International Journal of Smart Home
Vol.8, No.5 (2014), pp.11-24
http://dx.doi.org/10.14257/ijsh.2014.8.5.02

A Study on the Android-to-Bada Smart Game Content Converter

JaeHyun Kim[1] and YangSun Lee[1*]

[1]*Dept. of Computer Engineering, Seokyeong University*
16-1 Jungneung-Dong, Sungbuk-Gu, Seoul 136-704, KOREA
*statsr@skuniv.ac.kr, *yslee@skuniv.ac.kr*

Abstract

The Android platform developed by Google is an optimized platform for mobile devices with a perfect combination of an operating system, middleware, and application programs. Bada platform developed by Samsung in korea is a smart phone platform, which is created for wide range of device. "Bada" is a Korean word that means "ocean" and "seashore". "bada" in itself embodies the open possibilities of the ocean. Due to the use of different smart phone platforms, developers must create content designed specifically for each platform or use a conversion process to provide game content to consumers. In this paper, to resolve this problem, the Android-to-Bada game converter was designed to automatically translate game contents from the Android platform to the Bada platform for smart phones using a content analyzer, a resource converter, a platform mapping engine and a source translator. Through the Android-to-Bada converter, resources such as images and sounds can be converted, APIs can be converted using a platform mapping engine. These and all other content conversion functions were examined. Test results indicate that the graphics, image output, sound output, and other functions of converted Bada games were equivalent to those of the Android games before conversion.

Keywords: *Android-to-Bada Converter, Automatic Smart Game Converter, Android, Bada, Content Analyzer, Resource Converter, Platform Mapping Engine, Source Translator*

1. Introduction

The Android platform developed by Google is an optimized platform for mobile devices with a perfect combination of an operating system, middleware, and application programs [1-3]. Bada platform developed by Samsung is a smart phone platform, which is created for wide range of device. "Bada" is a Korean word that means "ocean" and "seashore". "bada" in itself embodies the open possibilities of the ocean[4-6]. Due to the use of different mobile platforms such as Android, iOS, and Bada for each of the mobile communications companies, mobile contents developers must repeat development process to create different versions of games that match the different characteristics of the different smart phone platforms if they aspire to service their games. This has led to the need for developers to convert contents that have been already developed for use on smart phone platforms. converting (porting and retargeting) Consequently, considerable time and expense are being invested to analyze and convert(port and retarget) the sources and resources of one smart game's content for use on the smart phone platform [7-15].

In this paper, to resolve this problem, the Android-to-Bada automatic smart game converter system was designed to automatically translate game contents from the Android platform to the Bada platform for smart phones. The Android-to-Bada converter consists of a

* Corresponding Author

ISSN: 1975-4094 IJSH

content analyzer, resource converter, platform mapping engine and the source translator [8-14]. The content analyzer [10-11] analyzes the content that is input, and produces an output in which the resource data and source code stored within the content are separated. The resource converter [10-12, 14-15] is a system which converts the text or binary resource data from the game to be converted into image, sound and user data so that it can be used on the target platform's file system. The platform mapping engine [10, 12, 14-15] is a system which provides API functions which allow the previous platform's execution environment to be recreated using the target platform's wrapper functions. The source translator [8, 10, 13] receives the Android source codes that have been produced by the contents analyzer and translates it into Bada source codes that are semantically equivalent and carry out the same function on the Android platform.

By automatically converting the Android game contents to the Bada smart phone game contents, the Android game contents can be retargeted quickly to a Bada platform. As a result, the reusability will be increased, and the labor, time, and cost will be reduced.

2. Related Studies

2.1. Android

The Android platform developed by Google is an optimized platform for mobile devices with a perfect combination of an operating system, middleware, and application programs. The Android platform adopted an open source policy, and it consists of the Linux kernel, library, run-time environment, application framework, and applications. Figure 1 depicts the Android platform's hierarchical structure and components.

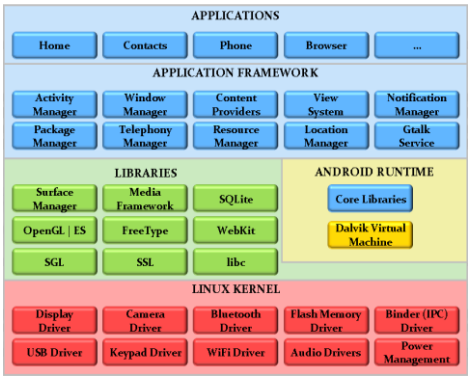

Figure 1. Android Platform's System Configuration

The Linux kernel uses core system services such as Linux version 2.6, which includes security, memory management, process management, a network stack, and a driver model. The kernel functions as an abstraction layer between the hardware and the software. The library consists of C and C++ and provides a C system library, media library, 3D library, and more. The application framework is a package component consisting of Java; applications can be created using the packages of this framework.

All application programs for the Android platform are created using Java and converted into class files through the Java compiler. After once conversion again into Dalvik Executable (DEX) files, they are executed using the Dalvik virtual machine [1-3].

2.2. Bada

Bada platform developed by Samsung is a smart phone platform, which is created for wide range of device, unveiled in 2010. "Bada" is a Korean word that means "ocean" and "seashore". "bada" in itself embodies the open possibilities of the ocean: it can accommodate the various applications created by developers and it provides an interesting new space that offers unprecedented enjoyment to its users. Figure 2 depicts the Bada platform's hierarchical structure and components.

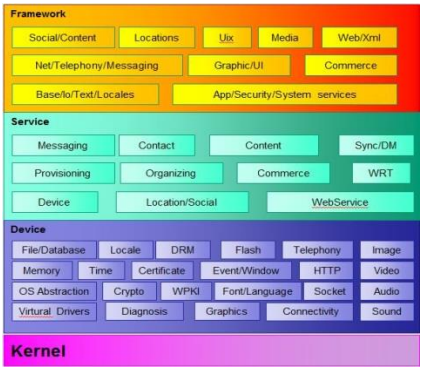

Figure 2. Bada Platform's System Configuration

The Bada platform consists of a kernel layer, a device layer, a service layer, and a framework layer. The kernel layer contains either the real-time operating system or the Linux kernel, depending on device hardware configuration. The device layer contains the core functions of the mobile device platform. The service layer contains the service-oriented functions that are provided by application engines and server-assisted components. The application engines provided by the service layer include Contact and Messaging engines. The framework layer contains C++ and Web frameworks of bada. The C++ framework consists of the application framework, as well as interfaces and classes that provide access to the functionality on the underlying layers. The Web framework provides well-established standards and features, such as HTML, CSS, and JavaScript, as well as JavaScript-based cross-platform APIs for UI controls and events [4-6].

2.3. Existing Mobile Game Converters

To date, despite the very active mobile market, there has been a lack of research on mobile content converters, so there are few examples to which we can refer. Furthermore, existing content converters generally only allow conversion of content having a similar programming language environment or do not allow automatic conversion at all. The reality is that programmers must convert content by hand.

An existing feature phone mobile game content converter using XML has attempted to convert Java content [16-19]. In addition, the functions of the API used in the source code to be converted were imitated and redefined using wrapper functions. Therefore, there is no need to convert the source code if the same functions are used. The mutual conversion of BREW C and WIPI C [20] and the conversion GVM C into BREW C [21] have been examined; however, these studies were flawed because the source code was not automatically converted, so users had to intervene and convert it manually.

On the other hand, studies of automatic conversion of mobile game content using a compiler writing system [22-23] have been attempted. A method of increasing the reusability of game content and enhancing productivity by converting the mobile C content of the GVM platform into WIPI C, Java, or MIDP Java has been suggested in feature phone [7]. In addition, other studies are underway to convert existing mobile game content for use in the growing smart phone market for operating systems such as Android, iOS and Windows Phone [8-15], for example, GNEX-to-Android converter [8], GNEX-to-iOS converter [9], the WIPI-to-iOS converter [10], the WIPI-to-Windows Mobile converter [12], WIPI-to-Android converter [13], iOS-to-Android converter [14], iOS-to-Bada converter [15], Android-to-iOS converter system.

3. The Android-to-Bada Game Converter

The Android-to-Bada automatic smart game converter receives Android game contents in source form and converts it into the source form that is run on the Bada platform. Figure 3 shows a model of the Android -to-Bada automatic smart game content converter system.

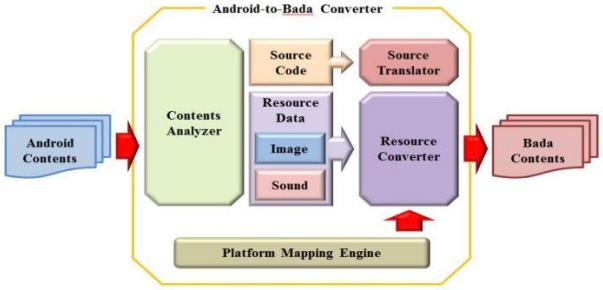

Figure 3. The Android-to-Bada Smart Game Converter System

3.1. Content Analyzer

The content analyzer [7-11] is a system that analyzes the content that is input, and produces an output in which the resource data and source code stored within the content are separated. Thus, the content analyzer must separate the image or sound resource data, in the form of variables, according to the variable. Then it must create a list of resources, including information on the structure, and deliver the data to the resource converter. The remaining source code, excluding the resource components, is delivered to the source translator. The Android content analyzer receives the Android content as an input and analyzes it so that it can be easily converted into content for the Bada platform. Then it divides the files into source files, resource files, and other files. Figure 4 shows a model of the Android game content analyzer.

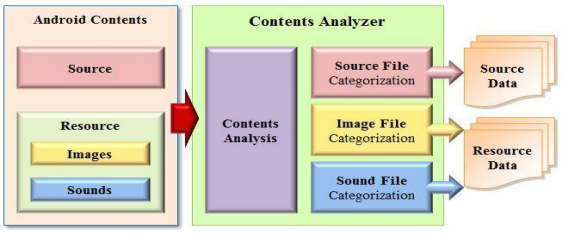

Figure 4. The Android Game Content Analyzer

Figure 5 shows the result of the content analyzer. The content analyzer creates a Converter folder containing copies of all the files within the original folder in order to prevent changes to the original files, and then categorizes the files as described above.

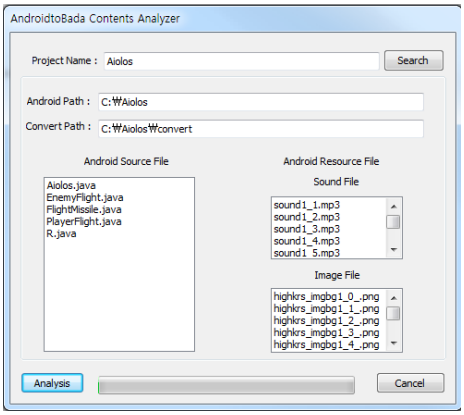

Figure 5. Result of the Android Game Content Analyzer

3.2. Resource Converter

The resource converter [7, 10-12, 14-15] is a system that converts the resource data, which is in text or binary form, into image data, sound data, and user data for use in the target platform's file system. The image file formats used in each platform (*e.g.*, BMP, PNG, JPEG), sound formats (*e.g.*, WAV, MP3, MMF), and user data must be researched and converted for use in the target platform. Thus, the image file formats provided in existing smart platforms and sound file formats can all be used in the Android and Bada platforms. Consequently, only a management file must be created to show which resources have been used. The Android-to-Bada resource converter receives, analyzes, and produces files from the Android content, including resource files such as image and sound files, and converts them into resource file formats for the target Bada platform. Figure 6 shows a model of the resource converter system.

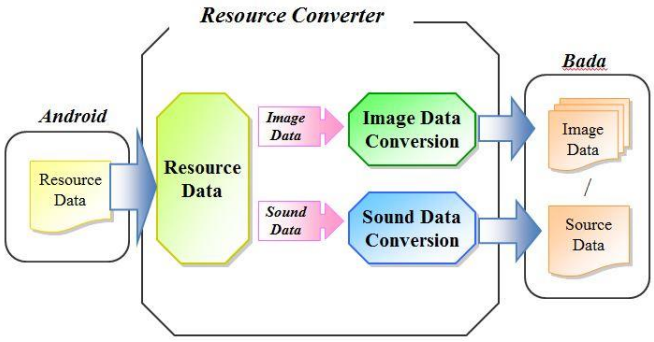

Figure 6. Android Resource Converter System

Figure 7 shows the result of the resource converter.

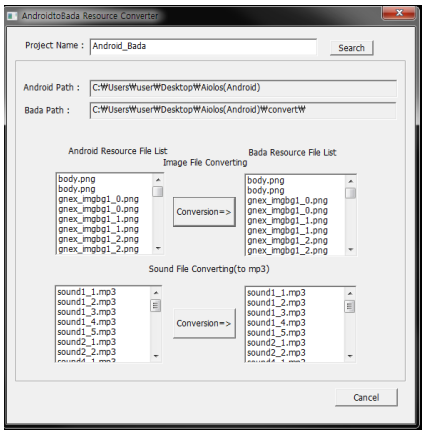

Figure 7. Result of the Android-to-Bada Resource Converter

3.3. Platform Mapping Engine

The platform mapping engine [7-10, 12, 14-15] creates an execution environment on the Bada platform that is identical to that in the Android platform so that the same environments can be executed identically. Thus, Android is enabled to run in its original form on the Bada platform. On the basis of the created execution environment, a wrapper function format is provided that enables identical execution of the Android API on the Bada platform.

The only function of the source translator is to automatically translate the Android platform's source program into the Bada platform's source program. However, the translated program cannot be run on the Bada platform right away. The Android's API, system variables, and system environment used in the source code must be converted into or matched with formats that can be used on the Bada platform. Figure 8 shows the structure of the Android-to-Bada platform mapping engine, in which API and runtime environment from Android can be used in Bada platform through wrapper functions.

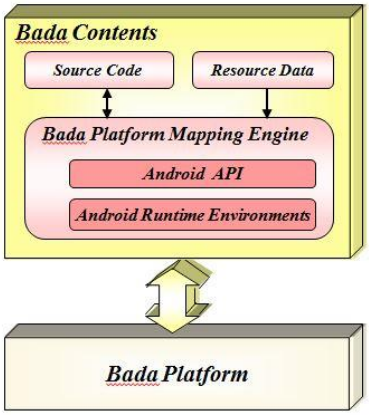

Figure 8. Platform Mapping Engine System Model

3.4.1. System Environments: The Android program's life cycle is divided into six steps: Created(onCreate()), Started(onStart()), Resumed(onResume()), Paused(onPause()), Stopped(onStop()) and Destoryed(onDestroy()). Each step is a step in the application's life cycle and each time a change is made, onCreate(), onStart(), onResume(), onPause(), onStop() and onDestroy() methods are called respectively. These methods are very important as they

are automatically called when converted in the system step and becomes the base of an Android platform system.

In Bada, slightly different to Android, it has four life cycle steps: Initialzing, Running(Foreground), Running(Background), and Terminating. Each time a stage is changed, OnAppInitializing, OnForeground, OnBackground, OnAppTerminating methods are called respectively. In this paper, these points were put into consideration and the Android platform's system environment was matched one on one to the Bada platform's system environment. For example, Android's onResume and Bada's OnForeground have a similar role. Therefore, when onResume in Android was called for, OnForeground method in Bada was called. Figure 9 shows an Android life cycle mapped to the Bada life cycle.

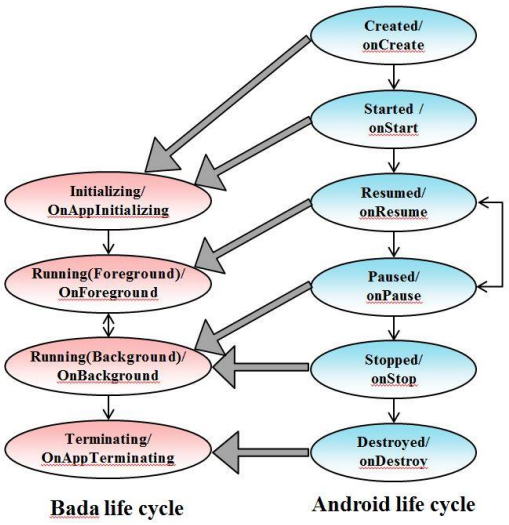

Figure 9. System Mapping between Android and Bada

3.4.2 Graphic Environments

In Android, the image is called as a Bitmap, received as a Canvas and uses onDraw to draw in View. The setContectView method is used to put in Activity and it is called by Intent to display the graphic on the Device Screen. The graphic environment similar to Android was built for Bada. In Bada, the Form class and Canvas class play similar roles to Android's graphic environment. Form class and Canvas class were used to build a graphic environment that runs in Android. Bada has the Canvas class, a drawing tool that is similar to the CGContext class. Figures 10 and 11 shows the LCD output methods of iOS and Bada, respectively.

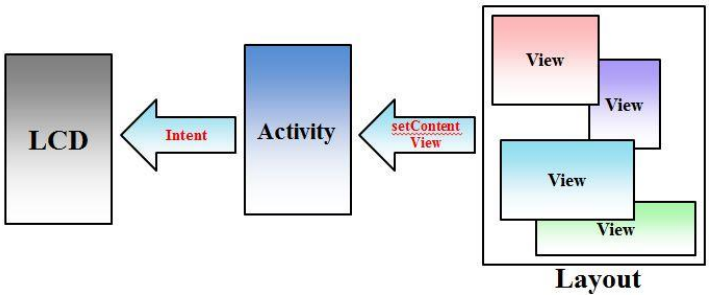

Figure 10. Android's Graphic Output Method

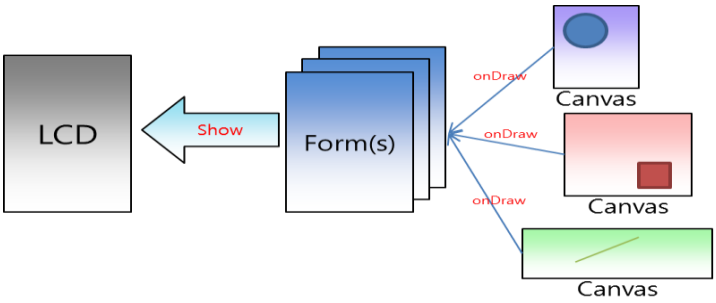

Figure 11. Bada's Graphic Output Method

3.4.3. Library Functions (APIs)

Bada's API has been implemented to use the same name of Android API in Bada's platform in order to functions exactly the same. Bada's class and method are provided through the Namespace Library and if there are classes or methods required, it uses the Namesapce Library. The translated Bada source code appears the same as Android API function by the wrapper function. Therefore, it can be used in the same form as Android which allows to keep the original form of the source and can be executed in Bada platform without further modifications. Table 1 shows an API mapping table supported in Android and Bada.

Table 1. Supported API Mapping Table between Android and Bada

class	Android APIs	Bada APIs
System(3)	GetDate, GetTime, Exit	GetDate, GetTime, Exit
Handset Control(11)	seekTo, start, pause, setVolume, vibrate, cancel, SetBackLight, GetUserNV, PutUserNV, Alive, LastTime, Slice, Repeat	PlaySound, StopSound, SetVolume, StartVib, StopVib, SetBackLight, GetUserNV, GetUserNV, PutUserNV, SetTimer, SetTimer1, SetTimer2, ResetTimer, ResetTimer1, ResetTimer2
String(15)	length, StrCpy, delete, append, substring, charAt, insert, parseInt, PutByte, GetByte, toCharArray, toString, format	StrLen, StrCpy, StrSub, StrCat, GetChar, PutChar, AsciiToInt, PutByte, GetBytes , MakeStr1, MakeStr2, MakeStr3
Graphic(24)	setAlpha, SetClip, ResetClip, SetActiveBuffer, setColorFilter, SetColor, setTextSize, SetFontColor, SetFontAlign, SetFontType, SetPalette, eraseColor, postRotate, setPixel, createBitmap, drawBitmap, setStyle, getWidth, getHeight, drawLine, drawRect, drawText, drawRoundRect, postInvalidate	SetClip, SetClipBounds, SetActiveBuffer SetAlpha, SetGamma, SetColorComponents, SetFont, SetTextColor, SetVerticalAlignment, SetHorizontalAlignment, SetTextAbbreviationEnabled,SetFontType, DrawText, SetBounds, DrawStrSolid, DrawStrSolid2, FillRectangle, DrawBitmap, DrawLine, Start
Mathermatics (8)	abs, nextInt, sin, cos, toRadians, ArrayToVar, ArrayToArray, HitCheck	Abs, Rand, RandRatio, Sin100, Cos100, ArrayToVar, ArrayToArray, HitCheck

3.4. Source Translator

The source translator receives the Android source codes that are output by the contents analyzer and translates them into Bada source codes which are semantically equivalent and execute the execute the same actions as the Android contents. Source translators have been created so that they can overcome the differences of the platforms and automatically translate the game source programs using compiler writing technology. Compiler technology analyzes

programming grammar and syntax and provides a method for automatic translation into another language [7-8, 10, 13]. Figure 12 is a depiction of the source translator model.

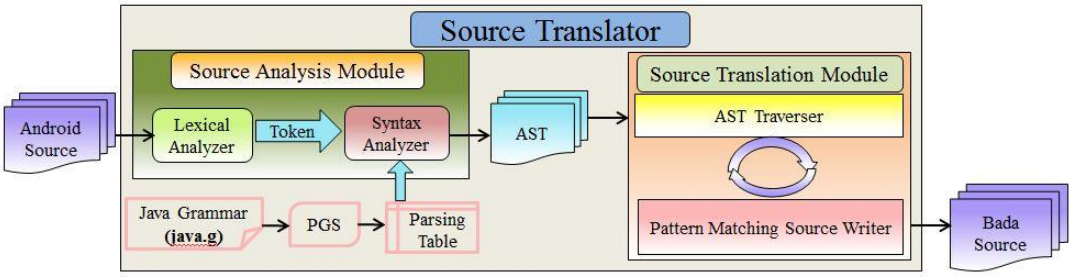

Figure 12. Source Translator Model

Source translators can be largely divided into the source analysis module and source translation module. The source analysis modules receive Android source code inputs and carries out lexical and syntax analysis to create an Abstract Syntax Tree (AST). The lexical analyzer analyzes the source codes and delivers the results to the syntax analyzer. The syntax analyzer uses the token information obtained from the lexical analyzer and the parsing table created by the Parser Generating System (PGS) to analyze the syntax of the program. The results of the syntax analyze output a syntax tree for correct syntax. This tree is the Abstract Syntax Tree (AST) which is used in the source translation module.

The source translation module traverses the AST and generates Bada source codes which are semantically equivalent to the Android source codes. The source translation module which receives AST as an input, begins a successive search from the tree's most significant root. During the search process, if a significant node appears, the pattern matching source writer receives the node and translates it into a Bada sentence. When the entire AST search process is finished, the pattern matching source writer analyzes the nodes until now and creates each of the translated source codes into one file, this is the Bada source code. Figure 13 shows the execution result of the source translator.

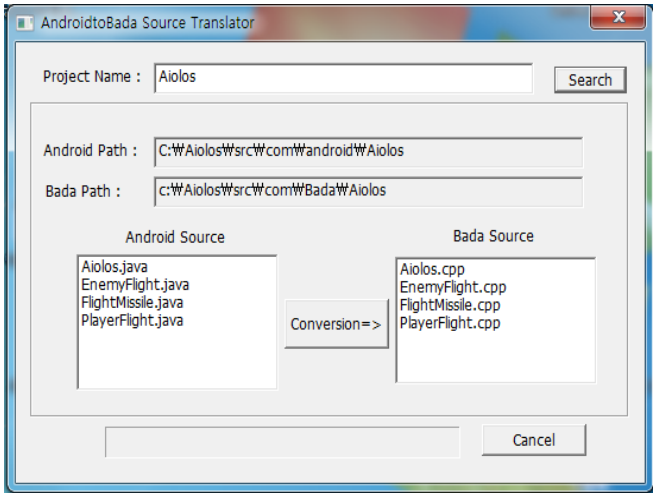

Figure 13. Execution Result of the Source Translator

Figure 14 shows an example of searching the AST of an Android program created by the syntax analyzer and converting it into a Bada program.

AST

```
Nonterminal: PROGRAM
    Nonterminal: CLASS_DCL
        Terminal: SuperClass
        Nonterminal: CLASS_BODY
            Nonterminal: FIELD_DCL
                Nonterminal: DCL_SPEC
                    Nonterminal: INT_TYPE
                Nonterminal: VAR_ITEM
                    Nonterminal: SIMPLE_VAR
                        Terminal: a
                Nonterminal: VAR_ITEM
                    Nonterminal: SIMPLE_VAR
                        Terminal: b
            Nonterminal: CONSTRUCTOR_DCL
                Nonterminal: CONSTRUCTOR_ITEM
                    Nonterminal: SIMPLE_NAME
                        Terminal: SuperClass
                Nonterminal: CONSTRUCTOR_BODY
                                (start : 3  end : 5)
                    Nonterminal: EXPRESSION_ST
                        Nonterminal: ASSIGNMENT
                            Nonterminal: SIMPLE_NAME
                                Terminal: a
                            Nonterminal: ASSIGN_OP
                            Terminal: 1
                    Nonterminal: EXPRESSION_ST
                        Nonterminal: ASSIGNMENT
                            Nonterminal: SIMPLE_NAME
                                Terminal: b
                            Nonterminal: ASSIGN_OP
                            Terminal: 2
            Nonterminal: CONSTRUCTOR_DCL
                Nonterminal: CONSTRUCTOR_ITEM
                    Nonterminal: SIMPLE_NAME
                        Terminal: SuperClass
                    Nonterminal: PARAM_DCL_LIST
                        Nonterminal: PARAM_DCL
                            Nonterminal: DCL_SPEC
                                Nonterminal: INT_TYPE
                            Nonterminal: SIMPLE_VAR
                                Terminal: a
                        Nonterminal: PARAM_DCL
                            Nonterminal: DCL_SPEC
                                Nonterminal: INT_TYPE
                            Nonterminal: SIMPLE_VAR
                                Terminal: b
                Nonterminal: CONSTRUCTOR_BODY
                                (start : 6  end : 8)
                    Nonterminal: EXPRESSION_ST
                        Nonterminal: ASSIGNMENT
                            Nonterminal: INSTANCE_FIELD
                                Nonterminal: THIS
                                Terminal: a
                            Nonterminal: ASSIGN_OP
                            Nonterminal: SIMPLE_NAME
                                Terminal: a
                    Nonterminal: EXPRESSION_ST
                        Nonterminal: ASSIGNMENT
                            Nonterminal: INSTANCE_FIELD
                                Nonterminal: THIS
                                Terminal: b
                            Nonterminal: ASSIGN_OP
                            Nonterminal: SIMPLE_NAME
                                Terminal: b
    Nonterminal: CLASS_DCL
        Terminal: SubClass
        Nonterminal: EXTENDS
            Nonterminal: CLASS_INTERFACE_TYPE
                Nonterminal: SIMPLE_NAME
                    Terminal: SuperClass

                        :
                        :
```

Android program

```
class SuperClass{
        int a, b;
        SuperClass(){
                a = 1; b = 2;
        }
        SuperClass(int a, int b){
                this.a = a; this.b = b;
        }
}
class SubClass extends SuperClass{
        int c;
        SubClass(){
                c = 3;
        }
        SubClass(int a, int b, int c){
                super(a, b);
                this.c = c;
        }
}
public class Exam{
        public static void main(String[] args){
                SubClass obj1 = new SubClass();
                SubClass obj2 = new SubClass(4, 5, 6);
                System.out.printf("a = %d, b = %d, c = %d\n",
                                obj1.a, obj1.b, obj1.c);
                System.out.printf("a = %d, b = %d, c = %d\n",
                                obj2.a, obj2.b, obj2.c);
        }
}
```

Bada program

```
class SuperClass{
public:
        int a; int b;
        SuperClass(){
                a = 1; b = 2;
        }
        SuperClass(int a, int b){
                SuperClass::a = a;
                SuperClass::b = b;
        }
};
class SubClass : public SuperClass{
public:
        int c;
        SubClass() : SuperClass(){
                c = 3;
        }
        SubClass(int a, int b, int c): SuperClass(a, b){
                SubClass::c = c;
        }
};
int main(void){
    SubClass *obj1 = new SubClass();
    SubClass *obj2 = new SubClass(4, 5, 6);
    AppLog("a = %d, b = %d, c = %d\n", obj1->a, obj1->b, obj1->c);
    AppLog("a = %d, b = %d, c = %d\n", obj2->a, obj2->b, obj2->c);
}
```

Figure 14. Example of the AST and a Source Translation

4. Experimental Results

In this paper, the Android-to-Bada smart game converter was used to automatically convert smart game contents from the Android platform to the Bada platform in source form. The

results of the conversion were then compared. The emulators that were used to execute the content are the Android 4.0.3 emulator and the Bada 2.0.6 emulator.

Table 2. Platform and Emulator

Platform	Emulator	Method
Android	Android 4.0.3 Emulator	Native
Bada	Bada 2.0.6 Emulator	Native

Figure 15, 16 and Figure 17 compare the execution of the games such as "Aiolos", "Elemental Force" and "Joro".

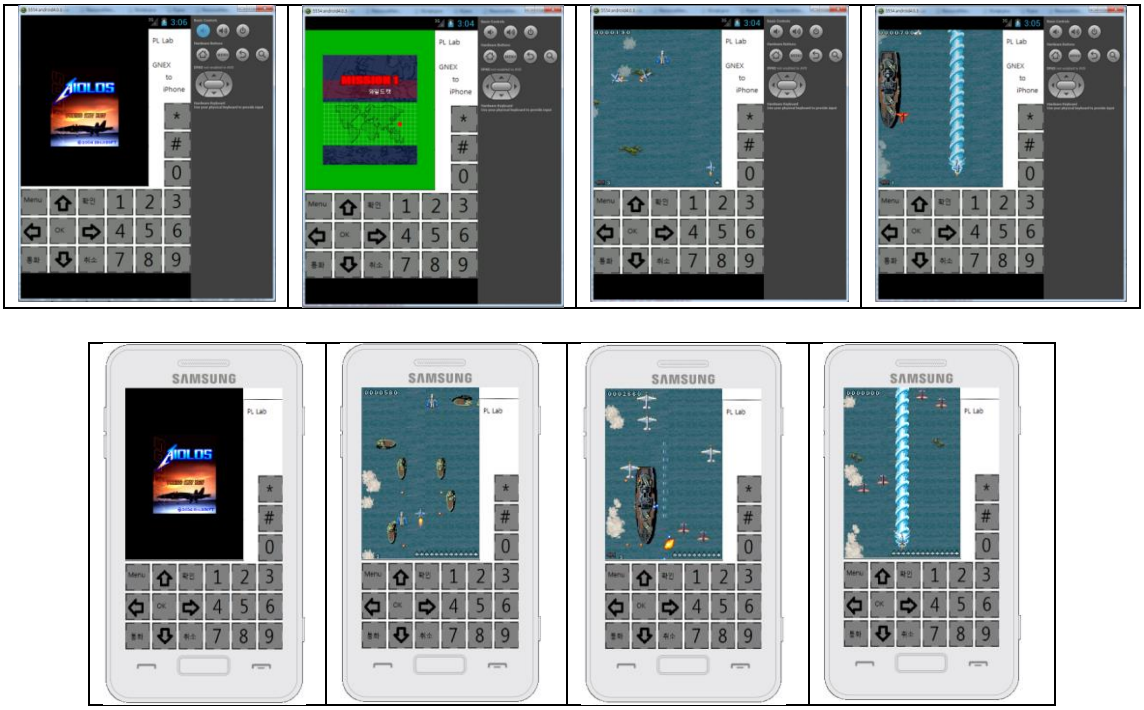

Figure 15. Execution Results of the Game "Aiolos" in Android and Bada

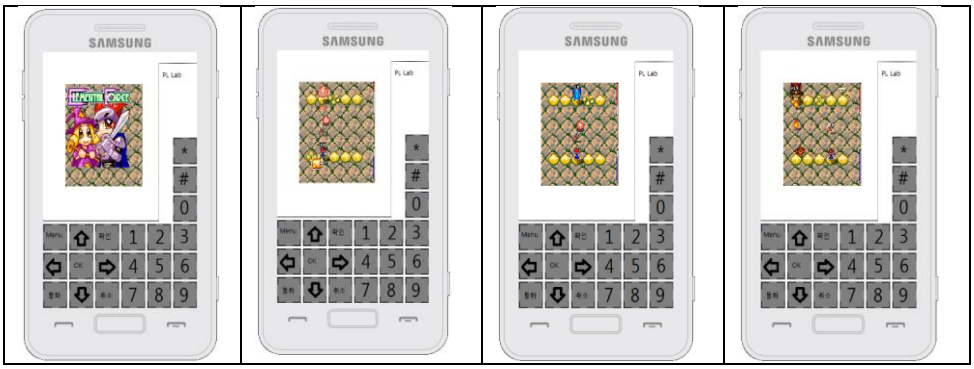

Figure 16. Execution Results of the Game "Elemental Force" in Android and Bada

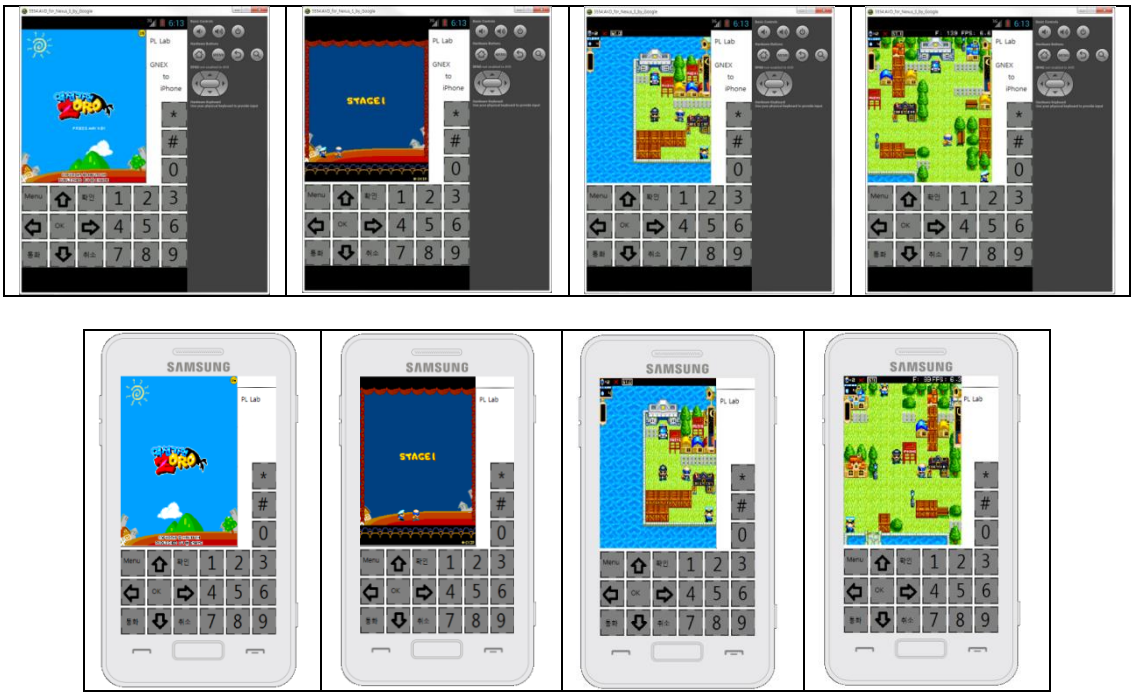

Figure 17. Execution Results of the Game "Joro" in Android and Bada

5. Conclusions and Further Researches

The Android-to-Bada automatic smart game converter presented in this paper offers a means to solve the problems of different smart platforms. It can ensure quick and automatic conversion of existing Android game content into game content for the Bada platform, thus increasing the reusability of existing content and providing smart phone users with more diverse content. In addition, the time and expense required throughout the development and conversion processes can be significantly reduced. Consequently, productivity can be enhanced, and the time and expense thus saved can be invested in developing new game content. In the future, research on increasing the running speed of games and experiments in actual devices' environments are expected to create optimized graphics, source code translations, and APIs for each platform and device. It is also expected that by supplementing

and expanding the converter systems' functions, existing content can be run on various of the increasingly numerous smart phone platforms: Android, iOS, Windows Phone, and Bada.

Acknowledgements

This Research was supported by Seokyeong University in 2012.

References

[1] Google, Android, http://code.google.com/intl/ko/android/
[2] R. Schwarz, P. Dutson, J. Steele and N. To, "The Android Developer's Cookbook: Building Applications with the Android Sdk", Addison-Wesley Professional, **(2013).**
[3] P. Deitel, H. Deitel and D. MacLean, "Android: How to Program", Prentice Hall, **(2014).**
[4] Samsung, Overview of Bada, http://developer.bada.com /library/help
[5] D. K. Jung, "Bada Mobile Programming", Answer Books, **(2011).**
[6] J. I. Ahn, S. H. Jo and J. S. Kim, "Bada Phone Programming", Books Holic **(2011).**
[7] Y. S. Lee, "Design and Implementation of the GNEX C-to-WIPI Java Converter for Automatic Mobile Contents Translation", Journal of Korea Multimedia Society, vol. 13, no. 609, **(2010).**
[8] Y. S. Son, S. M. Oh and Y. S. Lee, "Design and Implementation of the GNEX C-to-Android Java Converter using a Source-Level Contents Translator", Journal of Korea Multimedia Society, vol. 13, no. 1051, **(2010).**
[9] Y. S. Lee, H. J. Choi and J. S. Kim, "Design and Implementation of the GNEX-to-iPhone Converter for Smart Phone Game Contents", Journal of Korea Multimedia Society, vol. 14, no. 577, **(2011).**
[10] Y. S. Lee, "Automatic Mobile Contents Converter for Smart Phone Platforms", Journal of Korea Multimedia Society, vol. 15, no. 54, **(2011).**
[11] Y. S. Lee, J. S. Kim and M. J. Kim, "Development of the Contents Analyzer and the Resource Converter for Automatic Mobile Contents Converter", Journal of Korea Multimedia Society, vol. 14, no. 681, **(2011).**
[12] Y. S. Lee and Y. S. Son, "A Study on the WIPI-to-Windows Mobile Game Contents Converter using a Resource Converter and a Platform Mapping Engine", Information-an International Interdisciplinary Journal, International Information Institute, vol. 16, no. 2157, **(2013).**
[13] Y. S. Lee, Y. S. Son, "A Study on the Source Translator of the WIPI-to-Android Mobile Game Converter", Information-an International Interdisciplinary Journal, International Information Institute, vol. 16, no. 739, **(2013).**
[14] J. H. Kim and Y. S. Lee, "A Study on the iOS-to-Bada Converter using a Resource Converter and a Platform Mapping Engine", International Journal of Software Engineering and Its Applications, SERSC, vol. 8, no. 73, **(2014).**
[15] J. H. Kim and Y. S. Lee, "A Study on the iOS-to-Android Converter using a Resource Converter and a Platform Mapping Engine," International Journal of Software Engineering and Its Applications, SERSC, vol. 8, no. 427, **(2014).**
[16] S. H. Kim, "Design and Implementation of A Mobile Contents Conversion System based on XML using J2ME MIDP", Master's Thesis, Hannam University **(2003).**
[17] Y. S. Kim and D. C. Jang, "A Design for Mobile Contents Converting Using XML Parser Extraction", Journal of Korea Multimedia Society, vol. 6, no. 267, **(2003).**
[18] S. I. Yun, "Integrated Conversion System for Wired and Wireless Platform based on Mobile Environment", Ph.D. Thesis, Hannam University, **(2003).**
[19] Y. S. Kim and S. Y. Oh, "A Study on Mobile Contents Converting Design of Web Engineering", Journal of Korea Information Processing Society, vol. 12, no. 129, **(2005).**
[20] Y. J. Lee, "A Method of C Language based Solution Transformation between WIPI and BREW Platform", Master's Thesis, Chungnam National University, **(2007).**
[21] C. U. Hong, J. H. Jo, H. H. Jo, D. G. Hong and Y. S. Lee, "GVM-to-BREW Translator System for Automatic Translation of Mobile Game Contents", Game Journal of Korea Information Processing Society, vol. 2, no. 49, **(2005).**
[22] Y. S. Lee, "Design and Implementation of the MSIL-to-Bytecode Translator to Execute .NET Programs in JVM platform", Journal of Korea Multimedia Society, vol. 7, no. 976, **(2004).**
[23] Y. S. Lee and S. W. Na, "Java Bytecode-to-.NET MSIL Translator for Construction of Platform Independent Information Systems", LNAI, Springer, vol. 3215, no. 726, **(2004).**

Authors

JaeHyun Kim, he received the B.S. degree from the Dept. of Mathematics, Hanyang University, Seoul, Korea, in 1986, and M.S. and Ph.D. degrees from Dept. of Statistics, Dongguk University, Seoul, Korea in 1989 and 1996, respectively. He was a chairman of Dept. of Internet Information 2002-2007. Currently, he is a member of the Korean Data and Information Science Society and a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include mobile programming, cloud system and data analysis.

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.