# 실제적인 차량 시뮬레이션을 위한 효율적인 물리 컴포넌트 설계

이병윤, 최종화, 신동규, 신동일
세종대학교 컴퓨터 공학과
e-mail : {over2015, com97}@gce.sejong.ac.kr, {shindk, dshin}@sejong.ac.kr

# Efficient Physics Components Architecture for Realistic Vehicle Simulation

Byungyoon Lee, Jonghwa Choi, Dongkyoo Shin, Dongil Shin
Dept. of Computer Engineering, Sejong University

요 약

본 논문에서는 현재 게임 및 각종 동역학 관련 시뮬레이션에서 요구되는 물리 효과를 효과적으로 이용하기 위한 컴포넌트와 그 구조를 제시한다. 특히 가장 다양한 분야에 적용되는 차량 시뮬레이션에 특화된 구조를 제시하고 있으며 사용되는 각종 컴포넌트의 구조를 세부적으로 설명하였다. 또, 위의 컴포넌트를 기반으로 구현한 제작 툴을 이용하여 효과적인 개발에 필요한 여러 가지 요소를 만족시키고 있다.

## 1. 서 론

하드웨어 기술의 발전에 힘입어 불과 몇 해 전까지만 해도 상상하지 못했던 실제적인 영화, 게임 등 IT 전반에 걸쳐 믿을 수 없는 진보가 이루어 졌다. 이러한 발전의 추세는 계속해서 가속화 되고 있으며, 이는 '무어의 법칙' 혹은 최근 등장한 '황의 법칙'과 맞물려 현실로 다가왔다. 최근의 추세를 보면, 현실적인 게임이나 시뮬레이션, 혹은 영화를 보여주기 위해서 실제적인 '모습'에 치중하는 것만이 아니라 실제적인 '동작'에 초점을 맞추고 있다는 것이다. 이는 물리 엔진의 등장 및 물리 연산 칩(PPU)의 개발 등으로 미루어 짐작 할 수 있다.[1]

이러한 진보를 통하여 현재 실시간으로 처리 가능한 물리 연산의 대표적인 분야는 차량 역학이라고 할 수 있다. 각종 조인트(Joint)와 충돌 검출(Collision Detection)및 충돌 반응(Collision Responsibility)을 다루며, 현재 다방면의 시뮬레이션에서 사용되고 있다.[2, 3]

또한, 실제 모델링 작업에 적합하도록 차량 자체의 객체화가 쉽고, 동역학의 대부분의 공식을 적용하기에 쉬운 것이 바로 차량 시뮬레이션이다. 물리 엔진의 필요성이 대두되는 추세에 맞추어 본 논문에서는 물리 컴포넌트를 설계하여 개발자의 필요에 따라 재사용과 수정이 용이하도록 설계하고, 툴을 구현하여 구조의 당위성을 제시한다.[4]

## 2. 기초 컴포넌트

일반적으로 객체들은 특정한 상호 작용에 의해 완성된다. 즉, 차량이라고 하는 객체는 바퀴와 몸체로 이루어지며, 각각의 바퀴와 몸체는 하나의 관계에 의해서 상호 작용하게 된다. 바퀴의 개수는 차량에 따라 다르며 각각의 관계 역시 표현하려는 차량의 종류에 따라 다르지만, 기본적으로 바퀴와 몸체라는 객체와 그것들 간의 관계에 의해 하나의 차량이 정의 될 수 있다.

또, 차량과 외부 세계간의 상호 작용이 발생한다. 크게는 세계와 차량이라는 객체와 그것들 간의 충돌에 의한 상호 작용이 그것이다.


그림 1. 외부 세계와 차량, 충돌 관련 객체의 관계

그림 1. 은 위에서 설명한 객체와 그 관계에 대한 자료이다. 다시 말하자면, 외부 세계라는 객체와 차량이라는 객체는 충돌 검출과 반응을 담당하는 일종의 충돌 관리자로 인하여 상호 관계가 성립된다는 것이다.

외부 세계는 그 내부에 있는 객체에 직접 힘을 가할 수도 있으며, 차량이 이동하여 충돌이 발생할 수도 있다. 결과적으로 외부 세계는 개발자가 구현하는 가상환경을 의미하며, 실제 사용자가 능동적으로 조정 할 수 있는 부분이 아니다. 그러나 차량 객체의 경우는 그 반대로 최초의 설정만을 개발자가 담당할 뿐, 그 이후의 작동은 전적으로 사용자가 책임지는 객체이다. 위의 두 가지 상황에서 보면, 전자의 경우, 바다 위에 부는 바람의 힘을 이용하여 앞으로 나아가거나 방향을 바꾸는 행위와, 후자의 경우, 장애물들과의 충돌을 피하는 행위와 같이 능동적인 객체이다.

충돌 관리자는 바로 이러한 사용자의 능동적 행위가 가상환경 속에서 어떻게 적용되는지를 처리 하는 객체로 외부 세계와 차량 객체의 행동에 대한 반응을 처리한다.

3. 컴포넌트 전체 구조

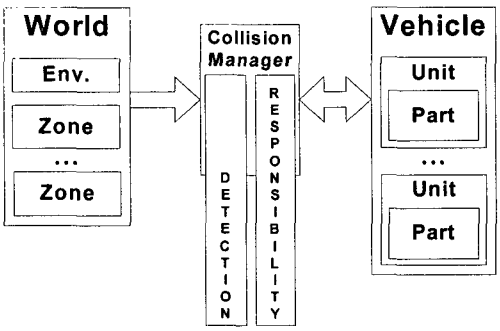


그림 2. 간략하게 표현한 전체 구조도

그림 2는 그림 1을 실제 적용되는 객체들을 추가하여 세분화한 구조도 이다. 위에서 언급한 외부 세계를 World 객체로 구현하면, 그 내부는 물리적 환경 요소를 지니는 Zone 객체들로 구성된다. 각각의 Zone 객체는 World를 대표하는 Environment 객체의 영향을 받아 특정한 환경의 모습을 표현하게 된다.

반대편의 Vehicle 객체는 최소 단위의 Part 객체와 두 개 이상의 Part가 모여 구성되는 하나의 Unit 객체들의 집합으로 표현 할 수 있다. 예를 들어 바퀴는 하나의 Part이며 바퀴와 차량의 몸체가 연결되면 그 부분은 두 개의 Part를 연결한 Unit 객체로써 사용된다. 이러한 Unit들의 집합이 하나의 Vehicle 객체로 처리된다.

이렇게 구성된 World와 Vehicle은 Collision Manager 객체를 통하여 객체간의 충돌 검출 및 반응에 대한 계산이 이루어지고, 결과가 다시 Vehicle에 적용되어 화면에 출력되는 것이다. 단, 이 후 환경 요소들의 파괴가 가능한 수준의 구현이 된다면 충돌 반응은 World 객체에도 영향을 주게 될 것이고 위의 구조는 약간 변형이 가해져야 한다.

## 4. 각 컴포넌트

### 4.1. World 객체와 그 하부 구조

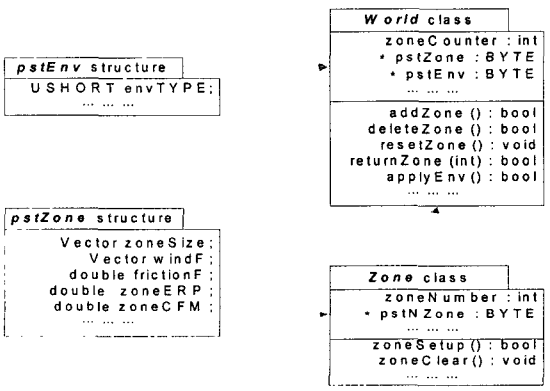


그림 3. World 객체와 그 파생 객체들

World 객체를 살펴보면, Zone 객체를 리스트로 저장하기 위한 주소 값이나 현재 등록된 Zone의 개수 및 Zone의 추가/삭제를 담당하는 일종의 매니저 클래스의 역학을 하고 있음을 알 수 있다. 또, 그러한 Zone 객체는 여러 물리 환경 변수를 갖고 있는 구조체를 취하며, 해당 구조체를 셋업 할 수 있는 함수를 갖고 있다. Zone은 일정한 크기의 육면체로써 각각의 Zone이 타일 형태로 집합하여 하나의 World를 구성하는 구조이다. 또, World 객체에는 하나의 Environment 구조가 추가 될 수 있는데, 이 구조체는 현재의 World의 템플릿을 지정하는 역할을 하며, 만일 아무것도 지정되지 않는 다면 일반적인 지형을 의미하게 된다.

### 4.2. Vehicle 객체와 그 하부 구조

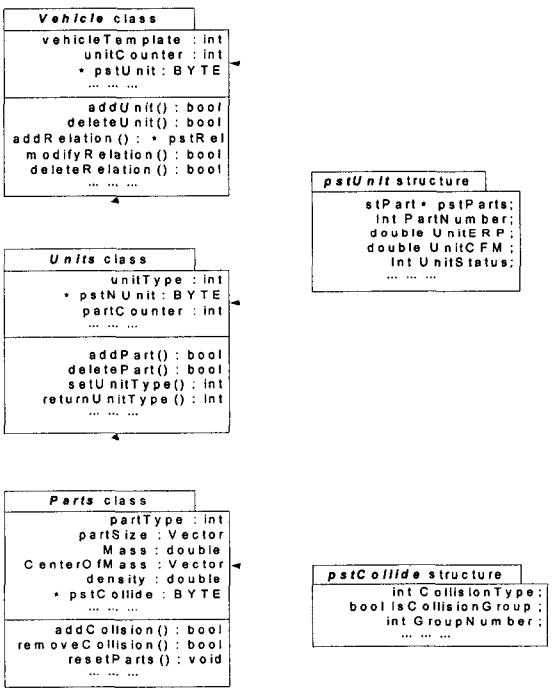


그림 4. Vehicle 객체와 그 파생 객체들

Vehicle 객체는 Unit 객체들의 집합으로 구성되며, World 객체와 마찬가지로 일종의 매니저 클래스의 형태를 취하고 있다. 단 Unit 간의 관계를 설정 할 수 있도록 pstUnit 구조체를 통하여 Unit의 속성 값들을 처리 하는 차이점이 있다.

하부로 내려가면 Unit 객체도 Part 객체를 관리하기 위한 매니저의 성향을 띠게 된다. 단, 기존의 객체들과의 차이점은 Part 객체와 Collision Manager 의 관계가 형성된다는 점으로 Part 객체는 Vehicle 의 최소 단위이자 충돌 처리의 최소 단위이기도 하다는 것이다.

그림 3과 4에서 볼 수 있듯이 두 객체는 각 객체로써는 완전한 반면 상호 간의 작용을 할 수 있는 공통부분은 전혀

없다. 2 장에서 언급한 상호 작용을 양 객체에서 처리 하고자 각각의 객체 내부에 이러한 처리 모듈을 삽입하게 되면, 결국 컴포넌트 구조의 불균형을 초래 하며, 컴포넌트의 수정 및 관리에 불필요한 작업량을 증가시키는 시간 낭비가 발생하게 되기 때문이며, 이를 효율적으로 해결하고자 다음 절에서 언급 할 충돌 매니저 객체를 생성하게 된 것이다.

### 4.3. Collision Manager 객체

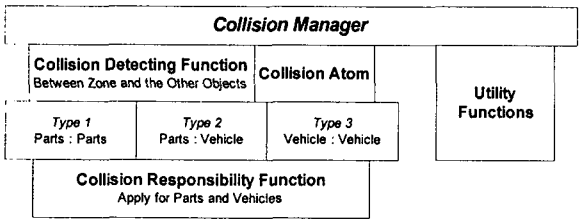


그림 5. Collision Manager 객체

충돌 처리 및 반응을 담당하는 Collision Manager 객체는 Zone과 Part간의 충돌, Part 간의 충돌, Part와 Vehicle 간의 충돌, Vehicle 간의 충돌 등을 검출한다. 충돌에 관한 검출 방식은 AABB 또는 OOB 같은 일반적인 방법들을 사용하며, 충돌 범위를 수정하기 위한 각종 유틸리티 함수를 제공한다. 일단 충돌 검출이 이루어지면 Zone에서 제공하는 환경 변수와 Vehicle 및 충돌에 참여한 객체들이 제공하는 변수를 조합한 하나의 계산식을 통하여 충돌 반응에 필요한 결과 값을 제공한다. 이를 Part 및 Vehicle 객체에 적용시켜 충돌 반응을 구현하여 화면에 출력하게 된다.[5]

## 5. Implementation

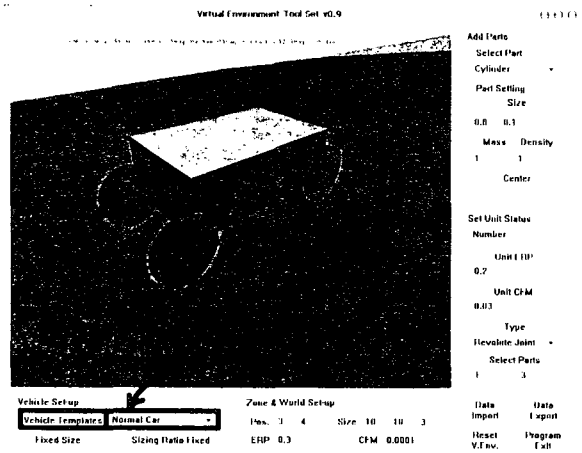


그림 6. Virtual Environment Tool Set 화면

간단히 위의 차량을 만드는데 필요한 객체는 바퀴 4개와 차량 몸체 1개, 바닥 면이 있으며, 각 객체마다의 데이터 설정과 각 객체간의 상호 작용을 정의하기 위한 데이터의 입력과 같은 수많은 작업이 필요하며, 이는 매우 비생산적인 일 일수 밖에 없다.

실제 차량 시뮬레이션에 사용하는 셋 업 툴을 보면, 굵은 줄로 표시된 곳에 차량 템플릿이 존재한다. 즉, 일정한 구조의 차량은 자동으로 생성한다는 것인데, 템플릿 형태로 제공된다. 또한 Zone, Unit, Part 등의 데이터를 설정하는 창이 존재하며, 데이터는 일정한 구조체에 담겨 저장 가능하다. 실제 어플리케이션에서는 이러한 저장된 데이터를 프로그램 시작 시 호출하여 사용하면 시뮬레이션이 간단하게 제작된다. 물론 차량을 움직이는 등의 추가적인 구현은 개발자의 몫이지만, 이러한 개발 툴의 구현만으로도 개발자의 부담은 매우 줄어들게 되는 것을 알 수 있다.

## 6. 결론 및 향후 연구

해외의 상용 및 공개 물리 엔진들의 구조는 어느 것 하나에 특화되어 있다기보다 굉장히 많은 양의 API와 객체들을 제시해주고 개발자가 자신의 목적에 맞게 이것을 수정하여 사용하도록 되어 있다. 그런 이유로 본래 제공되는 엔진의 성능에 비하여 오히려 성능이 하락하는 등의 문제점이 발생하는데 반하여, 본 논문에서 제시한 이러한 구조의 접근 방법을 이용하면, 차량 시뮬레이션을 개발하기 위한 툴이나 실제 어플리케이션 제작이 간결하게 됨을 알 수 있었다.

물론 현재의 컴포넌트의 구조는 차량 시뮬레이션이라는 제한사항이 있으나, 동역학을 골자로 하는 대부분의 물리 시뮬레이션으로의 적용과 이를 기반으로 제작 할 통합 툴의 구현을 목표로 하고 있다.[6, 7]

## 7. References

[1] AGEIA physX Processor, http://www.ageia.com/technology.html
[2] Jimenez P.; Thomas F., "Torras C, 3D collision detection: a survey", Computers and Graphics, Volume 25, Number 2, pp. 269-285(17), April 2001
[3] Graham Morgan, Kier Storey. "Scalable Collision Detection for Massively Multiplayer Online Games," aina, vol. 01, no. 1, pp. 873-878, 19th 2005.
[4] R Serban, EJ Haug, "Globally Independent Coordinates for Real-Time Vehicle Simulation" ASME J. Mech. Des, 2000
[5] James T. Klosowski, Martin Held, Joseph S.B. Mitchell, Henry Sowizral, Karel Zikan, "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs", IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 4, NO. 1, JANUARY-MARCH 1998
[6] Havok, http://www.havok.com/products/physics.php
[7] O.D.E.(Open Dynamics Engine), http://ode.org