Request Date: 12-JAN-2010
Expiration Date: 20-JAN-2010
ILL Number:

ILL Number: 3628740
Call Number: UCSB:Main Lib T385 .C558 Sciences Engineering Library
Format: Article Printed
Title: Computer Graphics World
Article Author: Carey, Scott S
Article Title: Shades of a Higher Dimension
Part Pub. Date: 1987-10
Pages: 93-94
Requester: UCI Science Library

TGQ or OCLC #:

TGQ or OCLC #: 3628737
ID: UI3

ISBN/ISSN: 0271-4159

v. 10 pt 1 / pt 2 (1987)

Publisher: PennWell Publishing Corp.
Address: University of California - Irvine
Document Access & Delivery
Science Library
P.O. Box 19557
Ariel: 128.200.102.110, Fax: 949-824-3695

Patron Name: Black,Donald Vaughn (Graduate [])
Patron e-mail: DBLACK@UCI.EDU
Service Level: Normal - Full Search
Delivery Method: Electronic Mail
Request Note: Digital media (PDF format) preferred
Need by Date:
Verification Source: MELVYL-UCLinks-sfx:citation
Supplier Reference:
Owned By: UCSB Library

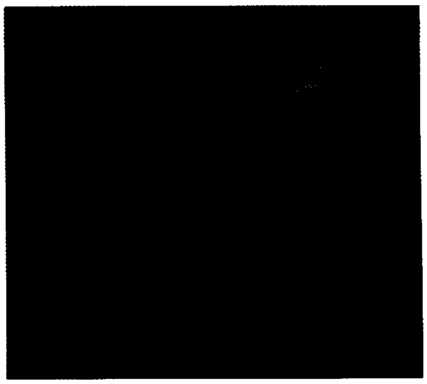
Service Type: Copy non returnable
Max Cost: USD50
Payment Type: IFM

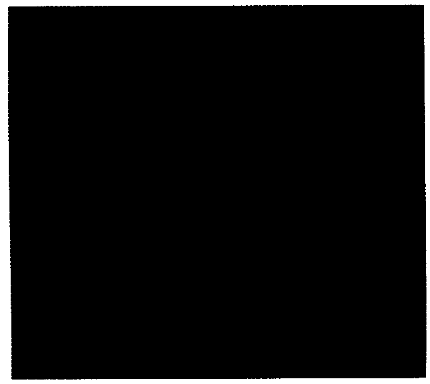
Copyright Compliance: CCG
Requester Symbol:
Return To: UCSB Interlibrary Loan/Davidson Library/525 UCEN Road/Santa Barbara, CA, 93106/U.S.A./ UCSB Ariel IP 128.111.96.251; Email ill@library.ucsb.edu

Shades of a Higher Dimension

Logic leads to a cogent display of *n*D reality

By Scott A. Carey,
Robert P. Burton, &
Douglas M. Campbell





A 4D hypercube shaded using polygon shading (left) looks more familiar than the more accurate solid-shaded version (right).

Just as extrapolations of 2D geometric theory led to the development of sensible 3D graphical displays, 3D theory has often been a stepping stone to representing 4D objects in a 2D projection. Following this fruitful line of logic, Brigham Young University researchers—Smith, Reeder, Armstrong, Liu, Isaacson, and Steiner—continue to generate ever-more-useful 4D scenes, recently adding hidden-volume removal to the list of methods used in Cartesian hypergraphics ("Hidden-Volume Removal in Four Dimensions," February, p.71).

In presenting this method, BYU researcher Victor Steiner also implemented a 4D shading model based on the 3D model described by James Foley and Andries Van Dam (1982).

A 3D shading model determines shades for visible surfaces, accounting for the characteristics of light sources and surfaces, including their positions and orientations. The shading model consists of a rule, or equation, for calculating the shade of a given point and the application of that rule to visible objects.

Foley and Van Dam's 3D shading rule includes terms representing ambient, diffuse, and specular light. The rule calls for calculating the diffuse and specular terms for each light source in a scene and adding their sum to the ambient term. Shades of colored surfaces are calculated by using this equation to determine the individual red, green, and blue components of each shade.

Extrapolation to 4D requires two changes to the premises underlying the 3D equation: (1) Whereas the 3D shading rule uses x,y,z vectors, a 4D rule uses x,y,z,w vectors; and (2) whereas a 3D shading rule calculates on the basis of normals to polygons (surfaces), a 4D rule uses normals to solids. Implementation of the rule is also slightly different. Instead of mapping the results of the equation to a square of pixels, 4D objects are mapped to a cube of voxels.

Given these changes, Foley and Van Dam's equation can be used to calculate the shade of a point in 4D. [*Interested readers may contact the authors c/o* CGW *for the exact equation.*] From this point on, the challenge lies in applying the rule to visible objects in a 4D scene and projecting that scene in 2D.

A valid projection of an n-dimensional scene is a projection which represents all portions of the scene which would be visible in n dimensions. Since all surfaces which bound a solid are visible in 4D, a valid 2D projection should show all surfaces that bound the solid, a paradoxical problem for representative graphics because the solids in 4D scenes are usually opaque. As a convention, Steiner uses transparency to show all visible portions of the scene concurrently. Within the model, then, the shading rule could be applied to either surfaces or solids.

Implementing a Shading Rule

Surface shading uses the polygons bounding the solids that make up 4D objects. It shades only points which belong to the visible surfaces in the scene. Solid shading, on the other hand, uses the solids composing the 4D objects. It shades all points which belong to the visible solids in the scene. The former

Scott A. Carey, Robert P. Burton, and Douglas M. Campbell are associated with the Computer Science Dept. of Brigham Young University (Provo, UT).

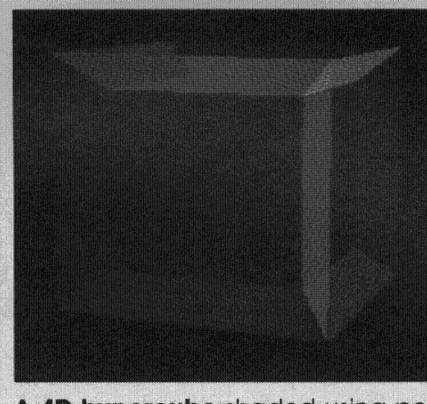
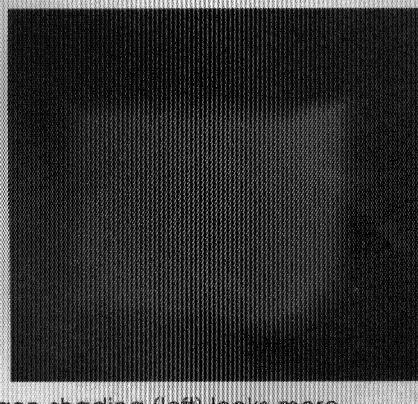
GRAPHICS

TECHNOLOGY

Shades of a Higher Dimension

Logic leads to a cogent display of *n*D reality

By Scott A. Carey,
Robert P. Burton, &
Douglas M. Campbell

A 4D hypercube shaded using polygon shading (left) looks more familiar than the more accurate solid-shaded version (right).

Just as extrapolations of 2D geometric theory led to the development of sensible 3D graphical displays, 3D theory has often been a stepping stone to representing 4D objects in a 2D projection. Following this fruitful line of logic, Brigham Young University researchers—Smith, Reeder, Armstrong, Liu, Isaacson, and Steiner—continue to generate ever-more-useful 4D scenes, recently adding hidden-volume removal to the list of methods used in Cartesian hypergraphics ("Hidden-Volume Removal in Four Dimensions," February, p.71).

In presenting this method, BYU researcher Victor Steiner also implemented a 4D shading model based on the 3D model described by James Foley and Andries Van Dam (1982).

A 3D shading model determines shades for visible surfaces, accounting for the characteristics of light sources and surfaces, including their positions and orientations. The shading model consists of a rule, or equation, for calculating the shade of a given point and the application of that rule to visible objects.

Foley and Van Dam's 3D shading rule includes terms representing ambient, diffuse, and specular light. The rule calls for calculating the diffuse and specular terms for each light source in a scene and adding their sum to the ambient term. Shades of colored surfaces are calculated by using this equation to determine the individual red, green, and blue components of each shade.

Extrapolation to 4D requires two changes to the premises underlying the 3D equation: (1) Whereas the 3D shading rule uses x,y,z vectors, a 4D rule uses x,y,z,w vectors; and (2) whereas a 3D shading rule calculates on the basis of normals to polygons (surfaces), a 4D rule uses normals to solids. Implementation of the rule is also slightly different. Instead of mapping the results of the equation to a square of pixels, 4D objects are mapped to a cube of voxels.

Given these changes, Foley and Van Dam's equation can be used to calculate the shade of a point in 4D. [*Interested readers may contact the authors c/o* CGW *for the exact equation.*] From this point on, the challenge lies in applying the rule to visible objects in a 4D scene and projecting that scene in 2D.

A valid projection of an n-dimensional scene is a projection which represents all portions of the scene which would be visible in n dimensions. Since all surfaces which bound a solid are visible in 4D, a valid 2D projection should show all surfaces that bound the solid, a paradoxical problem for representative graphics because the solids in 4D scenes are usually opaque. As a convention, Steiner uses transparency to show all visible portions of the scene concurrently. Within the model, then, the shading rule could be applied to either surfaces or solids.

Implementing a Shading Rule

Surface shading uses the polygons bounding the solids that make up 4D objects. It shades only points which belong to the visible surfaces in the scene. Solid shading, on the other hand, uses the solids composing the 4D objects. It shades all points which belong to the visible solids in the scene. The former

Scott A. Carey, Robert P. Burton, and Douglas M. Campbell are associated with the Computer Science Dept. of Brigham Young University (Provo, UT).

method yields a more familiar image while the latter is technically more accurate. Steiner uses surface shading in his hidden-volume algorithm.

To implement surface shading, reflectance characteristics are assigned to the surfaces of the visible solids in a 4D scene. The shading rule is used to determine the shades of all points on the surfaces. The surfaces are scan-converted, and the result is mapped to a cube of voxels.

At first glance, surface shading in 4D looks like conventional 3D shading. The important difference—also true for solid shading in 4D—is that the normal to the solid, rather than the normal to the surface, is used; the normal is constant over every part of the cube, rather than changing between faces as it does in 3D.

Because surface shading only determines the shades of the elements bounding a 4D object, the points inside a 4D object that should be visible aren't in the 2D projection. While generating a less familiar and slightly fuzzier image, solid shading overcomes this inaccuracy.

In solid shading, reflectance characteristics are assigned to the solids rather than to the surfaces of a 4D scene. The shade of each point in each visible solid is calculated, the solids are scan-converted to determine the shading value for every interior point, and the results are mapped to a cube of pixels.

The 2D projection shows every point which would be visible in 4D, providing an image that's faintest where the object is thinnest—fewer voxels mapped—and brightest where it's thickest—most voxels mapped. Because fewer voxels are mapped to the projection along the edges of the object, the image looks fuzzy.

The same principles used to scan-convert surfaces in 3D are applied to scan-converting solids in 4D. Solids are scan-converted by intersecting each solid of a 4D object at each y value in the 2D projection. If the solid intersects the current scan plane, then a polygon is formed from the intersection of the solid and the scan plane. That polygon is then scan-converted in the x,z plane to determine the shade of all points inside it. Executing these steps for every y value in the scene scan-converts the solid, thereby determining shading values for all points inside the solid.

[Interested readers may contact the authors c/o CGW *for the scan-conversion algorithm.]*

Surface and solid shading can be combined such that solid shading is used to shade the visible solids of a scene and surface shading to shade the surfaces belonging to each solid. The reflectance characteristics of each solid are assigned to each surface in each solid.

The resulting scene does not, however look much different from that generated by solid shading. The surface shading adds only a constant

In fact, the shading model extends to any dimension by following the pattern used to go from 3D to 4D.

shade to each solid in the scene. But if the viewpoint and light source are far away from the object, solid shading also gives the appearnace of a constant shade across all the surfaces of a given solid. (The only variables that change from point to point in solid shading are the distance to the light source, the vector to the viewpoint, and the vector to the light source.)

Objects characterized by either 4D surface or solid shading are inherently ambiguous in that any projection to a lower dimension always involves some loss of information. But both present valid if distinctly different 2D projections of the 4D objects they characterize.

As noted, surface shading yields more familiar-looking, if not insightful or accurate, representations than does solid shading. Given its representation of all visible points, solid shading is a technically more accurate method.

Shadows are also difficult to represent accurately using surface shading. In 3D, a shadow may fall on a surface such that only points inside the surface are shadowed. In 4D, a shadow may likewise fall entirely on points inside one of the solids of a hyperobject—points found only in solid shading.

Using surface shading, the proper shade for points in such a shadow can be calculated only if the shadow is cast on points which lie on the bounding surfaces of a solid. There are two ways to calculate a shadow using a predetermined shadow value, but they yield only crude approximations.

Because all inside points are shaded, considerably more time is required to calculate a solid-shaded 4D scene. Surface shading is thus less computationally expensive.

The shading model presented here includes a 4D extrapolation of a 3D shading rule and two methods—surface and solid shading—to apply that rule to 4D objects. Each method has its merits and flaws; choosing between them is a matter of assigning specific graphics application priorities.

The important thing is that shading can be used to characterize 4D scenes with greater sensibility and accuracy. In fact, the shading model can be extended to nD by following the pattern used to go from 3D to 4D:

- All vectors become $n$tuples
- Normals to $(n - 1)$D objects are used
- nD objects are mapped to an $(n - 1)$D hypercube of hyxels

Prior to Steiner, researchers used line drawings to represent hyperobjects. His hidden-volume elimination research used shaded images to show the results of hidden-volume removal. Steiner applied a rudimentary shading rule to the bounding polygons of each visible solid in the scene. This is the only reported attempt to shade hyperobjects. The pictures calculated through application of polygon shading and solid shading demonstrate that shading can be used to shade hyperdimensional objects. The logic extends through application of the nD shading rule to $(n - 1)$D entities. **CGW**