THE VIRTUAL CAMERA

VIRTUAL CAMERA ADVANTAGES

Weightless - Capable of infinitely fast or infinitely slow movement (time control)
Scale independent - Capable of impossible movement
Invisible
Easily recreated / modified

LECTURE SUMMARY

The Virtual Viewer or Camera Model

- Key Parameters: describing a virtual camera
- Projection: from 3D to 2D
- Occlusion and Hidden Surface Removal

RECALL: TYPICAL OBJECT REPRESENTATION

Start with Points/Vertices on the surface of an object
- Stored as positional vectors
- \( x, y, z \) from the origin

Edges are line segments on the surface, defined by pairs of points.

Closed polygons are made up of a number of co-planar edges.
**3D OBJECT REPRESENTATIONS**

Polygon Mesh a.k.a. Wireframe data

- **3D Coordinates**
- **Normals** (could be calculated at run-time)
- **Tex-coords**

**PROLOGUE**

3D Object Transformations

- **Translation**
- **Scaling**
- **Rotation**

- Points are often represented as homogeneous coordinates \( x, y, z, w \) where \( w = 1 \)
- Multiplying a point by a matrix "transforms" the point
- Repeating this for all points on the mesh transforms the object

- **Translation**
- **Scaling**
- **Rotation**

**TRANSFORM**

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} =
\begin{bmatrix}
x + w \\
y + b \\
z + c
\end{bmatrix},
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} =
\begin{bmatrix}
x + \cos \theta \ y - \sin \theta \\
y + \sin \theta \ x + \cos \theta \\
z + c
\end{bmatrix},
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} =
\begin{bmatrix}
x + \cos \phi \ y - \sin \phi \\
y + \sin \phi \ x + \cos \phi \\
z + c
\end{bmatrix}
\]

**TAKEAWAY MESSAGE**

Multiply point by transform matrix to transform it

When a graphics system (e.g., 3DS max or other standalone 3D program) needs to transform an object, all points on the object are multiplied by the matrix related to that transform.
VIEWING TRANSFORMS
Camera Positioning and Movement

THE VIEWING PROCESS

To create a view of a scene we need:
- a description of the scene geometry
- a view definition (camera)

We need to flatten the 3D objects onto a 2D plane — projection plane (or view plane)

A window is defined on this plane — a sub-part of the full plane

This window is then mapped to a viewport

Window: rectangular region of interest
Viewport: Rectangular region on screen (in pixel coordinates)
In 3D graphics the virtual camera is an essential part of the scene.

- Determines how a scene is drawn.
- Relatively easy to model but quite powerful in what we can do with it.

This lecture will discuss:

- what parameters are required by 3D applications for modelling a camera view.
- how these are applied to represent 3D worlds on a 2D display device.

*A: Alternatively, some texts refer to a virtual eye or viewer instead of a camera.

### View parameters

A view is described in terms of:

- **Camera location** position in the world - xyz-coordinates representing displacement from the origin
- **Viewing direction** which direction are we aiming the camera - a direction vector
- **Camera orientation** usually defined by an up vector

![Camera Diagram](image)

**N.B.** MAX’s “FREE” Cameras can be fully modelled using these three vectors.

### 3D Camera Viewing

Note that the following scenes should produce the same image.

In 3D Computer Graphics, we prefer to think of all possible situations in terms of something like scene2.

We call this the **Camera Co-ordinate System**: a re-definition of the scene – with the camera at the origin looking down the z-axis.

### Viewing Coordinates

A Viewing Co-ordinate system (“the xyz of the camera”) is defined by the following viewing parameters:

- The position of the camera $<x_0, y_0, z_0>$
- Three basis vectors:
  - Viewing direction (N)
  - View up vector (V): establishes orientation of “camera”

Position, N and V are parameters provided by the user. Normalized versions $\hat{n}, \hat{v}, \hat{u}$ are calculated:

- $\hat{n} = \frac{N}{\|N\|}$ ("direction")
- $\hat{u} = \frac{v \times n}{\|v \times n\|}$ ("left")
- $\hat{v} = \hat{u} \times \hat{n}$ ("up")

### 3D Viewing Transformation

Based on these vectors and the Viewing Reference Point $<x_0, y_0, z_0>$ the Viewing Transformation can be defined by composition of the following two transformations:

- **Translation** $T = 
\begin{pmatrix}
1 & 0 & -x_0 \\
0 & 1 & -y_0 \\
0 & 0 & 1
\end{pmatrix}
$

- **Orientation** $R = 
\begin{pmatrix}
u_0 & v_0 & u_0 & 0 \\
v_0 & v_0 & v_0 & 0 \\
u_0 & v_0 & n_0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
$

Combine these two (multiply them together) to get a single viewing transform.

$$M_{VC} = R \cdot T$$

Apply this to all object/vertices to get their positions in the camera co-ordinate system.
PROJECTION

3D Graphics deals with 3D data, which in most cases is displayed on a 2D display device.

Projection involves re-interpretting 3D objects as 2D data.

The main types of projection used in 3D Computer Graphics are:
- Parallel Projections
- Perspective Projection

In addition to scene geometry and camera parameters, we will introduce one more key viewing component.
- The projection plane (a.k.a. viewplane) is an imaginary plane onto which we will project our 3D data.

PROJECTIONS ONTO A PLANE

PARALLEL PROJECTIONS

A very simple type of operation that reduces 3D data to 2D data.
Not representative of what takes place in a real camera but useful in design applications.

Defined with:
- a direction of projection (DOP)
- a projection plane
- width and height of window

If DOP is orthogonal (90 degrees) to viewplane then we have Orthographic projection otherwise we have Oblique projection.
Essentially, we extend projection lines parallel to the direction of projection (viewing direction) from each point on scene. Find out where these lines intersect the projection plane to determine where to draw the object.

### Parallel Projections

Parallelogram: parallel projection lines are parallel in the viewing direction. They intersect the projection plane at an angle.

Orthographic means that the DOP is perpendicular to the projection plane, i.e. both angles 90 degrees and all projection lines are thus perpendicular to the plane.

Oblique means DOP is not perpendicular, i.e. angles not equal to 90 degrees.

### Oblique Projections

Two commonly used forms of oblique projection:

- Cavalier: Lines along the viewing direction are drawn at the original length (but rotated).
- Cabinet: Lines along the viewing direction are drawn at half their length (but rotated). Slightly more realistic.

Viewer can then easily infer the original dimensions of the object from the projected image. Useful for tech drawings.
**GENERAL PARALLEL PROJECTION**

Technical Aside (optional info)

How to find the projected value \((x_p, y_p)\) for any point \((xyz)\)

\[
x_p = x + d \cos \phi \\
y_p = y + d \sin \phi
\]

where \(d = \frac{y}{\tan \phi}\)

\[PM = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\]

**ISOMETRIC PROJECTIONS**

A well-known style of Orthographic projection

**ISOMETRIC VIEWS**

Isometric views in games

Qbert, Sim City series, Diablo, Farmville

**AXONOMETRIC PROJECTIONS**

Biaxometric

Triaxometric

**PERSPECTIVE PROJECTION**

Although there is evidence that people were aware of perspective since 4000 BC however they were unable to recreate it effectively until much later 1400 AD.

In the painting on the right the lines on the side of the building don’t converge properly and create an ambiguous idea of the structure. This is apparent in many pre-1400 paintings.
LINEAR PERSPECTIVE

The discovery of Linear Perspective with one vanishing point is attributed to an architect named Filippo Brunelleschi in about 1415.

- "Best Innovation in Painting; Everything in Perspective" New York times article: http://query.nytimes.com/gst/fullpage.html?res=9B07E3DE1131F93BA25757C0A9D0A95820
- The Origins of Perspective: http://www.dartmouth.edu/~matc/math5.geometry/unit11/unit11.html

Many artists quickly picked up the technique soon after.

PERSPECTIVE PROJECTION

Piero della Francesca “Flagellation of Christ”, late 1450s.

PERSPECTIVE PROJECTION

Pietro Perugino (1481-82)

PERSPECTIVE PROJECTION

Vermeer (1670) is believed by many to have used a "camera obscura" to create his extremely realistic projections.

DECONSTRUCTING VERMEER

Perspective projections produce a perspective foreshortening effect: "objects in the distance appear smaller”

They tend, therefore, to appear more realistic than parallel projections.

Parallel lines in the 3D model which are not parallel to the projection plane, converge to a vanishing point.
Extend lines from each point on the scene to the center of projection (camera position). Where these lines intersect with the projection plane is where we draw the object.

**Perspective Projections**

Defined with the following parameters:
- centre of projection (COP)
- field of view (θ, ϕ)
- projection direction
- up direction

**Technical Aside (optional info)**

Consider a perspective projection with the viewpoint at the origin and a viewing direction oriented along the positive z-axis and the view-plane located at $z = -d$.

Preserving ratios in the triangles:

$$\frac{y}{z} = \frac{y'}{z'} \Rightarrow y' = \frac{y}{z}z'$$

A similar construction is used for finding $x'$. Where $M$ is

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

**Lens Configurations**

- 10mm Lens (fov = 122°)
- 20mm Lens (fov = 84°)
- 35mm Lens (fov = 54°)
- 200mm Lens (fov = 10°)

**Perspective Illusions**

Basically: use the floor as your projection plane — similar to a shadow (with a light source at the eye position).
JULIAN BEEVER STREET ART

Breaking the illusion – seen from a different angle or with the artist’s shadow, we can see what’s really going on.

MASSACIO

Massacio – Holy Trinity 1428

A fresco that uses perspective to create the illusion of an alcove.

RAPHAEL 1509

ANDREA POZZO (1642 – 1709)

Fresco at Church of Sant’Ignazio, Rome

ANDREA POZZO (1642 – 1709)

Fresco in Jesuit Church, Vienna

AMES ROOM OPTICAL ILLUSION
HIDDEN SURFACE REMOVAL

After projection we reduce objects to 2D. Essentially we depth detail. For 2D information it is not automatically evident what objects should obscure others. **HOW DO WE ENSURE CORRECT OCCLUSION?**

Note that this is largely handled by systems such as 3DS max for you but sometimes we need to know how this works.

**BACKFACE CULLING**

- A quick test for fast eliminating some occluding polygons.
- Render only "front facing" polygons which are identified based on their normal.

If $-90 < \theta < 90$ then the polygon can be considered front facing and can be drawn otherwise it is ignored from rendering.

**THE PAINTERS ALGORITHM**

Sort polygons according to their distance from camera and render from back to front.

Draw over the polygons in the back (more or less). Problems arise when polygons overlap or are allowed to pierce one another. In this case split the polygons (but this is time-consuming).

**HSR ALGORITHMS**

In Graphics we commonly use Backface Culling and Z-buffer methods.

But they are many different approaches varying in Efficiency w.r.t. time and storage space:

- Object Precision Methods: Based on Modelling coordinates. Correct to the precision of the machine (and the model).
- Image Precision: Per pixel, decide which colour it should be drawn in. Correct to the precision of screen window system.

**THE Z-BUFFER ALGORITHM**

The most widely used Hidden surface removal algorithm.

We rasterize polygon by polygon and determine which (parts of) polygons get drawn on the screen.

Store two images:

- The actual rendered image (the frame-buffer image)
- And a depth image has the same dimensions as the frame-buffer but only stores depths at each position.
THE Z-BUFFER ALGORITHM

- Initialize all depth pixels to D
- Initialize all frame buffer pixels to background color

For each polygon:
- For each pixel \((x,y)\) on polygon
  - Evaluate depth value \(z\)
  - Compare \(z\) to depth buffer pixel at this position
    - \(d = \text{depth buffer}(x,y)\)
    - \(\text{if } z > d\)
      - \(\text{depth buffer}(x,y) = z\)
  - Frame buffer \((x,y)\) = color of this polygon

No hidden surface removal!

For culling, we need to ensure that objects are proper solids. This teapot is not quite a proper solid and as a result the image is incorrect. However, combining backface culling with more expensive depth testing is usually a good practice.