CS174A Lecture 16

Announcements & Reminders

- ***Dec 1: projects due***
- ***Dec 3 and 5: project presentations, in class***
  - We will post some guidelines about the order of presentations, how to gather scores from teammates, class, etc.
  - Test your laptops, adapters, etc. beforehand with class projector
- ***Dec 12 (Thursday): final exam***
  Time: 11:30 AM – 2:30 PM
  Place: Kinsey Pavilion 1220B
- ***Course evaluations: Nov 28 – Dec 7***

TA Session This Friday

- *HAPPY THANKSGIVING: no TA session this Friday*

Next Week:

- *Team projects & logistics*
- *Final exam review*

Last Lecture Recap

- ***Ray Tracing***
  - Issues: speed, shadows, aliasing
  - Stochastic ray tracing
- ***Transparent Objects***
- ***Particle Rendering***
- ***Prof Demetri: Biometric Human Simulation***

Next Up

- *Volume Rendering*
- *Aliasing/Anti-Aliasing*

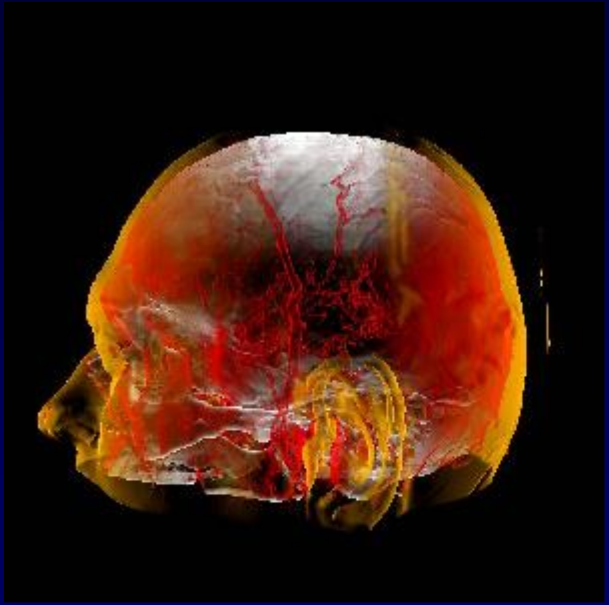
Prof Demetri Terzopoulos

Demetri Terzopoulos is Distinguished Professor and Chancellor's Professor of Computer Science at the University of California, Los Angeles, where he directs the UCLA Computer Graphics & Vision Laboratory. He is also Co-Founder and Chief Scientist of VoxelCloud, Inc. He is or was a Guggenheim Fellow, a Fellow of the Association for Computing Machinery (ACM), a Fellow of the Institute of Electrical and Electronics Engineers (IEEE), a Fellow of the Royal Society of London, a Fellow of the Royal Society of Canada (RSC), and a member of the European Academy of Sciences (EAS), the New York Academy of Sciences (NYAS), and Sigma Xi.

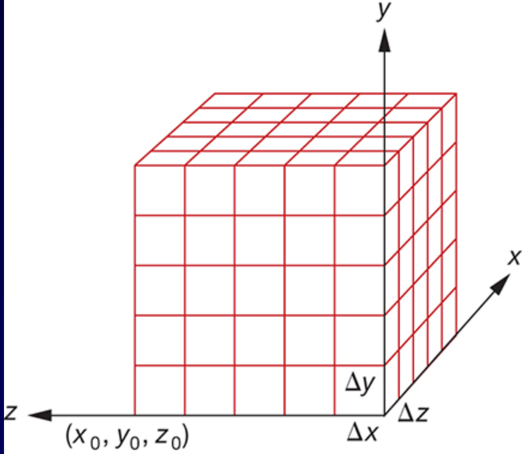
Prof Demetri Bio

Volume Rendering

- *2D projection of 3D sampled dataset*
- *Volume rendering algorithms:*
  - Usually no illumination or shadows, just compositing
  - Therefore only ray-casting, no recursive ray-tracing
  - No perspective, only parallel projection



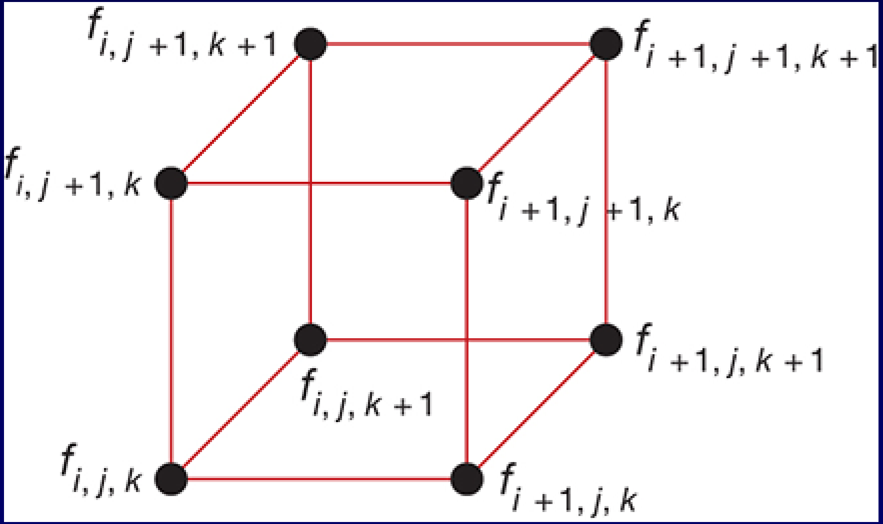
Volume Rendering: Voxels

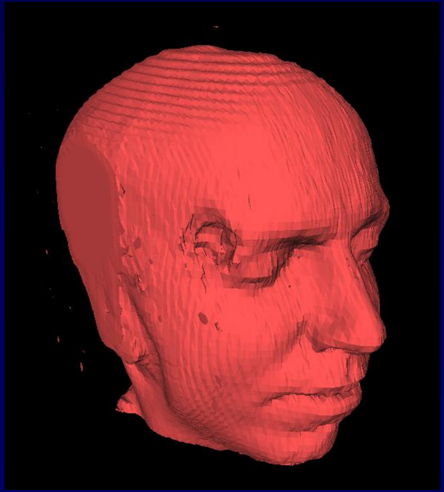


- *Volume dataset: 3D regular grid of voxels*
- *Voxel: a small cube at i,j,k with sides Δx,Δy,Δz*
- *Each grid point has a scalar value f(x,y,z)*
- *For example, density, intensity, CT scan, MRI*
- *Voxelize more complex implicit surfaces*
- *If $\Delta x = \Delta y = \Delta z \Rightarrow$ structured volume dataset*
- *Transfer function: to map lattice scalar values to RGBA*
- *Based on viewer location, there's a natural ordering of voxels*
- *Composite front-to-back or back-to-front*

Volume Rendering: Marching Cubes

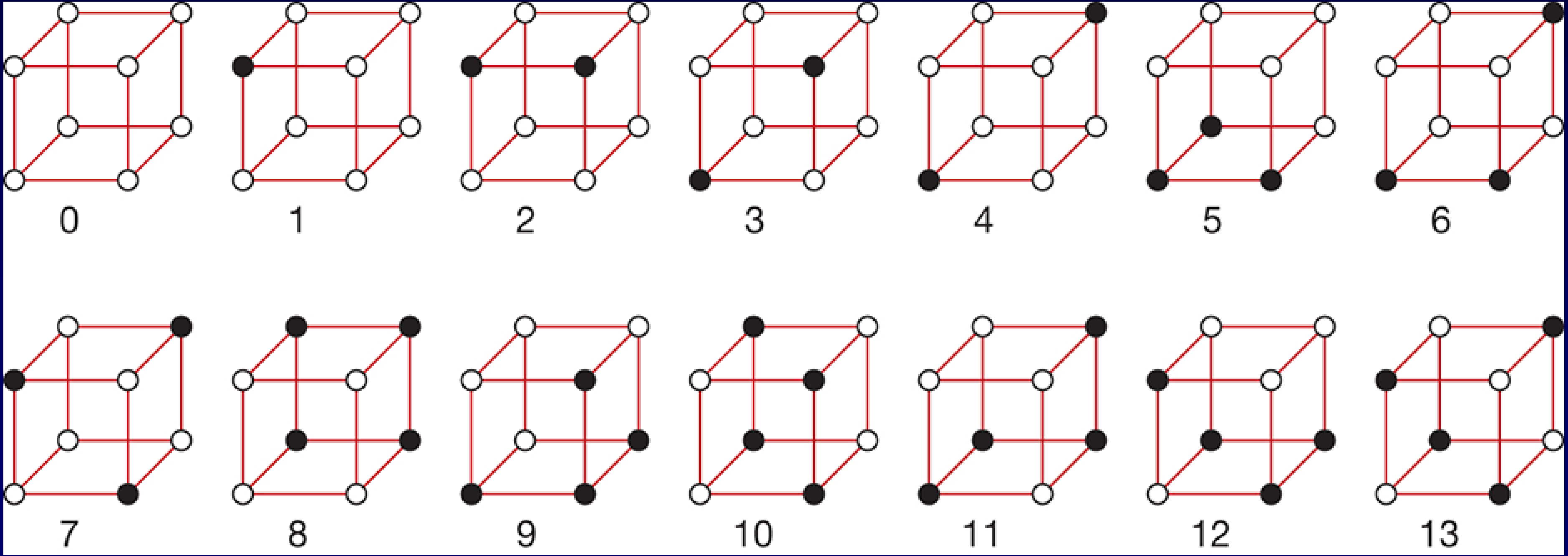
- *Object based volume rendering technique*
- *Create poly mesh by extracting isosurfaces: $f_{i,j,k} = c$*
- *Color vertices: if $f_{i,j,k} < c$, then white, else black*





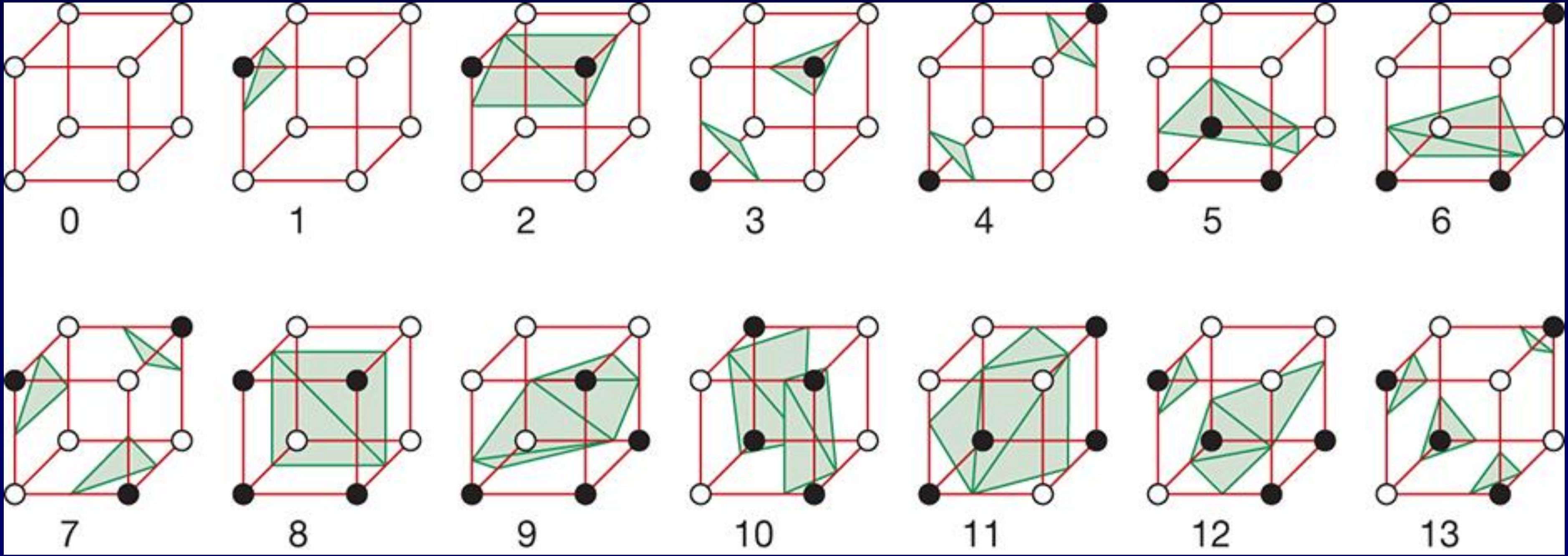
Volume Rendering: Marching Cubes

- *Total 2^8 = 256, but only 14 unique cases*



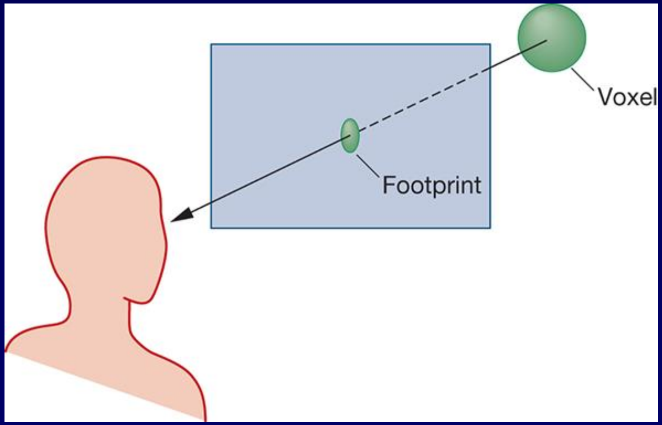
Volume Rendering: Marching Cubes

- *Once volume is polygonised, GPU can be used to render*



Volume Rendering: Splatting

- ***Object based volume rendering technique***
- ***Each volume element (voxel) is splatted on screen as a snowball***
- ***Voxels splatted in BTF order wrt to the viewer***
- ***Splats are rendered and composited as disks on the screen***
- ***Circular, ellipsoidal or Gaussian splats***

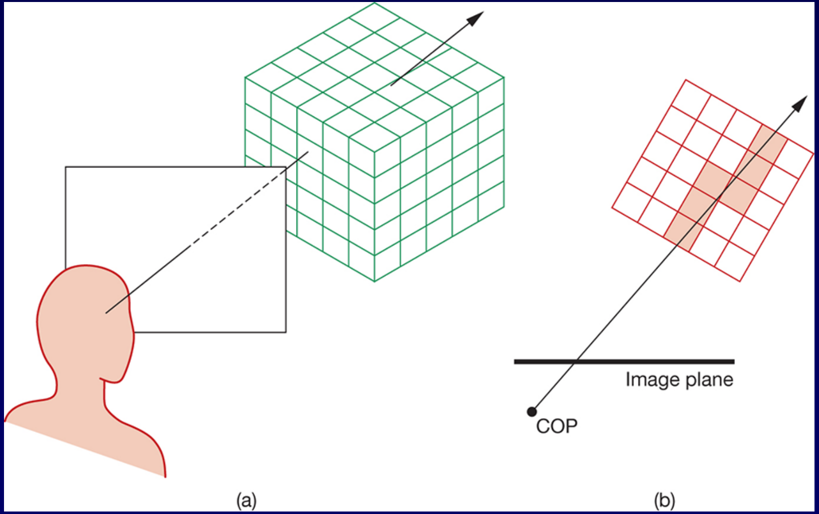


Volume Rendering: V-Buffer

- ***Image based volume rendering technique***
- ***Ray casting through volume***
- ***Trilinear interpolation to determine RGBA at non-lattice point***
- ***Accumulate color and opacity***
- ***3 levels of sampling:***
  - Voxel lattice: $x_{i,j,k}$
  - Sampling along ray: y_i
  - Image plane: $z_{i,j}$
- ***2 pipelines (color/opacity):*** $c = c_1 + c_2(1 - \alpha_1)$; $\alpha = \alpha_1 + \alpha_2(1 - \alpha_1)$

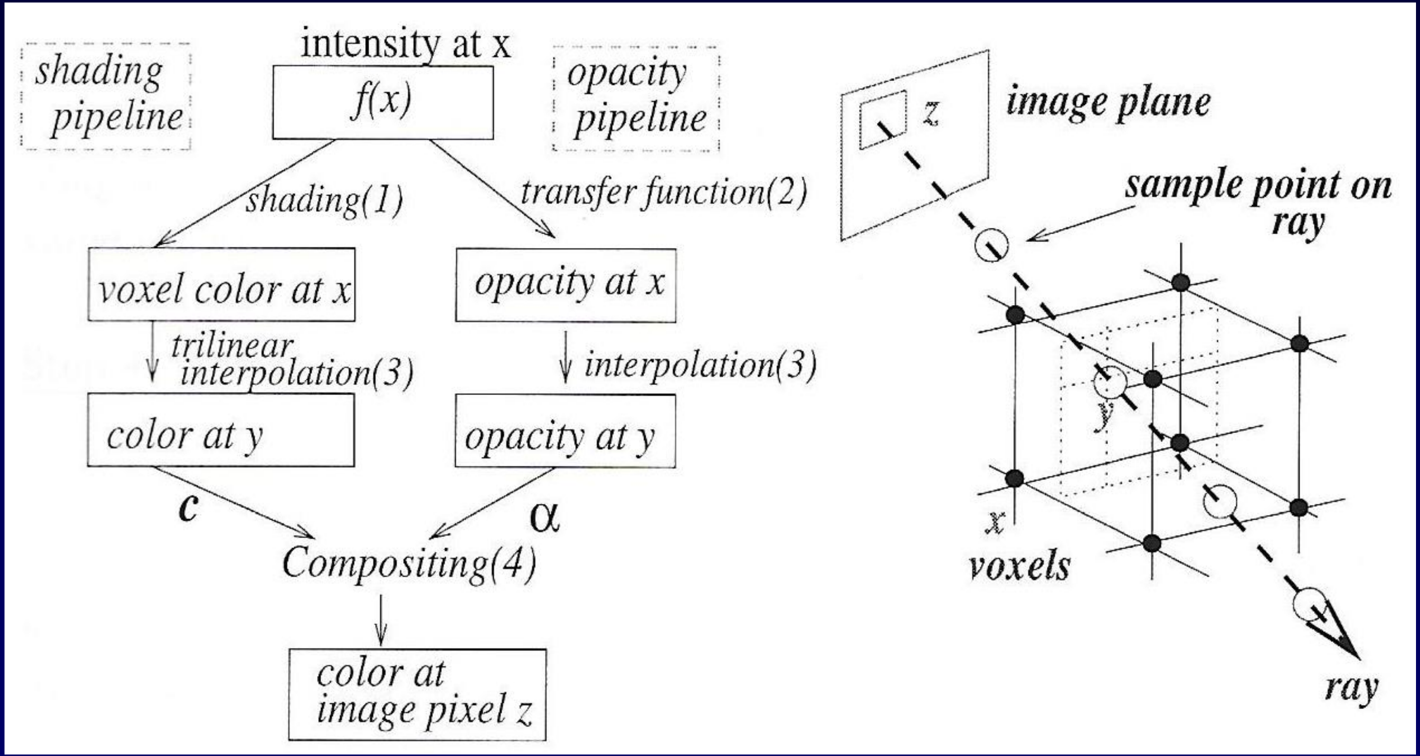
Volume Rendering: V-Buffer

- ***Ray casting through volume***
- ***Parametric eqn of ray: p + t*d***
  - p: pixel location
  - d: ray direction
  - t: parameter along ray
- ***Step through ray by incrementing t***



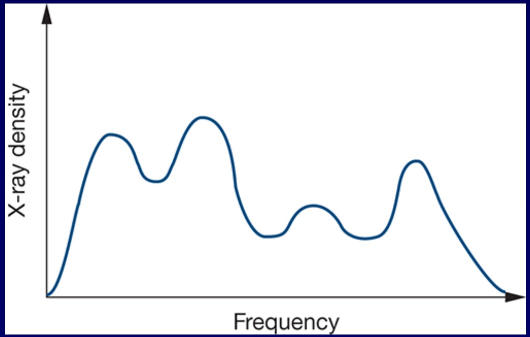
Volume Rendering

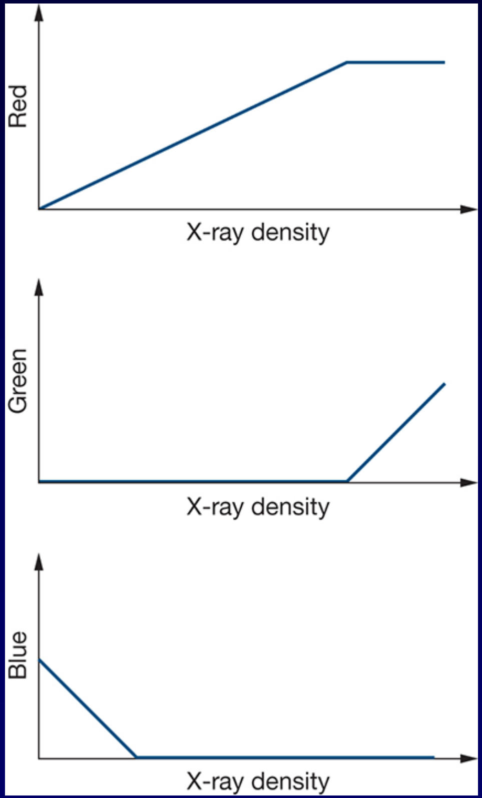
Algorithm



Volume Rendering: Transfer Function

- *X-Ray density data for each voxel*
- *Assign different color to each peak in histogram*
- *Opacity values based on emphasis*



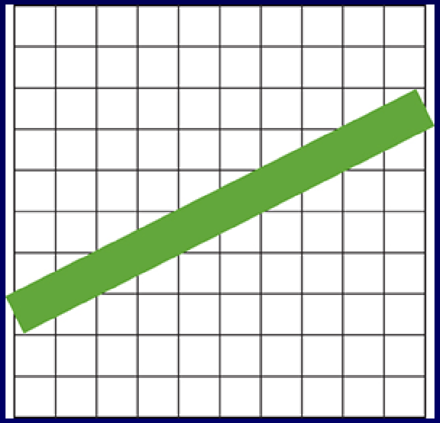


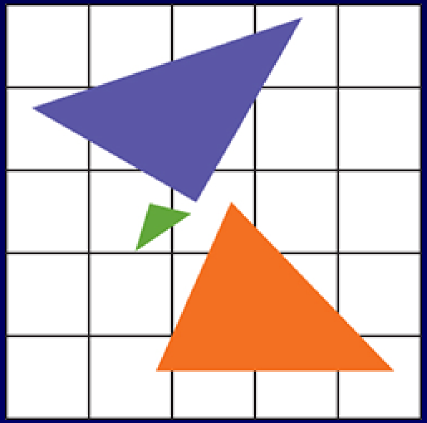
Volume Rendering: V-Buffer Speedups

- ***Early ray termination***
  - Stop when opacity reaches 1
  - Or when ray exits volume
- ***Empty space skipping***
- ***Octree or BSP trees***
- ***Temporal use of voxels***

Aliasing: Rasterization

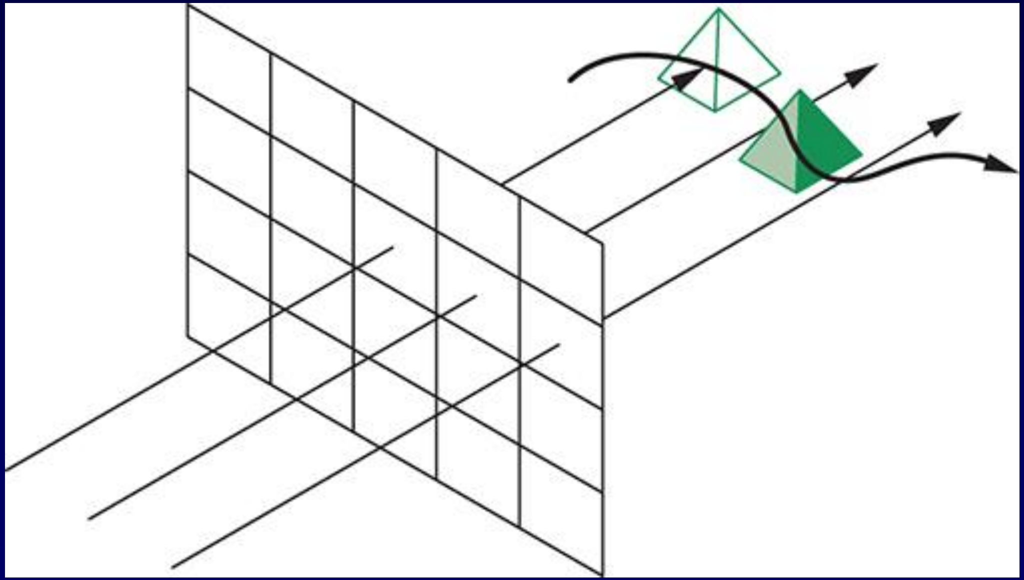
- ***Spatial aliasing in CG***
  - Jagged lines while rasterization
  - Going from continuous representation to a sampled approximation, which has limited resolution
  - Pixels on screen have fixed number, size, shape and location





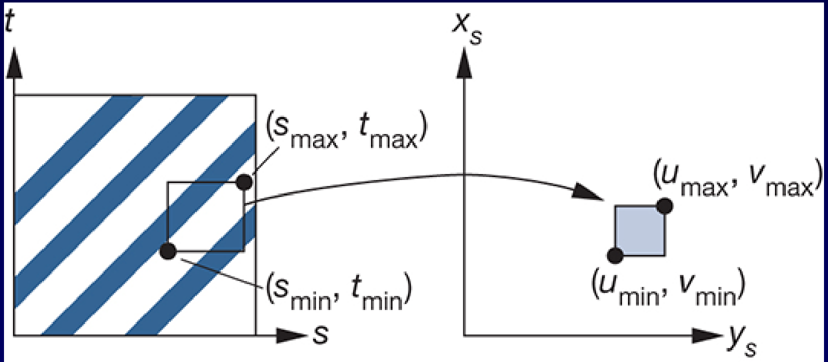
Aliasing: Temporal

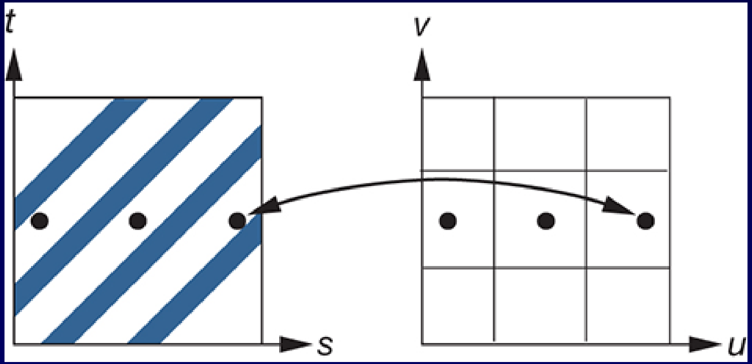
- *Temporal aliasing in CG*



Aliasing: Mappings

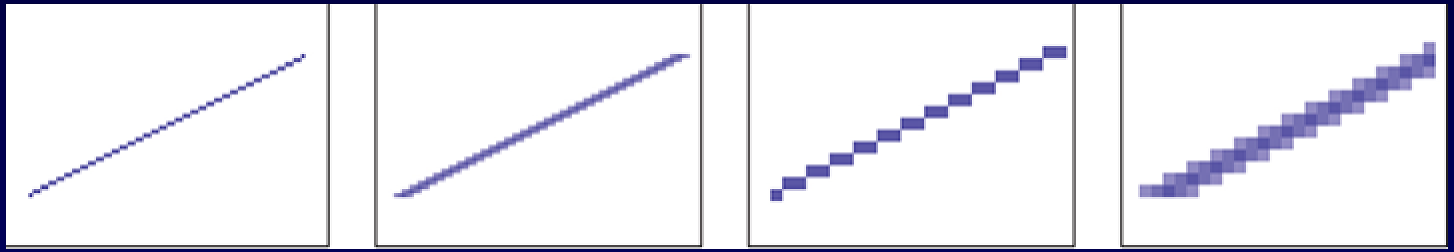
- *Due to high-frequency patterns*





Anti-Aliasing

- *Area averaging*



- *Super-sampling, then averaging or blending*
- *In h/w, use super-sampled offline buffer, then average to frame buffer*