Java 3D Transformations

- The Java 3D model for 4 x 4 transformations is

```
[ m00 m01 m02 m03 ]     [ x  ]     [ x' ]
[ m10 m11 m12 m13 ]  .  [ y  ]  =  [ y' ]
[ m20 m21 m22 m23 ]     [ z  ]     [ z' ]
[ m30 m31 m32 m33 ]     [ w ]      [ w' ]
```

```
x' = m00 . x  +  m01 . y  +  m02 . z  +  m03 . w
y' = m10 . x  +  m11 . y  +  m12 . z  +  m13 . w
z' = m20 . x  +  m21 . y  +  m22 . z  +  m23 . w
w' = m30 . x  +  m31 . y  +  m32 . z  +  m33 . w
```

- When transforming a `Point3f` or a `Point3d`, w = 1
- When transforming a `Vector3f` or a `Vector3d`, w = 0

Java 3D – Transform3D

- `Transform3D` objects represent geometric transformations such as rotation, translation, and scaling
- The transformations represented by a `Transform3D` object are used to create `TransformGroup` objects that become Nodes the scene graph

`void rotX(double angle)`
Sets the value of this transform to a counter clockwise rotation about the x-axis, in radians

`void rotY(double angle)`
Sets the value of this transform to a counter clockwise rotation about the y-axis, in radians

`void rotZ(double angle)`
Sets the value of this transform to a counter clockwise rotation about the z-axis, in radians

`void set(Vector3f translate)`
Sets the translational value of this matrix to the `Vector3f` parameter values and sets the other components of the matrix as if this transform were an identity matrix

Some Transform3D Methods

`Transform3D()`

Constructs and initializes a transform to the identity matrix

`Transform3D(double[] matrix)`

Constructs and initializes a transform from the double precision array of length 16; the top row of the matrix is initialized to the first four elements of the array, and so on

`Transform3D(Matrix3f m1, Vector3f t1,float s)`

Constructs and initializes a transform from the rotation matrix, translation, and scale values

`Transform3D(Quat4d q1, Vector3d t1, double s)`

Constructs and initializes a transform from the quaternion, translation, and scale values

More Transform3D Methods

`add(Transform3D t1)`

Sets the value of this transform to the result of adding transform t1 to it (this = this + t1)

`mul(Transform3D t1)`

Sets the value of this transform to the result of multiplying it by transform t1 (this = this * t1)

`equals(Transform3D t1)`

Returns true if all of the data members of transform t1 are equal to the corresponding data members in this Transform3D

Java 3D – Transform Groups

- `Transform3D` objects define affine homogeneous transformations but are not nodes in the scene graph
- `Transform3D` objects are linked into the scene graph via `TransformGroup` nodes
- `TransformGroup` objects possess a capability bit which permits certain modifications to be made after they have been made *live* (added to the scene graph) or *compiled*
- `TransformGroup` objects also possess a behaviour bit which permits animation, etc.

Scene Graphs & Transform Groups

- Once a branch graph is made *live* or *compiled* the Java 3D rendering system converts it to a more efficient internal representation
  - The most important effect of converting to the internal representation is to improve rendering performance
- Making the transformation to the internal representation has other effects as well
  - One effect is to fix the value of transformations and other objects in the scene graph
  - Unless specifically provided for in the program the program no longer has the capability to change the values of the scene graph objects after they have been made *live*

Scene Graph Objects

- `SceneGraphObject` is the superclass of nearly every class used to create a scene graph (both the actual Nodes of a scene graph and the Node Components associated with a scene graph)
  - Group
  - Leaf
  - NodeComponent

- Scene graph objects have *capability bits* which can be accessed, set and cleared

  `void clearCapability(int bit)`

  Clears the specified capability bit

  `boolean getCapability(int bit)`

  Retrieves the specified capability bit

  `void setCapability(int bit)`

  Sets the specified capability bit

Transform Group Capabilities

- **ALLOW_TRANSFORM_READ**
  - Allows access to the transform information
- **ALLOW_TRANSFORM_WRITE**
  - Allows the transform information to be written (changed) after this group is made live or compiled
- **ALLOW_CHILDREN_READ**
  - Allows references to children nodes to be read
- **ALLOW_CHILDREN_WRITE**
  - Allows references to children nodes to be written (changed) after this group is made live or compiled
- **ALLOW_CHILDREN_EXTEND**
  - Allows children to be added after this group is made live or compiled
- **ALLOW_COLLISION_BOUNDS_READ**
  - Allows the collision bounds information to be accessed
- **ALLOW_COLLISION_BOUNDS_WRITE**
  - Allows the collision bounds information to be written (changed) after this group is made live or compiled