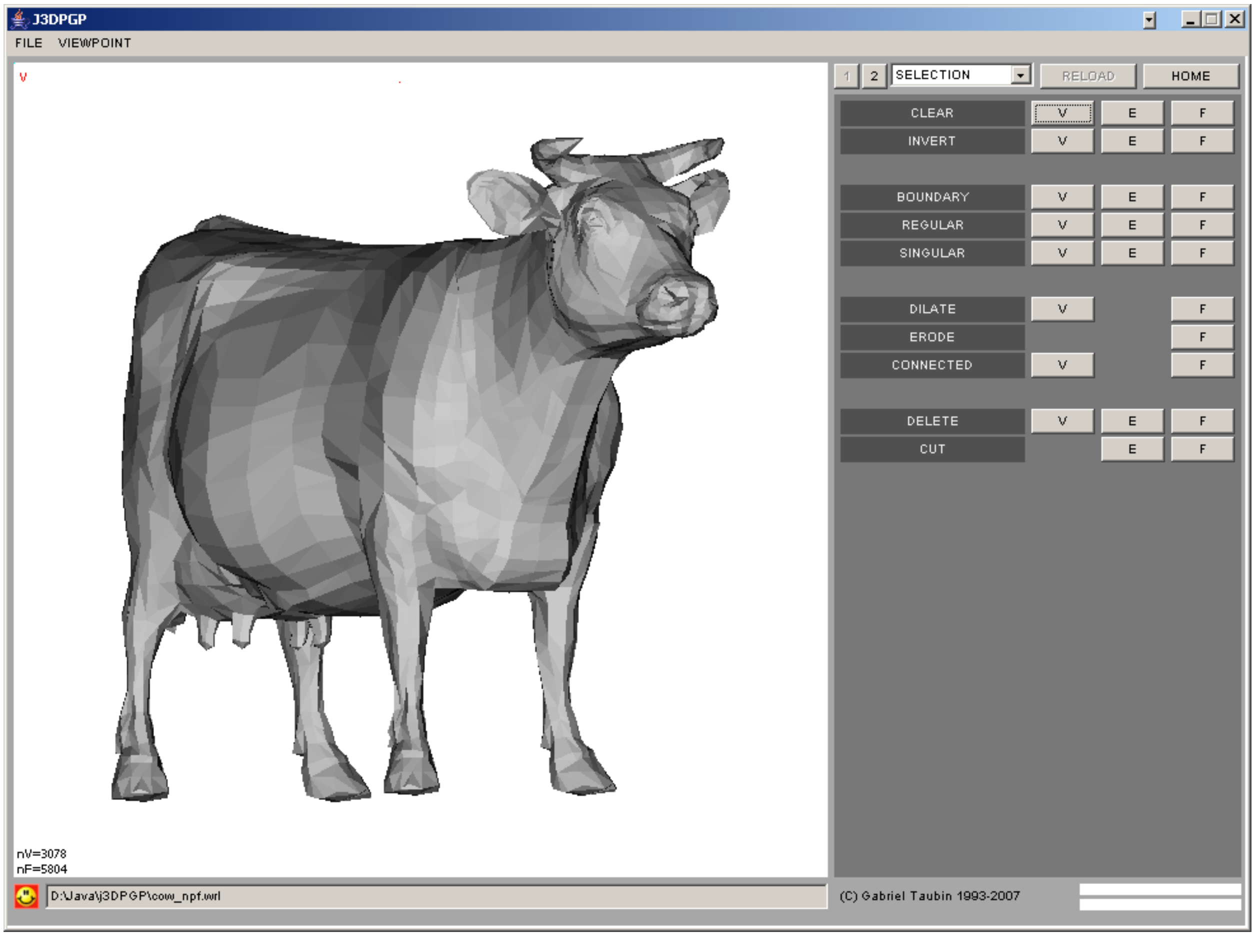
The j3DPGP Application 2008



nV=3078
nF=5804

D:\Java\j3DPGP\cow_npf.wrl

(C) Gabriel Taubin 1993-2007

This application will be used for several software assignments. It provides basic functionality to operate on polygon meshes. It can read a polygon mesh from a file, edit it, and write the modified polygon mesh to a file. In the assignments you will extend the application by writing a number of geometry processing and 3D shape capture modules. For example, in the first assignment you will implement the functionality corresponding to the pushbuttons labeled V, E, and F in the screenshot shown above.

#VRML V2.0 utf8
Shape {
 geometry IndexedFaceSet {
 …
 }
}
	#VRML V2.0 utf8
	Shape {
	 appearance Appearance {
	  material Material {
	   …
	  }
	 }
	 geometry IndexedFaceSet {
	 …
	 }
	}
		#VRML V2.0 utf8
		Shape {
		 appearance Appearance {
		  texture ImageTexture {
		   url "xxx.jpg"
		  }
		 }
		 geometry IndexedFaceSet {
		 …
		 }
		}

The main internal data structure is the **WrlSceneGraph** class, which roughly corresponds to a VRML'97 scene graph. We are interested in a scene graph with one or more **Shape** nodes; each one containing an **IndexedFaceSet** node in their `geometry` field, and a **Material** node or an **ImageTexture** node in the `appearance` field. The internal data structures corresponding to these nodes are `WrlShape`, `WrlIndexedFaceSet`, `WrlMaterial`, and `WrlImageTexture`. The previous table shows some examples of VRML'97 files that this application can parse, as well as the format of the files written out.

In these three cases j3DPGP accepts any valid syntax with the **IndexedFaceSet** node, although some fields of these nodes are ignored by the rendering engine. The third case corresponds to textured meshes. In this case the `url` field of the **ImageTexture** node must be a JPEG file stored in the same directory as the VRML file.

Installation

For the first assignment you will receive the basic application in j3DPGP-A1.zip. To install the application you only need to unzip it in an empty directory. You should have the following files

```
J3DDesktopPanels.java
J3DPanel.java
J3DPanelColors.java
J3DPanelOptions.java
J3DPanelSceneGraph.java
J3DPanelSelection.java
Makefile
cow_npf.wrl
cow_npv.wrl
j3DPGPapp.zip
j3DPGPlib2.zip
j3dpgp.bash
j3dpgp.lnk
```

as well as a directory named

```
doc
```

containing files describing the public interface to some of the classes.

To recompile the source code and to run the application you will need the latest Java compiler and interpreter, which are included in the Java SE Development Kit (JDK). You can download the latest JDK from http://java.sun.com/javase/.

If you work in a Windows environment, you can also install Cygwin from http://www.cygwin.com/. This gives you a Unix-like development environment, including `make` which we use to automate compilation. An alternative to is to install an interactive development environment (IDE) such as Eclipse, which you can download from http://www.eclipse.org/.

Running the Application

To run the application on a Windows machine, double-click `j3dpgp.lnk` within Windows Explorer. If it doesn't work on the first try, you need to edit the link properties. Right-click on the link in windows explorer, select properties, and set the "Start in:" field to the name of the directory where the two zip files are stored.

On a Linux machine, or in a Windows-Cygwin environment, open a bash shell; change directory to the location where you unzipped the files, and run `j3dpgp.bash. Alternatively, at the command line run`

```
> javaw -cp "j3DPGPapp.zip;j3DPGPlib.zip" J3DPGP -w 975 -h 675
```

In a Linux environment you need to run `java` instead of `javaw,` and probably use ":" as a separator in the classpath instead of ";". The Java interpreter allocates a fixed amount of memory to each application. The default is usually not enough for large meshes. You can specify the amount of memory to be allocated to the application on the command line. For example, the following command will run the application with 1GB of (virtual) memory.

```
> javaw -Xmx1000M -cp "j3DPGPapp.zip;j3DPGPlib.zip" J3DPGP -w 975 -h 675
```

For your first assignment, you will modify the file `J3DPanelSelection.java and` recompile `j3DPGPapp.zip. Afterwards,` you can run the application with your extensions as described above.

User Interface

Once the application is up and running, you should see two main panes, as in the screen dump shown above. The rendering panel is located on the left-hand side and the command panel is on the right-hand side. (Initially, the rendering panel will be empty.) The application actually supports multiple command panels, which can be selected from the drop-down menu (showing SELECTION in the screen capture). In the first assignment you will modify the panel defined in the file `J3DPanelSelection.java`. In subsequent assignments you will write new panels. We can reduce clutter and group functionality in a logical manner using these panels; at the same time, we'll isolate the new code that we write from the core application.

Loading and saving data files

You can load a data file in two ways; you can drag and drop a file onto the rendering panel, or you can select LOAD from the FILE menu in the application menu bar.

After the data has been edited, you can also save the data in two ways. If you select SAVE from the FILE menu in the application menu bar, the data will overwrite the original file. The application will not ask any questions. If you don't have a backup copy of the original input file, you will loose it. You can also select FILE→ SAVE AS → WRL, which will let you choose a new name for the output file.

Exporting images

j3DPGP can export the currently rendered frame as either a JPEG or EPS file. This functionality will be useful for documenting your results in the homework assignments. To save a JPEG image, select FILE→ EXPORT → IMG JPEG. Similarly, to save an EPS file, select FILE→ EXPORT → IMG EPS. Note that the output images are rendered using the fixed size of the render frame. To increase the resolution you must increase the j3DPGP window size at runtime.

Navigation

The rendering panel is divided into 3x3 logical regions. The vertical middle and horizontal center bands are wider than the corner regions. If you click and drag in the middle-center region, the object rotates in 3D. If you click and drag on the right-middle or right-upper regions, the camera zooms in and out. If you click and drag on any of the three left regions or the lower-center region, the object translates according to the translation of the mouse with respect to the initial click. In all of these cases the light sources move with the object. If you click and drag on the upper-center region, the light sources rotate with respect to the object (but the object remains in a fixed position). The rendering engine is rather simple; it assumes that light sources are located at infinity and neglects shadows.

Rendering options

If you right click on the rendering panel a pop-up menu becomes visible. The first two options are RENDER and PAINT. Each one has an associated submenu which allows you to control how the mesh is rendered and what parts and associated properties of the mesh are rendered.

Selection

The third option in the rendering panel pop-up menu is SELECT. If you choose VERTICES, EDGES, or FACES in the SELECT submenu, you can select subsets of vertices, edges, and/or faces for the current mesh. These selected elements can be used as additional inputs to the algorithms that you will write. To add to the current selection, first press the SHIFT key and then click and drag on the rendering panel. A red selection box will be drawn, and all the elements (vertices, edges, or faces) contained within or intersected by the box will be selected. The selected elements are rendered with a different color than the unselected elements. To subtract from the selection, first press the CTRL key and then click and drag on the rendering panel. To make more precise selections you may want to zoom in to operate on a small area, then change the viewpoint and continue adding or removing from the selection. (Note that you must release the mouse button before SHIFT or CTRL, otherwise no objects will be selected.)

Compilation

In the first assignment, you will edit the file `J3DPanelselection.java`. To compile just run `make` from the command line (assuming a Linux or Cygwin environment with the bash shell). If you create a new panel, you will also need to modify `J3DDesktopPanels.java` and the `Makefile`.

VRML'97

The Virtual Reality Modeling Language (VRML'97) is the International Standard ISO/IEC 14772-1:1997. The specification can be found here

`http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/`

j3DPGP only supports a very limited subset of the VRML'97 standard. However, the files that this application can parse and those that it writes are valid VRML'97 files. The VRML standard has been superseded by the X3D standard. X3D is the enhanced successor to VRML, however the VRML'97 specification and many VRML tools are still very useful and will remain so while developers update their products to support X3D.

As described above, an input file should be a valid VRML'97 file containing one `Shape` node, which comprises two fields: an `appearance` field and a `geometry` field. Either one of the two fields can be missing in the file, in which case they are assigned the default value `NULL`. However, a `Shape` node with a `NULL` `geometry` field will not be very useful for our purposes.

```
Shape {
  SFNode appearance NULL
  SFNode geometry   NULL
}
```

If not `NULL`, the `appearance` field of the `Shape` node should contain an `Appearance` node.

```
Appearance {
  SFNode material          NULL
  SFNode texture           NULL
  SFNode textureTransform  NULL
}
```

Again, any of the three fields of the `Appearance` node may be missing in the file, in which case it will be assigned the default value `NULL`. If the `material` field of an `Appearance` node is not `NULL`, then it must contain a `Material` node

```
Material {
  SFFloat ambientIntensity  0.2         # [0,1]
  SFColor diffuseColor      0.8 0.8 0.8 # [0,1]
  SFColor emissiveColor     0 0 0       # [0,1]
  SFFloat shininess         0.2         # [0,1]
  SFColor specularColor     0 0 0       # [0,1]
  SFFloat transparency      0           # [0,1]
}
```

In j3DPGP, if the `texture` field of an `Appearance` node is not `NULL`, then it must contain an `ImageTexture` node, and the `url` field of the `ImageTexture` node should contain the name of a JPEG file stored in the same directory as the VRML file.

```
ImageTexture {
  MFString url     []
  SFBool   repeatS TRUE
  SFBool   repeatT TRUE
}
```

In j3DPGP, if the `geometry` field of a `Shape` node is not `NULL`, then it must contain an `IndexedFaceSet` node, which is where polygon meshes can be represented

```
IndexedFaceSet {
  SFBool  ccw              TRUE
  SFBool  convex           TRUE
  SFFloat creaseAngle      0         # [0,∞)
  SFBool  solid            TRUE
  MFInt32 coordIndex       []        # [-1,∞)
  SFNode  coord            NULL
  SFBool  colorPerVertex   TRUE
  MFInt32 colorIndex       []        # [-1,∞)
  SFNode  color            NULL
  SFBool  normalPerVertex  TRUE
  SFNode  normal           NULL
  MFInt32 normalIndex      []        # [-1,∞)
  SFNode  texCoord         NULL
  MFInt32 texCoordIndex    []        # [-1,∞)
}
```

All the valid VRML'97 property bindings are supported in j3DPGP.

j3DPGP Data Structures

The fundamental data structure in j3DPGP is the **`WrlSceneGraph`** class, which corresponds to the information contained in an input file as described above (i.e, a file containing one or more `Shape` nodes, each with an `Appearance` node in its `appearance` field and an `IndexedfaceSet` node in its `geometry` field, and with the `Appearance` node having a `Material` node in its `material` field and an `ImageTexture` node in its `texture` field). Nodes not present in the input file receive default values when parsed. On output, fields and nodes with default values are not written in the output file. The following is the public interface to the **`Ifs`** class. It includes a number of other classes, which we proceed to describe.

```
public class WrlSceneGraph
{
  public              WrlSceneGraph();
  public void         erase();
  public String       getFileName();
  public void         setFileName(String fileName);
  public void         setDone(boolean value);
  public boolean      getDone();
  public void         waitUntilDone();
  public boolean      getHasChanged();
  public void         setHasChanged(boolean value);
  public void         def(String name, WrlNode defNode);
  public WrlNode      use(String name);
  public String       toString();
  public String       toString(boolean withDetails);
  public void         write(PrintWriter out);
  public int          getNumberOfShapes();
  public void         updateBBoxes();
  public void         showAll();
  public void         showInvert();
  public int          getNumberOfCoord();
  public int          getNumberOfFaces();
  public boolean      isTextured();
  public boolean      hasColors();
  public boolean      hasNormals();
  public boolean      hasFaces();
  public boolean      hasEdges();
  public boolean      hasMaterials();
  public void         addMissingMaterialNodes();
  public WrlSelection getSelection();
  public boolean      hasSelection();
  public void         makeSelection();
}
```

The nodes of a scene graph forom a tree, which can be traversed using the `WrlSceneGraphTraversal` class

```
public class WrlSceneGraphTraversal
{
  public         WrlSceneGraphTraversal(WrlSceneGraph wrl)
  public void    restart();
  public WrlNode getCurrentNode();
  public int     getCurrentDepth();
  public void    advance();
}
```

The following code segment shows how to do so

```
WrlSceneGraph wrl=null;
// load wrl from file or create
WrlSceneGraphTraversal t = new WrlSceneGraphTraversal(wrl);
WrlNode child = null;
while((child=t.getCurrentNode())!=null) {
   // do something with the node here
   if(child instanceof WrlIndexedFaceSet) {
   WrlIndexedFaceSet ifs = (WrlIndexedFaceSet)child;
   WrlNode appearance = ifs.getAppearance();
   WrlNode geometry = ifs.getGeometry();
   // ...
   }
   t.advance();
}
```

The main class we want to operate on is WrlIndexedFaceSet

```
public class WrlIndexedFaceSet
{
  public WrlIndexedFaceSet();

  // constants used in various fields and to report property bindings
  public enum Field {
    CCW,
    COLOR,
    COLORINDEX,
    COLORPERVERTEX,
    CONVEX,
    COORD,
    COORDINDEX,
    CREASEANGLE,
    NORMAL,
    NORMALINDEX,
    NORMALPERVERTEX,
    SOLID,
    TEXCOORD,
    TEXCOORDINDEX
  };

  public void       setCcw(boolean value);
  public boolean    getCcw();
  public void       setConvex(boolean value);
  public boolean    getConvex();
  public void       setSolid(boolean value);
  public boolean    getSolid();
  public void       setCreaseAngle(float value);
  public float      getCreaseAngle();
  public void       setNormalPerVertex(boolean value);
  public boolean    getNormalPerVertex();
  public void       setColorPerVertex(boolean value);
  public boolean    getColorPerVertex();
  public void       setCcw(String s);
  public void       setColorPerVertex(String s);
  public void       setConvex(String s);
  public void       setCreaseAngle(String s);
  public void       setNormalPerVertex(String s);
  public void       setSolid(String s);
  public WrlNode    getCoord();
  public WrlNode    getColor();
  public WrlNode    getNormal();
  public WrlNode    getTexCoord();
  public VecFloat   getCoordValue();
  public VecFloat   getNormalValue();
  public VecFloat   getColorValue();
  public VecFloat   getTexCoordValue();
  public void       setCoord(WrlNode coord);
  public void       setColor(WrlNode color);
  public void       setNormal(WrlNode normal);
  public void       setTexCoord(WrlNode texCoord);
  public int        getNumberOfCoord();
  public int        getNumberOfColor();
  public int        getNumberOfNormal();
  public int        getNumberOfTexCoord();
  public VecInt     getCoordIndex();
  public VecInt     getColorIndex();
  public VecInt     getNormalIndex();
  public VecInt     getTexCoordIndex();
  public void       setCoordIndex(VecInt coordIndex);
  public void       setColorIndex(VecInt colorIndex);
  public void       setNormalIndex(VecInt normalIndex);
  public void       setTexCoordIndex(VecInt texCoordIndex);
  public boolean    hasFaces();
  public void       makeFaces();
  public int        getNumberOfFaces();
  public MeshFaces  getFaces();
  public boolean    hasEdges();
  public void       makeEdges();
  public int        getNumberOfEdges();
  public GraphFaces getEdges();
  public boolean    hasColorPerVertex();
  public boolean    hasColorPerFace();
  public boolean    hasColorPerCorner();
  public boolean    hasColors();
  public boolean    hasNormalPerVertex();
  public boolean    hasNormalPerFace();
  public boolean    hasNormalPerCorner();
  public boolean    hasNormals();
  public boolean    hasTexCoordPerVertex();
  public boolean    hasTexCoordPerCorner();
  public boolean    hasTexCoords();

  public final static int PB_NONE       = 0;
  public final static int PB_PER_VERTEX = 1;
  public final static int PB_PER_FACE   = 2;
  public final static int PB_PER_CORNER = 3;

  public int        getCoordBinding();
  public int        getNormalBinding();
  public int        getColorBinding();
  public int        getTexCoordBinding();
  public String     toString();
  public String     toString(boolean withDetails);
  public void       write(PrintWriter writer, String indent);
  public int        getNumberOfFields();
  public void       setField(int iField, String s);
  public boolean    fieldIsEditable(int iField);
  public String     getFieldName(int iField);
  public String     getFieldValue(int iField);
}
```

Since it is necessary for many algorithms to support variable length arrays for the connectivity (`coordIndex`), geometry (`coord`), and attached properties (`color`, `colorIndex`, `normal`, `normalIndex`, `texCoord`, `texcoordIndex`), the classes `VecInt` and `VecFloat` are provided.

```
public class VecInt
{
  public VecInt()
  public VecInt(int[] i)
  public VecInt(int nI_reserve)
  public VecInt(int nI_reserve, int value)
  public VecInt(VecInt src)
  public int    capacity()
  public void   copy(VecInt dst)
  public VecInt duplicate()
  public void   erase()
  public int    get(int j)
  public int    getBack()
  public int    getFront()
  public void   popBack()
  public void   popBack(int n)
  public void   pushBack(int[] v)
  public void   pushBack(int v)
  public void   pushBack(int v0, int v1, int v2)
  public void   pushBack(int v0, int v1, int v2, int v3)
  public void   pushBack(int v0, int v1, int v2, int v3, int v4)
  public void   pushBack(int n, int v)
  public void   pushBack(VecInt v)
  public void   reserve(int n)
  public void   set(int j, int vj)
  public int    size()
  public void   swap(VecInt otherVecInt)
}
```

```
public class VecFloat
{
  public VecFloat()
  public VecFloat(float[] f)
  public VecFloat(int nI_reserve)
  public VecFloat(int nI_reserve, float value)
  public VecFloat(VecFloat src)
  public int      capacity()
  public void     copy(VecFloat copyVecFloat)
  public VecFloat duplicate()
  public void     erase()
  public float    get(int j)
  public float[]  getArray()
  public float    getBack()
  public void     getBack(float[] v)
  public float    getFront()
  public void     popBack()
  public void     popBack(int n)
  public void     popBack(float[] v)
  public void     pushBack(float[] v)
  public void     pushBack(float v)
  public void     pushBack(float v0, float v1)
  public void     pushBack(float v0, float v1, float v2)
  public void     pushBack(int n, float v)
  public void     pushBack(VecFloat v)
  public void     reserve(int n)
  public void     set(int j, float vj)
  public void     set(int j, float vj0, float vj1)
  public void     set(int j, float vj0, float vj1, float vj2)
  public boolean  eq(int j, float vj0)
  public boolean  eq(int j, float vj0, float vj1)
  public boolean  eq(int j, float vj0, float vj1, float vj2)
  public void     add(int j, float vj)
  public void     mult(int j, float vj)
  public int      size()
  public void     swap(VecFloat otherVecFloat)
}
```

Other classes of interest are

```
public class MeshFaces
{
  public         MeshFaces(VecInt cIndex);
  public void    setCoordIndex(VecInt cI);
  public int     getNumberOfVertices();
  public VecInt  getCoordIndex();
  public int     getNumberOfFaces();
  public int     getNumberOfTriangles();
  public boolean isTri();
  public boolean isQuad();
  public int     getNumberOfCorners();
  public int     getNumberOfFaceIndices(int i);
  public int     getFaceCoordIndex(int iF, int iFC);
  public void    setFaceCoordIndex(int iF, int iFC, int iV);
  public int     getCorner(int iF, int iV);
  public int     getFaceCornerCurrPos(int fi, int vj);
  public int     getFaceCornerPrevPos(int fi, int vj);
  public int     getFaceCornerNextPos(int fi, int vj);
  public int     getFacePrevCoordIndex(int fi, int vj);
  public int     getFaceNextCoordIndex(int fi, int vj);
  public boolean isFaceEdge(int iF, int iV0, int iV1);
}

public class GraphEdge
{
  public GraphEdge(int v0, int v1, int indx, int next)
  public GraphEdge(int v0, int v1, int indx)
  public GraphEdge(int v0, int v1)
  public GraphEdge(VecInt vI, int j)
  public int     getVertex(int i)
  public int     getIndex()
  public void    set(VecInt vI, int j)
  public void    setIndex(int indx)
  public int     getNext()
  public void    setNext(int next)
  public int     getOtherVertex(int vi)
  public boolean isVertex(int vi)
}

public class Graph {
  public Graph()
  public Graph(Graph src)
  public Graph(boolean isOriented)
  public Graph(int nV)
  public Graph(int nV, boolean isOriented)
  public Graph(VecInt coordIndex, int nVsrc)
  public boolean   isConst()
  public boolean   isOriented()
  public int       getNumberOfVertices()
  public int       getNumberOfEdges()
  public GraphEdge getEdge(int i, int j, GraphEdge e)
  public GraphEdge getEdge(int i, int j)
  public GraphEdge getInverseEdge(GraphEdge e)
  public boolean   hasInverseEdge(GraphEdge e)
  public GraphEdge getFirstEdge(int i, GraphEdge e)
  public GraphEdge getFirstEdge(int i)
  public GraphEdge getNextEdge(GraphEdge e)
  public int       getIndexEdge(int i, int j)
  public int       getIndexEdge(GraphEdge e)
  public void      setIndexEdge(int i, int j, int indx)
  public void      setIndexEdge(GraphEdge e, int indx)
  public GraphEdge getOneEdge(int i, GraphEdge e)
  public GraphEdge getOneEdge(int i)
  public void      enumerateEdges()
  public void      insertEdge(int iV0, int iV1, int indx)
  public void      insertEdge(int i, int j)
  public void      insertEdge(int[] e)
}

public class GraphFaces extends Graph
{
  public GraphFaces(VecInt coordIndex, int nV)
  public int     getNumberOfFaces()
  public int     getNumberOfEdgeFaces(int eIndx)
  public int     getNumberOfEdgeFaces(GraphEdge e)
  public int     getEdgeFace(int eIndx, int i)
  public int     getEdgeFace(int eIndx)
  public int     getEdgeFace(GraphEdge e, int i)
  public int     getEdgeFace(GraphEdge e)
  public int     getOtherEdgeFace(int eIndx, int fi)
  public int     getOtherEdgeFace(GraphEdge e, int fi)
  public boolean isEdgeFace(int eIndx, int fi)
  public boolean isEdgeFace(GraphEdge e, int fi)
}

public class WrlSelection
{
  public                   WrlSelection(WrlSceneGraph wrl);
  public int               getNumberOfShapes();
  public int               getFirstVertex(int iShape);
  public int               getLastVertex(int iShape);
  public int               getNumberOfVertices(int iShape);
  public int               getNumberOfVertices();
  public int               getFirstEdge(int iShape);
  public int               getLastEdge(int iShape);
  public int               getNumberOfEdges(int iShape);
  public int               getNumberOfEdges();
  public int               getFirstPolyline(int iShape);
  public int               getLastPolyline(int iShape);
  public int               getNumberOfPolylines(int iShape);
  public int               getNumberOfPolylines();
  public int               getFirstFace(int iShape);
  public int               getLastFace(int iShape);
  public int               getNumberOfFaces(int iShape);
  public int               getNumberOfFaces();
  public WrlMatrix         getMatrix(int iShape);
  public WrlShape          getShape(int iShape);
  public int               getShapeIndex(WrlNode node);
  public void              setWrl(WrlSceneGraph wrl);
  public void              clear();
  public void              clear(int iShape);
  public void              clearVertices(int iV0, int iV1);
  public void              clearAllVertices();
  public void              clearAllVertices(int iShape);
  public void              clearEdges(int iE0, int iE1);
  public void              clearAllEdges();
  public void              clearAllEdges(int iShape);
  public void              clearFaces(int iF0, int iF1);
  public void              clearAllFaces();
  public void              clearAllFaces(int iShape);
  public void              clearPolylines(int iL0, int iL1);
  public void              clearAllPolylines();
  public void              clearAllPolylines(int iShape);
  public void              selectVertices(int iV0, int iV1);
  public void              selectAllVertices();
  public void              selectAllVertices(int iShape);
  public void              selectEdges(int iE0, int iE1);
  public void              selectAllEdges();
  public void              selectAllEdges(int iShape);
  public void              selectFaces(int iF0, int iF1);
  public void              selectAllFaces();
  public void              selectAllFaces(int iShape);
  public void              selectPolylines(int iL0, int iL1);
  public void              selectAllPolylines();
  public void              selectAllPolylines(int iShape);
  public void              invertVertices(int iV0, int iV1);
  public void              invertAllVertices();
  public void              invertAllVertices(int iShape);
  public void              invertEdges(int iV0, int iV1);
  public void              invertAllEdges();
  public void              invertAllEdges(int iShape);
  public void              invertFaces(int iF0, int iF1);
  public void              invertAllFaces();
  public void              invertAllFaces(int iShape);
  public void              invertPolylines(int iV0, int iV1);
  public void              invertAllPolylines();
  public void              invertAllPolylines(int iShape);
  public void              setNumberOfVertices(int nVertices);
  public boolean           isSelectedVertex(int iV);
  public int[]             getSelectedVertices();
  public int               getClearVertexIndex();
  public void              setClearVertexIndex(int value);
  public int               getDefaultVertexIndex();
  public void              setDefaultVertexIndex(int value);
  public void              newDefaultVertexIndex();
  public int               getVertexIndex(int i);
  public void              setVertexIndex(int i);
  public void              selectVertex(int i);
  public void              clearVertexIndex(int i);
  public void              invertVertexIndex(int i);
  public int               getVertexIndex(int iShape, int iV);
  public int               getShapeFromVertex(int i);
  public int               getShapeVertexFromVertex(int i);
  public void              setNumberOfEdges(int nEdges);
  public boolean           isSelectedEdge(int iE);
  public int[]             getSelectedEdges();
  public int               getClearEdgeIndex();
  public void              setClearEdgeIndex(int value);
  public int               getDefaultEdgeIndex();
  public void              setDefaultEdgeIndex(int value);
  public void              newDefaultEdgeIndex();
  public int               getEdgeIndex(int i);
  public void              setEdgeIndex(int i);
  public void              selectEdge(int i);
  public void              clearEdgeIndex(int i);
  public void              invertEdgeIndex(int i);
  public int               getEdgeIndex(int iShape, int iE);
  public int               getShapeFromEdge(int i);
  public int               getShapeEdgeFromEdge(int i);
  public void              setNumberOfFaces(int nFaces);
  public boolean           isSelectedFace(int iF);
  public int[]             getSelectedFaces();
  public int               getClearFaceIndex();
  public void              setClearFaceIndex(int value);
  public int               getDefaultFaceIndex();
  public void              setDefaultFaceIndex(int value);
  public void              newDefaultFaceIndex();
  public int               getFaceIndex(int i);
  public void              setFaceIndex(int iF);
  public void              selectFace(int iF);
  public void              clearFaceIndex(int i);
  public void              invertFaceIndex(int i);
  public int               getFaceIndex(int iShape, int iF);
  public int               getShapeFromFace(int i);
  public int               getShapeFaceFromFace(int i);
  public void              setNumberOfPolylines(int nPolylines);
  public boolean           isSelectedPolyline(int iL);
  public int[]             getSelectedPolylines();
  public int               getClearPolylineIndex();
  public void              setClearPolylineIndex(int value);
  public int               getDefaultPolylineIndex();
  public void              setDefaultPolylineIndex(int value);
  public void              newDefaultPolylineIndex();
  public int               getPolylineIndex(int i);
  public void              setPolylineIndex(int i);
  public void              selectPolyline(int i);
  public void              clearPolylineIndex(int i);
  public void              invertPolylineIndex(int i);
  public int               getPolylineIndex(int iShape, int iL);
  public int               getShapeFromPolyline(int i);
  public int               getShapePolylineFromPolyline(int i);
  public void              setShapeVertexIndex(int iV);
  public void              clearShapeVertexIndex(int iV);
  public void              setShapeFaceIndex(int iF);
  public void              clearShapeFaceIndex(int iF);
  public void              clearAllMatches();
  public WrlIndexedLineSet getMatchesIndexedLineSet();
  public void              clearPartialMatch(int iV);
  public void              clearPartialMatch(int iShape, int iSV);
  public void              setPartialMatch(int iShape, int iV);
  public void              setPartialMatch(int iV);
  public void              setMatch(int iShape0, int iV0, int iShape1, int iV1);
  public void              clearMatch(int iShape0, int iV0, int iShape1, int iV1);
}

public class Partition {
  public Partition(int n)
  public void  restart(int n)
  public int   find(int i)
  public int   join(int i, int j)
  public int   getN()
  public void  end()
  public void  makeParts()
  public int   getNumberOfParts()
  public int   getNumberOfElementsInPart(int i)
  public int[] getPart(int i)
}

public class StaticHalfEdges
{
  public StaticHalfEdges(int nV, VecInt  coordIndex)
  public int     getSrcVertex(int c)
  public int     getDstVertex(int c)
  public int     getNextCorner(int c)
  public int     getPrevCorner(int iC)
  public int     getTwinCorner(int c)
  public int     getNumberOfCorners(int iV, int jV)
  public int     getCorner(int iV, int jV)
  public int     getStarFirst(int iV)
  public int     isStarFirst(int iC)
  public int     getStarNext(int iC)
  public int     getTwinCorner(int iV, int jV)
  public int     getVertex(int c)
  public int     getOldVertex(int c)
  public int     getTwinVertex(int c)
  public int     getOneCorner(int iV)
  public int     getFace(int c)
  public int     getFaceCorner(int iF, int i)
  public int     getFaceVertex(int iF, int i)
  public int     getNumberOfVertices()
  public int     getNumberOfFaces()
  public int     getNumberOfCorners()
  public int     getNumberOfFaceCorners(int iF)
}
```