경량 윈도우 관리기를 이용한 퍼스널자바 AWT 구현

(Implementation of PersonalJava™ AWT using Light-weight Window Manager)

김 태 현 † 김 광 영 †† 김 형 수 †††
(Taehyoun Kim) (Kwangyoung Kim) (Hyungsoo Kim)
성 민 영 †††† 장 래 혁 ††††† 신 현 식 †††††
(Minyoung Sung) (Naehyuck Chang) (Heonshik Shin)

요 약 자바는 플랫폼 독립성, 높은 보안성, 멀티쓰레드 지원 등의 다양한 장점을 가지고 있어서 내장형 시스템을 위한 실행 환경으로 기대를 모으고 있다. 널리 쓰이고 있는 자바 수행 환경 중 하나인 Sun 사의 퍼스널자바 (PersonalJava™)는 다양한 GUI를 쉽게 생성할 수 있도록 Truffle이라는 AWT 구조를 제공하고 있어서 셋톱박스나 PDA 등의 다양한 내장형 시스템에 활발히 이식되고 있다. 기본적으로 Truffle은 기존의 마이크로소프트 Win32 API나 X 윈도우 API를 기반으로 하고 있다. 그러나, 이러한 윈도우 관리기들은 많은 양의 디스크나 메모리 공간을 요구하므로 시스템 자원이 한정된 내장형 시스템에는 적합하지 않다. 본 연구에서는 내장형 시스템의 요구조건을 만족시킬 수 있도록 내장형 리눅스 상에서 경량 윈도우 관리기인 마이크로윈도우즈(Microwindows)를 플랫폼 그래픽 시스템으로 채택하고 퍼스널자바 AWT API를 구현하였다. 마이크로윈도우즈(Microwindows)는 경량이면서도 기존의 윈도우 관리기들과 유사한 기능을 제공하며, 별도의 그래픽 시스템 지원을 필요로 하지 않아서 다양한 플랫폼 상에 쉽게 이식될 수 있다. 또한, 소스 코드가 공개되어 있어서 응용에 따라 수정 및 확장이 용이하다. 본 연구에서는 내장형 리눅스 상에서 동작하는 마이크로윈도우즈를 이용하여 퍼스널자바 AWT를 구현하였으며 다양한 응용프로그램을 이용하여 그 효용성을 입증하였다.

Abstract Java is a promising runtime environment for embedded systems because it has many advantages such as platform independence, high security and support for multi-threading. One of the most famous Java run-time environments, Sun's PersonalJava™ is based on Truffle architecture, which enables programmers to design various GUIs easily. For this reason, it has been ported to various embedded systems such as set-top boxes and personal digital assistants(PDA's). Basically, Truffle uses heavy-weight window managers such as Microsoft Win32 API and X-Window. However, those window managers are not adequate for embedded systems because they require a large amount of memory and disk space. To come up with the requirements of embedded systems, we adopt Microwindows as the platform graphic system for PersonalJava AWT onto Embedded Linux. Although Microwindows is a light-weight window manager, it provides as powerful API as traditional window managers. Because Microwindows does not require any support from other graphics systems, it can be easily ported to various platforms. In addition, it is an open source code software. Therefore, we can easily modify and extend it as needed. In this paper, we implement PersonalJava AWT using Microwindows on embedded Linux and prove the efficiency of our approach.

† 비 회 원 : 서울대학교 컴퓨터공학부
thkim@cslab.snu.ac.kr
†† 비 회 원 : LG전자 DigitalTV연구소 ASW그룹 연구원
cameron@lge.com
††† 비 회 원 : (주) 네오싸이언 개발이사
kim@neocyon.com
†††† 학생회원 : 서울대학교 컴퓨터공학부
minyoung@cslab.snu.ac.kr
††††† 종신회원 : 서울대학교 컴퓨터공학부 교수
naehyuck@snu.ac.kr
shinhs@snu.ac.kr
논문접수 : 2001년 2월 9일
심사완료 : 2001년 4월 6일

1. 서 론

자바는 내장형 시스템 상에서 동작하는 응용 프로그램 작성에 사용되어 온 C, C++ 등의 언어를 사용할 때 발생하는 복잡성과 개발과정의 오버헤드를 감소시키기 위해 개발되었으며[1], 코드 재사용성, 플랫폼 독립성, 고수준의 GUI(Graphic User Interface)와 분산 환경에 적합한 특성 때문에 인터넷 프로그래밍 언어로 널리 사용되게 되었다. 자바가상기계(Java Virtual Machine)는 이러한 자바 언어로 작성된 프로그램을 수행시키는 추상 컴퓨터이다. 따라서 자바 응용 프로그램은 자바가상기계가 구현되어 있는 어떠한 플랫폼에서도 수행될 수 있는 장점을 가지고 있다. 현재 대부분의 운영 체제에서 자바가상기계를 지원하고 있으며 앞에서 언급한 많은 장점에 힘입어 자바는 인터넷 응용 뿐 아니라 내장형 시스템(embedded system)을 위한 응용 개발에까지 그 영역을 확대해 나아가고 있다.

이러한 추세에 따라 내장형 시스템을 위한 경량 자바 수행 환경들이 발표되고 있으며, 그 중의 하나로 Sun 사에서는 네트워크 연결이 가능한 내장형 시스템을 위한 퍼스널자바(PersonalJava™)[2]를 발표하고, 셋톱박스(set-top box)나 PDA와 같은 내장형 시스템에 적용하고 있다. 일반적으로 내장형 시스템은 개인 컴퓨터와 달리 그 용도나 입출력 장치의 특성에 따라 GUI의 모습과 기능이 달라질 수 있으므로 개발에 어려움이 따른다. 자바는 그래픽 지원을 위해 AWT(Abstract Windowing Toolkit) 라이브러리를 제공하고 있으며, 퍼스널자바에서는 마이크로소프트 윈도우즈 Win32 API를 이용한 버전과 Motif를 이용한 버전의 두 가지 구현을 제공하고 있다. 특히 최근에는 플랫폼 의존적인 코드를 줄이고, 많은 부분을 자바로 구현한 Truffle(Graphical Toolkit) 구조[3]를 제공하여 AWT의 이식성을 높이고 있다. 그로 인해 효율적이고 다양한 GUI를 쉽게 만들 수 있도록 함으로써 퍼스널자바는 현재 다양한 운영체제 및 하드웨어에 활발히 이식되고 있다.

그러나, Truffle도 여전히 Win32 API나 X 윈도우 API를 가정하고 있으므로 그래픽 지원이 전혀 없거나 경량의 윈도우 시스템만을 갖추고 있는 내장형 시스템으로의 이식이 쉽지 않다. 이러한 문제점을 해결하기 위한 상용 솔루션으로는 WindRiver 사에서 실시간 운영체제인 VxWorks에 퍼스널자바(버전 : 3.0.2)를 이식한 Personal JWorks가 있다[4]. Personal JWorks는 Truffle 구조에 기반하고 있으며 플랫폼 그래픽 시스템으로 UGL(Universal Graphics Library)을 사용한다. Personal JWorks는 운영체제로 VxWorks를 사용하므로 실시간 특성을 요구하는 내장형 시스템에 적합하다는 장점을 가지지만 리눅스와 달리 소스가 공개되어 있지 않아 비용 면에서 부담이 되고 여타 플랫폼에 이식이 불가능하다는 단점이 있다.

이러한 한계를 극복하기 위해 본 연구에서는 플랫폼 그래픽 시스템으로서 내장형 시스템을 고려한 경량 윈도우 시스템인 마이크로윈도우즈(Microwindows™)[5]를 채택하였다. 마이크로윈도우즈는 경량이면서도 Win 32 API나 X 윈도우 API와 유사한 API를 제공한다. 또한 운영체제나 별도의 그래픽 시스템의 지원이 필요 없고, 리눅스와 같이 프레임 버퍼(frame buffer)를 지원하는 경우에는 성능 향상도 가능하다. 그리고 소스가 공개되어 있어 다양한 수준의 수정 및 개발이 가능하다는 장점도 가지고 있다. 현재 마이크로윈도우즈는 다양한 프로세서와 다양한 운영체제 상에서 동작하며, 본 논문에서는 PowerPC 프로세서와 프레임 버퍼를 지원하는 내장형 리눅스 상에서 동작하는 마이크로윈도우즈를 이용하여 퍼스널자바 AWT를 구현한 내용을 소개한다.

본 논문의 구성은 다음과 같다. 2장에서는 마이크로윈도우즈의 특징 및 구조에 대해 살펴본다. 3장에서는 퍼스널자바 AWT에 대해 설명한다. 4장에서는 퍼스널자바 AWT의 구현 환경과 마이크로윈도우즈를 이용한 퍼스널자바 AWT의 구현 과정 및 관련 내용을 설명하고, 구현 결과를 보여준다. 본 논문은 5장에서 결론 및 향후 과제를 제시하며 마친다.

2. 마이크로윈도우즈의 특징 및 구조

본 연구에서 구현한 퍼스널자바 AWT는 플랫폼 그래픽 시스템으로서 내장형 시스템을 고려한 경량 윈도우 시스템인 마이크로윈도우즈(버전: 0.89pre2)를 채택하였다.

2.1 마이크로윈도우즈의 특징

마이크로윈도우즈는 기존의 마이크로소프트 윈도우즈나 X 윈도우 시스템과 달리 경량(16비트 시스템에서 64KB 이하, 32비트 시스템에서 100KB 이하)이면서도 좋은 성능을 가지고 있다. Intel x86, MIPS R4000 Strong ARM, PowerPC 등 다양한 프로세서와 프레임 버퍼를 지원하는 32비트 리눅스, 16비트 리눅스 ELKS, MS-DOS (real and protected mode), RTEMS 등 다양한 운영 체제 상에서 동작하는 마이크로윈도우즈는 운영체제나 별도의 그래픽 시스템의 도움 없이 직접 이미지를 처리할 수 있다. 또한 리눅스 상에서 동작하는 마이크로윈도우즈는 프레임 버퍼를 이용하여 성능 향상

도 가능하다. 이와 같은 여러 가지 특징으로 인해 마이크로윈도우즈는 많은 내장형 시스템으로 그 사용 영역이 넓어지고 있다.

2.2 마이크로윈도우즈의 구조

마이크로윈도우즈는 그림 1과 같이 계층적으로 설계되어 있어 요구사양에 따라 계층별로 수정이 가능하다 [5].

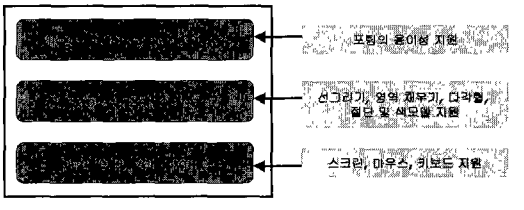


그림 1 마이크로윈도우즈의 계층구조

각 계층별로 기능과 계층간의 관계를 살펴보면 다음과 같다.

가. 디바이스 드라이버 계층

하위의 디바이스 드라이버 엔진 계층은 화면 구성 장치나 입력 장치에의 접근 루틴을 제공한다. 중간의 장치 독립적인 그래픽 엔진 계층의 루틴은 디바이스 드라이버 엔진 계층의 장치 드라이버를 통해 하드웨어 장치에 대한 연산을 수행토록 한다. 이러한 구성을 통해 마이크로윈도우즈에 여러 가지의 다른 하드웨어 장치가 전체 시스템의 동작 방식에 영향을 주지 않으면서 추가될 수 있도록 하고 있다.

크게 보면 스크린 드라이버와 마우스 드라이버, 키보드 드라이버 세 종류의 장치 드라이버가 제공된다. 스크린 드라이버에는 리눅스(커널 버전 : 2.2.x) 프레임버퍼 시스템을 위한 드라이버와 16비트 ELKS와 MSDOS를 위한 드라이버, 그리고 VGA 하드웨어를 직접 초기화하게 되어 있는 드라이버 등이 있다. 본 연구에서는 리눅스에서 프레임 버퍼를 지원하므로 성능을 고려하여 프레임 버퍼 드라이버를 사용한다. 현재 마이크로윈도우즈는 리눅스용 GPM 드라이버, 리눅스나 ELKS 용 시리얼 마우스 드라이버, MSDOS 용 드라이버 등 세 가지를 지원하며, 본 연구에서는 입력 디바이스로 시리얼 마우스를 사용하므로 리눅스용 시리얼 마우스 드라이버를 사용한다. 키보드 드라이버는 리눅스와 ELKS 용 드라이버와 PC BIOS 용 드라이버 두 가지가 포함되어 있는데 본 연구에서는 키보드를 사용하지 않으므로 키보드를 널(null)로 설정하였다.

나. 그래픽 엔진 계층

화면 구성 장치나 입력 장치와 인터페이스하기 위해 이들 장치에 대한 드라이버들을 호출하는 마이크로윈도우즈의 중요 그래픽 기능은 장치 독립적인 그래픽 엔진 계층에 구현되어 있다. 이 계층의 루틴들은 응용 프로그램으로부터 직접 호출될 수 없고 API 함수를 통해서만 호출될 수 있다.

다. API 계층

현재 마이크로윈도우즈는 마이크로윈도우즈 API와 Nano-X API 두 가지의 다른 API를 제공한다. 마이크로윈도우즈 API는 마이크로소프트 Win32 API와 Win CE GDI 표준에 따르는 것으로 메시지 전달(message-passing) 방식인 반면, Nano-X API는 X 윈도우 시스템의 Xlib API를 따르는 것으로 클라이언트/서버 형태로 구현되어 있다. 본 연구에서 사용한 퍼스널자바 소스 코드는 Solaris용이므로 구현의 용이성을 생각한다면 Nano-X API를 사용하여야 하지만, 클라이언트와 서버 간의 통신 오버헤드로 인한 성능 저하를 막기 위해 마이크로윈도우즈 API를 사용하였다. 마이크로윈도우즈 API에는 표 1과 같은 그래픽 관련 API들이 있으며 응용 프로그램은 이들 API를 호출함으로써 그래픽 엔진 계층의 루틴에 접근한다.

표 1 그래픽 관련 주요 API

기능	API 함수
색 지정 및 변환	SetTextColor(), SetBkColor(), GetSysColor(), SetSysColor()
그리기 및 색 채우기 지원	SetBkMode(), SetPixel(), MoveToEx(), LineTo(), Rectangle() SetROP2(), FillRect()
그래픽 관련 객체 생성, 삭제 등 관리	CreatePen(), CreateSolidBrush(), CreateCompatibleBitmap(), CreateCompatibleDC(), DeleteDC(), DeleteObject(), SelectObject()
비트맵 지원	BitBlit(), DrawDIB()
텍스트 지원	TextOut(), ExtTextOut(), DrawText()
기본 속성 검색	GetStockObject(), GetSystemPaletteEntries()

3. 퍼스널자바 AWT

본 장에서는 퍼스널자바 AWT 구조에 대해 소개한다. 1장에서 언급한 바와 같이 퍼스널자바 AWT는 Truffle 구조에 기반하여 이식성을 높이는 방향으로 설계되어 있다. Truffle은 모두 자바로 작성된 peer set 구현과 다양한 look & feel 디자인을 지원하는 프레임워크를 제공한다.

또한 Truffle은 네이티브(native) 윈도우 시스템이나 그래픽 시스템이 없는 플랫폼을 위한 윈도우 및 그래픽 시스템을 포함하고 있다. 그림 2는 Truffle 구조를 보여 준다.

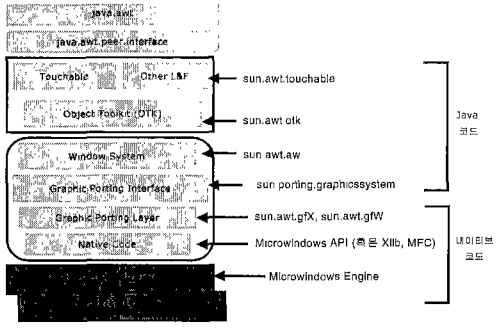


그림 2 Truffle 구조

3.1 Peer set

자바 응용 프로그램의 수행 환경은 AWT의 상위 클래스에 대한 구현인 peer set을 제공해야 한다. 예를 들어, Solaris 에서 동작하는 데스크탑 버전의 JDK 수행 환경은 상위 클래스인 java.awt.Font 클래스를 위한 폰트 peer 컴포넌트를 제공하는데, 이때 peer set의 구현을 위해 Motif 라이브러리를 이용한다. 퍼스널 자바의 경우 Truffle은 모두 자바로 작성된 peer set을 제공한다.

3.2 Touchable look & feel

AWT는 자바 응용 프로그램을 위한 GUI를 만들기 위한 툴킷이다. 내장형 시스템은 사용 목적이나 기능에 따라 다른 모습의 GUI를 필요로 한다. 초기 내장형 시스템의 GUI는 데스크탑의 GUI를 그대로 가져와서 불필요한 기능을 삭제하고 사용하는 형태였다. 그러나 다양한 종류와 기능의 내장형 시스템이 나오면서 이처럼 데스크탑의 GUI에 의존해서 GUI를 만드는 것은 한계에 다다랐다. 내장형 시스템은 데스크탑에 비해 보다 다양한 요구조건을 가지며 입출력 장치도 작고 간단하다. 또한 데스크탑에서 사용하는 '끌어다 놓기(drag and drop)'와 같은 인터페이스는 내장형 시스템에 부적합한 경우도 많다. 이처럼 데스크탑과 달리 내장형 시스템의 GUI는 다양한 용도에 따라 그 기능이 달라질 수밖에 없기 때문에 이를 개발하는 일이 어렵다.

Sun 사에서 나온 퍼스널자바의 Truffle은 다양한 look & feel 디자인을 지원함으로써 내장형 시스템 GUI 개발을 손쉽게 하고 있다. 기본적으로는 터치 스크린에 기반한 스크린 폰과 같은 내장형 시스템을 위한 touchable look & feel을 제공하는데 이를 이용하여 다른 종류의 look & feel 디자인을 설계할 수 있다.

3.3 OTK(Object Toolkit)

Truffle은 look & feel을 구현하는데 사용되는 OTK를 포함하고 있다. 그림 5에서 볼 수 있듯이, java.awt 클래스들은 peer set을 통해 구현되고, peer set 클래스들은 look & feel을 통해서, look & feel의 클래스들은 OTK를 통해서 구현된다. 따라서, 현재 퍼스널자바에 포함되어 있는 touchable look & feel 디자인 대신에 다른 look & feel 디자인을 사용하고자 할 경우에 peer set과 OTK를 모두 고려해야 한다. OTK는 플랫폼에 의존적인 look & feel 디자인을 작성하는데 있어 하부 시스템과 무관하므로 개발을 용이하게 한다.

3.4 윈도우 및 그래픽 시스템

Truffle은 대부분 자바로 작성된 윈도우 및 그래픽 시스템을 포함하고 있다. 플랫폼 그래픽 시스템에 독립적인 부분(Window System, Graphics Porting Interface, Graphics Porting Layer)은 자바로 작성되어 있고, 플랫폼 그래픽 시스템에 의존적인 부분(Graphics Portability Layer)은 C나 C++로 작성되어 있다. Graphics Porting Layer와 Graphics Portability Layer로는 gfX 버전과 gfW 버전이 있다. gfX는 X 윈도우 시스템 기반이며, gfW는 마이크로소프트 윈도우즈 시스템 기반이다. 본 연구에서는 내장형 리눅스 상에서 동작하는 마이크로윈도우즈를 플랫폼 그래픽 시스템으로 사용하는데, 마이크로윈도우즈 API와 마이크로소프트 윈도우즈 API의 유사성을 이용하기 위해 마이크로소프트 윈도우즈 시스템을 위한 gfW 버전의 Graphics Porting Layer와 Graphics Portability Layer를 사용한다.

4. 구현 내용

4.1 구현 환경

그림 3에서 보인 목표 시스템의 사양은 다음과 같다. 목표 시스템은 MPC860 프로세서를 탑재하고 있으며,

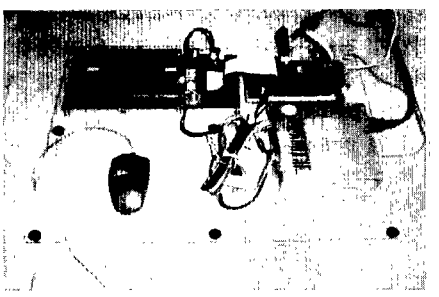


그림 3 목표 시스템

64MB SDRAM, 8MB 플래쉬 메모리, 256KB SGRAM, 4 KB 명령어/데이타 캐쉬, 직렬 통신 제어기, 이더넷 인터페이스를 기본 구성으로 한다. 디스플레이로는 640×480 해상도의 LCD를 장착했으며 입력 장치로는 시리얼 마우스를 사용한다. 운영체제로는 내장형 시스템의 특성에 맞게 이식된 내장형 리눅스[6]를 채택하였다.

4.2 구현시 접근 방법

내장형 리눅스에서 동작하는 마이크로윈도우즈를 이용하여 퍼스널자바 AWT를 구현하는 작업은 먼저 퍼스널자바(AWT 제외)를 내장형 리눅스에 이식하는 과정을 필요로 한다. 퍼스널자바를 데스크탑용 리눅스(프로세서 : i386)에 이식하는 작업을 먼저 진행하여 OS 의존적인 부분을 수정하고, 이를 다시 MPC860 용 내장형 리눅스에 이식하는 작업을 진행하여 프로세서 의존적인 부분을 수정하였다. 이는 데스크탑 환경에서의 개발이 용이함에 착안하여 대부분의 작업을 데스크탑 개발환경에서 진행함으로써, 개발 기간과 소요를 절감하기 위한 것이었다. 퍼스널자바 AWT 구현에 있어서도 동일한 접근방법을 사용되었다. 즉, 먼저 데스크탑용 리눅스 상에서 마이크로윈도우즈를 이용하여 퍼스널자바 AWT를 구현한 후, 이를 다시 MPC860 용 내장형 리눅스에 설치된 마이크로윈도우즈를 이용하도록 수정하였다.

4.3 구현 내용

본 연구에서는 퍼스널자바 AWT의 마이크로소프트 윈도우즈 버전 포팅 계층(gfW)을 바탕으로 마이크로소프트 윈도우즈 API를 마이크로윈도우즈 API로 바꾸어 가면서 퍼스널자바 AWT를 이식하였다. AWT 구현에 사용된 마이크로소프트 윈도우즈 API와 마이크로윈도우즈 API가 1:1로 매핑되지 않는 관계로 마이크로윈도우즈에서 제공하지 않는 API에 대해서는 수정 혹은 새로 구현하였다.

가. 리눅스에서 프레임버퍼 사용

리눅스 커널 2.2.0 이상에서는 사용자 응용 프로그램이 프레임 버퍼를 통해서 그래픽 메모리로의 접근이 가능하다. 내장형 리눅스에서 마이크로윈도우즈를 사용할 때 커널이 지원하는 프레임버퍼 기능을 사용하기 위해서 커널의 설정을 변경하고, frame buffer를 제어하는 장치 드라이버 부분을 새로 수정하였다.

나. 입력 디바이스 지원을 위한 리눅스 커널 수정

본 연구에서는 입력 디바이스로 시리얼 마우스를 사용하였다. 기본적으로 MPC860은 SMC와 SCC 라는 통신 컨트롤러를 내장하고 있으며 이들을 시리얼 디바이스로 사용할 수 있다. 목표 시스템에서 지원하는 시리얼 포트는 SMC와 SCC 당 각각 하나인데 SMC의 경우 콘솔의 용도로 사용하였으며, SCC 컨트롤러 중 SCC4를 마우스용으로 사용하였다. 리눅스의 UART 디바이스 드라이버는 SCC4를 현재 지원하지 않으므로 SCC4를 사용할 수 있도록 리눅스의 UART 디바이스 드라이버를 수정하였다.

다. 엔트리 포인트 차이점

윈도우즈 프로그램은 엔트리 포인트가 main()이 아니라 WinMain()이다. 마이크로윈도우즈는 리눅스 환경에서 동작하므로 엔트리 함수가 main이 된다. 따라서, 인터페이스의 호환성을 맞추기 위해 마이크로윈도우즈의 엔트리 포인트로 WinMain()을 만들어 주었다.

라. 마이크로윈도우즈 초기화

리눅스로 이식할 때는 윈도우즈 환경에서 동적 라이브러리의 엔트리 포인트에 해당하는 DllMain이라는 부분은 필요가 없어지므로, DLL 초기화 부분에 해당하는 내용을 자바 코드에서 동적 라이브러리를 처음 시작하는 곳에 넣어 주어야 한다. 동적 라이브러리에서 가장 먼저 불리는 부분은 GraphicsSystem.cpp의 InitJNI() 부분이며, 여기에 마이크로윈도우즈 그래픽 관련 초기화 루틴이 들어 있는데 마우스, 키보드, 스크린 같은 장치를 확인한다.

마. 그래픽 이식 계층(Graphic porting layer)의 수정

특정 윈도우 관리기를 이용해 퍼스널자바 AWT를 구현할 때 가장 주요한 작업은 Truffle 구조 상에서 상위의 플랫폼 독립적인 자바 코드와 하위의 플랫폼 종속적인 네이티브 코드(윈도우 관리기)를 연결해 주는 그래픽 이식 계층의 수정이다. 본 연구에서는 먼저 퍼스널자바 수행 이미지를 만들 때 JNI(Java Native Interface), NMI(Native Method Interface)를 사용하기 위해서 동적으로 만들어 주는 헤더 파일들이 유닉스 기반의 퍼스널자바를 기준으로 작성되어 있으므로 이 부분을 마이크로소프트 윈도우즈 버전의 퍼스널자바 소스에 맞게 변경해 주는 작업을 수행하였다. 퍼스널자바의 그래픽 부분은 마이크로소프트 윈도우즈 버전의 소스를 이용하고 나머지 부분은 Solaris 버전을 기반으로 했기 때문에 Xlib를 이용하는 자바 클래스(sun.awt.gfX) 대신에 마이크로소프트 윈도우즈 API에 기반한 자바 클래스(sun.awt.gfW)를 사용하도록 변경하고, 이와 관련해서 시스템 관련 속성 sun.graphicssystem도 sun.awt.gfW.GraphicsSystem으로 변경해 주었다. 퍼스널자바에서는 그래픽과 관련해서 SharedDC를 사용하는데 여기에서 마이크로소프트 윈도우즈 API 중 상호 배제 프리미티브(Mutex)를 사용한다. 이와 유사한 기능을 제공하기 위해 본 연구에서는 내장형 리눅스의 pThread 패키지에서 제공되는 p_thread_mutex_t 자료구조를 사용하여 WaitForSingle

Object(), CreateMutex(), Release Mutex() 등의 상호 배제 프리미티브들을 구현하였다. 또한, 퍼스널자바의 Truffle gfW에서 사용하는 키 매핑과 마이크로윈도우즈에서 사용하는 키 매핑의 차이점을 일치시켜 주었다. 마이크로소프트 윈도우즈에서 사용되는 ASCII 문자열과 유니코드 문자열 간의 변환, 커서 생성 및 관리, 키보드 스캔 등의 기능을 각각 마이크로윈도우즈에서 제공되는 API 함수와 내부 구현 함수를 이용하여 구현해 주었다.

5. 퍼스널자바 AWT 구현 결과 검증

5.1 PJCK(PersonalJava Compatibility Kit) 테스트

PJCK는 퍼스널자바 플랫폼의 구현이 퍼스널자바 플랫폼 명세서 및 Sun 사의 참조 구현에 부합하는지를 검증하기 위하여 사용되는 테스트 도구이다[7]. PJCK에 포함되어 있는 테스트들은 자바 클래스 라이브러리 테스트, 자바 프로그래밍 언어 명세서 테스트, 자바가상기계 테스트 등 3가지 종류로 분류된다. 본 연구에서 내장형 리눅스 상에서 동작하는 마이크로윈도우즈를 이용하여 구현한 퍼스널자바 AWT의 경우, 자바 클래스 라이브러리 테스트의 AWT 관련 테스트들을 수행해 봄으로써 그 호환성을 검증받을 수 있으며, PJCK의 AWT 관련 테스트 138개를 테스트해 본 결과 모두 성공하였다.

5.2 자바 그래픽 테스트 응용 프로그램 수행

PJCK 테스트 이외에 직접 작성한 테스트 프로그램들도 수행시켜 본 연구의 구현 결과를 알아보았다. 현재까지 수행한 테스트는 다음과 같으며 그림 4부터 7은 실제 실행 화면이다.

- ▶ 기본 도형(사각형, 원 등) 그리기 및 색 채우기
- ▶ 버튼 등 기본적인 컴포넌트 그리기 및 마우스를 통한 이벤트 처리
- ▶ 텍스트 출력

그림 4는 기본 도형, 즉 원과 사각형 등을 그리고 특정 영역을 색상으로 채우는 예제이다.

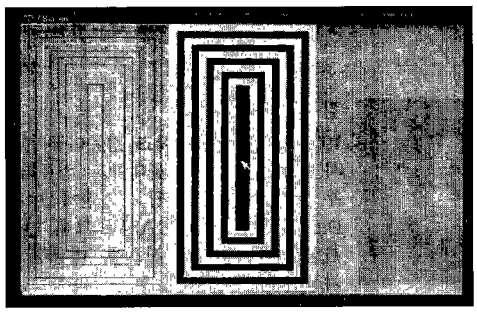


그림 4 기본 도형 그리기 및 색 채우기

그림 5와 6은 버튼과 체크박스 등 AWT의 기본 컴포넌트들을 그리고 해당 컴포넌트를 입력 디바이스 (본 논문에서는 마우스)로 선택했을 때 처리 결과를 보여주는 실행 예제이다. 그림 5의 경우 4가지 색상을 나타내는 버튼을 그리고 특정 버튼을 마우스로 선택했을 때 전체 프레임의 배경을 해당 버튼이 지시하는 색상으로 채우는 예제이다.

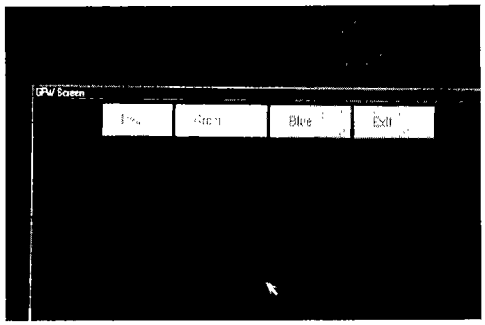
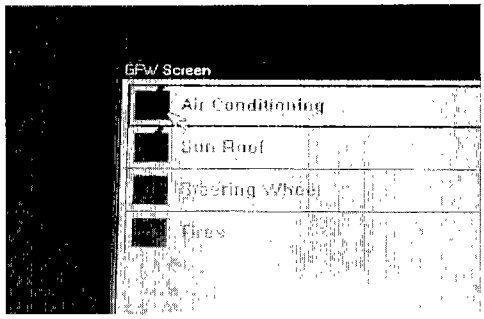
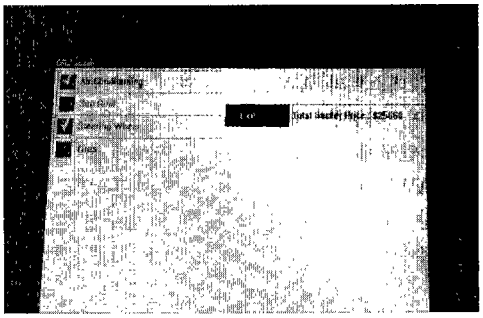


그림 5 버튼 그리기 및 입력 처리 예



(a) 체크박스 확대 화면



(b) 이벤트 처리를 포함한 전체 화면

그림 6 체크박스 리스트 그리기 및 이벤트 처리

그림 6은 체크박스 리스트 예제로 체크박스 리스트로 표현된 옵션 사항을 마우스로 선택했을 때 이벤트 처리가 정상적으로 처리되어 전체 비용을 계산하는 것을 보이는 예제이다.

그림 7은 텍스트 처리 예로 채팅 서버를 목표 시스템에서 수행시킨 상태에서 클라이언트들이 로그온/로그오프했을 때의 상태 정보와 채팅 내용을 화면에 보여 주는 것이다.

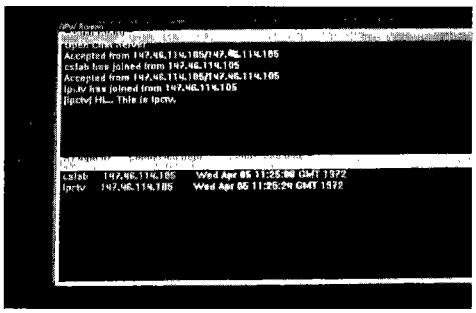


그림 7 텍스트 처리 (응용 프로그램: 채팅 서버)

6. 결론 및 향후 과제

본 논문에서는 내장형 리눅스 상에서 동작하는 마이크로윈도우즈를 이용하여 퍼스널자바 AWT를 구현하였다. 퍼스널자바 AWT는 다양한 GUI를 쉽게 디자인할 수 있도록 계층화된 Truffle 구조에 기반하고 있어 많은 내장형 시스템에 이식되고 있다. 그러나 Truffle 구조도 여전히 마이크로소프트 윈도우즈 API나 X 윈도우 API를 기반으로 하고 있어 내장형 시스템으로의 이식에 부담이 된다. 따라서 본 연구에서는 경량이면서도 성능이 뛰어나고, 기존의 윈도우 관리기들과 유사한 API를 제공하여 다양한 플랫폼 상에 이식이 용이한 마이크로윈도우즈를 플랫폼 그래픽 시스템으로 사용함으로써 이러한 문제를 해결하였다. 구현의 결과로 Sun 사에서 인증한 PJCK 테스트의 AWT API 관련 테스트를 모두 통과하였으며, 다양한 응용 프로그램을 이용한 테스트를 통해 본 연구에서 개발한 마이크로윈도우즈 기반 퍼스널자바 AWT의 효용성이 증명되었다. 본 연구에서 개발한 경량 윈도우 관리기 기반 AWT 솔루션은 위에 언급한 장점들로 인해 내장형 시스템을 위한 통합 자바 솔루션을 구축하는 데 널리 활용될 수 있다.

그러나, 현재 마이크로윈도우즈는 기존의 윈도우 관리기와 유사한 API를 제공하고 있지만 이들 API를 이용하여 AWT의 모든 기능을 구현하는 데는 어려움이 있다는 한계가 있다. 예를 들어 AWT의 기본 컴포넌트 중 하나인 버튼의 경우 round rectangle 형태로 구현되어야 하나 마이크로윈도우즈에서는 이를 지원하고 있지 않다. 또한 마이크로윈도우즈 자체는 이미지 파일 처리 기능을 가지고 있으나 이를 퍼스널자바 AWT에서 사용하는 형태로 맞춰 주기 위해서는 많은 작업이 추가로 필요하다.

본 연구의 향후 과제로는 이미지 파일의 처리과 터치스크린, 태블릿 등과 같은 다양한 입력 디바이스를 지원할 수 있도록 현재의 구현을 확장하는 것을 들 수 있다.

참 고 문 헌

[1] Sun Microsystems Inc. The Java Language Environment : A White Paper. 1995

[2] PersonalJava, http://java.sun.com/products/personaljava/index.html.

[3] Truffle Graphical Toolkit Customization Guide, http://java.sun.com/products/personaljava/truffle/index.html.

[4] Personal JWorks, http://www.windriver.com/products/html/persjwks_ds.html.

[5] Greg Haerr, Microwindows Architecture, http://microwindows.censoft.com/microwindows_architecture.html. 2000.

[6] 강경태, 김태웅, 박상수, 신현식, 장래혁, "Power PC에 기반한 내장형 시스템을 위한 리눅스의 이식", 한국정보과학회 27회 추계학술발표회, 2000년 10월.

[7] Frequently Asked Questions about PersonalJava ™ Technology, http://java.sun.com/products/personaljava/faq.html.

김 태 현

1994년 2월 서울대학교 컴퓨터공학과 학사. 1996년 2월 서울대학교 컴퓨터공학과 석사. 1996년 3월 ~ 현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 실시간 시스템, 내장형 시스템 소프트웨어, 내장형 자바, 실시간 메모리 관리기법 등.

김 광 영

1999년 2월 서울대학교 컴퓨터공학과 학사. 2001년 2월 서울대학교 컴퓨터공학부 석사. 2001년 3월 ~ 현재 LG전자 Digital TV 연구소 ASW 그룹 연구원. 관심분야는 내장형 시스템, 내장형 자바 가상기계.

김 형 수

2000년 2월 경북대학교 컴퓨터공학과 학사. 2000년 3월 ~ 현재 서울대학교 컴퓨터공학부 석사과정. 2000년 10월 ~ 현재 (주)네오싸이언 개발이사. 관심분야는 내장형 시스템, 무선 이동통신 등.

성 민 영

1995년 2월 서울대학교 컴퓨터공학과 학사. 1997년 2월 서울대학교 컴퓨터공학과 석사. 1997년 3월 ~ 현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 실시간 시스템, 내장형 시스템 소프트웨어, 통신 시스템 성능 분석, 실시간 무선 통신 기법 등.

장 래 혁

1989년 서울대학교 제어계측공학과 공학사. 1992년 서울대학교 제어계측공학과 공학석사. 1996년 서울대학교 제어계측공학과 공학박사. 1997년 9월 ~ 현재 서울대학교 컴퓨터공학부 조교수. 관심분야는 저전력 시스템, 디지털 시스템 설계, 실시간 내장형 시스템 및 실시간 자바

신 현 식

1973년 서울대학교 응용물리학과 공학사. 1980년 미국 텍사스 대학교 의공학과 공학석사. 1985년 미국 텍사스 대학교 전기·컴퓨터공학과 공학박사. 1886년 ~ 현재 서울대학교 컴퓨터공학부 교수. 관심분야는 실시간 계산, 분산 시스템, 입출력 처리