LAB 1: POV-RAY INTRO

10/10/2016
This lab will look at using POV-Ray for:

- Geometry: Preset shapes
- Colour
- Positioning geometry (transformations), camera and light source with vectors

POV-Ray should already be installed in the lab from last week. But due to some maintenance work you may need to reinstall one more time.

**Task:** Model one or both of the following objects in POV Ray.

I recommend you don’t try the second object before the next lab unless you have experience with 3d vectors and transforms.
TYPICAL POV-RAY SCENE FILE

3D Scene

Camera

camera
{
    <location>
    <direction>
    [other optional settings]
}

Light

light_source
{
    <location>, colour,
    [other optional settings]
}

Geometric Object(s)

Object_name
{
    Geometry description
    Colour,
    [other optional settings]
}
Before you get started, you might want to use the following snippet for camera and light source objects. We will discuss more details about these later. For now just copy and paste this into POV-Ray

```plaintext
camera
{
    location < 0, 0, -8 >
    look_at <0, 0, 0>
}

light_source
{
    <10, 10, -10>,
    rgb <1, 1, 1>
}
```

Your task today deals with geometry and vectors for positioning objects (or points)
sphere
{
<0,-1,0>, 1
pigment { rgb <1, 1, 0> }
}

cylinder
{
<-3, 0, 0>,
< 3, 0, 0>, 0.5
pigment { rgb <0, 0, 1> }
}
SPHERES

sphere
{
  <0,-1,0>, 1
  pigment { rgb <1, 1, 0>}
}

A vector denoting the position of the sphere’s center
The sphere’s radius
A vector denoting the colour of the sphere
The cylinder's radius

A vector denoting the colour of the cylinder

2 vectors denoting the position of the ends of the cylinder

cylinder
{
    <-3, 0, 0>,
    < 3, 0, 0>,
    0.5

    pigment { rgb <0, 0, 1> }
}

CYLINDERS
Your task this week is to model the object on the right in POV-Ray.

Verbal Description of the Scene:
- Four white cylinders are arranged to form a square.
- A red sphere is centred at each corner of the square and has a radius about twice that of the cylinders.

Use and modify the sphere and cylinder code provided to create this scene.

N.B. You do not have to model the objects in this particular view (camera angle).

It will be easier to model them as if looking at them from directly above. You can move the camera later.

```plaintext
camera {
    location < 0, 8, 0 >
    look_at  <0, 0, 0>
}
```
MORE ADVANCED STUFF

This is optional for this lab
(Only attempt this if all previous is completed or obvious)
CAMERA (SIMPLEST EXAMPLE)

camera
{
    location < 0, 0, -8>
    look_at <0, 0, 0>
}

Position of camera
A point that the camera is pointing at
LIGHT SOURCE (SIMPLEST EXAMPLE)

light_source
{
    <10, 10, -10>,
    rgb <1, 1, 1>
}

Position of light source (assumed to be radiating equally in all directions)

Colour of the light

LIGHTSOURCE
(N.B. For illustration only. You’ll never actually see the source)
plan

{
  <0, 1, 0>, 4
  pigment { rgb <0.5, 0.5, 0.5> }
}

The normal to a surface is a line that is orthogonal (90 degrees) to it. Kind of like the pin on a thumb-tack.

**TASK 2 (OPTIONAL): MORE COMPLEX SHAPES**

A pyramid shape

A cube shape
A house shaped object made up of three boxes; the two roof planes are boxes rotated by small amounts.

The objects on the roof are a repeat of the big “house”, scaled down, rotated and translated.

A slightly more complex example. You will need to use the box object shown below. Fill in the blanks with the right values.

```plaintext
box
{
  < ??? >,< ??? >

  pigment { rgb <???>}
  rotate ??? * <???>
  scale < ??? >
  translate < ??? >
}
```

Angle

Axis (A vector)
box
{
  <-1, -1, -1>, <1, 1, 1>
  pigment {rgb <0, 0, 1>}
}
Several objects can be combined to make up one object using the union object:

```plaintext
union
{
    sphere { <0,0,0>, 1 pigment{rgb<0, 0, 1>}}
    cylinder {<0,-1,0>, <0, 1, 0>, 1 pigment{rgb<1, 0, 0>}}
}
```

More on union and other such operators tomorrow, but if you are trying to complete Task 2, you may need to use `union`. 
RE-USING BITS OF CODE

Example syntax:

```plaintext
#declare MY_OBJECT = union
{
  sphere {... }
  box { ... }
}
```

Then when you want to draw this, you just have to add the following line (you can do this as many times as you want)

```plaintext
object
{
  MY_OBJECT
  translate <2, 0, 0>
}
```

N.B. This is any name you want to give it

N.B. You can attach a name to any object (doesn't have to be a union)

N.B. You can use any other transform (e.g. translate/rotate/scale) here..
You can also change texture/pigment values
A LITTLE BIT MORE ABOUT POVRAY

If you want to skip ahead a little bit, you can look at POV-Ray documentation for:

- Texture
  - Pigments
  - Finish
- Materials
- Bumps
- Constructive Solid Geometry
- Camera Parameters

These will be used in the first major assignment

For a lighter read, a nice free online reference tutorial is available here:
http://www.alexandre.eletrica.ufu.br/cg/tutorial.pdf
What is a scalar?

What is a vector?

How can we represent the following in terms of vectors:

- Position
- Colour
- Direction
POV-Ray is freely available for multiple platforms if you want to download and install it at home or on your laptop, from ... http://www.povray.org

Documentation & Examples

- Function definitions and examples are provided in help files press F12.
- Additionally in: http://www.povray.org/documentation/
- For a lighter read, a nice free online reference tutorial is available here: http://www.alexandre.eletrica.ufu.br/cg/tutorial.pdf
- Some samples provided in: My Documents\POV-Ray\v3.7\scenes