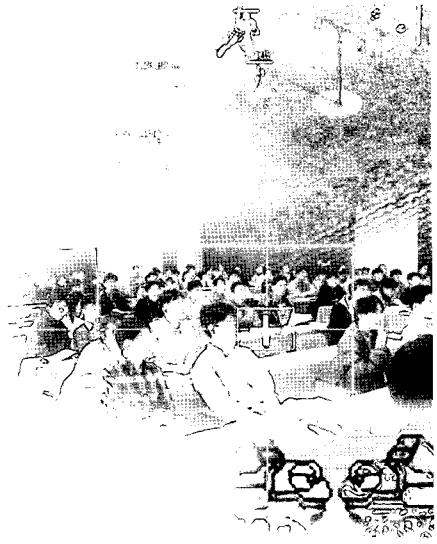

게임 프로그래밍의 교육 방향

김경식 호서대학교 게임공학전공 교수

요 약

본 고에서는 대학에서 교육시키는 게임 프로그래밍의 교육에 대해 기술한다. 게임 개발 과정 속에서 프로그래밍의 역할이 무엇인지, 학생들을 교육할 때 게임 프로그래밍 과목들의 운영은 어떻게 할지 등 게임 학과를 운영하며 게임 프로그래밍을 교육시킨 경험을 통해 그 효율적인 교육 방향에 대해 기술한다.

게임제작은 각 분야의 전문가들이 하나 된 제품을 만들어내기 위한 공동작업이므로 개발팀의 원활한 정보 공유 및 팀원간의 협력이 매우 중요하다.

1. 서론

게임 산업은 멀티미디어 기술 및 인공지능 등 첨단 정보 기술(Information Technology)이 집약된 디지털 엔터테인먼트 산업으로서 21세기를 이끌어갈 문화 산업 중에서도 가장 부가 가치가 높은 유망 산업이며, 세계 시장에서나 국내 시장에서나 디지털 경제를 이끌어갈 대표적인 산업으로 꼽히고 있다.

세계 경제는 소프트웨어, 콘텐츠 중심의 지식기반 문화 컨텐츠 산업(CT, Culture Technology)으로 급속히 이동 중에 있다. 미국은 미디어 · 엔터테인먼트 산업을 군수 산업에 이은 2대 산업으로서 미국 경제를 견인할 사업으로 육성하고 있고, 일본은 닌텐도, 소니 등의 회사를 통해 가정용 게임기 산업의 세계 시장을 이미 선점하고 있으며 영국 등 선진 각국은 컨텐츠 산업을 국가 미래 전략 산업으로 선택하여 집중 투자하면서 세계 시장 선점에 총력을 경주하고 있다.

급격한 시장 확대로 인력 수요가 커서 대학과 민간의 게임 교육기관이 다수 설립되고 있으나 업계의 수요를 충족시키기에는 배출인력의 기술력과 인원수가 크게 부족한 것이 현실이다. 가장 필요한 인력은 학교에서는 게임 교육을 체계적으로 가르칠 전문 교수 요원이며, 게임 개발 업계에서는 핵심 기술을 보유하고 있는 고급 기술 인력이다.

한편 게임 개발 업계의 가장 필요로 하는 인력은 프로그래머로서 단기간에 배출되는 인력이 아니라 더욱 수요가 크다. 아무리 게임 디자인이 우수해도 게임 프로그래밍으로 이를 구현하지 못한다면 출시를 못하는 상황이라서, 때에 따라서는 프로그래밍 기술의 한계로 인해 기획과 시나리오를 변경하기도 해야한다.

본 고에서는 대학에서 교육시키는 게임 프로그래밍의 교육에 대해 기술한다. 게임 개발 과정은 어떤 것인지, 그 속에서 프로그래밍의 역할은 무엇인지, 학생들을 교육할 때 게임 프로그래밍 과목들의 운영은 어떻게 할지 등 게임학과를 운영하며 게임 프로그래밍을 교육시킨 경험을 통해 그 효율적인 교육 방향에 대해 기술하고자 한다.

2. 게임 개발 과정과 프로그래밍

게임은 영화와 마찬가지로 문화, 지식 산업의 한 종류로서 아주 다양한 분야의 기술이 접목되어 만들어진다. 때문에 영화와 매우 유사한 점도 많이 보이지만 본질적으로 게임과 영화는 정보전달 방식체계가 다른 상이한 매체이기 때문에 제작과정에서도 많은 차이점을 보인다. 시나리오, 캐릭터, 영상, 사운드 등 영화나 게임을 구성하는 요소들은 매우 흡사하며 제작에 투입되는 스텝진(개발자) 역시 비슷한 구성을 보인다. 그러나 일방적인 감동의 전달방식이 기본구조인 영화와는 달리 게임은 게임을 즐기는 게이머로 하여금 자신의 의지를 표출시킴으로써 감동을 전달하는 구조를 갖고 있다. 따라서 게임 제작 과정의 실무 경험이 없는 사람들은 게임이 영화와 비슷하다는 느낌으로 받아들여서는 안 된다. 게임의 기본적인 성격은 쌍방향성(Interactivity)이다. 게임의 기본구조를 설계하는 디자이너, 세부 기획서를 코드로 창출해 내는 프로그래머, 도트(dot)하나에 창조적 영감을 담아내는 그래픽 디자이너 등 모든 게임 개발에 참여하는 제작진들

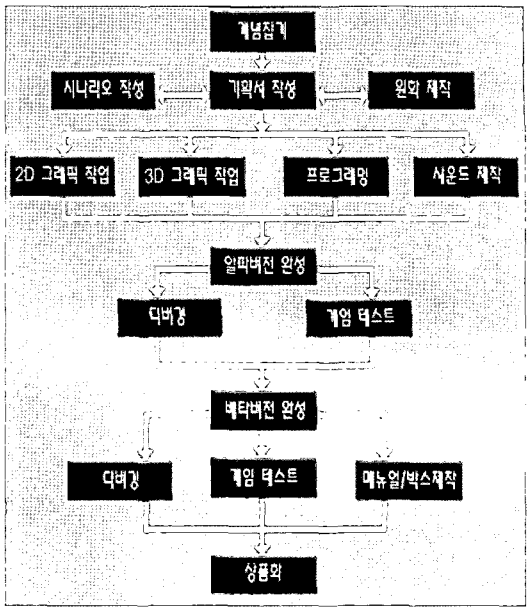

그림1 . 게임 제작 과정

이 위에서 언급한 게임의 기본 성격을 이해하고 있어야만 성공할 수 있는 게임이 만들어 질 수 있다. 더불어 게임제작은 각 분야의 전문가들이 하나 된 제품을 만들어내기 위한 공동작업이므로 개발팀의 원활한 정보공유 및 팀원간의 협력이 매우 중요하다.

이러한 게임의 제작 과정을 종합하여 그림 1에 보인다.

가. 프로그래밍

프로그래밍 언어의 종류를 열거하자면 어셈블리어(Assembly), 베이직(Basic), 파스칼(Pascal), 코볼(Cobol), 포트란 (Fortran), C, C++ 등을 들 수가 있는데 이중에서 게임을 만들기 위해 대표적으로 쓰이는 언어는 어셈블리어, C언어, C++언어 등을 들 수가 있다. 하지만, 이들 언어들은 어떤 플랫폼 (SNES, Playstation등)의 게임을 만드느냐에 따라서, 사용하는 언어가 조금은 다르다.

예를 들어서 과거 닌텐도의 SNES용 게임은 어셈블리어를 사용했지만, 소니의 PS용 게임은 C언어를 사

용한다. 또, DOS용 게임에서는 어셈블리어의 사용이 많고 매우 요긴하게 쓰였지만, 윈도우 게임에서는 단연 C++가 우세하게 쓰이고 있다.

프로그래밍 언어를 다루는 컴파일러도 DOS용 게임에서는 Watcom이란 컴파일러가 주로 사용되었고, 윈도우 게임에서는 Microsoft 사의 VisualC++이 사용되고 있다. 비디오 게임을 위한 컴파일러는 게임기를 만드는 회사에서 자체적으로 자사의 게임기에 맞는 컴파일러를 개발해서, 소프트웨어 개발자들에게 공급하고 있다.

프로그래밍 단계는 그 내부적으로 다시 여러 단계와 부분으로 나누어진다. 이러한 단계와 부분은 프로그래머의 숫자와 진행해야 할 게임의 장르에 따라 다르기 때문에 여기서는 가상적으로 3명의 프로그래머가 실시간 전략 게임이란 장르를 만든다는 것에 초점을 맞추어 단계를 나누도록 하겠다.

실시간 전략게임을 만들려면 프로그래밍에서는 크게 3등분으로 나눌 수가 있는데, 첫째는 인공지능 프로그래밍, 둘째는 보조 도구 프로그래밍, 마지막으로 네트웍 프로그래밍을 생각할 수 있다. 또한 프로그래머들은 메인 프로그래머와 두 명의 서브 프로그래머로 나누어진다.

프로그래밍 기획 초반에 메인 프로그래머는 두 명의 서브 프로그래머와 함께 기획에서 주어진 게임 제작의 완성을 위해서, 어떻게 전체 게임 프로그래밍의 골격을 잡아야 할지 기획회의를 갖고 세부적인 작업에 들어간다. 여기서 기본 설계가 끝나면, 메인 프로그래머는 주로 인공지능을 담당하게 되며, 전체 골격을 만들어 가는 작업을 시작한다.

이때, 서브 프로그래머 1명은 보조 도구들을 만들어 나간다. 여기서 보조 도구들이란 스프라이트 툴(sprit tool), 에니메이션 툴(animation tool : 에니메이션 툴에서 스프라이트 툴 기능도 지원하는 경우도 있다), 그리고 맵 툴(map tool) 등을 말하는데, 이러한 보조 도구들은 제작되는 게임의 성격에 맞게 주문 제작되어 야지만 효율성이 높기 때문에, 이 도구를 실지로 사용하는 디자이너 등과 상의 하에 툴을 제작한다.

보조 도구 제작 기간은 다른 프로그래밍 분야에 비해서 제작에 많은 기간을 두지 않으며, 기본적인 모양이 나오면 제작을 어느 정도 완료하고 게임이 끝날 때까지 필요할 때마다 조금씩 수정하여 단계를 높이는 것이 보통이다. 이 작업을 끝낸 서브 프로그래머는 다시 메인 프로그래머를 도와서 본격적인 게임 제작에 참여하게 된다.

나머지 한 명, 네트웍 (network) 프로그래밍을 담당하는 서브 프로그래머는 게임프로그래밍의 초반부터 종반까지 이 한가지 일만을 담당하는 것이 보통이다. 그것은 실시간 전략 시뮬레이션의 경우에는 네트웍 상에서 많은 수의 사람들이 멀티 플레이가 가능하도록 해야 하기 때문에 네트웍 프로그램 제작에는 많은 시간과 노하우가 필요하기 때문이다.

게임 제작에 있어서 프로그래머들은 제작 초반에는 여유가 많고 후반에 갈수록 시간에 쫓기는 경우가 대부분이다. 하지만, 숙련되고 경험 많은 프로그래머일수록 전체적인 시간분배에 실패하지 않는다. 이것이, 전체 게임 제작의 시간 내 완성을 좌우하는 비결이다.

나. 디버깅

디버깅이란 게임 테스트에서 발견된 문제점들을 재검토하고, 이를 빠르게 수정하는 것을 의미한다. 디버깅하면 보통 프로그램만 생각하기 쉬운데, 프로그래머, 디자이너, 기획자 등 게임 제작을 담당한 모두에게 해당한다. 외국의 대작 게임일수록 버그가 없다는 사실은 그만큼은 디버깅을 시간적 여유를 두고 충분히 했다는 이야기이다. 아무리 재미있는 게임이라도 여기저기서 버그가 발견된다면, 게이머들에게 엉성한

게임이라는 불신감을 줄 수밖에 없다.

기획이 훌륭하고 빈틈없는 개발자들이 뭉쳤다고 하더라도 게임 제작은 팀 작업이기 때문에 여러 가지 이유로 많은 버그를 만들게 된다. 따라서, 게임 초반, 스케줄을 작성할 때 디버깅 기간을 충분히 두어서 버그 없는 (bug free) 게임을 제작해야 한다. 이것이 개발사에게 있어서 게임의 성공 못지않게 중요한 것은 우리의 게임을 사준 소비자에게 우리가 해야 할 가장 기본적인 서비스이기 때문이다.

다. 게임테스트

알파버전이 나온 후의 테스트는 베타버전으로 진행하기 위해서 수정되어야 할 문제점들을 조기에 발견해 내기 위하여 행하여지는 테스트이다. 이 과정을 통해서 처음의 기획대로 어떤 부분이 처리가 잘 되었고 어떤 부분이 미흡한지, 그리고 어떤 부분은 필요가 없고, 어느 부분이 첨가되면 좋은지를 하나 하나 검증하게 된다. (개념적 생각에서만 기획되고 만들어진 게임 요소들은 실제로 게임화 했을 경우 생각했던 것 보다 재미가 없거나 부자연스러운 분위기를 연출하는 경우가 많기 때문이다. 역으로 이 과정을 통해서 생각하지 못했던 새로운 아이디어가 많이 떠오르기도 한다)

이 때의 테스트는 주로 개발하는 사람들 자체적으로 테스트를 하게 되며, 우리가 흔히 생각하는 실질적인 게임 테스트라고 할 수 있겠다. 알파버전 때의 테스트가 다분히 기술적인 부분을 보는 것이라면, 여기서의 테스트는 개발자가 아닌 사용자 입장에서의 테스트가 가해진다. 게임이 얼마나 잘 되어 있는가?, 재미는 있는가?, 버그는 없는가?, 진행은 매끄러운가? 등 다분히 추상적인 문제이고, 게임으로 보면 굉장히 중요한 요소인데, 이 과정을 통해서 게임의 재미 요소를 극대화시키는 작업을 하게 된다. 또한, 게임의 난이도를 결정하는 문제도 여기서 정하게 된다.

이 게임 테스트를 위해서, 실지로 게임을 많이 하는 청소년층의 게이머들을 고용하는 경우가 많으며 정성을 들인 게임일수록 이 테스트 기간의 길이가 길어진다. 이 마지막 테스트는 게임의 성격에 따라 다르지만, 짧게는 2~3개월에서 대작인 경우는 드물게 1년에서 2년 가까이 하는 경우도 있다.

3. 게임 프로그래밍 교육 방법 고찰

3.1 실습 환경

▶ 1인1대 PC (Visual C++, DirectX SDK 설치 필요)

게임 프로그래밍을 교육하는데는 실습실이 필수적으로 필요하다. 1인 1대의 PC실습 환경이 갖추어져야 하며, PC는 윈도우 프로그래밍이 가능하도록 Visual C++ 컴파일러가 설치되어 있고, DirectX SDK 가 설치되어 있어야 한다.

▶ 다목적 수업을 위해 PC들은 LAN으로 연결되어 외부 인터넷 또는 인트라넷에 접속이 가능해야 한다.

▶ 위 정도의 기본 시설이며 게임 프로그래밍을 교육시킬 수 있다. 물론 그래픽 모델링 작업이나 컴퓨터 음악 및 음향 효과의 데이터는 LAN을 통해 전달받아야 한다.

3.2 과목 운영

3.2.1 프로그래밍 언어(C/C++)

선수 과목으로서 프로그래밍 작성 방법에 대해 터득하면서 자기 스타일의 코딩 기법을 개발하고 자기 나름대로의 코드 라이브러리를 만들어 나간다. 소요기간은 4년제 대학에서는 1학년 두학기 동안 기초 선수 과목으로 배울 만 하다(주4시간씩 – 이론 2시간, 실습 2시간). 게임 기획이나 게임 그래픽 담당할 사람

이라도 기본 프로그래밍을 할 수 있도록 유도하여 종합적인 식견을 갖추게 한다. 단기간으로는 한학기 또는 2달만에도 기본을 습득할 수 있다.

본 과목은 실습으로 소스코드를 꼭 확인하고 스스로 적용해보는 것이 필요하다. 본인이 코딩해 보는 라인수가 자기 실력이 될 것이다. 많이 짜 본 사람이 짧은 시간에 정확한 프로그래밍을 할 수 있을 것이다. 실습 환경은 Visual C++ 컴파일러를 갖춘 PC실(1인 1대)이다.

교재에 나와 있는 소스코드는 물론 컴파일 하면서 확인하고 변형시켜 보면서 자기 코드로 습득할 수 있도록 하고 바람직한 것은 교재의 매 장 뒤의 코딩 연습 문제를 매일 1, 2개씩 해보는 것이다.

3.2.2 윈도우 프로그래밍

PC Windows 95/98/2000/XP에서 이벤트 발생과 처리를 어떻게 하는지 API 들을 잘 활용할 수 있도록 실습을 병행해야 한다. 실습 환경은 Visual C++ 컴파일러를 갖춘 PC실(1인 1대)이며 C/C++ 언어를 알고 있어야 한다. 과목 교육 기간은 최소 2달(주4시간씩 – 이론 2시간, 실습 2시간)에서 앞뒤의 연결 및 API 응용 과제를 겸하여 1학기를 잡는 것이 바람직하다.

앞뒤의 연결은 C/C++언어의 간단한 복습 및 시험이며, 매 실습시간에는 교재에 나와 있는 소스코드를 컴파일 하면서 확인하고 변형시켜 보면서 자기 코드로 습득할 수 있도록 해야하고 바람직한 것은 API 들로 윈도우 게임을 제작하도록 과제를 주는 것이다.

예를 들어 지뢰찾기 등 마우스 클릭으로 상황(이벤트)을 발전시켜 나갈 수 있는 게임을 제작하도록 학기초에 과제를 주고 강의와 실습을 진행해 나가면 학생들이 목표 의식을 갖고 공부해 나갈 수 있을 것이다.

3.2.3 DirectX

국내 출판사들이 2002년부터 DirectX 8.0 버전의 외국 서적들을 번역하여 출간하고 있으며 국내 기술로 출간 또는 준비중인 책들이 있다. 아울러 MicroSoft에서 제공하는 문서들과 소스를 분석하여 파악하고 있다.

실습 환경은 Visual C++ 컴파일러를 갖춘 PC실(1인 1대)이며 C/C++ 언어와 윈도우 API들을 알고 있어야 한다. 과목 교육 기간은 2학기 (주4시간씩 – 이론 2시간, 실습 2시간)를 잡는 것이 바람직하다. 1학기에는 2D게임 프로그래밍을 가르치고 2D 게임 제작을 위해 DirectX Graphics(7.0 DirectDraw로 1달 병행) 와 DirectInput, DirectX Audio 등을 실습과 함께 진행한다. 2학기에는 3D게임 프로그래밍으로 버텍스 쉐이딩, 픽셀 쉐이딩 등 본격적인 DirectX Graphics와 온라인 게임을 지원하는 Direct Play를 배울만 하다.

과목 시작하면서 2D와 3D 슈팅게임을 보여주고 DirectX를 배우면서 과제로 게임을 만들어 보게하는 것이 좋다. 공부할 때의 집중도와 응용력이 높이 훈련될 것이다.

3.2.4 게임 엔진

자주 사용되는 루틴들을 모아놓은 것을 엔진이라 한다. 특히 게임을 개발하면서 사용하는 루틴들은 다른 게임에도 사용될 수 있어 게임 엔진은 계속 향상 진보 시켜가면서 게임 개발과 함께 나가고 있다.

Genesis3D 등 외국 게임엔진은 고가로 판매되고 있어 일반으로 사용하기 어려워, 국가에서 지원하여 저렴하게 사용할 수 있는 3D게임 엔진을 개발하였다(문화관광부 Infinity3D, 정보통신부 Dream3D). 그 전까지 이미 자체 게임엔진을 보유하고 있는 회사들은 게임 개발자들의 경험을 통해 축적된 코드들을 가

대다수의 회사들이 게임 프로그래머를 찾고 있다. 그만큼 쉽게 배출되지 않으며 프로그래머 본인이 실습으로 노력하면서 실력을 터득해야 한다.

지고 있게 있다. 그러나 이 엔진들에 대해서 검증을 못 받았기 때문에 한계가 어디까지인지 알 수 없다.

게임 엔진은 게임 제작을 쉽게 할 수 있게 해준다. 윈도우 API 호출하는 것처럼 DirectX를 부르거나 만들어진 함수를 호출하여 게임을 크게 만들어 나간다. 그러나 게임 시나리오나 기획 의도를 살리기 위해 때로는 게임 엔진을 변형시키기도 해야 한다.

수업에서 접근할 수 있는 게임 엔진 소스를 각자 구해놓고서 (인터넷 상에서 100불 정도면 쓸만한 것을 구입 가능, 호서대에서는 한국과학재단의 지원으로 그간의 기술을 집약한 온라인3D엔진을 개발하여 무료 공개중) 기능을 터득하고 바로 게임 제작에 들어가는 훈련을 받는다.

4. 맺음말

본 고에서는 대학에서 교육시키는 게임 프로그래밍의 교육에 대해 기술하였다. 게임 개발 과정은 어떤 것인지, 그 속에서 프로그래밍의 역할은 무엇인지, 이러한 기술을 습득하기 위한 방법과 절차는 어떤 것이 있는지, 학생들을 교육할 때 게임 프로그래밍 과목들의 운영은 어떻게 할지 등 게임학과를 운영하며 게임 프로그래밍을 교육시킨 경험을 통해 그 효율적인 교육 방향에 대해 기술하였다.

현 게임산업에서 대다수의 회사들이 게임 프로그래머를 찾고 있다. 그만큼 쉽게 배출되지 않으며 프로그래머 본인이 실습으로 노력하면서 실력을 터득해야 한다. 2년제 대학에서는 기간이 짧아 목표하기가 쉽지 않을 것 같다. 4년제는 1학년때부터 C/C++ 언어를 배우고 2학년때 윈도우 프로그래밍과 DirectX (3학년에도 계속)를 배우면 좋을 것이다. 자신들의 실력 향상을 위해서 방학중 업체에 현장 실습을 나가거나 동아리의 선후배 간을 통해 게임개발기술을 연마하는 것도 좋은 계획이다.

게임 프로그래머의 길은 본인의 꾸준한 노력과 새것에 대한 끊임없는 공부가 있어야 한다. 기획자와 그래픽 디자이너와 함께 협력하면서 좋은 게임을 만들어 보는 경험을 쌓는다면 현장에서 바로 활용할 수 있는 인력이 될 것이다.

참고 문헌

<자료, 통계>

[1] 2002 대한민국 게임백서, (재)게임종합지원센터, 2002. 3.

[2] "CT산업 발전전략", 차세대 성장산업 발전전략회의 (제8차 국민경제자문회의), 교육인적자원부 · 문화관광부 · 산업자원부 · 정보통신부, 2001. 8. 17.

[3] 2002 국내외 게임시장 현황통계조사보고서, (사)한국첨단게임산업협회, 정보통신부, 2002. 6.

[4] 온라인게임 산업 육성방안, 한국전자통신연구원, (사)한국첨단게임산업협회, 정보통신부, 2000. 12.

<기획, 시나리오>

[5] 앤드류 롤링스, 데이브 모리스, 한쿨임 팀 역, 게임 아키텍처 & 디자인, Coriolis, 제우미디어, 2001. 3.

[6] 서민철, 게임 시나리오 작법, 컴피플, 2000. 5.

[7] 유형오, 게임 비지니스 엔진, 테크북, 2001. 8.

[8] 밥 베이츠, 송기범 역, 게임 디자인: 아트 & 비지니스, Premier Press, 제우미디어, 2001. 10.

[9] 박상호역, Game Design Secrets fo the Sages 이것이 게임기획이다! 150인 고수들의 실전 노하우, 민프레스, 2001. 7.

<C/C++언어>

[10] 황희융역, C언어 기초+α, 교학사, 1997. 1.

[11] Eric S. Roberts, The Art and Science of C, Addison Wesley, 1994. 12.

[12] H. M. Deitel, P. J. Deitel, 구자영 역, C++와 객체 지향, 대영사, 1998. 1.

<윈도우 프로그래밍, 2D>

[13] Ian Parberry, 이주리역, Learn computer Game Programming with DirectX, 민프레스, 2001. 5.

[14] Robert Dunlop with Dale Shepherd and Mark Martin, Teach Yourself DirectX 7 in 24 Hours, SAMS, 2000.

[15] Andre LaMothe, 이주리역, Tricks fo the Windows Game Programming Gurus, 민프레스, 2000. 12.

[16] DirectX8 실천 프로그래밍, 공학사(일), 2001. 4.

<3D 프로그래밍>

[17] Jone De Goes, T3엔터테인먼트팀역, 3D 게임 프로그래밍 with C++, C Group, Coriolis, 2001. 9.

[18] Adrian Perez with Dan Royer, Advanced 3-D game Programming Using DirectX 7.0, Wordware Publishing, Inc. 2000.

[19] Adrian Perez with Dan Royer, 이주리역, Advanced 3D Game Programming using DirectX, 민프레스, 2001. 2.

[20] Daiyun Nobori, DirectX8.0 3D 액션게임 프로그래밍, 공학사(일), 2001. 3.

<네트워크 프로그래밍>

[21] Todd Barron, Multiplayer Game Programming, Prima Tech, 2000.

[22] 인피니티, 3D 네트워크 게임 프로그래밍 가이드, Shuwa System Co. Ltd(일)., 2001. 4.

[23] 김경식, 온라인3D 슈팅게임 만들기, 형설, 2001. 9.

<기타: PDA, 핸드폰 게임 프로그래밍>

[24] Jonathan S. Harbour, 조재권역, Pocket PC Game Programming: Using the Windows CE Game API, 민프레스, 2001. 9.

[25] Michael Morrison, Tech Yourself Internet Game Programming wth Java in 21 Days, sams.net, 1996.