베가프라임 엔진상에서 다중입력장치 호환을 위한 랩퍼 클래스 개발

김광태 [1], 신현실 [2], 박현우 [3] 이동훈 [4], 윤태수 [5]
동서대학교 인터넷공학부 멀티미디어 공학과 [1],
동서대학교 소프트웨어 대학원 유비쿼터스 네트워크공학과 [2]
동서대학교 지역기술혁신센터 [3]
동서대학교 디지털콘텐츠학부 디지털 VR 전공 [45]
{n121352[1], spinwind75[2], schark[3]}@nate.com, {dhl[4], tsyun[5]}@dongseo.ac.kr

Development of wrapper class for compatibility of Multi Input Device in Vega Prime™ engine

Kwang Tae Kim[1], Hyun Shil Shin[2], Hyun Woo Park [3],
Dong Hoon Lee[4], Tae Soo Yun[5]
Dept. of Multimedia Engineering, Dongseo university.[1]
Dept. of Ubiquitous Network Engineering, Dongseo university.[2]
UCGA for TIC, Dongseo university[3]
Div. of Digital Contents, Dongseo university.[4 5]

요 약

VR 엔진은 일부 입력장치에 대해서만 제한적으로 지원하기 때문에, 개발자가 원하는 입력장치를 사용하지 못하는 경우가 있으며, 가격 또한 고가이기 때문에 특수한 입력장치를 사용하기 위해, 다른 VR 엔진이나 별도의 옵션을 구매하기에는 경제적인 부담이 많이 든다. 이러한 문제를 해결하기 위해 본 논문에서는 개발자가 사용하고자 하는 입력장치와 VR 엔진의 호환을 위한 랩퍼 클래스를 제안한다. 개발한 랩퍼 클래스는 VR 엔진에서 조이스틱을 제어할 수 있는 조이스틱 클래스와 USB 캠을 통하여 영상을 획득하기 위한 USB 캠 클래스이다. 조이스틱 클래스는 입력장치 클래스를 상속받은 후 DirectX 를 이용하여 입력장치를 셋업 하고, 입력장치의 데이터 값을 처리한 후 VR 엔진의 API 로 값을 넘겨주기 전에 후킹하여 조이스틱을 제어할 수 있다. USB 캠 클래스는 VFW(Video for Window)를 사용하여 캠의 영상을 획득하여 버퍼에 저장한 후 VR 엔진의 디스플레이 버퍼에 값을 넘겨서 캠의 영상을 VR 엔진에서 디스플레이 할 수 있다. 이러한 방법을 통해 조이스틱, USB 캠 같은 입력장치를 VR 엔진과 호환할 수 있으며, 다른 종류의 입력장치에 대하여서도 본 연구에서 개발한 랩퍼 클래스를 상속받아 사용할 수 있다. 본 논문에서 사용한 VR 엔진은 Vega Prime 엔진이며, Vega Prime 엔진의 API 에 개발한 랩퍼 클래스를 추가하여 드라이빙, 영상인식 시뮬레이터를 개발한 결과, 효과적이고 경제적으로 입력장치의 연동이 가능함을 확인할 수 있었다.

Keyword : VR 엔진, 입력장치, Vega Prime 엔진, 랩퍼 클래스

1. 서 론

컴퓨터 그래픽스 기술과 하드웨어 장치의 발달로 컴퓨터를 이용하여 가상의 환경을 실제와 같이 느끼고자 하는 가상현실에 대한 연구가 다방면으로 진행되어 왔다. 오늘날의 가상현실은 거의 모든 분야에서 사용되고 있다고 말해도 과언이 아닐 것이다. 과거에는 주로 군사 시뮬레이션 위주로 사용되었지만 현재는 시뮬레이션, 원격 통신, 교육, 산업, 국방, 쇼핑, 문화 등 거의 모든 분야에서 가상현실 기법을 응용하고 있다. 그러나 컴퓨터 안에서 가상환경을 만들고 실제와 같은 다양한 인터랙션이 가능하도록 하는 것은 매우 복잡하고 어려운 작업과정들을 거쳐야만 하고, 실시간 적인 요소들이 충분히 만족되어야 하기 때문에 기술적인 개발에 많은 어려움이 따른다.

이러한 문제점들을 개선하기 위해 VR 엔진들이

개발되어 상용화 되고 있다. VR 엔진의 종류는 Tachyon, RTRE, Quake, Unreal, Lithtech, Eon, Virtool 등 다양한 기능을 하는 엔진들이 있다 [1~3]. 이러한 엔진들 중에는 군사용 시뮬레이터나 항만 시뮬레이터를 쉽고, 편리하게 개발 할 수 있는 Vega Prime 이 있다.

Vega Prime 은 3 가지 요소로 구성되어 있다. 첫째로 간단한 환경이나 object 를 모델링할 수 있는 Multigen creator 툴을 제공한다. Multigen creator 는 Maya 나 3D-Max 에서 모델링된 데이터를 로드해서 사용할 수 있는 기능을 제공하고 있고, 실시간성을 위해서 정보량을 최적화하는 기능을 제공하고 있다. 둘째로 날씨나 기후등과 같은 환경적인 요인들을 간단한 세팅을 통해서 구현할 수 있도록 하는 LynX Prime 툴을 제공한다. 가상환경을 쉽게 구성하고 다양한 네비게이션 방법들을 통해 보다 인터랙티브한 가상현실을 구현할 수 있다. 마지막으로 Vega API 를 통해 확장성있는 프로그램 개발을 지원한다[4]. 그러나 Vega Prime 엔진의 사용은 환경제작이나 간단한 인터랙션 제작에는 용이하지만, 실감형 가상현실을 구현하기 위한 외부장치와의 연결이 용이하지 않다는 단점이 있다. 이를 충족시키기 위해서는 또 다른 고가의 툴을 구입해야만 한다.

따라서 본 연구에서는 실감형 가상현실을 효과적, 경제적으로 구현하기 위해 외부입력장치와 Vega Prime 엔진을 호환시키는 랩퍼 클래스를 개발하였다. 개발한 랩퍼 클래스는 Vega Prime 엔진에서 조이스틱을 제어하기 위해 DirecX 와 연동하여 데이터 값을 전달하는 조이스틱 클래스와 vfw 를 사용하여 USB 캠의 영상을 버퍼에 저장한 후 Vega Prime 엔진의 디스플레이 버퍼에 그 값을 넘겨서 캠의 영상을 Vega Prime 엔진에서 처리할 수 있는 USB 캠 클래스 이다. 이러한 랩퍼 클래스의 개발은 실감형 가상현실을 구현하는데 매우 중요한 요소들로, 효과적이고 경제적인 작업환경을 제공할 수 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 랩퍼 클래스 구현에 필요한 레퍼런스 클래스와 상속관계, 문제점 및 랩퍼 클래스 구현방법에 대해 설명한다. 3 장에서는 랩퍼 클래스를 이용하여 개발한 드라이빙 시뮬레이터와 영상인식 시뮬레이터를 소개한다. 4 장에서는 결론을 맺으며 향후 연구과제에 대해 소개한다.

2. 랩퍼 클래스 구현

본 논문에서는 Vega Prime 엔진에 외부입력 장치인 조이스틱과 USB 캠을 호환하기 위한 랩퍼 클래스를 제안하였다. 아래 그림 1 은 현재 Vega Prime 엔진의 System 구성이다.

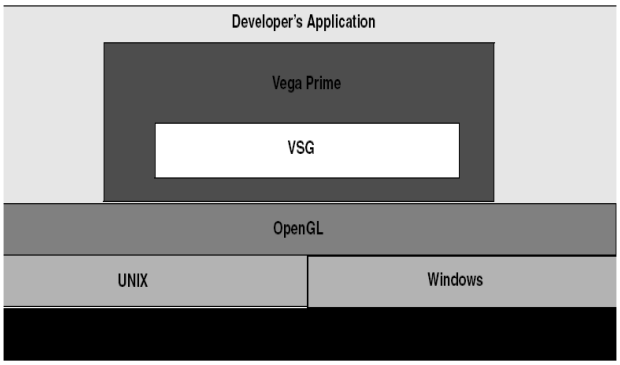

그림 1 Vega Prime System Component.

그림 1 에서 Vega Prime 은 확장성이 뛰어난 실행환경을 제공하기 때문에 시스템을 최대한 활용하여 최대의 생산성을 구현할 수 있다는 것을 확인할 수 있다.

2-1 랩퍼 클래스 : 조이스틱

Vega Prime 은 입력장치를 통하여 object 와 인터랙션할 수 있으며 이를 위해 레퍼런스 클래스를 지원하고 있다. 2-1-1, 2-1-2 에서는 조이스틱 클래스를 구현하는 과정에서 사용한 vpMotion, vpInput 두 클래스에 관하여 설명한다.

2-1-1 vpMotion 클래스

vpMotion 클래스는 LynX Prime 에서 만들어진 모션을 Vega API 에서 조작하기 위해 모션 객체를 레퍼런스하는 기본적인 클래스로써, object 들의 모션을 정의하며, 입력장치 각각의 값을 받아서 모션 객체에 적용시킨다. 그림 2 는 vpMotion 클래스의 base 및 chilld 클래스 다이어그램이다.

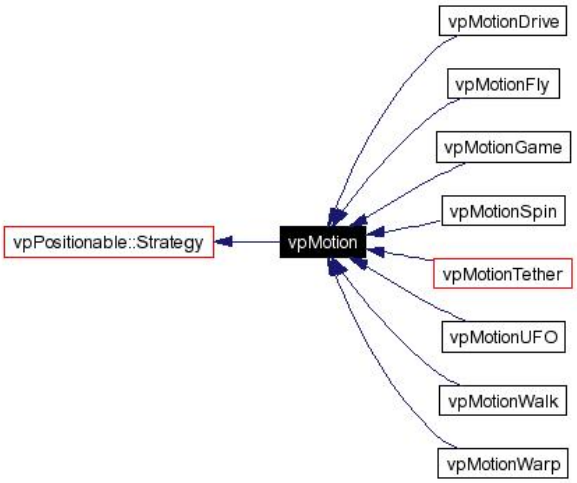

그림 2 Inheritance diagram for vpMotion.

vpMotion 클래스에는 자동차의 이동효과를 주기위한 모션 타입으로 vpMotionDrive 클래스가 있다. vpMotionDrive 클래스의 멤버변수에는 두 개의 열거형 데이터가 있는데, 첫 번째 멤버 변수는 enum SourceBoolean 으로 속도의 증가, 감소, 정지의 세가지 경우에 대한 정의가 되어있다. 두 번째 멤버 변수는 enum SourceFloat 으로 핸들의 x 축에 해당하는 헤딩 값을 다루는 값이 정의 되어있다[5].

2-1-2 vpInput 클래스

vpInput 클래스는 입력장치를 나타내는 기본적인 클래스로써, 입력 장치로부터 넘어오는 데이터를 저장하는 vector 를 가진다[6]. 그림 3 은 vpInput 클래스의 base 및 child 클래스 다이어그램이다.

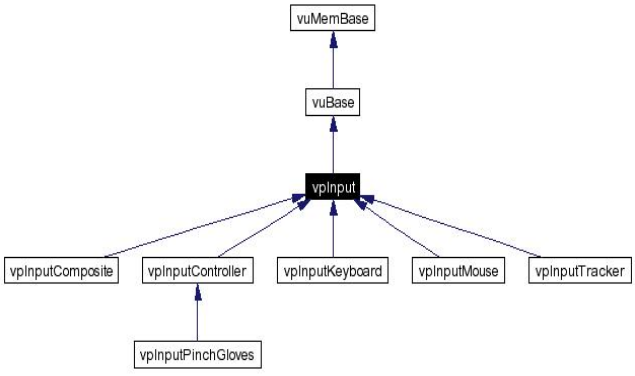

그림 3 Inheritance diagram for vpInput.

vpInput 이 지원하는 입력장치의 종류에는 키보드, 마우스, 트래커, 핀치글로브 등이 있으며, 2-1-3 에서는 Vega Prime 에서 지원하지 않는 조이스틱의 구현에 관해 설명한다.

2-1-3 조이스틱 클래스의 구현

본 절에서는 조이스틱 클래스의 구현과정에 대해 설명한다. vpInputJoystick 클래스는 개발한 조이스틱 클래스 이름으로써 vpInput 클래스를 상속받아 구현하였으며, 그림 4 는 개발에 사용된 Logitech Wheel Joystick 이다.

그림 4 Logiteck Wheel Joystick.

vpInputJoystick 클래스는 조이스틱을 입력장치로 사용하기 위해 vpInput 클래스로부터 유도된 임의의 클래스이다. 그러나 Vega Prime 의 클래스만으로는 조이스틱을 인식할 수가 없기 때문에 DirectX 를 사용하여 조이스틱 값을 vpInputJoystick 으로 가져와야 한다. DirectX 를 초기화하고, 조이스틱의 조작 데이터를 가져오기 위한 클래스로 vpInputJoystickDXImpl 을 만들고 vpInputJoystick 의 멤버 변수로 가진다. 그림 5 는 vpInputJoystick 클래스의 동작과정을 나타낸 다이어그램이다.

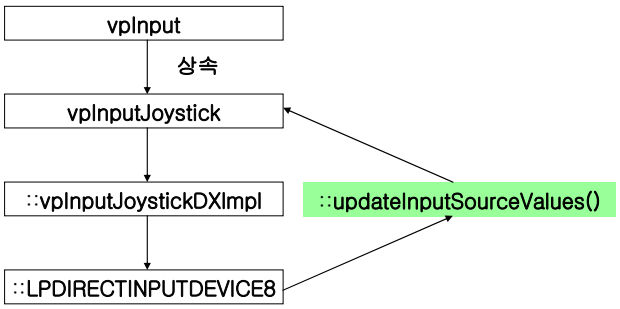

그림 5 vpInputJoystick 클래스 동작 과정.

그림 3 의 vpInputComposite 은 vpInput 으로부터 유도된 클래스이며, 입력장치로부터 가져온 데이터

를 vpMotion 과 연결을 위해 vpMotion::setInput-(vpInput *Input)함수의 인자로 넘겨진다. vpInputJoystick 의 값을 vpInputComposite 에 넘겨줄 때는 그림 6 과 같이 vpMotion 의 열거형 변수에 정의된 값의 순서대로 삽입해야 한다[7].

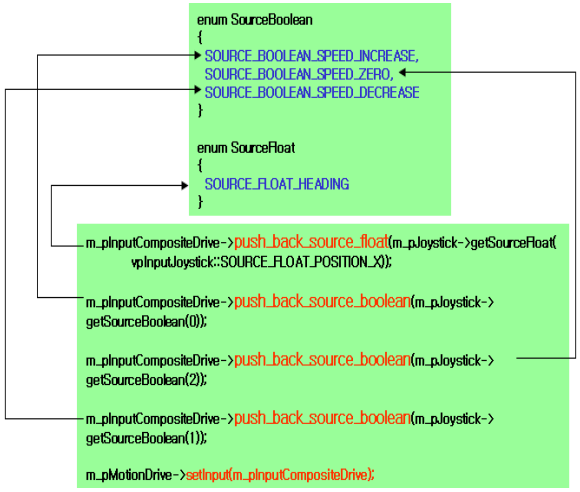

그림 6 vpInputComposite 에 값을 삽입하는 예.

페달이 없는 일반적인 조이스틱이라면 vpInput-Composite 을 통해 vpMotion 의 열거형 변수에 정의된 값의 순서로 삽입하면 그 값이 이상 없이 전달 될 것이나, 본 논문의 개발에 사용된 조이스틱은 그림 4 에서 보는 바와 같이 페달이 존재하며, 페달의 유격에 따른 float 형 Y 축 좌표 값을 가져온다. vpMotion 을 상속받은 vpMotionDrive 는 자동차의 속도를 버튼의 눌려짐과 눌려지지 않음을 판단하는 Boolean Vector 에서 처음 3 개의 값을 가져와 속도를 제어하는 식으로 되어있다. 그러므로 기존의 설계되어 있는 vpMotionDrive 을 그대로 사용하여서는 페달에 의한 속도조절이 불가능한 상태인 것이다.

이러한 문제를 해결하기 위해서 본 논문에서는 DirectX 를 이용하여 조이스틱의 값을 얻고, vpInp-utJoystick 으로 값을 넘겨주기 이전에 페달의 값을 수정하는 알고리즘을 삽입, 속도조절에 관련된 Bool 값을 임의로 변경하여 vpInputComposite 에 넘겨준다. 그림 7 은 문제해결 방법의 과정이다.

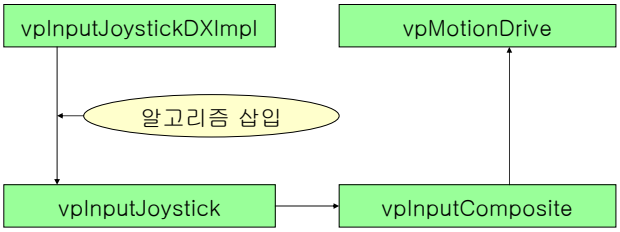

그림 7 문제해결 방법.

조이스틱 클래스의 구현 방법에 대해 요약하자면 다음과 같다.

1) LynX 에서 만들어진 모션타입을 레퍼런스 하기 위해 API 에서 vpMoiton 객체를 생성한다.

2) 입력장치로부터 데이터를 가져오기 위한 vpIn-putJoystick 을 생성하고 DirectX 를 이용하여 조이스틱의 데이터를 저장한다.

3) vpInputJoystick 으로 들어온 데이터를 vpMotion 에 삽입하기 위해 vpInputComposite 에 넣은 뒤 vp Motion 의 멤버함수를 이용하여 vpMotion 에 조이스틱의 데이터를 적용한다.

그림 8 은 조이스틱 클래스 전체 다이어그램이다.

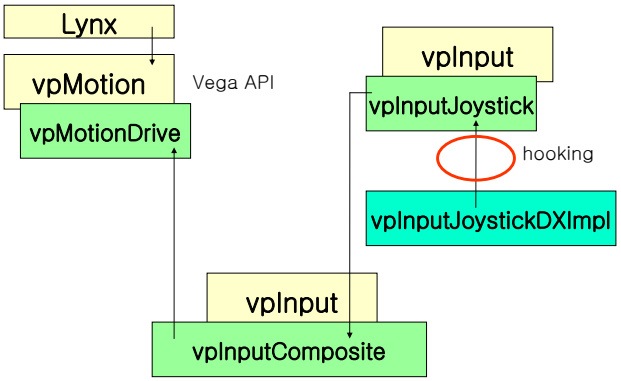

그림 8 조이스틱 클래스 전체 다이어그램.

2-2 랩퍼 클래스 : USB 캠

VFW(Video for Window)를 사용하여 Vega Prime 에서 지원하지 않는 외부입력 장치인 USB 캠에 관한 클래스를 개발 하였으며, 그림 9 는 개발에 사용된 신테크 Smile Cam 이다.

그림 9 신테크 Smile Cam.

2-2-1 vpOverlay2DImage 클래스

vpOverlay2DImage 클래스는 vpOverlay 클래스의 자식클래스로써, 화면에 디스플레이 되고 있는 프레임의 영역에 이미지를 로드하는 기능을 한다. 그림 10 은 vpOverlay2DImage 클래스의 상속관계를 나타낸다.

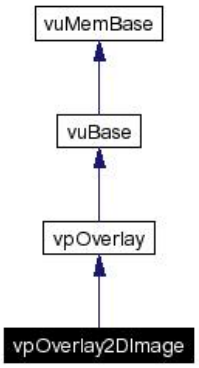

그림 10 Inheritance diagram for vpOverlay2DImage.

2-2-2 USB 캠 클래스의 구현

USB 캠 클래스는 VFW 를 사용하여 먼저 캠을 셋업하여 영상을 획득하여야 한다. 영상의 획득 및 출력하는 방법은 다음과 같다[9].

1) 캡쳐 윈도우 생성 : VFW 의 capCreateCaptureWindow()를 이용하여 생성한다.

2) 드라이버 연결 : VFW 의 capDriverConnect()를 이용하여 캡쳐 윈도우와 드라이버를 연결한다.

3) 캡쳐속도 설정 : VFW 의 capPreviewRate()를 이용하여 초당 캡쳐되는 프레임 수를 설정한다.

4) 프리뷰 : VFW 의 capPreview()를 이용하여 화면에 디스플레이 모드를 설정한다.

이러한 과정을 거쳐 캠으로부터 획득한 영상을 버퍼에 저장하는데, 획득한 영상을 Vega Prime 으로 가져와 화면에 디스플레이 하기 위해서 vpOverlay2DImage 클래스의 addImageFile(const char* filename)에 버퍼의 주소를 넘겨준다. addImageFile()는 화면에 디스플레이 할 이미지파일의 주소를 인자로 받는데, 여기서는 캠을 통하여 획득한 이미지를 저장한 버퍼의 주소를 넘겨받음으로써, 캠의 영상을 Vega Prime 에서 디스플레이 할 수 있다. 그림 11 은 USB 캠 클래스를 이용하여 테스트한 것으로 처음엔 아무것도 없는 빈 화면(LynX 에서 미리 세팅)에서 캠의 영상을 받아서 디스플레이 하는 것을 확인할 수 있다.

그림 11 USB 캠 클래스 결과 화면.

3. 랩퍼 클래스를 이용한 응용프로그램 개발

그림 12 는 조이스틱 클래스를 이용하여 개발한 드라이빙 시뮬레이터이다. 드라이빙 시뮬레이터는 조이스틱의 핸들(x)과 엑셀, 브레이크(y) 값을 받아서 구현하였다. 그림의 오른쪽 모니터는 자동차의 왼쪽과 오른쪽 밖의 모습과 미러를 구현함으로써 운전하는 듯한 느낌을 증폭시킨다.

그림 12 드라이빙 시뮬레이터.

그림 13 은 USB 캠 클래스를 이용하여 개발한 영상인식 시뮬레이터이다. 개발한 영상인식 시뮬레이터는 RGB 값이 빨강색에 가까운 값을 인식하여 파랑색 사각형을 그리게 구현하였다. 그림에서 알수 있듯이 흰색 벽면(a)에 빨간색 포인트가 나타나면(b) 그때부터 카메라가 빨간색을 찾아 다니면서 파랑색 사각형을 그리는 것(c,d)을 확인할 수 있다[10]. USB 캠 클래스는 VR 시뮬레이터에 컴퓨터 비젼의 영상인식분야를 접목시킴으로써 더욱 다양한 VR 응용 컨텐츠를 개발 할 수 있다.

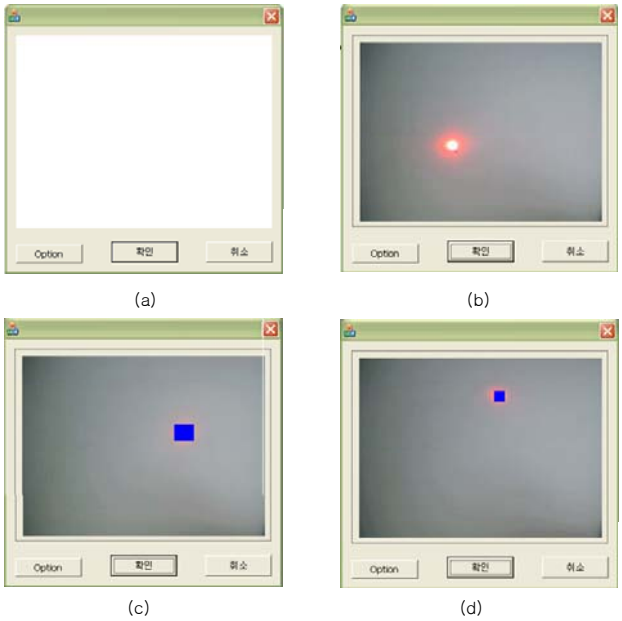

그림 13 영상인식 시뮬레이터.

4. 결 론

본 논문에서 제안한 것은 Vega Prime 엔진에서 특정 외부 입력장치인 조이스틱, USB 캠과 호환을 위한 랩퍼 클래스를 제안하였다. 개발한 랩퍼 클래스인 조이스틱과 USB 캠 클래스는 개발에 사용된 조이스틱 및 USB 캠이 아니더라도 Vega Prime 과 호환을 이루는데 문제가 없도록 입력장치의 제약사항들을 준수하여 표준에 맞게 개발하였다.그림 14 는 랩퍼 클래스를 추가한 전체 시스템 구성도 이다.

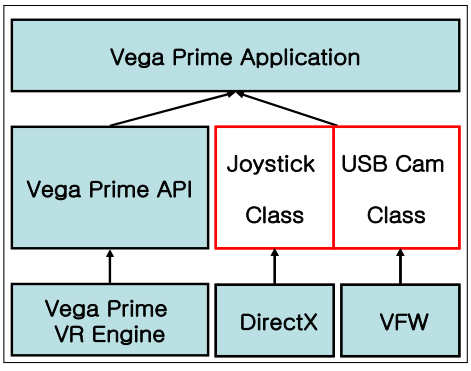

그림 14 랩퍼 클래스를 추가한 System Component.

개발한 랩퍼 클래스를 이용하여 드라이빙, 영상인식 시뮬레이터를 개발한 결과 Vega Prime 과 호환을 이루는데 문제가 없었으며, 성능 면에서도 효과적이었다.

현재 본 연구에서는 조이스틱 클래스에 차량의 동역학 부분을 추가하여 더욱 실감적인 드라이빙 시뮬레이터를 개발 할 것이며, 영상인식 시뮬레이터를 기반으로 하여 VR 엔진에 컴퓨터 비젼분야의 연구를 접목시켜 폭 넓은 응용 컨텐츠 개발을 할 것이다.

감사의 글

본 연구는 산업자원부의 지역혁신 인력양성사업의 연구결과로 수행되었음

참고문헌

[1] CGWAVE INC. , http://www.cgwave.co.kr.

[2] Mediatree, http://www.mediatr.co.kr.

[3] NETMEDIA, http://netmedia.kjist.ac.kr.

[4] KCEI, http://www.kcei.co.kr.

[5] Vega Prime Reference Guide, vpMotion class.

[6] Vega Prime Reference Guide, vpInput class.

[7] Vega Prime Reference Guide, vpInputComposite class.

[8] Vega Prime Reference Guide, vpOverlay2Dimage class.

[9] MSDN Library, Win32 and COM Development, Graphics and Multimedia, Windows Multimedia , Video for Windows.

[10] 영상처리 이론과 실제 홍릉과학출판사, RANDY CRANE 저. 최형일, 이근수, 이양원 공역.