CSCI-510 - Final Project

Due: Thursday, December 19, 2013

Late submissions will not be accepted!
Extensions cannot be used on this assignment!

Last updated 2013/11/25 15:07:23

Update history:

```
2013/11/25: initial version
```

---

1. Introduction

In this project, you will test your skills with shader-based OpenGL by creating and rendering a 3D scene. This involves:

- 3D object definition and placement
- Shading
- Camera placement and modification
- Texture mapping

---

2. Programming Environment

Unlike the midterm project and the programming assignments, no programming environment is being provided to you for this assignment. You are free to re-use components from earlier assignments.

---

3. Tasks

You must create a "complex" 3D scene with the following characteristics:

- Multiple objects (three or more), modeled in local coordinates and positioned in the scene using model transformations
- Non-default camera positioning
- At least one light source
- Use of at least one texture
- Shading on all objects which aren't texture-mapped, with at least two different sets of material properties

The scene you render should be your attempt at recreating a piece of artwork that depicts a 3D scene; there must be discernable objects in the scene, and it must have a definite appearance of depth.

You should use your chosen subject as a guide for setting up the scene (objects, camera, materials, lights, textures). You will *not* be graded on artistic merit!

If you need some motivation, you may find this video in which a famous painting by Van Gogh is recreated to be of help. The image was created in the virtual world of Second Life; it is well beyond what you must do for this assignment in its complexity, and it makes use of the world-building capabilities of that environment, but it provides a nice example of the task at hand.

---

4. What to Submit

Submit your solution using `try` on the CS systems using the following command:

```
try  wrc-grd  cg-final  files
```

where *`files`* is the list of file names you are submitting.

For this assignment, you should submit the following:

- **All** of your source files, including your shader files and all C/C++/Java source files. The file containing your `main()` routine must be named `finalMain.`*`x`*, where *`x`* indicates your language of choice (C, C++, or Java).
- All of the texture image files you are using.
- Either an image of the artwork you are reproducing in a file named `artwork.`*`x`* (where *`x`* is a suffix indicating the image file format - see the list below), **or** a file named `artwork.txt` which contains a pointer to an online resource where such an image can be found. The 'try' scripts will accept image files in any of these formats:

  ```
  ase bmp dat gif jpg m3d map
  mtl obj png ppm raw rgb rle
  sgi tga tif tmp txt
  ```

  (One of `gif`, `jpg`, or `png` would be preferable, but any of these will be accepted.)

If you are using the SOIL image library to load your texture image(s), the header file should be included

as either `<SOIL/SOIL.h>` or `<SOIL.h>` on the CS systems. I will use a `header.mak` file that automatically links your code against the SOIL library (see the `header.mak.soil` file in my CG pub directory).

If you are using one of the matrix libraries described in the *Notes* section of the programming assignment 5 writeup, please **do not** submit copies of those files; the 'try' scripts will include them automatically. If you are using a matrix library other than the ones described there, then you must submit those files with your solution.

If you are using any libraries other than OpenGL, GLUT, GLEW or SOIL, you will need to do one of the following:

- **(Preferred option.)** If the library exists on the CS Ubuntu systems in the standard system library directories (e.g., the JPEG, GIF, PNG, or TIFF image library), submit a `header.mak` file with modifications to the `LDLIBS` macro definition to tell the compiler where to find the library. For example, to use the JPEG library, you would add `-ljpeg` to the `LDLIBS` macro (because the JPEG library is in a file named `libjpeg.a` in the system standard library directory).

- If the library exists on the CS Ubuntu systems but is not in the standard system library and/or header file directories, submit a `header.mak` file with modifications to the `INCLUDE`, `LDLIBS`, and `LIBFLAGS` macros, as follows:

  - `INCLUDE`: add `-I`*`dir`*, where *`dir`* is the directory where the header files are located
  - `LDLIBS`: add `-l`*`name`*, where *`name`* is the middle part of the library file name `lib`*`name`*`.a`
  - `LIBFLAGS`: add `-L`*`dir`*, where *`dir`* is the directory where the library file is located

  Please do not use libraries installed in student accounts. If you're using a library that you have installed in your account and you believe it would be useful to have installed for general access, let me know and I'll see if I can make it happen.

- If the library is not available on the CS Ubuntu systems, you must submit the source code for the library (along with any necessary header files) with your solution; it will be compiled and linked along with your code.

Your submission will be compiled and linked to ensure that it is complete. If it does not compile or link, it *will not be accepted*, which means that *you will not have submitted your solution*. (Warning and informational messages are acceptable, but fatal compilation or linking errors are not.)

Remember that the correctness of your shader files **cannot be verified** by `try` because they are not compiled until your application is run (which `try` won't be doing). The `try` scripts will verify that you have turned in an even number of shader files (i.e., files with names ending in `.glsl`), but that's about the extent of what I can do without diving into the source code to try to figure out what you named these files.

Important dates:

- First date on which submissions can be made: **Friday, December 13, 2013**
- Last date on which submissions can be made: **Thursday, December 19, 2013**

---

5. Grading

Your submission will be graded out of 100 points, as follows:

- 20 points for submitting code that compiles, runs, and produces a window.
- 15 points for object construction (5 points x three required objects)
- 9 points for model transformations (3 points x three required objects)
- 6 points for non-default camera position
- 5 points for using at least one light
- 10 points for using material properties on at least two objects
- 10 points for texture mapping on at least one object
- 10 points for resemblance of your image to the original artwork
- 15 points for programming style and extras (extra objects, animation, notable creativity in your image, well-structured and well-documented code, etc.)

The timeline for grading these submissions is very tight. I will begin grading them the day after the due date. If I discover a problem with your shader files when I am grading your submission (e.g., there are fatal compilation errors in your shader code, you have forgotten to submit one or more shader files, etc.), I will attempt to contact you via email to both your CS and RIT email addresses that day. If I do not hear back from you in time to allow me to regrade your submission before final grades are due, I will **assign you a grade of 'I' for the course** which I will change after I receive your corrected shader files and am able to grade your submission.

---

6. Notes

This point is critical, so I will repeat it: Remember that the correctness of your shader files **cannot be verified** by `try` because they are not compiled until your application is run (which `try` won't be doing). The `try` scripts will verify that you have turned in at least two shader files (i.e., files with names ending in `.glsl`), but that's about the extent of what I can do without diving into the source code to try to figure out what you named these files.
