한국게임학회 논문집
제2권 제1호 2002년 5월

격투게임 제작 기법과 알고리즘에 관한 연구

주정규
(주)서울게임대학 학장

A Study on Technique of Fighting Game Producing and Game Algorithm

Jung-Kyu Joo

요 약

본 논문은 격투게임의 기본적인 게임제작 기법과 격투자간 충돌처리 기법, 격투의 기(技:Skill) 처리, 적 캐릭터의 사고루틴 및 사고루틴의 처리방법, 격투게임의 알고리즘 구조에 관하여 고찰하였으며, 격투게임에 많이 적용하고 있는 디자인패턴 클래스인 Template Method와 Strategy Method 기법의 알고리즘에 관해서도 고찰 연구하였다.

Abstract

This paper describes basic producing techniques of fighting game, collision detection between fighters, fighting skill, thinking routine and processing method of enemy characters, and algorithm structure of fighting game. Also, I studied as for algorithm of the template and strategy methods of design pattern class applied to a fighting game.

***Key words** : fighting game, thinking routine, collision detection, demage check, fighting skill, the template method, the strategy method*

1. 서론

격투게임은 캐릭터와 캐릭터간 또는 캐릭터 그룹간의 주먹치기(Punch), 발차기(Kick), 머리치기(Butt) 등의 다양한 동작을 서로 교환하면서 진행하는 액션게임 중의 하나이다. 본 논문에서는 격투게임의 개요와 격투게임의 특징 및 격투 게임제작의 기본적인 기법에 관하여 기술하며, 컴퓨터 측면이나 상대방의 움직임을 결정하는 상대방(적)의 이동이나 동작에 필요한 캐릭터의 사고루틴, 상대방인 적에게 맞거나 공격을 당했을 때의 충돌판정, 상호간 공격 및 방어의 경우 상호간의 데미지 판정, 주먹치기(punch)나 발차기(킥:kick), 필살기 등의 각종 기(技:skill)의 처리, 격투게임의 데이터구조, 적 캐릭터의 사고루틴 알고리즘, 전체적인 격투게임의 알고리즘과 게임의 흐름구조 등에 관하여 기술한다.[8] 또한, 코드 재사용(code reuse)을 위하여 디자인패턴 클래스 사용 기법인 the Template Method와 the Strategy Method 기법을 격투게임에 적용하는 알고리즘에 관하여 고찰한다.[1] 본 논문의 구성은 2장에 격투게임의 개요를 설명하고, 3장은 격투게임 제작의 특징인 캐릭터의 이동, 충돌판정, 가격시의 데미지의 판정, 격투의 기(技) 처리, 캐릭터의 사고루틴을 알아보고, 4장에서는 격투게임의 제작 알고리즘의 일부분인 데이터의 저장 알고리즘, 적 캐릭터의 사고루틴, 격투 게임의 전체적인 알고리즘의 흐름구조와 기의 판단 방법 등에 관해서 고찰한다.

2. 격투게임의 개요

격투게임은 두 사람 이상의 캐릭터가 일정한 배경화면에 등장하여 싸우는 개념의 게임이다. 격투기에는 '주먹치기(punch)', '발차기(kick)', '머리치기(butt) 등의 다양한 격투기술이 있으며[5][7], 두먹치기와 발차기 및 머리치기 등을 이용한 다양한 응용 동작이 있다. 또한 기본동작으로 전진과 후퇴, 엎드린 자세의 전진과 후퇴, 점프, Step in/out, 횡이동과 종이동, 역회전 등의 동작이 있다.[5]

공격동작에는 왼손 주먹치기, 오른손 주먹치기, 왼발 차기, 오른발 차기, 상단공격, 중단 공격, 하단 공격 등이 있으며, 응용공격 동작으로 점프차기, 연속차기, 태클, 슬라이딩 공격기(技), 어깨공격, 방어불능의 공격(필살기) 등이 있다. 또한, 방어동작에는 서서 방어하기(상단공격의 방어자세), 허리 엎드려 방어하기(하단공격의 방어자세) 등이 있다.[7] 기타 공격과 방어의 응용동작으로는 다운공격(상대방 다운시, 점프하여 체중을 실어 위에서 아래로 공격하기), 일어서면서 공격하기, 앞으로 돌고 뒤로 도는 공격준비자세, 옆으로 돌아 공격하기, 하단 및 중단 발로차기, 스프링식 발로차기(Spring kick), 스프링크로스 발로차기 등 다양한 공격과 방어의 격투 기(技:skill)가 있다.[5]

또한, 격투게임은 본래 액션게임에서 파생된 게임의 한 장르로서 인간의 격투기(태권도, 쿵푸 등)를 컴퓨터(전자)게임화 한 것이다. 기본적인 격투게임의 화면표시 시점은 캐릭터나 배경을 세로 측면에서 본 상태로 처리하며, 캐릭터가 화면의 끝에 도달하게 되면 배경화면을 스크롤시켜 게임을 진행한다. 이와 같은 스크롤기능은 롤플레잉 게임의 화면처리와 매우 유사하며, 많은 적 캐릭터가 연속하여 공격할 때 적 캐릭터를 쓰러뜨리는 패턴은 슈팅게임과 유사한 구조를 가지고 있다. 초기에 제작된 대부분의 격투게임의특징은 단순한 격투의 기(skill)인 발차기, 주목치기 만을 사용하였으나, 현재는 태권도, 권투 및 유도 경기와 같이 '시간제한'과 '라운드 제도' 등을 추가하고 있는 추세이다.

2.1 격투게임의 재미와 매력

격투게임의 재미 요소는 적 캐릭터를 쓰러뜨리는 쾌감을 제공하는 게임으로, 권투나 태권도와 같은 격투를 재현하여 대리만족을 제공하는 게임이다.

이미 보편화된 슈팅게임과 비교 고찰해 보면, 격투게임의 1대1 격투는 슈팅게임의 적 캐릭터와의 싸움에 해당되며, 슈팅게임의 '클라이막스'를 바로 즐길 수 있는 것이 격투게임의 특징이기도 하다. 일반적으로 슈팅게임에서는 캐릭터 자체에 인기는 별로 없으나, 격투게임에서는 캐릭터가 절대적인 인기를 차지하고 있으므로, 캐릭터 상호간의 격투에서 승자가 된 '영웅'캐릭터로부터 플레이어에게 감정이입을 제공하게 된다.[5]

또한, 격투게임은 단순한 액션게임보다 강한 공격성을 갖게 함으로써 감정이입을 높이는 결과를 초래하므로, 감정이입을 증가시키는 요소의 설정이 중요한 역할을 한다. 예를 들면 '철권 시리즈' 게임의 경우와 같이 스토리를 갖는 격투게임도 있다. 즉, '각 캐릭터는 각각 특별한 목적을 가지고 싸움을 반복해 가면서 그 목적을 달성시킨다'라는 내용이 많은 부분을 점유하고 있다. 또한 격투의 기(技:skill) 처리 방법도 다양화되고 있으며, 기와 기를 조합하면 보다 강력한 공격이 가능하며, 체력이 부족할 경우에는 '필살기'와 같은 공격방법으로 기(技)를 자주 사용한다.[7]

2.2 다양한 게임요소가 결합된 격투게임

단순히 '적을 쓰러뜨린다'는 격투게임의 재미성과 사실성을 인정하기는 어려우며, 다양한 즐거움이 복잡하게 조합되어 있는 것이 격투게임의 핵심적인 포인트이다. 과거부터 인기 있었던 격투게임은 이러한 점을 잘 활용하고 있으며, 현재 발매되고 있는 격투게임도 '스토리', '캐릭터', '화려한 배경화면' 및 '기(技)의 결합(Combo)과 난이도'라는 측면에서 차별화를 시도하고 있다. 특히 즐거움을 제공하는 요소로는 '사실적인 배경화면 처리'와 '캐릭터의 공격 및 방어자세의 움직임'이나, 어색하게 처리되면 감정이입은 발생하기 어렵고 게임성도 감소한다는 사실이다.[5]

또한 플레이어의 분신과 같은 캐릭터가 격투하는 장면에서 캐릭터 자신의 신변위험을 체험하는 리얼리티는 3차원 그래픽으로 처리해야 실감을 느낄 수 있다. 최근에 발매되고 있는 대부분의 격투게임은 3차원 그래픽으로 제작하고 있는 추세이나, 본 논문에서는 2차원 그래픽을 사용하였으며 화면표시에 관한 그래픽처리를 제외한 '캐릭터의 사고루틴'은 2차원 그래픽이나 3차원 폴리곤그래픽은 동일한 구조를 가지고 있는 특징이 있다. 인기 있는 게임으로는 '철권' 시리즈, '버쳐파이터'시리즈, '스트리트파이터'시리즈, '더킹오브파이터'시리즈 등 대작게임이 많이 있다.

3. 격투게임 제작의 기본구조

실제로 화면에 표시되는 배경이나 오브젝트의 내용을 통하여 프로그램의 움직임을 고찰한다. 격투게임의 화면 구성은 대체로 '캐릭터'를 표현하는 그래픽파트와 '배경'으로 나뉘어져 있다. 실제 격투게임의 화면은 파트와 배경을 합성한 것이며, 프로그램의 처리도 각각 별개로 다루고 있다.

3.1 캐릭터의 이동

이동키를 누르면 캐릭터가 지정된 방향으로 움직이며, 전후좌우 사방으로 움직이면 캐릭터는 걷는 그래픽을 표시한다. 이것을 프로그램에서 데이터로 변경시키려면 파트의 '캐릭터 애니메이션 데이터'와 각각의 키에 대응하는 형태로 되어있으므로 미리 캐릭터 애니메이션의 움직임을 지정하기 위해서 파트를 어느 순번으로 표시할 것인가를 결정하는 '스크립트 데이터'도 필요하게 된다.

3.2 충돌의 판정

적 캐릭터에게 격투를 당했거나 필살기가 통했다면, 체력게이지가 얼마만큼 감소하였으며, 격투의 가격여부를 조사하기 위해서 '충돌판정'을 하고 있다.(그림 1.참조) 충돌판정의 방법은 여러 가지가 있으며, '스프라이트 색채를 이용하여 판정하는 방법', '캐릭터가 이동하는 최소량의 이동량 구역을 나누어 일정단위의 칸단위로 캐릭터를 투영시켜 판정하는 방법', '캐릭터가 위치한 좌표와 크기를 기준으로 겹침의 여부를 판단하는 방법'[4] 또는 '수학적으로 물체가 겹치고 있는가를 판단하는 방법' 등과 같은 방법이 있으며, 그림.1은 전자의 3가지의 방법을 그림으로 표현한 것이다.(그림.1 참조)

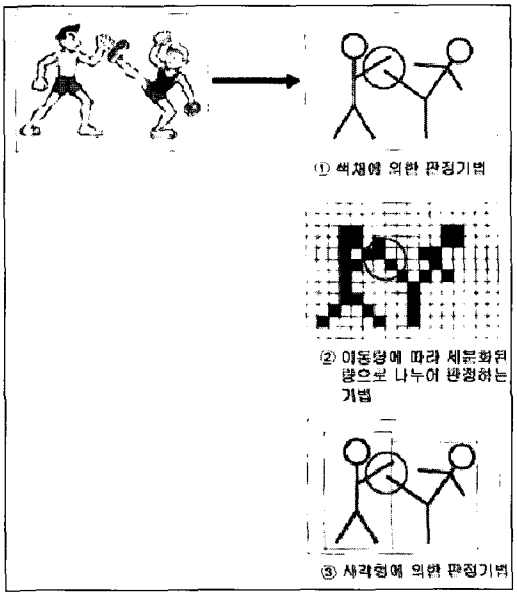


그림 1. 격투게임에서 많이 사용하는 충돌 판정 기법

3.3 데미지의 판정

격투시 상대캐릭터에 얼마정도(수치로 표현함)의 데미지(손상)를 주었는가? 라는 값은 공격종류와 대응하는 테이블(파라미터 테이블)을 사용하여 얻으며, 격투하는 캐릭터가 바뀐 경우에도 참조하는 테이블을 바꾸어서 각 기(技:skill)에 대응하는 데미지의 값을 코드부분에서 간단하게 얻어 처리한다. 키 코드의 대응부분도 테이블로 관리하며, 데미지를 받은 캐릭터의 체력 게이지가 0이 되면 게임은 종료한다.

3.4 기(技:skill)의 처리

격투기 게임의 측면에서 '기의 처리'와 '컴퓨터 측면의 캐릭터의 사고루틴'을 알아본다. 기(技)의 처리는 이동기술과 같이 '스크립트'로 처리하여 데이터를 생성하며, 펀치, 킥의 처리는 한가지 키를 사용하여 처리하며 이동기술과 같이 처리한다. 키를 누르면 각 캐릭터의 애니메이션 패턴이 연속하여 표시되며, 기를 사용하는 도중에 다른 기를 사용하면 기의 애니메이션 패턴이 전부 나올 때까지 대기한다. 필살기 등 특수 키 조작의 경우는 '복수개의 키를 동시에 눌렀을 때'의 처리방법으로 키의 읽기를 키가 눌린 시점에서 판단하는 것이 아니라, 특정 시간단위로 읽어 처리한다.

예를 들면, '펀치', '킥', '펀치' 등의 순서로 연속하여 키를 눌러 필살기를 구현할 경우에는 3가지 동작의 각 시간에서 필요한 키가 눌려져 있는가를 확인하면 된다.(그림 2. 참조)

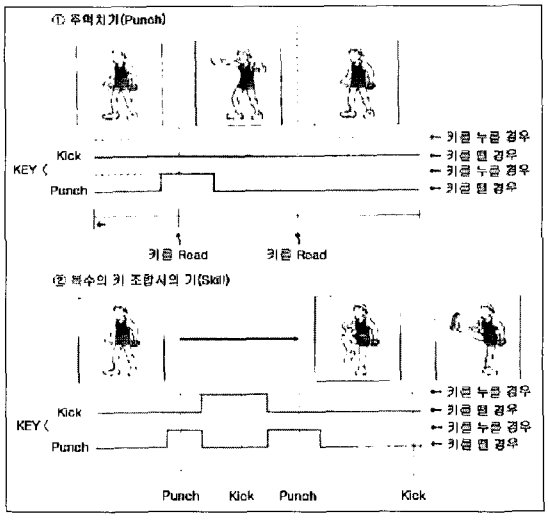


그림 2. 시간단위의 키 조합 알고리즘

그림.2는 시간단위의 키 조합 알고리즘으로 키보드를 1회 누르면 한 번의 이벤트가 발생하는 알고리즘이다. 이 때 입력장치를 시간매개로 처리하는 것은 상당한 어려움이 있으므로, 사용하는 키의 수만큼의 테이블을 준비해야 하고, 테이블의 각 데이터가 각 키에 대응하는 키가 '눌러졌을 때'와 '누르지 않았을 때'로 구분하여 테이블의 값을 변화시키며, 하나의 키가 눌러질 때마다 동시에 눌러져있는 키가 있는가 없는가를 조사한다. 이러한 키 상태가 연속하면 키가 눌러 있지 않음을 확인한 후 다음의 키 조합을 조사하며, 같은 방법으로 몇 번 조정하여 처리한다.

3.5 사고(思考) 루틴

슈팅게임의 적 캐릭터의 움직임은 출현하는 '위치'와 '시간'에 따라 '정해진 움직임과 방향' 등을 새롭게 설정해 두어 처리하였으나[6], 격투게임에서는 컴퓨터가 제어하는 적 캐릭터의 동작은 인간의 생각을 프로그래밍 하는 사고루틴이 절대적으로 필요하다. 컴퓨터 측의 캐릭터 움직임을 결정할 때는 '상대의 이동이나 동작'에 비례하는 형태가 되며, 기본적으로 '무엇인가'가 발생하면 '어떻게 대처한다'와 같은 사고를 각 조건에 따라 분기한 형태로 처리하게 된다.

이는 '무슨 조건이 어떤 만족을 제공하고 있는 것인가에 관하여 어느 쪽을 우선하고 있다'와 같은 비례조건을 만들기 위하여 데이터로 정의하여 처리한다.

4. 격투게임의 제작 알고리즘

화면표시에 사용하는 파트 데이터는 '파트용 그래픽 데이터의 포인터'를 배열형태로 처리하고 있으며, 기(技) 등의 스크립트 데이터는 키 데이터와 대응시켜서 처리한다. 단일 '펀치'의 경우에는 '0'을, '펀치' =)'킥' =〉 '펀치'와 같은 필살기의 경우는 '1'을 지정하여 '키로부터 기(技)의 정수'를 작성하는 테이블을 준비한다. 이 값을 기본 값으로 '기에 대응하는 각종 데이터'를 결합시킨 테이블을 작성하며, 스크립트 데이터를 작성하여 처리한다.([프로그램 리스트-1] 참조)

```
// 기(技)-데이터의 저장

// 형 선언
typedef struct { // 기(技)의 정의
word Num; // 기(技)의 종류
word Damage; // 상호간의 데미지
pTScriptItems Script; // 스크립트 포인트
char Caption[20]; // 기(技)의 명칭
} TSkillItem;

// 정수 선언
#define CharacterPunch 0
#define CharacterKick 1
#define CharacterAskill 2
#define CharacterBskill 3
#define CharacterCskill 4
#define CharacterSkillLimit 5

static TSkillItem SkillItem[CharacterSkillLimit + 1] = {
{CharacterPunch, 2, &scrPunch, "펀치"},
{CharacterKick, 2, &scrKick, "킥"},
{CharacterAskill, 5, &scrAskill, "A 기술"},
{CharacterBskill, 8, &scrBskill, "B 기술"},
{CharacterCskill, 10, &scrCskill, "C 기술"},
};
```

[프로그램 리스트-1] 기(技) 데이터의 저장 알고리즘

4.1 적 캐릭터의 사고루틴

적 캐릭터의 사고루틴은 '현재의 상황'에서부터 '다음에 취해야 하는 행동'을 얻는 일종의 테이블과 같은 형태이며, '현재의 상황'이 되는 '키'는 '상대와의 거리'와 '현재의 체력이나 시간'을 표현한다. 랜덤한 특정의 기(技)를 사용하는 경우는 하나의 키를 설정하여 처리하며, '상호간 특정의 공격을 받을 경우에 대응하는 행동'이나, 행동을 실행하지 않는 '견제와 같은 행동'의 경우도 키에 설정하여 처리한다. 키에 설정된 기는 '킥으로 공격', '이동하기', '점프'와 같은 구체적인 행동의 내용이 되며, 캐릭터가 취할 수 있는 행동의 수만큼 키로부터 설정된 기(技)가 되기도 한다.

본 논문에서 구현한 방법은 우선 각각의 행동패턴을 데이터화하고 그 데이터에 대하여 하나의 파라미터를 갖게 한다. 이 파라미터의 가능은 '무엇을 하고 싶은가'에서 '의사

(意思)'를 지칭하는 것을 고려하면 그 개념이 명확해진다.

키의 값에 따라서 각 의사의 크기(파라미터)를 증감시켜 의사가 보다 큰 것이 최종적으로 다음에 취해야만 하는 행동이 되며, 행동 자체는 스크립트로 처리하여 정의하고, 테이블에 정의하여 처리한다.(그림 3. 참조)

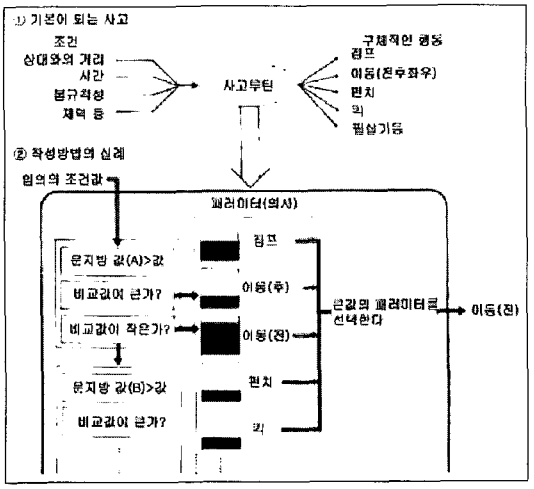


그림 3. 사고(思考)루틴의 처리방법

(1) 문지방(threshold) 값에 의한 조건판단

각 조건의 판단처리는 '의사'가 되는 파라미터를 조작하여 처리하며, 단순하게 어느 한가지 조건을 기본으로 각각의 의사를 바꾸는 것과 같다. 이들 조건 판단에는 '문지방 값'이 존재하기도 한다. 예를 들면, '상대 캐릭터가 근접해 있는가의 여부'와 또한 '그 접근이 근접인가 아닌가'를 판단하기 위한 값이 필요하다. 이때의 값은 10 또는 5의 값과 같은 여러 가지의 정수가 되며, 어떤 값을 '문지방 값'으로 정하여 그 값 이하이면 '근접', 이상이면 '근접하지 않음'으로 판단할 수 있다. 특히, '근접' 보다 '걸어서 접근' 또는 '점프' 등의 의사 차이를 표현하는 것이 가능하며, 판단하고 싶은 조건의 수만큼 '문지방 값'을 준비해 두고, 이 값을 게임중이나 캐릭터에 변동시킴으로써 캐릭터의 개성 등을 보다 인간적인 움직임으로 표현한다.

조건판단 처리는 각 조건마다 함수로 작성하며, 게임 장면에 따라서는 필요하지 않은 조건도 나타나게 되는데, 그럴 경우에는 필요하지 않은 조건판단처리의 그 자체를 제외시킨다. 이의 처리는 함수 포인터를 사용하여 나열한 리스트의 형태를 취하며, 조건을 더하거나 제외하고자 할 때는 이 리스트를 조작하여 처리한다. 데이터를 처리하는 부분은 포인터를 연속하여 지정하여, 리스트가 끝난 시점에서 처리를 멈추게 한다.

(2) 캐릭터에 따른 강약의 부여

위와 같은 방법은 조금은 까다로운 방법으로 만약 속도면에서 문제가 되므로 단순한 방법을 사용하며, 만약 캐릭터마다 강함과 약함이라는 것을 나타내고자 할 경우에는 사고루틴' 측의 최적화도 필요하나, '판단의 시간적 느림과 빠름'이라는 '반응속도'로 차이를 나타낼 수가 있다.

또한, 캐릭터의 동작이 느린 경우에는 늦은 만큼의 플레이어의 공격을 쉽게 받게되므로 결과적으로 '약한 캐릭터'가 되며, 역으로 캐릭터의 동작이 빨라지면 플레이어 측이 공격을 피하기 어려워지므로 '강한 캐릭터'로 남게 되어 강력한 캐릭터로 군림하게 된다.[1]

4.2 격투게임의 알고리즘

여기에서는 격투게임의 전체적인 게임프로그램의 흐름과 격투게임의 개념을 알아본다.(그림 4. 참조) 격투게임에서 사용하는 '타이머'(timer)를 정기적으로 기동하며, 타이머의 시간 단위로 게임의 동작처리를 연속적으로 진행한다. 캐릭터의 표시는 시간마다 파트를 전환함에 따라서 애니메이션 효과를 나타내고 있으며, 적 캐릭터의 움직임이나 각종 스크립트의 읽기도 여기에서 수행하게 된다. 현실과 같은 움직임을 요구하는 격투게임은 빠른 처리속도를 엄격하게 요구하게 되며, 속도를 향상시키기 위한 방법으로 대부분은 테이블을 많이 사용한다.

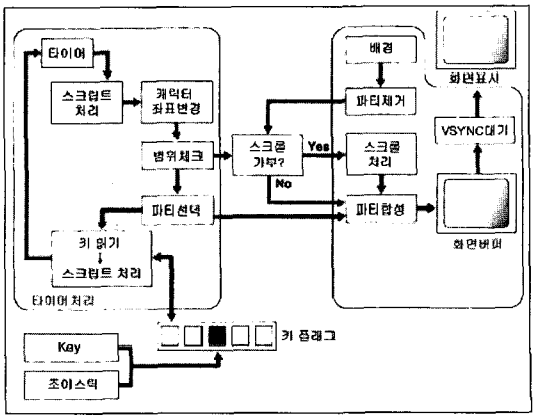


그림 4. 격투게임 알고리즘의 흐름구조

(1) 기(技)의 판단방법

키-코드의 입력은 화면 바뀜과 함께 입력하면, 짧은 시간에 필요한 키를 모두 눌러야 하므로, 어느 정도 '읽기 시간'을 갖도록 해야 한다. 실제 '타이머가 4회의 기동으로 카를 읽는다'와 같은 내용으로 처리를 하며, 읽은 키는 각각 플래그라는 변수 형태에 저장한다. 기(技)로 사용되는 키-조합의 정의는, 키-사용 구조체인 '링크드 리스트'를 사용한다. 그러므로 기의 판단방법으로 '각 단계에서 공통인 기'를 모아 '맞는 키가 없음'의 단계에서 '기의 스크립트'를 나무구조로 하면 기 판단의 탐색이 편리하게 된다. 하나의 캐릭터가 갖는 기(技)는 많지 않으므로 데이터화를 쉽게 할 수 있으며, 탐색 방법은 최초에 누른 키로서 해당-키에 대응하는 키를 리스트의 맨 앞의 항목으로부터 탐색한다.

만일 탐색을 완료한 경우에는 그 항목의 포인터를 보존한다. 다음의 키를 누를 때에는 항목에 있는 키-포인터에 대응하는 것을 처음과 같이 동일하게 보존하며, 이렇게 하여 키를 누를 때마다 포인터를 체크하여 포인터가 없는 경우에는 그때의 항목에 정의되어 있는 기를 실행한다.(그림 5. 참조). 기의 키-조합이 잘 맞지 않거나, 키-누름의 간격이 빈 경우에는 현재 누른 키에 대응하는 동작을 실행하여 탐색에 사용하고 있는 포인터는 파기하여 처리한다.

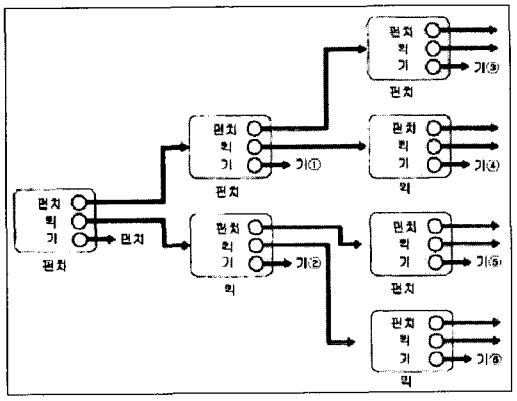


그림 5. 펀치-키로 시작하는 기(技)의 나무구조

예를 들어, (그림. 5)에서 '펀치' => '펀치' => '펀치' 순으로 키를 누르면 기(技)(3)이 되며, '펀치' => '킥' => '킥' 순으로 누르면 기(技) (6)를 출력하게 된다.

(2) 충돌의 판정방법

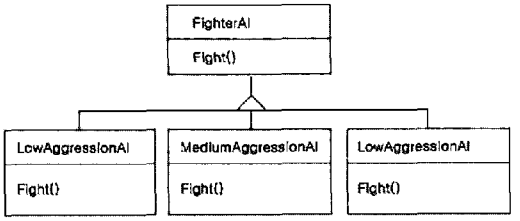
슈팅게임에서 많이 사용하는 '이동량의 판단에 따른 충돌방법'을 설명하면 각 캐릭터가 있는 위치에 상당하는 부분을 '충돌판정용 버퍼'에 대응하는 장소에 값을 입력한다. 만약 8-도트 단위로 동작하는 하는 경우, 640×480의 화면에서 80×60=4800바이트의 메모리 용량에 불과하므로, 이 정도의 캐릭터에 따른 클리어시간은 별로 문제되지 않는다. 그러므로 충돌 판정시 충돌판정용 버퍼에 값을 넣을 때에는, 이미 그 버퍼에 값이 저장되어 있는지의 여부를 수행한다. 최근 3D 폴리곤그래픽 게임의 경우는 수학적으로 물체가 충돌한 상태의 여부를 판단하는 경우를 많이 채용하고 있다.

4.3 격투게임에 응용한 Template 및 Strategy Method

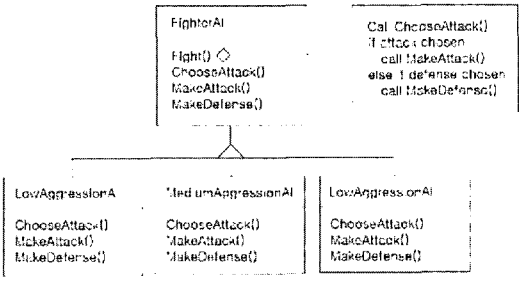
격투게임에 적용하고 있는 이 두 가지 기법은 근본적으로 유사한 기법으로, Template 기법은 전체 알고리즘의 동적 선택을 허용하는 반면, Strategy 기법은 미세한 량의 교체를 허용하며, 알고리즘의 개별적인 파트를 교체하게 된다. 예를 들면, 격투게임에서 컴퓨터제어의 캐릭터는 격투 유형을 다르게 하기 위하여 템플리트 기법을 사용한다. 이 알고리즘 기법은 모든 컴퓨터제어 캐릭터에 사용되고 있는 격투자 제어클래스를 구현하기 위하여 허용하고 있으며, 개별적인 캐릭터의 주문을 런-타임으로 하는 새로운 격투 전략을 첨가하는 기능도 있다. 또한 이 알고리즘은 프로젝트 내에서 코드의 재사용을 증진시킬 수 있으며, 정의된 모듈 구획 단위에 코드를 효율적으로 분리하여 적용할 수 있는 기능을 가지고 있다. 그림 6.은 이 알고리즘의 동작 흐름을 나타낸 그림이다.

격투게임의 경우 Strategy Method는 공격의 다양한 레벨의 격투 스타일을 허용하며 보다 많은 모듈의 제공을 확장할 수 있으나, 격투 스타일당 많은 클래스를 산출하게 되므로 유지보수에 다소 어려움을 줄 수 있다는 단점이 있다. 또한 서로 다른 격투 스타일을 Strategy Method를 사용하여 선택할 수 있으며, 격투 스타일의 개별적인 변경은 그림 6. ②와 같은 Template Method를 사용하여 적용한다. Template Method는 기본적인 클래스를 하나의 알고리즘의 스탭 순서로 정의하며, 도출된 클래스는 알고리즘의 개별적인 스탭을 자유롭게 변화시킬 수 있는 장점이 있다.

이들 알고리즘을 구현하기 위해서는 하나의 알고리즘을 여러 개의 파트로 구분하며, 각 파트는 가상함수를 이용하



① 격투게임에 적용한 Strategy Method 클래스



② 격투게임에 적용한 Template Method 클래스

그림 6. Template 및 Strategy Method의 클래스 구성

여 구현한다. 이 파트는 알고리즘 그 자체가 기본 클래스에서 사용되고 있어도, 각 알고리즘의 각 상황에 따라서 서브클래스를 허용한다. 이 알고리즘의 상세 동작의 실례로 [프로그램 리스트-2]에 설명하고 있다.

```
class CGraphicObject
{
public :
// 템플리트 기법인 Draw( )함수는 그래픽 오브젝트 드로잉
// 알고리즘을 위한 골격을 정의하며, 서브클래스에 의해서
// 정의되는 개별적인 스탭을 요구한다.
bool Draw( )
{
  if (true == NeedToDraw() && true == CheckAndSetZBuffer() )
{
return InsertIntoDrawList() ;
}
return false ;
}
// Draw 리스트에 이 토큰을 삽입한다
void InsertIntoDrawList() ;
protected :
virtual bool NeedToDraw() = 0 ;
virtual bool CheckAndSetZBuffer() = 0 ;
} ;
```

[프로그램 리스트-2] Template Method의 사용 예

위의 [프로그램 리스트-2]는 그래픽-렌더링 프레임워크로 유도된 각 클래스는 2개의 가상함수인 CheckAndSetZBuffer()와 NeedToDraw()를 지배하는 기회를 갖게 된다. 단, 여기에서 NeedToDraw() 함수는 모양과 크기의 2개의 가상함수의 투명성과 그리는데 필요한 다른 오브젝트를 고려하기 위하여 기본 클래스인 Draw()를 호출하여 처리하고 있다.

5. 결론

본 논문은 국내에서 현재 활발하게 제작되고 있지 않는 게임장르인 격투게임의 기본적인 제작기법, 적-캐릭터의 사고루틴 알고리즘, 충돌체크 알고리즘 루틴, 격투의 기(skill)처리 루틴, 주먹치기(punch)와 발차기(kick) 등의 상호간 충돌판정 기법의 고찰을 통하여 격투게임 개발의 전체적인 흐름을 이해하게 하므로 격투게임의 개발과 제작에 큰 도움을 줄 수 있으리라 판단되며, 격투게임의 프로그래밍에 필요한 기본적인 격투 알고리즘의 이해에 도움을 줄 수 있을 것으로 사료된다. 또한 격투게임에서 많이 사용하는 Template Method 와 Strategy Method의 알고리즘을 적용한 실례를 연구 고찰하므로 이 알고리즘을 적용한 격투게임의 개발에도 도움을 줄 수 있다고 판단된다.

향후 연구과제로는 이들 알고리즘을 활용한 실제 격투게임의 개발에 필요한 격투게임 개발-툴을 개발하는 것이며, 이들 툴을 적용한 상품화된 실용적인 격투게임의 개발에 적용하는 것이다. 또한, 격투게임에 활용할 수 있는 고급 인공지능의 사고루틴과 고급알고리즘을 연구 개발하여 국기인 태권도 등의 격투게임을 개발하여 국내 격투게임 시장을 개척하는데 앞장서야 하겠다.

참고문헌

[1] Andrew Rollings & Dave Morris, "Game Architecture and Design", Coriolis Group, 2000.

[2] Edited Marc saltzman, "Game Design -secrets of the sages", Brady Publishing, 1999.

[3] "Playstation game catalog", 日本 藝文社, 1997.

[4] 이영재, "게임을 위한 2D 충돌감지 알고리즘 비교분석에 관한 연구" 한국게임학회 논문지, 제1권 1호, pp. 43-45, 2001.

[5] 남코사 발행, '철권 태그토너먼트' 격투게임 메뉴얼, 2000.

[6] 주정규, "게임알고리즘의 기초", 서울게임대학 간이교재, 2001.

[7] 철권 시리즈 게임타이틀 매뉴얼, 일본 NAMCO사, 2001.

[8] 有馬 元嗣, 게임프로그래밍, Softbank, 1997.

[9] 게임파워 "The King of Fighters 99", 공략 매뉴얼, 1999.

[10] http://www.namco.co.jp

[11] http://www.sega.co.ip

[12] http://www.gamasutra.com

주 정 규

1979 숭실대학교 전자공학과 학사
1984 단국대학교 대학원 전자공학과 공학석사
1992 Pacific Western Univ. Computer Engineering 공학박사
1993-1999.2 숭의여자대학 전자계산학과 및 컴퓨터게임과 교수
1999.3-2000.12 청강문화산업대학 컴퓨터게임과 교수
2001.1-2002.현재 서울게임대학 학장
관심분야 : 게임컨텐츠창작, 게임기획 및 게임디자인, 게임프로듀싱, 게임알고리즘, 게임인공지능