차 례

0 들어가며

(원래 출판을 목적으로 썼던 책인데 출판사 사정으로 책 출판이 어려워져 인터넷에 그대로 공개합니다.)

0.1 들어가며

안녕하세요. 캐나다 렐릭 엔터테인먼트에서 시니어 그래픽 프로그래머로 일하고 있는 포프입니다. 이 쉐이더 입문 책은 제가 2007년 1월부터 2009년 12월까지 3년간 캐나다의 The Art Institute of Vancouver 대학에서 쉐이더 프로그래밍 강의를 하면서 축적한 자료와 지식을 글로 옮겨 놓은 것입니다.

0.2 책을 쓰게 된 배경

2007년에 제가 강의를 시작할 때, 수업시간에 사용할 교과서를 찾으려고 참 많은 노력을 했습니다. 하지만 시중에 나와있는 책들 중, 쉐이더 입문과목에 적합한 놈이 없더군요. (몇 년이 지난 지금에도 마찬가지인 것 같습니다.) 시중에 나와있는 쉐이더 책들은 대부분 이미 쉐이더 코드를 짤 줄 아는 중고급 프로그래머를 위한 것이었습니다. 따라서 쉐이더에 입문하는 학생들이 보면 뭔 소린지 몰라서 그냥 포기할 게 뻔했죠. 그나마 쉐이더 입문 내용이 DirectX 책에 담겨있는 경우가 있었지만 그 중에서도 마땅한 책이 없다고 생각했던 이유가

- 쉐이더는 구색 맞추기 식으로 넣어놓아서 너무 수박 겉 핥기 식이다.
- 학계에 계신 분들이 쓴 책은 너무 이론이나 문법에만 치우쳐져 있다.
- 실무에 그다지 쓸모가 없는 내용들을 너무 많이 담고 있다.
- 지면수만 많아 책값이 너무 비싸다.

등 이었습니다. 그래서 결국엔 교과서 없이 강의를 시작했죠. 이론이나 수학에 치우치기 보다는 실무에 곧바로 쓸 수 있거나 실무에서 쓸 수 있는 기법의 기초가 되는 내용들만을 가르치는데 주력했습니다. 강의를 하면서 좋았던 점은 저는 그리 어렵지 않다고 생각해왔는데 학생들이 이해하지 못하는 부분들을 알아낼 수 있다는 거였죠. 그래서 그걸 다시 쉽게 이해시킬 수 있도록 강의자료를 다듬고 다듬은 결과가 바로 이 책입니다. 강의를 하는 3년 내내 게임프로그래밍 학과 학생들이 이 과목을 AI 대학의 가장 훌륭한 수업으로 꼽을 정도였으니 (좀

부끄럽지만) 이 책을 자신있게 권해드릴 수 있을 것 같습니다. 그리고 제 과목에서 만든 데모 프로그램을 포트폴리오로 삼아서 Ubisoft 및 EA같은 세계 유수의 회사에 취직한 학생들도 몇 됩니다. 가슴 뿌듯한 일이죠. M

현재는 게임개발에 좀 더 집중해 보려고 대학강의를 중단한 상황이지만 이 내용을 그냥 썩혀두기엔 아깝다고 생각되어 책을 내기로 결정을 했습니다. 이 책이 쉐이더를 배우시려는 분들에게 도움이 될 수 있었으면 좋겠습니다.

0.3 이 책의 기본원칙

강의에서도 그랬듯이 이 책을 쓸 때 다음의 원칙을 기초로 삼았습니다.

- 실습 위주: 물론 쉐이더를 짤 때 수학이나 이론을 전혀 무시할 수는 없습니다. 하지만, 이론을 먼저 배우고 그걸 코드로 옮기는 것보단 일단 코드를 좀 짜본 뒤에 뭔가 막히면 이론을 찾아보는 것이 훨씬 훌륭한 학습방법입니다. 이렇게 문제를 해결하기 위해 찾아본 이론은 기억에 오래 남습니다. 따라서 이 책은 실습위주로 구성되어있습니다. 책의 내용을 한 줄씩 따라하면서 코드를 짜다 보면 어느덧 배경 이론까지 적당히 이해하시게 될 겁니다.

- 쉬운 설명: 제 수업에 청강을 하러 오는 게임아트 학과 학생들도 꽤 있었습니다. 따라서 아티스트들도 이해할 수 있도록 쉽게 설명을 하는 것이 제 목표 중 하나였습니다. 그러려면 무언가를 설명할 때, 수학공식을 보여주기 보다는 실제 생활에서 일어나는 현상을 예로 드는 것이 낫더군요. 이 책을 쓸 때도 마찬가지 원칙을 따랐습니다. 책을 읽으시다 보면 100% 이론적으로 옳지 않은 설명들도 가끔 보실 겁니다. 이것은 말 그대로 설명을 쉽게 하기 위해 제가 이론들을 적당히 무시하였거나 아니면 저 조차 이론을 100% 제대로 이해하지 못하는 경우입니다. 게임 그래픽은 어차피 눈에 보이는 결과가 맞으면 그게 정답인 분야이므로 이론적으로 약간 틀려도 결과만 맞으면 전 크게 신경 쓰지 않습니다.

- 입문자만을 위한 책: 이 책은 순수하게 입문자를 위한 책입니다. 이미 고급기법을 다루는 훌륭한 책이 많이 나와있는 상황에서 굳이 그 책들과 경쟁할 필요를 못 느끼고, 중복되는 내용을 다루면서 지면수를 늘리고 싶지도 않기 때문입니다. 이 책을 보신 후에 쉐이더에 재미가 붙으신 분들이 다른 고급기법들을 즐겁게 찾아 보실 수 있다면 전 행복합니다. 그리고 정말 괜찮은 새 기법을 찾으시면 저에게 살짝 귀띔이라도 해주시면 더 좋겠지요. M

- 순서대로 배우는 내용: 강의를 할 때 좋았던 점은 쉬운 내용부터 어려운 내용까지 순서대로 가르칠 수 있었다는 것입니다. 이 책도 그런 식으로 진행이 됩니다. 처음 장부터 시작해서 천천히 지식을 축적해간다고 할까요? 따라서 뒷장으로 가면 갈 수록 기본적인 내용은 다시 설명을 하지 않습니다. 예를 들면, 법선매핑을 배우기 전에 이미 조명기법들을 배워보므로 법선매핑에서는 조명기법에 대해 다시 설명하지 않는거죠. 따라서 이 책을 읽으실 때는 대학강의를 들으시듯이 처음부터 순서대로 읽으셔야 합니다. 이 책이 다른 쉐이더 책들처럼 여러 논문을 한군데 모아놓은 게 아니니 그 정도는 이해해주시면 좋겠습니다. 그리고 지면수도 그리 많지 않으니 무리한 요구는 아닐 거라고 믿습니다.

0.4 이 책에서 다루는 내용

이 책에서 다루는 내용은 정점쉐이더와 픽셀쉐이더를 이용한 쉐이더 기법들입니다. 이 책은 크게 세 부분으로 나뉘어져 있습니다.

- 제1부: 쉐이더의 정의를 알아 본 뒤, 모든 쉐이더 기법의 기초가 되는 색상, 텍스처 매핑, 조명 쉐이더를 만들어 봅니다.
- 제2부: 1부에서 배운 내용에 살을 붙여 게임에서 널리 사용하는 스페큘러 매핑, 법선매핑, 그림자 매핑 등의 기법들을 구현합니다.
- 제3부: 요즘 게임에서 점점 중요해져 가는 2D 이미지 처리 기법을 배워봅니다.

이 책에서 DirectX 10과 11에서 새로 추가된 지오메트리(geometry), 헐(hull), 연산(compute) 쉐이더들을 다루지 않는 이유는 초급자에겐 좀 어려운 내용일 뿐만 아니라 아직 실무에서 널리 이용되지 않기 때문입니다. 따라서 실용적인 내용을 알려드리기가 좀 어렵죠. DirectX 10이 처음 소개될 때만 해도 홍보자료에서는 엄청 대단한 것처럼 광고를 해댔지만 실제 실무에서 제대로 이용한 경우가 없으니까요. 일단 이 책에서 기초를 다잡으시면 몇 년 뒤에 이 내용을 배우셔도 크게 문제가 없을 겁니다.

0.5 대상독자

프로그래머

제가 가르쳤던 학생들은 게임 프로그래밍 학과 2학년 학생들이었습니다. 제 과목을 듣기 전에 C++, 3D 수학, DirectX 등을 이미 마친 학생들이었지요. 게임개발자 분들이 쉐이더 프로그래밍에 입문하는 과정도 이와 다르지 않다고 생각합니다. 최소한 DirectX는 마치신 뒤에 쉐이더를 살펴보시는 게 보통이니까요. 이 책의 대상독자도 마찬가지로 하겠습니다. 이 책을 보시려면 최소한 C++과 DirectX 정도는 공부하셨어야 합니다. 3D 수학까지 아시면 더 도움이 되겠습니다.

테크니컬 아티스트

요즘 들어 프로그래머와 아티스트 사이를 조율해주는 테크니컬 아티스트 분들의 입지가 높아지고 있습니다. 그리고 이제 테크니컬 아티스트들이 쉐이더 프로토타입을 만드는 경우도 허다합니다. 강의를 하는 도중에 일반 아티스트(청강생)들도 어느 정도 이해를 했던 내용들이니 테크니컬 아티스트 정도 되시면 아무 문제가 없으시겠지요? 테크니컬 아티스트들은 굳이 DirectX를 직접 다루지 않아도 되니 별다른 준비사항 없이 이 책을 보셔도 될 것 같습니다. (보시다 이해가 안 되는 수학 같은 게 있으시면 정석 책을 열어보시거나 인터넷 검색을 좀 하셔야 할지도 모르지만요. i 각 장의 마지막에 DirectX 프레임워크를 다루는 부분이 있는데 그 부분만 건너 뛰시면 됩니다.

0.6 온라인 커뮤니티

이 책을 보시다가 궁금하신 것이 있으시면 제 블로그로 오시기 바랍니다. 토론장을 열어두겠습니다. 그 외에 정오표나 기타 업데이트들도 이 사이트를 통해 공개할 예정입니다.

`http://kblog.popekim.com/`

(사실 이미 제 블로그에 이 글을 올리는 마당에 정오표나 기타 업데이트들을 굳이 따로 올릴 필요가 있나 모르겠습니다. −_−)

0.7 감사의 말씀

이 책이 나오기까지 많은 분들이 도움을 주셨습니다. 이 자리를 빌어 감사의 뜻을 표현하는 것이 최소한의 도리라고 생각합니다.

우선 이 책을 쓸 수 있는 계기를 마련해주신 조진현님께 감사의 말씀을 드리고 싶습니다.

강의실에서 학생들과 직접 얼굴을 맞대면서 가르친 내용을 책으로 옮기는 건 사실 쉬운 일이 아니었습니다. 강의실 환경과는 달리 책은 일방적인 의사소통 수단이어서 과연 제가 말하고자 하는 바가 독자분들께 잘 전달이 될런지 매우 걱정

이 되더군요. 이 때, 이 책의 내용과 샘플코드들을 꼼꼼히 테스트 해주신 개발자 분이 두 분 계십니다. 두 분 다 제 대상독자층에 속한 분이셨죠. 한 분은 이미 게임개발업계에 꽤 계셨지만 쉐이더 프로그래밍은 안 하셨던 분이고, 다른 분은 일반 프로그래머 일을 시작한 지 얼마 안 되시는 분입니다. 이 분들이 책을 처음부터 끝까지 꼼꼼히 읽어주시고, 코드를 한 줄 씩 직접 테스트해 주신 덕에 잘못된 내용을 최소한으로 줄일 수 있었습니다. 또한 이 분들이 보내주신 피드백에 따라 부족한 내용을 보완한 덕에 더욱 튼실한 책을 만들 수 있었죠. 유스하이텍의 이경배님과 네오플의 송진영님, 정말 많은 도움이 되었어요. 고맙습니다.

마지막으로 이 책의 준비 단계부터 블로그와 트위터를 통해 많은 관심을 가져주시고 응원해주신 전직/현직/미래 게임개발자 분들과 일반인(?) 분들께 감사의 말씀을 드리고 싶습니다. 강다니엘, 고경석, 김동환, 김성완, 김영민, 김정현, 김혁, 김호용, 박경희, 박수경, 손기호, 신성일, 안진우, 유영운, 이경민, 이상대, 최재규 님, 책 나왔어요

2011년 3월 캐나다 밴쿠버에서 포프 올림

1 쉐이더란 무엇이죠?

1.1 제1장 쉐이더란 무엇이죠?

쉐이더의 정의

제가 학생들을 가르치면서 제일 처음 듣는 질문 중 하나가 '도대체 쉐이더가 뭐에요?'였습니다. 사실 뒤돌아보면 제가 쉐이더를 처음 접할 때도 스스로 이런 질문을 던지곤 했었는데 그 누구도 저에게 이해하기 쉽도록 '쉐이더란 바로 이런 것이다!' 라고 설명을 해준 적이 없더군요. 또한, 기존의 자료들도 찾아봤는데 학생들이 쉽게 이해할만한 정의를 내려주는 자료를 찾을 수 없었습니다. 그래서 제 맘대로(?) 아주 쉽게 정의를 내렸습니다. **쉐이더란 화면에 출력할 픽셀의 위치와 색상을 계산하는 함수입니다.** 어휘적/구조적인 측면에서 쉐이더를 살펴보면 이를 자세히 이해할 수 있을 겁니다.

어휘적 접근 사실 영어만 잘해도 거저 주워 먹는 것이 많은 분야가 컴퓨터 프로그래밍입니다. 쉐이더만 해도 크게 예외는 아닌데요. 쉐이더(shader)란 '색의 농담, 색조, 명암 효과를 주다.'라는 뜻을 가진 shade란 동사와 행동의 주체를 나타내는 접미사 '-er'을 혼합한 단어입니다. 즉, 색의 농담, 색조, 명암 등의 효과를 주는 주체가 쉐이더란 뜻이지요. 컴퓨터 그래픽에서 색이라 하면 당연히 화면에 등장하는 픽셀의 색상이므로 이를 다시 정리하면 다음과 같습니다.

쉐이더는 픽셀의 농담, 색조, 명암을 결정한다. 여기서 농담, 색조, 명암이라고 하니 '아니, 그렇다면 쉐이더가 출력하는 결과가 3개나 된다는 말인가요?'라고 하시는 분들이 계실 듯한데 그런 것은 아닙니다. 쉐이더의 최종결과는 농담, 색조, 명암 효과를 전부 짬뽕해서 나온 RGBA색상 값 하나입니다. (쉐이더가 반드시 한가지 색상만을 출력해야 하는 것은 아닙니다. 고급 쉐이더 기법들에서는 다수의 결과를 동시에 출력하는 경우가 있습니다.) 미술시간에 수채화 그려봤던 것 기억하시죠? 일단 밑그림을 완성하면 물감의 색을 고르고, 여기에 물을 혼합시키는 양을 바꿔가면서 다양한 명암효과를 냅니다. 하지만, 일단 그림이 완성되면 캔버스에 있는 결과는 결국 최종색상뿐이죠? 쉐이더도 이와 마찬가지입니다. 온갖 기법들을 이리저리 섞어서 픽셀들의 최종 색상 값을 구하는 것이 바로 쉐이더입니다.

구조적 접근 저희가 이 책에서 다루는 쉐이더는 정점쉐이더(vertex shader)와 픽셀쉐이더(pixel shader)인데 위의 어휘적 접근에서 살펴봤던 쉐이더의 정의는

이 중 하나에만 적용됩니다. 어떤 것일까요? 네, 그렇습니다. 픽셀쉐이더 입니다. 그렇다면 정점쉐이더란 무엇일까요? 이걸 이해하려면 3D 그래픽파이프라인의 구조를 살펴봐야겠군요.

3D 파이프라인이 존재하는 이유 중 하나는 3차원 공간에 존재하는 물체를 컴퓨터 모니터라는 2차원 평면 위에 보여주기 위해서입니다. 우선 3D 그래픽파이프라인을 극단적으로 간략화시킨 그림 1.1을 살펴봅시다. (이 그림은 정점쉐이더 및 픽셀쉐이더의 역할을 이해하기 위해서 극단적으로 간략화시킨 버전입니다. 실제 그래픽 파이프라인은 이 그림에 나와 있는 것보다 훨씬 복잡합니다.)

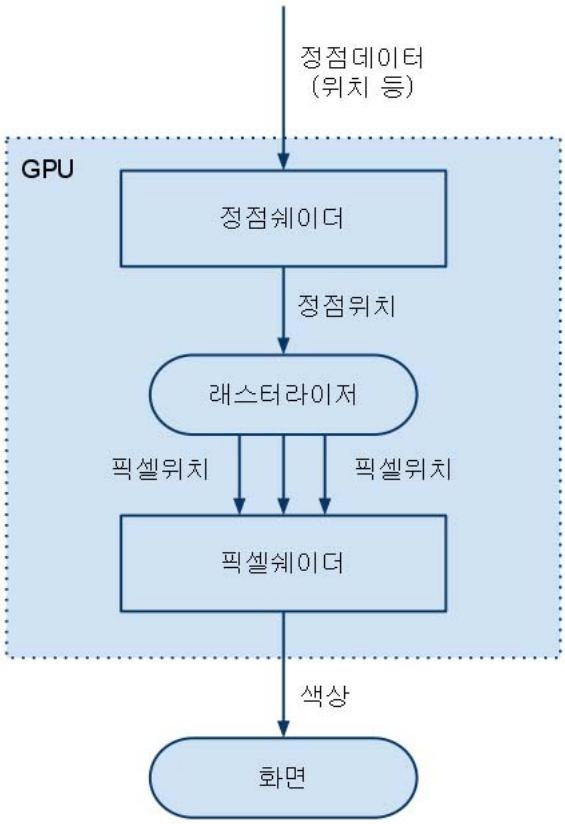

그림 1: 극단적으로 간략화시킨 3D 파이프라인

그림 1.1에서 정점쉐이더가 입력 값으로 받는 것은 3D 모델 자체입니다. 3D 모델은 폴리곤(polygon, 다각형)으로 구성하는 것이 업계표준인데, 폴리곤이란 결국 삼각형들의 집합에 지나지 않습니다. 삼각형은 3개의 정점(vertex, 꼭짓점

이라고도 합니다) 으로 이뤄져 있죠? 그러니 정점데이터가 정점쉐이더의 입력 값이라고 해도 전혀 틀린 게 아니겠네요.

정점쉐이더가 수행하는 가장 중요한 임무는 3D 물체를 구성하는 정점들의 위치를 화면좌표로 변환하는 것입니다. 이를 화가에 비유한다면 투시원근법을 사용하여 실제세계에 있는 물체들을 캔버스 위에 옮겨 그리는 과정이라 할까요? 이렇게 물체의 위치를 다른 공간으로 옮기는 과정을 공간변환(space transformation)이라고 부르는데 이에 대한 자세한 설명은 다음 장에서 드리도록 하겠습니다. 조금 전에 3D 모델은 결국 정점들의 집합이라고 말씀드렸었죠? 따라서 모든 정점을 하나씩 공간 변환하면 3D 물체 자체를 공간 변환하는 것과 똑같은 결과를 얻을 수 있습니다. 이게 바로 정점쉐이더가 하는 일이지요. 그렇다면 정점쉐이더 함수는 몇 번이나 호출될까요? 다음 문장을 보시면 답을 아실 수 있겠네요.

정점쉐이더의 주된 임무는 각 정점의 공간을 변환하는 것이다. 네, 그렇습니다. 정점쉐이더는 3D 물체를 구성하는 정점의 수만큼 실행됩니다.

정점쉐이더가 반드시 출력하는 결과 값은 화면공간 안에 존재하는 정점의 위치(이외에도 다양한 정보를 정점쉐이더의 결과 값에 담을 수 있습니다. 자세한 내용은 이 책의 뒷부분에서 살펴볼 것입니다.)입니다. 이 위치를 3개씩 그룹 지으면 화면에 출력할 삼각형을 만들 수 있지요.

자, 그렇다면 이 삼각형 안에 픽셀이 몇 개나 들어갈까요? 화면을 구성하는 단위는 픽셀이니까 화면에 뭔가 그림을 그리려면 픽셀을 어디에 몇 개나 그려야 하는지를 알아야겠죠? 이게 바로 래스터라이저(rasterizer)란 장치가 하는 일입니다. 래스터라이저는 정점쉐이더가 출력하는 정점의 위치를 차례대로 3개씩 모아 삼각형을 만든 뒤, 그 안에 들어갈 픽셀들을 찾아냅니다. 자, 그러면 픽셀쉐이더 함수는 몇 번이나 호출될까요? 래스터라이저가 찾아내는 픽셀 수 만큼이겠죠?

그렇다면 위에서 보여드렸던 초 간략 파이프라인의 마지막 단계인 픽셀쉐이더가 하는 일은 무엇일까요? 이미 위에서 살펴본 것 같지만 그래도 다시 한번 반복해 드리지요.

픽셀쉐이더의 주된 임무는 화면에 출력할 최종색상을 계산하는 것이다. 이제 정점쉐이더와 픽셀쉐이더의 임무를 합치면 아까 제 맘대로 내렸던 쉐이더의 정의가 나오죠?

쉐이더란 화면에 존재하는 각 픽셀의 위치와 색상을 계산하는 함수이다. 솔직히 이 정도 말씀을 드려도 쉐이더를 처음 접하시는 분들은 아직도 감이 안 잡히실 겁니다. 사실 쉐이더를 짜 보지 않으면 이해가 어렵습니다. 3D 물체를 화면에

그릴 때, 그 물체를 구성하는 픽셀들의 위치와 색을 프로그래머 맘대로 조작하는 거라고 하면 이해가 좀 더 되실까요? 아직도 이해가 안되시더라도 크게 걱정은 마세요 이 책을 읽으시다 보면 '아 이런 거였구나 '하고 갑자기 이해가 되실 겁니다. M

1.2 쉐이더 프로그래밍

샘플파일 받기

자, 그럼 쉐이더가 무엇인지는 대충 알아보았는데 쉐이더를 코드를 짠다는 것은 무슨 뜻일까요? 일단 그림 1.1을 다시 한번 살펴보죠. 그림 1.1을 보면 사각형으로 표현한 파이프라인 단계도 있고 둥글게 표현한 것도 있죠? 원형으로 표현한 단계들은 **GPU(graphics processing unit**, 그래픽 처리장치)가 알아서 처리해주는 – 즉 프로그래머가 따로 제어할 수 없는 – 단계들입니다. 그와 반대로 사각형으로 표현한 단계들은 프로그래머가 마음대로 제어할 수 있는 단계들이죠. 이 단계에서 사용할 함수를 작성하는 것이 바로 쉐이더 프로그래밍입니다. 그림 1.1에서 사각형으로 표현된 단계들은 정점쉐이더와 픽셀쉐이더 뿐인 거 보이시죠? 따라서 정점쉐이더와 픽셀쉐이더에 사용할 함수를 하나씩 만드는 것이 쉐이더 프로그래밍입니다. (DirectX 10과 11에서 새로운 쉐이더들이 추가되었습니다. 하지만 아직 실무에서 널리 사용되지 않아서 실용적인 접근이 어렵고, 입문자에게 적당하지 않은 내용이라 이 책에서 다루지 않습니다.)

시중에 나와 있는 여러 쉐이더 언어 중에 이 책에서 사용할 언어는 DirectX에서 지원하는 HLSL입니다. **HLSL(High Level Shader Language**, 고수준 쉐이더언어)은 C와 매우 비슷한 문법을 사용하는 언어로 **GLSL**(OpenGL Shader Language의 약자로 OpenGL에서 지원하는 쉐이더 언어입니다. HLSL과 문법 정도가 조금 다릅니다.)이나 **CgFX**(엔비디아에서 지원하는 쉐이더 언어입니다. HLSL과 한두 개 빼고는 완전히 똑같습니다.)등의 기타 쉐이더 언어와 매우 흡사합니다. 따라서 HLSL을 배우시면 다른 쉐이더 언어를 익히시는데도 큰 무리가 없을 것입니다.

한 언어를 배우는 최선의 방법은 직접 코딩을 하면서 배우는 것입니다. 이 언어의 철학은 이러네, 이 언어의 문법은 저러네 하면서 백날 떠들어봐야 입문자들은 하품만 하고 무슨 이야긴지 알아듣지도 못합니다. 일단 재미있게 코드를 짜봐야 프로그래밍에 애착도 생기고, 애착이 생기면 보다 나은 프로그래머가 되기 위해 노력을 하지요. 따라서 이 책에서는 쓸데없이 HLSL 문법을 나열하면서 독자분들의 짜증을 부추기는 대신 무조건 아주 쉬운 쉐이더부터 짜보는 방법으로 HLSL을 배우도록 하겠습니다. 정 문법이 궁금하신 분들은 부록을 참고하시길 바랍니다.

하지만 HLSL 코드를 곧바로 짜기 전에 준비해야 할 것들이 좀 있군요. 이건 좀 지루하시더라도 꾹 참고 따라 해주시기 바랍니다.

쉐이더 프로그래밍을 위한 기본준비

서문에서도 말씀드렸듯이 이 책의 초점은 쉐이더 프로그래밍입니다. 이 책에서 DirectX에 대한 내용을 자세히 다루지 않기로 결정한 이유는 이미 훌륭한 DirectX 입문 책들이 시중에 나와있는데 굳이 DirectX를 다시 처음부터 소개하면서 쓸데없이 지면을 낭비하고 싶지 않았기 때문입니다. (지면이 늘어나면 쓸데없이 책 값도 오릅니다.) 또한 프로그래머 분들 외에 테크니컬 아티스트 분들도 이 책을 읽으실 수 있도록 하기 위해서입니다.

마찬가지 이유로 이 책에서 쉐이더를 만드는 과정도 둘로 나눴습니다. 첫 번째 단계는 쉐이더 작성만을 하는 단계로 AMD(전 ATI) 사의 렌더몽키(render monkey)라는 프로그램을 사용합니다. 이 단계는 프로그래머와 아티스트 분들을 모두 대상으로 하므로 반드시 따라 해 주시기 바랍니다.

두 번째 단계는 렌더몽키에서 만든 쉐이더를 C++/DirectX 프레임워크에서 불러와 사용하는 것으로 프로그래머 분들을 위한 단계입니다. 프로그래머이시더라도 C++/DirectX 프레임워크에 관심이 없으신 분들은 이 단계를 건너 뛰셔도 됩니다. 실제로 쉐이더 코드를 작성하는 곳은 첫 번째 단계입니다.

자, 그러면 위 두 단계에서 쉐이더를 배우는 데 필요한 것들을 준비해보죠.

렌더몽키 렌더몽키는 AMD사에서 제공하는 쉐이더 작성도구로 프로토타이핑에 유용합니다. 부록 디스크에서 /RenderMonkey/ RenderMonkey.2008-12-17-v1.82.322.msi를 찾아 설치해 주세요. 그냥 기본(default) 옵션으로 설치하시면 되겠습니다.

선택사항: 간단한 DirectX 프레임워크 C++/DirectX 프레임워크에서 쉐이더를 실행해보고 싶으신 분들만 이 절을 따라 해주세요.

우선 비주얼 C++ 2008과 DirectX SDK를 설치하시기 바랍니다. 비주얼 C++을 소장하고 계시지 않으신 분들은 마이크로소프트사의 웹 페이지에서 공짜 버전인 익스프레스 버전을 다운받으실 수 있습니다. DirectX SDK는 부록 CD의 DXSDK 폴더에 포함되어 있습니다.

위 두 프로그램의 설치를 마치셨다면 비주얼 C++ 2008에서 부록 CD에 있는 $samples/01_DxFramework/BasicFramework.sln$ 파일을 여시기 바랍니다. 별다른 수정 없이 이 프로그램을 실행하면 다음과 같은 파란 화면을 보실 수 있을 것입니다.

그림 1.2. 별볼일 없는 초 간단 프레임워크

이 프레임워크는 다음과 같은 기능들을 구현합니다.

- 창의 생성 및 메시지 루프 등의 기본적인 윈도우 기능

- Direct 3D 장치 생성
- 텍스처, 모델, 쉐이더 등의 자원 로딩
- 간단한 게임루프
- 간단한 키보드 입력처리

참고로 말씀드리는데 이 프로그램은 쉐이더 코드를 재빨리 실행할 수 있도록 매우 간단하게 만든 프레임워크입니다. 그 결과, 모든 함수들이 .cpp 파일 하나 안에 들어있고, 클래스나 개체도 사용하지 않지요. 따라서 모든 함수들은 C스타일로 작성되어 있고, 모든 변수들도 전역적으로 선언되어 있습니다. **실제 게임을 만드실 때, 이렇게 프레임워크를 만드시면 절대 안됩니다.** 다시 한 번 말씀드리는데 이 프레임워크는 쉐이더 데모를 실행할 수 있도록 만든 프로그램일 뿐입니다.

자, 그럼 적당히 주의도 드렸으니 이제 프레임워크를 살펴보도록 합시다. 우선 BasicFramework.h를 엽니다.

```
1  //
       **************************************************************************
2  //
3  // ShaderFramework.h
4  //
5  // 쉐이더 데모를 위한 C스타일의 초간단 프레임워크입니다.
6  // (실제 게임을 코딩하실 때는 절대 이렇게 프레임워크를
7  // 작성하시면 안됩니다. --)
8  //
9  // Author: Pope Kim
10 //
11 //
       **************************************************************************
12
13
14 #pragma once
15
16 #include <d3d9.h>
17 #include <d3dx9.h>
18
19 // ---------- 선언 ----------------------------------------
20 #define WIN_WIDTH  800
```

```
21 #define WIN_HEIGHT  600
22
23 // ---------------- 함수 프로토타입 ----------------------------
24
25 // 메시지 처리기 관련
26 LRESULT WINAPI MsgProc( HWND hWnd, UINT msg, WPARAM wParam, LPARAM
       lParam );
27 void ProcessInput(HWND hWnd, WPARAM keyPress);
28
29 // 초기화 관련
30 bool InitEverything(HWND hWnd);
31 bool InitD3D(HWND hWnd);
32 bool LoadAssets();
33 LPD3DXEFFECT LoadShader( const char * filename );
34 LPDIRECT3DTEXTURE9 LoadTexture(const char * filename);
35 LPD3DXMESH LoadModel(const char * filename);
36
37 // 게임루프 관련
38 void PlayDemo();
39 void Update();
40
41 //렌더링 관련
42 void RenderFrame();
43 void RenderScene();
44 void RenderInfo();
45
46 // 뒷정리 관련
47 void Cleanup();
```