UDC 004.623:004.922 (043.2)

Glazok O.M., c.t.s.; Khodchenko F.S. (National Aviation University, Ukraine, Kyiv)

AN ALGORITHM FOR INFORMATION PROCESSING IN IMPLEMENTATION OF LANDSCAPE DATA DYNAMIC LOADING

The use of Continious Level of Detail (CLoD) approach for visualization of flight over the huge landscape is considered. Foe the tasks of visualization of aircraft flight, it is necessary to provide acceptable performance at high speed of movement. In order to achieve this goal, the algorithm of phased processing of chunks of the landscape is offered.

Multiresolution modeling, or multiresolution visualization, is a process that allows for adjusting the level of detail (LoD) of the presented scene, while maintaining a constant (or at least appropriate) frame rate and assuring interactivity to the user [1]. Multiresolution visualization has become a matter of interest for the last dozen of years, with development of various real-time applications, such as computer and video games, virtual reality and scientific simulation. As these applications require rendering of complex models for realism, graphics rendering engines include multiresolution modeling techniques, which have become widely used.

The multiresolution modeling techniques presented in the literature are classified by the criterion of the two main approaches to managing level of detail (LoD): Discrete LoD (DLoD) and Continuous LoD (CLoD). The DLoD approach manages a small number of independent levels of detail (LoDs), where each approximation or LoD represents the original object using a different number of faces. CLoD is introduced as an alternative which provides a wide range (virtually a continuous range [2]) of different approximations, such that the LoD can be adapted to the application requirements with a high degree of accuracy. CLoD has been extended to provide view-dependent LoDs, which is sometimes considered as a third approach.

There are some substantial differences between the two approaches (CloD and DLoD) on the stages of model construction, data loading, and run-time rendering. Let us consider them in the application to a practical problem – to perform a visualization of the global landscape with the view from the unmanned aerial vehicle (UAV). Based on the scales and detalization, this problem may be classified as the problem of loading and rendering of the “huge terrain”. The “huge terrain” is the terrain that is many times greater that its part that can be viewed from any point of view, available for the system, and has the amount of information many times greater that the available space in the operative memory of the computer system.

It is quite obvious that for loading and rendering of a huge terrain its data should be divided into parts (chunks). Complete loading of the terrain is not possible due to the memory space and time limitations. At the same time, if one tries to process the whole data set, the speed of processing will be too low for real-time interactivity. The fact that such terrains are always seen partially also leads to the

idea of partial processing. Such an idea got the name Chunked Level of Detail [3], also abbreviated as CLoD, and is a subclass of LOD [2].

Basically, the landscape is divided into the rectangular (or square) parts. This promotes coordination and visibility determination operations.

After the division, the algorithm of visible parts determination should be applied. The simplest visibility criterion is distance. In this case, one should define maximal distance from which an object can be visible. In case of flight visualization, it is possible also to chop off the parts which do not get in the angle of visibility. Because any flying object can turn only with limited angular velocity, there is no sense to load memory objects located behind that angle (except for the necessity of reflection of back view). As a softer variant, it is possible to decrease distance of visibility for the objects that lay outside of the angle of visibility.

At flight visualization, the higher the flying vehicle is, the more distant the horizon line. In other words, the radius of visibility increases with the height increase. At large values of heights, there may be necessity to render quite a large piece of terrain. At the same time, distant parts will have little angular sizes and, therefore, their exact rendering has no sense. As a consequence, one should introduce and apply a special factor for the level of detalization of parts. This factor should also be taken into account as a correction factor at visibility determination.

There may be two variants of algorithm of detalization change. The first one requires that every element of a landscape had several models (resources) of different detalization levels. The models should be loaded as needed. The second way is creation of a certain quantity of separate elements in every part of a landscape separate some number of separate detalizing elements. Every element will be represented only at the certain range of detalization levels (while a level of detalization is given as a numerical factor). It is possible also to unite these elements hierarchically. The first method requires more memory, while the second method is more difficult in realization. There is also a possibility to combine these two methods.

The data loading algorithm is important factor of system productivity. Every resource before its rendering passes the two stages: downloading from hard disk into the main memory, loading and construction from the main memory into the videomemory. Also, in the process of load, can be present information cashing (placing it into a temporal storage), which will accelerate downloading from a hard disk. For example, cache may help to avoid the stage of recoding of an image, or to hold a necessary file in the pre-loaded state). Cache operations may be considered as the third stage.

It is possible to pass all of the stages at once, as soon as there arises a necessity to display an object. However, such an approach will substantially complicate loading and movement on a map; in many cases it will lead to place jerks (usually called “popups”). It is possible to enter the factor of safety, connected with the object sizes, and load less critical information in a separate stream (with lower priority), then the elements of a map will be loaded beforehand, as possible. But efficiency of this approach will be small, if there is a lot of parts and information which must be loaded, which is just the case for the flight visualization problem.

The second approach is to pass the stages beforehand and gradually. This approach is referred to as phase processing [4, 5]. In this approach it is needed to enter the two factors: the main memory radius and the cache radius. These radiuses are numerical characteristics that determine the necessity of specific pieces of data loading. At this approach, each chunk of data is classified as the chunk belonging to one of the three areas (Fig.1). All the data that are outside the radius of cashing should be kept on a hard disk as basic data. All the data that gets inside the radius of cashing should be processed: the data that come from outside the system should be buffered in memory; the data that come from within should be written down in a hard disk cache. Such approach allows to promote the productivity and smoothness of loading, but at the same time it increases the expense of memory.

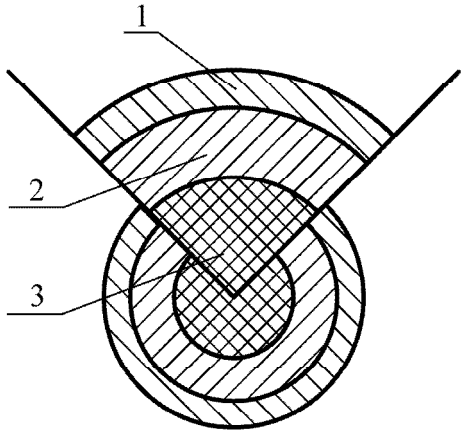

Fig. 1. The three areas of data chunks location. 1 – cache area; 2 – memory area; 3 – visible area.

Let us consider an example design of the system on the basis of all considered before. For the sake of simplicity, we will suppose that the information to load is the bitmap images of a map. Let us use the sequential (stage-by-stage) algorithm of loading, and limit detalization to the multilevel model detalization.

The program keeps metadata about every part of a map. The metadata include the size, position and reference (path, name) to the file with the loaded information. Consequently, we should have a "Map element" object, which will keep metadata. Because the subsystem of the map resources of card will be engaged in loading, metadata will contain references to the "Resource unit" object, in which the information necessary for loading will be kept.

The elements of landscape may be kept in a simple list. There is no sense to create a hierarchy. Using a list allows to perform quick check for visibility of the stored elements.

The resources subsystem consists of the list of resource elements. The "Resource unit" object keeps the unit state (current stage of processing) and pointers to the "Resource" objects for every stage, such as "File resource", "Cache resource", "Memory resource" and "Visible resource". The "File resource" object exists permanently and stores the path to the necessary file. The other objects can be created and deleted during work. Each object includes a function of creation of an

object of the next stage. At transition to the next level, unnecessary objects are deleted, and corresponding pointers get the null values.

At each iteration of the cycle all parts of the landscape are checked for belonging to the cache, memory and visibility areas. Depending on the area found, the proper processing stage and the level of detalization are determined for each piece of terrain data.

The procedure of determination of belonging to an area is done as following: at first each vertex is checked for presence in the view angle (using the scalar product of the vector of direction of a camera and the vertex vector). According to the results, the verification radiuses of visible and invisible areas are corrected. Here, the radiuses for visible areas may be enlarged and the radiuses for invisible areas may be made smaller if necessary. Then the distance from a camera to the element (rectangle) is calculated. Next, the obtained distance is compared with the verification values of the areas. At this stage the level of detalization may be determined.

Conclusions

The base algorithm of data processing for flight visualization over a huge landscape has been proposed. It is further possible to perfect and optimize them for minimization of expenses of memory and increase of the productivity. It is possible also to obtain a greater smoothness utilizing multithreading and MPI (Message Passing Interface). Provided the due optimization is done, the proposed approach is capable to provide the high performance of load at high-speed flights. The basic lack of this approach is a high consumption of main memory.

A method can be applied in navigational complexes, visualization software of simulators and research projects. It perfectly fits for parallelizing and work over a network.

References

1. Ribelles J. An Improved Discrete Level of Detail Model Through an Incremental Representation./J. Ribelles, A. Lopez, O. Belmonte. – Proceedings of TPCG. – 2010. – Pp. 59-66.

2. Luebke D. Level of Detail for 3D Graphics /David P. Luebke. – Morgan Kaufmann Publishers, 2003. – 390 p.

3. Thatcher U. Rendering Massive Terrain Using Chunked Level of Detail Control /Ulrich Thatcher. – Режим доступа: http://tulrich.com/geekstuff/sig-notes.pdf

4. Lindstrom P. Visualization of large terrains made easy /P. Lindstrom and V. Pascucci. – IEEE Visualization 2001 Proceedings. – Oct 2001. – Pp. 363-370.

5. Hansen C. The Visualization Handbook /Charles D. Hansen, Chris R. Johnson. – Academic Press, 2004. – 984 p.