제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

모바일 콘텐츠의 재사용을 위한 GVM C-to-WIPI Java 변환기의 설계 및 구현

박상훈*, 권혁주**, 김영근**, 이양선*
*서경대학교 컴퓨터공학과
**(주)넥솔텔레콤 부설연구소
e-mail : {shpark*, yslee*} @skuniv.ac.kr
{sptunik**, carotple**} @nexoltelecom.com

Design and Implementation of a GVM C-to-WIPI Java Converter for a Mobile Contents Reuse

Sang-Hoon Park*, Hyeok-Ju Kwon**, Young-Koun Kim**, Yang-Sun Lee*
*Dept of Computer Engineering, SeoKyeong University
**Research Institute, Nexol Telecom

요 약

현재 국내에는 GVM(GNEX), WIPI, BREW, MIDP 등 다양한 모바일 플랫폼이 존재하고 있으며, 각 이동통신사별로 서로 다른 플랫폼을 채택하여 사용하고 있다. 이로 인해 개발자는 하나의 콘텐츠를 서비스하기 위하여 각 이동통신사들의 플랫폼에 맞추어서 다양한 버전의 콘텐츠를 개발하거나, 개발된 콘텐츠를 다른 플랫폼에 이식하기 위하여 콘텐츠를 변환하여야 한다. 이러한 이유로 콘텐츠 서비스를 위해 많은 시간과 비용이 소모되고 있으며, 모바일 사용자는 다양한 콘텐츠를 제공받지 못하고 있다.

본 연구팀은 이러한 문제점을 해결하기 위하여 콘텐츠를 자동으로 분석하여 변환해주는 콘텐츠 자동 변환기 시스템을 연구하였고, GVM 콘텐츠를 WIPI Java 콘텐츠로 자동으로 변환해주는 GVM C-to-WIPI Java 변환기를 설계하고 구현하였다. 본 변환기는 한번 개발한 콘텐츠를 단기간 내에 다른 플랫폼의 콘텐츠로 변환할 수 있도록 지원하여, 콘텐츠를 다른 플랫폼으로 이식하기 위한 개발 기간 및 비용을 단축시켜 준다. 또한, 모바일 사용자에게 다양한 콘텐츠를 제공할 수 있도록 한다.

1. 서론

현재 국내 모바일 산업에는 각기 다른 특성을 지니고 있는 GVM(GNEX), WIPI, BREW, MIDP 등 다양한 모바일 플랫폼이 공존하고 있으며, 이동통신사별로 각기 다른 플랫폼을 채택하여 사용하고 있다. 이런 상황에서 개발자는 하나의 모바일 콘텐츠를 서비스하기 위하여 각 이동통신사의 플랫폼에 맞추어서 다양한 버전의 콘텐츠를 개발하여야 한다. 때문에 개발 비용 및 기술자 수급에 있어서 큰 문제점을 가지고 있다[1,3]. 또한, 한 플랫폼의 모바일 콘텐츠를 다른 플랫폼으로 이식하여 재사용하기 위한 분석 및 변환 작업에 많은 시간과 비용이 소모되고 있다[2].

본 연구팀은 모바일 콘텐츠 개발 과정 또는 변환 과정에서의 시간과 비용의 소모를 최소화하기 위하여 소스 형태의 콘텐츠를 다른 콘텐츠의 소스 형태로 자동 변환해주는 콘텐츠 자동 변환기 시스템을 연구하였다. 콘텐츠 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 리소스 형식으로 변환되어야 한다. 또한, 동일한 프로그래밍 환경, 이벤트 환경을 위한 API 라이브러리가 제공되어야 한다. 콘텐츠 자동 변환기 시스템은 개발자가 단기간 내에 개발한 콘텐츠를 자동 변환할 수 있도록 지원한다.

본 논문에서는 GVM 콘텐츠를 WIPI Java 콘텐츠로 자동 변환해주는 GVM C-to-WIPI Java 변환기를 설계, 구현하였다.

2. 관련연구

2.1 GVM(General Virtual Machine)

GVM은 표준 C언어를 기반으로 순수 국내 기술로 개발한 어플리케이션 다운로드 솔루션으로, 무선 인터넷에 접속하여 Mobile C로 개발한 프로그램을 단말기에 다운로드 받아 실행시킬 수 있는 환경을 제공한다. GVM은 단말기에 탑재되는 GVM Player, 응용 프로그램 개발도구로 Mobile C 컴파일러를 포함하는 GVM SDK와 무선 네트워크를 통해 응용 프로그램을 다운로드 하는 GVM Server로 구성되어 있다[1,4]. GVM에서 사용하는 언어인 Mobile C는 표준 C 언어를 기반으로 모바일 플랫폼에 맞추어 응용 프로그램을 작성할 수 있는 프로그래밍 언어이며, 표준 C 언어와 동일한 문법 구조를 가지고 있다. 또한, 부동 소수점을 지원하지 않지만 이미지와 사운드 등 멀티미디어 자료형을 지원함으로써 멀티미디어 프로그래밍을 하기에 적합하게 설계되었다[1]. GVM에서는 제작된 이미지와 사운드를 GVM SDK의 이미지와 사운드 저작도구를 통해 GVM 규격으로 변환하여 Mobile C 소스 파일(*.mc)에 포함시킨다.

2.2 WIPI(Wireless Internet Platform for Interoperability)

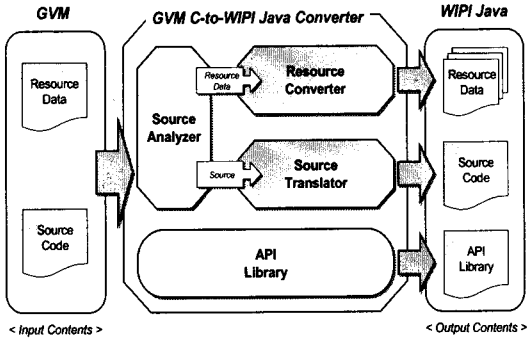
WIPI는 한국무선인터넷표준화포럼(KWISF)에 의해 제정되고, 한국정보통신기술협회(TTA)에 의해 TTAS.KO-60.0036으로 채택된 이동통신 단말기용 응용 프로그램 실행 환경을 표준화한 규격이다. 이동통신 사업자들이 각기 다른 플랫폼을 사용함으로써 단말기 제조사와 콘텐츠 업체들의 개발 부담이 크다는 점 등을 들어 국내 무선인터넷 플랫폼 표준화의 필요성이 대두되면서 제정한 국내 표준 규격이다. WIPI는 응용 프로그램 개발 언어로 Native 방식의 C와 Java를 모두 지원한다. Java의 경우 AOTC(Ahead Of Time Compiler)를 통해 C로 변환하여 다시 Native 방식으로 실행된다. 위피 규격은 크게 HAL(Handset Adaptation Layer)과 기본 API(Application Programming 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 C 와 Java API로 구성되어 있다[5]. 하지만, 각 이동통신사의 확장 API가 상이하게 발전함으로 인해, 이동통신 사업자간 호환성이 떨어진다. 표준 API를 사용한 콘텐츠의 경우 소스 레벨에서의 호환성은 유지되나 이동통신 사업자의 서비스 차원의 차별화를 위한 규격을 정의한 부분에 있어서 현재는 이동통신 사업자별 3가지 WIPI 플랫폼이 존재한다.

3. GVM C-to-WIPI Java 변환기 시스템

GVM C-to-WIPI Java 변환기는 GVM 콘텐츠를 WIPI Java 콘텐츠로 자동 변환해주는 시스템이다. 소스 형태의 GVM 콘텐츠를 입력으로 받아 WIPI Java 콘텐츠의 소스 형태로 변환한다.

3.1 시스템 구성도

GVM C-to-WIPI Java 변환기는 입력받은 GVM 콘텐츠에서 리소스와 소스 코드를 구분해 주는 소스 분석기(Source Analyzer), GVM의 Mobile C 소스 코드를 WIPI Java의 소스 코드로 변환시켜주는 역할을 하는 소스 변환기(Source Translator), GVM의 리소스 형식을 WIPI Java에서 사용 가능한 리소스 형식으로 변환해주는 리소스 변환기(Resource Converter), WIPI Java에서 GVM 플랫폼의 디스플레이 및 이벤트 환경을 동일한 방법으로 사용할 수 있도록 관련 예약어 및 함수들을 제공해주는 API 라이브러리(API Library)로 구성되어있다. (그림 1)은 GVM C-to-WIPI Java 변환기의 구성도이다.



(그림 1) GVM C-to-WIPI Java 변환기 구성도

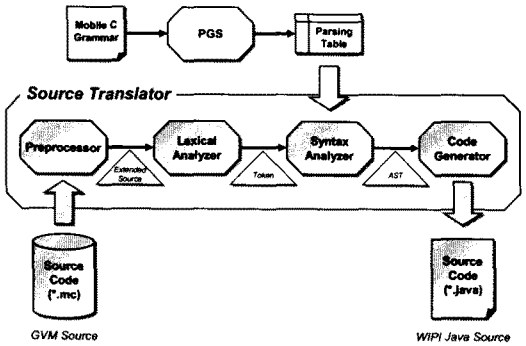
3.2 GVM C-to-WIPI Java 변환기의 구현

3.2.1 소스 분석기

소스 분석기는 입력받은 GVM 콘텐츠를 분석하여 Mobile C 소스 코드 형태로 변형된 리소스 데이터를 추출한다. image, sound 타입의 변수 형태인 리소스 데이터를 각 변수별로 분리하여 리소스 변환기에 전달한다. 그리고 리소스 부분을 제외한 나머지 소스 코드 부분은 소스 변환기에 전달한다.

3.2.2 소스 변환기

소스 변환기는 절차적 언어인 Mobile C와 객체 지향 언어인 Java 소스 코드 간의 언어적 차이를 극복하기 위하여 컴파일러 제작 기술을 바탕으로 구현하였다. GVM의 소스 코드를 입력으로 받아 구문과 문법을 분석하여 의미적으로 동등하며, 그 형태도 유사한 WIPI Java의 소스 코드로 변환한다. 소스 변환기의 구조는 (그림 2)와 같다.



(그림 2) 소스 변환기의 구조

먼저, Mobile C의 문법을 작성하고 PGS(Parser Generating System)를 이용하여 소스 변환기에서 구문과 문법을 분석하는데 필요한 파싱 테이블(Parsing Table)을 생성한다. 전처리기(Preprocessor)에서는 입력받은 GVM 소스 코드의 #define, #include 등의 지시문을 처리하여 확장된 소스 코드를 어휘 분석기에 전달한다. 어휘 분석기(Lexical Analyzer)는 확장된 소스 코드를 문법적으로 의미를 갖는 최소의 단위인 토큰으로 분할한다. 구문 분석기(Syntax Analyzer)는 어휘 분석기에서 토큰들을 전달받아, PGS에서 생성된 파싱 테이블을 이용하여 소스 코드에 대한 에러를 체크하고 올바른 문장에 대해서는 구문 구조(syntactic structure)를 추상구문트리(Abstract Syntax Tree)로 생성한다. 코드 생성기(Code Generator)는 추상구문트리를 입력으로 받아 운행(traverse)하면서 WIPI Java의 소스 코드를 생성하여 파일로 출력한다. (그림 3)은 소스 변환 알고리즘이다.

3.2.3 리소스 변환기

GVM에서는 VDI(Variable Depth Image)라는 자체 이미지 형식을 사용하여 다른 플랫폼에서는 제공하지 않는 그래픽 기능들을 제공하고 있다. VDI 형식을 WIPI Java에서 사용할 수 있는 BMP, GIF 등의 형식으로 바꾸면 이러한 기능들을 제공하기 힘들게 된다. 그래서 이미지 형식을 VDI 형식으로 유지하며, VDI 형식을 처리할 수 있는 API 라이브러리를 제공하는 방법을 선택하였다. GVM에서 지원하는 사운드 형식은 Yamaha의 MMF 형식이다. WIPI Java에서도 MMF 형식을 지원하기 때문에 사운드 형식에는 변환 없이 Mobile C 소스 형태의 SSD 파일에서 MMF 파일 형태로 변환하여 출력한다. GVM에서 파일 시스템을 제공하지 않는 특성에 의해 사용자 데이터를 image, sound 타입 변수에 Mobile C 소스 코드 형태로 입력하여 사용한다. 때문에 image, sound 타입 변수의 데이터를 분석하여 사용자 데이터를 분리하여 출력한다.

```
void codeGenerator(AST_NODE *root) {
  AST_NODE *node = root->son;
  while (node != NULL) {
    if (declaration part) {
      processDeclaration(node->son);
    }
    else if (statement part) {
      processStatement(node->son);
    }
    node = node->brother;
  }
}

void processDeclaration(AST_NODE *dcl_node) {
  traverse a declaration node, and then
  generate and output a declaration part code for the WIPI Java
}

void processStatement(AST_NODE *state_node) {
  traverse a statement node, and then
  generate and output a statement part code for the WIPI Java
}
```

(그림 3) 소스 변환 알고리즘

3.2.4 API 라이브러리

GVM C-to-WIPI Java 변환기에서 제공하는 API 라이브러리는 GVM의 API 라이브러리와 유사한 형태로 동일한 기능의 수행하도록 구현하였다. 이를 위하여 GVM 플랫폼의 디스플레이, 그래픽, 사운드 출력, 이벤트, 진동, 데이터 저장 등의 시스템 환경과 유사한 가상 환경을 구축한다. 이 가상 환경을 바탕으로 GVM 플랫폼의 시스템 변수 및 함수들을 WIPI Java에서 동일하게 동작하도록 구현하였다. 또한, GVM에서 사용하는 image, sound 타입의 멀티미디어 자료형을 WIPI Java에서도 사용할 수 있도록 지원하였다. GVM에서 사용되는 타이머는 Java의 쓰레드(Thread)를 사용하여 같은 기능을 수행하도록 구현된 WIPI Java용 타이머를 제공한다.

4. 실행 결과 및 분석

본 논문에서 구현한 GVM C-to-WIPI Java 변환기를 통해 GVM 콘텐츠인 "Elemental Force"를 변환하였다. (그림 4)은 "Elemental Force"의 소스 코드 중 일부분이다.

```
                                (중간생략)
void Game_Method()
{
    ClearBlack();
    for( i=0; i<=((swHeight-32)/6); i++ )
    {
        CopyImage( g_ScreenMidX-59, g_ScreenStartY+24+i*6,
BackGround_Mid_Left );
        CopyImage( g_ScreenMidX+55, g_ScreenStartY+24+i*6,
BackGround_Mid_Right );
    }
    CopyImage( g_ScreenMidX, g_ScreenStartY, BackGround_Top );
    CopyImage( g_ScreenMidX, g_ScreenEndY, BackGround_Down );
    tx = g_ScreenStartX + 7;
    ty = g_ScreenStartY + 25;
    if( swHeight < 120 )    tt = swHeight/9;
    else                    tt = 14;
                                (중간생략)
```

(그림 4) GVM의 Mobile C 소스 코드

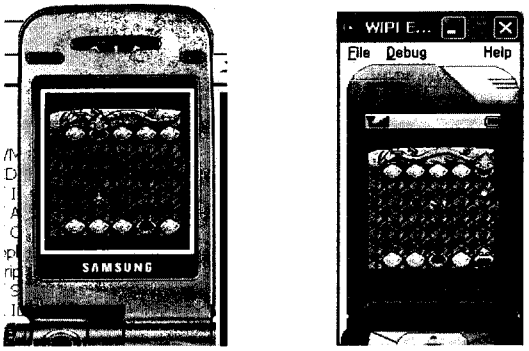
(그림 5)는 GVM의 Mobile C 소스 코드가 WIPI Java의 소스 코드로 변환된 모습이다. Mobile C와 Java의 언어적 차이점을 해결하고, 의미적으로 동등한 소스 코드가 출력됨을 알 수 있다. 또한, 소스 코드의 형태가 유사하여 분석과 수정이 용이하다.

```
import org.kwis.msp.lcdui.Display;
import org.kwis.msp.lcdui.EventQueue;
import org.kwis.msp.lcdui.Jlet;
import org.kwis.msp.lwc.AnnunciatorComponent;
public class ElementalWIPI extends Jlet {
                                (중간생략)
}
class Game extends GameBaseGVM implements game_res_id {
void Game_Method()
{
    ClearBlack();
    for (i = 0; ((i <= ((swHeight - 32) / 6)) ? 1 : 0) != 0; i++)
    {
        CopyImage((g_ScreenMidX - 59), (g_ScreenStartY + 24 + i * 6),
BackGround_Mid_Left);
        CopyImage((g_ScreenMidX + 55), (g_ScreenStartY + 24 + i * 6),
BackGround_Mid_Right);
    }
    CopyImage(g_ScreenMidX, g_ScreenStartY, BackGround_Top);
    CopyImage(g_ScreenMidX, g_ScreenEndY, BackGround_Down);
    tx = g_ScreenStartX + 7;
    ty = g_ScreenStartY + 25;

    if (((swHeight < 120) ? 1 : 0) != 0)
    {
        tt = swHeight / 9;
    }
    else
    {
        tt = 14;
    }
                                (중간생략)
```

(그림 5) 변환된 WIPI Java의 소스 코드

(그림 6)는 "Elemental Force"를 GVM 에뮬레이터와 WIPI 에뮬레이터에서 실행시켜 비교한 모습이다. WIPI 에뮬레이터에서 GVM 에뮬레이터에서와 동일하게 실행되는 것을 확인할 수 있었다.



(그림 6) 에뮬레이터 구동 화면

5. 결론 및 향후연구

국내 이동통신사들이 서로 다른 플랫폼을 채택하여 사용함으로 인해 콘텐츠 개발 또는 이식 과정에서 많은 시간과 비용이 소모되고 있다. 본 논문에서 구현한 GVM C-to-WIPI Java 변환기는 한번 개발한 콘텐츠를 자동으로 다른 플랫폼의 콘텐츠로 변환해주어서 콘텐츠 이식을 위한 개발 기간 및 비용을 단축시켜 준다. 나아가 단축된 기간을 신규 콘텐츠 개발에 투여함으로써, 모바일 콘텐츠의 생산성 및 경쟁력 향상을 가져오게 할 것이라 기대한다.

본 연구팀은 앞으로 GVM 콘텐츠를 BREW, MIDP 등 다른 플랫폼의 콘텐츠로도 변환되도록 할 예정이며, 더 나아가 다른 플랫폼의 콘텐츠도 변환 가능한 변환기를 구현 할 예정이다.

참고문헌

[1] 강진영, 정찬성, "WIPI GNEX를 이용한 모바일 프로그래밍", 생능출판사, 2006

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

[3] 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

[4] 신지소프트, GNEX SDK, http://www.gnexclub.com/download/download.jsp, 2006

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