RECAP OF POVRAY
Key points yesterday’s class

- Position and movement with vectors
- Intro to POV-Ray

- Some peripheral details
  - Using a script (getting used to syntax)
  - Using an IDE
POV-RAY OBJECTS

Usually in the form....

Object_name
{
  //Basic parameters (these may vary in type and number but general scalars or vectors)

  //pigment {} (pigment relates to colour of object and is required if you want to see anything) - some objects don’t require this (see later)

  //Transforms (optional)
}

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

cylinder
{
  <-3, 0, 0>,
  < 3, 0, 0>, 0.5
  pigment { rgb <0, 0, 1> }
}
If it was your first time being exposed to vectors and you had some trouble getting the last lab working, here’s one possible way of going about it.

1. First of all, let’s simplify it by turning it into a 2D problem (Note that all the Y values are zero). Think of modelling a square using vectors.

It’s best to create the scene centred at the origin: <0,0,0>.

What are the co-ordinates of the four corners? There are many correct answers. Note the symmetry (a “2” almost everywhere).
2. Place a sphere, centred at each of your square’s corners. This is relatively easy if you already know what the corners are. Choose a radius for your sphere.
3. Place a cylinder at each edge of your square.

For a cylinder all you need to know is a radius and two end points. Once again these endpoints are just the corners of the box (you just need to decide which ones).
YESTERDAY’S SOLUTION

When you’re done it should look like this:

Now move the camera to somewhere more interesting.

[* TIP#1*] when you are trying different positions and orientations of objects, whilst creating the scene, it is best to change as little as possible, otherwise it’s hard to determine what number actually caused a change in the appearance of the scene.

[* TIP#2*] If you must move the camera during creation choose very basic camera positions, such straight above \(<0, 8, 0>\), straight in front \(<0, 0, -8>\), straight to left \(<8, 0, 0>\) of the origin.

Here’s a link to the full solution file (I added a plane to it – not discussed here):

**Task #2 (In Brief)**

**The Pyramid:**
- Choose a fifth point over the origin, just slightly lifted e.g. \( <0, 3, 0> \)
- Then create 4 more cylinders. Each must contain this new point and one of the previous four.

![Diagram of the Pyramid](https://www.cs.tcd.ie/John.Dingliana/cs7029/samples/task2a.pov)

**The Cube**
- Create almost an identical copy of the “square” except “lift” it a bit by giving all the new cylinders and spheres a bigger \( Y \) value.

![Diagram of the Cube](https://www.cs.tcd.ie/John.Dingliana/cs7029/samples/task2b.pov)

- Create four more cylinders, connecting each of the original spheres to the new one above it.

```pov
cylinder {
    \(<-2, 0, 2>, \<2, 0, 2>, 0.5\)
    pigment { rgb \( <0, 0, 1> \) }
}
```

Note that these points are almost identical except for the \( Y \) value i.e. one is directly above the other.


The solution to Task 3 will be discussed during the rest of the lecture.
In this lecture:

• How are 3D Models Created
• How are 3D Models Represented
• How are 3D Models Modified

3D OBJECT MODELLING
MODELLING ISSUES

1. Where do models come from?

2. How are models stored?

3. How are models modified?
MODELLING PROCESSES

Creation / Acquisition

- Image & 3D Scanners
- 3D Model-extraction (from images/video)
- Procedural Modelling
- Mechanical Digitizers
- Modelling Tools
- From Scratch

Manipulation / Editing

- Rendering
- Rasterization
- Procedural Animation
- Animation Tools
- Data Driven
- Direct Manipulation

HIGHLY AUTOMATED → MANUAL
3D SCANNING
1. Airborne Modelling
3D models of rooftops and terrain shape from airborne laser scans and photos.

2. Ground-Based Modeling
Generation of 3D models of facades and street scenery as seen from street level.

3. Model Fusion
3D MECHANICAL DIGITIZERS

Co-ordinate measurement machine (CMM) record 3D points

- Similar to “tracing” in 3D

Ghost 3D Microscribe

Ghost 3D Microscribe Laser
Some useful patterns (in nature) can be approximated quite well by some simple math e.g. the Fibonacci Sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, ..
FIBONACCI SEQUENCES

Fibonacci in Nature

© Benjamin Martinek

Images from Pinterest © of respective authors

Computer Generated Models

© TeXample.net

© wakeup-world.com

© lava360.com

© Benjamin Martinek
PROCEDURAL MODELLING
PROCEDURAL MODELLING
PROCEDURAL MODELLING
MODELLING 3D OBJECTS
MODELLING TECHNIQUES

2D Images and Models
- Points, Lines, Polygons
- Raster Images
- Curves

3D models
- Polygonal
- Volumetric
- Mathematical surfaces
OPTIONS FOR MODELLING 3D SHAPE

Boundary Representation
define the surface or shell of an object

Spatial Partitioning
define full volume of object (including insides)

Continuous Models
defines complete smooth shape (infinitely detailed)

Discrete Models
approximate shape using primitive building blocks
Start with Points/Vertices on the surface of an object

- Defined as positional vectors \( \langle x, y, z \rangle \) from the origin

Edges are line segments on the surface, defined by pairs of points.

Closed polygons are made up of a number of co-planar edges.
POLYGON MESH

Polygon List

- P1: e1, e6, e7
- P2: e7, e3, e2
- P3: e5, e4, e6

Edge List

- e1 → v1, v2
- e2 → v2, v3
- e3 → v3, v4
- e4 → v4, v5
- e5 → v5, v1
- e6 → v1, v4
- e7 → v2, v4

Vertex List

- v1 → {x1,y1,z1}
- v2 → {x2,y2,z2}
- v3 → {x3,y3,z3}
- v4 → {x4,y4,z4}
- v5 → {x5,y5,z5}
POLYGON MESH

This is the basis for modelling a large range of complex shapes in 3D applications
In most cases, mathematical models need to be converted to discrete representations before we use them in scenes. We need to find an appropriate *resolution* for this.
ADAPTIVE REFINEMENT

Sometimes the trick is to use larger polygons in flat areas and a greater number of small polygons in areas of fine detail.

Alternatively more detail in silhouette areas.
VOLUMETRIC MODELS

Voxels (Volume elements/Volumetric pixels)
MATHEMATICAL MODELS

Given an equation that defines a shape as a function of \( t \).
At display time, give it various values of \( t \) and this gives us lots of \( x \) and \( y \) values as required

\[
x = \sin 4t \\
y = \cos 2t
\]

Parametric

Given an equation for a shape. This form of mathematical equation is better for answering questions such as:
- Does a point (e.g. \(<2, 2, 2>\)) lie on the surface?
- Is the point inside or outside the objects surface?

\[
x^2 + y^2 + z^2 = r^2
\]

Implicit
CONSTRUCTIVE SOLID GEOMETRY
SCENE MODELLING

Often we want to reuse individual models

- For repeating geometry in different parts of the scene
- For animation

Instead of “re-modelling” the object again we can re-create it in a different configurations using transformations.
Individual object geometries are first modelled. These are then linked using positional/orientational relationships (transforms).
A SCENE GRAPH

There can be multiple levels of transform hierarchy e.g. the “door glass” is linked to the “door” which is linked to the “body”
TRANSFORMS
Rigid Transformations

- Translate
- Scale*
- Rotate

Other transformations
Translate by $<0, 1, 0>$
Please Note: distance from the origin also scales
**ROTATION**

Rotation of 45° about the Z axis

Offset from origin rotation

In POV-Ray:
rotate 90°<0,0,1>

Angle

Directional vector describing axis of rotation
Transform order is important

Rotate by 45 then Translate by <1, 0> then Rotate by 45

Translate by <1, 0> then Rotate by 45
TRANSFORMS IN POV-RAY

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

Top, left, front corner

Bottom, right, back corner
box
{
  <-1, -1, -1>, <1, 1, 1>
pigment {rgb <0, 0, 1>}
rotate ??? * <????>
scale < ??? >
translate < ??? >
}

Place any required transforms after the rest of the object description.
PARTIAL SOLUTION TO TASK #3 OF YESTERDAY’S LAB

1. First create a rectangular block as the base of the “house”

2. For the left half of the roof:
   - Make a thinner narrower box, same length.
   - Rotate it by a small angle
   - Lift it up a bit to roof height

3. Repeat for the right half of the roof:

4. Put it all together
   - Note: I’ve changed the camera position for a better view

Please Note: I’ve left out the camera and light code here (this needs to be added for this to work)

See: https://www.cs.tcd.ie/John.Dingliana/cs7029/samples/task3.pov
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
#declare house_shape = union
{
  box
  {
    <-1, 0, -2>, <1, 1, 2>
    pigment{ rgb1, 0.5, 1 } 
  }
  box
  {
    <-1.3, -.1, -2>, <0, .1, 2>
    pigment{ rgb1, 0.5, 1 } 
    rotate<0, 0, 35>
    translate <0, 1.6, 0>
  }
  box
  {
    <1.3, -.1, -2>, <0, .1, 2>
    pigment{ rgb1, 0.5, 1 } 
    rotate<0, 0, -35>
    translate <0, 1.6, 0>
  }
}

//draw a big house shape
object
{
  house_shape
}

//draw a small version of the house
object
{
  house_shape
  scale <0.5, 0.5, 0.5> //halve the size
  rotate <0, 90, 0> //rotate by 90 degrees
  //around the Y axis
  translate <0, 0.75, 1> //move up a bit
}

//second set of windows
object
{
  house_shape
  scale <0.5, 0.5, 0.5>
  rotate <0, 90, 0>
  translate <0, 0.75, -1>
}