Journal of Korea Multimedia Society Vol. 13, No. 7, July 2010(pp. 1051-1061)

소스 레벨 콘텐츠 변환기를 이용한 GNEX C-to-Android Java 변환기의 설계 및 구현

손윤식[†], 오세만[††], 이양선[†††]

요 약

국내 이동통신사들이 서로 다른 모바일 플랫폼을 채택하여 사용함으로 인해 개발자는 하나의 모바일 게임 콘텐츠를 서비스하기 위하여 각각의 플랫폼 특성에 맞추어 변환 작업을 하여야 한다. 하지만, 모바일 게임 콘텐츠를 타 플랫폼으로 이식하기 위한 변환 작업에 많은 시간과 비용이 소모되고 있다. 이는 다양한 콘텐츠가 제공되지 못하고 있는 원인이기도 하다. 본 논문에서는 이런 문제를 해결하기 위해 소스 레벨 콘텐츠 번역기를 이용하여 GNEX 플랫폼의 모바일 C 게임 콘텐츠를 스마트 플랫폼인 Android 플랫폼의 자바 콘텐츠로 자동으로 변환해주는 콘텐츠 자동 변환기 시스템을 구현하였다. GNEX C-to-Android Java 콘텐츠 자동 변환기 시스템은 단시간 내에 다른 플랫폼으로 콘텐츠를 이식할 수 있도록 하여 동일 콘텐츠를 다른 이동통신사에 서비스하는데 소모되는 시간과 비용을 최소화해준다. 또한, 기존 콘텐츠를 자동 변환하여 타 플랫폼에 서비스함으로써 콘텐츠의 재사용성을 높이고, 신규 콘텐츠의 생산성을 높여 사용자에게는 다양한 모바일 게임 콘텐츠를 제공할 수 있도록 지원한다.

Design and Implementation of the GNEX C-to-Android Java Converter using a Source-Level Contents Translator

Yun-Sik Son[†], Se-Man Oh[††], Yang-Sun Lee[†††]

ABSTRACT

Since Korean mobile communication companies each use different mobile platforms, developers must configure and translate their game contents to run under each of the platforms so that they can be serviced correctly. Nevertheless, such translation tasks require lengthy times and costs. This is one of the reasons why a variety of contents could not be provided. In order to mitigate such difficulty, this paper implemented an automatic mobile contents translating system that automatically translates mobile C game contents of the GNEX platform to mobile java contents of the Android platform as a smart platform using a source-level contents translator. The GNEX C-to-Android Java automatic contents translation system helps minimize the amount of time and cost required in servicing contents to different mobile communication companies by promptly translating a platform-specific-content to run under other platforms. Also, the automatic translation and servicing of existing contents increases the reusability of these contents and also the productivity of new contents thereby offering users with a more variety of games.

Key words: Automatic Mobile Contents Translator(모바일 콘텐츠 자동 변환기), Mobile Platform(모바일 플랫폼), Android Platform(안드로이드 플랫폼), Source-Level Contents Translator(소스 레벨 콘텐츠 변환기)

※ 교신저자(Corresponding Author) : 오세만, 주소 : 서울시 중구 필동 3가 26 동국대학교 컴퓨터공학과, 전화 : 02-2260-3342, FAX : 02-2265-8742, E-mail : smoh@dongguk.edu
접수일 : 2010년 3월 12일, 수정일 : 2010년 4월 5일
완료일 : 2010년 4월 5일

[†] 동국대학교 전문연구원
(E-mail : sonbug@dongguk.edu)
[††] 동국대학교 컴퓨터공학과 교수
[†††] 서경대학교 컴퓨터공학과 교수
(E-mail : yslee@skuniv.ac.kr)

1. 서 론

최근까지 이동 통신 산업은 하드웨어와 소프트웨어 측면에서 지속적인 성장을 하고 있다. 통신 기술의 비약적인 발전은 표준화된 모바일 플랫폼에 대한 필요성으로 WIPI 플랫폼이 등장하게 되었고, 기술의 확산과 저변화로 iPhone이나 Android, Window Mobile과 같은 다양한 스마트 폰 플랫폼이 등장하게 되었다. 모바일 콘텐츠에 대한 사용자의 수요가 증가하고 있는 시점에서 새로운 플랫폼은 통신사에 특화된 기존 플랫폼에서 개발된 콘텐츠를 사용할 수 없기 때문에 단기간에 다양한 콘텐츠를 개발하는데 어려움이 따르고 있다. 따라서 특정 통신사의 플랫폼에서 서비스되는 다양한 콘텐츠를 새로운 모바일 플랫폼으로 이식할 필요성이 제기되고 있다[1-6].

본 논문에서는 컴파일러 제작 기술을 이용하여 소스 레벨 콘텐츠 변환기를 설계하고 구현하여 서로 다른 언어 간의 변환 문제에 대한 해결 방법을 제시한다. 변환 대상이 되는 언어들을 언어적 측면과 라이브러리 측면에서 분석하여 문법적인 문제를 해결하고 제공되는 라이브러리의 매핑 관계를 정의한다. 변환기는 기능별로 독립적인 모듈로 설계하며, 모듈간의 상호 호환성을 고려한다. 실험에서는 제안한 방법론을 적용하여 GNEX의 모바일 C 콘텐츠를 Android 플랫폼에서 실행 가능한 Java 콘텐츠로 변환하는 소스 레벨 콘텐츠 변환기를 개발하였다. 변환기 개발을 통해 GNEX 모바일 C로 개발된 많은 콘텐츠를 Android 플랫폼에서 서비스할 수 있는 콘텐츠로 자동 변환할 수 있는 환경을 구축하여 다양한 콘텐츠를 제공할 수 있게 하였다.

제안한 방법론은 GNEX 모바일 C 언어에 대한 활용도를 증대 시키고 GNEX 모바일 C 언어로 개발된 다양한 휴대폰 콘텐츠의 활용 폭을 확대할 수 있다. 이러한 연구 효과는 휴대폰 콘텐츠 산업 활성화에 크게 기여할 수 있을 것으로 기대된다.

2. 관련연구

2.1 GNEX 플랫폼

GNEX 가상기계(Virtual Machine)는 GNEX 응용 프로그램을 해석하고 실행하는 역할을 한다. GNEX 커널은 다양한 시스템 인터페이스를 제공하며 메모리 관리자 탑재로 GNEX 시스템을 보고하고 응용 프로그램의 크기 및 힙(Heap) 메모리 제약 등을 해소하는 역할을 한다. 이벤트 핸들러(Event Handler)는 플랫폼의 이벤트를 받아 GNEX 이벤트로 변환하고 각 이벤트에 대응되는 알고리즘을 호출하여 처리한다. MIDD(Mobile Interface Device Driver)는 사운드 재생, LCD 출력 등 단말기의 하드웨어 관련 기능을 단말기 플랫폼에서 제공하는 API를 이용하여 구현한 것이며, 플랫폼 API의 호출과 실행 결과를 처리한다.

GNEX 응용 프로그램은 ANSI C 언어를 기반으로 한 모바일 C 언어로 개발한다. GNEX는 VDI (Variable Depth Image)라는 단말기 전용으로 설계된 이미지 형식을 사용한다. VDI는 픽셀(Pixel) 당 할당되는 비트 수를 가변적으로 정의하여 사용하는 일종의 비트맵 형식의 이미지 규격이다. GNEX 응용 프로그램에서는 BMP와 같은 이미지 파일을 GNEX SDK에서 제공하는 이미지 도구를 통해 VDI 형식으로 변환하여 모바일 C 소스 코드에 포함시킨다. 사운드 리소스 역시 GNEX 응용 프로그램에서 사용할 수 있도록 GNEX 규격의 사운드 파일로 변환하여 모바일 C 소스 코드에 포함하여 사용한다[7-10].

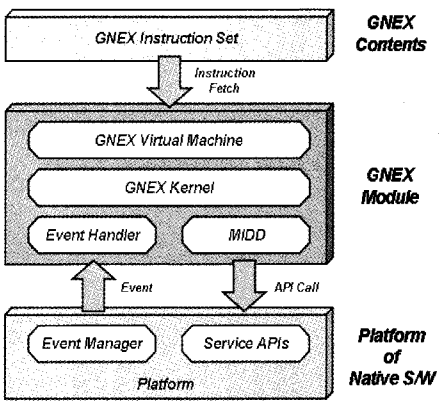


그림 1. GNEX의 구조

2.2 Android 플랫폼

Android 플랫폼은 구글에서 개발한 운영체제, 미들웨어, 응용 프로그램을 포함하는 모바일 기기에 최적화된 플랫폼이다. Android 플랫폼은 오픈 소스 정책을 채택하였으며, 리눅스 커널, 라이브러리, 런타임, 애플리케이션 프레임워크, 애플리케이션으로 구

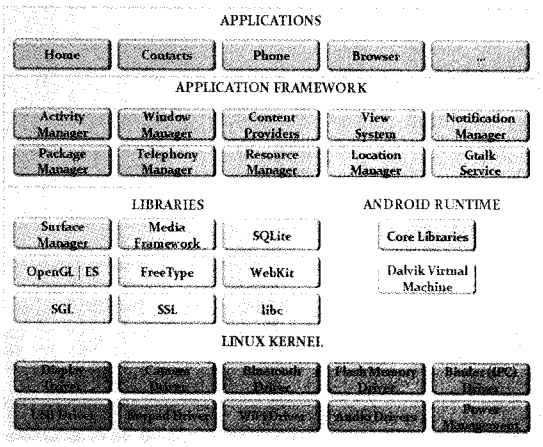


그림 2. Android 플랫폼의 계층 구조

성되어 있다[5,6].

그림 2는 Android 플랫폼의 계층 구조와 구성요소를 도시한 것이다. 리눅스 커널은 보안, 메모리 관리, 프로세스 관리, 네트워크 스택, 드라이버 모델과 같은 리눅스 버전 2.6의 핵심 시스템 서비스를 이용하며, 커널은 하드웨어와 소프트웨어 간의 추상 계층으로 동작한다. 라이브러리는 C와 C++로 구성되었으며 C 시스템 라이브러리, 미디어 라이브러리, 3D 라이브러리 등을 제공한다. 애플리케이션 프레임워크는 자바로 구성된 패키지 컴포넌트이며, 애플리케이션은 이 프레임워크의 패키지를 사용하여 작성할 수 있다.

Android 플랫폼의 모든 응용 프로그램은 자바로 작성되며, 작성된 응용 프로그램은 자바 컴파일러에 의해 클래스 파일로 변환된다. 생성된 클래스 파일은 실행에 앞서 DEX(Dalvik Executable File) 형식으로 다시 변환되며, dalvik 가상기계에 의해 실행된다. DEX 파일은 저장 공간과 메모리 사이의 효과적인 매핑 정의에 최적화된 포맷이며, dalvik 가상기계는 레지스터 기반의 가상기계로 제한된 메모리에 최적화된 특징을 가진다.

2.3 기존의 모바일 콘텐츠 변환기

그동안 국내 모바일 시장이 활성화 되었음에도 불구하고 모바일 콘텐츠 변환기에 대한 연구는 매우 부족하여 사례가 많지 않은데다가 기존의 모바일 콘텐츠 변환기는 대부분 동일한 프로그래밍 언어 환경에서의 변환만을 지원하거나 자동변환을 지원하지 않고 있어 프로그래머가 수작업으로 많은 부분을 변환시켜줘야 하는 실정이다.

먼저, 기존의 모바일 콘텐츠 변환기 중에는 XML을 이용하여 자바 콘텐츠의 변환을 시도한 연구가 있었으며[11-14], GVM 플랫폼의 모바일 C 콘텐츠를 WIPI C나 Java로 변환하는 연구가 있었다[15,16]. 또한, 변환 대상 소스 코드에서 사용되는 API를 실행 대상 환경에 거의 유사한 형태로 동일 기능을 구현한 Wrapper 함수를 정의하여 소스 코드의 변경 없이 변환 목적 플랫폼에서 동일한 동작을 하도록 BREW C와 WIPI C를 상호변환하거나[17], GVM C를 BREW C로 변환하는 연구가 있었지만[18] 변환할 때 소스코드가 자동으로 변환되지 않아 사용자가 개입하여 수동으로 변환을 해야 하는 등의 단점을 가지고 있다. 한편, 컴파일러 제작기술[19,20]을 활용하여 모바일 콘텐츠의 자동변환에 대한 연구가 시도되어 GVM 플랫폼의 모바일 C 콘텐츠를 WIPI C나 자바로 또는 MIDP 자바로 변환하는 연구가 발표되어[1,2,4] 시간과 비용을 최소화하고 콘텐츠의 재사용을 높여 생산성을 향상시킬 수 있는 방법이 제시되었다. 본 논문에서는 이와 같은 선행연구를 기반으로 컴파일러 제작기술과 플랫폼 매핑기술을 사용하여 모바일 콘텐츠를 스마트폰 콘텐츠로 자동 변환하는 콘텐츠 변환기를 개발하였다.

3. GNEX C-to-Android Java 변환기의 설계 및 구현

본 장에서는 재목적 소스변환기 모델을 이용하여 소스 레벨 콘텐츠 변환기를 구성하며, GNEX C와 Android Java 언어 변환 시 발생하는 특징과 GNEX C와 Android 플랫폼의 라이브러리 매핑 구조를 기술한다.

3.1 소스 레벨 콘텐츠 변환기

소스 레벨 콘텐츠 변환기는 서로 다른 언어로 작성된 프로그램을 소스 레벨에서 변환하는 변환기이다. 소스 레벨에서 변환은 프로그래밍 언어 차원에서의 변환을 의미하며, 이는 서로 다른 언어 간의 차이점을 해결하여, 동등한 의미를 갖는 서로 다른 언어로의 변환을 의미한다. 그림 3은 다양한 대상 언어를 생성할 수 있는 재목적 소스 변환기를 그림으로 도식

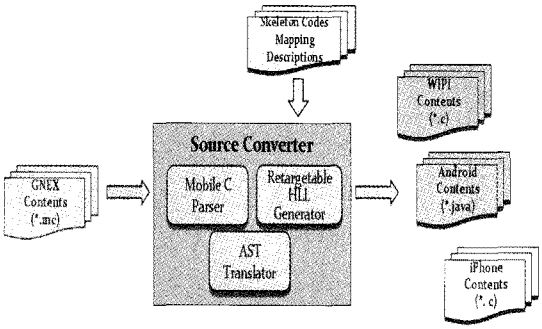


그림 3. 재목적 소스 변환기 시스템 구성도

화한 것이다.

재목적 소스 변환기는 특정 콘텐츠를 입력 받으며, 대상 언어에 의존적인 골격 프로그램, 매핑 정보를 사용하여 다양한 종류의 언어로 변환이 가능하다. 입력으로 GNEX 모바일 C 콘텐츠를 사용하며, 소스 변환기의 내부 모듈은 모바일 C 콘텐츠를 파싱하여 AST(Abstract Syntax Tree)를 생성하는 파서와 AST를 효과적으로 사용하기 위한 AST 변환 모듈이 있으며, 변환된 AST를 사용하여 대상 언어에 대한 소스 프로그램을 생성하는 재목적 HLL(High Level Language) 생성기로 구성되어 있다. 본 논문에서는 소스 변환기의 결과물이 Android Java 프로그램이므로, 재목적 소스 변환기를 적용하여 GNEX C-Android Java 소스 변환기의 시스템을 설계하였다.

3.2 GNEX C 언어 및 프로그램 분석

GNEX C 언어를 Android Java로 변환하기 위해서는 먼저 GNEX C의 특징과 프로그램 구조를 확인해야한다. GNEX C는 Mobile C라고도 하며, GNEX에서 실행되는 응용 프로그램을 작성하는 언어규격이다. GNEX C는 ANSI C 언어의 규격을 작은 크기의 LCD, 적은 메모리, 상대적으로 컴퓨팅 파워가 열세한 단말기의 특성에 맞게 최적화하여 만든 것으로 문장 구조는 ANSI C와 동일하지만, 자료형(Data Type) 및 API에서 차이가 있다. 표 1은 GNEX C의 언어적 특징 중 자료형에 대한 내용을 요약한 것이다.

표 1에서 나타나듯이 GNEX C는 ANSI C와 비교하여 상이한 면이 있다. 먼저, 자료형을 4바이트 정수형으로 제한하고 문자형, 실수형은 지원하지 않는 대신에 image, sound, voc, string 형을 제공한다. 또한, 자동변수(Automatic Variable)를 지원하지 않으며, 모든 변수는 정적 변수(Static Variable)로 처리된다. GNEX C에서 지원하는 기본 자료형과 확장 자료형은 별도의 내부 메모리 모델을 가지는데, 이는 GNEX 콘텐츠를 변환하는 경우 반드시 고려해야하는 부분이다. 그림 4는 GNEX에서 일반 자료형 변수와 확장 자료형 변수가 간접적으로 접근되는 구조를 보여주고 있다.

프로그램 구조상으로도 ANSI C가 절차형 언어인 반면, GNEX C는 이벤트 구동(Event-Driven) 방식으로, 프로그램 진입점(응용프로그램 시작 이벤트 핸들러)인 main 함수 뿐만 아니라, 키패드 입력, 타이머 이벤트, 네트워크 데이터수신, 응용프로그램 종료 등의 이벤트 핸들러 집합으로 구성된다.

표 1. GNEX C 자료형의 특징

자료형	지원 유무	비 고
정수형	○	4byte signed integer
열거형	○	열거형 변수는 지원되지 않음.
상수형	○	
1,2차원 정수형 배열	○	배열을 함수의 매개변수로 사용할 수 없음.
실수형	×	
구조체	○	구조체변수는 단순구조변수, 구조배열변수, 구조 포인터 변수, 배열 구조 포인터변수로 사용할 수 있으며, 구조배열의 경우 1차원 배열만 지원함.
유니온	×	
문자형	×	
포인터	○	연산자를 사용하여 단순변수와 1차원 배열의 포인터 변수 정의가 가능함. Function Pointer를 지원하지 않음. malloc()함수를 지원하지 않음. GNEX Library로 정수형 메모리 할당의 경우 MallocInt()5를 사용하고, 미디어 메모리 할 당의 경우 SetMediaSize()를 사용함.
확장 자료형 지원		string string 1차원 배열 Image, sound, voc, binary Image, sound, voc, binary 1차원 배열

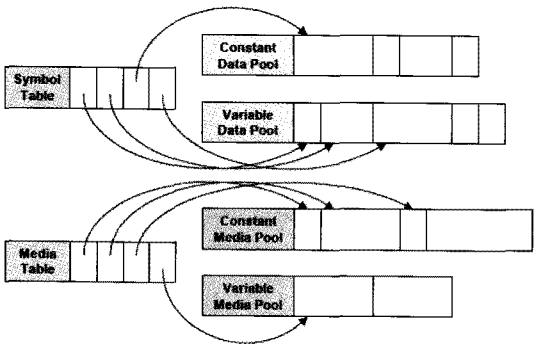


그림 4. GNEX에서의 메모리 관리 모델

3.3 GNEX C-to-Android Java 변환기

소스 레벨 콘텐츠 변환기를 이용하여 구성한 GNEX C-Android Java 변환기의 시스템 구성도는 그림 5와 같다.

소스 변환기는 먼저, GNEX C 소스 프로그램을 분석에 용이한 AST 형태로 변환하기 위한 C 전처리기(preprocessor), 어휘분석기(Scanner), 구분분석기(Parser) 모듈과 AST에 자바 소스 생성을 위한 정보를 추가하고 ANSI C 언어에 적합한 AST로 변환하는 AST 번역기 모듈로 구성되며, C 언어에서 자바 언어로 변환하기 위한 모듈 클러스터, 코드패턴과 골격 프로그램, 라이브러리 등을 기반으로 한 자바 고급언어 생성기로 구분된다.

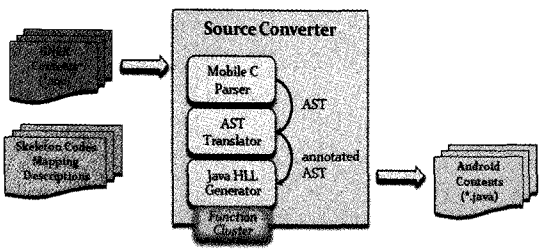


그림 5. GNEX C-Android Java 변환기의 시스템 구성도

3.3.1 소스 코드 변환

3.2절에서 살펴본 GNEX C의 특징을 기준으로 GNEX C로 작성된 콘텐츠를 Android Java 콘텐츠로 변환하기 위한 언어 변환은 물리적인 변환과 의미적인 변환 두 가지로 구분할 수 있다.

먼저 물리적인 변환은 파일로 구분된 함수 기반의 C 프로그램을 클래스 기반의 자바 프로그램으로 변경하기 위해 고려되는 내용으로 주로 매핑관계에 의해 처리 가능한 내용이다. C 언어로 작성된 프로그램은 함수 단위로 구성되어있는데 반해, 자바로 작성된 프로그램은 클래스 기반의 구조를 가진다. 따라서 언어 변환 시 클래스 생성 방법, 전역 변수, 지역 변수 등에 대한 변환 연구가 필요하며, 본 논문에서는 다음과 같이 파일, 타입, 변수, 함수 측면에서 변환 규칙을 정의한다.

먼저, 하나의 GNEX C 소스 파일은 하나의 자바 파일로 변환한다. 데이터 형은 GNEX C와 자바에서 공통적으로 제공하는 기본형 이외에 구조체와 열거형에 대한 변환을 한다. GNEX C 언어의 변수는 전역 변수와 지역 변수로 선언이 가능하지만 실제로는 정적 변수이므로, 전역 변수는 클래스의 필드 형태로 변경되고 로컬 변수는 메소드의 정적 지역 변수 형태로 변환된다. 마지막으로 함수는 메소드로 대체된다. 그리고, GNEX C 프로그램에서 전역 변수 선언과 사용자 정의 자료형의 선언 등의 데이터와 같은 공용 데이터는 별도의 자바 파일로 관리한다. 이는 다른 소스 파일에서 사용할 가능성이 높은 자료에 대한 효과적인 관리를 위한 것이다[5,6,9,10,21]. 표 2는 GNEX C의 기본 자료형외에 다른 자료형과 확장 자료형을 자바로 변환할 경우의 대응관계를 나타낸 것이다.

다음으로 의미적인 변환은 동일하거나 유사한 기능이 존재하지 않는 경우 의미적인 관점에서 동등한 기능으로 변환이 이루어지는 경우로, 본 논문에서는 포인터에 대한 내용을 다룬다. GNEX C-Android Java 변환 시에는 앞서 언급한 파일 및 클래스 구조에 대한 효과적인 분리 방법과 더불어 포인터 처리 방법 또한 신중히 설계해야 한다. 포인터는 C 언어에서 높은 사용빈도를 보이고, 프로그래밍 언어적인 측면에서 매우 민감한 요소이다. 자바에서는 포인터를 제공하고 있지 않기 때문에 C 언어를 자바 언어로

표 2. GNEX C와 Android Java의 자료형 대응 관계

GNEX C	Android Java	비 고
struct	클래스	
enum	상수 필드	public static final int
string	StringBuffer 클래스	
image	GnexImage 클래스	사용자 정의 클래스
sound	GnexSound 클래스	사용자 정의 클래스

변환 시 포인터 처리 방법론을 마련하는 것이 큰 비중을 차지한다고 할 수 있다. 기존에 제시된 포인터 처리 방법론[22]은 많은 객체와 메모리 공간을 사용하고, 변수 참조에 대한 접근 횟수가 큰 폭으로 증가하기 때문에 제한된 리소스를 갖는 모바일 환경에 적용하기에 어려움이 있다.

포인터 변환 방법을 설계하기 위해서는 포인터 연산과 메모리 관리 측면에 대한 고려가 필요하다. 포인터 연산은 산술 연산과 간접 참조, 형 변환이 있으며, 메모리 관리 측면에서는 메모리의 할당과 제거 문제가 있다. 자바에서 주요 해결 문제는 간접 참조, 주소 계산, 메모리 할당/제거, 형 변환 문제를 생각할 수 있다. 이와 같은 문제점을 해결하기 위해서는 간접 참조와 메모리 관리자를 에뮬레이션하고 포인터 변수와 포인터를 참조하는 변수에 대한 정적 분석[23]이 필요하다. 포인터 연산의 처리는 인덱스를 위한 변수를 사용하는 방법과 포인터 변수를 위한 래퍼(Wrapper) 클래스를 정의하는 방법이 있다. 메모리 관리 측면에서는 포인터 자료형을 위한 정적 배열을 사용하는 방법과 자바에서 제공하는 ArrayList와 같은 클래스를 사용하는 방법이 있다.

본 논문에서는 속도와 메모리 공간 문제를 고려하여 일반 자료형 변수에 대해 인덱스를 위한 배열과 ArrayList 클래스를 사용하여 포인터 문제를 해결한다. 반면, 확장 자료형인 image형과 같은 경우는 표 2에서 언급한 바와 같이 GnexImage 클래스와 같이 사용자 정의형 클래스를 이용하여 참조 및 할당과 같은 문제를 해결한다. 또한, GNEX에서 사용하는 간접 주소 메모리 모델을 올바르게 반영해야 때문에 본 논문에서는 GNEX 메모리 모델을 에뮬레이션하여 포인터와 동적 메모리 할당 문제를 해결하였다. 그림 6은 포인터 처리 방법론을 도식화한 것이다.

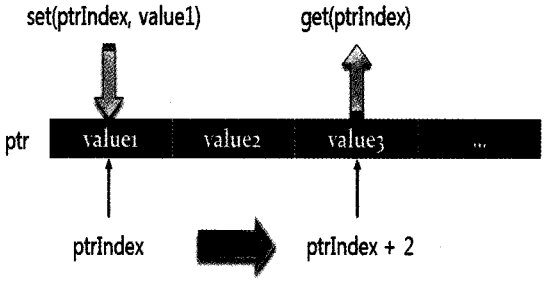


그림 6. 포인터 처리 방법

3.3.2 API 매핑

API 매핑은 GNEX C에서 제공하는 API를 Android API를 이용하여 구성하는 것을 의미한다. GNEX C는 16개의 그룹으로 분류된 425개의 API를 제공한다. 콘텐츠 동작에 필요한 주요 함수는 8개 그룹에 포함된 237개의 함수이며, 대부분 그래픽, 핸드셋 콘트롤 API가 이에 해당된다. 그림 7은 GNEX C에서 제공하는 API 카테고리를 정리한 것이다[9].

Android API는 총 122개의 패키지로 나누어져 있으며, 2,254개의 클래스에서 총 17,018개의 메소드를 제공한다(Android 1.2 기준)[5]. Android API의 패키지 분류는 표 3과 같다.

GNEX C와 Android API 매핑 관계에는 단순 매핑, 복합 매핑이 존재하며, 매핑되지 않는 API도 존재할 수 있다. 단순 매핑은 반환값, 매개변수, 함수의

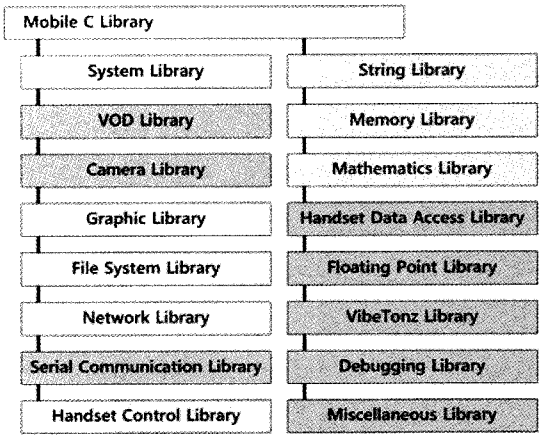


그림 7. GNEX C의 API 카테고리

표 3. Android API의 패키지 분류

패키지 분류	설 명
android.*	Android 애플리케이션 제작을 위한 기본 패키지
com.google.android.maps.*	구글맵을 사용하기 위한 패키지
dalvik.system.*	Android의 VM 관련 패키지
java.*	자바 제공 패키지들
javax.*	자바 제공 패키지들
junit.framework.*	테스트 케이스 작성을 위한 패키지
org.*	기타 패키지(XML, 아파치 관련 등)

표 4. Android의 주요 패키지

패키지 이름	설 명
android.util	낮은 수준의 유틸리티 클래스들 (특수한 컨테이너 클래스, XML 유틸리티 등)
android.os	기본적인 OS 서비스, 메시지 전달, 프로세스간의 통신
android.graphics	핵심 렌더링 패키지
android.text	텍스트 처리 도구, 입력 메소드 등
android.database	데이터베이스와 관련된 작업을 위한 낮은 수준의 API 들
android.content	디바이스상의 데이터에 접근하기 위한 서비스 (디바이스에 설치된 애플리케이션, 자원, 동적 데이터 등)
android.view	핵심 사용자 인터페이스 프레임워크
android.widget	표준 사용자 인터페이스 요소(리스트, 버튼 등)
android.app	Activity들을 사용하여 구현된 높은 수준의 애플리케이션 모델 제공

의미가 모두 동일한 경우로서 대부분 1:1 매핑이 가능하다. 복합 매핑은 하나의 API가 여러 개의 API로 표현되는 경우이며, 1:N 매핑이 가능하다. 또한 매개변수, 반환값의 자료형 변환이 필요한 경우와 API 매핑 단계에서 실매개변수, 반환값의 의미 차이로 인해 데이터를 가공해야 하는 경우가 있을 수 있다. 매핑이 불가능한 API는 알고리즘을 직접 구현해서 제공할 수 있으며 Android 플랫폼에서 구현이 불가능한 GNEX C API는 제외한다. 그림 8은 라이브러리 매핑 관계를 그림으로 표현한 것이다.

표 5는 GNEX 라이브러리의 그룹별 매핑 관계를 클래스 수준에서 분석한 결과이다.

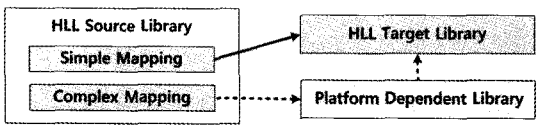


그림 8. 라이브러리 매핑 관계

4. 실험 결과 및 분석

소스 레벨 콘텐츠 변환기를 이용한 GNEX C-to-Android Java 변환기의 입력에 대한 출력 결과는 그림 9와 같은 방식으로 이루어진다.

표 5. 그룹별 매핑 관계 분석 결과

GNEX	Android
System	android.telephony.PhoneNumberUtils java.util.Date
Graphic	android.graphics.Canvas android.graphics.Color android.graphics.drawable.Drawable
File System	java.io.File java.io.FileOutputStream java.io.FileInputStream
Network	java.net.Socket; android.net.NetworkInfo java.net.URL
Handset Control	android.media.MediaPlayer android.media.AudioManager android.os.Vibrator
String	android.text.SpannableString java.lang.StringBuffer
Memory	직접 구현
Mathematics	java.lang.math

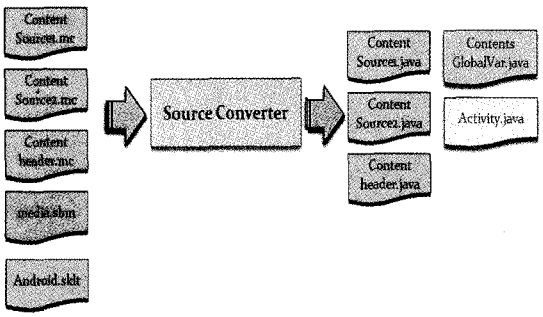


그림 9. 소스 레벨 콘텐츠 변환기

소스파일의 구조 정보를 유지하면서, 공통적으로 사용되는 정보는 ContentsGlobalVar.java 파일에 기록되며, 콘텐츠를 구동하기 위한 기본정보를 골격 파일(Android.sklt)을 이용하여 Activity.java 파일로 생성된다. 표 6과 표 7은 각각 GNEX C Aiolos 콘텐츠에 대한 ContentsGlobalVar.java 파일과 Activity.java 파일의 내용이다.

변환된 콘텐츠를 위한 API에 대한 검증은 게임 콘텐츠 개발에서 주로 사용되는 GNEX C API를 모두 테스트할 수 있도록 테스트 콘텐츠를 선별하여 진행하였으며, 선별된 콘텐츠를 구현된 GNEX C-to-Android Java 소스 변환기를 통해 Android 콘텐츠로 자동 변환하고, Android 플랫폼에서 실행 결

표 6. ContentsGlobalVar.java 파일 예(Aiolos)

```
package com.android.gnex.aiolos;

import com.android.gnex.GnexImage;
import com.android.gnex.GnexGlobalVar;

//*************************************************************
********
//                          Data classures
//*************************************************************
********

class _TouchKey {
                int left, top, right, bottom;
                int code1, code2;
                StringBuffer str;
}

class ContentsGlobalVar
{
        public static final int SWAP_NOTE_G3 = 0;
        public static final int SWAP_NOTE_G3P = 1;
        public static final int SWAP_NOTE_A3 = 2;
        // ...
        public static GnexImage ImgLogo[] = new GnexImage[12];
        public static GnexImage imgEnemyHadd[]
                                        = new GnexImage[4];
        public static GnexImage imgEnemyH[] = new GnexImage[2];
        // ...
        public static int _touchKeyView;
        public static StringBuffer HelpText[] = {
              new StringBuffer ("서기2090년...      "),
              new StringBuffer ("                   "),
              new StringBuffer ("인류는 새로운 대체"),
              // ...
        };
        // ...
        public static int MissionTextIdx[][] = {
                { 2, 4, 6, 9 },
                { 2, 6, 9, 13 },
                { 2, 6, 10, 15 },
                { 2, 6, 13, 16 }
        };
        public static int NVROM[] = new int[16];
        public static int gFlightMissileS[] = new int[10];
        // ...

}
```

표 7. Activity.java 파일 예(Aiolos)

```
package com.android.gnex.aiolos;

import android.app.Activity;
import android.os.Bundle;
import android.view.Window;


import com.android.gnex.aiolos.Aiolos.ContentsThread;

public class AiolosActivity extends Activity {
        private ContentsThread mThread;
        private Aiolos mView;

        @Override
        public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              requestWindowFeature(Window.FEATURE_NO_TITLE);

              setContentView(R.layout.main);

              mView = (Aiolos) findViewById(R.id.gnex);
              mThread = mView.getThread();
              setContentView(R.layout.main);
        }
}
```

과를 확인하여 변환된 API의 정상적인 동작을 검증하였다. 또한 각 콘텐츠의 실행 파일 크기를 비교하여 그 차이를 확인하였다. 표 8은 API에 대한 실험 결과이다.

기능별로 선정된 콘텐츠의 기능과 의미는 다음과 같다. MMI(Man Machine Interface)는 단말기의 전화번호, LCD 크기, 시간, SMS 등의 라이브러리를 테스트하기 위한 콘텐츠이며, 주로 클리핑 영역 계산, 시간 동기화 등을 구현할 때 사용된다. 그래픽 콘텐츠는 모바일 게임 콘텐츠 개발 시 가장 중요한 라이브러리로 다양한 게임 이미지를 화면에 표시하기 위해 사용된다. 수학 콘텐츠는 제곱근, 삼각함수 등 수치적인 계산을 위한 라이브러리를 테스트한다. 모바일 게임 콘텐츠에는 수치적인 계산 알고리즘을 적용하는 경우가 빈번하게 발생하며, 수치적인 계산 알고리즘을 지원하기 위한 수학 라이브러리는 계산 결과 오차가 발생할 경우 예상하지 못한 버그를 양산시킬 수 있다. 파일 콘텐츠를 통해서는 파일의 읽기와 쓰기 기능을 테스트할 수 있으며, 이와 같은 기능은 상황에 따라 실행 중인 콘텐츠를 중단시키고 진행 상황, 데이터 등의 정보를 저장하고 읽어오는 기능을 구현하는 데 사용된다. 하드웨어 콘텐츠는 콘텐츠 개발 시 필수적으로 사용되는 사운드 재생 기능을 포함한다. 마지막으로, 네트워크 콘텐츠는 네트워크 기능을 테스트할 수 있는 콘텐츠로서 추가적인 아이템 구매, 게임 기록 등록 등을 구현할 때 주로 사용된다.

기능별 테스트를 통하여 주요 기능이 정상적으로 동작함을 확인하였지만, 변환된 콘텐츠는 GNEX의 커널과 API를 모두 포함하기 때문에 콘텐츠의 크기가 증가되었다. 그러나 최근 출시되는 안드로이드 단말기는 높은 컴퓨팅 파워와 대용량의 메모리를 갖고

표 8. API 실험 결과

API	시험 프로그램	GNEX C 실행파일 크기 (KByte)	Android 패키지 크기 (KByte)
MMI	00311_MMI	96	245
그래픽	00350_Graphic10	88	246
수학	00276_Math	26	195
파일	00314_File	38	195
하드웨어	00312_HW1	100	258
네트워크	00320_Network	16	197

있기 때문에 충분히 감내해 낼 수 있는 수준이다.

다음은 실험용 게임 콘텐츠와 SKT에서 상용서비스 되고 있는 2개의 게임 콘텐츠를 대상으로 한 실험 결과이다. 그림 10은 "Aiolos", "레이싱 포커", "신맞고"의 3개의 게임을 GNEX 에뮬레이터와 Android 에뮬레이터에서 실행시켜 비교한 모습이다. 그래픽, 이미지 출력, 사운드 출력, 키 이벤트, 타이머 구동, 데이터 저장 등의 동작이 모두 동일하게 실행됨을 확인할 수 있었다.

콘텐츠 실행 속도는 초당 프레임 수(FPS: Frame Per Second)를 측정하여 비교하였다. 그림 11에서와 같이 변환된 WIPI 자바 콘텐츠는 GNEX 콘텐츠와 비슷한 속도를 보여주고 있어 GNEX C-to-Android Java 변환기 시스템이 안정적임을 보여주고 있다.

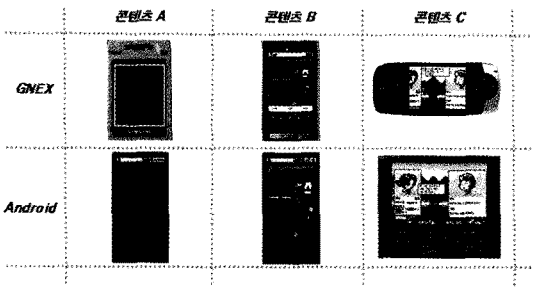
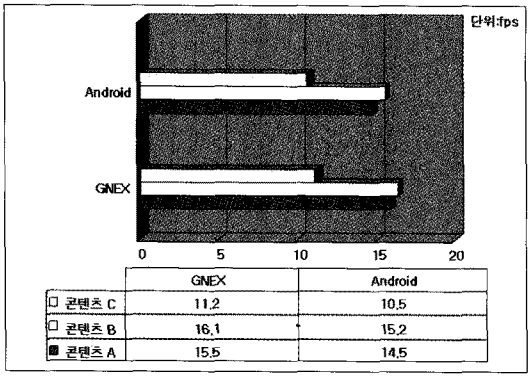


그림 10. GNEX와 Android 플랫폼에서의 콘텐츠 실행결과



	GNEX	Android
콘텐츠 C	11.2	10.5
콘텐츠 B	16.1	15.2
콘텐츠 A	15.5	14.5

그림 11. 콘텐츠 실행속도 비교

5. 결론 및 향후 연구

모바일 게임 콘텐츠 시장은 매년 높은 성장률의 시장 규모를 이어가며 모바일 시장 최고의 킬러 콘텐츠로 자리 잡았다. 하지만 모바일 플랫폼이 달라 하나의 모바일 게임 콘텐츠를 서비스하기 위하여 콘텐츠를 중복 개발하거나 변환해야 하는 문제점이 있다.

본 논문에서 구현한 모바일 콘텐츠 자동 변환기인 GNEX C-to-Android Java 변환기는 이런 문제를 해결할 수 있는 하나의 방법이다. 본 변환기는 시스템 소프트웨어인 컴파일러 제작기술을 활용하여 GNEX 콘텐츠에서 Android 자바 콘텐츠로의 변환 작업이 단기간 내에 자동으로 진행될 수 있도록 하여 GNEX 콘텐츠를 Android 플랫폼으로 이식하기 위한 개발 과정, 변환 과정에 중복 투자되던 시간 및 기간을 상당히 단축함으로써 생산성을 향상시킬 수 있도록 하였다. 또한, 단축된 시간 및 기간을 신규 콘텐츠 개발에 투여함으로써, 우수한 모바일 콘텐츠의 개발이 가속화되어 모바일 산업의 생산력 증강에 이바지할 것으로 기대된다.

앞으로 콘텐츠의 실행 속도를 높이기 위한 연구와 콘텐츠 변환기 시스템의 기능을 보완, 확장하여 기존의 콘텐츠들을 현재 확장되고 있는 스마트폰의 iPhone OS나 Android, Windows Mobile과 같은 다양한 스마트폰 플랫폼에서 실행될 수 있도록 변환기를 확장할 예정이다

참 고 문 헌

[1] 박상훈, 권혁주, 김영근, 이양선, "모바일 콘텐츠의 자동변환 위한 GVM C-to-MIDP 변환기의 설계 및 구현," 한국멀티미디어학회 학회지, Vol. 9, No.2, pp. 215-218, 2006.

[2] 박상훈, 권혁주, 김영근, 이양선, "모바일 게임 콘텐츠를 위한 GVM-to-WIPI 자동 변환기의 설계 및 구현," 한국정보처리학회 게임 논문지, 제3권, 제1,2호, pp. 51-60, 2006.

[3] 이양선, "모바일 콘텐츠의 자동변환을 위한 GNEX C-to-WIPI Java 자동 변환기의 설계 및 구현," 한국멀티미디어학회 논문지, Vol.13, No.4, pp.520-528, Apr 2010.

[4] Mobile C - WIPI C 소스 변환기의 개발, 연구보고서, 동국대학교 산학협력단, 2008.

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

[6] Ed Burnett, Hello, *Android : Introducing Google's Mobile Development Platform*,

Oreilly & Associates Inc, 2008.
[7] 강진영, 정찬성, WIPI GNEX를 이용한 모바일 프로그래밍, 생능출판사, 2006.
[8] 신지소프트, GNEX SDK, http://www.gnex-club.com/download/download_a1.jsp
[9] 신지소프트, Mobile C Library Function Reference, http://www.gnexclub.com/
[10] 신지소프트, Mobile C Programming Guide, http://www.gnexclub.com/
[11] 김미영, 무선 인터넷 서비스를 위한 XML 기반 컨텐츠 변환 시스템의 설계 및 구현, 영남대학교 석사학위논문, 2003.
[12] 김석훈, J2ME MIDP를 이용한 XML 기반의 모바일 컨텐츠 변환 시스템 설계 및 구현, 한남대학교 석사학위논문, 2003.
[13] 김영선, 장덕철, "XML Parser 추출에 의한 모바일 콘텐츠 변환 설계," 멀티미디어학회 논문지, Vol.6, No.2, pp. 267-274, April 2003.
[14] 김은수, 김석훈, 윤성일, "무선인터넷 서비스를 위한 유무선 마크업 언어 간의 컨텐츠 변환 모듈 설계 및 구현," 한국컴퓨터정보학회 논문지, 제9권, 제4호, pp. 149-155, 2004.
[15] 신수원, 최윤석, 정기원, "Mobile C 기반의 GVM 응용프로그램을 Java 기반으로 변환하는 process," 한국정보과학회 학회지, 제29권, 제2호, pp. 103-105, 2002.
[16] 윤성일, Mobile 기반의 유무선 플랫폼 통합 변환 시스템, 한남대학교 박사학위논문, 2003.
[17] 이영종, WIPI와 BREW 플랫폼 간 C 언어 기반 솔루션 변환 방법, 충남대학교 석사학위논문, 2007.
[18] 이양선 외 4인, "모바일 게임 콘텐츠의 자동변환을 위한 GVM-to-BREW 번역기 시스템," 한국정보처리학회 게임 논문지, Vol.2, No.1, pp. 49-64, June 2005.
[19] 이양선, 황대훈, 나승원, "JVM 플랫폼에서 .NET 프로그램을 실행하기 위한 MSIL-to-Bytecode 번역기의 설계 및 구현," 한국멀티미디어학회 논문지, Vol.7, No.7, pp. 976-984, 2004.
[20] 이양선, 나승원, 황대훈, "Intermediate Language Translator for Execution of Java Programs in .NET Platform," 한국멀티미디어학회 논문지, Vol.7, No.6, pp.824-831, June 2004.
[21] Reto Meier, *Professional Android Application Development*, John Wiley & Sons Inc, 2008.
[22] Erik D. Demaine, "C to Java: Converting Pointers into References," *Concurrency: Practice and Experience*, Vol.10, pp. 851-861, 1998.
[23] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin, *Principles of Program Analysis*, Springer, 2005.
[24] A.V.Aho, R.Sethi, J.D.Ulman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, 2007.
[25] 오세만, 컴파일러 입문, 정익사, 2006.
[26] 한국 무선인터넷 표준화 포럼, 모바일 표준 플랫폼 규격 WIPI V1.2, http://www.kwisforum.org/file/public_pds/KWISFS.K-05-001R2.pdf, 2003.
[27] 황기태, 김남윤, 위피 자바, Jlet 모바일 프로그래밍, 생능출판사, 2009.

손 윤 식

2004년 동국대학교 컴퓨터공학과 공학사
2006년 동국대학교 컴퓨터공학과 공학석사
2009년 동국대학교 컴퓨터공학과 공학박사
2010년~현재 동국대학교 전문연구원

관심분야 : 프로그래밍 언어, 컴파일러, 모바일/임베디드 컴퓨팅, 로봇 소프트웨어, 시큐어 코딩

오 세 만

1977년 서울대 수학교육학과 졸업
1979년 한국과학기술원 전산학과 석사 졸업.
1985년 한국과학기술원 전산학과 박사 졸업.
1985년~현재 동국대학교 컴퓨터공학과 교수.
1993년 3월~1999년 2월 동국대학교 컴퓨터공학과 대학원 학과장.
2001년 11월~2003년 11월 한국정보과학회 프로그래밍 언어연구회 위원장.
2004년 6월~2005년 11월 한국정보처리학회 게임연구회 위원장.

관심분야 : 프로그래밍 언어, 컴파일러, 모바일 컴퓨팅, 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, 게임엔진 등