

CPSC 314

Assignment 3

Due 4PM, Wednesday, Dec 1, 2004

Answer the questions in the spaces provided on the question sheets. If you run out of space for an answer, use separate pages and staple them to your assignment.

Note: Some of the questions in the assignment are based on material to be covered between now and Dec 1. Answer those after the topics are covered in class.

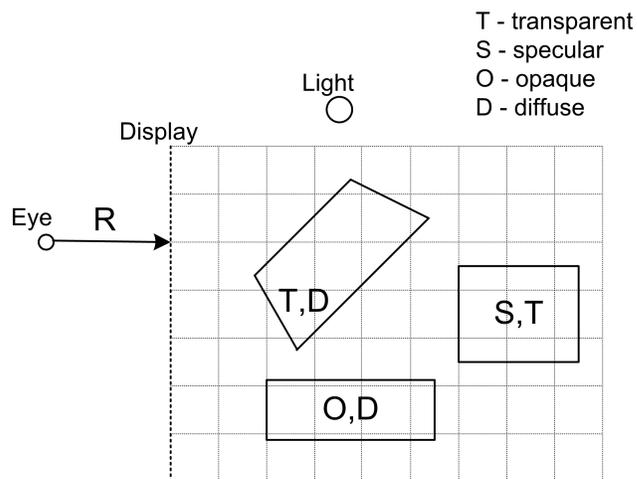
Name: _____

Student Number: _____

Question 1	/ 5
Question 2	/ 5
Question 3	/ 5
Question 4	/ 5
Question 5 (bonus)	/ 5
Question 6	/ 90
TOTAL	/ 100

- (5 points) Rasterization: Write the algorithm (pseudo-code) for integer only (Bresenham type) rasterization of the parabola $y = x^2$ for $1 \leq y \leq 25$.

- (5 points) Ray-Tracing: Draw the ray tree for the ray R shown below. Assume index of refraction c_1 for air is 1 and index of refraction for all the transparent objects in the scene is $c_2 = \frac{1}{\sqrt{2}}$. Use Snell's law to obtain refraction angles.



3. Classify the following statements as right or wrong (provide a *short* explanation).
- (a) (1 point) If all objects in a scene have specular coefficient $k_s = (0, 0, 0)$ then using Phong model or Blin-Phong model makes no difference to the way the scene is rendered.

 - (b) (1 point) If all objects in a scene have specular coefficient $k_s = (0, 0, 0)$ then replacing a point light-source with a directional light source will not change the way the scene is rendered.

 - (c) (1 point) If a scene is illuminated with only ambient light, all the objects look flat.

 - (d) (1 point) A diffuse, non-specular unit cube at the origin, lit by 6 directional light sources along the $+x, -x, +y, -y, +z, -z$ directions, rendered using flat shading, looks as if it was illuminated by ambient light.

 - (e) (1 point) The polygon scan conversion (edge-walking) algorithm works for non-simple (self-intersecting) polygons.

4. (5 points) Curve Continuity:

Given the following parametric curves:

$$\begin{aligned} f^1(u) &= (u, u), & u \in [0, 0.5] \\ f^2(u) &= (1 - u, 1 - u), & u \in [0.5, 1] \\ f^3(u) &= (2u^2 - u + 0.5, 2u^2 - u + 0.5), & u \in [0.5, 1] \\ f^4(u) &= (2u^2, 2u^2), & u \in [0.5, 1] \end{aligned}$$

will the curves defined in the table below be continuous (C^0 , C^1 , G^1) at $u = 0.5$ (each curve is defined on $[0, 1]$ by combining the functions on $[0, 0.5]$ and $[0.5, 1]$).

	C^0	C^1	G^1
f^1 and f^2			
f^1 and f^3			
f^1 and f^4			

5. (5 points) **Bonus** Given points $P_1, \dots, P_n, P_{n+1} = P_1$ in 3D develop the formula for a smooth C^2 curve passing through them (Hint: use Hermite curves for each segment P_i, P_{i+1}).

6. (90 points) Graphics Extravaganza or Dogs at Large.

In this assignment, you will be using all the OpenGL/Graphics knowledge you acquired in the course so far, to generate a fun interactive environment based on your dogs from assignment 1. Options include an animation, a game, or a simple modeling software. It is recommended that you work in groups of two. Hand in one project. Suggested projects are given below. In addition to assignment 1 you can use any parts of the OpenGL demo programs provided during this course as a starting point. Document what demo parts you used in your README file. If you need help regarding how to implement any particular features, do not hesitate to ask the instructor or the TAs. Be sure to develop your project in testable stages. The best projects will be glorified forever in the 314 Hall of Fame!

Grading To have your programming assignment graded out of 90, you need to implement four standard features and one advanced feature (in this case the total grade of assignment 3 will be marked out of 110 - not counting bonuses). Alternatively, you can implement all five standard features and no advanced. In this case your programming assignment will be graded out of 80 (in this case the total grade of assignment 3 will be marked out of 100 - not counting bonuses). It is recommended therefore that you implement four of the standard features first and only then turn to the advanced ones.

Standard Features Whichever of the projects below you choose to implement, it must include at least four of the five standard OpenGL features below.

- Camera motion around the scene supporting rotation, translation and zoom.
- Multiple (colored?) light sources.
- Complex environment including imported (obj files) objects.
- Texture mapping for at least some of the objects in the scene. Either create a simple texture or load one (use parts of assignment 2 template for this). Note: OpenGL only supports textures of size $2^k \times 2^k$.
- Interaction using mouse operations. Glut provides a set of commands to support this.

Think of how best to incorporate those into your project.

Advanced Features

You should also implement one of the advanced features described below as part of your project (if you want it to be graded out of 90 - see above).

- Shadows - implement those with simplified lighting conditions (one light source). Hint: use multiple copies of Z-buffer to generate the scene with/without lighting. The OpenGL commands used to access Z-buffer include `glDrawPixels`, `glCopyPixels`, `glReadPixels`, `glDrawBuffer`, `glReadBuffer`. Read the relevant entries in OpenGL manual to understand how to use them.
- Smooth motion path using splines (this can be integrated in project options 1 and 2).

- Environment mapping - use multiple copies of Z-buffer. To figure out how to access z-buffer read the references for shadows (above).
- Picking - your code would need to recognize which object the mouse location corresponds to (when mouse or keyboard callback is evoked). For a tutorial see <http://www.lighthouse3d.com/opengl/picking/> and <http://www.ugrad.cs.ubc.ca/cs314/Vsep2003/slides/week4.wed.col.pdf>

Think of how best to incorporate those into your project.

Document the standard and advanced features you have implemented.

You have the following three project options. Be creative and add your own features to those, as you wish. make sure your readme documents all the interactive options of your project. If you want to develop a different project, you must consult with the instructor or the TAs *prior* to starting it.

Dog Walk

Create a park environment to take your dog for a walk (e.g. paths, trees, lawns). Implement a mouse or keyboard interface for steering your dog around in this world. Your dog should perform different actions (e.g. walking, sitting, running, etc...) and interact with the world (e.g. bark at people, avoid collision, etc...). Develop some mechanism for your program to decide which action to perform. Make the dog's motion as smooth as possible.

Dog Circus

Create a short animated scene of a dog show. Create a circus type environment and make your dogs do a few tricks with different objects (e.g. throw balls, jump through hoops, etc...). Create funky environment effects, e.g. lights, mirrors, smoke,... Make the motion as smooth as possible.

Dog Modeling

Implement a modeling environment for generating the dog of your dreams. Allow users to change the color/texture/size/shape of the dog components. You can add extra primitives, as you like. This project fits well with implementing the picking advanced feature option above - use it to select parts to be modified. Provide an option to do vertex level changes to your model, where a user can specify a particular vertex and modify its color, location, etc... To support this, you should replace the glut based ellipsoid construction function with your own modeling primitive. Try to find a way to store the dogs you generate, so you can re-load them afterwards.

BONUS The best three projects will get extra bonus points (10 for first, 8 for second, and 5 for third) and they will be entered into the 314 hall of fame.

Hand-in Instructions

You do not have to hand in ANY printed code. Create a README file that includes your name, your login ID, and any information you would like to pass to the marker.

When working in pairs, only one of the partners needs to submit the code. Make sure however that **both** names are listed in the README file.

Create a folder called 'assn3' under your cs314 directory and put all the source files, your makefile, and your README file there. Also include any images that are used as texture maps. Do not use further sub-directories.

The assignment should be handed in with the exact command:

```
handin cs314 assn3
```

Programming Assignment Grading

Similarly to assignment 1, the programming part of this assignment will be graded with face-to-face demos in front of the TAs/Instructor.

Document all the extra features you implemented in the README. If need be, you can also concisely summarize the arguments in your README about incomplete work.

- We will hang a signup sheet for demo slots in the lab (CS 011). Each slot will be 10 minutes. The demo sessions will be Wednesday, Dec 1, Thursday, Dec 2, Friday, Dec 3, and Monday Dec 6. (except during scheduled lab/lecture times).
- You must ensure that your program compiles and runs on the lab machines. If you worked on this assignment elsewhere, it is your responsibility to test it in the lab. The face to face grading time slots are short, you will not have time to do any 'quick fixes'! If your code as handed in does not run during the grading session, you will fail the assignment.
- The code that you demo must match exactly what you submitted electronically: you will show the TA a long listing of the files that you're using, so that he can quickly verify that the file timestamps are before the submission deadline.
- Have a printout of your README file for the TA to read.
- Arrive at CICSR 011 at least 10 minutes before your scheduled session. Log into a machine, and double-check that your code compiles and runs properly. Then delete the executable.
- When the TA comes to your computer you will type the following:

```
ls -l  
make
```
- You will then run your assignment. Since you have exactly 10 minutes, think in advance how you want to demo all the cool features of your project. If you fail to mention a feature, it might be overlooked.