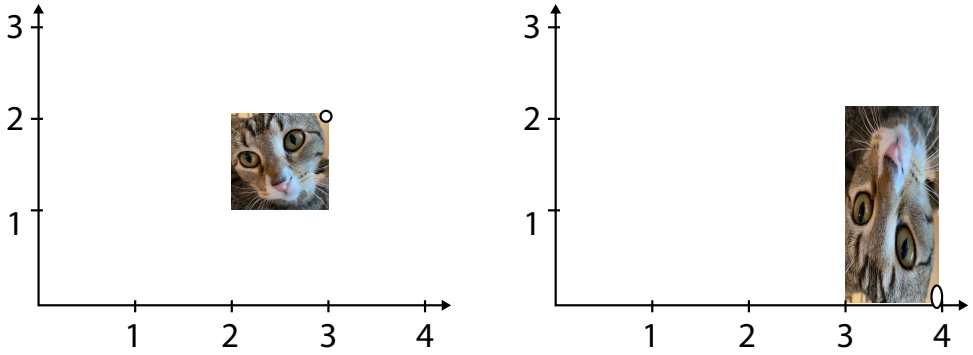
Stanford CS248: Interactive Computer Graphics
Participation Exercise 1

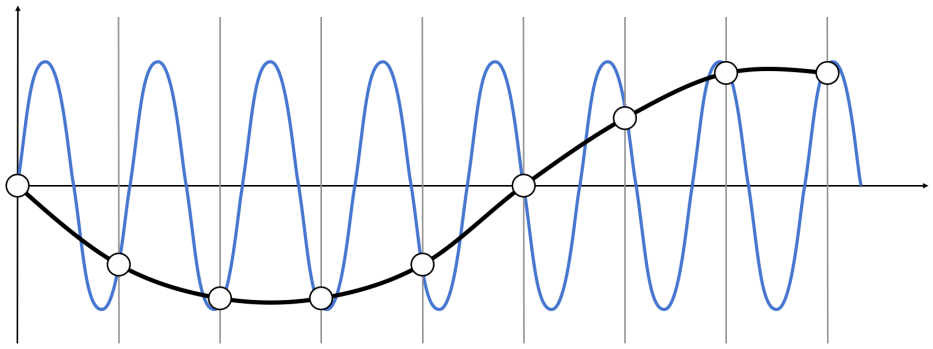
Problem 1. Please give a sequence of transforms that takes the 1×1 square object centered at (2.5, 1.5) to its position and orientation on the right. Valid transforms include, translation, rotation (counter-clockwise), scale, and shear. Since rotations and reflections can look similar, note that we've placed a dot on the image to help you determine the correct transforms.



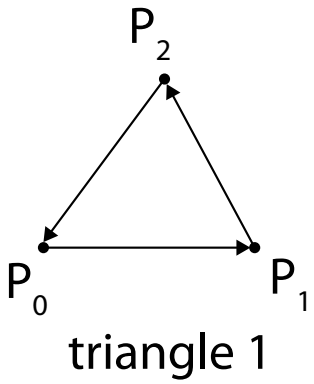
Problem 2. You are asked to make "super-rad lighting effects" for the Computer Science Department's Spring techno dance party. You are given access to strobe lights for which you can control the strobe frequency. Assume that the dance party is going to be held in a room with big ceiling fans rotating at 10 revolutions per second. If you want to make the ceiling fans appear to the dancers as if they are **slowly moving backwards**, how many strobes per second would you configure your strobe lights? (Give a number, but please explain your reasoning.)

Problem 3. Assume you are rendering an image to a screen that is 1500 pixels wide and 1500 pixels tall. Your viewpoint is set so that all points contained in a rectangular region with bottom-left corner coordinate (2,2) and top-right corner coordinate (12,22) are visible on screen. *Assume the bottom-left corner of the box maps to the bottom-left corner of the screen. (Note this is a different mapping of coordinates to screen points than what you saw in programming assignment 1.)* In what output screen pixel does the point at coordinate (4.01, 3.1) appear on screen? Assume pixel (0,0) is the bottom-left pixel on screen and pixel (1499, 1499) is the top-right pixel. Please show your work.

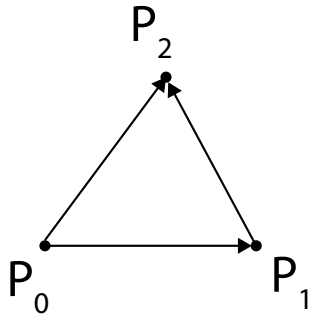
Problem 4. A major theme in the class so far has been drawing pictures via sampling, and dealing with the objectionable image artifacts that can come when undersampling causes aliasing. Using the figure below, which shows a high frequency signal (blue line) sampled at a low rate (whit dots are the sample points), describe why we call the artifacts created by reconstructing an undersampled signal *aliasing*. (The reconstructed signal is the black line.) Hint: An alias is another name for a person. In this example, what can we not tell the difference between given the information contained in our samples?



Problem 5. You might have noticed that when implementing point in triangle tests, it's not just the position of vertices that define a triangle, but the order of vertices can matter as well. Consider the points P_0, P_1, P_2 below. Consider triangle 1 with edges $e_0 = P_1 - P_0$, $e_1 = P_2 - P_1$, and $e_2 = P_0 - P_1$. also triangle 2 with edges $e_0 = P_1 - P_0$, $e_1 = P_2 - P_0$, and $e_2 = P_2 - P_1$. Why does the algorithm for `inside_triangle(x,y)` given in class (where the sample point (x,y) should be on the negative half plane of all edges), not work for triangle T_2? In other words, why is it important for your triangles to have *"consistent winding"*. Note triangle 1 is using a counter-clockwise winding. How would you change your edge tests if you knew triangles were supposed to have a consistent, but clockwise winding?



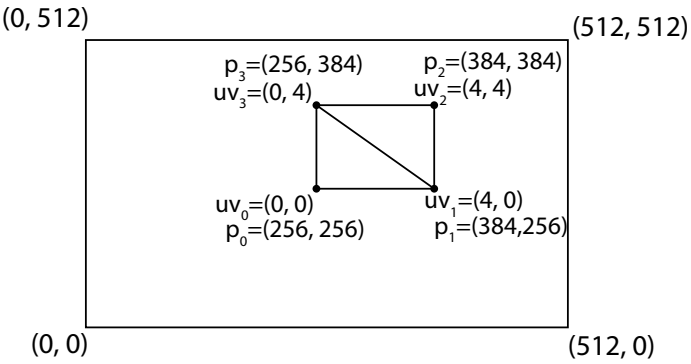
triangle 1



triangle 2

Problem 6. Consider rendering a texture-mapped quadrilateral formed by the two triangles shown below. The quadrilateral is rendered onto a 512×512 pixel screen, and the screen coordinates of the quadrilateral's vertices are given below. (The quad is 128×128 on screen.)

512x512 texture



Notice that the texture coordinates (uv) associated with each vertex are not constrained to be between 0.0 and 1.0. In assignment 1 you implemented texture border behavior of "clamp to edge" but another common behavior when texture mapping is to have texture coordinates "wrap". *In other words, texture coordinates are still interpolated over the surface of a triangle as normal, but if the value of the texture coordinate at a sample point is a, then texture mapping would use the fractional part of a, in other words, $a - floor(a)$, for use in the texture lookup*. (Yes, this means that when bilinear filtering is enabled, a bilerp operation might blend between pixels on the right column of the image and the left column (similarly for the top and bottom row)).

Problem 6.A. Describe the image that you will see when the scene is rendered. In particular how many ice cream cones will you see on the quad?

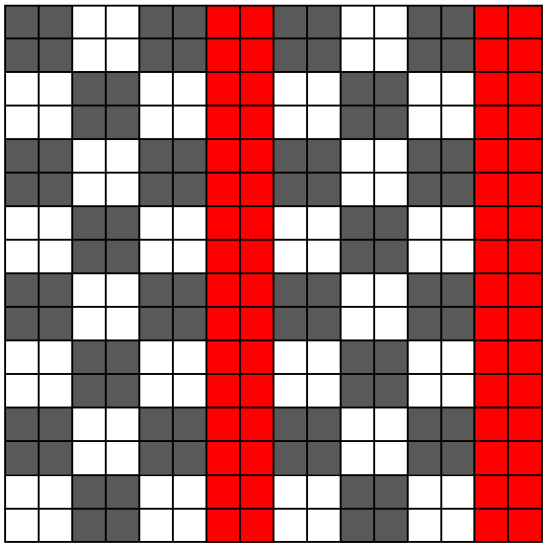
Problem 6.B. Consider the sample located at the center of pixel (256,256) in the image, its screen-space coordinate is (256.5, 256.5). What is the value of the texture coordinate at this location?

Problem 6.C. Given the geometry and texture coordinates in the scene, what is the size of a pixel in screen space when projected into texture space?

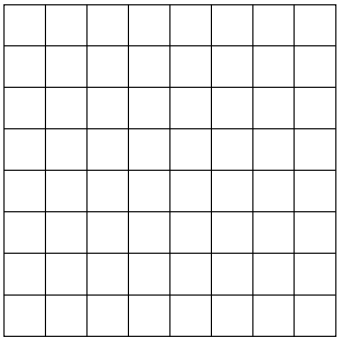
Problem 7. (Extra Practice) Sampling, Prefiltering, and Texturing

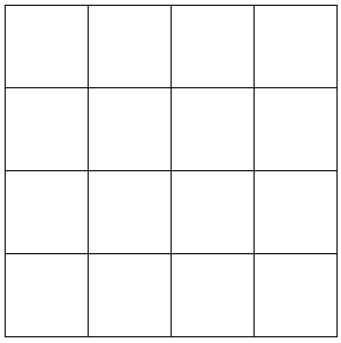
THIS PROBLEM WAS INCLUDED FOR PRACTICE AND IS NOT CONSIDERED PART OF THE PARTICIPATION GRADE.

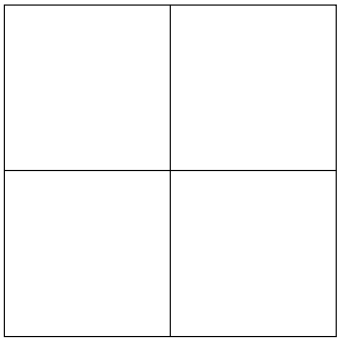
Consider a 1024×1024 checkerboard texture where each check is a 2×2 square of similarly colored pixels, and every 4th check in the horizontal direction is red, as shown below. (not all 1024×1024 pixels are shown)



A. In the diagrams below draw levels 1, 2 and 3 of the mipmap resulting from this texture. Note that mipmap level 0 is the original texture. You can use 'RG' to denote reddish gray, and 'G' to denote gray if you wish.



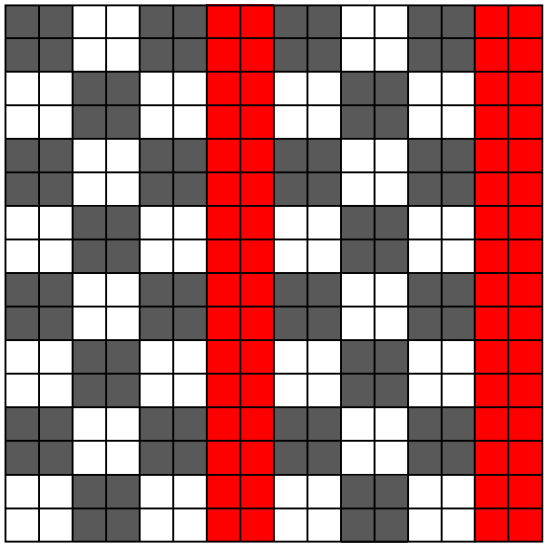




B. Consider a triangle with vertex screen positions (0,0), (0.5,0), and (0.5,0.5) rendered onto a 1024×1024 screen. The triangle has texture coordinates (0,0), (1,0), and (1,1) corresponding to these vertices. Assuming that we render the triangle without mipmapping (so, only accessing level 0) and bilinear filtering, what does the image look like, and why (describe the spacing of the texture samples in screen space)?

C. Now consider the case where triangle vertices are given screen positions (0,0), (1/8,0), (1/8,1/8). Again, like in A, no mipmapping is used, but bilinear filtering is used. Describe the appearance of the triangle now (describe the pattern you see, and what colors are seen). **In your answer comment on whether aliasing is occurring and why?**

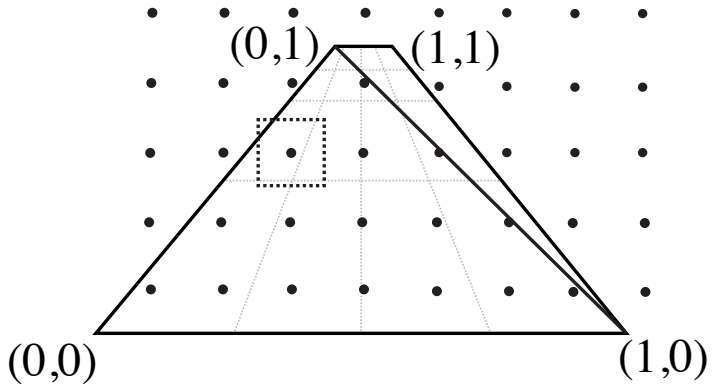
D. We've copied an image of the original texture below for convenience. Assuming the triangle vertex positions are the same as in part C above, draw the region of texture space corresponding to the part of the surface that is visible in pixel (0,0) on screen. (Just draw a region on the figure.)



E. When computing the color for pixel (0,0) for the triangle in parts C and D, imagine your code calls a function `sampletexture(mytexmap, uv)`. Assuming that **THERE IS STILL NO MIPMAPPING ENABLED** describe an implementation of `sampletexture()` that avoids artifacts due to undersampling. There is only one call to `sampletexture()` used to determine the color of a sample, and there is exactly one sample per pixel.

F. Now imagine the conditions are the same as in part E, except now mipmapping is enabled in the implementation of `sampletexture()`. Describe how mipmapping allows for exactly the same solution as in the previous problem, but at significantly lower cost.

G. Now consider using the texture map on a ground plane, the camera looking toward the horizon. Consider shading the highlighted pixel. Describe why in this case, a single mapmap lookup cannot be used to determined the properly prefiltered answer.



H. Consider a system WITHOUT MIPMAPPING, but with a rasterizer that is configured to super-sample coverage N times per pixel (N samples per pixel). The color for each of these samples is determined by a `sampletexture()` lookup as before. ASSUME THAT THE IMPLEMENTATION OF `sampletexture()` only performs a single bilinearly interpolated texture lookup. Describe why this scheme, as N increases, will also accurately approximate the result given in Part E.