Design and Implementation of a General Purpose 2D CAD System

Tunhua Wu and Qinqin Shen
School of Information & Engineering, Wenzhou Medical College, Wenzhou, China
Email: wth@zjut.edu.cn

Changle Zhou
School of Information Sci. & Tech., Xiamen University, Xiamen, China
Email: dozero@xmu.edu.cn

Ping Wang
Dept. of Environmental Science, Wenzhou Medical College, Wenzhou, China
Email: fruitful@xmu.edu.cn

***Abstract* - A general purpose 2D CAD system was established by Object-Oriented technology. Key technologies for realizing the system were introduced. The methods for forming shape container, manipulating shape container and constructing graph template library were described in detail. Basic shapes, such as line, rectangle, ellipse, polygon, B-Spline curve and textbox, can be grouped to be a shape container, and the shape containers can also be grouped to form a more complex one. The hierarchy of shape container is a Tree. Hence, to manipulate shape container is to access the container recursively. The Redo and Undo functions were realized by serializing canvas status to temporary files. And all of the canvas statuses were managed by undo-stack and redo-stack. The purposefully formed shape container can be reformed to a graph template, and the graph templates of the same type can be stored as a library. Different graph template library corresponds to different application. Various types of graph template library ensure the generality and flexibility of this system. Experimental results showed that the proposed methods are effective. This system can be applied to a lot of fields, such as construction, machinery and electronics.**

***Index Terms* - CAD, graph element, Object-Oriented technology, tree structure, B-Spline curve**

I. INTRODUCTION

Computer-aided design (CAD) software provides the user with input-tools for the purpose of streamlining design, drafting, documentation, and manufacturing processes [1-4]. CAD technology has been developed for decades, and there have been much excellent computer-aided drawing software, such as AutoCAD and Microsoft Visio. And a lot of second development technologies based on the mature products appeared [5-8]. However, as the commercial software is confidential, and many key technologies of CAD, such as the methods for forming shape container and the method for manipulating shape container (to rotate, move, resize or serialize shape container), are rarely mentioned or described not detail enough in teaching materials or papers. So, the key technologies mentioned above should be detailed in this paper. The research goal is to establish a general purpose 2D CAD system for fulfilling the needs of a variety of area, such as construction, machinery and electronics. In our system, basic shapes can be grouped to a shape container, and the shape containers can also be grouped to a more complex shape container. Besides, shape container can be reformed to a graph template, and the graph templates of the same type can be stored as a library. Therefore, shapes or graph templates in our system are reusable. Various types of graph template library ensure the generality and flexibility of our system.

Corresponding author: Wang Ping

II. DATA STRUCTURE

Six simple and basic shape classes were defined in this system. Fig.1 shows the relationship among them.

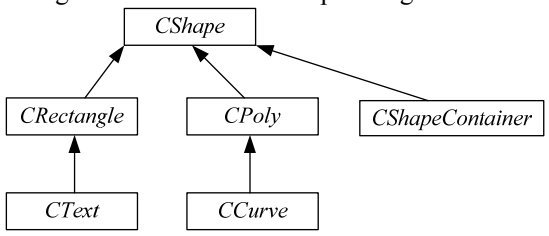

Figure 1. Hierarchy diagram about shape classes.

A. Base Cass of All Shapes – CShape Class

CShape class is derived from CObject (one of the most important Microsoft Foundation Classes). It encapsulates the common properties and methods of all kinds of 2D shapes. The main properties of CShape are: shape style (type), coordinates and size, line color and style, fill color and style, rotate angle, etc. An enumeration data type 'ShapeStyle= {LINE, RECTANGLE, ROUNDRECT,

doi:10.4304/jcp.7.3.666-671

ELLIPSE, TEXT, POLYGON, CURVE, CONTAINER}' was defined to specify the possible value of shape style property '*m_shapeStyle*' in shape entities. The main methods are: drawing method, serialization method, control-handle related methods, shape manipulation methods, intersection detection, etc. All the methods are defined as virtual functions.

B. Shape Classes Derived From CShape

(1) Line, square, rounded rectangle, ellipse and circle are classified into CRectangle class, because they have many very similar properties and methods. Line can be taken as a diagonal of rectangle, so lines can be represented by rectangle. When the time to draw a line, just invoke the function for drawing line according to the shape style. Similarly, the bounding box of ellipse, circle or rounded rectangle is rectangle, so they can be represented by rectangle too. The distinguished property of CRectangle is coordinates of rectangular vertexes. And the position of rectangle is defined as left-top corner and right-bottom corner coordinates. Though the main methods of CRectangle have the same form with the corresponding methods in CShape, the implementation processes are totally different (CRectangle-oriented virtual functions).

(2) CPoly is a polygon class. The distinguished property of CPoly is the coordinates of polygonal vertexes which are stored in a CPoint array. Most of the methods of CPoly are different with CRectangle, especially in the zoom and rotation aspects. Closed polygon differs from open polygon only in one line between the first vertex and the last vertex. So, closed polygon and open polygon subject to the same class.

(3) CShapeContainer is a compound shape class. Any basic shape can be grouped into a shape container, and any shape container can also be grouped into a more complex one. So, the data structure of CShapeContainer is a Tree. We defined a list of CShape type '*sonList*' in this class to store the sub-shapes of container (i.e. a tree node). In order to identify whether a shape is container or non-container, a new property of Boolean type '*m_bIsContainer*' was added to CShape. If *m_bIsContainer* equals 'true', the shape is a container. In addition, the rectangular bounding box of container's sub-shapes should be recorded in this class. The manipulation methods of shape container are described in section IV.

C. CText Class and CCurve Class

(1) CText is a textbox class derived from CRectangle. As the border of textbox is rectangular (including elliptical border, square border and so on), CText share all of the properties and methods in CRectangle. The distinguish properties of CText are: text color, font related properties, rotate angle of text, etc.

(2) CCurve is a class of B-Spline curves derived from CPoly. The vertexes of polygon are exactly the control-points for B-Spline curve. The curve can be reformed to various patterns by moving the control-points. The biggest difference between CCurve and CPoly is the drawing method. In polygon, vertexes are connected by lines. While B-Spline curve is controlled by vertexes. In addition, closed polygon can generate closed B-Spline curve. Cubic B-Spline curve requires at least 4 control points to control a piece of the curve. Suppose the number of control points is n ($n \geq 4$), then there are n-3 pieces of smoothly connected curves in the spline curve. As shown in Fig.2, $C_i(u)$ is the i^{th} piece of the spline curve, and P_{i-1}, P_i, P_{i+1} and P_{i+2} are the control points to $C_i(u)$. In programing process, Δu=1/8 and 8 connected lines are used to simulate $C_i(u)$.

$$C_i(u) = (1/6)\cdot\begin{pmatrix} u^3 & u^2 & u^1 & 1 \end{pmatrix}\cdot\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}\cdot\begin{pmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{pmatrix}, u \in [0,1]$$

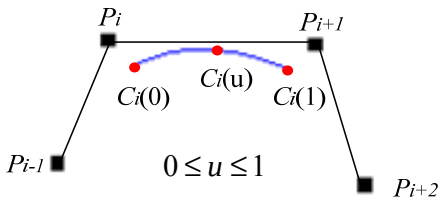

Figure 2. A piece of cubic B-Spline curve controlled by 4 points.

III. GROUP OR UNGROUP SHAPES

Suppose the whole shapes on canvas are stored in a list '*m_shapeList*', and the selected shapes on canvas are stored in a list '*m_shapeSelectedList*'. To group some shapes, we must first construct a CShapeContainer entity and insert all the shapes to be grouped into the container's *sonList*. Then, insert the container into *m_shapeList* and remove the shapes to be grouped from *m_shapeList*. Finally, empty *m_shapeSelectedList* and insert the newly constructed container into *m_shapeList*. To ungroup a container, on the contrary, we must put all the sub-shapes of container back to *m_shapeList*, and then remove the container from *m_shapeList* and *m_shapeSelectedList*.

IV. SHAPE MANIPULATIONS

The main shape manipulations are: selection, moving, rotation, scaling, copy, paste, clone, undo, redo, graph layer modification, properties modification and serialization. Several key solutions to realize the manipulations are introduced in this section.

A. Interaction Model Based on Drawing Tool Classes

Drawing tool classes are responsible for processing mouse events. The mouse events occur on canvas view will be transferred to one of these tools for processing. Fig.3 illustrates the relationships among these classes. CDrawTool is the base class of them. Through mouse interactions we can manipulate shapes.

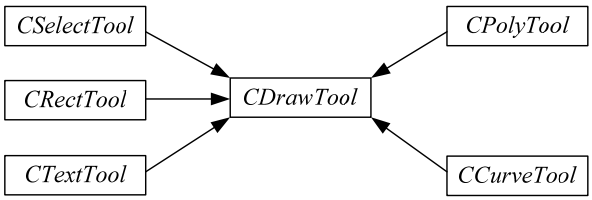

Figure 3. Diagram about CDrawTool and the classes derived from it.

Each drawing tool has four mouse event response functions: OnMouseMove, OnLButtonDown, OnLButtonUp and OnLButtonDblClk. But the function in different drawing tools has different result. Utilizing the mouse event response functions of CRecTool, we can draw CRectangle shapes on canvas. Similarly, utilizing the mouse event response functions of CPolyTool, we can draw CPoly shapes on canvas. Utilizing the mouse event response functions of CSelectTool, we can select and manipulate the shapes on canvas.

Fig.4 illustrates the interaction model of this system. Firstly, mouse events occur on canvas will be captured by CDrawView. Then, the events will be transferred to a selected drawing tool. Third, an event response function of the drawing tool will be activated, and the function will utilize some manipulation methods of the selected shapes to handle the event. Below is an example about shape translation for demonstrating how the interaction is done in this system.

(1) Step 1. Select a shape on canvas. A WM_LBUTTONDOWN event occurs when left mouse button is clicked. Then, the event invokes CDrawView::OnLButtonDown function to select a CSelectTool entity for handling the event and invoke OnLButtonDown function defined in the tool. Next, current mouse position will be used to judge whether the mouse point is in the range of certain shape by calling intersection detection method, and the intersected shape is taken as selected. Finally, insert the selected shape into *m_selectedShapeList* and plot the control-handles of the shape. Here, control-handles are small squares plotted on the vertexes or center position of a shape. Dragging the control handle will scale, rotate or modify the shape.

(2) Step 2. If left mouse button is not released, once the mouse is moved, a WM_MOUSEMOVE event occurs. A CSelectTool entity will handle this event by invoking its OnMouseMove function to get the possible new position of the selected shape. And a temporary shape on possible new position should be plotted whenever the mouse is moved. But the real coordinates of the selected shape should not be changed until left mouse button is released.

(3) Step 3. When left mouse button is released, a WM_LBUTTONUP event occurs. A CSelectTool entity will handle this event by invoking its OnLButtonUp function to translate the coordinates of selected shape to the target position. In addition, coordinates translation can be done by invoking the translation method of the selected shape.

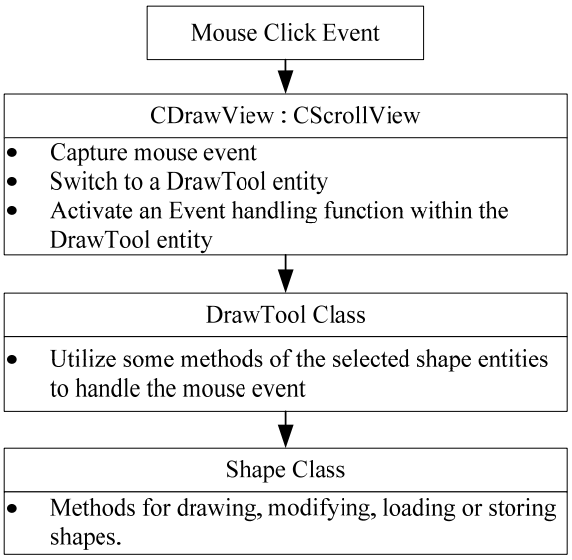

Figure 4. The interaction model of this system.
It illustrates the response process for mouse events.

B. Methods for Manipulating Shape Container

As the data structure of shape container is Tree, to manipulate shape container is to recursively manipulate all the descendant shapes of that container. The methods for different kinds of manipulations are very similar. So, an example of the manipulation methods can illustrate the problem. Here, we take the serialization method for shape container as an example, and the implementation was illustrated on Fig.5. Each shape has serialization method for storing its important properties to archive or loading its important properties from archive.

```
void CShapeContainer::Serialize(CArchive &ar){
  Shape::Serialize(ar); // Serialize container's basic properties
  int num=0, i, j;
  if (ar.IsStoring()){   // If the Archive is in Storing mode
    std::list<Shape*>::iterator itr = this->sonList.begin();
    for (; itr != this->sonList.end(); ++itr) ++num;
    // Record the number of sub-shapes in the container
    ar << num;
    for(itr = this->sonList.begin();itr !=this->sonList.end();++itr){
      j = (*itr)->m_shapeStyle;
      // Record the type of each sub-shape in the container
      ar << j;
     // Recursively serialize the sub-shape if it is also a container
      (*itr)->Serialize(ar);
    }
  }else{   // If the Archive is in Loading mode
    // Get the number of sub-shapes in the container
    ar >> num;
    for(i = 0 ; i < num; i++) {
      Shape * pShape = NULL;   // Construct a new sub-shape
      ar >> j;   // Get the type of each sub-shape in the container
      switch(j){
      case RECTANGLE:   pShape = new CRectangle;   break;
```

```
        case LINE:   pShape = new CRectangle(LINE);   break;
        case ELLIPSE:
                pShape = new CRectangle(ELLIPSE);   break;
        case ROUNDRECT:
                pShape = new   CRectangle(ROUNDRECT);   break;
        case POLYGON:   pShape = new CPoly;   break;
        case CURVE:   pShape = new CCurve;   break;
        case TEXT:   pShape = new CText;   break;
        case CONTAINER:   pShape = new CShapeContainer;
        }
        pShape->Serialize (ar);   // Recursively serialize the newly
                        // constructed shape if it is also a container
        this->sonList.push_back(pShape);   // Add the newly
         // constructed shape to container's sub-shape list : sonList
    }
  }
}
```

Figure 5. Method for serializing shape container. It may be a recursive process for the data structure of shape container is a tree.

C. Solution for Rotating Shapes

Shapes can be rotated by geometric transformations. Suppose the center point of a shape is O, and each control point of the shape must turn around O. There is a special control point P of the shape, and the distance of $\overrightarrow{OP}$ is a constant (e.g. 100). The direction of rotation is fixed to counter-clockwise. Tilt angle of the shape is represented by the angle between $\overrightarrow{OP}$ and x-axis. And the tilt angle must be saved to archive for the sake of reconstructing the shape in loading process. If the control point P was dragged to a new position P', we could get the rotation angle $\triangle\theta$ by calculating the angle of $\angle POP'$. Then, coordinates of the other control points of the shape can be calculated by the following formula:

$$\begin{pmatrix} x2 \\ y2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x0 & -y0 & 1 \end{pmatrix} \bullet \begin{pmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) & 0 \\ -\sin(\Delta\theta) & \cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x0 & y0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x1 \\ y1 \\ 1 \end{pmatrix}$$

Here, (x_0, y_0) is the coordinate of O, (x_1, y_2) is the coordinate of an old control point, and (x_2, y_2) is the coordinate of new control point after rotation. Once all the coordinates were updated, the rotated shape can be constructed. However, there is a special case: textbox. The bounding box of text can be rotated by the method mentioned above. As to the text, we should modify a text-related property of Windows API: LOGFONT::lfEscapement. For example, if LOGFONT::lfEscapement = 100, the tilt angle of text is $+10^{\circ}$ (i.e. count unit is 0.1°).

D. Solution for Realizing Undo and Redo Functions

Undo and Redo functions of this system were realized by utilizing two stacks (undo-stack & redo-stack) and serialization method. Whenever a shape is manipulated or a graph template is added to canvas, all the critical properties of document and canvas view should be serialized to a temporary canvas status file. Meanwhile, serial number of the temporary file should be pushed to undo-stack and redo-stack should be cleared. Suppose the maximum depth of undo operations is H, then the length of undo-stack is H too. When undo-stack is too full to accommodate a new element, the bottom of the stack should be removed. Hence, H temporary files can satisfy the needs of Undo and Redo functions. When the time to undo an operation, pop the top element of undo-stack and push it to redo-stack. Then, get the top element of undo-stack S_i (i.e. serial number of the previously accessed temporary canvas status file), and serialize the temporary file with a serial number of Si to document and canvas view (i.e. return to the previous canvas status). When the time to redo operations, pop the top element of redo-stack S_j and push it to undo-stack. Then serialize the temporary file with a serial number of S_j to document and canvas view (i.e. change to the latter canvas status).

E. Solution for Realizing Shape Layers

Shapes are displayed on canvas according to their sequences in *m_shapeList*. The tail element of *m_shapeList* is displayed on the top layer, and the head element of *m_shapeList* is displayed on the bottom layer. The shapes may be overlapped, so sometimes we want to put some shapes forward or back. To put a shape to bottom layer is just to move the shape to the head of *m_shapeList.* To put a shape to top layer is to move the shape to the end of *m_shapeList*. To put a shape to lower layer is to exchange its position with the previous shape in *m_shapeList*.

V. Graph Template Library

Graph templates are the shape containers which had been translated and scaled. And the templates are stored in a library and displayed in another pane of system interface (compared with canvas pane). Suppose the bounding box of shape container is M, and the top-left corner of M is P, to form a graph template, we must first subtract the coordinate of P from all the coordinates of container and its descendant shapes. By doing so, the coordinate of P should be changed to (0, 0), and the coordinates inner the container should be transferred to the coordinates relative to P. Then, limit the length and width of the container to D by down scaling. Finally, insert the translated and scaled container into a graph template list '*m_templateList*'. When the time to use template, just add the coordinates of a random point in canvas to the coordinates of that template, and then insert the template into *m_shapeList*. In this system, canvas is an entity of CDrawView and the pane for displaying graph template library is an entity of CTemplateView. CDrawView and CTemplateView are all derived from CView. In order to realize the communication between canvas and graph template library, *m_shapeList* and *m_templateList* are stored in the same document. The serialization method of graph template library and canvas are almost the same.

VI. Experiments And Results

This system was developed under Microsoft Visual C++ 2010 environment. The basic drawing methods come from Microsoft GDI+ Library.

A. Experimental Results

Fig.6 shows a NE5532 audio preamplifier circuit diagram plotted by this system (NE5532 is a high-performance operational amplifier combining excellent dc and ac characteristics designed by TI). To plot the diagram, we must first construct some electronics templates, such as resistance template, capacitance template and amplifier template, just by grouping some simple shapes. Hence, all the electronics templates form an electronics library. Then, the necessary templates for the diagram are added to canvas. Finally, the diagram is plotted by moving, rotating or resizing the templates on canvas.

Fig.7 illustrates two car models plotted by cubic B-Spline curves. We can get various models simply by dragging any control point of the curves. Fig.8 shows a house model plotted by this system too. Building-related templates, such as window templates and door template, were constructed and placed on the left side of the system. Roof of the house model is a polygon filled with black color. In addition, each shape has magnetic effect to connecting line. Once the connecting line is close enough to the target shape, the junction would be automatically set to the point of intersection between the connecting line and shape.

The examples mentioned above show the expansibility and flexibility of this system. All the shapes in this system are reusable. We can construct various graph template libraries to satisfy the needs of various applications.

B. Experimental Analysis

This system realizes many key technologies for constructing 2D CAD system. And the reusability and expansibility were achieved. It has the following features or advantages compared with the famous 2D CAD systems (such as AutoCAD and Microsoft Visio):

(1) Cubic B-Spline curve was embedded. We can get any kind of curve by adjusting the positions of control points. When the control point is not enough, any number of control points can be added to the curve. Likewise, if a control point was redundant, just remove it. So, the curve is deformable and flexible.

(2) A simple but efficient solution for realizing Undo and Redo was proposed for the first time. We can simply serialize canvas status to temporary files and manage all the status by two stacks (i.e. undo stack and redo stack).

(3) The system takes up less memory and hard-disk space, and the running speed is faster. The size of this system is only 76KB, while Visio takes up more than 100MB hard-disk space.

However, compared with the famous 2D CAD systems, there are some deficiencies in this system. The main deficiencies were described as follows:

(1) It has no compatibility with other CAD system or office software (such as Microsoft Word and PowerPoint). The design cannot be modified out of this system. This problem may be solved by utilizing the second development interface of other system, such as Microsoft Office VBA.

(2) The fill-modes of closed-shape cannot be customized. The system provides the most common fill-modes, such as opaque, transparent, horizontal line and vertical line. But it cannot fill picture or customized shape. To solve this problem may involve seed-fill algorithm and fractal algorithm.

(3) It has no second development interface. This problem may be solved by studying the second development interface of AutoCAD.

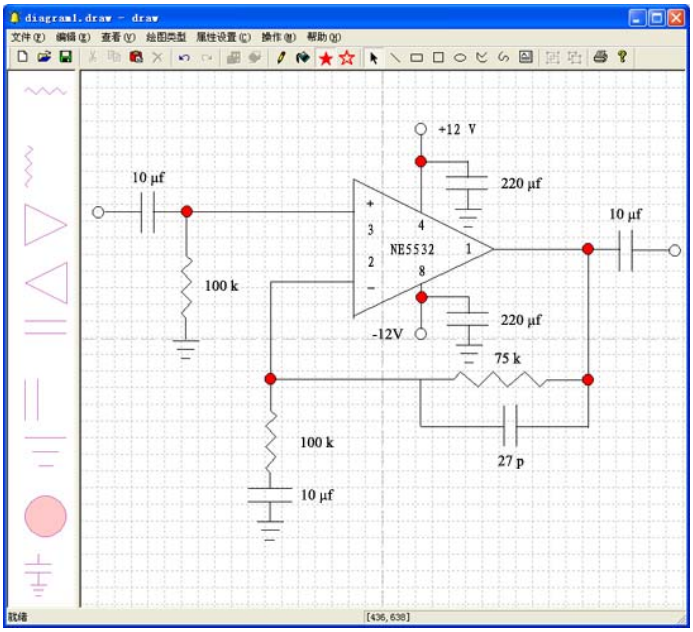

Figure 6. A NE5532 audio preamplifier circuit diagram plotted by this system. Left part of the software interface is electronics template library, and the right part is canvas.

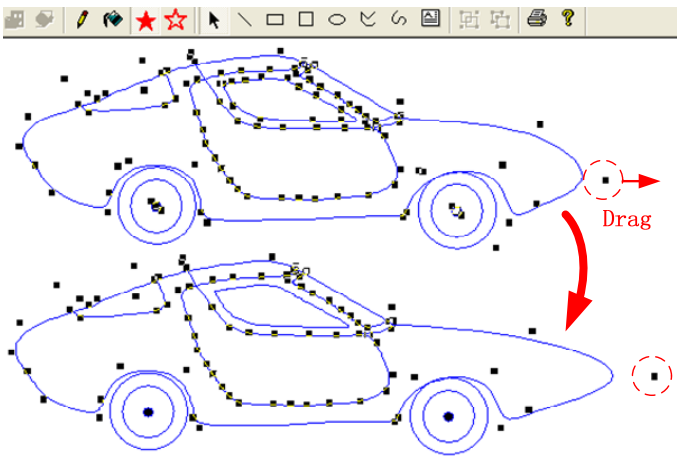

Figure 7. Two car models plotted by cubic B-Spline curves. We can get various models by dragging any control point of the curves.

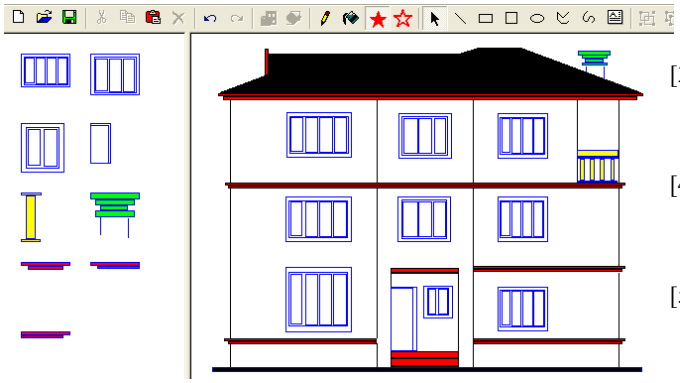

Figure 8. A house model plotted by this system. Building-related templates were constructed and placed on the left side of the system.

VII. CONCLUSIONS

The design and implementation of a general purpose 2D CAD system was introduced in this paper. The method for manipulating shape container and the method for constructing graph template library were described in detail. Various graph template libraries illustrate the reusability and flexibility of our system. The experimental results prove the effectiveness of the proposed methods.

ACKNOWLEDGEMENT

This work is supported by the Natural Science Foundation of China (No.11005081, 60873179), Natural Science Foundation of Zhejiang Province (No.Y1110322), Scientific Research Project of Zhejiang Education Department (No.Y201016244) and the Scientific Research Project of Wenzhou Medical College (No.QTJ09009).

REFERENCES

[1] Wiki, "Computer-Aided Design," *Wikipedia.com*, April 2011. [Online]. Available: http://en.wikipedia.org/Computer-aided_design. [Accessed:April 7, 2011].

[2] C.E. Kicklighter and W.C. Brown, *Drafting & Design: Engineering Drawing Using Manual and CAD Techniques (7th Edition)*, USA: Goodheart-Willcox Co, March 2008.

[3] M.B. Niu, D.M. Xu and H.F. Niu, "Research and Implement of Visible Diagram Elements with Object-Oriented Method," *Microelectronics & Computer*, vol.24, no.3, pp.191-194, March 2007 (In Chinese).

[4] T. Tetsuo, "Intelligent computer-aided design systems: Past 20 years and future 20 years," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol.21, no.1, pp.27-29, January 2007.

[5] A. Thakur, A.G. Banerjee and S.K. Gupta, "A survey of CAD model simplification techniques for physics-based simulation applications," *Computer-Aided Design*, vol.42, no.2, pp.65-80, February 2009.

[6] O. Ransen, *AutoCAD programming in C/C++*, USA: Wiley, April 1997.

[7] J. Sutphin, *AutoCAD 2006 VBA: A Programmer's Reference*, USA: Apress, September 2005.

[8] R.H. Wang, "Development of Parametric Drawing Program Based on AutoCAD VBA," *Proceedings of the 2010 International Conference on Computational Aspects of Social Networks*, IEEE Press, pp.757-760, September 2010.

Tunhua Wu received his Ph.D from Xiamen University in 2008. His research interests lie in the areas of digital image analysis, pattern recognition and computer graphics. Email: wth@zjut.edu.cn.

QinQin Shen received her B.S. from Wenzhou Medical College in 2011.

Changle Zhou is a professor of artificial intelligence institute at Xiamen University. He received his PhD from Peking University in 1990. His research interests lie in the areas of artificial intelligence, especially computational linguistics, brain theory and computer arts with an emphasis on computational models for metaphor, awareness and creativity. His project works lie in intelligent techniques about Chinese traditional medicine information processing. Email: dozero@xmu.edu.cn

Ping Wang received her Ph.D from Xiamen University in 2009. Her research interests lie in the areas of computational chemistry and analytical chemistry. Email: fruitful@xmu.edu.cn.