CS 184: Computer Graphics and Imaging Lecture Notes

Week 1: Lecture 1 Introduction (1/18)

Why study Computer Graphics?

Computer Graphics

- Def: Use of computers to synthesize and manipulate visual information

Computer Graphics use

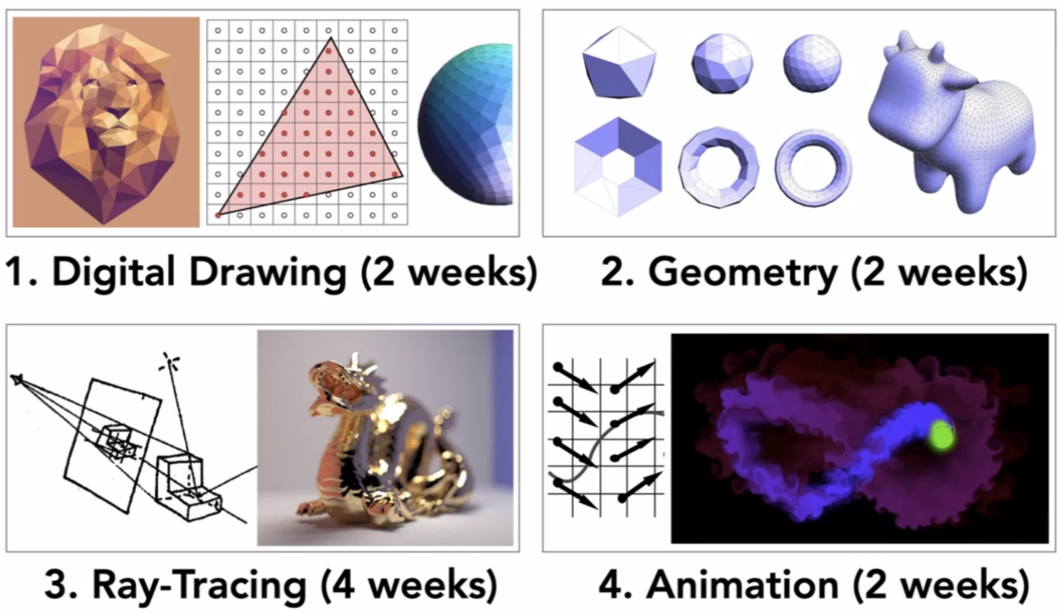
- Movies vs stop motion previously
- Motion capture for suits that capture location of facial expressions and actions
- Product Design and Visualization
    - Interaction of light, Razezation graphics, ray-tracing, path-tracing
- Typography
    - Control points, bezier curve
- Digital Illustrated design
- Computer-Aided Design
- Architectural Design
- Visualization
- Graphical User Interfaces
- Imaging in Mapping
- Virtual Reality

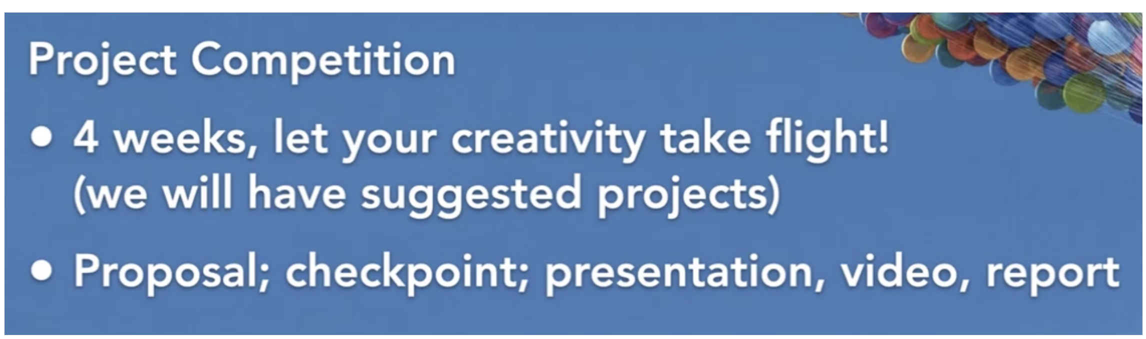
Foundations of Graphics and Imaging

- Applications require sophisticated theory and systems
- Science and Math
    - Physics of light, color optics
    - Math of curves, surfaces, geometry, perspective
- Technology and systems
    - Input devices, GPUs, displays
    - Cameras, lenses, sensors
- Art and Psychology
    - Perception: color, stereo, motion, image quality
    - Art and design: composition, form, lightning

Course Overview

- Overview of core ideas in graphics and imaging
    - Modeling the world, image synthesis
    - 3D graphics: geometry, rendering, animation
    - Image capture, manipulation and display
- Acquire core concepts and skills
    - Representations (geometry)

- Algorithms (sampling, subdivision, ray-tracing)
- Technology (GPUs, displays, cameras)
- Drawing digital Images (rasterization)
- Filtering and sampling
- Modeling geometry
- Modeling material properties
- Modeling lighting
- Light transport and Image synthesis (photograph CCD) vs computer rendering
- How do cameras work
- VR
- 

- 

Logistics

-

Week 1: Lecture 2 Digital Drawing (1/20)

Today: Drawing Triangles to the Screen by Sampling

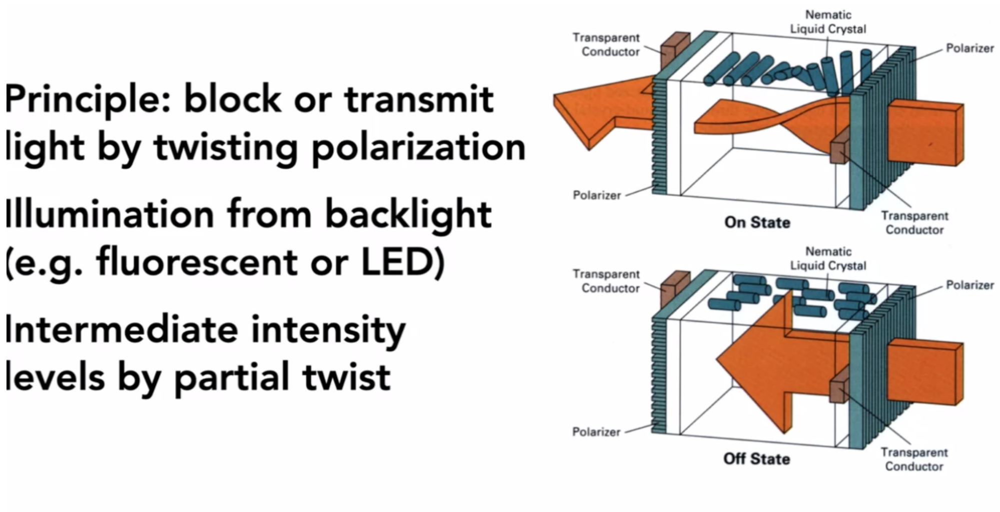
Drawing Machines

- CNC Drawing Machine, laser cutters

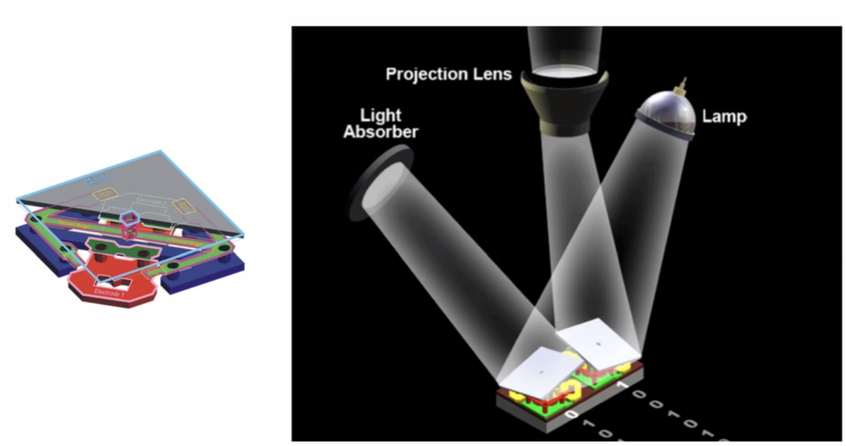
Frame buffer: memory for a raster display
- For every pixel what color should be
- Digital to analog convertors to emit color we want

<u>A sampling of different raster displays</u>

1. Flat Panel displays
    a. Los res LCD, Color LCD, OLED
    b. LCD (liquid crystal display)



    c. LED Array Display
        i. Light emitting diode array, each individual pixel
    d. DMD Projection (Digital MIcro Mirror Device)
        i. Pixels are mirrors and reflect light ornot toward mirror



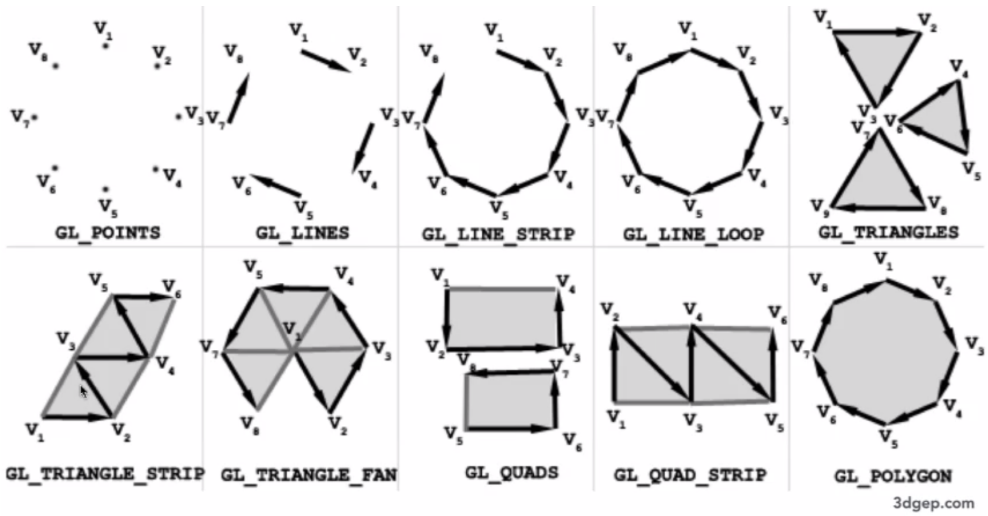
    e. E-ink displays
    f. Smartphoen screen pixels
        i. Array of light emitors

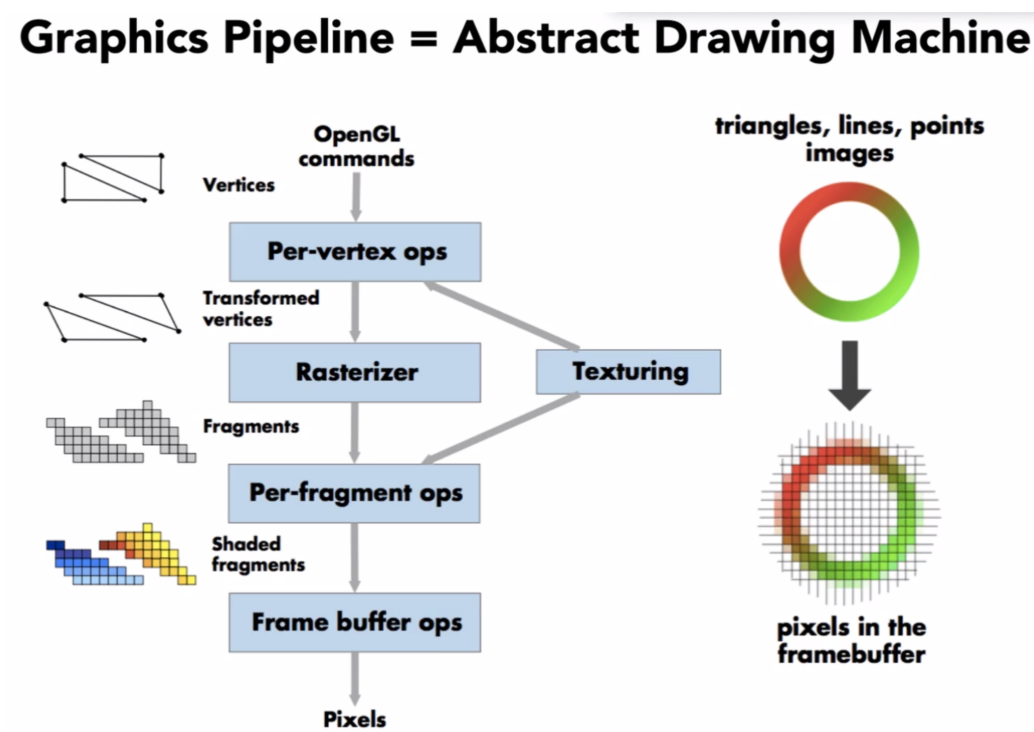
<u>Drawing to Raster Displays</u>

Polygon Meshes

Shape Primities

- OpenGL: API for raster drawing machines





-
- Raserization: transform triangles to fragments or pixels

Triangles - Fundamental Area Primitive

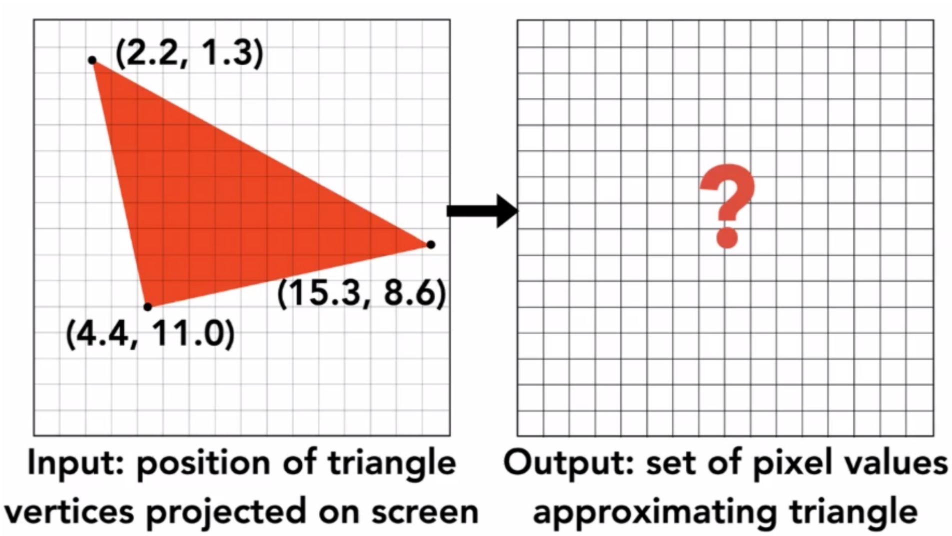
- Why Triangles?
  - Most basic polygon
    - Break up other polygons
    - Optimize one implementation
  - Triangles have unique properties
    - Guaranteed to be planar
    - Well defined interior

- Well-defined method for interpolating values at vertices over triangle

Drawing a Triangle to the Framebuffer ("Rasterization")

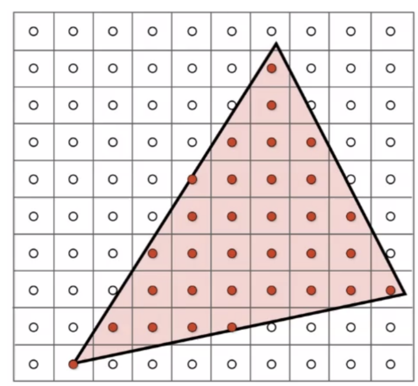
Rasterization : sampling a 2d indicator function

What Pixel Values approximate a Triangle



-

Sampling a Function

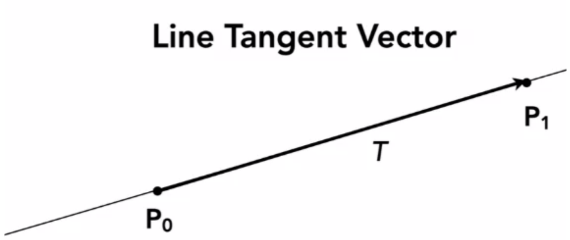


- Evaluating a function at a point is sampling
- Descretize a function by periodic sampling
    - Sample time (1d), area (2d), angle (2d), volume (3d)

1. Sample if each Pixel center is inside triangle

$$\texttt{inside(t,x,y)} = \begin{cases} \texttt{1} & \texttt{(x,y) in triangle t} \\ \texttt{0} & \texttt{otherwise} \end{cases}$$

Triangle = Intersection of three half planes



$$T = P_1 - P_0 = (x_1 - x_0, y_1 - y_0)$$

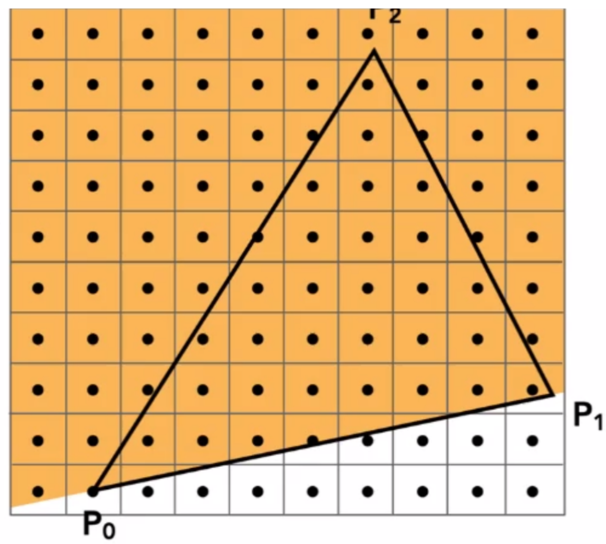
$$\mathrm{Perp}(x, y) = (-y, x)$$

$P_i = (X_i, Y_i)$

$dX_i = X_{i+1} - X_i$
$dY_i = Y_{i+1} - Y_i$

$$L_i(x, y) = -(x - X_i)\, dY_i + (y - Y_i)\, dX_i$$
$$= A_i x + B_i y + C_i$$

$L_i(x, y) = 0$: point on edge
< 0 : outside edge
> 0 : inside edge



$L_0(x, y) > 0$

Some Details

- Sample point coved by triangle 1, triangle 2 or both
- Modern approach: tiled Triangle Traversal
    - Parallel ism

<u>Signal Reconstruction on Real Displays</u>

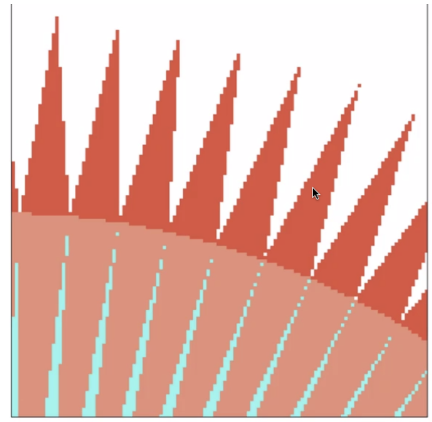
Assume Display Pixels Emit Square of light

- Send the display
- Jaggies

Potential topics for your pair discussion:

- **Ideas for "higher quality" pixel formula?**
- **What are all the relevant factors?**
- **What's right/wrong about point sampling?**
- **Why do jaggies look "wrong"?**

<u>**Week 2: Lecture 3 Intro to Signal Processing: Sampling, Aliasing, Antialiasing (1/25)**</u>

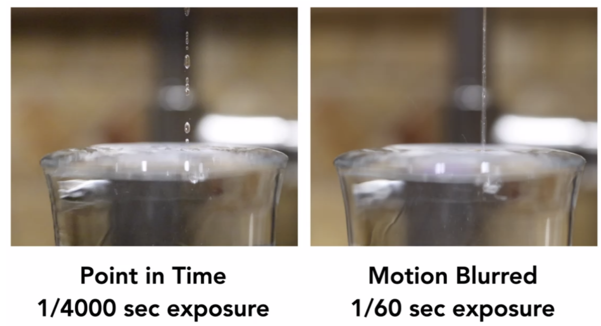


Photograph = sample Image Sensor Plane
Ray tracing = sample rays
Sampling Artifacts in Graphics and Imaging

- Jaggies (staircase patter)
- Wagon wheel effect – sampling in time
- Fast-changing signals, when we sample too slowly

Antialiasing Idea: Filter out high frequencies before sampling

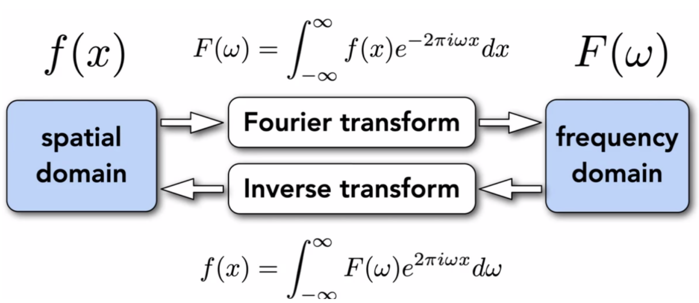
- Sharp is a fast varying signal, blurry is low frequency
- Antialiasing filter out the higher frequencies

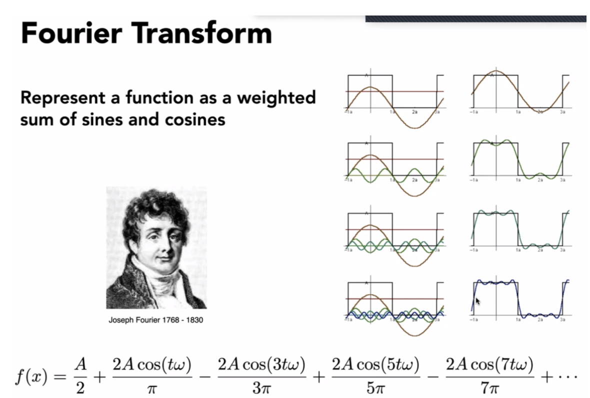


This Lecture

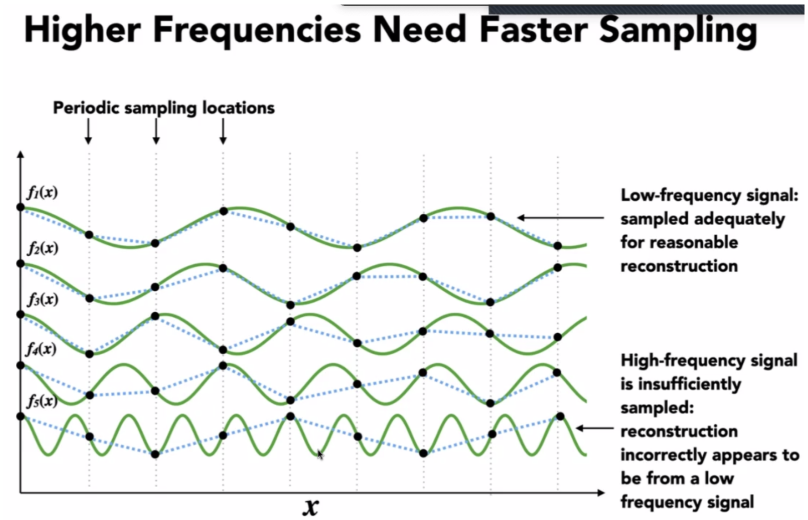
- Fundamental reasons why this works
- Look at how to implement antialiased rasterization

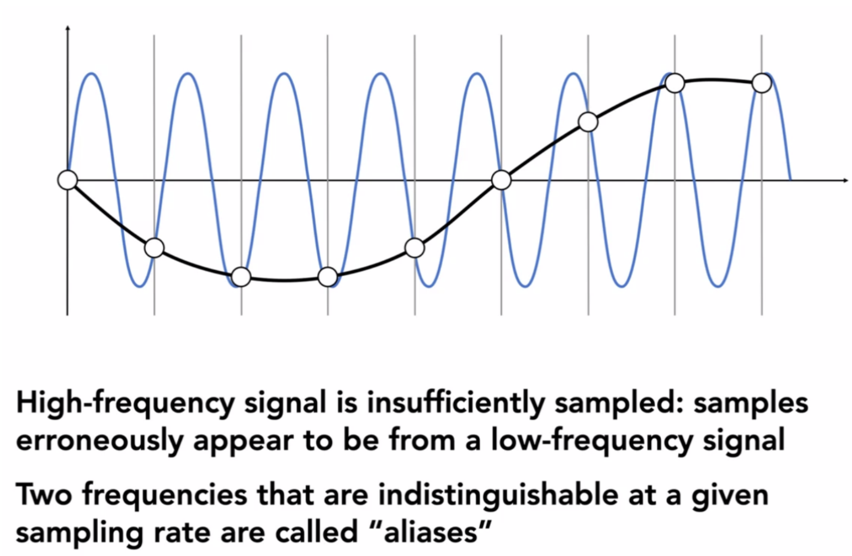
Frequency Space
Fourier Transform

- 



Higher Frequencies Need Faster Sampling

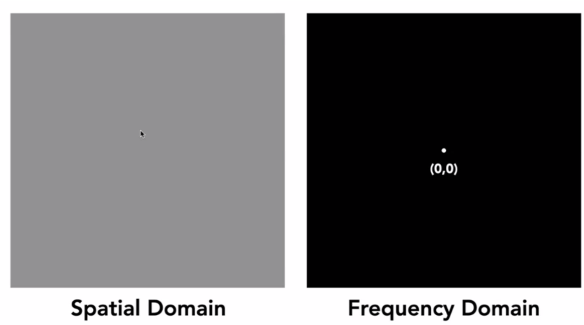


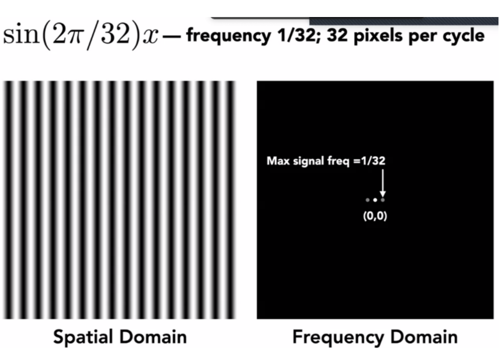


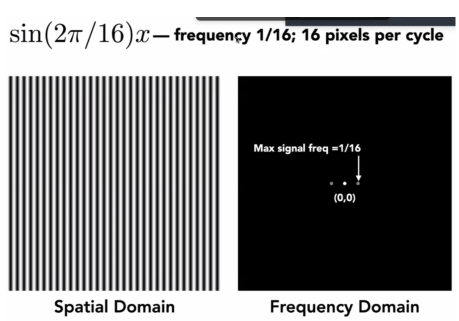
- Aliases are false identities

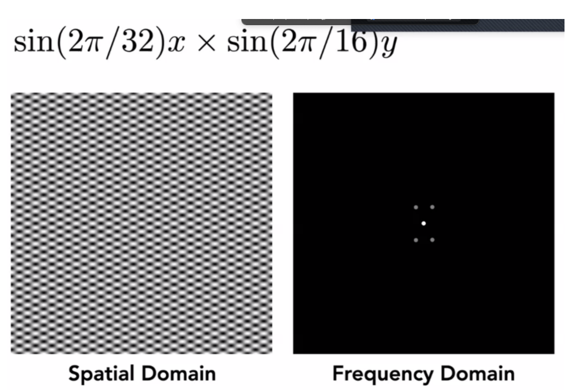
2D frequency domain

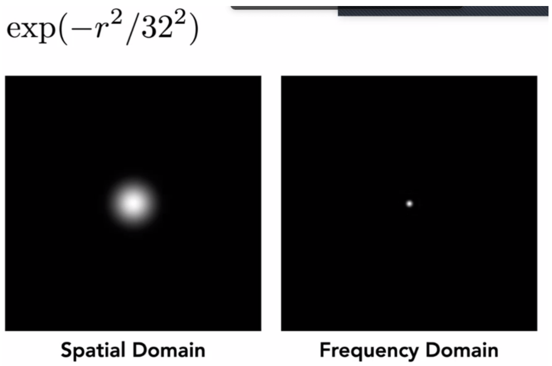
- Low frequency means its varying very slowly, closer to the middle
- More it varies the f
- Sin correspond to different points on the frequency domain

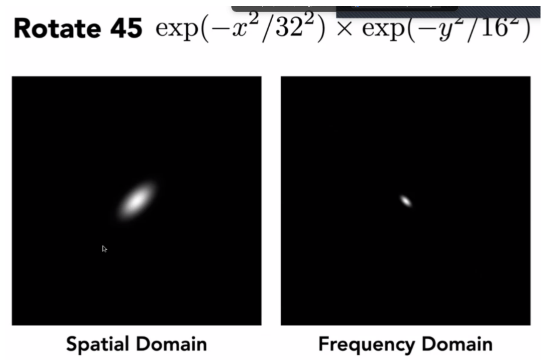






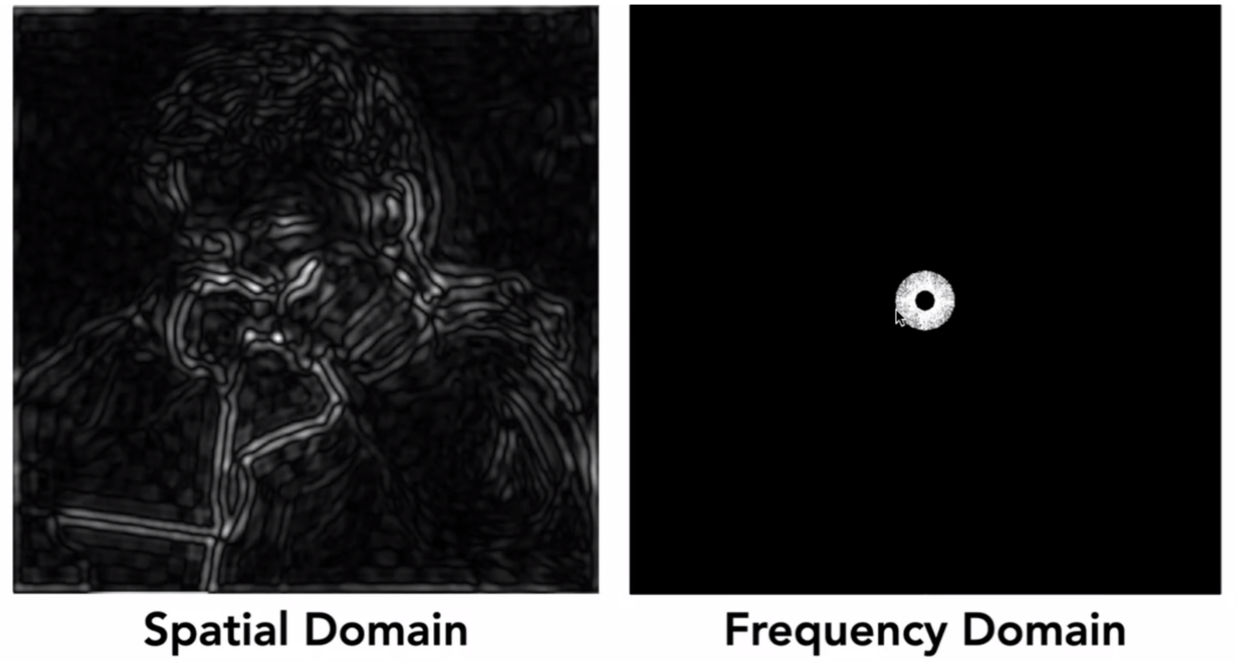






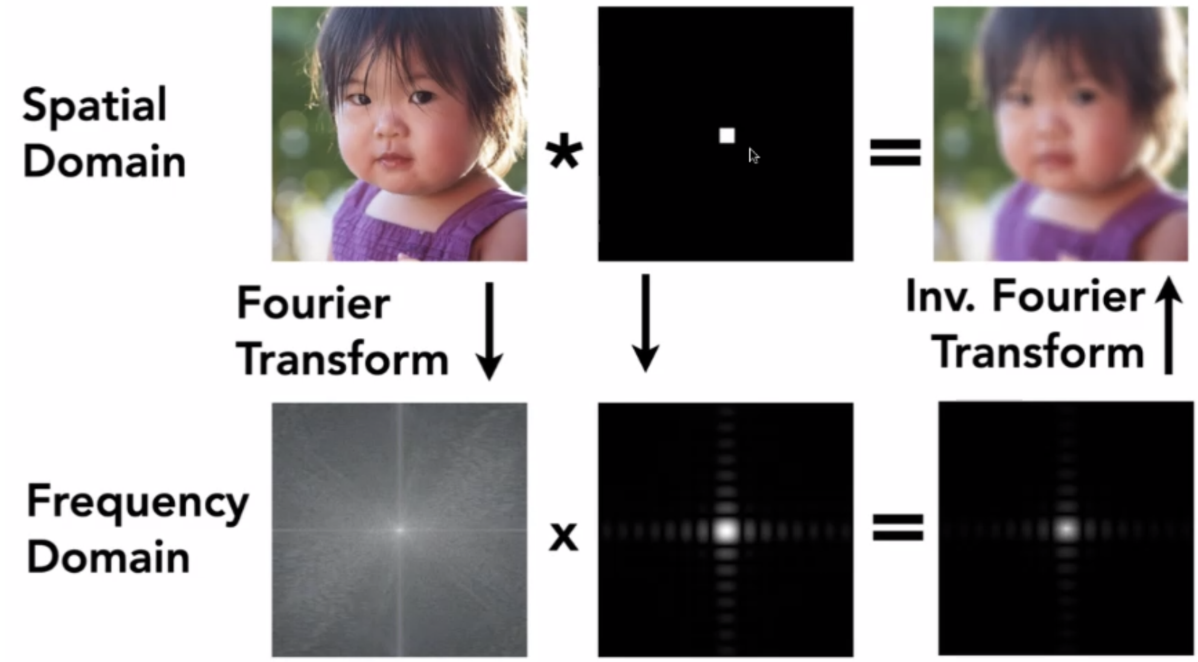
Filtering

- Filter out high frequencies



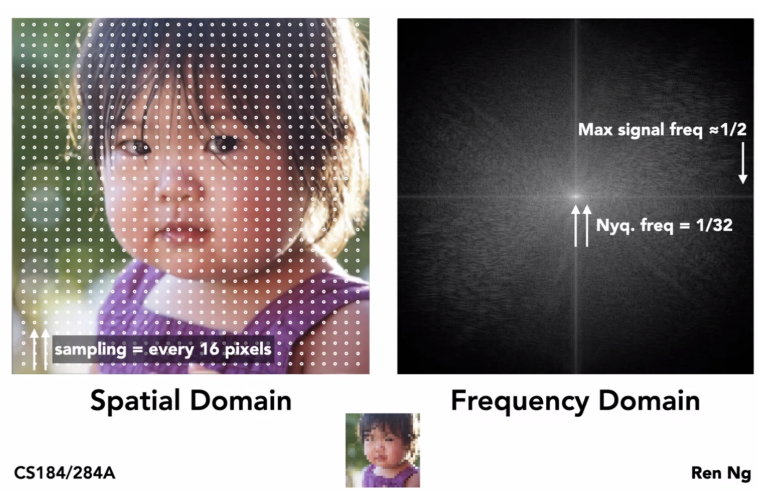
Further away the finer the rate of fluctuations
Filtering = Convolution

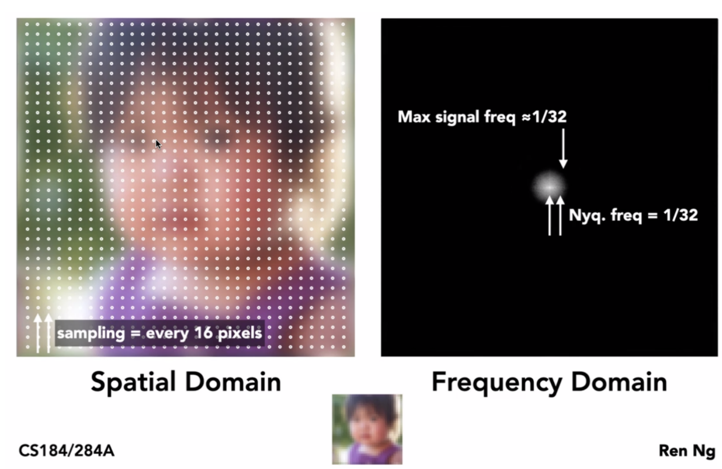
- SLiding weighted filter across the signal
- Convolution Theorem
    - Convolution in the spatial domain is equal to multiplication in the frequency domain and vice versa
    - Option 1
        - Filter by convolution in the spatial domain
    - Option 2
        - Transform to frequency domain
        - Multiply by fourier transform of convolution kernel
        - Transfrom back to spatial domain

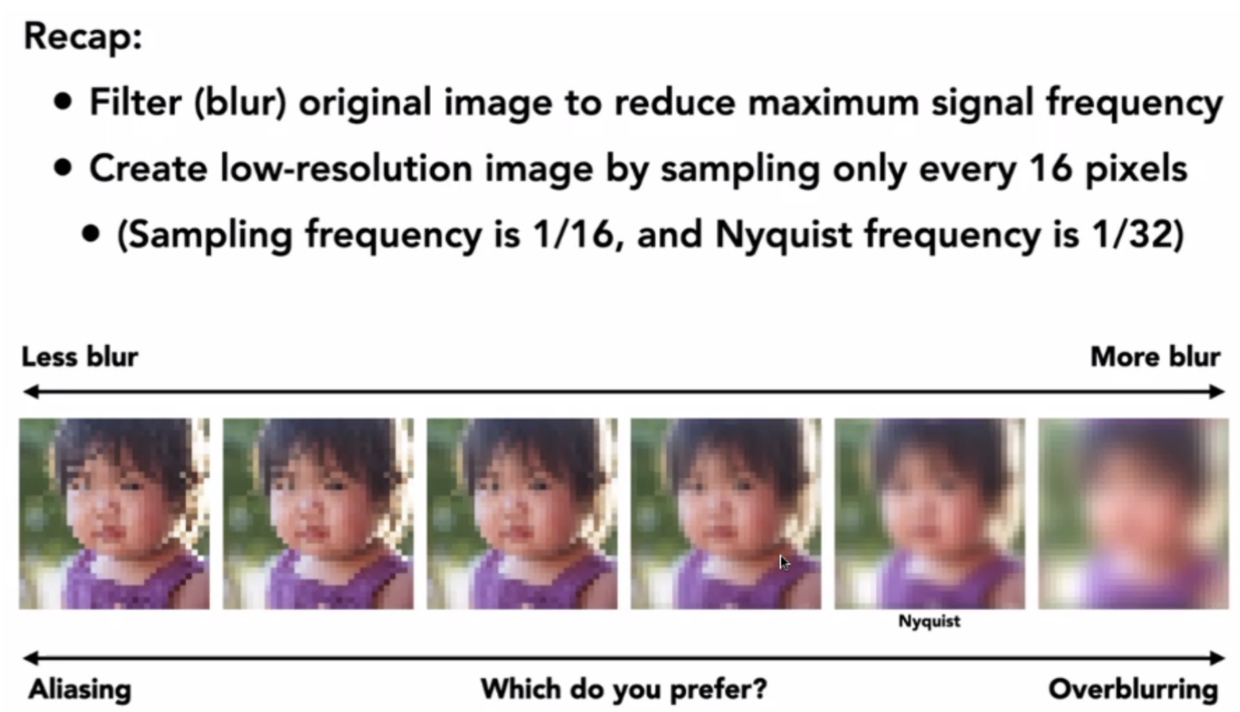


Nyquist Frequency & Antialiasing
- Nyquist Theorem: We get no aliasing from frequencies in the signal that are less than the Nyquist frequency (defined as half the sampling frequency)
- Half of sampling frequency

Visual Example: Image Frequency



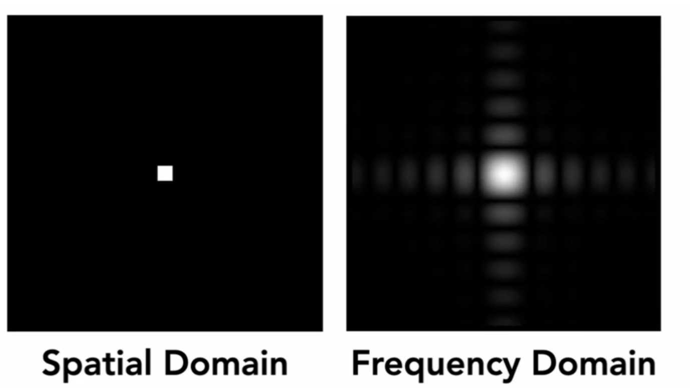




How Can we reduce Aliasing error

- Increase sampling rate (increase Nyquist Frequency)
    - Higher resolution displays, sensors, framebuffers
    - But: costly & may need very high resolution
- Antialiasing
    - Simple idea: remove signal frequencies that are high
    -

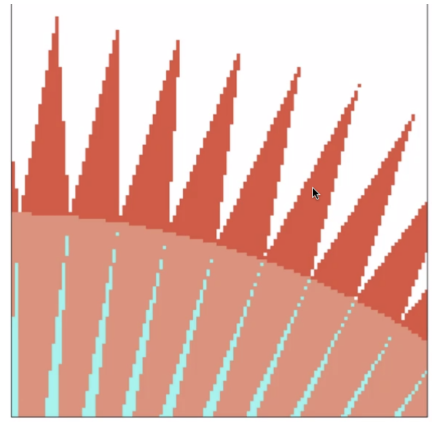
Box Filter

- 

AntiAlisaing By Averaging Vlaues in Pixel Area

- Option 1
    - Convolve f(x,y) by a 1-pixel box-blur
    - Then sample at every pixel
- Option 2
    - Compute the average value of f(x, y) in the pixel

Week 2: Lecture 4 Transforms (1/27)



Rotation, Scale

Transforms are functions acting on points

- (x′, y′, z′) = F(x, y, z)

Project Polygons in 3D to 2D screen

- Depends on the camera angle

Why Study Transforms

- Modeling
    - Define shapes in conventient coordinates
    - Dnable multiple copies of the same object
    - Represent hierarchical scenes
- Viewing
    - World coordinates to camera coordinates
    - Parallel / perspective projections from 3D to 2D

Lecture Outline

- Think about trasnformations
    - Types: rotate, translate, scale
    - Coordinate frames
    - Composing multiple transformations
    - Hierarchical transforms
    - Perspective projection
- How to implement
    - Represent transforms as matrices
    - Homogeneous coordinates

Linear Transforms = Matrices

- Scale matrix, reflection matrix, shear matrix

Linear Transforms = Matrices

$$x' = a\,x + b\,y$$
$$y' = c\,x + d\,y$$

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

$$\mathbf{x}' = \mathbf{M}\,\mathbf{x}$$

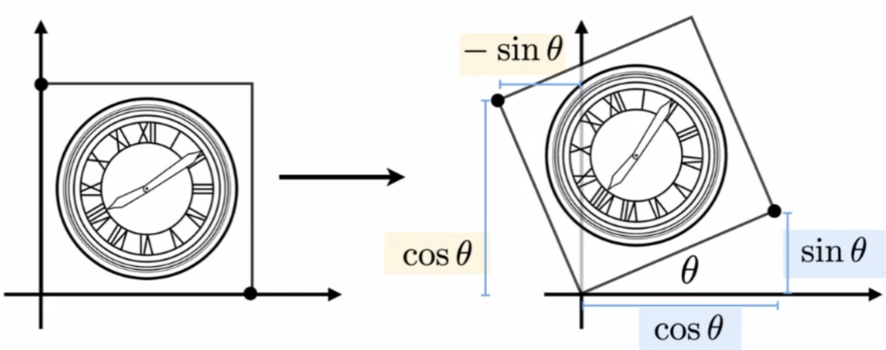
2D Coordinate Systems

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

$$\begin{bmatrix} a \\ c \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Rotation Matrix



$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Translation

$$x' = x + t_x$$
$$y' = y + t_y$$

Homogenous Coodinates

- Add a third coordinate (w-coordinate)
- 2D point, $(x,\ y,\ 1)^T$ 2D vector $(x,\ y,\ 0)^T$
- Express translation as a matrix

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

-

Affine Transfomrations

- Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

-

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

-

2D Transformations

Scale

$$\mathbf{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation

$$\mathbf{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

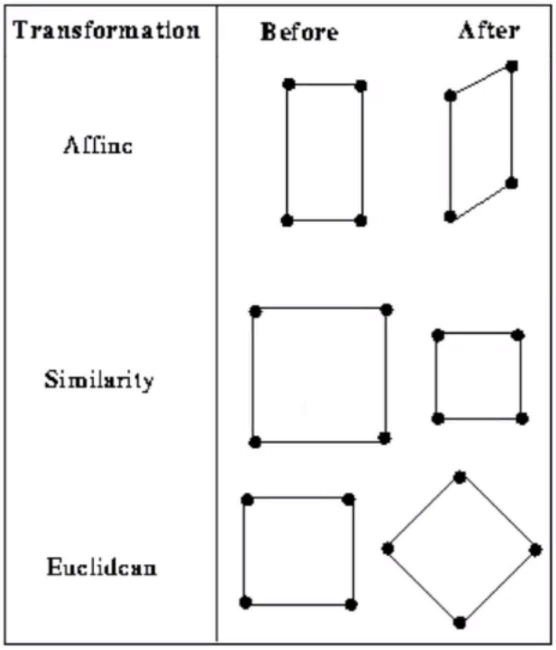
Translation

$$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

"Similarity Transform"

"Rigid Transform"

-
- Transform Ordering matters

- 

- Matrix multiplication is not commutative

$$R_{45} \cdot T_{(1,0)} \neq T_{(1,0)} \cdot R_{45}$$

- Composing transformations
    - Compose by matrix multiplication

$$A_n(\ldots A_2(A_1(\mathbf{x}))) = \underbrace{\mathbf{A}_n \cdots \mathbf{A}_2 \cdot \mathbf{A}_1}_{\text{Pre-multiply } n \text{ matrices to obtain a single matrix representing combined transform}} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

    -

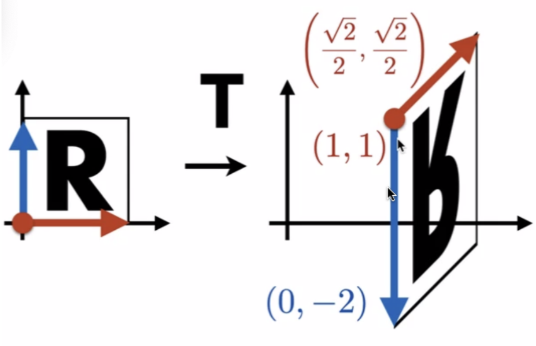
Decomposing Complex Tranforms

Coordinate Systems

- New coordinate frame is defined by origin (point) and two unit axes (vectors)
- Given coordinates in (o, u, v) reference frame, transform to coordinates in the (x, y) frame

$$F = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{o} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u_x & v_x & o_x \\ u_y & v_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

Coordinate system transform



$$\begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{o} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & 1 \\ \frac{\sqrt{2}}{2} & -2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

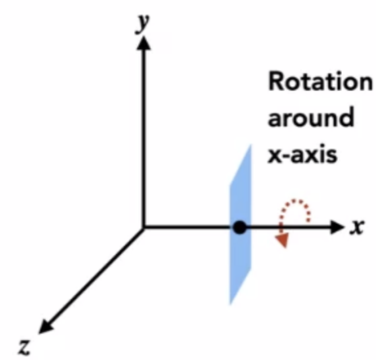
-

Rotation around x-, y-, or z-axis

$$\mathbf{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

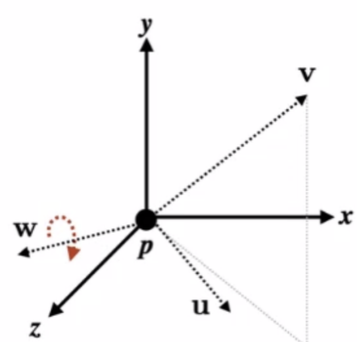
$$\mathbf{R}_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



-

Construct orthonormal frame transformation F with p, u, v, w, where p and w match the rotation axis

Apply the transform (F $R_z(\theta)$ F^{-1})



Rodrigues’ Rotation Formula

Rotation by angle α around axis $\mathbf{n}$

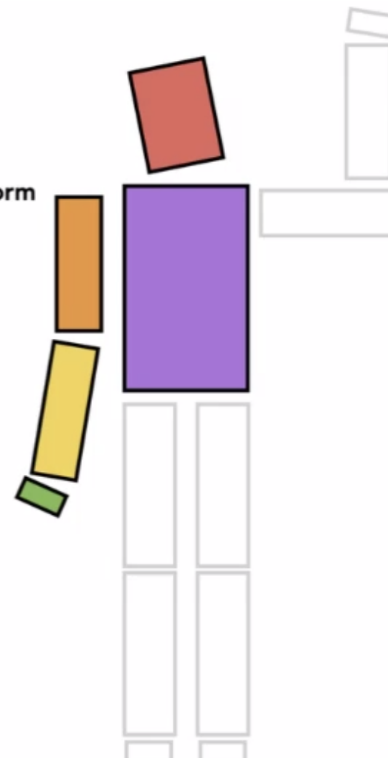
$$\mathbf{R}(\mathbf{n}, \alpha) = \cos(\alpha)\,\mathbf{I} + (1 - \cos(\alpha))\,\mathbf{n}\mathbf{n}^T + \sin(\alpha)\underbrace{\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}}_{\mathbf{N}}$$

-

Hierarchical Representation

- Grouped representation (tree)

```
translate(0, 10);
   drawTorso();
      pushmatrix();  // push a copy of transform onto stack
         translate(0, 5);  // right-multiply onto current transform
         rotate(headRotation);  // right-multiply onto current transform
         drawHead();
      popmatrix();  // pop current transform off stack
      pushmatrix();
         translate(-2, 3);
         rotate(rightShoulderRotation);
         drawUpperArm();
         pushmatrix();
            translate(0, -3);
            rotate(elbowRotation);
            drawLowerArm();
            pushmatrix();
               translate(0, -3);
               rotate(wristRotation);
               drawHand();
            popmatrix();
         popmatrix();
      popmatrix();
      ....
```



- **CS184/284A** **Ren Ng**

Viewing and Perspective for Transforms

-

Camera Space

- Camera located at the origin
- Looking down negative z-axis
- Vertical vector is y-axis
- (x-axis) orthogonal to y & z
- U = up vector
- V = view direction
- E = eye point (position of camera)
- Input: e, u & v given in world space coordinates

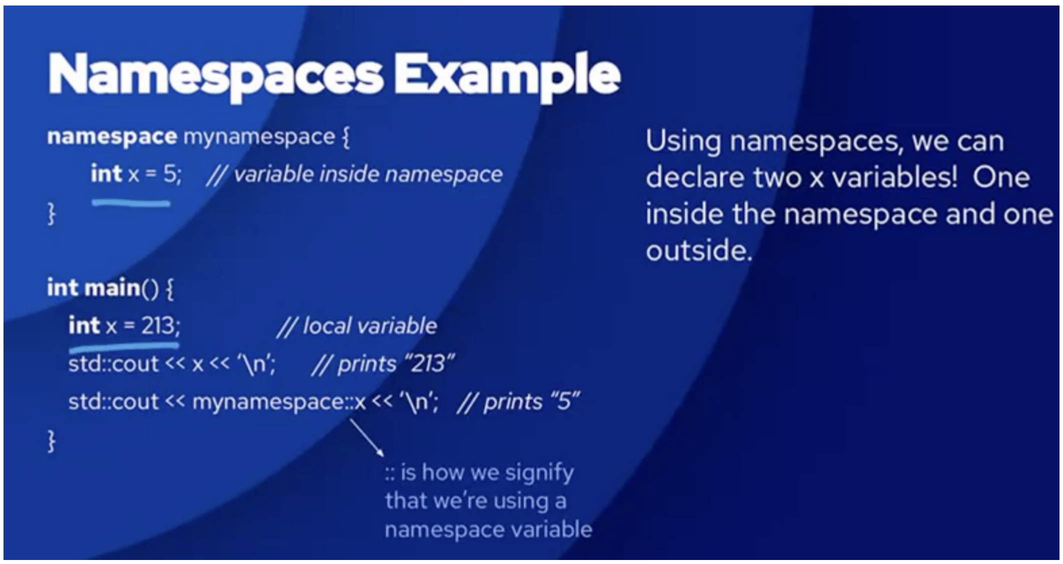
- Output: transform matrix from world space to standard camera space
- Camera “look-at” transformations
-

Intro to C++

Why C++

- Graphics = Comutation-heavy
- Faster than Java, more control over computer resources, statically typed,
- Used frequently in games/animation software

Namespaces

- Provide additional scope for variables,
- 

Classes and Objects

- All methods/attributes of C++ class are private, unless explicitly public
- Structs

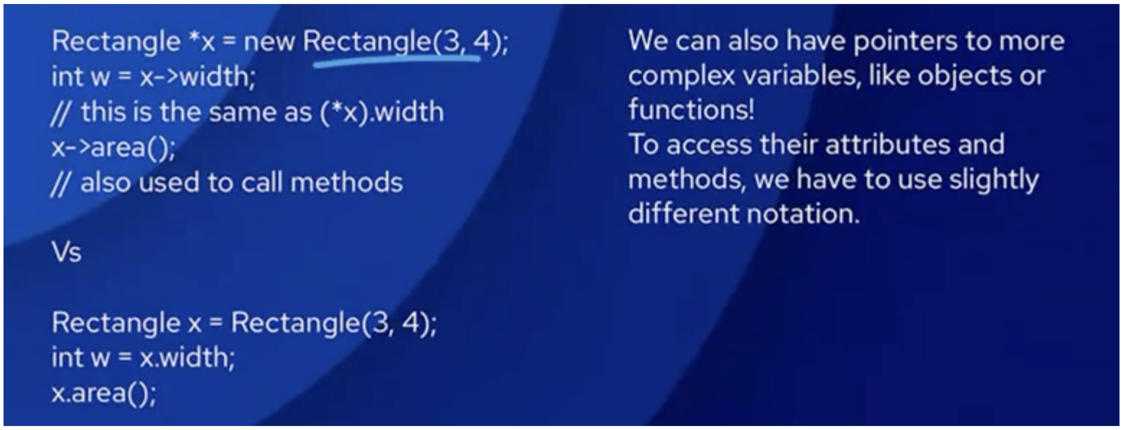
Memory: Pointers & Addresses

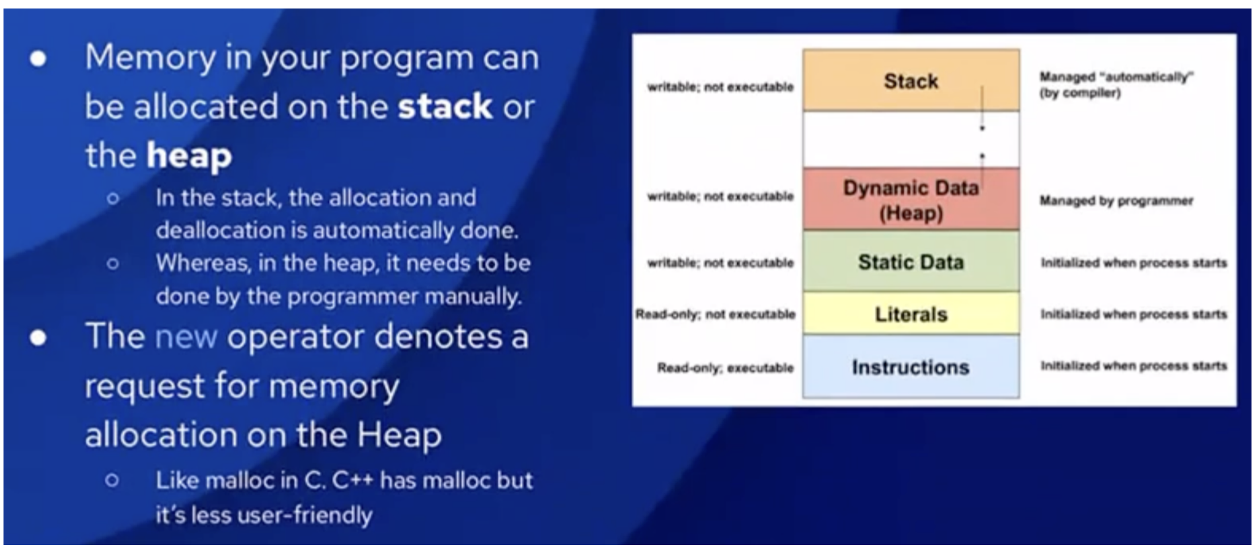
- C++ assigned an address in memory

Using * and &

- & (reference) operator gives you address occupied by a variable
- Values stored in memory address, use dereference operator (*)

Pointers to Objects

- 



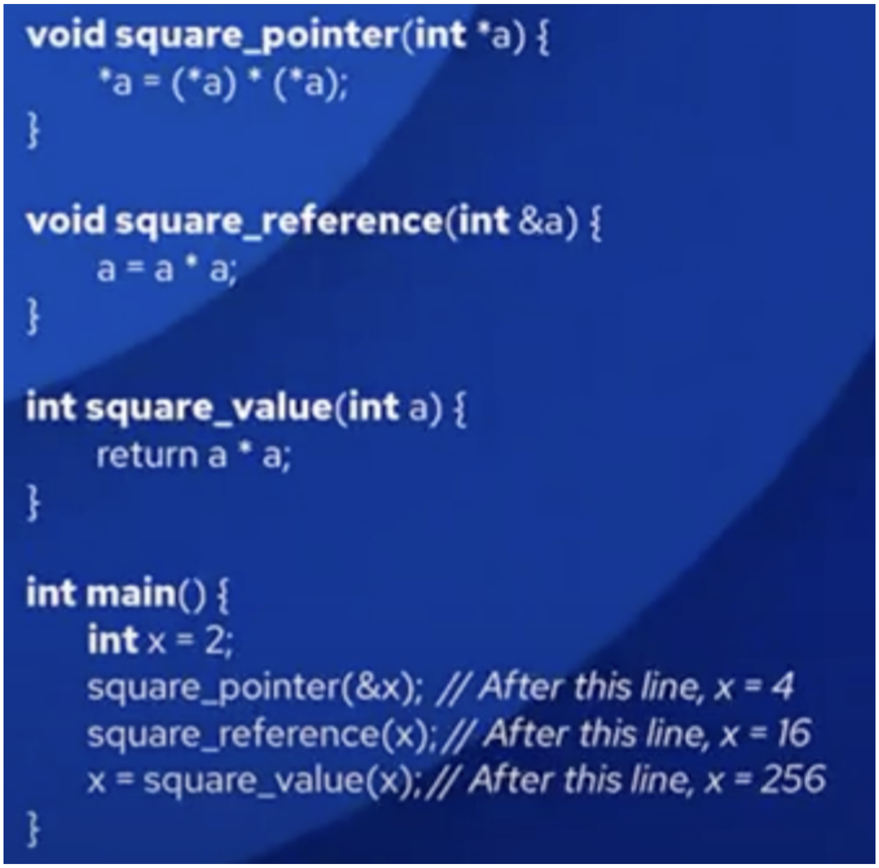
Stack vs Heap

- Stack Rectangle rect
    - No need to delete, use rect.width
- Heap: Rectangle *rp = new Rectangle();
    - Must call delete rp or will exist until your program ends
    - To get attributes use (rp->width)
    - If constructor has arguments, Rectangle *rp = new Rectangle(3, 4)

Passing Arguments

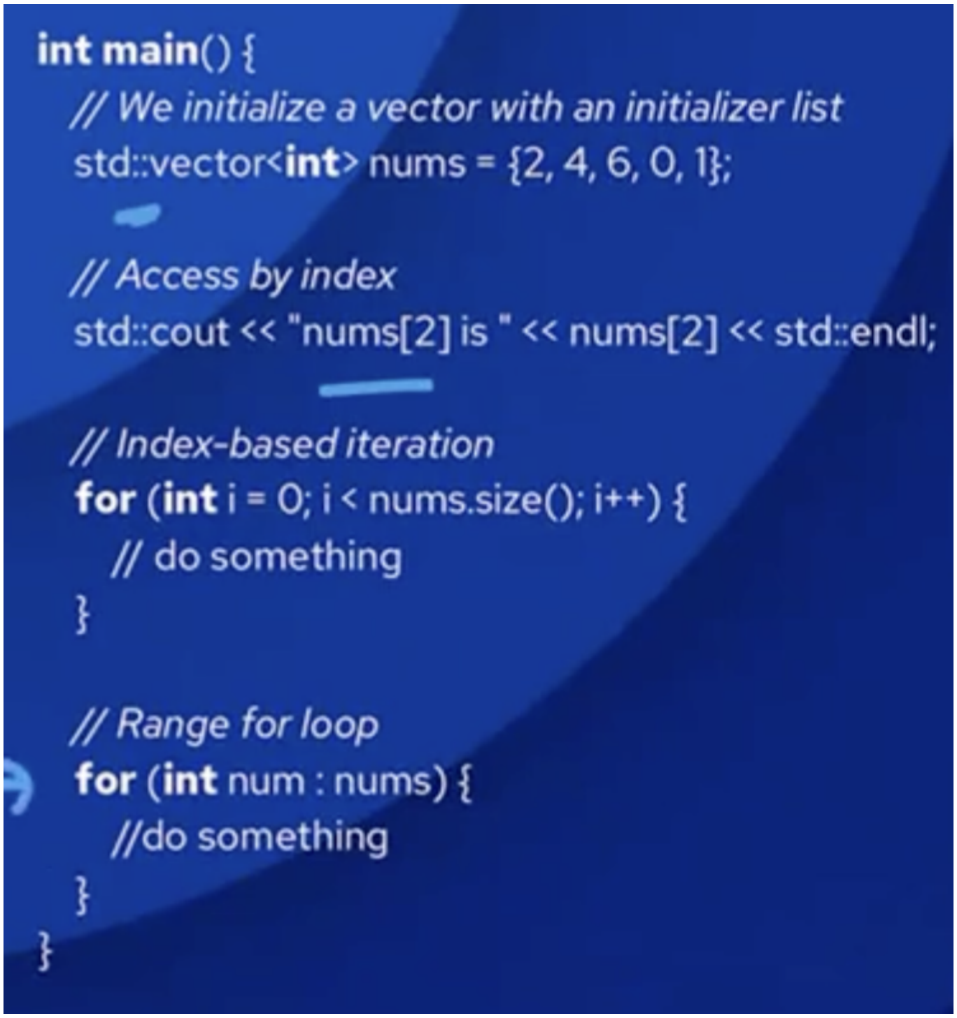
- Pass by value
- Pass by pointer
    - Pass in a pointer to a variable, instead of a copy of the variable value

    - Directly change th value stored at the passed in address
- Pass by Reference
    - Pass in areference to a variable, instead of copy, directly change, cleaner than pass by pointer

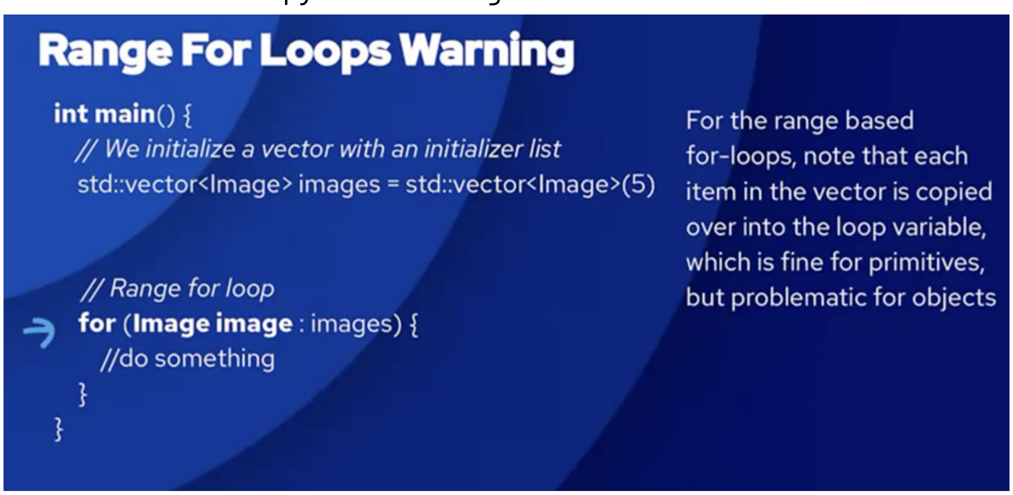


Vectors
- Std::vector

- Ordered list of items, similar to a Java ArrayList



- Not efficient to copy over the image each time



.cpp vs .h

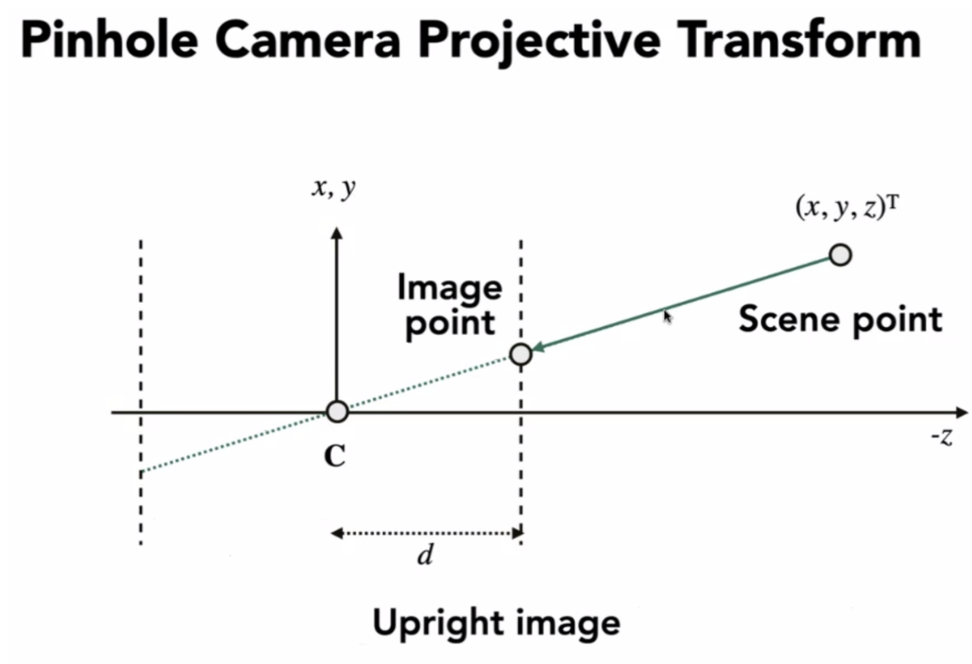
- .cpp: Source code
    - Write code and logic in .cpp files
- .h: header file, included in .cpp files
    - Describe functions or classes/methhods using function declarations

Week 3: Lecture 5 Texture Mapping (2/1)

Perspective in Art

- Learned converging lines and vanishing point

Pinhole Camera Model

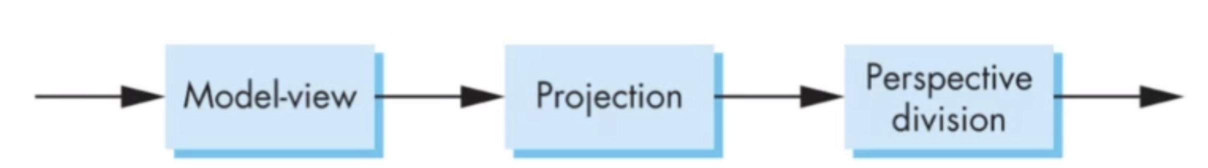


Homogenous Coordinates (3D)

$$\mathbf{p} = \begin{pmatrix} wx \\ wy \\ wz \\ w \end{pmatrix} \longleftrightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$$

Note non-zero term in final row.
First time we have seen this.

$$\mathbf{q} = \mathbf{M} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} \longleftrightarrow \begin{pmatrix} xd/z \\ yd/z \\ d \\ 1 \end{pmatrix}$$



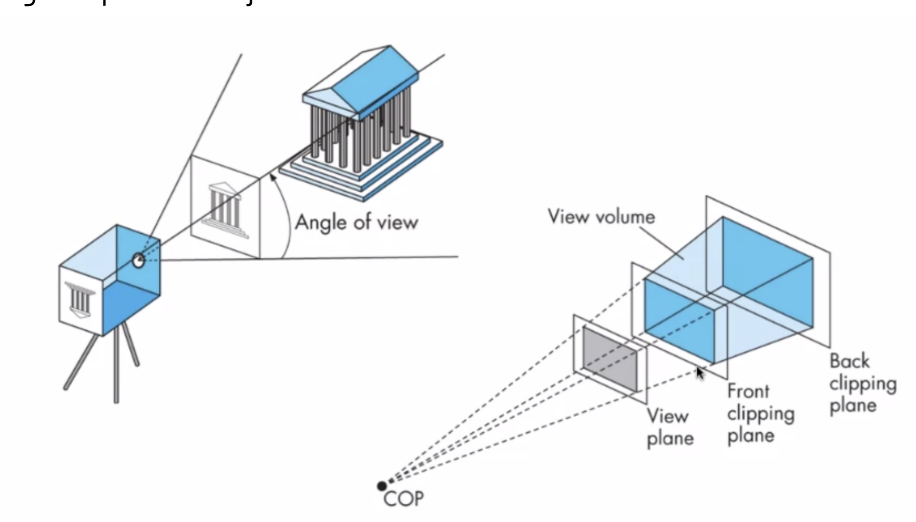
Produces all linear perspective effects,

- Converging lines + vanishing points
- Closer objects appear larger in images

Specifying real-world parameters

- Perspective composition = camera position + angle of view

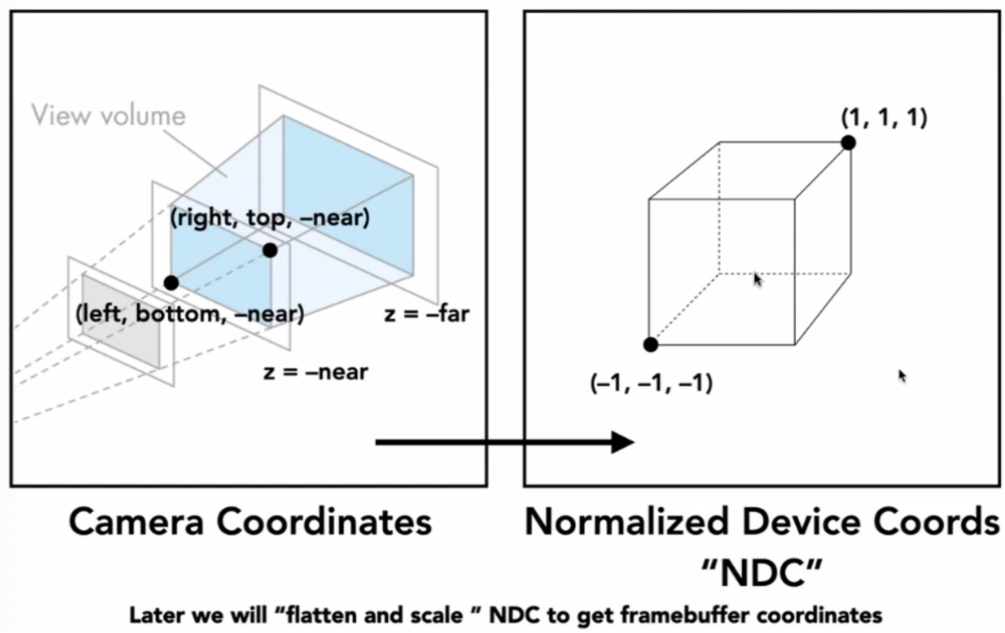
Specifying Perspective Projection



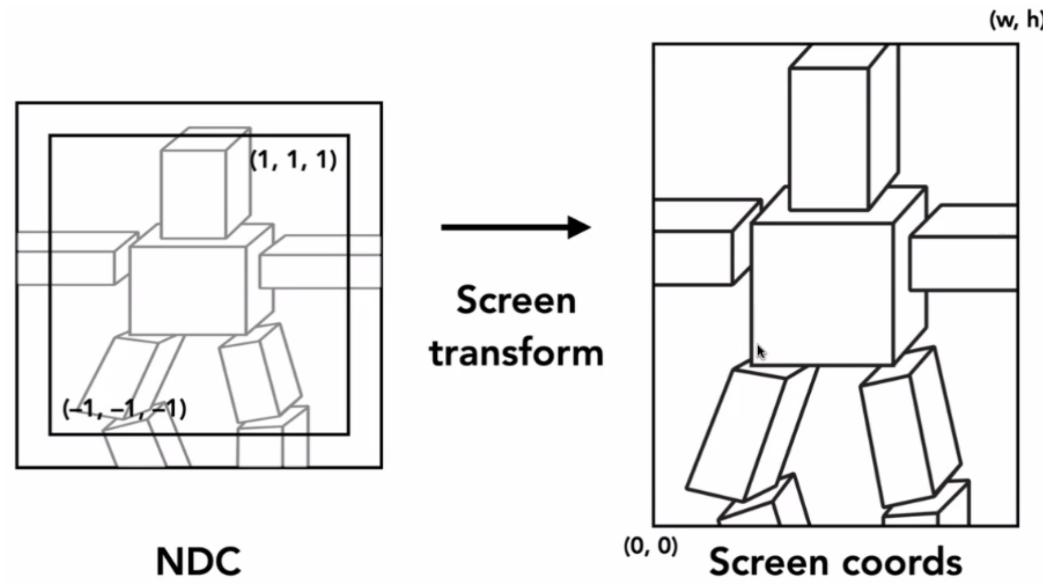
From Angel and Shreiner, Interactive Computer Graphics

-

Perspective Projection Transform

- 

- Coordinate Systems
  - Object coordinates
    - Apply modeling transforms…
  - World (scene) coordinates
    - Apply viewing transform…
  - Camera (eye) coordinates
    - Apply perspective transform + homog. divide…
  - Normalized device coordinates
    - Apply 2D screen transform…
  - Screen coordinates

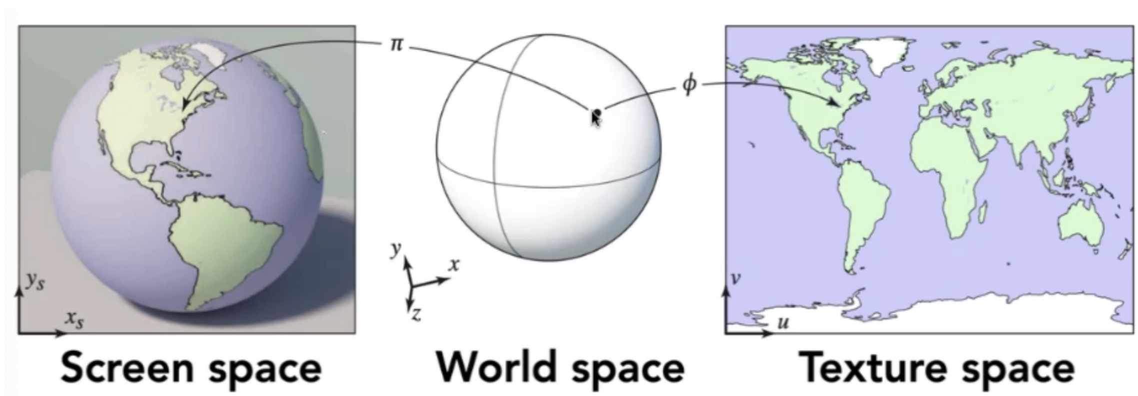


Texture Mapping

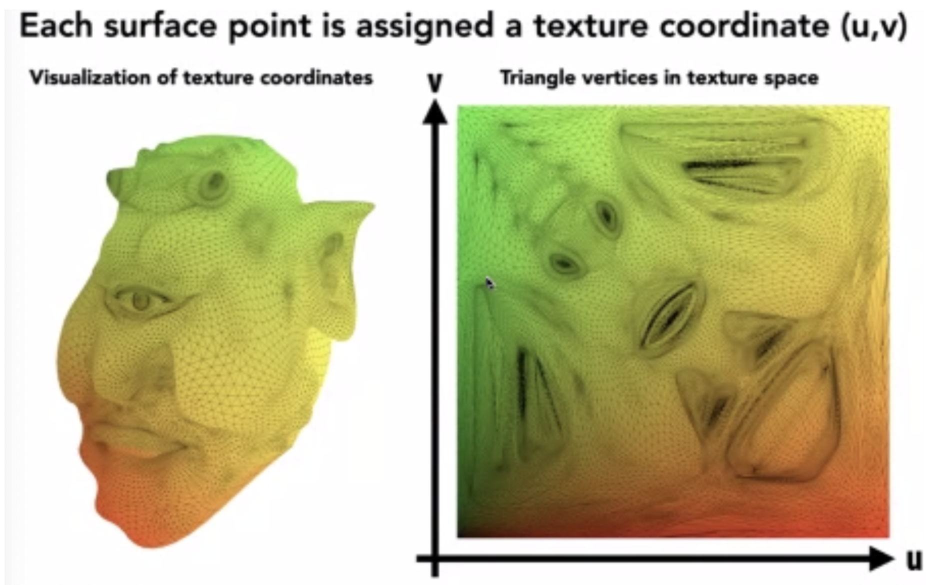
Describe Surface Material Properties

- Add details without raising geometric complexity
- Paste image onto geometry or define procedurally

Texture coordinate mapping

- 

Each surface point was assigned a texture coordinate (u, v)

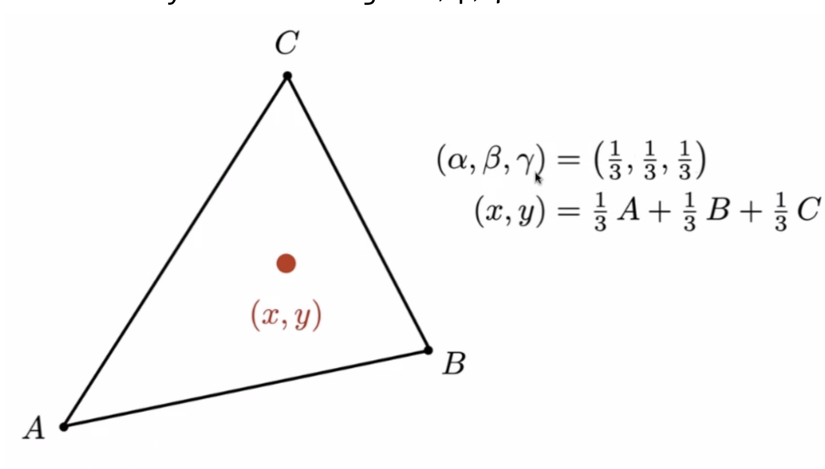


Interpolation across triangles

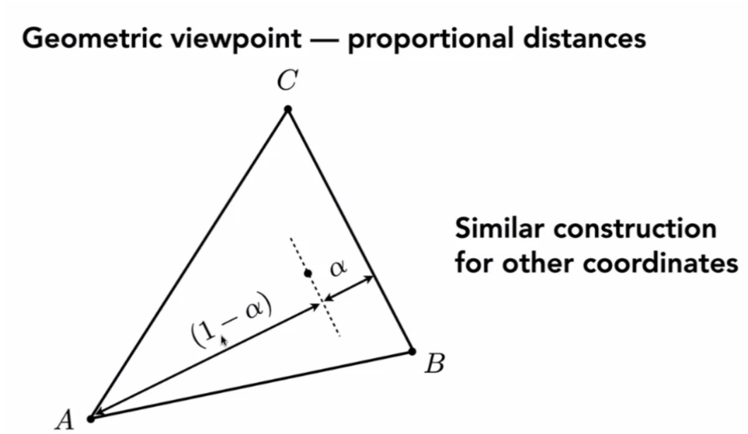
- Barricentric Coordinates
- Specify values at vertices and obtain smoothly varying values across surfaces
- What do we want to inerpolate
    - Texture coordinates, colors, normal vectors

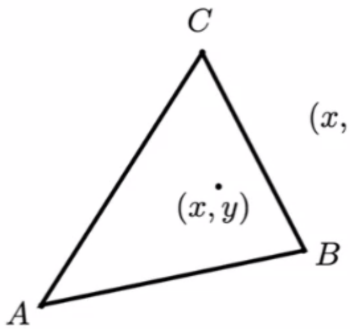
Barycentric Coordinates

- Coordinate system for triangles α, β, γ



-





$$(x, y) = \alpha A + \beta B + \gamma C$$
$$\alpha + \beta + \gamma = 1$$

$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$
$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}$$
$$\gamma = 1 - \alpha - \beta$$

Perspective Projection and Interpolation
- Linear interpolation in world coordinates yields nonlinear interpolation in screen coordinates

Applying Textures is Sampling
- **Simple Texture Mapping Operation**

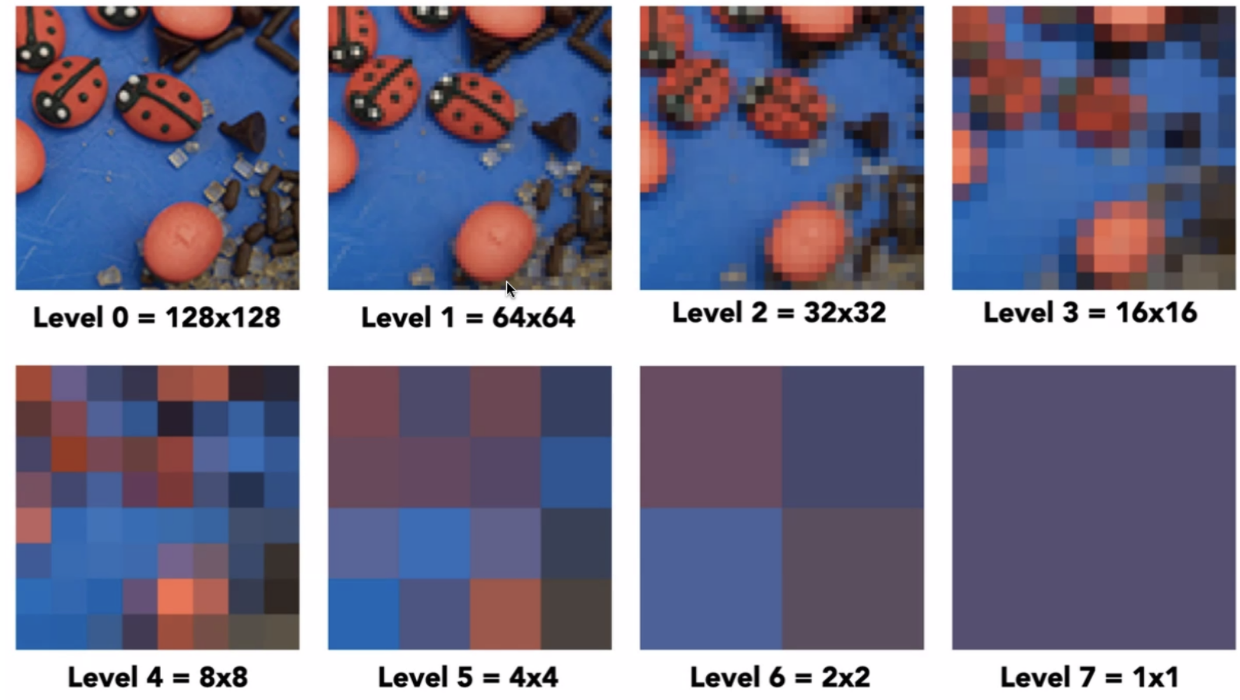
  ```
  for each rasterized screen sample (x,y):
      (u,v) = evaluate texcoord value at (x,y)
      float3 texcolor = texture.sample(u,v);
      set sample's color to texcolor;
  ```

Week 3: Lecture 6 Texture Mapping (2/3)

Texture Minification - Hard Case
- Many texels can tributo pixel footprint
- Idea;
    - Take texture image file, low-pass filter it
    - Fewer pixels

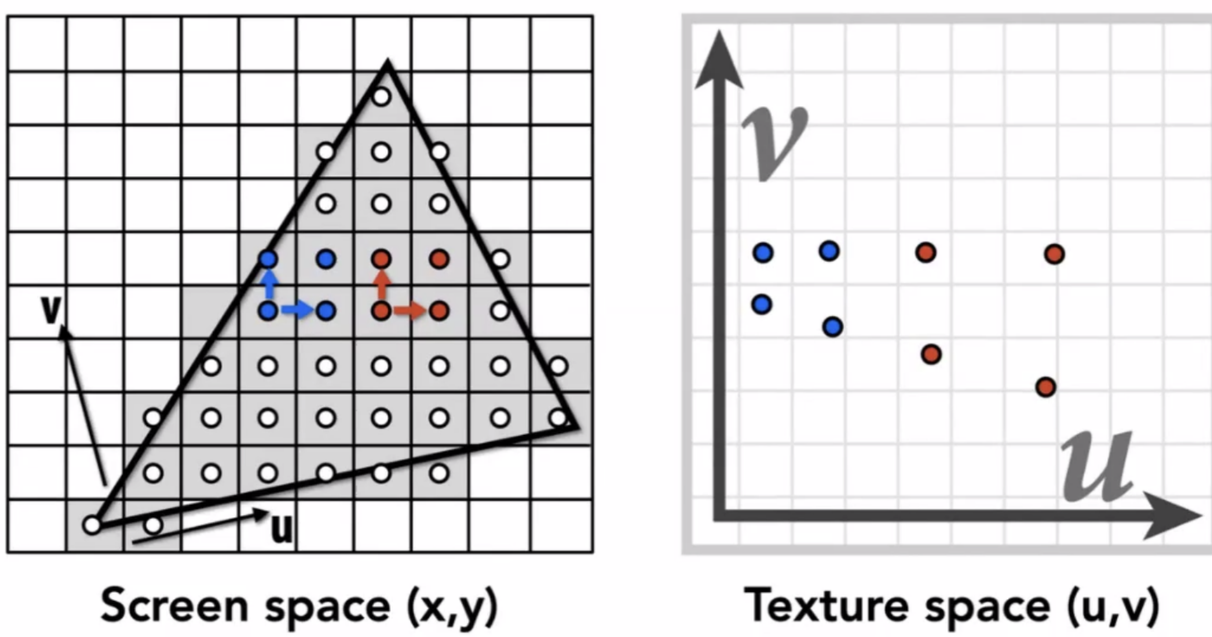
- Blur in the front, but background no aliasing

Mipmap (L. Williams 83)



Computing Mipmap level D

- Screen space (x, y) , Texture space (u, v)

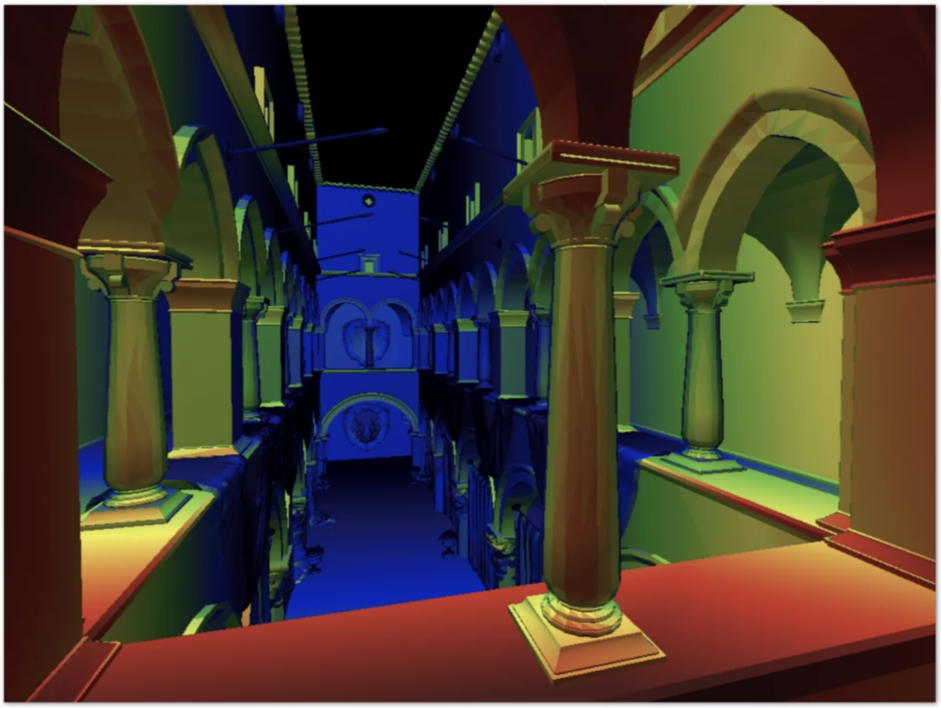


$$D = \log_2 L$$

$$L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

Visualization of Mipmap level

- Continuous D value

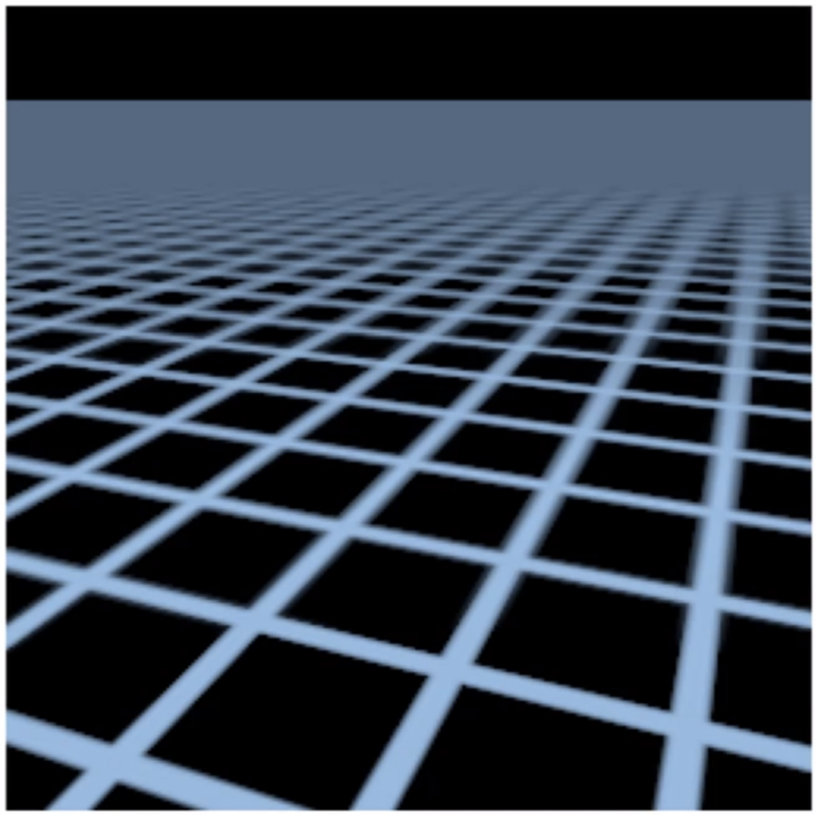


Trilinear filtering: visualization of continuous D

- Interpolate between them
    - Bilinear vs Trilinear Filtering Cost
        - Bilinear resampling
            - 4 texel reads
            - 3 lerps (3 mul + 6 add)
        - Trilinear resampling
            - 8 texel reads
            - 7 lerps (7 mul + 14 add)

Mipmap Limitations

- Overblur in mipmap trilinear sampling



Uses for texting

- GPUs, texture = memory + filtering

Environment Map

- Function from sphere to colors, stored as a texture

Cube Map

- Vector maps to cube points along that direction, textured with 6 square texture maps
- Give a displacement map to give non perfect
- Bump mapping to put displacement mapping, faster

Things to remember texture mapping

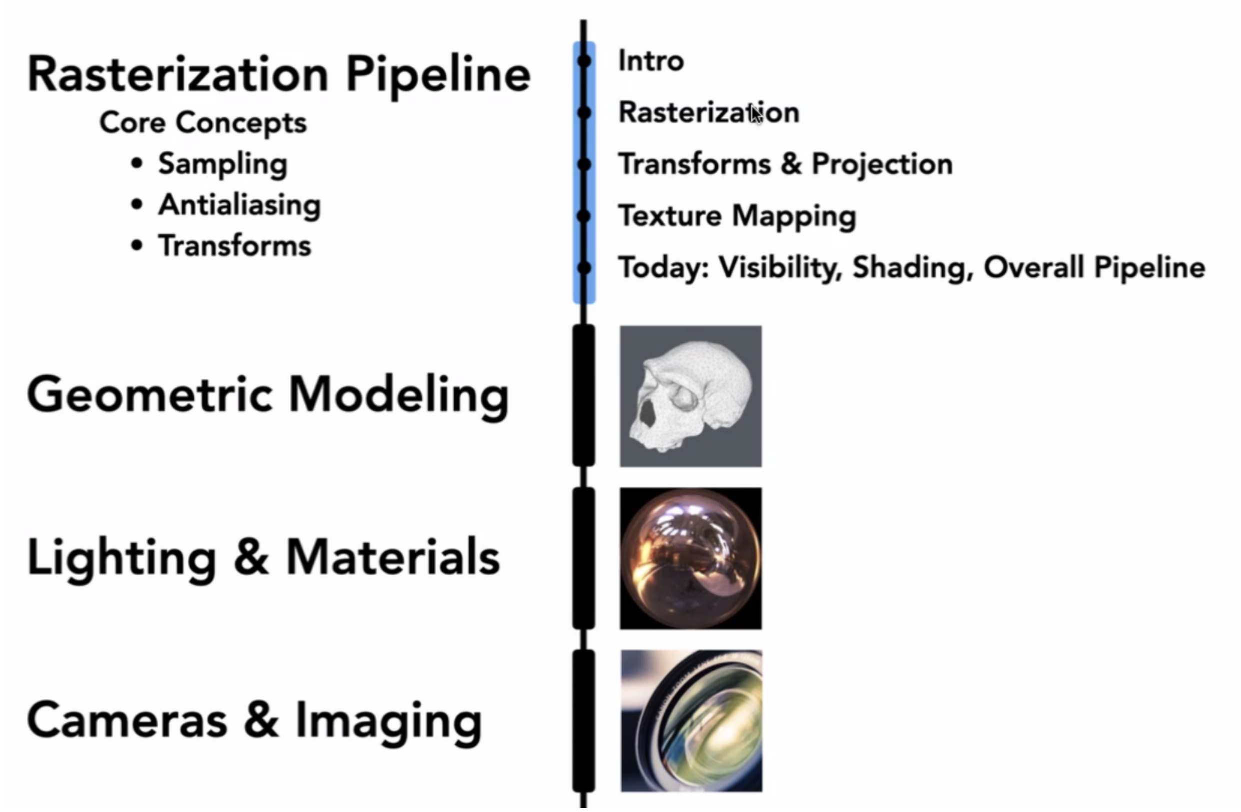
Many uses of texturing

- **Bring high-resolution data to fragment calculations**
- **Colors, normals, lighting on sphere, volumetric data, ...**

How does texturing work?

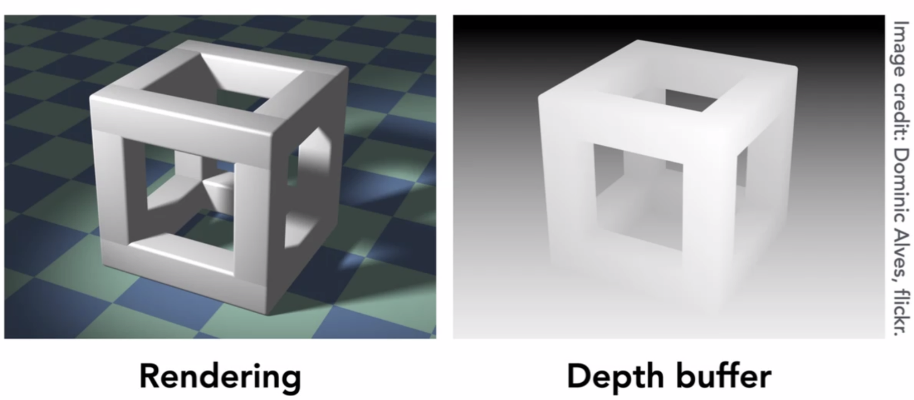
- **Texture coordinate parameterization**
- **Barycentric interpolation of coordinates**
- **Texture sampling pattern and frequency**
- **Mipmaps: texture filtering hierarchy, level calculation, trilinear interpolation**
- **Anisotropic sampling**

The Rasterization Pipeline
Surface representations
Lighting and materials



Painter's Algorithm
- Back to front, overwrite in the frame buffer

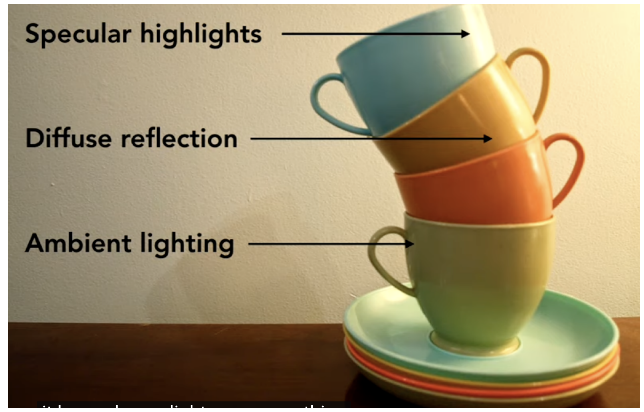
Z-Buffer
- *Hidden surface removal* algorithm that eventually won
- Store current min z value for each sample position



0-1 for the pixel depth close/far from camera

-

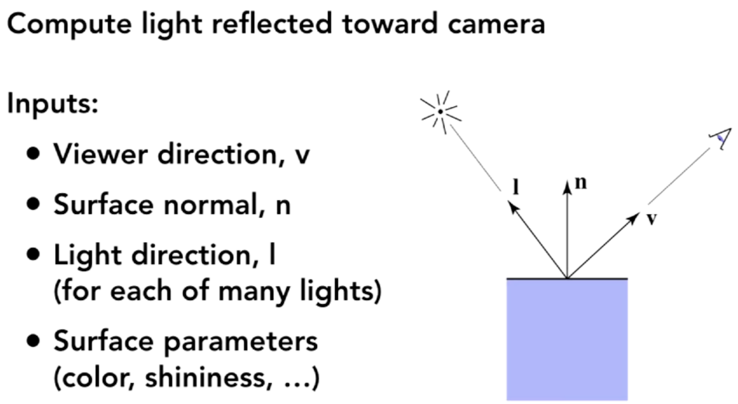
Week 4: Lecture 7 Shading, Geometry, Splines (2/8)



Shading usually takes a lot of computation

Local Shading

-



Diffuse Reflection

- Light is scattered uniformly in all direction
- Surface color is the same for all viewing directions

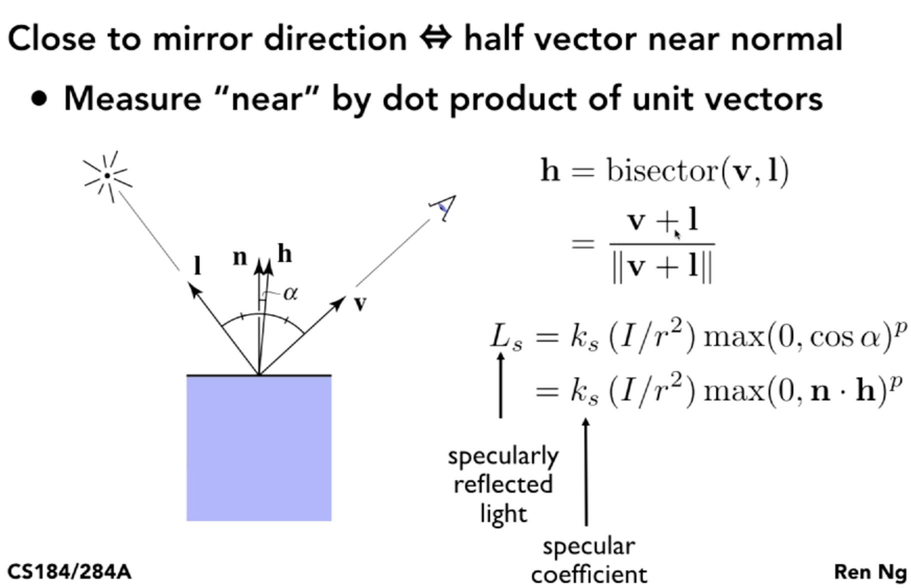
Light Falloff

- Intensity = $\frac{I}{r^2}$

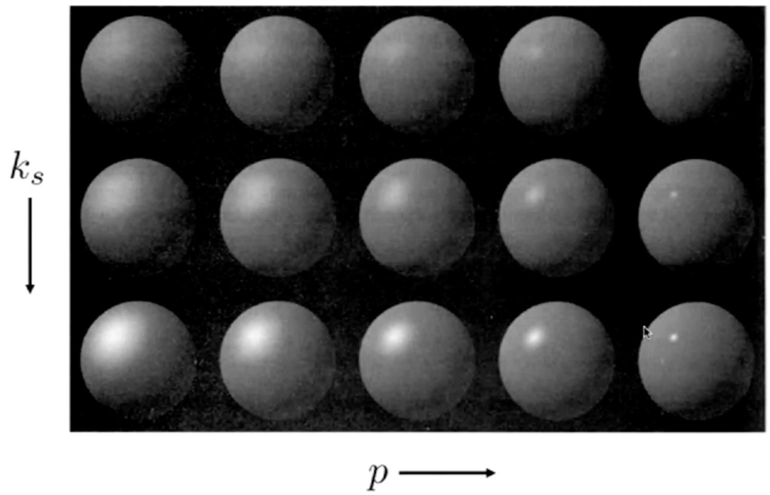
Lambertian Shading

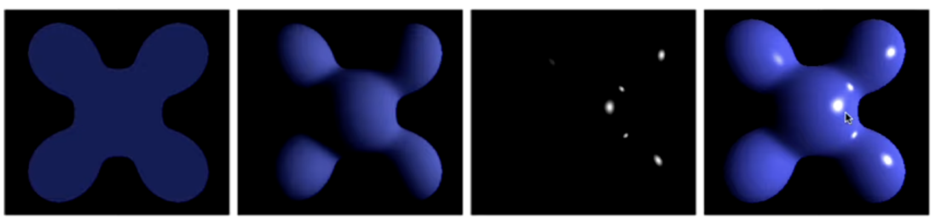
- $L_d = k_d(I/r^2)\ \max(0, n * 1)$
- k_d: diffuse coefficient
- L_d: diffusely reflected light

Specular Shading (Blinn-Phong)



$$L_s = k_s\,(I/r^2)\max(0, \mathbf{n}\cdot\mathbf{h})^p$$





$$\begin{aligned} L &= L_a + L_d + L_s \\ &= k_a\,I_a + k_d\,(I/r^2)\max(0, \mathbf{n}\cdot\mathbf{l}) + k_s\,(I/r^2)\max(0, \mathbf{n}\cdot\mathbf{h})^p \end{aligned}$$

Shade each pixel (“Phong” shading)

- Can get normal vector

Shader Programs

- Progam vertex and garment processing stages
- Describe operation on a single vertex

GPU: Heterogeneous, Multi-Core Processor

- Tera-Op’s fixed function compute capability over here

<u>Geometry Bezier Curve Splines</u>

Many Ways to represent geometry

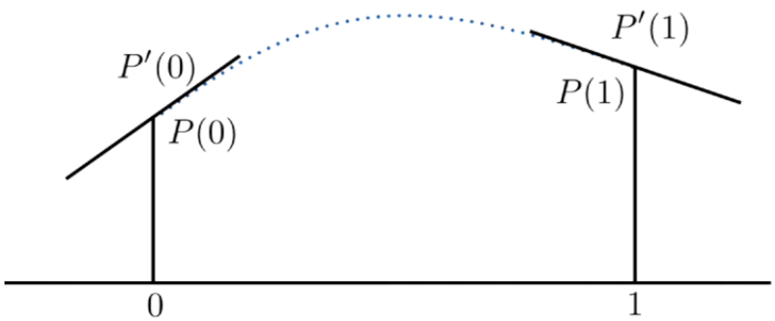
- Explicit
  - Point cloud
  - Polygon mesh
  - subdivision , NURBS

- Implicit
    - Level sets, algebraic surface, distance functions

Spline Topics

- Interpolation
    - Cubic hermite interpolation
    - Catmull rom interpolation

Cubic Hermite Interpolation



- **Inputs: values and derivatives at endpoints**
- Cubic Polynomial Interpolation: $P(t) = at^3 + bt^2 + ct + d$
-

$$
\begin{aligned}
h_0 &= d \\
h_1 &= a + b + c + d \\
h_2 &= c \\
h_3 &= 3a + 2b + c
\end{aligned}
$$

$$
\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}
$$

$$
\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix}
$$

$$
= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix}
$$

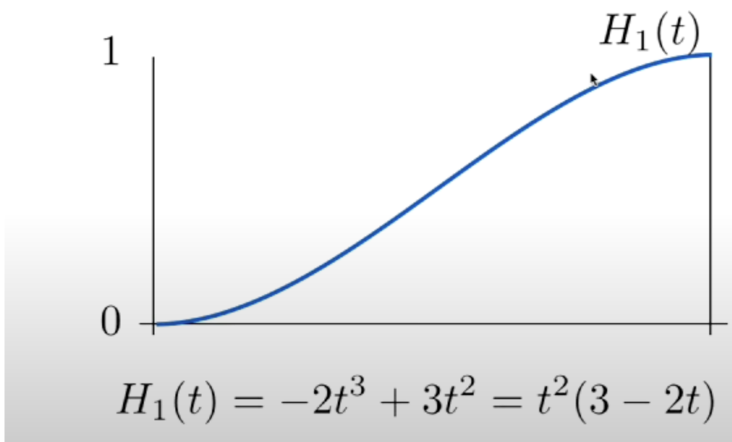
Matrix form of hermite function

- Either basis can represent any cubic polynomial through linear combination

Ease function

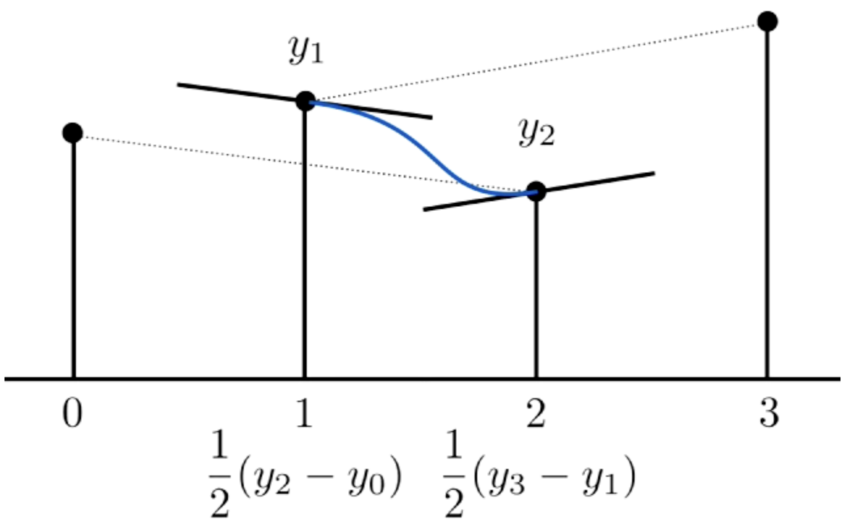
- Very useful function, start and stop gently (zero velocity)

-



Catmull-Rom Interpolation

- Rule for derivatives : match slope between previous and next values



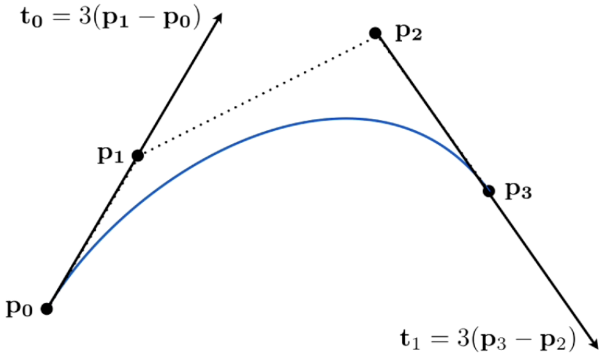
- Catmull-Rom matrix form

$$P(t) = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}^T \begin{bmatrix} -\frac{1}{2} & \frac{3}{2} & \frac{3}{2} & \frac{1}{2} \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

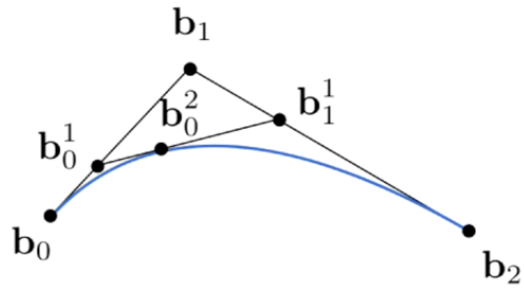
$$= C_0(t)\ \mathbf{p}_0 + C_1(t)\ \mathbf{p}_1 + C_2(t)\ \mathbf{p}_2 + C_3(t)\ \mathbf{p}_3$$

- **Matrix columns = Catmull-Rom basis functions**

Bézier curves with tangents



-

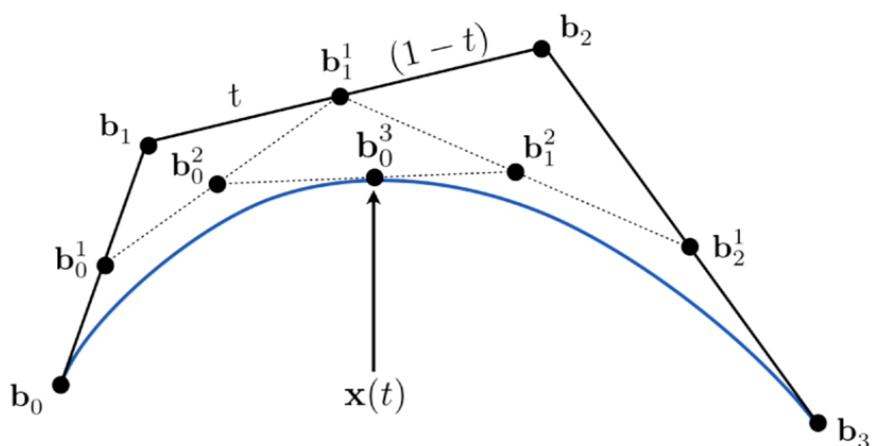


4 points used to define tangent and create bezier curve

De Casteljau Algorithm

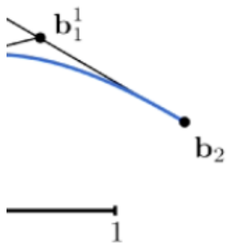
- Insert a point using linear interpolation
- "Corner cutting" recursive subdivision
- Successive linear interpolation

Consider four points
Same recursive linear interpolations



-

- Can apply in different dimensions



$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$$
$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1 + t\mathbf{b}_2$$

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1$$

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2$$

- Final point on bezier curve

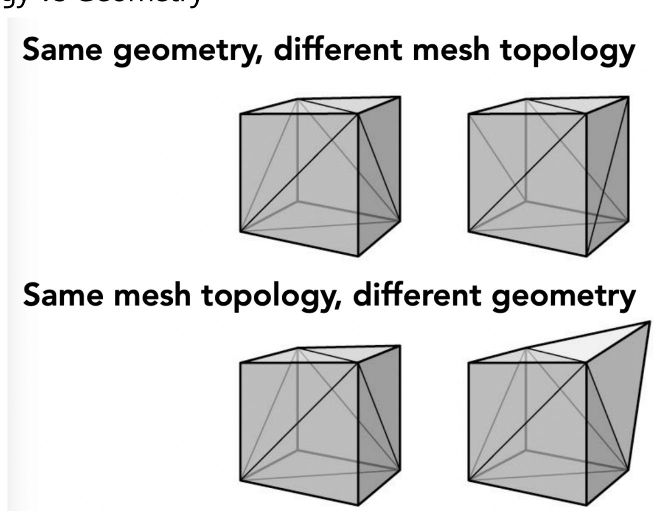
Bernstein form of a Bézier curve of order n:

$$\mathbf{b}^n(t) = \mathbf{b}_0^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t)$$

Bezier surfaces
- Bicubic bezier patches
- 2D de casteljau algorithm

Week 4: Lecture 8 Mesh Representations & Geometry Processing (2/10)

Mesh Representations
- Triangles, points + triangles

Topology vs Geometry
- 

Topological Validity: manifold

- Def: 2D manifold is surface when cut with a small sphere always yields a disk
    - Mesh if manifold, useful properties
    - Edge connects two faces
    - Edge connects two vertices
    - Face consists of a ring of edges and vertices
    - Vertex consists of a ring of edge and faces
    - Euler: F - E + V = 2

Triangle-Neighbor Data Structure

- Struct Tri {
    - Vert *v[3]
    - Tri *t[3]
- }

Half-Edge Facilitates Mesh Traversal

- Process vertex, edge, face points to go through a face

Half-Edge : Edge Flip

- Long list of pointer reassignments

Subdivision Surfaces

- Coarse control to manipulate a smooth curve
- Start with carse polygon mesh
    - Subdivide each element
- Interpolating or approximating
- Continuity at vertices

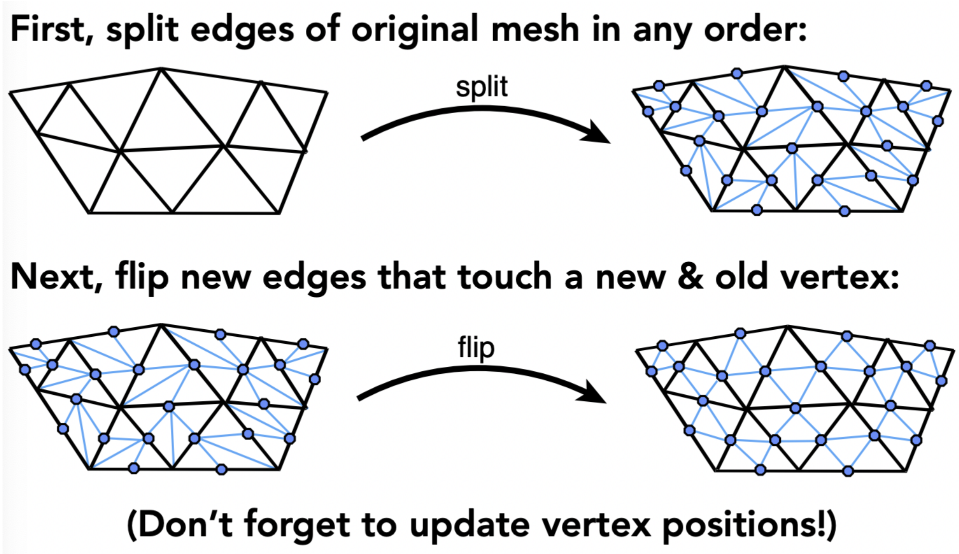
Core Idea: Let subdivision define the surface

    - Evaluation by subdivision
    - Evaluation by algebra
- Insight that leads to subdivision surfaces

Loop Subdivision

- C2 smoothnessa way from extraordinary vertices

- Move all edges that touch a new & old vertex and reset it



-

Catmull-Clark Subdivison

- Each subdivision step
- Add vertex in each face
- Add midpoint on each edge
- Connect all new vertices

Catmull-Clark Vertex Update Rules (General Mesh)

- f = average of surrounding vertices

Sharp Creases

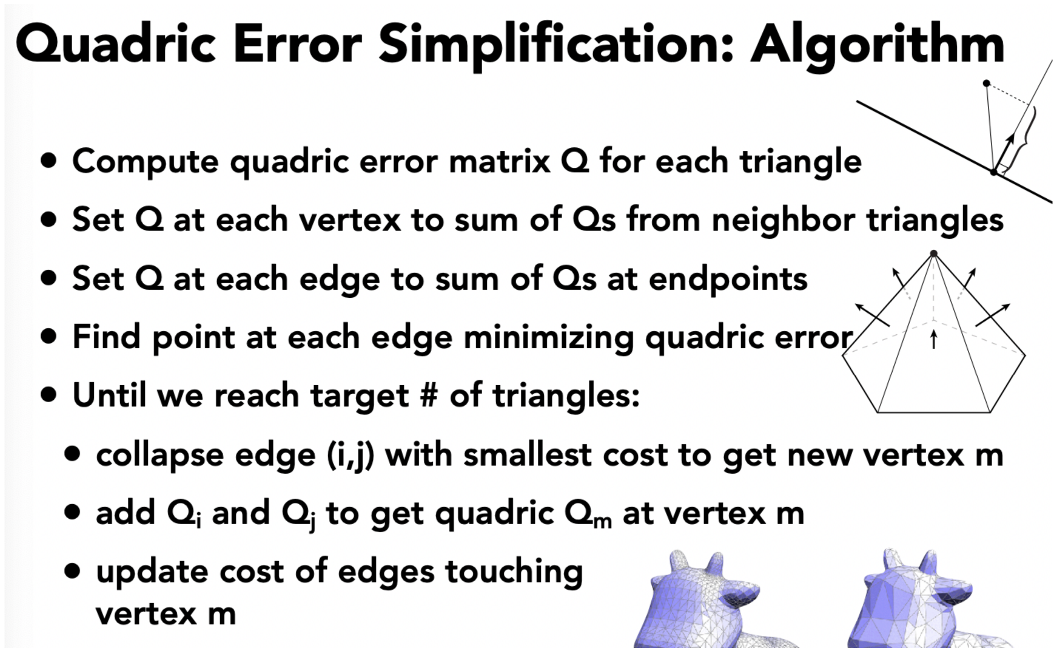
- Make some edges always high frequency

Mesh Simplification

- Goal: reduce number of mesh elements without maintain overall shape

How do We resample meshes

- Iteratively collapse edges
- Edge split is local upsampling
- Assign score with quadrice error metric
- Approx distance to surface as sum of distance to planes containing triangles
- Iteratively collapse edge with smallest score
- Greedy algorithm
- Score: surface displacement, symmetric matrix endcodes distance to plane
- Quadric Error at vertex: approx distance to vertex triangles as sum of distances to each triangles plane. Encode this as a sinngle quadric matrix for the vertex



What makes a good triangle mesh
- Delaunay
- Circumcircle interiors contain no vertices

Make triangle more round
- Center vertex
-

Week 5: Lecture 9 Intro to Ray-Tracing (2/15)

Towards Photorealistic rendering

Basic Ray tracing
1. Generate an image by casting one ray per pixel
2. Check for shadows by sending a ray to the light

Generating Eye Rays
- Eye ray -> light source

Recursive Ray Tracing
- Go from light source to all the other objects
- Specular reflection is mirror ray
- Refractive rays (specular transmission)

Building Eye Rays
- Ray Equation
  - $R(t) = E + t(P - E)$
  - $t \in [1... + \infty]$
  - Through eye at $t = 0$

Shadow Rays
- Test for occluder

        - Shade normally
        - Skip light
    - Self shadowing
    - Recursion depth, truncate at fixed number of bounces

Ray Equation

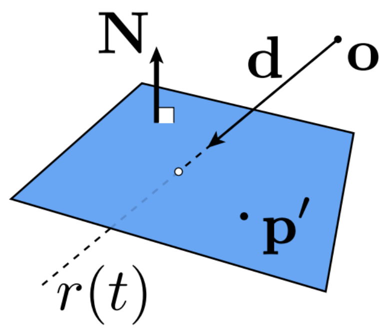
- Ray defined by its origin and direction vector

Ray equation:

$$\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d},\ 0 \le t < \infty$$

Plane equation:

$$\mathbf{p} : (\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = 0$$



Solve for intersection

Set $\mathbf{p} = \mathbf{r}(t)$ and solve for t

$$(\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = (\mathbf{o} + t\,\mathbf{d} - \mathbf{p}') \cdot \mathbf{N} = 0$$

$$t = \frac{(\mathbf{p}' - \mathbf{o}) \cdot \mathbf{N}}{\mathbf{d} \cdot \mathbf{N}}$$ **Check:** $0 \le t < \infty$

-

Can Optimize: e.g. Möller Trumbore Algorithm

$$\vec{\mathbf{O}} + t\vec{\mathbf{D}} = (1 - b_1 - b_2)\vec{\mathbf{P}}_0 + b_1\vec{\mathbf{P}}_1 + b_2\vec{\mathbf{P}}_2$$

$$\begin{bmatrix} t \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{\vec{\mathbf{S}}_1 \bullet \vec{\mathbf{E}}_1} \begin{bmatrix} \vec{\mathbf{S}}_2 \bullet \vec{\mathbf{E}}_2 \\ \vec{\mathbf{S}}_1 \bullet \vec{\mathbf{S}} \\ \vec{\mathbf{S}}_2 \bullet \vec{\mathbf{D}} \end{bmatrix}$$

Where:

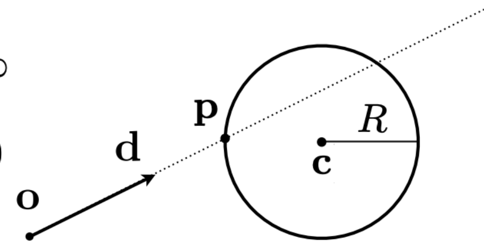
$$\vec{\mathbf{E}}_1 = \vec{\mathbf{P}}_1 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{E}}_2 = \vec{\mathbf{P}}_2 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}} = \vec{\mathbf{O}} - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}}_1 = \vec{\mathbf{D}} \times \vec{\mathbf{E}}_2$$
$$\vec{\mathbf{S}}_2 = \vec{\mathbf{S}} \times \vec{\mathbf{E}}_1$$

Cost = (1 div, 27 mul, 17 add)

Ray Intersection With Sphere

Ray: $\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}, \; 0 \le t < \infty$

Sphere: $\mathbf{p} : (\mathbf{p} - \mathbf{c})^2 - R^2 = 0$

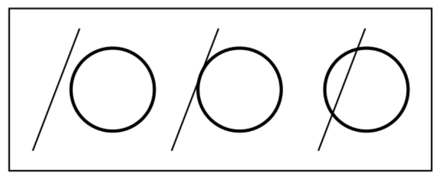


Solve for intersection:

$$(\mathbf{o} + t\,\mathbf{d} - \mathbf{c})^2 - R^2 = 0$$

$$a\,t^2 + b\,t + c = 0, \text{ where}$$
$$a = \mathbf{d} \cdot \mathbf{d}$$
$$b = 2(\mathbf{o} - \mathbf{c}) \cdot \mathbf{d}$$
$$c = (\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - R^2$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



Bound each object with a bounding box and test bvol first, then test object if it hits

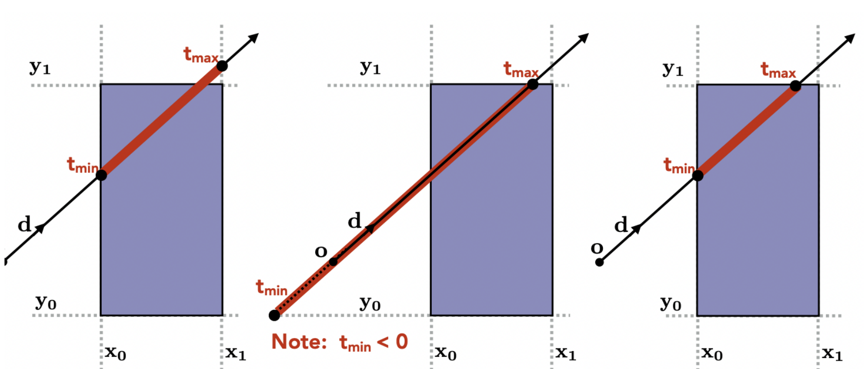
Week 5: Lecture 10 Ray-Tracing (2/17)

Ray Tracing - Acceleration

- Pixels * log(# objects)

Bounding Volumes

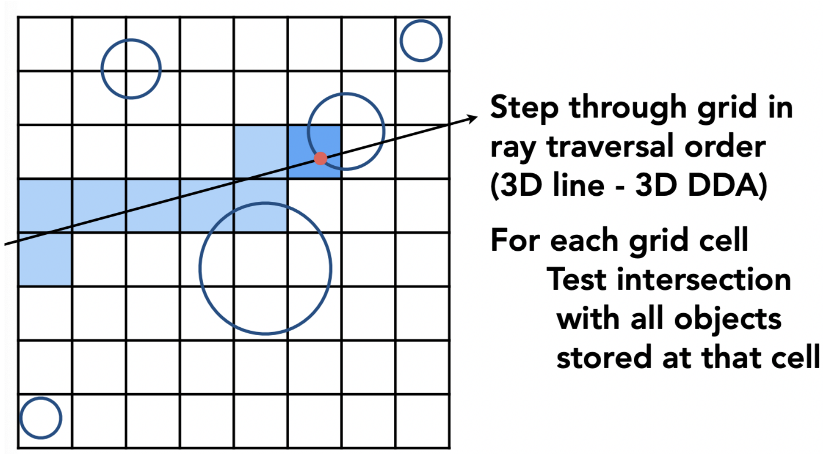
- Axis aligned bounding box
- Find



Intersections with x planes | Intersections with y planes | Final intersection result

- Ray intersection with Axis aligned base
- When it enters and exits the box

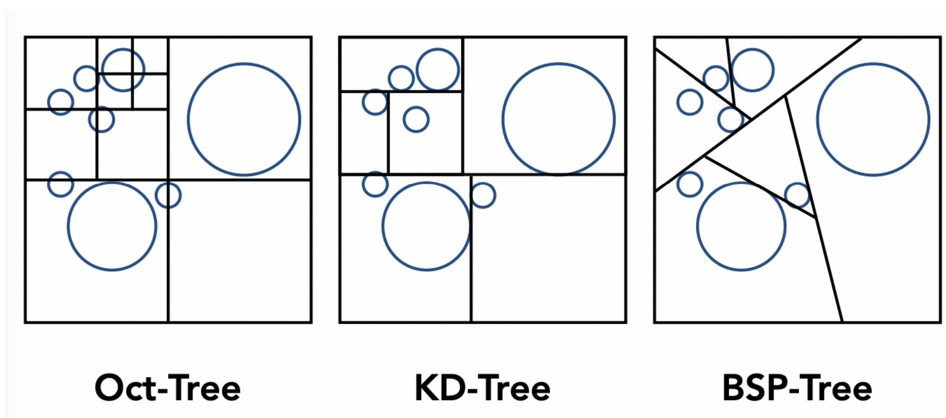
Uniform Spatial Partitions (Grids)



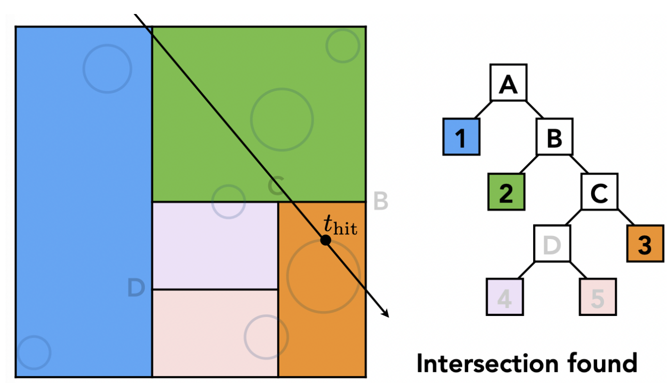
- Check if intersection is inside the cell

Uniform Grids - when they fail
- Teapot in a stadium

Non-Uniform Spatial Partition: Spatial Hiearchies



KD Trees



- Internal nodes store
  - Split axis: x-, y-, z-
  - KD Tree Preprocessing
- Pre-processing
  - Choosing the split plane
    - Simple: midpoint, median split
    - Ideal: split to minimize expected cost of ray intersection
- Terminiation Criteria
  - Simple: common to prescribe maximum tree depth
  - Ideal: stop when splitting does not reduce respected cost of ray intersection

<u>Object Partitions & Bounding Volume Hierarchy (BVH)</u>

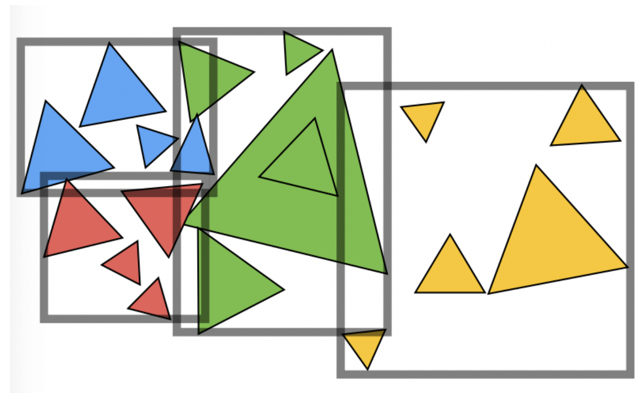
Spatial Parition (KD-tree)

- Partition space into non-overlapping regions
- Objects can be contained in multiple regions

Object partition (BVH)

- Partition set of objects inot disjoin subsets
- Bounding boxes for heaps

BVH

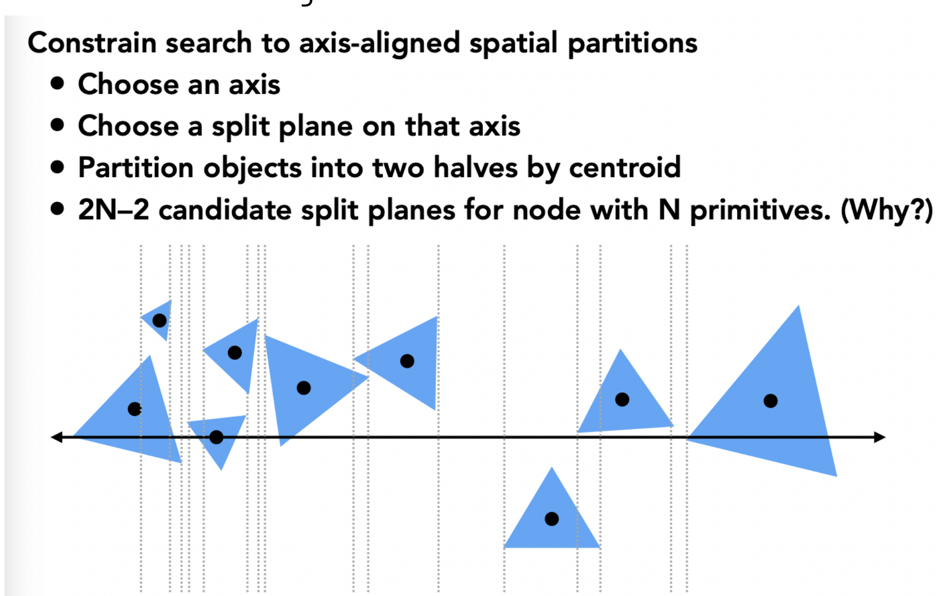


- BVH Preprocessing
    - Find bounding box
    - Recursively split into set of objects in two subsets
- BVH Recursive Traversal
    - If ray misses node.bbox : return
    - If node is leaf node
        - Test intersection with all objs
        - Return closester intersection
    - Hit1 = intersect(ray, node.child1)
    - Hit2 = intersect(ray, node.child2)
    - Return closer of hit1, hit2

How to split into two sets (BVH)

- Want a split that works best spatial, sometimes almot all triangles in one side sometimes half half

Which Hierarchy is fastest

- A good partition minimzes the average cost of tracing a ray
    - csot(node)
- Surface Area Estimating the cost with a heuristic
- 

Things to remember

- Linear vs log ray-intersection techniques
- Many techniques for accelerating ray-intersection

Week 6: Lecture 11 Radiometry/Photometry (2/22)

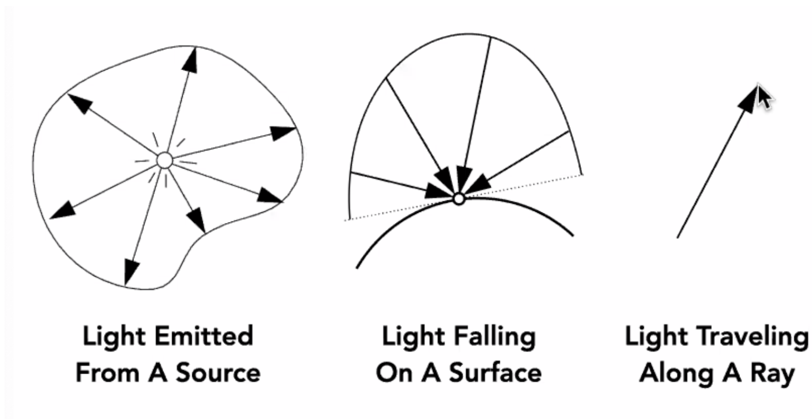
Lights

Radiant Energy and Flux (Power)

- Radiant energy is the energy of electromagnetic radiation in Joules
    - Q [J = Joule]
- Radiant flux is energy emitted, reflected, transmitted or received, per unit time

Photometry

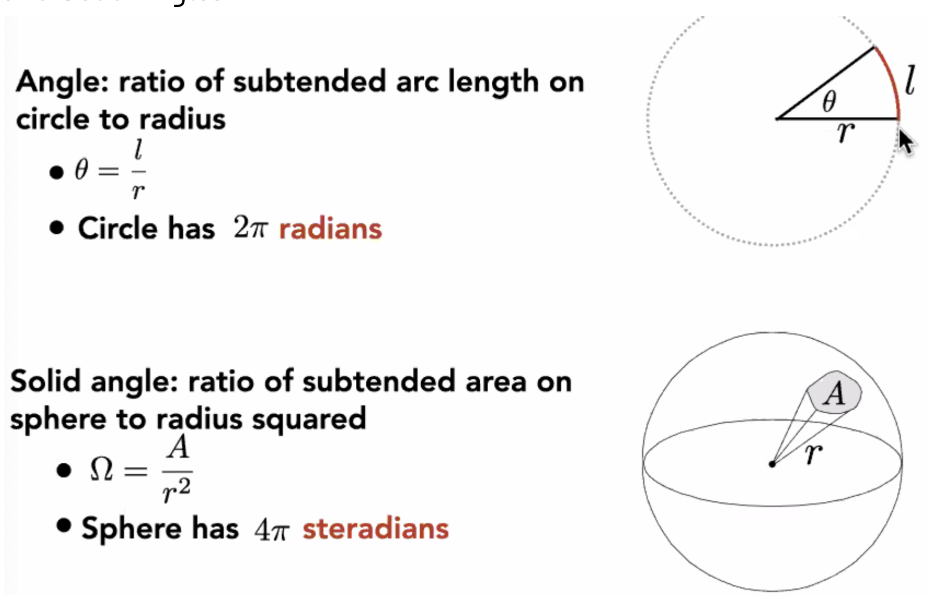
- Accounts for the response of the human visual system



Radiant Intensity

- Radiant intensity is power per unit solid angle emitted by a point light source
- $I(\omega) \;=\; \frac{d\phi}{d\omega}$

Angles and Solid Angles



- $d\omega = \frac{dA}{r^2} = \sin\theta d\theta d\phi$
- Very important for rendering when you are far away

Radiance

1. Fundamental field quantity that describes the distribution

Surface Radiance

- Reflected radiance is power emitted, reflected, transmitted or received by a surface, per unit solid angle, per unit projected area

$$L(\mathrm{p},\omega) \equiv \frac{\mathrm{d}^2\Phi(\mathrm{p},\omega)}{\mathrm{d}\omega\,\mathrm{d}A\cos\theta}$$

$\cos\theta$ accounts for projected surface area

$$\frac{\mathrm{W}}{\mathrm{r\,m}^2}\Bigg]\ \left[\frac{\mathrm{cd}}{\mathrm{m}^2} = \frac{\mathrm{lm}}{\mathrm{sr\,m}^2} = \mathrm{nit}\right]$$

Incident & Exiting surface radiance differ

- Distinguish between incident radiance and exitant radiance functions

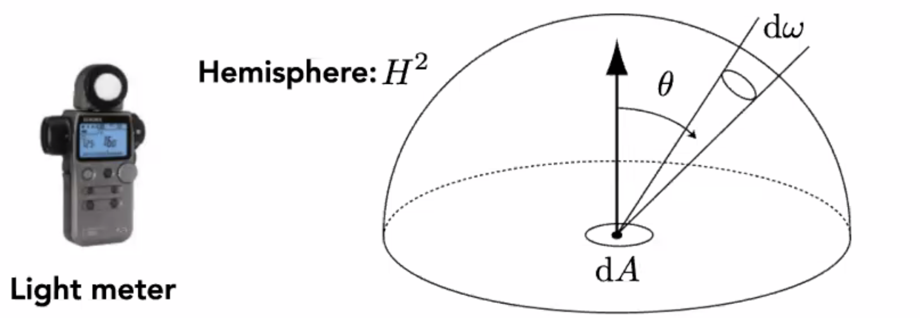
Irradiance from the Environment

Computing flux per unit area on surface, due to incoming light from all directions.

$$dE(\mathrm{p},\omega) = L_i(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega$$

Contribution to irradiance from light arriving from direction ω

$$E(\mathrm{p}) = \int_{H^2} L_i(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega$$



Multi-Camera Array -> 4D light field

Spherical Gantry -> 4D light field

- Take photographs of an object from all points on an enclosing sphere
- Captures all light leaving an object like a hologram
-

Week 6: Lecture 12 Monte Carlo Integration (2/24)

High Dimentional Integration

Complete set of samples: $N = \underbrace{n \times n \times \cdots \times n}_{d} = n^d$

- **"Curse of dimensionality"**

Numerical integration error: $\text{Error} \sim \frac{1}{n} = \frac{1}{N^{1/d}}$

Random sampling error: $\text{Error} = \text{Variance}^{1/2} \sim \frac{1}{\sqrt{N}}$

In high dimensions, Monte Carlo integration requires fewer samples than quadrature-based numerical integration

Monte Carlo Integration
- Idea: estimate integral based on random sampling of function
- Advatanges
  - General and relatively simple method
  - Requires only ufnciton evaluation at any point
  - Good for general functions
  - Efficient for high-dimensional integrals
- Disadvantages
  - Noise: integral estimate is random, only correct "on average"
  - Can be slow to converg need a lot of samples

Random Variables
- X

Probability Distribution Function
- N discrete values x_i with probability p_i
- $\sum_{i=1}^{n} p_i = 1, p_i \geq 0$

Continuous Probability Distribution Function
- $X \sim p(x)$
- Expected value of X $E[X] = \int xP(x)\,dx$

Monte Carlo Integration
- Estimate the integral of a function by averaging random rsamples of the function's value
- Define the monte carlo estimator for the definite integral of given function $f(x)$

Definite integral	$\int_a^b f(x)dx$	
Random variable	$X_i \sim p(x)$	**Note: p(x) must be nonzero for all x where f(x) is nonzero**
Monte Carlo estimator	$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}$	

Basic Monte Carlo Estimator

- Sampel with a uniform random variable
- Uniform random variable
    - $X_i \sim p(x) = C$

Unbiased Estimator

- A randomized integral estimator is unbiased if its expected value is the desired integral

Proof Monte carlo unbiased

$$
\begin{aligned}
E[F_N] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[\frac{f(X_i)}{p(X_i)}\right] \\
&= \frac{1}{N}\sum_{i=1}^{N}\int_a^b \frac{f(x)}{p(x)}\,p(x)\mathrm{d}x \\
&= \frac{1}{N}\sum_{i=1}^{N}\int_a^b f(x)\,\mathrm{d}x \\
&= \int_a^b f(x)\,\mathrm{d}x
\end{aligned}
$$

Properties of expected values:

$$E\left[\sum_i Y_i\right] = \sum_i E[Y_i]$$

$$E[aY] = aE[Y]$$

Expected value of monte carlo is the desired integral

Variance of a RV

Definition

$$V[Y] = E[(Y - E[Y])^2]$$
$$= E[Y^2] - E[Y]^2$$

Variance decreases linearly with number of samples

$$V\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] = \frac{1}{N^2}\sum_{i=1}^{N} V[Y_i] = \frac{1}{N^2} N\, V[Y] = \frac{1}{N} V[Y]$$

Properties of variance

$$V\left[\sum_{i=1}^{N} Y_i\right] = \sum_{i=1}^{N} V[Y_i] \qquad V[aY] = a^2\, V[Y]$$

Direct Lightning Estimate

- Idea: sample directions over hemisphere uniformly in solid angle
- $E(p) = \int L(p,\ \omega)\, cos\,\theta\, d\omega 0$

Given surface point *p*

Initialize Monte Carlo estimator F_N to 0

For each of N samples:

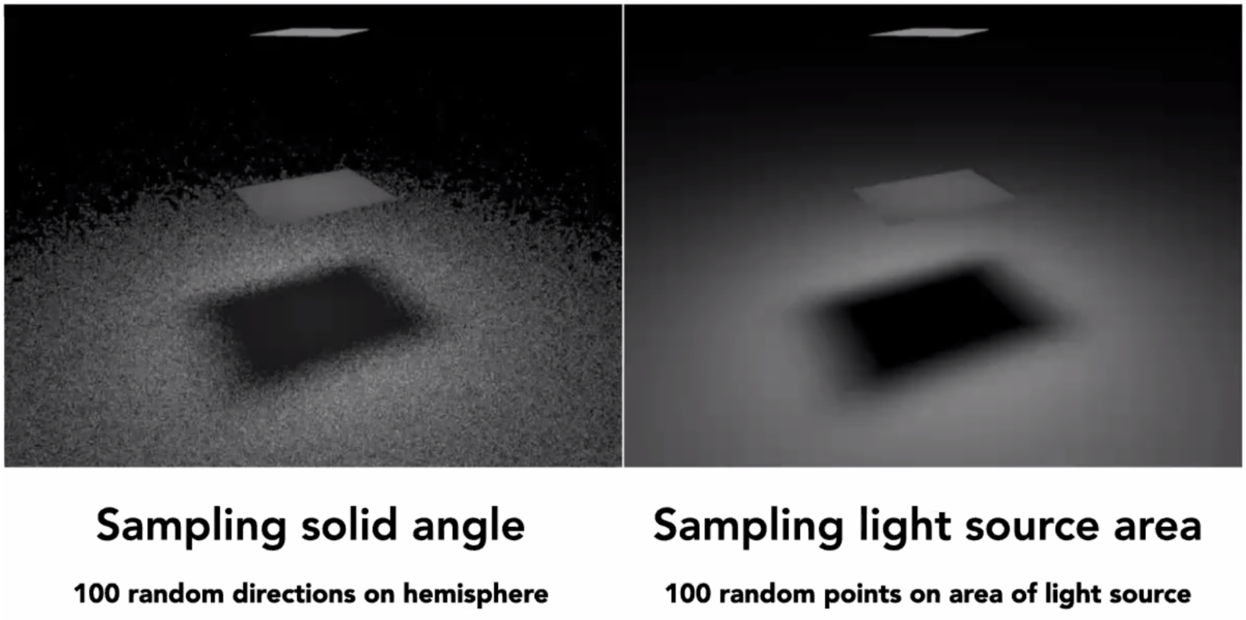
A ray tracer evaluates radiance along a ray

Generate random direction: ω_i

Compute incoming radiance L_i arriving at *p* from direction ω_i

Increment the Monte Carlo estimator: $F_N := F_N + \frac{2\pi}{N} L_i \cos\theta_i$

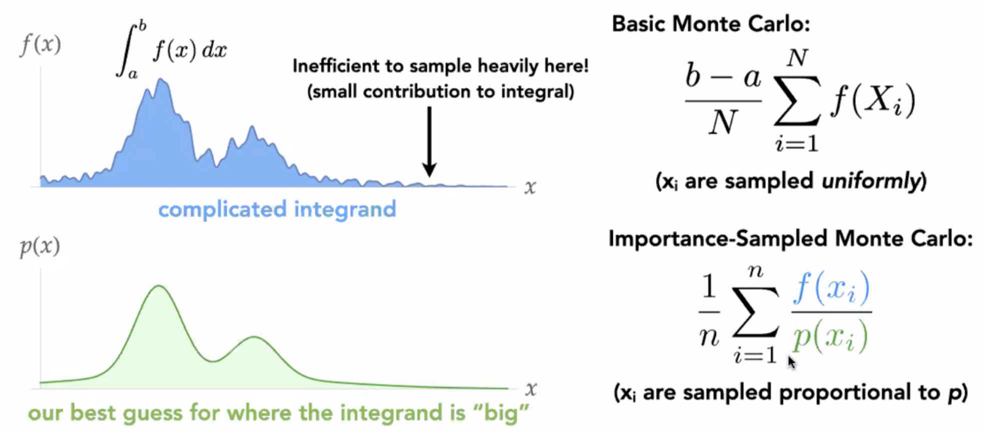
-
- MC estimator uses different random directions at each pixels, only some directions point towards the light
- Important sampling



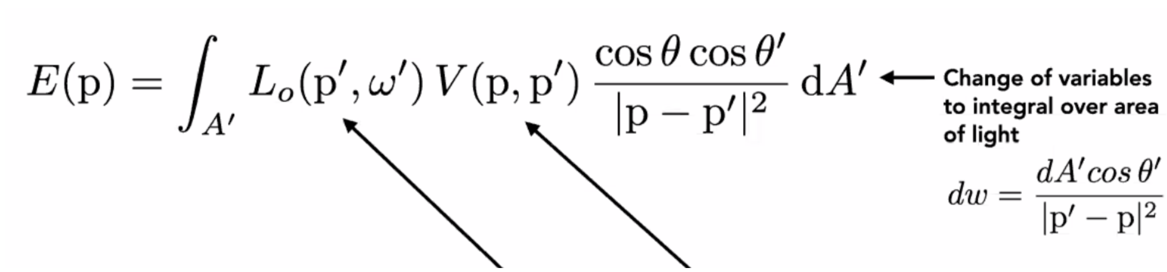
-

Importance Sampling

- Sample the integrand according to how much we expect it to contribute to the integral



Changing Basis of Integration: sampling hemisphere -> sampling light source area



Randomly sample light source area A' (assume uniformly over area)

$$\int_{A'} p(\mathrm{p}')\,\mathrm{d}A' = 1$$

$$p(\mathrm{p}') = \frac{1}{A'}$$

Monte Carlo Estimator

$$F_N = \frac{A'}{N}\sum_{i=1}^{N} Y_i$$

$$Y_i = L_o(\mathrm{p}_i', \omega_i')\,V(\mathrm{p}, \mathrm{p}_i')\,\frac{\cos\theta_i \cos\theta_i'}{|\mathrm{p} - \mathrm{p}_i'|^2}$$

How to draw samples from a desired probability Distirbution: Inversion Method

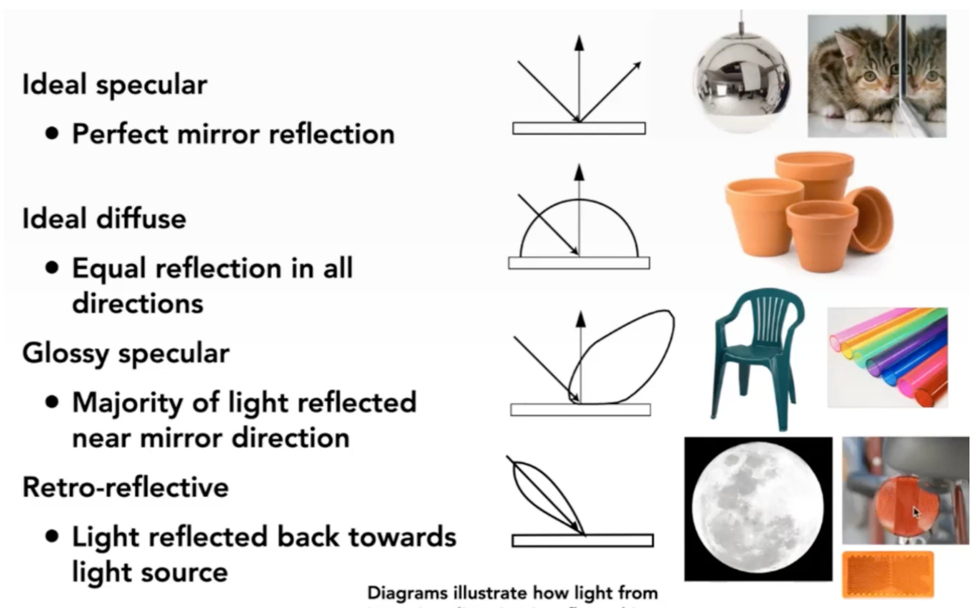
- Task: Draw a RV from a given PDF
- Draw a random value X from this PDF

1. Calculate cumulative pdf
2. Given a uniform RV, choose $X = x_i \; st \; P_{i-1} < \zeta \leq P_i$, calculate inverse
3.

Week 7: Lecture 13 Global Illumination & Path Tracing (3/1)

Material Reflection

- Ideal Specular
    - Perfect mirror reflection
- Ideal Diffuse
    - Equal reflection in all directions
- Glossy Specular
    - Majority of light reflected near mirro direction
- Retro Reflectiove



Reflection at a point

- Light integrated

- Solving the reflection equation

$$L_r(\mathrm{p}, \omega_r) = \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_r)\, L_i(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

- Use monte carlo integration
- Direct lighting, uniform RV

Global Illuminiation: deriving the rendering equation

- Reflected radiance depends on incoming radiance
- Rewrite as transport function:

- **Emitted radiance function (all surface points & outgoing directions)** $L_e(\mathrm{p}, \omega)$
- **Incoming/outgoing reflected radiance (all surface points & in/out directions)** $L_i(\mathrm{p}, \omega),\ L_o(\mathrm{p}, \omega)$
- **Transport function - returns the first scene intersection point along given ray** $tr(\mathrm{p}, \omega)$
- **Reflection operator:**

$$R(g)(\mathrm{p}, \omega_o) \equiv \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, g(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

$$R(L_i) = L_o$$

- **Transport operator:** $T(f)(\mathrm{p}, \omega_o) \equiv f(tr(\mathrm{p}, \omega), -\omega)$

$$T(L_o) = L_i$$

- Natural scenes alot of the light comes from other sources of lights, bouncing lights
- 1 bounce path connecting ray to light, can bounce back to the light source
- Sum over all paths of all lengths

Try Monte Carlo sum over paths

Russian Roulette- Unbiased random termination

- Evaluate original estrimator with probability p reweighted and has the same expected value of the original estimator

Week 7: Lecture 14 Path Tracing (3/3)

Path Tracing Overview

- Terminate paths randomly with Russian Roulette
- Partition the recursive radiance evaluation.
  - Direct lighting, indirect lighting
- Monte Carlo estimate for each partition separately,

Path Tracing Code

- At least one bounce will get the bounce after one bounce

```
AtLeastOneBounceRadiance(p, wo)                    // out at p
   L = OneBounceRadiance(p, wo);                   // direct il

   wi, pdf = p.brdf.sampleDirection(wo);            // Imp. sa
   p' =  intersectScene(p, wi);
   cpdf = continuationProbability(p.brdf, wi, wo);
   if (random01() < cpdf)                          // Russ. Rou
      L += AtLeastOneBounceRadiance(p', -wi)  // Recursive
      * p.brdf(wi, wo) * costheta / pdf / cpdf;// indirec
   return L;

OneBounceRadiance(p, wo)                           // out at p
    return DirectLightingSampleLights(p, wo); // direct
```

Global illumination
- Multiple all bounces of light

Multiple Light sources
- Consider multiple lights in direct lighting estimate
- Loop over all n lights, sum Monte-carlo estimates

Summary Intiutition on Global Illumination and Path Tracing
- Trace N paths through a pixel, sample radiance
- Build paths by recursively tracing to next surface points and choosing a random reflection direction, use reussion roulette to kill probabilities, use improtance sampling to reduce noise

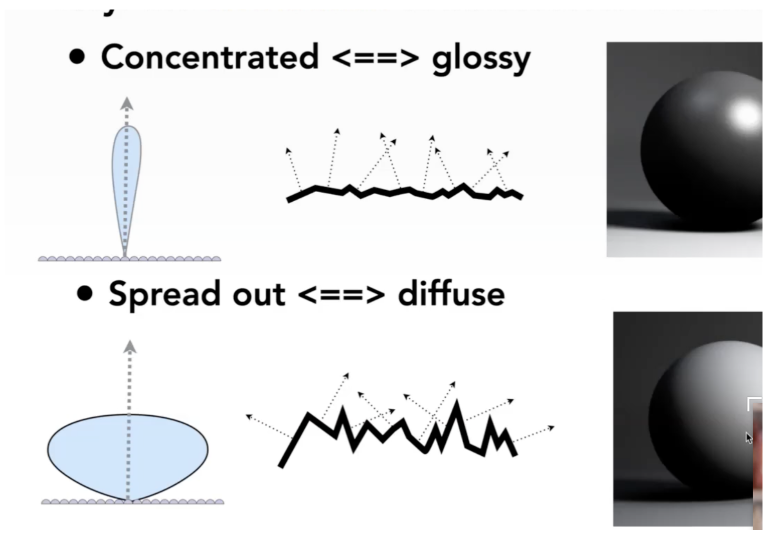
Point lights / Ideal Specular materials

Floating Point Precision Remedies
- Double rather than float
- Ignore reintersection with the last object hit
- Offset origin along ray to ignore
- Preject intersection point to surface

<u>Introduction to Material Modeling</u>

Perfect specular reflection

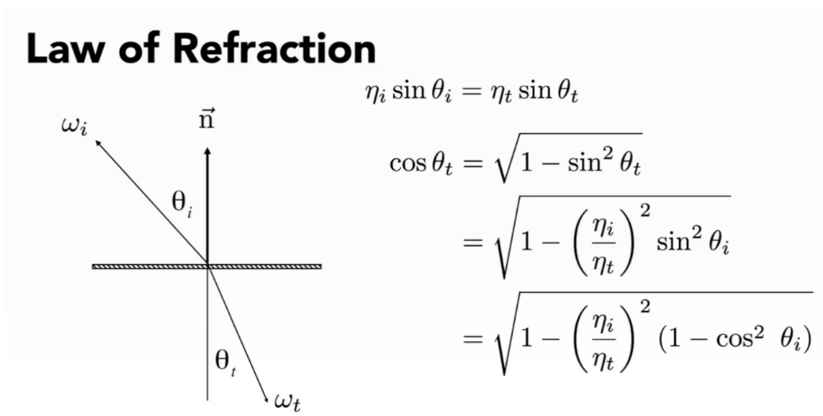
Microfacet Theory
- Macroscale: flat& rough
- Micro scale: bumpy & specular



-
- Individual elements of surfaces look like mirrors
- Fresnel Reflection Term
- Reflectance increases with grazing angle

Microfacet BRDDF

- Distribution of microfacet's normals



Anisotropic BRDFs

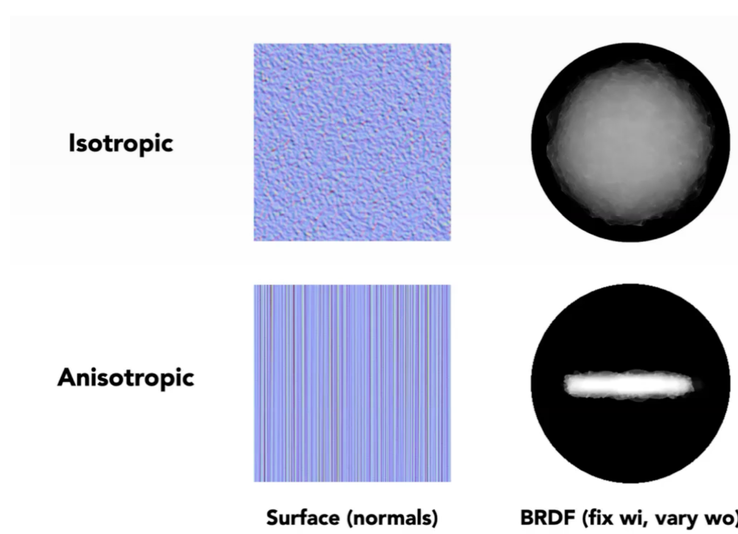
- Point light + Metal = Rough / Elliptical highlight

Isotopic : directionality of underlying surface

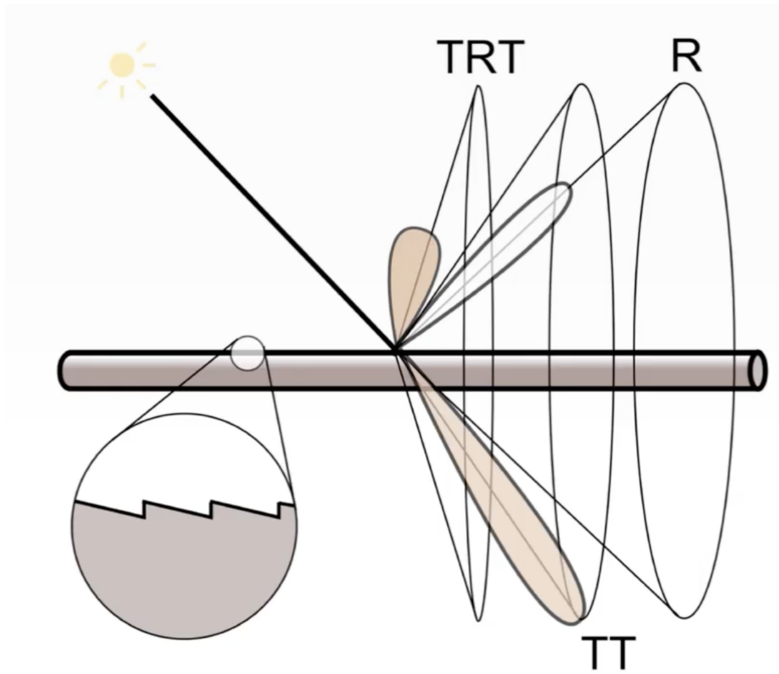
Create glinty with surface normal distribution

Reflection off fibers

Kamiya-Kay Model



- If tiny bounces different when bouncing in a cylinder



- Reflect into multiple cones

Fur appearance

- fur fibers have a more different rendering object, need to have a different model with a bigger medulla
- Fog must be partially absorbed and scattered

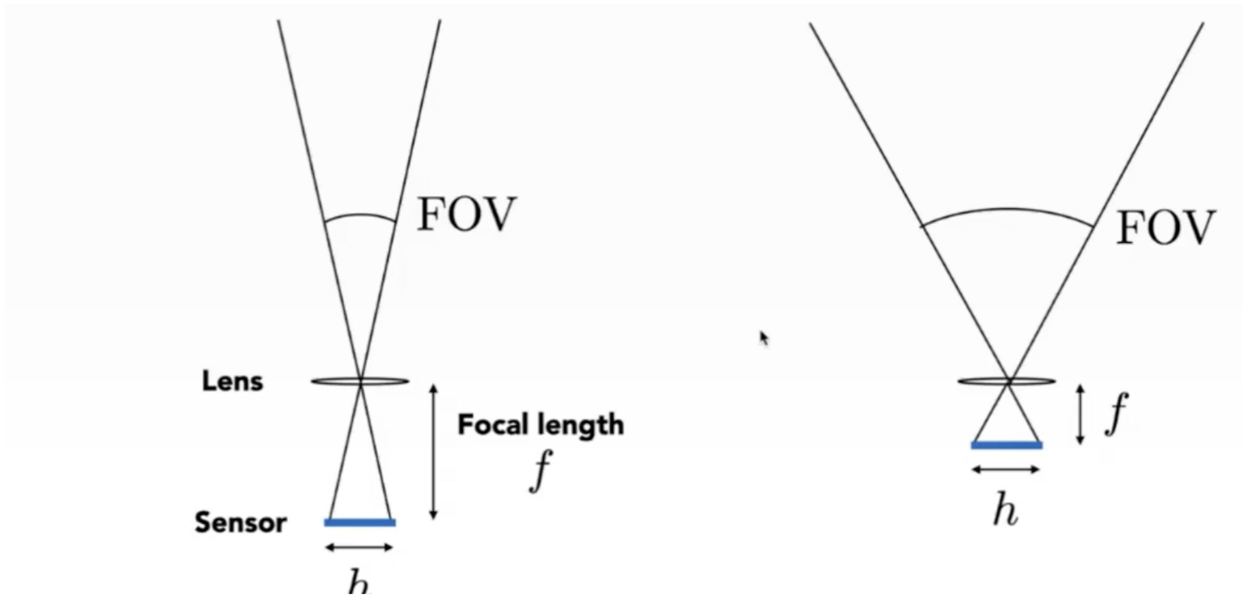
Week 8: Lecture 15/16 Cameras & Lenses (3/8)

Image Capture Overview

- DSLR: opens up lense for sensor

Optics of Image Formation: Field of View

- Effect of Focal Length on FOV



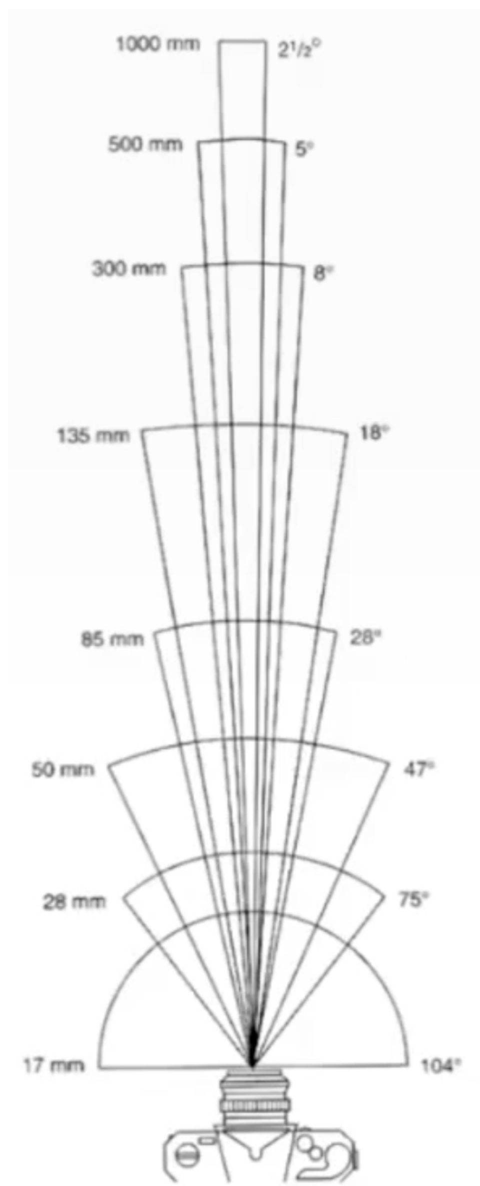
-

Effect of Sensor size of FOV

- full frame sensor, larger picture

Focal Length v Field of View

- Changing focal length on smartphones



- 17mm is wide angle 104 degree
- 50mm is a “normal” lens 47
- 200mm is telephoto lens 12 “degree”

“Choose your perspective before you choose yours lens”

Improve your Own Photography

1. Make sure you have a strong subject, ⅓ of the image
2. Choose a good perspective relationship (relative size) between subject and background
    a. Complement don’t compete with the subject
3. Change the zoom and camera distance to your subject
    a. Actively zoom,and move your camera

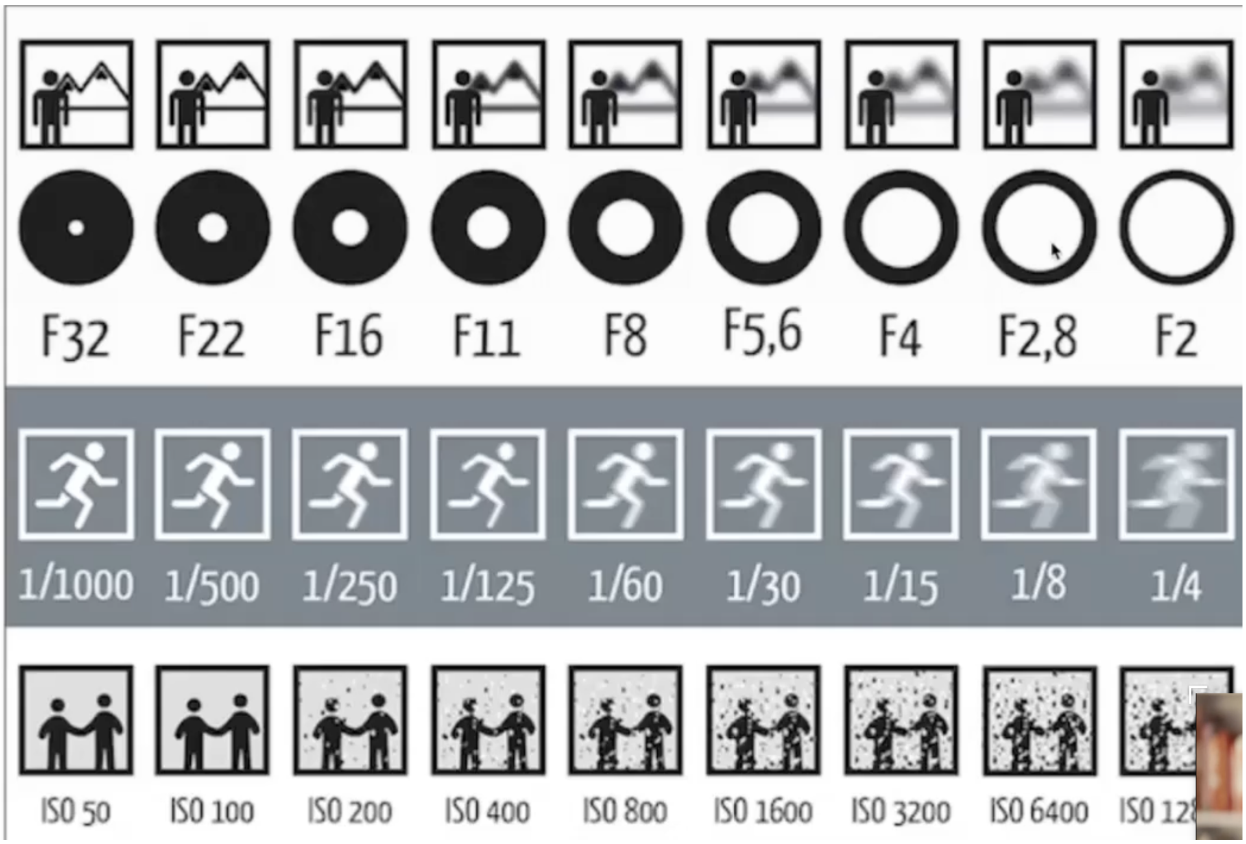
Exposure: Fast and slow photography

- High-Speed photography
    - Objects you can’t see
- Long exposure
    - Leave light on
    - Q = T x E
    - Exposure = time x irradiance
    - Exposure time (T)
        - Controlled by shutter

- Irradiance (E)
    - Power of light falling on a unit area of sensor
    - Controlled by lens aperture and focal length

Exposure levels

- 1 “stop” = 2x exposure
- Bracketing with +/- 1 stop exposure



F-Number of a lens

- defined as the focal length divided by the diameter of the aperture
- Common f-stop on real lenses: 1.4, 2, 2.8, f
- 1 stop doubles exposure

Exposure controls; Aperature, shutter, gain (ISO)

- Aperture size: change the f-stop by opening/ closing the Aperature
- Shutter Speed: Change the duration the sensor pixels integrate light
- ISO: change the amplification between sensor values and digital image value s

Constant exposure: f-stop vs shutter speed

- pairs of aperture and shutter speed give equivalent exposure
- Too bright/dark can adjust one of them

Lower iso has less noise

- more iso has more noise

Electornic Shutter

- Pixel is electronically reset to start exposure
- Fills with photoelectrons as light falls on sensor
- Reading out pixel electronically "ends" exposure
- Most sensors read out pixels sequentially to read entire sensor
- Rolling shutter artifact
    - Mechanical don't suffer from the artifact as much

Lenses

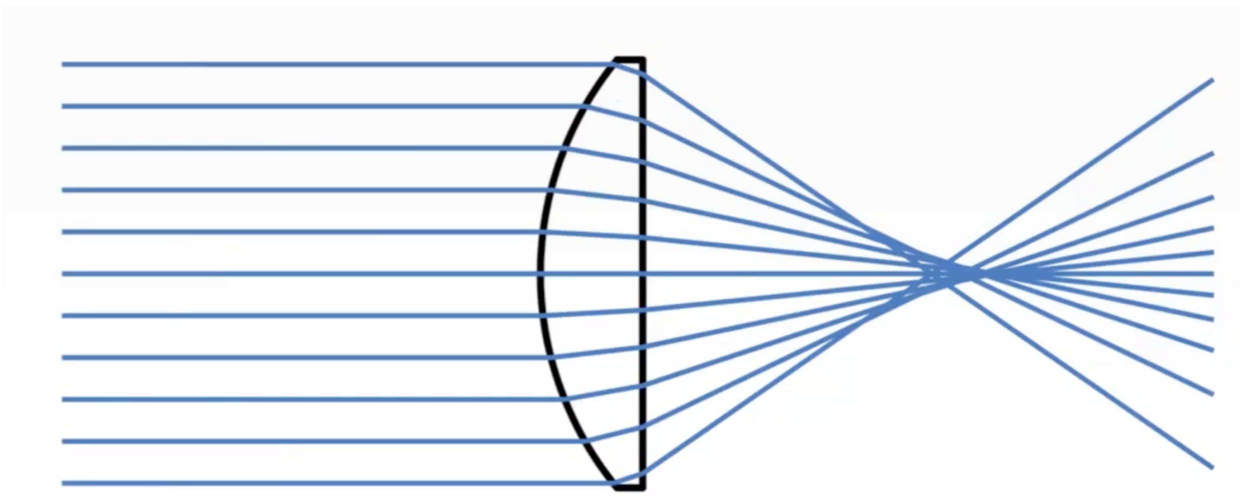
- Thin lens approximation
- Assume all parallel rays entering a lens pass through its focal point

Week 8: Lecture 16 Cameras & Lenses (3/10)

Lenses

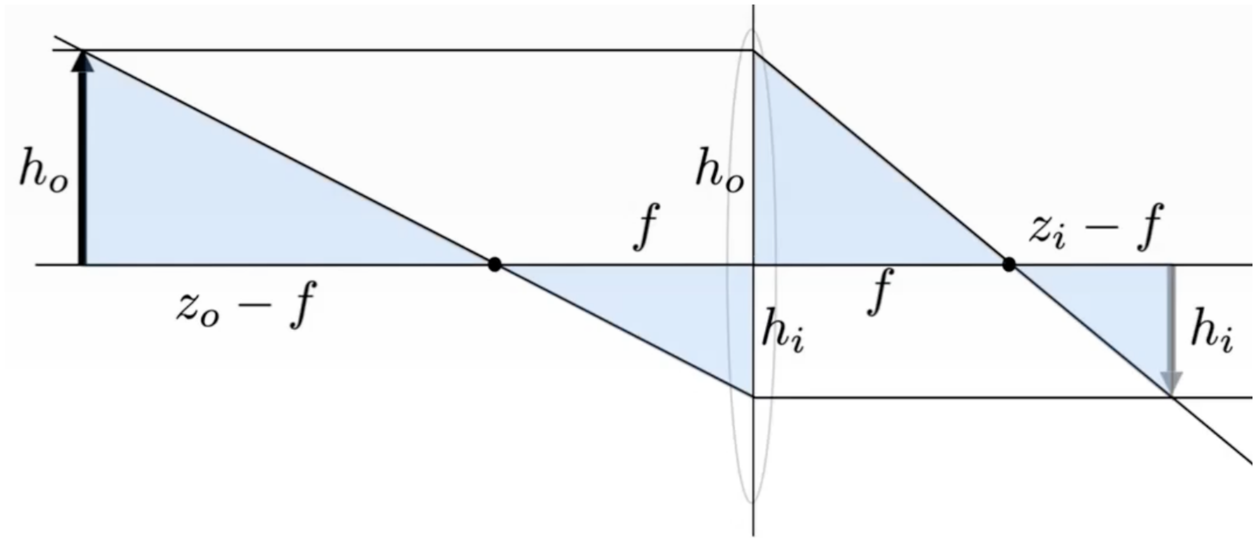
Aberrations

- not ideal convergence

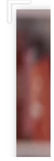


Ideal Thin Lens: parallel rays entering lens pas through its focal point

Gauss' Ray Diagrams



$$\frac{h_o}{z_o - f} = \frac{h_i}{f} \qquad \frac{h_o}{f} = \frac{h_i}{z_i - f}$$



-

$$\frac{z_o - f}{f} = \frac{f}{z_i - f}$$

Object / image heigh
factor out - applies t

$$(z_o - f)(z_i - f) = f^2$$

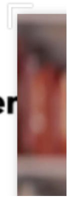
Newtonian Thin Lens

$$z_o z_i - (z_o + z_i)f + f^2 = f^2$$

$$z_o z_i = (z_o + z_i)f$$

$$\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o}$$

Gaussian Thin Ler



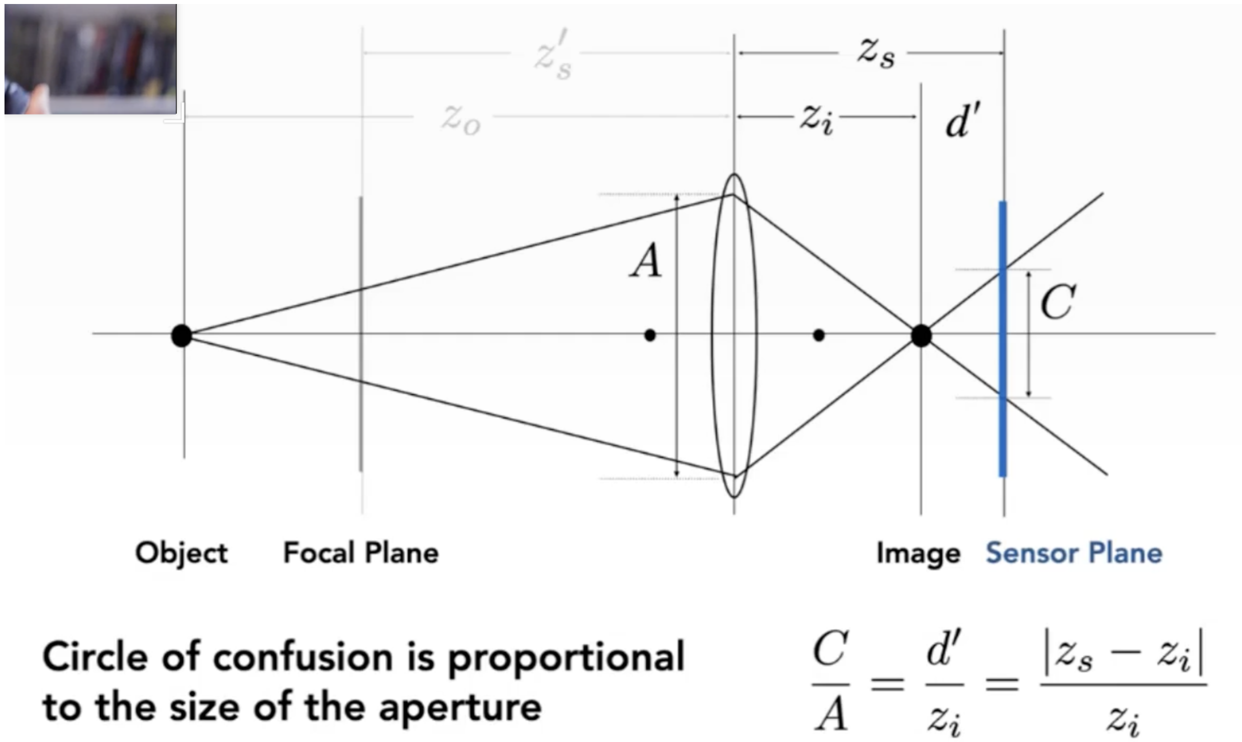
Z_i and Z_o are conjugate points
Magnification example - focus at infinity
Thin Lens

- compressed in depth for low magnification
- 1:1 in 4d for unit
- Stretched for high magnification

Defocus blurr

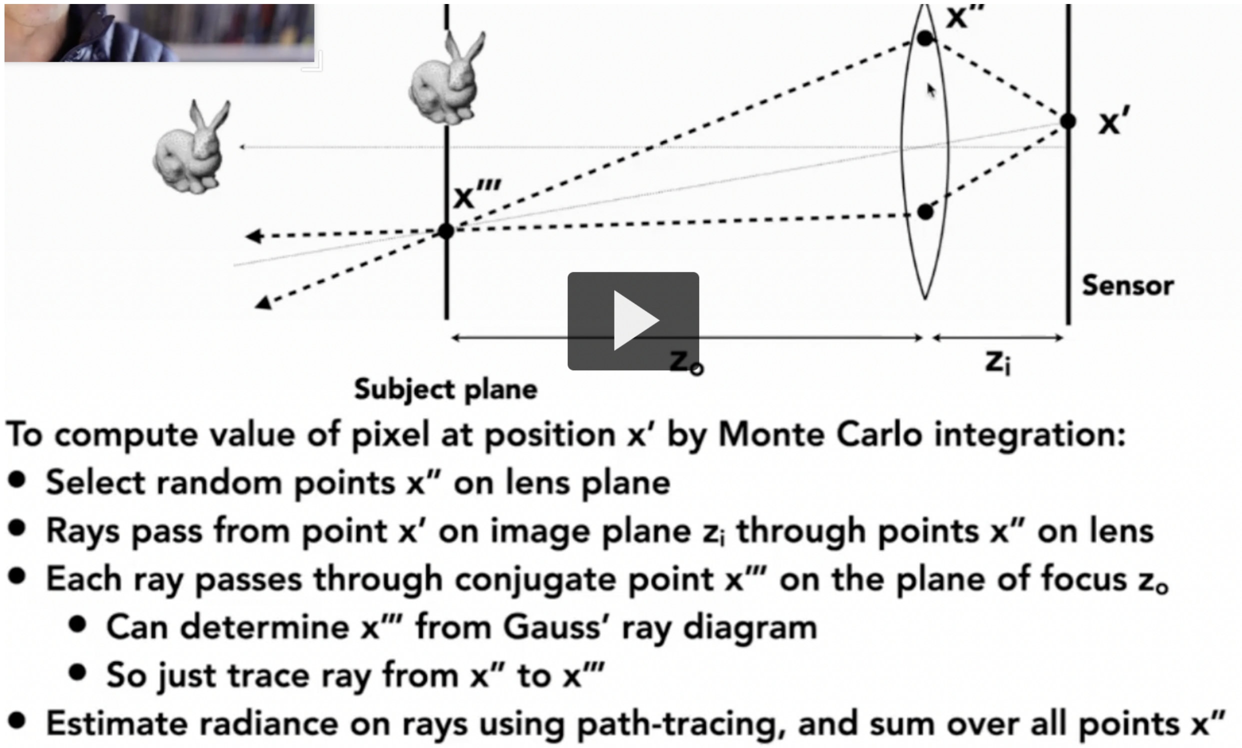
- size of blur kernel depends on depth from focal plane

- Only see the blur kernel itself if you have a point light



Exposure Tradeoffs
- Depth of Field vs Shuttter speed
- Larger blur when larger f stop

Ray Tracing for Defocus Blur (thin Lens)



-

Bokeh

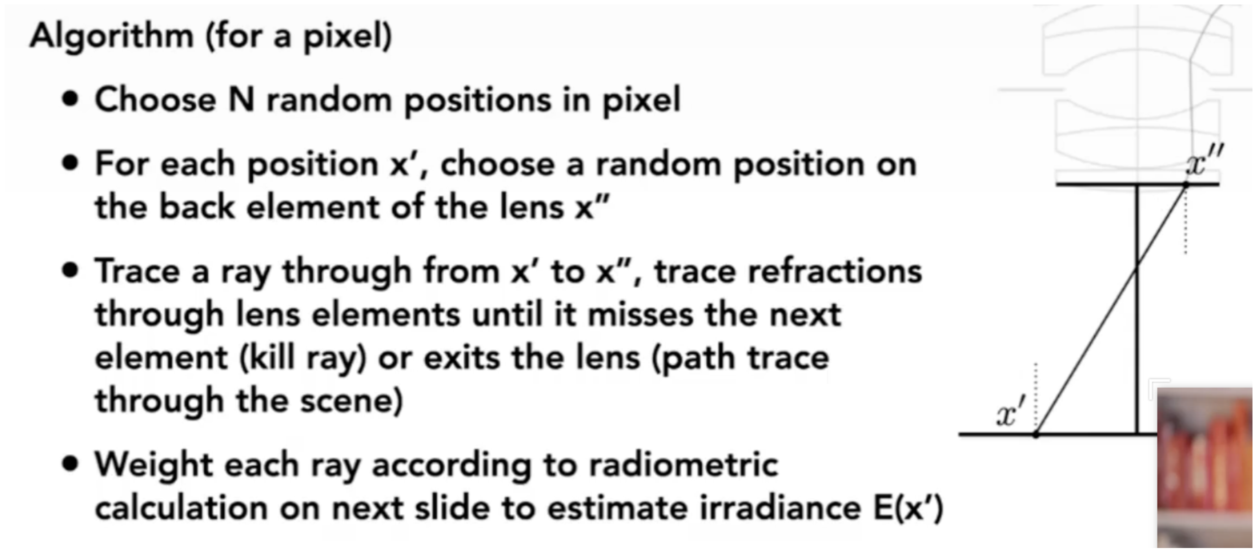
- Shape and quality of the out-of-focus blur

Modern Lens Designs are highly complex

- many pieces of glass
- Snails law
    - Designed to provide optical lens point
    - Ray tracing through real lens designs

Ray Tracing Real Lens Designs

- Monte Carlo approach: compute integral of rays incident on pixel area arriving from all paths through the lens

- 

<u>Week 9: Lecture 17 Intro to Animation (3/15)</u>

Rigging skeleton structure, skimming: putting skin on
Animation
- Bring things to life, aesthetic issues often dominate technical issues
- An extension of modeling
- Output: sequence of images provide a sense of motion

First Film
- Scientific tool rather than entertainment
- Development of animation

Animation Principles
- Squash and Stretch:
    - defining the rigidity and mass of an object by distorting its shape during an action, shape of object changes during movement not its volume
- Anticipation:
    - prepare for each movement for physical realism, direct audience attention
- Staging:
    - Picture is 2D, make situation clear, audience looking in right place, action clear in silhouette
- Follow Through:
    - overlapping motion, motion odesn't stop suddenly, continue at different rates, one motion starts while previous is finishing
- Ease-In and Ease-Out:
    - movement doesn't start & stop abruptly
- Arcs:
    - Move in curves, not in straight lines
- Secondary Action:

        - Motion that results from some other action, needed for interest and realism
    - Timing
        - Rate of acceleration conveys weight, speed and acceleration of charcter's movements convey emotion
    - Exaggeration
        - Helps make actions clear, emphasize story points and emotion, balance with non-exagggerated parts
    - Appeal
        - Attractive to the eye, strong design, avoid symmetries
    - Personality
        - Action of character is result of its thoughts, know purpose & mood before animating each action, no two characters move the same way

Keyframe Animation
- Keyframes done by lead animator, tweens done by computer or other animators

Frame as vector of parameter values

Keyframe Interpolation of each parameter
- need smooth/controllable kinematices

Forward Kinematics
- Articulated skeleton: topology (what connected to what), geometric relations from joints, tree structure
- Join types: Pin (1d rotation), ball (2d rotation), prismatic joint (translation)

Inverse Kinematics
- Given the end effector position, find the joint angles
- Goals: keep end of limb fixed while body moves
    - Position end of limb by direct manipulation

Inverse Kinematics
- multiple solutions separated in configuration space
- Multiple solutions connected in configuration space
- Solutions may not always exist
- Numerical solution

- Choose an initial configuration
- Define an error metric
- Compute gradient of error as a function of configuration
- Apply grad descent

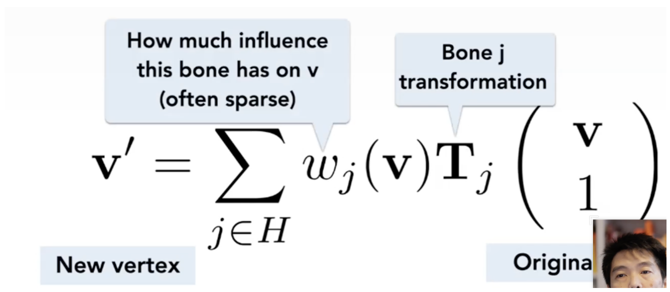
Kinematics Pros and Cons

- Direct control is convenient
- Implementation is straightforward
- May be inconsistent with real life

Skinning

- Move the surface along with assigned bones or “handles”
- Transform each vertex with each bone rigidly
- Blend the results using weights, or assignments

Common Approach: Linear Blend Skinning (LBS)



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

Blend Shapes

- Not all deformation is from bones
- Interpolate surfaces between key shapes
- Create triangular mesh model, linearly blend the faces

<u>Week 9: Lecture 18 Animation, Physical Simulation (3/17)</u>

Rigging

- Augment character with controls to easily change its pose, create facial expressions,

Motion Capture

- Data-Driven approach to creating animation sequences
- Record real-world performances

Optical Motion Capture

- positions by triangulations from multiple cameras, 8+ cameras, 240 Hz, markers on subject
- Capture large amounts of real data quickly
- Realism can be high
- Complex and costly set-ups, not need artistic needs,

<u>Physical Simulation</u>

Newton’s Law: F = ma

Particle Systems

- Single particles are very simple
- Large groups can have interesting effects
    - Gravity, friction, collisions, force fields

Mass and Spring systems

- Mass Spring mesh
- Can be used to model cloths

A simple Spring

- Force pulls points together

$$\boldsymbol{f}_{a\rightarrow b} = k_S \frac{\boldsymbol{b}-\boldsymbol{a}}{||\boldsymbol{b}-\boldsymbol{a}||}(||\boldsymbol{b}-\boldsymbol{a}|| - l)$$

Rest length

- Non-Zero Length Spring
- Dot notation fo r derivatives

Simple Motion Damping

- Behaves like viscous drag on motion, slows down motion in the direction of motion , k_d is a damping coefficient

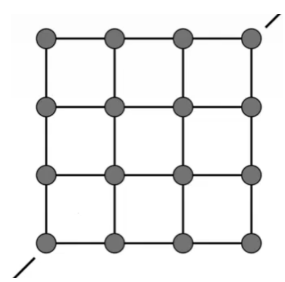
Internal damping for Spring

$$\boldsymbol{f}_a = -k_d \frac{\boldsymbol{b}-\boldsymbol{a}}{||\boldsymbol{b}-\boldsymbol{a}||}(\dot{\boldsymbol{b}} - \dot{\boldsymbol{a}}) \cdot \frac{\boldsymbol{b}-\boldsymbol{a}}{||\boldsymbol{b}-\boldsymbol{a}||}$$

- Damp only the internal, spring-driven motion

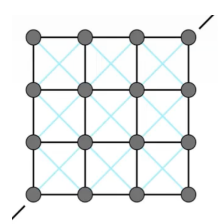
Spring Constants

- Consider the staring = change in length as a frace of original length



Structures from Springs

- Sheet: structure linkages, will not resist shearing
- Will resist shearing but not out of plane



Particle Simulation

- Euler’s Method
    - Simple, commonly used method

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t\, \dot{\boldsymbol{x}}^t$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t\, \ddot{\boldsymbol{x}}^t$$

Errors and Instability

- Solivng by numerical integration with finite differences leads to two problems
- Errors at each time step accumulate,
- Instability: errors can compound, cause the simulation to go to infinity, diverge

Methods to combat instability

- Modified euler
    - Avg velocities at starta nd endpoint
- Adaptive step size, compare one step and two half-steps
- Implicit methods
    - Use the velocity of the next tiem step

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \frac{\Delta t}{2}\left(\dot{\boldsymbol{x}}^t + \dot{\boldsymbol{x}}^{t+\Delta t}\right)$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t\, \ddot{\boldsymbol{x}}^t$$

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t\, \dot{\boldsymbol{x}}^t + \frac{(\Delta t)^2}{2}\, \ddot{\boldsymbol{x}}^t$$

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t\, \dot{\boldsymbol{x}}^{t+\Delta t}$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t\, \ddot{\boldsymbol{x}}^{t+\Delta t}$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \mathsf{V}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}, t + \Delta t)$$
$$\ddot{\boldsymbol{x}}^{t+\Delta t} = \mathsf{A}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}, t + \Delta t)$$

- Position-based/Verlet integration
    - Constrain positions and velocities of particles after time step
    - Idea:
        - After modified Euler forward-step, constrain positions of particles to prevent divergent, unstable behavior, use constrained psoitions to calulate velocity, will dissipate
    - Fast, highly recommended

Algorithm 1 Position-based dynamics

for all vertices i **do**
  initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = 1/m_i$
end for
loop
  for all vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$
  for all vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
  for all vertices i **do** genCollConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
  loop solverIteration **times**
    projectConstraints($C_1, \ldots, C_{M+M_{\text{Coll}}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
  end loop
  for all vertices i **do**
    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
    $\mathbf{x}_i \leftarrow \mathbf{p}_i$
  end for
  velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
end loop

Particle Systems
- Model dynamical systems as collectison of large numbers of particles,
- Each motion is defined by a set of physical forces

Particle System Animations
- For each frame in animation
  - Create new particles
  - Calculate forces on each particle
  - Update each particles position and velocity
- Gravitational Attraction
- Model flocking each bird as a particle
- Attraction to center of neighbors, repulsions from individual neighbors, alignment toward avg trajectory of neighbors

Week 11: Lecture 19 Intro to Color Science (3/29)

Automatic White Balance
- Divide by r, b, g

Color Perception is Highly Adaptive
- Surrounding colors have influence

Perception operates on “opponent” color axes

Color Reproduction Problem We will study
- Goal: At each pixel, choose R, G, B values for display so that the output color matches the appearance of the colors in the real world

What is Color?

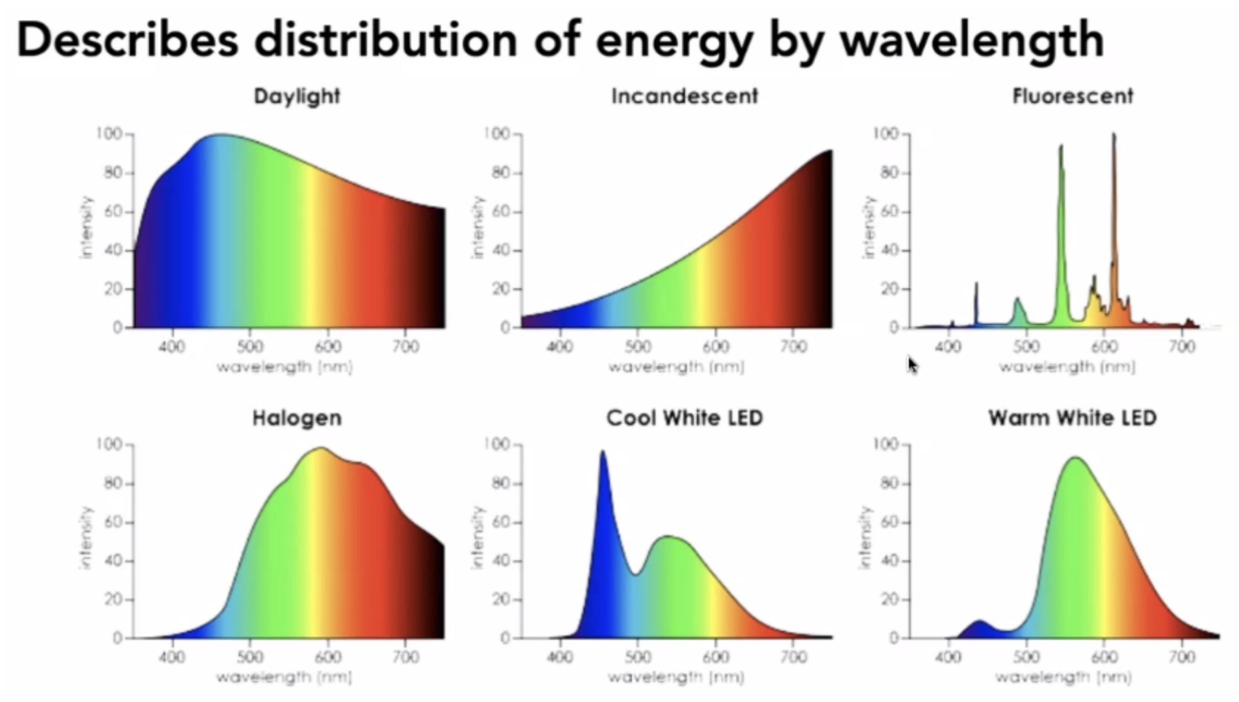
- Color is a human perception, not a universal property of light

Colors are visual sensations that arise

Monochromatic

Spectral Power Distribution (SPD)

- Salient property in measuring light
- Amount of light present at each wavelength
- Radiometric units / nanometer
- Spectral Power distributions vary



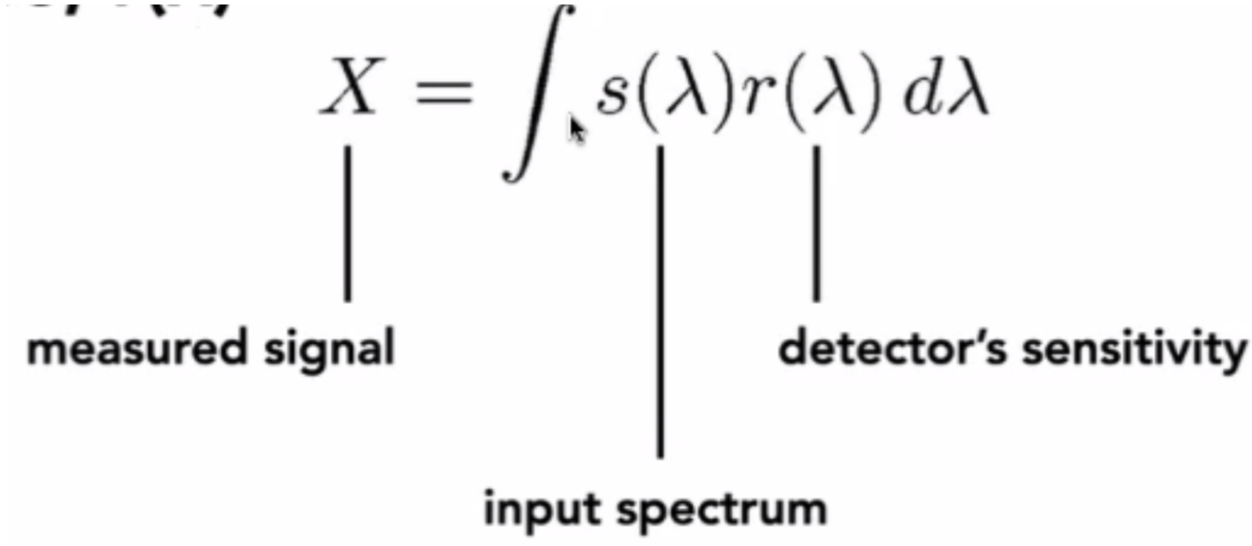
A simple model of a light detector

- produces a scalar value when photons land on it

$$X = \int n(\lambda) p(\lambda)\, d\lambda$$

Mathematics of Light Deteciton

- Light entering the detector has its spectral power distribution
- Detector has its spectral sensitivity or spectral response



Sampled representations rather than continuous functions
Tristimulus Theory of Color

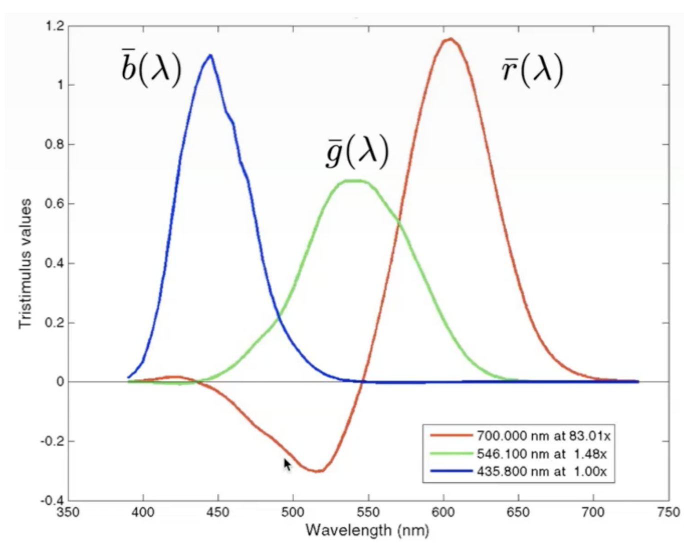
Week 11: Lecture 20 Intro to Color Science II (3/31)

Tristimulus Theory of Color
Experiement
Out of gamut of light using 3 lights

- Can adjust it on the other side
- Linearly of colors

Dimensionality of Human Color Perception

- "Dimension" equals the rank of a basis for the linear space
- For subjects with normal color vision, 3 primary colors are necessary to match any test color
- Primary: 700 nm red, 546 nm green, 435 nm blue
- Color matching set fully characterizes set of all colors



Biological Basis of Color

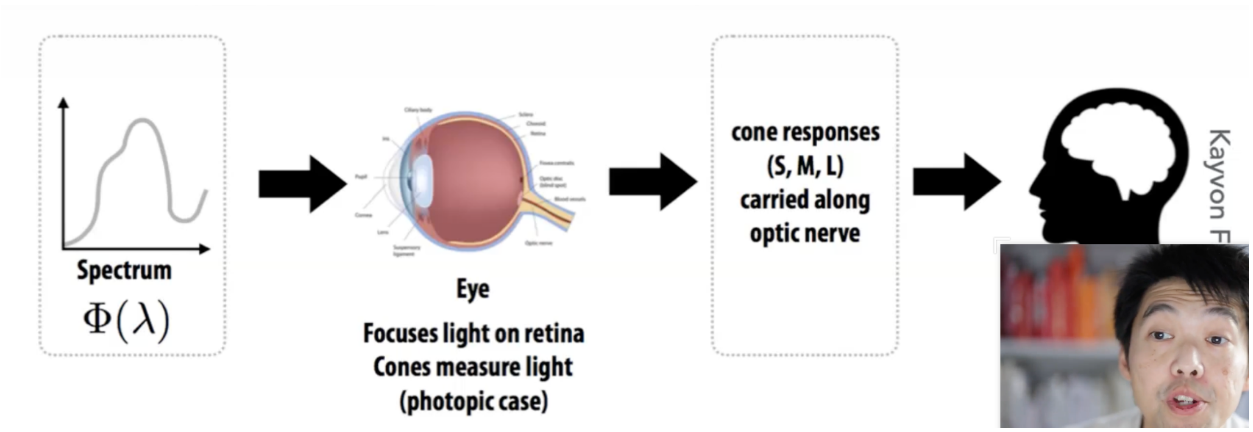
- Rods: 120 million rods in eye only shades of gray
- Cones: photopic: 6-7 million cones in eye
- Different spectral provide sensation of color

Dimensionality Reduction From ∞ to 3

- SPD is function of wavelength

Human Visual System

- Eye measures 3 response values only SML at each position in visual field and this is only spectral info available to brian
- Result of
-



-

Metamerism

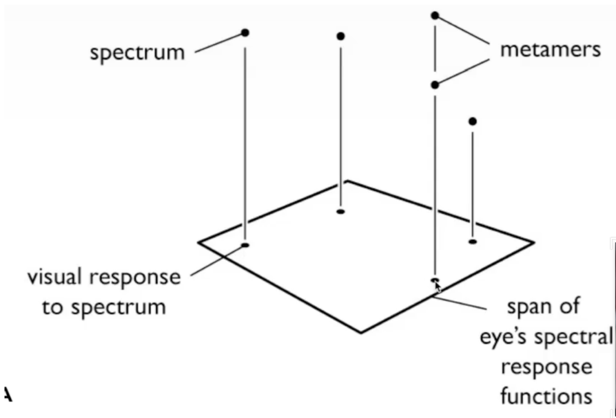
- Metamers: two different spectra that project to the same (SML) 3 dim response
- Will have the same color to a human
- Existence of metamer is critical to color reproduction

Color Reproduction Problem

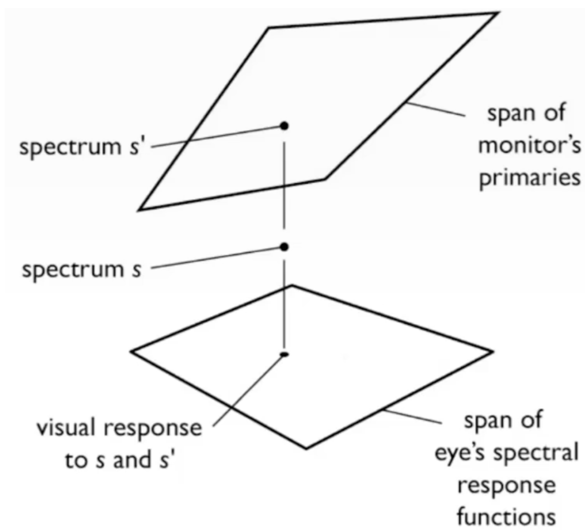
- At each pixel, choose RGB values for display so that the output color matches the appearance of the target color in the real world

Psuedo-Geometric Interpretation

- Projecting a high dimensional vector onto a low-dimesnional subspace
- Difference that are perpendicular to the basis vectors of the low-dimensional space are not detectable



- Display can only produce a low-dimensional subspcace of all possible spectra
- Given spectrum want to choose spectrum st that s' and s project ot the same low-dimensional subspace of the eyes SML response



Color reproduction as linear algebra

- Spectrum projected by display given values RGB
- What color do we perceive when we look at the display
- What the displayed spectrum as a metamer

Color perceived for display spectra with values R,G,B

$$\begin{bmatrix} S \\ M \\ L \end{bmatrix}_{\text{disp}} = \begin{bmatrix} — & r_S & — \\ — & r_M & — \\ — & r_L & — \end{bmatrix} \begin{bmatrix} | & | & | \\ s_R & s_G & s_B \\ | & | & | \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Color perceived for real scene spectra, s

$$\begin{bmatrix} S \\ M \\ L \end{bmatrix}_{\text{real}} = \begin{bmatrix} — & r_S & — \\ — & r_M & — \\ — & r_L & — \end{bmatrix} \begin{bmatrix} | \\ s \\ | \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \left(\begin{bmatrix} — & r_S & — \\ — & r_M & — \\ — & r_L & — \end{bmatrix} \begin{bmatrix} | & | & | \\ s_R & s_G & s_B \\ | & | & | \end{bmatrix} \right)^{-1} \begin{bmatrix} — & r_S & — \\ — & r_M & — \\ — & r_L & — \end{bmatrix} \begin{bmatrix} | \\ s \\ | \end{bmatrix}$$

$$\underset{\textbf{3x1}}{RGB} = \underbrace{(\underset{\textbf{3xN}}{\mathbf{M}_{SML}}\,\underset{\textbf{Nx3}}{\mathbf{M}_{RGB}})^{-1}\underset{\textbf{3xN}}{\mathbf{M}_{SML}}}_{\textbf{3xN}}\,\underset{\textbf{Nx1}}{s}$$

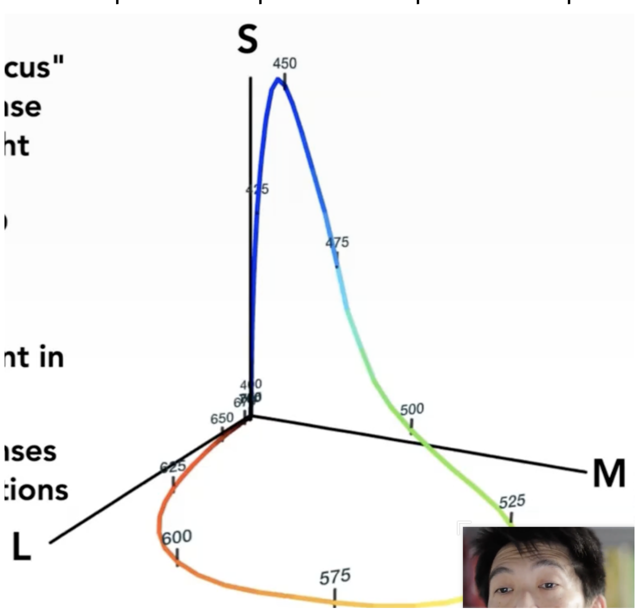
Week 12: Lecture 21 Image Sensors (4/5)

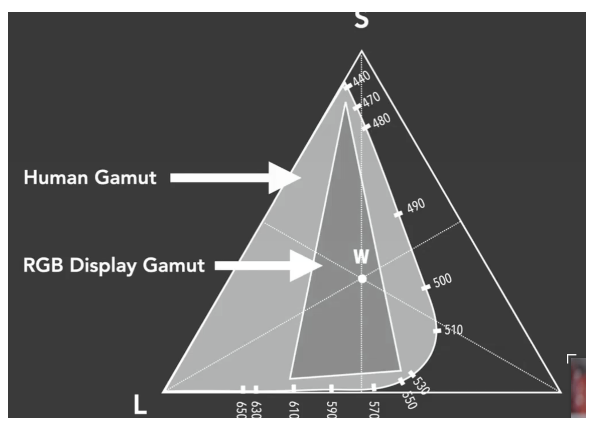
Gamut is a set of colors

- Visualization of “spectrral locus” of human cone cells response to monochromatic light

LMS responses plotted as 3D color space

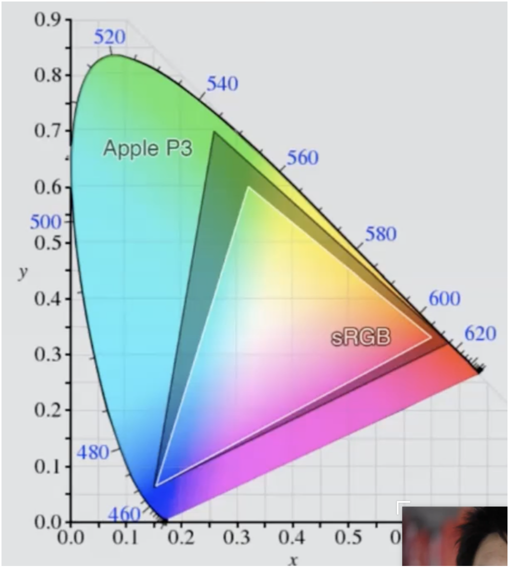
- Plot of SML as a point in 3D space
- Space of all possible responses are possible linear comibiations of points on this curve





Chromaticity Diagram

- 3x3 basis change to x, y LMS to XYZ



Color Spaces

- Need 3 numbers to specify a color
- Color space is an answer to this question
- Define by what RGB scalar values will produce them on monitor
- Device dependent

Standardized RGB (sRGB)

- Makes a particular monitor RGB standard, usable as an interchange space: still widely used today, gamut is limited

Historical “Standard” Color Space: CIE XYZ

- Imaginary set of standard color primaries XYZ
- Designed such that XYZ span all observable colors
- Matching functions are strictly positive

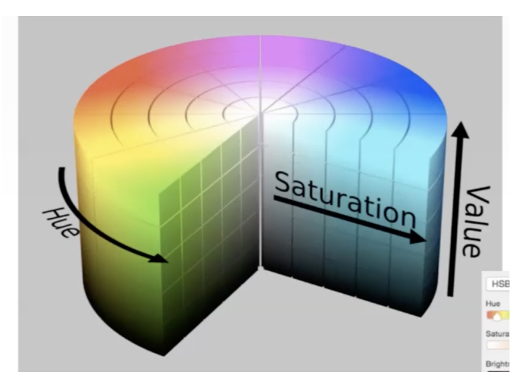
Luminance (Lightniess)

- Y is luminance
- $Y = \int \phi(\lambda)\, V(\lambda)\, d\lambda$
- Chromaticity

CIELAB (aka L*a*b)

- Chromatic adaptation (whitebalance)
- Perceptual uniformity

HSV Color Space (Hue s



Hue: kind of color
Saturation: colorfulness
Lightness: overall amount of light

Additive Color

<u>Image Sensors</u>

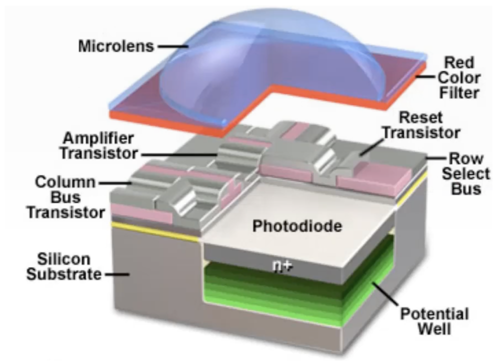
Photoelectric Efffect

- If straight incident photos, it will eject electronis

CMOS APS (Active pixel sensor)

- Like memory layed out in 2d array

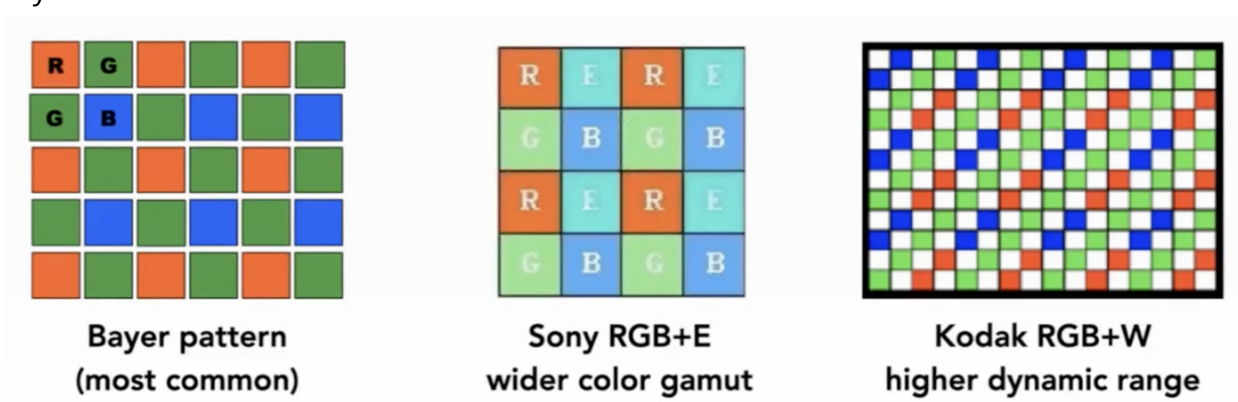
Anatomy of the Active Pixel Sensor



Quantum Efficienty

- Depends on quantum efficiency of the device
- $QE = \frac{\#\,electrons}{\#\,photons}$

Color Architectures

- Color FIlter Arrays (mosaics)
- Bayer Pattern
- 

Demosicking Algorithns

- Interpolate sparse color samples into RGB at every output image pixel
- Simple algorithm: bilienar interpolation
    - Avg 4 nearest neighbors of the same color
- Consumer cameras use more sophisticated techniques
    - Try to avoid

Week 12: Lecture 22 Image Sensors & Digital Image Processing (4/7)

Dynamic Range of the world is great

High Dynamic Range Image (HDR)

- Multiple exposures
- Synthetic Motion blur (8 bit image)

DIY HDR

- Sequence of pictures at different exposures automatically on phones
- Front-Side-illuminated CMOS
    - Photodiodes 50% fill factor, metal 1, metal 2, metal 3, metal 4
- Pixel Fill Factor: light to flow through to photodiode

Most CMOS Sensors are BSI

Pixel Aliasinig Antialiasing

- Insufficient sampling rate

Antialiasing Filter

- Optical low-pass filter
- Layer of birefringent
- Antialisaing reduces details

Imagine

Signal-to-Noise Ratio (SNR)

- $SNR = \frac{mean\ pixel\ value}{standard\ deviation\ of\ pixel\ value} = \frac{\mu}{\sigma}$
- $SNR\ (dB) = 20 \log_{10}(\frac{\mu}{\sigma})$

Photon Shot Noise

- Number of photons arriving during an exposure varies from exposure to exposure and from pixel to pixel, even if the scene is completely uniform
- Number is governed by the Poisson distribution

Poisson Distribution

- Probability that a certain number of random events will occur during an interval of time
    - Known mean rate
    - Independent event s
- $\mu = \lambda$
- $\sigma^2 = \lambda,\ \sigma = \sqrt{\lambda}$
- Error grows slower than the mean
- $SNR = \sqrt{\lambda}$
- Opening aperture by 1 f/stop increases the photons by 2 so SNR increases by $\sqrt{2}$ or +3dB

Pixel Noise: Dark Current

- Electronics disploged increases linearly with exposure time

Pixel Noise: Hot Pixels

- Electrons leaking into well due to manufacturing defects

- Solution #1: Chill the sensor , dark frame subtraction

Pixel Noise: Read Noise

- Thermal noise in readout circuitry

Effect of Downsizing on Image Noise

- SNR increases as sqrt(# of frames) neglecting read noise

Things to remember

- Photoelectric effect

<u>Image Processing</u>

JPEG Compression: The big ideas

- Low-frequency content is predominant in images of the real world
- Human visual system is
    - Less essntitive to detail in chromaticity than in luminance
    - Less sensitive to high frequency sources of error
- Y'CbCr Color space (luma), Cb, Cr chroma channels
    - Compress in Y' channel is much worse than compression in CbCr channels
- 4:2:2 representation
    - Store Y' at full resolution
    - Store cb, cr at half resolution in horizontal
- 4:2:0
    - Store Y' at full resolution
    - Store cb, cr at half resolution in both directions

Transforms into Discrete Cosine transform (DCT)

JPEG Quantization: Prioritize Low Freqeuncies

- Changes by quantization Matrix
- Convert to Y'CbCr, for each color channel do DCT

Theme: Exploit Perception in Visual Computing

- JPEG is example o theme of exploiting characteristics of human perception to build efficient visual computing systems

Week 13: Lecture 23 Light Field Cameras (4/12)

Smarter Blue preserves Crisp Edges

- Denoising

Convolution

$$\underbrace{(f * g)(x)}_{\text{output signal}} = \int_{-\infty}^{\infty} \underbrace{f(y)}_{\text{filter}}\underbrace{g(x - y)}_{\text{input signal}}dy$$

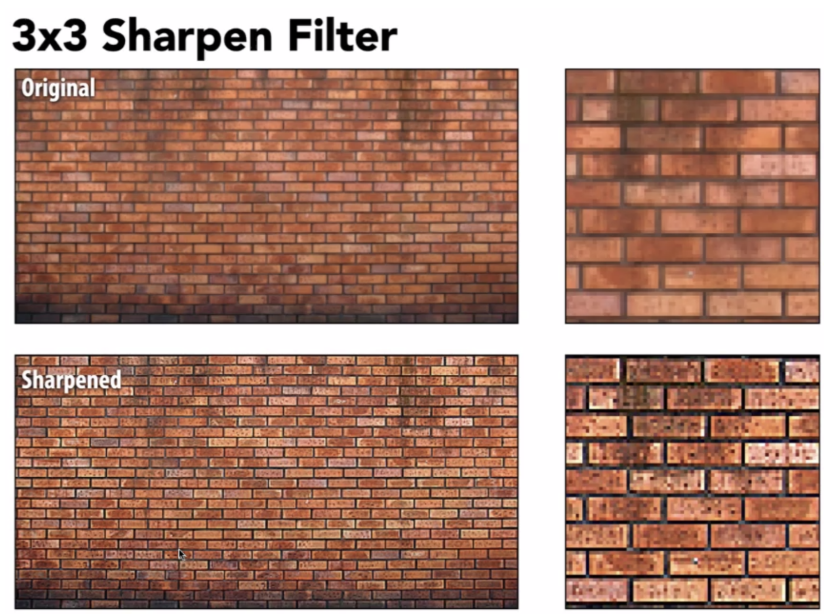
2D Convolution

$$\underbrace{(f * I)(x, y)}_{\text{output image}} = \sum_{i,j=-\infty}^{\infty} \underbrace{f(i, j)}_{\text{filter}}\underbrace{I(x - i, y - j)}_{\text{input image}}$$

Gaussian Blur

- Obtain filter coefficients from sampling 2D Gaussain
- $f(i,\ j)\ =\ \exp()$

Sharpen filter



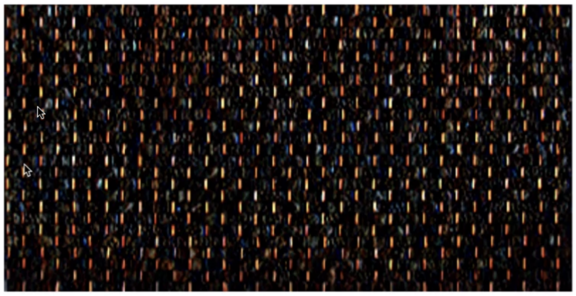
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Gradient Detection

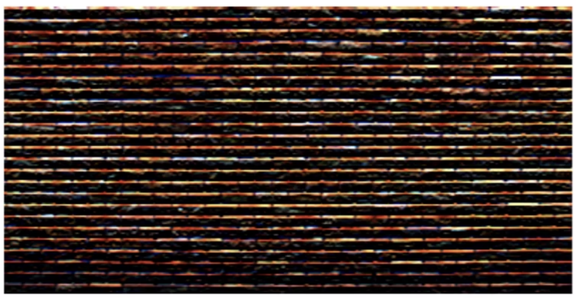
$$\mathrm{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathrm{I}$$

$$\mathrm{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathrm{I}$$

Gradient Detection Filters



Horizontal gradients



Vertical gradients

Note: you can think of a filter as a "detector" of a pattern, and the magnitude of a pixel in the output image as the "response" of the filter to the region surrounding each pixel in the input image (this is a common interpretation in computer vision)

Algorithmic Cost of Convolution-Based Image Processing

- Convolution: N^2 * width * height

Fast 2D Box Blue via two 1d Convoution

- $2N$ * Width * Height

Fast Fourier is $n^2 \log n$ and multiplication is n^2

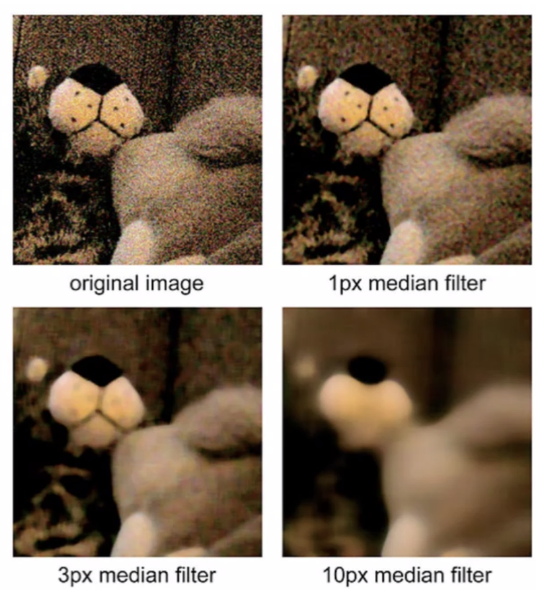
- Otherwise spatial domain is n^4

Efficiency

- When is itfaster to implement by convolution and filter by multiplication in frequency domain

Data-Dependent Filters

- Median Filter
- Bilateral Filter

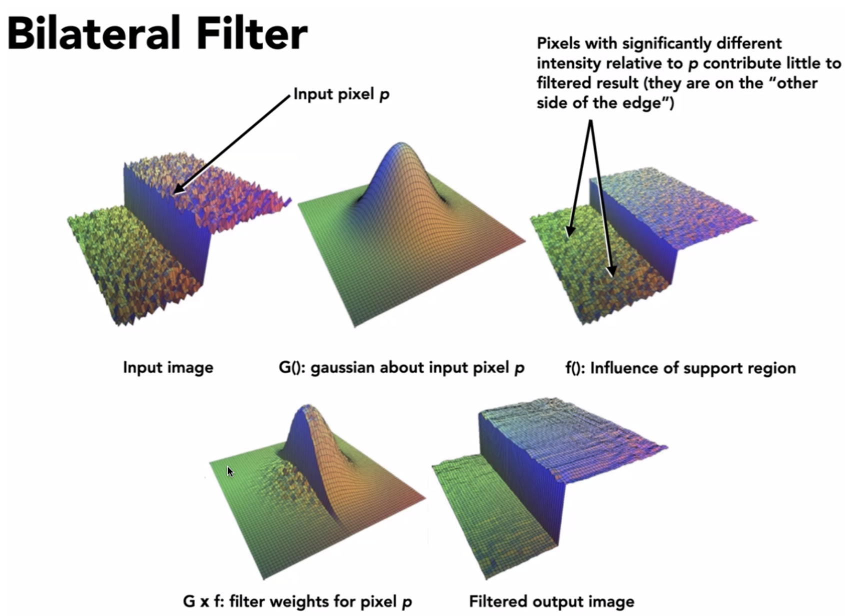


Isotropic Filters vs Anisotopic, data dependent filter

- Distinct flavors in different parts, bilateral fileter separation

Bilateral Filter

$$\mathrm{BF}[I](x,y) = \underbrace{\sum_{i,j}}_{\text{For all pixels in support region of Gaussian kernel}} \underbrace{f(\|I(x-i,y-j) - I(x,y)\|)}_{\text{Re-weight based on difference in input image pixel values}} \overbrace{G(i,j)}^{\text{Gaussian blur kernel}} \overbrace{I(x-i,y-j)}^{\text{Input image}}$$



Data-Driven Image Processing: "Image Manipulation by Example"

- Denoising with Non-Local Means (averaging)
- Instead of average over nearby pixels, search over image that look similar and using those data values to help

Denoising Using non-local means

- Non-parametric Texture Synthesis
- Find a probability distribution function for possible values of p, based on its neighboring pixels
- More texture synthesis examples

Things to remember

- JPEG as an example of exploiting

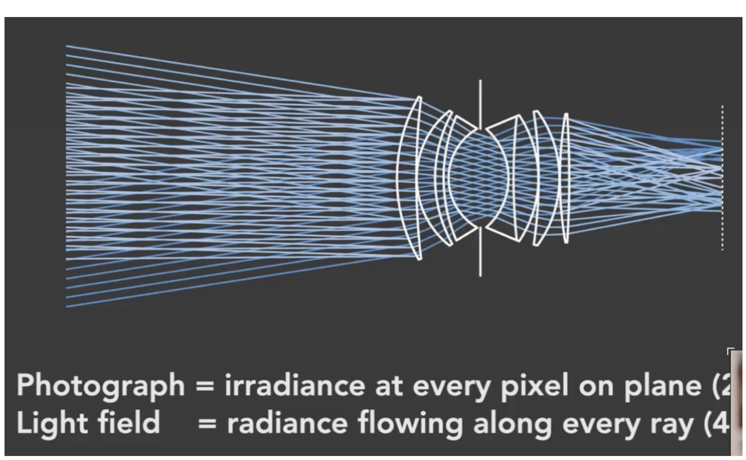
Week 13: Lecture 24 Light Field Cameras (4/14)

3 focus related problems in 2d photography

1. Must focus before taking picture
2. Trade off between depth of field and motion blur
3. Lens designs are complex due to optical aberrations

Capture full range of light fields

4D light fields



How to record light field
- Plenoptic camera

Lens Aberration Example
- Real spherical lens does not converge to a single point
- Computationally redirect rays from physical trajectory to ideal location
- Computationally correct

4D light field: radiance along every ray

Light field camera
- Capture light field flowing into lens in every shot
- Light fiel sensor = microlens array in front of sensor

Computational refocusing
- Refocusing = reproject rays assuming new sensor depth
- Can think of this as shift-and-add of sub aperture images

Computational lens aberration correction with light fields
- Correction = reproject rays assuming no aberration

Week 14: Lecture 25 VR (4/14)

VR: virtual reality
- Completely immersed in virtual world

AR: augmented reality
- Overlay that augmetns normal world

VR Applications
- Gaming, video chat

VR Displays
- Displays attached to head
- Head orientation tracked physically, synchronized to head orientation in low latency

3D visual cues
- Occlusion, perspective, shading focus blue, z buffer
- Stereo in 3d cues

Ocoulus Quest 2
- QC SNapdragon XR2
- 7M total pixels up to 120 Hz display
- Role of eyepiece lenses

1. Create wide field of view
2. Place focal plane at several meters away from eye

Human Visual Field of View
- About 160 per eye and 200 degrees overall

Current VR headset field of view and resolution
- Approx 100 degrees per eye
- 6MP pixel display 24 pixels per degree

VR Display at Human Visual Acuity

- 160 degree
- 8k x 8k display per eye 50 ppd 128 MPixel, current at 24 ppd

Binocular Stereo and Eye Focus

- Passive
- Present each eye with perspective view corresponding to that eye's location relative to the other eye
- Eyes will converge by rotating physically in osckets in order to bring closer and further objects into physical laignment on retina
- At the same depth as optical lenses

Motion Parallax from eye motion

- Environement supported vision based tracking
- External camera find the xyz position and

Quest

- Uses SLAM to estimate 3d structure of the world and position/orientation of camera in the world
- Cameras also track the position and orientation of the controllers 15 infrared LEDs to aid tracking