Protect Interactive 3D Models via Vertex Shader Programming

Zhigeng Pan[1,2], Shusen Sun[1], Jian Yang[3], and Xiaochao Wei[1]

[1] College of Computer Science, Zhejiang University
310027 Hangzhou, China
{zgpan, sss, weixch}@cad.zju.edu.cn
http://www.cad.zju.edu.cn/vrmm/index.htm
[2] Institute of VR and Multimedia, Hangzhou Dianzi University
310012 Hangzhou, China
[3] Centrality Communications Corp. 201206 Shanghai, China
jyang@centralitycomm.com.cn

Abstract. In 3D games, virtual museum and other interactive environments, 3D modes are commonly used interactively. Many of these models are valuable and require protection from misuse such as unlawful exhibition, vicious distribution etc. A practical solution is to avoid the interactive user to reconstruct precise 3D models from data stream between applications and 3D APIs (such as Direct3D, OpenGL, etc) under condition of not affecting interaction. The scheme proposed in this paper protects 3D modes via vertex shader programming. The data of 3D models are encrypted in 3D application first and then decrypted in vertex shader.

1 Introduction

Today, with fast development of computer in software and hardware, the graphics capabilities of personal computers have achieved great advance. Personal computers have been becoming competent for running 3D games, interactive cruise of 3D scene (VRML), online exhibition of virtual cultural relics [7], entertainment and so on. Under these interactive environments, 3D models are commonly used.

Most of these 3D models need to be protected for various reasons. In 3D games, it will take many artists and lots of financial resources to create 3D models using 3D modelling software such as MAYA, 3DS Max, etc [6]. If the competitive companies acquire these models, the loss is hard to imagine. In addition, some models in virtual museum are created from cultural heritage under contract. For instance, the Stanford Digital Michelangelo Project [7] created many 3D models of large statures of Michelangelo. The contact with the Italian authorities permits these models are distributed to established scholars for noncommercial use. Again, the 3D models recovered and created from the Terra Cotta Warriers and Horses, which represent China's cultural institutions, can

be exhibited [21]. But it is a critical problem to avoid malicious audience's creating accurate physical objects from these 3D models. More examples can be found here [1, 3]. In a word, most of 3D models should be protected for different reasons.

This paper is organized as follows. In next section, the related works are disccused first. Then, Section 3 talks about the factors affecting the security of 3D models. In Section 4, the detail of protecting 3D models via vertex shader programming is presented, followed by Section 5, the system implementation. Conclusions and future work are given in Section 6.

2 Related Works

To protect the 3D models, some solutions are adopted such as image based remote rendering not using 3D APIs [5], 3D model watermarking technology [8, 14–16, 19], matching technology for copyright protection [4].

Ohbuchi introduces 3D watermarking in 1997 [13]. Watermarking techniques for 3D models are faced with many difficulties and challenges [9]. First of all, it is unacceptable for 3D models to import slight distortion under some conditions such as cultural archaeological artifacts. Secondly, the nature of 3D data (no implicit order, not regular sampling, etc) makes it hard to extend 2D analysis methods and watermarking algorithms to 3D. In addition, most of the current 3D models watermarking algorithms are not blind, i.e. it requires the original 3D models to detect watermarks. Under some conditions, it is impossible or impractical to get original 3D models. It is a content-based retrieval (CBR) problem to get original model according to watermarked one. And CBR itself is a challengeable work. Another challenge for 3D watermarking is there are too many attacks for 3D watermarking. So far there is no blind, robust and readable 3D watermarking algorithm.

Ko presented an object-matching methods for the point vs. NURBS surface and the NURBS surface vs. NURBS surface cases [4]. Matching techniques needs to deal with CBR problem as 3D watermarking does. Moreover, many 3D models are not represented by NURBS surface. At SIGGRAPH'2004, Koller etc presented a remote rendering system to protect interactive 3D graphics from reconstruction[5]. To avoid the live-end user to reconstruct 3D models from received information, the snapshots of the rendering results are shown according to users' requirement in client machine using a remote rendering system, which does use 3D APIs like OpenGL or Direct3D. The limitation of this solution is that the reaction time of the system is affected by the bandwidth of network. It is not suit for real time application (3D games).

3 Possible Hijacking Methods

From 3D model file to images displayed on the screen, there occur many chances that the data of 3D models can be hijacked. Koller described the possible attacks in traditional graphics pipeline in [5]. Here we study the total process from

3D model data to display device including programmable vertex shader and fragment shader as shown in Fig. 1, which has some differences with traditional graphics pipeline. The possible hijacking points including:

Hijack Original 3D Model Data Stream Fig. 1(I) 3D data stream used by 3D applications comes from local storage or Internet. Now most of the 3D games locate the 3D models in local computer. Most of the live-ends get the model data though Internet in culture heritage exhibition system. If the local files and network packets are encrypted, it is very difficult to recover 3D models using reverse-engineering from encrypted local file or network packets.

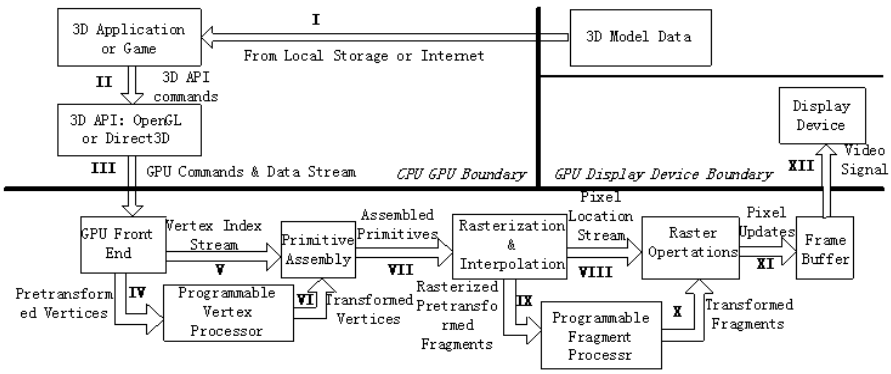


Fig. 1. Programmable Graphics Pipeline

Hijack 3D API Commands Fig. 1(II) Most of 3D games,CAD development suites and other 3D applications use 3D APIs, either Direct3D or OpenGl, to render 3D models. For diverse purposes such as tracing, debugging or implementing distributed graphics architecture, many users, malicious or not, replace the original dynamic library with instrumented versions. For example, Jian Yang's AnyGL system [18], Mohr's stylized rendering system [10, 11]. Nunn's Direct3D benchmark [12], Yan's interactive controlling of Direct3D based games [17], and so on. For 3D Direct3D or OpenGL, there are standard formats for their APIs. Once the original dynamic library is replaced, the 3D models can be reconstructed accurately. Shaping from Rendering Results Fig.1(XII) Framebuffer holds the rendering result which is not transmitted to the display devices. The image displayed on the screen is the last rendering result. Since the first shaping-from-shading technique was developed in early 1970s by Horn, many different algorithms have been emerging. The details are described in [20]. For any interactive 3D system, the goal is to show some pictures on the screen. So the 3D model can be reconstructed if there are enough rendering results. The problem is that the reconstructed model is accurate or not.

4 Our Solution

It is impossible to implement absolute security unless the interactive system does not show any results on the screen. In [5], Koller proposed a scheme to protect interactive 3D graphics which uses remote rendering system to prevent attackers from reconstructing accurate 3D models from rendering results. What displayed on live-on terminal is the snapshots of the rendering results in remote server machina rather than the results rendered with Direct3D or OpenGl. The method cannot be used to protect 3D data of games running in local computer. Here, what we concerned is how to protect 3D API interfaces from being hijacked to reconstruct 3D models. Considering most 3D application need hardware acceleration, the approaches such as software-only rendering, image-based rendering or remote rendering are not appropriate. On the other hand, graphics hardware developed very fast in last few years. Now many graphics cards have programming ability. The architecture of programmable graphics pipeline is shown in Fig. 1. The programming ability of the graphics card can be used to protect 3D models. Our solution encrypts the data of 3D models in 3D application and decrypts the encrypted data in vertex shader.

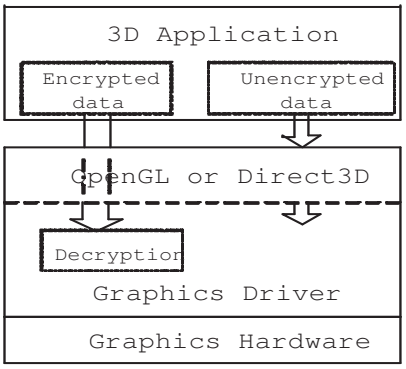


Fig. 2. En-/De-cryption Architecture

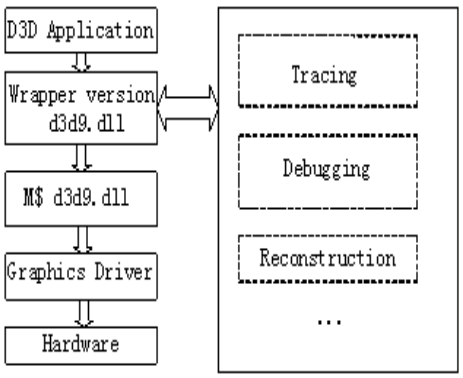
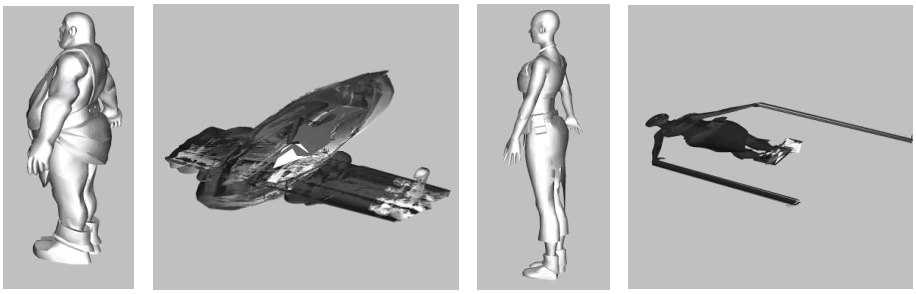


Fig. 3. Advanced D3D Wrapper Architecture

For fixed graphics pipeline, each type of data has its standard format. Only if 3D API commands are hijacked, 3D models can be reconstructed without any error. Now, with the support of programmable graphics hardware, we can first encrypt 3D data stream in 3D applications and let them pass through 3D API commands, then decrypt them in programable vertex shader, as shown in Fig. 2. In addition, to prevent the malicious users from constructing accurate 3D models from frame buffer, 3D data can be deformed slightly before encryption or after decryption with the method in [5]. Thus stream data through interfaces between application and 3D rendering engines are not plain data, and it is very difficult to disasmbley the optimized shader code in assemble language after all. Base



(a) (b) (c) (d)

Fig. 4. The models reconstructed from non-encrypted/encrypted data stream. (a) and (c) are models reconstructed from unencrypted data stream passing through Direct3D APIs; (b) and (d) are models reconstructed from encrypted data stream corresponding to (a) and (c) passing through Direct3D APIs.

on modern cryptography, encryption and decryption are not a difficult problem. But the operations supported by programmable graphics pipeline is limited so far. So many encryption operations cannot be used. With the quick development of GPU, this problem will be solved soon.

5 System Implementation

In order to test our ideas, we implement our method. An advanced Direct3D wrapper is developed to reconstruct models through hijacking 3D APIs. Fig.3 illustrates the architecture of the wrapper. The models shown in Fig. 5 (a) and (c) are reconstructed models with the advanced Direct3D wrapper from 3Dmark03. It is obvious that the reconstruction is accurate if the data passing through Direct3D APIs are not encrypted.

Considering the limitation of the operations supported by graphics processor [2], the geometry information of one vertex is processed as a group. In fact, the texture coordinates can also be processed in same way.

The encryption/decryption procedures include nonlinear and linear operations. Encryption matrix Men and decryption matrix Mde are used to implement linear operation. This way makes full use of the advantages of GPU in matrix processing. The nonlinear operation is used to avoid linear equation based attacks.

The encryption procedure is part of 3D applications. To finish encryption, the 3D data stream are multiplied by encryption matrix and then processed with nonlinear operation. Fig. 5 gives the pseudo code of the encryption procedure in C++ language.

The decryption procedure is implemented as part of vertex shader. On the whole, decryption is the inverse process of encryption. The pseudo code of the decryption procedure is given in Fig. 6 in CG language.

```
//Encryption procedure in C++ language
// Men is 4 by 4 encryption matrix
vector <double, 4> dVector = { 0.0, 0.0, 0.0, 1.0 };
vector <double, 4> dTem = { 0.0, 0.0, 0.0,1.0 };
for(int i=0;i<vertexcount; i++){
        absmean=( abs(vi[0])+ abs(vi[1])+ abs(vi[2]) )/3.0;
        xtem = vi[0]; ytem= vi[1]; ztem= vi[2];
        abs(vi[0])< absmean? vi[0] =- ytem: vi[0] = ytem;
        abs(vi[1])< absmean? vi[1] =- ztem: vi[1] = ztem;
        abs(vi[2])< absmean? vi[2] =xtem: vi[2] = -xtem;
        dTem[0] = vi[0]; dTem[1] = vi[1]; dTem[2] = vi[2];
        D3DXMatrixMultiply(&dVector,&dTem ,&Men);
        vi[0] =dTem[0]; vi[1] =dTem[1]; vi[2] =dTem[2];
};
.....
```

Fig. 5. The pseudo code of encryption procedure

```
//Decryption procedure in CG language
struct OUTPUT{
     float4 position : POSITION;
     float4 color : COLOR;
};

OUTPUT vshader(float4 position : POSITION,...){
        OUTPUT Output;
        float x,y,z,x1,y1,z1;
        //Mde is 4 by 4 decryption matrix Mde = Men^-1
        float4 PosTem;
        .....
        PosTem = mul (position, Mde);
        x1= PosTem.x;
        y1=PosTem.y;
        z1= PosTem.z;
        absmean =( abs(x1) + abs(y1) + abs(z1))/3.0;
        abs(y1)> absmean? Output.x= z1 : Output.x =-z1;
        abs(z1)> absmean? Output.y= -x1 : Output.y=x1;
        abs(x1)>absmean? Output.z= -y1 : Output.z=y1;
        ... ...
        return Output;
}
```

Fig. 6. The pseudo code of decryption procedure

6 Conclusion

The copyright protection of 3D models attract people's great attention in recent years. Many techniques are tried to cope with this problem. There is no perfect technical solution to this problem so far. In this paper, a new solution based on GPU's programmability is proposed. The latest achievement in computer graphics is employed in the scheme. The prototype system shows that the method is promising.

However, the operations provided by GPU has many limitations now. For example, neither bitwise operators nor pointer operations are supported currently by vertex shader. Many advanced encryption algorithm cannot be applied. In addition, only one vertex's data can be accessed, so it's impossible now to encrypt the topology of the model. With the improvement of programming capabilities of GPU, it will be more convenient to implement more complex encryption algorithms. It will be ideal if 3D APIs supply secure mechanism.

Acknowledgment

The research work is supported by National Science Foundation of China (No. 60473111) and the Excellent Young Teacher Program of MOE, PRC.

References

1. Corcoran, J. Demaine, M. Picard, L. G. Dicaire, and J. Taylor. Inuit3d: An interactive virtual 3d web exhibition. Museums and the Web 2002, Apr. 18-20 2002.
2. R. Fernando and M. J. Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.
3. Janson and etc. Web3d security discussion. http://www.sandyressler.com/about/library/weekly/aa013101a.htm.
4. K. H. Ko. *Algorithms for Three-Dimensional Free-Form Object Matching*. Phd. thesis, Massachusetts Institute of Technology, March 2003.
5. D. Koller, M. Turitzin, M. Levoy, and etc. Protected interactive 3d graphics via remote rendering. *ACM Trans. Graph.*, 23(3):695–703, 2004.
6. D. F. Kosak. Sid meier's pirates: Behind the scenes. http://archive.gamespy.com/articles/march04/pirates/index.shtml, March 8 2005.
7. M. Levoy, K. Pulli, B. Curless, and etc. The digital michelangelo project: 3D scanning of large statues. In K. Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144. ACM Press, 2000.
8. L. Li, D. Z. Z. G. Pan, and J. Y. Shi. Watermarking 3d mesh by spherical parameterization. *Computers & Graphics*, 28(6):981–989, 2004.
9. C. M., M. Barni, and F. Bartolini. Towards 3d watermarking technology. In *Proceed-ings of the IEEE Region 8 EUROCON 2003 Conference*, pages 393 –396, 2003.
10. A. Mohr and M. Gleicher. Non-invasive, interactive, stylized rendering. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 175–178, New York, NY, USA, 2001. ACM Press.

11. A. Mohr and M. Gleicher. Hijackgl: reconstructing from streams for stylized rendering. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 13–18, New York, NY, USA, 2002. ACM Press.
12. R. A. Nunn. Unamed direct3d benchmarking tools. http://www.users.on.net/triforce/d3dbench.zip.
13. R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygonal models. In *Proceedings of the ACM International Conference on Multimedia '97*, pages 261–272, Seattle,USA, November, 1997. ACM Press.
14. R. Ohbuchi, A. Mukaiyama, and S. Takahashi. A frequency-domain approach to watermarking 3d shapes. *Computer Graphics Forum*, 21(3):373–382, 2002.
15. E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. In A. Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 49–56, Los Angeles, 1999. Addison Wesley Longman.
16. S. S.Sun, Z. G.Pan, L. Li, and J. Y. Shi. Robust 3d model watermarking against geometric transformation. In *Proceeding of CAD/Graphics'2003*, pages 87–92, Macao, Oct.29-31, 2003.
17. W. Yan. Direct3D pipeline wrapper platform. Bachelor thesis, Zhejiang University, 2004.
18. J. Yang. *AnyGL: A Larger Scale Hybrid Distributed Graphics System.* Phd. thesis, Zhejiang University, 2001.
19. K. K. Yin, Z. Pan, and J. Y. Shi. Texture watermarking in vrml scenes. *Journal of Engineering Graphics*, (3):126–132, 2003. (in Chinese).
20. R. Zhang, P. S. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transac-tions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.
21. J. Y. Zheng. Virtual recovery and exhibition of heritage. *IEEE MultiMedia*, 7(2):31–34, 2000.