실시간 연필 렌더링

이현준 * 권성태 † 이승용 ‡
POSTECH

Real-Time Pencil Rendering

Hyunjun Lee Sungtae Kwon Seongyong Lee
POSTECH

요 약

본 논문에서는 주어진 삼차원 메쉬를 실시간에서 연필화로 표현하는 방법을 소개한다. 이를 위해 본 논문에서는 연필화의 특징을 분석한 후 분석된 특징들을 빠르고 간단하게 모사하여 표현하는 알고리즘을 제시한다. 이 알고리즘은 크게 윤곽선을 그리는 방법과 내부를 그리는 방법으로 나뉘며, 제시된 모든 알고리즘은 그래픽스 하드웨어를 이용하여 실시간으로 동작한다. 우선 물체의 윤곽선을 그리기 위해 사람이 윤곽선을 그릴 때의 특징을 분석하여 실제의 윤곽선과 비교해 약간의 오차를 지니면서 여러 번 겹쳐 그린 듯한 윤곽선을 만들어 내는 방법을 제시한다. 또한 물체의 내부를 그리기 위해서 먼저 연필의 특징을 잘 반영하는 텍스쳐를 만든 후 이를 물체의 표면에 매핑시켜 물체의 특징을 연필화풍으로 잘 표현하는 방법을 제시한다. 이 과정에서 연필화의 느낌을 강조하기 위해 물체가 그려지는 종이의 질감을 표현하는 방법과 물체의 명암비를 조정하는 방법 역시 제시한다.

제 1 절 서론

회화는 회화를 그리는 재료와 표현 방법에 따라서 매우 다양한 종류가 있다. 이런 다양한 회화를 표현하기 위한 여러 비사실적 표현 기법이 컴퓨터 그래픽스 분야에서 개발되어 왔다. 여기에는 펜과 잉크 [27, 28, 11, 5, 26], 수채화 [3, 10, 14], 목탄 [2, 15] 등의 많은 방법들이 있다.

다양한 재료들 중에서도 연필은 가장 많이 쓰이는 재료 중 하나이다. 연필은 구하기 쉽고 다루기도 쉬우며 지우개를 사용해 쉽게 수정할 수 있다. 또한 연필은 연필의 종류와 그리는 방법에 따라 물체의 다양한 밝기와 양감 그리고 질감을 표현할 수 있다.

컴퓨터로 연필화를 만들어 내기 위한 여러 방법들이 제시되었다. 그 중 한 방법으로 실제 연필과 종이 그리고 지우개 등 연필화를 구성하는 요소들을 분석한 후 이를 시뮬레이션하여 삼차원 메쉬로부터 연필화를 만들어 내는 방법이 제시된 바 있다 [23, 24]. 이와는 달리, 삼차원 메쉬가 아닌 이차원 영상을 입력으로 받아들여 이를 연필화처럼 만들어 내는 방법 역시 제시되었다 [16].

컴퓨터 하드웨어의 발전은 컴퓨터 그래픽스에 큰 변화를 가져왔다. 보다 더 빠른 연산이 가능해짐에 따라 실시간에서 다양한 사실적, 그리고 비사실적 렌더링이 가능해졌다. 그 중 실시간 선영 기법(Real-Time Hatching) [20, 25]은 삼차원 메쉬를 받아들여 이를 실시간으로 선영하는 방법을 제시하였다.

그러나 아직까지 실시간에서 삼차원 메쉬를 연필화로 표현하는 방법은 제시된 바 없다. 시뮬레이션 기반의 방법은 실제로 연필화와 매우 흡사한 영상를 만들어 내지만, 복잡한 계산 과정으로 인해 실시간에서 동작하지 못한다. 또한 실시간 선영 기법은 기본적으로 펜과 잉크를 기준으로 하고 있기 때문에 이를 연필화에 적용하기에는 무리가 따른다.

본 논문에서는 삼차원 메쉬를 입력으로 받아 이를 실시간으로 연필화로 표현해 내는 알고리즘을 제시한다. 이를 위해 본 논문에서는 다음과 같은 기법들을 제시한다.

- **윤곽선 겹쳐 그리기:** 사람이 실제 물체의 윤곽선을 그리는 과정을 살펴 보면, 윤곽선을 그릴 때 정확히 그리지 못하고 약간씩 오차가 존재한다는 사실과 이를 보정하기 위해 윤곽선을 여러 번에 걸쳐 겹쳐 그린다는 사실을 알 수 있다. 본 논문에서는 이러한 특징들을 잘 표현하는 간단하고 빠른 알고리즘을 제시한다 (그림 3).
- **내부 그리기:** 실시간 선영 기법 [20]에서는 주어진 스트로크로부터 방향성을 띈 텍스쳐를 생성한 후 이를 물체에 매핑하여 [19] 삼차원 메쉬를 선영하는 방법을 제시하였다. 본 논문에서는 이를 기본으로 하되 이 과정을 영상 공간 상에서 수행함으로써 보다 간단하고, 또한 그래픽스 하드웨어의 지원을 효과적으로 이용하여 빠르게 연필화의 특징을 모사하는 방법을 제시한다.
- **텍스쳐의 생성:** 실시간에서 연필화를 만들어 내기 위해 본 논문에서는 연필 텍스쳐를 이용한다. 이를 위해 주어진 연필선으로부터 다양한 밝기를 지닌 텍스쳐를 만들어 낸다. 또한 이 때 연필과 종이의 특징을 분석하여 이를 텍스쳐를 만드는 과정에 적용시켰다.
- **GPU 기반 실시간성 보장:** 본 논문에서 제시하는 거의 모든 알고리즘은 프로그램 가능한 그래픽스 하드웨어를 이용하여 구현되었다. 이를 통해 그래픽스 하드웨어의 가속을 효과적으로 이용할 수 있게 되었고 일반적인 복잡도를 지닌 메쉬로부터 초당 약 15에서 62프레임 정도의 렌더링 속도를 보장한다.

*crowlove@postech.ac.kr, http://home.postech.ac.kr/~crowlove
†zelong@postech.ac.kr, http://home.postech.ac.kr/~zelong
‡leesy@postech.ac.kr, http://www.postech.ac.kr/~leesy

제 2 절 관련 연구

Sousa와 Buchanan [23, 24]은 연필화 재료의 다양한 특성들을 분석한 후 이를 시뮬레이션하였다. 연필, 종이, 지우개, 문지르개 등을 분석하고 이를 시뮬레이션한 결과로 만들어진 영상은 실제 연필화에 비교하여도 상당한 수준의 결과를 보여 준다. 그러나 이러한 접근 방법은 많은 계산량이 요구되며 결과적으로 실시간으로 동작하는 데는 무리가 따른다.

Mao 등 [16]은 이차원 영상을 연필화로 변환시키는 방법을 제시하였다. 이 방법은 우선 주어진 영상에서 벡터 필드(vector field)를 계산한 후, 여기에 LIC(Line Integral convolution) 필터를 적용하여 연필화를 만들어 냈다. 본 논문에서는 이차원 영상이 아닌 삼차원 메쉬를 입력으로 받아 연필화를 만들어 낸다.

Winkenbach와 Salesin [27]은 삼차원 메쉬로부터 펜과 잉크로 그려낸 영상을 만드는 방법을 개발하였다. 또한 Lake 등 [11]은 다양한 기법들에 적용될 수 있는 실시간 렌더링 기법을 제시하였다. Saito와 Takahashi [21]는 곡률 방향(curvature direction)을 이용하여 주어진 물체의 표면을 효과적으로 선영할 수 있는 방법을 소개하였으며, Salisbury 등 [22]은 이차원 영상에서 방향 필드(direction field)를 계산한 후 여기에 회전 가능한 텍스쳐를 입혀서 펜과 잉크로 그린 영상을 만들어 냈다. 그러나 위의 방법들은 연필화의 특징을 고려하였다고는 볼 수 없다.

그래픽스 하드웨어가 발전하면서 비사실적 렌더링 분야에서도 다양한 실시간 기법들이 개발되었다. Praun 등 [20]은 실시간 선영 기법에서 주어진 삼차원 메쉬의 상세도(level of detail)를 고려하여 실시간으로 선영하는 방법을 제시하였다. Webb 등 [25]은 이 기법을 확장시켜 삼차원 텍스쳐를 이용하는 방법과 메쉬에 입혀지는 선의 밝기를 조절하는 방법을 제시하였다. Majumder와 Gopi [15]는 그래픽스 하드웨어를 이용하여 삼차원 물체를 목탄화로 표현하는 실시간 기법을 소개하였다. 본 논문에서는 삼차원 메쉬를 실시간에서 연필화로 표현하는 방법을 제시한다. 본 논문에서 제시하는 방법은 기존의 방법에 비해 간단하게 다양한 방향의 선들을 텍스쳐로 메쉬에 입히며 이는 GPU 프로그래밍을 통해 구현되어 빠르게 동작한다.

윤곽선을 그리는 부분에 있어서 Mohr와 Gleicher [17]는 연필화에서 윤곽선을 겹쳐 그린 효과를 모방하기 위해 지터링(jittering)을 이용하였다. Loviscach [13]는 그래픽스 하드웨어를 이용하여 윤곽선을 겹쳐 그리는 방법을 소개하였다. 본 논문에서는 다중 텍스쳐링(multitexturing)과 픽셀 쉐이더를 이용하여 윤곽선에 오차를 주고 겹쳐 그리는 보다 효율적인 방법을 제시한다. 이 방법은 윤곽선 자체를 다루는 것이 아니라 윤곽선이 그려지는 평면을 왜곡시키기 때문에 어떤 복잡한 윤곽선도 매우 효과적으로 다룰 수 있다.

제 3 절 시스템 개요

실제 연필화를 그릴 때는 일반적으로 윤곽선을 그린 후 내부를 칠한다. 본 논문에서 제안하는 방식 역시 이를 이용하여 우선 주어진 메쉬에서 윤곽선을 렌더링한 영상과 내부를 렌더링한 영상을 각각 실시간에서 생성해 낸 후, 생성된 두 결과를 합쳐서 최종 결과 영상을 얻는다 (그림 1).

윤곽선을 그리는 과정은 크게 세 단계를 거친다. 먼저 주어진 메쉬의 깊이 정보(depth map)와 법선 정보(normal map)를 렌더링하여 그 결과를 저장한다. 그런 다음 저장된 깊이, 법선 정보를 이용하여 메쉬의 윤곽선을 찾는다. 윤곽선을 찾아낸 이후에는 찾아낸 윤곽선에 약간의 오차를 준 후 여러 번 겹쳐 그린 다음 여기에 연필 텍스쳐를 입혀서 최종 윤곽선 영상을 얻는다. 윤곽선을 찾고 이를 꾸미는 과정은 모두 텍스쳐를 이용한 영상 공간에서 이루어지며, 이는 픽셀 쉐이더를 이용하여 빠르게 동작 가능하다.

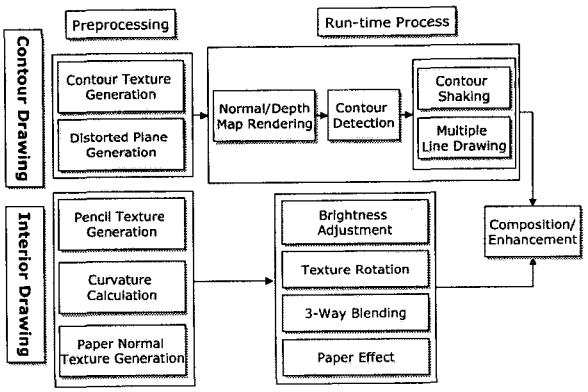


그림 1: 시스템 개요

메쉬의 내부를 그리기 위해서 우선 메쉬의 밝기를 계산한다. 그런 다음 연필화로 표현하기 전에 메쉬의 양감을 강조하기 위해 메쉬의 밝기를 조절하여 명암비를 더 뚜렷하게 한다 [15]. 메쉬를 선영할 때는 연필선의 방향과 특성을 잘 표현하기 위하여 연필 텍스쳐를 메쉬의 곡률 방향을 따라 회전시키고, 이를 세 방향으로 혼합하여 표현한다. 또한 이 과정에서 메쉬가 그려지는 종이의 재질감을 살리기 위해 종이의 질감을 픽셀 단위의 법선 맵으로 표현한 후 이 정보를 연필 텍스쳐를 그릴 때 같이 적용시킨다. 메쉬의 곡률 방향은 버텍스 쉐이더에서 계산된 후 영상 공간에 투영되며 나머지 모든 과정은 한 픽셀 쉐이더에서 처리된다. 결과적으로 내부를 그리는 과정은 한 번의 렌더링 과정을 통해 모두 이루어진다.

윤곽선과 내부를 모두 그리고 난 후에는 윤곽선이 그려진 영상과 내부가 그려진 영상이 각각 텍스쳐로 저장된다. 그러므로 최종적으로 이 두 영상을 픽셀 쉐이더를 이용하여 적절히 합침으로써 우리가 원하는 최종 결과 영상을 얻는다.

위에서 설명한 과정들은 모두 실시간에서 이루어진다. 실시간성을 보장하기 위해서는 여러 정보들이 미리 계산되어 있어야 하며 이는 전처리 과정에서 계산된다. 우선 윤곽선을 그리기 위해 윤곽선의 연필 재질감을 표현할 수 있는 연필 텍스쳐를 만든다. 윤곽선이 그 자체로 이미 방향성을 지니고 있으며 실제로 그려지는 부분은 적은 편이므로, 연필의 재질감을 살리는 정도의 텍스쳐로도 충분하다. 또한 윤곽선에 오차와 겹쳐 그린 효과를 주기 위해 윤곽선이 그려질 왜곡된 평면들을 만들어 둔다. 내부가 그려질 때는 내부에 연필 텍스쳐를 입히는 방법을 사용한다. 그러므로 내부를 표현할 여러 밝기를 지닌 연필 텍스쳐를 만든다. 이 연필 텍스쳐는 연필선의 방향을 나타내기 위해 일정한 방향성을 지니도록 한다. 또한 연필 텍스쳐가 입혀질 방향을 결정하기 위해서 메쉬의 정점 단위 곡률을 계산한 후 이를 메쉬의 각 정점에 저장해 둔다. 마지막으로 메쉬가 그려질 종이의 재질감을 표현하기 위해 각 픽셀에 종이의 질감을 나타내는 법선 벡터가 저장된 종이 텍스쳐를 만들어 둔다.

전처리 과정에서 미리 만들어 둔 정보를 바탕으로 실제

프로그램이 동작할 때는 그래픽스 하드웨어를 이용하여 메쉬를 실시간에서 연필화풍으로 그리는 것이 가능해진다. 만들어진 영상의 윤곽선은 약간의 오차와 여러 번 겹쳐 그린 효과로 인해 실제 사람이 그린 듯한 결과를 보여준다. 또한 내부 역시 연필 텍스쳐와 종이 텍스쳐를 이용하여 연필화의 느낌을 살린다. 각각의 자세한 알고리즘들은 이어지는 각 절에서 상세하게 다룬다.

제 4 절 윤곽선 그리기

4.1 윤곽선 검출

삼차원 메쉬에서 윤곽선을 찾는 방법은 많이 연구되었다. 본 논문에서는 영상 공간에서의 윤곽선 검출 알고리즘을 이용한다. 그 이유는 영상 공간의 알고리즘이 그래픽스 하드웨어를 쉽게 활용할 수 있으며 다양한 영상 처리 기법을 활용할 수 있기 때문이다 [9, 18]. 영상 공간의 알고리즘은 삼차원 메쉬의 법선과 깊이 정보를 이용하여 윤곽선을 찾아 낸다. 잠재적 윤곽선(Suggestive Contour) [4] 역시 영상 공간에서 찾아낼 수 있다.

윤곽선을 찾을 때 픽셀 쉐이더에 광원 정보를 넘겨 주면 찾아낸 윤곽선 부분에 해당하는 삼차원 메쉬의 밝기를 알 수 있다. 이를 이용하여 찾아낸 윤곽선이 메쉬의 밝은 부분은 더 밝고 어두운 부분은 더 어둡게 그려지도록 한다. 이를 수식으로 표현하면 다음과 같다.

$$I_c' = I_c I_l, \quad (1)$$

여기서 I_c와 I_c'는 각각 원래 윤곽선의 밝기와 수정된 윤곽선의 밝기를 나타낸다. I_l은 메쉬의 밝기를 나타내며 이는 광원 방향과 물체의 법선 정보를 이용하여 계산 가능하다. 0을 가장 어두운 밝기로 놓고 1을 가장 밝은 밝기로 놓는다면, 결과적으로 찾아낸 윤곽선은 0에서 1 사이의 값을 지닌다. 윤곽선을 찾고 찾아낸 윤곽선의 화소 값을 결정하는 과정은 픽셀 쉐이더를 이용하여 영상 공간에서 빠르게 구현 가능하다.

4.2 윤곽선 흔들기

4.1절에서 설명한 방법으로 찾아낸 윤곽선들은 사람이 실제로 그린 윤곽선과는 차이가 있다. 사람이 그리는 그림은 항상 오차가 존재하므로 연필선을 그리기 위해서는 찾아낸 윤곽선에 오차를 주어야만 한다 [12, 8]. 우선 사람이 연필로 선을 그리는 과정을 분석하면 다음과 같은 특징들을 찾을 수 있다. 사람은 결코 완벽하게 윤곽선을 그리지 못하므로 사람이 그린 선은 실제의 윤곽선과 비교해 항상 오차가 존재한다. 그러나 사람이 선을 그리다가 스스로 오차를 인식하면 이를 줄이기 위해 원래 그려야 하는 방향으로 선을 그린다. 이러한 특징을 수학적으로는 주기함수로 비슷하게 표현할 수 있다.

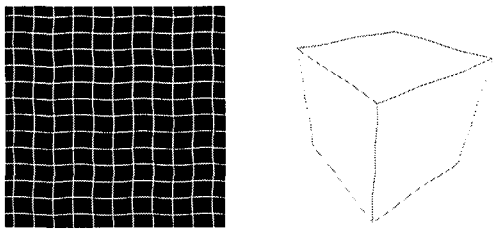
그러므로 본 논문에서는 주기함수 중 하나인 사인(sine) 함수를 이용하여 윤곽선의 오차를 근사한다. 이를 수식으로 나타내면 다음과 같다.

$$y = a\sin(bx+c)+r, \quad (2)$$

이 식에서 a는 더해지는 오차의 범위를, b는 오차의 주기를 결정한다. c는 오차 함수를 약간씩 이동시키며 마지막 r은 오차에 약간의 임의성을 더한다.

수식 (2)에서 정의된 오차를 윤곽선 영상에 더하면 오차가 주어진 윤곽선을 만들 수 있다. 그러나 이를 픽셀 쉐이더에서 바로 구현하기는 쉽지 않다. 우선 윤곽선이 저장된 텍스쳐에서 윤곽선이 있는 지점을 찾아낸 후 여기에 오차를 더하여야 한다. 이 계산은 복잡하고, 또한 픽셀 쉐이더는 현재 화소의 위치 외에 다른 화소에 값을 쓸 수 없다. 그러므로 본 논문에서는 이를 해결하기 위해 윤곽선 자체를 흔드는 대신 윤곽선이 그려질 평면을 왜곡시켜서 윤곽선에 오차를 주는 방법을 제시한다.

전처리 단계에서는 뷰포트를 일정한 크기의 사각형들로 나눈다. 그런 후에 각각의 사각형들의 꼭지점에 윤곽선 영상에 해당하는 텍스쳐 좌표를 할당하여, 결과적으로 여기에 윤곽선 영상을 덧씌워 그리면 윤곽선 영상이 다시 그려지게 한다. 이 과정에서 사각형들의 꼭지점에 할당되는 텍스쳐 좌표를 왜곡시킴으로써 우리가 원하는 왜곡된 윤곽선 영상을 얻을 수 있다. 각각의 사각형 꼭지점에 수식 (2)에서 계산된 오차를 더한 후 여기에 윤곽선 영상을 덧씌워 그려 우리가 원하는 오차가 주어진 윤곽선을 그린다. 그림 2(a)는 텍스쳐 좌표가 왜곡된 평면에 직선들을 그린 그림이고, 그림 2(b)는 이 평면에 윤곽선 영상을 그려서 오차를 준 결과이다.



(a) 왜곡된 평면 (b) 오차가 주어진 윤곽선

그림 2: 윤곽선 흔들기

4.3 윤곽선 겹쳐 그리기

실제 연필화에서 윤곽선을 그리는 과정을 보면 사람은 선을 한 번에 그리지 않고 여러 번 겹쳐서 그린다는 사실을 알 수 있다. 이를 표현하기 위해 왜곡된 평면을 여러 벌 만든 후 같은 윤곽선 영상을 서로 다른 왜곡된 평면에 여러 번 그림으로써 우리가 원하는 겹쳐 그린 윤곽선을 얻는다. 다중 텍스쳐링을 이용하여 사각형의 각 정점에 여러 텍스쳐 좌표를 할당하면, 한 번에 여러 벌의 겹쳐 그린 윤곽선을 얻을 수 있다. 겹쳐 그리는 횟수로는 실험적으로 3~5번 정도가 적당하다.

또한 윤곽선을 겹쳐 그릴 때 여러 선들이 교차하는 부분에는 더 어두운 색을 할당하여 실제로 겹쳐 그린 듯한 효과를 강조한다. 동시에, 전처리 과정에서 만든 윤곽선 텍스쳐를 이 과정에서 투영하여 겹쳐 그린 영상이 연필의 재질감을 갖도록 한다. 윤곽선 영상이 이미 밝기를 지니고 있으므로 이 밝기에 윤곽선 텍스쳐의 밝기를 곱하는 것만으로도 우리가 원하는 윤곽선 영상를 얻을 수 있다.

그림 3에서는 윤곽선 겹쳐 그리기의 예제를 보여 준다. 왼쪽의 그림은 서로 다른 사인 함수로부터 얻어진 왜곡된 윤곽선 영상이고, 오른쪽 그림은 이 윤곽선 영상들이 합쳐진 결과이다. 확대된 영상을 보면, 겹쳐진 윤곽선들이 연필선의 질감을 잘 나타내고 있음을 알 수 있다.

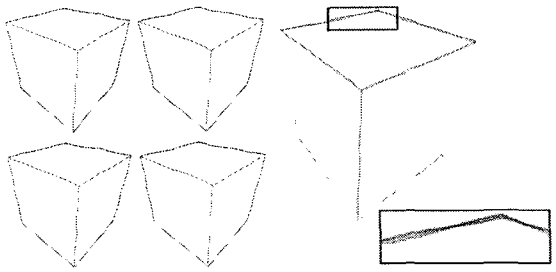


그림 3: 여러 윤곽선 영상과 합쳐진 결과

제 5 절 내부 그리기

연필화에서 물체의 표면을 그릴 때는 연필의 색, 질감, 연필선의 방향, 굵기, 종이의 질감과 문지르기 효과, 지우개 등의 다양한 요소가 적용된다 [23]. 이를 표현하기 위해 전처리 단계에서는 연필의 색과 질감, 그리고 방향성을 지닌 텍스쳐와 종이의 질감을 지닌 텍스쳐를 만들어 낸다. 또한 연필선이 그려질 방향을 결정하기 위한 메쉬의 정점 단위 곡률 방향 역시 계산해 둔다. 메쉬가 그려질 때는 메쉬의 밝기와 곡률 방향 정보를 이용해 텍스쳐를 회전시키고 혼합하여 연필로 그린 듯한 영상을 만들어 낸다. 또한 이 과정에서 종이의 질감 역시 덧입혀 실제 종이에 그린 듯한 결과를 만든다.

5.1 연필 텍스쳐 생성

내부를 그리기 위한 텍스쳐는 연필선의 밝기, 굵기, 질감을 잘 표현할 수 있는 텍스쳐여야 한다. 또한 연필선에는 방향성이 있고 이는 물체의 양감을 강조하는 역할을 하기도 하므로, 연필 텍스쳐에는 방향성 역시 있어야 한다. 본 논문에서는 실시간 선영 기법 [20]의 방법을 기본으로 하면서 픽셀 쉐이더를 이용하여 연필의 특징을 날 나타내는 텍스쳐를 만드는 방법을 소개한다.

실시간 선영 기법에서 물체의 밝기는 그려지는 선들의 밀도로 표현되었다. 그러나 연필화는 연필선의 밀도보다는 연필선의 밝기로 물체의 밝기를 표현한다. 그러므로 텍스쳐를 만들 때 그림 4처럼 연필선이 텍스쳐의 모든 부분에 균일하게 그려지도록 한다. 또한 연필선의 방향을 나타내기 위해 연필선이 일정한 방향으로 위치하게 하여 방향성이 있는 텍스쳐를 만든다. 이 때 연필선이 그려지는 방향에는 약간의 임의성을 주어서 사람이 그린 듯한 효과를 준다.

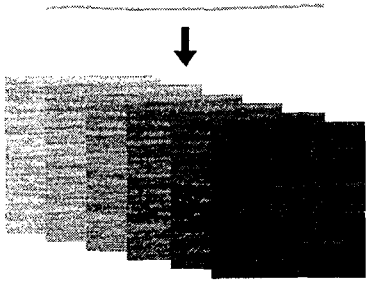


그림 4: 처음 연필선과 만들어진 연필 텍스쳐

연필 텍스쳐의 밝기는 연필선이 얼마나 많이 겹쳐 그려졌는가로 표현된다. 이 때 관찰에 의하면 연필은 여러 번 겹쳐 그려서 색이 어두워질수록 한 번 더 겹쳐 그려도 어두워지는 정도가 적어진다. 그러므로 이를 적용하면 연필 텍스쳐에서 연필선이 그려지는 한 화소의 밝기를 결정하는 식을 다음과 같이 만들 수 있다.

$$\begin{aligned} c_l' &= c_l - \mu_b c_a, \\ c_a &= c_l(1.0 - c_s), \end{aligned} \tag{3}$$

여기서 c_l와 c_l'는 각각 원래의 텍스쳐 색과 연필선이 그려지고 난 이후의 텍스쳐 색을 의미한다. c_s는 그려지는 연필선의 색을 의미하며, c_a는 얼마나 텍스쳐가 어두워질지를 결정한다. μ_b는 텍스쳐가 어두워지는 정도를 결정하는 상수이며 이 값이 클수록 텍스쳐는 더 빨리 어두워진다. 실험에 의하면 0.1에서 0.3 정도가 μ_b의 값으로 적당하다.

실제 연필로 그려진 그림을 관찰해 보면 연필의 흑연이 묻은 부분이 있고 그렇지 않은 부분이 있다는 것을 알 수 있다. 또한 여러 번 연필선을 겹쳐 그릴 때 한 번 흑연이 묻지 않아서 하얗게 남은 부분은 다음에 계속 겹쳐 그려도 하얗게 남아 있다는 사실을 알 수 있다. 이를 텍스쳐를 만드는 과정에 적용하면 수식 (3)에 다음과 같은 조건을 추가할 수 있다.

$$\text{If } c_l \text{ is white enough, } c_a = \mu_w c_a. \tag{4}$$

여기서 μ_w가 작아지면 하얀 부분은 더 잘 보존된다. 일반적으로 μ_w는 0.3에서 0.5 정도의 값이 적당하다.

다양한 밝기의 연필 텍스쳐를 저장하기 위하여 삼차원 텍스쳐를 사용할 수 있다 [25]. 가장 밝은 부분은 일반적으로 아무 것도 그리지 않으므로 삼차원 텍스쳐에서 가장 밝은 부분에는 흰 영상을 저장한다. 본 논문에서는 이 흰 영상을 포함하여 32개의 연필 텍스쳐를 삼차원 텍스쳐로 저장하였다.

5.2 밝기 보정

메쉬의 밝기는 일반적인 쉐이딩 기법을 이용해서 계산할 수 있다. 그러나 계산된 밝기를 그대로 적용하면 메쉬의 양감을 효과적으로 표현하지 못한다. Majumder와 Gopi [15]는 양감을 효과적으로 표현하기 위해 계산된 밝기를 거듭제곱하는 방식을 사용하여 명암비를 강조하는 방법을 제시하였다.

연필화의 경우에는 어두운 부분보다 밝은 부분에서의 명암차가 더 두드러지며, 이를 효과적으로 표현하는 것이 중요하다. 그러므로 본 논문에서는 계산된 밝기의 제곱근을 계산하여 이를 메쉬의 밝기로 이용한다. 이를 통해 물체의 밝은 부분이 더 확장되고 명암의 차이가 부드럽게 계산되도록 한다.

5.3 텍스쳐 회전

그림을 그릴 때, 그려지는 선의 방향은 일반적으로 물체의 최소 곡률 방향을 따른다는 것이 알려져 있다 [6, 7]. 본 논문에서는 이를 이용하여 메쉬의 최소 곡률 방향으로 내부를 선영한다. 우선 전처리 단계에서 메쉬의 정점 단위 곡률 방향을 계산해 둔다. 메쉬의 곡률 방향은 Alliez 등 [1]이 제시한 방법을 이용하여 계산하였다.

메쉬를 그릴 때에는 연필 텍스쳐를 영상 공간에서 매핑시킨다. 본 논문에서는 픽셀 쉐이더를 이용해 영상 공간에 투영된 이차원 삼각형에 텍스쳐를 입혔다. 이 때 메쉬의 각 투영된 정점에 대하여 텍스쳐를 정점의 최소 곡률 방향으로 회전시킨 후 이를 그 정점을 포함하고 있는 면에 매핑시킨다 (그림 5).

정점의 최소 곡률 방향은 삼차원 텍스쳐 좌표로 버텍스 쉐이더에 전달된다. 버텍스 쉐이더에서는 OpenGL 투영 행렬을 이용하여 전달된 곡률 방향을 영상 공간으로 변환한 후 이를 픽셀 쉐이더에 전달한다. 픽셀 쉐이더에서는 영상 공간에 투영된 곡률 방향을 이용하여 연필 텍스쳐의 회전각 θ를 계산할 수 있다. 그러나 실제로 텍스쳐 자체를 회전시킬 수는 없으므로 그 대신 텍스쳐 좌표를 $-\theta$만큼 회전시켜서 우리가 원하는 메쉬의 최소 곡률 방향으로 영상에 텍스쳐를 입힌다.

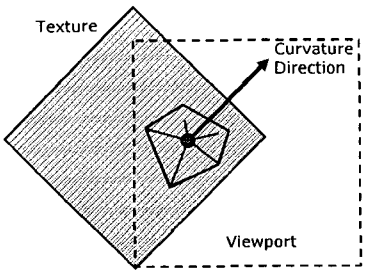


그림 5: 텍스쳐 회전

5.4 삼방향 블렌딩

각 정점 단위로 텍스쳐를 회전시키게 되면 메쉬의 각 삼각형에는 서로 다른 방향을 가진 세 개의 연필 텍스쳐가 입혀진다. 여기서 각 텍스쳐의 방향은 해당하는 정점의 최소 곡률 방향을 따른다. 실제로 이 세 개의 연필 텍스쳐를 메쉬의 각 삼각형에 입히기 위해 본 논문에서는 삼각형을 그릴 때 각 정점에 삼각형의 최소 곡률 방향을 다중 텍스쳐링을 이용해 모두 넣었다. 그런 후 픽셀 쉐이더에서는 각 정점의 곡률 방향으로 세 번 텍스쳐를 그린 후 이 값을 블렌딩하여 물체의 내부를 표현한다 (그림 6).

삼방향 블렌딩으로 메쉬의 내부를 그린 결과를 보면, 원기둥의 표면과 같이 곡률이 일정한 부분에서는 거의 일정한 방향으로 연필선이 위치한다. 또한 복잡한 형태를 지닌 물체에서는 곡률의 변화에 따라 연필선의 방향이 부드럽게 변하며 서로 다른 방향을 지닌 연필선이 메쉬의 표면에 겹쳐져서 그려진다. 이는 사람이 실제로 연필화를 그릴 때 연필선을 그리는 특징을 잘 모사하며 결과적으로 다양한 형태의 메쉬에 대해 자연스러운 영상을 만들어 낸다.

그림 6: 삼방향 블렌딩

여기에 더하여, 실제 연필화를 보면 어두운 부분에서는 더 많은 선들을 여러 방향으로 겹쳐 그린다는 것을 알 수 있다. 이를 적용하여 메쉬의 어두운 부분을 표현할 때에는 우선 최소 곡률 방향으로 메쉬를 그린 후 최대 곡률 방향으로 한 번 더 그려 그 값을 더해 준다 (그림 7).

5.5 종이 질감 표현

연필화는 기본적으로 종이에 그려지기 때문에 종이의 질감 역시 연필화의 결과에 영향을 미친다. 그러므로 이를 표현하기 위해 전처리 단계에서 종이의 질감을 나타내는 텍스쳐를 만든다. 노이즈를 이용해 높이 맵을 만들어 종이의 질감을 표현하는 방법이 Curtis 등 [3]에 의해 소개되었다. 본 논문에서는 이를 이용해 종이의 높이 맵을 만든 후 이것을 이용해 종이 텍스쳐에 각 화소 단위로 법선 벡터를 만들어 이를 종이의 결로 정의한다.

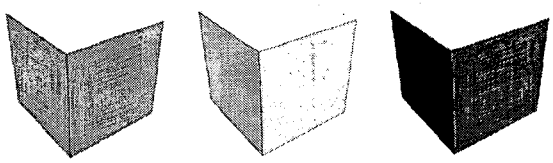


그림 7: 어두운 부분의 표현

내부를 그릴 때는 만들어진 종이 텍스쳐를 이용해 종이의 결이 연필 텍스쳐의 방향과 거치는 부분에서는 텍스쳐의 색을 더 어둡게 하고, 그 반대인 경우에는 텍스쳐의 색을 밝게 함으로써 종이의 질감을 표현한다. 이는 다음과 같은 식으로 표현할 수 있다.

$$c_t' = c_t - \mu_p(\mathbf{d}\cdot\mathbf{n}) \quad (5)$$

c_t와 c_t'는 각각 원래의 텍스쳐 색과 변화된 텍스쳐 색이다. $\mathbf{d}$는 텍스쳐가 입혀지는 곡률 방향이며, $\mathbf{n}$는 종이 텍스쳐의 법선 벡터 방향이다. μ_p는 종이의 질감이 얼마나 표현될 것인가를 결정하는 상수이며 실험적으로 0.0에서 0.1 정도의 값이 적당하다.

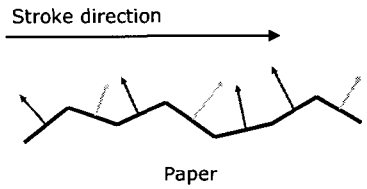


그림 8: 종이 질감 표현

제 6 절 합성 및 후처리

메쉬의 윤곽선과 내부를 따로따로 그린 후에는 결과로 윤곽선이 그려진 영상과 내부가 그려진 영상이 생긴다. 그러므로 픽셀 쉐이더에서 이 두 영상을 합쳐서 최종 영상을 만든다.

사람들이 그림을 그릴 때는 일반적으로 윤곽선을 먼저 그린 후 내부를 그리므로, 이를 적용하여 우선 윤곽선 텍스쳐의 색을 입힌 후 표면 텍스쳐를 입힌다. 이 때 만약 윤곽선이 칠해진 부분에 표면 텍스쳐가 칠해진다면 표면 텍스쳐 색의 일부분만이 적용되게 하여 표면의 색이 적용되면서도 윤곽선이 나타나게 한다.

5.2절에서는 메쉬의 밝기를 조절하여 내부 영상의 명암비를 뚜렷하게 하였다. 이와 비슷하게 픽셀 쉐이더를 이용해 결과 영상의 밝기를 보정할 수 있다. 수식 (6)은 결과 영상의 밝은 부분이 보다 더 밝아지게 함으로써 명암비를 강화한다.

$$c_i' = c_i(1.0 - \mu_c c_i)\mu_c c_i, \quad (6)$$

여기서 c_i와 c_i'는 각각 원래의 색과 보정된 색을 의미한다. μ_c는 명암비를 보정하는 상수로 0에서 1 사이의 값을 가진다.

마지막으로 결과 영상은 하얀 표면 위에 그려졌으므로 여기에 연필의 색감을 지닌 텍스쳐를 덧입힌다. 이 텍스쳐는 종이의 재질을 표현하는 텍스쳐와 무관하며 단지 종이의 색감을 덧입힌다. 연필선이 많이 칠해져서 어두운 부분일수록 종이의 색은 보이지 않고 밝은 부분일수록 종이가 뚜렷이 보이므로, 이를 적용하면 최종 영상에 종이 텍스쳐는 다음과 같이 덧입혀진다.

$$c_i' = c_i c_p, \quad (7)$$

c_i와 c_i'는 각각 원래의 색과 종이 텍스쳐가 입혀진 색이며 c_p는 종이 텍스쳐의 색이다.

제 7 절 결과

본 논문은 윈도우즈에서 OpenGL과 OpenGL Shading Language를 이용하여 구현되었다. 본 논문의 예제는 Pentium IV 630의 연산장치와 2G 메모리를 지닌 컴퓨터에서 만들었으며, 사용된 그래픽 카드는 256M 비디오 메모리의 nVIDIA GeForce 7800GTX이다.

그림 9(a)는 윤곽선을 그린 결과이다. 사람이 실제로 윤곽선을 그리는 과정을 모사하기 위해 약간씩의 오차가 주어진 윤곽선이 겹쳐 그려진 것을 확인할 수 있다. 그림 9(b)는 최종적으로 그려진 결과 영상의 일부분을 확대한 그림이다. 연필선이 메쉬의 곡률 방향을 따라 자연스럽게 그려지는 것을 확인할 수 있다. 또한 메쉬의 밝기에 따라 밝기가 부드럽게 변하는 모습 역시 확인할 수 있다.

그림 10에서는 같은 물체를 서로 다른 밝기의 두 연필선에서 만들어진 연필 텍스쳐로 각각 표현한 결과이다. 그림 11의 오른쪽은 왼쪽에서 만들어진 영상에 명암비를 보정하여 만든 결과이다. 그림 10, 11에서는 같은 메쉬로부터 서로 다른 느낌의 연필화를 그리 어렵지 않게 얻어낼 수 있음을 보여 준다.

그림 12, 13은 본 논문에서 구현한 몇몇 결과 영상들의 예제이다.

표 1은 본 논문에 나온 예제들의 렌더링 속도이다. 몇만 개 정도의 다각형을 가진 메쉬에서 충분한 실시간성을 보장하고 있음을 알 수 있다. 복잡한 메쉬는 렌더링 속도가 느리지만, 본 논문에서 제시한 방법은 오히려 어느 정도 단순한 메쉬에서 더 좋은 결과를 보여 준다. 거의 모든 과정이 픽셀 쉐이더에서 영상의 화소 단위로 이루어지므로 메쉬가 반드시 복잡할 필요는 없으며, 복잡한 메쉬는 곡률 방향의 복잡한 변화로 인해 결과가 더 안 좋아질 수도 있다. 그러나 일반적으로 사용하는 적당한 복잡도의 메쉬에서는 실시간성과 좋은 결과를 보여 준다.

그림	모델	삼각형 개수	평균 FPS
그림 9 (a)	팬디스크	12,946	45
그림 9 (b)	지구본	9,728	47
그림 10	수류탄	17,464	44
그림 11	비너스	1,416	62.5
그림 12	드릴	41,210	16

표 1: 렌더링 성능

제 8 절 결론 및 향후 과제

본 논문에서는 실시간으로 주어진 삼차원 메쉬를 연필화로 그리는 방법을 제시하였다. 이를 위해 본 논문에서는 연필화의 특성을 분석한 후에 이를 적용하여 메쉬를 연필화로 그려 냈다. 본 논문에서 제시하는 거의 모든 알고리즘은 버텍스/픽셀 쉐이더와 다중 텍스쳐링을 이용하여 그래픽스 하드웨어의 가속을 충분히 활용할 수 있다.

본 논문에서는 픽셀 쉐이더를 이용하여 영상 공간에서 윤곽선과 내부를 그린다. 이는 빠르고 그래픽스 하드웨어의 지원을 활용하기 쉬우나 앨리어싱(aliasing)의 문제가 있다. 또한 본 논문에서는 시간 개연성(temporal coherence) 문제에 대해서는 특별한 고려를 하고 있지 않다. 그러나 실제 구현된 결과 애니매이션에서는 어느 정도의 시간 개연성이 존재함을 알 수 있다. 연필 텍스쳐가 이차원 영상 위에서 입혀지지만 매 순간마다 텍스쳐의 방향이 메쉬의 곡률 방향을 따라 회전하므로 기존의 텍스쳐 투영 방식의 문제점인 샤워 도어(shower-door) 효과는 나타나지 않는다. 그러나 애니매이션에서 나타날 수 있는 문제점을 해결하기 위해서는 시간 개연성을 고려하는 알고리즘의 개발이 필요할 것이다.

향후 과제로는 우선 현재 가지고 있는 문제점들의 해결이 있을 것이다. 또한 지우개나 문지르기 효과 등의 보다 더 많은 연필화의 특징들 역시 고려되어야 할 사항이다. 크로키(croquis)와 같은 보다 더 추상화된 회화 양식의 표현 방법의 개발은 어렵지만 흥미로운 연구 주제가 될 수 있을 것이다.

감사의 글

논문을 작성하는 데 많은 도움을 준 이윤진에게 감사를 드립니다. 본 논문에서 사용된 소화전 모델과 수류탄 모델은 Amazing3D(www.amazing3d.com)에서 구하였습니다. 본 연구는 BK21 사업을 통하여 포항공과대학교 전자.컴퓨터공학부에 주어진 교육부의 재정 지원, 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 및 지원사업의 지원을 얻어 수행되었습니다.

참고 문헌

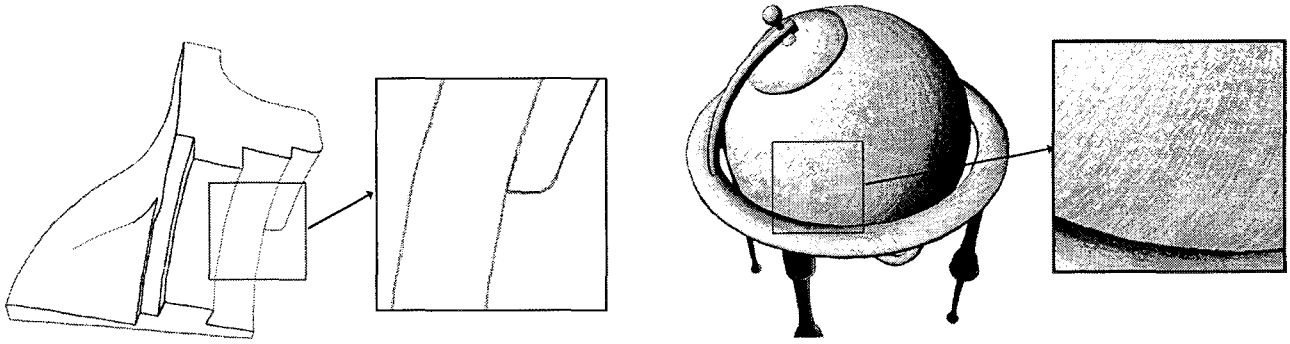
[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Computer Graphics (Proc. of SIGGRAPH 2003)*, pages 485–493, 2003.

[2] T. W. Bleser, J. L. Sibert, and J. P. McGee. Charcoal sketching: Returning control to the artist. *ACM Transactions on Graphics*, 7(1):76–81, January 1988.

[3] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 421–430, Aug. 1997.

[4] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848–855, July 2003.

[5] O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 13–18, July 2000.



(a) 팬디스크(Fandisk)의 윤곽선 (b) 지구본의 렌더링 결과

그림 9: 윤곽선과 내부 그리기

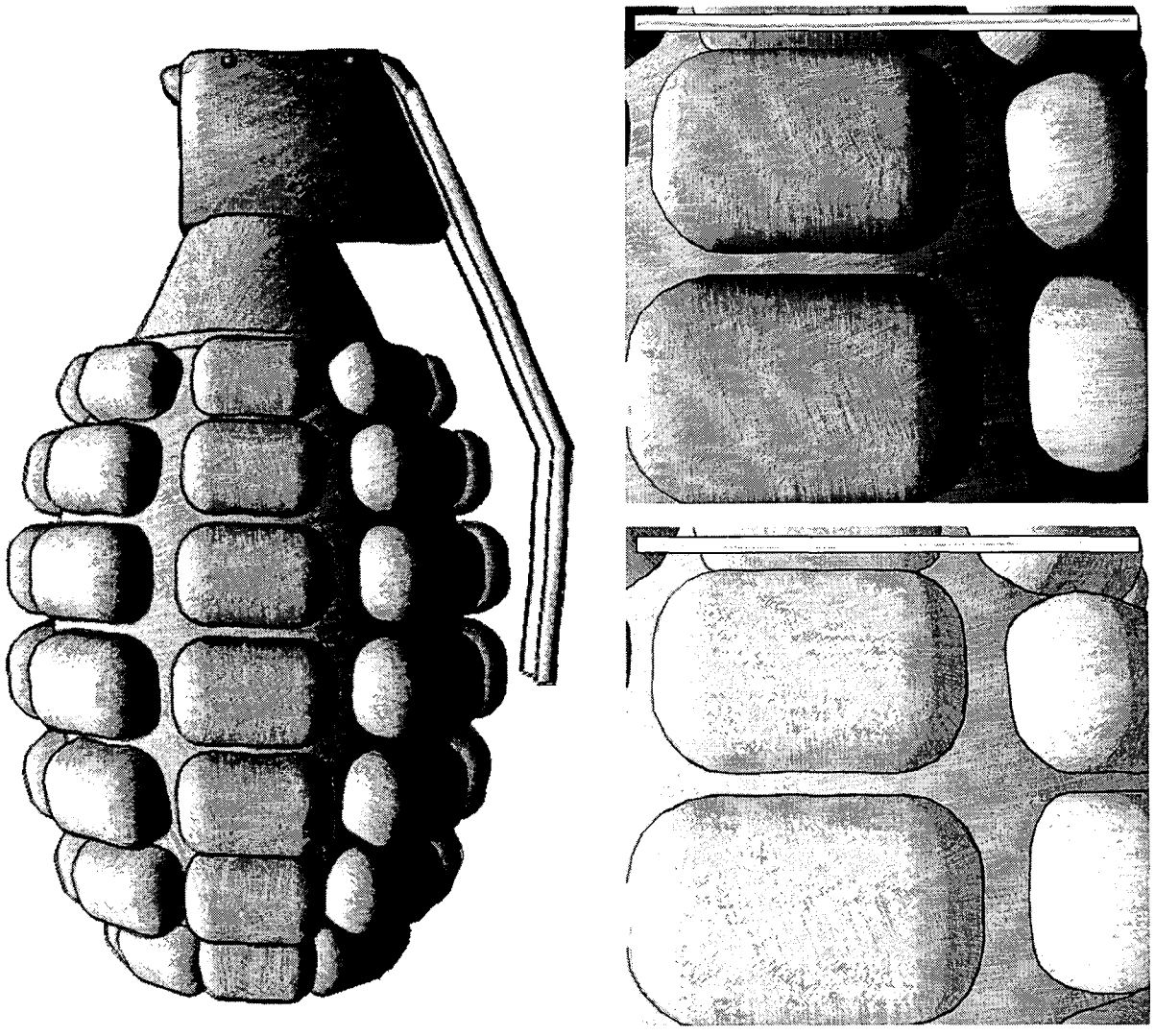


그림 10: 서로 다른 연필선에서 만들어진 테스쳐로 그린 수류탄: (왼쪽) 렌더링 결과; (오른쪽 위) 왼쪽 그림을 확대한 모습; (오른쪽 아래) 밝은 텍스쳐로 그린 결과

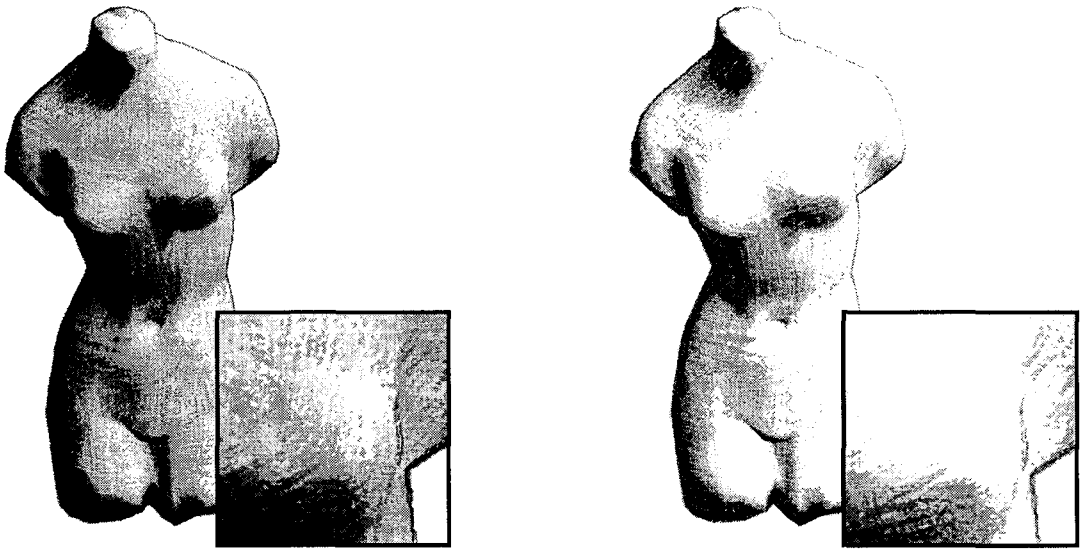


그림 11: 비너스(Venus): (왼쪽) 렌더링 결과; (오른쪽) 명암비를 강조한 결과

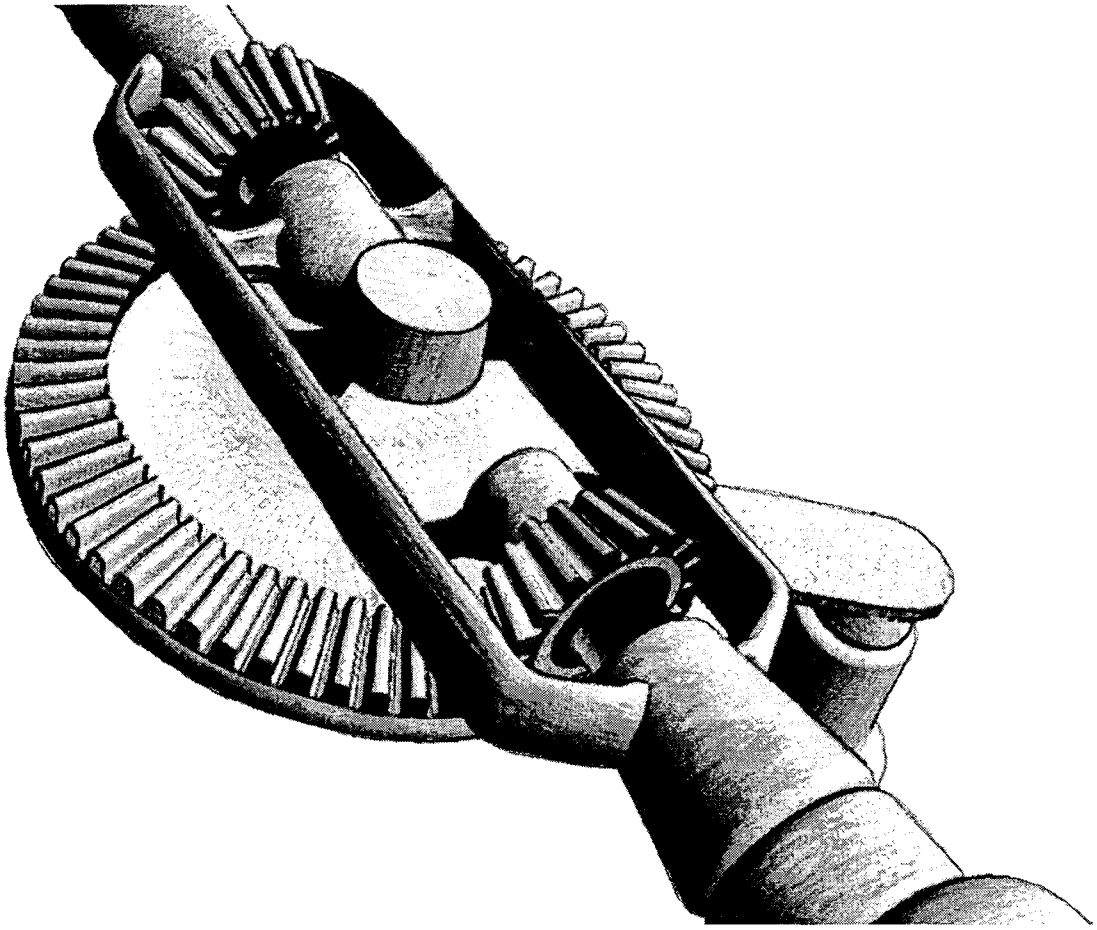


그림 12: 드릴(Drill)

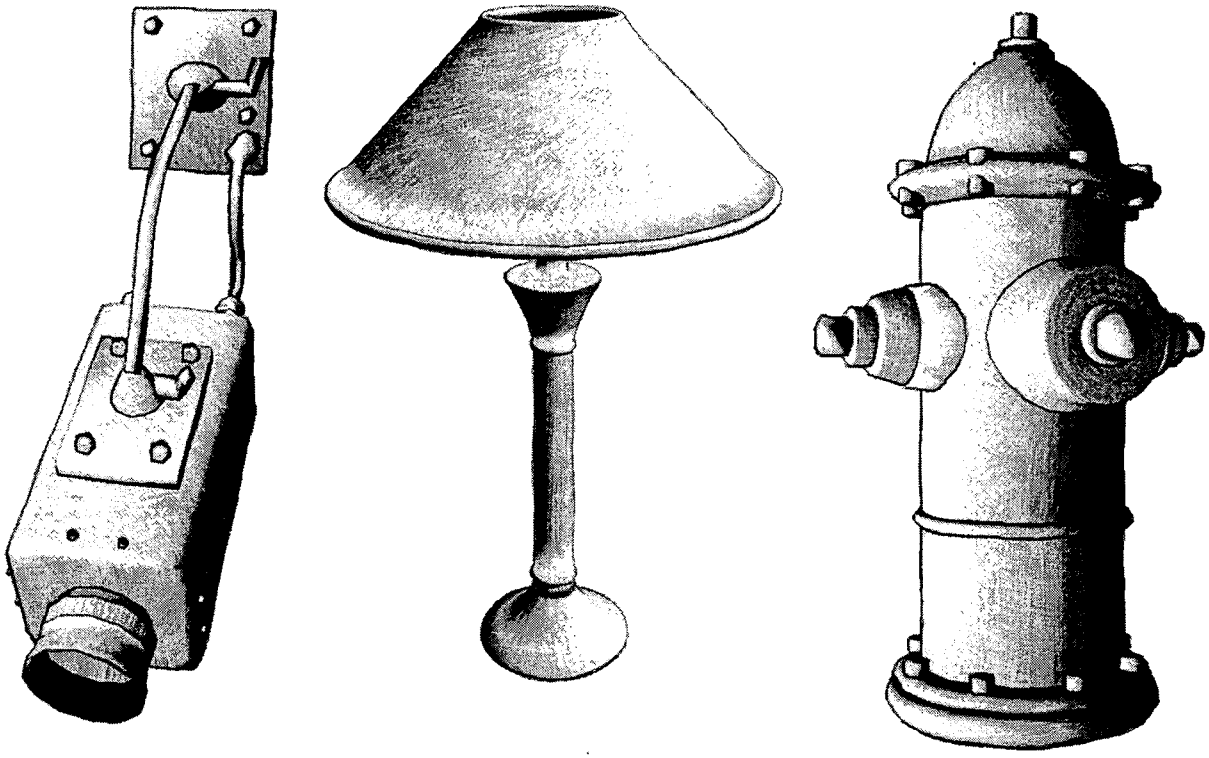


그림 13: 렌더링 결과 예제: (왼쪽) 사진기; (가운데) 램프; (오른쪽) 소화전

[6] A. Girshick, V. Interrante, S. Haker, and T. Lemoine. Line direction matters: an argument for the use of principal directions in 3d line drawings. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 43–52, New York, NY, USA, 2000. ACM Press.

[7] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 517–526, July 2000.

[8] S. N. Ho and R. Komiya. Real time loose and sketchy rendering in hardware. In *Proceedings of the 20th Spring Conference on Computer Graphics*, pages 83–88, Budmerice, Slovakia, 2004.

[9] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A developer's guide to silhouette algorithms for polygonal models. *IEEE Comput. Graph. Appl.*, 23(4):28–37, 2003.

[10] T. V. Laerhoven and F. V. Reeth. Real-time simulation of watery paint. *Computer Animation and Virtual Worlds*, 16(3-4):429–439, September 2005. Special Issue: CASA 2005.

[11] A. Lake, C. Marshall, M. Harris, and M. Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering*, pages 13–20, June 2000.

[12] J. Loviscach. Rendering artistic line drawings using off-the-shelf 3-d software. In *Eurographics 2002 Short Paper*, pages 125–130, 2002.

[13] J. Loviscach. Stylized haloed outlines on the gpu. In *SIGGRAPH 2004 Poster Session*, 2004.

[14] T. Luft and O. Deussen. Interactive watercolor animations. In *13th Pacific Conference on Computer Graphics and Applications (PG'05)*, Macau, October 12-14th 2005.

[15] A. Majumder and M. Gopi. Hardware accelarated real time charcoal rendering. In *2nd International Symposium on Non-Photorealistic Animation and Rendering (NPAR'02)*, Annecy, France, June 3-5 2002.

[16] X. Mao, Y. Nagasaka, and A. Imamiya. Automatic generation of pencil drawing from 2d images using line integral convolution. In *Proceedings of the 7th International Conference on Computer Aided Design and Computer Graphics CAD/GRAPHICS2001*, pages 240–248, August 2001.

[17] A. Mohr and M. Gleicher. Non-invasive, interactive, stylized rendering. In *2001 ACM Symposium on Interactive 3D Graphics*, 2001.

[18] M. Nienhaus and J. Doellner. Edge-enhancement . an algorithm for real-time non-photorealistic rendering. *WSCG'2003*, 11(1), February 3-7 2003.

[19] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[20] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 579–584, Aug. 2001.

[21] T. Saito and T. Takahashi. Comprehensible rendering of 3D shapes. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):197–206, August 1990. Held in Dallas, Texas.

[22] M. P. Salisbury, M. Wong, J. F. Hughes, and D. H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *Proceedings of SIGGRAPH '97*, 1997.

[23] M. C. Sousa and J. W. Buchanan. Computer-generated graphite pencil rendering of 3D polygonal models. *Computer Graphics Forum*, 18(3):195–208, September 1999.

[24] M. C. Sousa and J. W. Buchanan. Observational models of graphite pencil materials. *Computer Graphics Forum*, 19(1):27–49, March 2000.

[25] M. Webb, E. Praun, A. Finkelstein, and H. Hoppe. Fine tone control in hardware hatching. In *NPAR 2002: Second International Symposium on Non Photorealistic Rendering*, pages 53–58, June 2002.

[26] B. Wilson and K.-L. Ma. Representing complexity in computer-generated pen-and-ink illustrations. In *NPAR 2004: Third International Symposium on Non Photorealistic Rendering*, June 2004.

[27] G. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pages 91–100, July 1994.

[28] G. Winkenbach and D. H. Salesin. Rendering parametric surfaces in pen and ink. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 469–476, Aug. 1996.