SUBJECT CODE: CS 6620

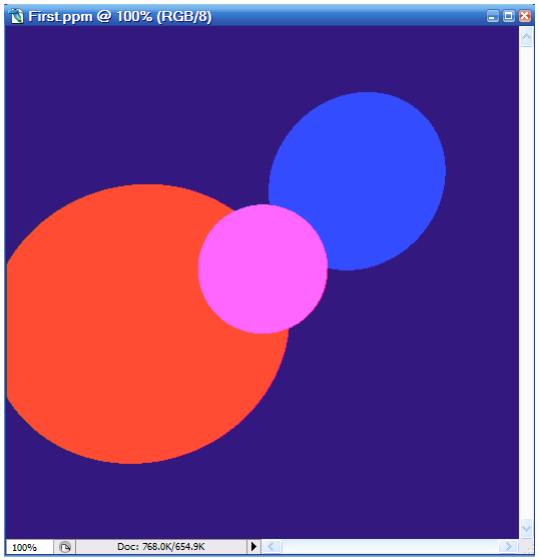
SUBJECT: ADVANCED COMPUTER GRAPHICS II

LEENA KORA
EMAIL:leenak@cs.utah.edu

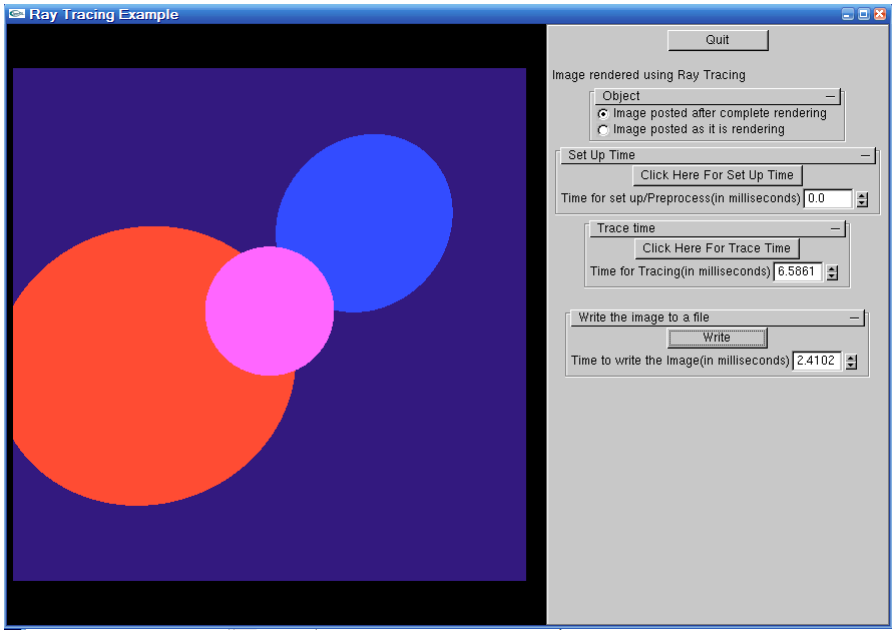
Unid: u0527667

PROGRAM 1

PART A(Required Image)



PART B(Extra Credit Part)



Link to my code: http://www.cs.utah.edu/~leenak/Program1.txt

Description of my design choices

Part I

I first created a color class by defining it in RGB_Color.h and implementing it in RGB_Color.cpp file. I added 'red', 'green' and 'blue' as data members of the color class to color each pixel with all the three color components. Then I added all the necessary functions for adding, subtracting and multiplying two colors. Then I decided to separate out points from vectors by creating Vector.h and Point.h header files and implementing them in their respective .cpp files. I did so because I thought separating points from vectors could give more readability and meaning to my code and ray tracing concept. I added all the necessary computational tasks that can be done using points and vectors by adding them as member functions in respective classes. Then I implemented 'ray' class by using vector and point objects. Then I checked the implementation of the three classes vector, point and ray by creating a small program and running it. I then created a sphere class with 'point' object and 'double' data type as it data members. I added only one member function called 'Intersects' to check whether the incoming ray hits the sphere or not. Then I created an Image class by adding two very important functions 'set' and 'write_file' to set the color of each pixel and to write it to a .ppm file. I chose to use PPM format because it is easy to create the programs that handles these files. As I was programming in Visual Studios C++ on Windows, I didnot use Time.h and Time.cpp files provided to us. So I used GetTickCount() function which gives time in milliseconds. I used it in my program by adding windows.h header file for its definition.

Part II(Extra Credit)

For this part I used 'glui.h' and 'glut.h' for creating front end or GUI and handling windowing system respectively. I used opengl API's along with Glut commands to create my Image Viewer. I used glDrawPixels() command to render the image on the screen. I also provided the option for the users to switch between image posted after complete rendering and image posted as the image is rendering. I also provided the buttons for the users to have a look over preprocess time, tracing time and time in millsecond required for writing the image to a file.