Journal of Korea Multimedia Society Vol. 11, No. 10, October 2008(pp. 1420-1426)

트리 구조를 가진 범용 자원관리시스템의 설계

신현규[†], 이충호[††]

요 약

최근의 프로그램들은 그림 파일이나 오디오와 같은 대용량의 자원을 사용하는 경향이 있다. 이와 같은 대용량 자원을 사용하는 경우 그 자원의 크기가 클수록 자동 관리 시의 효율성이 더 많이 저하되며, 자원의 수가 많을수록 관리의 어려움이 증가된다. 이에 본 논문은 파일시스템의 트리 구조의 인터페이스를 자원관리 시스템에 적용하여 편의성과 효율성을 도모한 자원관리시스템을 제안한다. 이 시스템은 문자열경로 기반의 접근방식을 채택하여 높은 호환성을 가지고 있으며, 트리 구조 지원으로 높은 구조성을 도모하고, 가상 노드의 지원으로 높은 효율성을 보인다. 본 시스템은 프로세스 안에서의 자원 관리를 위해 설계되었으나, 프로세스 외부로부터의 자원제어도 지원한다.

Design of General-Purpose Resource Management System with Tree Structure

Hyunkyu Sin[†], Choong Ho Lee[††]

ABSTRACT

The most recent programs tend to use picture or audio files which need mass storage resources. For these mass storages, the performance of automatic management decreases as the sizes of these resources increases. Moreover, the more the number of resources increases, the more the management of resources gets difficult. This paper suggests a resource management system to which the interface of tree-structure file system is applied in order to increase convenience and efficiency. The system uses the access technique based on character string path to have more compatibility, and supports tree structure which is more systematic. Further, it supports virtual nodes to show high performance. The system can be used to control resources from outside as well as inside of processes.

Key words: File System(파일 시스템), Resource Management(자원 관리), Design Pattern(설계 패턴)

1. 서 론

자원의 관리가 중요한 예로 게임 프로그래밍을 들 수 있다. 여러 개의 스테이지로 구성된 게임이 있다고 가정할 때 시스템의 안정성을 위해 각각의 스테이지는 가능한 독립성을 지녀야 한다. 즉 자원이 스테이지 단위로 로드 되고 해제 되어야 한다. 그러나 스테이지 간에 중복 사용되는 자원이 있을 수 있으므로 성능을 개선하기 위한 다양한 관리 방법이 적용될 수 있다[1].

자원 관리 시스템은 효율성뿐만 아니라 다수의 자원 관리도 염두에 두고 설계되어야 한다. 각각의 자원을 쉽게 구분할 수 있고 다수의 자원을 제어할 수 있으면서 다수의 자원과 개개의 자원을 동일한 방법으로 제어할 수 있다면 이상적이다. 또한 'C#'이나 'Java'의 가비지 컬렉션과 같이 자원은 최소한 자동

※ 교신저자(Corresponding Author) : 이충호, 주소 : 대전광역시 유성구 덕명동 산 16-1(305-719), 전화 : 042)821-1221, FAX : 042)821-1595, E-mail : chlee@hanbat.ac.kr
접수일 : 2007년 12월 28일, 완료일 : 2008년 8월 8일

[†] 정회원, DTVinteractive 연구소 연구원
(E-mail : rurouni0@dreamwiz.com)
[††] 정회원, 한밭대학교 정보통신전문대학원 부교수

으로 해제 되도록 하여 안전성을 보장하여야 하며, 부가적으로 수동 관리를 지원하도록 함으로써 효율성을 높일 수 있다. 나아가 반자동 관리를 지원한다면 프로그래머의 편의성도 높아진다[1,2]. 하지만 지금까지 연구된 자원관리시스템은 각각의 응용프로그램에서 특정 자원관리방식을 채택하여 시스템에 구현되어 있을 뿐이어서, 모든 응용프로그램의 실행시에 범용 적으로 사용될 수 있는 효과적인 자원관리 시스템은 아직 설계되어 구현된 바 없다.

따라서, 본 논문에서는 특정 상황에서의 자원 관리가 아니라 범용 적으로 사용될 수 있는 자원 관리의 방법으로 파일시스템에서 많이 사용되고 있는 트리 구조의 인터페이스도입을 제안한다. 링크 노드를 지원하는 논리적인 구조는 자원의 중복을 막아주며 폴더 개념의 도입은 독립성을 구축하고 반자동적인 관리를 가능하게 한다. 여기에 이벤트 개념의 추가로 활용성을 높일 수 있다.

본 시스템은 자원이 주로 프로세스 안에서 사용되는 점에 착안하여 프로세스를 루트로 하는 일종의 파일시스템을 구축하는 방식을 제안하며, 가비지 컬렉팅과 같은 전자동 자원제어가 낮은 효율을 보이는 환경에서의 이용을 목표로 한다[3].

2. 관련 연구

본 시스템의 설계에 있어 파일시스템의 분석은 필수적이다. 또 구현에 있어서 디자인패턴의 적용은 프로그램의 효율성을 높여준다.

2.1 파일 시스템

파일 시스템은 운영체제차원에서 지원하는 기초적인 저장소로 운영체제마다 다양한 형식의 파일시스템을 지원한다. 본 논문에서는 현재 국내에서 가장 많이 쓰이고 있는 윈도우의 파일 시스템을 분석한다.

자원 저장소로서 파일 시스템은 몇 가지 문제를 가진다. 첫째로 확장자로 표현되는 타입정보는 그 정확성을 보장하지 않으며 둘째로 경로문자열에서 폴더와 파일의 구분이 불분명하다. 본 논문에서는 파일시스템의 특징인 경로문자열 식별자와 트리 구조를 채택하면서 위에 열거한 단점을 개선한 자원관리 시스템을 제안한다.

2.2 디자인 패턴

본 시스템은 트리 구조를 이용[4]하여 자원을 제어한다. 이런 구조에는 컴포지트 패턴의 적용이 적합하다. 참고문헌 [5]에 따르면 컴포지트 패턴이란 부분과 전체의 계층을 표현하기 위해 복합 객체를 트리 구조로 만드는 것으로, 컴포지트 패턴으로 구성된 클래스들은 사용자가 개별 객체와 복합 객체를 동일하게 다룰 수 있도록 한다. UML(Unified Modeling Language)로 표현한 컴포지트 패턴은 그림 1과 같다.

그림 1에 나타났듯이 클라이언트는 복합 객체와 단말 객체를 컴포넌트라는 상위 클래스로 접근하여 이용하기에 복합객체인지 단일객체인지 여부에 상관없이 이용할 수 있으며 복합 객체는 자기 자신을 자식으로 포함할 수 있기 때문에 연산을 재귀적으로 단말 노드에 이를 때까지 전파하게 된다.

본 논문에서 제안하는 자원관리 시스템은 파일 시스템과 마찬가지로 경로문자열을 통해 전역적으로 접근 할 수 있으며 이를 구현하기 위해 싱글톤 패턴을 적용할 수 있다. 참고문헌 [5]에 따른 싱글톤 패턴은 유일한 인스턴스를 생성하고 이에 접근하기 위한 유일한 접근 방법을 제공하는 패턴이다.

3. 제안하는 자원 관리 시스템

제안하는 자원 관리 시스템은 다음과 같은 특징들을 가진다.

3.1 자원 식별자

자원을 식별하기 위한 방법에는 여러 가지가 있

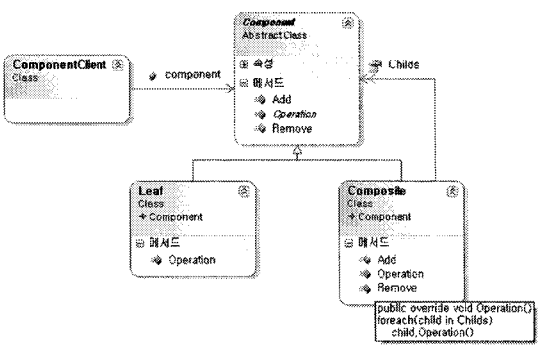

그림 1. 컴포지트 패턴

다. 가장 직접적인 방법으로 인스턴스를 사용하는 방법이 있고, 또 다른 방법으로 식별자를 부여하는 방법이 있다. 본 시스템에서는 식별자를 부여하는 방식을 따르지만 일반적인 정수형 식별자가 아니라 좀 더 많은 정보를 포함하는 문자열의 식별자를 사용한다. 이 식별자는 PRL(Process's Resource Locator)로서 프로세스를루트로 하며 경로정보와 타입정보를 포함하고 있다. 이는 개발자가 디스크의 파일에 접근하는 것과 유사하게 메모리의 자원에 접근하는 것을 가능하게 한다.

PRL의 경로에 포함된 자원 폴더 노드(이하 폴더 노드)들은 ':' 문자로 끝나며 폴더 이름과 자원 이름은 알파벳과 숫자와 몇 가지 기호로 구성된다. PRL은 타입정보를 포함하는데 타입정보는 파일명이 될 수 없는 문자('?')로 시작하여 PRL과 파일시스템의 경로가 직접적으로 대응되는 것을 막는다. 이는 문자열이 자원시스템의 식별자인지 파일시스템의 식별자인지 구분할 수 있도록 한다. 본 시스템에서 제안하는 PRL의 완전한 정규 표현식은 식(1)과 같다.

:([._a-zA-Z0-9]:)*[_.a-zA-Z0-9]+\?[_.a-zA-Z0-9]+ (1)

PRL은 항상 절대경로를 지정하며 상대경로를 지원하지 않는다. 그러나 3.2절의 명령어 파서에 추가적인 구현으로 상대경로를 지원토록 할 수 있다.

3.2 가상 자원 노드

본 시스템의 자원 노드는 시스템 자원과 1:1 대응되는 물리적인 구조가 아니라 N:1로 매칭되는 논리적인 구조를 지닌다. 이를 구현하는 가상 자원 노드(이하 가상 노드)를 링크 자원 노드(이하 링크 노드)라 칭하며 이 노드는 기존의 자원을 이용하여 생성되기 때문에 일반 노드보다 효율성이 높다.

링크 노드는 참조 무결성을 가지지 않는 단순 링크 노드로 원본이 변경되더라도 링크 노드에는 영향을 끼치지 않는다.

본 시스템에서 제안하는 또 다른 가상 노드로 엔트리 노드가 있다. 엔트리 노드는 초기화 되지 않았지만 초기화 정보를 가지고 있는 일종의 프록시 노드를 말한다. 엔트리 노드와 3.3절의 이벤트의 결합으로 자원의 JIT(Just-In-Time) 로드가 가능해진다.

표 1. 자원 관련 이벤트

이벤트	설명
Load	자원 노드 생성시의이벤트
Unload	자원 노드 제거시의 이벤트
Move	지원 노드 이동시의 이벤트
Link	자원 노드 링크시의 이벤트
Update	자원 노드 수정시의 이벤트

3.3 이벤트

리스너를 호출해 주는 이벤트는 자동화의 핵심이다. 본 시스템에서는 표 1과 같은 이벤트를 지원한다.

일반적인 경우 Load 이벤트와 Move 이벤트는 연달아 발생하게 되지만 가상 자원 노드의 경우 Move 이벤트가 먼저 발생하는 경우도 있다. Move 이벤트는 폴더에서 일어나는 경우 하위노드들에게도 전파되지만 Link 이벤트는 전파되지 않는다.

3.4 자원 루트 노드

본 시스템은 내부 인터페이스를 자원 루트 노드(이하 루트 노드)라는 싱글톤을 통해 제공한다. 루트 노드 그 자체는 싱글톤으로 존재하는 폴더 노드로써 모든 자원 노드가 루트 노드 아래에 생성되게 된다. PRL에서 폴더 노드는 이름과 접미어 ':' 문자로 구성되는데 루트 노드는 예외적으로 이름 없이 ':' 문자만으로 표현된다.

3.5 자원 제어 문자열

본 시스템은 컴파일 타임에서의 제어를 위한 API와 런 타임에서의 제어를 위한 명령어를 포함한다. API는 명령어를 구현을 위해 만들어 졌지만 문자열 파싱의 부하를 줄이기 위해 개발자가 임의로 사용하는 것도 가능하다.

명령어 파서는 자원관리 시스템의 상위에 존재하는 쉘(Shell)로 볼 수 있으며 사용의 편리를 위해 다양한 명령들이 추가될 수 있다. 그러나 호환성을 위해 필수적인 명령은 표준으로 정할 필요가 있다. 자원 관리에 필수적인 명령의 표준 사항은 표 2와 같다.

FTP (File Transfer Protocol) 클라이언트 프로그램이 서버 플랫폼에 관계없이 Ftp 명령어를 통해 파일을 제어하듯이 본 시스템도 외부에서 내부구현에

표 2. 자원 관리 명령어

입력 명령어	결과
new type PRL (value)	자원 노드를 생성
delete PRL	자원 노드를 삭제
move PRL PRL	자원 노드를 이동
link PRL PRL	자원 링크 노드를 생성
update PRL (value)	자원 노드를 변경

상관없이 관리 명령어를 이용해 자원을 제어하는 것이 가능하다.

3.6 외부 인터페이스

본 시스템의 API(Application Program Interface)는 인터널(internal)형의 접근 권한을 가져 모듈 내에서의 사용만을 강제하며 외부에서의 제거는 명령어 문자열에 의해 이루어 진다. 시스템이 문자열 명령어 시스템을 포함하고 있기 때문에 외부 인터페이스는 문자열을 입력 받아 결과를 출력하는 함수 하나만으로 구현할 수 있다. 이 방식은 명령어가 추가되거나 제거되는 변동이 있어도 인터페이스의 변경이 필요 없는 장점이 있다. 본 시스템에서 노출하는 인터페이스는 식 (2)와 같으며 문자열을 받아 결과 문자열을 돌려준다.

public string TryResourceCommand(string) (2)

4. 자원관리시스템의 자동화

4.1 Load 이벤트를 이용한 JIT 로드

자원의 로드는 프로그램을 지연시켜 사용자의 불편을 야기하기에 자원을 필요한 시점에만 로드 하는 방식으로 이 지연을 분산 시킬 수 있다. 본 시스템은 자원 노드와 별도로 엔트리 노드를 통해 JIT 없이 JIT 로드의 이점을 누릴 수 있다. 상태 변수를 이용하는 방법이 매번 상태를 점검해야 하는 오버헤드가 있는 반면 상태 패턴을 응용해 설계된 엔트리 노드는 일단 자원 노드로 전환되고 나면 어떠한 상태 점검도 필요하지 않게 된다.

4.2 Unload, Move 이벤트를 이용한 참조무결성

링크 노드는 참조무결성을 만족하지 않으며 원본 노드의 변화가 링크 노드에 영향을 미치지 않는다. 그러나 때때로 참조무결성이 필요한 경우가 있으며 본 시스템에서는 이를 구현하는 것이 가능하다. 노드의 변화 이벤트를 추적하여 링크 노드를 정정하게 되면 참조무결성이 만족되는데 이에 Remove 이벤트와 Locate 이벤트를 이용할 수 있다. 링크 노드의 데이터는 원본 노드의 데이터를 가리키기 때문에 원본 데이터의 변화가 링크 노드의 데이터에 자동으로 적용이 되며 따라서Update이벤트는 추적하지 않아도 된다.

4.3 파일시스템과 매핑

외부 저장소와의 매핑은 개발시간을 단축시켜 준다. 본 시스템의 단말 노드는 파서를 지정해 파일과 1:1로 매핑 하는 것이 가능하기 때문에 파일 시스템의 폴더를 자원관리 시스템의 폴더 노드에 매핑 하는 것도 역시 가능하다. 폴더 단위의 작업은 관리의 편의성을 극대화 시켜준다.

5. 자원관리 시스템의 보안

본 시스템이 의해 관리되는 자원은 제공되는 외부 인터페이스에 의해 간단히 노출된다. 이는 사용자에 의한 커스터마이즈를 가능하게 한다는 장점과 악의적 접근을 허용하는 단점이 있다. 관리의 편의성과 보안성은 반비례 관계에 있기에 개발자는 둘 중 하나를 포기해야 한다. 좋은 대책 중 하나는 외부 인터페이스를 디버그 빌드에서만 도구와의 연동을 위해 사용하고 릴리즈에서는 외부 인터페이스를 제외시키는 방법이다. 이 방식은 릴리즈 시 성능상의 이점도 얻을 수 있다.

또 다른 방식으로 외부 인터페이스를 통해 암호화된 문자열을 주고 받는 그림 2와 같은 방법이 있다. 올바른 키에 의해 암호화된 명령어만이 올바르게 실

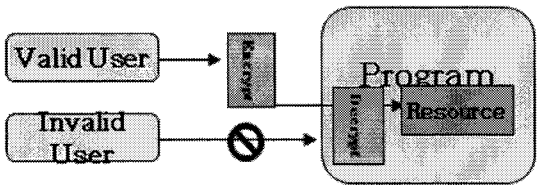

그림 2. 명령어 암호화 방식

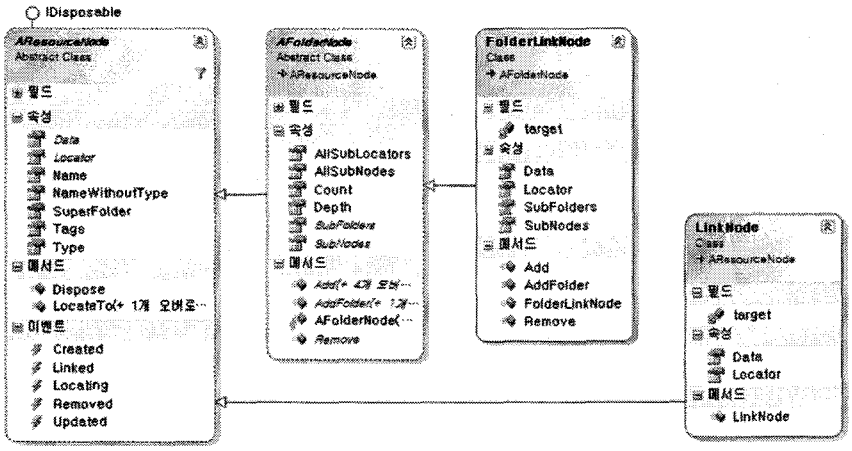

그림 3. 가상 자원 노드의 클래스 다이어그램

행될 것이고 다른 경우 오류를 일으킬 것이다. 이 방식은 성능상의 단점이 있지만 릴리즈 버전에서도 자원관리 시스템의 기능을 활용할 수 있다.

6. 자원관리시스템의 구현 및 고찰

제안하는 설계를 시험하기 위해 자원관리시스템과 이를 이용하는 프로그램을 'C#'으로 구현하여 소개한다. 구현은 환경과 목적에 따라 다를 수 있으나 이용자는 제안된 식별자, 이벤트, 명령문자열을 이용해 구현에 무관하게 자원관리시스템을 사용할 수 있다. 구현은 본 논문의 주제를 넘어가기에 자세히 설명하지 않으며 시스템의 특징을 모두 포함하는 가상 자원 노드의 클래스 다이어그램만을 그림 3에 소개한다. 컴포지트 패턴의 컴포지트에 해당하는 폴더 노드와 가상 폴더 자원 노드가 표시 되어 있으며 리프 노드에 해당하는 가상 자원 노드가 표시 되어 있다. 모든 노드들은 자원 노드라는 추상 클래스를 상속받으므로 동일한 연산으로 접근할 수 있다.

본 논문에서 제안한 자원관리시스템과 이를 이용해 스킨을 변경할 수 있는 프로그램을 구현하였다. 그림 4와 같이 프로그램은 내장된 그림파일을 자원관리시스템에 '/img/Background*image' 라는 노드로 로드하고 화면에 표시하여 준다.

그리고 이 프로그램은 추가로 파일시스템을 읽어들여 자원시스템과 자동맵핑을 시도하며 기존의 노드가 존재하는 경우 그를 덮어쓴다. 그림 5와 같은 파일시스템을 맵핑하는 경우 'Background'라는 노드를 덮어쓰게 되어 그림 6과 같이 프로그램의 외관이 변경된다. 이 방식의 스키닝은 별다른 작업 없이 스킨을 입히는 것이 가능하며 부분적인 스키닝 지원

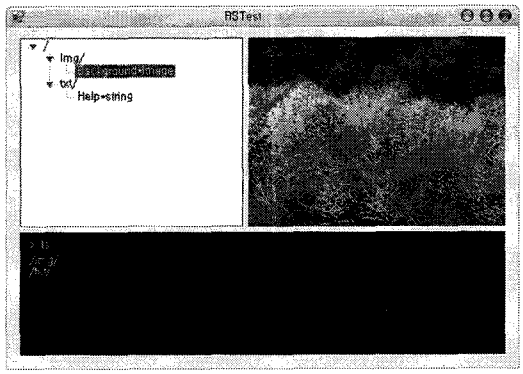

그림 4. 프로그램에서 구현한 자원 트리

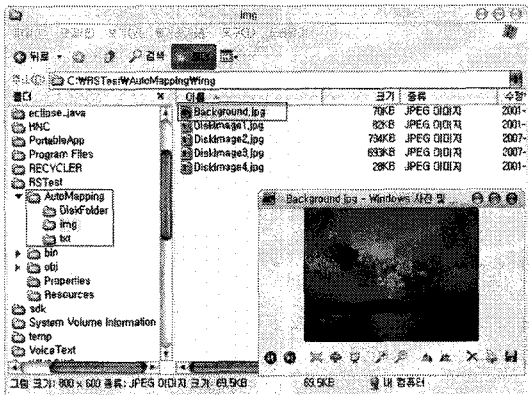

그림 5. 맵핑할 파일시스템

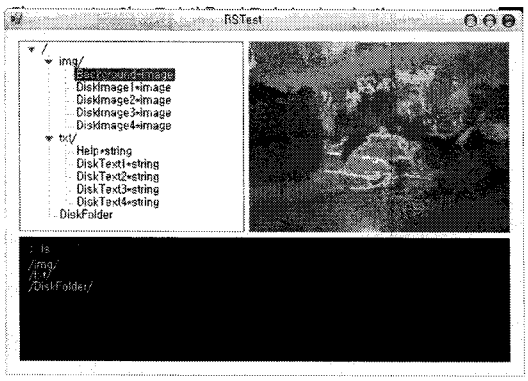

그림 6. 맵핑 후 스킨이 변경된 프로그램

과 스킨을 임의로 제거하더라도 프로그램에 문제를 일으키지 않는 장점이 있다.

7. 자원관리시스템의 평가

본 논문은 자원에 문자열 식별자를 부여하여 제어하는 방법을 제안하였으며 이 방법에는 크게 2가지 장점이 있다. 하나는 모든 자원의 제어에 복사나 삭제와 같은 동일한 인터페이스를 사용 할 수 있다는 것이고, 다른 하나는 여기서 말하는 자원이 문자열로 표현할 수 있는 어떠한 대상도 될 수 있다는 것이다. 예를 들어 클래스를 파일에서 생성하는 리플렉션, 인스턴스를 파일로 저장하거나 읽어오는 직렬화/역직렬화 같은 기법들을 본 시스템에서는 단순히 복사로 사용할 수 있다.

본 시스템은 자동화를 통해 자원의 관리비용을 절감하기 위해 디자인 되었으며 같은 부모 아래에 있는 자원을 단 1번의 명령으로 제어할 수 있다. 단 자원에 접근할 때에는 식별자를 거치는 과정이 필요하며 이 과정은 각 부모노드마다 반복된다. 이 과정의 부하를 줄이기 위해 저장소 캐시를 도입할 수 있으며 캐시 히트시 한 번의 식별자 검색으로 자원에 접근할 수 있다. 이 사항들을 제안시스템을 사용하지 않은 경우와 비교한 결과를 표 3과 같이 정리할 수 있다.

본 자원관리시스템의 설계는 특정 알고리즘의 사용을 고려하지 않는다. 따라서 리스트를 대상으로 한 삽입, 삭제, 정렬 등의 알고리즘을 적용하는 경우 구현에 따라 성능 차이가 있을 것이며, DFS, BFS 등의 트리 알고리즘은 일반적으로 원 알고리즘의 시간 복잡도를 유지할 것이다. 본 시스템에서는 추상 폴더 노드를 통해 다양한 폴더 노드를 구현하는 것이 가능하기에 다양한 구현을 제공하여 경우에 따라 이용하게 하는 방법도 채택할 수 있다.

표 3. 자원관리시스템의 평가

	제안 시스템	힙 메모리 할당자	가비지 컬렉터
제어비용 (N:자원수)	$O(1)\sim O(N)$	$O(N)$	$O(N)$
접근속도 (M:부모수)	$O(2)\sim O(2M)$	$O(M)$	$O(MN)$
자원누수	없음	있음	없음
자동화	일부 제공 및 구현 표준 제공	임의 구현	임의 구현

8. 결론 및 향후 연구과제

본 논문은 다수의 자원 관리와 대용량의 자원 관리에 초점을 둔 자원관리시스템의 한 예를 보였고 이를 위해 파일시스템의 인터페이스를 응용하였다. 그 결과 자원을 폴더별로 정리하고 제어할 수 있기 때문에 다수의 자원을 효율적으로 관리할 수 있었고, 링크 노드의 지원으로 폴더의 독립성을 유지하면서도 자원을 재할당 하는 일 없이 효율적으로 공유할 수 있었다

많은 프로그램이 데이터를 소스코드에 포함시키기보다 외부에 저장한다. 이렇게 외부에 저장된 데이터 자원은 실행시간에 로드되기 때문에 독립적인 편집이 가능해지는 장점이 있다. 이런 장점에 추가로, 본 논문에서 제안하는 시스템에서는 로드 된 자원의 편집까지 가능하게 함으로 프로그램의 유지보수성을 향상 시킨다.

본 시스템이 문자열에 의한 제어를 지원하기 때문에 이 시스템을 사용하는 프로세스들은 서로의 자원에 대한 정보를 얻을 수 있다. 이를 이용해 리소스 탐색기와 같은 일반적인 프로그램을 작성하는 것이 가능하다.

본 논문에서 시스템은 'C#'으로 구현되었으나 'C++' 등 다른 언어의 구현도 가능하다. 특히 외부 인터페이스를 COM (Component Object Model) 등의 기술을 이용하여 구현하면 다른 언어에서 본 시스템을 활용하는 것이 가능하다.

참 고 문 헌

[1] L. Valente and A. Conci, "Automatic resource management as a C++ design pattern," *Proceedings of the 2nd conference on USENIX Conference on Object-Oriented Technologies*, Vol.2, pp. 2~3, June 1996.

[2] ECMA, Standard ECMA-334: C# Language Specification, 3rd Ed., Dec. 2002.

[3] D. A. Park and Stephen V. Rice, "A Framework for Unified Resource Management in Java," *ACM International Conference Proceeding Series*, Vol.178, pp. 113~122, 2006.

[4] D. A. Park, Simplifying Resource Management in Java, Technical Report 2006-05., *Master's Thesis, University of Mississippi*, May 2006.

[5] E. Gamma, et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, Massachusetts, 1995.

신 현 규

2002년 3월~2006년 2월 한밭대학교 정보통신컴퓨터공학부 공학사

2006년 3월~2008년 2월 한밭대학교 정보통신전문대학원 정보통신공학과 공학석사

2008년 3월~현재 DTVinteractive 연구소 연구원

관심분야 : 소프트웨어 공학, 3D 그래픽, 디지털 통신

이 충 호

1981년 3월~1985년 2월 연세대학교 전자공학과 공학사

1985년 3월~1987년 2월 연세대학교 대학원 전자공학과 공학석사

1995년 4월~1998년 3월 토호쿠대학 대학원 정보과학연구과 공학박사

1987년 2월~2000년 2월 KT 멀티미디어연구소

2000년 2월~현재 한밭대학교 정보통신전문대학원 정보통신공학과 부교수

관심분야 : 디지털신호처리, 영상처리, 패턴인식, 컴퓨터 그래픽