Name: ____

Section 1: Conceptual Questions

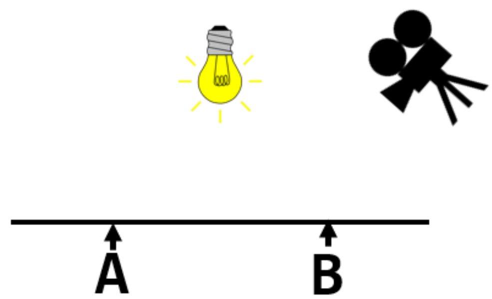
Problem 1 [2 points]: Consider the depth map used for shadows as described in class. Explain, in your own words, why calculating the Z component of each position is sufficient for describing the depth of a pixel relative to the viewer. Then, summarize how we can use depth to work out shadow locations for each fragment in our scene:

We loosely defined depth to be "further into the clip space box", and is thus proportional to distance. Since our z component gives us distance in a perspective projection (and arguably in an orthographic projection), we can directly just use this number. Given a depth of a fragment relative the camera, we can determine if there is *something* with smaller depth, which in turn tells us for each fragment if it's in shadow or in light.

Problem 2 [2 points]: Explain the difference between a **perspective** and **orthographic** camera. You may find it helpful to include diagrams:

A perspective camera scales each object proportional to the distance from the camera (where size ~ 1/z). An orthographic camera, in contrast, only cuts off objects if they fall outside of the viewing plane, but does not scale more-distant objects.

Problem 3 [4 points]: Consider the following image:



For each point **A** and **B** in this image, identify whether or not the following BRDF components of light would (likely) be visible to the camera:

- Ambient
- Diffuse
- Specular

Briefly explain your reasoning

At point A, we would likely observe Ambient and Diffuse components of light, but not specular. Ambient light is visible to any fragment visible to the camera, and Diffuse light is visible (at least somewhat) to any surface visible to the camera and facing vaguely towards the light. The specular component, however, is likely not visible, since the bounce of the specular will shoot to the left rather than towards the camera.

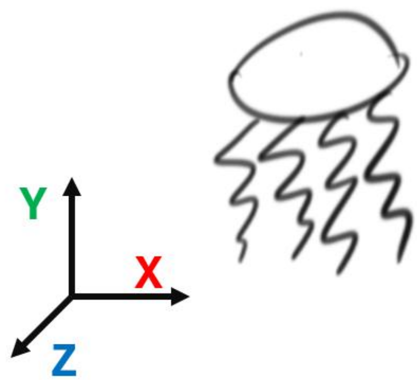
At point B, we would likely observe Ambient and Diffuse components of the light, for reasons identical to point A. We would, however, likely also observe some amount of the specular light component, since the specular "bounce" seems to line up pretty exactly with the camera position.

Name: ____

Section 2: Rotations

Meta Note: I'm only including two (harder) rotation problems in this practice to give a sense of the format.

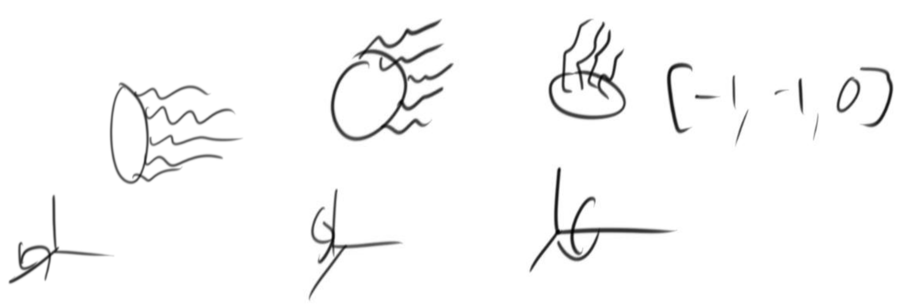
Problem 4 [3 points]: Consider the following Jellyfish, oriented to face in the [0, 1, 0] direction:



Now suppose we rotate this jellyfish using Euler angles in the order Yaw-Pitch-Roll:

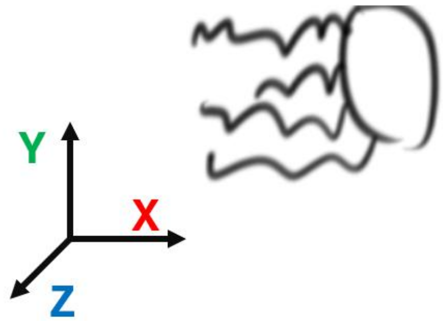
90°, 45°, 90°

Draw the resulting jellyfish and write down its orientation in 3D space. See the "Jellyfish orientation reference" for images to help with your sketch. You can show your steps for partial credit, so long as you circle your final result:



Name: ____

Problem 5 [3 points]: Consider the following Jellyfish, oriented to face in the [1, 0, 0] direction:



Now suppose we rotate this jellyfish using the following (normalized) quaternion:

`[1/2, 0, 1/2, -√2/2]`

Draw the resulting jellyfish and write down its orientation in 3D space. See the "Jellyfish orientation reference" for images to help with your sketch. Including a diagram/description of the quaternion rotation you're applying may result in partial credit:

$$2 * acos\left(\frac{-\sqrt{2}}{2}\right) = 2 * -45 = -90°$$

around $\left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right)$

$\left[\frac{1}{2}, \frac{1}{2}, -\sqrt{2}\right]$

Name: ____

Section 3: GLSL Shaders

For the following two problems, consider this GLSL fragment shader `redshift.frag`:

```
uniform float u_RedShift;
varying vec3 v_Color;
int main() {
  vec3 shifted = vec3(v_Color.r * u_RedShift, v_Color.gb);
  gl_FragColor = vec4(shifted, 1.0);
}
```

Problem 6 [2 points]: For a given fragment, we are given `v_Color=[0.5, 0.2, 0.8]` and `u_RedShift=0.5`. For this fragment, what color do we expect the shader to produce? Give your answer as a `vec4`, and show your work for partial credit:

shifted := [0.25, 0.2, 0.8]

gl_FragColor := [0.25, 0.2, 0.8, 1.0]

Problem 7 [3 points]: Suppose we use this shader on a grey teapot (so every vertex has the color `[0.5, 0.5, 0.5]`). If we vary `u_RedShift` between 0 and 2 for this setup, what do we expect to happen to the teapot? For full credit, concretely describe how the teapot would appear with values of `u_RedShift<1` and `u_RedShift>1`:

This shader just multiplies the red component of each fragment by our redshift value. When u_RedShift is 2, for example, our entire teapot would have the color [1, .5, .5], which would be much more visibly red. If u_RedShift is less than 1, however, our teapot would become “less red” (as that component is reduced), making the teapot appear more greenish-blue (aka teal).

Name: ____

Section 4: Textures

For this section, we define the following colors:

```
R = (255, 0, 0)
G = (0, 255, 0)
B = (0, 0, 255)
W = (255, 255, 255)
```

For the following two problems, consider the following 5x5 texture, defined using the GLSL (u, v) coordinate system given in class (where (0, 0) is in the lower-left of the texture):

.9	R	R	G	G	R
.7	B	G	G	G	P
.5	P	P	P	P	R
.3	B	B	B	B	W
.1	B	B	B	W	W
	.1	.3	.5	.7	.9

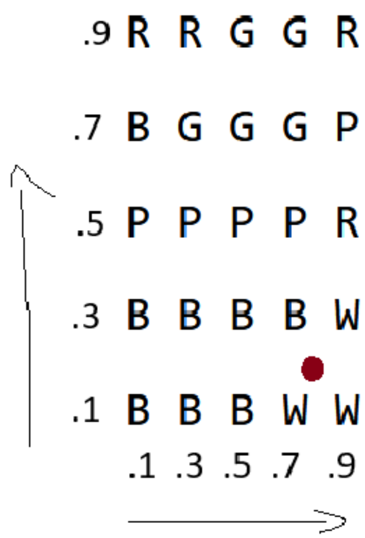
Problem 8 [2 points]: Suppose this texture is *magnified* using the gl.NEAREST flag. What color do you expect from sampling at the following (u, v) points?

```
(0.25, 0.3)
```

.9	R	R	G	G	R
.7	B	G	G	G	P
.5	P	P	P	P	R
.3	B	B	B	B	W
.1	B	B	B	W	W
	.1	.3	.5	.7	.9

→ B = [0, 0, 255]

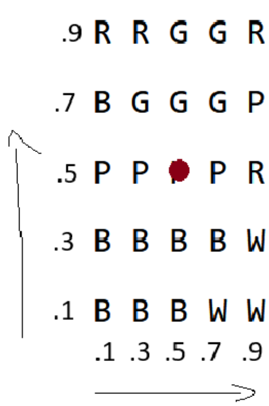
(0.74, 0.195)



→ W = [255, 255, 255]

Problem 9 [2 points]: Suppose the texture is *magnified* using the gl.LINEAR flag. What color (written as a vec3) do you expect from sampling at the following (u, v) points?

(0.5, 0.5)



→ P = [200, 0, 200]

Note that only one point contributes in this case

```
(0.7, 0.2)
```

.9	R	R	G	G	R
.7	B	G	G	G	P
.5	P	P	P	P	R
.3	B	B	B	B	W
.1	B	B	B	W	W
	.1	.3	.5	.7	.9

→ color = (W * (.3 - .2) + B * (.2 - .1)) / (.3 - .1) ~ ([25, 25, 25] + [0, 0, 25]) / .2 ~ [25, 25, 50] / .2 ~ [125, 125, 250]