特輯 技術解說

워크스테이션을 위한 그래픽 기술과 동향

魚 吉 秀*, 黃 時 永**
三星綜合技術院 情報시스템硏究所
先任硏究員*, 責任硏究員**

Ⅰ. 소 개

반도체의 공정기술과 시스템 설계기술의 획기적 발전에 힘입어 고성능의 computer가 일반 사용자에게도 저가로 공급될 수 있게 되었다. 그러나 어디까지나 인간은 기계를 이용하여 자기가 하고자 하는 작업을 편하게 수행하고자 하므로 인간과 기계사이의 대화 -message의 상호전달- 의 형태가 수월하게 이루어 질 수 있어야 할 것이다. Man-machine interface라 일컬어지는 이 대화형태의 문제를 해결하는 일이 혹자는 미래사회의 문을 여는 열쇠라고도 이야기 하고 있다. Man-machine interface가 어떤 형태를 가져야 하는가는 앞서 언급한 데로 지극히 인간 중심, 즉 인간이 편해지는 관점에서 해결해야 한다. 인간은 외부 세계로부터 정보를 받아들이는 5가지 창구를 가지고 있다. 즉 시각, 청각, 후각, 미각, 촉각의 인간의 오감이 그것이다.

이중 단위 시간당 가장 많은 정보를 받아들이는- 시간당 전달 정보의 밀도가 가장 높은- 감각이 역시 시각이다. 따라서 기계(machine)가 인간에게 전달하는 정보의 형태는 그래프, 챠트, 단면도 등의 화상정보의 형태가 가장 효과적임을 알 수 있다.

한편 컴퓨터 그래픽스가 짧은 역사에도 불구하고 급격하게 발전하게 된 것은 자동화와 정보사회의 대두로 그 필요성이 커진데 이유가 있다. 컴퓨터 그래픽스란 3차원적 물체를 정의한 수량적 데이타를 특정 관찰 지점에서 본 2차원적 스크린으로 가시화 시키는 일로 정의된다. 초기에는 wireframe drawing에 의한 caligraphic display에서 반도체 메모리의 가격 하락과 대용량화에 힘입어 raster display 형태로 발전하였다.

Wireframe drawing을 위한 display 장비인 DVST (direct-view storage tube) 방식은 높은 resolution을 가지는 점, refresh가 필요없고 flicker-free 하다는 점 그리고 비교적 빠른 속도로 drawing 할 수 있다는 점등의 장점이 있으나 화면의 국부적인 부분만을 update 시킬 수 없고 color를 가지지 못한다는 치명적인 단점때문에 interactive system에 부적합했던 것이다.

메인 프레임이나 미니 컴퓨터에 다수의 사용자가 매달려 작업을 하던 시분할 방식이 사용자로 하여금 지루함과 좌절을 느끼게 하면서부터 1인 1컴퓨터 형태에 대한 필요성이 대두되었다. 이러한 요구에 부응하기 위하여 탄생된 것이 워크스테이션이다. 즉 host와는 network으로 연결되어 데이타 및 storage를 공유하면서 독자적인 데이타 처리능력과 빠른 전송기능을 갖춘 환경을 구축하고자 했던 것이다. 이러한 성능을 달성하기 위한 워크스테이션의 기본 사양을 5M이라 하는데 5M이란 1Mips 이상의 CPU성능 1MB 이상의 메인 메모리, 1M pixel 이상의 screen resolution, 1Mbps 이상의 communication bandwidth 그리고 mouse interface를 말한다. 현재 상용으로 나오는 워크스테이션들은 대부분 이러한 사양을 훨씬 초과하고 있다.

한편 개인용 컴퓨터 (PC)의 성능이 현저히 발전하면서 PC와 low-end 워크스테이션의 구분이 점차 어려워지고 있다. 본고에서는 워크스테이션을 위한 컴퓨터 그래픽 기술을 2D graphics, 3D graphics 그리고 graphics hardware로 나누어 설명하고자 하며 마지막으로 워크스테이션을 위한 그래픽의 추세를 살펴보고자 한다.

Ⅱ. 2D Graphics

Computer graphics라 함은 본고에서는 3D (3 dimension) graphics를 의미하고 있으나 응용 소프트웨어가 요구하는 interface 기능의 상당부분이 2D graphics로 처리될 수 있으므로 특별히 이것을 3D graphics와 분리하여 기술하고자 한다. 2D graphics의 주요 내용은 line/circle drawing, polygon filling, line clipping, polygon clipping 등으로 이루어진다. Line drawing 문제는 raster scan 방식에서는 주어진 두점 사이를 직선으로 연결하는 것으로 직선성, 균일성, 속도 등의 요구조건이 따른다. Raster scan 방식에서는 화면이 2차원적 pixel의 array로 정의되기 때문에 원칙적으로 직선을 정확히 긋는다는 것은 X축, Y축 그리고 45°각도의 선들을 제외하고는 불가능하지만 화면의 resolution이 높아지면 사람의 눈에 직선으로 보일 수가 있다. 가장 잘 알려진 방법은 DDA (digital differential analyzer)에 기반을 둔 Bresenham's algorithm이다. DDA는 순차적으로 interpolation data를 산출해 내는 테크닉인데 나누기 연산이 제거된 것이 특징이다. Circle drawing 역시 DDA에 기반을 두고 있다. Polygon filling이란 화면상에 다각형으로 정의된 특정 영역을 동일한 색깔로 채우는 것으로 OEL (ordered edge list), seed filling, edge filling의 세가지 대표적 방법이 있다. OEL 방법은 그림 1에 설명되어 있는 바와 같이 물이 차 올라가는 모습으로 이루어진다. Scanline Yc와 교차하는 다각형의 내부 span, Q1Q2, Q3Q4를 칠하게 된다. 이 방법은 shading을 병행할 수 있고 frame buffer를 read할 필요가 없는 장점이 있으나 Q1,Q2,Q3,Q4, 등의 교차점을 계산해야 하는 어려움이 있다. 이와는 대조적으로 seed filling algorithm은 frame buffer의 내용을 읽고 쓸 수 있는 하드웨어를 요구한다.

다각형의 경계선을 앞서 언급한 line drawing algorithm으로 그리면 하나의 페루우프를 정의하게 된다. 그 루우프의 내부점 하나를 seed로 지정하여 동서남북방향으로 recursive하게 화소를 방문하여 만약 방문한 화소가 이미 칠해져 있으면 return하고 그렇지 않으면 정해진 color를 write하고 recursion을 계속한다. 미리 페루우프가 형성되어 있으므로 그 루우프 내부의 화소만 방문하게 된다. 그러나 seed filling은 다각형 내부에 있음이 확실한 seed를 정해주어야 하므로 interactive한 그래픽 작업에 적합하다고 본다. 그림 2는 edge filling을 설명한다. Filling 작업은 채우고자 하는 다각형의 변을 따라서 수행되는데 각

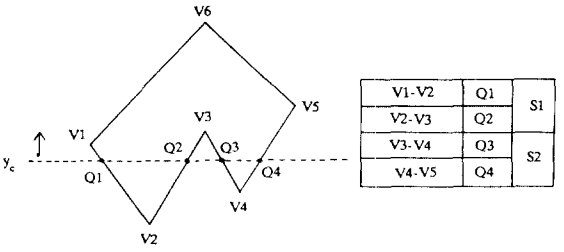


그림 1. OEL filling

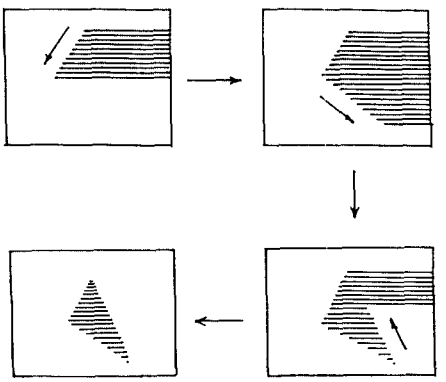


그림 2. Edge filling

변상의 한점으로부터 화면상 오른쪽 끝까지의 pixel의 bit 값을 complement 시킴으로써 filling이 이루어진다. 이 밖의 filling 방법으로는 edge flag, fence filling, scanline seed filling 등의 방법이 있는데 모두 위의 세가지 방법에 기초를 두고 있다. Line clipping 이란 직선을 특정한 window -보통 직사각형 영역-의 경계선으로 잘라서 영역 내부에 있는 부분만을 취하는 것을 말한다. 자르고자 하는 직선의 양끝점과 window의 4면과의 상관관계를 부호화 함으로써 계산의 효율을 높이는 Cohen-Suthreland algorithm이 널리 쓰이고 있는데 최근에 발표된 Nicholl-Lee-Nicholl algorithm은 출력되는 교차점에 한해서만 교차점 계산을 수행하기 때문에 가장 좋은 방법으로 평가되고 있다.

이밖에도 hardware로의 구현이 용이한 midpoint

subdivision 방법이 있는데 binary search technique에 근거를 두고 있다. 그림 3 은 polygon clipping을 제시하고 있다. 단순히 보면 line clipping과 다를 바가 없어 보이지만 polygon은 경계선에 둘러싸인 내부 영역으로 정의되기 때문에 line clipping만으로는 해결되지 않는 경우가 있다. (그림 3 에서 polygon A를 참조할 것.)

대표적 algorithm에는 Sutherland-Hudgenman algorithm과 Weiler-Atherton algorithm이 있다. 그림 4 는 Sutherland-Hudgeman algorithm을 설명한다. 이 알고리듬은 polygon의 vertex의 list를 따라가면서 각 clip boundary별로 sequential하게 clipping을 수행하는데 strategy가 그림 4 에 설명되어 있다. Region A 내에 있는 vertex들은 모두 clip boundary 밖에 존재하므로 report 되지 않으나 region B에 있는 vertex 들은 내부에 존재하므로 report 된다. 한편 내부에서 외부로, 혹은 외부에서 내부로 나가거나 들어오는 edge는 교차점 C1, C2 등으로 report된다. Wieler-Atherton algorithm은 이에 비해 좀더 일반화된 algorithm이다.

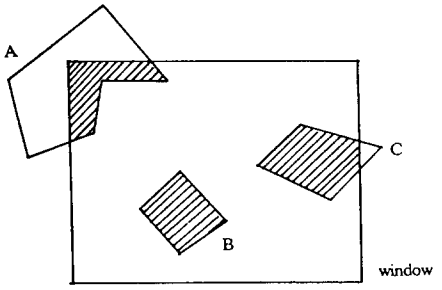
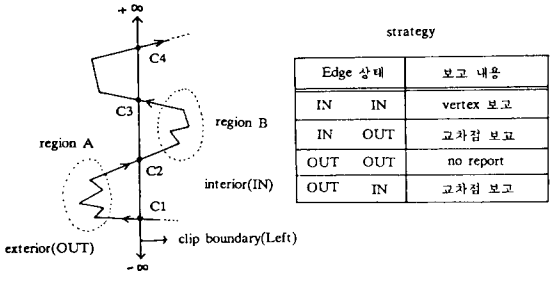


그림 3. Polygon clipping



strategy

Edge 상태		보고 내용
IN	IN	vertex 보고
IN	OUT	교차점 보고
OUT	OUT	no report
OUT	IN	교차점 보고

그림 4. Sutherland-Hodgeman polygon clipping algorithm

Ⅲ. 3D Graphics

2D graphics의 제문제는 screen상에서 발생하는데 비하여 3D graphics의 문제는 원칙적으로 object domain에서 발생한다. Object domain이란 물체가 정의된 3 차원 공간을 의미한다. 물론 개중에는 계산의 편의성을 위하여 ojbect domain을 screen상에 projection 하여 screen domain에서 3 차원 물체를 취급하는 경우도 있다. 3D graphics의 topic들에는 3D transformation, 은면제거(hidden surface removal), 조명모델(right modeling), shading, shadowing 등이 있다. 또 종합적 방법으로 ray tracing (광선 추적) algorithm이 있다.

3D transformation (변환)은 계산의 편의성을 위하여 좌표계(coordinate system)을 변화시키는 것으로 vector(좌표치)와 matrix(4*4, transformation matrix)곱셈으로 일어난다.

그림 5 는 3D transformation과 projection의 순서 및 입·출력을 정의하고 있다. Polyhedral(단면체) model에서 물체는 vertex list와 polygon list로 정의된다. 즉 vertex list는 각 꼭지점 위치를 정의해 주며 polygon list는 정의된 vertex list가 어떻게 polygon을 구성하는가 하는 topological relationship 을 정의한다.

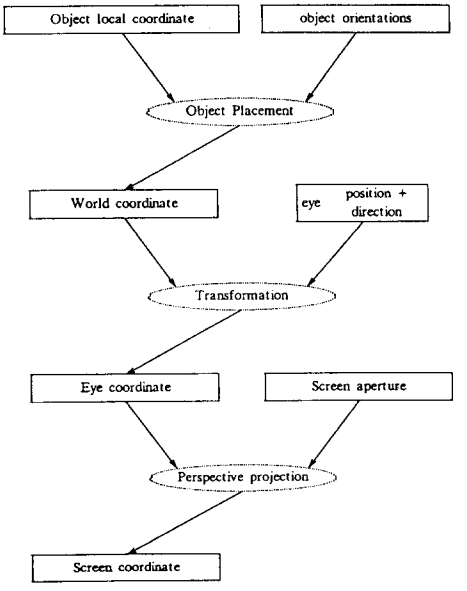


그림 5. Transformation과 projection

그림 5에는 4가지 좌표계가 나타나 있다. 이와 같이 여러개의 좌표계가 사용되는 이유는 물체의 정의와 알고리듬의 적용을 편하게 하기 위해서이다. 자동차를 예로들면 자동차는 타이어, 문, 차체 등으로 이루어진다. 이러한 부속품들은 저마다의 좌표계에서 정의되는데 이 좌표계를 그림 5에서는 object local coordinate라고 명명되어 있다. 각각의 부속품들이 자동차를 구성하기 위해서는 제 위치에 놓여져야 한다.

이것을 object placement라고 하며 object orientation은 놓여지는 위치, 방향 등을 지정해 준다. 물론 이 부속품들은 더 작은 부속품들로 구성되며 그것들도 제각기의 좌표계를 갖게 되는데 어떤 장면을 구성하는 object domain은 이와 같이 각각의 좌표계를 갖는 물체들의 계층적(hierarchical) 구조로 구성된다.

이 object domain이 갖는 좌표계를 world coordinate라고 부르는데 눈의 위치와 방향, 광원의 위치등도 world coordinate상에서 정의된다. 그런데 world coordinate system은 은면제거(hidden surface removal) 알고리즘을 적용하기가 어려우므로 적절한 좌표변환을 하게 된다. 은면제거는 눈으로부터의 가시성(visibility)를 결정하는 것이므로 눈의 위치와 방향을 될 수 있는대로 단순화시킬 필요가 있다. 즉 눈의 위치를 원점이 되게 그리고 눈의 방향을 +Z축 방향이되게 좌표계를 변환시키게 되는데 이 과정을 view-transformation이라고 하며 이렇게 해서 얻은 좌표계를 eye coordinate라고 부른다. (그림 5 참조)

여기에 원근감을 넣어주기 위하여 (X, Y)-coordinate값을 Z-coordinate 값으로 나누어 주는 과정을 perspective projection이라 하며 display device의 physical coordinate로 바꾸기 위해서 적절히 linear transformation을 함으로써 screen coordinate를 얻을 수 있다. 은면제거 작업은 보통 제일 마지막 좌표계인 screen coordinate system에서 일어난다.

은면제거에는 Z-buffer algorithm, painter's strategy, quadtree algorithm 그리고 scanline algorithm등이 있다. (여기서 ray tracing은 따로 취급하기로 한다.)

Frame buffer는 각 화소의 color data만을 가지는데 반하여 Z-buffer는 Z-depth 값-screen coordinate상의 Z축 값으로 눈으로부터의 거리로 삼을 수 있다-을 갖는다. 초기에 Z-buffer는 $+\infty$ 값으로 set 되며 어떤 화소가 rendering(묘화)될 때 그 화소가 이미 가지고 있는 Z값이 현재 rendering 하고자 하는 물체의 Z값에 비하여 크면 frame buffer의 color 값과 Z-buffer의 Z 값을 update시키고 그 반대이면 그냥 놔두는 방법이 Z-buffer algorithm이다. Z-buffer를 구성하기 위해서는 방대한 memory를 추가로 요구하는 문제점이 있었으나 근래에 와서 반도체 memory 칩의 대용량화, 저가격화로 인하여 현재 market에 나오는 모든 graphics super workstation의 graphics hardware가 이 algorithm으로 구현되어 있다. (IRIS, Titan, DN10000, GS1000 등). Painter's strategy란 말 그대로 화가가 그림을 그리는 전략을 말한다. 즉 멀리 있는 물체부터 그리자는 방법으로 sorting을 필요로 하며 물체들의 순서가 명백하지 않을 경우에는 Newell-Newell-Sancha algorithm으로 물체를 분할하여 서로 명백한 순서를 정해 주어야 한다.

Quadtree algorithm은 divide-and-conquer 전략을 따름으로써 screen domain에서의 일치성(coherence)을 최대한 활용하는 방법인데 Warnock에 의해서 제안되었다. Image screen이 quadtree에 의해서 어떻게 정의되는가가 그림 6에 설명되어 있다. 즉 screen을 네개의 동일한 크기를 갖는 영역으로 계속 쪼개면 전체 screen을 quadtree로 대응시킬 수 있다. 쪼개진 subscreen 영역을 그것과 물체와의 상관관계에 따라 surrounding, intersecting, containde, disjoint의 네가지 상태로 구분하여 visibility를 판정한다.

이 algorithm은 recursion을 수행되는데 sub pixel 영역까지 쪼갬으로써 adaptive super-sampling 에 의

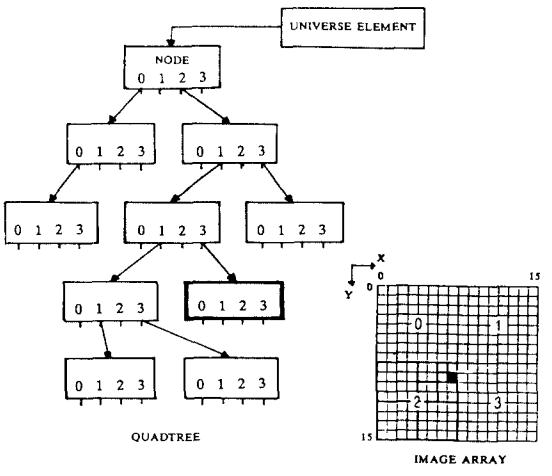


그림 6. Quadtree에 의한 screen정의

한 anti-aliasing을 할 수 있다.

Scanline algorithm은 지금까지 가장 널리 쓰였던 방법으로 Z-buffer algorithm이 방대한 memory를 요구하는 문제점을 해소하기 위한 scanline Z-buffer algorithm과 painter's strategy를 1개의 scanline으로 축소시킨 spanning scanline algorithm의 두가지가 있다. 물체를 Y축 방향으로 sorting하는 preprocessing을 요구한다.

조명모델(light modeling)은 광원의 빛이 물체의 표면에서 반사, 투과, 흡수되는 성질을 수식화하는 것이다. 사실성(realism)에 상당히 영향을 미치므로 정확한 조명모델은 매우 중요하다. 보통 광원은 삼원색(RGB)으로 모델링된다. 광원에서 나온 빛은 물체의 표면에서 확산 반사(diffuse reflection)와 윤택 반사(specular reflection)를 일으키며 거기에 난반사에 의한 환경빛(ambient light)을 각 화소의 색깔을 결정하는 3요소로 취급하는 모델이 널리 쓰이고 있다. 이들을 어떤 수식으로 modelizg 하느냐에 따라 phong model, sparrow-torrance model, hall model등이 있다. 부드러운 물체의 표면을 작은 다각형으로 근사화 시켜서 표현하는 것을 tessellation 이라고 하는데 아주 작은 다각형으로 쪼개지 않으면 다각형의 갯수가 너무 많아져서 메모리 문제 뿐만 아니라 계산시간도 길어지는 문제점이 있다. 그런데 선형 보간(linear interpolation)에 의하여 이런 문제에 부딪히지 않고 부드러운 곡면을 표현하는 임기응변적 방법이 있는데 gouraud shading과 phong shading이 그것이다. 이들은 모두 앞서 언급한 scanline hidden surface algorithm에 적용된다. 그림 7은 polygon들로 이루어진 다면체의 일부를 보여주고 있다.

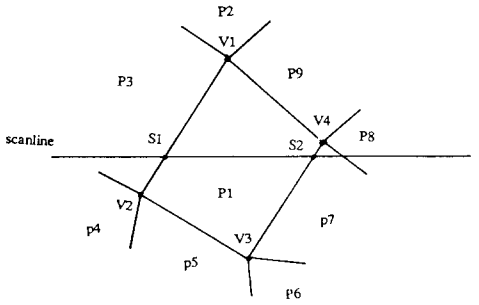


그림 7. Linear interpolation for a scanline

각 꼭지점에서의 surface normal vector는 그것을 포함하는 다각형들의 surface normal vector의 평균으로 구해지며 그 꼭지점에서의 빛의 세기를 결정하는 데 사용된다. 그림 7에서 점 S1에서의 빛의 세기는 V1과 V2에서의 빛의 세기를 보간(interpolation)에서 구하고 S2에 대해서는 V4와 V3에서의 값들을 보간함으로써 구하며 S1과 S2 사이에 있는 화소에서의 빛의 세기는 S1, S2에서의 값을 보간해서 채운다. 이것을 gouraud shading이라 하며 대부분의 graphics super workstation은 이 방법으로 shading을 수행한다. 반면에 phong shading은 각 꼭지점에서의 surface normal로 빛의 세기를 구하는 것이 아니라 surface normal을 계속적으로 보간하여 내려가 pixel level에 가서야 비로서 빛의 세기를 구한다. 이것은 빛의 세기를 구하는 과정이 nonlinear한 계산과정이므로 그 과정을 가장 마지막에 수행함으로써 오차를 줄이려고 하는 것으로 output image의 사실성이 gouraud shading의 것보다 좋다. 그러나 3개의 숫자로 이루어지는 surface normal을 보간하므로 빛의 세기만을 보간하는 gouraud shading에 비해 보간을 위한 계산량은 3배가 된다.

관측자 (눈)의 위치가 광원과 일치하지 않을 때는 그림자가 생긴다. 그림자는 물체상호간의 관계를 인지시켜 주는데 도움이 되기 때문에 image의 사실성을 높여준다. 특히 건물의 설계에 있어서 태양의 궤도 변화에 따른 일조상태의 파악에도 긴요하게 쓰일 수 있다. Shadow (그림자 계산)의 대표적 방법으로는 Weiler-Atherton algorithm, Crow's shadow volume method, 그리고 광선 추적 (ray tracing) 등이 있다. Wieler-Atherton algorithm은 polygon clipping을 이용한 방법이고 shadow volume 방법은 광원과 물체의 silhouette에 의해서 생성되는 그림자 영역을 가상적 volume으로 취급하는 것으로 scanline algorithm과 잘 부합되어 많이 응용되고 있으며 global illumination 기법에 속하는 ray tracing은 광선과 교차하는 물체 표면상의 교차점에서 광원을 향하여 ray를 쏨으로써 직접적으로 그림자의 존재 여부를 검사한다. 그런데 단순히 기하학적 계산만으로 어떤 물체 표면상의 한점이 그림자에 속하는가 아닌가만을 검사하다 보면 반그림자(panumbra)와 같은 사실성을 묘사할 수 없다.

현실적 광원이란 점 광원(point light source)이 아니라 면/체 광원(area/volume light source)이므로 그림자 지는 부분과 그렇지 않은 부분의 경계선에는 반드시 반그림자가 지게 마련이다. 일본과 미국에서

이미 이 분야에 관해 많은 연구가 이루어진 바 있다.

Ray tracing 기법은 1968년 Appel에 의해서 제안되었으나 계산량이 너무도 방대하여 실용화되지 못하고 있다가 1980년에 Whitted에 의해서 bounding volume의 hidrarchy를 이용하여 intersection 계산의 획기적 감소가 입증되면서부터 활발한 연구가 시작되었다. 앞서 언급한 global illumination은 광원과 물체 표면의 orientation만으로 밝기를 결정하는 local illumination과는 달리 광원과 물체, 물체와 물체의 관계를 모두 고려하는 것으로 광원에서 방사된 광선(ray photon)의 복잡한 행적을 하나 하나 추적하겠다는 의지를 포함하고 있다. 그런데 광원에서 나온 ray 중에서 극소수만 눈에 도달하므로 광원에서부터 눈으로의 경로 추적(forward tracing)은 매우 큰 계산의 낭비를 초래하므로 ray는 눈에서부터 추적하는 backward tracing이 효율적이다. 그림 8은 backward tracing에 의한 ray tracing 과정을 설명하고 있다. 눈과 screen의 orientation은 초기 ray의 parameter를 결정해 준다. 눈에서 출발된 ray는 screen상의 pixel의 중심을 통과한 후 object domain에서 물체를 만나면 그 교차점에서 반사와 투과를 일으켜 두 갈래의 ray로 나누어진다. 이 과정은 recursive하게 수행되어 binary tree로 match 될 수 있는데 그림 8에 대한 tracing tree가 그림 9에 나타나 있다. 그런데 어떤 ray는 object domain을 벗어나기도 하고 또 pixel intensity에 대한 contribution이 떨어져서 더이상 추적이 무의미해 지기도 하므로 tracing tree는 반드시 balanced binary tree가 되지는 않는다. 경로 추적에 의해서 생성된 tracing tree를 retrieval 함으로써 빛의 세기를 계산한다. 오늘날의 graphics super workstation에서는 ray tracing의 적용이 이루어지지 않고 있으나 processor의 computing power가 계속 증가하고 있고 적절한 다중처리 architeture가 개발되면 머지않아 이루어질 것으로 예상한다.

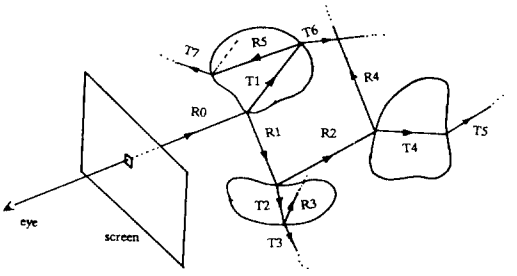


그림 8. Ray tracing

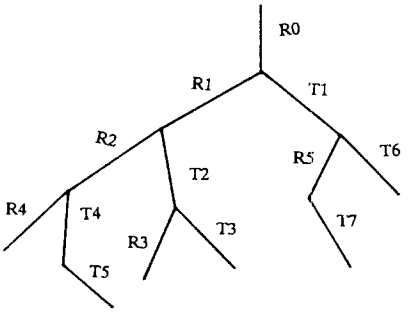


그림 9. 그림 8의 tracing tree

Ⅳ. Graphics Hardwares

미니 컴퓨터를 사용했을 때 software에 의한 rendering 시간은 가장 빠른 scanline algorithm의 경우에도 수분에서 수십분이 소요된다. 더구나 ray tracing의 경우 소요시간은 수시간에서 며칠이 소요된다. 따라서 실시간 처리를 요구하는 그래픽 워크스테이션은 그래픽을 위한 전용 하드웨어가 필수적으로 요구된다. 본고에서는 지금까지 미국, 일본 등지에서 개발된 대표적 그래픽 하드웨어 시스템을 적용된 알고리듬별로 소개하고자 한다.

먼저 scanline 알고리즘을 사용한 하드웨어 시스템 중에서 마이크로 프로세서를 이용한 IBM의 shading engine[1], Kyoto 대학의 EXPERTS[2]를 들 수 있다. 이들은 공통적으로 화면을 수직으로 분할하고 각 분할된 화면에 1개의 scanline 프로세서를 할당하는 parallel-pipeline 구조로 되어 있는 것이 특색이다. 또 scanline 알고리즘을 custom IC로 구현한 예로는 super buffer[3], SLAM[4]등을 들 수 있고, Z-buffer 알고리즘을 응용한 것으로 pixel planes[5], point classifier[6]를 들 수 있다. 특히 pixel planes는 frame buffer의 기능을 확장시켜 각 화소가 계산기능 (ALU)를 갖추고 있어 parallelism이 전 화면에 걸쳐서 일어나며 point classifier는 CSG(constructive solid geometry) 형태로 정의된 물체를 처리하도록 설계되어 있다. Ray tracing 하드웨어로는 65개의 마이크로 프로세서로 구성된 Osaka 대학의

LINK 시스템[7]을 들 수 있고 bezier 표면과 ray와의 교차점 계산을 위한 전용 chip을 설계한 예는 HP의 것[8]이 있다. Ray tracing을 위한 전용 하드웨어의 구조는 아직 초보 단계에 있다.

Market에 나오고 있는 graphics super workstation들은 대부분 고유의 graphics engine을 ASIC으로 만들어 장착하고 있다. Silicon Graphics사의 IRIS GTX grpahics architecture는 대략 7가지의 ASIC chip 50개로 구성되고 parallel-pipeline으로 구동하며 묘화 속도를 높이기 위하여 화면을 20개로 분할하도록 되어 있다. 초당 120K polygons/sec의 묘화 속도를 갖는다. 이외에도 Ardent사의 Titan, Stellar사의 GS 1000, Apollo(HP)사의 DN10000등의 워크스테이션도 100K polygons/sec 이상의 고성능 그래픽 기능을 보유하고 있다.

V. 표준 라이브러리와 X-Window

응용 소프트웨어의 보편성(machine-independency)를 증대시키기 위해서 ANSI 및 ISO 등의 기구에서는 CGI(computer graphics interface), GKS(graphics Kernel system), PHIGS(programmer's hierarchical interactive graphics system) 및 3-D GKS와 같은 표준 software interface의 채택을 진행시키고 있다. 응용 소프트웨어 프로그래머는 이러한 표준 소프트웨어를 이용하여 하드웨어의 개별적 특성을 고려하지 않고도 원하는 응용 소프트웨어를 만들어 낼 수가 있다. 이중에서 3차원 그래픽을 위해서 가장 높이 평가받고 있는 interface는 PHIGS로서 Apollo, Sun, Ardent, Stellar등 대부분의 high-end workstation 메이커들이 자사의 시스템에 포함시켜 놓고 있다. Network-Transparency, 계층적 윈도우 시스템을 위해서 MIT에서 개발된 X-window는 초기에는 2D graphics 기능만이 포함되어 있었으나 3D graphics 기능에 대한 요구가 일반화되기 시작하면서부터 version11이후로는 PEX(PHIGS extension for X)가 추가되었다. PEX protocol은 client와 server 사이의 통신을 정의하는데 주로 socket이나 shared memory 형태로 실현한다.

VI. 결 론

워크스테이션의 man-machine interface 기능을 담당하고 있는 컴퓨터 그래픽스의 기술을 2D, 3D로 나누어 소개하고 현재까지 발표된 그래픽스 하드웨어를 살펴 보았다. 고부가가치를 창출하기 위해서 워크스테이션의 그래픽 기능은 사실적이며 실시간으로 이루어질 수 있어야 하고 이를 위해서는 ASIC 접근 방법에 의한 전용 하드웨어의 설계 및 제작에 연구의 초점을 맞추어야 할 것이다. 이와 병행하여 우리나라에 ASIC의 용이한 제작을 가능케 하는 환경이 하루 속히 구축되어야 할 것이며 시스템 아키텍춰 설계 능력이 절실히 요청된다 하겠다.

參 考 文 獻

[1] K.C. Nunan, A.G. Lean, and C.P. Wang, "The Shading Engine-A Low Cost Parallel Processor for Rendering High Quality Color Images in Graphics Workstation," Proceeding 1st International Conference on Computer Workstation 1985, pp. 38-44.

[2] Haruo Niimi, Yoshirou Imai, and Massayoshi Murakami, "A Parallel Processor System for 3-D Color Graphics," Computer Graphics (SIGGRAPH), 1984, pp. 67-76.

[3] N. Gharacholoo, C. Pottle, "SUPER BUFFER: A Systolic VLSI Graphics Engine for Real Time Raster Image Generations," 1985 Chapel Hill Conference on VLSI, pp. 285-305.

[4] S. Demetrescu, "High Speed Rasterization Using Scan Line Access Memories," 1985 Chapel Hill Conference on VLSI, pp. 221-243.

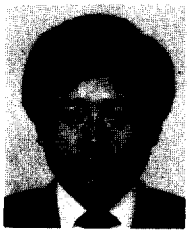
[5] J. Poulton, H. Fuchs, J.D. Austin, J.G. Eyles, J. Heinecke, C.H. Hsieh, J. Goldfeather, J.P. Hultquit, S. Spach, "Pixel-Planes: Building a VLSI-Based Graphics System," 1985 Chapel Hill Conference on VLSI, pp. 35-60.

[6] G. Kedem, S.W. Hammond, "The Point Classifier: A VLSI Processor for Displaying Complex 2-D objects," 1985 Chapel Hill Conference on VLSI, pp. 337-392.

[7] H. Nishimura, H. Ohno, T. Kawata, I. Shirakawa, and K. Omura, "LINKS-1: A Parallel Pipelined Multimicrocomputer System for Image Creation," IEEE 1983 Conference Proceedings of the 10-th Annual International Symposium on Computer Architecture, pp. 387-394.

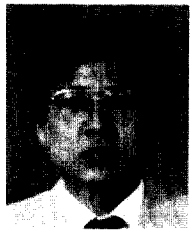
[8] R. Pulleyblank and J. Kapenga, "The Feasibility of a VLSI Chip for Ray Tracing Bicubic Patches," *IEEE CG & A*, vol. 7, no. 3, pp. 33-44, March 1987.

筆 者 紹 介



魚 吉 秀

1958年 2月 16日生
1982年 2月 서울대학교 전자공학과 졸업(학사)
1984年 2月 KAIST 전기 및 전자공학과(석사)
1989年 2月 KAIST 전기 및 전자공학과(박사)
1989年 3月～현재 삼성종합기술원 정보시스템 연구소 선임연구원
관심분야 : Computer Graphics, VLSI CAD, Special Purpose Hardware등



黃 時 永

1953年 9月 27日生
1976年 2月 서울대학교 계산통계학과 졸업(학사)
1978年 2月 KAIST 전산학과(석사)
1986年 2月 KAIST 전산학과(박사)
1978年～1980年 삼성전자 컴퓨터부문 개발실
1985年～1987年 삼성전자 5연구실 선임연구원
1987年～현재 삼성종합기술원 정보시스템 연구실장
관심분야 : Parallel Processing System, Network Architecture, Computer Graphics, Distributed Processing등