UCLA

CS174A : Introduction to Computer Graphics

Royce 190
TT 4-6pm

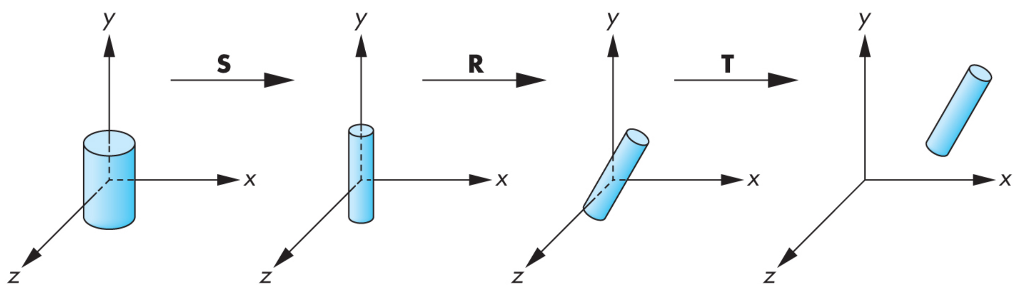
Scott Friedman, Ph.D
UCLA Institute for Digital Research and Education

UCLA

Representing models

- So far we have been rendering models more or less resembling the following.

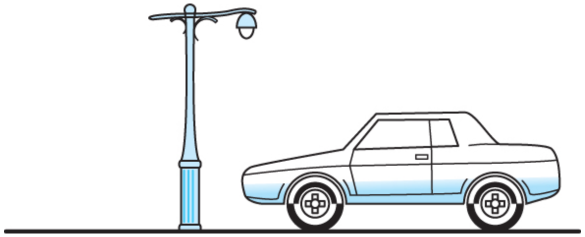
```
mat4 instance;
mat4 modelViewBase;

instance = Trans( x, y, z ) * Rx( rx ) * Ry( ry ) * Rz( rz ) * Scale( sx, sy, sz );
mat4 modelView = modelViewBase * instance;
modelViewToShader( modelView );
drawCylinder( );

Repeat...
```



UCLA

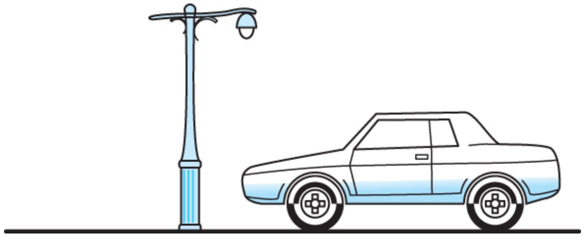
Representing models

- As the number of objects to render increases.
    - This approach quickly becomes unmanageable.
- We also typically render objects that have some relationship to each other.
    - Take the example of a car from the book.
    - Wheels in relation to the chassis.
    - Chassis in relation to the road and light post.



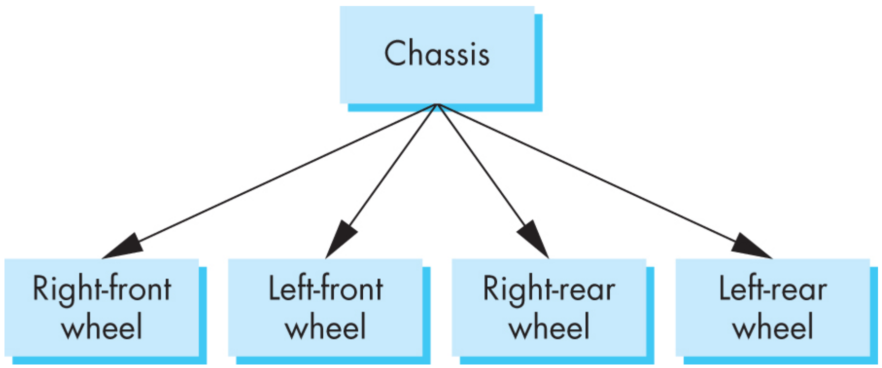
Representing models

- Representing all this in straight code will lead to a giant mess.
  - Each element has its own copy of transformations.
  - A lot of unnecessary duplication
- There is a better way.



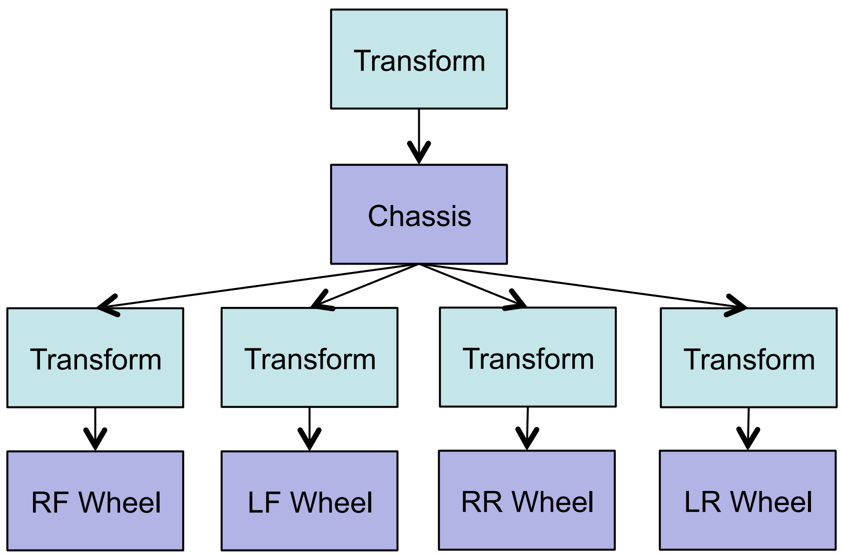
Representing models

- Scene graphs are a better way
  - Here objects can be represented in a logical way
  - Object relationships can be established in relation to other objects in a hierarchical way.
  - Useful as the wheels can now be specified in relation to the chassis, not the whole world.



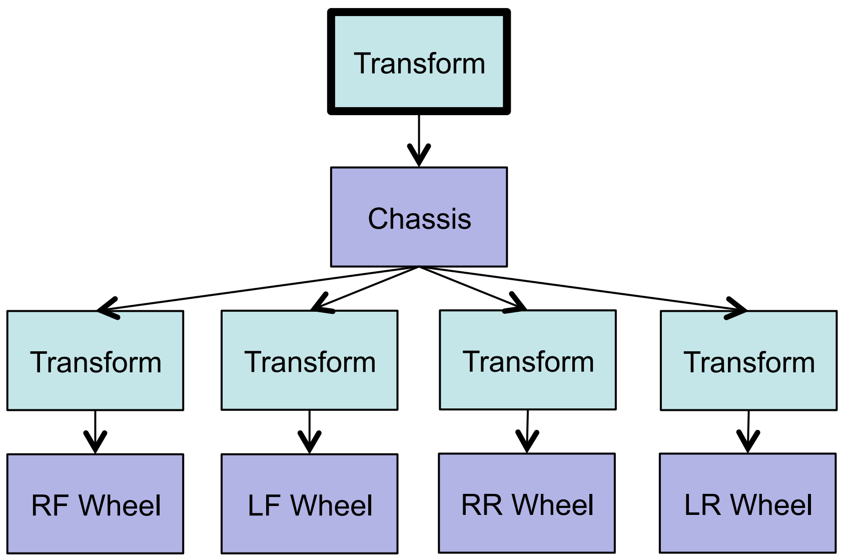
Representing models

- More often the graph (or tree) would look like the following.
  - Each transform node contains a full TRS transformation
    - TRS = Translate, Rotate, Scale



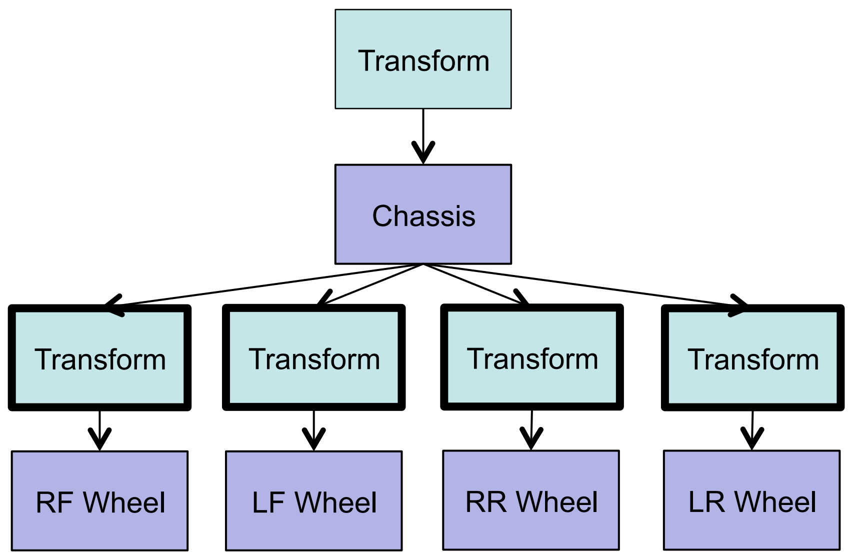
Representing models

- Moving, sizing, orienting the entire car is as simple as manipulating the top transform node.



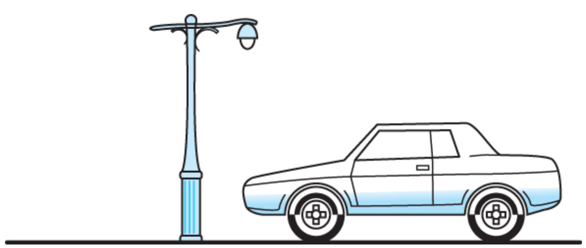
Representing models

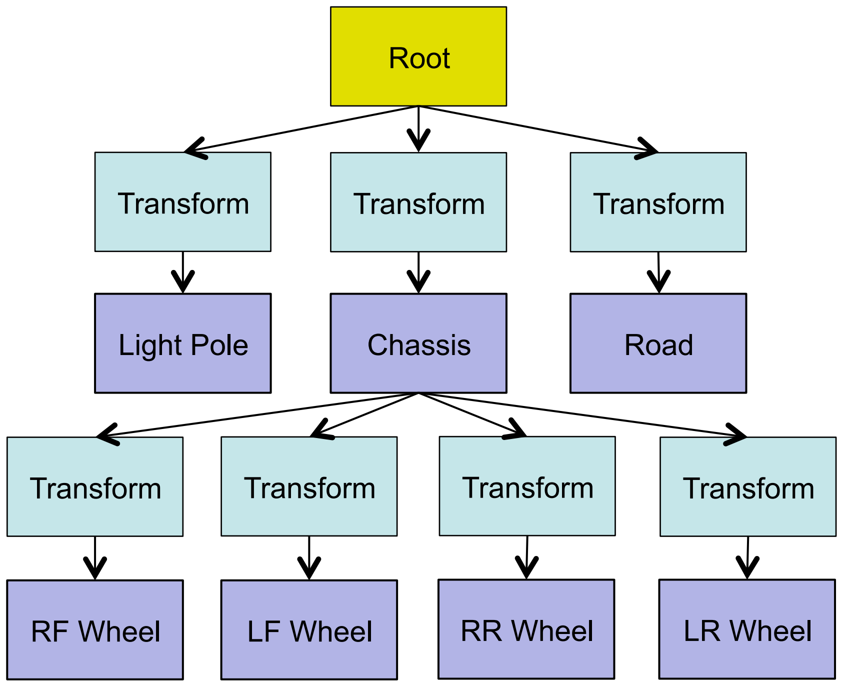
- These transforms are in relation to only the chassis itself.
    - Really, in the coordinate frame of the chassis.



Representing models

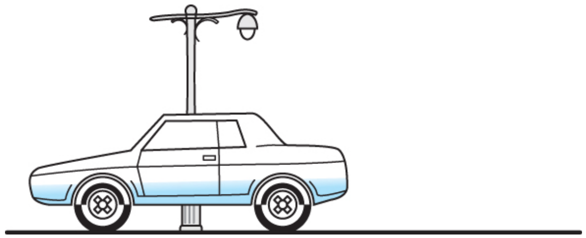
- Let's make the scene graph complete
  - Recall our little scene.

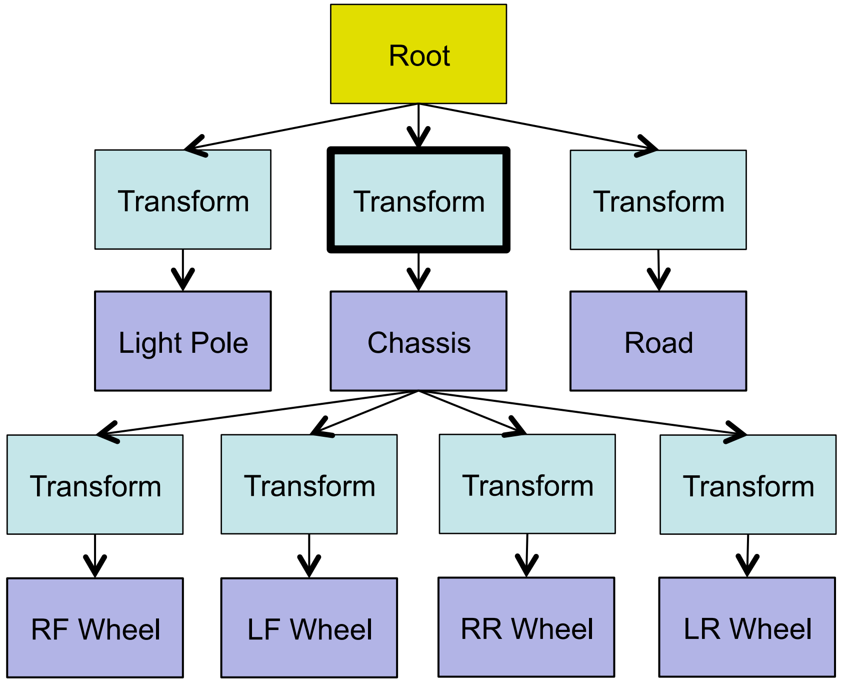




Representing models

- If the car is moved/animated
  - We update the single transformation.





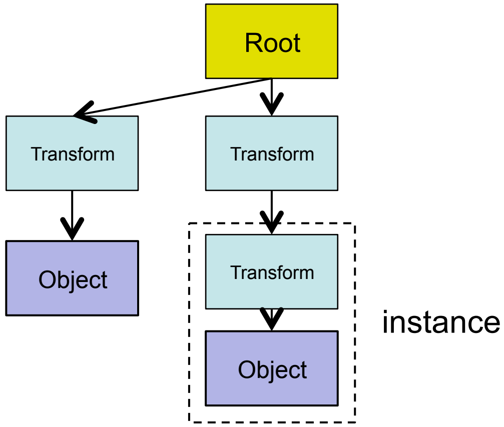
UCLA

Representing models

- How do we process a scene graph?
  - Traverse the tree structure – depth first.
  - Transformations are processed like a stack as we descend into the tree.
  - Pushing (saving) the current transformation on the way down.
  - Popping (restoring) the previous transformation on the way up.
  - There are variations on the exact representation.
    - But the stack metaphor is at its base.
  - You may want to track other application/graphics state in this way as well.

Representing models

- Elements of a scene graph?
  - In this example transformations are kept separate from the geometry.
    - Just my preference based on my experience.
    - This approach gives more flexibility, for example.
    - You could ‘flatten’ the scene by combining transform nodes
    - Could even apply transformations to static geometry.



Representing models

- Elements of a scene graph?
  - So far we have seen three types of nodes.
    - Root
    - Transform
    - Object
  - Other types of nodes can be imagined
    - Group
      - » Common parent
    - Attribute
      - » Set graphics state (e.g. texture)
    - Predicates or switches
      - » Only render a particular child of the node based on some rule or index.

UCLA

Representing models

- Elements of a scene graph?
  - Each node type would define some predefined behavior and possibly callbacks for application specific functionality.
  - Let's look at some basic nodes types you would see in a scene graph library (or if you implemented a simple one)

Representing models

- Elements of a scene graph?
  - Root node
    - Parent of all nodes in the scene graph
    - Global state about the scene might be kept here
      - Lighting is an example.
    - You might think storing camera information would go here but it generally does not.
      - The scene graph represents the scene alone.

Representing models

- Elements of a scene graph?
  - Group node
    - Groups children nodes/graphs.
    - Usually just a tree structuring element.
    - Often also a base class for many other node types.

UCLA

Representing models

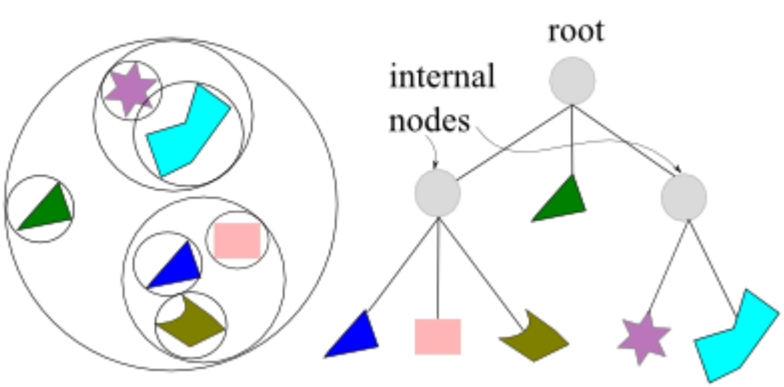
- Elements of a scene graph?
  - Transformation node
    - Usually based on a Group node.
    - Applies a transformation to all child nodes.
    - In some scene graph APIs there are two versions
      - Dynamic, which a transforms that change over time
      - Static, which do not – these often had the option of being flattened. This mattered when processors were much slower.

Representing models

- Elements of a scene graph?
  - Object node.
    - Stores geometry for an object.
    - A simple scene graph could store references to everything needed to draw.
    - Often, what is actually stored are pointers to data structures that represent the data.
      - Vertices, color, normals, texture coordinates, textures, shaders, etc.

Representing models

- Elements of a scene graph?
  - Most scene graphs also compute a ***bounding volume*** for each node of the tree.
  - A bounding sphere for all group based nodes is common.
    - Root, group, transform, etc.
    - We talked about computing bounding volumes last time.



UCLA

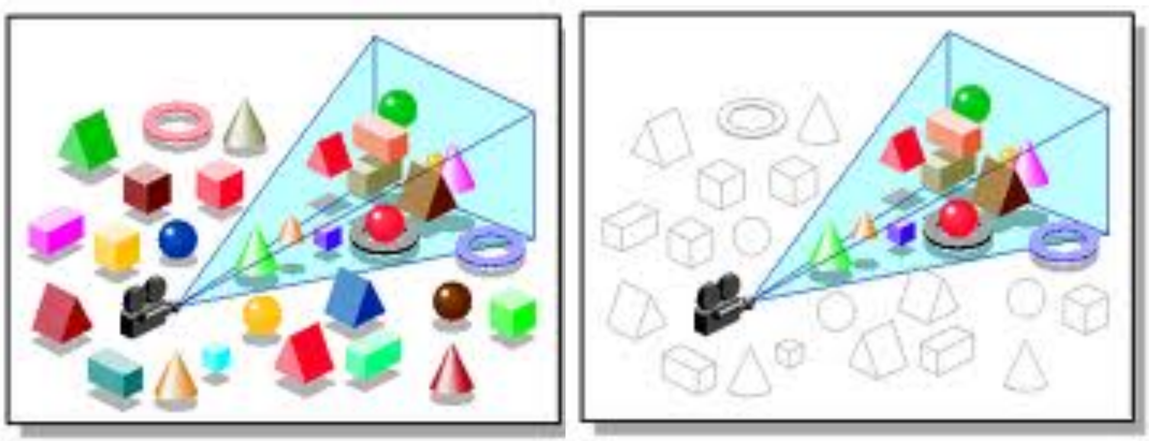
Representing models

- Elements of a scene graph?
  - Object nodes that hold geometry usually use some tighter fitting bounding volume.
    - These ***bounding volumes*** are used for ***culling*** the scene graph
    - Can also be used for collision detection with the scene.

Representing models

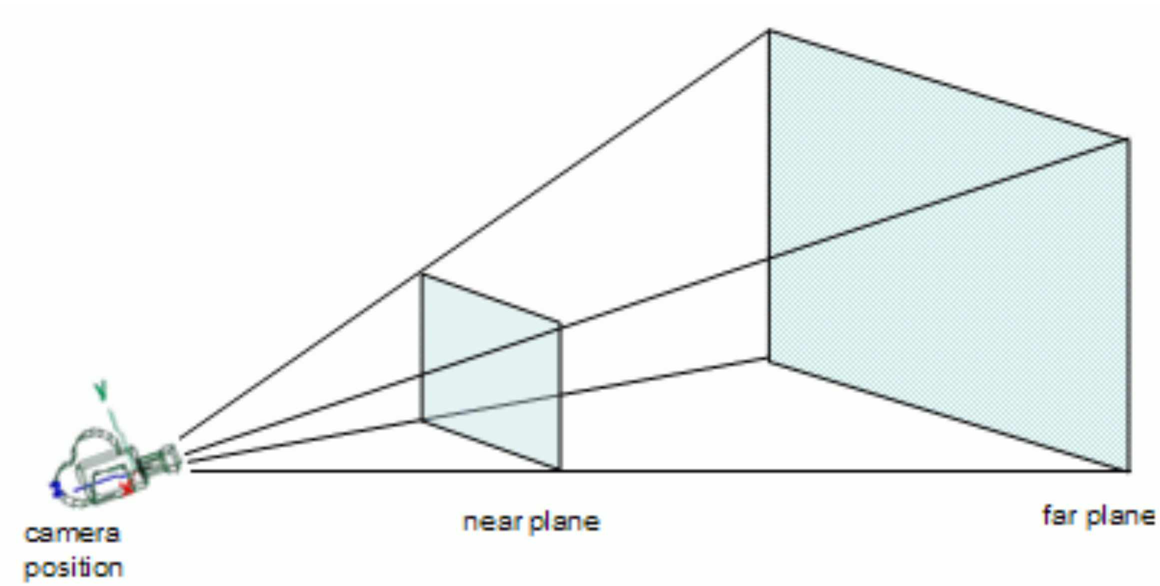
Representing models

- Elements of a scene graph?
  - Culling involves determining which parts of the scene are outside the current view fustrum.
  - Only parts that are inside or partially inside the fustrum are drawn.
    - Partially intersecting has to be drawn too.



Representing models

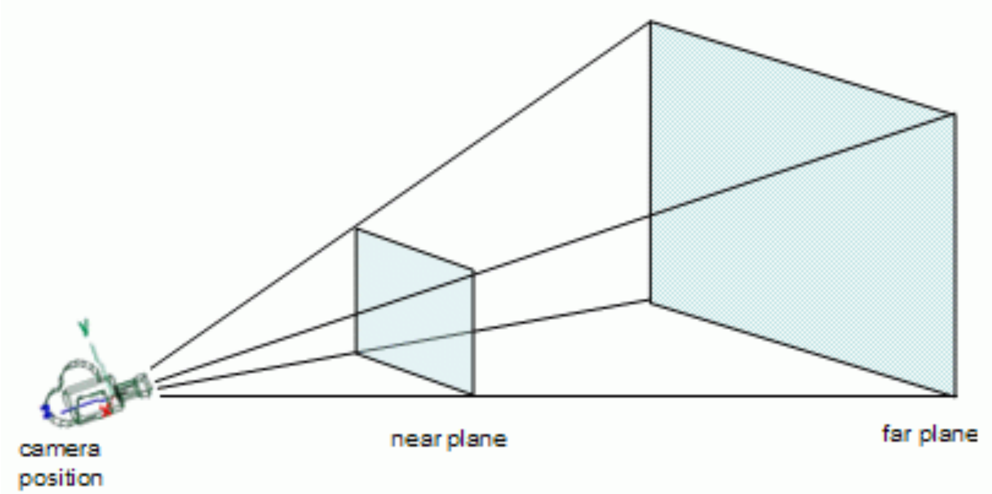
- Elements of a scene graph?
  - Culling involves comparing the bounding volumes of the scene with the six planes that make up the view fustrum.



UCLA

Representing models

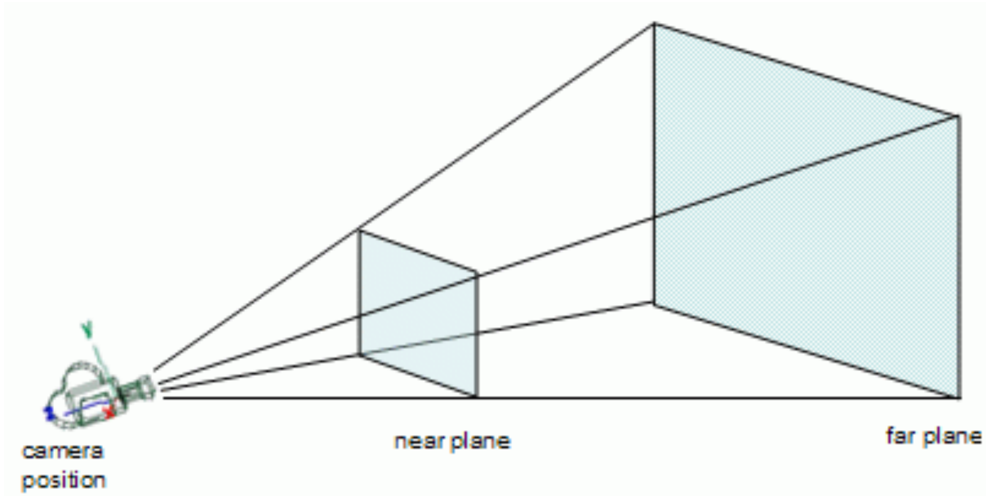
- Elements of a scene graph?
  - If an object is *completely inside* the six planes of the fustrum, we need to render the object.
  - If an object *intersects at least one* of the six planes of the fustrum, we need to render the object.
  - All other objects can be ignored.



UCLA

Representing models

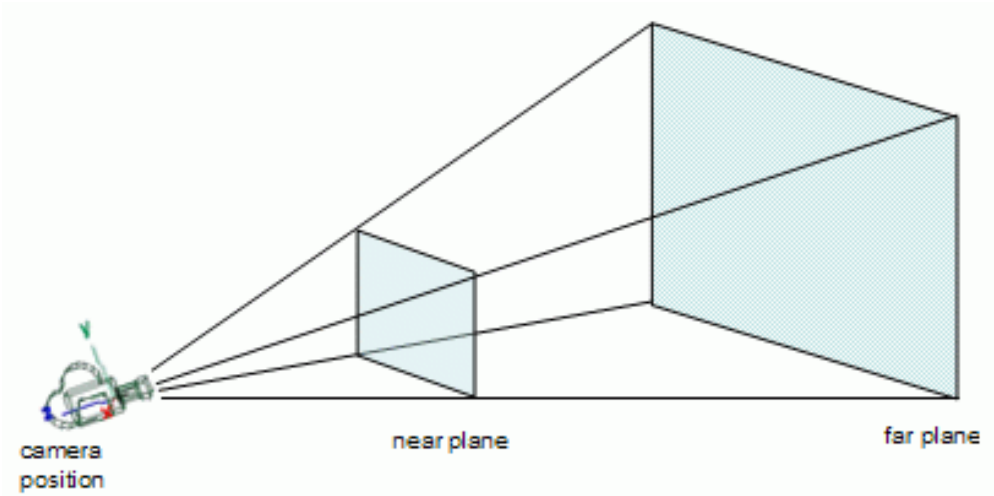
- Elements of a scene graph?
  - Computing the intersections requires that we know the plane equations of all six sides of the fustrum.
  - Where do these come from?



UCLA

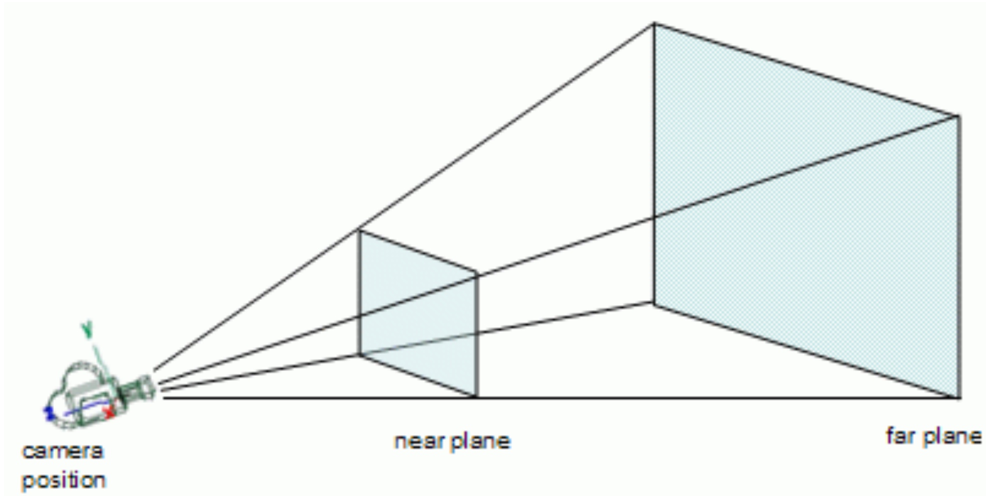
Representing models

- Elements of a scene graph?
  - Turns out we can extract the planes directly from the projection matrix!
  - huh?



Representing models

- Elements of a scene graph?
  - It's actually easiest to extract the planes in clip space.
  - Recall that in clip space the view volume (fustrum) is a cube centered around the origin.
  - However, coordinates are in homogeneous coordinates still – remember *w*?



Representing models

- Elements of a scene graph?
  - Recall that after applying the model-view and projection matrix vertices are in homogeneous *clip* coordinates.
  - Normalizing performs the perspective division.

$$pc = (xc, yc, zc, wc) = PMp$$

$$pcn = \left(\frac{xc}{wc}, \frac{yc}{wc}, \frac{zc}{wc}\right) = (x', y', z')$$

UCLA

Representing models

- Elements of a scene graph?
  - Those normalized coordinates are where the view volume exists at (-1,1),(-1,1),(-1,1).
  - Then, if *pcn* is inside the view fustrum:

$$pcn = (x', y', z')$$

$$-1 < x' < 1$$

$$-1 < y' < 1$$

$$-1 < z' < 1$$

Representing models

- Elements of a scene graph?
  - But we do not have access to the values after the perspective division (inside the hardware).
  - What to do?
  - It turns out that the following relationship is equivalent
    - and *pc* is inside the fustrum if:

$$pc = (xc, yc, zc, wc) = PMp$$

$$-wc < xc < wc$$

$$-wc < yc < wc$$

$$-wc < zc < wc$$

UCLA

Representing models

- Elements of a scene graph?
  - That's nice. What does that mean?
  - It means that, for example, the following is true when *pc* is to the right of the left plane of the fustrum.

$$-wc < xc$$

  - If we recall that *p* and PM are

$$p = (x, y, z, 1) \text{ and } PM = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Representing models

- Elements of a scene graph?

$$p = (x, y, z, 1) \text{ and } PM = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

  - Then we have

$$xc = x \times a_{11} + y \times a_{12} + z \times a_{13} + a_{14}$$
$$yc = x \times a_{21} + y \times a_{22} + z \times a_{23} + a_{24}$$
$$zc = x \times a_{31} + y \times a_{32} + z \times a_{33} + a_{34}$$
$$wc = x \times a_{41} + y \times a_{42} + z \times a_{43} + a_{44}$$

Representing models

- Elements of a scene graph?

$$xc = x \times a_{11} + y \times a_{12} + z \times a_{13} + a_{14}$$
$$wc = x \times a_{41} + y \times a_{42} + z \times a_{43} + a_{44}$$

  - Then, recall and a little algebra

$$-wc < xc$$
$$xc + wc > 0$$
$$x(a_{11} + a_{41}) + y(a_{12} + a_{42}) + z(a_{13} + a_{43}) + (a_{14} + a_{44}) > 0$$

  - Looks more like a plane equation…

Representing models

- Elements of a scene graph?

$$-wc < xc$$

$$xc + wc > 0$$

$$x(a_{11} + a_{41}) + y(a_{12} + a_{42}) + z(a_{13} + a_{43}) + (a_{14} + a_{44}) > 0$$

  - Maybe this looks more familiar?

$$(Ax + By + Cz + D) = 0$$

$$A = a_{11} + a_{41}$$

$$B = a_{12} + a_{42}$$

$$C = a_{13} + a_{43}$$

$$D = a_{14} + a_{44}$$

UCLA

Representing models

- Elements of a scene graph?
  - If a point *p* evaluated to 0 the point is on the plane.
  - If a point *p* evaluates to >0 we are on the positive side of the plane – or *inside*.
  - If a point *p* evaluates to >0 we are on the negative side of the plane – or *outside*.

$$(Ax + By + Cz + D) = 0$$

$$A = a_{11} + a_{41}$$

$$B = a_{12} + a_{42}$$

$$C = a_{13} + a_{43}$$

$$D = a_{14} + a_{44}$$

UCLA

Representing models

- Elements of a scene graph?
  - For intersecting with a sphere we need to normalize for the distance values to be meaningful.
  - Remember, also, that the normal to this plane is defined by (A,B,C)

$$(Ax + By + Cz + D) = 0$$

$$A = a_{11} + a_{41}$$

$$B = a_{12} + a_{42}$$

$$C = a_{13} + a_{43}$$

$$D = a_{14} + a_{44}$$

Representing models

- Elements of a scene graph?
  - The planes of the fustrum need to be extracted each time the camera or objects in the scene are moved.
    - Every frame, basically.
  - It is trivial to look up the values for all six planes.
  - As the scene graph is traversed the bounding volume is checked against the fustrum.
  - If it falls outside the fustrum that branch of the scene graph can be skipped.