JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 1 : 119~126 Jan. 2015

항만 컨테이너 시뮬레이션을 위한 대규모 3D 가시화 시스템 개발

옥수열*

A Large-Scale 3D Visualization System for Port Container Terminal Simulation

Soo-yol Ok*

Department of Game Engineering, Tongmyoung University, Busan 608-711, Korea

요 약

본 논문에서는 항만컨테이너 운영 모니터링 및 계획 시뮬레이션의 결과를 3차원으로 정적·동적 개체를 포함하는 주변 환경의 거동을 시각적으로 표현하고 실제 작업이 이루어지는 애니메이션을 통해 운영모델을 직관적으로 평가가 가능한 시뮬레이션 가시화 시스템을 제안한다. 제안한 항만 시뮬레이션 가시화 시스템(PSVS)은 객체의 상태를 XML기반으로 설계하여 기존의 항만 운영 시뮬레이션(CATOS)과 상호운용성에 있어서 연동성을 극대화하고 실시간으로 대규모 객체에 대한 애니메이션 표현이 가능하도록 구현하였다. 본 논문에서는 PSVS시스템의 설계 방법에 대해서 설명하고 실험을 통해 PSVS의 타당성을 검토하였다.

ABSTRACT

In this paper, we propose a 3D visualization system that expresses the simulation results of port container operation monitoring and planning in terms of the movements of the surrounding environment including static and dynamic objects. It also enables us to objectively evaluate the operation model by way of animating the actual port operations. With XML-based design of object states, the port simulator visualization system (PSVS) that we propose has been implemented so as to maximize the interoperability with the existing port operation simulation (CATOS), and express the realtime animation of large-scale objects. The design method of PSVS system is explained, and its validity is experimentally examined.

키워드 : 항만 컨테이너 시뮬레이션, 3D가시화, 대규모 객체 렌더링, XML

Key word : Port Container Terminal Simulation, 3D Visualization, Large-Scale Object Rendering, XML

접수일자 : 2014. 12. 03 심사완료일자 : 2014. 12. 22 게재확정일자 : 2015. 01. 02

* **Corresponding Author** Soo-Yol Ok(E-mail:sooyo@tu.ac.kr, Tel:+82-51-629-1252)
Department of Game Engineering, Tongmyoung University, Busan 608-711, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2015.19.1.119 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근 항만 컨테이너 터미널은 선박의 고속화 및 초대형화 추세에 따라 양적하 작업의 생산성 향상과 운영관리의 효율성 증대를 위해 항만 장비 통합 모니터링 기반의 지능형 컨테이너 터미널 운용시스템의 도입이 요구되고 있다[1]. 그 중에서도 최적 작업계획 설계 및 운영전략을 도출하기 위한 방법으로 항만 시뮬레이션기술은 필수적인 요소 중의 하나이다[2, 3].

반면에 기존 항만 시뮬레이션 방법은 효율적인 컨테이너 운용 방법을 고려해 실제 컨테이너 터미널의 동작 표현이 가능해야 함에도 불구하고 각 객체의 배치 정보, 운송로의 위치 및 배치 정보 등의 각 터미널들의 특징적 정보를 고려하지 못하고 단순한 모델들로만 시뮬레이션 시스템이 설계되어 실제 산업현장의 현실성과는 많은 거리가 있다는 문제점을 가지고 있다[4]. 이와 같은 기존 항만 컨테이너 시뮬레이션 시스템의 문제점을 개선하기 위해서는 항만의 3차원 정적·동적 개체를 포함하는 주변 환경의 거동을 시각적으로 표현하고 실제 작업이 이루어지는 애니메이션을 통해 운영모델을 직관적으로 평가가 가능한 시뮬레이션 가시화 시스템이 매우 중요하다.

이에 본 논문에서는 기존 2D기반의 항만 터미널 운영 시스템인 TSB사의 CATOS(Computed Automated Terminal Operating System)엔진[5]과 네트워크 통신을 통하여 실시간 모니터링과 함께 직관적인 시뮬레이션 결과를 가시화 시켜주는 항만 시뮬레이션 가시화 시스템PSVS(Port Simulator Visualization System: 이하 PSVS)을 제안하고자 한다. 제안한 PSVS시스템은 객체의 상태를 XML기반으로 설계하여 CATOS와 상호운용성에 있어서 연동성을 극대화하고 GPU버퍼관리기법을 적용하여 실시간으로 대규모 객체에 대한 애니메이션 표현이 가능하도록 하였다. 본 논문에서는 기존의 항만 운영시스템과 연동하여 멀티뷰를 통해 현재 상태에 다양한 시점에서 효율적인 모니터링이 가능하고 또한 사전 운영 시뮬레이션 결과를 실시간으로 3차원 애니메이션이 결합한 가시화를 통해 직관적으로 평가할 수 있는 PSVS시스템에 대해서 설명하고자 한다. 본 논문은 다음과 같이 구성되어 있다. II장은 관련연구로 항만 컨테이너 터미널 시뮬레이션 가시화기술에 대해 살펴보고, III장에서는 제안한 PSVS를 설명한다. 그리고 IV장에서는 PSVS에 대한 실험과 결과분석을 통해 제안 시스템을 검증하고, V장에서는 결론과 추후과제를 제시한다.

Ⅱ. 관련 연구

항만 시뮬레이션 가시화 관련 연구로서 Tasi[6] 등은 항만 운영 시뮬레이션 결과를 3차원 객체들의 폴리곤을 줄이는 저수준 디테일 방법을 이용한 대규모의 객체들을 렌더링 할 수 있는 기법을 제안하였다. 이러한 방법은 저품질 모델들로 대규모의 장면을 연출하거나 렌더링 할 수 있는 장점이 있지만 사실적인 묘사나 대규모의 객체를 표현하는 데 있어서는 문제점을 내포하고 있다. Bijl[7] 등은 게임 그래픽기술을 활용해 항만 터미널 운영 시뮬레이터와 연동한 효율적인 관제 시스템 및 시뮬레이션 가시화 방식을 제안하였다. 이 방식은 최신 게임그래픽엔진기술을 이용하여 사실감 이 뛰어난 묘사가 가능하지만 상황에 따라 다양한 장면에서 여러 각도로 볼 수 있게 하는 멀티뷰 시점에서는 오히려 그 알고리즘이 더 비효율적인 결과를 초래할 수 있다는 단점이 존재한다.

기존의 항만 터미널 운영 시뮬레이션 가시화 상용 툴로서는 Arena[8], FrexSim[9], Simio[10] 등이 있다. 이들 상용엔진의 경우는 고가의 라이선스 정책과 제한적인 API에 따른 최적화 및 상세함 등을 표현하는 데는 한계가 있고 또한 시뮬레이터 시스템의 유지보수에 상당한 비용이 소모되는 등의 걸림돌이 있다는 단점이 있다[11,12]. 또한 항만 운영 시뮬레이션 가시화 시스템에서는 운영 시뮬레이션 모델에서 나온 결과를 3차원 애니메이션과 다양한 측면에서의 관찰이 가능한 멀티뷰 기능 등의 현실감 있는 직관적인 뷰를 제공 가능한 가시화 시스템이 요구되고 있지만 현재 일반적으로 사용하고 있는 3차원 그래픽 렌더링 엔진은 실제 항만을 그대로 묘사하기 위한 대규모의 객체들을 효율적으로 렌더링, 애니메이션하기에는 한계를 나타내고 있다.

Ⅲ. CATOS와 PSVS간의 통신 프로토콜

기존의 CATOS는 컨테이너 터미널 운영 모니터링 및 성능 분석 시뮬레이션 결과를 3D 가시화 처리를 할 경우 대규모의 객체 렌더링에 따른 프레임 저하로 인하여 실시간으로 처리하기 어려움이 있다.

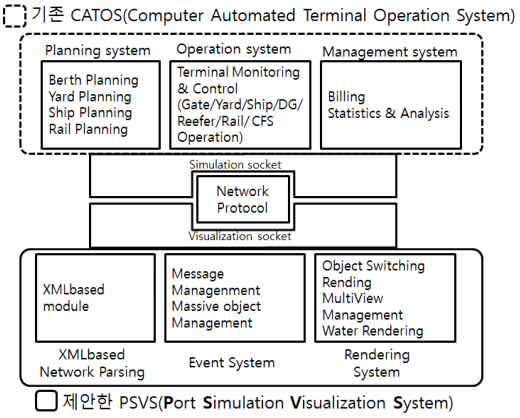


그림 1. CATOS와 PSVS와의 상호 연동 네트워크 구조
Fig. 1 Interoperable network architecture for CATOS and PSVS

이에 본 논문에서는 가시화 성능향성을 위해서 네트워크 모듈을 통해 터미널 운영 시뮬레이션 모듈 부분과 가시화 모듈 부분을 분리하는 가시화 시스템을 제안한다. 제안한 PSVS는 그림 1과 같이 항만 운영에 대한 모니터링과 시뮬레이션을 할 수 있는 CATOS시스템과 연동하여 컨테이너 운영상황 및 시뮬레이션 결과를 네트워크를 통해서 3D 애니메이션으로 가시화 할 수 있는 시스템으로 구성된다.

제안한 항만 컨테이너 가시화 시스템은 CATOS을 통해 얻어진 항만 운영 시뮬레이션 결과를 신뢰성 높은 네트워크 통신 모듈을 사용하여 XML 구조로 스트림화하여 PSVS에 전달하도록 하였다. PSVS는 CATOS시스템에서 스트림으로 전송된 메시지 명령어 태그를 처리하기 위해 크게 3가지 처리모듈로 구성되어져있다. 먼저 객체관리 모듈은 선박, 장치장 장비, 컨테이너, 트럭등과 같은 항만 컨테이너 터미널 구성 객체를 가시화하는 모듈이다. 가시화 뷰 관리모듈은 다양한 각도에서 시뮬레이션결과를 평가할 수 있도록 멀티뷰로 사용자가 주목하고자 하는 부분에 대한 가시화 뷰를 추가하거나 가시화 뷰를 일시정지시켰다가 다시 진행 시키는 것과 같은 장면을 제공할 수 있는 뷰 관리모듈이다. 마지막으로 액션관리모듈은 각 객체에 3D 애니메이션 명령을 할 수 있는 모듈로 액션 명령은 각 객체에게 구분하여 메시지를 전달하고, 해당 이벤트 의 결과는 확인하여 다시 CATOS에게 전달한다. 그림 2는 CATOS와 PSVS간의 메시지 흐름을 나타내고 있다.

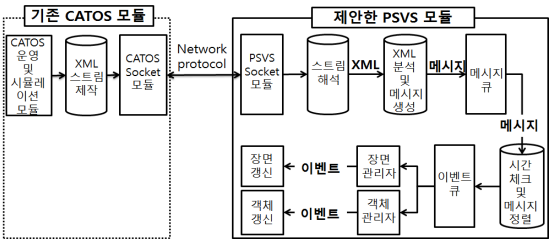
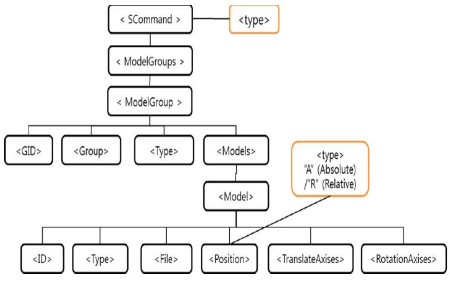


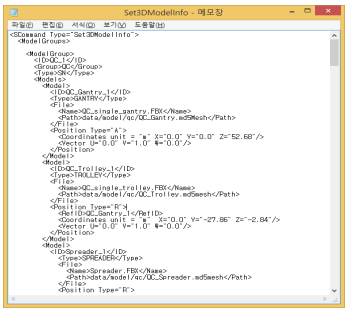
그림 2. CATOS와 PSVS모듈간의 메시지 흐름
Fig. 2 Message sequence between CATOS and PSVS

3.1. PSVS의 구조

본 논문에서는 CATOS에서 가시화에 필요한 가시화 뷰관련 명령, 객체 생성 및 삭제 등의 객체관련명령, 그리고 각 객체들의 행동을 지시하는 액션관련 명령을 XML태크로 구조화하여 PSVS와 메시지 통신하도록 프로토콜을 설계하였다. 먼저 객체관리 모듈은 객체 관리 XML명령어를 처리하는 모듈로서 주로 객체 생성 및 삭제 그리고 객체들을 드로잉하거나 이벤트를 처리한다.



(a)



(b)

그림 3. PSVS의 객체 구성 요소 XML 설계 (a) 제안한 3D 객체 구성 XML 구조 (b) 3D 객체 구성 XML예
Fig. 3 XML design of object elements for PSVS (a) The XML structure of the proposed 3D objects (b) An XML example of constructing 3D objects

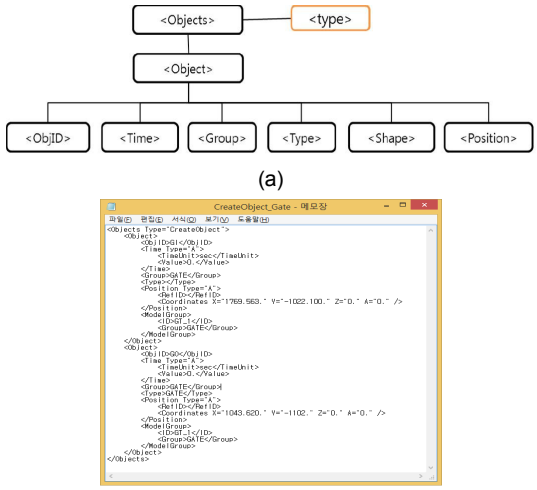


그림 4. PSVS의 객체 생성 XML 설계 (a) 제안한 객체 생성 XML 구조 (b) 게이트 객체 생성 XML 예
Fig. 4 XML design of 3D object creation in PSVS (a) The XML structure for generating the proposed 3D objects (b) An example of generating a gate object

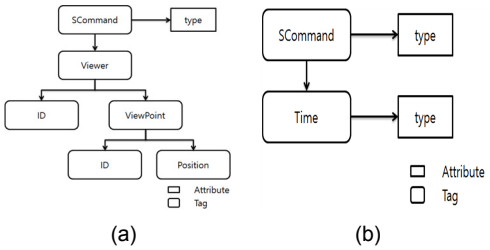


그림 5. 제안한 가시화 화면 관리 XML구조 (a) 생성뷰어 (b) 스타트 뷰어, 클로저 뷰어
Fig. 5 XML structure for visualization scene management (a) CreateViewer (b)StartViewer, CloseViewer

객체관리 모듈은 시뮬레이션의 객체 구성요소가 되는 컨테이너, 크레인 등 여러 객체들을 나타내기 위해 그림 3(a), 그림 4(a)와 같은 3D객체 구성요소 XML과 이들 객체 구성요소에 따라 객체 생성 XML 파일을 통해 얻은 정보로 가상 3D공간상에 생성하고 관리 한다. 이 모듈은 객체 생성관리를 위한 CreateObject클래스와 객체들을 드로잉 하는 DrawObject 클래스로 구성되어 있다. 그림 3(b)와 그림 4(b)는 3D 객체 구성 XML 예와 게이트 객체 생성 XML 예이다. 가시화 뷰 관리 모듈은 가시화 뷰 XML명령어 처리 모듈로서 CATOS와 PSVS간의 가시화 뷰에 관련되는 메시지를 분석해 가시화 뷰를 생성시키는 Create Viewer 와 가시화 뷰를 정지시키는 StopViewer, 그리고 정지된 가시화 뷰를 다시 재생하는 StartViewer명령어를 처리한다.

CreateViewer는 그림 5와 같이 가시화 뷰 생성에 관련된 속성 등을 설정하는 명령어들이 포함되어 있다. 그리고 StartViewer 와 CloseViewer는 시간 설정에 따른 가시화 뷰 재생을 위한 시작과 끝을 처리한다.

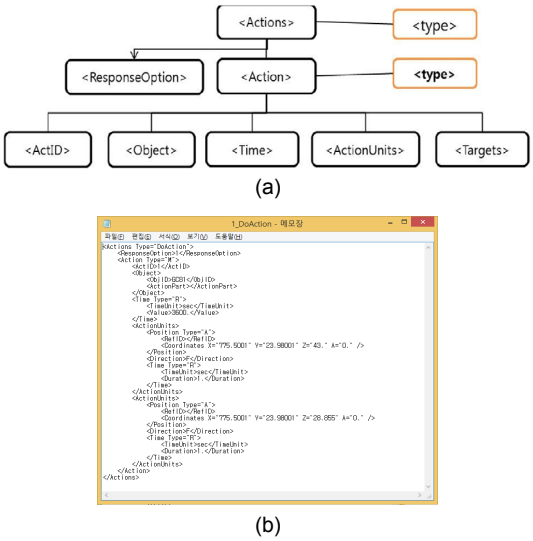


그림 6. 동적 객체의 움직임을 위한 Action XML 구조 및 예 (a) 제안한 액션 XML 구조 (b) 액션 XML의 예
Fig. 6 Action XML structure and example for movements of dynamic objects (a) An XML structure of the proposed Action (b) An example of action XML

항만 객체의 애니메이션 표현하기 위한 액션관련 XML명령어는 주로 컨테이너 터미널에서 일어나는 양적하 작업의 움직임을 애니메이션으로 표현하기 위한 명령어들로 구성되어 있다. 특히, 이들 명령어들은 그림 6과 같이 야드 크레인과 야드 트레일러들에 대한 작업준비공정과 처리공정을 정의하기 위한 것으로 컨테이너를 집어 올리기 위해 야드 크레인의 이동이나 상차작업 그리고 야드 트레일러들의 이동 등에 대한 액션의 타입 그리고 GET(집어올리기) 이나 PUT(상차)할 시 어떠한 컨테이너 객체를 집을지 또한 어떠한 객체 위에 놓을지 를 정하는 타겟 컨테이너 객체의 ID 등으로 구성이 된다. 제안한 PSVS는 CATOS엔진간의 통신을 위하여 XML 파일 구조를 스트림화하여 통신하

도록 설계하였다. CATOS는 실시간 관제 시스템의 상황이나 시뮬레이션을 XML스트림으로 처리하기에는 관제 시스템에 부담이 된다. 이에, 시뮬레이션 경우는 결과를 PSVS에 주거나 미리 예측되는 메시지가 있으면 해당시간에 실행될 메시지를 제작하여 보내도록 하였다. 이는 메시지에 따라서 메시지를 받는 즉시 그 메시지를 실행해야 할 수도 있지만 그렇지 않을 수도 있기 때문에 메시지를 구분하여 수행하여도 함으로서 효율성을 높이도록 하였다. 이렇게 해서 만들어진 해당 메시지는 구조체 형식으로 CATOS와 PSVS간을 연동하게 된다. 메시지 구조체는 XML태그들을 모아놓은 Msg와 해당 메시지를 실행할 Time으로 구성되어 메시지 큐 형태로 저장된다. Msg는 XML의 태그들과 속성들을 분석하여 그것들을 간단한 리스트구조로 가지고 있다. 그리고 Time은 메시지의 실행될 시간을 가지고 있는 값으로 정의한다.

PSVS은 CATOS부터의 메시지들이 대기 없이 바로 실행되는 경우도 있지만 대기상태에서 일정 시간 후에 실행되는 메시지들도 있기 때문에 메시지의 실행 시간에 따라 정렬하여 메시지 규로 관리한다. 또한 한꺼번에 여러 개의 메시지가 도착하거나 메시지를 처리하는 도중 새로운 메시지가 추가 되는 경우 동기화 문제가 발생한다. 이러한 문제점을 해결하기 위해 이벤트 큐를 두어 관리하도록 하였다. 그림 7은 메시지와 이벤트의 처리흐름을 나타내고 있다.

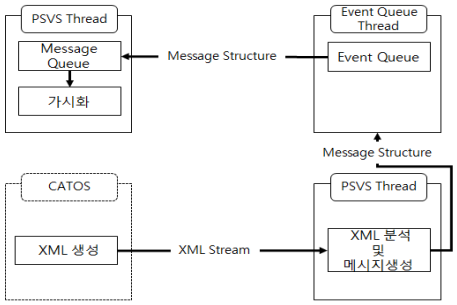


그림 7. 제안한 PSVS의 이벤트 처리 흐름
Fig. 7 Event processing flow of PSVS

3.2. 대규모 항만 객체 실시간 렌더링

항만 터미널 시뮬레이션 가시화 시스템에서 대규모 항만 개체들의 움직임을 실시간으로 가시화하는 것이 중요한 고려사항 중의 하나이다[13]. 최근의 컴퓨터 그래픽스기법에서 대규모 개체의 렌더링 할 때 사용되는 최적화 기법으로는 LOD나 절두체 컬링, Quad Tree 컬링 등 다양한 기법이 있다. 하지만 이러한 기법들은 PSVS에서 가시화 뷰가 늘어나면 계산량이 급격히 증가하는 문제가 있다. 이에 본 논문에서는 크레인이나 야드 트레일러, 컨테이너등과 같은 동적인 객체들의 효율적인 렌더링 기법을 통해 다수의 가시화창에서도 대규모 개체에 대한 실시간 렌더링 방법을 제안한다. 대규모 개체 렌더링의 경우에 GPU상의 N개의 객체 정점 버퍼를 프레임마다 N번 열었다가 닫는 기존 방법은 객체 개수가 늘어나면 날수록 많은 렌더링 시간이 요구된다. 이를 개선하기 위해 먼저 동적 개체가 움직일 때 와 목표지점에 도착할 때에 버퍼를 열고 닫음으로 정점버퍼를 열고 닫는 횟수를 줄임으로서 렌더링 시간을 단축할 수 있다. 즉, 대규모 객체를 그림 8과 같이 하나의 단일 정점 버퍼에 넣어서 한 번에 렌더링 함으로서 한 번에 그리는 함수 호출로 대규모의 객체를 실시간으로 렌더링 하는 것이 가능하다.

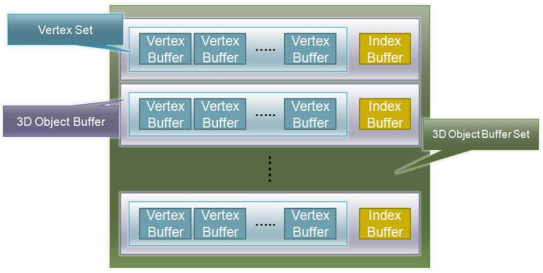


그림 8. GPU상의 대규모 객체 처리 자료 구조
Fig. 8 Data structure of large-Scale object processing on GPU

❶ 단일 정적 정점버퍼에 있는 객체 중 동적 이벤트 발생한 정적 객체를 임의의 비가시화영역으로 이동한다.
❷ 임의 비가시화영역으로 이동시킨 정적 객체를 대체할 신규 동적 객체 정점버퍼 생성한다.
❸ 단일 정적 정점버퍼에 있는 객체를 먼저 1차 렌더링 그리고 동적 객체로 전환한 객체를 2차적 렌더링 한다.
❹ 이벤트가 있을 경우 ❷ ❸를 반복하고 이벤트가 완료하면 동적 객체는 삭제하고 그 위치에 비가시화 영역에 있는 정적 객체로 전환 후 단일 정적 정점버퍼를 유지한다.

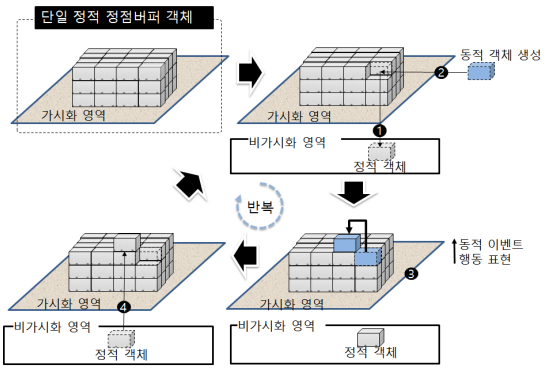


그림 9. 객체 스위치 렌더링 알고리즘
Fig. 9 Algorithm for object switch Rendering

단, 여기서 객체가 동적으로 움직일 때 매 프레임 마다 정점버퍼를 열어서 정점을 수정한 후 다시 닫아야한다는 문제점을 가지고 있다. 이를 개선하기 위해 객체 스위칭 렌더링 기법을 적용하였다. 항만컨테이너터미널을 구성 하는 객체는 동적으로 움직이는 동적 객체보다는 정적 개체수가 훨씬 많다. 이러한 측면을 고려해서 정적 객체와 동적객체를 구분하여 이를 스위칭 하여 렌더링 함으로서 성능을 개선하였다. 그림 9는 객체 스위치렌더링 기법의 처리 흐름을 나타내고 있다. 아래의 표 1은 객체 스위치렌더링 기법의 의사코드를 나타내고 있다.

표 1. 대규모 객체 렌더링 성능 평가표
Table. 1 Pseudocode of object switch Rendering

```
void function initBuffer  //정점버퍼 초기화 모듈
int totalVertex = 0
for (i = 1; i<=objectCount; i++) {
  for(j = 1; j<=object.VertexCount; j++) {
    staticBuffer[totalVertex + j - 1] =object.[i-1]Vertex
  }
 totalVertex += object[i-1].VertexCount
}
```

```
void function startAnimate // 동적 객체 이벤트 처리 시작 모듈
createDynamicObject // 동적 객체 생성
for(i = 1; i<= object.VertexCount ;i++){
   int objectPoint = object.id *object.VertexCount;
   move(staticBuffer[objectpoint])
}
```

```
void function endAnimated // 동적객체 이벤트처리 종료 모듈
for(i = 1; i<= object.VertexCount ;i++){
 int objectPoint = object.id *object.VertexCount;
 move(staticBuffer[objectpoint], getDynamicObjectPosition)
}
deleteDynamicObject  //동적 객체 삭제
```

객체 스위칭 렌더링 기법은 객체가 동적으로 움직일 때와 처음 버퍼를 열 때 해당 객체를 뷰포인트에서 전혀 보이지 않는 임의의 비가시화영역으로 보내고 임의의 비가시화 영역으로 이동시킨 후 이 정적 객체를 대체할 신규 동적 객체 정점버퍼를 생성한다. 그러고 난 후 단일 정적 정점버퍼 내에 있는 객체를 먼저 1차 렌더링 하고, 다음으로 동적 객체로 전환한 객체를 2차 렌더링 하여 움직임을 표현한다. 마지막으로 이벤트가 완료한 동적 객체는 삭제하고 그 위치에 비가시화 영역에 있는 정적 객체로 전환하여 단일 정적 정점버퍼를 유지하도록 한다. 제안한 객체 스위칭 렌더링 기법은 정적 객체별로 매 프레임 정점버퍼를 열고 닫을 필요 없이 한 번에 렌더링하고 나머지 동적 객체에 대한 정점 버퍼만을 별도로 렌더링 함으로써 PSVS엔진에서 대규모 객체를 실시간 처리 할 수 있다.

Ⅳ. 실험 결과

본 논문에서 제안한 PSVS시스템은 MS사의 window 7 64bit 운영체제를 사용하는 intel 3.47GHz i7 CPU와 24GDDR3 RAM , NVidia GTX 590 GPU 환경에서 해서 구현하여 실험하였다. 먼저 대규모 객체 실시간 렌더링 성능평가를 위해 PSVS에서 컨테이너 객체를 단계별로 증가시켜 최대 30만개 까지 했을 때 아래 표와 같은 결과를 얻었다. 사용된 객체 모델은 정점 8개로 이루어진 간단한 컨테이너 박스 객체이다

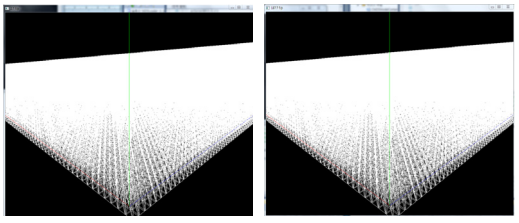
표 2. 대규모 객체 렌더링 성능 평가표
Table. 2 Performance evaluation table of large-scale object rendering

방식 / 객체수	개별 정점버퍼 렌더링 방식	단일 정점버퍼 렌더링 방식
100	710	860
1,000	258	827
5,000	63	732
10,000	31	657
50,000	6	397
100,000	3	250
3000,000	1	180

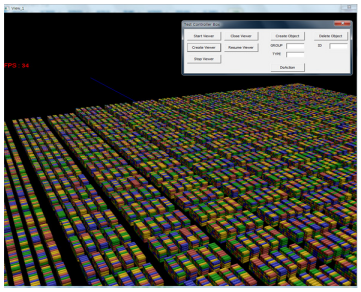
표2와 같이 개별 정점 버퍼 방식은 렌더링 하는 객체가 늘어남에 따라 프레임이 현저히 낮아지는 것을 볼

수 있다. 하지만 단일 정점 버퍼 방식은 객체가 늘어남에 따라 프레임이 낮아지는 것은 개별 버퍼 방식과 동일하지만 낮아지는 비율이 낮다는 것을 알 수 있다. 이 실험결과로 기존의 개별 정점 버퍼 방식 보다 제안한 단일 정점 버퍼방식으로 한 번에 렌더링 했을 경우가 약 60배 이상 성능향상이 있다는 것을 확인 할 수 있다.

그림 10은 실험결과로 그림 10 (a)는 개별 정점 버퍼 렌더링 방식 결과 이미지, 그림 10(b)는 단일 정점 버퍼 렌더링 방식 결과 이미지, 그림 10(c)는 텍스쳐 맵핑한 단일 정점버퍼 렌더링 처리 결과이다. 특히, 그림 10(c)의 경우는 텍스쳐 맵핑한 단일 정점 버퍼 렌더링 결과로 30fps이상으로 실시간 렌더링 성능이 나오는 것이 확인 할 수 있다.



(a) (b)



(c)

그림 10. 대규모 객체 렌더링 결과 비교 (a) 개별 정점 버퍼 렌더링결과(30 만개 컨테이너 객체) (b) 단일 정점 버퍼 렌더링결과(30 만개 컨테이너 객체) (c) 텍스쳐 맵핑한 단일 정점 버퍼 렌더링 결과(30 만개 컨테이너 객체)
Fig. 10 Comparative results of large-scale object rendering (a) Result of individual vertex buffer rendering (Object with 300,000 containers) (b) Result of single vertex buffer rendering (Object with 300,000 containers) (c) Rendering result of Texture-mapped single vertex buffer

그리고 그림 11은 CATOS와 연동해서 항만 시뮬레이션 가시화를 수행한 결과의 모습이다. 좌측 상단에 위치한 명령어 뷰는 CATOS와 XML스트리밍통신을 제어하는 메인이다. 바로 아래의 콘솔화면은 액션명령에 대한 이벤트 처리이다. 그리고 좌측하단의 명령어 제어화면은 가시화 뷰 제어를 위한 컨트롤 박스이다. 가운데 및 우측화면은 CATOS와 연동해서 가시화 시뮬레이션을 결과를 보여 주고 있는 화면으로 실시간으로 멀티뷰로 결과를 실시간으로 확인할 수 있다.

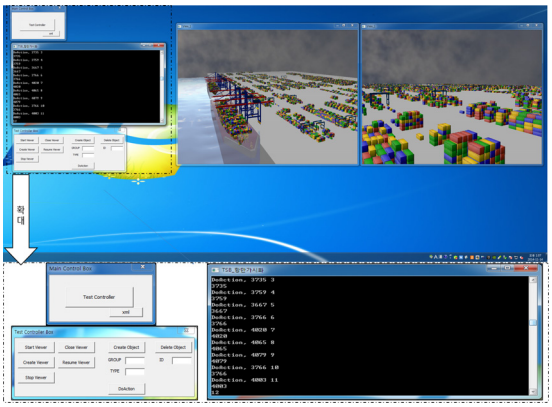


그림 11. 제안한 PSVS의 시뮬레이션 결과 모습
Fig. 11 Results of the proposed PSVS simulation

Ⅴ. 결 론

본 논문에서는 항만 터미널 운영 시스템인 TSB사의 CATOS엔진과 네트워크 통신을 통하여 실시간 모니터링과 함께 직관적인 시뮬레이션 결과를 가시화 시켜주는 항만 시뮬레이션 가시화 시스템 PSVS을 제안하였다.

제안한 PSVS시스템은 객체의 상태를 XML기반으로 설계하여 CATOS와 상호운용성에 있어서 연동성을 극대화하고 실시간으로 대규모 객체에 대한 애니메이션 표현이 가능하도록 구현하였다. 특히, 종래의 2D기반의 항만 컨테이너 시뮬레이션의 가시화 시스템에 대한 문제점을 개선하여 항만의 3차원 정적·동적 개체를 포함하는 주변 환경의 거동을 멀티뷰를 통해 현재 상태에 대한 다양한 시점에서 효율적인 모니터링이 가능하고 또한 사전 운영 시뮬레이션 결과를 실시간으로 3차원 애니메이션이 결합한 가시화를 운영모델을 직관적으로 평가가 가능하다. 또한 본 논문에서는 대규모 항만 객체를 효율적으로 렌더링 하는 기법을 통해 실시간

으로 가시화하였다. 이는 대규모의 객체 렌더링이 요구되는 영상분야에서 충분히 활용될 수 있을 것으로 기대된다. 향후 ,본 논문에서 제안한 PSVS를 현장적용 평가와 함께 보다 사실적인 가시화 및 인터페이스 개선을 통해 PSVS의 활용성을 높이고자 한다.

감사의 글

본 연구는 2014년도 부산광역시 Busan Brain 21(BB21) 사업의 지원에 의하여 이루어진 연구로서, 부산광역시에 감사드립니다.

REFERENCES

[1] K. C. Nam, J. H. Lee, " A theoretical Review on mega ship and mega hub," *Journal of Korean Navigation and Port Research,* vol. 26, no. 79, pp. 455-463, 2002.

[2] U. Clausen, J. Kaffka, "CONTSIM – container terminal management with simulation," *Procedia - Social and Behavioral Sciences*, vol. 54, no. 4, pp. 332 – 340, Oct. 2012.

[3] W. Y. Yun, C. G. Ahn, Y. S. Choi, K. H. Kim ,"An evaluation of the operation plans for port container terminal Using simulation," *Journal of the Korea Society for Simulation*, vol. 7, no. 2, pp. 91-104, Dec. 1998.

[4] J. H. Seo, Y. J. Lee, Y. B. Kim, K. S. Lee, "Development of 3D virtual simulator for performance evaluation in port container terminal," *in Proc. of the 10th Korea Automatic Control Conference,* pp. 421-426, Oct. 2005.

[5] TSB CATOS-Computer Automated Terminal Operating System [Internet]. Available: http://totalsoft.en.ec21.com/CATOS-Computer_Automated_Terminal_Operating_System--7633719_7633970.html

[6] Tsai, Yichang., Ai, Chengbo., Wu, Yichinig., and Siplon, "Simulating port logistics operations using 3D visualization technology", *Proceedings of the International Conference on Computing in Civil and Building Engineering (ICCCBE)*, Nottingham, UK, June, 2010.

[7] J. L. Bijl and C. A. Boer, "Advanced 3D visualization for simulation using game technology," *In Proceedings of the 2011 Winter Simulation Conference*, Arizona, pp. 2810-2821, 2011.

[8] Arena Port and Terminal Simulation Software [Internet]. Available: https://www.arenasimulation.com/industry-solutions/port-terminal-simulation-software

[9] FlexSim Container Terminal Simulation [Internet]. Available: https://www.flexsim.com/flexsim-ct/

[10] Simio Port Simulation Software [Internet]. Available: http://www.simio.com/applications/port-simulation-software/ port-simulation-software.php

[11] D. H. Yeun, Y. S. Choi, "A study on productivity improvement of container terminals," *Journal of Industrial Economics and Business,* vol. 25, no. 5, pp. 2983-2998, 2012.

[12] H. G. Park, "The data envelopment analysis of container terminals to transshipment cargo," *Journal of Korea Port Economic Association*, vol 26, no. 1, pp. 1-19, 2010.

[13] D. G. Aliaga, J. Cohen, A. Wilson, E. Baker, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Sturzlinger, R. Bastos, M. C. Whitton, F. Brooks, and D. Manocha. "MMR: an interactive massive model rendering system using geometric and image-based acceleration," *ACM Symposium on Interactive 3D Graphics*, New York: NY, pp. 199 – 206, 1999.

옥수열(Soo-Yol Ok)

1994년 2월 : 동아대학교 산업공학과 (공학사)
1998년 3월 : 일본 Tsukuba대학교 이공학연구과 (공학석사)
2001년 8월 : 일본 Tsukuba대학교 공학연구과 (공학박사)
2001년 8월 ~ 2004년 1월 일본 통신종합연구소(CRL) 연구원
2004년 3월 ~ 현재 동명대학교 게임공학과 부교수
※관심분야 : 게임 인터페이스, 게임 인공지능, 컴퓨터 그래픽스, 가상현실, GPU병렬처리