협동설계시스템을 위한 오브젝트 Picking과 Concurrency

윤보열[U] 송승헌 김응곤
순천대학교 컴퓨터과학과
byyoon@maesan.org song9232@shinbiro.com kek@sunchon.ac.kr

Object Picking and Concurrency for Collaborative Design System

Bo-yul Yoon [U] Seung-heon Song Eung-kon Kim
Dept. of Computer Science Sunchon National University

요 약

협동설계시스템에서의 공유 오브젝트는 3D 도형이 되며, 사용자가 임의의 오브젝트를 picking하는 문제와 그 오브젝트에 어떤 조작을 취할 때 동시성제어(concurrency)하는 문제가 생긴다. 본 논문에서는 오브젝트의 picking이 마우스 포인터에서의 ray와 오브젝트간에 intersection을 계산하는 방법 외에 scene graph의 노드에 picking 속성을 주는 방법, bounds를 설정하는 방법, picking test의 범위를 한정하는 방법을 사용하여 computation의 부담을 줄이고 효과적인 picking이 이루어지도록 한다. 그리고 picking된 오브젝트에 대해 협동작업에 참여한 사람이 공유작업공간에서 효과적인 동시성제어가 이루어지도록 action에 따라 공유(shared)lock과 전용(exclusive)lock을 사용한다

1. 서론

전통적인 협동작업은 일정한 시간에 일정한 장소에서 함께 만나 자료를 보고 서로 의견을 말하면서 진행되었다. 오늘날에는 컴퓨터와 통신 기술의 발달로 시간과 공간의 제약 없이 공유된 가상 공간에서 상호작용을 하면서 효율적인 작업을 하는 새로운 시스템이 대두되고 있다[1,2].

지금까지 개발되어 이용해온 협동작업시스템은 전자우편을 비롯하여 화상회의, 공동프로그래밍, 전자결재, 원격교육, 원격 진료 등이 있다. 그러나 대부분의 시스템은 특정한 플랫폼과 그룹웨어를 사용하여 폐쇄적으로 공동작업이 이루어지는 경우가 많고, 윈도우의 탐색기 형태를 취하고 있어서 공유 오브젝트가 폴더나 문서, 메모 등에 그치고 있다[3].

우리가 연구한 협동시스템은 인터넷 상에서 웹브라우저를 통해 서버에 접근하여 협동작업이 이루어지도록 되어 있으며, 가상공간에서 협동설계작업을 수행하기 때문에 공유 오브젝트가 3D 도형이 되고 있다[3]. 따라서 다수의 사용자가 임의의 공유 오브젝트를 선택(picking)하고, 그 오브젝트에 어떤 조작을 취할 때 동시성제어(concurrency)하는 것이 중요한 문제가 된다.

본 논문에서는 Java 3D를 이용해 오브젝트의 picking을 단순히 마우스 포인터에서의 ray와 오브젝트간에 intersection을 계산해 내는 것 뿐 만 아니라, scene graph의 노드에 picking 속성을 주는 방법, bounds를 설정하는 방법, picking test의 범위를 한정하는 방법을 사용하여 computation의 부담을 줄이고 효과적인 picking이 이루어지도록 한다. 그리고 picking된 오브젝트에 대해 어떤 조작을 가할 때 협동작업에 참여한 사람이 공유작업공간에서 효과적인 동시성제어가 이루어지도록 action에 따라 공유(shared)lock과 전용(exclusive)lock을 사용하도록 한다.

2. 오브젝트의 picking

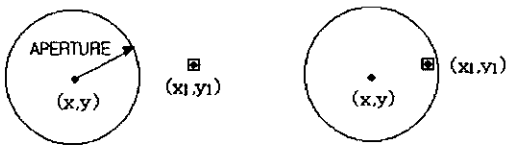
picking이란 화면 내부의 물체 혹은 물체의 일부분을 구성하는 메쉬 등을 선택하는 것이다[4]. 이는 포인팅 디바이스 등으로 어떤 오브젝트를 선택하여 인터액션이 이루어지도록 하기 위한 것으로 마우스 포인터 위치에서 내부 가상 세계에 광선을 쏘아 오브젝트와의 겹치는 부분(intersection)을 찾아내는 것이다.

마우스 포인터(x,y)와 오브젝트(x_1,y_1)와의 거리(D)를 구해 일정한 범위(APERTURE) 내에 들어있는지를 확인한다. <그림1>의 왼쪽은 점(x_1,y_1)이 i)의 경우로 picking이 안되고, 오른쪽 그림은 ii)의 경우로 점(x_1,y_1)이 picking이 된다.

$D^2=(x-x_1)^2 + (y-y_1)^2$

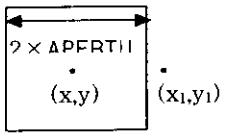
i) $(x-x_1)^2 + (y-y_1)^2 > APERTURE^2$

ii) $(x-x_1)^2 + (y-y_1)^2 < APERTURE^2$



<그림1> 오브젝트가 점인 경우의 picking

앞에서 살펴본 D < APERTURE 를 확인하는 것은 곱셈이 많고 제곱근을 계산해야 하므로 연산시간이 많이 걸린다. 따라서 <그림2>처럼 점(x,y)를 중심으로 한변의 길이가 2×APERTURE인 정사각형의 내부에 점(x_1,y_1)이 들어오는 가를 확인한다. $|x-x_1| + |y-y_1| < APERTURE$



<그림2> 정방형에 의한 테스트

다음은 한 점과 선분과의 거리를 구해 일정한 영역 안에 있으면 picking한다. 주어진 한 점에서 선분에 수선을 그어 그 교점과 주어진 점과의 거리를 구한다. 한 점(x_0,y_0)과 선분 ax + by = 0 와의 거리 D는 다음과 같다.

$$D=\frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$$

또 주어진 한 점(x,y)과 두 점 (x_1,y_1), (x_2,y_2)를 양 끝점으로 갖는 선분과의 거리 D는 다음과 같다.

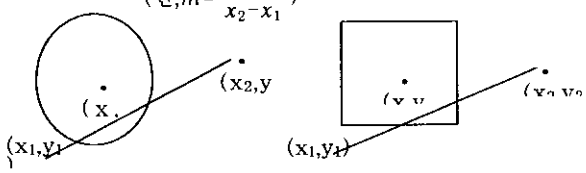
$$D=\frac{|(x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)|}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}$$

따라서 picking이 되기 위해서는 <그림3>처럼 선분이 원 안으로 들어와야 하므로 다음과 같은 조건을 만족하여야 한다. <그림4>는 많은 곱셈 연산을 피하기 위해 원 대신에 정사각형을 사용하였다.

$$\frac{((x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1))^2}{(x_2-x_1)^2+(y_2-y_1)^2} < APERTURE^2$$

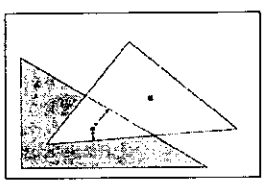
$$\min(|m(x-x_1)+y_1-y|,|\frac{(y-y_1)}{m}+x_1-x|) < APERTURE$$

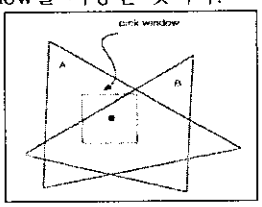
$$(\text{단}, m=\frac{y_2-y_1}{x_2-x_1})$$



<그림3>picking된 선분 <그림4>정사각형을 통한 picking

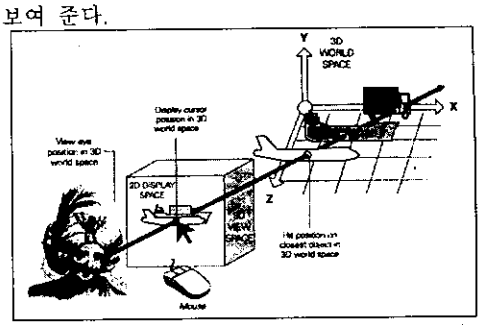
어떤 오브젝트의 내부에 마우스를 클릭하면 그 오브젝트가 선택되었다고 하는데, 하나의 레이어에서 두 오브젝트의의 중첩된 부분을 클릭하였을 때는 어떤 오브젝트를 picking하게 되는지 문제가 된다. 이 경우에는 클릭한 좌표에서 가장 가까운 물체가 picking되도록 한다. <그림5>에서는 점과 가까운 거리에 빗변이 놓인 B가 선택되고, <그림6>은 pick window를 사용한 것이다.





<그림5>중첩된 부분의 picking <그림6>pick window
(그림출처: [4] p.65)

3D 공간에서의 picking은 Z축으로 가장 앞에 놓인 오브젝트를 picking하도록 한다. <그림7>은 마우스에서 ray를 쏘아 첫 번째 비행기의 오브젝트를 picking하는 것을 보여 준다.



<그림8>3D World의 object picking (그림출처:[6]p.249)

앞에서 언급한 방법들은 컴퓨터 내부적으로 많은 산술연산을 요구하고 있다. Java 3D를 이용하여 3차원 가상공간에서 효과적으로 오브젝트를 picking하기 위해 다음과 같은 방법을 사용한다[7].

① scene graph node의 attribute와 capability
② bounds 설정
③ pick testing의 scope를 한정

첫째 방법은 노드에 setPickable() 메소드를 사용하여 true와 false값에 따라 선택한 노드의 하위구조까지도 intersection연산을 하지 않도록 한다.

setPickable(boolean pickable)

또한 특정 Group Node에만 적용하는 capability설정이 있다.

ENABLE_PICK_REPORTING
ALLOW_BOUNDS_READ | WRITE
ALLOW_PICKABLE_READ | WRITE

둘째는 복잡한 도형에 대해 일정한 영역을 설정하여 연산을 빠르고 간단하게 하도록 한다.

USE_BOUNDS
USE_GEOMETRY

셋째는 pick testing의 scope를 한정하여 intersection을 찾기 위해 모든 부분에 대해 연산하는 것을 줄일 수 있다.

4. Concurrency

공유작업공간에서는 협동작업의 일관된 상태를 유지하면서 그룹의 참가자들이 작업을 수행할 수 있도록 지원해야 한다. 이런 협동 작업의 일관성을 유지하기 위해서는 동기화기법이 필요하다. 멀티 뷰를 통한 협력시스템에서의 뷰 동기화(View Synchronization) 유지를 위해 참가자의 조작은 서버로 전달되고 서버는 그룹의 각 참가자들에게 브로드캐스팅하므로 WYSIWIS를 유지하도록 한다[8].

협동 작업의 동시성 제어 (Concurrency Control)를 위해 ordering과 locking을 이용하는 방법이 일반적이다. ordering방법은 여러 참여자의 작업을 일정한 순서대로 계속 적용시켜 나간다. locking방법은 공유객체의 작업

권한을 한 참여자에게 줌으로써 일관성을 유지하는 방법이다. 본 시스템에서는 협동 작업의 객체를 순서대로 제어하는 것이 아니라 동시에 제어해야 하므로 후자인 locking방법을 사용한다.

locking은 locked(잠김)과 unlocked(풀림)의 두가지의 상태로 1과 0의 값 중 하나를 가진다. 하나의 lock은 하나의 오브젝트와 연관된다. 오브젝트 X에 관해 lock(X)=1이면 이 오브젝트를 요청한 action은 오브젝트 X에 접근 할 수 없다. lock(X)=0이면 오브젝트 X의 접근이 가능하다.

두 개의 연산 lock과 unlock은 오브젝트에 어떤 action을 취할 때 반드시 포함되어야 한다. 참가자의 어떤 action이 오브젝트 X에 접근하려 할 때에는 먼저 lock(X)연산을 실행한다. 이 때 lock(X)=1이면 이 action은 기다리고, lock(X)=0이면 이 action은 lock(X)의 값을 1로 고정시키고 오브젝트 X에 접근한다. 이 action이 오브젝트 X의 사용이 끝나면 unlock(X)연산을 실행하여 lock(X)의 값을 0으로 고정시킨다. 이 때 다른 참가자나 다른 action이 오브젝트X에 다시 접근할 수 있게 된다[9].

하나의 action만이 특정 오브젝트에 대해 lock을 걸고 사용하기 때문에 협동작업에 있어서는 너무 제약적이다. 어떤 action이 오직 읽기 위해 한 오브젝트에 접근한다면 여러 action이 이 오브젝트에 병행적으로 접근하더라도 문제가 없을 것이다. 그러나 어떤 action이 오브젝트에 대해 변형을 가하거나 저장하려고 한다면 그 action이 오브젝트X에 대해 독점적 접근을 가져야 한다. 읽기 만을 위해 병행접근을 허용한다면 두 가지 형태의 lock연산이 필요하다.

① 공용로크 (Shared Lock (lock-S)): 어떤 action이 오브젝트X에 대해 공용로크(Lock-S)를 걸면 이 오브젝트에 대해 읽어 올 수 는 있으나 기록할 수 는 없다. 이 때 이 오브젝트X에 대해 다른 action도 공용 lock를 걸 수가 있다.

② 전용로크 (Exclusive Lock(lock-X)) : 어떤 action이 오브젝트X에 대해 전용로크(Lock-X)를 걸면 참가자는 이 오브젝트에 대해 읽고 기록할 수 있다. 이 때 이 오브젝트X에 대해 다른 action은 어떤 lock도 걸 수가 없다.

오브젝트를 생성한 사람 만이 삭제하거나 저장할 수 있도록 한다. 오브젝트의 action에 따른 lock table은 <표1>과 같다.

ACTION \ LOCK	lock-S	lock-X
object create	○	×
object open	○	×
object delete	×	○
object translation	○	×
object zooming	○	×
object rotation	○	×
object save	×	○

<표1>object의 action에 따른 lock table

오브젝트 X에 대해 공용lock이 필요한 action은 lock-S(X)연산을 실행해야 하고, 전용lock이 필요한 action은 lock-X(X)연산을 실행해야 한다. 오브젝트 X에 대한 잠금을 풀기 위해서는 unlock(X)연산을 수행하면 된다. 참가자가 오브젝트를 사용할 때 다음과 같은 locking규정을 따르도록 한다.

1. 참가자가 오브젝트 X에 대해 읽거나 조작하고자 할 때는 lock-S(X)나 lock-X(X)연산을 먼저 실행한다.
2. 참가자가 오브젝트 X에 대해 저장하고자 할 때는 lock-X(X)연산을 먼저 실행한다.
3. 참가자가 lock-S(X)나 lock-X(X)연산을 하려고 할 때, 오브젝트 X가 이미 양립될 수 없는 lock이 걸려 있으면 그것이 unlock이 될 때까지 기다려야 한다.
4. 참가자가 모든 action의 실행을 종료하기 전에 lock을 건 오브젝트에 대해 반드시 unlock(X)를 실행해야 한다.
5. 참가자가 lock을 걸지 않은 오브젝트 X에 대해 unlock(X)를 실행할 수 없다.

5. 결론

협동설계시스템은 가상공간에서 3D 도형을 공유 오브젝트로 사용하게 된다. 따라서 다수의 사용자가 임의의 공유 오브젝트를 picking하고, 그 오브젝트에 어떤 조작을 취할 때 병행수행하는 것이 중요하다.

본 논문에서는 Java 3D를 이용해 오브젝트의 picking을 scene graph의 노드에 picking 속성을 주는 방법, bounds를 설정하는 방법, picking test의 범위를 한정하는 방법을 사용하여 computation의 부담을 줄이고 효과적인 picking이 이루어지도록 했다. 그리고 picking된 오브젝트에 대해 어떤 조작을 가할 때 협동작업에 참여한 사람이 공유작업공간에서 효과적인 동시성제어가 이루어지도록 action에 따라 공유(shared)lock과 전용(exclusive) lock을 사용하도록 했다.

[1] F. Faure, C.Faisstnauer, G.Hesina, "Collaborative animation over the net", IEEE 1999, p107-116
[2] Icgiro Hagiwara and Shinsuke Noda, "Homotopical Modeling as the Basis of New CAD Standard Homotopy CAD for Collaboration Engineering", IEEE 1999, p231-237
[3] 윤보열, 김응곤, "웹 기반 협동CAD시스템에 관한 연구", 한국해양정보통신학회 논문지 Vol.4. No.4, 2000
[4] 주우석, "3차원 컴퓨터그래픽스", 도서출판 그린 p64-65, 1999
[5] 심재홍, "컴퓨터그래픽스",이한출판사, p391-395 1996
[6] Jon Barrilleaux, "3D User Interfaces with Java 3D" , manning, 2001
[7] Denis J Bouvier, "Getting started with the Java 3D API", Sun microsystems, 2000
[8] Jonathan Munson and Prasun Dewan, "A concurrency control framework for collaborative system", Proceedings of the ACM 1996 Conference on CSCW
[9] 이석호, "데이타베이스 시스템", 정익사, 1995