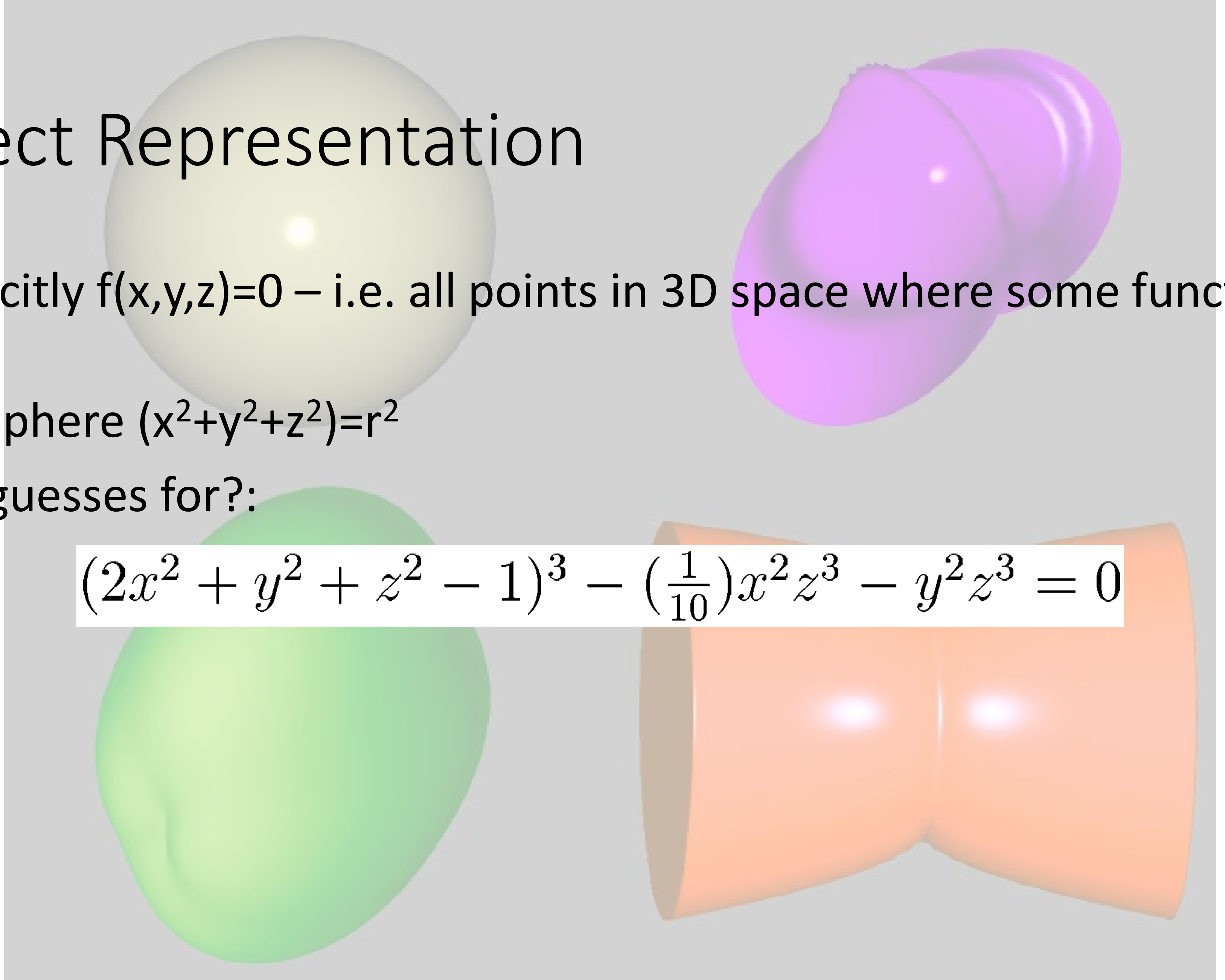
Object Representation

- Implicitly f(x,y,z)=0 – i.e. all points in 3D space where some function is zero.
- e.g. sphere $(x^2+y^2+z^2)=r^2$
- Any guesses for?:

$$(2x^2 + y^2 + z^2 - 1)^3 - (\tfrac{1}{10})x^2z^3 - y^2z^3 = 0$$

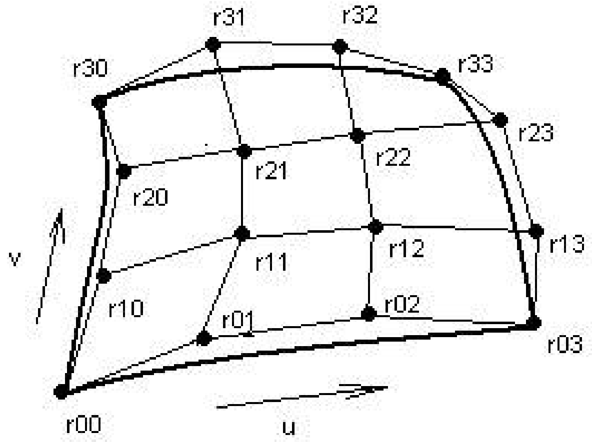


Object Representation

- **Implicitly** f(x,y,z)=0 – i.e. all points in 3D space where some function is zero.
- e.g. sphere $(x^2+y^2+z^2)=r^2$
- Any guesses for?:
  $$(2x^2 + y^2 + z^2 - 1)^3 - (\tfrac{1}{10})x^2z^3 - y^2z^3 = 0$$
- It's a cartoid (heart!)
- **Implicits**:
  - Extremely compact storage

Object Representation

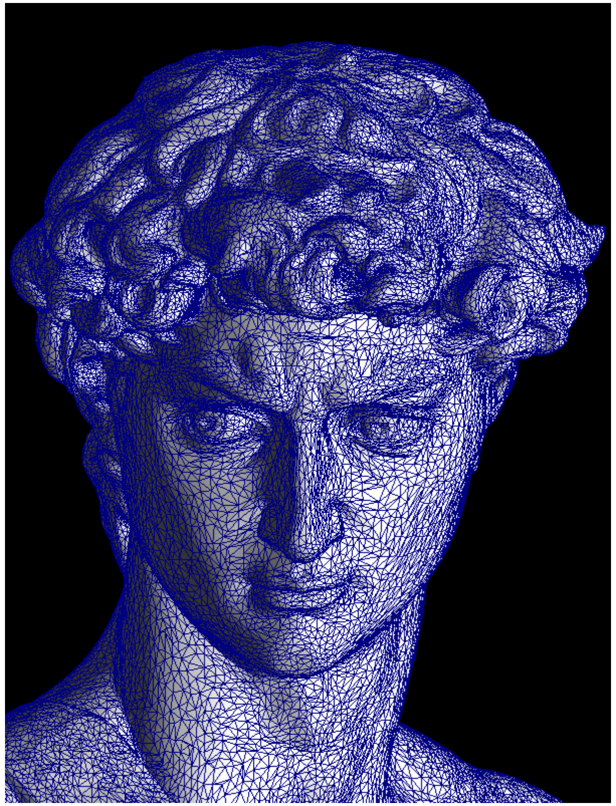
- Parametrically
  e.g. Bezier surface patch



- Smooth joins, very good for curved surfaces

Object Representation

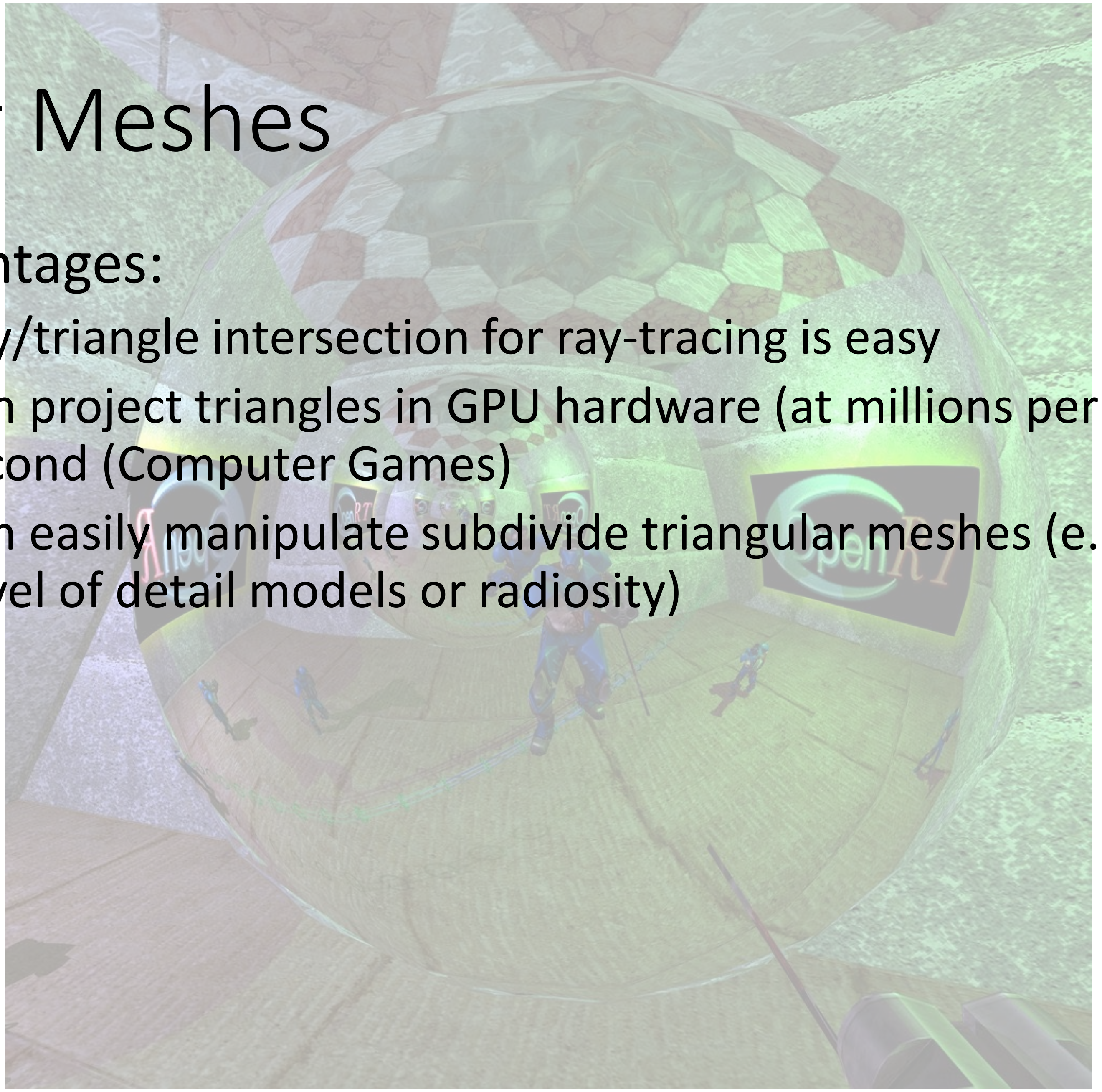
- Polygon meshes (e.g. triangular mesh)
- Bruno Levy (Inria) of Michaelangelo's David



2 billion polygons, 32GB

Triangular Meshes

- Advantages:
  - Ray/triangle intersection for ray-tracing is easy
  - Can project triangles in GPU hardware (at millions per second (Computer Games)
  - Can easily manipulate subdivide triangular meshes (e.g. for Level of detail models or radiosity)



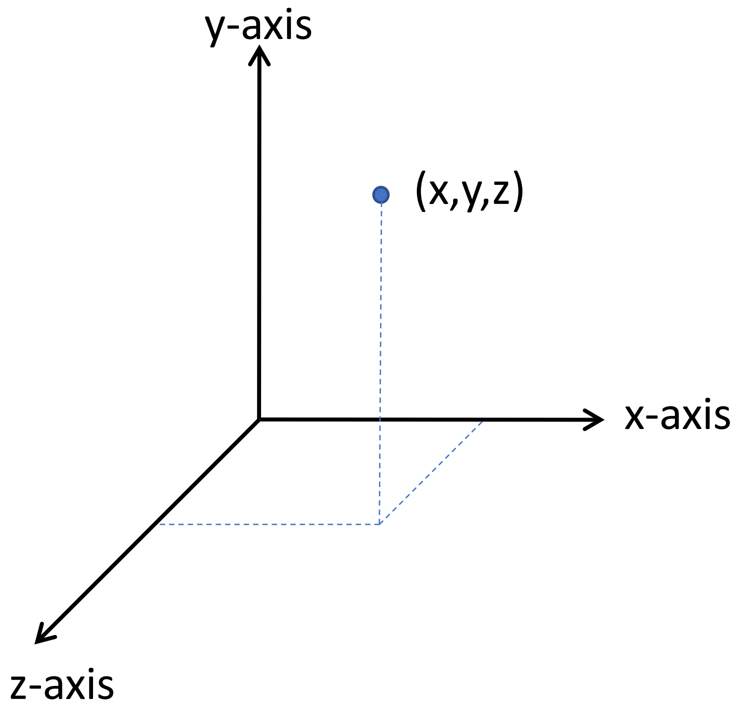
Terminology

- GPU (Graphics Processing Unit) – a chip dedicated to the processing of vertices for the purposes of display
- OpenGL – Open Graphics Language – a language for programming graphics applications. The language maps to GPU hardware and is OS independent
- Direct3D – Microsoft’s graphical programming language. The language maps to GPU hardware and is OS dependent (Windows)

Modelling: 3D Primitives

- Point
- 3D location in space
- Represented by coordinates

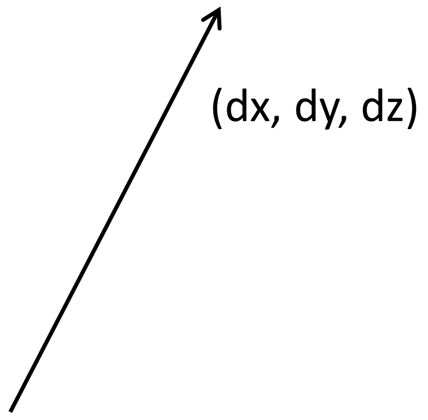
```
class Point {
private:
   float x, y, z;
...
};
```



Modelling: 3D Primitives

- Vector
- 3D direction and magnitude (no position)

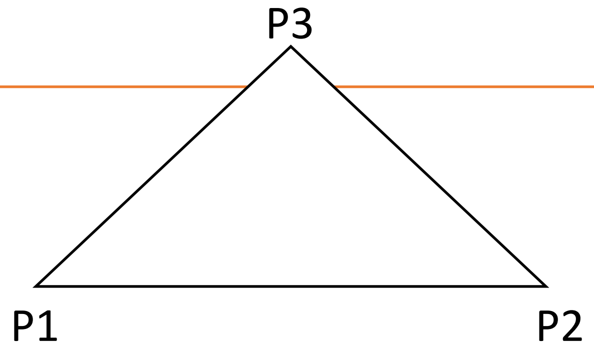
```
class Vector {
private:
   float dx, dy, dz;
public:
   float Magnitude() const {
      return
      sqrt(dx*dx+dy*dy+dz*dz));
   }
...
};
```



Modelling: 3D Primitives

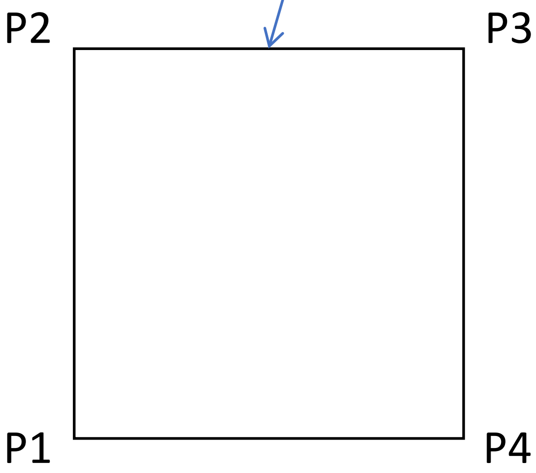
- Triangle
- These explicit representations lead to duplicated points
- This problem is examined in the next slides

```
class Triangle {
private:
   Point P1, P2, P3;
...
};
```



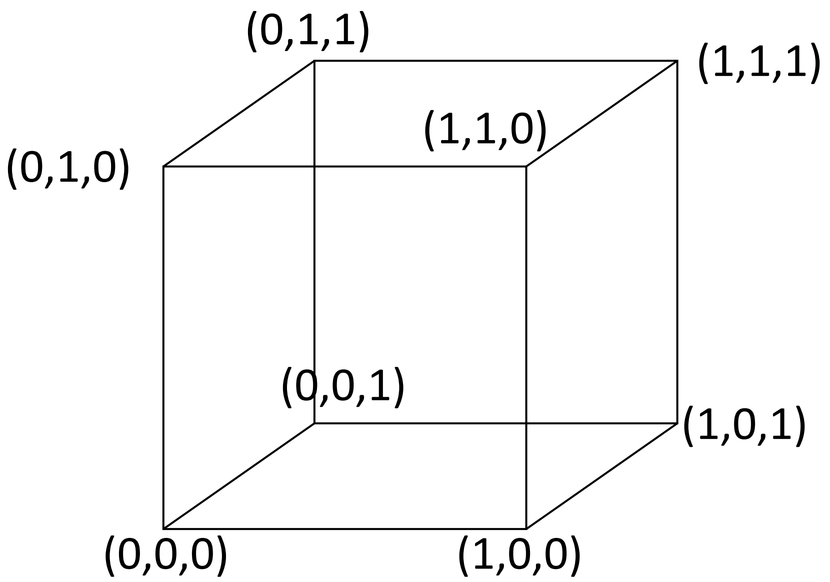
- Quad

```
class Quad{
private:
   Point P1, P2, P3, P4;
...
};
```



Explicit Representation

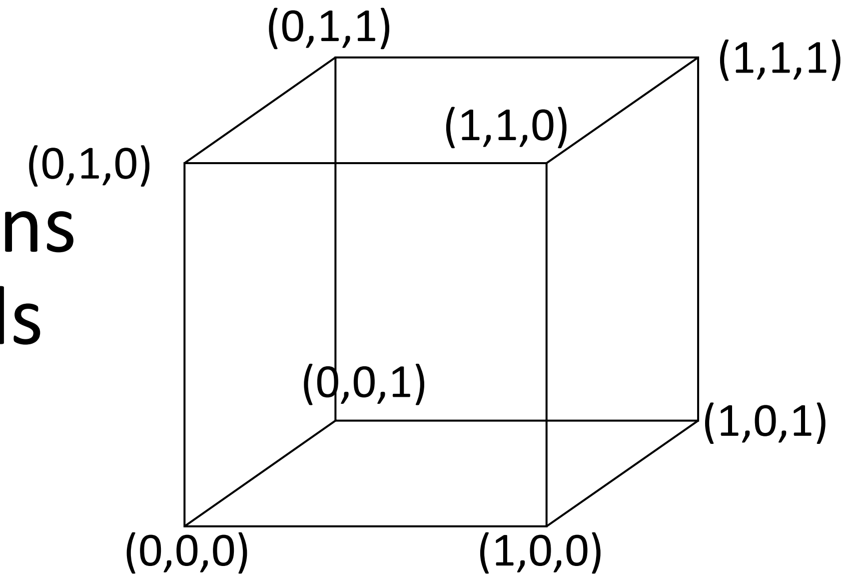
- Cube (6 faces / quads)
- (0,0,0) (0,0,1) (1,0,1) (1,0,0)
- (1,0,0) (1,1,0) (0,1,0) (0,0,0)
- (1,1,0) (1,1,1) (0,1,1) (0,1,0)
- (1,1,1) (1,0,1) (0,0,1) (0,1,1)
- (1,0,0) (1,0,1) (1,1,1) (1,1,0)
- (0,0,1) (0,0,0) (0,1,0) (0,1,1)
- Drawbacks:
- 3D transformations of 24 vertices (not 8)
- Draw 24 edges (rather than 12)
- Rounding errors – consider picking vertices



Pointers to Vertex List

- Vertices / Points
- 0=(0,0,0)
- 1=(0,0,1)
- 2=(0,1,0)
- 3=(0,1,1)
- 4=(1,0,0)
- 5=(1,0,1)
- 6=(1,1,0)
- 7=(1,1,1)

- Polygons / Quads
- 0 1 5 4
- 4 6 2 0
- 6 7 3 2
- 7 5 1 3
- 4 5 7 6
- 1 0 2 3



Advantages:

  3D transformations of just 8 vertices.

  Rounding errors not a problem.

Drawbacks:

  Draw 24 edges (rather than 12)

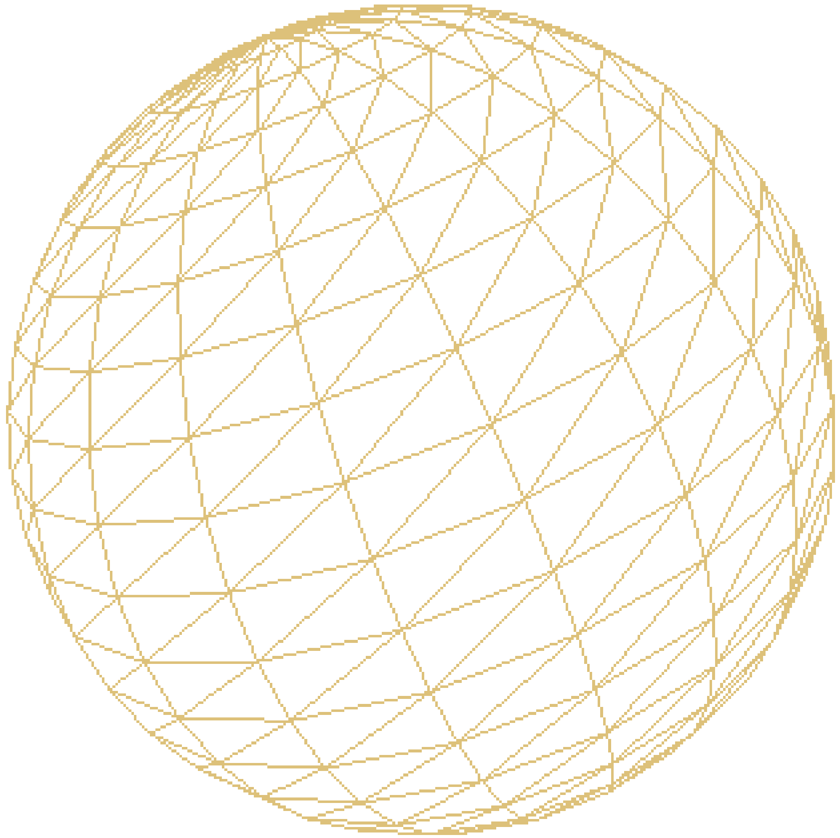
  Extra memory

  Extra processing during modelling

3D Primitives

- Pointers to Vertex List widely used (although see triangle strips)
- Each triangle vertex is a pointer to a 3D point
- An object is a list of triangles (or quads)

```
class Triangle {
private:
   Point *P1, *P2, *P3;
...
};
```

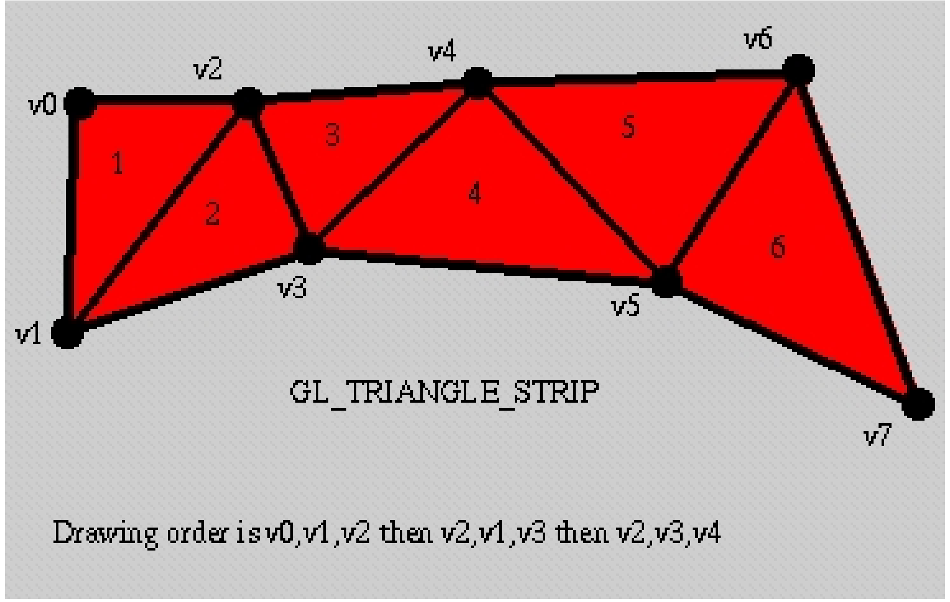


Example

- For example, the previous sphere consists of 382 vertices and 760 triangles
- Each vertex is 3 floats (3x4 bytes=12)
- For explicit representation:
  - 760 triangles x 3 vertices each x 12 bytes per vertex=27,360 bytes
- For pointers to vertex list:
  - Each triangle is a list of 3 pointers (3x4 bytes=12). 382x12 (vertex memory)+760x12 (pointers)= 13,704 bytes
- Using (next) triangular strip model uses 762x12=9,144 bytes

Triangular Strips

- Compact (n triangles represented using n+2 vertices)
- Therefore transmission to GPU is lower
- Very efficient when drawing (particularly in hardware)
- Can be hard to create triangle strips from arbitrary geometry

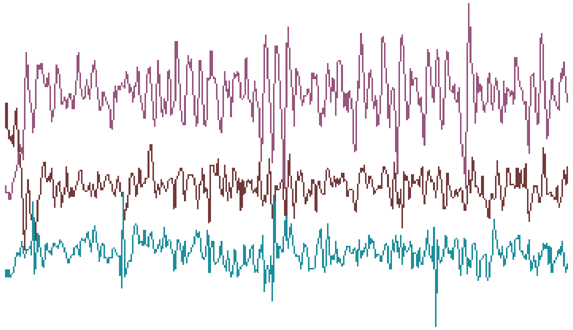


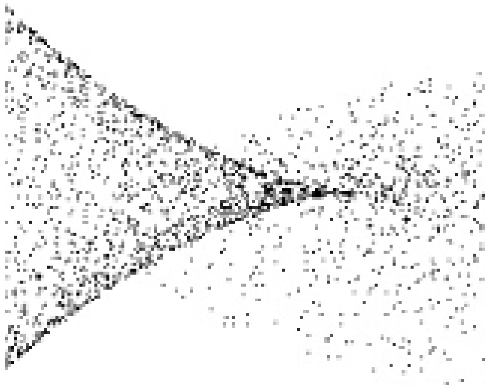
OpenGL

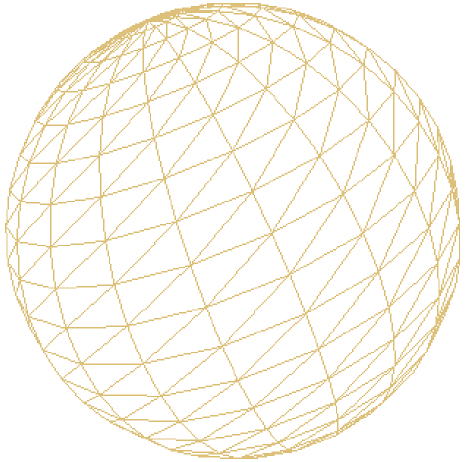
Direct3D Drawing Primitives

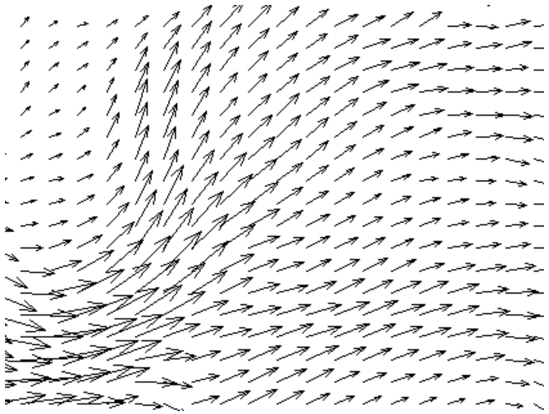
- D3D_POINTLIST
  A list of isolated points (n vertices=n points)
- D3D_LINELIST
  A list of isolated lines (each pair of points are the ends of a line) (2n vertices=n lines)
- D3D_LINESTRIP
  The vertices make a continuous line (n+1 vertices=n lines)
- D3D_TRIANGLELIST
  Each group of 3 points define an isolated triangle (3n vertices=n triangles)
- D3D_TRIANGLESTRIP
  (Previous slide) (n+2 vertices=n triangles)
- (Direct3D allows pointers to a vertex list using VERTEX BUFFERS)

What shall we use?
(Answers in lecture)

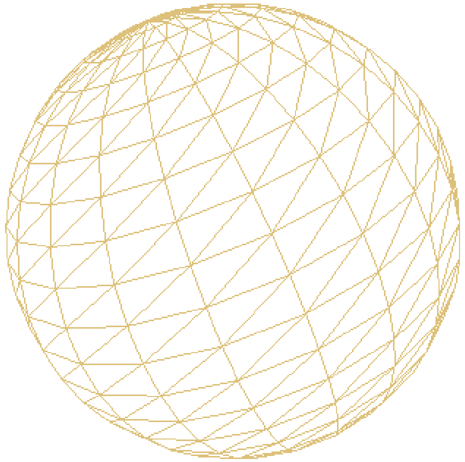








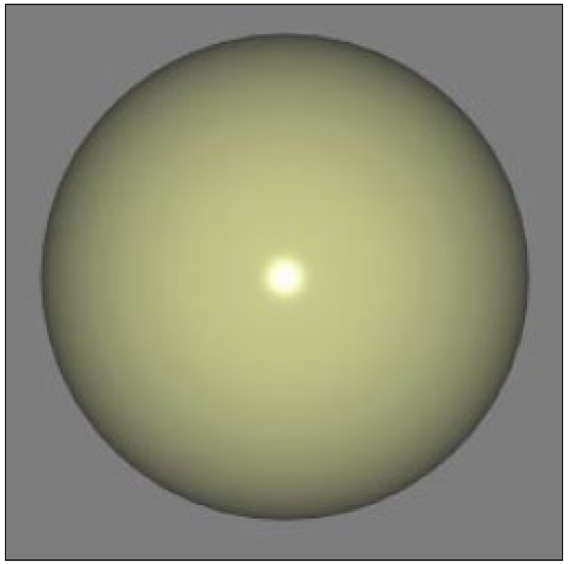
Rendering



Model / scene comprised of geometric primitives in 3D coordinate space

Rendering

Transformation of 3D space



Raster image