
CS770/870 Fall 2008

3D Viewing

Related material in Hill and Kelley: Ch. 5.6 and 7

Preview

- Scene description
 - Hierarchical object modeling
- Scene rendering
- Projections
 - Parallel projection
 - Perspective projection
- Synthetic camera model
- OpenGL Viewing

Scene Description

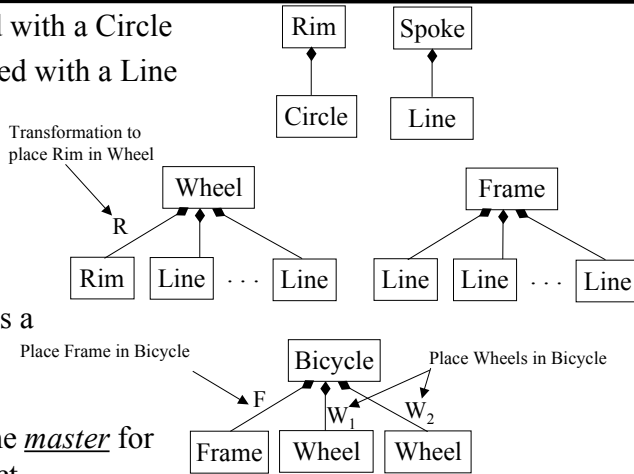
- A scene to be rendered is defined in a world coordinate system -- arbitrary floating point
- A scene is composed of multiple objects
 - Each object is defined (modeled) in its own object coordinate system
- Object coordinates must be mapped to world coordinates in order to be placed in the scene

Object Definition (Modeling)

- Objects are defined in object coordinates
 - usually the object is centered around the origin of the object coordinate system
- An object can be defined in terms of other objects (actually, instances of other objects)
 - the component object must be placed in the coordinate system of the composite object
 - requires mapping one object coordinate system to another

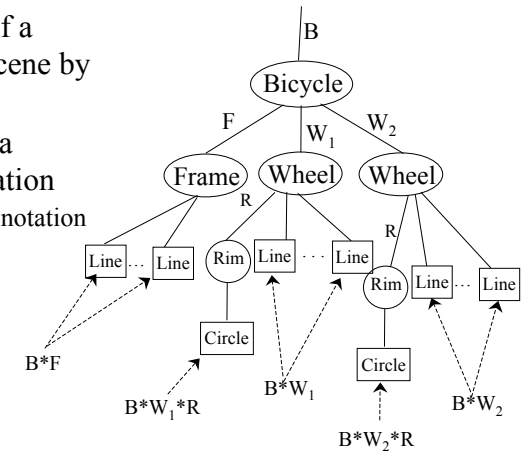
Composite Object Definition

- A Rim is modeled with a Circle
- A Spoke is modeled with a Line
- A Wheel contains
 - a Rim
 - 10 Spokes
- A Frame contains
 - a bunch of Lines
- A Bicycle contains a
 - Frame
 - 2 Wheels
- These represent the *master* for each class of object

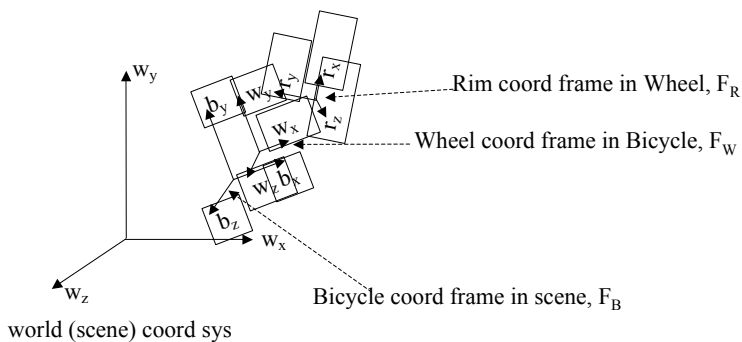


Composite Object Instance

- Represent an *instance* of a Bicycle, placed in the scene by matrix B
- Each component needs a transformation specification
 - We label the arcs with a notation for the matrix
- What is the composite matrix for each set of primitives?



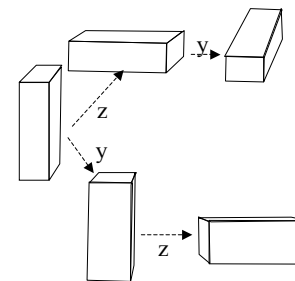
Composition as Change of Coordinate Frame



- To map the Rim's circle to the world, coordinates of the circle will be transformed by: $F_B F_W F_R = BWR$
- So, the modeling transforms are the coordinate frame transforms

Are 3D Rotations Symmetric?

- Given two affine transforms, M_1 and M_2
 - Does $M_1 * M_2 = M_2 * M_1$?
 - try 90° rotation about y and z.



Rendering a Scene

- Fixed view model
 - All rendering is based on a fixed view (one for parallel projection, one for perspective projection)
 - The user application transforms the objects in the scene so the desired portion of the scene is located in the view
- Variable view model
 - User defines a (more or less) arbitrary view
 - The graphics system transforms the scene to the viewing coordinate system

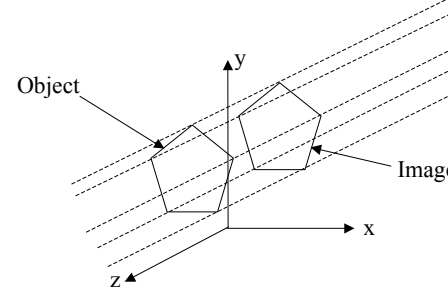
17:18

CS770/870 Fall 2008 Bergeron/Plumlee

9

Parallel Projection

- Object coordinates are “projected” using parallel projectors onto a 2D plane
- A rectangular portion of that plane (the clip window) is mapped to the display window



A common fixed parallel projection view:

- view plane: xy plane
- view plane normal = (0 0 1)
- window: (0,0) to (1,1)

Makes projection and clipping very easy:

- Ignore z
- map clip window to display:
 $d_x = (\text{int}) w_x * \text{windowWidth}$
 $d_y = (\text{int}) (1-w_y) * \text{windowHeight}$

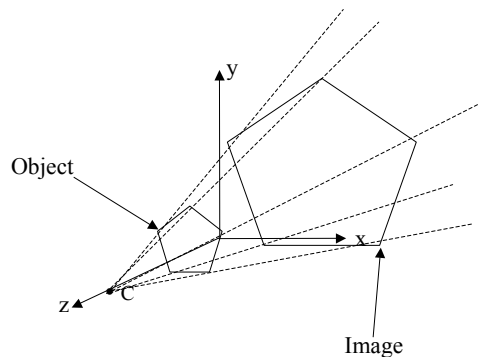
17:18

CS770/870 Fall 2008 Bergeron/Plumlee

10

Perspective Projection

- Projectors originate at a center of projection; intersection with the projection plane yields 2D point



One common fixed perspective view:

- view plane: xy plane
- view plane normal = (0 0 1)
- window: (-1,-1) to (1,1)

Makes projection and clipping very easy:

- See next slide
- map clip window to display:
 $d_x = (1 + w_x) * \text{displayW} / 2$
 $d_y = (1 - v_y) * \text{displayH} / 2$

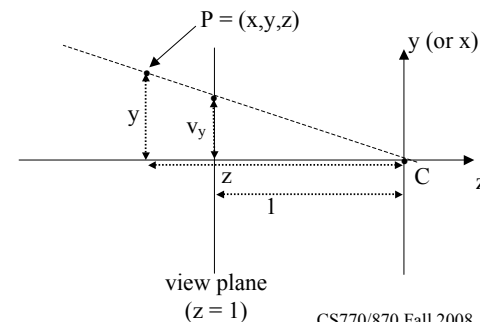
17:18

CS770/870 Fall 2008 Bergeron/Plumlee

11

Perspective Projection Example

- Simple projection specification yields simple computations:
 - view plane normal: (0 0 1)
 - view plane origin: (0 0 1)
 - center of projection: $C = (0 0 0)$
 - window bounds: (-1,1) to (1,1)



1. Project $P=(x,y,z)$ onto view plane: by symmetric triangles:

$$v_y / y = 1 / z$$

i.e., $v_y = y / z$
 and $v_x = x / z$
 where v_x, v_y are *view plane* coordinates

2. Map clip window to display:

$$d_x = (1+v_x) * \text{displayW} / 2$$

$$d_y = (1-v_y) * \text{displayH} / 2$$

17:18

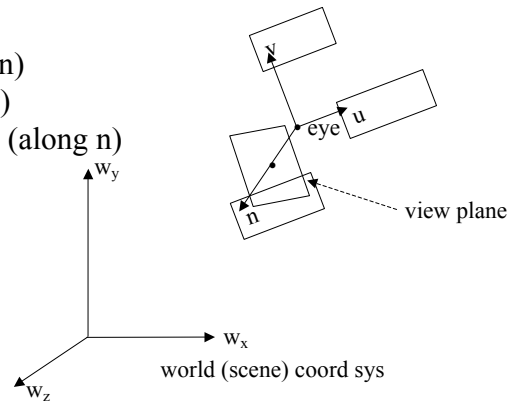
CS770/870 Fall 2008 Bergeron/Plumlee

12

Synthetic Camera Model

- Allow user to define a view coordinate system in world coordinates:

- eye point
- view plane normal (n)
- view up direction (v)
- view plane location, (along n)
- clipping window in view plane



17:18

CS770/870 Fall 2008 Bergeron/Plumlee

13

OpenGL Synthetic Camera

- OpenGL's version of the synthetic camera is (mostly) represented by the `gluLookAt` function


```
gluLookAt( eye.x, eye.y, eye.z,
            look.x, look.y, look.z,
            up.x, up.y, up.z )
```

 - **eye**: is the origin of the viewing coordinate system in world coords
 - **look**: viewer is looking *at* the look point in world coords
 n -axis of viewing coordinate system = $eye - look$
 - **up**: any line parallel to up (in world coords) is vertical in the image
 v -axis of viewing coord system is computed from n and up
 u -axis of vcs is computed from n and v .
- Also need a projection specification (in view coordinates):
`glFrustum(left, right, bottom, top, front, near)`

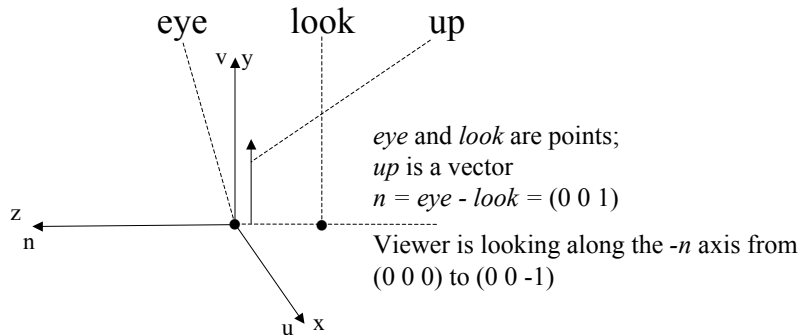
17:18

CS770/870 Fall 2008 Bergeron/Plumlee

14

gluLookAt

- Default : (0, 0, 0, 0, 0, -1, 0, 1, 0)



17:18

CS770/870 Fall 2008 Bergeron/Plumlee

15

OpenGL Orthographic Projection

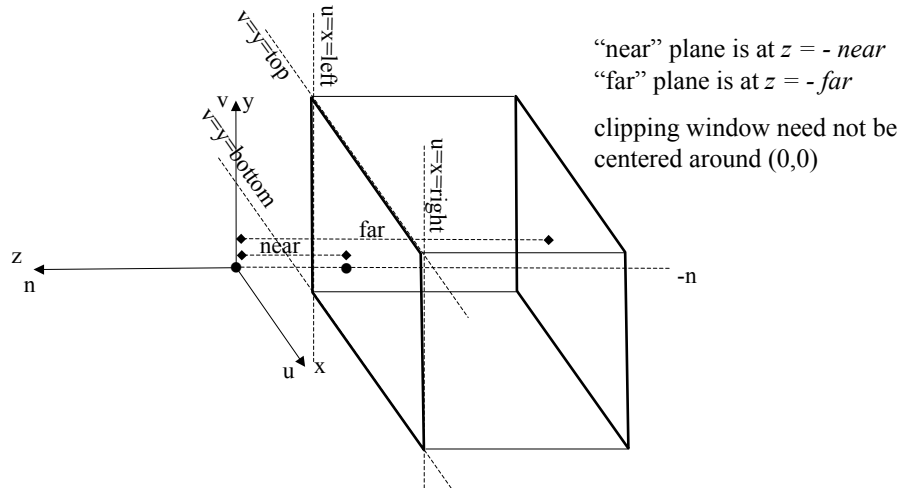
- The simplest OpenGL view is orthographic: `glOrtho(left, right, bottom, top, near, far)`
 - parameters are in viewing coordinates
 - *left, right, bottom, top* define the bounds of the 2D window in the projection plane
 - *left* and *right* are measured on the u -axis
 - *bottom* and *top* are measured on the v -axis
 - *near, far* are the distances of the near and far clipping planes from (0,0,0) along the $-n$ axis!

17:18

CS770/870 Fall 2008 Bergeron/Plumlee

16

OpenGL Default Orthographic View



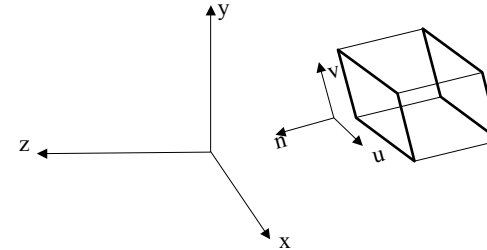
17:18

CS770/870 Fall 2008 Bergeron/Plumlee

17

“Arbitrary” OpenGL View

- *gluLookAt* (and *glOrtho*) can place the viewing coordinate system (frame) anywhere



- OpenGL needs to transform world coords to viewing coords

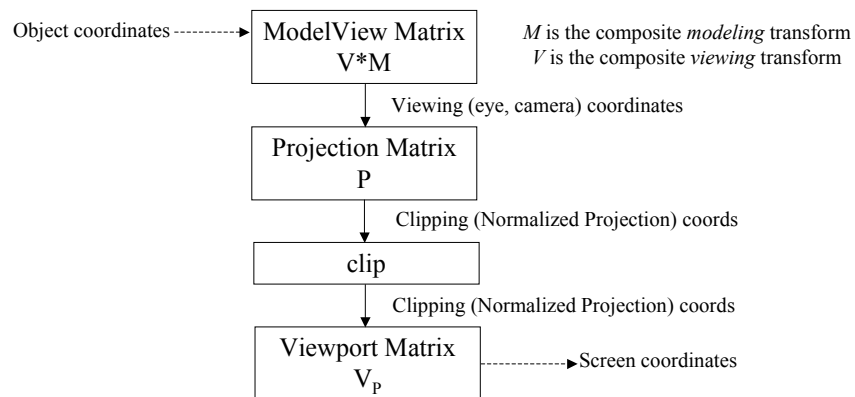
17:18

CS770/870 Fall 2008 Bergeron/Plumlee

18

OpenGL Viewing Pipeline

- Viewing pipeline transforms scene to screen



17:18

CS770/870 Fall 2008 Bergeron/Plumlee

19

Review

- Scene description
 - Hierarchical object modeling
- Scene rendering
- Projections
 - Parallel projection
 - Perspective projection
- Synthetic camera model
- OpenGL Viewing

17:18

CS770/870 Fall 2008 Bergeron/Plumlee

20