한국컴퓨터정보학회 하계학술대회 논문집 제20권 제2호 (2012. 7)

스마트폰 가속도 센서를 이용한 모션 제스쳐 프레임워크의 설계 및 구현

이기문*, 이신영*, 김종현°
*KAIST 무학과
°동의과학대학교 컴퓨터정보계열
e-mail:{gaianofc, class255}@kaist.ac.kr*, jhkim@dit.ac.kr°

Design and Implementation of a Motion Gesture Framework based on the Smart Phone Accelerometer Sensor

Gimun Lee*, Shinyoung Yi*, Jonghyun Kim°
*Korea Advanced Institute of Science and Technology
°Dept. of Computer & Information, Dongeui Institute of Technology

● 요 약 ●

본 논문에서는 스마트폰 가속도 센서를 이용한 새로운 사용자 인터페이스인 모션 제스쳐 프레임워크를 제안한다. 최근 IT 패러다임의 변화를 주도하고 있는 스마트폰의 대표적인 특징 중의 하나가 터치 인터페이스이다. Apple 사의 iPhone은 세계 최초로 터치스크린으로 가능한 사용자 입력들을 체계적으로 정리하여 터치 제스처를 정의하고 상용화에 성공 하였다. 본 연구에서는 기존의 터치 인터페이스를 대체하거나 보완할 새로운 인터페이스로서 스마트폰의 가속도 센서를 이용한 모션 제스처 프레임워크를 설계하고, 실제 스마트폰 응용 프로그램의 사용자 인터페이스에 적용하여 활용 가능성을 보여준다.

키워드: 모션(motion), 제스쳐(gesture), 가속도 센서(Accelerometer)

I. 서론

최근 Apple 사의 iPhone을 선두로, 스마트폰은 전 세계에 새로운 IT 혁명을 일으키고 있다. 이제 많은 사람들이 기존의 PC에 발을 묶이지 않고, 언제 어디서나 자신의 단말기를 꺼내 원하는 작업을 할 수 있는 IT 패러다임의 변화가 도래하였다.

스마트폰이 가지고 있는 다양한 센서는 기존의 PC에서 다룰 수 없었던 새로운 형태의 애플리케이션 개발을 가능하게 한다. 이를테면 입으로 바람을 불면 화면 안의 화톳불이 흔들린다든지, 세게 흔들면 마치 화면이 부서지는 듯한 효과를 내는 등의 애플리케이션을 들 수 있다. 이와 같이 주위의 소리, 온도, 디바이스의 위치, 움직임 등을 인식하여 애플리케이션이 사용자가 원하는 편이에 훨씬 가까이 다가갈 수 있다[1].

특히 디바이스의 움직임을 감지하는 가속도 센서와 자이로스코프 센서가 iPhone4에 탑재되어 많은 이들에게 각광받고 있다. 1인칭 시점의 슈팅 게임이나 레이싱 게임과 같은 경우는 기존의 게임기에 없었던 모션 인터페이스를 지원하여 보다 실감나는 플레이가 가능함을 보여준다[3].

한편, 터치스크린 기술은 이미 PC 등 다양한 장비에서 차세대 인터페이스 기술로 각광받고 있다. 그러나 스마트폰 상에서의 터치스크린은 Apple 사에서 사용자의 가능한 입력들을 체계적인 제스처들로 정리하여 비로소 직관적으로 사용할 수 있게 되었다. 사용자들은 현재 스마트폰 등 스마트 기기에서 손가락을 벌리거나(Pinch) 두 손가락을 쓸어 넘기는 것이 보편화되었다[1,2].

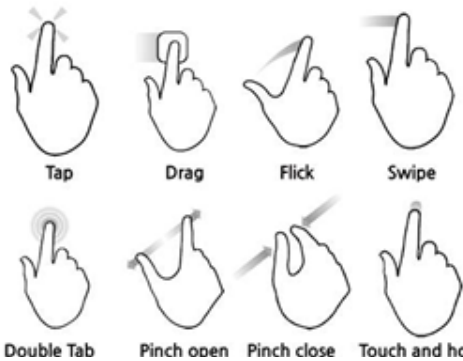

그림 1. Apple 사에서 제안한 터치스크린 제스처
Fig. 1. Touch Screen Gesture proposed by Apple inc.

본 논문에서는 iPhone iOS에서 사용자들의 가속도, 회전 각속도 값들을 인식하는 모션 제스처 사용자 인터페이스 프레임워크를 설계하고 기존의 스마트폰 응용 프로그램에 적용하였다. 본 논문

에서 제안한 모션 제스처를 이용하면 사용자들은 보다 편리한 디바이스 조작이 가능하다. 웹툰이나 음악 목록 등의 기다란 컨텐츠를 볼 때, 원격 프리젠테이션, 스마트 TV 리모콘 등 다양한 스마트 디바이스에 적용되어 새로운 형태의 사용자 인터페이스를 제공한다. 모션 제스처는 스마트 기기들이 점점 다양화되어 가고 있는 시대에 기존의 터치 인터페이스를 대체하거나 보완할 수 있는 새로운 방향으로 발전할 수 있는 가능성을 제시한다.

II. 관련 연구

1. 가속도 센서(Accelerometer)

iPhone의 Accelerometer는 중력 센서의 기능을 한다. 이것은 디바이스에 어떤 방향으로 어느 정도의 가속도가 작용하는지 탐지하고 그 값을 이용하여 사용자에게 여러 가지 편의를 준다. 이것이 이용되는 가장 기본적인 기능으로는 디바이스를 세로로 들 때는 세로로 된 화면을 보여주고, 가로로 들고 있을 때는 화면을 자동으로 회전시켜 가로로 넓은 화면을 보여주는 화면 방향 전환 기능 등이 있으며, 개발자들은 이를 이용해 다양한 어플리케이션이 개발되고 있다. 최근에는 자동차용 운행기록장치(블랙박스)에도 탑재되어 차량 충돌 시 충돌 방향을 기록하는 등 다양하게 활용되고 있다.

그림. 2 iPhone 가속도 센서의 데이터 형태
Fig. 2. Data type of the iPhone Accelerometer Sensor

2. 터치 제스쳐

iPhone에서 터치 스크린은 정전식으로 화면에 미약한 전류를 흘려 사용자의 몸이나 터치펜 등 전도체로 흘러 들어가는 지점을 탐지하여 위치 정보를 제공한다. 아이폰의 터치스크린은 정전식이기 때문에 멀티터치의 구현이 쉬운데, 애플은 이러한 이점을 적극 활용하여 여러 가지 터치 제스처들을 구현하였다[1,2,3].

III. 설계 및 구현

1. 설계

UIGestureManager 클래스는 개발자가 원하는 시기동안 주기적으로 현재 디바이스의 상태를 인식하여 필요할 때 타겟 객체에게 메시지를 전달한다. 싱글톤 패턴으로 UIGestureManager 객체가 필요할 때 인식을 하며 델리게이트에게 메시지를 보내면 타겟에 카테고리로 추가해놓은 델리게이트 메소드들이 실행된다.

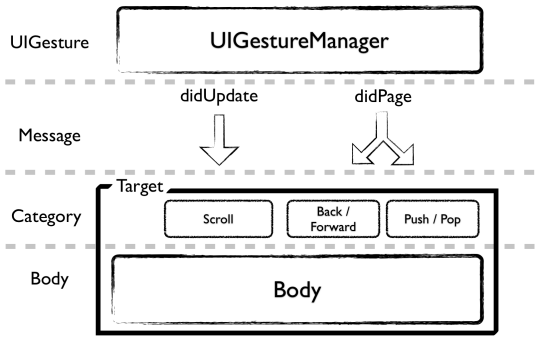

그림. 3 UIGestureManager와 Target 구조
Fig. 3. Architecture of UIGesture and Target

1.1 UIGestureManager 클래스

UIGestureManager가 타겟 객체와는 별도로 존재하되, 타겟이 UIGestureManager의 델리게이트로써 존재하여 UIGesture가 때가 되었을 때 자신의 델리게이트에게 메시지를 보낼 수 있다.

1.2 UIGestureManagerDelegate 카테고리

타겟 클래스에 카테고리로써 우리가 필요로 하는 메소드들을 추가하여 타겟에게 UIGesture관련 기능을 추가하는 것이다. 이는 기존의 작업을 센서를 이용해 컨트롤하도록 확장시키자는 본래의 취지에 알맞다. 또한 카테고리로써 메소드를 만드는 것은 그 객체 스스로에게 기능을 추가하는 것이기 때문에 객체 내부의 멤버나 다른 메소드에 접근하기도 자연스럽다. 또한 무엇보다 상위의 클래스에 카테고리를 추가하는 것만으로도 이를 상속하는 다른 클래스들에게도 같은 내용이 들어가므로 훨씬 체계적이다.

2. 세부 알고리즘

2.1 UIGestureManager

별도의 객체인 UIGestureManager 클래스는 CMMotionManager를 상속하여, 자신에게 startUpdateAccelerometer, startUpdateGyro 메시지를 보낸다. 이와 함께 하나의 타이머를 추가해 제스처 여부를 확인하는 update 메소드를 주기적으로 부른다. 또한 시작하면서 현재 디바이스 자세로부터 phi값과 theta값을 계산해 저장한다.

update 메소드에서는 (1) 기울이기와 (2) 페이지 넘기기를 확인한다.

가속도센서의 값을 이용해 지금의 phi값과 theta값을 구하여, 시작할 때 저장했던 값과 비교하여 얼마나 기울어졌는지를 인식한다. 이 각의 변화를 이차함수를 통해서 스크롤해야하는 양으로 바꾸어 멤버 변수에 저장한다. 기울이기는 매 업데이트 때마다 전달해주어야 하므로, update 메소드마다 꼭 한번씩 델리게이트에게 didUpdate 메시지를 보내 새로운 값으로 갱신됨을 알린다.

자이로스코프 센서의 값도 확인하여 특정 각속도 이상으로 돌아가고 있다면 델리게이트에 didPage 메시지를 보낸다. 이 때, 직후 일정 시간 동안은 이 확인을 생략한다. 실제 돌리는 행동은 순간적인 것이 아니기 때문에, 한 번의 모션에서 여러 번 인식하는 것을 방지하기 위해서다.

```
@interface UIGestureManager : CMMotionManager {

    id<AngleManagerDelegate> _delegate;

    BOOL isRunning; //메시지를 보내야하는지 여부
    BOOL isAccReady; //가속도센서를 업데이트할 초기 설정 여부
    BOOL isGyroReady; //자이로스코프 센서를 업데이트할 초기 설정 여부

    Posture _initPos; //기준 방향
    Posture _currPos; //현재 방향

    CGPoint _scroll; //각도 차에 의해 계산된 스크롤양

    NSTimer* _timer; //주기적으로 update를 부르는 타이머
};

@property(nonatomic, retain) id<AngleManagerDelegate> delegate;

@property(nonatomic, assign) Posture initPos;
@property(nonatomic, assign) Posture currPos;
@property(nonatomic, assign) CGPoint scroll;

@property(nonatomic, retain) NSTimer* timer;

- (id) init;
- (void) update; //매 프레임마다 모션 인식

- (void) startUpdate;
- (void) stopUpdate;

@end
```

그림. 4 UIGestureManager의 정의
Fig. 4. Definition of UIGestureManager

2.2 모션 인식과 스크롤

2.2.1 회전 각속도 계산

현재 가해지는 중력의 벡터를 내적을 이용해 옆으로, 앞뒤로 얼마나 회전해있는지 각 phi와 theta를 구한다.

procedure Compute_Accelerometer_Position
{
// 중력센서 값으로 Device가 기울어진 각도 계산
Vector3D : 현재 중력가속도 벡터 (gX,gY,gZ)
Double theta := $\cos^{-1}\frac{-g_z}{\sqrt{g_y^2+g_z^2}}$ *// Device의 y축과 수평면이 이루는 각도(rad)*
if (gY > 0) : theta = -theta // Revise theta to be in [-π,π]
Double phi := $\cos^{-1}\frac{\sqrt{g_x^2+g_y^2}}{\sqrt{g_x^2+g_y^2+g_z^2}}$ *// Device의 x축과 수평면이 이루는 각도(rad)*
// Revise phi to be in [-π,π]
if (gX > 0 & gZ > 0) : phi -= π
if (gX > 0 & gZ <= 0) : phi = -phi
if (gX <= 0 & gZ > 0) : phi = π - phi
}

2.2.2 스크롤 양 계산

디바이스가 돌아간 정도로부터 얼마나 스크롤할지를 계산한다. 사용자는 스크롤양이 각에 선형인 것 보다는 점점 더 빨라지는 것을 원하기 때문에 2차함수를 적용했다.

procedure Scroll by_Acceleration
{
Double thetaInitial, phiInitial // 프로그램 실행 시의 theta, phi값
Double thetaCurrent, phiCurrent // 현재 theta, phi값
thetaD = thetaInitial - thetaCurrent
// theta의 변화량
phiD = phiInitial - phiCurrent // phi의 변화량
Revise (thetaD), (phiD) to be in // add or subtract π

// Scroll
2DVector deltaOffset :=
(AngleToSpeed(phiD),
ANgleToSpeed(-thetaD))
MoveScroll(deltaOffset)
}

procedure AngleToSpeed(ang)
{
// 기울어진 각도에 따라 스크롤 할 정도 리턴
$speed = ang^2 \times 200 + |ang| \times 40$
if (speed > 500) : speed = 500 // Maximum Limit
*speed *=* $\frac{ang}{|ang|}$ *// ang와 부호와 같도록*
return speed
}

4. 구현 및 평가

4.1 UITableView 적용

UITableView가 가지는 주요 동작은 화면 위아래로 목록을 움직이는 것이다. 그래서 옆으로 기울인 것은 무시하고 앞뒤로 기울인 phi값으로부터 위아래 스크롤양을 계산한다. 목록이 굉장히 길 때 편리하게 디바이스를 앞뒤로 기울이는 것만으로도 전체를 훑어볼 수 있다. 이는 기존의 손가락으로 바쁘게 드래그해야만 가능한 일이었다.

4.2 UIWebView 적용

UIWebView가 가지는 주요 동작들은 화면을 움직이는 스크롤과

웹서핑 중 'page back' 버튼으로 돌아가고, 다시 'page forward' 버튼으로 돌아가는 것이다. 그래서 기울인 정도를 통해 스크롤하고, 좌로 넘기면 뒤로, 우로 넘기면 앞으로 가도록 하였다. 이용자는 터치 모드와 모션 모드를 아래쪽에 있는 컨트롤을 이용해 선택할 수 있다.

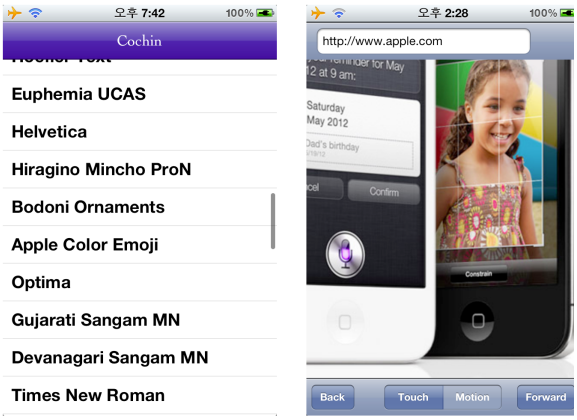

그림. 6 UITableView와 UIWebView 구현
Fig. 6. Implementation of UITableView And UIWebView

IV. 결론

본 논문에서는 스마트폰에서 가속도, 회전 각속도 값들을 인식하는 모션 제스처 프레임워크를 설계하고 기존의 스마트폰 응용 프로그램에 적용하였다. 우리는 기울이기와 페이지 넘기기 모션들을 가속도 센서의 값을 통해 인식하여 델리게이트에 메시지를 보내는 UIGestureManager 클래스를 설계하였고, 이를 기존의 iOS UIKit 프레임워크에서 UITableView와 UIWebView에 적용하였다. 현재 모션 위치 인식 알고리즘은 디바이스가 지상에 거의 수직하면 다소 불안정면이 있다. 따라서 y축 가속도 값이 어느 정도 커지면 자이로스코프 센서를 활용하여 좀 더 정밀한 방법으로 각을 인식해야한다. 또한 애초에 계획했던 터치 제스처를 대체할 만한 모션 제스처의 체계화를 위해서는 아직 더 풍부한 제스처 모델들이 필요하다. 그렇지만 이후 다른 제스처들을 위한 연구 역시 이번 연구의 틀을 크게 벗어나지 않을 것이기 때문에, 비록 제스처의 범위가 좁지만 충분히 의미 있었다.

참고문헌

[1] Alasdair Allan, "Programming the Accelerometer, Gyroscope, Camera and Magnetometer, Basic Sensors in iOS" O'REILLY, 2011.

[2] Apple Inc. "iOS Human Interface guidelines", 2011.10.12

[3] 토코로 유타, 김은철 역, "아이폰 프로그래밍 UIKit 핵심 바이블" 2011.

[4] Apple Inc. "The Objective-C 2.0 Programming Language", 2008.6.9.