CompSurf: An Environment for Exploring Surface Reconstruction Methods on a Grid

June 10, 2003

Michael B. Gousie[1], Gregory Williams,
Trevor Agnitti, and Nicholas Doolittle
Department of Mathematics & Computer Science
Wheaton College
Norton, MA 02766
e-mail: mgousie@wheatoncollege.edu

[1]Corresponding author

Abstract

CompSurf is a novel visualization system that enables researchers in the area of Geographic Information Systems (GIS) to compare multiple surface reconstruction techniques simultaneously. The system displays two digital elevation models (DEM) and grid-based contours from which the surfaces are derived. The researcher can view the surfaces from any angle and examine statistics for each. The system's object oriented design allows the researcher to add new interpolation or approximation methods easily; a dynamically linked library obviates the need for system recompilation. A graphical user interface allows easy manipulation or computation of surfaces, and automatically incorporates new reconstruction methods added to the system.

Keywords: contours, interpolation, DEM, visualization, GUI

1 Introduction

The use of geographic information systems (GIS) is continually increasing, as more and more town planners, environmentalists, and traditional researchers use such systems in their work. These systems layer patterns or colors, which depict data such as soil type, roads, census information, and the like, over a two-dimensional map. One of the newer features of such systems is the ability to view such data in three dimensions, as can be done in ArcView and MapInfo. The user can then view the desired data in the context of the surrounding topology.

Digital Elevation Models (DEM) are often used to store three-dimensional elevation data via a regular grid. Because DEMs may not be readily available for a specific area due to costs, proprietary formats, and so forth, and/or because they are storage intensive, they are often interpolated or approximated from contour or other sparse data, freely available from the U.S. Geological Survey (USGS) or other sources. A GIS may incorporate several surface reconstruction techniques; the user may create a surface with one of the available methods, but has no way to determine if the resulting surface is "good." Such systems are not designed for the surface reconstruction researcher to evaluate the properties of a newly computed surface or to compare one surface to another easily.

CompSurf is a novel system that allows a surface reconstruction researcher to develop and test new reconstruction methods. The system allows two DEMs to be imported or computed, and then be displayed simultaneously, along with the underlying data. Although we assume the initial elevation data are contours digitized to a regular grid, the system can be extended easily to incorporate other kinds of data. The grid-based approach has been used successfully in many interpolation and approximation methods, such as TOPOGRID (Hutchinson, 1988), available in ArcInfo. When the data consists of contours, raster methods do not have the problem of flat triangles that may be produced by TIN methods (Jaakkola and Oksanen, 2000). The researcher can perform various transformations that are applied to all of the data on the screen simultaneously. This keeps all of the surfaces in the same orientation, allowing easy and intuitive qualitative evaluation of the displayed surfaces.

CompSurf's object-oriented design allows researchers to include easily new surface reconstruction methods in the system by incorporating objects in the hierarchy. System elements are written in an abstract manner such that they may be replaced, extended, or removed from the system without affecting the rest of the program. Elements of the system make use of dynamic linking to provide the ability to introduce new elements at run-time with no need for recompilation or static linking. These features provide the researcher with a flexible environment which may be extended to suit specific needs while at the same time reducing the effort necessary to implement such extensions. By allowing the researcher to focus on the development of new reconstruction methods and not the supporting application framework, time may be more efficiently spent on the research of new methods.

The unique interface of CompSurf gives the user the choice of loading existing contour and DEM files or selecting a reconstruction algorithm to compute a DEM from a contour file. Once DEMs have been loaded, they can be viewed and compared side by side. The viewing angle can be changed to a set of predefined positions, or rotated arbitrarily. The surfaces can be further manipulated through zooming, resolution, and projection options. When one surface view is changed in these ways, all other surface views are likewise changed, allowing the user to focus on the differences due solely to the reconstruction methods. Finally, the researcher can invoke statistics for each surface to allow for quantitative comparisons.

2 Previous Work

Visualization in the geosciences is complex, in part because of the vast amount of data that is available and the interaction among that data that researchers wish to explore. Of course, there are many full-fledged GIS as well as smaller systems available. All such systems will not be mentioned here, but rather, we give a sampling of those that allow direct comparisons of surfaces or have other related features.

The U.S. Army's Topographic Engineering Center has just completed an extensive survey of 550+ terrain visualization software products [10]. These range from the large systems such as ArcInfo, to small systems (see below), to software that does some mapping as an extra feature or for simple visualization purposes; Matlab is such an example. Some, such as 3D Studio MAX, are general modeling and animation packages, and do not have any specific terrain tools associated with them.

Large systems, such as ArcView and MapInfo, allow the user to view multiple surfaces created by different interpolation methods. For example, inverse distance weighting, thin plate interpolation, Triangular Irregular Network (TIN) generation, and kriging are all available in ArcView or ArcInfo. However, it is difficult for the surface reconstruction researcher to compare the various methods because each produces its own "view." In general, many GIS are difficult to use, requiring specialists to find the desired information and options to display the results in the desired way (Elvins and Jain, 1998). This means surface reconstruction researchers must learn much about a GIS even if they only need to understand a small subset of the entire system. Elvins and Jain suggest creating a GUI by following human-factors engineering practices.

Smaller systems generally do not have interpolating or other surface reconstruction features. One such system is Descartes, which allows researchers to perform visual data exploration on the Web (Andrienko and Andrienko, 1999). The system is easier for non-cartographers to use than traditional systems, but does not support 3D visualization. Comparisons, including some statistics, can be made between two DEMs through a dialog box in Autometric's SoftPlotter 2.0 [8]. However, the statistics generated are not meant for surface reconstruction researchers and are intended rather for simple DEM comparison purposes. CLRview (Hoinkes and Lange, 1995),[1] is a system for GIS visualization intended for Silicon Graphics IRIS workstations. Although some of the controls seem non-intuitive (i.e., many options are accomplished through the keypad or other keys), one nice feature allows the user to store an eye position so that a view can be duplicated later. The system also has the option of skipping polygons for faster rendering. MapRender3D [6] is a nice terrain viewer very good at importing/exporting various file formats. GeoTerrain [3], part of the larger GEOPAK package, is a vector-based system that allows one to compare two TINs or lattices. MapCalc [5] is a grid-based tool that allows statistical comparison between two maps. It is generally used with other visualization software, such as Surfer [9]. Delta3D [2] uses gridded elevation data and claims that such data can be visually and mathematically compared. LI Contour V+ [4] is a surface modeling and contouring system that can compare two surface models to calculate the volumetric difference. SIGNAL [7] allows users to statistically compare measurement data. The system allows multiple surfaces to be viewed and compared. This tool is geared toward wireless communications planning. Finally, (Huang et al., 2001) discuss an integrated GIS and virtual reality system for the Web.

3 CompSurf

CompSurf is a system that allows for surface reconstruction, the visualization of the resulting surfaces, and has features for comparing the quality of the output. In general, four specific problems manifest themselves in visualization tools (Gahegan, 1999): the speed of graphical rendering, the problems due to visual effects, the range of approaches and mappings, and the orientation of the user within the environment. CompSurf addresses some of these issues.

Because CompSurf is tailored to researchers in surface reconstruction, some of the complexities of general visualization tools are obviated. The researcher can focus on the problems of surface reconstruction and the comparisons between generated surfaces. The speed of rendering is addressed by allowing the user to change the terrain resolution. One of the main goals of the system is to ensure that the user does not "get lost" when viewing several surfaces at once; the orientation of all surfaces on the screen are always identical. Finally, visual effects are minimized by allowing the user to choose either perspective or parallel projections, depending on the desired outcome.

The system displays a window divided into four quadrants, similar to SIGNAL [7], the lower right of which contains the main graphical user interface (GUI). The GUI was written without the use of any templates or pre-made buttons so as to allow maximum freedom in design. Custom texture maps are used to create unique buttons and icons that need minimal explanation. The remaining three quadrants are used to display the initial data, in the form of contours, and the reconstructed surfaces. Fig. 1 shows the system displaying a contour file and two DEMs based on that data, along with the input/compute GUI.

The rendering and GUI portions of CompSurf are written in C++; OpenGL is the graphics API. All testing was done on a computer running Linux. Source code bundled in a tar file is available at http://cs.wheatoncollege.edu/mgousie/research.html.

3.1 Getting Data and Computing a DEM

The current system supports grid-based elevation data. Contour data files may be opened and the contour lines displayed. A DEM can be computed either from such contour data or imported as a complete file. The system currently reads data in ASCII Grid form, similar to the GRID format in

ArcView, but is easily extensible to handle other data file types (see Section 4).

The user typically begins by opening a contour file or other data source. This is accomplished by pressing the **Display/Compute** button, revealing a the GUI as shown in Fig. 1. In this case, an 800×800 raster contour file taken from a USGS map of Mt. Washington, NH (tuckermans.grid) and a corresponding DEM (tuckermans.maxc.grid) have been loaded previously. The GUI now shows that the user chose **Open DEM File**, and then selected tuckermans.beta.grid, a different DEM based on the same contour data, from the pulldown menu. The pulldown menu shows files stored in a system-created data directory; the user may type in a filename instead. By default, contours are displayed in the lower left quadrant. The user has the choice of where to display the resulting surface; in this case, the upper left quadrant was chosen, as indicated by the $\times$ in the corresponding quadrant of the "mini window" in the lower right of the GUI. Note the names of the files are shown in the lower left of each quadrant in which data is displayed.

Surfaces are rendered by triangulating the regular grid and displaying the triangles using OpenGL. Note that the image looks rather rough; by default, the resolution is coarse so as to give better performance when manipulating the images. This is achieved by skipping elevation points before triangulating, resulting in larger triangles and thus a much rougher appearance (see Section 3.2 for more details).

To compute a new DEM using one of the system's interpolation or approximation methods, the user starts by clicking the **Display/Compute** button, and then **Compute DEM**. The user can choose a method from the list, which currently includes inverse distance weighting, thin plate approximation, and some experimental algorithms. The user can then type in a file name to save the result, or let the system save it to a default name. The user may also choose the quadrant in which to display the result.

3.2 Manipulating and Assessing Surfaces

Once data has been loaded, the system looks as in Fig 2. The main GUI, the details of which are shown in Fig. 3, is used to manipulate the surface(s). To enhance performance, the surfaces are displayed in low resolution by default. This is done by skipping data points before triangulating. The resolution slider can be moved to the right to increase the resolution. The finest resolution is reached when all DEM data points are used. More refined methods for changing the resolution of terrain surfaces can be found in (Cignoni et al., 1997), (Puppo, 1996), and (Brown, 1996). For consistency, the same resolution is always applied to all of the displayed surfaces. When rotating the image(s), however, the resolution is decreased automatically to speed rendering. If the user wishes to compromise speed for accuracy, pressing the "lock" next to the resolution slider maintains the current resolution when rotating or scaling. However, the user still may move the slider, overriding the lock, if desired.

A typical problem faced by the surface reconstruction researcher is to compare surfaces computed from the same data by two different methods. While visually comparing surfaces, it is critical to keep the DEMs in the same orientation so that any perceived differences in the surfaces are not due to slight variations in the viewing angle. Fig. 2 shows what occurs when the user chooses to rotate the data: all of the surfaces and the contour data are rotated simultaneously to keep the viewer's orientation constant. Clearly, the right surface is smoother than the left in this view. The surfaces can also be rotated to common views by clicking on one of the "eyeballs" in the upper right of the GUI (Fig. 3). The "eyeballs" are positioned around a sample DEM, giving the user the option to view the scene from one of the four sides or from the top.

An important consideration for surface reconstruction researchers is to compare surfaces (DEMs) to one another and to the initial data, not only visually but quantitatively. We have included several statistics, including the well known root-mean-square error (RMSE) (Rinehart and Coleman, 1988) and total squared curvature. For a square grid of n^2 total points, the latter is found by comparing each computed elevation value to its four neighbors (Briggs, 1974), as shown in Eq. 1.

$$C_{sq} = \sum_{i=2}^{n-1} \sum_{j=2}^{n-1} (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j})^2 \tag{1}$$

where each u represents the elevation at the grid location indexed by i and j. A low total squared curvature indicates a smooth surface. Because small local imperfections may bias the total squared curvature, CompSurf computes an average absolute curvature as well (Eq. 2).

$$C_{ave} = \frac{1}{(n-2)^2} \sum_{i=2}^{n-1} \sum_{j=2}^{n-1} |(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j})| \tag{2}$$

Statistics for the surface are displayed by pressing the button for the indicated surface. The RMSE of a DEM in one quadrant relative to a DEM in another quadrant and/or relative to the original contours is displayed as appropriate. The statistics are visible in Fig. 2; in this case, they are representative of the surface in the left quadrant. The RMSE relative to the contours is computed by comparing the tuckermans.beta.grid DEM and the contours only at grid locations where there is a valid elevation in both files. Note that the RMSE is not zero, indicating that this surface is an approximation and not an interpolation. The RMSE compared to the surface refers to the error found when comparing the two surfaces, in this case tuckermans.beta.grid and tuckermans.maxc.grid. Other statistical measures can be added easily.

CompSurf also features zooming. Researchers often desire to view a small portion of a DEM. This can be done by pressing the zoom in button. Once again, the action is applied to both surfaces to keep the views consistent; see Fig. 4, which clearly shows that the peak area in the right surface is much smoother than the left. Finally, the surfaces can be viewed using a parallel projection instead of the default perspective projection. The nature of a perspective projection may distort a DEM enough to make it difficult to determine if the data or the projection is causing an unwanted artifact.

If the user is more text input oriented, CompSurf has analagous keyboard input for all the GUI options; for example, the arrow keys can be used for rotation. Finally, the system can be run using a full screen window or a smaller three-quarter screen window size.

4 Implementation Details

The system uses an object oriented design, programmed in C++. The class hierarchy is shown in Fig. 5. By using object orientation, the system allows modular extensions to be written and used without re-compiling or linking the system executable. In addition, by using class inheritance, new interpolating/approximating methods may be built upon existing methods, greatly simplifying and reducing the amount of code necessary to add a method to the system. This is similar to the general visualization tool ADVIZOR (Eick, 2000), in which extensibility was incorporated into the design making it easy to add new interactive operations.

The CompSurf design allows discrete units of code (classes) to modify the surface data in a pipeline. These classes each inherit from the Element class, implementing an interface which allows the querying of the surface data. Element classes are further divided between Filters and

Inputs; Filter elements each hold a pointer to the preceding element in the pipeline while Input elements are the source of data - the beginning of the pipeline. While only GridInput has been implemented in the current system, the design readily supports adding classes to handle input from other sources (e.g. Digital Line Graph files).

Filter classes represent two types of elements: interpolating methods and Sinks. Sinks are filter classes which are responsible for utilizing the data at a specific point in the pipeline. Sinks may be thought of as outputs for the pipeline. For example, the Surface sink uses OpenGL to render the surface to a display device, while the GridOutput sink writes a Grid file with the surface data.

The modular design of the system allows elements to be loaded at run time through an abstraction in the ElementFactory class. The system's ElementFactory class is responsible for instantiating elements. It makes use of the native operating system's dynamic loading feature allowing element classes to be dynamically linked into the running system. Since the ElementFactory class is the single point of contact with operating system dependent function calls, it may be replaced at compile time with a version suitable for the native operating system. The version implemented in the original system supports the dlopen API native to SunOS and Linux. Future versions of the ElementFactory might implement support for the Mac OS/NeXT dyld API or the BSD dld API. Even the ElementFactory class is written in a modular fashion, so that only the `load_so()` function need be replaced to support other dynamic loading APIs.

Adding an element class to the system is a straightforward process of implementing the Element interface. A simple example of the methods required to conform to the Element and Filter interfaces may be seen in the Normalize class. The constructor, `getRows()`, `getColumns()`, and `getHeight()` methods all have trivial one line implementations, while the lines of code in `preload()`, `refresh()`, and `message()` each number in the single-digits.

The most important part of implementing a new element is the `elementinfo()` function which provides metadata to ElementFactory's dynamic loader. Using the Normalize class as an example, the `elementinfo()` function is implemented as follows:

```
#ifdef SHAREDOBJ
ElementInfo& elementinfo ( void ) {
#else
ElementInfo& normalizeelementinfo ( void ) {
 5:  #endif
 6:        ElementInfo* info       = new ElementInfo;
 7:        strcpy( info->classname, "Normalize" );
 8:        strcpy( info->description, "Normalize Data" );
 9:        info->factory           = normalizefactory;
10:        strncpy( info->type, "UTIL", 4 );
11:        return *info;
12:  }
```

Lines 1 through 5 conditionally name the metadata function `elementinfo()` if the system is being compiled with support for dynamic loading, or `normalizeelementinfo()` if the system will be compiled statically into a single binary (If the system is compiled without dynamic loading enabled, the metadata function in each element needs a unique name since they will all share a single namespace). Allowing compiling without support for dynamic loading allows using the system with new methods even on systems for which a dynamic loader has not yet been written. Line 6 instantiates a new ElementInfo structure to hold the metadata. Lines 7 and 8 define identifying strings for the class. Line 9 assigns a function pointer (pointing to the `normalizefactory()` function) to the factory slot of the struct. The factory function is responsible for instantiating and returning a new object; The Normalize factory has a trivial implementation:

```
Element* normalizefactory ( void* p ) {
     return new Normalize ( (Element*) p );
}
```

The factory function takes an argument p (of type pointer to Element) since Normalize is a Filter class; p is a pointer to the rest of the pipeline. The possible types of element are interpolating method ("INTR"), source ("SRC"), sink ("SINK"), and a miscellaneous utility method ("UTIL"). Line 10 of the `elementinfo()` function defines Normalize as implementing a utility method. Finally, line 11 returns the metadata structure.

The makeelement script included with the system may be used to compile a new element as a dynamically loadable extension:

```
# ./makeelement -s Filter normalize.cpp
```

The makeelement script will handle the compilation of an element into a shared object while insuring dependencies on superclasses are met. The -s argument may be used to specify the element's superclass insuring that at run time the superclass is linked into the system before the subclass. After running the makeelement script, the element may be immediately used in the system. In the case of interpolating methods, the method name will appear in the GUI as an available method for interpolation.

The extensibility of the system was tested by adding five interpolation/approximation functions, including inverse distance weighting and several experimental algorithms, such as RomSpline. In each case, the new method included an `elementinfo()` function, a constructor, and functions to fill in the virtual functions of the base class. The makeelement script was then invoked to add the new algorithm to CompSurf. The new algorithms were added to the GUI automatically without recompilation of the entire system.

5 Conclusions and Future Work

We have implemented CompSurf, a novel visualization system for the surface reconstruction researcher. The system allows the user to view contour data and DEMs, as well as compute new DEMs from contours using the available interpolation/approximation methods. Among the features of the non-traditional GUI is rotation, in which all of the comparison surfaces remain in the same orientation. Faster rendering is achieved by adjusting the resolution of the surface(s). This feature and all others are applied to all of the displayed surfaces to keep the user's view consistent. Statistics, including curvature and RMSE, can be displayed for the quantitative assessment of a surface.

CompSurf was designed with extensibility in mind. To that end, we employed an object-oriented approach. New functionality, such as a new interpolation method, can be added easily without the need for recompilation. Several interpolation/approximation methods have been implemented to test the system's extensibility.

This is an experimental system, and as such, it has its faults. Its rendering speed could be improved, and some of the GUI elements may need more refinement; for example, while one may rotate the surfaces, there is no option for translation. Additional features, such as profile comparisons, coordinate displays, and other statistics may be added in the future.

Acknowledgments – This work was funded in part by Mars and Wheaton Fellowships.

References

Andrienko, G. L., Andrienko, N. V., 1999. Interactive maps for visual data exploration. International Journal of Geographical Information Science 13(4), 355–374.

Briggs, I., 1974. Machine contouring using minimum curvature. Geophysics 39(1), 39–48.

Brown, P. J. C., 1996. Selective mesh refinement for interactive terrain rendering. Technical report. Computer Laboratory, Cambridge UniversityPembroke St, Cambridge, CB3 QSG, UK, 16 pp.

Cignoni, P., Puppo, E., Scopigno, R., 1997. Representation and visualization of terrain surfaces at variable resolution. The Visual Computer 13(5), 199–217.

Eick, S. G., 2000. Visual discovery and analysis. IEEE Transactions on Visualization and Computer Graphics 6(1), 44–58.

Elvins, T. T., Jain, R., 1998. Engineering a human factor-based geographic user interface. IEEE Computer Graphics and Applications 18(3), 66–77.

Gahegan, M., 1999. Four barriers to the development of effective exploratory visualisation tools for the geosciences. International Journal of Geographical Information Science 13(4), 289–309.

Hoinkes, R., Lange, E., 1995. 3d for free – toolkit expands visual dimensions in GIS. GEO World 8(7), 54–57.

Huang, B., Jiang, B., Li, H., 2001. An integration of GIS, virtual reality and the internet for visualization, analysis and exploration of spatial data. International Journal of Geographical Information Science 15(5), 439–456.

Hutchinson, M. F., 1988. Calculation of hydrologically sound digital elevation models. In: Proceedings of the Third International Symposium on Spatial Data Handling. International Geographical Union, Columbus, Ohio, pp. 117–133.

Jaakkola, O., Oksanen, J., 2000. Creating DEMs from contour lines: Interpolation techniques which save terrain morphology. GIM International 14(9), 46–49.

Puppo, E., 1996. Variable resolution terrain surfaces. In: Proceedings of the 8th Canadian Conference on Computational Geometry, pp. 202–210.

Rinehart, R. E., Coleman, E. J., 1988. Digital elevation models produced from digital line graphs. In: Proceedings of the ACSM-ASPRS Annual Convention. Vol. 2. American Congress on Surveying and Mapping, American Society for Photogrammetry and Remote Sensing, pp. 291–299.

Internet References

[1] CLRview. http://www.clr.utoronto.ca/CLRVIEW/cvmain.html.

[2] Delta3D. http://www.tec.army.mil/TD/tvd/survey/Delta3D.html.

[3] GeoTerrain. 1998. http://selectservices.bentley.com/technotes/faqs/6163.htm.

[4] LIContourV+. http://www.tec.army.mil/TD/tvd/survey/LI_Contour_V.html.

[5] MapCalc. 2002. http://www.redhensystems.com/.

[6] MapRender3D. http://www.maprender3d.com/index.htm.

[7] SIGNAL. http://www.edx.com.

[8] Softplotter. http://techsupport.autometric.com/Products/softplotter.

[9] Surfer. http://www.goldensoftware.com/.

[10] Survey of Terrain Visualization Software. 2002. http://www.tec.army.mil/TD/tvd/survey/.

Figure Captions

Figure 1. Opening new DEM. Contours and DEM have been loaded previously.

Figure 2. Rotated, high resolution DEMs, associated contours, and statistics.

Figure 3. Main GUI.

Figure 4. Close-up view of peak area.

Figure 5. Class hierarchy; dashed lines indicate virtual classes.

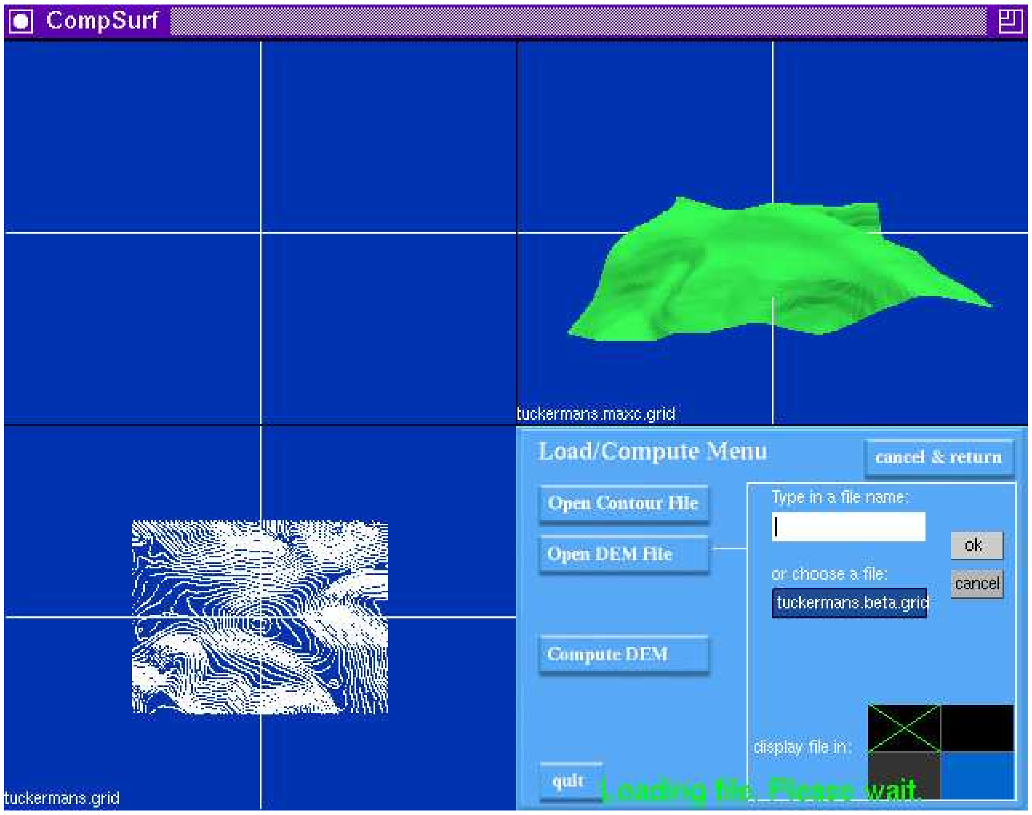
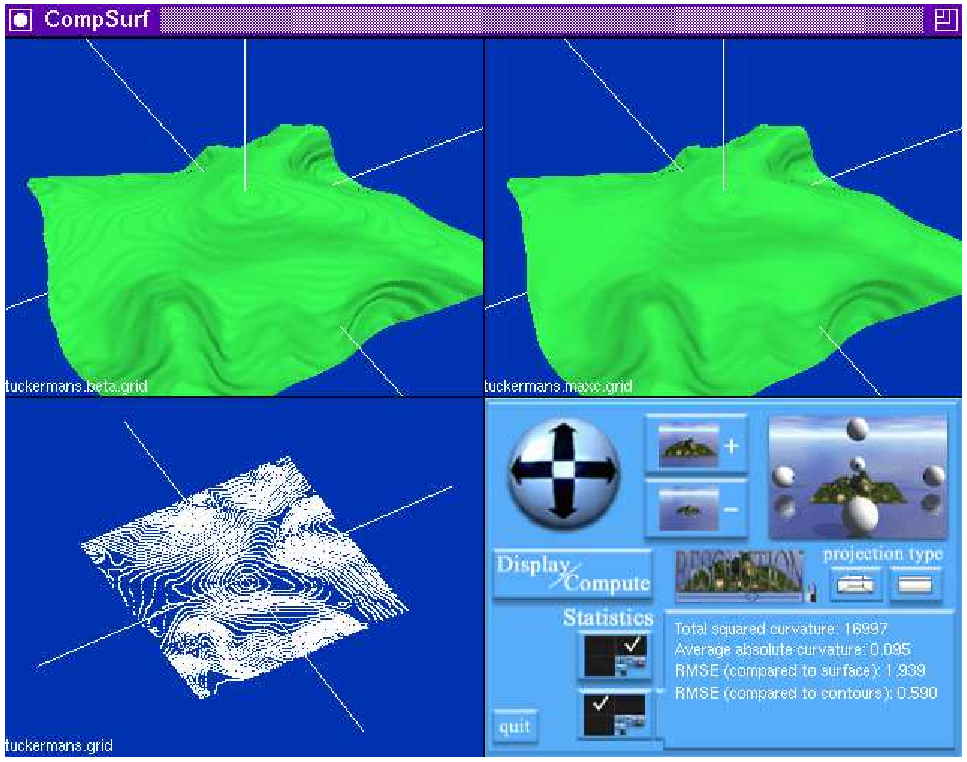


Figure 1: Opening new DEM. Contours and DEM have been loaded previously.



```
Total squared curvature: 16997
Average absolute curvature: 0.095
RMSE (compared to surface): 1.939
RMSE (compared to contours): 0.590
```

Figure 2: Rotated, high resolution DEMs, associated contours, and statistics.

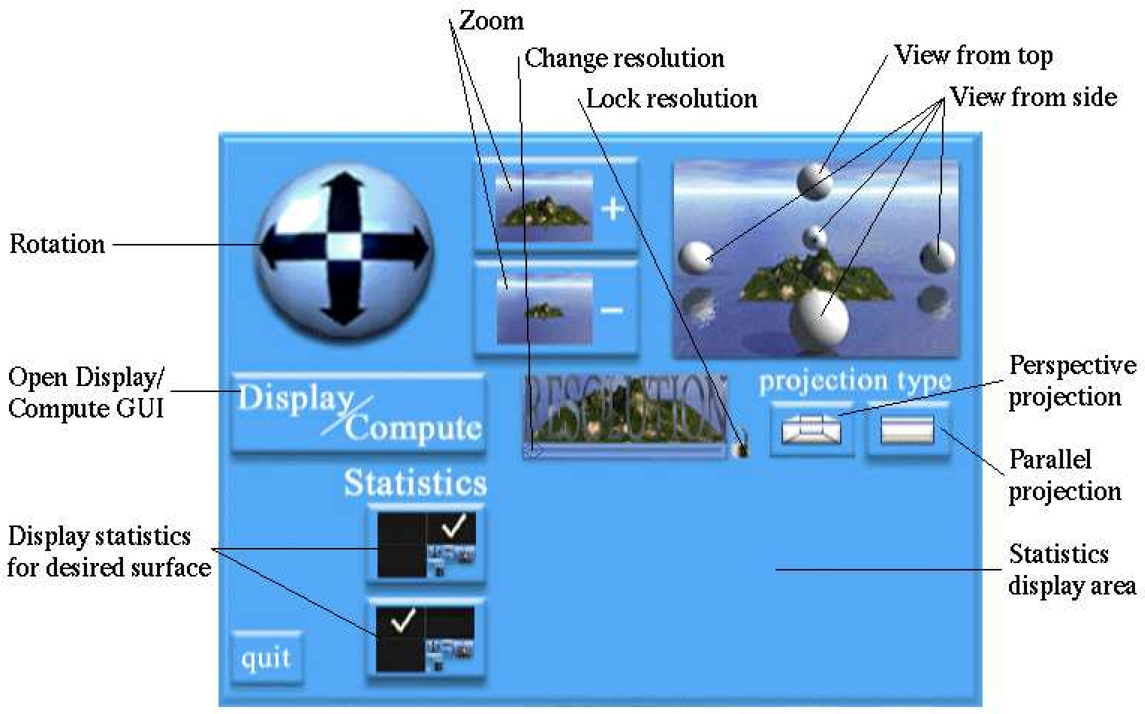
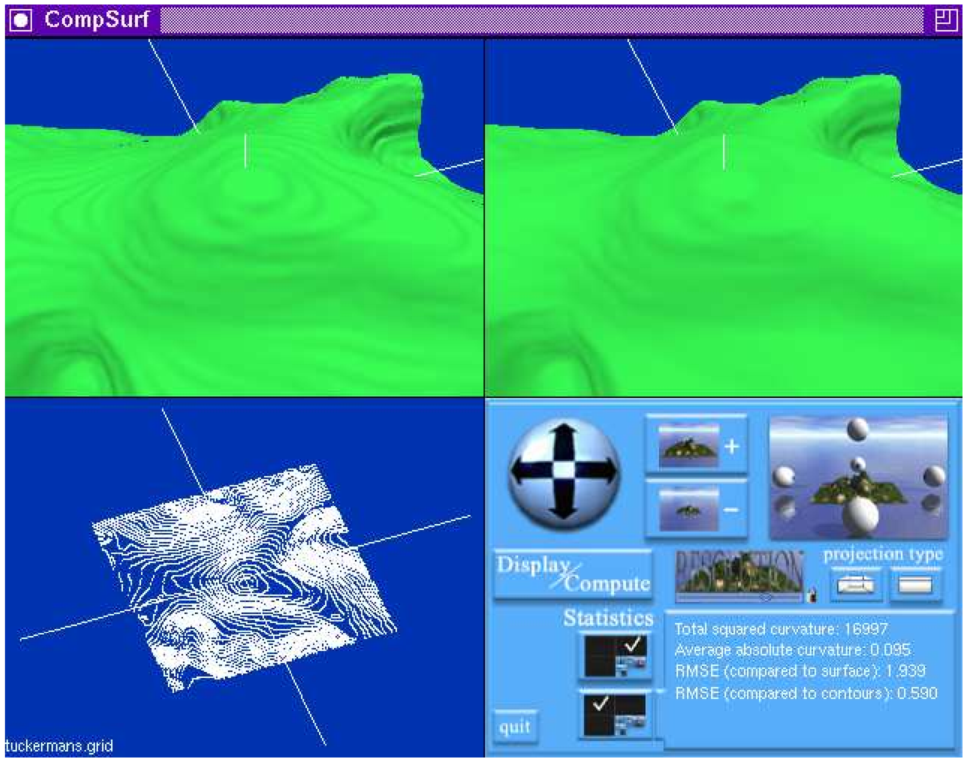


Figure 3: Main GUI.



Total squared curvature: 16997
Average absolute curvature: 0.095
RMSE (compared to surface): 1.939
RMSE (compared to contours): 0.590

Figure 4: Close-up view of peak area.

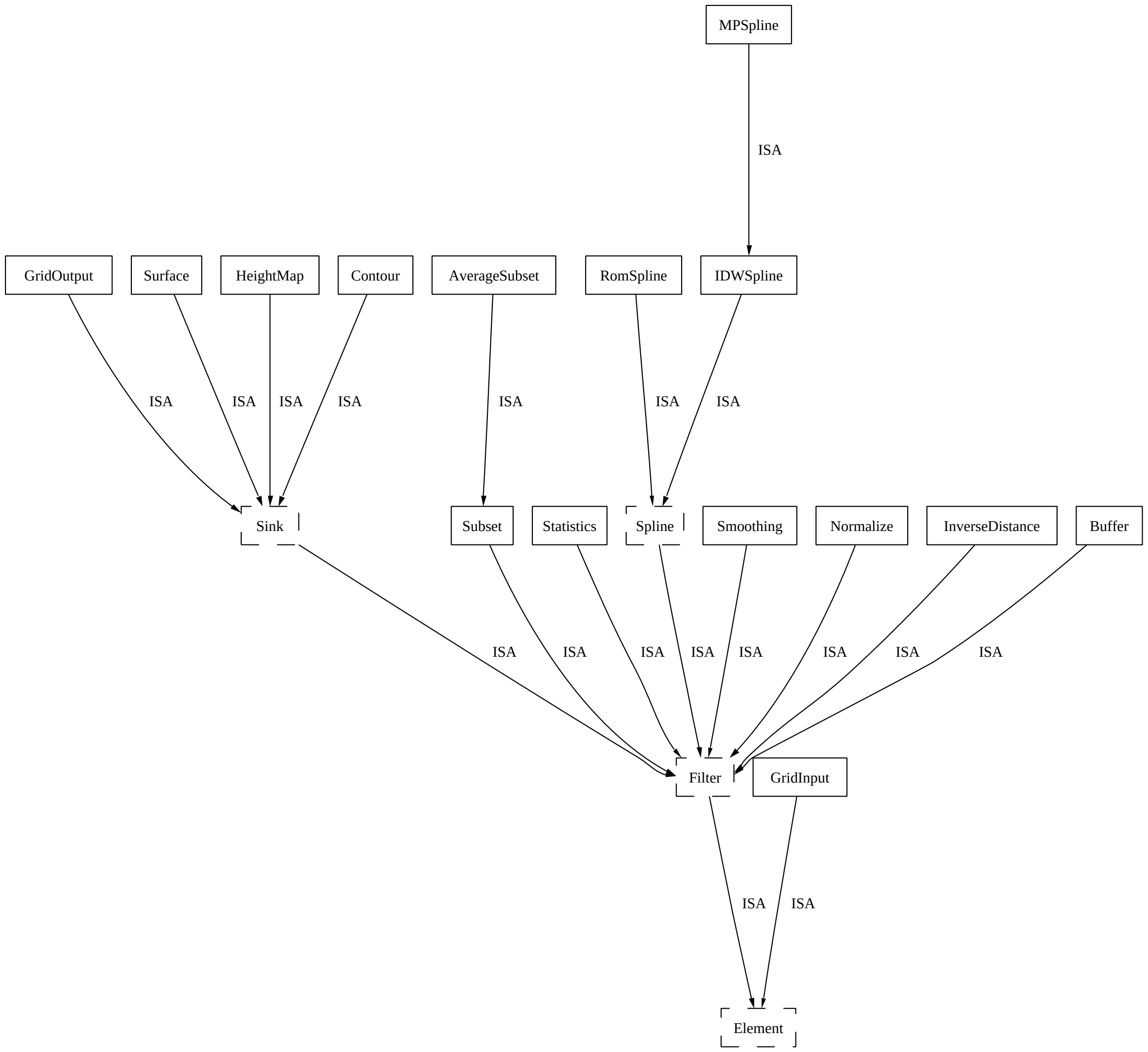


Figure 5: Class hierarchy; dashed lines indicate virtual classes.