EECS 487: Interactive Computer Graphics

Lecture 15:

- Scenegraph
- Lighting and Reflection

Hierarchical Modeling

Hierarchical modeling is essential for transforming objects with attached parts, e.g., in animation:

- eyes move with head
- hands move with arms
- feet move with legs
- …

Without such structure the model falls apart, e.g., eyes don't follow when head moves

This idea can be extended to the entire scene $\longrightarrow$ scene graph

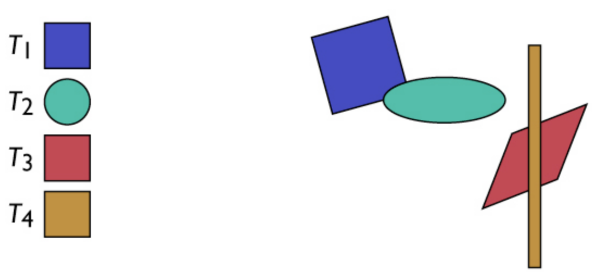
- collect every objects into a single hierarchy

Scene Representation

How to represent a scene?

- list of objects
- transform of each object
  - can use minimal primitives: an ellipse is a transformed circle
  - transform applies to points on object

Scene representation: data structures+transforms

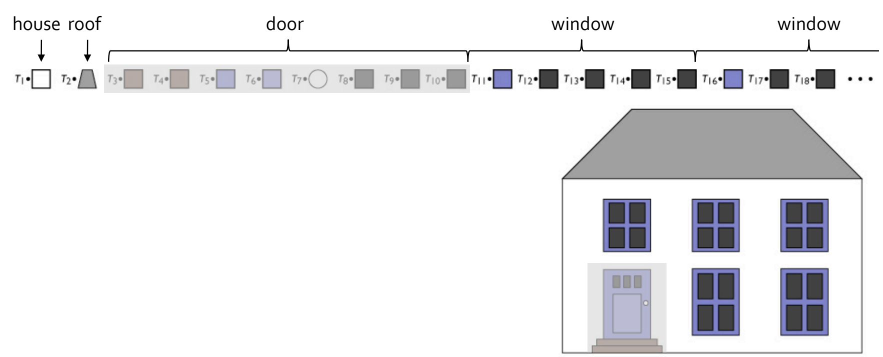


Marschner

Scene as a Flat List of Objects

Can represent scene as a flat list of objects

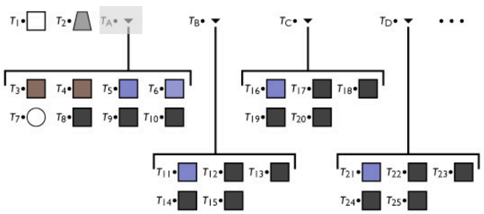
- but editing (e.g., delete) requires updating many nodes



Marschner

Hierarchical Representation

Introduce a new abstract data type: group
- treats a set of objects as one object (group)
- contains list of references to member objects
- lets the data structure reflect the rendering structure
- enables high-level editing by changing just one node



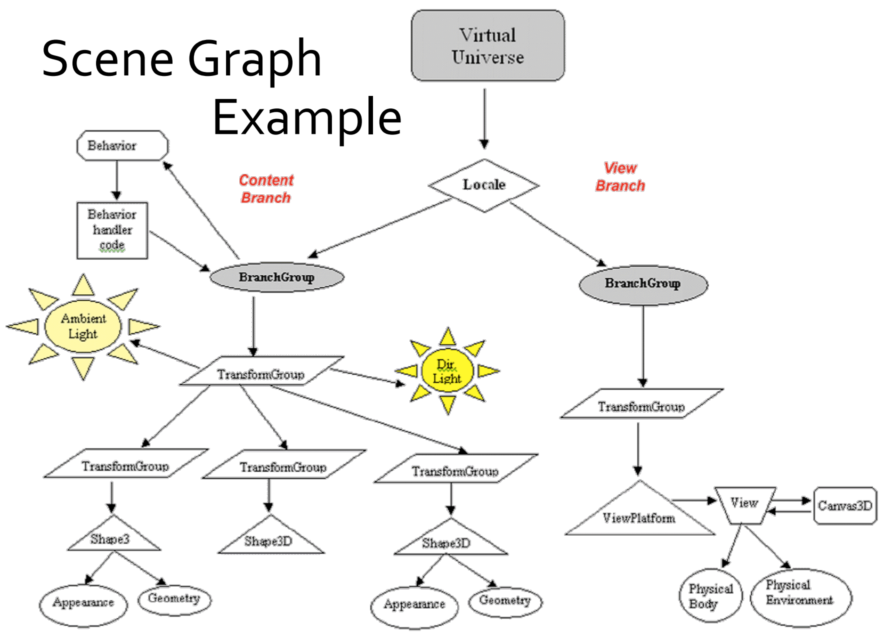
Marschner

Scene Graphs

All parts of the scene are represented in one graph
- each node in the graph is one scene element, including
  - objects, cameras, lights, materials, transformations, …
  - switch/select: specify which children to enable, etc…
  - simulation procedures, shaders
  - other scene graphs
- simplest form: tree
  - every node has one parent
  - interior nodes = groups
  - leaf nodes = objects in the scene
  - edges = membership of object in group
  - transforms are associated with nodes or edges
    - each transform applies to all geometry below it

Marschner,TP3,Schulze

Scene Graph Example



Marschner,Schulze

The Graphics Software Stack

applications
- modeling programs use scene graph to manage complexity, e.g., Maya, 3dsmax, etc.
- games, visualization, virtual reality, web apps

scene graph/rendering engine
- as "scene graph API": middleware for graphics API
- as "3D toolkit": implement graphics functionalities commonly required in applications

graphics API
- interface to graphics hardware, e.g., OpenGL, Direct3D

GPU

Schulze

Scene Graphs

To draw the scene, the graph is walked

- each time a node is traversed, either the rendering state is changed or something is rendered with the current state
- an operation performed on a node, such as rendering, culling, and transform, affects all of its children
  - e.g., traversing a light node turns on the light for all its children
  - transforms accumulate along path from root to leaves

Makes modeling and animation of complex scenes easier by breaking them down into a hierarchy of simpler ones with their own local behavior

Chenney

Scene Graph Advantages

Hierarchical processing

- each sub-hierarchy naturally defines a bounding volume, e.g., for culling, collision detection, or ray-tracing computation

Object-oriented paradigm

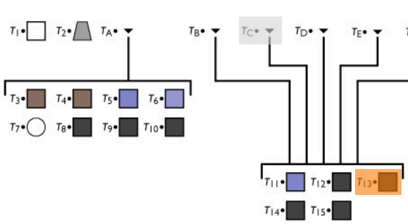
- each object is defined in its own local coordinate systems
- objects can have other properties besides shape
  - color, shading parameters
  - approximation parameters (e.g., degree of tessellation)
  - user interaction, etc …
  - property nodes can be applied to sub-hierarchy, e.g., paint entire window green
- objects are self-contained and re-usable
- instancing: an object can be a member of multiple groups

TP3,Marschner

Instancing Example

Allow multiple references to nodes

- reflects more of drawing structure
- allows editing of repeated parts in one operation

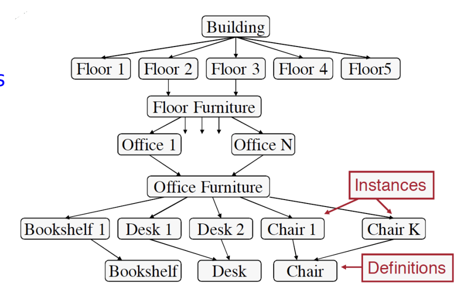


Marschner

Multiple Instantiations

Object defined once, used many times, in many places in the scene

- an object with multiple instantiations has multiple parents
  - not the "make a copy" instantiation of C++
- transforms still accumulate along path from root to leaf
  - objects may have multiple paths from root to leaves
  - transform may be different for each instance
- graph is no longer a tree, but a directed acyclic graph (DAG, no cycle)



Schulze,Marschner,James

Scene Graph Toolkits and APIs

No broadly accepted standard

APIs focus on different applications

- OpenSceneGraph (openscenegraph.org)
  - scientific visualization, virtual reality, GIS
  - optimized for memory requirements
  - open source version of historical scene graph APIs for SGI IRIS GL
    - Open Inventor (oss.sgi.com/projects/inventor/)
    - OpenGL Performer (oss.sgi.com/projects/performer/)
- Ogre3D (www.ogre3d.org) and a host of others
  - games, optimized for high-performance rendering (speed)
- Javascript scenegraphs, WebGL compatible:
  - three.js (threejs.org)
    - "a lightweight 3D library with a very low level of complexity"
  - sceneJS (scenejs.org)
    - CAD, medical, and engineering visualization
- Modeling systems' proprietary libraries
  - optimized for editing flexibility

Basic Scene Graph Operations

High-level scene management

- edit transformation
  - need good UI
- transform object in world coordinate frame
  - traverse path from root to leaf
- grouping and ungrouping
- re-parenting
  - moving node from one parent to another

Marschner

Common Functionalities

Resource management

- asset management (geometry, textures, materials, animation sequences, audio)
- shader management
- memory management
- multi-threading
- (server clustering)

Rendering libraries:

- bump mapping
- shadows
- particle system

Schulze

Performance Optimizations

Culling

- early discard of invisible parts of scene

Level-of-detail

- use lower poly count version for distant (small) object

Computing bounding volume hierarchy for

- culling
- collision detection
- rendering, e.g., ray-tracing, qsplat

Scene graph compilation/optimization

- render objects with similar attributes (textures, materials, shaders, geometry) in batches
  - efficient use of low-level API
  - avoid state changes in rendering pipeline

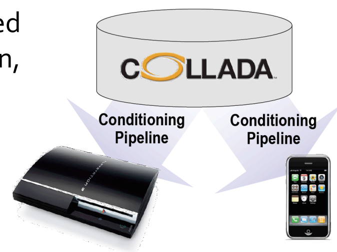
Serious scene graphs should have implementation of these techniques

Schulze

Scene Graph Encoding

Collada

- asset exchange using an XML schema
  - e.g., passing models to a physics engine
- asset transformation from high-level modeling description to platform-specific optimized description
- can describe everything to do with a scene: geometry with full skinning, advanced material and visual effects, animation, physical properties and collisions



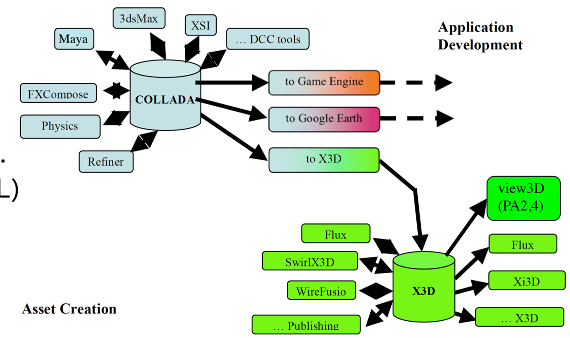
Scene Graph Encoding

X3D (`web3d.org`)

- VRML with XML syntax, replaced VRML in July 2004
- primary goal is for interactive visualization of 3D assets
- specifies behaviors and interactions and includes
  - a run-time model that enables viewing, navigation, picking, and scripting
  - an API to manipulate the scene-graph at runtime

X3**DOM**

- HTML5/X3D integration
- declarative 3D (vs. procedural WebGL)
- `x3dom.org`



EECS 487: Interactive Computer Graphics

Lecture 15:

- Scenegraph
- Lighting and Reflection

Object Appearance in CGI

Object appearance in CGI depends on its

- shape: the geometry of its surfaces and position wrt camera
- shade: its illumination environment and optical properties

Rendering program separates:

- geometric processing: transformation, hidden surface removal, etc. from
- optical processing: propagation and filtering of light

Illumination Models

A rendering process can be modeled as an integral equation representing the transport of light through the environment ⇒ the rendering equation

Local illumination: an approx. of the rendering eqn.
- assumes light is scattered only once: light from light source is reflected by a surface and modulated on its way towards the eye

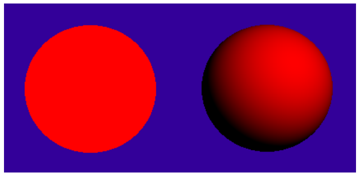
Global illumination:
- light rays traveling from light to surface may be
  - blocked by intervening surfaces (shadows) or
  - bent or scattered by intervening material (refraction and atmospheric effects) or
- light arriving at a surface may come indirectly via another surface (reflection and color bleed)

Global Illumination Effects

Properly determining the right color is really hard
- translucency
- refraction
- particle scattering
- color bleed

Local Illumination

A photograph of a lit sphere shows not a uniformly colored circle but a circular shape with many gradation or shades of color, giving the impression of 3D



Local illumination consists of two major aspects:

1. light source distribution function
2. surface reflectance distribution function

Light Sources

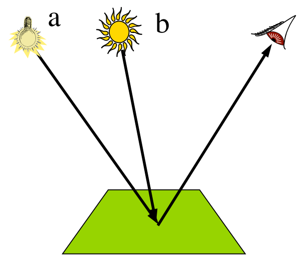
Light is approximated by the RGB components emitted from the light source

For light, the RGB coefficients represent percentages of full intensity of each color
- $\mathbf{c} = (1.0, 1.0, 1.0)$ is white
- $\mathbf{c} = (0.5, 0.5, 0.5)$ is white at half intensity, which appears gray

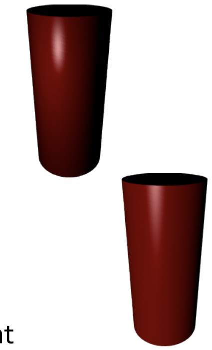
Light Sources

Grassman's Laws:

- if two lights emit at $\mathbf{c}_1 = (R_1, G_1, B_1)$ and $\mathbf{c}_2 = (R_2, G_2, B_2)$, the light that arrives at the eye is
  $\mathbf{c} = \mathbf{c}_1 \oplus \mathbf{c}_2 = (R_1 + R_2, G_1 + G_2, B_1 + B_2)$
- scaling light intensity: $\mathbf{c}(s\ \mathrm{a}) = s\ \mathbf{c}(\mathrm{a})$

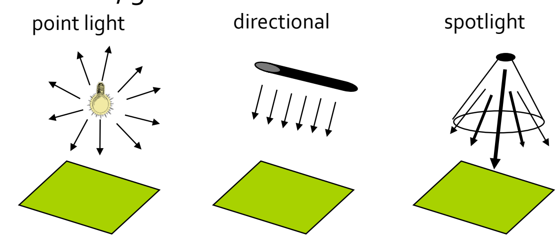


Light Sources



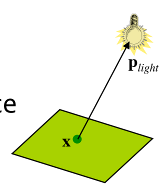
Types of light sources

- point light, e.g., light bulb: light direction changes over surface
- directional light, e.g., sunlight: "distant" light, direction is constant
- spotlight: point source with directional fall-off
- area source: luminous 2D surface: radiates light from all points on surface, generates soft shadows



Point and Directional Sources

Point light: $\mathbf{l}(\mathbf{x}) = \dfrac{\mathbf{p}_{light} - \mathbf{x}}{\left\|\mathbf{p}_{light} - \mathbf{x}\right\|}$



- light arriving at a point ($\mathbf{x}$) on the surface
- $\mathbf{l}$ always points towards the light
  - must be normalized
- to specify an OpenGL light at light position $(1, 1, 1)$:

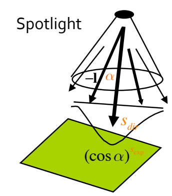
```
Glfloat light_position[] = { 1.0, 1.0, 1.0, 1.0 };
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Directional light: $\mathbf{l}(\mathbf{x}) = \mathbf{l}_{light}$

- the $\mathbf{l}$ vector does not vary across the surface
- OpenGL light shining from direction $(1, 1, 1)$:

```
Glfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Spotlight



Point source, with intensity a function of $-\mathbf{l}$, specified with:

- position: the location of the source
  `glLightfv(GL_LIGHT0,GL_POSITION,light_posn);`
- direction (s_{dir}): the center axis of the light
  `glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,light_dir);`
  - intensity maximal along direction s_{dir}
  - when light moved, direction must be updated along with position
- cut-off (α): how broad (in degree) the beam is
  `glLightfv(GL_LIGHT0,GL_SPOT_CUTOFF,45.0);`
  - intensity falls off angling away from s_{dir}
- exponent (s_{exp}): how the light tapers off at the edges of the cone
  `glLightfv(GL_LIGHT0,GL_SPOT_EXPONENT,1.0);`
  - intensity scaled by exponent: $s_{spot} = \max(-\mathbf{l} \bullet s_{dir}, 0)^{s_{exp}}$

OpenGL Light Sources

`glLightfv(lightname,param,value)`

- parameters
  - `GL_AMBIENT`
  - `GL_DIFFUSE`
  - `GL_SPECULAR`

  - `GL_POSITION`
  - `GL_SPOT_DIRECTION`
  - `GL_SPOT_CUTOFF`
  - `GL_SPOT_EXPONENT`

  - `GL_CONSTANT_ATTENUATION`
  - `GL_LINEAR_ATTENUATION`
  - `GL_QUADRATIC_ATTENUATION`

How Lights Are Positioned

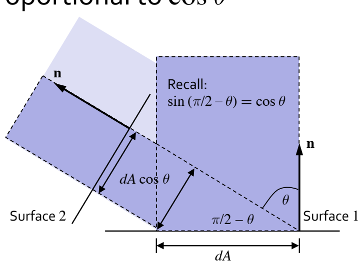
All computations are carried out in eye coordinates

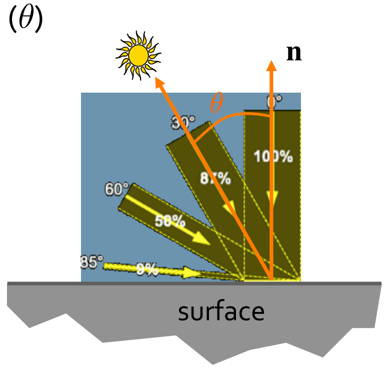
- store lights in eye coordinates
- lights converted to eye coordinates using current ModelView transform
- lights move with eye
  - default `GL_LIGHT0` – directional from the back, with specular component
  - `glEnable(GL_LIGHTING);`
  - `glEnable(GL_LIGHT0);`
  - don't forget to set the normals properly

Why is Winter Light Weaker than Summer Light?

The amount of light received and reflected by a surface depends on angle of incidence (θ)

- bigger at normal incidence
- smaller slanted, by how much?
- Lambert's Cosine Law: proportional to $\cos\theta$





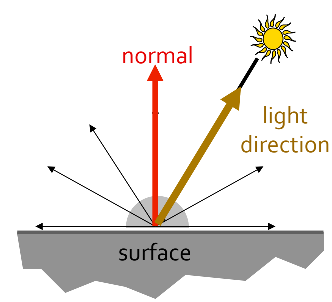
Durand, FvD94

Surface Normal

The intensity of a surface color depends on the orientation of the surface wrt the light and viewer

The surface normal vector describes this orientation at a point

- is perpendicular to the tangent plane of the surface (recall how to transform normals)
- is often called just "the normal vector" or "the normal"
- will use $\mathbf{n}$ or $\mathbf{N}$ to denote



Normals are either supplied by the user or automatically computed

Specifying Normals

Normals can be specified using `glNormal3*()`

Normals are associated with vertices

Specifying a normal sets the current normal

- remains unchanged until user alters it
- usual sequence:
  `glNormal3, glVertex,`
  `glNormal3, glVertex,`
  `glNormal3, glVertex,…`

Normals are not normalized by default

- can be automatically normalized by calling
  `glEnable(GL_NORMALIZE)` or
  `glEnable(GL_RESCALE_NORMAL)`
- but this is slow, instead normalize as needed

Material Appearance

Factors effecting materials appearance

- color
- texture
- intensity and shape of highlights
- glossiness

For surface color, the RGB coefficients represent percentages of reflected proportions of each color


Ngan,Hanrahan

OpenGL's Simple Reflectance Model

If the light is emitting $\mathbf{c}_l = (R_l, G_l, B_l)$ and the material reflects $\mathbf{c}_m = (R_m, G_m, B_m)$, the light that arrives at the eye is $\mathbf{c} = \mathbf{c}_l \otimes \mathbf{c}_m = (R_l R_m, G_l G_m, B_l B_m)$

A red ball in white light reflects red and absorbs green and blue

A red ball in green light appears black (no light is reflected)

OpenGL Lighting and Reflectance

```
/*  Initialize material property, light source,
    lighting model, and depth buffer. */
void init(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

   glClearColor(0.0, 0.0, 0.0, 0.0);
   glShadeModel(GL_SMOOTH);

   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}
```

Clamping vs. Scaling

RGB coefficients must be in $[0.0, 1.0]$ range

Reflected color $\mathbf{c} = (\mathbf{c}_1 \oplus \mathbf{c}_2) \otimes \mathbf{c}_m =$
$[(R_1 + R)R_{m2}, (G_1 + G_2)G_m, (B_1 + B_2)B_m]$ may have component > 1.0, e.g., bright orange is $(2.5, 1.5, 0.5)$

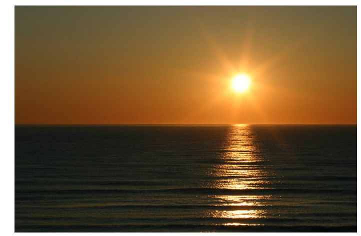
- if clamped to 1.0, $(1.0, 1.0, 0.5)$ is yellow
- if scaled by $1/2.5$ instead, we get $(1.0, 0.6, 0.2)$, which retains the original orange hue and saturation

Torrance-Sparrow Reflectance Model

Microfacet Theory: model surface as a collection of tiny mirrors [Torrance & Sparrow 1967]

Example of microfacet distribution:

- surface of the ocean
- viewer sees "bright" pixels
  - when microfacets are pointing halfway between the sun and the eye
  - other microfacets are obstructed, either in shadow or hidden

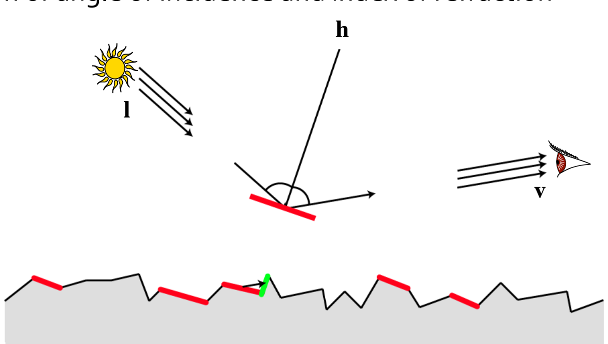


Durand

Microfacet Model [Cook&Torrance82]

Reflectance at $(\mathbf{l}, \mathbf{v})$ is a product of the

- number of mirrors oriented halfway between $\mathbf{l}$ and $\mathbf{v}$,
- percentage of unblocked mirrors, and
- Fresnel coefficient: fraction of light reflected (not absorbed), function of angle of incidence and index of refraction



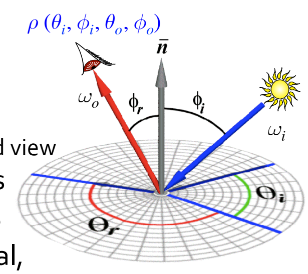
Durand

Measure of Reflectance: BRDF

Different material emits, absorbs, or reflects light differently

Bidirectional Reflectance Distribution Function (BRDF) $\rho\ (\omega_i, \omega_o)$:

- ratio of radiance incoming from one direction that gets reflected in another direction
  - relates incoming light energy to outgoing
  - function based on directions of incidence and view
- unifying framework for many materials
- (assume isotropic material, reflectance is invariant to rotation about the normal, unlike velvet or satin, e.g.)



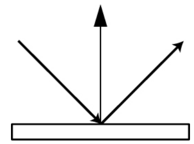
Lozano-Perez

Types of Reflection

We generally recognize 3 types of reflection:

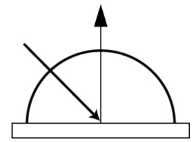
Ideal Specular

- Reflection Law
- Mirror



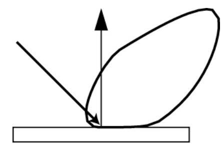
Ideal Diffuse

- Lambert’s Law
- Matte



Rough Specular

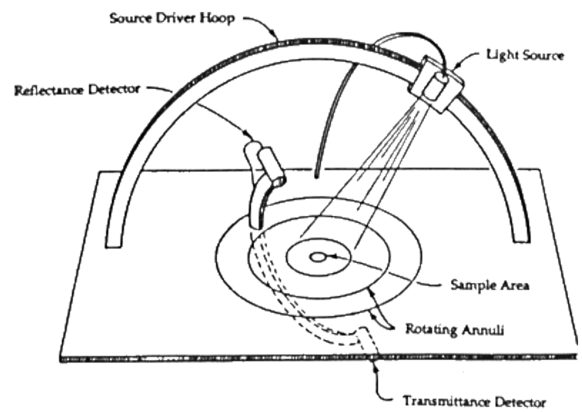
- Directional diffuse
- Glossy

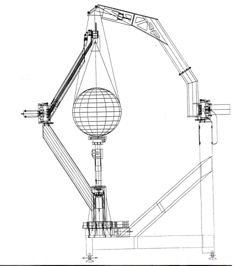


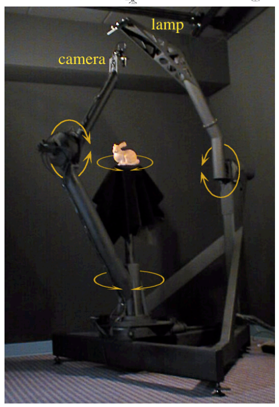
Hanrahan

How to Obtain BRDF?

Gonioreflectometer







Ward,Hanrahan