Journal of Korea Multimedia Society Vol. 16, No. 2, February 2013(pp. 190-197)
http://dx.doi.org/10.9717/kmms.2013.16.2.190

스마트 크로스 플랫폼을 위한 스마트 가상기계의 설계 및 구현

한성민[†], 손윤식[††], 이양선[†††]

요 약

최근 국내외 플랫폼 업체와 이동통신사들이 서로 다른 스마트 플랫폼을 채택하여 사용함으로 인해 개발자는 하나의 스마트 콘텐츠를 서비스하기 위하여 각각의 스마트 플랫폼 특성에 맞추어 콘텐츠를 개발하거나 변환 작업을 해야 한다. 하지만, 기존의 스마트 콘텐츠를 다른 스마트 플랫폼으로 이식하기 위한 변환 작업에 많은 시간과 비용이 소모되고 있다. 이런 이유로 최근에 개발 언어의 제약 없이 한번 프로그램을 작성하면 어떤 플랫폼에서도 실행할 수 있는 OSMU(One Source Multi Use)의 핵심기술인 스마트 크로스 플랫폼(Smart Cross Platform) 또는 하이브리드 플랫폼(Hybrid Platform)에 관한 관심이 높아져 폰갭(PhoneGap), HTML5를 기반으로 한 센차터치(Sencha Touch)와 같은 시스템이 소개되고 있다. 본 논문에서는 플랫폼에 의존적인 기존의 Android나 iOS, Windows Phone과 달리 스마트 기기에 탑재되어 플랫폼에 독립적으로 응용 프로그램을 다운로드하여 실행할 수 있는 스마트 크로스 플랫폼 기반의 스마트 가상기계(Smart Virtual Machine)를 개발하였다. 스마트 가상기계는 Java 언어를 사용하는 썬 마이크로시스템스사의 JVM이나 C/C++/C# 언어를 사용하는 마이크로소프트사의 .NET 프레임워크와 같은 기존 기술들과 차별적으로 C/C++와 Java 언어를 모두 지원하여 콘텐츠 개발자들로 하여금 개발 언어 선택의 제한 없이 스마트 콘텐츠를 개발 할 수 있는 환경을 제공하여 준다.

Design and Implementation of the Smart Virtual Machine for Smart Cross Platform

Seong-Min Han[†], Yun-Sik Son[††], Yang-Sun Lee[†††]

ABSTRACT

Since domestic and foreign platform companies and mobile carriers adopt and use different kinds of smart platforms, developers should develop or convert contents according to each smart platform to provide a single smart content for customers. It takes long time and a lot of money to convert the conventional smart contents in order to serve other smart platforms. For the reason, more attention has been paid on Smart Cross Platform or Hybrid Platform, the core technologies of OSMU(One Source Multi Use) in which, once a program is coded, it can be executed in any platforms regardless of development languages. As a result, PhoneGap and HTML5 based Sencha Touch have been introduced. In this paper, we developed the smart virtual machine, which is built in smart cross platform based smart devices, unlike Android, iOS, Windows Phone devices being dependent of platforms, and helps to download and execute applications, being independent of platforms. the smart virtual machine supports C/C++, and Java language, being differentiated from JVM by sun microsystems that supports only Java language and .NET framework by microsoft that supports only C, C++ and C#. Therefore, it provides contents developers with the environment where they can get a wide range of options in choosing a language and develop smart contents.

Key words: Smart Platform(스마트 플랫폼), Smart Cross Platform(스마트 크로스 플랫폼), Smart Virtual Machine(스마트 가상기계), OSMU(One Source Multi Use)

※ 교신저자(Corresponding Author) : 이양선, 주소 : 서울시 성북구 서경로 124 서경대학교 컴퓨터공학과, 전화 : 02) 940-7743, E-mail : yslee@skuniv.ac.kr
접수일 : 2013년 1월 10일, 수정일 : 2013년 1월 31일
완료일 : 2013년 2월 1일
[†] 준회원, 서경대학교 컴퓨터공학과 대학원생
(E-mail : sungals@skuniv.ac.kr)
[††] 정회원, 동국대학교 컴퓨터공학과 전문연구원
(E-mail : sonbug@dongguk.edu)
[†††] 종신회원, 서경대학교 컴퓨터공학과 교수
※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (No.20110006884).

1. 서 론

최근 스마트폰은 기본적인 휴대폰 기능에 컴퓨터와 같은 기능이 포함된 차세대 핸드폰으로 시장의 각광을 받으며 널리 확산되고 있는 실정이다. 스마트폰의 플랫폼으로는 해외에선 Google이 Java를 사용할 수 있는 Android 플랫폼을, Apple이 C, Objective-C 언어를 사용할 수 있는 iOS(iPhone)를, 마이크로소프트가 C/C++, C#, Java 등을 사용할 수 있는 Windows Phone 플랫폼을, 그리고 노키아가 C/C++, Java 등을 사용할 수 있는 Symbian 플랫폼을 발표하였고, 국내에서는 삼성이 Bada 플랫폼을 발표하였다[1-4].

현재 스마트 플랫폼 분야에서는 C 언어와 Java 언어 등을 모두 수용할 수 있는 통합 플랫폼과 가상기계의 개발은 거의 전무하고, 컴파일러와 번역기 기술에 대한 연구가 프로그래밍 언어나 목적기계 플랫폼에 의존적이어서 프로그래머가 프로그램을 한번 작성하면 운영체제나 아키텍처와 같은 플랫폼에 의존하지 않고 어느 시스템에서나 실행할 수 있는 기술에 대한 연구는 매우 미흡한 실정이다.

스마트 가상기계(Smart Virtual Machine)는 스마트 플랫폼 환경의 스마트 기기에 탑재되어 스마트 콘텐츠를 실행하는 소프트웨어로, 순차적 언어인 C 언어와 객체지향 언어인 C++, Java 언어를 모두 수용함으로써 콘텐츠 개발자로 하여금 개발 언어의 제약 없이 스마트 콘텐츠를 개발할 수 있는 환경을 제공한다. 또한, 하나의 언어로 개발된 스마트 콘텐츠를 스마트 가상기계가 탑재된 스마트 기기에서 플랫폼 독립적으로 실행하게 해 줌으로써 스마트 콘텐츠를 플랫폼마다 개발해야하는 시간과 노력을 줄이고, 하나의 스마트 콘텐츠를 스마트 가상기계가 탑재된 여러 플랫폼에서 실행할 수 있으므로 스마트 콘텐츠의 OSMU(One Source Multi Use)를 구현할 수 있다[5-11].

2. 관련연구

2.1 JVM

JVM(Java Virtual Machine)은 1991년 썬 마이크로시스템즈사의 제임스 고슬링 등의 연구진들에 의해 개발된 가상기계로, 현재 가장 널리 사용되고 있다. 그림 1은 JVM의 구조를 나타낸 것이다.

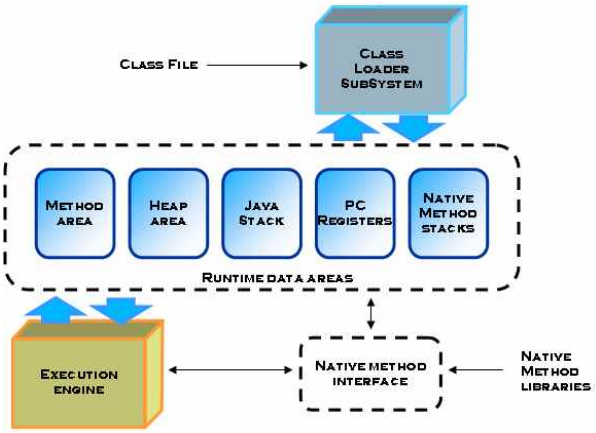

그림 1. JVM의 구조

JVM은 클래스 로더와 실행시간 데이터 영역, 실행 엔진, 네이티브 메서드 인터페이스로 구성되어있다. 클래스 로더는 JVM에서 실행 가능한 중간코드인 클래스파일을 입력으로 받아 로딩 후, 실행 엔진에서 실행시간 데이터 영역을 사용하여 명령어들을 실행한다. 네이티브 메서드 인터페이스는 JVM의 명령어들이 아닌 다른 언어를 사용하여 구현된 네이티브 메서드 스택를 이용하여 실행 엔진에서 실행할 수 있게 해준다[12-16].

2.2 .NET 플랫폼

.NET 프레임워크는 미국의 마이크로소프트사에 의해 개발된 응용 프로그램을 작성하고 실행하기 위한 플랫폼이다. 프레임워크의 핵심은 CLR(Common Language Runtime)과 .NET 프레임워크 클래스 라이브러리(FCL)에 있다. 그림 2는 .NET 프레임 워크의 구조를 나타낸 것이다.

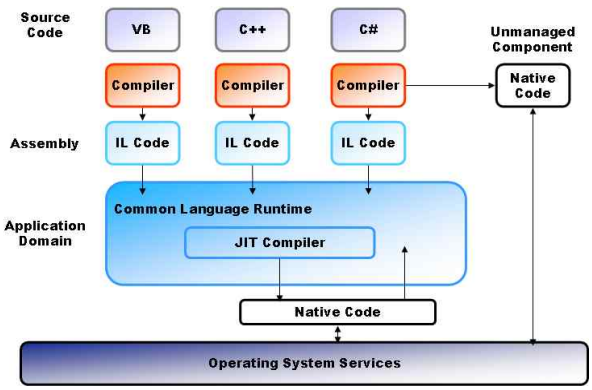

그림 2. .NET 프레임 워크의 구조

.NET IL은 .NET 환경에서 언어 통합을 위해 설계된 중간 언어이며 MSIL(MicroSoft Intermediate Language)이라고도 부른다. .NET IL은 하드웨어 및 플랫폼에 독립적이고 실행은 JIT(Just-In Time) 방법을 사용한다.

.NET 플랫폼에서는 C#, Managed C++, VB.NET 등 다양한 프로그래밍 언어를 수용할 수 있다. 이와 같은 프로그래밍 언어로 작성된 소스 코드는 각 언어에 해당하는 컴파일러에 의해 PE(Portable/ Executable) 파일 형태로 번역된다. PE 파일은 .NET 플랫폼에서 사용되는 파일 포맷으로 .NET 플랫폼의 가상기계에 해당하는 CLR에 의해 실행된다[17-20].

3. 스마트 가상기계의 설계 및 구현

3.1 스마트 크로스 플랫폼과 스마트 가상기계

스마트 크로스 플랫폼(Smart Cross Platform)은 본 연구팀이 개발한 스마트 기기를 위한 가상기계 기반의 플랫폼이다. 스마트 크로스 플랫폼은 중간 언어를 사용하는 가상기계 기반의 플랫폼으로 여러 스마트기기에 탑재되어 독립적으로 수행이 가능하며, 중간언어인 SAF(Smart Assembly Format)는 순차적 언어인 C와 객체지향 언어인 C++과 Java를 모두 수용한다. 그림 3은 스마트 크로스 플랫폼의 구조를 나타낸 것이다.

스마트 크로스 플랫폼은 응용 프로그램을 컴파일하여 중간코드인 SIL(Smart Intermediate Language) 코드를 생성하는 컴파일러와 중간코드인 SAF 파일을 입력으로 받아 실행 파일인 SEF(Smart Executable Format) 파일로 변환하는 어셈블러, 그리고 실행파일인 SEF 파일을 입력으로 받아 실행하여 결과를 출력하는 가상기계의 세 부분으로 구성된다. 스마트 가상기계 시스템은 계층적인 구조로 설계

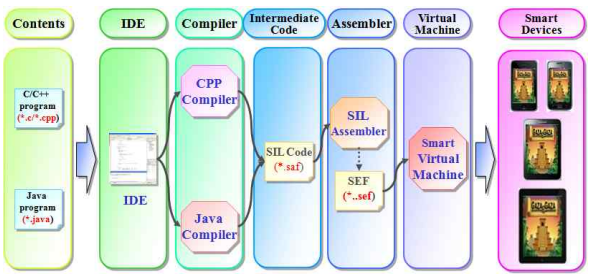

그림 3. 스마트 크로스 플랫폼의 구조

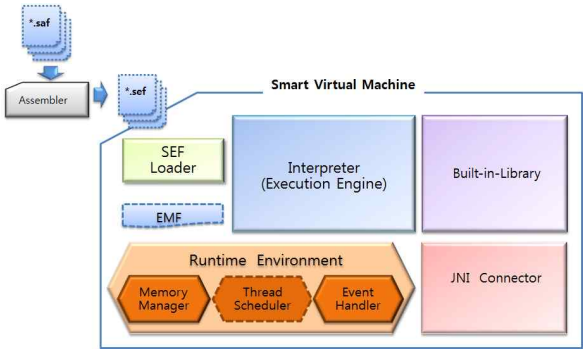

그림 4. 스마트 가상기계(SVM)의 구조

되어 다른 디바이스와 운영환경으로의 재목적 과정에서 발생하는 부담을 최소화하도록 설계되었다[5-11, 21-27].

스마트 가상기계(Smart Virtual Machine)는 스마트 콘텐츠를 입력으로 받아 컴파일러와 어셈블러가 생성한 실행파일을 입력으로 받아 콘텐츠를 실행하고 결과를 출력한다. 그림 4는 스마트 가상기계의 구조를 나타낸 것이다. 스마트 가상기계는 그림 4와 같이 실행파일(SEF) 로더, 인터프리터, 내장라이브러리, 실행환경, JNI 커넥터로 구성되며 실행환경은 메모리 관리자, 스레드 스케줄러, 이벤트 핸들러로 구성된다[5-11].

3.2 실행파일(SEF) 로더

SEF(Smart Executable Format)는 실행파일 포맷으로 스마트 가상기계에서 사용하는 파일 포맷이다. 실행파일(SEF) 로더는 어셈블러에 의해 생성된 SEF 파일을 입력으로 받아 스마트 가상기계에서 콘텐츠를 실행하는데 필요한 정보들을 메모리 형태로 변환한 EMF(Executable Memory Format) 파일을 생성한다.

EMF 파일의 헤더 영역에는 코드와 리터럴 값이 적재된 RO 영역의 크기와 초기화 값을 가진 변수인 RW 영역의 크기, 초기화 값이 없는 ZI 영역 크기, 그리고 프로그램 로딩 후 최초 실행되는 함수인 초기화 함수 주소 정보와 그 후에 프로그램이 시작되는 함수의 정보인 엔트리 함수 주소 정보, 그리고 이벤트 발생시 필요한 각종 시스템 함수 주소 정보와 시스템 변수 주소 정보가 들어있다.

코드 영역은 실행파일(SEF) 상의 코드 정보와 코드의 파라미터 정보를 담고 있으며, 상수 풀 영역은

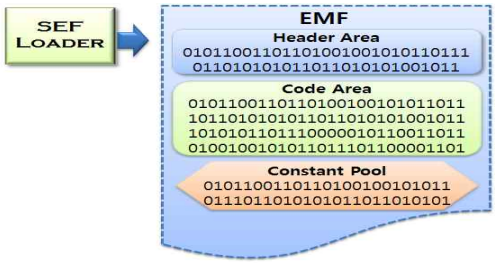

그림 5. EMF의 구조

코드에서 쓰이는 각종 상수 값을 가지고 있는 영역으로써 1바이트 변수의 배열로 구성되어 있다[5-10, 21-23]. 그림 5는 EMF의 구조를 나타낸 것이다.

3.3 인터프리터

인터프리터는 실제로 코드를 연산하는 부분으로 가상기계의 핵심 역활을 수행하는 모듈이다. 그림 6은 인터프리터의 구조를 나타낸 것이다. EMF (Executable Memory Format)에서 명령어 디스패처 (Instruction Dispatcher)를 통해 프로그램 카운터에 해당하는 명령어를 실행 엔진에 보낸다. 실행 엔진은 그 명령어를 실행 환경을 통해 실행하게 되는데, 실행환경에서 가상 메모리 환경인 로컬 영역과 글로벌 영역, 힙 영역에서 연산하며 실행한다.

스마트 가상기계에서 현재 지원하는 명령어는 198개이며, 각 명령어는 스택 명령어, 산술 명령어, 흐름 제어 명령어, 형 변환 명령어, 객체 명령어 등으로 나뉜다. 각 명령어 끝의 x는 자료형의 니모닉이다. 표 1은 각 명령어의 종류와 의미를 나타낸 것이고, 표 2는 명령어의 자료형에 관한 니모닉을 나타낸 것이다[5-10, 21-25].

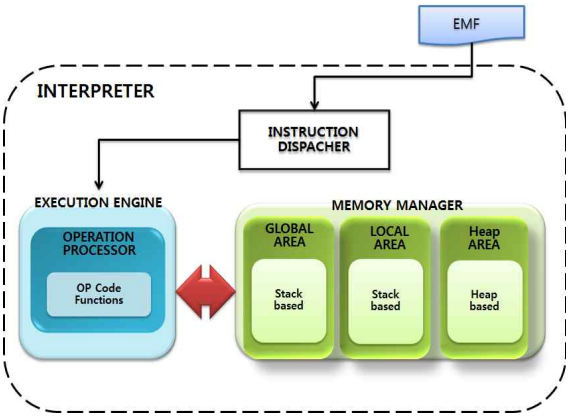

그림 6. 인터프리터의 구조

표 1. 각 명령어의 종류와 의미

명령어	종 류
스택 명령어	pop/push
	swap
	dup
	ldc.x
	lod.x/str.x
	ldi.x/sti.x
	lda
	ldfnc
산술 명령어	add.x, sub.x, mul.x, div.x, mod.x
	eq.x, ne.x,ge.x, gt.x,le.x, lt.x
	and.x, or.x, not.x
	band.x, bor.x, bxor.x, shl.x, shr.x, bcom.x
형 변환 명령어	cvx.y
흐름 제어 명령어	tjp, fjp, ujp

표 2. 명령어의 자료형에 관한 니모닉

니모닉	자료형	니모닉	자료형
i	int	ui	unsigned int
l	long	ul	unsigned long
s	short	us	unsigned short
f	float	d	double
p	pointer	c	char
t	structure		

3.4 라이브러리

스마트 가상기계의 대부분의 내장 라이브러리는 각 운영체제의 라이브러리를 이용하여 래퍼(wrapper) 함수로 구현하였으며 그래픽 라이브러리의 경우 스마트 가상기계의 프로그램 인터페이스의 제공을 위해 운영체제의 라이브러리를 이용하여 스마트 가상기계만의 라이브러리로 변환하여 구현을 하였다.

스마트 가상기계는 현재 ANSI C 라이브러리와 콘텐츠 라이브러리가 있으며 그 중 그래픽 라이브러리, 핸드셋 컨트롤 라이브러리, 스트링 라이브러리, 시스템 라이브러리, 수학 라이브러리 등이 지원되고 있다.

스마트 가상기계는 각 운영체제의 라이브러리를 사용하므로 다른 운영체제에 이식하려면 각 운영체제 환경에 맞게 내장 라이브러리를 수정해야 하는

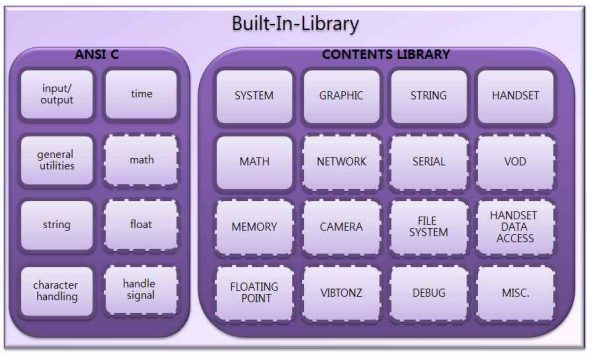

그림 7. 라이브러리 구조

문제점이 있으나 운영체제 마다 서로 다른 라이브러리를 공통적인 인터페이스의 스마트 가상기계 라이브러리로 만들어 줌으로써 콘텐츠 공급자들의 입장에서는 어떤 운영체제에서든지 같은 인터페이스로 라이브러리를 사용할 수 있어 콘텐츠의 수정 없이 다른 플랫폼에서 같은 콘텐츠로 사용이 가능한 환경을 구축할 수 있다[5-10, 21-23].

3.5 실행환경

인터프리터는 어셈블리 명령어를 실행할 때 프로그램의 지원환경인 실행환경의 지원을 받아 실행한다. 먼저, 실행환경의 메모리 관리자에서 각 명령어에 따라 스택의 연산과정이 일어나고, 상수 풀의 내용은 글로벌 영역에, 객체정보는 힙 영역에, 변수정보는 로컬영역에서 액티베이션 레코드, 오퍼레이션 스택, 디스플레이 벡터로 나뉘어 처리된다. 그리고 스케줄러는 스레드 스케줄링을 하여 멀티 스레딩이 가능한 프로그래밍 환경을 만들고, 이벤트 핸들러는 이벤트-구동 방식의 콘텐츠 개발 환경을 제공한다[5-10, 21-25].

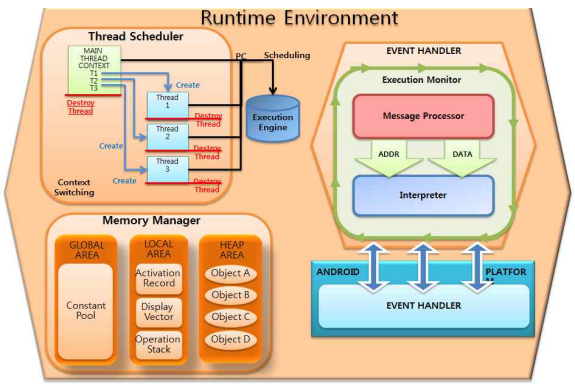

그림 8. 실행 환경의 구조

3.6 JNI 커넥터

JNI(Java Native Interface)는 Java가 아닌 다른 언어로 작성된 코드를 자바에서 호출하도록 만들어진 규약인데 스마트 가상기계에서 Java 언어를 사용하는 안드로이드 플랫폼과 상호 데이터 처리가 이루어지기 위해서는 JNI 커넥터를 사용해야 한다.

그림 9는 가상기계에서 구현된 JNI 커넥터의 구조를 나타낸 것이다.

스마트 가상기계에서 JNI를 쓰는 이유는 빠른 처리 속도를 요구하는 루틴이나 하드웨어 제어, 기존의 C/C++ 프로그램을 소스 코드의 변경 없이 사용하기 위하여 사용된다. 또한, 스마트 가상기계에서 사용되는 그래픽 처리와 같은 작업을 할 때 속도개선을 할 수 있고, 기존의 콘텐츠에서 자바가 제공하는 서비스를 이용할 수 있기 때문에 C/C++로 만들어진 스마트 가상기계를 안드로이드 플랫폼 환경에 탑재하여 실행시키기 위해서는 반드시 필요한 부분이다[5-10].

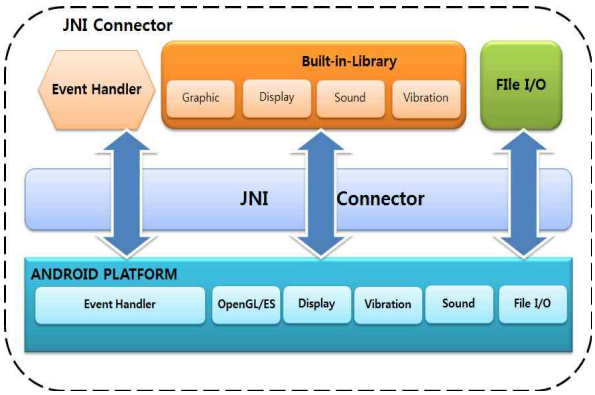

그림 9. JNI 커넥터의 구조

4. 실험 결과 및 분석

본 논문에서 제안하고 구현한 스마트 가상기계의 기능을 검증하기 위해 안드로이드 플랫폼이 탑재된 다양한 스마트 기기에서 기존의 모바일 게임 콘텐츠를 스마트 크로스 플랫폼의 컴파일러와 어셈블러, 그리고 가상기계로 변환하여 실행시켰다. 그림 10은 실험에 사용된 “Aiolos” 게임 콘텐츠의 소스를 나타낸 것이다.

그림 11은 컴파일러가 “Aiolos” 게임 콘텐츠의 소스를 입력으로 받아 중간언어 기반의 SIL 코드를 출력하여 나타낸 것이다.

```
#include "Z:\3_SVM\5_GnexGame\[SVM]AioloS\sys_lib.h"
#include "Z:\3_SVM\5_GnexGame\[SVM]AioloS\SScript.h"
                . . . . . 중간 생략 . . . . . .
int swData;
int swWidth;
int swHeight;
int swFrame;
int swFrame2;
int swFrame3;
int swFrame6;
                . . . . . 중간 생략 . . . . . .
void main()
{
        externalInit();
#ifdef DISPLAY_FPS
        NOFPS = 1;
        SetStartTime();
#endif
        SetBackLight(S_ON);

        gCX = swWidth / 2;
        gCY = swHeight / 2;
        gGameState = GS_LOGO;
        gPauseState= PS_MENU;
        SoundInplay = 0;
        gSilentMode = 0;
        gSpecialCount = 0;

        ReadRom();
        GameLogo(ES_START, swData);
        SetTimer(50, 1);

}
                . . . . . 이하 생략 . . . . . .
```

그림 10. Aiolos.c - 게임 소스 프로그램

```
%%HeaderSectionStart
  %DefinedLiteralCount    99
  %InitializedVariableCount          323
  %UninitializedVariableCount        234
  %ExternalVariableCount 0
  %ExternalFunctionCount0
  %InitFunctionName       0
  %EntryFunctionName      &main
  %SourceFileName         Aiolos.c
%%HeaderSectionEnd
%%CodeSectionStart
                . . . . . 중간 생략 . . . . . .
  %FunctionStart
    .func_name            &main
    .func_type            2
    .param_count          0
    .opcode_start
        proc              0           1           1
        ldp
        call              &externalInit
        ldc.i             1
        str.i             0           $NOFPS
        ldp
        call              &SetStartTime
        lod.i             0           $swWidth
        ldc.i             2
        div.i
        str.i             0           $gCX
        lod.i             0           $swHeight
        ldc.i             2
        div.i
        str.i             0           $gCY
        ldc.i             0
        str.i             0           $gGameState
        ldc.i             0
        str.i             0           $gPauseState
  %FunctionEnd
                . . . . . 중간 생략 . . . . . .
%%DataSectionStart
  %LiteralTableStart
        .literal_start    @0          0           17
0x46,0x3a,0x20,0x25,0x64,0x20,0x46,0x50,0x53,0x3a,0x20,0x25,0x64,0x2
e,0x25,0x64,0x00
        .literal_end
        .literal_start    @1          0           3
0x25,0x64,0x00
                .literal_end
                . . . . . 이하 생략 . . . . . .
```

그림 11. Aiolos.saf - 컴파일러가 생성한 중간코드 어셈블리 파일

스마트 가상기계는 어셈블러가 생성한 실행파일인 SEF 파일을 입력으로 받아 실행하여 결과를 출력한다. 그림 12는 안드로이드 플랫폼이 탑재된 다양한 스마트 기기에서 스마트 가상기계가 실행파일인 SEF를 입력으로 받아 프로그램을 실행한 결과를 나타낸 것이다. 그림 12에서 보는 것과 같이 “Aiolos” 게임이 잘 실행되는 것을 확인 할 수 있다.

그림 12. 스마트 기기와 갤럭시 탭에서 Aiolos 실행화면

5. 결론 및 향후 연구

본 논문에서 스마트 크로스 플랫폼을 지원하는 스마트 가상기계를 설계하고 구현하였다. 스마트 가상기계를 구현함으로써 게임 콘텐츠 같은 프로그램 제작을 플랫폼에 구애받지 않고 스마트 기기에서 콘텐츠를 실행할 수 있는 환경을 제공할 수 있다. 특히, C/C++, Java 언어를 모두 수용함으로써 프로그래머는 언어의 제약 없이 프로그램을 개발하고, 플랫폼의 제약 없이 프로그램을 실행할 수 있다.

또한, 하나의 언어로 개발된 스마트 콘텐츠를 스마트 가상기계가 탑재된 스마트 기기에서 플랫폼 독립적으로 실행하게 해 줌으로써 스마트 콘텐츠를 플랫폼마다 개발해야하는 시간과 노력을 줄이고, 하나

의 스마트 콘텐츠를 스마트 가상기계가 탑재된 여러 플랫폼에서 실행하게 함으로써 스마트 콘텐츠의 OSMU(One Source Multi Use)를 구현할 수 있다.

앞으로 스마트 가상기계에 다양한 그래픽 라이브러리를 추가하여 다양한 종류의 게임을 실행할 수 있도록 확장해야 하며, 스마트 가상기계를 iOS 플랫폼과 Windows Phone 플랫폼 등에 탑재하도록 확장해야 한다. 또한, 스마트 가상기계의 성능을 향상시키기 위해 최적화 작업을 진행할 예정이다.

참 고 문 헌

[1] Apple, iOS Reference Library, iOS Technology Overview, http://developer.apple.com/devcenter/ios, 2013.

[2] Goole, Android-An Open Handset Alliance Project http://code.google.com/intl/ko/android/ 2013.

[3] Microsoft, Windows Phone Dev Center, http://dev.windowsphone.com/en-us/develop, 2013.

[4] Samsung, http://developer.bada.com, 2013.

[5] 이양선, "임베디드 시스템을 위한 가상기계 기술," 멀티미디어 학회지, 제6권, 제2호, pp. 36-44, 2002.

[6] 오세만, 이양선, 고광만, "임베디드 시스템을 위한 가상기계의 설계 및 구현," 멀티미디어학회 논문지, 제8권, 제9호, pp. 1282-1291, 2005.

[7] YangSun Lee and YunSik Son, "A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms," *International Journal of Smart Home, SERSC,* Vol. 6, No. 4, pp. 93-105, 2012.

[8] YangSun Lee and YunSik Son, "A Study on Verification and Analysis of Symbol Tables for Development of the C++ Compiler," *International Journal of Multimedia and Ubiquitous Engineering, SERSC,* Vol. 7, No. 4, pp. 175-186, 2012.

[9] YunSik Son and YangSun Lee, "Design and Implementation of an Objective-C Compiler for the Virtual Machine on Smart Phone," *CCIS(Communications in Computer and Information Science)*, Vol. 262, No. 1, pp. 52-59, 2011.

[10] YunSik Son and YangSun Lee, "A Study on the Java Compiler for the Smart Virtual Machine Platform," *CCIS(Communications in Computer and Information Science)*, Springer, Vol. 353, No. 1, pp. 135-140, 2012.

[11] YangSun Lee and YunSik 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,* Vol. 7, No. 4, pp. 95-106, 2012.

[12] Tim Lindholm and Frank Yellin, *The Java Virtual Mahine Specification*, Addison-Wesley, 1999. USA.

[13] Bill Venners, *Inside the JAVA Virtual Machine*, 2nd ed., McGraw-Hill, 1999. USA.

[14] Joshua Engel, *Programming for the Java (TM) Virtual Machine*, Addison-Wesley, 1999. USA.

[15] James E.Smith, *Virtual Machines*, Morgan Kaufmann, 2005. USA.

[16] Sun Microsystems, *Memory Management in the Java HotSpot Virtual Machine*, April. 2006. USA.

[17] John Gough, *Compiling for the .NET Common Language Runtime (CLR)*, Prentice Hall PTR, 2001. USA.

[18] Kevin Burton, *.NET Common Language Runtime*, SAMS, 2002. USA.

[19] Serge Lidin, *Inside Microsoft .NET IL Assembler*, Microsoft Press, 2002. USA.

[20] Levine, John, *Linkers and Loaders*, Morgan Kaufmann, 2000. USA.

[21] 박진기, 임베디드 가상기계를 위한 번역기 시스템, 서경대학교 공학박사 학위논문, 2005.

[22] 손민성, 박진기, 이양선, "u-VM을 위한 CPP 컴파일러의 개발," 한국멀티미디어학회 춘계학술발표논문집, 제10권, 제1호, pp. 755- 758, 2007.

[23] 최홍석, 이양선, "유비쿼터스 게임 플랫폼을 위

한 SAF 어셈블러의 설계 및 구현," 정보처리학회 춘계학술발표대회 논문집, pp. 1516- 1519, 2007.

[24] 백대현, Real-Time 운영체제를 위한 KVM의 클래스 로더와 메모리 시스템의 구현, 충남 대학교 석사학위 논문, 2005.

[25] Marc Berndl, Benjamin Vitale, Mathew Zaleski, and Angela demke Brown, "Contxt Threading : A Flexible and Efficient Dispatch Technique for Virtual Machine Interpreters," *Proc. International Symposium on Code Generation and Optimization(CGO'05), pp. 15-26,* 2005.

[26] 이양선, 임베디드 시스템을 위한 기상기계코드 기반의 C++ 컴파일러 개발, 서경대학교 산학협력단, 2006.

[27] 이양선, iDTV를 위한 Java 컴파일러와 디버깅 도구 개발, 서경대학교 산학협력단, 2008.

한 성 민

2011년 서경대학교 컴퓨터공학과 공학사
2013년 서경대학교 컴퓨터공학과 공학석사
2013년～현재 인크로스 연구원
관심분야 : 스마트 가상기계, 스마트 크로스 플랫폼, 스마트 프로그래밍, 모바일 컴퓨팅 등

손 윤 식

2004년 동국대학교 컴퓨터공학과 공학사
2006년 동국대학교 대학원 컴퓨터공학과 공학석사
2009년 동국대학교 대학원 컴퓨터공학과 공학박사

2010년～현재 동국대학교 전문연구원
관심분야 : 프로그래밍 언어, 컴파일러, 모바일/임베디드 컴퓨팅, 로봇 소프트웨어, 시큐어 코딩

이 양 선

1985년 동국대학교 전자계산학과 공학사
1987년 동국대학교 대학원 컴퓨터공학과 공학석사
1993년 동국대학교 대학원 컴퓨터공학과 공학박사

1994년 3월～현재 서경대학교 컴퓨터공학과 교수
1996년 3월～2000년 2월 서경 대학교 전자계산소 소장
2000년 2월～현재 한국멀티미디어학회 이사
2005년 1월～2006년12월 한국멀티미디어학회 총무이사
2006년 1월～현재 한국정보처리학회 게임연구회 위원장
2006년 1월～현재 한국정보처리학회 이사
2009년 1월～2009년 12월 한국멀티미디어학회 부회장, 논문지 편집위원장
관심분야 : 프로그래밍 언어, 컴파일러, 모바일/임베디드 소프트웨어, 스마트 크로스 플랫폼 등