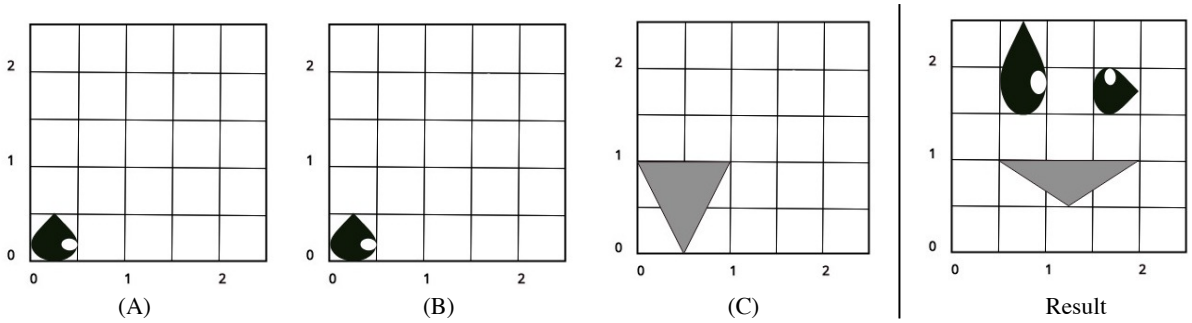
Stanford CS248: Interactive Computer Graphics
Exercise 1

Problem 1. You are working in a silly face factory, and you've been given three parts for making the silly face depicted below. Describe the sequence of transforms used on parts A, B, and C to create the silly face shown at right. (Example transforms: translate, rotate, shear, scale, reflect about an axis.) Rotations are performed in a counter-clockwise fashion about the origin. There are multiple correct answers. **Be careful, rotations and reflections can sometimes look similar! Also note that each grid cell in the figures is 0.5 units, not 1.0.**



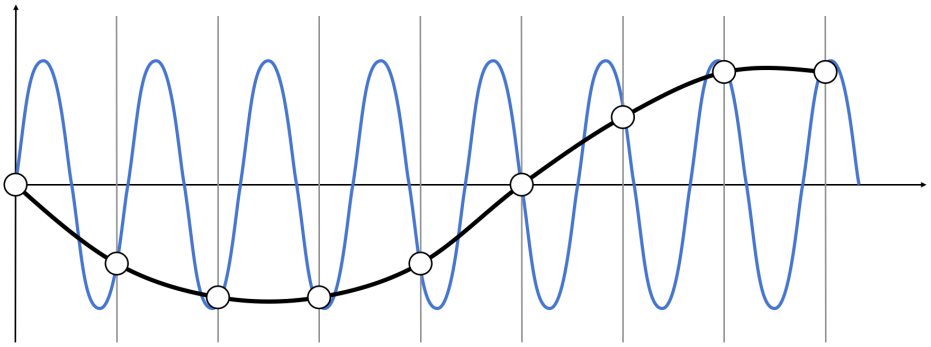
(A) scales 2.0 along y, followed by translation of +0.5 in x, +1.5 in y
(B) rotates +90 deg, reflect along y, then translate +2 in x, +1.5 in y
(C) scale 0.5 along y and 1.5 along x, then translate (+0.5, +0.5)

Problem 2. Assume you are rendering an image to a screen that is 1000 pixels wide and 500 pixels tall. Your viewpoint is set so that all points contained in the box with bottom-left corner coordinate (1,1) and top-right corner coordinate (6,11) are visible on screen. *Assume the bottom-left corner of the box maps to the bottom-left corner of the screen.* In what pixel does a point at coordinate (2,2) appear on screen? Assume pixel (0,0) is the bottom left of the screen and pixel (999, 499) is the top-right. Please show your work.

0) If screen's (0, 0) is at top-left instead of bottom-left, reflect on x-axis.
1) Translate by (-1, -1) [moves box bottom-left to normalized coordinates origin]
2) Scale by (1000/5, 500/10) [Size of screen/size of box]

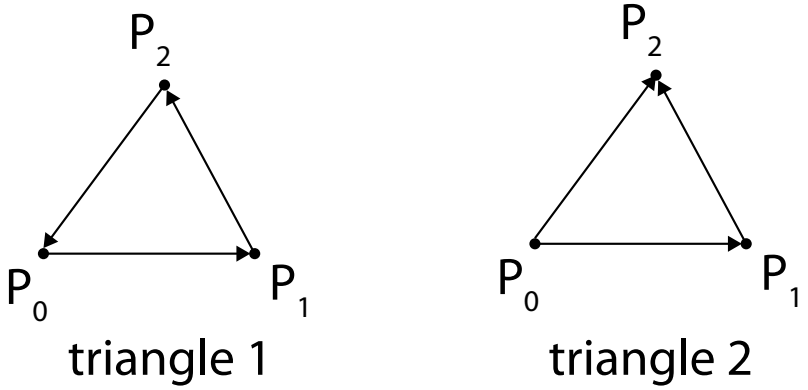
(2, 2) appears as (200, 50) on screen

Problem 3. 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 sampled at a low rate, describe why we call the artifacts created by reconstructing an undersampled signal *aliasing*. (In other words what can we not tell the difference between?)



it's called aliasing since the reconstructed signal looks like another signal and not like the original

Problem 4. 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?



The test function from class doesn't work since not all edges have the same winding. When a point has "L < 0" for (P2 - P0) in triangle 2, it is actually outside the triangle and not inside.

If the triangles now have clockwise winding, change the condition for "inside" to "L > 0" instead of "L < 0" for all edges.