PCS 정보단말기에서의 효과적인 윈도우 관리를 위한 방법

최희석, 염근혁
부산대학교 컴퓨터공학과

A method for efficient window management in PCS

Hee Seok Choi, Keunhyuk Yeom
Dept. of Computer Engineering, Pusan National University

요 약

GUI 환경에서 작업에 필요한 여러 윈도우를 작은 화면에 동시에 나타내기 위하여 많은 윈도우 시스템이 오버래핑 윈도우 관리 전략을 채택하고 있다. 하지만 개별 윈도우를 독립적으로 다룸으로써 작업 전환시 개별 윈도우 단위의 정렬을 하여야 하는 등의 불필요한 작업을 수행하게 된다. 특히 본 연구의 목표 기종인 PCS 정보단말기는 320*240 크기의 소형 디스플레이 장치를 가지고 있기 때문에, 가시적인 작업 단위의 윈도우 그룹에 대한 빈번한 변경을 다루는 데 부적합하다. 따라서 본 논문에서는 작업 단위로 윈도우를 그룹화하는 방식으로 윈도우를 관리함으로써 보다 효율적으로 작업 전환을 수행하는 방법을 제안하고, 성능 평가를 통하여 제시된 방법의 효율성을 보여준다.

1. 서론

GUI(Graphical User Interface) 환경에서 윈도우는 결과물을 출력하고, 사용자 입력에 반응하고, 사용자와의 상호작용을 지원하기 위한 디스플레이 화면상의 사각 영역이다. 또한 모든 응용 프로그램은 운영 체제로부터 메시지를 받기 위하여 적어도 하나의 윈도우를 필요로 한다. 그런데 컴퓨터 기술의 발달과 더불어 네트웍, 인터넷의 진보로 사용자는 많은 양의 정보를 다루게 되었고, 작업당 필요한 윈도우의 수도 증가하고 있는 추세이다[1]. 그리고 현재의 멀티태스킹 환경은 평균 작업 수행 시간을 줄이는 등 사용자의 작업 수행 능력에 향상을 가져왔는데, 이러한 멀티태스킹에서 더 많은 이익을 얻기 위하여 윈도우 시스템은 작업간의 전환을 효과적으로 지원해 줄 수 있는 좋은 메커니즘을 제공하여야 한다[1]. 이와 관련하여 윈도우 관리 전략으로서 크게 두 가지가 있다. 하나는 타일 방식으로 윈도우를 배치하는 것으로서 탐색 및 정보 추출 작업에 적합한 인터페이스를 구성하는 데 효과적이나, 소형의 디스플레이 환경에서 각 단위 작업 영역이 작아지는 단점이 있다. 다른 하나는 여러 윈도우를 오버래핑 시키는 방식으로 윈도우를 배치하는 전략으로서 작업간의 빠른 전환을 돕고 작업에 필요한 여러 윈도우를 작은 화면에 동시에 디스플레이 시킬 수 있다는 장점이 있으나, 화면상의 많은 윈도우가 서로 오버랩하고 있으므로 적절한 윈도우를 위치시키는 것이 어렵다[2,3,4].

본 연구는 PCS 정보단말기용 소형 운영체제 개발의 개발 환경인 BRUTUS SA-1100 Evaluation Board의 작은 디스플레이 환경(320 * 240)을 고려한다. 따라서 기존의 오버래핑 윈도우 관리 전략에서 개별 윈도우를 독립적으로 다루기 때문에 문제가 될 수 있는 윈도우 정렬에서의 불필요한 작업을 줄인다. 이렇게 함으로써 작업 전환을 보다 효율적으로 지원한다[5].

이 논문의 구성은 다음과 같다. 2장에서는 기존 윈도우 시스템에서의 윈도우 관리 방법에 대하여 기술하고, 3장에서는 이 논문에서 제안하는 윈도우 관리 방법에 대하여 기술한다. 다음으로 4장에서는 제안한 방법에 대한 성능 평가 및 결과 분석에 대한 내용을 다루고, 마지막으로 5장에서는 결론 및 향후 연구과제에 대하여 알아본다.

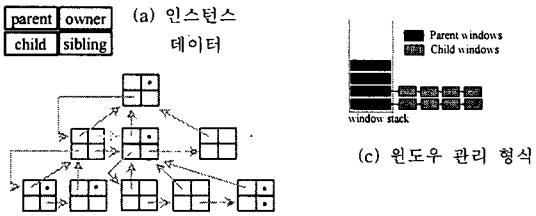
2. 관련 연구

다음 세 가지 윈도우 시스템에서의 윈도우 관리 방법에 대하여 살펴본다.

1) Windows NT/CE System
2) X-Widnow System
3) CellVic

먼저, 1) Windows NT/CE System에서는 윈도우 관리의 세 가지 측면인 가시도, 소유관계, 부모/자식 관계를 다루기 위해 Z order를 이용한다[6,7,8,9]. 그래서 그림 1(a)에서와 같이 윈도우 관리를 위하여 각 윈도우가 가지는 윈도우 인스턴스 데이터를 정의한다. 그리고 각 윈도우는 자신이 가지고 있는 인스턴스 데이터를 이용하여 그림 1(b)와 같은 계층 구조를 형성한다. 여기서 desktop 윈도우는 윈

도우 시스템을 초기에 부팅시킬 때의 루트 윈도우가 되는 것이고, top-level 윈도우는 루트 윈도우를 부모 윈도우로 하여 생성되는 윈도우로서 화면상의 가시도에 따라 정렬된다. 이처럼 Windows NT/CE System에서의 윈도우 관리 구조는 그림 1(c)에서와 같이 간단히 모델링 될 수 있다.



(a) 인스턴스 데이터

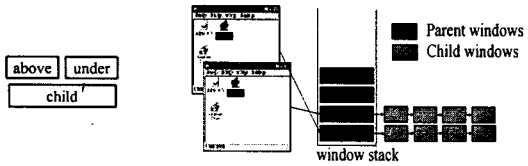
(c) 윈도우 관리 형식

(b) 윈도우 계층 구조

그림 1 Windows NT/CE의 윈도우 관리 구조

이렇게 Z order를 따르는 하나의 리스트를 이용하여 윈도우를 관리함으로써, 윈도우들의 그룹화, 가시도, 계층구조 뿐만 아니라 사용자의 이벤트를 효율적으로 처리할 수 있다. 그러나 각각의 윈도우들이 독립적으로 다루어지고, 일정한 윈도우 정렬 규칙에 따라 윈도우를 정렬하게 되므로 소형의 윈도우 시스템 환경에서는 작업 전환시 불필요한 처리 과정이 존재한다[6].

다음으로 2) X-Window System과 3) JTEL에서 만든 CellVic에서의 윈도우 관리에 대하여 살펴보면 다음과 같다[10,11]. 이들도 위 1)의 시스템과 유사한 방법으로 윈도우를 관리하는데, 그림 2(a)에서처럼 윈도우를 스택킹하는 데 필요한 데이터를 정의한다. 그리고 그 데이터를 이용하여 그림 2(b)에서와 같이 하나의 리스트를 이용하여 겹쳐져 나타나는 윈도우와 자식 윈도우를 관리한다. 이러한 방법도 위 1)에서 제시된 방법과 유사하게 전체 윈도우들을, 개별적으로 다루기 때문에 작업 전환시 불필요한 정렬 작업을 수행하게 된다.



(a) 인스턴스 데이터

(b) 윈도우 관리 형식

그림 2 X-window System과 CellVic에서의 윈도우 관리

다음은 작업 전환시 기존 방법에서 일어나는 일련의 작업을 보여주기 위하여, 그림 3을 이용하여 윈도우즈 시스템에서의 윈도우 관리 방법에 대하여 설명한다.

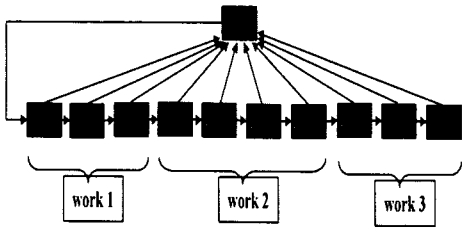


그림 3 Z order를 이용한 윈도우 관리 구조

<작업 전환 과정>

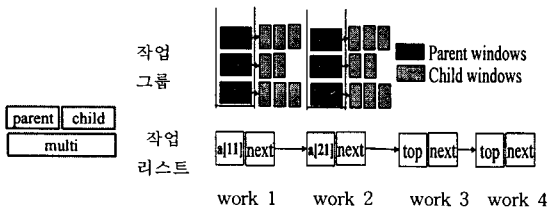
1. 그림 3에서, 현재 work 1이 활성화되어 있는 상태에서 work 2를 활성화시킨다.
2. 윈도우를 재정렬한다.
   2.1 그림 3에서, work 2의 바로 앞의 가시도를 가지는 작업인 work 1의 win1에 대한 핸들을 찾는다.
   2.2 그림 3에서, work 2의 최하위 윈도우인 win2에 대한 핸들을 찾는다.
   2.3 그림 3에서, 2.1과 2.2 과정을 통해 얻은 win1과 win2의 핸들을 이용하여 win1과 own7을 연결한다.
   2.4 work 2를 root에 연결한다.
3. 정렬된 리스트에 따라 화면을 다시 그린다.

따라서, 작업 전환에 따른 이러한 윈도우 정렬 방법은 작업 수와 작업 내에서 오버랩하는 owned 윈도우 수에 따라 작업 전환의 수행 속도가 많은 영향을 받는다.

3. 그룹 단위의 윈도우 관리 방법

목표 기종인 PCS 정보 단말기는 기존 데스크탑 컴퓨터보다 더 작은 320 * 240 크기의 디스플레이 화면을 가지고 있다. 따라서 본 논문에서는 이러한 개발 환경을 고려하여 동시에 화면상에 존재할 수 있는 윈도우의 집합을 하나의 작업 단위(어플리케이션 단위)로 정의하여 윈도우들간의 가시도를 보다 단순화시켰다.

그림 4의 (a)는 본 논문에서 제안하는 방법으로 윈도우를 관리하기 위한 윈도우 인스턴스 데이터이다. 여기서 parent 필드는 자신의 부모 윈도우에 대한 핸들을 가지고, multi 필드는 이전 활성화 윈도우 또는 자식 윈도우의 경우 형제 윈도우에 대한 핸들을 가진다. 그리고 child 필드는 자식 윈도우에 대한 핸들을 가진다. 그림 4의 (b)는 이러한 인스턴스 데이터와 작업별 윈도우 그룹을 단위로 윈도우들을 관리할 경우, 작업 전환시 일어나는 일련의 작업을 설명하기 위한 그림이다.



(a) 인스턴스 데이터

(b) 윈도우 관리 형태

그림 4 그룹 단위의 윈도우 관리 구조

<작업 전환 과정>

1. 그림 4(b)에서, 현재 work 1이 활성화되어 있는 상태에서 work 2를 활성화시킨다.
2. 가시적인 윈도우 그룹을 변경한다.
   2.1 작업 리스트에 저장된 윈도우의 multi 필드값이 a[21]인 윈도우를 찾는다.
   2.2 2.1에서 찾은 윈도우의 multi 필드의 값을 지금 활성화되는 작업의 윈도우 a[21]의 multi 필드의 값으로 변경한다.
   2.3 현재 활성화되는 작업인 work 2 그룹의 최상위 윈도우의 multi필드의 값을 최근의 활성화 작업의 최상위 윈도우 a[11] 핸들로 변경한다.
3. 변경된 윈도우 그룹을 화면에 다시 그린다.

이러한 방법은 작업 단위로 오버랩하는 owned 윈도우의 수에는 영향을 받지 않으며, 작업 수에 있어서도 각 작업별로 최상위 윈도우에 대하여 한번의 검색만 이루어지므로 작업 수에도 크게 영향을 받지 않는다. 따라서 간단한 과정을 통해 작업 전환이 이루어지므로 작업 전환의 수행 속도에 향상을 가져온다.

4. 성능 평가

이 장에서는 본 논문에서 제안한 그룹 단위의 윈도우 관리 방법에 따라 작업 전환을 할 경우 Windows CE 시스템과 비교하여 성능을 평가한다[12]. 아래 평가 결과는 펜티엄 II 프로세서(266MHz)를 이용하여 수행한 결과이다.

먼저 그림 5는 작업 수를 4로 하고 각 작업이 가지는 오버랩하는 owned 윈도우 수를 2로 하였을 경우, 작업 전환의 수에 따른 수행 시간을 보여준다. 여기서 알 수 있듯이 기존 방법은 본 논문에서 제안하는 방법에 비해 약 3배 정도의 수행 시간이 소요된다.

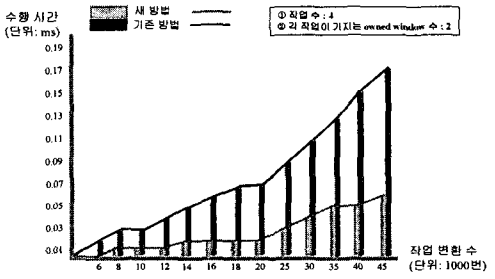


그림 5 작업 전환의 수에 따른 수행 시간

다음으로 그림 6은 작업 수와 작업 전환 수를 동일하게 하고 단지 각 작업내에서 오버랩하는 owned 윈도우 수를 증가시켜 나갈 경우의 성능 비교에 대한 것이다. 여기서 보면 기존 방법은 owned 윈도우 수에 크게 영향을 받지만, 본 논문에서 제안하는 방법은 영향을 받지 않는다.

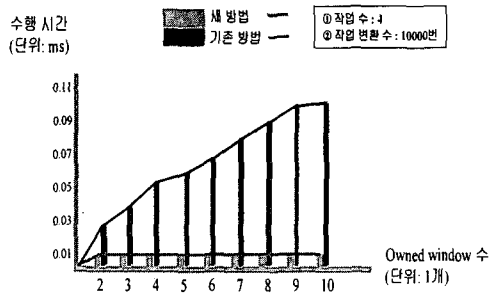


그림 6 owned 윈도우 수에 따른 수행 시간

마지막으로 그림 7은 작업 수와 owned 윈도우 수, 작업 전환 수를 다양하게 하였을 경우의 전체적인 성능 비교를 나타낸다. 여기서 보면 기존 방법은 작업 수와 작업내의 오버랩하는 owned 윈도우 수에 따라 선형적으로 수행 시간이 증가한다. 그러나 본 논문에서 제안하는 방법은 작업 내의 owned 수에는 영향받지 않고, 작업 수에도 크게 영향받지 않음을 알 수 있다.

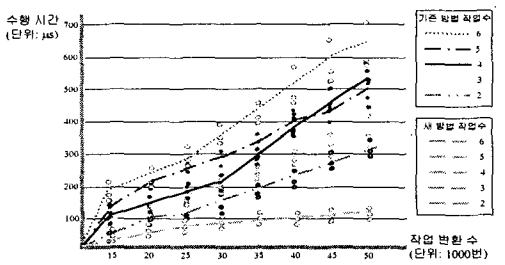


그림 7 작업수, owned 윈도우수, 작업전환수에 따른 수행시간

이상에서 살펴본 바와 같이 본 논문에서 제안한 윈도우 관리 방법은 작업(어플리케이션) 단위로 윈도우 그룹을 형성하여 작업별로 윈도우를 정렬함으로써, 기존의 윈도우 관리 방법에 비하여 수행 속도가 빠를 뿐만 아니라, 하나의 작업 내에 겹쳐져 존재하는 owned 윈도우의 수에 영향받지 않으므로 보다 안정적인 수행 시간을 보장해 줄 수 있다.

5. 결론 및 향후 연구방향

이 연구에서는 목표 기종인 PCS 소형 정보단말기에 적합한 윈도우 관리 방법으로서, 그룹 단위의 집합적 윈도우 관리 방법을 제안한다. 따라서 기존의 윈도우 관리 방법에서 개별 윈도우를 독립적으로 다룰 때 수행되는 작업 전환시의 불필요한 윈도우 정렬 작업을 줄임으로써 작업 전환의 수행 시간을 향상시켰다. 향후에는 window operation의 효율성을 향상시키기 위하여 타일 방식의 윈도우 관리 전략[5]이 아닌 오버래핑 윈도우 관리 전략에서의 개선 방법에 대한 연구가 필요하다.

참고문헌

[1] Kandogan, E., Shneiderman, B., "Elastic Windows: Imporved Spatial Layout and Rapid Multiple Window Operation", Proc. Advanced Visual Interfaces '96, ACM, New York, NY, pp. 29-38, May 1996

[2] J.Corde Lane, Steven P.Kuester, and Ben Shneiderman, "User Interfaces for a Complex Robotic Task: A Comparison of Tiled vs. Overlapped Windows", Morgan Kaufman, 1997

[3] Shneiderman, B., "Designing the User Interface: Strategies for Effective Human-Computer Interaction, Third Edition", Addison-Wesley, 1998

[4] Bly, S., Rosenberg, J., A comparison of tiled and overlapping windows", Proc. CHI '86 Conference - Human Factors in Computing Systems, ACM, New York, NY, pp. 101-106, 1986

[5] Kandogan, E., Shneiderman, B., "Elastic Windows: Evaluation of Multi-Window Operations", Proc. ACM CHI'97, pp. 250-257, March 1997

[6] Microsoft, "MSDN Online Library", http://msdn.microsoft.com/library, 1999

[7] Microsoft, "Microsoft Windows CE Platform SDK version 2.0", http://www.microsoft.com/windowsce, 1998

[8] John Murray, "Microsoft Inside Windows CE", Microsoft Press, pp 80-106, 1999

[9] Charles Petzold, "Programming Windows 95", Microsoft Press, 1996

[10] XConsortium,"X Window System", http://www.rahul.net/kenton/xsites.html#XConsortium, 1999

[11] JTEL, "CellVic OS 1.0 Documents", http://www.jtel.co.kr, 1999

[12] Ellis Horowitz, Sartaj Sahni, Susan Aderson-Freed, "Fundamentals of Data Structures in C", Computer Science Press, 1993