Mobile C 기반의 GVM 응용프로그램을 Java 기반으로 변환하는 process.

신수원, 최윤석, 정기원
숭실대학교 컴퓨터학과
(cookzy@hanmail.net, yschoi@it.soongsil.ac.kr, chong@comp.ssu.ac.kr)

A transformation process of GVM application program based on from Mobile C to Java.

Soowon Shin, Yunseok Choi, Kiwon Chong
Dept. of Computing, Soogsil Univ.

요 약

본 논문에서는 Mobile C 언어 기반으로 만들어진 GVM 응용프로그램을 자바 기반의 모바일 응용프로그램으로 변환하는 절차를 제안한다. 자바 기반으로 변환된 모바일 프로그램은 높은 확장성 및 새로운 기능 추가의 용이함을 기대할 수 있다. GVM에서 자바 기반으로의 변환을 위해 필요한 구조를 분석하였으며, 이러한 구조로 응용프로그램을 변환하기 위한 6개의 단계를 제시하였다. 제안된 변환절차는 GVM 코드의 재사용에 중점을 두고 있으므로, 재사용을 통해 개발시간과 변환시간을 단축 시킬 것으로 기대된다.

1. 서 론

최근 몇 년간 모바일 분야의 발전이 급속도로 빨라지고 있다. 또한 이에 쏠리는 관심 역시 날로 늘어나고 있다. 현재 국내에는 GVM(General Virtual Machine: 국내 신지소프트에서 개발한 C 기반의 무선인터넷 플랫폼),SKVM(XCE가 독자적으로 개발한 자바 기반의 무선인터넷 플랫폼),Java Station(Veloxsoft에서 개발하고 019단말기에 탑재된 자바 기반의 무선인터넷 플랫폼), Brew(Qualcomm에서 개발한 C++ 기반의 무선인터넷 플랫폼)의 모바일 가상기계(VM : Virtual Machine)를 통해서 3개의 이동통신사에서 다양한 서비스를 제공하고 있다. 이 중에서 GVM이 현재까지 가장 많이 보급된 가상기계다. 또한 가장 많은 응용프로그램이 개발되었다. 하지만 외국의 경우 우리와는 다르게 GVM보다는 대부분의 단말기 제조사들이 자바를 기본 가상기계로 채택하고 있다. 따라서 국내의 양질의 응용프로그램들이 자바 기반의 가상기계에서 실행될 수 있도록 재 작성되어야 할 필요성이 있다. 본 연구에서는 국내 모바일 응용프로그램을 자바 기반인 SKT의 SKVM 또는 LGT의 Java Station으로의 변환을 위해서는 어떤 절차를 통해서 체계적으로 변환할 수 있을 가에 대한 변환 프로세스를 제안한다. 여기서 제안하는 변환 절차는 현재 서비스되고 있는 다양한 모바일 응용프로그램 중에서 가장 활성화 되어있는 엔터테인먼트 응용프로그램의 변환을 적용사례를 통해서 설명한다.

2. 관련 연구

2.1 GVM과 SKVM, Java Station의 특징

국내에서 자체 개발한 GVM과 SUN사의 자바를 바탕으로 국내에서 개발한 SKVM과 Java Station에 관한 특징을 [표 1]에서 나열하였다. 확장성이 GVM보다는 자바가 높다고 하는 이유는 기존의 PC기반의 다양한 기능의 API들이 모바일 자바로의 재 개발되어 쉽게 사용할 수 있기 때문이다. 자바가 그래픽 처리가 느리다는 내용은 단말기의 중앙처리장치의 성능에 따라서 다른 결과가 나올 수도 있다. 구조적 개발방법으로 개발되는 Mobile C와 객체지향 언어인 자바의 비교라 할 수 있다.

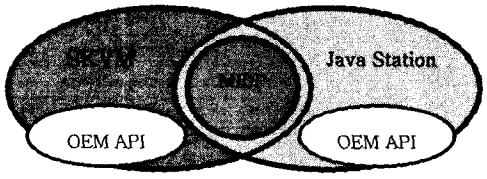
[표 1] GVM과 SKVM, Java Station 비교

GVM과 SKVM , Java Station 비교		
	GVM	SKVM, Java Station
사용언어	Mobile C(구조적언어)	자바(객체지향언어)
지원 API	제작사에서 제공하는 표준 API(약 120개)	MIDP와 OEM API.
객체사용	배열로 구현	객체 사용 가능
Thread	Timer 3개까지	Thread 지원
이미지	ImageMaster2 라는 별도 도구 사용 (Mobile C 소스로 변환)	Png(gif) GIF 데이터를 byte 배열로 변환해서 사용 가능
그래픽 처리 (*)	국내 자바 보다는 빠름	느림
확장성(*)	제한적이다	확장성이 아주 높다 XMLParser등 다양한 패키지 추가 가능.

(*)은 단말기와 가상기계의 버전에 따라서 다름.

2.2 MIDP, SKVM, Java Station의 관계.

국내에서 서비스되고 있는 자바 기반의 가상기계는 SUN에서 정의한 모바일 자바의 표준인 MIDP와 각 가상기계 제조사에서 자체적으로 제작한 API로 구성되어 있다. OEM API는 각 단말기를 제어하기 위한 단말기 종속 API를 말한다. 이 API의 사용으로 인해서 자바 가상기계간 변환이 어려워진다.



[그림 1] MIDP, SKVM, Java Station의 관계

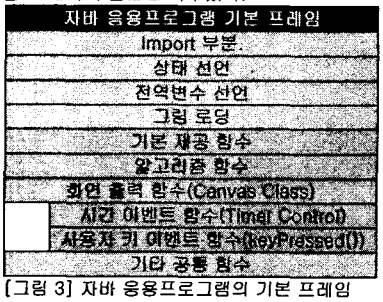
2.3 GVM 응용프로그램 개발 시 고려 사항.

GVM 기반의 응용프로그램 개발 초기부터 몇 가지 사항을 고려하면서 작성해야 한다. 선언 부분과 알고리즘과 그리고 화면 출력 부분을 확실히 분리하는 것을 권장한다. 화면 출력에 관련된 함수는 Draw라는 접두어를 붙이므로 인해서 후에 자바로 변환할 때 작업 시간을 단축 시킬 수 있다. 다시 그 알고리즘과 화면 출력 부분을 여러 개의 작업단위에 따라서 함수로 분리한다. 즉 하나의 작업 단위는 하나의 함수로 작성한다. [그림2]와 같이 총 8개의 부분으로 나누어진다.(개인 프로그램 방법에 따라서 다를 수도 있다.)

[그림 2] GVM 응용프로그램의 기본 프레임

2.4 자바 응용프로그램 개발 시 고려 사항.

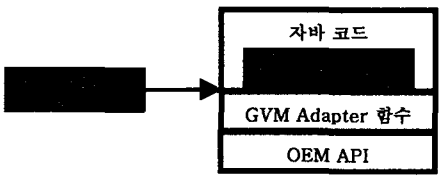
자바의 장점이자 개발자 편의를 위해서 지원되는 쓰레기 수집기가 모바일 단말기에서 그 성능을 최대한 발휘한다고 해도 단말기의 자원을 재활용 하기에는 아직 부족한 부분이 많이 있다. 따라서 과다한 객체 생성은 모바일 프로그램의 제한적인 자원에 치명적인 영향을 줄 수가 있다. 가능한 자주 사용하는 변수의 경우 전역변수 선언 부분에서 생성되도록 한다 한번 생성된 변수는 최대한 재사용 하도록 한다. [그림 3]은 자바 응용프로그램을 10개의 부분으로 나누었다.



[그림 3] 자바 응용프로그램의 기본 프레임

3.구조의 변환

GVM 코드의 자바 기반으로의 변환 후 그림과 같은 구조를 이룬다. 기존의 GVM 코드의 상당 수를 재사용한다. 자바 코드는 모바일 응용프로그램을 구현하기 위해서 추가되는 자바 코드를 말한다. GVM Adapter API는 GVM 코드와 자바 코드의 다리 역할을 해준다. OEM API는 각 단말기에 종속되는 단말기 제어를 위해서 추가된 MIDP의 표준 API외의 것을 말한다.



[그림 4] C에서 자바로 변환 시 구조

4. 자바 기반의 모바일 응용프로그램으로의 변환 절차 제안.

Mobile C와 자바는 모두 C를 기반으로 해서 개발되어서 코드의 비슷한 점이 많다. [표 2]과 [표 3]을 참고 하면서 다음과 같은 절차를 따라서 GVM 코드를 수정하면 체계적으로 변환할 수 있다.

[그림 5] 변환 절차 6단계.

- **외부 데이터 변환**

1. **그림 파일 변환 작업.**
   - A. GVM에서는 별도의 도구(ImageMaster2)에서 Mobile C소스 형태로 변환했지만 자바에서는 PNG또는 GIF형태로 변환 작업을 해주어야 한다.
   - B. SKVM에서는 큰 그림에서 일 부분을 출력해주는 API를 지원하지만 Java Station에서는 지원하지 않는다. 따라서 각각의 그림 마다 별도의 파일로 분리 시켜주어야 한다. 같은 자바라도 OEM API에서의 지원 여부에 따라서 별도의 작업을 더 해주어야 하는 경우가 있다. 더욱 자세한 내용은 각 가상기계의 OEM API 문서를 참고 하기 바란다.

- **선언 정보 수정**

2. **Header부분 변환**
   - A. Scripter Header를 자바의 Import 부분으로 변환한다. 경우에 따라서 다르지만 주로 Graphics와 Image를 포함하고 있는 javax. microedition. lcdui.* 와 Middlet의 필수 함수들을 포함하고 있는 javax. microedition. midlet.*의 패키지를 import한다.
   - B. 응용프로그램에 따라서 별도의 MIDP 또는 OEM API 패키지를 import 한다.
3. **변수 선언 부분 수정**
   - A. GVM 코드의 선언부분의 일부를 재사용 한다. 하지만 각 언어의 특성의 차이로 변수 선언부분을 [표 3]의 내용에 따라서 수정해 준다.
   - B. 출력되는 그림의 수에 따라서 Image 변수 선언해준다. GVM에서는 ImageMaster2라는 도구와 컴파일러에서 처리해 주었던 이미지 로딩을 자바에서는 Image변수 선언을 통해서 로딩 부분을 개발자가 코딩 해주어야 한다.

- **코드 변환**

4. **메인 클래스 생성으로 알고리즘 함수 분리.**
   - A. 선언한 Image 변수에 이미지 데이터를 로딩하는 코드 추가한다.
   - B. 추가적인 사용자 키 이벤트(확인/취소 단추)를 처리해야 할 경우 메인 클래스에서 CommandAction() 함수를 추가 시켜서 처리한다.
5. **Canvas class 생성으로 화면 출력 함수 분리**
   - A. 그래픽 관련된 화면 출력 함수들은 모두 Canvas를 상속 받은 클래스로 복사한다. Draw라는 접두어가 붙은 모든 함수들이 여기에 속한다.
   - B. SetTimer 대신 Thread를 생성하고 run()함수를 정의한다. 시간 이벤트 함수를 Canvas 함수에서 복사해서 재사용 한다. Run()함수는 일정한 시간마다 시간 이벤트 관련 함수를 호출하도록 한다.
   - C. 사용자 이벤트 처리를 위해서 keyPressed()를 추가 시키고 GVM 코드의 사용자 키 이벤트 부분을 복사해서 재사용 한다. GVM 코드와 자바의 각 단말기 단추에 할당된 번호가 다르다 따라서 수정해준다.

- **Adapter API 작성.**

6. **GVM Adapter API 작성**
   - A. 아래 [표 2]의 목록에서 나열한 함수들을 변환해 줄 수 있는 Adapter 함수를 작성한다. 이 함수들은 계속적으로 재사용이 가능하다. 이 API들만 완성된다면 다음 프로젝트에서의 변환 작업은 더욱 쉬어진다.([표 2]는 대표적인 API의 예이며 세부적인 GVM API 내용은 API문서를 참고하기 바란다.)

- **세부 수치 정보 수정**

7. **좌표 또는 기타 수치 수정.**
   - A. 각 단말기의 LCD의 크기가 다른 관계로 그림이 출력되는 좌표를 수정해 주어야 한다.

- **테스트 및 최적화**
  - A. 컴파일과 테스트를 통해서 최적화 시킨다.

[표 2] 함수 변환 테이블

GVM	MIDP
SetTimer(300, 1);	(TIMER_1 = new Thread(this)).start();
EVENT_KEYPRESS()	public void keyPressed(int keyCode)
EVENT_TIMEOUT()	public void EVENT_TIMEOUT()
RestoreLCD();	대부분은 offGraphics를 이용해서 offImage에 그림을 출력해 주고 g.drawImage(offImage,0,0, Graphics.TOP | Graphics.LEFT); 를 통해서 다시 LCD에 출력해준다.
Flush()	
SaveLCD()	
CopyImage(x,y,img)	
CopyImageDir(x,y,img,1)	
SetColor(col)	
FillRect(x,y,w,h)	
DrawStr(x,y,STR)	
SetStrColor(i,ii)	
StrCpy(STR," ")	
Rand(0,2)	private Random myRandom = new Random();

[표 3]세부 변수 선언 테이블

GVM	MIDP
#define ST_INTRO 0	final static int ST_INTRO=0;
int SpiderX2[2];	int SpiderX2[];
SpiderX2[0]=SCRW-(SCRW/2)/2-2;	SpiderX2 = new int[3]; SpiderX2[0]=SCRW-(SCRW/2)/2-5;
GVM에서의 이미지 로딩은 별도의 도구에서 지원함	Image menu_top; menu_top=Image.createImage("/image/menu_top.png")

5. 적용 사례

[표 6]과 [표 7]은 GVM으로 작성한 코드를 자바로 변환할 경우 재사용 정도를 계산해 보았다. 이번 프로젝트에 한해서는 약 70% 정도의 기존의 코드를 수정 없이 재사용할 수 있음을 알 수 있다. 단 이 수치는 그림 로딩과 GVM API adapter를 제외한 수치다. GVM API adapter역시 한번 작성으로 계속적으로 다른 프로젝트에 재사용이 가능하고 이미지 로딩 역시 간단한 변환작업으로 수정이 가능하다고 판단되어 포함시키지 않았다.(적용 사례에서 작성한 코드는 다양한 모바일 응용 프로그램 중에서 그래픽 처리가 많은 엔터테인먼트 부분의 응용프로그램에 한한다. 그리고 네트웍 또는 업무용 프로그램의 경우에는 다른 수치 또는 아래의 수치가 맞지 않을 수도 있다.)

[표 6] LOC 변화율(단위 line)

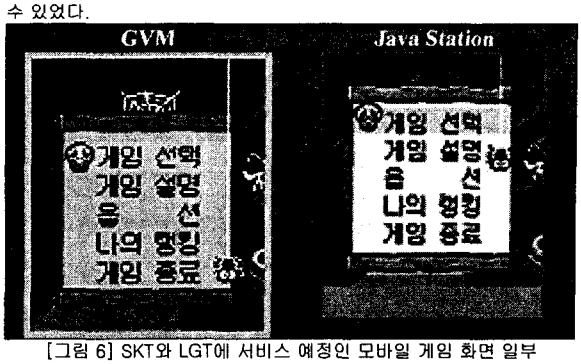
	GVM		자바	
LOC	2830	100.00%	3111	100.00%
선언 부분	145	5.12%	180	5.79%
그림 자료 로딩	0	0.00%	113	3.63%
알고리즘 부분	1289	45.55%	1170	37.61%
(Timer Control)	363	-	331	-
(Key Control)	338	-	339	-
Draw 부분	1260	44.52%	1364	43.34%
GVM API adapter	0	0.00%	84	2.70%
기타	136	4.81%	200	6.43%

[표 6]은 변환 작업을 수행하기 전에 GVM 코드의 줄 수를 나타내었다. 이는 어는 부분이 가장 많이 재사용 가능한지를 [표 7]을 통해서 확인해 보았다. 그리고 상당부분 재사용이 가능함을 알 수 있었다.

[표 7] 코드의 수정 정도(단위 줄).

	GVM	추가	추가+수정	%
선언 부분	145	35	100	68.95%
그림 자료 로딩	0	113	113	100.00%
알고리즘 부분	1289	0	80	6.21%
(Timer Control)	363	32	47	34.66%
(Key Control)	338	1	39	11.54%
Draw 부분	1260	104	158	12.54%
GVM API adapter	0	84	84	100.00%
평균	-	-	-	29.23%

[그림 6]은 GVM과 Java Station의 실행 화면이다. 본 논문에서 제안한 변환 절차에 따라서 GVM 코드를 수정한 결과 아래와 같은 결과를 얻을 수 있었다.



[그림 6] SKT와 LGT에 서비스 예정인 모바일 게임 화면 일부

6. 결 론

본 논문에서 제안한 Mobile C 기반의 GVM 응용프로그램을 자바 기반의 응용 프로그램으로 변환하는 절차는 기존에 개발되어 있는 Mobile C 기반의 응용 프로그램의 소스 코드 상당 부분을 재사용하고 이를 체계적으로 변환하므로 높은 생산성을 기대할 수 있다. 특히 그래픽 출력이 많은 멀티미디어 응용 프로그램의 경우 구조적 언어와 객체지향언어의 특성에서 오는 차이가 있을 뿐이므로, 알고리즘 및 화면출력 함수 등 상당 부분의 코드를 재사용할 수 있다. 또한 GVM API 의 어댑터 부분의 경우 다른 응용 프로그램의 변환 시에도 사용할 수 있다. 제시한 변환절차는 소스 코드의 재사용으로 인한 생산성 향상뿐만 아니라 개발 언어 자체가 갖는 장점의 수용, 체계적인 변환 절차에 따른 개발자의 오류 감소 등을 기대할 수 있다.

향후에는 제안한 절차를 적용하여 멀티미디어 응용 프로그램뿐만 아니라 기타 다른 분야의 프로그램의 변환 시에 대한 차이를 연구해야 하며, 변환 절차의 자동화에 대한 연구가 필요하다. 이 외에 네트워크 부분의 변환에 대한 연구가 필요하다.

[참고 문헌]

1. 엔슬래시닷컴 저, " GVM Programming- Mobile Series" ,상양출판사, 2001-11-15, p28~39, p120~p190.
2. Jonathan Knudsen 저 (공기식,송문배 공역), " Wireless Java: Developing with Java 2 Micro Edition" , 인포북, 2002-01-05, p23~p68, p163~p172
3. Yu Feng 저 ," 무선 자바 프로그래밍(Wireless Java Programming with J2ME)" ,정보문화사, 2001-08-14, p359~p424.
4. 오용석, 김명호, 유제정 공저," Mobile Java Programming" ,가남사, 2001-10-20, p626~p665.
5. http://developer.xce.co.kr
6. http://www.sinjisoft.co.kr
7. http://www.mobilejava.co.kr
8. http://www.javasoft.com