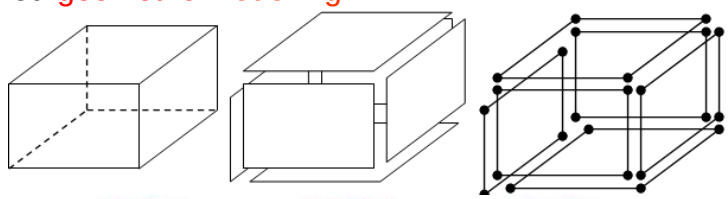
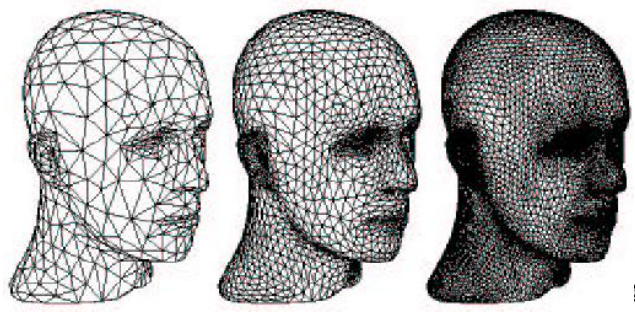
3D Graphics Programming Tools

Object modelling

Modelling

The generation of abstract descriptions of 3D objects is called geometric modelling.





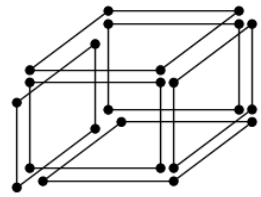
3D Graphics

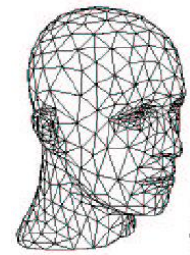
3D Graphics

Hierarchy in geometric models

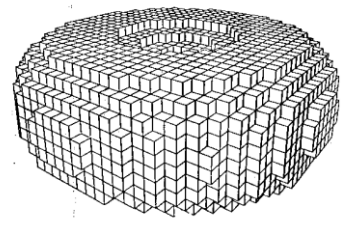
- Point-based

- Surface-based



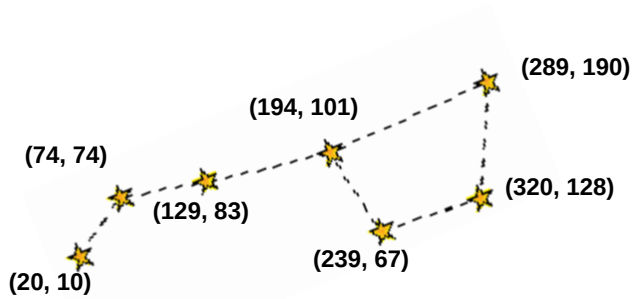


- Constructive



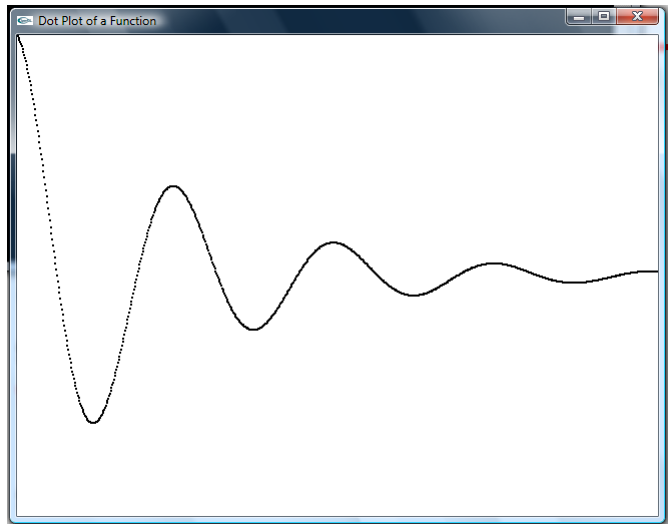
Queen Mary University of London

Point-based: the Big Dipper



Queen Mary University of London

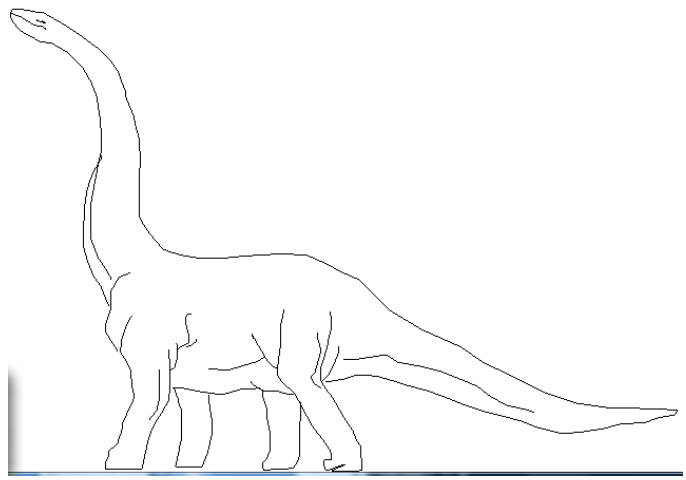
Dot plots



Queen Mary University of London

8

Drawing from a file

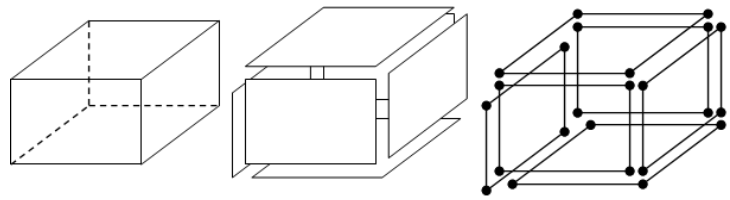


Queen Mary University of London

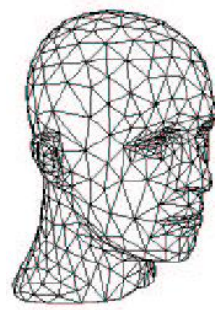
9

Surface modelling

Boundary representations (b-reps)



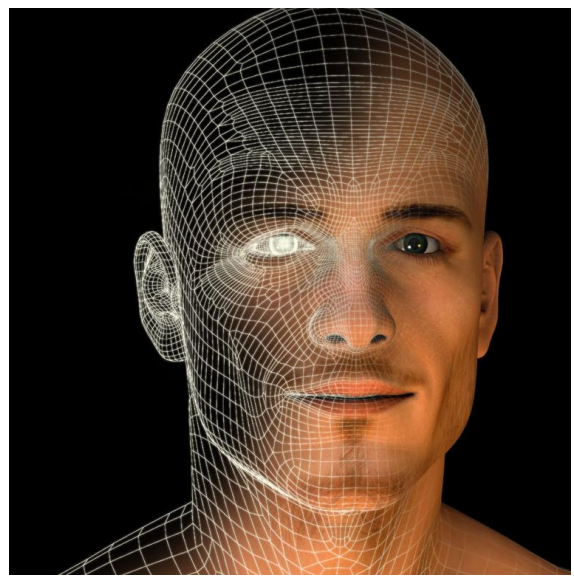
Polygon mesh



Queen Mary University of London

10

Polygon mesh



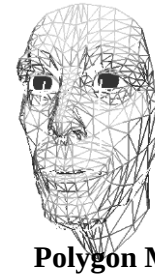
Queen Mary University of London

11

Representing polygon meshes

Polygon mesh (or "POLYHEDRON"):

- POLYHEDRON = geometric object with flat faces and straight edges
- Collection of polygons (faces), which are together connected to form the skin of the object
  - Faces
  - Edges → the boundary between faces
  - Vertices → the boundaries between edges, or where three or more faces meet
  - Normals, texture coordinates, colours, shading coefficients, etc



Polygon Mesh

Why polygons?

- They are:
  - Easy to represent
  - Easy to transform
  - Easy to draw
- Especially if they are:
  - Flat
  - Convex
  - Simple

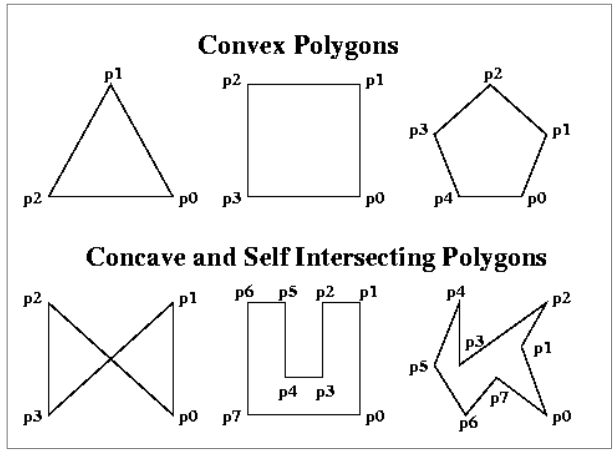
Polygon Mesh

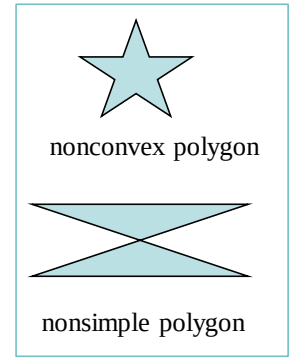
Polygon Issues

- OpenGL will only display polygons correctly that are
  - Simple: edges cannot cross
  - Convex: All points on line segment between two points in a polygon are also in the polygon
  - Flat: all vertices are in the same plane
- User program must check if above true
  - OpenGL will produce output if these conditions are violated but it may not be what is desired
- Triangles satisfy all conditions

Queen Mary University of London

Polygon Issues

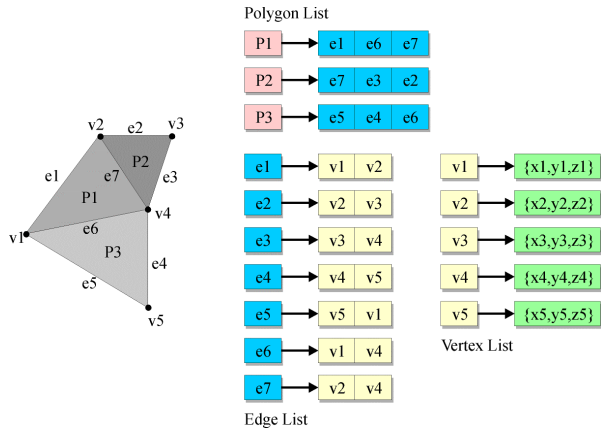




Queen Mary University of London

Representing polygon meshes

- Vertex list → locations of the vertices, geometric info
- Edge list → indexes into end vertices of edges, topological info
- Face list → indexes into vertices and normal lists, topological info
- Normal list → directions of the normal vectors, orientation info

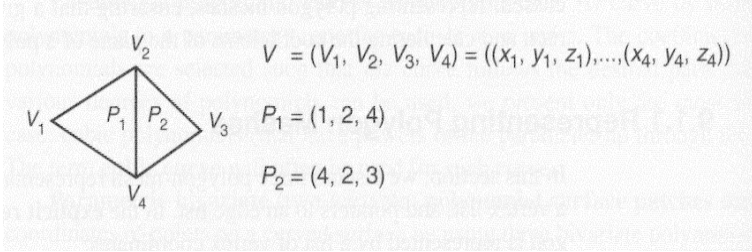


Queen Mary University of London

Representing polygon meshes

- Euler's Formula: $V - E + F = 2$
  - V: # of vertices
  - E: # of edges
  - F: # of faces

- Vertex list and face list are enough

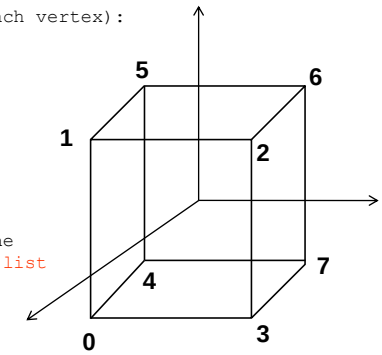


Queen Mary University of London

Polygon mesh example

A cube:

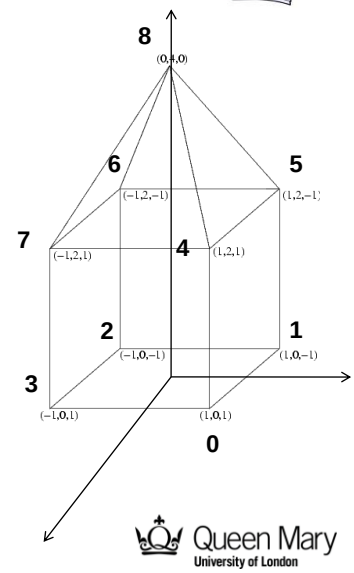
```
Coordinates of Vertices (3D, x, y, and z for each vertex):
-1 , -1 ,  1  # Vertex 0
-1 ,  1 ,  1  # Vertex 1
 1 ,  1 ,  1  # Vertex 2
 1 , -1 ,  1  # Vertex 3
-1 , -1 , -1  # Vertex 4
-1 ,  1 , -1  # Vertex 5
 1 ,  1 , -1  # Vertex 6
 1 , -1 , -1  # Vertex 7

Lists of 6 Faces (vertices are referenced to the
vertices above, -1 marks the end of the vertex list
of a face) :
0, 3, 2, 1, -1   # Front face=vertex 0,1,2,3
2, 3, 7, 6, -1   # Right side
3, 0, 4, 7, -1   # Bottom
1, 2, 6, 5, -1   # Top
4, 5, 6, 7, -1   # Back
5, 4, 0, 1, -1   # Left
```



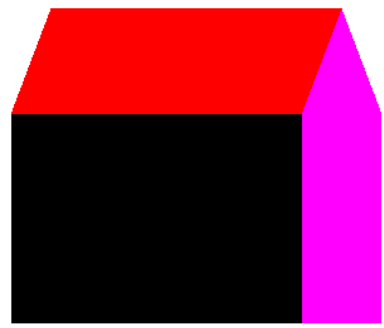
Polygon mesh example

A House:

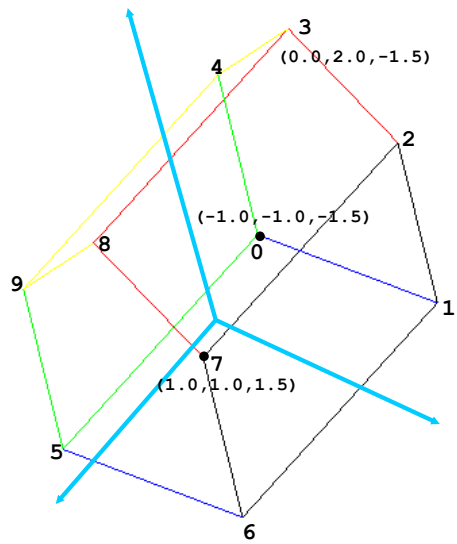
```
Coordinates of Vertices:    List of Faces?

 1   0    1     # Vertex 0
 1   0   -1     # Vertex 1
-1   0   -1     # Vertex 2
-1   0    1     # Vertex 3
 1   2    1     # Vertex 4
 1   2   -1     # Vertex 5
-1   2   -1     # Vertex 6
-1   2    1     # Vertex 7
 0   4    0     # Vertex 8
```



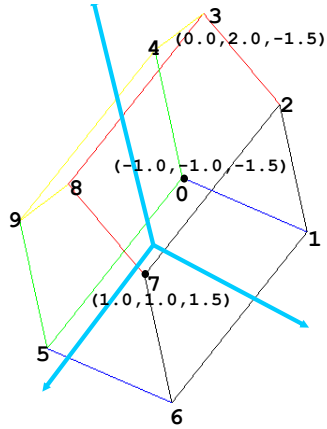
A simple barn



A simple barn



A simple barn



```
GLfloat GlobalVertices[][3] = { ? };
```

```
GLint BarnFaces[][6] = { ? };
```

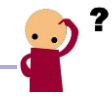

Drawing a polygon via a list of vertices

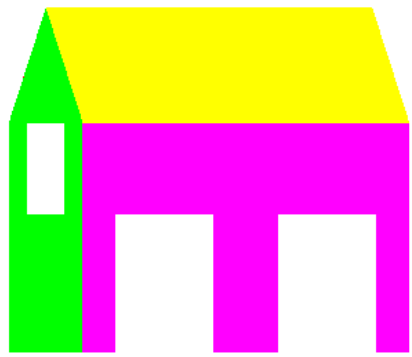
```
void a3dpolygon(GLfloat vertices[][3], GLint face[])
{
  /* draw a polygon (face) via list of vertices */
  int i = 0;
  int id;
  glShadeModel(GL_FLAT);
  glBegin(GL_POLYGON);
  while (face[i] > -1) {
          id = face[i];
          glVertex3fv(vertices[id]);
          i++;
  }
  glEnd();
}
```

A simple barn

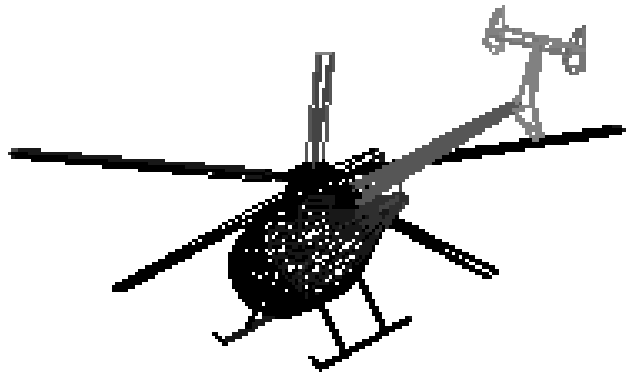
```
void barn()
{
  /* map vertices to faces */
  glColor3fv(colors[0]);
  a3dpolygon(BarnVertices, BarnFaces[0]);
  glColor3fv(colors[1]);
  a3dpolygon(BarnVertices, BarnFaces[1]);
  glColor3fv(colors[2]);
  a3dpolygon(BarnVertices, BarnFaces[2]);
  glColor3fv(colors[3]);
  a3dpolygon(BarnVertices, BarnFaces[3]);
  glColor3fv(colors[4]);
  a3dpolygon(BarnVertices, BarnFaces[4]);
  glColor3fv(colors[5]);
  a3dpolygon(BarnVertices, BarnFaces[5]);
  glColor3fv(colors[6]);
  a3dpolygon(BarnVertices, BarnFaces[6]);
}
```


A barn with windows …





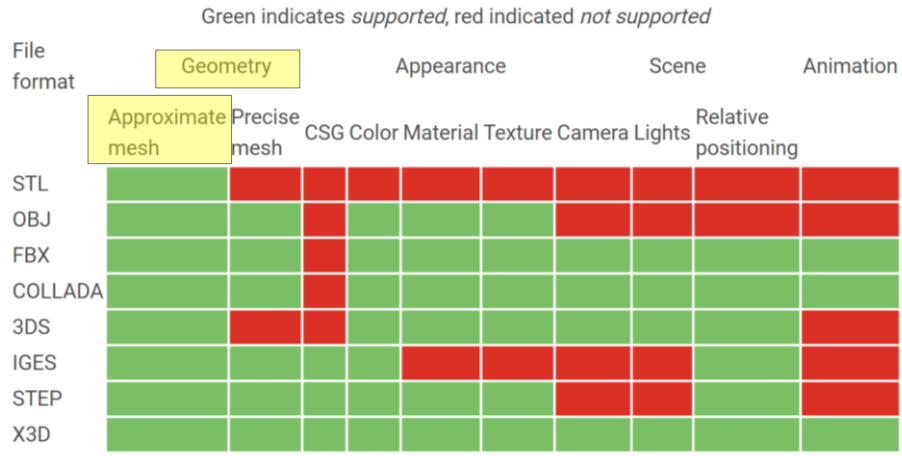
Polygon meshes



3D File Formats

- The basic purpose of a 3D file format is to store information about 3D models as plain text or binary data.
- There are hundreds of 3D file formats currently being used!
- Software manufacturers such as AutoDesk and Blender have their own proprietary format which is optimized for their software.
- To solve the problem of interoperability, *neutral* or *open source* formats were invented as intermediate formats for converting between two proprietary formats.
- Some popular formats: STL, OBJ, FBX, COLLADA, VRML and X3D**,** etc.

3D File Formats



STL (STereoLithography)

- STL is one of the most important neutral 3D file formats in the domain of 3D printing, rapid prototyping, and computer aided manufacturing.
- STL encodes the surface geometry of a 3D model approximately using a triangular mesh.

OBJ

- The OBJ file format is another neutral format widely used in the fields of 3D printing and 3D graphics.
- The OBJ file format supports both approximate and precise encoding of surface geometry (i.e. smooth curves and surfaces such as NURBS: *Non-Uniform Rational B-Spline)*, instead of polygons.
- When using the approximate encoding, it doesn't restrict the surface mesh to triangular facets (users can use polygons like Quadrilaterals).

FBX

- FBX is a proprietary file format which is widely used in the film industry and video games.
- Used as an exchange format which facilitates high fidelity exchange between 3DS Max, Maya, MotionBuilder, Mudbox and other proprietary software.

COLLADA

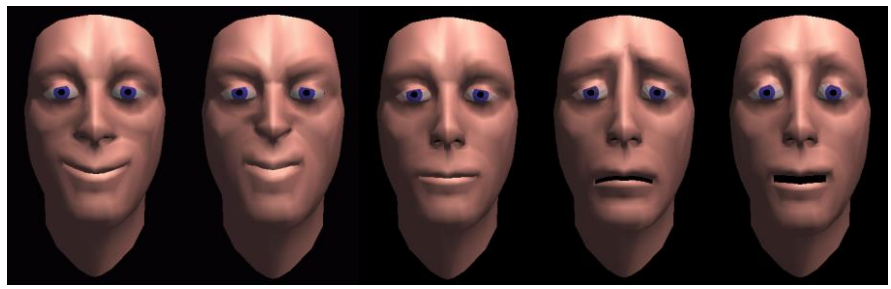
- Collada is a neutral file format used heavily in the video game and film industry.
- The file extension for the COLLADA format is .DAE
- The COLLADA format supports geometry, appearance related properties like color, material, textures, and animation.
- The COLLADA format stores data using the XML markup language.

VRML and X3D

- VRML stands for Virtual Reality Modeling Language.
- It is a 3D file format that was developed for the web.
- The VRML format uses a polygonal mesh to encode surface geometry and can store appearance related information such as color, texture, transparency etc.
- It has been succeeded by X3D.
- X3D is an XML based 3D file format.
- The X3D format adds NURBS encoding of the surface geometry, the capability of storing scene related information and support for animation.
- The goal of X3D is to become the standard 3D file format for the web, but it has not yet received wide acceptance.

Building a face mesh

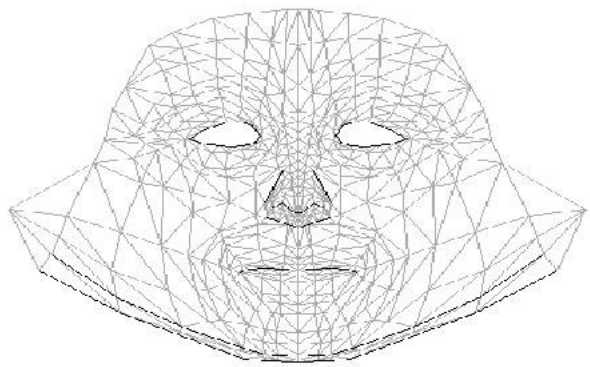
- Number of nodes
- Computational cost



by: ***Cem Yuksel****, BUPAM Bogazici University Pattern Analysis and Machine Vision Laboratory*
EBU5405

34

Generic Face Mesh and Mesh Adaptation



Advantages:

• Well-defined features

• Efficient Triangulation

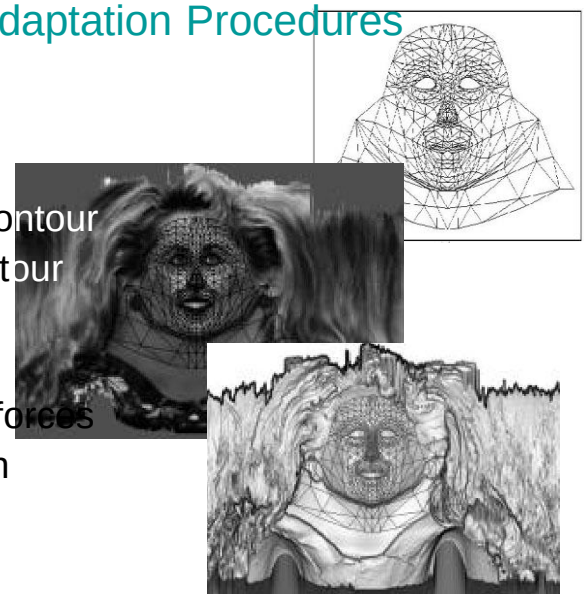
K. Waters. A muscle model for animating threedimensional facial expression. *Computer Graphics*, 1987.

EBU5405

35

Mesh Adaptation Procedures

1. Locate noise tip
2. Locate chin tip
3. Locate mouth contour
4. Locate chin contour
5. Locate ears
6. Locate eyes
7. Activate spring forces
8. Adapt hair mesh
9. Conform to 3D



*by: **Cem Yuksel**, BUPAM Bogazici University Pattern Analysis and Machine Vision Laboratory*