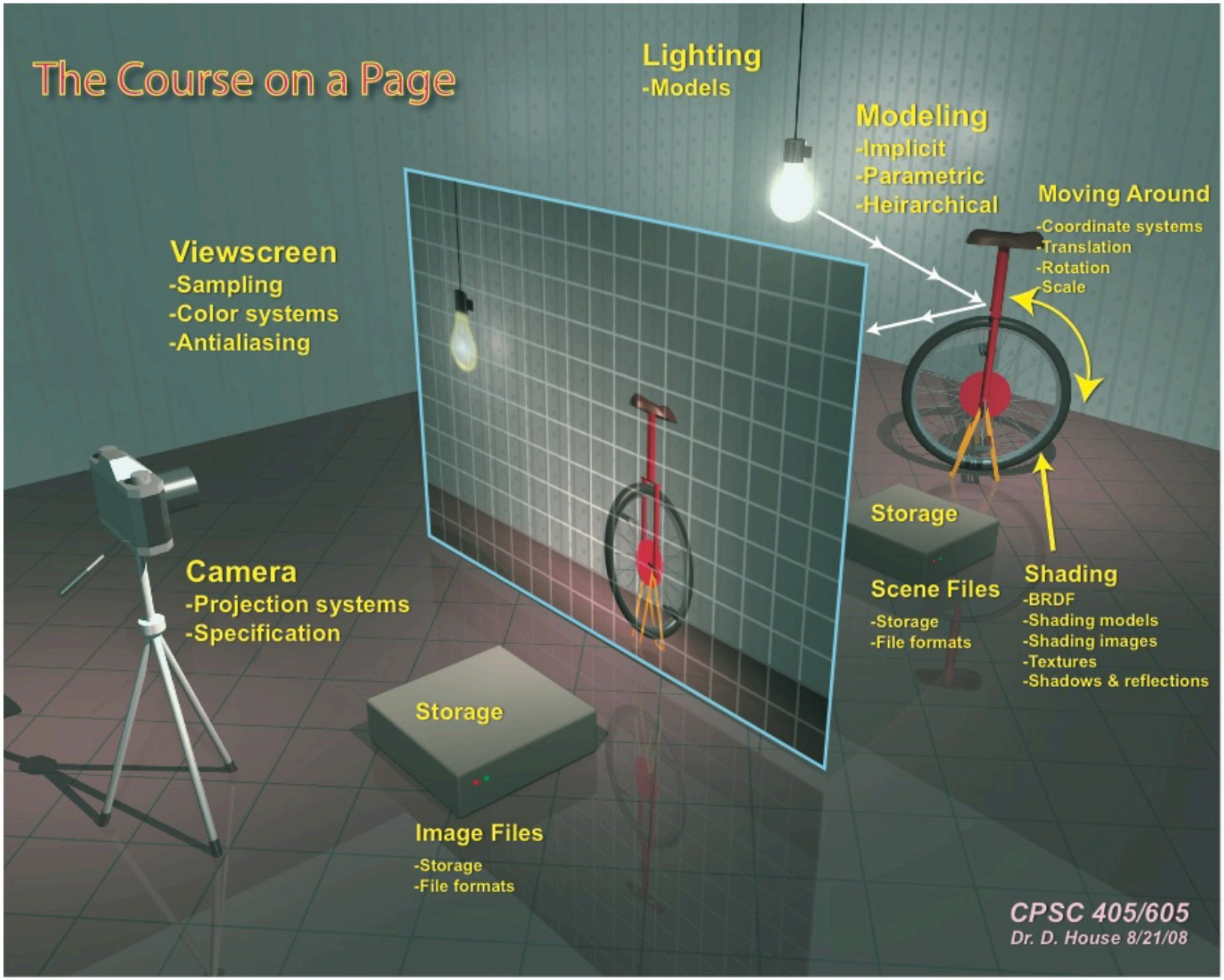
Modeling and Rendering


The Course on a Page
Lighting
-Models
Modeling
-Implicit
-Parametric
-Heirarchical
Moving Around
-Coordinate systems
-Translation
-Rotation
-Scale
Viewscreen
-Sampling
-Color systems
-Antialiasing
Storage
Camera
-Projection systems
-Specification
Scene Files
-Storage
-File formats
Shading
-BRDF
-Shading models
-Shading images
-Textures
-Shadows & reflections
Storage
Image Files
-Storage
-File formats
CPSC 405/605
Dr. D. House 8/21/08

Prelude: The Pointing Problem

- 2D screen and mouse input, 3D world
  - Keyboard
  - Object snaps
  - Construction planes
    - Explicit planes (including primary axes)
    - Implicit planes (existing geometry surfaces)
- Directional Locks? (ortho, etc)
- Multiple views (top, front, side) ?

Prelude: Vector vs. Raster

- Vector graphics
  - Objects (primitives: lines, circles, arcs, text, etc.)
  - Attributes (color, line weight, line style, layer, etc.)
  - Scalable without loss of quality
  - A data format (actual display probably raster)
- Raster graphics
  - Colors “samples” out of a continuous image
  - Resolution (pixels per frame or pixels per inch)
  - Colors: indexed (“by number”) or direct (bit-depth) 8 b/px->256 colors, 24 b/px->16million
  - Loses quality if scaled too much.

Modeling: data types

- **Point cloud**
  0D: points in space, possibly with color
- **Wireframe** data
  1D: edges (lines) in space, as if made of wire.
- Boundary representation (**B-REP**) data
  2D: “cardboard” or paper models, possibly open.
- **Solid models**
  3D: mathematically guaranteed to “hold water
- **Voxel** (“volume element”) 3D version of pixel
  3D: stacks of sugar cubes

3D data: Primitives & Attributes

- Primitives: Stored geometry data
  - 2D: Lines, circles, arcs, splines, rectanges, polygons
  - 3D: Polygons, bricks, regular solids (sphere, tetrahedron, cone, etc.)
- Attributes: Qualities that can be assigned
  - Appearance: (textures, visibility, shadow-casting/receiving, glow, rendering style, etc.)
  - Organizational: (name, layer, etc)
- Pseudo-primitives: interface vs. storage
  - Drawing features presented in the user interface as primitives, but not stored that way.

Modeling: Simple Operations

- Shape-preserving transformations
  - Translation
  - Rotation
  - Scaling
  - Mirroring
- Into 3D
  - Extrusion (uniform 2D sxn, straight line path)
  - Revolution (uniform 2D sxn, circular path)
  - Sweeps (uniform 2D sxn, arbitrary path)

Modeling: Complex Operations

- Booleans (intersection, union, difference, split)
- Lofting (connecting multiple arbitrary sxns)
- NURBS (mathematical formulation with
  - "Patches" (bounded regions)
  - Edges match (f(x,y) = g(x,y))
  - Slopes match(f'(x,y)=g'(x,y))
  - Curvatures match (f''(x,y)=g''(x,y))
- Metaballs (regions of influence & iso-surfaces)

Surfaces

- Normals (perpendicular to surface at a point)
  - surface has an “orientation” (front and back)
  - Can be found by cross-product of two vectors in the surface (e.g. two edges of polygon)
  - Concave polygons give ambiguous normals
  - Twisted-polygons don’t have uniform normals, causing problems in booleans and rendering
- Triangles solve many problems
- Polygonal approximations to surfaces are common.

Rendering

- Any display is a rendering
- Graphics Pipeline: alternative sequences of operations -- different quality vs. time
- Pipeline must address 3 basic problems:
  - Projection
  - Hidden surfaces
  - Shading (coloring & brightness)
- Some pipelines have been made into hardware

Projection: what kind of picture?

- "Squashing" 3D data into a 2D world
  - Parallel (axonometric) projection
  - Perspective projection
- Camera analogy
  - Position (eye-point)
  - Orientation (view-vector, to focal-point)
  - Lens ("normal"=perspective, "telephoto"=parallel)
  - "camera-up" direction (usually Z)

Hidden Surfaces: Can I see it?

- **Culling** or backsiding (based on orientation, surface normal compared to view-vector)
- **Depth sorting** (based on pixel over-writing, requires sorting polys by eye-space-Z, fails on intersections. Flat shading only.)
- **Z-buffering** (uses image-depth ("z-") memory ("buffer") in addition to color image memory, draws in image only if depth is less. Supports smoothing.

Shading: How bright is it?

- Diffuse vs. Specular assumptions
- Diffuse reflection, intensity based on cosine of angle of incidence.
- Answer depends on
  - Position of light source (in-model vs distant)
    - In-model -> angle varies across even flat polygons
  - Curvature of surface (vs 'smooth shading')
    - Smoothing -> simulate curvature (varying angle) by combining and interpolating normals or intensities

Shading: What color is it?

- Three ways to get the color:
  - Uniform (constant color)
  - Look-up-table (mapped color) – image maps
  - Calculated (mathematical) color – procedural maps (no repeats, wrap around edges, oriented)
- Relation of color pattern to Geometry?
  - Mapping type (planar, cubic, cylindrical, spherical)
  - Scaling, orientation, position

Lights: Sources of light

- Ambient (uniform, non-directional, mythical)
- Distant (directional, uniform, “sunlight”, *flat*)
- Point (radiant, fall-off, in-model, variable)
- Cone or Spot (direction with dispersion, fall-off (1- or 2-way), in-model)
- Projector (Cone with color variation by texture)
- Real lights (line, area, measured, environment)
- Faking lights (glow, fall-off, multiple)

Shadows: Geometry & Light

- Shadows not related to shading
- What the light-source “sees” vs. the camera
  - Project “shadow volumes” thru model (polys)
  - Compute “light’s view” of scene, project as texture, smooth edges (“soft shadows”)
  - Cast “shadow-ray” towards light(s), test for intersections
- Complexity = time, so geom att’s for “receives shadows”, “casts shadows”, etc. and light source attributes for “casts shadows” and type of shadow.

Shading redux: Radical approaches

- **Raytracing**: specular reflections, not diffuse
- Trace from pixel back into scene to establish tree of reflections, refractions & sources
- Images show *reflection*, *refraction*, but not *caustics*
- **Radiosity**: uses theory of energy exchange
- Depends on geometry, not camera (anim!)
- Renders *color-bleed*, *indirect lighting*, etc.
- *Simulates* rather than *renders*.

Reflections on reflection

- **Diffuse** (beam of light bounced in all directions). What you see depends on what light sources that surface sees. Phong, et al.
- **Specular** (beam of light bounced (reflected) in one direction, possibly transmitted too (with refraction). What you see depends on the history of the beam reaching your eye. Raytracing.
- **Diffuse inter-reflection** or **Global Illumination** (light bounces in all directions. Surfaces considered secondary lights. What you see depends on energy reaching surface). Radiosity.

Radiosity upside

- Most architectural materials are diffuse
- Most architectural environments have subtle lighting effects that are important.
- Radiosity renders inter-reflection phenomena: color bleed and indirect-lighting.
- Radiosity can be camera-independent, supporting high quality real-time walk-thrus.

Time based media: authoring

- What can be changed?
  - Camera position/etc. -> Fly-through
  - Geometry position/etc. -> Animated scene
  - Textures? (roto-scoping), etc. (varies)
- Managing change over time
  - Story-board (designing)
  - Key frames
    - ‘tweening (linear vs. spline)
    - Easing (spline tweening in time)
  - “Score” (parts for all players)

Animations: resource management

- Computation, storage and playback are resource intensive.
- Disk storage and communications bandwidth
  - COmpressor-DECompressor (CODEC) software
    - H.264, Photo JPEG, mpeg, etc.
- Sound can take more space than imagery
  - Stereo, sample freq, etc.
- Codec choice will influence playback options

Miscellaneous

- **Safe-area** – portion of screen guaranteed to be visible
- **NTSC color gamut** – colors a traditional TV screen can display (not all colors)
- **Color smear** – saturated color smears on CRT TVs due to cheap electronics/standards.
- **Quicktime** – Apple's time-based media framework: many codecs.
- **AVI / WMV** – Microsoft's time-based media frameworks: many codecs.