COMPUTER GRAPHICS - MIDTERM 1

Instructions:

- Answer 4 of the 5 questions below. The total points for each question is specified.
- Answer only one of the questions 3,4.
- **Write your name and email below.**

(1) **[27 points]**

(a) What would be the shading pattern of the moon, if its face were almost completely smooth? For instance, what if the moon were a perfect sphere of concrete? Assume only diffuse lighting effects, and that the sun is the only source of light. Of course, in reality, the moon is not smooth. Explain how the roughness of the moon's surface changes your answer.

(b) You are given the endpoints $(x_1, y_1), (x_2, y_2)$of a segment $s$in the xy-plane, where x_1, y_1, x_2, y_2 are integers between 1 and 1000. Suggest an algorithm that for each integer value $x' = 1, 2, 3...1000$, prints the value of the heighest point (x', y), where y is an integer and is below s. Your algorithm should be as efficient as possible.

(c) You are given the vertices of two triangles **B** (Blue) and **R** (Red) in 3D. Assume that they cross each other. Suggest an algorithm that does **not** use harware acceleration for rendering both of them using orthographic projection. Only the parts of R not occluded by B and the parts of B not occluded by R should be rendered. How would your answer change if the use of a hardware Z-buffer is allowed ?

(2) **[27 points]** Consider the scenario described in HW2. In generating some of the illumination (shading) effects, you had to compute the cosine (or dot-product) of certain values. Assume that the plate is in a fixed arbitrary pose (i.e. fixed orientation) such that the normal to the plate creates an angle θ with the z-axis.

(a) How many such cosine or dot-product operations are needed if the source of light is at infinity? Give only an order of magnitude: Once? small number (less than 10) ? Once per pixel? Once per raw etc. Justify your answer.

(b) How many such operations are needed if the source of light is close to the viewpoint ?

(c) **Bonous** Suggest a faster algorithm for shading lots of small triangles of similar but non-identical orientation. You may ignore the specular component of the Phong intepolation model. Explain how you would implement this model using the smallest number of multiplication/division operations.

(3) **[28 points]** This question is similar to HW1, but this time, rather than mapping the picture onto a plate, we will map it onto a cylinder (drawing-on whiteboard). Imagine that you have a picture printed on a piece of paper, which is then stuck to a cylinder whose axis is the line $\{(x, y.z) \mid y = 0\,;\ z = -100\}$, and with radius=30. You can think of this construction as a paper label on a soup can. Suggest how we might render this picture on the display screen, and how you would describe the effect of the cylinder rotating around its axis. Use perspective transformation. You do not have to specify all the parameters exactly, but give an overview of the algorithm and the main transformations mapping pixels in the image printed on the paper to the proper pixels on the display screen. Hint: you can consider each pixel in the paper label as a small polygon of uniform color.

(4) **[28 points]** Consider the rotating plate scenario described in HW1. Recall that every image pixel is mapped into a single display pixel, but that this transformation can be many-to-one. Let θ be the angle between the z-axis and the plate. Assume that for $\theta = 0$, there is a one-to-one correspondence between image pixels and display pixels, and in particular, their number is the same. Suggest an algorithm, as efficient as possible (but without using hardware acceleration) for the case

(a) θ is between 0 and 5 degrees - so quite small.

(b) θ is between 80 and 85 degrees, so many image pixels are mapped to each display pixel, and a large portion of the display is black. Assume that the color of each display pixel can be decided by a single image pixel mapped to it, and that you can set the dispay pixel's color to be any of them.

(5) **[28 points]** Consider a se S of triangles in 3D, each one with its own color. Assume that S is stored in an oct-tree, such that each triangle is stored in exactly one leaf region of the tree. Given a view point $q$suggest an efficient way to use the tree and the painter's algorithm to render the triangles as seen from q.Explain the correctness of your algorithm. (you might find it easier to give the description first in 2D, and then generalize to 3D).