Volume Rendering

Volume Rendering

Creating 2D images of volume data

Voxels (volume elements) typically stored in regular lattice

Voxel lattice represents 3D scalar field

Types of 3D Scalar Fields

X-ray absorbtion

Flow velocity (speed)

Temperature

Solidity (CSG, etc.)

Generating 3D Scalar Fields

- **May be measured or simulated**
- **Measured with CT scan or MRI**
- **CFD simulation of flow and temperature**
- **Sampling of CSG hierarchy**
- **Conversion from B-Rep to Solid**

Field Reconstruction

Tri-linear interpolation

Quadratic or cubic splines

Convolution with filter kernel

- **Each voxel's contribution to a point, p, measured by value in kernel, which is centered at p**

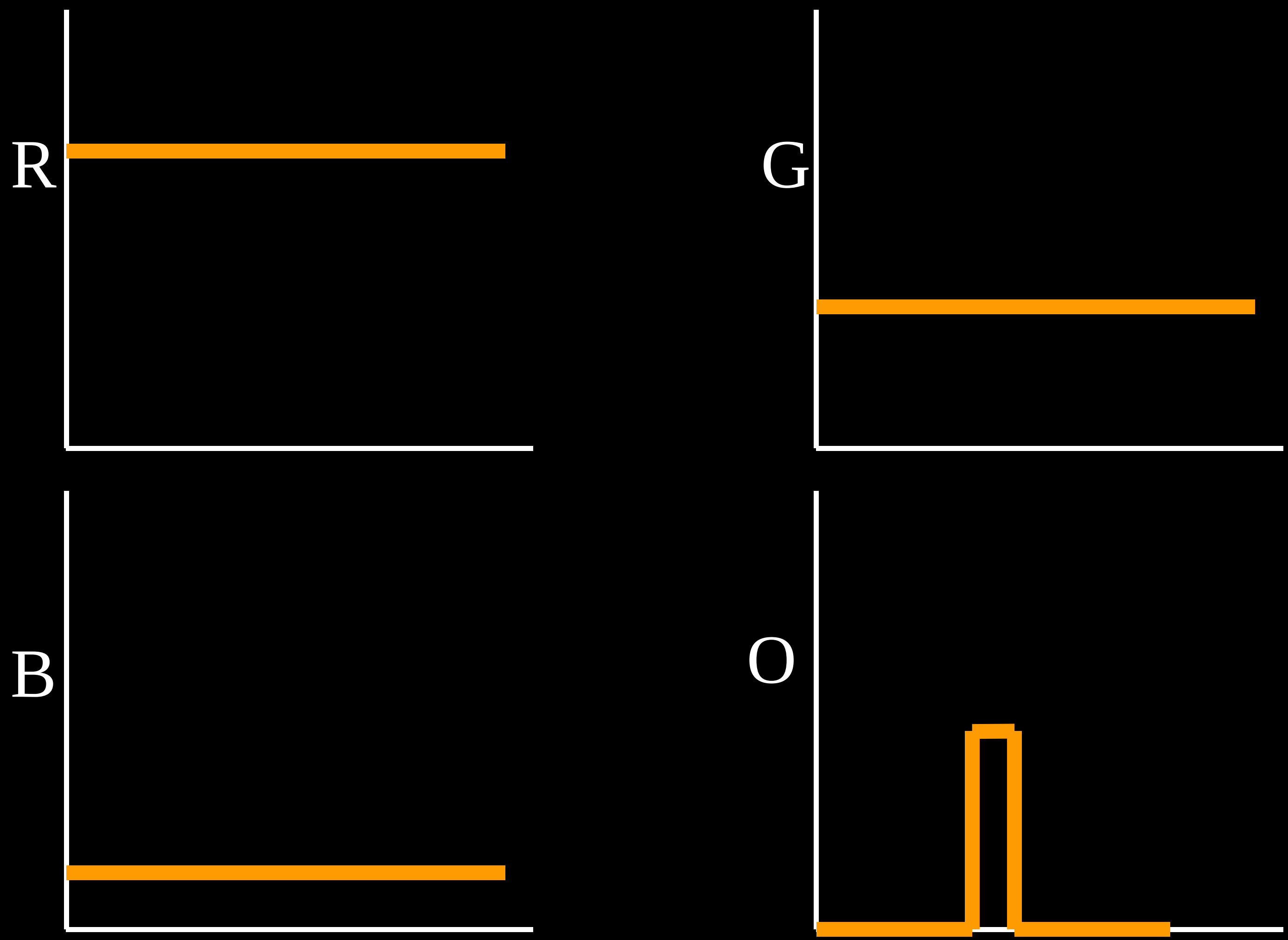
Mapping Values to Appearance

Often only a single value at each voxel

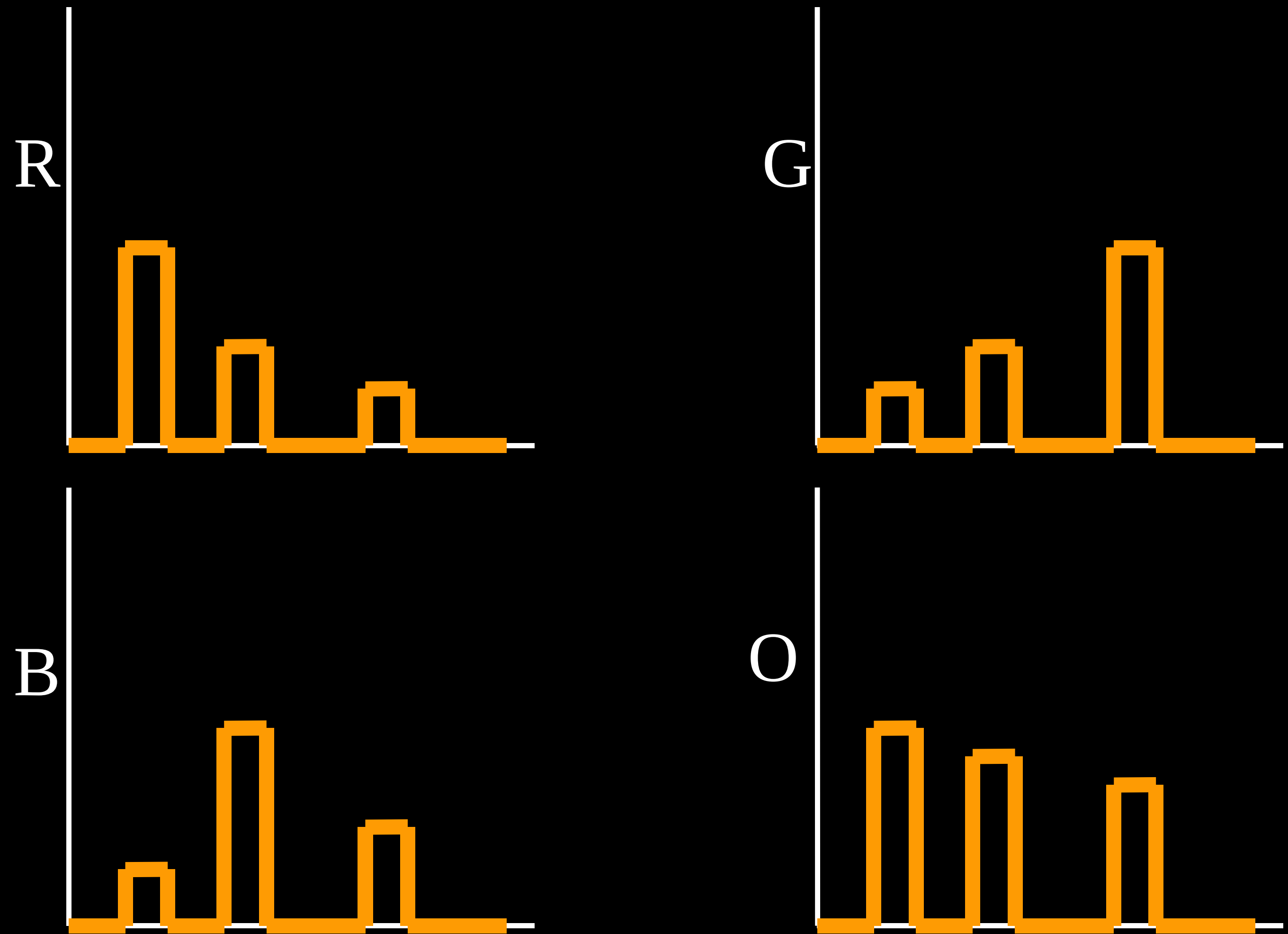
Want mapping to color and opacity

May emphasize certain value ranges or give all ranges equal emphasis in final image

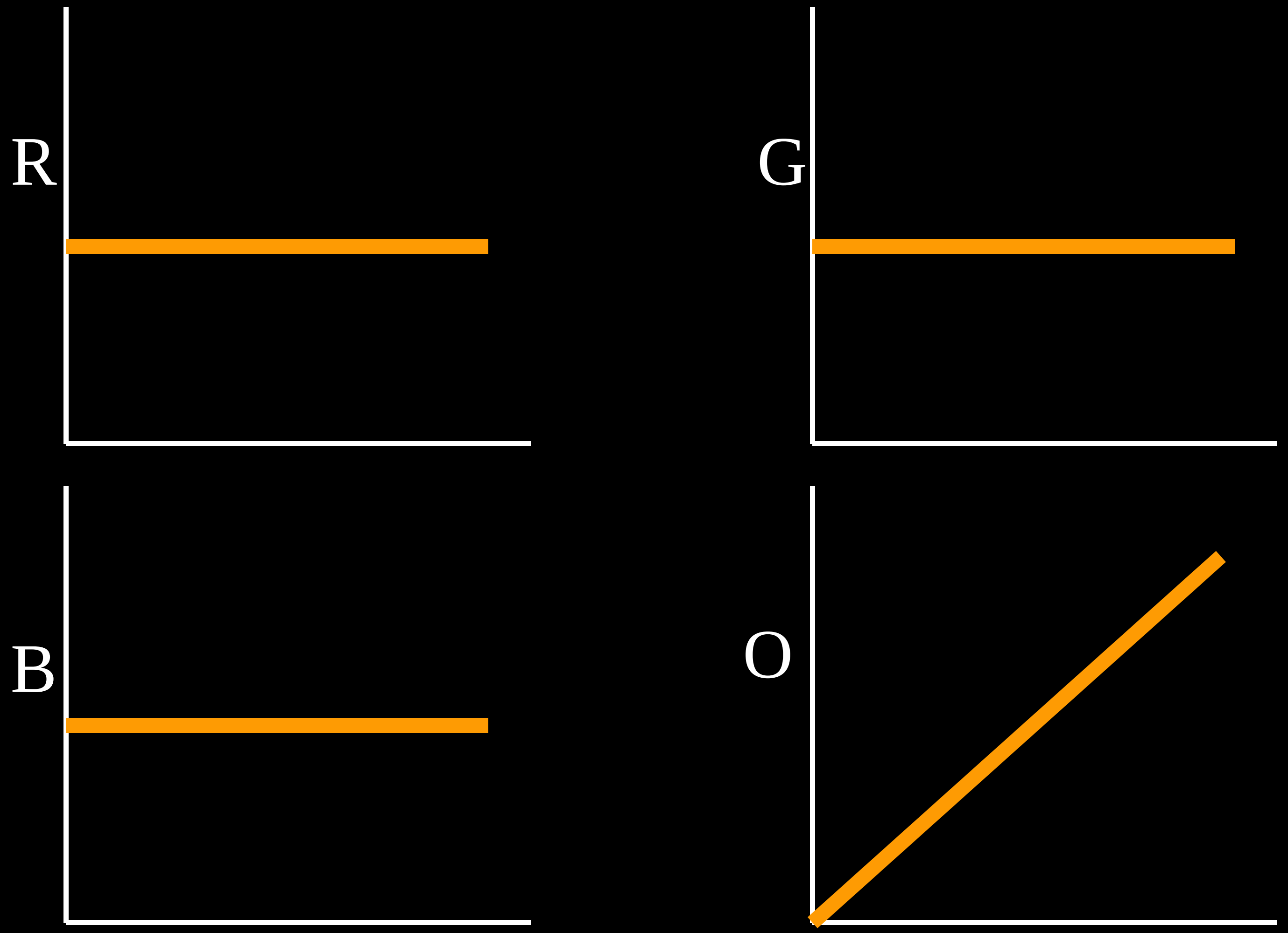
Emphasizing a Single Isosurface



Emphasizing 3 Isosurfaces



Visualizing Values as Opacity



Methods of Rendering

Solid texturing

Isosurface extraction

Image-space accumulation (ray casting)

Object-space accumulation (splatting)

Marching Cubes Algorithm

Performs isosurface extraction from voxel data

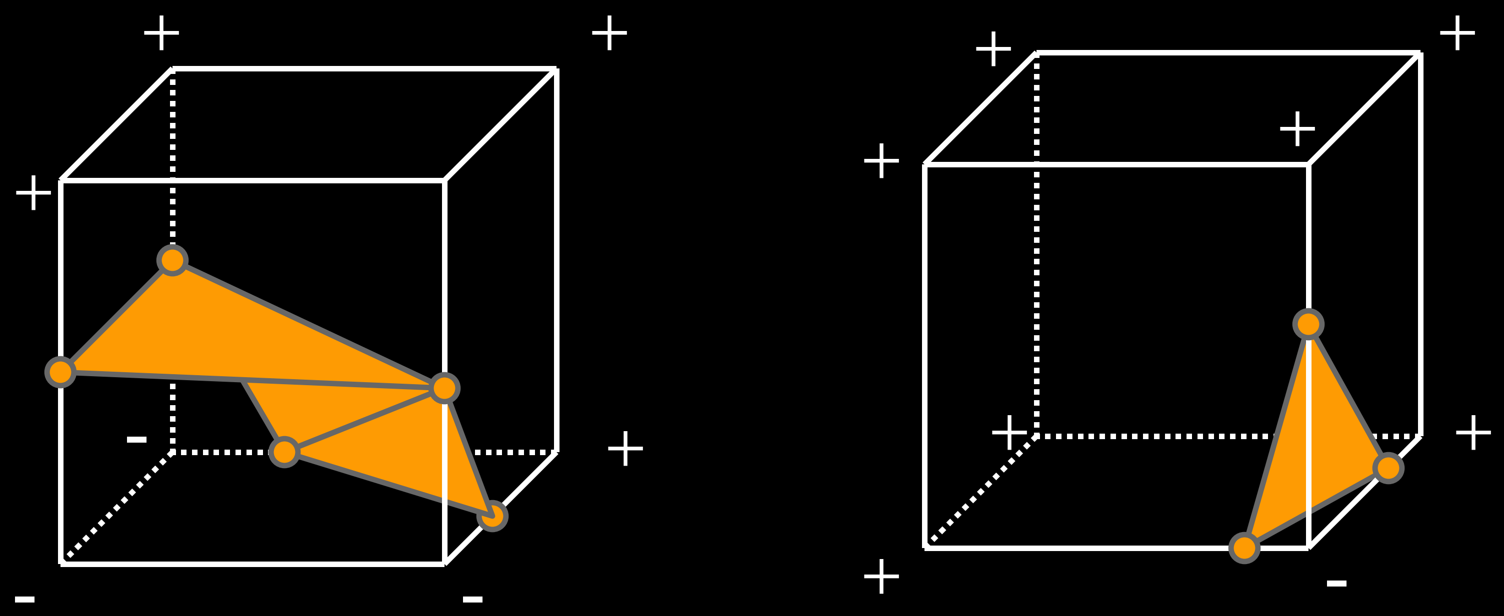
Creates a B-Rep, typically a set of triangles

B-Rep then rendering using standard rendering techniques

Finding/Creating the Surface

- Assume linear interpolation between data, stored at voxel corners
- Mark corners as inside or outside surface
- Find surface intersections along voxel edges
- Construct triangles connecting intersections

Marching Cubes Illustrations



- 2^8 = 256 possible configurations (only 16 unique)
- Store all cases in table, including number and connectivity of triangles
- Must connect properly along voxel sides for continuity

Rendering Isosurfaces

One or more isosurfaces may be rendered as totally opaque or partially transparent

Clipping planes may be used to illustrate interior surfaces

(see Figure 15 in Watt's *3D Computer Graphics*)

Volume Ray Casting

Loop over pixels, generating rays

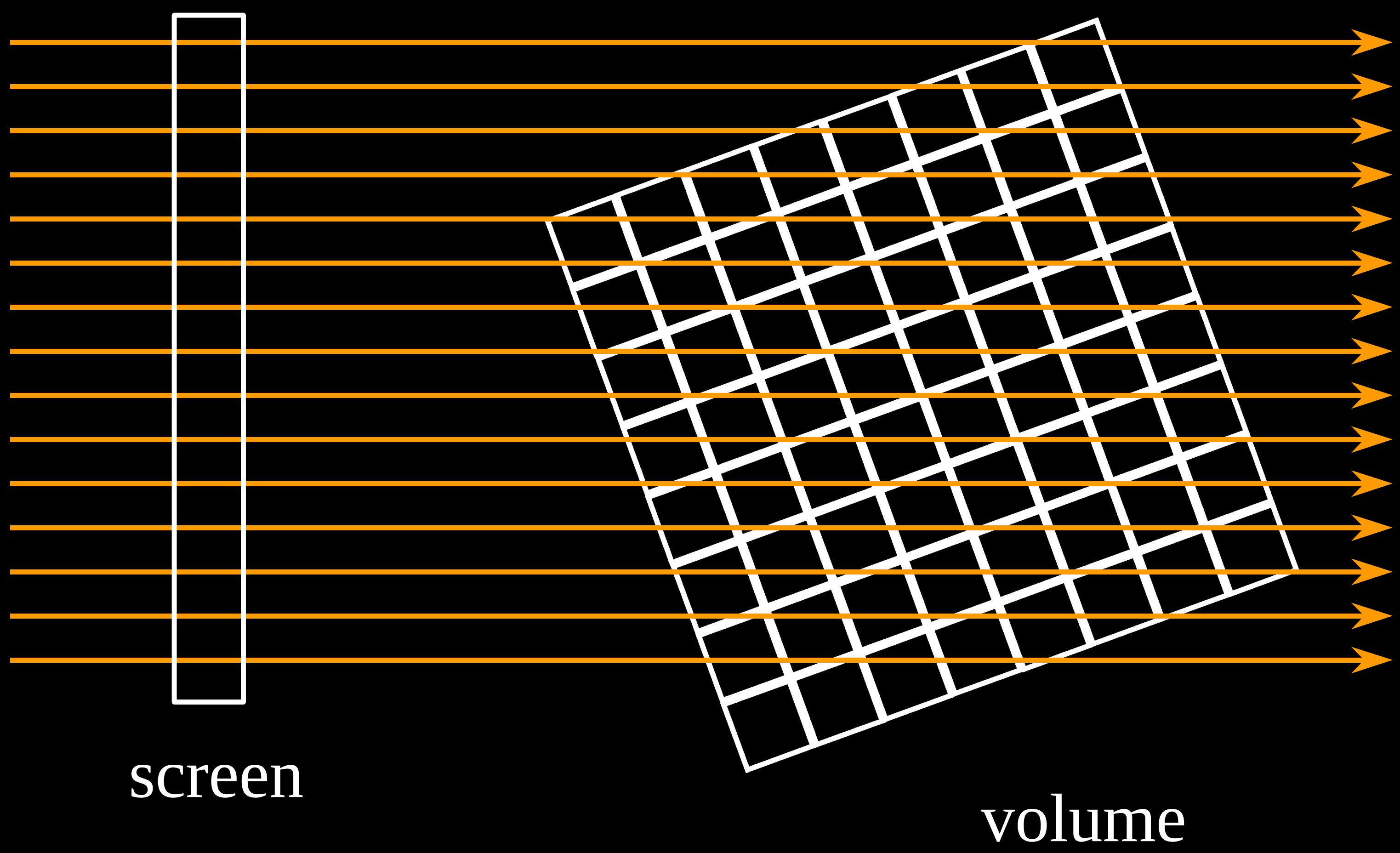
- **at least one per pixel, typically**

Trace each ray through the voxel grid

Accumulate color and opacity along ray

Stop when ray exits grid or reaches full opacity

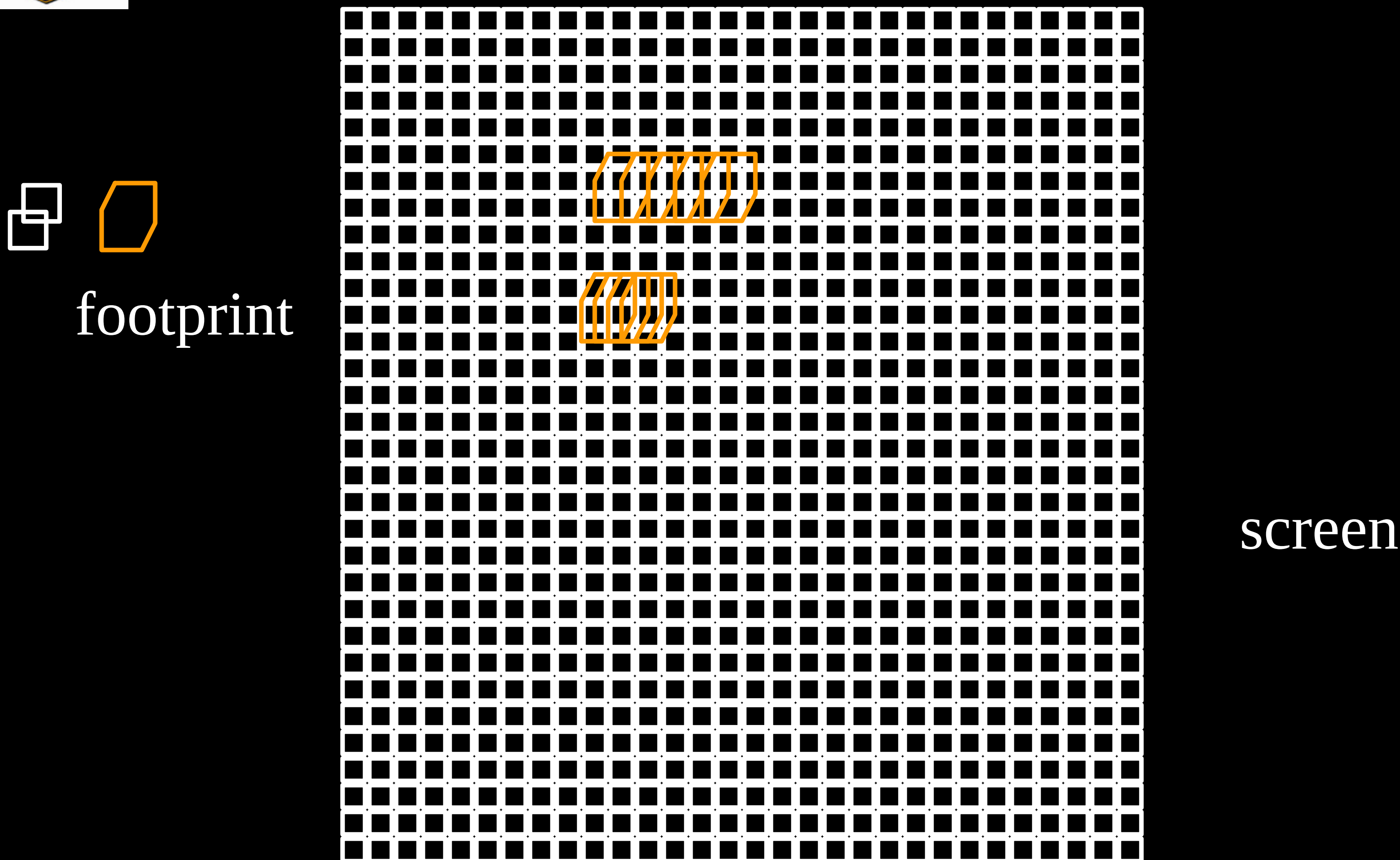
Ray Casting Illustration



Volume Splatting

- Traverse voxels in front to back order
  - traverse each voxel in plane, then move to next plane
- For each voxel, accumulate color and opacity to each pixel it covers
  - like throwing snowballs at the screen
- Voxel projection covers hexagonal footprint
- Smoother interpolation possible by applying kernel with fall-off away from sample point

Volume Splatting Illustration



Ray Casting vs. Splatting

Ray Casting

- **Point samples**
- **Random data access**
- **Easy for parallel or perspective projection**

Splatting

- **Area samples**
- **Ordered data access**
- **More difficult for perspective projection**

Color/Opacity Accumulation

Assume that each voxel emits a single color and filters colors by its opacity

$$C' = C * \alpha$$

$$C'_{out} = C'_{in} + C'_{voxel} * (1-\alpha_{in})$$

$$\alpha_{out} = \alpha_{in} + \alpha_{voxel} * (1-\alpha_{in})$$

$$C_{out} = C'_{out} / \alpha_{out}$$

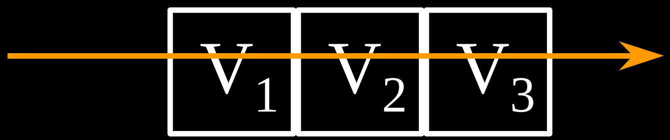
Accumulation Example

Suppose ray pases through 3 voxels (r,g,b,a):

$V_1 = (.3, .1, .1, .2)$

$V_2 = (.1, .3, .1, .3)$

$V_3 = (.1, .3, .1, .3)$



Premultiply colors by opacity:

$V_1' = (.06, .02, .02, .2)$

$V_2' = (.03, .09, .03, .3)$

$V_3' = (.03, .09, .03, .3)$

Accumulation Example (cont.)

- After passing through V_1, we have
  - (0,0,0,0) + (.06,.02,.02,.2)*(1-0) = (.06,.06,.02,.2)
- After passing through V_2, we have
  - (.06,.02,.02,.2) + (.03,.09,.03,.3)*(1-.2) = (.084,.092,.044,.44)
- After passing through V_3, we have
  - (.084,.092,.044,.44) + (.03,.09,.03,.3)*(1-.44) = (.1008,.1424,.0608,.608)
- Dividing by the final alpha, we get
  - (.17, .23, .1, 1)

Volume Illumination

Several possible models

- identify surfaces within voxels
- allow not only voxel emission, but attenuation of incoming light and surface reflection
- model as particle clouds of various densities

Drebin et al. 98 models voxels as mixtures of materials

- all measurements continuous, not discrete
- measure surface "strength" based on differences in material densities
- measure surface normals as value gradients

Parallel vs. Perspective Projection

Parallel

- **Even sampling**
- **Regular access**
- **Simple footprints**

Perspective

- **Uneven sampling**
- **Irregular access**
- **Complex footprints**