□특 집□

Fast Phong Shading을 위한 Rasterizer Architecture의 설계

삼성종합기술원 최상길·위영철·황시영*

목 차

Ⅰ. 서 론

최근 자동화와 정보화의 진전으로 컴퓨터 그래픽스의 응용분야가 급속히 확대되고 있으며 CAD/CAM, 컴퓨터 애니메이션, 시뮬레이션과 가상현실 등에서 많은 양의 정보를 효과적으로 처리함을 요구함에 따라 photorealistic 그래픽을 실시간으로 처리하기 위한 시스템의 개발이 활발히 진행되고 있다. 본 논문에서는 photorealistic image를 실시간으로 생성하기 위한 새로운 fast Phong shading 하드웨어 아키텍처에 관하여 논의 하고자 한다.

컴퓨터 그래픽스에서 물체의 표면을 채색하는 방법은 곡면을 직접적으로 처리하는 방법(Ray tracing, Radiosity)과 다각형으로 분할하여 칼라값을 구하는 방법(Gouraud shading, Phong shading)으로 크게 나눌 수 있다. 일반적으로 실시간 처리 컴퓨터 그래픽스 시스템에서는 많은 계산량을 줄이기 위하여 후자와 같은 방법을 사용한다. 이 시스템은 물체의 곡면을 작은 평면 다각형의 mesh로 나타내고 다각형의 각 꼭지점에 대한 3D transformation, light 계산, clipping, 2D projection과 다각형을 삼각형으로 분해하는 일을 수행하는 *geometry processing* 부분과 삼각형 내부의 각 화소에 대한 칼라값을 결정하는 *rasterizer* 부분으로 구성되어 있다[1].

Gouraud shading은 *geometry processing* 부분에서 계산된 삼각형의 꼭지점의 칼라값을 근거로 하여 *rasterizer*에서 삼각형 내부의 화소에 대한 칼라값을 linear interpolation하여 구하기 때문에 연산이 비교적 간단하고 ASIC으로 구현하기가 용이하다. 그러므로 비록 image quality는 Phong shading보다 떨어지지만 대부분의 시스템들이 주로 Gouraud shading을 이용한다. Phong shading은 image quality는 Gouraud shading보다 좋지만 삼각형 끝점들의 normal vector를 linear interpolation하여 식 (1), (2)와 같은 diffuse reflection light model과 specular reflection light model을 계산해야 하므로 연산이 복잡하고 ASIC화 하기가 용이하지 않다. 그러므로 Phong shading은 다량의 상용 microprocesser를 parallel로 연결하여 구성한 몇몇 시스템에서만 수행되고 있다.

$$F_d = I_{diffuse} = I_d K_d (L \cdot N)/|L|\,|N| \quad (1)$$

$$F_s = I_{specular} = I_s K_s \cos^n \alpha \quad (2)$$

* 정회원

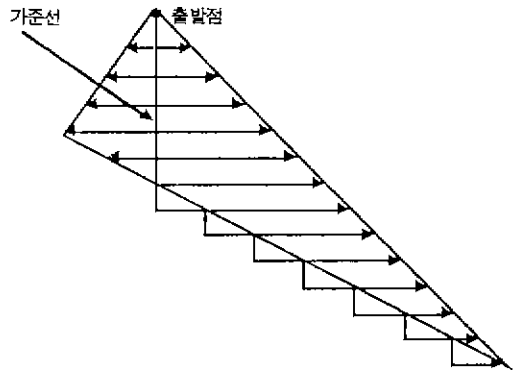
이와 같이 복잡한 연산을 요구하는 Phong shading을 간단한 연산으로 근사치를 구하는 방법이 Bishop[2]에 의해 제안되었다. 즉, 식 (1)의 $(L\cdot N)/|L|\,|N|$을 직접 계산하기 위해서는 7 addition, 6 multiplication, 1 division, 1 squre root가 소요된다. 이러한 연산을 줄이기 위하여 Bishop은 $(L\cdot N)/|L|\,|N|$을 Taylor series로 approximate한 다음 quadratic interpolation에 의하여 각 화소당 2 addition으로 구할 수 있는 방법을 제시하였다. 그리고 specular light model도 역시 같은 방법으로 계산하여 Phong shading의 근사치를 구하는 방법을 고안하였다. 이 방법을 Fast Phong shading이라고 한다.

Rasterization은 전통적으로 삼각형을 각 scanline에 대한 span들로 나누고 그 끝 점들의 칼라값과 depth 값을 구한 뒤, 이를 근거로 하여 각 span 내부의 화소에 대하여 칼라 값과 depth 값을 linear interpolation하여 수행하였다[3, 4]. Pineda[5]는 삼각형 내부를 traversal하면서 edge function을 이용하여 삼각형의 경계를 벗어나는가를 확인하고 칼라 값을 interpolation하는 새로운 방법을 제안하였고 [6]에서 이를 하드웨어로 구현하였다. 그런데 삼각형 rasterization 에서 처리해야될 데이타량은 삼각형 내부의 각 화소를 어떻게 탐색하느냐에 따라 크게 좌우되며 이는 곧, 그래픽스 시스템의 전체적 성능에 영향을 미친다. 따라서 시스템의 성능 즉, 데이타 처리 속도를 높이기 위하여 효율적인 삼각형 탐색 방법이 요구된다. Pineda의 삼각형 탐색 방법은 하드웨어로의 구현이 단순한 새로운 방법이나 비효율적인 traversal 방법으로 인하여 최대의 성능을 발휘하지 못하는 단점을 가지고 있다.

본 논문에서는 fast Phong shading을 가속화하기 위하여, triangle을 quadratic interpolation하는데 적합한 triangle traversal 방법과 처리 속도를 극대화하기 위해 pipeline 방식으로 구성한 rasterizer architecture를 제안한다.

II. Previous work

대표적인 삼각형 탐색 방법인 Pineda 방법을



(그림 1) Pineda의 삼각형 탐색 방법

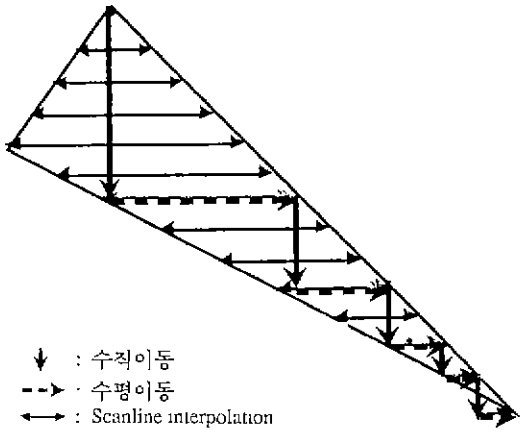
살펴보면, 그림 1과 같이 출발점의 x좌표를 기준으로 기준선을 따라 −y방향으로 내려오면서 각 y단계에 대해 +x, −x 방향으로 삼각형의 경계선까지 탐색한다. 또 기준선이 삼각형의 한 변을 벗어나게 되면 그 변을 따라 계단 모양으로 탐색한다. 이 방법은 삼각형 외부에서의 불필요한 탐색과 계산을 수행하게 되고 이에 따라 처리 속도가 길어지게 되는 단점이 있다. 그리고 기준점을 벗어난 변의 길이가 길면 길수록 삼각형 외부에 대한 불필요한 탐색과 연산이 커진다.

지금까지 개발된 대부분의 rasterizer(ASIC)는 모두 Gouraud shading을 처리할 수 있도록 제작되어 fast Phong shading은 수행할 수가 없는 구조이다.

III. 삼각형 탐색 방법

삼각형 탐색의 최초 출발점은 꼭지점의 y값이 최대인 꼭지점으로 하고 탐색이 종료되는 종착점은 y값이 최소인 꼭지점으로하여 위에서 밑으로 하향 이동하면서 탐색해 나간다. 탐색이 종료되는 점까지의 탐색 수렴을 위하여, 최소 y값을 갖는 꼭지점의 x값이 다른 꼭지점들의 x 값들의 가운데 값일 경우에는 y값이 최소인 꼭지점을 출발점으로하여 위로 상향 이동하면서 탐색해 나간다.

본 논문의 삼각형 탐색 방법은 그림 2와 같이 scanline interpolation과 탐색 기준점의 수직이동 및 수평이동이 되풀이됨으로써 수행된다. 먼저



(그림 2) 새로운 삼각형 탐색 방법

scanline interpolation은 기준점을 중심으로 +x방향과 −x방향으로 R, G, B 그리고 Z값을 interpolation하여 각 화소의 R, G, B, Z 값을 계산하는 동작이다. 즉, 기준점의 R, G, B, Z값과 x방향으로의 R, G, B, Z 기울기인 dR/dx, dG/dx, dB/dx, dZ/dx값들은 register에 저장되어 있으므로 화소의 x address가 1씩 증가 또는 감소함에 따라 기준점의 R, G, B, Z값에 계속적으로 x방향 R, G, B, Z 기울기값을 가감함으로써 수행된다. Scanline interpolation되는 화소가 삼각형의 경계를 벗어나는가는 Pineda 방법에서 사용된 삼각형 각 변에 대한 edge function의 interpolation에 의해 판별된다. Edge function값은 해당되는 변을 기준으로 임의의 좌표값에 대하여 그 점이 삼각형 내부에 있으면 양의 값을 가지고 외부에 있으면 음의 값을 가진다. Edge function의 계산은 interpolation에 의하여 간단히 구할 수 있다.

기준점의 수직이동은 기준점에 대한 scanline interpolation이 완료된 후에 −y방향(또는 +y방향)으로 한 화소만큼 이동한 화소의 R, G, B, Z값을 interpolation하여 기준점 register에 저장함으로써 수행된다. 수평이동은 다음에 수직이동될 기준점이 삼각형의 한 변을 벗어날 경우에 수행되며, 탐색 기준점 이동방향 결정기에 따른 방향에 대해 마지막 scanline interpolation된 화소의 R, G, B, Z값을 기준점 register에 저장하는 동작을 말한다. 수평이동된 후에는 다시 수직이동과 scanline interpolation을 행한다.

수평이동시 탐색 기준점의 이동방향 결정은 기준점의 수직이동시에 벗어난 변의 기울기를 이용한다. 기준점의 수직이동이 −y방향인 경우, 기준점이 벗어난 변의 기울기가 양이면 −x방향이고 음이면 +x방향이 된다. 기준점의 수직이동이 +y방향인 경우는 이와는 반대로 기준점이 벗어난 변의 기울기가 양이면 +x방향이고 음이면 −x방향이 된다.

이와 같은 삼각형 탐색 방법은 탐색 기준점이 일반적으로 삼각형 내부에 존재함에 따라 삼각형 외부에 대한 불필요한 계산을 없앨 수 있다. 그러나 종착점에 연결된 두 변의 사이각이 작을 경우에는 종착점 가까이에서 기준점이 1화소만큼 외부로 나갈 수도 있다.

IV. Fast Phong shading을 위한 quadratic interpolation 방법

앞에서 설명하였듯이 Bishop은 diffuse reflection model값과 specular reflection mode값인 F_d, F_s를 Taylor series로 식 (3)과 같이 x, y에 대한 이차식으로 근사화하였다[2]. 근사화에 의한 오차는 Phong shading의 표면 법선 벡터 보간과 diffuse, specular reflection 식 자체가 이미 근사화된 것이므로 그리 문제가 되지 않는다.

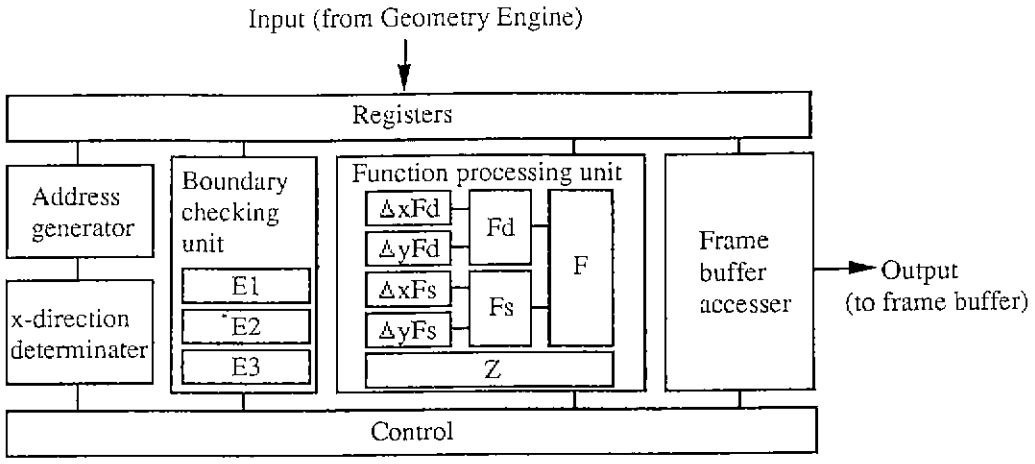
$$F_d(x, y)=F_s(x, y)=T_5x^2+T_4xy+T_3y^2+T_2x+T_1y+T_0 \quad (3)$$

F_d, F_s를 F라고 하면 F는 x, y에 대한 이차식이고 F의 x, y에 대한 미분값인 Δ_xF, Δ_yF는 각각 x, y에 대한 일차함수이며 Δ_x^2F, Δ_y^2F, $\Delta_x\Delta_yF$, $\Delta_y\Delta_xF$는 상수가 된다. 그러므로 forward difference 방법에 의하여 F는 식 (4), (5)와 같이 구할 수 있다.

$$F(x+1, y)= F(x, y)+\Delta_xF(x+1/2. y)$$
$$F(x, y+1)=F(x, y)+\Delta_yF(x, y+1/2) \quad (4)$$

여기서 $\Delta_xF(x+1/2, y)$와 $\Delta_yF(x, y+1/2)$는 식 (5)와 같이 구할 수 있다.

$$\Delta_xF(x+1/2, y)=\Delta_xF(x-1/2, y)+\Delta_x^2xF$$
$$\Delta_xF(x, y+1/2)=\Delta_xF(x, y-1/2)+\Delta_y\Delta_yF \quad (5)$$



(그림 3) 삼각형 rasterizer의 구조

$\Delta_y F(x, y+1/2) = \Delta_x F(x, y-1/2) + \Delta[\frac{}{}]F$

$\Delta_y F(x+1/2, y) = \Delta_v F(x-1/2, y) + \Delta_x \Delta_v F$

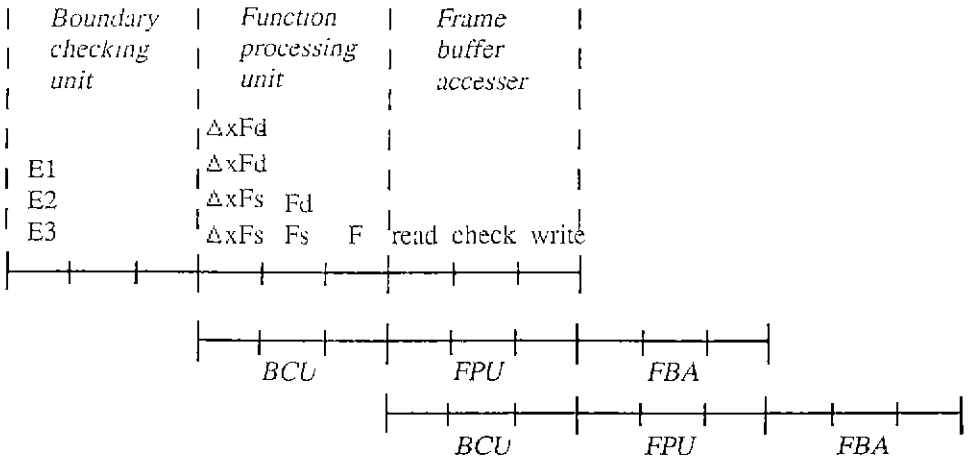
Quadratic interpolation은 식 (4), (5)에 의해 인접한 좌표의 F 및 $\Delta_x F$ 또는 $\Delta_y F$를 근거로 하여 두번의 덧셈을 수행함으로써 실행된다. 이러한 quadratic interpolation은 이차식으로 모델링되는 구(sphere)같은 것도 다각형으로 근사화하지 않고 바로 처리할 수 있으며, texture mapping을 수행할 때 texture address 생성에도 적용될 수 있다.

V. Rasterizer architecture

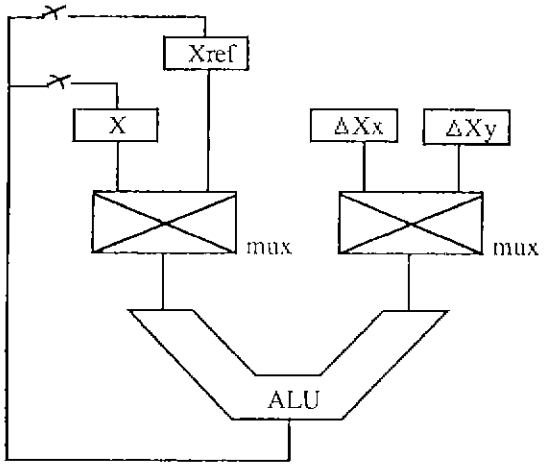
삼각형 rasterizer[7]는 그림 3과 같이 register 블록과 주소생성부(Address generator), 탐색 기준점 이동방향 결정기(x-direction determinater), 경계 판별기(Boundary checking unit), Function processing unit, 메모리 억세서부(Frame buffer accesser), control부로 구성된다.

Register 블록은 탐색 출발점에서의 R, G, B, Z 값과 edge function E1, E2, E3 값 그리고 이들의 x, y에 대한 기울기 값 등의 register들로 구성되어 있다. 이들은 geometry engine으로부터 입력된 값으로 초기에 setting된다. 주소 생성부(Address generator)는 탐색시에 해당되는 화소의 메모리 주소를 생성하는 일을 수행한다. 탐색 기준점 이동방향 결정기(x-direction determinater)는 기준점의 수평이동시 방향을 결정하는 일을 한다. 경계 판별기(Boundary checking unit)는 삼각형 탐색시 대응하는 화소가 삼각형의 경계를 내부에 위치하는가 또는 외부에 위치하는가를 판별하는 일을 수행하며 삼각형의 각 edge에 대한 edge function 값 E1, E2, E3를 interpolation하여 구한 다음 판별한다. Function processing unit는 대응하는 화소의 Z값을 interpolation하여 구하고, 칼라 값 F(R, G, B)를 구하기 위하여 R, G, B 각각에 대하여 $\Delta_x F_d$, $\Delta_y F_d$, $\Delta_x F_s$, $\Delta_y F_s$, F_d, F_s와 F블록을 가지고 있다. $\Delta_x F_d$, $\Delta_y F_d$, $\Delta_x F_s$, $\Delta_y F_s$, F_d, F_s 블록은 식 (4), (5)와 같이 interpolation을 수행하고 F블록은 F_d와 F_s를 add하여 F값을 구한다. 메모리 억세서부(Frame buffer accesser)는 frame buffer로부터 Z값을 읽어온 다음 계산된 Z값과 비교하여 계산된 Z값이 더 큰 값(또는 작은 값)이면 R, G. B, Z를 frame buffer에 update 시킨다.

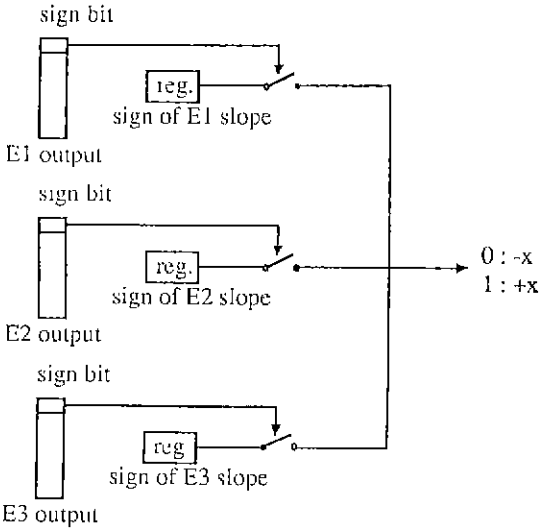
그림 3의 rasterizer는 대응하는 화소에 대하여 Boundary checking unit의 edge function interpolation, Function processing unit의 x, y에 대한 F_d, F_s의 일차미분값, F_d, F_s값, F값 그리고 frame buffer access순으로 연산 및 동작을 수행한다. 이 때, Boundary checking unit에서 1, Function processing unit에서 3, Frame buffer accesser에서 3의 machine cycle이 소요된다. 우리는 pro-



(그림 4) **Rasterizer pipeline operation**



(그림 5) $\Delta_x F_d$, $\Delta_y F_d$, $\Delta_x F_s$, $\Delta_y F_s$, F_d, F_s 와 E1, E2, E3를 위한 interpolater



(그림 6) 탐색 기준점 이동 방향 결정기

cessing 속도를 높이기 위하여 그림 4와 같이 각 블록별로 pipeline으로 동작하도록 구성하였다. Frame buffer accesser가 현재의 화소를 처리하고 있을 때 Function processing unit는 다음에 처리될 화소의 F값을 연산하고, Boundary checking unit는 다음 다음에 처리될 화소의 edge function값을 연산한다. 그러므로 하나의 화소에 대한 연산 및 처리를 3 machine cycle에 수행할 수 있다. 그런데, 정확한 칼라값을 계산하기 위해서는 F값에 상수값인 ambient term이 더해져야 하는데 이러한 adder를 더 추가하여 pipeline을 수행하려면 ambient term adder가 Function processing unit에 삽입되고 $\Delta_x F_d$, $\Delta_y F_d$, $\Delta_x F_s$, $\Delta_y F_s$블록이 Boundary checking unit가 수행하는 화소에 대하여 연산하도록 하면 역시 3 machine cycle에 동작이 가능하다.

그림 3에서 경계 판별기(Boundary checking unit)의 E1, E2, E3와 interpolater의 Function processing unit의 $\Delta_x F_d$, $\Delta_y F_d$, $\Delta_x F_s$, $\Delta_y F_s$, F_d, F_s 블록은 같은 구조의 interpolater로 그림 5와 같이 구현된다. 그림 5에서 X는 E1, E2, E3, $\Delta_x F_d$,

$\Delta_y F_d$, $\Delta_x F_s$, $\Delta_y F_s$, F_d, F_s를 나타낸다. X_{ref}는 기준점의 X값을 나타내는 register이고 X는 현재 화소의 X값, ΔX_x와 ΔX_y는 X의 x, y에 대한 기울기를 나타내는 register이다. X_{ref}와 X 그리고 ΔX_x와 ΔX_y는 각각 MUX에 의해 둘 중 하나가 ALU에 입력된다. Scanline interpolation시에는 먼저 X_{ref}에 ΔX_x를 더하여 (빼서) X에 저장하고 그 뒤로는 X에 ΔX_x를 더하여(빼서) X에 저장하는 동작을 수행한다. 기준점의 수직이동시에는 Xref에 ΔX_y를 더하여(빼서) Xref에 다시 저장한다. 기준점의 수평이동시에는 탐색 기준점 이동방향의 마지막 scanline interpolation을 수행하는 화소의 X값을 X_{ref}에 저장함으로써 수행된다.

탐색 기준점 이동방향 결정기는 각 변에 대한 edge function의 출력값의 부호가 음인 변의 기울기 부호에 의해 방향이 결정된다. 기울기 부호가 양이면 −x방향이고 음이면 +x방향이다. 이 결정기의 개념적 블록도는 그림 6과 같다.

VI. 결 론

본 논문에서는 triangle traversal을 수행하기 위한 Pineda의 삼각형 탐색 방법에서의 불필요한 삼각형 외부에 대한 계산을 줄이기 위한 새로운 삼각형 탐색 방법을 제안하였고, fast Phong shading을 가속화하기 위하여 pipeline 방식으로 구성된 quadratic interpolation을 수행할 수 있는 rasterizer architecture를 제안하였다. 그리고 이러한 방법을 하드웨어로 구현하기 위한 logic 회로를 C 언어로 simulation하여 알고리즘을 검증하였다.

참 고 문 헌

1. J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics : Principles and Practice, Second Edition*, Addison-Wesley, 1990.
2. G. Bishop and D. Weimer, "Fast Phong Shading", *Computer Graphics(SIGGRAPH '86 Proceedings)*, Vol. 20, Aug. 1986, pp. 103~106.
3. K. Akeley and T. Jermoluk, "High Performance Polygon Rendering", *Computer Graphics(SIGGRAPH'88 Proceedings)*, Vol. 22, No. 4, Aug. 1988, pp. 239~246.
4. D. Kirk and D. Voorhies, "The Rendering Architecture of the DN10000VS", 299~307.
5. Juan Pineda, "A Parallel Algorithm for Polygon Rasterization", *Computer Graphics(SIGGRAPH' 88 Proceedings)*, Vol. 22, No. 4, Aug. 1988, pp. 17~20.
6. G. J. Dunnett, M. White, P. F. Lister and R. L. Grimsdale, "The Image Chip for High Performance 3D Rendering", *Computer Graphics and Applications*, Vol. 12, No. 6, Nov. 1992, pp. 41-51.
7. 최상길, 최병균, 문상호, 위영철, "고성능 그래픽스 전용 프로세서 개발", 전자공학회지, Vol. 20, No. 7, 1993, pp. 841~849.

최 상 길

1985 부산대학교 전자공학과 공학사
1987 한국과학기술원 전기 및 전자공학과 공학 석사
1992 한국과학기술원 전기 및 전자공학과 공학 박사
1992 ~현재 삼성종합기술원 기반기술 연구소 선임연구원
관심 분야 : Computer Aided Design, Graphics Hardware

위 영 철

1983 미국 State University of New York at Albany 전산학과 이학사
1985 미국 State University of New York at Albany 전산학과 이학석사
1992 미국 State University of New York at Albany 전산학과 이학박사
1989 ~1990 State University of New York at Albany 연구원
1990 ~1993 삼성종합기술원 기반기술 연구소 선임연구원
1993 ~현재 삼성종합기술원 기반기술 연구소 수석연구원
관심 분야 : Computational Geometry, Graphics Hardware

황 시 영

1976 서울대학교 계산통계학과 졸업
1978 한국과학기술원 전산학과 공학석사
1986 한국과학기술원 전산학과 공학박사
1978 ~1980 삼성전자 컴퓨터사업부 개발근무
1985 ~1987 삼성전자 연구소 5연구실
1987 ~1990 삼성종합기술원 정보시스템연구소 연구 1실 실장
1990 ~1993 삼성종합기술원정보시스템연구소 I-Project 실장(MagicStation 개발)
1993 ~현재 삼성종합기술원 기반기술연구소 컴퓨터구조연구실 실장겸 연구원
관심 분야 : computer graphics, virtual reality, network architecture, distributed processing