Proceedings of the 12th
KACC, October 1997

굴삭기 주행 시뮬레이터를 위한 통합 프로그램

An Integrated Program of Driving Simulator for Excavators

°유창훈*, 손권**

*부산대학교 기계공학과(Tel:+82-51-510-3066; E-mail:chyou1@hyowon.cc.pusan.ac.kr)
**부산대학교 기계공학과(Tel:+82-51-510-2308; Fax:+82-51-512-9835; E-mail:kson@hyowon.cc.pusan.ac.kr)

Abstracts An integrated program of driving simulator has been developed for excavators using the Motif, OpenGL, and C compiler. The developed program not only offers a GUI but also covers graphic algorithms, therefore, the user can easily run the driving simulator whose components include a simplified visual graphics system. Several graphics technics are combined and applied to the simulator program in order to increase the speed of graphical representation, which access computer memories, mix 2D models with 3D ones, and use the basic position detection method. A text format environment file has been utilized for organizing more flexible driving circumstances.

Keywords Integrated Program, Simulator, Graphic User Interface(GUI), Basic Position Detection, Graphic Library

1. 서론

차량 및 항공기의 성능 평가와 향상을 위해 제작되어 사용되는 시뮬레이터는 해석 대상의 특성에 맞게 운영하는 프로그램이 필수적이다. 시뮬레이터 통합 프로그램은 그래픽 처리용 컴퓨터 및 동역학 해석용 컴퓨터, 운동을 재현하는 6자유도 스튜어트 플랫폼, 주행환경을 보여주는 프로젝터 등의 하드웨어를 총괄하는 역할을 담당한다.

최근에 개발되는 응용 프로그램은 사용자가 키보드를 이용하지 않더라도 마우스를 이용하여 아이콘이나 메뉴를 조작함으로써 간단하게 프로그램을 수행할 수 있도록 하는 윈도우라는 개념 바탕 위에 개발되고 있다. 통합 프로그램도 이러한 추세를 따르고 있으며, 많은 데이터를 처리하기 위해 유닉스(Unix) 시스템에서 동작하는 실시간 데이터 처리 프로그램을 모티프(Motif)와 C 컴파일러를 이용하여 제작하기도 하였다[1].

운전자는 주로 시각을 통해 주행 상황과 환경을 인지하므로 시뮬레이터에서 그래픽 처리는 현실감을 위해 대단히 중요하다. 그래픽 처리 과정은 계산량과 데이터의 양이 방대하여 주로 서버급의 컴퓨터가 사용되어 왔으나[2], 최근에는 시스템 성능의 향상으로 가격 면에서 저렴한 워크스테이션이나 PC에서 그래픽 가속기와 OpenGL 라이브러리를 이용하여 주행 환경 이미지를 현실감 있게 표현하기도 하였다[5]. 그리고, 주행 시나리오를 작성하기 위하여 아이오와 주행 시뮬레이터(IDS)는 타일 형식(tile-based)을 기초로 한 주행 환경을 사용하였다[3].

본 연구에서는 동역학적으로 해석된 굴삭기의 주행 상황을 그래픽 프로그램으로 보내어 시뮬레이션을 수행하고, 시뮬레이션이 종료한 후에는 결과를 도식적으로 나타내는, GUI 환경의 통합 프로그램을 모티프와 OpenGL 라이브러리와 함께 C 컴파일러를 통해 제작하였다. 제작된 프로그램은 그래픽 처리 과정을 포함하고 있으므로 최소한의 해석 결과로도 그래픽 화면을 구성할 수 있도록 하였다. 동역학 해석 및 그래픽 처리에 필요한 주행 환경의 데이터를 블록 단위로 설정하여, 별도의 컴파일 없이 사용할 수 있도록 하였다. 속도 향상을 위해 몇 가지 기술을 적용하였고, 필요에 따라 대상을 선택적으로 표현하는 기능도 첨가하였다.

2. 통합 프로그램

2.1 프로그램 운용 시스템

시뮬레이터에 사용되는 많은 양의 데이터를 빠르게 처리하기 위하여 실리콘그래픽스사의 워크스테이션(SGI Indigo2 Impact)을 사용하였다. 이 컴퓨터는 64 bit 구조의 R4400 CPU(250 MHz)를 탑재한 그래픽 시스템으로 고속 그래픽 처리 능력을 갖추고 있다. 3차원 그래픽 데이터를 고속으로 처리하기 위하여 24 bit Z 버퍼를 갖추고 있으며 1280×1024의 해상도를 갖고 있다. 운영체계로는 유닉스 계열의 Irix 5.3을 사용하고, X-윈도우와 3차원 모델링에 유용하게 사용되어지는 그래픽 라이브러리인 OpenGL과 프로그램 제작 도구인 모티프를 제공한다.

모티프는 유닉스 환경의 그래픽 표준인 X-윈도우에서 동작하는 사용자 인터페이스 라이브러리이며, C*컴파일러 및 X, Xt 라이브러리 등과도 호환이 가능하다. 편리한 GUI 환경을 제공하는 프로그램 제작에 널리 사용되며, 상용 패키지에도 많이 활용되고 있다. 프로그램 제작시에 원하는 인터페이스 집합을 코드에 포함시킴으로써 원하는 사용 환경을 쉽게 구성할 수 있다.

음영 처리된 3차원 CAD 모델을 표현하기 위해 OpenGL 그래픽 라이브러리를 사용하였다. OpenGL은 그래픽 전문회사인 실리콘그래픽스사에서 개발한 IRIS GL을 다른 시스템에서도 사용할 수 있도록 수정한 것으로 최근에 널리 활용되어 3차원 그래픽 라이브러리의 표준으로 자리잡고 있다. OpenGL은 3차원 데이터를 직접적으로 사용할 수 있으며, 하드웨어에 구속받지 않고 다른 그래픽 라이브러리보다 비교적 속도가 빠른 장점을 갖고 있다. OpenGL은 기본적인 기하학뿐 아니라 뷰(view) 설정, 좌표 변환과 빛의 효과 등을 고려한 함수도 제공한다. 보조적으로 지원되는 GLX 라이브러리를 이용하면 모티프와 연계한 프로그램을 작성할 수 있다[4].

개발된 프로그램은 사용자 인터페이스, 그래픽 처리, 시뮬레이션 데이터 입출력, 주행환경 데이터 베이스 모듈로 구성된다. 모듈별로 제작된 각각의 프로그램은 모티프와 OpenGL 라이브러리와 함께 C 컴파일러에 의해 하나의 실행 가능한 통합 프로그램으로 완성된다. 각 모듈은 독립적이지 않고 서로 연계되어 동작된다.

2.2 사용자 인터페이스

사용자 인터페이스 부분은 모티프를 이용한 이벤트 구동방식을 사용하였다. 이벤트 구동방식은 윈도우 환경에서 특정 영역을 클릭(click)함으로써 이에 해당하는 함수를 실행시키는 형식이다. 모든 기능은 풀다운 메뉴를 기본으로 제작되었고, 필요에 따라 팝업 메뉴와 아이콘 메뉴를 첨가하여 사용자의 편의성을 도모하였다. 각 메뉴는 마우스를 이용하여 간단히 조작할 수 있으며, 풀다운 메뉴는 키보드를 이용한 단축키로도 활성화된다.

프로그램은 사용자가 조작하는 화면과 프로젝트를 통해 투영될 주행 화면으로 크게 구성되며, 그림 1과 2는 각각 사용자 화면과 주행 환경 화면을 나타낸 것이다. 사용자 화면의 상단에는 풀다운 메뉴가 있고, 오른쪽에는 아이콘 메뉴 및 굴삭기의 위치와 자세에 대한 정보가 표시된다. 풀다운 메뉴 아래에 위치하는 큰 작업 영역에는 시뮬레이션 과정 및 결과 그래프뿐 아니라, 주행 환경 지도, 굴삭기 형상 등이 그래픽 처리되는 영역이다. 그래픽 영역에서 마우스를 이용하여 화면 확대, 회전 및 병진 이동을 수행할 수 있고, 팝업 메뉴도 활성화시킬 수 있다.

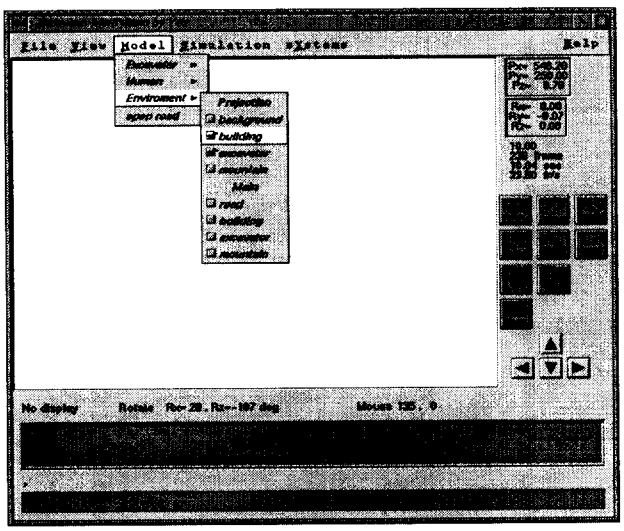

그림 1 사용자 화면
Fig. 1 User interface screen

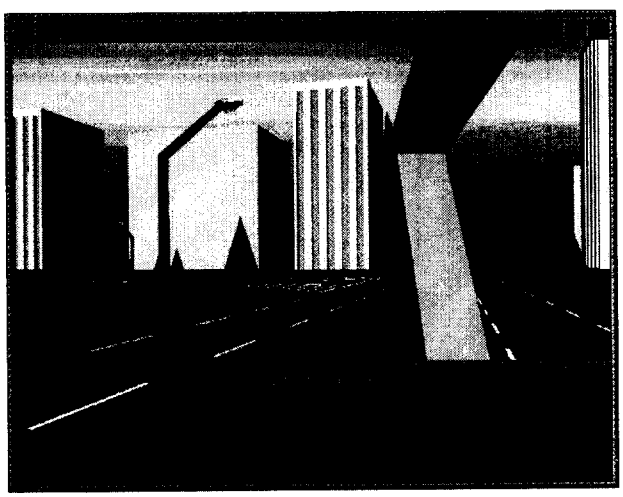

그림 2 주행 환경 화면
Fig. 2 Driving environment screen

2.3 그래픽 처리

시점으로부터 가까이 있는 대상은 크게, 먼 곳에 위치한 대상은 작게 표현된다. 이러한 원근 효과는 먼 거리에만 존재하는 대상에서는 중요하지 않고, 또한 대상이 시점에서 너무 가까이 존재하면 형상이 찌그러지는 왜곡 현상을 유발시키기도 한다. 그러나, 굴삭기 시뮬레이터와 같이 주변 환경을 표현해야 하는 경우에는 꼭 필요하며, 특히 주행 시뮬레이션에서는 움직이는 현실감을 표현에 중요한 역할을 한다.

음영 처리는 시각 시스템의 현실감을 위한 최소한의 요건으로 그래픽 처리의 대부분의 시간을 소비한다. 많은 데이터의 양을 동시에 처리해야하는 시뮬레이터의 특성을 고려하여 주변 환경을 최적화 하는 기술이 필요하다. 이러한 몇가지 기술에 대해서는 3 장에서 설명한다.

그림 2는 원근효과와 빛의 효과를 고려한 부드러운 음영처리 방법으로 제작한 굴삭기의 CAD 모델이다. 곡면을 표현하기 위해 연속적인 면의 벡터를 사용하였다. 굴삭기의 상부 프레임 및 작업 장치는 계층적 좌표계를 이용하여 운동이 가능하다.

2.4 시뮬레이션 데이터 입출력

시뮬레이션 수행중에는 주행시 굴삭기의 위치와 자세에 대한 자료가 필요하다. 현 단계에서는 동역학 해석 컴퓨터에서 계산된 굴삭기의 위치와 자세의 정보를 시간의 함수로 파일로 저장한 후, 통합 프로그램에서 읽어서 그래픽 처리 과정으로 보내는 시뮬레이션을 수행한다.

시뮬레이션 종료 후에는 시뮬레이션 결과를 도식적으로 나타내어 굴삭기의 가속·제동 성능 평가에 용이하게 사용할 수 있도록 하였다. 기본적인 결과인 위치와 자세에 대한 정보, 속도와 가속도의 각 방향의 값을 시간에 대한 그래프로 나타낼 수 있다. 그림 4는 도식화의 한 예로 평지를 초기에 10 m/s의 속도로 주행하던 굴삭기가 4 초 후에 제동했을 경우 굴삭기 중심의 Z 축에 대한 변위를 나타낸 것이다.

2.5 주행 환경 데이터베이스

굴삭기 주행 환경은 정해진 규칙에 따른 주행 환경 파일을 제작하여 읽어 들여 사용한다. 주행 환경 파일은 텍스트로 구성되어 작성과 수정이 용이하고, 별도의 컴파일 없이 사용할 수 있는 장점이 있다. 주행 환경 파일은 블록 단위로 작성된다. 가장 큰 블록을 가로, 세로의 크기를 500 m의 정사각형으로 하고, 그 블록을 4 등분하여 한 변의 크기가 250 m인 정사각형을 두 번째 블록으로 한다. 두 번째 블록은 한 변이 50 m인 정사각형 블록 25개로 구성되며 이 25개의 블록이 최소 단위가 된다.

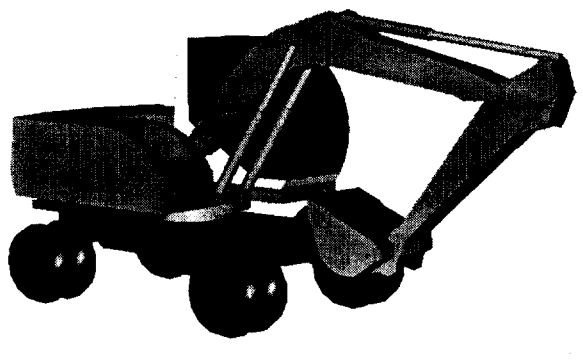

그림 3 굴삭기 CAD 모델
Fig. 3 CAD model of excavator

각 블록은 왼쪽 아래에서부터 시작하여 00, 10, 20 ... 으로 번호가 매겨진다. 블록의 번호는 자신이 소속된 블록에 대한 상대적인 값이므로 소속 블록의 번호만 바꿈으로써 전체 영역을 이동시킬 수 있으며, 주변 환경을 첨가하거나 확장할 때에도 간단하게 수정할 수 있다.

그림 5는 주행 환경 블록의 예를 나타낸 것이다. 빗금친 영역으로 표시된 정사각형이 가장 큰 블록이며, 번호가 매겨진 각각의 정사각형이 두 번째 블록이 된다. 그림에서 길게 연결된 선은 주행 도로를 나타낸다. 주행 환경 블록은 도로뿐 아니라 건물이나 자연물을 포함할 수도 있다. 주행 환경을 구성하기 위해서는 표 1과 같은 주행 환경 파일을 사용한다. 주행 환경 파일은 텍스트 형태이므로 작성 및 수정이 용이하고, 별도의 컴파일 없이 사용할 수 있다.

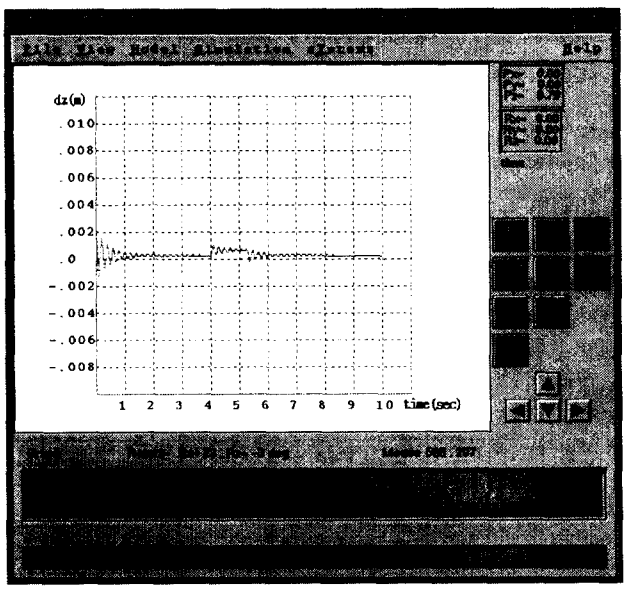

그림 4 결과의 도식화
Fig. 4 Presentation of result

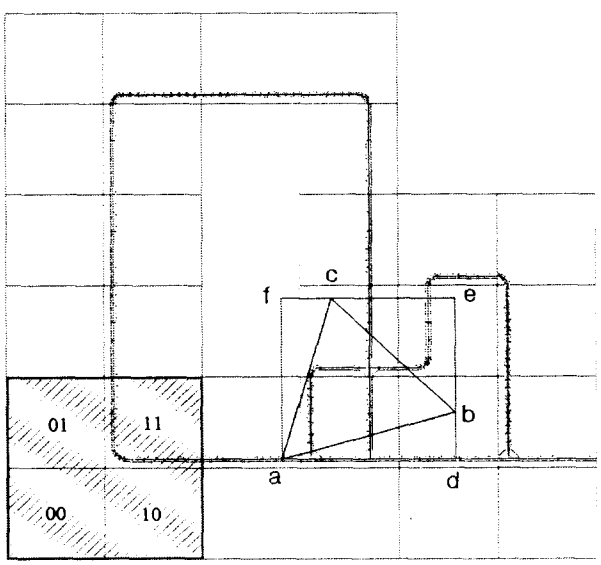

그림 5 주행 환경 블록
Fig. 5 Driving environment block

3. 그래픽 처리 속도

3.1 컴퓨터 메모리를 이용한 CAD 모델의 저장

프로그램이 기동되면 모델 데이터를 읽어서 메모리에 저장하여 디스플레이 리스트(display list)를 생성한다. 모델을 다시 화면에 표시할 때는 메모리에 저장되어 있는 데이터를 직접 이용하여 그래픽 처리에 이용하므로 처음보다 속도가 향상된다. 이러한 방법은 모델의 형상과 사용 횟수에 따라 2배 이상의 속도 향상을 나타내었다. 메모리를 이용한 방법은 사용 횟수가 많을수록, 모델의 자료가 복잡할수록 속도에서 많은 향상을 기대할 수 있다. 표 2는 메모리를 이용하지 않고 그래픽을 처리한 것과 메모리를 이용하여 디스플레이 리스트를 생성한 것과의 속도를 비교한 것이다. 속도는 100 프레임의 그래픽을 처리할 때 소요되는 시간을 계산하여 초당 처리하는 프레임 수를 구한 것이다.

3.2 2차원과 3차원 모델의 혼합

먼 거리에 위치한 배경은 원근 효과가 거의 나타나지 않고, 시점도 거의 고정되어 있다. 이러한 대상의 표현을 위해서 3차원의 자료를 처리한다는 것은 비효율적이므로, 먼 곳의 배경을 표현하기 위해 2차원 모델을 사용하였다. 2차원으로 그려진 그림을 수직으로 세워진 사각형의 면에 텍스쳐 맵핑시킴으로써 2차원 배경을 만든다. 텍스쳐 된 2차원 배경은 기준좌표계에 고정시키지 않고, 시점 즉 굴삭기와 일정 간격을 두고 이동하도록 하여 항상 그래픽 처리 영역 안에 있도록 하였다. 그림 2의 주행 환경에서 하늘과 구름, 멀리 있는 산의 모습은 이러한 기법을 통해 구성하였다. 다른 대상과는 다르게 텍스쳐 배경은 2차원이므로 굴삭기의 회전시 배경의 움직임을 나타내기 위해 굴삭기의 회전에 대응하는 각도만큼 평행 이동시켜 표현하였다.

표 1 주행 환경 파일
TABLE 1 Driving environment file

filename : road.map
500x500 1 1
250x250 1 0
50x50 0 0 curve 2
50x50 1 0 road 2
50x50 2 0 road 2
50x50 3 0 inter 5
50x50 4 0 road 2
:
250x250 1 1
50x50 0 0 road 2
50x50 1 0 road 2
50x50 2 0 road 2
50x50 3 0 curve 1
end
end500
endall

표 2 그래픽 처리 속도 비교
TABLE 2 Graphical representation speed in frame/sec

	메모리 사용 않음	메모리 사용
도로	12.0	14.5
도로+굴삭기	9.1	11.7
도로+굴삭기+건물+산	4.0	8.5

3.3 기준 위치 검색을 통한 대상 제거

주행 시뮬레이터는 다른 그래픽과는 달리 주변에 존재하는 대상의 수가 많다. 많은 대상 중 시야에 나타나는 것은 일부분이지만, 시야에 나타나지 않는 대상도 그래픽 처리시 계산되어 많은 시간이 소요된다. 본 연구에서는 실제 존재하지만 시야에 나타나지 않은 대상은 계산에서 제외하는 방법을 사용하여 그래픽 속도를 향상시켰다. 주행 환경 파일을 읽어서 구성한 주행 환경(그림 5)에 나타난 블록은 두 번째 단계의 블록으로 4개의 블록이 하나의 큰 블록을 생성한다. 두 번째 단계의 블록은 25개의 작은 블록으로 구성되며 기준 위치 검색의 최소 단위가 된다.

그림 5에서 a는 굴삭기의 위치이며, 삼각형 a-b-c는 시야에 나타나는 영역으로 이 영역 내에 포함되는 대상만이 시뮬레이터의 주행 환경으로 나타난다. 시야 영역의 기준위치는 삼각형의 꼭지점을 포함하는 직사각형 a-d-e-f의 중심이며, 주변 환경의 기준 위치는 각 단계의 블록의 중심이 된다. 시야 영역의 기준 위치 $V.basic$과 시야 영역의 크기 $V.size$는 다음 식과 같이 표현된다.

$$V.basic = \frac{1}{2}(Max(a,b,c) + Min(a,b,c)) \quad (1)$$

$$V.size = |Max(a,b,c) - Min(a,b,c)| \quad (2)$$

주행 환경의 기준 위치와 크기는 주행 환경 블록에 의해 고정된 값으로 $P.basic$와 $P.size$로 나타내고, 시야 영역과 비교하면 다음의 식을 얻을 수 있다.

$$|P.basic - V.basic| \le \frac{1}{2}(P.size + V.size) \quad (3)$$

시야 영역과 주행 환경 영역의 기준 위치를 식 (3)과 같이 비교하여 수식을 만족하지 않으면 영역을 벗어난 것으로 판단하여 계산에서 제외시킨다. 기준 위치 비교에 소요되는 시간을 단축하기 위하여 제일 먼저 가장 큰 첫 번째 단계의 블록 위치를 비교하여, 영역 내에 포함되면 두 번째 단계의 블록 위치를 비교한다. 이러한 방법의 반복으로 범위를 줄여 나간다. 결과적으로 그림 5의 직사각형 a-d-e-f 내에 포함된 최소 단위의 블록만을 계산하여 그래픽 처리를 하게 된다.

표 3은 모든 대상을 처리했을 경우와 기준 위치 검색을 통한 대상을 제거 알고리즘을 적용하였을 때의 속도를 비교한 것이다. 속도는 10 초 동안의 시뮬레이션 중에서 처리된 프레임 수를 계산하여 초당 처리된 프레임 수를 정리한 것이다. 기준 위치 검색으로 대상을 제거하면 다른 블록에서 표현되는 대상의 수가 많아지더라도 현재 블록에서의 처리 속도가 저하하지 않는 특징이 있다.

표 3 그래픽 처리 속도 비교

TABLE 3 Graphical representation speed in frame/sec

	모든 대상 처리	위치 검색을 통한 대상 제거 적용
도로	18.0	35.1
도로+굴삭기	14.4	23.8
도로+굴삭기+건물+산	7.3	18.0

4. 결론

본 연구를 통해 굴삭기 주행 시뮬레이터 전용 통합 프로그램을 개발하였다. 개발된 프로그램은 사용자의 편의성을 고려한 이벤트 구동 방식을 채택하여 GUI 환경에서 작업을 수행할 수 있도록 제작하였다. 주행 환경을 구성하기 위한 텍스트 형태의 파일을 작성하여 주행 환경의 제작과 수정을 쉽게 하였다. 통합 프로그램이 그래픽 처리 과정을 포함하고 있으므로 굴삭기의 위치와 자세에 대한 최소의 정보만으로도 주행 화면을 구성할 수 있도록 하였다. 그래픽 처리 속도를 개선하기 위하여 메모리를 이용하고, 3차원뿐 아니라 2차원 모델도 혼합하여 사용하였다. 기준 위치 검색을 통한 대상 제거 알고리즘을 적용하여 별도의 하드웨어 추가 없이 초당 20~30 프레임 정도의 그래픽 처리 속도 능력을 갖도록 하였다. 시뮬레이션 종료 후에는 시뮬레이션 결과를 도식적으로 나타내어 굴삭기의 가속·제동 성능 평가에 용이하게 사용할 수 있도록 하였다.

참고 문헌

[1] A. A. Romagosa, "Integrated Data Management Tools for Real Time Applications", SAE paper 950415, pp. 111-116, 1995

[2] D. H. Weir and A. J. Clark, "A Survey of Mid-Level Driving Simulators", SAE paper 950172, pp. 53-73, 1995

[3] J. Cremer, J. Kearney, and Y. Papelis, "Driving Simulation: Challenges for VR Technology", *IEEE Computer Graphics and Applications*, pp. 16-20, 1996

[4] J. Neider, T. Davis, and M. Woo, *OpenGL Programming Guide*, Addison-Wesley, mountain View, CA, 1994

[5] 조준희, 박경균, 이운성, 김정하, "차량 시뮬레이터의 시각 및 음향 시스템 개발", 한국자동차공학회지, pp. 476-481, 1997