가상환경을 위한 3차원 협동 저작 도구의 설계†

최명아[0], 최은정, 유지현, 차지은, 엄성용, 김명주, 이병걸

서울여자대학교 컴퓨터학과
{machoi[0], chej, jhyu, sharonj, osy, mjkim, byungl}@swu.ac.kr

A Design of 3D Cooperative Authoring Tool for Virtual Environment

Myung-Ah Choi[0], Eun-Jung Choi, Ji Hyun Yu, Ji-Eun Cha, Seong-Yong Ohm, Myuhng-Ju Kim, Byong Gul Lee
Department of Computer Science & Engineering, Seoul Women's University

요 약

본 논문에서는 가상환경을 위한 3차원 협동 저작도구를 제안한다. 이 시스템은 Client-Server 구조로서 Java 분산 어플리케이션 모델인 3-tier 모델을 이용하였고 3차원 장면 표현을 위해 자료구조는 Java 3D의 Scene Graph를 이용한다. 그리고 작업자의 역할 구분과 Client에서 공동작업물의 저장 기능을 부여하고 공동 작업의 결과를 Client에게 선택적으로 보여지게 함으로써 Client-Server 구조와 Pessimistic Locking의 단점을 보완한다.

1. 서론

세계적인 시장조사 회사인 Frost & Sullivan이 1998년에 발표한 자료에 따르면 3차원 소프트웨어 시장은 약 26%의 성장률을 기록하였다.[1] 시장의 증가는 기업 간 경쟁에서 살아남으려는 다양한 시도를 낳았으며 이러한 시도 중에는 3차원 설계와 제조분야의 분산적이고 협력적인 접근 또한 포함되어 있다. 이러한 분산적이고 협력적인 접근과 3차원 데이터와 같은 정보의 대용량화는 네트워크를 기반으로 한 협동작업 시스템에 대한 연구를 활성화시키고 있다. 초기에는 동일한 환경의 시스템에서만 적용되던 협동 시스템의 개발이 CORBA와 같은 분산 객체 기술의 발전과 플랫폼에 영향을 받지 않는 Java 언어의 개발 등에 힘입어 이기종간의 협동 시스템 개발로 변화되고 있다.

본 논문은 협동 작업에 참여하는 작업자의 역할 구분과 Client에 의한 선택적인 공동 작업 결과의 저장 및 열람을 가능하게 하여 기존 협동 시스템이 사용하는 Client-Sever 구조와 Pessimistic Locking의 문제점을 보완한다. 또한 Java 3D의 Scene Graph를 이용하여 장면을 표현함으로써 실시간 장면 정보의 변화를 빠르고 쉽게 반영하였고 플랫폼에 독립적인 협동 시스템을 설계한다.

본 논문의 구성은 다음과 같다. 2장 관련 연구에서는 동시성 제어 방법과 Java 3D에 대해 알아보고 기존 시스템을 분석한다. 3장에서는 설계한 시스템의 전체적인 구조를 설명하고 세부적인 설계 내용을 소개한다. 4장 결론 및 향후 연구 방향에서는 설계한 시스템의 특징을 정리하고 앞으로의 연구방향을 제시한다.

2. 관련 연구

2.1 동시성 제어 방법

기존의 동시성 제어 방법은 크게 *Pessimistic Locking* 방법과 *Optimistic Locking* 방법 그리고 예측 방법으로 나누어진다. Pessimistic Locking 방법은 사용자가 데이터를 변경하기 전에 Lock을 요청한다. 사용자는 Lock을 받을 때까지 기다렸다가 Lock 요청에 대한 응답이 오면, 데이터를 변경 할 수 있다. Optimistic Locking 방법은 Lock에 대한 응답을 기다리지 않고, 요청 후 즉시 데이터를 수정하는 것을 허용한다. 데이터를 수정 중 또는 수정 후에 Lock 허가 메시지를 받으면 계속 진행하고, 다른 사람과 충돌이 일어나서 Lock 부인 메시지를 받으면, 데이터 수정 전의 상태로 복귀해야 한다.

예측 방법은 Pessimistic Locking 방법과 Optimistic Locking 방법의 장점만을 취한 방법이다. 객체와 상호 작용 할 사용자를 예측하여 그 사용자가 데이터를 수정하기 전에 미리 Lock 을 주는 방법이다.

기존의 3차원 가상 환경 시스템들은 대부분 동시성 제어 방법으로 Pessimistic Locking 방법을 사용하고, 소수의 시스템만이 Optimistic Locking 방법 또는 예측 방법을 사용하였다.

2.2 기존 시스템의 분석

DIVE[2], Virtual Society[3]와 CAVERNsoft[4]는 Pessimistic Locking 방법을 사용한다. 이 시스템들의 공통적인 문제점은 Pessimistic Locking의 단점인 작업자가 작업을 위해 Lock을 기다려야 하기 때문에 가상환경 참여자들의 병행 작업이 그만큼 감소한다는 것이다. 또한 기존 시스템이 가진 다른 문제점으로는 공동작업에 참여하는 다른 작업자들의 관찰을 위한 사용자 인식과 사용자 그룹에 대한 정의가 모호하다는 것이다. 이러한 문제점은 BSCW[5]와 SPLINE[6]에서 잘 나타난다.

CAVERNsoft[4], SPLINE[6], BrickNet[7], dvs/DVS[8], Cooperate ARCADE[9], Shastra[10]는 Client-Server 구조를 사용한 시스템들이다. 이러한 Client-Server 구조의 문제점은 Server에 부하가 많이 걸리므로 Server의 병목 현상이 일어날 수 있고, 모든 참여자가 Server에 연결을 하고 있어야 하므로 Server의 네트워 자원에 한계가 있고, 따라서 다수의 사용자를 지원하는 데는 한계가 있다. Server가 다운 될 경우 전체 시스템이 정지하는 문제가 있다. 참여자 입장에서는 변경된 데이터가 Server를 거쳐 참여자에게 전달되므로 정보 전달 지연이 참여자들끼리 직접 주고받는 것에 비해 두 배의 지연시간이 걸리므로 반응 시간이 느려진다.

3. 가상환경을 위한 3차원 협동 저작 도구의 설계

3.1 시스템의 구조

설계에 이용한 기본적인 모델은 네트워크 환경에서 운용되는 Java 응용 프로그램 설계를 위한 Java 분산 어플리케이션 모델을 이용하였다. 이 모델은 기본적으로 Client-Server 구조로서 그림 1과 같이 크게 두 가지로 구분되어진다. 그것은 *2-tier* 모델과 *3-tier* 모델이다.

2-tier 모델은 Client에서 모든 기능을 수행하고 Server에서는 자료 보관 공간만을 제공한다. 3-tier 모델은 Client에서 응용프로그램의 기능만을 담당하고 Server는 공동 저작물의 관리와 실시간 업데이트와 같은 특화 된 작업을 처리한다.

가장 이상적인 Client-Server 구조가 각각의 시스템에게 특화 된

† 본 논문은 한국과학기술평가원 여자대학교 연구기반 확충사업의 연구비 지원으로 수행되었음.

작업만을 처리하도록 하는 것이기 때문에 3-tier 모델을 사용하였다. Client와 Server가 특화 된 기능을 담당하지 않고Client가 모든 기능을 담당한다면 과중한 Client는 증가하는 사용자와 데이터의 양에 따른 환경과 규모의 변화에 제대로 대처하지 못하는 단점이 있다. 만약 동시성 제어와 관련된 로직을 수정해야 한다면 2-tier 모델에서는 사용중인 프로그램을 수거하여 관련 부분을 수정하여 다시 배포해야 하는 등의 문제가 생긴다.

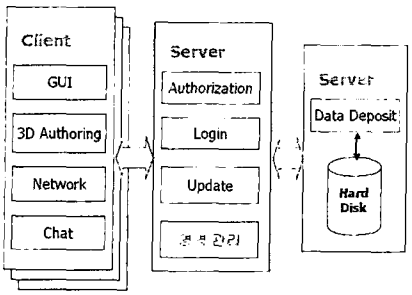

그림 1 전체적인 시스템의 구조

전체적인 구조는 그림 1과 같다. 크게 3D 저작 부분을 담당하는 Client와 공동 저작물의 실시간 업데이트와 공동 저작물을 저장하는 Server로 나누어진다.

3.2 작업자 구분

작업자를 구분하지 않고 설계를 한다면 Lock에 대한 전 과정을 Server에서 처리해야 한다. 그러나 Client를 역할별로 구분을 한다면 Lock관리를 일부 Client에서 맡게 됨으로 작업자를 구분하는 경우가 구분하지 않은 경우에 비해서 Server의 부담이 적어진다.

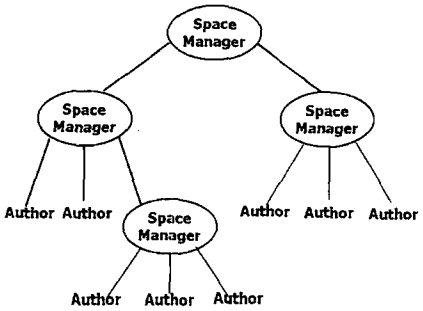

그림 2 작업자의 구분

작업자를 일반 작업자인 ***Author***와 영역을 관리하는 ***Space Manager***로 나누고 각 Author들은 자신이 작업하기를 원하는 영역을 선택 후 자신이 속한 영역의 Space Manager에게 작업권을 요청한다. 요청이 받아들여지면 작업이 가능하다.

공동 작업을 시작할 수 있는 권한을 가진 사람이 처음 Space Manager로 등록한 후 접속하서 설계를 시작한다. 설계 중 새로운 작업자가 필요한 경우 새로운 작업자인 Author를 등록한다. 마찬가지로 영역을 나눌 필요가 생기면 영역을 나누고 이 영역의 Space Manager를 등록한다. 모든 Space Manager들은 해당 영역에서 필요로 하는 새로운 작업자와 Space Manager의 등록, 맡은 영역에 대한 분할과 공동 작업을 Server에 저장하고 자신이 관리하는 영역의 Author가 작업권한을 요청하는 경우 이를 처리하는 등의 맡은 영역에 대한 모든 권한을 가진다. Space Manager는 Author처럼 저작에도 참여 할 수 있다. Space Manager 밑의 Author들은 자신이 작업하길 원하는 영역의 작업권을 Space Manager에게 요청하고 Space Manager가 이 권한을 수락하는 경우라면 이 영역에 다른 작업자의 접근을 막기 위해 Space Manager에 의해 Lock이 걸린다.

3.3 작업권의 요청

다수의 작업자가 공동 작업물에 접근해서 발생하는 충돌을 관리하기 위하여, 동시성 제어 방법으로 작업자 구분을 이용한 *Pessimistic Locking* 방법을 사용한다.

등록된 Author나 Space Manager가 접속하면 등록여부와 아이디와 패스워드를 확인한다. x, y, z의 위치 정보를 이용 작업자가 작업하기를 원하는 작업 영역을 정하고 Space Manager에게 이 영역에 대한 작업권한을 요청한다. 작업 권한의 요청은 작업 창에서 원하는 영역을 선택하거나 공동 작업하는 3차원 장면을 트리 형태로 표현한 '작업장 정보'에서 원하는 영역을 선택한다.

Java 3D에는 공간에 대한 Node는 없다. 장면을 차지하고 있는 도형에 대한 Node만이 있다. 요청자가 요구한 작업 영역의 좌표는 Lock Table이라는 영역 제한을 위해 제안된 Table에 등록된다. 작업자의 작업 요청 과정은 그림 3과 같다.

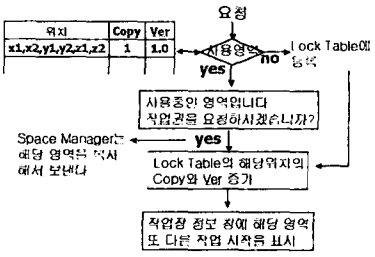

그림 3 작업 요청 과정

3.4 병행작업의 처리

기존 Pessimistic Locking의 경우 작업하기를 원하는 곳이 다른 작업자에 의해 선점이 되어있다면 작업권을 얻기 위한 대기시간이 발생한다. 그렇기 때문에 실제적으로 동시에 같은 작업물에 대한 병행작업은 감소하게 된다. 본 논문에서는 Space Manager의 관리를 이용하여 이러한 병행작업의 감소를 해결한다.

다른 작업자가 작업 요청을 할 경우 이 작업 영역에 대한 좌표가 Lock Table에 있는 지 검사한다. 검사할 때 좌표가 일정부분이 속한다면 일정부분의 정확한 정보를 주고 작업하려는 영역 중 이 부분이 이미 사용중이라는 메시지를 보내고 그 부분에 대해 Space Manager에게 작업요청을 하겠는지를 묻는다. 만약 모든 영역이 작업중이라면 선택 영역 전체가 이미 다른 사람에 의해 작업중이라는 메시지를 보내고 역시 작업을 요청할 것인지를 묻는다. 그리고 작업중인 영역에서의 작업 권한을 받지 않은 사람에 의해 해당 영역에 다우스 클릭과 같은 이벤트가 일어난다면 이벤트가 일어난 위치가 이 Lock Table의 위치정보에 포함되는지 확인하는 과정을 거침으로써 관련 이벤트를 작업권한을 얻은 사람이 아니라면 사용하지 못하게 한다. 또한 그림 4와 같이 전체 작업장을 보여주기 위한 '작업장 정보'의 장면 트리에서 작업하는 영역에 대해 현재 작업중이고 누가 작업하는지를 표시해 준다.

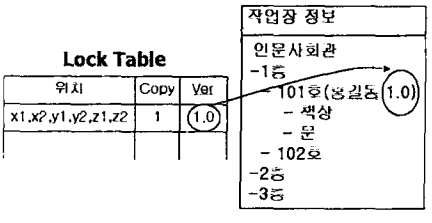

그림 4 Lock Table과 작업장 정보

모든 작업이 끝나면 작업자는 Space Manager에게 작업이 끝났음을 알리고 이 알림을 받은 Space Manager가 이 알림을 받았다고 확인하면 Lock Table에 있는 Copy 정보가 변경되고 만약 완료된 작업 공간이 더 이상의 작업자가 없는 경우라면 위치정보가 삭제되고 장면 트리에서의 표시 또한 사라진다. 아래 그림 5에서 이러한 과정을 간략하게 나타내었다.

같은 영역을 여러 명이 작업하기를 원하는 경우 그 영역의 Space Manager가 해당영역의 장면 트리를 복사해서 새로운 작업 요청자에게 준다. 작업중인 작업에 대한 새로운 요청이 들어왔다는 메시지가 영역 담당인 Space Manager에게 보내지고 해당 장면을 복사해서 주겠느냐고 묻는다. 확인을 선택하면 해당 영역의 장면 트리가 복사되어 작업 요청자에게 보내지고 작업창에 뿌려진다.

따라서 Server에 존재하는 이 Lock Table은 작업되고 있는 곳의 영역 좌표와 작업영역의 결과물을 관리하기 위한 Version과 관련된 정보가 포함되어 있다.

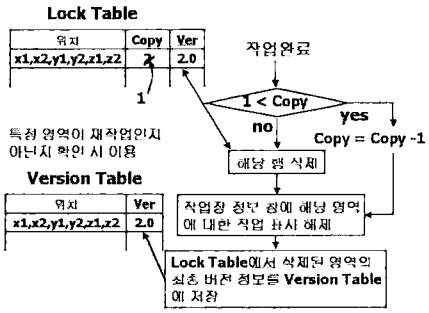

그림 5 작업 완료

영역의 Space Manager는 한 영역의 여러 작업자에 의한 결과들을 보고 가장 적절한 것을 선택한다. Space Manager가 선택한 작업 결과는 Space Manager가 승인을 하면 장면 트리에 반영이 되고 VRML 파일로 변경되어 Server에 저장이 된다. 한 번 작업이 끝난 영역을 다른 사람이 또 수정하기를 원한다면 그 영역의 참여자들의 의견을 반영하여 결정한다. 왜냐하면 작업자들은 이미 수정된 것에 맞추어서 작업을 하기 때문에 수정 후의 결과에 관심을 가지기 때문이다.

그리고 한 번 작업한 것을 수정가능 하도록 한 이유는 다양한 시도로 더 좋은 작업 결과가 나올 수 있는 가능성을 위해서이다. 작업의 수정을 관리하기 위해 Version Table을 사용한다. 이것은 수정의 대상이 되는 영역이 이전에 수정이 되었던 부분이 맞는지 확인하기 위해 사용된다. Lock Table에서 완료된 작업공간이 더 이상의 작업자가 없다면 이 영역에 대한 정보는 Lock Table에서 삭제가 된다. Lock Table만을 이용한다면 이런 경우 후에 다시 수정을 하기 위해 이미 작업이 되었던 공간인지 아닌지를 확인할 수가 없다. 따라서 Lock Table에서 영역정보가 완전히 삭제된 후라면 Version에 관련된 정보만 Version Table에 저장이 된다. 저장된 이 정보와 작업을 하려는 영역의 정보를 비교하여 Version Table에서 같은 영역의 정보가 발견되면 이미 수정이 끝난 영역으로 인식이 된다. 이런 경우 Space Manager에게 통보가 되고 Space Manager가 관리하는 Author들의 의견을 모으게 된다.

Space Manager가 영역에 대한 모든 권한을 가지지만 작업을 하는 Author들에게 직접적으로 영향을 미치는 경우이므로 Author들과의 의견 교환이 필요할 때 또는 영역 자체에 대한 변경을 해야 하는 경우에는 Space Manager들과 의견을 교환해야 한다. 이런 경우는 실시간 메시지 전송 방법을 이용한다.

3.5 공동 작업물의 저장 및 열람

위에서 언급했듯이 Server에 결과물을 저장하는 것은 Space Manager들만이 가능하지만 Space Manager가 본인의 시스템에 저장하는 것도 가능하다. 또한 Author 역할을 맡은 Client도 자신의 작업 시스템에 한해 현재 작업 내용을 저장 할 수 있다. 이렇게 Author나 Space Manager들에 의한 공동 작업물의 저장을 가능하게 하면 Client-Server 구조에서 Server가 다운되거나 했을 때도 작업물의 보관이 가능하게 된다. 협동 시스템을 사용하는 사용자들 중 대부분은 내가 아닌 다른 사용자들의 작업 상황을 보기를 원한다. 이러한 요구사항을 충족시키기 위해 개인 작업공간과 공동 작업공간을 구분하고 공동 작업장 접근과 사용에 관한 설계가 필요하다.

제안하는 협동 저작 시스템에서는 사용자가 시스템에 참여하면 기본적으로 자신이 사용하는 개인 작업장만 보이고 공동의 작업장은 사용자의 선택에 의해 보여진다. 공동 작업 영역의 정보를 트리 형태로 보여주는 '작업장 정보'를 이용하여 사용자가 보기를 원하는 공동 작업장의 상황을 볼 수 있다. '작업장 정보'에서 보기를 원하는 영역을 선택하면 해당 영역을 보여주기 위한 새로운 창이 뜬다.

이렇게 사용자가 원하는 영역만을 선택하여 공동 작업장을 보여주게 설계를 하면 불필요하게 많은 양의 데이터를 보내지 않고 각 Client가 선택한 부분에 맞게 데이터를 보내주면 된다. 이런 경우 Server가 Client에게 보내야 하는 데이터의 양이 적어지고 네트워크 병목현상을 줄일 수가 있다.

4. 결론 및 향후 연구 방향

설계한 시스템에서 제안한 작업자 모델, Client의 선택적인 공동 작업물 저장 및 열람 기능은 일관성을 완벽하게 보장해주는 Client-Server 방식의 Server 병목현상과 네트워크 상의 데이터 양의 증가 그리고 Pessimistic Locking의 사용자 병행작업 감소와 같은 단점들을 보완하기 위한 설계이다.

데이터 양을 줄이기 위해 Server의 기능 중 일부를 Client에게 넘김으로써 데이터의 양은 줄였지만 사용자의 반응시간은 고려하지 않았다. 동시성 제어에서 Server를 사이에 둔 Space Manager인 Client와 Author인 Client사이의 이벤트 처리 과정은 사용자의 반응시간을 늘어나게 하는 요인이 된다. 따라서 사용자의 반응시간을 줄이기 위한 연구가 향후 연구과제이다.

참고 문헌

[1] "미국 3D 애니메이션과 시장 현황 및 전망" http//dsp.hannam.ac.kr/pds/3dani/91106.html

[2] Olof Hagsand, "Interactive Multiuser VEs in the DIVE System.", IEEE Multimedia, Vol. 3, No. 1, pp. 30~39, 1996

[3] Lea, R., Honda, Y., Matsuda, K., Hagsand, O., and Stenius, M., "Issues in the design in a scalable shared virtual environment for the Internet," HICSS, 1997

[4] J. Leigh, A.E. Johnson, T.A. DeFanti, "CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments," Journal of Virtual Reality Research, Development and Applications, the Virtual Reality Society, Vol. 2.2, pp.217-237, 1997

[5] Bentley, R., Horstmann, T., Sikkel, K. and Trevor, J., "Supporting Collaborative Information Sharing with the World Wide Web : The BSCW Shared Workspace System", The World Wide Web Journal, O'Reilly, Dec. 1995

[6] Waters, R.C., Anderson, D.B., and Schwenke, D.L., "Design of the Interactive Sharing Transfer Protocol," IEEE WETICE, pp. 140 147, 1997.

[7] Singh, G., Serra, L., Png, W., and Ng, H., "BrickNet: A Software Toolkit for Network Based Virtual Worlds," Presence, MIT Press, Vol. 3, No. 1, pp. 19-34, 1994.

[8] C. Grimsdale, dVS-Distributed Virtual Environment System, Computer Graphics 91, 1993

[9] T.H. Dani, and R. Gadh, A Shape Design Tool in a Virtual Environment to Support Collaborative CAD, TeamCAD97, pp.11-16, 1997

[10] http://www.ticam.utexas.edu/CCV