Chapter 3

The Geometry of Virtual Worlds

3.1 Geometric models

- 3D Euclidean space w/ Cartesion coordinates
  - let R^3 denote real world, using (x,y,z)
- **Data Structures**
  - Geometric models usually encoded in clever data structures
    - **Doubly connected edge list** aka **Half-edge data structure**
      - Three kinds of data elements: *faces*, *edges*, and *vertices*
      - represent 2, 1, and 0-dimensional parts of model

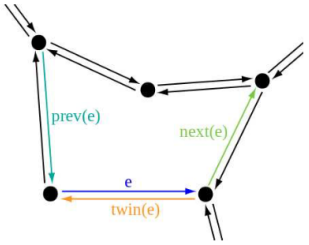


Figure 1:

Figure 3.3: Part of double connected edge list shown for face w/ five edges on boundary. Each half-edge structure *e* stores pointers to the next and prev edges along face boundary. Also stores pointer to its twin half-edge, which is part of boundary of adjacent face)

- **Inside vs. outside**
  - *Q: is object interior part of model?*
  - **Coherent model**: If model inside were filled w/ gas, could not leak
  - **Polygon soup**: Jumble of triangles that do not fit together nicely, could even have intersecting interiors
- **Why triangles?**
  - Triangles used because simplest for algorithms
- **Stationary vs. movable models**
  - Two kinds of models:
  1. **Stationary models**: keep same coordinates forever
     - ex: streets, floors, buildings
  2. **Movable models**: can be *transformed* into various positions and orientations
     - ex: vehicles, avatars
  - Motion can be caused either by
  1. tracking system (model match user's motions)
  2. controller
  3. laws of physics in virtual world
- **Choosing coordinate axes**
  - Don't be stupid.
- **Viewing the models**
  - *Q: How is model going to "look" when viewed on display?*
  - Two parts:
  1. Determining where points in virtual world should display
  2. How model should appear after lighting sources and surface properties defined in virtual world

3.2 Changing Position and Orientation

- Suppose movable model defined as mesh of triangles. To move, *apply single transformation to every vertex of every triangle*
- **Translations**
  - Consider triangle: $((x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3))$
  - Let x_t, y_t, z_t be amount we want to change triangle's position. **Translation** given by:

$$(x_1, y_1, z_1) \rightarrow (x_1 + x_t, y_1 + y_t, z_1 + z_t)$$
$$(x_2, y_2, z_2) \rightarrow (x_2 + x_t, y_2 + y_t, z_2 + z_t)$$
$$(x_3, y_3, z_3) \rightarrow (x_3 + x_t, y_3 + y_t, z_3 + z_t)$$

  in which $a \rightarrow b$ denotes a becomes replaced by b after transformation is applied
- **Relativity**

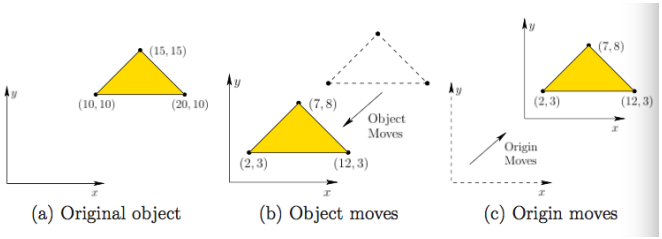


Figure 2:

Figure 3.4: Every transformation has two possible interpretations

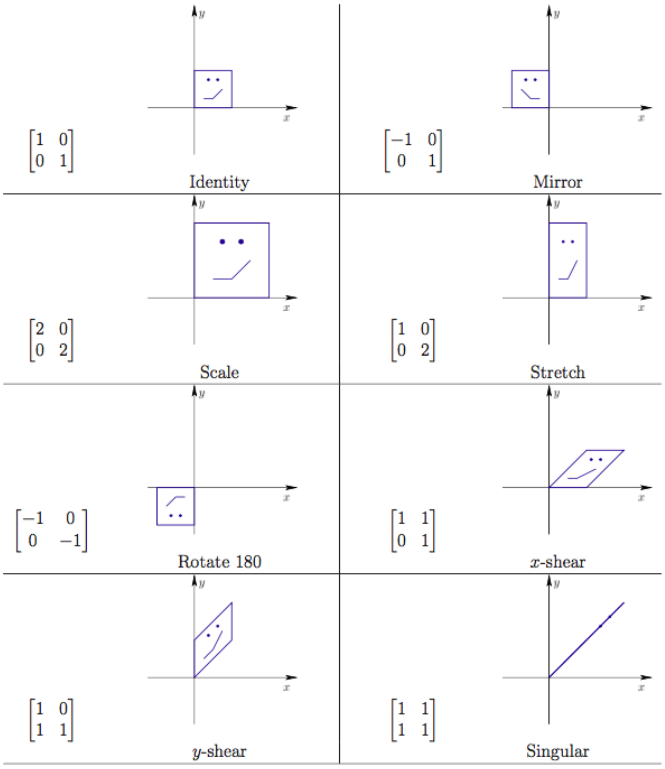
  - Transforming the coordinate axes results in an *inverse* of transformation that would correspondingly move the model.
  - *Relativity*: Did the object move, or did the whole world move around it?
    - If we perceive ourselves as having moved, VR sickness can increase
- **Getting ready for rotations**
  - Operation that changes **orientation** is called **rotation**
  - Simpler problem: 2D linear transformations
    - Consider a 2D virtual world with coordinates (x, y)

$$Let\ M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

  Performing multiplcation of matrix and point (x, y), we get:

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$
$$x' = m_{11}x + m_{12}y$$
$$y' = m_{21}x + m_{22}y$$

  in which (x', y') is the transformed point. Therefore, M is transformation for $(x, y) \rightarrow (x', y')$
- **Applying the 2D matrix to points**



1. Multiplying by the identity matrix does not transform
2. Mirrors over y axis, $(x, y) \to (-x, y)$
3. Scales to double the size, $(x, y) \to (2x, 2y)$
4. Stretches as only y is doubled, $(x, y) \to (x, 2y)$
   - **aspect ratio** distortion
5. Rotate 180°, as both x and y are mirrored, $(x, y) \to (-x, -y)$
6. Shear along x-axis, amount of displacement dependant on y, $(x, y) \to (x + y, y)$
7. Shear along y-axis, amount of displacement dependant on x, $(x, y) \to (x, x + y)$
8. Two-dimensional shape collapses into one dimension, $(x, y) \to (x + y, x + y)$
   - case of a **singular matrix**:
     - columns are not linearly independant
     - singular iff determinant $= 0$

- **Only some matrices produce rotations**
  - M must satisfy rules to be a rotation:

1. No stretching of axes
2. No shearing
3. No mirror images

  - Rule 1: columns of M ust have unit length

$$m_{11}^2 + m_{21}^2 = 1 \text{ and } m_{12}^2 + m_{22}^2 = 1$$

  - Rule 2: dot product must equal 0

$$m_{11}m_{12} + m_{21}m_{22} = 0$$

  - Rule 3: determinant of M equals 1

$$det \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = m_{11}m_{22} - m_{12}m_{21} = 1$$

- Using the unit circle, $x = \cos\theta$ and $y = \sin\theta$, M becomes

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- Note: reduced *degrees of freedom* from four to one, θ

- **The 3D case**
  - M is extended from 2D to 3D

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

  - Start with nine degrees of freedom
  - First two rules each remove three DOF, last rule doesn't remove any
  - Therefore, left with three degrees of rotational freedom

- **Yaw, pitch, and roll**

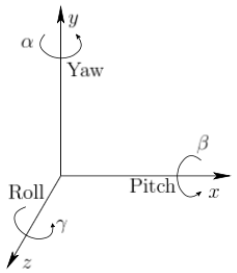


Figure 3:

Figure 3.7: Any 3D rotation can be described as sequence of yaw, pitch, and roll rotations

Let *roll* be counterclockwise rotation of γ about the z-axis.

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Let *pitch* be counterclockwise rotation of β about x-axis.

$$R_x(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix}$$

Let *yaw* be counterclockwise rotation of α about the y-axis.

$$R_x(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

- **Combining rotations**
  - Yaw, pitch, and roll rotations can be combined to make any possible 3D rotation

$$R(\alpha, \beta, \gamma) = R_y(\alpha)R_x(\beta)R_z(\gamma)$$

  - Note: ranges of α and γ are from 0 to 2π, but β only needs to range from $-\pi/2$ to $\pi/2$
  - Operations **ARE NOT COMMUNICATIVE**

- **Matrix multiplications are "backwards"**
  - Suppose we have point p and rotation matrices R and P
  - Let $p' = Rp$
  - Let $p'' = Qp'$
  - Note: p'' **DOES NOT** equal RQp. Instead, $p'' = QRp$

- **Translation and rotation in one matrix**
  - If we want to apply rotation R then translate by (x_t, y_t, z_t)

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}$$

  - This is impossible to fit in 3x3 matrix, but can be done in 4x4 **homogeneous transformation matrix**

$$T_{rb} = \begin{bmatrix} & & & x_t \\ & R & & y_t \\ & & & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4:

  - T_{rb} used for **rigid body transformation** (does not distort objects)
  - Because of extra dimension, we extend (x, y, z) to $(x, y, z, 1)$
  - Note: This is for rotation *followed by* translation. **NOT COMMUNICATIVE**