포트폴리오

금오공과대학교

컴퓨터소프트웨어공학전공

응용수학전공

08학번 김우현

CONTENTS

1. HISTORY

1-1. 수상 이력

'15.09.09	K-Hackathon 3 최우수상
'14.11.14	kit Engineering Fair 2014 은상
'14.07.11	2014 국가슈퍼컴퓨팅 SummerSchool@UNIST 장려상 및 공헌상
'09.12.01	창의적 종합설계 Festival 우수상
'09.10.16	금오공학제 공학작품 전시회 건축부문 금상

주요 SW 수상 이력

수상	K-Hackathon 3 최우수상 미래창조과학부 장관		
대상	Globell 팀(4명)	주최	미래창조과학부
일자	2015.09.09	주관	(사)앱센터
설명			

전국 대학생 앱 개발 챌린지 K-Hackathon 3에서 "Globell – 무료 인터넷 전화 서비스 플랫폼 개발"로 최우수상(미래창조과학부 장관상)을 수상하였다.

수상	2014 국가슈퍼컴퓨팅 SummerSchool@UNIST 장려상 및 공헌상 한국과학기술정보연구원장		
대상	의리 팀(2명)	주최	한국계산과학공학회
일자	2014.07.11	주관	KISTI 국가슈퍼컴퓨팅연구소
설명			

OpenMP 및 MPI를 통한 수학 계산 최적화 프로젝트로 우수한 성적을 거두어 장려상을 수상하고, 병렬화 프로그래밍 전파에 기여하여 공헌상을 수상하였다.

수상	kit Engineering Fair 2014 은상 금오공과대학교 총장		
대상	수적천석 팀(4명)	주최	LINC 사업단
일자	2014.11.14	주관	금오공과대학교 공학교육혁신센터
설명			

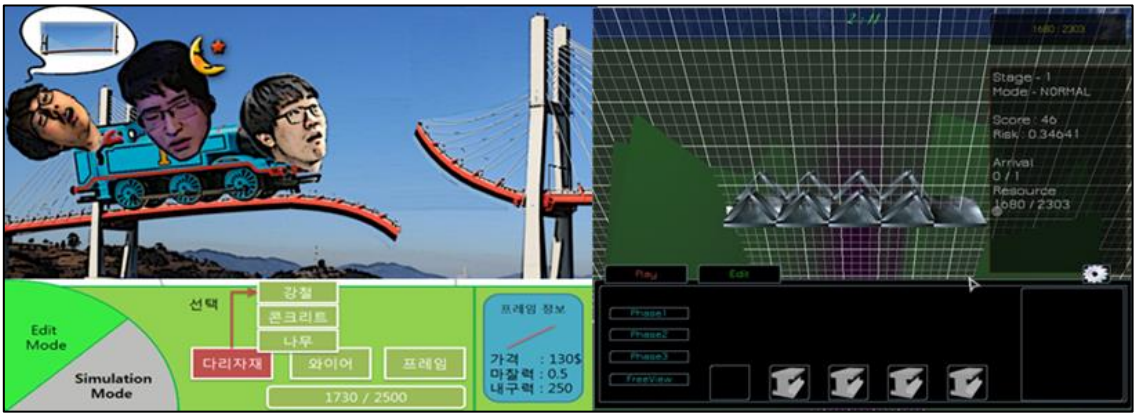
졸업 전시회에서 "FriendlyFarm - 농업 교육용 시뮬레이션 게임" 우수 작품으로 은상을 수상하였다.

1-2. 연도 별 개인 주요 약력

일자	내용
'15.09	☞ 금오공과대학교 8월 24일 학사 졸업
	☞ K-Hackathon 공모전, 'Globell' 무료 인터넷 전화 서비스 개발('15.07~), 최우수상
'15.07	☞ 창조벤처특화 인력양성과정 이수('15.05~, 140시간)
	☞ 삼성소프트웨어멤버십 정회원 수료('14.01~)
	☞ 삼성 DMC연구소 Graphics Lab, 3D Garment Modeling Tool 개발 B2B 진행('15.04~)
'15.03	☞ 'SmartVoice', 안드로이드 음성 프레임워크 개발('14.11~)
'15.01	☞ POSTECH, UNIST 'Irreducible Matrix의 고유치 문제를 통한 줄임 가능성과 스핀격자 모델 연구'('14.11~)
'14.11	☞ 교내 공모전 kit Engineering Fair 2014, 'Friendly Farm' 출품, 은상
'14.09	☞ 'Cross the bridge!', 3D 다리 건설 시뮬레이션 게임 개발('14.05~)
'14.07	☞ UNIST 여름방학캠프 OpenMP, MPI 병렬화 슈퍼컴퓨팅 교육 이수 및 수학/물리 계산 프로젝트(1주), 공헌상, 장려상
	☞ 'Friendly Farm', 교육용 3D 농업 시뮬레이션 게임 개발, 졸업 프로젝트('14.03~)
'14.06	**(교내)** 웹프로그래밍, 데이터베이스, 마이크로프로세서인터페이싱 등 수강
'14.05	☞ Tizen 웹 앱 개발 과정 이수(16시간)
'14.01	☞ 삼성소프트웨어멤버십 정회원 합격
'13.12	☞ 2013 슈퍼앱코리아 공모전 참여, '가족이야기' 안드로이드 앱 개발('13.09~)
	(교내) 객체지향소프트웨어공학, 컴퓨터그래픽스, 프로그래밍설계 등 수강
'13.08	☞ 벤처기업 ㈜온새메디, 한의 의료기기 SW R&D, 현장실습 활동('13.06~)
'13.06	**(교내)** 윈도우프로그래밍, Java, 자료구조, 소프트웨어공학, 통계학 등 수강
'12.12	☞ 미분기하 그래프 Tool 첫 개인 작품, 삼성소프트웨어멤버십 정회원 도전
	(복학) C#, C++, 컴퓨터네트워크, 운영체제개념 등 수강
'12.07	☞ UNIST, 여름방학캠프 OMP, MPI 병렬화 슈퍼컴퓨팅 교육 이수(1주)
'12.05	**(군 제대)**

2. MAJOR PROJECTS

2-1. 작품 명	Cross the bridge!	역할 및 팀	물리 엔진- (PL)김우현 게임 프레임워크- 송시윤 맵 에디터- 박재성



본인 개발 파트	3D 물리 엔진 개발, 사운드 프로그래밍	기간	'14.05.01 ~ '14.09.01
개발 환경	Windows 7	개발 도구 및 언어	C++, OpenGL, CEGUI Visual Studio 2010/2013, 3Ds Max,

개요

"Cross the bridge!" 프로젝트는 다리를 건설하고 안정성을 테스트하기 위한 시뮬레이션을 하며 즐기는 게임이다. 이 곳에서 물리 엔진 개발 파트를 진행하였다. 따라서 프로젝트와 함께 충돌 기반의 3D 강체 물리 엔진인 RPEngine에 주안점을 두었다.

내용

1. 개발 동기 및 주요 개발 명세

대학교에서 배웠던 수학, 물리학, 소프트웨어공학의 지식을 토대로 범용 물리 엔진을 개발하고 다리 건설을 위한 시뮬레이션 프로그램을 개발하고 싶었다.

물리 엔진을 개발하고 싶었던 이유는, 현실 세계의 일들을 시뮬레이션 한다면 많은 일들을 해결 할 수 있을 것이라 생각했기 때문이었다. 한 예로는 쿼드콥터가 비행하거나 임무를 수행하는 데에는 물리적 처리를 위한 알고리즘이 필요할 것이다. 혹은 차량, 항공 등의 기계 및 건설 분야에 적용한다면, 시뮬레이션을 통해 사고를 회피하는 기술을 개발하거나 그러한 일들을 예견하여 하드웨어를 설계에 도움을 줄 수 있을 것이다. 이런 점들을 생각해 볼 때, 현재와 미래 사회에 굉장히 필요한 기술이라고 생각하였다. 마지막으로 내 전공과 성격에 절묘하게 맞아서 굉장히 재미있게 개발할 수 있는 분야이기도 하였다.

3D맵 에디터 도구 개발을 원하는 회원과 게임 프레임워크 개발을 하고자 하는 회원으로 구성해 한 팀을 이루었다. 프로젝트를 진행하기 위해 팀원들과 함께 개발을 기획하여 현재의 "Cross the bridge !"라는 다리 건설 시뮬레이션 게임이 되었다. 주요 명세 및 시나리오는 다음과 같았다.

1) 프로그램을 통해 사용자는 다리를 건설하기 위한 지형 환경을 직접 만들거나 선정한다.
2) 해당 지형에서 제한된 자원 내에 다리를 건설한다.
  2.1) 다리 건설을 위한 자재(철, 나무 등)를 선택하고 프레임 길이를 직접 그릴 수 있다.
  2.2) 프레임을 하나씩 그리며 전체적인 다리를 설계한다. 단, 제한된 자원을 넘어설 수 없다.
3) 완성되었으면 시뮬레이션을 가동하여 충분히 튼튼한 다리인지 테스트한다. 만약 테스트가 성공적이라면 게임 보상(자재와 골드)을 받게 된다.

2. 개발 과정

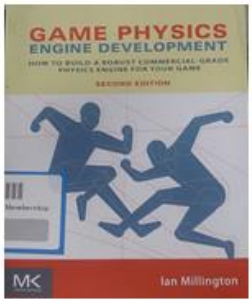
2.1. 물리 엔진 라이브러리 개발

먼저, 물리 엔진 개발을 위해 약 4 주 동안 교내 도서관 책을 찾아보기도 하고 웹 검색을 통해 관련 자료를 찾아 다녔다.

가장 쉽게 접할 수 있는 엔진 개발 관련 도서는 게임 관련 서적에서 물리 엔진 자료를 찾을 수 있다. 국내에 출판되어 있는 대부분의 도서들의 공통점은 이론적/추상적인 기술과 고급 기술에 대한 이야기는 많으나, 이론만을 가지고 실제 개발에 진입하기에는 부족하다고 판단하였다.

그러던 어느 날, 교내도서관의 컴퓨터/공학 코너가 아닌 해외 서적 코너 구석에서 Game Physics Engine Development 라는 최근 자료의 해외 서적을 발견했고, 이 책의 목차와 Introduction 을 보면서 실마리를 찾을 수 있었다.

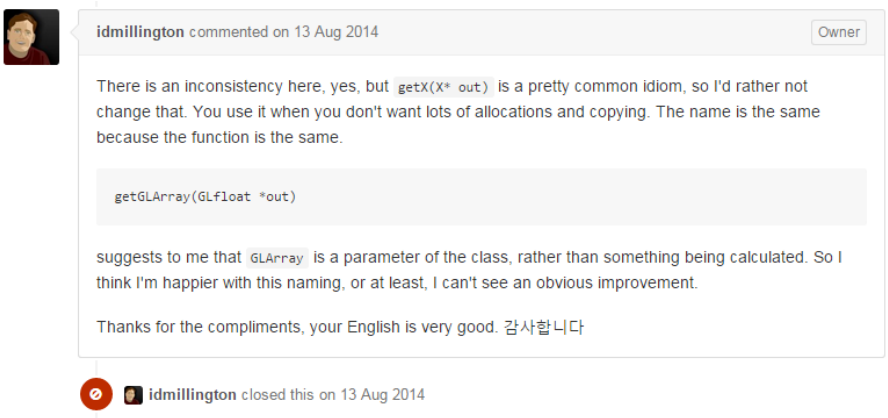
"...This book aims to step you through
the creation of a physics engine...
Use it as a base for further experimentation,
or make various design decisions and create your
Own system under the guidance that
this book provides... - 1.Introduction -"



기초적인 부분부터 시작하여 자신이 추천하는 설계 철학과 엔진 개발의 전체적인 과정을 세세히 담은 책이었다.

원서의 깊은 이해를 위해서는 내용을 반복적으로 봐야 하는 부분들이 많았다. 이 책에서 중요한 부분들을 이해한 후 직접 한글로 번역하면서 개발하기로 결심하였다. 개발을 진행한 기간 동안 약 100페이지 분량의 한글 파일을 작성하였다.

책의 내용과 소스코드 하나하나 따져보며 필요한 새로운 부분을 추가하거나 수정하면서 나만의 RPEngine 라이브러리를 개발하였다. 아주 사소한 것이라도 궁금한 사항은 Ian Millington에게 직접 물어보기도 하였다. 현재 github에서 RPEngine프로젝트를 찾을 수 있다.(하단 참조)



2.2 엔진 상위 API

이렇게 하여 약 4개월 동안 1.00 버전의 엔진인 "RPEngine"을 개발하였다. 그러나 중요한 것은, 게임 로직을 개발하는 단계에서 내가 개발한 물리 엔진의 사용법을 팀원이 익혀야 할 것이라는 문제를 예상하였다. 예를 들어, 다리 건설을 위한 '건설 자재'라는 객체는 어떠한 물리적인 속성을 지녔는지에 대한 정보 담기 위해서 물리 엔진의 API를 통해 작성해야 하였다. 따라서 팀원의 편의성을 위해 사용하기 쉽고 간단한 엔진 상위 프레임워크를 추가 개발하였다.

단위 테스트를 거쳐 Cross the bridge ! 프로젝트를 통합하면서 전체적인 완성도를 높여나갔다.

* 관련 자료

부록 4.1. - 상세 내용 확인

RPEngine 단위 테스팅 영상
https://www.youtube.com/playlist?list=PLFvdltGhnoAOEi8856S_SpO5De_5REK_o
Project 소개 영상 : https://youtu.be/LxMKwNS4k1Y

개발 엔진 Open Source Project : https://github.com/Woohyun-Kim/RPEngine
개인 도서 번역 : http://macgom.tistory.com/entry/Game-Physics-Engine-Development

참고 자료

Yan-Bin Jia. Quaternions and Rotations*. Com S, 477/577 Notes, 2015.
Ian Millington. (2014). Game Physics Engine Development, elsevier, San Francisco

2-2. 작품 명	Friendly Farm	역할 및 팀	FarmSystem - 김우현(PL) Unity3D UI - 윤주현 DefenseSystem - 이원철 Server & DB- 송영록

본인 개발 파트	- Farm System 개발	기간	'14.03.01 ~ '14.09.01 (6개월)
개발 환경	Windows 7	개발 도구 및 언어	C#, iKVM, Java, NGUI 2.7.0, Unity 3D 4.3,

개요

Friendly Farm은 농사와 디펜스 게임을 통해, 농업에 대한 지식을 함양하고 동시에 재미를 느낄 수 있는 게임이다.

내용

1. 개발 동기 및 목적과 목표

1.1. 개발 동기

현대 사회는 산업화, 공업화를 통해 많은 발전을 이루었지만, 인류의 농업에 대한 의존성이 축소되지는 않았다. 그러나 요즘 우리 세대는 농업에 대한 관심이 줄어든 실정이다. 우리가 먹는 음식은 대부분 농업과 관련이 깊으므로 여전히 농업은 우리 생활의 일부분을 차지할 것이다.

이에 우리는 농업에 대한 관심을 유도하고 농업과 관련된 지식을 함양시키기 위해, 교육적인 목적과 재미를 동시에 만족시킬 수 있는 3D 게임을 개발하기로 하였다.

1.2. 개발 목적

친구들과 함께 즐길 수 있는 Social Network Game(SNG)으로 개발한다. 교육이 이루어질 수 있도록 농사 시스템을 개발한다. 흥미를 유발하기 위해, 농사 시스템과 연동되는 농사 디펜스 게임을 개발한다.

1.3. 개발 목표

- 교육 : 농촌진흥청의 자료를 토대로 개발한다.
- 흥미 : 농작물을 통한 디펜스 게임을 개발하여 농사와 함께 즐길 수 있도록 개발한다.
- SNG : 데이터베이스 서버를 구축하여 친구들과 함께 플레이 할 수 있도록 개발한다.

1.4. Farm system 개발 주요 명세

Friendly Farm
☞채소/과일 등 다양한 작물을 재배할 수 있으면 좋겠다. →다양한 작물 자료를 조사하여 해당 데이터를 토대로 구현한다.
☞실제 농작물의 정보를 토대로 농작물 및 농사에 대한 지식을 습득할 수 있었으면 좋겠다. →실제 농작물 자료 및 농사법 자료에 근거한 데이터를 그대로 사용한다.
☞계절, 날씨, 온도 등 환경적 요소를 추가하여 실제와 좀 더 유사했으면 좋겠다. →계절별 환경, 날씨(맑음, 흐림, 눈, 비, 번개, 소나기), 온도, 습도 및 토질 속성을 부여한다.
☞병충해/재난/잡초 등의 방해 요소가 있으면 리얼하고 재미있을 것 같다. →게임 내 환경적 상태에 따라 방해 요소가 나타날 확률을 정의한다.
☞가상의 지역에서 농작물을 관리하면서 자신의 자산을 불려나가면 재미있을 것 같다. →자신의 정보는 DB서버로 저장. 농작물 수확을 통해 자산을 불려나갈 수 있도록 구현한다.
☞새로운 농작물을 키우면서 농작물 사전을 확인할 수 있다면 재미있을 것 같다. →농작물 사전을 구현한다.
☞농사가 지루하지 않기 위해 플레이어가 시간의 흐름의 속도를 제어할 수 있으면 좋겠다. →시간에 대한 배속 기능을 추가한다.

* 관련 자료

부록 4.2. - 상세 내용 확인

시연 영상 : https://www.youtube.com/playlist?list=PLFvdltGhnoAPDOS2YjjSNd1FlrL-JpJZt

본인 개발 소스코드 별도 파일 첨부

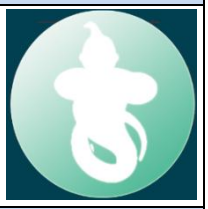
- **Java :** ./Friendly Farm/FriendlyFnarmSystem_ver1.03.zip 참조
- **C# Scripts :** ./Friendly Farm/farmingScr.zip 참조

2-3. 작품 명	SmartVoice "Android Jellyboys 4.3"	역할 및 팀	김우현 노진석(PL) 문용호

본인 개발 파트	- Smart Voice System 개발 - System Application 개발	기간	'14.11.01 ~ '15.03.01 (4개월)
개발 환경	Ubuntu 12.04 LTS, Android 4.3 Jellybeans, Windows 7, Galaxy Nexus	개발 도구 및 언어	C++, Java Visual Studio2010/2013, Eclipse Kepler, Git, Makefile

개요

Jellyboys 프로젝트는 기존의 안드로이드 Jellybeans 운영체제에 음성 기반 시스템을 추가하여 개발하는 프로젝트이다. 이를 통해 Jellyboys 버전의 안드로이드가 탑재되는 모든 기기들에 사용자의 음성을 분석하는 시스템 인터페이스를 구축하는 데에 의의를 두고 있다.



내용

1. 개발 동기 및 목적과 목표

1.1. 개발 동기

사용자 인터페이스의 궁극적인 목표는 사용하기 편리함이다. 일상 생활에서 사람들이 쉽게 소통하는 도구로는 음성이 있다. 미래 IoT시대에는 모든 사물들이 음성을 통해 서비스를 탐지하고 수행할 필요성이 있을 것이다. 따라서 우리는 안드로이드 운영체제를 수정하여, 3rd Party 개발자들에게 이러한 편리한 음성을 이용하여 쉽게 사물을 조작할 수 있는 인터페이스를 제공하는 시스템을 개발한다.

1.2. 개발 목적

실시간 음성 처리를 통해 사용자 음성으로 사물을 제어하거나 사물이 서비스를 제공하도록 Android OS에 음성 제어 시스템을 추가한다. 이를 통해 Android Application 개발자, Android Framework 개발자에게 음성 제어 시스템을 사용할 수 있도록, SDK를 통해 제공하도록 한다.

1.3. 개발 목표

- Android OS의 아키텍쳐 분석을 통해 음성 시스템을 설계 및 구축하여 이식한다.
- 음성 처리, 자연어 처리를 통해 서비스를 각자의 정책에 맞게 호출하는 모듈을 개발한다.
- 3rd Party / Framework 개발자에게 음성 서비스를 개발할 수 있는 인터페이스를 개발한다.
- 개발된 음성 시스템을 통해, 최종 사용자가 다양한 음성 기능을 실제 활용할 수 있도록 한다.

* 관련 자료

부록 4.3. - 상세 내용 확인

Project 소개 영상 및 시연 영상

https://www.youtube.com/playlist?list=PLFvdltGhnoANUmPkSl9IVSECUSKcswDoK

본인 개발 소스코드 별도 파일 첨부

Porting Guide : ./Smart Voice/소스코드/Android JellyBoys - 설명서.docx

Source Code : ./Smart Voice/소스코드/Android Framework.zip

3. APPENDIX

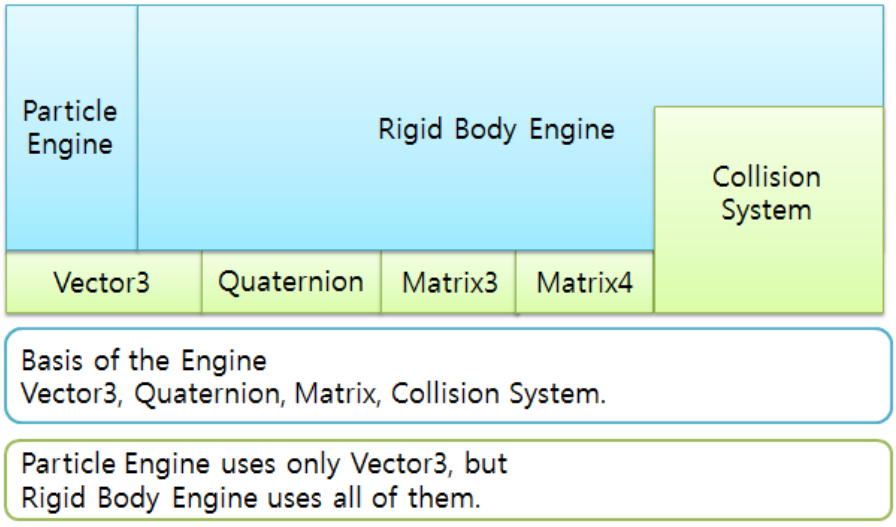
3-1. Cross the bridge!

1. 물리 엔진 개발

1.1. 주요 개발 내역

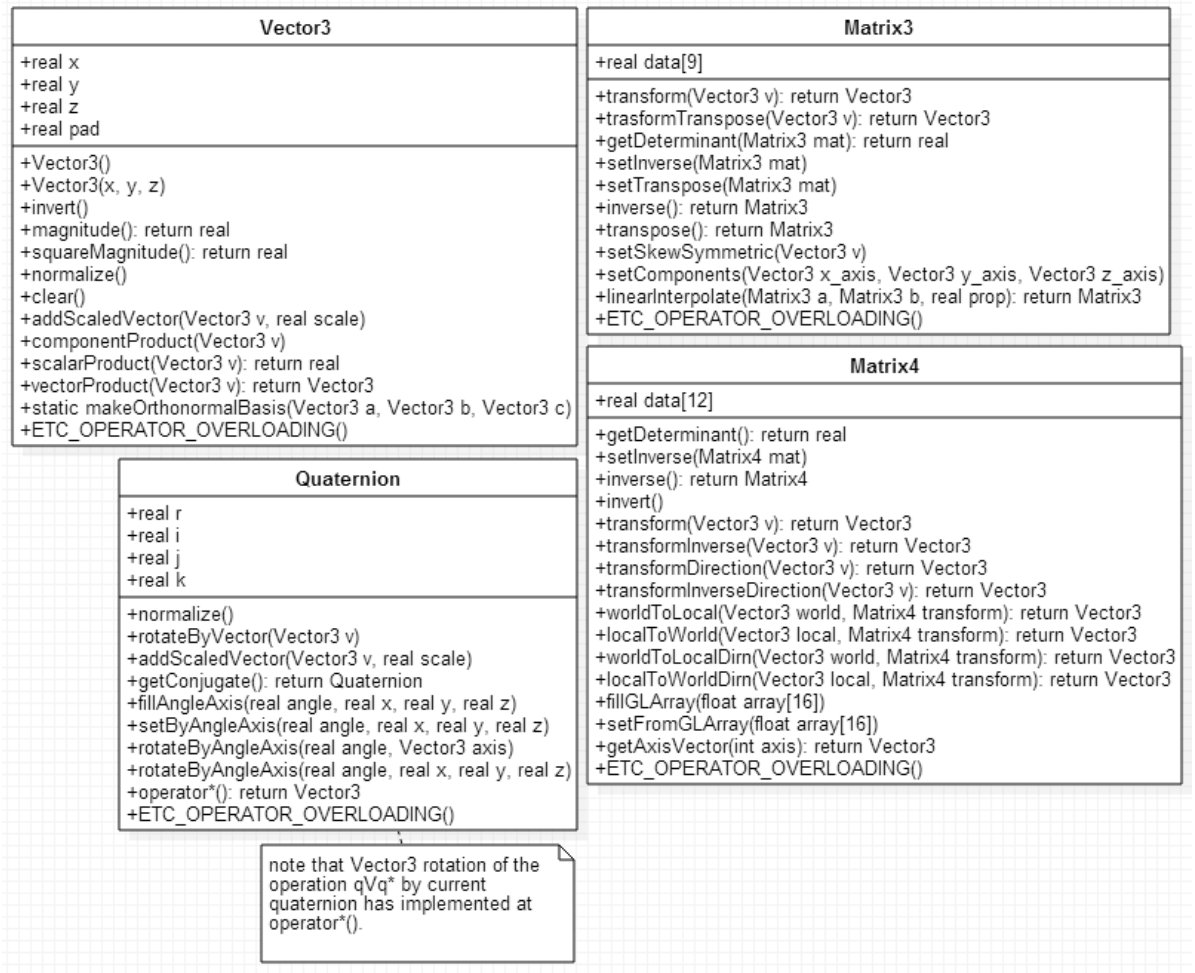
구현 리스트	설명
Basic	
Vector3/Matrix/Quaternion	좌표 공간 상의 기본 자료구조 및 연산 정의
운동 방정식	미적분을 대체하기 위한 뉴턴 동역학 구현
입자 물리 엔진	입자(Particle)의 단순 병진 운동을 구현하는 간이 엔진
3D Real Space	
물체의 회전 연산	병진 운동과 더불어 입자가 아닌 물체의 회전 상태 및 회전을 표현하기 위한 행렬, 사원수 알고리즘 구현
강체 물리 엔진	실제 세계의 사물들의 물리적인 표현이 가능한 엔진
Collision Detection	
충돌 탐지	경계 물체(Box, Sphere, Plane) 간 충돌 탐지 시스템 SAT 알고리즘 구현(차원 축소를 통한 탐지)
충돌 해결	충돌한 물체의 데이터 처리 시스템(Resolver)
충돌 이벤트	엔진 사용자의 충돌 탐지 관련 인터페이스 (Callback을 통한 구현)

1.2. 계층도(SW Hierarchy)



1.3. SW Architecture & Description

1.3.1. Base



1.3.1.1. Vector3

벡터 자료구조는 좌표공간 상의 x, y, z의 실수 데이터와 pad로 이루어져 있다.
Pad는 성능상의 이슈로 추가된 변수로, CPU의 Word-size로 인한 성능을 보완하기 위해 존재한다.
멤버 함수로는 벡터의 기본 연산 외에도 자주 사용되는 추가적인 연산 알고리즘을 포함시켰다.
그 외 연산자 오버로딩을 통해 사용하기 쉽도록 구성하였다.

1.3.1.2. Matrix3

엔진 내부에서는 실제로 관성텐서 값을 표현하고 회전 알고리즘에 사용하기 위해 구현되었다.
벡터와 마찬가지로 기본적인 행렬 연산과 자주 사용되는 연산 알고리즘을 포함시켰다.

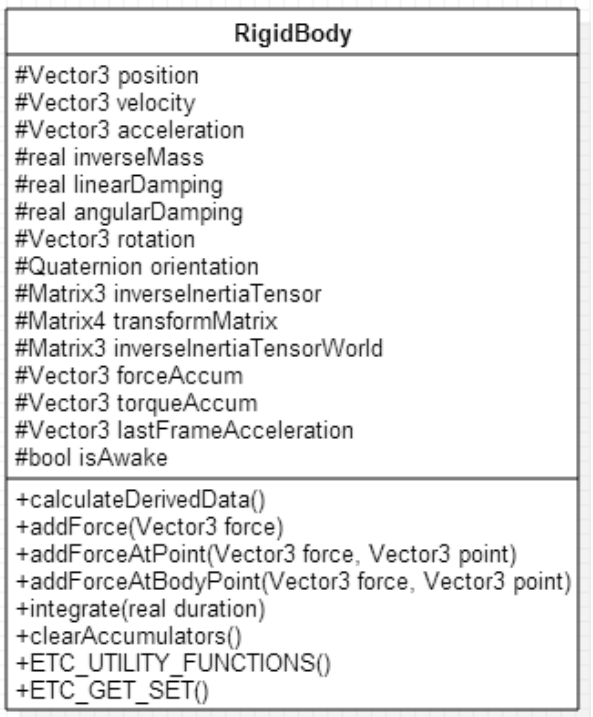
1.3.1.3. Matrix4

강체(물체)의 위치와 회전 상태를 표현하기 위한 자료구조이다. 4행의 데이터는 항상 (0 0 0 1)이므로 필요하지 않아 3x4 행렬로 구현하였다. 이는 OpenGL 행렬과 다른 정책이므로, 4x4로의 변환을 하는 함수를 추가하였다. 역시 기본 연산과 자주 사용되는 연산(MCS<->WCS)을 추가 구현해 두었다.

1.3.1.4. Quaternion

사원수는 충돌 알고리즘 내에서의 회전 연산을 간소화하기 위해 구현하였다. 회전 행렬 상에서의 계산을 하는 것 보다 사원수를 통해 회전을 표현하고 계산하는 것이 좀 더 빠른 성능을 나타낸다. 또한 짐벌락 문제를 해결할 수 있다. 사원수의 기본 연산과 더불어 사용자들이 사용하기 쉽도록 회전 관련 함수들을 포함시켰다.

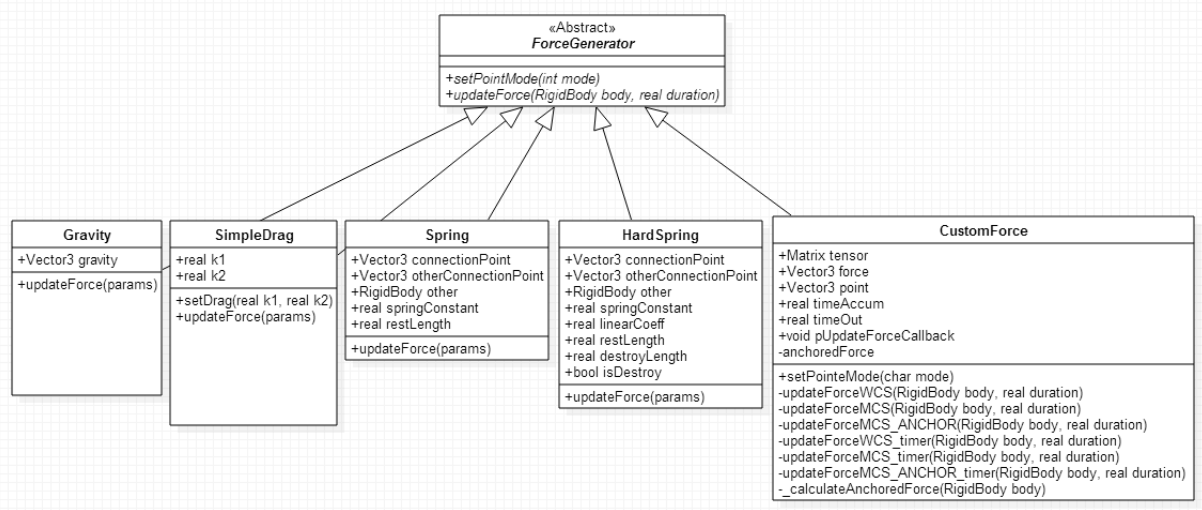
1.3.2. RigidBody



개발한 물리 엔진에서는 모든 물체를 '강체'로 간주한다. 강체란 외력이 가해지더라도 물체의 모양이 변하지 않는 물체를 의미한다. 정지된 시점에서의 프레임에서 강체의 위치, 속도, 가속도, 질량, 회전력에 대한 관성을 의미하는 관성텐서, 회전상태를 표현하기 위한 회전 행렬로 기본 물체의 속성을 포함한다. 또한 물리적 계산을 위해 특정 시점에서의 물체에 가해진 외력을 누산하기 위한 변수 등이 포함되었다. 해당 강체에 외력을 가하는 함수들과, 누산된 외력으로 강체의 상태를 계산하기 위한 함수 등이 추가되었다.

한 편, 강체는 자신의 기하적 모양과 크기에 대해 기술되어 있지 않다. 그 이유는 강체의 모양과 크기는 단순히 사용자 수준에서 자유롭게 렌더링 하는 것이기 때문이다.

1.3.3. ForceGenerator

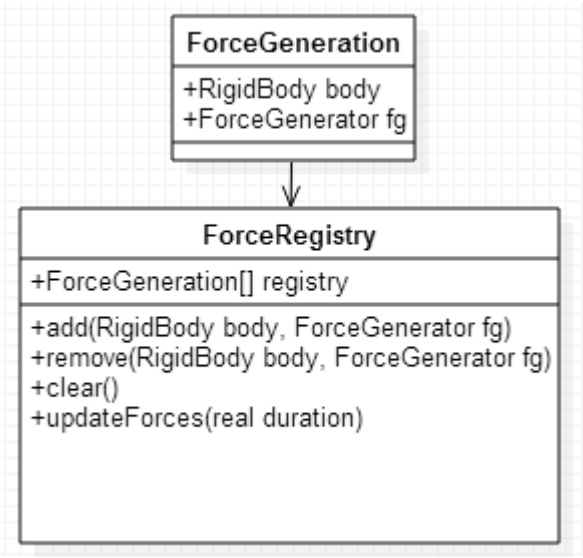


ForceGenerator는 현실 세계에서 볼 수 있는 다양한 외력들을 구현하기 위해 만들어진 인터페이스이다. 대표적인 외력으로는 중력을 예로 들 수 있다. 강체에 중력을 가하는 알고리즘은 updateForce 함수 내부에서 구현하고, 추후에 ForceGenerator 인터페이스의 updateForce함수 호출을 통해 간접적으로 강체에 중력이 누산된다. 따라서 강체는 한 순간에 다양한 외력을 받아서 누산될 수 있으며, 이는 물리학에서 말하는 달랑베르의 원리를 구현한 것이다.

이 외에 SimpleDrag는 저항력 알고리즘을 간단하게 구현했으며, Spring은 용수철과 같은 탄성력, HardSpring은 Cross the bridge 프로젝트에서 건설 자재 간의 장력을 표현하기 위한 알고리즘을 기술하였다.

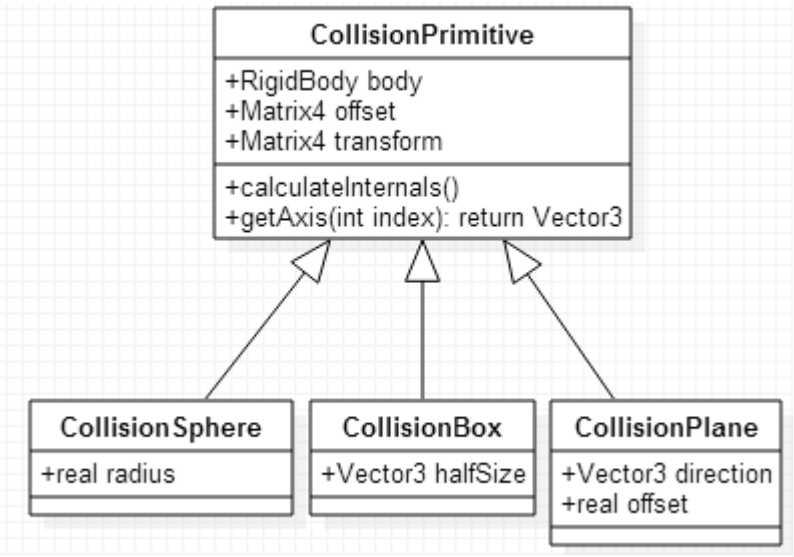
마지막으로, CustomForce는 외부 사용자가 엔진에서 제공하지 않는 다른 외력을 추가적으로 구현할 수 있도록 만든 함수이다. 자신이 구현할 외력 알고리즘을 기술한 후, pUpdateForceCallback로의 콜백 함수를 연결시키면 엔진 사용자의 Custom Force를 적용할 수 있도록 배려하였다.

1.3.4. ForceRegistry



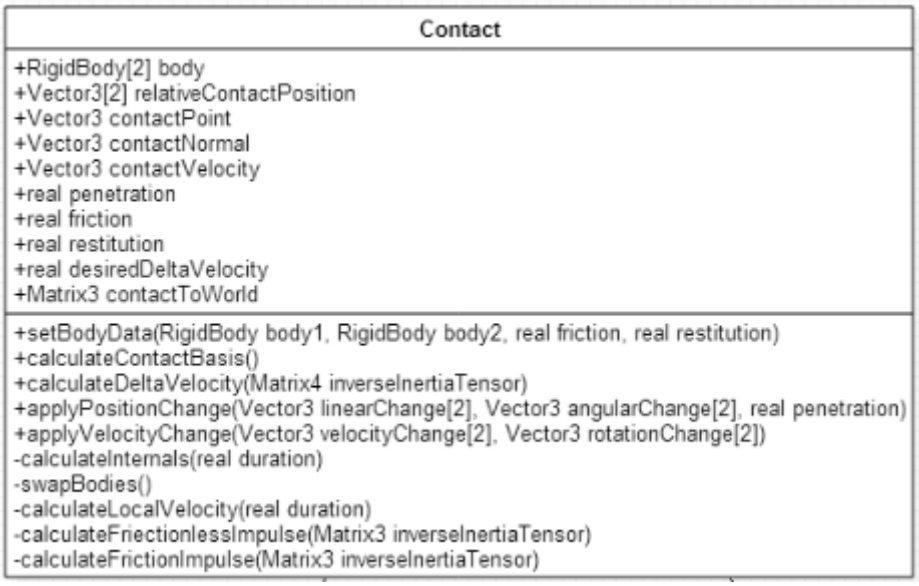
ForceRegistry는 외력과 강체를 등록하여 관리한다. ForceRegistry의 updateForces() 함수는 등록된 외력 F를 duration 시간동안 해당 강체에 가한다. 물론 updateForces는 내부적으로 해당 ForceGenerator.updateForce() 함수에 강체 인자를 넣어 호출하는 것이다. 이로써 강체에 외력을 가하는 구조가 완성된다.

1.3.5 CollisionPrimitive



강체의 충돌체 모양을 결정짓기 위한 인터페이스이다. 따라서 충돌체는 멤버변수로 강체를 포함한다. 지금까지 구, 박스, 평면 세 가지만 구현하였다. 충돌체의 모양이 세 가지 뿐이지만, 강체에 대한 렌더링은 사용자의 몫이므로 물체를 화면에 그려내는 데에는 문제가 되지 않는다. 충돌체를 구현한 이유는 렌더링이 아닌, 엔진 내부적으로 물체 간 접촉에 의한 충돌 결과를 기술하기 위해서이다.

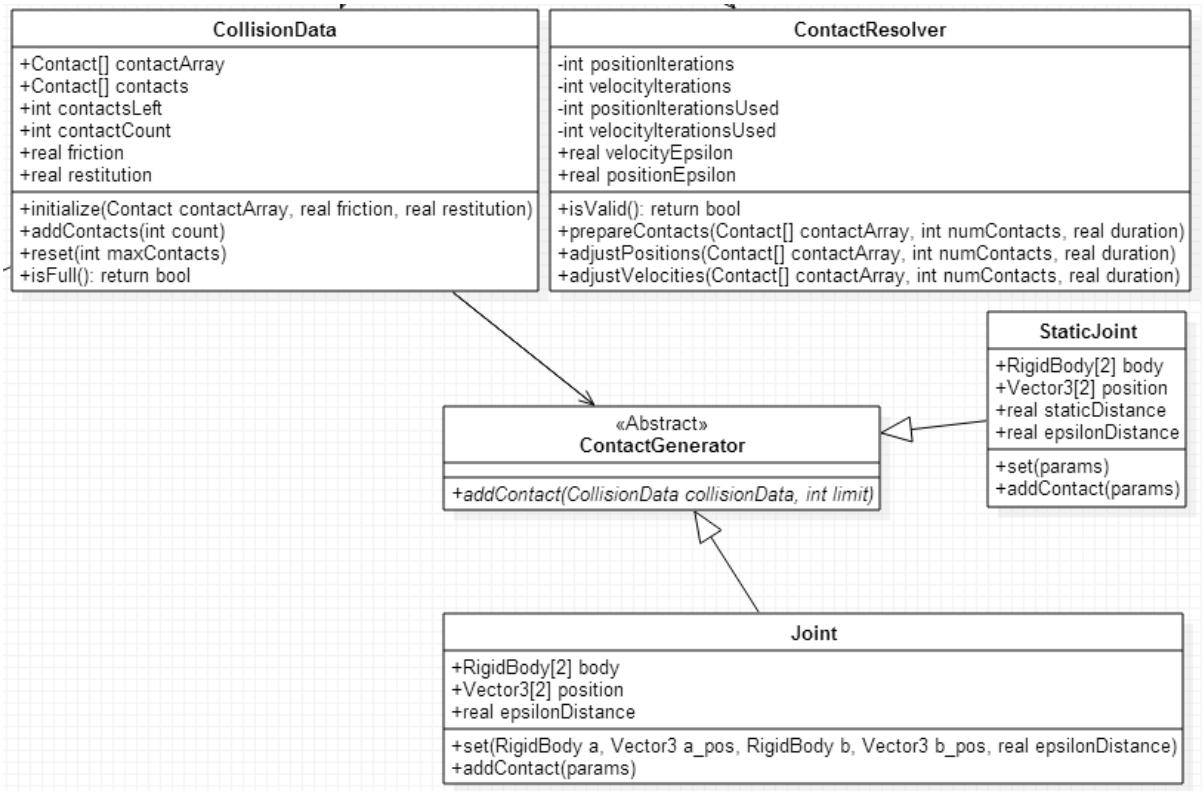
1.3.6. Contact



만약 어떤 두 강체가 충돌한다면, 그 충돌을 해결해야 할 것이다. 그렇다면 충돌을 해결하기 위해서는 충돌 해결을 위한 데이터가 필요하다. Contact는 어떤 두 물체가 충돌했을 때의 자료를 저장하기 위한 것이다. 어떤 강체들이 부딪혔는지, 충돌 지점의 위치, 충돌 방향과 충돌 속도 그리고 마찰계수에 대한 정보가 필요할 것이다. 또한 컴퓨터에서는 충돌을 통해 두 물체가 얼마나 서로를 관통했는지에 대한 정보도 필요하다.

이 후, 충돌 관련 데이터를 해결하기 위한 함수들이 구현되었다. 충돌 결과 두 물체간 운동에너지를 주고 받으며 서로 다른 병진운동과 회전운동을 일으킨다. 운동에너지 보존법칙에 의해 각 운동에 대한 강체의 위치 및 속도에 대한 알고리즘이 포함되었다.

1.3.7. CollisionData



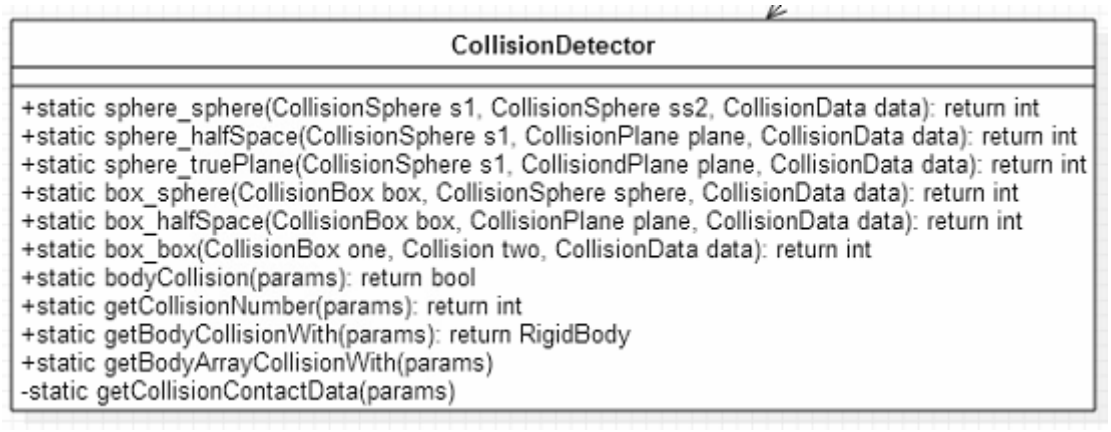
충돌 데이터(Contact)들을 관리하기 위한 구조가 필요하다. 이를 위해 CollisionData는 물체간의 충돌 데이터들을 담아 관리한다. 또한, 충돌 물체가 너무 많으면 성능이 떨어질 수 있으므로, 충돌 수를 카운트하는 역할을 수행한다.

ContactResolver는 함수의 인자로 받는 충돌 데이터들을 토대로 충돌을 해결하도록 지시하는 역할을 한다. 충돌 데이터가 쌓여있으면, 충돌 데이터를 재가공한 후 Contact 내부 충돌 해결 함수들을 호출하여 간접적으로 충돌 결과를 처리하는 구조이다.

ContactGenerator는 ForceGenerator와 비슷한 역할을 담당한다. 그러나 ForceGenerator가 외력을 구현하기 위해 존재했던 것과는 달리, ContactGenerator는 특별한 충돌을 구현하기 위해 존재한다. Joint 충돌은 두 물체가 서로 붙어있는 것처럼 행동하기 위해 만들어졌다. 따라서 두 물체는 마치 하나의 물체인 것처럼 행동한다.

StaticJoint는 Cross the bridge 프로젝트에서 다리 건설의 시작 위치와 도착 위치에 건설 자재를 절대적으로 고정시키기 위해 만들어졌다. Joint와는 약간 다르게, 한 물체가 특정한 위치에 절대적으로 매달려 움직이지 못하는 것을 구현하기 위해 추가적으로 구현된 충돌이다.

1.3.8. CollisionDetector

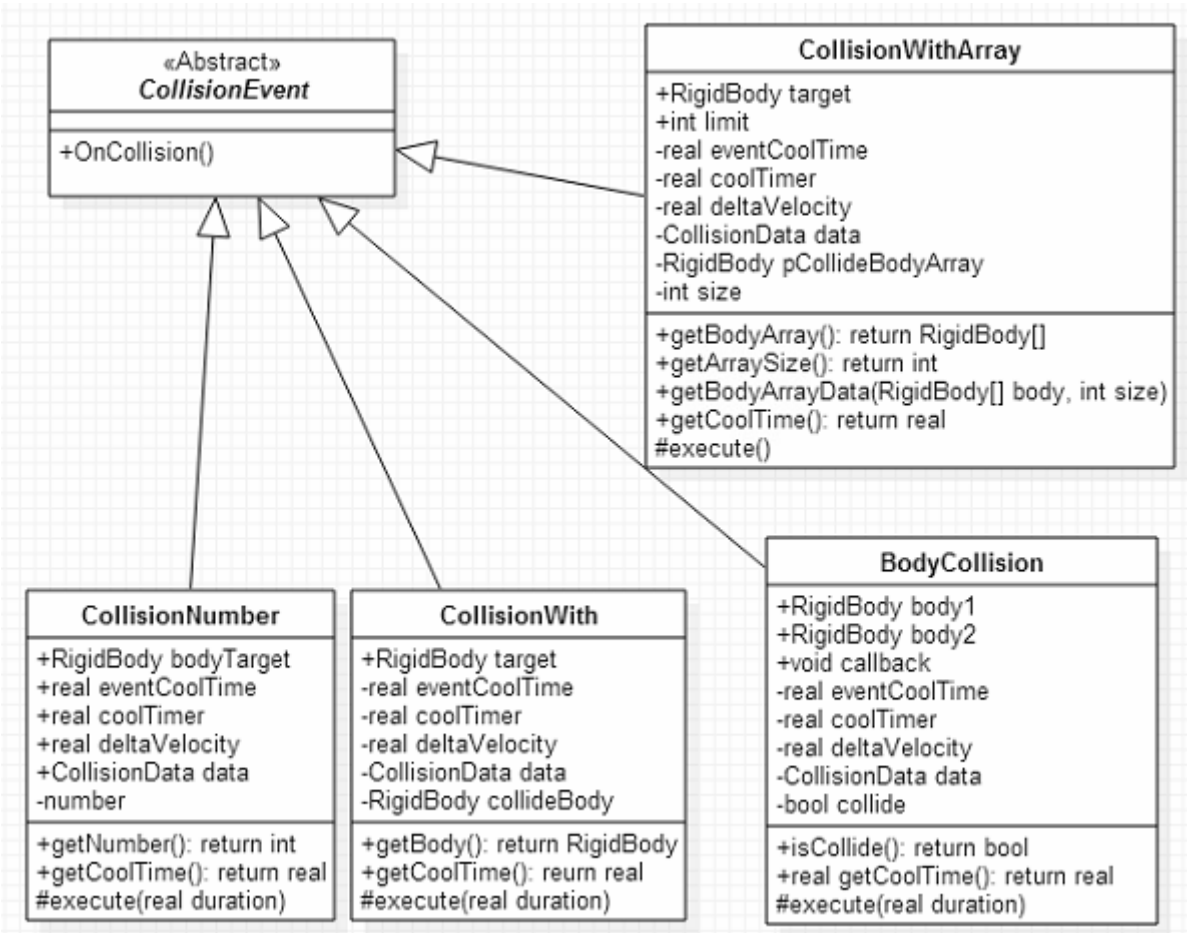


정지된 시점에서, 먼저 운동 중인 물체들이 충돌 한 상태인지 확인해야 한다. CollisionDetector는 충돌체들이 충돌한 상태인지 판별하고, 충돌이 탐지되었다면 충돌 데이터(Contact)를 생성해낸다. 생성된 데이터는 CollisionData에 추가된다. 충돌 탐지 알고리즘은 구-구, 구-평면, 구-박스, 박스-평면, 박스-박스 충돌이 구현되어 있다.

충돌 알고리즘은 SAT(차원축소를 통한 탐지)로 구현하였다. SAT알고리즘은 볼록 다면체에 대한 충돌에 대해서도 검출이 가능하므로 충돌체(CollisionPrimitive)의 확장성을 고려한 선택이었다.

한편 물리 엔진의 사용자가 충돌 상태를 확인하고, 충돌 시에 자신이 구현한 알고리즘을 수행해야 할 필요성이 있는데, 이를 수행하기 위해 충돌 여부를 사용자에게 알리기 위한 함수들이 구현되었다.

1.3.9. CollisionEvent



CollisionEvent는 사용자들이 충돌 이벤트를 받아볼 수 있도록 하기 위해 구현하였다. 또한 사용자가 특정 충돌 이벤트가 발생하기를 원하는 시점이 있다면, 해당 이벤트에 콜백 함수를 등록하도록 구현하였다.

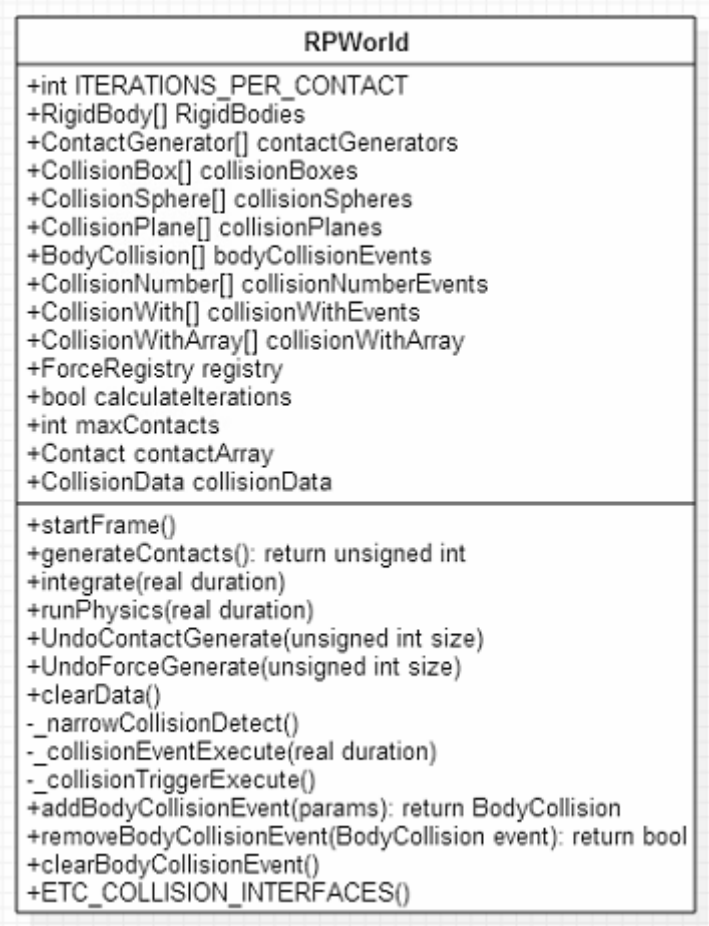
BodyCollision은 특정 두 강체 간의 충돌을 주시할 때 사용한다. 사용자가 주시하고자 하는 두 강체를 등록하면, 두 물체가 충돌할 때마다 콜백 함수를 호출한다. Cross the bridge 프로젝트에서는 금속 건설 자재가 특정 위치에 부딪히는 경우 굉음을 내는 것을 구현하기 위해 사용되었다.

CollisionNumber는 특정 강체가 다른 물체들과 동시에 충돌한 개수를 확인하기 위한 것이다. Cross the bridge에서는 동시 충돌이 너무 많은 경우 충돌 사운드 발생이 매우 큰 점이 있었는데, 이를 단순화하여 해결하기 위해 사용되었다.

CollisionWith는 특정 강체가 다른 물체와 충돌했을 때, 충돌된 대상을 확인하기 위한 것이다. BodyCollision을 좀 더 유연하게 만든 것으로, 사용자 수준에서 충돌된 대상을 알 수 있도록 만들었다.

CollisionWithArray는 특정 강체다 다수의 다른 물체들과 동시 충돌한 경우를 탐지하기 위한 것이다. CollisionWith를 확장한 것으로, 사용자가 동시 충돌된 다수의 대상을 알 수 있다는 장점이 있다.

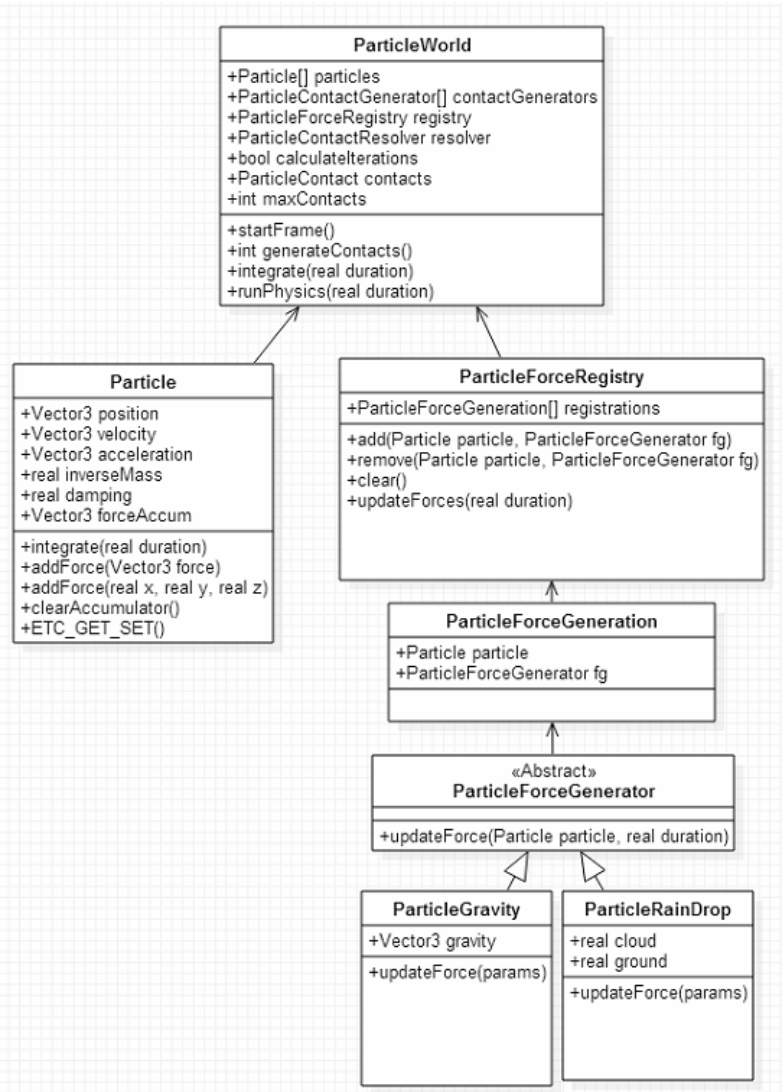
1.3.10. RPWorld



물리 엔진을 사용하고 제어하기 위한 Controller이다. 사용자는 물리 세계를 시뮬레이션 하기 위해 이 곳에서 물체를 등록하고 외력과 충돌 등을 등록할 수 있다. 물리 세계를 시작시키기 위해서는 runPhysics() 함수를 매 프레임마다 호출한다.

그러면 물체들에게 외력이 작용하고, 외력에 의해 물체들은 운동을 하게 된다. 그러면 충돌이 탐지되고, 충돌 데이터를 처리하여 다시 물체들의 상태를 변화시킨다. 만약 충돌 이벤트가 등록되었다면 콜백 함수를 통해 이벤트를 처리해준다. 프레임마다 이러한 전체적인 과정을 반복하면서 물리 세계를 시뮬레이션한다.

1.3.11. PWorld

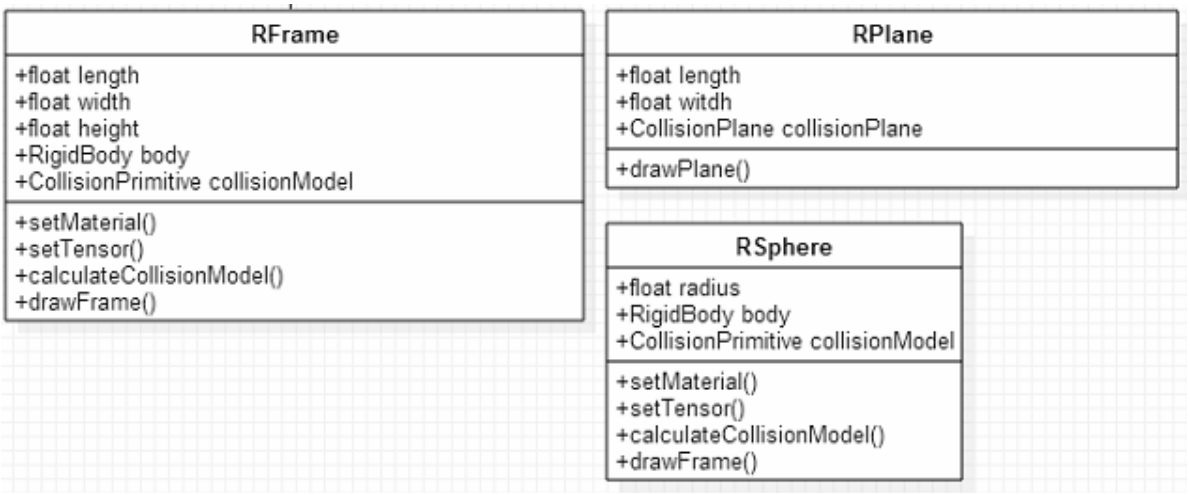


입자 물리 엔진도 구현하였다. 강체 물리 엔진과 동일한 설계를 따른다. 차이점은 입자 물리 엔진은 RigidBody를 사용하지 않는 대신, Particle을 사용한다. 입자는 회전하지 않는 것으로 간주한다. 또한 입자 엔진에서도 충돌은 가능하지만 구현하지 않았다. Cross the bridge에서의 입자 엔진은 단순히 비가 오는 장면을 시뮬레이션하기 위해 사용하였다.

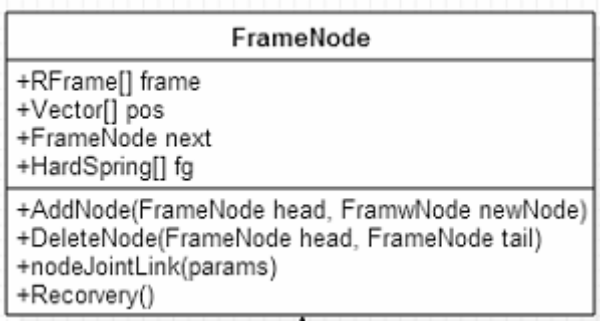
1.4. 엔진 상위 API

내가 만든 범용 엔진은 Cross the bridge 프로젝트 진행의 가장 하위 작업이었다. 실제 다리 건설 시뮬레이션 프로그램에 적용하기 위해서는 다리 건설과 관련된 좀 더 준비된 작업이 필요하였다. 그래서 내가 만든 엔진을 기반으로 하는 간단한 건설용 물리 API를 추가 개발하기로 하였다.

1.4.1. Wrapped Objects

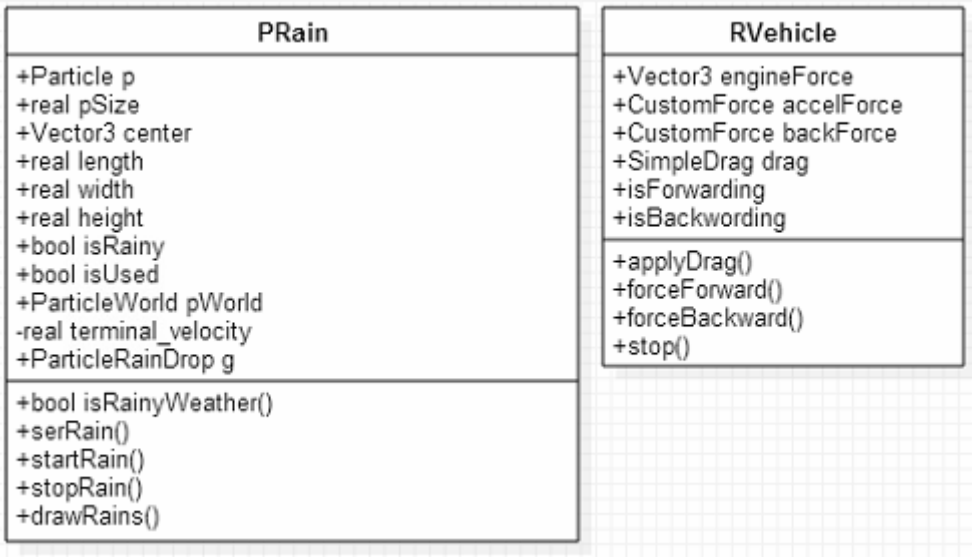


강체와 충돌체를 합하고, 관성텐서 행렬을 조사해서 물체의 크기, 모양 등이 변하면 내부 데이터를 스스로 계산하는 클래스들을 추가하였다. Cross the bridge의 건설 자재는 RFrame을 통해 육면체로 사용하게 되며, 사용자가 육면체의 크기와 모양을 변화시키더라도 내부적으로 물리적 속성을 재계산할 수 있도록 구현하였다. 이제 간단한 물체를 만들 때에는 RFrame, RPlane, RSphere만으로 구현할 수 있게 되었다.



건설 자재를 통해 다리를 건설할 때에는 HardSpring이라는 외력으로 자재들을 연결하였다. 탄성력을 조금 변형하여 좀 더 탄탄하게 구현하기 위함이었다. 사용자가 자재들을 배치할 때마다 외력으로 건설 자재를 연결하기 위해서 연결리스트(LinkedList)를 통해 알고리즘을 구현하였다.

1.4.2. Real Objects



Cross the bridge에서 입자 엔진을 통해 비가 오는 장면을 시뮬레이션 하기 위해 PRain을 추가하였다. 빗방울 입자의 개수, 비가 오는 위치 등을 설정할 수 있다. 랜덤함수를 통해 입자들마다의 위치를 결정하여 중력을 받아 떨어지도록 구현하였다. 빗방울이 종단 속도에 도달하게 되면 더 이상 빗방울은 가속되지 않는다.

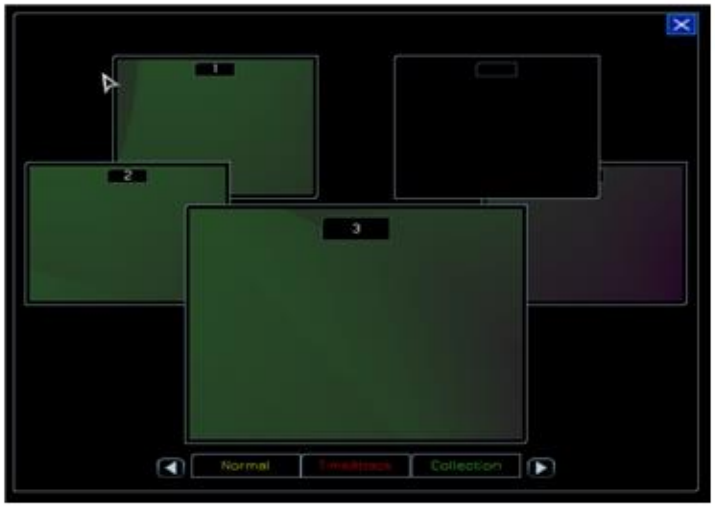
RVehicle은 Cross the bridge에서 다리를 건널 차량을 간단하게 구현하기 위해 만들었다. 자동차의 무게, 엔진의 힘, 전진 및 후진에 대한 가속력을 설정할 수 있다. SimpleDrag를 통해 차체가 저항을 받아 언젠가는 다시 멈추도록 구현하였다.

2. 결과

2.1. Cross the bridge!

Game Intro 화면

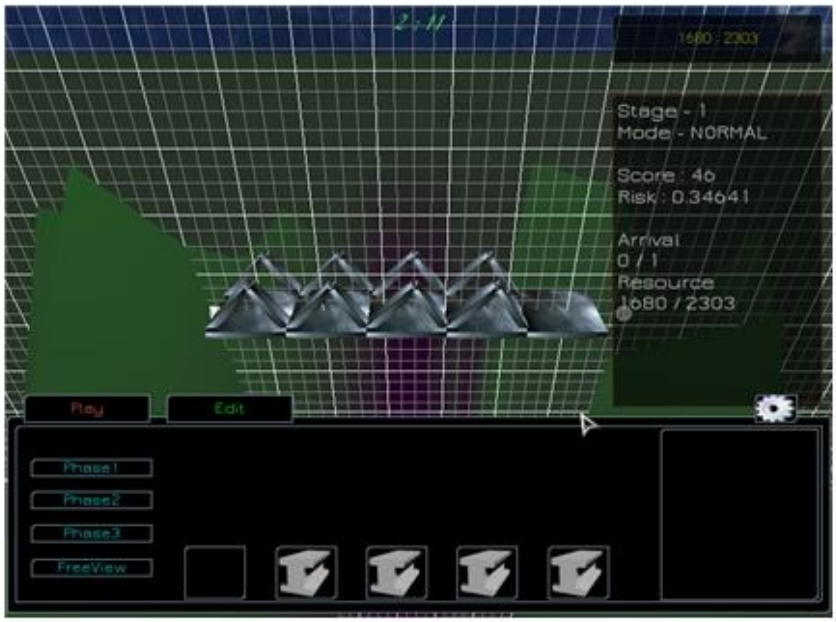
Cross the bridge 는 다양한 모드를 지원한다. Scenario 모드는 게임의 시나리오에 따라 순차적으로 진행되며, Map Edit 모드에서 사용자가 직접 맵을 생성/수정 할 수 있다. Custom 모드의 경우에는 Map Edit 에서 사용자가 수정한 맵을 직접 플레이어가 선택하여 게임을 플레이 할 수 있기 위한 컨텐츠이나 현재는 Scenario 모드와 통합되어 있다.



시나리오 모드 맵 선택화면

초기 게임 개발 예상 기획과 같이 게임은 세 가지 모드로 분류하였다. 가장 안전하면서 적은 비용을 사용하는 다리를 건설하는 Normal 모드, 미션 아이템(오브젝트)을 획득하는 Collection

Mode, 제한된 시간 내에 클리어하기 위한 TimeAttack 모드가 존재한다. 이 중 실제 구현된 게임은 Normal 모드를 중심으로 개발하였다.



Edit Mode

게임을 시작하면 다시 Edit Mode 와 Simulation Mode 로 나뉘는데 Edit Mode 에서는 다리를 건설하고 Simulation Mode 에서는 건설한 다리를 테스트해 볼 수 있다. 매 스테이지에는 예산(제한된 자원)이 존재한다. Edit Mode 에서는 다리 건설의 재료인 프레임을 선택하여 다리를 건설 한 후에, Simulation Mode 를 통해 자동차(오브젝트)가 안전하게 건널 수 있도록 다리를 짓는 것이 최종적인 목표이다. 각 스테이지는 시작점과 도착점이 존재하며 두 지점을 연결하는 것으로 시작한다.

시작점과 도착점 사이에 존재하는 다수의 연결점들은 플레이어가 다리의 기반을 둘 수 있게 해준다. 통과하지 못함으로 인해 가해지는 벌칙은 없다. 다만 예산을 최대한 적게 사용하는 동시에 가장 안전한 다리를 지을 수록 더 좋은 보상과 더 높은 점수를 얻게 된다. 그리고 이 점수가 개인 최고 기록인 경우 데이터를 갱신한다. 플레이어는 성공적인 디자인을 찾기 전까지 다양하게 설계해보고 테스트 할 수 있다.

완전한 건설을 위해서는 세 단계(Phase)를 거친다. 첫 번째 건설은 다리의 기반이 되는 도로를 건설하며 두 번째 단계에서는 뼈대를 형성하고 마지막 단계에서는 뼈대들을 z 축 방향으로 보강할 수 있다. 현재로써는 밸런싱 작업이 제대로 이루어져 있지 않기 때문에 Phase1 에서 간단한 다리를 건설하는 것이 가장 좋다. 다리 건설이 끝나면 Play 버튼을 통해 게임을 시작할 수 있으며 여기에서 Start 버튼을 선택하면 자동차가 목표지점으로 가속한다.

플레이 화면

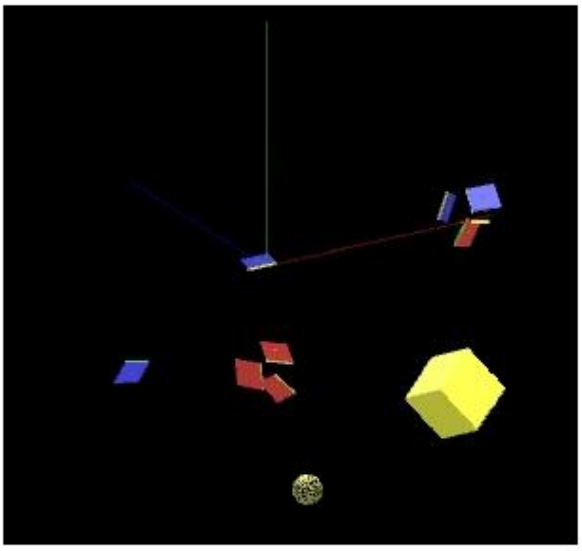
표 지점에 도달하게 되면 결과물로써 건설 자재 아이템을 획득하며, 아이템 마다 레벨이 있어서 약간씩 더 다른 물리적 시뮬레이션에 사용될 수 있도록 하는 것이 의도였으나 현재는 역시 게임 벨런싱 작업은 이루어지지 않았다.

건설 자재 아이템 드랍

그리고 아이템 드랍율은 게임 모드에 따라 확률 산출 방식이 약간씩 다르게 설정하였으며 현재 정상적으로 플레이할 수 있는 Normal Mode 에서는 안전성과 자원사용률의 50%의 비율로 산출된다. 그리고 만약 동일 종류의 아이템에서 더 낮은 레벨이거나 동일한 레벨의 아이템을 획득한 경우에는 해당 아이템 대신 사용자 추가자원을 획득하며, 이 자원은 플레이어 데이터에서 지속적으로 누적되므로 많은 시간 동안 플레이 한 플레이어일수록 더 크고 멋진 다리를 만들 수 있도록 하기 위해 추가 자원 획득으로 디자인하게 되었다.

2.2. 3D 물리 엔진 RPEngine(1.03)

오픈소스 물리 엔진인 Cyclone 의 개발자 Ian Millington 의 저서 GamePhysics Engine development 에서는 강체 물리 엔진 개발의 튜토리얼 격인 Cyclone 엔진의 개발과정과 노하우를 담고 있다. 저자는 자신의 개발 경험과 방식을 독자에게 물리 엔진 개발의 초석을 쌓도록 기여하는데 주력하고 있다. 그리고 이를 기초로 하여 RPEngine 1.03 버전이 탄생하게 되었다.



RPEngine 단위 테스트 장면

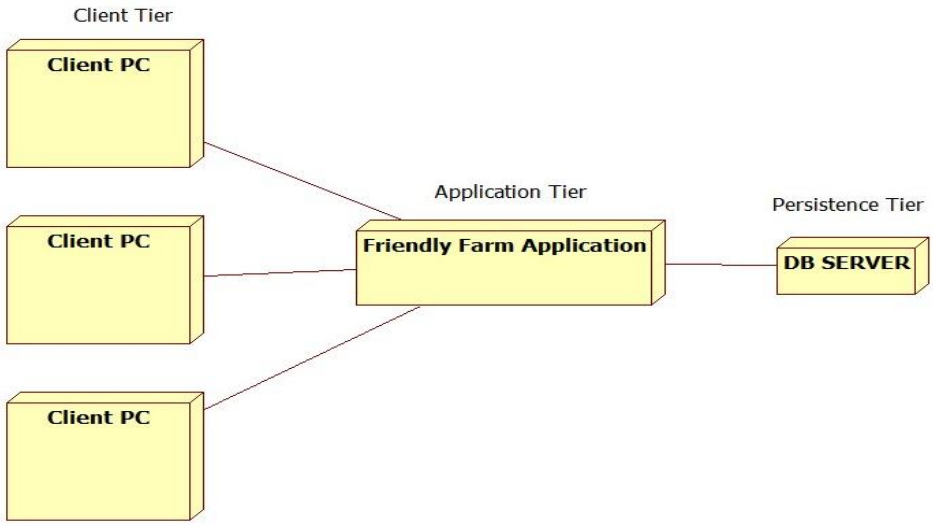
물리 엔진 라이브러리는 물리적으로 이성적인 결과를 도출하여야 하며 범용적이어야 하기 때문에 뉴턴의 고전역학을 기저로 하고 있다. 이를 위해 3D 좌표공간에서 사용하기 위한 기본 자료구조인 벡터(RPEngine::Vector3) 및 기본 연산(Scalar Product, Vector Product)들의 구현에서부터 개발을 시작하였다. 이 후 입자(Particle) 엔진 개발을 통해 물체의 성질인 위치, 속도, 가속도, 질량을 정의하고 각종 외력(중력, 공기저항력, 탄성력, 마찰력 등)에 의한 충돌의 병진운동을 구현한다. 이 후, 입자 엔진의 확장을 위해 Quaternion 및 Matrix 의 정의를 구현하였다. 사원수 및 행렬을 통해 회전운동을 구현하고 입자엔진은 비로소 강체 엔진으로 확장된다. 이제 실제 강체의 충돌이나 외력에 의한 움직임은 라이브러리 내에서 계산된다. 이러한 모든 계산은 뉴턴의 고전역학에 근거하므로 충돌하는 사용자가 정의하는 물체의 속성에 따라 연출되는 결과가 달라진다.

충돌 탐지의 기본 알고리즘은 SAT 차원축소 알고리즘으로 구현하였다. GJK 를 사용하지 않은 이유는 차원축소 알고리즘은 경계물체(혹은 유향상자)의 기하적 모형과는 독립적으로 탐지해 낼 수 있으므로 좀 더 범용적으로 사용할 수 있는 알고리즘이기 때문이다. 이 외에도 다양한 충돌 탐지 기법을 적용할 수 있는 방법이 존재한다. 추가 개발한다면 물리 엔진을 통해 개발하는 프로그램의 경험적인 특성에 기인하여, 보다 더 나은 성능을 내기 위해 물리 엔진에서 사용자가 알고리즘 정책을 채택할 수 있도록 구현하는 것으로 다른 엔진들과의 차별성을 두고 싶다.

3-2. Friendly Farm

1. 프렌들리 팜 개발

1.1. System Concept



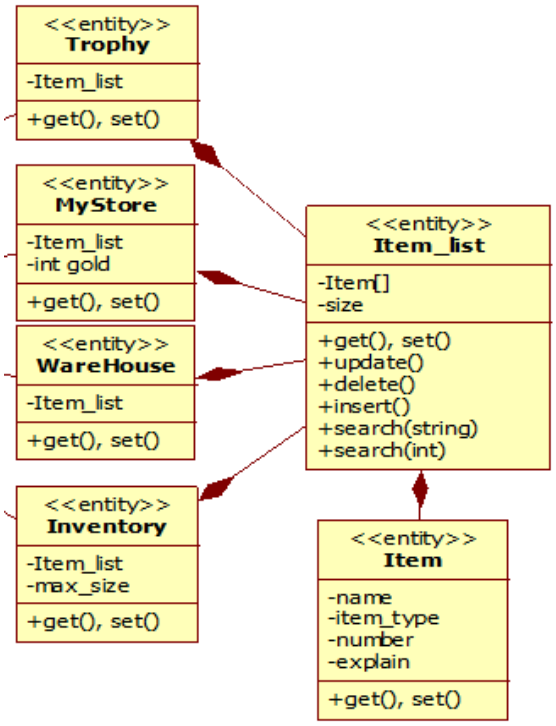
Social network service가 가능하기 위해 최소한의 데이터베이스 서버를 구축하고, 애플리케이션을 통해 다양한 유저들이 게임을 플레이 할 수 있도록 구현하였다.

따라서 플레이 정보를 구축한 데이터베이스 서버에 모두 저장할 것이다.

1.2. Farm 시스템 개발

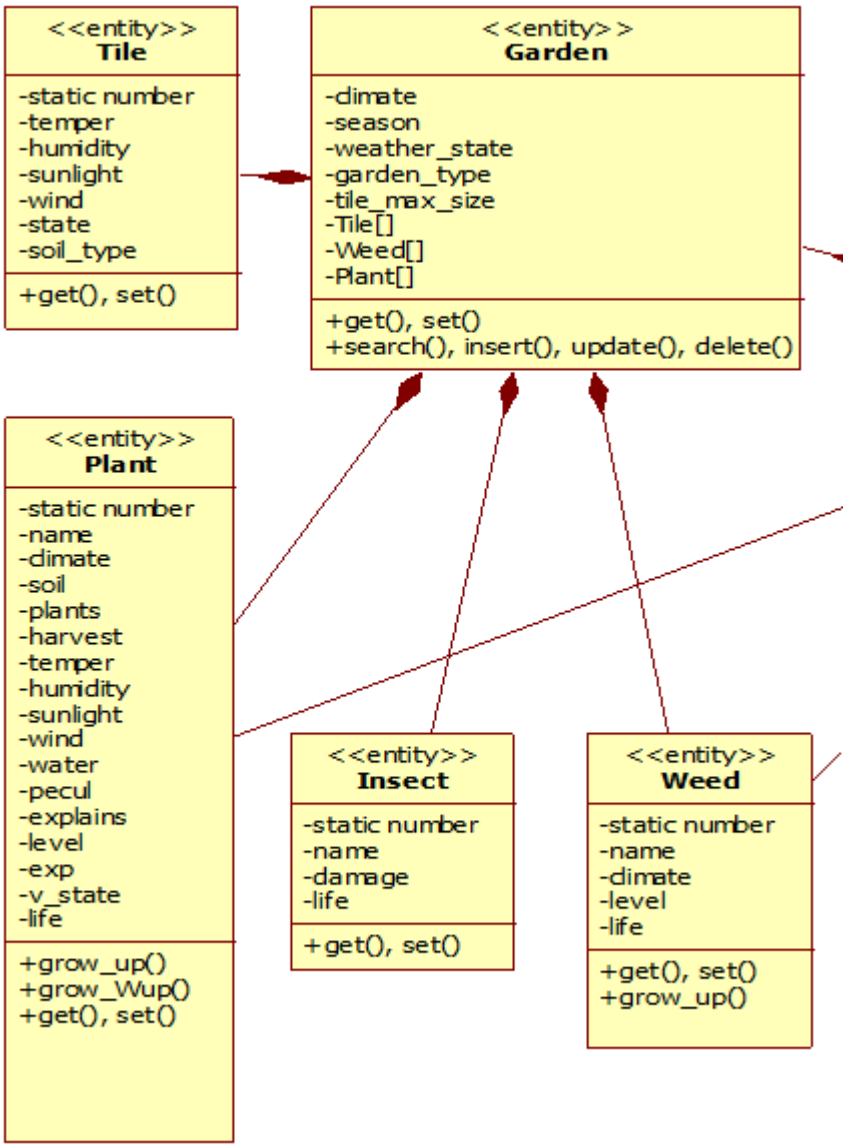
Unity3D의 농장 Scene에서 시스템을 제어하기 위해 Java에서의 구현 시스템을 이식한 Control 클래스를 C#스크립트로 구현하였다.

1.2.1 Entity Class



유저의 농사 관련 아이템을 저장하는 인벤토리(Inventory), 농장 수확물을 저장하기 위한 창고(WareHouse), 씨앗을 구매하기 위한 마켓(MyStore), 농사 정보가 정리된 도감 (Trophy)으로 분류하고, 공통적인 Item 클래스 및 Item_list를 사용하도록 구현하였다.

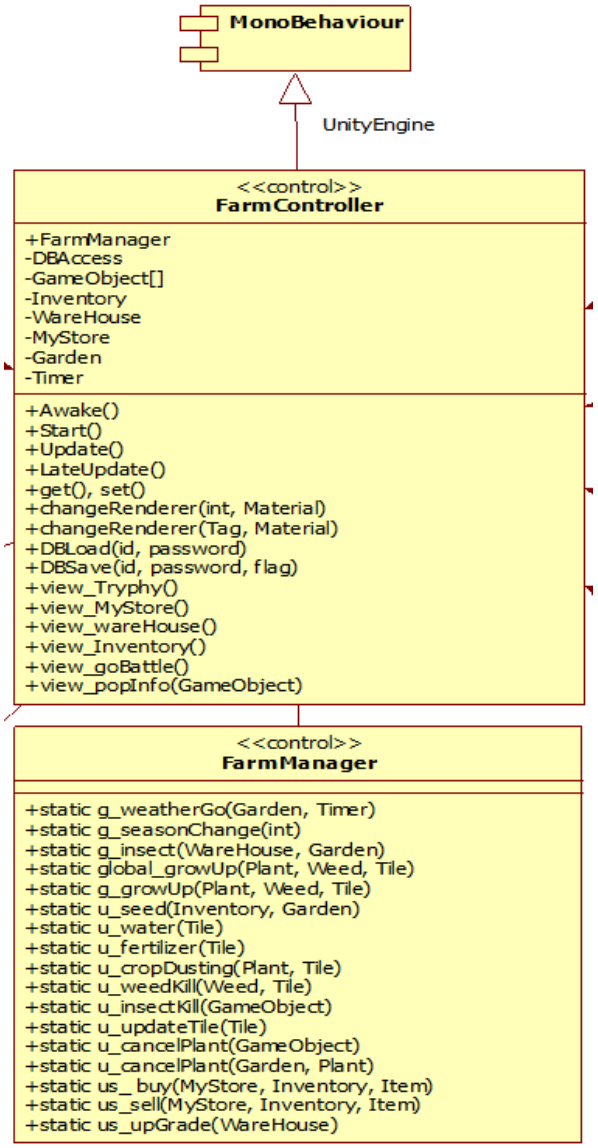
농사 관련 아이템으로는 비료, 농약, 주사기, 씨앗 등이 있다. 농장에서 기른 수확물은 WareHouse에 저장이 된다. 유저가 가급적 수확물을 사용하도록 유도하기 위해, 창고에 보관하는 수확물이 많을수록 농사를 짓는데 해충이 더 많이 발생하도록 하였다. 수확물은 타워 업그레이드나 마켓에서 판매할 수 있다.



농장을 이루기 위해 위의 5가지 클래스를 구현하였다. Garden은 유저의 농사 지역을 말한다. Tile은 농사를 지을 수 있는 토질의 최소 단위이며 Tile 위에 Plant를 심을 수 있다. 또한 Tile 에서는 해충이나 잡초 등이 생겨날 수 있어 유저는 이에 적절히 대응해야 한다.

농사 지역의 기후, 계절, 날씨 등을 구현하여 이러한 환경에 의해 농사에 많은 영향을 끼친다. 토양(Tile)은 Garden의 속성에 실시간 영향을 받아 수분이 증발하거나 온도가 변화한다. 따라서 토양의 속성은 식물의 성장 요소에 영향을 준다. 계절에 따라 서로 다른 해충과 잡초가 생겨나며, 겨울에는 나타나지 않는다. 해충이나 잡초를 제거하지 않으면 식물의 체력이 줄어들며, 건강한 상태가 지속되면 다시 체력을 회복하기도 한다. 농약을 사용하면 한동안 해충, 잡초에 대해 안전하지만 최종 수확량이 상대적으로 감소한다. 수확은 심은 품종에 따라 다양하며 3레벨 단계로 성장하면 수확할 수 있다. 실제 농사 자료를 토대로 구현하였기 때문에 실제 작물의 수확 철에 3레벨을 달성하도록 경험치 밸런스를 조절하였다.

1.2. Control Class



FarmManager는 농장에서 일어나는 모든 상호작용에 대한 로직을 정의하고 있다. 시간이 흐를 때 발생하는 시뮬레이션 함수와 유저의 상호작용 함수들을 정의하였다.

Unity3D의 C# Script로 구현한 FarmController 클래스는 Java에서 구현한 모든 모듈들을 사용한다. 농장 Scene이 로드될 때 함께 생성된다.

2. 결과

개발한 농장 파트와 전투(디펜스 게임) 파트를 통합하고 Unity Project 에서 데이터베이스 서버와 연동하였다.

농사 Scene 에서는 많은 환경요소 하에 작물을 기르고 수확할 수 있다. 작물을 기르기 위해서는 실제 농사에 대한 정보를 알아야 하며 도감에서 확인할 수 있다.

수확시기가 되면 수확물을 얻을 수 있다. 잘 키운 만큼 수확량에 대한 보상이 증가한다. 이렇게 농장 Scene 에서 수확한 작물은 판매하거나 전투에서 업그레이드에 사용할 수 있다.

디펜스가 성공하면 난이도에 따라 골드나 씨앗을 획득한다.

따라서 농사를 통해 더 높은 난이도의 전투가 가능하며, 그 반대로 전투를 할수록 농사를 짓는데 도움이 되기도 한다. 이로써 Friendly Farm 을 즐기며 자연스럽게 농사 지식을 함양하고 재미를 느낄 수 있다.

3-3. Smart Voice

1. SmartVoice 개발

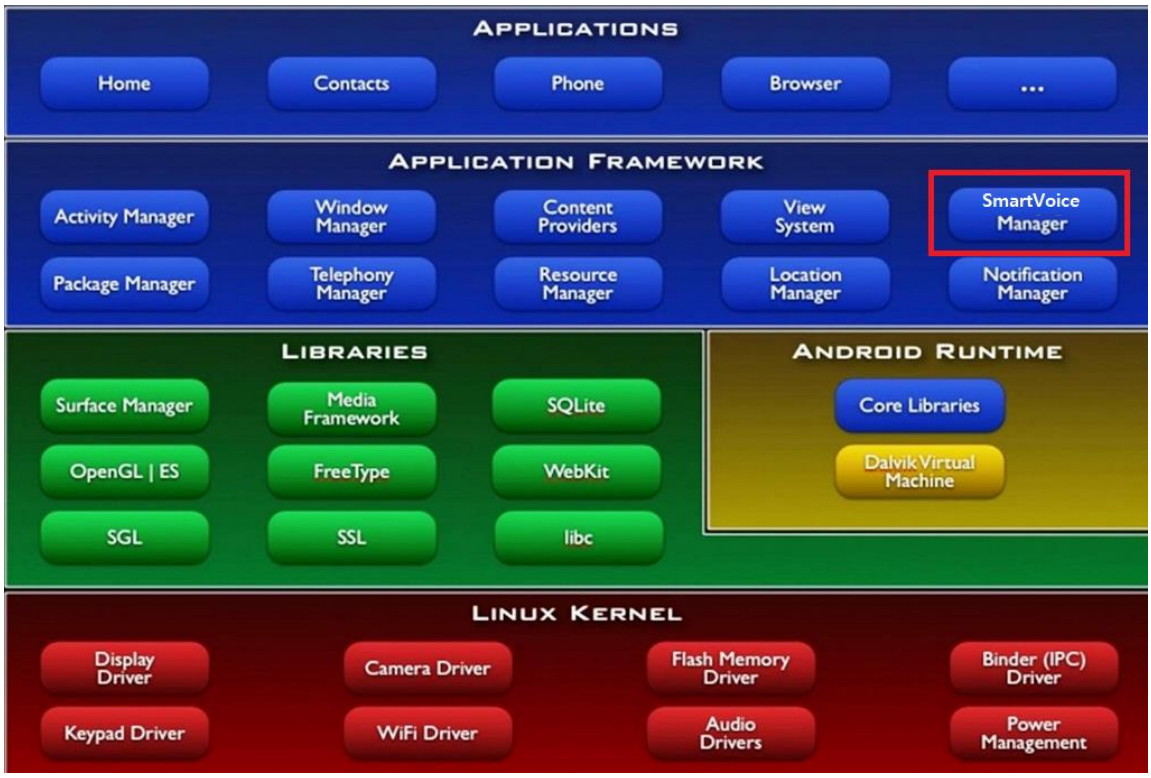
1.1. Concept

System에서 음성 탐지를 통해 결과 데이터를 전달 받으면, 해당 데이터를 기반으로 자연어 처리를 한다. 이 후 해당 자연어에 대한 처리 결과에 대응하는 가장 적절한 서비스를 찾아 수행하는 것이다.

시스템의 최종 사용자 관점에서는 크게 세 가지 과정을 거친다.

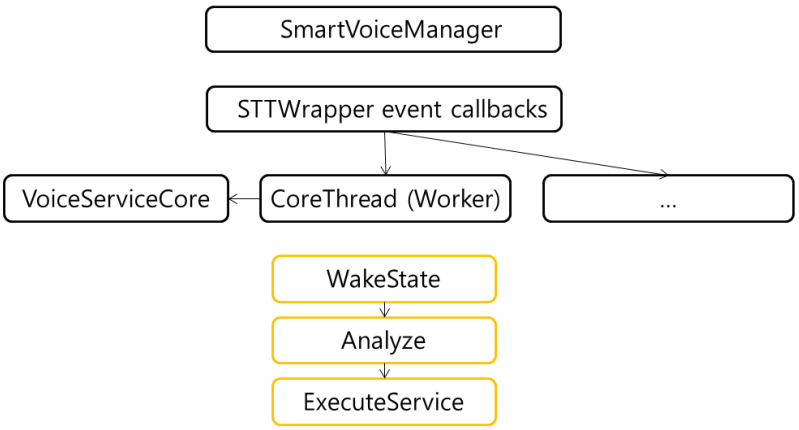
가장 먼저 자연어 처리 결과 시스템을 호출했다면 Sleep 상태에서 Wake(Hear) 상태로 전환한다. 이 때부터는 일시적으로 Wake up 상태가 된다. Wake up 상태 동안에는 사용자의 음성 수신 시 자연어 분석 후 가장 적절한 서비스를 찾아 수행하는 과정이 동작한다.

1.2. 계층도(Android Jellyboys SW Hierarchy)



Application Framework 계층에 개발한 시스템인 SmartVoiceManager를 추가하였다. 따라서 Application 개발자는 getSystemService()를 통해 SmartVoiceManager를 호출하여 음성 시스템에 접근할 수 있다.

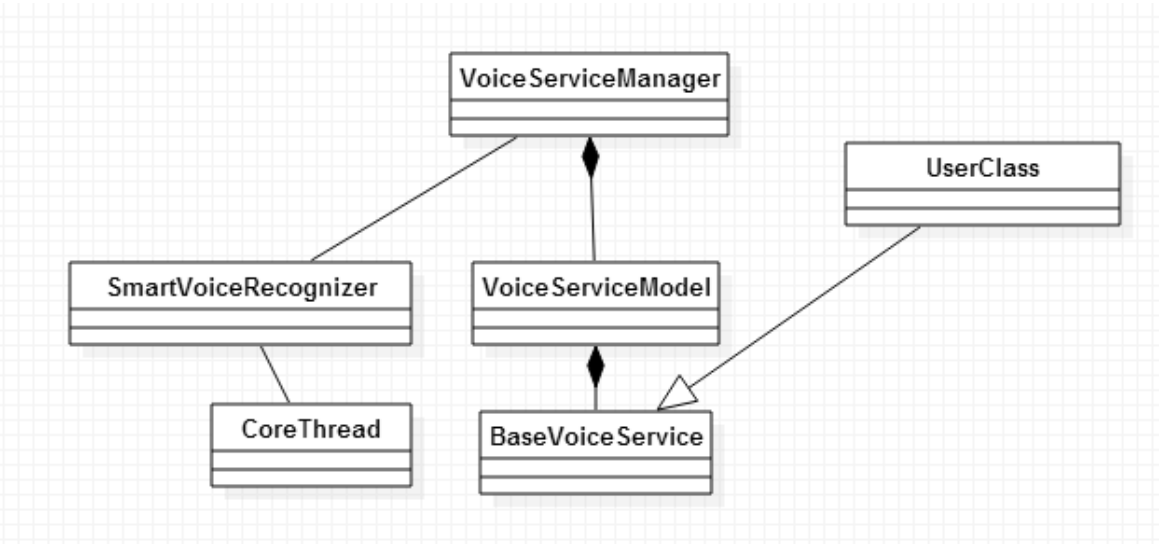
1.3. Implement



앞서 설명한 Concept와 동일하게, 시스템은 WakeState, Analyze, ExecuteService의 세 단계로 나뉘어진다. 최종 사용자가 시스템을 음성으로써 호출하면 CoreThread에서 음성을 분석하고 시스템을 활성화(Wake)한다. 이 후 자연어를 분석(Analyze)한다. 만약 사상된 서비스가 존재하면 Worker Thread를 생성하여 적합한 서비스를 실행(ExecuteService)하는 것이다.

1.4. Architecture & Description

1.4.1 Scheme

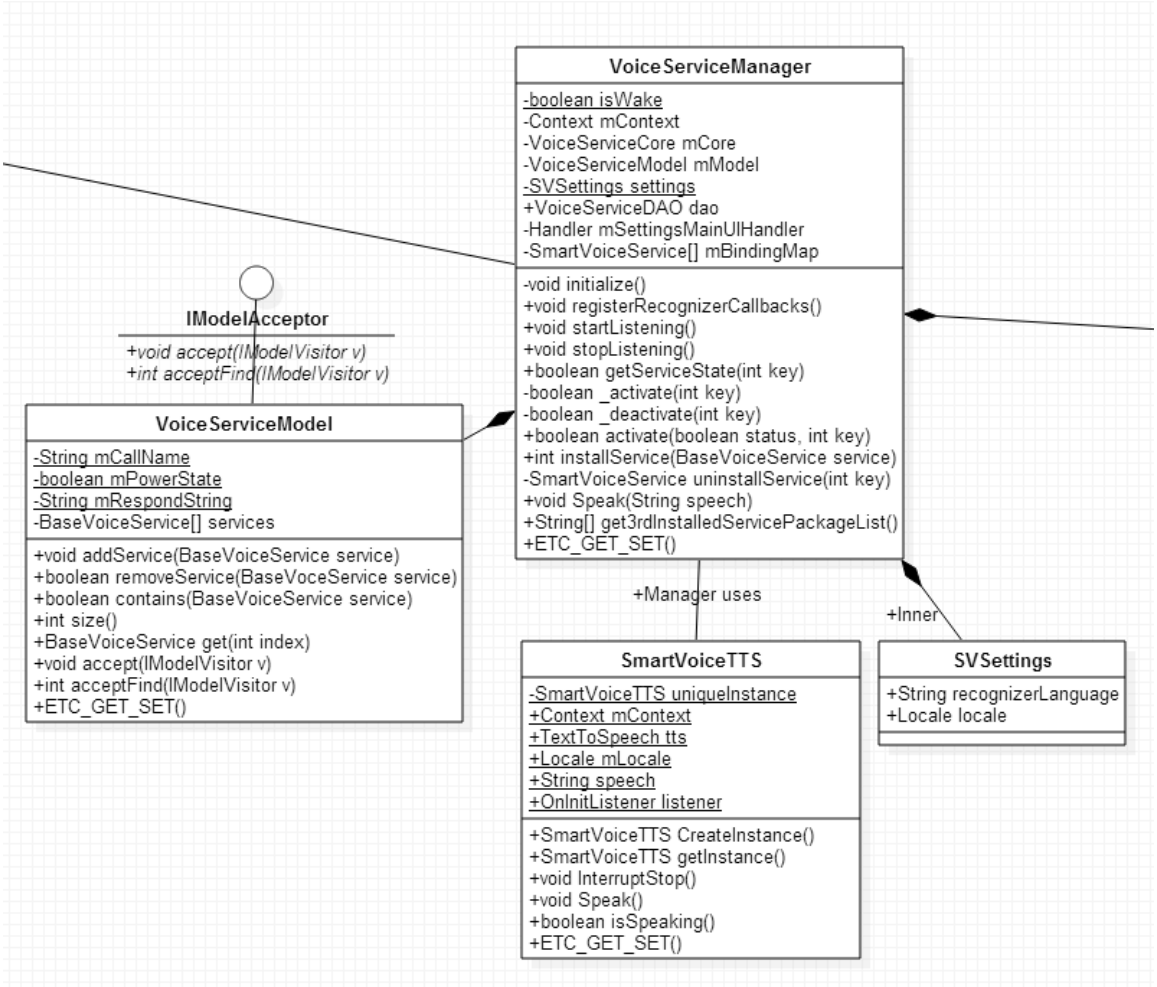


VoiceServiceManager(이하 매니저)는 음성 시스템을 총괄하는 클래스이다.

BaseVoiceService는 음성 서비스의 추상클래스로, 사용자가 구현한 음성 서비스는 매니저를 통해 VoiceServiceModel에 등록된다.

안드로이드 운영체제 부트 시 SmartVoiceManager는 시스템에 설치된 BaseVoiceService들을 로드하고, SmartVoiceRecognizer 서비스 쓰레드를 활성화한다. SmartVoiceRecognizer는 실시간 사용자의 음성을 탐지하여 자연어로 변환(STT)한다. 이 후 worker thread인 CoreThread가 생성하여 Wake up 상태이면 음성과 사상된 서비스가 있는지 확인한다. 서비스가 탐지된 경우, 해당 서비스를 실행하는 것이다.

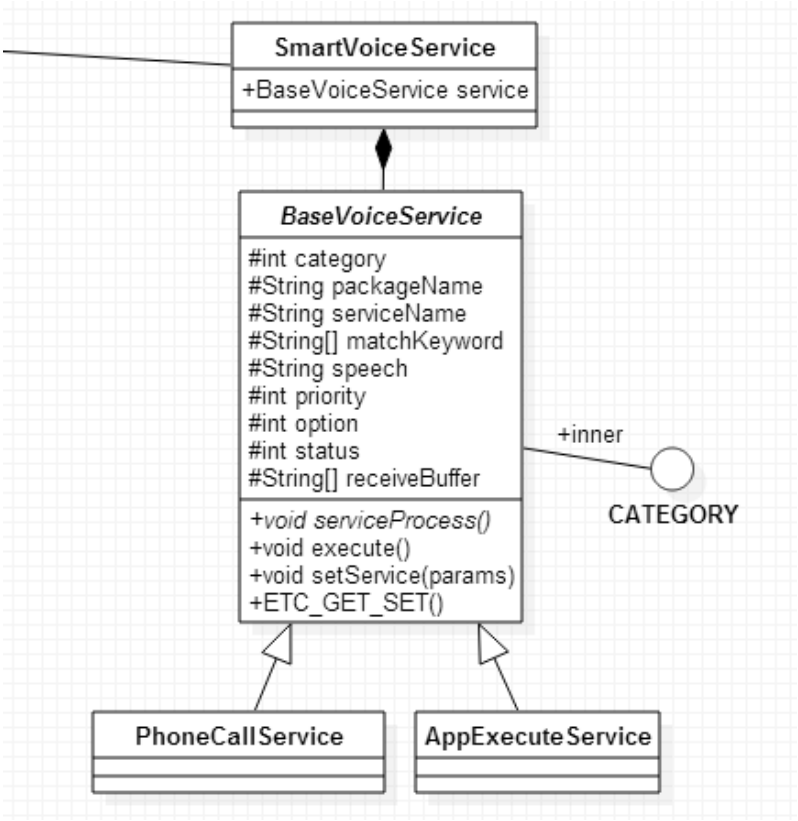
1.4.2 Manager & Model



SmartVoiceManager(이하 매니저) 생성 시 먼저 사용자 환경설정을 DB에서 로드한다. 그리고 TTS엔진의 Wrapper Class인 SmartVoiceTTS를 생성하고, 서비스 객체들을 저장하고 관리하는 VoiceServiceModel을 로드한다. 이 후 startListening() 함수를 호출하면 비로소 시스템이 작동한다.

VoiceServiceModel은 시스템의 자료구조들을 정의하는 클래스(Entity class)이다. 최종 사용자가 시스템을 호출을 하기 위한 mCallName과 응답 자연어인 mRespondString과 시스템의 활성화 상태정보인 mPowerState 등이 있다. 그리고 실제 음성 서비스 객체(BaseVoiceService)들을 이곳에 저장한다. 사용자의 음성 데이터(자연어)와 사상되는 서비스가 존재하는지 확인하기 위해 방문자 디자인 패턴을 적용하였다.

1.4.3 BaseVoiceService



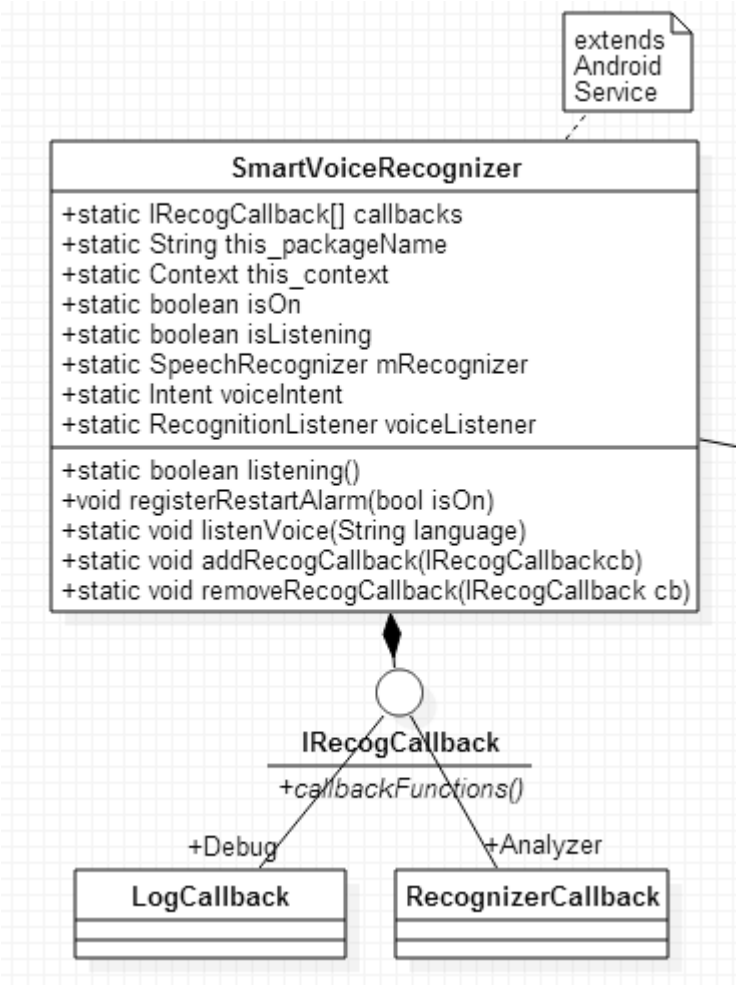
BaseVoiceService는 음성 서비스의 추상클래스이다. 새로운 음성 서비스를 구현하고자 하는 경우 BaseVoiceService를 상속 받아 구현한다.

Int category 서비스 종류	구현하는 서비스의 생활 카테고리를 설정한다.
String packageName 패키지 명	구현하는 애플리케이션의 패키지 명 프레임워크 계층 서비스인 경우 null
String serviceName 서비스 명	개발하는 서비스의 이름
String[] matchKeyword 호출 키워드	서비스 호출의 조건이 되는 자연어를 입력한다. 방문자 패턴을 통해 해당 키워드를 매칭하게 된다.
Int priority 우선순위	구현 서비스의 우선순위를 결정한다. 가장 높은 우선순위인 서비스를 호출한다.
Int option 서비스옵션	서비스의 자연어 처리 정책에 대한 옵션을 설정한다.
Int status 서비스상태	사용자의 서비스 사용여부를 설정한다.

CoreThread에 의해 자연어와 matchKeyword를 비교한다. 이 후 해당 서비스의 serviceProcess()를 호출한다. 따라서 개발자는 serviceProcess() 추상 함수를 구현하여 준다.

현재 음성을 통하여 통화(PhoneCallService) 및 애플리케이션을 실행(AppExecuteService)하는 2개의 기본 서비스가 구현되어 있다.

1.3.4. SmartVoiceRecognizer

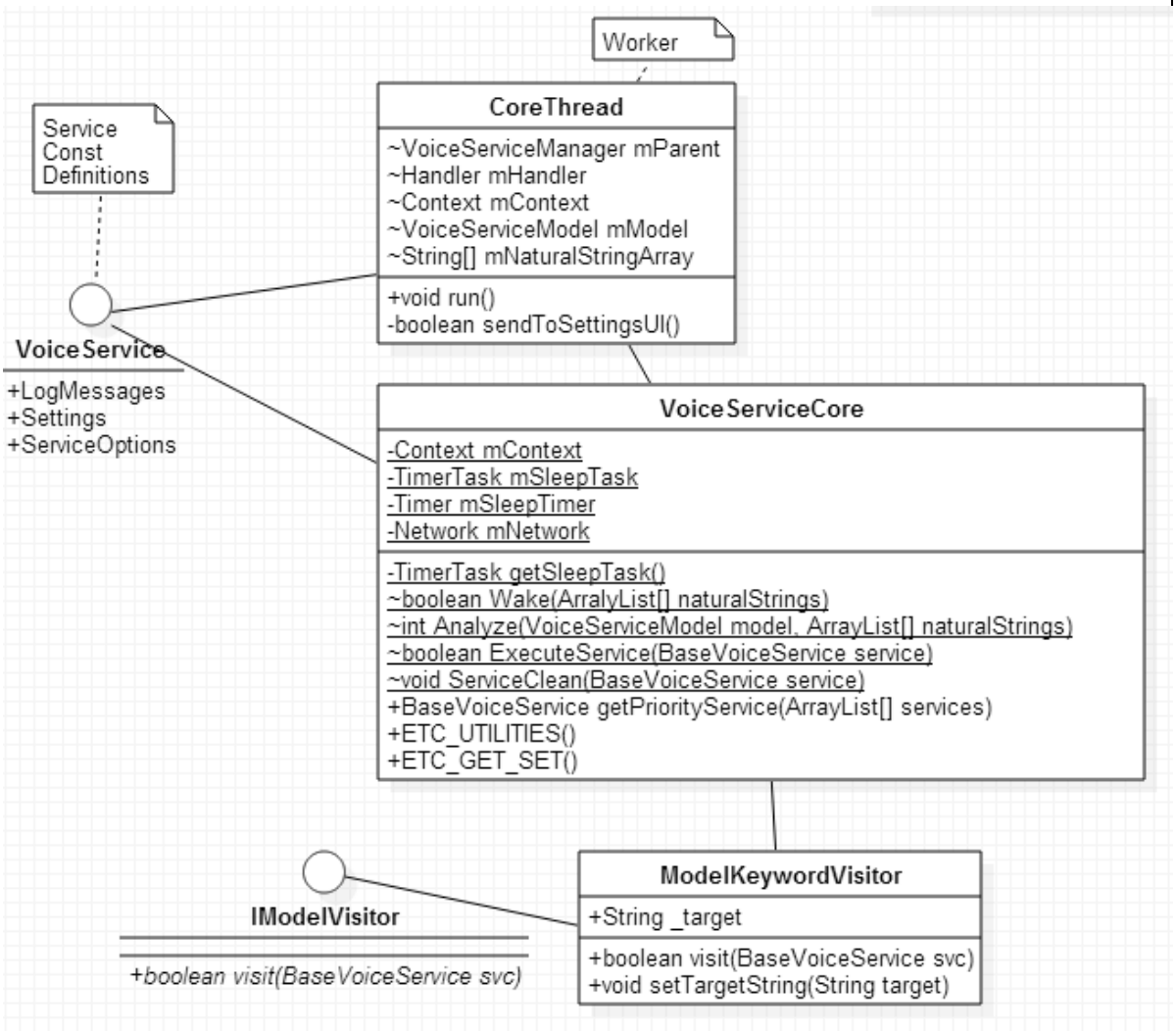


안드로이드의 service thread를 통해 사용자의 음성을 수신하는 모듈이다.

안드로이드 운영체제 부트 시 구현 시스템에서 SmartVoiceRecognizer를 활성화시킨다. 이후 SpeechRecognizer를 활성화시키고 콜백 함수들을 등록한다. 그러면 SpeechRecognizer가 음성을 수신할 때마다 등록되어 있는 모든 콜백 함수들을 호출한다.

RecognizerCallback 클래스는 매니저의 inner class로서 서비스들의 matchString과 자연어를 비교하여 serviceProcess()를 호출하는 클래스이다. LogCallback은 디버깅에 사용한다.

1.3.5. CoreThread



CoreThread는 일종의 Worker thread로써 자연어 분석을 통해 적합한 서비스를 호출하는 역할을 담당한다. SmartVoiceRecognizer는 사용자에게 자연어를 수신하면 CoreThread 객체를 만든다. 시스템 전체 수행단계인 Wake->Analyze->ExecuteService가 VoiceServiceCore에서 각각 함수로 구현되어 있는데, thread는 run()함수에서 해당 로직들을 절차적으로 호출한다. mModel의 멤버변수로 서비스 객체들을 참조하고, SmartVoiceRecognizer로부터 자연어 음성을 mNaturalStringArray에 수신 받아 이러한 로직들을 수행한다.

2. SmartVoice System Reference

2.1. 자연어 처리 정책

3rd Party 애플리케이션 개발자나 프레임워크 개발자가 그들 스스로 자연어 처리에 대한 정책을 선택하여 구현할 수 있도록 시스템에서 배려하였다.

정책을 구현하는 방법은 크게 세 가지로 나뉜다. 가장 먼저 개발자가 직접 자연어를 처리하기 원하는 경우(LOCAL_PROCESS), 그리고 개발자가 형태소 분석기 서버를 거쳐 가공된 데이터를 활용하는 경우(SERVER_PROCESS), 마지막으로 개발자가 외부 특정 서버와 직접 연동하여, 서버에서 미리 준비된 Proxy 서비스를 이용하는 경우(SERVER_PROCESS | GET_PROXY_DATA)이다. 따라서 개발자는 자신이 원하는 옵션에 따라 자연어를 자유롭게 처리할 수 있다.

만약 프레임워크가 자사에서 개발되었으며 자연어 처리를 담당하는 특정 서버를 연동한 경우, 실제 사용자들이 어떤 기능을 사물로부터 요구하는지 Data mining을 통해 가능하게 될 것이다. 이에 고객에게 즉각 대응할 수 있는 음성 서비스의 유지보수와 확장에 매우 유리할 것이다.

2.2. 음성 서비스 구현 방법

자연어 처리 결과 자신이 수행하고자 하는 서비스를 개발자 스스로 자유롭게 구현할 수 있다. 3rd Party/Framework 개발자는, 프레임워크에서 제공하는 "android.app.BaseVoiceService"를 import하고 상속함으로써 서비스 확장을 한다. 마치 android.app.Activity와 동일하다고 생각할 수 있다. 음성 서비스를 구현하는 방법은 크게 세 가지의 과정으로 나눌 수 있다.

1) serviceProcess()

개발자는 자신이 음성 프레임워크를 통해 서비스 하고자 하는 알고리즘을 Abstract 함수인 serviceProcess()를 구현하는 것으로 사용자에게 서비스를 제공할 수 있다.

2) setService(params)

구현한 서비스의 호출 조건과 옵션은 setService(params) 함수를 통해 설정할 수 있다. 함수에 설정하는 인자는 다음과 같다.

패키지 명	구현하는 애플리케이션의 패키지 명을 입력한다. 프레임워크 계층 서비스인 경우 null을 입력한다.
서비스 종류	서비스의 생활 카테고리를 설정한다.
서비스 명	개발하는 서비스의 이름을 설정한다.
호출 키워드	서비스 호출의 조건이 되는 자연어를 입력한다.
우선순위	서비스 호출 조건에, 기본 우선 순위를 결정한다.
서비스옵션	서비스의 자연어 처리 정책에 대한 옵션을 설정한다.
서비스상태	사용자의 서비스 사용 여부를 설정한다.

3) InstallService(service)

개발자는 getSystemService()를 통해 음성 프레임워크의 매니저인 VoiceServiceManager에게 자신이 개발한 서비스 객체를 설치하도록 SmartVoiceManager의 InstallService()함수에 서비스를 전달하면 모든 과정이 종료된다. 그러면 음성 프레임워크는 내부적으로 자연어를 수신하고 개발자가 신규 추가한 서비스를 호출하기 위해서 최소한의 자연어 처리를 수행 후, 등록된 서비스 중 적합한 서비스의 serviceProcess()를 호출하게 된다.

3. SmartVoice UI & System Application

음성 시스템의 현재 상태를 실시간 확인할 수 있도록, Android 부트 시 최상위 뷰에서 UI Symbol을 확인할 수 있다. 또한 음성 시스템에 대한 환경 설정을 담당하는 시스템 애플리케이션을 개발하였다. 사용자는 음성 시스템에게 이름을 붙여주어 호명할 수 있다. 원하지 않는 서비스는 환경 설정에서 비활성화할 수 있다.

4.1 System UI

Android의 System UI는 사용자에게 기기를 통해 정보를 항상 확인할 수 있도록 구현되어 있다. System UI에 종속된 대표적인 예로 Notification, QuickPanel 등이 있으며, 사용자의 기기 제어에 대한 상호작용을 담당한다.

이에 사용자와 음성 제어 시스템의 상호작용을 위해, 위 그림처럼 QuickPanel에 SmartVoice를 새롭게 추가하였다. 또한 사용자에게 음성에 대한 상호작용을 실시간으로 알릴 수 있도록 SmartVoice를 위한 System UI에 종속되는 Overlap UI 개념을 도입하였다.

<Sleep, Wake, Listening, Analyzing symbols>

4. 결과

해당 프로젝트를 진행하면서 구글 사의 안드로이드 운영체제 소스코드를 보며 디자인 패턴을 직접 배울 수 있었고, 강석민 강사님에게 이수했던 디자인 패턴(Singleton, Visitor, Observer 패턴 등)을 실전에 적용할 수 있어 아주 즐겁게 개발하였다.

개발한 음성 시스템을 탑재한 Android 4.3 Jellyboys 버전은 Galaxy Nexus 를 통해 Porting & Test 하였다. 샘플 음성 서비스인 앱 실행, 전화 걸기를 테스트하였고 추가로 심심이 API 를 활용하여 음성 서비스를 추가하여 테스트를 진행하였다. 만약 개발한 Android Framework 가 다양한 장치에서 Porting 을 하여 사용한다면, 장치 고유의 기능들을 사용자의 음성을 통해 수행할 수 있는 기기가 될 것이다.