Journal of Korea Multimedia Society Vol. 13, No. 4, April 2010(pp. 609-617)

모바일 콘텐츠의 자동변환을 위한 GNEX C-to-WIPI Java 변환기의 설계 및 구현

이양선[†], 함형범[††]

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 GNEX 플랫폼의 모바일 C 게임 콘텐츠를 WIPI 플랫폼의 자바 콘텐츠로 자동으로 변환해주는 콘텐츠 자동 변환기 시스템을 구현하였다. GNEX C-to-WIPI Java 콘텐츠 자동 변환기 시스템은 단시간 내에 다른 플랫폼으로 콘텐츠를 이식할 수 있도록 하여 동일 콘텐츠를 다른 이동통신사에 서비스하는데 소모되는 시간과 비용을 최소화해 준다. 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-WIPI Java Converter for Automatic Mobile Contents Translation

Yang-Sun Lee[†], Hyung-Bum Ham[††]

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 WIPI platform. The GNEX C-to-WIPI 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 Converter(모바일 콘텐츠 자동 변환기), Mobile Platform(모바일 플랫폼), GNEX(지넥스), WIPI(위피), 콘텐츠 분석기(Contents Analyzer), 소스 번역기(Source Translator), 리소스 변환기(Resource Converter), 폰트 생성기(Font Generator), 플랫폼 매핑엔진(Platform Mapping Engine)

※ 교신저자(Corresponding Author) : 이양선, 주소 : 서울 성북구 정릉동 16-1(136-704), 전화 : 02)940-7292, FAX: 02-940-7615, E-mail : yslee@skuniv.ac.kr
접수일 : 2009년 12월 28일, 수정일 : 2010년 1월 15일
완료일 : 2010년 1월 15일
[†] 종신회원, 서경대학교 컴퓨터공학과 교수
[††] 종신회원, 서경대학교 금융정보공학과 교수
(E-mail : hbham@skuniv.ac.kr)

1. 서 론

현재 국내 이동통신사별로 서로 다른 모바일 플랫폼을 채택하여 사용함으로 인해, 모바일 게임 콘텐츠 개발자는 하나의 콘텐츠를 서비스하기 위하여 각 플랫폼 특성을 고려하여 다양한 버전의 콘텐츠를 중복 개발해야 한다. 이는 개발자들로 하여금 이미 개발된 콘텐츠를 타 플랫폼으로 이식하는 작업에 대한 필연성을 유발하게 하였다. 하지만, 하나의 모바일 게임 콘텐츠의 소스 및 리소스를 분석하여 다른 플랫폼으로 이식하는 작업에 많은 시간과 비용이 소모된다. 새로운 콘텐츠를 개발하는 것보다는 하나의 콘텐츠를 서비스하기 위해 시간과 비용이 중복 투자되고 있다[1-5].

본 논문에서는 이런 문제를 해결하기위해 한 플랫폼의 콘텐츠를 다른 플랫폼에서도 실행할 수 있도록 자동으로 변환해주는 콘텐츠 자동 변환기 시스템을 연구하였다. 콘텐츠 자동 변환기 시스템은 모바일 게임 콘텐츠를 단시간 내에 다른 플랫폼으로 이식할 수 있도록 하여, 동일 콘텐츠를 다른 이동통신사에 서비스하는데 소모되는 인력, 시간, 비용을 절약해준다[6-10].

본 논문에서는 모바일 GNEX C 게임 콘텐츠를 WIPI Java 게임 콘텐츠로 자동 변환해주는 GNEX C-to-WIPI Java 변환기를 설계하고 구현하였다. 가장 많은 게임 콘텐츠를 보유하고 있는 GNEX 콘텐츠를 WIPI 콘텐츠로 자동 변환함으로써 기존 콘텐츠의 재사용성을 높이고 모바일 사용자가 보다 다양한 콘텐츠를 제공받을 수 있도록 하였다.

2. 관련 연구

2.1 GNEX 플랫폼

GNEX 가상기계(Virtual Machine)는 GNEX 응용프로그램을 해석하고 실행하는 역할을 한다. GNEX 커널은 다양한 시스템 인터페이스를 제공하며 메모리 관리자 탑재로 GNEX 시스템을 보고하고 응용프로그램의 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.

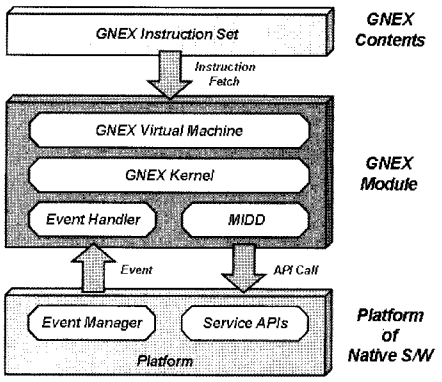


그림 1. GNEX의 구조

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[11-14].

2.2 WIPI 플랫폼

WIPI(Wireless Internet Platform for Interoperability)는 한국 무선 인터넷 표준화 포럼(KWISF)에 의해 제정되고, 한국정보통신기술협회(TTA)에 의해 TTAS.KO-60.0036으로 채택된 이동통신 단말기용 응용프로그램 실행 환경을 표준화한 규격이다. 이동통신 사업자들이 각기 다른 플랫폼을 사용함으로써 단말기 제조사와 콘텐츠 업체들의 개발 부담이 크다는 점 등을 들어 국내 무선인터넷 플랫폼 표준화의 필요성이 대두되면서 제정한 국내 표준 규격이다[15].

WIPI는 응용프로그램 개발 언어로 네이티브(Native) 방식의 C와 자바를 모두 지원한다. 자바의 경우 AOTC (Ahead Of Time Compiler)를 통해 C로

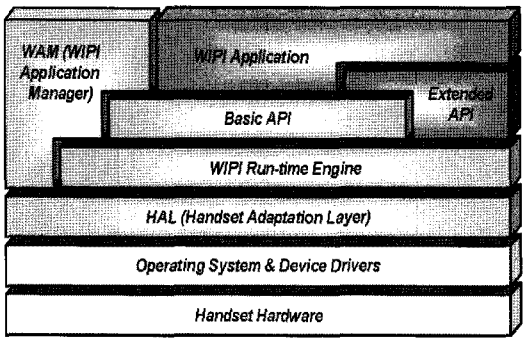


그림 2. WIPI 플랫폼의 구조

변환하여 다시 네이티브 방식으로 실행된다. 위피 규격은 크게 HAL(Handset Adaptation Layer)과 기본 API(Application Program ming Interface)로 구성된다. HAL은 플랫폼 이식성을 높이기 위한 표준화된 하드웨어 추상화 계층이다. 기본 API는 표준화된 플랫폼 호환성을 제공해 다양한 응용 프로그램 개발을 촉진하기 위한 기본 API 모음으로 C와 자바 API로 구성되어 있다[11,15,16].

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[1-4], GVM 플랫폼의 모바일 C 콘텐츠를 WIPI C나 Java로 변환하는 연구가 있었다[6,7]. 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[5], GVM C를 BREW C로 변환하는 연구가 있었지만[10] 변환할 때 소스코드가 자동으로 변환되지 않아 사용자가 개입하여 수동으로 변환을 해야 하는 등의 단점을 가지고 있다. 한편, 컴파일러 제작기술을[8,9] 활용하여 모바일 콘텐츠의 자동변환에 대한 연구가 시도되어 GVM 플랫폼의 모바일 C 콘텐츠를 WIPI C나 자바로 또는 MIDP 자바로 변환하는 연구가 발표되어 [6,7] 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 자동 변환하는 콘텐츠 변환기를 개발하였다.

3. GNEX C-to-WIPI Java 변환기 시스템

모바일 콘텐츠 자동 변환기 시스템은 소스 형태의 콘텐츠를 입력 받아 다른 플랫폼에서 실행되는 콘텐츠의 소스 형태로 변환하는 소스 레벨 변환을 한다. 소스 레벨의 자동 변환을 위해서는 우선, 소스 코드가 변환 대상 플랫폼에서 동일한 동작을 수행하는 소스 코드로 변환되어야 하고, 이미지, 사운드 등의 리소스 데이터도 변환 대상 플랫폼에서 사용할 수 있는 형식으로 변환되어야 한다. 또한, 동일한 프로그래밍 환경, 이벤트 환경을 위한 API 라이브러리가 제공되어야 한다[6-10]. 그림 3은 GNEX C-to-WIPI Java 변환기의 구성도이다.

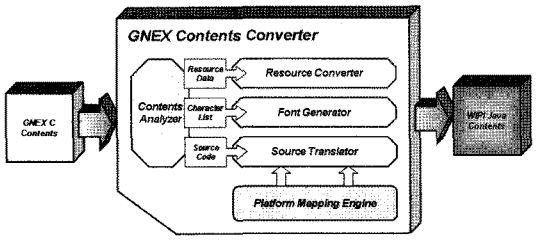


그림 3. GNEX C-to-WIPI Java 변환기 시스템 구성도

GNEX C-to-WIPI Java 변환기는 콘텐츠 분석기, 소스 번역기, 리소스 변환기, 폰트 생성기, 플랫폼 매핑 엔진으로 구성되어있다.

3.1 콘텐츠 분석기

콘텐츠 분석기는 소스 형태로 입력받은 GNEX 콘텐츠를 분석하여 소스 코드와 리소스 데이터를 분리하고 소스 코드 상에서 사용된 문자들을 추출한다. 리소스 데이터와 분리된 소스 코드는 소스 번역기로 전달되고, 리소스 데이터는 리소스 변환기로 전달되며, 추출된 문자 정보는 폰트 생성기로 전달된다.

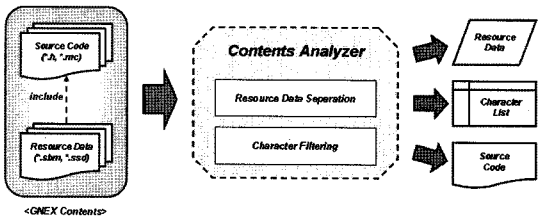


그림 4. 콘텐츠 분석기

GNEX에서는 리소스 데이터를 이진 파일 형태로 사용하지 않고 텍스트 파일 형태로 소스 코드에 입력하여 사용한다[11,13]. 따라서 리소스 데이터의 변환을 위해서는 소스 코드에 입력되어 있는 리소스 데이터를 분리해서 리소스 변환기에 전달하고, 리소스 부분을 제외한 나머지 소스 코드 부분은 소스 변환기에 전달한다. 다음 그림 5는 GNEX의 리소스 데이터의 예제이다.

```
const image bg2_shodow1 = {
        0x05, 0x22, 0x0D, 0x00, 0x0C,
        0x03, 0x04,
        0xFE, 0xFB, 0x9F, 0xFF, 0xFF, 0xBE, 0x47, 0xBF, 0xFF, 0x95, 0x11, 0xEF, 0xFF, 0xC0, 0x00, 0x38,
        0xFF, 0xC0, 0x00, 0x04, 0x7F, 0xC0, 0x00, 0x01, 0x1B, 0xF8, 0x00, 0x00, 0x02, 0xBF, 0x00, 0x00,
        0x00, 0x1B, 0xE0, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x13, 0x80,
        0x00, 0x00, 0x5F, 0xE0, 0x00, 0x00, 0xFF, 0xC0
};
const image bg_ab = {
        0x06, 0x0F, 0x06, 0x06, 0x01,
        0x04, 0x10, 0x1A, 0x00,
        0xAA, 0xA5, 0x55, 0x58, 0x00, 0x95, 0x55, 0x80, 0x29, 0x55, 0x58, 0x02, 0x55, 0x56, 0x80, 0xA5,
        0x55, 0xA0, 0x00, 0x2A, 0xA8, 0x00, 0x00, 0x00
};
const image bg_abdown = {
        0x06, 0x07, 0x07, 0xFF, 0x07,
        0x04, 0x10, 0x1A, 0x00,
        0x95, 0x56, 0x55, 0x59, 0x55, 0x65, 0x55, 0x25, 0x54, 0xA5, 0x50, 0x2A, 0x80, 0x00
};
```

그림 5. GNEX 콘텐츠의 이미지 리소스 데이터 예제

3.2 소스 번역기

소스 번역기는 콘텐츠 소스 언어의 문법과 구문을 분석하여 다른 언어로 번역할 수 있는 컴파일러 제작 기술을 응용하여 개발하였다. 소스 번역기는 GNEX의 모바일 C 소스 코드를 입력으로 받아 구문과 문법을 분석하여 WIPI 자바 소스 코드로 번역한다[15-20]. 소스 번역기의 구조는 그림 6과 같다.

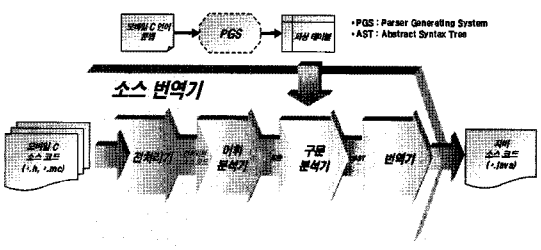


그림 6. 소스 번역기의 구조

소스 번역 과정은 우선, 전처리기(Preprocessor)에서 모바일 C의 #include, #define 등의 지시문을 처리하여 입력받은 소스 코드들 확장한다. 그리고 어휘 분석기(Lexical Analyzer)와 구문 분석기(Syntax Analyzer)를 사용하여 소스 코드에 대한 에러를 체크하고 문장의 구조를 트리 형태로 생성한다. 다음에 번역기(Translator)는 추상구문트리(AST)의 노드를 탐색하면서 WIPI 플랫폼에서 실행 가능한 자바 코드를 생성하여 출력한다.

다음 표 1, 2, 3은 GNEX C 언어의 자료형과 리소스 데이터, 그리고 포인터 타입을 WIPI Java로 번역한 예이다.

표 1. 변수와 자료형의 번역 예

	GNEX C	WIPI Java

기본 자료형	int i;
	int j=10;
	const int max=100;
		class Contents ... {
		static int i;
		static int j;
		static /*final*/ int max;
		
		Contents() { // 생성자
		j = 10;
		max = 100;
		}
		}

확장 자료형	image img;
	sound snd;
	string str;
	string str2="Hello";
		class Contents ... {
		static image img=
		new image();
		static sound snd=
		new sound();
		static string str=
		new string();
		static string str2=
		new string();
		
		Contents() { // 생성자
		str2.init("Hello");
		}
		}

표 2. 리소스 데이터 선언 번역 예

GNEX C	WIPI Java

image button={...};
sound logo={...};
	class Contents ... {
	image button = new image();
	sound logo = new sound();
	
	Contents() { // 생성자
	_initImage(button, "button.img");
	_initSound(logo, "logo.snd");
	}
	}

표 3. 포인터 선언 및 문장 번역 예

GNEX C	WIPI Java

int i = 10;
int *pt = &i;
 	
void func() {
  int *pt;
  ...
  *(pt++) = 10;
  tmp = *pt;
}
	class Contents ... {
	  ...
	  static int i[] = new int[1];
	  static int $i = $allocMem(i);
	  static int pt;
	  Contents() { // 생성자
	  i[0] = 10;
	  pt = $i;
	  }
	  void func() {
	  int pt;
	  ...
	  (($int)$v[pt++]).setValue(10);
	  tmp = (($int)$v[pt]).getValue();
	  }
	}

3.3 리소스 변환기

리소스 변환기는 GNEX 콘텐츠에 사용된 텍스트 형태의 리소스 데이터를 변환 목적 플랫폼인 WIPI 자바의 파일 시스템에서 사용할 수 있는 바이너리 파일 형태로 변환한다. 변환하는 리소스의 종류는 이미지 데이터, 사운드 데이터, 사용자 데이터가 있다. 다음 그림 7은 리소스 변환기의 구조이다.

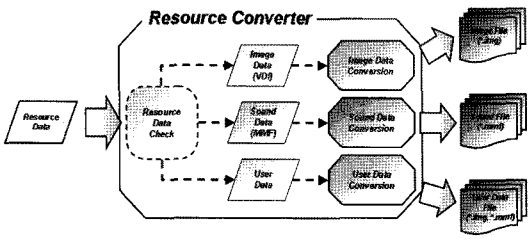


그림 7. 리소스 변환기

GNEX에서는 VDI(Variable Depth Image)라는 자체 이미지 형식을 사용한다. 모바일 C의 변수 형태인 VDI 형식을 사용하여 다른 플랫폼에서는 제공하지 않는 감마 변환, 팔레트 변환, 역상 출력 등의 그래픽 기능들을 제공하고 있다[11,14-16].

VDI 형식을 WIPI에서 사용할 수 있는 BMP, GIF 등의 형식으로 바꾸면 이러한 기능들을 지원하기 힘들게 된다. 따라서, GNEX의 모든 그래픽 기능을 지원하기 위하여 이미지 형식은 VDI를 유지하고, WIPI에서 사용할 수 있도록 라이브러리 함수와 GNEX의 그래픽 기능을 지원하는 함수를 제공하는 방법을 선택하였다. 다음 표 4는 GNEX VDI 형식의 이미지 타입 정보이다.

표 4. GNEX VDI 형식의 이미지 타입 정보

구분	Color Depth	Type	값	Bits Per Pixel	최대 사용 색상 수	Local Palette 크기(byte)
기본 VDI 형식	4 Gray	GRAY1	0×02	1	2	1
		GRAY2	0×03	2	4	2
		GRAY4	0×04	4	16	0
	Color	COLOR1	0×05	1	2	2
		COLOR2	0×06	2	4	4
		COLOR4	0×07	4	16	16
		COLOR8	0×08	8	182	0
	True Color	TCOLOR1	0×09	1	2	최대6
		TCOLOR2	0×0A	2	4	최대12
		TCOLOR4	0×0B	4	16	최대48
		TCOLOR8	0×0C	8	256	최대768

GNEX에서 사용하는 사운드 형식은 야마하의 MMF 형식이다. WIPI에서도 MMF 형식을 지원하기 때문에 사운드 형식에는 변환 없이 텍스트 형태에서 MMF 파일 형태로 변환하여 출력한다. 또한, 텍스트 형태의 사용자 데이터도 WIPI 파일 시스템을 통해 접근 가능하도록 바이너리 파일 형태로 출력한다.

3.4 폰트 생성기

본 논문에서 구현한 폰트 생성기는 GNEX 플랫폼의 폰트와 동일한 크기의 폰트를 생성하여 GNEX의 폰트 규격을 WIPI 플랫폼에서 사용할 수 있도록 하였다. 다음 그림 8은 폰트 생성기의 구조이다.

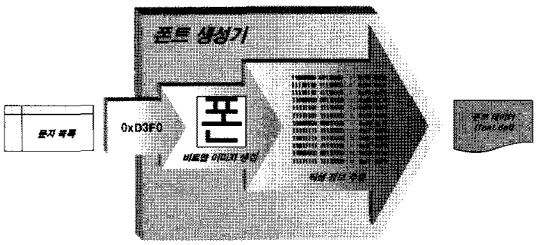


그림 8. 폰트 생성기

폰트 생성기는 폰트를 생성하고자 하는 문자의 비

트맵 이미지를 만들고 그 비트맵 이미지의 픽셀 정보를 추출하여 폰트를 화면에 출력하기 위해 필요한 데이터를 생성한다. 영어, 숫자와 같은 1바이트 문자의 경우 "SMALL", "MEDIUM", "LARGE" 규격에 해당하는 폰트 이미지 데이터를 생성하고 한글, 특수문자와 같은 2바이트 문자는 "LARGE", "HUGE" 규격에 해당하는 폰트 이미지 데이터를 생성한다. "DOUBLE"과 "HUGE DOUBLE" 규격은 플랫폼 매핑 엔진에서 제공하는 문자 출력에 관련된 함수에서 "LARGE"와 "HUGE" 규격의 데이터를 이용하여 확대 출력하도록 한다. 표 5는 GNEX와 WIPI 플랫폼의 폰트 규격을 비교한 것이다.

표 5. 폰트 규격 비교

플랫폼	폰트	폰트 크기	
		영어	한글
GNEX	SMALL	4 × 6	X
	MEDIUM	6 × 8	X
	LARGE	6 × 12	12 × 12
	DOUBLE	12 × 24	24 × 24
	HUGE	8 × 16	16 × 16
	HUGE DOUBLE	16 × 32	32 × 32
WIPI	SMALL	단말기 제조사 지정	
	MEDIUM	단말기 제조사 지정	
	LARGE	단말기 제조사 지정	

3.5 플랫폼 매핑 엔진

플랫폼 매핑 엔진은 GNEX 콘텐츠의 소스 코드에서 사용된 라이브러리 함수, 시스템 변수, 이벤트 핸들러 등을 변환 목적 플랫폼인 WIPI에서 사용할 수 있도록 자동으로 변환하여 GNEX 콘텐츠가 WIPI 플랫폼에서 응용 프로그램으로 실행할 수 있게 해 주는 기능을 한다. 이를 위해 GNEX와 동일한 실행 환경을 구축하고 이를 바탕으로 GNEX의 라이브러리 함수와 시스템 변수 등을 동일한 형태로 사용할 수 있도록 GNEX의 API를 WIPI 플랫폼의 API로 구현해서 제공한다. 이와 같이 함으로써 자바 언어로 번역된 소스 코드를 추가적으로 수정하지 않고 실행할 수 있도록 한다. 또한, GNEX와 동일한 형태의 API 사용으로 인해 번역된 소스 코드를 쉽게 이해하고 수정할 수 있다. 다음 그림 9는 플랫폼 매핑엔진의 구조이다.

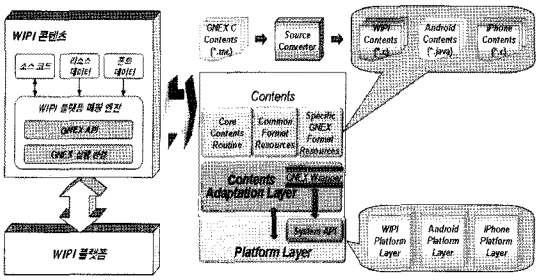


그림 9. 플랫폼 매핑 엔진

표 6은 GNEX-to-WIPI 변환기 시스템에서 지원하고 있는 GNEX의 API 목록이다. GNEX과 동일한 형태의 라이브러리 함수와 시스템 변수를 지원하기 위해 GNEX 플랫폼의 디스플레이, 그래픽, 사운드 출력, 이벤트, 진동, 데이터 저장 등의 시스템 환경과 유사한 환경을 구축하고, 이를 바탕으로 시스템 변수 및 함수들을 WIPI에서 동일하게 동작하도록 구현하였다. 또한, GNEX의 VDI 이미지 형식의 출력을 위한 환경, 함수가 제공된다. GNEX에서 사용하는 이미지, 사운드 타입의 멀티미디어 자료형과 스트링 자료형을 WIPI에서도 사용할 수 있도록 지원한다.

표 6. 지원 API 목록

구 분	WIPI Java 지원 함수
System(4)	GetHwConfig, GetDate, GetTime, Exit
Graphic(67)	pGetClip, GetGamma, GetActiveBuffer, GetColor, GetPixel, GetPaletteColor, GetPaletteColorRGB, GetImageAlpha, GetImageZoom, GetImageMirror, GetImageRotate, GetFontColor, GetFont, GetFontAlign, GetFontStyle, CopyLCD, ScrollLCD, DrawStr, DrawStrSolid, DrawText, CopyImage, CopyImageDir, CopyImagePal, Flush, ...
Handset Control(13)	PlaySound, StopSound, StartVib, StopVib, SetBackLight, GetUserNV, PutUserNV, SetTimer, SetTimer1, SetTimer2, ResetTimer, ResetTimer1, ResetTimer2

GNEX에서 사용되는 타이머는 자바의 스레드(Thread)를 사용하여 같은 기능을 수행하도록 구현된 WIPI용 타이머를 제공한다. 그밖에, GNEX의 시스템 라이브러리, 핸드셋 제어(Handset Control) 라이브러리, 스트링 라이브러리 등도 WIPI에서 동일한 동작을 하도록 구현하여 제공한다.

4. 실행결과 및 분석

본 논문에서 구현한 GNEX C-to-WIPI Java 변환기를 통해 GNEX 콘텐츠를 WIPI Java 콘텐츠로 변환한 후에 GNEX 콘텐츠 변환기의 성능을 측정하고 분석하였다. 콘텐츠를 실행하기 위해 각 플랫폼의 에뮬레이터를 사용했으며 그 내용은 표 7과 같다.

표 7. 사용된 에뮬레이터

플랫폼	에뮬레이터	구동방식
GNEX	GNEX 에뮬레이터	가상기계
WIPI	KTF 위피 에뮬레이터 1.2	네이티브 코드

그림 10은 "Aiolos", "Elemental Force", "무사도", "액션가면 Zoro"의 4개의 게임을 GNEX 에뮬레이터와 WIPI 에뮬레이터에서 실행시켜 비교한 모습이다. 그래픽, 이미지 출력, 사운드 출력, 키 이벤트, 타이머 구동, 데이터 저장 등의 동작이 모두 동일하게 실행됨을 확인할 수 있었다.

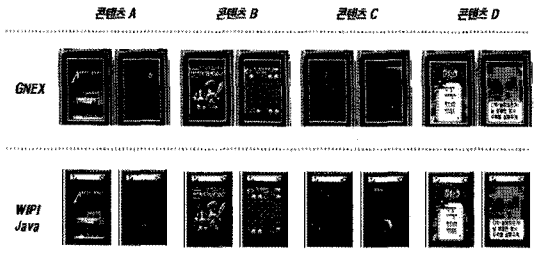
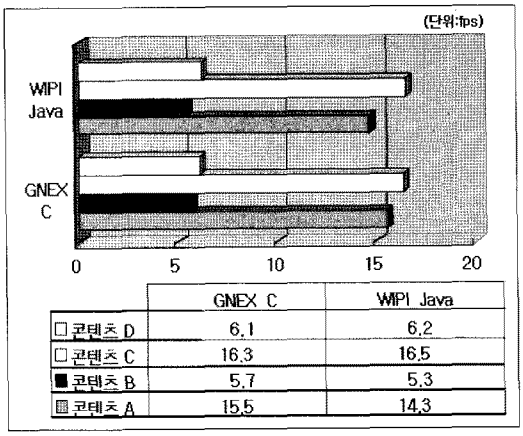


그림 10. GNEX, WIPI 플랫폼에서의 콘텐츠 실행결과

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-WIPI Java 변환기 시스템이 안정적임을 보여주고 있다.



	GNEX C	WIPI Java
콘텐츠 D	6.1	6.2
콘텐츠 C	16.3	16.5
콘텐츠 B	5.7	5.3
콘텐츠 A	15.5	14.3

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 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-WIPI Java 변환기는 이런 문제를 해결할 수 있는 하나의 방법이다. 본 변환기는 시스템 소프트웨어인 컴파일러 제작기술을 활용하여 GNEX 콘텐츠에서 WIPI 자바 콘텐츠로의 변환 작업이 단기간 내에 자동으로 진행될 수 있도록 하여 GNEX 콘텐츠를 WIPI 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 확산되고 있는 스마트폰의 iPhone이나 Android, Window Mobile과 같은 다양한 스마트폰 플랫폼에 적용될 수 있도록 할 예정이다.

참 고 문 헌

[1] 김미영, 무선 인터넷 서비스를 위한 XML 기반 컨텐츠 변환 시스템의 설계 및 구현, 영남대학교 석사학위논문, 2003.

[2] 김석훈, J2ME MIDP를 이용한 XML 기반의 모바일 컨텐츠 변환 시스템 설계 및 구현, 한남대학교 석사학위논문, 2003.

[3] 김영선, 장덕철, "XML Parser 추출에 의한 모바일 콘텐츠 변환 설계," 멀티미디어학회 논문지, Vol.6, No.2, pp. 267-274, Apr. 2003.

[4] 윤성일, Mobile 기반의 유무선 플랫폼 통합 변환 시스템, 한남대학교 박사학위논문, 2003.

[5] 이영종, WIPI와 BREW 플랫폼 간 C 언어 기반 솔루션 변환 방법, 충남대학교 석사학위논문, 2007.

[6] 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.

[7] 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.

[8] 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.

[9] 이양선, 나승원, 황대훈, "Intermediate Language Translator for Execution of Java Programs in .NET Platform," 한국멀티미디어학회 논문지, Vol.7, No.6, pp. 824-831, 2004.

[10] 홍철운, 조준호, 조현훈, 홍대기, 이양선, "모바일 게임 콘텐츠의 자동변환을 위한 GVM-to-BREW 번역기 시스템," 한국정보처리학회 게임 논문지, Vol.2, No.1, pp. 49-64, June 2005.

[11] 강진영, 정찬성, WIPI GNEX를 이용한 모바일 프로그래밍, 생능출판사, 2006.

[12] 신지소프트, GNEX SDK, http://www.gnexclub.com/download/download_a1.jsp

[13] 신지소프트, Mobile C Library Function Reference, http://www.gnexclub.com/

[14] 신지소프트, Mobile C Programming Guide, http://www.gnexclub.com/

[15] 한국 무선인터넷 표준화 포럼, 모바일 표준 플랫폼 규격 WIPI V1.2, http://www.kwisforum.org/file/public_pds/KWISFS.K-05-001R2.pdf, 2003.

[16] 황기태, 김남윤, 위피 자바, Jlet 모바일 프로그래밍, 생능출판사, 2009.

[17] Erik D. Demaine, "C to Java: Converting Pointers into References," *Concurrency: Practice and Experience*, Vol.10, pp. 851-861, 1998.

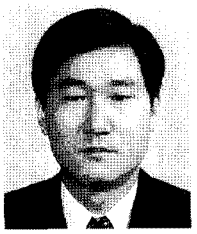
[18] Michael Kroll and Stefan Haustein, *Java 2 Micro Edition Application Development*, SAMS, 2002.

[19] Roger Riggs and Antero Taivalsaari, *Programming Wireless Devices with the Java Platform*, Addison-Wesley Professional, 2003.

[20] Sun Microsystems, CLDC Library API Specification 1.0, http://java.sun.com/javame/reference/apis/jsr030/

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 : 모바일 솔루션, 임베디드 SW, CT 기술, 게임 평가모델 등



함 형 범

1983년 동국대학교 통계학과 이학사
1985년 동국대학교 대학원 통계학과 이학석사
1991년 동국대학교 대학원 통계학과 이학박사
1992년~현재 서경대학교 금융정보공학과 교수
2004년~현재 한국정보처리학회 게임연구회 부위원장
2009년~현재 한국게임학회 이사
관심분야 : 게임평가모델, CT기술 가치평가, AHP, SEM