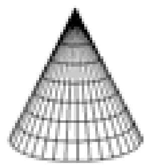
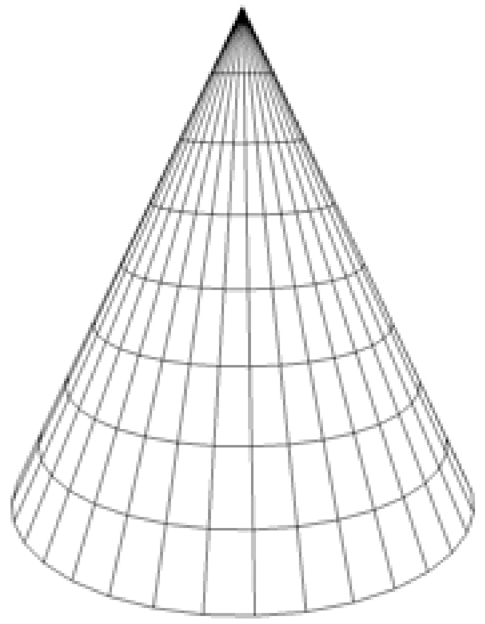



Monash University • Clayton's School of Information Technology

CSE3313 Computer Graphics

Lecture 15: Transformations in 3D

Homogeneous Transformations in 3D

- 2D homogeneous coordinates can be generalised to 3D in a straightforward way.

$$\mathbf{p} = \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

- Likewise 2D transformations generalise conveniently in 3D.

  Translation:

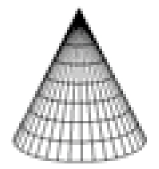
$$x' = x + \Delta x$$
$$y' = y + \Delta y$$
$$z' = z + \Delta z$$
$$w' = w$$

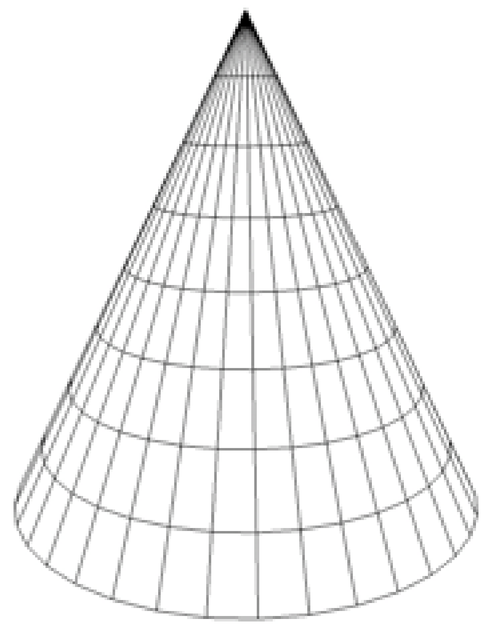
$$\mathbf{p}' = \mathbf{T}(\Delta x, \Delta y, \Delta z)\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{p}$$

Scaling

$$x' = \alpha x$$
$$y' = \beta y$$
$$z' = \gamma z$$
$$w' = w$$

$$\mathbf{p}' = \mathbf{S}(\alpha, \beta, \gamma)\mathbf{p} = \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p}$$





Rotation

- Rotation about the z-axis is the same as the 2D case as it leaves the z coordinate unchanged.

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$
$$w' = w$$

$$\mathbf{p}' = \mathbf{R}_z(\theta)\mathbf{p} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{p}$$

Rotation about the x axis:

$$\mathbf{p}' = \mathbf{R}_x(\theta)\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{p}$$

Rotation about the y axis:

$$\mathbf{p}' = \mathbf{R}_y(\theta)\mathbf{p} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{p}$$

Inverse Transformations

- Inverse transformations are also straightforward:

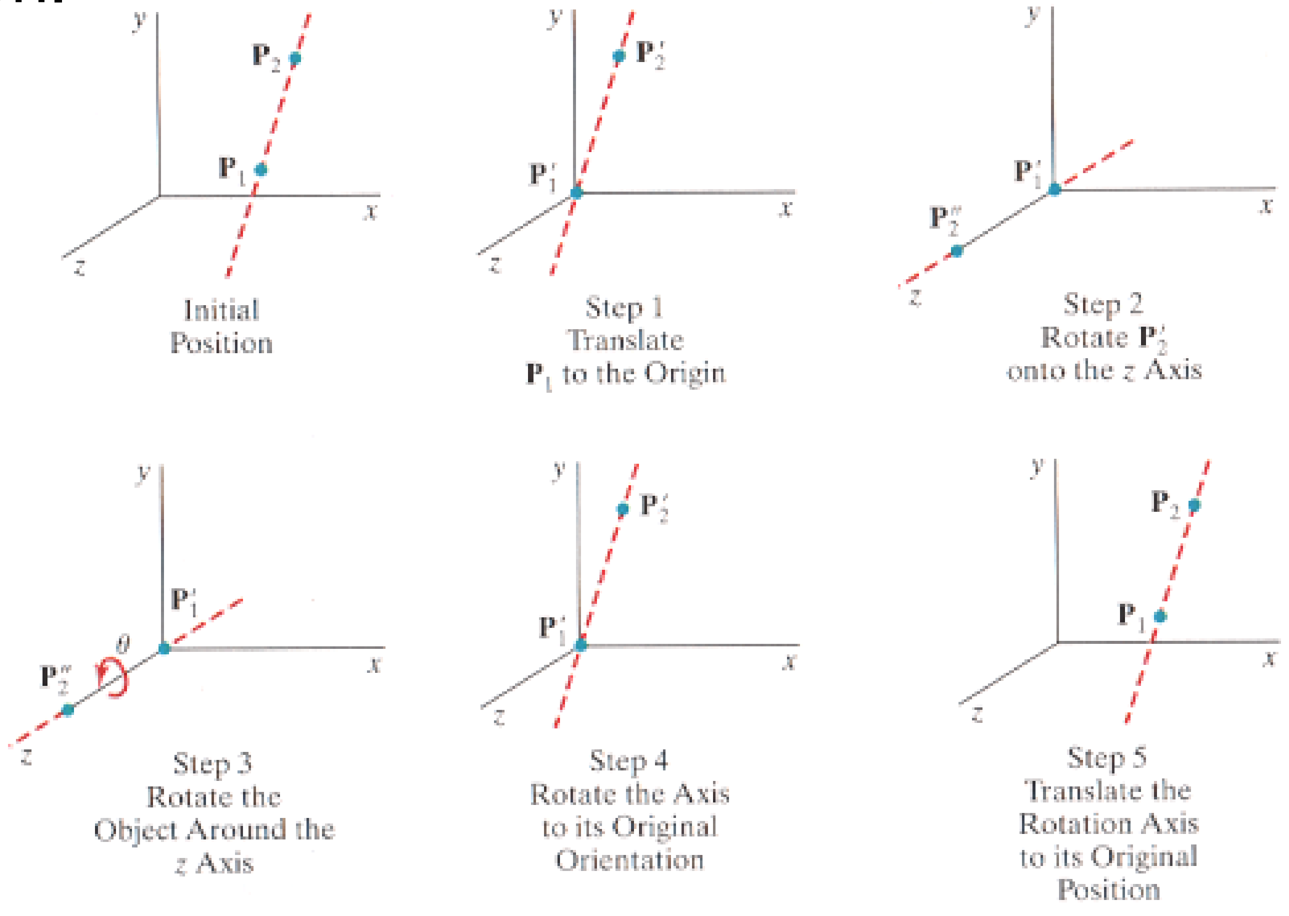
$$\mathbf{T}^{-1}(\Delta x, \Delta y, \Delta z) = \mathbf{T}(-\Delta x, -\Delta y, -\Delta z)$$

$$\mathbf{S}^{-1}(\alpha, \beta, \gamma) = \mathbf{S}(\frac{1}{\alpha}, \frac{1}{\beta}, \frac{1}{\gamma})$$

$$\mathbf{R}_x^{-1}(\theta) = \mathbf{R}_x(-\theta) = \mathbf{R}_x^T(\theta)$$

General Three-dimensional Rotations

- It is often convenient to specify a rotation about an arbitrary axis rather than one of the three coordinate axes (OpenGL rotates about an arbitrary vector).
- This can be done as a compound transformation where the axis is transformed to the origin, then rotated onto the *z* axis, the rotation performed, then the vector rotated and translated back to its original position.



y
x
z
P_2
P_1
Initial Position
P_2'
P_1'
Step 1 Translate P_1 to the Origin
P_1'
P_2''
Step 2 Rotate P_2' onto the z Axis
θ
P_1'
P_2''
Step 3 Rotate the Object Around the z Axis
P_2'
P_1'
Step 4 Rotate the Axis to its Original Orientation
P_2
P_1
Step 5 Translate the Rotation Axis to its Original Position

FIGURE 5-42 Five transformation steps for obtaining a composite matrix for rotation about an arbitrary axis, with the rotation axis projected onto the z axis.

General Three-dimensional Rotations (cont.)

- Composite matrix R for a rotation α around and axis specified by two points $p_1=(x_1,y_1,z_1)$ and $p_2=(x_2,y_2,z_2)$. Construct R as follows:

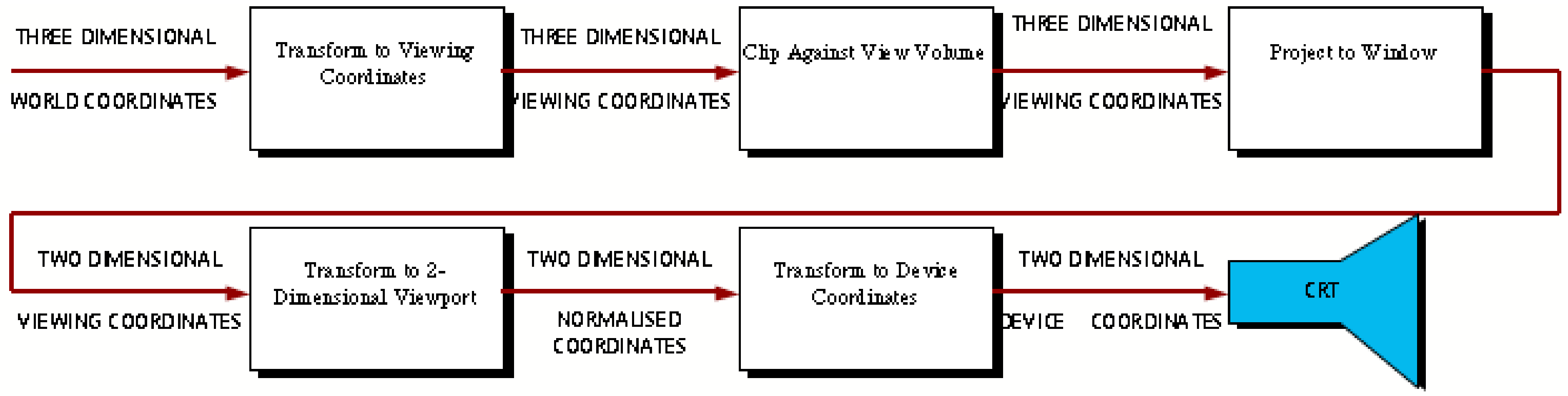
1. Translate so that p_1 is at the origin: $T(-p_1)$
2. Let $Q = (x,y,z) = p2 - p1$ be the other end of the line segment, anc convert this to spherical form (r, ϕ , θ)
3. Apply the rotation $-\theta$ about the *z* axis to bring Q on to the *zx* plane: $R_z(-\theta)$
4. Apply the rotation $-\phi$ about the *y* axis so that Q is coincident with the *z* axis: $R_y(-\phi)$

Let $M = R_y(-\phi) R_z(-\theta) T(-p_1)$, $M^{-1} = T(p_1) R_z(\theta) R_y(\phi)$. The complete transformation is:

$R = M^{-1} R_z(-\alpha) M$

3D Viewing

- The logical sequence of operations in 3D viewing is described in the diagram below:



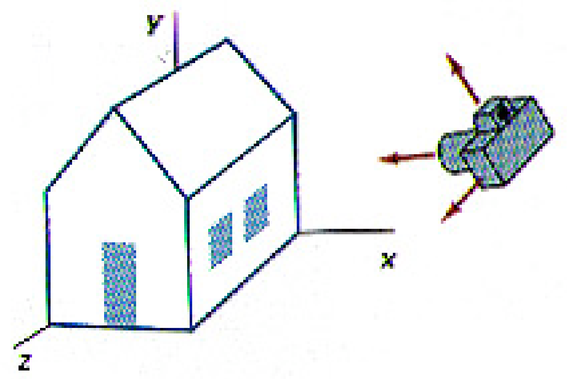
The scene is described in 3D **world coordinates.**

While the scene being viewed may be static, what is seen by an observer might change if the observer changes her position.

Viewing coordinates are a coordinate system based on an observer or synthetic camera with a location and orientation.

3D Viewing

- For example we might have a model of buildings in a street. The buildings are described in world coordinates and their coordinates do not change.
- We might be interested in presenting a series of views of an observer as she walks down the street. The position of the observer and hence, the positions of the buildings with respect to the observer (view coordinate system) will change.
- The scene to be viewed is clipped against a 3D **view volume** and **projected** into the window area defined on the view plane.
- Only those parts of the scene which project into the window on the view plane will actually be visible.
- Similarly, only those parts of the scene that are within the view volume are potentially visible.



Clipping and Hidden Surface Removal

- Clipping of 3D information could be done against the view volume while clipping of 2D information could be carried out once the scene had been projected onto the view plane.
- Clipping of 3D information is carried out because it occurs earlier in the process and reduces the amount of work that needs to be done in the latter stages.
- Removal of hidden surfaces can be regarded as a kind of clipping.
- Hidden surface removal algorithms which operate on 3D coordinate information are said to operate in **model space**, while those that operate on coordinate information after it has been projected onto the projection plane are said to operate in **image space**.
- There is no single best hidden surface removal algorithm.

Projection and Display

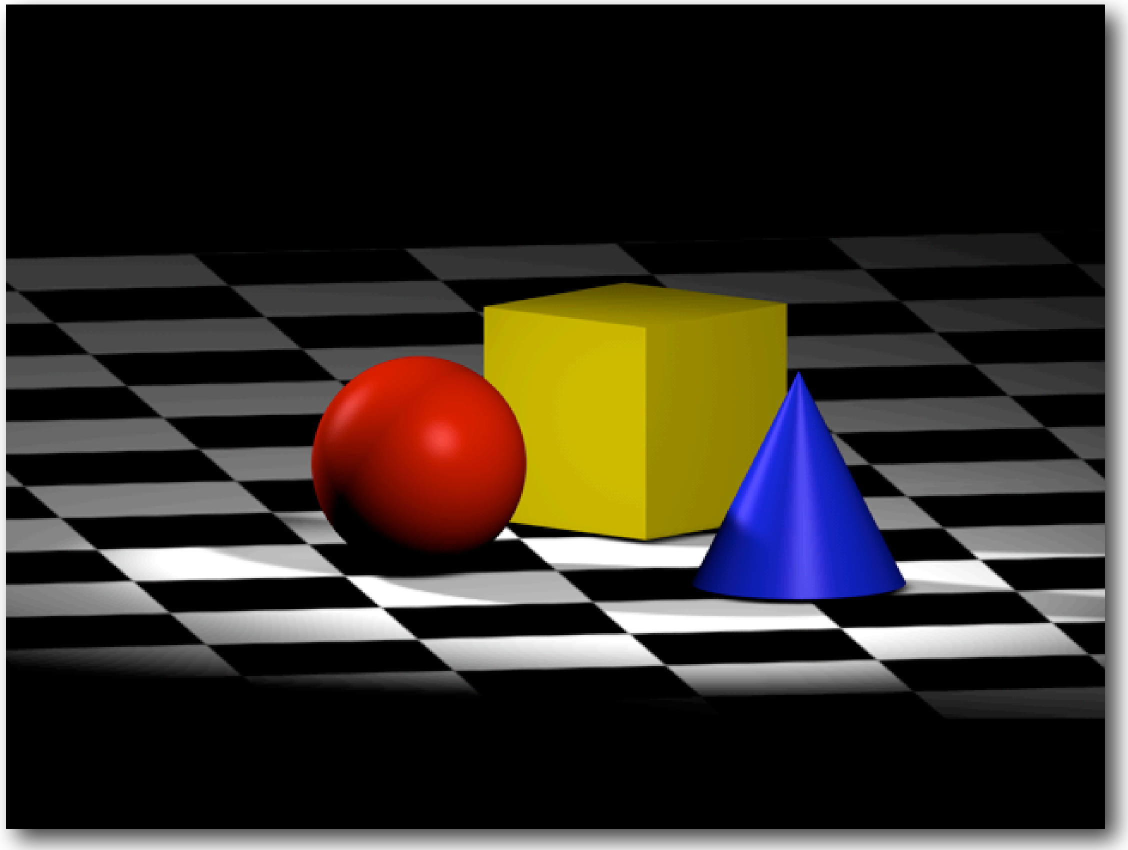
- Once the 3D information has been projected onto the view plane, the remaining stages of processing are the same as for 2D viewing (with the possible exception of hidden surface removal).
- 3D scenes can be represented by straight lines which define the boundaries of planar regions.
- Since the amount of data that has to be processed for 3D scenes can be large, in some cases only the boundaries of the polygons which make up the scene are displayed — **wire framing**.
- In addition to transformation, projection, and hidden surface removal, we may simulate light reflecting properties by performing **shading** over the surface of visible polygons.
- To enhance realism we can also compute **shadows** that may be cast, even simulate complex texturing using **texture mapping** techniques.

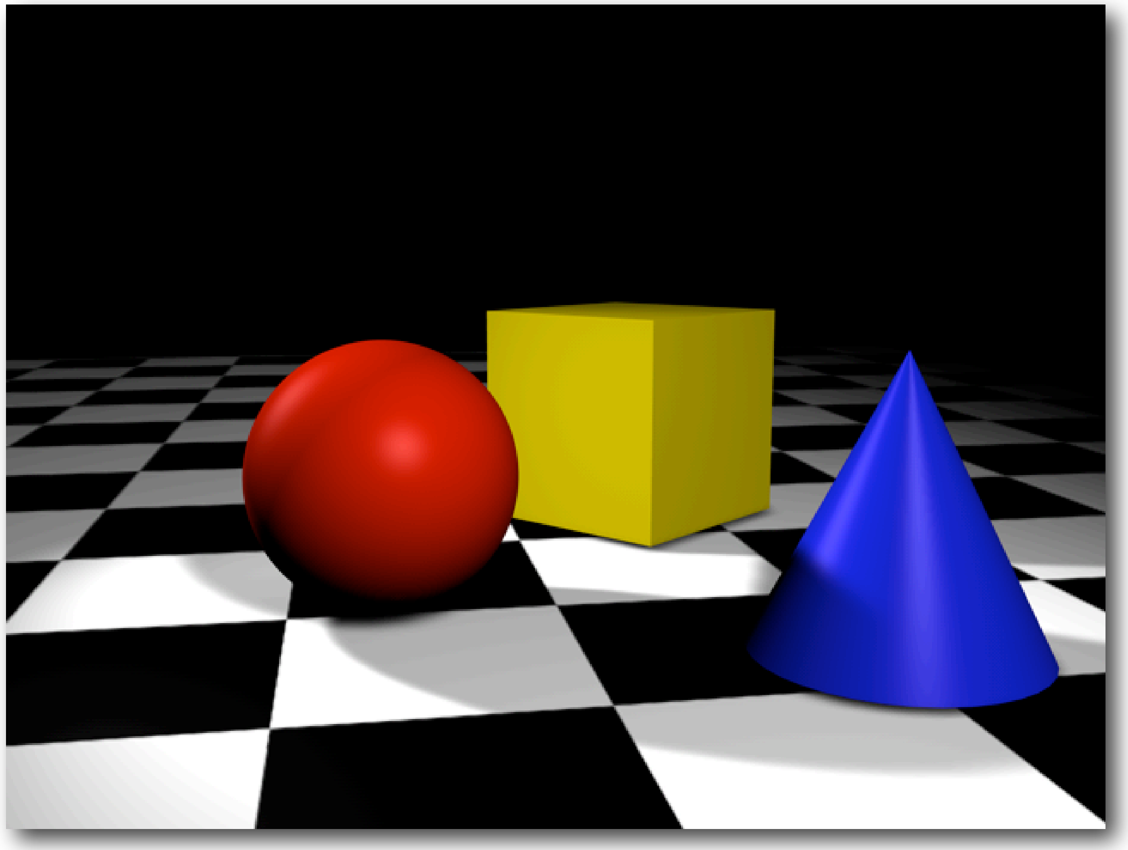
Projections

- Projecting 3D coordinates onto a 2D surface is not, in general, a reversible operation. That is, from the 2D coordinates we cannot recover the original 3D information.
- In addition there are many possible projections of 3D data onto 2D data.
- There are two basic methods for projecting 3D objects onto a 2D viewing surface:
  - (1) All points of the object can be projected along parallel lines: **parallel projection;**
  - (2) All points of the object can be projected to the surface along lines that converge to a position called the **centre of projection: perspective projection**.

Projections (cont.)

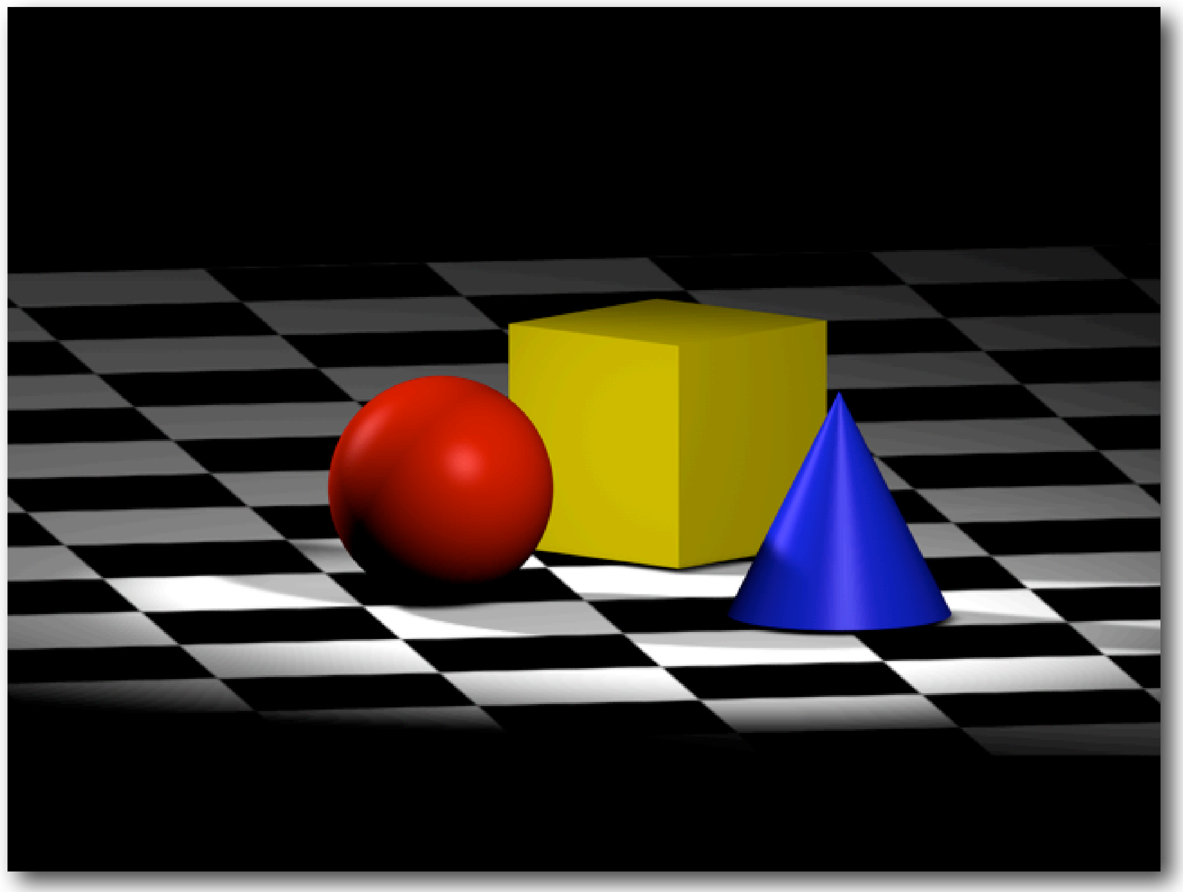
- In both cases the intersection of a projection line with a viewing surface determines the coordinates of the projected point on the projection plane.

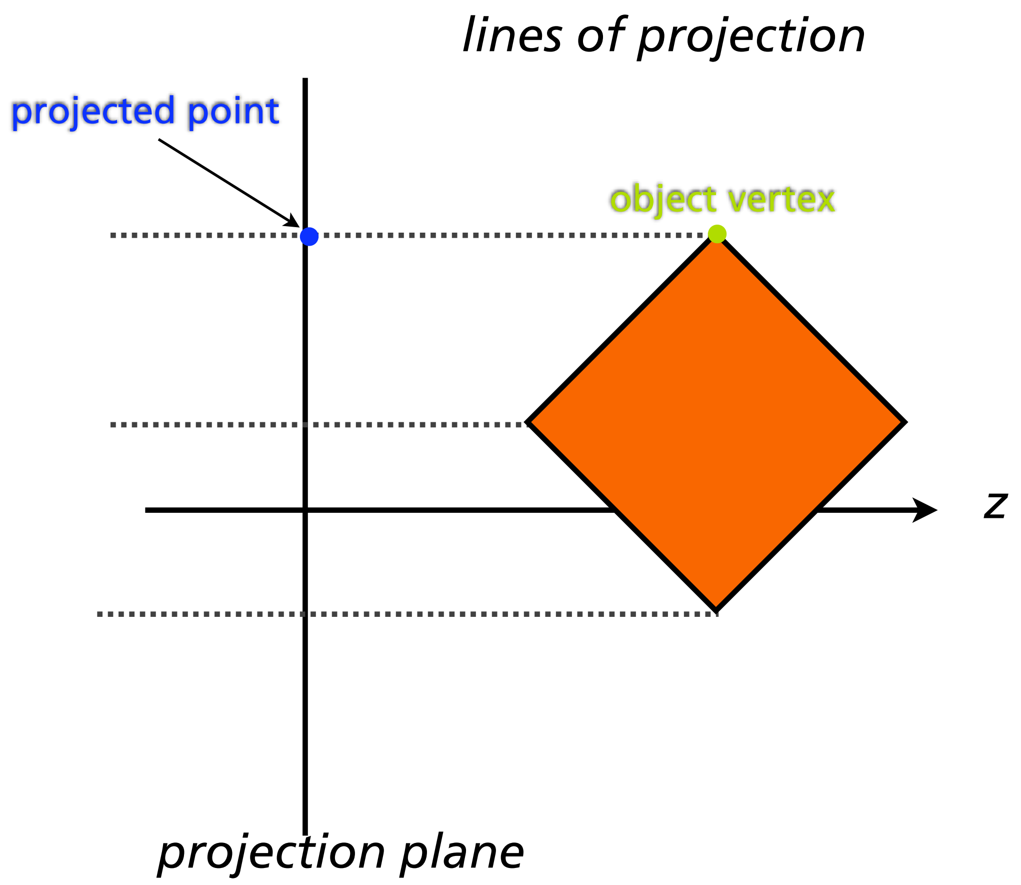

Parallel Projection


Perspective Projection

Parallel Projections

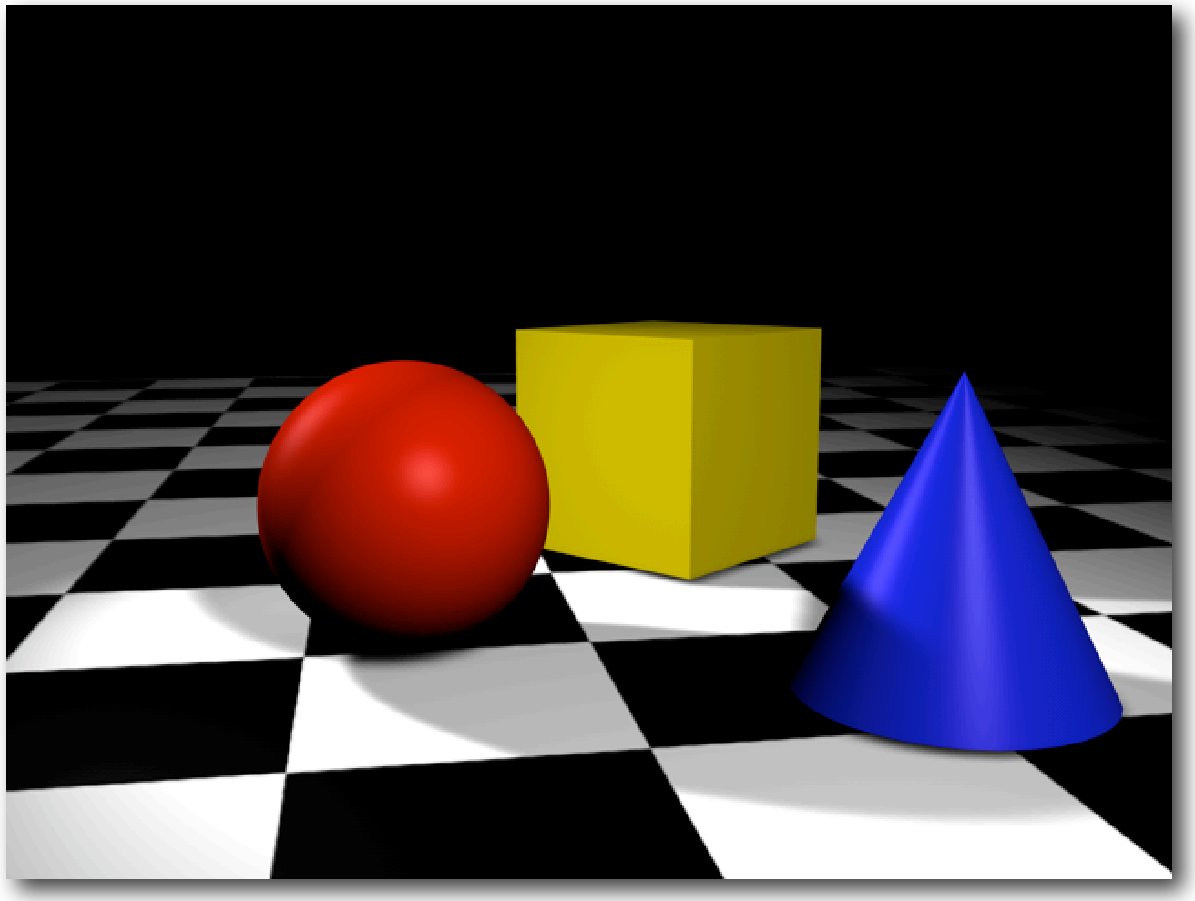
- Preserve relative dimensions of objects (used in 3D scale drafting);
- Generally does not give a realistic view of an object;
- Often used in 3D animation programs to give top, front and side views.

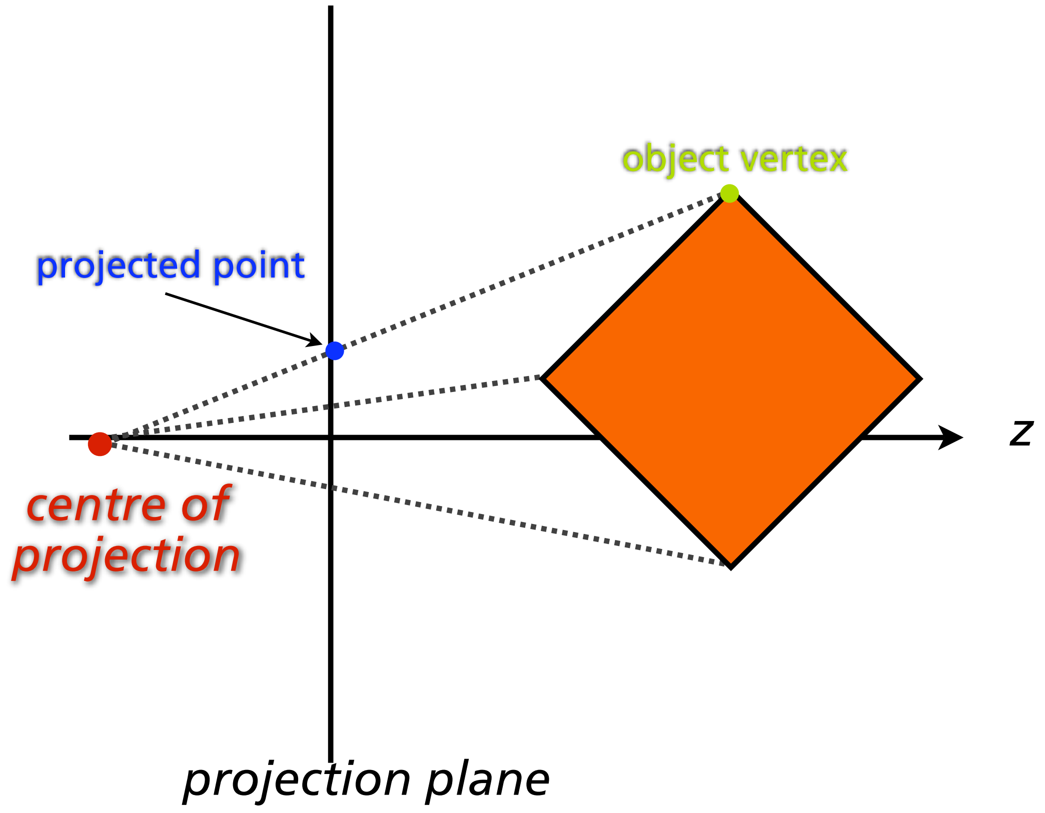




Perspective Projections

- Do not preserve the relative dimensions of objects;
- give a realistic view of objects;
- lines get scaled depending on their distance from the projection plane.




object vertex
projected point
z
centre of projection
projection plane