PSP 게임기용 크로스플랫폼 2D 게임 라이브러리 개발

Development of Crossplatform 2D Game Library for PSP Game Console

이대현
한국산업기술대학교 게임공학과

Dae-Hyun Lee(dustinlee@kpu.ac.kr)

요약

본 연구에서는 대표적인 휴대형 게임기인 SONY PSP용 게임 개발을 위한 2D 게임 라이브러리를 구현했다. 이 라이브러리는 SDL이라는 공개 2D 라이브러리를 PSP 용에 맞게 이식하여 개발되었으며, PC 상의 작동과 동일하게 PSP 상에서도 실행되는 크로스플랫폼 라이브러리이다. 따라서 개발자들은 PC 상에서 게임 개발을 수행한 후, 본 게임 라이브러리를 이용하여 PSP용 게임을 손쉽게 개발할 수 있다.

■ 중심어 : | 휴대형 게임기 | PSP | 게임 엔진 |

Abstract

The portable game market is growing so fast recently. However, the development activities of domestic game development companies have been very week, which is mainly due to the lack of efficient and economical game development library solutions. This paper deals with how to implement 2D game library for SONY PSP game console that is very popular thesedays as a portable game console. We have ported the open source SDL library to PSP. Consequently, developers could develop games in PC environment and port it to PSP very easly based on the game library

■ keyword : | Portable Game Console | PSP | Game Engine |

I. 서 론

휴대형 게임기인 Sony PSP는 2004년 발매 이후, 전 세계적으로 큰 인기를 구가하고 있는 게임기로써, 비교적 콘솔 게임기의 인기가 적은 우리나라에서도 이미 30여만대가 보급되는 등 상당한 인기를 누리고 있다. 게임기와 게임소프트웨어의 판매가 비교적 호조를 보이고 있으나, PSP 하드웨어의 국내 출시 후 아직까지 순수하게 국내 개발사를 통해서 개발 출시된 게임은 겨우 다섯개 밖에 되지 않는 점에서 보여지듯이, 국내 게임 개발사들의 개발이 활발히 이루어지지 않고 있다. 이러한 배경의 주요한 원인 중의 하나는 PSP 게임 개발자들이 턱없이 부족한 것도 원인이고, 또 이를 좀더 들여다보면, PSP와 같은 게임 콘솔용 게임 개발의 난이도가 상당히 높은 것도 주요한 원인으로 꼽히고 있다.

PSP 게임을 개발하기 위해서는 일반적인 윈도우 환경이 아닌 리눅스로 개발환경을 구성하고, GCC라는 다소 생소할 수도 있는 컴파일러를 이용해야 하며, PSP 하드웨어 개발 장비를 통해 개발을 해야 한다. 한편, PSP의 하드웨어에 대한 이해, 그리고 새롭고 낯선 API

접수번호 : #070110-002
접수일자 : 2007년 01월 10일
심사완료일 : 2007년 04월 24일
교신저자 : 이대현, e-mail : dustinlee@kpu.ac.kr

들을 이해해야 하고, 참고서적도 굉장히 제한적이어서 PSP 게임 개발에 접근하는데 큰 어려움이 되고 있다.

한편, 소니사가 정책적으로 라이센싱을 제한하고 있기 때문에, 개발자들이 PSP 의 개발 환경 자체에 접근하는 것이 허용되고 있지 않는 것도 큰 원인이 되어왔다.

따라서 본 연구에서는 PSP 게임 개발에 좀더 많은 개발자들이 쉽게 접근할 수 있도록 크로스플랫폼 개발 환경을 구축했다.

본 연구의 목표는 PSP 게임의 개발에 필요한 모든 작업을 PC 상에서 수행할 수 있도록 하는, 100% 호환이 가능한 크로스플랫폼 환경을 구축하는 것이다[그림 1].

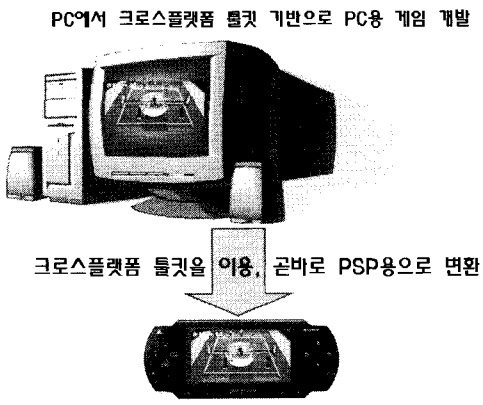

그림 1. 크로스플랫폼 게임 개발 프로세스

본 개발 환경 및 라이브러리를 이용하여 족구게임을 데모로 개발했으며, 이를 통해 본 개발 환경 및 라이브러리의 우수성을 확인할 수 있었다.

II. 기존의 크로스플랫폼 기술

최근 게임 개발비의 상승에 따라, 주요 게임 개발사들은 크로스플랫폼 기술을 이용하여 게임을 개발하는 정책을 적극적으로 채택하고 있다. PSP는 [표 1]과 같이 상용의 크로스 플랫폼 게임 엔진들이 제공되고 있다. 하지만, 이러한 엔진들은 가격도 고가이고, 또 PSP 게임 개발 라이센싱만을 가진 업체들만이 액세스가 가능하여, 일반 개발자들이 접근하는 것이 상당히 어려운 실정이다.

공개 소스로 되어 있어 무료로 사용할 수 있는 3D 엔진으로 OGRE, Crystal Space, Irrlicht 등이 있으나[1], 크로스플랫폼 기능이 미약하고, 기능적으로 PC용으로 제작된 것이어서, PSP용으로 사용하기는 힘들다.

PSP용으로 사용을 고려할 수 있는 라이브러리로써, SDL[2]과 Allegro[3]를 들 수 있다. SDL은 크로스 플랫폼 멀티미디어 개발용 API로서, 게임뿐만 아니라 게임 SDK, 에뮬레이터, MPEG 플레이어 그리고 다른 애플리케이션의 개발을 위해 사용된다. 윈도우 환경과 비교하면 게임이나 멀티미디어 애플리케이션 개발을 위한 API를 제공한다는 점에서 윈도의 DirectX와 유사한 라이브러리라고 생각할 수 있다. SDL의 가장 큰 장점은 여러종류의 플랫폼을 완벽히 지원한다는 점으로써, Win32, MacOS, 리눅스, BSD, WinCE, Symbian 등 PC 뿐만 아니라 모바일 플랫폼 역시 모두 지원하고 있다. 본 연구에서도 이와 같은 장점으로 인해, SDL을 PSP용으로 이식하기로 결정했다. 라이브러리의 소스가 모두 공개되어 있기 때문에, 이를 참조하여 비교적 쉽게 새로운 플랫폼으로 이식하는 것이 가능하기 때문이다.

표 1. PSP용 3D 게임 엔진

개발사	홈페이지	엔진 이름
Criterion Software	www.renderware.com	RenderWare
Vicious Cycle Software	www.viciousengine.com	Vicious Engine
Virtools SA	www.virtools.com	Virtools
Silicon Studio Corporation	www.alchemy.ne.jp	Silicon Studio

국내에서는 SDL을 이용한 개발이 매우 드물지만, 외국의 경우는 아마츄어 게임 개발자들을 중심으로 SDL을 이용한 게임 개발이 매우 활발히 이루어지고 있다[6][7].

알래그로 역시 공개 라이브러리로써, 비디오 게임과 멀티미디어 프로그래밍에 사용되는 것을 주목적으로 하는 포터를 라이브러리이다. DOS, UNIX, Windows, BeOS, MacOS 등과 같은 다양한 플랫폼 상에서 지원이 가능하고, 그래픽 함수, 각 플랫폼에 대해 그래픽 드라

이버, 사운드 함수, 사운드 드라이버, 수학 함수 등을 지원한다.

III. 통합 개발 환경의 구축

소니에서 PSP 게임 개발용으로 지원하는 개발환경인 PSP SDK는 크게 컴파일러와 디버거로 구성되어 있다[4]. 컴파일러는 GCC를 기반으로 개발된 것이어서, 리눅스 또는 PC의 cygwin에서 작동된다. PSP GCC 컴파일러는 코맨드 라인 방식으로 사용을 해야 하기 때문에, PC 윈도우 상에서 마이크로소프트 비주얼 C++와 같은 통합 개발 환경에서 프로그램 개발에 익숙해져 있는 일반 개발자들이 쉽게 접근하기가 어려운 것이 사실이다. 따라서, 이와 같은 점을 극복하고자 PSP 게임 개발 전용의 통합 개발 환경이 상용으로 제공되고 있으며, 그 대표적인 것이 MetroWorks사의 CodeWarrior이다. 하지만, 상당히 고가에 판매되고 있어, 국내의 중소 게임 개발사들이 이 툴을 이용하여 게임 개발에 활용하는 것이 어려운 실정이다.

따라서, 본 연구에서는 공개 소프트웨어로써 무료로 자유롭게 사용할 수 있으며, PSP 게임 개발에 효과적으로 활용할 수 있는 통합개발환경을 선택하기 위해, 인터넷에 공개된 여러 개의 통합 개발 환경을 조사했으며, 그 결과로 Code::Blocks 라는 통합 개발 환경이 가장 적합한 것으로 판단되었다[5][그림 2]

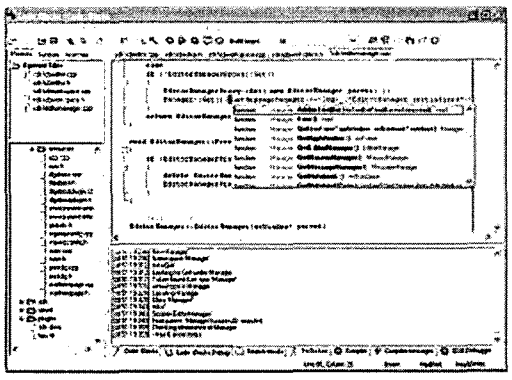

그림 2. 코드블럭 통합 개발 환경

이 통합 개발 환경은 비주얼 C++과 유사한 환경을 제공하여 기존 개발자들이 쉽게 접근할 수 있으며, SDL 라이브러리를 이용한 프로그램 개발 환경을 자체적으로 제공하기 때문에, PSP 게임 개발 통합 환경을 효율적으로 구축할 수 있다. 그리고, 크로스플랫폼 환경을 가정하고 만들어졌기 때문에, 여러 플랫폼에 맞게 다양한 컴파일러 설정을 쉽게 할 수 있다.

본 연구에서는 PC 윈도우 상에서 Cygwin을 설치하여, 리눅스 환경을 구축하고, 여기에 PSP SDK를 설치했다. 그리고, Colde::blocks를 설치한 후, 관련한 컴파일러 설정을 PSP에 맞도록 조정했다. 개발자가 이를 이용하여 PSP 게임을 개발하려면, SDL을 이용한 프로그램을 작성한 후, 먼저 Code::blocks 에서 컴파일러 타겟을 PC로 설정하여 컴파일 및 디버깅 과정을 통해서 PC 상에서 완벽하게 작동하는 프로그램을 개발한다. 이렇게 개발이 완료되면, Code::blocks의 컴파일러 타겟을 PSP로 설정한 후, 컴파일을 하여, 그 실행 화일을 PSP 하드웨어 개발 장비에 다운로드하여 실행함으로써 PSP 상에서 개발을 완료할 수 있게 된다.

IV. SDL 라이브러리의 이식

SDL은 다섯개의 라이브러리로 구성되며, 이식을 용이하게 하기 위해서 플랫폼 독립부와 플랫폼 종속부로 잘 구분이 되어있다. 본 절에서는, SDL 라이브러리의 구조와, 그 중 핵심적인 부분인 비디오 라이브러리 부분과 오디오 라이브러리 부의 이식에 대해서 상세히 기술한다.

1. SDL 라이브러리의 구성

SDL은 [그림 3]에서 나타낸 바와 같이 비디오, 오디오, 입력, 타이머, 및 쓰레드의 다섯 개의 라이브러리로 구성되어있다.

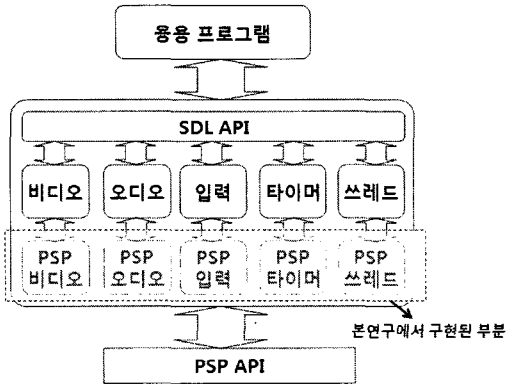

그림 3. SDL 라이브러리의 구조

비디오 라이브러리는 그래픽 프레임 버퍼 표면을 할당하고 액세스하는 함수들로 구성이 되어, 하드웨어의 그래픽 처리를 담당한다. JPG, TIFF, BMP 등, 여러 가지 형식의 이미지 파일들의 로딩을 지원하므로 편리하게 사용이 가능하다. 오디오 라이브러리는 웨이브파일, MIDI 파일, 및 MP3 파일을 읽어들여 재생해주는 라이브러리로써, 여러 개의 채널을 통해서 사운드를 쉽게 재생할 수 있다. 입력 라이브러리는, 키보드, 마우스, 및 조이스틱 등으로부터 사용자의 입력을 받아들이는 기능을 수행한다. 타이머 라이브러리는 하드웨어가 지원하는 시간 기능을 SDL을 통해서 액세스할 수 있게 해주며, 쓰레드 라이브러리는 멀티 쓰레딩을 지원하는 하드웨어의 사용을 가능하게끔 해준다.

이와 같은 다섯 개의 라이브러리는, 내부적으로 플랫폼 독립적인 부분과 종속적인 부분으로 논리적인 분할이 잘 되어 있어, 여러개의 플랫폼으로 쉽게 이식이 가능하다. 본 연구에서는 [그림 3]에서 나타내듯이, SDL API의 플랫폼 종속적인 부분을 PSP API와 연동시키는 부분을 구현하여, PSP 플랫폼에서 사용이 가능한 SDL 라이브러리를 구현했다. 이렇게 구현된 라이브러리 위에서 응용 프로그램이 작동하게 된다.

2. 비디오 라이브러리의 구현

SDL은 다양한 종류의 비디오 장치들을 지원하기 위해서 SDL_VideoDevice라는 구조체를 이용하여 비디오 디바이스를 상위 API에 대해서 추상화하고 있다. SDL_VideoDevice 구조체는 특정한 비디오 장치에 대한 정보와 함께, 비디오 장치에서 추상화해서 처리해야 할 함수들을 함수 포인터의 형태로 가지고 있다. 따라서, PSP 로 SDL의 비디오 라이브러리들을 포팅하려면 SDL_VideoDevice의 정보를 PSP에 맞게 설정하고, 구조체안의 함수 포인터들이 가리키는 실제 함수들을 PSP 에 맞게 작성해주면 된다. SDL_VideoDevice 구조체는 모두 23개의 함수 포인터를 담고 있다. 본 연구에서는 23개의 비디오 처리 함수를 모두 구현했으며, [표 2]에 주요한 함수들을 설명한다.

이 함수들 가운데 가장 핵심적인 함수들은 비디오 모드를 설정하는 SetVideoMode() 함수와 하드웨어 가속 기능을 이용하여 SDL 표면 위로 이미지를 고속 복사하는 CheckHWBlit() 함수이다.

표 2. 구현된 주요 비디오 라이브러리 함수들

함수	기능
VideoInit	하드웨어 비디오 시스템의 초기화
ListModes	지원되는 비디오 해상도 정보 제공
SetVideoMode	비디오 모드 설정(해상도, 색상깊이)
SetColors	팔레트 색상 설정
UpdateRects	화면을 지정된 부분을 갱신
AllocHWSurface	비디오 메모리 상에서 표면을 할당
CheckHWBlit	하드웨어적인 블리팅이 가능한지 판단
FillHWRect	화면을 지정된 부분을 칠함
FlipHWSurface	하드웨어 플리핑을 수행함
LockHWSurface	쓰기동작이 가능하도록 표면을 설정
UnlockHWSurface	읽기동작이 가능하도록 표면을 설정

2.1 PSP_SetVideoMode() 함수의 구현

PSP_SetVideoMode() 함수는 SDL 라이브러리의 SDL_SetVideoMode() 함수를 구현한 것으로써, 화면의 크기, 색상의 깊이 등을 설정한다. PSP는 화면 싸이즈가 480x272 로 항상 고정되어 있으므로, 스크린 표면의 플래그를 SDL_FULLSCREEN으로 설정하여, 풀스크린 모드로 항상 동작하게 한다. 색상의 깊이는 PSP에서 8비트, 16비트, 32비트 색상을 모두 사용할 수 있으나,

일반적으로 16비트로 사용하는 것이 PSP에서 최적의 성능을 보이는 것으로 알려져 있다.

한편, PSP_SetVideoMode() 에서 중요하게 처리할 일은, 하드웨어 표면을 생성하도록 SDL_SetVideoMode() 함수가 호출되었을 때, 이에 따른 PSP의 비디오 메모리를 할당하는 것이다. 그리고, 더블 버퍼를 사용하도록 플래그가 설정되어 있다면, 이에 따라서 PSP의 비디오 메모리를 추가로 할당하여 연결시켜주게 된다.

2.2 PSP_CheckHWBlit() 함수의 구현

이 함수는 하드웨어적으로 이미지 블리팅을 가속할 수 있는지 여부를 확인하고, 만일 블리팅 가속이 지원된다면, 하드웨어 블리팅 함수를 호출하는 역할을 한다. 본 연구에서는 HWAccelBlit()라는 PSP 전용의 하드웨어 블리팅 함수를 구현했다. 목적 표면이 현재 표시되고 있는 화면이고 또 동시에 하드웨어 표면이라면, PSP의 3D 가속기의 드로잉 함수를 통해서 이미지의 블리팅을 수행한다. 하드웨어 표면인 경우는 PSP의 비디오 메모리에 할당이 되어 있기 때문에, 3D 가속기를 통한 고속 블리팅이 가능하기 때문이다. 이 경우, 일반적인 이미지 블리팅과 같이 메모리 복사를 수행하지 않고, 3D 공간에서 여러 개의 사각형을 만들고, 각각의 사각형들에 텍스쳐를 매핑하는 방법을 취함으로써, 3D 가속기를 이용하는 방법을 사용했다. 한편, 목적 표면이 소프트웨어 표면이라면, 일반 메인 메모리에 표면이 할당되어 있는 상태이기 때문에, sceGuCopyImage() 함수를 통한 일반적인 메모리 복사를 수행한다.

3. 오디오 라이브러리의 구현

SDL의 오디오 라이브러리는 비디오 라이브러리와 유사하게, 여러 종류의 오디오 장치를 지원하기 위해 SDL_AudioDevice라는 구조체를 통해 플랫폼의 추상화를 구현한다. 이 구조체는 모두 7개의 함수를 포인터의 형태로 지니고 있으며, PSP에 맞도록 각각의 함수를 구현해주면 된다. 이중 가장 중요한 함수는 PlayAudio() 함수로써, 웨이브 파일 형태의 사운드 파일을 플레이해주는 부분이다. PSP가 지원하는 사운드 포맷은 16비트 선형 PCM 웨이브(44.1KHz 샘플링)와, ATRAC3라고 불리는 소니사의 고유 사운드 포맷이다. SDL에서는 ATRAC3를 지원하지 않고 있기 때문에, PSP_PlayAudio() 함수는 오직 선형 PCM 웨이브 파일만 플레이하도록 구현되었다. SDL에서 웨이브 파일을 읽어서 사운드 버퍼에 웨이브의 내용을 적당한 크기로 분할하여 사운드 버퍼에 담은 다음, PSP_Audio() 함수를 호출하게 된다. PSP_Audio() 함수는 이렇게 넘어온 사운드 버퍼의 내용을 sceWaveAuditoWriteBlocking() 함수를 이용하여, 플레이를 하게된다.

V. 데모 게임의 구현

구현된 크로스플랫폼 라이브러리의 효용성을 검증하기 위해서, 데모 게임을 구현했다. PSP 게임 개발 환경에 전혀 경험이 없는 개발자들로 팀이 구성되어, 먼저 PC에서 코드블럭 통합 개발 환경과 SDL 라이브러리를 이용하여 족구 게임을 개발했다[그림 4].

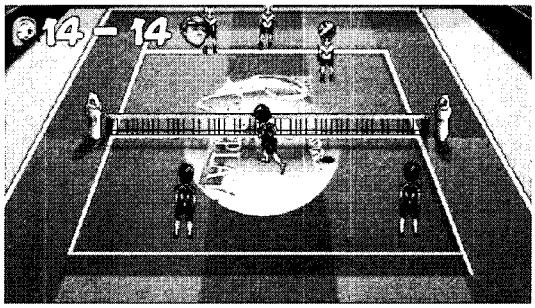

그림 4. 데모용 족구 게임

개발자들은 비주얼 C++ 환경만 접해왔지만, 코드블럭 환경을 익히는데는 큰 어려움이 없었다. PC상에서 완성된 게임의 코드를, 코드블럭 상에서 컴파일 타겟을 PSP로 하여, 컴파일한후, PSP에 올려서 실행한 결과 100% 완벽하게 PC와 동일한 작동을 하는 것이 확인되었다[그림 5][그림 6].

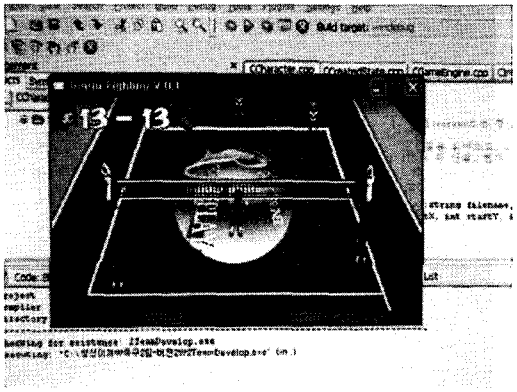

그림 5. PC 상에서의 작동 중인 족구 게임

이 과정에서 게임의 소스 코드는 전혀 수정할 필요가 없어서, 크로스플랫폼 라이브러리의 가치를 확인 할 수 있었다. 게임의 실행 속도는 PSP 쪽이 PC보다 PC 성능상의 차이로 인하여, 다소 느리게 수행되었으나, 기능적인 측면에서는 100% 호환되었다. 한가지 발견된 문제점은 데이터 파일을 로딩하는데 PSP 쪽이 많은 시간이 소모된다는 점이었다. 코드의 분석 결과, SDL에서 제공하는 이미지 로딩 함수가, 이미지를 로딩할 때 필요 메모리 용량을 줄이기 위해서, 하드 디스크를 자주 액세스하는 것이 확인되었다. PC와는 달리, PSP의 경우 파일 입출력의 소요 시간이 시스템적으로 오래 걸리는 특성이 있어, SDL의 이미지 로딩 함수에서 하드 디스크 액세스를 자주 하는 대신, 일단 메모리로 이미지를 한꺼번에 로딩하는 방식으로 수정한 결과, 데이터 파일의 로딩 시간을 대폭 줄일 수 있었다.

그림 6. PSP 상에서 작동 중인 족구 게임

VI. 결론

본 연구에서는 PSP 게임 개발의 효율성을 높이기 위해, SDL을 PSP로 이식하여, 크로스플랫폼 2D 라이브러리를 개발했으며, 이를 이용하여 족구 게임 데모를 제작했다. 그 결과, 구현된 크로스플랫폼 라이브러리가 PC와 PSP 모두에서 잘 작동하는 것이 확인되었으며, 그 작동 내용이 100% 일치하여, 크로스플랫폼 라이브러리로써의 기능을 잘 구현하고 있음을 확인할 수 있다. 개발자들은 PSP 개발 환경에 경험이 전혀 없었으나, 본 라이브러리를 통해 쉽게 PSP 게임 개발을 수행할 수 있었다.

기본적으로 크로스플랫폼 라이브러리는 PSP 게임 개발의 효율성을 높이기 위해 구현되었으나, PC 상에서 개발된 버전을 이용하여 마케팅에 활용해보는 것도 기대된다. 일반적으로 온라인 게임들은 인터넷을 통해 서비스 초기에는 무료로 제공됨으로써 사용자들의 반응을 조사하는 등, 초기 마케팅을 쉽게 할 수 있으나, PSP와 같은 콘솔 게임의 경우는 사용자들의 체험 마케팅 자체가 불가능하다. 하지만, 본 크로스플랫폼 라이브러리를 이용하면, 일단 PC용으로 개발하여, 인터넷을 통해 사용자들에게 체험판을 제공하여, 수요 조사와 더불어 초기 마케팅에 활용한 후, 이를 이용한 최종 게임 개발 및 마케팅에도 적극적으로 활용할 수 있는 장점이 있다. 현재, 본 라이브러리를 사용하여 상용 게임의 개발이 진행되고 있는 중이다. 본 연구에서 개발된 크로스플랫폼 라이브러리가 여러 게임 개발사들에게 보급되어 광범위하게 활용되는 것이 기대된다.

참 고 문 헌

[1] http://www.devmaster.net/engines/

[2] http://www.libsdl.org

[3] http://www.allegro.cc/

[4] *SONY PSP Software Development Kit*, Sony Computer Entertainment, 2005.

[5] http://www.codeblocks.org

[6] Ernest Pazera, *Focus on SDL*, Premier Press, 2003.

[7] Loki Software, Inc. and J. R. Hall, *Programming Linux Games*, No Starch Press, 2001.

저 자 소 개

이 대 현(Dae-Hyun Lee) 정회원

- 1993년 2월 : 서울대학교 제어계측공학과 학사
- 1995년 2월 : 한국과학기술원 전기및전자공학과 석사
- 2001년 2월 : 한국과학기술원 전기및전자공학과 박사
- 2001년 3월 ~ 2004년 12월 : 삼성전자 책임연구원
- 2005년 3월 ~ 현재 : 한국산업기술대학교 게임공학과 조교수

<관심분야> : 모바일 게임, PSP 및 NDS 게임 개발 및 체감형 게임 하드웨어 및 소프트웨어