ShapeTools

ShapeTools is a Java library that simplifies the tasks of writing application programs that manipulate geometrical Shapes.

Using ShapeTools

It is difficult to describe how to use the ShapeTools library. This is because any use of the library is inherently specific to the problem being solved. It is therefore more productive to describe the types of problems for which the ShapeTools library is designed. With this in mind,

"The ShapeTools library supports development of portable computer programs that read, manipulate and write geometrical shapes".

And ....

Definition of a Shape

A Shape is a collection of sets of points, edges, faces , solids and data attributes.

A Point is a point in an N-dimensional space, generally represented by a real valued tuple. Each point in a Shape has a unique index number. Points are stored in a PointSet.

An Edge is a directed line segment starting and ending at a Point. The start and end points of an edge need not differ. An Edge may be considered to be an ordered pair of Point indices. Each Edge has a unique index number. Edges are stored in an EdgeSet.

A Face is an ordered collection of Edges. Each Edge has a unique index number. Faces are stored in a FaceSet.

A Data Attribute is a named value that is associated with a Shape, or a Shape Component (PointSet, EdgeSet, FaceSet, SolidSet).

Shapes may also contain other shapes, in a parent-child relationship.

Applicability of Shapes

The foregoing definition of a shape is surprisingly broad. It encompases surface meshes as well as sets of arbitrarily connected points. Shapes may have some,all, or none of the listed components. They may have only points (see example program ShapeReporter) or model complex surface meshes (see ShapeViewer, InterpolateContour). While a shape can obviously describe any vector geometry model, Shapes can also usefully model the sparsely filled data volumes used in implicit surface computations.

Compiling a ShapeTools program

To compile and run a shape tools java program you must add the ShapeTools library to the **classpath** seen by your java tools. This is typically accomplished by the following commands.
Please note: your computer may require some modification of the examples, depending on the specifics of how the Java tools were installed upon your computer.

Compile a Java file

javac -classpath .:ShapeTools.jar YourClassFile.java

run a Java class

java -classpath .:ShapeTools.jar YourClassFile

Please see any java programming text for more details.

ShapeTools? program examples

The ShapeTools library distribution includes many programming examples. Some of them are:

ShowVersion

An extremely simple example. It prints the version string of the ShapeTools library being used. Because new capabilities are often added to ShapeTools, it is useful for an application program to determine if the library being used is compatible with the needs and expectations of the application program. Failiure to detect incompatibilities will generally result in errors when an application is run. This program is reproduced in it's entirety here:

```
/*
 * COPYRIGHT NOTICE
 * Copyright (c) Craig Schwartz and Arthur W. Toga
 * See README.license for license notices.
*/
package examples;

import edu.ucla.loni.ccb.shape.About;

  /**
   * Prints the version of the !ShapeTools library to the standard output.
   *
   * @version 28 June 2005
   */
public class ShowVersion
  {
    /**
     * Example program showing how to use the
     * find the version number of the !ShapeTools library.
     *
     * @param args command line arguments (unused).
     */
    public static void main(String[] args)
      {
        System.out.println("Version : " + About.version());
        System.out.println("Major Version Number: " + About.getMajorVersionNumber());
        System.out.println("Minor Version Number: " + About.getMinorVersionNumber());
        System.out.println("Authors : " + About.authors());
        System.out.println("Copyright : " + About.copyrights());
      }
  }
```

ReadSupportedFileType

Illustrates how to read a file without specifying it's format. The methods show use the ShapeTools ShapeIO package to automatically determine how to read a file. The essential java code is:

```
String shapeFilename = args[0];


try {
  Shape shape  = reader.readShape(shapeFilename);
  Print.printShape(shape);
} catch (IOException e) {
  System.out.println("Unable to read shape file: " +
      shapeFilename + " Reason:" + e.getMessage());
}
```

Template
: A template to use when creating a new ShapeTools application. It shows how to test the version of the shape tools library, process command line arguments, and set up a java task that can be run as a separate java thread.

DrawShapeWithIterators
: A skeleton of a drawing program that shows how to get the points, edges and faces of an arbitrary shape, independent of the data file format from which it was read. This does not actually contain any GUI drawing code.

ReadWrite4DUcf
: This sample program show how to read a LONI "4D" Ucf file. The Ucf file is read, it's "4d" data attribute is obtained and modified,and the resulting changed Ucf shape is written to an output file.

InterpolateContour
: The first version of the InterpolateContour program, this advanced example shows how to read data Ucf files, perform spline interpolations on both vertex points and data attrbutes,and how to write the final results.

Revision: 08 May 2006