Lecture 10

Announcements

Announcements

- Proposals delayed by 2 days, till after midterm
  - Now Thursday 2pm

- Remember: I have office hours!
  - 3pm till about 4:30 Weds
  - Engineering VI 372

Announcements

- Did you delete your .git folder?

Announcements

- Don't delete your git folder from its original place!
  - You’ll lose your entire history
  - GitHub Desktop won’t even navigate to it
  - If you never pushed, your history is gone forever
  - If you delete your other files, they’re gone forever

- I just meant don't include it in .zip!

Announcements

- Take advantage of your private repos!
  - A free backup solution (cloud storage)
  - Can’t lose everything

- After Proposals I'll give your *team* a free private repo.

Rest of Project Suggestions

Project Ideas - Further reading

Topics I probably won’t cover:

- Marching cubes (wiki) / drawing implicit fields & volumes
  - Videos
    - https://www.youtube.com/watch?v=MB9EuXzpCJA
    - https://www.youtube.com/watch?v=NG1gJvdCE4Q
    - https://www.youtube.com/watch?v=soWnbELQmCU
  - Level set method (wiki)
    - https://www.youtube.com/watch?v=IsHJhxGQ3wU

Project Ideas - Further reading

Topics I probably won't cover:

- Fractal generation & rendering – look up Mandelbulb videos
  - https://www.youtube.com/watch?v=Yb5MRbgNKSk
  - https://www.youtube.com/watch?v=cPbKP2ep05k
  - https://www.youtube.com/watch?v=KdJepLvW66U
  - https://www.youtube.com/watch?v=cPbKP2ep05k
- Mouse picking (clicking to correctly select objects)

Suggestion: Pick a “Theme” last

- Pick an advanced graphics topic, then:
  1. Implement it
  2. Game-ify it
  3. Theme it last

- If you do this, choose an advanced topic first.
  - Once it works, make it interactive or turn it into a challenge.

Suggestion: Pick a “Theme” last

- Alternatively, if you like modeling shapes more:
  - Build some models first
  - Theme based around whatever came out best

- If that’s more your thing, make 3D objects first, then tell a story about the objects you manage to make.
- Good stories are drawn from real events / existing media.
  - Even re-telling a story or re-implementing an established game can be worthwhile.

Midterm Review!

Some Topics

- Affine Transformations
- Coordinate Systems and Transformations
- Order of operations with Matrices
- Projection Transformations
- Understanding Lines and Points
- Operations with Vectors
- Programming

Some Topics

- Translate, Scale, Shear, Rotate
- How are they represented?
- What categories do they fall in?
- What effect do they have on objects?
- What effect do they have on angles?

Some Topics

- Projection Transformations
  - Given a point in x, y, z space, how do we calculate where it appears on the screen?
  - How is the perspective projection different from affine transformations?
  - What do perspective projections preserve?

Some Topics

- Coordinate Systems
  - Given a coordinate system and a point, can you give the coordinates of that point in the coordinate system?
  - If a matrix represents a coordinate system, what does it mean to multiply a point by the matrix?
  - Given basis vectors b_1,b_2,b_3, and origin O, how do we convert from the canonical coordinate system to the [b_1 b_2 b_3 O] coordinate system?
  - Given a picture of two coordinate systems, can you describe the operations that transform one to the other?

Some Topics

- How do we use an affine combination to find point C which is 20% of the way from A to B?
- How do we reflect a point across a line in 2D?
- What does a call to lookAt() make?

Some Topics

- If you understand the slides of every lecture, you should do well on the exam

Stacks

- In graphics, it’s common to declare stack objects to hold (and recall) old values of your matrix.

```
let model_transform = Mat4.identity();
let matrix_stack = [];
model_transform.post_multiply( Mat4.translation([ 1,0,0 ]) );
box.draw( model_transform );                              // box 1
matrix_stack.push( model_transform.copy() );
model_transform.post_multiply( Mat4.translation([ 1,0,0 ]) );
box.draw( model_transform );                              // box 2
model_transform = matrix_stack.pop();
model_transform.post_multiply( Mat4.translation([ 0,1,0 ]) );
box.draw( model_transform );                              // box 3
```

Stacks

- In graphics, it's common to declare stack objects to hold (and recall) old values of your matrix.
- If you're doing something you know you're going to undo, push your matrix to the stack first, and then pop it back out!
- Pop multiple times to go back multiple times in history
  - Example: Pop matrix once after drawing a dragonfly wing, to return to the tail segment to draw the next wing.
  - Pop the matrix twice to go back to the dragonfly's head, to get ready to draw antennae
    - Children of the head box

View Boxes

- View boxes always have six sides
- Anything inside the view box gets projected onto one of the planes
  - The one nearest to the camera
- Suppose a point is now projected onto the near plane somewhere. Where does it draw on the viewport?

Recall Non-Commutativity Example:

Matrix Multiplication is NOT commutative.

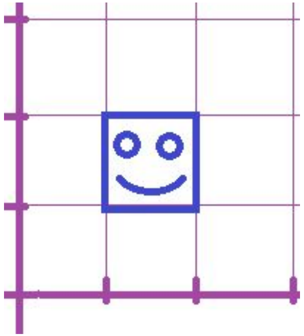
Remember our old rotation matrix:

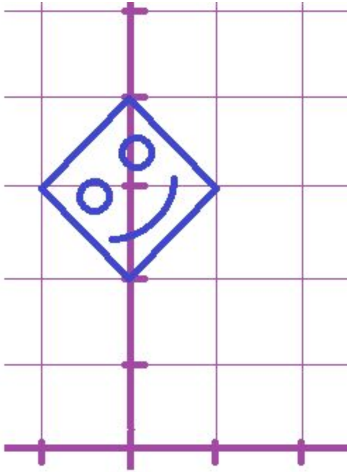
$$scale(\sqrt{2}) * rotate_z(45°) = ?$$

$$\begin{bmatrix} \sqrt{2} & \\ & \sqrt{2} \end{bmatrix} * \begin{bmatrix} \cos(45°) & -\sin(45°) \\ \sin(45°) & \cos(45°) \end{bmatrix} = ?$$

$$\Rightarrow \begin{bmatrix} \sqrt{2} & \\ & \sqrt{2} \end{bmatrix} * \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix} = ?$$

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

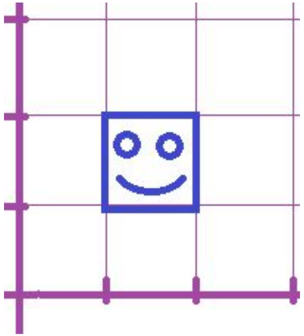




Matrix Multiplication is NOT commutative.

Suppose we modify it with a non-uniform scale matrix from the left:

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



Where do the corners of the face go if we use this one?

Matrix Multiplication is NOT commutative.

Suppose we modify it with a non-uniform scale matrix from the left:

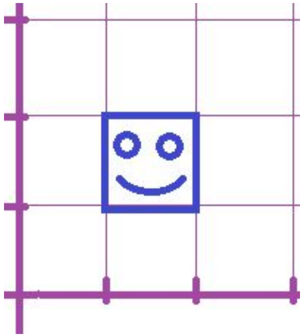
We sheared it!

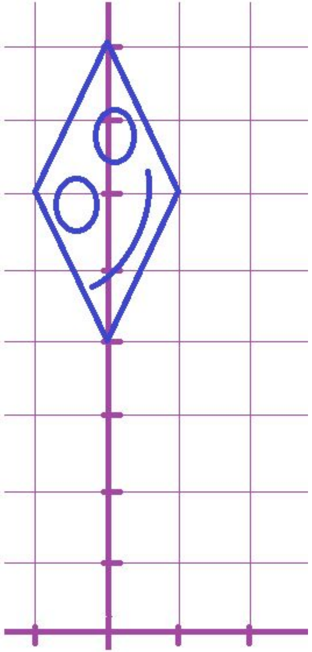
$$\begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix} * \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \end{bmatrix}$$



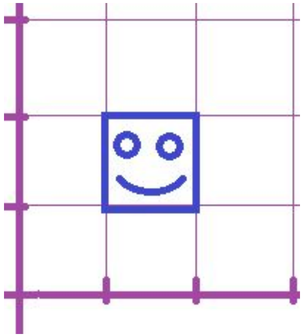


Matrix Multiplication is NOT commutative.

Suppose we modify it with a non-uniform scale matrix from the <u>right</u>:

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

Where do the corners of the face go if we use this one?



Matrix Multiplication is NOT commutative.

Suppose we modify it with a non-uniform scale matrix from the <u>right</u>:

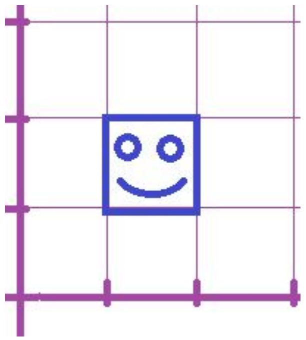
We didn't shear it!

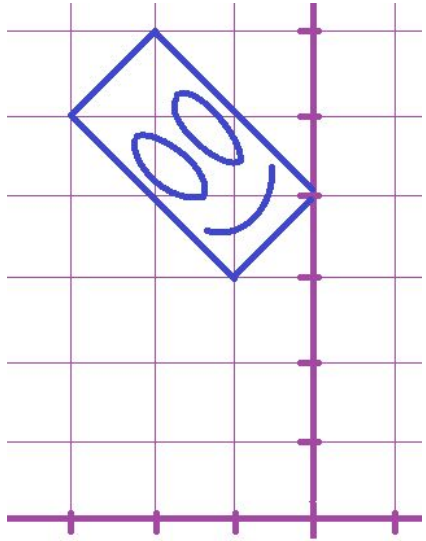
$$\begin{bmatrix} 1 & -2 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ 6 \end{bmatrix}$$





Confirm it with the Bases Game

http://bases-game.glitch.me

The Pumpkin Problem

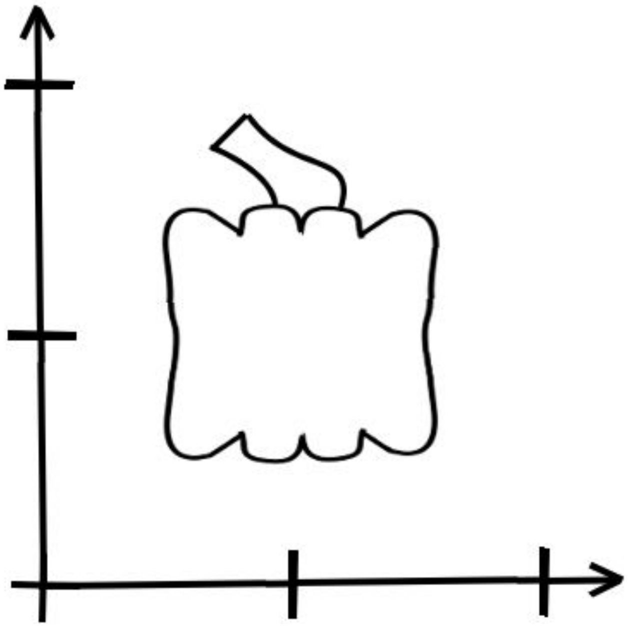
Transformations practice problem

Three Ways To Do Every Transformation Problem:

- 1. Intuition using moving bases (or axes)
  - Reading typical code forwards
  - Reading written product left-to right ending at p
  - Products formed via post-multiplication
- 2. Intuition using a moving point cloud (or shape)
  - Reading typical code backwards
  - Reading written product right-to-left starting at p
  - Products formed via pre-multiplication
- 3. Writing the product out, doing matrix multiplication by hand, and not relying on intuition at all

Drawing example: Pumpkin

Given this pumpkin at (1,1),



Drawing example: Pumpkin

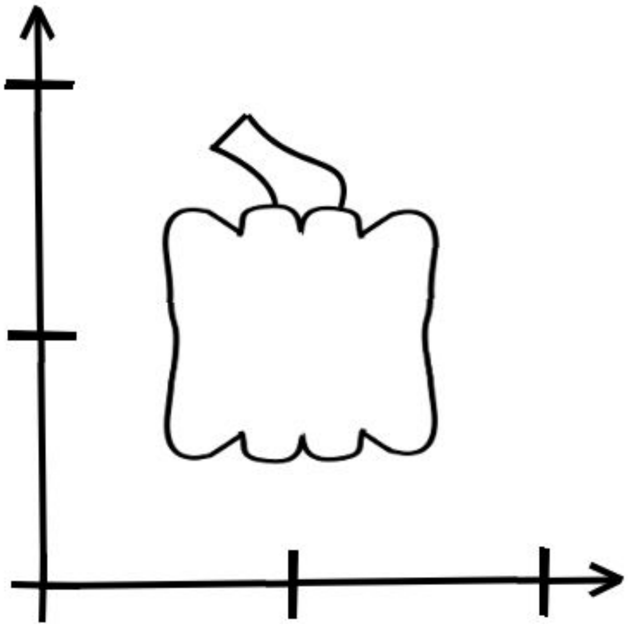
Given this pumpkin at (1,1), do the following:

model *= trans(x+2,y+2);

model *= $\text{rot}_z(90)$;

model *= $\text{scale}_x(-1)$;

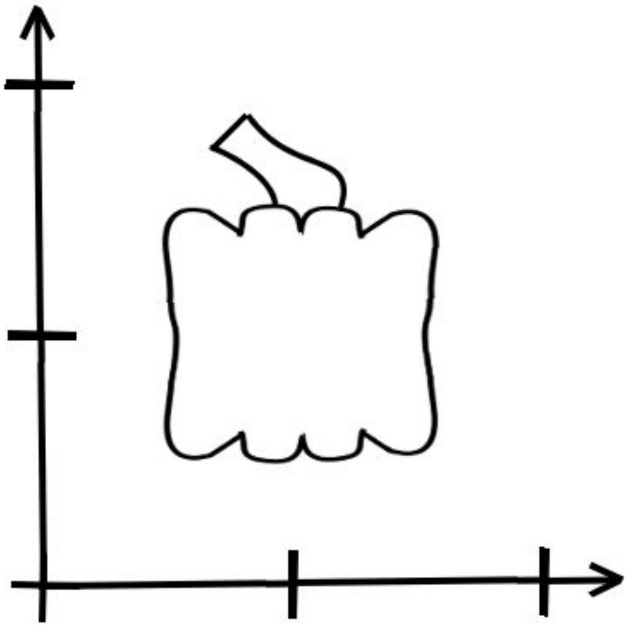
model *= trans(x-1,y-1);



Drawing example: Pumpkin

Given this pumpkin at (1,1), do the following:

trans(2,2) * rot_z(90) * $scale_x$(-1) * trans(-1,-1)

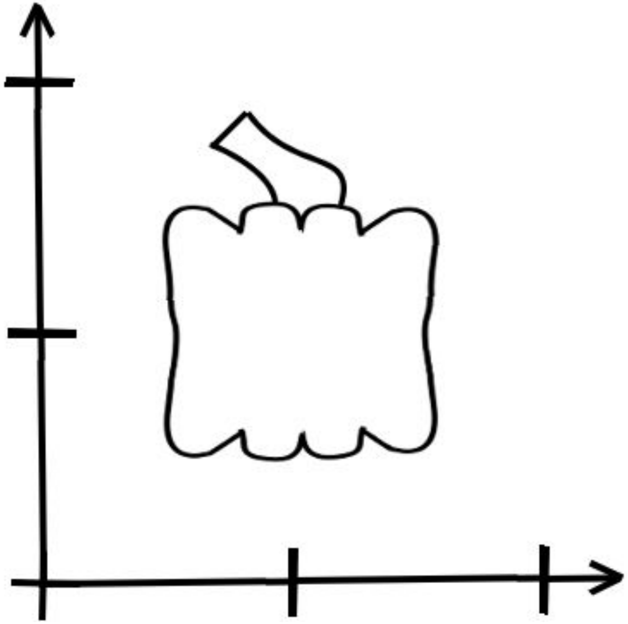


Drawing example: Pumpkin

Manually writing the product of matrices

trans(2,2) * rot_z(90) * $scale_x$(-1) * trans(-1,-1)

= what actual matrices?

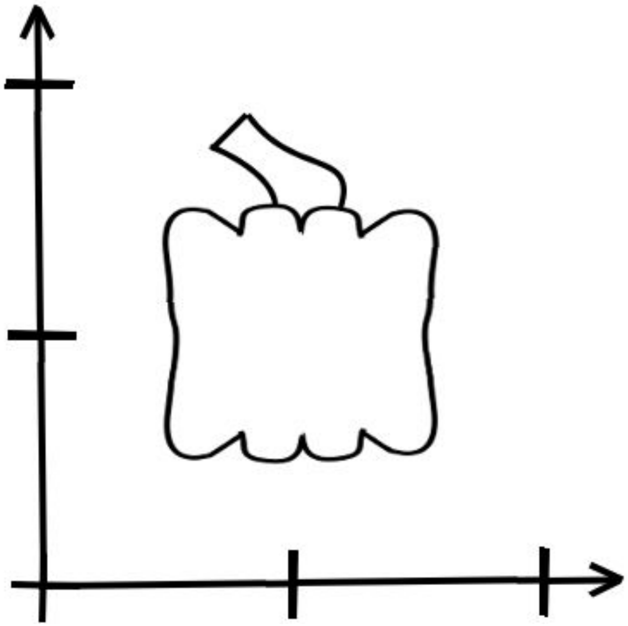


Drawing example: Pumpkin

- Manually writing the product of matrices

trans(2,2) * rot_z(90) * $scale_x$(-1) * trans(-1,-1) = ?

- Multiply out the product with the “drawing below” trick
- Apply the final product to some points (0,0), (0,2), (2,0)

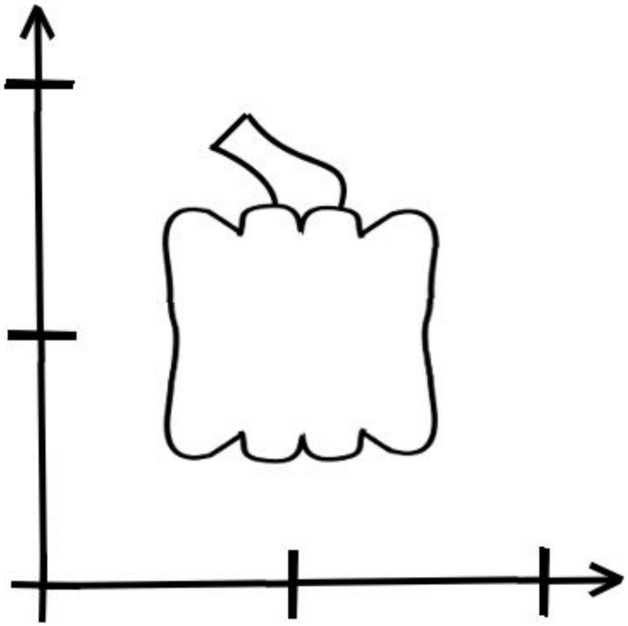


Drawing example: Pumpkin

- **Step 1: Points Perpective**
- Actually draw out where the pumpkin moves at each step of

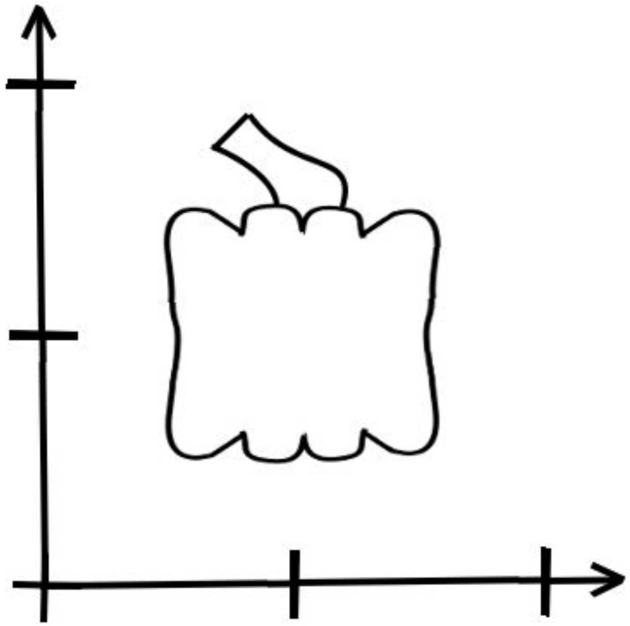
$$trans(2,2) * rot_z(90) * scale_x(-1) * trans(-1,-1)$$

- We're treating it like an image -> Start at point and move Right-to-Left
- Show that where it landed is consistent with where the product displaced the 3 points to



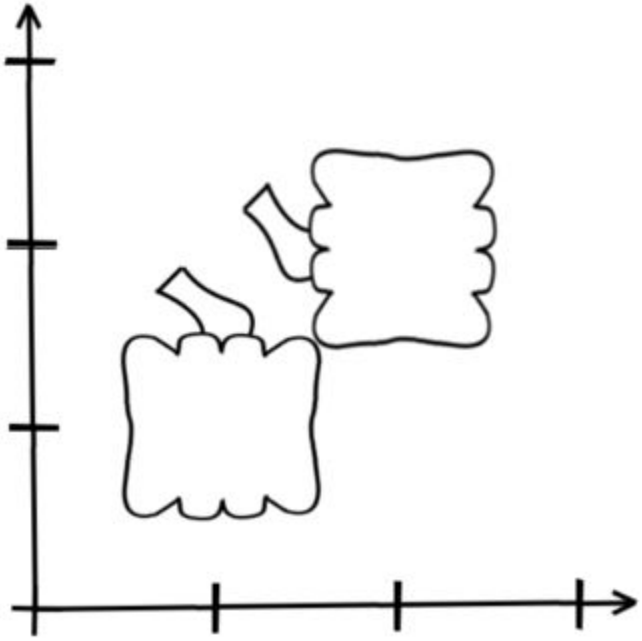
Drawing example: Pumpkin

- **Step 2: Bases Perspective**
- Actually draw out where a basis would move at each step (go left-right, maintain a basis as your temporary instead of a point)
- Wherever the origin winds up, draw the original image there using those axes

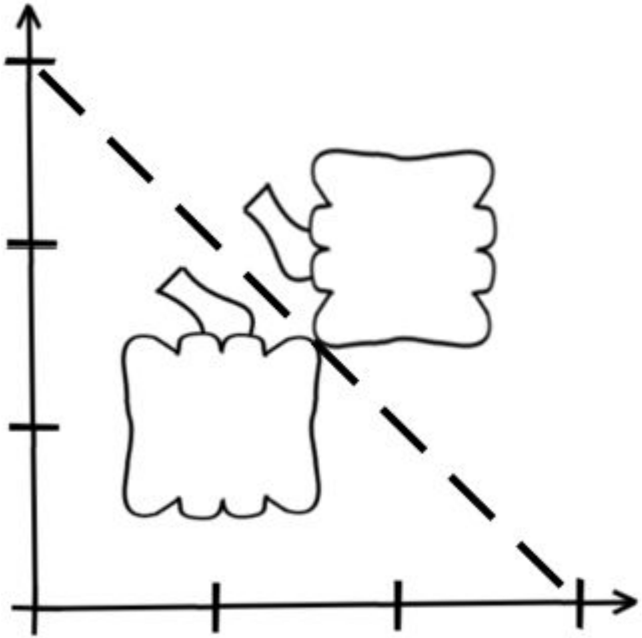


Drawing example: Pumpkin

- Why do we prefer left to right when building programs?
- Because of our temporary "partial matrices" when making the various products
  - Each sets us up for the next piece of a hierarchical model



Checking our Answer



Checking our Answer

- Easily summarized as a reflection around a line from (3,0) to (0,3)
- The sequence of transforms to do that reflection is different:
  - trans(0,3) * rot_z(-45) * $scale_y$(-1) * rot_z(45)* trans(0,-3)
  - What's the code for this?
- Numerically multiplying it out, it was the same matrix, surprise!!!