Programming Assignment #2
CS410 : Introduction to Computer Graphics
Fall 2015

Silhouettes
Due Thursday, Oct. 15th.

Motivation

In the first assignment, you transformed 3D polygonal models so that you could combine them into a single scene. For the second assignment, you will take the first step in rendering that scene. More precisely, you write a program that reads in a camera model and one or more objects, and produces a silhouette image in which output pixels are white if the ray from the focal point through the pixel intersects at least one polygon in the scene, and black if they ray does not intersect any polygon.

Task

Your program will take in three or more command line arguments. The first argument is a file containing the camera model. The last argument is the name of the image your program will write. In between are the names of ply model files (as written out by PA1). Note that there must be at least one object file, but there could be many.

Your program will implement the camera model by throwing a ray from the focal point through each camera pixel, and determining whether that ray intersects any of the polygons in any of the models. If it does, the corresponding pixel should be set to white (255, 255, 255); otherwise, the pixel should be black (0, 0, 0). Your program will then write out the resulting image as an ASCII PPM color image (P3 format, see below) to the filename in the last argument.

Data Formats

The camera model file is an ASCII file with 5 lines. The first line contains three floating point numbers, which represent the X, Y and Z coordinates of the camera's focal point. The second line contains three floating point numbers representing the coordinates of the look-at point. The third line contains three floating point numbers representing the VUP vector. The fourth line contains a single floating point number, which is the focal length of the camera. The fifth line contains four integers, which are the minimum u coordinate, the minimum v coordinate, the maximum u coordinate, and the maximum v coordinate. For example, a camera located at the origin, with a look-at point at (0,0,-100), a VUP

vector of (0,1,0), a focal length of 340 and image size of 256x256 might have the following file:

```
0.0  0 0.0
0.0  0.0 100.0
0 0 1.0
340.0
-128 -128 127 127
```

In case of a format error or physically impossible camera (e.g. a focal length of 0, or a maximum u value smaller than the minimum u value), your program should print out an error message and return without creating an image file.

Images should be written as legal ASCII PPM files. Although some variations are permissible (you can google the PPM standard), I recommend the following. The first line contains the characters P3 and nothing else. The next line contains the image width, the image height, and the number 255 (the maximum possible pixel value), all integers. Pixel values begin on the next line, and contain 3 values per pixel (a red value, a green value, and a blue value, in that order). Note that in this assignment, the RGB values in a pixel will all be same, and will always be 255 or 0. In future assignments, however, you will be generating color images, and it is easier to keep the format the same. Since the total number of pixels in an image is width times height, the number of values in the file (after the two header lines) must be 3 times width times height. To make images "readable" by humans (when they are small), I recommend putting a newline at the end of each row. The image generated for the camera model above might therefore begin with

```
P3
256 256 255
0 0 0 …
```

The models are in PLY format, as in programming assignment #1.

Hints

All the code written as part of PA2 will be directly used in PA3 and PA4, so design, document and debug it carefully. If PA2 does not work well, you will probably never recover.

Submission/Grading

Make a tar file that includes your source files, a makefile if appropriate, and a README.txt file that explicitly tells us (1) how to compile your program and (2) how to execute it. Submit this tar file via the Checkin script on the class web site. The GTA will unpack your tar file, compile your program, and then test it.. Note that your tar file should not contain executable or compiled files, just source files.

Reminder

There is no "late period". The program is due when it is due. All work you submit must be your own. You may not copy code from colleagues or the web or anywhere else. Cheating will not be tolerated, and will be handled in accordance with university and department policy.