게임 캐릭터 동작에 관한 설계 기법 비교 분석

임종선, 이대원[1)]
서경대학교 컴퓨터공학과
e-mail : {wwwlim, daelee}@skuniv.ac.kr

To Compare and Analysis Design Method about Motions of Game Character*

Jong-Seon Yim, Daewon Lee
Dept. of Computer Engineering, Seokyeong University

요 약

그래픽, 물리엔진 등의 다양한 라이브러리의 발전은 게임의 품질을 높이는 반면, 과도한 결합도, 클래스 정체성 상실 등의 문제를 발생시킨다. 이러한 문제를 해결하기 위해 본 연구에서는 게임 설계시 다양한 요소들 중 하나인 캐릭터 동작 설계에서 디자인패턴 중 상태 패턴 기법, 명령 패턴 기법 그리고 캐릭터 모터 기법을 사용하여 대기/걷기/뛰기/공격 동작을 구현하였고, 구현한 각 동작을 5000프레임씩 실행하여 프레임당 평균 소요시간과 메모리 평균 사용량을 비교 평가하였다.

1. 서론

그래픽, 물리엔진 등의 다양한 라이브러리들은 빠른 발전을 이루고 있고, 이는 게임엔진에서도 영향을 끼치고 있다. 이로 인하여 이전의 2D 그래픽 게임을 넘어 3D, AR(Augmented Reality), VR(Virtual Reality)까지 발전하였다.

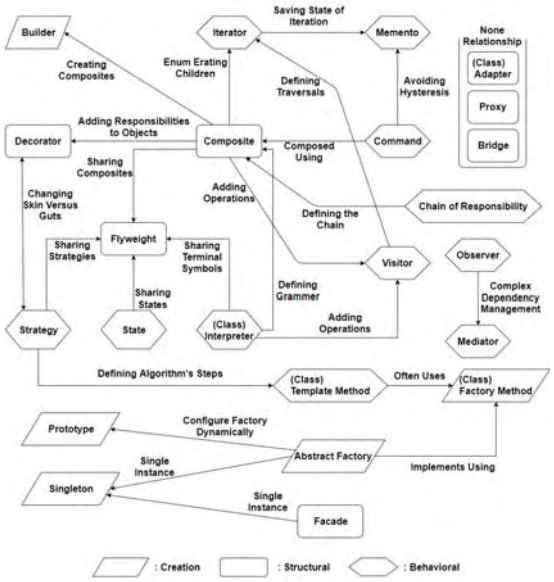
하지만 이러한 발전은 애니메이션, 그래픽 디자인, 클라이언트 개발 등 다양한 분야에서 개발 난이도의 급격한 상승을 유발하였다. 이러한 급격한 상승은 과도한 결합도, 불완전성, 기능의 중복성, 클래스 정체성 상실 등의 다양한 문제를 발생시킨다.

이러한 문제들을 해결하기 위해 본 연구에서는 디자인 패턴 중 상태 패턴 기법과 명령 패턴 기법 그리고 Unity Technologies의 캐릭터 모터 기법을 기반으로 대기/걷기/뛰기/공격의 4가지 행동을 Unity3D 엔진에서 구현하고, 구현에 적용된 기술간 프레임당 소요시간과 메모리 사용량을 비교 분석하여 캐릭터 동작 구현에 가장 적합한 기술을 찾아낸다.

2. 배경

가. 배경 기술

디자인 패턴의 기반이 되는 패턴 언어는 건축 구조 설계를 위하여 건축가 크리스토퍼 알렉산더가 처음 제안하였다[1]. 이후 객체 지향 프로그래밍 개발에 패턴 언어를 적용하려는 다양한 연구가 진행되었고[2], 앞서 연구된 패턴 언어는 소프트웨어 개발에 있어 디자인 패턴으로 정립되었다[3]. [3]에서 제시한 23가지 패턴 및 관계도는 (그림 1)과 같다.



(그림 1) 디자인 패턴 관계도

본 연구에서는 (그림 1)의 디자인 패턴 중 상태 패턴 기법과 명령 패턴 기법 그리고 Unity Technologies의 캐릭터 모터 기법을 사용하여 캐릭터 동작을 구현한다.

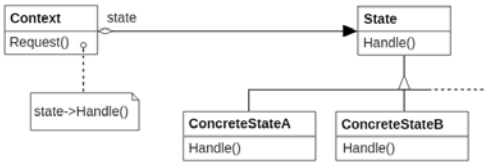
1) 교신저자

* 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2016R1C1B1008330)

나. 상태 패턴 기법

[3]에서는 상태 패턴 기법을 “객체의 내부 상태에 따라 스스로 행동을 변경할 수 있게 허가하는 패턴으로, 이렇게 하면 객체는 마치 자신의 클래스를 바꾸는 것처럼 보입니다.”라고 정의한다.

상태 패턴 기법은 FSM(Finite State Machine)으로부터 시작된다. FSM은 ‘상태’와 ‘전이’로 구성된다. 특징으로는, ‘한 번에 한 가지 상태만 가능’, ‘입력에 따른 상태 전이 보유’ 등이 있다. 하지만 상태가 세분화 되고 늘어나게 될수록 무수한 전이가 생기는 한계가 있으며, 이러한 한계를 극복하기 위하여 병행 상태 기계, 계층형 상태 기계, 푸쉬다운 오토마타 기법 등을 적용할 수 있다[4].

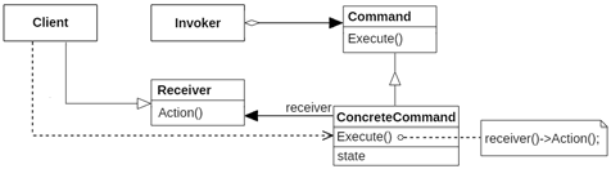


(그림 2) 상태 패턴 기법 표준 UML

다. 명령 패턴 기법

[3]에서는 명령 패턴 기법을 “요청 자체를 캡슐화하는 것입니다. 이를 통해 요청이 서로 다른 사용자를 매개변수로 만들고, 요청을 대기시키거나 로깅하여 되돌릴 수 있는 연산을 지원합니다.”라고 정의한다.

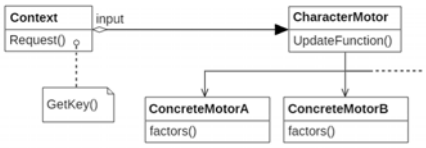
명령 패턴 기법의 가장 기본적인 구조의 요소에는 Command, Receiver, Invoker, Client가 있다. 특징으로는, ‘요청 객체와 수행 객체간의 느슨한 결합’, ‘요청 수행 시점 선택 가능’, ‘수행 명령 기록을 통한 Redo/Undo 제공’ 등이 있다. 하지만, 명령 패턴 기법의 수행 명령 기록은 Undo, Redo시 발생할 수 있는 일관성 위배는 이력현상을 발생시킨다. 이러한 문제를 극복하기 위하여 메멘토 패턴 기법을 적용할 수 있다.



(그림 3) 명령 패턴 기법 표준 UML

라. 캐릭터 모터 기법

캐릭터 모터 기법은 동작에 대한 데이터를 객체화한다. 각 동작에 대한 데이터를 객체로 관리하고, 입력 발생시 객체 내 데이터를 기반하여 동작을 수행한다.



(그림 4) 캐릭터 모터 기법 표준 UML

캐릭터 모터 기법은 Rigidbody[5]의 중력 효과와 지면에 대한 충돌 효과, 그리고 유동 객체와의 상호작용을 보조하는 Move Platform을 제공한다.

마. 기술 비교 분석

<표 1> 기술간 요소 비교 분석

기술 / 요소	상태 패턴	명령 패턴	캐릭터 모터
요청 수행 캡슐화	가능	가능	불가능
Undo/Redo	불가능	가능	불가능
요청 수행 시점 선택	불가능	가능	불가능
유동 물체와의 상호작용	불가능	불가능	가능
중력 효과 제공	불가능	불가능	가능
연계 동작 제한	쉬움	쉬움	어려움
특수 동작 전환	어려움	어려움	쉬움
요청간 결합도	낮음	낮음	높음
동작 결정 위치	내부	외부	내부

상태 패턴 기법과 명령 패턴 기법의 목적 중 하나는 ‘동작의 캡슐화’이다. 두 기법의 캡슐화의 차이점은 상태 패턴 기법의 경우 상태간의 참조가 있는 느슨한 캡슐화이고, 명령 패턴 기법의 경우 완전한 캡슐화이다.

상태 패턴 기법은 연계 동작 제한 기술을 제공하기 때문에 RPG, FPS게임과 같은 특정 동작에 연계되는 동작만을 선택하여 수행하는 경우의 구현에 유리하다. 반면, 명령 패턴 기법은 요청의 수행 시점을 선택할 필요가 있는 시간 단위 진행이나, Undo/Redo기능이 필요한 오목, 장기 등의 게임에 적합하다.

또한 캐릭터 모터 기법의 목적은 ‘유동 물체와의 상호작용’이다. 중력 효과 등의 물리를 필요로 하면서 유동 물체와의 충돌 등의 외부의 힘이 가해지는 경우의 구현에 유리하다.

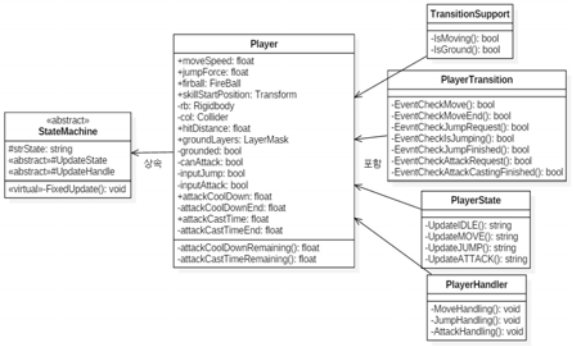
3. 구현 및 성능평가

성능평가를 위하여 Unity3D 엔진 2018.2.5.f1버전에서

C#을 이용하여 대기/걷기/뛰기/공격을 구현하였다. 또한, Unity에서 제공하는 Profiler를 통해 각 기법간 성능을 비교하였다.

가. 상태 패턴 기법 구현

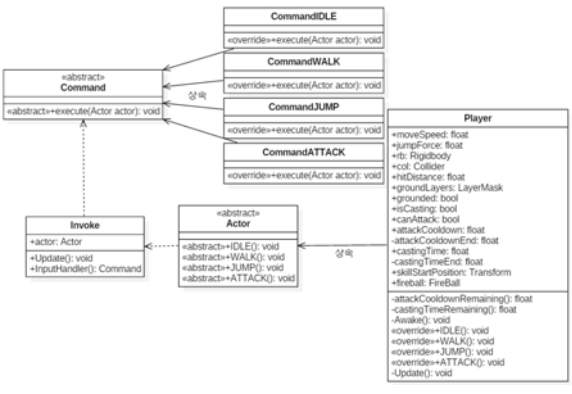
상태 패턴 기법은 '상태'와 '전이'로 구성된다. 캐릭터 동작을 구현하기 위하여 각 '상태'를 함수로 분류하였고, bool 함수를 이용하여 '전이'를 구현하였다. 또한 '입력'과 '상태수행'을 하는 상태 기계를 부모 클래스로 구성하였다. 구현 동작은 Player Handler를 통해 상태 입력을 받고, Player Transition에서 조건에 맞는 '전이'를 확인하다. 그 결과 Player State에서 적합한 상태로 전환된다.



(그림 5) 상태 패턴 기법 구현 UML

나. 명령 패턴 기법 구현

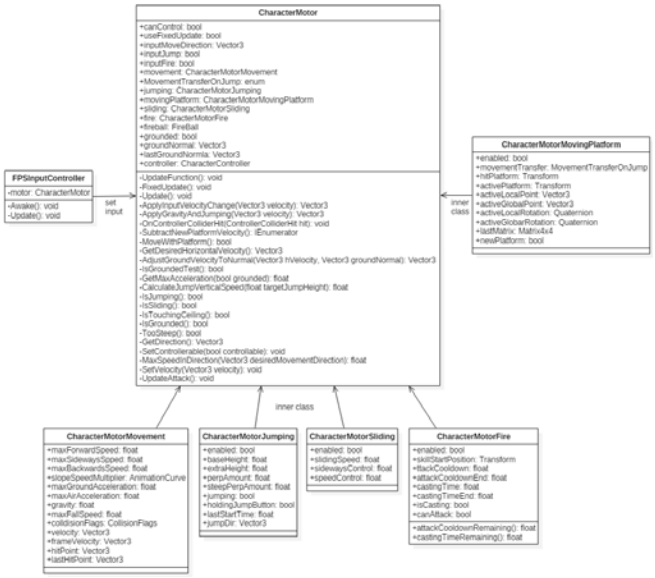
명령 패턴 기법은 'Command', 'Invoker', 'Receiver', 'Client'로 구성된다. 상태 패턴 기법과 동일한 동작을 구현하기 위하여 Command의 수행을 담당하는 Receiver를 Client에 해당하는 Actor클래스의 함수로 구현하였다. 또한, Actor를 Invoker에 캐싱할 수 있게 함으로써, 요청자와 수행자의 결합을 느슨하게 하였다. 구현 동작은 Invoker의 InputHandler함수로 입력을 받아 해당 입력에 맞는 Command를 선택하고, Actor에게 수행을 요청한다. 그 결과 Actor에서 요청받은 Command를 반영한다.



(그림 6) 명령 패턴 기법 구현 UML

다. 캐릭터 모터 기법 구현

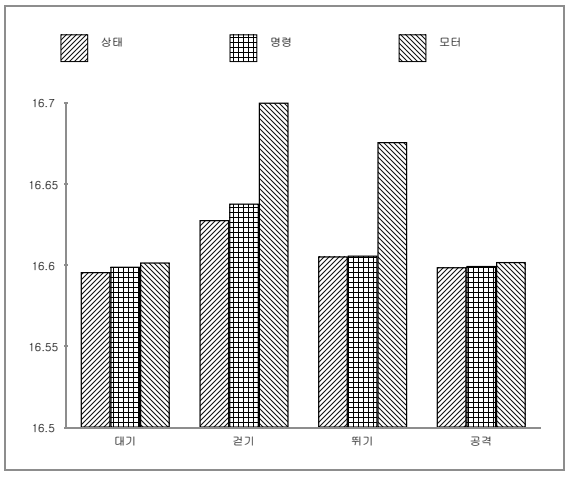
캐릭터 모터 기법은 'Move Platform'과 '모터'로 구성된다. 상태 패턴과 동일한 동작을 구현하기 위하여 Unity Technologies에서 제공하는 캐릭터 모터에 있는 이동 모터, 뛰기 모터, Move Platform에 공격 모터를 추가하였고, 공격 적용 함수를 구현하여 결정 부분에 추가하였다. 구현 동작은 Input Controller로부터 입력을 받고, 그 입력을 기반으로 모터 객체의 데이터를 통해 동작에 필요한 요소값을 계산한다. 그 요소값을 통해 최종 동작을 결정한다.



(그림 7) 캐릭터 모터 기법 구현 UML

라. 성능평가

프레임당 소요시간을 측정하기 위하여 Unity에서 제공하는 Time.time변수를 사용하였고, 각 기법별 동작을 5000프레임씩 실행하였다. 각 기법별 동작의 프레임당 평균 소요시간은 (그림 8)과 같다.

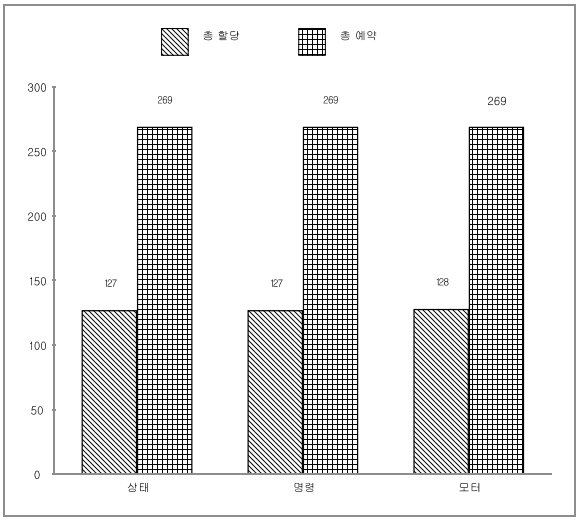


(그림 8) 프레임당 평균 소요시간

(그림 8)에서 상태 패턴 기법이 가장 적은 소요 시간

이 발생하였고, 상태 패턴 기법을 기준으로 명령 패턴 기법은 +0.5~1.5μs, 캐릭터 모터 기법은 +2~150μs의 차이를 보였다.

메모리 사용량에는 Unity에서 제공하는 Profiler를 사용하였고, 각 기법별 동작을 5000프레임씩 실행하였다. 각 기법별 동작의 메모리 평균 사용량은 (그림 9)과 같다.



(그림 9) 메모리 평균 사용량

(그림9)에서 상태 패턴 기법과 명령 패턴 기법은 동일한 메모리 사용량이 측정되었다. 반면, 캐릭터 모터 기법과는 총 할당 메모리에서 +1MB의 차이가 발생하였다.

두 실험 결과 상태 패턴 기법이 프레임당 소요 시간에서 가장 적은 시간이 발생했고, 메모리 사용량은 상태 패턴 기법과 명령 패턴 기법은 동일하였고, 캐릭터 모터 기법보다 우수하였다.

4. 결론

그래픽, 물리엔진 등의 다양한 라이브러리들의 발전으로 게임 개발의 접근성이 용이해졌다. 때문에 다양한 게임들이 등장하고 있으며, 게임의 품질도 높아졌다.

하지만 높은 품질을 추구하는 게임 개발은 과도한 결합도, 불완전성, 기능의 중복성, 클래스 정체성 상실 등의 다양한 문제를 발생시킨다. 이러한 다양한 문제들을 해결하기 위해 본 연구에서는 게임 설계 시 다양한 요소들 중 하나인 캐릭터 동작 설계에서 디자인 패턴 중 상태 패턴 기법과 명령 패턴 기법, 그리고 Unity Technologies의 캐릭터 모터 기법을 이용하여 걷기/대기/뛰기/공격 동작을 설계하고 구현하였다. 또한, 각각의 기법들로 구현한 동작을 5000프레임씩 실행한 결과, 프레임당 소요시간은 상태 패턴이 가장 적은 소요 시간이 발생하였고, 상태 패턴 기법을 기준으로 명령 패턴 기법이 +0.5~1.5μs, 캐릭터 모터 기법이 +2~150μs의 차이를 보였다. 또한, 메모리 평균 사용량은 상태 패턴 기법과 명령 패턴 기법은 동일한 사용량이 측정되었고, 캐릭터 모터 기법은 총 할당 메모리에서 +1MB의 차이가 발생하였다. 비교 실험을 통한 결과 상태 패턴 기법이 캐릭터 동작 구현에 가장 적합하다고 판단된다.

향후 연구로, 캐릭터 동시 동작 구현에 대하여 가장 적합한 디자인 패턴 기법을 탐구하고자 한다.

참고문헌

[1] Alexander, Christopher, et al. "A pattern language: Towns, buildings, construction (center for environmental structure)." (1977).

[2] Beck, Kent, and Ward Cunningham. "Using pattern languages for object-oriented programs." (1987).

[3] Vlissides, John, et al. "Design patterns: Elements of reusable object-oriented software." Reading: Addison Wesley 49.120 (1995): 11.

[4] Nystrom, Robert. Game programming patterns. Genever Benning, 2014.

[5] Unity Documentation, “Rigidbody“, https://docs.unity3d.com/kr/2018.2/Manual/class-Rigidbody.html