論文 89-26-1-13

개인용 컴퓨터상의 그래픽스 인터페이스 설계와 응용

(Design and Applications of Graphics Interface on Personal Computer)

金 鎭 漢*, 慶 宗 旻*

(Jin Han Kim and Chong Min Kyung)

要 約

한국과학기술원의 설계자동화연구실에서 IBM PC/AT 상에서 동작하도록 제작한 그래픽스 보드 "K"를 구동시키기 위하여 소프트웨어 인터페이스인 CGI-K를 TI사에서 제공한 C언어와 어셈블리 언어를 이용하여 작성하였다. CGI-K를 구성하는 디바이스 드라이버 루틴과 네모, 원, 현 및 호와 같은 그래픽스 프리미티브를 만드는 알고리듬을 제안하였으며, CGI-K는 EGA(enhanced graphics adapter)의 CGI보다 3배에서 10배 정도 빠른 속도를 갖는 것을 보였다.

CGI-K를 이용하여 2차원 그래픽스 에디터인 GRIM과 3차원 모델링 프로그램인 IPCHE를 만들었으며, GRIM은 그래픽스 프리미티브를 이용하여 원하는 그림을 그릴 수 있도록 하였으며 IPCHE는 3차원 물체의 데이타를 받아들여 보이지 않는 면을 제거하고 명암과 원근 효과를 갖는 3차원 물체를 화면상에 그려 준다.

Abstract

A software interface called CGI-K including device driver routines and graphics primitives for the graphic board "K" was designed, implemented in the Design Automation Laboratory of KAIST and installed on IBM PC/AT, using assembly and C language supported by TMS 34010 graphics processor. Several algorithms generating the graphics primitives such as box, circle, pie chord are proposed. The drawing speed of CGI-K on the graphic board K was found out to be three to ten times faster than that of the EGA for several examples.

A 2-D graphics editor called GRIM (graphics input and modification) and a 3-dimensional graphics renderer called IPCHE which can draw 3-D objects were developed as two major application programs running on CGI-K. The graphics primitives supported in GRIM include polygon, box, circle, and ace. The IPCHE receives a 3-D objects data file and displays the 3-D object on the screen with hidden surface removal, shading, and perspective scaling.

I. 서 론

VLSI의 집적도가 막대해짐에 따라 연산 속도가 빠른 컴퓨터가 VLSI 설계에 도입되었다. 컴퓨터를 이용하여 VLSI를 설계할 때는 컴퓨터와 인간과의 인터페이스가 필요하며, 인터페이스 방법으로 컴퓨터 그래픽스가 필요하다. 또한 직접 시험하기에는 많은 비용과 시간이 들며, 위험성을 포함하는 일에는 컴퓨터를 이용하여 시뮬레이션을 하면 비용과 시간이

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Electrical Eng., KAIST)

接受日字 : 1988年 4月 27日

절약되고 위험성은 제거된다. 따라서 이러한 컴퓨터 시뮬레이션이 많이 이용되며 여기에도 컴퓨터 그래픽스가 사용된다. 이밖에도 computer art, animation, 게임등에도 컴퓨터 그래픽스가 절대적으로 필요하다. 컴퓨터 그래픽스를 위해서는 컴퓨터에서 처리된 정보를 화면에 그려 주는 매체가 필요하며, 이러한 매체로 그래픽스 보드와 이를 이용하여 그림을 그려주는 프리미티브, 디바이스 드라이버 루틴등을 포함하는 그래픽스 인터페이스가 필요하다. 이러한 컴퓨터 그래픽스에 이용되는 그래픽스 보드로서, 국내에서 개발된 것으로는 한국과학기술원 전기 및 전자공학과의 설계자동화연구실에서 개발한 보드 "K"가 있다.

본 논문에서는 개인용 컴퓨터인 IBM PC/AT 상에서 동작하는 그래픽스 보드를 구동시키는 그래픽스 인터페이스를 설계하고, 이를 응용한 프로그램을 작성하였다. 본 논문에서 이용하는 그래픽스 보드는 Texas Instruments사의 TMS 34010 그래픽스 프로세서 칩을 사용하여 만들었으며, IBM PC/AT 상에서 동작하도록 제작한 보드"K"이다. 보드 "K"는 4096가지 색 중에서 선택된 16가지 색을 동시에 보여주며, 640×480의 해상도를 갖는 보드이다. TI사에서 제공한 C언어와 어셈블리 언어를 이용하여 보드 "K"에서 동작하도록 작성한 디바이스 드라이버 루틴과 그래픽스 프리미티브등의 그래픽스 인터페이스는 ISO 표준인 CGI(computer graphics interface)와 대등한 기능을 가지므로 CGI-K라 명명 하였다.

Ⅱ. CGI-K

그래픽스 인터페이스없이 보드 "K"의 프로세서인 TMS34010 GSP(graphics system processor)가 제공하는 그래픽 명령어만으로는 사용자가 원하는 도형을 그리기가 어려우며, 많은 시간과 노력이 필요하게 된다. 또한 보드 "K"의 구조 및 TMS34010 GSP가 제공하는 명령어와 어셈블리 언어를 알아야 한다. 그러나 사용자의 대다수가 보드 "K"의 구조 및 TMS34010 GSP의 명령어와 어셈블리 언어를 모르며, 그리고자 하는 도형을 손쉽게 그리기를 원한다. 사용자가 손쉽게 보드K를 사용할 수 있도록 제작한 그래픽스 인터페이스가 CGI-K이다. CGI-K상에 정의된 프리미티브를 이용할때는 그리고자 하는 프리미티브가 요구하는 파라메타를 주고, 해당 프리미티브를 부르면 된다. 그러면, 해당 프리미티브가 불려질 때 주어진 파라메타에 따라 프리미티브를 화면상에 그려주게 된다.

CGI-K의 구성은 프로그램을 수행하기 위해 IBM PC/AT에서 컴파일된 파일을 보드 "K"로 전송하는 다운 로딩 부분과 TMS34010의 레지스터를 조절하여 사용 화면의 크기나 drawing mode 등을 컨트롤하는 컨트롤 펑션과 사용자가 그림을 그릴수 있도록 해주는 그래픽스 루틴등 세 부류로 구성되어 있다.

1. Control Function

컨트롤 펑션은 TMS34010 GSP의 레지스터를 이용하여 그림을 그리는 모드를 결정하는 루틴, 즉 transparency mode로 그릴 것인가 opaque mode로 그릴 것인가를 결정하고 기존의 그림과 새로운 그림이 겹칠 때 drawing operation을 결정하는 루틴이 있다. Drawing operation은 AND, OR, NOT, Replace 등 16가지의 boolean operation과 ADD, SUB 등 6가지의 arithmetic operation이 있다. 또한, 그리고자 하는 도형의 색을 조절하거나 화면의 정보를 갖는 프레임 버퍼의 내용을 읽어 들여 각 화소 단위의 프로세싱을 하거나 프레임 버퍼의 내용을 저장, 복사하는 루틴, TMS34070 palette 칩이 제공하는 4096가지 색에서 사용하고자 하는 16가지 색을 선택하는 루틴, TMS34010의 I/O 레지스터들의 값을 조절하여 vertical sync. 신호와 horizontal sync. 신호를 적절한 시기에 발생하도록 조절하고 blanking ratio, RAM refresh ratio, screen refresh ratio등을 결정하여 사용하는 CRT와 보드 "K"를 매칭시키는 루틴등이 있다. 이 외에, 사용자와 컴퓨터간의 인터페이스를 위해 이용되는 마우스를 인스톨 시키는 마우스 드라이버 루틴이 있는데, 마우스 드라이버 루틴에서는 IBM PC/AT와 TMS34010 GSP간에 상호의 정보를 교환하면서 마우스의 위치를 CRT상에 보여 주며, 그 과정은 다음과 같다. 먼저, IBM PC/AT에서 마우스의 위치 및 버튼의 상태를 읽고, 데이타를 읽은 IBM PC/AT는 TMS34010 GSP를 halt 시킨후 읽어 들인 데이타를 보드 "K"에 보내 준다. 데이타를 전송한 후 halt를 제거하면 보드 "K"는 이 데이타를 읽어 해당 위치에 사용자가 정의한 커서를 pixel block transfer를 이용하여 그려 준다. 커서를 그린다음 버튼의 상태를 검사하여 버튼이 눌렸으면 미리 지정된 일을 수행하고 버튼이 눌리지 않았으면 위의 과정을 반복한다. 이때, 새로운 위치에 커서를 그리기 전에 기존의 커서 위치의 데이타를 복원 시킨다.

2. Graphics Routine

사용자가 직접 이용하여 그림을 그리게 해주는 부분인 그래픽스 루틴은 사용자가 그리고자 하는 프리미티브를 지정한 위치에 그려주는 루틴의 집합으로

점, text, polymarker, 직선, 네모, 다각형, 원, 타원, 현, 호등 11개의 프리미티브와 이들 중 filling이 가능한 네모, 다각형, 원, 타원, 현, 호의 solid filling과 pattern filling한 형태의 프리미티브로 구성되어 있다.

가장 기초적인 프리미티브인 점은 화면상의 한 화소를 지정한 색으로 그려 준다. 이는 직선, 원, 타원 text등 다른 10개의 프리미티브를 구성하는 기본 단위가 된다. Text는 메시지나 어떤 상태를 표시하기 위해 필요하며, 모든 text를 서브루틴으로 만들어 각 서브루틴에서 text의 크기, 너비, 위치를 받아 line drawing 방법을 이용하여 text를 그려 준다. 서브루틴을 이용하므로 속도가 느린 반면 크기의 조절이 용이하고, 메모리가 절약되는 장점이 있다. Polymarker는 text와 함께 실험결과를 그리거나 지정하는 임의의 점을 표시하는데 사용하며 점, 화살표, 네모, X자, +자등 5가지 종류가 있으며 이들의 크기는 기본 크기의 정수배로 조절할 수 있다. CRT 화면상에 직선을 그리는 방법은 계산량은 많으나 오차가 적은 8WS방법과 Bresenham의 line drawing 알고리듬을 이용하였다.[3]

네모는 가장 많이 이용되는 프리미티브중의 하나로 네모를 구성하는 대각선상의 두 좌표를 받아 들인 후 4개의 직선을 그려 사각형을 그려 준다. 다각형은 n개의 좌표를 어레이 형태로 받아 들여 그 좌표를 어레이의 인덱스 순서 대로 직선을 그리고 처음의 점과 마지막 점을 잇는 직선을 그림으로써 구현할 수 있다. 원은 8WS line drawing 방법을 이용하여 구현할 수 있다. 이때 원의 1/8만을 구하면 대칭성에 의해 원의 전부를 구할 수 있으며, 원의 중심이 원점이 아닌 경우 중심을 원점으로 가정하여 계산한 후 구한 각 점에 중심의 좌표를 더해주면 주어진 중심과 반경을 갖는 원이 된다. 직선, 원과 함께 그래픽스 application에서 자주 이용되는 것은 타원이다. 타원을 그리는 방법은 원과 비슷하나 기울기의 변화에 따라 다음 단계에서 결정할 화소의 집합이 변하게 되는점이 존재한다. 이 점은 접선의 기울기가 −1이 되는 점이다. 타원도 원과 같이 1/4 부분만을 구하면 대칭성에 의해 전체 타원을 그릴수 있으며, 임의의 점을 중심으로 하는 타원을 구할 수 있다. 현, 호는 원형과 타원형의 2가지 경우가 존재하며, 이들 각각을 그리는 방법은 동일하다. 그러므로 원형의 호에 관해서만 설명 하고자 한다. 먼저 그리고자 하는 호를 그림 1과 같이 각 사분면에 해당하는 호로 분리한 후 각 사분면에 호가 존재하면 시계방향으로 시작점과 끝점을 결정하고, 각 사분면

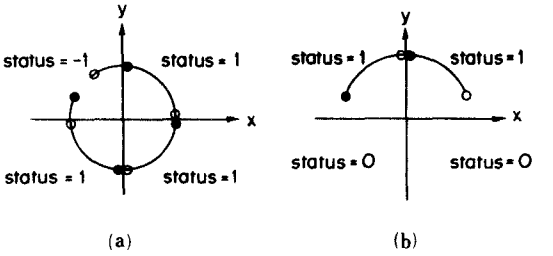


그림 1. Arc의 status
Fig. 1. Status of arc.

의 status를 결정한다. 각 사분면의 status는 −1, 0, 1의 값을 갖는데 0은 그 사분면에는 호가 존재하지 않는 경우이며 이때는 호를 그리지 않는다. Status가 1인 경우에는 시작점과 끝점 사이의 호를 그리는 경우이며 이들 사이의 호를 그려주고, −1인 때는 시작점과 끝점 사이의 호만을 그리지 않는 경우로서 해당 사분면의 시작점과 끝점 사이의 부분을 제외한 다른 부분만 호를 그린다. 이때 호는 원이나 타원을 구하는 방법과 동일하다. 이렇게 그린 호에서 시작점과 끝점을 직선으로 연결하면 현이 된다. 그림 1에서 검은 점은 시작점을 나타내며 흰점은 끝점을 나타낸다.

프리미티브의 2가지의 filling 방법중 Solid fill은 프리미티브의 내부를 지정한 색으로 전부 채워 넣는 filling이며 그 방법은 다음과 같다. 네모는 상변의 좌측과 하변의 우측 좌표를 받아 들여 TMS34010 GSP가 제공하는 FILL 인스트럭션을 이용하여 filling한다. 다각형은 스캔라인 알고리듬을 이용하여 filling하며, 스캔라인 알고리듬은 y축의 값에 따라 변하는 스캔 라인을 이용하여 그림을 그린다.[1] 원과 타원의 solid filling은 동일한 방법으로 이루어지며 간단하게 구현할 수 있다. 즉, 원점을 중심으로 가정하여 원

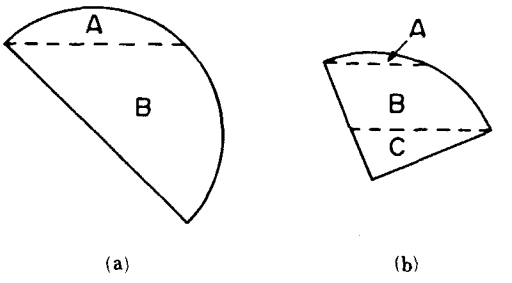


그림 2. Pie, Chord의 filling
Fig. 2. Filling of Pie and Chord.

이나 타원을 그리는 알고리듬을 적용하여 $x>0$ 인부분의 점 (x, y)을 구하고 이 점과 y축에 대칭인 점 $(-x, y)$를 구한다. 이들 점에 중점의 값을 더한후 두 점을 잇는 직선을 그려주면 원하는 중심을 갖는 원이나 타원의 solid filling이 된다.

현, 호의 solid filling은 원, 타원을 filling하는 방법과 다각형을 filling 하는 방법을 이용하여 스캔 라인을 따라서 fill한다. 즉, 그림 2 의 (a), (b)의 A와 같이 스캔 라인의 양끝이 모두 원이나 타원의 일부이면 그 스캔 라인은 원이나 타원을 filling하는 방법을 이용하고, 그림 2 의 (a), (b)의 B와 같이 스캔 라인의 한쪽은 원이나 타원의 일부와 만나고 다른 한쪽은 직선과 만나면 이 직선과 스캔 라인과의 교점을 구하여 두 점을 연결하는 직선을 그어 filling 한다. Pie의 경우에는 그림 2 의(c)의 C와 같이 스캔 라인의 양 끝이 모두 직선인 경우가 존재하며 이때는 양끝이 모두 직선이 되는 순간의 스캔 라인과 양 직선이 만나는 두 점과 중심을 잇는 삼각형을 filling 하는 다각형 filling 루틴을 불러 filling 한다. 또다른 filling 형태인 Pattern filling은 프리미티브의 내부를 사용자가 지정한 모양으로 채워주는 filling 방법으로서 boolean pixel operation과 solid filling 루틴을 이용한다. Pattern filling은 직사각형인 경우에만 가능 한pixel block transfer를 이용하므로 직사각형이 아닌 경우에는 그 도형을 포함하는 MBB에 원하는 도형의 pattern fill된 도형을 화면 밖의 메모리에 그린다. 이 MBB와 그리고자 하는 화면의 정보와 boolean operation을 하면 pattern filling이 가능하다.

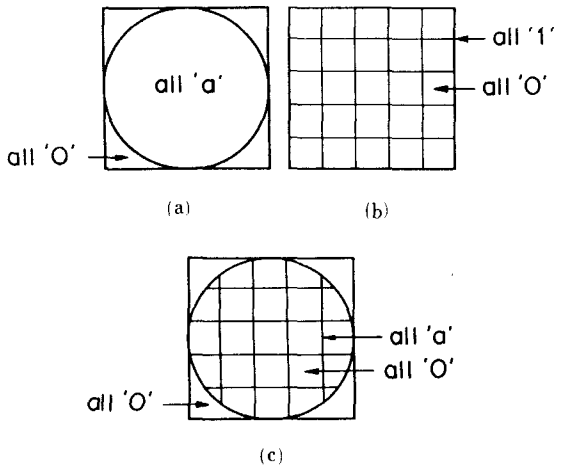


그림 3. Pattern-fill된 도형의 작성
Fig. 3. Making method of pattern-filled primitive.

원하는 도형의 pattern fill된 도형을 그리는 방법은 그림 3 과 같다. 이때 'a'는 그리고자 하는 도형의 색이고, '1'은 색을 표시하는 모든 비트의 값이 1인 경우(15번색)이고 '0'은 모든 비트의 값이 0인 경우(0번 색)이다. 먼저 그리고자 하는 도형의 MBB를 바탕색으로 결정한 후 원하는 색으로 solid fill된 도형을 그린다. 이 도형이 그림 3 의 (a)이며, 이 도형과 미리 지정된 pattern인 그림 3 의 (b)를 logical AND operation을 하면 그림 3 의 (c)와 같은 pattern fill된 도형을 구할 수 있다. 이렇게 구한 fill된 도형을 화면에 그리는 과정은 그림4와 같다. 먼저 화면 밖의 메모리에 모든 비트가 1인 도형의 MBB를 설정한다. 여기에 바탕색으로 solid fill된 도형을 그리면 그림 4 의 (a)와 같은 도형이 된다. 이 도형과 그리고자 하는 부분의 본래의 영상인 그림 4 의 (b)의 점선 내부를 logical AND 하여 (c)와 같은 새로운 도형을 만든 후 이 새로운 영상과 pattern fill된 도형인 그림 3 의 (c)를 logial OR를 하여 화면상의 그리고자 하는 위치에 그리면 그림 4 의 (d)와 같은 원하는pattern fill이 된다.

이런 과정을 위해서는 화면의 프레임 버퍼 이외의 메모리가 필요하며, 이때 필요한 메모리를 줄이기 위해 도형이 큰 경우에는 도형을 나누어서 각각을 프로세싱 할 수 있도록 했다.

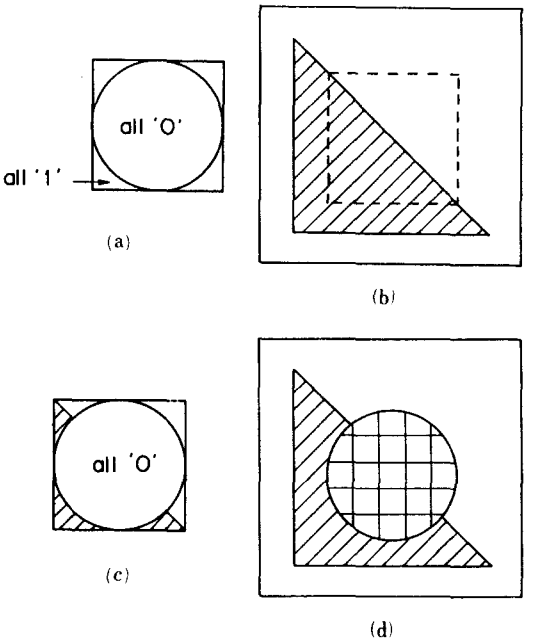


그림 4. Fill된 도형을 그리는 방법
Fig. 4. Drawing method of filled primitive.

3. Down Loading

그래픽스 보드 "K"는 에디터와 컴파일러를 제공하지 못하므로, 사용자가 앞에서 설명한 그래픽스 패키지를 이용하여 프로그램을 작성하는 일은 IBM PC/AT 상에서 수행하고 컴파일한다. 이렇게 만들어진 수행 파일은 IBM PC/AT상에 저장되나 이 파일은 TMS34010 GSP의 명령어로 구성되어 있으므로 IBM PC/AT에서 수행할 수 없고 TMS34010 GSP를 프로세서로 갖는 보드 "K"에서 수행할 수 있다. 따라서 이 프로그램을 수행하기 위해서 IBM PC/AT에서 보드 "K"로 수행 파일을 전송하는 다운로딩 프로그램이 필요하다. 따라서, 수행 파일의 이름을 받아들여 이 파일을 다운 로딩 시키고 수행시키는 프로그램인 DOWN을 작성하였다.

Ⅲ. IPCHE(3-D Computer Graphics Routine)

오늘날의 컴퓨터 그래픽스는 현실감을 지향하는 소프트웨어 분야와 실시간을 지향하는 하드웨어 분야의 두 가지 방향으로 활발한 연구가 진행되고 있다. 본 논문은 이들중 현실감을 추구하는 방향으로, 물체의 모양을 작은 다각형의 조합으로 이루어진 polyhedron으로 근사화시켜 표현하는 object-file과 관찰자의 위치와 광원의 위치를 갖는 user-file을 입력으로 하여 원근효과와 보이지 않는 면의 제거, 명암등의 효과를 갖는 물체를 구하고 이렇게 구해진 물체를 보드 "K"에 그려 준다. 이때 물체는 스캔 라인 방법을 이용하여 보드 "K"에 그려주며, 명암은 16-level의 gray color로 나타내었는데 이는 보드 "K"가 동시에 사용할 수 있는 색이 최대 16개 이기 때문이다.

컴퓨터에 의해 처리되는 3차원 물체는 작은 다각형으로 구성된 polyhedron으로 근사화 시켰으며 근사화 과정에서 생긴 문제점은 구나 도자기 같은 구형 물체의 표현이었다. 구와 같은 물체는 많은 사각형으로 나누어 근사화 시켰으나 사각형의 수가 너무많으면 데이타의 양이 많아져 계산 시간이 많이 걸리고 메모리의 부족으로 물체의 일부가 사라져 버린다. 또한 사각형의 수가 적으면 빠른 시간에 계산은 끝나지만 연속된 면에서도 큰 명암의 차가 생기고 구형의 물체가 아니라 각진 물체로 보이게 된다. 이러한 문제는 인접한 사각형의 명암의 차가 크지 않도록 사각형의 크기를 결정하여 사각형의 수를 적절하게 조절하면 되지만 완전하게 제거하기는 어려우며, 곡률 반경이 큰 경우에는 적절한 수의 사각형으로 나누기 어려우므로 연속된 사각형에서도 명암의 차가 생겨 각 사각형의 경계가 보이게 된다. 이는 gray level의 수를 늘이면 제거할 수 있으나 16-level로 제한된 보드 "K"에서는 완전하게 제거할 수 없었다. 물체를 표현할때 한번 정의된 물체는 다른 물체를 정의할 때 불러 사용할 수 있도록 하여 계층구조를 갖게 했다. 임의의 물체를 정의하는 과정은 다음과 같으며, 여기에서 V_{xxx}의 xxx는 물체를 형성하는 각 꼭지점의 분류번호이며 P의 color는 다각형의 색이며 $V_m, \cdots, V_n$은 다각형을 형성하는 꼭지점들의 번호이다. Call은 앞에서 정의한 물체를 불러 사용하는 경우로써 이들은 미리정의된 물체를 이동하거나, 회전, 미러링, 스케일링등을 하여 이용할 수 있다.

```
define object1;
Vxxx X YZ;/* Vertex list */
    ⋮
P color Vm, ···, Vn;/* Polygon list */
    ⋮
Call predefined-object orientation;/* Call object*/
    ⋮
define end;
```

평행한 2개의 직선이 눈에서 멀어지는 방향으로 놓여 있다면, 이 두개의 직선은 평행하게 보이지 않으며 결국 이 두직선은 소실점이라 불리는 한 점에서 만나는 것처럼 보인다. 물체를 그릴때는 이와 같은 효과를 갖도록 해야 실제의 모습과 비슷한 형태가 된다. 즉, 원근 효과가 실제의 모습을 표현하는 중요한 요소중 한가지가 된다. 본 논문에서는 z축에 소실점을 갖도록 원근 효과를 넣었다. 원근 효과를 갖는 물체로 구성하는 프로젝션 과정은 매트릭스를 이용하였는데, 그 이유는 프로젝션 과정은 여러 단계를 거쳐야 하며 매트릭스는 프로젝션 과정을 전부 포함하고 있으므로 한번의 계산만으로 여러점들을 손쉽게 프로젝션 시킬 수 있기 때문이다.

자연계에서 불투명한 물질은 빛의 투과를 막기 때문에 이 물질의 뒤에 놓여진 면은 눈에 보이지 않는다. 그러나 컴퓨터를 이용하여 물체를 그릴때는, 보이지 않는 면을 포함한 모든 면이 화면상에 나타난다. 따라서 실제와 같은 모양으로 나타내기 위해서는 보이지 않는 선이나 면을 제거하는 부분이 존재해야 한다. 본 논문에서는 감추어진 면을 제거하는 방법으로 스캔라인 알고리듬을 이용하였다.[2] 이 알고리듬중 곱셈과 나눗셈으로 구하여지는 interpolation 과정을 한번의 나눗셈으로 기울기를 구한후 덧셈만으로 interpolation을 수행하여 계산속도의 증가를 실현하였다.

3차원 물체의 raster scan 영상의 현실감은 물체 표면의 명암에 의해서 상당한 영향을 받으며, 명암은 실제의 자연계에 존재하는 빛과 면의 모든 행동

에 대해 정확하게 구하자면 많은 시간이 걸리므로 실제 행동에 근사화 시켜 구한다. 물체 표면의 명암을 구하는데 크게 영향을 주는 2가지 요소는 물체 표면의 특성과 표면에 비쳐지는 광원의 특성이다. 물체 표면의 특성에 의해 결정되는 것은 물체의 색과 재질감인데 물체의 색은 물체가 반사하는 색의 종류에 의해서 결정되고 재질감은 물체가 빛을 반사하는 정도에 의해 결정된다. 즉, 반사하는 빛의 색이 표면의 색이되며 반사하는 양이나 투과되는 양에 의해 거울이나 유리, 혹은 어떤 불투명한 물질등 재질감을 결정하게 된다. 면의 명암을 결정하는 광원들의 특성으로는 광원의 색과 광원의 세기등이 있다.

이러한 요소들에 의해 결정되는 물체 표면의 명암은 그 점에서의 diffuse reflection에 의한 명암과 specular reflection에 의한 명암의 합으로 구해진다. Diffuse reflection에 의한 명암인 I_d와 specular reflection에 의한 명암인 I_s는 다음과 같다.[1]

$$I_d = I_P K_d (L \cdot N)$$
$$I_s = I_P K_s (R \cdot V)^n$$

여기서, I_P는 광원의 광도이고, K_d는 분산계수, K_s는 윤택계수, n은 윤택도이며, L은 광원을 향하는 단위 벡터, N은 표면의 단위벡터, R은 반사광의 단위벡터, V는 관찰점을 향하는 단위 벡터이다.

물체를 구성하는 다각형내의 모든 화소의 명암을 구하기 위해서는 다각형에 존재하는 모든 꼭지점들의 명암을 먼저 구하고 linear interpolation 방법에 의해 구하는 Gouraud의 shading 방식과 표면의 수직벡터를 linear interpolation에 의해 구한 후 각 화소의 명암을 구하는 Phong의 shading 방식이 있는데, 이 논문에서는 Gouraud의 방법을 이용 하였다.[1]

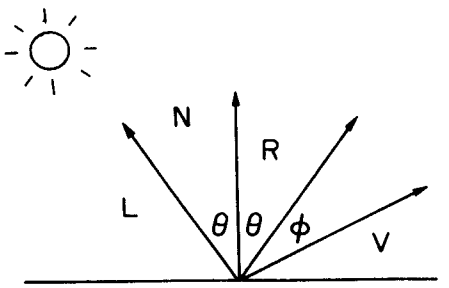


그림 5. 광원에 의한 intensity
Fig. 5. Intensity made by light source.

Ⅳ. GRIM (GRaphics Input and Modification)

컴퓨터 그래픽스의 한 응용분야로 마우스를 이용하여 대화식으로 회로를 설계하거나 2차원의 그림을 그리는 것이 있는데, 이 경우에는 컴퓨터가 프로세싱하는 시간은 비교적 적으며 대부분의 시간이 사용자와 컴퓨터가 정보를 교환하는데 사용된다. 즉, 데이타를 만드는 시간 보다는 어떤 기능을 원하는가를 받아 들이거나 그 기능을 수행하는데 필요한 데이타를 받아 들이는데 대부분의 시간을 소비하게 된다. 또한 결과가 순간 순간 그림으로 그려지므로 잘못된 부분을 쉽게 찾을 수 있고 고치기도 쉽다. 본 장에서는 2차원 그래픽 에디터인 GRIM에 대하여 설명하겠다. GRIM은 CGI-K를 이용하여 구현 하였으며 그림을 그리는 영역을 넓히고 필요한 순간에만 필요한 메뉴를 선택하게 하기위해 pop up 메뉴 방식을 도입 하였다.

GRIM은 사용자가 지정한 윈도우의 내부에 존재하는 부분만을 그려주는 크리핑 부분과 CGI-K에서 제공하는 그래픽 프리미티브들을 원하는 위치, 원하는 색, 원하는 형태를 마우스를 이용하여 받아 들여 그려주는 insert 부분, 그려진 프리미티브를 제거하는 delete 부분, 프리미티브들을 회전시키거나 이동시켜 프리미티브의 형태를 바꾸어 주는 update 부분, 현재의 데이타 구조에 포함되어 있는 모든 프리미티브들을 지정한 형태로 다시 그려주는 redraw 부분, 모든 일을 수행한 후 GRIM에서 빠져나가 다른 일을 하도록 해주는 Exit 등으로 구성되어 있다.

Insert에 의해 만들어진 프리미티브들을 제거하는 delete는 지정한 점을 포함하는 모든 프리미티브들을 제거하는 point delete와 지정된 영역 내부에 포함되어 있는 프리미티브들을 제거하는 area delete가 있다. Update는 insert에 의해 만들어진 프리미티브 들의 모양이나 위치를 바꾸어 주는 루틴으로 기준점을 중심으로 임의의 점으로 이동하는 move, 임의의 점을 중심으로 주어진 각도만큼 회전하는 rotate, 주어진 축이나 점을 중심으로 mirroring하는 mirror, 특정한 축의 값이나 모든 축의 값을 주어진 비율만큼 변화시키는 scale등으로 구성되어 있다.

프리미티브를 이동시키는 기준점을 $A(x_1, y_1)$라 하고 이동시키고자 하는 점을 $B(x_2, y_2)$라 할때 다음의 식에 의해 프리미티브가 이동하게 된다.

$$x = x + x_2 - x_1, \quad y = y + y_2 - y_1$$

회전의 경우 기준점을 $A(x_1, y_1)$라 하고 회전각을 결정하는 좌표를 $B(x_2, y_2)$라 하면 그림 6의 (a)와 같이 기준점을 원점으로 가정하여 회전각만큼 회전한다음

본래의 좌표로 바꾸어 주면 되므로 다음의 식에 의해 프리미티브가 회전된다.

$$x=(x-x_1)\cos\left(\tan^{-1}\frac{y_2-y_1}{x_2-x_1}\right)+(y-y_1)\sin\left(\tan^{-1}\frac{y_2-y_1}{x_2-x_1}\right)+x_1$$

$$y=-(x-x_1)\sin\left(\tan^{-1}\frac{y_2-y_1}{x_2-x_1}\right)+(y-y_1)\cos\left(\tan^{-1}\frac{y_2-y_1}{x_2-x_1}\right)+y_1$$

Mirror의 경우 기준이 되는 축의 값을 반대로 하면 미러링이 된다. 따라서, $A(x_1, y_1)$점을 기준으로 X축에 미러링한 경우의 새점의 좌표는 그림 6의 (b)의 경우이므로

$$x=-(x-x_1)+x_1=-x+2x_1,\ y=y$$

이 되며, A점을 기준으로 mirroring하면 다음과 같이 된다.

$$x=-x+2x_1,\ y=-y+2y_1$$

스캐일링은 특정한 축의 값이나 전체의 값을 지정한 비율만큼 변화 시키는 루틴으로 기준이 되는 길이와 스캐일링하고자 하는 비율 만큼의 길이를 마우스를 이용하여 입력하여 스캐일링한다.

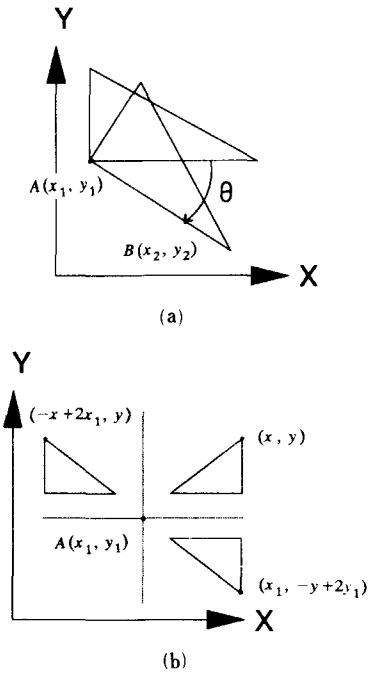


그림 6. 회전 및 미러링
Fig. 6. Rotation and Mirroring.

V. 결　　과

CGI-K에 포함되는 그래픽스 프리미티브들을 그리는 속도를 IBM PC/AT와 인터페이스시켜 사용하는 그래픽스 보드인 EGA에서 동작하도록 작성한 CGI와 비교한 경우를 표 1에 나타냈다. 표 1은 CGI-K를 이용하여 보드 "K"에 프리미티브를 그리는 속도와 EGA 보드상에서 프리미티브를 그리는 속도를 msec 단위로 나타낸 것이다. 여기서 비교의 대상으로 선택된 프리미티브는 100개의 화소를 지나는 직선과, 100×100의 화소로 구성되는 네모, 이 네모를 칠한 경우, 반경이 100개의 화소길이인 원, 이원을 칠한 경우 등이다. 표 1에서 보드 "K"의 프로세서가 제공하는 어셈블리 언어만을 이용하여 작성한 line과 box-fill은 EGA보다 150배에서 300배 정도 빨랐으며, 어셈블리 언어와 C언어로 작성되어 있으며 어셈블리언어를 이용하여 그림을 그리는 시간이 C언어를 이용하여 계산하는 시간보다 많은 네모, circle-fill의 경우에는 EGA보다 약 3배 빨랐다. 또한 C언어로 작성된 프로그램에 의해 계산되는 시간이 어셈블리 언어에 의해 그림을 그리는 시간보다 많이 소요되는 원의 경우에는 EGA 보드 보다 약 1.1배 정도의 시간이 걸렸다. 그러나, 모든 계산과정을 어셈블리 언어 로 작성한다면 모든 그래픽스 프리미티브들을 EGA 보드를 위한 CGI보다 매우 빠른 속도로 그릴 수 있다. 보드K에서 네모는 대각선의 두 점을 입력으로 하여 직선의 기울기등을 소프트웨어적으로 계산하여 4개의 직선을 그려 완성하는 반면, box fill의 경우에는 대각선 상의 두 좌표를 이용하여 하드웨어적으로 수행하기 때문에 box fill이 네모보다 빠른 시간에 그릴 수 있다.

보드 "K"와 보드 "K"에서 동작되는 CGI-K를 이용하여 작성한 IPCHE와 GRIM의 결과는 다음에 보여준 그림7, 8, 9, 10과 같다. 그림 7은 컵을 사각형으로 구성하여 입력한 파일을 이용하여 보이지 않는 면

표 1. Drawing 속도의 비교
Table 1. Compare of drawing speed.

	Board K	EGA
Line (100×1)	0.123	1.92
Box (100×100)	1.75	4.34
Box Fill (100×100)	0.65	18.51
Circle (r=100)	86.5	78.4
Circle Fill (r=100)	166	460

Unit : msec

그림 7. Wire-frame 형태로 그린 컵
Fig. 7. Cup of Wire-frame type.

그림 8. 컵
Fig. 8. Cup.

그림 9. 도우넛
Fig. 9. Doughnut.

을 제거하고 원근 효과를 갖는 컵으로 만들어 wire-frame 형태로 그린 것이며, 그림 8은 보이지 않는 면을 제거하고 명암과 원근 효과를 넣어 구한 최종 결과를 보여 주며, 빛을 직접 받는 면은 밝은 색으로 그려지고 빛을 받지 못하는 면은 어둡게 그려져 있다. 그림은 두가지 방향에서 비춰지는 광원에 의한 명암 효과를 잘 보여 주고 있으나 사용할 수있는 색이 16가지로 제한되어 있으므로 둥근 물체가 각진

그림10. GRIM을 이용하여 그린 그림
Fig. 10. Figure using GRIM.

물체로 보이고 있다. 그림 9는 도우넛을 PC/AT에서 처리 가능한 최대수의 다각형으로 구성한 경우로 각진 물체로 보이는 정도가 그림 8의 경우보다 줄어들었다. PC/AT의 처리능력이 증대된다면, 물체를 더 많은 색과 더 많은 다각형으로 구성할 수 있어 실제와 거의 동일한 모양을 보일 것이다.

그림10은 GRIM을 이용하여 그림을 그린 예이다. GRIM에서는 IBM PC/AT와 보드"K"가 마우스를 구동시키기 위하여 특정한 메모리를 공유하여 프로세싱 하는데, 이 부분의 메모리를 IBM PC/AT와 보드 "K"가 동시에 억세스 하면 보드 "K"의 프로세서상의 메모리 arbiter가 조절한다. 그러나 이 과정이 반복됨에 따라 시스템이 불안정 해지는데 이를 제거해야 완전한 시스템이 될 것이다.

Ⅵ. 결 론

보드 "K"만으로는 손쉽게 그래픽 application 프로그램을 작성할 수 없으므로 직선, 원, text 등 기본적인 그림을 그려주는 그래픽 패키지와 보드 "K"의 기능을 조절하는 컨트롤 펑션인 CGI-K를 TI(texas instruments)에서 제공하는 C언어와 어셈블리 언어를 이용하여 작성 하였다. CGI-K에 포함되어 있는 pattern-fill, 현, 호등 여러가지 프리미티브들은 기존의 그래픽 패키지인 EGA 보드상에서 동작하는 CGI 보다 3배에서 300배까지 빠른 속도를 갖는다. 그래픽 application 프로그램을 TMS34010 GSP를 이용하여 제작한 보드 "K"와 CGI-K를 이용하여 작성 하였으며, IBM PC/AT 상에서 주어진 과정에 의해 데이타를 계산하고, 계산된 결과를 보드 "K"에 보여 주는

IPCHE에서는 중간 결과인 감추어진 면을 제거하고 원근 효과를 갖는 wire frame 형태로 그릴 수도 있고, 최종 결과인 명암까지 포함된 완전한 형태로 그릴 수 있도록 하였다. 이러한 IPCHE는 lattice C를 이용하여 구현하였다. 마우스를 이용하여 사용자와 컴퓨터가 대화 하면서 원, 타원, 다각형등 9가지의 기본 도형을 이용하여 원하는 그림을 그릴 수 있는 GRIM을 TI사에서 제공한 C언어를 이용하여 구현하였다.

參 考 文 獻

[1] William M. Newman and Robert F. Sproull, "Principles of interactive computer graphics," 2-nd-edition, Mcgraw Hill Book Company.

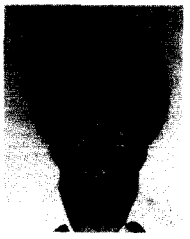
[2] J.D. Foley and A. Van Dam, "Fundamentals of interactive computer graphics," Addison Wesley.

[3] Jerry Van Aken and Mark Novak, "Curve drawing algorithms for raster displays," ACM Trans. on Graphics, vol. 4, no. 2 pp. 147-169, April 1985.

[4] 어길수, 최훈규, 경종민 "3차원 그래픽스 시뮬레이터 : SOFTGRA와 RACA," 전기, 전자공학 학술대회 논문집, 1987, pp. 1528-1531.

[5] 어길수, 최훈규, 경종민 "RACA : Raycasting에 의한 그래픽 시뮬레이터," 전기재료, 반도체및 CAD학술대회 논문집, 1987, pp. 152-154. *

著 者 紹 介



金 鎭 漢(正會員)

1963年 7月 6日生. 1986年 2月 고려대학교 전자공학과 졸업 학사 학위취득. 1988年 2月 한국과학기술원 전기 및 전자공학과 졸업 석사학위 취득. 1988年 3月~ 현재 한국과학기술원 전기 및 전자공학과 박사과정. 주관심분야는 Computer Architecture, Computer Graphics 등임.

慶 宗 旻 (正會員) 第25卷 第10號 參照

현재 한국과학기술원 전기 및 전자공학과 부교수.