실시간 운영체제 UbiFOS™ 상에서 KVM GUI의 설계 및 구현

손필창°, 강희성, 정명조, 이철훈
충남대학교 컴퓨터 공학과
{pcson°, hskang, mjjung, chlee}@cnu.ac.kr

The Design and Implementation of GUI in KVM on Real-Time Operating System, UbiFOS™

Pil-Chang Son°, Hui-Sung Kang, Myoung-Jo Jung, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.

요 약

임베디드 장치나 모바일 같은 기기들은 자바 환경을 적용하기 위해 SUN사의 CLDC(Connected Limited Device Configuration)에서 정의하고 있는 KVM(K Virtual Machine)을 탑재하여 사용하게 된다. 자바의 GUI를 제공하기 위해서 CLDC는 MIDP(Mobile Information Device Profile)에서 명세하고 있는 GUI 표준 API를 사용하게 되는데, 이는 운영체제의 네이티브(Native) 함수와의 상호 연동에 의해 동작한다. 이에 본 논문에서는 실시간 운영체제인 UbiFOS™ 기반의 KVM GUI를 구현하는데 있어 그래픽 윈도우 시스템과 GUI API와의 상호 동작을 위한 네이티브 함수와 이벤트 처리에 대해 설계 및 구현한 내용을 기술한다.

1. 서 론

최근 IT 산업의 발전과 더불어, 임베디드 시스템과 정보가전 제품에 플랫폼 독립성(Platform Independency), 보안성(Security), 네트워크 이동성(Network Mobility) 등의 장점을 가진 자바를 사용하기 위해 K 가상 머신(K Virtual Machine: KVM)의 탑재가 증가하는 추세에 있다. KVM은 이러한 제품을 사용하는 사람들에게 편리한 그래픽 환경을 지원하기 위해 GUI(Graphic User Interface)를 제공하고 있다.

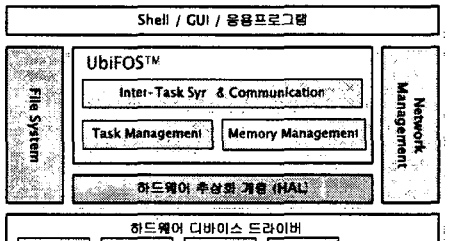
편리한 그래픽 환경을 제공하기 위해서 Sun사의 J2ME의 프로파일인 MIDP (Mobile Information Device Profile)는 자바의 GUI를 위한 표준 API를 javax.microedition.lcdui패키지로 명세하고 있는데, 이는 운영체제의 그래픽 윈도우 시스템과 연계되어 동작하게 된다. 본 논문에서는 실시간 운영체제인 UbiFOS™ 상에서 그래픽 윈도우 시스템과 GUI API와의 상호 동작을 위한 네이티브(Native) 함수(기본 도형 그리기, 색 지정, 텍스트 처리), 그리고 이벤트 처리에 대해 설계 및 구현한 내용을 다루고 있다[1][3].

2장에서는 이에 대한 관련연구에 대해서 기술하고, 3장에서는 KVM의 GUI와 그래픽 윈도우 시스템의 상호 동작을 위한 네이티브 함수와 이벤트 처리에 대한 설계 및 구현을, 4장에서는 테스트 환경 및 결과를, 5장 에서는 결론 및 향후 연구 과제를 기술한다.*

2. 관련 연구

2.1 실시간 운영체제 UbiOS™

본 논문에서 KVM의 GUI를 구현하기 위해 기반으로 사용한 UbiFOS™ 는 우선순위 기반 선점형 멀티 쓰레드 실시간 운영체제이다. 즉 실시간 운영체제 커널과 응용 프로그램이 통합되어 하나의 큰 프로그램으로 동작하는 구조로써 공통의 메모리 영역을 자유롭게 접근할 수 있고 사이즈는 약 24Kbyte 정도이다[2].

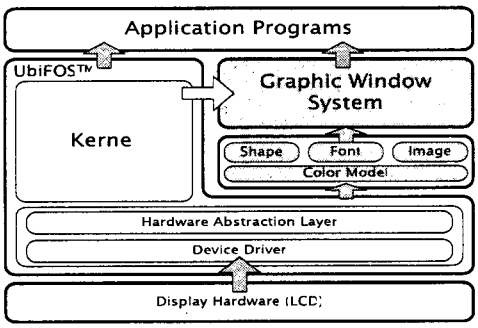


[그림 1] UbiFOS™의 기능 블록 다이어그램

[그림 1]은 UbiFOS™의 기능을 블록 다이어그램으로 나타낸 것으로, 운영체제에서 제공되는 기능은 태스크 관리, 태스크간 동기화 / 통신, 동적 메모리 관리 등이 있다.

* 본 연구는 정보통신부의 선도기반기술개발사업의 지원으로 수행되었습니다.

2.2 UbiFOS™의 그래픽 윈도우 시스템



[그림 2] UbiFOS™의 그래픽 윈도우 시스템

[그림 2]는 현재 UbiFOS™ 상에서 구현된 그래픽 윈도우 시스템으로 타겟 플랫폼이 바뀌었을 경우 쉽게 이식시키기 위해 Layered Architecture Design 방식으로 설계되어 있다[2].

2.3 MIDP (Mobile Information Device Profile)

MIDP는 모바일 인포메이션 디바이스(MID)를 위해 CLDC를 기반으로 설계된 자바 클래스 라이브러리에 대한 명세로서 애플리케이션의 모델, 유저 인터페이스와 이벤트 핸들링, 영속적인 저장공간, 네트워킹, 타이머 지원을 정의하고 있다. 그 중 GUI와 밀접한 관련이 있는 유저 인터페이스와 이벤트 처리 부분에 대해서만 살펴보도록 하겠다.

2.3.1 유저 인터페이스 (User Interface)

MIDP에서는 lcdui라는 새로운 유저 인터페이스를 설계하여 사용하는데 다음과 같은 디자인 원칙에 입각하여 유저 인터페이스를 디자인 하였다.

- 한 손으로 입력 가능
- 작은 스크린 사이즈에 적용가능
- 포인팅 디바이스가 없는 단말기에 적용가능
- 기존의 입출력 방법과 통합

이러한 MIDP의 유저 인터페이스 API는 javax.microedition.lcdui 패키지를 통해 제공되고 있다.

2.3.2 이벤트 처리

MIDP에서는 이벤트 처리에 대해 javax.microedition.lcdui 패키지의 Command 클래스와 Canvas 클래스에서 추상화 하고 있다.

[표 1] 이벤트 관련 클래스 및 함수

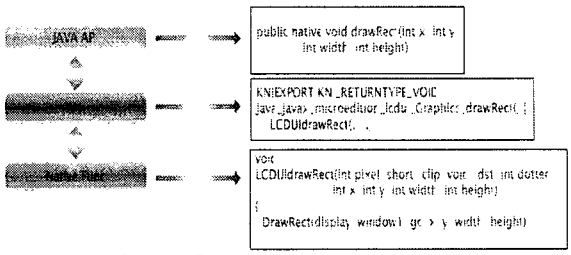
클래스 이름	함수 이름
Command	Void commandAction(Command c, Screen s);
Canvas	Public void keyPressed(int keyCode);

[표 1]은 자바 애플리케이션 작성시 이벤트 처리가 필요할 경우 사용하는 클래스와 함수 이다. 이벤트가 발생되었을 때 필요한 동작들을 [표 1]의 함수에 정의하여 애플리케이션을 작성하면 된다.

3. KVM의 GUI 및 이벤트 처리의 설계 및 구현

3.1 GUI의 네이티브(Native) 함수 구현

본 논문에서는 UbiFOS™의 그래픽 윈도우 시스템과 lcdui API의 상호 동작을 위한 네이티브 함수를 구현 하였다.



[그림 3] KVM의 GUI 동작 메커니즘

[그림 3]은 JAVA API 즉 lcdui API부분과 Native Func, 즉 UbiFOS™ 상에서 동작 가능하도록 구현한 네이티브 함수와의 동작 메커니즘을 나타낸 그림이다. 네이티브 함수는 UbiOS™ 의 그래픽 윈도우 시스템을 이용하여 구현하였고, lcdui API와 네이티브 함수의 연결은 KNI(K native Interface)를 이용하여 lcdui API와 상호 동작이 될 수 있도록 구현하였다.

[표 2] 네이티브(native) 함수

```
//사각형 그리는 함수
{
   <생략>
   // 사각형 그리는 부분
   if ((width < 1) || (height < 1)) {
      if ((width >= 0) && (height >= 0)) {
         DrawLine(pixel, clip, dst, dotted,
                  x, y, x + width, y + height);
      }
      return;
   } else {
     DrawRectangle(display, window1, gc, x, y, width, height);
   }
}
```

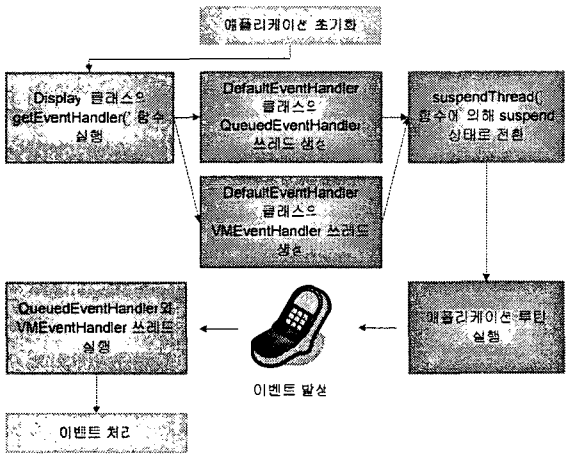
[표 2]는 사각형을 그려주는 네이티브 함수이다. 이러한 메커니즘을 이용하여 UbiFOS™ 상에서 동작하는 기본 도형 그리기, 색 지정, 텍스트 처리에 관련된 API들을 구현하였다

3.2 GUI의 이벤트 처리 구현

이벤트는 마우스, 키보드 상태의 비동기적인 변화를 KVM GUI에 전달하는 방법이다. UbiFOS™에서는 KEY 이벤트와 Touch Screen 이벤트를 지원하며, 이는 인터럽트 방식으로 처리한다.

본 논문에서 구현한 이벤트 처리는 [그림 4]와 같다.

MIDP에서는 애플리케이션 초기화 시 Display 클래스에서 DefaultEventHandler 클래스의 QueuedEventHandler 쓰레드와 VMEventHandler 쓰레드를 생성해서 실행시킨다. 그 쓰레드들은 이벤트가 발생하기 전까지 Suspend 상태로 있다가 KEY, Touch Screen과 같은 이벤트 발생하게 되면 다시 깨어나 동작하게 된다[4].



[그림 4] 이벤트 처리 동작 과정

네이티브 함수에서는 이벤트가 발생했을 때 이벤트 발생과 발생한 이벤트의 종류를 알리는 flag 변수를 셋팅 함으로써 이벤트 동작을 실행하게 된다. 발생한 이벤트의 정보를 StoreKVMEvent()함수 안에서 StoreMIDPEvent() 함수를 호출함으로써 저장한다.

[표 3] StoreKVMEvent() 함수

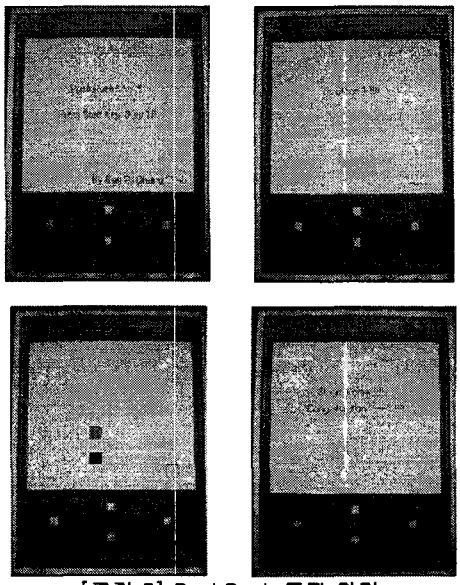
```
void StoreKVMEvent(int eventOccurr)
{
    Event event;
    KVMEventType evt;
    event.type = 0;

    while(eventOccurr > 0){
        if(eventOccurr && KeyPress){
            event.type = KeyPress;
        }else if(eventOccurr && TouchScreen){
            event.type = TouchScreen;
<생략>
    if(event.type == KeyPress) {
        evt.type = keyDownKVMEvent;
        evt.chr = translateKey();
        StoreMIDPEvent(&evt);
elseif(event.type == TouchScreen){
<생략>
        eventOccurr=0;
```

StoreMIDPEvent()함수에 의해 저장된 이벤트는 DefaultEventHandler 클래스의 쓰레드들에 의해 읽혀지게 되고 이벤트처리 루틴을 실행하게 된다.

4. 테스트 환경 및 결과

본 논문의 테스트 환경은 MBA2440 보드에서 실시간 운영체제로 UbiFOS™ 를 사용하였고, 개발도구는 ARM SDT v2.51 을 사용하였다. 그리고 디버그를 위해서 OPENice-A1000 Emulator를 사용하였다. 본 논문에서 구현한 그래픽 윈도우 시스템과 KVM의 GUI API와의 상호동작을 위한 네이티브 함수(기본 도형, 색 지정, 텍스트 처리, 이벤트 처리)를 테스트하기 위해 모바일 게임 프로그램인 PushPush를 수행하였다. 이벤트 처리는 KEY 이벤트와 Touch Screen 이벤트만 테스트 하였다. 그 결과 [그림 5]처럼 PushPush가 잘 동작 하는 것을 확인할 수 있었다.



[그림 5] PushPush 동작 화면

5. 결론 및 향후 연구과제

본 논문에서는 실시간 운영체제인 UbiFOS™에서 그래픽 윈도우 시스템과 KVM의 GUI API의 상호 동작을 위한 네이티브 함수를 구현하였고, 이를 통하여 기본 도형 그리기, 색 지정, 텍스트 처리에 관련된 API를 구현하였다. 그리고 이벤트 처리에 관련된 API도 구현하였다. 향후 연구과제로 gif나 png같은 이미지 파일을 처리하는 방법과 한글 폰트, 폰트의 속성을 변화 시키는 방법 그리고 펜, 마우스에 대한 이벤트 처리에 대해 연구할 예정이다.

참고 문헌

[1] Sun Microsystems, " *Mobile Information Device Profile(JSR-37*" , 2000

[2] 윤기현, 김용희, 박희상, 이철훈, " Design of Graphic User Interface for the Real Time Operating System" ,한국정보과학회, Vol.29, No2(I), pp400-402, 2002

[3] 백대현, 성영락, 이철훈, " *Design and Implementation of Event Handling in AWT for Java Virtual Machine GUI*" , 한국정보과학회, Vol. 30, No. 1(B), pp.94-96, 2003

[4] Scott Oaks & Henry Wrong, " *JAVA Threads, 2nd Edition Java™ 2*" , 1999