iPhone용 3차원 점 집합 가시화

김선정*

요약

본 논문에서는 3차원 점 집합 가시화 방법을 iPhone에 적용시킨다. 3차원 측정 기술의 발전으로 측정 장비에서 얻을 수 있는 3차원 점 집합의 크기는 점점 방대해지고 있다. 이렇게 얻어진 점 집합을 3차원 메쉬로 복원하는 과정 없이, 직접 가시화 하는 방법은 수행 시간과 메모리 사용에 있어 매우 효율적이다. 더구나 iPhone과 같은 스마트 폰에서는 메모리와 같은 리소스가 한정적이기 때문에, 3차원 모델의 표현이 메쉬 보다는 용량이 작은 점 집합이 더 유리하다. 본 논문에서는 iPhone4에서 OpenGL ES 2.0 기반으로 3차원 점 집합의 가시화 방법을 소개한다. OpenGL ES 2.0의 쉐이더 기술을 이용하여 휴대 단말기에서 정교한 3차원 그래픽 렌더링 결과를 보여준다.

3D Point Set Rendering for iPhone

Sun-Jeong Kim*

ABSTRACT

In this paper, 3D point set rendering algorithm is adapted to iPhone. Recent advances in scanning technology have led to a rapid increase in the availability and size of geometric data set. Instead of reconstructing a 3D mesh from a given point set, directly rendering it is very efficient in the running-time and space requirement. In addition, the smart phones such as iPhone have the advantage of rendering a point set because they may have not enough resources to display a 3D mesh whose size is bigger than the size of a point set representation. This paper introduces the rendering scheme of a 3D point set using OpenGL ES 2.0 in iPhone4. Shader techniques in OpenGL ES 2.0 enables to obtain the high-quality results of 3D point set rendering in the smart phones.

Key Words : Point set rendering, iPhone application, OpenGL ES 2.0, Vertex & Fragment Shaders, GPU Programming

---

* 한림대학교 유비쿼터스 컴퓨팅학과(✉sunkim@hallym.ac.kr)
· 제1저자(First Author) : 김선정 · 교신저자(Correspondent Author) : 김선정
· 접수일(2010년 11월 10일), 수정일(1차 : 2010년 12월 7일), 게재확정일(2010년 12월 10일)

Ⅰ. 서 론

레인지 스캐너의 발달로 인해 복잡한 형상의 3차원 모델들이 점점 상용화되고 있는 추세이며, 때로는 수억 개의 점들을 포함할 정도로 데이터 용량도 커지고 있다. 이러한 3차원 점 집합은 일반적으로 삼각 메쉬로 복원된 후 간략화 시켜 사용된다. 그럼에도 불구하고 실시간 디스플레이에 활용되기에는 너무 많은 개수의 삼각형을 가지고 있어, 실행시간 동안 적은 메모리를 차지하는 메쉬 간략화, 시점에 따라 점진적인 디스플레이 등과 같은 많은 알고리즘들이 연구되고 있다.

최근 들어 삼각형 기반 렌더링 알고리즘의 대체 수단으로 점 기반 렌더링 알고리즘이 제안되고 있다. 가장 큰 장점은 점 집합에서 삼각 메쉬로 복원하지 않으므로, 점들 사이의 어떠한 연결 정보나 위상학적인 일관성 등을 유지할 필요가 없다는 점이다. 더구나 계층적 자료 구조를 가질 수 있기 때문에, 효과적인 렌더링 속도 및 시각적 품질의 조절이 가능하다.

기존에 제안된 대부분의 점 집합의 가시화 기법은 대용량의 메쉬로부터 샘플링 된 데이터를 다루기 때문에, 점 집합의 법선 벡터를 원본 메쉬로부터 얻을 수 있다. Grossman과 Dally[1]는 기하학적인 모델로부터 점 집합을 얻어서 렌더링 하였다. 그들의 논문에서는 주로 샘플링 비율과 점 집합 가시화의 결과 이미지에서 발생되는 틈에 대해 논하고 있다.

Surfel[2]은 위 방법에서 연구 동기를 받아 제안된 기술로, 팔진 트리의 자료 구조와 포워드 와핑을 사용한다. Surfel의 최대 공헌은 불규칙하게 샘플링 된 점 집합으로부터 연속적인 표면을 복원하여 점 집합을 가시화 한다는 점이다. 그리고 틈을 제거하기 위해 스플랫(Splat)을 이용하여 디스플레이 하는 방식을 제안하였다. QSplat[3]은 점 집합을 포함하는 바운딩 구를 계층적으로 구성하고, 이미지 공간에서 점 대신 타원이나 직사각형 모양의 스플랫을 사용하여 렌더링 하였다.

바운딩 구의 반지름과 법선 벡터의 콘은 저장되어 실시간 렌더링의 임계값으로 사용된다. Surface Splatting[4]은 오브젝트 공간의 기술로서 텍스처 필터링에 중점을 둔 기술이다. EWA Splatting이라는 가우시안 재샘플링 커널을 통해 점 집합의 가시화를 수행하여, 안티 앨리어싱이 가능한 최상 품질의 결과 이미지를 만들어낸다.

특히, 이 기술은 PointShop3D[5]라는 상호작용이 가능한 형상 디자인 프로그램으로 응용되었다. Botsch et al.[6]은 참조표를 구성하여 법선 벡터, 바운딩 구, 탄젠트 원판 등에 대한 정보를 제한적으로 저장한 후 렌더링에 사용하였다. Kalaiah와 Varshney[7]는 모든 점들에 대해 지역적 미분 정보를 저장하여 렌더링에 사용하였는데, 렌더링 품질은 좋아진 반면 불규칙적으로 샘플링 된 점 집합에는 적용하기 힘든 방법이다.

Wu와 Kobbelt[8]는 조밀한 점 집합에서 의미 있는 점들을 샘플링 하는 방법을 제안하였다. 주어진 허용 오차를 만족하는 최소 개수의 점 집합에 대해 스플랫을 생성하여 렌더링을 수행하였다.

Phong Splatting[9] 기법은 각각의 스플랫 마다 법선벡터를 보간 하여 조명을 계산함으로써 점 집합에 대한 퐁 쉐이딩을 수행하였다. 퐁 쉐이딩, 필터링, 블렌딩의 기술들을 GPU 상의 정점 쉐이더와 프래그먼트 쉐이더를 이용하여 구현하였다.

본 논문에서는 Phong Splatting 기법을 iPhone 어플리케이션으로 만든다. 이를 위해서는 OpenGL ES를 다루어야 하고, 특히 iPhone4에서는 OpenGL ES 2.0을 이용해야 한다. OpenGL ES 2.0은 OpenGL ES 1.x와 하위 호환성을 갖지 않고, 더 이상 고정 함수 파이프라인을 지원하지 않는다. 그러므로 변환과 조명 계산을 모두 정점 쉐이더와 프래그먼트 쉐이더에서 구현해야 한다. 그리고 스마트 폰이라는 특성상 리소스 사용을 최소화해야 한다는 제약사항을 고려하여 알고리즘을 구현해야 한다.

II. 본 론

2.1 시스템 개요

입력받는 점 집합은 적절한 품질로 재샘플링 되었다고 가정한다. 그 이유는 만약 iPhone에서 재샘플링 과정까지 수행한다면 작업량의 부담이 너무 크기 때문이다. 그러므로 서버에서 입력 점 집합에 대해 점들의 간격과 밀도에 대해 고품질의 렌더링 결과가 나오도록 적절하게 재샘플링을 수행한 후 iPhone 단말기로 송신한다고 가정한다. 또한 재샘플링 과정에서 법선 벡터 정보도 계산되어, 점의 위치 정보와 법선 벡터 정보를 함께 입력 받는다.

입력 받은 점 집합에 대해 포인트 스프라이트(Point Sprite)와 정점 쉐이더, 프래그먼트 쉐이더를 이용하여 가시화를 수행한다. 정점 쉐이더에서는 3차원 점을 2차원으로 투영하고, 각 점마다 사용되는 스프라이트의 크기를 카메라의 거리로부터 결정한다. 즉, 카메라에 가까운 점은 크기가 큰 스프라이트를, 거리가 먼 점은 크기가 작은 스프라이트를 이용하여 가시화를 하기 때문에, 점과 점 사이에 틈이 발생하는 것을 방지할 수 있다. 프래그먼트 쉐이더에서는 퐁 쉐이딩을 위해 법선 벡터를 보간 하여 조명 방정식을 계산하고, 계산된 결과를 포인트 스프라이트의 텍셀 값과 함께 알파 블렌딩을 이용하여 가시화를 수행한다.

2.2 점 집합의 가시화 알고리즘

점 집합의 가시화 알고리즘은 Phong Splatting[9] 기법에 기반 한다. 우선, 한 점에 대해 카메라로부터 거리와 투영 변환, 뷰포트의 크기를 가지고, 포인트 스플랫의 크기를 계산한다. 방법은 Phong Splatting에서 제안한 수식을 그대로 사용한다.

$$size = 2r \cdot \frac{n}{z_{eye}} \cdot \frac{h_{vp}}{t-b} \qquad (1)$$

여기서 r은 3차원 상에서 스플랫의 반지름 길이이고, n, t, b는 투영 변환에서 사용되는 near, top, bottom의 값이며, hvp는 뷰포트의 높이로 픽셀 단위이다. 그러므로 결과 값 size는 픽셀 단위로 변환된 스플랫의 반지름 길이가 된다.

각 스플랫은 퐁 쉐이딩으로 렌더링 되며, 가우시안 커널을 이용하여 알파 블렌딩 된다. 가우시안 커널은 쉐이더에서 함수를 이용해 계산해도 되지만, 수행 시간 단축을 위해 (그림 1)과 같은 텍스처를 이용하여 알파 채널을 읽어온다.

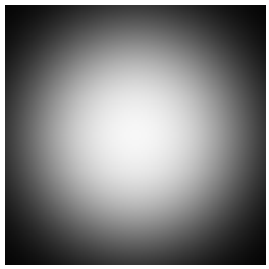

그림 1. 가우시안 커널 텍스처 이미지
Fig. 1 Texture Image for Gaussian Kernel

2.3 iPhone 어플리케이션 구현

프로그램 구성은 (그림 2)와 같다.

코코아 터치 프레임워크의 근간을 이루는 디자인 패턴인 MVC(Model- View-Controller) 모델에 맞춰, 점 집합을 읽어 들이는 모델 파트(PointLoader Class)와 점 집합을 렌더링 하는 뷰 파트(PointRenderViewController Class), 그리고 이를 중앙에서 제어하는 컨트롤러 파트(PointRenderAppDelegate Class)로 나누어 구현되었다. 뷰 파트에는 OpenGL ES를 초기화 하는 클래스(EAGLView Class)와 정점 및 프래그먼트 쉐이더, 그리고 행렬변환과 텍스처 매핑을 위한 기본적인 유틸리티들이 포함되어 있다.

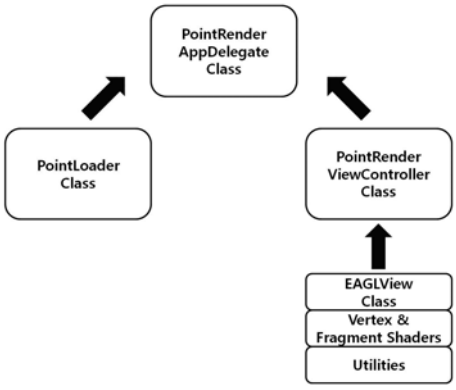

그림 2. 프로그램 구성도
Fig 2. Program Design

프로그램은 XCode에서 제공하는 OpenGL ES Application 템플릿을 이용하여 생성된다. OpenGL ES 2.0에서는 고정 함수 파이프라인을 지원하지 않기 때문에, 기존의 OpenGL 함수로 계산할 수 있었던 투영 및 변환 행렬을 사용자가 계산하여 직접 쉐이더로 전달해야 한다. 그리고 정점 쉐이더에서는 스플랫의 크기 뿐만 아니라, 포인트와 법선 벡터의 뷰잉 좌표계에서의 위치를 계산하여 프래그먼트 쉐이더에 전달한다. varying이라는 변수를 이용하여 정점 쉐이더에서 프래그먼트 쉐이더로 데이터를 전달할 수 있는데, OpenGL ES가 기존 OpenGL과 다른 점 중 하나가 varying 변수에 정밀도 한정자(precision qualifier)를 지정해야 한다는 점이다. 하위, 중위, 또는 상위 정밀도를 지정하여 계산 속도와 전력 효율 사이의 trade-off를 결정할 수 있다. 프래그먼트 쉐이더에서는 퐁 쉐이딩과 가우시안 커널에서 읽어들인 알파채널을 최종 프래그먼트의 알파 값으로 지정하여 알파 블렌딩이 되도록 만든다.

iPhone의 멀티터치(multi-touch) 이벤트를 다룰 수 있도록 활성화 시켜, 3차원 점 집합에 대해 사용자가 원 터치 드래그로 평행 이동이나 회전을, 멀티 터치 드래그로 크기 변환을 할 수 있도록 구현한다.

III. 실험 결과

본 알고리즘은 MacBook Pro(Mac OS X 10.6.5버전), 2.4GHz Intel Core 2 Duo 프로세서와 4GB 메모리 환경에서 구현되었다. 사용된 소프트웨어는 XCode (3.2.4버전)와 iPhone 시뮬레이터 4.1이다.

표 1. 사용된 3차원 점 집합의 개수와 메모리 사용량
Table 1. The number of points for 3D models and their memory usage

모델	Bunny	Rabbit	Teeth	Venus	Buddha
점 개수	34,834	67,039	116,604	134,345	543,652
메모리 사용량	28.2MB	28.8MB	30.0MB	29.4MB	39.9MB

(그림 3)은 구현 결과를 보여 주고 있다. (표 1)에서 볼 수 있듯이 다양한 크기의 3차원 점 집합을 가시화 할 수 있도록 견고하게 만들어 졌다. 기본적인 뷰어를 위한 메모리 사용량은 12.9~14.1MB이고, 각각의 모델의 크기에 따른 메모리 사용량은 (표 1)과 같다. 점의 개수가 많이 늘어도 메모리 사용량이 급격하게 증가하지 않는 이유는, 컬링(culling)을 수행하기 때문이다. 즉 뒷면에 있는 점들은 렌더링 파이프라인에 들어올 수 없도록 쉐이더에서 구현되어 있기 때문에, 메모리 사용량이 줄어들 뿐만 아니라 수행 속도도 향상된다. 수행 속도는 전처리 과정에서 모든 점들의 법선 벡터와 스플랫 기본 크기가 계산되기 때문에, 수행 시간은 렌더링 되는 시간으로 앞면을 향하는 점들의 개수에 비례한다. 다만, iPhone에서 실행 속도를 빠르게 하기 위해, 1 pass만 이용하여 렌더링을 수행하고 있어 날카로운 모서리나 실루엣 등이 잘 표현하지 못하고 있다. 또한 스플랫이 많이 겹친 부분은 여러 번 알파 블렌딩이 되어 주변보다 약간 밝아진 경향을 볼 수 있다.

그림 3. iPhone 시뮬레이터에서 실행된 3차원 점 가시화 결과
Fig. 3 Results of 3D point set rendering in iPhone simulator

IV. 결론 및 향후 연구

본 논문에서는 iPhone에서 3차원 점 집합 가시화를 방법을 소개하였다. 고품질 렌더링을 위해 OpenGL ES 2.0 기반으로 Phong Splatting 기법을 적용하였다. 즉 가우시안 모양의 알파 채널을 갖는 텍스처를 이용한 포인트 스프라이트 기술로 스플랫을 렌더링 하였다.

향후 연구로는 2 pass 렌더링 알고리즘으로 확장시켜 첫 번째 pass에서는 깊이 버퍼와 알파 값만을 버퍼에 쓰고, 두 번째 pass에서 날카로운 모서리 복원과 알파 값의 정규화 하여 주위보다 밝아진 부분이 제거되도록 만들 것이다. 또 사용자에게 편의성을 제공하기 위해 iPhone의 UI도 개선할 것이다.

Acknowledgement

"이 논문은 2010년도 한림대학교 교비연구비 (HRF-2010-031)에 의하여 연구되었음."

참고문헌

[1] J. Grossman and W. Dally. "Point sample rendering", *In Proc. of 9th Eurographics Workshop on Rendering*, pp. 181-192, 1998.

[2] H. Pfister, M. Zwicker, J. Baar, and M. Gross, "Surfels: surface element as rendering primitives", *In Proc. of SIGGRAPH 2000*, pp. 335-342, July 2000.

[3] S. Rusinkiewicz and M. Levoy, "QSplat: a multi- resolution point rendering system for large meshes", *In Proc. of SIGGRAPH 2000*, pp. 343-352, July 2000.

[4] M. Zwicker, H. Pfitser, J. Baar, and M. Gross, "Surface splatting", In Proc. of SIGGRAPH 2001, pp. 371-378, August, 2001.

[5] M. Zwicker, M. Pauly, O. Knoll, and M. Gross, "Pointshop3D: an interactive system for point-based surface editing", *ACM Transactions on Graphics (Proc. of SIGGRAPH 2002)*, Vol. 21, No. 3, pp. 322-329, July 2002.

[6] M. Botsch, A. Wiratanaya, and L. Kobbelt, "Efficient high quality rendering of point sampled geometry", *In Proc. of 13th Eurographics Workshop on Rendering*, pp. 53-64, 2002.

[7] A. Kalaiah and A. Varshney, "Differential point rendering", *In Proc. of the 12th Eurographics Workshop on Rendering*, pp. 139-150, 2001.

[8] J. Wu and L. Kobbelt, "Optimized sub-sampling of point sets for surface splatting", *Computer Graphics Forum(Proc. of Eurographics 2004)*, Vol. 23, No. 3, pp. 643-652. September 2004.

[9] M. Botsch, M. Spernat, and L. Kobbelt, "Phong splatting", *In Proc. of the Eurographics Symposium on Point-Based Graphics*, pp. 25-32, August 2004.

김선정(Sun-Jeong Kim)

1996년 고려대학교 컴퓨터학과 학사
1996년 고려대학교 컴퓨터학과 석사
2003년 고려대학교 컴퓨터학과 박사
2005년~현재 한림대학교 유비쿼터스 컴퓨팅학과 부교수
※ 관심분야 : 컴퓨터 그래픽스, 3차원 게임, 가상현실