단말기 동작인식과 상호 통신을 결합한 스마트폰 게임의 구현

이숭희*

Implementation of Smartphone Games Combining Motion Recognitions and Mutual Communications of Terminals

Soong-Hee Lee*

요 약

스마트 폰의 일반화로 인해 스마트폰 게임이 많이 출시되고 있지만, 주로 터치 기능을 이용하여 화면 상의 상호작용을 통한 게임이 주류를 이루고 있다. 한편 기존의 가속도센서, 방향센서 외에 자이로센서 등 단말기의 위치와 동작을 인식하는 기능은 나날이 발전하고 있다. 따라서 사용자의 동작을 단말을 통해 인식하여 사용자 단말을 게임 도구로 사용하는 게임의 개발이 기술적으로 가능해지고 있으나, 아직 구체화된 사례는 발견되지 않고 있다. 본 논문에서는 단말기의 동작 인식과 단말기간 상호 통신을 결합한 새로운 형태의 게임의 기본 틀과 알고리즘을 제시하고, 그 구현 내용과 결과를 제시한다.

ABSTRACT

The generalization trend of smart phones have brought many smartphone games into daily lives. These games are mainly dependent on the interactions on the display of the phone using finger touches. On the other hand, functions for detecting the positions and actions of the phones such as gyro-sensors have been rapidly developed over the former orientation sensors and acceleration sensors. Though it has become technologically possible to detect the users' motion via the smartphone devices and to use the phone device directly as the game device, it is hard to find the actualized cases. This paper proposes a new paradigm including basic frameworks and algorithm for the games combining the motion recognitions and mutual communications on the smartphones and finally presents the details of its implementation and results.

키워드

스마트폰, 동작인식, 가속도센서, 방향센서, 블루투스

Key word

Smartphone, motion detection, acceleration sensor, orientation sensor, Bluetooth

* 정회원 : 인제대학교 정보통신공학과(icshlee@inje.ac.kr)

접수일자 : 2012. 04. 09
심사완료일자 : 2012. 06. 05

Open Access http://dx.doi.org/10.6109/jkiice.2012.16.9.2064

Ⅰ. 서 론

근래에 통신시장에 격변을 몰고 온 스마트폰의 주된 장점 중 하나는 인터넷에 접속하여 수많은 종류의 다양한 앱을 다운로드 받아서 사용할 수 있다는 것이다. 그 많은 앱들 중에서도 사용자들이 가장 많이 다운로드하고 있고 관심을 끌고 있는 것은 단연 게임 앱이다. 대부분의 게임 앱들은 주로 터치 기능을 통해 화면과 상호작용하는 형태로 구성되어 있다.

스마트폰이 일반화되는 추세와 함께 자이로센서 등 단말기의 위치와 동작을 세밀하게 추적이 가능한 신기술들이 등장하고 있다. 그럼에도 이러한 단말기의 위치나 동작을 인식하여 단말기 자체를 게임 도구로 사용하는 게임은 아직 별로 눈에 띄지 않는다. 스마트폰에는 기존의 게임 시장을 석권했던 닌텐도위에서 사용하는 수준과 버금가는 수준의 센서 기능과 블루투스 등의 상호통신이 가능한 기능들이 내장되어 있어서 사용자의 게임 욕구를 증대시키고 관심을 끌 수 있는 새로운 패러다임의 게임 개발이 기대된다.

단말기의 위치나 동작을 인식하여 적용한 기존의 사례들을 조사해 보면, 주로 가상 현실 기술과 접목되거나 증강현실 기술과 접목되어 적용된 사례들이며, 스마트폰의 위치 인식 등에 대한 연구가 일부 이루어진 정도이다[1]~[4].

본 논문에서는 스마트폰에 내장된 가속도센서와 방향센서를 이용하여 단말기의 위치를 좌표값으로 추정하고 그 값을 두 단말기 사이에 블루투스를 통하여 주고받음으로써 스타워즈 광선검 대결, 서부극 대결, 야구 등의 게임 개발에 적용이 가능한 새로운 패러다임의 게임 구조를 제시하고자 한다. 2장에 게임을 위한 가상공간 모델을 수식을 이용하여 제시하고, 3장에서 사용자 단말의 게임 도구 적용 방안에 대해 설명한 후, 4장에서 동작인식과 통신을 결합한 게임의 구현 결과를 제시하고 결론을 맺을 것이다.

Ⅱ. 게임을 위한 가상공간 모델

게임의 성공적인 개발을 위해서는 개발하려는 목표가 뚜렷해야 한다. 본 논문에서는 지엽적인 부분은 생략하고 핵심이 되는 요소인 동작인식과 상호통신에 대해 주로 설명할 것이다.

단말기의 동작인식을 하기 위해 우선 기준이 되는 좌표의 설정이 요구된다. 더욱이 게임에 참여한 두 사람 사이에서 공통의 기준이 되는 좌표가 3차원 공간에서 설정이 되려면, 게임을 위한 가상공간이 가정되어야 하고 게임 시작 시점에 두 단말기의 위치를 이 가상공간의 원점에 해당하는 동일 위치에 두고 게임을 시작해야 한다. 그림 1은 이 개념을 설명한 그림이다.

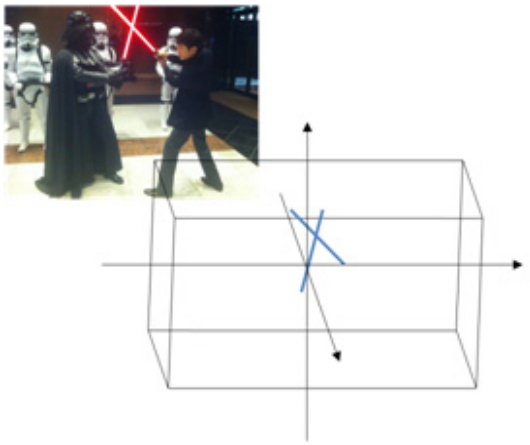


그림 1. 게임을 위한 가상공간
Fig. 1 Virtual space for the proposed game

참고 문헌 [5]에 의하면 스마트폰의 센서 중에서 방향센서를 이용하여 *x, y, z* 방향의 기울어진 정도를 알 수 있다. 또한 가속도센서로부터 *x, y, z* 각 방향의 가속도 정보도 알 수 있다.

이제 방향센서의 기울어짐 정보와 가속도센서의 가속도 정보를 사용하여 단말의 위치 변경에 따라 *x, y, z* 각 방향에서 이동한 거리를 추정하여 각 단말의 양끝단 두 점의 좌표를 알아내는 방법을 찾아보자.

단말기 중심의 초기 위치를 $(0, 0, 0)$로 하고 초기 속도가 각 방향으로 모두 0이라고 하면 시간 t 경과시 단말기 중심의 위치는 초기거리 s_0, 초기속도 v_0, 가속도 a라고 할 때 $s_0 + v_0 t + \frac{1}{2}at^2$ 으로 주어지므로, $t = 0$ 일 때, 원점으로부터의 거리 $x_0 = 0$, 속도 $u_0 = 0$, $t = \Delta t$ 경과시, 원점으로부터의 거리 $x_1 = \frac{1}{2}a_1(\Delta t)^2$, 속도 $u_1 = a_1 \Delta t$, $t = 2\Delta t$ 경과 시, 원점으로부터의 거리

$x_2 = x_1 + u_1\Delta t + \frac{1}{2}a_2(\Delta t)^2$, 속도는 $u_2 = u_1 + a_2\Delta t$, $t = 3\Delta t$ 경과 시, 원점으로부터의 거리 $x_3 = x_2 + u_2\Delta t + \frac{1}{2}a_3(\Delta t)^2$, 속도는 $u_3 = u_2 + a_3\Delta t$ 등으로 주어진다.

따라서 시간 $k\Delta t$ 경과 시 x 축 방향의 가속도가 a_k, 이전 시점의 속도가 u_{k-1}, 이전 시점의 거리가 x_{k-1}일 때 x 축에서 원점으로부터의 거리는

$$x_k = x_{k-1} + u_{k-1}\Delta t + \frac{1}{2}a_k(\Delta t)^2 \quad (1)$$

이다.

마찬가지로 시간 $k\Delta t$ 경과 시 y 축 방향의 가속도가 b_k, z 축 방향의 가속도가 c_k이고. 이전 시점의 속도가 v_{k-1}, w_{k-1}, 이전 시점의 거리가 y_{k-1}, z_{k-1}일 때 y, z축에서 원점으로부터의 거리는

$$y_k = y_{k-1} + v_{k-1}\Delta t + \frac{1}{2}b_k(\Delta t)^2 \quad (2)$$

$$z_k = z_{k-1} + w_{k-1}\Delta t + \frac{1}{2}c_k(\Delta t)^2 \quad (3)$$

로 주어져 시간 $k\Delta t$ 경과 후 단말기 중심의 위치 좌표는 (x_k, y_k, z_k) 로 나타난다.

단말기 중심의 위치 좌표는 알아냈지만 단말기의 중심이 게임 시작시의 원점으로부터 얼마나 떨어져 있는지만 파악이 된 상태이므로, 방향센서로부터 얻은 정보를 이용하여 두 단말의 양끝단의 좌표를 찾아야 한다.

단말기의 길이가 l 이면, 단말기 중심에서 단말기 끝단까지 거리는 $\frac{l}{2}$ 이며, 방향센서로부터 얻어진 pitch 각도가 θ, azimuth 각도가 ϕ 이면, 중심의 좌표가 (0, 0, 0) 일 때 단말기 끝부분의 좌표는 $(\pm\frac{l}{2}sin(\alpha+\theta).\cos\phi,$ $\pm\frac{l}{2}sin(\alpha+\theta).\sin\phi,$ $\pm\frac{l}{2}cos(\alpha+\theta))$ 로 주어진다. 방향센서값과 좌표계의 각도가 90^o 차이가 나므로 $\alpha = 90^o$를 더해줘야 한다. 단말기로부터의 roll 정보는 게임의 편의를 위해 고려하지 않기로 한다.

중심의 좌표가 (1)~(3)식에 의해 주어지면, 단말기의 양 끝의 좌표값은

$$x = x_k \pm \frac{l}{2}sin(\alpha+\theta).\cos\phi \quad (4)$$

$$y = y_k \pm \frac{l}{2}sin(\alpha+\theta).\sin\phi \quad (5)$$

$$z = z_k \pm \frac{l}{2}cos(\alpha+\theta) \quad (6)$$

로 되어 이제 가상공간 내에서의 단말기의 양 끝 단의 좌표도 구할 수 있게 되었다.

이제 광선검 게임 등에서 중심이 되는 경우인, 두 단말기의 길이 방향으로 연장시킨 두 직선이 그림 2와 같이 가상공간 내에서 교차되는 경우에 대해 살펴보자.

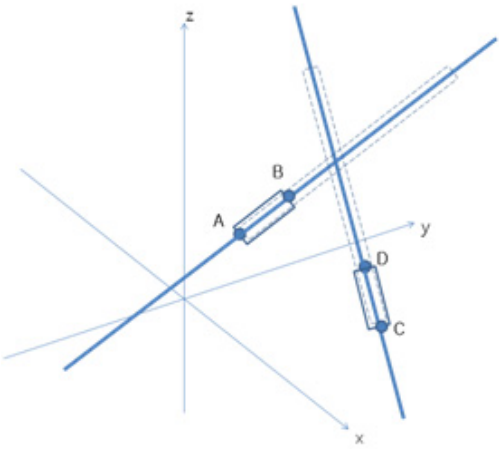


그림 2. 가상공간 상에서 단말기 길이 방향으로 연장시킨 두 직선의 교차
Fig. 2 Intersection of two straight lines enlarged along the longitudinal axes of terminal devices in virtual space

그림 2에 나타낸 게임에 사용되는 두 단말의 양끝점 A, B와 C, D는 (4)~(6)식에서 주어진 좌표값을 가진다. 이 때 점 $A(x_1, y_1, z_1)$과 $B(x_2, y_2, z_2)$ 를 지나는 직선의 식은 아래와 같다.

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1} = t \quad (7)$$

점 $C(x_3, y_3, z_3)$과 $D(x_4, y_4, z_4)$를 지나는 또 다른 직선의 방정식은 아래와 같다.

$$\frac{x-x_3}{x_4-x_3}=\frac{y-y_3}{y_4-y_3}=\frac{z-z_3}{z_4-z_3} \tag{8}$$

식 (7)로부터 다음의 관계가 성립된다.

$$\begin{aligned} x &= (x_2-x_1)t+x_1 \\ y &= (y_2-y_1)t+y_1 \\ z &= (z_2-z_1)t+z_1 \end{aligned} \tag{9}$$

두 직선이 교차되는 점에서 위 (9)식의 각각의 x, y, z 값은 (8)식에서도 같은 값을 가지므로 대입하면

$$\begin{aligned} &\frac{(x_2-x_1)t+x_1-x_3}{x_4-x_3}=\frac{(y_2-y_1)t+y_1-y_3}{y_4-y_3} \\ &=\frac{(z_2-z_1)t+z_1-z_3}{z_4-z_3} \end{aligned} \tag{10}$$

으로 주어지며, 이 식에서 다음의 관계가 성립한다.

$$\begin{aligned} &(x_2y_4-x_2y_3-x_1y_4+x_1y_3-x_4y_2+x_3y_2+x_4y_1-x_3y_1)t \\ &=x_4y_1-x_3y_1-x_4y_3-x_1y_4+x_1y_3+x_3y_4 \\ &(y_2z_4-y_2z_3-y_1z_4+y_1z_3-y_4z_2+y_3z_2+y_4z_1-y_3z_1)t \\ &=y_4z_1-y_3z_1-y_4z_3-y_1z_4+y_1z_3+y_3z_4 \\ &(z_2x_4-z_2x_3-z_1x_4+z_1x_3-z_4x_2+z_3x_2+z_4x_1-z_3x_1)t \\ &=z_4x_1-z_3x_1-z_4x_3-z_1x_4+z_1x_3+z_3x_4 \end{aligned} \tag{11}$$

위 (11)식에서 매개변수 t의 값은 두 직선의 좌표값들을 적용하여 계산이 가능함을 알 수 있다. 이 t의 세 값이 일치하면 두 직선이 교차한다는 의미이며 이 때 교차된 점의 좌표는 (9)식에서 구할 수 있다. 이렇게 구한 교차점의 좌표가 점 A, B, C, D로부터 단말의 가상 길이(예로써 광선검 게임의 경우 광선검의 길이)의 범위를 벗어나는 경우에는 충돌 상황에서 제외시킨다. (11)식의 계산에서 어느 한 식의 분모가 0이 되어 t의 값을 구할 수 없으면 다른 식에서 t의 값을 구하여 서로 같은지 비교하거나 일치하는 좌표값들을 고려하여 교차 여부를 판단하고 교차점의 좌표값을 구한다.

위의 전체 과정을 알고리즘으로 나타내면 다음과 같다.

```
init_loc(x,y,z)=(0,0,0)
init_vel(x,y,z)=(0,0,0)
prev_loc(x,y,z)=init_loc(x,y,z)
prev_vel(x,y,z)=(0,0,0)
tik=system_cycle_time
leng=(DEVICE_SIZE)/2
rec=90 degree

repeat until game_end
{
  input cur_acc(x,y,z), azim, pitch

cur_loc(x,y,z)=prev_loc(x,y,z)+prev_vel(x,y,z)*tik+((cur_acc(x,y,z))*(cu
r_acc(x,y,z))*tik*tik)/2
  cur_vel(x,y,z)=prev_vel(x,y,z)+cur_acc(x,y,z)*tik
  A_head(x)=cur_loc(x)+leng*sine(pitch+rec)*cosine(azim)
  A_tail(x)=cur_loc(x)-leng*sine(pitch+rec)*cosine(azim)
  A_head(y)=cur_loc(y)-leng*sine(pitch+rec)*sine(azim)
  A_tail(y)=cur_loc(y)+leng*sine(pitch+rec)*sine(azim)
  A_head(z)=cur_loc(x)+leng*cosine(pitch+rec)
  A_tail(z)=cur_loc(x)-leng*cosine(pitch+rec)
  if this is master
  {
      receive B_head(x,y,z), B_tail(x,y,z)

num=B_tail(x)*A_head(y)-B_head(x)*A_head(y)-B_tail(x)*B_head(y)-A
_head(x)*B_tail(y)+B_head(x)*B_tail(y)

den=A_tail(x)*B_tail(y)-A_tail(x)*B_head(y)-A_head(x)*B_tail(y)+A_h
ead(x)*B_head(y)-B_tail(x)*A_tail(y)+B_head(x)A_tail(y)
          +B_tail(x)*A_head(y)-B_head(x)*A_head(y)
      if den is not 0
      {
            temp1=num/den
            status1=ok
      }
      else
            status1=nok

num=B_tail(y)*A_head(z)-B_head(y)*A_head(z)-B_tail(y)*B_head(z)-A
_head(y)*B_tail(z)+B_head(y)*B_tail(z)

den=A_tail(y)*B_tail(z)-A_tail(y)*B_head(z)-A_head(y)*B_tail(z)+A_hea
d(y)*B_head(z)-B_tail(y)*A_tail(z)+B_head(y)A_tail(z)
          +B_tail(y)*A_head(z)-B_head(y)*A_head(z)
      if den is not 0
      {
            temp2=num/den
            status2=ok
      }
      else
            status2=nok
```

```
num=B_tail(z)*A_head(x)-B_head(z)*A_head(x)-B_tail(z)*B_head(x)-A
_head(z)*B_tail(x)+B_head(z)*B_tail(x)

den=A_tail(z)*B_tail(x)-A_tail(z)*B_head(x)-A_head(z)*B_tail(x)+A_he
ad(z)*B_head(x)-B_tail(z)*A_tail(x)+B_head(z)A_tail(x)
        +B_tail(z)*A_head(x)-B_head(z)*A_head(x)
    if den is not 0
    {
        temp3=num/den
        status3=ok
    }
    else
        status3=nok
    if status1=status2=status3=ok
    {
        if temp1=temp2=temp3
        {
            cross_stat=YES

cross(x,y,z)=(A_tail(x,y,z)-A_head(x,y,z))*temp1+A_head(x,y,z)
        }
        else
            cross_stat=NO
    }
    else if status1=status2=ok and status3=nok
        if temp1=temp2
        {
            cross_stat=YES

cross(x,y,z)=(A_tail(x,y,z)-A_head(x,y,z))*temp1+A_head(x,y,z)
        }
        else
            cross_stat=NO
    }
    else if status2=status3=ok and status1=nok
        if temp2=temp3
        {
            cross_stat=YES

cross(x,y,z)=(A_tail(x,y,z)-A_head(x,y,z))*temp2+A_head(x,y,z)
        }
        else
            cross_stat=NO
    }
    else if status3=status1=ok and status2=nok
        if temp3=temp1
        {
            cross_stat=YES

cross(x,y,z)=(A_tail(x,y,z)-A_head(x,y,z))*temp3+A_head(x,y,z)
        }
        else
            cross_stat=NO
    }
    else if status1=ok and status2=status3=nok
    {
        if A_head(x,y)=B_head(x,y) and A_head(z) != B_head(z) or
          A_tail(x,y)=B_tail(x,y) and A_tail(z) != B_tail(z) or
          A_head(x,y)=B_tail(x,y) and A_head(z) != B_tail(z) or
          A_tail(x,y)=B_head(x,y) and A_tail(z) != B_head(z) or
          A_head(y,z)=B_head(y,z) and A_head(x) != B_head(x) or
          A_tail(y,z)=B_tail(y,z) and A_tail(x) != B_tail(x) or
          A_head(y,z)=B_tail(y,z) and A_head(x) != B_tail(x) or
          A_tail(y,z)=B_head(y,z) and A_tail(x) != B_head(x) or
          A_head(z,x)=B_head(z,x) and A_head(y) != B_head(y) or
          A_tail(z,x)=B_tail(z,x) and A_tail(y) != B_tail(y) or
          A_head(z,x)=B_tail(z,x) and A_head(y) != B_tail(y) or
          A_tail(z,x)=B_head(z,x) and A_tail(y) != B_head(y)
        {
            cross_stat=YES

cross(x,y,z)=(A_tail(x,y,z)-A_head(x,y,z))*temp3+A_head(x,y,z)
        }
        else
            cross_stat=NO
    }
    else
        cross_stat=NO
    if cross_stat=YES and cross(x,y,z)<MAX(x,y,z)
    {
        send cross_stat(x,y,z)
        turn on (vibration motor, effective sound, lightening spark)
    }
  }
  else
  {
    send A_head(x,y,z), A_tail(x,y,z)
    wait cross_stat
    if cross_stat received
    if cross_stat=YES and cross(x,y,z)<MAX(x,y,z)
    {
        send cross_stat(x,y,z)
        turn on (vibration motor, effective sound, lightening spark)
    }
  }
  prev_vel(x,y,z)=cur_vel(x,y,z)
  prev_loc(x,y,z)=cur_loc(x,y,z)
}
```

Ⅲ. 사용자 단말의 게임도구 적용

스마트폰에서 사용자 단말은 센서나 통신 기능 등 다양한 기능들을 갖추고 있음에도 불구하고 현재까지의 게임 어플리케이션에서는 주로 화면 내에서의 현상에만 초점을 맞추고 있다. 논 본문에서 제안하는 새로운 현태의 게임에서는 스마트폰 단말기의 센서 기능들을 적절히 적용하여 스마트폰를 매개체로 하여 게임에서 사람 사이에 실제로 상호 작용이 일어나는 것처럼 느낄 수 있게 구성이 가능하다. 하나의 적용예로서 이미 언급한 광선검 게임 외에도 야구, 탁구, 테니스 등의 다양한 스포츠 게임에 적용이 가능하다. 이제 본 논문에서 제안하는 방식을 적용한 각 게임의 동작 형태에 대해 제시한다.

먼저 야구 게임에 적용하는 경우를 보면, 그림 3에 나타낸 바와 같이 투수 역할을 하는 사용자의 단말기의 가속도가 줄어드는 시점에 단말기 방향으로 연장된 직선을 투수가 던진 공의 궤적으로 볼 때 타자 역할을 하는 사용자의 단말의 배트 부분에 해당하는 단말기의 연장선과 교차되는지 여부와 교차하는 각도로부터 타구 결과가 파울, 홈런, 안타, 아웃 중에 하나로 선택되게 할 수 있다. 이는 광선검 게임을 구현하는 기술에서 약간의 변경으로 구현이 가능할 것으로예상된다. 또한 탁구나 테니스 등의 스포츠 게임에도 유사한 형태로 적용할 수 있다.

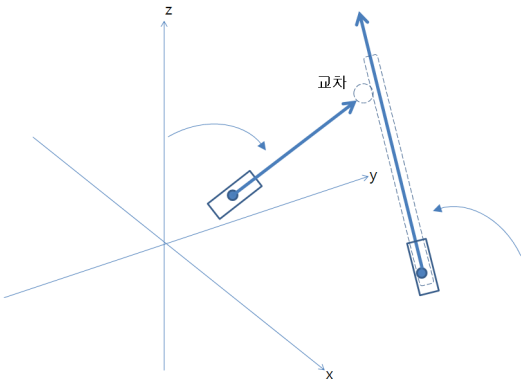


그림 3. 야구게임에서 단말기 길이 방향으로 연장시킨 두 직선의 교차

Fig. 3 Intersection of two straight lines enlarged along the longitudinal axes of terminal devices in the baseball game

다른 형태로 적용이 가능한 게임으로 서부극에서 등장하는 대결을 모사한 서부 대결 게임도 그림 4와 같은 형태로 적용이 가능하다. 각 사용자의 단말기 방향으로 연장된 직선을 그 사용자가 쏜 총알의 궤적으로 볼 때 이 직선이 상대 사용자의 동체 부분(상대 사용자의 좌표값을 중심으로 인체의 크기 만큼의 영역을 가정)과 교차되는지 여부를 판단하여 상대방을 맞추었는지를 판단하게 할 수 있다. 전술한 광선검 게임이나 야구 게임과 유사한 형태이므로, 서부 대결 게임도 제안하는 게임 방식과 알고리즘을 적용하고 프로그램의 일부를 추가 수정하는 것만으로 구현이 가능할 것으로 예상된다.

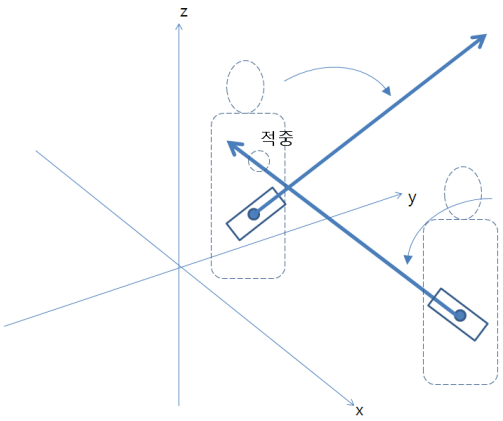


그림 4. 서부 대결 게임에서 단말기 길이 방향으로 연장시킨 직선과 상대 사용자의 교차

Fig. 4 Intersection of straight lines enlarged along the longitudinal axes of terminal devices and the counter users in the western gunfight game

Ⅳ. 동작인식과 통신을 결합한 게임의 구현

이미 전술한 바와 같이 본 논문에서 제시하는 알고리즘을 적용하여 다양한 게임의 구현이 가능하다. 이렇게 하기 위해서는 스마트폰 단말기에 내장된 센서로부터의 정보 수집과 단말 사이의 동기화를 위한 통신이 필요하다.

먼저 일반적인 스마트폰에서의 가속도 센서와 방향 센서에 대해 알아본다. 가속도 센서로부터는 단말기의 동작에 대해 매 측정 시점에 얻어진 가속도값을 프로그램으로 받아올 수 있다. 또한 방향 센서로부터 그림 5에서와 같은 세 방향에 대한 각도값을 받아올 수 있다. Z축을 중심으로 하는 0o에서 360o 까지 범위의 azimuth 값과, X축을 중심으로 하는 -180o에서 +180o까지 범위의 pitch 값, Y축을 중심으로 하는 -90o에서 +90o까지 범위의 roll 값이 주어진다[5]. 이 각도값과 가속도센서로부터의 가속도값을 적용하여 식 (1)~(11)의 과정을 거쳐 각 시점에서의 단말기 및 단말기 양끝단의 위치 좌표를 알아낼 수 있다.

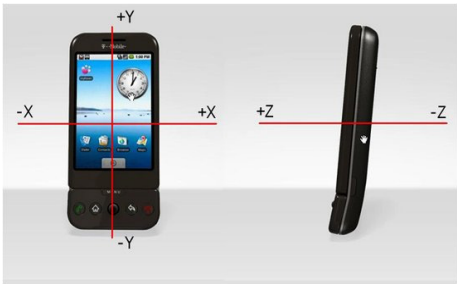


그림 5. 스마트폰 단말기와 세 방향축
Fig. 5 A smartphone with three axises

한편 두 단말기 사이에 동기가 유지되어야 두 단말기의 연장선의 교차 정보를 거의 동시에 공유하여 사용자가 게임을 할 때 실제적인 느낌을 전달할 수 있으므로 단말기 사이의 통신 연결 유지가 매우 중요해진다. 이를 위해 대부분의 스마트폰이 보유하고 있는 통신 기능인 와이파이나 블루투스 통신 기능을 적용할 수 있다. 두 단말이 통신하는 경우에는 주단말기(master)와 부단말기(slave)의 구분이 필요한데 블루투스 통신에 의해 결정되는 주,부단말기 결정을 따르거나 전화번호, MAC 주소 등을 주고받아 결정할 수 있다. 그림 6, 7에 주,부단말기의 동작 흐름을 나타내었다. 센서 데이터를 읽고 송수신하는 기능 외의 교차점 계산 등의 처리는 주단말기에서 이루어지도록 구성하였다.

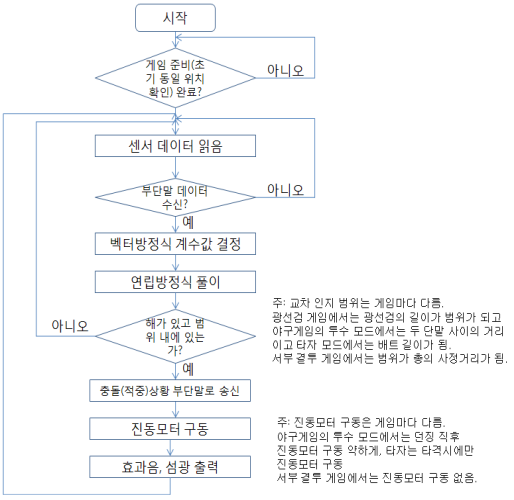


그림 6. 주단말기의 동작 흐름
Fig. 6 Information flow of the master terminal device

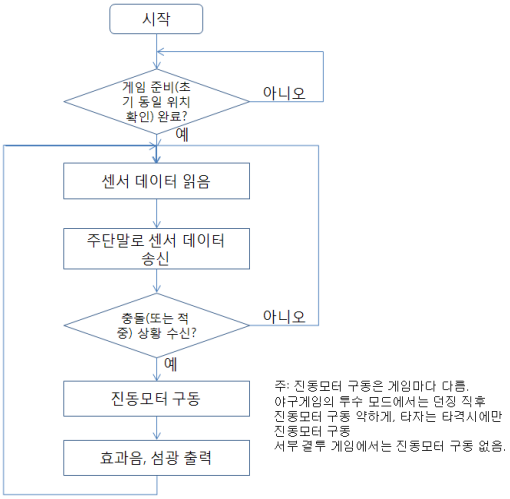


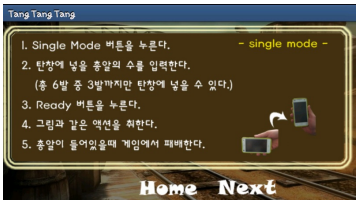
그림 7. 부단말기의 동작 흐름
Fig. 7 Information flow of the slave terminal device

제안된 게임 방식과 알고리즘을 적용하여 구현한 결과는 아직 완성되지 못 하였지만 대신에 제안된 방식과 알고리즘을 일부 적용한 결과를 소개하고자 한다. 본 논문에서 제안하고 있는 단말기의 위치 및 방향 파악 알고리즘은 적용되지 않았지만 두 단말기 사이에 블루투스 통신을 통하여 동기화하고 각 단말기의 동작을 가속도 센서로 감지하여 서부영화에서 보는 대결과 유사한 형태로 구성하였다. 그림 8은 이 구현된 스마트폰 게임을 시연한 화면이다.

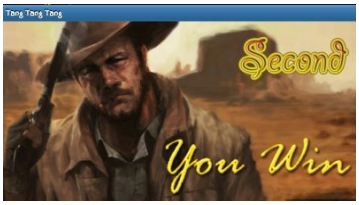
상기의 구현 결과 기본적인 게임 기능은 잘 동작이 되었고 새로운 형태의 게임으로 느껴졌지만, 본 논문에서 제안하는 단말기의 연장선 교차 감지 기능이 빠져있어서 현실감이 부족한 문제가 있었다. 비록 본 논문에서 제안하는 바를 모두 적용하지는 못 했지만 이 구현 결과를 통해 제안하는 새로운 게임 패러다임의 가능성은 확인할 수 있었다.

제안하는 방식과 알고리즘을 적용한 게임의 구현에는 많은 튜닝 과정이 소요되어 아직 구현이 완료되지 않았으며 이 과정을 줄이는 방법과 자이로센서 등의 새로운 센서 정보를 적용하는 방법 등에 대해서는 향후 과제로 남겨둔다.

(a)



(b)



(c)

그림 8. 구현된 게임 시연 화면
(a) 시작 화면 (b) 설명 화면 (c) 승리한 화면
Fig. 8 Demonstrated images of the implementated game
(a) Start image (b) Help image (c) Resultt image

V. 결 론

본 논문에서 제안하는 새로운 스마트폰 게임 방식에 대한 기본적인 알고리즘과 동작 원리는 모두 구성이 되었으나, 실제 구현에 있어서는 센서값의 민감도 정도와 정확성, 통신 속도와의 정합성 요인 등에 의해 시행착오를 거쳐야 하는 등 튜닝 작업에 많은 시간이 필요한 것으로 사료된다. 또한 자이로센서 등의 더 정교한 센서로부터 얻은 정보를 적용하여 본 논문에서 제안한 방법을 더 개선하여 더 실감나는 게임의 구현이 가능할 것으로 보이며 추후 연구 과제로 남겨두고자 한다.

참고문헌

[1] 김은길, 염미령, 김종훈, “방향 센서를 활용한 좌표 및 면적 측정 안드로이드 애플리케이션 개발,”한국정보교육학회지, 제15권, 제3호, pp.439-447, 2011.
[2] 용호중, 백종원, 장태정, “위치/방향 센서 및 진동촉각 장치를 이용한 입체 영상 기반 가상 현실 게임,” 한국HCI학회 Proceedings, 1부, pp.88-93, 2006.
[3] 윤종현, 박종승, “사용자 손 동작 추적에 기반한 증강현실 게임 인터페이스,” 한국게임학회 논문지, 제6권, 제2호, pp.3-12, 2006.
[4] 김부년, 김종호, 김태영, “가상 현실 게임 환경에서의 가상 사용자 손 인식 방법,” 한국게임학회 논문지, 제10권, 제2호, pp.49-56, 2010.
[5] http://andre-world.tistory.com/4

저자소개

이숭희(Soong-Hee Lee)

‘95 경북대 전자공학과 공학박사
‘97~현재 인제대학교
정보통신공학과 부교수

※관심분야 : 통신네트워크, 차세대통신기술, 미래인터넷