다단계 메쉬를 이용한 점진적 실루엣 렌더링

김성수
한국전자통신연구원
kss62937@etri.re.kr

Progressive Silhouette Rendering using Level-of-Detail Meshes

Sung-Soo Kim
Computer and Software Technology Research Laboratory
Electronics and Telecommunications Research Institute

요 약

실루엣 에지를 찾거나 디스플레이하는 것은 컴퓨터 비전에서부터 비사실적인 렌더링(nonphotorealistic rendering)에 이르기까지 여러 응용분야에서 중요한 의미를 가진다. NPR(nonphotorealistic rendering)에서 가장 초점을 두는 것은 컴퓨터로 렌더링된 결과가 사람이 그린듯한 효과를 줄 수 있느냐에 있다.

본 논문에서는 점진적 메쉬와 실루엣 에지 렌더링을 통해 사람이 임의의 객체를 점진적으로 스케치하는 효과를 보여줄 수 있는 방법을 제시한다. 다단계 메쉬 생성은 대용량의 메쉬를 효율적으로 다루기 위한 것으로 점진적인 렌더링이나 전송, 상세도 제어등에 이용되는 기술이다. 메쉬 간략화에서는 볼륨과 경계를 보존할 수 있는 간략화 기법을 이용하였다. 간략화 수행시 원본 메쉬에 대한 히스토리를 저장할 수 있는 점진적 메쉬(Progressive Mesh)구조를 구성한 뒤 기본메쉬를 점진적으로 상세하게 해 나감으로써 실루엣 렌더링을 수행하게 된다. 실루엣 에지에 대해 파라미터화된 브러쉬 함수(Parameterized Brush Functions)를 적용하여 다양한 스타일로 점진적인 스케칭 효과를 낼 수 있는 기법을 제시한다.

1. 서론

컴퓨터 그래픽스의 흐름은 크게 물리학을 기반으로한 사물이나 현상을 더욱 사실적(photorealistic rendering)으로 표현하는데 중점을 두는 분야와 비사실적(nonphotorealistic rendering:NPR)이고 사람이 그린듯하게 표현하는데 초점을 두는 두 분야로 나눌 수 있다.

비사실적표현은 사진과 같이 사실적으로 나타내는 것보다 펜이나 잉크,목탄등을 사용하여 그리는 대상을 추상화시켜 간략하게 나타내기때문에 전체적인 윤곽에 집중할 수 있게 만든다. 예를 들어, 거의 모든 의학교과서는 일러스트레이션을 이용하여 사람의 신체와 같은 복잡한 구조를 알아보기 쉽게 나타내고 기계장치의 수리 메뉴얼과 같은 책은 사진을 이용하기보다는 일러스트레이션으로 간략하고 명확하게 나타낸다. 그리고 어린이들을 위한 책에서는 좀 더 코믹하고 만화같은 느낌으로 재미를 더해주는 카툰방식을 이용하여 표현하고 있다.

최근 비사실적 표현을 위한 시스템이 많이 만들어졌는데 그 시스템들의 입력방식에 따라 크게 2가지로 나누어 진다. 하나는 3차원 모델을 바탕으로 컴퓨터로 렌더링(rendering)하는 것이고, 다른 하나는 2차원 이미지를 바탕으로 사용자가 개입하여 그리거나 이미지 프로세싱 기술을 이용하여 그리는 것이다.

본 논문에서는 대용량 3차원 모델을 바탕으로 메쉬 간략화로 구해진 점진적 메쉬에 대하여 실루엣 에지(silhouette edge)를 추출해 다양한 브러쉬 스타일로 렌더링하는 알고리즘을 제안한다.

2. 관련 연구

최근 몇 년간 메쉬 간략화를 통한 LOD 계산에 관한 많은 알고리즘들이 제안되었다. 메쉬 간략화의 목표는 적은 정점으로 면을 나타내더라도 원본 모델의 위상을 잘 유지하도록 좋은 근사(approximation)를 제공하는 것이다. 메쉬 간략화의 장점은 대용량 모델의 저장공간을 줄일 수 있고, 자료 구조 구축시간을 줄이며, 빠른 속도로 모델을 가시화할 수 있다는 것이다. 주요한 메쉬 간략화 알고리즘으로는 vertex clustering, edge collapses[2], vertex removal, quadric error metrics[1], simplification envelopes, memoryless simplification[4] 등이 있다.

본 논문에서는 이전에 많은 연구들에서 사용한 에지 축약 방법을 사용하여 메쉬 볼륨과 경계 보존을 고려하여 메모리를 효율적으로 이용하고 처리 속도를 개선한 메쉬 간략화 기법을 제시한다.

실루엣 에지를 추출해내는 것은 오랫동안 NPR과 일러스트레이션에서 많이 이용되었는데 실루엣 에지를 추출해 내는 가장 큰 목적은 3차원 모델을 컴퓨터를 통해 비사실적인 그림으로 표현해 내는데 있다.

Salesin[7]이 3차원 정보를 이용하여 스트록과 여러 문양을 바탕으로 컴퓨터로 펜과 잉크 일러스트레이션을 하는 방법을 제시하였고 Markosian[3]은 인접 정보를 가지고 있는 정적 다면체 모델(static polyhedral model)을 실시간으로 다양한 NPR 스타일로 표현하고 수행시간을 향상시켰다.

Rossignac[6]의 방법은 인접 정보가 필요하지 않고 와이어 프레임 표현에서는 depth-buffer를 하얀 장면(scene)으로 먼저 렌더링 한 후 검은색의 와이어 프레임 모드로 렌더링한다. 즉, 표면이 채워져있는 보이지 않는 다각형(back-facing) 대신에 보이지 않는 다각형의 에지를 렌더링하는 방법이다. 그렇게 되면 모든 다각형들이 와이어 프레임으로 렌더링된다. 이 depth function을 'Less than or Equal'이라 한다. Raskar[5]는 Rossignac의 depth function방법을 사용하여 라인 두께를 불규칙하게 증가시켰다.

본 논문에서는 제안한 간략화 방법을 통해 얻어진 점진적 메쉬(progressive mesh)를 바탕으로 실루엣 에지를 찾아 다양한 스타일로 사람이 임의의 객체를 점진적으로 스케치한 듯한 효과를 줄 수 있는 실루엣 에지 렌더링 알고리즘을 제시한다.

3. 점진적 실루엣 에지 렌더링

점진적인 실루엣 렌더링을 위해서 제시하는 알고리즘은 크게 메쉬 간략화 알고리즘과 실루엣 렌더링 알고리즘이 있다. 먼저 실루엣 렌더링을 위해 점진적 메쉬를 생성하게 되는데, 이는 메쉬 간략화를 통해 전처리과정에서 얻어진다. 그림 1은 제시하는 점진적 실루엣 렌더링 시스템의 처리 흐름을 나타낸다.

3.1 메쉬 간략화 알고리즘

본 논문에서는 메쉬 간략화의 연산으로 에지축약(edge collapse)을 이용하여 메쉬를 간략화한다(그림 2). 메쉬 간략화를 수행하기 위해 다음과 같은 두 가지 주요한 과정이 있다.

- 에지축약 우선순위 결정
- 에지축약 이후 새로운 정점 위치선정

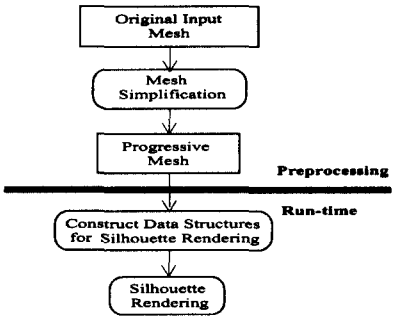


그림 1: 점진적 실루엣 렌더링을 위한 처리 흐름

본 논문에서는 위 두가지 사항을 결정하는 데 있어 볼륨(volume)과 면 영역 정보를 이용하여 볼륨과 경계를 보존할 수 있는 방법을 이용하였다.

먼저 모델의 볼륨을 변화를 최소화하는 에지를 결정하고 모델의 볼륨을 보존하기 위해 정점의 위치를 식 (1)과 같이 설정하였다[4].

$$\sum_i V(v, v_0^{t_i}, v_1^{t_i}, v_2^{t_i}) = \sum_i \frac{1}{6} \begin{vmatrix} v_x & v_{0_x}^{t_i} & v_{1_x}^{t_i} & v_{2_x}^{t_i} \\ v_y & v_{0_y}^{t_i} & v_{1_y}^{t_i} & v_{2_y}^{t_i} \\ v_z & v_{0_z}^{t_i} & v_{1_z}^{t_i} & v_{2_z}^{t_i} \\ 1 & 1 & 1 & 1 \end{vmatrix} = 0 \quad (1)$$

위 식을 v에 대해 풀면, 다음과 같은 볼륨 보존 비용함수를 얻을 수 있다.

$$F_{vol}(e, v) = \sum_i \left(v_0^{t_i} \times v_1^{t_i} + v_1^{t_i} \times v_2^{t_i} + v_2^{t_i} \times v_0^{t_i}\right)^T v$$

$$= \sum_i {n_i}^T v = \sum_i \left[v_0^{t_i}, v_1^{t_i}, v_2^{t_i}\right]$$

여기서, n_i는 삼각형 t의 법선 벡터가 된다.

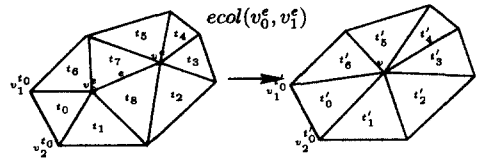


그림 2: 에지축약(Edge Collapse) 연산

모델이 지닌 경계(boundary)의 모양을 보존하기 위해 볼륨 보존에서 사용했던 방법과 유사한 알고리즘을 이용한다. 여기서는 에지 축약 이후 경계의 영역 변화를 최소로 하는 정점을 찾을 수 있다. 따라서, 평면 경계의 영역(area) 변화를 평가하는 비용함수 F_{boun}를 설정할 수 있다.

$$F_{boun}(e, v) = \sum_i A(v, v_0^{e_i}, v_1^{e_i}) =$$

$$\left\| \sum_i \frac{1}{2} (v \times v_0^{e_i} + v_0^{e_i} \times v_1^{e_i} + v_1^{e_i} \times v) \right\|$$

여기서, 각 항은 경계 평면과 수직한 벡터가 된다. 각 벡터의 방향은 변화된 경계 영역의 부호를 결정한다.

메쉬 간략화를 위해 에지 축약의 우선 순위 결정은 중요한 과정이다. 원래의 메쉬에서 에지를 제거하기 위한 제거기준으로는 에지의 길이 $L(e)$, 두 정점의 차수(degree)의 합 $d(v)$가 있다. method)을 적용한 것이다. 이러한 기준들에 따라 각 에지에 해당하는 값을 구하는 함수를 비용함수(cost function)로 정의한다. 에지 우선 순위를 결정하기 위한 비용함수 F_C를 최소로 하는 에지를 우선으로 한다. F_C는 각각의 비용함수를 선형조합한 것이다.

$$F_C = \lambda F_{vol} \cdot d(v) + (1 - \lambda) L(e)^2 \cdot F_{boun} \quad (2)$$

여기서, λ는 임계치 값(threshold value)으로 실험에서는 $\lambda = \frac{1}{3}$로 설정하였다.

제시된 비용함수에 의해 간략화된 메쉬는 아래와 같은 점진적 메쉬(progressive mesh)[2] 자료구조로 저장된다.

```
struct CPMesh {
    CMesh base_mesh;        // base mesh M^0
    Array<Vsplit> vsplits; //{vsplit_0, ... ,vsplit_{n-1}}
    int full_nvertices;     // number of vertices in M^n
    int full_nwedges;       // number of wedges in M^n
    int full_nfaces;        // number of faces in M^n
};
```

여기서 에지축약의 역연산인 Vsplit(*vertex split*) 연산들을 수행함으로 써 기본메쉬에서 원본메쉬로 점진적으로 세밀화해 나갈 수 있다. 그림 3은 간략화된 기본메쉬로부터 점진적으로 세밀화해 나가면서 실루엣 에지를 랜더링한 결과를 보여주고 있다.

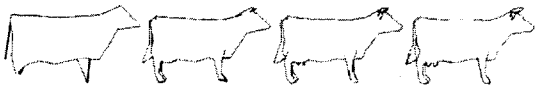


그림 3: 점진적인 실루엣 랜더링

3.2 실루엣 에지 랜더링 알고리즘

본 논문에서 제안하는 실루엣 랜더링 시스템(*Stylized Silhouette edge Renderer, SSR*)에서는 효과적인 랜더링 결과를 얻기 위해 아래와 같은 단계를 수행하게 된다.

1. 시점(view point)에서 보이는(visible) 실루엣 에지를 찾는다.
2. 위 단계에서 찾아진 실루엣 에지들을 파라미터화한다.
3. 실루엣의 탄젠트(tangent) 벡터와 법선(normal) 벡터사이의 거리를 사용하여 실루엣 브러쉬 함수를 정의한다.
4. 위 파라미터화된 브러쉬 함수를 이용하여 여러가지 스타일의 실루엣을 랜더링한다.

본 논문에서는 수정된 Appel 알고리즘을 사용하였고, 실루엣 에지 추출을 위해 한 에지에 인접한 두 삼각형의 법선벡터 계산을 빠르게 하기 위해 *Half Edge* 자료구조를 이용하여 구현하였다.

브러쉬 함수를 이용하여 랜더링하기 위해서 주어진 입력 메쉬에 대해 해당 메쉬에 대한 Half Edge 정보를 생성한다. 생성을 위해서 필요한 정렬과정을 거치므로 Half Edge 정보를 생성하는 데에는 $O(n \log n)$ 시간 복잡도가 요구된다.

실루엣 에지 파라미터화 단계에서는 얻어진 모든 실루엣 에지에 대해 하나씩 파라미터화하게 된다. 각 실루엣 에지의 끝점을 각각 다른 크기로 조절하여 하나의 선분(line)으로 파라미터화하게 된다. 이렇게 함으로써, 실제 사람이 스케치하는 듯한 랜덤한 효과를 얻을 수 있다. 브러쉬 함수(*Brush Function*)는 아주 기본적인 레벨(low level)에서 특정 브러쉬가 어떻게 에지를 표현할 것인가를 정의한 함수를 말한다. 전 단계에서 랜덤한 효과를 얻기 위해 파라미터화 된 각 실루엣 에지에 대해 법선 벡터 뿐만 아니라 에지의 탄젠트 벡터를 파라미터화 할 수 있다. 탄젠트와 법선 벡터는 각 정점에 대한 벡터 옵셋(offset)에 대한 브러쉬 함수와 함께 사용되어 진다. 다음 수식은 t번째 정점에서의 브러쉬 함수에 대한 수식이다.

$$q(t) = p(t) + v_x(t) \cdot p'(t) + v_y(t) \cdot n(t)$$

여기서, $q(t)$는 t에서의 브러쉬 스트로크 지점, $p(t)$는 원래의 정점의 좌표위치, $v_x(t)$와 $v_y(t)$는 각각 브러쉬 함수의 x, y 좌표값, $p'(t)$, $n(t)$는 에지의 탄젠트 값과 법선 벡터이다. 표 1은 본 논문에서 사용된 브러쉬 함수의 종류와 기능을 정리한 것이다.

브러쉬 함수	기능
Charcoal($v_c(t)$)	높은 주파수를 갖는 톱니모양 생성
Lazy($v_l(t)$)	낮은 주파수의 포물선 형태를 생성
Hand($v_h(t)$)	법선과 탄젠트에 적용되는 적은 노이즈
Rough($v_r(t)$)	법선에만 적용되는 높은 노이즈

표 1. 브러쉬 함수와 기능

표에서 설명되어진 각 브러쉬 함수는 아래와 같은 수식으로 정의되어 진다.

$$v_c(t) = t/C * dist_z$$

$$v_l(t) = t/L * dist_z$$

$$v_h(t) = (r * dist_z)/\alpha$$

$$v_r(t) = (r * dist_z)/\beta$$

여기서, C, L은 상수, α, β는 임계값이며, $r = rand()/MAX_{rand}$이고 $dist_z = Z_{max} + |Z_{min}|$이다.

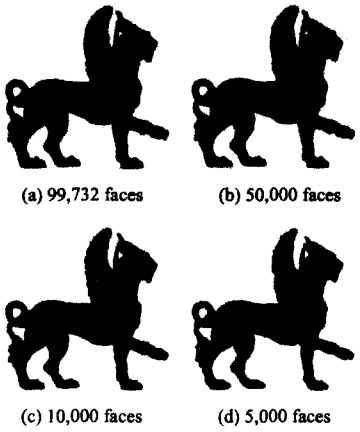
Algorithm 1 RenderWithBrush

```
for each halfEdge i do
  This = CalculateDotProduct(i →origin);
  Twin = CalculateDotProduct(i →twin);
  if (Twin <=0 and This >=0) or (Twin >=0 and This <=0) then
    ParameterizeEdge(i);
    CalculateBrushFunction(i);
    DrawBrushEdge(i);
  end if
end for
```

알고리즘 1은 브러쉬 함수를 이용해 렌더링하는 알고리즘이다. 여기서, CalculateDotProduct 함수는 현재 시점 벡터와 해당 삼각형의 법선벡터를 dot product한 결과를 구한다. 에지에 인접한 두 삼각형의 dot product한 결과가 서로 부호가 다르거나 0이면 실루엣 렌더링을 수행하게 된다. 정점의 갯수가 n개인 메쉬에 대해서 $3n$개의 에지가 존재하므로 전체 Half Edge의 갯수는 $6n$개가 된다. 따라서, Half Edge가 구성된 이후 위 알고리즘 1의 시간 복잡도는 $O(n)$이다.

4. 실험결과

본 논문의 실험은 PentiumIII-850MHz(M.M : 512MB)에서 OpenGL 라이브러리를 사용하여 C++로 구현하였다. 실험을 위한 데이터는 California Institute of Technology로부터 획득한 모델(feline, horse)과 CMU graphics lab의 모델(cow)을 사용하여 실험하였다. 그림 4는 제시한 메쉬 간략화 알고리즘을 이용하여 간략화한 결과를 보여주고 있다.



(a) 99,732 faces (b) 50,000 faces

(c) 10,000 faces (d) 5,000 faces

그림 4: 메쉬 간략화 결과(feline model)

그림 5는 점진적인 실루엣 렌더링을 위해서 사용된 horse모델이다. 그림 6는 horse 모델을 점진적 메쉬구조로 저장한 뒤 스타일 브러쉬 중 rough brush를 사용하여 점진적 실루엣 렌더링을 수행한 결과를 보여주고 있다.

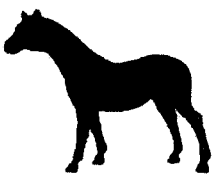


그림 5: horse model : 48,485 vertices, 96,966 faces

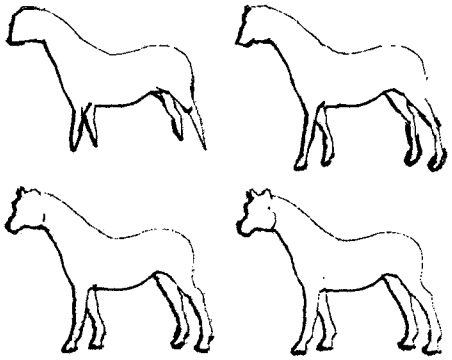


그림 6: 점진적 실루엣 에지 렌더링 결과(500 faces, 1000 faces, 5000 faces, 50000 faces)

5. 결론 및 향후과제

본 연구는 점진적 메쉬와 실루엣 에지 렌더링을 통해 사람이 임의의 객체를 점진적으로 스케치하는 효과를 보여줄 수 있는 방법을 제시하였다. 간략화 수행시 원본 메쉬에 대한 히스토리를 저장할 수 있는 점진적 메쉬(progressive mesh)구조를 구성한 뒤 기본메쉬를 점진적으로 상세하게 해 나감으로써 실루엣 렌더링을 수행할 수 있었다. 간략화로 얻어진 다단계 메쉬 모델을 비사실적 렌더링하기 위해 실루엣 에지를 찾고 파라미터화된 브러쉬 함수를 적용하여 다양한 스타일로 점진적으로 렌더링하는 알고리즘을 제안하고 구현하였다. 향후 연구 과제로는 구현 측면에서 전처리된 점진적 메쉬를 이용하여 실루엣 에지 렌더링을 수행하는 것이 아니라, 점진적 메쉬구조의 데이터 구조와 실루엣 에지 렌더링을 위한 자료구조를 통합하여 메모리 면에서 효율적인 구조로 개선해야 할 것이다.

참고문헌

[1] P. S. Heckbert and M. Garland. Surface simplification using quadric error metrics. In *SIGGRAPH '97 Proc.*, Aug. 1997.

[2] H. Hoppe. Progressive meshes. In *SIGGRAPH '96 Proc.*, pages 99–108, Aug. 1996.

[3] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, , and J. F. Hughes. Real-time nonphotorealistic rendering. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, Aug. 1997.

[4] L. P. and T. G. Fast and memory efficient polygonal simplification. In *Visualization '98 Proc.*, pages 279–286, 1998.

[5] R. Raskar and M. Cohen. Image precision silhouette edges. In *ACM Symposium on Interactive 3D Graphics '99*, April. 1999.

[6] J. Rossignac and M. van Emmerik. Hidden contours on a framebuffer. In *Workshop on Computer Graphics Hardware, Eurographics ('92 Proceedings)*, pages 31–38, Sept. 1992.

[7] M. P. Salisbury, S. E. Anderson, R. Barzel, , and D. H. Salesin. Interactive pen-and-ink illustration. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 101–108, July. 1994.