CSC 240: Computer Graphics

Final Quiz: Fall 2015

Monday, December 14, 2:40-4pm

- This is an in-class final quiz on the material from the second half of the semester.
- It is closed notes, closed Internet, and closed technology, but you may use a "cheat sheet".
- Your cheat sheet must be hand-written, created by you, 8.5 × 11 inches, and double-sided (at most).
- If you have a question, either come to the front, or raise your hand and I'll come over.
- If there is a clarification to be made, I will write it on the board.
- I will be in and out of the room since there are also exams in other rooms.
- Do not discuss the exam with other students and respect the honor code of doing your own work.
- If you are unable to make progress on any part of the exam, tell me what you tried; describe your thought process.
- Even if you are not finished, your exam must be turned in at the end of class to maintain fairness.

Name	Solution Set (sketches)

✓	Part 1	/25
✓	Part 2	/20
✓	Part 3	/20
✓	Part 4	/10
✓	Part 5	/25
	Total	/100

Part 1: Projection

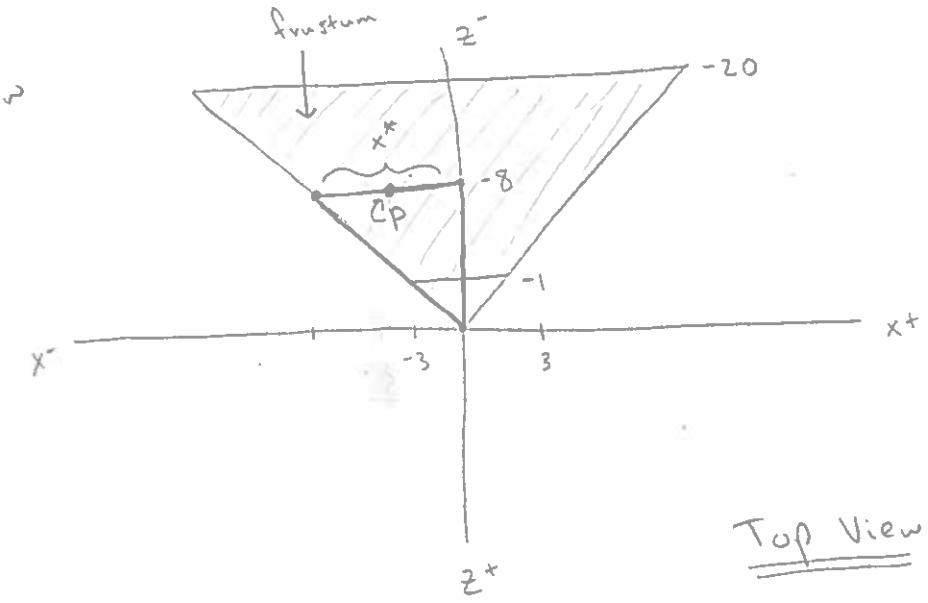
(15) (a) Given a camera at the origin and the frustum below:

```
glFrustum(-3, 3, -5, 5, 1, 20),
```

does the point $\vec{P} = (-12, 0, -8)$ lie inside the viewing frustum or does it get clipped out?

(10) i. What viewpoint(s) would be helpful for answering this question? Draw out one such viewpoint, labeling the positive/negative axes and shading the area inside the frustum.

either the top or the bottom view



not to scale

Top View

(5) ii. Use your diagram to determine whether $\vec{P}$ is inside the viewing frustum or not. Show clearly how you arrived at your answer and label $\vec{P}$ (approximately) on your diagram.

$$\frac{x^*}{-8} = \frac{-3}{-1} \Rightarrow \boxed{x^* = -24}$$

Since $P_x = -12 > -24 = x^*$, $\vec{P}$ is inside the viewing frustum

(easier to see that $P_y = 0$ and $P_z = -8$ place $\vec{P}$ inside the frustum)

(b) You are given the following 8 vertices of a cube in world space, a camera at the origin, and a viewport at $z = -1$.

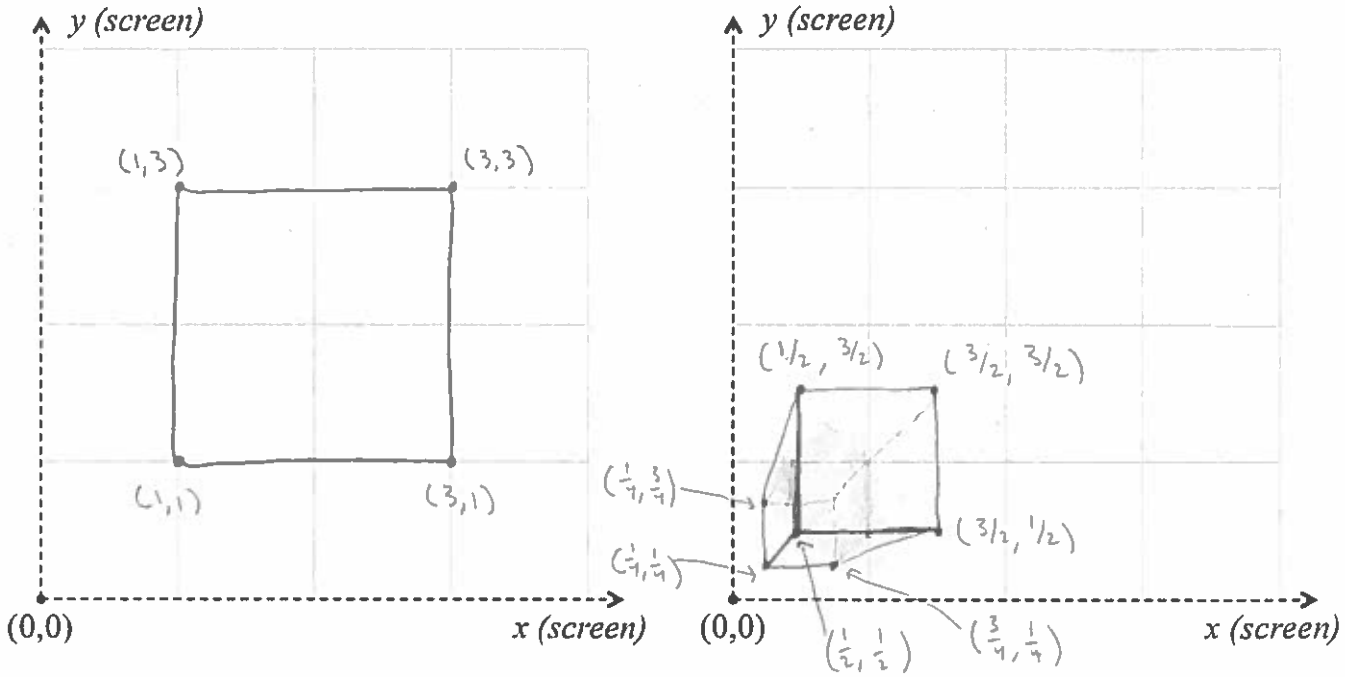
world coordinates	orthographic projection	perspective projection
(1,1,-2)	(1,1)	(1/2, 1/2)
(3,1,-2)	(3,1)	(3/2, 1/2)
(3,3,-2)	(3,3)	(3/2, 3/2)
(1,3,-2)	(1,3)	(1/2, 3/2)
(1,1,-4)	(1,1)	(1/4, 1/4)
(3,1,-4)	(3,1)	(3/4, 1/4)
(3,3,-4)	(3,3)	(3/4, 3/4)
(1,3,-4)	(1,3)	(1/4, 3/4)

(5) i. Fill in the table above with the 2D screen coordinates for each type of projection. Imagine that the viewport is unlimited in the x and y directions, so clipping is not an issue.

orthographic: just drop the z-coordinate

perspective: divide by the absolute value of the z-coordinate

(5) ii. Draw what the "viewer" would see in each case (only a portion of the viewport is shown).



Orthographic

very different!

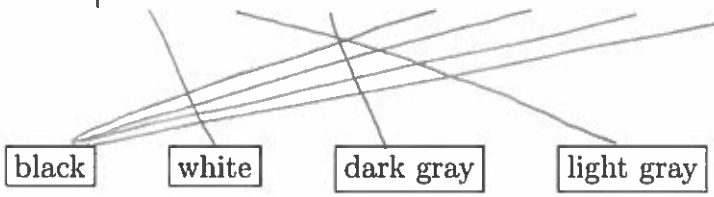
Perspective

Part 2: Lighting

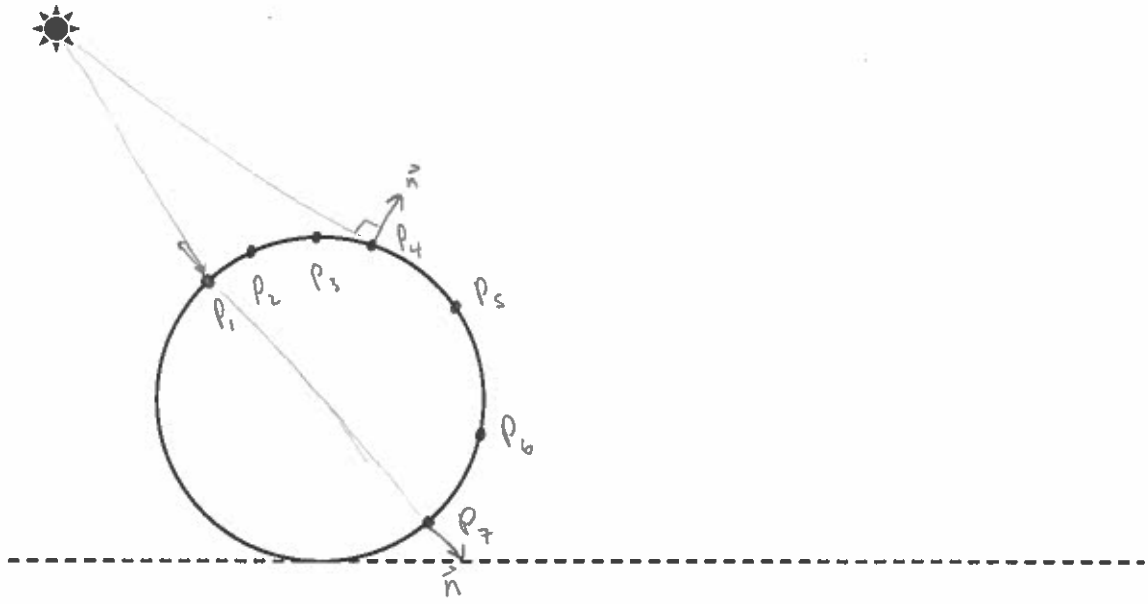
You are given 7 different points on the surface of a sphere. First you compute the unit normal vector at each point, then you compute the dot product of the normal vector with the unit light vector at each point.

(a) Draw a line from each dot product to the corresponding shading color. Justify your answers for partial credit.

point on the sphere	$\vec{P}_1$	$\vec{P}_2$	$\vec{P}_3$	$\vec{P}_4$	$\vec{P}_5$	$\vec{P}_6$	$\vec{P}_7$
dot product	-1	-0.8	-0.2	0	0.2	0.8	1



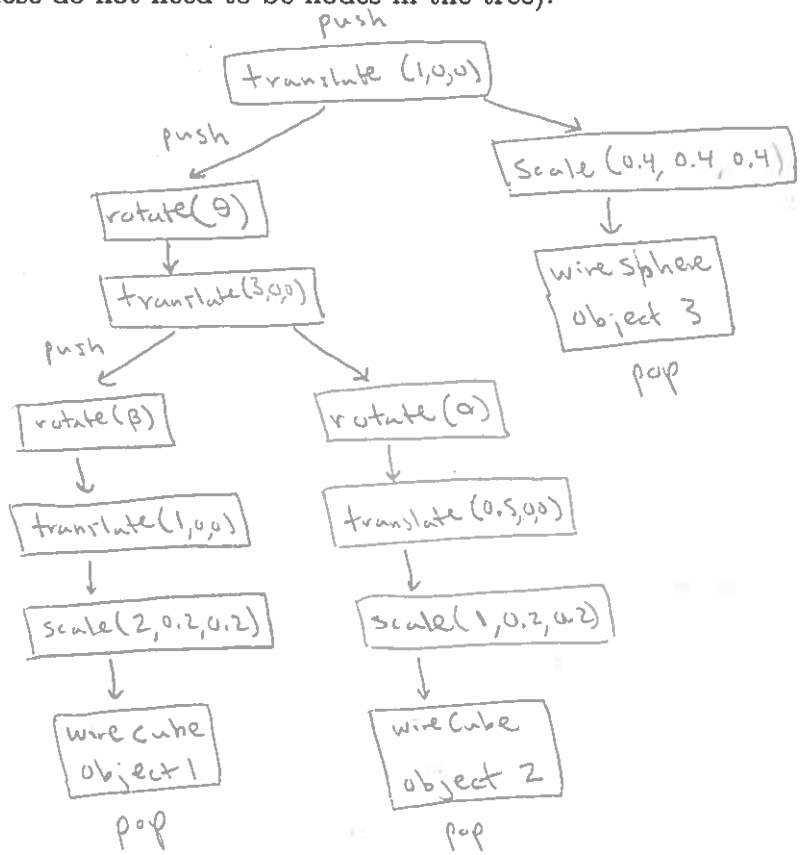
(b) In the picture (side view) of a sphere and a light source below, label 5 points that could be $\vec{P}_1, \cdots, \vec{P}_5$ (i.e. those points would give roughly the dot products shown above).



Part 3: Hierarchical Models

(a) The following OpenGL code represents a hierarchical model animation. From this code, draw the corresponding tree, using one node for each transformation and each object (also denote the placement of push and pop, but these do not need to be nodes in the tree).

```
- glPushMatrix()
  glTranslate(1,0,0)
- glPushMatrix()
  glRotate(theta, 0, 0, 1)
  glTranslate(3, 0, 0)
- glPushMatrix()
  glRotate(beta,0,0,1)
  glTranslate(1,0,0)
  glScale(2,0.2,0.2)
  glutWireCube(1) # object 1
- glPopMatrix()
  glRotate(alpha,0,0,1)
  glTranslate(0.5,0,0)
  glScale(1,0.2,0.2)
  glutWireCube(1) # object 2
- glPopMatrix()
  glScale(0.4,0.4,0.4)
  glutWireSphere(1,20,20) # object 3
- glPopMatrix()
```



(b) For each of the three objects, list the transformations that are applied to it, in order. Indicate which transformation is applied first and which transformation is applied last.

object 1

first → ① scale(2, 0.2, 0.2)
② translate(1, 0, 0)
③ rotate(β)
④ translate(3, 0, 0)
last → ⑤ rotate(θ)
⑥ translate(1, 0, 0)

object 2

first → ① scale(1, 0.2, 0.2)
② translate(0.5, 0, 0)
③ rotate(α)
④ translate(3, 0, 0)
⑤ rotate(θ)
last → ⑥ translate(1, 0, 0)

object 3

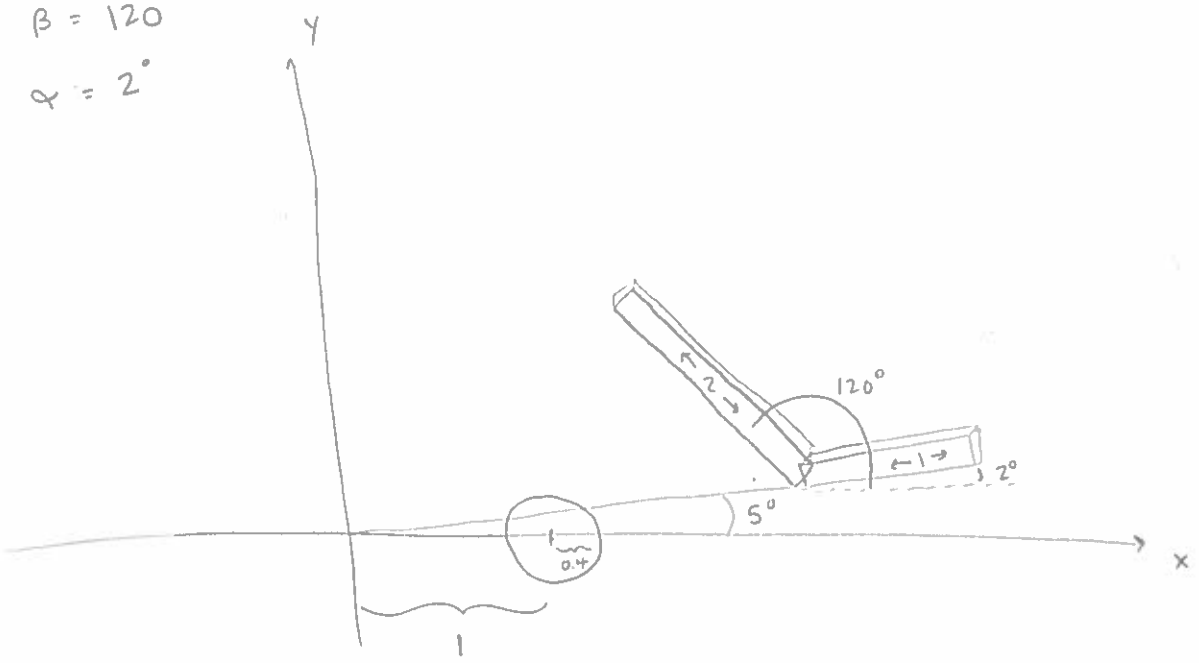
first → ① scale(0.4, 0.4, 0.4)
last → ② translate(1, 0, 0)

(c) For each call of the display method, angle theta (θ) is incremented by 0.5 degrees, angle beta (β) is incremented by 12 degrees, and angle (α) is incremented by 0.2 degrees. Draw a picture of what this system looks like after 10 display calls. Be as accurate as possible with the values of each transformation.

after 10 display calls $\theta = 5^\circ$

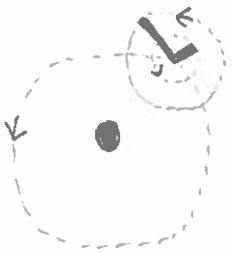
$\beta = 120^\circ$

$\alpha = 2^\circ$



(d) Describe the general motion of the system in words.

- The sphere does not move
- the two scaled boxes act like the hands of a clock (the ratio of their angles is 60 to 1)
- The clock itself is rotating in a circle around the sphere, and the whole system is translated one unit to the right.

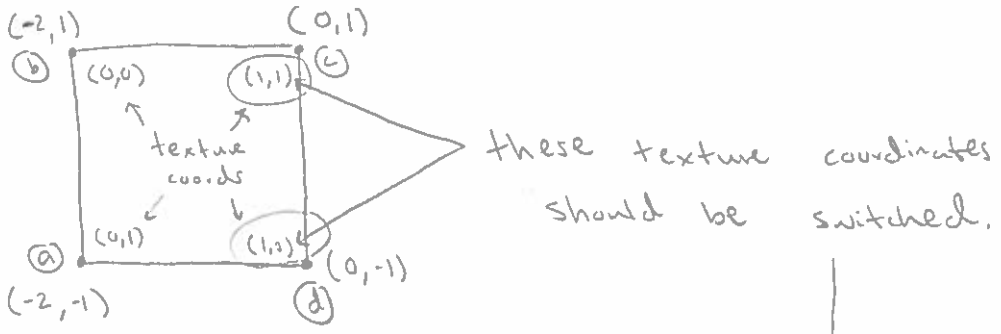


Part 4: Texture Mapping

The goal of the following OpenGL code is to texture map a square (with the texture image correctly oriented). Assume the *upper left* corner of the texture is $u = 0$ and $v = 0$.

```
glBegin(GL_QUADS)
glTexCoord2f(0, 1)
(a) glVertex(-2, -1, 0)
glTexCoord2f(0, 0)
(b) glVertex(-2, 1, 0)
glTexCoord2f(1, 1)
(c) glVertex(0, 1, 0)
glTexCoord2f(1, 0)
(d) glVertex(0, -1, 0)
glEnd()
```

(a) Draw a picture of these vertices and how they match up to the texture coordinates.



(b) Explain the error in this code and how to fix it.

Part 5: Ray-tracing

You are trying to figure out whether a circular mirror on the wall of your scene is visible from a certain pixel on the screen. The direction of the ray and the pixel on the screen are, respectively:

$$\vec{R}_d = \left(\frac{3}{5}, 0, -\frac{4}{5}\right), \quad \text{and} \quad \vec{S} = (-1, 2, -1).$$

(a) The camera is 5 units away from pixel $\vec{S}$. Where is the camera located (x, y, z coordinates)?

let $R_o = (c_x, c_y, c_z)$

$R(t) = R_o + tR_d$

$$\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} + 5 \cdot \begin{bmatrix} 3/5 \\ 0 \\ -4/5 \end{bmatrix}$$

$\Rightarrow$ $c_x = -1 - 3 = -4$

$c_y = 2 - 0 = 2$

$c_z = -1 + 4 = 3$

$\Rightarrow$ $\boxed{R_o = (-4, 2, 3)}$ (camera location)

(b) The circular mirror is located on the wall represented by the plane $x = 5$. What are the coordinates of $\vec{P}$, the point where this ray intersects the wall? How far away is $\vec{P}$ from the camera?

$$R(t) = \begin{bmatrix} -4 \\ 2 \\ -3 \end{bmatrix} + t \begin{bmatrix} 3/5 \\ 0 \\ -4/5 \end{bmatrix}$$

let $P = (p_x, p_y, p_z)$

know: $p_x = 5$

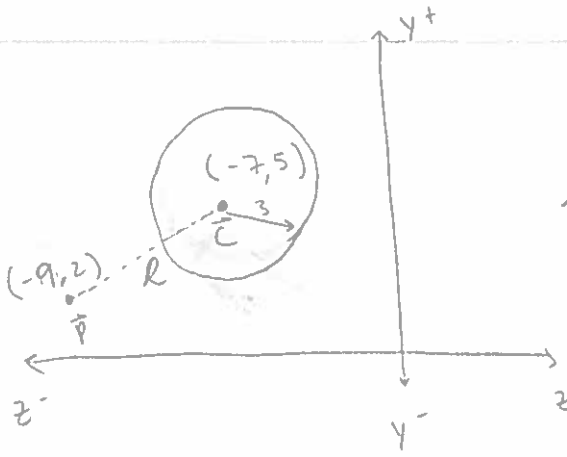
x-coord $\Rightarrow$ $p_x = 5 = -4 + t \cdot 3/5$ $\Rightarrow$ $t \cdot \frac{3}{5} = 9$ $\Rightarrow$ $\boxed{t = 15}$

y-coord $p_y = 2 + t \cdot 0$ $\Rightarrow$ $\boxed{p_y = 2}$

z-coord $p_z = 3 + 15\left(-\frac{4}{5}\right)$ $\Rightarrow$ $p_z = 3 - 12$ $\Rightarrow$ $\boxed{p_z = -9}$

$\Rightarrow$ $\boxed{\vec{P} = (5, 2, -9)}$

(c) The circular mirror has center point $\vec{C} = (5, 5, -7)$ and radius $r = 3$. Does this ray intersect the mirror? Justify your answer (it might be helpful to draw a picture of the wall).



$$l = \sqrt{(-9-(-7))^2 + (2-5)^2} = \|\vec{P} - \vec{C}\|$$

$$l = \sqrt{4+9}$$

$$l = \sqrt{13} > 3 = r$$

left view

no, not inside the circle

(d) Create a general algorithm for determining whether a ray intersects a circular object lying on a given plane. You don't need to use code or pseudocode, just a general description of how to find the equivalent of the point $\vec{P}$, and then an inequality in terms of $\vec{P}, \vec{C}$, and r.

First, find the intersection point $\vec{P}$ of the ray and the plane the circle lies on. Then, to test whether that point $\vec{P}$ is inside the circle, check:

$$\|\vec{P} - \vec{C}\| \leq r$$

if yes, ray does intersect circle

if no, ray does not intersect circle.