A Survey of Advanced Interactive 3-D Graphics Techniques

Dieter Schmalstieg
Vienna University of Technology, Austria
email: dieter@cg.tuwien.ac.at - http://www.cg.tuwien.ac.at/dieter/

1. Motivation from human factors

Interactive 3-D Graphics are a new emerging field enabled by powerful and cost-effective computer systems. The goal is to present a three-dimensional image of a scene stored in a computer to give a human user the impression that he or she is looking at a virtual world. In producing such an illusion, we would like to always be able to produce images that match or exceed the limits of human visual perception in all aspects. Unfortunately, limitations in the performance of the image generators we employ for this task defeat this goal. The ever-increasing demands of applications make it unlikely that this situation will ever change significantly. Working solutions require us to trade off image fidelity for graphical complexity and performance of the application. In doing so, it is vital to understand those human factors that dominate the perception of interactive 3-D computer graphics [Helm93].

Visual acuity denotes the degree to which visible features can be perceived, and is commonly measured as the angle subtended at the eye. On-axis resolution is around 1 arc minute; rendering graphical features that fall below the threshold of perception would be wasteful, and the effort could be used better elsewhere.

The human **field of view** is limited to about 180 degrees horizontally and 120 degrees vertically. These are the biological constraints on the visible portion of the environment, which are usually further restricted by display systems. Again, putting effort in displaying graphics outside the field of view is a waste of performance.

Latency is the time measured from the setting of an input until the corresponding output is manifested. Many factors contribute to latency: input devices, software architecture, rendering time, display scan-out. For the rendering portion of the system, latency is typically taken as the time after the eyepoint is set until the last pixel of the corresponding frame is scanned out by the display device. Excessive latency can lead to over-compensation and control oscillations induced by the user.

A sufficiently **high frame rate** is necessary to fool the human eye into seeing a continuous and smooth motion. Generally, most displays have refresh rates close to or above the flicker limit (60-80Hz) of the human visual system, so the display itself introduces no significant problems. However, the time it takes to generate an image by the image generator is bound to scene complexity. Low frame rates make motion choppy and are especially problematic when rapid motion is possible as with a head-mounted display. Furthermore, latency is increased since a low frame rate means a longer time until the next image can be presented.

Constant frame rates are desirable as variations in frame rate tend to distract the user from the task at hand. Variable frame rates also cause temporal inaccuracies because the change in frame rate affects the latency. An unanticipated change in latency leads to a frame not being displayed at the time it was planned for, and results in jerky motion.

2. The rendering pipeline

On most current image generators, rendering is implemented as a pipeline (see Figure 1) that usually includes the following stages:

- **Database traversal** is typically done on the host CPU and can become a bottleneck if the traversal is unable to keep the graphics pipeline full.
- **Polygon processing** includes vertex transformations, lighting calculations and 2-D triangle setup. On very low end image generators, this stage is performed by the host CPU, but the trend goes towards geometry processors that are specialized for this job.
- **Pixel processing** involves operations such as depth buffer testing, anti-aliasing, texturing, and alpha blending. Most of these operations require access to memory, e.g. for looking up texture values, which can have great impact on the performance. Pixel fill rates can also depend on the size of the primitives being rendered because of the per-primitive setup effort.

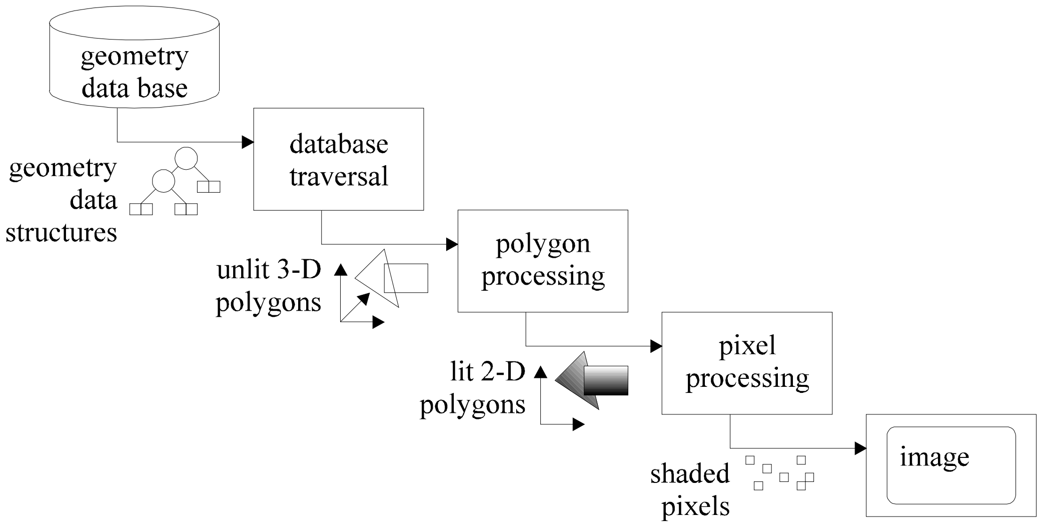

Figure 1: The rendering pipeline

3. Performance of the rendering system

Maximizing frame rate and image quality becomes a problem of making the best use of the each available stages and avoiding bottlenecks. The remainder of this chapter discusses methods and techniques that aim to achieve optimal performance, and also take into account the relevant human factors.

3.1 Reduction of geometric complexity

The usual measure for scene complexity is the number of geometric primitives of which the scene is composed. As there is a hard limit on this number imposed by the hardware, any *reduction of geometric complexity* is desirable. We are primarily interested in such simplifications that cannot be perceived by the human user due to the biological limitations mentioned in section 1. If the simplification cannot be concealed, there usually is a trade-off in image fidelity versus frame rate that has to be resolved. The most important methods for reducing geometric complexity are *visibility processing* (section 4), *levels of detail* (section 5), and the relatively new field of *image based rendering* (section 6).

3.2 Optimizing runtime rendering

Besides reducing the load on the rendering system, it is also necessary to ensure efficient use of the available capacity. This involves tuning the geometric database to avoid suboptimal structures in the data. Some of the measures are relatively simple: Many pipelined image generators are optimized for triangle strips sharing adjacent triangles. Restructuring polygonal data in long strips helps to boost performance. As switching graphics mode and graphical attributes (such as color or shading) involves a performance penalty, the data should be presorted by type and mode if possible to avoid changes in the state of the image generator. Transformations that are important for the modeling process can hurt performance during rendering. Preprocessing allows all static transformation to be eliminated. Many

animations do not depend on run-time parameters, so they can be precomputed and simply replayed at run-time, thus saving computation time.
Beyond these simple optimizations, there are performance issues that require more sophisticated techniques, in particular latency management (section 7) and management of large geometric databases (section 8).

4. Visibility processing

In the early days of raster graphics, a lot of attention has been paid to algorithms that resolve the problem of visibility of opaque surfaces. With the invention of the z-buffer, a relative brute force solution replaced all other methods in workstation-class image generators on the merit of its efficient hardware implementation. However, the virtual environments we would like to render today are composed of so many geometric primitives that resolving occlusion with the help of a z-buffer alone is infeasible. We rather have to compute as accurately as possible those portions of the scene that are actually visible, and let the hardware deal with this subset. Backface culling is a trivial example.

This task of visibility processing is carried out by the host CPU, and can be divided into two phases: *Visibility preprocessing* is an off-line task that computes data that can later be used in the *visibility culling* at run-time to quickly eliminate large portions of the scene that are certainly invisible.

The simplest way of visibility culling is *culling on the viewing frustum*. The problem here is to not let the rendering/clipping process do the work, but rather to rapidly discard the portion of the scene that lies outside the viewing frustum. A thoughtful spatial organization of the geometric primitives in the scene is necessary, which is usually done by grouping spatially coherent primitives into objects. Hierarchical bounding volumes can be used to sort out the invisible geometry. Naylor proposed to use a binary space partitioning (BSP) tree for the scene that can be efficiently intersected with a viewing frustum of any shape [Nayl95].

Virtual environments can broadly be categorized into sparse (i. e. most of the geometry within the viewing frustum can at least partly be seen, up to the virtual horizon), and densely occluded (such as building interiors where most of the geometry contained in the viewing frustum is hidden behind walls and other large occluders).

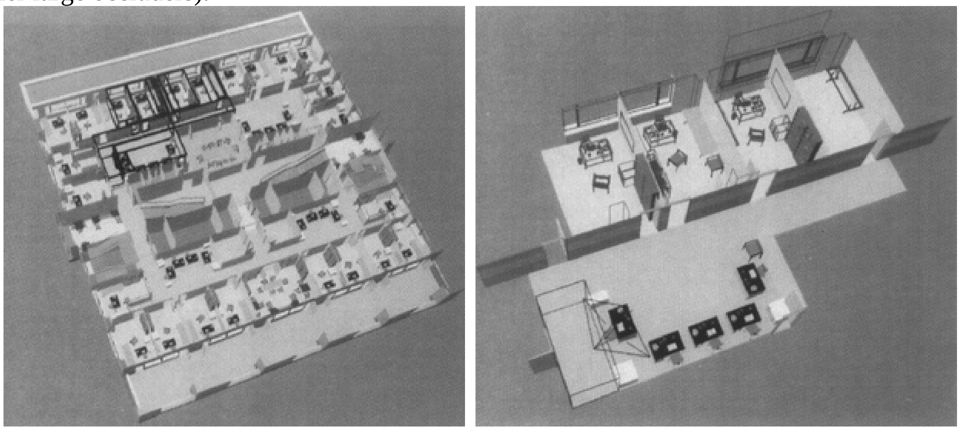

Figure 2: Visibility culling of a building interior from [Funk92]. Left is the unpruned model, right shows the portion visible from the observer

Airey [Aire90] first proposed to make use of the structure of densely occluded environments by employing *Potentially Visible Sets* (PVS). The environment is decomposed into cells. The criterion for the decomposition into cells is that the visible portion of the scene should be roughly the same from any viewpoint within a cell. This is usually approximately equivalent to the rooms of a building that are interconnected only by doors, windows, stairs, and hallways. Any such opening is referred to as a *portal*. The method precomputes a subdivision into cells, builds the cell adjacency graph, and associates with each cell a PVS, which is a conservative overestimate of the actual visible portion that can be efficiently computed. The geometry associated with the PVS is then rendered, and a standard z-buffer is used to resolve exact visibility. Airey used a shadow volume BSP buffer to estimate the PVS. Teller and Sequin [Tell91] have taken this concept further and found an analytic solution to the portal-portal visibility problem. Using linear programming, they compute a complete set of cell-to-cell and cell-to-object visibilities (compare Figure 2). This approach is computationally intensive.

Therefore Luebke and Georges [Lueb95] propose a variant that works without visibility preprocessing: At runtime, they compute 2-D bounding boxes of the portals projected to screen space. During the traversal of the cell adjacency graph, as each successive portal is traversed, its bounding box is intersected with the aggregate culling box using only a few min-max comparisons. The content of each cell is tested for visibility through the current portal sequence by comparing the screen-space projection of each object's bounding box.

Greene, Kass & Miller [Gree93] developed an algorithm called hierarchical z-buffer visibility that works well not only for densely occluded environments, but also for environments that have open spaces. The algorithm uses an octree in object space. The scene is rendered by recursively traversing the octree front to back. Before the geometry associated with a particular node in the octree is rendered, its visibility is estimated by testing the cube associated with the octree node for visibility. This test is further accelerated by a z-pyramid maintained in image space. Using the octree and the z-pyramid as auxiliary data structures, the algorithm is able to exploit simultaneously object space and image space coherence, and temporal coherence for animated walkthroughs as well. However, today's hardware does not fully support the features required by the algorithm, and it has to rely on software rendering.

Coorg and Teller [Coor96] present a spatially and temporally coherent visibility algorithm that exploits properties of the scene by distinguishing large occluders from occludees. The algorithm works entirely in object space, using fast conservative visibility tests. It uses an octree-based subdivision, and eliminates large portions of the model without touching most invisible polygons.

Another class of visibility algorithms deals with the reduced problem of 2 ½ D visibility, that is representative for all environments where the user's movement is constrained to a plane (such as walking on a single floor of a building). Yagel and Ray [Yage96] present a visibility algorithm for cells based on a regular grid subdivision in 2-D. They classify the cells (grid elements) as open, occluded, or containing a wall, and compute approximated eye-to-cell visibility. Schmalstieg and Tobler [Schm97a] developed an algorithm that is based on a 2-D triangular mesh. The edges of the triangles can be elevated and interpreted as walls. They propose a recursive algorithm that can be rapidly executed at run-time and exploits spatial coherence by yielding not only the PVS but also exact visible portions of the walls. This algorithm lends itself especially for low-end platforms that do not have 3-D hardware acceleration.
Visibility computation is generally restricted to walkthroughs of static environments, because they include a heavy preprocessing stage that constructs an auxiliary spatial data structure. Sudarsky and Gotsman [Suda96] make a fist attempt to include dynamically moving objects by introducing temporal bounding volumes that can be used for incremental updates of the spatial data structure.

5. Levels of detail

In very large virtual environments it is commonly the case that many objects are very small or distant. The size of many geometric features of these objects falls below the perception threshold or is smaller than a pixel on the screen. To better use the effort put into rendering such features, an object should be represented at multiple levels of detail (LODs). Simpler representation of an object can be used to improve the frame rates and memory utilization during interactive rendering.

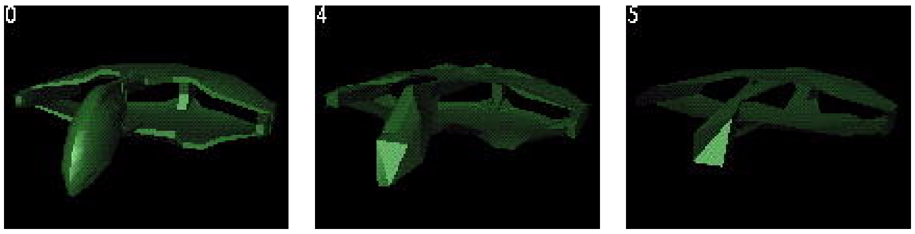

Figure 3: Three levels of detail for a Romulan warbird

This technique was first described by Clark already in 1976 [Clar76], and has been an active area of research ever since. The important questions for the application of levels of detail are: What strategy to use for selecting an appropriate level of detail for each object? How to best stage the transition between two successive levels of detail? And how to create good levels of detail for an original high-fidelity geometric object?

5.1 Level of detail selection

There is no unique way to characterize the best selection of levels of detail for a group of objects comprising a scene, since human perception and aesthetics are hard to catch in a single formula. Instead, heuristics are used. Simple heuristics use the distance of the object from the observer or the size projected to the screen as a measure for the LOD. Unfortunately, these static heuristics do not adapt to variable load on the graphics pipeline: If too many complex objects are close to the observer, an overload can neither be detected nor avoided, and if rendering load is low, the image generator may be idle. Therefore, reactive level of detail selection is employed by flight simulators and real-time rendering toolkits such as Performer [Rohl94] with adaptive level of detail selection according to the time required by the last frames. However, such a strategy still fails to guarantee bounded frame rates, since sudden changes in the rendering load can be underestimated. Funkhouser and Sequin attacked this problem with a predictive selection algorithm [Funk93] formulated as a cost/benefit optimization problem: What selection of levels of detail for each objects produces the best image while the accumulated cost for rendering each objects stays below the maximum capacity of the image generator at the desired frame rate? They use a cost heuristic based on the polygon and pixel capacity of the image generator, and a benefit heuristic constructed as a weighted average of factors such as the object's size, accuracy, and importance. The optimization problem is a variant of the well-known knapsack problem and can be incrementally and approximately solved with tractable computation effort for every frame.

5.2 Level of detail switching

For the use of levels of details, one may not neglect the issue how to stage the transition between two successive representations. The simplest way to do the transition is hard switching: At some point, the simpler model replaces the more complex model. This meets the performance goal, but can cause visual popping which may be disturbing. Instead of simply switching the models, for a short transition period they can be drawn blended together. This substantially reduces the popping effect, but temporarily increases the rendering load while both models are being drawn. Yet superior to blending is geometric morphing of one object into another. While this approach has certainly the best visual and performance effect, it works only for levels of detail with well-defined geometric correspondences.

5.3 Level of detail generation

The principal challenge of level of detail generation is to develop a way that takes a detailed model as input and automatically simplifies its geometry, while preserving appearance as good as possible. While in principle level of detail generation is relevant for complex models composed of any type of geometric primitive, in practice almost exclusively polygonal models are used [Deer93], so research has focused on this class.

Certain aspects are important in the classification of an algorithm that performs the task of polygonal simplification:

- *Local vs. global*: Local techniques operate on individual primitives such as vertices, adjacent edge segments or use some polygon characteristics. They are more apt to fulfill requirements related to small features such as a local preservation of shape. Global techniques attempt to optimize the polygon mesh based upon more general, high-level features of the model.
- *Error bounds*: Some of the methods guarantee user-controlled error bounds on the quality of the simplified objects according to some metric (such as the maximum distance between the surface of the original and the simplified model).

Besides these fundamental consideration, level of detail generation algorithms can be distinguished by algorithmic principles:

Mesh simplification algorithms work with polygonal meshes, often triangle meshes. Local operations on the surface of the object are performed, with a stress on the preservation of important visual features such as shape and topology. As an input, these algorithms expect topologically sound, manifold meshes.

Unfortunately, this criterion is often not met by models generated with CAD packages, which leads to all sorts of practical problems. These algorithm also put more weight on feature preservation, for example, the simplification ratio is bound by the requirement of not reducing the genus of the object. The best results are achieved for smooth, organic objects that are over-tessellated, such as models obtained from a 3-D scanner.

The decimation algorithm by Schroeder, Zarge & Lorensen [Schr92] analyses the vertices of the original model for possible removal based upon a distance criterion. A local re-triangulation scheme is then used to fill the hole resulting from the removed vertex.

Turk’s re-tiling method [Turk92] optimizes a triangle mesh by introducing new vertices to form a new representation. The new vertices are uniformly distributed on the surface of the original object. The original vertices are then iteratively removed, and the surface is locally re-triangulated to best match the local connectivity of the surface.

Hoppe et al. [Hopp93] present a triangular mesh simplification process which was based upon their surface reconstruction work. This technique introduced the concept of an energy function to model the opposing factors of polygon reduction and similarity to the original geometry. The energy function, used to provide a measure of the deviation between the original and the simplified mesh, is minimized to find an optimal distribution of vertices for any particular instantiation of the energy function.

Vertex clustering algorithms ignore topology in both input and output data. As a result, the algorithms perform robustly for degenerate input data, and can achieve arbitrary high compression for any kind of geometry. On the downside, the generated artifacts are much more severe, and local features are not preserved so well.

The fundamental idea of vertex clustering is to reduce the number of vertices of a polygonal model (usually a triangle mesh). Due to perspective distortion individual vertices of an object move closer together on the screen as the distance to the observer increases until they finally fall into one pixel. By creating a cluster of such close vertices and replacing all cluster members by a representative vertex, the number of vertices is reduced. The set of triangles is modified to include only the vertices in the new set. In the course of that process, triangles will degenerate to lines or points and can be removed. Therefore the set of triangles is reduced as well, and any such intermediate data set can be used as an individual LOD.

Several selection criteria have been presented to choose the vertices that are to be clustered. Rossignac and Borrel [Ross93] propose a simple, yet efficient uniform quantization in 3-D. Schaufler and Stürzlinger [Scha95] use a hierarchical clustering method. [Schm97c] proposes a method for the VRML standard, based on octree quantization.

Reconstruction algorithms do not try to simplify the original object, but rather aim to build a new object from scratch that is gradually refined to better approximate the original object.

DeHaemer and Zyda [DeHa91] combine two approaches for approximation of quadrilateral meshes topologically equivalent to regular grids. One approach tries to fit polygons to the original mesh by recursively subdividing them. The other approach starts with a polygon of the original mesh and tries to grow it by merging it with its neighbors until a fitting threshold is exceeded.

He et al. [He95] propose to sample and low-pass filter the object into multi-resolution volume buffers and apply the marching cubes algorithm to obtain a triangular mesh. This method is very robust and has the advantage that it can also deal with non-polygonal (e.g. CSG) input models. However, the resulting meshes are still over-triangulated as a result of the marching cubes algorithm.

Progressive representations take the idea of reconstruction algorithms a step further by representing the original object by a series of approximations that allow a near-continuous reconstruction and can be encoded incrementally in a very compact way.

Lounsbery et al. [Loun93] use wavelets to polygonal objects a multi-resolution data set of wavelet coefficients obtained from a triangular mesh with subdivision connectivity. Levels of detail can easily be constructed by omitting higher order detail coefficients in the reconstruction process. [Eck95] presents a method to transform an arbitrary mesh into an equivalent one with the required subdivision connectivity. This work is taken further in [Cert96] to include colored meshes and support progressive reconstruction of the model.

The progressive meshes introduced by Hoppe [Hopp96], based on edge collapse operation, yields a lossless, continuous-resolution representation for triangular meshes. The representation is generated as a sequence of repeated edge collapses, and is simply inverted in the progressive reconstruction process. The order of applied operation is determined by adopting the mesh simplification method from [Hopp93]. A similar approach that is also based on edge collapse operations is presented in [Ronf96].

Schmalstieg and Schaufler present a progressive geometry data structure called smooth levels of detail [Schm97b]. It is derived from a vertex cluster tree and also offers significant compression.

5.4 Terrain

A specialized problem worth discussing is the representation and simplification of terrain. Digital terrain is generally represented using an elevation model or height field of sample points, effectively a two-dimensional discrete function. Often the sample points are arranged in a regular grid. Such data is easily obtained from sources such as satellite range images and exhibits excessive detail. For speedy rendering, a triangulation of the sample points with a low polygon count must be obtained. Such triangulation schemes can roughly be categorized into regular subdivisions and triangular irregular network (TIN) models. For a survey, see [DeFl96]. Interactive rendering is best achieved by using multi-resolution subdivisions, so that levels of detail can be selected at run-time individually for different regions of the terrain. [Falb93] outlines how real-time management of multi-level terrain data can be achieved. Recent work by Lindstrom et al. [Lind96] proposes a scheme for computing continuous levels of detail for a regular subdivision height field that can be computed incrementally at run-time and supports a user specified error threshold.

6. Image based rendering

The relatively new field of *image based rendering* tries to take advantage of the observation that while the complexity of geometry in a scene is potentially unbound, the complexity of images (of a given resolution) is finite and can easily be estimated in advance for guaranteed rendering performance. Display algorithms typically require modest computational effort and are apt for low-cost and entertainment devices. Furthermore, the source of images can be computer models or digitized photographs, with the option of mixing the two together.

The most established technique that falls in that area is *texture mapping*, which simulates detail by mapping images (often defined using bitmaps) onto flat surfaces. Partially transparent textures can be used to simulate geometry with complex outlines. The widely available hardware-support makes texture mapping the most popular choice for visually rich virtual environments. However, the disadvantage of artifacts stemming in the finite resolution of texture maps cannot fully be overcome by sampling strategies such as mip-mapping [Fole90]. Texture mapping is complemented by *environment mapping*, to capture the light entering a scene from outside in a special texture map, a technique which is also available in hardware now.

A relative straight extension of texture mapping for the purposes of virtual environments are *billboards* [Rohl94]. Radially or spherically symmetric objects such as trees can be approximated by a single texture-mapped polygon, which is always oriented to face the observer.

Maciel and Shirley [Maci95] introduced the concept of an imposter: An image of an object in the approximate direction of the observer is presented in place of the object itself by rendering it as a texture map onto single polygon. Schaufler extended this concept to the dynamic generation of impostors at run-time [Schau96a], rather than as a preprocessing step: The imposter is generated by finding a screen-aligned rectangle surrounding the object and rendering the object into a corresponding rectangular frame buffer using graphics hardware. The resulting image is read from this buffer and used to define the texture on the imposter rectangle.
Subsequently, Schaufler and Stürzlinger [Scha96c] and Shade et al. [Shad96] concurrently developed a hierarchical image cache that uses the concept of imposters to accelerate the rendering of very large polygonal scenes up to an order of magnitude: The scene is decomposed into cells by a hierarchical spatial data structure such as an octree or a BSP tree. This data structure is traversed depending on the projected size of the cells, and a cache of impostors for each node is created and updated as required by an error metric on the validity of the imposter.

The first proposal to build hardware that supports this idea is Talisman [Torb96]. Aimed at the low-end image generator market, this architecture discards the concept of a frame buffer in favor of small image layers that at are composed on the fly at full rendering speed. During the composition process, a full affine transformation is applied to the layers to allow translation, rotation and scaling to simulate 3-D motion. Temporal image coherence is exploiting by re-using the image layers in a way similar to imposters.

Image based rendering for rendering polygonal scenes as outlined above are probably the most sophisticated acceleration tools suitable for the class of scenes categorized as sparse earlier. Beyond polygonal scenes, some work has been recently developed that attempts purely image based rendering, so that the notion of geometric complexity is completely abandoned.

Chen and Williams [Chen93] proposed to synthesize a dynamic view of the environment from a set of environment maps that are composed by image warps. Regan and Post [Rega94] developed a hardware featuring multiple frame buffers that are combined by evaluating depth values. Re-rendering of objects can be delayed until the an object’s view becomes too erroneous.

Quicktime VR [Chen95] is an attempt to use cylindrical or spherical image maps that are warped in real time to simulate camera panning and zooming. The method works very efficiently on low-performance platforms and is now successfully used in entertainment products.

A new approach to model the appearance of objects without the use of explicit geometry was simultaneously introduced by Levoy and Hanrahan [Levo96] as the light field and Gortler et al. [Gort96] as the lumigraph. The object’s appearance is represented a 4-D function, which is a subset of the plenoptic function describing the flow of light from all directions in all directions. This function is sampled to synthesize an image from any given viewpoint.

7. Managing latency

All effort to reduce rendering complexity to fit the maximum capacity of the image generator at the target frame rate can easily be defeated by a suboptimal utilization of the hardware. In particular, the rendering process is constantly facing deadlines in the form of refresh cycles for the display device. If the rendering is not completed before the scan-out of the new frame starts, a whole frame’s time is lost, so it is essential to complete rendering timely in order to keep the graphics system occupied.

One way to achieve this is to make use of both host CPU and graphics processor. A typical setup of the pipeline for the CPU includes an application task (simulation, modification of the scene graph), a database task (scene graph traversal, visibility culling, level of detail selection) and a draw task (transferring data to the graphics subsystem). As with any pipeline, these task should take even time slices to achieve a maximum utilization, and should furthermore be tuned to meet the refresh cycle deadlines. This goal can be defeated by dynamic shifts in the load of individual stages. However, if any stage always picks up the latest available instance of the pipelines data, the pipeline is at least kept from stalling.

One limiting factor in this setup are the fixed time slices enforced by the refresh cycle of the display. This factor was criticized by Bishop et al. in [Bish94]. They proposed frameless rendering, where individual random pixel are updated rather than frames, and always use the latest viewpoint information, which is particularly suitable for immersive systems employing head-tracking. However, current image generators do not support this approach. Mazuryk, Schmalstieg and Gervautz [Mazu97] proposed a simple scheme that trades traditional double buffering in favor of a copy/zoom operation supported by 2-D bit block transfer hardware, which is a standard component of today’s image generators. New frames can be presented independently of the refresh cycle, and the zoom operation can be used to compensate pixel-dominated rendering overload for low-cost image generators. This approach can be combined with 2-D image deflection to reduce dynamic viewing errors in head-tracked displays. This approach was further refined by Schaufler, Mazuryk and Schmalstieg in [Scha96b] by replacing 2-D image deflection with the more capable 3-D image deflection on the basis of dynamic imposters.

8. Managing large geometric databases

For very large scenes, interactive rendering also involves a number of issues in database management, for the sheer size of the involved data sets. Conservative use of main memory can enable handling of larger geometric databases. While memory is gradually moving away from being the most limiting factor in large-scale applications, this constraint is readily replaced by the limited bandwidth of network connections such as the Internet, that are essential in distributed virtual environments. Slow network transmission of large geometric data sets interferes with the responsiveness requirements of interactive 3-D applications.

An important aspect is the paging if geometry and texture from the secondary storage into memory. An implementation should take care of:

- Achieving full I/O bandwidth by arranging the data so it can be fetched from the secondary storage in large chunks rather than small items.
- Minimizing impact on frame rate by either dividing the I/O task over multiple frames, or running it as an asynchronous process that does not interfere with the main rendering task.
- Accurate prediction and timing to avoid situations where the required data is unavailable, and rendering must be stalled.

Funkhouser, Sequin and Teller [Funk92] present an application capable of presenting an interactive walkthrough of a database much larger than memory. The data is pre-loaded by predicting the users movements and reducing the required data based on visibility considerations for building interiors.

For networks, progressive level of detail representations as discussed above allow to make instant use of partially transmitted data. A related topic is geometry compression: Data sets can be transmitted in a compressed form, then expanded at the receiver for more rapid rendering. The first step in this direction was made by Deering [Deer95], who proposed a compression scheme for triangular meshes based on generalized triangle strips, including normals and colors. Taubin and Rossignac [Taub96] introduce a procedure they call topological surgery, which transforms a triangular mesh into an alternate representation based on spanning trees for triangle strips and vertices, that can be encoded in an extremely compact form and yields high compression rates. Levoy [Levo95] proposed to combine JPEG compression of animated sequences with simple polygonal rendering data to yield higher compression ratios for digital movies.

9. Summary

Interactive 3-D graphics is a rapidly evolving field inside computer graphics. While traditionally, computer graphics has focused on the quality of images alone, the real-time requirements of interactive applications make it necessary to employ all sorts of trade-offs to maintain frame rates and satisfy ergonomic needs. Techniques such as visibility processing, levels of detail, image based rendering, and the management of latency and large geometric databases make it necessary to adopt methods not only from computer graphics, but also from real-time systems, databases, networking and many other domains. Despite the many shortcomings that current implementations have, the importance of interactive 3-D graphics is certain to grow at an enormous pace.

References

[Aire90] J. M. Airey, J. H. Rohlf, F. Brooks Jr.: Towards Image Realism with Interactive Update Rates in Complex Virtual Building Enviroments. Computer Graphics, Vol. 24, No. 2, pp. 41 (1990)

[Bish94] G. Bishop, H. Fuchs et al.: Frameless Rendering: Double Buffering Considering Harmful. Proceedings of SIGGRAPH'94, pp. 175-176 (1994)

[Cert96] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, W. Stuerzle: Interactive Multiresolution Surface Viewing. Proceedings of SIGGRAPH'96, pp. 91-98 (1996)

[Chen93] S. Chen, L. Williams: View interpolation for Image Synthesis. Computer Graphics (Proceedings SIGGRAPH), pp. 279-288 (1993)

[Chen95] S. Chen: Quick Time VR - An Image-Based Approach to Virtual Environment Navigation. Computer Graphics (Proceedings SIGGRAPH), pp. 29-38 (1995)

[Clar76] J. Clark: Hierarchical Geometric Models for Visible Surface Algorithms. Communications of the ACM, Vol. 19, No. 10, pp. 547-554 (1976)

[Coor96] S. Coorg, S. Teller: A Spatially and Temporally Coherent Object Space Visibility Algorithm. Tech. Report TM-546, Laboratory for Computer Science, MIT (1996)

[Deer95] M. Deering: Geometry Compression. Computer Graphics (Proceedings SIGGRAPH), pp. 13-20 (1995)

[DeFl96] L. De Floriani, P. Marzano, E. Puppo: Multiresolution models for topological surface description. Visual Computer, Vol. 12, pp. 317-345 (1996)

[DeHa91] M. DeHaemer, M. Zyda: Simplification of Objects Rendered by Polygonal Approximations. Computers & Graphics, Vol. 15, No. 2, pp. 175-184 (1991)

[Eck95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle: Multiresolution Analysis of Arbitrary Meshes. Computer Graphics (Proceedings SIGGRAPH), pp. 173-182 (1995)

[Falb93] J. Falby, M. Zyda, D. Pratt, L. Mackey: NPSNET: Hierarchical data structures for realtime 3-dimensional visual simulation. Computers & Graphics, Vol. 17, No. 1, pp. 65 (1993)

[Fole90] J. Foley, A. van Dam, S. Feiner, J. Hughes: Computer Graphics: Principles and Practice. Addison-Wesley Publishing Co., ISBN 0-201-12110-7 (1990)

[Funk92] T. Funkhouser, C. Sequin, S. Teller: Management of Large Amounts of Data in Interactive Building Walkthroughs. SIGGRAPH Symposium on Interactive 3D Graphics, pp. 11-20 (1992)

[Funk93] T. Funkhouser, C. Sequin: Adaptive Display Algorithm for Interactive Frame Rates During Visualisation of Complex Virtual Environments. Computer Graphics (Proceedings SIGGRAPH), pp. 247-254 (1993)

[Gort96] S. Gortler, R. Grzeszczuk: The Lumigraph. Computer Graphics (Proceedings SIGGRAPH), pp. 43-54 (1996)

[He95] T. He, L. Hong, A. Kaufman, A. Varshney, S. Wang: Voxel Based Object Simplification. Proc. SIGGRAPH Symposium on Interactive 3D Graphics, pp. 296-303 (195)

[Helm93] J. L. Helman: Designing VR Systems to Meet Psysio- and Psychological Requirements, SIGGRAPH Course Notes, No. 23 (1993)

[Hopp93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle: Mesh Optimization. Computer Graphics (Proceedings SIGGRAPH), pp. 19-26 (1993)

[Hopp96] H. Hoppe: Progressive meshes. Proceedings of SIGGRAPH ‘96, pp. 99-108 (1996)

[Levo95] M. Levoy: Polygon-Assisted JPEG and MPEG Compression of Synthetic Images. Computer Graphics (Proceedings SIGGRAPH), pp. 21-25 (1995)

[Levo96] M. Levoy, P. Hanrahan: Light Field Rendering. Computer Graphics (Proceedings SIGGRAPH), pp. 31-42 (1996)

[Lind96] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, G. Turner: Real-Time Continuous Level of Detail Rendering of Height Fields. Computer Graphics (Proceedings SIGGRAPH), pp. 109-118 (1996)

[Loun93] M. Lounsbery, T. DeRose, J. Warren: Multiresolution analysis for surfaces of arbitrary topological type. Technical Report, 93-10-05b, Dept. of Comp. Sci., Univ. of Washington (1994)

[Lueb95] D. Luebke, C. Georges: Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets. Proceedings SIGGRAPH Symposium on Interactive 3D Graphics, pp. 105-106 (1995)

[Maci95] P. Maciel, P. Shirley: Visual Navigation of Large Environments Using Textured Clusters. SIGGRAPH Symposium on Interactive 3-D Graphics, pp. 95-102 (1995)

[Mazu97] T. Mazuryk, D. Schmalstieg, M. Gervautz: Zoom Rendering: Improving 3-D Rendering To appear in: The International Journal of Virtual Reality. (1997)

[Nayl95] B. Naylor: Interactive playing with large synthetic environments. SIGGRAPH

Symposium on Interactive 3D Graphics (1995)

[Rega94] M. Regan, R. Pose: Priority Rendering with a Virtual Reality Address Recalculation Pipe. Proceedings of SIGGRAPH'94, pp. 155-162 (1994)

[Rohl94] J. Rohlf, J. Helman: IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. Proceedings of SIGGRAPH'94, pp. 381 (1994)

[Ronf96] R. Ronfard, J. Rossignac: Full-range Approximation of Triangulated Polyhedra. Proceedings of EUROGRAPHICS' 96, pp. 67-76 (1996)

[Ross93] Jarek Rossignac, Paul Borrel: Multi-Resolution 3D Approximation for Rendering Complex Scenes. IFIP TC 5.WG 5.10 II Conference on Geometric Modeling in Computer Graphics (1993)

[Scha95] G. Schaufler, W. Stürzlinger: Generating Multiple Levels of Detail from Polygonal Geometry Models. Virtual Environments'95 (ed. M. Göbel), Springer Wien-New York (1995)

[Scha96a] G. Schaufler: Integrating impostors into a level of detail algorithm. Proceedings of Virtual Reality Annual International Symposium (VRAIS'96) (1996)

[Scha96b] G. Schaufler, T. Mazuryk, D. Schmalstieg: High Fidelity for Immersive Displays. Short paper, ACM SIGCHI'96 conference companion. Also technical report TR-186-2-96-02, Vienna University of Technology, Austria (1996)

[Scha96c] G. Schaufler, W. Stürzlinger: A Three-Dimensional Image Cache for Virtual Reality. Proceedings of EUROGRAPHICS'96 (1996)

[Schm97a] D. Schmalstieg, R. Tobler: Exploiting Coherence in 2 1/2 D Visibility Computation. Computers & Graphics, Vol. 21, No. 1 (1997)

[Schm97b] D. Schmalstieg, G. Schaufler: Smooth Levels of Detail. Proceedings of the Virtual Reality Annual International Symposium (VRAIS'97), pp. 12-19, Monterey CA (1997).

[Schm97c] D. Schmalstieg: An Octree-Based Level of Detail Generator for VRML. Proceedings of the ACM SIGGRAPH 1997 Symposium on VRML (VRML'97), pp. 127-133, Monterey CA, Feb 24-27 (1997)

[Schr92] W. Schroeder, J. Zarge, W. Lorensen: Decimation of Triangle Meshes. Proceedings of SIGGRAPH'92, pp. 65-70 (1992)

[Shad96] J. Shade, D. Lischinski, D. Salesin, T. DeRose, J. Snyder: Hierarchical Image Caching for Accelerated Walkthruoghs of Complex Environments. Computer Graphics (Proceedings SIGGRAPH), pp. 75-82 (1996)

[Suda96] O. Sudarsky, C. Gotsman: Output-Sensitive Visibility Algorithms for Dynamic Scenes with Applications to Virtual Reality. Proceedings of EUROGRAPHICS' 96, pp. 249-258 (1996)

[Taub96] G. Taubin, J. Rossignac: Geometric Compression Through Topological Surgery. Technical Report RC-10340, IBM T. J. Watson Research Center (1996)

[Tell91] S. Teller, C. Séquin: Visibility Preprocessing For Interactive Walktroughs. Computer Graphics (Proceedings SIGGRAPH), Vol. 25, No. 4, pp. 61-69 (1991)

[Torb96] J. Torborg, J. Kajiya: Talisman: Commodity Realtime 3D Graphics for the PC. Computer Graphics (Proceedings SIGGRAPH), pp. 353-364 (1996)

[Turk92] G. Turk: Re-Tiling Polygon Surfaces. Computer Graphics (Proceedings SIGGRAPH), pp. 55-64 (1992)

[Yage95] Yagel R., Ray W.: Visibility Computation for Efficient Walkthrough of Complex Environments. Presence, Vol. 5, No. 1, pp. 45-60 (1995)