“FROM DATABASES TO VISUALISATION”

Providing a User Friendly Visual Environment for Creating Geological Block Models

Tao Lin[1], Matthew O. Ward[2], Bill Power[3], and Danielle Landy[1]

Abstract

Geological structures are irregular and intricate, so creating a geological block model is a complicated process. A software system can only partially automate this process, since geological knowledge and experience are required. Therefore, the interaction of a geologist is critical for such a system. This paper discusses some useful advanced functions for enhancing the visualisation and manipulation of geological block models: flexible attribute display, multiple views, auxiliary display objects, and geologically sensitive operations. It also describes some experiments to explore the effectiveness of these functions and presents sample output. Most existing geological block modelling systems are implemented in an ad hoc way which greatly reduces their reusability and extensibility. These deficiencies may be improved by using the generic framework for geological block modelling systems, based on object-oriented technology, presented in this paper. A modelling process based on drill hole data is discussed as a case study.

KEYWORDS: Geological Block Model, Visualisation, Manipulation, GeoEditor, GeoModel, GeoSystem, 3-D Graphics, Entity, MetaObject, Adjustor, View

1 Introduction

A common task for geologists is to create, analyse, and modify geological block models derived by examining data from diverse sources; these data sources include magnetic data, gravity data, and drill hole data. The creation process of a geological block model is difficult to automate, as most geological structures are irregular, fractured or distorted regular solids [**?**]. Thus software systems for geological block modelling are required to allow a geologist to perform this task interactively. As many geological data are 3-D, a logical approach is to use 3-D graphics to allow the geologist to visualise and directly manipulate the models. [**?**].

The fundamental functions of a software system for geological block modelling can be divided into two categories: *visualisation*, through which a geologist can view the geological block models, and *manipulation*, through which the geologist can create and alter geological block models.

Here the subsystem of visualisation and manipulation, called the *GeoEditor*, is separated from the rest of the system, called the *GeoSystem*. A *GeoModel* is the data management component in a GeoSystem. The minimum functionality of a GeoSystem is a GeoModel, however, a GeoSystem can be a more complicated geological application. For example, a GeoSystem may provide the functionality to analyse or partially produce the geological block models. Figure 1 illustrates the conceptual relationships between a GeoEditor, a GeoSystem, and a GeoModel.

[1]CSIRO Division of Information Technology, Canberra Laboratory, GPO Box 664, Canberra, ACT 2601, Australia. E-mail: {taolin, dani} @cbr.dit.csiro.au.

[2]Worcester Polytechnic Institute, Department of Computer Science, 100 Institute Rd., Worcester, MA 01609 USA. E-mail: matt@cs.wpi.edu.

[3]CSIRO Division of Exploration & Mining, 39 Fairway, PO Box 437, Nedlands WA 6009, Australia. E-mail: w.power@dem.csiro.au.

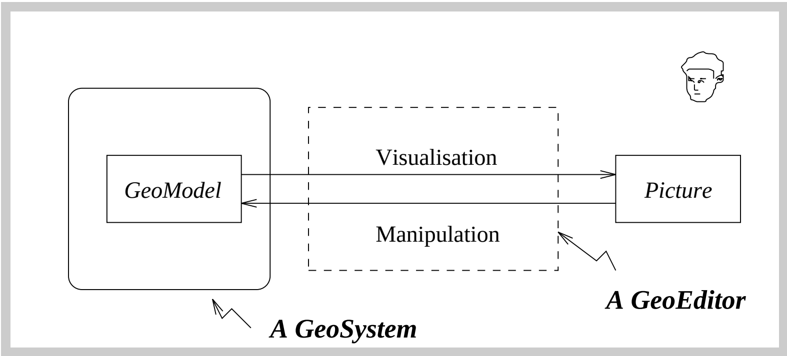


Figure 1: The conceptual relationship between a GeoEditor and a GeoModel.

The advantages in separating the GeoEditor from the GeoSystem are:

- To increase the reusability and extensibility. The implementation of a GeoEditor is expensive, since there are many technologies involved; such as user interfaces, 3-D graphics, geology and so on. Such a separation can reduce the dependency of these two parts [**?**]; that is the changes to a GeoEditor cannot affect the GeoSystem, and vice versa. Therefore, the extensibility of a GeoEditor can be increased. Furthermore, a GeoEditor can be used for different GeoSystems. Thus, reusability can be improved as well.
- To design and implement a GeoEditor in a systematic way. Most existing interactive visualisation applications were designed and implemented in an ad hoc way [**?**]. By separating them from the rest of the system, the user dependent components can be designed and implemented in a systematic way [**?**, **?**]. A toolkit for GeoEditors can be developed independently from the GeoSystems. Such a toolkit can be developed by specific domain experts; such as user interfaces or 3-D graphics experts, and reused and enhanced for specific applications.

A GeoEditor should be improved in the following aspects:

- Lack of geologically-sensitive functions. A GeoEditor is used for a geological application. Although geological structures are complicated, there are some particular topological or geological patterns found in specific kinds of geological units. To capture the geological block model effectively, a GeoEditor should provide some geologically sensitive functions (both visualisation and manipulation functions); for example, a semi-automatic surface generation function may use the geological context.
- Lack of advanced visualisation and manipulation functions. Although systems for 3-D geological block modelling processes have existed for many years, the visualisation and manipulation functions provided are still not effective enough, especially for manipulation. Furthermore, these systems mainly support geometry only. Besides geometric attributes, topological and geological attributes also need to be visualised and manipulated in a GeoEditor. To visualise and manipulate these attributes, other visualisation technology is required, especially for handling relational information.

This paper investigates these problems, analyses the requirements, and performs the experiments to explore some effective functions for enhancing the visualisation and manipulation functions.

The rest of this paper is organised as follows. Section 2 introduces a drill hole based geological block modelling procedure. Section 3 analyses the requirements for some advanced functions. Section 4 presents a generic framework for GeoEditors. Section 5 describes the implementation of these advanced functions. Section 6 investigates related work and Section 7 concludes the paper.

2 A Modelling Procedure

Geological block models are created from diverse data sources such as magnetic data, gravity data and drill hole data. This section will give a modelling procedure based on drill hole data.

Creating geological block models from drill hole data is one of the common approaches. Figure 2 illustrates a drill hole based modelling procedure[4].

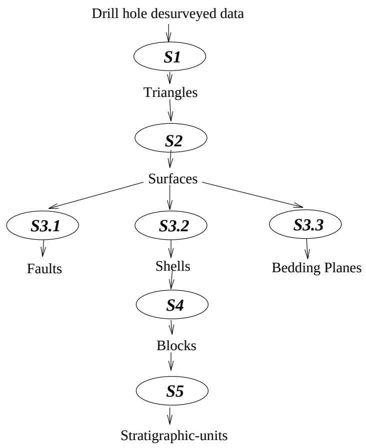


Figure 2: A drill hole based modelling procedure.

Given a set of "Drill Hole Desurveyed Data", the output of this procedure is to produce "Stratigraphic-units", "Faults" and "Bedding-planes" which are the leaf nodes in the modelling process tree shown in Figure 2. A *frame* is a set of geological objects which represent 3-D geological block models from a certain aspect; for example, all the triangles form a triangle frame. The frames are ordered from bottom to top in the tree structure shown in Figure 2. The operation of *promotion* converts the objects in a lower frame to the objects in a higher frame. The modelling procedure consists of a set of *modelling steps*. For example, the modelling step S1 generates the "Triangles" from "Drill Hole Desurveyed Data". One "Desurveyed Drill Hole Data" is composed of a set of drill hole segments which have several geological attributes besides geometry. Conceptually, given three drill holes, S1 creates a "Triangle" according to the geological attribute values of the associated drill hole segments. Figure 3 illustrates the modelling step, supposing that the values for all the "black" solid drill hole segments are the same. A set of "Surfaces" can be promoted to "Faults", "Shells", or "Bedding-planes", and the modelling steps are S3.1, S3.2, and S3.3 respectively. Other modelling steps follow similar principles.

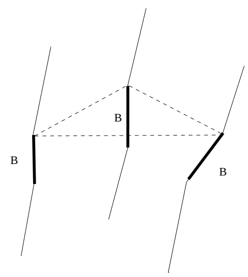


Figure 3: Creating a "Triangle" from three "Drill Hole Desurveyed Data".

[4] For more details of the geological objects used above, see [?].

3 Requirements

This section investigates the requirements for GeoEditors from two aspects: that of the systematic design of GeoEditors, and that of the functions which have not been well supported in existing systems. To solve the problems of the first aspect, Section 4 presents a generic framework for GeoEditors, and Section 5 shows the implementation of the functions addressed in the second aspect.

3.1 Functionality

The difficulty in interactively editing a 3-D data set arises from two main causes: the user can only see 2-D projections of the 3-D object and geological objects are irregular, complicated, and tend to obscure each other. The purposes of the first three functions discussed below are for providing better visualisation. The last one is aimed at improving the promotion operations.

Flexible attribute display

There could be a large number of geological attributes associated with a geological object. For example, a drill hole segment can have attributes such as rock type, gold grade, fracture density, and the certainty of the values. In contrast, the number of graphic attributes is limited; such as colour, transparency, texture, and drawStyle. To effectively use graphic attributes to represent geological attributes, a component is needed which should have the following functions:

- To dynamically change the current geological attributes to be visualised; for example, to change the "colour" for rock type to fracture density.
- To dynamically change the mapping rules. Taking "colour" as an example, according to the values of the gold grade of the geological objects, different "colour" values could be assigned for visualisation, and the transformation of geological attributes to visual attributes is defined by mapping rules.

Auxiliary display objects

Since geological units are irregular, complicated, and tend to obscure each other, it is difficult to manipulate a geological object correctly. To solve this problem, some auxiliary graphic objects would be helpful. This paper only focuses on one auxiliary display object: FrameGrid, which is used to provide the following functions:

- To provide referent graphic objects. It is difficult to find the right position in three dimensions. Some auxiliary display objects are required for reference.
- To provide boundary graphic objects. Normally, a GeoEditor only loads in a small portion of the geological block model from the GeoModel[5]. The integration of the currently modelled portion of the geological block models in a GeoEditor and the whole GeoModel is very critical. A boundary object is used to specify the boundary of the portion which is currently being modelled by a GeoEditor. Some special operations are required if manipulations go beyond the boundary. This boundary should also be visualised.
- To provide "feeling" graphic objects. Users may need the auxiliary graphic objects which can help them concentrate on the specific geological objects, and use them to touch the geological unit of interest, since users cannot be inside the 3-D geological block models.

Multiple views

A user may need multiple views to present the same geological data set:

[5] Figure 1 illustrates the relationships between a GeoEditor and a GeoModel.

- Users need to have separate views to present geological objects in different frames.
- Users need to view the geological objects in different categorisations; for example, several types of rocks may be clustered under a single view.
- Users need to view the geological objects according to different geological attributes, since users can view the geological attributes only through limited graphic attributes. To view several different geological attributes simultaneously, multiple views are required. Even to view the same geological attribute, users may need to view the geological data with different mapping rules simultaneously.
- Geological data are often irregular 3-D structures. To edit these efficiently, views with different viewing directions would be helpful [?]. Furthermore, users need to view the data in different spatial contexts; for example, when a user wants to create or modify a specific rock unit using a GeoEditor, they not only want to concentrate on the specific rock units, but also need to examine and perhaps alter the surrounding geological entities. Similarly, camera type and position, scale, light mode, transparency mode and shading type can also affect the user's perception as well. The views with different settings may help users to get more information from different aspects.
- Multiple views with different auxiliary display settings can also help users to get more information.

Geologically sensitive operations

Section 2 briefly presents a drill hole based modelling process. The modelling process is very expensive, since to manually do the promotion is quite time-consuming. To promote drill holes to triangles may take several weeks; even more, as several thousand drill holes is a typical case.

Since geological structures are irregular and intricate, the automation of the modelling process is very difficult. However, some particular topological or geological patterns can still be found in specific geological units. According to these patterns, some semi-automatic operations can be developed; that is the results of these semi-automatic operations may still need further modification or a user's interaction may be required for some ambiguous stages. Although such a semi-automatic operation cannot totally free users, the cost of the modelling would be greatly reduced.

The topological or geological patterns depend on specific geological interpretations of the geological units. Such geological interpretations vary from one geological area to another, even from one particular geologist to another. Therefore, very high flexibility is required for such a semi-automatic operation. Thus, one semi-automatic operation may cope with different geological interpretations and requests from different users.

To support semi-automatic operations, geological objects must be well structured: specific topological and geological constraints can be easily integrated with these geological objects, and various relationships between geological objects can be easily discovered (for example, it should be easy to find both the object being promoted and promoted from with a specific geological object). Thus, the specific restrictions on the geological units can be maintained by both semi-automatic operations and manipulations, and the objects in different frames can be treated consistently.

3.2 Systematic Design

From the discussion above, the characteristics of a GeoEditor can be summarised as:

- Specific geological context dependent. Firstly, a GeoEditor for drill hole based modelling is quite different from a GeoEditor for magnetic data based modelling. Even for the same drill hole based modelling procedure, one GeoEditor may differ from another if they are based on different geological interpretations. Secondly, the behaviour of a GeoEditor can vary from one user to another.
- Very expensive. There are many technologies required for the design and implementation of a GeoEditor; such as geology, 3-D graphics, human-computer interaction, graphic user interface,

computational geometry, and so forth. Furthermore, a GeoEditor is a very complicated system, and it needs many resources to develop and maintain.

Since a GeoEditor depends on specific geological interpretations, it is not too difficult to see that a single GeoEditor cannot satisfy all the requirements of different geological interpretations, and the modelling procedures from different data sources. To solve this problem, one possible solution is to extend the functionality of a single GeoEditor. This may cause the single GeoEditor to become unmanageable. Furthermore, it is difficult to solve the conflict between different geological interpretations. This is a common problem for many 3-D geological block modelling packages. The alternative solution is to have a specific GeoEditor for a particular geological block modelling procedure.

Since it is very expensive to design and implement a GeoEditor, ad hoc design and implementation is not acceptable. Therefore, a systematic design approach should be used for the design and implementation of GeoEditors. For this reason, a generic framework is addressed in the next section.

4 A Generic Framework

A single GeoEditor cannot satisfy the requirements of every specific geological application, so a specific GeoEditor is required for a particular modelling procedure. However, a GeoEditor is complicated and expensive to design and implement.

There are many data sources from which geological block models may be created, and many geological modelling procedures based on different geological interpretations. The GeoEditors for various applications may be needed. Therefore, the demand of GeoEditors is very large.

Although a GeoEditor may differ from another in various aspects, there are many common and similar functions. To repeat the implementation of many common and similar functions would make the ad hoc design and implementation very expensive.

To reduce the expense of the design and implementation of a GeoEditor, a systematic approach is needed. This approach should support the following characteristics:

- Reusability. It is desirable to reuse the components of existing GeoEditors.
- Extensibility. It is desirable to modify or extend the functions of the components of existing GeoEditors.

To support these characteristics, a generic framework is critical [?]. Such a generic framework provides a common structure and the interfaces between the components of GeoEditors. Thus, the development of different GeoEditors can follow the same design and implementation principles. Furthermore, this framework provides many components which are commonly used across different GeoEditors. Thus, the developer for a specific GeoEditor needs only to focus on some domain specific functions.

If the modification of component A may require the modification of another component B, A is called *code dependent* of B. A generic framework is aimed at minimising the code dependencies between the components [?]. Thus, the developer of GeoEditors can easily integrate, reuse and extend the existing components with minimum effort to change the associated components.

This generic framework partitions the components of the GeoEditors into the fixed functional fields. Thus, different GeoEditors will share a common structure, but have individual implementations for some components. By this way, the components with the same functional fields can be organised as a toolkit which consists of a set of implementations for the component of the same functional field. Thus, a GeoEditor developer may choose the best suitable implementation for a specific GeoEditor.

In the generic framework, there are four basic kinds of components:

- *Entities* are the objects which have been directly involved in the transformation of the information between the geological block models and the pictures. There are three entity class categories: logical entity, visual entity and graphical entity.
- *MetaObjects* are the objects which have been treated as the reference for the transformation between entities. The examples of metaObjects are the "colour mapping rules" presented in Section 5.1 and the "FrameGrid" presented in Section 5.2.
- An *adjustor* is used to update, organise, and modify metaObjects. The examples of adjustor are "FAD adjustor" described in Section 5.1 and "FGController" described in Section 5.2.
- A *view* encapsulates the other components for certain visualisation and manipulation purposes. There are four kinds of views: main view, logical view, visual view, and graphical view. Section 5.3 will further discuss these views.

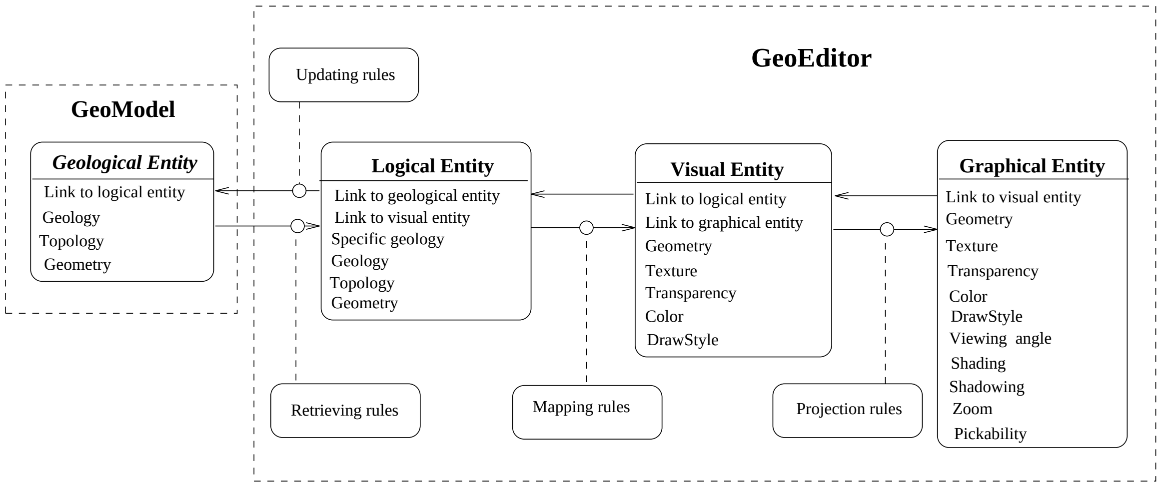


Figure 4: A visualisation and manipulation pipeline.

To minimise the code dependencies between components, this framework partitions the entities into three functional fields:

- *Logical entities* have the attributes derived from *geological entities* which are the objects for handling the data storing in a GeoModel. The attributes in the logical entities are composed from the associated logical entities.
- *Visual entities* are the intermediate objects which have the visual attributes converted from the attributes of logical entities. The attributes of visual entities are in an internal format which are different from the data format being used in the graphics facilities, such as Open Inventor.
- *Graphical entities* are the objects which can invoke the graphic functions of a specific graphics facility directly. Graphical entities provide the functions to display themselves and receive the user's interaction.

Figure 4 illustrates the entity pipeline to convert the data in the GeoModel (visualisation), and to transform the user's interaction on the picture (graphical entities) to the GeoModel (manipulation).

The GeoEditor is a user centred system with many parameters which depend on an individual's preferences. In most systems these parameters are hard coded which greatly reduces the flexibility [?]. A GeoEditor also depends on a specific application, some of the parameters may vary from one GeoEditor to another. To change some of the parameters, it is possible to use a component in another GeoEditor. It would be more advantageous to separate out those parameters. Here, the metaobject protocol technology [?] is adopted which is beyond a simple separation. A metaObject does not only provide the separated parameters, but the operations for handling these parameters as well.

To visualise the geological entities, logical entities are converted. In this process, a metaObject: "Retrieving rules"[6] is used as the reference for the conversion of the attributes of geological entities to the attributes of logical entities. Since different geological attributes may be used in different GeoEditors, the retrieving rules vary from one GeoEditor to another. However, the same conversion mechanism may be used. Figure 4 shows the other metaObjects: "Mapping rules", "Projection rules", and "Updating rules".

Only through graphical attributes displayed on the screen can a user get the perception of the geological attributes. There are many geological attributes used in various GeoEditors. In contrast, there are a few graphical attributes that can be used; such as colour, transparency, texture, and drawStyle. A metaObject: mapping rules, is used as the reference to convert the geological attributes to visual attributes. Section 5.1 further discusses the "colour" mapping rules.

There are geometry attributes in logical entity, visual entity and graphical entity respectively. In a comparison to the surface area of the modelling portion in a GeoEditor, the maximum depth of the drill holes is relatively smaller. To obtain a better perception, the length of the drill holes are required to be extended. A metaObject: projection rules are used to convert the geometry of visual entities to the geometry of graphical entities. In most cases, the geometry of visual entity and the geometry of its associated logical entity are the same. However, several logical entities might be converted into one visual entity for a higher abstraction. Thus, the geometry of logical entity and the geometry of its associated visual entity are not identical. While a modification is acceptable, the updating of the geological block models in GeoModel may be required. The metaObject "Updating rules" are used as the reference for this operation.

For the manipulation process, the modification of the geological attributes can be divided into two parts: geometry and the other attributes. To modify the geometry, a user interacts with the graphical entities directly, and the message of the modification is passed back to the logical entities through the links. The logical entities are based on a Data Model developed by the CSIRO Division of Information Technology, and Division of Exploration and Mining [?, ?]. This Data Model can provide different data consistency levels. After a user's modification, the consistency checking is conducted by logical entities. Once this action violates the data consistency, the associated logical entities inform the relevant visual and graphical entities to backup through the links. The modification of the other attributes are conducted in logical entities.

A user may need to dynamically modify the metaObjects. An adjustor is used for this purpose. An adjustor should provide the following basic functions:

- To allow the users to modify the parameters.
- To use other metaObjects which provide the same parameters, but support different operational behaviours.
- To save the current setting of the parameters and to reset the parameters to a setting saved previously.

The partition of entity functions is essential for the multiple views architecture of this generic framework. Section 5.3 discusses multiple views in more detail.

5 Implementation

This section presents the implementation of the advanced functions discussed in Section 3. Section 4 gave a generic framework for GeoEditors. The discussion of the implementation of these advanced functions is based on the generic framework.

[6]Figure 4 shows the relationship between geological entities, logical entities, and retrieving rules.

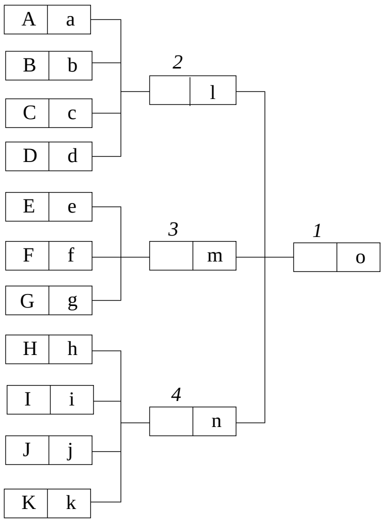
5.1 Flexible Attribute Display

Figure 4 shows a pipeline of visualisation and manipulation. To let the users understand the geological context, the geological attributes must be converted into graphical attributes which are displayed on the screen.

Logical attribute Visual attribute

LAttri.	Color
A	a
B	b
C	c
D	d
E	e
F	f
G	g
H	h
I	i
J	j
K	k

(a) A linear color mapping rules



(b) A hierarchical color mapping rules

Figure 5: Linear and hierarchical colour mapping rules.

To convert the geological attributes in logical entities to the visual attributes in visual entities, mapping rules are used. There are several visual attributes which can be used; such as colour, transparency, texture, and drawStyle. This section only discusses the "colour mapping rules" and the adjustor for handling the "colour mapping rules". The technology discussed here can be applied to handle the other attributes.

Figure 5.(a) shows linear colour mapping rules which are commonly used. Hierarchical colour mapping rules are illustrated in Figure 5.(b). Suppose that the geological units with attribute values as A, B, C, and D can be abstracted as a group labelled as 1. A user may need to view the geological units in both abstraction levels on different occasions.

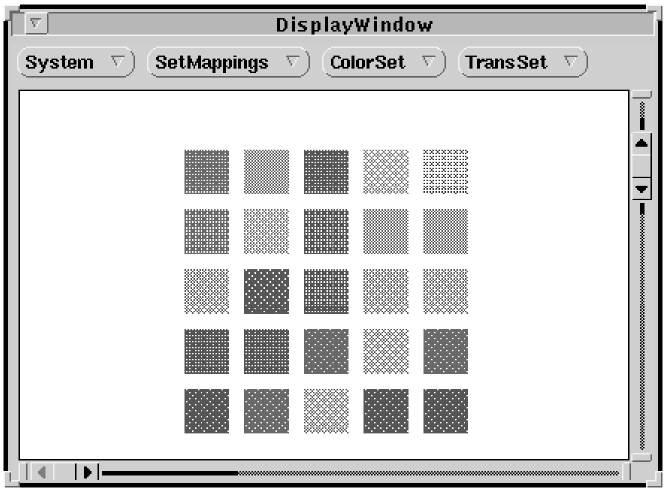


Figure 6: A picture of simulated geological units.

The adjustor for handling mapping rules is called a *Flexible Attribute Display* (FAD) Adjustor. Suppose

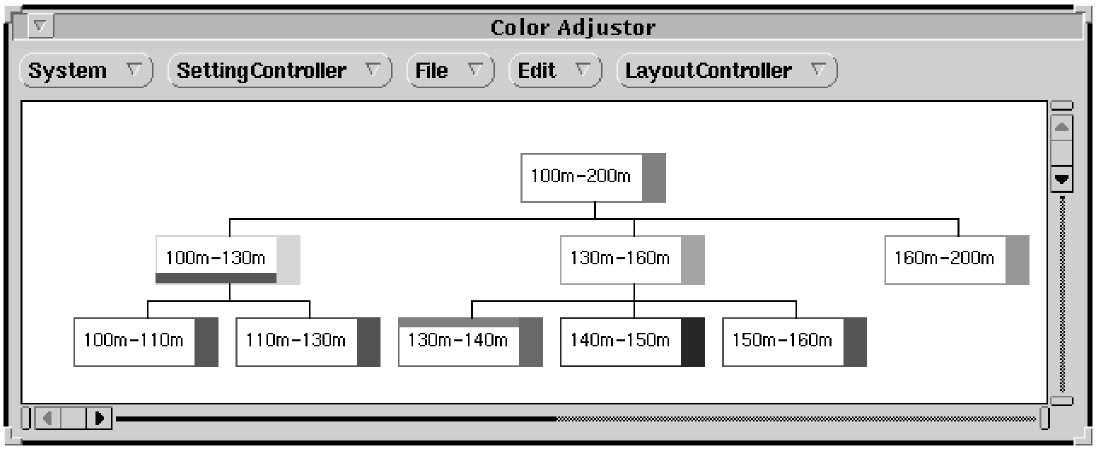


Figure 7: A colour FAD adjustor.

that a "filled-rectangle" is used to simulate a geological unit, Figure 6 gives a picture of a group geological units. Figure 7 illustrates the colour mapping rules being used.

This adjustor has the following functions:

- To use "colour" to visualise different geological attributes.
- To let the user change the colour mapping rules.
- To save the setting of colour mapping rules.
- To reset the colour mapping rules from a previously saved one.
- To dynamically control the mapping rules. There are two kinds operations which a user may operate on the nodes of the hierarchical colour mapping rules:
  - "Abstraction" forces all children to use the colour of the selected node.
  - "Inheritance" forces the selected node to use the colour of its parent.

  In Figure 7, an "Abstraction" has been operated on the node labelled 100m-130m, and an "Inheritance" has been operated on the node labelled 130m-140m. Figure 8 shows the modified result from Figure 6.

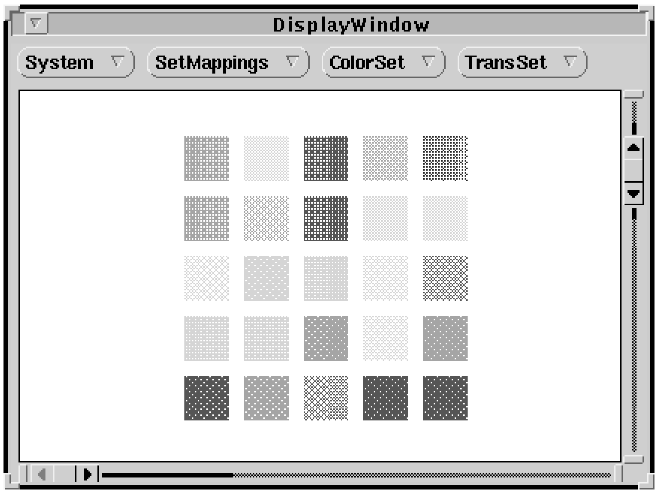


Figure 8: The simulated geological units modified by a colour FAD adjustor.

The results shown in this section are implemented using Parc Place Smalltalk VisualWorks on a Sun Workstation.

5.2 Auxiliary Display Objects

There could be many kinds of auxiliary display objects. This paper discusses only one of them: FrameGrid. Figure 9 illustrates the structure of a FrameGrid. A FrameGrid is a metaObject which consists of three parts:

- "Grid". "Grid" consists of a series of lines in the X, Y, and Z directions. Grids provide the referent positions which can be referred to, to get the relative spatial relationship between a set of geological objects, or only a geological unit.
- "Frame". A bounding box is used to represent the boundary of the the portion of geological block models which is currently being modelled by a GeoEditor. "Frame" is a rectangular box which is the visualised bounding box of the modelled position of the geological block models. Besides the frames of the box, "Frame“ also contains four corners which can be used to modify the size of bounding box.
- "Wall". "Wall" consists of three components: back, bottom, and side. "Wall" can be used as the "feeling" objects through which users can "touch" the graphic entities in the pictures. Three walls can be dragged along the 3 axises: X, Y, and Z individually.

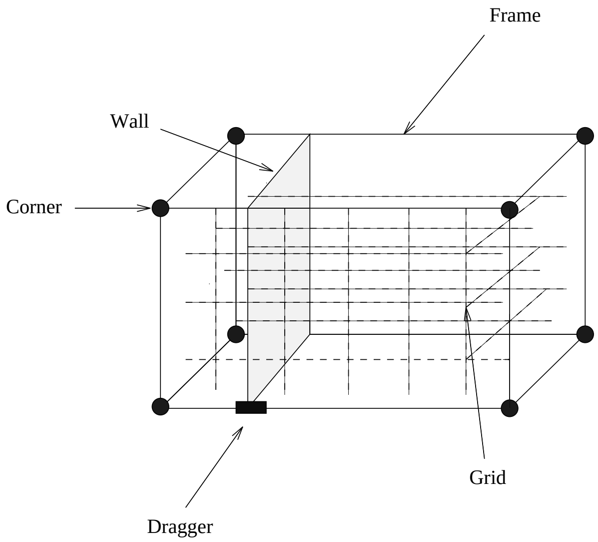


Figure 9: The structure of a FrameGrid.

A FrameGrid Controller (FGController) is an adjustor used to modify a FrameGrid.

Figure 11 shows a view with FrameGrid, and a FGController. With a FGController, a user can modify the following properties in a FrameGrid:

- To change the graphical attributes used for different components. For example, Figure 12 has changed the transparency of the walls and frames shown in Figure 11.
- To change the size of the graphical objects for the frames of the bounding box and the corners, and the gaps between the grid lines.
- To save the current FrameGrid and load a previously saved one.
- To turn on or off the display of various components of a FrameGrid. Figure 12 turns off the Grids shown in Figure 11.

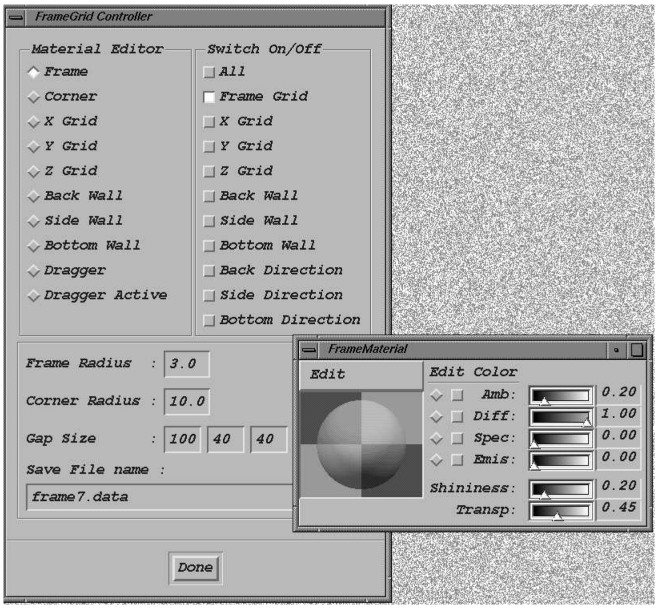


Figure 10: A FGController.

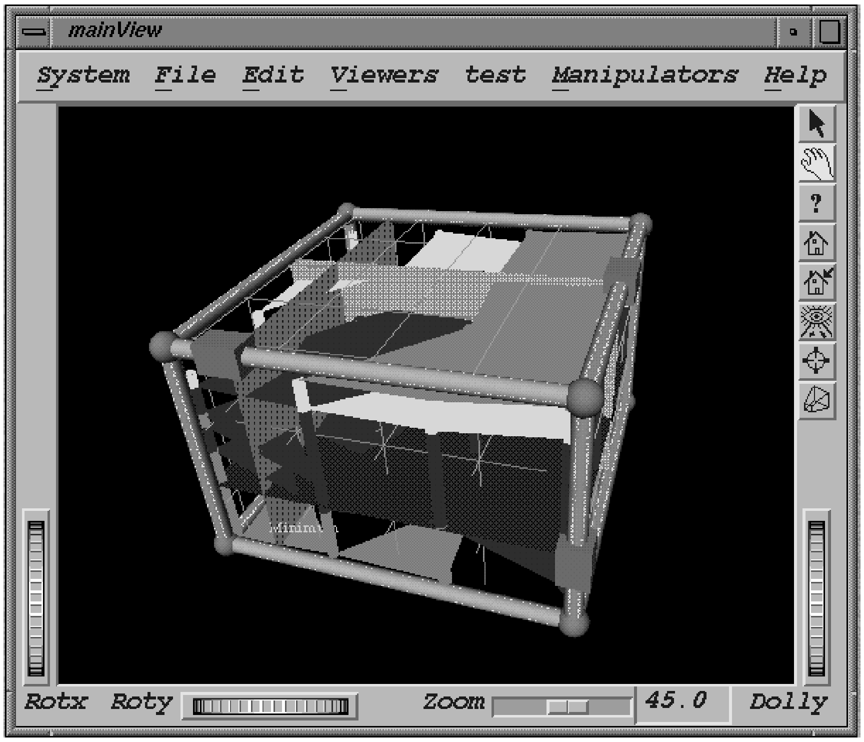


Figure 11: A FrameGrid with graphical entity.

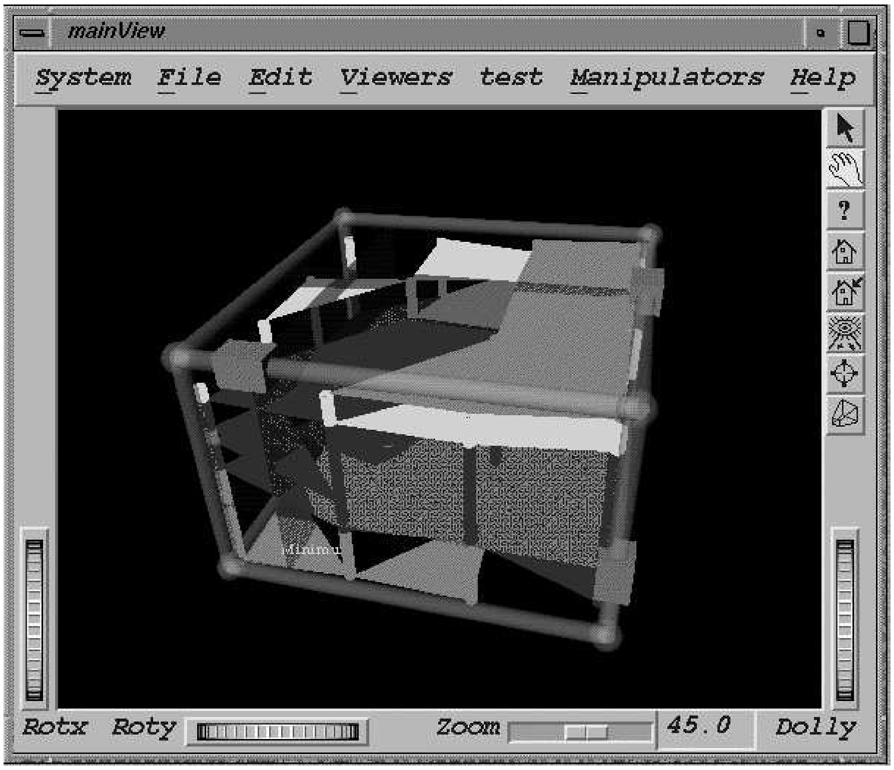


Figure 12: The FrameGrid with transparent walls and no grid lines.

5.3 Multiple Views

Since geological structures are very complicated it is difficult for users to obtain a complete mental image of them. A specific view can only show a specific aspect of the geological block models. In many cases, users may need to view several different aspects simultaneously, and multiple views are required.

Four basic kinds of views may be required in a GeoEditor:

- Main view. A main view supports all the functions for handling logical, visual, and graphical entities, which includes both visualisation and manipulation functions. The other views support just visualisation functions.
- Graphical view. A graphical view just supports the visualisation functions for handling graphical entities. Each graphical view has its own setting of camera type and position, scale, light mode, shading type, and the FrameGrid object. Figure 13 shows two views with different scales. It allows the user to manipulate data in a local area while maintaining a global perspective. Graphical views allow the user to view different graphical aspects such as viewing angles, scales, and degrees of "beauty" (for example, whether using curves instead of lines, and the density of the drawing).
- Visual view. A visual view provides the visualisation functions for handling visual and graphical entities. The visual view inherits from the graphical view and controls the mapping of geological to visual attributes. Visual views allow the user to view different visual aspects such as colour, transparency, texture and drawing style. Figure 5 and Figure 7 show two views with different colour mapping rules.
- Logical view. A logical view only supports the visualisation functions for handling logical, visual, and graphical entities. Logical views allow the user to view different logical aspects. The logical view is derived from the visual view, so it has all the functions provided by the visual view. For example, one logical view may show the drill holes, and the other may show both drill holes and surfaces. Figure 14 and Figure 15 illustrate the functions of these two logical views.

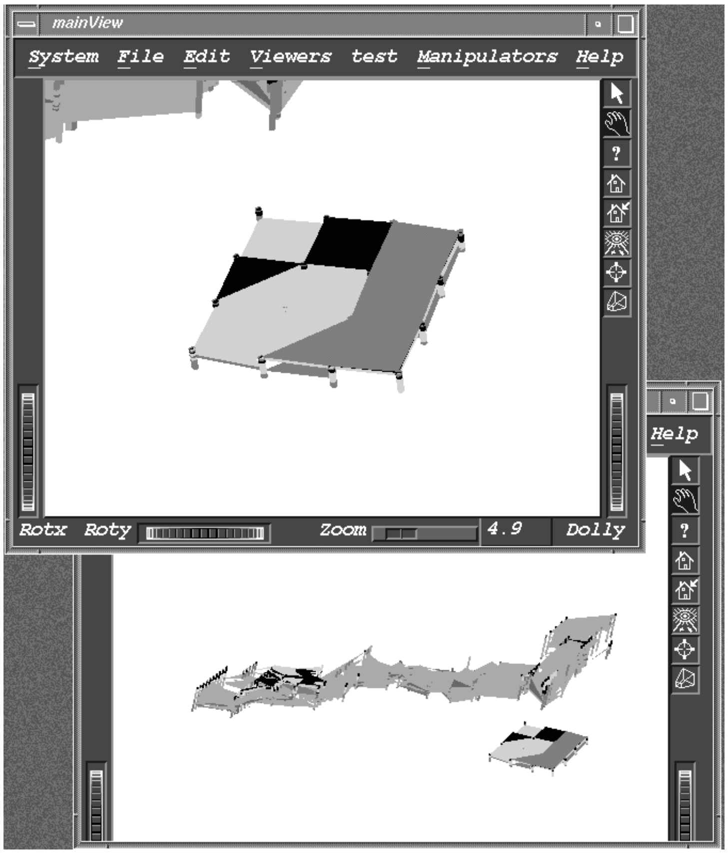


Figure 13: The graphical view displaying the model at two different scales.

This architecture reduces the additional duplication of the components and the entities. For example, if a user only needs to have views just with different scales, the duplications of the components and entities can be avoided.

5.4 Geologically Sensitive Operations

It is very time-consuming to do the promotion from drill desurveyed data to blocks manually. Since there can be particular topological or geological patterns in specific geological units, some semi-automatic operations can be developed. The expertise of geologists is still required for further refinement of the results produced by the semi-automatic operations.

The basic principles followed by a prototype [?] in generating semi-automatic Shells from drill hole data are :

- Extract the drill hole collar positions.
- Use Delaunay triangulation to obtain triplets of adjacent holes.
- Use dynamic programming to optimally align each triplet.
- For each 3-way match, find the extents of the geological values involved, and expand the surfaces.
- Link adjacent patches to form shells.

Figure 14 and 15 show the original drill hole data and the shells generated respectively.

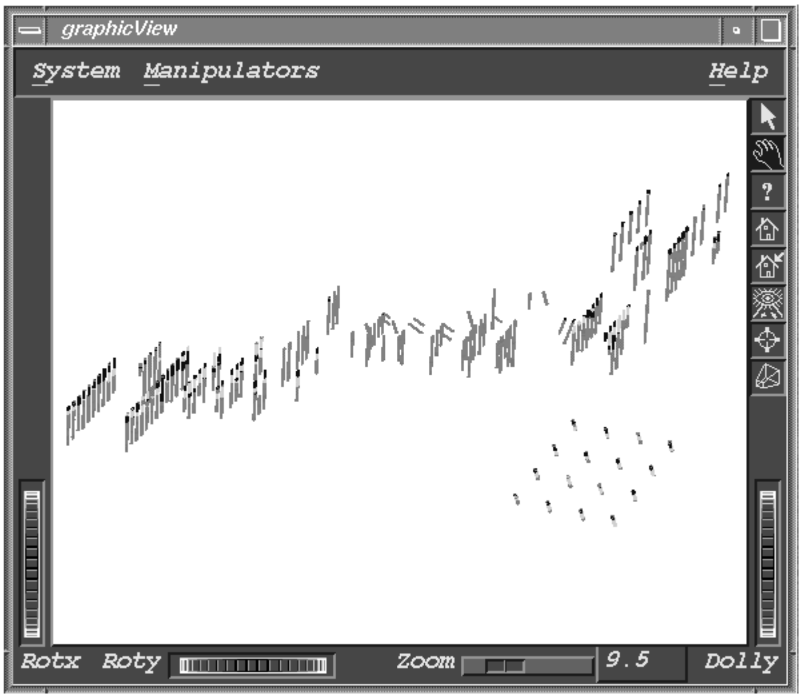


Figure 14: Drill hole data including a subset of data.

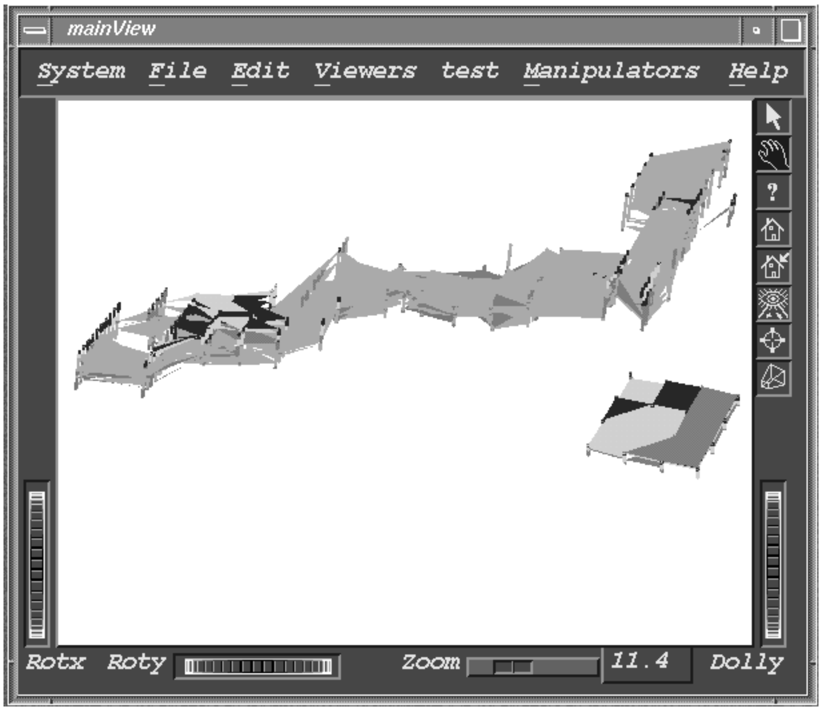


Figure 15: Drill holes and a semi-automatically generated surface.

6 Related Work

There are many GeoEditor-like systems used for 3-D geological block modelling [?, ?, ?]. Using these systems, users can visualise and manipulate geological block models. These systems have some common problems[7]:

- Lack of geologically sensitive functions.
- Lack of advanced visualisation and manipulation functions.

It is difficult for these systems to solve these problems, since most of them use only traditional CAD techniques [?]. Furthermore, many high-level CAD systems offer no, or almost no, possibilities for manipulation functions; instead they rely on command line input and menus [?].

To support geologically sensitive and advanced visualisation and manipulation functions, intensive development of 3-D graphics is required. Several complex visualisation packages, such as AVS [?] and Visualization Data Explorer [?], now exist on the market. Data flow diagrammatic presentations are used to compose modules and show the execution of a process [?]. These systems are still deficient in manipulation functions. Furthermore, it is very difficult to integrate specific geologically sensitive functions and support advanced visualisation and manipulation functions with these systems.

Only recently, 3-D graphics workstations became popular and affordable. The availability of some advanced 3-D graphics libraries, such as Open Inventor [?, ?] and UGA [?], can greatly reduce the expense of the development of 3-D graphics applications. However, many existing systems for geological block modelling were developed more than one decade ago, and it is very difficult to adapt new technologies to these systems.

A GeoEditor depends on the specific geological context. Therefore, a single GeoEditor cannot cope with the requirements of different geological block modelling procedures, and specific a GeoEditor for particular geological application is required. However, it is expensive to design and implement a GeoEditor. Thus, a systematic design and implementation approach is required which will allow maximum reusability, flexibility and extensibility. A generic framework is critical for such a systematic approach [?].

The representations of the data structures to be transformed between the applications and pictures were partitioned first by Kamada [?]. An object-oriented framework based on this partition was developed for Diagram User Interfaces [?].

Besides geometric attributes, the geological and topological relationships between geological units are also very important for users. This information can usually be described using relational information which is normally represented by diagram presentations [?, ?]. Using diagrammatic representations to show the hidden information for graphics editing was illustrated by [?]. Section 5.1 shows that diagrammatic representations can be applied to solve this problem further. Some 3-D technologies have been applied to visualise a large data set of relational information [?]. To analyse the relationship between a set of geological units, high-dimensional visualisation techniques [?] can provide assistance.

7 Conclusion

It is a logical approach to geological block modelling to let geologists visualise using 3-D graphics, and manipulate them directly.

This paper investigated the problems in existing geological block modelling systems: the lack of geologically sensitive and advanced visualisation and manipulation functions, and the ad hoc way for the design and implementation. These problems became the obstacles in using such systems.

[7]Section 1 gives more explanation.

Only recently, 3-D graphics workstations have become popular and affordable. Some advanced 3-D graphics libraries, such as Open Inventor [?], have greatly reduced the cost for the development of 3-D graphics applications. With the support of such a 3-D graphics library, it becomes possible for the further inventigation of advanced visualisation and manipulation functions. This paper analysed the requirements for advanced visualisation and manipulation functions, and geologically sensitive functions, and showed some results.

To investigate the common functions and reuse those functions, this paper presented a generic framework for GeoEditors. Hopefully this framework may lead to the development of GeoEditors in a systematic manner. This paper discussed the advanced functions based on this generic framework. This clean framework makes it easier to adapt these functions to existing software packages for geological block modelling.

Except for the FAD adjustor which is implemented using Smalltalk Visualworks, the advanced functions presented in this paper are implemented on a SGI Indy workstation using Open Inventor [?], IRIS Viewkit and C++.

Currently, the generic framework for GeoEditors is begin formalised, and the useful advanced visualisation and manipulation functions based on this generic framework are being investigated. For the second task, investigation into user modelling is required to provide more suitable functions for the geological block modelling process. Thus a toolkit for the development of GeoEditors may be developed.