Interactive Computer Graphics: Lecture 2

Transformations for animation
The most useful operations: 
Previously defined transformation matrices

- Translation

\[
\begin{pmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
= 
\begin{pmatrix}
x + t_x \\
y + t_y \\
z + t_z \\
1
\end{pmatrix}
\]

- Scaling

\[
\begin{pmatrix}
s_x & 0 & 0 & 0 \\
0 & s_y & 0 & 0 \\
0 & 0 & s_z & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
= 
\begin{pmatrix}
s_x x \\
s_y y \\
s_z z \\
1
\end{pmatrix}
\]
Rotations about $x$, $y$ and $z$ axes.

\[ \mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 & 0 \\ 0 & \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \]

\[ \mathbf{R}_y = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \]

\[ \mathbf{R}_z = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \]
Rotations about $x$, $y$ and $z$ axes.

We now consider more complex transformations which are combinations of translations, scalings and rotations.

\[
R_x = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & -\sin \theta & 0 \\
0 & \sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
R_y = \begin{pmatrix}
\cos \theta & 0 & \sin \theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
R_z = \begin{pmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Flying sequences

• In generating animated flying sequences, we require the viewpoint to move around the scene.
• This implies a change of origin
• Let

  – the required viewpoint be $C = (C_x, C_y, C_z)$

  – the required view direction be $d = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}$
Recall the canonical form for perspective projection

We look along the $z$-axis and the $y$-axis is ‘up’
Transformation of viewpoint

Coordinate system for definition

Coordinate system for viewing

Graphics Lecture 2: Slide 7
Flying Sequences

• The required transformation is in three parts:
  
  1. Translation of the origin
  2. Rotate about y-axis
  3. Rotate about x-axis

• The two rotations are to line up the z-axis with the view direction
1. Translation of the Origin

\[
\mathbf{A} = \begin{pmatrix}
1 & 0 & 0 & -C_x \\
0 & 1 & 0 & -C_y \\
0 & 0 & 1 & -C_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
2. Rotate about $y$ until $d$ is in the $y$-$z$ plane

\[
\mathbf{B} = \begin{pmatrix}
\cos \theta & 0 & -\sin \theta & 0 \\
0 & 1 & 0 & 0 \\
\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
d_z/v & 0 & -d_x/v & 0 \\
0 & 1 & 0 & 0 \\
d_x/v & 0 & d_z/v & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

$\|\mathbf{v}\| = v = \sqrt{d_x^2 + d_z^2}$

$\cos \theta = \frac{d_z}{v}$

$\sin \theta = \frac{d_x}{v}$
3. Rotate about $x$ until $d$ points along the $z$-axis

$$v = \sqrt{d_x^2 + d_z^2}$$

$$\cos \phi = \frac{v}{|d|}$$

$$\sin \phi = \frac{d_y}{|d|}$$

$$C = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \phi & -\sin \phi & 0 \\
0 & \sin \phi & \cos \phi & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & v/|d| & -d_y/|d| & 0 \\
0 & d_y/|d| & v/|d| & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}$$
Combining the matrices

• A single matrix that transforms the scene can be obtained from the matrices $A$, $B$ and $C$ by multiplication

$$T = CBA$$

• And for every point $P$ of the scene, we calculate

$$P_t = TP$$

• The view is now in ‘canonical’ form and we can apply the standard perspective or orthographic projection.
Verticals

• So far we have not looked at verticals

• Usually, the \( y \) direction is treated as vertical, and by doing the \( R_y \) transformation first, things work out correctly

• However it is possible to invert the vertical
Transformations and verticals

view direction $\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$
Rotation about a general line

- Special effects, such as rotating a scene about a general line can be achieved by multiple transformations

- The transformation is formed by:
  - Making the line of rotation one of the Cartesian axes
  - Doing the rotation (about the chosen axis)
  - Restoring the line to its original place
Rotation about a general line

- The first part is achieved using the same matrices that we derived for the flying sequences

\[ CBA \]

- This rotates the general line so it is aligned with the \( z \)-axis.

- We then carry out the rotation about the \( z \)-axis then follow this by the inversion of the initial matrices.

- So the full matrix \( T \) of the combined transformation is

\[ T = A^{-1}B^{-1}C^{-1}R_zCBA \]
Other effects

• Similar effects can be created using this approach

• e.g. to make an object shrink (and stay in place)

1. Move the object to the origin
2. Apply a scaling matrix
3. Move the object back to where it was
Projection by matrix multiplication

• Usually projection and drawing of a scene comes after the transformation(s)

• It is therefore convenient to combine the projection with the other parts of the transformation

• So it is useful to have matrices for the projection operation
Orthographic projection matrix

• For (canonical) orthographic projection, we simply drop the \( z \)-coordinate:

\[
\mathbf{M}_o = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
\mathbf{M}_o \begin{pmatrix}
x \\
y \\
z \\
1 \\
\end{pmatrix} = \begin{pmatrix}
x \\
y \\
0 \\
1 \\
\end{pmatrix}
\]
Perspective projection matrix

- Perspective projection of homogenous coordinates can also be done by matrix multiplication:

\[
M_p = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/f & 0
\end{pmatrix}
\]

\[
M_p \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/f \end{pmatrix}
\]
Perspective projection matrix: Normalisation

- Remember we can normalise homogeneous coordinates, so

\[
\mathcal{M}_p \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/f \end{pmatrix}
\]

which is the same as

\[
\begin{pmatrix} x f/z \\ y f/z \\ f \\ 1 \end{pmatrix}
\]

- as required.
Projection matrices are singular

- Notice that both projection matrices are singular (i.e. ‘non-invertible’, zero-determinant, …)

\[
\mathbf{M}_p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \quad \mathbf{M}_o = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\]

- This is because a projection transformation cannot be inverted.
- Given a 2D image, we cannot in general reconstruct the original 3D scene.
Homogenous coordinates as vectors

- We now take a second look at homogeneous coordinates, and their relation to vectors.
- In the previous lecture we described the fourth ordinate as a scale factor.

<table>
<thead>
<tr>
<th>Homogeneous</th>
<th>Cartesian</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x)$</td>
<td>$(x/s)$</td>
</tr>
<tr>
<td>$(y)$</td>
<td>$(y/s)$</td>
</tr>
<tr>
<td>$(z)$</td>
<td>$(z/s)$</td>
</tr>
<tr>
<td>$(s)$</td>
<td></td>
</tr>
</tbody>
</table>
Homogenous coordinates and vectors

Homogenous coordinates fall into two types:

1. Position vectors
   - Those with non-zero final ordinate \((s > 0)\).
   - Can be normalised into Cartesian form.

\[
\begin{pmatrix}
x \\ y \\ z \\ s
\end{pmatrix}
\]

2. Direction vectors
   - Those with zero in the final ordinate.
   - Have direction and magnitude.

\[
\begin{pmatrix}
x \\ y \\ z \\ 0
\end{pmatrix}
\]
Adding direction vectors

- If we add two direction vectors we obtain a direction vector

\[
\begin{pmatrix} x_i \\ y_i \\ z_i \\ 0 \end{pmatrix} + \begin{pmatrix} x_j \\ y_j \\ z_j \\ 0 \end{pmatrix} = \begin{pmatrix} x_i + x_j \\ y_i + y_j \\ z_i + z_j \\ 0 \end{pmatrix}
\]

- This is the normal vector addition rule.
Adding position and direction vectors

- If we add a direction vector to a position vector, we obtain a position vector:

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix} + \begin{pmatrix}
x \\
y \\
z \\
0
\end{pmatrix} = \begin{pmatrix}
X + x \\
Y + y \\
Z + z \\
1
\end{pmatrix}
\]

Nice result.
Ties in with definition of straight line in Cartesian space which uses a point and a direction.
Adding two position vectors

- If we add two position vectors, we obtain their mid-point

\[
\begin{pmatrix}
X_a \\
Y_a \\
Z_a \\
1
\end{pmatrix} + \begin{pmatrix}
X_b \\
Y_b \\
Z_b \\
1
\end{pmatrix} = \begin{pmatrix}
X_a + X_b \\
Y_a + Y_b \\
Z_a + Z_b \\
2
\end{pmatrix} = \begin{pmatrix}
(X_a + X_b) / 2 \\
(Y_a + Y_b) / 2 \\
(Z_a + Z_b) / 2 \\
1
\end{pmatrix}
\]

- This is reasonable since adding two position vectors has no real meaning in vector geometry
The structure of a transformation matrix

- The bottom row is always 0 0 0 1
- The columns of a transformation matrix comprise three direction vectors and one position vector

\[
\begin{bmatrix}
q_x & r_x & s_x & C_x \\
q_y & r_y & s_y & C_y \\
q_z & r_z & s_z & C_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
q_x \\
q_y \\
q_z \\
0
\end{bmatrix}
\begin{bmatrix}
r_x \\
r_y \\
r_z \\
0
\end{bmatrix}
\begin{bmatrix}
s_x \\
s_y \\
s_z \\
0
\end{bmatrix}
\begin{bmatrix}
C_x \\
C_y \\
C_z \\
1
\end{bmatrix}
\]
Characteristics of transformation matrices

- Direction vector: Zero, in the last ordinate ⇒ not affected by the translation.

\[
\begin{pmatrix}
q_x & r_x & s_x & C_x \\
q_y & r_y & s_y & C_y \\
q_z & r_z & s_z & C_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
* \\
*
\end{pmatrix}
= 
\begin{pmatrix}
* \\
*
\end{pmatrix}
\]

- Position vector: 1 in the last ordinate ⇒ all vectors will have the same displacement.

\[
\begin{pmatrix}
q_x & r_x & s_x & C_x \\
q_y & r_y & s_y & C_y \\
q_z & r_z & s_z & C_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
* \\
*
\end{pmatrix}
= 
\begin{pmatrix}
* + C_x \\
* + C_y \\
* + C_z \\
1
\end{pmatrix}
\]

- If we do not shear the object the three vectors \(q\), \(r\) and \(s\) will remain orthogonal, ie:

\[
q \cdot r = r \cdot s = q \cdot s = 0
\]
What do the individual columns mean?

- To see this, consider the effect of the transformation in simple cases.
- For example take the unit direction vectors along the Cartesian axes
  - e.g. along the $x$-axis, $\mathbf{i} = (1, 0, 0, 0)^T$

\[
\begin{pmatrix}
    q_x & r_x & s_x & C_x \\
    q_y & r_y & s_y & C_y \\
    q_z & r_z & s_z & C_z \\
    0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    1 \\
    0 \\
    0 \\
    0
\end{pmatrix}
= 
\begin{pmatrix}
    q_x \\
    q_y \\
    q_z \\
    0
\end{pmatrix}
\]
What do the individual columns mean?

- The other axis transformations:
  Similarly, we find the following transformations of unit vectors \( \mathbf{j} \) and \( \mathbf{k} \)

\[
\begin{align*}
\mathbf{j} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} & \rightarrow \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \\
\mathbf{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & \rightarrow \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}
\end{align*}
\]
What do the individual columns mean?

- Transforming the origin:
  - If we transform the origin, \( (0, 0, 0, 1)^T \), we end up with the last column of the transformation matrix

\[
\begin{pmatrix}
q_x & r_x & s_x & C_x \\
q_y & r_y & s_y & C_y \\
q_z & r_z & s_z & C_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}
= 
\begin{pmatrix}
C_x \\
C_y \\
C_z \\
1
\end{pmatrix}
\]
The meaning of a transformation matrix

Putting everything together …

The columns are the original axis system after transforming to the new coordinate system

\[
\begin{pmatrix}
q_x & r_x & s_x & C_x \\
q_y & r_y & s_y & C_y \\
q_z & r_z & s_z & C_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

q  transformed \(x\)-axis  
\(r\)  transformed \(y\)-axis  
\(s\)  transformed \(z\)-axis  
C  transformed origin
Effect of a transformation matrix

Before

\[
\begin{pmatrix}
q_x & r_x & s_x & C_x \\
q_y & r_y & s_y & C_y \\
q_z & r_z & s_z & C_z \\
0 & 0 & 0 & 1 \\
\end{pmatrix} = [q \ r \ s \ C]
\]

Tells us the old axes and origin in the new coordinate system.
What we want is the other way round

- Normally,
  - We are not given the transformation matrix that moves the scene to that coordinate system, we need to find it
  - We are given a view direction $\mathbf{d}$ and location $\mathbf{C}$

To see how to get the matrix, we introduce the idea of the dot product as a projection
The dot product as a projection

• The dot product is defined as
  \[ \mathbf{P} \cdot \mathbf{u} = |\mathbf{P}| |\mathbf{u}| \cos \theta \]

• If \( \mathbf{u} \) is
  – a unit vector then \( \mathbf{P} \cdot \mathbf{u} = |\mathbf{P}| \cos \theta \)
  – along a co-ordinate axis then \( \mathbf{P} \cdot \mathbf{u} \) is the ordinate of \( \mathbf{P} \) in the direction of \( \mathbf{u} \)
Changing axes by projection

- Extending the idea to three dimensions we can see that a change of axes can be expressed as projections using the dot product.

For example, call the first coordinate of $P$ in the new system $P_x^t$.

$P_x^t = (P - C) \cdot u$

$= P \cdot u - C \cdot u$
Transforming point \( \mathbf{P} \)

- Given point \( \mathbf{P} \) in the \((x, y, z)\) axis system, we can calculate the corresponding point in the \((u, v, w)\) system as:

\[
P_x^t = (\mathbf{P} - \mathbf{C}) \cdot \mathbf{u} = \mathbf{P} \cdot \mathbf{u} - \mathbf{C} \cdot \mathbf{u}
\]
\[
P_y^t = (\mathbf{P} - \mathbf{C}) \cdot \mathbf{v} = \mathbf{P} \cdot \mathbf{v} - \mathbf{C} \cdot \mathbf{v}
\]
\[
P_z^t = (\mathbf{P} - \mathbf{C}) \cdot \mathbf{w} = \mathbf{P} \cdot \mathbf{w} - \mathbf{C} \cdot \mathbf{w}
\]

- Or, in matrix notation:

\[
\begin{pmatrix}
P_x^t \\
P_y^t \\
P_z^t \\
1
\end{pmatrix}
= \begin{pmatrix}
u_x & u_y & u_z & -\mathbf{C} \cdot \mathbf{u} \\
v_x & v_y & v_z & -\mathbf{C} \cdot \mathbf{v} \\
w_x & w_y & w_z & -\mathbf{C} \cdot \mathbf{w} \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
P_x \\
P_y \\
P_z \\
1
\end{pmatrix}
\]
Verticals revisited …

Unlike the previous analysis we now can control the vertical

i.e. we can assume the $v$-direction is the vertical and constrain it in the software to be upwards
Back to flying sequences

• We now return to the original problem
  – Given a viewpoint point $C$ and a view direction $d$, we need to find the transformation matrix that gives us the canonical view.
  – We do this by first finding the vectors $u$, $v$ and $w$.

We know that $d$ is the direction of the new axis, so we can write immediately

$$w = \frac{d}{|d|}$$
Now the horizontal direction

- We can write $u$ in terms of some vector $p$ in the horizontal direction
  \[ u = \frac{p}{|p|} \]
- To ensure that $p$ is horizontal we set
  \[ p_y = 0 \]
- so that $p$ has no vertical component
And the vertical direction

• Let \( q \) be some vector in the vertical direction, we can then write \( v \) as

\[
\Rightarrow v = \frac{q}{|q|}
\]

• \( q \) must have a positive \( y \) component, so we can say that

\[
q_y = 1
\]
So we have four unknowns

\[ \mathbf{p} = [p_x, 0, p_z] \quad \text{new horizontal} \]

\[ \mathbf{q} = [q_x, 1, q_z] \quad \text{new vertical} \]

To solve for these we use the cross product and dot product.

We can write the view direction \( \mathbf{d} \), which is along the new \( z \) axis, as

\[ \mathbf{d} = \mathbf{p} \times \mathbf{q} \]

(We can do this because the magnitude of \( \mathbf{p} \) is not yet set)
Evaluating the cross-product

\[
d = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \mathbf{p} \times \mathbf{q} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p_x & 0 & p_z \\ q_x & 1 & q_z \end{vmatrix} \\
= -p_z \mathbf{i} + (p_z q_x - p_x q_z) \mathbf{j} + p_x \mathbf{k} = \begin{pmatrix} -p_z \\ p_z q_x - p_x q_z \\ p_x \end{pmatrix}
\]

\[
d_x = -p_z \\
d_y = p_z q_x - p_x q_z \\
d_z = p_x 
\]

So we can write vector \( \mathbf{p} \) completely in terms of \( \mathbf{d} \)

\[
\mathbf{p} = \begin{pmatrix} d_z \\ 0 \\ -d_x \end{pmatrix}
\]
Using the dot product

• Lastly we can use the fact that the vectors $\mathbf{p}$ and $\mathbf{q}$ are orthogonal

\[ \mathbf{p} \cdot \mathbf{q} = 0 \]

\[ \Rightarrow p_x q_x + p_z q_z = 0 \]

• And from the cross product (previous slide)

\[ d_y = p_z q_x - p_x q_z \]

• So we have two simple linear equations to solve for $\mathbf{q}$ and write it in terms of the components of $\mathbf{d}$
The final matrix

- Once we have expressions for \( p \) and \( q \) in terms of the given vector \( d \), we have

\[
\begin{align*}
u &= \frac{p}{|p|} \\
v &= \frac{q}{|q|} \\
w &= \frac{d}{|d|}
\end{align*}
\]

- We already know \( C \) as that is also given. So we can write down the matrix

\[
\begin{pmatrix}
u_x & \nu_y & \nu_z & -C \cdot u \\
v_x & \nu_y & \nu_z & -C \cdot v \\
w_x & w_y & w_z & -C \cdot w \\
0 & 0 & 0 & 1
\end{pmatrix}
\]