2D viewing

- Three coordinate systems are common in graphics
  - 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
- Need to construct 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).
• View this as a sequence of transformations in homogenous coords, 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:

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

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

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

(Need to compute theta)
\[ \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \]

(VerteX order does not correspond, need to flip)
Notice that choice of new width, height, and center give translation to either normalized device coords, or to device coordinates.

\[
\begin{bmatrix}
\frac{x'}{w_{new}} & \frac{y'}{w_{old}} & 0 \\
0 & \frac{h_{new}}{h_{old}} & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
• Get overall transformation by multiplying transforms.
• This gives a single transformation matrix, whose elements are functions of window/viewport coordinates.

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

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:

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

- Now assume that we 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:

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

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

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

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

$M_{p_2} = q_2$, $M_{p_3} = q_3$ 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 (instanting)
Hierarchical modeling

- **Model form**
  - Directed acyclic graph.
  - Each node consists of 0 or more objects (lines, polygons, etc).
  - Each edge is a transformation

- 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:**

\[ T_{\text{door}}^{\text{room}} \]

Then to render a door, use the transformation:

\[ T_{\text{device}}^{\text{world}} T_{\text{corridor}}^{\text{floor}} T_{\text{corridor}}^{\text{room}} T_{\text{room}}^{\text{corridor}} T_{\text{door}}^{\text{room}} \]

To render a body, use the transformation:

\[ T_{\text{device}}^{\text{world}} T_{\text{corridor}}^{\text{floor}} T_{\text{corridor}}^{\text{room}} T_{\text{room}}^{\text{corridor}} T_{\text{body}}^{\text{room}} \]
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 $T_{\text{root}}$ device
  – 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
• For efficiency we would store “running” products, IE, the stack contains: W, W*C1, W*C1*R1, W*C1*R1*D1.
• 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 (Watt chapter 1)

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

\[ \begin{align*}
  \text{y} \\
  \text{x} \\
  \text{z (out of page)}
\end{align*} \]

\[ \begin{align*}
  \text{y} \\
  \text{z (into of page)} \\
  \text{x}
\end{align*} \]
Transformations in 3D

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

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

- Translation:

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

- Anisotropic scaling:

| x  | sx | 0 | 0 | 0 | x |
| y  | 0  | sy| 0 | 0 | y |
| z  | 0  | 0 | sz| 0 | z |
| 1  | 0  | 0 | 0 | 1 | 1 |

- Shear (one example):

| x  | 1  | 0 | a | 0 | x |
| y  | 0  | 1 | 0 | 0 | y |
| z  | 0  | 0 | 1 | 0 | z |
| 1  | 0  | 0 | 0 | 1 | 1 |
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).
- Sign of rotation follows the 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

\[
M = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
Rotation about an arbitrary axis (likely to be needed for an assignment!)
Rotation about an arbitrary axis

Strategy--rotate A to Z axis, rotate about Z axis, rotate Z back to A.
Rotation about an arbitrary axis

Tricky part:
rotate A to Z axis

Two steps.
1) Rotate about x to xz plane
2) Rotate about y to Z axis.
Rotation about an arbitrary axis (assignment hint)

Tricky part:
rotate $A$ to $Z$ axis

Two steps.
1) Rotate about $X$ to $xz$ plane
2) Rotate about $Y$ to $Z$ axis.