AN EFFICIENT METHOD FOR GAME DEVELOPMENT USING COMPILER

Jae Seong Jeong and Soon Ghon Kim
Dept. of Information Science, Joongbu University, 101 Daehakro, Chubu, Gumsan, Chungnam, South of Korea

Keywords: Index Function and Index Memory, Paser Table, Entire compiling processes, Compiler possible for Online Update, Executing compiled files, Quest Event, Quest API.

Abstract: As the development of an online game is being more and more extensive, higher manpower become more essential in the development for the game. Especially in programming, it happens that that original scheme that came from the planning department could not be fully developed and expressed in the programmer, depending on the ability of the programmer, coming out with a different result which is less enjoyable than the programmer expected. It is essential to spend much time for checking the error came from planning or programming. Due to solve these kinds of problems, we have developed the complier for only games which uses API for game graphic and API for quest that have been used in development for game. The compiler helps the game planner to find out logical problem directly and manually through manual source coding. Also, through the special game compiler, it would help the game developer to come out with a various kinds of efficient plans. It has the advantages of lowering dependency on the game programmer as well as to lower the cost of production and labor resources.

1 INTRODUCTION

Recently as the scale of game development gets large along with the performance improvements of communication infrastructure and computer, a lot of time and resource are committed in the development of games. Also since a lot of human resources are needed in the development process, a lot of time is required in correcting development errors as well.

The development error can be largely divided into planning errors and programming errors. In case of programming error, programmers can correct them easily; however in case of planning errors, it is inevitable that the time taken in correcting them can be further extended.

A number of methods have been introduced to cut the time and development cost that are the key points in game development. Of these, the most frequently used method is to use a game development tool. Solving various problems occurred during the development process of games or contents requires us not only to simplify various overlapped processes but also to minimize error occurrence in these processes.

However only by this method, it is not possible to reduce the time, costs and human resources. Especially in case of online games, relevant technologies are needed since they have been developed as to apply the complex parts of current society to a game. Due to this reason, we would like to suggest a special game compiler as an alternative

2 THEORETICAL BACKGROUND

2.1 Development Background

While the events such as quest, event and siege warfare are not bothered by the speed, these tasks require much time since they are quite complex from various kinds.

If the planner can test the plan in real-time in a way desired so as to do the coding easily by adding a module after indexing the compiler and API in accordance with these parts, the planning errors can be easily corrected.

Also if we could make a special game operator (binary) code optimized through a compiler while considering the speed, this may satisfy the speed, planning and time requirements.

2.2 Understanding the Special Game Language

2.2.1 Game Language for Quest

As the program to run the quest according to the plan, it is made in a form of text file. This includes various local functions needed for the quest.

2.2.2 Basic Operating Conditions of Quest

When checking all the conditions in a script since the speed of quest script is slow, it is possible to cause a load on the program. Hence, this refers to the basic conditions commonly used for all quests. Basically, this checks the min level, max level and NPC index.

2.2.3 C-language Local Function for Quest

When checking all the conditions in a script since the speed of quest script is slow, it is possible to cause a load on the program. Hence, this refers to the basic conditions commonly used for all quests. Basically, this checks the min level, max level and NPC index.

3 STUDY METHOD

3.1 Applicable Method

The data processing and communication parts needed when executing a quest on the network are processed by the server. The client processes the parts related to the speed of screen display and communication of the data received from the server or occurred by user behaviour.

The design is first prepared on the basis of single user processing without communication and later, it is divided into the server and client that have added the communication part, dividing further into the communication and DB processing parts. And then, simply add to the design the part that can accommodate a partial update desired by user.

When the programmer can set the memory handling part and memory structure or internal structure of compiler and APIs and this structure can be handled automatically by one command, the memory handling part that is the most difficult part of programming can be processed.
The structuring part and engine part that require speed can be developed directly by programmer in a form of API. The planner makes the index functions that can be called by the planner by gathering these APIs without needing the planner to call these commands. And, it can be processed in a way that the memory needed in execution is allocated automatically.

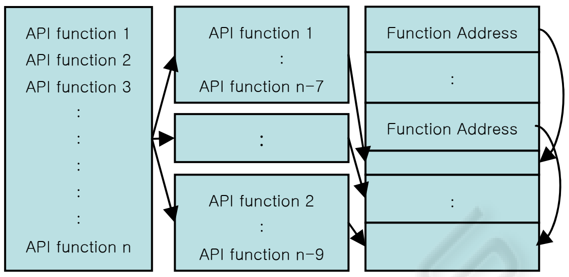

Figure 1: Relationship between Function and Index.

And by gathering the structures and API modules developed by programmer, make the functions used frequently by quests or events and attach an index number to each function prepared in this way. The function attached with index number shall hold the start address that has defined the index function at the foremost part of memory while securing the information related to calling parameters in the order of index.

Execution of the special game language can be done by the memory if the structure is set and the memory is allocated. However, it is not easy for the planner to manage the memory. In order to solve these problems, the index-type of function should be automatically executed and the memory to save the resulting value should be allocated.

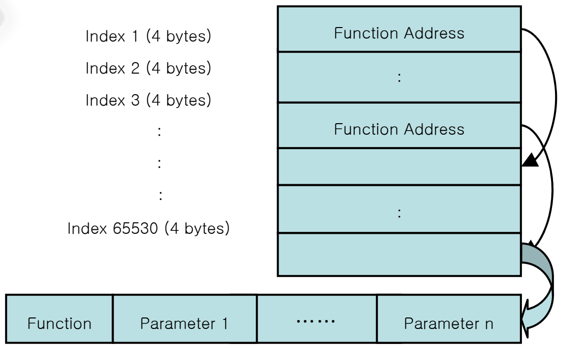

Figure 2: Index Function and Index Memory.

Since the quest or event of generating many processes is increased as much as the capacity of database, the speed shall be considered.

- Limit the maximum memory to process the quest and maximum amount of memory to execute the quest at the same time.
- Save the parameter value to execute quest, value corresponding to the index value given by the quest, and resulting value of quest.

- The program shall be designed in a way that is not impacted when a user connects to the system again while using the parameter value to be executed, index value given to the quest, and result of the quest.

3.2 Quest and Event Handling Process via the Special Game Language

Explaining briefly about the handling process of compiler, this reads in the coded file appropriate for the quest by using the C-language exclusively for game use as shown on the Figure 3. Analyze the read-in files through the scanner a preprocessor and generate the intermediate language and assembly code for game use while referring to the paser table made through PGS by using the analysis result. Then, make the operator (binary) executable file that can execute the code made in this way.

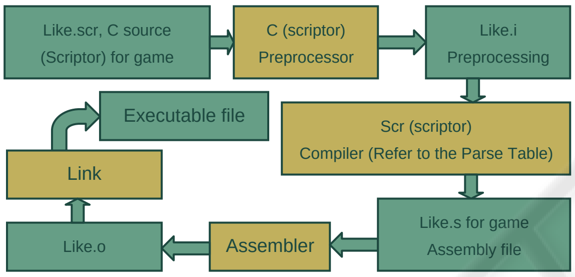
Like.scr, C source (Scriptor) for game
C (scriptor) Preprocessor
Like.i Preprocessing
Executable file
Scr (scriptor) Compiler (Refer to the Parse Table)
Link
Like.o
Assembler
Like.s for game Assembly file

Figure 3: Entire compiling processes.

Figure 3 is designed for use by a common role-playing game and this can be processed in an online game. Figure 4 is similar to Figure 3, enabling to extend the language itself. However, its use is not possible in the general game.

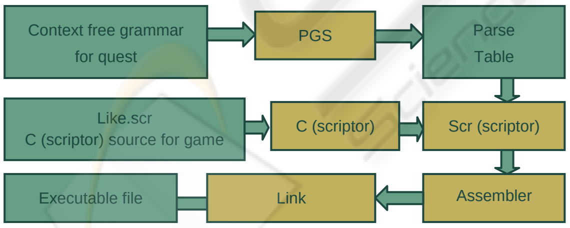

Figure 4: Compiler possible for on-line update.

The difference from Figure 3 is that the added part can be executed simply by updating the context-free grammar without updating the entire program when the language was modified or extended since the parse table generator is built into the client and server. Hence when needing to add various indices to the online game, this is made to cope with the requirements of game user by enabling to add necessary parts easily.

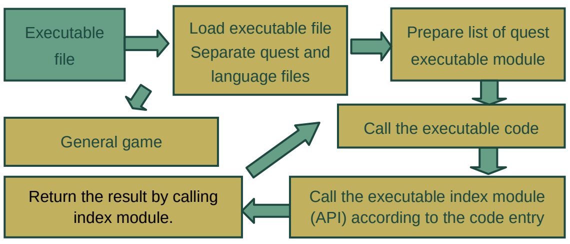
Executable file
Load executable file Separate quest and language files
Prepare list of quest executable module
General game
Call the executable code
Return the result by calling index module.
Call the executable index module (API) according to the code entry

Figure 5: Executing compiled files.

Lastly, Figure 5 is the executable file of C-language for game use, which was compiled on Figure 3 and Figure 4. The code within this executable file is constructed in a way that is divided into two parts. In case of not processing the executable file, simply add the code executing unit that can handle the basic C language within the quest execution module. Although there may be some loss in the perspective of speed, the difference can be ignored since the programmer has handled the API or memory part that requires speed.

3.3 C-language for Quest Use (script)

While all the basic grammar structures follow those of C-language, the memory part is based on the BASIC language. The functions such as pointer, structure, user-defined type and preprocessing are not part of this language. The basic form Functions is made to use it as an internal command rather than one function by indexing the function. When adding new script function, define the index number after defining the function name and parameter. Then, update the file after registering and compiling as an index function.

3.3.1 Basic Structure

```
QUESTNR = 10;    // Setting global variable
Quest_Start(){ .. }  // Start condition of quest.
Quest_End(){…}  //End and compensation of quest.
```

3.3.2 Example of Application

```
QUESTNR = 10;
Quest_Start() {
   if( SeeNPC(17) == -1 ) { return SetFALSE(); }
   if( MainQuestCreate( QUESTNR ) != -1 )
   {   InsertVar( 1, 1, 2);  // Two mops in the zone 1
      SetEventProbability(100); // Item probability
        return SetTRUE(); }
   return SetFALSE(); }
Quest_End(){  //Including the compensation condition
   if( InsertItem(11,1)==0)
```

```
        { return SetNoInvenSpace(); }
        if( InsertRandomItem( 3, 1, 11, 21 ) == 0 )
        { DeleteItem( 11, 1 ); return SetNoInvenSpace(); }
        QuestDestroy( QUESTNR );
        return SetTRUE(); }
```

4 STUDY RESULT

When the compiler is used, we could see that the costs, time and manpower could be saved more than the existing development method. Although similar result was obtained in case of quest or siege warfare event, this has required twice as much manpower in making compiler API functions as compared to quest in case of siege warfare and event.

4.1 In Case of Developing Quest

Table 1: In Case of Developing Quest.

Item		Previous Development	Compiler
Quest PG Developer	Manpower	5	0
	Period	12	0
Compiler Quest API	Manpower	0	1
	Period	0	3
Planner	Manpower	2	2
	Period	12	8
Quest Server Developer	Manpower	1	1
	Period	12	3
Compiler Developer	Manpower	0	2
	Period	0	10
Comment	- In case of the compile, the period of server developer compiler Quest API developer has not increased signify-cantly from an increase in the number of quests; however in case of general development, the number of manpow-er commitment has increaseed com-paratively to the number of quests. (Based on 500 quests)		

4.2 Analysis Result

4.2.1 In the Perspective of Program

- Quite useful in the complex event handling of game.
- Possible to find easily whether it is a program error or planning error.
- Combining conveniently by indexing the API modules.

4.2.2 In the Perspective of Planning

- The planner can code easily. (adjusting the degree of difficulty directly)
- Can discover planning error easily
- Possible to plan diversely according to the intention of planner.

4.2.3 In the Economic Perspective

- Cutting the development costs and time. (reducing 70%)
- Reducing the programming manpower requirement.

5 CONCLUSIONS

In the existing game development method, it is difficult to check whether the error came from planning or programming. However by applying a compiler to game development, the planning part and programming part can be separated and hence, the programming efficiency has increased greatly. Additionally, this study has improved the quality of game by enabling to express the intent of game designer(planner) accurately.

Especially in the online game environment, various changes occur in a short period time. Hence by introducing the compiler engine to the game development environment, the game development company can secure competitiveness by building the game development environment that can be applied easily and quickly to various changes.

REFERENCES

Jae-Seong Jeong, Jae-Sun Mun, Tae-Jin Kim, 2003. “Kahn Online Game Compiler” by Mirinae Ent.

Sang-Woo Noh, 2007. “Cross Platform” “LALR Paser” by E3net.

Young-Sook Sung, 2005. E3net “Smart Phone Converter Solution “ by E3net.

Se-Man Oh, 1998. Introduction to Compiler by Se-Man Oh.

Chun-Hyun Jang, Se-Man Oh, Jae-Woo Woo, 1996. C++ Compiler, Program and Environment Develop-ment, Final Research Report by the National Research Foundation of Korea, KOSEF 93-0100-01-3.

Jin-Ki Park, Se-Man Oh, 1998. “Automatic Configuration of AST,” Paper Collection at Dongguk University

R. J. Parikh, 1966. “Oh Context-Free Lanquages” JACM Vol. 13, No.4

T. Anderson, J. Eve And J.J. Horing, “Efficient LR(1) Parsers” Acta Informationca, Vol.2, No.1 1973.