Assignment #2

Point Value: 130 points

Due Date: March 20, 11:59pm

Spring 2015

Prof. James O'Brien

For this assignment you will write a ray tracer and use it to generate beautiful pictures that will astound all your friends.

1. **This assignment is due as indicted above. Projects turned in late will lose points as described in the policies handout. This assignment should be done in pairs. You may share ideas with other groups, but you may not share code.**

2. **You may develop on Unix, OS X, or Windows. The platform you use will be the one used to grade assignments. Keep in mind that there are slight variations due to OS versions, different libraries, and other factors, so you should verify that your code runs on the instructional machines appropriate for you platform choice.**

3. **We will be using the submit software for submission of this assignment. Instructions for using the submission software are here. You should include a README file that at the minimum contains the following information:**

   - Your (and your partner's) name
   - The platform your code runs on
   - The location of your source code (i.e. indicate who in your group has done the submission, and on what platform). Only one of the people in your group should submit the actual code. The other people should only submit the README file.

   All files needed to compile your code should appear in the submitted directory. It is your responsibility to make sure that they will compile and run properly.

   - Windows: The grader should be able to recompile your program by simply opening the project and rebuilding it from scratch.
   - Unix and OS X: The grader should be able to recompile your program simply by typing "make".

   You will also turn in some images. These should be named "image-nn.xxx" where nn is a number (e.g. 01, 02, 03...) and xxx is an appropriate extension (e.g. tif, jpg, ppm, etc.) The main input files needed to render those images should be named "input-nn". Finally there should be a "notes-nn" file for each input stating: the command line used to produce image-nn, how long it took to run, what features are demonstrated by the image, and any other comments you'd like to add about the image.

4. **Once you have your assignment working, you should also update your class web page to include an "Assignment 02" link to a page with images generated by your code. Make sure that your images demonstrate all the features that you have implemented in your code.**

   If you work in a group, you should all link to the same web page and the web page should list your group members.

5. **This assignment will be graded by looking at your web page to see that you have posted examples demonstrating all required features. The images on your website should correspond to the ones you submitted.**

   If you want to get credit for any required or optional feature, you must include an image that clearly demonstrates that feature!

   Each image should include a caption stating the command line used to produce the image, how long it took to run, what features are demonstrated by the image, and any other comments you'd like to add about the image. (*i.e. the same as the notes-nn file)*

   After you have submitted the assignment, you may add additional images to your webpage. , but these images must be clearly labeled as added after submission. Also, any images generated using a version of your code different from what was submitted must be labeled as such.

6. ***Do not wait until the last minute to start this assignment.* Even a minor bug in a ray tracer typically produces a black image with no other clue about what is wrong... that makes them very hard to debug. If you don't give yourself enough time, you will be quite unhappy.**

   Check the news group regularly for updates on the assignment or other clarification. We will assume that anything posted there is henceforth known to all.

7. **Submitting an image that was not generated by your code is considered cheating. Because raytracers may take a long time to generate a given image we can only spot check and you are largely on your honor that the images you show are yours. Please don't violate this trust.**

8. **Grading will include points for aesthetics and creativity as demonstrated by the images on your web page.**

   It is suggested that you take some time to create a variety of nice test scenes and render several images. Some images should be plain and simple to demonstrate individual features. Other images should be more complex and demonstrate a creative use of your software to generate interesting images.

   Several polygon models in the .obj format will be posted to Piazza. You are strongly encouraged to either make use of them or to obtain different models on your own.

   I also encourage you to share scene files on Piazza. If your scene file is creative and used by others then you will still be the one to receive extra credit for it. Also, if you use someone else's scene file then include that information in your caption along with a description of any ways in which you modified it.

9. The required features that you must implement for full credit are:

- Render arbitrarily oriented ellipsoids (actually, spheres are sufficient primitives so long as you can properly apply scales and rotations to them)
- Render polygons (*i.e. read a simple .obj file and render the polygons in it*)
- Use simple Phong Shading (in color)
- Compute shadows
- Compute reflections
- Apply linear transformations to objects
- Use point and directional lights
- Write its output to a standard image format such as jpg, ppm, png, or tif.

10. Optional features that you can implement for extra credit are:

- Use some reasonable method for accelerating ray tests (e.g. BSP trees or AABs)
- Transparency with refraction
- Anti-aliasing
- Lens effects / depth of field
- Super quadrics
- Programmable shading
- Texture, bump, and/or displacement mapping
- Spot lights and/or area lights
- Other ***interesting*** features

All features should be clearly documented in your README file and demonstrated in your example images.

11. The input format that should be read by your program is appended to the end of this document.

12. Your output needs to be in a standard image format that most image viewers can read. Suggested formats are: JPEG, TIFF, PNG or PPM. Do not use an indexed color format like GIF. You can write your own code to write the files or you can use a standard library. PPM is very easy to write, but offers no compression and may cause you to run out of disk space. JPEG,

TIFF, and PNG are supported by many standard libraries. (libTIFF, libPNG, ImageMagik, freeimage, and many others) You are responsible for figuring out how to use these libraries.

13. **Your images should be at least 1000x500 and no more than 3000x3000.**

14. **There is no reason to be using OpenGL for this assignment.**

15. **This assignment is purposefully open ended. I am continually impressed by what Berkeley undergrads can do when given a bit of freedom, so here it is. You are encouraged to be creative and enjoy this assignment.**

16. **Questions should be posted to the discussion group or emailed to cs184.**

Raytracer Input Format

- **Each line in the file specifies some sort of object, light, material, camera, or transformation.**
- **The camera object is specified by a line of the following form:**

  cam ex ey ez llx lly llz lrx lry lrz ulx uly ulz urx ury urz
- **There will only be one camera line in a file.**
- **A sphere is specified by:**

  sph cx cy cz r
- **A triangle is specified by:**

  tri ax ay az bx by bz cx cy cz
- **A .obj file is specified by:**

  obj "file name"
- **A point light source is specified by:**

  ltp px py pz r g b [falloff]

  Where falloff is 0 for no falloff, 1 for linear, or 2 for quadratic, and the default is 0
- **A directional light source is specified by:**

  ltd dx dy dz r g b
- **An ambient light source is specified by:**

  lta r g b
- **A material is specified by:**

  mat kar kag kab kdr kdg kdb ksr ksg ksb ksp krr krg krb

  Where ka, kd, ks, and kr are the coefficients for ambient, diffuse, specular, and reflective.
- **A transformation is specified by any of the following:**

  xft tx ty tz

  xfr rx ry rz

  xfs sx sy sz

  These refer in order to translation, rotation, and scaling. Rotations are exponential maps in degrees.

- **Transformations apply cumulatively in the reverse order they appear in the file.**

  Thus the following lines:

  ```
  xfr 45 0 0
  xfs 1 1 2
  xfr -45 0 0
  xft 1 2 3
  ```

  Would result in the following transformation matrix:

  ```
  [r 45 0 0]*[s 1 1 2]*[r -45 0 0]*[t 1 2 3]
  ```

- **When an object line is read, the current transformation will be applied to that object.**
- **Transformations also apply to lights or the camera. However, you are not required to implement this feature. If your input files contain camera and all light lines before the first transformation then this feature would have no effect.**
- **The current transformation may be reset to the identity with the following line:**

  ```
  xfz
  ```

- **When an object line is read, the last read material will be applied to that object.**
- **Lines with an unrecognized type should result in a warning message to stderr indicating an unsupported feature and the program should then ignore the line.**
- **If a line has extra parameters then those parameters should be ignored and a warning message should be printed to stderr.**
- **Any number of tabs and spaces is equivalent to a single space.**