1 Overview

3i innovation for digital media

- Immersive
- Interactive
- Intelligent

2 Computer Graphics Pipeline

Computer graphics: "the method of using computer technology to convert data into visual representations". 3D graphics uses 3d representation of geometric data stored in computers to perform calculations and render digital images.

3D graphic phases

- Modelling, creating the shape of objects
- Animation, defining movements and positions of objects overtime
- Rendering, generate images/videos

Computer graphics pipeline

- 3d geometry modeling
- Motion syn
- rigging
- textures
- lighting
- rendering

2.1 Geometric modeling

Geometry

- Explicit geometry: triangle meshes, patches, point clouds
- Implicit geometry: parametric surfaces, procedural generation
- Volumetric data: voxel grid by scientific scans
- Generated by: 3d scans, CAD, from images

Triangle meshes are the most common. Neural representation uses a neural network to map multiple 2d images into a 3d scene.

2.2 Motion Synthesis

Generated from

- Motion capture,

- Key frames
- Kinematics/simulation
- Scripting

2.3 Rigging

Allow geometry deformation by motion. Created by

- weighted blending from skeletal motion
- physics
- real world data

2.4 Material/Textures

Material types

- Color (diffuse/spectral)
- Reflectance model (BRDF)
- Geometry details (normal maps)

To map material onto 3d models

- Texture mapping
- Environment mapping

To create materials

- Scanning real world objects (cameras, light stage)
- Artist creation
- Procedural generation

2.5 Lighting

Lighting types

- Point and spot lights
- Directional lights
- Area lights
- Environment lights (radiance map)

2.6 Rendering

Rendering is to produce the final images using computer graphic algorithms.

Inverse rendering is to derive scene information from a rendered image/video, which estimates: cameras, lights, materials, and geometry. This is used in computer vision.

2.7 Composition

Composition is the combining of multiple renders into one. Methods

- Green screen, blue screen
- LED screens

3 Display and Interaction

Pixels

- Images are stored in the pixel array that is displayed on the screen
- A frame buffer is this block of memory dedicated to containing the pixel array.
- A pixel is a basic element of a frame buffer
- Resolution is the dim of the frame buffer
- Color depth is the number of bits per pixel
- Alpha is a component for transparency
- Buffer size computed by mul resolution and color depth.

There are 2d displays (phones, computers), 3d displays (lasers, volumetric), immersive displays (VR, large screen).

Interaction

- Key component for communication and control
- Physical interactions using hardware (physical, 3d sensing, haptic device)
- Logical interfaces are abstract input mechanisms implemented per application (string, locator, pick, choice, dials)

3.1 Applications

Lists of applications

- Movie and films
- Computer games
- Visualization
- Simulation
- XR (VR/AR)
- Training
- Design

4 3D Geometry

4.1 Explicit geometry

Polygon Mesh

- A mesh is a collection of triangles. Each mesh models an object.
- Each triangle is defined by 3 vectors in 3d under a coordinate system.
- Vertices are 3d points. Can include additional data like: color, normal, texture coordinate
- Edges are lines linking two vertices
- Faces are planar elements bounded by edges
- Polygons are planar closed shape made of straight edges
- Surfaces are continuous shape formed by a connected set of polygon faces

Triangles (why triangles)

- Minimum number of points to form a surface
- Any polygon can be subdivided into triangles, including other triangles
- Always defines a unique plane. Four or more points may not lie on the same plane
- Efficient for rasteriation and rendering pipeline
- Easy interpolation using barycentric coordinates

Topology of polygon mesh

- Mesh topology is how the vertices/faces are arranged in a mesh.
- Need to answer queries like: vertices adjacent to a given vertex, faces sharing the same vertex.
- Need to store mesh connectivity information
- Vertex-vertex representation, mapping every vertex to neighboring vertices. Simple, lightweight, does not explicitly store faces. Good for vertice neighboring queries. Not good for edge/face topology queries
- Face-vertex represetation, mapping every face to a list of vertices defining it. (Optionally can have the reverse mapping from vertices to faces). Used in mesh file formats (OBJ). Suitable for wide range of applications, efficient storage

Topology mesh data formats

- OBJ
- FBX
- PLY
- Other prop ones

Cross platform asset sharing

- Incompatiabilities in file formats, materials, animation, and scene structures.

- Solved using open USD file format

OBJ file format (by wavefront tech)

- # is a comment
- `v x y z (w)` for vertex
- `vt u (v) (w)` for vertex texture coord
- `vn x y z` for vertex normal
- `f v1 v2 ...` for faces to vertex index
- For faces, can include vt and vn by `f v1/vt1/vn1 v2/vt2/vn2 ...` where both the vn and vt can be optional
- o for object names
- g for group names
- `mtlib file.mtl` for current material

Points and vectors

- Points (vertices) are positions in space
- Vectors (normals) represents directions
- Relationships between points and vectors are:
- Point sub point is vector
- Point add vector is point
- Vector add vector is vector
- Point add point is undefined

Translation of points and vectors

- Rendering pipeline transforms vertices/normals/texcoords.
- Transformation on points and vectors nets different results: Translating points moves it to a new position. Translating a vector doesn't affect its direction or magnitude
- Homogeneous coordinates allows unified transformation of both points and vectors (later)

Vector operations

- Dot product, $v \cdot w = |v||w| \cos\theta$. Zero if perp
- Cross product, produce vector perp to both vectors. Use det trick. Use the right handed rule to get direction: u at index, v at middle, thumb is cross.
- Normalized vector is a unit vector. To normalize, divide each component by its length

Coordinate system

- Form L with index and thumb, thumb points to $+x$, index points to $+y$.

- Left hand coordinate system use left hand middle finger direction as $+z$. Right hand coordinate system use right hand middle finger direction as $+z$. Unity use left-hand coordinate system

Vertex winding order

- Triangle specified by three vertices in order
- Assume left-handed coordinate system. For right-handed, flip the rotation
- Face with clockwise vertex ordering is front
- Face with counter-clockwise vertex ordering is back
- Normal vector point outwards on the front face
- Culling will stop rendering back faces, which is fine since most objects are closed.

Normals

- Surface normal n at point P is a vector perp to the tangent plane of the surface at P.
- Vertex normal at a vertex v is the normalized sum of the surface normals of all faces that shares the vertex v. Vertex normal is used for smooth shading in Phong.
- The number of vertices and vertex normals may be different due to reuse
- Surface normal of a triangle computed by taking normalized cross product of edges $(B-A)\times(C-A)$ in counter clockwise order if right handed, or clockwise if left handed coordinate system (normal points outwards on front face)
- Vertex normal is computed by normalizing sum of surface normals

$$n = \frac{\sum_i n_{f_i}}{|\sum_i n_{f_i}|}$$

4.2 Implicit geoemtry

Implicit geometry

- Implicit form is an equation that holds for all points (x, y, z) on geometry. Parametric form are equations that traces the points on geometry given free/parametric variables
- For sphere, implicit

  $$x^2 + y^2 + z^2 = r^2$$

  parametric

  $$\begin{aligned} x &= r\sin\phi\cos\theta \\ y &= -r\cos\phi \\ z &= -r\sin\phi\sin\theta \end{aligned}$$

Line and ray

- Line is a line between two 3d points. Parametric form

  $$P(t) = P_1 + t(P_2 - P_1)$$

  for $t \in [0, 1]$

- Ray is a line that starts at a point and extends infinitely in a direction. Parametric form

$$R(t) = P_0 + tv$$

for $t > 0$

Plane

- Given point P_0 on plane and normal vector n. Points P are on the plane if

$$n \cdot (P - P_0) = 0$$

Alternatively the implicit form

$$ax + by + cz = d$$

- For ray-plane intersection, sub ray into plane equation, we get

$$t = -\frac{n \cdot (P_r - P_p)}{n \cdot r}$$

and if $n \cdot r = 0$ then they are parallel

4.3 Volume data

Volume data

- Represents object by sampling its interior at points. Stored uniformally on a 3d grid, where samples do not explicitly represent geometry
- Sampled points are randomly distributed in 3d space. Containing: scalars (temperature/density) or vectors (velocity/force)
- Obtained from scanning devices, simulations, or procedural meshes

Regular grid

- array of points defined by x, y, z
- spacing can be uniform or non-uniform
- visualized as stacked 2d images
- voxel is the smallest unit volume grid, representing sampled value at its position on the grid

Volume rendering

- Treat volume as continuous field and do ray casting or ray marching
- Cast rays through volume. Sample field at discrete intervals d along ray. Use a transfer function to convert voxel scalar values to color and opacity

5 Fundamentals of Raytracing

Rendering

- Convert 3d polygon meshes to an image with color

- Notable algorithms like: raycasting, phong shading, texture mapping, whitted raytracing, distributed raytracing

Rasterization vs Raytracing

- Rasterization: common in interactive graphics accelerated by GPU; independent per object shading; local illumination without considering entire scene; iterate over objects
- Raytracing: common in photo-realistic rendering offline; global illumination using full scene; iterate over pixels using viewing rays; accurate shadows/reflections/refractions

OpenGL rasterization pipeline

- Data stream
- Vertex shader
- Geometry shader
- Rasterization using textures
- Fragment shader
- Frame buffer (optionally output to textures)

Raytracing pipeline

- Ray generation, cast ray from camera for each pixel to determine its value
- Ray traversal, if ray hits, spawn additional rays for shadow/reflection/refraction
- Shading, if object is lit, compute its color based on direct light sources
- Blending, accumulate colors from direct and recursive rays for final pixel color

Ray casting

- Developed in 1968
- Raytracing without the recursive ray traversal
- Still widely used for volume rendering. Also revisited using Neural Radiance Fields

Recursive raytracing

- Developed by Whitted in 1980
- Extends raycasting by introducing recursive rays on: reflection (mirrors), refraction (transparent), shadows (lights)
- Supports global illumination
- Basis for realism

5.1 Ray Generation

Ray generation

- Generate a ray from eye P_0 to point P_1 on an image plane
- Simply formula
$$R(t) = P_0 + (P_1 - P_0)t$$

5.2 Ray Intersection

Ray Intersection

- Need to find the first/closest surface point in scene geometry that intersects a ray.
- Ray-plane intersection: done in last chapter
- Ray-sphere intersection for sphere

  $$|P - P_c|^2 = r^2$$

  is solving for t in

  $$\begin{aligned} at^2 + bt + c = 0 \\ a &= d \cdot d \\ b &= 2(P_0 - P_c) \cdot d \\ c &= |P_0 - P_c|^2 - r^2 \end{aligned}$$

  If there are zero solutions, no intersection. If there is a unique solution, barely intersect. If two solutions, through sphere and take smallest t

- Ray-triangle intersection: two methods

Barycentric coordinates (triangle)

- Any point P inside a triangle can be written as

  $$P = uA + vB + wC, u + v + w = 1, u > 0, v > 0, w > 0$$

  where u, v, w are barycentric coordinates of P on triangle

- Used for interpolation using vertex values and point in triangle tests
- Imagine u as the signed triangle area opposite to A, and v as triangle area opposite to B, etc. Therefore to check intersection, compute u, v, w and if any is negative then P is outside the triangle. (Alternatively, sum the abs coordinates and check if not 1)
- To compute u, v, w

  $$\begin{aligned} u &= \frac{BCP}{ABC} \\ v &= \frac{CAP}{ABC} \\ w &= \frac{ABP}{ABC} \end{aligned}$$

  where the area of the triangle is half the parallelogram area computed using the length of the cross product

Ray-triangle intersection (method one)

- Do a ray plane intersection where plane is

  $$n \cdot (P - A) = 0$$

  using the triangle point A and normal n. Get intersection point P

- Check if P is inside the triangle using barycentric coordinates.

Ray-triangle intersection (method 2)

- Solve a 2d linear system for u, v constraining w
- Not covered, just copy lecture slide formulas lmao

5.3 Ray traverse

Traverse after ray hits object

- Shadow ray: cast towards light source to check occulusion (occlusion is when some object blocks point from light). If blocked, no color. Otherwise, compute direct illumination from light
- Reflection ray: cast in reflection direction using view ray and surface normal at hit point

$$v - 2(v \cdot n) \cdot n$$

- Refraction ray: cast through surface using snell's law. Simulates glass or water. Use

$$\begin{aligned}\eta = \frac{n_i}{n_t} &= \frac{\sin\theta_t}{\sin\theta_i} \\ \omega_t &= \eta\omega_i + n(\eta\cos\theta_i - \sqrt{1 - \eta^2(1 - \cos^2\theta_i)}) \\ \cos\theta_i &= -\omega_i \cdot n\end{aligned}$$

- All ray derivations from lecture slides

5.4 Shading

Shading

- C_a is ambient
- $C_{d,l}$ is the diffuse from light l
- $C_{s,l}$ is the specular from light l
- S_l is a boolean variable dictating shadowness

Phong illumination model

$$C = C_a + \sum_l (C_{d,l} + C_{s,l})$$

Whitted

$$C = C_a + \sum_l S_l(C_{d,l} + C_{s,l}) + k_r C_r + k_t C_t$$

5.5 Blending

Blending

- Accumulates the color information from both direct lighting and indirect lighting into a single pixel color

5.6 Computational Cost

Cost

- Rasterization is $O(N_o N_f)$ where N_o is num objects and N_f is pixels per object
- Raytracing is $O(N_o N_p)$ where N_p is number of pixels to sample
- BVH can accelerate raytracing to $O(N_p \log N_o)$, since rendering time depends on the number of ray-object intersection tests. Tradeoffs: time to build data-structure, dynamic scenes, efficient updates

5.7 Distributed Raytracing

Distributed/stochastic ray tracing

- Developed in 1984
- Trace multiple perturbed rays per pixel
- Anti-alising by jittering rays within pixel to smooth edges
- Soft shadows by sampling shadow rays on area light sources
- Depth of field by custom ray distribution from camera over lens aperture
- Motion blur by sampling rays at different times during shutter

Samples and noise

- Can use efficient sampling methods, monte carlo integration, and denoising algorithms to reduce noise in distributed raytracing
- CNN can be used to denoise low sample pixels

6 Geometry Transformation

Column major form

- Vectors are column matrices
- Matrix elements are indexed by row then column
- Multiplication is done right to left

Transformation usages

- Convenient coordinate systems: object, world, camera, screen
- Hierarhcical modelling for reuse
- Forward and inverse kinematics
- Control projection and rigid body animations

6.1 Two dimensions

Translations of $T(t_x, t_y)$ on p is

$$p' = p + t$$

Rotation of $R(\theta)$ on p is

$$p' = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} p$$

6.2 Three dimensions

Translation is the same as two dimensions. Properties

- Zero identity is zero vector
- Additive using vector addition
- Commutative
- Inverse by negation of vector

Rotation has three matrices for rotation around the x/y/z axis. Each matrix is simply the 2d rotation matrix on the two unused axis and one on the diagonal for the axis of rotation. Properties

- Zero identity is the Identity matrix
- Additive along the same axis
- Commutative along the same axis. Not on different axis
- Inverse is the transpose

OpenGL has methods for translation and rotation.

6.3 Homogeneous Coordinates

Insights

- In Euclidean space, two parallel lines never intersect
- In projective space, two parallel lines can intersect at infinite

Homogeneous coordinates

- Represents an N-dimensional point using $N + 1$ numbers
- (x, y, z, w) in homogeneous space is $(x/w, y/w, z/w)$ in euclidean space
- (x, y, z) in euclidean space has infinitely many representations in homogeneous coordinates by varying w. We typically set $w = 1$
- If $w = 0$, the point is at infinity in the direction of (x, y, z). Therefore it is a vector (x, y, z)
- Scale invariant. Multiplying all by the same non-zero scalar doesn't change the point

Homogenous coordinates pros

- Express perspective projection as matrix

- Combine rigid transformations (translation, rotation, scaling, projection) in a single 4x4 matrix
- Natural way of representing points at infinity (vector)

Translation matrix writes the translation on the 4th column. Rotation matrix is unchanged with one at the 4th dimension.

Point vector distinction

- Point has a non-zero w
- Vector has a zero w
- All point vector operations are simply vector operations
- Vector plus vector is a vector
- Point plus vector is a point
- Point minus point is a vector
- Transformation matrices on points and vectors are identical. Translation matrix on vectors are basically identity.

6.4 Affine Transformation

Transformation types

- Euclidean transformation: preverses shape and size. Translation, rotation, reflection
- Affine transformation: preserves parallelism, collinearity, and proportions (does not preserve lengths or angles). Euclidean transformation and scaling/shearing.
- Perspective transformation is not affine
- All are just 4x4 matrices with constraints.

Affine transformation

- Composition of affine transformations are also affine
- For any two triangles, there exists an affine transformation mapping one to the another. This allows us to transfer affine transformations on one mesh to transformations on another mesh (multiply affine transformation between triangles and applied affine transformation)
- Affine transformation matrix form in homogeneous coordinates is

  $$\begin{bmatrix} p' \\ 1 \end{bmatrix} = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

  or

  $$p' = Ap + t$$

  where A is the affine transformation, and t is the translation.

Shearing

- Sliding an object in a direction parallel to a coordinate axis or plane
- Shearing factor determines the amount of sliding

- Represented by unsymmetric off-diagonal values

Composition

- To combine transformations, multiply them into a single matrix
- Order matters when combining transformations since matrix mul is not commutative. Typical order: rotation, scaling, translation
- Rightmost transformation applied first in matrix form. Lastmost transformation in OpenGL applied first in OpenGL.

Inverse transformation

- Every affine transformation (except zero) is invertible
- The matrix for the inverse transformation is the transformation's matrix A inverse. A is invertible if it is non-singular.

$$p = A^{-1}p' - A^{-1}t$$

6.5 Quaternions

Fixed angle rotations

- 4x4 matrix form has the top left 3x3 matrix as rotation matrix

$$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$$

  However, rotation interpolation not straightforward, and it takes 9 numbers.
- Fixed angle form stores the rotation angles about fixed global axis x/y/z (euler angles pitch yaw roll). The rotations are then composed in a fixed order.

Fixed angle (euler angles) problems

- Gimbal lock. A loss of DOF in 3d rotation when two rotation axes are parallel, eliminating one rotation axis.
- Interpolating euler angles may not produce a unique result (even if both start and end orientation are identical, and interpolation is linear). This is because there are non-unqiue euler angles for the same orientation.

Quaternions

- 4D complex numbers with i, j, k plus a real axis. Defined as

$$(s, v)$$

- Assocative in multiplication but not commutative
- Allows compact representation of quaternions and smooth and unique interpolation between rotation

Quaternion rotations

- The quaternions q to rotate θ around $n = (x, y, z)$ is (where n is normalized)

$$q = \cos(\theta/2) + \sin(\theta/2)(xi + yj + zj)$$

  Then to rotate a vector v by angle θ around n is

$$(\_, v') = q(0, v)q^{-1}$$

- Combining rotations is just quaternion multiplications
- Can convert quaternions to rotation matrix forms (for OpenGL MVP on GPU), and from rotation matrices to quaternions for interpolation. Formula on lecture slides

7 Camera transformations

Coordinate system

- A frame of reference defined by an origin and a set of basis vectors. The basis vectors are orthonormal.
- Any vector in the space is a linear combination of vasis vector. The column form of the vector is simply this linear combination weights.

Coordinate system pipeline

- Object space, world space, camera space, NDC, screen space
- Model transform, camera/view transform, projection transform, viewport transform

Orthogonal matrices

- Columns are orthogonal basis vectors that defines the axes of the frame
- A square matrix is orthogonal iff $MM^T = I$, meaning that columns are orthogonal unit vectors
- Properties: inverse is transpose, closed under multiplication, determinant is plus minus 1, length preserved, angles preserved

Rotation matrix is orthogonal, so its inverse is easy to compute.

7.1 Model Transform

Coordinate system transformation (model transform)

- The world position for point with local coordinates p where the coordinate frame matrix is M is Mp.
- To handle translation, the coordinate frame matrix is the position of the origin point of the new coordinate system
- To handle rotation, the coordinate frame matrix has columns the basis vectors of the coordinate frame's basis vectors
- The inverse of this transformation involves inverting the matrix.

7.2 View Transform

Pinhole camera

- Idealized model of a camera where everything is in sharp focus. Can be replaced with an aperture for realism
- Computer generated look

View transformation

- Convert world coordinates to camera coordinates. Camera view frustum is the area that is visible after projection
- Under basis, the matrix is

$$\begin{aligned} C_w &= (O_w, e) \\ C_c &= (O_c, b) \\ V &= \begin{bmatrix} R(b,e) & O_c - O_w \\ 0 & 1 \end{bmatrix} \end{aligned}$$

- The formulas are mostly on the slides

7.3 Projection Transform

Projection Types

- Orthographic projection. Preserves parallel lines, no distortion
- Perspective projection. Produces foreshortening, mimics human vision

Orthographic projection

- Need to map the z values from near/far plane to $[0, 1]$
- Don't need to touch w
- For raycasting, cast rays in a fixed direction

Perspective projection

- Need to map z values from near/far plane too. Also need to scale down x, y by z
- Uses FOV for the mapping/scaling
- For raycasting cast from point to plane

Panoramic camera

- For raycasting cast from origin to all directions
- Usually lighting maps

8 Hierarchical modelling

Skeleton

- Represents hierarchical structure of parts (rigidbodies)

- Joints connect parts, different joint types allows different movements (with different model transformations)
- Ball-and-socket joints for rotation. Linear actuators for translation

Joints

- Defines the relative positions between two rigidbodies
- Creates a hierarchy of connected rigidbodies to actuator
- DoF is the number of independent motions available.
- DoF of a full body is the sum of the DoF of each joint
- Each joint's rotation/translation are defined accordining to its local coordinate system

Rigidbody model transformation

- Assume rigidbodies are in a skeleton
- For forward kinematics (object to world), its model transform is the product of joint transform matrices.
- If the object already has a world position, we can recompute its position when moving a joint by: undoing the model transformation until the joint, applying new joint transformation, reapplying model transformation after joint

Motion capture data

- ASF file defines the skeleton structure
- AMC file defines the motion

9 Rasterization and Shading Models

Concepts in the rendering pipeline

- Physically based rendering
- Computational photography
- Learning base reconstruction
- 3d display
- Foveated rendering stereo image
- Perceptio

Rasterization vs Ray tracing

- Rasterization projects triangles to the screen and fills the pixels. Ray tracing shoots rays from camera, find interactions
- Rasterization uses z-buffer to determine visibility of a pixel. Ray tracing uses the nearest ray-object intersection
- Rasterization is fast and real time. Ray tracing is computationally heavy
- Rasterization used in games. Ray tracing used for photorealistic renders

Rasterization pipeline

- Data: scene with pure 3d geometry data
- Shader: vertex shader is ran per vertex, often doing MVP transformation to screen space scene data
- Shader: fancy stuff like primitive assembly, tessellation, geometry shader, which overall outputs a list of triangles
- Shader: rasterization converts float triangle vertices to pixels filled
- Shader: fragment shader to color each pixel

The only programmable shaders are the vertex and fragment shader. Both are ran on the GPU.

Triangle rasterization process

- Need to find pixels whose centers are inside the triangle. Note that pixel centers are $(x + 0.5, y + 0.5)$.
- A point is inside the triangle if it is on the same side of all the triangle's edges. This is checked via 2d signed cross products between $(P - P_0) \times (P_1 - P_0)$ and it is inside if all three has the same sign.
- If it is 3d, check if all cross products are pointing in the same direction (against a plane normal using the dot product)
- Aliasing is when there arent enough pixels so there are jagged edges. SSAA (super sampling anti aliasing) improves this by sampling $n \times n$ points per pixel (in a fixed uniform grid), and averaging their fragment shader colors.
- AA produces gradient colored edges that are blurrier

Painters algorithm

- Render objects from back to front, later objects overwrite existing colors
- Can have unresolvable depth ordering

Z-buffer

- Idea: save the current minimum z-value per pixel, and only overwrite if new z-value is smaller
- Algorithm: initialize depth buffer to inf, for each coloring, skip if z-value higher than buffer, overwrite pixel and z-value if z-value is lower than buffer
- Z-buffering (depth buffering) used in conjunction with rasterization. It has the same shape as the pixel buffer.

Phong reflection model

- involves: ambient, diffuse, and specular
- $L = L_a + L_d + L_s$.

Diffuse reflection

- Diffuse reflection assumes that light scatters uniformly towards every direction in the semi-sphere. This means that it is independent of view direction

- Lambert's cosine law states that the proportion of light received is $\cos\theta = n \cdot w$ where θ is angle between object normal n and light ray w.
- Light falloff follows the inverse square law.
- The diffuse reflection contribution for light w is therefore

  $$L_d = K_d \frac{I}{r^2}(n \cdot w) = K_d \frac{I}{r^2} \max(0, n \cdot w)$$

  where I is the light color, and we bound the cosine factor to be positive to fix backface lighting

Specular reflection

- Specular reflection assumes that light reflects mostly in the direction as if the surface is a mirror. Depends on the view direction
- The smaller the angle between the view direction and reflection direction, the brighter the color
- Phong specular contribution is therefore

  $$L_s = K_s \frac{I}{r^2} \max(0, v \cdot w_r)^p$$

  where I is the light color, w_r is the reflected light vector, v is the view vector, and p is the specular/sharpness term. The larger the specular term, the sharper/concentrated the specular reflection
- To compute the reflection vector, use

  $$w_r = 2(w \cdot n)n - w$$

  since $w + w_r$ is in the direction of the normal, but twice the length of w projected onto n.
- Blinn-phong specular reflection uses a simplier approximation to phong. Let h be the half vector (average) between v and w, $h = \frac{v+w}{|v+w|}$, then the specular contribution is

  $$L_s = K_s \frac{I}{r^2} \max(0, v \cdot h)^p$$

Ambient term

- Assumes that there exists some minimum light level
- Ambient contribution is

  $$L_a = K_a I_a$$

10 Shading frequency model

The three shading frequency models are

- Per face (flat shading)
- Per vertex (Gouraud shading)

- Per pixel (Phong shading)

Flat shading

- Shade all pixels on each face once using face normal vector for lighting
- Not good if desires a smooth surface
- Done outside a shader

Gouraud shading

- Shade each vertex of a triangle using its vertex (averaged) normal
- Interpolate using the colors of the three vertices for pixels inside the face
- Not perfect shading
- Done in vertex shader

Phong shading

- For each pixel, interpolate the vertex normals on the three vertices for its normal. Use interpolated normal to compute shading for each pixe
- Not the Blinn-Phong reflection model
- Slowest but most accurate shading model
- Done in fragment shader

Shading frequency model comparison

- As vertex number increases, all three models converge to phong shading
- Use phong at low vertices, and flat/gouraud at high vertices

Shader program

- Operations on a vertex or fragment
- GPU will run all in parallel

Barycentric interpolation

- Interpolation to smooth vertex values across face
- Values include: colors, normals, uv, positions, depth, material attributes
- To compute the barycentric coordinates, use the cross product area formula between all edges and P.
- Barycentric coordinates are not invariant under 3d to 2d non-orthographic projection. It is however invariant under homogeneous coordinates, so use 2d + depth to compute 3d barycentric coordinates post-projection

11 Texture mapping

Texture mapping

- Texture colors/information are displayed by substituting K_d, K_s (or even normals) in the shading program per pixel
- Texture mapping is the mapping of surfaces in 3d space into the 2d texture space. Each pixel in a 3d object space is uniquely mapped to a point on 2d image space (output) and 2d texture space (texture)
- Essentially, each triangle copies a piece of the texture image onto the surface
- Assume that the mapping is known

Texture coordinates

- Every vertex is assigned a 2d (u, v) texture coordinate. $u, v \in [0, 1]$. This is then interpolated per pixel and used to sample texture colors.
- Some textures are tilable, meaning that tiling the image across space will lead to no seams.

Texture mapping algorithm

- For each rasterized pixel (x, y)
- Find texture coordinate (u, v) using 3d barycentric interpolation
- Sample from the texture map using uv
- Set K_d to sampled color

Texture sampling frequency

- The frequency is about how frequent (close) pixels are sampled from the textures. There are three:
- 1 to 1 mapping is ideal, each screen space pixel is mapped to one texture pixel
- magnification, multiple screen pixels mapped to one texture pixel
- minification, each screen pixel mapped to multiple texture pixels
- Requires separate techniques for magnification and minification sampling

Texture sampling, mangification

- Nearest sampling samples the nearest pixel on the texture. Looks pixely
- Bilinear interpolation. Treat texture pixels as texels, for each uv, sample 4 nearest texels weighted by their center distance to uv. Essentially square barycentric interpolation on texels.

Texture sampling, minification

- Creates: Moire, disjoint pixels at distance; aliasing at close. This disconnect is due to farther pixels covering more effective areas on textures, vice versa.
- Supersampling. Costly
- Mipmap. Trilinear interpolation. Too smooth
- Anisotropic filtering. Ripmap. Elliptical weighted average (EWA filtering)

Mipmap

- Notice that
  1. Low texture resolution looks fine at distance. Blurry at near side
  2. High texture resolution looks fine at near side. Aliasing at distance
- Mipmap precompute textures at multiple resolutions/levels into a pyramid, dynamically select the level per pixel.
- Each level has dimensions/resolution $\frac{R}{2^N}$
- To compute N per pixel, find uv coordinates of nearby pixels (one up, one right), compute distances between uv coordinates. The larger the distance, the more minified/zoomed-out the sampling, the lower the resolution (higher N) we should ues, vice versa. Given distance L, use
$$N = \log_2(L)$$
- Rounding N may create obvious switching between texture resolutions. Instead, trilinear interpolation interpolates on each texture (bilinear) as well as between levels.

Ripmap

- Mipmap is restricted to averaging across axis-aligned squared zones for texture sampling
- Ripmap: tile mipmap along diagonal, and stretched resolutions off-diagonal.
- Anisotropic means different behavior in different orientations. Ripmap achieves this by using the stretched off-diagonal resolutions.
- Simple ansiotropic filtering allows averaging in non-square areas in any orientations
- Elliptical weighted average filtering. Averages in an orientated gaussian region around center.

Advanced texture mapping applications

- Environment maps. Texture information represents reflection shading. Captured using a reflective sphere. Used to match lighting of an object to a different environment
- Displacement map. Texture are sampled to offset vertex positions.
- Normal map (bump map). Texture represent normal vectors to simulate displacements
- Baked shading. Textures represent ambient occlusion
- Procedure texture generation
- 3d texture rendering using marching cubes

12 Advanced Ray Tracing

Recall that rasterization cannot handle

- Soft shadows (light sources have area)
- Glossy reflection (blurred reflections)
- Indirect illumination (bouncing lightrays onto indirect surfaces)

Whitted style raytracing

- Shading contains: local blinn-phong reflectance model (ambient, diffuse, specular), recursive reflection, recursive refraction
- No need for z-buffer due to per-pixel shading with ray intersection distance check
- High recursion depth displays more nested reflections/refractions. 2 is the minimum for reflection (surface + reflection). 3 is the minimum for refraction (surface + inside + outside)

Accelerating ray-triangle intersection

- Idea: contain each object/set in a larger bounding box. Ray intersects the object only when it intersects with the bounding box
- Axis-aligned bounding box is defined by 3 pairs of planes on each axis. The enclosed area is the AABB

AABB intersection

- Idea: the ray enters the AABB when it enters all pair of planes; the ray exits the AABB when it exits any pair of planes
- Compute the ray intersection with each pair of planes. Denote them as t_{min}, t_{max}, corresponding to entry and exit points.
- Let $t_{enter} = \max(t_{min})$ and $t_{exit} = \min(t_{max})$ over all 3 plane pairs. Note that all $t > t_{enter}$ could be inside the box, and all $t > t_{exit}$ are outside the box
- If $t_{enter} < t_{exit}$, there must be a section of ray inside the box.
- Assuming previous, if $t_{exit} \geq 0$, the ray must intersect the ray as it either starts inside or outside the box. Otherwise, the ray would starts outside the box.
- Both conditions must sat for intersection

BVH

- A binary tree structure containing triangles
- Nodes are either leafs with faces, or non-leafs with two branches. All nodes have an AABB
- To build, recursively split along the longest axis and build child nodes. Terminate when (leaf) node contains fewer than some set number of faces
- To intersect, check AABB intersection. If non-leaf, check left and right recursively. If leaf, check all faces

13 Path Tracing

Whitted style ray tracing problems

- Reflection/refraction rays only follow mirror/refraction direction
- Only considers direct illumination from light sources
- Violates energy conservation laws (point to eye energy of a point should always be lesser than the total light to point energy)

- Hard shadows

Physically accurate rendering equation

$$L_o(\omega_o) = L_e(\omega_o) + \int_\Omega L_i(\omega_i)(n \cdot \omega_i) f_r(\omega_i, \omega_o) d\omega_i$$

- Light emitted to ω_o is the sum of the emission of the surface and the total illuminated light from all directions
- Total illuminated light computed by integrating over all possible light direction on their colors weighted by intensity (cosine/dot term) and BRDF material properties

Material properties

- Diffuse material tend to reflect light uniformally
- Coated material tends to reflect light along mirror direction
- Translucent material tends to refract light in the refraction direction

Bidirectional Reflectance Distribution function

- Describes how light bounces off the surface. Different for each material
- $f_r(\omega_i, \omega_o)$ is a number specifying the proportion of light reflected along direction ω_o from incoming light from direction ω_i
- A 4D function. Each vector represented by two polar angles (yaw and pitch angle)
- Conservation of energy requires

  $$\int_\Omega f_r(\omega_i, \omega_o) d\omega_o \leq 1$$

  so that the total reflect light energy is less than incident light energy
- Reciproical

  $$f_r(\omega_i, \omega_o) = f_r(\omega_o, \omega_i)$$

BRDF capture

- The phong reflectance model technically constructs a BRDF in combining the diffuse and specular terms
- Can also be measured irl

Path tracing

- Rendering algorithm following the rendering equation
- Produces true photo-realistic images with: glossy reflections, global illumination, soft shadows
- Each sample traces the ray recursively using a randomly sampled reflection direction per surface under the BRDF distribution. Average the pixel color over many samples. This is a monte carlo sampling over the continuous rendering equation

Path tracing properties

- More samples leads to less noisy renders

- Global illumination with higher bounce counts tends to increase overall scene brightness (especially in shadows).

14 Computer Animation

Hand drawn animation

- Skilled animator draws the key poses
- Assistant fills in the inbetween frames, tweens
- Follows the principles of animation

Computer animation

- Skilled animator is the editor, making keyframes
- Assistant animator is the computer

Keyframe animation

- Keyframes are data points that defines the important poses. Each stores the exact desired state of the model at a specified itme
- Timing refers to where in time keyframes are placed. Timing dictates the speed of the motion
- In-betweens are computer generated interpolation. Important distinction between what (angle or endpoints) and how (linear, splines) to interpolate
- Ideal interpolation is smooth with user controls

Parametric curves

- The 3d position of an object under animation is a parametric curve with time parameter u
- Formula is $Q(u)$, where $u \in [0, 1]$

Linear interpolation

- Between p_0 and p_1 is
$$Q(u) = (1 - u)p_0 + up_1$$
- Produces a straightline using weighted average between endpoints.
- For piecewise linear interpolation, linear interpolate each segment, and mod time to get the correct segment and interpolation parameter u.

Spline curves

- A piecewise polynomial curve providing smooth and natural animation.
- Types: interpolating through points, approximating near points
- Order-n spline uses $n + 1$ control points
- Basic cubic spline is
$$Q(u) = a_0 + a_1u + a_2u^2 + a_3u^3$$

with four parameters each determined by four constraints (endpoints and tangents). Sufficient but hard to specify

- Basis function cubic spline is

$$Q(u) = B_0(u)P_0 + B_1(u)P_1 + B_2(u)P_2 + B_3(u)P_3$$

  where $B_i(u)$ are basis functions that determines the behavior of the spline, and P_i are control points.

Cubic bezier spline

- Uses Bernstein polynomials. Two endpoints, two control points
- Equations from lecture slides
- Curve doesn't pass control points
- Curve inside the convex hull of control points
- Intuitive for design and modeling (CAD)

Cubic hermite spline

- Equation from lecture slides. Two endpoints, two tangents
- Curve passes control points
- Tangent can specify the smoothness

Catmull-rom spline

- Special case of cubic hermite
- Tangent automatically calculated by nearby control points.
- Curve passes all control points
- Good for animation and path design
- Each segment is defined by 4 points. Two endpoints and two points immediately before and after to derive tangents

Continuity

- Linear interpolation is C_0 continuous
- Catmull-rom is C_1 continuous

Arc length parameterization

- Equally spaced parameter values do not produce equally spaced points.
- Leads to non-uniform motion by increasing u
- Arc-length is a functiono mapping u to true distance. Reparameterize to a new arclength parameter so uniform speed along the curve

Chord length parameterization

- Exact arc-length too expensive
- Approximate using chord length between uniform samples of u

- Construct a table mapping equal-spaced u to chord length (prefix sum to approximate arc length)
- To find arclength at u', divide u by sampled interval, linearly interpolate chord length between two closest sampled point of u next to u'.
- Similar calculations to invert the arclength table (mapping arclength to u)
- Reparameterize curve to $Q(S^{-1}(s))$ so equal s samples are equal-distance (and constant speed) apart

Rotation interpolation

- Interpolating rotation matrices doesn't work since it is not a valid rotation matrix. Also redundancies in matrix coefficients and non-unique solution
- Interpolating quaternions perform better than interpolating euler angles
- SLERP produces smooth constant angular speed rotation along spherical angles. Formula on lecture slides.
- Chaining SLERP is only C^0, so visible kinks at keyframes
- SQUAD is quaternion splines. Takes two endpoints and two control quaternions. Formula on lecture slide uses SLERP.
- Chaining SQUAD is C^1 continuous (constant angular velocity).

15 Skinning/Rigging

Rigidbody animation

- Objects moving as a whole piece in: translation, rotation, scaling
- No internal deformation.

Non-rigidbody animation

- Mesh deformation
- Physics based

Skeletal animation

- Skeleton is a hierarchical structure containing bones and joints
- Bones have their local transforms
- Mesh is bound to the skeleton. As skeleton/bones move, meshes should follow

Human animation

- Human articulated by skeleton (rigid motion) and soft tissue mesh (non-rigid motion)
- Requires anatonmical interactions between rigid and non-rigid materials

Skinning

- The process of binding a mesh to a skeleton
- Vertex position influenced by one or more bones

- Methods: linear blend skinning, dual quaternion skinning, pose space deformation

Linear blend skinning

- Most commonly used
- Vertex position is weighted average of translation of nearby/influencing bones
- Pros: efficient gpu friendly, well for games and real-time rendering
- Cons: collapsing joints, candy-wrapper twists

Dual quaternion skinning

- Alternative LBS method to produce accurate rotations (no candy-wrapper twist)
- The rotation quaternions are averaged first before application
- Pros: still widely supported, more expensive to compute

Skinning weights

- Each vertex influenced by 1-4 bones
- Either manually created by painting, or automatically by distance
- Weights must sum to 1

Pose space deformation

- Corrective meethods applied after skinning
- Use example true poses to create vertex corrections.
- Interpolate vertex corrections based on current pose and sample pose information
- Use for high-quality characters

Body vs face animation

- Body is driven by skeleton and skinning
- Face uses blendshapes (morph targets) that are predefined facial poses. Allows lip-syncing and showing expressions

Blendshapes

- Are predefined facial poses
- Can blend between sample shapes using weighted average between blendshapes. Or blend using shape parameters (essentially normalized blendshape weights against a base mean shape)
- Blending using shape parameters is preferred
- Formulas on lecture slides

Blendshape pros and cons

- Pros: can handle deformation independent of skeletal movements; intuitive interpolation between expressions; simple and fast

- Cons: requires all shapes to have identical topology; limited to subspace of key shapes; cannot control detailed/specific deformation

16 Input mechanism

Covers the ways that a CG application can take inputs actions from users.

Interactive computer graphics uses

- Video games
- Simulations
- CAD
- Data visualizations
- AR/VR

Input mechanisms

- Keyboard
- Mouse
- Controller
- Eye tracking
- Touch

Keyboards

- Influenced by typewriters
- Comes in many forms and layouts today not just qwerty and ansi layout
- Try to leverage people's mental models (or past experiences) when creating control schemas (ctrl for crouch, space for jump)
- Avoid right side of keyboard if the mouse is used
- Don't assume non-standard setups

Mouse

- Don't expect people to have more than the standard number of mouse buttons
- Use mouse for 3d viewing camera controls
- If a lot of keybinds, allow user to rebind their extra mouse buttons to them. Opt-in instead of opt-out here.

Controller

- Assume standard PS5 and Xbox one controller scheme

Eyetracking

- Uses laser beam, device placed under screen
- Allows hands-free control

- Hard to use/make it well
- Nowdays used for VR

Touch

- Faster operations compared to keyboards
- Easy to use, intuitive and requires less coordinations
- Accessibility benefits, helps users with physical limitations in interaction
- Device size, requires smaller screens

GUI

- The on-screen UI consists of: information display elements, interactable components
- Fitts law. Models how difficult it is for someone to hit a target

  $$ID = \log_2(\frac{2D}{W})$$

  where ID is the index of difficulty, D is distance to target center, W is the width of the target.

  $$MT = a + bID$$

  where MT is the movement time to hit the target, and a, b are empirical constants depending on device.

  $$TP = \frac{ID}{MT}$$

  where TP is the throughput (index of performance), the amount of information that the user can contribute
- Therefore make targets larger and closer to maximize the user contribution

Infinite edges

- For targets on the edge, model it as a target with infinite width/height
- For targets on corners, model it as targets with infinite width and height
- Put the most frequently pressed buttons on edge or corners

17 Human cognition

Covers how humans react to input/information of a CG application.

Flawed humans

- Humans make mistakes, even if the developers think otherwise
- Mistakes due to: inexperience, ambiguity, lack of symbol mapping, confusing symbol mappijng
- Requires special design choices to prevent such mistakes from happening (color coding, hardware changes, symbol placement)

Attention

- User attenion may need to be grabbed at the right moements
- Design primitives: color, font, weight, sound, light
- Cues and highlights: highlight current location and intended future path

Progressive disclosure

- a hierarchy of details that are progressively shown if the user intends to

Color palette

- Don't use eye burning palettes. Use a simple color palette
- Most colors are distractful. Most should be uninteresting
- Eye fatigue: blue and red, blue and yellow, green text on red (vice versa)
- Use blue in large areas not thin lines
- Careful about colorblindness: red and green in center
- Black, white, and yellow in periphery
- Contrast and saturation, choose colors that maximizes them

Shapes and perception

- Use shapes to make objects bigger
- Use edge rendering in place of full shapes

Aesthetics

- Aesthetics are cruical to how players perceive the game
- Understanding human perception helps to design games that feel right

Mental models and memory

- Use real life symbols to denote actions similar to real life actions
- Short term memory: rapid access but limited capacity
- Long term memory: slow access but huge capacity
- Sensory memory, buffer for stimuli received through senses, continuously overwritten

Learning curve

- Low number of features is boring
- Ok number of features is fun
- A lot of features makes it complicated
- Good game design allows the user to learn by doing. Avoid requiring a manual or FAQ page.
- Change your game to match the players

18 Evaluation Techniques

Human-centered design

- Iterate repeatly with your users
- The user should be the focus. Modify your app to fit the users
- Typically done multiple times to check perception.

User experience evaluation

- User Testing: who is the end user, how they will use my app
- Usability Testing: can the end user use my app
- Basically: how users are using the app, can users use the app expectedly

18.1 Observational method

Observational method

- Watch users and see how they do in game. See where things breakdown
- Involves: think aloud, cooperative evaluation, post-task walkthrough
- Ideally perform all methods.

Think aloud

- Participant plays the game. When they play, ask them to vocalize their thinking
- Take notes on their behavior/thoughts. No interfering
- What are they doing? Why? What they will do next? What do you expect to happen?
- Pros: simple and insightful
- Cons: subjective, awkard, might change how they play

Cooperative evaluation

- Variation of think aloud, where user can interact with the experiment by asking questions. Experimenter can also ask questions
- Pros: much more relaxed, break up awkward silence, user won't get stuck, encourage to critize software
- Cons: overlap and hide problems, less rigorous

Post-task walkthrough

- Talk to players afterwards after walking them play
- Ask users about specific things they have done
- To help with memory: show recording of gameplay, make task small
- Pros: avoid interruptions, make playing more natural
- Cons: they might forget stuff, post-hoc interpretation may be different to in-the-moment

Capturing observations. Can have multiple methods done by multiple people.

- Pen-paper
- Audio
- Video
- Computer Logging

18.2 Querying technique

Querying technique

- Specifically ask the users about their experience
- Involves: interviews, questionaire

Interviews

- Question users one-on-one based on prepared questions. After they've played the game.
- No specific focus on one stretch of gameplay compared to post-task walkthrough, Focuses on the general game experience
- Pros: cheap, varying detail, elicit user views
- Cons: time consuming to analyse, hard to balance depth and breadth

Interview questions

- Question types: closed questions with predetermined answer format, open questions with no format
- Closed questions are easier to analyse but lack details. Should avoid
- Avoid: long questions, leading questions, compound questions, jargon

Interview styles

- Structured: verbal questionnaire, lacks depth
- Semi-structured: more difficult to do, requires background knowledge, can deviate at interesting answers
- Open-ended: having only a starting question, requires no knowledge, hardest to administer and analyse

Questionnaires

- Fixed, written questions given to users
- Pros: easy to administer and analyse, easy to compare
- Cons: lacks depth

Custom questionnaire

- Design the questions to investigate specific topics
- Questions: open-ended, scalar, multichoice, ranked

Standard scales

- Created by researchers to establish norms

- Plenty each targeting different aspects. Not tailored to the specific evaluation needs, but can make it easy to compare results between people.
- Allows computing a standardized score that can be used under a threshold criteria
- Types: post-study questionnaries, questionnaires for website, experimental comparison of post-study questionnaire, etc (lecture slides)

Questionnaire process

- Decide what to measure
- Find standardized questionnaire and create your own
- Deploy game and collect data
- Analyze data

18.3 Physiological measures

Eye tracking

- What the player is looking at, at a particular scene
- Gives insight into cognitive process
- Types of eye movements: fixation (focusing), saccades (rapid movement), smooth pursuit (following)

Other measurements

- Heart rate
- Blood pressure
- Muscle activity
- Brain activity
- Sweat glands
- Hard to interpret, but provide some insight into the emotional state of user

18.4 Receiving feedback

Expected to use user evaluation results to improve your game. Improvements must be explained in your report.

Your audience is good at recognizing problems but bad at solving them.

Cursed problems are unsolvable design problems, rooted in a conflict between player promises (what each player think the game should be).

- Focus on mastery vs focus on winning
- Safety vs addiction
- May only be discovered using user evaluation

19 VR

Reality virtuality (RV) continuum

- A spectum from: real environment, augmented reality, augmented virtuality, virtual environment
- Real environment is real life. Virtual environment are all abstract. The inbetweens are a mixture between reality and virtuality
- VR, near the virtual environment
- MR, the middle
- XR, all except real life

Virtual reality

- Fully immersive and interactive CG environment
- Immersion is the perception of being physically present in a non-phyiscal world
- Presence is the sense of being there

VR history

- Sensorama
- Sword of damocles
- Army prototypes: super cockpit, vecta
- Cave systems
- Nintendo virtualboy
- Oculus rift
- Facebook acquires oculus
- Google cardboard
- Tethered headsets

VR applications

- Gaming
- Training and Education
- Design and Architecture
- Social VR

Steoreoscopy

- A method to make 3d images for VR
- Virtually, two cameras offset by the interpupillary distance rendering the scene
- IRL, two eyes seeing two different (slightly shifted) projections from the virtual cameras

Optics

- Lens are needed to simulate distance with a closeup display
- Different people have different eyes/distance, so adjustments mechanisms are required

Head and controller tracking

- IMU to track rotation
- Cameras to detect controller movements and gesture detection

Body tracking

- External sensors: kinect
- Inside-out tracking: meta-quest
- Market-based tracking
- Wearables
- Ai based tracking

19.1 VR technical problems

VR open problems

- Haptics: for real immersion, touching things is needed. Hard to provide haptics, common methods are: passive haptics, active haptics, mid-air haptics, haptics retargeting. Always a tradeoff between accuracy and diversity
- Space limitation: cannot navigate large virtual worlds while confined to a small physical space. Locomotion techniques: continuous input (joystick), teleportation, walking in place, treadmill, redirected walking

Real world awareness

- VR makes people lose real world awareness, can damage IRL things

Motion sickness

- Cybersickness is a known problem
- Some people are more prone to it than others
- Getting better with improvements in latency and precision

Gorilla arm effect

- Tiring to holdup arms all the time; full body gestures can cause fatigue and shoulder pain
- Consumed endurance model suggests fatigue after just 90 seconds
- Gun slinging proposed as a solution, but removes hand-display feedback which is bad

19.2 VR social problems

Social problems

- Social acceptability
- Black mirror effect (VR making people anti-social)

- Planetary boundaries (climate effects)

20 Mixed/Augmented Reality

MR

- An environment that presents the real world and virtual world objects together
- AR combines real and virtual and is interactive in real time

MR technologies

- Headset
- Projection mapping
- Mobile AR

AR 2.0

- Mobile based vision, with virtual objects correctly situated in the real world
- People create and share geolocated information with each other

Optical see through

- User see the real world directly through transparent lenses
- Digital content projected onto lenses
- Waveguides used to merge virtual content with irl content

Video see through

- Real world view captured by cameras on the headset and displayed on screens
- Virtual content painted into video feed

MR requirements

- Occulusion (layering), spatial awareness (shapes), physics
- SLAM algorithm processes data from IMUs (camera feed mostly), and repeatedly: build environment map, locate device in environment.

SLAM

- Since objects in space have a spatial relationship with each other, sensor data can get a good probability dist on where positions are. Many algorithms can be used to calculate the probability distributions
- Camera feed can identify unique space features (feature extraction), which are stored as descriptors along with their neighborhood information.
- For SLAM, the most data, the higher the precision. But some things are hard to distinguish (white walls, transparent objects, reflective surface)
- Once enough feature points are collected, can connect the dots and create a mesh for the real world

Implementation

- Sources on lecture slide

20.1 AR technical problems

Geo-referencing

- SLAM generates the local environment knowledge, but cannot share that underestanding
- Azure spatial anchors may solve it
- Marker based AR mitgate this easily, but less flexible

Compositing

- Compositing is about bringing the visual elements together so that virutal objects looks belonging in irl
- Includes occulusion, lighting, shading, and coloring/texturing

Plus all the technical problems from VR

20.2 AR societal problems

Same as VR

21 Speaker Lecture