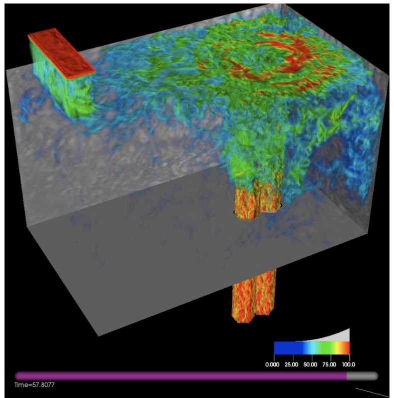
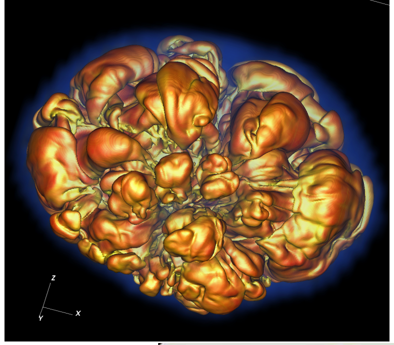
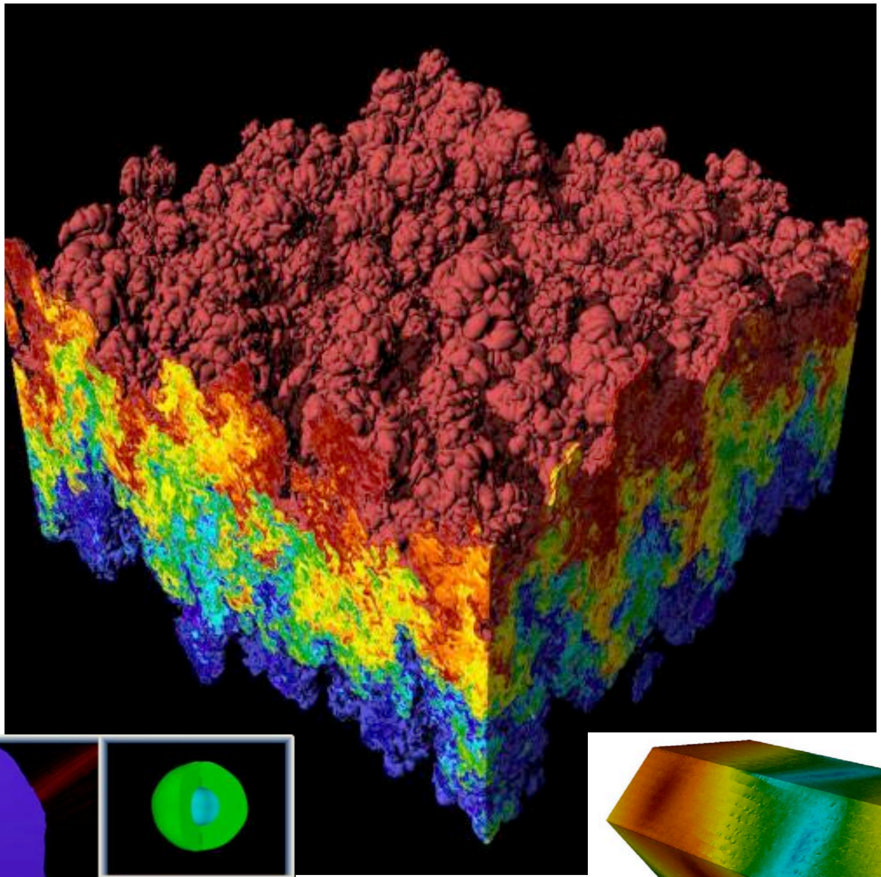
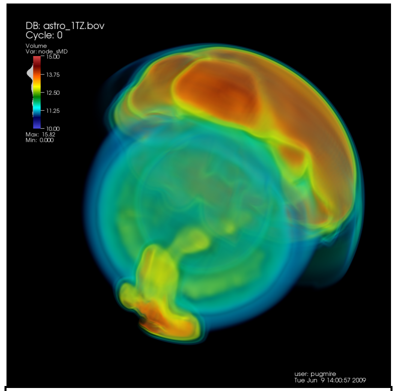
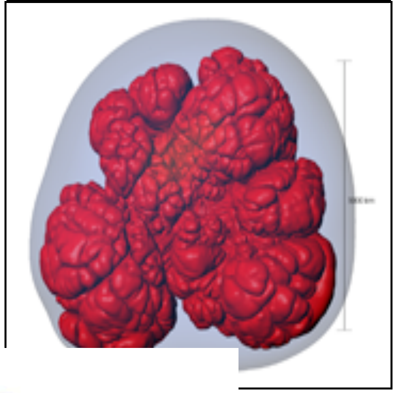
CIS 441/541: Intro to Computer Graphics
Lecture 15: Mirrors

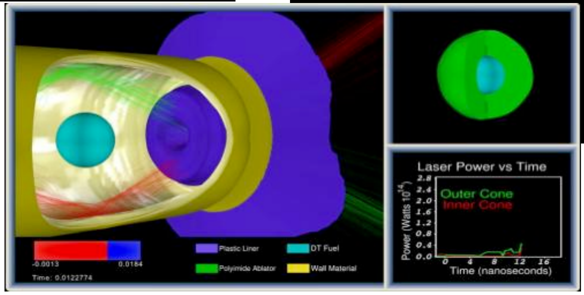


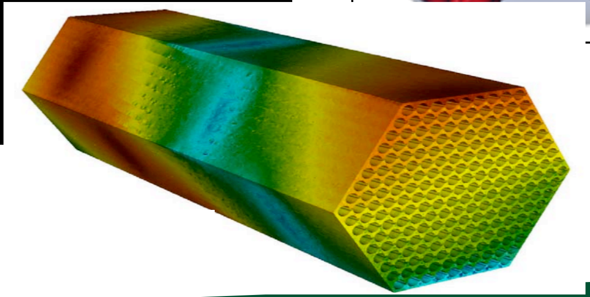












June 1, 2021

Hank Childs, University of Oregon

Office Hours

- Now only 1 OH per week
  - Hank, Weds 230-330
- Abhishek still doing a lot...

Plan for Thursday

- 830-910am: group Q&A on final projects
- 910-915: discuss how makeup quiz works
- 915-945: makeup quiz

- Don’t want group Q&A?: just call in at 910am
- Don’t want makeup quiz?: drop off at 910am
- Don’t want either?: just skip on Thursday

Makeup Quiz

- Can re-take any of previous quizzes
- Score can replace previous score
  - New score is higher: use new score
  - Old score is higher: use old score
- Will not be on Canvas – just email Hank
- Probably will not be graded until after Tuesday June 8th

3.X

- 3A: 1D textures
- 3B: collisions
- 3C: level of detail
- 3D: mirrors (worth double)
- 3E: physically-based rendering (worth double)

- Plans to get 3 projects:
  - 3A+3B+3C
  - 3A+3D
  - 3B+3E
  - Etc.

Final Projects

- Tuesday June 8th @8am, 2021
  - $\rightarrow$ We will start at 8am, not 815am
- Via class Zoom
- -4 points if you do not attend (presenting or not)
- Presentations: 3 minutes

What to Turn In: Custom Projects

- Tarball/zip
- “README” is appreciated
  - Where you spent your time
  - What you are proud of
  - What files to look at
- Evidence of your efforts
  - Code you wrote
  - For art assets, it may be screenshots and not digital uploads
  - PowerPoints from your presentations
  - I do not plan to compile/run these

What to Turn In: Project 3.X

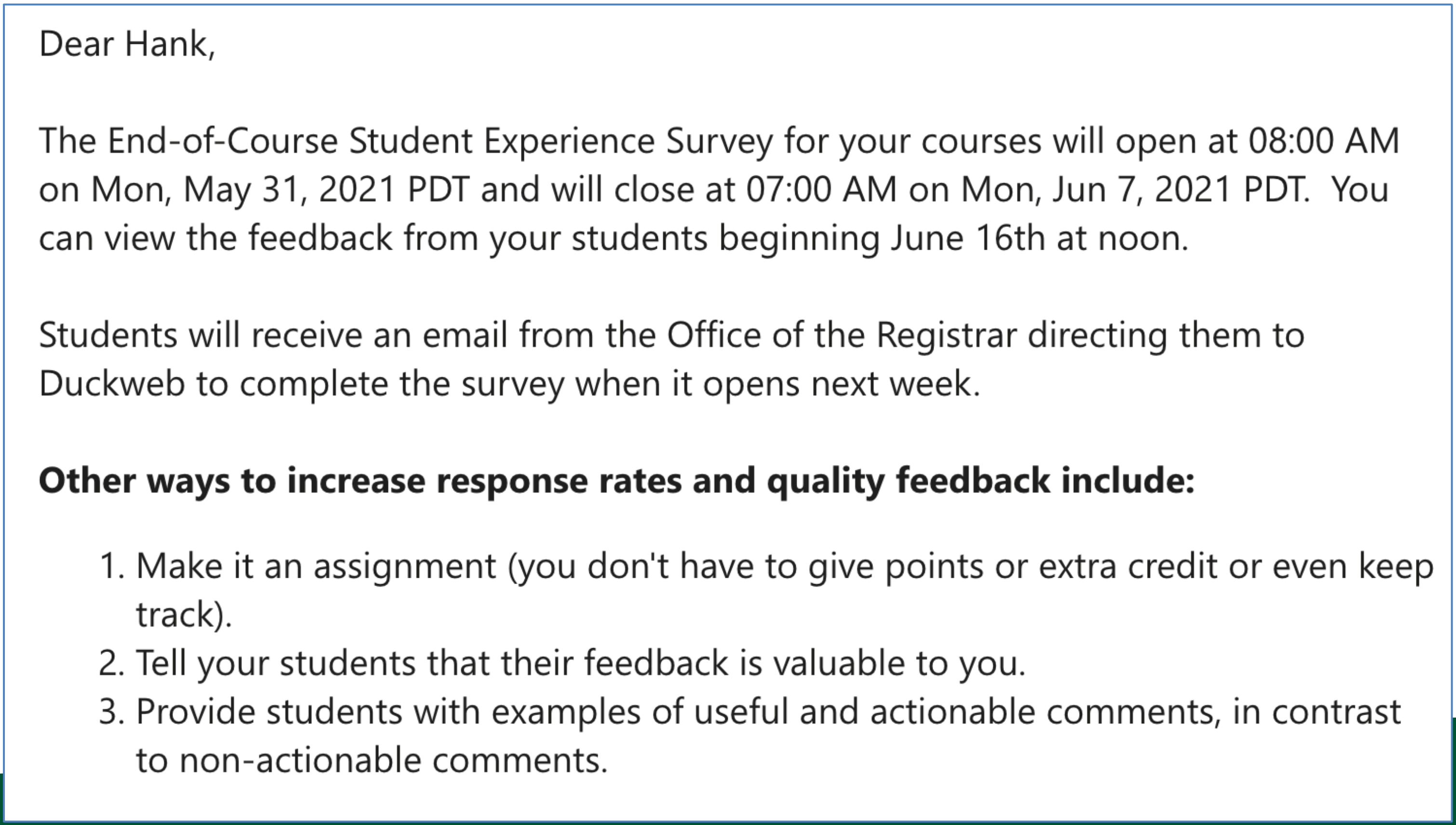
- Tarball/zip
- One directory per project
- Example:
  - Directories for 3A, 3B, 3C
- Each directory should contain your source code
- In some cases, a report
- I do plan to compile/run these

Submission Details

- Canvas entry called “Final Project”
- Due at 8am Tuesday June 8th
  - Should be submitted before the presentations start
- <u>I disabled submissions after 8am</u>
  - I do not want to deal with situations where people upload them late and pretend they were not late

End-of-Course Student Experience Surveys

- Long, but important



Dear Hank,

The End-of-Course Student Experience Survey for your courses will open at 08:00 AM on Mon, May 31, 2021 PDT and will close at 07:00 AM on Mon, Jun 7, 2021 PDT. You can view the feedback from your students beginning June 16th at noon.

Students will receive an email from the Office of the Registrar directing them to Duckweb to complete the survey when it opens next week.

Other ways to increase response rates and quality feedback include:

1. Make it an assignment (you don't have to give points or extra credit or even keep track).
2. Tell your students that their feedback is valuable to you.
3. Provide students with examples of useful and actionable comments, in contrast to non-actionable comments.

(Slide I presented last week) Grading

- Generally doing good
  - 2A graded yesterday
  - 1A-1E, Quiz1 – Quiz4 all graded
- To do
  - 1F: hoping for this weekend
  - 2B: next few days
  - Quiz 5: by Tuesday for sure

Late Passes

- I will apply them, and connect with those who have decisions
  - (Still haven’t done this yet)
  - (Need to get 1F graded)

Github

- I know many of you want to put these on github
- No 1A-1E – what's the point?
- 1F: please remove phrases like CIS441, ceil__441, floor__441, etc.
- Description:
  - A software-based computer graphics system that renders imagery via rasterization, including Phong shading, hidden surface remove, and arbitrary camera positions .

My Favorite 2B's

Brian Gunnarson

Seth Tal

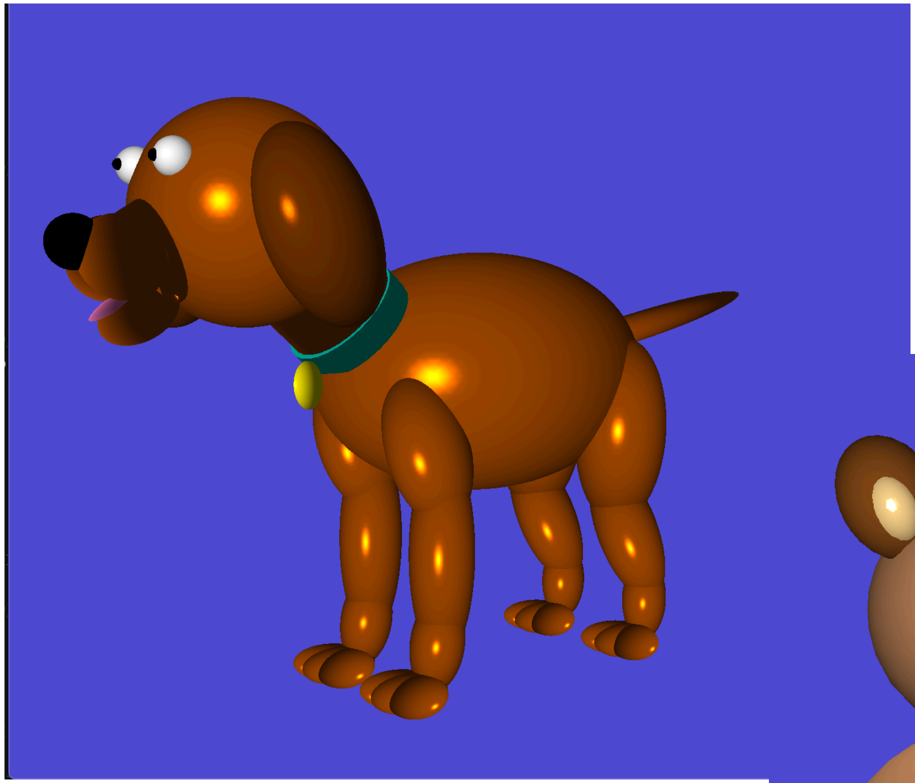
Joe Johnson

Simon Rosenthal

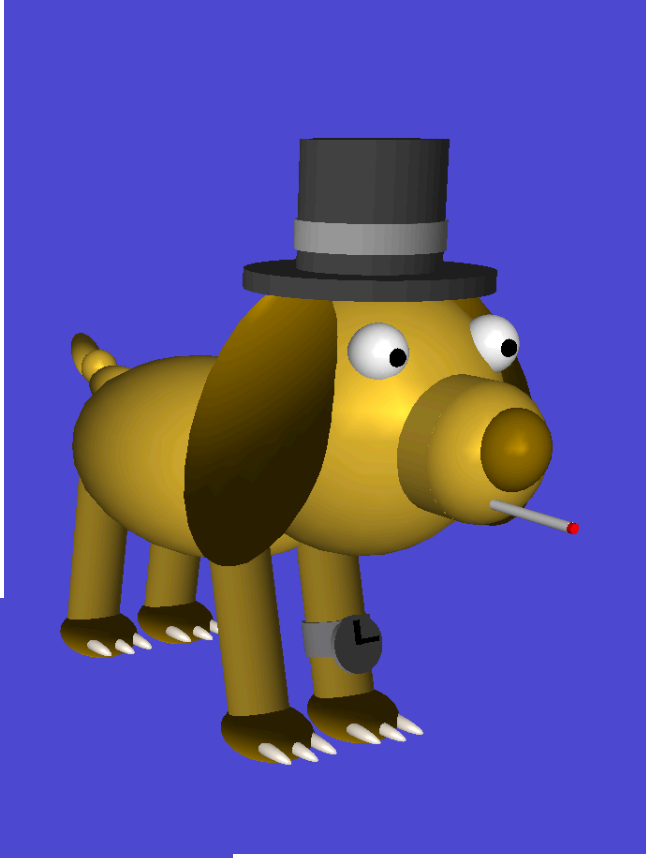
UNIVERSITY OF OREGON

Stephen Leveckis

My Favorite 2B's



Evan Podrabsky



Max Terry

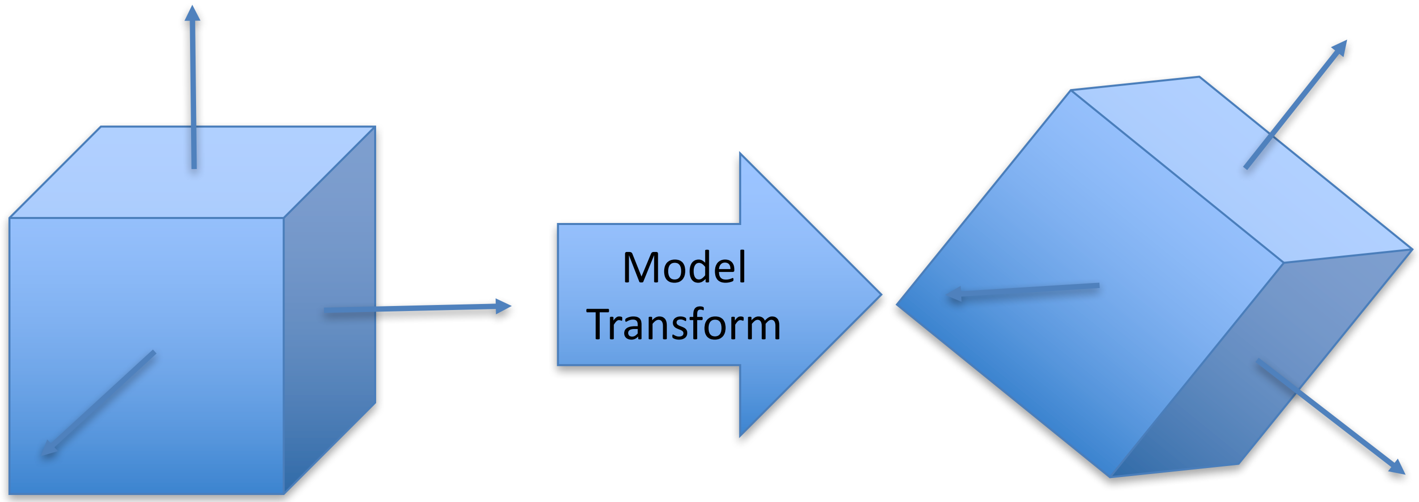
My Favorite 2B’s

- Raul Patel
- Brandon Bower

Problem with 2B – Whoops!

- Lighting was wrong in some cases
- Why?...

Problem with 2B



A cube object.
Normals for each
face shown with
an arrow.

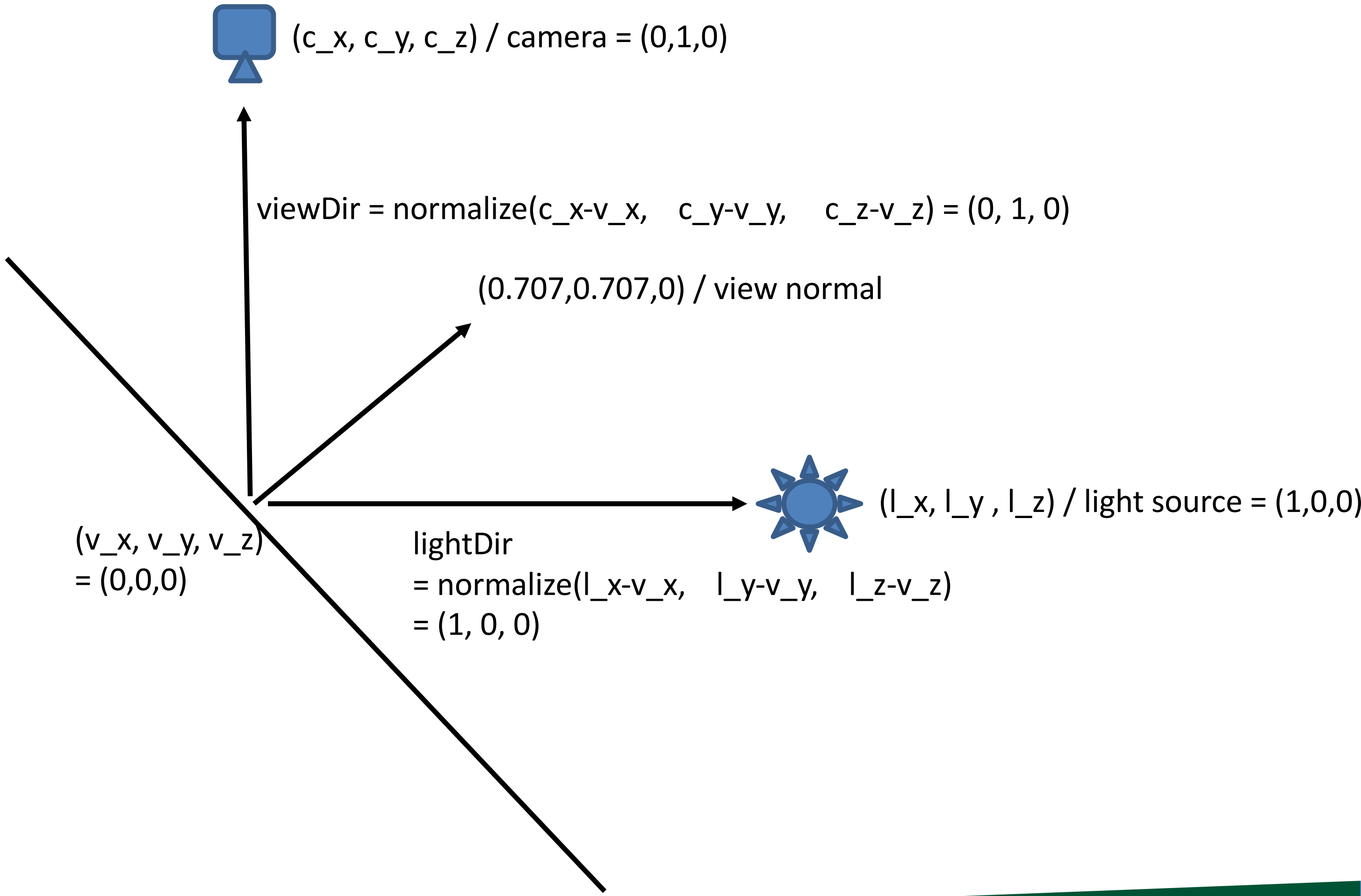
A cube object
after a rotation.
Normals correctly
rotated with faces

Assume the left is a “source” object and the right is the source object being transformed for a scene

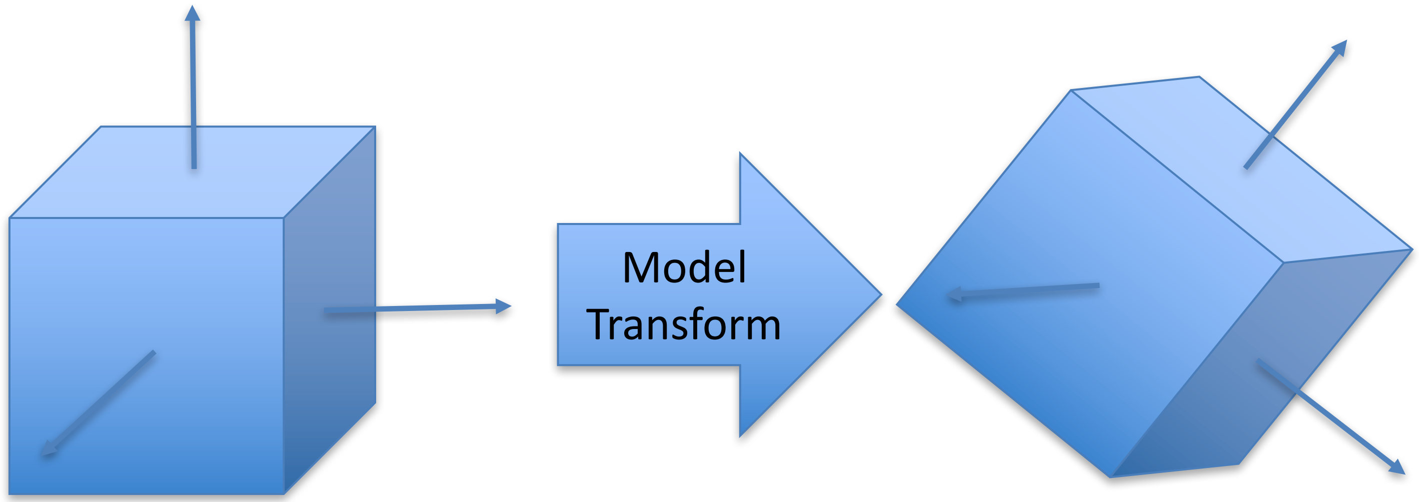
Reminder: Lighting Calculation Happens in World Space

This example has a triangle vertex, v, at the origin, the camera one unit along the Y-axis and the light source one unit along the X-axis.

The lightDir and viewDir formulas show the conventions we should use for direction for general positions.


(c_x, c_y, c_z) / camera = (0,1,0)
viewDir = normalize(c_x-v_x, c_y-v_y, c_z-v_z) = (0, 1, 0)
(0.707,0.707,0) / view normal
(l_x, l_y , l_z) / light source = (1,0,0)
(v_x, v_y, v_z) = (0,0,0)
lightDir = normalize(l_x-v_x, l_y-v_y, l_z-v_z) = (1, 0, 0)

So Which Cube Is In World Space?

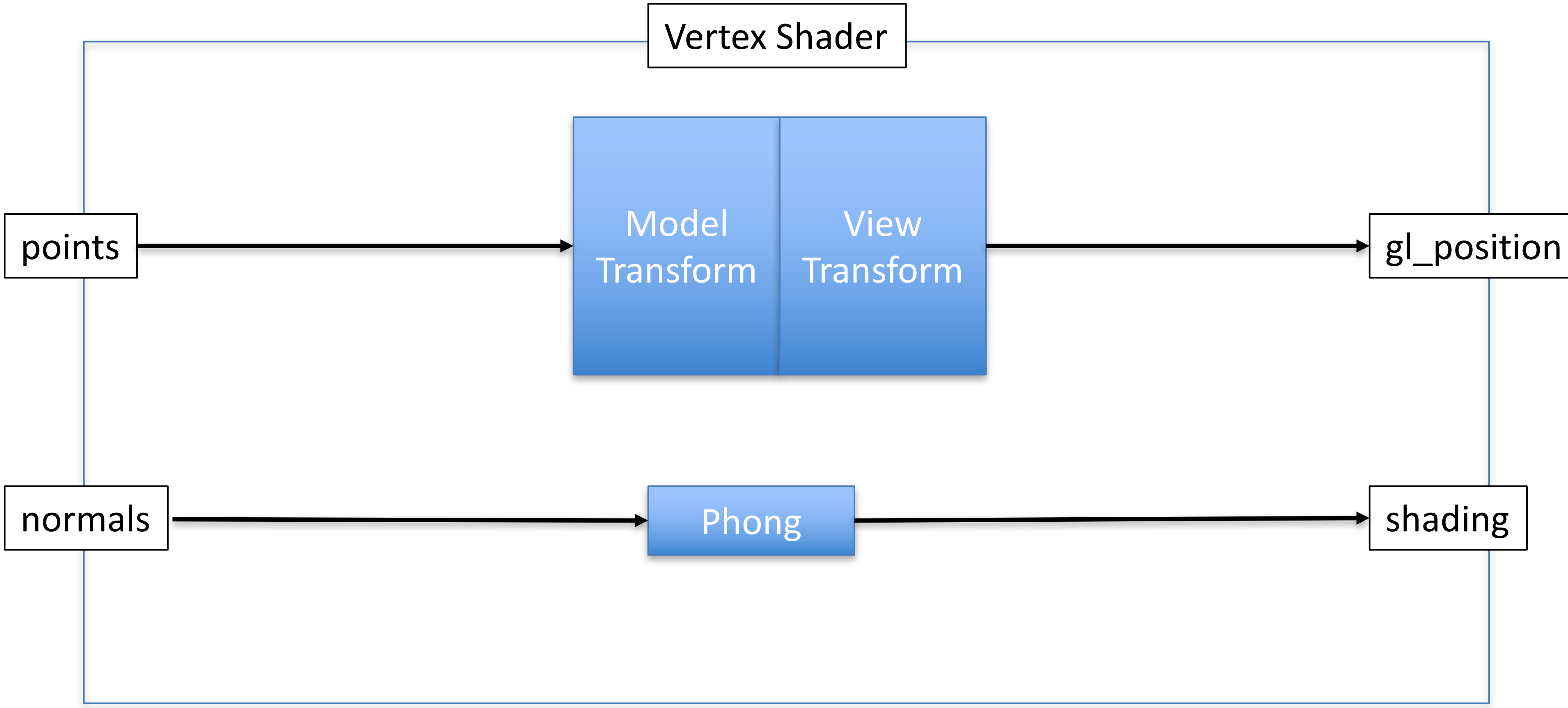


A cube object.
Normals for each
face shown with
an arrow.

A cube object
after a rotation.
Normals correctly
rotated with faces

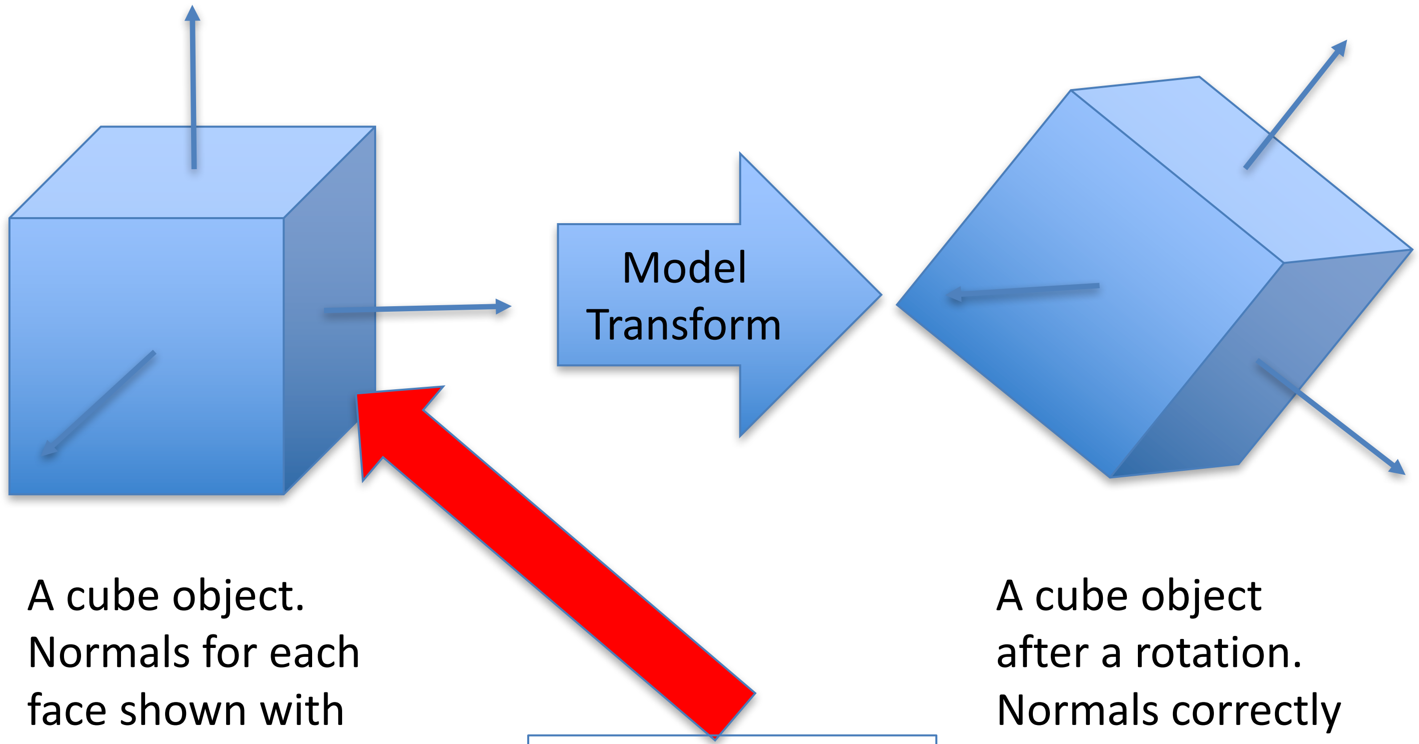
Answer: the right one. It represents how we want the object to appear in world space.

When Do We Do Lighting?



This is bad. Model is changing points, but not normals (see next slide)

So Which Cube Is In World Space?

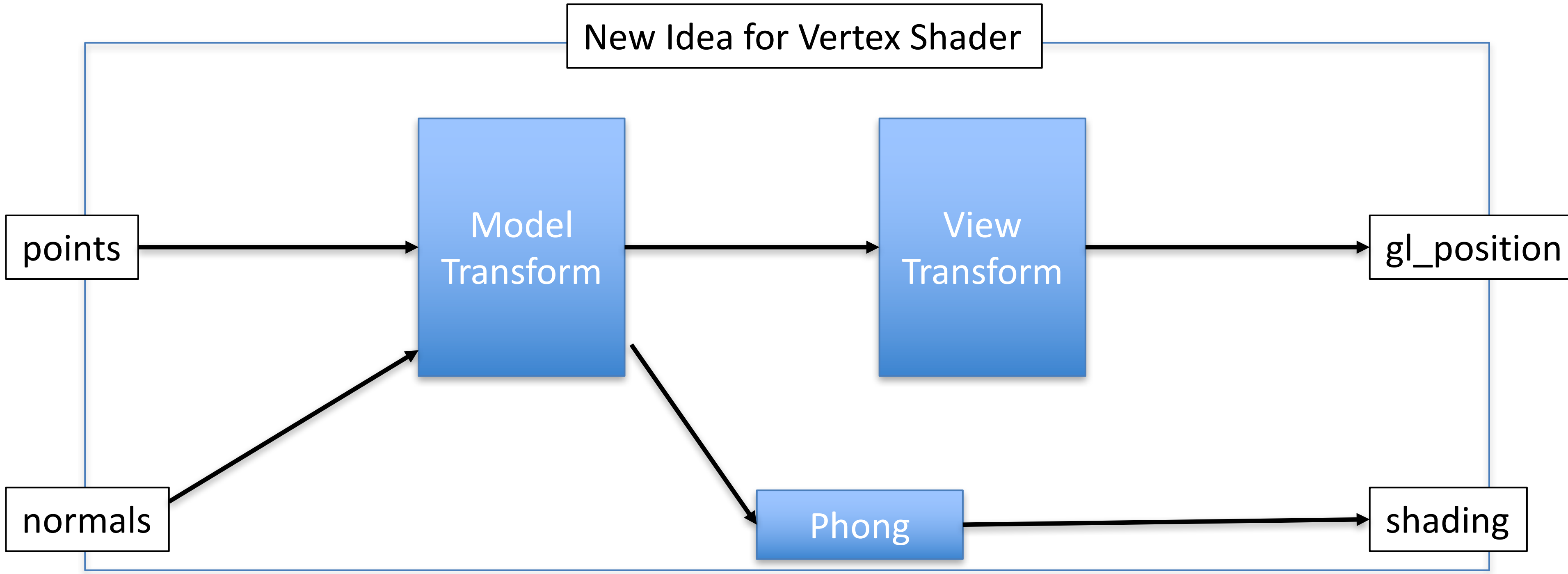


A cube object.
Normals for each
face shown with
an arrow.

<u>We are doing
Phong shading on
these normals</u>

A cube object
after a rotation.
Normals correctly
rotated with faces

Idea: Separate Model and View



Now need to send in model and view separately.
And modify vertex shader to transform normals

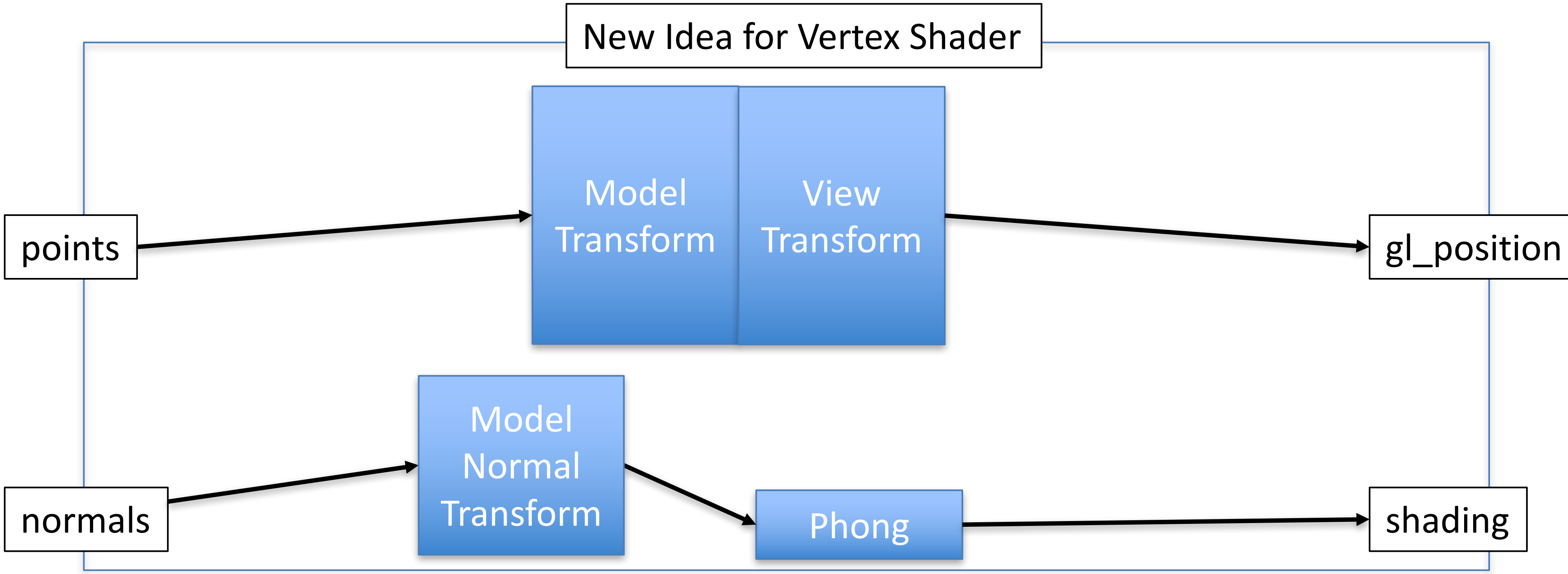
But there is another problem

Transforms and Normals

- Can a normal be rotated: yes
- Can a normal be scaled: yes
- Can a normal be translated: no
- Example:
  - Normal: (1,0,0)
  - Translation: (0,1,0)
  - Normal after translation: (1,1,0)
  - But still want it to be (1,0,0)

So separating model and view (idea from last slide) would work except translation

Idea #2: Generate Transform for Just Models



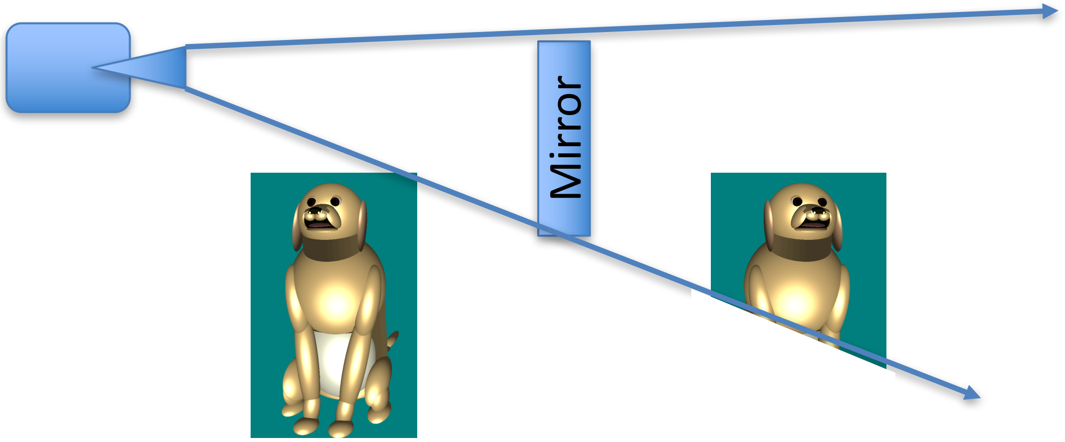
Now can send in modelview as one matrix.
But need a second matrix for normal transforms
Second matrix much like model transform, but without translations.

Notes

- There is more to this topic:
  - Apparently there are tricks where you can multiple by the upper left 3x3 of a 4x4 matrix and it will discard translation
  - Some folks do lighting calculations in “eye space” and this allows for different formulations

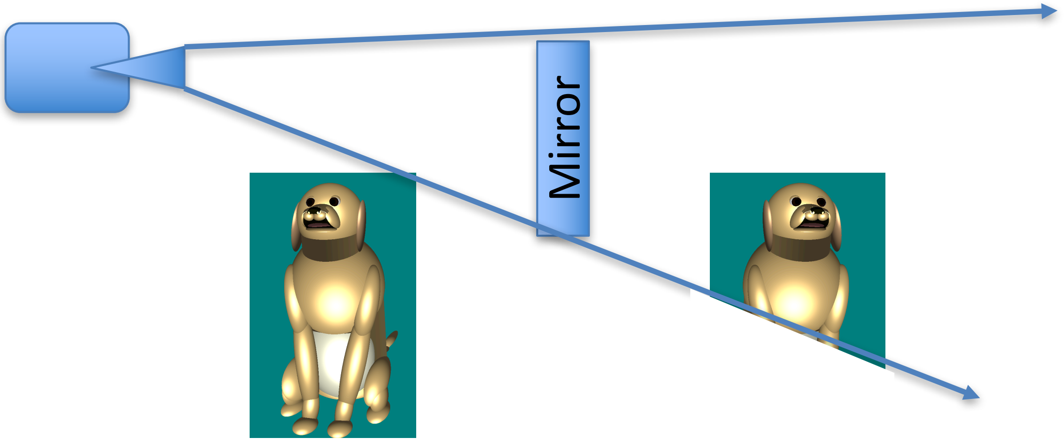
Mirrors

- Two big ideas:
  - Reflecting geometry across plane
  - Limiting rendering of reflections to be only within mirror



Mirrors

- Two big ideas:
  - – Reflecting geometry across plane
  - – Limiting rendering of reflections to be only within mirror



Reflecting Geometry Across a Plane

- Assume there is a 4x4 matrix that does reflections (there is)
- Then reflecting geometry looks like:

```
glm::mat4 identity(1.0f);
RenderScene(identify);
Glm::mat4 reflection = ...;
RenderScene(reflection);
```

What About Two Mirrors?

```
glm::mat4 identity(1.0f);
RenderScene(identify);
glm::mat4 reflection1 = ...;
RenderScene(reflection1);
glm::mat4 reflection2 = ...;
RenderScene(reflection2);
```

But what about objects reflection off both mirrors?

What About Two Mirrors?

```
glm::mat4 identity(1.0f);
RenderScene(identify);
glm::mat4 reflection1 = ...;
glm::mat4 reflection2 = ...;
RenderScene(reflection1);
RenderScene(reflection2);
RenderScene(reflection2*reflection1);
RenderScene(reflection1*reflection2);
```

And three mirrors?

The Mirror Problem

- One idea: use ray tracing
- Another idea: just limit the number of mirrors
  - Common in practice!
- Another idea:
  - Limit the number of reflections

So What Is the Reflection Matrix?

- Easiest reflection: across plane Z=0
  - (or plane X=0 or plane Y=0)

```
      (1 0 0 0)
M=  (0 1 0 0)
      (0 0 -1 0)
      (0 0 0 1)
```

And: M*(x y z 1) = (x y –z 1)

What If I Don't Want to Reflect Across XY, YZ, or XZ?

- Answer: use more matrices to transform the problem to the simple one
- Example:
  - Want to transform across plane Z=4
  - Three steps:
    - Translate by (0,0,-4)
    - Reflect across plane Z=0
    - Translate by (0,0,4)
  - This is three matrices. Compose them.

Making a Combined Matrix

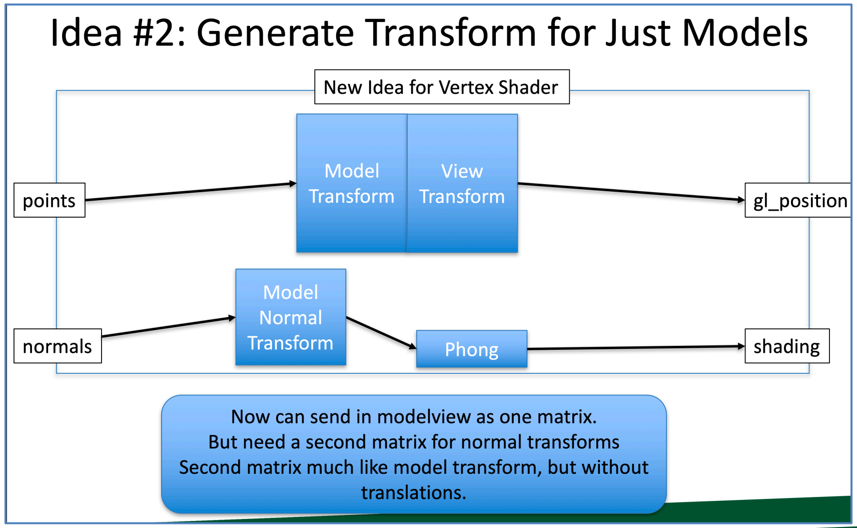
$$\underbrace{\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 4\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\text{Step 3: translate by Z=4}} * \underbrace{\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\text{Step 2: reflect across Z=0}} * \underbrace{\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & -4\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\text{Step 1: translate by Z=-4}} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 8\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Transforming Point (3,3,2)

(1 0 0 0) * (3 3 2 1) = (3 3 6 1)
(0 1 0 0)
(0 0 -1 8)
(0 0 0 1)

Normals

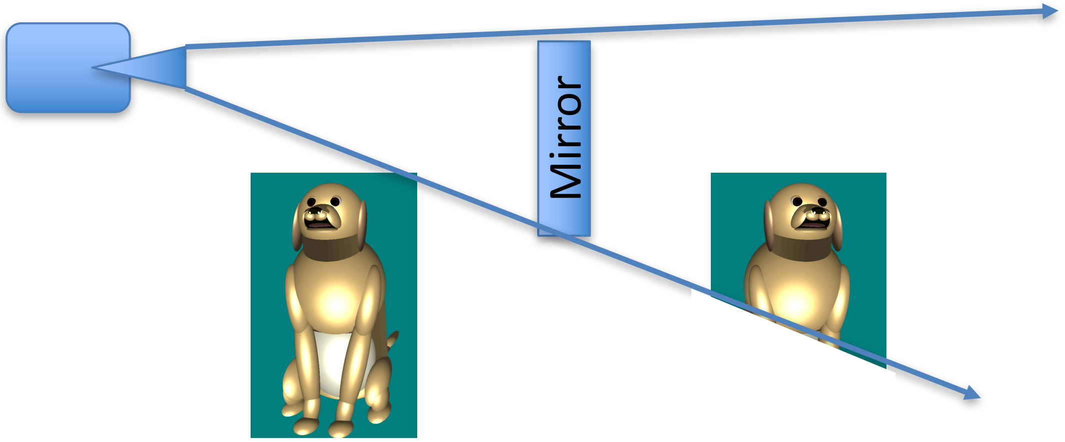
- Will need to reflect normals as well
- Plays well with previous discussion
  - (and normals can be reflected)



Now can send in modelview as one matrix.
But need a second matrix for normal transforms
Second matrix much like model transform, but without translations.

Mirrors

- Two big ideas:
  - Reflecting geometry across plane
  - Limiting rendering of reflections to be only within mirror

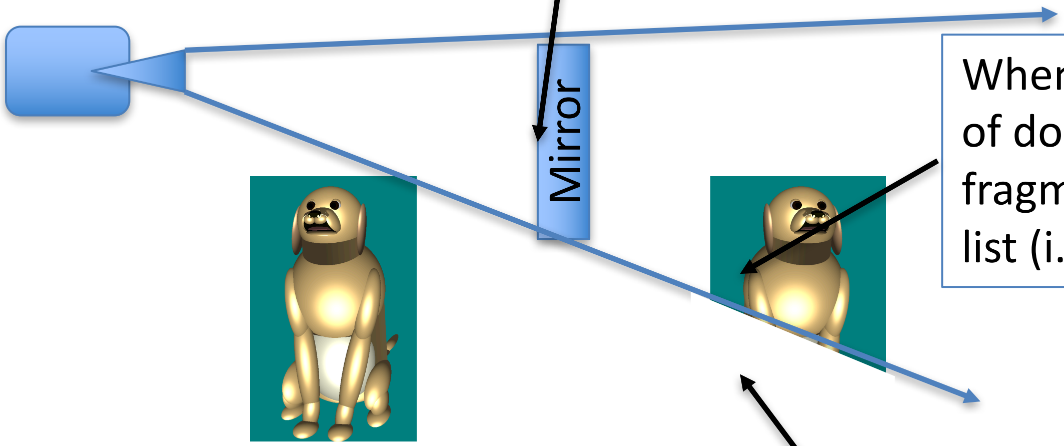


The Big Idea

- (1) render the geometry for the mirror
- (2) take note of which pixels the mirror overlaps with
- (3) when rendering reflected geometry, only accept the fragment if it overlaps with the mirror (consulting with data from (2))

Mirrors

The mirror overlaps with some list of pixels, P1, P2, P3, …, PN.
Calculate this list by rendering mirror first.



When rendering reflection of dog, accept this fragment since it is on the list (i.e., is some P_i).

When rendering reflection of dog, reject this fragment since it is not on the list (i.e., not any of the P_i).

OK, this plan (sort of) makes sense.
But how do we instruct GPU to do this?

New Idea: Stenciling

- Definition from the internet:
  - 1: a piece of material (as a sheet of paper or plastic) that has lettering or a design cut out and is used as a guide (as in painting or drawing)
  - 2: a pattern, design, or print produced with a **stencil**.

Image credit: stencilrevolution.com

New Concept in GPU: Stencil Buffer

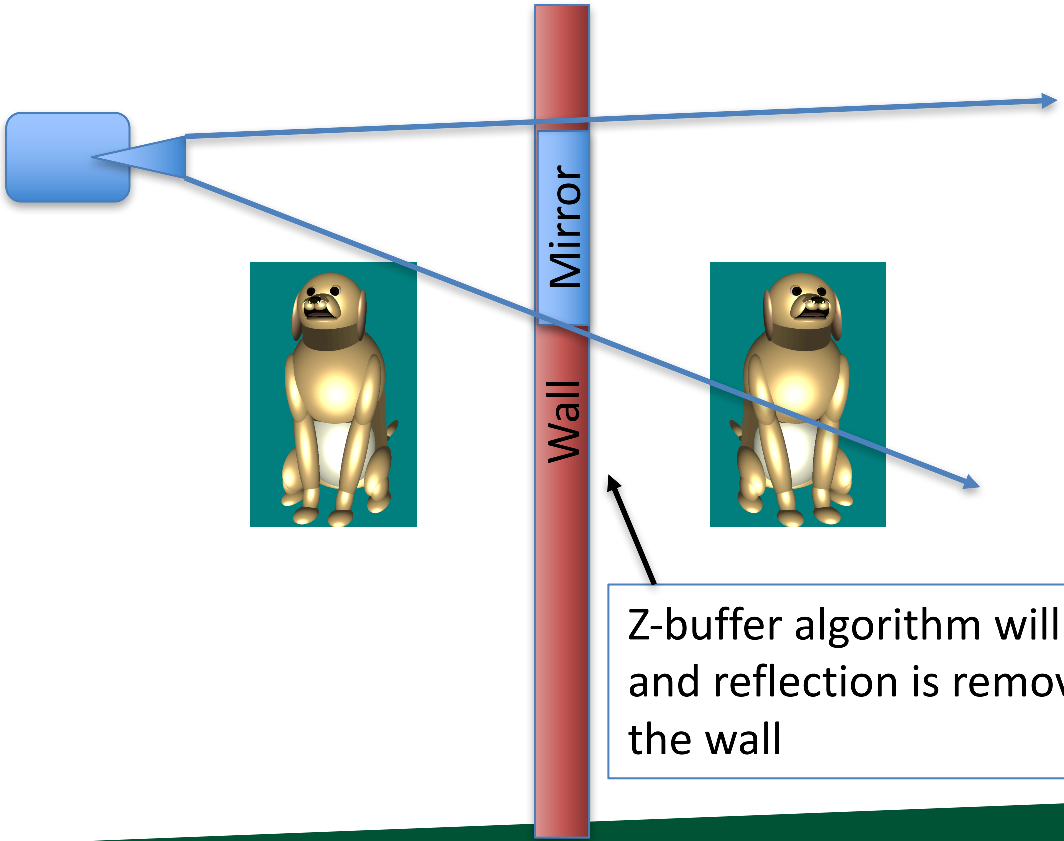
- Stores a byte for every pixel
- Can be used in many ways
- For this use case:
  - Let 1 mean that the pixel overlaps with the mirror
  - Let 0 mean that the pixel does not overlap with the mirror

Using the Stencil Buffer for Mirrors

- Render mirror and store the stencil buffer
- Render reflections and consult the stencil buffer
  - For each fragment, if corresponding pixel in buffer is 1, then accept, else reject
- Notes:
  - We need to modify our fragment shaders
  - Different than my conceptual slide – not a list of pixels that are covered, but rather a 0/1 for every pixel

Oh. There Is a Simpler Way When Walls Are Involved.

- What if the mirror is on a wall?



Z-buffer algorithm will put wall in the image, and reflection is removed since it is behind the wall

Project 3D

- This is a double project
  - Worth two projects to do it
- Extend 2B
- Place dog in room
- Room has 4 mirrors
- Do reflections
  - just one level – don't need reflections of reflections
- Modify shaders to get normals right
- No starter code

Pizza

- Looking promising, but I need to get final approval from CIS
- Hoping for an announcement soon