Ray Tracing

Assessable assignment 2 **Weighting: 20%**

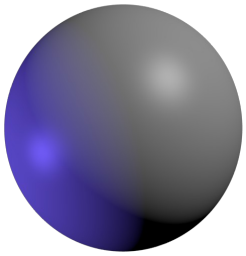
DUE DATE: Friday, 11th May 2012 at 5pm
NO LATE ASSIGNMENTS ACCEPTED

For this assignment you are asked to write a basic "sliced sphere" ray tracer.
Formally, a 'sliced sphere' is a unit sphere ($x^2 + y^2 + z^2 = 1$), where MIN < x < MAX, for given values MIN, MAX

- If the values for MIN and MAX are larger than 1 (e.g., MIN = -2.0, MAX = 2.0), then the 'sliced sphere' will represent a normal sphere. If you are inside the sphere, then you will only be able to see the inside surface.
- If only one of the values is less than one (e.g., MIN = -2.0, MAX = 0.0), then the sphere will have a 'hole' in it – with the example values given here the hole actually cuts off half of the sphere leaving a hemisphere. Depending on how the object is aligned to the camera, you will be able to see both the outside surface, *and* the inside surface of the sphere.
- If both of the values are smaller than 1 (e.g., MIN = -0.5, MAX = -0.2), then the sphere will have two 'holes' in it, and may look like a ring or bracelet. Again, you may be able to see both the outside, and the inside surface of the object. (See the example image of a 'ring shape' below.)

Note: there is no constraint that says MIN is negative and MAX is positive (note the previous example!),
the only constraint is MIN < MAX (also, MIN < +1 and MAX > -1 ... why?)

The scenes you will raytrace will be made up of these 'primitive' sliced spheres, each of which will have a number of different transformations applied (translations, rotations, stretches). An object can have any number of transformations applied to it, and these transformations could be specified in any order.



1. Starting with the skeleton program that we provide, write and test routines that ray trace and shade the objects in the scene. The skeleton program defines the required user interface, reads scene information from a file (the format is described overleaf), and sets up many (but not all) useful data structures.
2. The sample file only aims to demonstrate valid use of the scene file format. You must test your program with scenes of your own design.
3. Write a *short* report, describing how your program works and how you tested it. Describe any known flaws in your program.

The ray tracer skeleton is in the directory `/coursework/342/pickup/ass2/`, along with some sample test files. Submit your work by using the **submit342** script. Make sure your submission is in a directory with your name on it, and that it contains your source code, makefile, report and at least one of your own test scenes which we will use in an art display for the course. Also make sure that your name is included in the report and in the introductory comments in your program. We will e-mail you to confirm that we have received your assignment within a week of the due date. Keep a copy of your work at least until the assessment has been returned to you.

The report must be submitted as a plain text file (**NOT** OpenOffice or Word) and must be *not more than two pages long*. We do not want any assignments handed in on paper!

The skeleton code and test file will be available on the 3rd of April.

The marking scheme for this assignment is as follows:

- 3 marks Basic ray intersection showing colours and geometry of translated objects with ambient light.
- 1 mark Program also works for uniformly stretched objects.
- 1 mark Program also works for arbitrarily stretched objects.
- 2 marks Program also works for rotated objects.
- 3 marks Diffuse shading.
- 2 marks Phong shading.
- 3 marks Shadows.
- 2 marks Mirror reflections.
- 3 marks Report and design of scene file.

Scene description file format:

Important details not included in the file:

Our view plane is a square view plane that extends from (top-left) (–2, –2, 0) to (bottom-right) (2, 2, 0).
We have a right handed coordinate system.
A positive rotation about the x-axis moves the y-axis towards the z-axis.
A positive rotation about the y-axis moves the z-axis towards the x-axis.
A positive rotation about the z-axis moves the x-axis towards the y-axis.
Use the object's diffuse coefficients when calculating ambient lighting effects.

First lines:

Any of the following settings may be specified.
If a setting is not included in the scene description file, then the associated default value(s) will be used.

`imagesize n`

The text "imagesize" followed by an integer, n, which means the final picture is n by n pixels.
You should fire exactly one ray through the centre of each pixel.
The default image size is 320

`view_distance d`

The text "view_distance" followed by a real value, d, which tells us the distance from the viewing position to the viewplane.
This means we are looking at the view plane from the position (0, 0, -d) [note the negative!]
The default view distance is 4.0

`background br bg bb`

The text "background" followed by real numbers:
br, bg, bb gives the red, green and blue components of the background colour.
These components will have real values in the range 0..1 .
The default background colour is 0.0 0.0 0.0

`ambient ar ag ab`

The text "ambient" followed by real numbers:
ar, ag, ab gives the red, green and blue components of the ambient light.
These components will have real values in the range 0..1 .
The default ambient light is 0.0 0.0 0.0

Successive lines:

`light x y z r g b`

The text “light” followed by real numbers:

x, y, z gives the location of a point light source.

r, g, b gives the red, green, and blue intensity of the light in the range 0..1.

The intensity of the light source does not diminish with distance.

We will test your program with **eight** or fewer point light sources.

.... OR ....

`sphere min max`

The text “sphere” followed by the real numbers:

min will be a double value < max. If min < -1 then it will not 'slice' the sphere.

max will be a double value > min. If max > +1 then it will not 'slice' the sphere.

The sliced sphere will be those points on a unit sphere where min < x < max.

and immediately following will be the line:

`material dr dg db mr mg mb pr pg pb p`

The text “material” followed by real numbers:

dx dy dz diffuse reflection coefficients for red, green, and blue.

mr mg mb mirror reflection coefficients for red, green, and blue.

pr pg pb Phong lighting coefficients for red, green, and blue.

(The diffuse, mirror, and Phong coefficients will have values in the range 0..1).

p Phong shading exponent.

We will test your program on scenes with **fifty** or fewer spheres.

Each object will be initialised with an identity transformation, which will be modified by any number of the following transformations, until the next object in the scene is defined.

`translate tx ty tz`

The text “translate” followed by real numbers.

Indicates that the current object is to be translated along the x-axis by tx, the y-axis by ty and the z-axis by tz.

`stretch sx sy sz`

The text “stretch” followed by real numbers.

Indicates that the current object is to be stretched along the x-axis by sx, the y-axis by sy and the z-axis by sz.

`rotate axis theta`

The text “rotate” followed by an axis code and a real number representing and angle in degrees.

The axis code is a single character, 'X', 'Y', or 'Z'.

The current object is to be rotated about the axis by theta degrees.

Last line:

`endview` The text “endview”.

Remember this is an assignment in graphics not in production programming. You will not get any extra credit for extravagant solutions. High marks will be given for a well explained solution that is easy to follow. Programs must be commented of course, but no more than necessary for us to be able to read them.

You may discuss conceptual issues relating to this assignment with others, but all the work that you hand in must be your own, except for the parts of the given skeleton program.