블렌딩 곡면의 GPU 기반 렌더링

고대현°
서울대학교
jyune1@snu.ac.kr

GPU-based Rendering of Blending Surfaces

Dae-Hyun Ko°
Seoul National University

요 약

곡면은 다각형 메쉬에 비해 간결한 표현을 가지고 부드러운 형상을 표현할 수 있지만, 현재 그래픽스 하드웨어는 곡면 렌더링을 지원하고 있지 않다. 최근의 GPU를 이용하여 프로그래밍 가능한 그래픽스 파이프라인은 기존의 다양한 그래픽스 알고리즘들을 가속화시키는 데 이용될 수 있기 때문에, 이 논문에서는 GPU 하드웨어를 이용하여 블렌딩을 기반으로 생성된 임의의 위상을 가진 곡면을 빠르게 렌더링하는 방법을 제시한다. 제어 메쉬상에서 샘플링된 매개변수와 지역 곡면을 표현하는 정보들을 그래픽스 파이프라인으로 전달하여, 정점 프로세서에서 이러한 정보들을 가지고 블렌딩된 곡면상의 위치 정보와 노말 벡터를 계산한다. 이러한 방법은 CPU에서 블렌딩 곡면을 계산하는 것보다 훨씬 빠르게 수행된다.

Abstract

Although free-form surfaces can represent smooth shapes with only a few control points contrary to polygonal meshes, graphics hardware does not support surface rendering currently. Since modern programmable graphics pipeline can be used to accelerate various kinds of existing graphics algorithms, this paper presents a method that utilizes the graphics processing unit (GPU) to render blending surfaces with arbitrary topology fast. Surface parameters sampled on the control mesh and geometric data for local surfaces are sent to the graphics pipeline, and then the vertex processor evaluates the surface positions and normals with these data. This method can achieve very high performance rather than CPU-based rendering.

키워드 : 블렌딩 곡면, 다양체, 그래픽스 하드웨어, GPU
Keywords : blending surface, manifold, graphics hardware, GPU

1. 서 론

그래픽스 파이프라인에서 수행되는 많은 병렬처리 계산들을 빠르게 처리하기 위해 사용되는 GPU는 CPU보다 훨씬 빠르게 발전하고 있으며, 과거 고정된 그래픽스 파이프라인(fixed graphics pipeline)에서 벗어나 사용자가 직접 그래픽스 파이프라인을 프로그래밍하는 것을 가능케 하고 있다. 하지만 여전히 그래픽스 렌더링 하드웨어는 삼각형에 최적화되어 설계되었고 곡면을 기본 요소(primitive)로 지원하고 있지 않기 때문에, 곡면을 렌더링하기 위해서는 먼저 다각형으로 세분화(tessellation)한 다음 파이프라인으로 전달해야 한다.

따라서 최근에 이러한 GPU를 활용하여 스플라인 곡면(spline surface)이나 분할 곡면(subdivision surface), 음함수 곡면(implicit surface)과 같은 곡면들의 렌더링을 가속화하려는 시도가 이루어지고 있다.

이 논문에서는 임의의 위상(topology)을 가진 부드러운 곡면을 쉽게 모델링할 수 있는 블렌딩 곡면을 그래픽스 하드웨어상에서 렌더링하는 방법을 제시한다. 먼저 전처리 과정에서 임의의 위상을 나타내는 제어 메쉬를 생성한다. 제어 메쉬의 각 정점에서 인접한 면들을 바탕으로 차트가 정의되는데 각 차트상에서 제어 메쉬를 근사하는 지역 곡면과 가중치 함수를 생성한다.

런타임시에는 지역 곡면의 계수들과 같이 곡면을 표현하는 정보들과 제어 메쉬의 각 면에서 샘플링된 매개변수들을 그래픽스 파이프라인에 전달하고, 정점 프로그램에서 입력받은 매개변수를 인접한 차트들의 매개변수로 변환하고 각 차트상에서 정의된 지역 곡면의 위치 정보와 노말 벡터 및 가중치를 계산한다. 최종적으로 이러한 지역 곡면들을 블렌딩하여 부드러운 곡면을 생성한다. 이러한 방법은 CPU에서 곡면을 직접 계산하는 것보다 약 15–20배정도 빠르게 수행됨을 확인하였다.

논문의 구성은 다음과 같다. 2절에서는 관련 연구들을 소개하고, 3절에서 블렌딩을 기반으로 임의의 위상을 가진 곡면을 모델링하는 방법을 설명한다. 그러한 곡면을 하드웨어상에서 구현하는 방법을 4절에서 설명하고 5절에서 실험결과를 제시한다. 끝으로 6절에서 결론을 맺는다.

2. 관련 연구

이 절에서는 다양체 기반의 블렌딩 곡면 모델링 방법과 그래픽스 하드웨어를 이용한 여러 곡면의 렌더링에 관련된 기존 연구들을 소개한다.

2.1 블렌딩 곡면

Grimm 등 [1]은 컴퓨터 그래픽스 분야에서 처음으로 다양체에 기반하여 임의의 위상을 갖는 곡면을 모델링하는 방법을 제시하였다. 사용자가 스케치 메쉬를 생성하면 그 메쉬의 정점, 모서리, 면 등 모든 원소에서 차트를 정의하였다. 따라서 정의해야 할 차트의 수가 너무 많았고 차트 사이의 전이 함수는 다소 주먹구구식으로 정의되었다. 그리고 차트에서 블렌딩과 임베딩(embedding) 함수를 정의하기가 복잡하였다.

최근에 Ying 등 [2]은 연속성과 시각적 품질(visual quality)에 초점을 맞춘 연구를 발표하였다. 그들은 메쉬의 연결정보에 대해 어떠한 가정을 두지 않고 또 각 정점에 대해서만 하나의 차트를 생성하므로, 이전 연구들에 비해 훨씬 적은 수의 차트를 사용하였고 C^{∞} 연속성을 보장하였다.

앞의 Ying 등 [2]의 연구를 바탕으로 Yoon [3]은 점 집합(point set)으로 주어진 데이타에 대해 기반 곡면(base surface)을 구성하고 점 집합에서 기반곡면으로 투영한 변위값을 가지고 변위 함수(displacement function)를 생성하여 변위 곡면(displaced surface)을 생성하였다.

2.2 하드웨어 기반 곡면 렌더링

Guthe 등 [4, 5]은 전처리과정에서 NURBS와 T-spline 곡면을 근사하는 3차 Bézier 곡면들의 계층구조를 만들고 런타임시에 곡면의 매개변수 영역을 샘플링한 격자를 파이프라인으로 전달함으로써 정점 프로그램에서 렌더링을 수행하였다. 컬링(culling)과 LOD(level of detail) 기법을 함께 활용하여 높은 성능향상을 얻었다.

Bolz 등 [6]은 기저 함수(basis function)의 선형 결합(linear combination)의 세분화는 기저 함수의 세분화를 선형 결합한 것과 같다는 점에 착안하여, 전처리과정에서 기저 함수를 세분화한 결과를 텍스쳐로 저장하고 런타임시에 제어 메쉬의 제어점을 전달함으로써 프래그먼트 프로그램(fragment program)에서 Catmull-Clark 분할 곡면을 계산하는 방법을 제안하였다.

Loop 등 [7]은 2차와 3차 Bézier 곡선으로 둘러싸인 영역을 그리기 위해 각 픽셀에서 곡선의 내부여부를 결정하는 함수를 계산하는 프래그먼트 프로그램을 구현하였다. 이는 Loop 등 [8]에서 Bézier 사면체로 정의되는 대수 곡면(algebraic surface)의 경우로 확장되었고, 세분화할 필요없이 곡면을 픽셀단위로 직접 렌더링하였다.

한편, GPU상에서 3변수 B-스플라인(trivariate B-spline)을 직접 계산함으로써 자유 형상 변형(free-form deformation)에 적용한 예가 Schein 등 [9]에 의해 소개되었다.

3. 블렌딩 곡면

다양체(manifold) [10]를 바탕으로 임의의 위상을 가진 곡면을 생성하는 방법을 소개한다.

3.1 다양체

임의의 위상을 가진 곡면을 표현하기 위해 다양체 기반의 곡면생성 방법을 적용하였다. 기존의 스플라인 기반 접근방법에서는 제어 메쉬(control mesh)상의 각 면에서 곡면 패치(surface patch)를 정의하여 표현하는데, 이 패치들의 경계상에서 연속성을 보장하도록 각 패치들을 정의하는 것이 어려웠다. 반면에 다양체 기반 접근방법에서는 제어 메쉬를 덮는 차트들을 서로 겹치게 정의할 수 있고 이 차트상에서 제어 메쉬를 근사하도록 정의된 지역 곡면들을 블렌딩함으로써 그러한 어려움을 피하고 있다.

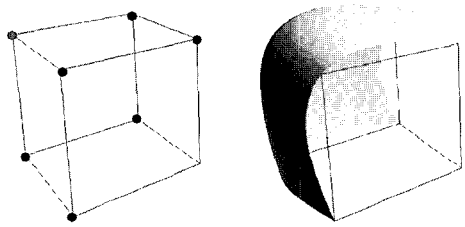
3.2 차트

차트(chart)는 제어 메쉬의 각 정점에서 정의된 파라미터 도메인이다. 본 논문에서 사용한 차트는 Ying 등 [2]에 의해 제안된 방법을 바탕으로 하며, 이것은 그 정점을 둘러싸는 이웃한 면들(one-ring neighborhood)을 복소평면으로 변환시킴으로써 형성되고, 이웃한 차트간의 전이 함수(transition map)를 부드럽게 정의할 수 있어 다음에 설명할 지역곡면들을 부드럽게 블렌딩할 수 있게 한다. 제어 메쉬의 각 면의 파라미터 도메인에 해당하는 단위 사각형(unit square)은 일련의 회전변환과 등각사상(conformal mapping)의 결합에 의해 차트상의 적절한 부분으로 매핑되며, 이 변환은 정점의 수가(valence)에만 의존한다.

3.3 지역 곡면

전역 곡면(global surface)을 형성하기 위해 제어 메쉬의 각 정점에서 이웃한 면들을 근사하는 지역 곡면(local surface)을 생성한다. 일반적으로 곡면은 파라미터 2개에 의해 정의된 평

면상의 단위 사각형에서 정의되지만, 여기서 언급하는 지역 곡면은 앞에서 언급한 복소평면상의 차트상에서 정의된다는 점이 다르다. 이웃한 면들을 근사하기 위해 Ying 등 [2]은 제어 메쉬의 Catmull-Clark 분할 곡면상의 점들을 샘플링하여 최소 제곱법(least square fitting) 과정을 통해 지역 곡면을 생성하였고, Yoon [3]은 이웃한 면들의 모든 정점들을 보간하는 2차 다항식 곡면을 이용하였다. 본 논문에서는 2개의 벡터와 1개의 행렬의 곱셈으로 간단히 표현되는 Yoon [3]의 지역 곡면을 이용하였는데, 벡터-행렬 연산에 최적화되어 있는 그래픽스 하드웨어상에서 간단히 계산할 수 있기 때문이다. 그림 1 (a)의 붉은 색으로 표시된 정점에서 정의된 지역 곡면은 그 정점을 포함하여 검은 색으로 표시된 인접한 면들의 정점들을 보간하도록 생성되며 그림 1 (b)에 나타나 있다.



(a) (b)

그림 1: 지역 곡면

3.4 가중치 함수

가중치 함수는 지역 곡면과 마찬가지로 동일한 차트상에서 정의되며, 지역곡면의 제어 메쉬상에서의 영향력을 조절한다. 즉, 지역 곡면이 정의된 정점에서는 가중치가 가장 크고 인접한 면들의 경계로 갈수록 작아진다. 부드러운 블렌딩 결과를 얻기 위해 부드러운 지역 곡면을 사용할 뿐만 아니라 가중치도 부드럽게 변해야 하므로 허밋 스플라인(Hermite spline)을 이용하였다. 그림 2처럼 허밋 스플라인의 텐서곱(tensor product)을 통해 단위 사각형에서 정의된 가중치 분포함수를 얻고 이것을 차트상으로 변환하여, 차트의 중심에서 가중치가 가장 크고 경계로 갈수록 부드럽게 작아지는 가중치 분포함수를 얻는다.

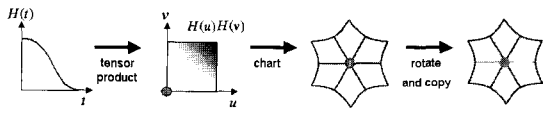


그림 2: 가중치 함수

3.5 전역 곡면

앞에서 정의된 지역 곡면과 가중치 함수를 바탕으로 partition of unity를 사용하여 블렌딩함으로써 그림 3처럼 최종적인 전역 곡면을 생성한다. 먼저 제어 메쉬의 표면상에서 샘플링된 각 점에 대하여 이 점을 포함하는 모든 차트들을 찾고 각 차트상으로 매핑한다. 그 결과를 가지고 지역 곡면과 가중치 함수를 계산하고 블렌딩함으로써 전역 곡면상의 점을 얻는다. 노말 벡터도 같은 방법으로 구할 수 있다.

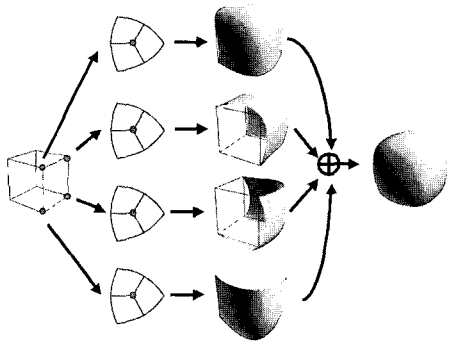


그림 3: 블렌딩

4. 하드웨어 렌더링

이 절에서는 앞에서 언급한 블렌딩 곡면을 그래픽스 하드웨어상에서 렌더링하기 위한 방법을 제시한다. 그림 4는 이러한 전체적인 과정을 나타내고 있다.

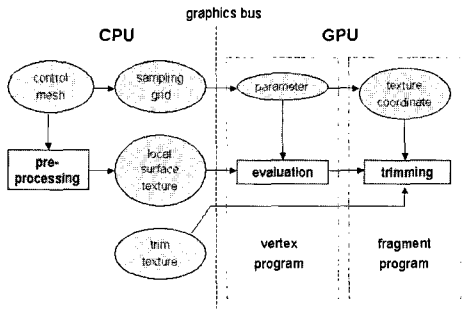


그림 4: 전체 과정

4.1 전처리과정

먼저, 전처리 단계에서 제어 메쉬의 각 정점에서의 지역 곡면을 나타내는 2차 다항식 곡면을 계산한다. 곡면의 계수들은 9개의 점들로 구성되는데 이들을 2차원 텍스쳐에 저장하여 텍스쳐 메모리로 전달한다. 정점 프로그램(vertex program)에서 이러한 텍스쳐 메모리에 접근할 수 있는 기능(vertex texture fetch)은 shader model 3.0 이상에서만 지원되는 기능이다. NVIDIA GPU의 경우 GeForce 6 시리즈 이상에서만 가능하다. 텍스쳐 대신 직접 배열에 저장하여 상수 레

지스터(constant register)로 전달할 수도 있는데 이 경우 텍스쳐 메모리를 접근하는 것보다 더 빠른 성능을 얻을 수 있지만, 레지스터 공간의 제약때문에 제어 메쉬의 정점의 갯수, 즉 지역곡면의 갯수가 적은 경우를 제외하고는 텍스쳐를 이용한다.

4.2 정점 프로그램

그래픽스 파이프라인상의 정점 처리단계(vertex processing)에서 곡면을 계산할 수 있도록 정점 프로그램을 구현하였다. 제어 메쉬의 각 면을 샘플링한 점을 그래픽스 파이프라인으로 전달하면 각 점에 대해 정점 프로그램이 실행된다. 그 점을 인접한 차트상의 매개변수로 변환하기 위해 필요한 정보들도 같이 전달한다. 이러한 정보들은 제어 메쉬의 동일한 면에서 샘플링된 점에 대해서 모두 같은 값을 가지므로, uniform 변수로 전달함으로써 각 면에 대해서는 한 번만 전달하도록 한다.

차트 매개변수: 정점 프로그램에서 가장 먼저 수행할 일은 입력받은 점을 인접한 차트상의 매개변수로 변환하는 것이다. 제어 메쉬의 각 면이 4각형인 경우 인접한 차트는 4개이므로 각 차트에 대하여 모두 4번의 변환을 수행해야 하지만, 그래픽스 하드웨어는 벡터와 행렬을 기본 데이타형으로 사용하므로 1번의 변환과정을 통해 모든 차트상에서의 파라미터를 구할 수 있다. 각 변환과정은 일련의 2차원 회전변환과 등각사상의 결합으로 구성된다. 이 과정에서 sine과 cosine 삼각함수를 이용하게 되는데, 쉐이딩 언어에서 제공하는 둘의 값을 한꺼번에 계산할 수 있는 효율적인 함수를 이용한다. 2차원 회전변환의 경우 x, y 좌표별로 별도의 계산을 수행하지 않고 회전행렬과 위치벡터를 곱함으로써 한 번에 수행한다.

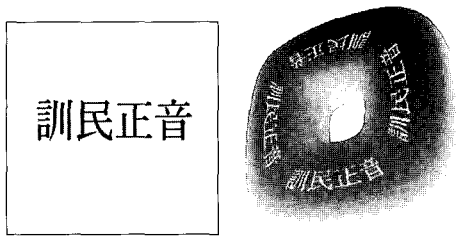
지역 곡면: 다음은 앞에서 변환된 매개변수를 가지고 각 차트상에서 정의된 지역 곡면의 위치와 노말 벡터를 계산해야 한다. CPU에서 지역곡면을 계산하기 위해서는 x, y, z 각 좌표에 대해 12번의 곱셈연산과 8번의 덧셈연산, 총 60번의 연산이 필요하다. 반면에 그래픽스 하드웨어상에서는 2개의 벡터와 1개의 행렬을 구성한 다음, x, y, z 각 좌표별로 곱셈연산 1번, 내적연산 1번 총 6번의 연산을 통해 위치를 계산할 수 있다. 노말 벡터도 마찬가지로 2개의 편미분값을 계산하는데 각각 6번의 연산만이 필요하다. 그리고 벡터 데이타형을 위해 하드웨어에서 제공하는 효율적인 외적연산과 노말라이즈(normalize) 연산을 이용한다.

블렌딩: 곡면계산의 마지막 단계로 앞에서 구한 위치와 노말 벡터를 가중치를 이용하여 블렌딩하여 전역 곡면상에서의 위치와 노말 벡터를 계산한다. 이를 위해 먼저 가중치 함수를 계산해야 하는데 쉐이딩 언어에서 제공하는 허밋 보간 함수를 이용하였다. CPU에서 4개의 지역 곡면상의 점을 블렌딩하려면 12번의 곱셈연산, 9번의 덧셈연산이 필요하다. 반면에 그래픽스 하드웨어상에서는 4개의 가중치 값을 4개의 성분를 가진 벡터에 담고 4개의 지역 곡면상의 점의 x 좌표값을 담은 벡터를 생성하여, 두 벡터간의 한번의 내적연산만을 수행하면 블렌딩된 전역 곡면상의 위치의 x 좌표값을 얻는다. 따라서 y, z 좌표를 포함하여 총 3번의 내적연산을 통해 그 위치를 구할 수 있다. 노말 벡터에 대해서도 마찬가지다.

T&L(Transformation & Lighting): 지금까지 계산한 블렌딩 곡면의 최종위치는 모델링 좌표계상의 정점이므로, CPU상에서 수행되는 응용 프로그램으로부터 현재의 모델뷰 프로젝션(modelview projection)에 해당하는 행렬을 전달받아서 위에서 계산된 정점과 곱하여 모델뷰 프로젝션 변환을 수행한다. 그 결과 생성된 동차좌표계(homogeneous coordinate)상의 정점을 반환하여 그래픽스 파이프라인의 다음 단계로 전달한다. 뿐만 아니라 사용자정의 조명(lighting) 함수를 구현한 다음, 노말 벡터를 이용하여 현재 정점에서의 색상을 계산하여 반환한다. 이 색상은 래스터화(rasterization) 과정을 거치면서 선형보간되어 곡면을 구성하는 프래그먼트(fragment)상의 색상이 된다.

4.3 프래그먼트 프로그램

트리밍: 프래그먼트 프로그램을 이용하여 곡면에 대해 트리밍(trimming)을 수행할 수 있다. 먼저 제어 메쉬의 각 면에 대응하는 텍스쳐(trim texture)를 생성한다. 이 텍스쳐는 그림 5 (a)처럼 곡면상의 잘려나갈 부분이 검은색으로 설정된 바이너리 이미지(binary image)이다. 이 면에 해당하는 전역 곡면상의 부분이 래스터화 과정을 거치면서 생성된 프래그먼트들은 선형보간된 텍스쳐 좌표들를 가지고 있다. 이 좌표는 정점 프로그램에서 입력받은 점의 파라미터를 텍스쳐 좌표로써 반환하고 이 좌표가 래스터화 과정을 거치면서 선형보간된 것으로서, 이 값을 가지고 트림 텍스쳐에 접근하여 잘려나갈 것인지 아닌지 판별한다. 그에 따라 해당 프래그먼트를 프레임버퍼로 전송하거나 버림으로써 트리밍을 수행할 수 있다. 그림 5 (a)의 트림 텍스쳐를 가지고 그림 6 (b)의 곡면에 트리밍을 적용한 예가 그림 5 (b)에 나타나 있다.



(a) (b)

그림 5: 트리밍

5. 실험 결과

하드웨어 렌더링 알고리즘은 NVIDIA의 Cg 쉐이딩 언어 [11]를 사용하여 구현되었다. 실험에 사용한 컴퓨터 사양은 펜티엄4 3.2 GHz CPU, 1GB 메모리, NVIDIA GeForce 6800 Ultra GPU이다.

논문에서 실험에 사용한 제어 메쉬와 블렌딩 곡면들은 그림 6과 같으며 각각 CPU/GPU를 이용하여 측정된 성능은 그림 7에 나타나 있다. 그림 8은 그림 6 (b)의 제어 메쉬에 대하여 각 면을 샘플링한 간격을 증가시킴에 따라 CPU/GPU에 의한 성능 변화를 비교한 결과이다. GPU를 이용할 경우 약 15–20배 정도 성능향상을 얻을 수 있음을 확인할 수 있었다.

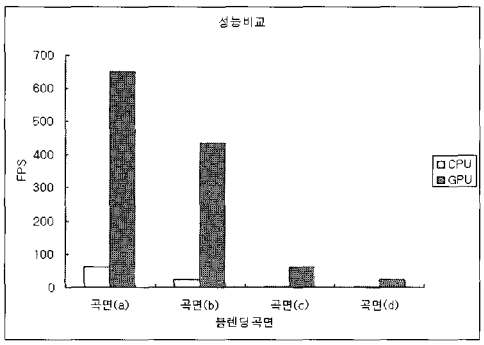


그림 7: 여러 곡면에 대한 성능비교

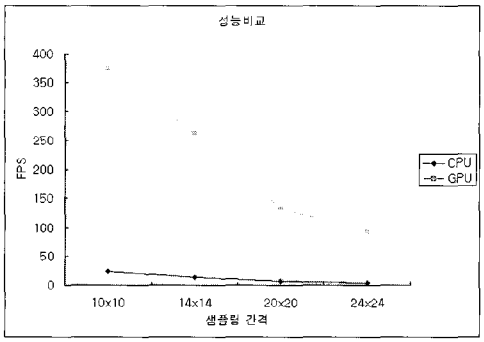


그림 8: 샘플링 간격에 따른 성능비교

6. 결 론

임의의 위상을 가진 부드러운 곡면은 다양체에 기반한 블렌딩을 이용하면 연속성 보장을 위한 까다로운 조건없이 간단히 표현할 수 있다. 그리고 계산과정을 살펴보면 병렬처리하기에 적당한 구조를 가지고 있기 때문에 이 논문에서는 이러한 블렌딩 곡면을 그래픽스 하드웨어를 이용하여 빠르게 렌더링하는 방법을 소개하였다.

일반적으로 곡면을 렌더링기 위해서는 평면상의 단위 사각형에서 샘플링된 파라미터를 가지고 곡면상의 위치와 노말 벡터를 계산하여 세분화된 다각형 메쉬를 구성한 다음, 그래픽스 API의 정점과 노말 벡터 관련 명령어를 통해 그래픽스 하드웨어로 전달하게 된다. 즉, 곡면 계산과 렌더링의 과정이 각각 CPU와 GPU로 구분되어 있다.

반면에 그래픽스 하드웨어 기반에서 곡면을 렌더링한다는 말은, CPU에서는 세분화를 위한 파라미터를 생성하는 일만 담당하고, 이 파라미터를 그래픽스 파이프라인으로 전달하여 GPU에서 실제 곡면상의 위치와 노말벡터를 계산하고 렌더링도 함께 수행한다는 것이다. 이러한 개념은 일반적인 매개화 곡면(parameterized surface)에 적용될 수 있으며, 여기서는 블렌딩 곡면을 통해 그래픽스 하드웨어가 곡면 렌더링을 가속화할 수 있음을 보여주었다.

블렌딩 곡면은 분할 곡면과 비슷한 부드러운 곡면을 빠르게 생성할 수 있으므로 실시간 응용분야에서 유용할 것이며, 다른 종류의 지역 곡면과 블렌딩 함수를 사용한다면 보다 다양한 블렌딩 곡면을 생성할 수 있을 것으로 기대된다.

감사의 글

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음

참고 문헌

[1] C. Grimm and J. Hughes, Modeling surfaces of arbitrary topology using manifolds, Proceedings of ACM SIGGRAPH '95, pp.359-368, 1995.

[2] L. Ying and D. Zorin, A simple manifold-based construction of surfaces of arbitrary smoothness, ACM Transactions on Graphics, 23(3):271-275, 2004.

[3] S.-H. Yoon, A surface displaced from a manifold, Lecture Notes in Computer Science, 4077:677-686, 2006.

[4] M. Guthe, Á. Balázs, R. Klein, GPU-based trimming and tessellation of NURBS and T-Spline surfaces, ACM Transactions on Graphics, 24(3):1016-1023, 2005.

[5] M. Guthe, Á. Balázs, R. Klein, GPU-based Appearance Preserving Trimmed NURBS Rendering, Journal of WSCG, 14(1-3):1-8, 2006.

[6] J. Bolz and P. Schröder, Evaluation of Subdivision Surfaces on Programmable Graphics Hardware, submitted for publication, 2003.

[7] C. Loop and J. Blinn, Resolution independent curve rendering using programmable graphics hardware, ACM Transactions on Graphics, 24(3):1000-1009, 2005.

[8] C. Loop and J. Blinn, Real-time GPU rendering of piecewise algebraic surfaces, ACM Transactions on Graphics, 25(3):664-670, 2006.

[9] S. Schein and G. Elber, Real-time Free-form Deformation using Programmable Hardware, Proceedings of 2005 Israel-Korea Bi-National Conference on New Technologies

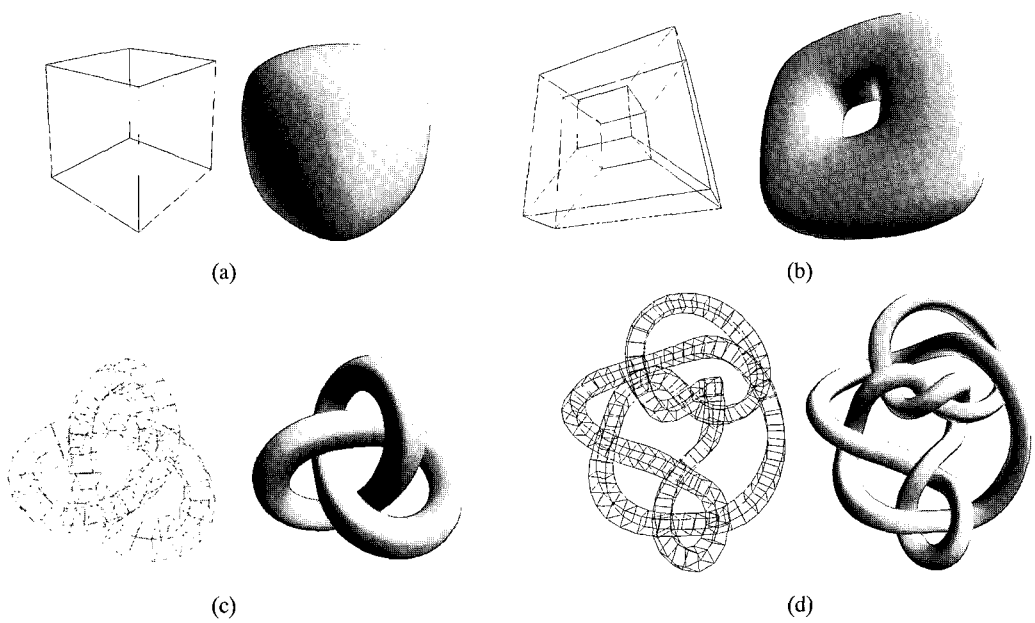


그림 6: 제어 메쉬와 블렌딩 곡면

and Visualization Methods for Product Development on Design and Reverse Engineering, pp.42-46, 2005.

[10] C. Grimm and D. Zorin, Surface Modeling and Parameterization With Manifolds, SIGGRAPH 2006 Course Notes, 2006.

[11] NVIDIA, Cg Toolkit User's Manual, 2006.