Some Remarks on Developing Application Programs in Visual

K. Y. Cheng, M. S. Hwu, and S. Y. Hsu

Institute of Information Science, Academia Sinica
and
Computer Science and Information Engineering Department,
National Taiwan University, Taipei, Taiwan, R. O. C.

ABSTRACT

A visual programming environment for users to develop their own defined icon-based Chinese input methods is presented. The purpose is to discuss some interesting remarks on developing application programs dealing with icon objects. In this paper, we show that Chinese characters are concrete icons, which are representable as webs (decomposition graphs), a grammar that generates these webs is a context-free web grammar, and so a Chinese input simulator is a machine that accepts these webs. Developing such a simulator can be regarded as the construction of an user-defined application program. We show that a form-based visual approach can be used to specify the simulation behaviors for defining and evaluating icon-based Chinese input methods, although the visual objects dealt with are concrete icons.

1. INTRODUCTION

The programming technique that utilizes certain interactive devices such as screen cursor, joystick, mouse, touch screen, etc. and certain screen operations such as menu, window, form, diagram, icon etc. has gained widespread popularity in personal computer systems and workstations recently. Under such a programming environment, an user who possesses with little or no knowledge of traditional programming languages or systems can effectively develop his/her own application programs in visual. In other words, this programming technique requires the users merely to express their own expertise and knowledge in terms of certain visual objects and relationship between objects. For instance, an office person, with whatever his/her operational knowledge on office forms and relationship between forms, can develop application programs to fit his/her needs under a visual programming environment for office informational use.

There are a few visual objects which can be described in

terms of form-based, diagram-based, or icon-based representations. They have been used extensively in a variety of applications such as office information [19, 20, 21, 24], computer-aided instruction [7, 8, 9], computer-aided design [1], data base interface [13, 14], computer graphics [10, 11], image handling [5, 15, 23], scene understanding [6], ideographic language understanding [4], ..., etc. This reveals a great portion of human knowledge representation can be expressed visually. As we examine closely to these applications, we may find a common fact that form-based or diagram-based or icon-based visual objects can be represented in graphs.

Therefore, conceptually we may view the expression of human knowledge in visual as a construction of labeled graphs, each labeled node represents a visual object and each labeled arc represents a relationship between objects. For example, to learn the ideographic formation of certain Chinese characters like 明 (bright), the character can be viewed as a decomposition graph which has 日 (sun) and 月 (moon) as its labeled nodes and part-of and left-of predicates as its labeled arcs. Thus, based upon this fact, the graph can be used to encode the character and conversely the decoding can be used to identify the character. An immediate application of this observation is to develop a Chinese input method by this notion of unique decoding of characters. One may also think of developing a software program to simulate what an user might have in mind about what a Chinese input process looks like and how good the method will be. This is indeed a typical case where the notion of visual programming can apply, because this simulation software can be sketched visually. The sketched simulator can provide Chinese character analysts (experts) to simulate each of their own Chinese input methods in a manner of what-they-sketched-are-what-they-get. But, how can the experts express and simulate their Chinese input methods in a visual programming environment? How can they describe the

evaluation procedures to know the efficiency of their designed methods? In this paper, we shall use the development of Chinese input simulator as an example to discuss how to develop certain types of application programs in visual.

In the following, we shall divide our discussion into sections. Section II gives a formal definition of visual language and shows that the ideographic formation of Chinese characters is an icon-based visual language. Section III shows that the decomposition graphs with the concrete icons as labeled nodes and the part-of predicates and the spatial relations as labeled arcs can be regarded as sentences generated by a context-free web grammar. Section IV describes what are the icon-based Chinese input methods and some factors that affect their efficiency so that requirements for the users to develop their application programs are specified. Section V shows a visual programming environment to support the development of a Chinese input simulator. Section VI shows some illustrated processes in developing the application programs.

2. DEFINITIONS

In this section, we shall give the visual language a formal definition which is based on the notion of the labeled graphs. Since a visual object may be either a primitive or a compositive one, we shall use a decomposition graph to represent it. A decomposition graph may be regarded as a representation of sentential forms of the visual language. Therefore, the visual language may be formally defined as

$$L_V = \{ x \mid x \text{ is a graph representation which can be generated by a visual grammar } G_V \},$$

the visual grammar G_V is a four tuple

$$G_V = (V_N, V_T, P, S)$$

where V_N is a set of nonterminals (compositive visual objects), V_T is a set of terminals (primitive visual objects), P is a set of production rules, and S is a set of starting symbols.

The language L_V is called a t-based visual language if the representation of visual objects used in the decomposition graph is t-based. Among them, form-based[7,8,19,20,21,24,25],

diagram-based [2,12,16], and icon-based [15,17,23] visual languages are the most often seen. For example, the ideographic formation of Chinese characters is an icon-based visual language. Fig. 2.1(a) shows the formation of the character 明 as a decomposition graph and Fig. 2.1(b) shows the visual grammar with $V_N = \{ \text{明} \}$, $V_T = \{ \text{日}, \text{月} \}$ and the production rule that generates 明. However, in practice, more than one t-based (as we shall see in later sections, we mix form-based and icon-based) visual languages may be used simultaneously in the development of an application program.

Fig. 2.1 The ideographic formation of the Chinese character 明
(a) the decomposition graph of 明,
(b) the grammar that generates 明.

The process that utilizes visual languages to construct an application program such that the constructed result is acceptable by a machine is called visual programming. The environment under which the users can freely develop their application programs by the visual programming technique is a visual programming environment. Notice that the visual programming technique is different from the conventional one in that it is a two-dimensional graphic construction. As we shall see in later sections, the visual programming technique is easier to construct certain application programs which are otherwise rather difficult to be implemented in conventional ways.

3. CONCRETE ICONS

As stated previously, an icon may be either primitive or compositive. A concrete icon is a compound of primitive icons with spatial relations and descendent relations only or itself a primitive icon. A concrete icon may be represented as a decomposition graph $D_C = (V, E)$, where the node set V contains both the primitive and the composite icons and the arc set E contains the part-of predicate and spatial relation arcs. The part-of predicate indicates the descendent relationship from one

Fig.3.1 shows such an example, where the Chinese character 腦 (brain) is represented as a decomposition graph. The terminal nodes are primitive icons and the rest are compositive icons, the part-of predicate arc is showing in an arrowed dash-line and the spatial relation arc an arrowed solid-line.

In order to understand the formation steps, we need to distinguish the part-of predicate from the spatial relations, and so we introduce a new graph called the derivation diagram as follows. A derivation diagram is obtained from a decomposition graph by preserving the inherited relationship of the spatial relations from an ancestor node to all of its descendant nodes. However, the readers are encouraged to consult [3,18] for a detailed procedure.

$$\alpha \rightarrow \beta, \gamma$$

where α and β are webs and γ is an embedding of β . In order to substitute a subweb α of a web ω by another subweb β , we have to specify how to embed β in ω in place of α . For example, the web grammar that generates the web is $V_N =$

$\{\text{腦}, \text{昏}, \text{血}, \text{〃}, \text{〃}\}, V_T = \{\text{月}, \text{‘}, \text{口}, \text{×}\},$
 $S = \{\text{腦}\},$ and $P =$ production rules shown in Fig. 3.3.

Fig. 3.3. The web production rules that generate the character 腦.

To derive web production rules from a derivation diagram, we shall consider two graphs. One is called the skeleton and the other is called the section. A skeleton is a subdiagram obtained from a given derivation diagram by removing all of its spatial relation edges and leaving only part-of predicate edges and their connecting nodes. For example, Fig. 3.4 shows a skeleton of the diagram of Fig. 3.2. Notice that the skeleton is a tree structure whose root or subroot corresponds to a web α and its descendents to another web β and thus a rewriting (or production) rule $\alpha \rightarrow \beta$ where $\alpha \in V_N$ and $\beta \in V_N \cup V_T$ is obtained. The grammar thus obtained is context-free because of the nature of tree structure.

Fig. 3.4 The skeleton of fig. 3.2.

A section is a subdiagram that corresponds to a sentential form of the language generated by the web grammar. We are only interested in the terminal section that shows the external code symbols in order. For example, Fig. 3.5 shows the terminal section of Fig. 3.2. Suppose $n_0, \dots, m_i$ is a path from root node n_0 to a frontier node (or leave node) m_i of the skeleton. Let $m_1, m_2, \dots, m_k$ be all the frontier nodes. Then a set C of nodes of the derivation diagram is a cross-cut set if $C \cap [n_0, \dots, m_i]$ is a singleton for all $1 \leq i \leq k$. A cross-cut set C , together with all the edges of the derivation diagram between nodes of C , is called a section.

Fig. 3.5 The terminal section of fig. 3.2.

To obtain the web grammar from a given derivation diagram, a skeleton is first obtained to derive the production rules with $\alpha \rightarrow \beta$, $\alpha \in V_N$ and then a section is obtained to describe the embedding relationship of a production rule. Notice that the number of production rules with $\alpha \rightarrow \beta$, $\alpha \in V_N$ is equal to the number of internal nodes in the skeleton. Also notice that the nodes in the terminal section is a string of primitive icons in V_T .

4. ICON-BASED CHINESE INPUT SIMULATOR

An icon-based Chinese input method employs a set of primitive icons to compose each Chinese character. The construction process to generate a string of ordered primitive icon codes is guided by a set of predefined decomposition rules. For example, suppose the general rule to decompose a character is from left-to-right then from top-to-down then from outside-to-inside, then the character 腦 may be decomposed into a string of primitive icons 月, 𠂇, 𠂇, 𠂇, 口, 𠂇, this, in turns, identifies the character.

However, in practice, the number of primitive icons is limited to at most the number of terminal keys, i.e., the selected primitive icons are a reduced version of the original ones, for example by using 田 to represent 囧 and 女 to represent

𠂇 (see Fig. 4.1). Because of this, a string of primitive icon codes may correspond to many different characters which results the so called collision problem. An useful Chinese input method shall keep the collision rate low. In addition , a few requirements such as the average number of key-in operations shall be less, the decomposition speed shall be faster, the decomposition rules shall be easier to use, etc. need to be considered. In particular, the decomposition speed shall be fast enough to meet ordinary typing speed standard.

Fig. 4.1 The reduced decomposition graph of the character 腦 .

As far as the decomposition act is concerned, the decomposition speed depends on the number of derivation steps (or the number of production rules applied) in character

decoding as well as the frequency of the decomposition rules being used. The fewer derivation steps and the easier to apply the decomposition rules the faster the decomposition speed will be obtained. As stated in the previous section, the number of derivation steps can be obtained from counting the number of production rules with $\alpha \rightarrow \beta$, $\alpha \in V_N$ and $\beta \in V_N \cup V_T$ being applied, and the number of keys depressed is the number of the terminal nodes. The decomposition speed and ease of use of the decomposition rules may be directly measured from counting the nodes and the appearance rate of the decomposition rules used. In other words, we may use the decomposition graphs as media not only to simulate a Chinese input method but also to derive production rules from them (from parsing of the decomposition graphs) in order to obtain its performance evaluation.

A Chinese input simulator is a machine that accepts user-defined decomposition graphs and corresponding decomposition rules as the input and produces the evaluation result as the output. Thus a simulator is a six tuple

$$CIS = (Q, S, \Delta, \delta, q_0, F)$$

where:

1. Q is a finite set of states.
2. $S = (P, E)$ is a finite set of decomposition graphs, the node set P is a finite set of external code symbols and the edge set E consists of two types of edge, a dash-line arrow specifies a part-of predicate and a solid-line arrow specifies a spatial relation.
3. Δ is a finite set of output values.
4. δ is a mapping from $Q \times (S \cup \{\lambda\})$ to a finite subset of $Q \times \Delta^*$, and λ denotes a null element.
5. $q_0 \in Q$ is the initial state.
6. $F \subseteq Q$ is the set of final states.

It is worthwhile to note that some values may be generated and accumulated at each intermediate state so that the evaluation result is obtained simultaneously at the state. In the following section, we shall show a visual programming environment under which an user can define an application software for developing Chinese input methods.

5. A VISUAL PROGRAMMING ENVIRONMENT

To support the development of an user-defined Chinese input simulator, a visual programming environment provides some facilities and tools for the user to select during his construction of the application program. An user may first select a table (the decomposition rule table) for the description of his/her own set of decomposition rules, then select another table (the primitive icon table) to define a set of primitive icons as terminal nodes for the decomposition graphs, then select another table (the spatial relation table) to define spatial relations for arcs construction. After that, the user can create a working area (decomposition graph window) to construct each decomposition graph of the corresponding Chinese character according to decomposition rules and a reporting form window to display the intermediate evaluation results. The system then converts each decomposition graph into a derivation diagram and records the information of skeleton and terminal section subgraphs into some system generated tables for later use in the evaluation process. A build-in function set defined in the evaluation procedure table is called by the system to generate values to measure the efficiency of the input method just defined. The evaluation result then shows in the reporting form window with format defined in a reporting form schema for recording and displaying both the intermediate evaluation result during execution and the final evaluation result after execution.

Fig. 5.1 shows the system organization of a visual programming synthesizer. There are four definition languages for users to define the decomposition rules, the primitive icons, the spatial relations, and the evaluation procedures. The results are some tables, the decomposition rule table (DRT), the primitive icon table (PIT), the spatial relation table (SRT), and the evaluation procedure table (EPT). Notice that they are defined by a form-based visual language, though they are going to be used in the construction of icons and icon related information. After these tables are defined, another form-based language can be used to define the reporting form schemata for recording and displaying the intermediate and final evaluation

results. A schema defines the specification of the reporting form which consists of a form heading (defining the form structure) with form fields, the form field sources (where are they coming from), field data types, and specification items. Each specification item (in a set of build-in function items) selects some form fields for its specification. Finally, the program structure is described into an internal representation of the

application program via a V-Form interface language. The internal structure of the application program defined in this way is an extended version of our previous designed visual programming synthesizer, the EVIPS [7,8], because now we have to include two more V-Form types DG (stands for decomposition graph) and REPORT. DG is a type of window for the user to define decomposition graphs and to select the corresponding decomposition rules, while REPORT is a type of window where the intermediate and final reports show.

Fig. 5.1 The system organization of a visual programming synthesizer.

Fig. 5.2 shows a summary of the command menu used in the visual programming environment. On the top is a main menu for the selection of commands such as defining tables, specifying reporting form schemata, constructing application programs, executing the constructed application program, setting up or exiting from the system , or asking for help. The table command menu is for the construction of four tables, the decomposition rule table, the primitive icon table, the spatial relation table, and the evaluation procedure table with some editing operations, build-in function calls, and file manipulations. The reporting form schema command menu is for the construction of reporting form heading and its body. The program construction command menu is very much the same as that of EVIPS, except now the tables are PIT, DRT, EPT and SRT.

Fig. 5.2 The summary of the command menu.

6. PROGRAM CONSTRUCTION

From the user's point of view, the construction of a Chinese input simulator, under the previously described visual programming environment, must follow certain steps in order. Fig. 6.1 shows such an application program construction flow diagram.

Fig. 6.1 Application program construction flow.

As mentioned above, four tables : DRT, PIT, SRT, and EPT (Fig. 6.2 shows their appearance) and a reporting form schema (Fig. 6.3 shows its appearance) are first defined. To

define a reporting form schema, the user can load in a reporting form heading if it has been defined previously and select the proper form fields for each specification item.

Primitive Icon Table		Spatial Relation Table		Decomposition Rule Table		
Icon	Internal Code	Icon	Description	ID	Type	Description
月	02	l	left-of	dr1	D	左右分
女	11	r	right-of	dr2	D	上下分
田	20	a	above-of	er2	E	全部取五碼
口	24			er3	E	字首取兩碼
土	30					

Evaluation Procedure Table		
Name	Parameters	Definition
f	x	$\text{SORT} [0.8 + 0.2 * (\text{NO-POG}(\text{IN-RULE}(x)) / \text{NO-PR})]$
ds	1,x	$1, \forall x \in \text{Terminal Nodes}$
ds	h,x	$\text{SORT} [\text{SUM} (\text{ds}(h-1, y)) * f) / \text{SUM} (\text{HEIGHT}(y))], \forall y \in \text{CHILD}(x)$

Fig. 6.2 An example of part of four tables.

Form Heading	Form Body	File	Help	Exit
IRPT : Chinese Input_evaluation				
(Character Result)				
	character symbol name	occur rate (%)	decomp speed	(Rule)
			name	total occur. times
			name	total occur. times
SOURCE	sybm(cic)	rate(cic)	ds(cic)	sybm(lie) occ(cic) ru(lie) irate(nic)
DATA TYPE				
KEY	✓			
COUNT	✓			
AVG		✓	✓	✓
MIN		✓	✓	✓
MAX		✓	✓	✓

Fig. 6.3 The definition of the reporting form schema.

Fig. 6.4 shows an insertion process of a V-Form with type: REPORT and name: result to a V-Form page, the VFORM:Decomp_Result (a program segment), the result is showing in Fig. 6.5.

Fig. 6.6 shows the execution of the program segment VFORM:Decomp_Result. As can be seen, the user is using two tables, the primitive icon table and the spatial relation table, to construct the decomposition graph of the Chinese character 腦 (brain) and describing the decomposition rules being used associated with the character in a table (Corresponding Rule Table). After the construction of the decomposition graph of the character, the evaluation result (accumulative updated values) appears in the right reporting form. To continue the construction

Fig. 6.4 The insertion process of a V-Form with type REPORT.

Fig. 6.5 The result after V-Form REPORT:Schema1:result is inserted.

of the next decomposition graph, the V-Form Next Character is selected. When all the decomposition graphs have been constructed, the V-Form Final Result is selected to get the overall evaluation result of the Chinese input method just designed (the result shows in Fig. 6.7).

7. CONCLUSION

In this paper, we show that an icon-based Chinese input method is an iconic language. An icon-based Chinese input method is defined in terms of a set of decomposition graphs. A decomposition graph can be transformed into a derivation diagram. A web is a derivation diagram for a context-free web grammar. The number of production rules is the number of decomposition steps in parsing the visual routines to decode a Chinese character. Based upon these observations, a Chinese

Fig. 6.6 The execution of VFORM:Decomp_Result.

Chinese Input.exe P I T S R T

The following reporting form is the summary of the evaluation results

SAVE		PRINT		HELP	
Total No. of characters	Average decomposition speed	Total no. of primitive icons used	Average no. of keys depressed	Average collision rate	Average no. of rule used in one character
13900	0.825	105	3.2	0.27	4.5

Next Input Method END

Fig. 6.7 The overall evaluation result of the Chinese input method designed.

input simulator can be described as a machine that accepts the specification languages of the decomposition graphs and the decomposition rules. We then show that a visual programming environment can be established for users to define their own Chinese input simulators (application programs) and run their own defined Chinese input methods on it.

REFERENCES

- [1] M. Beretta, P. Mussio, and M. Protti, "Icons : Interpretation and Use", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.149-158.
- [2] Alfs T. Bertziss, "Specification of Visual Representations of Petri Nets", Third IEEE Computer Society Workshop on Visual Language, Aug. 1987, pp.225-233.
- [3] J. M. Brayer and K. S. Fu, "Some multidimensional grammar inference methods", in Pattern Recognition and Artificial Intelligence, (C. H. Chen, ed.), Academic Press, New York, 1976.
- [4] S. K. Chang, "Icon Semantics - A Formal Approach to Icon System Design", Int'l Jour. of Pattern Recognition and Artificial Intelligence, World Scientific Press, 1987.
- [5] S. K. Chang, E. Jungert, S. Levialdi, G. Tortora, and T. Ichikawa, "An Image Processing Language with Icon-Assisted Navigation", IEEE Transactions on Software Engineering, August 1985, pp.811-819.
- [6] S. K. Chang, Q. Y. Shi, and C. W. Yan, "Iconic Indexing By 2D strings", second IEEE Computer Society Workshop on Visual Language, June 1986, pp.12-21.
- [7] K. Y. Cheng, C. C. Hsu, I. P. Lin, M. C. Lu, and M. S. Hwu, "VIPS: A Visual Programming Synthesizer", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.92-98.
- [8] K. Y. Cheng, M. S. Hwu, and C. C. Hsu, "An Extended Visual Programming Synthesizer for Computer Aided Instruction Applications", Third IEEE Computer Society Workshop on Visual Language, Aug. 1987, pp.147-160.
- [9] W. Finzer and L. Gould, "Programming by Rehearsal", Byte, Vol. 9, No. 6, June 1984, pp.187-210.
- [10] G. F. McCleary, Jr., "An Effective Graphic "Vocabulary"", IEEE Computer Graphics and Applications, March / April 1983, pp.46-53.
- [11] F. S. Montalvo, "Diagram Understanding: Associating Symbolic Descriptions with Images", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.4-11.
- [12] E. P. Glinert and S. L. Tanimoto, "Pict : an Interactive Graphical Programming Environment", IEEE computer Magazine, November 1984, pp.7-25.
- [13] C. F. Herot, "Spatial Management of Data", ACM Trans. on Database Systems, Vol. 5, No. 4, 1980, pp.493-514.
- [14] K. Hoehne, "The ISQL Language - A unified Tool for Managing Images and Non-Images Data Management System", in Visual Languages, edited by S. K. Chang et. al., Plenum Pub. Co. 1986.
- [15] T. Ichikawa, "HI-VISUAL: A Language Supporting Visual Interaction in Programming", in Visual Languages, edited by S. K. Chang et. al., Plenum Pub. Co. 1986.
- [16] R. J. K. Jacob, "A State Transition Diagram Language of Visual programming", IEEE Computer, Vol. 18, No. 8, Aug. 1985, pp.51-59.
- [17] M. A. Musen, L. M. Fagan, and E. H. Shortliffe, "Graphical Specification of Procedural Knowledge for an Expert System", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.167-178.
- [18] J. L. Pfaltz and A. Rosenfeld, "Web grammars", Proc. First Int. Joint Conf. Artif. Intell., May 1969, Washington, D.C., pp.609-619.
- [19] N. C. Shu, et al., "Specification of Forms Processing and Business Procedures for Office Automation", IEEE Transactions on Software Engineering, Vol. SE-8, No. 5, Sep. 1982, pp.499-512.
- [20] N. C. Shu, "A Forms-oriented and Visual-directed application development system for non-programmers", First IEEE Computer Society Workshop on Visual Language, Dec.1984, pp.162-170.
- [21] N. C. Shu, "FORMAL : A Form-Oriented, Visual-Directed Application Development System", IEEE Computer, Vol. 18, No. 8, 1985, pp.38-49.
- [22] Tadashi Ae and Reiji Aibara, "A Rapid Prototyping of Real-Time Software Using Petri Nets", third IEEE Computer Society Workshop on Visual Language, Aug. 1987, pp.234-241.
- [23] I. Yoshimoto, N. Monden, M. Hirakawa, M. Tanaka, and T. Ichikawa, "Interactive Iconic Programming Facility in HI-VISUAL", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.34-41.
- [24] M. M. Zloof, "Query-by-Example: A Data Base Language", IBM System Journal, Vol. 16, No. 4, 1977, pp.324-343.
- [25] M. M. Zloof, "QBE/OBE: A Language for Office and Business Automation," IEEE Computer, Vol. 14, No. 5, 1981, pp.13-22.