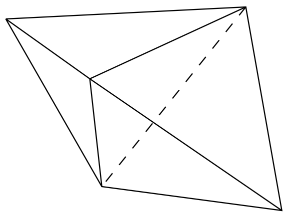
CS450/550 — Program 3 — Due February 13, 1996

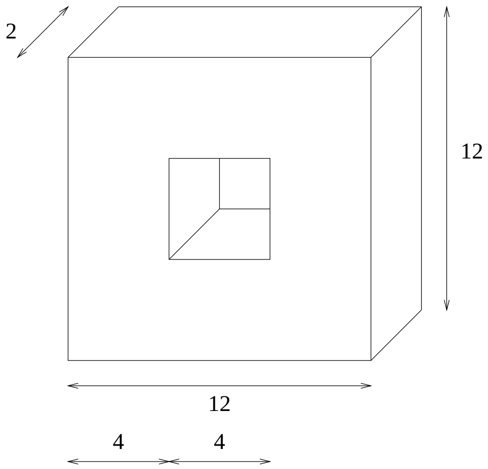
In this assignment, you will gain experience with creating `tmesh` objects and with using the view transformation to move the camera through a scene.

The first thing you must do is create two files containing `tmesh_t` objects. See page 377 of the textbook for an explanation of the format of a `tmesh_t` file. The first file (give it the name "hexahedron.data") should contain a "hexahedron"—a six-sided object where each face is an equilateral triangle.



You should orient this so that it is centered at the origin with the central plane lying in the X-Z plane. Each edge should have length 2.

The second file (name it "window.data") should contain a "window" object. This is a square slab with a square hole in it:



This object should also be centered at the origin with the Z axis going straight through the center of the hole. Both objects should have RGB values of (1 1 1).

Your program should read in both of these `tmesh` objects and rotate and translate them as follows. The "window" should be centered at $(0, 0, 5)$ and rotated so that the line $y = 0, z = 5$ passes through the center of the hole. The hexahedron should be centered at $(-10, 0, 5)$. After rotating and translating, you should place these two objects into a `scene3_t`.

The initial image produced by your program should look something like the image in Figure 1. Note that the interior faces of the "hole" are visible, because the "frontface" test believes they are visible from the camera. This is ok for this assignment.

Your program will then generate a sequence of 10 views in which the camera moves along a circular arc from its starting point (at $(0, 0, -10)$) to the point $(15, 0, 5)$. During this move, the camera should always be facing the point $(0, 0, 5)$ (the center of the hole), as shown in Figure 2.

Figure 1: Initial Image

The program should then generate a sequence of 20 views in which the camera moves along the line $y = 0, z = 5$ so that it passes straight through the hole in the window and heads directly toward the hexahedron as shown in Figures 3 and 4.

All files needed for this project should be copied from `~tgd/public_html/p3.tar` on the engineering lab machines (or from the class web page). The tar file will create a `p3` directory that will contain two main programs. One program is called "spin". To compile it, issue the command `make spin`. To execute the program, type "spin *tmesh*", where *tmesh* is the name of a file containing a `tmesh` definition. Spin will then prompt for 3 angles of rotation and 3 values for translation and then it will enter a loop in which it repeatedly applies these rotations and translations to the object. The loop is controlled by mouse clicks. Click the left button to start spinning and any button to interrupt spinning. When spinning is interrupted, the middle button will "single step" the object; the right button will exit. This is very helpful for checking whether your `tmesh` has been properly entered.

The other program is a skeletal program named "p4". You should modify "p4.c" to read in and position the two `tmesh` objects and then move the camera. To move the camera, you should use the function `scene3_t::viewtrans`. For the first 10 iterations through the animation loop, you should move the camera slightly in the positive X and Z directions so that it moves along the desired circular path. For the remaining iterations, you should move the camera in the Z direction, so that it will go straight through the hole. Remember that the coordinate system for the view transformation is specified with respect to the *current* coordinates (i.e., which are not the original coordinates except for the very first iteration). Read the code in the file `Shape3d/scene3.c` to understand what arguments `viewtrans` is expecting.

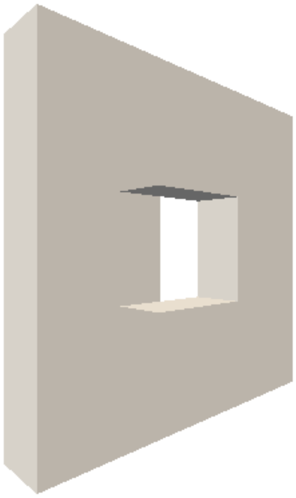


Figure 2: A view during the first phase of the animation.

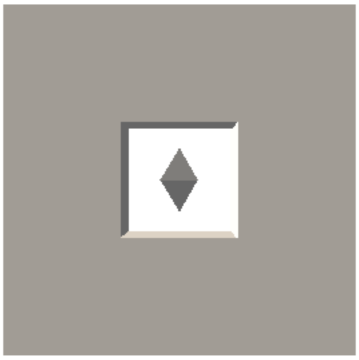


Figure 3: A view from the second phase of the animation.

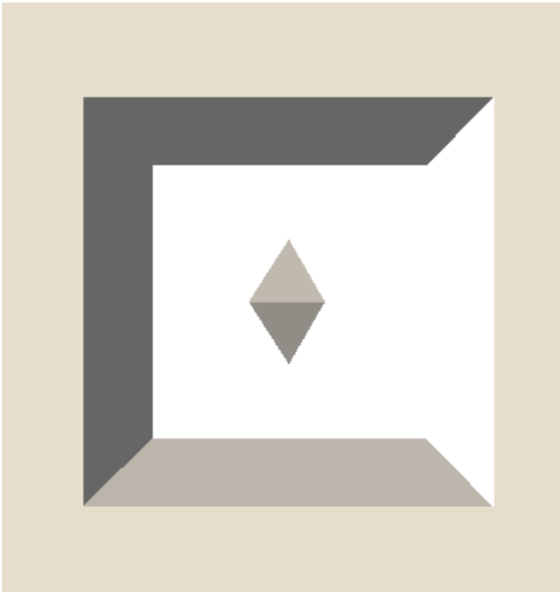


Figure 4: A few steps later.

Students in CS550

You will notice that the code for `scene3_t::viewtrans` is very inefficient: it calls the `viewtrans` member function for each object in the scene. Every one of these calls recomputes the rotation/translation matrix, which is very wasteful (e.g., look at `Shape3d/tmesh.c`).

What we would like to do is compute the matrix once and then apply it to all of the objects in the scene. Similarly, for repeated camera motions, it would be nice to compute the matrix once, and then repeatedly apply it to the whole scene, rather than recomputing it for each move.

For this assignment, you should do the following:

- Define a new member function `scene3_t::viewmat` that takes the same arguments as `viewtrans` but returns a pointer to a new `mat43_t`. This is a transformation matrix with 4 rows but only 3 columns (since the last column is always the same).
- Define a new member function `scene3_t::transform` that takes a `mat43_t` and applies it to each object in the scene and to the lighting sources in the scene.
- Define new `transform` member functions for `prim_t` and `tmesh_t` that take a `mat43_t`. The `prim_t::transform` function should be declared to be `virtual`, but otherwise it should do nothing. The `tmesh_t::transform` function should apply the transformation matrix to each of its faces.

With these changes to the code, you should then write the main program to compute two different transformation matrices: A and B. Matrix A should have the effect of moving the camera one-tenth of the way from $(0, 0, -10)$ to $(15, 0, 5)$ along a circular arc. You can then animate this move by applying A to the scene for 10 steps. Matrix B should have the effect of moving the camera forward (in the +Z direction) by 1 unit. You can complete the animation by applying this matrix the the scene 20 times.

What to turn in:

For CS450, you should be able to email a `shar` file containing your source file and a `README` file that tells how to run it. For CS550, you will need to email a uuencoded tar file that includes copies of all files that you modified. It should include a `README` file also.