Introduction to Computer Graphics

CS5600 Spring 2012
Project 1 - 2D Scene Rasterization
Due: 11:59PM Wednesday, February 1st

Abstract

The goal of this assignment is to produce a command-line style program capable of rendering scenes composed of 2D geometry to an in-memory raster and then save that raster to an image file. The input data to the program will be a scene file and referenced RAW triangle files. The output is an image saved to file in PPM and BMP formats. You may use C, C++, Objective-C, or C# to write the program, as described in the Handin section of this document.

This class necessarily involves the recreation of the wheel. To that end, we are writing code for as much of the process as is prudent. We will not be writing the foundation of a programming language, a basic math library, or a file I/O library. However, any other functionality you need you need to write yourself. Essentially, if it is functionality provided by the C Standard Library (see C99 Wikipedia article) in the language you are targeting, feel free to use it. Otherwise, write it yourself. For example you must write your own matrix and vector library, as well as file output routines for this assignment. Do not use more advanced libraries provided by the language. Do not use code or libraries from the internet. Certainly don't use code written by friends or fellow students. Doing this may be annoying to start, but there is a lot of opportunity to learn programming concepts here.

Feel free to refer to the videos from class for programming implementation ideas. Don't copy the code from those videos. Direct use of code from the videos, other students, or the internet will constitute academic misconduct and will result in an E in the course and disciplinary action by the CS college. That being said, basically if you write and understand all the code you turn in, you shouldn't be worried about these policies. Read, code, debug, recode, learn.

Components

- **Scene File Reader:** Include in the same directory as your compiled program a file named "scene.u2d". This file will describe a scene composed of 2D geometry, specifically 2D triangle lists. The file does not contain the geometry data itself, but instead includes references to RAW-formatted files containing the geometry data. You will need to load and parse this file and use it to create the output image. Having a scene object in your program is useful but not required. The file is ASCII formatted and composed of:
  - A "magic number" identifying the file type followed by a newline. The magic number is the 8-bit ASCII characters "U2".
  - An integer width, integer height identifying the raster size used to draw the scene, separated by whitespace, followed by a newline.
  - 4 floating-point numbers giving the minX, minY, maxX, and maxY that should map to fill the output raster. For example, if a raster of size 500, 500 is specified in the previous line, and the numbers -1.0 -1.0 1.0 1.0 are given, a triangle that fills most of the raster, has a point at the top-middle of the raster, and has a flat base at the bottom of the raster, would have coordinates (-0.9, -0.9), (0.9, -0.9), (0.0, 0.9). Note that triangle data after being transformed should fit within these bounds to end up on the raster!

    - An unlimited number of lines specifying geometry and transformations on geometry. A single character plus whitespace identifies what data will follow on a line and ends with a newline. Numbers will be specified as floating-point numbers. Your program should be able to handle blank lines as well.
        - g filename.raw - Identifies a RAW file (described below) from which geometry should be loaded. This will be a list of triangles that should be somehow included in a group in your program. The files will have an implicit origin at (0, 0) on which the triangle data depends.
        - c r1 g1 b1 r2 g2 b2 r3 g3 b3 - Identifies 3 colors that should be applied when rendering the triangles making up the geometry last specified in a “g” line. One color should be applied to each of the 3 points in each triangle. So that means that all the triangles in a group will have the same gradient color pattern.
        - t tx ty - Identifies a translation transformation by tx, ty that should be applied to the geometry last specified in a “g” line.
        - r theta - Identifies a rotation transformation by theta that should be applied to the geometry last specified in a “g” line.
        - s sx sy - Identifies a scale transformation by sx, sy that should be applied to the geometry last specified in a “g” line.
- **RAW File Reader:** ASCII files in the same directory as the scene file containing 3D floating-point triangle coordinates in XYZ order. Each triangle consists of 9 numbers separated by whitespace or newline characters. For this assignment, ignore the Z coordinates. Your program will need to define a class or other structure to store these triangles together in a group so transformations can be applied to all of them.
- **Color, Vector, and Matrix:** Your program should have classes or modules that define an integer ARGB color, a floating-point ARGB color, a 2D vector, and a 3x3 matrix, as well as the needed operations that apply to those constructs (eg. vector add, matrix concatenate, color scale). It may contain a class or module for a triangle and a class or module to store a group of triangles.
- **Raster:** Your program should have a class or module that defines a raster object. The raster should store a set of colors representing pixels with x,y integer coordinates. The raster’s origin can be at the top-left or top-right of the raster, but must be able to save to PPM and BMP files, which have requirements on the ordering of the pixels. The raster should have some way by which 2D triangles with interpolated colors can be drawn on the raster. The raster should be saved to a BMP file and a PPM when the scene has been drawn. The files should be in the same directory as your compiled program and the files should be named “scene.ppm” and “scene.bmp”.

Handin

We are allowing submissions of code using 3 project archive formats and 4 languages. Supported project archives are:

- **Microsoft Visual Studio 2010 Solution:** Create a C, C++, or C# project using the command line tool template.
- **Xcode 4 Project:** Create a project using the Mac Command Line Tool template. Use C, C++ or Objective-C with this setup.
- **Linux Make File:** Use a directory with a makefile and code files inside. Your project should build on a POSIX system with GCC installed by simply changing to the directory and typing “make”.

You should hand in your zipped project using the CADE lab by using the command:

handin cs5600 project1 your_project_zip_file.zip

Examples

scene.u2d

```
U2
500 500
-1.0 -1.0 1.0 1.0
g /Users/Matt/Desktop/triangle.raw
c 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1
s 0.5 1.0
t 0.3 -0.2
```

triangle.raw

```
-0.5 -0.5  0.0
 0.5 -0.5  0.0
 0.0  0.5  0.0
```

image.bmp