EMBEDDED GIS VECTOR DATA MODEL BASED on PDA

Xiaohua Tong [*] **Jin Zhang** [**] **Gen Tian** [**]

*** Department of Surveying and Geo-informatics, Tongji University, Shanghai, 200092**

**** Department of Surveying Science and Technology, Taiyuan University of Technology, Taiyuan Shanxi, 030024**

ABSTRACT

In this paper, vector data models are divided into two kinds: one is the object-oriented data organization model, the other is selection set based data model. In the object-oriented data models in embedded GIS, four models are presented; the advantages and disadvantages of these models are analyzed and compared. Based on the practical experiment, a suitable vector data model based on PDA is proposed.

KEY WORDS: Embedded GIS; Vector Data Model; Personal Digital Assistant (PDA)

1. INTRODUCTION

The complete GIS data model is thoroughly studied for desktop and workstation hardware based GIS. In above hardware environment, researchers rarely worry about memory allocation, data storage and data structure model. While with the rapid development of micro-electronic and photoelectron technologies, a new era has come to out life. Mobile computing technology such as mobile GIS based on personal digital assistant（PDA）has been used widely in location based service, navigation, web map service. Since smaller memory and storage (Usually 32M RAM and 32M ROM), lower CPU speed and smaller display screen (Usually for Palm, the size is 160 X 160，and for Windows CE, the size is 320 X 240), the data models used in desktop and workstation GIS are not suitable for embedded GIS on PDA. Therefore, the embedded GIS vector-based data model on PDA is focused to solve the conflict between the great amount of GIS data and the limits of PDA hardware platform, make the best of efficient data logical organization to make up the deficiency of hardware platform, and realize the fast visualization, query and analysis of spatial information. In this paper, vector data models are divided into two kinds: one is the object-oriented data organization model, the other is selection set based data model. In the object-oriented data models in embedded GIS, four models are presented; the advantages and disadvantages of these models are analyzed and compared. Based on the practical experiment, a suitable vector data model based on PDA is proposed.

2. OBJECT-ORIENTED DATA MODELS OF GIS

Currently vector data models of GIS are divided into two kinds: one is the object-oriented data organization model, the other is selection set based data model. These different data models have different advantages and disadvantages and different applications.

2.1 The First Data Model

In object-oriented data model, the geographic features are regarded as objects to deal with according to the object-oriented idea. The data and operations are all encapsulated in the objects, thus the maintenance of geographic features relies only on the required objects, while not affecting other features. In the data model, each geographic feature has a unique ID, coordinate sequence, color, line type and visible flag. According to the logical organization, the object-oriented data models can be divided into several kinds as follows.

The first model is shown in Figure 1. In this model, each graphic element is an object, therefore, point objects, line objects and area objects are all stored in a pointer array, and each object has a unique ID, coordinate sequence, layer, color, line type and visible flag. If the property of an object is needed to be modified, the object should be firstly found and then revised. The visible control of the objects depends on the member "m_CanDraw". And the control of object's layer relies on the member "layer". The detailed definitions of an object are described as follows:

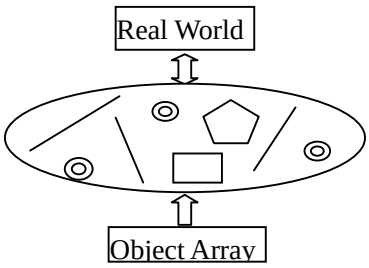

Figure 1 The First Model

Step 1: A double precise point, a double precise rectangle and a class named CEmRectangle are firstly defined to determine the boundary and operations of the graphic element. Then point object array is defined to store point objects that created by "typedef CArray<DOUBLEPOINT, DOUBLEPOINT&> CEmPointArray".

Step 2: A graphic element base class named CEmObject is then defined. Deriving from this base class, point class, line class and polygon class are defined. In line and polygon classes, a member from CEmPointArray is included to manage the coordinate sequences of line and area objects.

Step 3: In this model, the storage of point, line and polygon objects are all managed by their base class – CemObject that created by CTypedPtrArray<CObArray, CEmObject *> m_aObjects. Each times a point, line or polygon object is created, a corresponding point, line and polygon object will be added in the array m_aObjects.

In this model, the operation of an object will firstly search in object array m_aObject, after the object is found, the coordinate sequences of this object is obtained from the array m_pointArray, thus the object can be manipulated. The advantage of this model lies in the easy operation of graphic elements and modification of object property, so it is easy to be implemented. Meanwhile the disadvantage of this model is also obvious. When searching for a graphic element, all elements will be searched regardless the element classification and visibility, so its search efficiency is very low. Therefore, this model is not suitable for PDA.

2.2 The Second Data Model

The second model is shown in Figure 2. In this model, the real entities are categorized as point object, line object and polygon object. Each graphic element is regarded as an object storing in point, line and polygon object arrays according to its category, i.e., point element is stored in point object array, line element is stored in line object array and polygon element is stored in polygon object array. Thus, one object array in the first model

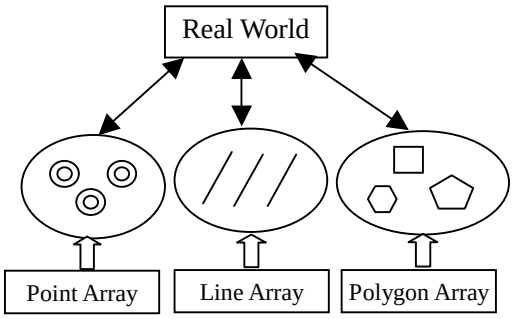

Figure 2 The Second Model

is divided into three object arrays in this model. The detailed definitions of an object are described as follows:

Step 1 and 2 are the same as the first model.

Step 3: After point class (CEmPoint) is derived from base class, point type object array (CpointPtrArray) is created by typedef CTypedPtrArray<CObArray,CEmPoint*> CpointPtrArray.

Step 4: Line type object array (CEmlinePtrArray) to store line class object (CEmLine) is created by typedef CTypedPtrArray<CObArray, CEmLine *> CEmlinePtrArray.

Step 5: Polygon type object array (CEmPogonPtrArry) to store line class object (CEmPolygon) is created by typedef CTypedPtrArray<CObArray, CEmPolygon *> CemPogonPtrArry.

Each graphic element is also operated through each object, while different from the first model, the object is searched from different object arrays. For example, in the operation of point objects, the size of point object array CpointPtrArray is firstly obtained, then each object is iterated from the array. The operation of line object and polygon object are dealing with line object array (CEmlinePtrArray) and polygon object array (CEmPogonPtrArry). The advantage of storing point, line and polygon objects in different arrays is obvious. The efficiency of searching point, line and polygon is rather high. The reason lies in that when searching a point object, only the point object array is dealt with no regarding to line and polygon object arrays. While the disadvantage of this model is that if an object is invisible status, it is still to be searched and this will certain lost some times.

2.3 The Third Data Model

The third model is shown in Figure 3. This is a layer-based model. In this model, the graphic element is managed with layers as units, i.e., a layer an object composed of the same kind objects, and the overlay of different layers constitutes the whole geographic landscape. Each layer object is managed by a layer object array, and point objects, line objects and polygon objects are all stored in the layer object. Each graphic element has no property member of IsVisible, while the layer object has this property member. The point, line and polygon objects belong to one layer are all stored in the layer object array, while the point, line and polygon objects belong to another layer are all stored in that layer object array. The detailed definitions of an object are described as follows:

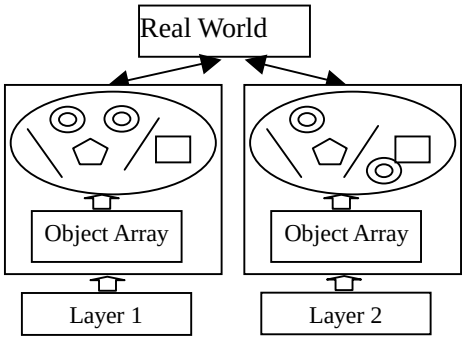

Figure 3 The Third Model

Step 1 and 2 are the same as the first model.

Step 3: A layer class is defined as CmapLayer, in which the member controlling the

layer visibility (m_CanDraw), layer name (m_LayerName) are included. And object array named m_aObjects to manage graphic elements is created by CTypedPtrArray<CObArray, CEmObject *> m_aObjects. The point, line and polygon objects belong to this layer are added in the array m_aObjects.

Step 4: The layer object array named m_aLayers to store layer objects is created by CTypedPtrArray<CObArray, CMapLayer*> m_aLayers.

In this model, the graphic element is operated with the layer as unit. When searching an object, the size of the layer object array m_aLayers is firstly obtained, then each layer is iterated from this array, thus, the object array is got from the corresponding layer object array, and the needed point, line and polygon object is therefore obtained from this object array. This model has great advantage. When searching a graphic element, the closed layers that are invisible status will not be operated, these will then greatly improve the efficiency. The disadvantage of this model lies in that it increases the difficulties in graphic management, especially when the layer of some graphic elements should be changed. For example, when a graphic element is changed from one layer to another layer, this element should be firstly deleted from object array of the layer, and then this element should be added into a new object array of another layer. This will follow a series of problems such as object ID management. Further, since this model does not distinguish point, line and polygon object, when only a point object is searched, line and polygon objects will be searched though it is not necessary.

2.4 The Forth Data Model

The forth model is shown in Figure 4. This model is a layer-indexed model that is similar to the second model, while the difference is that the common properties of point, line and polygon are abstracted to create a new layer object. For example, the properties of a line include ID, Points (to store the coordinates), Layer, Color, Line type and Visible Flag etc. And another point has the same property; therefore, a new class named Clayer is created whose members include Color, Line type and Visible Flag, and a series of layer object instance of the class Clayer are created to represent the different layers. The graphic elements belong to this layer have the properties of Color, Line type and Visible Flag. This actually builds a layer index for each graphic element. The detailed definitions of an object are described as follows:

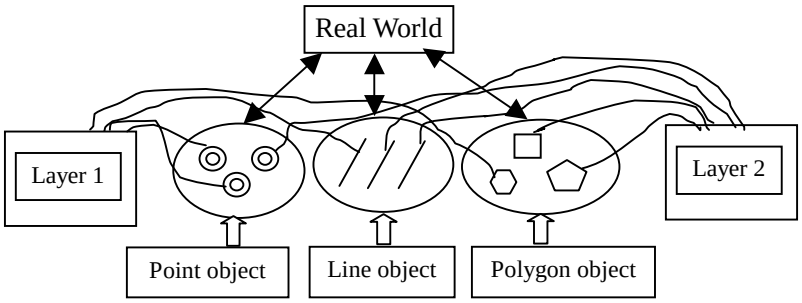

Fig.4 The Forth Model

Step 1, 2, 3, 4 and 5 are the same as the third model.

Step 6: A layer index class named Clayer is defined. This class has the properties of Color, Line type, Visible Flag and layer code. The layer index objects is stored in the object array named m_indexlayer, which is created by CTypedPtrArray<CObArray, CLayer*> m_indexlayer.

The operation of this model is the same as the third model. The difference lies in that the properties of Color, Line type and Visible Flag of each object are not stored by each object itself but by layer index object. Each graphic element has layer code property, and each graphic element obtains Color, Line type and Visible Flag according to the corresponding layer code in layer index objects. The advantage of this model lies in that the data storage volume becomes less since the properties of each element stored before are stored in layer objects. On the contrary, this is also the disadvantage of this model. Since each graphic element does not keep the properties such as Color and Line type, it uses only the values of the layer that the element belongs to.

2.5 The Fifth Data Model

The fifth model is shown in Figure 5. Analyzing the advantages and disadvantages of above model, this model is presented that is believed to be a better model. First, this model is a layer-based model, secondly, in each layer, the objects are divided into point, line and polygon models, thirdly, the common properties of the graphic elements are abstracted as a layer class, therefore, it is a layer-indexed model. The detailed definitions of an object are described as follows:

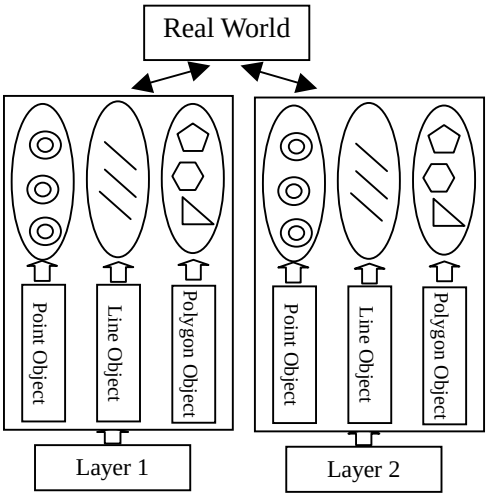

Figure 5 The Fifth Model

Step 1 and 2 are the same as the first model.

Step 3: A layer class is defined as CmapLayer, in which the member controlling the layer visibility (m_CanDraw), layer name (m_LayerName), Color and Line type are included. In the layer class, point, line and polygon object arrays are created as its members. The point object named CpointPtrArray is created by typedef CTypedPtrArray<CObArray,CEmPoint*> CpointPtrArray, the line object named CemlinePtrArray is created by typedef CTypedPtrArray<CObArray, CEmLine *> CemlinePtrArray, and the polygon object is created by typedef CTypedPtrArray<CObArray, CEmPolygon *> CemPogonPtrArry. Thus, the point, line and polygon objects are added into the corresponding object arrays.

Step 4: The layer objects are stored in layer object array named m_aLayers created by TypedPtrArray<CObArray, CMapLayer*> m_aLayers.

In this model, the operation of the graphic elements is also based on layer. Firstly, the size of layer object array is obtained, and then each layer object is got from the array. Secondly, from each layer object, the sizes of point, line and polygon object arrays are obtained. When dealing with point object, the point object array is iterated, and then each point object is obtained. When dealing with line and polygon objects, the line and polygon object arrays are iterated. Furthermore, each layer object keeps the common properties of the graphic elements, avoiding that all graphic elements in the same layer keep the repetition information.

This model has the advantages of all above models. When one layer is closed, then the

objects in this layer will not be iterated. When a line object is being searched, then the point and polygon objects will be not iterated. Meanwhile, the repetition information will not be stored. For example, assuming that there are two layers as shown in Figure 5 and the Layer 2 is closed, when searching a point object in current layer, then the point object array in layer 1 will be iterated, and all the objects in layer 2 and the line and polygon objects in layer 1 will not be iterated. Therefore, the searching efficiency of this model is rather high. And since the properties of Color, Line type of all graphic elements in each layer is stored only once in the layer object, the data volume will be less. The disadvantage of the model lies in its complex in data management.

3. RECORD SET BASED DATA MODEL OF GIS

The data record set based data model is actually like the object-oriented data models discussed above. The difference lies in that this model can deal with many objects based on the record sets. The model is shown in Figure 6. The point objects are managed by point object arrays. While the management of line and polygon objects are different. The line object and polygon object can manage many lines and polygons based on the record set. The advantage of this model lies in that the data record set can manage lots of line and polygon objects at the same time. The operations of the line and polygon objects such as add, deletion and modification become much easier. The disadvantage of this model is that it is more complex than the object-oriented data model. And the searching efficiency is much lower since it needs to iterate all the graphic elements.

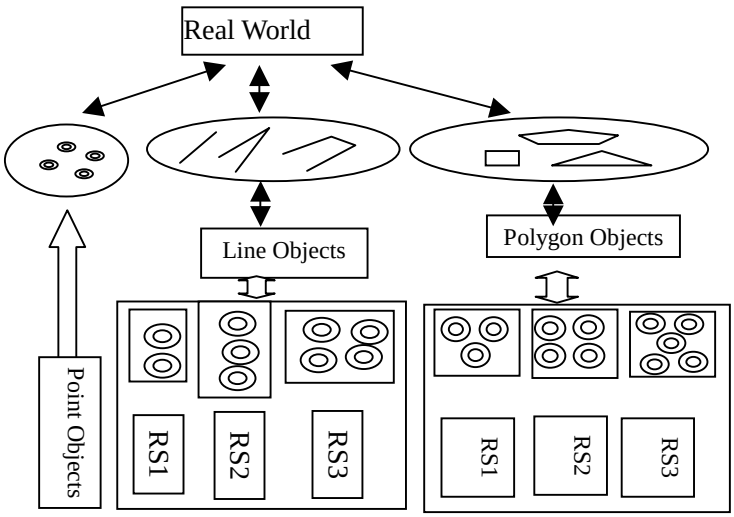

Figure 6 The Record set Based Data Model

4. CONCLUSIONS

Through experiment studies and programming tests with eMbedded Visual C++3.0 on Pocket PC PDA, the conclusions are reached that different data models have different advantages and disadvantages during different applications and different environments. In general, the fifth model presented in this paper is a better model that has much more completeness and adaptability.

ACKNOWLEDGEMENTS

The work described in this paper was substantially supported by the National Natural Science Funds of China (Data and Multi-scale Data Fusion and Uncertainty Analysis in GIS) and the National 863 Project (Number: 2002AA134050).

REFERENCES

Gong, J.Y. (2001). *Fundamental of Geographic Information System*, Beijing: The Surveying and Mapping Press.

Li, L.Q., Li, C.M. and Lin, Z.J. (2002). "The study on vector based raster data structure for PDA", *Computer Application Research*, 7, 118-120.

Chen, J.C. (2000). *Development of GIS with Visual C++*, Beijing: Electronic Industry Press.