자바가상머신 GUI 를 위한 AWT 에서 이벤트 처리 방법 설계 및 구현

백대현[0], 성영락[†], 이철훈
충남대학교 컴퓨터공학과, [†] 국민대학교 전자정보통신공학부
(dhbaek[0], chlee)@ce.cnu.ac.kr, [†] yeong@mail.kookmin.ac.kr

Design and Implementation of Event Handling in AWT for Java Virtual Machine GUI

Dae-Hyun Baek[0], Yeong Rak Seong[†], and Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.
[†] School of Electrical Engineering, Kookmin Univ.

요 약

자바가상머신(Java Virtual Machine: JVM)을 이용하는데 있어서 GUI(Graphic User Interface)는 JVM 을 탑재한 제품을 사용하는 사람들에게 제품에 대한 편리한 그래픽 환경을 제공하는데 목적이 있다. AWT(Abstract Window Toolkit)는 Java™ 프로그램에서 GUI 를 제공하기 위한 표준 API 인 JFC(Java Foundation Class)의 일부분이다. 이에 본 논문에서는 리눅스 기반 자바 AWT API를 구현하는데 있어 가장 핵심 부분인 이벤트 처리가 X 윈도우 시스템과 자바 AWT API 사이에서 어떻게 상호작용하며 이루어지는지에 대해 기술하고 있다.

1. 서론

현재 데스크탑은 물론 임베디드 시스템과 정보가전 제품에 자바가상머신(Java Virtual Machine: JVM)의 탑재가 증가하는 추세에 있다. JVM 은 이러한 제품을 사용하는 사람들에게 제품에 대한 편리한 그래픽 환경 지원하기 위해 GUI (Graphic User Interface)를 제공하고 있다. AWT(Abstract Window Toolkit)는 Java™ 프로그램에서 GUI 를 제공하기 위한 표준 API 인 JFC(Java Foundation Class)의 일부분이다 [1][2].

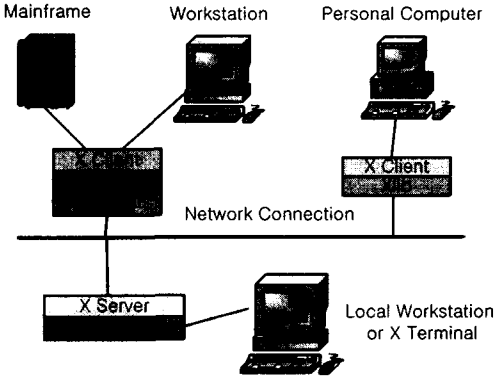
자바 AWT 는 네이티브(native) 윈도우를 바탕으로 하는데 추상화(Abstract)라는 말에서도 알 수 있듯이 AWT 를 이용해 윈도우를 작성하게 되면, MS 윈도우에서는 그것의 윈도우를 사용하고, 리눅스에서는 X 윈도우를 사용하는 등 운영체제에 따른 윈도우로 작성된다. 그러므로 자바 AWT 로 작성된 프로그램은 실행되는 환경에 따라 모양이 조금씩 다르게 나타나고, 이벤트 처리 방법도 조금씩 다르다.

본 논문에서는 리눅스를 기초로한 AWT API 를 구현하는데 있어 가장 핵심이 될 수 있는 이벤트에 대한 사항을 다루고 있다. 이벤트를 구현하는데 있어 고려해야 할 사항과, 자바 측면에서 이벤트를 처리하는 방법보다는 X 윈도우 시스템에서 발생한 이벤트를 처리하는 방법과, 그 이벤트를 어떻게 자바 클래스로 전달해 처리하는지에 대해 기술하고 있다. 본 논문에서는 2 장에서 관련 연구에 대해 기술하고, 3 장에서는 자바 이벤트 모델을, 4 장에서는 이벤트 초기화 및 처리를 5, 6 장에서는 실험 환경 및 결론과 향후 연구과제에 대해 기술한다.

2. 관련 연구

2.1 X 윈도우 시스템

X 윈도우 시스템의 구조는 클라이언트-서버 방식에 기초를 두고 있다. 주로 워크스테이션 상이나 그래픽 디스플레이가 가능한 퍼스널 컴퓨터 상에서 실행되는 서버는 입출력을 관장하며, 윈도우를 생성하고 조작하고, 텍스트나 그래픽을 출력한다[3].



[그림 1] 클라이언트 서버 모델

2.1.1 X 윈도우 시스템 이벤트

이벤트란 X 서버가 클라이언트에게 보내는 통보로서 어떤 조건이 변경되었거나 무엇인가가 일어났다는 것을 알리는 것이다. 서버는 관련된 모든 클라이언트에게 각 이벤트를 보내며, 클라이언트는 이벤트의 유형을 살펴봄으로써 어떤 이벤트가 발생하였는지 파악한다. X 서버는 모든 이벤트를 하나의 이벤트 큐에 넣고, XNextEvent() 함수를 이용하여 이벤트 큐로부터 이벤트를 꺼낸다. 각 이벤트 타입(type)은 C 구조로 정의되고, XEvent 구조는 모든 이벤트 타입을 결합

한 C union이다. 모든 이벤트들은 아래와 같은 핵심적인 정보를 포함한다.

- 이벤트 타입
- 이벤트가 발생한 디스플레이
- 이벤트 윈도우
- 서버에 의해 마지막으로 처리된 요청의 일련번호
- 이벤트가 서버에 의해 생성되었는지 또는 다른 클라이언트에 의해 생성되었는지 표시하는 플래그

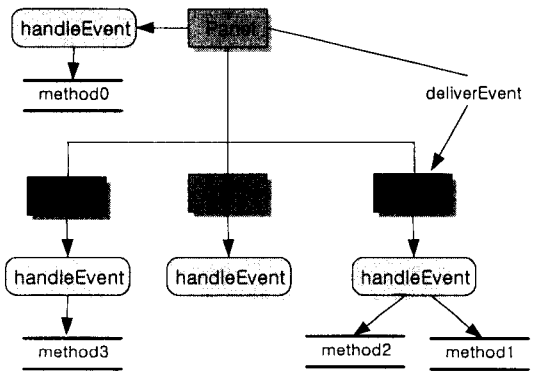
그밖에 이벤트를 구별해 주는 이벤트 마스크(Mask)가 있다.

3. AWT 이벤트 모델

이벤트(Event)란 GUI 컴포넌트에 가해지는 사용자로부터의 입력을 말하는 것으로 마우스 이벤트, 윈도우 이벤트, 키 이벤트 등이 있다.

3.1 Java 1.0 이벤트 모델

Java 1.0 에서 지원되는 이벤트 처리 모델은 상속을 기초로 구현된다. 이벤트가 발생되면 JVM 에 의해 모든 이벤트를 전달하기 위해 하나의 Event 클래스의 인스턴스가 생성된다. GUI 이벤트를 캐취(catch)하고 처리하기 위해서는 GUI 컴포넌트의 서브클래스에서 action()이나 handleEvent() 함수를 오버라이드 한다. 발생된 이벤트를 처리했으면 true 를 리턴하고, 그렇지 않으면 false 를 리턴하여 이벤트 소스(Source)의 부모(Parent)에게 이벤트 처리를 요구하게 된다 [4].

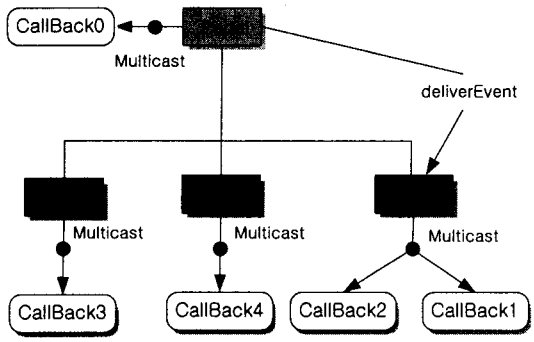


[그림 2] Java 1.0 Event Mechanism

[그림 2]에서 컴포넌트 Blue 가 이벤트 타겟(Target)이고, 컴포넌트 Panel 이 Blue 의 부모이다. Blue 에서 이벤트가 발생했을 때 그 이벤트는 부모로 전달되고, handleEvent() 함수가 Blue 클래스에서 오버라이드 되면 method1 또는 method2 에 의해 처리된다. 발생한 이벤트를 이 함수에서 처리하지 못한다면, 부모인 Panel 의 handleEvent() 함수가 호출되어 method0 에 의해 처리된다.

3.2 Java 1.1 이벤트 모델

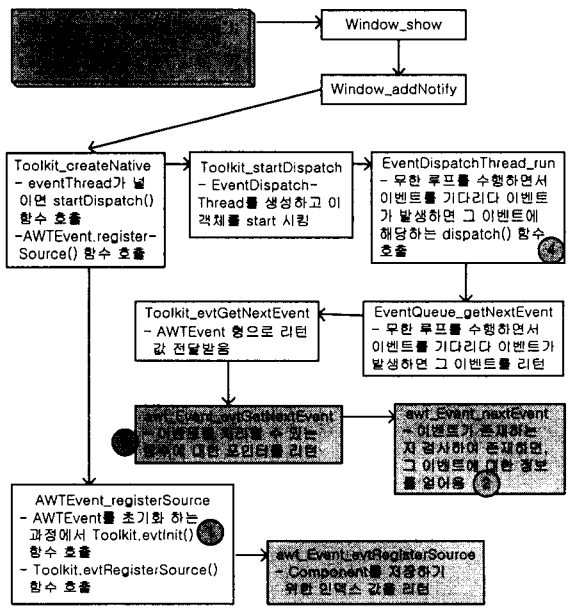
Java 1.0 이벤트 모델에서는 이벤트 타입에 따라 필터링(filtering)하지 못하고 하나의 함수를 통해서 구현되었다. 그러나 Java 1.1 이벤트 모델로 오면서 각각의 AWT 컴포넌트에 대해 그들 자신만의 리스너(Listener)를 등록하게 함으로써 리스너 interface 들을 통해 이벤트를 multicast 한다. 또한 processEvent() 함수를 사용해 이벤트를 처리하며, 모든 이벤트를 전달하기 위해 java.util.EventObject 를 상속받는 여러 개의 다른 이벤트 클래스를 사용한다. 그리고 특정한 컴포넌트가 발생시킬 수 있는 이벤트에 대한 이벤트 마스크를 유지한다. 이러한 새로운 접근은 특정한 이벤트가 타겟이 그것을 위해 기다리지 않는다면, 생성되거나 처리되지 않기 때문에 더 효율적이다[4][5].



[그림 3] Java 1.1 Event Mechanism

4. 이벤트 초기화 및 처리

본 논문에서 구현한 내용은 X 윈도우와 자바 AWT API 간 이벤트를 어떻게 주고 받으며 처리하는가에 대한 내용이다. Frame 이나 Window 객체를 생성한다고 해서 그 객체가 화면에 보여지거나 이벤트를 처리할 준비가 되는 것이 아니고, show()나 setVisible(true) 함수를 호출해야 한다. [그림 4] 에서는 X 윈도우에서 발생한 이벤트가 어떠한 과정을 통해 자바레벨로 전달되고, 어떻게 그 이벤트를 처리하는지에 대한 과정을 보여준다.



[그림 4] 이벤트 초기화 및 처리과정

4.1 이벤트 초기화

X 윈도우 시스템에서 발생한 이벤트를 자바레벨로 전달해 주기 위해선 X 윈도우 시스템에서 발생 가능한 각각의 이

벤트에 대해 자바레벨의 이벤트 처리 클래스와 연결 지어우어야 한다. [표 1]은 native 함수에서 X 윈도우 시스템과 자바레벨에서 사용될 마우스 이벤트에 대한 연결을 보여준다. 이 밖에도 ComponentEvent, FocusEvent, WindowEvent, KeyEvent, PaintEvent, WMEvent 가 있다.

[표 1] 이벤트 초기화

```
MouseEvent=(*env)->FindClass(env,
  "java/awt/MouseEvt");
<생략>
getMouseEvent=(*env)->GetStaticMethodID(env,
  MouseEvent,"getEvent",
  (IIIII)Ljava/awt/MouseEvt;");
```

4.2 이벤트 정보를 얻는 과정

X 윈도우 시스템에서는 이벤트가 발생하게 되면 이벤트 큐에 이벤트들을 저장하게 된다. 이벤트 큐에 저장된 첫번째 이벤트에 대한 정보를 얻어오기 위해 XNextEvent() 함수를 사용한다. [표 2]에서는 다음 이벤트에 대한 정보를 얻어오는 과정을 보인 것이다. preFetched 와 pending 필드를 사용하는 이유는 이벤트에 대한 정보를 얻어올 때 좀더 효율적으로 사용하기 위함이다. 이벤트 큐에 이벤트가 존재하면 그 이벤트에 대한 정보를 X->event 구조체에 저장하고, 펜딩(pending) 되어있는 이벤트 개수를 하나 줄이며, 이벤트 정보를 받았다는 결과로 1 을 리턴한다.

[표 2] 다음 이벤트 정보를 얻어오는 과정

```
int nextEvent(JNIEnv* env, jclass clazz, Toolkit *X,
  int blockIt) {
    if (X->preFetched)
      return 1;
    if (X->pending <= 0) {
      if ((X->pending = XEventsQueued(X->dsp,
        QueuedAfterFlush)) == 0)
          return 0;
      }
      XNextEvent(X->dsp, &X->event);
      X->pending--;
          return 1;
  }
```

4.3 X 윈도우 레벨 이벤트 처리

[표 2]에서 얻어온 X->event 구조체를 분석해 이 이벤트가 어떤 이벤트인지 판별해 그 이벤트를 처리할 함수를 호출한다. [표 3]은 위에서 판별한 이벤트가 마우스 이벤트일 경우이다. 마우스 이벤트를 처리할 수 있는 native 레벨 함수에서는 자바레벨에서 마우스 이벤트를 처리할 수 있는 함수를 CallStaticObjctMethod() JNI 함수를 호출하여 이벤트 타입이나 이벤트 소스, 좌표 등과 같은 이벤트 정보를 전달한다.

[표 3] Native 레벨에서 이벤트 처리 과정

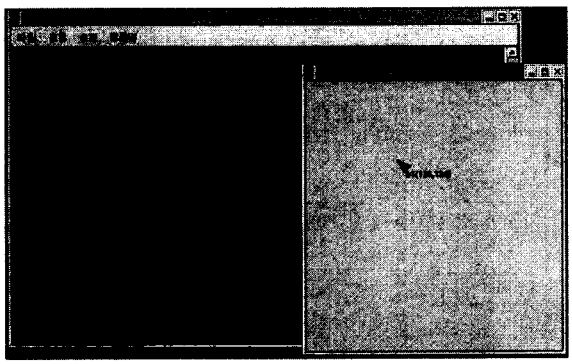
```
jobject mouseNotify(JNIEnv* env, Toolkit* X)
{
  X->evtId = (X->event.xany.type == EnterNotify) ?
    MOUSE_ENTERED : MOUSE_EXITED;

  return (*env)->CallStaticObjectMethod(env,
    MouseEvent, getMouseEvent, X->srcIdx,
    X->evtId, 0, X->event.xcrossing.x,
    X->event.xcrossing.y);
  }
```

4.4 자바 레벨 이벤트 처리

[표 3]에서 리턴된 값은 마우스 이벤트를 처리할 수 있는 자바 클래스 객체에 대한 포인터이다. 이 객체 포인터를 이용해 이벤트를 처리하기 위해 MouseEvt 클래스에 있는 dispatch() 함수를 호출하여 어떤 마우스 이벤트(ENTERED, MOVE, PRESSED 등)가 발생했는지 판단하여 그 이벤트를 처리한다.

5. 실험환경 및 결과

실험환경은 Red Hat Linux 7.2 운영체제에 Kaffe 1.0.7 JVM 과 클래스 라이브러리는 Kaffe 1.0.7 의 클래스 라이브러리에 자체 개발한 AWT API 를 대체하였다. 컴파일러는 Java Development Kit 1.2.2 의 javac 와 gcc 를 사용하였다. 또한 native 함수는 C 와 JNI 를 이용해 구현하였다. 본 논문에서 결과를 검증하기 위해 JDK1.1.8 Class Library 에제에 있는 마우스 이벤트를 처리하는 프로그램을 수행하였다. 그 결과 [그림 5]처럼 마우스 이벤트를 받아 처리함을 확인할 수 있었다.



[그림 5] 마우스 이벤트 처리 결과

6. 결론 및 향후 과제

본 논문에서는 자바가상머신에서 그래픽 환경을 제공해주는 GUI 의 일 부분인 AWT API 를 구현하는데 있어 X 윈도우에서 발생한 이벤트를 어떻게 자바레벨에 전달하고 처리할 것인가에 대한 방법에 대해 설계 및 구현을 하였다. 향후 연구과제로는 자바 AWT 에서 gif 나 jpeg 과 같은 이미지 파일을 처리하는 방법과, 한글 폰트 지원에 관한 문제에 대한 연구를 진행할 예정이다.

7. 참고문헌

[1] http://www.inestech.com
[2] http://java.sun.com
[3] Douglas A. Young, *The X Window System Programming and Applications With Xt,* 1994
[4] http://purana.cas.iisc.ernet.in
[5] Delegation Event Model http://developer.java.sun.com/developer/Books/GJ21AWT/ch9.pdf
[6] Patrick Chan, Rosanna Lee, *The Java Class Libraries,* 2nd Edition.
[7] http://java.sun.com/products/jdk/1.2/docs/api/index.html