한국 CAD/CAM학회 논문집
제 10권 제 5호 2005년 10월 pp. 339-348

학술논문

군집행동개체의 실시간 애니메이션을 위한 단계별 상세화 표현

조성현*, 채영호**

LOD Representation for the Realtime Simulation of Flocking Objects

Cho, S.H.* and Chai, Y.H.**

ABSTRACT

In this paper, LOD (Level Of Detail) for flocking, which is the real-time simulation on the movement of some groups such as fighting soldiers, moving fishes and flying birds, is presented. And a flocking LOD algorithm that uses factors such as the speed of fish, direction, and shape is proposed. Model data is modified for LOD in advance, so as to reduce strange edge collapse and unwanted holes. The errors of model data were identified by transforming polygonal model into octree-based cubes and revised before rendering. Experimental results show that the proposed algorithm considering flocking characteristics shows fast frame rates as compared with the conventional continuous LOD algorithm.

Key words : Level Of Detail, Flocking

1. 서 론

컴퓨터가 그래픽 파이프라인을 통해 실시간으로 처리할 수 있는 다각형의 수는 한정되어 있다. 하나의 다각형이 모니터의 픽셀로 나타나기까지는 복잡한 그래픽 파이프라인을 거쳐야 하기 때문에 과거의 컴퓨터로는 불과 수 천 개의 다각형만을 실시간으로 나타낼 수 있었다. 그래서 한정된 컴퓨터의 그래픽 처리 능력으로 보다 많은 것을 표현하기 위해서 단계별 상세화(LOD : Level Of Detail) 알고리즘이 개발되었고 초기에는 멀리 있는 물체를 스프라이트와 빌보드를 이용해서 단순한 2D이미지로 표현하는 것이 단계별 상세화의 기술이었다.

현재의 컴퓨터는 수 십 만개 이상의 다각형을 실시간으로 처리할 수 있어 상세화의 필요성이 적어졌을 수도 있지만 그만큼 사용자의 눈도 높아지고 가상으로 표현할 수 있는 환경은 무한하기 때문에 단계별 상세화는 현재까지 실시간 렌더링에서 중요한 연구 분야로 간주되고 있으며 초기의 2D 빌보드와 스프라이트에서 현재는 사용자가 포핑 효과(Popping Effect)를 느끼기 힘든 연속 단계별 상세화(Continuous LOD)로까지 발전했다. 연속 단계별 상세화는 주로 1인칭 또는 3인칭 시점의 액션 게임에서 사용되고 있다. 현재 액션게임은 대부분 주인공과 소수의 적이 싸우는 소규모 전투 방식이었는데 하드웨어의 성능이 향상되면서 많은 수의 집단들이 싸우는 대규모 전투 방식으로 바뀌어 가고 있는 추세이다. 단계별 상세화 측면에서도 이런 대규모의 집단들을 실시간으로 렌더링 할 수 있는 알고리즘의 개발이 필요한 상태이다.

2. 연속 상세화 알고리즘

2.1 점진적 메쉬

점진적 메쉬(Progressive Meshes)[1]는 1996년 시그라프(SIGGRAPH)에 발표된 최초의 연속 단계별 상세화 알고리즘 이다. 이전의 상세화 방법은 2D 빌보드와 스프라이트 또는, 미리 만들어 놓은 여러 수준의 3D 모델 데이터를 거리에 따라 불러 놓이는 방식이었다. 이전 방식들은 거리에 따라 모델을 바꿔가며 렌더링 함으로서 모델이 바뀌는 시점에서 포핑효과가 발생했다. 점진적 메쉬는 하나의 모델의 세부 수준을 실

*중앙대학교 첨단영상대학원
**교신저자, 종신회원, 중앙대학교 첨단영상대학원 부교수
- 논문투고일: 2004. 12. 15
- 심사완료일: 2005. 05. 19

시간으로 변화시켜 주는 방식으로 포핑문제를 해결하게 되었다.

Fig. 1은 점진적 메쉬의 모서리 제거방식을 나타낸 것으로 U가 V로 합쳐 지면서 U와 V사이에 있는 삼각형 $T1$, $T2$가 없어지게 되고 역으로 V와 합쳐져 있던 U가 원래의 위치로 돌아 오면서 $T1$, $T2$가 다시 나타나게 된다. 이와 같은 방식을 모서리 제거/꼭지점 분리(Edge collapse/Vertex split)라 한다. 그리고 점진적 메쉬에서는 제거할 모서리를 찾은 방식은 메쉬 최적화(Mesh Optimization)[4]의 에너지 방정식을 이용했다.

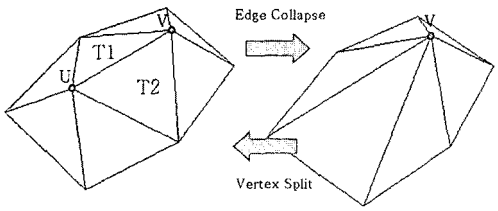

Fig. 1. Edge collapse/vertex split process.

에너지 방정식은 다각형 모델의 기하학적 모양을 하나의 스칼라 값으로 나타낸 식이다. 물론 이 값이 같은 모델은 모두 같은 모양을 하고 있지는 않다. 고차원의 다각형 모델을 1차원으로 줄였기 때문에 역(Inverse)이 성립하지는 않지만 이 값의 변화량으로 하나의 모델이 기하학적으로 많이 변했나 적게 변했나 하는 이진(Binary)적인 판단을 할 수 있게 된다.

Fig. 1에서 모서리 제거 전의 모델이 가지고 있는 에너지를 $E1$이라 하고 모서리 제거후의 에너지를 $E2$라 하면 모델의 에너지 변화 ΔE는 $E2 - E1$이 된다. 이 값이 작은 쪽이 변화 후에 모델의 기하학적 손실이 적은 것이다. 이런 방식으로 가장 작은 ΔE를 찾아내 모서리를 제거하는 방식이 점진적 메쉬를 통한 단계별 상세화이다. 점진식 메쉬의 단계별 상세화 과정을 좀더 효과적인 자료구조로 만들어 성능을 향상시킬 수 있다[5].

2.2 쿼드릭 에러 메트릭(Quadric Error Metrics)

쿼드릭 에러 메트릭[6]은 점진적 메쉬와 같은 모서리 제거/꼭지점 분리방식을 사용한다. 하지만, 모델의 여러 모서리들 중 가장 덜 중요한 모서리를 찾은 방법으로 2차 오류거리를 이용한 것이다. 이 알고리즘은 점진적 메쉬에 비해서 쉽게 구현할 수 있고 모양도 좋아 실시간 렌더링에 매우 적합하다.

Fig. 2에서 보면 점 U 주변에는 $\{a, b, c, d, e, f\}$ 삼각형이 둘러싸고 있다. 점 U는 각 삼각형들의 평면의 방정식을 모두 만족한다. 그러나 점 U가 V 방향으로 이동하면서 오류가 나기 시작한다. 이 오류 값의 제곱이 가장 작은 모서리가 없애기 가장 좋은 모서리가 된다.

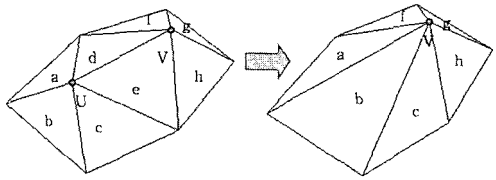

Fig. 2. Edge collapse by Quadric Error Metrics.

U에 이웃 하는 다각형을 i라 하고 법선 벡터를 N_i라면, 평면의 방정식 상수는 $D_i = -N_i \cdot U$이고 $P_i = (N_i, D_i)$로 표시할 수 있다. 대응점을 $V = (V, 1)$로 표시하면 모서리의 비용은 $E(U, V) = \Sigma(P_i^T, V)^2$ 로 나타낸다.

조금 더 오류를 줄이기 위해서 $U \rightarrow V$ 방식에서 U, V의 모서리 사이에 오류가 가장 작은 X를 찾는 변형된 방식을 사용하는 경우가 많다. 이와 같이 쿼드릭 에러 메트릭에서 U, V 사이의 X를 찾아 이용한 방식을 QEM②라 하고 U, V를 이용하는 방식을 QEM①라 하겠다. 쿼드릭 에러 메트릭을 확장하여 계산된 비용을 모델의 재질과 텍스쳐에 적용한 알고리즘[7]도 있다.

2.3 단순 다각형 감소(SPR: Simple Polygon Reduction)

단순 다각형 감소를 이용한 단계별 상세화 방식 역시 모서리제거/꼭지점 분리 방식을 사용한다. 모델의 모서리들 중 제거할 모서리를 찾은 방법은 간단한 인접 삼각형들의 법선 벡터를 이용하므로 모양은 좀 나쁘지만 알고리즘이 쉽다는 장점이 있다.

4가지의 상세화 방법 모두 모서리 제거/꼭지점 분리 방식을 사용하는 이유는 모델의 기하학적 정보 변화 없이 위상정보만 변화시킴으로 속도가 빨라서 실시간 렌더링에 적합하기 때문이다. 모델은 점의 위치변화 없이 삼각형의 위상변화로만 단계별 상세화 된다. 본 논문에서는 4가지 방법 중에서 모양이 가장 좋은 QEM②를 이용했고, 3가지 단계별 상세화 한 비교결과를 Fig. 3에서 보여준다.

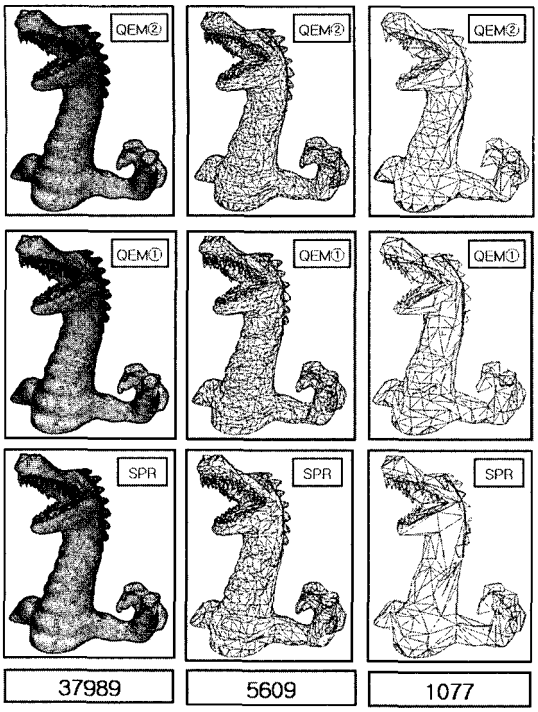

Fig. 3. Polygon reduction by different LOD algorithms.

3. 상세화 데이터의 보정

3.1 단계별 상세화될 데이터의 기하학적 오류

단계별 상세화의 실험에서 많이 사용되는 스탠포드 머니, 드래곤, 모짜르트 등 대표적인 데이터는 원단 모양이 좋게 나오고 널리 사용되므로 다른 알고리즘과 비교 분석이 쉽다. 모양이 잘 나온다는 것은 단계별 상세화 하기에 적당한 데이터들이란 뜻이다. 특징적으로 본다면 메쉬의 격자가 균일하고 표면자체가 하나의 물체이며 표면이 닫혀 있다. 일반적으로 모델러는 단계별 상세화를 고려하지 않고 3D 모델을 디자인하기 때문에 격자가 균일하다는 등의 특징을 가질 경우는 상세화 적용 시에 모양이 잘 나오지만 그렇지 않을 경우 기하학적 오류가 발생하여 단계별 상세화를 적용하지 못하는 경우가 발생한다.

일반적으로 단계별 상세화 할 때 오류가 발생하는 첫 번째 이유는 서로 다른 물체의 다각형 간의 간섭에 의해 오류가 발생하는 경우이다. Fig. 4는 모서리 제거/쪽짓점 분리 과정에서 간섭에 의한 오류가 발생하는 것을 보여준다. 모델에서 간섭이 일어나는 부분은 상반 되는 기하학적 정보가 교차하는 지역임으로 특히 눈에 잘 보이는 부분이다. Fig. 4에서 타원의 안쪽이 가리키는 부분이 간섭이 일어나는 부분인데 이 부분이 모서리 제거/쪽짓점 분리 과정을 반복하면 마치 파도가 치는 것과 같은 현상이 나타난다. 또 이 부분 때문에 단계별 상세화를 할 때마다 포핑효과가 발생하여 연속 단계별 상세화의 장점을 살릴 수 없게 된다.

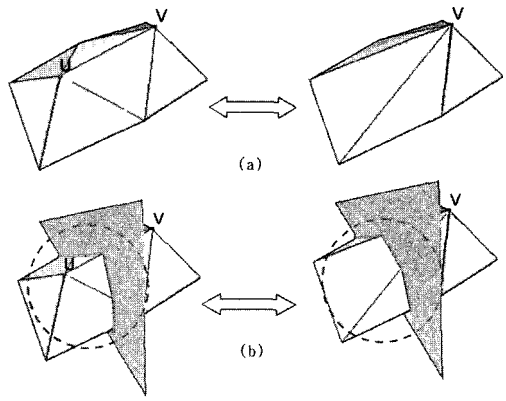

Fig. 4. (a) Normal edge collapse/vertex split (b) Interference error in LOD process.

둘째는 모델에 구멍이 발생하거나 물체가 분리되는 현상이 나타나는 경우이다. Fig. 5의 경우는 앞의 간섭부분에서 일어나는 현상과 거의 같지만 모델이 분리 될 수가 있다는 점에서 차이가 있다. 이 같은 경우가 극단적으로 발생하면 인체 모델을 단계별 상세화 했을 때 팔 다리가 떨어져 나가고 피부에 구멍이 나는 현상이 나타나게 된다. 세 번째는 열린 표면에 단계별 상세화를 적용할 경우이다. 앞에서 언급한 상세화의 대상이 되는 모서리 값은 모두 닫힌 표면을 기준으로 구한 값이다. 모서리가 열려 있는 부분이 있을 경우

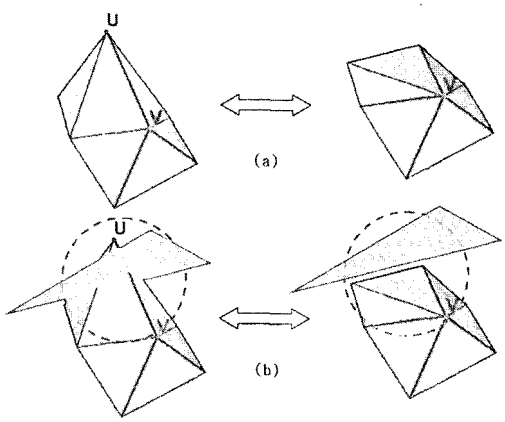

Fig. 5. (a) Normal edge collapse/vertex split (b) Hole error in LOD process.

대부분 모서리 값이 다른 곳보다 작게 나오기 때문에 이 부분에서 단계별 상세화가 진행되고 모델이 손상된다.

3.2 단계별 상세화 될 데이터의 오류 보정

본 논문에서는 군집행동 하는 물고기 모델뿐만 아니라 일반적인 데이터를 단계별 상세화에 적합하도록 보정하는 방법을 제안한다. 앞에서 설명한 첫 번째와 두 번째의 경우는 서로 다른 물체간의 연결부분 또는 중첩된 부분에서 발생하게 된다. 다각형 하나 하나의 입장에서 본다면 물체간의 연결된 부분과 중첩된 부분은 Fig. 6과 같이 세가지로 구분된다.

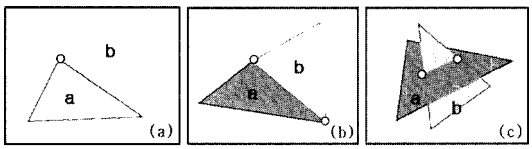

Fig. 6. Error area in LOD process (a) vertex (b) edge (c) surface.

Fig. 6을 보면 a다각형과 b다각형은 서로 다른 물체의 다각형이다. ①과 ②의 경우는 모델의 각 점의 위치 정보에 의해 알 수 있지만 ③의 경우는 쉽게 알아낼 수 없다. 또한 ①과 ②의 경우도 다각형 하나 하나의 점의 위치를 다른 물체의 것과 비교해야 됨으로 많은 양의 계산이 필요하다.

물체간의 중첩된 부분을 찾기 위하여 모델 전체를 감싸는 경계 상자를 만들고 옥트리를 이용해 공간을 작은 큐브로 쪼갠 후 이들 중에 다각형들을 포함하는 큐브 들을 찾아낸다.

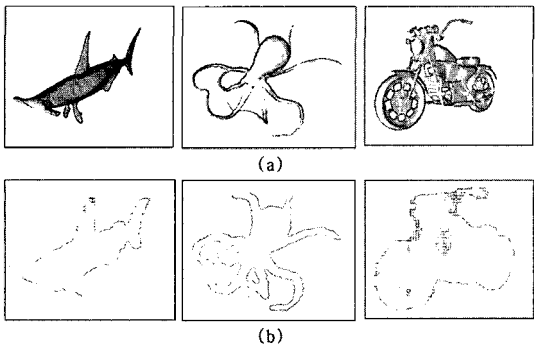

Fig. 7. (a) Polygonal model (b) Octree-based cubes.

Fig. 7은 다각형으로 되어 있는 모델을 큐브로 구성된 모델로 변환한 그림이다. 큐브의 격자가 작고 균일할수록 원래 모델에 가깝게 표현된다. 이렇게 다각형 모델을 큐브 모델로 변화하면 물체간의 중첩된 부분과 연결 부분을 쉽게 찾을 수 있다. 두 물체간의 연결 부분 또는 중첩된 부분은 모두 공통된 큐브를 포함하고 있다. Fig. 8은 Fig. 6에서 제시한 다각형간의 연결 또는 중첩의 세가지 유형에 대해서 다각형을 큐브 형태로 변형시켜 연결(중첩)부분을 찾아낸 그림이다. 다각형 모델을 통해 찾기 어려운 연결(중첩)부분을 모델을 큐브 형태로 변환시켜 쉽게 찾아 낼 수 있다.

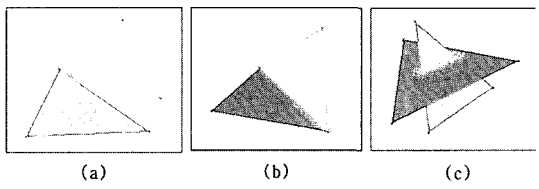

Fig. 8. Error cubes in LOD process (a) vertex (b) edge (c) surface.

다음은 연결(중첩)부분과 함께 상세화에 방해가 되는 요소인 표면의 열린 모서리를 찾는 부분이다. 열린 모서리를 찾기 전에 우선 상세화에 필요한 기하학 정보에 대해 알아 보겠다. 다각형으로 물체를 그릴 때 필요한 정보는 각 점들의 위치정보와 각 다각형을 구성하고 있는 점들의 인덱스 정보이다. 이 두 정보만 가지고 있으면 물체를 그릴 수 있는 최소한의 정보를 가지고 있는 것이다. 그러나 이 두 정보만을 가지고는 단계별 상세화를 할 수 없다. 단계별 상세화를 하기 위해서는 한 점에 이웃 하는 점들의 정보와 다각형들의 정보를 알고 있어야 한다. 이는 초기화 과정에서 미리 계산되는데, 한 점에 이웃 하는 다각형들의 정보를 이용해서 한 모서리에 포함된 다각형의 수를 알 수가 있다. 그 수가 1개인 모서리가 열린 모서리이다.

위 과정을 통해서 중첩되거나 열린 부분을 모두 찾아 낼 수 있고 중첩되거나 열린 부분은 상세화를 하기 전 상세화 노드에서 제외시킨다. Fig. 9는 그 과정을 보여준다. 점 U는 V1부터 V5까지 5개의 인접모서리를 가지고 있다. V4와 V5가 열린 혹은 중첩된 부분일 경우 V1, V2, V3모서리만을 이용하여 단계별 상세화를 하면 9-(b)와 같이 왜곡이 없고, V4, V5를 이용하면 9-(c)와 같이 기하학적 왜곡을 포함한 단계별 상세화가 진행된다.

군집행동 시뮬레이션에 쓰일 물고기 데이터를 앞에

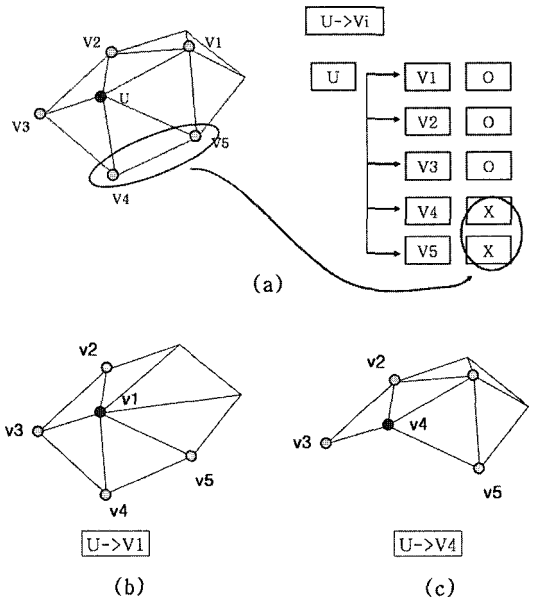

Fig. 9. LOD nodes (a) adjacent nodes (b) LOD without distortion (c) LOD with distortion.

서와 같은 방법으로 상세화에 적합하도록 보정하였다. Fig. 10은 보정 전의 데이터를 이용하여 단계별 상세화 했을 때 모양 변화를 나타낸 그림이다. 아래쪽의 단계별 상세화 된 물고기의 데이터에 오류가 발생했음을 알 수 있다. Fig. 11은 보정 과정을 거쳐서 단계별 상세화 한 그림이다. 위의 그림이 상세화 하기전의 그림이고 아래 그림이 단계별 상세화 한 후의 그림이다. Fig. 10의 보정하지 않은 데이터와 비교하면 오류가 없이 상세화가 잘 된 것을 볼 수 있다.

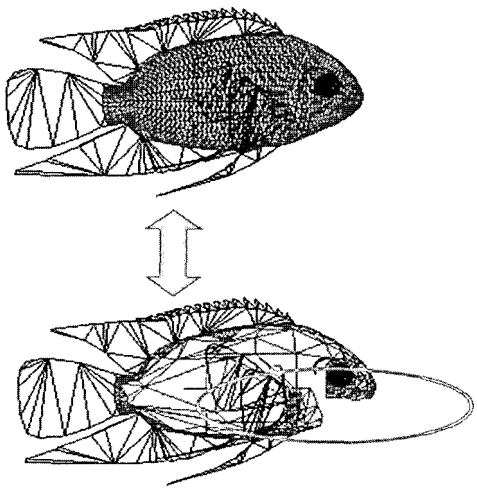

Fig. 10. LOD without the modification of the initial data.

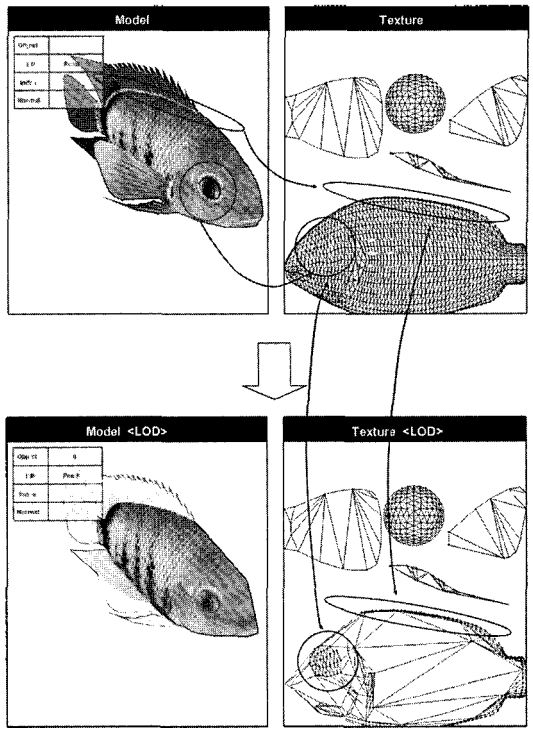

Fig. 11. LOD with the modification.

4. 단계별 상세화를 이용한 군집행동 시뮬레이션

4.1 물고기의 군집행동 시뮬레이션

군집행동(Flocking)은 1987년 시그라프에 발표된 논문[16]에서 등장한 기법이다. 레이놀즈는 새떼나 물고기떼, 또는 벌떼 등의 집단 행동을 모사 하기 위해서 다음의 3가지 규칙을 제시했다.

① 분리 - 주변 개체 들과 충돌하지 않도록 방향을 돌린다.

② 정렬 - 주변 개체 들과 같은 방향을 가리키도록 한다.

③ 응집 - 주변 개체 들의 평균 위치쪽으로 방향을 돌린다.

3가지 규칙 중에 분리 규칙은 개체가 주변의 다른 개체들과 적당한 거리를 유지하도록 하는 것이다. 응집 규칙만 있다면 개체들은 서로 부딪히게 되는데, 그것을 방지하는 것이 분리 규칙이다. 무리의 각 개체는 주변 개체와의 거리를 판단하고 적당한 거리를 유지할 수 있도록 방향을 돌리는 방식으로 이 규칙을 구현한다. 정렬 규칙은 개체가 주변의 다른 개체들과 동일

한 방향을 유지 하도록 하는 것이다. 이 규칙에 의해 개체들이 모인 무리가 마치 하나의 개체처럼 움직이게 된다. 응집 규칙은 개체 들을 하나의 무리로써 모이게 하는 역할을 한다.

집단행동은 모사할 대상에 따라 그 내용이 매우 다르게 나타난다. 새와 같은 경우는 집단행동을 하는 개체의 수가 적고 응집 형태도 덜 촘촘한데 반해서 물고기의 경우 개체의 수가 매우 많고 개체간의 간격이 촘촘해 마치 하나의 개체가 움직이는 것처럼 보여진다.

본 논문에서는 집단행동 대상인 물고기의 움직임을 다음과 같은 방법으로 간단하게 모사했다. 일단 움직임의 주체가 되는 대장 물고기가 있다. 대장물고기 만이 이동하는 목적지를 알고 있어 목적지를 향해 이동한다. 나머지 물고기들은 대장이 지시해준 위치를 가지고 있는데 이 위치를 유지하려는 성질을 가지게 된다. 대장물고기는 목적지를 향해 가고, 대장물고기 주위에는 가상의 타원이 둘러 쌓여 있게 된다. 대장물고기는 나머지 물고기의 개체 수에 따라 적절하게 자신의 가상의 타원 안에 물고기들의 목적지를 부여하게 된다. 대장물고기로부터 목적지를 부여 받은 물고기들은 대장물고기 좌표계를 기준으로 그 자리를 유지하려는 성질을 가지게 되서 자동으로 정렬과 응집의 두 가지 성질을 가지게 된다. Fig. 12에서 대장물고기가 O 에서 O'로 이동하면서 나머지 물고기의 목표점이 $P1 \rightarrow P1'$ 그리고 $P2 \rightarrow P2'$로 바뀐다. 이 과정을 연속적으로 하면 간단한 물고기의 집단 행동이 나오게 된다.

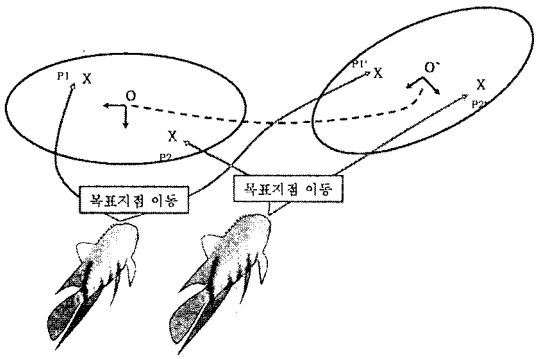

Fig. 12. Movement of flocking fishes.

4.2 집단행동에 적합한 단계별 상세화 방법의 설계

상세화는 주로 카메라와 물체 사이의 거리를 가지고 계산한다. 모델의 최고의 해상도 부분을 1.0이라 하고 가장 해상도가 낮은 부분을 0.0이라 하고 카메라와의 거리를 d라 하면 함수 $F(d) = V$(V는 상세화 레벨로 0.0에서 1.0사이의 값이다.)로 나타낼 수 있다.

이 형태는 가장 일반적인 것이고 시점종속(View Dependent) 단계별 상세화는 물체의 방향에 따른 요소가 고려되고, 움직이는 물체의 경우 속도 단계별 상세화는 카메라와의 상대속도를 이용한다. 또한 편심(Eccentricity) 단계별 상세화는 물체가 카메라 중심에 어느 정도 위치하는가를 나타내는 요소가 사용된다. 지금까지 언급한 요소들은 어떤 개체 하나와 카메라 사이에 존재하는 요소들이다.

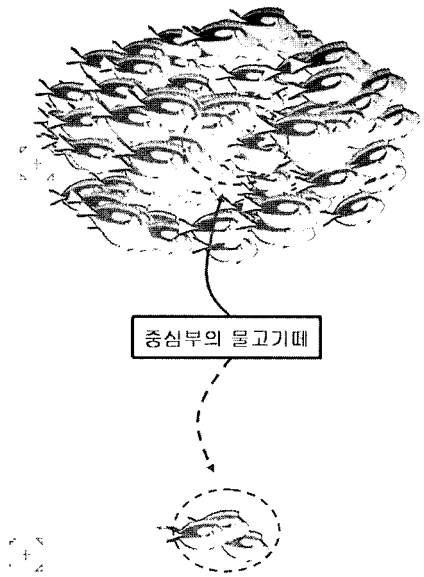

Fig. 13. Elliptic layer of flocking fishes.

이들 요소 이외에 군집행동을 하는 물고기떼에서만 나타낼 수 있는 상세화 요소는 물고기 계층이다. Fig. 13에서와 같이 물고기떼가 물고기떼의 중심 쪽에 있으면 다른 물고기에 가려 잘 안보일 것이고 타원의 바깥 쪽에 있으면 잘 보일 것이다. 물고기떼의 가장 외각에 있으면서 카메라 좌표를 기준으로 Z값이 가장 작은 곳에 있는 물고기떼가 가장 잘 보일 것이다.

본 논문에서는 군집행동 물고기떼와 카메라 사이의 거리를 단계별 상세화의 요소로 사용하였다. 각 물고기들의 카메라와의 거리는 대장물고기와 카메라 사이의 거리를 구한 다음 방향도함수를 이용해서 카메라 거리의 증가 방향 벡터를 구한 후 이 벡터에 대장물고

기 좌표계로 나머지 물고기들의 위치를 내적 시켜 나머지 물고기의 거리를 계산하였다.

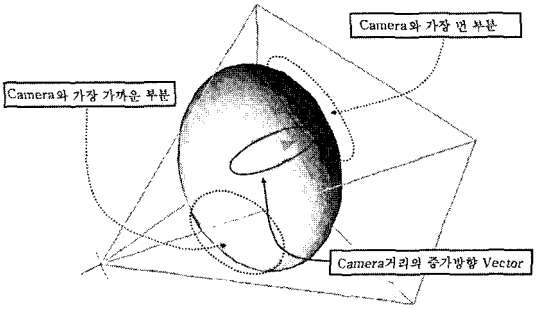

Fig. 14. Distance between the camera and fishes.

Fig. 14를 보면 가장 밝은 곳이 카메라에 가장 가까운 부분이고 가장 어두운 부분이 카메라에서 멀리 떨어진 부분이다. 카메라 거리 값이 증가하는 방향의 방향도함수는 다음과 같이 구할 수 있다. 대장물고기의 물체행렬을 F_0라 하고 카메라의 물체행렬을 C 그리고 각 물고기들의 물체좌표를 $F_0 - F_n$이라 하면, 카메라의 방향은 $\vec{V_c} = \langle 0, 0, -1, 0 \rangle$ 이고 물고기 위치에서 카메라거리의 증가방향은 $\vec{V_f} = (F_0)^{-1} C(\vec{V_c})^T$ 이다.

또한, 대장물고기와 카메라와의 거리를 d라 하면, i 번째 물고기의 카메라와의 거리 d_i는 $d + \vec{V_f} \cdot ((F_0)^{-1} F_i (V_C)^T)^T$ 가 된다.

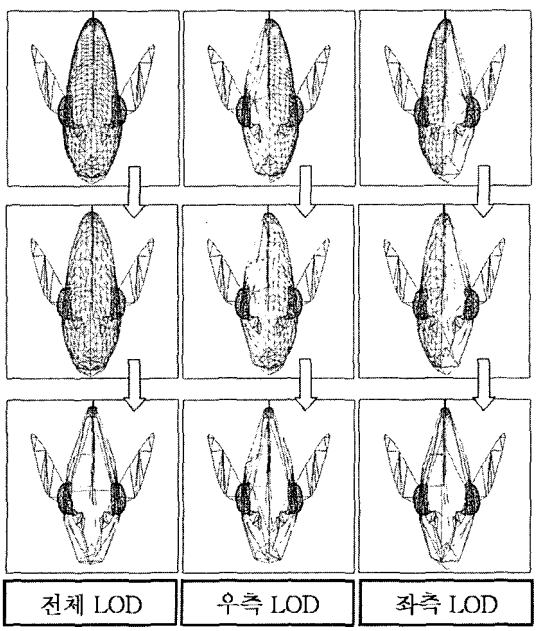

Fig. 15. LOD by the direction of fishes.

군집행동물체 단계별 상세화의 다음 요소는 방향성이다. 군집행동을 하는 물고기들은 같은 방향으로 진행하려는 성질을 가지고 있다고 보면, 각각의 방향을 하나하나 구할 필요가 없다. 대장물고기의 방향을 구한 후 이것을 모든 물고기에 적용하면 된다.

물고기는 대부분 유선형의 몸체를 가지고 있어 좌측과 우측을 나누어 상세화 시키면 좀더 좋은 결과를 얻을 수 있다. 본 논문에서는 세가지 방향에서만 상세화 노드를 만들어 사용했다.

Fig. 15는 하나의 물고기 모델에 3가지 상세화 노드를 만드는 과정을 나타낸 그림이다. 렌더링 할 때 3가지 상세화 노드 중 어느 것을 쓸 것인가는 카메라 위치와 물고기의 방향에 따라서 결정된다. 카메라의 물체행렬을 C, 물고기의 물체행렬을 F라 하고 카메라 방향과 좌측을 각각 $\vec{V_1} = ((F^{-1})C(V_C)^T)^T$ 와 $\vec{V_2} = \langle 1, 0, 0, 0 \rangle$라 하면 그 사이 각은 $\beta = \cos^{-1}(\vec{V_1} \bullet \vec{V_2}) / \|\vec{V_1}\| \times \|\vec{V_2}\|$ 이다.

여기서 $-60° < \beta < 60°$이면 좌측을 상세화 시키고 $120° < \beta < 240°$이면 우측 상세화를 그리고 나머지 구간은 전체를 상세화 시켜 사용하면 된다. Fig. 16은 좌측 또는 우측으로 상세화 시킬 영역을 나타내는 그림이다. 시점이 이 영역 안에 없으면 전체를 상세화시켜 사용한다.

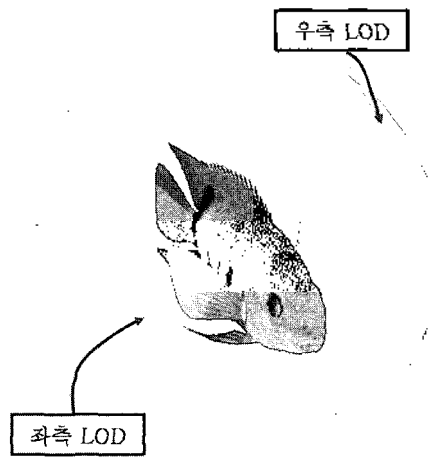

Fig. 16. Region of LOD.

4.3 성능 평가

성능을 평가할 컴퓨터의 CPU는 듀얼 Pentium 800, 램은 512M, 비디오 카드는 GeForce FX 5600을 사용하였고, 프로그램 언어는 VC++6.0와 OpenGL을 사용하였다. 그리고 상세화의 대상이 되는 물고기

는 상세화를 고려하지 않고 모델링 된 일반적인 모델을 사용하였다. 각 모델들은 앞에서 설명한 보정 과정을 거쳐 좌측 상세화, 우측 상세화, 전체 상세화의 세가지 상세화 노드를 가지게 된다. 물고기 군은 한 군이 150마리의 물고기가 모여 다닌다고 설정했다. Fig. 17이 물고기 군을 나타낸 그림이다. 각 물고기들의 다각형 수가 보통 4천 개 이상이므로 상세화를 하지 않을 경우 엄청난 수의 다각형을 렌더링 해야 한다.

비교는 군집행동 상세화, 일반 상세화, 그리고 상세화를 적용하지 않은 경우로 크게 나누었고 군집행동 상세화 요소인 계층과 방향, 상대거리를 따로 비교해서 이 요소들이 각각 렌더링에 미치는 영향을 살펴 보았다. 비교는 초당 프레임 수로 하였고 일반 상세화와 군집행동 상세화의 인자들은 동일하게 맞추었다.

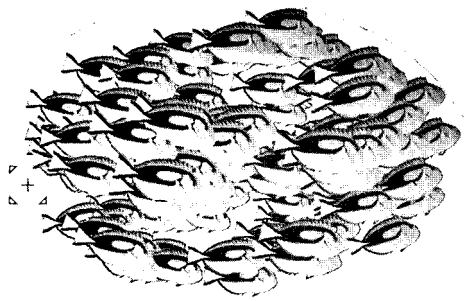

Fig. 17. Group of fishes for the performance evaluation.

비교 방법은 카메라가 지형상에서 움직이는 하나의 물고기 군을 따라 다니며 그 주위를 돌면서 초당 프레임 수의 변화를 조사하였다. Fig. 18은 상세화의 성능 평가를 하는 프로그램이다. 이 프로그램을 통해 상세화를 하지 않았을 때와 일반적인 상세화를 적용했을 때, 그리고 군집행동 특성을 이용한 상세화를 적용했을 때를 비교해 보았다. 그림의 화면은 군집행동 상세화를 적용한 화면이다. 우측 중간에 상대 속도를 나타내는 그래프가 있는데 대장 물고기의 속력을 1.0으로 봤을 때 나머지 물고기들의 속력을 비례적으로 나타낸 그래프이다. 그림을 보면 속력이 유사함을 알 수 있다. 또 그 옆에는 각도를 나타내는 그래프인데 이 값들은 대장물고기의 방향과 다른 물고기의 방향을 내적한 값이다. 이 값이 1.0이면 방향이 완전히 일치하는 것이고 코사인의 역함수에 집어 넣으면 사이 각을 알 수 있다. 그림을 보면 거의 모든 값이 1.0 가까이에 있음을 알 수 있다.

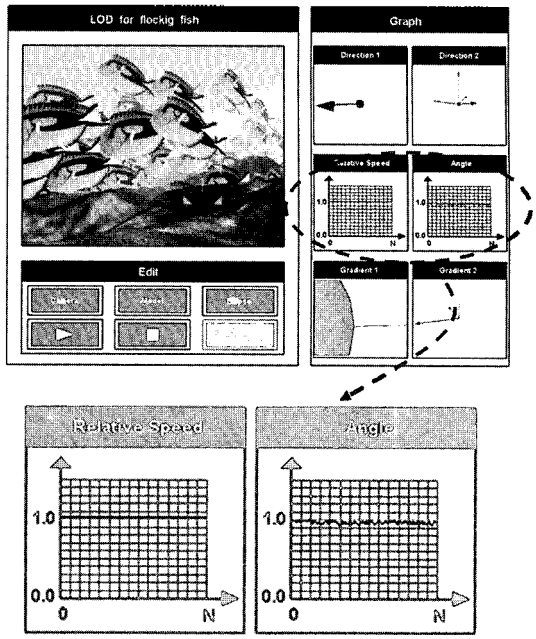

Fig. 18. Performance evaluation program.

Fig. 19는 군집행동 단계별 상세화의 모든 요소인 계층, 방향, 상대거리 모두를 적용한 경우와 여기서 방향 요소를 빼고 상세화 한 경우를 비교한 그림이다. 방향 상세화는 Fig. 15, 16과 같이 카메라의 위치에 따라 좌측/우측 상세화를 적용할 때와 전체 상세화를 적용할 때로 나눠지는데, 좌측/우측 상세화가 적용될 때에는 보이지 않는 물고기의 뒷면을 최소한의 다각형으로 렌더링 함으로 화면의 질을 유지하면서 프레임 수를 올릴 수 있다. Fig. 19의 중간 부분이 카메라가 방향 상세화에 적용을 받는 가장 긴 구간인데, 이 구간에서 프레임 수의 차이가 있음을 알 수 있다.

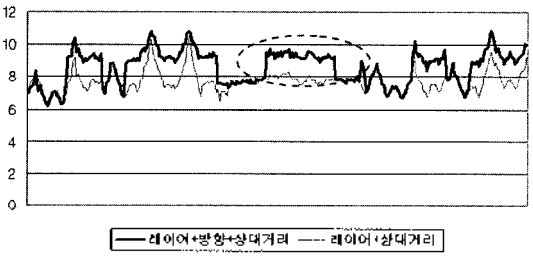

Fig. 19. Performance evaluation (1).

Fig. 20은 군집행동 단계별 상세화의 모든 요소인 계층, 방향, 상대거리 모두를 적용한 경우와 여기서 계층 요소를 빼고 상세화 한 경우를 비교한 그림이다. 계층 요소는 물고기군의 중심부에 가까이 있는 물고

기일수록 다른 물고기에 가려서 잘 보이지 않음으로 물고기의 정밀도를 낮추어 렌더링 하는 기법이다. Fig. 20을 보면 카메라가 멀리 있는 구간은 전체적인 물고기의 정밀도를 낮추고 렌더링 함으로 큰 차이가 없지만 카메라가 물고기군에 근접하면, 가까운 거리에서 정밀하게 렌더링 해야 할 중심부의 물고기들을 정밀도를 낮추어 렌더링 함으로 효과가 크게 나타남을 알 수 있다.

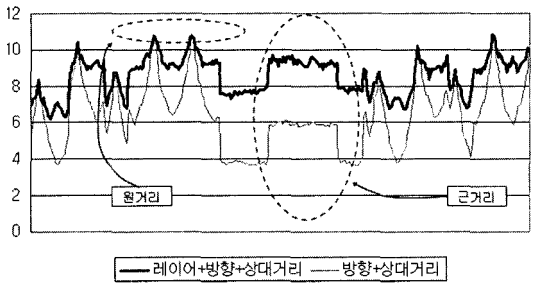

Fig. 20. Performance evaluation (2).

다음은 군집행동 단계별 상세화, 일반 단계별 상세화, 그리고 일반 상세화에 상대거리 요소를 적용한 경우를 비교해 보았다. Fig. 21은 이 3가지를 비교한 그림이다. 일반 단계별 상세화의 경우는 카메라가 물고기에 가까워 졌다 멀어 졌다 하는 동작을 반복할 때 프레임이 큰 폭으로 변화했지만 군집행동 단계별 상세화 적용 시 여러 가지 과정으로 최적화를 시켜 큰 변화 없이 비교적 높은 프레임을 유지했다. 그리고 군집행동 단계별 상세화의 상대거리 요소는 물고기들의 위치를 대장물고기와의 상대위치를 통해서 구하는 것인데 각각의 물고기들의 위치를 구하는 것에 비해 계산 양을 많이 줄일 수 있지만, 150마리의 물고기를 대상으로 해서 그 계산 양 자체가 다른 것에 비해 작기 때문에 Fig. 21처럼 큰 효과를 얻지는 못했다. 그러나 하드웨어 성능이 좋아져 물고기군의 물고기 수가 어느 이상일 경우 좋은 효과를 얻을 수 있을 것이다.

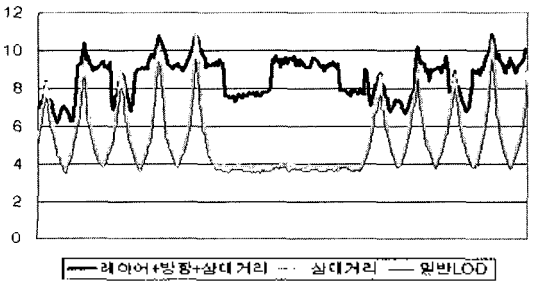

Fig. 21. Performance evaluation (3).

Fig. 22는 본 논문의 핵심이 되는 군집행동 단계별 상세화, 일반 상세화 그리고 상세화 비적용을 비교한 그림이다. 상세화를 적용하지 않을 경우는 일정 프레임을 유지 하였고, 일반 상세화의 경우는 카메라가 원거리에 있을 때 큰 효과를 얻었고 군집행동 단계별 상세화는 카메라가 원거리에 있을 때와 근거리에 있을 때 모두 일반 상세화보다 좋은 결과를 얻었다.

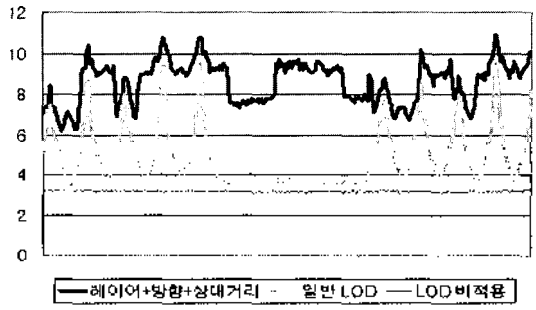

Fig. 22. Performance evaluation (4).

5. 결 론

본 연구에서는 상세화를 고려하지 않고 만들어진 일반적인 3D데이터를 상세화를 하기 전에 오류가 날 부분을 미리 파악해서 보정하는 알고리즘을 제안하고 단계별 상세화를 고려하지 않고 모델링 된 물고기 데이터들을 이 프로그램을 통해 보정을 한 후에 군집행동 단계별 상세화의 대상으로 사용하였다. 군집행동 단계별 상세화에서는 물고기들의 집단 행동의 특징인 방향성과 속력이 유사하다는 것과 군집행동의 중심에 있는 물고기 일수록 다른 물고기에 가려 잘 안 보인다는 것을 이용했다. 군집행동 하는 물고기에 가상 타원형의 계층을 만들어서 층별 상세화 함수를 적용 하였다. 카메라와 가까이 있어도 군집행동의 중심부는 해상도가 가장 낮은 모델을 사용하였다. 그리고 모든 물고기의 속력과 방향을 일일이 구하지 않고 대상 물고기의 속도와 방향으로 대처 해서 단계별 상세화 하고, 그 결과를 단계별 상세화 하지 않은 것과 물고기 각각을 단계별 상세화 한 결과와 비교하였다. 군집성을 고려한 단계별 상세화가 일반 단계별 상세화보다 좀더 좋은 결과를 얻을 수 있었다. 그러나 화면의 질이 일치한다고는 볼 수가 없기 때문에 정확한 비교라고는 말할 수 없었다. 그리고 카메라가 물고기군 안으로 들어 왔을 경우를 고려하지 않았다는 것과, 물고기가 장애물을 만났을 때를 따로 고려하지 않은 점이 아쉬운 점이다.

앞으로의 보안할 점은 장애물을 만났을 때 응집도의 변화에 따른 본 알고리즘의 효율성을 알아보고 이에 대한 개선 방향을 찾는 것이다.

참고문헌

1. Hoppe, H., "Progressive Meshes", *Proceedings of SIGGRAPH'96*, pp. 99-108, 1996.
2. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., "Mesh Optimization", *Proceedings of SIGGRAPH' 93*, pp. 19-26, 1993.
3. Hoppe, H. "Efficient implementation of progressive meshes", *Computers and Graphics*, Vol. 22, No.1, pp. 27-36, 1998.
4. Garland, M. and Heckbert, P., "Surface Simplification Using Quadric Error Metrics", *Proceedings of SIGGRAPH' 97*, pp. 209-216, 1997.
5. Garland, M. and Heckbert, P., "Fast Polygonal Approximation of Terrains and Height Fields", Technical Report CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, 1995.
6. Reynolds, C. W., "Flocking, Herds, and School: A Distributed Behavioral Model", *Proceedings of SIGGRAPH'87*, pp. 25-34, 1987.

조 성 현

2002년 중앙대학교 기계공학사
2004년 중앙대학교 첨단영상대학원 영상공학석사
관심분야: Virtual Environments

채 영 호

1989년 중앙대학교 기계공학사
1994년 SUNY at Buffalo 기계공학석사
1997년 Iowa State University 기계공학박사
1989년~1992년 (주)삼성전기 CAD/CAM 실 연구원
1998년 CASE Virtual Prototyping Lab. Consultant
1998년~1999년 중앙대학교 기계공학부 조교수
1999년~현재 중앙대학교 첨단영상대학원 부교수
관심분야: Haptics, Virtual Design, Physically based Modeling