http://dx.doi.org/10.5392/JKCA.2013.13.01.055

GUI 어플리케이션 제어를 위한 제스처 인터페이스 모델 설계

Design of Gesture based Interfaces for Controlling GUI Applications

박기창*, 서성채**, 정승문*, 강임철*, 김병기**
디지털콘텐츠 협동연구센터*, 전남대학교 전산학과**

Ki-Chang Park(kcpark@dsu.ac.kr)*, Seong-Chae Seo(scseo@jnu.ac.kr)**,
Seung-Moon Jeong(jsm@dsu.ac.kr)*, Im-Cheol Kang(softkang@dsu.ac.kr)*,
Byung-Gi Kim(bgkim@jnu.ac.kr)**

요약

사용자 인터페이스 기술은 CLI(Command Line Interfaces), GUI(Graphical User Interfaces)를 거쳐 NUI(Natural User Interfaces)로 발전하고 있다. NUI는 멀티터치, 모션 트래킹, 음성, 스타일러스 등 다양한 입력형식을 사용한다. 기존 GUI 어플리케이션에 NUI를 적용하기 위해서는 이러한 장치관련 라이브러리 추가, 관련 코드 수정, 디버그 등의 과정이 필요하다. 본 논문에서는 기존 이벤트 기반 GUI 어플리케이션의 수정 없이 제스처 기반 인터페이스를 적용할 수 있는 모델을 제안한다. 또한 제안한 모델을 명세하기 위한 XML 스키마를 제시하고, 3D 제스처와 마우스 제스처 프로토타입 개발을 통해 제안모델의 활용방안을 보인다.

■ 중심어 : | 인간-컴퓨터 상호작용 | 내추럴 유저 인터페이스 | 제스처기반 인터페이스 | 마크업 언어 |

Abstract

NUI(Natural User Interfaces) has been developed through CLI(Command Line Interfaces) and GUI(Graphical User Interfaces). NUI uses many different input modalities, including multi-touch, motion tracking, voice and stylus. In order to adopt NUI to legacy GUI applications, he/she must add device libraries, modify relevant source code and debug it. In this paper, we propose a gesture-based interface model that can be applied without modification of the existing event-based GUI applications and also present the XML schema for the specification of the model proposed. This paper shows a method of using the proposed model through a prototype.

■ keyword : | Human-Computer Interaction | Natural User Interfaces | Gesture-based Interface | Markup Language |

I. 서 론

내추럴 유저 인터페이스(Natural User Interface, NUI)는 몸짓, 음성 등 인간의 다양한 인식능력을 활용한 인간-컴퓨터 상호작용(Human Computer Interaction, HCI)기술이다. NUI는 명령어 기반의 CLI(Command Line Interface), 그래픽 기반의 GUI(Graphic User Interface)를 거쳐 발전된 기술로 여기에는 센싱 기술, 음성인식기술, 동작인식기술등 요소기술이 포함된다[1][2]. NUI를 이용한 연구 및 제품으로는 Kinect[3], EyeToy[4], PixelSense[5], 3D Immersive Touch[6], Dragon Naturally Speaking[7], Perceptive Pixel[8]등

접수번호 : #121121-002
접수일자 : 2012년 11월 21일
심사완료일 : 2012년 12월 20일
교신저자 : 박기창, e-mail : kcpark@dsu.ac.kr

이 있으며, 이를 활용한 게임, 교육용 콘텐츠에 대한 연구도 활발하다[9][10]. NUI 관련 기술 중 사람의 제스처를 활용한 제스처 기반 사용자 인터페이스는 제스처의 입력 장치에 따라 마우스 제스처, 터치 제스처, 3D 제스처로 구분할 수 있다. 마우스 제스처는 구글의 크롬, MS의 인터넷 익스플로러와 같은 웹 브라우저에서 메뉴나 단축키를 대신하여 특정명령을 실행할 수 있는 Plug-in 형태로 개발되어 활용되고 있다. 터치 제스처는 스마트 폰, 태블릿 PC, Digital Signage 시스템에서 마우스, 키보드 등의 인터페이스 장치를 대체하여 활용되고 있으며, 3D 제스처는 게임 콘솔에서 3차원 게임 및 가상현실 콘텐츠에 주로 활용되고 있다.

한편, GUI 어플리케이션은 WIMP(Windows, Icon, Menu, Pointer)기반의 그래픽 요소를 이용하여 사용자로 부터 발생하는 키보드, 마우스 등 입력디바이스로부터 발생하는 이벤트(Event)를 감지하고, 이벤트에 적합한 메시지(Message)를 발생시킴으로써 상호작용을 지원한다. 이러한 GUI 어플리케이션에 제스처 기반의 NUI를 적용하여 인터페이스를 개선하기 위해서는 제스처 인식 장치 추가에 따른 장치드라이버 로드, 장치 관련 코드의 추가, 이벤트 코드의 수정, 디버그 등의 과정을 거쳐서 어플리케이션을 개발해야 하므로 이에 따른 시간과 비용을 발생시킨다. 본 논문에서는 이미 개발된 GUI 어플리케이션에 제스처 기반의 NUI를 적용하기 위한 제스처 모델을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 NUI관련 기술을 소개하고, 3장에서는 GUI 어플리케이션에 NUI를 적용하기 위한 모델을 제안하고, 제안모델의 명세를 위한 XML 스키마를 제시한다. 4장에서는 웹 브라우저 제어를 통한 사례연구를 통해 제안 모델의 활용방안을 살펴보고, 5장에서는 결론 및 향후 연구방향을 논한다.

II. 관련연구

WIMP기반의 GUI 어플리케이션에서의 사용자 인터페이스 기술은 UML(Unified Modeling Language)을 이용한 객체지향적 명세 기법과, Object-Z, Petri Net등을 이용한 정형 명세(Formal Specification)기법 등의 다양한 접근법들이 연구되었다. XML의 등장으로 이를 이용한 UIML, AUIML, XIML, Seescoa XML, Teresa XML, WSXL, MIML, JIML 등이 연구되어 스탠드얼론, 웹, 모바일 등 어플리케이션의 다양한 실행 환경에서의 GUI 명세 및 구현에 적용되었다[11-13]. 터치, 제스처, 음성인식 등 NUI 환경에서는 멀티터치 인터페이스 환경에서 적용가능한 도메인 명세언어인 GDL (Gesture Definition Language)[14]이 연구되었지만 터치 제스처 명세만을 지원한다.

3D 카메라를 통한 제스처 인식을 기존 응용에 적용하기 위한 연구로는 미국 USC(University of Southern California)의 ICT(Institute for Creative Technologies)에서 개발한 FAAST(the Flexible Action and Articulated Skeleton Toolkit)가 있다[15]. FAAST는 Kinect 센서를 통해 사용자의 골격모델(Skeletal Model)을 기반으로 제스처를 인식하고 이를 마우스와 키보드 이벤트로 변환하여 VR 어플리케이션 및 게임을 컨트롤 할 수 있는 툴킷으로 body, position, angular, velocity, time등 5가지 형태의 제스처 입력형식과 keyboard, mouse, time등 8가지 출력형식을 지원한다.

그림 1. 제스처 인터페이스를 이용한 게임컨트롤

III. NUI를 위한 제스처 인식 모델

본 장에서는 기 개발된 GUI 어플리케이션에 제스처 인터페이스를 추가하기 위한 JUHA 모델과 JUHA 모델에서의 제스처-이벤트 정보를 명세하기 위한 JUHA XML 명세언어에 대하여 설명한다.

1. JUHA(Just Ui for Human Activity)모델

제스처를 이용하여 GUI 어플리케이션을 제어하기 위한 제스처 인식 모델로 정다각형을 이용한 JUHA(Just Ui for Human Activity)모델을 제안한다.

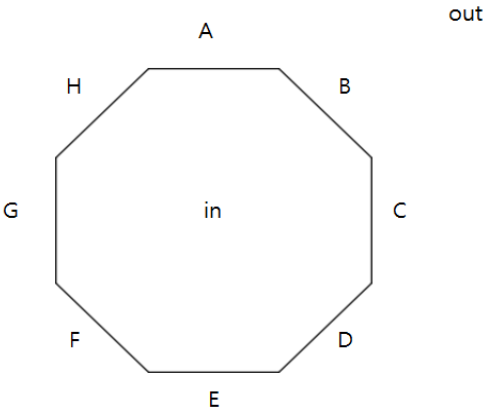


그림 2. JUHA 모델

JUHA모델은 사용자로부터 입력되는 제스처를 구분하기 위해 정다각형의 변(Side)과 영역(Region)을 이용하는데, 예를 들어[그림 2]와 같이 정팔각형을 적용했을 경우, A~H까지 총 8개의 변과 8각형의 안쪽 영역 in, 바깥영역 out으로 구성된다. JUHA 모델에서 변은 SIDE로 제스처는 G로 표시하며, 제스처 핸들러가 out 영역에서 in 영역으로 이동할 때 지나는 변을 x라하고, in 영역에서 out 영역으로 지나는 변을 y라 할 때, (x, y)를 하나의 구분된 제스처로 정의한다.

$$SIDE = \{A, B, C, D, E, F, G, H\}$$
$$G = \{(x, y) \mid x \in SIDE, y \in SIDE\}$$

이때, 제스처 핸들러는 마우스 제스처의 경우 마우스 포인트, 터치 제스처의 경우 사람의 손가락 포인트, 3D 제스처일 경우 깊이 카메라를 통해 획득한 신체 일부 등 응용에 따라 다양한 형태가 될 수 있다. [그림 3]은 JUHA 모델에서 발생할 수 있는 제스처의 예로, ⓐ의 경우, G = (C, A)로 제스처 핸들러가 out → in 으로 이동할 때 C를 지나고 in → out으로 이동할 때 A를 통과한 예이며, ⓑ의 경우, G = (B, H)로 B와 H를 순차적으로 이동하는 제스처를 표현한 것이다.

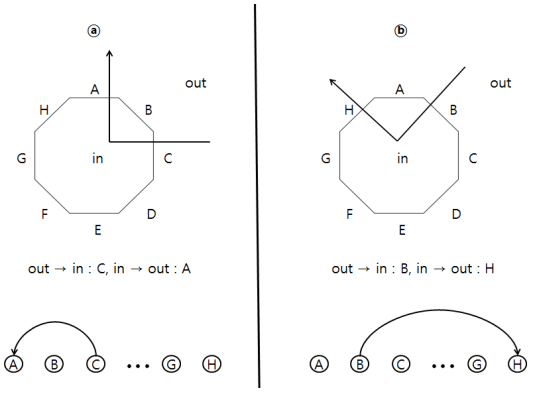
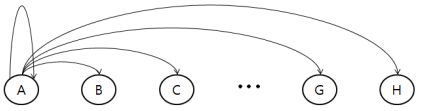


그림 3. JUHA 모델에서의 제스처 구분

정팔각형을 적용한 JUHA 모델에서 식별 가능한 제스처는 [그림 4]와 같이 (x, y)값에서 x 가 A 일 때, y는 A부터 H까지 8가지 경우가 발생할 수 있다. x도 A부터 H까지 8가지 경우가 발생할 수 있으므로 이 경우, 총 64가지 체스처를 식별할 수 있다.



$$(A, A), (A, B), (A, C), \ldots (A, G), (A, H)$$

그림 4. x = A일 때, 가능한 y

2. JUHA XML 명세

본 논문에서 제안한 JUHA 모델을 시스템이 인식가능한 형태로 표현하기 위해서 XML기반의 명세언어를 설계한다. 명세대상 항목은 각 제스처를 구성하는 한 쌍의 SIDE와 어플리케이션 제어를 위한 마우스 또는 키보드 이벤트에 대한 세부 정보가 포함된다. 마우스 이벤트의 경우 오른쪽 버튼 클릭, 왼쪽 버튼 클릭, 휠 스크롤 등 다양한 이벤트가 매핑 될 수 있으며, 키보드 이벤트의 경우 기능키(Function Key) 또는 Ctrl 키와 Alt 키 등을 조합한 형태가 매핑 될 수 있다. 마우스 이벤트의 경우, 마우스 이벤트 전달을 위한 스크린 상에서의

마우스 포인트 위치(x, y) 좌표의 명세가 추가적으로 필요하다. JUHA XML 스키마는 주 명세 요소인 JUHA complexType과 여기에 포함되는 어트리뷰트의 각 타입을 정의한 useType, keyType, eventType, controlType, SIDE 등 5개의 simpleType으로 구성되며 그 구조는 [그림 5]와 같다.

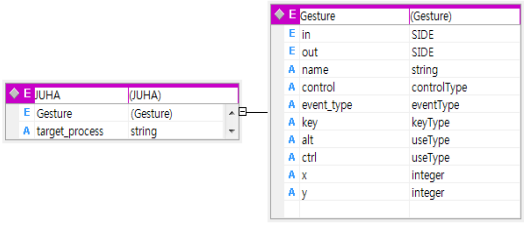


그림 5. JUHA모델의 XML명세를 위한 스키마 구조

스키마는 3개의 요소(Element)와 9개의 속성(Attribute)으로 구성되어 있으며, 여기에는 제스처를 통해 제어 대상이 되는 GUI 어플리케이션의 프로세스 이름과, JUHA 모델에서 제스처를 구성하는 한 쌍의 SIDE(in, out) 정보 등을 포함한다. 주요 명세요소와 기능은 [표 1]과 같고 전체 스키마는 [그림 6]과 같이 정의하였다.

표 1. 명세요소와 기능

요소명	설명
JUHA	Root 요소
Gesture	제스처 이벤트 매핑을 위한 요소
target_process	목표 GUI 응용 프로세스 결정
in	제스처 핸들러의 out->in 이동시 SIDE
out	제스처 핸들러의 in->out 이동시 SIDE
name	제스처 명칭
control	제스처 핸들러 결정
event_type	마우스, 키보드 이벤트 결정
key	키보드 이벤트 발생 키
alt	alt 키 사용 여부
ctrl	control 키 사용 여부
x	마우스 이벤트 발생 x 좌표
y	마우스 이벤트 발생 y 좌표

```
<xs:element name="JUHA">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="Gesture">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="in" type="SIDE" />
            <xs:element name="out" type="SIDE" />
          </xs:sequence>
          <xs:attribute name="name" type="xs:string" use="required" />
          <xs:attribute name="control" type="controlType" use="required" />
          <xs:attribute name="event_type" type="eventType" use="required" />
          <xs:attribute name="key" type="keyType" use="required" />
          <xs:attribute name="alt" type="useType" use="required" />
          <xs:attribute name="ctrl" type="useType" use="required" />
          <xs:attribute name="x" type="xs:integer" use="optional" />
          <xs:attribute name="y" type="xs:integer" use="optional" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="target_process" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>

<xs:simpleType name="useType">
  <xs:restriction base="xs:string">
    <xs:pattern value="Yes|No"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="keyType">
  <xs:restriction base="xs:string">
    <xs:pattern
value="[0-9]|[A-Z]|Left_Arrow|Right_Arrow|Up_Arrow|Down_Arrow|Plus|Minus|
                F1|F2|F3|F4|F5|F6|F7|F8|F9|F10|F11|F12"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="eventType">
  <xs:restriction base="xs:string">
    <xs:pattern value="keyboard|mouse"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="controlType">
  <xs:restriction base="xs:string">
    <xs:pattern
value="head|shoulder_center|shoulder_right|shoulder_left|elbow_right|elbow_left|
      wrist_right|wrist_left|hand_right|hand_left|spine|hip_center|hip_right|hip_left|
       knee_right|knee_left|ankle_right|ankel_left|foot_right|foot_left"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SIDE">
  <xs:restriction base="xs:string">
    <xs:pattern value="A|B|C|D|E|F|G|H"/>
  </xs:restriction>
</xs:simpleType>
```

그림 6. JUHA모델의 XML명세를 위한 전체 스키마

3. 제안모델 검증을 위한 프로토타입 설계 및 구현

JUHA 모델을 이용한 제스처인식을 테스트하기 위한 프로토타입을 구현하였다. 프로토타입은 3D 제스처와 마우스 제스처를 지원하는 2개의 어플리케이션으로 구성되어 있다. 3D 제스처의 경우, Kinect센서를 통해 추

출된 사용자의 오른손 위치를 제스처 핸들러로 이용하였다. Kinect 센서를 통해 Full-Body Tracking이 가능하며, 30fps의 성능으로 머리, 목, 가슴, 팔꿈치, 손목 등 인체의 주요 골격에 대한 (x, y, z) 3차원 위치 값을 추적(tracking)할 수 있다[16]. 마우스 제스처는 윈도우 상에서의 마우스 버튼의 down, move, up에 해당하는 이벤트 처리를 통해 이루어진다. 프로토타입 설계를 위한 클래스 다이어그램은 [그림 7]과 같다. XML 명세 파싱을 위한 JUHAParser 클래스와 GUI 이벤트 처리를 담당하는 GUIEvent 클래스는 JUHA 패키지로 구성되어 있으며, Kinect 센서를 이용한 3D 제스처의 UI를 담당하는 JUHA3DSimulator 클래스와 마우스 제스처 UI를 담당하는 JUHAMouseSimulator 클래스에서 공통적으로 사용하는 구조로 되어있다.

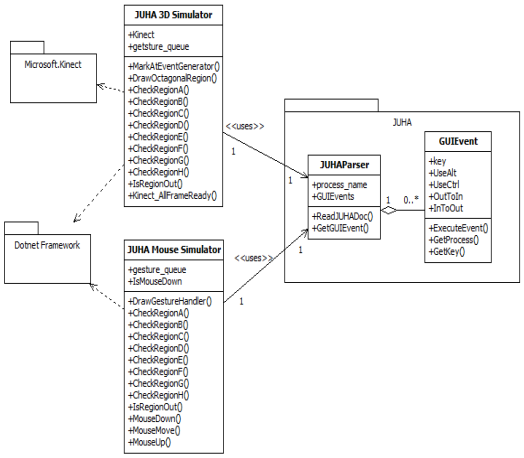


그림 7. 프로토타입 클래스 다이어그램

사용자로부터 발생한 제스처를 판별하는 알고리즘은 [그림 8]과 같다. JUHA 모델에서 제스처는 정다각형의 SIDE 쌍으로 결정되며, 제스처 핸들러의 (x, y) 좌표 값이 SIDE 영역으로 진입할 때부터 정다각형의 영역을 벗어 날 때 까지 각 SIDE 영역에 위치할 때, 큐(Queue)에 정보를 기록한다. 이때, 큐에 보관된 처음 값이 out → in 에 해당되고, 마지막 값이 in → out일 때의 side로 결정하며 중간과정의 정보는 무시한다.

```
입력 : 제스처 핸들러 좌표 X, Y
출력 : 한 쌍의 입출력 Side in, out

begin
 while(다각형 in 영역에 X, Y 가 포함)
   for (다각형 out 영역의 각 Side 영역 A~H)
     X, Y를 포함하면 해당 Side를 Queue에 추가;
   end for
 end while

 return Queue의 처음 값 in, 마지막 값 out
end
```

그림 8. 제스처 판별 알고리즘

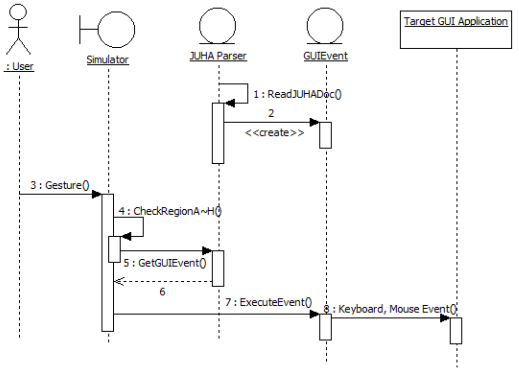


그림 9. 프로토타입 순차 다이어그램

프로토타입의 순차다이어그램은 [그림 9]와 같다. 제스처 발생으로부터 목표 GUI 어플리케이션 제어까지의 일련의 프로세스는 프로토타입을 구성하는 3개의 주요 클래스를 통해 이루어진다. 개발자에 의해 작성된 JUHA XML문서는 JUHAParser 클래스를 통해 파싱이 되고, 제스처-이벤트 쌍(pair) 정보를 갖는 GUIEvent객체를 생성하게 된다. 생성된 GUIEvent 객체들의 집합(set)은 JUHAParser 클래스의 멤버로 저장된다. JUHA3DSimulator는 Kinect 센서로부터 입력된 영상과 Octagonal 영역, 이벤트 핸들러의 합성을 통한 UI를 구성한다. Octagonal 영역의 각 SIDE 영역에서의 제스처 핸들러의 움직임을 감지하며, 사용자가 발생한 제스처가 JUHAParser의 멤버로 저장된 GUIEvent 집합에 포함되어 있으면 해당 GUIEvent 객체의 내용을 검사하여 이벤트를 발생시키고, 이 이벤트는 GUI어플리케이션에 전달되어 기능이 실행된다.

3D 제스처 프로토타입의 실행 화면은 [그림 10]과 같

이 영상 출력 부분과 결과 출력부분으로 구분된다. 영상 출력부분은 Kinect 센서를 통해 발생한 영상에 제스처 판별을 위한 정팔각형 영역을 합성하여 표시한다. 사용자의 오른손 위치를 제스처 핸들러로 활용하여 JUHA 모델을 테스트 할 수 있다.

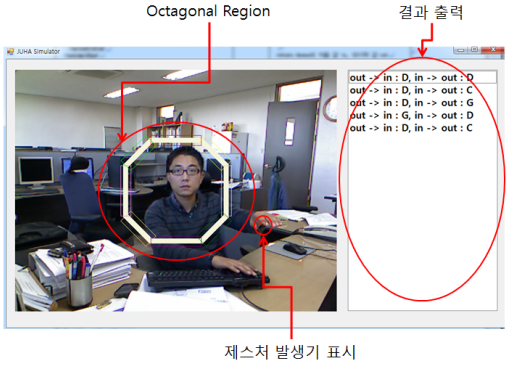


그림 10. 3D 제스처 프로토타입 실행 화면

마우스 제스처 프로토타입의 경우 [그림 11]과 같이 실행 창에 Octagonal 영역을 표시하고, 사용자의 마우스 제스처 궤적과 이에 따른 결과를 출력한다.

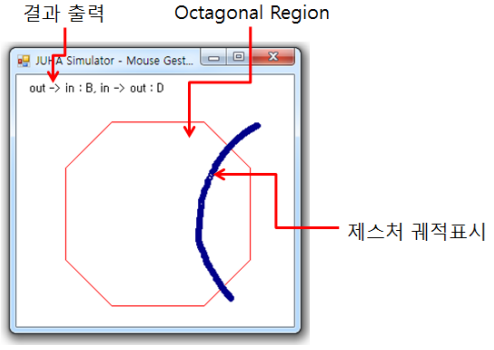


그림 11. 마우스 제스처 프로토타입 실행 화면

IV. 사례연구

1. GUI 어플리케이션 제스처 맵핑

제안한 모델을 이용하여 GUI 어플리케이션을 제어하기 위해서는 제스처 모델링이 필요하다. 현재 웹 브라우저에서 마우스 제스처를 지원하는 다양한 플러그인이 개발되어 사용되고 있다. 기존 솔루션에서 웹 브라우저를 제어하기 위한 마우스 제스처 명령[17]을 [표 2]와 같이 JUHA 모델로 맵핑하여 테스트 하였다.

표 2. 웹브라우저 제어를 위한 제스처 모델링

기능	마우스 제스처 명령	제안모델 맵핑	out → in	in → out
탐색히스토리 이전이동	←	←	C	G
탐색히스토리 이후이동	→	→	G	C
페이지 위쪽으로 스크롤	↑	↑	E	A
페이지 아래쪽으로 스크롤	↓	↓	A	E
현재탭의 왼쪽 탭 표시	↪	↪	C	C
현재탭의 오른쪽 탭 표시	↩	↩	G	G
현재 페이지의 맨 위 스크롤	⇓	⇓	E	E
현재 페이지의 맨 아래 스크롤	⇑	⇑	A	A
새 창 열기	↰	↰	E	G
새 탭 열기	↱	↱	E	C
전체화면 보기로 전환	↲	↲	A	G
현재 탭 닫기	↳	↳	A	C
페이지 시계방향 회전	↻	↻	C	A
페이지 반시계방향 회전	↺	↺	C	E
현재 페이지 크기 한단계 확대	⤴	⤴	G	A
현재 페이지 크기 한단계 축소	⤵	⤵	G	E

[표 2]의 제스처를 다음과 같이 G-WebBrower set으로 정의할 수 있다.

$$G-WebBrowser = \begin{Bmatrix} (C,G),(G,C),(E,A),\\ (A,E),(C,G),(G,G),\\ (E,E),(A,A),(E,G),\\ (E,C),(A,G),(A,C),\\ (C,A),(C,E),(G,A),\\ (G,E) \end{Bmatrix}$$

2. 실행결과

사용자로부터 발생한 제스처에 따른 웹 브라우저 기능수행은 [그림 12]와 같이 XML 문서로 명세한다. 이는 [그림 6]의 스키마를 만족하며, 프로토타입을 통해 이벤트발생으로 목표 GUI 어플리케이션을 제어할 수 있다.

```
<JUHA target_process="chrome">
  <Gesture name="Left" control="hand_right" event_type="keyboard" key="Left_Arrow"
alt="No" ctrl="Yes">
    <in>C</in>
    <out>G</out>
  </Gesture>
.
.
.
  <Gesture name="Up and Right" control="hand_right" event_type="keyboard"
key="T" alt="No" ctrl="Yes">
    <in>E</in>
    <out>C</out>
  </Gesture>
  <Gesture name="Down and Left" control="hand_right" event_type="keyboard"
key="F11" alt="No" ctrl="No">
    <in>A</in>
    <out>G</out>
  </Gesture>
  <Gesture name="Right and Up" control="hand_right" event_type="keyboard"
key="Plus" alt="No" ctrl="Yes">
    <in>G</in>
    <out>A</out>
  </Gesture>
  <Gesture name="Right and Down" control="hand_right" event_type="keyboard"
key="Minus" alt="No" ctrl="Yes">
    <in>G</in>
    <out>E</out>
  </Gesture>
</JUHA>
```

그림 12. 웹브라우저 제어를 위한 JUHA 명세

사용자로부터 입력된 제스처가 (G, A)인 경우 키보드의 ctrl 키와 + 키에 해당하는 KeyDown 메시지를 웹 브라우저로 전송하여 결과적으로 [그림 13]의 ⓐ와 같이 웹브라우저에 표시된 웹 페이지의 화면을 확대하는 결과를 보인다. 마찬가지로 (G, E)에 대하여 화면 축소, (E, C)에 대하여 새 탭(New Tab)생성 등 ⓑ, ⓒ와 같이 제스처를 통해 제어할 수 있다. 이와 같이 JUHA 모델은 하나의 XML명세로부터 3D 제스처 모델과 마우스 제스처 모델을 통해 목표 GUI 어플리케이션을 제어할 수 있음을 확인할 수 있다.

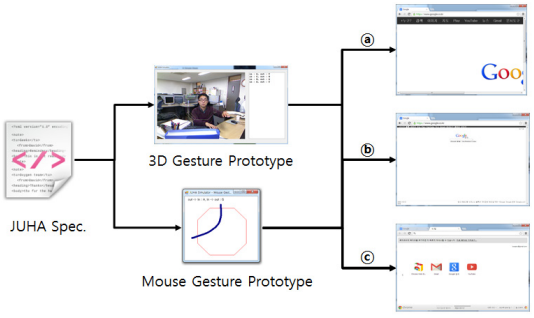


그림 13. 제안모델을 이용한 웹 브라우저 제어

3. 기존연구와의 비교

FAAST 툴킷은 사용자가 직접 3D 제스처를 정의하여 기존 어플리케이션에 적용할 수 있는 장점이 있지만 MS Kinect 센서에 종속적인 한계가 있다. 제안한 JUHA 모델은 정다각형에 종속적인 2D 기반의 제스처를 정의할 수 있으며, 3D, 마우스, 터치 제스처등 다양한 환경에 적용이 가능한 장점이 있다.

V. 결론 및 향후 연구과제

본 논문에서는 이벤트 기반의 GUI 어플리케이션을 제어할 수 있는 JUHA 모델을 제안하였다. 또한 시스템이 인식 가능한 XML 명세를 제시하였으며, 프로토타입 개발을 통해 사용자의 손 움직임 및 마우스 움직임을 통해 웹 브라우저를 제어하는 테스트를 수행하였다. 제안한 모델은 구분 가능한 방향지향적인 제스처를 표현할 수 있으며, 이를 통해 GUI어플리케이션을 제어할 수 있다. 향후, 터치제스처를 포함하여 다양한 제스처 환경에서 제안모델의 활용방안과 이에 따른 명세언어의 정제 및 확장에 따른 연구가 필요하다.

참 고 문 헌

[1] A. Valli, "Notes on Natural Interaction," http://www.citeulike.org/user/eckel/article/4324923

[2] W. Liu, "Natural User Interface-Next Mainstream Product User Interface," IEEE, CAIDCD, 2010.

[3] http://www.microsoft.com/en-us/kinectforwindows

[4] http://us.playstation.com/ps2/accessories/eyetoy-usb-camera-ps2.html

[5] http://www.microsoft.com/en-us/pixelsense

[6] http://www.immersivetouch.com

[7] http://www.dragonnaturallyspeaking.com

[8] http://www.perceptivepixel.com

[9] 송대현, 박재완, 이칠우, "제스처 인식 대형 놀이 시스템 기반 한자 학습 콘텐츠", 한국 콘텐츠학회 논문지, 제10권, 제9호, pp1-8, 2010.

[10] 옥수열, "전자주사위 및 손동작 인식을 활용한 아동용 에듀테인먼트 게임 콘텐츠 개발에 관한 연구", 한국멀티미디어학회 논문지, 제14권, 제10호, pp1348-1364, 2011.

[11] M. Abrams and C. Phanouriou, "UIML: An XML Language for Building Device-Independent User Interface," XML'99, 1999.

[12] 박기창, 서성채, 김병기, "J2ME MIDlet 사용자 인터페이스 자동생성을 위한 XML 언어", 한국정보처리학회 논문지, 제15-D권, 제3호, pp.327-336, 2008.

[13] 이동수, 박기창, 김병기, "사용자 인터페이스 명세 언어를 이용한 위피 비즈니스 프로그램 저작 도구 구현", 한국콘텐츠학회 논문지, 제9권, 제2호, pp.152-162, 2009.

[14] S. H. Khandkar and F. Maurer, "A Domain Specific Language to Define Gestures for Multi-Touch Applications," ACM, DSM'10, 2010.

[15] E. A. Suma, B. Lange, A. Rizzo, D. M. Krum, and M. Bolas, "FAAST:The Flexible Action and Articulated Skeleton Toolkit," IEEE, Virtual Reality, 2011.

[16] C. Waithayanon and C. Aporntewan, "A Motion Classifier for Microsoft Kinect," IEEE, ICCIT, 2011.

[17] 정 혁, "제스처 기반 사용자 인터페이스 표준화 동향", 한국정보기술학회지, 제9권, 제3호, pp.29-33, 2011.

저 자 소 개

박 기 창(Ki-Chang Park) 정회원

- 2001년 2월 : 전남대학교 물리학과(이학사)
- 2003년 2월 : 전남대학교 전산학과(이학석사)
- 2005년 2월 : 전남대학교 전산학과(박사수료)
- 2008년 4월 ~ 현재 : 디지털콘텐츠 협동연구센터 선임연구원

<관심분야> : 객체지향시스템, 소프트웨어 모델링

서 성 채(Seong-Chae Seo) 정회원

- 1994년 2월 : 전남대학교 전산학과(이학사)
- 1997년 2월 : 전남대학교 전산학과(이학석사)
- 2006년 2월 : 전남대학교 전산학과(이학박사)
- 2012년 8월 ~ 현재 : 전남대학교 유비쿼터스 정보가전사업단 박사후연구원

<관심분야> : 소프트웨어 모델링, 요구공학, 소프트웨어 보안, 정형기법 등

정 승 문(Seung-Moon Jeong) 정회원

- 1991년 2월 : 동신대학교 컴퓨터공학과(공학사)
- 1999년 2월 : 동신대학교 컴퓨터공학과(공학석사)
- 2004년 2월 : 동신대학교 컴퓨터공학과(공학박사)
- 2004년 ~ 현재 : 디지털콘텐츠 협동연구센터 부장
- 2006년 ~ 현재 : 동신대학교 디지털콘텐츠학과 교수

<관심분야> : 디지털 크리처, 가상현실, 인공지능 등

강 임 철(Im-Chul Kang) 정회원

- 1991년 2월 : 전남대학교 전산통계학과(이학사)
- 1997년 2월 : 전남대학교 경영학과(경영학석사)
- 2005년 2월 : 전남대학교 전자상거래협동과정(경영학박사)
- 2005년 9월 ~ 현재 : 동신대학교 디지털콘텐츠학과 교수
- 2011년 3월 ~ 현재 : 디지털콘텐츠 협동연구센터 소장

<관심분야> : 전자상거래, 디지털콘텐츠, 3D 애니메이션, 가상현실 등

김 병 기(Byung-Gi Kim) 정회원

- 1978년 2월 : 전남대학교 수학과(이학사)
- 1980년 2월 : 전남대학교 수학과(이학석사)
- 2000년 2월 : 전북대학교 수학과(이학박사)
- 1981년 2월 ~ 현재 : 전남대학교 전자컴퓨터공학부 교수

<관심분야> : 소프트웨어공학, 객체지향시스템 등