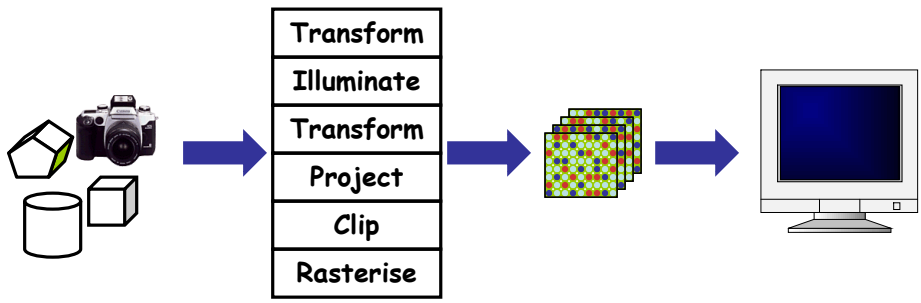
3D Graphics Programming Tools

The rendering pipeline (summary/revisions)

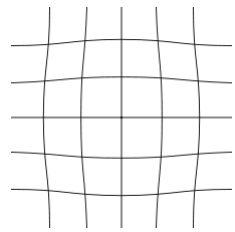
Rendering 3D scenes

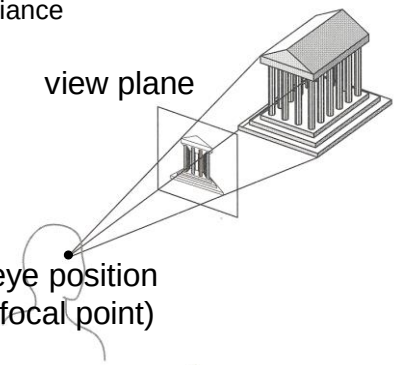


model & camera parameters | rendering pipeline | framebuffer | display

Camera models

- Most common model: pin-hole camera
  - All captured light rays arrive along paths toward a focal point without lens distortion, everything is in focus
  - Sensor response proportional to radiance
  - Note: other models consider:
    - » depth of field
    - » motion blur
    - » lens distortion





The rendering pipeline

- Move models
- Illuminate
- Move camera
- Project to display
- Clip
- Rasterise

The rendering pipeline: 3-D

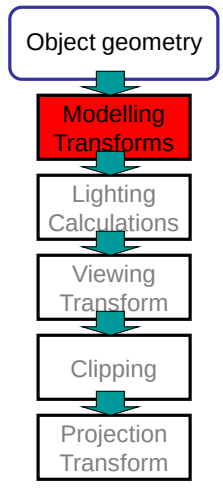
Transformations

- Transformations → used in three ways
  - modelling transforms
  - viewing transforms
    - move the camera
  - projection transforms
    - change the type of camera

The rendering pipeline – ex.

- Paper 2017-18
  - Q1 a)
- Paper 2016-17
  - Q1 a)
  - Q1 b)
- Paper 2015-16
  - Q1 a)
  - Q1 d)

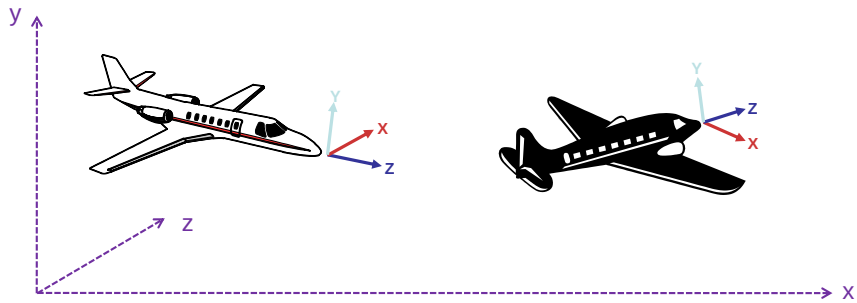
The rendering pipeline: 3-D



result: all vertices of scene in shared 3D “world” coordinate system

Modelling Transformations

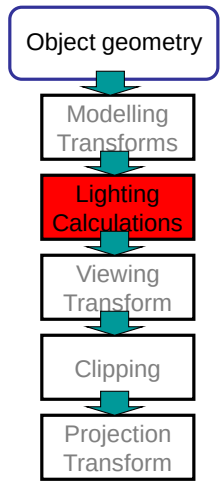
- Modelling transforms
  - Size, place, scale, and rotate objects and parts of the model with respect to each other
  - Object coordinates → world coordinates



Modelling and Modelling transformations – ex.

- Paper 2017-18
  - Q2 a)
  - Q2 b)
- Paper 2016-17
  - Q2 a)
  - Q2 b)
- Paper 2015-16
  - Q2 d)
  - Q3 a)
  - Q3 c)

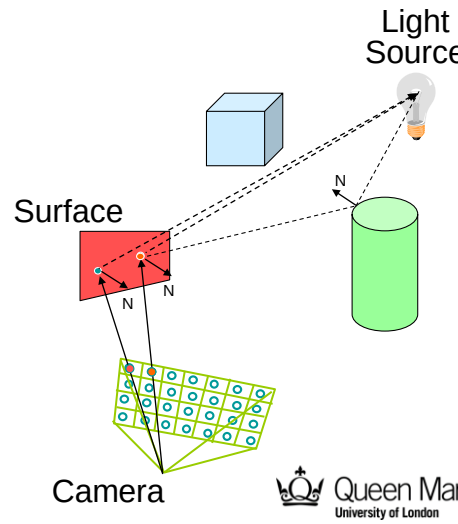
The rendering pipeline: 3-D



Result: all geometric primitives are illuminated

Lighting simulation

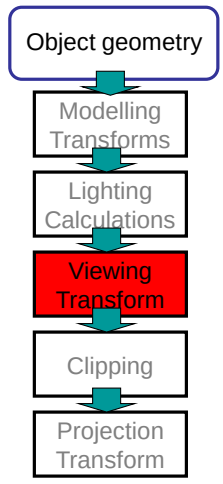
- Lighting parameters
  - light source emission
  - surface reflectance
- Direct illumination
- Global illumination



Lighting calculations – ex.

- Paper 2017-18
  - Q3 a)
  - Q3 b)
- Paper 2016-17
  - Q3 a)
  - Q3 b)
- Paper 2015-16
  - Q4 a)
  - Q4 b)
  - Q4 c)
  - Q4 d)
  - Q4 e)

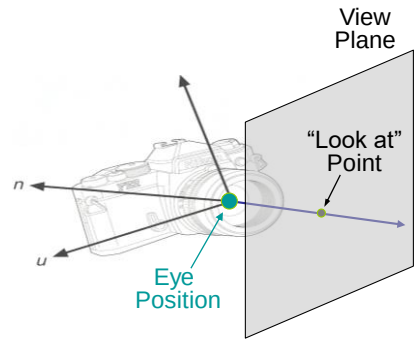
The rendering pipeline: 3-D



Result: scene vertices in 3-D "view" or "camera" coordinate system

Camera parameters

- Position
  - eye position (px, py, pz)
- Orientation
  - view direction (dx, dy, dz)
  - up direction (ux, uy, uz)
- Aperture
  - Field of view (xfov, yfov)

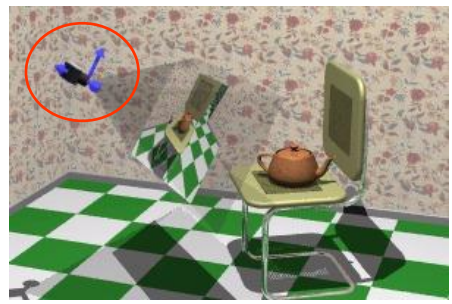


Viewing Transformations

- Viewing transform
  - rotate & translate the world to lie directly in front of the camera
    - typically place camera at origin
    - typically looking down Z axis
  - world coordinates → view coordinates

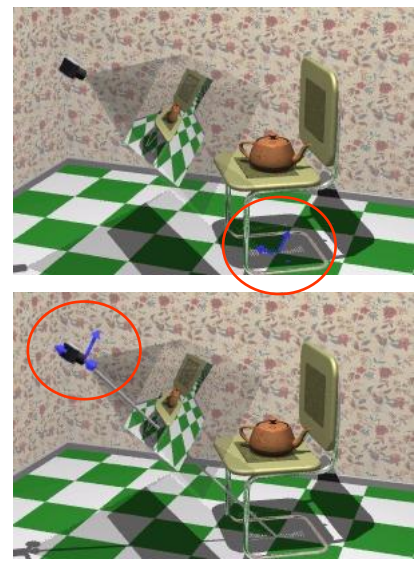
Viewing transformations

- Create a camera-centered view
  - camera is at origin
  - camera is looking along negative z-axis
  - camera's 'up' is aligned with y-axis



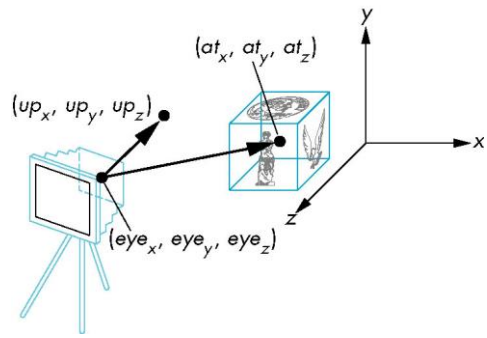
2 basic steps

- Align the two coordinate frames by rotation
- Translate to align origins



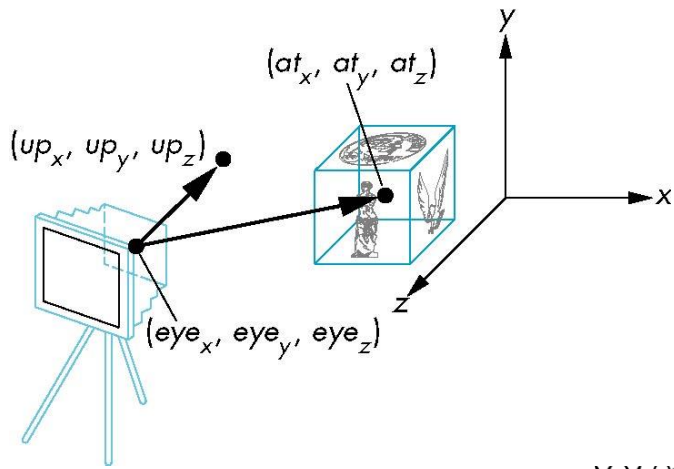
Creating camera coordinate space

- Specify
  - the eye point → a point where the camera is located in world space
  - the lookat point → a point in world space that we wish to become the center of view
  - the up vector → a vector in world space that we wish to point up in camera image



gluLookAt

```
gluLookAt(eyex, eyey, eyez, atx, aty, atz, upx, upy, upz)
```



Constructing viewing transformation

- Create a vector from eye point to lookat point

$$\begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix} = \begin{bmatrix} lookat_x \\ lookat_y \\ lookat_z \end{bmatrix} - \begin{bmatrix} eye_x \\ eye_y \\ eye_z \end{bmatrix}$$

- Normalise the vector

$$\hat{l} = \frac{\bar{l}}{\sqrt{l_x^2 + l_y^2 + l_z^2}}$$

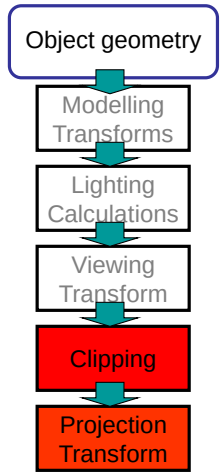
- Desired rotation matrix V should map this vector to $[0, 0, -1]^T$

$$\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \mathbf{V}\hat{l}$$

Viewing transformation – ex.

- Paper 2017-18
  - Q2 c)
- Paper 2016-17
  - Q4 a)
- Paper 2015-16
  - Q2 b)

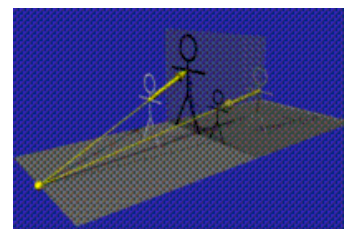
The rendering pipeline: 3-D



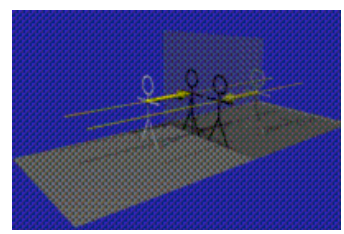
Result: remove geometry that is out of view

Transformations

- Perspective camera

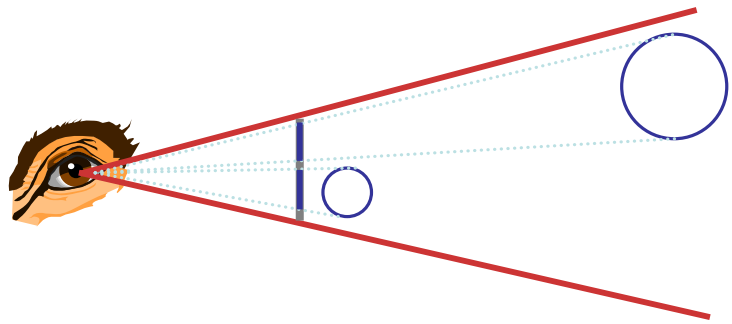


- Orthographic camera



Projection Transformations

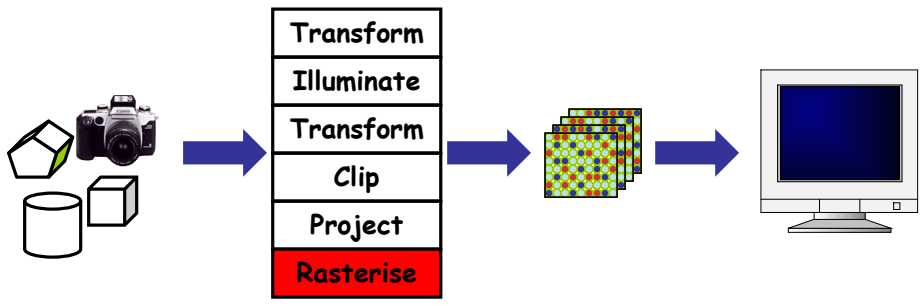
- Projection transform
  - Apply perspective foreshortening
    - Distant = small: the pinhole camera model
  - View coordinates → screen coordinates



Projection transformation – ex.

- Paper 2017-18
  - Q3 c)
- Paper 2016-17
  - Q3 c)
- Paper 2015-16
  - Q3 d)

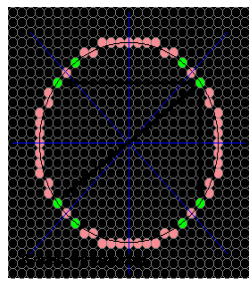
Rendering 3D scenes

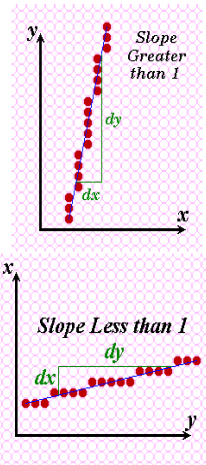


model & camera parameters | rendering pipeline | framebuffer | display

Rasterise

- Convert screen coordinates to pixel colors





Rasterisation – ex.

- Paper 2017-18
  - Q4 b)
- Paper 2016-17
  - Q4 b)

The rendering pipeline

- Camera models
- Modelling transforms
- Lighting
- Viewing transforms
- Clipping
- Projection transforms
- Rasterise