International Journal of Smart Home
Vol.8, No.3 (2014), pp.223-234
http://dx.doi.org/10.14257/ijsh.2014.8.3.21

Design and Implementation of HTML5 based SVM for Integrating Runtime of Smart Devices and Web Environments

Yunsik Son[1], Seman Oh[2] and Yangsun Lee[3*]

[1,2]Dept. of Computer Engineering, Dongguk University
26 3-Ga Phil-Dong, Jung-Gu, Seoul 100-715, KOREA
[1,2]{sonbug,smoh}@dongguk.edu
[3]Dept. of of Computer Engineering, Seokyeong University
16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, KOREA
[3]Corresponding Author: yslee@skuniv.ac.kr*

Abstract

Current mobile environments, smart device platforms and web based platform are emerged. The smart device platforms are represented by Apple's iOS and Google's Android, and web based platform's core is HTML5. Our previous researches are focused on integrating the developmental environments of smart devices and producing the same runtime environment.

*The **Smart Cross Platform** - our research result - guarantees content compatibility on various smart devices and supports multiple programming languages like C/C++, Java, and Objective-C which is a solution to increase developmental productivity. The **SVM** (Smart Virtual Machine) is the core virtual machine of the smart cross platform. But the Smart Cross Platform and the SVM is not cover the web environment like HTML5.*

In this paper, a technique which allows previous SVM which operates on smart devices to operate on web environment is introduced. In order to provide SVM in a web environment, an interpreter includes SIL code that was designed and implemented in JavaScript. Each SIL code's semantics are reflected by JavaScript and an HTML5 based API and event model was combined to construct the overall SVM.

In the whole design and implementation of the SVM which operates in the web based environment, this paper focuses on the interpreter which is a method to execute the contents. The implemented interpreter loads the SVM contents, creates a list of commands, analyzes on a stack base, executes the commands and reflects the results on the internal memory and status.

Keywords: *Smart Cross Platform, SVM, Interpreter, HTML5, Web based Virtual Machine*

1. Introduction

The smart cross platform is a virtual machine based solution which allows execution of same contents on various smart devices. The research team built a smart cross platform in iOS and Android through previous research and developed a compiler to support contents composed of C/C++, Java, and Objective-C language [1].

In this paper, a technique is introduced in which the previous research was extended to allow the smart cross platform built for smart device OS purposes to run in web environment. Currently, HTML5 which represents web technique is quickly going through standardization and its influence as an open platform is gradually increasing [2, 3]. Thus, the objective of this research is to extend HTML5 which is an open platform to be supported by the smart cross

ISSN: 1975-4094 IJSH

platform's SVM and execute the smart cross platform purpose contents effectively in the HTML5. Especially, in this paper, the execution technique of the SVM (Smart Virtual Machine) which can be executed on a web base is focused from a design and implementation perspective.

The composition of this thesis is as follows. First in chapter 2, the smart cross platform and HTML5 technique in this research are introduced. In chapter 3, the interpreter model is introduced and explains the design and implementation. In chapter 4, the experiment with the implemented interpreter are introduced. Lastly in chapter 5, the conclusion of this research and further research is discussed.

2. Related Researches

2.1. Smart Cross Platform

The smart cross platform's objective is to effectively execute game and multimedia contents as a contents execution platform developed by the research team on various OS and devices. Also it does not depend on specific development language and supports numerous programming languages.

For this, the smart cross platform executes contents as a virtual machine base using the SVM. It supports numerous languages using the compiler set. Also, it is a concept of connecting various languages, compilers and virtual machines. The middle language is defined as SIL and programs are converted into SIL code which is converted into the executable file format SEF by the assembler which is executed in the SVM. Figure 1 shows the smart cross platform's model [4, 5].

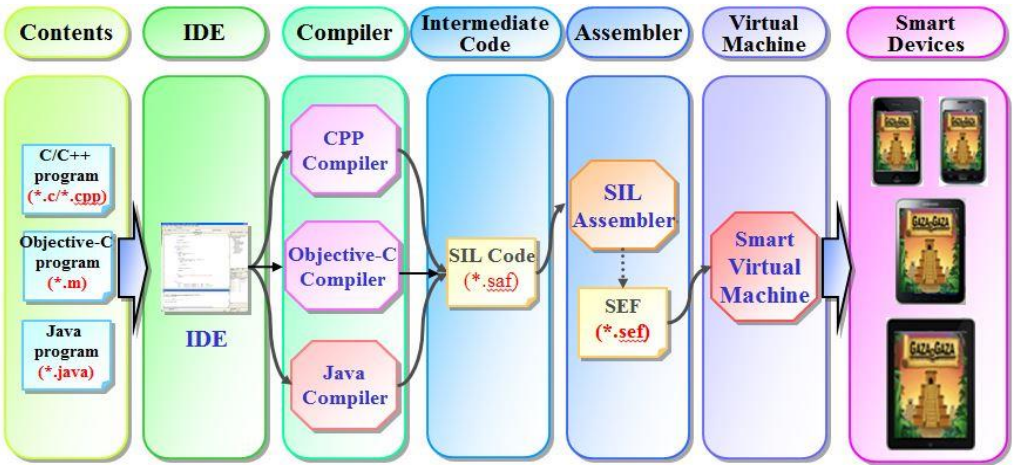


Figure 1. Smart Cross Platform Model

Currently, the smart cross platform can be executed in Windows, iOS and Android. The programming language and compilers supported are C/C++, Java and Objective-C [6, 7]. In this research, techniques to extend the execution environment from previous Windows and smart device purpose OS to web bases are introduced. Through this, previously developed contents can be executed in web browsers supporting HTML5 without any further modification.

2.2. HTML5

HTML5 is the next version of HTML which is a World Wide Web core markup language. The root comes from the web application 1.0 of WHATWG in 2004 as a working detail operation task [2, 3].

HTML5 as an improved from the previous HTML which shows text and hyperlinks to a web programming language which can show and implement multimedia and applications. Also, it autonomously provides audio, video, graphic handling, location information and other functions. Therefore, no additional plug-in or libraries are required. The script language itself is enough to implement various functions. Especially, web browsers supporting HTML5 allow same applications to be executed which gives it a characteristic of being hardware and operating system independent platform. The figure below shows the major features provided by HTML5.

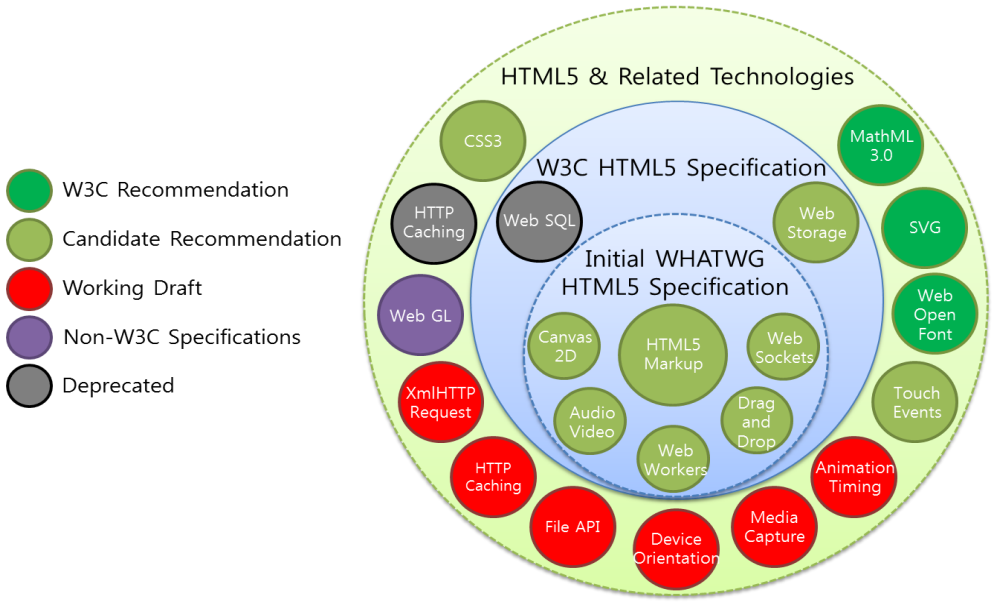


Figure 2. Components of HTML5

As shown in the Figure 2, the HTML is composed of early WHATWG HTML5 spec W3C provided functions, Web SQL and HTTP Caching and additional functions provided. HTML5 is an advancement from the previous markup language which is used to create documents to a language which provides applications [2].

Understanding the structure of HTML5 based applications is important in order to support SVM for web bases. In HTML5, each application is composed of HTML, CSS3 and JavaScript. HTML composes the entire frame of the web page using various components and properties. CSS3 is used to express the web page's exterior and design components. JavaScript is used to compose the web pages dynamically as program writing and events handling methods.

3. Design and Implementation of the SVM Interpreter

The web based **SVM** (Smart Virtual Machine) interpreter executes the contents through a stack based operation. In order to do this, the interpreter loads the SEF file and executes the relevant command. The results are reflected on the memory model and interpreter status. In

this chapter, the characteristics of the SEF format which is an executable file and information for the interpreter will be looked into. Also, the interpreter model's design and implementation process will be introduced.

3.1. SIL(Smart Intermediate Language) & SEF(Smart Executable Format)

SIL is the virtual machine code for SVM that was designed as a standardized virtual machine code model for ordinary smart devices and embedded systems [8, 9]. SIL is a stack based command set which holds independence as a language, hardware and a platform and defined based on the analysis of existing virtual machine codes such as bytecod, .NET IL and *etc*., SIL is composed of a meta-code which carries out particular jobs such as class creation and an operation code with responds to actual commands. An operation code has an abstract form which is not subordinated to specific hardware or source languages. It is defined in mnemonic to heighten readability and applies a consistent name rule to make debugging in assembly language levels easier. In addition, it has a short form operation code for optimization. SIL has 6 groups (except optimization group) of operation codes.

SEF is a smart cross platform executable file format. The SEF standard has a large effect on the implementation of SVM and interpreter. The structure of the SEF composed of the header domain which expresses the SEF file's composition, programming segment and debug domain which expressed debugging information. The content execution code exists in the program segment which is divided into code and data expression domains. The figure below shows a simplified SEF structure.

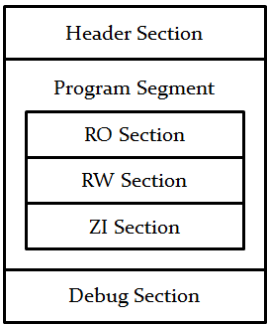


Figure 3. SEF Structure

3.2. Interpreter Model

Main purpose of the interpreter as a core module of SVM is executes instructions in the given contents, and the position of the interpreter on SVM is shown in following figure. Interpreter is a component to execute the SIL code actually, and it has subroutines for the each instruction. The subroutines matched with instructions that are simulate the instruction's semantics using loaded meta information on the memory while running the content. On while the content executing, generated or evaluated data are stored in stack and heap [10-12]. Figure 4 shows a SVM model structure with an interpreter.

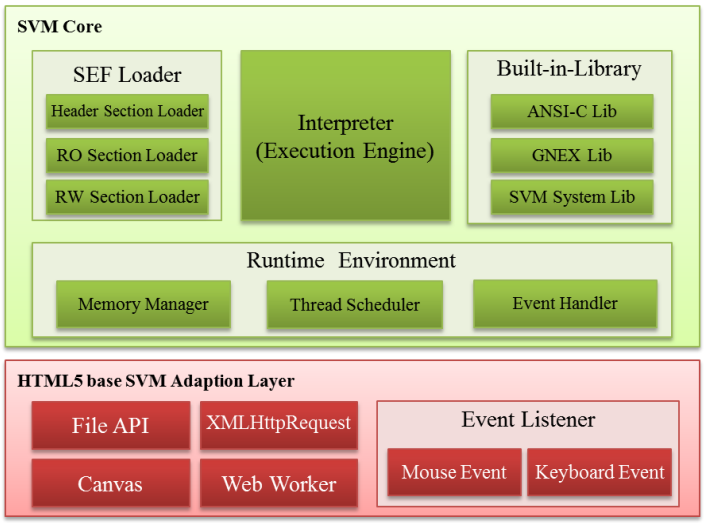


Figure 4. SVM Model Structure with an Interpreter

Figure 5 shows execution steps of the interpreter. The interpreter takes the starting address of the program extracted for the SEF file and saves it in the program counter. The value is used to execute the calculation code of the corresponding code section. Each calculation code has an execution function which the function table is mapped to. The calculation code execution is implemented by the calling of the mapped function.

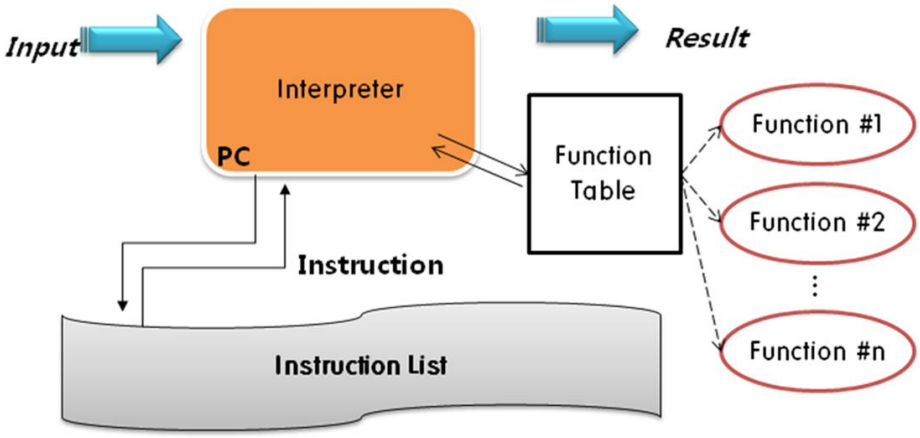


Figure 5. Interpreter Execution Model

Currently, the SIL (Smart Intermediate Language) calculation code is divided into 198 stack command, calculation command, flow control command, format conversion command, object command and others. Each function mapped to the calculation code must be implemented in the inside the interpreter.

During the execution of the calculation code, each function converts the interpreter's status and internal memory model to fit the calculation code's meaning structure. A series of calculation code execution allows the execution of the contents. The interpreter is executed until the exit corresponding calculation code appears or reaches the exit address of the code section. All the steps are implemented in JavaScript to allow execution in HTML5.

3.3. Memory Model of Interpreter

The interpreter's memory model was designed as a runtime workspace structure which is independent of hardware and platform. Runtime workspace is mainly composed of three components; runtime stack, activation record and display vector. The memory model and relationship between each components are shown in the Figure 6.

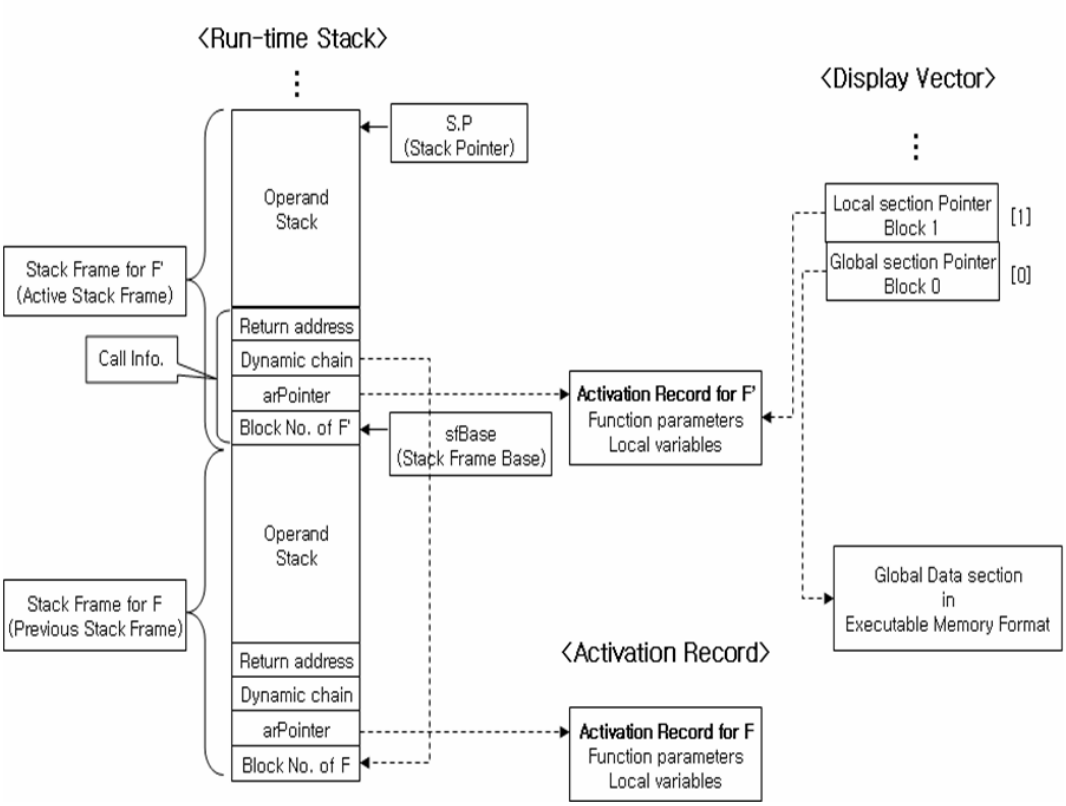


Figure 6. Interpreter Memory Configuration

Runtime stack is a memory space for operands used by the interpreter when carrying out calculations and it also contains information related to function call chain. The information is composed of reference information of activation record which is a storage space for local variables and display vector which is used to refer to higher blocks or global variables within functions.

Next, activation record is a space to store local variables. When a function is called for, one activation record is assigned and when a function is finished and stack frame is withdrawn, the activation record is also cleared. Figure 7 shows the configuration of runtime stack with multiple stack frames, and Figure 8 depicts the configuration of runtime stack with multiple stack frames, and an object reference Model on the HTML5 base SVM.

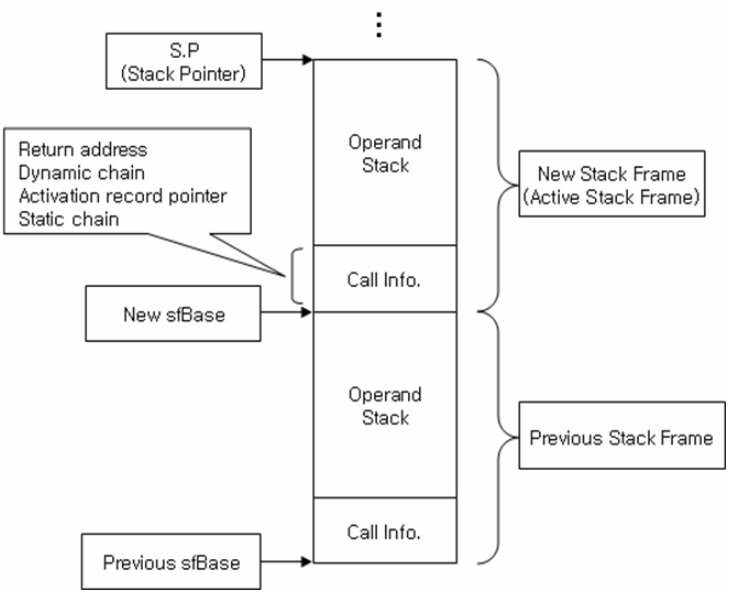


Figure 7. Structure of Runtime Stack with Multiple Stack Frame

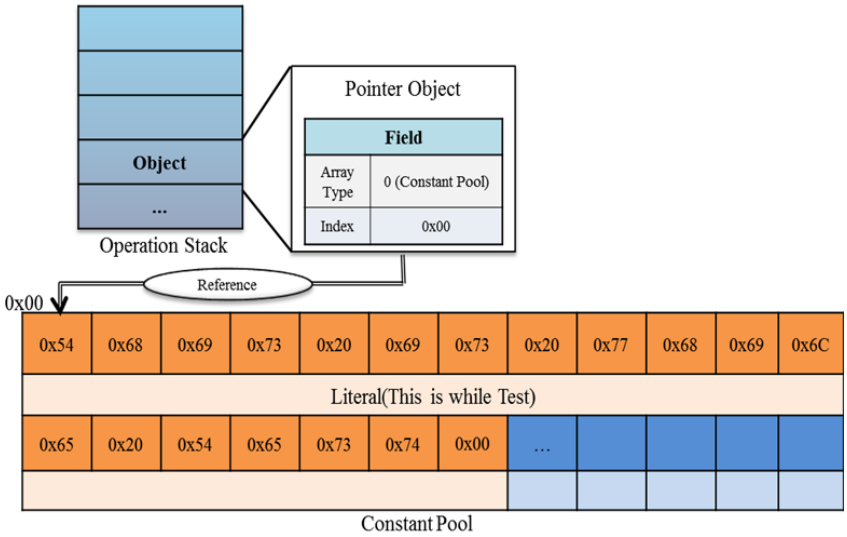


Figure 8. Object Reference Model on the HTML5 base SVM

Lastly, the display vector is information used to maintain stack chain between data domain for each stack frame. It is used to refer whole variable and local variable information.

Other than runtime workspace, a technique for referring specific memory when executing contents must be provided. Especially, the previous SVM interpreter was implemented in C/C++ while the web based interpreter was implemented in JavaScript. Therefore, the approach method to the memory provided by referring to the C/C++ can't be used. The figure 8 below shows the JavaScript based memory approach method implemented for this purpose. Array reference information and index information are expressed in object form, and using the arithmetic operation for C/C++'s pointer was created.

4. Experiments

In this paper, an interpreter for a web based SVM was designed and implemented. In this chapter, an experiment was carried out to test if the implemented interpreter was suitable as the executable technique on the web based smart cross platform. The experimental game content was composed in C language and was implemented as an event operating method.

Next, the results from executing the same contents on previous iOS SVM [4, 5] and using the interpreter implemented in this research to execute in the web browser are compared. Figure 9 shows the results of the same contents executed on the web browser.

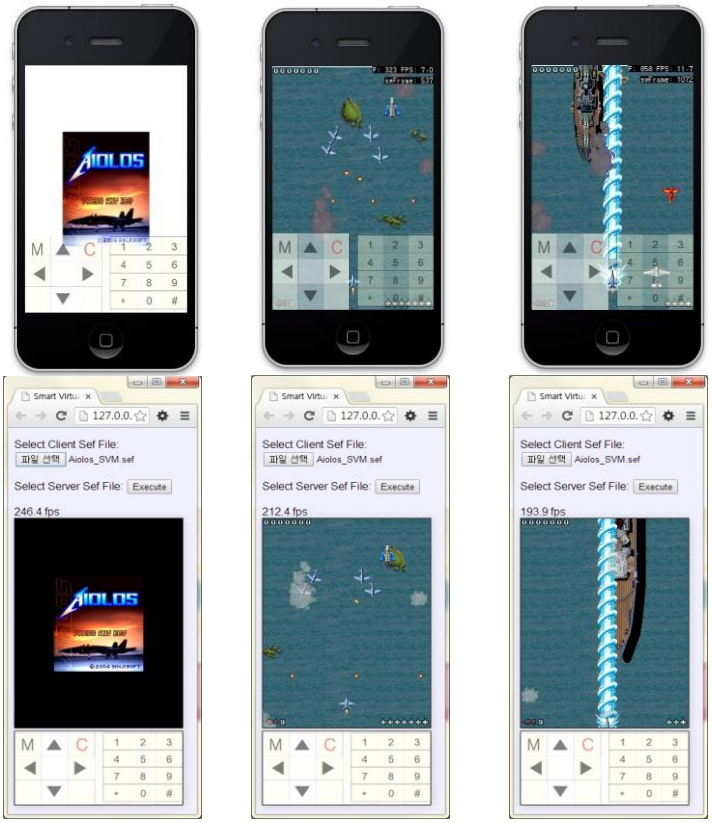
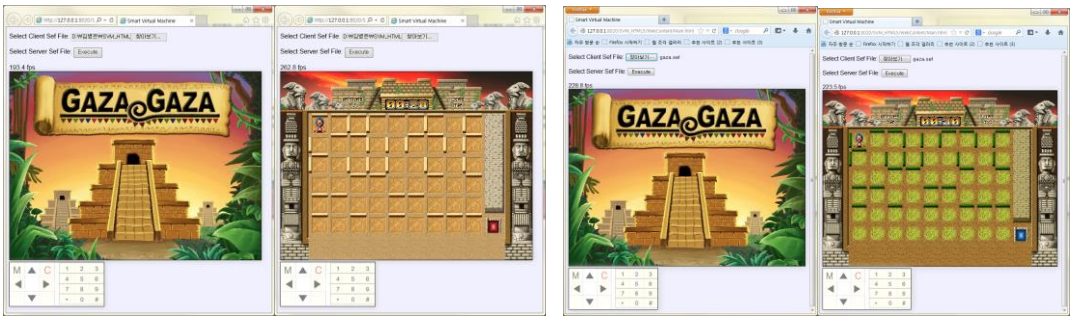
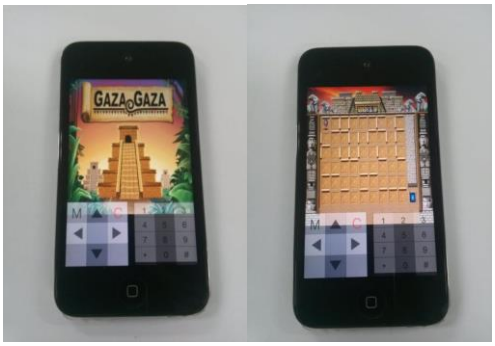


Figure 9. Execution Results for "Aiolos" Game on iOS SVM and HTML5 SVM (Google Chrome)



a) Internet Explorer b) Firefox



c) iOS SVM

Figure 10. Execution Results for "GazaGaza" Game on HTML5 SVM & iOS SVM

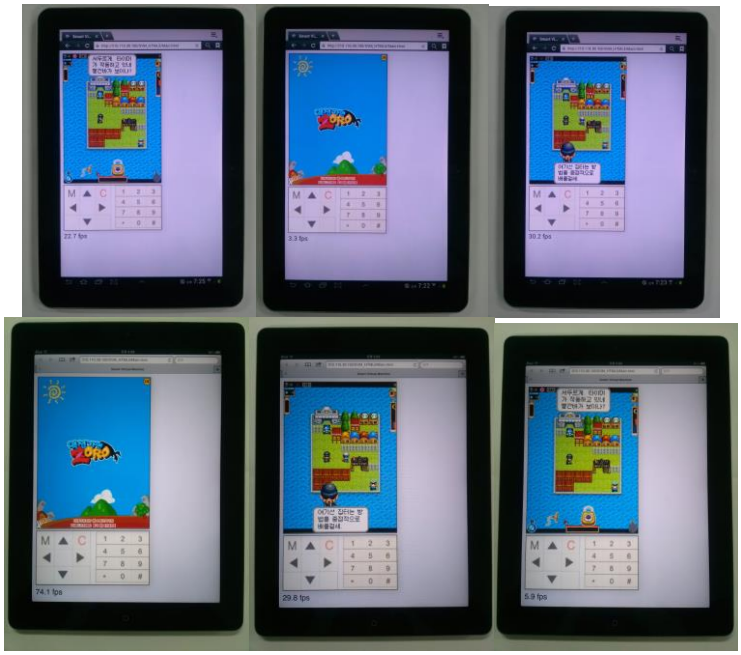


Figure 11. Execution Results for "Joro" Game on HTML5 SVM (Google Chrome & iPad2)

Figure 10 is screenshots for puzzle game content "GazaGaza". The screenshots show equivalent execution results of implemented SVM in 2 kind of different web browsers. Figure 10 c) is the execution results on the iOS SVM developed in previous research, and a) and b) are the running results on the HTML5 SVM. Through the figures, it can be seen that implemented SVM in this paper can execute the same content and generating equivalent results.

Lastly, the following Figure 11 shows the execution results on web browser provided by iOS and Android. Implemented HTML5 base SVM can be executed on the browsers with given experimental content "Joro". Upper figures are result of execution on Samsung Galaxy Tab 10.1, and bottoms are running result on iPad2.

Along with the results of executing the same contents, the performance of the web base interpreter is also an important index of evaluation. The table below provides data

to analyze the performance of the implemented interpreter. Each content have an average performance of 20fps.

5. Conclusions

The smart cross platform is a solution which allows contents developed in various languages such as C/C++, Java, Objective-C and etc. to be executed on various OS and devices. In this thesis, the smart cross platform was extended to allow execution on web base by designing and implementing an interpreter.

With this, the smart cross platform can be used to execute developed contents on web browsers supporting Windows, iOS, Android and HTML5 and the results from the experiment were used to verify. Therefore, an environment where all OS with HTML5 based web browsers installed which can execute same contents can be provided

In the future, optimization of an execution engine with JavaScript characteristics applied which will operate more efficiently than the web based interpreter. Also the native interface techniques to use HTML5 internal libraries within the contents will be researched.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and future Planning(No.2013R1A2A2A01067205).

References

[1] Y. S. Son and Y. S. Lee, “A Study on the Smart Virtual Machine for Smart Devices”, Information: An International Interdisciplinary Journal, vol. 16, no. 1465, **(2013)**.
[2] HTML Standard, http://www.whatwg.org/.
[3] Open Web Resource Site, http://www.html5rocks.com/.
[4] S. M. Han, Y. S. Son and Y.S. Lee, “Design and Implementation of the Smart Virtual Machine for Smart Cross Platform”, Journal of Korea Multimedia Society, vol. 16, no. 190, **(2013)**.
[5] Y. S. Son, J. H. Kim and Y. S. Lee, “Design and Implementation of the Smart Virtual Machine on iOS Platform for the Mobile Game Portability”, International Journal of Smart Home, vol. 8, no. 23, **(2014)**.
[6] Apple, iOS Reference Library, iOS Technology Overview, http://developer.apple.com/devcenter/ios.
[7] Goole, Android-An Open Handset Alliance Project, http://code.google.com/intl/ko/android/.
[8] Y. S. Son and Y. S. Lee, “A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms”, International Journal of Smart Home, vol. 6, no. 93, **(2012)**.
[9] S. L. Yun, D. G. Nam, S. M. Oh and J. S. Kim, “Virtual Machine Code for Embedded Systems,” International Conference on CIMCA, Osaka, Japan, **(2004)**.
[10] Y. S. Lee, “The Virtual Machine Technology for Embedded Systems”, Journal of the Korea Multimedia Society, vol. 6, **(2002)**, pp. 36-44.
[11] Y. S. Son and Y. S. Lee, “Design and Implementation of the Virtual Machine for Smart Devices”, Proc. of the 2011 Fall Conference, Korea Multimedia Society, **(2011)**.
[12] J. E. Smith, “Virtual Machines”, Morgan Kaufmann, **(2005)**.
[13] Y. S. Son and Y. S. Lee, “An Objective-C Compiler to Generate Platform-Independent Codes in Smart Device Environments”, Information: An International Interdisciplinary Journal, vol. 16, no. 1459, **(2013)**.
[14] M. Berndl, B. Vitale, M. Zaleski and A. Brown, “Context Threading: A Flexible and Efficient Dispatch Technique for Virtual Machine Interpreters”, International Symposium on Code Generation and Optimization, **(2005)**.
[15] Y. S. Son and Y. S. Lee, “The Contents Execution Technique for the Web-based Smart Virtual Machine”, Workshop on Multimedia, **(2014)**.

Authors

Yunsik Son, He received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 2004, and M.S. and Ph.D. degrees from the Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 2006 and 2009, respectively. Currently, he is a Researcher of the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea. His research areas include smart system solutions, secure software, programming languages, compiler construction, and mobile/embedded systems.

Seman Oh, He received the B.S. degree from the Seoul National University, Seoul, Korea, in 1977, and M.S. and Ph.D. degrees from the Dept. of Computer Science, Korea Advanced Institute of Science and Technology, Seoul, Korea in 1979 and 1985, respectively. He was a Dean of the Dept. of Computer Science and Engineering, Graduate School, Dongguk University from 1993-1999, a Director of SIGPL in Korea Institute of Information Scientists and Engineers from 2001-2003, a Director of SIGGAME in Korea Information Processing Society from 2004-2005. Currently, he is a Professor of the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea. His research areas include smart system solutions, programming languages, and embedded systems.

YangSun Lee, He received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 1985, and M.S. and Ph.D. degrees from Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 1987 and 2003, respectively. He was a Manager of the Computer Center, Seokyeong University from 1996-2000, a Director of Korea Multimedia Society from 2004-2014, a Vice President of Korea Multimedia Society from 2005-2006, 2009. Also, he was a Director of Korea Information Processing Society from 2006-2013 and a President of a Society for the Study of Game at Korea Information Processing Society from 2006-2010. And, he was a Director of Smart Developer Association from 2011-2014. Currently, he is a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include smart mobile system solutions, programming languages, and embedded systems.