INTERACTIVE VISUALIZATION – PROBLEMS AND APPROACHES

Ani Dimitrova

***Abstract**: The visualization of large datasets at interactive rates has been a great challenge for decades. As a result a lot of approaches have been used and a lot of techniques have been developed. The presented article tries to summarize the main approaches. First, a definition of the term "large data" is given and the need of out-of-core techniques is explained. Then, the approach known as Spatialization is discussed. The article focuses on the meaning and the types of the so called Level Of Details (LODs) as well as on the caching and prefetching using the real interactive visualization system, called iWalk for illustrations. In the end some additional techniques for graphics pipeline optimization are mentioned.*
***Key words:** large dataset, out-of-core visualization, spatial data structures, model simplification, level of detail, cashing and prefetching.*

1. INTRODUCTION

To visually explore the data from a dataset there is a need of something more than just generating images or sequences of images. A user wants to interact with graphical information presented on the display using one or more input devices. This interaction need creates a branch in the field of computer science called interactive visualization.

For a visualization to be considered interactive it must satisfy two criteria:

- To has a possibility for user input that gives a control of some aspect of the information being represented;
- To has a response time guaranteed that the system produces a real-time results;

According to Penny Rheingans [1], discusses the "power of interaction", and said that there is a distinction "between dynamic and interactive control". She explains that "with interactive parameter control, the displayed image only updates periodically" but "with dynamic manipulation, the displayed image changes as the viewer moves a continuous input devices, such as a slider, joystick, mouse, or tracker". Because of the latter "the researcher not only sees the initial and final representations, but also the representations in between".

Researchers (or users as a whole) can deal with different type of input data in the visualization process. Depend on the data type the following types of visualization can be distinguished [4]:

Terrain Visualization. Flight simulation can be an example of this visualization technique. Mention that "data may be acquired from satellite imaging" and "with typical sample resolution ranging from one kilometer down to several meters, and current data size up to 993 million samples" [4].

Visualization of 3D scanned models. The process as laser scanning is the main source of this type of data which "provide wonderfully detailed representations of physical objects" [4]. The number of elements depend on the complexity of the object and given examples are "the Stanford bunny model, comprising 70 000 triangles" and "statues from Stanford's Digital Michelangelo project, such as David, at 56 million triangles, and St. Matthew, at 372 million triangles" [4].

Scientific and Medical Visualization. Data in this domain originates in different simulations and 3D medical imaging where we can get 1 million or even 1 billion or more elements.

Computer-Aided Design and Synthetic Environments. These "CAD models may represent relatively simple machine systems, like automobiles, or complex machine systems, such as aircrafts, ships, factories, etc". They exist to "prototype equipment and perform simulations" and is characterized with "high geometry complexity, comprising thousands to millions of individual parts that total millions to billions of primitives" [4].

The advances in modeling tools and simulation techniques have led to generation of 2D, 3D, 4D and even 5D complex geometrical models and respectively very large datasets representing that models.

Having a data we can apply different computer graphics algorithms (using surfaces or volume rendering techniques) for primitive extraction, and after that passing these primitives to graphics hardware for further processing to fulfill the visualization task.

Today's graphics processing units (GPUs) are optimized to rendering more and more primitives. Graphics cards use parallel processors composed of up to several thousands of cores. A particular

graphics card characterized itself with a specific architecture that is a vendor dependent. For example NVIDIAs Maxwell architecture that "delivers incredible performance, unmatched power efficiency, and cutting-edge features" [2] is the base of the GeForceGTX 980 GPU with the main feature as [3]:

- Sixteen Streaming Multiprocessors (SMs).
- Each SNs uses a quadrant-based design with four 32-core CUDA processing blocks, which estimates the numbers of CUDA cores to 2048.
- Standard memory configuration of 4096MB.
- Memory bandwidth of 224.3 GB/sec.
- Support of recent technologies such as OpenGL 4.4, Microsoft DirectX, CUDA, Dynamic super resolution, NVIDIA SLI Ready, NVIDIA G-Sync Ready, and others.

According to this example we can say that GPUs have a vast amount of computational power. For rendering at interactive rates (more than 10 fps., and more than 30 fps. for real-time systems) we need all these computational power but be aware to so called bandwidth requirements. Memory bandwidth parameter is important because it effects how quickly and smoothly the data can be visualized or with other words how qualitative the interactive process can be.

One approach for rendering very large dataset (so large that even can fill the core memory) at interactive rates is by using a faster hardware (parallel configurations such as computer clusters). This is "expensive and often has limited scalability" [4]. The other possibility is to use a more "cost-effective way" [4], that consists of "sophisticated approach at the software level" [4]. The latter approach is a must if we want to use a commodity PC for a visualization processing a so called "big data", gathering for example with the help of magnetic resonance imaging (MRI) scan technique, which could produce detailed picture of the inside of the human body.

2. LARGE DATASETS IN THE VISUALIZATION FIELD

The term "big data" is dated back in the far 1997, where scientist from NASA in their paper defined it [7] and explained the problem this term involves: "data sets are generally quite large, taxing the capacity of main memory, local disk, and even remote disk". The authors conclude that "when datasets do not fit in main memory (in core)", or "do not fit even on local disk", the common solutions/algorithms have drawbacks and that we must find some other decisions. So they proposed that the problem must be solved with "out-of-core visualization" "when a single data set is larger than the capacity of main memory", and with "remote out-of-core visualization" "when a single dataset is larger than the capacity of local memory and disk".

Almost ten years later in 2005, Charles Hansen and Chris Johnson editing The Visualization Handbook [6] "have tried to compile a thorough overview" of the visualization field "by presenting the basic concepts of visualization providing a snapshot of current visualization software systems, and examining research topics that are advancing the field". The book consists of many authors' materials, and in one of them, Desktop Delivery: Access to Large Datasets, Philip Heermann and Constantine Pavlakos from Sandia National Laboratories give almost the same definition of the term "big data" as this from 1997 saying that "large data will be defined as datasets that are much greater than the memory capacity of the desktop machine". They propose more formally description of this definition (1)

$$D >> 10MD \quad (1)$$

Where D is the dataset of interest and MD is the random access memory (RAM) of the desktop machine. They mention that for machines "with two to four gigabytes (GB) of system memory, a large dataset would be hundreds of gigabytes to terabytes and perhaps even petabytes".

3. INTERACTIVE VISUALIZATION APPROACHES

3.1. Out-of-core processing

Out of core algorithms are also known as external or secondary-memory algorithms. They must be efficient in the process of management of large dataset.

General approach to handle dataset larger than main memory is "to break dataset into manageable pieces, and bring the appropriate level of detail (LOD) of each piece of the dataset into memory on demand" [8]. All these steps construct so called "out-of-core hierarchical representation for the model at preprocessing time" [9], and at runtime we can "load on demand" these hierarch nodes [9].

Terms as *spatialization, LODs,* and *cashing and prefetching* are used in out-of core techniques to explain respectively the process of breaking the dataset into pieces (constructing spatial data structure), to create an appropriate level of details (LODs) from the model, and to manage what pieces come in and out of the memory.

3.2. Spatialization of large datasets

Bulk loading is known as the process constructing the out-of-core spatial data structures. The overview of the use of spatial data structure is given at [10]. The main focus of the article is on hierarchical data structures, including "a number of variants of quadtrees, which sort the data with respects to the space occupied by it" [10]. Techniques that are used are known as spatial indexing methods. Because of these methods we have a data structure designed to unable fast access to the working data. Working (spatial) data "consists of spatial objects made up of points, lines, regions, rectangles, surfaces, volumes,

and even data of higher dimension which includes time" [10].

Many different kinds of spatial data structures: octree, kd-tree, BSP tree, R-trees, hierarchy of boxes, hierarchy of spheres can be used [10, 12, 13]. Some of them may work in a system that produces "accurate images of large datasets at high frame rate" or so called real-time system [11].

In [8] are given an examples for spatial data structure usage "in many commercial and academic graphics systems". The author says that octrees are used "in innumerable contexts, including view-frustum culling, occlusion culling, ray tracing, and volume rendering. SGI's optimizer uses a hierarchy of boxes to spatialize the scene graph. Id Software's Quake 3 game uses a BSP tree. The QSplat system uses a hierarchy of spheres".

3.3. Model Simplification

The idea of model simplification results in hierarchical-model representations. The process produces a hierarchy from a complex model. Using this hierarchical representation during the interactive rendering process the user or the application can dynamically balance the need for interactivity against the need for an image quality.

The first hierarchical approach dated back in 1976, when James Clark published his article [5]. Doing his research concerning visible surface algorithms he concludes that we can only benefit if we structure the environment being rendered with "single, unified, structural approach". To explain this approach he uses an example of a human body model and analogy of a, as he said, "traditional motion structure used to position objects relative to the "world" and subobjects relative to objects" (fig.1). When we view the body from a very large distance, it covers only 3 to 4 display raster units, and it is sufficient to model the body with a rectangular polyhedron with appropriate color. That is why we can use the "uppermost node, or "object", for this body to represents this simple description". Otherwise, if the body is viewed from a closer distance (covers 16 raster units) the topmost description is no longer sufficient, and "the next level of more refined description is needed". At this next level the body description might be a collection of appropriately attached to each other rectangular polyhedral (for each for the arms and legs, the head and the torso). We can continue with the process to "whatever maximum level of detail will be needed".

According to Clark the body described could be an object from a large environment and that the "significant point" worth mention "is that in a complex environment, the amount of information presented about the various objects in the environment varies according to the fraction of the field of view occupied by these objects". Having structural representation Clark asks the following question: "how does one select only that portion of a potentially very large hierarchy that is meaningful in the context of the viewpoint and the resolution of the device", or as he explain "this implies finding the visible nodes of the tree" (fig. 2). This is said to be "clipping operation" and we could perform it with so called "clipping algorithm which recursively descends the tree", with the help of "some minimal description of object sizes". The example given uses "a boundary rectangular box or a boundary sphere" to "test whether an object is totally within or totally outside of the field of view".

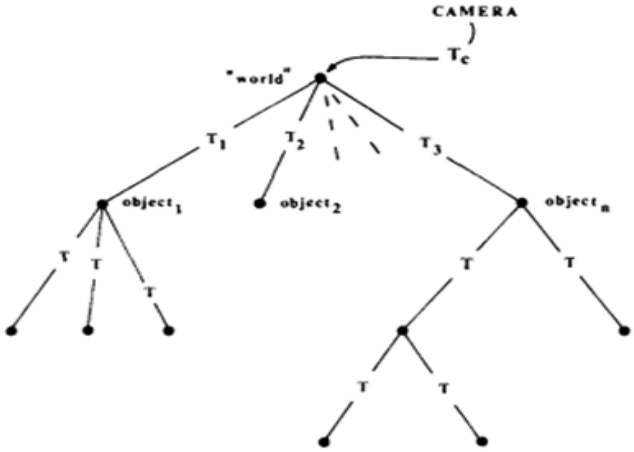

Fig.1: The traditional motion structure used to position objects relative to the "world" and subobjects relative to objects. Each arc in the graph represents a transformation [5].

The author uses the term "working set" for the data that is stored in the fast core memory of the computer system. So the working set in the context of fig. 2 "is that set of objects in the hierarchy that are "near" to the field of view, inside it, or "near" to the resolution of the image space". According to the interactivity if the speed is not high, difference between one scene and the next are usually small and the working set will change slowly. On the other hand, the high speed differences are large, working set will change fast and the scene could be rendered with less detail.

Jonathan Cohen and Dinesh Manocha in their work Model Simplification, conclude that hierarchies could be classified as discrete, continuous, or view-dependent [6]. The three hierarchy types are graphically represented at fig.3. There is one more hierarchy type, shown on the far right part of the fig. 3, named Hierarchical Level Of Detail (HLOD). It is similar to the View-Dependent hierarchy type, but with much more levels and according the authors "allowing faster management and more efficient rendering".

According to the authors, the Discrete type is the simplest hierarchy type and most common form of the hierarchy. The original model can be encoded by multiple levels of detail (LODs). We can choose this LODs in a way that "each successive LOD has half the complexity of its predecessor". This is going to double the storage requirements of the original model. There are two benefits of this hierarchy type. First, "each level of detail may be easily compiled into an optimized form for efficient rendering". Second, "the

management of one or more discrete hierarchies within the interactive application is not too computationally expensive". Discrete hierarchy could have a practical use in virtual environments, 3D scanned objects, and some CAD environments.

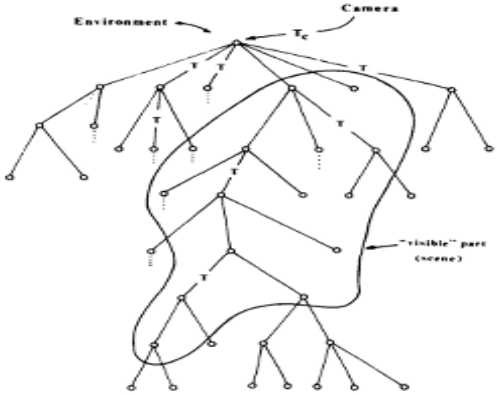

Fig.2: A very deep hierarchy that structure the environment much more than the traditional motion structure. [5]

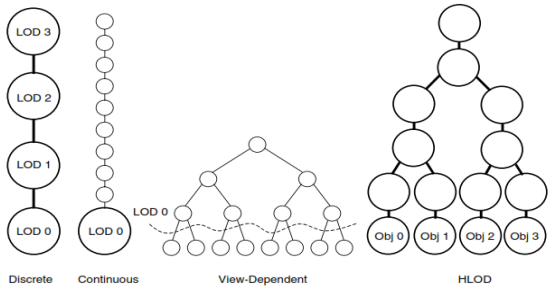

Fig. 3: Simplification hierarchies with different degrees of control [4]

Continuous type has many levels of detail. This hierarchy is "well suited to store the individual small changes to the data rather than storing each level of detail as a complete, stand-alone model". The hierarchy could be useful "for the same class of models as the discrete hierarchy" in particular virtual environments, 3D scanned objects, and some CAD environments.

View-Dependent hierarchy according to the authors is "the richest and most complex type". We could represent this kind of hierarchy by a tree data structure or a directed acyclic graph. This form of model simplification is typically used in terrain visualization, large isosurface visualization, and visualization of some large-scale CAD structures.

Hierarchical Level Of Detail (HLOD) could be used for complex scenes with multiple objects. The important thing here is that "each of these objects may be represented as a discrete, continuous, or view-dependent hierarchy". A scene-graph is an appropriate structure for a representation of this type of hierarchy, and "each leaf nodes is the finest resolution representation of each of the individual objects". Some discrete LODs are stored for each object, and "these objects are eventually merged together". The authors explain that "this type of hierarchy works well with occlusion culling algorithms and has been used for interactive walkthrough of large and complex CAD environments".

3.4. Caching and prefetching

To render a model larger than main memory, the visualization system keeps on disk a spatial data structure (octree for example) that is a representation for a model. The contents of the octree nodes are then loaded in the cache memory on demand. As is said in [8] cashing alone is not enough to deliver smooth frame rates. The explanation said that "even small changes in visibility may cause the system to stall because of bursts of disk activity" [8]. One technique to minimize this problem is called speculative prefetching, "which tries to bring into memory the pieces of geometry that will become visible soon" [8].

The place of geometry caching and prefetching in the rendering process is illustrated on fig. 4. The sane figure shows the multi-traded out-of-core rendering approach used in the iWalk system [9].

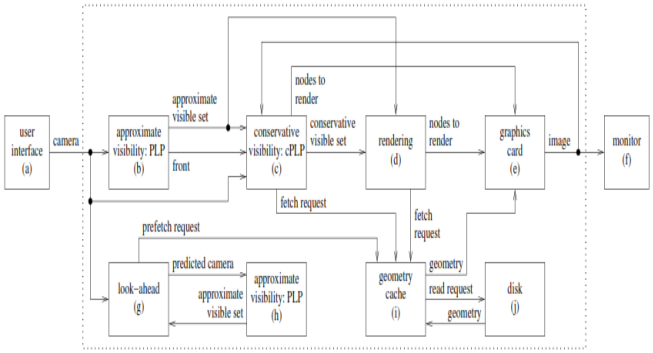

Fig.4 The multi-threaded out-of-core rendering approach of the iWalk system [9]

The procedure represented on fig. 4 can be described briefly as follows [9]. "For each new camera (a), the system finds the set of visible nodes using either approximate visibility (b), or conservative visibility (c). For each visible node, the rendering thread (d) sends a fetch requests to the geometry cache (i), and then sends the node to the graphics card (e). The look-ahead thread (g) predicts future cameras, estimates the nodes that the user would see then (h), and sends prefetch requests to the geometry cache (i)".

4. OPTIMIZING RENDERING PROCESS

In addition to the approaches explained above (spatialization, model simplification and geometry cash and prefetching) there is a large number of rendering techniques aiming to optimize rendering process. Some of these techniques include:

- Back-face culling – not rendering geometry that faces away from the user;

- View-frustum culling – not rendering geometry that is outside the field of view of the user's camera;
- Occlusion culling – not rendering geometry hidden by other geometry, or in other words, only rendering the geometry that is visible;
- Hardware assisted rendering – the possibility to implement some graphics algorithms in the GPU, using so called API (Application programming interface) such as OpenGL or DirectX.

Detailed analysis of the listed above techniques is outside of the scope of the article, but all of them are closely related to the achievement of the interactive rates of visualization.

5. CALCULATION

Technical advances in both fields - hardware and software – have led to more and more sophisticated results in computer graphics and visualization. We have data that are collected from new advanced technologies in medicine, engineering, entertainment and etc., so that is way the term "big data" or "large dataset" is common in nowadays.

Visualization is a process that can give an answer to the following questions: 1) How to explain data in a purpose to solve some specific problems, and 2) How to explore large dataset for a purpose of better understanding. Explanation and exploration are among the most important activities for engineers, researchers, medics and other professional groups so we can say that the need of visualization is growing.
Interactive visualization helps different aspects of the data to be found. The interactivity gives a two way communication between the system and the user. With the help of an input devices user can control the visualization process or with other words he/she can modify the displayed image in an appropriate way. Important parameter for the quality produced by the visualization system is so called interactive rendering rates. Higher rendering rates guarantee quick system response essential for some data to be analyzed and understand in an appropriate way.

Dealing with large dataset and achieving interactivity need special approaches. Using the right spatial data structure helps the "right" data to be extracted and the search for specific data to be optimized. Appropriate level of details guarantee us that we can deal with smaller dataset. Cashing and prefetching allow us to work with data that are only visible from a given view-point or are likely to be visible very soon.

Working with a part of the dataset not with a whole one increase the rendering frame rate and achieve better interactivity so the explanation and exploration of the data can be much more useful and effective.

ACKNOWLEDGEMENT

The present article is a part of a contract №142ПД0037-09 - Environment and tools investigation helping for data visualization.

REFERENCES:

[1] Penny Rheingans, Visualization Viewpoints, IEEE, 2002
[2] http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-980
[3] http://devblogs.nvidia.com/parallelforall/maxwell-most-advanced-cuda-gpu-ever-made
[4] Jonathan D. Cohen, Dinesh Manocha, Model Simplification, The Visualization Handbook, 2002.
[5] James Clark, Hierarchical Geometric Models for Visible Surface Algorithms, Communication of the ACM, 1976.
[6] Christopher Jonson, Charles Hansen (editors), The Visualization Handbook, Elsevier Inc., 2005.
[7] Michael Cox, David Ellsworth, Application-Controlled Demand Paging for Out-Of-Core Visualization, Report NAS-97-010, July 1997.
[8] Wagner Toledo Correa, New Techniques for Out-Of-Core Visualization of Large Datasets, a Phd. Dissertation of Princeton University, 2004.
[9] Wagner Correa, James Klosowski, Claudio Silva, iWalk: Interactive Out-Of-Core Rendering of Large Models, CiteSeerx, 2002.
[10] Hanan Samet, Spatial Data Structures, CiteSeerx, 1995.
[11] Mark Duchaineau and al, ROAMing Terrain: Real-time Optimally Adapting Meshes, IEEE, 1997.
[12] Pankaj Agarwal and al., A Framework for Index, Bulk Loading and Dynamization, Springer, 2001.
[13] Lars Arge and al., Efficient Bulk Operations on Dynamic R-trees, ASM, 1999.

About the authors:

Ani Dimitrova, Assist. Prof. FKSU, Technival University, E-mail: adimitrova@tu-sofia.bg, Sofia, Bulgaria.