

게임 클래스의 시공간 개념 확장을 위한 클래스 분류 방법에 관한 연구

김달중, 하수철

대전대학교 공과대학 컴퓨터공학과 Media&S.E. Lab

A Study on a Class Classification of Game Classes for the Extention of Spatio-Temporal Concepts

Dal-Joong Kim, Soo-Cheol Ha

Media&S.E. Lab, Dept. of Computer Engineering, Taejon Univ.

요 약

최근의 급속한 정보 통신 기술의 발달로 인하여 음향 효과, 정교한 그래픽 처리 등을 이용하는 실제와 유사한 게임 소프트웨어들이 제작되고 있으며, 게임 개발자들에게 개발 기간 단축과 개발의 편의성을 위한 멀티미디어 게임 저작 도구가 필요하게 되었다. 이러한 멀티미디어 게임 저작 도구의 핵심 구성 요소가 되는 클래스 라이브러리 개발을 위해 멀티미디어 게임 객체와 객체들의 행위들을 정교하게 분류할 수 있는 게임 클래스 분류법이 필요하다. 본 논문에서는 게임 클래스 시공간 개념에 대한 정교한 분류 방법을 제안한다. 이를 위해 시공간 개념 중심으로 Enumerative 분류 방법을 이용하여 게임 클래스들을 분류하는 방법을 제시하였으며 게임 클래스의 유사도에 의한 클래스 클러스터링을 통하여 C++ 언어의 특징인 클래스들 사이의 계층 구조를 표시할 수 있으며 소프트웨어의 클래스 구조를 쉽게 확장하여 클래스 구조를 변형할 수 있도록 제시하였다.

1. 개 요

최근 게임 소프트웨어의 빠른 발전에 의하여 게임 소프트웨어가 대형화되고 있으나 게임 소프트웨어에 대한 사용자의 새로운 요구를 적기에 충족시키지 못하므로써 게임 소프트웨어의 생산성 향상에 대한 관심과 소프트웨어 재사용 증대가 관심을 보이고 있다.

이러한 게임 개발의 복잡성을 간소화하기 위하여 게임 제작자들은 게임 저작 도구를 사용하는데, 멀티미디어 게임 저작 도구는 멀티미디어 게임 타이틀을 쉽게 제작할 수 있도록 지원하는 소프트웨어 개발 도구이다. 게임 저작 도구의 주요 기능은 첫째로 게임에 사용되는 다양한 미디어

디바이스의 원활한 연결과 컨트롤이며, 둘째로 게임 개발자의 다양한 요구에 따른 게임 객체들 상호간의 동기 문제들이다.

본 논문에서는 게임 공간에 사용되는 게임 클래스들에 대한 영역을 분석하고 시간 관계성과 공간 관계성을 토대로 게임 클래스들에 대한 체계적인 분류를 한다. 게임 클래스 분류는 멀티미디어 게임 저작 도구와 연동되는 핵심 구성 요소의 클래스 라이브러리를 구성하여 게임 제작의 용이성을 제공하기 위함이다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 클래스 분류 방법들에 대하여 알아보고, 3장에서는 게임 클래스들을 Enumerative 분류 방법을 적용하여 시공간 개념

을 확장한 분류 방법으로 제시하며, 4장에서는 새로운 클래스가 등록될 때 사용되는 클러스터링에 관한 연구를 논한다.

2. 관련 연구

클래스 분류 방법으로는 패킷에 의한 분류와 Enumerative 분류 방법, 클러스터링에 의한 분류 방법이 있고 이러한 분류 방법을 통해 클래스 라이브러리를 구성할 수 있다[2]. 게임 클래스의 분류 방법은 게임에서 사용되는 객체들에 대한 시공간 개념의 특성을 반영하여 클래스에 대한 대상 영역 특징을 잘 표현할 수 있어야 한다.

2.1 패킷에 의한 클래스 분류 방법

패킷에 의한 클래스 분류 방법은 클래스의 성질을 잘 나타내줄 수 있는 패킷에 대해 제한된 어휘(controlled vocabulary)를 제공하여 그 어휘를 사용하여 클래스를 대표하는 방법이다.

클래스의 성질을 나타내는 패킷은 크게 클래스의 기능적인 면을 설명하는 부분과 클래스가 실행되는 환경에 대해 설명하여 주는 부분으로 나누어진다. 게임 객체는 객체 지향 언어로 작성될 수 있는 클래스이며 그 속성으로 멤버 데이터와 멤버 함수의 두 가지 형태의 멤버를 가진다. 이것은 클래스의 기능을 수행하기 위해 필요한 두 가지 형태의 속성을 가진다. 이 두 가지 형태로 추상화한 클래스에 기능적인 측면을 설명한다.

클래스가 실행되는 환경에 대한 설명은 새로운 환경에 적용되어도 프로그램의 기능은 변화하지 않는 내부적인 환경과 환경의 변화로 프로그램이 변화할 수 있는 외부적인 환경으로 나뉜다[3].

이러한 패킷에 의한 클래스 분류는 계층성이 없어 명시적 표현이 불가능하며 객체 지향에는 알맞지 않은 단점이 있다[3].

2.2 Enumerative 분류 방법

운동들의 집합을 점차 좁은 클래스로 분할해 가면서 그들 사이의 계층적인 관련성을 표현하는 방식으로 모든 가능한 클래스들이 미리 정의된다. 이 방법은 운동이 갖는 여러 가지 속성들을 한꺼번에 고려하면서 분류 도표(classification map)에 표현하기 때문에 운동들간의 계층 관계가 자연스럽게 나타나는 반면, 이러한 경우 분류도표가 매우 커지고 복잡해 지므로 새로운 운동의 추가가 어렵기 때문에 확장성이 좋지 않다[4].

이 분류 방식의 예는 아래와 같다.

생리학(physiology)

호흡(respiration)

재생 작용(reproduction)

바다 동물(water animals)

바다 동물의 생리학(physiology of water animals)

바다 동물의 호흡(respiration of water animals)

바다 동물의 재생 작용(reproduction of water animals)

육지 동물(land animals)

육지 동물의 생리학(physiology of land animals)

육지 동물의 호흡(respiration of land animals)

육지 동물의 재생 작용(reproduction of land animals)

무척추 동물(invertebrates)

무척추 동물의 생리학(physiology of invertebrates)

etc.

이러한 분류 방식[5]은 예상되는 모든 클래스들을 미리 계층 구조로 정의한 후에 각 부품들을 가장 적합하다고 생각되는 계층 구조상의 클래스에 사용자가 직접 할당하는 방법이다. 예로서 존 두이의 심진 분류(도서관에서 이용되고 있다), ACM의 Computing Reviews[6] 등이 있다. 이 방법은 클래스들 간의 계층 구조들을 표현하지만 계층 구조상에서 상속성을 표현할 수 없고, 새로운 클래스들을 등록하기 어렵기 때문에 확장성이 좋지 않다[4].

2.3 클러스터링

클래스를 등록할 때 유사도 계산을 통해 클래스의 계층 구조를 생성하여 클래스 상호간에 상속성을 추가하여 상속 계층 구조를 만드는 방법이다. 클러스터링을 위한 계층 구조로 최상위 구조인 카테고리 계층 구조와 하위 계층 구조로써 클러스터 안에 있는 클래스들 사이의 상속과 유사도에 의한 클래스 계층 구조가 있다.

클러스터링을 통한 게임 클래스의 분류는 재사용을 위한 클래스 라이브러리를 구축하는데 목적이 있으며 객체 지향 개념의 상속관계를 지원하는 것에 알맞으며 클래스 상호간의 유사성 측정을 기반으로 객체 지향 상속관계를 지원한다. 유사성 측정 방법에는 데이터 유사도(data similarity),

함수 유사도(function similarity), 클래스 유사도(class similarity)로 나누어진다[7].

· 데이터 유사도

$$S_d(A,B) = \frac{d(A \cap B)^2}{d(A) \times d(B)} \dots \dots \dots (1)$$

d(A) : 클래스 A의 멤버 데이터의 개수
 d(B) : 클래스 B의 멤버 데이터의 개수
 d(A ∩ B) : 클래스 A와 B의 공통된 멤버 데이터 개수

· 함수 유사도

$$S_f(A,B) = \frac{f(A \cap B)^2}{f(A) \times f(B)} \dots \dots \dots (2)$$

f(A) : 클래스 A의 멤버 함수의 개수
 f(B) : 클래스 B의 멤버 함수의 개수
 f(A ∩ B) : 클래스 A와 B의 공통된 멤버 함수의 개수

· 클래스 유사도

$$S(A,B) = P \times S_d(A,B) + (1-P) \times S_f(A,B) \dots \dots (3)$$

P : 두 클래스의 멤버 데이터의 총 수를 두 클래스의 멤버 데이터와 함수의 총 수로 나눈 값

3. 게임 클래스의 분류

본 논문에서는 게임 클래스의 분류를 Enumerative의 분류 방법과 유사도 계산을 통한 클러스터링 방법을 사용한다.

Enumerative의 분류 방법에서는 게임 클래스를 하향식으로 먼저 분류하며 클래스 사이의 계층적인 관련성을 표현하고 클래스 사용자에 의한 인위적 분류를 통하여 게임 클래스 카테고리들을 만든다. Enumerative 분류 방법을 이용하여 게임 클래스를 나누는 기준은 다음과 같이 분류한다.

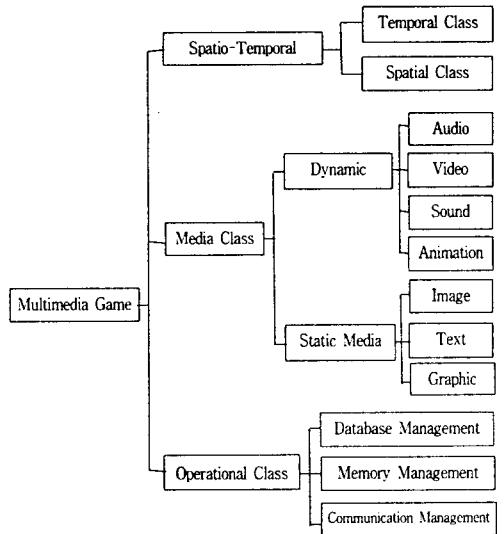
게임 클래스의 분류는 미디어들을 시간 순서에 맞게 처리할 수 있는 시간 관계성과 공간상에서의 가시적 크기와 접촉점을 일치시키는 공간 관계성을 명시하고, 서로 동기화 시킬 수 있는 객체들을 포함하는 시공간 클래스(Spatio-Temporal Class)와 동적 미디어와 정적 미디어들의 다양한 멀티미디어 표현 객체들을 포함하고 있는 미디어 클래스(Media Class), 그리고 시공간 클래스와 미디어 클래스를 제외한 구성요소들로 이루어진 운용 클래스(Operational Class) 등으로 구분한다[8].

게임 클래스의 미디어 클래스는 게임에서 활용되는 미디어 객체들의 시간 흐름에 근거하여 동적(dynamic)/정적

(static) 미디어로 전통적인 미디어 분류 방법에 기초한다[11]. 시공간 클래스는 미디어 객체들이 게임에 사용될 때 시간 관계성과 공간 관계성을 중시하여 미디어 상호간의 동기화 정보를 관리하기 위한 클래스가 된다. 그 외의 운용에 관계한 클래스들과 데이터베이스 관리, 메모리 관리, 통신 등을 관리하는 운용 클래스로 분류할 수 있다.

새로운 클래스를 등록하려 할 때는 클러스터링 분류 방법을 이용하여 클래스들 상호간의 유사도에 따라 클래스 계층구조를 형성하고, 클래스들 사이의 중복을 회피하여 멀티미디어 게임 클래스의 계층 구조 특성을 유지하도록 구성하기 위해 게임 클래스 카테고리에 계층적으로 삽입하게 된다.

이러한 멀티미디어 게임 클래스들은 Enumerative 분류를 통해 계층 구조를 갖는 최상위 구조의 카테고리 계층 구조로 나타낼 수가 있다. 게임 클래스 분야의 카테고리 계층 구조는 해당 분야의 영역 분석과 계층적 관련성을 토대로 <그림 1>과 같은 구조가 만들어진다.



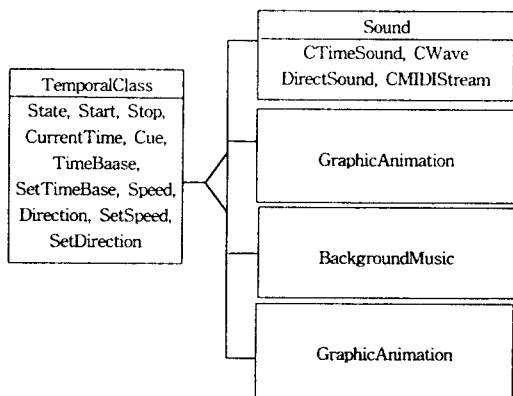
<그림 1> 게임 클래스 카테고리 계층 구조

3.1 시공간 클래스 분류

시공간 클래스는 두 가지 서브클래스를 가지며 시간 클래스(TemporalClass)와 공간 클래스(SpatialClass)로 나누어진다. 또한 나누어진 두 클래스들의 하위 계층들의 클래스로 구성된다. 클래스 상호 간에 가지고 있는 유사성과 클래스들이 행하는 작업들로 분류가 되며 상위 클래스가 가지고 있는 데이터와 메소드들을 공유하며 상관 관계를 가진다.

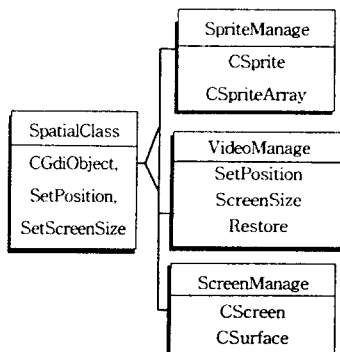
게임에 사용되는 사운드와 같은 시간 의존적 미디어는

시간적 사건에 따라 미디어들을 동기화할 필요가 있다. 이러한 시간적 동기화 방안은 Allen[9]이 제시한 13가지 시간 관계성을 가지고 시간 클래스가 적용될 수 있도록 구성한다. 시간 클래스는 사운드(Sound), 그래픽(Graphic), 배경음악(Background), 효과음(EffectSound) 등으로 분류된다.



<그림 2> Temporal Class의 예

공간 클래스(SpatialClass)는 공간 동기화와 밀접한 관계를 가지고 있다. 공간상의 가시적인 크기는 사용자에게 설정된 공간으로 멀티미디어 객체가 출력 디바이스를 통해 디스플레이 될 때 사용자에게 의해 인식될 수 있는 멀티미디어 객체의 표시 공간을 의미하며, 접속점은 출력 디바이스의 표시공간을 나타내는 표시 크기(presentation size)안에서 가시적 크기를 결정하는 좌표이다. 이와 같은 멀티미디어 객체의 가시적 크기와 접속점의 일치를 공간 동기화라 한다[7].



<그림 3> Spatial Class 분류 예

멀티미디어 게임 클래스에서 스크린에 공간적 위치를 가

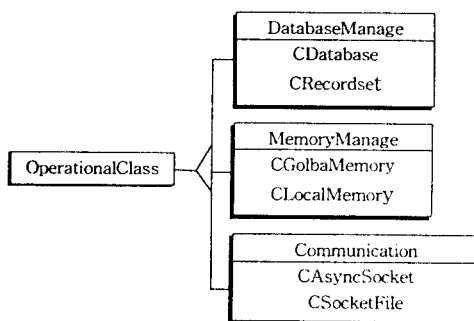
지고 보여지도록 관리하는 클래스들을 공간 클래스로 분류한다. 공간클래스의 분류는 공간적 위치에 대한 클래스의 역할을 고려한 분류를 하며 클래스의 새로운 등록은 클러스터링 방법의 클래스 유사도를 계산하여 클래스들 상호간의 상관관계를 중심으로 계층구조를 나타내어 재사용성을 증대시킨다.

3.2 미디어 클래스와 운용 클래스

시공간 클래스 외에 미디어 객체들을 가지는 미디어 클래스와 게임에서의 특별한 객체를 포함하는 운용 클래스가 있다.

미디어 클래스는 오디오(audio), 비디오(video), 뮤직(music), 애니메이션(animation) 등을 포함하는 동적 미디어(DynamicMedia)와 텍스트(text), 그래픽(graphic), 이미지(image) 등을 포함하는 정적 미디어(StaticMedia)로 분류되며 이러한 미디어들의 형식이나 변환에 대한 정보를 가지게 된다[11].

운용 클래스는 게임에서 비디오 파일을 플레이 할 때 비디오 샘플들이 특정한 시간동안에 변환기에 도달해야 하므로 데이터의 검색 및 전송에 의존적이다. 멀티미디어 게임 객체들이 메모리에 효과적으로 적재될 수 있도록 하는 것은 게임의 진행 속도에 많은 영향을 미친다. 운용 클래스는 멀티미디어 게임 데이터들의 효율적인 저장과 관리를 하는 데이터베이스, 메모리, 통신 등으로 분류한다.



<그림 4> Operational Class의 예

4. 게임 클래스의 유사도에 의한 클러스터링 방법

클래스의 유사도를 구하기 위해서 2장에서 연구한 유사도 구하는 공식을 이용한다. 유사도를 구하기 위해서는 먼저 C++ 원시 코드가 가지고 있는 클래스들을 추출해야 한다. 이 때 추출되는 정보는 클래스 이름, 부모 클래스

(super class), 자식 클래스(subclass), 멤버 데이터(member data), 멤버 함수(member function) 등이다. 이렇게 추출된 정보들은 데이터 유사도와 함수 유사도를 구하는데 사용된다.

클래스 유사도를 구하기 위해서는 데이터 유사도와 함수 유사도를 구해야하며, 우선 함수 유사도를 구하는 방법을 설명한다.

클래스 A에 있는 f(A)(클래스 A의 멤버 함수의 개수)와 f(B)(클래스 B의 멤버 함수의 개수)를 구한다. 다음으로 f(A ∩ B)(클래스 A와 B의 공통된 멤버 함수의 개수)를 구한다. f(A ∩ B)를 구하기 위해서는 멤버 함수의 유사한 정도를 측정하여 두 함수가 유사한지 그렇지 않은지 결정을 내려야한다. 함수의 유사한 정도를 구하기 위해서 Boyer-Moore의 패턴 매칭 알고리즘(pattern matching algorithm)[12]을 이용한다. 데이터 유사도에서는 데이터 타입이 같고 멤버 데이터의 이름이 유사한 경우에만 값을

찾지만 데이터 타입이 다르고 이름이 같은 경우는 제외한다.

두 함수의 유사한 정도가 0.70 이상이면 이 두 함수는 유사하다고 판단한다. 클래스 유사도의 임계값은 논문에서 0.65로 설정하며 이것은 기존의 연구에서[13] 제안된 유사도 임계값인 클래스 유사도 값 0.7 이상이면 유사하다고 판단하고 있으며 소프트웨어 클래스 분류 방법 기술 연구[4]에서는 클래스들의 유사도를 0.6 이상에서 유사하다고 보고 있다. 본 논문에서는 두 유사한 값의 중간 값인 0.65를 임계값으로 정하여 판단한다. 함수 유사도와 데이터 유사도는 그 유사한 정도의 임계값을 0.7로 적용시킨다.

클래스 유사도를 구하기 위해서 Bullet 클래스와, Item 클래스로 예를 들어 설명한다. 클래스 내부 정보를 추출한 결과를 보면 <그림 5>와 같다. 추출된 정보를 가지고 유사도를 구하면 다음과 같다.

$$\begin{aligned}
 f(\text{Bullet}) &= 7, f(\text{Item}) = 6, f(\text{Bullet} \cap \text{Item}) = 4 \\
 S_r(\text{Bullet}, \text{Item}) &= 0.38095 \\
 d(\text{Bullet}) &= 4, d(\text{Item}) = 2 \\
 S_d(\text{Bullet}, \text{Item}) &= 0.50000, \\
 P &= 0.46154 \\
 S(\text{Bullet}, \text{Item}) &= (0.50000 \times 0.46154) + (0.38095 \times 0.53846) \\
 &= 0.41855
 \end{aligned}$$

Bullet 클래스와 Item 클래스의 클래스 유사도 값이 약 0.41855의 값이 나온다. 이 값은 본 논문에서 제안하고 있는 임계값 0.65 보다 작은 값이므로 두 클래스를 유사하다고 판단하지 않지만 이 두 클래스는 같은 슈퍼 클래스로부터 상속을 받고 있기 때문에 같은 계층에 분류된다.

이렇게 구해진 값들과 Enumerative 분류 방법을 가지고 카테고리 계층을 구성한다. 클래스 유사도는 카테고리 에 등록된 클래스 중에서 유사한 값을 갖는 계층에 클래스를 계층적으로 등록한다. 유사한 클래스 값이 존재하지 않으면 Enumerative 방법을 적용하여 임의적으로 카테고리에 등록을 하게 된다.

5. 결론

본 연구에서는 C++ 프로그램 코드로 구성된 게임 클래스를 재사용하기 위한 기본 연구로 클래스들을 분류하는 방법을 제안하고 이 방법을 이용하여 클래스들의 계층 구조를 구성하였다. 게임에 사용되는 기존의 많은 클래스들은 Enumerative 방식에 의해 카테고리를 구성하고 새로운 클래스를 등록할 경우에는 클래스들의 유사성을 기반으

클래스 이름	Bullet		
	슈퍼클래스	멤버 함수 개수	멤버 데이터 개수
	1	7	4
슈퍼 클래스	BaseObject		
멤버 함수	static void ReadSpriteImage(S8 *filename); static void DestroySprite(void); Bullet(U16 X,U16 Y,U8 MX,U8 MY,FourDir FirstDir); void CheckSurround(void); void Action(); SPR GetSprite(U8 Num); ~Bullet(void);		
멤버 데이터	FourDir Direction; U8 Life; S8 ScrXStep,ScrYStep; static SPR Sprite[MaxBullSpr];		

클래스 이름	Item		
	슈퍼클래스	멤버 함수 개수	멤버 데이터 개수
	1	6	2
슈퍼클래스	BaseObject		
멤버 함수	static void ReadSpriteImage(S8 *filename); static void DestroySprite(void); Item(U16 X,U16 Y,U8 MX,U8 MY,U8 SprN); void Action(); SPR GetSprite(U8 Num); ~Item(void);		
멤버 데이터	U8 Life; static SPR Sprite[MaxItemSpr];		

<그림 5> Bullet 클래스와 Item 클래스의 멤버 함수와 멤버 데이터 추출 결과

로 클러스터링 방법을 사용하였다. 이 방법은 Enumerative 분류 방법이 가지고 있는 계층적인 분류를 보장하고 유사도에 의한 분류 방법으로 객체 지향 언어가 가지고 있는 상속성을 추가하여 구성하였다.

게임 소프트웨어가 대형화되면서 고품질의 게임과 게임의 개발 기간 단축에 많은 노력들이 보이고 있다. 논문에서는 게임 소프트웨어 개발에 사용되고 있는 다양한 클래스들을 이용하여 시공간 개념 중심으로 Enumerative 분류 방법을 이용하여 게임 클래스들을 분류하는 방법을 제시하였으며 게임 클래스의 유사도에 의한 클래스 클러스터링을 통하여 C++ 언어의 특징인 클래스들 사이의 계층구조(상속성)를 표시할 수 있으며 소프트웨어의 클래스 구조를 쉽게 확장하여 클래스 구조를 변형할 수 있도록 제시하였다. 이렇게 분류된 게임 클래스들은 게임 클래스 라이브러리를 구성하여 멀티미디어 저작 도구에서의 사용과 객체 지향적인 게임 개발에 재사용이 증대될 것이다.

게임 개발을 위한 멀티미디어 게임 저작 도구 기술의 개발을 위한 본 연구의 멀티미디어 게임 클래스 분류는 게임 클래스 라이브러리를 효과적으로 구성하는데 목적이 있다. 효과적으로 구성된 게임 클래스 라이브러리는 객체 지향적인 개념을 포함하는 멀티미디어 게임 저작도구 개발을 앞당길 것이다.

추가 연구로는 게임 클래스의 시공간 관계성을 바탕으로 정교하게 분류할 수 있는 기술이 필요하며 멀티미디어 게임 저작 도구와 게임 클래스 라이브러리 사이를 연결할 수 있는 미들웨어의 개발이 계속되어야 할 것이다.

참고문헌

[1] 강수진, 남종호, "시간 및 공간적 사건을 위한 멀티미디어 동기화 명시 방법", 정보과학회 논문지 제24권 제1호, pp. 1~12, 1997.

[2] 우치수, "객체 클래스 라이브러리 기술 개발에 관한 연구", 한국전자통신연구원 최종보고서, 1996.11

[3] 이병정, 오재현, 권용길, 김희천, 우치수, "CORBA 기반의 클래스 라이브러리 설계 및 구현" 정보과학회, Vol.24, No.2, 1997.

[4] 남윤석, "소프트웨어 클래스 분류 방법 기술 분석", 전자통신연구소 기술 연구 보고서, 1997.

[5] R.Prieto-Diaz, "Implementing Faceted Classification For Software Reuse." Comm. ACM, vol.34, no.5, pp. 89~97, May, 1991.

[6] J.E.Sammet and A.Ralston, "The New Computing Reviews Classification System-Final Version."

Comm. ACM, vol.25, no.1, pp. 13~25, 1983.

[7] 허계범, 최영근, "래피드 프로토타이핑 기법을 사용한 객체 지향 클래스 계층 구조 설계 방법", 정보처리학회 논문지 제5권 제1호, pp. 86~96, 1998.1.

[8] 김단중, 권혁훈, 하수철, "멀티미디어 게임 클래스의 분류 방법에 관한 연구", 산학연 멀티미디어 산업기술 학술 대회, pp. 56~61, 1998.

[9] J.F.Allen, "Maintaining Knowledge about Temporal Intervals", CACM, Vol.26, No.11, pp. 832~843, Nov, 1993.

[10] Seong Bae Eun, Eun Suk No, "Specification of Multimedia Composition and A Visual Programming Environment", ACM Multimedia, pp. 167~173, 1993.

[11] Simon J.Gibbs, Dionysios C.Tsichritnis, "Multimedia Programming", Adison-Wesley Publishing Co. 1997.

[12] Robert Sedgewick, "Algorithms in C", ADDISON WESLEY, 1991.

[13] 김갑수, 신영길, "소프트웨어 재사용을 위한 C++ 클래스 계층 구조 변형 방법", 정보과학회 제22권 제1호, pp. 88~96, 1995.

※ 본 논문은 1998년 정보통신연구관리단 대학기초 연구지원 사업으로 수행되었음.