#1$2+3WELCOME TO THE

HELIOS

RADIOSITY RENDERER

Introduction

Radiosity is an advanced computer graphics technique that you can use to synthesize photorealistic images without ray tracing. **HELIOS** brings this new and exciting technology to your personal desktop computer.

Commands

Glossary

[1] main_index
[2] Help Index
[3] index:0005

#4$5+6**What is Radiosity?**

There are two approaches to generating photorealistic images -- digital pictures that are difficult to distinguish from real photographs -- in computer graphics. The first approach involves ray-tracing techniques; the second approach is *radiosity*.

Radiosity is in a sense the complement of ray tracing. Ray-tracing techniques excel in the rendition of point light sources, specular reflections, and refraction effects. Radiosity methods accurately model area light sources, diffuse reflections, color bleeding effects, and realistic shadows. Whether you choose ray tracing or radiosity will depend in part on what effects you consider to be more important in your images.

Radiosity has the advantage of view independence. Using ray-tracing techniques, the number of ray-surface intersection calculations can increase geometrically with the complexity of the scene. Change your point of view and you typically have to start from scratch to generate a new image. With radiosity, however, you only need to perform the lighting calculations once for a given environment. Once they have been completed, you can quickly render a view of the environment as seen from any position and orientation.

Folklore has it that you need a high-powered graphics workstation or even a supercomputer to experiment with radiosity rendering techniques. Not so! All you need is an IBM PC-AT clone with an 80386 CPU and a math coprocessor, 4 megabytes of RAM, and a 256-color video display adapter.

Of course, a faster machine is always better. Using an off-the-shelf desktop computer equipped with a 66 MHz 80486 CPU, you can create photorealistic images in ... are you ready for this? ... less than a minute. Compared to the hours typically needed to ray trace photorealistic images, there is something to be said for radiosity techniques.

HELIOS was written to demonstrate the power of radiosity-based rendering. It is a fully functional program for Microsoft Windows, and was designed for those interested in experimenting with advanced computer graphics techniques.

Radiosity Explained

So, what is radiosity? Well, imagine a room that has a chair sitting on the floor, a picture hanging on the wall, a floating apple, and two light fixtures suspended from the ceiling. In addition, imagine that each surface is divided into a mesh of elements.

To determine exactly how the room is illuminated, we begin with the light fixtures. They are emitting light, and so we find those surface elements in the room that are visible to them and calculate how much of the light is transferred to each element. (This is easier said than done, but we can ignore the mathematical details here.)

Some elements will receive more light than others, and different surfaces will reflect different amount of light. Still, it is clear that each element will absorb some of the light, and so that the total amount of light reflected back into the room will be less than that emitted by the light fixtures.

This is the first step of our radiosity calculations. We now find the element that is reflecting the most amount of light and repeat the process. In other words, we consider the element to be a secondary light source and calculate how much of its reflected light is transferred to those other elements in the room that are visible to it.

We can repeat this process, one step at a time, until the amount of light remaining in the room is vanishingly small in comparison to the light originally emitted by the light fixture. We then say that our radiosity calculations have converged to a solution. By adding up all the light that has been reflected from each surface element, we can calculate the luminance (or, colloquially speaking, the *brightness*) of each one.

We know the geometry of each element in the room -- in computer graphics parlance, it is a three-

[4] HELP_WHAT_IS_RADIOSITY
[5] What is Radiosity?
[6] commands:010

dimensional polygon. If we know its luminance, we can use a three-dimensional graphics package to directly render a photorealistic image of the room from any viewpoint.

That's all there is to it! Radiosity explained in 500 words or less. To stop imagining and start doing, see ~~Using HELIOS~~. The room described above can be found in ROOM.WLD.

Learning More About Radiosity

There is of course much more that can be said about radiosity. Try **HELIOS** and see what you think of it. If you want to learn more, take a look at ~~Read This Book!~~

[7$8+9]Using HELIOS

This release of **HELIOS** is a demonstration program that includes one sample environment. This environment consists of a room with a chair on the floor, a floating apple and an oblate sphere, a picture on the wall, and two fluorescent lighting fixtures that illuminate the room.

While this room may not be the most exciting of environments, it clearly demonstrates radiosity's ability to model area light sources, diffuse reflections, color bleeding effects and realistic shadows.

The associated environment and entity files are:

ROOM.WLD
APPLE.ENT
CEILING.ENT
CHAIR.ENT
FLOOR.ENT
LIGHT_1.ENT
LIGHT_2.ENT
PICTURE.ENT
SPHERE.ENT
WALL.ENT

All of these files should be stored in the same directory.

To display a view of the room, we can:

1. Choose *File* from the menu bar.
2. Choose the *Open*... menu item to display the *Open* dialog box.
3. Select the ROOM.WLD file.
4. Press <Enter> or select the *OK* button.

An *Environment Statistics* dialog box will appear with an enumeration of the instances, surfaces, patches, elements and vertices in the environment. Press <Enter> or select the *OK* button.

If the entity files are not in the same directory as ROOM.WLD, an error message will appear in a dialog box. Rather than exiting **HELIOS**, we can:

1. Choose *File* from the menu bar.
2. Choose the *Entities*... menu item to display the *Entities Directory* dialog box.
3. Enter the correct file path in the *Entities File Path* edit control and repeat the previous three steps to select the ROOM.WLD file again.
4. Press <Enter> or select the *OK* button.

When the environment file is successfully opened, **HELIOS** will automatically display a wireframe image of the room from a default viewpoint. This image will automatically resize itself whenever the **HELIOS** display window size is changed.

You can change this default viewpoint by clicking on the right mouse button to display the interactive mode menu. There are six modes to choose from: *Dolly*, *Orbit*, *Pan*, *Rotate*, *Tilt*, and *Zoom*. Only one of these modes can be activated at a time. The mouse is enabled whenever a mode is active and the left mouse button is depressed. The wireframe or bitmap image will be redrawn when the mouse button is released.

These modes behave as follows:

1. **Dolly** - the camera eye position and focus position are shifted forward or backward along the camera's line of sight (i.e., the view direction vector). Moving the mouse upwards on the screen moves the eye and focus positions forward; moving the mouse downwards moves them backward.

[7] HELP_USING_HELIOS
[8] Using HELIOS
[9] commands:010

2. **Orbit** - the camera eye position is rotated about the focus position, with the camera's line of sight remaining fixed on the focus position. Moving the mouse in a given direction on the screen moves the eye position in the same direction.

3. **Pan** - the camera eye position and focus position are shifted horizontally and vertically perpendicular to the camera's line of sight. Moving the mouse in a given direction on the screen moves the eye and focus positions in the same direction.

4. **Rotate** - the camera's view direction and focus position are rotated horizontally and vertically about the eye position. Moving the mouse in a given direction on the screen rotates the view direction vector and moves the focus position in the same direction.

5. **Tilt** - the camera's view-up vector (and hence the viewed image) is tilted relative to its default vertical orientation. Moving the mouse in a given direction on the screen tilts the view-up vector in the same direction.

6. **Zoom** - the camera's view distance is decreased or increased, resulting in a larger or smaller field of view. Moving the mouse upwards on the screen decreases the view distance; moving the mouse downwards increases the view distance. (This is equivalent to adjusting the focal length of a zoom lens on a camera.)

If you get lost at any time and cannot see the room, you can return to the default view by:

1. Choose *Camera* from the menu bar.
2. Choose the *Default View* menu item to display the *Camera Parameters* dialog box.
3. Enter 2 in the *View Distance* edit control.
4. Press <Enter> or select the *OK* button.

You can also set the camera's view distance (field of view) and tilt angle with the help of the *Camera Parameters* dialog box. For example:

1. Choose *Camera* from the menu bar.
2. Choose the *Set Camera* menu item to display the *Camera Parameters* dialog box.
3. Enter 2 in the *View Distance* edit control.
4. Press <Enter> or select the *OK* button.

This sets the camera view distance at 2.0 units, giving a field of view roughly equivalent to a 35-mm lens on a 35-mm camera. The default *Window Dimensions* values tells **HELIOS** to display the image as a horizontally oriented bitmap of 640 x 480 pixels. We can change this to whatever we want, from a minimum of 32 to a maximum of 1,024 pixels.

The synthetic camera's eye and focus positions can be similarly set with the help of the View Parameters dialog box. For example:

1. Choose *Camera* from the menu bar.
2. Choose the *Set View...* menu item to display the *View Parameters* dialog box.
3. Enter 2.6 in the *Eye Position X-Axis* edit control.
4. Enter 1.4 in the *Eye Position Y-Axis* edit control.
5. Enter 1.3 in the *Eye Position Z-Axis* edit control.
6. Enter 0.5 in the *Focus Position X-Axis* edit control.
7. Enter 0.8 in the *Focus Position Y-Axis* edit control.
8. Enter 0.5 in the *Focus Position Z-Axis* edit control.
9. Press <Enter> or select the *OK* button.

We can go back and change any of the previous entries to change the camera parameters; the wireframe image will update itself accordingly. (Selecting the *Update* button in the *Camera Parameters* or *View Parameters* dialog box will update the wireframe image without closing the dialog box. The eye position, focus position, view distance, and camera tilt fields will also be updated whenever you use the mouse to change the view.)

To display a full-color shaded (but not photorealistic) bitmap image, we can:

1. Choose *Render* from the menu bar.

2. Choose the *Shaded* menu item.

A dialog box titled *Rendering Calculations* with the message "Polygon shading in progress..." will appear. If the computer is capable of displaying only 256 colors, it will be followed by message "Color quantization in progress...” to indicate that color quantization is being performed.

It will take a few seconds or more before the image is displayed, depending on the CPU speed. Increasing the bitmap size in either direction increases the display calculation time accordingly. The surface colors are those specified in the entity files, without any lighting taken into account.

To generate a photorealistic bitmap image, we must:

1. Choose *Render* from the menu bar.
2. Choose the *Rendering* menu item.

A dialog box titled *Radiosity Calculations* will appear to indicate the current and maximum number of steps, the current and minimum convergence, and the elapsed time. You can press <Enter> or select the *Stop* button at any time to stop the radiosity calculations. The default number of steps is 100, which will require approximately 68 seconds to complete on a 66 MHz 80486 CPU machine.

The *Rendering Calculations* dialog box as described above will appear when the radiosity calculations are completed. This will be followed by a *Convergence Statistics* dialog box that displays the number of steps, convergence and elapsed time for the radiosity and rendering calculations.

The photorealistic image will appear behind the *Convergence Statistics* dialog box. Press <Enter> or select the *OK* button to close the dialog box. (If the computer is only capable of displaying 256 colors, the colors may appear to be incorrect. However, they will be displayed correctly once the dialog box is closed.)

Having performed the radiosity calculations for the environment, we do not need to choose *Rendering* again until we open a new environment file. That is, we can change the camera parameters to look at the environment from whatever direction we choose, using the *Wireframe* display to provide a quick preview. Once we have what we want, we can:

1. Choose *Render* from the menu bar.
2. Choose the *Redisplay* menu item.

to redisplay a photorealistic image of the room.

We can redisplay the image in gray scale or pseudocolor by:

1. Choose *Options* from the menu bar.
2. Choose the *Set Display*... menu item to display the *Display Parameters* dialog box.
3. Select either the *Grayscale* or *Pseudocolor* radio button from the *Color Display* group.
4. Press <Enter> or select the *OK* button.

(You can also select the *Update* button if you do not want the dialog box to close after the image has been redisplayed.)

If the computer cannot display 16.7 million (24-bit) colors, there may be some noticeable color banding in the bitmap image. To alleviate this problem, try:

1. Choose *Options* from the menu bar.
2., Choose the *Set Display*... menu item to display the *Display Parameters* dialog box.
3. Check the *Enabled* check box in the *Color Reduction* group.
4. Press <Enter> or select the *OK* button.

By default, the radiosity calculations terminate after 100 steps. However, the Convergence Statistics dialog box indicates that there is still some 15 percent of the light emitted by the light fixture that has not been absorbed (as shown by the convergence value). A more accurate photorealistic image can be obtained by allowing the radiosity calculations to run until there is only 1/10 percent of the light remaining. To do this, we can:

1. Choose *Options* from the menu bar.

2. Choose the *Set Convergence*... menu item to display the *Convergence Parameters* dialog box.
3. Enter 1000 in the *Maximum Steps* edit control.
4. Press <Enter> or select the *OK* button.
5. Choose Render from the menu bar.
6. Choose the *Rendering*... menu item.

The difference between the images are subtle but important. Look in particular at the shadows and the diffuse color bleeding (i.e., reflections) between the boxes and the adjacent walls.

Finally, we can save the bitmap image to a Microsoft Paintbrush-compatible BMP graphics file by:

1. Choose *File* from the menu bar.
2. Choose the *Save As*... menu item to display the *Save As* dialog box.
3. Specify an appropriate directory and file name for the file.
4. Press <Enter> or select the *OK* button.

This completes our guided tour of **HELIOS**. There are some dialog box features that we have not examined, but the on-line help explains what they are for and how to use them.

#10$11+12**HELIOS File Formats**

First, a word of warning: the data file formats used by **HELIOS** are extremely rudimentary. They were not designed for compatibility with any existing graphics standard or CAD program.

Using these file formats requires: (a) a text editor; and (b) *lots* of patience. When you have to include the source code for an entire data file parser in a book devoted to radiosity rendering, your options are limited ... these file formats were *not* designed for ease of use.

The next version of **HELIOS** will feature AutoCAD DXF file input support, which means that the current data file formats will be abandoned. They are documented here only for those interested in experimenting with radiosity rendering.

There are two data file formats: one to describe individual entities and another to describe the transformations necessary to create instances of them in an environment. The descriptions of these file formats are complete but necessarily abbreviated. Full details are provided in the book, ***Radiosity: A Programmer's Perspective*** (Wiley, 1994).

Another word of warning: the file parser employed by **HELIOS** does very little error checking. If you open an invalid environment (*.WLD) or entity (*.ENT) file, **HELIOS** may crash rather than issue a diagnostic error message.

Entity File (*.ENT) Format Specification

Each entity is composed of a set of planar surfaces, with each surface having a specified reflectance (defined as an RGB color) and a specified emittance (also defined as an RGB color. Only light sources have non-zero emittance values.

Each surface is subdivided into an array (mesh) of one or more triangular and/or quadrilateral patches. Each patch is further subdivided into an array (mesh) of one or more elements.

The elements and patches are defined by their vertices. Because most vertices will be shared by two or more elements, they are specified separately. The elements and patches then specify their vertices by an array of indices into this array.

The reason for the hierarchy of surfaces, patches, and elements is simple. Imagine a light source that is illuminating a surface. At each step of our radiosity calculations, the light reflected and/or emitted by one element is transferred to all the other elements in the environment that are visible to it. However, in many cases, the amount of light transferred to a group of adjacent elements will be the same (proportional to their area). It makes sense, therefore, to transfer the light from an element to a patch, and then allocate that light among the elements of the patch in accordance with their relative areas. This is much faster than transferring the light to each individual element within the patch.

Manually dividing surfaces into patches and elements is something of an art. If the elements or patches are too small and numerous, the radiosity calculations will take too long to converge. If, on the other hand, the elements or patches are too large, the image will lack well-defined shadows. This is evident in the room image, where the shadows surrounding the objects do not appear fully realistic. In practice, the floor surface should be more finely divided into patches and elements, particularly where there are shadow edges. Unfortunately, it is difficult to know where the shadow edges will occur until you render an image of the environment. (A future release of **HELIOS** will overcome this problem by automatically dividing surfaces into a hierarchy of patches and elements.)

```
COMMENT Entity Data File
ENTITY entity_name
VERTEX
< x1 y1 z1 >
< x2 y2 z2 >
```

[10] HELP_FILE_FORMATS
[11] HELIOS File Formats
[12] commands:010

```
...
< xm ym zm >
END_VERT
SURFACE
[ rr1 rg1 rb1 ] [ er1 eg1 eb1 ]
[ rr2 rg2 rb2 ] [ er2 eg2 eb2 ]
...
[ rrn rgn rbn ] [ ern egn ebn ]
END_SURF
PATCH
s1 { v10 v11 v12 v13 }
s2 { v20 v21 v22 v23 }
...
sp { vp0 vp1 vp2 vp3 }
END_PATCH
ELEMENT
p1 { v10 v11 v12 v13 }
p2 { v20 v21 v22 v23 }
...
pp { vp0 vp1 vp2 vp3 }
END_ELEM
END_ENTITY
```

The syntactic rules for the entity file format specification are:

1. The data file consists of ASCII characters.
2. Each line must be terminated with an environment-specific "newline" character (typically <CR><LF> for MS-DOS and <CR> for UNIX systems).
3. The maximum length of a line is 256 characters, including the newline character(s).
4. Multiple whitespace (ASCII space and horizontal tab) characters between data values and separators are ignored.
5. Comment lines beginning with the keyword "COMMENT" are ignored.
6. The data file consists of one ENTITY section.
7. The ENTITY section header begins with the "ENTITY" keyword, followed on the same line by an optional entity_name character string that identifies the entity. Any printable ASCII character is permitted in the string.
8. The ENTITY section header is followed by a VERTEX subsection. It begins with the "VERTEX" keyword, followed on subsequent lines by a list of four or more vertex vectors. A maximum of 65,536 vertex vectors are allowed. Each vertex is implicitly assigned an index number according to its position in the list, beginning with zero. The "END_VERT" keyword terminates the subsection.
9. Each vertex vector consists of a '<' separator, followed by three floating point numbers representing the x, y and z values of the vertex co-ordinates respectively, followed by a '>' separator.
10. The VERTEX subsection is followed by a SURFACE subsection. It begins with the "SURFACE" keyword, followed on subsequent lines by a list of one or more RGB color vector pairs. The first vector of each pair represents the surface reflectance for the entity; the second vector represents the surface's initial surface spectral radiant exitance. A maximum of 65,536 surfaces are allowed. Each surface and its associated reflectance and initial exitance vectors are implicitly assigned an index number according to its position in the list, beginning with zero. The "END_SURF" keyword terminates the subsection.
11. Each reflectance vector consists of a '[' separator, followed by three floating point numbers representing the red, green and blue primary color values respectively, followed by a ']' separator. The color values must be in the range 0.0 to 1.0.
12. Each initial exitance vector consists of a '[' separator, followed by three floating point numbers representing the red, green and blue primary color values respectively, followed by a ']' separator. The color values must be equal to or greater than 0.0.

13. The SURFACE subsection is followed by a PATCH subsection. It begins with the keyword "PATCH", followed on subsequent lines by one or more patch identifiers. A maximum of 65,536 polygon identifiers are allowed. The "END_PATCH" keyword terminates the subsection.
14. Each patch identifier consists of an integer number indicating the index number of the surface to which the patch belongs, followed by a '{' separator, followed by four integer numbers indicating the indices of the four patch vertices v0, v1, v2 and v3 respectively, followed by a '}' separator. If the patch is a triangle, the third and fourth vertex indices must be identical.
15. The PATCH subsection is followed by an ELEMENT subsection. It begins with the keyword "ELEMENT", followed on subsequent lines by one or more element identifiers. A minimum of 65,536 element identifiers are allowed. The "END_ELEM" keyword terminates the subsection.
16. Each element identifier consists of an integer number indicating the index number of the patch to which the element belongs, followed by a '{' separator, followed by four integer numbers indicating the indices of the four element vertices v0, v1, v2 and v3 respectively, followed by a '}' separator. If the element is a triangle, the third and fourth vertex indices must be identical.
17. The ELEMENT subsection is followed by an "END_ENTITY" keyword, which terminates the file.

To clarify the above, here's an example of a small entity data file that describes a colored cube:

```
ENTITY colored cube
VERTEX
< 0.0 0.0 0.0 >
< 1.0 0.0 0.0 >
< 1.0 0.0 1.0 >
< 0.0 0.0 1.0 >
< 1.0 0.0 0.0 >
< 1.0 1.0 0.0 >
< 1.0 1.0 1.0 >
< 1.0 0.0 1.0 >
< 1.0 1.0 0.0 >
< 0.0 1.0 0.0 >
< 0.0 1.0 1.0 >
< 1.0 1.0 1.0 >
< 0.0 1.0 0.0 >
< 0.0 0.0 0.0 >
< 0.0 0.0 1.0 >
< 0.0 1.0 1.0 >
< 0.0 0.0 0.0 >
< 0.0 1.0 0.0 >
< 1.0 1.0 0.0 >
< 1.0 0.0 0.0 >
< 0.0 0.0 1.0 >
< 1.0 0.0 1.0 >
< 1.0 1.0 1.0 >
< 0.0 1.0 1.0 >
END_VERT
SURFACE
[ 0.0 0.0 1.0 ] [ 0.0 0.0 0.0 ]
[ 1.0 1.0 0.0 ] [ 0.0 0.0 0.0 ]
[ 1.0 1.0 1.0 ] [ 0.0 0.0 0.0 ]
[ 0.0 1.0 1.0 ] [ 0.0 0.0 0.0 ]
[ 1.0 0.0 0.0 ] [ 0.0 0.0 0.0 ]
[ 0.0 1.0 0.0 ] [ 0.0 0.0 0.0 ]
END_SURF
PATCH
0 {  0  1  2  3 }
1 {  4  5  6  7 }
2 {  8  9 10 11 }
```

```
3 { 12 13 14 15 }
4 { 16 17 18 19 }
5 { 20 21 22 23 }
END_PATCH
ELEMENT
0 {  0  1  2  3 }
1 {  4  5  6  7 }
2 {  8  9 10 11 }
3 { 12 13 14 15 }
4 { 16 17 18 19 }
5 { 20 21 22 23 }
END_ELEM
END_ENTITY
```

For the sake of simplicity, the surfaces described here consist of one patch each. Similarly, each patch consists only one element. Clearly though, any surface or patch can be subdivided into multiple patches and elements by defining additional vertices and patch or element identifiers.

Environment File (*.WLD) Format Specification

Building an environment consists of copying and transforming entities into instances. For this, we need an environment data file format to describe which entities are to be copied and what 3-D linear transformations are to be applied to them.

```
WORLD world_name
COMMENT Environment Data File
entity_file_name
< sx sy sz >
< rx ry rz >
< tx ty tz >
entity_file_name
...
END_FILE
```

Similar to our entity data file format, the following syntax rules apply:

1. The data file consists of ASCII characters.
2. Each line must be terminated with an environment-specific "newline" character (typically <CR><LF> for MS-DOS and <CR> for UNIX systems).
3. The maximum length of a line is 256 characters, including the newline character(s).
4. Multiple whitespace (ASCII space and horizontal tab) characters between data values and separators are ignored.
5. Comment lines beginning with the keyword "COMMENT" are ignored.
6. The data file begins with the keyword "WORLD", followed on the same line by an optional world_name character string that identifies the environment. Any printable ASCII character is permitted in the string.
7. The remainder of the data file consists of one or more entity sections, followed by the "END_FILE" keyword. Any lines after this keyword are ignored.
8. An entity section consists of an entity_file_name character string, followed in sequence by a scaling vector, a rotation vector and a translation vector.
9. The entity_file_name is an environment-specific file name that uniquely identifies the entity data file.
10. A scaling vector consists of a '<' separator, followed by three floating point numbers representing the x-axis, y-axis and z-axis scaling factors respectively, followed by a '>' separator.
11. A rotation vector consists of a '<' separator, followed by three floating point numbers representing the x-axis, y-axis and z-axis rotation angles (in degrees) respectively, followed by a '>' separator.
12. A translation vector consists of a '<' separator, followed by three floating point numbers

representing the x-axis, y-axis and z-axis translation values respectively, followed by a '>' separator.

Here's an example of a data file that places two instances of our previously defined colored cube entity in a world environment:

```
WORLD colored cube
COMMENT first instance
col_cube.ent
< 2.0 3.0 1.0 >
< 30.0 45.0 0.0 >
< 2.0 0.0 0.0 >
COMMENT second instance
col_cube.ent
< 1.5 1.0 0.5 >
< 30.0 45.0 30.0 >
< 0.0 0.0 1.0 >
END_FILE
```

Again, these data file formats are only provided for those masochistic enough to write their own entity and environment files using a text editor and lots of patience!

#13$14+15**Read This Book!**

If this program intrigues you, then you may be interested in:

Radiosity: A Programmer's Perspective

by Ian Ashdown

Published by John Wiley & Sons, Inc., New York, NY
Paperback, 498 pages with 12 color plates, 1994

ISBN 0-471-30444-1 (without diskette), $39.95 US
ISBN 0-471-30488-3 (with 3.5-inch MS-DOS diskette), $54.95 US

Radiosity: A Programmer's Perspective offers step-by-step guidance for the development of a fully functional, radiosity-based rendering program for Microsoft Windows and other graphical environments, including:

=> A detailed explanation of radiosity theory and its associated algorithms (no knowledge of higher mathematics required!)

=> Complete, fully documented, and compiler-independent C++ source code (7,500+ lines) for **HELIOS***, a radiosity renderer for Microsoft Windows 3.1, Windows 95, and Windows NT.

=> An extensive guide to the computer graphics radiosity literature.

* The version of **HELIOS** presented in the book (Version 1.00A) employs a simplified MS-Windows user interface due to lack of space for the source code. All of the radiosity-related code, however, is identical to that included with this release of **HELIOS**.

The accompanying diskette includes C++ source code and MS-Windows 3.1 executable files for **HELIOS**. It also contains several sample environment files, a color quantization utility for MS-Windows Paintbrush-compatible BMP files, and a sample AutoCAD DXF file parser.

Make files are included for the Microsoft Visual C++ Version 1.5 and Borland C++ Version 4.5 compilers. Most of the 7,500+ lines of C++ source code are compiler-independent, and can be easily ported to other graphical environments.

[13] HELP_BOOK
[14] Read This Book!
[15] commands:010

[16][17][18]What's New?

Version 1.00A

The development of **HELIOS** is an ongoing project. This version was written specifically for the book, ***Radiosity: A Programmer's Perspective*** (Wiley, 1994).

Version 1.02A

This version was released as a demonstration program to illustrate what radiosity-based rendering techniques have to offer for photorealistic image synthesis. New features included:

* integral color quantization
* TARGA graphics file support
* image intensity and contrast controls
* parallel projection
* MS-Windows 3.1 multitasking support
* revised user interface
* radiosity calculations progress display
* MS-Windows on-line help
* automatic detection of display device capabilities

Version 1.02B

This is a minor maintenance release that adds the following features:

* antialiasing support
* minor user interface revisions
* 25 percent improvement in radiosity calculation and rendering speed
* **HELIOS** environment and entity file format documentation

Version 1.02C

This is another minor maintenance release. It adds no new features, but it does include *full C++ source code* for **HELIOS**, as well as project make files for Microsoft Visual C++ V1.5 and Borland C++ Version 4.5.

Please note that the source code is copyrighted. It may be freely copied, redistributed, and/or modified for personal, non-commercial use ONLY, provided the copyright notice is included with all source code files.

Version 1.03A

This release adds the following features:

* interactive environment viewing, including dolly, orbit, pan, rotate, tilt, and zoom
* default view display
* minor user interface revisions
* environment statistics reporting

Future releases of **HELIOS** are scheduled to include the following features:

* AutoCAD DXF file input
* automatic mesh generation
* adaptive subdivision
* hierarchical radiosity
* texture mapping support
* IESNA photometric data file support

The radiosity-related C++ source code included with this release of **HELIOS** is fully documented in the

[16] HELP_WHATS_NEW
[17] What's New?
[18] commands:010

book ***Radiosity: A Programmer's Perspective*** (Wiley, 1994). See ~~Read This Book!~~ for further details.

#19$20+21**Credits**

HELIOS was written by:

byHeart Software Limited
620 Ballantree Road
West Vancouver, B.C.
Canada V7S 1W3

Attn: Ian Ashdown, President

Web URL: http://www.ledalite.com
e-mail: iashdown@ledalite.com

The radiosity-related C++ source code included with this release of **HELIOS** is fully documented in the book ***Radiosity: A Programmer's Perspective*** (Wiley, 1994). See ~~Read This Book!~~ for further details

[19] HELP_CREDITS
[20] Credits
[21] commands:010

#22$23+24File Menu commands

~~Open~~

Opens an existing environment (*.WLD) file. The current environment file (if any) is automatically closed.

~~Save As~~

Saves the current bitmap image as a Microsoft Paintbrush-compatible BMP graphics file.

~~Entities~~

Specifies the directory where the entity (*.ENT) files can be found. If no directory is specified, the current directory is assumed.

Statistics

Displays the *Environment Statistics* dialog box that indicates the number of instances, surfaces, patches, elements, and vertices in an environment. This menu item is active only after an environment has been loaded.

Exit

Closes the current environment file (if any) and terminates the program.

[22] HELP_FILE_MENU
[23] File Menu
[24] commands:010

#25$26+27**Camera Menu commands**

~~Set Camera~~

Sets the field of view and clipping plane distances of the synthetic camera.

~~Set View~~

Specifies the position and orientation of the synthetic camera in three-dimensional space.

Default View

Initializes the position and orientation of the synthetic camera to its default values.

Interactive

Displays the interactive mode menu for controlling the synthetic camera's position and orientation using the mouse.

There are six interactive modes. Only one mode can be activated at a time. The mouse is enabled whenever a mode is active and the left mouse button is depressed. The wireframe or bitmap image is redrawn when the mouse button is released.

1. **Dolly** - the camera eye position and focus position are shifted forward or backward along the camera's line of sight (i.e., the view direction vector). Moving the mouse upwards on the screen moves the eye and focus positions forward; moving the mouse downwards moves them backward.
2. **Orbit** - the camera eye position is rotated about the focus position, with the camera's line of sight remaining fixed on the focus position. Moving the mouse in a given direction on the screen moves the eye position in the same direction.
3. **Pan** - the camera eye position and focus position are shifted horizontally and vertically perpendicular to the camera's line of sight. Moving the mouse in a given direction on the screen moves the eye and focus positions in the same direction.
4. **Rotate** - the camera's view direction and focus position are rotated horizontally and vertically about the eye position. Moving the mouse in a given direction on the screen rotates the view direction vector and moves the focus position in the same direction.
5. **Tilt** - the camera's view-up vector (and hence the viewed image) is tilted (rotated) relative to its default vertical orientation. Moving the mouse in a given direction on the screen tilts the view-up vector in the same direction.
6. **Zoom** - the camera's view distance is decreased or increased, resulting in a larger or smaller field of view. Moving the mouse upwards on the screen decreases the view distance; moving the mouse downwards increases the view distance. (This is equivalent to adjusting the focal length of a zoom lens on a camera.)

The interactive mode menu can also be displayed by clicking the right mouse button.

[25] HELP_CAMERA_MENU
[26] Camera Menu
[27] commands:020

#28$29+30**Render Menu commands**

Wireframe
Displays a wireframe view of the environment.

Shaded
Displays a shaded bitmap image without performing any radiosity calculations. Useful for examining the specified surface colors of entities in the environment.

Rendering
Performs radiosity calculations for the environment, then displays a true-color bitmap image.

Redisplay
Updates and redisplays the bitmap image without performing any radiosity calculations.

[28] HELP_RENDER_MENU
[29] Render Menu
[30] commands:020

#31$32+33Options Menu commands

~~Set Display~~

Sets the bitmap image display parameters.

~~Set Color~~

Sets the bitmap image color parameters.

~~Set Convergence~~

Sets the radiosity calculation convergence parameters.

[31] HELP_OPTIONS_MENU
[32] Options Menu
[33] commands:020

#34$35**Glossary**

ambient lighting
back plane distance
clipping plane
color quantization
convergence
dolly
element, surface
emittance
entity
environment
exitance
eye position
field of view
focus position
front plane distance
gamma
jitter
instance
luminance
orbit
pan
patch
positive overshooting
radiosity calculations
rotate
step
stopping criterion
synthetic camera
tilt
view direction
view distance
view window
view-up vector
zoom

[34] glossary
[35] Glossary

#36$37+38Open command (File menu)

Opens an existing environment (*.WLD) file. The current environment file (if any) is automatically closed.

Dialog Box Options

File Name
Select or type the name of the environment file to be opened. This box lists files with the filename extension selected in the List Files of type box.

List Files of Type
Selects the type of file for the File Name list. The extension .WLD indicates an environment (or *world*) file.

Directories
Selects the directory where the environment file is located.

Drives
Selects the drive where the environment file is located.

[36] HELP_OPEN
[37] Open
[38] commands:020

#39$40+41**Save As command (File menu)**

Saves the current bitmap image as a Microsoft Paintbrush-compatible BMP or TARGA graphics file.

Dialog Box Options

File Name
Select or type the name of the bitmap image file to be saved. If the file already exists, you will be asked whether you want to overwrite it.

Save File As Type
Select either BMP or TGA (TARGA) file format. The file will be saved in either 8-bit (256 color) or 24-bit (16.7 million color) format, depending on what color resolution is available or has been chosen. (See the ~~Set Display~~ command for further information.)

Directories
Selects the directory where the file is to be saved to.

Drives
Selects the drive where the file is to be saved to.

[39] HELP_SAVE_AS
[40] Save As
[41] commands:020

[#42$43+44]Entities command (File menu)

Sets the directory for the entity (*.ENT) files.

Dialog Box Options

Entities File Path
Enter the full path name of the directory where the entity files specified in the environment file can be found.

If no directory is specified, the current directory is assumed.

[42] HELP_SETDIR
[43] Directories
[44] commands:020

#45$46+47Set Camera command (Camera menu)

Defines the synthetic camera parameters.

Dialog Box Options

Camera Distances
Specifies the view distance to define the field of view and the front and back clipping plane distances.

The view distance must be greater than zero.

The front and back clipping distances are measured from the view window in the view direction.

The front clipping plane distance must be less than the back clipping plane distance.

Window Scale
Specify the window scaling factor.

Window Dimensions
Specify the horizontal and vertical dimensions of the bitmap in screen pixels.

The allowable range is from 32 to 1024 pixels.

Projection
Specify either perspective or parallel projection.

[45] HELP_SETCAMERA
[46] Set Parameters
[47] commands:020

#48$49+50**Set View command (Camera menu)**

Set the position and orientation of the synthetic camera in three-dimensional space.

Dialog Box Options

Eye Position
Sets the x-y-z coordinates for the eye position.

Focus Position
Sets the x-y-z coordinates for the focus position.

View Direction
Sets the horizontal and vertical angles for the view direction vector.

View-Up Vector
Sets the horizontal and vertical angles for the view-up vector.

NOTE: The view direction and view-up vectors cannot be collinear.

[48] HELP_SETVIEW
[49] Specify View
[50] commands:020

#51$52+53**Set Display command (Options menu)**

Specifies the bitmap image display parameters.

Dialog Box Options

Intensity
Sets the bitmap image intensity. The default value is 1.0.

NOTE: Adjusting the image intensity is equivalent to adding or subtracting ambient lighting to the environment.

Contrast
Sets the bitmap image contrast. The default value is 1.0.

Gamma
Sets the gamma value for the display. The default value of 2.2 is typical for most color monitors.

Color Reduction
Color reduction is performed by randomly jittering the RGB luminance of each 24-bit color pixel. It is useful for alleviating color banding effects visible in 24-bit color images displayed on systems that support 32,768 or 65,536 colors.

The Noise Level parameter determines the amount of jittering applies to each pixel. The allowable values are 1 to 8.

Antialiasing
Antialiasing is performed by supersampling with a Bartlett (i.e., triangular) filter. Increasing the resolution reduces the staircase effect (or “jaggies”) that are otherwise evident in the bitmap image.

[51] HELP_SETDISPLAY
[52] Set Parameters
[53] commands:020

[54][55][56]Set Color command (Options menu)

Specifies the bitmap image color parameters.

Dialog Box Options

Color Display
Specifies the type of color display for the bitmap image. The choices are:

RGB Color - true color display (24-bit or 8-bit color).
Grayscale - monochrome (8-bit grayscale).
Pseudocolor - false color display based on pixel luminance.

Color Resolution
Specifies number of colors in bitmap image. The choices are:

16.7 Million Colors - 24-bit color bitmap
256 Colors - 8-bit color bitmap

NOTE: The 16.7 Million Colors option is only available on computer systems capable of displaying at least 32,768 colors. On 256-color systems, the bitmap is first calculated as a 24-bit color image, then color quantized to convert it to an 8-bit color bitmap for display.

[54] HELP_SETCOLOR
[55] Set Parameters
[56] commands:020

#57$58+59Set Convergence command (Options menu)

Specifies the convergence parameters for the radiosity calculations.

Dialog Box Options

Maximum Steps
Specifies the maximum number of steps performed by the radiosity calculations. Each step represents the transfer of light between two surface elements. The allowable values are 1 to 2000.

Stopping Criterion
Specifies the stopping criterion for the radiosity calculations. This value represents the amount of light that has not yet been absorbed by the surfaces, expressed as a fraction of the original amount of light emitted by the light sources. The allowable value are 0.0 to 1.0.

Ambient Exitance
Specifies whether the currently remaining unabsorbed light is to be distributed equally throughout the environment as ambient lighting. Useful for quickly generating bitmap images after only a few steps, although the images will not be photometrically accurate. (See also exitance.)

Positive Overshoot
Specifies whether the radiosity calculations employ positive overshooting. The radiosity calculations will usually converge more quickly (that is, fewer steps will be required) when this option is enabled.

[57] HELP_SETCONVERGE
[58] Set Convergence
[59] commands:020