UCLA

CS174A : Introduction to Computer Graphics

Royce 190
TT 4-6pm

Scott Friedman, Ph.D
UCLA Institute for Digital Research and Education

Volume Rendering

- We have only considered surfaces so far.
- While these surfaces have been in three dimensional space.
- We have not looked at how values over a three dimensional field might be rendered.

- Today we look at different approaches to this type of rendering.

Volume Rendering

- Usually volume rendering concerns itself with how to display a scalar field.
- A function that returns a scalar value at a particular location.

$$scalar = f(x, y, z)$$

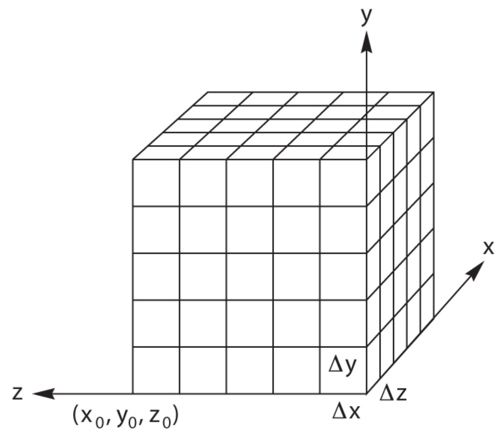
- This poses some challenges.

Volume Rendering

- Challenges
  - A lot more data needed to represent a volume.
  - Managing volume data is more difficult
    - Think how you would store it in a file
  - How to display in a meaningful way?
  - Here we attempt to display not just the surface of an object but some aspect of its interior.

Volume Rendering

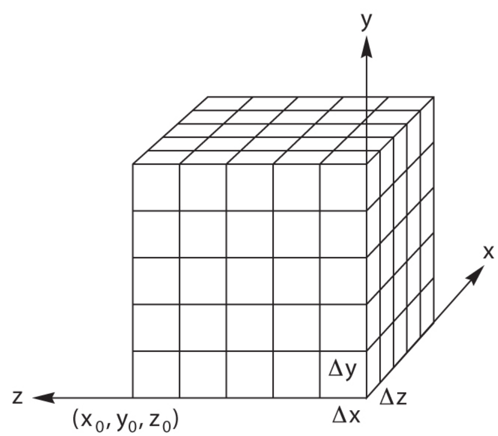
- Data
  - The data itself could be a sampling of some physical process.
    - Medical scan
    - Physical simulation of some type
  - Results in a 3d array of scalar sample values.



UCLA

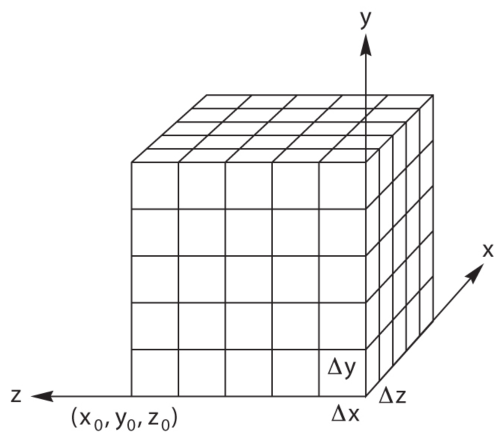
Volume Rendering

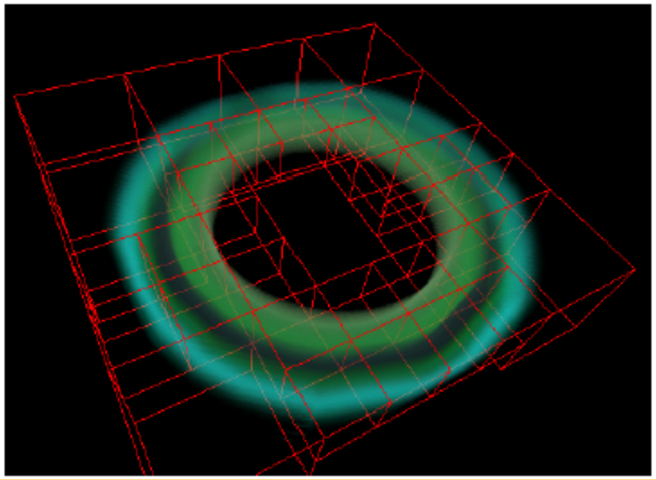
- Data
  - Each one of these 3d cells containing a sample is called a ***voxel***.
  - The scalar value that each *voxel* represents is understood to be average value for that cell.



Volume Rendering

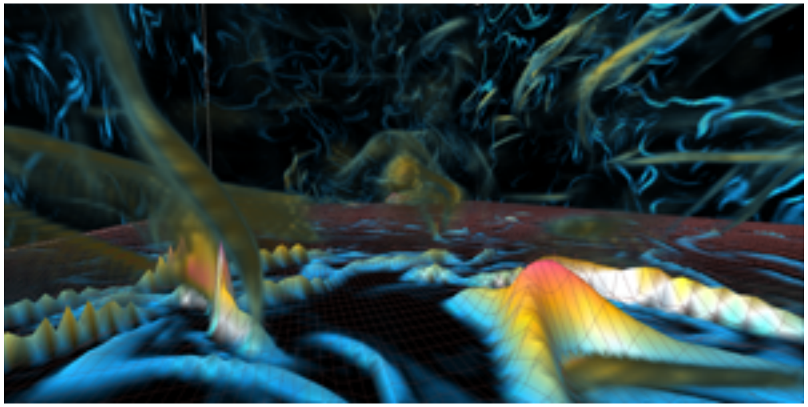
- Data
  - Here the 3d array of *voxels* is equally spaced which is often referred to as a structured data set.
  - Irregularly spaced data is, not surprisingly, called unstructured data set.

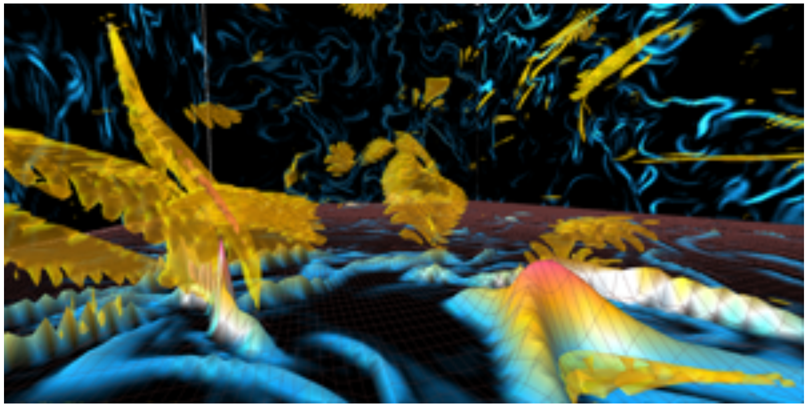




Volume Rendering

- Rendering
  - There are two basic approaches to volume rendering.
  - Direct volume rendering
    - Use every voxel to produce an image.
  - Iso-surface rendering
    - Use an implicit equation to determine what is rendered.





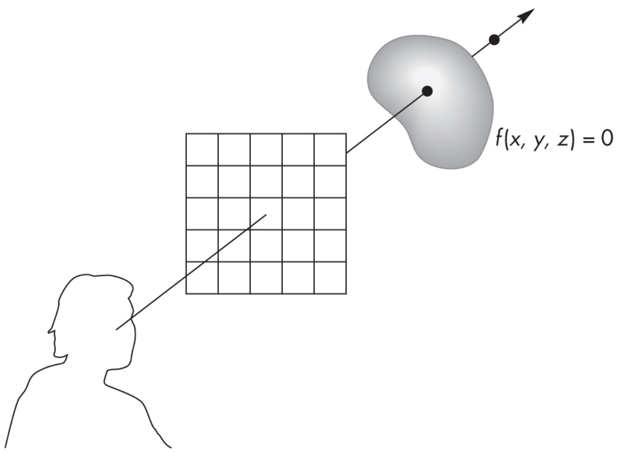
Volume Rendering

- Rendering – iso-surfaces
  - Implicit equation, $c=f(x,y,z)$
  - Extension of contours to 3d
  - Finding the "surface" that exists where the implicit function describing our data equals a particular value.
  - The scalar values of this data field could represent things like
    - Temperature, pressure, density, etc.
  - We want a surface that shows where that value is true within a data set.
    - Where, within the 3d field, is the temperature 275 degrees?

UCLA

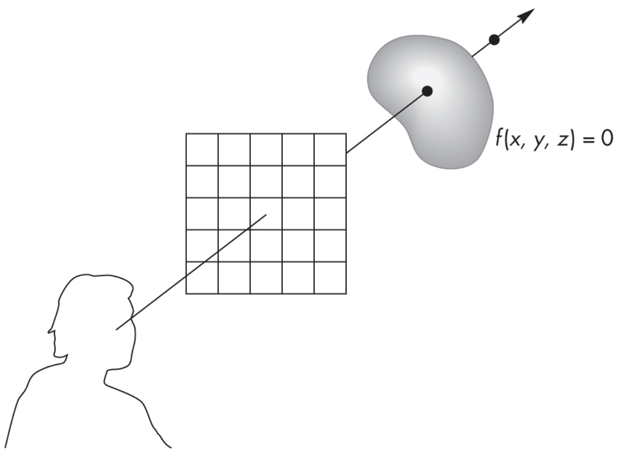
Volume Rendering

- Rendering – iso-surfaces
  - We could project a ray from our eye/camera through every pixel on the display and into our data field.
  - This, however, requires being able to compute the point along the line where the value is equal to our target value.
    - i.e. where along the ray is the scalar value equal to the value of interest?



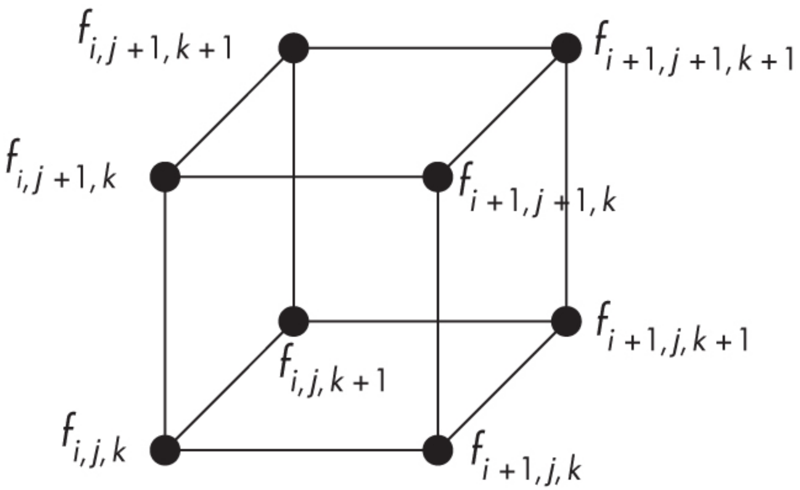
Volume Rendering

- Rendering – iso-surfaces
    - This is computationally expensive since there is no simple way to do this unless we use a basic, known, quartic – like a sphere.
    - Additionally, volume data is generally discretized and not continuous.
    - We have a bit of a problem…



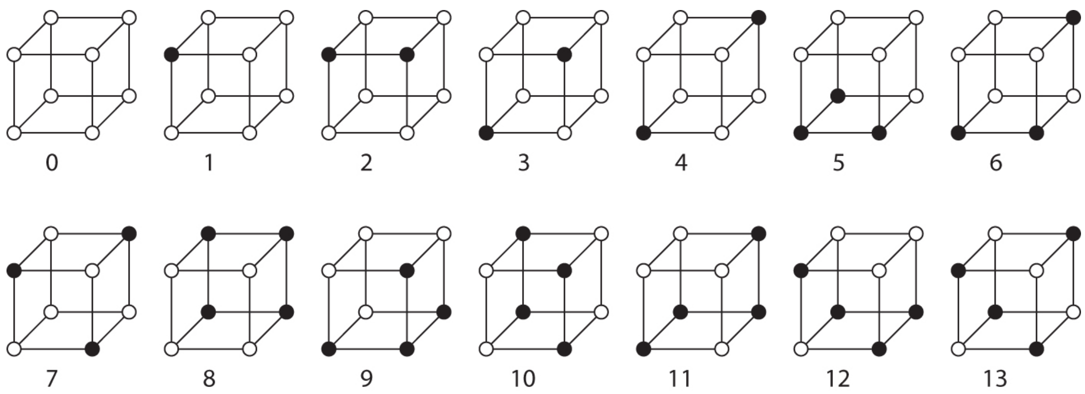
Volume Rendering

- Rendering – iso-surfaces
  - *Marching cubes* is the best way to solve this problem.
  - This technique forms a surface mesh representing the approximate value in question over a discretized scalar field.
  - Below is a representation of a voxel with it's corners defined by the scalar function *f*.



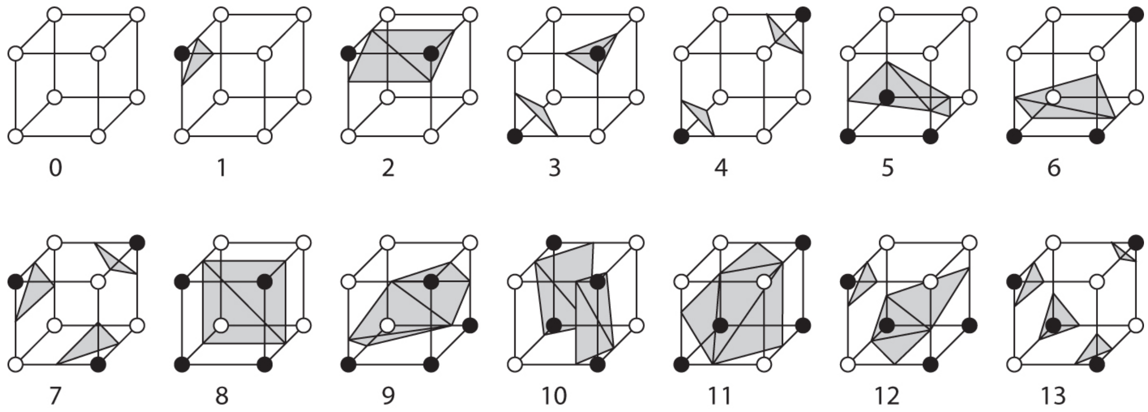
Volume Rendering

- Rendering – iso-surfaces
  - The vertices of these cells are compared to the iso-surface value in question, *c*.
  - Each vertex is colored based on whether the value is greater or less than the iso-surface value.
  - Accounting for symmetry there are a total of 14 variations.



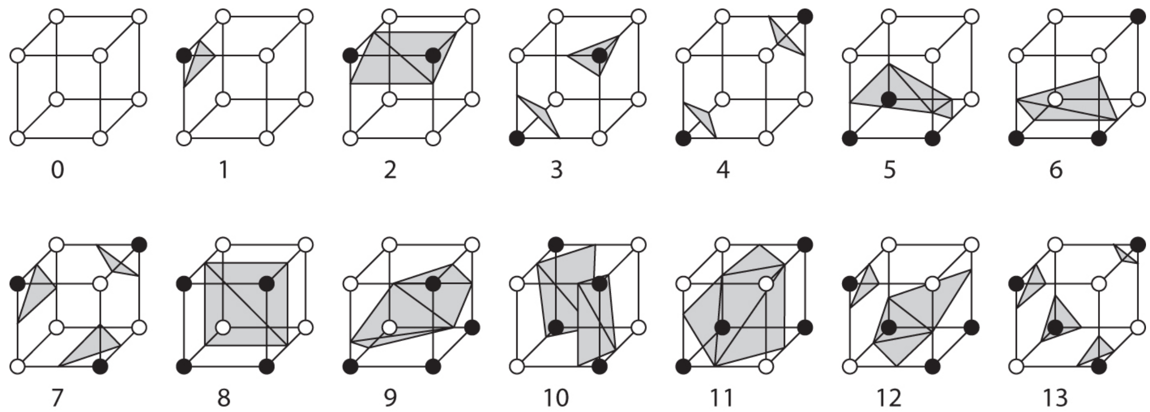
Volume Rendering

- Rendering – iso-surfaces
  - Specific intersections along each edge can be found by interpolation.
  - Triangles are used to tessellate the portion of the desired iso-surface mesh passing through the cell.



Volume Rendering

- Rendering – iso-surfaces
  - Each voxel contributes to eight of these cells.
  - When processing we move from one cell to the next, row by row and then plane by plane.
    - Hence the name, marching cubes.



Volume Rendering

- Rendering – iso-surfaces
  - As each cell is processed the triangles can be sent directly to the rendering pipeline.
    - It is also reasonable to collect all the triangles and build a more efficient mesh.
    - There can be ambiguities in the resulting mesh – but there is not enough information in the data to resolve them all the time.
  - In some sense iso-surfaces make rendering volumes manageable as they significantly reduce the amount of data needed to be rendered.
    - Only a fraction of the cells are ultimately rendered.

UCLA

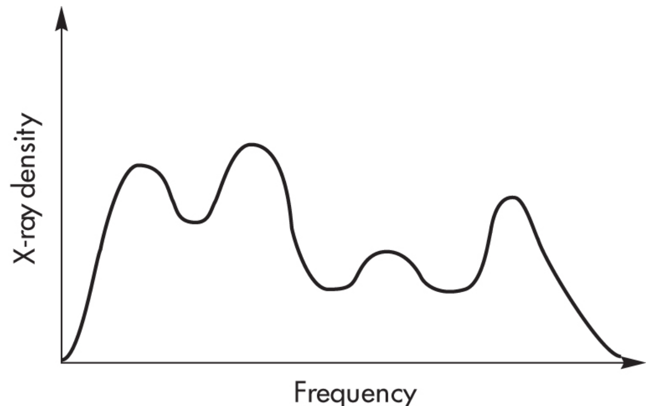
Volume Rendering

- Rendering – direct methods
    - Iso-surface rendering is useful, if you know the value specified will show you what you are looking for.
        - It is also pretty simple implementation-wise.
    - It only shows a single value, however, and the remainder of the data is ignored.
    - We would like a method that somehow uses contributions from all the data.
    - This requires that we form some kind of mapping of the scalar values to color and opacity.
        - Otherwise we will not know how each value contributes.

UCLA

Volume Rendering

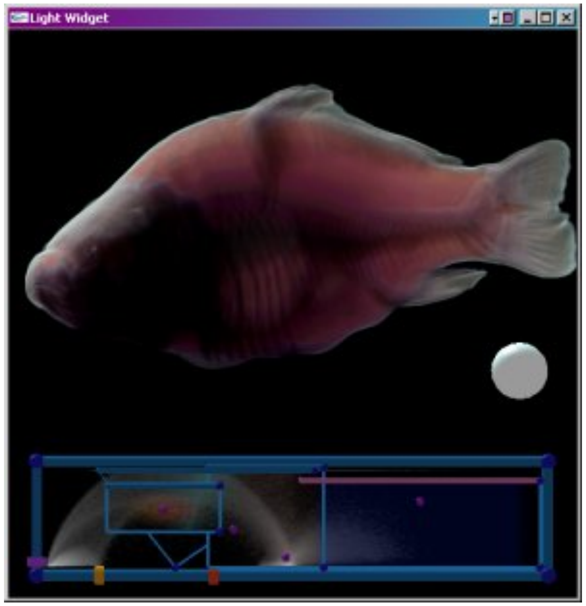
- Rendering – direct methods
  - Selecting colors and opacities is really more of an art than a science.
  - Information on the data can help to make these choices, however.
  - The peaks can inform elements that are interesting to call out.
    - Bone, soft tissue



UCLA

Volume Rendering

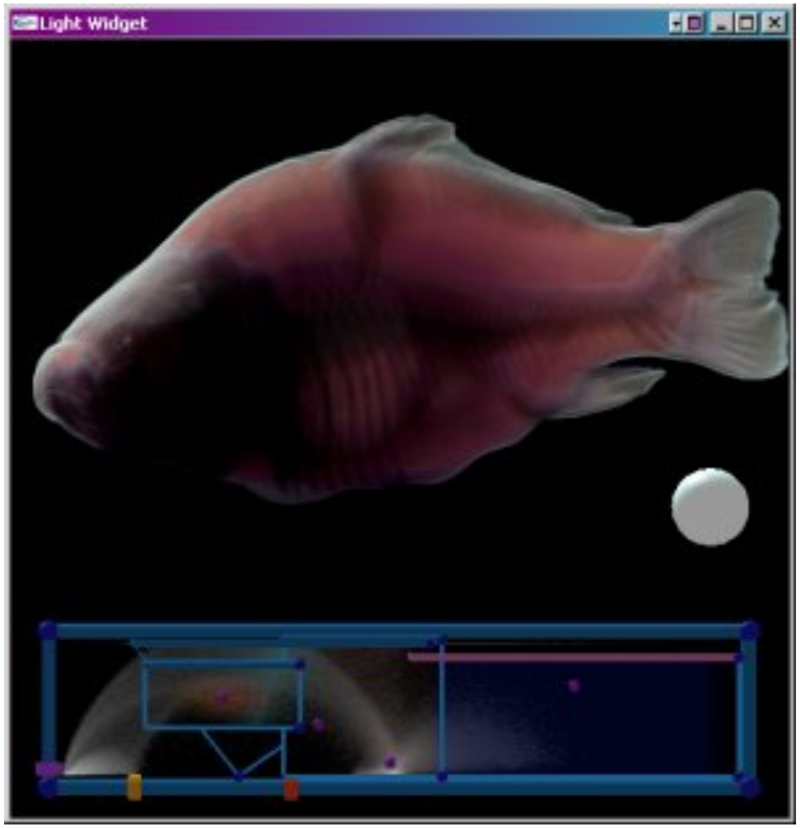
- Rendering – direct methods
  - Adjusting the opacity of any color helps to call out a particular structure within the volume.
  - Usually, setting these values is something a user does interactively.
  - Known as:
    - Transfer Function



UCLA

Volume Rendering

- Rendering – direct methods

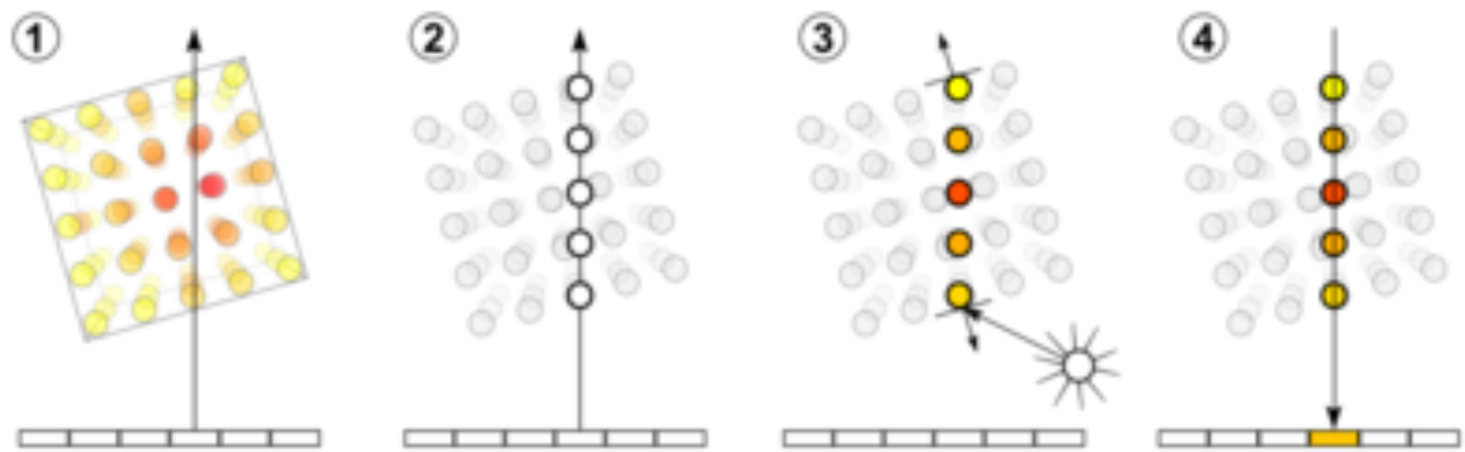


Volume Rendering

- Rendering – direct methods – splatting
  - Splatting is a simple method where a shape is assigned to all voxels.
  - The exact shape to use for a splat is a research topic
    - Ellipsoids are often used.
  - These shapes are projected and composited onto the viewing plane in back to front order.
    - So transparency/blending looks correct
  - That order is defined by the orientation of the data grid to the view point.
  - A spatial data structure such as an oct-tree can help with the sorting.

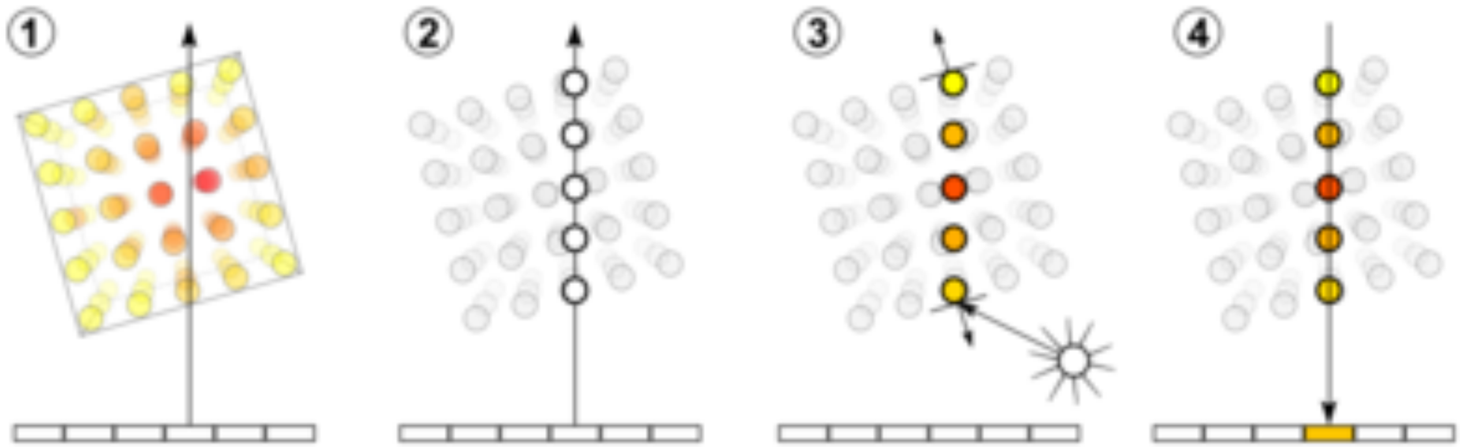
Volume Rendering

- Rendering – direct methods – ray casting
  - Ray casting is similar to the implicit method for iso-surface rendering in that we project a ray through the volume.
  - Values along the ray are tri-linearly interpolated to color and opacity values.



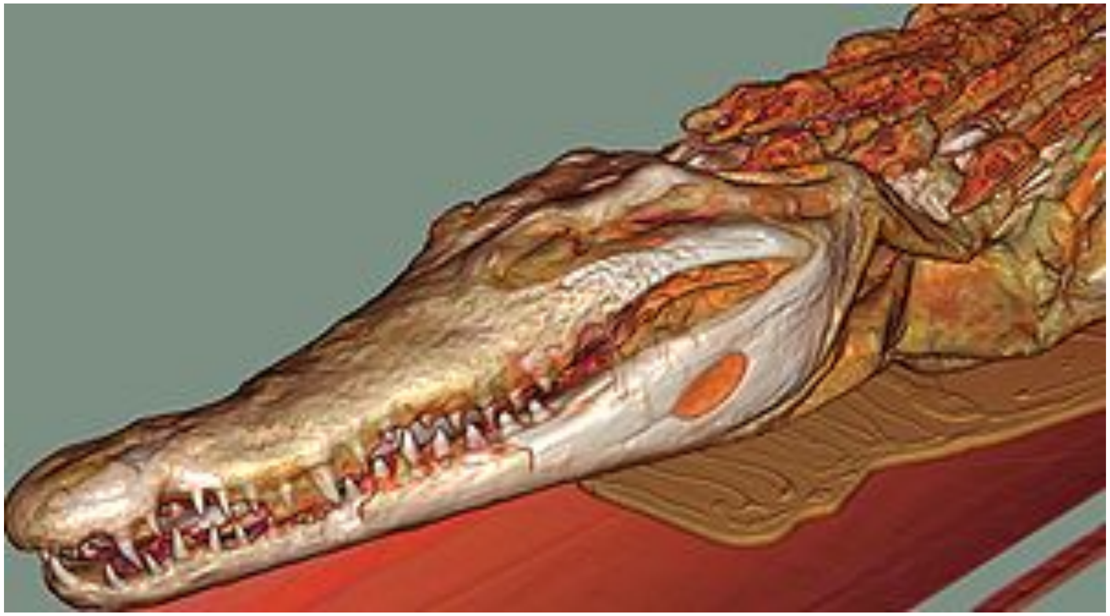
Volume Rendering

- Rendering – direct methods – ray casting
  - Each of the samples is lit normally using the gradient of the voxel with respect to the light source and viewer.
  - Finally, the samples are composited in back to from order to obtain a final pixel value.



Volume Rendering

- Rendering – direct methods – ray casting
    - Very high quality results can be obtained with this approach.
    - GPUs can be used to speed this technique up significantly.



UCLA

Volume Rendering

- Rendering – direct methods – textures
  - Recall during our discussion of texture mapping that OpenGL support 3d textures.
  - Here each ‘slide’ (or plane) of a data volume is stored in a texture map.
    - Colors and opacities are “baked” in to the images.
  - Applying **3d** texture coordinates to geometry allows arbitrary slices of the data set to be rendered.
  - Sampled values are tri-linearly interpolated from the texture data.

UCLA

Volume Rendering

- Rendering – direct methods – textures