Administrative
Assignment Example
2D transformations (continued)

- The transformations discussed so far are invertible (why?). What are the inverses?
2D viewing (§5.5 and more)

- 3 Significant coordinate systems are usual
  - World coordinates or modeling coordinates - where the model is defined (meters, miles, etc.)
  - Normalized device coordinates; usually (0-1) in each variable.
  - Device coordinates: the actual coordinates of the pixels on the frame-buffer or the printer

- Main issue: constructing transformations between coordinate systems

- Terminology:
  - window = region on drawing that will be displayed (rectangle)
  - viewport = region in NDC’s/DC’s where this rectangle is displayed (often simply entire screen).
Element in modelling coordinates

Transform into Normalised Device Coordinates

Transform into Device Coordinates

Clip

Draw
• view this as a sequence of transformations in h.c.’s, then determine each element in closed form.

• compute numerically from point correspondences.
• write \((wx_i, wy_i)\) for coordinates of \(i\)'th point on window

• translation is:

\[
\begin{pmatrix}
    x' \\
    y' \\
    1
\end{pmatrix} =
\begin{pmatrix}
    1 & 0 & -\bar{wx} \\
    0 & 1 & -\bar{wy} \\
    0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    x \\
    y \\
    1
\end{pmatrix}
\]

(overbar denotes average over vertices, i.e., 1,2,3,4)
Rotate to line up with axes

\[
\begin{pmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
\]

(Need to compute theta)
\begin{align*}
\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} &= \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\end{align*}

(Vertex order does not correspond, need to flip)
Scale and translate

\[
\begin{pmatrix}
\begin{array}{ccc}
\frac{w_{\text{new}}}{w_{\text{old}}} & 0 & x_{\text{new}} \\
0 & \frac{h_{\text{new}}}{h_{\text{old}}} & y_{\text{new}} \\
0 & 0 & 1
\end{array}
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
= 
\begin{pmatrix}
x' \\
y' \\
1
\end{pmatrix}
\]

Notice that choice of new width, height, and center give translation to either normalized device coords, or to device coordinates.
• Get overall transformation by multiplying transforms.
• This gives a single transformation matrix, whose elements are functions of window/viewport coordinates.
• Notice notational advantage of homogeneous coordinates - no extra vectors hanging around.

\[ x' = \begin{bmatrix} M_{\text{translate origin to viewport cog, scale}} & M_{\text{flip}} & M_{\text{rotate}} & M_{\text{translate window cog->origin}} \end{bmatrix} x \]

NDC’s/DC’s                  World coords

(cog==window center of gravity)
Affine transformations

• Another approach to determining the whole transform for the pipeline; this is an affine transform.

• Matrix form:

\[
\begin{pmatrix}
a & b & c \\
d & e & f \\
0 & 0 & 1
\end{pmatrix}
\]

• Now assume that I know that \( M p_1 = q_1, M p_2 = q_2, M p_3 = q_3 \)

• Quick way to determine transform, because this is the same as six linear equations, in six variables, which are the entries in the matrix:

\[
\begin{pmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_1 & y_1 & 1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_2 & y_2 & 1 \\
x_3 & y_3 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_3 & y_3 & 1
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
c \\
d \\
e \\
f
\end{pmatrix}
= 
\begin{pmatrix}
u_1 \\
v_1 \\
u_2 \\
v_2 \\
u_3 \\
v_3
\end{pmatrix}
Details

- $\text{Mp}_1 = q_1$ gives first two rows
- $p_1 = (x_1, y_1, 1)^T$, $q_1 = (u_1, v_1, 1)^T$

\[
\begin{pmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  y_1 \\
  1
\end{pmatrix}
= 
\begin{pmatrix}
  u_1 \\
  v_1 \\
  1
\end{pmatrix}
\]

\[ax_1 + by_1 + c = u_1\]
\[dx_1 + ey_1 + f = v_1\]

$\text{Mp}_2 = q_2$, $\text{Mp}_3 = q_3$ give other rows
Hierarchical modeling

Consider constructing a complex 2d drawing: e.g. an animation showing the plan view of a building, where the doors swing open and shut.

Options:
- specify everything in world coordinate frame; but then each room is different, and each door moves differently. (hugely difficult).
- Exploit similarities by using repeated copies of models in different places (instancing)

Each arrow represents a transformation.
Hierarchical modeling

- Model form
  - Directed acyclic graph.
  - Each node consists of 0 or more objects (lines, polygons, etc).
  - Each edge is a transformation
- Notice there can be many edges joining two nodes (e.g. in the case of the corridor - many copies of the same room model, each transformed differently).
- Every graphics API supports hierarchies - some directly (meaning you have to learn a language to express the model) some indirectly with a matrix stack

- Write the transformation from door coordinates to room coordinates as:
  \[ T_{door} \]
  Then to render a door, use the transformation:
  \[ T_{world} T_{corridor} T_{room} T_{door} \]
  To render a body, use the transformation:
  \[ T_{world} T_{corridor} T_{room} T_{body} \]
Matrix stacks and rendering

• Matrix stack:
  – stack of matrices used for rendering
  – applied in sequence.
  – Pop=remove last matrix
  – Push=append a new matrix
  – in previous example, body-device transformation comes from door-device transformation by popping door-room and pushing body-room

• Algorithm for rendering a hierarchical model:
  – stack is $T_{\text{device}}^{\text{root}}$
  – render (root)

• Render (node):
  – for each child:
    • push transformation
    • render (child)
    • pop transformation
Now to render door on first room in first corridor, stack looks like: W C1 R1 D1

Note that we do not need two copies of corridor, or 16 copies of body; we render one copy using 16 different transformations. This is known as instancing.

Animation requires care: if D1 is a single function of time, all doors will swing open and closed at the same time.
- Stack is W
- render kneecap
- Stack is W L
- render ankle
- Stack is W L F
- render foot
- Stack is W L S
- render shin
- Stack is W T
- render thigh
Transformations in 3D

- Right hand coordinate system (conventional, i.e., in math)
- In graphics a LHS is sometimes also convenient (Easy to switch between them--later).
Transformations in 3D

• Homogeneous coordinates now have four components - traditionally, $(x, y, z, w)$
  – ordinary to homogeneous: $(x, y, z) \rightarrow (x, y, z, 1)$
  – homogeneous to ordinary: $(x, y, z, w) \rightarrow (x/w, y/w, z/w)$

• Again, translation can be expressed as a multiplication.
Transformations in 3D

- Translation:

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  1 & 0 & 0 & tx \\
  0 & 1 & 0 & ty \\
  0 & 0 & 1 & tz \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]
3D transformations

- Anisotropic scaling:

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1 
\end{pmatrix} \rightarrow \begin{pmatrix}
  sx & 0 & 0 & 0 \\
  0 & sy & 0 & 0 \\
  0 & 0 & sz & 0 \\
  0 & 0 & 0 & 1 
\end{pmatrix} \begin{pmatrix}
  x \\
  y \\
  z \\
  1 
\end{pmatrix}
\]

- Shear (one example):

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1 
\end{pmatrix} \rightarrow \begin{pmatrix}
  1 & 0 & a & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1 
\end{pmatrix} \begin{pmatrix}
  x \\
  y \\
  z \\
  1 
\end{pmatrix}
\]
Rotations in 3D

• 3 degrees of freedom
• Det(R)=1
• Orthogonal
• Many representations are possible.
• Our representation: rotate about coordinate axes in sequence.
• Sequence of axes is arbitrary, but choice does affect the angles used (cannot use same angles with different order).
• Convention: look down coordinate axis (towards origin), anticlockwise rotation is positive angle.
• Likely easier way to remember is the following Right Hand Rule--point thumb along axis in direction of increasing ordinate--then fingers curl in the direction of positive rotation).
Rotations in 3D

• About z-axis

\[ M = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 
\end{bmatrix} \]
Finding a Normal Vector

A vector normal to the plane of \( \vec{P} \) and \( \vec{Q} \) is the cross (vector) product of \( \vec{P} \) and \( \vec{Q} \). No 2D analog.

Direction is into the page. (Right hand rule--if \( \vec{P} \) is thumb, \( \vec{Q} \) is index finger, the \( \vec{P} \times \vec{Q} \) is middle finger.

Formula: \((x_1, y_1, z_1) \times (x_2, y_2, z_2) = (y_1z_2 - z_1y_2, z_1x_2 - z_2x_1, x_1y_2 - y_1x_2)\)