AMS 그래픽 시뮬레이션을 위한 애니메이터 개발에 관한 연구

김 병 희, 최 병 규

한국과학기술원 산업공학과

Abstract

본 논문에서는 자동화 제조시스템(Automated Manufacturing System, AMS)의 시뮬레이션 분석을 위한 그래픽 애니메이션 S/W(or 애니메이터)의 개발 방법을 제시하고 있다. AMS의 설계 및 분석을 위한 많은 방법들 중에서 컴퓨터 시뮬레이션이 가장 적합한 방법으로 여겨지고 있으며, 그래픽 애니메이션은 시뮬레이션 모델의 검증(verification)과 타당성 평가(validation), FA 기술자와의 의사전달, 시뮬레이션 결과의 발표자료 등으로 시뮬레이션 수행과정에서 필수적인 도구로 사용되고 있다.

본 연구에서 제시된 애니메이터는 대상 AMS의 3차원 layout을 생성하는 기능과 시뮬레이션 프로그램에 의해 생성된 "trace file"을 입력으로 하여 시뮬레이션 진행 과정을 "playback type"으로 애니메이션 하는 기능을 가지고 있다. 본 논문의 주요 연구결과는 다음과 같다. 1) 애니메이션 특성에 따라 AMS 구성 설비들과 작업물을 분류하였고, 2) 애니메이터 시스템의 구조(architecture)를 제시하였고, 3) C++과 GL을 이용하여 애니메이터를 구현하였고, 4) 자동창고 시뮬레이션에 개발된 애니메이터를 적용하여 보았다.

1. 서 론

생산계획, 각종 가공설비 및 물류기기, 작업자들이 유기적으로 구성되어 있는 자동화 제조시스템(Automated Manufacturing System, AMS)의 설계 및 분석을 위한 여러 접근방법 중에서 컴퓨터 시뮬레이션이 가장 적합한 방법으로 여겨지고 있다[Gibson 1993, Green 1985].

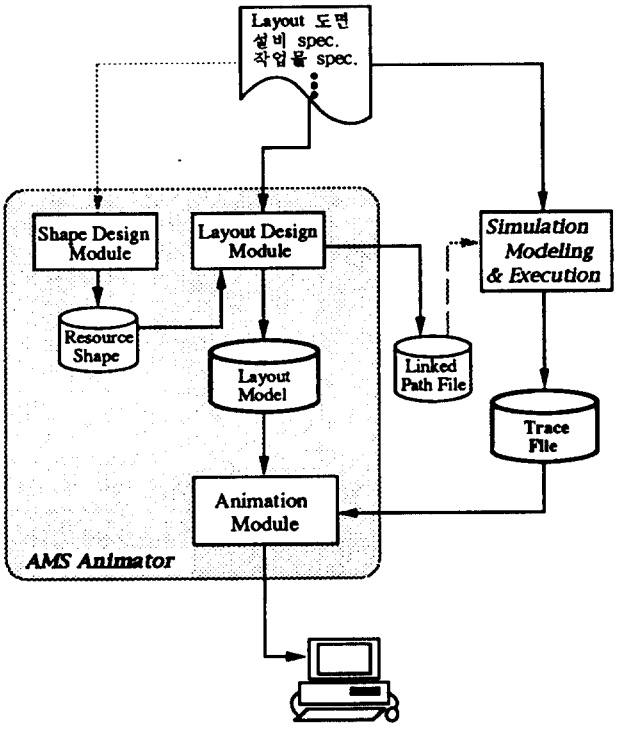
이와 같이 시뮬레이션 연구의 수행 과정에서 그래픽 애니메이션의 유용성은 다음과 같이 요약할 수 있다 [Smith '87, Johnson '88, Law '92].

1) 구현된 시뮬레이션 모델의 검증(verification)
2) 시뮬레이션 모델의 타당성 평가(validation)
3) 시스템 동적 행동의 분석과 이해
4) 의사전달(communication)과 발표자료 등

그래픽 애니메이션은 애니메이션 수행을 위한 정보전달의 시점을 기준으로 concurrent 방식과 playback 방식으로 나눌 수 있다[Law '92]. Playback 방식의 소프트웨어는 애니메이션을 위한 별도의 작업이 필요하나, 시뮬레이션 진행과정의 상태정보를 trace 화일을 통해 직접 입력하게 됨에 따라 매우 빠른 속도의 애니메이션을 수행할 수 있다. 또한 현재의 애니메이션 시간에서 전후의 관심 있는 시점으로의 즉각적인 이동이 가능하므로, 원하는 시점에서의 애니메이션을 신속하고 손쉽게 수행할 수 있는 장점이 있다.

애니메이션의 그래픽 수준에 따라 분류하면 character 그래픽스, bit-map 방식 그래픽스, 2차원 벡터 그래픽스 및 3차원 벡터 그래픽스로 나눌 수 있으며, 점차 3차원 벡터 그래픽스로 발전해 가고 있다[Carrie '88].

본 연구의 목적은 자동화 제조시스템을 대상으로 layout 정보와 구성설비의 속성정보를 입력받아 대상시스템의 layout을 생성하고, 생성된 layout상에서 시뮬레이션 프로그램으로부터 trace 화일을 입력받아 3차원 그래픽 애니메이션을 수행하는 playback 형식의 AMS 전용 애니메이터를 개발하는 것이다.



[그림 1.1] AMS 애니메이터를 이용한 애니메이션 수행과정

[그림 1.1]에는 AMS 애니메이터를 이용한 애니메이션의 수행과정이 나타나 있다. 애니메이션 수행과정을 살펴보면 먼저 AMS 애니메이터의 layout design module을 이용하여 대상시스템의 layout을 입력한 후, 생성된 layout model을 화일에 저장한다. 다음은 외부의 시뮬레이션 프로그램으로부터 생성된 trace 화일(시뮬레이션 수행과정이 담겨져 있음)과 화일에 저장된 layout model을 읽어서 3차원 애니메이션을 수행한다.

2. 애니메이션을 위한 AMS 구성설비의 분류

2.1 Resource의 정의와 분류

자동화 제조시스템을 구성하는 설비들을 애니메이션 표현특성에 따라 다음과 같이 static resource, route resource, moving object로 분류한다.

1) Static Resource

Static Resource는 애니메이션 수행도중 시스템 내부에 고정된 위치를 가지며 위치의 변화없이, 가공(processing), 검사(inspection), 저장(storage), 작업 준비(preparation) 등의 기능을 수행하는 설비를 말한다.

2) Route Resource

Route Resource는 작업물이나 대차(Vehicle) 등이 지나가는 path를 나타내는 설비를 말한다. Conveyor와 같이 스스로 transport를 위한 구동장치를 가지고 있는 설비와 AGV path처럼 transport를 담당하는 대차의 route를 나타내는 설비로 구분된다.

3) Moving Object

Moving Object는 route resource의 path를 따라 시스템 내부를 흘러 다니며 transport 기능을 수행하는 설비(Moving Resource)나 작업물(Job)이다.

2.2 단일설비와 정형화된 물류이송 및 저장 설비의 Library

자동화 제조시스템을 구성하는 설비들 중에서 MCT(Machining Center)나 loading/unloading station은 단일 설비만으로 자신의 기능을 수행할 수 있다. 그리고 Conveyor-net, AGVS, AS/RS와 같은 설비들은 몇 가지 설비들이 모여서 하나

의 정형화된 복합설비를 이룬다. 이와 같은 단일설비와 정형화된 물류이송 및 저장 설비들을 사용자가 편리하게 모델링할 수 있도록 다음과 같이 library화한다. 그리고 각 설비들은 설비간 물류흐름을 연결하기 위한 port를 가지고 있도록 한다.

1) 단일설비

한 개의 static resource로 구성되며 자신의 고유기능을 수행하는 설비이다. 단일설비를 사용하여 MCT나 선반 등의 가공설비, loading /unloading station이나 table 등의 저장설비들을 모델링할 수 있다.

2) Conveyor network

Conveyor는 크게 ground형과 overhead형으로 구분할 수 있으며[Rembold '93], 본 연구에서 언급되는 conveyor network는 ground형 conveyor를 나타낸다. Conveyor network는 직선 또는 원호 형태의 conveyor segment와 방향전환 장치인 diverter로 구성되며 물류의 이송을 담당하는 설비이다. Conveyer net.은 accumulation과 non-accumulation의 두 가지 type을 가진다. Conveyor net.을 사용하여 roller, belt, chain 형 ground conveyor를 모델링할 수 있다.

3) AGV network

직선 또는 원호의 단위 곡선으로 이루어진 path와 path위를 이동하는 대차(vehicle)로 구성되며 물류이송을 담당하는 설비이다. AGV network를 사용하여 AGVS (Automated Guided Vehicle System), RGVS(Rail Guided Vehicle System)등의 무인 운반차 시스템과 overhead conveyor 등을 모델링할 수 있다.

4) AS/RS

AS/RS는 물류의 이송과 저장 기능을 담당하는 설비로, 하나 이상의 rack과 rack사이의 aisle 개수만큼 stacker crane을 갖는다. 이것은 AS/RS을 모델링 하는데 사용한다.

2.3 Job과 Transfer Path

1) Job

Job은 pallet이나 작업물(part)과 같은 물류의 객체를 말한다. Job은 한 개의 moving object로 이루어지며, 애니메이션 수행시 사용자가 임의로 같은 종류의 job을 생성 또는 소멸시킬 수 있다.

2) Transfer Path

Transfer path는 각 설비간 job이 이동되는 경로를 말한다. 이는 각 설비별 I/O port간 transfer 연결 정보와 transfer 시간을 입력받아서 애니메이터 내부에서 자동으로 생성한다. 이때 output port의 제어점은 transfer path의 시작점이 되고 input port의 제어점은 끝점이 된다.

3. AMS 애니메이터의 설계 및 구현

본 연구에서 개발될 애니메이터가 가져야 할 기능적 요구사항을 정리하면 다음과 같다.

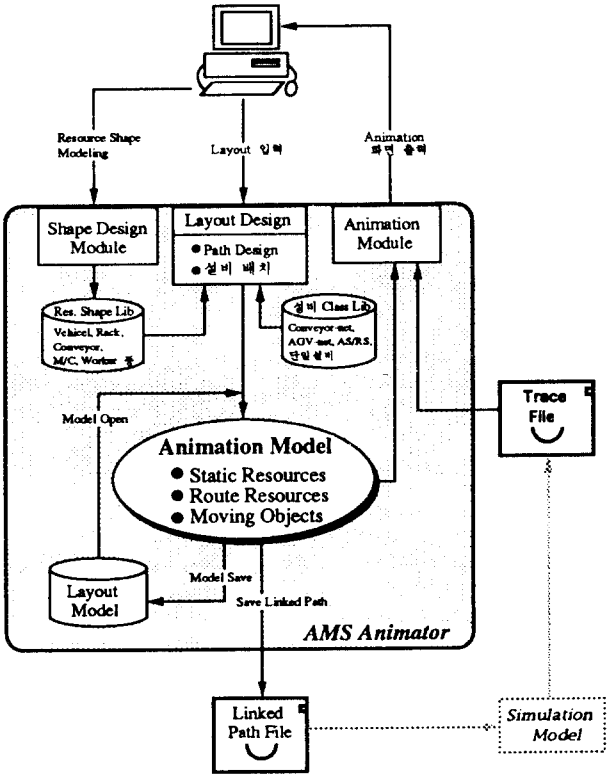
① 시스템의 정적인 layout을 표현.
② 작업물이 이동되는 경로를 표현.
③ Playback 형태의 3차원 애니메이션 수행.

본 장에서는 AMS 그래픽 애니메이터 시스템의 구조(Architecture)를 제시하고 애니메이터 구현에 관한 내용을 다룬다. AMS 애니메이터는 객체지향 언어인 C++로 구현되었고, 3차원 Graphics 처리를 위하여 Silicon Graphics사의 Graphics Library™(GL)[McLendon '91]를 이용하였다. 또한 GUI(Graphic User Interface)는 X Window System™에서 Motif™[OSF '91]를 이용하여 구현하였다.

3.1 AMS 애니메이터의 시스템 구조

AMS 애니메이터는 설비의 형상을 Design하는 Shape Design Module, 각 resource의 속성정보를 입력하고 설비 배치를 하는 Layout Design Module, 그리고 trace 화일을 읽어들여 애니메이

션 명령어에 따라 애니메이션을 수행하는 Animation Module로 구성되어 있다. 애니메이터 시스템의 구조는 [그림 3.1]과 같다.



[그림 3.1] AMS 그래픽 애니메이터의 구조

AMS 애니메이터를 구성하는 3가지 모듈의 기능은 다음과 같다.

1) Shape Design Module

Shape Design 모듈은 제조 시스템을 구성하는 설비들의 형상을 모델링하여 화일에 저장하는 기능을 수행한다. 여기서 모델링 작업을 편리하게 수행하기 위해서 기본적인 primitive와 library가 제공된다. 화일에 저장된 각 설비의 형상은 제조시스템의 layout을 구성하는데 이용된다.

2) Layout Design Module

Layout Design 모듈에서는 애니메이터에서 지원하는 복합설비(Conveyor-net, AGV-net, AS/RS)와 단위설비의 class library를 이용하여 시뮬레이션 대상 시스템의 layout과 route resource의 path를 모델링하고, 애니메이션에 필요한 각 설비의 속성을 입력하는 기능을 수행한다.

3) Animation Module

Animation Module은 layout 모델(layout design module에서 생성)과 trace 화일(외부 시뮬레이션 프로그램에서 생성)을 읽어들여 일정한 시간 간격으로 애니메이션 모델의 상태 변화를 화면에 보여주는 기능을 수행한다.

3.2 애니메이터의 자료구조

본 절에는 C++로 구현된 애니메이터의 기본적인 자료구조와 함수들을 설명한다. 객체지향 언어인 C++은 data abstraction, data encapsulation, dynamic binding, inheritance 등의 특징을 가지고 있다[Stroustrup '91]. 이중에서 inheritance는 상위 class의 데이터와 method를 상위 class로부터 파생된 class가 이어받는 특징을 말한다. 이러한 특징에 의하여 Territory class를 제외한 애니메이터의 모든 class는 root class인 Object class에서 파생된다.

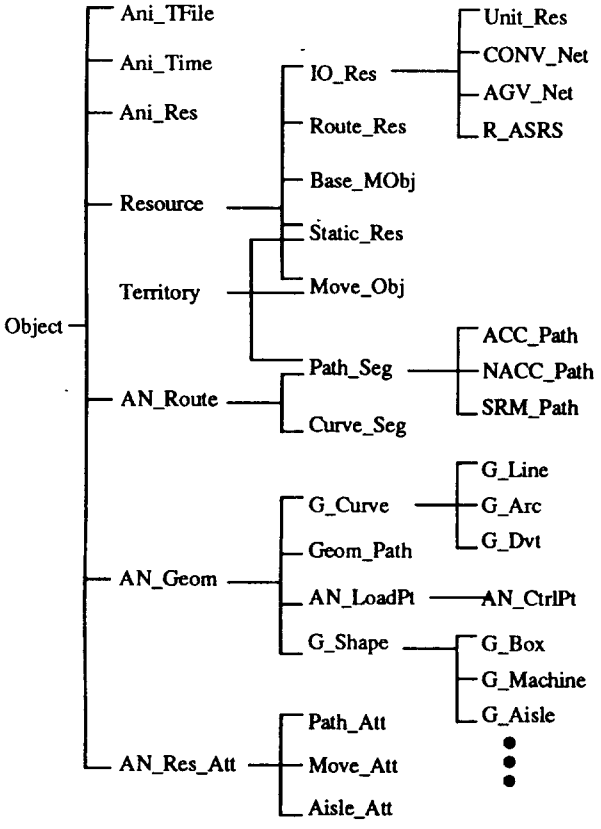
애니메이터의 기본적인 class library를 구축하기 위하여 본 연구에서는 NIH class library를 이용하였다[Gorlen '90]. [그림 3.2]는 GUI와 Graphics에 관련된 class들과 NIH class library을 제외한 애니메이터의 class 계층 구조를 나타내고 있다.

애니메이터를 구성하는 class는 크게 애니메이션 모델을 구성하기 위한 class와 애니메이션 모델을 이용하여 애니메이션을 수행하기 위한 class로 나눌 수 있다.

먼저 애니메이션 모델을 구성하는 상위 class에는 Resource class, AN_Route class, AN_Geom class, AN_Res_Att class가 있다. Resource class는 설비들의 library를 구축하기 위한 class이고, AN_Route class는 moving object가 이동하는 path를 구성하기 위한 class이며, AN_Geom class는 설비의 geometry 정보를

다루기 위한 class이다. 끝으로 AN_Res_Att는 설비의 속성정보를 저장하기 위한 class이다.

다음으로 애니메이션을 수행하기 위한 class에는 애니메이션 모델을 관리하는 Ani_Res class, trace 화일을 읽어 애니메이션 명령어를 parsing 해주는 Ani_TFile class, 그리고 애니메이션 시간을 관리하는 Ani_Time class가 있다.



[그림 3.2] 애니메이터의 Class 계층 구조

3.3 Path type에 따른 Moving Object의 움직임 구현

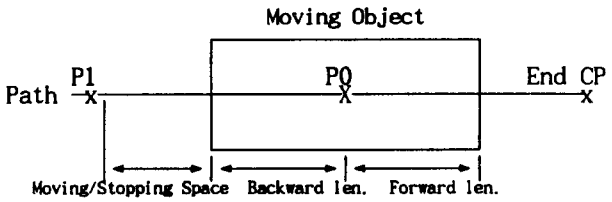
Path segment는 [그림 3.2]에 나타나 있듯이 Path_Seg class에서 path의 종류에 따라 다음과 같이 3개의 class로 파생된다.

- ACC_Path : accumulation type path.
- NACC_Path : non-accum. type path.
- SRM_Path : stacker crane의 path.

ACC_Path, NACC_Path, SRM_Path에는 Path_Seg class에서 선언된 moving object의 움직임을 위한 가상함수(virtual function)들이 각 path의 특성에 따라 구현되어 있다.

1) Accumulation Type Path

Accumulation path는 moving object가 이동할 경우에는 moving object들 사이에 일정한 거리(Moving Space)이상 떨어져 움직이고, moving object가 정지하는 경우에는 뒤따르던 moving object는 정지거리(Stopping Space)를 유지한 채로 쌓이는 path이다.



[그림 3.3] Path segment에서 moving object의 움직임

[그림 3.3]은 accumulation path에서 moving object의 움직임을 보여준다. 먼저 path segment의 끝점에 가장 가까운 moving object의 위치(P0)를 정한 후, 다음 moving object가 놓일 수 있는 한계 위치(P1)를 정한다. 그리고 그 뒤에 오는 moving object는 앞에 있는 moving object의 한계 위치를 넘지 못하는 범위에서 자신의 위치를 정하게 된다.

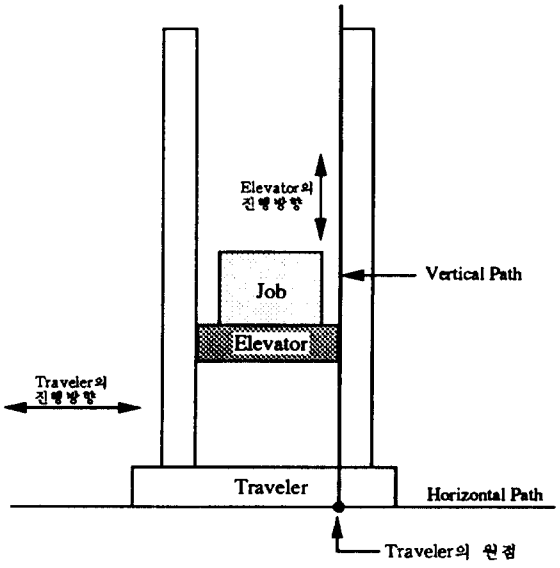
2) Non-accumulation Type Path

Non-accum. path는 moving object들 사이에 거리가 변하지 않는 path를 말한다. 즉, path segment의 선두에 있는 moving object가 정지하면 path segment상의 모든 moving object가 정지하고, 선두에 있는 moving object가 움직이면 path segment상의 모든 moving object가 동시에 움직이게 된다.

Non-accum. path의 움직임은 path segment의 선두에 있는 moving object의 위치를 결정한 후, 선두 moving object가 움직인 거리만큼 뒤에 있는 moving object의 위치를 변화시킨다.

3) Stacker Crane의 움직임

Stacker crane은 아래 [그림 3.4]에 나타나 있듯이 수평 path위를 움직이는 주행장치(traveler)와 수직 path위를 움직이는 승강장치(elevator)로 구성되어 있다. 승강장치를 위한 수직 path는 주행장치에 붙어 있고 작업물의 운반은 승강장치에 의해 이루어진다.



[그림 3.4] Stacker Crane을 위한 Path

애니메이션 수행도중 S/C의 위치는 다음과 같은 순서로 결정된다.

① 절대좌표계 상에 주행장치의 위치를 계산한다.

② 주행장치의 좌표계에서 승강장치의 위치를 계산한다.

③ 승강장치의 위치를 주행장치의 좌표계에서 절대좌표계로 환산한다.

3.4 애니메이션 명령어와 애니메이션 수행 흐름

1) 애니메이션 명령어

Trace 화일을 작성하기 위한 애니메이션 명령어와 trace 화일의 형태는 다음과 같다.

① Animation Command

- Event time 표시 : TIME
- Create/Destroy Job : CREATE, DESTROY
- Moving object의 움직임 표현 : PLACE ON, MOVE TO
- Positioning : PLACE IN, PLACE AT
- 움직임의 종속관계 : ATTACH, DETACH
- 애니메이션 종료 : END

② Trace file format

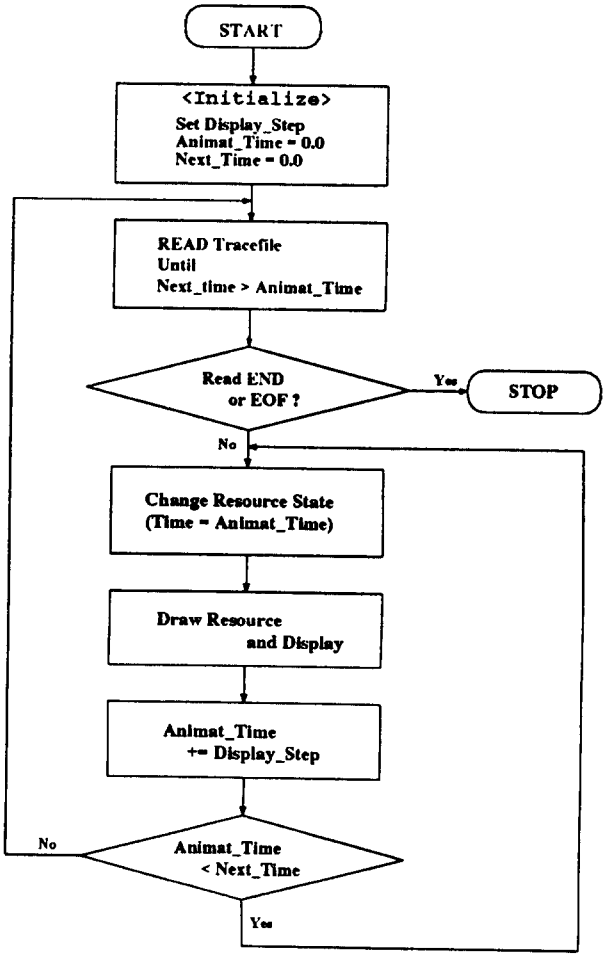
<1> Event 발생 시간을 TIME 명령어를 이용하여 기록한다.

<2> <1>의 시간에 발생할 event를 애니메이션 command를 이용하여 기록한다.

<3> <1>와 <2>을 반복하여 모든 event를 기록한다.

<4> 모든 기록이 끝나면 마지막 줄에 END를 기록한다.

2) 애니메이션 수행 흐름



[그림 3.5] 애니메이션 수행 흐름도

[그림 3.5]에는 애니메이션 수행에 대한 전체적인 흐름이 나타나 있다. 먼저 TIME 명령어의 값(Next_Time)이 현 애니메이션 시간을 나타내는 Animat_Time보다 클 때까지 trace 화일을 읽은 후, 현재 애니메이션 시간에 맞도록 resource의 상태를 변화시킨다. 그리고 변화된 resource의 상태를 화면에 출력하고 Animat_Time을 Display_Step만큼 증가시킨다.

4. AMS 애니메이터를 이용한 그래픽 시뮬레이션 예

본 연구에서 개발된 AMS 애니메이터를 국내에 가장 일반적으로 보급된 자동화 제조·물류 시스템 중에 하나인 자동창고(AS/RS)[1]를 대상으로 그래픽 시뮬레이션 예를 보이고자 한다.

4.1 대상 자동창고의 구성설비와 layout

대상시스템은 6개의 aisle을 가지고 있으며, 한 개의 aisle은 2개의 rack과 수직·수평운동을 하는 물류이송설비인 S/C(Stacker Crane)을 한대씩 가지고 있다. 입고작업과 출고작업을 위하여 입고장 및 출고장이 존재하고 입고장에는 3개의 직선형 roller conveyor와 2개의 diverter로 이루어진 conveyor-net이 형성되어 있으며, 출고장은 3개의 직선형 roller conveyor로 구성되어 있다. 입고장과 출고장 그리고 aisle간의 물류공급을 위해 6대의 대차를 갖는 순환형 RGV(Rail Guided Vehicle) 시스템이 존재한다. 또 RGV와 aisle간의 물류의 완충작용을 위해 conveyor형태의 Inlet buffer와 Outlet buffer가 존재한다.

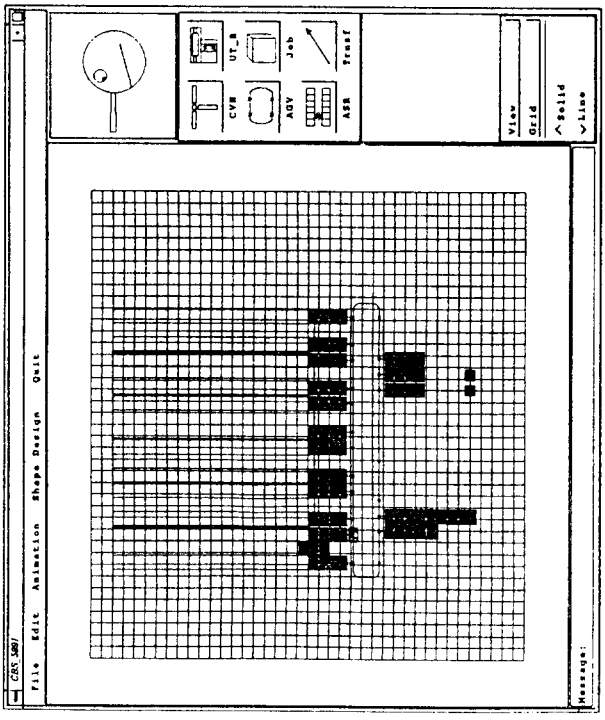
4.2 Layout과 설비의 속성정보 입력

Layout design module의 GUI(Graphic User Interface)를 이용하여 자동창고의 구성설비와 layout 정보를 입력한다. [표 4.1]은 자동창고의 구성설비를 애니메이터에서 제공하는 설비 library로 모델링하기 위한 mapping 관계를 나타낸다.

[표 4.1] 자동창고 구성설비와 애니메이터의 설비 lib.

Station	구 성 설 비	애니메이터의설비 lib.	비 고
입 고 장	Roller conveyor, diverter	Conveyor -net	Accumulation type conveyor로 모델링
출 고 장	Roller conveyor	Conveyor -net	
Conveyance	RGV, rail	AGV-net	
In/outlet buffer	Roller conveyor	Conveyor -net	Accumulation type conveyor로 모델링
Aisle	Rack, S/C	AS/RS	

[표 4.1]의 모든 설비의 입력이 끝나면 물류의 객체인 Job을 입력한다. 그리고 transfer정보를 입력하여 transfer path를 생성시키면 layout 입력이 끝난다. [그림 4.1]은 모든 입력이 끝난 자동창고의 layout을 위에서 바라본 것이다.



[그림 4.1] 입력된 자동창고의 layout

1) 본 자동창고의 예는 과기처 수탁연구과제인 "CAD Based System Simulator(CBSS) 개발"[전차수 '94] 과제에서 시뮬레이션 예로 사용된 자동창고 시스템이다.

4.3 Trace 화일에 의한 애니메이션

시스템의 layout입력이 끝나면 layout 모델에 생성된 path 정보를 "Linked Path" 화일에 출력시켜 시뮬레이션 프로그램에서 trace 화일을 출력하는데 이용한다. Linked path 화일은 각 설비별 path segment와 transfer path의 이름, 시작과 끝점의 좌표, path 길이 정보를 담고 있다.

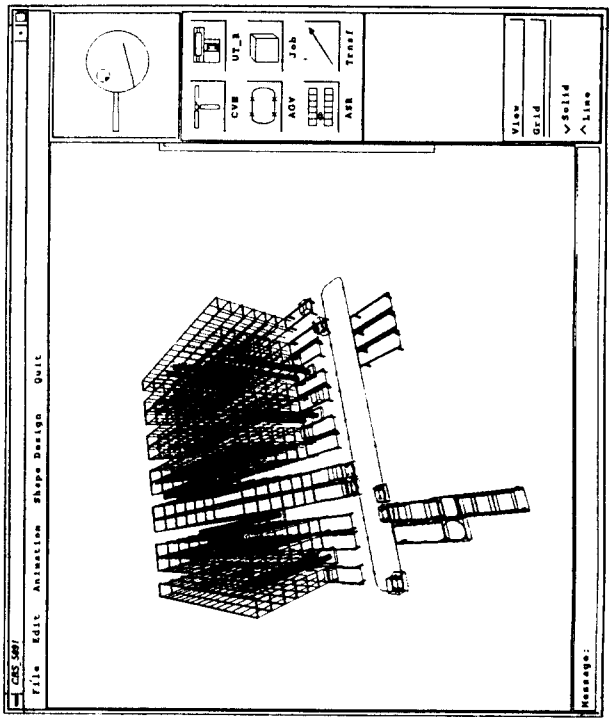
Trace 화일을 출력하기 위한 시뮬레이션모델은 "CAD Based System Simulator"(CBSS)[전차수 '94]의 시뮬레이션 Engine을 이용하여 생성되었다. [그림 4.2]와 [그림 4.3]은 trace 화일을 입력으로 한 애니메이션 상황을 보여준다.

5. 결론 및 추후 연구 과제

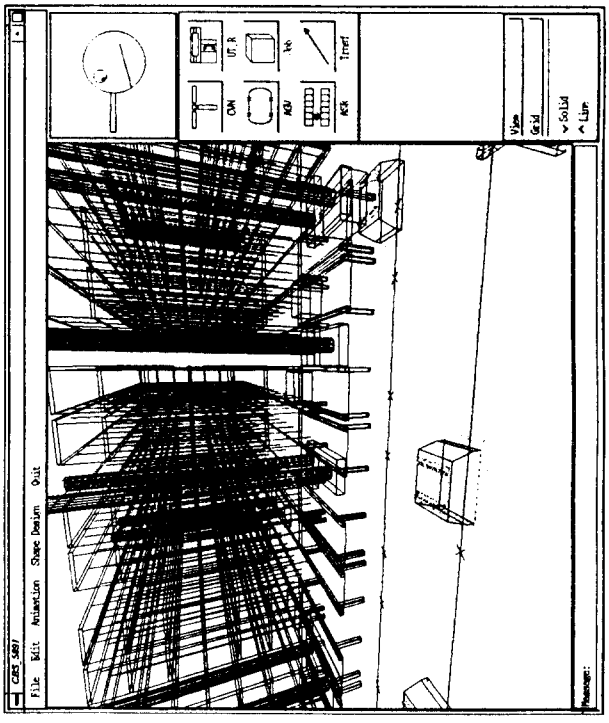
본 연구에서는 애니메이션을 이용하여 시뮬레이션 모델의 검증과 대상 시스템에 대한 사용자의 이해를 돕기 위한 AMS 그래픽 애니메이터 시스템의 구조(architecture)를 제시하였고, GL 그래픽 환경을 갖는 EWS(Engineering Workstation)에서 C++언어를 이용하여 애니메이터를 개발하였다. 그리고 자동창고를 대상으로 개발된 애니메이터를 이용하여 애니메이션을 수행해 봄으로써 자동화 제조시스템을 대상으로 하는 AMS 전용 애니메이터로의 발전 가능성을 제시하였다.

AMS 애니메이터는 자동화 제조 시스템의 layout 생성 기능 및 생성된 layout상에서 시뮬레이션 trace 정보에 의한 3차원 그래픽 애니메이션을 수행하는 기능을 가지고 있다. AMS 애니메이터의 기능을 향상시키기 위한 추후 연구과제는 다음과 같다.

1) 자동화 제조 시스템의 구성 설비들 중 Robot의 움직임이나 Stacker crane의 fork 움직임 같은 기구학적(Kinematics)인 동작을 나타낼 수 있는 연구가 필요하다.
2) 애니메이션 속도를 향상시키는 연구가 필요하다.



[그림 4.2] 자동창고의 애니메이션 화면 1



[그림 4.3] 자동창고의 애니메이션 화면 2

3) 입력된 시스템 layout을 프린트하는 기능과 타 그래픽 소프트웨어와 Neutral File[2]로 그래픽 정보를 교환하는 기능에 관한 연구가 필요하다.

[참 고 문 헌]

[Breugnot, '91] Daniel Breugnot, etc. "GAME: An Object-Oriented Approach to Computer Animation in FMS Modelling", *IEEE Sim. Sym.*, 217-227, 1991.

[Carrie '88] Allen Carrie, *"Simulation of Manufacturing System"*, 139-151, John Wiley &Son, 1988.

[Gibson '93] Gibson Peter and Welgama, Palitha S, "Simulation methodology in facilities design : knowledge from a practical application", IE, Vol. 25, No.9, pp.52-58, 1993.

[Gorlen '90] Keith E. Gorlen, Sanford M. Orlow, Perryt S. Plexico, *"Data Abstraction and Object-oriented Programming in C++"*, John Wiley & Sons, NewYork, NY, 1990.

[Green, '85] Green, R. and Shale, A.J.A, "Simulation integration", *Proc. of the 2nd Int. Conf. Automated Materials Handling* Edited by Hollier, R.H., 15-17 May 1985, Birmingham, UK., pp.125-129.

[Johnson '88] M. Eric Johnson, Jacob P. Poorte, "A hierarchical approach to computer animation in simulation modeling", Simulation 50:1, 30-36, 1988.

[Law '92] Averill M. Law, Michael G. McComas, "How to Select Simulation Software For Manufacturing Applications", IE, July, 29-35, 1992.

[McLendon '91] Patricia McLendon, *"Graphics Library Programming Guide"*, Silicon Graphics, Inc., 1991.

[OSF '91] Open Software Foundation, "OSF/Motif™ Programmer's Reference", Revision 1.1, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[Rembold '93] U. Rembold, B.O. Nnaji, A. Storr, *"Computer Integrated Manufacturing and Engineering"*, Addison-Wesley Pub. Co., 1993.

[Smith '87] Richard L. Smith, Lucille Platt, "Benefits of animation in the simulation of a machining and assembly line", Simulation 48:1, 28-30, 1987.

[Stroustrup '91] B. Stroustrup, *"The C++ Programming Language"*, 2nd ed. , Addison-Wesley Pub. Co., 1991.

[WSC '92] Wolverine Software Corporation, "Using Proof Animation™", 1992.

[전차수 '94] 전차수 외, "CAD-Based System Simulator 개발", 한국소프트웨어개발연구조합, 제2차년도 연차보고서, 1994.

2) IGES(Initial Graphics Exchange Specification : ISO, 미국), DXF(Drawing Interchange File : AutoCAD) 등이 널리 쓰이고 있다.