Faculty of Science
FINAL EXAMINATION

COMPUTER SCIENCE   COMP 557
FUNDAMENTALS OF COMPUTER GRAPHICS

Examiner:	Prof. Michael Langer	April 29, 2015
Associate Examiner:	Prof. Gregory Dudek	6 PM – 9 PM

LASTNAME, FIRSTNAME: ______________________________

STUDENT ID: ______________________________

Q1		Q6		Q11	
Q2		Q7		Q12	
Q3		Q8		Q13	
Q4		Q9		Q14*	
Q5		Q10		Q15	

* Q14 was graded out of 2 rather than 4, so the exam was out of 58 (not 60).

Instructions:

The exam has 9 pages, including the cover page.

Use the back side of the page if you need more space.

The exam consists of 15 questions. There are 4 points per question, for a total of 60 points.

The exam is open book.

No calculators or electronic devices are allowed.

1. Consider a sequence of OpenGL instructions that does the following. (You do *not* need to write out the instructions.) It defines a unit square object centered at the origin in the $z = 0$ plane. The unit square is transformed as follows. First, it is scaled by 2 in the y direction. Then, it is rotate by 45 degrees around the x axis. Then, it is translated by -5 in the z direction.

   What is the `GL_MODELVIEW` matrix that performs this transformation? (It is fine if your answer is a sequence of matrices.) State any assumptions you make about the camera.

SOLUTION:

This was an easy question, a warmup.

Assume the world coordinates are the same as camera coordinates, so the world to camera mapping is the identity.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

GRADING:

1 point for stating your assumptions about the camera, as stated in the question.

1 point for each of the three transformations.

(Lose 1 point if the order of the transformations was inverted.)

2. Suppose you have two cameras. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the camera positions and let $\{\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1, \hat{\mathbf{z}}_1\}$ and $\{\hat{\mathbf{x}}_2, \hat{\mathbf{y}}_2, \hat{\mathbf{z}}_2\}$ be the respective camera axes. All of these are expressed in a world coordinate system.

   Write out the sequence of matrices that takes any scene point that is expressed in camera 1's coordinate system and maps it to camera 2's coordinate system.

   SOLUTION:

   If you did the assignments and you are comfortable with linear algebra, then this should have been an easy question.

   The main idea is to map from camera 1 coordinates to world coordinates and then map from world coordinates to camera 2 coordinates.

$$\begin{bmatrix} & \hat{\mathbf{x}}_2 & & 0 \\ & \hat{\mathbf{y}}_2 & & 0 \\ & \hat{\mathbf{z}}_2 & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & -\mathbf{p}_2 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \mathbf{p}_1 \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & & & 0 \\ \hat{\mathbf{x}}_1 & \hat{\mathbf{y}}_1 & \hat{\mathbf{z}}_1 & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   GRADING:

   1 point for general idea of using world coordinates as an intermediary.

   1 point for idea of using two rotations.

   1 point for idea of using two translations (or combining them into one translation by the difference vector).

   1 point for putting them together correctly.

   Some students wrote symbols T_1, R_1, T_2, R_2 and defined them (translating from camera to world, etc). But most students went further and expressed those matrices in terms of the given vectors as in the solution above.

3. Suppose we want to interpolate the RGB values in a texture image. We have seen how to do this using linear interpolation and bilinear interpolation.

   Another common method that is used to solve this problem is *bicubic image interpolation.* Explain how this method could work, by applying the idea of a similar method namely bicubic surface interpolation.

   To be concrete, assume you want to interpolate the RGB values for texture position (s_p, t_p) which might not be integers.

   SOLUTION:
   This was a more challenging question. It required that you apply the concept of bicubic surface interpolation to a different domain.

   A bicubic surface has the form $(X(s,t), Y(s,t), Z(s,t))$ and in the lecture we discussed how to interpolate the values XYZ for some (s,t), given the values of XYZ at a set of control ponints. The question is asking you to apply this idea to image texture interpolation. Images are defined by $(R(s,t), G(s,t), B(s,t))$. So, you just needed to associate RGB with XYZ.

   For full points, you need to specify that the (s,t) values are texture coordinates. These could be pixel positions on which a texture RGB image is defined or points between pixels, where the RGB values are not defined i.e. and so you need to interpolate them e.g. for texture mapping.

   To do the interpolation, we could take (s_p, t_p) values in a 4×4 pixel neighborhood that contains the (s,t) value in question. We would fit a bicubic surface $(R(s,t), G(s,t), B(s,t))$ in that 4×4 neighborhood and substitute the particular (s,t) to obtain the interpolated RGB value.

   GRADING:

   The answers for this question were in many cases very fuzzy.

   2 points for the stating the main idea that the RGB values play the role of the XYZ values and so the goal is to interpolate RGB colors.

   2 points for stating that the (s,t) are image texture coordinates and that the 4×4 pixels in the neighborhood of a desired RGB(s,t) would serve as the control points.

4. Suppose that a shiny ground plane, $y = -2$, is illuminated by sunlight. Let the sun be in direction $(1, 2, 3)$ and let the camera be at position $(x, y, z) = (-6, 4, -4)$. Determine the position on the ground plane at which the peak of the highlight occurs.

   SOLUTION:
   This question should have been very easy. It was similar to one in the exercises.

   The reflection direction $\mathbf{r}$ is $(-1, 2, -3)$. This is easy to see if you understand how the reflection direction is defined. Many of you computed the reflection direction using the formaul given in class, and this led to a more complicated expression, namely you normalized the vector which led to carrying around a $1/\sqrt{14}$ term. The derivation I give does not use that term.

   We first consider the ray from the reflection point to the eye:

   $$(x, -2, z) = (-6, 4, -4) + t(-1, 2, -3)$$

   Use the y coordinate to solve for t gives $t = -3$. Substituting gives $(x, y, z) = (-3, -2, 5)$.

   GRADING:

   This question directly followed exercises questions and it was handled very well. We took of 0.5 points if you made a careless error in the calculation. Some students put the light source at position $(1, 2, 3)$ rather than in direction $(1, 2, 3)$ which gave a different answer and we took off 0.5 for that.

5. Describe the visual effect of using the procedural shading formula:

$$I_{RGB}(\mathbf{x}) = \Big(\frac{1+\mathbf{n}(\mathbf{x})\cdot\hat{\mathbf{l}}}{2}\Big)(1,0,0) + \Big(\frac{1-\mathbf{n}(\mathbf{x})\cdot\hat{\mathbf{l}}}{2}\Big)(0,1,1)$$

where $\hat{\mathbf{l}}$ is the unit light source direction.

SOLUTION:

As many of you noted, the model here is similar to that of the hemispheric light source of Exercise 19 Q2.

For the model in this question, the surface would be pure red at points where the normal points in direction $\mathbf{l}$ and it would be cyan at places where the normal direction is opposite to direction $\mathbf{l}$.

For all other points, the color would be vary between pure red and pure cyan.

The color would be neutral (0.5, 0.5, 0.5) at points where the normal is perpendicular to the light source.

GRADING:

1 point for saying anything that indicated there would be non-zero intensity for all surface normals (not just those such that $\mathbf{n}\cdot\mathbf{l}\geq 0$).

1 point for the pure red observation.

1 point for the pure cyan (or blue+green) observation.

1 point for the gray observation.

6. For the previous question, could there be any difference in the resulting I_{RGB} values if you were to implement this shading model using a vertex shader versus a fragment shader? If so, give an example. If not, why not?

SOLUTION:

To compute I_{RGB} using a vertex shader, you would color the vertices using that model. The fragment shader would then interpolate the vertex colors across the fragments.

To compute I_{RGB} using a fragment shader, you would apply the model for each fragment, based on the surface normal of that fragment. This requires using interpolated normals, namely each fragment has its own normal the shading model is applied based on that normal.

Yes, there are differences between the two cases. For example, suppose two vertices of a triangle have different normals $\mathbf{n}$ but these two vertices have the same value of $\mathbf{n}(\mathbf{x}) \cdot \hat{\mathbf{l}}$. Using the vertex shader for I_{RGB}, the RGB values would be interpolated along the line between the two vertices and this would give a constant RGB on that line since the RGBs at the vertices are the same. Using the fragment shader for I_{RGB}, the normals would be interpolated along the line. This might not give a constant RGB values. For example, if the $\mathbf{l}$ were halfway between the two normals, then there would be a saturated red point between the two normals.

GRADING:

2 points for saying there could be a difference because the surface normal is changing and this would matter because the normal is used in one case but not the other.

2 points for giving an example, as requested in the question.

Some said there could be a difference but could not explain why, other than to say that in Blinn-Phong shading there is a difference so there would be a difference here too.

Some said that using the fragment shader would produce a smoother solution since the normals vary. This is at best too vague, and at worst it is wrong since, within a polygon, the vertex shader would provide a smoother solution (namely bilinear interpolation). Indeed the point of Phong shading is to be able capture sharp highlights within a polygon.

7. A 300×200 texture image is mapped onto a 3D quad which is a parallelogram defined by vertices $\mathbf{p}_{00}, \mathbf{p}_{10}, \mathbf{p}_{01}, \mathbf{p}_{11}$. Pixel $(0,0)$ in the texture maps to $\mathbf{p}_{00}$. Pixel $(299,0)$ maps to $\mathbf{p}_{10}$, etc.

   What is the mapping from pixel (s_p, t_p) in the texture image to 3D coordinates on the quad ?

   Write you answer as a product of matrices and be sure to specify the dimensions of the matrices.

   SOLUTION:
   This question should have been easy. This scenario was very similar to the one in lecture 16 except that here we have a parallelogram and there we had a triangle.

$$\begin{bmatrix} \mathbf{p}_{10} - \mathbf{p}_{00}, & \mathbf{p}_{01} - \mathbf{p}_{00}, & \mathbf{p}_{00} \\ 0 & 0 & 1 \end{bmatrix}_{4\times 3} \begin{bmatrix} \frac{1}{299} & 0 & 0 \\ 0 & \frac{1}{199} & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3\times 3}$$

   GRADING:

   1 point for the scaling matrix.

   3 points for the 4×3 matrix mapping to $\Re^3$.

   Took off 0.5 if homogeneous coordinates (4th row) not used.

   It was fine if you used 300 and 200 for the scaling matrix instead of 299 and 199.

8. We saw how in OpenGL one can render a glossy surface using point light sources and a surface material with a specular component.

   It is also possible in OpenGL to render a glossy surface by MIP-mapping an environment map. Briefly describe how and why this would work. A few hints:

   - Recall that environment mapping assumes the surface is a mirror. But the question is asking about rendering a surface that appears glossy.
   - Assume the MIP-mapped environment is represented by a set of cube maps.

SOLUTION:

This was one of the more challenging questions on the exam.

First, you need to say something about how environment mapping could be used to render glossy surfaces. (This idea was hinted at in lecture 18 slide 42 but there I only planted the seed for you think about.) What happens when you illuminate a glossy surface with an "environment"? Consider a ray from a point on the surface to the camera. If the surface were a mirror, then there would be a single environment direction that would be reflected in the direction of that ray. If the surface were glossy, then what would happen? There would be a cone of environment directions that would be reflected to the camera along that ray. To compute the RGB value of the ray in the glossy case, we would need to add up the contributions from the environment directions in this cone. The shinier the surface, the smaller the area of the environment map.

Second, you needed to say how MIP-mapping could be used here. MIP mapping means we have multiple levels (or sizes) of the texture or in this case the environment map. Each pixel at level i of the environment map represents a larger area on the sphere of environment directions when i is larger. If the shininess exponent e is small, we would use a larger value of level i.

GRADING:

We did not expect the full explanation above, of course. We were just looked for the main ideas.

2 points for explaining how environment mapping could be used for glossy surfaces (the idea that a cone of rays arriving at the incoming surface could be contribute to each ray to the camera).

2 points for saying that MIP mapping the environment map means that multiple sizes of the environment map would be used and larger i corresponds to smaller shininess (smaller exponent, more scattering).

If your explanation for either of the above was too fuzzy, you got less than full points.

9. Normal maps are related to bump maps, but one stores unit normals instead of bumps. A normal map is of the form:

$$(s_p, t_p) \to (n_x, n_y, n_z) \in [-1,1] \times [-1,1] \times [0,1]$$

where we note that $n_z > 0$, i.e. outward normal.

Assuming a normal map is stored as an RGB texture, with 8 bits per pixel , i.e. $\{0, ..., 255\}$, write out a matrix representation of the mapping from RGB values to unit normal vectors (n_x, n_y, n_z) that you would need to use such a normal map. Your answer may be a product of matrices.

[EDIT: The question should have said "8 bits per channel" not "8 bits per pixel."]

<u>SOLUTION:</u>

The challenge here was to understand the scenario and to isolate in your mind what I was asking for. There were several elements needed to set up the scenario, and it was easy to get distracted by them: bump maps vs normal maps, number of bits, the mapping from *st* to **n** (which is different from the mapping I was asking for).

Now for the solution: You were asked to map the 0 to 255 range (of the RGB representation) to the range of the n_x, n_y, and n_z values. The n_x, n_y have range -1 to 1 and n_z has range 0 to 1.

There were a few ways to write out the answer, but this was the most common one:

$$\begin{bmatrix} n_x \\ n_y \\ n_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{255} & 0 & 0 & 0 \\ 0 & \frac{1}{255} & 0 & 0 \\ 0 & 0 & \frac{1}{255} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_R \\ I_G \\ I_B \\ 1 \end{bmatrix}$$

<u>GRADING:</u>

3 points for the scaling transformations (n_x and n_y treated differently from n_z)

1 point for the translation.

10. In the ambient occlusion method explained in the lecture, one precomputes a fractional visibility at each vertex. This visibility is a value in $[0, 1]$. Suppose one were instead to pre-compute a boolean visibility map for each vertex e.g. using a cube map data structure. The advantage would be that environment maps could be used, and cast shadows could properly accounted for.

    Describe how to compute such boolean visibility maps *using the technique of shadow mapping.*

    SOLUTION:
    This was a more challenging question.

    You usually think of shadow maps as being defined by a single point source, either in the scene or at infinity. But now I am asking you to think about shadowing for the situation in which ambient occlusion is used, namely that the light can come from any direction *e.g.* the sky. Moreover, the question says that we would like to use an environment map; so rather than assuming the sky (environment) is uniformly bright, we allow a more general environment distribution.

    The question doesn't ask you to explain how environment maps could be used (although in hindsight, I could have added that). Rather it just asks how the "boolean visiblity map" could be computed. It specifically suggests a cube map data structure at each vertex.

    Now, to the answer: Each pixel on the cube map defines a direction vector pointing to a point at infinity i.e. the environment is assumed to be far away. For each of these directions, make a shadow map using orthographic projection. Once you have that set of shadow maps, now take each vertex and define a cube map at these vertices as defined in the qustion, namely for each direction of the cube map at a vertex, decide if the "sky" is visible in that direction. Do that using the shadow map defined by that direction.

    GRADING:

    4 points for the main idea of defining one shadow map for each of the N directions on a cube map, and then for each vertex in the scene, compute a boolean cube map using the N shadow maps.

    For part marks:

    2 points if you said to do ray tracing (or depth buffer algorithm): for each vertex, cast a set of rays and see which go to infinity (true) and which don't (false). Represent the result as a boolean cube map. This solution doesn't answer the question, namely how to use shadow mapping.

    1 point for a solution that used "for each light source".

    1 point for put lights at infinity, since the question is about ambient occlusion which usually is about whether the sky (or points beyond some radius) is visible.

11. Sharp Electronics sells a color television that has four light emitting elements (RGBY) at each pixel, where Y is yellow. The company claims it "delivers color never seen before on TV".

    Use linear algebra to explain what this claim means, in terms of the color spectra emitted by such TV's and the eye's ability to measure color spectra.

    SOLUTION:
    A display with RGBY has a P matrix which is N x 4 and you can assume it produces a 4D vector space of emitted light spectra, instead of 3D space produced by RGB monitors. Why can you assume this? Obviously the fourth (Y) column of the P matrix would have a different spectrum than any of the first three columns, since we are calling it Y and not RGB. Moreover, the fourth column would *not* be a linear combination of the first three columns, since that be very limiting. The whole point is to increase the dimensionity of the space spanned.

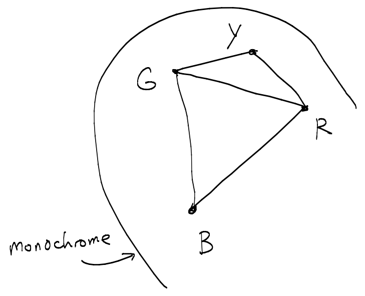
    The issue is whether RGBY "delivers more colors" than RGB. The answer is YES. Why ? The cone photoreceptor space is 3D and so you might think that a 4D spectrum space would not reach more RGB colors. However, keep in mind that you must send out positive spectra only i.e. you cannot have negative I_{RGB} values in the image you are displaying. Similarly, you cannot have negative cone (or camera) sensor response values. This concept came up in Exercise 23 Q2.

    BTW, you may be wondering how an RGB (say JPG) image can be used to display an RGBY image. One way is to set the strength of the emiited Y to be the average of the R and Gs. Many of you did mention the idea that "yellow = red + green".

    GRADING:

    1 point for saying that the Y adds a 4th column the matrix P.

    1 point for saying the Y emittor increases the dimension of the spectrum space from 3 to 4.



    2 points for explaining something about how the eye can see more colors in RGBY displays than in RGB displays, even though the eye has only 3 types of photoreceptors. A figure such as above as commonly used.

12. How could alpha blending be used for anti-aliasing, when drawing lines over a pre-existing image? Assume a line has a thickness attribute.

    Where would this occur in the OpenGL pipeline? Explain.

    SOLUTION:
    Anti-aliasing of lines was discussed earlier in the course but at that time we didn't have the idea of alpha and transparency. We 'solved' the problem by using grey pixels near the line instead of forcing ourselves to black or white. The point of this question is for you to notice you can manipulate alpha instead of manipulating the color.

    To answer the question.... In the OpenGL pipeline, a line is broken up into fragments by the rasterizer. If a line of a certain thickness only partly covers a pixel, the rasterizer would assign the fragment an alpha value between 0 and 1.

    The fragments of the line would then be blended with the background in the usual F over B way.

    GRADING:

    1 point for saying alpha is computed by the rasterizer.

    2 point for saying the alpha value of a "fragment" is based on how much the line covers the (little square) pixel. Only 1 point for saying pixel and not using the word "fragment" in the answer.

    1 point for saying the blending is done by the fragment shader.

13. Describe a 3D data set for which:
    (a) an octree could be used to speed up volume rendering.
    (b) an octree could *not* be used to speed up volume rendering.

SOLUTION:

Recall how ray casting with octrees improved the speed relative to uniform spatial partition. The ray could jump over big parts of empty space since there was nothing to intersect with. The question is asking you to apply this to volume rendering.

(a) If a large contiguous part of the volume has zero opacity, then there is no need to go step by step through the layers and blend successive layers. Instead you could skip over many layers. (good)

(An alternative argument might be that if the opacity is constant over a contiguous part of the volume, then you could use a formula like OpenGL fog to avoid doing layers one by one.)

(b) If the opacity is varying and non-zero over space, then octrees would be useless. (bad)

GRADING:

Many of you did not seem to know that volume rendering concerns data that is defined on a 3D grid e.g. a medical image. You did not use the word "opacity" or "transparency" in your answer and instead it seemed you were answering the question as if it used the term "ray casting" instead of "volume rendering".

For part marks, we gave 3 points if you answered it in terms of ray casting and didn't mention 'alpha' or 'transparency' but instead you discussed having lots of objects distributed over the space and leaving no big gaps (b), versus having big gaps (a).

Some of you distinguished static from dynamic scenes and we gave 2 points for that if you explained it (although that was generous).

14. In photography, *backlighting* refers to a situation in which the direction of the light source is greater than 90 degrees away from the viewer direction.

    Suppose in OpenGL you enable *backface culling* and you illuminate the scene with backlighting. Under what conditions (if any) will a polygon be visible in the image and also illuminated ? Explain.

    SOLUTION:

    First of all, back face culling has nothing to do with lighting. Backfacing culling only concerns geometry; it determines whether a polygon is rendered or not: if a polygon is a backface ($N \cdot V < 0$), then it will not be rendered. Specifically in OpenGL, it will not have any fragments generated.

    If a surface is NOT backface culled, it still CAN be illuminated by back lighting. For example, take the case of a "crescent moon".

    The geometric concepts needed for this question are familiar to you, if you understood the environment mapping examples, in particular, where the light source direction can more than 90 degrees away from the viewer and yet the light can reach the surface at points that are visible to the eye.

    GRADING:

    During the exam, several asked me to clarify the definition of backlighting. Evidently the wording was poor. Because of the poor wording, we graded this question out of 2 rather than 4.

    1 point for saying what the two conditions are, in which a polygon could be illuminated (namely the surface normal must be at most 90 degrees from the viewing direction, and the light source direction must be at most 90 degrees from the normal).

    1 point for describing a situation in which the two conditions could be satisified.

    An example is that the light source (sun) is say 135 degrees away and the surface is a sphere. There will be points near the rim of the sphere that satisfy this condition.

    Note that the concepts needed for understanding this situation are very similar to the concepts discussed in environment mapping exercises, in particular, where we considered points that are further in depth than the mirror object we are rendering and yet are still visible in the mirror. (If you are confused reading this, then note that this question is NOT about mirrors. I'm just saying here that the concepts are similar. As always, feel free to see/email me if you have questions.)

15. Adobe Lightroom is an application that allows you to process digital images. In particular, it can read in a RAW image and it allows you to specify the camera response function by manipulating a set of control points.

    Consider an image of a forest where the ground is partly covered in snow. Sketch an appropriate camera response function that would show the detail within both the snow and the non-snow regions of the image. Explain your answer, and state your assumptions.

SOLUTION:

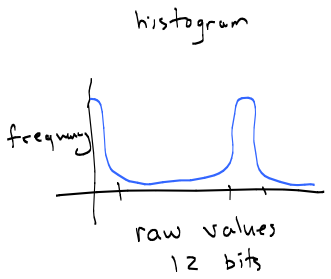
How to approach this question? From the description of the scene, you should conclude that the intensity frequency distribution (”histogram”) will have two peaks, corresponding to the trees and snow covered ground respectively. You shoudl also assume that the dynamic range of such a scene will be large, and so your camera response function will have to do something to compress it into 8 bits per intensity, in order to show the details within *both* the snow and non-snow regions.

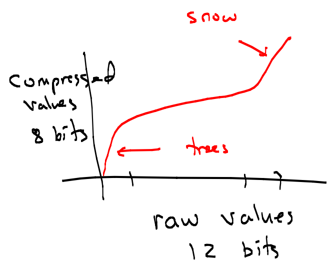
Now to the answer...

The intensities of the trees will have a distribution of small values only, so to discriminate these values you need a steep slope for the transfer function over those intensities.

For the snow covered ground (or snow on the branches of the trees), you will also need to use a high slope in order to show the details in those intensities.

Since there will be a gap between the two peaks of the intensity distribution, the camera response function should have a shallow slope over the gap, so you don't waste too many coding values for the gap region.





GRADING:

This was a challenging question. I expected that many of you would get it for the dark region, since the lecture explicitly discussed how to use a steep slope in the camera response to get more detail in dark regions. The tricky part was to handle the gap and the snow.

2 points for a steep slope over the high (snow) intensities, but getting the tree part wrong.

2 points for a steep slope over the low (trees) intensities, but getting the snow part wrong.