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

Tamara Munzner

Transformations 5

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

Assignments

- project 1
  - out today, due 11:59pm sharp Tue Feb 2
    - projects will go out before we've covered all the material
      - so you can think about it before diving in
  - build star-nosed mole out of cubes and 4x4 matrices
    - think cartoon, not beauty
    - http://www.ugrad.cs.ubc.ca/~cs314/Vjan2016/p1.pdf
  - template code gives you program shell
    http://www.ugrad.cs.ubc.ca/~cs314/Vjan2016/p1.zip
- theory homework 1
  - out today, due 2pm sharp Wed Jan 27 (start of class)
  - theoretical side of material
    - http://www.ugrad.cs.ubc.ca/~cs314/Vjan2016/h1.pdf

Real Star-Nosed Moles

http://aninfopage.blogspot.ca/2011/12/star-nose-mole.html

http://www.rsba.ca/recherche_espece/fiche_espece.php?recordID=334

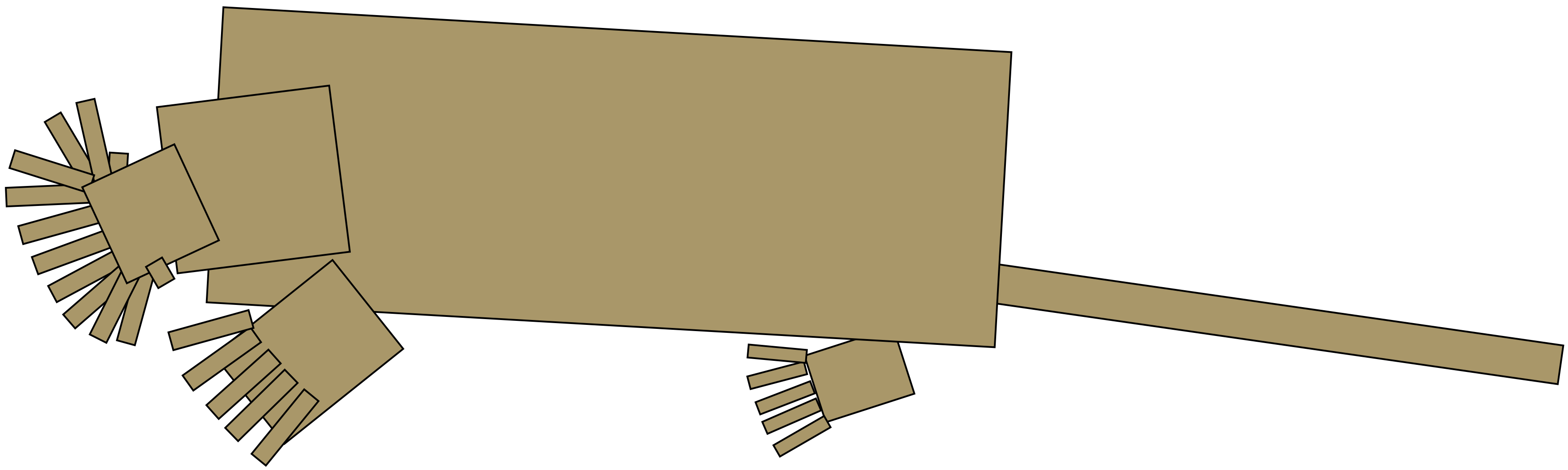
http://animals.howstuffworks.com/mammals/mole-info.htm

http://www.biokids.umich.edu/critters/Condylura_cristata/

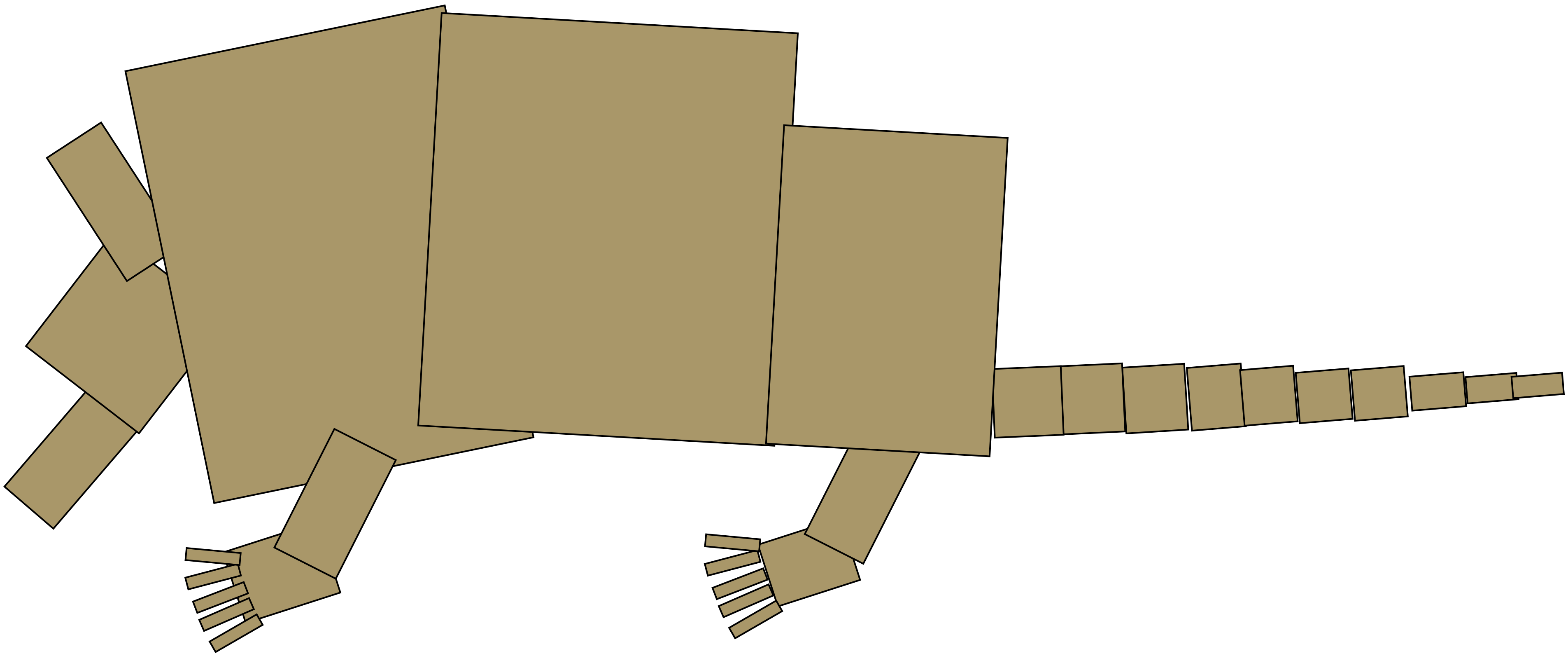
https://youtu.be/RCB2VT3Nw1I

Star-Nosed Moles!

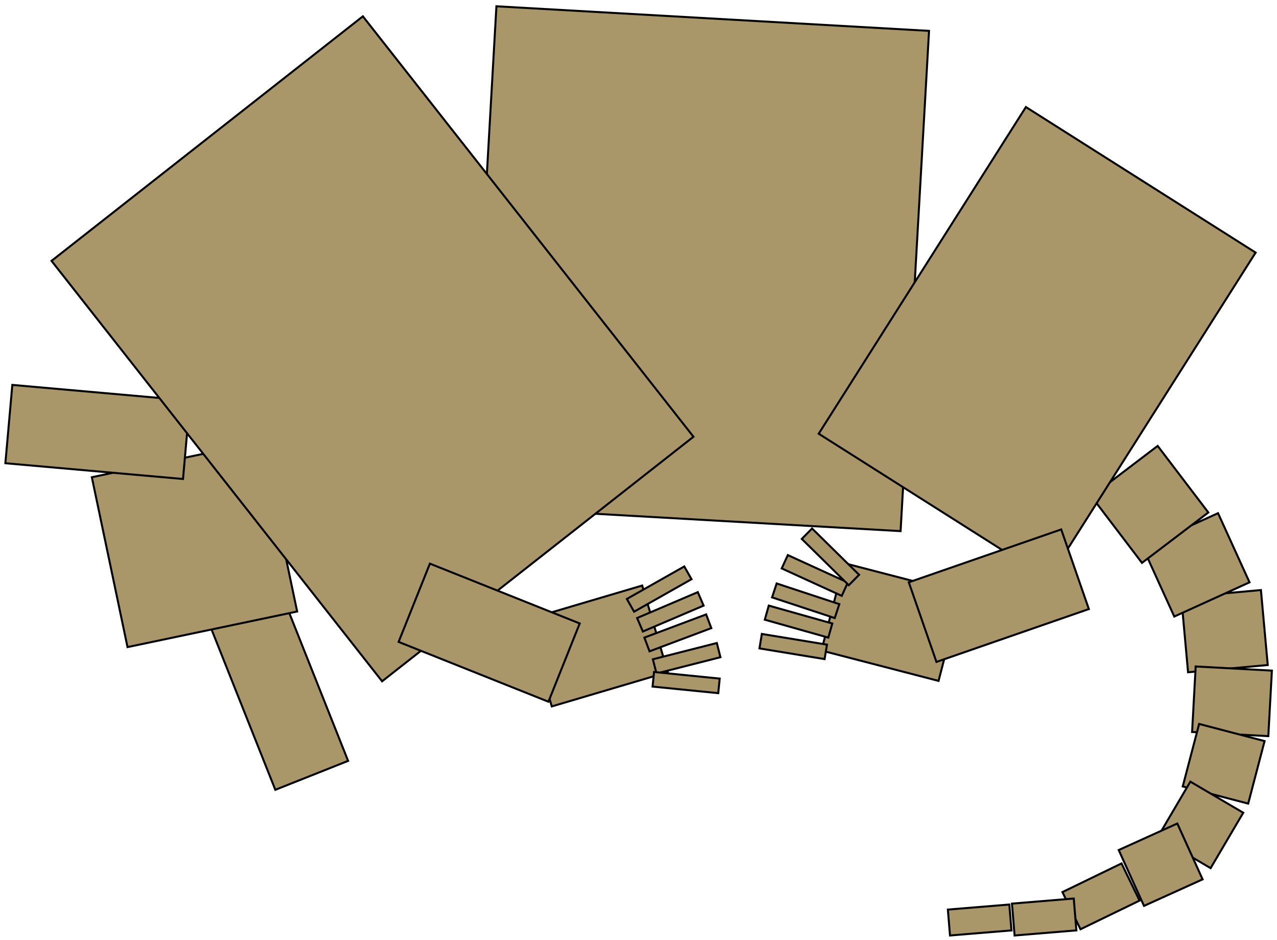
- **out of boxes and matrices**



Cartoon motion: armadillo jumpcut



Cartoon motion: armadillo jumpcut



Project 1 Advice

- do not model everything first and only then worry about animating
- interleave modelling, animation
  - for each body part: add it, then jumpcut animate, then smooth animate
  - discover if on wrong track sooner
  - dependencies: can't get anim credit if no model
  - use body as scene graph root
- check from multiple camera angles

Project 1 Advice

- finish all required parts before
  - going for extra credit
  - playing with lighting or viewing
- construct your 4x4 matrix by hand
  - without rotate(), translate(), scale() commands in Three.js
  - do not interpolate numbers within matrix
    - even though it's safe to linearly interpolate parameters you use to create matrix

Project 1 Advice

- smooth transition
  - change happens gradually over X frames
  - key click triggers animation
  - one way: redraw happens X times
    - linear interpolation:

      each time, param += (new-old)/30
  - or redraw happens over X seconds
    - even better, but not required

Style

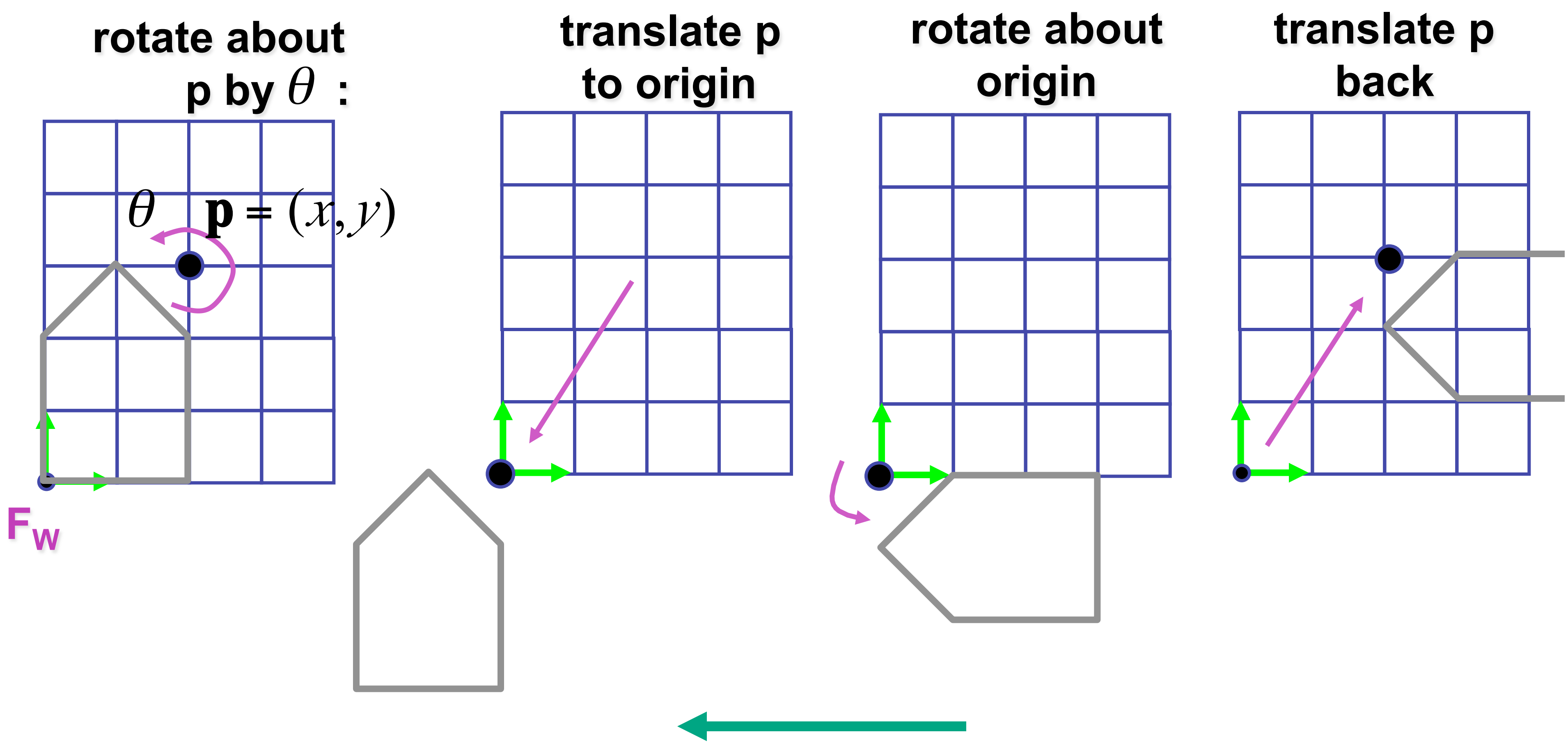
- you can lose up to 15% for poor style
- most critical: reasonable structure
  - yes: parametrized functions
  - no: cut-and-paste with slight changes
- reasonable names (variables, functions)
- adequate commenting
  - rule of thumb: what if you had to fix a bug two years from now?
- global variables are indeed acceptable

Version Control

- bad idea: just keep changing same file
- save off versions often
  - after got one thing to work, before you try starting next
  - just before you do something drastic
- use version control software
  - strongly recommended: easy to browse previous work, revert
  - use meaningful comments to describe what you did
    - “started on tail”, “fixed head breakoff bug”, “leg code compiles but doesn’t run”
- useful when you’re working alone, critical when working together

General Rotation

Rotation About a Point: Moving Object



$$\mathbf{T}(x, y, z)\mathbf{R}(z, \theta)\mathbf{T}(-x, -y, -z)$$

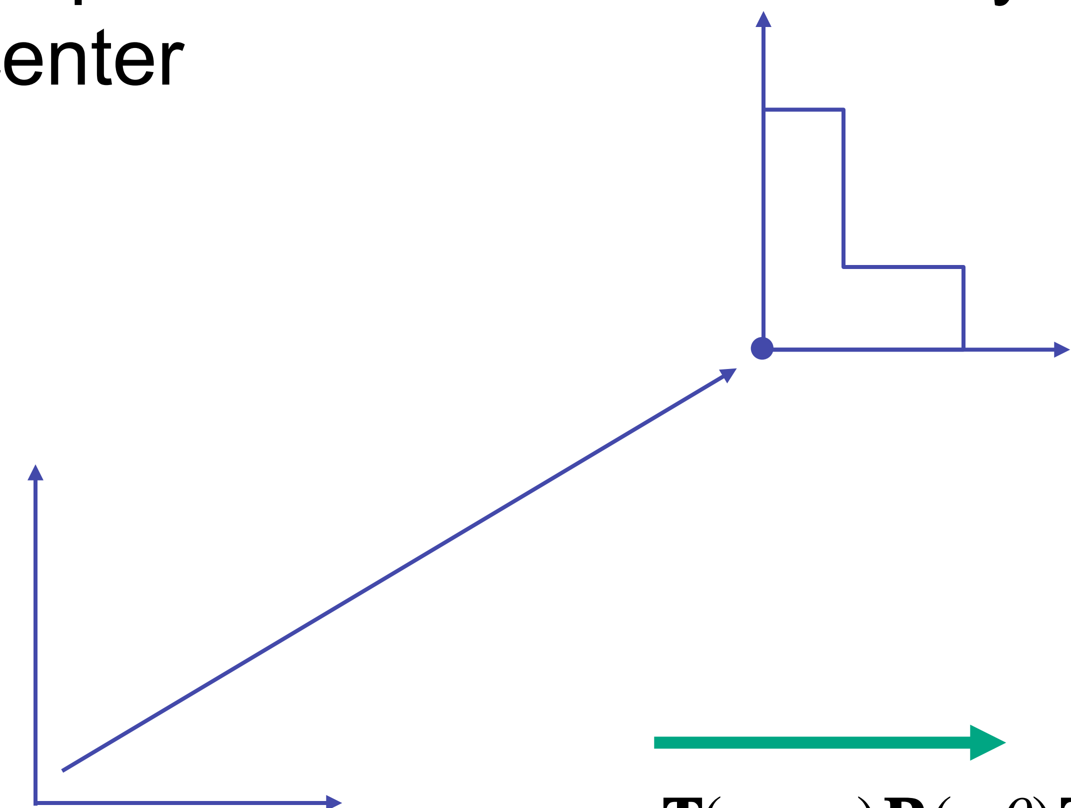
Rotation: Changing Coordinate Systems

- same example: rotation around arbitrary center

$$\mathbf{T}(x, y, z)\,\mathbf{R}(z, \theta)\,\mathbf{T}(-x, -y, -z)$$

Rotation: Changing Coordinate Systems

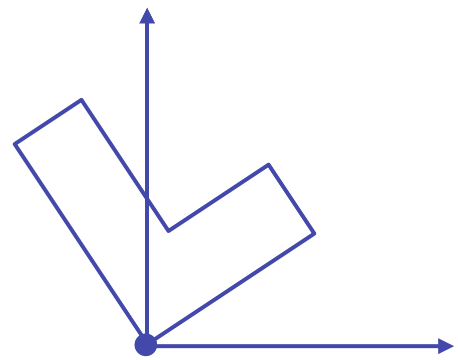
- rotation around arbitrary center
  - step 1: translate coordinate system to rotation center



$$\mathbf{T}(x,y,z)\,\mathbf{R}(z,\theta)\,\mathbf{T}(-x,-y,-z)$$

Rotation: Changing Coordinate Systems

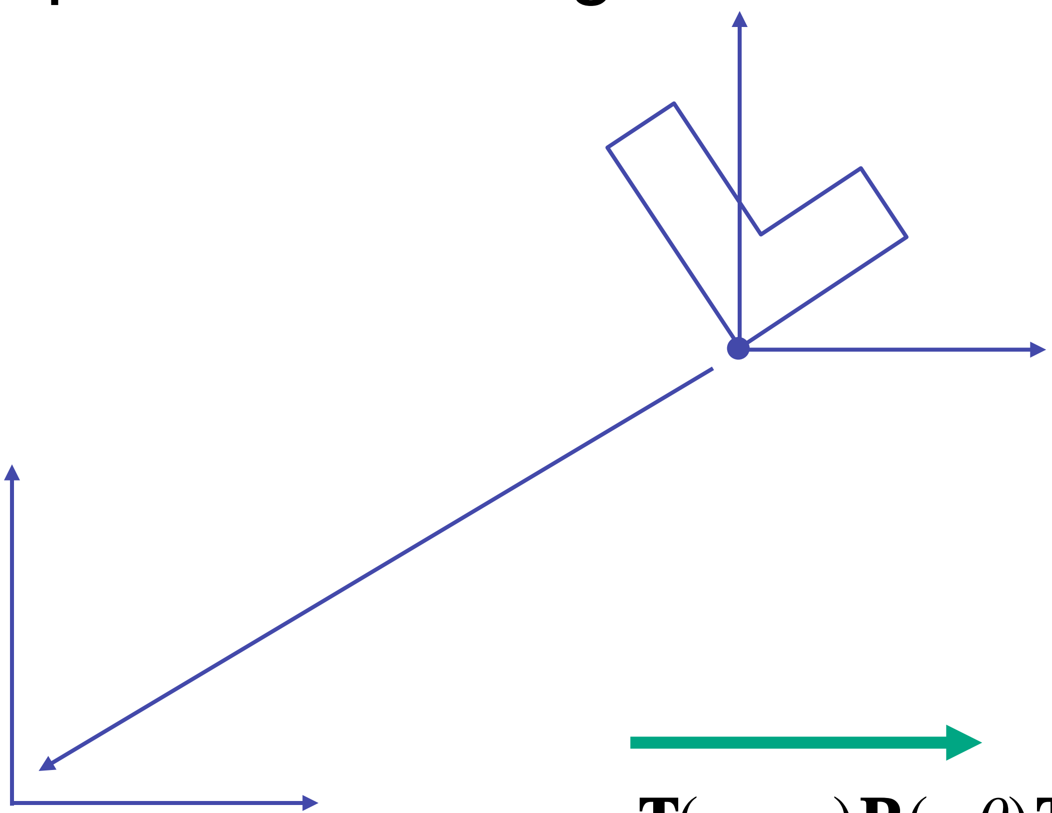
- rotation around arbitrary center
  - step 2: perform rotation



$$\mathbf{T}(x,y,z)\,\mathbf{R}(z,\theta)\,\mathbf{T}(-x,-y,-z)$$

Rotation: Changing Coordinate Systems

- rotation around arbitrary center
  - step 3: back to original coordinate system



$$\mathbf{T}(x,y,z)\,\mathbf{R}(z,\theta)\,\mathbf{T}(-x,-y,-z)$$

General Transform Composition

- transformation of geometry into coordinate system where operation becomes simpler
  - typically translate to origin

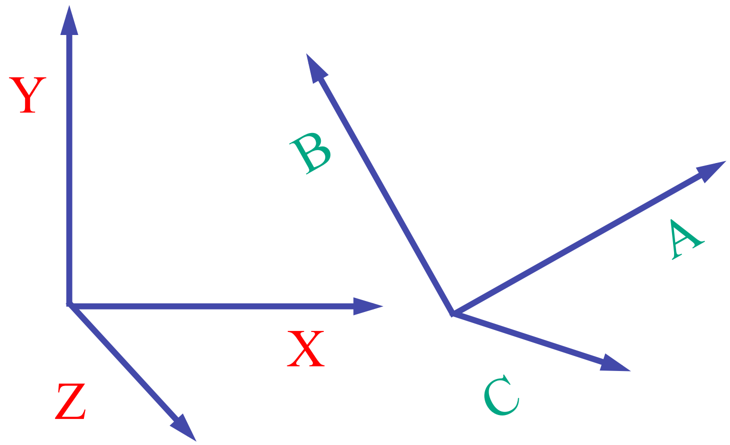
- perform operation

- transform geometry back to original coordinate system

Rotation About an Arbitrary Axis

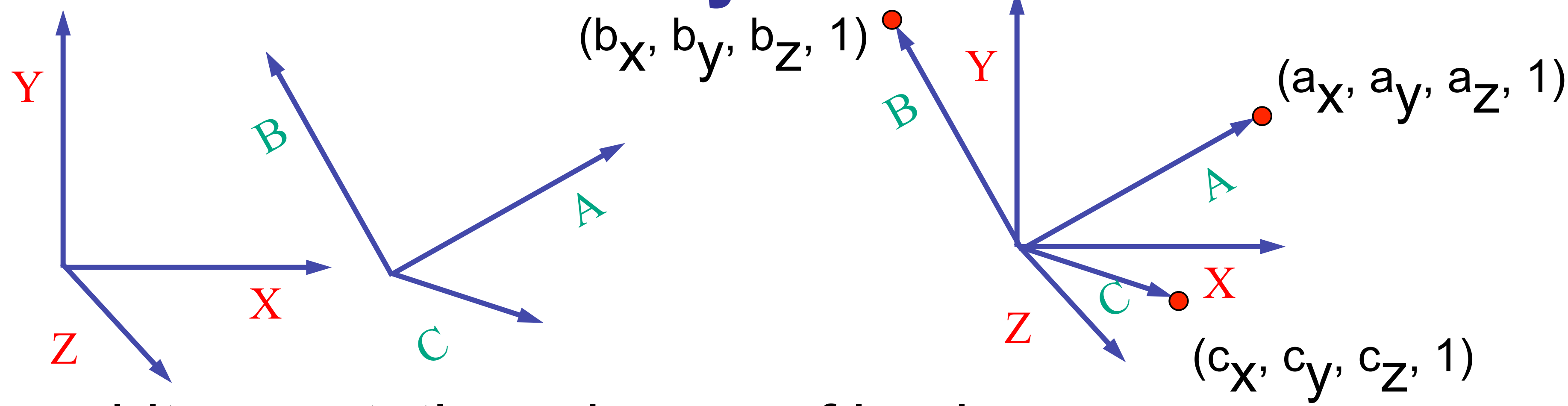
- axis defined by two points
- translate point to the origin
- rotate to align axis with z-axis (or x or y)
- perform rotation
- undo aligning rotations
- undo translation

Arbitrary Rotation



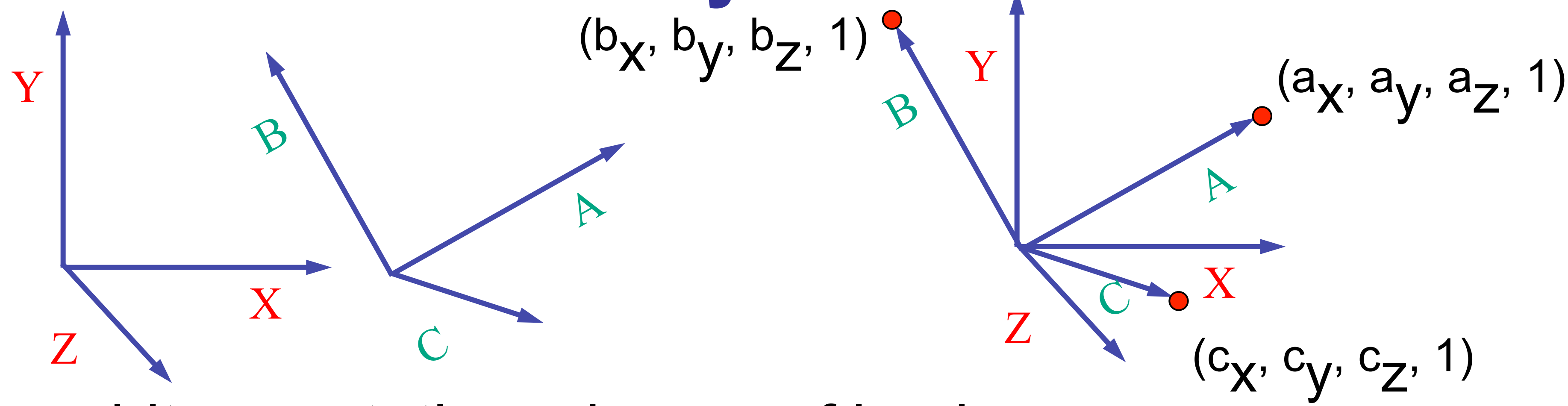
- arbitrary rotation: change of basis
  - given two orthonormal coordinate systems *XYZ* and *ABC*
    - *A*'s location in the XYZ coordinate system is $(a_x, a_y, a_z, 1)$, ...

Arbitrary Rotation



- arbitrary rotation: change of basis
  - given two orthonormal coordinate systems *XYZ* and *ABC*
    - *A*'s location in the XYZ coordinate system is $(a_x, a_y, a_z, 1)$, ...

Arbitrary Rotation



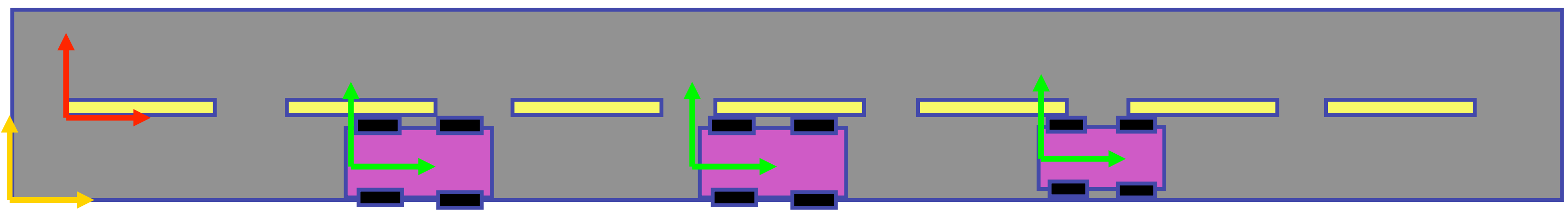
- arbitrary rotation: change of basis
  - given two orthonormal coordinate systems *XYZ* and *ABC*
    - *A*'s location in the XYZ coordinate system is $(a_x, a_y, a_z, 1)$, ...
- transformation from one to the other is matrix R whose columns are *A,B,C*:

$$R(X) = \begin{bmatrix} a_x & b_x & c_x & 0 \\ a_y & b_y & c_y & 0 \\ a_z & b_z & c_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = (a_x, a_y, a_z, 1) = A$$

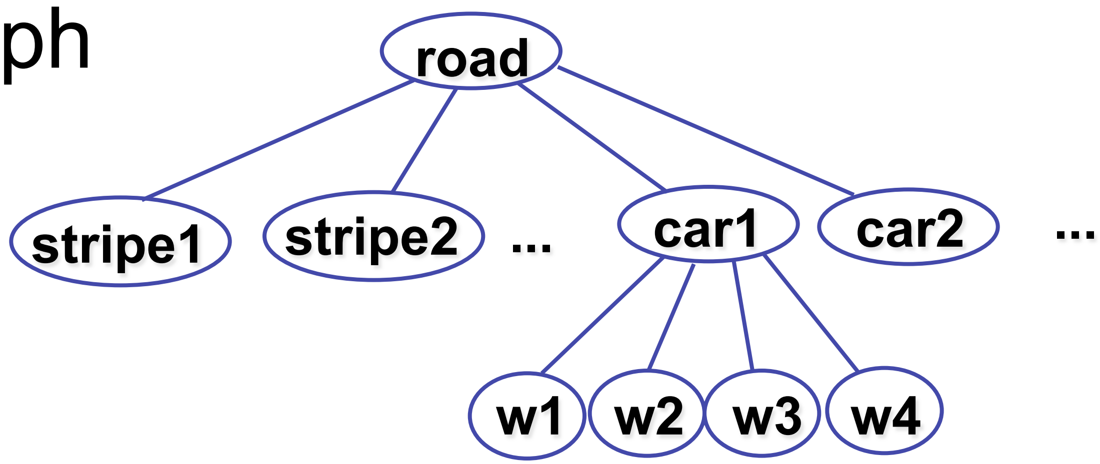
Transformation Hierarchies

Transformation Hierarchies

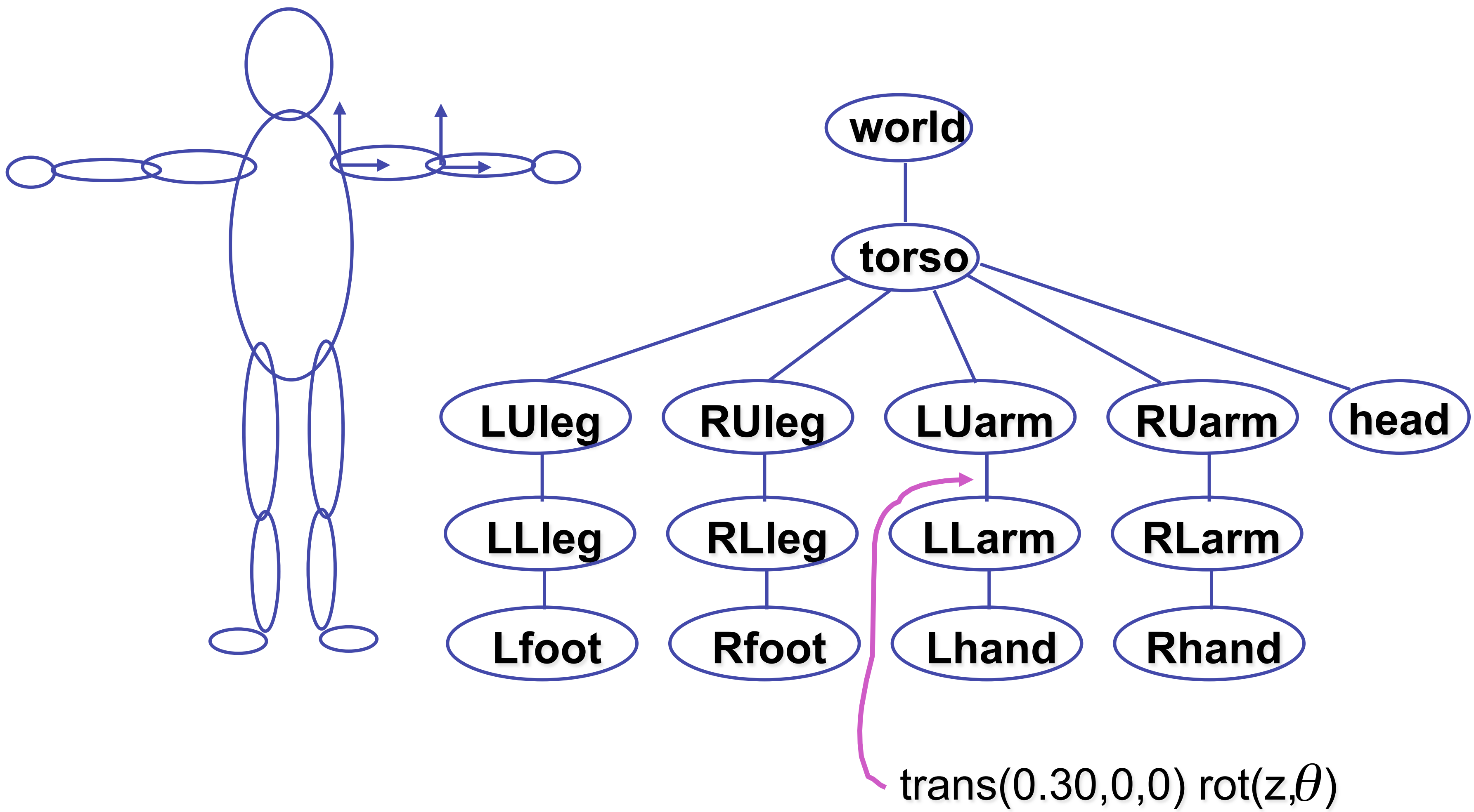
- scene may have a hierarchy of coordinate systems
  - stores matrix at each level with incremental transform from parent's coordinate system



- scene graph

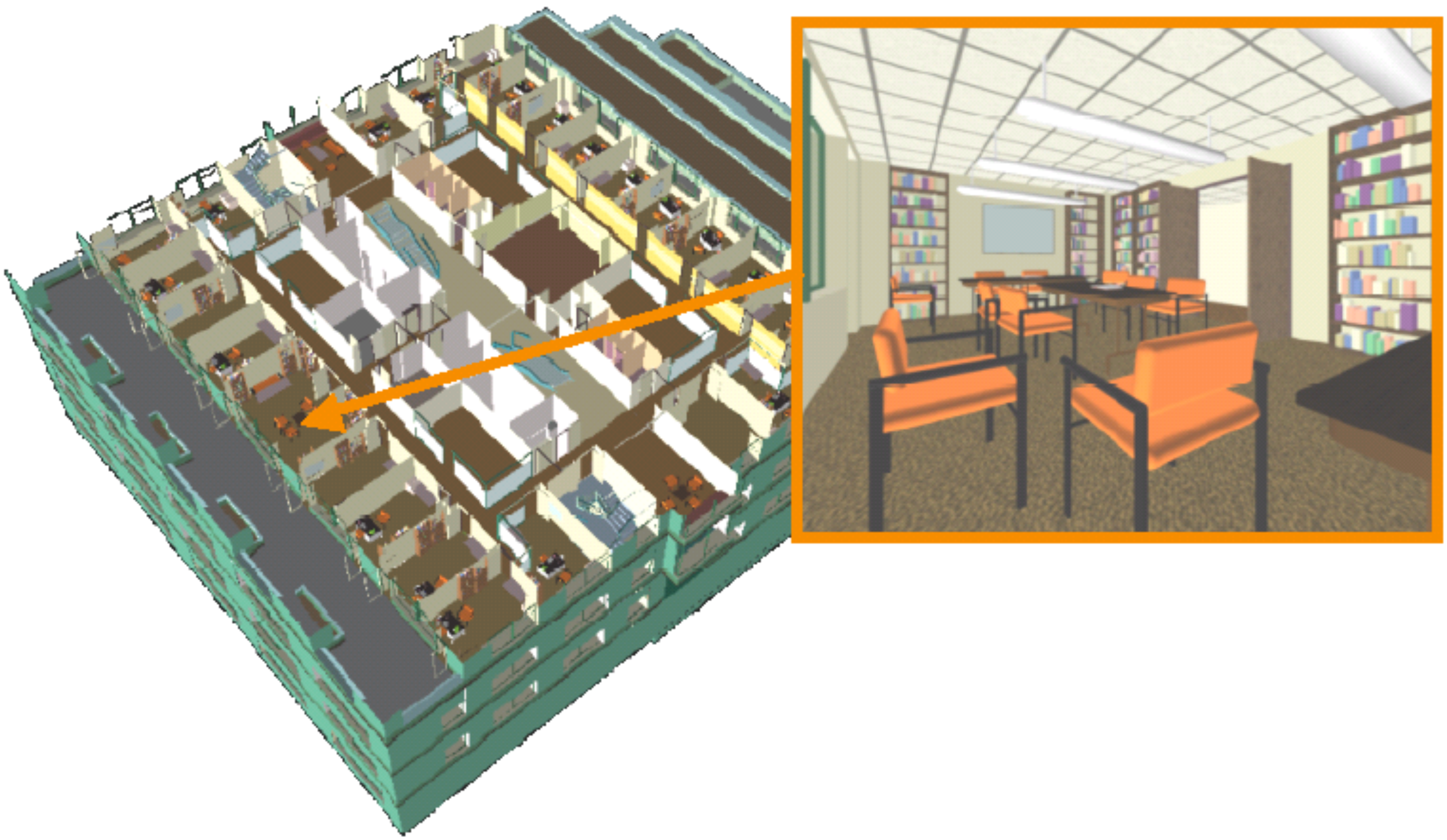


Transformation Hierarchy Example 1



Transformation Hierarchy Example 2

- draw same 3D data with different transformations: instancing



Matrix Stacks

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