Volume 2, Issue 7, July 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Dynamic Load of Environment in Rendering of Real Time Simulators of Computer and Computer Games and Applying PVS Algorithm

Mohammad Ali Haierian, Dr Mohsen Najafi
Department of computer Engineering,
Arak Branch, Islamic Azad University,
Arak, Iran

*Abstract: **In real time renderers, one of the common problems is loading of textures with high quality applied in environment .in this paper, Compression methods and also mega texture have been explained in summary and their advantages and disadvantages will be inspected .then one method for estimation of loading time and unloading of textures based on PVS algorithms that its application is in scene management has been presented.***

Keywords: mega texture , texture comparison, real time rendering, PVS, load time of texture, efficient use of video memory.

I. Introduction

Due to the recent development in hardware of graphical processors and video RAM of table computers and console games, complexity of scenes and 3D spaces also have improved simultaneously .Modern graphical usable programs that have related to real time rendering, are dependent on new algorithms and techniques for efficient use of these hardware facilities. Such needs that new generation of these programs face, will be the fact that how we can use textures with high quality for rendering of more accurate and with more quality scenes .As we know in the new generation of graphical cards, these sets have separate video RAM for their processors, one of the recent challenges in this industry is how to use space of this graphical RAM for loading the textures. In this paper after defining the rendering process and PVS algorithm, at first, we take inspect to Compression of textures and transmitting them to video RAM and explain their merits and problems and in the next step, we express modern mega texture techniques and emphasize its problems and merits, and in the last step we pose a new idea for estimation of the load time of textures with PVS algorithm.

II. PVS technique

Calculating the potentially visible set is not often achievable. Limited resolution and the question of sample point count and placement are just few of the problems. Over the years many various techniques have been developed to perform this duty. These so solutions can be loosely categorized into those applied either run-time of during a pre-process. With a few exceptions, the former determine the visibility from a single point, while the latter establish the subset of geometry visible from any point within a area. *From-region* visibility partitions the view-point space into area of cells, rather than trying to preprocess the infinite number of possible camera positions. The principal advantage of these techniques is that their considerable computation cost can be shifted to a pre-process, consequently removing feat runtime visibility computation. The results may then be saved to disk for future use.[1] The disadvantage is that, as a pre-process , only static scenes can be fully treated. *From-point* visibility algorithms are less costly computationally than from-region approaches, allowing them to be applied on a per-frame basis at run-time. They are also better suited to unpredictably dynamic environments, where the share or position of objects may change between frames.[2] Occlusion culling algorithms may be further categorized according to their accuracy in differentiating between visible and hidden polygons. Nirenstein et al. [3] compare algorithms by image quality (correct or containing errors) and run-time performance (optimal or sub-optimal) and discriminate between *conservative*, *aggressive*, *approximate* and *exact* visibility algorithms.

Conservative techniques consistently overestimate visibility and incur a wrong visibility error: hidden polygons are considered visible**.** This naturally gives correct images but results in sub-optimal run-time performance because some polygons are submitted to rendering pipeline. In decent conservative algorithms the conservatively should be no more than 30%.[4]

Aggressive methods always underestimate the set of visible geometry and exhibit wrong invisibility, where visible polygons are erroneously excluded. Aggressive visibility causes image error but can be useful if the perceptual impact of the error is acceptably small or the algorithm handles scenes that cannot be solved effectively with conservative alternative due to excessive overestimation. However, in practice even rather small image errors are usually quite mentionable and therefore unacceptable.[5]

Approximate visibility techniques give both wrong visibility and wrong invisibility errors. They are so useful usually only when the pre-processing efficiency is the overriding concern.

Exact visibility solutions provide both accurate images and optimal rendering performance. An exact visibility query will produce no more or less than the union of the polygons visible from all viewpoints within the area.
There exist many research of various computer graphics occlusion culling techniques and algorithms. An interested reader may look for example Zhang [6], Saona- Vasquez et al. [7], Durand [8], Cohen-Or et al. [9] and Bittner. [10]

REAL TIME RENDERING

Rendering is a process that we can make a model of picture out of it. In this process specific software (and often hardware) is intervening .this model includes geometric features, texture , lighting and overcasting shading. One rendered image of this model can be produced with applying features like lighting, shading, fog, mapping of texture or bump and other features. Also for more attractiveness of effects of post-process like ignition the light, effects related to lenses of camera like rings and depth of fields are added to ultimate produced image. Modern applicable graphical programs like computer games and 3d simulations that are using 3d real time rendering, at the time of rendering maybe face with so many data .one of the efficient methods for accelerating to render process and reducing this load is that preventing from spiritual rendering that is not being visible from viewpoint . These kinds of algorithms are named culling. And as a matter of fact , this algorithm only detect visible objects and spaces and convey to render pipeline .[11]
During years, different techniques for doing this have been developed that PVS algorithm is one of them. In spite of recent brilliant Improvement in graphical processors and the capacity of video RAM applied in graphical hardware such as graphical cards with several processors cores and with high frequency and also one or two gigabyte RAM for special home systems, but the complexity of scenes and 3d spaces also simultaneously , with these improvement have increased .for example loading textures and light mappings with common quality of 512 in 512 and color depth of 32 bit, for a outdoor space is considered enormous challenge .Because if we don't use Compression techniques and dynamic loading of these textures ,in a rendering system with common 1 gigabyte video RAM ,only 600 textures situate in memory(and remains of memory will be filled with information of geometric features and different kind of buffers and back buffers) that it does not seem enough for the present generation. So what is the solution? In following number of efficient use solution from video memory will been expressing.

COMPRESSION OF TEXTURES AND THEN CONVEYING THEM TO VIDEO MEMORY

Base on this fact that about 80 percent (this percentage in different games and different simulation application might be different) of the space of video memory has been captured, Compression these textures will help in optimum usage from graphical memory. With considering the limitation in hardware rasterizer, a technique that is applied from Microsoft and Open GL.13 and still is in progress is s3tx Compression technique based on the color cell that has been presented in year 1970.Opposit of most of the image Compression techniques, this technique make a constant rate of Compression and is considered a image Compression. the reason that causes this technique to be appropriate for 3D graphical hardware accelerators is that the Compression process has a constant rate and it needs only achieving memory for one time .in MS DirectX, the format of used texture is known as DXTN that based on the amount of n ,amount of Compression from one to six or one to four is various. Also in OpenGL, there is specific suffix for s3tx Compression process. In spite of this fact that using this Compression technique both in bandwidth saving and the rate of captured memory, mentionable economization will be achieved. But based on this reason that this Compression has some waste, the quality of the ultimate image will been decreasing and limiting the usage of compressed textures .For example using this Compression in normal maps causes making artifacts in rendered image or in another example with Compression the images that are cartoon like or have been drawing with hand, there will be a great downfall in a quality of ultimate rendered picture that is not desirable. For handling this problem, we need to increase the dimension of the picture that again causes increasing memory needed for saving in video memory.

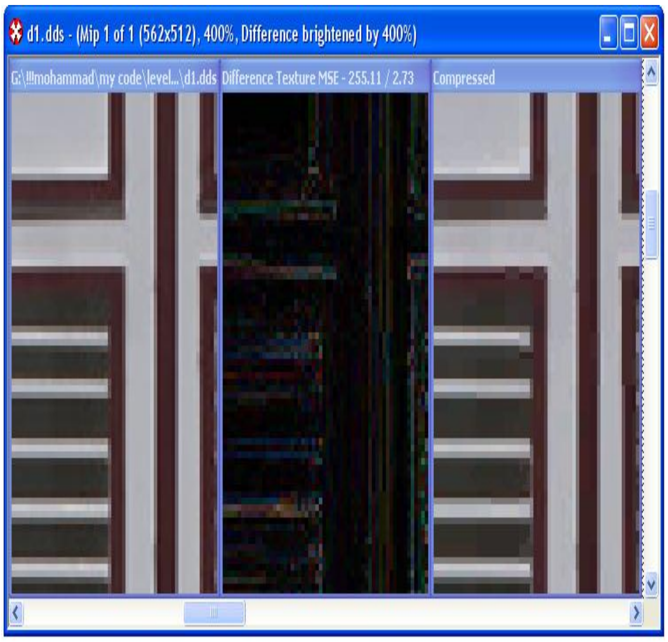
d1.dds - (Mip 1 of 1 (562x512), 400%, Difference brightened by 400%)
G:\!!!mohammad\my code\level...\d1.dds
Difference Texture MSE - 255.11 / 2.73
Compressed

Fig. 1: In above picture, original texture is located in left .in right part, the compression of the same texture with DXT1 is located .As it is clear, the quality of texture has declined .In middle part, the difference between 2 picture can be visible by multiply 4.This picture is supplied from the Compressor 1.50 software .

MEGA TEXTURE

One of the modern techniques used in real time rendering of computer games and 3D simulation is one that is known mega texture or virtual texture or as sdk MSDX9 has been said, it is known content streaming. In fact when the resources(textures) in 3D space is so much big that is

not situated in graphical memory or even in system memory, one way is to decrease the size of texture .But this is not always the best solution. Sometimes we want to have textures in high numbers in space and high quality. With using content streaming, we only load the textures that are now visible without interfering in renderer program (means the texture streaming process will be done in background).[12]
In summary, this process is being done based on the conceptions like resource reuse cache and memory-mapped IO. Resource reuse cache (RRC) is probating the resources of graphical hardware that are constant numbers .[13]when the program ask for the use of one resource (texture), it simply control loaded resources and if the resource is not among them , it makes new decisions based on the loading or temporary using of low quality texture. In MMI/O, when reading the file is asked for, data are not being removed to memory. In return, file will map over the virtual address of program and if it is necessary, will be paging. [14] In fact instead of reading the data directly from files, one pointer will be introduced to the primary virtual address at the beginning of the file and so has a very high speed.

III. THE PROBLEMS OF MEGA TEXTURE METHOD

In spite of this fact that it seems that content streaming is an ideal solution for the shortage of graphical memory, but this method has its own specific problems and restrictions. The first problem is the rate of captured space over the hard disk that is from 4 gigabyte to 32 gigabyte for vast environments.

Another restriction is in using texture filters like bilinear that it needs 4 times indirect look up and different mip-maps. So 4 times is needed to be looked up in case that increases efficiency. In addition ,based on this fact that updating the textures in scene might still be done with retardation, when the camera move fast ,observer might distinguish that in scene ,part of the texture is changing and quality of it is improving

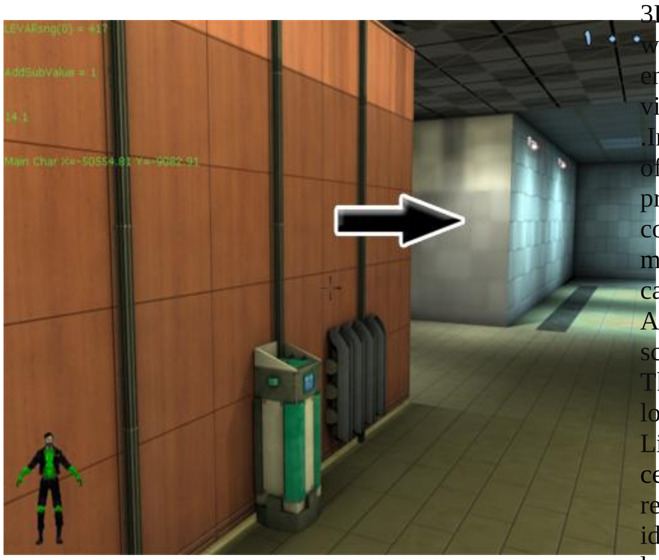

Fig. 2: In this picture, arrow pointed to texture that opposite other textures in the scene, doesn't have sufficient accuracy (Resolution).

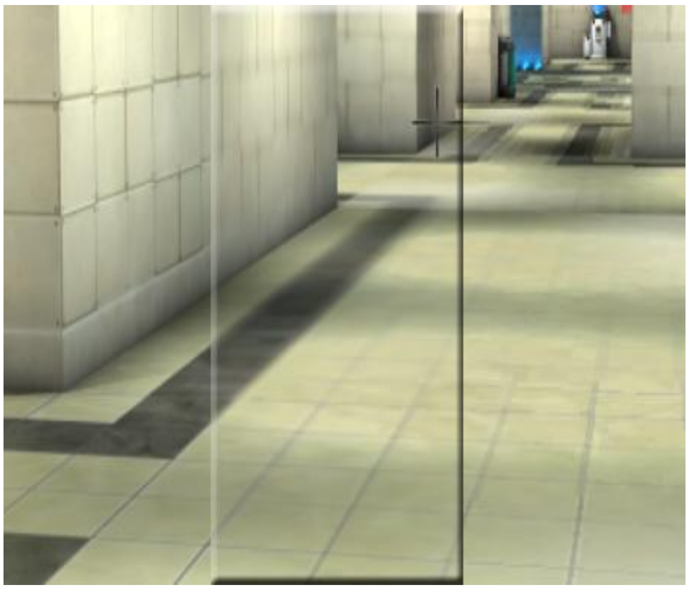

Fig. 3: In this picture that has been taken from environment of computer game, in right part, ANISOTROPIC filter has been used, in middle part, from LINEAR and in right part POINT has been used. As it is clear, filters have a great influence on rendered picture quality

In result, for loading textures, we need method to determine the loading time of the textures that we will likely need them in a near future and in a proper time before the time for using textures, load the textures entirely.

PREDICTING THE LOAD TIME OF TEXTURES

One of the simplest and most essential methods is that based on the distance that these textures have with camera, take consideration loading or omitting them in graphical memory. Even though this method with using 3D ordering algorithms is applicable and performable .But we can be hopeful that just in outdoor environments and environments that have less natural due for blocking the view of camera, this algorithm present better efficiency .In 3D real time simulators , most of the times the control of camera is not directed by user and camera run a predefined path for many times .In return, in this state in computer games and simulators that are in interactive mode ,anticipating the coordination and next spinning of camera is not simply possible.
As we know, PVS algorithm is usable in OC and in the scene management of renderer engines is being applied. The idea that we can use this algorithm for anticipating load time of the specific texture is said in following:
Like the principle of PVS algorithm, we should Classify cells but instead of saving in any cell, the information related to visible polygon, just save information related to identification of visible textures that we want to load/unload dynamically. At the time of rendering, based on the distance between camera and determined cell, we load or unload textures existed in memory.

I PRACTICAL EXAMPLE

In following, camera passes the A, B, C , … cells in order. Cells that are signed , are the ones that texture of star is visible from them, and the texture of star has to be loaded before entering these areas. Using this applied technique that it is exactly specified that which time and which location texture has to be loaded. For example, loading process can be performed when the camera is in "I" cell and has 2 cells far from the area that texture of star is visible. Choosing these 2 cells, distance is dependent on factors like speed of camera, the rate of frame, volume and numbers of textures and etc, to ensure us before entering signed cells(the same cells that texture of star is visible form them) loading of texture has been done entirely.

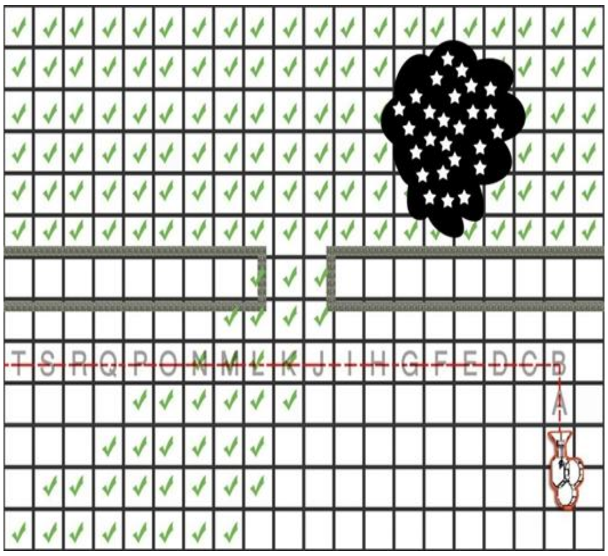

Fig. 4: In above figure, in cell classified environment, star texture is put in the top of the picture at right hand and camera is put in the bottom at the right hand and paces cells A B C …T. Signed cells are the cells that star texture is visible from them.

Based on this fact that the path of movement of camera is been specified, the mission of unloading texture of star from memory can be performed exactly after reaching camera to the cell O.

We can add that if this technique is not used and the simple technique (distance of texture from camera) is used, in the above example at the beginning of camera movement, texture might be loaded .This example showed that efficiency of this simple method is low. As it mentioned ,In 3D interactive simulation and the engines of video game , possibility of guessing the next situation of camera is not simply achievable .A simple idea but efficient is that all visible textures in cells around the cell that camera is located in , are loaded. In other words, new textures which are in cells around the present cell with radius R , are loaded. And other textures which far from this adjacent R , are unloaded. Problem of this method is if we expand adjacent radius , we need organism to load textures near this adjacent radius sooner than other textures .in following, an algorithm has been represented to perform this arranging for adjacent radius [max-dist] for all cells calculated in PVS algorithm and ultimately, one sorted array from distance of visible textures around PVS cells is returned .algorithm of arranging visible textures with maximum distance [max-dist]in process is expressed as following:

```
1-For all existing cells in PVS algorithm, do following steps:
A-For adjacent radius from zero to [max-dist], do following steps
A1-Find existed cells in current adjacent radius and call them $C_1$ to $C_n$ . We have to add that N is number of cells existed in this adjacent.
A2-For $C_1$ to $C_n$ , if this cell is visible from the selected cell at first phase, do following steps:
A2I- Add existed textures in cell $C_1$ to Alltex array
A3-Remove repeated ones from Alltex array
A4- Arrange texture existed in Alltex array based on the distance of these textures from chosen cells in step 1.
A5- Copy existed textures in Alltex array in outtex based on the same arrangement in Alltex array, if they are not existed in outtex array.
A6-Empty Alltex array
B-Print outtex array for the chosen cell in step1
C-Empty outtex array
```

Briefly, above algorithm for all existed cells, starts from the nearest adjacent cell and accumulate textures based on the remoteness and nearness and also not being repeated in outtex array.

A real time renderer engine can use this arranged array. After obtaining this arranged array, engine has to decide how much of this array will be loaded that this decision depends on factors like the rate of frame, remaining video memory and etc. Nevertheless we can be sure that any cell that camera is in, arranged list of textures that in near future might be observed, relegate to engine.

IV. CONCLUSION

Compression of textures methods and also mega texture methods, each technique has its own advantages and disadvantages .Advantages and disadvantages that limit using them .in this paper, it is advised that before needing of use of one texture in renderer, texture is loaded slowly and before process. For estimation exact time that we need to use texture, it can be predicted by some changes in PVS algorithm that is a management scene algorithm. In this paper, we can obtain one array, helping cells from PVS algorithm , those which have been processed before and also with helping our invented algorithm for existed cells in environment. In this array, there is a list of textures that are visible from near the present cell and with getting far from cells of these textures, array stands at the end of the list.

REFERENCES

[1] Andujar, C., Saona-Vasquez, C., Navazo, I. and Brunet, P. 2000. Integrating Occlusion Culling and Levels of Detail Through Hardly-Visible Sets. Computer Graphics Forum, 19(3):499-506, 2000.

[2] Bittner, J., Wonka, P. and Wimmer, M. 2001. Visibility Preprocessing for Urban Scenes Using Line

Space Subdivision. , pages 276-284. IEEE, October 2001. ISBN 0-7695-1227-5.

[3] udarsky, O. and Gotsman, G. 1999. Dynamic Scene Occlusion Cullingô÷ þ ö , 5(1): 13-29. IEEE Computer Society, 1999..

[4] Schaufler, G., Dorsey, J., Decoret, X. and Sillion F. 2000. Conservative Volumetric Visibility with Occluder Fusion._ï , pages 229-238, July2000. ISBN 1-58113-208-5.

[5] Cohen-Or, D., Fibich, G., Halperin, D. and Zadicario, E. 1998. Conservative Visibility and Strong Occlusion for Viewspace Partitioning of Densely Occluded Scenes. ÷ 17(3):243-254, 1998.

[6] Zhang, H. 1998. Effective Occlusion Culling for the Interactive Display of Arbitrary Models, University of North Carolina at Chapel Hill, 1998..

[7] Saona-Vasquez, C., Navazo, I. and Brunet, P. 1999. The Visibility Octree: a Data Structure for 3D Navigation, 23(5):635-643, October 1999. ISSN 0097-8493.

[8] Durand, F. 1999_ 3D Visibility, Analysis and Applications. U. Joseph Fourier, 1999 #,

. [9] Cohen-Or, D., Chrysanthou, Y., Silva, C. and Durand, F. 2001ä A Survey of Visibility for Walkthrough Applications. August 2001.

[10] Bittner, J. 2002Ú Hierarchical Techniques for Visibility Computations. Department of Computer Science and Engineering. Czech Technical University in Prague. October 2002.

[11] Mark Segal and Kurt Akeley. The opengl graphics system: A specification (versión 2.1). Technical report, 2006.

[12] Advanced Topics in Virtual Garment Simulation Part 1 B Thomaszewski, M Wacker, W Straßer in Simulation (2007

[13] T Advanced Messaging Topics Web Services, Reliable Messaging in The Definitive Guide to SOA (2008)

[14] Advanced Topics in FEMLAB Niklas Rom, Ph D in Flux (2005)