Blurring

- To blur across pixels, define a mask:
  - Whose values are non-negative
  - Whose value is largest at the center pixel
  - Whose entries sum to one.

Edge Detection

- To find the edges in an image, define a mask:
  - Whose value is largest at the center pixel
  - Whose entries sum to zero.

Forward Mapping – BAD!

- Iterate over source image

Multiple source pixels can map to same destination pixel

Some destination pixels may not be covered

Reverse Mapping – GOOD!

- Iterate over destination image
  - Must resample source
  - May oversample, but much simpler!

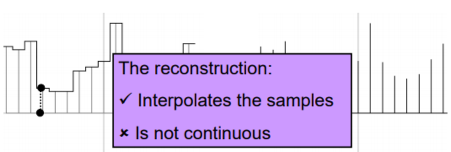
Resampling

- Evaluate source image at arbitrary (u, v)

(u, v) does not usually have integer coordinates

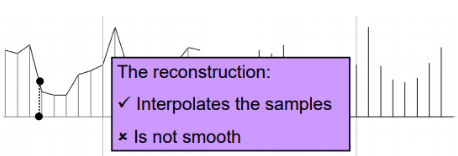
Nearest Point Sampling

The value at a point is the value of the closest discrete sample.



Bilinear Sampling

The value at a point is the (bi)linear interpolation of the two surrounding samples.



Gaussian Sampling

The value at a point is the Gaussian average of the surrounding samples.

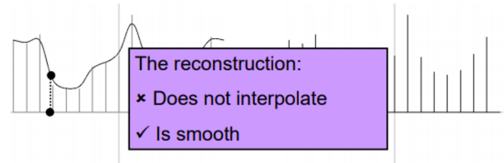
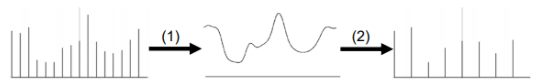


Image Sampling

Typically this is done in two steps:

1. Reconstruct a continuous function from input sample
2. Sample a continuous function at a fixed resolution.



Challenge:

Key Idea:
Of all possible reconstructions, we want the one that is smoothest (has lowest frequencies).

Signal processing helps us formulate this precisely.

Question

- As higher frequency components are added to the approximation, finer details are captured.
- If we have n frequencies, how many samples can we fit?

Each frequency component has two degrees of freedom:
- Amplitude
- Shift

With n frequencies we can fit $2n$ samples.

Sampling Theorem

Shannon's Theorem:

A signal can be reconstructed from its samples, if the original signal has no frequencies above $1/2$ the sampling rate -- a.k.a. the *Nyquist Frequency*.

Terminology:
- A signal is *band-limited* if its highest non-zero frequency is bounded.
- The frequency is called the *bandwidth*.
- The minimum sampling rate for band-limited function is called the *Nyquist rate* (twice the bandwidth).

Aliasing

- When a high-frequency signal is sampled with insufficiently many samples, it can be perceived as a lower-frequency signal. This masking of higher frequencies as lower ones is referred to as **aliasing**.

Sampling

- There are two problems:
  - You don't have enough samples to correctly reconstruct your high-frequency information
  - You corrupt the low-frequency information because the high-frequencies mask themselves as lower ones.

Summary – Nearest

✓ Can be implemented efficiently because the filter is non-zero in a very small region.
? Interpolates the samples.
× Is discontinuous.
× Does not address aliasing, giving bad results when a high-frequency signal is under-sampled.

Summary – Linear

✓ Can be implemented efficiently because the filter is non-zero in a very small region.
? Interpolates the samples.
× Is not smooth.
× Partially addresses aliasing, but stills give bad results when a high-frequency signal is under-sampled.

Summary – Gaussian

× Is slow to implement because the filter is non-zero in a large region.
? Does not interpolate the samples.
✓ Is smooth.
✓ Addresses aliasing by killing off high frequencies.

Summary – Sinc

× Is slow to implement because the filter is non-zero in a large region.
? Does interpolate the samples.
× Assigns negative weights.
× Ringing at discontinuities.
✓ Addresses aliasing by killing off high frequencies.

Alpha Channel

- Encodes pixel coverage information
  - $\alpha = 0$: no coverage (or transparent)
  - $\alpha = 1$: full coverage (or opaque)
  - $0 < \alpha < 1$: partial coverage (or semi-transparent)
- Single Pixel Example: $\alpha = 0.3$

Partial Coverage

Semi-Transparent

Image Processing

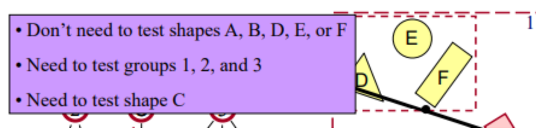
- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Composite
  - Morph

Intersection Testing

Accelerated techniques try to leverage:
- Grouping:
  Discard groups of primitives that are guaranteed to be missed by the ray.
- Ordering:
  Test nearer intersections first and allow for early termination if there is a hit.

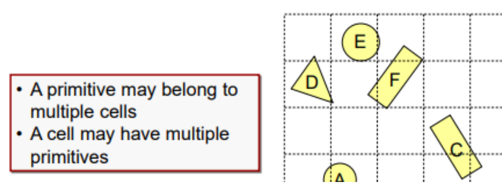
Bounding Volume Hierarchies

- Use hierarchy to accelerate ray intersections
  - Intersect nodes only if you haven't hit anything closer



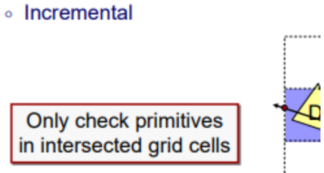
Uniform (Voxel) Grid

- Construct uniform grid over the scene
  - Index primitives according to overlaps with grid cells



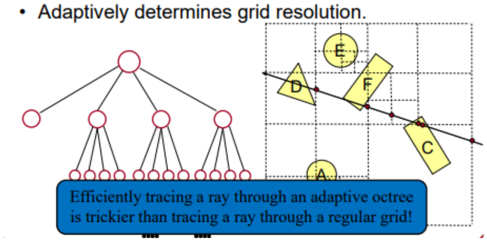
Uniform (Voxel) Grid

- Trace rays through grid cells
  - Fast
  - Incremental



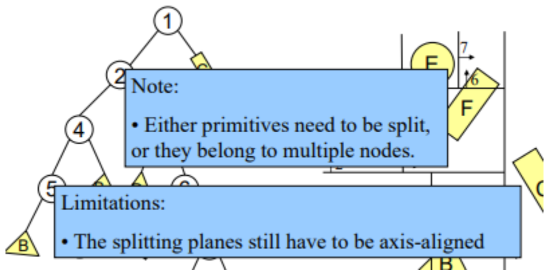
Octrees

- In an octree, we only subdivide regions that contain more than one shape.
- Adaptively determines grid resolution.



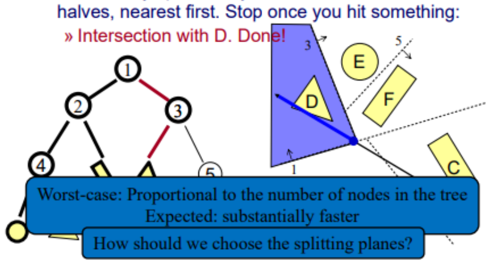
k-D Trees

- Alternate between splitting along the x-axis, y-axis, and z-axis.



Binary Space Partition (BSP) Tree

- Example: Ray Intersection 2
  - Recursively split the ray and test nearer and farther halves, nearest first. Stop once you hit something:
    - » Intersection with D. Done!



Transparency and Shadow

- Problem:
  - If a surface is transparent, then rays to the light source may pass through the object
  - Need to modify the shadow term so that instead of representing a binary (0/1) value, it gives the fraction of light passing through.
  - ⇒ Accumulate transparency values as the ray travels to the light source.

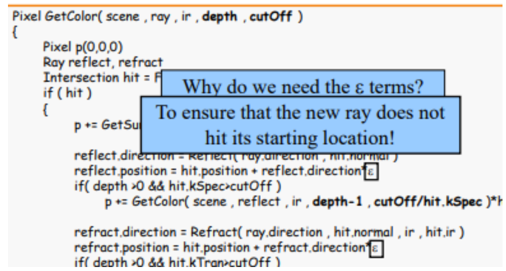
Snell's Law and Shadows

- Problem (Caustics):
  - If a surface is transparent, then rays to the light source may not travel in a straight line
  - This is difficult to address with ray-tracing

Termination Criteria

- How do we determine when to stop recursing?
  - Depth of iteration
    - » Bounds the number of times a ray will bounce around the scene
  - Cut-off value
    - » Ignores contribution from bounces that contribute very little

Termination Criteria



Summary

- Ray casting (direct Illumination)
  - Usually use simple analytic approximations for light source emission and surface reflectance
- Recursive ray tracing (global illumination)
  - Incorporate shadows, mirror reflections, and pure refractions

Linear Transformations

- Linear transformations are combinations of …
  - Scale, and
  - Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Properties of linear transformations:
  - Satisfies: $T(s_1 \cdot p_1 + s_2 \cdot p_2) = s_1 \cdot T(p_1) + s_2 \cdot T(p_2)$
  - ⇒ Origin maps to origin
  - ⇒ Lines map to lines
  - ⇒ Parallel lines remain parallel
  - ⇒ Closed under composition

Affine Transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Closed under composition

Projective Transformations

- Projective transformations …
  - Affine transformations, and
  - Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Closed under composition

Matrix Composition

- Be aware: order of transformations matters
  - » Matrix multiplication is not commutative

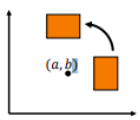
$$p' = T \cdot R \cdot S \cdot p$$

"Global" "Local"

Matrix Composition

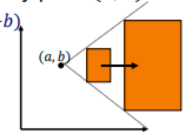
- Rotate by θ around arbitrary point (a, b)
  - $M = T(a,b) \circ R(\theta) \circ T(-a,-b)$

The trick:
First, translate (a, b) to the origin.
Next, do the rotation about origin.
Finally, translate back.



- Scale by (s_x, s_y) around arbitrary point (a, b)
  - $M = T(a,b) \circ S(s_x, s_y) \circ T(-a,-b)$

(Use the same trick.)



Scene Graphs

- Allow us to have multiple instances of a single model – reducing model storage size and making it easier to make consistent changes
- Allow us to model objects in local coordinates and then place them into a global frame – particularly important for animation
- Accelerate ray-tracing by providing a hierarchy that can be used for bounding volume testing

Applying a Transformation

- Position

$$p' = M(p)$$

- Direction

$$\vec{v}' = M_L(\vec{v})$$

- Normal

$$\vec{n}' = \left(M_L^t\right)^{-1}(\vec{n})$$

$$\underset{M}{\overset{\text{Affine}}{\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}}$$

Barycentric Coordinates

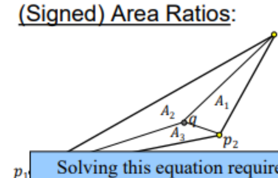
Barycentric coordinates are needed in:

- Ray-tracing, to test for intersection
- Rendering, to interpolate triangle information

Barycentric Coordinates

Given the points p_1, p_2, and p_3, how do we compute the barycentric coordinates of a point q in the plane spanned by p_1, p_2, and p_3?

(Signed) Area Ratios:



$$\alpha_q = \frac{A_1}{A_1 + A_2 + A_3} \qquad \beta_q = \frac{A_2}{A_1 + A_2 + A_3} \qquad \gamma_q = \frac{A_3}{A_1 + A_2 + A_3}$$

Solving this equation requires computing the areas of three triangles for every point q.

Barycentric Coordinates

Given the points p_1, p_2, and p_3, how do we compute the barycentric coordinates of a point q in the plane spanned by p_1, p_2, and p_3?

Solving this equation requires inverting a matrix. However, the matrix is independent of the point q and can be computed once and re-used for all points q.

Ma

Recall:

The system can be singular even for non-degenerate triangles.

Barycentric Coordinates

Linearity:

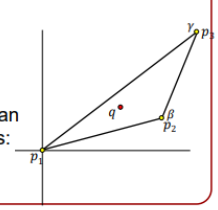
$$\begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{pmatrix} \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix}$$

If we set:

$$v_\beta = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \quad \text{and} \quad v_\gamma = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

the barycentric coordinates can be expressed as dot-products:

$$\beta = \langle q, v_\beta \rangle$$
$$\gamma = \langle q, v_\gamma \rangle$$



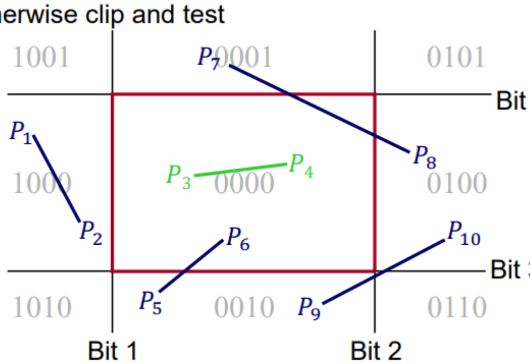
Perspective vs. Parallel

- Perspective projection
  - ✓ Size varies inversely with distance - looks realistic
  - ✓ Angles are preserved on faces parallel to the view plane
  - × Distance are not preserved
  - × Only parallel lines that are parallel to the view plane remain parallel

- Parallel projection
  - ✓ Good for exact measurements
  - ✓ Parallel lines remain parallel
  - ✓ Angles and distance are preserved on faces parallel to the view plane
  - × Less realistic looking

Cohen-Sutherland Line Clipping

- Associate an outcode to each vertex
- **If both outcodes are 0, line segment is inside**
- If AND of outcodes not 0, line segment is outside
- Otherwise clip and test

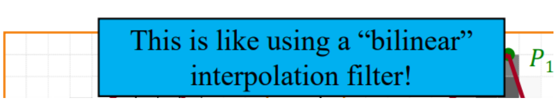


Antialiasing Techniques

- Display at higher resolution
  - Corresponds to increasing sampling rate
  - Not always possible (fixed size monitors, fixed refresh rates, etc.)
- Modify pixel intensities
  - Vary pixel intensities along boundaries for antialiasing
  - Must have more than bi-level display

Antialiasing

- Method 1: Area sampling
  - Calculate percent of pixel covered by primitive
  - Multiply this percentage by desired intensity/color



Antialiasing

- Method 2: Supersampling (aka postfiltering)
  - Sample as if screen were higher resolution
  - Average multiple samples to get final intensity

Antialiasing

Note that this makes things harder because pixels are no longer "owned" by a single triangle.

- Triangles contribute color rather than set color
- Along edges the total contribution must sum to one.
- × Makes depth-testing more complicated.

Flat Shading

- Can take advantage of spatial coherence
  - Make the lighting equation constant over the surface of each primitive

	Surf
Emissive	
Ambient	
Diffuse	
Specular	

- If the normal is constant over the primitive, and
- if the light is directional,

the diffuse component is the same for all points on the primitive

- If the normal is constant over the primitive,
- if the light is directional, and
- if the direction to the viewer is constant over the primitive

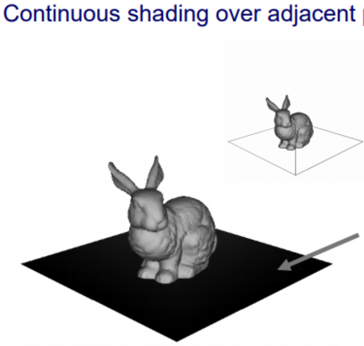
the specular component is the same for all points on the primitive

Flat Shading

- Objects look like they are composed of polygons
  - OK for polyhedral objects
  - Not so good for smooth surfaces

Gouraud Shading

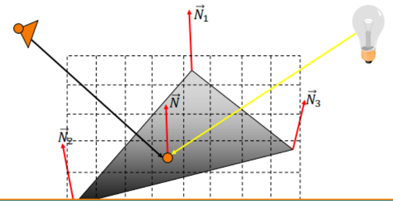
- Produces smoothly shaded polygonal mesh
  - Continuous shading over adjacent polygons



What happens with large polygon & spotlight?

Phong Shading

- Linearly interpolate surface normals at vertices down and across scan lines



This was not supported in early generation graphic cards but can now be implemented in the fragment shader.

Back-face detection

This method:

- ✗ Does not handle overlapping primitives
- ✗ Does not work for non-solid models and/or models without a well defined orientation.

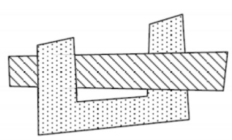
Ideal Solution

Painter's Algorithm:

- Sort primitives front to back and draw the back ones first, over-writing pixel values with information from the front primitives as they are processed.

Problem:

- In general you can't sort the primitives.

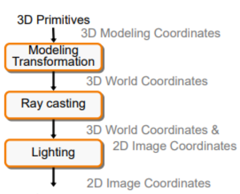


- ...Unless you are allowed to split them

z-Buffer

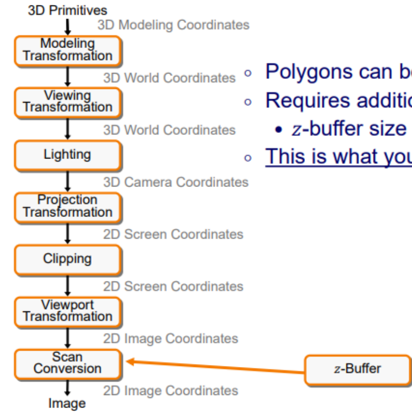
- Store color & depth of closest object at each pixe
  - Initialize depth of each pixel to ∞
  - Update only pixels whose depth is closer than in buffe
  - Depths are interpolated from vertices, just like colors

Ray Casting Pipeline



Ray casting
- $P(p \log n)$ for p pixels and n shapes
- May (or not) use pixel coherence
- Simple, but generally not used

3D Rendering Pipeline



- Polygons can be rasterized in any order
- Requires additional memory
  - z-buffer size ≈ frame buffer
- This is what your graphics card does!

Scan Conversion Example

A line segment in 2D projected onto a 1D screen.

- How do we interpolate correctly?

Recall: The 2D point (x,z) maps to the point (x/z) in 1D.

If p_1 ... e α
for a ... olv

To compute the interpolation weights correctly, we need to perform a perspective divide:

$$\frac{(1-\alpha)x_1+\alpha x_2}{(1-\alpha)z_1+\alpha z_2}=\frac{x}{1}$$

This is not the same as solving for the blending value in the image plane:

$$(1-\alpha)\frac{x_1}{z_1}+\alpha\frac{x_2}{z_2}=\frac{x}{1}$$