University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2016

Tamara Munzner

Transformations 6

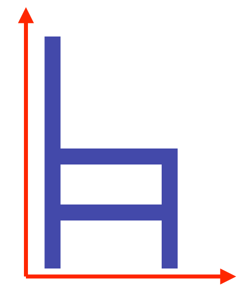
http://www.ugrad.cs.ubc.ca/~cs314/Vjan2016

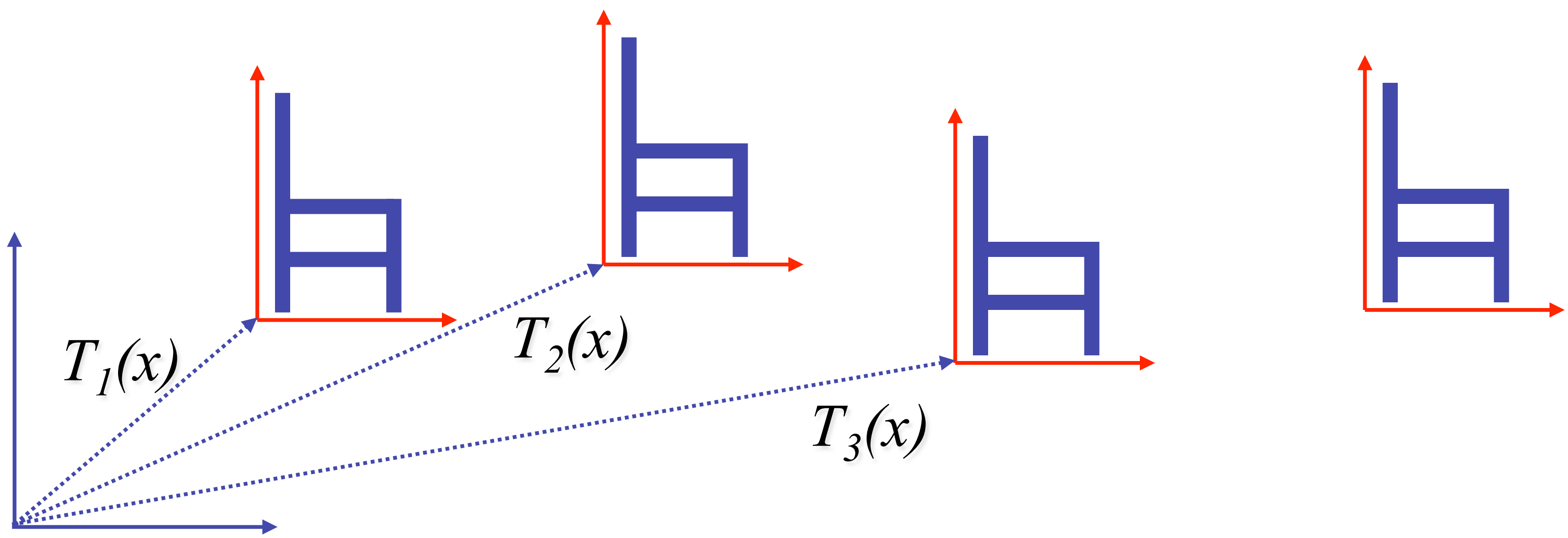
Transformation Hierarchies

Scaling and Rotating

Matrix Stacks

- challenge of avoiding unnecessary computation
  - using inverse to return to origin
  - computing incremental T_1 -> T_2



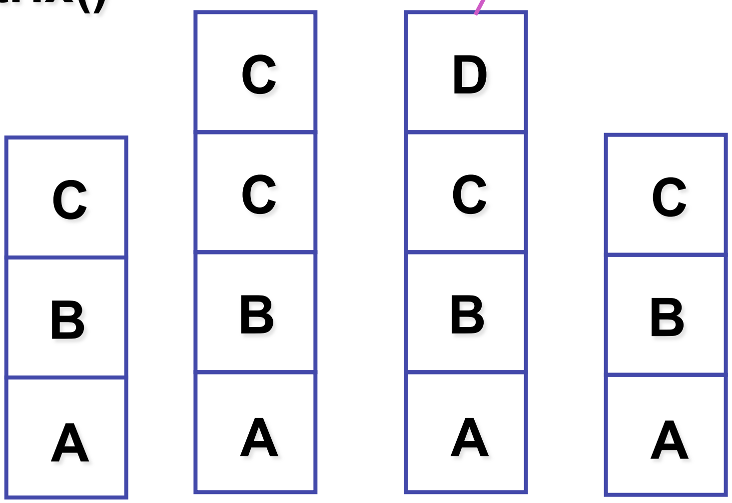


Matrix Stacks

pushMatrix()

popMatrix()

D = C scale(2,2,2) trans(1,0,0)



drawSquare()

pushMatrix()

scale(2,2,2)

translate(1,0,0)

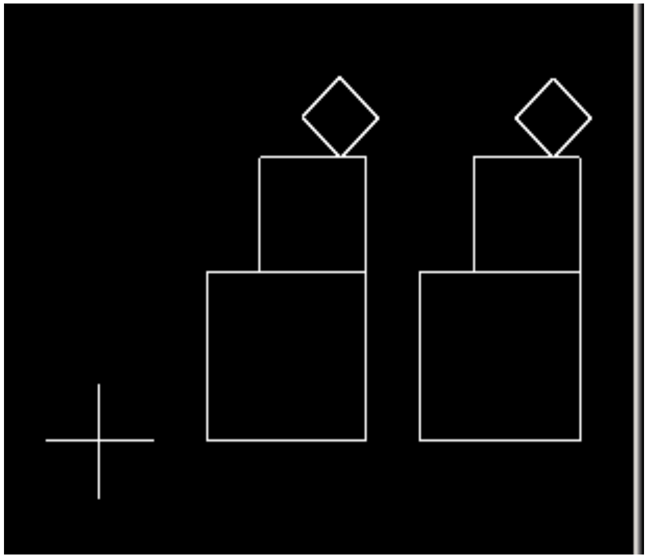
drawSquare()

popMatrix()

Modularization

- drawing a scaled square
  - push/pop ensures no coord system change

```
void drawBlock(float k) {
  pushMatrix();
  scale(k,k,k);
  drawBox();
  popMatrix();
}
```

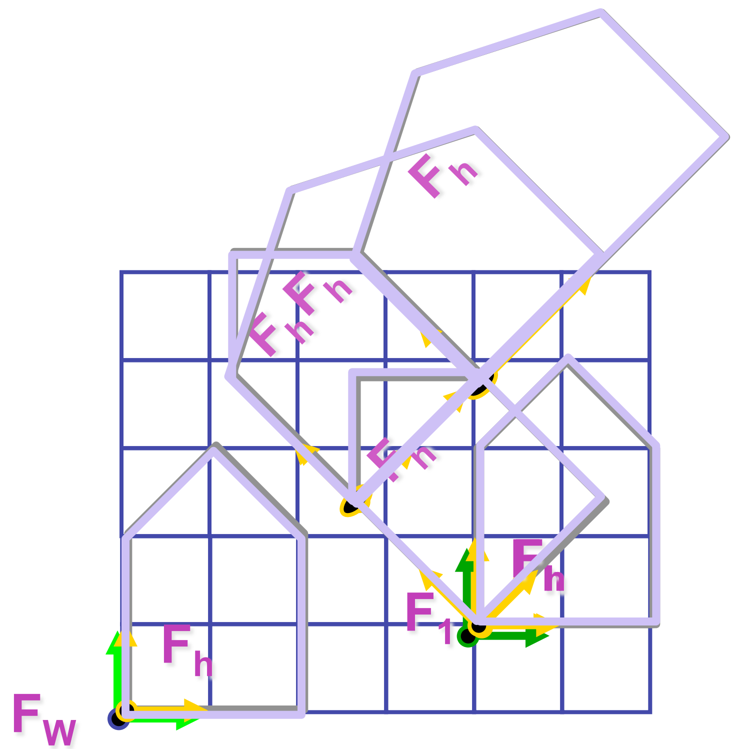


Matrix Stacks

- advantages
  - no need to compute inverse matrices all the time
  - modularize changes to pipeline state
  - avoids incremental changes to coordinate systems
    - accumulation of numerical errors

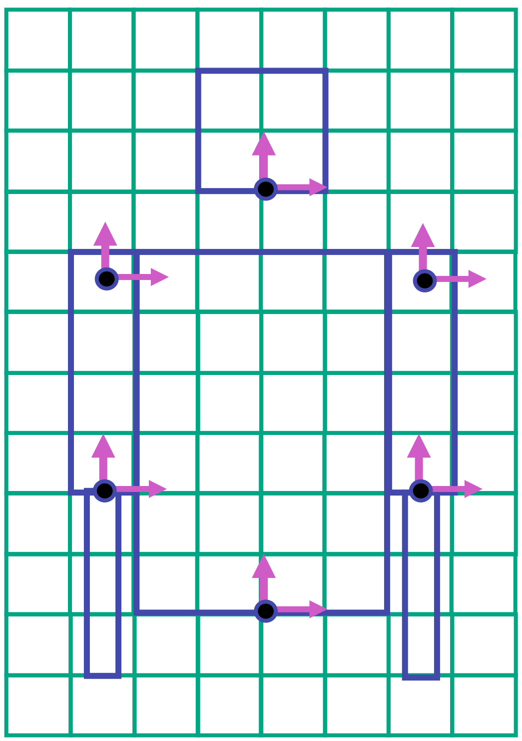
- disadvantages
  - not built in to WebGL
    - but easy to implement with Array.pop/push
    - see also
      https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial/Animating_objects_with_WebGL#More_matrix_operations

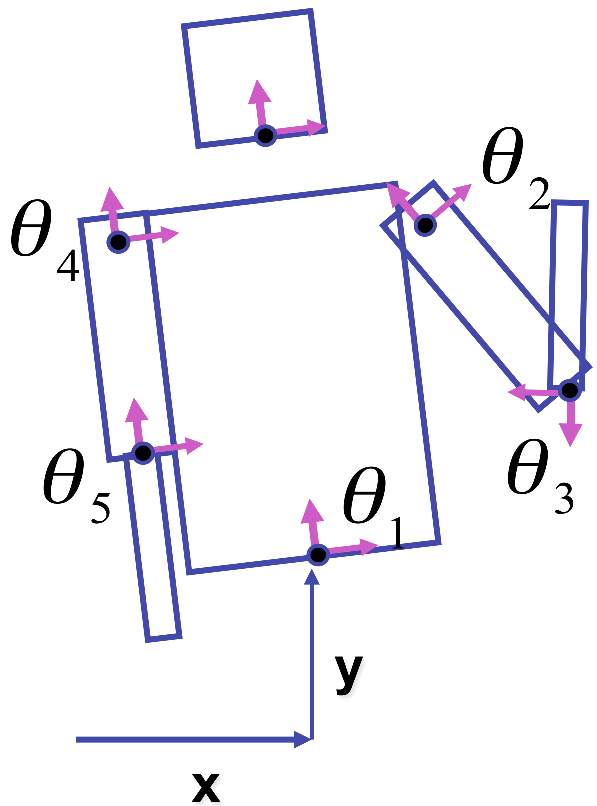
Transformation Hierarchy Example 3




```
loadIdentity();
translate(4,1,0);
pushMatrix();
rotate(45,0,0,1);
translate(0,2,0);
scale(2,1,1);
translate(1,0,0);
popMatrix();
```

Transformation Hierarchy Example 4






```
translate(x,y,0);
rotate(t1,0,0,1);
DrawBody();
pushMatrix();
  translate(0,7,0);
  DrawHead();
popMatrix();
pushMatrix();
  translate(2.5,5.5,0);
  rotate( t2,0,0,1);
  DrawUArm();
  translate(0,-3.5,0);
  rotate(t3,0,0,1);
  DrawLArm();
popMatrix();
... (draw other arm)
```

Hierarchical Modelling

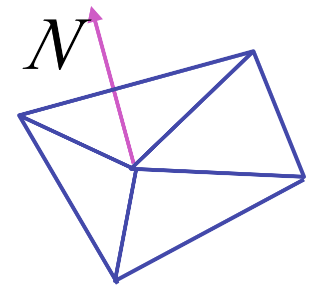
- advantages
  - define object once, instantiate multiple copies
  - transformation parameters often good control knobs
  - maintain structural constraints if well-designed
- limitations
  - expressivity: not always the best controls
  - can't do closed kinematic chains
    - keep hand on hip
  - can't do other constraints
    - collision detection
      - self-intersection
      - walk through walls

Transforming Normals

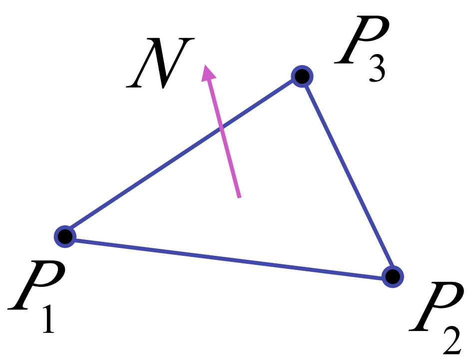
Transforming Geometric Objects

- lines, polygons made up of vertices
  - transform the vertices
  - interpolate between
- does this work for everything? no!
  - normals are trickier

Computing Normals



- normal
  - direction specifying orientation of polygon
    - w=0 means direction with homogeneous coords
    - vs. w=1 for points/vectors of object vertices
  - used for lighting
    - must be normalized to unit length
  - can compute if not supplied with object



$$N = (P_2 - P_1) \times (P_3 - P_1)$$

Transforming Normals

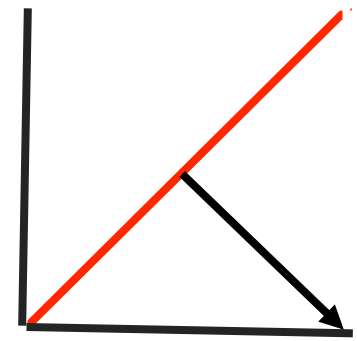
$$\begin{bmatrix} x' \\ y' \\ z' \\ 0 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

- so if points transformed by matrix **M**, can we just transform normal vector by **M** too?
  - translations OK: w=0 means unaffected
  - rotations OK
  - uniform scaling OK

- these all maintain direction

Transforming Normals

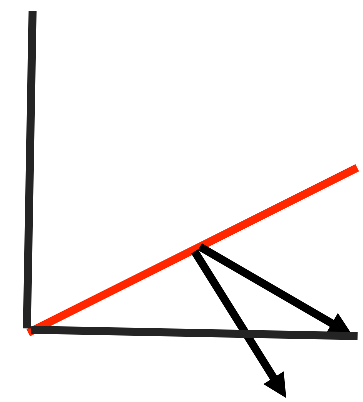
- nonuniform scaling does not work
- x-y=0 plane
  - line x=y
  - normal: [1,-1,0]
    - direction of line x=-y
    - (ignore normalization for now)



Transforming Normals

- apply nonuniform scale: stretch along x by 2
  - new plane x = 2y
- transformed normal: [2,-1,0]

$$\begin{bmatrix} 2 \\ -1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}$$



  - normal is direction of line x = -2y or x+2y=0
  - not perpendicular to plane!
  - should be direction of 2x = -y

Planes and Normals

- plane is all points perpendicular to normal
  - $N \bullet P = 0$ (with dot product)
  - $N^T \bullet P = 0$ (matrix multiply requires transpose)

$$N = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, P = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- explicit form: plane = $ax + by + cz + d$

Finding Correct Normal Transform

- transform a plane

$$P \longrightarrow P' = MP$$
$$N \longrightarrow N' = QN$$

given M, what should Q be?

$$N'^T P' = 0$$

stay perpendicular

$$(QN)^T (MP) = 0$$

substitute from above

$$N^T \underbrace{Q^T M}P = 0$$

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$$

$$Q^T M = I$$

$$\mathbf{N}^T\mathbf{P} = 0 \text{ if } \mathbf{Q}^T\mathbf{M} = \mathbf{I}$$

$$\boxed{\mathbf{Q} = \left(\mathbf{M}^{-1}\right)^{\mathbf{T}}}$$

thus the normal to any surface can be transformed by the inverse transpose of the modelling transformation