CS 130, Midterm

Solutions

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Read the entire exam before beginning. **Manage your time carefully.** This exam has 40 points; you need 32 to get full credit. Additional points are extra credit.

Short answer. For each question below, provide a brief 1-3 sentence explanation.

Problem 1 (2 points)

Plants absorb a portion of the sunlight that they receive and use it to produce sugars in a process called photosynthesis. What colors of light do you think the plants are using?

Plants appear green because they reflect the green light that they receive. They absorb the colors that are not green. They use red/orange light and blue/violet light for photosynthesis. They do not use green or yellow.

Problem 2 (2 points)

What is self-shadowing, and how do we prevent it?

Self-shadowing is when we cast a shadow ray to see if any object blocks the light and discover that the light is blocked by an intersection with ourself at distance zero. We prevent it by discarding intersections that are very close to zero.

Problem 3 (2 points)

Which of these features lead to recursion in a ray tracer? No explanation is required, but no partial credit is possible. (1) Texture mapping, (2) transparency shader, (3) Phong shader, (4) antialiasing, (5) reflective shader, (6) bump mapping, (7) area lights.

Only reflective and transparency shaders lead to recursion, since they recursively shade reflected (and possibly also transmitted) rays. Note that shadow rays are not recursive, so there is no recursion associated with lights or Phong shaders. Antialiasing and area lights

involve casting more rays, but not recursively. Bump mapping and texture mapping just add image lookups to the shading calculation.

Problem 4 (2 points)

When looking out a window during the day, we see whatever is outside. When we look out a window at night, we see ourselves. Why?

Transparent objects both reflect and transmit light. During the day, the light from the bright outdoors dominates, and we mostly just see what is outside. At night, there is little light coming from outside, so the reflected light dominates.

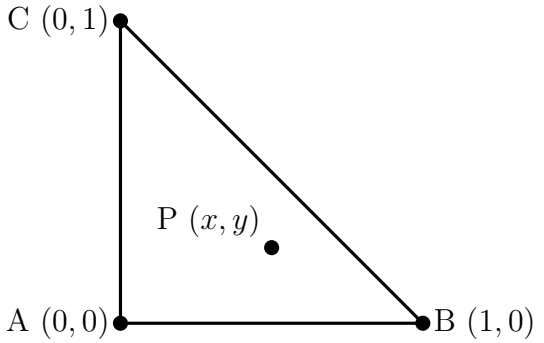
Problem 5 (2 points)

Construct a normalized vector that is orthogonal to the 2D vector $\langle x, y \rangle$.

$\begin{pmatrix} -\frac{y}{\sqrt{x^2+y^2}} \\ \frac{x}{\sqrt{x^2+y^2}} \end{pmatrix}$ or its negation.

Problem 6 (4 points)

Find the barycentric weights for the point P in the triangle below.



To do this, we need to compute the areas of the triangles, which we can then use to compute barycentric weights. Note that all but one of the triangles we need have an edge that is horizontal or vertical, so $A = \frac{1}{2}bh$ is easy to compute.

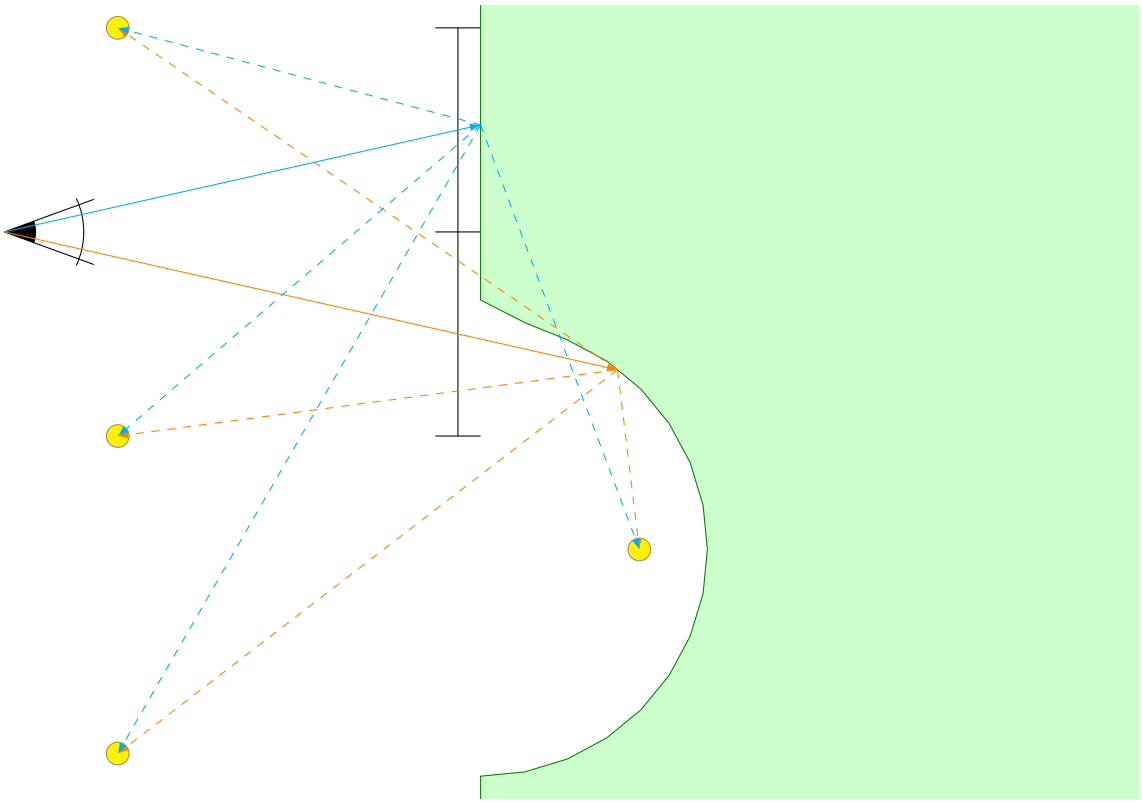
$$\text{area}(ABC) = \frac{1}{2} \qquad \text{area}(APC) = \frac{x}{2} \qquad \text{area}(ABP) = \frac{y}{2}$$

$$\text{area}(PBC) = \text{area}(ABC) - \text{area}(APC) - \text{area}(ABP) = \frac{1-x-y}{2}$$

$$\alpha = \frac{\text{area}(PBC)}{\text{area}(ABC)} = 1 - x - y \qquad \beta = \frac{\text{area}(APC)}{\text{area}(ABC)} = x \qquad \gamma = \frac{\text{area}(ABP)}{\text{area}(ABC)} = y$$

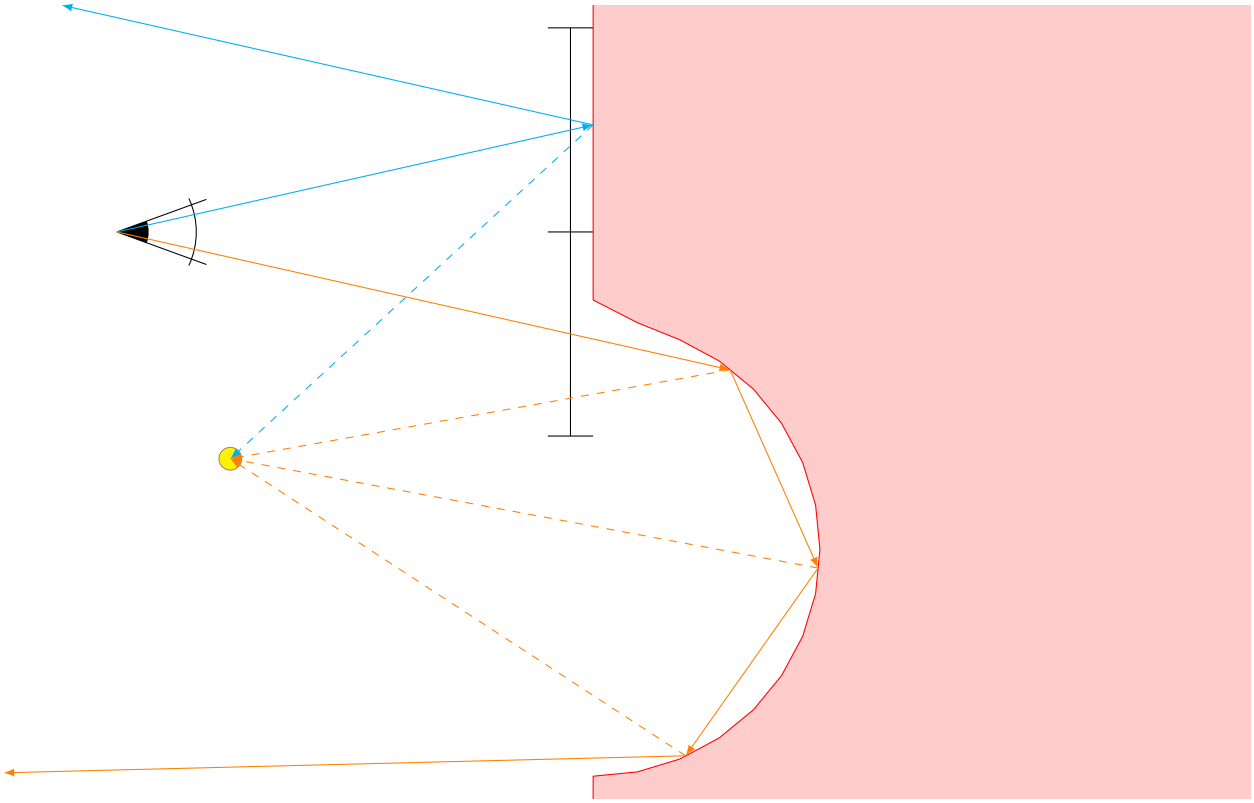
Problem 7 (4 points)

Below is a simple 2D raytracing setup. The 1D image has two pixels. The green object is made of wood. The yellow circles are point lights. Draw all of the rays that would be cast while raytracing this scene. (Tip: use the edge of a piece of paper or a pencil to draw rays; it helps.)



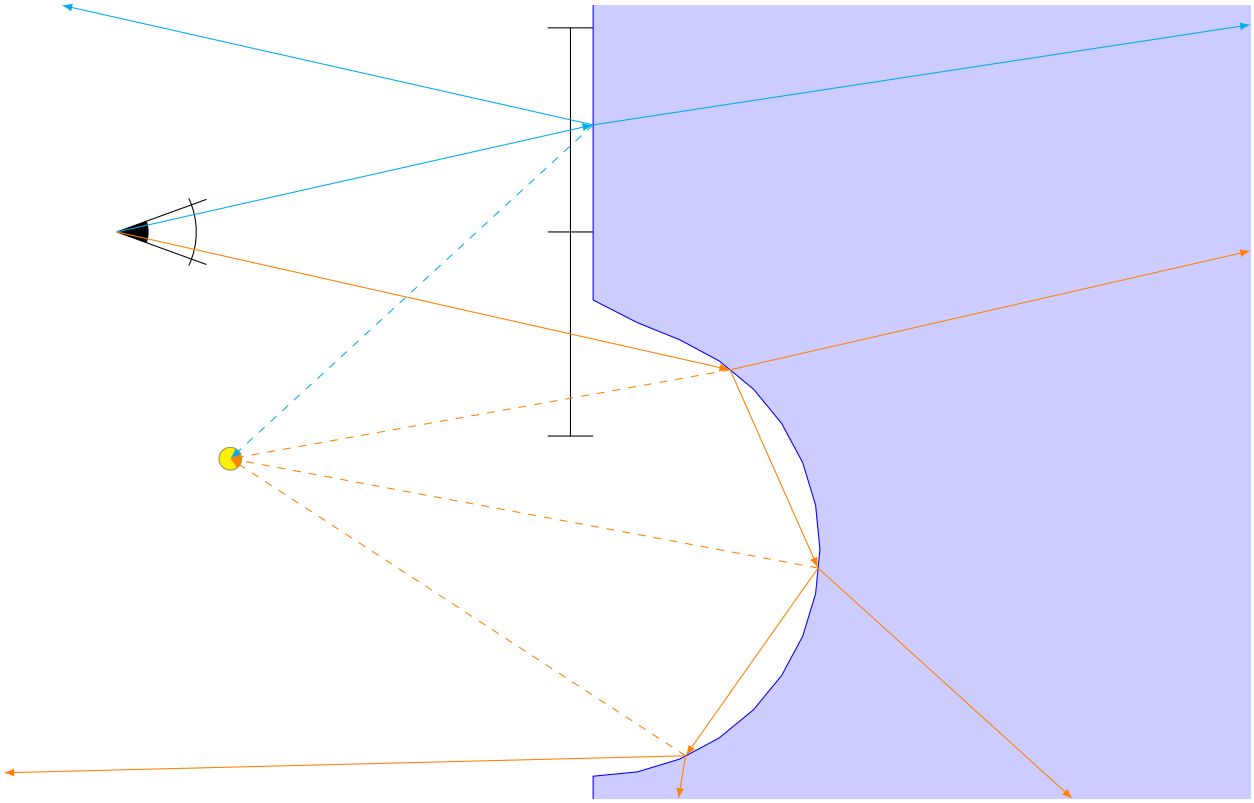
Problem 8 (4 points)

Below is a simple 2D raytracing setup. The 1D image has two pixels. The red object is reflective. The yellow circle is a point light. Draw all of the rays that would be cast while raytracing this scene. (Tip: use the edge of a piece of paper or a pencil to draw rays; it helps.)



Problem 9 (4 points)

Below is a simple 2D raytracing setup. The 1D image has two pixels. The blue object is made from glass. The yellow circle is a point light. Draw all of the rays that would be cast while raytracing this scene. (Tip: use the edge of a piece of paper or a pencil to draw rays; it helps.)



Problem 10 (2 points)

Imagine that we are ray tracing a simple scene with a camera at $(-1, 2, 0)$. We are rendering a pixel at location $(0, 1, 0)$. What are the endpoint and direction for the ray that we will cast?

The ray's endpoint is $(-1, 2, 0)$. The (un-normalized) direction is $\begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$, which normalizes to the ray direction $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$.

Problem 11 (2 points)

The scene contains only one object, a plane passing through the origin with normal $\langle 0, 1, 0 \rangle$. Find the ray-plane intersection location.

The intersection location is $(1, 0, 0)$. (At distance $t = 2\sqrt{2}$ along the ray.)

Problem 12 (2 points)

What is the surface normal at the intersection location?

It is just the plane's normal: $\langle 0, 1, 0 \rangle$.

Problem 13 (2 points)

There is a point light located at $(3, 3, 1)$. Compute the endpoint and direction for the shadow ray that must be cast.

The endpoint is the intersection location $(1, 0, 0)$. The direction is the normalization of $\begin{pmatrix} 3-1 \\ 3-0 \\ 1-0 \end{pmatrix}$, which is $\frac{1}{\sqrt{14}} \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$.

Problem 14 (2 points)

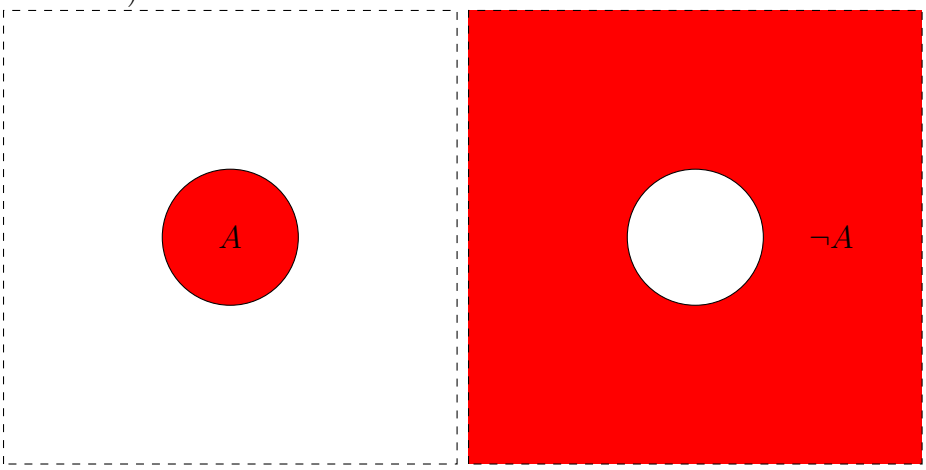
The plane is reflective, so we must cast a reflection ray. Compute the endpoint and direction for the reflection ray that must be cast.

The endpoint is the intersection location $(1, 0, 0)$. The direction is the reflected direction

$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$.

Problem 15 (4 points)

In class, we discussed an algorithm for ray tracing intersections, unions, and differences. Given intersection lists for two objects A and B, we are able to construct an intersection list for the desired boolean operation. Construct an algorithm to compute the intersection list corresponding to the *complement* of A (denoted $\neg A$). (That is, the set of points that are not in A.)



If the intersection list begins with a 0, remove the zero. Otherwise, append a 0 to the front. (Note that I am taking the convention that any terminal ∞ is omitted.)