Nonlinear Analysis: Modelling and Control, Vilnius, IMI, 2000, No 5

Levels of Detail: An Overview

Zoran Constantinescu

Dept. of Computer & Information Sciences,
Norwegian University of Science and Technology
Gloshaugen, 7491 Trondheim, Norway
zoranc@acm.org

Received: 05.09.2000
Accepted: 11.10.2000

Abstract

This paper overviews some aspects of using different levels of accuracy and complexity for the visualization of large data sets. Current status of the volume of data sets that can be generated is presented, together with some of the inherent problems due to such large data volumes, visualization requirements, and display limitations of existing hardware graphics. Some methods for selecting, generating and implementing different levels of detail are presented.

Keywords: computer graphics, visualization, level of detail, implementation.

1 Introduction

Very fast development of the computers in the last years made possible for the scientists to simulate larger and larger numerical models of physical processes. For example, the recent ASCI Red supercomputer is capable of 1.3 Terra-flops, making it possible to generate data sets of complex phenomena in very short time. One of the problems which come with such large data sets is *how to understand* them. Scientific Visualization or the use of computer graphics to represent the data in human understandable way, has an important role in dealing with it. As R. Hamming stated: *"The purpose of computing is insight, not numbers"*.

[1]http://www.idi.ntnu.no/žoran

There are mainly two ways of handling such large data sets: feature extraction and data inspection. We will concentrate on the second method, the interactive navigation through the data set, describing some of the problems imposed by the volume of data and the limitations of current hardware, and how it is possible to deal with them using reduced versions of the data sets,

One way of displaying large data sets is to use different *levels of detail* (LOD) to represent the data. Depending on how important the data is for the user, it can be displayed in a more or a less complex definition, or resolution. This is closely related to the way the human visual system works. The level of detail solution reduces displayed detail in order to improve system responsiveness, instituting a tradeoff between visual and temporal fidelity.

2 Problems

Current powerful computers used in numerical simulations are capable of producing huge amounts of data. The first Terra-operations per second supercomputer, ASCI Red was finished in July 1997. It was developed by Intel Corporation and Sandia National Laboratories, and consists of more than 9,000 Intel Pentium-Pro processors. The power of this computer allows the scientists to simulate larger physical models than before. Data sets of hundreds of Giga bytes and even Terra bytes can be the results of such simulations. These large data sets are very important, since they can reveal new complex phenomena in the models. Understanding the data results of this magnitude is a significant challenge, both from the technical and human side. High-end hardware equipment is required to process and transmit this volume data in reasonable time.

There are two basic approaches in dealing with such large amounts of data. One is to *extract* some particular (known) *features* from the data. This means an automated way of identifying a specific phenomena and then creating or extracting a much smaller subset of the original data which describes that particular phenomena. The other one is to interactively *browse* or navigate through the data, looking for new aspects of the simulation and understanding the phenomena. It allows to investigate the original data, which can contain new phenomena, whereas the first method would extract only the known ones. The disadvantage of the second approach is that it needs a more powerful computer system which is capable of accessing, processing and visualizing the data in an interactive way.

There is a minimum number of frames per second that should be generated in order to be able to navigate in an interactive way through the data. When talking about interactive visualization, a rate of around 15-30 frames per second is required. Usually, a frame rate of 15 frames/sec is considered acceptable for the human visual system. However, for a real-time navigation a frame rate of minimum 30 is required.

The interactive navigation through large data sets is currently limited by the existing capabilities of the graphics hardware. The maximum number of frame rate the graphical hardware can achieve depends directly on the complexity of the rendered image. It means that a less complex image can be rendered at a higher frame rate than a more complex one. For large data sets, i.e. more complex images, the number of frames that can be displayed in one second is dramatically reduced.

Current hardware for 3D image rendering impose a limit in the complexity and size of the large data sets that can be displayed at such rates. Recent advances in 3D accelerated graphic cards for PCs have performance of rendering about 10-30 million triangles per second. High end graphical systems with multiple graphic pipelines are capable of achieving rendering of at most one order of magnitude higher. However, with very large and complex models, the number of frames per second which can be rendered can drop easily. This means that the graphics hardware itself is not enough for this amount of data, and some more advanced software techniques must be used.

3 Classification

One of the possible solutions for representing such large data sets is to use the idea of varying level of detail. This refers to model and rendering complexity, which can be varied to ensure rendering at some acceptable frame rate.

The concept of *Level of Detail* (LOD) is generically related to the possibility of using different representations of a geometric object (a surface, a volume, an image, etc.), having different levels of accuracy and complexity [PS97]. The visualization system can select a particular representation to use for each object. The outline of the object could be considered as the least complex representation (the coarsest mesh), whereas the full detail object is the most complex representation (the finest mesh). Using a less complex representation of the object we can obtain a much better displaying speed than using a more complex representation.

Of course there is a price for this: the less complex objects are represented in a lower detail, while the complex ones offer a more accurate representation. However, the human visual system is far more tolerant to reductions in image quality than to delays in the visual updates. This means that there is possible a tradeoff between the human performance impact of changed responsiveness and the impact of the corresponding change in displayed visual detail. Using principles of the human visual perception can result in more efficient visualization system, that can work with larger data sets.

3.1 Selection of LOD

There are different techniques for selecting a specific level of detail. They attempt to sidestep the LOD tradeoff by reducing detail only when it is not perceivable, giving the benefits of improvements in system responsiveness without the corresponding cost of detail loss.

We can organize these methods in the following groups. One group which focuses on removing details that don't need to be rendered by the graphics display hardware. Here we have different culling techniques, which based on certain criteria remove some parts of the model. A second group consists of techniques that remove details that cannot be rendered by the hardware. Methods based on distance and object size fall in this group. In another group we have algorithms that try to strip away details that could not be perceived by the human viewer. These are methods based on eccentricity, depth of field and velocity [Red97]. There is also another method with the main goal of maintaining a constant high frame rate, regardless of the complexity of the model.

3.1.1 Culling

There are certain situations when some parts of the geometry are never seen by the user. In these cases, those parts are not represented by the graphical system. In the following, some of these situations are presented [Cab97].

The *view frustum culling* method removes all the objects that are outside the user's viewing pyramid. Such a case is presented in figure 1 a). All the objects which are outside the frustum are discarded (the gray ones in the figure). This improves the performance of the graphical representation by eliminating unnecessary work in early stages of the rendering pipeline.

Another technique is *contribution culling.* The objects whose visual

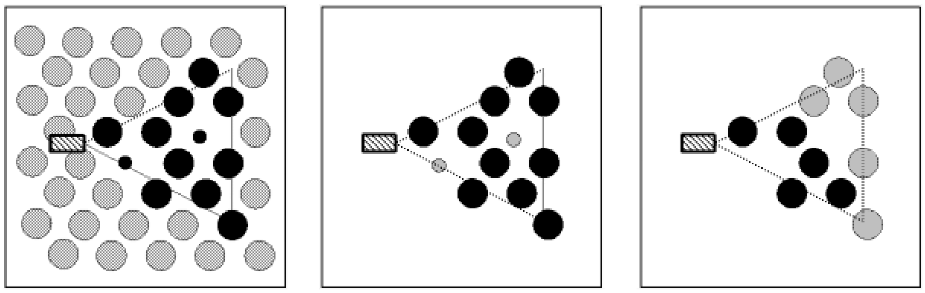
a) View frustum culling b) Contribution culling c) Occlusion culling

Figure 1: Culling techniques (from [Cab97])

contribution is smaller than a certain threshold are discarded, even if they may be visible. The factors defining the contribution are generally object size and range, display window size, and monitor resolution. This situation is presented in figure 1 b). Some of the small objects (in gray) are discarded, even though they are inside the frustum. This method reduces the high-fidelity provided by the original scene, but it can reduce very much the complexity especially when we have a lot of small objects.

Perhaps the most powerful culling techniques is *occlusion culling*. In this case, presented in figure 1 c), the culling is obtained by further finding and removing all the objects that are hidden or occluded by other objects. It is also possible, as further refinement, to discard objects that are only partially occluded.

3.1.2 Distance

A second technique considers how far the objects are from the viewer. The measure is based upon the Euclidean distance between the viewpoint and a predefined point inside the object. This approach is based on the theory that as an object progresses further away from the viewpoint, fewer of its high detail components are visible. This means that we can select a lower detail without greatly affecting the fidelity of the image. Use of numerical methods like wavelets or cosine transform can be very efficient for representation, expressing the details in terms of frequency components.

There are two main advantages for this method. One is its simplicity: all that is required is to check if a distance exceeds a predefined threshold. The second one: it is efficient in that only one computation needs to be done, the distance between the view point and the object. The main disadvantage is how to choose the point from the object's volume. If we choose a fixed point in the object, the actual distance between the near side of the object and the viewpoint can change depending upon the orientation of the object. For large or very close objects, the choice of this point is not very easy. Use of some numerical measure of the complexity of the object can help in selecting such a point.

Another problem is that if we scale the object (to make it larger or smaller), or if we use a different display resolution, then the original distance threshold is no longer valid and must be scaled appropriately. This technique has been successfully used in the flight simulator field and for terrain visualization.

3.1.3 Size

A second technique takes advantage of the eye's reduced ability to perceive objects as the size of those objects decreases. It takes into consideration the size of the objects represented, the smaller objects being represented at a lower detail, while the larger ones at a higher detail. It can be considered also as another way of implementing distance LOD, as objects which move further away will appear smaller on the display device.

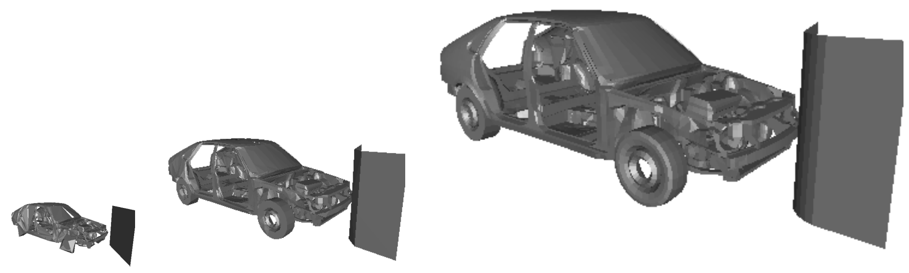

Figure 2: LOD by size

There are however a number of advantages over the distance approach. It provides a measure to determine the visibility of features within an object, regardless of displaying resolution or object scaling. There is no more need to select a point for the calculation of the distance. The method has also a main disadvantage. Compared to the previous method, it is more computationally expensive, because we need to project a number of world coordinates into view coordinates and then compute the projection's size. However, there exists some very efficient implementations for this in the literature.

3.1.4 Eccentricity

Another technique for selecting a LOD takes advantage of the eye's reduced ability to perceive objects out of the center of the field of view (periphery). The center of the retina (fovea) is used for a high detail search in the area of fixation, hence a higher resolution, while the periphery of the retina is used during selection of the next fixation point. This suggests in a way to divide the display in a central, high detail area, corresponding to the center of the eye's field of view, and a surrounding, simpler periphery area, corresponding to the peripheral areas of the field of view.

Experiments shown that we can reduce visual complexity in the periphery without adversely affecting visual search task performance. This suggests that a useful LOD management system might be implemented using a peripheral degradation approach. This peripheral degradation technique requires knowledge of the user's current gaze direction. For this, head and/or eye movement should be tracked. It is also possible to assume that the user is looking towards the center of display. Elements that the user is looking at will be displayed in higher resolution than those in the visual periphery.

3.1.5 Depth of Field

The representation of an object can be selected based on the depth of focus of the user's eyes. Objects in front or behind this fusional area are unfocused. This means that objects out of the fusional area will appear in lower detail, whereas those in that area will have a higher detail. Both eyes focus on the objects in the fusional area. In order to obtain an correct value of the convergence distance both eyes must be tracked accurately.

3.1.6 Velocity

Velocity LOD is when an object's representation is selected based upon its angular velocity relative to the viewer's gaze. Objects moving quickly across the screen appear blurred. They can be seen for only a short period of time, and hence the user may not be able to seen them clearly. It is possible then to represent these objects at lower level of detail. Different metrics can be used for selecting the detail, for example the ratio of the object's apparent speed to the size of an average polygon.

3.1.7 Fixed Frame Rate

Essential for good interactivity is to maintain a high and consistent frame rate. This means that once chosen a specific frame rate, it is maintained without fail regardless of the complexity of the view.

This techniques usually includes a scheduler, whose job is to analyze the system load and assign LOD ratings to each object accordingly. There are different approaches for this: reactive system, predictive system, preemptive system.

The reactive system simply adjusts the detail based upon whether the previous frame was rendered within the target frame rate. If the last frame was completed after the deadline, then detail is reduced, otherwise detail can be increased.

A predictive system estimates the complexity of the frame about to be rendered and enforces level of detail assignments to ensure that the update deadline is never exceeded.

In a preemptive system [Wie96], the objects from the model are prioritized and rendered in priority order. The user sets the desired frame rate, and if the allowed time for a frame has elapsed, the rendering for that frame is stopped and started for the next frame. So, if a frame rate is chosen and the computer is not powerful enough, then less detail will be seen from the model when navigating. When the navigation is stopped, the rendered frames being the same, it is possible to fill in the missing detail.

3.2 Generation of LOD

3.2.1 Illumination Models

Using different illumination techniques for lighting and shadowing, the displayed detail can be enhanced and different levels of detail can be obtained for the objects. This means that, for example, we can use less polygons and an improved illumination algorithm to obtain a similar representation as a more-polygons representation of the object. However, some of these techniques require a lot of computation to be done, so that not always their use is appropriate. Such lightning models include Phong and Gouraud (smooth shading), which produce more realistic results than the Lambertian model (flat shading).

3.2.2 Texture Mapping

Another way of representing different levels of detail is by using some textures. Regions with high geometric detail could be replaced by a single textured polygon. The polygon's texture is simply a rendered image of that section, from a certain viewpoint and distance. However, this optimization can introduce visual artifacts if the model is viewed from a different viewpoint or distance. Different techniques exist for this, including warping the texture image or the adjacent geometry, or smoothly transforming between geometry and texture.

3.2.3 Polygon Reduction

The objective of a polygon reduction algorithm is to take a high-detail model with many polygons and to generate a simpler model with fewer polygons that looks reasonably similar to the original, retaining its important visual characteristics. The advantage of the simplified representation is that it can be rendered faster than the original model, possible at interactive rates.

The simplification must be made in such a way that the general shape of the model is preserved. Some of the features that a simplification algorithm must be aware of are [KBGT97]:

- *planar area* is identified inspecting the normals of adjacent polygons; these polygons can be merged to form bigger ones. This is the easiest type of simplification;
- *sharp edges* are found by comparing the angles between the normals of adjacent faces; they can be simplified by merging connected edges which are nearly collinear;
- *pointed edges* (like the tip of a pyramid) must be preserved; they can be detected by using the local curvature around a vertex.

Algorithms for polygon reductions require some kind of heuristics to choose the relevant primitives and control the simplification. A few simple operators can be used for removing primitives from a model [AP94]:

- *normalization*: removal of degenerated faces or edges and any primitive defined multiple times;
- *vertex simplification*: merging of all points included within a volume (a sphere or a grid cell);

- *edge simplification*: removal of all edges shorter than some threshold;
- *angle based simplification*: removal of edges which form a closed angle;
- *face size simplification*: removal of all faces which have an area smaller than some threshold;
- *face normal simplification*: merging of all adjacent faces with near-parallel normals.

To produce a good low-polygon model, at each step is selected such an element that, when collapsed, will cause the smallest visual change to the model.

There are three main categories of simplification algorithms:

- *geometry removal*: an algorithm that simplifies a model by removing vertices or polygons from its description; the selection is made using some kind of heuristics; this is one of the most popular algorithms for simplification;
- *sampling*: an algorithm that samples a model's geometry and then attempts to generate a simplified model that closely fits the sampled data; it is usually difficult to choose the samples in a manner that preserves the overall shape of the object;
- *adaptive subdivision*: an algorithm that begins with a simple base model and recursively subdivides it, adding further detail to local regions of the model at each iteration.

3.3 Implementation of LOD

3.3.1 Static LOD

This is the simplest case for a LOD representation of a mesh, consisting of a collection of meshes of different sizes, each representing the object at a different resolution. Each of these is defined in an independent way from the other meshes, and is associated with a certain *range* of detail, representing a measure of the mesh. The range is used later to select between meshes for representation. Each representation is generated in a preprocessing stage and stored. At run time, one of the representation is chosen based on certain LOD selection criteria.

One of the most important advantage of static LOD is the simplicity of programming. The generation has no real time rendering constraints, since it is made in a preprocessing step. We have a separation of the simplification algorithm and the rendering, resulting in a simpler programming. The only thing to do at run time is to select the appropriate LOD. Another advantage is that the preprocessing stage can generate better representations which fit better the existing graphics hardware, like triangle strips and display lists. This polygonal representations can render much faster than unorganized representations.

There are also some drawbacks for such static representations of LODs. Each mesh being stored independently, the requirement for memory increases with the number of levels of detail stored. In practice there are always some constrains for memory requirement, which means that we can't store too many levels. A problem of static LODs is how to choose the scales at which the generated representations are stored. Usually some heuristic measures are used for these scales (for example reducing the complexity of each mesh by a factor of two from the previous mesh).

Because the number of levels of detail stored are very limited, the changes between consecutive representation levels are usually abrupt, creating undesirable popping effects in visualization during switches from one level to another. Different methods are used for the transition between these levels of detail. The simplest one is based on instantaneous switch, one level of detail being rendered during one frame, and a different one during the following frame. This is the most efficient of the methods, but results in the most noticeable artifacts. Another technique is the morphed transition, which involves gradually changing the shape of the surface as the transition occurs. This requires the use of some correspondence between the two levels of detail.

3.3.2 Dynamic LOD

In a dynamic LOD a data structure is created from which any desired level of detail can be extracted at run time. Such a data structure is called a multiresolution mesh. Depending on the complexity of such a data structure, it is possible to obtain a smaller or a higher number of meshes at different resolutions, or even a continuous range of meshes. The number of different levels of detail that such a model can provide will not be fixed a priori. Some of desirable properties of a multiresolution mesh are: it must be able to provide a mesh at a given resolution in a very short time (real-time); the size of the model should not be considerable higher than the size of the highest resolution mesh it can provide; the transition between different meshes extracted from the model must be as smooth as possible, avoiding abrupt changes when moving from a mesh to another at a close LOD.

Because the total number of possible LOD representations for an original mesh can be very large, creating and storing independently each of the representation is not possible. Instead, the common information for these representations is used to create a unified single representation. From this unified representation the desired LOD is extracted at run time, depending on the viewing parameters.

There are a few main classes in which data structures for dynamic LOD can be divided. One of them is based on the evolution of the mesh throughout the simplification or refinement process. Such a representation is called a progressive mesh and is simply the original mesh plus an ordered list of the simplification operations performed on the mesh [Hop96]. It is generally more convenient to reverse the order of this and store the simplest mesh plus the inverse of each simplification operation. To extract a desired LOD, a number of such operations is performed. Another class of representations for dynamic LOD is based on tree-like hierarchies. These structures, for example, can capture the dependency of each simplification operation on certain previous operations in a vertex hierarchy [Hop97], or can represent the mesh as a hierarchy of nested region subdivisions, each region being recursively subdivided in a set of smaller regions covering it exactly. A general framework for the representation of multiresolution meshes as multi-triangulation can be found in [PS97]. Other methods for generating dynamic LOD structures can be based on wavelet-based representations.

Dynamic levels of detail perform some of the simplification as a preprocess, but defer some of the work to be computed by the visualization system at run time. This allows to incorporate more view dependent criteria in the selection of LOD representation. Another advantage is that dynamic LOD can adjust detail gradually and incrementally, reducing visual pops. The shortcoming of such representations is that they have more memory and computation requirements in the visualization system than the static LOD.

4 Conclusions

The main idea is that less detailed models will be rendered when the user will perceive less detail. This suggests to represent some parts of the object at different resolutions than other parts. For example more close parts at a higher definition, while more distant ones at a lower definition.

There are many challenges remaining in the visual representation of large data models. Rendering of extremely large models (with millions and more polygons) in real time is not easy, limitations of current hardware graphics capabilities being a major drawback. Implementing a system which uses contemporary models of visual perception in order to efficiently modulate levels of detail of the objects is also a challenge.

References

[AP94] Peter Astheimer and Maria-Luise Pöche. Level of Detail Generation and Its Applications in Virtual Reality. In *Proc. of VRST '94*, pages 299–312, 1994.

[Cab97] Brian Cabral. OpenGL Optimizer 1.0: The Power of Silicon Graphics' Next-Generation Visualization Technology. *Developer News*, 1997.

[Cla76] James H. Clark. Hierarchical Geometric Models for Visible Surface Algorithms. *Communications of the ACM*, 19:547–554, October 1976.

[Hop96] Hugues Hoppe. Progressive Meshes. In *Proc. SIGGRAPH '96*, 1996.

[Hop97] Hugues Hoppe. View Dependent Refinement of Progressive Meshes. In *Proc. SIGGRAPH '97*, 1997.

[KBGT97] Mike Krus, Patrick Bourdot, Françoise Guisnel, and Guillaume Thibault. Levels of Detail and Polygonal Simplification. *ACM Crossroads*, (3.4), 1997.

[Lue98] David P. Luebke. *View-Dependent Simplification of Arbitrary Polygonal Environments.* PhD dissertation, University of North Carolina at Chapel Hill, 1998.

[PC99] Gerald Pitts and Daniel Cornell. Peripherality Based Level of Detail Switching as a Visualization Enhancement of High-Risk Simulations. In *Proc. of ACM Symposium on Applied Computing*, pages 98–104, 1999.

[PS97] Enrico Puppo and Robert Scopigno. Simplification, LOD, Multiresolution: Principles and Applications. *EUROGRAPHICS*, 16(3), 1997.

[Red97] Martin Reddy. *Perceptually Modulated Level of Detail for Virtual Environments.* PhD dissertation, University of Edinburgh, 1997.

[Wie96] Tim Wiegand. Interacting with Very Large CAD Databases. NavisWorks, `http://www.arct.cam.ac.uk/mc/cadlab`, 1996.

[WWH97] Benjamin Watson, Neff Walker, and Larry F. Hodges. Managing Level of Detail through Head-Tracked Peripheral Degradation: A Model and Resulting Design Principles. In *Proc. ACM VRST '97*, pages 59–63, 1997.