객체지향기법을 이용한 게임 그래픽엔진 설계

조승욱, 하수철
대전대학교 컴퓨터공학과

Design of A Graphic Game Engine Using Object Oriented Technology

Seung-Uk Jo, Soo-Cheol Ha
Dept. of Computer Engineering, Daejeon University

요 약

게임이 대중화되고 발전함에 따라 게임을 개발하는 방법 또한 발전하고 있다. 특히 게임엔진은 이제 게임을 개발할 때 필수적인 요소로 등장한다. 게임 엔진이란, 게임을 개발할 때 게임의 하부에서 처리할 수 있는 모든 일을 처리하는 API의 모임이라고 말할 수 있다.

본 논문에서는 구조의 파악이 쉬운 잘 설계된 엔진을 제작하기 위한 방법으로서 클래스 기반의 객체 지향 기법을 이용해 설계하는 방법을 제안한다. 게임엔진에 포함되는 가장 기본적인 그래픽 엔진을 소프트웨어 설계 있어서 현재 가장 강력하고 가장 많이 사용되는 모델링 도구인 UML을 이용해서 학습이 쉽고 구조파악이 쉬운 엔진을 설계한다.

1. 서론

게임이 발전함에 따라 게임을 개발하는 기술 또한 발전하고 있다. 그래픽 처리기법이나 클라이언트 서버 기술, 인공지능의 성능향상, 알고리즘의 새로운 이론 등 새로운 기술이 점점 쏟아져 나오고 있을 뿐 아니라 이제 게임도 한 가지 학문으로서 자리 잡아 가고 있는 실정이다.

오늘날 게임을 만들기 전에 우선적으로 게임엔진을 만드는 것은 너무나도 당연스럽게 받아 들여 지고 있다. 게임 개발자에게 있어 게임 엔진을 만드는 일은 매우 고달프고 힘겨운 작업이면서 게임 개발사의 입장에서 보면 많은 비용과 시간을 감수해야하는 작업이다. 근래에는 전문적으로 게임엔진을 개발하는 개발사의 제품을 구입하여 게임을 만드는 일이 보편화 되어가고 있다. 국내의 유명한 게임사들도 외국의 유명 엔진을 도입해서 이를 분석하고 게임을 제작했다. 하지만 게임 개발사가 원하는 범용성과 확장성, 그리고 빠른 랜더링 속도, 손쉽게 학습할 수 있는 엔진구조, 쉽고 명료한 매뉴얼을 만족시키는 엔진은 거의 없다고 해도 과언이 아니다[1]. 유명 엔진을 도입해서 게임을 개발하는 경우도 그 엔진의 분석에 소요되는 시간이 게임개발일정에 상당부분을 차지한다. 이러한 문제점은 습득하기 쉽고 알아보기 쉽게 잘 설계된 엔진 구조로 문제를 해결할 수 있다. 단순히 엔진 구조만 가지고 모든 문제를 해결할 수는 없지만 명확한 엔진 구조를 갖도록 설계하면 속도도 빠를 뿐 아니라, 안정적이고, 무엇보다도 사용자가 쉽게 접근 할 수 있다는데 그 장점이 있다[1].

현재 시스템 설계 기법은 객체 지향적인 디자인을 지향하고 있다. 객체 지향적 설계는 실세계를 모델로 하기 때문에 그 구조가 이해하기 쉬우며, 그로 인해 각광받는 시스템 설계기법이다[2]. 이러한 객체 지향 시스템 설계를 하는데 있어 여러 가지 도구가 사용되고 있지만 그중 UML이라는 것은 아주 강력한 능력을 가지고 있다. UML은 그래픽적으로 시스템의 설계 내용을 보여주기 때문에 시스템에 대한 이해가 쉽고, 분석이 용이하다. 본 연구에서는 이 UML도구를 사용해서 게임 그래픽 엔진을 설계해 봄으로써 학습이 쉬운 시스템을 설계하는 연구를 한다.

2. 객체지향과 UML

객체지향이란, 문제의 모든 초점을 실세계(Real

world)에 존재하는 실객체를 중심으로 생각하는 것을 뜻한다. 즉 객체지향 설계(Object Oriented design)란 문제 해결을 위한 목적 시스템를 설계하는 데 있어, 실세계의 실객체들이 설계의 기초가 설계 기법을 말하며, 객체지향 언어(Object Oriented Language)란 프로그램을 작성하는 데 있어, 실객체를 소프트웨어 객체로 쉽게 표현할 수 있도록, 이와 관련된 많은 기능을 가지고 있는 언어를 지칭한다. 그리고 객체지향 프로그래밍이란 실세계를 프로그램 세계로 표현하는 데 있어, 프로그램의 구성이 실객체를 표현한 소프트웨어 객체들로 이루어지고, 프로그램의 수행은 이러한 소프트웨어 객체들이 상호 동작하여 전체 프로그램이 진행되는 프로그래밍 기법이다.

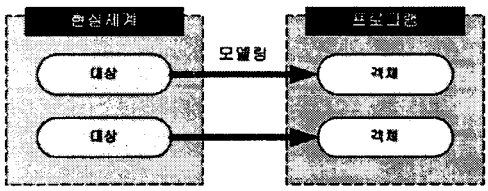

그림 1. 객체 지향 프로그램의 현실세계 모델링

객체 지향 개념은 재사용성이나 고속의 개발 시간 등 많은 이점을 제공한다. 이러한 이유로 요즘 개발되는 애플리케이션들은 객체 지향 기법 사용을 통해 설계되고 만들어지고 있다.[3-4]

UML은 Unified Modeling Language의 약자로 현재 객체지향 시스템 개발 분야에서 가장 각광받는 설계 도구 중 하나이다. UML은 객체지향 설계를 위한 모델링 언어로서 개발되었고, 소프트웨어 개발자 뿐 아니라 비즈니스 영역을 포함한 여러 영역에까지 모델링 할 수 있게 많은 도구를 제공한다. UML은 시스템을 모델링하는데 적합하며, 기업 정보시스템으로부터 분산처리의 웹 기반 애플리케이션까지 그리고 견고한 실시간 내장 시스템까지도 범위에 포함한다. 소프트웨어 중심 시스템의 산출물을 가시화하고, 명세화하고, 구축하고, 문서화한다. UML의 목적은 약속된 표기법으로 그림을 그려가면서 모델링을 구체화하고, 작성된 명확한 문서를 통해 여러 사람이 이해를 목표하는 시스템을 빨리 구현하는 것이다. UML의 표현력은 매우 풍부하고 시스템들을 개발하고 배치하는데 필요한 모든 관점을 다룬다. UML은 시스템의 개발자가 자기의 비젼(vision)을 구축하고 반영하는데 있어 표준적이고 이해하기 쉬운 방법으로 설계할 수 있도록 도와주며, 이러한 결과물을 다른 사람과 효과적으로 공유할 수 있는 매커니즘을 제공한다. UML은 디자인표기법이며, UML의 여러 가지 그래픽 요소는 하나의 다이어그램을 그리는데 사용된다.[5-6]

3. 게임 그래픽 엔진 설계

본 논문의 시스템은 게임을 만들기 위해 그래픽을 출력하는 역할을 담당하는 그래픽 엔진을 객체지향기법에 따라 설계하고자 하는 목적으로 디자인되었다. 시스템은 그래픽 엔진이 가져야하는 가장 기본적인 기능으로만 이루어져 있으며, 설계를 통해 향후 분석과 고급 기능들을 확장이 용이하도록 한다.

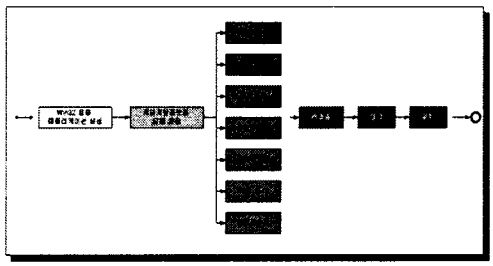

그림 2. 개발자 입장의 인터페이스 흐름

본 연구의 시스템은 비디오메모리에 주표면과 보조 후면버퍼를 지원하는 그래픽 시스템으로 모든 그래픽 요소들은 이 비디오메모리의 버퍼를 통해서 렌더링 된다. 기본적인 그리기요소를 포함하고 있으며, DirectDraw를 설정하고 그래픽을 빠르게 생성할 수 있는 기능을 제공한다.

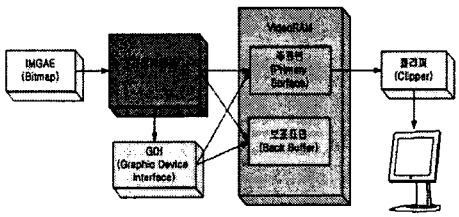

그림 3. 그래픽엔진 개념도

객체지향설계 기법을 이용해 설계를 위한 객체의 도출을 위해 그래픽 엔진이 가지는 기능과 속성을 분석해서 각 기능별로 객체를 도출하고 각 객체를 클래스로 정의하였다.

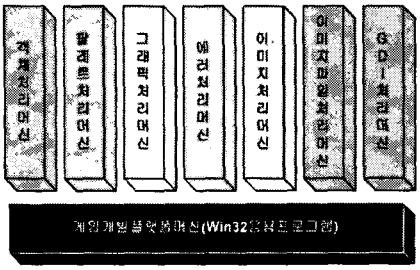

그림 4. 그래픽엔진 시스템 구성도

그림 4는 도출해낸 객체들로 구성된 엔진의 구성도이며, 각 기능을 담당하는 처리머신들은 설계 시 클래스로 대응된다. 시스템을 사용하는 사용자의 시점에서 시스템을 모델링하기 위해 유스케이스를 이용한다. 개발자는 프로젝트를 생성하고 개임개발에 필요한 플랫폼을 생성한다. 엔진이 제공하는 객체처리, 에러처리, 이미지처리, 그래픽처리, 이미지파일처리, 팔레트처리, GDI처리 기능 중 작업하고자 하는 기능을 선별해서 호출한다. 게임개발 플랫폼 내로 호출하면 다음으로 호출된 처리머신의 행동양식을 결정한다. 플랫폼 내에서는 윈도우 출력하는 모듈을 포함하고 있으며, 모든 작업 후 화면출력이 가능하다. 사용자는 출력물을 보고 원하는 사항을 입력하면 게임개발 플랫폼의 메시지처리기로 전달되어 프로세스한다. 시스템의 작업흐름을 통해 그래픽엔진을 유스케이스 다이어그램으로 모델링 한 것은 그림 5와 같다.

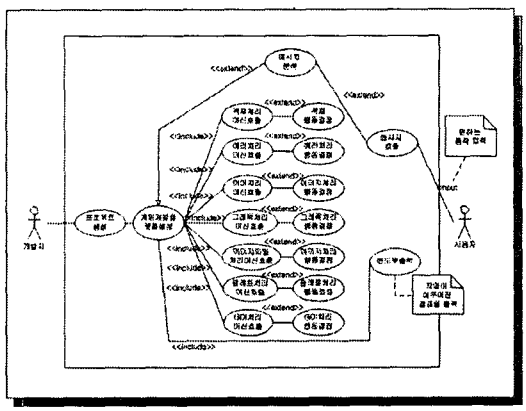

그림 5. 시스템의 유스케이스 다이어그램

각 객체간의 교류는 어떠한 시퀀스 내에서 발생하며, 시퀀스가 시작해서 끝나기 위해서는 시간이 소요된다. 시간의 흐름에 의해 시스템을 표현한 시퀀스다이어그램은 그림 6과 같다.

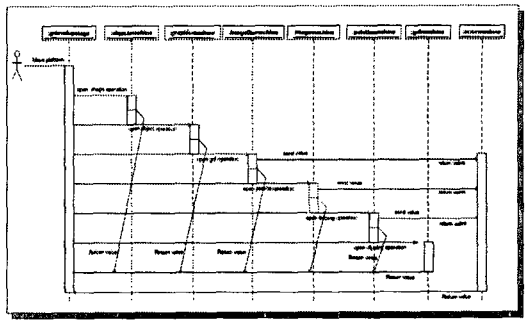

그림 6. 시스템의 시퀀스 다이어그램

개발 플랫폼 객체가 생성되면 각 객체들은 개별적으로 개발이 가능하다. 각 객체를 생성하고 오퍼레이션을 작성하는 것은 특별한 순서가 없으며 그림 6의 시퀀스 다이어그램은 애플리케이션 개발에 있어 진행될 수 있는 여러 가지 경우 중 한가지의 시간적 흐름을 보여준다.

그래픽 엔진의 동작 과정을 UML의 다이어그램 표기법 중에서 액티비티 다이어그램을 통해 그림 7과 같이 모델링 하였다.

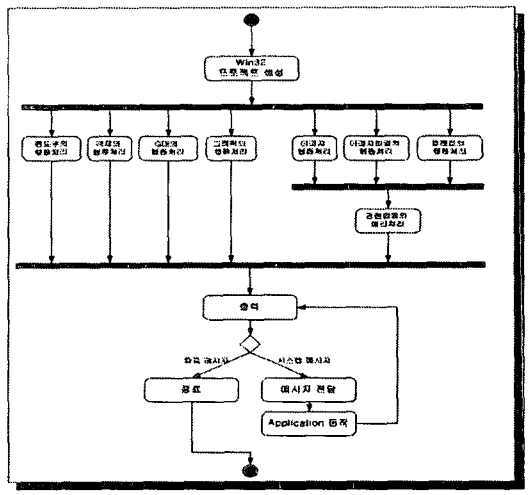

그림 7. 시스템의 액티비티 다이어그램

클래스의 설계는 그래픽을 출력하기 위한 엔진의 가장 기본적인 기능에 의해 모델링 하였으며, 추가적인 기능은 속성을 분류하여 현재의 클래스에서 하위 클래스를 생성하여 확장이 용이하도록 모델링 되었다. 클래스다이어그램의 멤버데이터와 멤버함수, 각 클래스간의 연관관계를 표현한 것으로 표현된 다중성 및 연관관계를 그대로 계승하여 다이어그램을 완성했다.

그림 8의 클래스 다이어그램은 설계된 시스템의 클래스명과 각 클래스에 포함된 함수를 보여주고 있다.

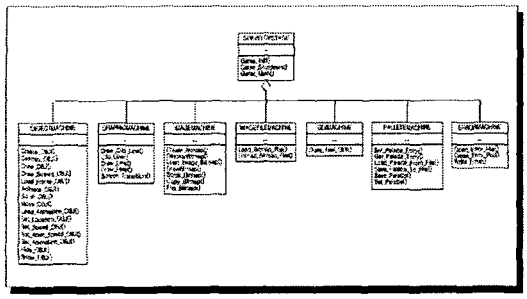

그림 8. 시스템의 클래스 다이어그램

본 시스템은 각 기능을 클래스 기반으로 행위를 하는 대상에 그 초점을 맞추어 설계하고 객체지향 설계 도구인 UML을 이용하여 설계를 디자인적으로 표현하였다.

4. 결론

본 논문에서는 그래픽엔진의 개발에 있어 객체 지향 기법을 이용해 클래스 기반으로 엔진을 모델링하고 설계하였다.

게임에 있어 그래픽을 담당하는 기본적인 기능을 제공하는 엔진의 모듈을 기능별로 분류하고 추상화된 클래스를 통해서 함수를 배치함으로써 각 클래스가 담당하는 역할을 구체화하고 정형화하였다.

본 논문에서 설계된 객체 지향 기법을 이용한 엔진은 UML을 이용하여 디자인적으로 표현해 그 구조가 파악하기 쉬우며, 차후 추가적인 기능을 가진 모듈을 삽입할 때 적용이 쉽도록 만든다.

이러한 객체 지향적인 게임엔진 설계는 엔진 개발이나 이렇게 설계된 엔진을 도입 시 분석이 용이하고 게임 개발자 입장에서 많은 비용과 개발 시간을 절감할 수 있는 효과를 가져 올 수 있을 것이라고 기대되며, 사용자의 입장에서 쉽게 접근할 수 있어 엔진 설계에 있어 모든 측면은 아니지만 많은 문제들을 해결할 수 있을 것이라고 생각된다.

엔진 설계부분에서 좀 더 발전적인 객체 지향 적인 개발 기법의 적용과 엔진 설계에 대한 관련 연구가 지속적으로 이루어진다면 좀더 분석이 쉽고, 관리가 쉬우며, 유지보수와 범용성, 확장성을 가질 수 있는 엔진을 개발할 수 있을 것으로 전망된다.

[참고문헌]

[1] 김상규, "UML and 3D Game Engine Designed", Game Developer, 2002.5

[2] Grady booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide", Addison Wesley

[3] Joseph Schmuller, "UML 객체지향설계 Second Edition", SAMS

[4] 전병선, "객체지향이야기", compeople

[5] Andrew Wolfe, Derek B.Noonburg, "A Superscalar 3D Graphics Engine", http://citeseer.nj.nec.com/281318.html

[6] Lars Bishop, Dave Eberly, and Turner, Mark Finch, Michael Shantz, "Designed a PC Game Engine", IEEE Computer Graphics and Applications p46 - p53, January/February 1998