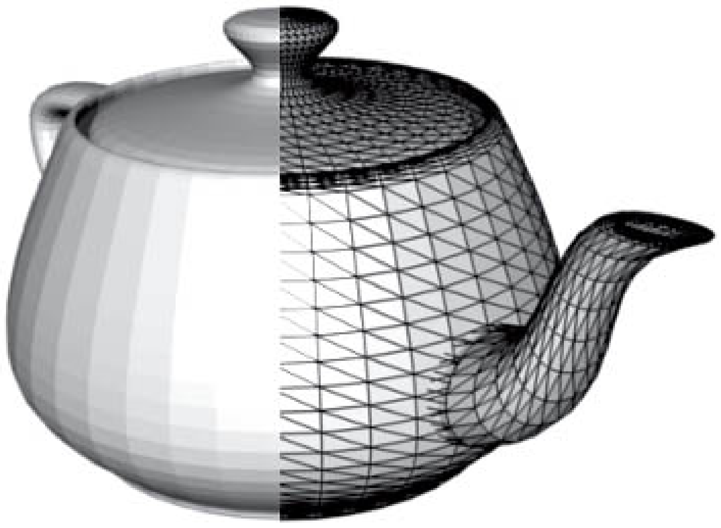
II. Computer Graphics

1. Coordinates and Transformations

- Most computer visualization systems use Cartesian coordinates
  - The axes are at right angles (orthogonal).
  - The position of a point in a Cartesian Coordinate System is determined by three distances that are typically used to specify spatial dimensions: width, height, and depth (x, y, z).
  - In a right-handed Cartesian Coordinate System, the thumb (the Z axis) points out of the X-Y plane when the hand rotates from the X to the Y-axis.

- In general, 3D virtual worlds are created out of 3D polygons, usually triangles or squares arranged to form 3D surfaces that comprise more complex objects.



◦ A line is the path of a point that moves in a plane and always takes the *shortest route* between any two of its positions and does not change direction.

– A segment $\mathbf{ab}$ contains precisely those points of the line through $\mathbf{a}$ and $\mathbf{b}$ that lie between $\mathbf{a}$ and $\mathbf{b}$; $\mathbf{a}$ and $\mathbf{b}$ themselves are included in the segment.

◦ Closed plane figures with straight edges are called polygons.

– If segments connecting any two points inside the polygon always lie inside that polygon, it is convex; otherwise, it is concave.

– Triangles are the simplest polygon and the most efficient to represent, process, and visualize.

. Currently, graphics cards can draw hundreds of millions of colored triangles per second.

• In a 3D environment, triangles and any other 3D shape can take three basic coordinate transformations: translation, rotation, and scaling.

◦ A vertex $\mathbf{v}$ is composed of three Cartesian coordinates (v_x, v_y, v_z).

– This set of three coordinates is a 3D vector, an array of three values, which are the vector components.

– Any vertex in 3D space can be represented by its homogeneous coordinates as [x, y, z, 1].

- Vector $\mathbf{v}'$ is at the transformed position of $\mathbf{v}$ after a translation of t_x units over the X axis, t_y units on Y , and t_z units on the Z axis is calculated as follows:

$$\mathbf{v}' = T \cdot \mathbf{v} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix}$$

  – When applied to all the vectors (vertices) that comprise a 3D shape, this transformation will move the shape by the magnitude and direction of vector [t_x, t_y, t_z]

- The scaled version of $\mathbf{v}$ is calculated as follows:

$$\mathbf{v}' = S \cdot \mathbf{v} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot s_x \\ y \cdot s_y \\ z \cdot s_z \\ 1 \end{bmatrix}$$

  – A scaled 3D shape normally changes both its size and its location, and Only shapes centered on the origin will remain in the same position.

  – Scaling may also be done with reference to a point $\mathbf{P}_o = [x_o, y_o, z_o]$, which may be the center of the 3D shape.

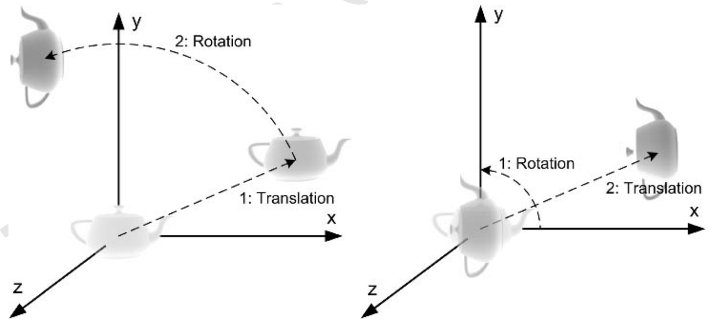
- The three transformation matrices for rotation around the three axes are as follows:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

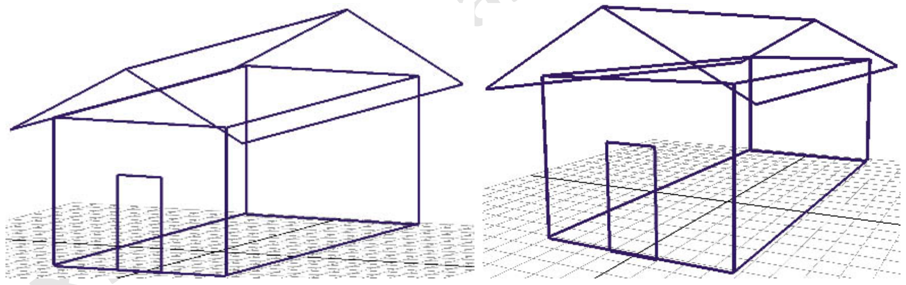
$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Due to non-commutativity of transformations composition, it is critical to understand *the order in which transformation matrices are multiplied*.



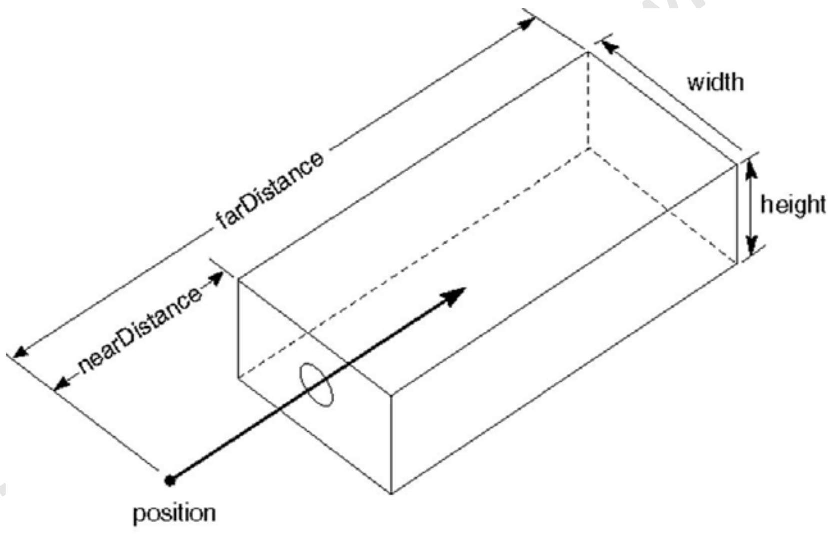
2. Projections

- There are two main types of planar projections: orthographic and perspective.
  - Since computer displays are mostly planar viewing surfaces, the projection surface is usually a plane.



  - Other projection surfaces, such as spheres or cylinders, could also be useful for Virtual Reality applications displayed on hemispherical or cylindrical screens.

- The most commonly used plane of orthographic projection is one that is parallel to the X-Y plane.

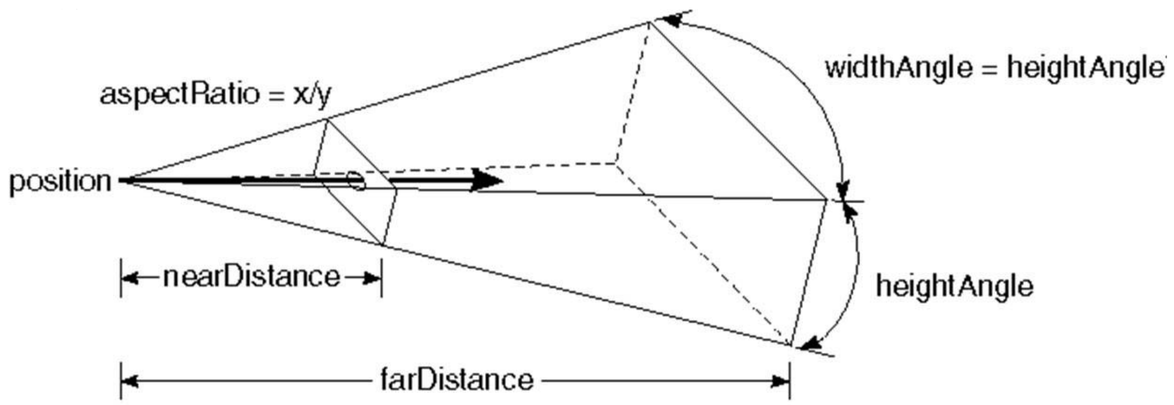


- The transformation matrix with projectors in the Z direction is defined as:

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

- The viewing volume is rectangular formed by the far, near, left, right, top, and bottom clipping planes.
- Anything outside this volume is not drawn.

• Perspective projection works like a pin-hole camera, forcing the light rays to go through one single point.



- The transformation matrix with projectors in the Z direction is defined as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/f & 1 \end{bmatrix}$$

- ◦ In the matrix, f is the distance between the viewpoint and the plane of projection.
- ◦ The projected coordinates on a 2D plane at a distance f from the viewpoint are: $[fx/z,\ fy/z]$.
- ◦ This projection adds the effect that distant objects appear smaller than nearby objects.

3. 3D Modeling

- • Geometric Representations
  - ◦ There are three basic primitives or fundamental ways to create a 3D shape: points, curves, and polygons.
  - ◦ Points are the simplest primitive.
    - – A 3D shape can be composed of a large amount of points to cover the whole 3D surface.

- ◦ All 3D surface can be approximated by polygons,
  - – At the lowest level, any 3D shape should be represented by a set of triangles.
- • Curves
  - ◦ A more practical and compact way of representing 3D shapes is using curves.
    - – Curves can be used to describe complex surfaces.
    - – Curves can also be approximated by lines or points.
  - ◦ Curves could be represented either visually as a list of points or analytically with mathematical definitions.
    - – In visual representation, the higher the number of points used to represent a curve, the more accurate and smooth will be its visualization.

- Analytical representation have the advantages of precision, compact storage, and ease of calculation of intermediate points.

◦ Nonparametric representations are either explicit in the form as $y = f(x)$ or implicit as $f(x, y) = 0$.

- Closed or multiple-value curves, e.g., circles, cannot be represented explicitly.
- Nonparametric representations are axis-dependent, i.e., the choice of coordinate system affects the ease of use.

◦ In parametric form, the coordinates of each point in a curve are represented as functions of a single parameter that varies across a predefined range of values.

- For a 3D curve with t as the parameter, the vector representing the position of a point on the curve is:

$$\mathbf{Q}(t) = [x(t), y(t), z(t)]$$

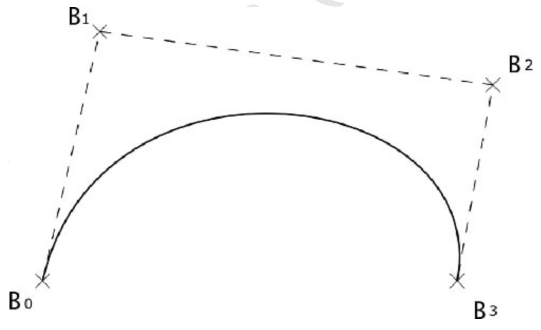
◦ (Cubic) Bézier Curves

- These are one of the most frequently used splines. They are used in imaging, animation, and typesetting.
- They are defined in equation or in matrix notation:

$$\mathbf{Q}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3$$

$$\mathbf{Q}(t) = \mathbf{TBP} = [t^3\ t^2\ t\ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

– Important properties



- The curve generally follows the shape of the polygon formed by the control points.
- The first and last points on the curve are coincident with the first and last control points.
- The curve is contained within the convex hull of the polygon defined by the control points.
- The tangent vectors at the ends of the curve have the same direction as the first and last points of the defining polygon. This is important for achieving continuity.
- The curve can be transformed (translation, scale, rotation) by transforming only its control points.
- *It impossible to locally change within a curve*

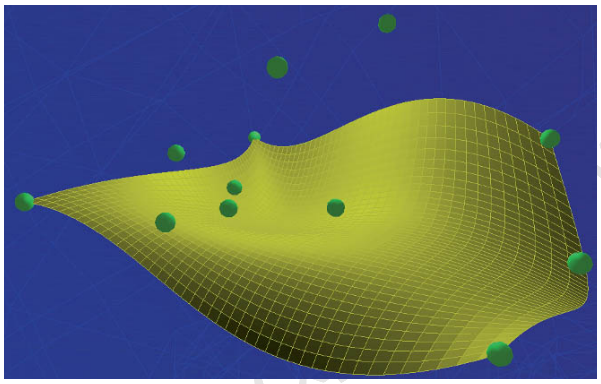
- Surfaces
  - (Cubic) Bézier Patches
    - The matrix specification is:

$$\mathbf{Q}(u,v) = [u^3\ u^2\ u\ 1]\mathbf{BPB}^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

where:

$$\mathbf{B} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \ and\ \mathbf{P} = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix}$$

    - Important properties



. There are 12 controlling points.

. The four interior points specify the internal shape.

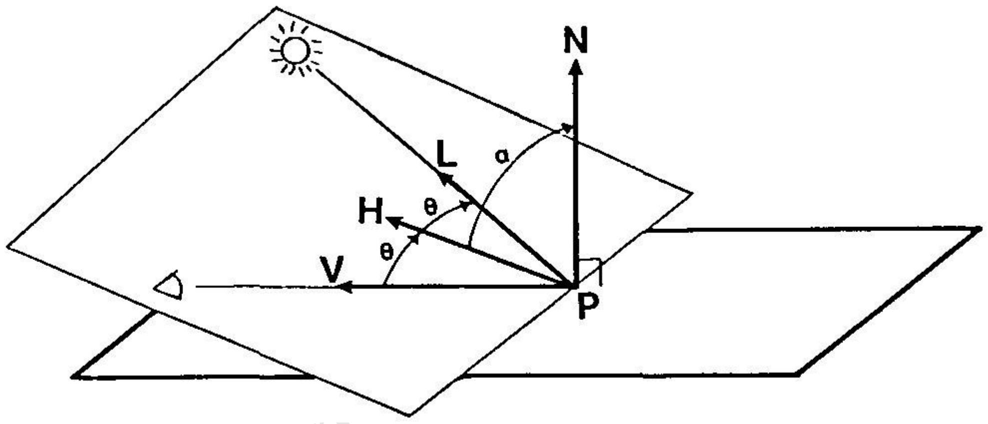
. Only the four corner vertices lie in the surface.

- Other types of (cubic) parametrized surface patches exhibit different properties, and they can be converted from each other.

4. 3D Rendering

- 3D rendering is the process of generating 2D projections of 3D shapes, taking into account the effects of color, light, surface, and other characteristics of the shapes.
- Local Illumination Model
  - Though incomplete from the physics point of view, local illumination produces acceptable results for real-time interaction.
  - The local illumination model considers that the light intensity (I) at one point on a surface has three components: ambient, diffuse, and specular reflections:

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$



- The ambient term considers background illumination of unlit surfaces.
  - It allows for some global control of brightness in a scene:

$$I_{ambient} = I_a k_a$$

where I_a is an ambient illumination constant defined for the entire scene, and $k_a \in [0, 1]$ is an ambient reflection coefficient

- Diffuse reflection is characteristic of light reflected from a dull, matte surface.
  - The reflected light intensity is independent of the position of the viewer and is proportional to the incident light energy I_i.
  - Lambert's cosine law:

$$I_{diffuse} = I_i k_d \cos\theta$$

where k_d is the diffuse reflection coefficient and θ is the angle between the incident light direction and the outside normal to the surface.

  - If we calculate the normalized direction vector from the object surface to the light source **s**, the same equation can be expressed by the dot product[1] of **s** and **n**:

$$I_{diffuse} = I_i k_d (\mathbf{s} \cdot \mathbf{n})$$

- Specular reflection is characteristic of light reflected from a shiny surface, where a bright highlight appears from certain viewing directions.
  - Specular reflection depends on the perfect reflection direction **r**, the viewer direction **v**, and the surface normal **n**:

$$I_{specular} = I_i k_s (\mathbf{r} \cdot \mathbf{v})^n$$

[1] $\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}|\angle(\mathbf{a}, \mathbf{b})$

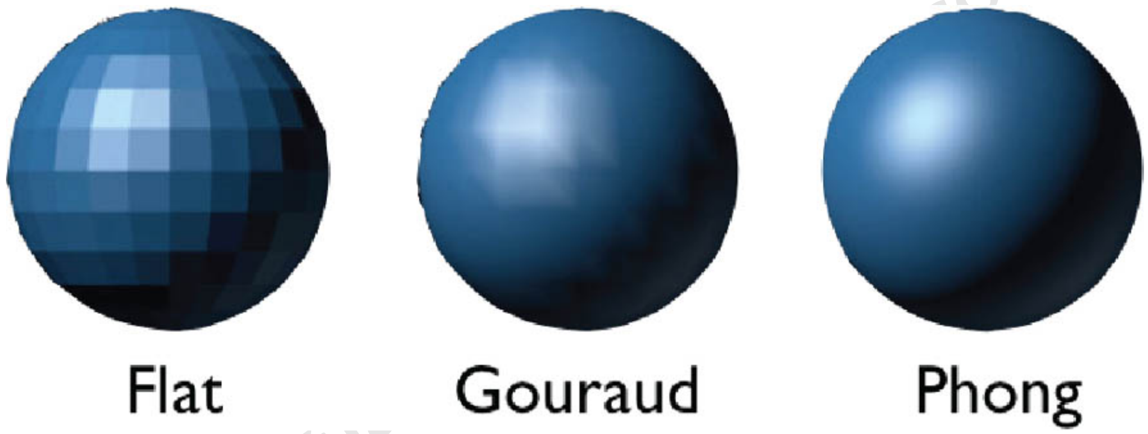
where I_i is the incident light energy, k_s is the specular reflection coefficient, and *n* can be considered a surface roughness parameter.

- Different colors are obtained by specifying different intensity levels for each of the color components in a color model.
  - There are color models, but computer graphics and Virtual Reality applications usually use the RGB model.
    - In the RGB model, display devices generate colors by mixing the three primary colors: red, green, and blue; and The maximum intensities for the three primary colors are normalized to the range [0,1].

- The HSV model uses the concepts of hue, saturation, and brightness to define colors.
- The CIE model specifies colors as the mixture of the three primary colors with different intensities whose sum is 1.
- There are formulas to specify a given color using any of the three color models.

– Producing a colored image requires calculating the light intensity (ambient + diffuse + specular reflections) three times, once for each color component.
  - Surfaces reflect the three color components with different reflection coefficients.

○ Shading

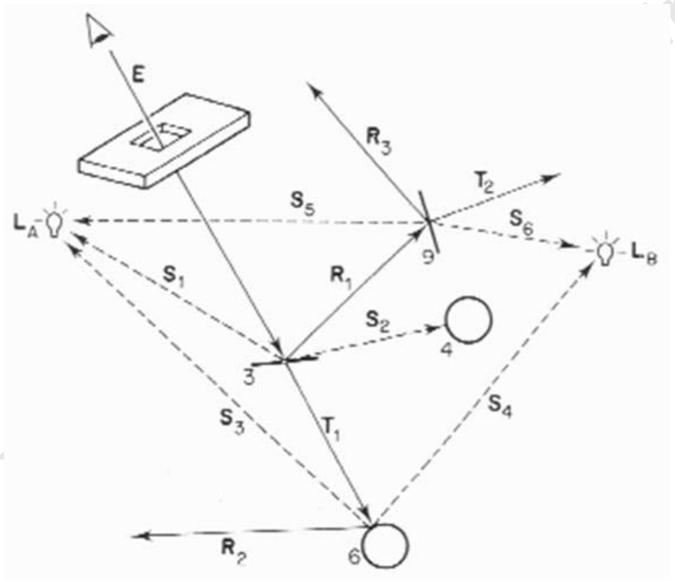


– Flat-shading uses light intensity on a per-polygon basis.
  - The discontinuous changes in light intensity produces the mach band effect.

– Gouraud shading uses intensity interpolation.
  - Light intensities are calculated for each vertex of the polygon.
  - Linear interpolation is used to determine the color at each point inside the polygon.

– Rather than the interpolation of light intensities, Phong shading uses surface normal interpolation.
  - At each point on the polygon, the normal is calculated as an interpolation of the normal vectors at the vertices of the polygon.
  - Each vector is then normalized and for each point, and an illumination model is used to determine the light intensity.

- Global Illumination Model
  - The global illumination model has two components:



- *Direct illumination*: light arriving directly from sources such as lamps or the sun.
- *Indirect illumination*: effects of light reflecting off surfaces in the environment.
- A light ray may reach the surface indirectly via reflection in other surfaces, transmission through partially transparent objects, or a combination of both. The result is the most realistic images and extremely high processing overheads.

○ The global illumination model states that the transport light intensity from one surface point to another is the sum of the emitted light and the total intensity scattered toward x from all other surface points.

$$L_o(x, \mathbf{w}) = L_e(x, \mathbf{w}) + \int_\Omega f_r(x, \mathbf{w}', \mathbf{w}) L_i(x, \mathbf{w}')(\mathbf{w}' \cdot \mathbf{n}) d\mathbf{w}'$$

– $L_o(x, \mathbf{w})$ is light leaving point *x* in direction **w**.

– $L_e(x, \mathbf{w})$ is the light emitted, a property of the surface at point *x*.

– $\int_\Omega \cdots d\mathbf{w}'$ is an infinitesimal sum over a hemisphere of inward directions.

- $f_r(x, \mathbf{w}', \mathbf{w})$ is the bidirectional reflectance distribution function, the proportion of light reflected at point x.
- $L_i(x, \mathbf{w}')$ is the incident light on point *x* from direction $\mathbf{w}'$.
- $(\mathbf{w}' \cdot \mathbf{n})$ is light attenuation due to incident angle.

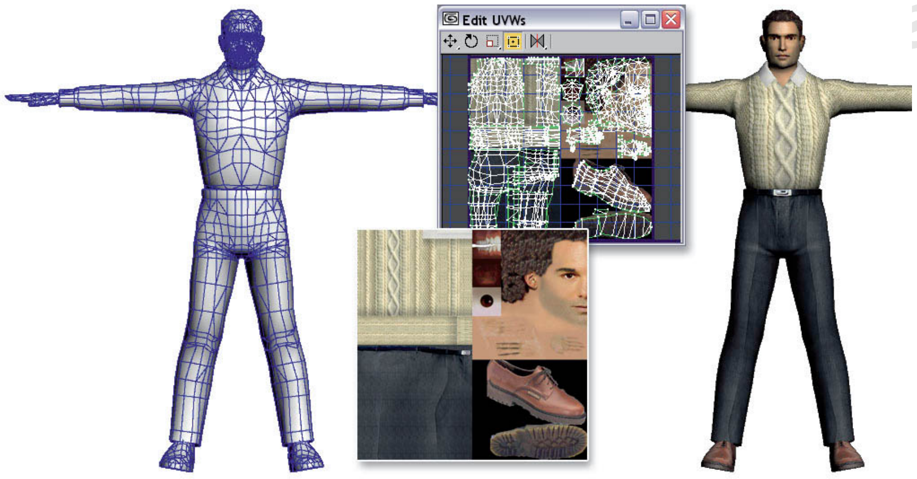
∘ Ray tracing uses the following principle to simulate the trajectories of light rays emitted by light sources and reflected by objects in the scene until some of them reach the eye or the camera.

– Only rays arriving at the eye are simulated.
– The algorithm checks the intersections between rays from the viewpoint and objects in the scene.
– A recursive approach can be applied to simulate reflections: rays intersecting an object generate reflected rays, which are then checked for intersections with other objects.
– At each intersection the rays contribute a color intensity, the sum of the contributions defines a color for each pixel on the screen.

◦ The basic principle of radiosity is the energy balance in a closed scene: the radiosity (energy per unit surface) of a surface depends on its self-emitted radios-

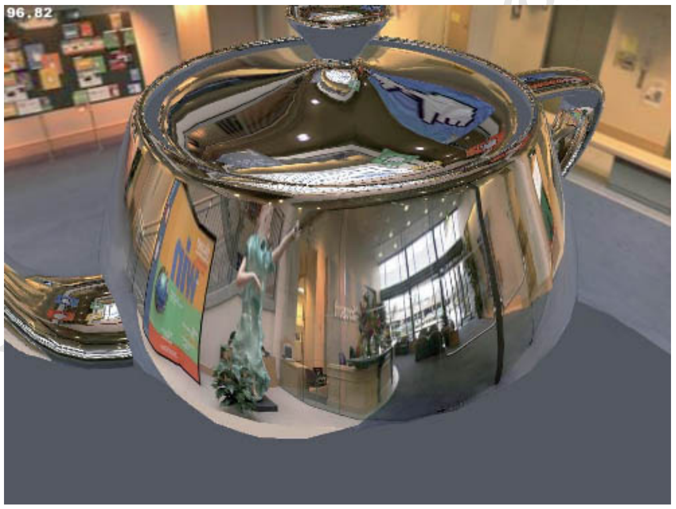
ity and on the energy (light) emitted by all the other surfaces, received and reflected by the surface.

- Textures

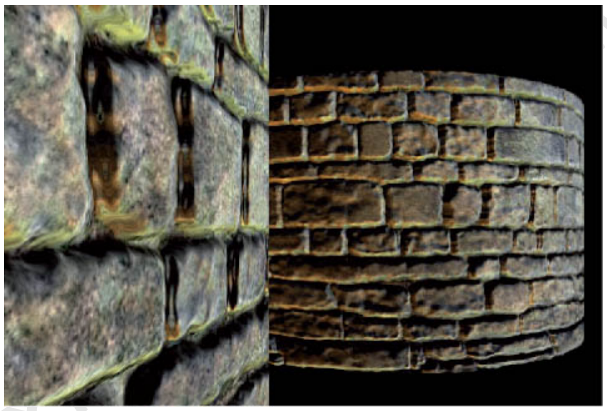


  - Texture mapping improves visual appearance by mapping a 2D image (texture) to a 3D shape, in such a way that the 3D surface gets colored by the image.

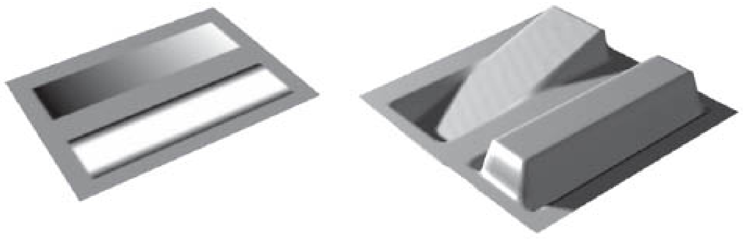
- Other rendering algorithms
  - Environment mapping is a view-dependent texture mapping technique that supplies a specular reflection to the surface of objects.



– Bump mapping produces an effect of 3D relief or wrinkles on a surface by modifying parameters of the shading equation such as the surface normal.



– Displacement mapping introduces a parametrization of the surface defined by the texture coordinates to modify the shading.



◦ Texture mapping, including environment mapping and bump mapping, are implemented in hardware in current graphics cards and supported by the most common 3D graphics APIs.

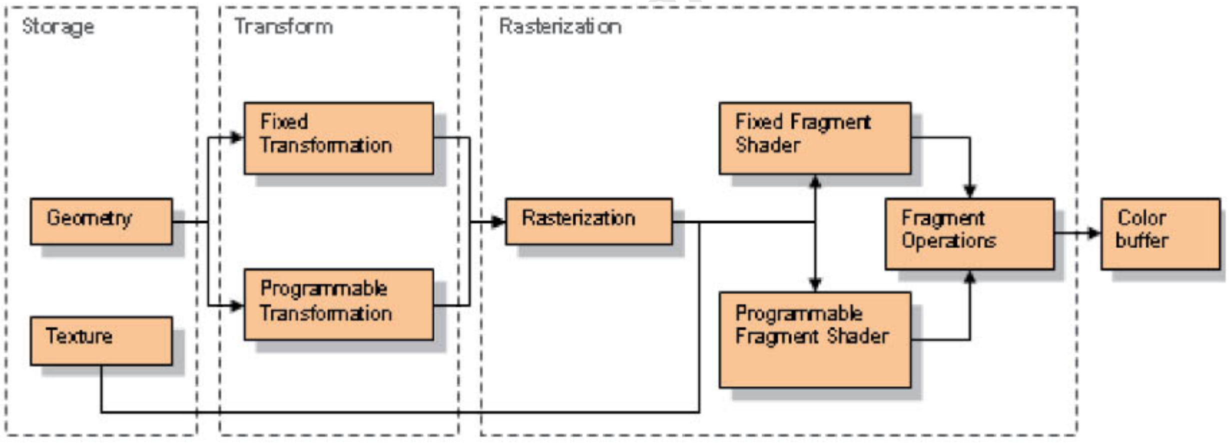
• Rendering Pipeline

◦ Rendering pipeline is the process that standard computer graphics APIs process data to render 3D images.

– It is usually implemented with graphics hardware.

– Computer software can also emulate the data processing, but in slow performance.

◦ A simplified version of the rendering pipeline according to the OpenGL specification:


Storage
Transform
Rasterization
Fixed Transformation
Geometry
Texture
Programmable Transformation
Rasterization
Fixed Fragment Shader
Programmable Fragment Shader
Fragment Operations
Color buffer