CS 770/870 Assignment 3

Due: Tuesday Sept. 27th, 2022

Lateness Penalty: Wed: 5%, Thu 10%, Fri 20%, Sat/Sun/Mon 50%, Tue 100%

1. Pixels:

    a) We can see nearby objects easily, but to see distant objects clearly we need a telescope. Also, we need microscopes to see small objects clearly. For some reason, we can't see things in infinite detail. There is a physical limitation in the human visual system that limits the amount of detail we can see. What is it? *KEEP YOUR ANSWER SHORT*: at most two sentences. *Hint: why is this a "Pixels" question?*

    b) Animations consist of many images (frames) shown in rapid succession, at 60 frames per second. Suppose a single frame of an animation contains 1280×1024 pixels, and each pixel takes up three 8-bit bytes (r, g, and b). Suppose we can compress video by a factor of 100:1 (a hundred-to-one). If so, how many seconds of video can be stored in 3 gigabytes (3,000,000,000 bytes)?

2. Points and Vectors: Consider the following pyramid:

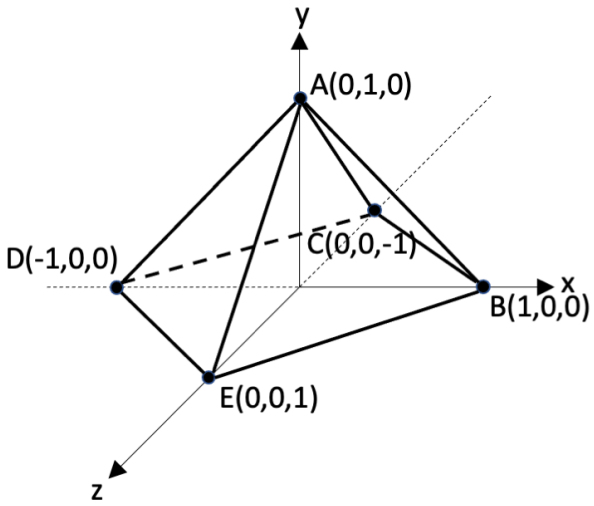

Figure 1: .

    a) The pyramid has eight edges: AB, AC, AD, AE, BC, CD, DE, and EB. Compute the normalized (length=1) direction vectors of these edges.

    b) The pyramid has five faces: ABC, ACD, ADE, AEB, and BCDE. Compute the normalized **outward facing** normal vectors of these faces.

Suggestion: use the `glm::vec3` classes, and a short program (look at the `cross` and `normalize` functions). If you do this, *DO NOT* turn in your program. Just copy its output into the answer file.

3. Clipping: When a polygon P is clipped against a rectangular window, a new polygon CP is obtained, which may have a more or fewer vertices than P.

   For each of the following questions, explain your reasoning with a picture.

   a) If P is a triangle, how many vertices will CP have, at most?

   b) If P is a quadrilateral, how many vertices will CP have, at most? (P could be non-convex!)

   c) If P is convex and has `n` > 4 vertices, how many will CP have, at most (as a function of `n`)?

   d) If P is convex and has `n` > 4 vertices, how many will CP have, at LEAST? (Assume P is not trivially clipped, *i.e.*, part of P is wholly inside the window and part is wholly outside).

4. Suppose a raster image is stored in a 2-dimensional array, with one byte per pixel:

   ```
   unsigned char image[n][n];
   ```

   so that `image[y][z]` stores a (one-byte) pixel, at column `x` and row `y` (y=0 is the bottom row). Write a short piece of code that rotates the image counter-clockwise by 90 degrees, *in situ* (without creating a new image). Here is an example:

```
   Original:            Rotated:

X X X X X X X          X - - - - - X
X - - - - - -          X - - - - - X
X - - - - - -          X - - - - - X
X X X X - - -          X - - X - - X
X - - - - - -          X - - X - - X
X - - - - - -          X - - X - - X
X X X X X X X          X X X X X X X
```

5. Transformations:

   a) Write a function `draw_box()`, which draws a 1×1 box, with bottom-left corner on the origin. The box should be gray (rgb = .5 .5 .5), and should be outlined in white.

   b) Write a function `draw_robot()` that will draw this figure:

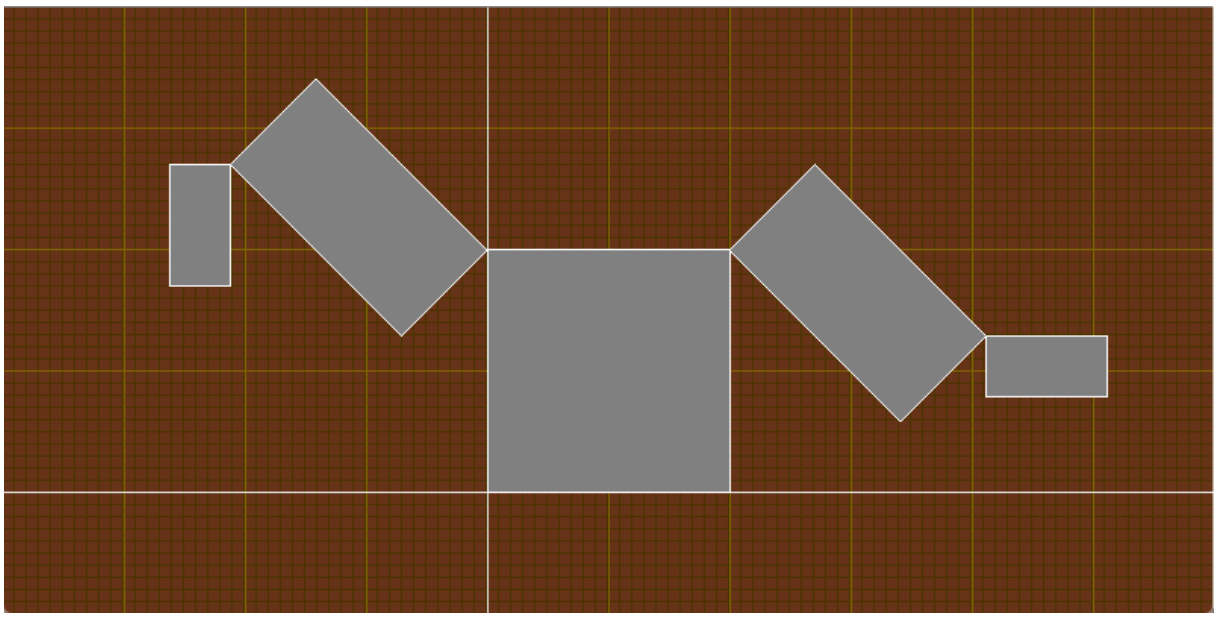

Figure 2: .

The boxes are 1×2, 0.5×1, or 2×2, and all the angles are 45 degrees.

To do this, create a stack of transformation matrices. Then implement these operations:

- init: create the stack (`std::stack<glm::mat4x4> modeling_stack`).
- push: get the `.top()` matrix, and push it, thus saving a copy.
- pop: discard the top matrix (restore to a previous state).
- translate: multiply the top matrix on the right by a translation matrix.
- rotate: multiply the top matrix by a rotation matrix.
- scale: multiply the top matrix by a scaling matrix.
- get top: get the top matrix, and use it draw a transformed box.

Submitting Your Work

For questions 1, 2, and 3, prepare a *PDF* file called `asn03-q123.pdf`, with your answers in it.

For question 4, edit the file `turn_raster.cpp`, and implement the `turn_image()` function.

For question 5, edit the file `robot.cpp`, and implement the `draw_box()` and `draw_robot()` functions.