

Modeling for VR

February 9th 2009

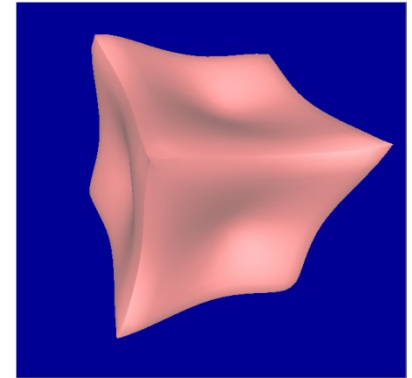
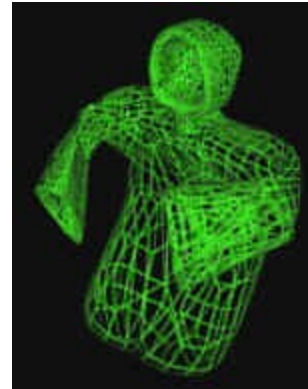
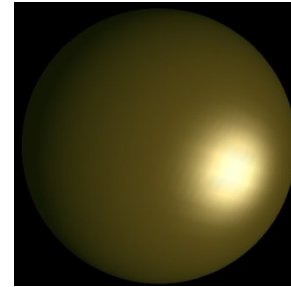
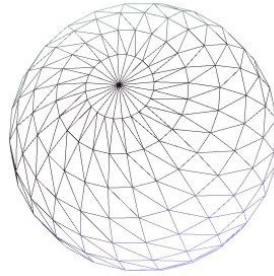
MAE 574, Virtual Reality Applications and
Research

Instructor: Govindarajan Srimathveeravalli

Steps to a VR model

- Geometric
- Material/Surface properties/Shading
- Texture
- Kinematics
- Physical
- Behavior

- Build it OR Import it
 - From scratch
 - CAD files
 - Digitizers
 - Purchase



Scanning Devices

- Multiple technologies
 - Hard probes – mechanical linkages
 - Touch trigger probes
 - Drag probes
 - Non contact trigger probes
 - Laser point triangulation
 - Structured light
 - Stereoscopy

Geometric Modeling



Venus de Milo created using the HyperSpace 3D digitizer, 4200 textured polygons using NuGraph toolkit

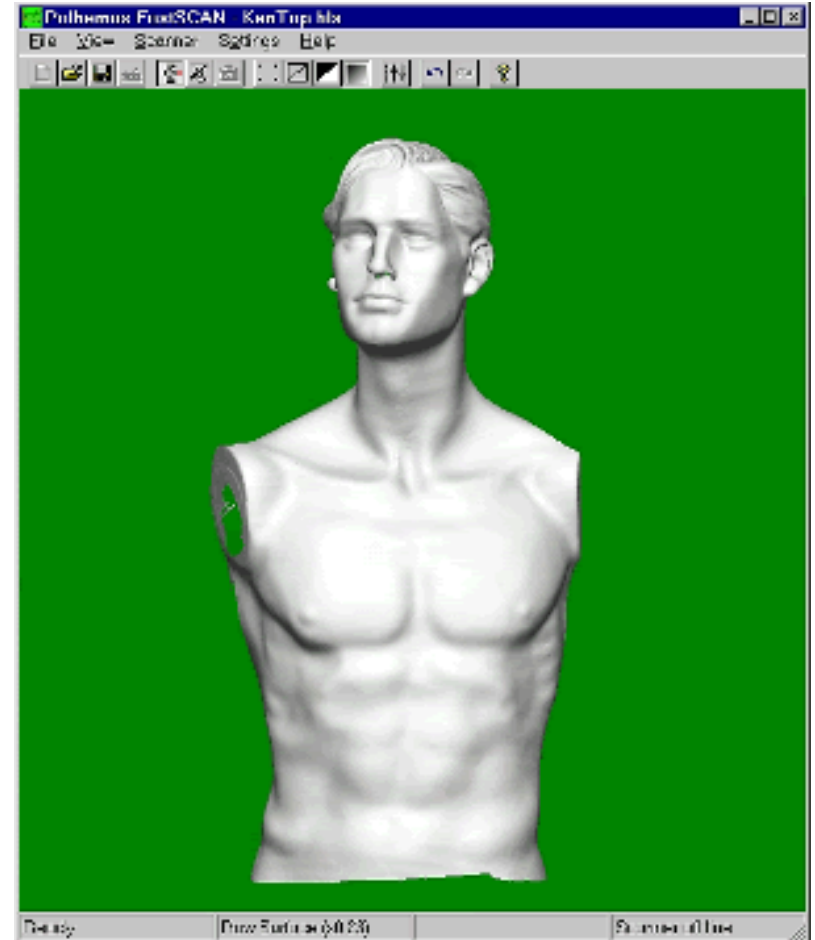


Polhemus 3-D scanners:

- ✓ Eliminate direct contact with object.
- ✓ uses two cameras, a laser, and magnetic trackers (if movable objects are scanned)
- ✓ Scanning resolution 0.5 mm at 200 mm range;
- ✓ Scanning speed is 50 lines/sec;
- ✓ Range is 75-680 mm scanner-object range

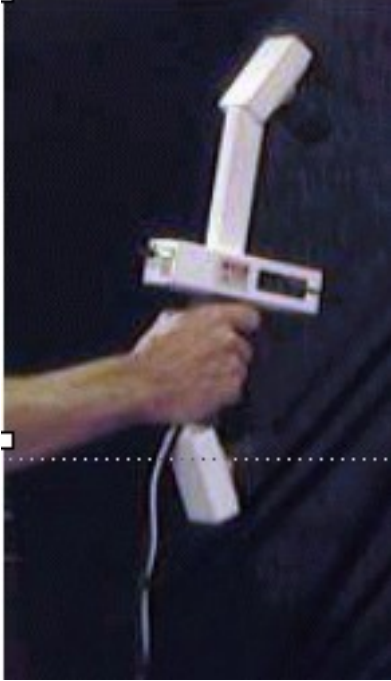


Geometric Modeling



Polhemus FastScan 3D scanner (can scan objects up to 3 m long).

Polhemus scanner



DeltaSphere 3000 3D scanner



Feature	Polhemus scanner	DeltaSphere scanner
Range	0.56 m	14.6 m
Resolution	0.5 mm @ 0.2 m	0.25 mm
Control	manual	automatic
Speed	50 lines/sec	25,000 samples/sec

DeltaSphere 3000 image



**DeltaSphere 3000
software-compensated
image**





Whole body scans – Cyberware

Microscribe –
Immersion



Laser Scanner
- Laser Design
Inc.



Class 10

Handheld Laser
Scanner – 3D
Scanner Inc.

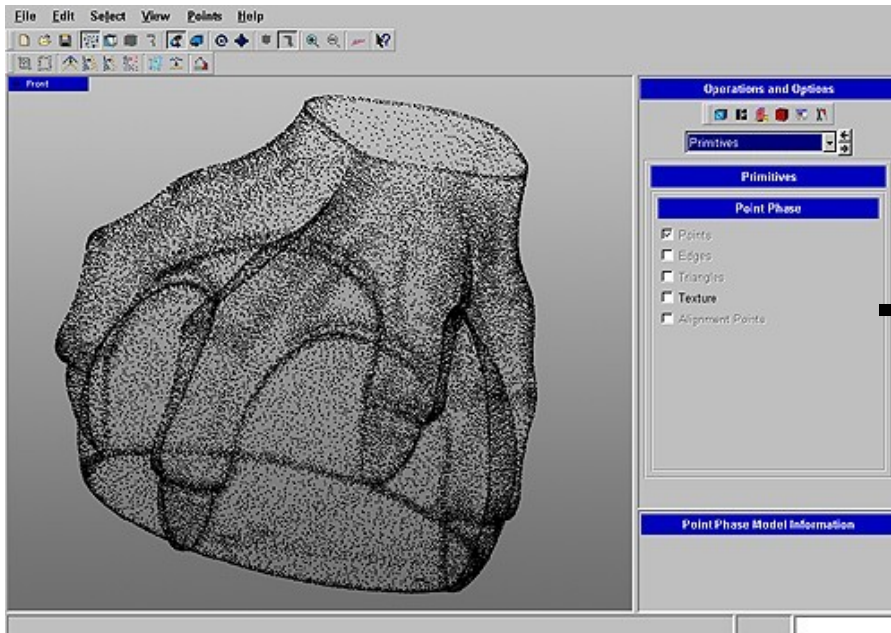


Govindarajan Srimathveeravalli,
VR Lab, University at Buffalo

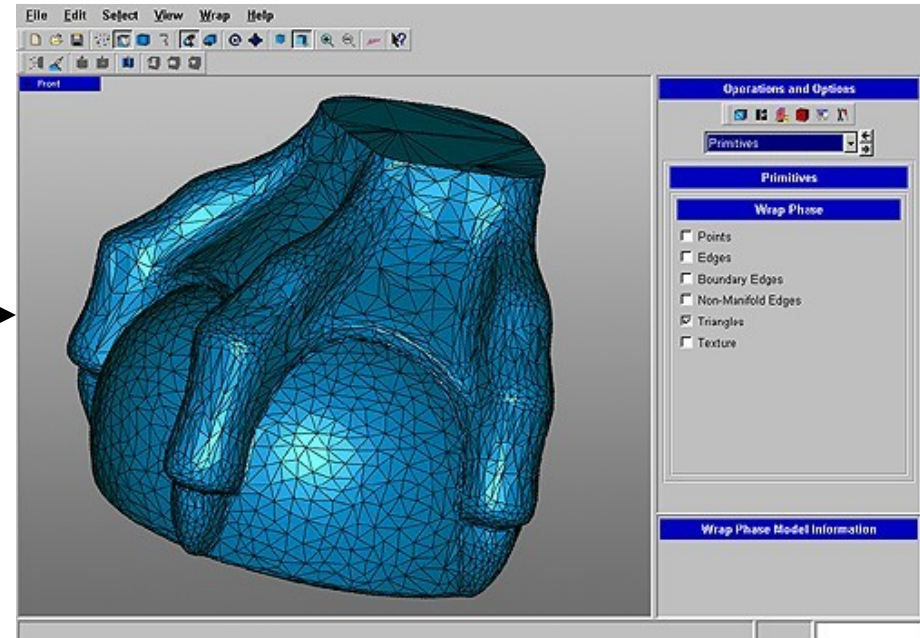
Conversion of scanner data:

- ✓ Scanners produce a dense “cloud” of vertices (x,y,z).
- ✓ Using such packages as *Wrap* (www.geomagic.com) the point data is transformed into surface data (including editing and decimation)

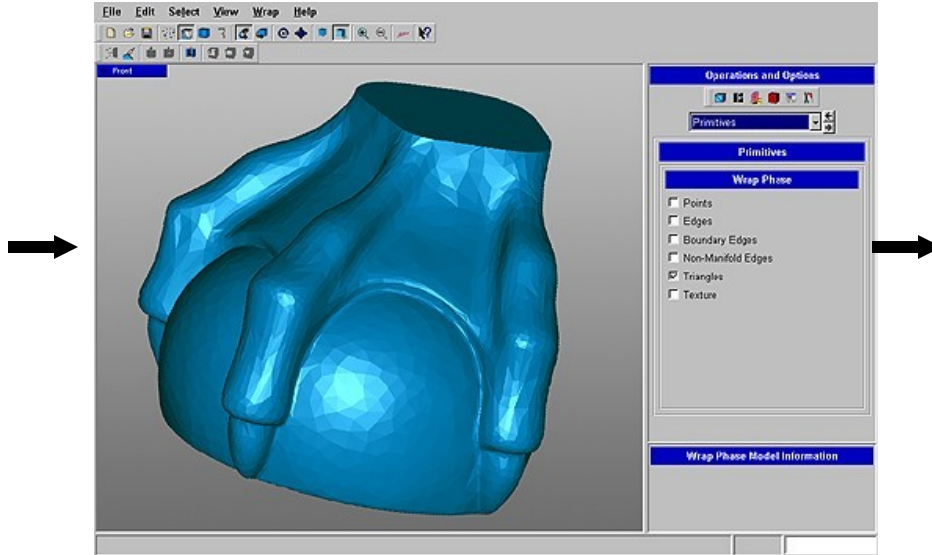
Point cloud from scanner



Polygonal mesh after decimation

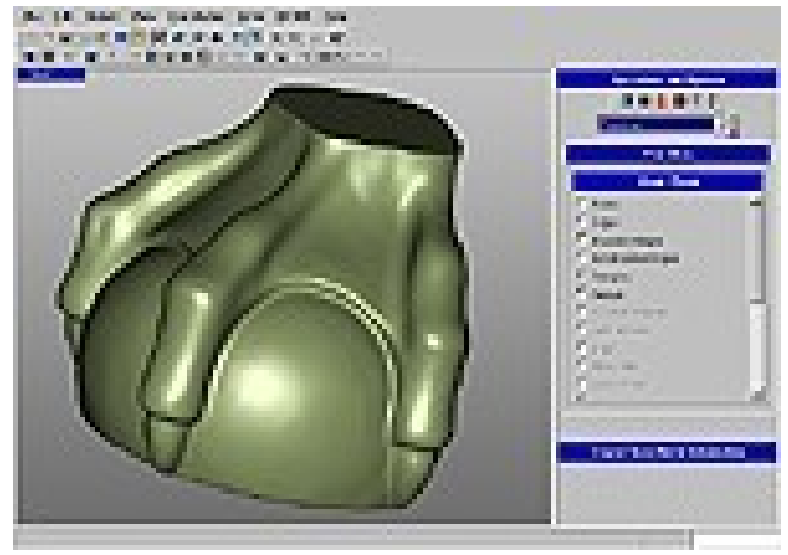
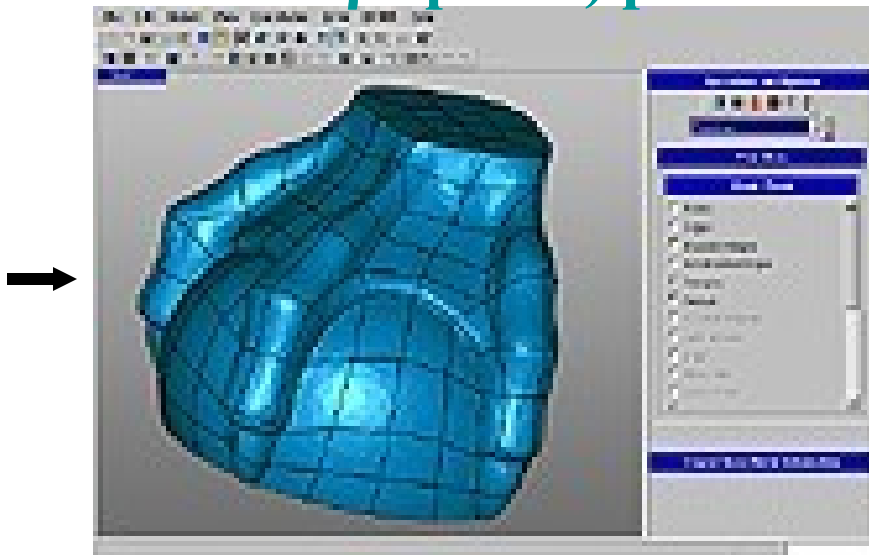


Polygonal surface



NURBS (non-uniform rational β -splines) patches

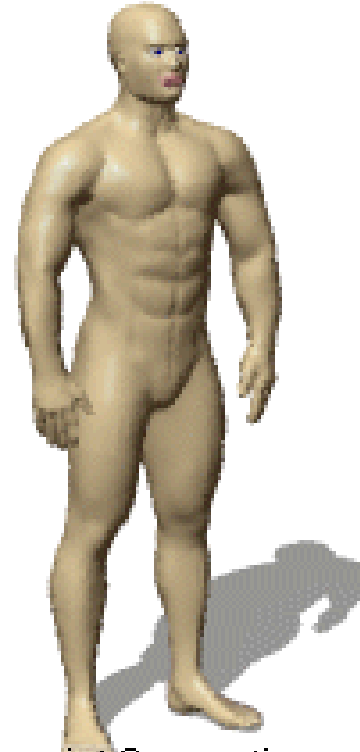
NURBS surface



Geometric Modeling – using online databases



Copyright © 2000 Viewpoint Corporation or its affiliates



2000 Viewpoint Corporation or its affiliates

**Low res. Model
– 600 polygons**

**Higher resolution model
> 20,000 polygons.**

Geometric Modeling

METHOD	FEATURE	SOURCE
Toolkit Editors	Tedious, requires skill	OpenGL, Starbase, PHIGS
CAD Programs	Interactive, existing technology	AutoCAD (Autodesk) 3-D Studio, etc.
3-D Digitizers	Interactive, Allows custom models	Autodesys Inc. Mira Imaging, Polhemus Inc., etc.
3-D Scanners	Fast multi-point acquisition Large objects	Polhemus Inc., Cyra Technologies, etc.
Commercial 3-D databases	Vertice list, connectivity, static model, level of detail	Viewpoint Inc., etc.

Geometry – Primary

- Points or Vectors ?
 - What vector ?
- Triangles
 - Normal
- Meshes
 - Triangle sets
 - Triangle strips
- Reading
 - STL
 - .OBJ
- Making your own
 - From point clouds

Triangles

- Easiest and simplest way to represent objects is as triangle based meshes.
- Each triangle needs to have
 - 3 vertex information
 - 1 normal information
 - Normal can also be calculated later on
 - Index information

```
for( unsigned int i=0; i< stlMesh.size(); i++)
{
    cVector dummy;
    glBegin( GL_TRIANGLES );

    dummy = stlMesh[i]->getNormal();
    glNormal3f( dummy.x, dummy.y, dummy.z );

    // vertex 1
    stlMesh[i]->getVertex( 1, dummy );
    glVertex3f( dummy.x, dummy.y, dummy.z );

    // vertex 1
    stlMesh[i]->getVertex( 2, dummy );
    glVertex3f( dummy.x, dummy.y, dummy.z );

    // vertex 1
    stlMesh[i]->getVertex( 3, dummy );
    glVertex3f( dummy.x, dummy.y, dummy.z );

    glEnd();
}
```

Code that actually draws the triangle

Vertex Arrays

- 3 steps to using arrays
 - Enable upto 6 arrays
 - Add Data
 - Draw
 - Access individual values
 - Access sequential values
- glEnableClientState to initialize and ready the arrays for use.
 - VERTEX, COLOR, INDEX, NORMAL, TEXTURE etc.
- Set a pointer to the array:- glVertexPointer(size, type, stride, pointer)
- Obtaining values :- glArrayElement(int)
 - Or glDrawElement(mode, count, type, indices)
 - Under indices list out values that need to be taken to draw.

```
// enable the vertex arrays
glEnableClientState( GL_VERTEX_ARRAY );
glEnableClientState( GL_NORMAL_ARRAY );

glVertexPointer( 3, GL_DOUBLE, 0, vertArr );
glNormalPointer( GL_DOUBLE, 0, normArr );
```

```
// vertex values, then we can render the same
vertArr = new double[ stlMesh.size()*9 ]; //
normArr = new double[ stlMesh.size()*3 ];
```

Display Lists

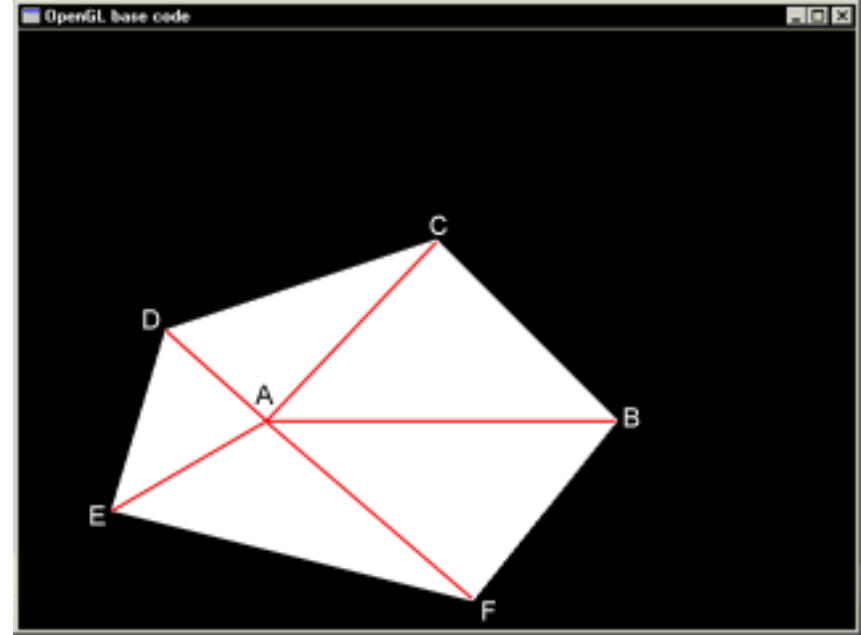
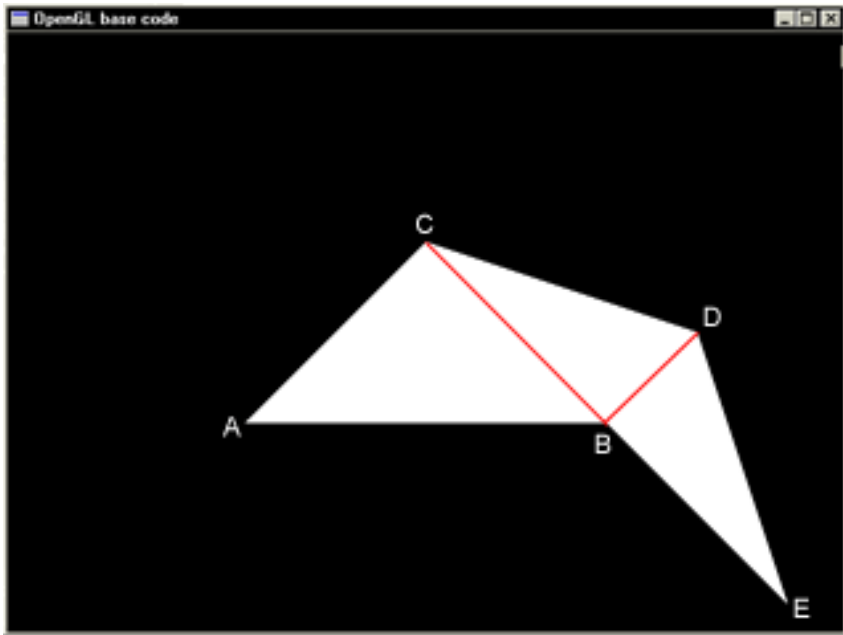
- Cache of drawing code for later rendering
- Very useful if object is to be replicated multiple times
- Create the list using `glNewList(value, GL_COMPILE)`
- Commands called between `glNewList()` and `glEndList`
- Execute the display list using `glCallList(value)`;
- All states set inside display list affect what's outside !
 - `glPushAttrib` , `glPopAttrib`
- A display list will not update values from outside (think key presses etc.)

```
// create the display list
aShape = glGenLists(1);
glNewList( aShape, GL_COMPILE );
    makeShape();
glEndList();

// Draw using display
glCallList( aShape );

glutSwapBuffers();
}
```

Triangle Strips and Fans



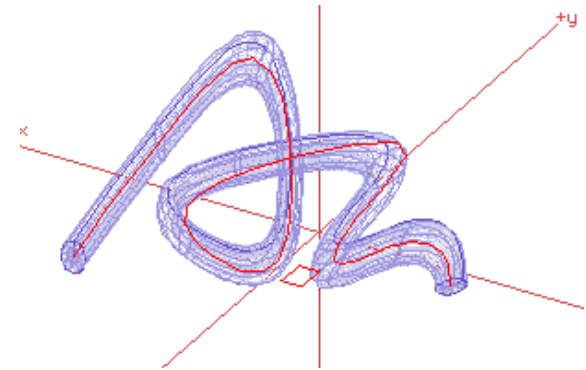
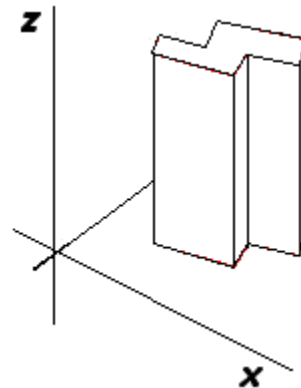
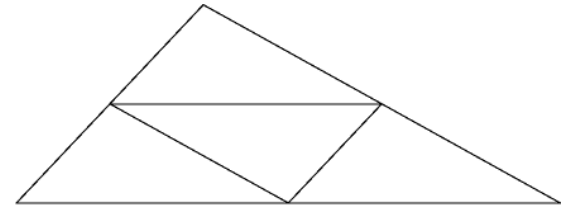
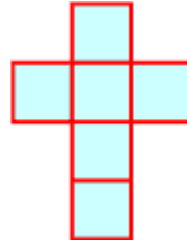
Images courtesy : Fallout Software

- First three vertices define the first triangle
- Subsequent triangles defined by two from previous and one new

- First three vertices define the first triangle
- Subsequent triangles defined by one from previous, one central and one new

Creating Objects

- Folding/Subdivision modeling
- Extrusions
- Sweeps
- Revolutes
- Equations
 - Implicit
 - Explicit
 - Parametric
- CSG



$$P(u, v) = (\sqrt{(1-v^2)} \cos(u), \sqrt{(1-v^2)} \sin(u), v)$$
$$v = [-1, 1] \quad u = [0, 2\pi]$$