Photorealistic image generation of molecular structure on PC screen using the ray-tracing technique

Sungwook Kim, Chang Woo Yoon, Byung Jin Mhin, Ho Soon Kim, and Kwang S. Kim

Department of Chemistry and Center for Biofunctional Molecules, Pohang Institute of Science and Technology, Pohang, Republic of Korea

A PC version of three-dimensional molecular graphics package has been developed to run under MS-DOS environment on IBM PC-compatible computers equipped with a VGA graphics board. The program consists of two parts: a menu-driven interactive system module in EGA mode, and a ray-tracing module in VGA mode. In the 256-color VGA mode, ray-tracing images are represented with a 4-color map, with 64 levels for each color: 32 levels of illuminance and 32 levels of saturation. Molecular structure can be analyzed along various directions with various light sources. Ray-tracing images are also represented in a 16-color EGA mode with the half-toning method, which can display 76 gray levels for each color. To obtain good photo-realistic images in an efficient way, we have used two light sources, with an intensity ratio of 7:3, which are located in front of the top right and bottom left corners of the screen.

Keywords: ray-tracing, molecular graphics, VGA, EGA, PC

INTRODUCTION

To obtain high-quality ray-tracing images, one may need expensive workstations with high-resolution devices displaying various colors. At present, PCs are widespread among the general public. Equipped with high-resolution video adaptors, a PC can display simultaneously 16 colors in EGA mode and 256 colors in VGA mode. So, ray-tracing techniques[1-5] may be utilized in PC. Fortunately, recent computer graphics techniques have found a new solution, fast ray tracing.[1] In this paper we discuss the three-dimensional (3D) representation of molecules on a PC using the fast ray-tracing technique. There are various methods, such as stereo pair, depth queuing, and perspective view, for displaying 3D information on a two-dimensional screen. However, the ray-tracing technique is known to yield the best images at the present time. To obtain a 3D representation of molecules using the ray-tracing method, the limitation on the number of colors available in the PC needs to be overcome to obtain realistic image representation. Thus, the half-toning method can be utilized. Further, the variations of the orientations and intensities of light sources are very important. So, we can vary the number of light sources, changing their orientations and intensities. As an illustration, we will apply our program to the photo-realistic image generation of the structure of carboxypeptidase-A (CPA)[6] with a substrate of glycyl-tyrosine. The enzyme of CPA is involved in peptide hydrolysis. In addition, the water hexamer structures will be represented in EGA mode.

HARDWARE

To generate a ray-tracing color image, a graphics color display is needed that can display various colors. The computer used is an IBM PC 386-compatible, with 20-MHz clock speed. The PC was equipped with an Intel 80387 math coprocessor and an AST-VGA graphics card (256 Kbytes VRAM). The graphics card has a maximum resolution of 800 × 600 pixels, but our multifrequency monitor can display only up to 640 × 480 pixels in 16 color mode, or 320 × 200 pixels in the 256-color mode.

PROGRAM

Our program was written in C, with all real variables in double precision, and was compiled with Microsoft C version 6.0 under MS-DOS 3.30. The program uses the EGA mode (640 × 350 pixels, 16 colors) for the menu-driven part, and the VGA mode (320 × 200 pixels, 256 colors) for the ray-tracing part. The main features of the program are as follows:

(1) Computation of the pixel images for the ray-tracing representation of complex molecular structures.
(2) User-friendly menu-driven program execution.
(3) Three-dimensional Euler angle rotation by single key stroke.

Color Plates for this article are on pages 227–228.

Address reprint requests to Prof. Kwang S. Kim at the Department of Chemistry, Pohang Institute of Science and Technology, P.O. Box 125, Pohang 790-600, Republic of Korea.
Received 19 September 1991; accepted 14 January 1992

(4) Constructive structured window system. (To make a user-friendly window system, a constructive recursion algorithm was applied. The window system is hierarchical, so that subwindows can be opened. To make the window system efficient in speed and size, the Graphical Kernel System (GKS) has been constructed.)
(5) Portability, flexibility and extendibility of the program, except for the screen output routines.

SHADING MODEL

A shading model is used to calculate the intensity of light that human eyes may feel from the real surfaces. The intensity calculation is based on the optical properties, positions, and orientations of the surfaces with respect to light sources. In ray tracing, point light sources are used. Light intensity on a surface comprises the ambient light, diffuse point-source illumination, and specular reflection. Specular reflection produces a spot of reflected light having the same color with the incident light. We used the Phong shading model,[2-3] which makes the light intensity vary smoothly on the surfaces. The light intensity is calculated by the following equation:

$$I = I_a k_a + I_p [k_d(\bar{N} \cdot \bar{L}) + k_s(\bar{R} \cdot \bar{V})^n]/(d + d_0)$$

where the notations are given as follows:

I_a—intensity of ambient light, k_a—coefficient of ambient reflection, I_p—point light source's intensity, k_d—coefficient of diffuse reflection, $\bar{N}$—surface normal vector, $\bar{L}$—unit vector from surface to light, k_s—coefficient of specular reflection, $\bar{R}$—direction of light reflection, $\bar{V}$—unit vector from surface to viewpoint, n—specular reflection exponent, d—distance from the light source to the object, and d_0—positive constant.

Here, the reflection coefficients k_a, k_d, and k_s are constants between 0 and 1.

The calculation of diffuse illumination due to a point light source is based on Lambert's cosine law, which states that the intensity of the reflected light depends on the angle of illumination, and that the brightness of an illuminated surface depends on the distance from the light source. Theoretically, the light intensity arriving at a surface is proportional to $1/d^2$, where d is the distance from the surface to the point source. But in reality, the intensity decreases less because any light source has a finite size. So the light source attenuation factor is commonly used as $1/(d + d_0)$ instead of $1/d^2$.

In the VGA mode of our PC, 256 colors can be displayed. Diffuse illumination depends on material's color, but it is independent of the object color when a white light source is used for the specular reflection. We used a 4-color map: white, red, yellow, and blue. Each color had 64 color levels: 32 levels of luminance and 32 levels of saturation.

Through our experiments, we set some constants: $n = 8$, $k_d = 0.53$, and $k_s = 0.53$. For large values of n, specularly reflected lights were focused, giving sharp highlights. But for small values of n, the reflected lights were broadly distributed, producing gentle fall-off. Large values of k_d and k_s increased the intensities of the diffuse and specular reflections, respectively. Small values of d gave object images having vivid contrasts with bright and dark sides, while large values of d gave dark images with little contrast. The value of d_0 was set to a positive constant so as to make the light-source attenuation factor, $1/(d + d_0)$, always less than 1.

REAL IMAGE

In ray tracing we used point light sources for the light intensity calculation. A ray-tracing image obtained by this method shows a sharp image that differs from the real scene. To represent the best 3D images, we investigated how the variation of the position and number of point light sources affects the screen image.

Color Plate 1 shows the ray-tracing images of the molecular structure according to the position and number of light sources. Here, the structure is a part of the active site of CPA[6] obtained by molecular dynamics simulation in our laboratory.[7] In Color Plate 1, one should note that multiple light sources allow the hidden part of the active site to be more clearly seen. Color Plate 1a has the light source in front of the center of the monitor, while Color Plate 1b has the light source in front of the top right corner of the monitor. Due to shadows cast by atoms in the molecule, Color Plate 1b provides a better 3D view. But some dark parts of the molecule give a disadvantage in analyzing the molecule. In Color Plate 1c, two light sources were used, with an intensity ratio of 7:3. It shows a stereoscopic and realistic image, like a photograph. Also, this image helps us analyze the difference between the far and near parts of the molecule and the structural arrangement of atoms and spaces in the molecule.

When three light sources were used, the 3D view was similar to the case when two light sources were used. The greater the number of light sources, the more computing time is needed. So we feel that using two light sources is the most suitable choice for the ray-tracing method, particularly for a PC. To obtain more realistic images, the location and intensity ratio of the two light sources were varied. A conspicuous change was not found. But, from our experience, the recommended intensity ratio of the two light sources is 7:3. Two lines drawn from the origin to the two light sources are in the same plane, while the recommended angle made by the two lines is 45°. A ray-tracing image of CPA is shown in Color Plate 2 using two such light sources.

Synthesizing an image with a digital computer is very different from exposing a piece of film to a real scene. The differences are essential, although much of computer graphics is directed to making the differences as small as possible. To reduce such a problem, there are techniques to avoid or reduce aliasing, known collectively as anti-aliasing techniques. We used the *adaptive supersampling* method.[4-5] In our program, this technique partially enhanced the picture resolution in the VGA mode.

In Color Plates 1 and 2, the effect of anti-aliasing can be seen at the boundaries of atoms. The effect is noticeable on boundaries having the same color, while it is negligible on boundaries having different colors. The negligible effect for the latter case is caused by the restriction of the number of usable colors. The overall effects of anti-aliasing in Color Plates 1 and 2 seem to be small not only because of the small number of available colors, but also because of the small number of pixels.

In our IBM PC 386, it took 6 hours to generate the image metafile of Color Plate 1 (~ 100 atoms) and 60 hours to generate that of Color Plate 2 (~ 3000 atoms). The time taken to generate the image metafile can be proportional to the number of atoms. But, utilizing the fact that the hidden atoms need not be taken into account for calculation, the time has been reduced down to the two-thirds power of the number of atoms. Using workstations, it took several minutes to obtain an image metafile that can be sent to a PC so as to be represented on a PC screen. Once the image metafile is generated, it takes only half a minute to represent the image on the screen, with no dependence on the number of atoms. The image metafile can be kept on hard disks for future usage.

HALF-TONE METHOD IN EGA MODE

In our graphics program, the interactive handling part uses the EGA mode, which uses 16 colors.[8] The palette is composed of 8 colors, each of which has two intensities. Thus, it is impossible to represent ray-tracing images only with colors that the EGA mode supports. We used the half-tone method[9] to obtain various shading levels for each color. The half-tone method uses spatial integration. If we view a very small area from a sufficiently large viewing distance, our eyes average fine details within the small area, and record only the overall intensity of the area.

In EGA mode, we can use four intensity levels for each color: dark, light, white, and black. If we use a 2 × 2 pattern with a total of four pixels, this allows us to display 13 intensities (4 × 3 + 1), as shown in Figure 1.

The structure of the water hexamer predicted by *ab initio* calculation[10] is given in Color Plate 3 as an example of a graphic output using the half-tone method in the EGA mode. In this case, we used a 5 × 5 pattern with a total of 25 pixels. This allowed us to display 76 intensities (25 × 3 + 1).

CONCLUSION

The ray-tracing method is the highest class of computer graphic techniques, since it generates photo-realistic pictures of solid models. Due to difficulties in writing a ray-tracing program, and expensive computing time needed to generate ray-tracing images, the technique was seldom used in the area of computer-aided molecular design (CAMD), except for the case of advertisement. Fortunately, our molecular graphics system somewhat solved the problem due to the limited number of colors in the PC, and provided some advantages in graphics display of molecular structures. Compared with other commercial CAMDs, our graphics program is characterized particularly by the following features:

(1) Once metafiles of molecules are generated, interactive handling of molecules provides user friendliness with a constructive window system.
(2) We can represent ray-tracing images in the EGA mode by implementing the half-toning method, which can display 76 gray levels for each color.
(3) To obtain the best 3D photo-realistic image, we chose to place two light sources with a light intensity ratio of 7:3 in front of the top right and the bottom left corners of the screen.

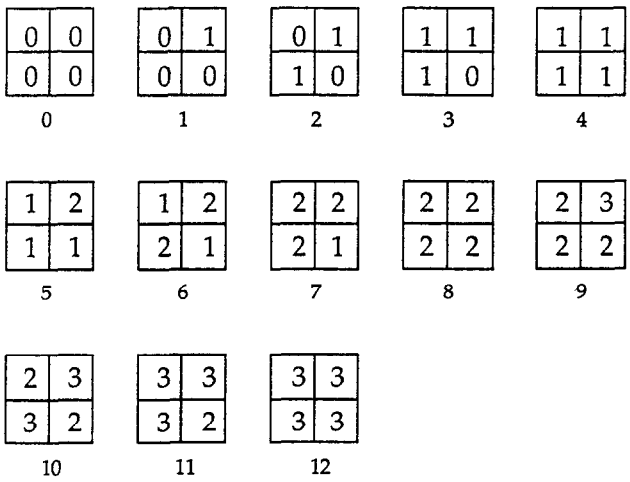

Figure 1. Intensity levels 0 through 12, obtained by the half-toning technique using 2 × 2 pixel grids of a four-level system.

REFERENCES

1 Arvo, J. and Kirk, D. Fast Ray Tracing by Ray Classification. *Comp. Graphics.* 1987, **21**(4), 55
2 Foley, J.D. and Van Dam, A. *Fundamentals of Interactive Computer Graphics.* Addison-Wesley, Reading, 1982, p. 575
3 Rogers, D.F. *Procedual elements for Computer Graphics.* McGraw-Hill, NY, 1985, p. 296
4 Glassner, A. Adaptive Precision in Texture Mapping. *Comp. Graphics.* 1986, **20**(4), 297
5 Turkowski, K. Anti-aliasing in Topological Color Spaces. *Comp. Graphics.* 1986, **20**(4), 307
6 Christianson, D.W. and Lipscomb, W.N. *Acc. Chem. Res.* 1989, **22**, 62
7 Kim, K.S. *Chem. Phys. Lett.* 1989, **156**, 261
8 Sutty, G. and Blair, S. *Programmer's Guide to the EGA/VGA.* Simon & Schuster, New York, 1988
9 Holladay, T.M. An Optimum Algorithm for Halftone Generation for Displays and Hard Copies. *SID Digest.* 1979, **10**, 102
10 Mhin, B.J., Kim, H.S., Kim, H.S., Yoon, C.W., and Kim, K.S. *Chem. Phys. Lett.* 1991, **176**, 41

APPENDIX

Code segments to determine intensity levels in the EGA mode

```
/*Gloval Variables & Constants*/
struct { double Criterion;
         double Part[3];
       }msr;
struct { int MaxX, MaxY;
              double NumOfCells;
              ......
       }cfg;
```

```
typedef short intensity__t;
.....
intensity__t b = 0;
intensity__t w = 255;
short Cellsize = 5;
double x, y;
.....
msr.Criterion = (w-b)/4.;
for(i=0; i<3; i++) msr.Part[i] = msr.Criterion*(i+1);
cfg.NumOfCells = (cfg.MaxX+1.)/Cellsize;
.....
#define Pai 3.141592653589
#define Alpha (cfg.NumOfCells*2.*Pai/cfg.MaxX)
#define Beta (cfg.NumOfCells*2.*Pai/cfg.MaxY)
#define Modul(x, y)
(intensity__t)(msr.Criterion*sin(Alpha*x)*sin(Beta*y))
/* Distribute intensity in a cell */
.....
/* A function displaying a pixel on the screen with given
intensity and color */
void EGA__PutDot(x, y, intensity, color)
double x, y;
intensity__t intensity;
short color;
{
double all;
.....
all = intensity + Modul(x, y);
.....
if(all<msr.Part[0]) setcolor(BLACK);
else if(all<msr.Part[1]) setcolor(lowcolor);
else if(all<msr.Part[2]) setcolor(highcolor);
else if(all<msr.Part[3]) setcolor(WHITE);
.....
}
.....
```

Code sgements to set the VGA color map

```
/* Gloval Variables & Constants */
.....
#define RGB(r, g, b)
((long)((long)((b)<<8|(g))<<8)|(long)(r))
/* | b | g | r | */
.....
long int Colors[256];   /* pointers for color map */
enum OFFSET {OFF0SET, OFF1SET=64,
OFF2SET=128, OFF3SET=192};
.....
/* A function to set color map for blue */
void saveblues(offset)
int offset;
{
int i, j;
for(i=offset, j=0; j<64; i++, j+=2)
  Colors[i] = RGB(0, 0, j);
for(;j<128; i++, j+=2);
  Colors[i] = RGB(j-64, j-64, 63);
}
.....
main()
{
double segment;
.....
saveblues(OFF0SET);
savereds(OFF1SET);
savebrowns(OFF2SET);
savewhites(OFF3SET);
__remapallpalette(Colors);
.....
segment = intensity / 256.0 * 63.0;
__setcolor(segment + OFF0SET);
.....
}
```