Point Sets with Attributes and Volume Modeling

Prof. Alexander Pasko

http://hm.softalliance.net/

Alexander Pasko, Evgenii Maltsev

Basic terms

Shape Modeling is an area of computer science, which studies methods and tools for modeling point sets in geometric spaces
(e.g., an N-dimensional Euclidean space).

Volume modeling studies

3D Point Sets with Attributes

Point Sets with Attributes

- Physical and abstract objects and phenomena are modeled as point sets
- Properties are modeled as attributes:
  - photometric (opacity, color, reflectance)
  - material, density, temperature, etc.
- Volume model:
  3D point set + point attributes
- Each point: $\{(x,y,z),(s_1,s_2,\ldots,s_n)\}$

Application areas

- Fabrication of objects with multiple materials and varying material distribution;
- Physics based simulations for the analysis of physical fields distribution over the given geometric areas;
- Analysis of geological structures;
- Medical examination and surgery simulation using computer tomography and other scanning devices;
- Visualization of objects with varying optical characteristics, amorphous and gaseous phenomena.

Areas of study

- **Solid modeling**:

1) 3D homogeneous solids with constant attributes assigned to the entire solid
2) Primitives with attributes + operations on attributes
3) General object model (Kumar et al. 1999)

- **Computer graphics**:

1) Surface + texture mapping
2) Any 3D geometry + solid texturing with *solid texture functions T(u,v,w)*

Point Sets with Attributes

- **Volume graphics**:

1) Geometric block +
   discrete field (grid) of attributes
   (homogeneous and heterogeneous volumes)
2) Isosurface geometry + grid of attributes

- **Volume modeling**:

1) Volume Model [Nielson, 2000] as a function of 3D or 4D point coordinates
2) Constructive Volume Geometry (CVG) [Chen & Tucker 2000]
3) Constructive Hypervolume Model [Schmitt et al. 2001]

Hypervolume model

3D point set

+ time

+ more dimensions (parameters)

+ point attributes

⇨ time-dependent volume or hypervolume

Each point: $\{(x,y,z,t,u,v,\ldots),(s_1,s_2,\ldots,s_n)\}$

Aspects of a model

The following aspects of different approaches to be discussed:

- Model of a point set
- Point set dimensionality
- Operations on points sets
- Type of attributes
- Attribute models
- Operations on attributes

Survey

Pasko A., Adzhiev V., Schmitt B., Schlick C., **Constructive hypervolume modeling**, *Graphical Models*, special issue on Volume Modeling, vol. 63, No. 6, November 2001, pp. 413-442.

http://cis.k.hosei.ac.jp/~F-rep/Fhypervol.html

Constructive Volume Modeling

- Volumetric CSG extension
- Constructive Volume Geometry
- Volume Scene Graphs

Volumetric CSG extension

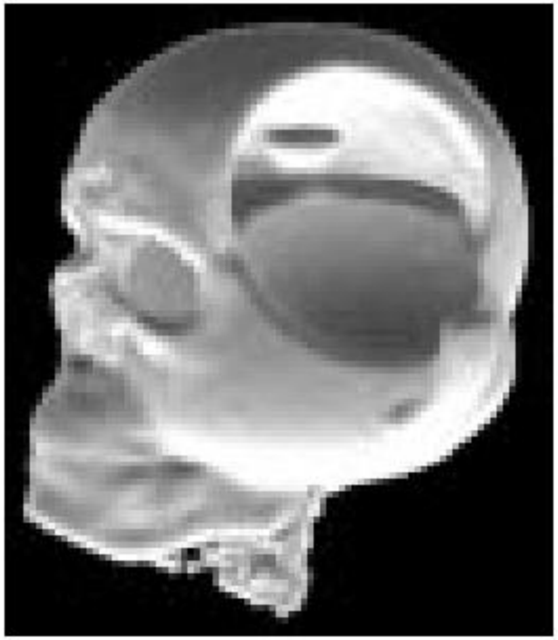
Shiaofen Fang and Duoduo Liao, **Fast CSG voxelization by frame buffer pixel mapping**, IEEE Symposium on Volume Visualization (Volviz'00), 2000, pp 43-48.

Duoduo Liao and Shiaofen Fang, **Fast volumetric CSG modeling using standard graphics system**, ACM Solid Modeling, 2002, pp. 204-211.

Volumetric CSG Model

- Extended CSG primitives:
  - Standard algebraic primitives
  - Boundary surface primitives
  - Objects extracted from voxel arrays using intensity thresholding
- Conversion of a CSG model into a voxel array
- Hardware support for rendering

Volumetric CSG Model



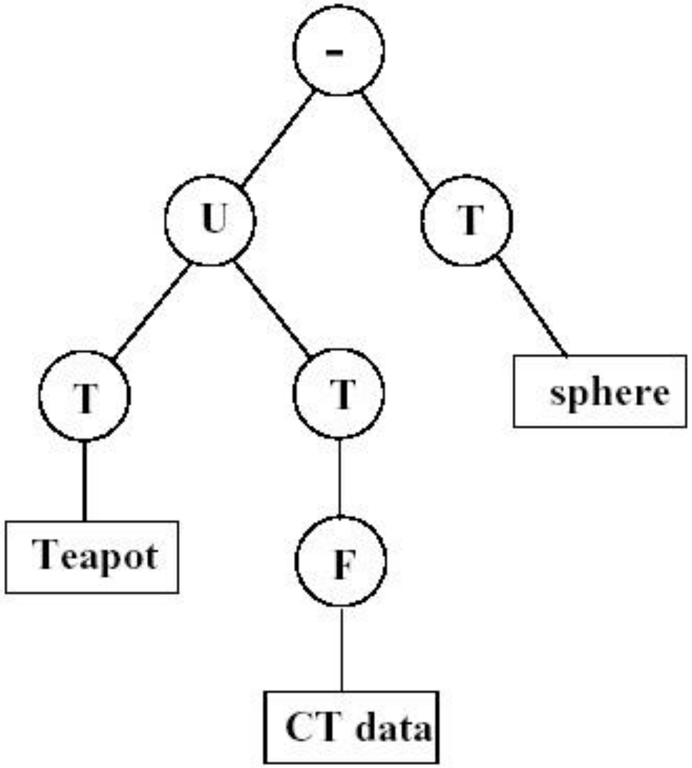


Figure 1: A CSG model using a CT data set. **T**: affine transformation; **F**: thresholding operation

Thresholoding operation for intensity *I* at point *P*:

$$F_{[s,t]}(V) = \{P \in R^3 \; : \; I(P) \in [s,t]\}$$

Binary CSG Voxelization

Apply the point membership classification with respect to the CSG object for all sampling points in a volume space:

- Intensity value = 1 an inside point
- Intensity value = 0 an outside point

Point Classification

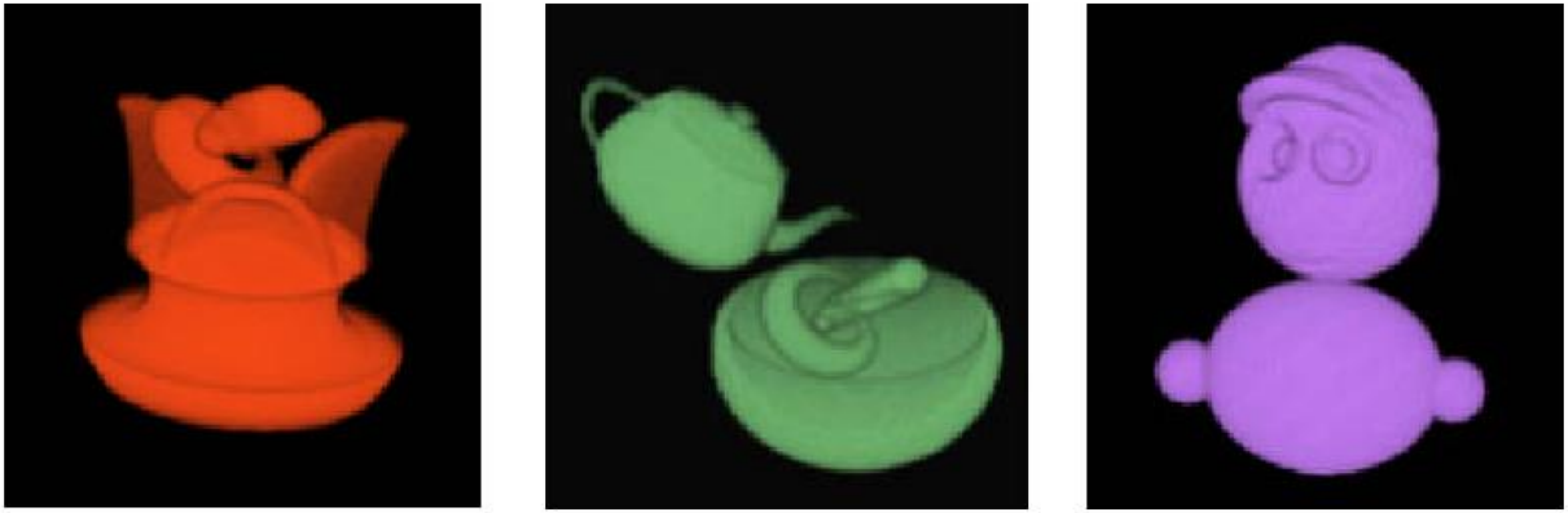
```
PCM PROCEDURE CSG_PCM (CSG_NODE N)
BEGIN

  IF (N is a leaf node)
     RETURN the PCM of the primitive;
  ELSE
     PCM_left = CSG_PCM (N.left);
     PCM_right = CSG_PCM (N.right);
     PCM_combined =
       combine (PCM_left, PCM_right, N.op);
     RETURN PCM_combined;
  ENDIF

END
```

CSG Volume Rendering

- Input: binary volume in the system texture memory
- Direct rendering: use 3D texture mapping
- Aliasing problem:
  - Trilinear interpolation in 3D texture mapping
  - Multi-pass 3D texture mapping approach:

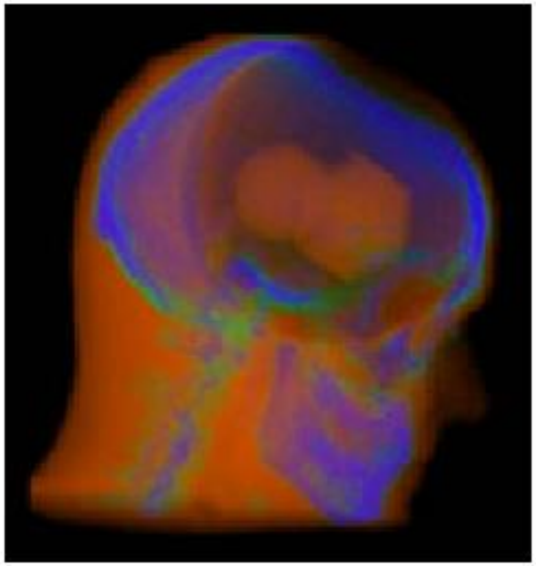
    Iteratively writing each texture mapped slice from frame buffer back to the texture memory.

Examples (1)

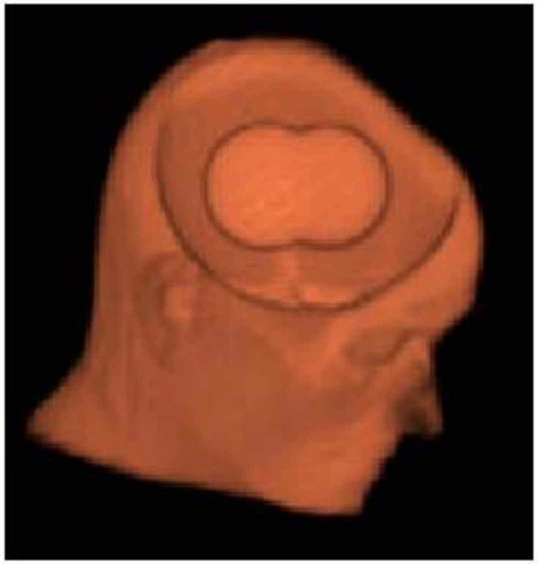


(a) (b) (c)

Examples (2)



(d)

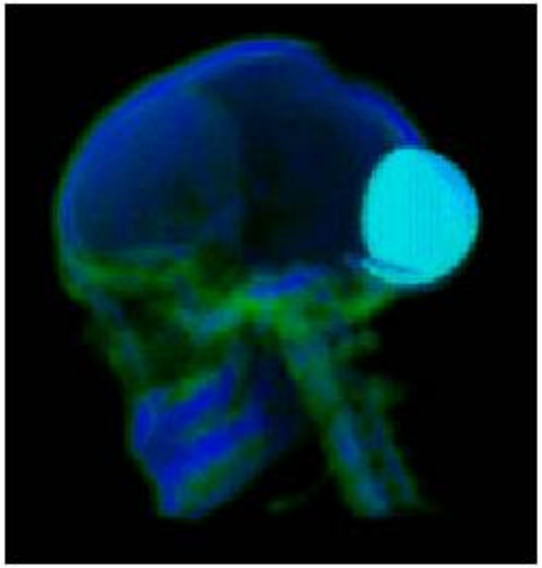


(e)

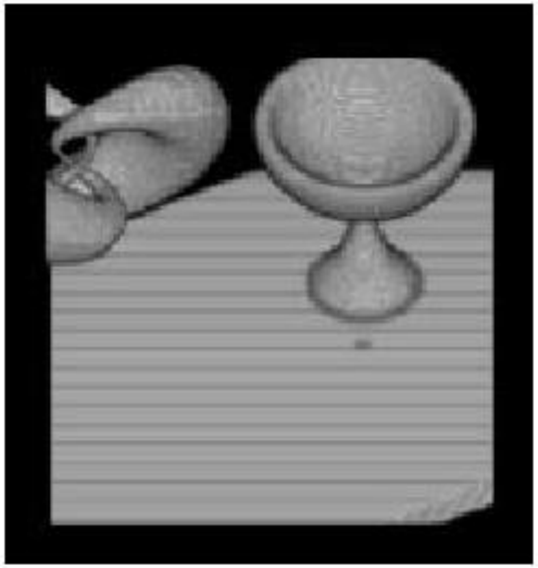
(f)

Examples (3)

(g)



(h)



(i)

Constructive Volume Geometry

Min Chen and John V. Tucker,
Computer Graphics Forum,
vol. 19, No. 4, 2000, pp. 281-293.

Motivation

- Conventional voxel-based modelling method:
  - lack of a high-level method for building complex objects from simple ones;
  - lack of a mature mathematical specification for combinational operations on volumetric datasets;
- Schemes like CSG and implicit surfaces are unable to specify internal structures and to model amorphous phenomena.

Constructive Volume Geometry (CVG)

CVG is an algebraic framework used to model complex spatial objects using combinational algebraic operations.

Components of CVG

- A CVG algebra consists of a set of spatial objects and some operations on these objects.
- By specifying the operations of a CVG algebra and applying them to some spatial objects, more complex spatial objects are created.

Scalar Fields and Spatial Objects

- Scalar fields
  - Let R denote the set of all real numbers, and E^3 denote 3D Euclidean space. A scalar field is a function F: $E^3 \rightarrow R$.
- Spatial objects
  - A spatial object is a tuple, o=(O, A_1, …, A_k), of scalar fields defined in E^3.
  - An opacity field O:$E^3 \rightarrow [0,1]$ specifying the “visibility” of every point p in E^3. It is the most elementary field in CVG.
  - Other attribute fields $A_1, \ldots A_k : E^3 \rightarrow R, k \geq 0$.
  - $A_1, \ldots A_k$ can be colours, reflection coefficients or non-graphical properties such as magnetic or distance.

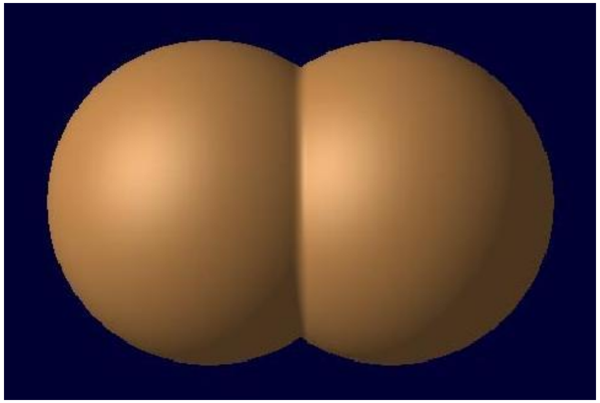
Operations on Spatial Objects

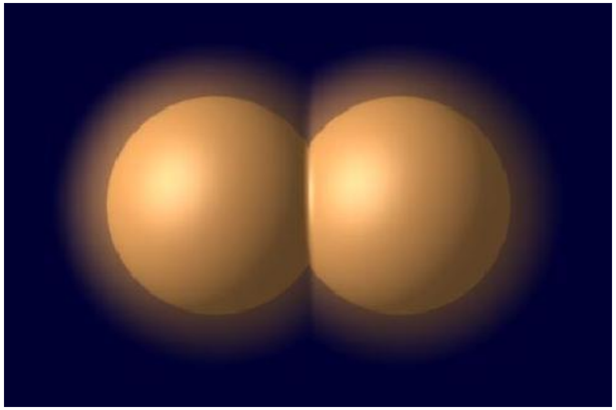
- A CVG algebra is simply a spatial object class and a family of operations $\phi_1, \phi_2, \ldots, \phi_m$.
- Given a CVG algebra, we can choose some spatial objects and apply a sequence of operations to create a complex spatial object in the algebra.
- Operations defined upon spatial objects can be decomposed into simple arithmetic operations on scalars through a series of operational decompositions.

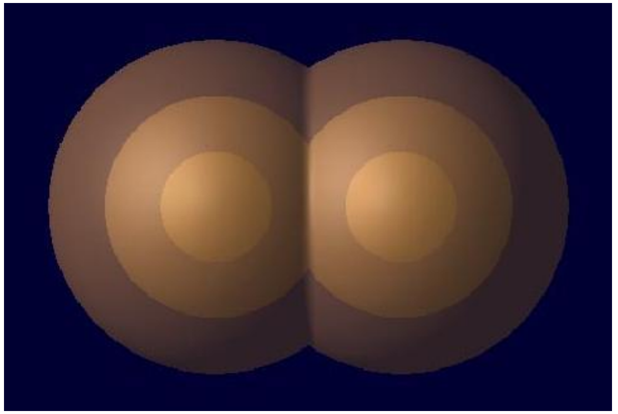
Opacity

- Opacity field $O: E^3 \rightarrow [0,1]$ that determines the visibility of the object.
- Three basic combinational operations:
  - Union: $\cup (o_1, o_2) = MAX(O_1, O_2)$
  - Intersection: $\cap (o_1, o_2) = MIN(O_1, O_2)$
  - Difference: $-(o_1, o_2) = SUB_{01}(O_1, O_2)$

CVG vs CSG







O(p) = 1 if r≤ 1

O(p) = 0 o/w

O(p) =1 if r<0.6

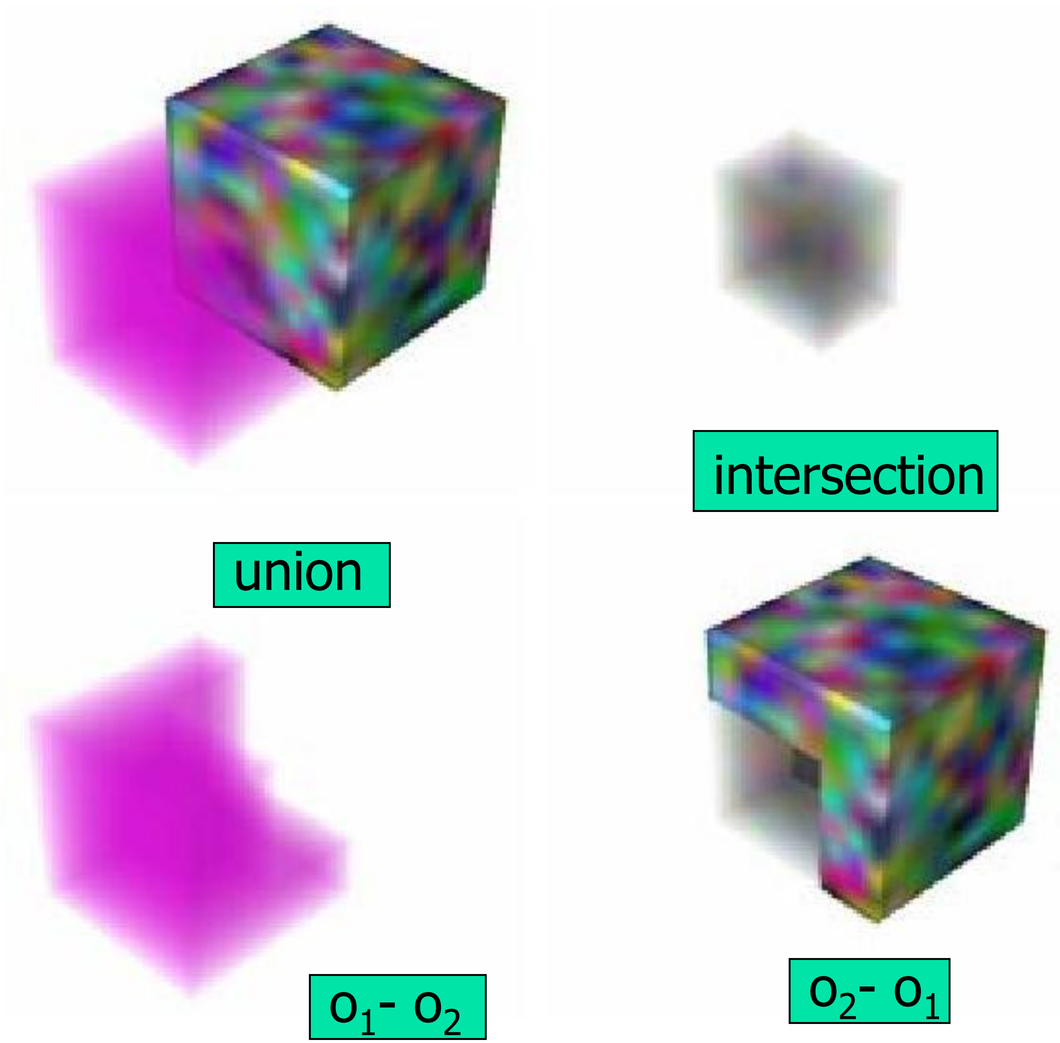
O(p) = 0.9-r if 0.6≤r≤0.9

O(p) = 0 o/w

O(p) =1-r if r ≤ 1

O(p) = 0 o/w

CVG operations

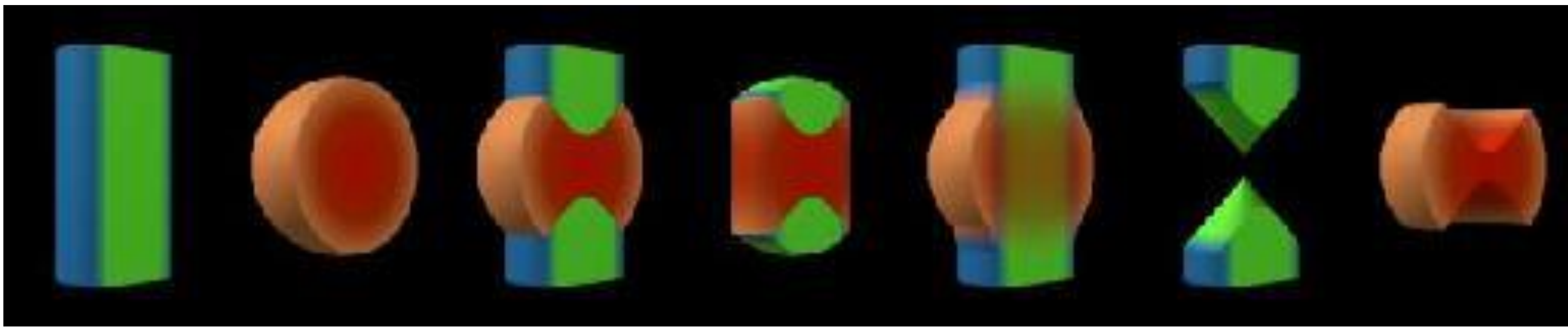


Applying basic CVG operations to two cubic spatial objects.

O1 is of opacity 0.5 and is associated with a constant color;

O2 is of opacity 1, and it's color fields are interpolated.

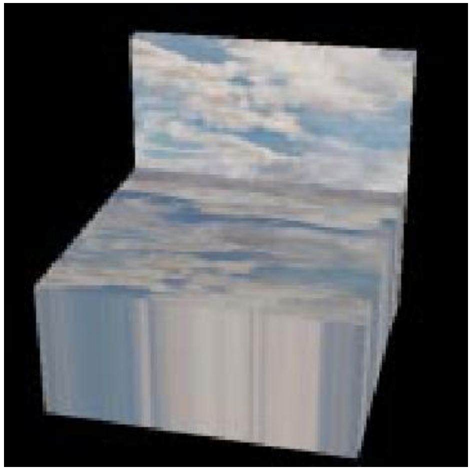
CVG operations (2)



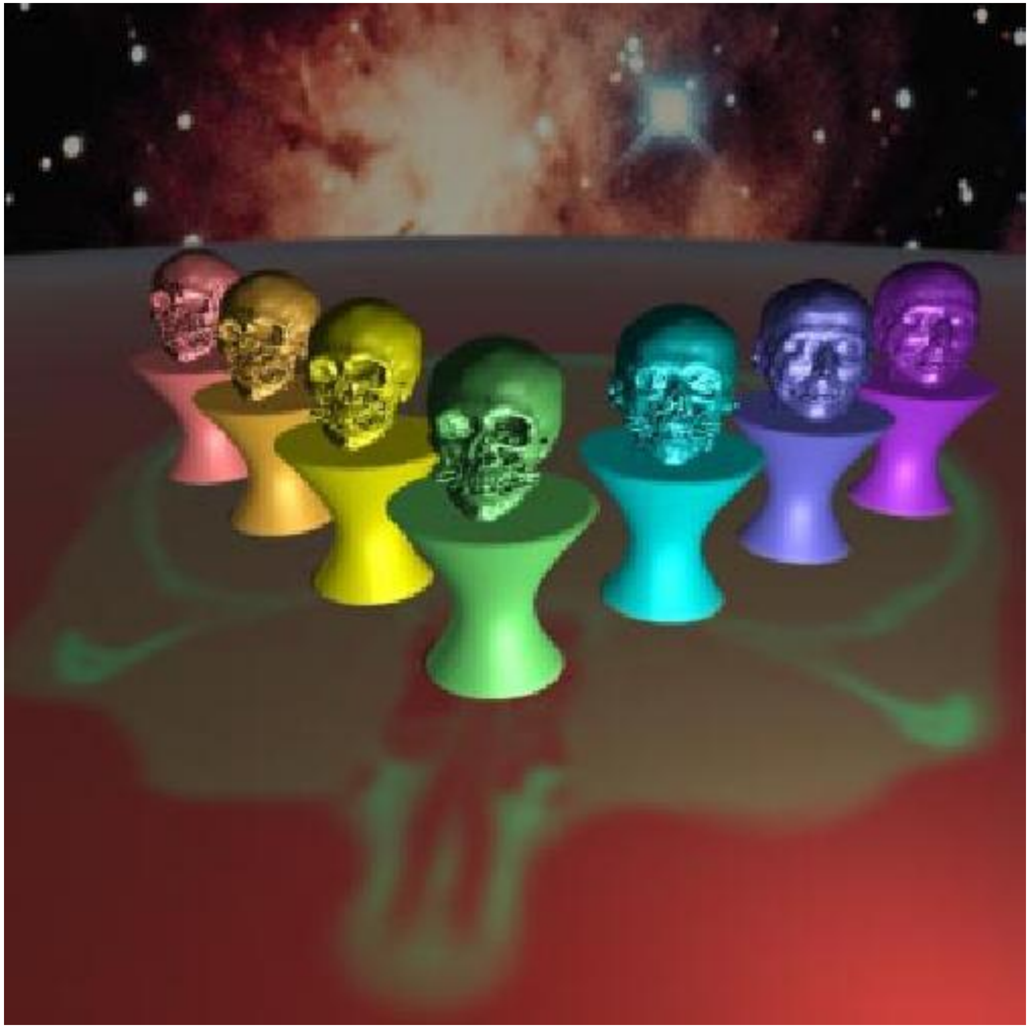
Effect of combinational operations to the interior of the objects

From left to right, they are cylinder c, sphere s, U(c,s), ∩(c, s), +(c,s), -(c,s) and –(s, c)

CVG operations (3)



CVG Operations (4)



Each head or skull is combined with the corresponding stool through a union operation before color mapping is applied

Problems of CVG

- Usage of the opacity field to ”implicitly define the visible geometry of an object”

  ⇨ The shape of an object does not necessarily predefine its photometric characteristics and vice versa.

- Limited operations, min/max for set-theoretic operations.
- Only static 3D objects are supported.

Volume Scene Graphs

D. Nadeau, IEEE Symposium on Volume Visualization (Volviz'00), 2000 pp. 49-56.

Scene Graphs

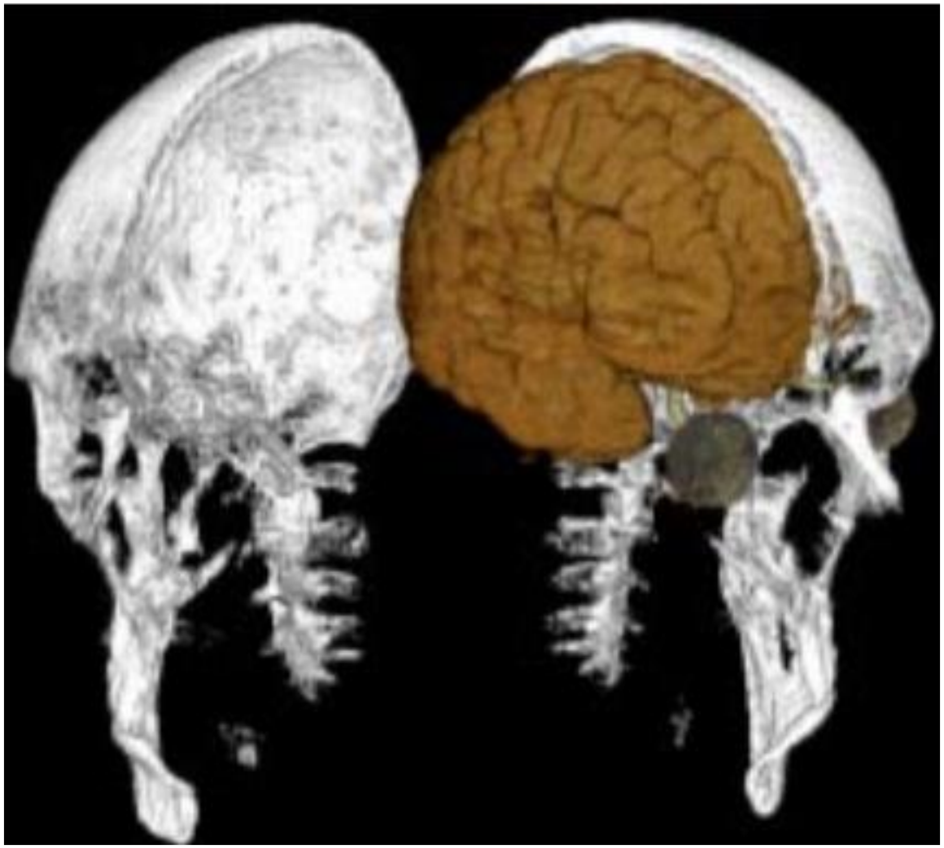
- A scene graph is a ***hierarchical organization*** of shapes, groups of shapes, and groups of groups that collectively define the content of a scene
- Shapes and groups may be shared among groups, creating a directed acyclic graph

Volume Scene Graphs

- VSG support volumetric objects defined procedurally or by volume data sets of differing resolution
- Operators in VSG specify transforms, filtering, cutting, etc.
- VSG are an explicit scene description reminiscent of CSG trees and related to CVG trees

Example of VSG

Three overlapping volume data sets at differing resolutions



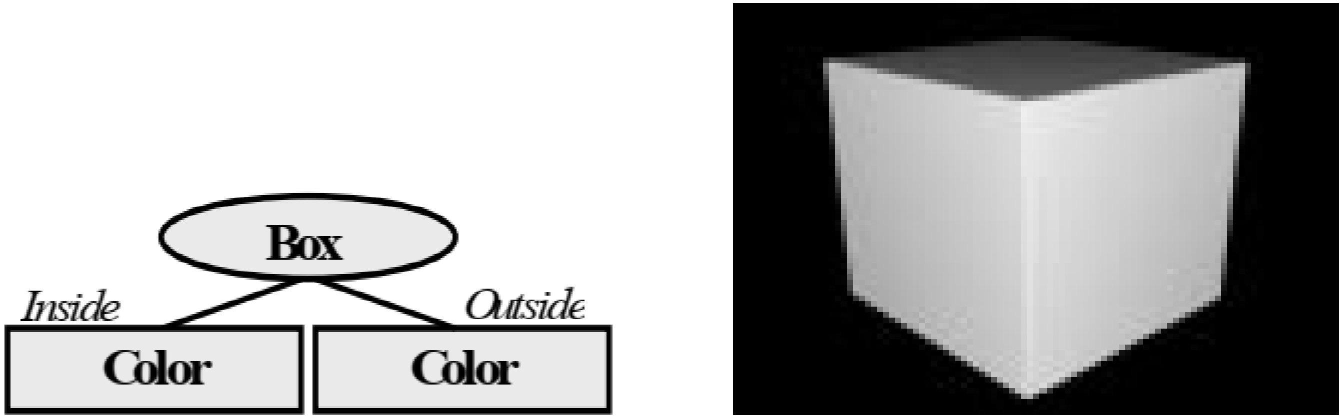
VSG Nodes

- Each node is a function that operates over a location of a point in space and time.

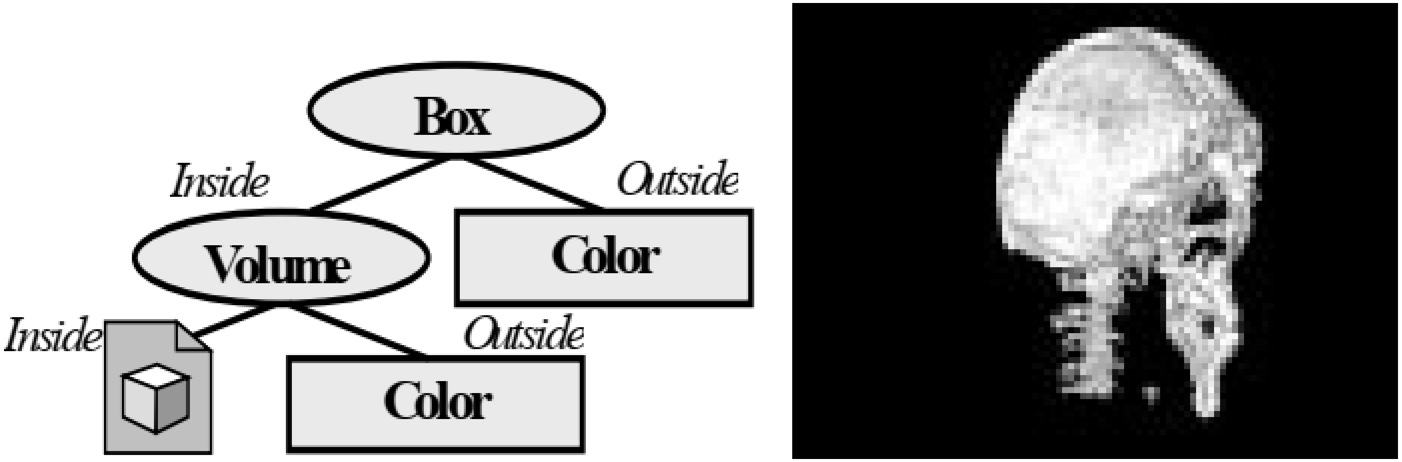
$$s = f(x, y, z, t)$$

s is a sample, for example (R,G,B)

Examples of Nodes



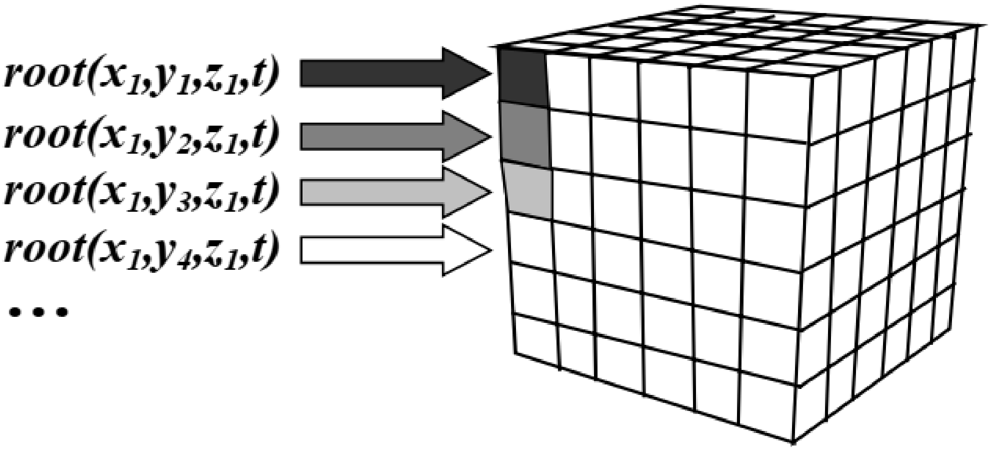
A scene graph defining a volumetric box.



Cutting a volume data set in half

Voxelizing VSG

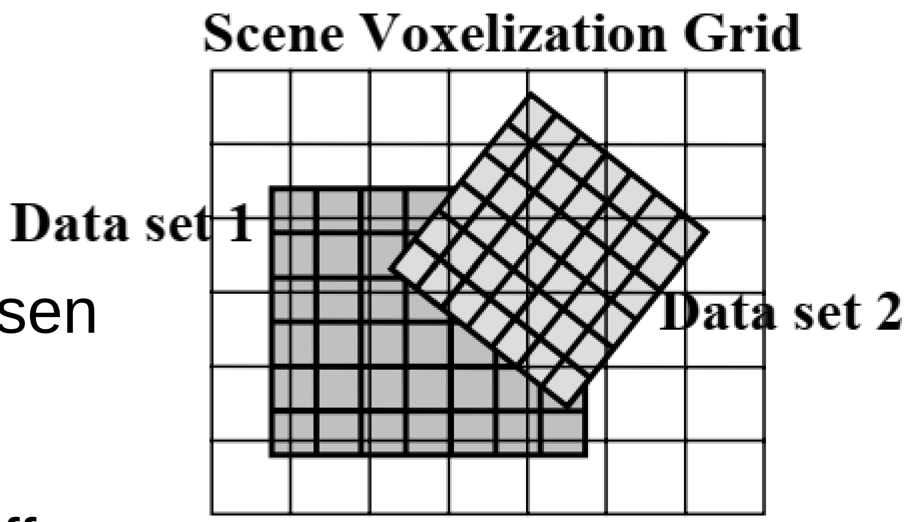
- VSG describes scene content, not its rendering
- In order to render, the scene must be voxelized



Voxelizing a scene graph on a grid.

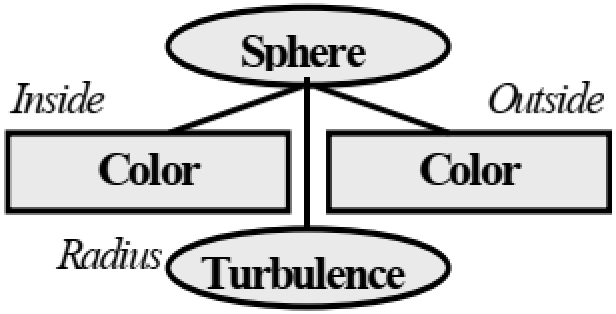
Alignment

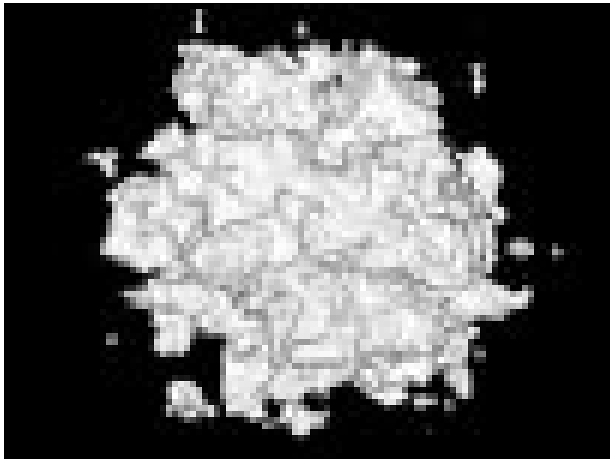
Scene graph voxelization samples the scene on a chosen grid.
The size, orientation, and resolution of that grid may differ from that of any child data set



The scene's voxelization grid need not align with that of any child data set

Node Fields



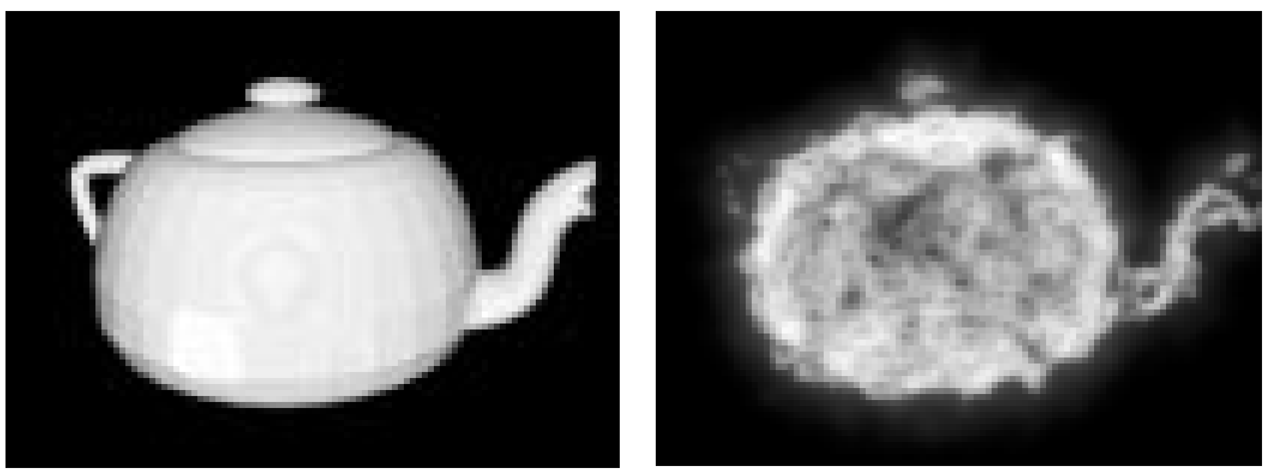


A sphere with its radius set by turbulence.

Any value can be replaced by a node evaluated at an *(x,y,z,t)* location.

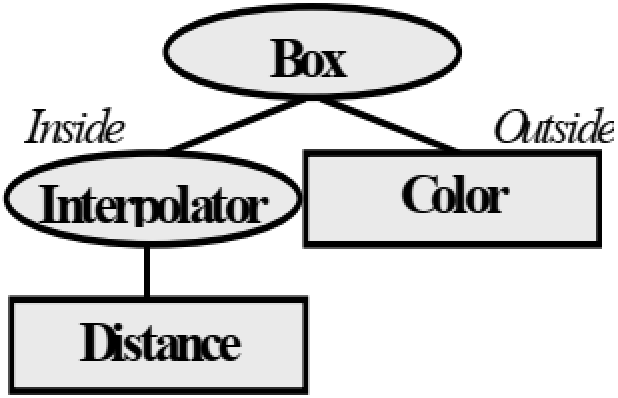
Node Examples

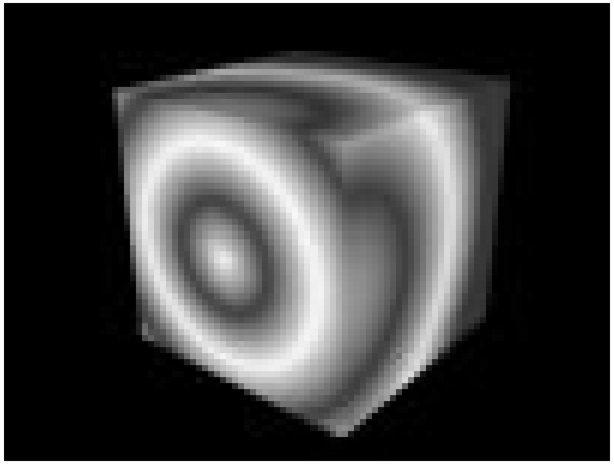
Polygonal data may be incorporated into the scene graph using a *SurfaceDistance* node. The node computes a *distance field* that, at each point in space, returns the distance from that point to the nearest point on a polygonal surface. The distance can be used with an *Interpolator* to vary opacity with distance from the surface. Or use *Turbulence* to deform the surface:



A distance field for a smooth surface (left), and with turbulence to deform the surface (right).

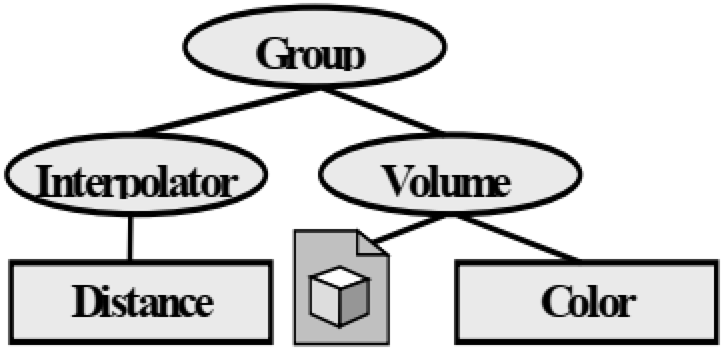
Node Examples

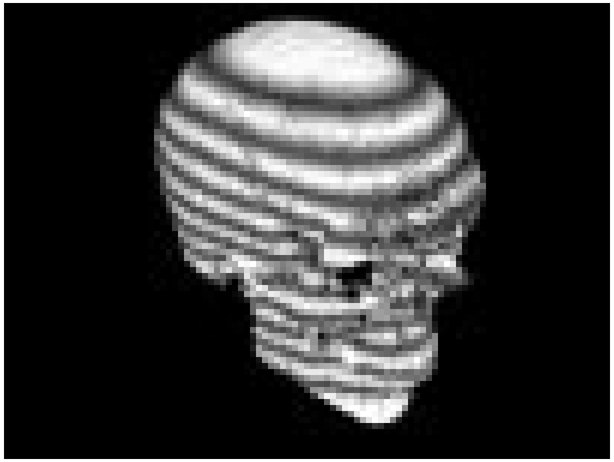




A spatial color gradient.

Node Examples





Use a color gradient to highlight contour.

Orion Nebula - Complex VSG

- 2112 nodes
- Cutting in 86 subtrees
- Separate voxelization
- Volume rendering