제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

스킵 스트립 렌더링을 위한 퇴화 삼각형 제거 필터[1)]

김경화*, 양성봉**
연세대학교 컴퓨터과학과

e-mail : schubelt@cs.yonsei.ac.kr

A Degenerate Triangle-Free Filter for Rendering A Skip Strip

Kyoung-Hwa Kim*, Sung-Bong Yang**
Dept of Computer Science, Yonsei University

요 약

삼각형 스트립은 삼각형 메쉬에 대한 간결한 표현을 제공하는 데이터 구조이다. 스킵 스트립은 실시간으로 삼각형 스트립을 이용하여 3차원 모델을 다해상도로 표현하는 효율적인 방법이다. 그렇지만 스킵 스트립은 모델이 점점 단순화될수록 삼각형 스트립의 과정에서 동일한 정점이 여러 번 축적된다. 이 정점들은 렌더링에 불필요한 정점의 수를 증가시키는 퇴화 삼각형을 만들게 된다. 본 논문에서는 이러한 퇴화된 삼각형을 줄이기 위해서 퇴화 삼각형 제거 필터를 제안하며, 실험결과 단순 삼각형 스트립 스캐너의 방법보다 평균 22.2% 의 정점 감소 효과를 얻을 수 있었다.

1. 서론

삼각형 메쉬는 3차원 모델을 표현하는 기본 모델이다. 각각의 삼각형은 3개의 정점을 그래픽 엔진에 개별적으로 보냄으로 렌더링될 수 있다. 일반적인 정규 메쉬(regular mesh)에서 정점(vertex)의 가수(degree)는 6이므로, 전체 메쉬에 대해 각 정점은 대략 6번씩 전송된다.

반복되는 정점의 개수를 줄이기 위해, 삼각형의 두 정점을 재사용하는 삼각형 스트립은 널리 쓰이는 기술이다[3]. 삼각형 스트립은 반복되는 정점의 수를 대략 3분의 1정도로 감소시키며, 현재 그래픽 하드웨어에서 직접 지원된다[10][2]. 임의의 삼각형 메쉬에서 최적의 삼각형 스트립을 찾는 방법은, 주어진 메쉬의 듀얼 그래프에서 해밀토니안 사이클(Hamiltonian cycle)을 찾는 문제로 말할 수 있다. 하지만 이 문제는 NP-complete으로 증명되어 최적 해를 찾기 어렵다[1]. 이로 인해 다양한 휴리스틱(heuristic) 방법들이 제안되었다[6].

3차원 모델 데이터에 대해, 메쉬의 다해상도 모델링(multiresolution modeling)이 많이 사용된다. 이러한 다해상도 모델링에서도 삼각형 스트립의 간결성을 이용하기 위해서 스킵 스트립[4]이 제안되었다. 그러나 스킵 스트립에서는 모델이 단순화 될수록 동일한 정점의 반복이 많이 발생하여, 그래픽 엔진으로 보내는 정점의 수가 증가한다.

본 논문은 스킵 스트립에서, 그래픽 엔진으로 보내는 불필요한 정점을 줄이는 효율적인 필터링 알고리즘을 제안한다. 이 필터는 스킵 스트립으로부터 얻어진 정점 시퀀스를 통해 연속적인 두 삼각형과 방향을 추출하고 두 삼각형의 관계정보를 이용하여 새로운 스트립을 구성한다. 그 결과 스킵스트립이 가지는 문제점인 불필요한 퇴화 삼각형들의 수를 감소시킨다.

2. 관련연구

본 절에서는 삼각형 스트립과 스킵 스트립을 간

단히 살펴본다.

2.1 삼각형 스트립(Triangle Strip)

T개의 삼각형을 렌더링하기 위해 삼각형을 독립적으로 렌더링하면 3T개의 정점이 그래픽 엔진으로 보내진다. 하지만 삼각형 스트립은 삼각형의 이전 두 정점을 재사용하기 때문에 삼각형 스트립은 T+2개만 필요하므로 더 효율적이다(그림1).

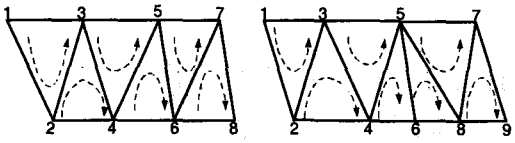


그림 1. 삼각형 스트립(a) : (12345678)
삼각형 : (123) (234) (234) (456) (567) (678)
스왑연산(b) : (1234565 789)

만약에 i 번째 삼각형을 i, (i+1), (i+2)번째 정점으로 나타내면 그림 1과 같다. 또한 스트립을 구성하기 위해 스왑연산(Swap operation)을 사용하기도 하는데, 스왑연산은 이전의 두 정점의 순서를 바꾸는 역할을 한다. 삼각형 스트립에서 스왑 연산은 이전 정점을 반복함으로 표현된다(그림 1(b)의 스트립 시퀀스 참조). 또 스왑 연산은 스트립을 길게 유지할 수 있지만 퇴화 삼각형을 만든다. 예를 들면 그림 1(b)에서 삼각형 (565)는 그래픽 엔진으로 전송되어 렌더링 되지만, 결국 선으로 나타난다.

2.2 스킵 스트립(Skip-Strip)

2.2.1 스킵 스트립구조

머지트리(merge tree)[11]는 실시간 시점 의존 렌더링(view dependent rendering)으로 모델을 표현하는데 필요한 데이터 구조이며, 스킵 스트립은 머지트리(merge tree)를 리스트로 표현한 데이터 구조이다(그림2 (a)). 스킵 스트립에서 각각의 노드는 정점 정보, 부모 포인터, 자식 포인터를 포함한다(그림 2 (c)).

노드의 자식 포인터 중 하나만 활성화(active)로 표시되는데 단순화 과정에서 그 노드가 부모 노드로 쇠퇴(collapse)되면 그 노드의 부모 포인터는 활성화로 표시된다. 그렇지 않으면 비활성화(inactive)로 표시된다.

스킵 스트립에서 삼각형 스트립은 랜더링을 위해 두 가지 데이터 타입이 필요하다. 하나는 원본 모델을 전처리 과정을 통해 얻은 삼각형 스트립이고, 나머지 하나는 현재의 레벨에 필요한 정점으로 이루어진 디스플레이 스트립(display strip)이며 렌더링 시 얻어진다(그림2 (b)). 디스플레이 스트립은 렌더링 단계에서 스킵 스트립으로부터 추출된다. 부모 포인터가 활성화이면, 비활성화 포인터를 가지는 노드가 나올 때까지 계속 활성화인 부모 포인터를 따라간다. 여기서 비활성화 포인터가 가리키는 노드는 렌더링 시에 필요하게 된다.

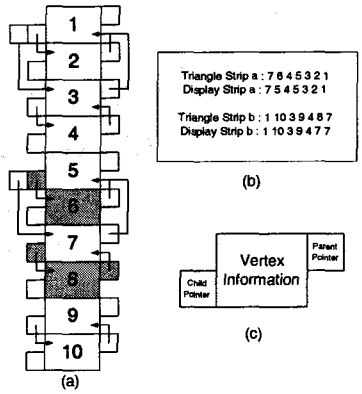


그림 2. Skip Strip 구조

모델이 단순화될수록, 디스플레이 스트립 시퀀스 안에 동일한 정점들이 여러 번 나타나게 된다. 그러한 정점들은 그려지지 않는 퇴화 삼각형을 만들게 되고, 불필요한 정점의 수를 증가시켜 그래픽 엔진으로 보내므로 효과적인 필터링 작업이 필요하다. 스킵 스트립에서는 필터링을 위해 단순 삼각형 스트립 스캐너(STSS)를 제안했다[4].

2.3 단순 삼각형 스트립 스캐너(STSS)

STSS는 정점의 시퀀스로부터 (aa)+ 와 (ab)+ 의 패턴을 감지하여 (aa) 와 (ab)로 바꾼다. 예를 들어 스트립의 정점 시퀀스가 2 2 2 2 3 5 3 5 인 경우 2 2 3 5 로 바꾼다. 모델 레벨이 높은 경우에 STSS는 퇴화 삼각형을 잘 제거하지만 레벨이 낮은 경우엔 여전히 비효율적이다.

3. 제안된 필터링 알고리즘

본 절에서는 이러한 퇴화 삼각형을 효율적으로 제거하는 필터링 알고리즘을 제안한다.

3.1 퇴화 삼각형 제거 필터(DTFF)

먼저 DTFF는 STSS와 같이 스킵 스트립으로부터 정점의 시퀀스들을 살피고, 모두 다른 세 정점과 그 방향(orientation)을 추출한다. 방향은 반시계 방

향(CCW)인 경우 0, 시계방향(CW)인 경우 1로 나타낸다. 이 때 0과 1이 교대로 반복하는 방향 패턴으로 삼각형을 뽑아낸다. 예를 들면, 정점 시퀀스가 *aabcbcbcddccdcffxxyzz* 이고 방향이 0부터 시작하면 DTFF는 삼각형 패턴으로 *abc*(1), *bcd*(0), *dcf*(0), *xyz*(1) 을 찾아낸다. 그 다음 DTFF는 두 삼각형의 연속적인 패턴을 조사하여 다음과 같이 처리한다.

3.1.1 레귤러(Regular)

삼각형 정점의 연속적인 패턴이 방향이 서로 다르고 (*abc*) 와 (*a'b'c'*) 일 때 ((*b*==*a'*) && (*c*==*b'*)) 라면, 그림 3의 (a) 와 같이 *abcc'* 의 시퀀스로 나타낼 수 있다. 위의 예에서 *abc*(1), *bcd*(0) 는 *abcd* 로 변환된다.

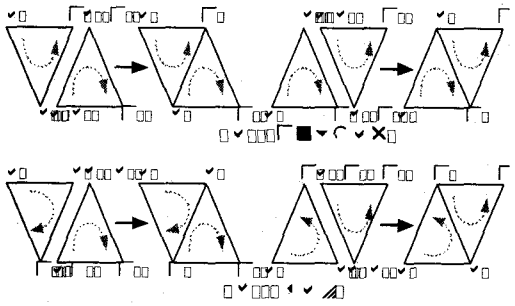


그림 3. DTFF (a) 레귤러인 경우 (b) 스왑인 경우

3.1.2 스왑(Swap)

삼각형 정점의 연속적인 패턴이 방향은 서로 같고 (*abc*) 와 (*a'b'c'*) 일 때 ((*b*==*b'*) && (*c*==*a'*)) 라면, 그림 3 (b) 와같이 시퀀스 *abcbc'* 로 삼각형 스트립을 만든다. 이 경우 한 패턴 (*a'b'c'*) 에 대해서 두 개의 정점이 필요하고 퇴화 삼각형 (*bcb*)가 스왑으로 인해 생성된다. 위의 예에서 *bcd*(0), *dcf*(0) 는 *bcdcdf* 로 변환된다.

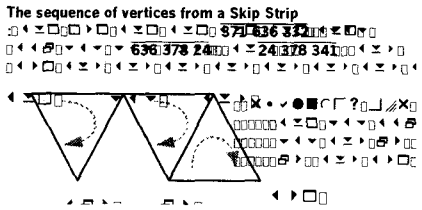


그림 4. 스킵 스트립의 정점 시퀀스
(실험에서 사용한 LOD 10% eight 모델에서 추출한 데이터)

3.1.3 리셋(Reset)

만약 연속적인 두 패턴이 스왑과 레귤러의 두 경우가 아니라면, 새로운 삼각형 스트립을 구성하게 된다. 이것은 하나의 스트립 (*abc*)이 끝나고 새로운 스트립 (*a'b'c'*)이 시작되는 것을 말한다. 위의 예에서 *dcf*(0), *xyz*(1) 가 이에 해당된다.

그림 4는 실험에서 사용한 eight 모델의 스킵 스트립으로부터 정점의 시퀀스를 보여준다. 이 시퀀스에서 STSS는 371 141 371 371 636 332 636 636 378 24 378 341 378 378 712 712 712를 얻고, DTFF는 636 337 636 334 (-1) 378 636 378 24 378 341를 얻는다. 여기서 (-1) 은 삼각형 스트립이 리셋 되어 다시 시작됨을 나타내고 정점 데이터와는 무관하다. 그래픽 엔진으로 보내지는 정점의 개수는 각각 STSS가 17, DTFF가 10 개가 된다.

4. 실험결과

표 1 은 실험에서 사용한 3차원 모델의 삼각형, 정점, 스트립 안의 정점, 스트립과 스왑연산의 개수를 나타낸다. 각각의 3차원 모델에 대하여 STRIPE 2.0[5]을 이용하여 모델을 스트립 데이터로 변환했다. 단순화는 반쇠퇴 연산(half edge collapse)을 이용했고 에러 메트릭은 QEM(Quadratic Error Metric)을 이용했다[7].

단순화의 연산으로 반쇠퇴 연산을 사용하는 이유는 이전 레벨의 정점에 대한 정보를 재사용할 수 있기 때문이다[8]. 또한 속도가 빠르고 정확하며 구현하기 쉬운 장점을 가지고 있다[9]. 그림 6은 실험에서 사용한 다양한 모델들을 보여준다.

Models	Number of Triangles	Number of Vertices	Number of Strips	Number of Swap
Bishop	496	250	1	72
Eight	1536	766	24	64
Femur	7798	3897	237	1982
Triceratops	5660	2832	144	1424

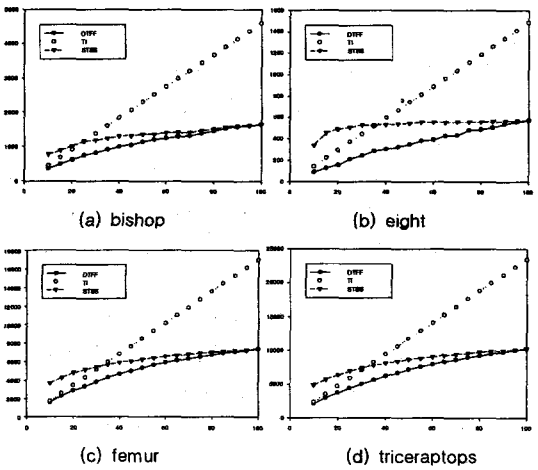
표 1. 3차원 데이터모델

실험은 삼각형을 개별적으로 렌더링 하는 TI, DTFF, STSS의 세 가지 방법으로 비교했고, 각각의 3차원 모델에 대해 다양한 레벨별로 전송되는 정점의 비율을 계산했다(그림 5). 표 2에서는 모델 eight 의 각 레벨별 정점 감소비율을 나타낸다. 다른 모델들도 표 2와 비슷한 수치로 감소한다(그림 5).

실험결과에서 DTFF는 STSS보다 상당히 많은 정점의 수가 감소했음을 보여주는데, 특히 낮은 레벨로 모델이 변해갈 수록 성능의 차이가 커진다. 그림 5와 표 2에서 알 수 있듯이, STSS는 레벨 30% 미만에서 오히려 TI보다 많은 수의 정점들을 그래픽 엔진으로 보내므로 삼각형 스트립의 장점을 이용할 수 없다.

LODs for original model	eight		
	DTFF vs TI	STSS vs TI	DTFF vs STSS
10%	18.4 %	-72.4 %	52.7 %
20%	32.0 %	-11.1 %	38.8 %
30%	39.9 %	14.1 %	30.0 %
40%	45.9 %	29.1 %	23.7 %
50%	50.3 %	40.9 %	16.0 %
60%	54.5 %	48.8 %	11.0 %
70%	58.6 %	55.7 %	6.7 %
80%	60.3 %	58.6 %	4.1 %
90%	61.7 %	61.2 %	1.5 %
100%	64.2 %	64.2 %	0.0 %
Average	49.0 %	31.3 %	17.8 %

표 2. 렌더링에 필요한 정점 감소비율 (eight 모델)



(a) bishop (b) eight

(c) femur (d) triceraptops

그림 5. 그래픽 엔진으로 보내지는 정점 수.
(X 축 : LOD 레벨, Y 축 : 정점 수)

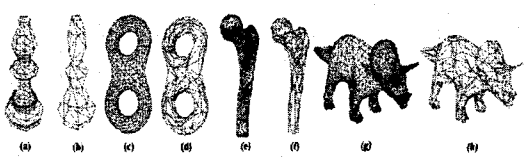


그림 6. 실험에 사용된 모델

a) original bishop b) 30% LOD bishop c) original eight
d) 40% LOD eight e) original femur f) 10% LOD femur
g) original triceratops h) 20% LOD triceratops

5. 결론

본 논문에서는, 스킵 스트립으로부터 삼각형 스트립을 최적화하기 위한 필터링 알고리즘을 제안했다. STSS의 경우 모델이 단순화 될수록 오히려 삼각형을 개별적으로 보내는 방법보다 비효율적이었다. 하지만 본 논문에서 제안한 퇴화 삼각형 제거 필터는 모델이 단순화 될수록 STSS보다 더 효율적이고, 삼각형을 독립적으로 렌더링 하는 방법보다도 효율적이다. 실험결과에서 알 수 있듯이, DTFF는 모델의 레벨 변화에 상관없이 삼각형 스트립의 장점을 계속 이용할 수 있는 필터링 알고리즘이다.

참고문헌

[1] Arkin E, Held M, Mitchell J, Skiena S "Hamiltonian triangulations for fast rendering" Lec ture Notes in Computer Science, Vol. 855. Springe r-Verlag, Second Annual European Symposium on Algorithms, pp.36-47, 1994.

[2] Chow M "Optimized geomerty compression for real-time rendering" In IEEE Visualization 1997, pp.403-410, ACM/SIGGRAPH Press, October 1997.

[3] Deering M "Geometry Compression" SIGGRA PH, pp.13-20, 1995.

[4] Jihad El-Sana, Francine Evans, Aravind Kalai ah "Effciently Computing and Updating Triangle Strips for Real-Time Rendering" Computer Aided Design vol.32(13), pp.753-772, 2000.

[5] Evans F, Azanli E, Skiena S, Varshney A "S tripe ver 2.0" http://www.cs.sunysb.edu/stripe

[6] Evans F, Skiena S, Varshney A "Optimizing Triangle Strips for Fast Rendering" IEEE Visuali zation, pp.319-236, 1996.

[7] Garland M, Heckbert P "Surface Simplification Using Quadratic Error Metrics" Proceedings of SIGGRAPH, pp.209-216, 1997.

[8] Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W "Mesh Optimization" SIGGRAPH, pp.19-26, 1993.

[9] Luebke D, Reddy M, Cohen J D, Varshney A, Watson B, Huebner R "Level of Detail for 3D Graphics" Morgan Kaufmann, 2003.

[10] Neider J, Davis T, Woo M "OpenGL Program ming Guide"Addison-Wesley, Reading, MA, 1993.

[11] J. Xia, J. El-Sana, and A. Varshney. "Adap- tive real-time level-of-detail-based rendering for polygonal models" IEEE Transactions on Visualiz- ation and Computer Graphics, 3, No.2, pp.171-183 , 1997.