객체 지향 방법을 이용한 CSCW 설계 및 구현

○
김형진 최종명 최재영

숭실대학교 컴퓨터학부

Design and Implementation of CSCW using Object-Oriented Design

Hyungjin Kim Jongmyung Choi Jaeyoung Choi

School of Computing, Soongsil University

요 약

현재 웹상에서 구현된 많은 공동작업 그래픽 편집기들은 디스플레이 정보를 이미지 형식으로 처리하기 때문에 수정이 어렵고, 속도 및 확장성이 떨어지는 단점이 있다

본 논문에서는 공동작업을 위하여 그림 정보, 디스플레이 갱신 그리고 발언권 제어 등을 하나의 완전한 객체로 정의하고 객체가 위치하는 최소영역만을 화면 갱신하며 여러 가지의 정책을 객체내에 포함시킴으로써 확장성과 이식성을 높이고 빠르고 쉽게 구현하도록 하는 객체 지향 설계 방법에 대하여 기술한다 참여자는 객체를 주고 받음으로서 공동작업을 수행하는 객체 브로드케스팅(Object Broadcasting) 방법을 제시한다

1. 서 론

CSCW 시스템은 웹의 폭발적인 인기로 웹상에서 수행될 수 있는 새로운 응용분야들이 계속적으로 연구되어지고 있다. 실시간 공동작업 그래픽 작업 환경은 참여자들이 "당신이 보고 있는 것을 내가 보고 있다"라는 의미의 WYSIWIS 환경에서 작업을 수행하기 때문에 뷰의 일관성을 유지하는 것이 매우 중요하다. 그리고 그래픽 중심의 실시간 공동작업 응용 프로그램은 네트워크상에서 데이터가 이동하는데 소요되는 시간보다는 뷰를 갱신하고 디스플레이의 일관성을 유지하는데 걸리는 시간이 네트워크 대기시간의 95%를 차지하기 때문에 효율적인 뷰의 갱신이 필요하다[3]. 그러나 기존의 많은 실시간 공동작업 그래픽 편집기들은 다음과 같은 기능상 몇 가지의 취약한 점이 있다

첫째, 객체단위의 수정이 불가능하다 이미지 형태의 디스플레이는 이벤트의 정보를 단순히 이미지상에 디스플레이하며 이미지 형태로 저장된다. 이벤트 정보가 이미지 또는 점에 대응되기 때문에 이미지상의 수정은 가능하지만 객체단위의 수정은 불가능하다. 둘째, 화면 갱신 속도가 느리다 마우스 드레그와 같은 많은 양의 이벤트가 이미지 혹은 점에 대응하는 정보이기 때문에 네트워크를 통한 이벤트 정보가 많아지고 빈번한 화면 갱신으로 속도가 느려지게 된다 셋째, 확장성이 떨어진다. 이벤트가 모든 참여자에게 브로드케스팅 되면 이벤트 충돌[4]과 발언권 제어[6] 그리고 MVC 구조[3] 등을 이용하여 일관성을 유지하는 등 여러 정책들이 수반되어 완전한 공동 작업을 수행한다. 여러 가지의 정책들이 독립적이고 복잡하게 처리되면 확장성이 떨어지며, 동기화 정책과 많은 양의 정보 전달에 따른 네트워크의 대기시간을 증가시킨다.

본 논문은 실시간 공동작업 그래픽 편집기를 객체 중심으로 설계하는 방법을 제시한다. 공동작업은 완전한 객체를 정의하여 객체 단위로 저장, 수정, 삭제, 저장 등이 이루어지도록 하며, 디스플레이 갱신은 화면 전체가 아닌 객체가 위치하는 최소 영역에서 디스플레이 갱신이 이루어지도록 한다. 또한 여러 가지의 복잡하고 독립적인 정책을 피하고 확장성을 높이기 위하여 완전한 객체를 참여자에게 전달하는 객체 브로드케스팅(Object Broadcasting)을 제안한다. 모든 참여자는 객체를 받아 버퍼에 저장 또는 수정하여 직접 화면을 갱신하기 때문에 일관성 유지나 이벤트 충돌 등을 위한 정책은 필요로 하지 않는다

본 논문에서는 2장에서 객체 지향 설계 방법에 대하여 기술하였다. 3장에서는 이미지 형태의 디스플레이와 객체단위의 디스플레이 갱신 속도를 이벤트 개수의 처리시간으로 측정하였으며, 마지막으로 4장에서는 결론 및 앞으로 연구 방향을 제시한다.

2. 객체 지향 방법을 이용한 CSCW 설계 및 구현

공동작업을 위한 객체 지향 설계는 기존의 공동작업 그래픽 편집기에서 볼 수 있듯이 이벤트 등의 정보를 받아 화면을 갱신하기 위하여 MVC 구조, 이벤트 브로드케스팅, 이벤트 충돌, 일관성, 발언권 제어 등 여러 가지의 정책들을 이용하여 해결하지만 이러한 정책들을 완전한 객체내에 정의함으로써 확장성과 이식성을 높이고 빠르고 쉽게 구현하도록 하기 위함이다. 참여자는 객체를 주고 받음으로서 공동 작업 수행한다

2.1 객체 설계

하나의 완전한 객체는 그림 정보뿐만 아니라 디스플레이 갱신과 발언권 제어 등을 함께 정의한다. 그림 1은 여러 개의 서브 객체를 완전 객체로 설계하는 방법을 보여준다.

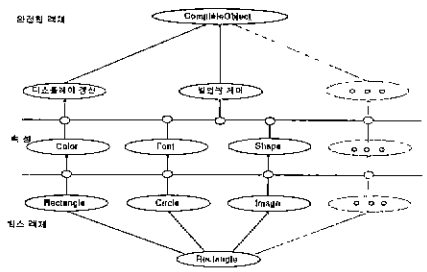

그림 1 완전한 객체 설계

그림 1에서 Rectangle 클래스는 객체 단위의 디스플레이 갱신을 위한 객체의 최소영역을 나타낸다. 최소 객체는 일반적인 그림 정보(사각형, 원 등)를 나타낸다. 속성은 그림 정보에 대한 속성 값(색, 폰트 등)을 부여한다. 디스플레이 갱신은 객체의 최소영역의 화면을 갱신하기 위함이며, 발언권 제어는 잠금 등을 이용한 객체의 점유 정책을 정의한다. 계층적 구조로 된 하나의 완전한 객체가 공동작업의 기본 단위이다 (CompleteObject).

2.2 객체단위의 디스플레이 갱신

마우스 드래그와 같은 많은 양의 이벤트 정보들이 네트워크를 통하여 전송될 때 화면전체를 계속적으로 디스플레이 한다면 속도가 느려지고 결국에는 네트워크의 대기시간을 증가시키는 결과를 초래하게 된다. 따라서 디스플레이 갱신은 선택된 객체의 최소영역만을 갱신하면 된다.

그림 2는 객체단위의 디스플레이 갱신이 이루어지도록 하는 자바 클래스를 보여주고 있다. 클래스는 크게 3 부분으로 분류된다. CompleteObject 클래스는 공동작업을 위한 완전한 객체를 정의한다 특히, CompleteObject 클래스 내에 있는 objectView()는 객체단위의 디스플레이 갱신이 이

```
public class CompleteObject extends ObjectAttribute
{
  public void objectView(Graphics g) {
  }
}
public class ObjectController {
}
  private        View view,
  private        Myobject obj,
  private        Vector bufferObjects,
  public void updateDisplay() {
    view repaint(obj),
  }
}
public class View extends Panel {
  Graphics       offGraphics,
  Image          offImage;
  private        MyObject obj;
  private setDoubleBuffering(int width, int height){
    offImage = createImage(width, height),
    offGraphics = offImage getGraphics();
  }
  public void repaint(Rectangle rc) {
    super repaint(rc.x,rc,y.rc.width,rc height),
  }
  synchronized public void update(Graphics g) {
    Rectangle    clipRect;
    clipRect = g getClipRect();
    offGraphics.fillRect(clipRect.x,clipRect.y
                  ,clipRect.width,clipRect height),
    obj.updataDisplay(offGraphics),
    g.drawImage(offImage, 0, 0, null);
  }
}
```

그림 2. 객체단위의 디스플레이 갱신

루어지도록 하는 메소드이다. View 클래스는 Panel 클래스를 상속받는 일반적인 사용자 디스플레이이다 ObjectController 클래스는 이벤트를 객체로 변환하여 서버로 전달하거나 네트워크를 통하여 전달된 객체를 받아 버퍼에 저장한다. 작업의 흐름을 살펴보면 전달되는 객체는 객체 제어기(ObjectController)가 받는다 객체 제어기는 객체를 버퍼에 저장 또는 수정한 다음 디스플레이의 갱신을 위하여 객체를 뷰(View)에 전달한다 (updateDisplay()). 최종적으로 View내에 있는 update() 메소드가 호출되면 선택된 객체의 최소 영역의 디스플레이가 갱신된다(objectView()). 사용자 디스플레이는 더블 버퍼링을 이용한다.

2.3 객체 브로드케스팅(Object Broadcasting)

객체 브로드케스팅은 그림 정보, 디스플레이 갱신, 발언권 제어 등을 객체내에 포함시킴으로서 확장성과 이식성을 높이고 빠르고 쉽게 구현하도록 하는 기술이다.

그림 3은 객체 브로드케스팅의 처리 과정을 보여준다. 참여자가 발생시킨 이벤트는 객체로 변환하여 서버로 전달되고, 서버는 객체 브로드케스팅 방법을 이용하여 객체를 공동작업에 참여한 모든 참여자에게 전달한다 모든 참여자는 객체를 받아 버퍼에 저장 또는 수정하여 직접 화면을

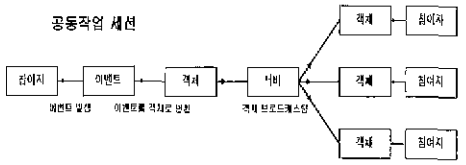

그림 3 객체 브로드케스팅

갱신하기 때문에 일관성 유지나 이벤트 충돌 등을 위한 정책은 필요로 하지 않는다.

3. 실험 방법 및 실험 결과

디스플레이 갱신 속도 향상 면에서 볼 때, 실험 방법은 이벤트의 개수가 객체와 이미지 형식의 디스플레이에서 각각 처리되는 시간을 측정한다. 실험 방법을 구체적으로 살펴보면 다음과 같다.

(1) 입력은 계속해서 증가되는 이벤트의 개수이고 출력은 사각형으로서, 각각의 디스플레이에 대한 입력과 출력은 같다.
(2) 측정값은 각각의 디스플레이 형태에 따라 이벤트가 처리되는 시간을 측정한다
(3) 측정값은 계산된 이벤트 처리시간을 누적시키고 디스플레이 형태에 따라 각각 도식화한다

디스플레이 구조 / 이벤트 개수	객체	이미지
100,000	28	29
200,000	56	59
300,000	83	88
1,000,000	278	293
5,000,000	1388	1461

표 1 디스플레이 형태에 따른 이벤트 처리 시간 (단위:초)

실험에 사용한 시스템은 Intel 펜티엄 CPU 133MHz와 64 MByte의 메모리로 구성되어 있다. 표 1은 디스플레이 형태에 따른 이벤트 처리시간을 측정한 값이다.

그림 4는 디스플레이 형태에 따른 효율성을 도식화한 것으로서 객체 형태가 전반적으로 이미지 형태보다 효율성이 좋았으며, 이벤트의 개수가 많을 때 더 효율적임을 알 수 있다.

4. 결론과 향후과제

본 논문에서 우리는 공동작업을 위하여 디스플레이 정보에 대한 하나의 완전한 객체를 정의하였고, 디스플레이 갱신속도를 향상시키기 위하여 객체단위의 디스플레이 갱신 방법을 제시하였으며, 참여자가 발생한 이벤트를 완전한 객체로 변환하여 모든 참여자에게 다중화함으로써 확장성과 이식성을 높이고 일관성을 유지하도록 하는 객체 브로드케스팅 방법을 제안하였다. 이미지 형태보다는 객체 형태가 디스플레이 갱신 속도 면에서 볼 때 효율성이 월등하게 좋았다. 앞으로 우리는 OLE 객체의 표현과 참여자들간에 음성 통신이 가능하도록 하는 작업을 진행할 예정이다.

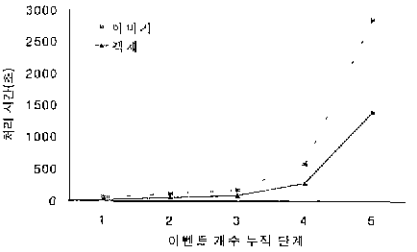

그림 4. 디스플레이 형태에 따른 효율성

[참고 문헌]

[1] 경용득, 최종명, 최계영, 송후봉 "웹미팅(WebMeeting) : 인트라넷 그룹웨어 시스템," 정보과학회 춘계 학술 발표 논문집, p. 437, 1997.

[2] Ulrich Gall and Franz J. Hauck, "Promondia: A Java-Based Framework for Real-Time Group Communication in the Web," Techn. Report, Number TR-I4-96-08, p. 9, 1997

[3] T.C. Nicholas Graham, Tore Urnes and Roy Nejabi, "Efficient Distributed Implementation of Semi-Replicated Synchronous Groupware," Proceedings of the Ninth Annual Symposium on User Interface Software and Technogy, pp. 1-10, ACM Press, November 6-8 1996.

[4] James Begole, Craig A. Struble and Clifford A. Shaffer, "Collaboration Transparency in Java through Event Broadcasting," *IEEE Internet Computing*, Vol. 1, No. 2. March - April 1997.

[5] Philip Isenhour, Clifford A. Shaffer, James Begole, Jeff Nielsen and Marc Abrams, "A Java-Based Framework for Collaborative Interactive Modular Visualization Environments," Virginia University. TR-97-17, October 24, 1997.

[6] Hans-Peter Dommel and J.J. Garcia-Luna-Aceves. "Design issues for floor control protocols," California Santa Cruz Univ. June 1995