

■ 2010년도 학생논문 경진대회 수상작

모바일 게임을 위한 번역 기법 (Translation Techniques for Mobile Games)

박 지 우 [†] 오 세 만 [‡]
(Jiwoo Park) (Seman Oh)

요약 하나의 콘텐츠를 다양한 플랫폼에서 서비스하는 것은 모바일 게임 시장에서 매우 중요한 사항이다. 서로 다른 플랫폼에서 서비스하기 위해서는 기존의 콘텐츠를 특정한 플랫폼에 적합한 형태로 재개발해야하는 추가 비용이 발생한다. 더욱이, 새로운 플랫폼이 등장하는 경우에 다수의 콘텐츠를 단기간에 개발하는 것은 많은 노력과 비용을 요구한다. 따라서 기존의 콘텐츠를 새로운 플랫폼에서 실행될 수 있도록 변환하는 기법에 관한 연구가 필요하다.

본 논문에서는 모바일 콘텐츠를 새로운 플랫폼에서 실행 가능한 형태로 변환하는 모바일 게임을 위한 번역 기법을 제안하고 이를 적용하여 안드로이드 플랫폼에 대한 소스 레벨 콘텐츠 변환기를 설계하고 구현한다. 컴파일러 이론을 적용하여 개발하였으며, 서로 다른 프로그래밍 언어 간의 변환 문제를 부분적으로 해결하였다. 또한 언어 간의 번역뿐만 아니라 커널 이식, 라이브러리 매핑 등 체계적으로 소스 레벨 콘텐츠 변환도구를 구현하였다. 이러한 연구는 기존 모바일 게임에 대한 활용도를 증대시키고 모바일 콘텐츠 산업 활성화에 기여할 수 있을 것으로 기대된다.

키워드 : 콘텐츠 변환기, 모바일 게임, 안드로이드, 언어 번역

Abstract In the mobile gaming market, it is desirable for a specific content to be served on various platforms. Recently, it is a very important issue in the market for mobile games. This problem incurs extra cost, because we need to redevelop existing contents to be executed on another platforms. Moreover, the release of a new platform has spent much more effort and cost developing many contents in short period of time. Therefore, we need to research about translation techniques that enable the existing contents to run on a different platform.

In this paper, we propose translation techniques for mobile games. The techniques can be applied on converting mobile contents to runnable contents on a new platform. To realize it, we design and implement a source-level contents translator which is targeting Android platform using the suggested techniques. Our source-level contents translator is implemented systematically by applying compiler theory. Also, we solve partially translation problems between different programming languages. The translator has been implemented separately divided into three tasks such as kernel porting, library mapping as well as language translation. We expect these techniques to increase utilization of existing mobile games and contribute to vitalizations of mobile contents industry.

Key words : Contents Translator, Mobile Game, Android, Language Translation

[†] 학생회원 : 동국대학교 컴퓨터공학과

jojaryong@dongguk.edu

[‡] 종신회원 : 동국대학교 컴퓨터공학과 교수

smoh@dongguk.edu

(Corresponding author)

논문접수 : 2010년 4월 26일

심사완료 : 2010년 7월 9일

Copyright©2010 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

정보과학회논문지 : 컴퓨팅의 실제 및 레터 제16권 제9호(2010.9)

1. 서 론

모바일 산업은 통신 서비스와 단말기 개발, 콘텐츠 제작 등으로 분류할 수 있으며, 최근까지 하드웨어와 소프트웨어 측면에서 지속적인 성장을 하고 있다. 특히, 모바일 게임 시장은 2009년 기준으로 전 세계 게임 시장에서 점유율 8.4%, 매출액 71억 달러가 예상되는 유망한 분야이다[1].

모바일 게임 시장의 특징은 여러 개의 유사한 모바일

플랫폼이 존재하기 때문에 하나의 콘텐츠를 성공적으로 서비스하기 위해서는 모든 플랫폼에서 서비스할 수 있도록 콘텐츠 변환 작업을 거쳐야 한다는 것이다. 또한, 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 모바일 표준 플랫폼으로 이식할 필요성이 제기되고 있다. 다행히도 모바일 플랫폼은 프로그래밍 언어가 같지 않고, 실행 모델이 다르더라도 동일한 휴대폰 기기를 바탕으로 하기 때문에 기능적으로는 동일한 구성 요소로 이루어져 있다. 따라서 타 플랫폼으로의 콘텐츠 변환 작업은 플랫폼 간에 서로 대응되는 구성 요소에 대한 매핑을 통해 해결할 수 있다.

본 논문에서는 컴파일러 이론을 적용하여 모바일 게임을 소스 레벨에서 변환하는 기법을 제안한다. 또한, 제안된 기법을 바탕으로 이기종간의 모바일 게임 변환을 용이하게 하기 위한 도구인 소스 레벨 변환기를 설계하고 구현한다. 원활한 연구 진행을 위해 다수의 콘텐츠를 보유하고 있는 GNEX 플랫폼용 콘텐츠를 안드로이드 플랫폼용 콘텐츠로 번역하는 것으로 연구 범위를 제한한다. 변환기는 기능별로 독립적인 모듈로 설계하며, 모듈간의 상호 호환성을 고려한다. 크게 3개의 변환 작업으로 구성되며 언어 번역(Language translation), 커널 이식(Kernel porting) 그리고 라이브러리 매핑 등이다.

언어 번역은 컴파일러 기법을 적용하여 소스 언어에 대한 AST를 생성하고, 타겟 언어를 위한 어노테이티드-AST로 변환한 후, 최종적으로 타겟 언어로 번역한다. 특히, 언어 번역에서는 자료형 번역과 클래스 구성 방법론, 포인터 처리 방법론을 중점적으로 다룬다. 커널 이식은 타겟 플랫폼에 적합하게 실행 모델을 변환하는 작업이며, 플랫폼의 이벤트 처리 방법과 함께 시스템 구조와 밀접한 관련이 있는 그래픽 엔진의 이식 방법, 터치 인터페이스 처리, 내장 폰트 표현 방법 등을 제안한다. 라이브러리 매핑은 소스 언어에서 제공한 API를 대응되는 타겟 언어의 API로 변환하는 작업이며, 단순 매핑과 복합 매핑 등으로 구분하여 처리한다.

제안된 방법론을 적용하여 GNEX(General & NEXt multimedia player)-안드로이드(Android) 변환기를 개발하였으며, 실제로 GNEX 용으로 개발된 많은 상용 게임 콘텐츠를 안드로이드 플랫폼에서 서비스할 수 있는 콘텐츠로 자동 변환하였다.

이와 같은 방법론은 자동 변환을 통하여 기존의 콘텐츠를 새로운 플랫폼에서 서비스할 수 있는 환경을 구축

하여 다양한 콘텐츠를 제공할 수 있게 하며, 따라서 모바일 게임 개발의 부담을 줄이고 이미 개발된 휴대폰 콘텐츠의 활용 폭을 확대할 수 있다. 더 나아가, 휴대폰 콘텐츠 산업 활성화에 크게 기여할 수 있을 것으로 판단된다.

2. 관련연구

2.1 GNEX

GNEX는 GVM(General Virtual Machine)을 기반으로 업그레이드된 모바일 C 언어를 위한 멀티미디어 엔진이다[2]. GNEX는 모듈이 탑재된 단말기와 GNEX 응용 프로그램 개발을 위한 GNEX SDK, 네트워크 통신 기능을 수행하는 GNEX 서버 기술이 결합된 시스템으로 구성된다. 초기 GVM 서비스와의 차이점을 살펴보면 GVM은 3가지 각각의 분야에 대한 규격을 정의하고 기능을 제공하나, GNEX는 위피와 같은 플랫폼의 등장으로 응용 프로그램 다운로드 및 관리 기능은 플랫폼 기능을 활용하고 응용 프로그램 개발을 위한 개발 도구 및 단말기의 실행 환경에 대한 기능만을 제공하고 있다.

GNEX는 안정성과 빠른 속도를 구현할 수 있는 계층 구조를 가졌으며, 가상기계, 커널, 응용 프로그램 관리자로 구성되어 있다. 가상기계는 콘텐츠 해석 모듈이며, 게임/멀티미디어에 특화되어 빠르고 안정적인 실행 환경을 제공한다. 커널은 다양한 시스템 인터페이스를 제공하며 메모리 관리자 탑재로 시스템을 보호한다. 마지막으로 응용 프로그램 관리자는 다운로드 된 콘텐츠를 로딩하며, 단순하고 구조적인 설계로 최적화된 성능을 보장한다. 그림 1은 GNEX의 시스템과 위피 플랫폼 연동 모델을 표현한 것이다.

GNEX는 모바일 C 기반의 32bit 가상기계를 채용, 다양한 최적화 기법 적용, 고속 이미지 출력 알고리즘을 적용해 빠른 실행속도를 보여준다. 또한, 다양한 표현 능력의 2D Effect, OpenGL 3D 그래픽 라이브러리를 사용하여 화려한 그래픽 구현이 가능하다. GNEX 콘텐츠 내에서 SIS 애니메이션 재생이 가능하며, 게임과 동영상을 오가는 멀티미디어 콘텐츠 구현으로 다양한 멀티미디어 솔루션 연동이 가능하다.

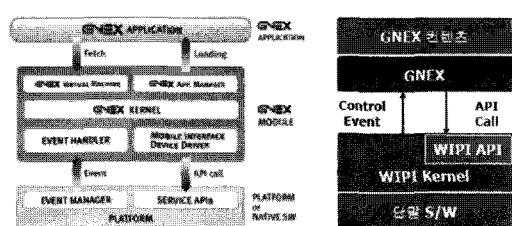


그림 1 GNEX 시스템과 위피 플랫폼 연동 모델

2.2 안드로이드

안드로이드는 구글에서 개발한 운영체제, 미들웨어, 응용 프로그램을 포함하는 모바일 기기에 최적화된 플랫폼이다[3]. 안드로이드는 오픈 소스 정책을 채택하였으며, 리눅스 커널, 라이브러리, 런타임, 애플리케이션 프레임워크, 애플리케이션으로 구성되어 있다. 그림 2는 안드로이드 플랫폼의 구조를 그림으로 나타낸 것이다.

리눅스 커널은 보안, 메모리 관리, 프로세스 관리, 네트워크 스택, 드라이버 모델과 같은 리눅스 버전 2.6의 핵심 시스템 서비스를 이용하며, 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.

안드로이드의 응용 프로그램은 그림 3과 같이 생성(OnCreate), 시작(OnStart), 재개(OnResume), 일시 정지(OnPause), 정지(OnStop), 파괴(OnDestroy)의 6개 상태 전이 사이클을 갖도록 구성되어 있다.

자바로 작성된 안드로이드 응용 프로그램은 Dalvik 가상기계에 의해 실행된다. Dalvik 가상기계는 레지스터 기반의 가상기계로 제한된 메모리에 최적화되었다. Dalvik 가상기계는 DEX(Dalvik Executable File) 파일을 실행 한다. DEX 파일은 저장 공간과 메모리 사이의 효율적인 매핑 정의에 최적화된 포맷이다. 안드로이드 프로그램은 자바 컴파일러에 의해 클래스 파일로 컴파일된 후,

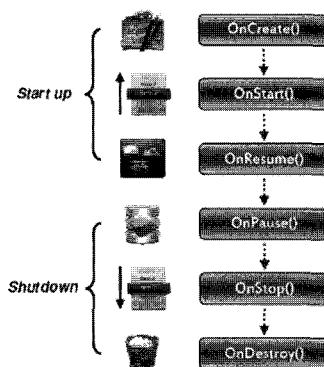


그림 3 안드로이드 응용 프로그램의 사이클

클래스 파일과 안드로이드 바이트 코드 사이의 마이그레이션(Migration)을 통해 DEX 파일로 변환된다.

2.3 GVM 콘텐츠의 BREW 콘텐츠 자동 변환 도구

GVM 콘텐츠의 BREW 콘텐츠 자동 변환 도구는 GVM의 소스 코드, 이미지, 사운드 데이터를 BREW의 소스 코드, 이미지, 사운드와 추가적인 데이터로 변환하는 도구이다[4]. GVM2BREW 자동변환 도구는 총 6개의 변환 도구로 이루어져 있으며, 그림 4는 이를 도식화한 것이다.

개별 변환도구에 의해 GVM 게임을 BREW에 대응되는 코드 및 데이터로 변환한다. 자동변환도구의 개발로 인해 개발자가 직접 콘텐츠를 변환하는 것과 비교하여 콘텐츠 변환 시간을 단축할 수 있으며, 결과물에 대한 안정성과 신뢰성을 보장할 수 있다. 또한, 결과물은

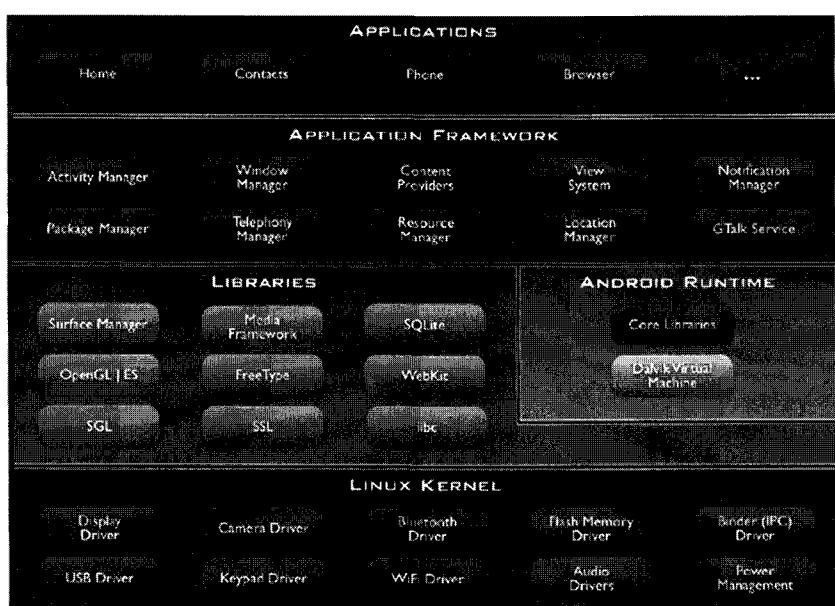


그림 2 안드로이드 플랫폼의 계층 구조

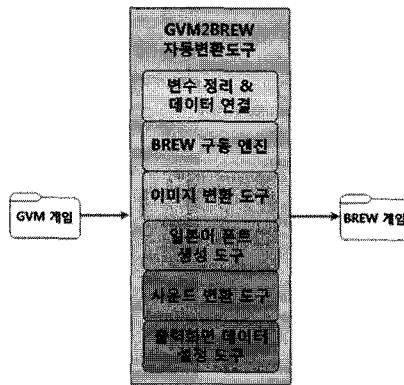


그림 4 GVM2BREW 자동변환도구 시스템 구성도

KDDI-BREW 플랫폼에서 실행되기 때문에 수출을 위한 기술적 교류보를 마련할 수 있다. 그러나 GVM2BREW는 자동 변환 도구가 개별 도구로 분리되어 있기 때문에 변환 작업에 사용자의 개입에 의해 처리되는 부분이 존재한다. 또한, 타겟 언어의 변경에 대한 고려가 없기 때문에 활용 폭이 제한적이다.

2.4 GNEX - WIPI 소스 변환기

GNEX - WIPI 소스 변환기는 모바일 C로 작성된 GNEX 콘텐츠를 WIPI 콘텐츠로 변환하며, AST 생성기, AST 변환기, HLL 생성기 모듈로 구성되어 있다. AST 생성기는 HLL 소스를 처리하여 AST를 생성하는 모듈로서 어휘 분석기와 구문 분석기로 이루어져 있다. 어휘 분석기는 HLL 소스의 토큰을 분리하여 구문 분석기에 전달하며, 구문 분석기는 HLL 소스의 문법을 체크하고 AST를 생성하는 역할을 한다. AST 변환기는 소스를 위한 AST를 운행하여 HLL 타겟을 위한 AST를 생성한다. 이때 생성된 AST는 ANSI C의 AST 형태를 가진다. HLL 생성기는 타겟을 위한 AST와 전처리 문장, 매핑 정보를 이용하여 원하는 플랫폼에서 실행될 수 있는 언어 형태로 변환한다. 전처리 문장은 HLL 소스의 상수 및 타입 정의, 함수 원형 선언 등의 정보를 통하여 생성되는 데이터이다. 매핑 정보는 HLL 소스 언어와 목적 언어 간의 언어적 특성을 극복할 수 있는 정보의 표현이다. 정보는 HLL 소스 언어와 목적 언어 간의 변환 정보와 상호 라이브러리 매핑 관계를 표현하는 정보로 이루어져 있다. GNEX - WIPI 소스 변환기는 그림 5와 같은 구조를 가진다.

3. 모바일 게임 번역 기법

현재까지의 콘텐츠 번역을 위한 연구는 이미지 포맷 등의 멀티미디어 데이터의 변환 및 처리가 제한적이었으며, 언어 번역, 커널 이식, 라이브러리 매핑을 구분하지 않고 변환기에서 모두 처리하였다. 따라서 다양한 번

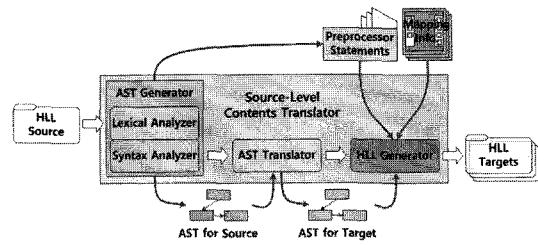


그림 5 GNEX - WIPI 소스 변환기

역 패턴을 모두 고려하여 처리하기에는 어려움이 있었다. 또한 번역 과정에서 사용자의 개입이 요구되었다[4].

본 논문에서는 제시하는 번역 기법은 GNEX 플랫폼에서 동작하는 콘텐츠를 안드로이드 플랫폼에서 실행 가능하도록 번역하는 것을 목표로 한다. 즉, GNEX 플랫폼의 개발 언어인 모바일 C 언어를 안드로이드 플랫폼의 개발 언어인 자바로 번역하는 것으로 정의할 수 있다. 모바일 게임 변환 작업은 크게 언어 번역, 커널 이식, 그리고 라이브러리 매핑의 세 부분으로 구성된다. 언어 번역은 소스 프로그램을 동등한 의미를 갖는 타겟 프로그램으로 번역하는 것이며, 커널 이식은 소스 콘텐츠의 실행 모델을 타겟 플랫폼의 실행 모델로 변환하는 작업을 의미한다. 그리고 라이브러리 매핑은 소스 언어에서 제공하는 API를 대응되는 타겟 언어의 API로 변환하는 작업이다.

언어 번역 부분에서는 C 언어와 자바 언어의 번역 시 필수적으로 해결해야 할 자료형 번역 방법, 클래스 생성 방법, 포인터 처리 방법을 설명한다. 커널 이식 부분에서는 GNEX에서 사용하는 독창적인 그래픽 엔진, 이미지 포맷, 이벤트, 내장 폰트 등을 안드로이드 플랫폼용 콘텐츠로 번역하기 위한 방법론을 설명한다. 마지막으로 라이브러리 매핑 부분에서는 GNEX의 라이브러리를 안드로이드의 라이브러리 형태로 구성하는 방법을 설명한다.

GNEX 콘텐츠는 본 논문에서 구현한 소스 레벨 콘텐츠 번역기를 통해 안드로이드 콘텐츠로 자동 번역되며, 이는 안드로이드 플랫폼의 콘텐츠 형태로 이식된 커널과 라이브러리의 결합을 통해 실행된다. 그림 6은 GNEX 콘텐츠가 안드로이드 콘텐츠로 변환되어 실행되는 과정을 개괄적으로 보여준다.

3.1 언어 번역

언어 번역은 서로 다른 언어 간의 차이점을 해결하여, 동등한 의미를 갖는 서로 다른 언어로의 번역을 의미한다. 본 논문은 이기종간의 모바일 콘텐츠 변환을 위한 번역 기법을 제시하고 이를 적용한 소스 레벨 콘텐츠 번역기를 구현하는 것이며, 입력으로 GNEX 용 모바일 C 프로그램을 받으며, 출력으로는 동등한 의미를 가지는 안드로이드 용 자바 프로그램을 생성하는 것이다. 본

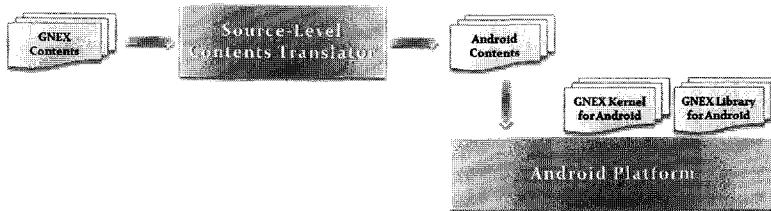


그림 6 GNEX 콘텐츠의 변환 및 실행 과정

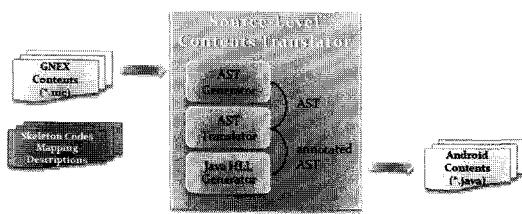


그림 7 소스 레벨 콘텐츠 번역기 구성도

논문에서의 번역 기법은 제한된 리소스를 갖는 환경을 고려하며, C 언어와 자바 언어 번역에 대한 연구 범위를 자료형 번역, 클래스 분류 방법, 포인터 처리의 3가지로 부분으로 제한하여 해결한다. 자료형 번역은 기본형과 사용자 자료형으로 구분하여 해결하였으며, 클래스 분류 과정에서 분석 비용이 많이 소모되는 함수간의 결합도는 연구 대상에서 제외한다.

소스 레벨 콘텐츠 번역기는 컴파일러 이론을 통한 체계적인 소스 레벨 언어 번역 방법론을 적용하여 모듈 간의 상호 의존성을 배제시킬 수 있도록 설계하였으며, 기능별로 독립적인 모듈로 구현하였다. 그림 7은 번역기의 기능을 그림으로 도식화한 것이다.

AST 생성기는 모바일 C 소스 코드를 처리하여 AST를 생성하는 모듈로서 컴파일러의 전단부에 해당하는 어휘 분석기와 구문 분석기로 구성된다. 어휘 분석기는 소스 코드의 토큰을 분리하여 구문 분석기에 전달하며, 구문 분석기는 소스 코드의 구문 구조를 체크하여 AST를 생성하는 역할을 한다. AST 번역기는 소스를 위한 AST를 운행하여 타겟 언어로의 번역이 용이한 형태의 어노테이티드-AST를 생성한다. 자바 HLL 생성기는 어노테이티드-AST와 골격 코드(Skeleton codes)를 이용하여 안드로이드용 자바 소스 파일을 생성한다.

모바일 C를 자바로 번역하는 과정에서는 중요한 2가지 사항을 고려해야 한다. 첫 번째는 모바일 C의 자료형 중 자바에서 제공하지 않는 자료형을 번역하는 것이고, 두 번째는 전역 변수 선언, 사용자 정의 자료형, 함수 정의로 구성된 모바일 C 프로그램을 클래스 기반 구조의 자바 프로그램으로 번역하는 방법론을 제시하는 것이다.

첫 번째로 자료형 번역시에는 모바일 C와 자바에서

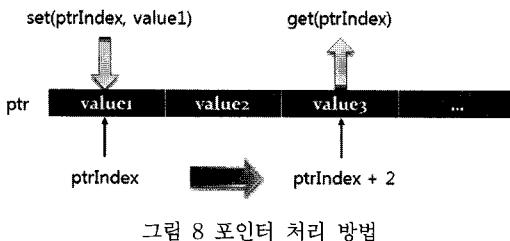
표 1 모바일 C와 자바의 자료형 대응 관계

모바일 C	자바	비고
struct	클래스	
enum	상수 필드	public static final int
string	StringBuffer 클래스	
image	GnexImage 클래스	사용자 정의 클래스
sound	GnexSound 클래스	사용자 정의 클래스

공통적으로 제공하는 기본형 이외에 구조체와 열거형, 미디어형에 대한 변환 방법을 제시해야 한다. 표 1은 자료형 대응 관계를 나타낸 것이다.

모바일 C의 자료형 중 포인터는 언어적인 측면에서 매우 민감한 요소이다. 자바에서는 포인터를 제공하고 있지 않기 때문에 모바일 C를 자바로 번역 시 포인터 처리 방법론을 마련하는 것이 큰 비중을 차지한다고 할 수 있다. 기존에 제시된 포인터 처리 방법론[5]은 많은 객체와 메모리 공간을 사용하고, 변수 참조에 대한 접근 횟수가 큰 폭으로 증가하기 때문에 제한된 리소스를 갖는 모바일 환경에 적용하기에 어려움이 있다. 따라서 본 논문에서는 모바일 환경에 적합한 포인터 번역 방법을 제안하고 이를 적용한다.

포인터 번역 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 애플레이션하고 포인터 변수와 포인터를 참조하는 변수에 대한 정적 분석[6]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 *ArrayList*와 같은 클래스를 사용하는 방법이 있다. GNEX-안드로이드 소스 번역기에서는 속도와 메모리 공간 문제를 고려하여 인덱스를 위한 배열과 *ArrayList* 클래스를 사용하여 포인터 문제를 해결한다. 그림 8은

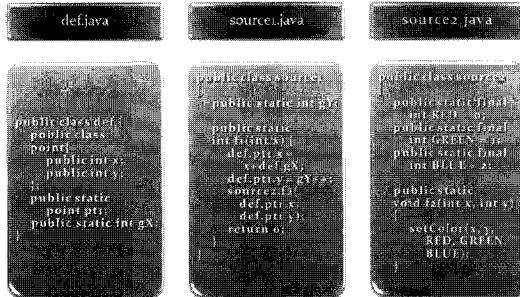
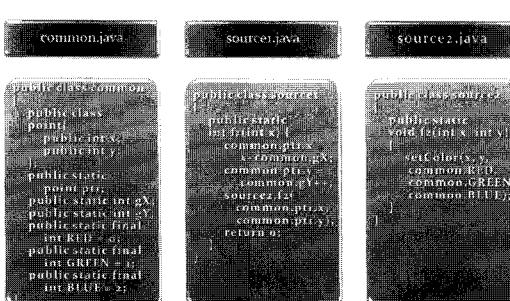
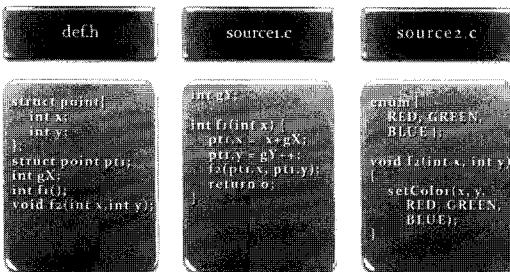


포인터 처리 방법론을 그림으로 도식화한 것이다.

두 번째 번역 작업은 C 언어의 전역 변수 선언, 사용자 정의 자료형, 함수 정의 등을 클래스의 구성 요소인 필드와 메소드로 번역하는 작업을 의미한다. 따라서 언어 변환 시 클래스 생성 방법론에 대한 연구가 필요하다. 다양한 측면에서 연구를 거듭한 결과, C를 자바로 변환 시 두 가지 형태를 고려할 수 있다. 그림 9는 두 가지 클래스 변환 방법론을 설명하기 위해 작성한 예제 코드이다.

예제 코드는 총 3개의 파일로 구성되어있다. *def.h* 파일은 C 언어의 헤더 파일로서, 구조체와 전역 변수, 함수 원형이 정의되어있다. *source1.c*와 *source2.c*는 전역 변수, 열거형, 함수 정의로 구성되어 있다.

첫 번째 방법론은 C 언어의 소스 파일별로 자바 파일을 생성하고 전역 변수, 타입 정의에 대한 공용 자바 파일을 생성하는 것이다. 그림 10은 첫 번째 방법론에 의



해 생성되는 클래스를 그림으로 표현한 것이다. 하나의 소스 파일이 하나의 자바 파일로 생성되었으며, 전역 변수로 선언된 데이터 부분은 별도의 파일로 생성된다.

두 번째 방법론은 헤더 파일을 포함한 모든 모바일 C 소스 파일을 별도의 자바 파일로 생성하는 것이다. 그림 11은 두 번째 방법론에 의해 생성된 클래스를 그림으로 표현한 것이다. 헤더 파일을 포함한 각각의 소스 파일이 단순히 하나의 자바 파일로 생성된다.

두 가지 방법론을 검토한 결과, 모바일 C 프로그램에서 전역 변수 선언과 사용자 정의 자료형의 선언 등의 데이터 부분은 다른 소스 파일에서 사용할 가능성이 높은 것으로 판단할 수 있다. 따라서 공용 데이터를 모아서 별도의 자바 파일로 관리하는 첫 번째 방법론이 프로그램의 가독성과 유지, 보수 측면에서 효과적이다. 본 논문에서는 첫 번째 방법론을 적용하여 소스 레벨 콘텐츠 변환기를 구현한다.

3.2 커널 이식

커널 이식은 콘텐츠의 제어 구조를 변환하는 것을 의미한다. 즉, GNEX-안드로이드 번역 기법에서 커널 이식은 GNEX 콘텐츠의 제어 구조를 대응되는 안드로이드 콘텐츠의 제어 구조로 변환하는 것을 의미한다. GNEX 커널의 안드로이드 플랫폼 이식 과정을 본 논문에서는 이벤트 처리 방식 변환 방법, GNEX의 그래픽 엔진 및 이미지 포맷 포팅, 터치 인터페이스 처리, 내장 폰트 표현 방법으로 구분하여 설명한다. 그림 12는 GNEX와 안드로이드 콘텐츠의 동작 방법을 비교한 것이다.

GNEX와 안드로이드 콘텐츠는 유사한 구조의 동작 방식을 따른다. GNEX는 GNEX 커널에서 해당 콘텐츠의

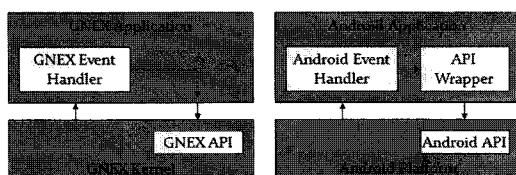


표 2 GNEX와 Android 이벤트

GNEX 이벤트	Android 이벤트	비고
EVENT_START	onStart()	
EVENT_KEYPRESS	onKeyDown(), onKeyUp()	
EVENT_GENERIC	onTouchEvent()	
EVENT_TIMEOUT		스레드로 구성
EVENT_NETWORK		스레드로 구성

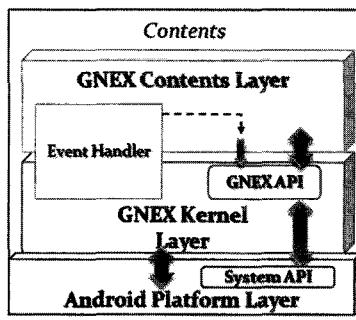


그림 13 GNEX 커널 포팅 방법

이벤트 처리기를 호출하고 안드로이드에서는 안드로이드 플랫폼이 GNEX 커널과 유사한 작업을 수행한다. GNEX는 GNEX 커널에서 모든 이벤트를 해석하여 콘텐츠의 이벤트 처리기를 호출하는 구조를 가진다. 안드로이드는 플랫폼에서 발생하는 시스템 이벤트에 대해 콘텐츠의 콜백 메소드를 호출하여 처리한다. 단, 타이머의 경우, 프로그래머가 별도의 스레드를 생성하여 콘텐츠에서 처리해야 한다. GNEX 커널을 안드로이드 플랫폼에 이식하는 기법을 그림으로 도식화하면 그림 13과 같다.

GNEX에서 발생하는 주요 이벤트는 EVENT_START, EVENT_KEYPRESS, EVENT_TIMEOUT, EVENT_GENERIC, EVENT_NETWORK 가 있다.

EVENT_TIMEOUT과 EVENT_NETWORK 이벤트는 GNEX의 타이머와 네트워크 이벤트를 에뮬레이션 할 수 있는 자료 구조와 이벤트 처리 메소드를 정의하고 안드로이드의 스레드와 연계해서 동작하도록 작성해야 한다.

커널 이식 중 이벤트에 대한 처리는 결국 대상 플랫폼의 모든 이벤트를 처리할 수 있는 템플릿 소스를 구현하는 것을 의미한다. GNEX의 콘텐츠를 안드로이드 콘텐츠로 수동 변환하여 분석된 구조를 토대로 골격 프로그램을 구현하였으며, 골격 프로그램 구현에 적용된 GNEX의 이벤트 목록은 표 3과 같다.

GNEX에서 터치 스크린과 관련된 이벤트는 EVENT_GENERIC 이벤트이다. 터치 이벤트를 처리하기 위해 안드로이드의 doTouch 이벤트로부터 콘텐츠의 EVENT_GENERIC() 이벤트 처리기를 호출하도록 구성한다. 또한, 높은 해상도를 가지는 안드로이드 플랫폼 단말기

표 3 골격 프로그램에 적용된 GNEX의 이벤트

이름	설명
EVENT_START	시작 이벤트
EVENT_KEYPRESS	키 입력 이벤트
EVENT_TIMEOUT	타임 아웃 이벤트
EVENT_GENERIC	터치 이벤트
EVENT_NETWORK	네트워크 이벤트
Timer 처리 루틴	타이머

```
boolean doTouch(MotionEvent event)
{
    int x = (int)event.getX();
    int y = (int)event.getY();
    switch (event.getAction())
    {
        case MotionEvent.ACTION_DOWN:
            // ...
            break;
        case MotionEvent.ACTION_UP:
            // ...
            break;
        case MotionEvent.ACTION_MOVE:
            // ...
            break;
    }
    return true;
}

void EVENT_GENERIC()
{
    int x = swData0;
    int y = swData1;

    switch (swData2)
    {
        case GNEX_POINTER_PRESS:
            // ...
            break;
        case GNEX_POINTER_DCLICK:
            // ...
            break;
        case GNEX_POINTER_MOVE:
            // ...
            break;
        case GNEX_POINTER_RELEASE:
            // ...
            break;
    }
}
```

그림 14 안드로이드 doTouch 이벤트와 GNEX 터치 이벤트

를 고려하여 확대된 화면에 대응하기 위해 물리 좌표계와 논리 좌표계의 변환을 고려한다. 그림 14는 안드로이드 doTouch 이벤트와 GNEX 터치 이벤트의 구성한 예이다.

안드로이드 플랫폼에서는 터치 스크린 이벤트로 Down, Up, Move 이벤트를 제공한다. 그러나 GNEX에서 제공하는 더블 클릭 이벤트는 명시적으로 제공하지 않기 때문에 Down 이벤트와 Up 이벤트를 조합하여 더블 클릭 이벤트를 에뮬레이션할 수 있도록 구성해야 한다.

GNEX는 별도의 그래픽 포맷(VDI format)과 그래픽 엔진을 가지고 있다. 따라서 GNEX의 그래픽 라이브러리는 GNEX의 커널 포팅과 깊은 연관성을 가진다. 그래픽 라이브러리는 108개로 구성되어 있으며, 이는 GNEX의 게임 콘텐츠에서 사용빈도가 매우 높다. 그래픽 라이브러리를 구현하기 위해서 GNEX의 그래픽 포맷과 엔진을 에뮬레이션할 수 있는 GNEX 이미지를 위한 클래스를 구현한다.

GNEX 이미지 클래스는 핫 스팟(Hot Spot)을 나타내는 (x, y) 좌표 값과 이미지의 너비와 높이, 팔레트 정보,

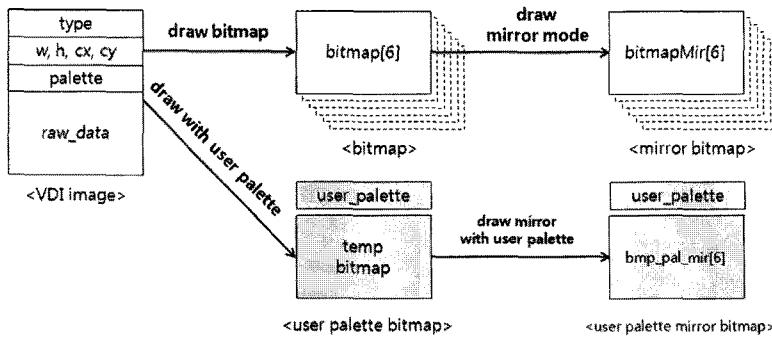


그림 15 이미지 클래스와 이미지 처리 방법

가공되지 않은 이미지 정보로 구성된다. 안드로이드 플랫폼은 그래픽 라이브러리로 이미지 그리기, 회전 등을 제공한다. 그림 15는 이미지 클래스의 구조와 그래픽 라이브러리에 따른 처리 방법을 그림으로 표현한 것이다.

이미지의 가공되지 않은 데이터는 GNEX 컨텐츠 변환시 안드로이드의 리소스 파일 형태로 생성된다. 생성된 이미지 데이터는 안드로이드 컨텐츠의 실행 시점에 메모리에 로딩된다. 이미지 그리기, 회전 등의 기능은 안드로이드에서 제공하는 라이브러리로 구성할 수 있지만, 팔레트 변경, 미러 이미지 등을 적용하기 위해서는 픽셀 단위 연산에 의해 처리할 수밖에 없다. 그러나 콘텐츠에서 이미지를 그리는 시점에서 픽셀 연산에 의해 처리하는 것은 심각한 속도 저하를 초래할 가능성이 존재한다. 따라서 속도에 많은 영향을 미칠 수 있는 이미지 데이터를 미리 생성하여 성능을 높이는 방법을 채택한다. 최초에는 VDI 이미지로부터 프레임에 따라 최대 6개의 비트맵 이미지가 생성된다. 추가적으로 미러 기능과 사용자 팔레트 기능을 사용하는 경우 해당 비트맵을 동적으로 생성하고 메모리 상황에 따라 제거하도록 처리하여 속도와 메모리 사용량을 효과적으로 개선한다.

GNEX의 폰트는 KSC5601 표준을 사용하고 있으며, KSC5601 기준으로 ASCII 128자, 한글 2,350자, 심볼 1,115자가 정의되어 있다. 또한, 폰트의 크기로 Small, Medium, Large, Huge, Double의 5가지 크기를 제공한다. 안드로이드에서 제공하는 폰트를 사용할 경우, GNEX 환경에서 실행되는 컨텐츠와 서로 다른 모양의 문자열이 표시된다. 따라서 안드로이드로 변환된 컨텐츠에서 동일한 모양과 크기의 문자열을 출력하기 위해서는 GNEX의 폰트 데이터를 내부적으로 포함해야 한다. GNEX의 폰트 데이터의 크기는 약 155KB이며 Huge를 제외하면 약 56KB이다. Double 폰트는 Large 폰트를 단순히 2배 확대한 것이므로, 별도의 폰트 데이터를 요구하지 않는다. 표 4는 GNEX의 폰트 데이터의 크기를 나타낸 것이다.

표 4 GNEX의 폰트 데이터의 크기(단위: KByte)

폰트 종류	ASCII	한글	심볼	Total
Small(4x6)	0.3	-	-	0.3
Medium(6x8)	0.6	-	-	0.6
Large(6x12)	0.9	36.7	17.4	55.0
Huge(8x16)	1.8	66.5	31.6	99.9
Total	3.6	102.2	49.0	154.8

3.3 라이브러리 매핑

라이브러리 매핑은 GNEX에서 제공하는 라이브러리를 안드로이드의 라이브러리를 이용하여 구성하는 것을 의미한다. GNEX는 16개의 그룹으로 분류된 425개의 라이브러리를 제공한다. 컨텐츠 동작에 필요한 주요 함수는 8개 그룹에 포함된 237개의 함수이며, 대부분 그래픽, 핸드셋 콘트롤 라이브러리가 이에 해당된다. 그림 16은 GNEX에서 제공하는 라이브러리 카테고리를 나타낸다[7].

안드로이드 라이브러리는 총 122개의 패키지로 나누어져 있으며, 2,254개의 클래스에서 총 17,018개의 메소드를 제공한다.

GNEX와 안드로이드 라이브러리 매핑 시 주요 해결

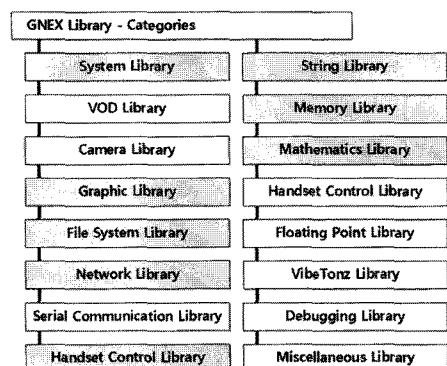


그림 16 GNEX의 라이브러리 카테고리

사항은 안드로이드 소스 코드로 GNEX 라이브러리를 구조화하는 방법을 결정하는 것과 라이브러리 매핑 관계를 정의하는 것이다. GNEX 라이브러리를 안드로이드 소스 코드로 구성 시 GNEX 라이브러리 카테고리별로 별도의 클래스를 구성하고 각 클래스의 모든 필드와 메소드는 public static 속성을 갖도록 한다. 또한 클래스 간의 공유 데이터가 발생하면, 이를 처리하기 위한 공용 클래스 작성하도록 설계한다.

GNEX와 안드로이드 라이브러리 매핑 관계에는 단순 매핑과 복합 매핑이 존재하며, 매핑이 불가능한 라이브러리가 존재할 수 있다. 단순 매핑은 반환 값, 매개변수, 함수의 의미가 모두 유사한 경우로서 대부분 1:1 매핑이 가능하다. 1:1 라이브러리 매핑 단계에서는 실 매개변수, 반환 값의 의미 차이로 인해 데이터를 공유해야 하는 경우가 있을 수 있다. 복합 매핑은 하나의 라이브러리가 여러 개의 라이브러리로 표현되는 경우이며, 1:N 매핑이 가능하다. 그림 17은 라이브러리 매핑 관계를 그림으로 표현한 것이다.

매핑이 불가능한 라이브러리는 알고리즘을 직접 구현해서 제공할 수 있다. 표 5는 GNEX 라이브러리의 그룹별 매핑 관계를 클래스 수준에서 분석한 결과이다.

매핑 관계에 대한 분석 자료를 근거로 단순 매핑이 가능한 라이브러리를 파악할 수 있다. 단순 매핑은 1:1로 완전히 매핑되는 경우와 표 6과 같이 매개변수와 반환 값의 변환이 필요한 경우로 구분됨을 알 수 있다. 1:1 매핑 구조를 갖는 함수 중 대부분은 반환 값 혹은 매개변수의 의미상의 차이로 인해 자료형과 자료의 의미, 자료 구조 등을 변경해야 하는 경우가 발생한다.

복합 매핑은 이미 생성된 객체를 사용하는 경우와 표 7과 같은 새로운 객체를 생성하는 경우로 구분할 수 있다. 복합 매핑으로 재구성된 라이브러리가 제한된 리소스를 사용하는 경우에는 반드시 해제 시점을 고려해야 한다.

주요 라이브러리에 대한 매핑을 통해 시스템 라이브러리, 파일 시스템 라이브러리, 핸드셋 콘트롤 라이브러리, 수학 라이브러리, 그래픽 라이브러리, 문자열 라이브러리, 네트워크 라이브러리 등의 매핑 관계를 구성하였다.

GNEX 커널에서 해결한 그래픽 관련 API와 더불어 하드웨어 의존적인 부분이 포함된 사운드를 처리하기 위해 다양한 자료를 분석한 결과 안드로이드에서는 GNEX의 사운드 데이터인 MMF 포맷을 직접 플레이하는 것이

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

표 6 1:1 매핑 - 매개변수와 반환 값의 변환이 필요

GNEX	int Sin100(int v) // sin(v) * 100
안드로이드 변환 시	public static int Sin100(int v) { return (int)(Math.sin(v) * 100); }

표 7 복합 매핑 : 새로운 객체를 생성하는 경우

GNEX	void PlaySound(sound m) // 음악을 연주
안드로이드 변환 시	public static void PlaySound(sound m) { StopSound(); GnexGlobalVar.gnexChannel[0] = m; GnexGlobalVar.gnexChannel[0].mp = MediaPlayer.create(GnexGlobalVar.mContext, GnexGlobalVar.gnexChannel[0].resID); GnexGlobalVar.gnexChannel[0].mp.start(); }

불가능하다. 따라서 사운드 데이터에 대한 음악 파일을 생성하여 재생하는 방법을 선택한다. 음악 파일을 재생하기 위해서는 사운드 데이터를 XMF, MXMF, WAV 혹은 MP3 등 안드로이드에서 제공하는 음원 형식으로 변환하는 방법을 채택한다. 사운드 데이터 처리를 위해 MediaPlayer 객체와 사운드 파일에 대한 리소스 번호로 구성된 별도의 GnexSound 클래스를 정의한다.

4. 실험 결과

소스 레벨 콘텐츠 번역기의 기능을 검증하기 위하여 먼저 기능별로 테스트하였고, 그 다음에 상용 콘텐츠를 변환하여 검증하였다. 기능별 테스트를 위해 GNEX의

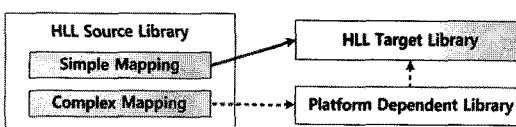


그림 17 라이브러리 매핑 관계

표 8 기능별 테스트 콘텐츠의 변환 결과

콘텐츠 종류	리소스		GNEX		Android	
	이미지	사운드	코드 길이	콘텐츠 용량	코드 길이	콘텐츠 용량
00311_MMI	0	12	1,567	96 KB	2,348	245 KB
00350_Graphic10	25	0	2,073	88 KB	2,655	246 KB
00276_Math	0	0	1,284	26 KB	1,867	195 KB
00314_File	0	0	1,921	38 KB	2,201	195 KB
00312_HW1	0	18	827	100 KB	1,701	258 KB
00320_Network	0	4	1,447	16 KB	2,022	197 KB

주요 라이브러리를 전반적으로 테스트할 수 있도록 테스트 콘텐츠를 선별하였으며, 구현된 GNEX-안드로이드 소스 레벨 콘텐츠 번역기를 통해 안드로이드 콘텐츠로 자동 변환하고, 안드로이드 플랫폼에서 실행 결과를 확인하여 변환된 콘텐츠의 정상적인 동작을 검증한다.

기능별 테스트를 위해 수많은 콘텐츠를 변환하여 확인하였지만, 본 논문에서는 상용 콘텐츠와 밀접하게 관련이 있는 6개 콘텐츠의 변환 내용을 분석하였으며, 표 8은 이와 같은 분석 결과를 요약한 것이다.

기능별로 선정된 콘텐츠의 기능과 의미는 다음과 같다. MMI(Man Machine Interface)는 단말기의 전화번호, LCD 크기, 시간, SMS 등의 라이브러리를 테스트하기 위한 콘텐트이며, 주로 클리핑 영역 계산, 시간 동기화 등을 구현할 때 사용된다. Graphic 콘텐트는 모바일 게임 콘텐츠 개발 시 가장 중요한 라이브러리로 다양한 게임 이미지를 화면에 표시하기 위해 사용된다. Math 콘텐트는 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. File 콘텐트를 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. HW(Hardware) 콘텐트는 콘텐츠 개발 시 필수적으로 사용되는 사운드 재생 기능을 포함한다. 마지막으로, Network 콘텐트는 네트워크 기능을 테스트할 수 있는 콘텐트로서 추가적인 아이템 구매, 게임 기록 등록 등을 구현할 때 주로 사용된다.

기능별 테스트를 통하여 주요 기능이 정상적으로 동

작함을 확인하였지만, 변환된 콘텐츠는 GNEX의 커널과 라이브러리를 모두 포함하기 때문에 콘텐츠의 크기가 증가되었다. 그러나 최근 출시되는 안드로이드 단말기는 높은 컴퓨팅 파워와 대용량의 메모리를 갖고 있기 때문에 충분히 감내해 낼 수 있는 수준이다.

기능별 콘텐츠의 테스트와 더불어, 상용화된 모바일 게임인 아이올로스(Aiolos), 레이싱 걸 포커(Racing girl poker), 신맞고(Shinmatgo) 등을 제안한 기법을 적용하여 변환한 결과 성공적으로 실행되었다. 표 9는 상용 콘텐츠의 변환 결과를 보여준다.

아이올로스, 레이싱 걸 포커, 신맞고 콘텐츠에 포함된 음원은 MMF 형식이다. 안드로이드에서는 MMF 형식을 재생할 수 없기 때문에 각각 MIDI, WAV, MP3로 음원을 변경하였다. 실험 자료를 분석한 결과 아이올로스는 콘텐츠의 용량이 감소한 것에 비해, 레이싱 걸 포커와 신맞고는 증가했다. 원인을 분석한 결과, 콘텐츠 용량의 증가는 음원의 용량에 비례하는 것을 확인할 수 있었다. 아이올로스는 저용량의 MIDI 음원을 사용하였으며, 레이싱 걸 포커는 비압축 음원인 WAV를 사용하여 콘텐츠 용량이 크게 증가함을 알 수 있었다. 따라서 상용 콘텐츠에서는 MIDI 음원을 사용하거나 신맞고 콘텐츠와 같이 WAV와 MP3를 조합하여 사용하는 것이 바람직하다. 그림 18은 레이싱 걸 포커를 안드로이드 SDK 1.0과 애뮬레이터를 사용하여 실행한 결과이다.

그림 19는 신맞고를 안드로이드 SDK 2.0과 애뮬레이터를 사용하여 실행한 결과이다.

실험 결과, 안드로이드 SDK의 모든 버전에서 정상적으로 동작하였으며, GNEX에서의 실행 모습과 비교한 결과 안드로이드에서도 게임의 기능적 측면뿐만 아니라, 그래픽, 사운드 등이 모두 동일하게 동작함을 확인할 수 있었다. 또한, 동일한 소스 코드를 안드로이드 플랫폼

표 9 상용 콘텐츠의 변환 결과

콘텐츠 종류	리소스		GNEX		Android	
	이미지	사운드	코드 길이	콘텐츠 용량	코드 길이	콘텐츠 용량
아이올로스	318	19	6,587	1,249 KB	10,175	799 KB
레이싱 걸 포커	743	71	15,220	1,431 KB	20,660	4,368 KB
신맞고	1,317	67	11,031	3,957 KB	21,321	4,702 KB

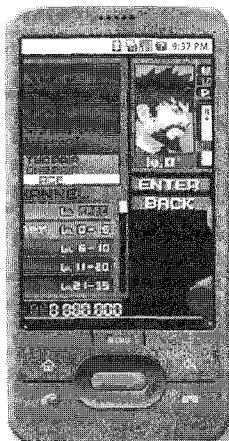


그림 18 레이싱 결 포커 실행 결과

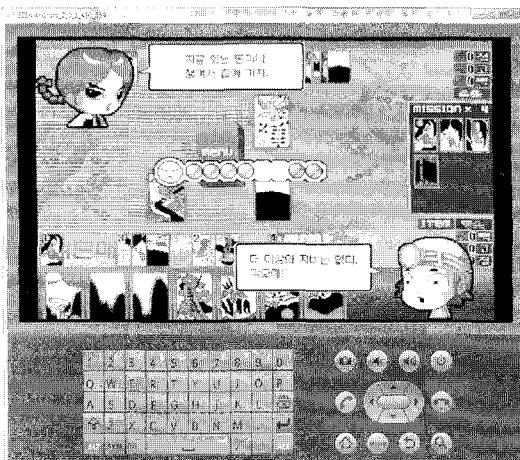


그림 19 신맞고 실행 결과

탑재 단말인 HTC Dream의 G1, G2와 모토로라의 모토로이(Motoroi)에 탑재하여 정상적으로 동작함을 검증하였다.

5. 결론 및 향후 연구

하나의 모바일 게임을 여러 플랫폼에서 서비스하는 것은 모바일 게임 시장에서 매우 중요한 사항이다. 특

히, 오픈 소스 플랫폼인 안드로이드는 리눅스 기반의 강력한 기능을 제공한다. 따라서 안드로이드를 휴대폰 플랫폼으로 채택한 제품의 비중이 크게 증가할 것으로 예측할 수 있다. 그러나 안드로이드에서 다양한 콘텐츠를 서비스하기 위해서는 기존의 콘텐츠를 재개발해야하는 비용이 발생한다. 따라서 이러한 노력과 비용을 감소시키기 위해서는 서로 다른 플랫폼에서 서비스되는 다양한 콘텐츠를 안드로이드 플랫폼으로 자동 변환하기 위한 도구가 필요하다.

다행히, 콘텐츠 변환 문제는 기존의 모바일 플랫폼(WIPI, BREW, GNEX 등)이 새로운 플랫폼인 안드로이드와 기능적으로 동일한 구성요소로 이루어져 있기 때문에 언어 번역 기법과 서로 대응되는 구성요소에 대한 매핑으로 해결이 가능하다.

본 연구에서는 컴파일러 이론을 적용한 체계적인 모바일 게임 번역 기법을 제안하였으며, 이를 적용하여 GNEX 용으로 구현된 콘텐츠를 동등한 의미를 가지는 안드로이드 용 자바 소스 프로그램으로 변환하는 소스 레벨 콘텐츠 변환기를 개발하였다. 구현 방법은 서로 다른 언어 간 번역 방법론, GNEX 커널 포팅, GNEX와 안드로이드 라이브러리 매핑의 3가지 영역으로 분리하여 진행되었다. 또한, 변환기는 기능별로 독립적인 모듈로 설계하였으며, 모듈간의 상호 호환성을 고려하여 개발하였다. 현재 개발된 변환기는 커널 포팅 및 라이브러리 매핑의 제한성으로 인하여 특별히 모바일 게임 콘텐츠에 적합하다고 판단된다. 표 10은 본 논문에서 제안한 변환기와 기존 변환기를 비교한 것이다.

본 논문에서 개발한 소스 레벨 콘텐츠 변환기는 언어 번역만을 수행하며, 커널의 일부 내용이 포함된 폴격 프로그램을 바탕으로 코드를 생성하기 때문에 번역에 소요되는 시간을 절약할 수 있으며, 커널과 라이브러리는 독자적인 안드로이드 소스 파일 형태로 구성되기 때문에 안정성을 높일 수 있다. 또한, 최근 각광 받고 있는 터치스크린 인터페이스의 자동 생성 기법을 도입하였다. 터치스크린 인터페이스 자동 생성 기법은 터치스크린 인터페이스를 고려하지 않은 기존의 모바일 게임에 터치스크린 인터페이스의 적용을 자동화할 수 있는 방법론이다[8].

표 10 언어 변환기의 비교

	마크업 언어 변환	GVM to BREW	GNEX to WIPI	GNEX to Android
언어종류	마크업 언어	C to C	C to C	C to Java
사용자 개입	없음	변환 후 통합	없음	없음
변환과정	1 pass	6 (6개의 분리된 도구)	1 pass	1 pass
언어간 변환	O	X	X	O
변환 대상	전체	전체	언어, 커널 일부	언어, 커널 일부
변환 미포함	X	X	커널, 라이브러리	커널, 라이브러리
터치 인터페이스	X	X	X	O

터치스크린 인터페이스 자동 생성 기법을 사용하면, 기존의 콘텐츠에 터치스크린 인터페이스를 추가 비용 없이 적용할 수 있다. 따라서 콘텐츠 재개발에 따른 추가적인 개발 비용을 절감하는 효과를 얻을 수 있다.

개발된 소스 레벨 변환기를 사용하여 기존의 상용 GNEX 콘텐츠인 아이올로스, 레이싱걸 포커, 신맞고 등의 모바일 게임을 자동적으로 변환하여 안드로이드 플랫폼에 탑재된 단말기인 HTC Dream의 G1, G2와 모토로라의 모토로이드 등에서 정상적으로 실행됨을 확인하였다[9].

본 논문에서 제안된 방법론과 변환기를 통해 다수의 콘텐츠를 가진 GNEX용 콘텐츠를 안드로이드 콘텐츠로 자동 번역함으로써, 짧은 시간에 다수의 안드로이드 콘텐츠를 확보할 수 있는 장점이 있다. 또한, GNEX-안드로이드 콘텐츠 변환기는 다음과 같은 기대효과를 창출 할 수 있다. 첫째로, 소스 변환기를 통해, 모바일 게임과 같은 기준에 개발된 다양한 콘텐츠를 새로운 실행환경으로 쉽게 이식할 수 있다. 둘째로, 이식 과정의 자동화를 통해 콘텐츠의 다양화를 이룰 수 있으며, 이식 과정에서 발생하는 비용의 절감으로 국내 콘텐츠 산업의 경쟁력을 향상시킬 수 있다. 특히, 안드로이드는 구글에서 추진하는 모바일 플랫폼으로 본 논문에서 제안한 모바일 게임 변환기를 통해 국내 콘텐츠를 쉽게 세계 시장에 제공할 수 있다. 마지막으로, 새로운 플랫폼이 등장 하더라도 제안된 번역 기법을 적용하면 단시간 내에 적은 비용으로 다수의 콘텐츠를 서비스할 수 있게 된다.

향후에는 변환된 콘텐츠의 성능을 평가하고 성능을 향상시키기 위한 연구가 필요하고, 매핑 정보를 위한 매핑 테이블 생성기를 자동화 기법을 도입하여 설계함으로써 모바일 게임 변환의 전 과정을 자동화할 수 있는 연구가 요구된다. 또한, 클래스 생성 시 함수 사이의 결합도를 분석하여, 클래스를 분류할 수 있는 기법에 대한 연구도 필요하다.

참 고 문 헌

- [1] Korea Game Industry Agency, Ministry of Culture, Sports and Tourism, 2008 *White Paper on Korean Games: Guide to Korean Game Industry and Culture*, Korea Game Industry Agency, 2008.
- [2] Jinyoung Kang, Chansung Jeong, *Mobile Programming using WIPI GNEX*, Lift and Power Press, 2006.
- [3] *Android - An Open Handset Alliance Project*, Google, 2008.
- [4] Junho Jo, Chulwoon Hong, and Yangsun Lee, "Development of Automatic Tool for Converting GVM Contents to BREW Contents in Mobile Environment," *Journal of KMMS*, vol.9, No.2, pp.38-49, 2005.
- [5] Erik D. Demaine, "C to Java: converting pointers into references," *Concurrency: Practice and Experience*, vol.10, pp.851-861, 1998.
- [6] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin, *Principles of Program Analysis*, Springer, 2005.
- [7] *Mobile C Library Function Reference*, SINJISOFT, 2008.
- [8] Seokhoon Ko, Yunsik Son, Jiwoo Park, and Seman Oh, "Automatic Generation Technique of Touch-Screen Interface for Mobile Game Contents," *Journal of KIISE : Computing Practices and Letters*, vol.15, no.9, pp.866-870, 2009.
- [9] Seman Oh, et al., *Development of GNEX to Android Source Converter*, Sinjisoft Corporation, Project Report, May, 2009.



박지우

2004년 2월 동국대학교 전자공학과 졸업(학사). 2006년 9월~2008년 8월 동국대학교 컴퓨터공학과 졸업(석사). 2009년 3월~현재 동국대학교 컴퓨터공학과(박사과정). 관심분야는 Secure Coding, 프로그래밍 언어, 컴파일러



오세만

1985년 3월~현재 동국대학교 컴퓨터공학과 교수. 1993년 3월~1999년 2월 동국대학교 컴퓨터 공학과 대학원 학과장 2001년 11월~2003년 11월 한국정보과학회 프로그래밍언어연구회 위원장. 2004년 6월~2005년 12월 한국정보처리학회 게임연구회 위원장. 관심분야는 Secure Coding, 프로그래밍 언어, 컴파일러, 모바일 컴퓨팅