Design and Implementation of a Graphical Programming Tool for Children

Chen Xiajian
Graduate University
Chinese Academy of Sciences
Beijing, China
chenxiajian1985@gmail.com

Wang Danli, Wang Hongan
Institute of Software
Chinese Academy of Sciences
Beijing, China
danliwang2009@gmail.com

***Abstract*— Since the computer is more widely used by children, the existing software for children's amusement and study are not able to satisfy their needs any longer. To solve this problem, a kind of graphical programming language for children is proposed in the paper. We analyze the design principles of graphical programming language for children, and implement ten types of common graphical blocks, integrating the basic knowledge of object oriented programming. We also provide the function to bridge the gap between graphical and text programming. Additionally, we designed and developed a graphical programming tool – KidPipe based on Pen Interaction. Finally, a user study is carried out to verify reliability and usability of KidPipe.**

Keywords-Children Programming; Graphical Programming tool; Object Oriented Programming

I. Introduction

With the development and population of Information Technology, more and more children have opportunity to get access to computers, gaining knowledge, knowing society and perceiving world through computer. Computer is changing the way of children's life and study. Compared with adults, children have their own characteristics. The proper use of computer can help and guide children enhance their cognitive and creative ability [1].

With children's using computers deeply, current software are unable to satisfy children's need. Although children are not professional programmers, they are also eager to create their own programs. When children participate in problem-solving activities, they will have a high learning motivation [1]. Therefore, we believe that building an appropriate programming environment has great potential to promote children to join and enjoy creative activities. One of difficulties in the field of children's programming is how to use proper programming language and interactive modes to express children's operation intention [2].

It's difficult for young children to understand the hard codes in programming languages. Many researchers carried out a series of efforts on the improvement. Some researchers tried to simplify parameter settings in the programming process [3], while some researchers used objects with sensors as an interactive mode to reduce children's cognitive burden for the programming language [4]. In addition, the graphical programming has also simplified the traditional programming pattern, and enriched the material library which provides more diverse and creative materials. However, the simplification of the function parameters reduces the burden on the operation of children's programming to some extent, but it still doesn't completely get rid of text programming. In tangible programming children's interactive objects are entities familiar to children. Tangible programming has effectively improved children's interest. However, the semantics that the tangible entities can express is limited, as well as that constructing programs need a certain space. In contrast, graphical expression is one of the most direct and natural ways by which children can perceive the world, also easy to understand for children. On the other hand, graphical programming can provide richer materials. In fact most of the graphical programming software does not fully make the blocks understood easily, and some blocks also contain some text codes, which bring younger children more cognitive load.

II. Related Work

At present, most of programming systems provided an editable environment for children to express and modify their creative ideas. Logo [5] is a very successful and popular programming language for children, which is based on the Lisp text language, and use easy-to-learn syntax for children. StarLogo [6] extends the metaphor of Logo, and it can process in parallel. LogoBlocks [7] system allows children to build a program by Logo bricks. EToys [8] made a further expansion, allowing children program to produce music, language and mathematics. The inspiration of the design of Alice came from EToys [9]. Similarly, Alice makes novice understand programming easily through built-in editor. Alice provides users with a 3D scene. The program structure in Alice retains some specific text keywords in traditional text programming language. It is difficult for children to understand; therefore Alice is more suitable for users who have higher cognitive level. Scratch [10] is designed for 8 to 16-year-old users, which defines commands and action blocks at high level for games and animation works. Scratch is more inclined to simplify the creation process [11].

Besides, some programming tools for children are not designed through the obvious block. My Make Believe Castle [12] focuses on how to make users create animation simply. Users can put roles on the screen, and then draw lines tracking roles' movement. The system was designed for those who are 4 to 7 years of age. Stagecast [13], Hands [14] and Rapunsel [15] mainly support users to create their own games. Users can control roles in games to finish a program.

Above all, in order to reduce the complexity of learning programming for children, some key techniques are used, including simplifying the syntax of programming languages [16] and designing programming language more consistent with children's cognitive level and matches the way of describing roles' behavior for children [13]. Besides, they help users build program by assembling graphical blocks [7] [8] [9] [16] and programming through presenting actions in 3D animation scene [17].

On the basis of the background that children learn programming in entertainment, we built a programming environment for children based on Pen Interaction, in which children can expand their imagination and creative power, and enhance their logical thinking ability.

III. DESIGN OF GRAPHICAL PROGRAMMING FOR CHILDREN

3.1 Analysis of graphical programming language

According to the needs of children, we designed the following categories of premier languages based on object-oriented programming: movement, situation, appearance, sound, event, arithmetic, logic and structures. The action category is used to set the basic movement of roles. The situation category is used to display the current character's attributes. The event control category is designed to control the events in the program. Events include start event, stop event, click event and keyboard event. Logic class is used to determine the conditions of the program logic. Different logic may be combined to make the conditions complex. The structure category is a very important type of primitive language and a major obstacle to children's programming. There are two types of structures in the graphical programming language, one is conditional branch (if-else), and the other is loop branch (do-while, while-do).

Function is a very important part in programming language. On one hand, function represents the concept of module in programming. On the other hand, the function represents the integration of the basic block in a higher level. In graphical programming for children, function is provided for children to understand the meaning of function and gain problem-solving abilities.

3.2 Graphical representation of programming languages

The graphical blocks enhance the use experience for children because they can effectively improve children's learning interest. In this paper, Children's programming process is to arrange and combine graphical blocks in a certain sequence by which syntax errors will be reduced compared with text programming. Children will focus on how to solve logical problems rather than syntax errors.

3.2.1 Design principles for graphical block

We design the blocks according to the pipeline which children are familiar with when childhood. The basic principles of the design of graphical blocks as follows: the graphical blocks are requested simple and clear firstly, and can clearly show the meaning of code secondly; in grammar we request the splicing block of code be able to directly express the syntax, and blocks which have grammar errors cannot be connected. Blocks with the four design principles showed in *Fig.* 1.

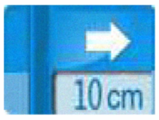


Figure 1. Design principles of block

We can divide the graphical blocks into four categories in the appearance: 1) Outlet mode, as shown in *Fig.* 2 (1). The block of the single-exit mode is used to express the event that inspires the beginning block; 2) Entry mode, as shown in *Fig.* 2 (2). The block of the single-entry mode is used to indicate the end of an event; 3) Dual-outlet mode, as shown in *Fig.* 2(3). We can connect this type of blocks together to achieve a lot of different behaviors; 4) Single entry & multiple outlet mode, as shown in *Fig.* 2(4). This block is used to represent branching structure, like the IF conditional statement or loop.

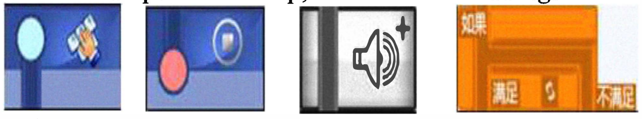
We used different colors to distinguish the different graphical blocks shown in *Fig.* 3. Different colors represent different categories, so that children can understand the classification from color

3.2.2 Action Block

A research shows that children can express ideas and communicate with each other by creating animations, games and telling stories. These programs are not complicated for children [18]. In order to satisfy the need of children, we designed four types of action blocks: movement, situation, appearance, and sound shown in *Fig.* 4.

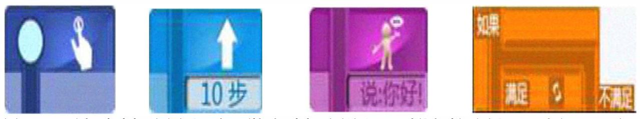
3.2.3 Logical Block

Logic plays a very important role in programming and an important obstacle during learning programming. Logics used in KidPipe include loop, condition shown in *Fig.* 5.



（1）Outlet (2) Entry (3) Dual-outlet (4) Single entry & multiple outlet

Figure 2. Classes of blocks



(1)Event(dark blue)(2)Action(light blue)(3)Sound(Pink) (4)Branch(orange)

Figure 3. Different colors of blocks

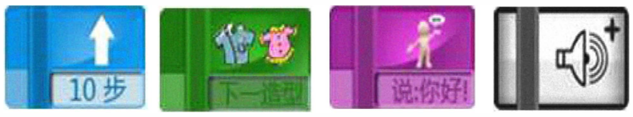
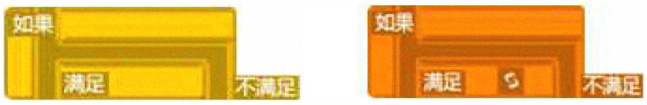


Figure 4. Action block



（1）Condition (IF..Else) （2）loop (While)

Figure 5. Logical block

3.2.4 Event Trigger Block

Event trigger mechanism aims at allowing users to add some interaction in their own works. It increases the degree of users' participation when the program is running. Several basic event blocks have been designed according to the types of event control, shown in *Fig.* 6.

3.2.5 Function Block

In order to manage functions easily, the function block must be in accordance with the basic principles of blocks. While defining a function, children must select the functions category firstly, and then add comment into the function block.

IV. Design and Implementation of KidPipe

4.1 Framework of KidPipe

KidPipe is designed to provide children with an interactive programming environment, helping children learn the basic programming idea in the process of creating works. The overall framework of the system is shown in *Fig.* 7.

The main role of the translation engine is to convert the graphical language into the language computers can understand. Translation engine receives graphical input. Firstly, analyze syntax and understand semantics. Secondly, graphical input is converted into intermediate C++ codes by text language parser and then converted into machine codes. It is the codes computer can comprehend that are used as output of the translation engine. The program compiler is responsible for translating the intermediate code into objective code by code generator.

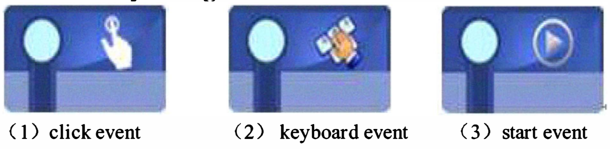


Figure 6. Event Trigger Block

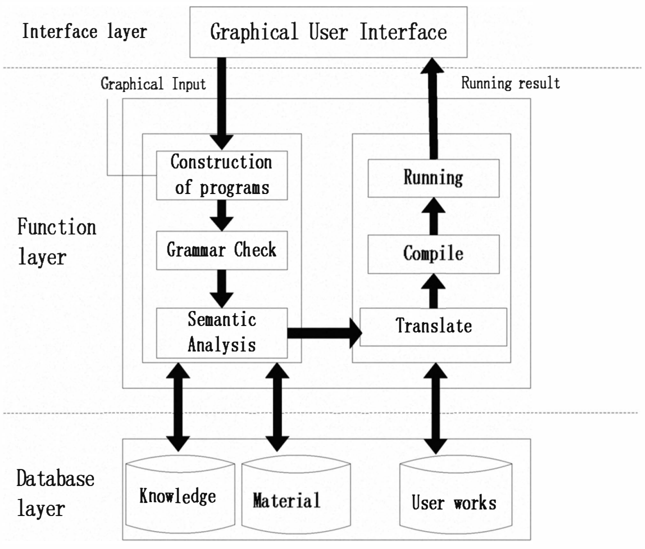


Figure 7. Framework of KidPipe

4.2 Interaction Patten in KidPipe

Natural interaction and humanization interface are trend of human-computer interaction [18]. Compared with traditional interaction pattern such as mouse and keyboard, Pen interaction mode neither need to remember lots of letters and positions on keyboard, nor need train before using it, and shows harmonious and natural advantage for children. The graphical programming system KidPipe is based on Pen interaction. Drag and drop is used as the primary programming way to arrange graphical blocks in KidPipe. Input parameters through writing parameter in the panel of graphical blocks. Pen interaction is more appropriate in these usage scenarios, and more coincident to children’s habit.

4.3 Implementation of KidPipe

Figure 8 shows the interface of KidPipe, which is divided into scenario area, script area, optional operation block area, role object management area, object attribute area and resource area. Executive results will be shown in scenario. Code snippets display in the script area. The role object management area is used to manage objects needed in programming. Required resources in creating works are provided in the resource area, and of course, users can import resources externally

4.4 Usage example of KidPipe System

Figure 9 shows a simple animation program example constructed under pipe connection rules. The meaning of the above graphical program is that when clicking the girl by pen, the execution will start. The girl move right for ten steps and then hit the condition judgment, which detects whether the girl encounters other roles. If the condition is satisfied, she is going to switch another image, and say “Hello”, at the same time playing background music. Otherwise the girl will continue to move right for ten steps. The corresponding C++ codes are shown at *Fig.* 10.

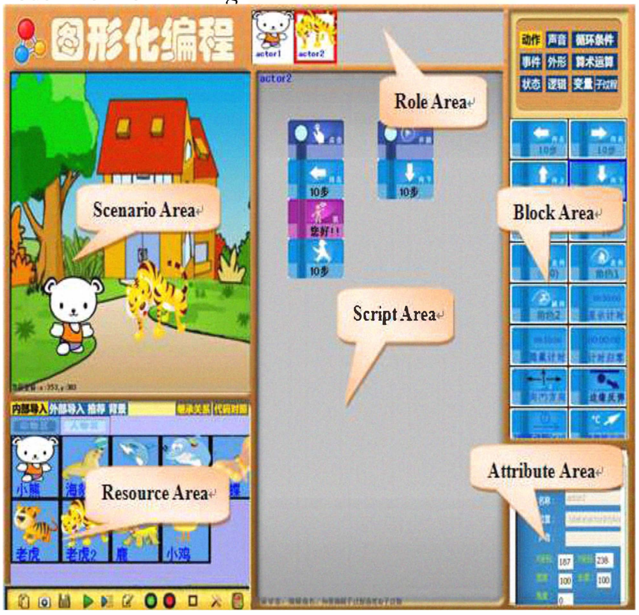
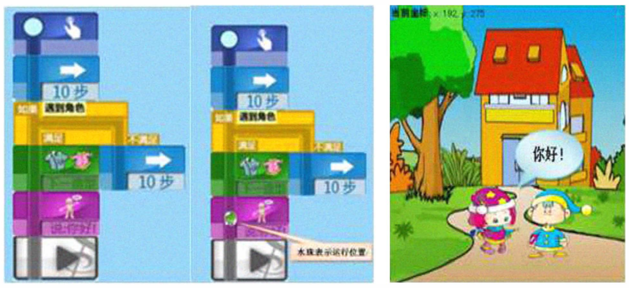


Figure 8. Interface Design of KidPipe System



(1) Graphical Program (2) Running Effects (3) Executive Results

Figure 9. A simple animation program example

```
void Girl::OnClick()
{
    this->moveRight(10);
    if(this->enCounterOther())
    {
        this->changeAppearance();
        this->say("你好!");
    }
    else
    {
        this->moveRight(10);
    }
}
```

Figure 10. C++ codes corresponding to graphical blocks

V. USER STUDY

In the user study, firstly, we wanted to validate the usability of KidPipe. By a contrast experiment with EToys [8], we did quantitative analysis on six aspects including learnability, easy use, reliability, naturalness, efficiency, satisfaction, in order to find the advantages and disadvantages of KidPipe. Secondly, by observing the participants' operations and interviewing them, we hoped to dig deeper significance and using value of the graphical programming, in order to obtain a panoramic view of the application of graphical programming, and to guide the further improvement of our programming tool.

KidPipe focus on children who are 8-16 years old and know some Chinese. We have invited 16 children to participate in our user evaluation. 16 children are divided into 2 groups and each group has 8 children. Group 1 use KidPipe to finish the tasks and Group 2 use EToys. The contrast experiment is not including the object-oriented module, since EToys is not designed for teaching object-oriented programming.

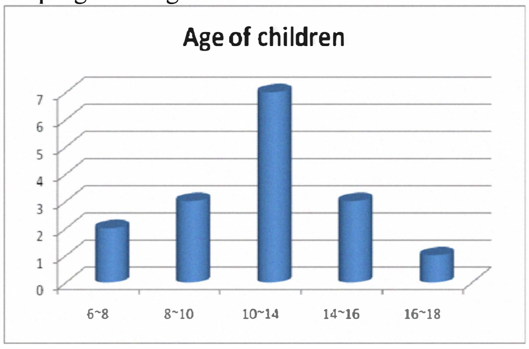


Figure 11. Analysis of participant

5.1 *Evaluation process*

We used user testing (contrast experiment) and interview to evaluate our graphical programming tool KidPipe. In the user testing, we observed users' operations and recorded it with a camera. After they finished the tasks we asked them to finish the questionnaire. After the user testing, we interviewed them to collect their opinions and suggestions which were helpful to inspire users' divergent thinking, guiding them to make deep and overall evaluation.

Based on plenty of user research and analysis, we designed the evaluation plan, including training slides, tasks and questionnaire and so on. Training slides were used to explain the usage of KidPipe and EToys. The tasks consist of four parts: basic block programming; function modules; object-oriented module; free programming. Task 1, Task 2 and Task 3 were used to get the quantitative data in order to evaluate the usability of KidPipe. Task 4 allowed users to program freely, in order to get further observation on users' operational habit. The questionnaire was used collect users basic information and grades KidPipe. There were 18 questions in it, covering six aspects: learnability, easy use, reliability, naturalness, efficiency, satisfaction.

5.2 *Data Analysis*

After the user testing, we analyzed the data we got from the evaluation by SPSS.

5.2.1 *Analysis of the objective data*

1. Duration of the former three tasks

We counted the time participants used to finish former three tasks. The result is shown Table 1.

On the whole, the two groups can complete the assigned task in a very short time (3 minutes), indicating that KidPipe is ease of use. For task 1, the times used are the same to some extent in both groups. We found that, group 1 spent more time in drag operation in which EToys is better than KidPipe. Also we found that the block of KidPipe can be recognized easier than EToys. Because EToys uses the text blocks and brings a lot of cognitive burden to children. KidPipe adopt the graphical block so that children can recognize the block easily.

In the Task 2, we found KidPipe better than EToys in sub process module. We can also see that children can quickly understand the knowledge of function. The time used in Task 3 is a little longer than the former two tasks since children are not familiar with the concept of object-oriented concept.

2. Incorrect operations

We counted the incorrect operations during users' evaluation. The result is shown in Table 2.

TABLE I. DURATION OF THE FORMER THREE TASKS (UNIT: SECOND)

	KidPipe		**EToys**	
	Mean	***s.d.***	***Mean***	***s.d.***
Task 1	117.32	50.19	115.45	50.18
Task 2	110.75	45.72	121.25	52.25
Task 3	121.69	61.24		

TABLE II. INCORRECT OPERATIONS

	KidPipe	EToys
Task 1	2	2
Task 2	3	4
Task 3	5	

By observation, we found that none of the participants did incorrect operations when they arranged roles. All the incorrect operations appeared in the step of programming. There were 10 times of incorrect operations in the former three tasks totally for group 1, which were made by three of the participants. Most of the participants used the two softwares smoothly. And the incorrect operations didn't cause any system corruption, and could be withdrawn easily. The number of incorrect operation in Task 3 are a little higher than the former two tasks, which dues to children are not familiar with the concept of object-oriented programming.

5.2.2 Analysis of the subjective data

We collected users' subjective opinions by questionnaire and interview. There were 18 questions in the questionnaire covering six aspects. The result is shown in Table 3.According to the data we collected, in all the six aspects both of KidPipe surpassed the average level of 3.00. In the aspect of learnability, KidPipe has been recognized by participates. Participants mentioned that in the interview, KidPipe use pen-based operation, so that the users can finish the programming process though drag and drop operation. The results validate our design. From the overall perspective, KidPipe can effectively help children learn programming.

VI. CONCLUSION

In this paper, we reason why text programming tools are not suitable for children, and summarize current research status in graphical programming system. Besides, we propose the graphical programming thinking way for children according to children's cognitive ability and learning theory. The paper focuses on design and implementation of the graphical programming tool for children. Our graphical programming tool – KidPipe helps children to learn programming when creating works. And user study shows that the simple interactions do not bring cognitive burden to children.

TABLE III. GRADE OF KIDPIPE

	Mean	*s.d.*
Learnability	4.55	0.50
Easy-use	4.63	0.47
Reliability	4.22	0.93
Naturalness	4.50	0.40
Efficiency	4.43	0.62
Satisfaction	4.61	0.42
Overall	4.39	0.42

In the future, we'll improve the current graphical programming tool in the following aspects. First of all, provide more effective and clear mistake prompts for children in the course of programming and debugging so that children can recognize mistakes easily and modify the mistakes. Secondly, we hope to provide children with individual service, for instance, supporting customized pictures, sound effect and so on. In addition, we will extend interaction patterns of programming tools further, and do more research work on programming issue with the real object interaction.

REFERENCES

[1] Kafai, Y.B. Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies. Games and Culture, Vol. 1(1), 2006, 36-40.

[2] Pane, J.F, B.A.Myers, and L.B.Miller, "Using HCI Techniques to Design a More Usable Programming System", Proceedings of IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC 2002), Arlington, VA., 2002, pp. 198-206.

[3] Repenning, A. and Ioannidou, A. Agent-BasedEnd-User Development. Comm. ACM, 47(9):43-6. 2004

[4] Zhou ZY, Cheok AD, Tedjokusumo H, Omer GS (2008) wIzQubesTM–A novel tangible interface for interactive storytelling in mixed reality. Int J Virtual Real 7(4):9–15.

[5] Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books New York, NY, 1980

[6] M.Resnick, Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds. Boston: The MIT Press, 1994.

[7] Begel, A. LogoBlocks: A Graphical Programming Language for Interacting with the World. EECS, MIT Boston, MA, 1996

[8] Kay, A. Etoys and Simstories in Squeak.http://www.squeakland.org/author/etoys.html

[9] Maloney, Burd, Kafai, Rusk, Silverman, and Resnick, M.Scratch: A Sneak Preview. In Proc of Creating, Connecting, and Collaborating through Computing (2004). 104-109.

[10] Resnick, Maloney, Monroy-Hernandez, Rusk, Eastmond, Brennan: Scratch: Programming for Everyone. Communications of the ACM. (2009).

[11] Resnick, M., Maloney, J., Hernandez, A.M., Rusk,N., Eastmond, E., Brennan, K., Millner, A.,Roenbaum, E., Silver, J., Silverman, B., & Kafai, Y."Scratch: Programming for everyone," inCommunications of the ACM (in press).

[12] Logo Computer Systems, Inc., My Make Believe Castle, 1995

[13] Smith, D., Cypher, A., and Tesler, L. Programming by example: novice programming comes of age.Communications of the ACM 43, 3 (2000), 75-81.

[14] Pane, J. Myers, B.A., and Miller, L.B. Using HCI Techniques to Design a More Usable Programming System. In Proc. HCC 2002, IEEE (2002), 198-206

[15] Flanagan, M., Howe, D. and Nissenbaum, H. Values at play: design tradeoffs in socially-oriented game design.In Proc. CHI 2005. ACM Press (2005), 751-760.

[16] Bruckman, A. MOOSE Crossing: Construction,Community, and Learning in a Networked Virtual World for Kids. MIT Media Lab. Boston, MA., 1997

[17] Fernaeus, Y., Kindborg, M., and Scholz, R. Rethinking Children's Programming with Contextual Signs. In Proc. IDC 2006. ACM Press (2006), 121-128.

[18] Harel, I. Children Designers. Ablex Publishing Norwood, N.J., 1999