모바일 시스템을 위한 연결 데이터 압축 알고리즘

김대영 이성열 이혜영
홍익대학교
dykim99@gmail.com, (leesy, leeh)@cs.hongik.ac.kr

A Connectivity Encoding of 3D Meshes for Mobile Systems

Daeyoung Kim Sungyeol Lee Haeyoung Lee
Hongik University

요 약

모바일 시스템은 메모리, CPU 등의 자원이 기존 데스크탑 시스템 보다 상대적으로 많이 부족하고 정수형 연산 처리만을 사용해야 하는 제약이 있다. 본 논문에서는 모바일 시스템의 특성에 적합한 새로운 모바일 3D 연결 데이터 압축 기법을 소개 하고자 한다. 연결 데이터 압축 시 옥트리 구조와 메쉬 삼각화 특성을 활용하여 연결되는 점간의 거리에 기초한 새로운 압축 방법을 개발하였다. 또한, 압축 및 해제 시에 정수형 연산만을 사용하고 동적 메모리 할당을 최소화하였다. 실제 핸드폰에서 실시간 압축 해제된 3D 메쉬 테스트 자료도 소개하여 본 알고리즘의 실용성을 높이고자 한다.

Abstract

Mobile systems have relatively limited resources such as low memory, slow CPU, or low power comparing to desktop systems. In this paper, we present a new 3D mesh connectivity coding algorithm especially optimized for mobile systems (i.e., mobile phones). By using adaptive octree data structure for vertex positions, a new distance-based connectivity coding is proposed. Our algorithm uses fixed point arithmetic and minimizes dynamic memory allocation, appropriate for mobile systems. We also demonstrate test data to show the utility of our mobile 3D mesh codec.

키워드: 무선 시스템, 3차원 메쉬, 연결 데이터 압축
Keywords: Mobile System, 3D Mesh, Connectivity Coding

1. 소 개

현재 3D 스캐닝 및 관련 기술의 급속한 발전으로 3D 데이터의 생성, 편집, 디스플레이 등이 과거보다 상대적으로 용이하게 되었다. 그 결과 유선시스템에서 3D 데이터는 게임, 애니메이션, 과학시뮬레이션, 가상현실 등 다양한 응용분야에서 새로운 미디어로 부각되고 있다. 또한 모바일 시스템에서도 음악, 이미지 및 동영상 등 1D/2D데이터를 이용한 다양한 서비스가 제공되고 있으나, 기존 서비스 성장률이 정체되고 있는 실정이어서 새로운 대안으로 3D 데이터를 활용한 서비스에 대한 관심이 증가하고 있다. 하지만, 시스템 자원이 풍부한 유선시스템을 위한 3D 데이터 처리기법을 메모리, CPU등의 자원이 상대적으로 부족하고 정수형 연산만이 가능한 현재의 모바일 시스템에 그대로 적용하기는 어렵다. 이에, 본 논문에서는 모바일 시스템의 특성에 적합한 새로운 모바일 3D 삼각 메쉬의 연결 데이터 압축 알고리즘을 소개하고자 한다.

본 알고리즘은 데스크탑 시스템에서 3D 메쉬 데이터를 압축한 후 모바일 기기로 전송하여 압축해제를 실시하고 디스플레이하는 시스템이다. 부동 소수점 연산장치가 없는 모바일 시스템에 맞게 실수 연산 대신 정수 연산을 사용하기 위해 옥트리 자료구조를 사용한다. 정수 값으로 표현되는 옥트리 셀 번호를 이용하여 기하 데이터의 위치 정보를 표현하고 이 정보를 연결 데이터 코딩을 위한 거리계산에서 사용한다. 평균적으로 원형 연결데이터 크기의 약 0.4% 크기로 압축할 수 있었으며 테스트 결과 압축률과 과정은 <표 2, 3>과 (그림 4)에 나타나 있다.

1.1 관련연구

3D메쉬 압축은 기본적으로 기하(geometry) 데이터 압축과 연결(connectivity) 데이터 압축으로 구성된다. 기하데이터는 메쉬를 구성하는 3D 공간상의 점(x, y, z)을 의미하고 연결 데이터는 각 삼각형의 면을 구성하기 위해 연결된 세 점의 인덱스 리스트를 의미한다.

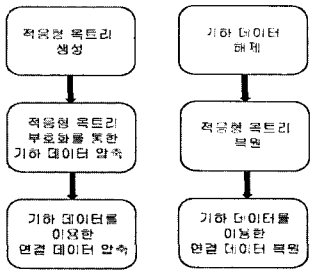
연결 데이터 압축은 크게 방향이 주어진 선분(edge)인 게이트(gate) 기반의 라벨(label) 코딩 방식과 한 점(vertex)에 연결된 선분의 수인 베일런스(valence) 코딩 방식으로 나뉜다. 초기의 대표적인 라벨 코더로는 Edgebreaker[1]가 있다. 라벨 코더는 메쉬의 면을 순회하면서 게이트 리스트(gate list)를 생성한다. 이 리스트 중 하나의 게이트를 선택하여 이 게이트와 삼각형을 이루는 나머지 한 점간의 연결 관계를 부호화하는 방식이다. 최근 발표된 라벨 방식으로 메쉬의 기하학적 특성에 기반한 적응형 메쉬 순회 방식을 도입하여 압축률을 향상시킨 Angle-Analyzer[2]가 있다.

베일런스 코더는 1998년에 발표된 Touma-Gotsman[3]의 full-valence 코더가 유명하며, 이후로 변형되고 개선된 많은 알고리즘들이 나왔다. 이 코더도 메쉬의 면을 순회하면서 하나의 포커스 점(focus vertex)을 선택하여 이 점의 full valence를 부호화한다. 현재까지 가장 높은 압축률을 보이는 코더는 FreeLence[4]이다. 이 코더는 현재 선택된 포커스에 연결된 점들 중 아직 방문하지 않은 이웃 점(free valence)들의 개수를 부호화 한다. 메쉬 순회는 Angle-Analyzer의 적응형 메쉬 순회 기법을 사용하여 포커스의 양쪽 게이트가 이루는 각도가 가장 작은 것을 선택한다. 또한 각 포커스의 각도를 컨텍스트(context)로 사용하여 압축률을 더 향상시켰다.

위에 소개된 모든 메쉬 연결 데이터 압축 방법은 자원이 풍부한 데스크탑 시스템을 위한 알고리즘이다. 따라서 압축률 향상을 위해 적응형 메쉬 순회를 위한 조건 계산이나 컨텍스트를 찾기 위한 확률계산 등을 위한 많은 실수형 연산이 필요하고 다수의 게이트 리스트 및 게이트 리스트 스택의 생성과 관리를 위한 동적메모리 할당 등을 사용하므로 메모리가 부족하고 정수형 연산만을 사용해야 하는 모바일시스템에 적용하기 어렵다. 이에, 본 논문에서는 압축률 향상을 위한 복잡한 알고리즘 보다는 처리과정을 단순화한 알고리즘을 소개하고자 한다. 본 알고리즘은 3D 메쉬 연결데이터 압축을 위한 알고리즘으로 간단한 자료구조를 사용하여 동적 메모리 할당을 최소화 하고 정수형 연산만을 사용하여 모바일 시스템 환경에 최적화 된 알고리즘이다. 또한, 실제 모바일 시스템에 본 알고리즘을 구현하고 테스트 모델을 사용한 데이터를 소개하여 본 알고리즘의 활용도를 높이고자 한다 (그림 6, 7 참조).

2. 모바일 3D 삼각메쉬 압축 알고리즘

메쉬 압축은 크게 점(vertex)의 위치를 압축하는 기하(geometry) 압축과 점 간의 연결 정보를 압축하는 연결(connectivity) 압축으로 구성된다. 모바일 시스템을 위한 3D 삼각메쉬의 기하 데이터 압축을 위해 본 논문에서는 옥트리(octree) 자료 구조를 활용한다. (그림 1)은 3D 삼각 메쉬의 압축 및 복원 전체 과정을 보여준다.



(그림 1) 삼각 메쉬의 압축 및 해제 과정

옥트리는 8개의 자식 노드(셀, cell)을 갖는 트리 데이터 구조로 3차원 공간을 표현하는 데 많이 사용된다. 본 연구에서는 적응형(Adaptive) 옥트리[5, 6] 자료구조를 사용하였다. 적응형 옥트리는 각 셀이 1개 이하의 점을 가질 때까지 모든 셀을 8개로 분할하는 방법으로 점진적 알고리즘에서 많이 사용되는 자료구조이다. 정확도를 위해 주어진 레벨(보통 12)까지 1개의 점을 포함한 단말 셀은 계속 분할한다.

옥트리를 압축하기 위하여 [6]은 확률계산을 통한 컨텍스트 모델링을 도입하고 점을 포함하는 자식 셀의 위치를 나타내는 3비트를 추가로 제공하였다. [5]는 단순히 0과 1로 적응형 옥트리를 부호화 하였고 점이 포함된 단말 셀에서 예측점과 실제점간의 차이를 압축하였다. 본 논문에서는 모바일 시스템을 위한 단순화된 기하데이터 압축 방법으로 [5]의 적응형 옥트리 부호화 방법을 적용하였고 추가로 점을 포함하는 단말의 정확도를 위해 3비트를 추가로 압축하는 방법[6]을 선택하였다. 복잡한 컨텍스트 모델링과 예측 점과 실제 점간의 차이 계산을 피하여 실수형 연산을 제거하고 정수형 연산만을 사용하여 압축함으로써 모바일 시스템에 적합하도록 하였다.

2.1 연결 데이터 압축

3D메쉬의 연결 정보는 각 면을 구성하는 공간상 점의 연결 상태를 표현한다. 본 논문에서는 메쉬의 유형을 삼각메쉬로 제한한다. 삼각메쉬의 연결 데이터는 세 점의 인덱스를 나열하여 표현한다. 각 점은 여러 면에 속하기 때문에 같은 인덱스가 중복 표현된다. 본 방법은 삼각형의 모든 면을 순회(mesh traversal)하면서 기준이 되는 삼각형의 한 변(directed edge or gate)과 다른 한 점간의 연결 상태를 부호화하여 데이터의 중복을 최소화 한다. 특히 manifold mesh의 기하학적 특성을 활용한 삼각화(Triangulation)의 원리를 활용한다. 즉, 삼각형 내의 한 점은 나머지 두 점에서 가장 가까운 거리에 있을 확률이 높다는 특징을 이용해서 연결 정보를 거리 정보로 부호화 한다. 다음 2.1.1에서 <표 1>, (그림 2), (그림 3)을 통해 연결데이터 압축에 사용될 용어를 설명하고 2.1.2에서는 압축 방법을 설명한다.

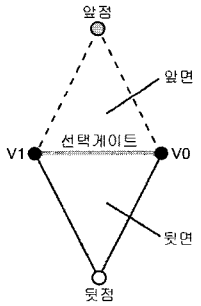
2.1.1 용어

<표 1> 용어

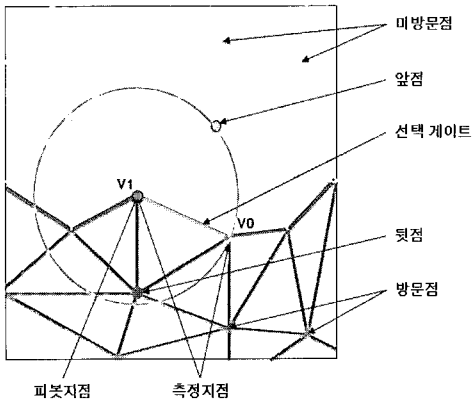
용어	설명
게이트 (gate)	삼각형의 두 점을 연결한 방향성 있는 벡터(V_0->V_1)
선택 게이트 (current gate)	현재 선택되어 처리중인 게이트
미방문점 (unvisited vertex)	아직 인코딩되지 않은 점
방문점 (visited vertex)	이미 처리된 점
앞면 (front face)	선택 게이트에 인접한 삼각형 중 현재 처리해야 할 면
뒷면 (back face)	선택 게이트에 인접한 삼각형 중 이미 처리된 면
앞점 (front vertex)	선택 게이트의 두 점과 삼각형을 이루는 앞면의 나머지 한 점
뒷점 (back vertex)	선택 게이트의 두 점과 삼각형을 이루는 뒷면의 나머지 한 점
피봇지점 (pivot point)	앞 점의 후보를 찾기 위한 중심 점 (V_1)
측정지점 (measuring point)	후보 점들과의 거리를 재는 기준 점으로 선택 게이트의 양 끝점(V_0, V_1)
후보영역 (search space)	피봇지점 V_1을 중심으로 앞점을 포함하는 구
후보집합 (candidate set)	후보영역에 포함되는 점들의 집합

2.1.2 연결 데이터 부호화

본 알고리즘은 메쉬내의 모든 면을 순회하면서 압축을 수행하는데, 게이트를 따라 면을 순회한다. 생성된 게이트들은 한 개의 리스트로 만들어 관리한다. 맨 처음 초기면(seed face)을 정하고 이 초기면으로부터 3개의 게이트를 생성하여 게이트 리스트에 추가한다.



(그림 2) 삼각 메쉬 용어



(그림 3) 연결정보 부호화를 위한 용어

게이트 리스트에서 게이트를 하나 선택한 후 이 게이트의 양 끝점(V_0, V_1)과 후보 점들과의 거리를 측정하고 그 결과를 정렬하여 앞 점의 인덱스를 부호화하는 방식이다. 각 게이트에 대해서 후보 영역을 정할 때 앞 점이 포함되도록 하기 위해 압축시에는 2-pass 알고리즘을 사용하였다. 전체 과정을 다음 수도 코드를 통해 설명한다.

- 수도 코드

- 1st Pass

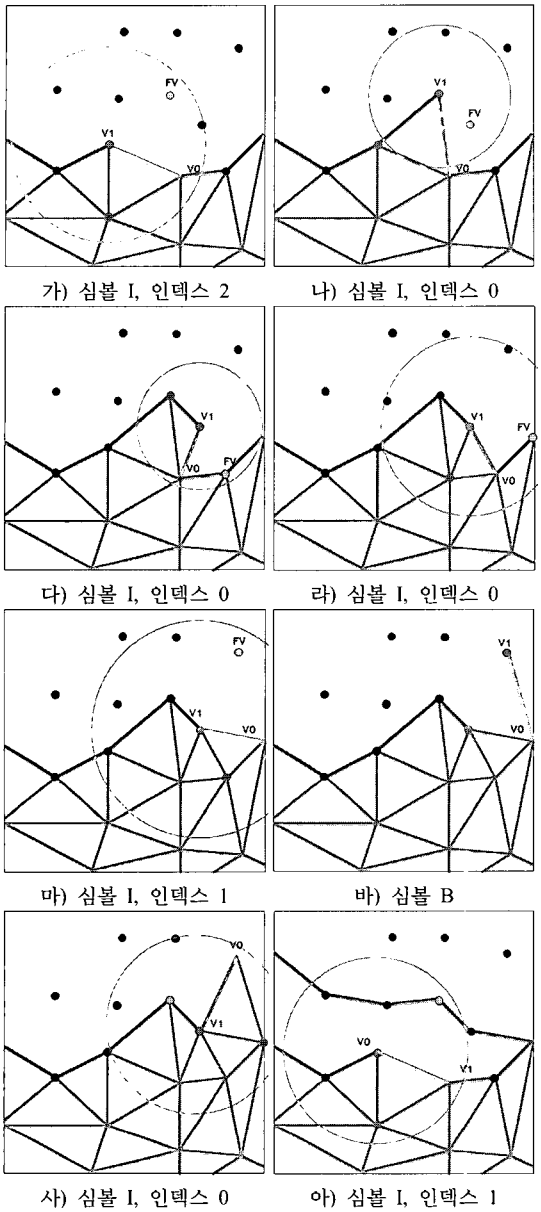
1) 초기화 단계:
   ① 임의의 초기면(seed face)을 선택한다.
   ② 초기면의 3개 게이트를 게이트 리스트에 저장한다.

2) 메쉬 순회(mesh traversal) 단계:
   (1) 더 이상 처리할 게이트가 없을 때 까지 다음 반복.
      ① 게이트 리스트에서 선택 게이트를 지정
      ② 피봇지점 V_1과 앞 점까지 거리 측정 및 저장
      ③ 선택 게이트와 앞 점이 이루는 삼각형을 만들고 연결 처리 결과에 따라 새 게이트 추가 또는 기존 게이트 삭제
   (2) (1)-②과정의 거리 중 가장 큰 값(MaxDist)를 구하여 압축과 복원 시 후보영역(피봇지점 V_1이 중심이고 MaxDist를 반지름으로 하는 구)을 구할 때 사용

- 2nd Pass

1) 초기화 단계: 1st Pass와 동일

2) 메쉬 순회(mesh traversal) 단계:
   (1) 더 이상 처리할 게이트가 없을 때 까지 다음 반복.
      ① 게이트 리스트에서 선택 게이트 지정
      ② 후보영역 내의 점들을 후보 집합에 저장
      ③ 측정지점 V_0, V_1과 후보 점들과 거리 계산 후 정렬
      ④ 정렬된 배열 내의 앞 점 인덱스를 부호화 압축
      ⑤ 연결 데이터 처리 결과에 따라 새 게이트 추가 또는 기존 게이트 삭제



(그림 4) 연결 데이터 인코딩 과정

게이트를 리스트(스택 구조)에 저장하기 때문에 메쉬를 순회하는 방법은 깊이우선방식(Depth First Search)이고 2)-⑤ 과정에서 새 게이트를 추가할 때 왼쪽 게이트를 스택에 먼저 넣고 오른쪽 게이트를 나중에 스택에 넣기 때문에 전반적인 메쉬 순회는 시계방향(Clockwise)이 된다.

2nd Pass의 2)-③과정에서 후보 집합의 한 점(P_C)과 두 측정지점(V_0, V_1)과의 거리를 구하는 식은 다음과 같다. Distance를 구하는 방법은 식(3)을 참고한다.

$$Dist(P_C, V_0, V_1) = Distance(P_C, V_0) + Distance(P_C, V_1) \quad (1)$$

부호화된 정보는 두 개의 비트 스트림(bit stream)으로 암호화된다. 첫번째 스트림은 현재 게이트가 경계선(Boundary)에 해당하는지 내부에 속하는지 구분하는 심볼 스트림(Symbol Stream)으로 각각 B(boundary)와 I(interior)로 표현된다. 두번째 스트림은 인덱스 스트림(Index Stream)이다. 이 스트림은 앞 점이 후보 집합에서 측정지점으로부터 몇 번째로 가까운지를 나타내는 정렬인덱스(Sorting Index)로 0부터 시작하는 숫자 값으로 표현한다. 경계가 없는 메쉬의 경우에는 인덱스 스트림만 보내고 경계가 있는 메쉬의 경우에는 두 스트림을 모두 보낸다. 심볼 스트림이 B인 경우는 앞점 및 앞면이 존재하지 않는 경우이므로 인덱스 값이 생략된다. 인덱스 값을 최소한 적게 나오도록 하는 것이 압축률을 높이는데 유리하다. 그러기 위해서는 피봇지점을 중심으로 후보 영역의 크기가 앞점을 포함하는 최소의 구가 되도록 하는 것이 관건이다. (그림 4)를 통해 인코딩 되는 몇 단계의 과정을 볼 수 있다.

가)는 후보 집합에 5개의 점이 포함된 경우이다. 후보 영역에 포함되는 점들 중 V_0, V_1, 뒷점은 기본적으로 제외된다. 이 5개의 점들과 측정지점(V_0, V_1)과의 거리의 합을 구한 후 정렬하면 앞 점의 인덱스는 2(3번째)가 된다. 따라서 압축정보는 심볼 스트림 I, 인덱스 스트림 2가 생성된다. 나), 다, 라) 모두 앞 점이 제일 가까우므로 인덱스가 0이 된다. 바)에서는 선택게이트가 경계선에 해당되어 앞점이 존재하지 않으므로 인덱스 없이 코드 B만 생성된다. 아)에서는 심볼 I, 인덱스 1이 생성되는 경우로 Angle-Analyzer[2]나 FreeLence[4]와 같은 알고리즘에서는 조인(Join)에 해당된다. 본 알고리즘에서는 조인을 구분하는 코드가 필요 없다. 위의 두 알고리즘들은 두 게이트 간의 각도가 제일 작은 게이트를 선택하기 때문에 각도를 계산하기 위해 게이트 리스트 내에서 게이트들의 순서를 유지해야 한다. 따라서 조인이 발생하면 게이트 리스트 스택내의 게이트 리스트들을 합병하고 나누는 과정이 필요하다. 하지만 본 알고리즘에서는 각도와는 상관없이 한 개의 게이트 리스트에 저장된 순서대로 꺼내서 처리하기 때문에 조인이 발생해도 따로 이를 구분하는 코드가 필요 없이 일관된 방법으로 간단히 처리한다.

가), 나), 마)의 경우는 삼각형을 이루는 두 게이트(왼쪽, 오른쪽)를 게이트 리스트에 추가하기만 하면 된다. 그러나 다), 라), 사)의 경우에는 두 게이트를 추가할 경우 중복되는 게이트가 발생한다. 이 때 중복되는 게이트는 추가하지 않고 오히려 게이트 리스트에서 해당 게이트를 찾아 제거한다. 경계선에 해당하는 바)의 경우에는 아무 게이트도 추가하지 않는다. 그리고 모든 경우에 처리를 끝낸 후 선택 게이트를 게이트리스트에서 제거한다.

압축해제는 위 수도 코드에서 ④번 과정만 제외하면 압축과 똑같은 과정을 거친다. 디코더의 ④번 과정은 정렬 결과에서 인코더로부터 받은 인덱스 값에 해당하는 점을 앞점으로 선택하여 삼각형을 구성함으로써 메쉬의 연결 정보를 복원한다. 본 방법을 적용한 연결 데이터 압축 결과는 <표 2, 3>에 있다.

2.2 모바일 특성을 고려한 데이터 압축 및 해제

본 알고리즘은 리소스가 적은 모바일 시스템을 위한 알고리즘이기 때문에 부동소수점 연산을 피하고 동적 메모리 할당을 최대한 사용하지 않는다.

정수형 연산 : 기하 데이터를 표현할 때 점의 위치에 해당하는 실수 값 대신 정수 0과 1로 표현되는 적응형 옥트리와 추가 위치 정보(3비트)를 사용하기 때문에 실수형 연산이 필요 없다. 연결 데이터 압축에서도 점간의 거리를 측정 할 때 실수 값을 사용하지 않고 점의 위치에 해당하는 최하위(n) 레벨 셀의 위치 값인 정수 (i, j, k)를 사용한다. 다음은 두 점 A, B 사이의 거리를 계산하는 공식이다. 실수 연산을 피하기 위하여 두 점사이의 거리를 잴 때 루트를 사용하지 않았다.

$$Cell(A) = (a_i, a_j, a_k)$$

옥트리에서 점 A 위치에 해당하는 최하위(n) 레벨 셀의 위치 값 (정수) (2)

$$Distance(A, B) = (Cell(A) - Cell(B)) = (a_i - b_i)^2 + (a_j - b_j)^2 + (a_k - b_k)^2 \quad (3)$$

정적 메모리 할당 : 후보 영역에 포함되는 후보 집합의 최대 크기, 메쉬 순회단계에서 생성되는 게이트리스트의 최대 크기 등을 인코딩 할 때 계산하여 헤더파일을 통해 디코더에게 전달한다. 따라서 모바일 시스템에서 디코딩할 때 헤더파일을 참조하여 최대크기만큼 정적 메모리를 할당해서 사용하므로 동적 할당을 최소화할 수 있다.

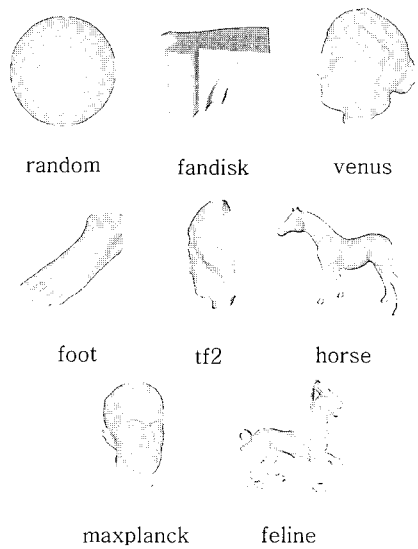
3. 압축 결과

3.1 구현 결과

본 논문에서 제시한 3D 메쉬 압축 알고리즘은 상대적으로 자원이 부족한 모바일 시스템에서 적합하도록 고안된 새로운 방법으로 압축률 향상 보다는 단순한 자료구조와 간단한 처리방법에 초점을 두었다. 이를 위하여 정수형 데이터로 표현되는 옥트리를 사용하여 3D 메쉬의 기하 데이터를 표현하고, 점을 포함하는 옥트리 셀 간의 거리를 기반으로 연결 데이터를 압축하였다.

<표 2>는 제시된 알고리즘을 사용하여 여러 모델을 압축한 결과를 보여준다. 기존의 압축 방법 중에서 FreeLencer[4] 방식과 비교했을 때 압축률은 떨어지지만 모바일 시스템에 적용하기에는 적절한 수준이다. 그리고 FreeLencer와 같은 알고리즘은 압축률 향상을 위해 많은 실수형 연산과 동적 메모리 할당이 필요하므로 메모리가 부족하고 정수형 연산만을 사용해야 하는 모바일시스템에 적용하기 어렵다는 단점이 있다. 본 논문은 연결 데이터에 초점을 두었고 새로운 기하 데이터 압축 방법을 개발 하지 않았으므로 자세한 기하데이터 압축률 비교는 무리가 있다. 참고로 <표 2> 모델들의 기하 데이터 압축률은 평균적으로 19.63(b/v)이고, 연결 데이터는 2.06(b/v)이다. 연결 데이터가 전체 압축률에서 차지하는 비율은 대략 10% 정도로 적어 보이나 이는 기하 데이터는 실수형이고 연결 데이터는 정수형이라는 데이터 특성 자체에서 기인하며 이웃하는 점, 선, 면 간의 관계를 이용한 처리를 위해서 연결데이터는 중요한 역할을 한다.

<표 3>은 원본 wrl 파일의 연결 데이터 크기와 압축된 파일의 크기를 비교한 결과이다. 평균적으로 압축된 파일 크기는 원본 파일 크기의 0.4%정도로 전송 및 가용 메모리가 적은 모바일시스템에 저장하기에 적합한 크기로 줄어든 것을 확인할 수 있다. 경계를 포함하는 egea_u, tf2, maxplanck 모델을 제외한 나머지 모델들은 심볼 스트림이 필요하지 않기 때문에 표에서 볼 수 있듯이 값이 0이다.



(그림 5) 테스트 모델

<표 2> 연결 데이터 압축률 비교

모델명	점의 개수	연결 데이터 FV (b/v)	연결 데이터 DT (b/v)
random	4,338	0.36	1.41
egea_u	5,315	0.54	0.89
fandisk	6,475	0.74	2.35
venus	8,268	1.73	3.46
foot	10,016	1.34	2.92
sphere	10,242	0.03	0.18
tf2	14,169	0.60	1.76
horse	19,851	0.96	1.94
max	25,445	1.19	3.05
feline	49,864	1.23	2.63

<표 3> 모바일 3D 압축 알고리즘 적용 결과

모델명	연결 데이터(KB)		합계 (KB)	원본 (KB)	압축률 (%)
	심볼 (code)	인덱스 (index)			
random	0	0.77	0.77	273	0.28
egea_u	0.22	0.38	0.60	328	0.18
fandisk	0	1.99	1.99	339	0.50
venus	0	3.85	3.85	526	0.73
foot	0	4.14	4.14	639	0.65
sphere	0	0.25	0.25	655	0.04
tf2	0.24	3.06	3.30	730	0.45
horse	0	5.11	5.11	1,362	0.38
max	0.1	10.29	10.39	1,762	0.59
feline	0	17.29	17.29	3,523	0.49

<표 4> 기하/연결 데이터 처리 시간 및 메모리 사용량

모델명	로딩 시간 (sec)	로딩시 메모리 사용량(KB)	로딩후 메모리 사용량(KB)
random	25.73	1,217	1,034
egea_u	35.01	1,483	1,251
fandisk	38.36	1,818	1,545
venus	63.05	2,338	1,985
foot	63.75	2,830	2,396
sphere	63.97	2,859	2,441
tf2	108.20	3,988	3,364
horse	137.80	5,600	4,747
max	182.42	7,168	6,079
feline	-	14,030	11,925

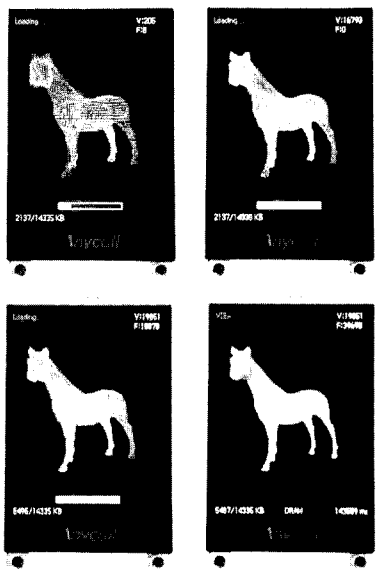
<표 4>는 핸드폰에 각 3D 메쉬 파일(기하 데이터와 연결 데이터 모두)을 로딩할 때 소용되는 시간과 로딩시/로딩후 메모리 사용량을 보여준다. feline 모델은 크기가 5M로 실험에 사용한 핸드폰 기종에서는 메모리 부족으로 로딩 시 이웃(neighboring) 정보를 계산하여 저장할 수 없어서 정확한 시간 및 메모리 측정이 불가능 하였다.

3.2 압축/해제 과정 예

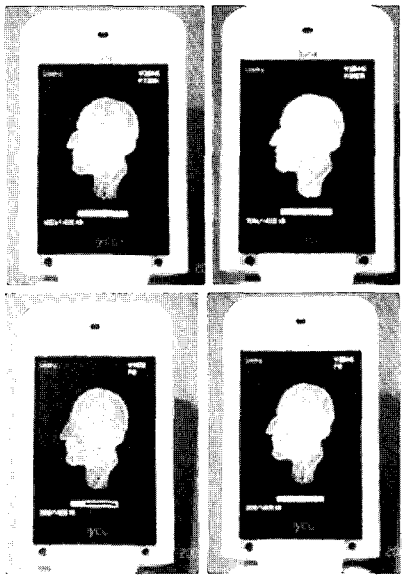
본 논문의 알고리즘을 구현하기 위해 인코더는 Visual Studio 2005 (C++)를 사용하여 개발하였다. 디코더의 모바일 플랫폼은 WIPI 1.2, API는 GIGA(Graphic Instruction Graphic Acceleration)이며 Visual C++ 6.0과 C언어를 사용하여 구현하였으며, 인코딩과 디코딩시에 arithmetic coder[7]를 사용하였다. 적응형 옥트리 레벨은 기하 데이터의 오차를 최소화하기 위해 12레벨로 설정하였다. (그림 6, 7)은 핸드폰에서의 해제 과정을 보여준다. 먼저 기하 데이터를 점진적으로 해제한 후 마지막 단계에서 연결 데이터를 해제하여 모델이 완성되는 모습이다.

4. 향후 연구 과제

향후 연구 과제로는 기하 데이터의 압축과 모바일 시스템에서 좀 더 사실적인 3D메쉬 디스플레이를 위해 3D 메쉬 데이터의 텍스쳐 데이터 압축 알고리즘을 개발할 예정이다. 또한 기존 데이터의 압축률 향상을 위한 연구도 지속할 예정이다. 특히, 컨텍스트 모델링에 대한 연구와 연결 데이터 압축을 위한 거리 측정 시 후보영역의 크기 설정을 위한 알고리즘을 개선하고자 한다.



(그림 6) 핸드폰에서 horse 모델 해제 과정



(그림 7) 핸드폰에서 maxplanck 모델 해제 과정

Acknowledgements

본 과제는 서울시산학연 협력사업 중 보유기술 사업화 지원 사업 (과제번호 11131)의 지원으로 수행되었습니다.

참고 문헌

[1] J. Rossignac: EdgeBreaker : Connectivity Compression for Triangle Meshes. IEEE Transactions on Visualization and Computer Graphics. Vol. 5(1). pp 47~61, 1999

[2] H. Lee and P. Alliez and M. Desbrun: Angle-Analyzer: A Triangle-Quad Mesh Codec. Computer Graphics Forum (Proc. Eurographics'02), Vol. 22(3), pp383~391, 2002Ö

[3] C. Touma and C. Gotsman:Triangle Mesh Compression. Graphics Interface 98 Conference Proceedings. pp 26~34, 1998

[4] F. Kalbere, K. Polthier: FreeLence : Coding with Free Valences. Computer Graphics Forum (Proc. Eurographics'05), Vol. 24(3), pp 469~478, 2005

[5] H. Lee, M. Desbrun, and P. Schröder, Progressive encoding of complex isosurfaces. In ACM SIGGRAPH, 2003.

[6] J. Peng and J. Kuo: Geometry-guided Progressive Lossless 3D Mesh Coding with Octree (OT) Decomposition. ACM Transactions on Graphics(Proc. Siggraph'05), Vol. 24(3). pp 609~616, 2005

[7] Wheeler, F., 1996. Adaptive Arithmetic Coding Source Code. http://www.cipr.rpi.edu/.wheeler/ac.