International Journal of Smart Home
Vol. 10, No. 1, (2016), pp. 265-278
http://dx.doi.org/10.14257/ijsh.2016.10.1.24

A Study on the iOS-to-Tizen Smart Game Converter using Resource Converter and Platform Mapping Engine

Jaehyun Kim[1] and Yangsun Lee[1*]

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

Abstract

The iOS platform developed by Apple is the world's most advanced mobile operating system, continually redefining what people can do with a mobile device. Tizen platform developed by Samsung is an open source smart phone platform, which is created for wide range of device. Due to the use of different smart phone platforms, mobile contents 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 iOS-to-Tizen smart game converter was designed to automatically translate game contents from the iOS platform to the Tizen platform for smart phones. Through the iOS-to-Tizen 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, sound, and other functions of converted Tizen games were equivalent to those of the iOS games before conversion.

pKeywords: *iOS-to-Tizen Converter, Automatic Smart Game Converter, iOS, Tizen, Content Analyzer, Resource Converter, Platform Mapping Engine*

1. Introduction

The iOS platform developed by Apple is the world's most advanced mobile operating system, continually redefining what people can do with a mobile device [1-3]. Tizen platform developed by Samsung is an open source smart phone platform, which is created for wide range of device [4]. Due to the use of different mobile platforms such as Android, iOS, and Windows Phone 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[5-17].

In this paper, to resolve this problem, the iOS-to-Tizen automatic smart game converter system was designed to automatically translate game contents from the iOS platform to the Tizen platform for smart phones. The iOS-to-Tizen converter consists of a content analyzer, resource converter, and platform mapping engine. The content analyzer 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 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 is a system which provides API functions which allow the

* Corresponding Author

ISSN: 1975-4094 IJSH

previous platform's execution environment to be recreated using the target platform's wrapper functions [7-17].

By automatically converting the existing mobile game contents to the smart phone game contents, the existing game contents can be transplanted quickly to a different platform. As a result, the reusability will be increased, and the labor, time, and cost will be reduced.

2. Related Studies

2.1. iOS

The iOS platform developed by Apple is the world's most advanced mobile operating system, continually redefining what people can do with a mobile device. Together, the iOS SDK and Xcode IDE make it easy for developers to create. Derived from core OS X technologies, the amazing user experience of iOS has been streamlined to take maximum advantage of iPhone, iPad, and iPod touch hardware. Technologies shared between iOS and OS X includes the OS X kernel, BSD sockets for networking, and Objective-C and C/C++ compilers for native performance.

The iOS delivers a wide-range of graphics capabilities, such as comprehensive 2D drawing, accelerated 3D rendering, and direct access to video playback and capture. Using high-level frameworks, you can create gorgeous animations and transitions within your app's UI. Figure 1 shows a system configuration of the iOS platform [1-3].

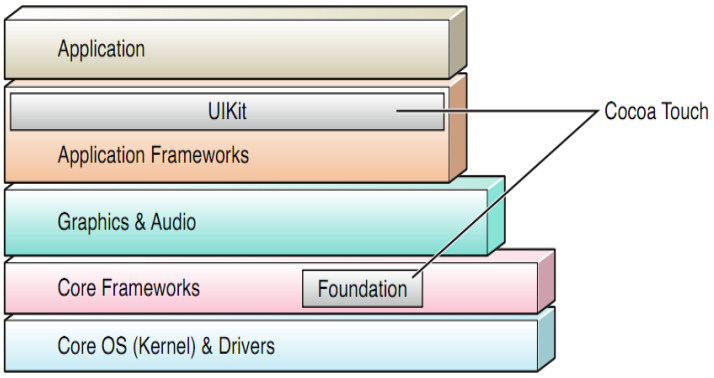


Figure 1. The iOS Platform's System Configuration

The iOS platform's bottom layer is the Core OS, which is the foundation of the operating system. It is in charge of memory management, the file system, networking, and other OS tasks, and it interacts directly with the hardware. The Core Frameworks(Services) layer provides an abstraction over the services provided in the Core OS layer. It provides fundamental access to iPhone OS services. The Graphics and Audio(Media) layer provides multimedia services that you can use in your iPhone and iPad applications. The Cocoa Touch layer provides an abstraction layer to expose the various libraries for programming the iPhone and iPad.

2.2. Tizen

Tizen platform developed by Samsung is an open source smart phone platform, which is created for wide range of device. To make the Tizen platform smarter, they added exciting features such as multipoint-touch, 3D graphics, an enhanced UI, and support web standards such as HTML5, Java Script, and CSS. Figure 2 depicts the Tizen platform's hierarchical structure and components.

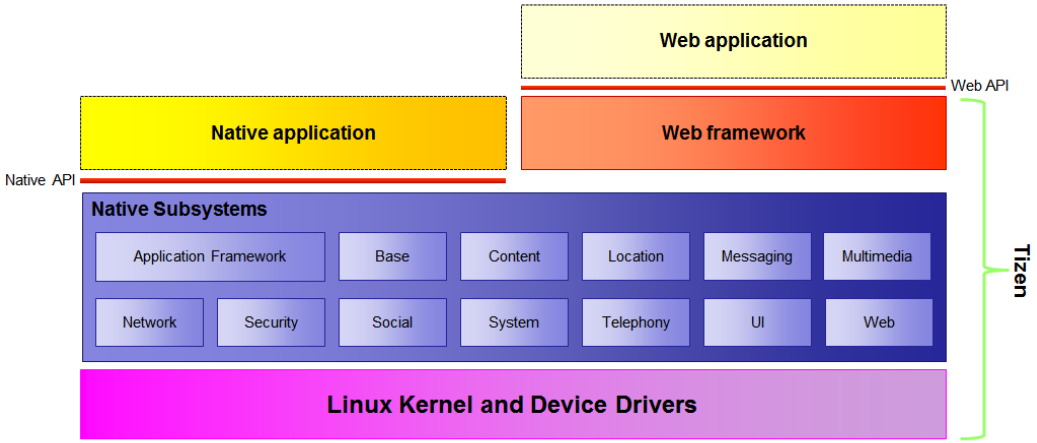


Figure 2. Tizen Platform's System Configuration

The Tizen platform consists of a kernel, a native framework, and a web framework. The kernel layer contains the Linux kernel and device drivers. The native framework is composed of system services and a set of native modules across various domains, with which native applications can be developed. The modules include, for example, Base, Application framework, Security, UI, Network, Messaging, Social, Locations, and Web. The framework also provides popular standard open source libraries, such as eglibc, libstdc++, libxml2, OpenGL® ES, OpenAL, and OpenMP® to support efficient application development and the migration of pre-existing applications using such libraries. The Web framework accommodates and leverages most up-to-date Web technologies. It provides a large number of HTML5 functionalities defined by W3C and other standardization groups, such as video, audio, form, 2D canvas, WebGL, CSS3, geolocation, vibration, Web socket, and Web worker. In addition, the framework defines various new device APIs, which enable you to access device functionalities, such as Bluetooth, near field communication (NFC), alarm, and messaging. The device functionalities are provided with a strict rule-based security control system that restricts the malicious use of the device APIs [4].

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 mobile game content converter using XML has attempted to convert Java content [18-21]. 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 [22] and the conversion GVM C into BREW C [23] 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 [24-25] 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 [5]. 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 and iOS[6-14], for example, the WIPI-to-iOS converter, WIPI-to-Android converter, GNEX-to-iOS converter, GNEX-to-Android converter, Android-to-iOS converter, and iOS-to- Android converter system.

3. The iOS-to- Tizen Smart Game Converter

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

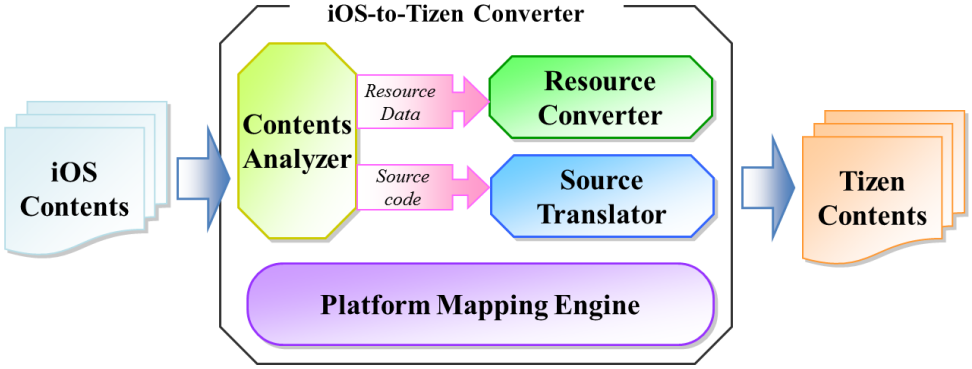


Figure 3. The iOS-to- Tizen Smart Game Converter System

For automatic conversion on the source level, first the source code must be converted into source code for the subject platform that executes the same action. Other data such as images and sound must also be converted into a form that can be used on the new target platform. In addition, an API library must be provided in order to maintain equivalent programming and event environments [7-14].

3.1 Content Analyzer

The content analyzer [5-9, 11] is a system that analyzes the input content, 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, indicating whether it is, for example, actual image or sound resource data, a map tile, or user data. The remaining source code, excluding the resource components, is delivered to the source translator so that it can be automatically translated into the target platform's source language.

The iOS content analyzer receives the iOS content as an input and analyzes it so that it can be easily converted into content for the Tizen platform. Then it divides the files into source files, resource files, and other files. Figure 4 shows a model of the iOS game content analyzer.

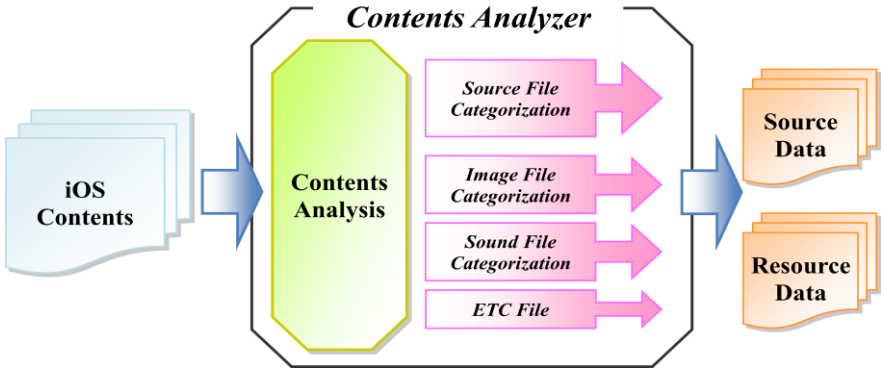


Figure 4. The iOS Game Content Analyzer

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.

3.2. Resource Converter

The resource converter [5-9, 11] 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.

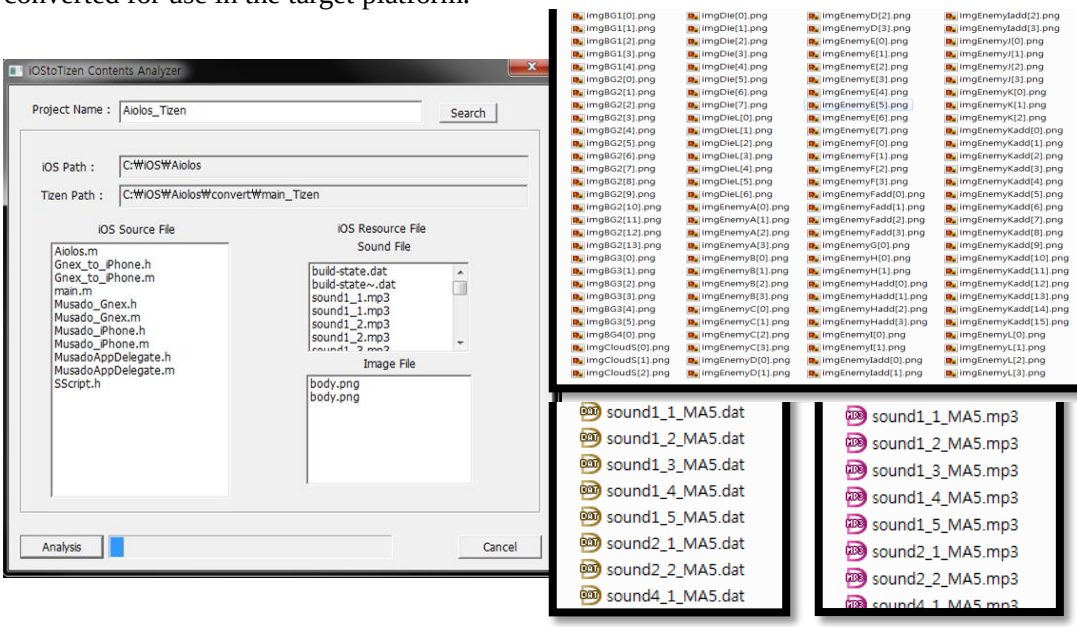


Figure 5. Result of the iOS Game Content Analyzer

Thus, the image file formats provided in existing smart platforms such as BMP, PNG, and JPEG and sound file formats such as WAV and MP3 can all be used in the Android and iOS smart phone platforms. Therefore, they can be produced without any conversion of text-form binary files or sound file formats. Consequently, only a management file must be created to show which resources have been used. In addition, if the original formats have been used, the format provided is maintained, and the text form is converted into the relevant platform's format and enabled for use by the API function providing the functionality. User data in text form is also made compatible with the relevant platform's

file system by producing binary file formats. Figure 6 shows a model of the iOS-to-Tizen resource converter system.

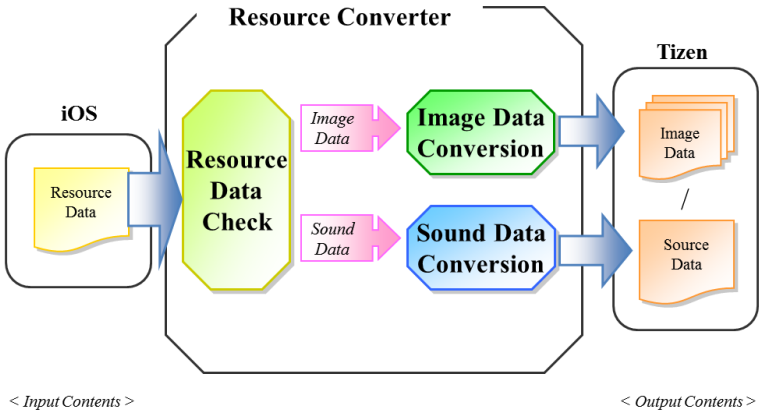


Figure 6. Resource Converter System

After converting the resources, the resource converter creates all files related to the Tizen project to enable a direct approach from the Tizen software development kit, and the converted resources are applied to the project. Figure 7 shows the result of the resource converter.

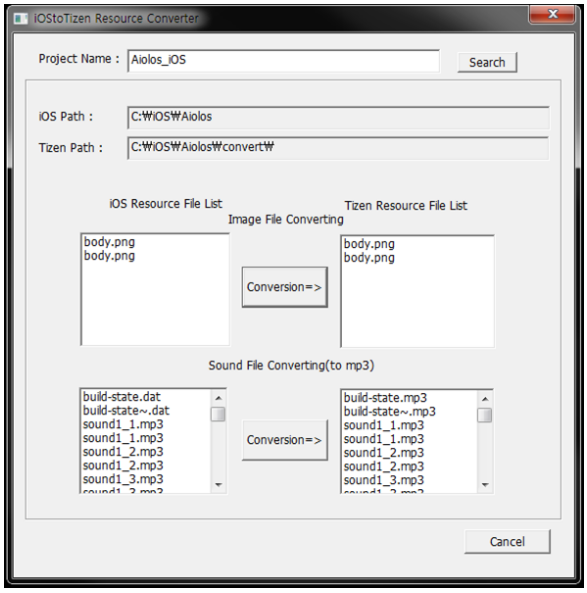


Figure 7. Result of the iOS-to-Tizen Resource Converter

3.3. Platform Mapping Engine

The platform mapping engine [5-9] creates an execution environment on the Tizen platform that is identical to that in the iOS platform so that the same environments can be executed identically. Thus, iOS is enabled to run in its original form on the Tizen platform. On the basis of the created execution environment, a wrapper function format is provided that enables identical execution of the iOS's API on the Tizen platform.

The only function of the source translator is to automatically translate the iOS platform's source program into the Tizen platform's source program. However, the translated program cannot be run on the Tizen platform right away. The iOS'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 Tizen platform. Figure 8 shows the structure of the iOS-to-Tizen platform mapping engine, in which API and runtime environment from iOS can be used in Tizen platform through wrapper functions.

3.3.1 System Environments

The iOS program's life cycle is divided into five steps: Not running, Inactive, Active, Background and Suspended. Each step is a step in the application's life cycle and each time a change is made, didFinishLaunchingWithOptions:, DidBecomeActive: DidEnterBackground:, WillEnterForeground: WillTerminate: 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 iOS platform system.

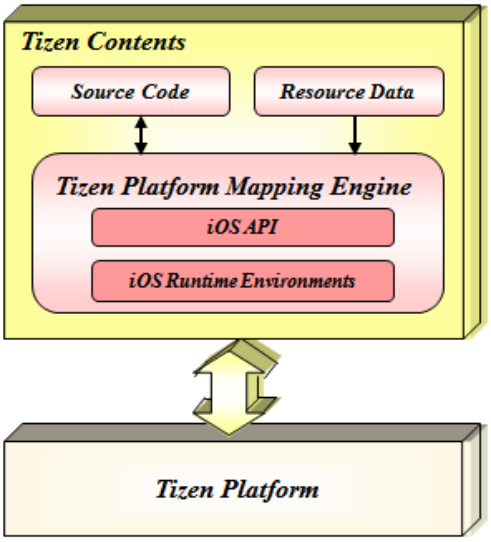


Figure 8. Platform Mapping Engine Model

In Tizen, slightly different to iOS, it has five life cycle steps: Initialzing, Activated, Deactivated, Minimized and Terminating. Each time a stage is changed, OnInitializing, OnFrameActivated, OnFrameDeactiveted, OnFrameMinimized, OnFrameRestored, OnTerminating methods are called respectively. In this thesis, these points were put into consideration and the iOS platform's system environment was matched one on one to the Tizen platform's system environment. For example, iOS's DidBecomeActive and Tizen's OnFrameActivated have a similar role. Therefore, when DidBecomeActive was called for, OnFrameActivated method was called. Figure 9 shows an iOS life cycle mapped to the Tizen life cycle.

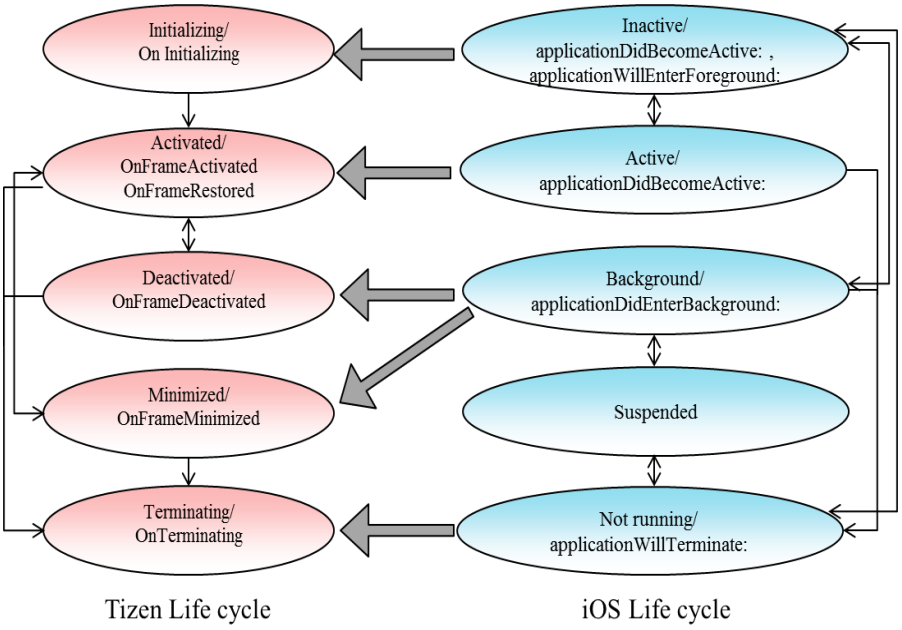


Figure 9. System Mapping Model between iOS and Tizen

3.3.2. Events

The platform mapping engine converts an event which occurred in iOS into a Tizen event form. The rising event is delivered to an event handler defined in the translated source code in order to be processed. The event handler calls for a defined API in the platform mapping engine when each event occurs.

The iOS touch event calls for dispatchFirstTouchAtPoint method when a touch occurs and recognizes the touched section and in order to recognize multiple touches, it calls touchesBegan method. There is a defined type for each event and each time an event occurs, it is automatically called for. When a touch is ended, touchesEnded method is called to close the event. In Tizen, when a touch event occurs, a different method is called depending on the touch state. For example, when a touch is recognized, OnTouchPressed method is called and when the touch is finished, OnTouchReleased method is called. The iOS touch event code depending on iOS touch event's Tizen mapping value is entered into each respective source code of Tizen's touch event.

3.3.3. Graphic Environments

The UIApplication class which manages all the programs in iOS is initialized to Window by delegate and the UIView is put on top so that the graphic is implemented on the UIView. The iOS image is output by assigning the image as a Bitmap and the CGContext image is output. At this point, the UIView receives CGContext and draws everything that is displayed on the screen in the drawRect section.

The graphic environment similar to iOS was built for Tizen. In Tizen, the Form class and Canvas class play similar roles to iOS's graphic environment. Form class and Canvas class were used to build a graphic environment that runs in iOS. Tizen has the Canvas class, a drawing tool that is similar to the CGContext class. Figures 10 and 11 show the LCD output methods of iOS and Tizen, respectively.

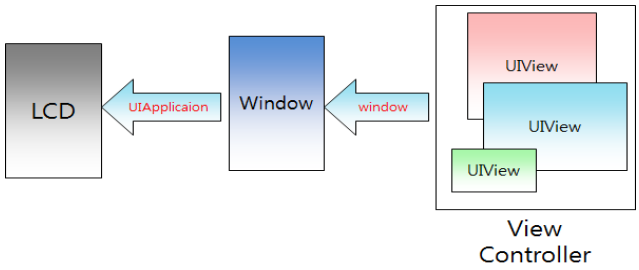


Figure 10. iOS's Graphic Output Method

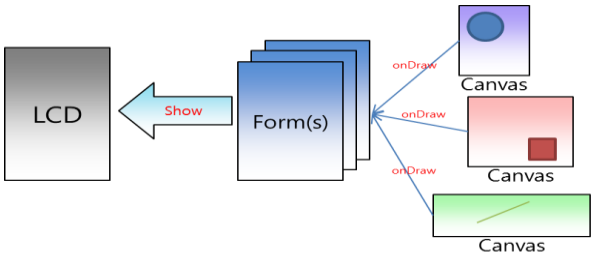


Figure 11. Tizen's Graphic Output Method

3.3.4. Font and Text Print

The iOS uses UIFont class to enter text and it is taken to the graphic environment to be printed. Font style and size are saved in the UIFont class and when the text is printed, the properties given are drawn. The default font style of iOS and Tizen are Plain, Bold and Italic. Table 1 shows methods for setting the font styles of iOS and Tizen. These structures were used to implement the UIFont of Tizen mapping engine.

Table 1. The Font Styles in iOS and Tizen

Font Style	iOS	Tizen
Pain (Normal)	(UIFont *)systemFontOfSize: (CGFloat)fontSize	Font.Construct(FONT_STYLE_PLAIN, int Size);
Bold	(UIFont *)boldSystemFontOfSize: (CGFloat)fontSize	Font.Construct(FONT_STYLE_BOLD, int Size);
Italic	(UIFont *)italicSystemFontOfSize: (CGFloat)fontSize	Font.Construct(FONT_STYLE_ITALIC, int Size);

3.3.5. Images and Sounds

The iOS image is output by assigning the image as a UIImage class and it is output directly or converted into a CGImage form and output as a CGContext image. In Tizen, the Image class was used to call the image and it was assigned the Bitmap class and output as a Canvas.

In iOS, sound is output by the sound file being executed by the AVAudioPlayer class. In Tizen, the sound file is executed by the Player class. The Play method plays a sound file, and the stop method makes a stop. These structures were used to map image and sound of iOS to Tizen.

Table 2. Image Methods in iOS and Tizen

Operation	iOS	Tizen
Load Image	(UIImage *)imageNamed: (NSString *)path	(Bitmap *) (Tizen::App::AppResource)-> GetBitmapN ((String *) imagePath)
	CGImageRef CGImageRetain ((UIImage *).CGImage)	
Draw Image	void CGContextDrawImage (CGContextRef c, CGRect rect, CGImageRef image)	(Canvas *)->DrawBitmap((Tizen::Graphics::Rectangle), (Bitmap *))

Table 3. Sound Methods in iOS and Tizen

Operation	iOS	Tizen
Load Sound	(id)initWithContentsOfURL:(NSURL *) *url* fileTypeHint:(NSString *)*utiString* error:(NSError **)*outError*	(Player *)->OpenFile((String *) mediaLocalPath)
Play Sound	[(AVAudioPlayer *) play]	(Player *)->Play();
Stop Sound	[(AVAudioPlayer *) stop]	(Player *)->Stop();

3.3.6 Library functions(APIs)

Tizen's API has been implemented to use the same name of iOS API in Tizen's platform in order to functions exactly the same. Tizen's classes and methods are provided through the Namespace Library and if there are classes or methods required, it uses the Namesapce Library. The translated Tizen source code appears the same as iOS API function by the wrapper function. Therefore, it can be used in the same form as iOS which allows to keep the original form of the source and can be executed in Tizen platform without further modifications. Table 4 shows an API mapping table supported in iOS and Tizen.

Table 4. Supported API Mapping Table between iOS and Tizen

class	iOS API	Tizen API
System(3)	GetDate, GetTime, Exit	GetDate, GetTime, Exit
Handset Control(11)	PlaySound, StopSound, SetVolume, StartVib, StopVib, SetTimer, SetTimer1, SetTimer2, ResetTimer, ResetTimer1, ResetTimer2	PlaySound, StopSound, SetVolume, StartVib, StopVib, SetBackLight, GetUserNV, GetUserNV, PutUserNV, SetTimer, SetTimer1, SetTimer2, ResetTimer, ResetTimer1, ResetTimer2
String(15)	StrLen, StrCpy, StrSub, StrCat, GetChar, GetCharString, PutChar, AsciiToInt, PutByte, GetByte, PutBytes, GetBytes, MakeStr1, MakeStr2, MakeStr3	StrLen, StrCpy, StrSub, StrCat, GetChar, PutChar, AsciiToInt, PutByte, GetBytes , MakeStr1, MakeStr2, MakeStr3
Graphic(36)	SetAlpha, SetClip, ResetClip, SetActiveBuffer, SetGamma, SetColor, SetFont, SetFontColor, SetFontAlign, SetFontType, SetPalette, SetImageAlpha, Clear, ClearRGB, ClearWhite, ClearBlack, PutPixel, CopyImage, CopyImageDir, CopyImagePal, CopyImageDirPal, CopyImageEx, CopyImageTile, DrawLine, DrawHLine, DrawRect, DrawStr, DrawText, DrawStrSolid, DrawStrSolid2, FillRect, FillRectEx, FillEllipse, SaveLCD, RestoreLCD, CopyLCD, Flush	SetClip, SetClipBounds, SetActiveBuffer SetAlpha, SetGamma, SetColorComponents, SetFont, SetTextColor, SetColorForeground, SetColorBackground, SetVerticalAlignment, SetHorizontalAlignment, SetTextAbbreviationEnabled, SetFontType, DrawText, SetBounds, DrawStrSolid, DrawStrSolid2, FillRectangle, DrawBitmap, DrawLine, Flush.Start
Mathermatics (8)	Abs, Rand, RandRatio, Sin100, Cos100, ArrayToVar, ArrayToArray, HitCheck	Abs, Rand, RandRatio, Sin100, Cos100, ArrayToVar, ArrayToArray, HitCheck

4. Experimental Results

In this paper, the iOS-to-Tizen smart game converter was used to automatically convert smart game contents from the iOS platform to the Tizen platform in source form. The results of the conversion were then compared. The emulators that were used to execute the content are the iOS 4.3 emulator and the Tizen 2.2 emulator.

Table 5. Platform and Emulator

Platform	Emulator	Method
iOS	iOS 4.3 Emulator	Native
Tizen	Tizen 2.2 Emulator	Native

Figure 12 and Figure 13 compare the execution of the games such as game "Aiolos" and "Elemental Force".

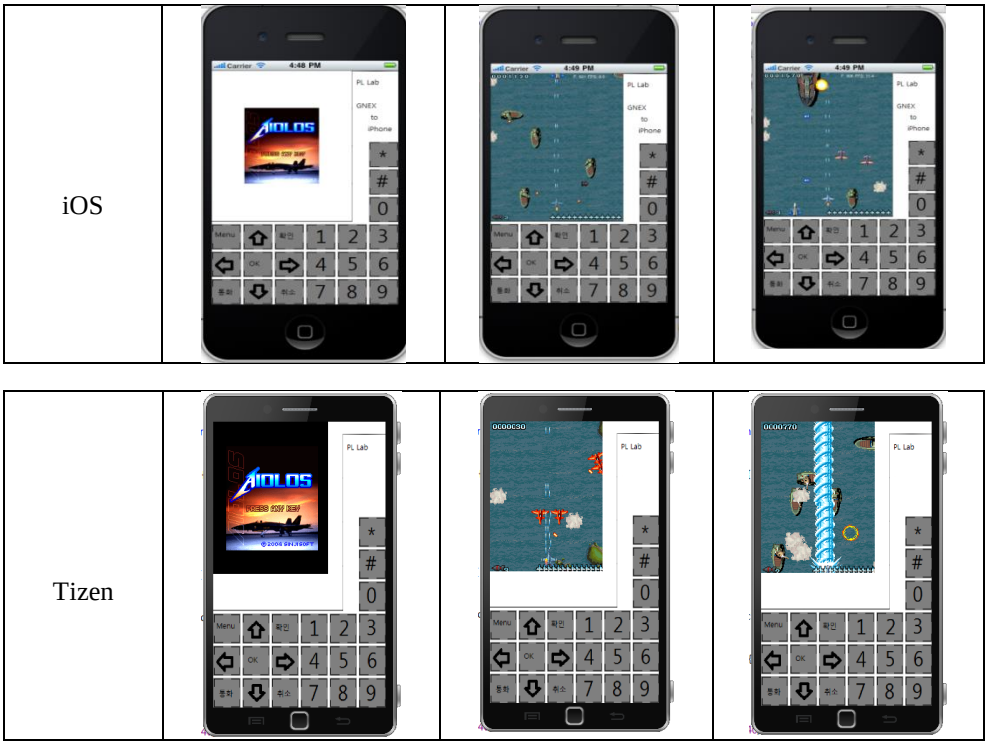
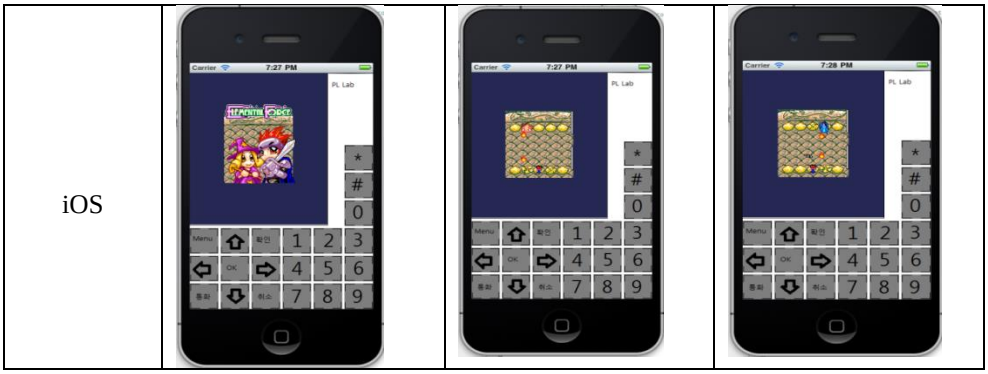


Figure 12. Execution Results of the Game "Aiolos"



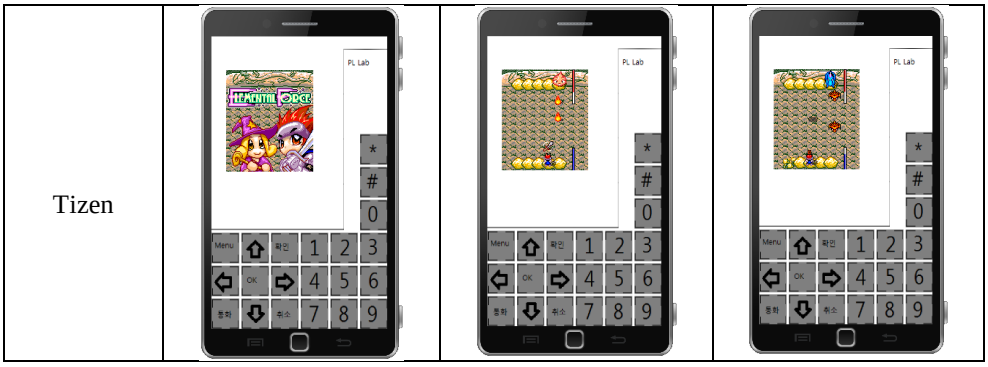


Figure 13. Execution Results of the Game "Elemental Force"

5. Conclusions and Further Researches

With the recent appearance of smart phones, the mobile game market is experiencing high growth rates each year, and game content has become killer content in the mobile market. However, differences in smart platforms have required repeated development or conversion of smart game content for use on multiple platforms.

The automatic smart game converter iOS-to-Tizen presented in this paper offers a means to solve the problems of different smart platforms. It can ensure quick and automatic conversion of existing iOS game content into game content for the Tizen 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 in order to provide game content designed for the iOS platform on the Tizen platform can be significantly reduced. Consequently, productivity can be enhanced, and the time and expense thus saved can be invested in developing new game content. is expected to accelerate the development of high-quality mobile games and create a basis for increasing the productivity of the mobile industry.

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 Tizen.

Acknowledgments

This Research was supported by Seokyeong University in 2014.

References

[1] Apple, iOS Technology Overview, http://developer.apple.com/devcenter/ios
[2] V. Nahavandipoor, iOS 5 Programming Cookbook : Solutions & Examples for iPhone, iPpad, and iPod Touch Apps, O Reilly Media **(2012)**
[3] J. Nozzi, Mastering Xcode 4 : Develop and Design, Peachpit Press **(2011)**
[4] Samsung, Tizen, https://www.tizen.org/, https://developer.tizen.org/development/
[5] Y. S. Lee, "Design and Implementation of the GNEX C-to-WIPI Java Converter for Automatic Mobile Contents Translation," Journal of Korea Multimedia Society, 13, 609 **(2010)**
[6] Y. S. Son, S. M. Oh, 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, 13, 1051 **(2010)**
[7] Y. S. Lee, H. J. Choi, J. S. Kim, "Design and Implementation of the GNEX-to-iPhone Converter for Smart Phone Game Contents, Journal of Korea Multimedia Society, 14, 577 **(2011)**

[8] Y. S. Lee, Y. S. Son, "A Platform Mapping Engine for the WIPI-to-Windows Mobile Contents Converter," CCIS, Springer, 262, 69 (2011)
[9] Y. S. Lee, "Automatic Mobile Contents Converter for Smart Phone Platforms," Journal of Korea Multimedia Society, 15, 54 (2011)
[10] Y. S. Son, Y. S. Lee, "The iOS-to-Android Contents Translator for Mobile Game Contents Re-Usability," The Asian International Journal of Life Sciences, Asia Life Sciences, 11, 613-624 (2015)
[11] J. H. Kim, Y. S. Lee, "A Study on the Android-to-iOS Smart Game Content Converter," International Journal of Software Engineering and Its Applications, SERSC, 8, 1 (2014)
[12] Y. S. Son, Y. S. Lee, "A Study on the WIPI-to-iOS Converter using Resource Converter and Platform Mapping Engine," International Journal of Applied Engineering Research, Research India Publications, 9, 29709 (2014)
[13] Y. S. Lee, Y. S. Son, "Design and implementation of the WIPI-to-Android Automatic Mobile Game Converter for the Contents Compatibility in the Heterogeneous Mobile OS," Journal of Systems Architecture, Elsevier, 60, 693 (2014)
[14] 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, 16, 739 (2013)
[15] 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, 14, 681 (2011)
[16] Y. S. Lee, Y. S. Son, "A Study on the Source Translator for Generating the Android Game Source from the WIPI Game Source," International Journal of Multimedia and Ubiquitous Engineering, SERSC, 7, 95 (2012)
[17] Y. S. Son, Y. S. Lee, "Automatic UI Generation Technique for Mobile Applications on Touch-Screen based Smart Phones," International Journal of Smart Home, SERSC, 6, 67 (2012)
[18] S. H. Kim, Design and Implementation of A Mobile Contents Conversion System based on XML using J2ME MIDP, Master's Thesis, Hannam University (2003)
[19] Y. S. Kim,D. C. Jang, "A Design for Mobile Contents Converting Using XML Parser Extraction," Journal of Korea Multimedia Society, 6, 267 (2003)
[20] S. I. Yun, Integrated Conversion System for Wired and Wireless Platform based on Mobile Environment, Ph.D Thesis, Hannam University (2003)
[21] Y. S. Kim, S. Y. Oh, "A Study on Mobile Contents Converting Design of Web Engineering," Journal of Korea Information Processing Society, 12, 129 (2005)
[22] Y. J. Lee, A Method of C Language based Solution Transformation between WIPI and BREW Platform, Master's Thesis, Chungnam National University (2007)
[23] C. U. Hong, J. H. Jo,H. H. Jo, D. G. Hong, Y. S. Lee, "GVM-to-BREW Translator System for Automatic Translation of Mobile Game Contents," Game Journal of Korea Information Processing Society, 2, 49 (2005)
[24] Y. S. Lee, "Design and Implementation of the MSIL-to-Bytecode Translator to Execute .NET Programs in JVM platform," Journal of Korea Multimedia Society, 7, 976 (2004)
[25] Y. S. Lee, S. W. Na, "Java Bytecode-to-.NET MSIL Translator for Construction of Platform Independent Information Systems," LNAI, Springer, 3215, 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 & 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 2004-2005, a General Director of Korea Multimedia Society from 2005-2006, a Vice President of Korea Multimedia Society in 2009, and a Senior Vice President of Korea Multimedia Society in 2015. Also, he was a Director of Korea Information Processing Society from 2006-2014 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 HSST from 2014-2015. Currently, he is a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include smart system solutions, programming languages, and embedded systems.