안드로이드 기반 제스처 구성 및 제스처 프로그래밍 기법

최수경*, 박영호*
*숙명여자대학교 멀티미디어과학과
e-mail: {sukyungchoi, yhpark}@sookmyung.ac.kr

A Study on Gesture Organization and Programming based on Android Platform

Su-Kyung Choi*, Young-Ho Park*
*Dept of Multimedia Science, Sookmyung Women's University

요 약

터치 스크린은 모바일 기기에서 애플리케이션과 상호 작용할 수 있는 좋은 방법이다. 사용자들은 스크린 상에서 다양한 액션을 취할 수 있으며, 안드로이드 프레임워크는 다양한 제스처를 지원한다. 본 논문에서는 안드로이드 API 1.6부터 지원된 새로운 제스처에 대해 분석하고 간단한 프로그래밍 예도 소개하고자 한다.

1. 서론

모바일 장비의 주 입력 장치는 터치 스크린이며 화면이 출력은 물론 입력까지 겸한다. 최근에는 스크린에서의 단순한 터치 뿐 만이 아니라 화면을 누르는 동작, 가볍게 두드리는 탭, 눌러서 끄는 드래그 같은 기본 동작에서부터 더블 탭, 롱 프레스 등의 응용 동작 까지 가능하다. 따라서 본 논문에서는 이러한 안드로이드의 제스처 입력 원리를 분석해보고 간단한 프로그래밍 예를 소개하고자 한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 사용자의 터치 입력을 논리적으로 해석하는 제스처 기법에 대해 분석한다. 제 3장에서는 여러 손가락의 입력을 동시에 받아들여 고수준의 명령으로 해석하는 멀티 터치 기법에 대해 설명한다. 제 4장에서는 프로그래밍 구현 결과를 보이고, 제 5장에서는 본 논문의 결론을 맺는다.

2. 제스처 기능

본 장에서는 안드로이드의 제스처 기능에 대해 설명하고 커스텀 제스처에 대한 프로그래밍 예를 소개한다.

2.1 기본 기능

안드로이드 모바일 디바이스는 터치화면을 통한 제스처 기능을 제공한다. 제스처 기능이란 스크린의 동작을 감지하는 것이다. 디바이스 자체의 센서를 감지하는 것은 아니고 화면 내의 터치를 감지하여, 어떠한 이벤트가 들어왔는지 분석하여 어떤 기능을 수행하도록 하는 것이다.

제스처는 일련의 터치 입력을 받아 패턴을 분석함으로써 다양한 명령을 감지하는 기능이다. 하드웨어로부터 입력된 정보를 그대로 전달하는 것이 아니라 누르거나 뗀 위치, 입력된 순서나 시간, 입력 속도, 이동 방향 등을 고려하여 2차적인 고수준의 명령을 생성해낸다. 여러 가지 조건을 종합하여 논리적인 명령으로 인식하므로 탭이나 드래그에 비해 훨씬 더 복잡한 동작을 입력받을 수 있다 [1][2].

2.2 동작 원리

제스처는 여러 터치 사건들의 관계로부터 정의되므로 일련의 터치 입력을 모아서 분석해야 한다. 가장 간단한 동작인 탭은 눌렀다 뗄 때 성립한다. 누른 자리에서 바로 떼야 하며 이동해서도 안 되고 누른 상태로 시간을 오래 끌어도 안 된다. 누른 자리에서 신속하게 바로 떼야 탭으로 인정된다.

더블 탭 동작의 경우 시간과 공간의 두 가지 조건을 만족해야 한다. 일정한 면적 안에서 일정한 시간 안에 터치가 연속으로 들어올 때만 더블 탭으로 인정된다.

제스처 감지를 직접 하려면 터치가 입력되는 onTouchEvent 메서드에서 전달된 모든 이벤트를 배열에 모아 두었다가 일련의 이벤트들을 분석해야 한다. 여기에는 이벤트 수집과 분석을 대신해 주는 클래스가 제공된다. (그림 1)의 GestureDetector 클래스는 터치 이벤트로 전달된 MotionEvent 객체를 모아 두었다가 어떤 제스처인지 분석하여 그 결과를 리스너로 통보한다.

```
GestureDetector (Context context,
GestureDetector.OnGestureListener listener, [
Handler handler, boolean ignoreMultitouch ])
```

(그림 1) GestureDetector 클래스

첫 번째 인수는 감지를 요청하는 컨텍스트이되 통상 액티비티 자신인 this를 전달한다. 두 번째 인수가 제스처 감지시 호출될 리스너이며 실질적으로 제일 중요한 정보이다. 나머지 두 인수는 신호를 받을 핸들러와 멀티 터치 무시 여부를 지정하되 필요 없을 시 생략 가능하며 보통 생략한다. 제스처 감지기는 터치 입력을 받아야 하므로 반드시 UI 스레드에서 생성해야 한다.

액티비티는 감지기 객체를 생성한 후 onTouchEvent 메서드에서 반드시 감지기의 onTouchEvent 메서드를 호출하여 터치 정보를 전달해야 한다. 감지기는 전달된 터치 정보를 차곡 차곡 모아 두었다가 터치 이벤트를 분석하여 제스처를 감지해 내며 이때 리스너의 대응되는 콜백 메서드를 호출한다.

2.3 커스텀 제스처

SDK 1.6부터는 기본동작들을 능가하는 새로운 API가 추가되어 더 복잡한 제스처도 추출해 낼 수 있게 되었다. 기본 감지기는 주로 점과 직선의 단순 동작만 감지하는데 비해 새로 추가된 API는 동그라미나 별모양 같은 복잡한 도형까지도 감지해낸다. 모양을 인식해 낸다는 점에서 거의 OCR 수준의 제스처 추출이 가능하다.

새로운 제스처 기능을 사용하려면 먼저 제스처의 모양부터 정의해야 하며 이때는 제스처 빌더를 사용한다. (그림 2)와 같이 원하는 제스처를 입력하고 등록할 수 있다 [3][4].

(그림 2) 커스텀제스처

정의한 제스처는 SD 카드의 루트에 gestures라는 이진 파일로 저장되며 제스처의 모양과 이름들이 기록되어 있다. 제스처 빌더로 정의한 제스처를 프로젝트에서 사용하려면 제스처 정의 파일을 리소스의 res/raw 폴더에 복사해 넣고 제스처 라이브러리로 읽는다.

사용자로부터 제스처를 입력받으려면 제스처를 그릴 영역이 필요한데 이 영역은 GestureOverlayView 클래스가 제공한다. onCreate에서 fromRawResource 메소드로 제스처 라이브러리를 초기화한다. 인수로 제스처 리소스 ID를 전달하고 load 메서드를 호출하면 리소스에 정의된 제스처들이 읽혀진다. 제스처 오버레이 뷰에게 감지 리스너를 전달하면 이후부터 감지가 시작된다. 사용자는 오버레이 뷰 위에 도형을 마음대로 그릴 수 있으며 오버레이 뷰는 사용자가 도형 하나를 완성할 때마다 리스너의 onGesturePerformed 메서드를 호출한다. 이때 gesture 인수로 사용자가 그린 도형이 전달된다. 콜백은 제스처 하나를 받을 때 라이브러리의 recognize 메서드로 일치하는 제스처가 있는지 조사한다. 다음 장에서는 이러한 제스처를 확장시킨 멀티 터치 기법에 대해 설명한다.

3. 멀티 터치 기법

본 장에서는 멀티 터치 기법의 기본 기능과 핀치 줌에 대해 설명한다.

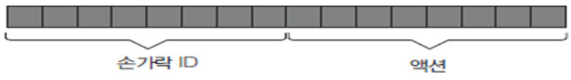
3.1 기본 기능

멀티 터치는 복수 개의 손가락으로 화면을 조작하는 입력 방법이다. 한 손가락으로 누르거나 끌기만 하는 단일 터치에 비해 여러 손가락의 동작을 조합하여 명령을 내리므로 복잡한 명령을 간편하게 전달할 수 있다.

안드로이드에서는 2.0 버전부터 지원되기 시작했으며 초기 버전에는 상당히 버그가 많았으나 SDK가 지속적으로 업그레이드되고 장비의 성능도 향상되면서 요즘은 차츰 개선되고 있다. 멀티 터치를 위해 별도의 클래스나 리스너가 따로 추가되지는 않았으며 터치를 받는 onTouchEvent 메서드가 멀티 터치를 인식한다.

여러 개의 터치가 동시에 전달될 수 있어야 하므로 터치 정보를 전달하는 MotionEvent 객체가 주로 확장되었다. 기존에는 DOWN, MOVE, UP 등의 액션과 좌표 정도만 전달되엇으나 복수 터치에 대한 액션이 추가되고 각 터치별 좌표도 전달된다. 첫 번째 손가락에 대해서는 기존의 액션이 동일하게 전달된다는 점에서 이전 방식과 동일하다. 그러나 두 번째 이후의 손가락에 대해서는 ACTION_POINTER_DOWN과 ACTION_POINTER_UP 액션이 추가로 전달 된다.

getAction이 조사하는 액션의 값 의미도 변경되었다. 이전에는 액션의 종류만 조사했으므로 정수값 하나만 리턴하면 되었지만 여기서는 몇 번째 손가락으로부터의 액션인지도 같이 조사해야 한다. 두 값을 하나의 정수로 리턴하기 위해 상하위 비트를 나누고 할당한다. (그림 3)과 같이 손가락의 인덱스에 8비트가 할당되어 있으므로 안드로이드는 이론상으로 256개의 손가락 동시 터치가 가능하다. 그러나 실제로는 장비의 화면 하드웨어가 받아들일 수 있는 동치 터치수에 제약이 있어 대부분 5개의 손가락까지 동시 터치를 인식해 낸다.



(그림 3) 인덱스 비트 할당

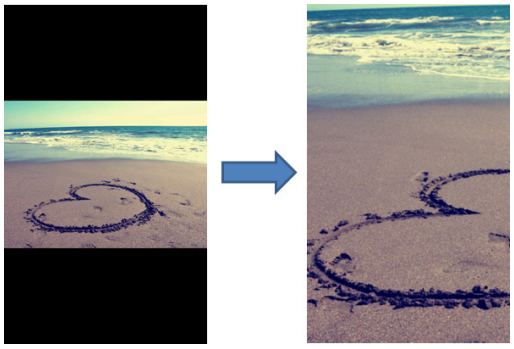
멀티 터치를 지원하는 프로그램은 getAction 메서드로 조사한 값으로부터 액션 종류와 손가락 인덱스를 추출해야 한다. 액션값만 추출하려면 하위 8비트만 제외하고 마스크 오프시키면 되므로 0xff와 & 연산을 취해 간단하게 구할 수 있다. 0xff가 ACTION_MASK 상수로 정의되어 있으므로 이 값과 &를 취하면 된다.

3.2 핀치 줌

멀티 터치의 가장 실용적인 예는 두 손가락의 거리를 조정함으로써 확대, 축소를 하는 핀치 줌이다. 핀치 줌의 구현은 둘 이상의 손가락이 눌러졌을 때 초기 거리를 구해 놓고 거리의 변화에 따라 배율만 조정하면 된다. 두 번째 손가락이 화면에 접촉되는 ACTION_POINTER_DOWN 액션이 발생할 때 확대/축소가 시작된다. 이때 초기 거리와 확대/축소를 시작할 때의 배율을 구해 놓는다. 이동 중에도 거리를 계속 구해야 하므로 getDistance 메서드를 따로 정의해 둔다. 이후부터는 ACTION_MOVE가 날아올 때 두 손가락의 거리값이 어떻게 바뀌었나를 계산하여 배율을 조정한다. 제 4장에서는 이러한 핀치 줌을 활용한 프로그래밍의 구현 결과를 보인다.

4. 구현 결과

본 장에서는 멀티 터치의 대표적인 예인 핀치 줌의 구현 결과를 보인다. (그림 4)의 왼쪽의 사진을 두 손가락으로 동시에 벌리면 오른쪽 사진과 같이 확대된다.



(그림 4) 구현 결과

5. 결론

본 논문에서는 사용자의 터치 입력을 논리적으로 해석하는 제스처 기법과 여러 손가락의 입력을 동시에 받아들여 고수준의 명령으로 해석하는 멀티 터치 기법에 대해 분석해 보고 프로그래밍 구현 결과를 보였다. 향후 이런 기법들을 잘 활용하면 사용자에게 다이나믹한 인터페이스와 더 풍부한 경험을 제공하는 애플리케이션을 만들 수 있을 것으로 기대한다.

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2011-0002707)

참고문헌

[1] 리토 마이어, "프로페셔널 안드로이드 개발," 2010, 제이펍

[2] 김상형, "안드로이드 프로그래밍 정복," 2010, 한빛미디어

[3] Kandroid, http://www.kandroid.org

[4] Android Debelovers, http://developer.android.com