MODEL MANAGEMENT IN VIRTUAL REALITY RESEARCH

Jungang Xu[1]*, Liang Zhou[2], Kun Zhang[2], and Wenyao Zhang[3]

*[*1]School of Information Science & Engineering, Graduate University of Chinese Academy of Sciences, 100080 Beijing, China.*
Email: xujg@gucas.ac.cn
[2]School of Information Engineering, Beijing University of Science & Technology, 100083 Beijing, China.
Email: {iamzhouliang, lilykun}@163.com
[3]School of Computer Science & Technology, Beijing Institute of Technology, 100081 Beijing, China.
Email: zhwenyao@gmail.com

ABSTRACT

In this paper, a Bi-angle Model Management method (BiMM) is proposed to manage models in virtual reality research. One angle is based on the model itself, which includes the model, model scheme, and texture; another angle is based on the model sort - each sort has its child sorts except leaf nodes. Based on this method, we have developed a model management application that has the following major functions: model management, model sort management, model query, model statistics, model registration into database as a whole, etc. With this method, researchers can manage model data conveniently and efficiently.

Keywords: Virtual Reality, Model, Model scheme, Texture, Model sort, XML

1 INTRODUCTION

Virtual reality technology was originally introduced in the late 1990s. It synthesized many technologies, such as digital image processing, computer graphics, multimedia, sensor technology, and so on, which facilitate the development of computer technology. Nowadays, virtual reality technology has been applied in many fields, such as computer games, industry, medicine, military, etc. Meanwhile, many universities and academies are also pursuing virtual reality technology research.

The virtual world created with virtual reality technology is artificial and imaginary, and there are many planar and three-dimensional models in this world. These models play different roles, and they can change their appearance according to changes in the environments around them. With the increase in the number of models, it is not enough to manage them manually. For example, to use a model during the course of building a virtual scene, a developer has to look for it in large numbers of models manually, which will negatively influence our work efficiency. Therefore, it is necessary to manage models with computer applications so that they can be found very easily and quickly.

One solution to this problem is proposed in this paper. In Section 1, the problem of how to manage a large number of models in VR research is put forward; in Section 2, a method of model management, called BiMM, is introduced; in Section 3, based on BiMM method, we develop an application using Visual Basic and Oracle 9i; and finally, we give the conclusions and directions for future work.

2 THE BiMM MODEL MANAGEMENT METHOD

In order to manage many models in virtual reality research, one method, called BiMM (Bi-angle Model Management), is proposed. As to Bi-angle, we consider model composition as one angle, and consider model sort as the other angle.

For the first angle, a model can be described by a 3-level architecture, which is 'model-model scheme-texture,' meaning that one model can have several model schemes, and each model scheme can have several textures. For example, 'Mary Zhang' is a virtual person in VR research, and she has different clothes for four seasons, including 'Spring clothes,' 'Summer clothes,' 'Autumn clothes,' and 'Winter clothes,' that is to say, model 'Mary Zhang' has 4 different model schemes. Each model scheme has its own textures; for example, Model scheme 'Winter clothes' has 4 textures, including 'Hat,' 'Jacket,' 'Trousers,' and 'Shoes'. This example is illustrated in Figure 1.

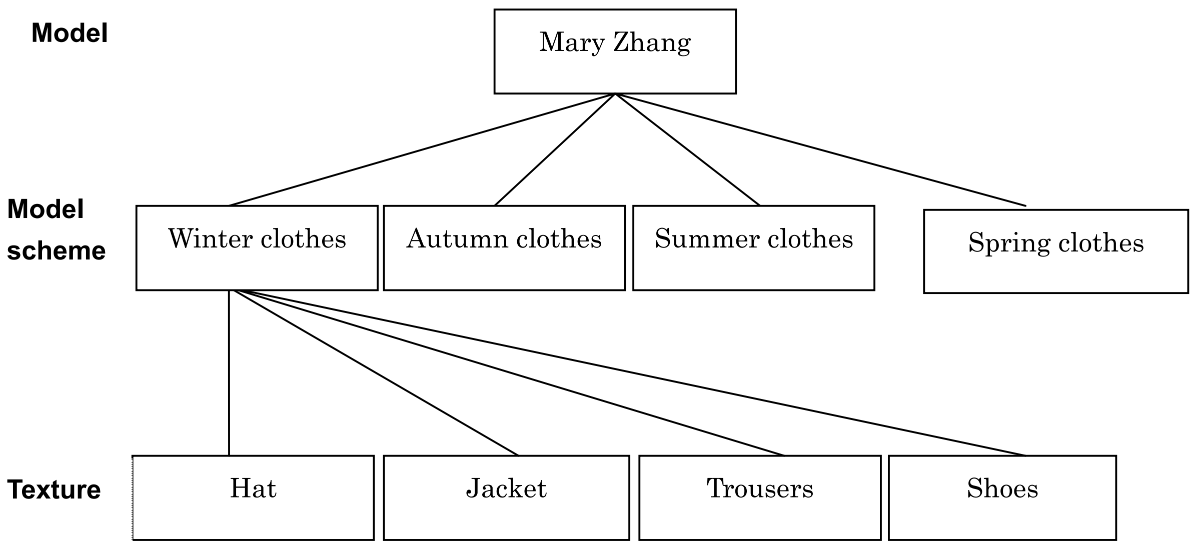


Figure 1. An example of model composition

As for the second angle, models can be classified into many model sorts, including one total sort (denoted as C_0), several 1-level sorts (denoted as C_1), several 2-level sorts (denoted as C_2), …, several N-level sorts (denoted as C_N). Therefore, model sorts can be denoted with a multidimensional array as follows: (C_0, C_1, C_2, …, C_N).

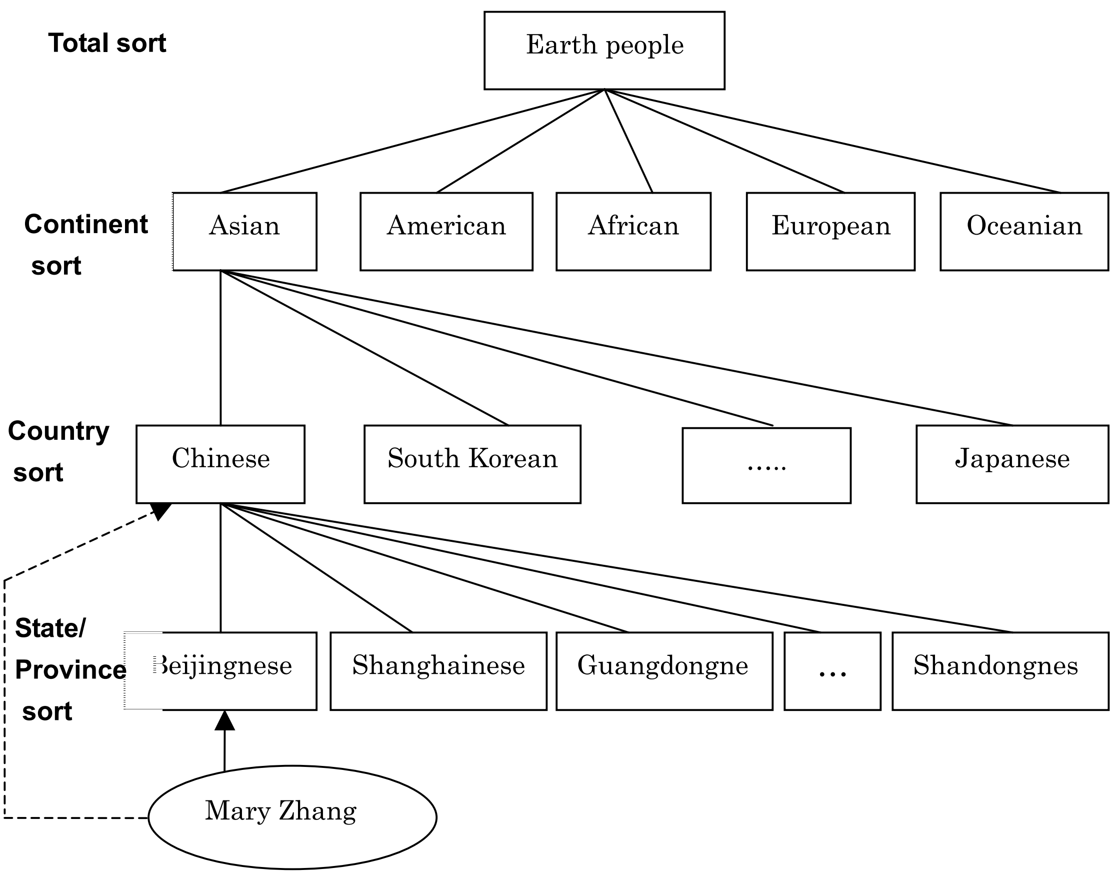


Figure 2. An example of model sorts

Note that a model sort has its child model sorts and its own models. For example, we can classify people into many sorts according to where they live, denoted as multidimensional array as follows: (Earth, Continent, Country, and State/Province). As for one person, such as 'Mary Zhang,' assume that she lives in Beijing, China, Asia, so we can place her in the 'Beijingnese' sort, or if her state/province attribute is not valuable, we can also place her in the 'Chinese' sort. This example is illustrated in Figure 2.

3 DATABASE DESIGN

3.1 The design of the E-R model and relation schema

Database design includes logic data model design and physical data model design. Based on the BiMM method proposed in section 2, we analyzed model information in virtual reality, and extracted four entities: Model, Model scheme, Texture, and Model sort. Their E-R model is displayed in Figure 3; the relationship between entity 'Model' and entity 'Model sort' is multiple-to-one (denoted as n:1); the relationship between entity 'Model' and entity 'Model scheme' is one-to-multiple (denoted as 1:n); and the relationship between entity 'Model scheme' and entity 'Texture' is multiple-to-multiple (denoted as m: n).

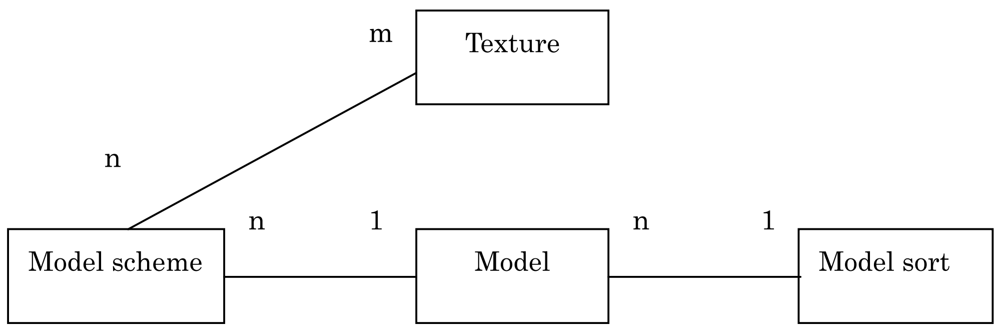


Figure 3. E-R model of model management in virtual reality

Based on the E-R model, we designed five relation schemas corresponding to the four entities described above. These five relation schemas are listed below; the underlined strings are keys or foreign keys of that relation schema.

Model(Model_ID,Model_Sort_ID,Model_Name,Model_Producer,Model_Texture_Number, Model_Register_Time,Model_Fize_Size,Model_Relative_Path);
Model_Scheme(Model_Scheme_ID,Model_ID,Model_Scheme_Name,Model_Sheme_Is_Defaut, Model_Scheme_Register_Time);
Texture(Texture_ID,Texture_Name,Texture_Sequence_ID,Texture_Producer,Texture_Register_Time, Texture_File_Size,Texture_Relative_Path);
Model_Scheme_Texture(Texture_ID,Model_Scheme_ID);
Model_Sort(Model_Sort_ID,Model_Sort_Name,Model_Sort_Up_ID,Model_Sort_Full_ID, Model_Sort_Depth,Model_Sort_Creat_Time).

3.2 The algorithm for storing and querying model sort data

A model sort has its child sorts, and these child sorts also have their own child sorts. In the relational schema Model_Sort, we used four data fields to describe the father sort and child sort of the current model sort, which are Model_Sort_ID, Model_Sort_Up_ID, Model_Sort_Full_ID and Model_Sort_Depth.

Model_Sort_ID is the self-increasing key of table Model_Sort, which has an integer data type; Model_Sort_Up_ID stores the value of Model_Sort_ID of the father model sort of the current mode sort, also an integer data type; Model_Sort_Full_ID stores the whole path from the top model sort node to the current model sort node, with a varchar data type and separates different level nodes with the comma character (,). Model_Sort_Depth stands for the depth of the current model sort, that is, the depth from the top model sort node to the current model sort node.

Figure 4 illustrates an example of model sorts. In this figure, the digit in each circle represents the value of Model_Sort_ID of that model sort. So, in the database, the values of Model_Sort_ID, Model_Sort_Up_ID, Model_Sort_Full_ID, and Model_Sort_Depth are listed in Table 1.

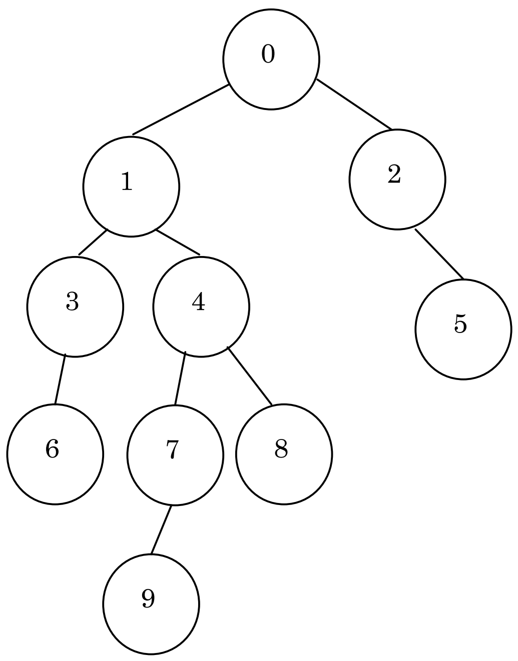


Figure 4. An example of multiple-level model sorts

Table 1. The values of Model_Sort_ID, Model_Sort_Up_ID, Model_Sort_Full_ID, Model_Sort_Depth

Model_Sort_ID （Integer）	Model_Sort_Up_ID （Integer）	Model_Sort_Full_ID （Varchar）	Model_Sort_Depth （Integer）
0	Null	0	0
1	0	0,1	1
2	0	0,2	1
3	1	0,1,3	2
4	1	0,1,4	2
5	2	0,2,5	2
6	3	0,1,3,6	3
7	4	0,1,4,7	3
8	4	0,1,4,8	3
9	7	0,1,4,7,9	4

Based on this algorithm, several examples of model or model sort queries are listed in Table 2; these queries are described with the SQL clause.

Table 2. Several examples of model or model sort queries

Query all child sorts of Model_Sort_ID=1sort	Select * from Model_Sort where Model_Sort_Up_ID=1
Query all child sorts of Model_Sort_ID=4 sort, and list them in sequence	Select * from Model_Sort where ‘,’+ Model_Sort_Full_ID +‘,’like ‘%,’+ (select Model_Sort_Full_ID from Model_Sort where Model_Sort_ID=4)+‘,%’ order by Model_Sort_Full_ID DESC
Query all models included by Model_Sort_ID=4 sort and its all child sorts	Select * from Model where Model_Sort_ID in (select Model_Sort_ID from Model_Sort where ‘,’+ Model_Sort_Full_ID +’,’ like ‘%,’+ (select Model_Sort_Full_ID from Model_Sort where Model_Sort_ID=4))

4 APPLICATION DESIGN AND IMPLEMENTATION

We used Microsoft Visual Basic as our programming language, used Oracle 9i as our database management system (DBMS), and implemented the model management module, model sort management module, model queries module and model statistics module. Meanwhile, we implemented model registration into database as a whole with XML technology. These modules will be discussed in section 4.1 and 4.2.

4.1 Application module design

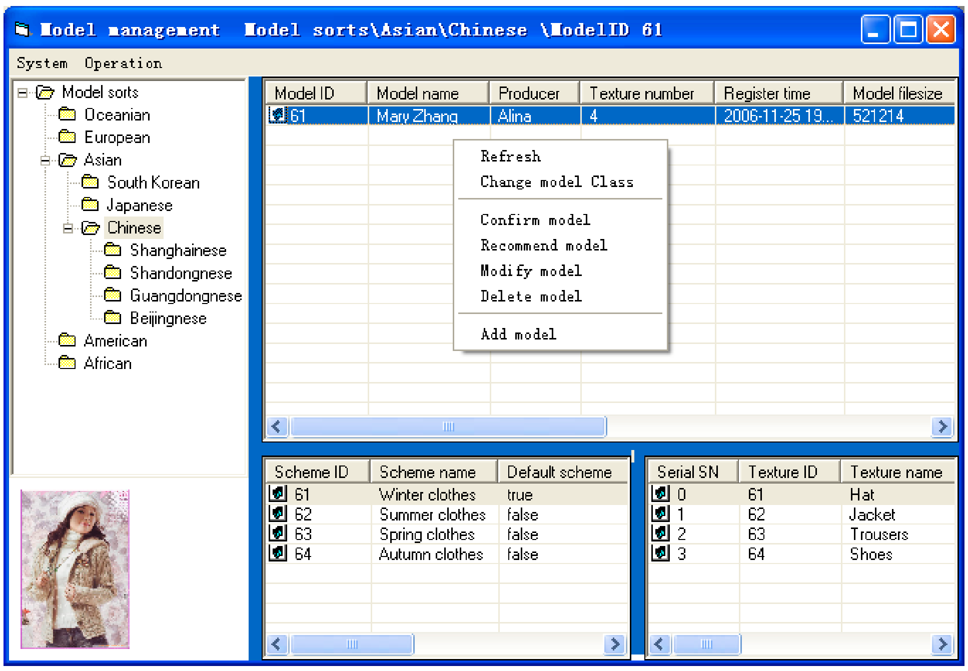
The main modules of this application are described as follows.

4.1.1 File system configuration

This module is used to configure the file folder path which is used to store model files and texture files. Three parameters need to be configured in this module: the virtual root directory (specifying the root directory of all related files), model file folder (specifying the model file path relative to virtual root directory), and texture file folder (specifying the texture file path relative to virtual root directory).

4.1.2 Model management

In this module, we use three windows to realize model management: the model window, model scheme window, and texture window. These three windows are not isolated; they are correlated, model scheme window displays all model schemes of the current model selected in model window, and texture window displays all textures of the current model scheme selected in model scheme window. Figure 5 displays these three windows, and also displays a model named 'Mary Zhang' in model window and its model schemes ('Spring clothes', 'Summer clothes', 'Autumn clothes' and 'Winter clothes') in model scheme window, textures ('Hat', 'Jacket', 'Trousers' and 'Shoes') of 'Winter clothes' are displayed in texture window.



- Model sorts
  - Oceanian
  - European
  - Asian
    - South Korean
    - Japanese
    - Chinese
      - Shanghainese
      - Shandongnese
      - Guangdongnese
      - Beijingnese
  - American
  - African

Model ID	Model name	Producer	Texture number	Register time	Model filesize
61	Mary Zhang	Alina	4	2006-11-25 19...	521214

Scheme ID	Scheme name	Default scheme
61	Winter clothes	true
62	Summer clothes	false
63	Spring clothes	false
64	Autumn clothes	false

Serial SN	Texture ID	Texture name
0	61	Hat
1	62	Jacket
2	63	Trousers
3	64	Shoes

Figure 5. Model management window

Adding a new model is easily done through a model window. The float menu will appear (illustrated in Figure 5), and by selecting the menu item 'add model,' a model adding window will appear. When adding a new model, at least one model scheme of this model (considered as default scheme) must be added, along with one texture of each mode scheme. Without this, the model data cannot be registered into the database.

4.1.3 Model sort management

Model sort management can be realized in the left window of the application. To add a new child sort under a model sort, this model sort is selected first; one float menu will appear (illustrated in Figure 6). By selecting the menu item 'add subclass,' a sort adding window will appear. Meanwhile, model sorts can be deleted but only when there are no models or child sorts under this model sort.

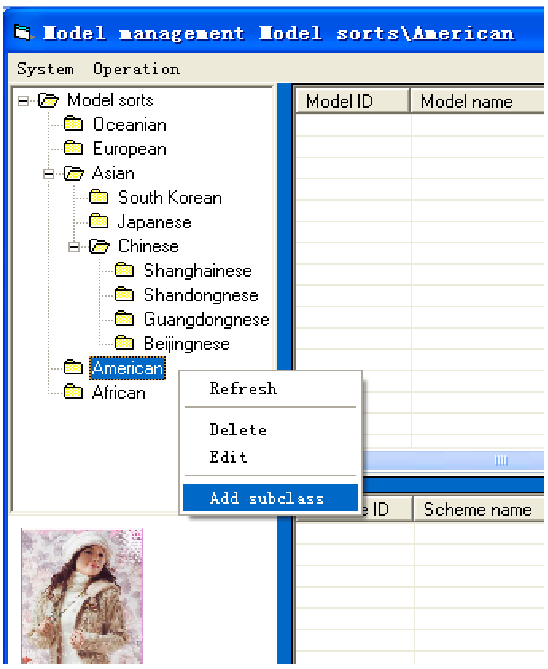


Figure 6. Model sort management window

4.1.4 Model queries

The model data stored in the database can be retrieved according to multiple query conditions, such as model sort, numbers of mode schemes, texture name, model registering time, and so on.

4.1.5 Model statistics

Model statistics can be realized according to various statistical requirements, such as model accounts registered during a particular time interval, model file size in a particular interval, all models under particular model sorts, numbers of textures included in one model, and so on.

4.2 Registering bulk model data into the database

During the course of registering model data into the database, if one model is registered at a time, the registering efficiency will be very low. However, when there are many similar models, it is possible to register them as a whole. Therefore, a module that can register many models into database at once has been designed and developed. It can greatly improve registration speed. In this module, XML technology is used to define the model set; to register this model set into the database, it must first be extracted from the XML file. The model set definition in the XML file is listed below.

```
<?xml version="1.0" encoding="utf-8" ?>
   <modelset packtime="2006-09-10 12:00:00">
      <model>
         <name>Mary Zhang</name>
         <filepath>c:\VrRoot\Modelfile</filepath>
         <producer>Alina</producer>
         <texturenumber>4</texturenumber>
         <schemes>
           <scheme name="Winter clothes" isdefault="true">
                  <textures>
                    <texture name="Hat">
                    <relative_path>texturefile</relative_path>
                    </texture>
                    <texture name="Jacket">
                    <relative_path>texturefile</relative_path>
                    </texture>
                    ……
            </textures>
          </scheme>
              <scheme name="Summer clothes" isdefault="false">
                <textures>
                ……
           </textures>
          </scheme>
          ……
         </schemes>
       </model>
        <model>
       ……
       </model>
   </modelset>
```

5 CONCLUSION

The BiMM method is proposed to manage models in virtual reality research. This method describes models from two angles: one angle is based on model composition, which includes the model, model scheme, and texture;

another angle is based on model sort, each sort has its child sorts except leaf nodes. Based on this method, we designed a model database, an E-R data model, and five relational schemas. The main modules include model management, model sort management, model queries, model statistics, model registration into database as a whole, etc. Now the application has been used in model data management in virtual reality research, and it has proven to be very effective.

In the future, we will manage more model information in this model management application, such as action, sound, video, specifications, and so on.

6 ACKNOWLEDGEMENTS

We would like to thank the members of the Beijing Scene China Corp. for their work in the development of this model management application.

7 REFERENCES

Cheung, N.T., Lam, A., Chan, W., & Kong, J.H. (2005) Integrating images into the electronic patient record of the hospital authority of Hong Kong. *Computerized Medical Imaging and Graphics 29 (2-3),* 137–142.

Kari, B., Mester, A.R., Gyorfi, Z., Mihalik, B., et al. (2005) Clinical evaluation of multi-modality image archival and communication system in combination of WEB based teleradiology. *International Congress Series 1281,* 974–979.

Vogl, R., Berreck, M., Brauns, T., et al. (2005) The Innsbruck Advanced Image Management (AIM) project—a central archive and distribution system for radiology and multimedia data. *International Congress Series 1281*, 871–876.

Ying, W., Yingyun, H., Haichao, Z., & Zhiqiang, L. (2003) An overview of virtual reality technology. *Computer and Digital Engineering 30(3),* 41–44.