PC 기반의 항공기 시뮬레이터 패널부분 개발*

Development of Panel Part in Flight Simulator based on PC

이준우**, 채상원***, 이칠기****
Jun-Woo Lee, Sang-Won Chae and Chil-Gee Lee

Abstract

The flight simulator should be made like a actual flight. For the scene of sight, instrument should show the condition of flight and the pilot should catch the altitude, speed, pose and rate of lift of the airplane.

This paper describes visual training program of driving airplane in practice. It is for beginners using joystick in PC, implements airplane physical equations. Flight simulator in it implements airplane panel parts in order to make simple modeling. And it uses rendering technology to implement vision parts of panel. It uses double buffering to make it faster and more efficient.

Key Words: Flight Simulator, Program, Panel, Double Buffering, Rendering

* 본 연구는 한국과학재단 목적기초연구(R01-2000-00250) 지원으로 수행되었음.

** 현대 모비스
*** 성균관대학교 전기전자컴퓨터공학부 석사2기
**** 성균관대학교 전기전자컴퓨터공학부 부교수

1. 서 론

시뮬레이터에는 많은 종류의 형태가 있다. 시뮬레이터는 원래의 시스템과 비슷하게 동작하도록 제작되어 사용자에게 실제 시스템을 이용하는 것과 같은 효과를 주는 것으로, 연구용, 훈련용, 오락용 등 그 사용 목적이 다양하다. 그 중 비행 시뮬레이터는 컴퓨터가 장족의 발전하기 전에도 부분적인 시뮬레이터가 사용되었다. 하지만 지금은 시뮬레이터 탑승 시간의 일부를 항공 법규상 비행시간으로 인정하는 수준까지 시뮬레이터가 발전하였다. 시뮬레이터를 이용하여 훈련할 때 실제 항공기만 가지고 훈련할 때보다 상당한 예산 절감이 된다는 것을 볼 수 있다. 이는 비행의 무경험자를 실제 비행과 비슷한 상황에서 비행을 경험하고 비행 훈련을 하면 그만큼 훈련시간이 줄어든다는 것을 나타낸다[4]. 아울러 (악천후, 계기고장 등) 비정상 상태에 대한 경험은 그보다도 더욱 더 효과적 가치를 갖고 있다. 이런 수준의 시뮬레이터를 구현하는 데 있어서 비행 시뮬레이션은 과거 10년 동안 복잡한 계산의 증가에 따라 시뮬레이션 호스트 컴퓨터의 계산량도 점점 증가하게 됐다. 이로 인해 컴퓨터의 실시간 처리를 위해 여러 개의 프로세서를 사용하여 각기 다른 일을 처리하게 하는 방법을 사용하였다. 이 때문에 시스템의 가격 상승을 불러왔고 재정적으로 어려운 곳에서는 프로세서 하나 또는 두 개 정도를 사용하여야 했다. 그렇기 때문에 실시간 처리가 불가능하였다. 한두 개의 적은 프로세서나 컴퓨터에서 실시간 처리를 위해서는 최대한 연산처리부분을 줄이는 방법을 사용하여야 하는데 모델을 최소화 시켜도 프로세서 자체 큰 부담을 가지게 된다. 이를 위해서 모델의 최소화와 비선형 부분은 항공기 제원으로부터 계산된 Look-Up table의 Parameter를 사용하여 처리함으로 계산량을 줄이는 방법을 사용하였다. 또한 실시간 처리를 가능하도록 하여 조종간의 조작으로부터 신호를 테이블에서 검색하여 그에 맞는 데이터를 모션 방정식에 의해 계산하여 비행 자세, 위치 등을 계산해 주게 된다. 이를 위해 실시간 시뮬레이션에 대한 기본적인 개념과 그를 만족시킬 수 있는 비행 시뮬레이터의 기본적인 수학적 모델에 비행 역학적인 data와 조종간의 탄성 방정식을 이용하여 최종적인 비행 자세와 위치에 대한 데이터를 얻게 된다.

현재 진행되고 있는 비행 시뮬레이터도 같은 맥락으로 진행되고 있는데 그 중에서도 항공기 계기판 구현에 초점을 두고 진행하고 있다. 항공기 시뮬레이터에 있어서 중요한 부분은 항공기 외부를 표현하는 Vision Part와 항공 역학 방정식을 통해 계기판의 값을 표현하는 Panel Part로 구분을 지을 수 있다. 본 논문에서는 항공 역학 방정식의 구현을 통해 최대한 초심자들에게 있어 조이스틱으로 운전하는 것을 각각의 계기판과 비행 모습의 시각화를 통해 실제 있을 비행 운전에 있어 미리 연습할 수 있도록 하는 것이 목적이다.

2. 항공기 시뮬레이터의 기존 연구

최초의 비행 시뮬레이터는 1910년 Sanders Teacher에 의해 고안된 Universal Joint 상에 실물 항공기를 설치한 시뮬레이터이며, 1918년에는 조종간과 러더 페달의 조작으로 항공기의 자세 변화가 전기식 모터로 구동되어 나타나게 되는 비행 시뮬레이터가 개발되었다. 1920년대에 이르러 항공기의 운동을 수학적 모델로 구성하기 위한 체계적인 이론이 정립되기 시작하였으며, 기술적으로는 Lender와 Heidelberg에 의해 공압으로 3축 작동과 조종면의 지시가 가능한 장치가 개발됨으로써 비행 시뮬레이터 기술은 진일보하게 되었다. 1931년에 Johnson이 조종 장치와 직접 연결하여 속도계, 경사계, 회전계를 작동시키는데 성공하였으며, Lender는 계기와 조종간을 유압장치와 캠으로 연결하여 입력(Control)에 대한 출력(Instrument)이 비선형 계산에 의해 나타내어지도록 하였다. 미국의 Link사는 공압 계산기를 사용한 Link Trainer를 개발하여 상업화에 성공하였으며,

Telecommunication Research Establishment 사에서는 전기식 아날로그 컴퓨터를 사용하여 엔진, 계기, 유압시스템 등과 같은 기본적인 비행 동작의 모의가 가능한 시뮬레이터를 개발하였다. 그러나 이 시기의 비행 시뮬레이터들은 항공기의 공력 운동을 선형화 시킨 미분방정식을 아날로그 컴퓨터로 적분하여 실제 계기에 나타내었기 때문에 비행 조건에 따른 운동변화를 제대로 재현하지 못하여 비행훈련의 극히 일부분만 대신할 정도였다[3][5].

1943년 ENIAC Computer가 개발된 이후 디지털 컴퓨터의 고속, 고성능화가 이루어지고 아날로그-디지털 변환 기술이 발달함에 따라 항공기의 공력운동을 실시간 처리하여 실제 상황과 거의 유사하게 모의할 수 있는 시뮬레이터들이 출현하게 되었다.

항공기의 운동 상황을 재현하는 요동장치에 관한 연구가 진행되어 1958년 Rediffusion사에서 피치 운동이 가능한 Comet Ⅳ Simulator를 개발하였고, 뒤이어 6자유도 운동을 할 수 있는 요동장치가 등장함에 따라 거의 모든 비행 운동의 재현이 가능하게 되었다. 1962년에는 Rediffusion사에서 활주로 등 특정 공항의 모형을 비행 자세에 맞추어 시뮬레이터의 창에 영사할 수 있는 천연색 영상장치가 개발되어 모의 비행훈련의 실감도가 크게 향상되었으며, 1967년에는 컴퓨터로 제작된 영상을 CRT나 영사막에 투영하는 CGI(Computer Generated Image) 시스템이 General Electric사에 의해 개발되었다[5].

3. Panel 모델링 구현

3.1 항공기 시뮬레이터 설계

3.1.1 항공기 시뮬레이터 구성 및 용도

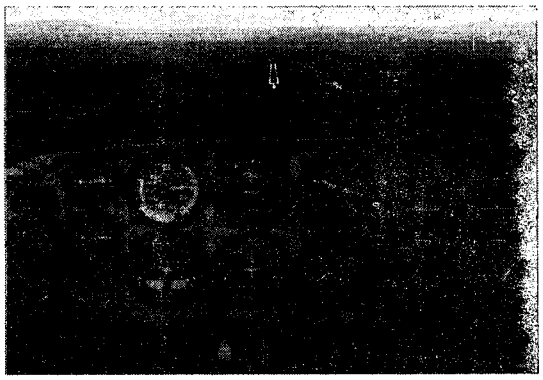
항공기는 이륙에서부터 착륙을 하지만 비행 시뮬레이터의 주목적이 초심자들을 위한 적응이므로 비행중인 상태에서 조이스틱을 조작하도록 비행 시뮬레이션 시나리오를 이용하여 구현하였다. 시뮬레이터에 사용된 조이스틱은 아날로그 방식보다 안정성이 우수한 디지털 방식의 조이스틱이며, 이 조이스틱은 PC의 USB 포트에 연결된다.

3.1.2 개발 환경

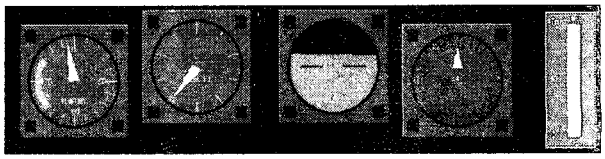
CPU	: Pentium Ⅱ 500MHz
RAM	: 32M byte
Graphic Card	: GeForce 2 MX
개발 Tool	: Visual Studio 6.0 Ent.
Library	: OpenGL

3.2 Panel 모델링

구현된 시뮬레이터의 모델은 <그림 1>에서 보이는 Cessna 182s의 Panel이다. 그러나 이러한 모든 계기판을 구현하기 위해서는 많은 필요한 데이터가 있어야 하는데 이러한 데이터를 얻을 수가 없는 관계로 <그림 2>에서 보는 것과 같이 가장 기본적인 계기판인 속도계, 고도계, 자세계, 방향계, 쓰로틀 만을 구현하게 되었다.



<그림 1> Cessna 182s의 Panel 모델



<그림 2> 비행 시뮬레이터의 Panel 모델

3.3. Panel Part 프로그램 구성

3.3.1 Pitch, Roll, Yaw Moment의 알고리즘

Pitch, Roll, Yaw Moment에 대해 비행기의 움직임을 구현을 하였다.

<그림 3> Roll, Pitch, Yaw 에 의한 움직임

1) Roll Moment는 앞에 있는 양 날개의 에일러론에 의해 발생된다. 보통 비행기의 기수를 틀 때 안정성을 위해 비행기를 양옆으로 기울여 준다. 보통 양쪽의 에일러론은 서로 반대방향으로 움직여 비행기의 Rolling을 가져온다.
2) Pitch Moment는 뒤에 붙어 있는 수평 꼬리날개의 엘리베이터에 의해 발생한다. 비행기의 양 앞날개를 축으로 하여 비행기의 상승 하강을 결정한다.
3) Yaw Moment는 주로 꼬리날개에 붙어 있는 러더에 의해 발생하며 수직 축을 중심으로 하여 좌우로 움직인다. 비행기의 주행 방향에 해당한다.

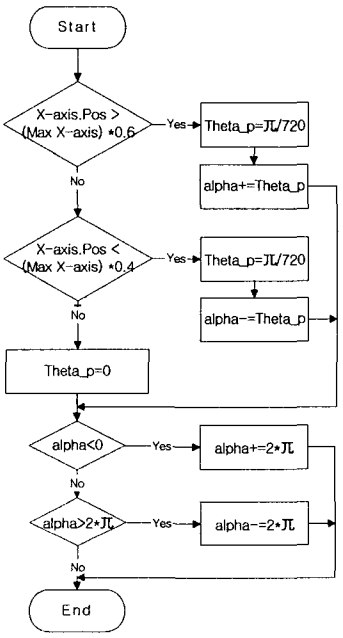
다음은 Roll Moment의 값을 구하기 위한 알고리즘이다.

```
if ( 조이스틱 X축 기울임값 > (조이스틱 X축 최대값) * 0.6 ) {
        Theta Moment = (PI/360); // 일정한 값을 입력.
        Alpha Moment += Theta Moment; // Roll에 대한
Alpha값 증가 }
  else if ( 조이스틱 X축 기울임값 < (조이스틱 X축 최대값) * -
0.4 ) {
        Theta Moment = (PI/360); // 일정한 값을 입력.
        Alpha Moment -= Theta Moment; // Roll에 대한
Alpha값 감소 }

  else   Theta Moment = 0; // 변화량이 없음
  if ( Alpha Moment < 0 )

        Alpha Moment += (float)(2*PI); // Alpha 값이 음수값
을 가질 경우
  else if ( Moment.alpha > (2*PI) )
        Alpha Moment -= (float)(2*PI); // Alpha 값이 360도를
넘을 경우
```

위와 같이 조이스틱 X축을 좌우로 움직였을 때의 값들을 통해 Roll Moment의 값이 변화한다. 또한 Pitch Moment도 동일한 알고리즘을 사용하였으며 Yaw Moment는 Pitch와 Roll의 값을 조합하여 나타낼 수가 있다.



<그림 4> Roll Moment값을 구하는 Flow Chart

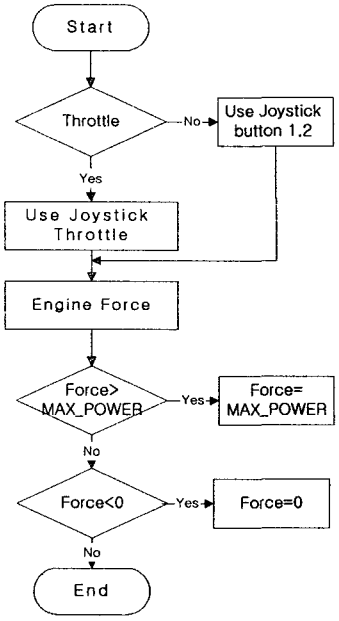
위의 플로차트는 Roll Moment에 대한 것으로 나타낼 수 있다.

또한 이러한 Moment들을 나타내기 위해서는 시간적 요소를 추가하여야 한다. 이러한 시간적 요소는 Flight Simulation에서 Engine Part에서 담당하였다. 현재 35FPS를 사용하여 1초에 화면이 35번 바뀌도록 하였다. 이 수치는 눈에 있어서 깜박거림을 느낄 수 없는 정도의 수치를 나타낸다. 계기판 부분에서는 별 차이는 없지만 Vision Part에서는 중요한 요소로 사용될 수 있는데 이는 Vision Part에서는 지형을 렌더링 하여야 하며 많은 데이터를 처리하여야 하기에 초당 프레임 수가 중요하게 여겨진다.

Engine Part에서는 항공기 엔진의 추력과 X, Y, Z 축에서의 속력을 계산하여 전체적인 항공기의 속력과 위치를 파악할 수 있게 된다.

3.3.2 엔진 부분 알고리즘

다음은 엔진의 추력과 속력 부분의 플로차트이다.



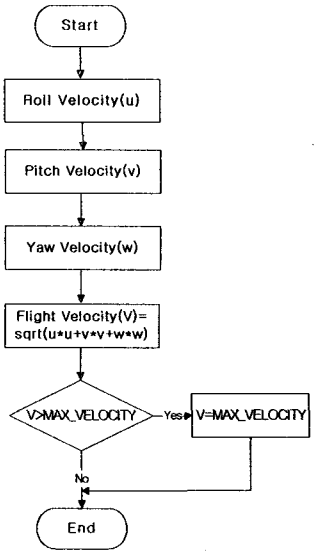
<그림 5> Engine Force값을 구하는 Flow Chart

위의 엔진 Power와 항공기의 속력을 통해 속도계, 고도계, 방향계, 자세계 등을 표현하기 위한 준비를 하였다.

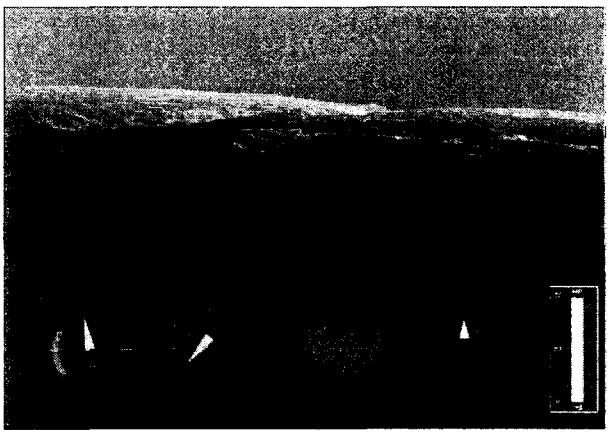
실제 Flight Panel에서는 위에서 나온 출력 값을 통해 각각의 계기판에 값을 전달하게 된다. 이러한 값을 통해 실제 화면에서 현재의 비행 속도, 고도, 기울어짐, 방향을 알아 볼 수 있다.

3.3.3 전체적인 항공기 시뮬레이터

다음은 Flight Simulator의 전체적인 화면이다.



<그림 6> Flight Velocity 값을 구하는 Flow Chart



<그림 7> Flight Simulator의 전체적인 화면

이러한 전체적인 부분에서 Panel Part로 나누어 보았다.

다음은 기본적인 순서를 나타낸 것이다.

1. 화면을 3:5로 분할하여 3을 Panel Part로 사용한다.
2. 콕피트를 구성하여 배경화면과 계기판 사이를 구분할 수 있게 한다.
3. 각 계기판의 기본 틀을 Function Call을 사용하여 나타낼 수 있도록 Function화 한다.
4. 각각의 계기판을 분리 위치시킨다.
5. 해당 계기판이 그려질 곳에 Function Call을 사용하여 기본 틀을 그린다.
6. 조이스틱 조종을 통해 값을 Engine Part에 전달한다.
7. Engine Part에서 계산된 값을 각 계기판에 전달한다.
8. 각 계기판에서는 전달받은 값을 화면에 Display한다.

이러한 계기들은 전체 윈도우를 보면 우선 Vision Part에서 지형을 보여주기 위해 사용되는 구간이 윈도우의 상단 5/8에 해당하는 부분이므로 이 부분을 다음과 같은 OpenGL 함수를 사용하여 제한하였다.

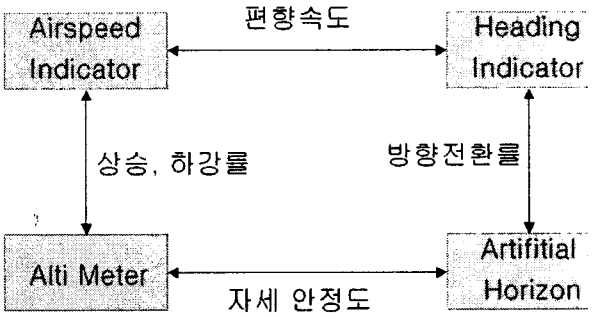
```
/*OpenGL로 그릴 윈도우의 영역을 설정한다.*/
glViewport(0, 3 * window_cy/8, window_cx, 5 * window_cy/8);
/*glScissor로 설정된 영역에서만 Vision part가 그려지게 한다.*/
glScissor(0, 3 * window_cy/8, window_cx, 5 * window_cy/8);
```

이와 같이 Vision Part에서 지형을 모두 그린 후에는 뷰포트를 전체 윈도우로 재설정한다. 현재 Vision Part에서 시각 좌표계를 사용하여 변환에 관계없이, 관측자의 시점에서 바라본 자표로 스크린 좌표를 설정할 수 있도록 설정되었는데 Panel Part에서는 그럴 필요가 없으므로 항상 일정한 방향으로만 보이도록 고정을 시켜주어야 한다.

```
//gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)
//eyex, eyey, eyez : 눈의 위치에 해당하는 x,y,z 좌표 값이다.
//centerx, centery, centerz : 바라보는 화면의 중심에 해당하는 x,y,z 좌표 값이다.
//ups, dpy, ups : up 벡터를 나타내는 x, y, z 좌표 값이다.
gluLookAt(0,0,0,0,0, -1, 0,1,0);
```

다음과 같은 OpenGL 함수를 사용하여 z축의 -1인 점에서 x-y 평면에 계기판을 고정시켜주는 것이다. 그리하여 Panel Part가 Vision Part에 상관없이 Panel을 나타낼 수가 있다.
콕피트(cock-pit) 부분은 항공기에 있어서 바깥부분과 안쪽 부분을 분리해주는 역할을 하고 있고 윈도우의 하단 3/8에 해당하는 부분에는 각 계기판이 균일하게 분할하여 나타내었다.
각각의 계기판은 다음과 같다.

이러한 계기판들은 <그림 6>과 같이 서로 연관돼서 작용하게 된다.



<그림 8> Indicator의 Data의 동기화

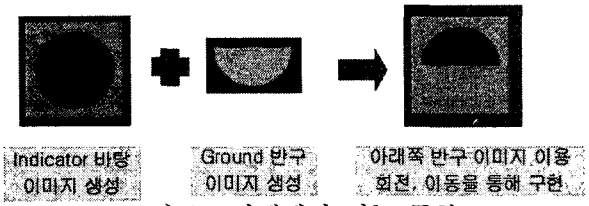
1) 속도계, 고도계, 방향계

속도계, 고도계, 방향계는 이미 만들어 계기의 틀을 사용하여 기본 바탕을 만든 후 각각에 필요한 부분을 추가시켜 준다. 속도계는 비행속도를 표시하며 단위는 Knot를 사용하며, 고도계는 비행고도를 표시하며 단위로는 Feet를 사용하고 방향계는 0도를 북으로 하며 시계방향으로 회전하면서 동->남->서->북으로 표시를 한다.

2) 자세계

계기의 틀을 바탕으로 하는 것은 위와 동일하지만 바탕 이미지를 하늘색으로 설정하고 반구의 흙색을 통해 비행기의 자세의 모양을 나타

내게 된다.



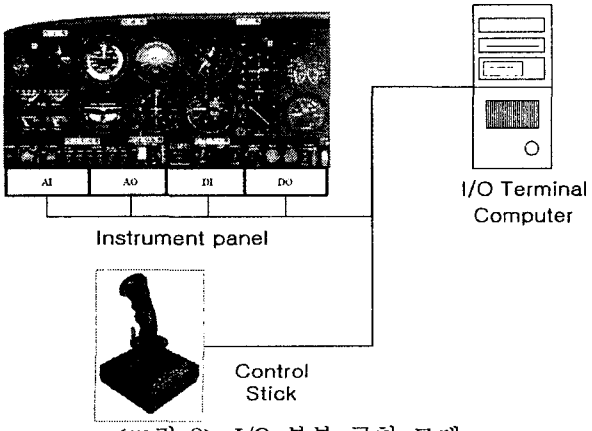
<그림 9> 자세계의 기능 구현

이러한 계기판은 각각의 기본 틀을 만들어 주는 함수를 사용하여 계기판이 형성되기 전에 기본 틀을 형성해주고 그 위치에 속도계, 고도계, 자세계, 방향계를 덧씌우는 것이다.

각각의 계기는 Engine Part에서 전달되는 값들을 받아서 화면에서 즉시 보여주는 역할을 하는 것이다. 그러나 이러한 것들을 보여주기 위해서는 화면의 깜박거림을 제거하여야 했는데 이는 현재 싱글 버퍼링을 사용하였던 방법을 더블 버퍼링이란 기능으로 대체 사용하여 해결을 하였다.

3.4 I/O Part 구현

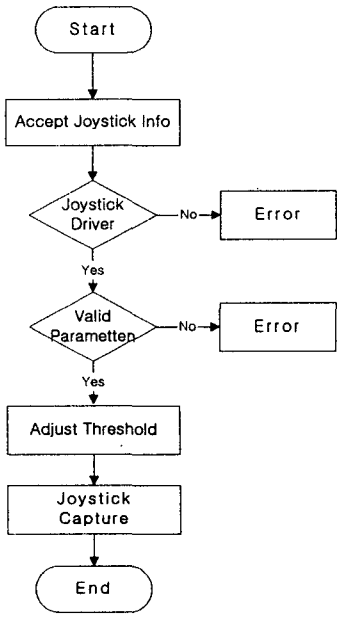
시뮬레이터의 조정석의 계기판에 있어 조이스틱 및 스위치들을 컴퓨터와 연결시켜 주는 부분이다. 계기판의 각종 스위치들과 조이스틱은 I/O 카드의 터미널 단자에 연결된다.



<그림 8> I/O 부분 구현 모델

실제적으로는 입력되는 스위치는 키보드를 통해 입력되는 값들로서 조이스틱과 마찬가지로 값을 Engine Part로 전달하여 주는 부분으로서 화면상의 항공기의 계기판에 Display해 준다.

다음은 조이스틱 초기화 부분과 조이스틱의 운전을 통해 값을 받아들이는 부분을 보여주고 있다.



<그림 9> Joystick Capture값을 구하는 Flow Chart

4. 결론

본 과제에서 구현한 Flight Simulator 개발은 PC를 기반으로 한 항공기 Panel에 대해 구현을 하였으며 항공기 Panel의 모델링 간소화에 목적을 두고 구현하였다.

항공기에 대한 모델링 중 관성에 의한 모멘트는 수학적 모델링을 통한 수치 적분으로 항공기가 기체에 대한 관성 모멘트를 구현하였고 항공기의 외부적인 영향, 즉 공력 데이터의 경우 많은 비선형적인 요소에 의해 수학적 모델의 복잡함에 의한 실시간 연산에 대한 프로세스의 부

담감을 줄이기 위해 계산되어지거나 실험에 의한 데이터들을 테이블로 만들어 검색 알고리즘을 사용한 공력 데이터를 사용하였다.

조종간 조작에 따른 엘리베이터의 움직임으로 변화해 가는 플렙각과 받음각에 대해 항공기의 상승계수를 사용하였으며, 각각의 받음각은 실험치로 구한 비행 상승 속도를 비행시뮬레이터에 적용함으로써 연산에 필요한 계산량이 줄어들 수 있었으며, 2차원의 간단한 테이블 검색으로 항공기의 정밀한 모션을 실시간으로 표현해 줄 수 있었다. 틀 운동에 의한 선회비행은 양력과 항공기 무게에 대한 중력과 원심력과의 합력에 대한 값을 힘에 대한 벡터값의 합을 이용하여 항공기의 미끄러짐 값을 계산하여 항공기의 속도에 적용하였고, 이와 같은 값들을 토대로 하여 실제와 같은 상황을 나타낼 수 있게끔 하였다.

본 논문에서 구현한 Flight Simulator와 현재 국내에서 사용하고 있는 Flight Simulator를 비교하면, 패널 부분이 하드웨어가 소프트웨어로 구현되어 있으며 국산 기술로 구현되어 보다 보편적인 시뮬레이터로 발전할 가능성이 있다. 기존의 시뮬레이터는 이륙부터 비행, 착륙까지 구현되어 있으나, 본 시뮬레이터에서는 비행 상태만을 후련 조종사에게 습득할 수 있도록 하였다.

Flight Simulator라는 특성상 비행기의 비행 자세만을 고려할 수 없으므로, 비행기의 이착륙 및 비행 상태, 공군 등에서 사용하게 될 비행 전술 훈련이나 모의 전투 훈련 등을 위해서 하나의 서버급 Main Computer와 여러 대의 Client Simulator가 서로 연결되어 Client에서 보내는 정보를 Server에서 프로그램을 받아 스크린을 통해서 훈련 비행사들에게 보여 줌으로써, 개인의 훈련뿐만이 아니라 여러 형태의 훈련도 가능해 질 것이므로 이런 방향으로 본 시뮬레이터 역시 발전해야 할 것이다.

참고문헌

[1] J.M. Rolfe, K.J. Staples, "Flight Simulation, Cambridge aerospace series", 1997

[2] 한국항공우주연구소, "Development of a R&D Flight Simulator(4)", 과학기술처, 1989

[3] Barnes, A. G., "Modeling requirement in flight Simulation" Aeronautical Journal, vol.98, pp 395-404, 1994

[4] 윤석준, "항공기 시뮬레이터 기술의 현재", 전자공학회지, 제25권 제2호, pp172-181, 1998

[5] Gawron, Valerie J., "When in-flight simulation is necessary", Journal or Aircraft, v.32 n.2, pp411-415, 1995

[6] 한국 항공우주 진흥 협회, "비행시뮬레이터 개발". 항공우주, pp18-21, 1994.11

[7] Pineiro, Luis A., "Real-time parameter identification applied to flight simulation", IEEE Transaction on Aerospace and Electronic System, v.29, n.2, pp290-301, 1993

[8] Chapleo, A. Q., "Some university experie -nces of aircraft simulator research", Aeronautical Journal, Vol.84, pp223-226, 1980

[9] 김정욱, 고상호, 고진영, 정일용, 이기범, 오태식, "PC를 사용한 연구용 시뮬레이터 개발", 한국항공 우주학회지, 26권 1호, pp120 -128, 1998

[10] 윤석준, "모의훈련 장비 구현을 위한 모델링 및 실시간 시뮬레이션 기법 연구", 한국항공 우주학회지, 27권 4호, pp143-149, 1999

[11] 이호근, "비행성 평가 시뮬레이터개발", 한국항공우주학회지, 제24권 제5호, pp157-164, 1996

[12] Hassan B.Diab, "Design and Implemen -tation of a Flight Simulation System", American University of Beirut, 1992

[13] Komoda, Masaki, "Some experience in

in-flight simulator development and flight simulation", 한국자동제어학술회의논문집 (국제학술편); Seoul National University, Seoul; 20-22 Oct. 1993, ppSS.-SS.8, 1993

[14] Richard S.Wright, Jr., Michael Sweet, "OpenGL SuperBible, Second Edition", Waite Group Press, 1999

● 저자소개 ●

이준우

2001 성균관대학교 전기전자컴퓨터공학부 졸업

2001~현재 현대 모비스 재직

채상원

2001 성균관대학교 전기전자컴퓨터공학부 졸업

2001~현재 성균관대학교 전기전자컴퓨터공학부 석사 재

이칠기

1980 성균관대학교 전자공학과 졸업

1979~1983 한국방송공사 근무

1985 Arizona State University 전기 및 컴퓨터 공학과 석사

1990 University of Arizona 전기 및 컴퓨터 공학과 박사

1990~1995 삼성전자 수석연구원

1995~현재 성균관대학교 전기전자 및 컴퓨터 공학부 조교수

관심분야 : 컴퓨터 시뮬레이션, 객체지향 모델링, 공장자동화, 전문가 시스템