex normals,
      iii. texture map coordinates, and
      iv. vertex colours.
   (b) Mesh topology: This defines the connectivity among the vertices by listing the indices into the shared vertex list.
      i. list of triangles, and/or
      ii. list of triangle-strips.
   (c) Texture images: The texture images are parameterised with a rectangular domain and mapping of certain areas on the images onto the triangles is done by specifying the texture coordinates within the parametric range of the map. The texture parameters exceeding the parametric domain are wrapped back into the image.

(d) Material properties: Modeling of the reflective behaviour of material in the given illumination conditions is specified in the form of ambient reflectance, diffuse reflectance, specular reflectance and emmissive properties.

3. Lights: The lights are defined with various components such as diffuse illumination, specular illumination, and other properties such as point source, spotlight source, etc. The lights are commonly modelled with red, green and blue components, which is a great simplification of the more elaborate spectrum model of light. This approximation, however, has been found to be adequate for most of the applications requiring consumption only by human visual system.

To encode these elements of information on storage and transmission media, numerous formats have been used. The stadard formats for data exchange over the Internet has been VRML1.0 and VRML97. However, there are also many proprietory file formats in use for the modeling packages like 3DStudio MAX, Alias Wavefront, and so on. These file formats have provisions for storing information that is specific to the algorithms and features of the packages. Hence they are extremely rich in the information they capture and store. In this section, we only briefly discuss the features of the VRML97 file format, since it is the current standard of World Wide Web Consortium and is widely accepted.

3.1 VRML97

The Virtual Reality Modeling Language (VRML) is a file format for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia. VRML may be used in a variety of application areas such as engineering and scientific visualization, multimedia presentations, entertainment and educational titles, web pages, and shared virtual worlds.

VRML has been designed to fulfill the following requirements:

1. *Authorability:* Enable the development of computer programs capable of creating, editing, and maintaining VRML files, as well as automatic translation programs for converting other commonly used 3D file formats into VRML files.

2. *Composability:* Provide the ability to use and combine dynamic 3D objects within a VRML world and thus allow re-usability.

3. *Extensibility:* Provide the ability to add new object types not explicitly defined in VRML.

4. *Be capable of implementation:* Capable of implementation on a wide range of systems.

5. *Performance:* Emphasize scalable, interactive performance on a wide variety of computing platforms.

6. *Scalability:* Enable arbitrarily large dynamic 3D worlds.

VRML is capable of representing static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. VRML browsers, as well as authoring tools for the creation of VRML files, are widely available for many different platforms.

VRML supports an extensibility model that allows new dynamic 3D objects to be defined allowing application communities to develop interoperable extensions to the base standard. There are mappings between VRML objects and commonly used 3D application programmer interface (API) features.

A VRML file consists of the following major functional components: the header, the scene graph, the prototypes, and event routing. The contents of this file are processed for presentation and interaction by a program known as a browser.

The scene graph contains nodes which describe objects and their properties. It contains hierarchically grouped geometry to provide an audio-visual representation of objects, as well as nodes that participate in the event generation and routing mechanism. Prototypes allow the set of VRML node types to be extended by the user. Prototype definitions can be included in the file in which they are used or defined externally. Prototypes may be defined in terms of other VRML nodes or may be defined using a browser-specific extension mechanism. While ISO/IEC 14772 has a standard format for identifying such extensions, their implementation is browser-dependent.

Some VRML nodes generate events in response to environmental changes or user interaction. Event routing gives authors a mechanism, separate from the scene graph hierarchy, through which these events can be propagated to effect changes in other nodes. Once generated, events are sent to their routed destinations in time order and processed by the receiving node. This processing can change the state of the node, generate additional events, or change the structure of the scene graph.

Script nodes allow arbitrary, author-defined event processing. An event received by a Script node causes the execution of a function within a script which has the ability to send events through the normal event routing mechanism, or bypass this mechanism and send events directly to any node to which the Script node has a reference. Scripts can also dynamically add or delete routes and thereby changing the event-routing topology.

The ideal event model processes all events instantaneously in the order that they are generated. A timestamp serves two purposes. First, it is a conceptual device used to describe the chronological flow of the event mechanism. It ensures that deterministic results can be achieved by real-world implementations that address processing delays and asynchronous interaction with external devices. Second, timestamps are also made available to Script nodes to allow events to be processed based on the order of user actions or the elapsed time between events.

The interpretation, execution, and presentation of VRML files will typically be undertaken by a mechanism known as a browser, which displays the shapes and sounds in the scene graph. This presentation is known as a virtual world and is navigated in the browser by a human or mechanical entity, known as a user. The world is displayed as if experienced from a particular location; that position and orientation in the world is known as the viewer. The browser provides navigation paradigms (such as walking or flying) that

enable the user to move the viewer through the virtual world.

In addition to navigation, the browser provides a mechanism allowing the user to interact with the world through sensor nodes in the scene graph hierarchy. Sensors respond to user interaction with geometric objects in the world, the movement of the user through the world, or the passage of time.

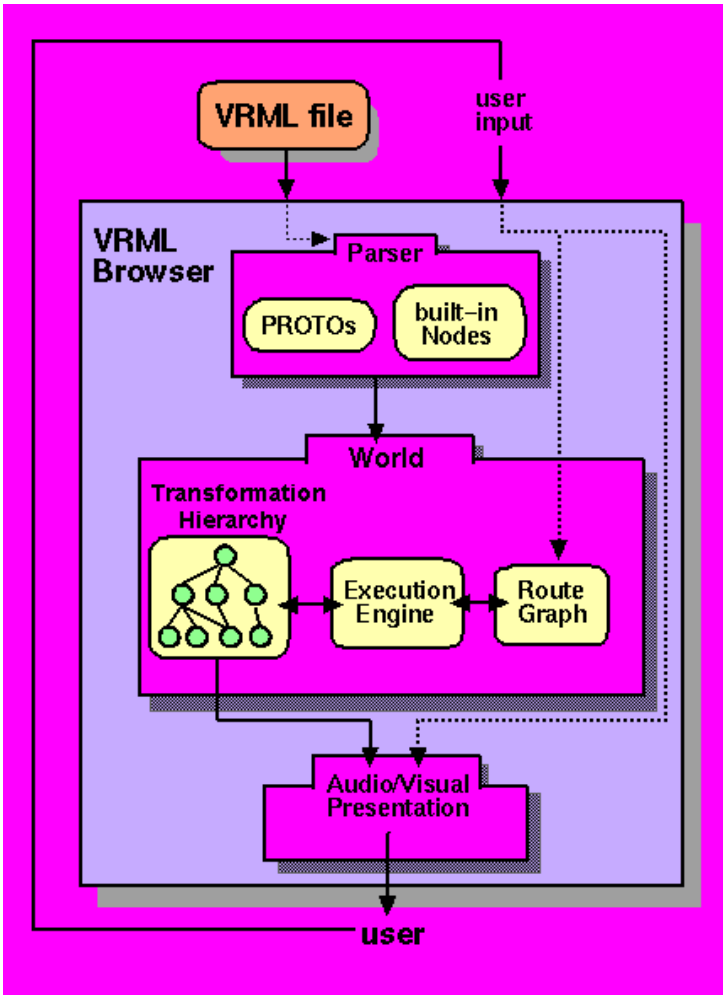

Figure 1: Copnceptual model of VRML browser

Figure 1 illustrates a conceptual model of a VRML browser. The browser is portrayed as a presentation application that accepts user input in the forms of file selection (explicit and implicit) and user interface gestures (e.g., manipulation and navigation using an input device). The three main components of the browser are: Parser, Scene Graph, and Audio/Visual Presentation. The Parser component reads the VRML file and creates the Scene Graph. The Scene Graph component consists of the Transformation Hierarchy (the nodes) and the Route Graph. The Scene Graph also includes the Execution Engine that processes events, reads and edits the Route Graph, and makes changes to the Transform Hierarchy (nodes). User input generally affects sensors and navigation, and thus is wired

to the Route Graph component (sensors) and the Audio/Visual Presentation component (navigation). The Audio/Visual Presentation component performs the graphics and audio rendering of the Transform Hierarchy that feeds back to the user.

3.2 Current research issues in file formats

The new file formats proposed have the following main concerns of enhancements: (a) static reduction of the redundant data to simplify the data, (b) compression of geometric scenes for compact representation, (c) progressive transmission of 3D scenes – such that the user experience of the model can start even before the entire model has arrived for the browser to start interpreting it, and (d) protection of intellectual property rights – the painstakingly constructed 3D models must not be stolen and used thereby denying the creator of the commercial benefits he/she should get for the efforts.

Various researchers have proposed schemes for handling these issues of handling the 3D models over the web. In the following sections, we report survey some of the work in progress in these area.

4 Simplification of Geometry

The highest complexity of the models is not always needed since very often a model may occupy very small footprint in the screen, and rendering such less significant model with all its details does not have any better visual effect than rendering a simplified, low resolution representation. Figure 2 shows an example. As the rendering rates are directly dependent on the combinatoric complexity of the models being handled, it is useful to have simpler models of the complex when the models are not required in their highest resolution.

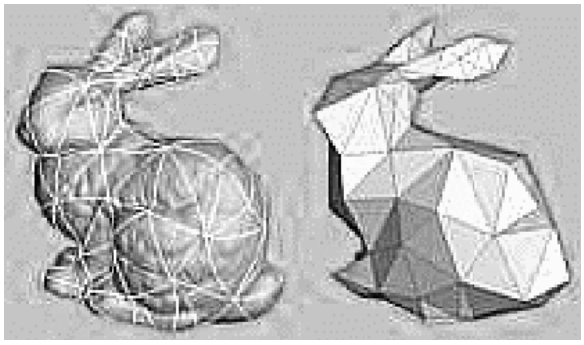

Figure 2: Simplified Mesh

Recent years have seen a lot of research being focussed on the problem of polygonal mesh simplification algorithms. Many algorithms have been reported in the literature and a few papers have surveyed [4] and compared the methods. In this report we aim to study the methods and classify them on the basis of the following:

- Based on the requirements of the applications using meshes at multiple resolutions.

- Based on the algorithmic strategies employed in the techniques

We also study some special rendering techniques utilized for efficient rendering of the meshes available at multiple resolutions.

4.1 Classification

Based on the requirements of the applications, certain characteristics of the original meshes representing the objects may or may not be retained. The most important of such characteristics is the topological features such as genus, protuberance and connectivity between vertices, edges and facets. The applications that demand real time frame-rates but can tolerate small visual artifacts can use mesh simplification algorithms that modify certain topological features. The other class of methods simplifies the combinatoric complexity of the meshes (in terms of the number of triangles) while retaining topological features and certain geometric features like sharp edges.

The order of simplification in terms of percentage of number of triangles decimated is much higher in case of the simplification algorithms that sacrifise topological features while retaining the visual similarity of the objects.

Various algorithmic strategies are employed to simplify meshes. These can be broadly classified as: (a) vertex cluster collapsing, (b) edge collapsing, and (c) triangle collapsing strategies. These strategies are employed to get both topology preserving algorithms as well as topology simplifying algorithms.

Visualization of multi-resolution meshes and selective simplification and refinement of the meshes is important since the level of detail of the mesh to be used must be decided by the application that wishes to render the objects. Although different multi-resolution meshes have different ways of parameterising the level of detail, the determination of the parameter value at a given instant is still a non-trivial and interesting issue.

In the following sections, we see mesh simplification techniques of the different classes mentioned above and examine the suitability of each algorithm for specific tasks. Most of the methods discussed below assume that the source meshes are triangle meshes. This assumption does not sacrifise generality of polygon meshes, since the polygons having more than three sides can be decomposed into triangles.

4.2 Topology Preseving Mesh Simplification

To maintain topological features of the original mesh across a hierarchy of simplified low resolution meshes representing the original object, most approaches have proposed a set of local operators. These operators incrementally increase or decrease the detail in the mesh. Schroeder et al [12] were among the first to report a triangle mesh decimation strategy where “visually insignificant” vertices are removed. Removal of a vertex and the triangles conected to it creates a hole in the mesh, which is triangulated and patched.

Hugues Hoppe [5] presented a technique called as Progressive Meshes that give a facility of generating smooth spectrum of levels of detail using two operators: edge-collapse

(e-col) and vertex-split (v-split). (see Figure 3.) The e-col operator collapses the given edge and also collapses the two triangles sharing the edge. The edge gets collapsed into a vertex. The connected neighbourhood is readjusted to acommodate the change. Each edge collapse operation creates a record of the operation "pushed" onto a stack. This record consists of the description of the original edge and information about the triangles that were connected to the edge before collapsing. Note that each e-col operation removed one vertex and at most two triangles from the mesh. This hold true provided that the mesh represents a manifold and each non-boundary edge has only two triangles connected to it. The v-split operation "pops" the record off the stack and reconstructs the local connectivity and geometric detail. Although the term stack is used for storing and retrieving the records for reconstructing the detail of the simplified mesh, it is often possible to retrieve refinement detail for the mesh.

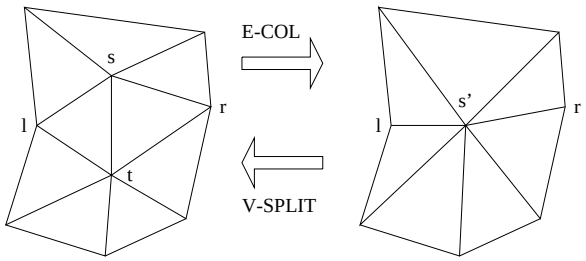

Figure 3: Progressive meshes

While these methods retain the topological features of the original mesh in the simplified low resolution versions, the order of simplification is limited. A complex object having many topological features cannot be simplified below a certain level of detail. However, in some critical applications, the topological details are required even at the lowest level of detail.

4.3 Topology Simplifying Techniques

Very often the footprint of a 3D mesh is too small to capture the detail modeled in the mesh. Many of the triangles have a subpixel projected size. At times even major topological features do not show up even if the object is drawn in all its detail. For many visual applications such as virtual walkthroughs of architectural models, the visually insignificant may be rendered as approximate representation. Topology simplifying algorithms for mesh simplification achieve higher order of reduction of the meshes. In this section, we briefly describe three representative techniques.

Jovan Popovic et al [9] proposed progressive simplicial complexes which is a generalization of the progressive meshes reported in [5]. This work extends the progressive meshes to work on meshes representing non-manifold sufaces. This enables them to handle 1-D curves, 2-D triangles, 3-D tetrahedra and even higher dimentional simplices. Also, rendering of these multi-resolution meshes is done by some representative shapes for spheres, cylinders, etc. when the collapsed geometry occupies a very small portion on the screen.

David Luebke et al [7] present vertex cluster collapsing strategy to simplify polygonal

scenes. This technique creates a *tight octree* of the vertices in the surface meshes describing the scene. The part of the scene that is visually significant is retained in its full detail and the triangles in the less significant regions are collapsed, reducing the number of triangles to be sent to the rendering engine for display. Collapsing of geometry within a region is done by selecting vertices in a given cell of the octree. The geomtry is un-collapsed when it becomes visally significant while collapsing the geometry that may become insignicant. To be able to un-collapse the geometry, the technique keeps hierarchies of collapsed geometries. Each level in the hierarchy represents a level of detail. Depending on the level of detail required, the appropriate level in the hierarchy is selected and displayed. Note that no special care is take to retain the topological features of the meshes while collapsing the geometry. Geometry is collapsed based on the proximity of vertices. These nearby vertices may belong to different meshes. The only issue that matters to this technique is visal similarity of the scene to the original one when drawn at different resolutions.

Jihad El-Sana et al [3] report their work on topology simplification for polygonal virtual environenments. This paper describes extensive geometry analysis techniques for identification of topological features like edges, corners, genus, protuberances and sockets. The topological features are based on geometric discontinuity of normals across neighbouring triangles. Significance of topological features is rated on the basis of geometric discontinuity, surface area of a protuberance or a socket, length of an edge, and so on. To simplify the meshes, their technique provides a set of operators. An operator is available for simplification of each of the topological features. The operators have two phases: removal of the feature and retriangulation of the region.

4.4 View-dependent simplification of meshes

In this section we shall look at some representative techniques determining at run-time which part of the geometry in the surface mesh should be simplified. Simplification and refinement criteria can be classified at top level as: view frustum based, surface orientation based and screen-space geometric error based.

The geometry that falls inside the view frustum of the virtual camera capturing the scene forms a good candidate for high level of detail, and the geometry falling outside the frustum may be simplified. To accelerate the selection of geometries that lie inside the view frustum, most implementations carry out bounding box inclusion test for each of the meshes in the scene. Some techniques use octree-based selection of parts of the geometries for simplification.

Back facing polygons of the meshes are not visible to the viewer. This fact can be used determine the part of the geometry to reduce the detail of the surfaces after they have been passed by the view frustum test. This selection method requires the original meshes to have a consistent and correct orientation. Also, the simplification algorithm must maintain the consistency of orientation of the simplified meshes.

The third refinement criterion is screen-space geometric error. The goal of this criterion is to adapt the mesh refinement such that the distance between the approximate mesh and the original mesh, when projected on the screen, is everywhere less than a screen space

tolerance ϵ. Hoppe defines a measure for screen space distance in [6].

Note that these criteria are not tricial to compute efficiently. To compute them fast, the implementations often use approximations of the criteria to quick determine the level of detail of the model to be used.

View dependent progressive meshes [6] describes all the above criteria for selecting the level of detail for progressive meshes. For efficiency, he pre-computes the hierarchy by using e-col operators. At run time, the parts of the meshes to be simplified are selected and corresponding v-split operators are performed to refine the required regions.

Luebke's work on view-dependent simplification of arbitrary polygonal environments [7] uses view-frusum based selection of cells of a packed octree which is used to classify the vertices and meshes in the scene. The scene is organized as a hierarchy of levels of detail which looks like pyramid. The base of the pyramid has the scene defined at the original level of detail, and the tip represents the coarsest appoximation. Depending on the viewing frustum, different parts of the scene is rendered at different levels of detail.

5 Geometry Compression

To reduce the storage and transmission time requirements, compression of data is the standard solution. The generic data compression techniques can yield compression ratios upto some levels when applied to geometry databases such as 3D scenes. However, by knowing the structure and form of the data representing such models, it is possible to arrive at special compression algorithms. Such algorithms take various lossy and non-lossy approaches to reduce the data required to obtain a compact representation for visual consumption.

Much of the work done in the area of geometry compression is based on clever encoding of the topological relationship between nodes in the meshes. These encodings minimize the repeated references to nodes, thereby achiving a compact description of topology. An interesting observation made in meshes representing manifolds is that, on an average, the number of triangles is twice the number of vertices and each vertex is referenced in 5 to 7 triangles. Hence, a lot of research has concentrated on aggressive attack on the problem of encoding of topological relationship between vertices.

Early examples of compact encoding of a mesh were seen in triangle rendering engines such as OpenGL, in the form of triangle-strips and triangle-fans. A lot of research has been carried out in generating maximal triangle-strip decomposition of given meshes, minimizing the repeatitions in the references to vertices. Deering [2] describes a more efficient "generalised triangle mesh" representation for further reducing the redundant referencing of vertex data. A set of four operators is defined to interpret a stream of vertex indices refering to the list of points. This leads to a very efficient encoding. This technique is now a part of Java3d's [8] compressed geometry node. Chow [1] extended the work of Deering by improved strategy of triangle strip traversal and encoding. Rossignac's work on Edgebreaker [11] algorithm achieves even greater compression by compact representation of topological relationship between vertices of a mesh.

Hugues Hoppe, in his work on Progressive Meshes [5], mentions the compression achieved

through the progressive representation of the meshes. The progressive addition of detail to the existing description of the mesh involves splitting of a vertex at a time by v-split operation, which adds a new point to the mesh. The additional encoding describes the left and right hand side triangles generated in the process of adding a new edge. The compactness of the representation comes from the fact that each vertex is stored and referenced only once in the progressive mesh data-structure.

Besides compactly encoding the topological relationships between vertices in the meshes, most of the work reported also uses lossy techniques for reducing the bits used for represeting vertex coordinates, normals and colour values. For example, Deering uses 3 x 16-bit representation for XYZ-values for points, 9-bit index into a global list of values for normals and 15-bit representation for colour values.

6 Progressive Transmission

While compression of the geometry representing the 3D models being shared over the net helps in reducing the transmission times, the experience of the 3D scene does not begin until the entire data has arrived and is uncompressed. The progressive transmission schemes for distributing 3D geometric scenes build in mechanisms that have the following characteristics:

1. Transmit the model such that the initial data elements represent the coarse model that can be used to render the currently viewed area or the coarse model of the entire scene.

2. The subsequent data elements are transmitted to refine the initial model on need-to-transmit basis.

3. At the receiving end, the browser begins rendering the 3D geometry and the attributes as soon as it receives an approximation of the model that is sufficient to start the experience of the 3d scene.

4. The subsequent details received are used to refine the model progressively.

The earliest work on progressive tranmission of 3D geometry was reported by Hoppe [5] through his paper on Progressive Meshes, which we have described earlier. However, his approach is largely effective on single meshes. The models having many meshes do not yield to the method effectively. Gabriel Taubin's [13] work towards Progressive Forest Split Compression algorithm provides another technique to achieve progressive representation of meshes. This algorithm views the mesh as a graph with vertices and edges corresponding to the geometric coordinates and the connectivity among them. The graph is split into subgraphs based on the geometric properties of the local regions. To move to a lower resolution of the mesh, the nearby subgraphs are merged and the information needed to reconstruct the geometry is stored.

The current efforts are concentrated on treating a scene as a whole for progressive transmission rather than dealing with one mesh in the scene at a time. The Multi-resolution

clustered meshes [14] presents a scheme for handling mesh clusters for progressive transmission. At the coarsest level of resolution, the 3D scene is represented by a set of vertices representing clusters of meshes and some connectivity. Subsequent details describe how these clusters should be transformed into meshes to progressively reconstruct the original scene.

7 Watermarking of 3D Geometry

Watermarking provides a mechanism for copyright protection of digital media by embedding information identifying the owner of the data. The bulk of the research on digital watermarking has focused on media as images, video, audio and text. Robust watermarks must be able to survive a variety of "attacks" including resizing, cropping and filtering. For resiliance to such attacks, recent watermarking schemes employ a "spread-specturm" approach – they transform the data into the frequency domain (e.g. using DCT) and perturb the coefficients of the perceptually most significant basis functions.

The field of *steganography* addresses the problem of hiding information within digital documents. The information, called the embedded object, is inserted into the original document, called the cover object, to produce a stego object. The term watermarking loosely refers to the use of steganography in the application areas of ownership assertion, authentication, content labeling, content protection and distribution channel tracing. It should be noted that no single watermarking scheme is suitable for all these applications, as some of the goals are incompatible.

For ownership claims, watermarks encode the fact that the document was created by the owner. To be effective, the watermark scheme must minimise the probability of *false-positive* results – incorrectly asserting that a document in watermarked when it is not. To decrease the probability of false-positive claims, the watermark is usually encoded in the document using vector coefficients. This vector is compared to that observed in the suspect document and the ownership claim is based on their statistical correlation.

The probability of *false-negative* results, that of failing to detect a watermarked document, is of lesser importance, and is more difficult to analyse, since it depends on the type and magnitude of attack. Such attacks may include inadvertant alterations like compression, blurring, sharpening, scaling, cropping, darkening and format conversion; but may also include malicious operation like intentional addition of noise, modification of low-order bits, or even insertion of other watermarks.

According to their resilience to attacks, watermarks can be *fragile* or *robust*. Fragile watermarks are used for authentication and for localization modifications. Much like digital signatures, their goal is to detect the slightest change to the document. In contrast, robust watermarks are designed to survive (remain detectable) through most attacks. While an attack if sufficient magnitude could erase the watermark, the hope is that an attacker would have to significantly degrade the document in order to destroy the watermark (i.e. to achieve false-negative result).

Watermarks could be *perceptible* or *imperceptible*, depending on whether they are directly detectable by the human senses. Perceptible watermarks are often used to display

copyright notices or to lower the commercial value of public or preview documents. In contrast, imperceptible watermarks can only be detected using a computational algorithm, which may or may not require the original unwatermarked document.

Numerous research papers have explored variants of this approach. Praun et al [10] have presented their work based on multiresolution meshes used to embed watermarks into triangle mesh surfaces.

8 Interaction Requirements

The models that are available on the Internet as of today are mostly in VRML format. To visualize and interact with these models, VRML browser plugins are commonly used. These programs have some generic and non-application specific in built interaction techniques. They are panning, examining, walking, zooming, etc.

However, the the interaction suite for specific application must be provided by the developer of the content. In the case of VRML, this is achived by scripting mechanisms. The action to be taken on user interactions with the visible and the invisible entities in the models must be coded using a JavaScript or VBScript node representing the behaviour.

9 Conclusion

The medium 3D geometric models, the WWW as a distribution channel and web browser as the tool for experiencing the media has posed many technical problems in achieving effective results. A large number of innovative solutions have been proposed in the last few years. In this report we surveyed some representative problems and the significant results and implementations.

While there has been a lot of work already done in this field, the problems have not been satisfactorily resolved yet. Also, increasing the volume of the content to be presented to the end-users poses newer problems.

References

[1] Mike Chow. Optimized geometry compression for real-time rendering. In *Proceeding of IEEE Visualization'97*, pages 347–354, 1997.

[2] Michael Deering. Geometry compression. In *SIGGRAPH '95 Proc.*, pages 13–22, 1995.

[3] J. El-Sana and Amitabh Varshney. Topological simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, 1998.

[4] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, CS Dept., Carnegie Mellon U., to appear.

[5] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Proc.*, pages 99–108, Aug. 1996.

[6] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH '97 Proc.*, Aug. 1997.

[7] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH '97 Proc.*, Aug. 1997.

[8] Sun Microsystems. Java3d API specification. http://java.sun.com/products/java-media/3D, June 1999.

[9] Jovan Popovic and Hugues Hoppe. Progressive simplicial complexes. In *SIGGRAPH '97 Proc.*, Aug. 1997.

[10] Emil Praun, Hugues Hoppe, and Adam Finkelstein. Robust mesh watermarking. In *SIGGRAPH '99 Proc.*, Aug. 1999.

[11] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. Technical Report GIT-GVU-98-35, GVU Center, Georgia Tech., Atlanta, USA, 1998.

[12] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proc.)*, 26(2):65–70, July 1992.

[13] Gabriel Taubin, Andre Gueziec, William Horn, and Francis Lazarus. Progressive forest split compression. In *SIGGRAPH '98 Proc.*, Aug. 1998.

[14] Gabriel Taubin, William Horn, and Paul Borrel. Compression and transmission of multi-resolution clustered meshes. Research Report RC21398(96630)2FEB1999, IBM Research Division, 1999.