CS174A Lecture 16

Announcements & Reminders

- ***Final exam study guide and book questions posted in Canvas***
- ***Confirm that your grades in Canvas are accurate***
- ***~~11/22/22~~ 11/28/22: Team project proposals due, final version***
- ***11/24/22-12/03/22: Student evaluations of course/instructors/TAs***
- ***11/29/22: Prof Demetri's talk***
- ***12/02/22 (Discussion Sessions): Team project presentations***
- ***12/05/22-12/06/22: Office hours for final exam, see Canvas***
- ***12/06/22: Final Exam, 6:30-8:30 PM PST, in class, in person***

Last Lecture Recap

- ***Ray Tracing***
  - Issues: speed, shadows, aliasing
  - Stochastic ray tracing

Next Up

- *Transparent Objects, Compositing*
- *Particle Rendering*
- *Volume Rendering*
- *Aliasing/Anti-Aliasing*

Transparency/Opacity

- ***Matte: coverage info***
- ***Add a 4th channel to color: α = opacity (RGBA), range [0..1]***
- ***α = 0 ⇒ fully transparent; α = 1 ⇒ fully opaque***
- ***Transparency = 1 – Opacity***
- ***Applications***
  - Image compositing, e.g., combining computer-rendered images with live footage
  - Particle rendering, e.g., smoke, snow, fire
  - Volume rendering

Transparency/Opacity: Blending

- ***Pre-multiplied vs straight alpha***
- ***Operators: over, in, out, atop, xor***
- ***Alpha blending or alpha compositing (over operator)***
  - Straight

    $$C_0 = \frac{C_f\alpha_f + C_b\alpha_b(1 - \alpha_f)}{\alpha_f + \alpha_b(1 - \alpha_f)}$$
  - Pre-multiplied

    $$c_0 = c_f + c_b(1 - \alpha_f)$$
    $$\alpha_0 = \alpha_f + \alpha_b(1 - \alpha_f)$$

Transparency/Opacity

- ***Example: Water Flowmap***
- ***Other good examples:***

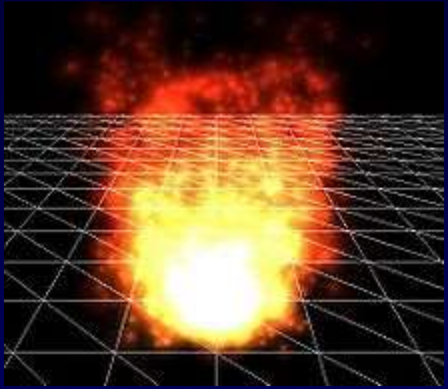
  marchingcubes, translucency, water / flowmap, lod, nearestneighbour, youtube, orientation / transform (quaternion math), *reflectivity, manualmipmap, multiple elements, shadowmap viewer*

Procedural Models

- ***For modeling cloud, smoke, water, crowd, flock: using behavior***
- ***Describe objects in an algorithmic manner (e.g., spheres and ellipsoids), generate polys only when necessary***
- ***Combine computer graphics with physical laws or for modeling solid objects***
  - Particle systems obeying Newton's Laws, e.g., fireworks, smoke, flame
  - Language based models for natural objects, e.g., plants, snowflakes
  - Fractal geometry for natural phenomenon, using level of detail, e.g., mountains
  - Procedural noise for realistic motion, e.g., clouds, fluid motion

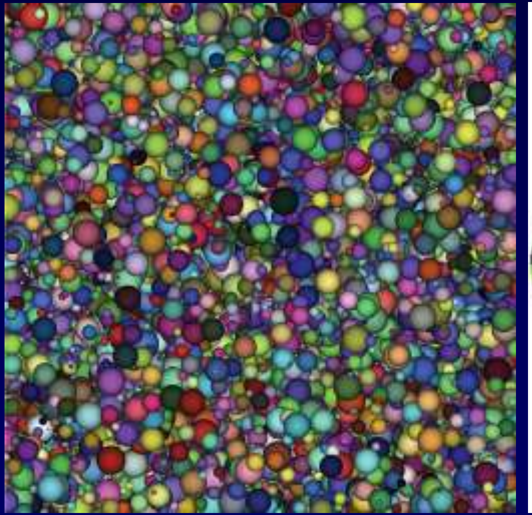
Particle Modeling

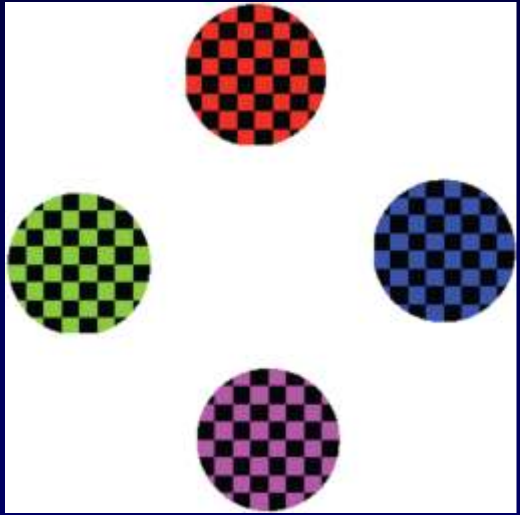
- *Used for simulating fuzzy phenomena*
- *Examples: chaotic, natural or chemical systems*
- *Like fire, smoke, explosions, snow, dust, etc.*
- *Modeled as an emitter, like sphere or box*
- *Particle behavior params*
  - Spawning rate
  - Initial velocity vector
  - Lifetime
  - Color, etc.
  - Collision?



Particle Rendering

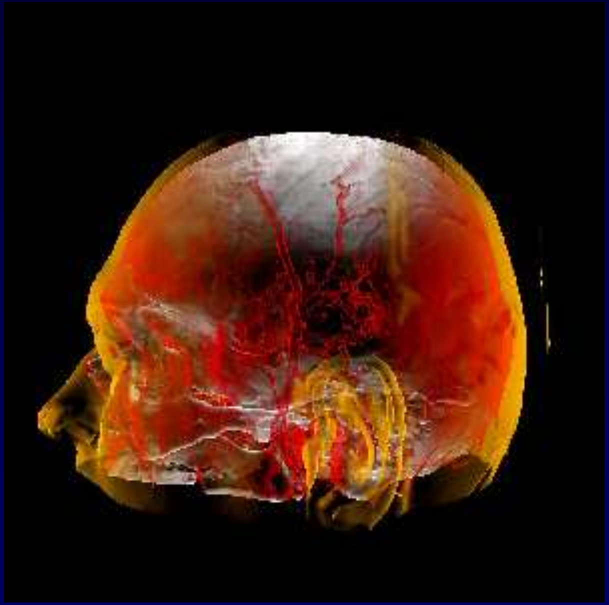
- ***Rendered usually as textured bill-boarded quads***
  - Sprites vs billboards
  - Integrate with z-buffer
- ***In games, as a single pixel***
- ***Examples:***
  - Particles & Billboards
  - Interactive 3D Graphics



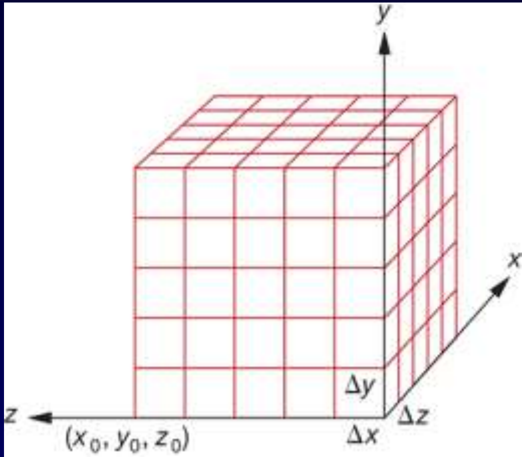


Volume Rendering

- ***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



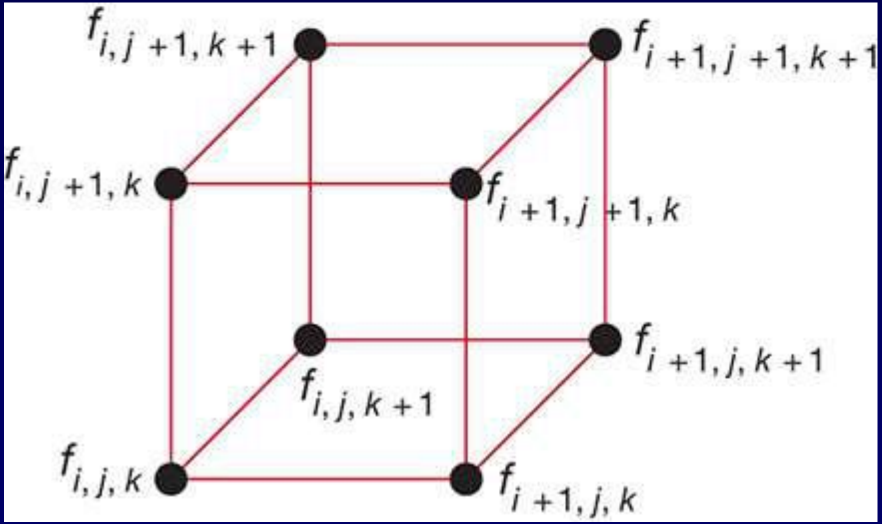
Volume Rendering: Voxels

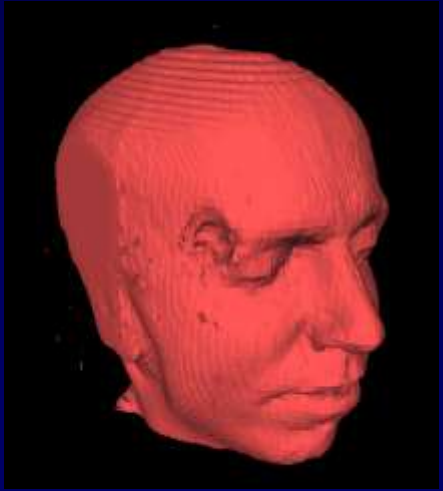


- *Volume dataset: 3D regular grid of voxels*
- *Voxel: a small cube at i,j,k with sides Δx,Δy,Δz*
- *Each grid point has a scalar value f(x,y,z)*
- *For example, density, intensity, CT scan, MRI*
- *Voxelize more complex implicit surfaces*
- *If Δx = Δy = Δz ⇒ structured volume dataset*
- *Transfer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

Volume Rendering: Marching Cubes

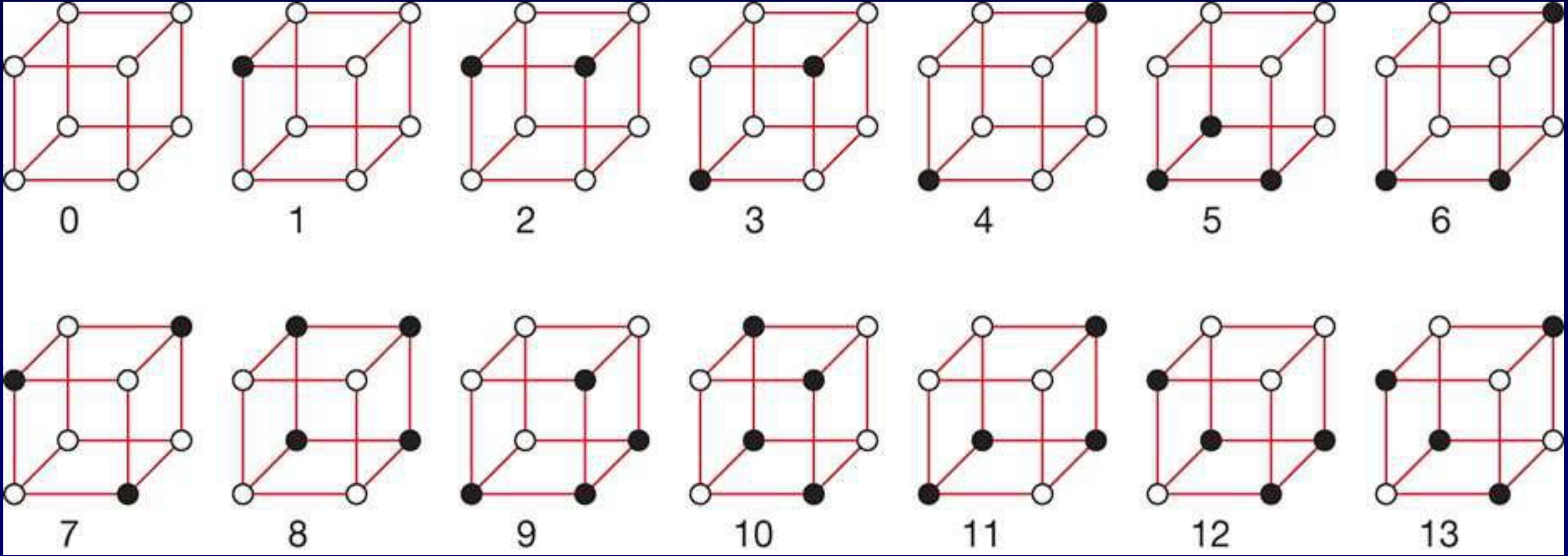
- ***Object based volume rendering technique***
- ***Create poly mesh by extracting iso-surfaces: $f_{i,j,k} = c$***
- ***Color vertices: if $f_{i,j,k} < c$, then white, else black***





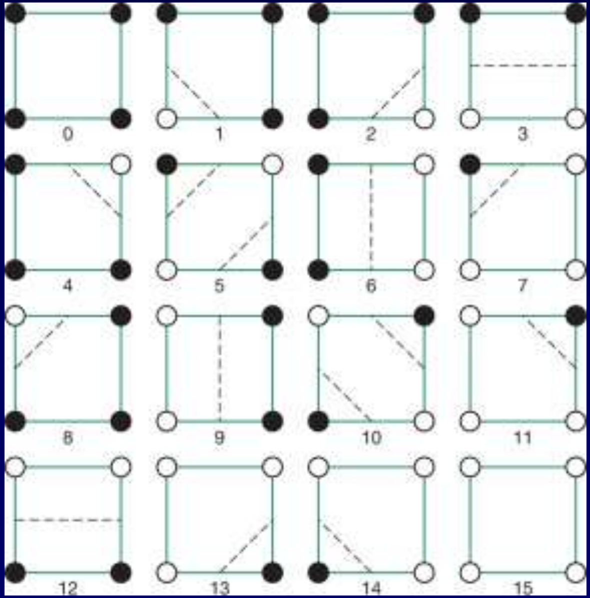
Volume Rendering: Marching Cubes

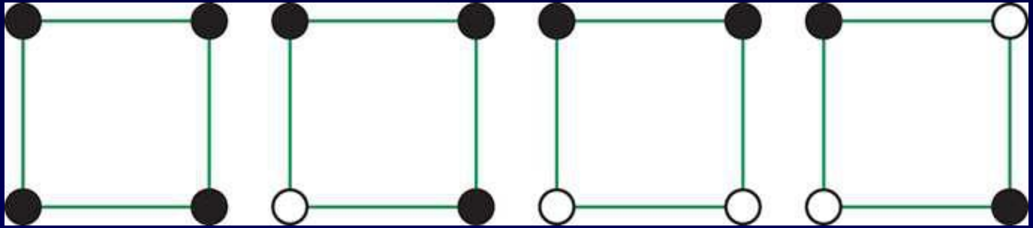
- ***Total $2^8 = 256$, but only 14 unique cases***

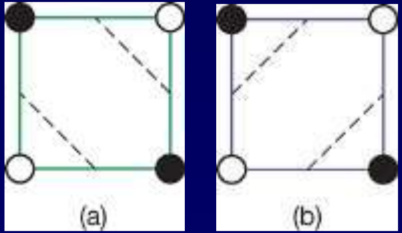


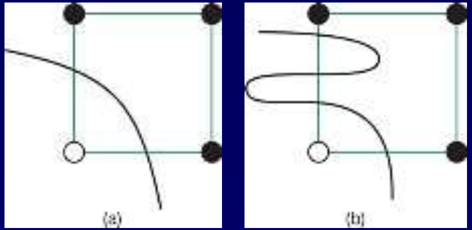
Example Using Squares

- *16 cases of vertex labeling, but only 4 unique cases*
- *Ambiguous cases*



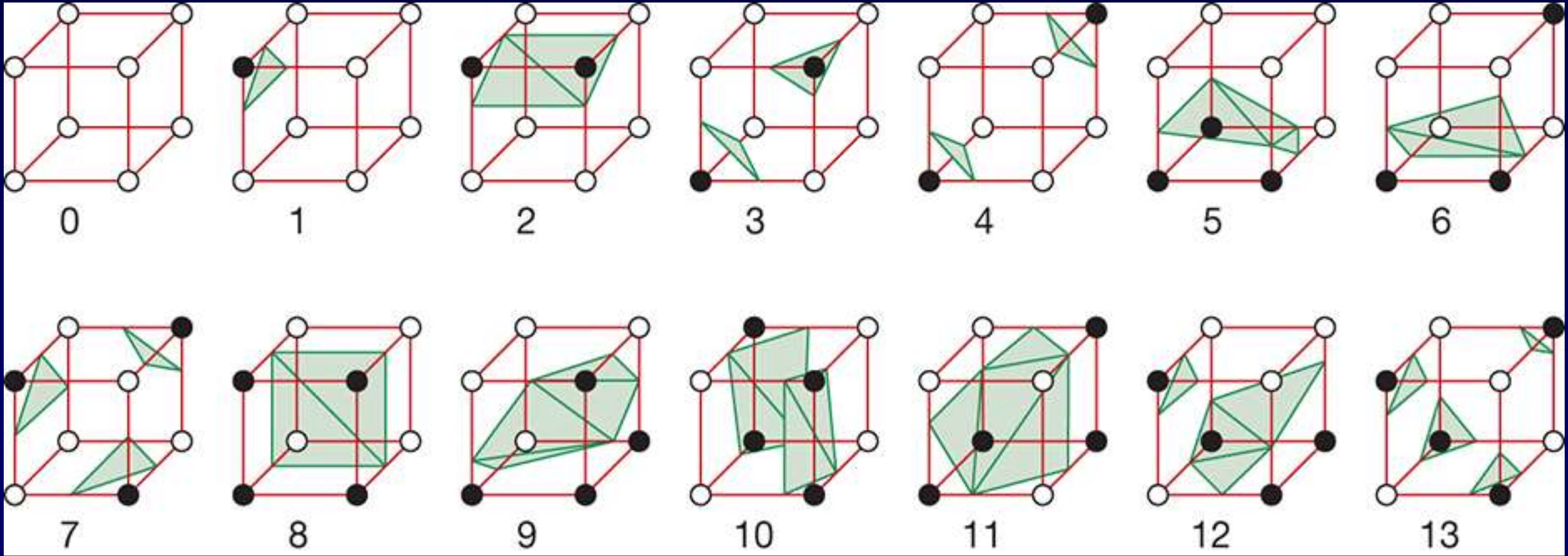






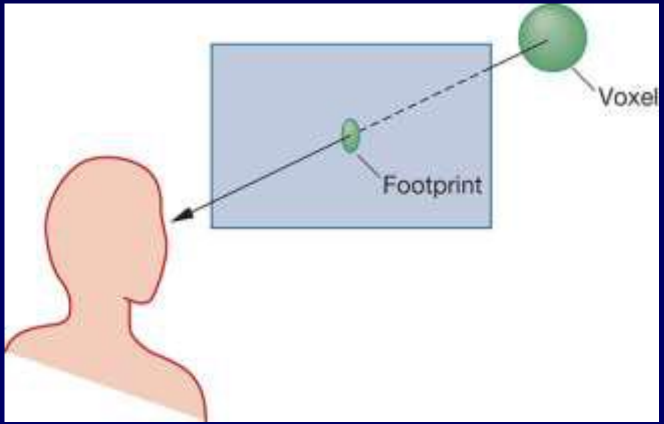
Volume Rendering: Marching Cubes

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



Volume Rendering: Splatting

- *Object-resolution 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*

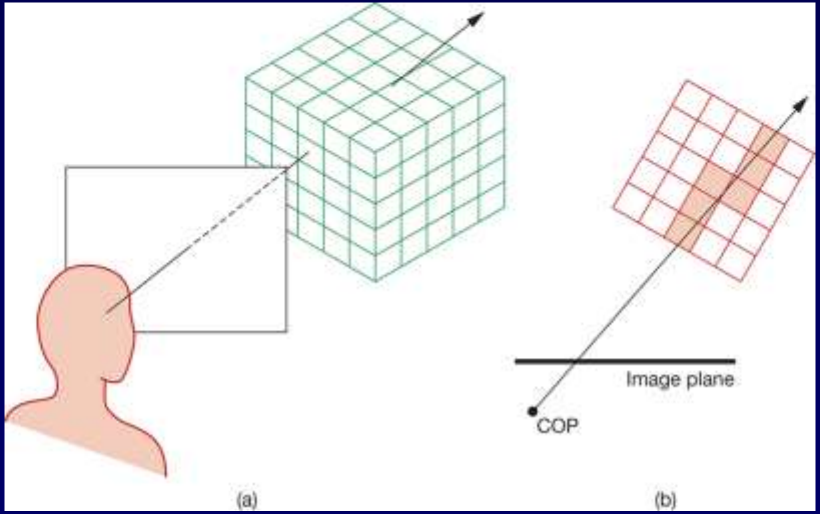


Volume Rendering: V-Buffer

- ***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):*** $c = c_1 + c_2(1 - \alpha_1)$; $\alpha = \alpha_1 + \alpha_2(1 - \alpha_1)$

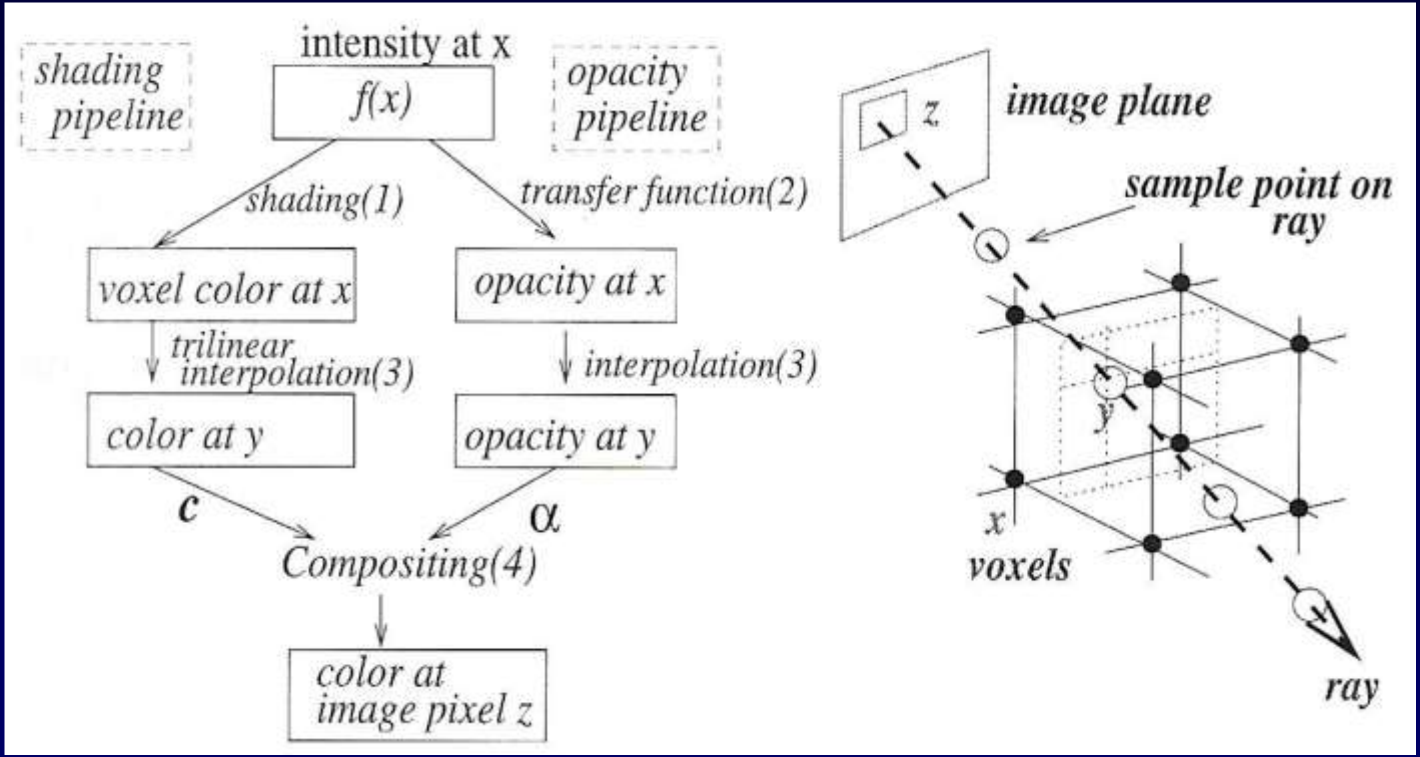
Volume Rendering: V-Buffer

- ***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***



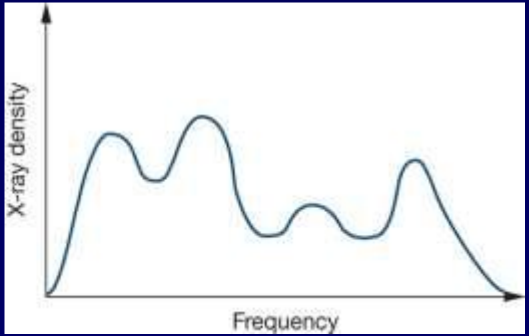
Volume Rendering

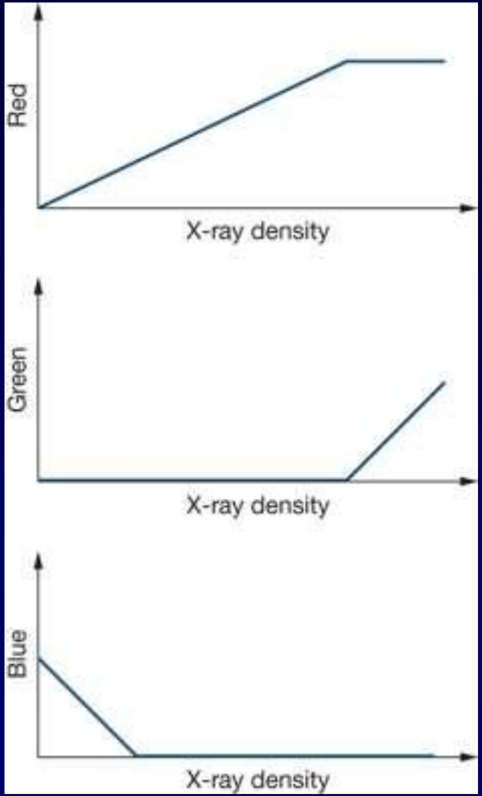
Algorithm



Volume Rendering: Transfer Function

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



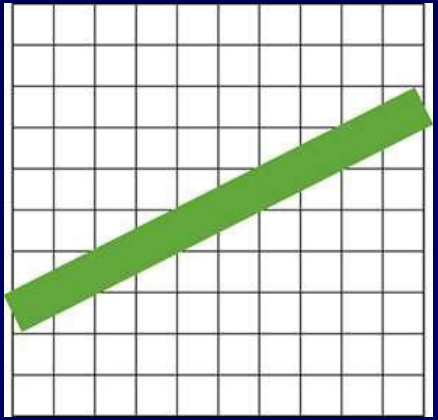


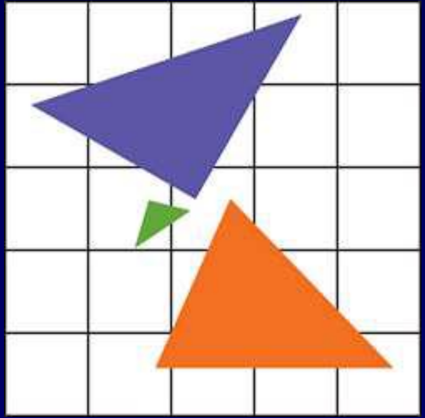
Volume Rendering: V-Buffer Speedups

- ***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

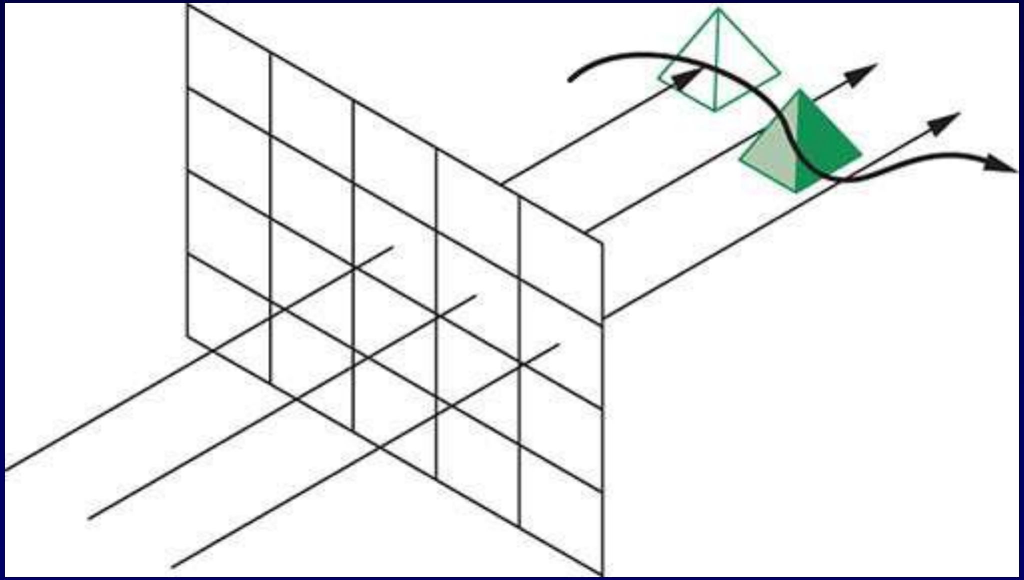
- ***Spatial aliasing in CG***
  - Jagged lines while rasterization
  - Going from continuous representation to a sampled approximation, which has limited resolution
  - Pixels on screen have fixed number, size, shape and location





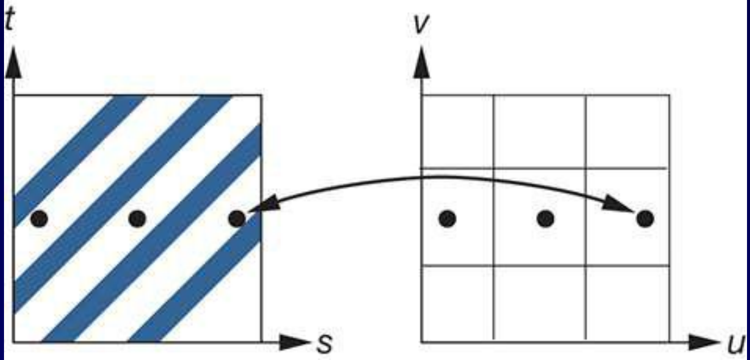
Aliasing: Temporal

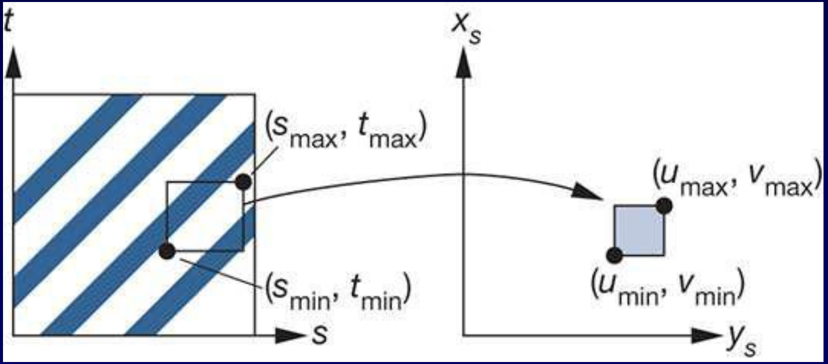
- *Temporal aliasing in CG*



Aliasing: Mappings

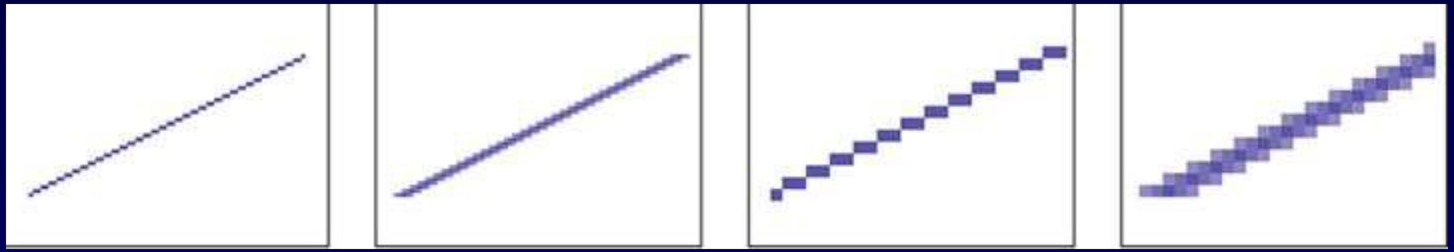
- *Due to high-frequency patterns*





Anti-Aliasing

- *Area averaging*



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