Instructions

Answer each of the following questions. Unless otherwise specified, you need not show your work when calculations are required. In general, brief answers are preferred over lengthy expositions. Do not copy material from a textbook or a web site. Do not work with others to produce your answers. Submit your written answers by Tuesday, March 29, 2016.

1. How long would it take to load an 800 by 600 frame buffer with 16 bits per pixel if 105 bits can be transferred per second? How long would it take to load a 32-bit-per-pixel frame buffer with a resolution of 1680 by 1050 using this same transfer rate?

2. Suppose we have a video monitor with a display area that measures 12 inches across and 9.6 inches high. If the resolution is 1280 by 1024 and the aspect ratio is 1, what is the diameter of each screen point?

3. Suppose you want to draw a square using OpenGL. The coordinates of the square are (1,1), (1,2), (2,1), and (2,2). There are at least four different glBegin function invocations and the corresponding glVertex* function calls that will achieve the drawing of this square. One of them is this:

   ```
   glBegin(GL_QUADS);
   glVertex2i(1,1);
   glVertex2i(1,2);
   glVertex2i(2,2);
   glVertex2i(2,1);
   glEnd();
   ```

   Give three other code sequences (from glBegin through glEnd) that will draw the square. Which of these will draw a filled square?

4. Briefly explain the relationship between the windowing system provided by something like Microsoft Windows or Apple OS X, the drawing produced by calls to the OpenGL library, and GLUT.

5. Explain the purpose of glutDisplayFunc and the function provided as its argument. Why is such a callback function required? In general, what is the purpose of a callback function (as provided by GLUT)?

6. There are two basic OpenGL approaches to specifying the vertices associated with drawing objects: (1) explicitly giving their values as arguments to functions like glVertex2i and (2) specifying a collection of vertices in a vertex array. Briefly describe the difference between these approaches. Then indicate a potential benefit of using vertex arrays.

7. What happens when OpenGL draws two filled objects (like triangles) so that one overlaps the other? Explain how this behavior can be controlled.

8. Bresenham’s algorithm for drawing a line with a slope \( m \) having an absolute value less than 1 (that is, \(|m| < 1.0|\) is given in the text on pages 143-4 and is also displayed below. Assume we
want to modify the algorithm so it performs line stippling in the same way as glLineStipple. Modify the function so it accepts two additional parameters (repeatFactor and pattern, the same as those used with glLineStipple), and so it will display the appropriately-stippled line. You do not need to actually compile the code; just provide a display of the modified code.

```c
#include <stdlib.h>
#include <math.h>

/* Bresenham line-drawing procedure for |m| < 1.0. */
void lineBres (int x0, int y0, int xEnd, int yEnd)
{
    int dx = fabs (xEnd - x0), dy = fabs(yEnd - y0);
    int p = 2 * dy - dx;
    int twoDy = 2 * dy, twoDyMinusDx = 2 * (dy - dx);
    int x, y;

    /* Determine which endpoint to use as start position. */
    if (x0 > xEnd) {
        x = xEnd;
        y = yEnd;
        xEnd = x0;
    }
    else {
        x = x0;
        y = y0;
    }
    setPixel (x, y);

    while (x < xEnd) {
        x++;
        if (p < 0)
            p += twoDy;
        else {
            y++;
            p += twoDyMinusDx;
        }
        setPixel (x, y);
    }
}
```

9. What is meant by the term *anti-aliasing*? Briefly describe a few techniques that could be used to achieve this.
10. What are the two major techniques used with OpenGL to specify the color of objects?
11. What are the coordinates of a line from (1,1) to (3,3) after it has been rotated 45 degrees clockwise? What would the coordinates be if it had been rotated 90 degrees counterclockwise instead? In both cases, assume the rotation is about the origin.
12. Determine the coordinates of the line in the previous question, but assume the rotations are about the point (2,2).
13. Translation of an object by \((\Delta x, \Delta y)\) is relatively simple: add \(\Delta x\) to the x component of each vertex, and add \(\Delta y\) to the y component of each vertex. In general, this isn’t the way OpenGL (and other graphics libraries) usually implement translation. What would you usually do to implement translation? Consider only two-dimensional drawings.

14. Explain the essential differences between a clipping window and a viewport.

15. Consider the three clipping algorithms (for straight lines and a rectangular clipping window) Cohen-Sutherland, Liang-Barsky, and Nicholl-Lee-Nicholl. These algorithms are all similar in one respect, and they also differ from each other in one significant way. What is the similarity, and what is the difference.

16. Consider the rectangular clipping window with \((1,1)\) as its lower left corner and \((3,3)\) as its upper right corner. What are the coordinates of the line from \((0.5,1.5)\) to \((2,4)\) when it is clipped? What are the coordinates of the line from \((1.5,3.5)\) to \((4,0)\) when it is clipped?

17. The Sutherland-Hodgman polygon-clipping algorithm works in stages. What is the difference between the stages? What is a potentially significant advantage of this algorithm over one that does not work in stages?

18. What makes line-clipping algorithms (like Cohen-Sutherland), by themselves, generally unsuitable for clipping the boundary lines, and thus the areas, of filled regions?

19. Text clipping can be done using three obvious methods. What are these?

20. Assume we are clipping the components of individual characters. That is, if a character’s boundary intersects with the clipping region boundary, then part of the character will be clipped and not be included in the clipping window. Which character type, outline or bitmapped, is likely to be more difficult to clip?