# CS130 - Colors

Name: Anand Mahadevan SID: 862132182

**1.** For each of the following RGB triples, identify the corresponding color.

(a) $(0, 0, 0)$ Black

(b) $(1, 0, 0)$ Red

(c) $(0, 1, 0)$ Green

(d) $(1, 1, 0)$ Yellow

(e) $(0, 0, 1)$ Blue

(f) $(1, 0, 1)$ Purple

(g) $(0, 1, 1)$ Teal

(h) $(1, 1, 1)$ White

**2.** For each of the following colors, give an approximate RGB value for the color, where each of R,G,B is 0, $\frac{1}{2}$, or 1. You must justify your answer.

(a) gray (0.5, 0.5, 0.5) because half white half black

(b) pink (1, 0.5, 0.5) because red + white from at least 0.5 RGB

(c) navy blue (0, 0, 0.5) because closer to 0 blue

(d) orange (1, 0.5, 0)because red greenish

(e) forest green (0, 0.5, 0) because closer to 0 green

(f) dark purple (0.5, 0, 0.5) because red + blue = purple, and closer to 0 = darker

(g) light green (0.5, 1, 0.5) because green + white from at least 0.5 RGB

**3.** Imagine that you have a fourth type of color sensitive cell in your eye that nobody else has.

(a) What would this allow you to do that others cannot?

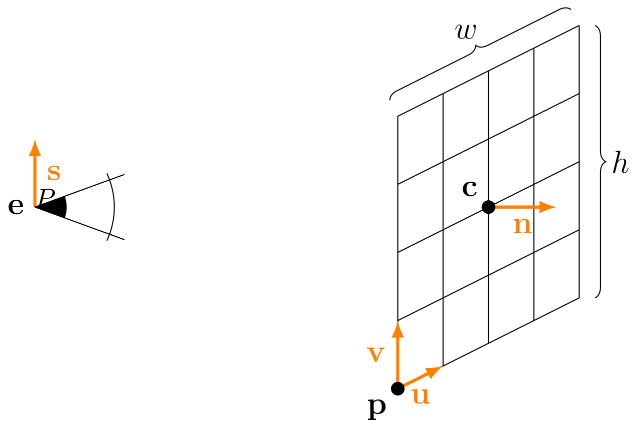
(b) Devise an experiment that would allow you to convince others that you actually have this ability?

(a) I can now see color combinations with the new color combined with the existing RGB colors, and of course the new color by itself.

(b) I would construct a straight set of 100 pair of doors that I can't see past, each pair labeled "Left" and "Right". Each pair would have 1 of the 2 doors lead to death, and the other to the next pair of doors, until the exit after the 100th pair of doors. I would paint the special color that only I can see on each door that does not lead to death. I would simply walk through the 100 "safe" doors to the exit, which would have a 1 / $2^{100}$ probability of occurring under random choice of decision making. So the only logical conclusion that would be made would be that there is information only present to me regarding door choices, which I have proven to be the paint on the correct doors.

## Ray tracing

**4.** In the figure, you are given the position of the camera (**e**), the location of the center of the image plane (**c**), an up vector (**s**), and the width $w$ of the image plane (in 3D space). You are also given the number of pixels in height and width $(H, W)$ of the image.



Compute the height $h$ of the image plane (in 3D space). You may assume that the pixels are squares.

$h = w * (H/W)$

**5.** Determine the height $h_p$ and width $w_p$ of a pixel.

$h_p = h/H$
$w_p = w/W$

**6.** The ray from **e** to **c** is assumed to be orthogonal to the image plane. Find the plane's normal **n**.

n = (c-e) / $\|c - e\|$

**7.** The up vector **s** we are given need not be orthogonal to **n**, so we cannot use it as our image's up vector **v**. Instead, we must first compute $\hat{\mathbf{u}}$ orthogonal to **n** and **s**. Then, we can compute $\hat{\mathbf{v}}$ from **n** and $\hat{\mathbf{u}}$. Both $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ should be normalized. If we are standing at **e** and looking at **c** with **s** pointing roughly upwards, then $\hat{\mathbf{u}}$ should be pointing to the right, and $\hat{\mathbf{v}}$ should be pointing upwards. Make sure you check that these two vectors point in the correct direction. Don't worry about their length yet.

$\hat{\mathbf{u}}$ = (n x s) / ||n x s||
$\hat{\mathbf{v}}$ = (n x $\hat{\mathbf{u}}$) / ||n x $\hat{\mathbf{u}}$||

**8.** Next, we scale $\hat{\mathbf{u}}$ to **u** and $\hat{\mathbf{v}}$ to **v** so that $\|\mathbf{u}\| = w_p$ and $\|\mathbf{v}\| = h_p$. In this way, adding **u** causes us to move right one pixel, and adding **v** causes us to move up one pixel.

u = $\hat{\mathbf{u}}$ * $w_p$
v = $\hat{\mathbf{v}}$ * $h_p$

**9.** Find the bottom left corner **p**.

p = c - (u * w / 2) - (v * h / 2)

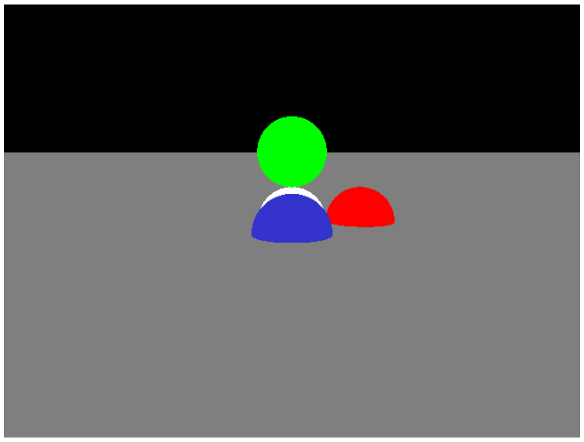
**10.** Given a pixel index $(i, j)$, find the corresponding point **x** on the image plane. Note that pixel index $(i, j)$ refers to the *center* of pixel $(i, j)$, with pixel $(0, 0)$ being the bottom left pixel and $(W - 1, H - 1)$ being the top right pixel. Note that **p** is at the corner of the image, not the center of the lower left pixel.

x = p + u * (i+0.5) + v * (j+0.5)

**11.** Find the endpoint and (normalized) direction of the ray that should be cast through pixel $(i, j)$.

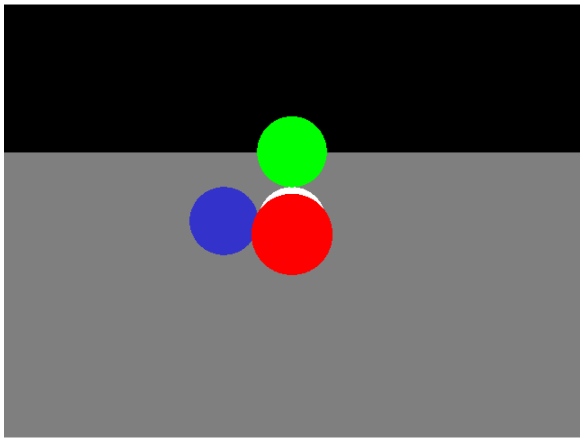
endpoint = e
direction (normalized) = (x - e) / $\|x - e\|$

**12.** Modify test case 04.txt so that it produces the following result:



Change Plane Point from (0, -1, 0) to (0, 0, 0)
OLD: plane Pg 0 -1 0 0 1 0
NEW: plane Pg 0 0 0 0 1 0

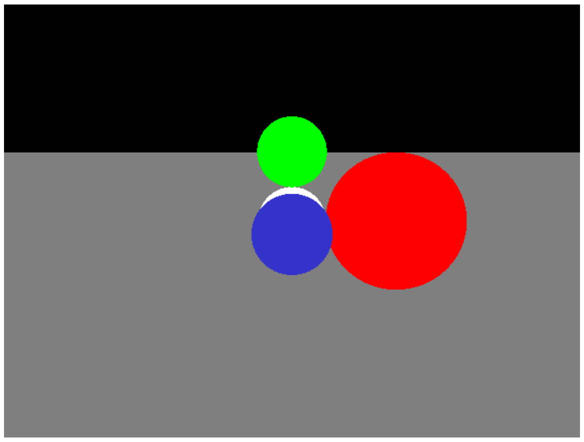
**13.** Modify test case 04.txt so that it produces the following result:



Change Red Sphere Point from (1, 0, 0) to (0, 0, 1)
OLD: sphere Sr 1 0 0 .5
NEW: sphere Sr 0 0 1 .5

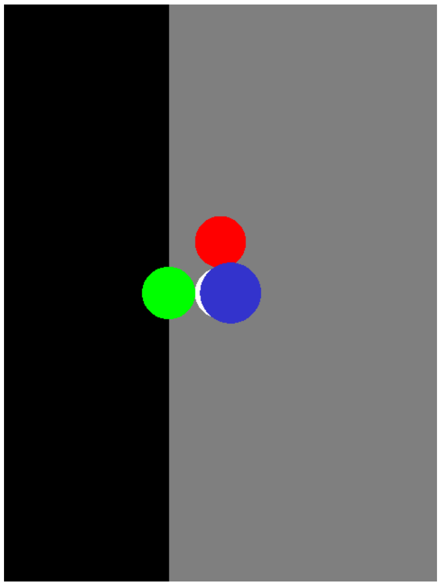
Change Blue Sphere Point from (0, 0, 1) to (-1, 0, 0)
OLD: sphere Sb 0 0 1 .5
NEW: sphere Sb -1 0 0 .5

**14.** Modify test case 04.txt so that it produces the following result:



Change Red Sphere Point from (1, 0, 0) to (1.5, 0, 0)
Change Red Sphere Radius from 0.5 to 1
OLD: sphere Sr 1 0 0 .5
NEW: sphere Sr 1.5 0 0 1

**15.** Modify test case 04.txt so that it produces the following result:



Change image pixel dimensions from 640 x 480 to 480 x 640
OLD: size 640 480
NEW: size 480 640
Change Camera pseudo-up-vector from (0, 1, 0) to (1, 0, 0)
OLD: camera 0 1 6 0 0 0 0 1 0 70
NEW: camera 0 1 6 0 0 0 1 0 0 70