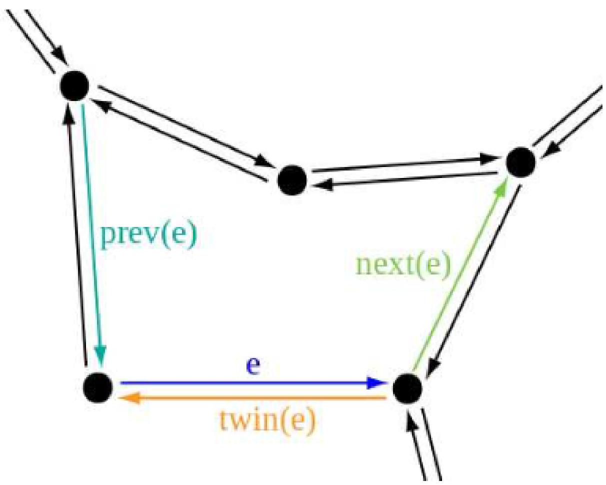
Chapter 3

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 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

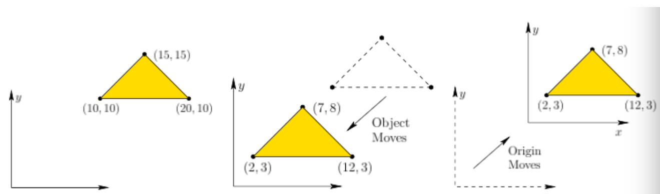
- **Choosing coordinate axes**
- Don't be stupid.
- **Viewing the models**
- *Q: How is model going to "look" when viewed on display?*
- Two parts:
    1. Determining where points in virtual world should display
    2. How each part of model should appear after lighting sources and surface properties 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:

$$
\begin{aligned}
(x_1, y_1, z_1) &\to (x_1 + x_t, y_1 + y_t, z_1 + z_t) \\
(x_2, y_2, z_2) &\to (x_2 + x_t, y_2 + y_t, z_2 + z_t) \\
(x_3, y_3, z_3) &\to (x_3 + x_t, y_3 + y_t, z_3 + z_t)
\end{aligned}
$$

in which $a \to b$ denotes a becomes replaced by b after transformation is applied

- **Relativity**
- 

(a) Original object (b) Object moves (c) Origin moves
    - 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**
-