CS 563 Advanced Topics in Computer Graphics

Transforming Objects – Chapter 21

by Wadii Bellamine

Outline

- Review of Affine Transformations
- Intersecting Transformed objects
- Instancing
- Beveled Objects

Review of Affine Transformations

- 3D Homogenous matrix transformations

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \boldsymbol{p}$$

$$\boldsymbol{T} \qquad \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x + d_x \\ y + d_y \\ z + d_z \\ 1 \end{bmatrix} \qquad \boldsymbol{p' = Tp}$$

Review of Affine Transformations

- Translation:

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T(d_x, d_y, d_z)[x\ y\ z\ 1]^T = [x + d_x \quad y + d_y \quad z + d_z \quad 1]^T$$

Review of Affine Transformations

- Scaling:

$$S(a,b,c) = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S(a,b,c)[x\ y\ z\ 1]^T = [ax\ \ by\ \ cz\ \ 1]^T$$

Review of Affine Transformations

- Rotation:

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

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

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

(Suffern,406)

Review of Affine Transformations

- Inverse transformations:

Translation.

$$T^{-1}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Scaling.

$$S^{-1}(a, b, c) = \begin{bmatrix} 1/a & 0 & 0 & 0 \\ 0 & 1/b & 0 & 0 \\ 0 & 0 & 1/c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Rotation.

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

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

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

(Suffern, 412)

Intersecting Transformed Objects

- Problem Setup:
  - We have a transformed object or primitive, and want to calculate the hit point(s) with the ray, and the normal to the object at that hit point.
  - Problem: How exactly can we do this without complicated arithmetic?
  - Solution: Transform the ray instead of the object!

(so that we can take advantage of a simple closed form solution)

Intersecting Transformed Objects

- Five steps:

$$T*\text{Object} = \text{Object}'$$

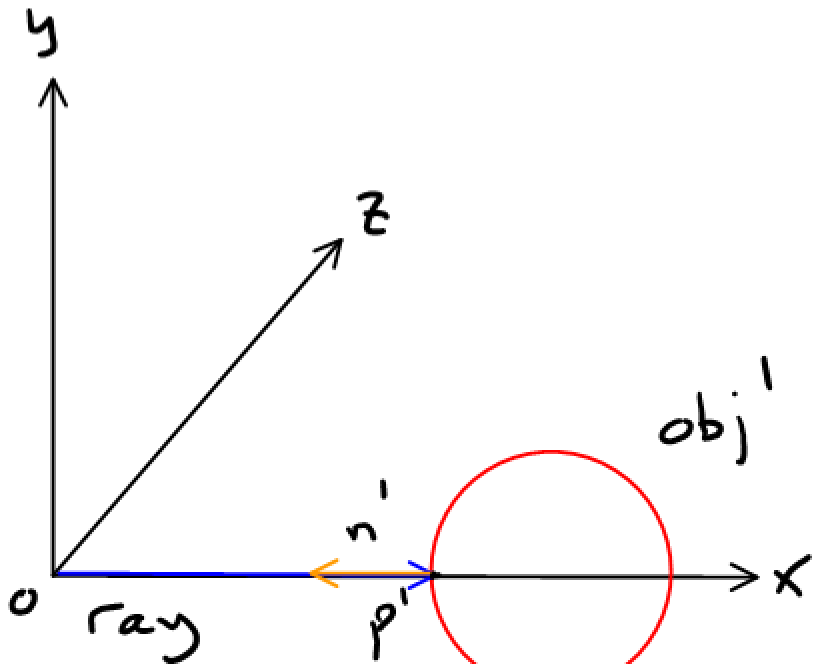
$$T^{-1}*\text{ray} = \text{ray}'$$

(=> means result)

1. Compute inverse transformation matrix of the transformed object, and apply this to the ray, to obtained an inverse transformed ray => **ray’**
2. Compute hit point of **ray’** and untransformed object **Object** => **p**
3. Compute normal to object at **p** => **n**
4. Apply **T** to **p** to obtain hit point of ray and **Object’** => **p’**
5. Apply **T** to **n** to obtain normal to **Object’** => **n’**

Intersecting Transformed Objects

- Simple illustration of procedure: Sphere

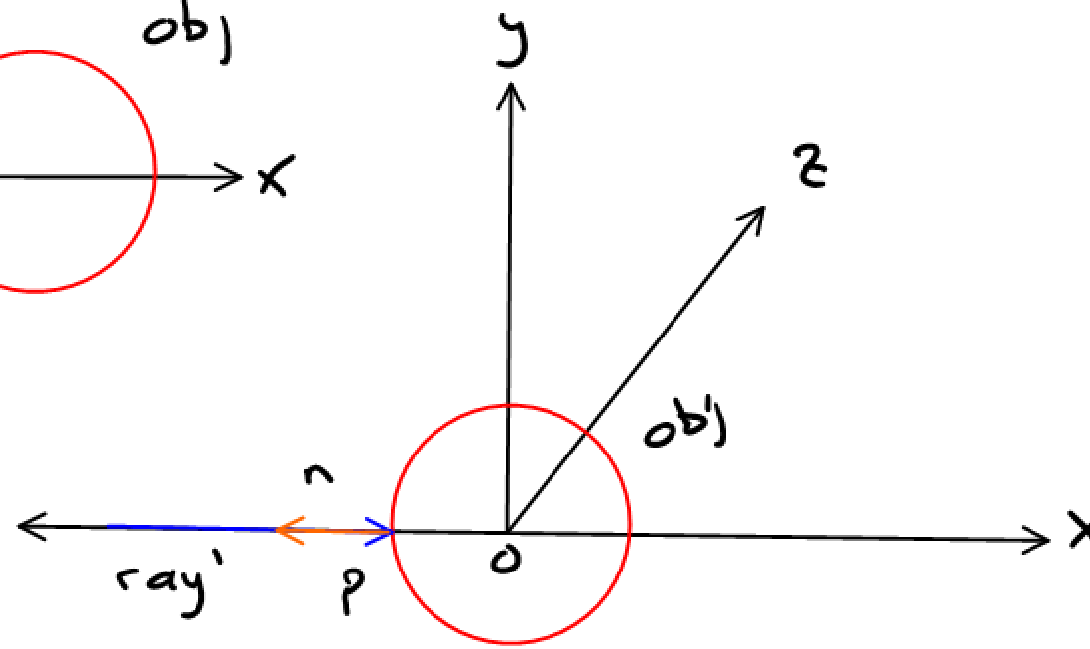


$$p' = Tp$$

$$n' = T^{T} n$$

$$p = T^{-1} p'$$

$$n = T^{-T} n'$$



Intersecting Transformed Objects – finding p'

- Computing ray': (blue = unknown)

$$ray \Rightarrow p' = o + td$$

$$p' = Tp$$

$$p = T^{-1}p'$$

$$p = T^{-1}p' = T^{-1}o + tT^{-1}d$$

$$T^{-1}o = o'$$

$$T^{-1}d = d'$$

$$p = o' + td' \Rightarrow ray'$$

Intersecting Transformed Objects – finding p'

- Computing t: Simple sphere example
  - Unit sphere centered at origin:

$$x^2 + y^2 + z^2 = 1$$

  - Equation of ray':

$$\boldsymbol{p} = \boldsymbol{o}' + t\boldsymbol{d}' = \begin{bmatrix} o_x + td_x \\ o_y + td_y \\ o_z + td_z \end{bmatrix}$$

  - *Combining the two equations, solve for t:*

$$(o_x + td_x)^2 + \left(o_y + td_y\right)^2 + \left(o_y + td_y\right)^2 = 1$$

Intersecting Transformed Objects – finding p'

- Computing **o'** and **d'**:

$$o' = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} o_x \\ o_y \\ o_z \\ 1 \end{bmatrix}.$$

$$\begin{aligned} o'_x &= m_{00}o_x + m_{01}o_y + m_{02}o_z + m_{03}, \\ o'_y &= m_{10}o_x + m_{11}o_y + m_{12}o_z + m_{13}, \\ o'_z &= m_{20}o_x + m_{21}o_y + m_{22}o_z + m_{23}. \end{aligned}$$

$$d' = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \\ 0 \end{bmatrix}.$$

$$\begin{aligned} d'_x &= m_{00}d_x + m_{01}d_y + m_{02}d_z, \\ d'_y &= m_{10}d_x + m_{11}d_y + m_{12}d_z, \\ d'_z &= m_{20}d_x + m_{21}d_y + m_{22}d_z. \end{aligned}$$

(Suffern, 420)

Intersecting Transformed Objects – finding p’

- Computing ***p’*** :

*“if the closest hit point **p** of the inverse transformed ray with the untransformed object occurs at $t=t_0$, the closest hit point **p’** of the original ray with the transformed object occurs at the same value of t: $t=t_0$ “ (Suffern, 421)*

Therefore:

$$\boldsymbol{p'} = \boldsymbol{o} + t\boldsymbol{d}$$

Intersecting Transformed Objects – finding n'

- Computing n':
  - First find ***n***, the normal to the untransformed object at point ***p**:*
    - *For a unit sphere, this is simply the vector from the origin to the hitpoint:* ***n=p – o***
  - Apply the transpose of the inverse transform to n:

n' = T

$$\boldsymbol{n}' = T^{-T}\boldsymbol{n}$$

$$\boldsymbol{n} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix} = \begin{bmatrix} m_{00} & m_{10} & m_{20} & 0 \\ m_{01} & m_{11} & m_{21} & 0 \\ m_{02} & m_{12} & m_{22} & 0 \\ m_{03} & m_{13} & m_{23} & 1 \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix}.$$

$$\begin{aligned} n'_x &= m_{00}n_x + m_{10}n_y + m_{20}n_z, \\ n'_y &= m_{01}n_x + m_{11}n_y + m_{21}n_z, \\ n'_z &= m_{02}n_x + m_{12}n_y + m_{22}n_z. \end{aligned}$$

(Suffern,424)

Instancing

- Instead of creating a new object every time we want to show a different transformation of the same object, we create a pointer to that object:
  - The instance class implements instancing by:
    - Having a pointer to the object
    - Storing a the forward and inverse transformation matrices, and materials of that instance.
  - Each instance transforms its own local copy of the ray.
  - Every time the instance is transformed by a new transform T, its forward matrix (which defaults to the unit matrix) is multiplied by T, and its inverse matrix is multiplied by T^{-1}

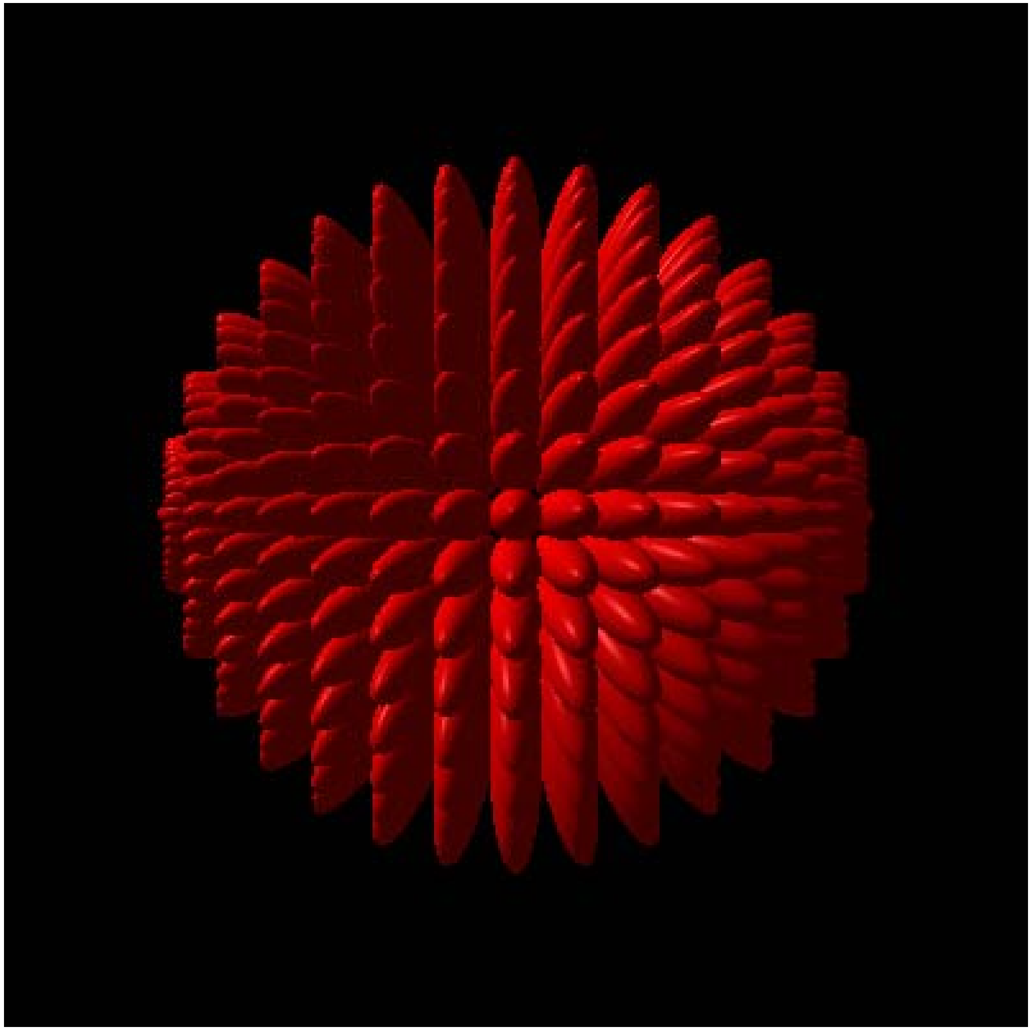
Instancing

- Instances can be nested:
  - An instance can point to another instance:
    - The hit function is called recursively until the object is untransformed.
  - This makes it easier to implement compound objects.
    - Q: Why?
    - A: Parts of the object can be transformed relative to a single part of the object. The entire object is then transformed relative to world coordinates.
    - A: Nested instances can partly share material properties.

Instancing

- Advantages of instancing:
  - Q: Can you guess some?
  - A: Less storage needed: we only store a pointer to the object, its transform matrices, and material properties, for each instance vs. storing a matrix (16 floats).

Instancing



256 instances of sphere

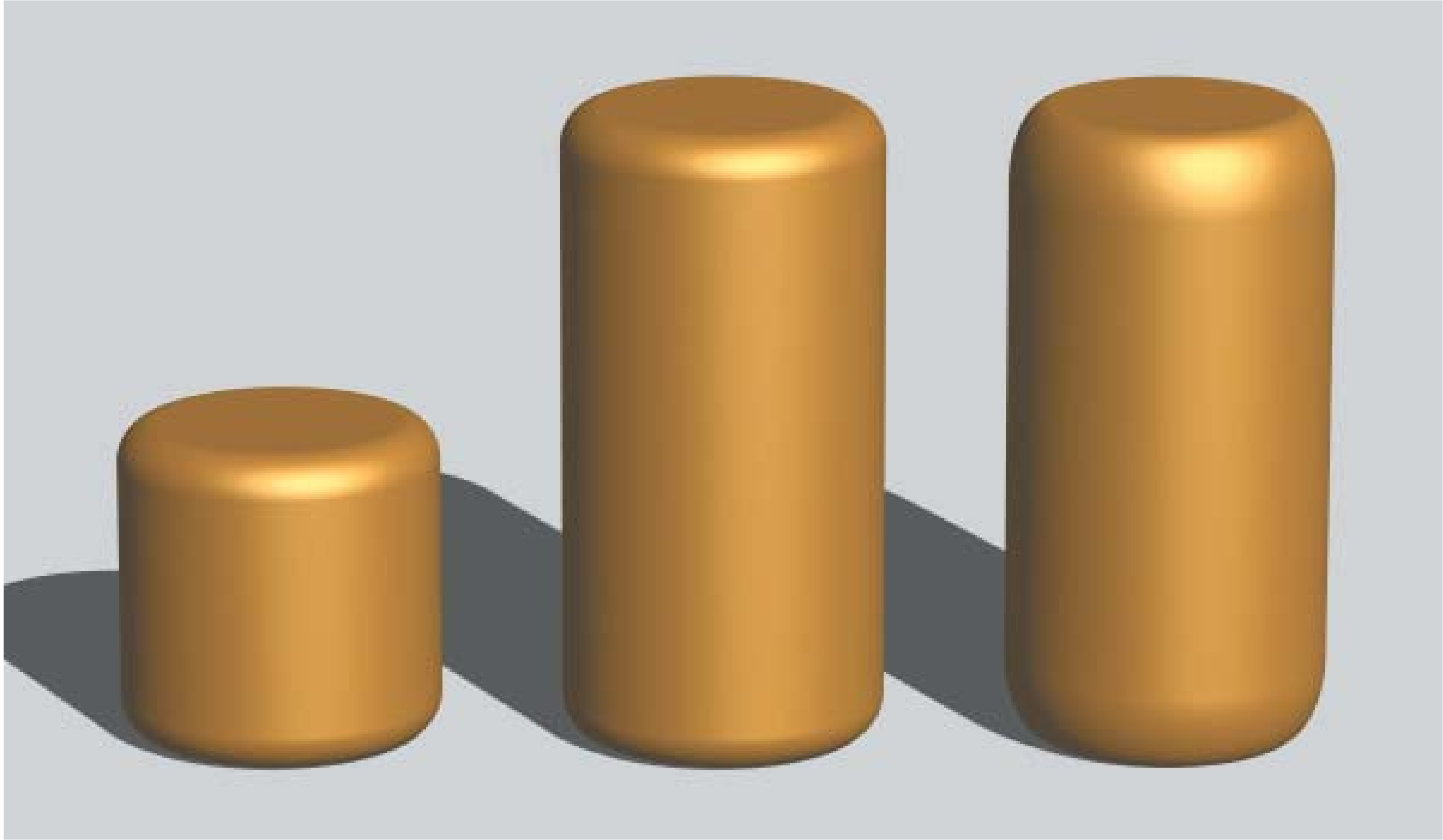
Instancing

- Code for previous image:

```
int n = 16;

    for(int i=0;i<n;i++)
    {
          for(int j=0;j<n;j++)
          {
                    Instance* ellipsoid_ptr = new Instance(new Sphere);
                    ellipsoid_ptr->set_material(phong_ptr);
                    ellipsoid_ptr->scale(1, 4, 1);
                    ellipsoid_ptr->translate(0, 10, 0);
                    ellipsoid_ptr->rotate_z((360.0f/(float)n)*(float)i);
                    ellipsoid_ptr->rotate_x((360.0f/(float)n)*(float)j);
                    add_object(ellipsoid_ptr);
          }
    }
```

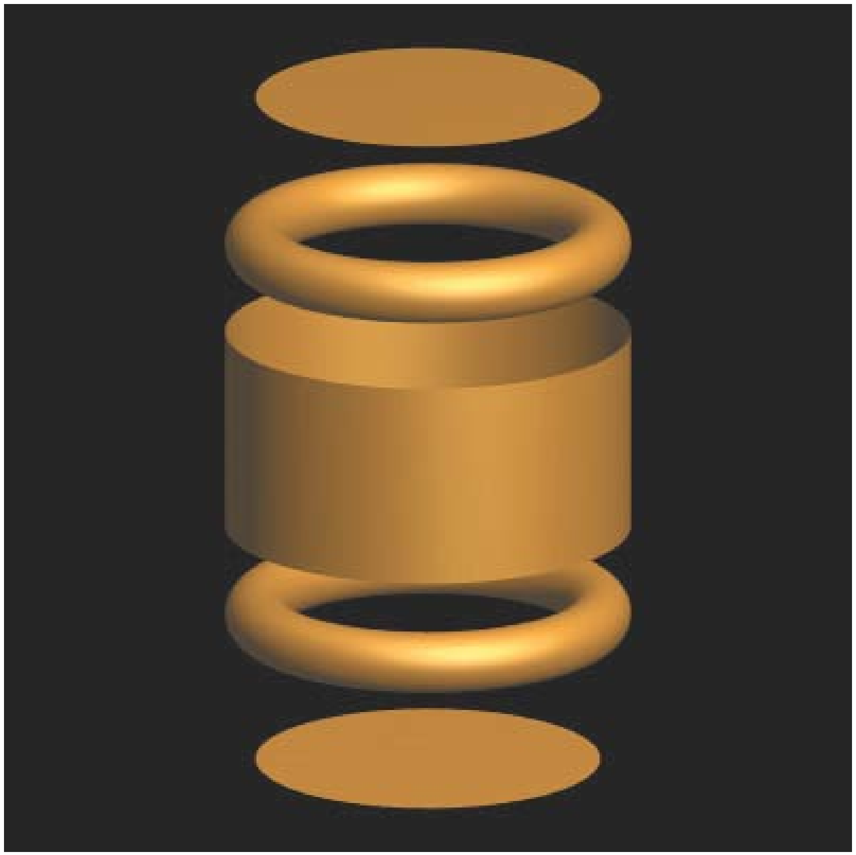
Beveled Objects

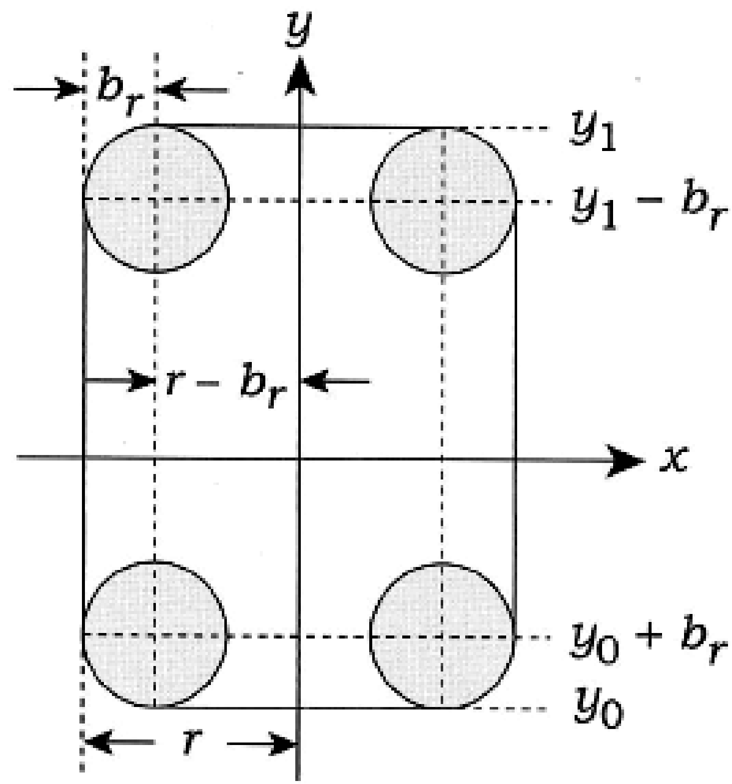


(Suffern, 437)

Beveled Objects

- Compose transforms of primitive objects:

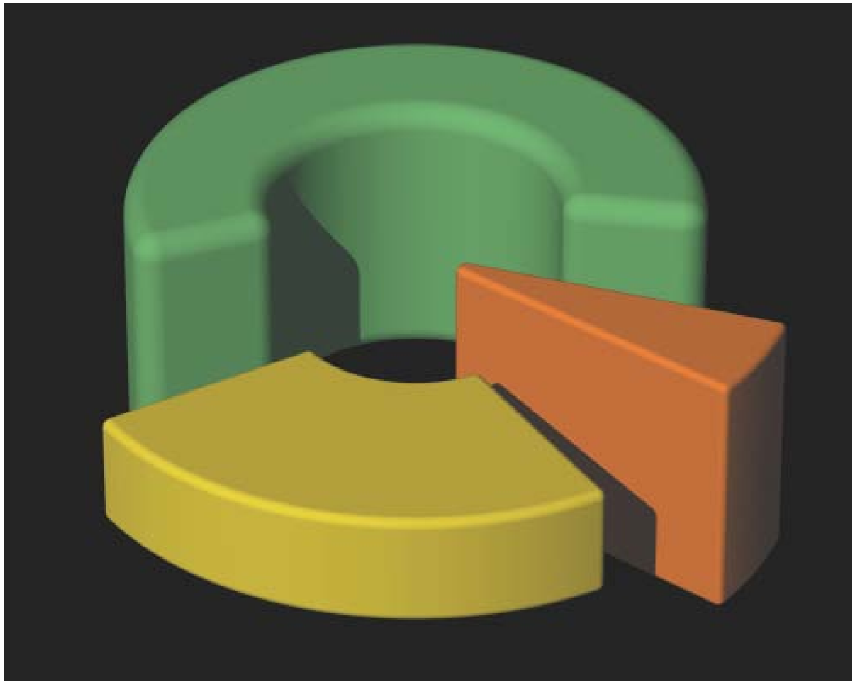


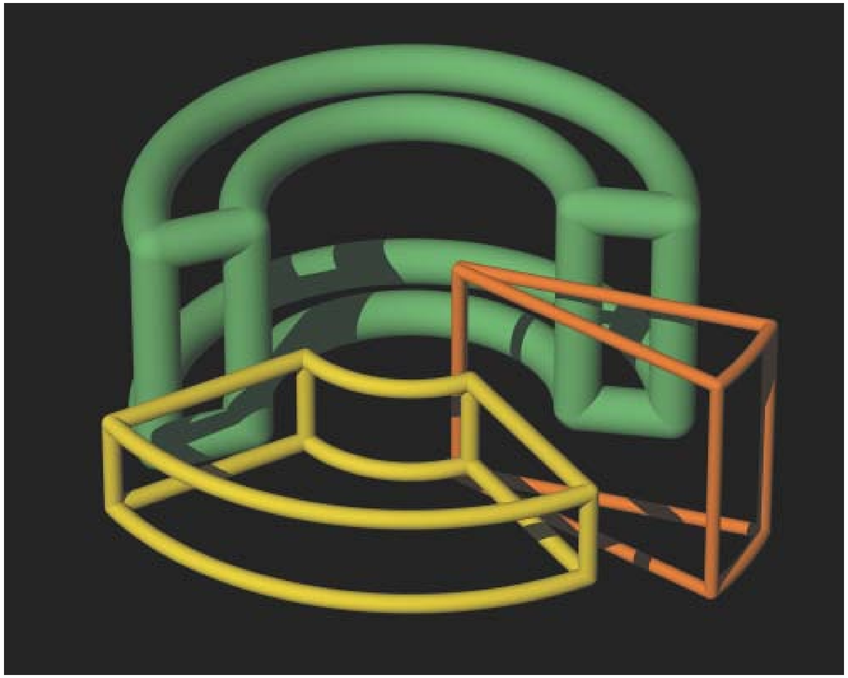


(Suffern,433)

Beveled Objects

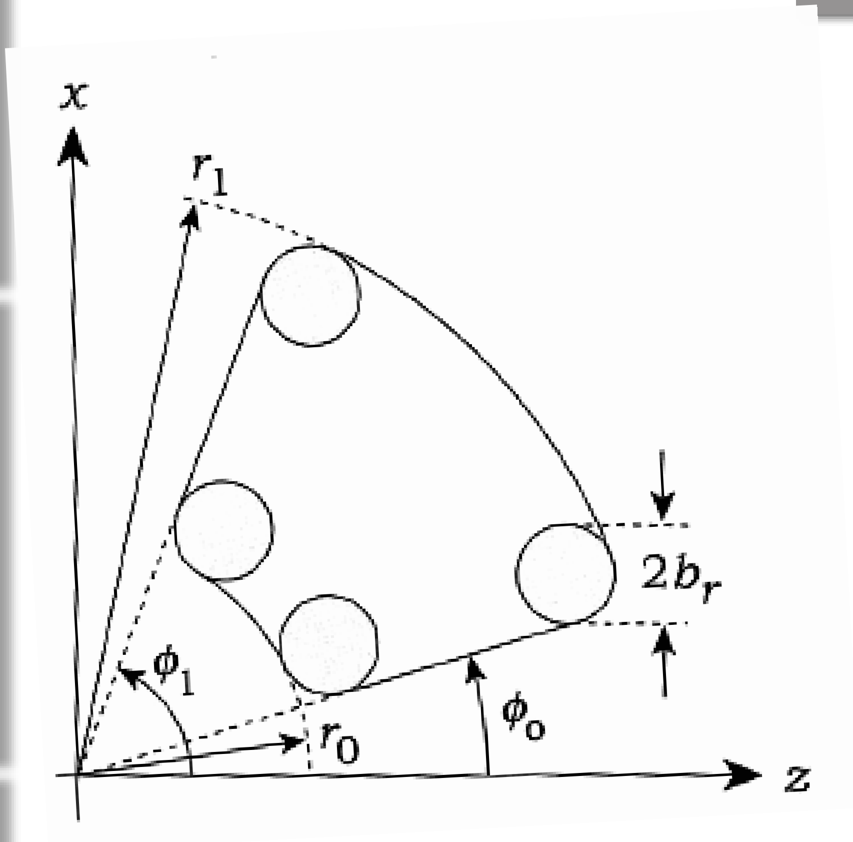
- Beveled wedges:





(Suffern, 434)

Beveled Objects



- Parameters:
  - y extents (not shown):
    - y_0 and y_1
  - Inner and outer radii:
    - r_0 and r_1
  - Min and max azimuth:
    - ϕ_0 and ϕ_1
  - Bevel radius:
    - b_r

- Restrictions:
  - $r_1 > r_0 > 0$ with min separations
  - $y_1 > y_0$
  - $0° < \phi_0 < \phi_1 < 360°$ with min separations

Beveled Objects

- Igloo composed with beveled objects:

http://www.andynicholas.com/thezone/content/download/ambicolor/Igloo_ambi.jpg

References

- Suffern, Kevin (2007). Ray Tracing from the Ground Up.pp. 405-434 Wellesley, MA: A K Peters, Ltd.