Homework 1

Due: 1:15 pm, Monday, September 12

Note:
1. While discussion on the course material is encouraged, you are required to work on the homework independently.
2. To get the credit, sufficient and appropriate intermediate steps are required in your answers or proofs.
3. Turn in the homework on time. Late homework will not be accepted.
4. Both undergraduate and graduate students must answer the all the questions.

1. (20pts) In the class, we generalize the RGB color system to RGB α color system.
   1) Explain the role of α in the RGB α color system.
   2) Given two colors A and B, can we claim \((A \text{ over } B) = (B \text{ over } A)\)? If yes, give your proof. If no, give a counter example.
   3) Now we want to composite three colors \(A, B,\) and \(C\). There are two ways to calculate \((A \text{ over } B \text{ over } C)\):
      i. \(((A \text{ over } B) \text{ over } C)\), which first composites \(A\) and \(B\);
      ii. \((A \text{ over } (B \text{ over } C))\), which first composites \(B\) and \(C\).

Do they always produce the same composite color? If yes, give your proof. If no, give a counter example.

Solution:

1) α is an extra channel introduced to RGB color system. It represents transparency of the pixel it represents. When \(α = 0\), it’s totally transparent, and \(α = 1\) means it’s opaque.

\[
\text{RGB value of } (r, g, b) \Rightarrow (rα, gα, bα, α)
\]

2) \((A \text{ over } B) \neq (B \text{ over } A)\)

   Counter example: suppose \(A (r_A, g_A, b_A, α_A), B (r_B, g_B, b_B, α_B)\), where \(α_A = α_B = 1\).

   Then \((A \text{ over } B) = α_A A + (1 - α_A) α_B B = A\), and \((B \text{ over } A) = α_B B + (1 - α_B) α_A A = B\), as long as \(A \neq B\), \((A \text{ over } B) \neq (B \text{ over } A)\).

3) Yes.

Proof:

\[
(A \text{ over } B) = α_A A + (1 - α_A) B, \quad α_{AB} = α_A + (1 - α_A) \cdot α_B
\]

\[
((A \text{ over } B) \text{ over } C) = (α_A A + (1 - α_A) B) + (1 - α_{AB}) \cdot α_C C
\]

\[
= (α_A A + (1 - α_A) B) + (1 - (α_A + (1 - α_A) \cdot α_B)) \cdot α_C C
\]

\[
= α_A A + (1 - α_A) B + (1 - α_A) \cdot (1 - α_B) \cdot α_C C
\]

\[
(B \text{ over } C) = α_B B + (1 - α_B) \cdot α_C C, \quad α_{BC} = α_B + (1 - α_B) \cdot α_C
\]

\[
(A \text{ over } (B \text{ over } C)) = α_A A + (1 - α_A) (α_B B + (1 - α_B) \cdot α_C C)
\]

\[
= α_A A + (1 - α_A) B + (1 - α_A) \cdot (1 - α_B) \cdot α_C C
\]

So

\[
((A \text{ over } B) \text{ over } C) = (A \text{ over } (B \text{ over } C))
\]
2. (15pts) In computer graphics, objects such as spheres are usually approximated by simpler objects constructed from flat polygons. Using lines of longitude and latitude, define a set of simple polygons that approximate a sphere centered at the origin. Can you use only quadrilaterals or only triangles?

**Solution:**
We can use only triangles.
If we use lines of constant longitude and lines of constant latitude, their intersections define a set of quadrilaterals. If we draw diagonals for each quadrilateral, we get a set of triangles. However, at the poles all the curves of constant longitude meet and we get triangles rather than quadrilaterals.

3. (15pts) Consider the clipping of a line segment in two dimensions against a rectangular clipping window. Show that you require only the endpoints of the line segment to determine whether the line segment is not clipped, is partially visible, or is clipped out completely.

**Solution:**
Suppose the line segment is described by its endpoints \( P_1 = (x_1, y_1) \) and \( P_2 = (x_2, y_2) \), where \( x_1 \leq x_2 \), and the window is defined by its upper left corner \( (x_{\text{min}}, y_{\text{min}}) \) and the lower right corner \( (x_{\text{max}}, y_{\text{max}}) \). There are several ways to check if the line segment is (a) not clipped, (b) partially visible, or (c) clipped out completely as shown in the figures below.

- The line segment is not clipped if \( x_{\text{min}} \leq x_1, x_2 \leq x_{\text{max}} \) and \( y_{\text{min}} \leq y_1, y_2 \leq y_{\text{max}} \)
- The line segment is partially visible if one of the cases is true:
  - \( x_{\text{min}} \leq x_1 \leq x_{\text{max}}, y_{\text{min}} \leq y_1 \leq y_{\text{max}}, \) and \( x_2 > x_{\text{max}} \)
  - \( x_{\text{min}} \leq x_1 \leq x_{\text{max}}, y_{\text{min}} \leq y_1 \leq y_{\text{max}}, \) and \( y_2 > y_{\text{max}} \)
  - \( x_{\text{min}} \leq x_1 \leq x_{\text{max}}, y_{\text{min}} \leq y_1 \leq y_{\text{max}}, \) and \( y_2 < y_{\text{min}} \)
  - \( x_{\text{min}} \leq y_2 \leq x_{\text{max}}, y_{\text{min}} \leq y_2 \leq y_{\text{max}}, \) and \( x_1 < x_{\text{min}} \)
  - \( x_{\text{min}} \leq y_2 \leq x_{\text{max}}, y_{\text{min}} \leq y_2 \leq y_{\text{max}}, \) and \( y_1 > y_{\text{max}} \)
  - \( x_{\text{min}} \leq y_2 \leq x_{\text{max}}, y_{\text{min}} \leq y_2 \leq y_{\text{max}}, \) and \( y_1 < y_{\text{min}} \)
The line segment is clipped out if one of the case is true:

- \( x_2 < x_{\text{min}} \),
- \( x_{\text{max}} < x_1 \),
- \( \max(y_1, y_2) < y_{\text{min}} \), or
- \( y_{\text{max}} < \min(y_1, y_2) \)

4. (15pts) Consider the perspective views of the cube shown in Figure 1.39 (Angle’s book). The one on the left is called a one-point perspective because parallel lines in one direction of the cube—along the sides of the top—converge to a vanishing point in the image. In contrast, the image on the right is a two-point perspective. Explain why one is a two-point perspective and the other a one-point perspective. Hints: you need to consider the particular relationship between the viewer (a camera) and the cube.

![Perspective views of a cube.](image)

**Solution:**
In a one–point perspective, two faces of the cube is parallel to the image plane and perpendicular to the optic axis of the camera, while in a two–point perspective only the edges of the cube in one direction are parallel to the image plane. In the general case of a three–point perspective there are three vanishing points and none of the edges of the cube are parallel to the image plane.

5. (35pts) (Note: You need to write and compile some codes for this problem.)
Develop a program to display a 2D polygon formed by five vertices \{(0,0.9), (-0.9,0), (-0.5,-0.9), (0.5,-0.9), (0.9,0)\}.

In turning in your homework, you need to submit a zipped file through dropbox including

- your code,
- at least two sample results, each of which has a different color, and
- a graph showing the triangulation scheme you used

**Extra credit (10pts): change the background color and show the polygon with a mixture of colors**

You are provided a simple C++ package with a sample code of generating a triangle, which is from “OpenGL Programming Guide, the Official Guide to Learning OpenGL, Version 4.3, the 8th Ed,” by D. Shreiner, G. Sellers, J. Kessnich, and B. Licea-Kane. This package is named “hw1_sample.zip.” You
need to run “unzip hw1_sample.zip” to uncompress it. The main function you are
going to work on is the file “triangle.cpp.” You can modify the sample code to
draw the polygon. This package has been tested in the Linux environment. To
compile it, simply type

    make

and the resulting executable is ‘runnable’ You can type ‘./runnable’ to run it.