Stanford Honor Code Statement:

The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive impermissible aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code;

The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.

Signature: _____

Name (Print): _____

Date: _____

CS148 Midterm, Summer 2010

You have 2 hours to complete this exam
The exam is closed-everything, including calculators

The question paper has 10 pages & 6 questions (including extra credit). Avoid attaching extra sheets as far as possible. You may use blank scratch sheets, but don't submit them. You don't need to show steps in a solution unless they're explicitly asked for.

NAME: _____

1. Light and Color *(8 points)*

1a) Andrew is playing with cyan, magenta and yellow paint. Assuming the paints mix perfectly (no chemical reactions), what colors would he observe when he mixes the following? *(4 points)*

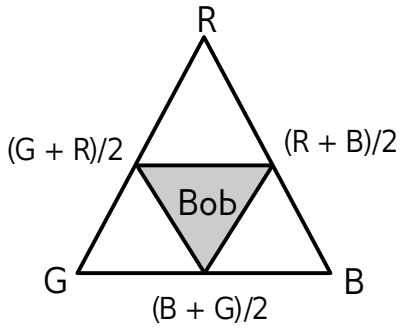
Cyan + Magenta = *Blue*

Cyan + Yellow = *Green*

Yellow + Magenta = *Red*

Cyan + Magenta + Yellow = *Black*

1b) Bob decides to stand up to the CIE and choose his own set of primaries in an additive color space (mixing beams of light). Bob's primaries are (R + B) / 2, (B + G) / 2 and (G + R) / 2. In other words, he constructs each of his primaries by taking two CIE primary beams at half the reference intensities, and mixing them together. Is the gamut of his color space smaller than, larger than, or equal to CIE RGB? Briefly justify your answer. *(4 points)*



The gamut of Bob's space is much smaller than CIE RGB. You must take positive amounts of primaries to construct colors in the gamut, which is the set of all physically realizable colors! Any linear combination of Bob's primaries with positive coefficients can be realized with positive amounts of the RGB primaries as well, but R, G and B themselves cannot be constructed in Bob's space, since each of his primaries has at least two of R, G and B. We also accepted the graphical chromaticity argument on the right. Note that the fact that the component beams are half-intensity has nothing to do with the answer. Bob's gamut would still be smaller if he used 2(R + B), 2(B + G) and 2(G + R) instead.

2. Cameras, Displays and Compression *(17 points)*

2a) Alice is exploring Central Asia, and is trying to take a picture of wild horses running across the plains, with high peaks in the distance. She places her camera on a tripod, focuses on the horses (which are fairly close to her), and clicks the shutter. Upon reviewing the shot, she finds the exposure is correct (so the picture is neither too bright nor too dark), but both the horses and the peaks are blurry. How should she adjust her camera variables to take a picture which is equally well-exposed, but in which everything is sharp? (Tick one option for each variable.) *(3 points)*

Shutter Speed:	Shorter exposure time ✔	Longer exposure time _____
Aperture:	Increase _____	Decrease ✔
Sensitivity (ISO):	Increase ✔	Decrease _____

(Shorter exposure time to freeze moving horses, smaller aperture for depth of field extending to the mountains, higher sensitivity to compensate for the other two changes.)

2b) Recall that a Bayer filter is a mask of red, green and blue filtering pixels placed over a digital sensor. State one advantage and one disadvantage of this filter. *(2 + 2 points)*

Advantage: *Provides sensing in color!*

Disadvantage: *Some are:*

- *Reduces sensitivity compared to base (monochrome) sensor*
- *Introduces demosaicing artifacts.*

(We realized in retrospect the question was ambiguous, since we had not specified what we were comparing a Bayer sensor to. We intended a comparison with the underlying monochrome sensor. All answers that correctly contrasted Bayer with other types of color sensors were also given full credit.)

2c) Camera lenses typically contain multiple individual elements (separate pieces of glass), even when the lens has a single focal length and does not zoom. Why do you think this is so? *(2 points)*

To correct aberrations.

2d) Daila tosses a fair coin a million times to produce a sequence of heads (0) and tails (1). Would you expect this sequence to have high or low Kolmogorov complexity? Briefly justify your answer (an informal argument is fine). *(2 points)*

With great probability, the string has high Kolmogorov complexity. We require <u>exact</u> reconstruction, so tossing the coin again is not a reconstructive procedure.

FYI: This observation is called the Incompressibility Theorem. A string of n bits is c-incompressible if its minimum description length exceeds n – c bits. Since there are only $1 + 2 + 2^2 + \ldots + 2^{n-c} = 2^{n-c+1} - 1$ *descriptions with at most n – c bits, the number of c-incompressible bitstrings is at least* $2^n - 2^{n-c+1} + 1$*, which rapidly approximates* 2^n *as c grows. In short, a long random string is almost certainly not compressible by more than a very small amount. (You didn't need to state or prove this theorem for full credit.)*

2e) Eve decides to construct her own "wavelet" encoding for sequences of positive numbers. For every successive pair of numbers <A, B>, she computes the "average coefficient" as sqrt(A * B), and the "detail coefficient" as sqrt(A / B), where sqrt denotes the (positive) square root. In other words, she replaces the arithmetic mean (A + B) / 2 of Haar wavelets with the geometric mean.

- Compute the "wavelet" transform of [4 4 1 1] using this encoding (Hint: this is another sequence of the same size.) *(4 points)*

<u>Average</u>	*<u>Detail</u>*
4 4 1 1	
4 1	*1 1*
2	*2*

So the "wavelet" transform is [2 2 1 1].

- Would this be a good way to compress data on a typical PC? If so, why, and if not, why not? *(2 points)*

It would not, for several reasons including:

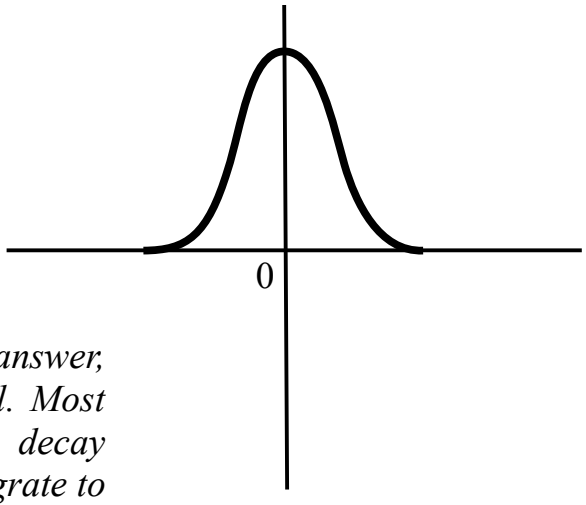
- *More floating point roundoff errors with multiplication, division and sqrt*
- *Inefficiency of sqrt*
- *Possibility of division-by-zero errors*

3. Image Processing *(7 points)*

3a) Is the dynamic range of an image that has 16 bits per channel necessarily greater than that of an image with 8 bits per channel? Argue for or against. *(2 points)*

No, it is not. 16 bits are better for resolving subtle variations in intensity, because they subdivide the intensity scale more finely. However, the upper and lower ends of the scale, indicating the maximum and minimum intensities that can be represented in the image (the dynamic range), are completely independent of the bit depth, and are typically related to the device used to capture the image or display it.

3b) Here is a signal *f* that needs to be smoothed (blurred). On the empty graph on the right, sketch a one-dimensional kernel *g* that you think will best smooth *f* when convolved with it. The overall scale of your drawing is not important. *(2 points)*



(A gaussian, as shown, is probably the best answer, but we accepted box, tent etc. filters as well. Most curves that are maximum at 0 and decay symmetrically to zero on either side (and integrate to 1, but we didn't require you to specify this) will work. A constant function is not a valid answer: that will just flatten the entire curve to a constant value.)

3c) Carol shrinks an image to half its size (one-quarter its area) by deleting every second row and every second column of pixels. Is this the best way to resize the image? If not, how would you do it? *(3 points)*

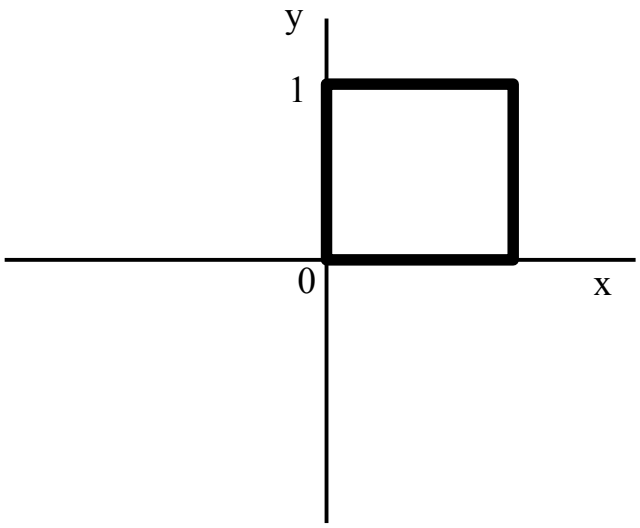
This is not a good way to resize an image, since it throws out too much data. (Consider, as one of you pointed out, what happens to an image where alternating columns of pixels are black or white.) Observe that each output pixel corresponds to the area covered by a 2x2 block of input pixels. Assigning each output pixel the average color of the corresponding 4 input pixels will instantly improve matters.

4. Geometry *(6 points)*

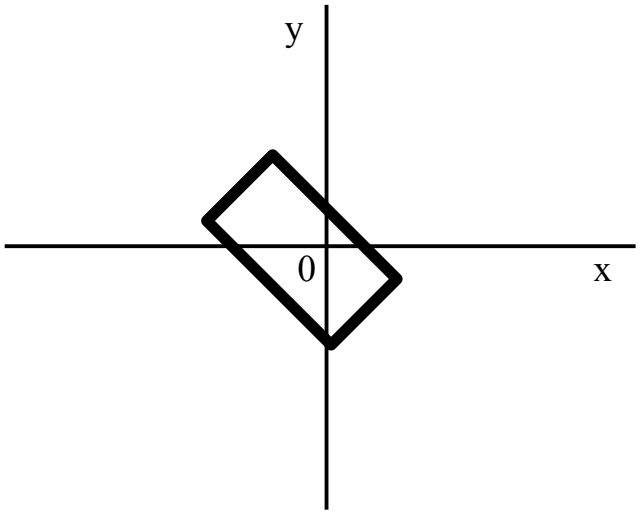
We'll use the following notation for 2D transformation matrices:

T(x, y): Translates a point by (x, y)
R(a): Rotates counter-clockwise around the origin by a degrees
S(sx, sy): Scales the coordinates of a point, relative to the origin, by (sx, sy)

You are given a unit square with minimum corner (0, 0):



In the graph below, sketch what happens to the square when we apply
T(0, -0.5) * R(45) * S(0.5, 1) to it. Remember: we apply transforms right-to-left. Overall scale of your drawing is not important.



5. Rendering *(12 points)*

Anand wants to render an outdoor scene with a fog effect. He observes that fog has the following characteristics:

- it has a color of its own, usually white but sometimes other colors
- shapes are faded to the fog color with distance

5a) Help Anand write a **fog**(*d*, *src_color*, *fog_color*, *fog_density*) function that takes 4 parameters:

- *d*: the distance to an object
- *src_color*: the original (shaded) color of the object
- *fog_color*: the color of the fog
- *fog_density*: how quickly shapes fade to the fog color – greater density implies a quicker fade

fog(...) returns the perceived color of the object seen through the fog. Write either a math formula or pseudocode for a plausible **fog**(...) function. (There is no single correct answer. Your solution should satisfy Anand's two criteria at least.) *(4 points)*

One simple solution is:

lerp(a, b, x) = $(1 - x) * a + x * b$
fog(...) = **lerp**(*src_color*, *fog_color*, min(d * *fog_density*, 1))

POV-Ray uses this variant instead:

fog(...) = **lerp**(*src_color*, *fog_color*, $1 - \exp(-d * \textit{fog\_density})$)

As a sanity check, you should ensure that your formula gives src_color when d = 0, and fog_color when d is very large.

5b) Extend the basic raytracing loop, in pseudocode, to handle a uniform fog throughout the scene. You may assume you are given the **fog**(...) function as a black box. Ignore refractions. *(8 points)*

Note: Yes, we do want you to write the whole loop, ignoring refractions. Syntax is not important, and you can assume you have all the math functions available to you.

We'll assume the constants FOG_COLOR and FOG_DENSITY are predefined. The really tricky part is modifying the diffuse shading calculations to take into account the fog between the intersection point and each light source. We'll make the simplifying assumptions that each light source is a point (or can be approximated as one) and lights the object directly. The maximum fog component in the net diffuse color is limited by FOG_COLOR. Hence we can't simply apply the **fog**(...) *function to the diffuse color contributed by each light source, since that could add up to a net fog component n * FOG_COLOR, where n is the number of lights. Nor can we apply* **fog**(...) *to the net diffuse color, since that gives equal weight to both near and far light sources. Our solution apportions (1 / n)th the total fog for each light source, which is a bit of a hack. You get close to full credit if you recognized this as an issue.*

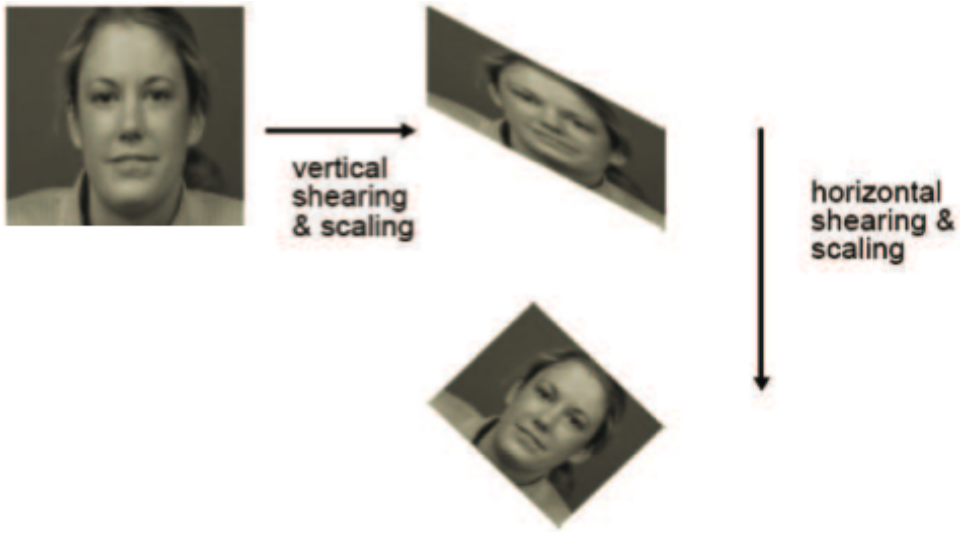
```
function traceRay(ray) returns Color
    (obj, intersection) = getFirstIntersection(ray)
    d = distance to intersection point
    if obj is a light source
        light_color = getLightColor(obj)
        return fog(d, light_color, FOG_COLOR, FOG_DENSITY)
    else
        diffuse_color_fog = RGB(0, 0, 0)
        n = total number of lights
        foreach light source light
            diffuse_color_nofog = getDiffuseShading(light, obj, intersection)
            d_light = distance to light
            diffuse_color_fog += fog(d_light, diffuse_color_nofog,
                                     FOG_COLOR / n, FOG_DENSITY)
        end foreach

        reflected_ray = getReflectedRay(ray, intersection)
        reflectance = getSurfaceReflectance(obj, intersection)

        color = combine(diffuse_color_fog, reflectance * traceRay(reflected_ray))
        return fog(d, color, FOG_COLOR, FOG_DENSITY)
    end if
end function
```

6. Extra Credit *(+10 points)*

Recall that a 2D shear matrix (for non-homogenous coordinates) is one that has the form $\begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$ (horizontal shear) or $\begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$ (vertical shear). There is a connection between shearing and rotation, as can be seen in this sequence.



This has been exploited in image processing, as shearing maps are easily performed by shifting scan-lines (columns or rows). Explore this connection by writing the 2D rotation matrix $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ as a product of a horizontal shear, then a vertical shear, and finally another horizontal shear. Show the steps in your reasoning.

Many of you got the approach right for this one. The decomposition was described by Alan Paeth in "A Fast Algorithm for General Raster Rotation", Graphics Interface '86. Writing out the desired relation in matrix notation, we have:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1+\alpha\beta & \alpha \\ \beta & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1+\alpha\beta & \gamma(1+\alpha\beta)+\alpha \\ \beta & \beta\gamma+1 \end{bmatrix}$$

Equating corresponding elements of the LHS and RHS, we get 4 equations in 3 unknowns (α, β and γ) which are fortunately consistent. Solving any 3 of them, we get

$\alpha = \gamma = (\cos \theta - 1) / \sin \theta$
$\beta = \sin \theta$

This is enough for full credit. However, a further simplification is possible, if you note (by standard trigonometric identities) that $(\cos \theta - 1) / \sin \theta = -\tan (\theta / 2)$. *So*

$\alpha = \gamma = -\tan (\theta / 2)$
$\beta = \sin \theta$