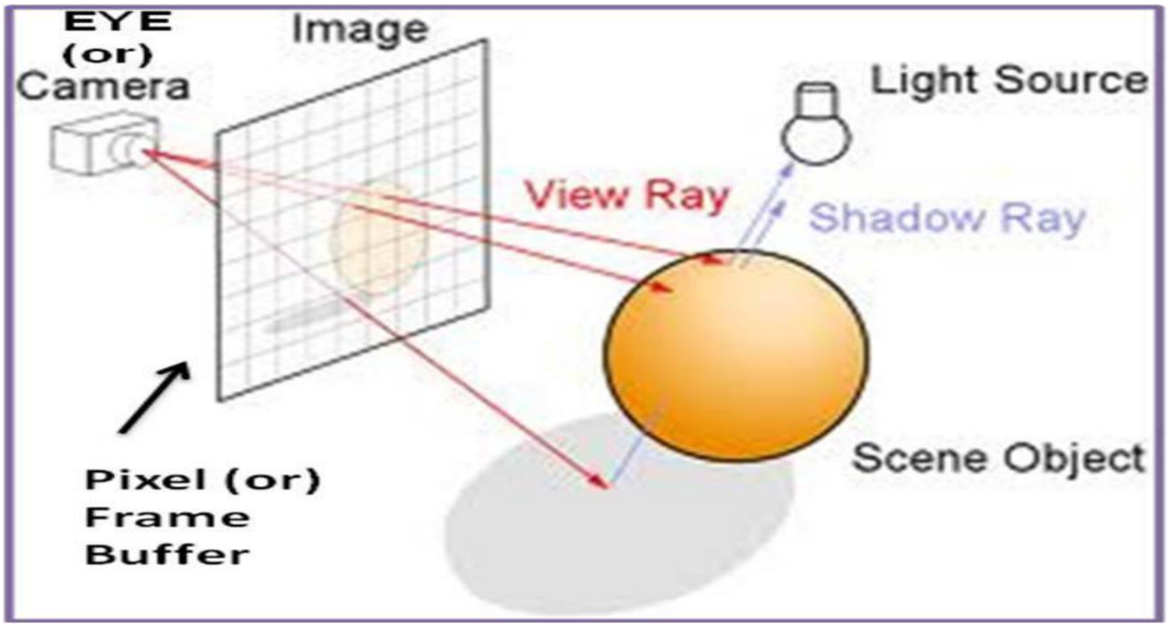
UNIT 6

Overview of Ray Tracing Intersecting rays with other primitives – Adding Surface texture –Reflections andTransparency – Boolean operations on Objects.

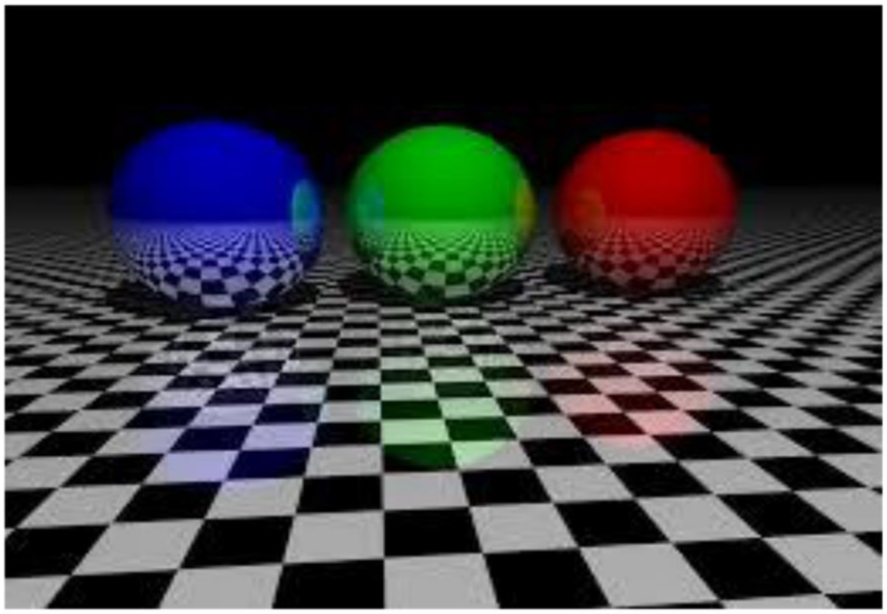
1 OVERVIEW OF RAY TRACING:

- Ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects.
- The technique is capable of producing a very high degree of visual realism, usually higher than that of typical scan line rendering methods, but at a greater computational cost.

- Ray tracing Provides a related, but even more powerful, approach to rendering scenes.
- A Ray is cast from the eye through the center of the pixel is traced to see what object it hits first and at what point.

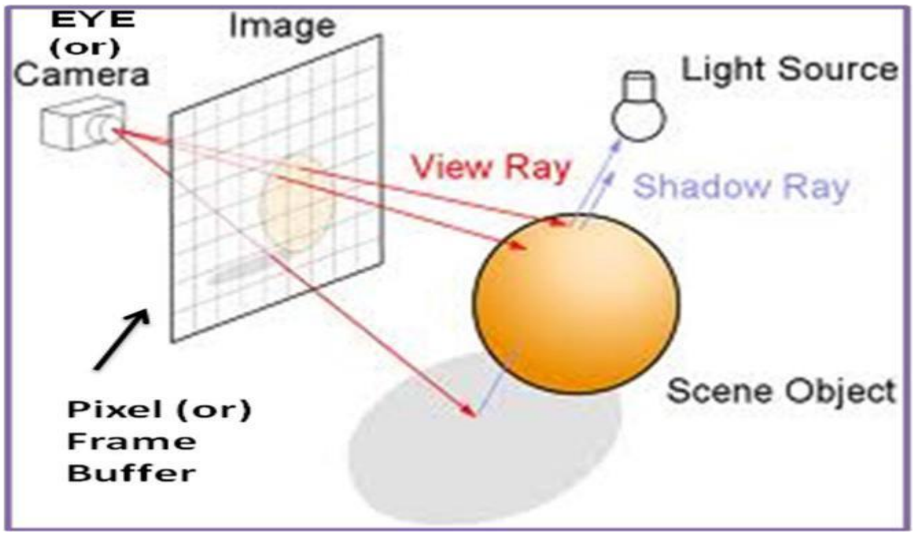


- The Resulting color is then displayed at the pixel, the path of a ray traced through the scene, interesting visual effects such as shadowing, reflection and refraction are easy to incorporate and producing dazzling images.



- Ray tracing can create realistic images.
- In addition to the high degree of realism, ray tracing can simulate the effects of a camera due to depth of field and
- aperture shape Ray tracing is capable of simulating a wide variety of optical effects, such as reflection and refraction, scattering,and dispersion phenomena (such as chromatic aberration).
- Descriptions of all then Objects are stored in an **object list.**
- The ray that interacts the Sphere and the Cylinder.
- The hit spot (**P_{HIT}**) is easily found wit the ray itself.

  The ray of Equation at the Hit time $\mathbf{t_{hit}}$: $\mathbf{P_{HIT}=eye + dir_{r,c}t_{hit}}$



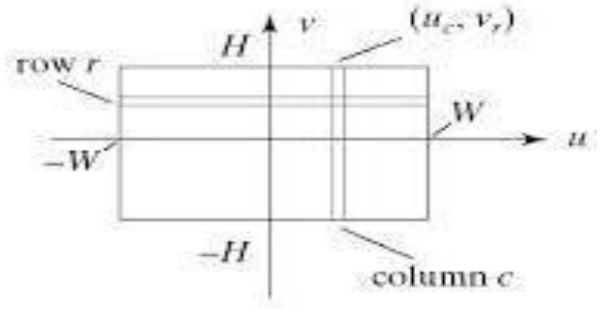
Pseudocode of a Ray Tracer

```
define the objects and light sources in the scene set up the
    camera for(int r=0 ; r < nRows ; r++)
    for(int c=0 ; c < nCols ; c++)
    {
        1.Build the rc-th ray.
        2.Find all interactions of the rc-th ray with objects in the
scene.
        3.Identify the intersection that lies closest to and infront
of the eye.
        4.Compute the Hit Point.
        5.Find the color of the light returning to the eye along the
ray from the point of intersection.
        6.Place the color in the rc-th pixel.
}
```

2 INTERSECTING RAYS WITH OTHER PRIMITIVES

First the ray is transformed into the generic coordinates of the object and then the various intersection with the generic object are computed.

1) Intersecting with a Square



The generic square lies in the z=0 plane and extends from -1 to 1 in both x and y.

The square can be transformed into any parallelogram positioned in space, so it is often used in scenes to provide this, flat surfaces such as walls and windows. The function hit(1) first finds where the ray hits the generic plane and then test whether this hit spot also lies within the square.

2) Intersecting with a Tapered Cylinder

The side of the cylinder is part of an infinitely long wall with a radius of L at z=0,and a small radius of S at z=1.This wall has the implicit form as F(x,

$$y, z)=x^2 + y^2 - (1 + (S - 1) z)^2, \text{ for } 0 < z < 1$$

If S=1, the shape becomes the generic cylinder, if S=0 , it becomes the generic cone. We develop a hit () method for the tapered cylinder, which also provides hit() method for the cylinder and cone.

3) Intersecting with a Cube (or any Convex Polyhedron)

The convex polyhedron, the generic cube deserves special attention. It is centered at the origin and has corner at (±1, ±1, ±1) using all right combinations of +1 and -1.Thus,its edges are aligned with coordinates axes, and its six faces lie in the plan.

The generic cube is important for two reasons.

- A large variety of intersecting boxes can be modeled and placed in a scene by applying an affine transformation to a generic cube. Then, in ray tracing each ray can be inverse transformed into the generic cube's coordinate system and we can use a ray with generic cube intersection routine.
- The generic cube can be used as an extent for the other generic primitives in the sense of a bounding box. Each generic primitives, such as the cylinder, fits snugly inside the cube.

PLANE	**NAME**	**EQUATION**	**OUTWARD NORMAL**	**SPOT**
0	TOP	Y=1	(0,1,0)	(0,1,0)
1	BOTTOM	Y=-1	(0,-1,0)	(0,-1,0)
2	RIGHT	X=1	(1,0,0)	(1,0,0)
3	LEFT	X=-1	(-1,0,0)	(-1,0,0)
4	FRONT	Z=1	(0,0,1)	(0,0,1)
5	BACK	Z=-1	(0,0,-1)	(0,0,-1)

4) Adding More Primitives

To find where the ray S + ct intersects the surface, we substitute S + ct for P in F(P) (the explicit form of the shape)

$$d(t) = f(S + ct)$$

This function is

- positive at these values of t for which the ray is outside the
- object. zero when the ray coincides with the surface of the object
- and negative when the ray is inside the surface.

The generic torus has the implicit function as

$$F(P) = \left(\sqrt{P_x^2 + P_y^2} - d\right)^2 + P_z^2 - 1$$

So the resulting equation d(t)=0 is quartic.

For quadrics such as the sphere, d(t) has a parabolic shape, for the torus, it has a quartic shape. For other surfaces d(t) may be so complicated that we have to search
numerically to locate t's for which d(.) equals zero. The function for super ellipsoid is

$$d(t) = ((S_x + C_x t)^n + (S_y + C_y t)^n)^{m/n} + (S_y + C_y t)^m - 1$$

where n and m are constant that govern the shape of the surface.

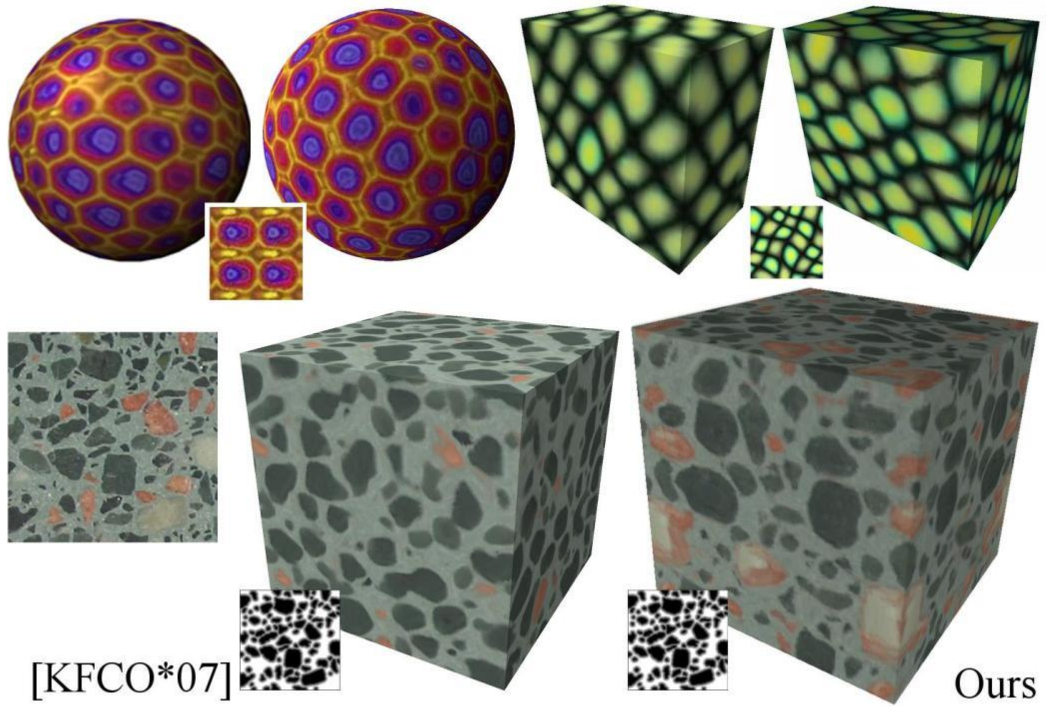
3 ADDING SURFACE TEXTURE

A fast method for approximating global illumination effect is environmental mapping. An environment array is used to store background intensity information for a scene. This array is then mapped to the objects in a scene based on the specified viewing direction. This is called as environment mapping or reflection mapping.

To render the surface of an object, we project pixel areas on to surface and then reflect the projected pixel area on to the environment map to pick up the surface shading attributes for each pixel. If the object is transparent, we can also refract the projected pixel are also the environment map. The environment mapping process for reflection of a projected pixel area is shown in figure. Pixel intensity is determined by averaging the intensity values within the intersected region of the environment map.

A simple method for adding surface detail is the model structure and patterns with polygon facets. For large scale detail, polygon modeling can give good results. Also we could model an irregular surface with small, randomly oriented polygon facets, provided the facets were not too small.

Surface pattern polygons are generally overlaid on a larger surface polygon and are processed with the parent's surface. Only the parent polygon is processed by the visible surface algorithms, but the illumination parameters for the surfac3e detail polygons take precedence over the parent polygon. When fine surface detail is to be modeled, polygon are not practical.



Texture Mapping:

A method for adding surface detail is to map texture patterns onto the surfaces of objects. The texture pattern may either be defined in a rectangular array or as a procedure that modifies surface intensity values. This approach is referred to as texture mapping or pattern mapping.

The texture pattern is defined with a rectangular grid of intensity values in a texture space referenced with (*s,t*) coordinate values. Surface positions in the scene are referenced with UV object space coordinates and pixel positions on the projection plane are referenced in *xy* Cartesian coordinates.

Texture mapping can be accomplished in one of two ways. Either we can map the texture pattern to object surfaces, then to the projection plane, or we can map pixel areas onto object surfaces then to texture space. Mapping a texture pattern to pixel coordinates is sometime called texture scanning, while the mapping from pixel coordinates to texture space is referred to as **pixel order scanning** or **inverse scanning** or **image order scanning**.To simplify calculations, the mapping from texture space to object space is often specified with parametric linear functions

$$U=f_u(s,t)=a_u s+ b_u t + c_u$$
$$V=f_v(s,t)=a_v s+ b_v t + c_v$$

The object to image space mapping is accomplished with the concatenation of the viewing and projection transformations.

A disadvantage of mapping from texture space to pixel space is that a selected texture patch usually does not match up with the pixel boundaries, thus requiring calculation of the fractional area of pixel coverage. Therefore, mapping from pixel space to texture space is the most commonly used texture mapping method. This avoids pixel subdivision calculations, and allows anti aliasing procedures to be easily applied.

The mapping from image space to texture space does require calculation of the inverse viewing projection transformation m_{VP}^{-1} and the inverse texture map transformation m_T^{-1} .

Procedural Texturing Methods:

Next method for adding surface texture is to use procedural definitions of the color variations that are to be applied to the objects in a scene. This approach avoids the transformation calculations involved transferring two dimensional texture patterns to object surfaces.When values are assigned throughout a region of three dimensional space, the object color variations are referred to as solid textures. Values from texture space are transferred to object surfaces using procedural methods, since it is usually impossible to store texture values for all points throughout a region of space (*e.g*) Wood Grains or Marble patterns Bump Mapping.Although texture mapping can be used to add fine surface detail, it is not a good method for modeling the surface roughness that appears on objects such as oranges, strawberries and raisins. The illumination detail in the texture pattern usually does not correspond to the illumination direction in the scene.

A better method for creating surfaces **bumpiness** is to apply a perturbation function to the surface normal and then use the perturbed normal in the illumination model calculations. This technique is called **bump mapping.**

If *P(u,v)* represents a position on a parameter surface, we can obtain the surface normal at that point with the calculation

$N = P_u \times P_v$Where P_u and P_v are the partial derivatives of *P* with respect to parameters u and v. To obtain a perturbed normal, we modify the surface position vector by adding a small perturbation function called a **bump function**.

$$P'(u,v) = P(u,v) + b(u,v)\, n.$$

This adds bumps to the surface in the direction of the unit surface normal $n=N/|N|$. The perturbed surface normal is then obtained as

$$N'=P_u' + P_v'$$

We calculate the partial derivative with respect to u of the perturbed position vector as

$$P_u' = \frac{\partial}{\partial u}(P + bn)$$

$$= P_u + b_u n + b n_u$$

Assuming the bump function b is small, we can neglect the last term and write

$$p_u' \approx p_u + b_u n$$

Similarly

$$p_v' = p_v + b_v$$

n.and the perturbed surface normal is

$N' = P_u + P_v + b_v (P_u \times n) + b_u (n \times P_v) + b_u b_v (n \times n)$.

But n x n =0, so that

$$N' = N + b_v (P_u \times n) + b_u (n \times P_v)$$

The final step is to normalize N' for use in the illumination model calculations.

Frame Mapping:

Extension of bump mapping is frame mapping.

In frame mapping, we perturb both the surface normal N and a local coordinate system attached to N. The local coordinates are defined with a surface tangent vector

T and a binormal vector

B x T x N.

Frame mapping is used to model anisotrophic surfaces. We orient T along the grain of the surface and apply directional perturbations in addition to bump perturbations in the direction of N. In this way, we can model wood grain patterns, cross thread patterns in cloth and streaks in marble or similar materials. Both bump and directional perturbations can be obtained with table look-ups.

To incorporate texturing into a ray tracer, two principal kinds of textures are used. With image textures, 2D image is pasted onto each surface of the object. With solid texture, the object is considered to be carved out of a block of some material that itself has texturing. The ray tracer reveals the color of the texture at each point on the surface of the object.

Solid Texture:

Solid texture is sometimes called as **3D texture**. We view an object as being carved out of some texture material such as marble or wood. A texture is represented by a function texture (x, y, z) that produces an (r, g, h) color value at every point in space. Think of this texture as a color or inkiness that varies with position, if u look at different points (x, y, z) you see different colors. When an object of some shape is defined in this space, and all the material outside the shape is chipped away to reveal the object's surface the point (x, y, z) on the surface is revealed and has the specified texture.

Wood grain texture:

The grain in a log of wood is due to concentric cylinders of varying color, corresponding to the rings seen when a log is cut. As the distance of the points from some axis varies, the function jumps back and forth between two values. This effect can be simulated with the modulo function.

rings(r) = ((int) r)%2

where for rings about z-axis,

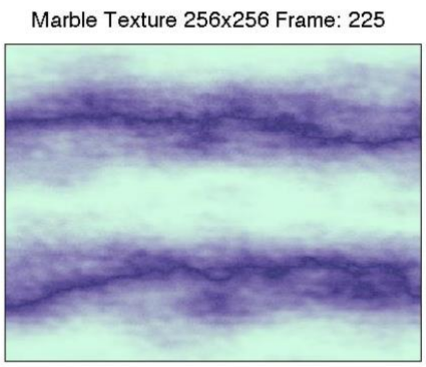
the radius $r = \sqrt{}x^2+y^2$.

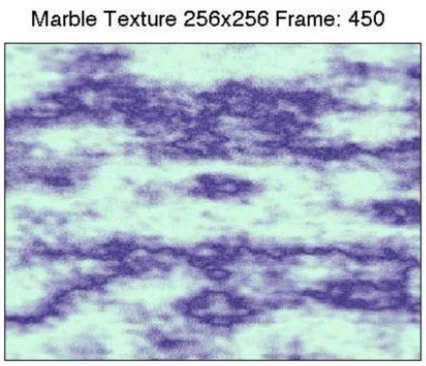
The value of the function rings () jumps between zero and unity as r increases from zero.

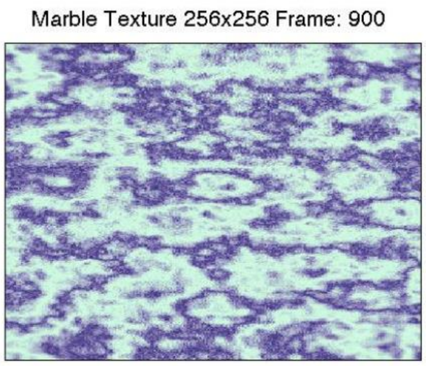
3D Noise and Marble Texture:

The grain in materials such as marble is quite chaotic. Turbulent riverlets of dark material course through the stone with random whirls and blotches as if the stone was formed out of some violently stirred molten material. We can simulate turbulence by building a noise function that produces an apparently random value at each point (x,y,z) in space. This noise field is the stirred up in a well-controlled way to give appearance of turbulence.

Marble Texture 256x256 Frame: 1







Turbulence:

A method for generating more interesting noise.
 The idea is to mix together several noise components: One that fluctuates slowly as you move slightly through space, one that fluctuates twice as rapidly, etc.
 The more rapidly varying components are given progressively smaller strengths
 The function
turb (s, x, y, z) = 1/2noise(s ,x, y, z) + 1/4noise(2s,x,y,z) +1/8 noise (4s,x,y,z).
The function adds three such components, each behalf as strong and varying twice as rapidly as its predecessor. Common term of a turb () is a

$$\text{turb}(s, x, y, z) = 1/2 \sum_{k=0}^{m} 1/2^{K}\text{noise}(2^{k}s, x, y, z).$$

Marble Texture:

Marble shows veins of dark and light material that have some regularity ,but that also exhibit strongly chaotic irregularities. We can build up a marble like 3D texture by giving the veins a smoothly fluctuating behavior in the z-direction and then perturbing it chantically using turb(). We start with a texture that is constant in x and y and smoothly varying in z.

Marble(x,y,z)=undulate(sin(2)).

Here undulate() is the spline shaped function that varies between some dark and some light value as its argument varies from -1 to 1.

4 REFLECTIONS AND TRANSPERENCY

The great strengths of the ray tracing method is the ease with which it can handle both reflection and refraction of light. This allows one to build scenes of exquisite realism, containing mirrors, fishbowls, lenses and the like. There can be multiple reflections in which light bounces off several shiny surfaces before reaching the eye or elaborate combinations of refraction and reflection. Each of these processes requires the spawnins and tracing of additional rays.

The figure 5.15 shows a ray emanating, from the eye in the direction dir and hitting a surface at the point P_h. when the surface is mirror like or transparent, the light I that reaches the eye may have 5 components

$$I=I_{amb} + I_{diff} + I_{spec} + I_{refl} + I_{tran}$$

The first three are the fan=miler ambient, diffuse and specular contributions. The diffuse and specular part arise from light sources in the environment that are visible at P_n. Iraft is the reflected light component ,arising from the light , Ik that is incident at P_n along the direction –r. This direction is such that the angles of incidence and reflection are equal,so

R=dir-2(dir.m)m

Where we assume that the normal vector m at P_h has been normalized.

Similarly I_{tran} is the transmitted light components arising from the light IT that is transmitted thorough the transparent material to Ph along the direction –t. A portion of this light passes through the surface and in so doing is bent, continuing its travel along – dir. The refraction direction + depends on several factors.

I is a sum of various light contributions, I_R and I_T each arise from their own fine components – ambient, diffuse and so on. I_R is the light that would be seen by an eye at Ph along a ray from P' to P_n. To determine I_R, we do in fact spawn a secondary ray from P_n in the direction r, find the first object it hits and then repeat the same computation of light component. Similarly IT is found by casting a ray in the direction t and seeing what surface is hit first, then computing the light contributions.

The Refraction of Light:

When a ray of light strikes a transparent object, apportion of the ray penetrates the object. The ray will change direction from dir to + if the speed of light is different in medium 1 than in medium 2. If the angle of incidence of the ray is θ_1, Snell's law states that the angle of refraction will be

$$\frac{\sin(\theta_2)}{C_2} = \frac{\sin(\theta_1)}{C_1}$$

where C_1 is the spped of light in medium 1 and C_2 is the speed of light in medium 2. Only the ratio C_2/C_1 is important. It is often called the index of refraction of medium 2 with respect to medium 1. Note that if θ_1 ,equals zero so does θ_2 .Light hitting an interface at right angles is not bent.

In ray traving scenes that include transparent objects, we must keep track of the medium through which a ray is passing so that we can determine the value C_2/C_1 at the next intersection where the ray either exists from the current object or enters another one. This tracking is most easily accomplished by adding a field to the ray that holds a pointer to the object within which the ray is travelling.

Several design polices are used,

1) Design Policy 1: No two transparent object may interpenetrate.
2) Design Policy 2: Transparent object may interpenetrate.

5 COMPOUND OBJECTS: BOOLEAN OPERATIONS ON OBJECTS

A ray tracing method to combine simple shapes to more complex ones is known as constructive Solid Geometry(CSG). Arbitrarily complex shapes are defined by set operations on simpler shapes in a CSG. Objects such as lenses and hollow fish bowls, as well as objects with holes are easily formed by combining the generic shapes. Such objects are called compound, Boolean or CSG objects.

The Boolean operators: union, intersection and difference are shown in the figure

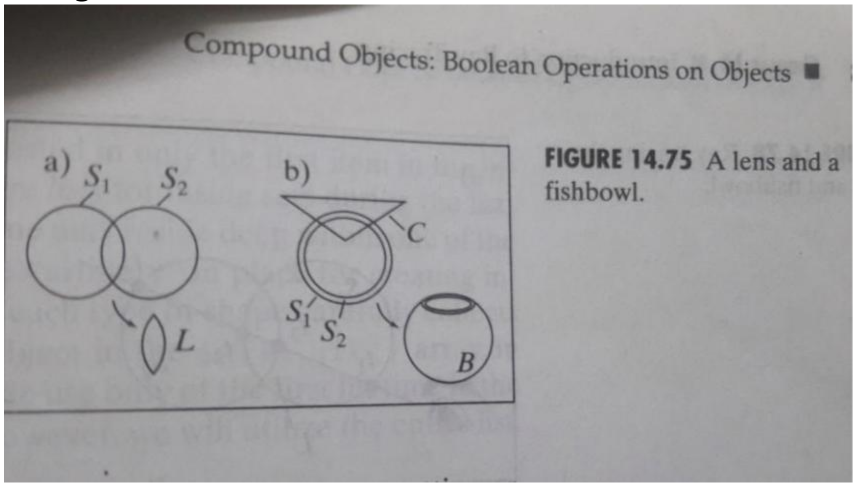

Compound Objects: Boolean Operations on Objects

FIGURE 14.75 A lens and a fishbowl.

Two compound objects build from spheres. The intersection of two spheres is shown as a lens shape. That is a point in the lens if and only if it is in both spheres. L is the intersection of the S_1 and S_2 is written as

$$L=S_1 \cap S_2$$

The difference operation is shown as a bowl.A point is in the difference of sets A and B, denoted A-B,if it is in A and not in B.Applying the difference operation is analogous to removing material to cutting or carrying. The bowl is specified by

$$B=(S_1-S_2)-C.$$

The solid globe, S_1 is hollowed out by removing all the points of the inner sphere, S_2,forming a hollow spherical shell. The top is then opened by removing all points in the cone C.

A point is in the union of two sets A and B, denoted AUB, if it is in A or in B or in both. Forming the union of two objects is analogous to gluing them together.

The union of two cones and two cylinders is shown as arocket.

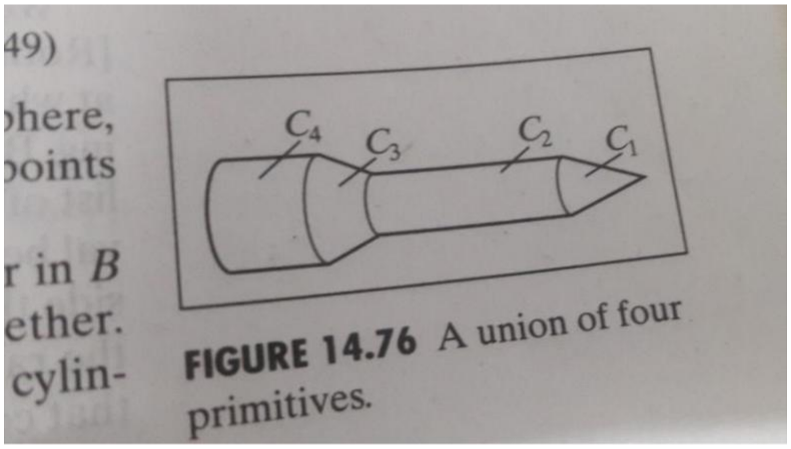


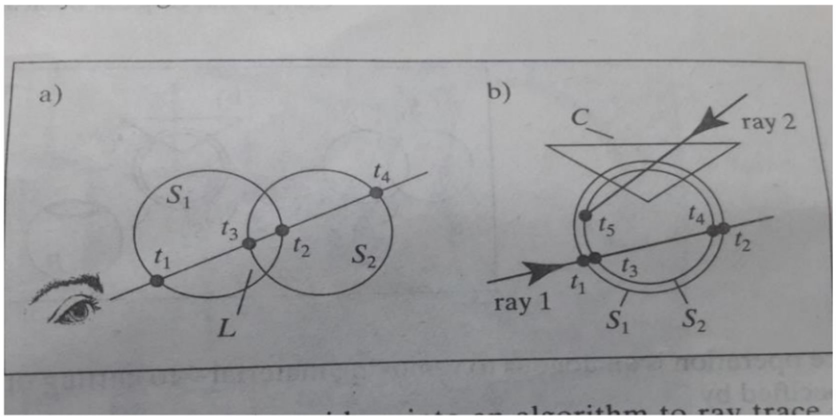
FIGURE 14.76 A union of four primitives.

$R=C_1 \cup C_2 \cup C_3 \cup C_4$.

Cone C_1 resets on cylinder C_2.Cone C_3 is partially embedded in C_2 and resets on the fatter cylinder C_4.

Ray Tracing CSC objects:

Ray trace objects that are Boolean combinations of simpler objects. The ray inside lens L from t_3 to t_2 and the hit time is t_3.If the lens is opaque, the familiar shading rules will be applied to find what color the lens is at the hit spot. If the lens is mirror like or transparent spawned rays are generated with the proper directions and are traced as shown in figure



Ray,first strikes the bowl at t1,the smallest of the times for which it is in S1 but not in either S2 or C. Ray 2 on the other hand,first hits the bowl at t5. Again this is the smallest time for which the ray is in S1,but in neither the other sphere nor the cone.The hits at earlier times are hits with components parts of the bowl,but not with the bowl itself.

Data Structure for Boolean objects:

Since a compound object is always the combination of two other objects say obj 1 OP Obj2, or binary tree structure provides a natural description.

Intersecting Rays with Boolean Objects:

We need to be develop a hit() method to work each type of Boolean object.The method must form inside set for the ray with the left subtree,the inside set for the ray with the right subtree,and then combine the two sets appropriately.

```
bool Intersection Bool::hit(ray in Intersection & inter)
{
   Intersection lftinter,rtinter;
   if (ray misses the extends)return false;
if (C) left
–>hit(r,lftinter)||((right–>hit(r,rtinter))) return
false;
return (inter.numHits > 0);
}
```

Extent tests are first made to see if there is an early out.Then the proper hit() routing is called for the left subtree and unless the ray misses this subtree,the hit list rinter is formed.If there is a miss,hit() returns the value false immediately because the ray must hit dot subtrees in order to hit their intersection.Then the hit list rtInter is formed.

The code is similar for the union Bool and DifferenceBool classes.
For UnionBool::hit(),the two hits are formed using

```
        if((!left-)hit(r,lftInter))**(|right-)hit(r,rtinter)))
        return false;
```

which provides an early out only if both hit lists are empty.
For differenceBool::hit(),we use the code

```
        if((!left−>hit(r,lftInter)) return
        false; if(!right−>hit(r,rtInter))

        {

                inter=lftInter;
                return true;
        }
```

which gives an early out if the ray misses the left subtree,since it must then miss the whole object.

Building and using Extents for CSG object:

The creation of projection,sphere and box extend for CSG object. During a preprocessing step,the true for the CSG object is scanned and extents are built for each node and stored within the node itself. During raytracing, the ray can be tested against each extent encounted,with the potential benefit of an early out in the intersection process if it becomes clear that the ray cannot hit the object.