3D Geometric Transformations

3D Affine Transformations

Quaternions

Transformations in Scene Graphs

Composition of Transformations

Affine Transformation

3D affine transformation

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

In homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{22} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Homogeneous Coordinates

- In homogeneous coordinates, any 3D point (x,y,z) can be represented by a line in 4D (w·x,w·y,w·z,w)

In homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{22} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Matrix Classes

Matrix3f
Matrix3d
Matrix4f
Matrix4d
GMatrix

```
void add(Matrix4d m1)
void sub(Matrix4d m1)
void mul(Matrix4d m1)
void invert()
void add(Matrix4d m1, Matrix4d m2)
void sub(Matrix4d m1, Matrix4d m2)
void mul(Matrix4d m1, Matrix4d m2)
void invert(Matrix4d m1)
void transpose()
void mul(double scalar)
double determinant()
```

Matrix Classes

- The Matrix4d class

```
double[] array = { 1.0, 2.0, 3.0, 1.0,
                   0.0, 1.0, 1.0, 1.0,
                   2.0, 0.0, 0.25, 2.0,
                   0.0, 0.0, 0.0, 1.0 };
Matrix4d matrix = new Matrix4d(array);
```

Matrix Classes

- The GMatrix class

```
double[] array = { 1.0, 2.0, 3.0, 1.0,
                   0.0, 1.0, 1.0, 1.0,
                   2.0, 0.0, 0.25, 2.0 };

GMatrix matrix = new GMatrix(3, 4, array);
```

Transform3D Class

Translation

$$\begin{bmatrix} 1 & 0 & 0 & b_1 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
void set(Vector3d trans)
void set(Vector3f trans)
void setTranslation(Vector3d trans)
void setTranslation(Vector3f trans)
```

Scale

$$\begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
void set(double scale)
void setScale(double scale)
void setScale(Vector3d scales)
```

Reflection and Shear

Reflection

void set(Matrix4d m1)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T(x) = x - \frac{2x \cdot u}{\|u\|^2} u$$

Shear

$$\begin{bmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad x' = x + sh_x z$$

$$\begin{bmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{aligned} x' &= x + sh_x z \\ y' &= y + sh_y z \end{aligned}$$

3D Rotation

About z-axis

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Set a general rotation in Transform3D

```
void set(AxisAngle4d r)
```

Quaternions

- Review of complex numbers
  - A complex number can be written as:
    $a + bi$
  - Where a and b are real numbers and
    $i^2 = -1$
- You add (or subtract) complex numbers by adding the corresponding parts
  $(2 + 3i) + (1 + 2i) = (3 – 5i)$
- You multiply complex numbers symbolically, using distributive law and $i^2 = -1$
  $(2 + 3i) * (1 + 2i) = 2*1 + 2*2i + 1*3i + 3i*2i$
  $= 2 + 7i + 6i^2 = 2 + 7i – 6 = -4 + 7i$

Quaternions

- The quaternions are an extension of the complex numbers
- A quaternion is a number: $a+b\cdot i+c\cdot j+d\cdot k$
  - where a, b, c, d are real numbers
  - And: $i^2=j^2=k^2=i\cdot j\cdot k=-1$
  - We also have that:

$$i\cdot j=k=-j\cdot i,\quad j\cdot k=i=-k\cdot j,\quad k\cdot i=j=-i\cdot k$$

- The conjugate of a quaternion $q=a+b\cdot i+c\cdot j+d\cdot k$ is $\bar{q}=a-b\cdot i-c\cdot j-d\cdot k$

Quaternion and 3D Rotation

A point represented as a pure quaternion

$$p = xi + yj + zk$$

A rotation defined by quaternion operations

$$T_q(p) = qp\bar{q}$$

$$q = \cos\frac{\theta}{2} + u\sin\frac{\theta}{2}, \quad u = ai + bj + ck$$

θ is the angle of rotation and *u* defines the axis of rotation through the origin
u is a unit vector

Euler Angles

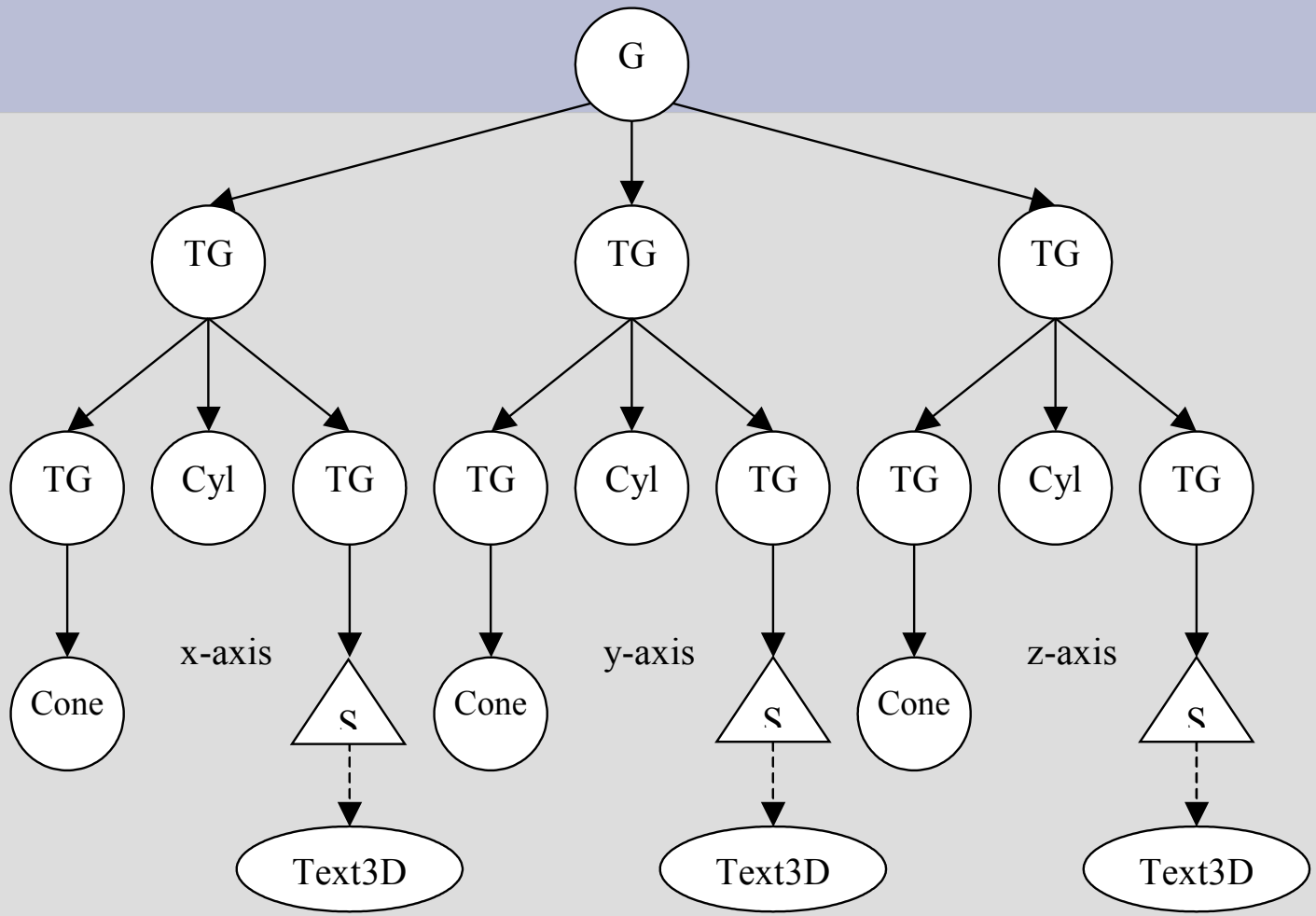
- Use three rotations about the coordinate axes
- Different Names:
  - (elevation, azimuth, tilt)
  - (roll, pitch, yaw)
  - (precession, nutation, spin)
  - (heading, altitude, bank)
- Transform3D method

```
void setEuler(Vector3d eulerAngles)
```

Quaternion to Euler Angle Conversion

```
public static Vector3d quatToEuler(Quat4d q1) {
  double sqw = q1.w*q1.w;
  double sqx = q1.x*q1.x;
  double sqy = q1.y*q1.y;
  double sqz = q1.z*q1.z;
  double heading = Math.atan2(2.0 * (q1.x*q1.y +
q1.z*q1.w),(sqx - sqy - sqz + sqw));
  double bank = Math.atan2(2.0 * (q1.y*q1.z +
q1.x*q1.w),(-sqx - sqy + sqz + sqw));
  double attitude = Math.asin(-2.0 * (q1.x*q1.z -
q1.y*q1.w));
  return new Vector3d(bank, attitude, heading);
}
```

TransformGroup Node



Composite Transforms

$$(T_2 T_1)(p) = (T_2(T_1(p))$$

$$T_1 T_2 \neq T_2 T_1$$

A useful pattern $T^{-1} R T$

Transform3D methods

```
void transform(Point3d p)
void transform(Point3d p, Point3d pOut)
void transform(Point3f p)
void transform(Point3f p, Point3f pOut)
void transform(Vector3d v)
void transform(Vector3d v, Vector3d vOut)
void transform(Vector3f v)
void transform(Vector3f v, Vector3f vOut)
void transform(Vector4d v)
void transform(Vector4d v, Vector4d vOut)
void transform(Vector4f v)
void transform(Vector4f v, Vector4f vOut)
```

Shape Construction

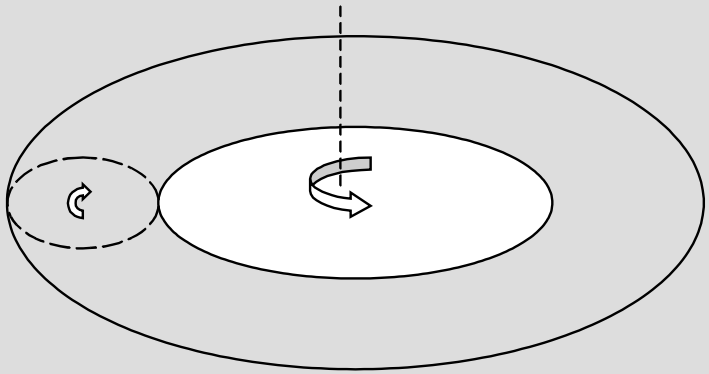
Extrusion:
Extend a 2D shape to 3D

Source

Rotation:
For example, constructing a torus

Source

Run



Transformation and Shared Branch