

RSL(Renderman Shading Language) 입문자의 셰이더 개발환경 구축 :

학생 / RSL 비 경험자의 개발 환경 구축을 중심으로

박기항 [현, Cg Production Engineer]

1. 이 챕터의 목적

우선 이 챕터를 소개하게 된 배경을 설명하고자 합니다. 이 챕터의 메인 컨셉은 **이하가 되겠습니다.**

RSL(Renderman Shading Language) 입문자가 RSL 을 학습하기 위한 **셰이더 개발**환경구축

CG 업계에 근무하고 있는 사람인 경우 이미 회사에 여러 가지 상용 소프트웨어를 사용할 수 있는 환경이 구축되어 있겠지만, 학생 또는 일반유저인 경우 자신이 테스트 환경을 구축하지 않으면 안될 것입니다. 본 챕터는 **그러한 경우 개발 환경작성에 좀더 손쉽게 접근할 수 있는** 하나의 QuickStart **메뉴얼로서 기본정보를** 제공하고자 합니다.

본 챕터에서 다루는 내용은 매우 기본적인 환경 설정을 가능한 상세하게 해설하는 것을 **중심**으로 합니다. 실제 CG 스튜디오에서 환경을 구축하는 경우에는 일반적으로 전혀 다른 내용될 수 있으므로 이점 참고해 주시기 바랍니다.

2. 본 챕터의 구성

본 챕터는 크게 나누어 다음의 3 가지 사항에 대하여 설명합니다.

가. VMWare Player 환경구축

나. 3Delight Pro Trial 환경구축

다. RSL 개발환경 테스트

각 사항에 대한 인스톨 가이드와 인스톨 후 RSL 을 이용한 셰이더 개발 테스트 과정을 안내합니다.

상기의 내용에서 VMWare Player 도입은 이하의 이유에서 선정되었습니다.

가. Linux 환경이 구축 가능 여부

나. 무료 사용 가능 여부

학생 및 일반 유저인 경우 전용 Linux 환경을 구축하기에는 여러가지 제한 사항이 있는

것이 현실이라고 보여집니다. 물론 소유한 PC가 복수인 경우 전용 Linux용 PC를 준비하는 것도 가능하겠지만, 대개의 경우 그러한 여유 있는 환경이 가능한 경우는 별로 없습니다. Dual Boot 환경을 구축하는 것도 가능하지만, Linux OS가 버전업이 필요한 경우에는 여러가지 귀찮은 대응이 필요하게 될 것입니다. Ubuntu를 사용하여 USB Boot 환경도 가능한 선택이지만, 개인 유저인 경우 익숙한 windows를 사용하지 않고 Linux만을 사용하는 환경은 역시 불편합니다. 또한 그렇다고 그때 그때 OS를 재부팅 하는 것도 귀찮은 일이 될 것입니다. 이러한 여러 가지 제한사항을 고려하여 볼 때, VMWare Player는 기존의 windows 환경에 영향 없이 Linux 환경을 테스트 할 수 있는 하나의 좋은 선택이라고 보여집니다. 따라서 본 챕터에서는 VMWare를 이용하여 Linux 환경을 구축합니다.

다음으로 3Delight Pro Trial 버전이 선정된 이유를 설명하고자 합니다. 본 챕터에서는 RSL을 사용하기 위한 환경을 구축합니다. Renderman 사양을 따르는 소프트웨어가 많이 있겠습니다만, 그 중에서 3Delight가 선정된 이유는 이하입니다.

- 가. RSL 사용 가능 여부
- 나. 무료 사용 가능 여부
- 다. 개발 장래성

쉐이더 개발을 경험하고자 할 때, RSL 쉐이더 개발은 개인적으로 보기에 좋은 출발점으로 보여집니다. 컴파일 환경등 개발에 관련된 주변사항을 신경 쓸 필요 없이, 쉐이더의 내부구조를 바로 조작할 수 있기 때문입니다. 또한 RSL을 사용할 수 있는 여러가지 많은 랜더러 중에서도 3Delight는 복잡한 구성을 없애 유저가 사용하기 편한 환경제공을 목표로 하고 있습니다.

--- 참고 ---

본 챕터에서는 사용하지 않는 기능이지만, Maya 상에서 3Delight를 사용하는 경우 Hypershade에서 작성한 쉐이더 네트워크를 자동으로 3Delight 용으로 변환시켜주는 기능은 매우 유용하다고 생각됩니다. 실제로 이러한 배려 때문에 3Delight를 사용하는 스튜디오도 꽤 있다고 보여집니다.

--- 참고 끝 ---

또한 추가의 중요한 점으로 무료버전의 기능이 상용버전과 별로 차이가 없을 정도로 충실합니다.

본 챕터에서는 상기의 이유로 3Delight를 이용한 기본적인 RSL 쉐이더 개발환경을 구축하는 과정을 제시하도록 합니다.

3. 테스트 환경 / 제한 사항

3-1. 테스트 환경

필자가 테스트에 사용한 환경은 이하입니다.

가. OS : Windows 7

--- 참고 ---

제 경우 일본어 버전으로 테스트 했습니다만, 특별히 언어에 의존하는 내용은 없습니다. 본 챕터에 실린 스크린 캡처의 일부는 언어관계로 VMWare 의 한글윈도우 xp 에서 캡처되었습니다.

--- 참고 끝 ---

나. 3Delight Pro Trial 버전 : 11.0.22

다. VMWare Player 버전 : 6.0.1

3-2. 각 소프트 웨어의 제한사항

본 챕터에서 사용되는 **VMWare Player/3Delight Pro Trial** 에는 각각 이하의 제한사항이 있습니다.

가. VMWare Player

학습 및 개인적인 사용(비영리적 사용)에 관해서는 무료사용이 가능하지만, 영리목적으로 사용할 경우에는 라이선스 구입이 필요합니다.

--- 참고 ---

본 원고를 작성한 시기(2014 년 1 월)에 준하는 내용입니다. 이후에 상황에 따라 변동이 있을 수 도 있겠습니다.

--- 참고 끝 ---

나. 3Delight Pro Trial

3Delight 는 테스트 버전으로 기능상 거의 제한 사항이 없는 완전한 상태의 렌더러를 제공합니다. 따라서 테스트 목적으로 사용하기에는 최적의 상태라고 볼 수 있습니다. **유일한** 제한사항으로서 렌더링 시에 최대 4 코어만이 사용 가능합니다. 때문에 테스트 버전으로는 PC 파워를 최대한으로 사용할 수 없는 경우도 있으므로 퍼포먼스 측정을 목적으로 할 경우에는 이 점 고려할 필요가 있습니다.

--- 참고 ---

본 원고를 작성한 시기의 정보에 준하는 내용입니다. 이후에 상황에 따라 변동이 있을 수

도 있습니다.

--- 참고 끝 ---

--- 참고 ---

테스트 버전의 제한 사항에 관하여는 이하의 페이지를 참조하십시오.

http://www.3delight.com/en/index.php?page=3DSP_download

--- 참고 끝 ---

4. VMWare Player 도입

우선 이후의 3Delight Pro Trial 의 Linux 도입에 사용하기 위한 VMWare Player 환경을 구성하겠습니다.

--- 참고 ---

만약 3Delight 를 windows 에서만 사용할 분은 이 부분을 건너뛰어 직접 다음 챕터, 3Delight Pro Trial 버전 도입부분을 참조하시기 바랍니다.

--- 참고 끝 ---

4-1. VMWare Player 소개

VMWare Player 는 OS 의 내부에 또 하나의 OS 를 가상으로 구축할 수 있게 해주는 프로그램입니다. 최근에 대두되어 있는 가상화(假想化)를 가능하게 해준다는 점에서 매우 흥미 있는 툴이라고 생각됩니다. 실제 업무에서도 여러 가지 사용법이 있겠지만, 일반 개인 유저를 생각해 볼 때, 다음과 같은 경우에 사용용도가 있겠습니다 .

가. 메인 OS 와 다른 OS 를 사용하고 싶은 경우

나. 메인 OS 에 부담을 주는 여러 가지 테스트를 시도하고 싶은 경우

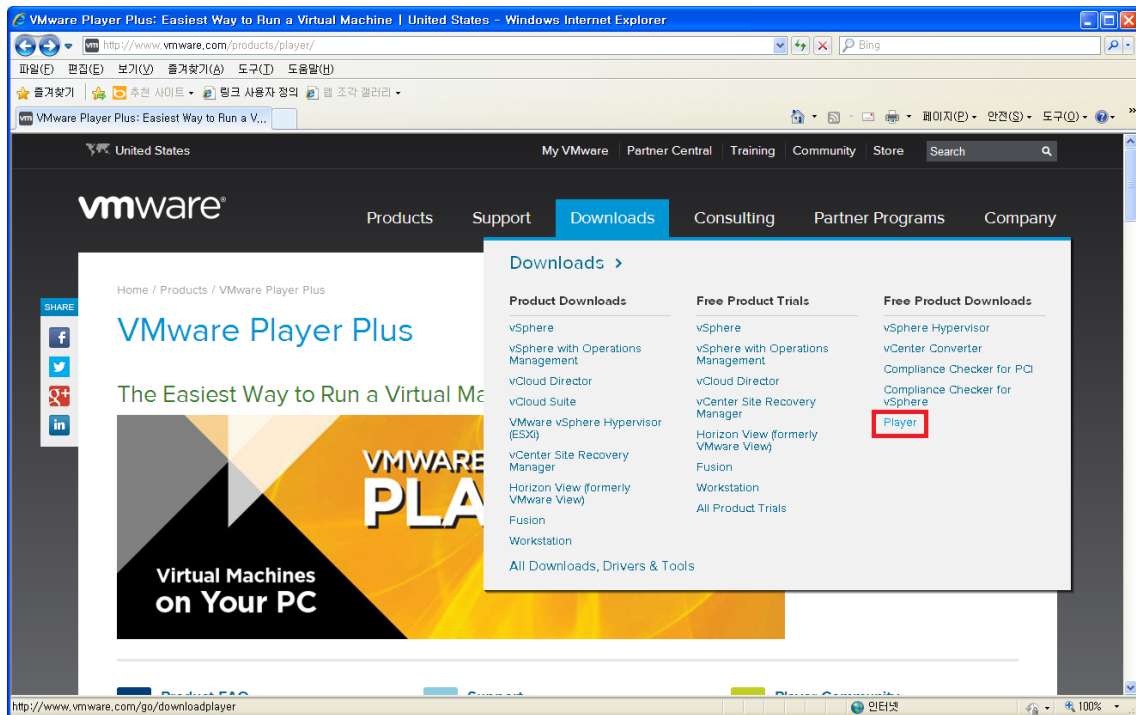
다. 단계적으로 OS 를 백업하면서 테스트를 진행하고 싶은 경우

4-2. VMWare Player / Cent OS 인스톨 가이드

a. 이하의 사이트에 접속하여 Player 부분을 클릭합니다.

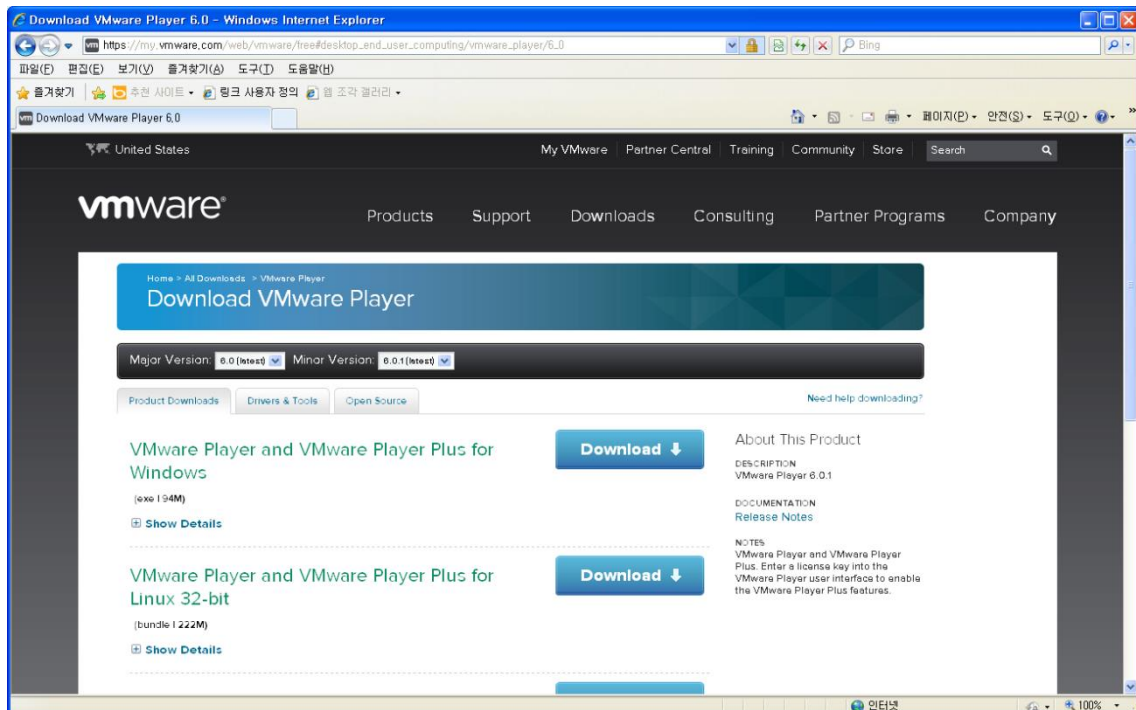
※ 붉은 사각형으로 표시한 부분입니다.

<http://www.vmware.com/products/player/>



[그림 1] VMware Player 사이트 접속 화면

b. 메인 OS 의 종류에 맞는 프로그램을 다운로드 합니다.



[그림 2] VMware Player 사이트 접속 화면

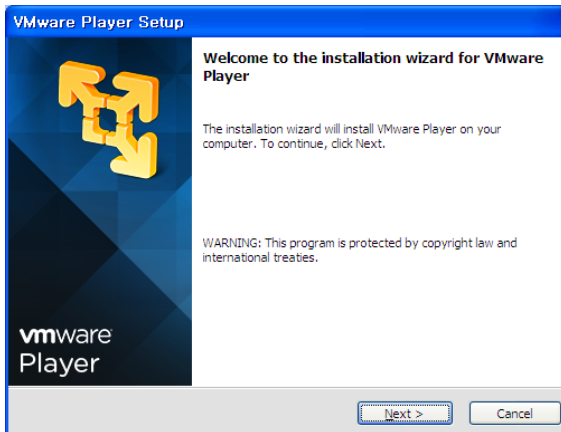
c. 다운로드 한 VMWare Player 프로그램을 실행합니다. 기동된 프로그램을 이하의 과정을

걸쳐 인스톨 합니다.

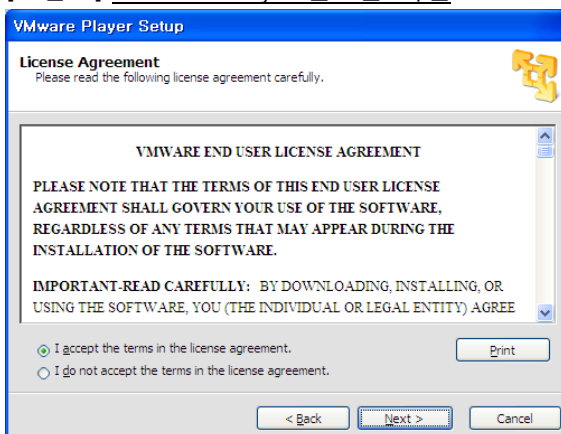
※ 수회에 걸쳐 OK / Next 를 클릭하게 됩니다. 기본적으로 Default 설정을 사용합니다.



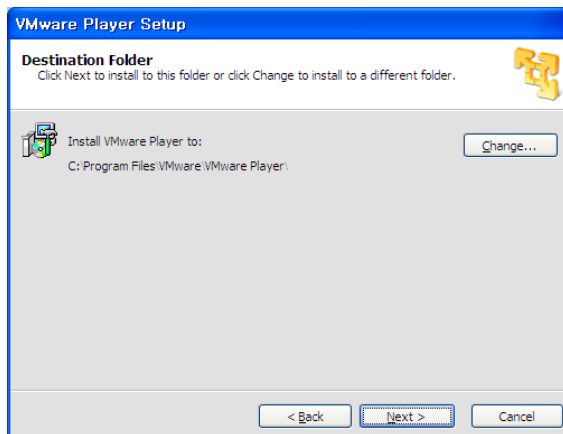
[그림 3] VMWare Player 인스톨 화면



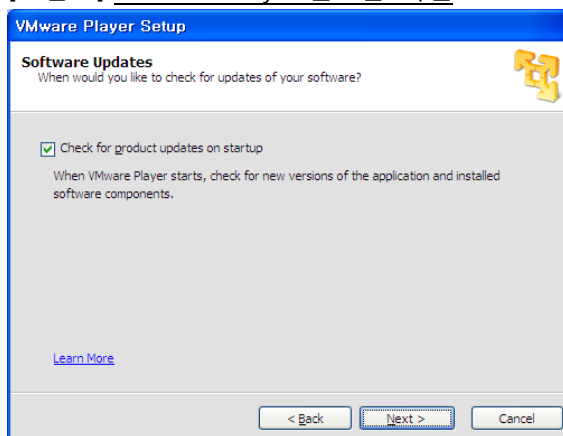
[그림 4] VMWare Player 인스톨 화면



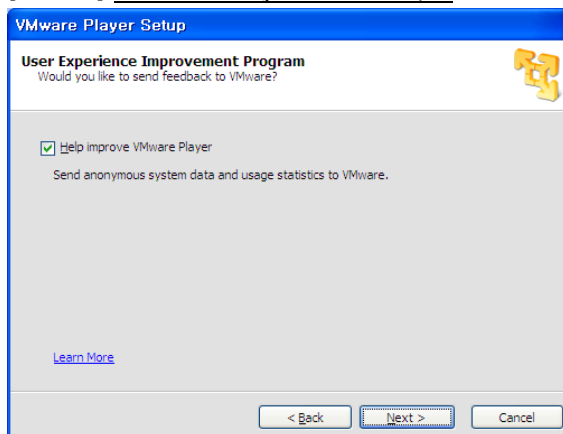
[그림 5] VMWare Player 인스톨 화면



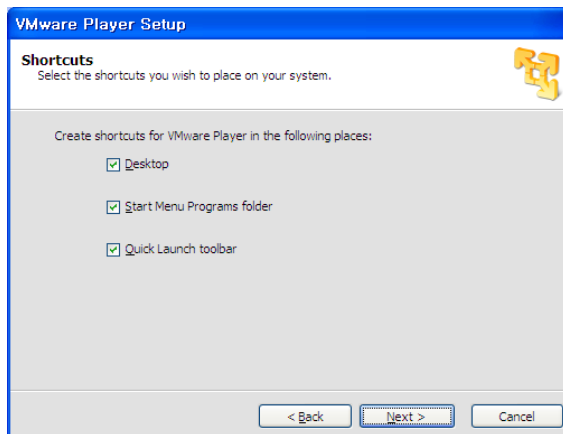
[그림 6] VMWare Player 인스톨 화면



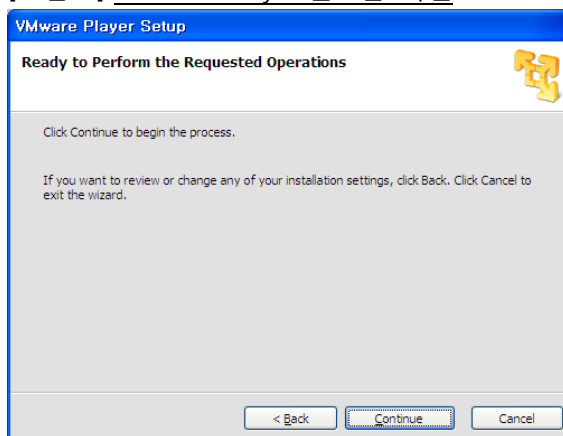
[그림 7] VMWare Player 인스톨 화면



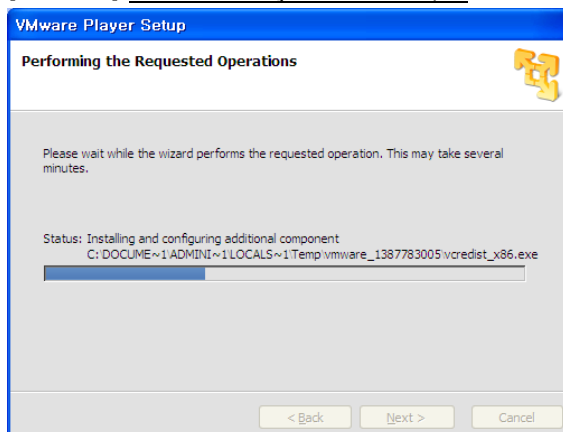
[그림 8] VMWare Player 인스톨 화면



[그림 9] VMWare Player 인스톨 화면



[그림 10] VMWare Player 인스톨 화면



[그림 11] VMWare Player 인스톨 화면

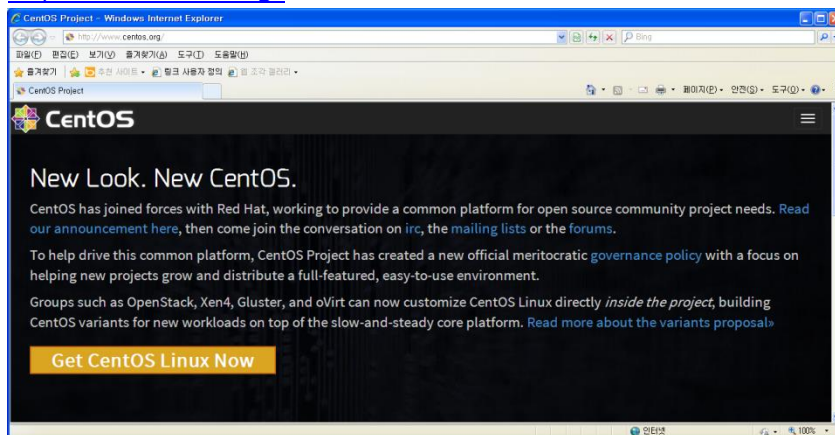


[그림 12] VMWare Player 인스톨 화면

이상의 과정으로 VMWare Player 의 인스톨이 종료되었습니다.

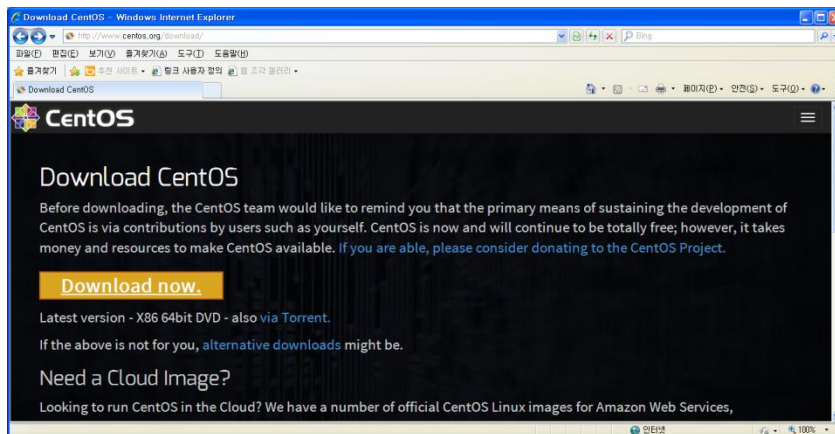
- d. Cent OS 의 이미지를 다운로드하기 위하여 이하의 사이트에 접속, Get CentOS Linux Now 를 클릭 합니다.

<http://www.centos.org/>



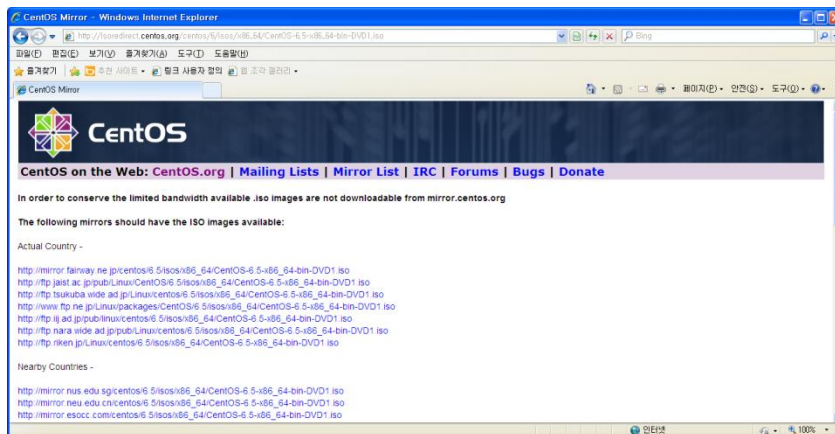
[그림 13] Cent OS 사이트 접속 화면

- e. 이하의 Download Now 를 클릭합니다.



[그림 14] Cent OS 사이트 접속 화면

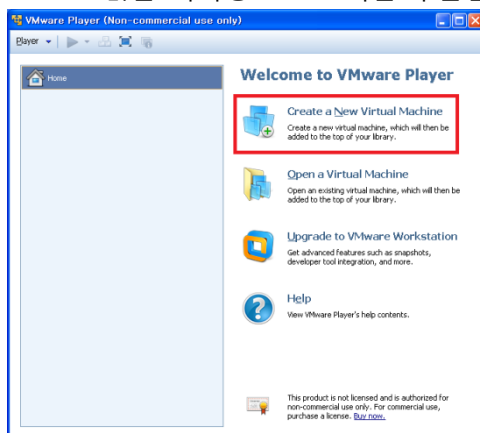
f. 이하의 리스트에서 Cent OS DVD 를 다운로드 합니다.



[그림 15] Cent OS 사이트 접속 화면

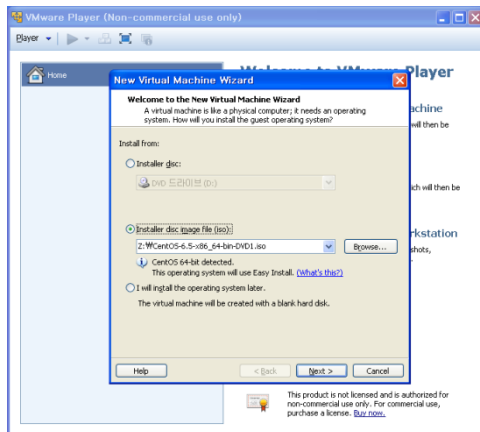
g. VMWare Player 를 실행, Create a New Virtual Machine 을 클릭합니다.

※ 붉은 사각형으로 표시한 부분입니다.



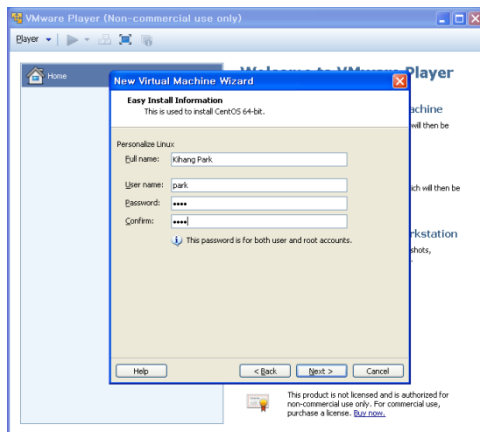
[그림 16] VMWare Player 실행 화면

h. 이하의 화면과 같이 다운로드 한 ISO 파일을 지정합니다.



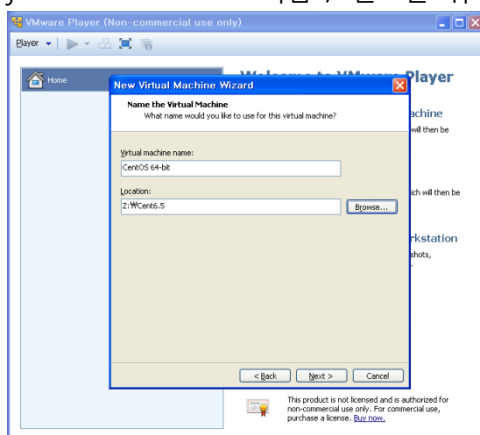
[그림 17] VMWare Player 실행 화면

i. 유저명 / 패스워드를 지정합니다.



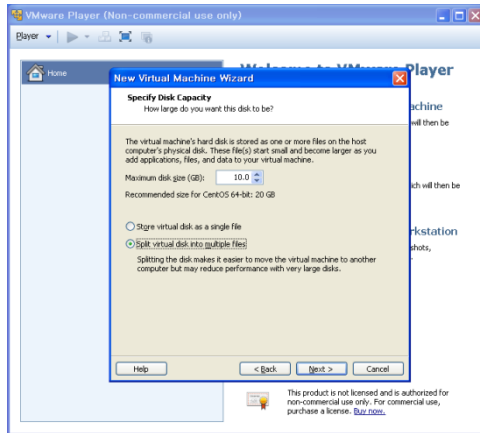
[그림 18] VMWare Player 실행 화면

j. Virtual Machine 이름 / 인스톨 위치를 지정합니다.



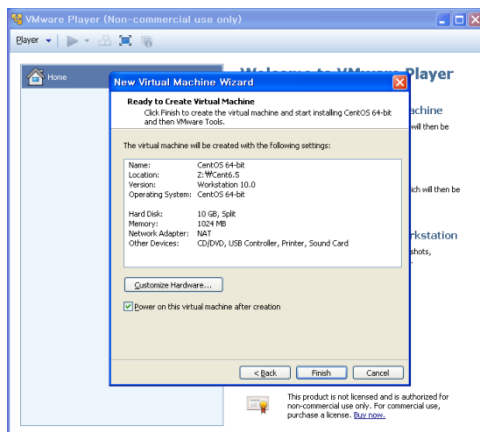
[그림 19] VMWare Player 실행 화면

- k. Virtual Machine 의 사이즈를 지정합니다. Virtual Machine 을 하나의 파일로 작성할지 분할하여 복수파일로 작성할지도 지정하도록 합니다.



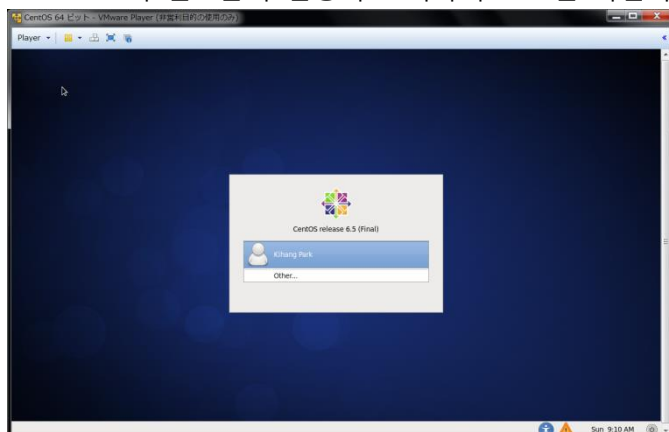
[그림 20] VMWare Player 실행 화면

- l. 이하의 화면을 확인하고 Finish 를 클릭합니다.



[그림 21] VMWare Player 실행 화면

- m. Linux 의 인스톨이 진행되고 이하의 로그인 화면이 나타납니다.



[그림 22] VMWare Player 실행 화면

이상의 작업으로 VMWare Player 에서 Cent OS 를 기동하는 환경을 구축하였습니다.

5. 3Delight Pro Trial 버전 도입

5-1. 3Delight 소개

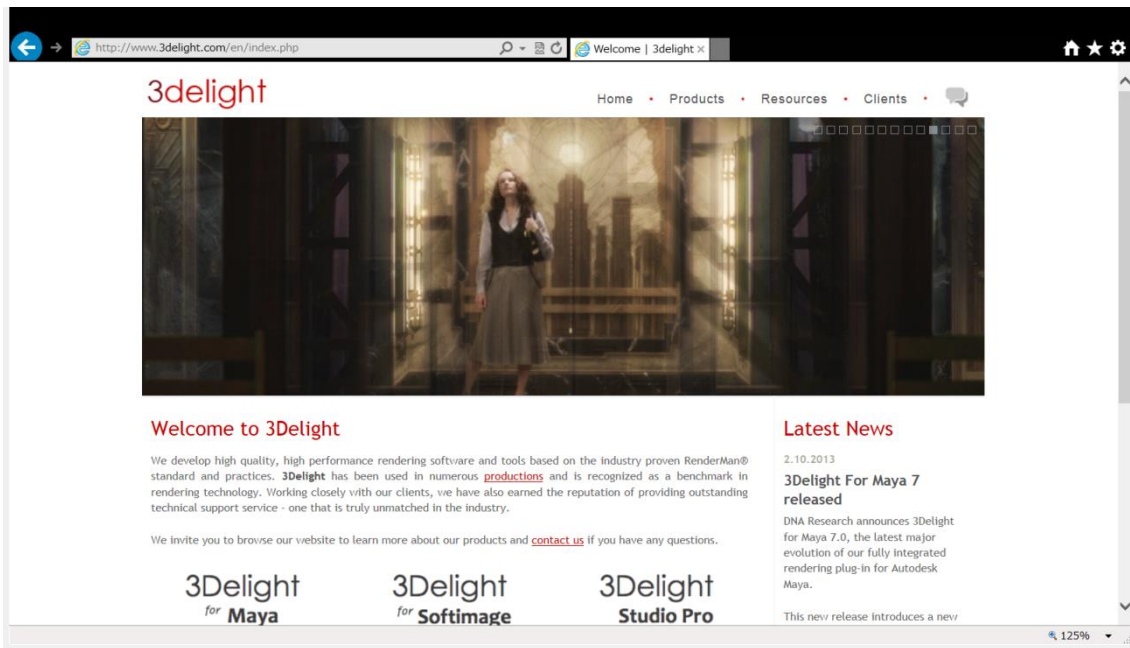
3Delight 에 익숙하지 않은 분들을 위하여 먼저 간단하게 소개하고자 합니다. 3Delight 는 Renderman 의 사양을 따르는 랜더러로, Renderman 과 비교하여 보다 사용하기 쉬운 User-Friendly 랜더러를 목표로 하고 있습니다. Renderman 사양을 따른다는 언급에서 알 수 있듯이 Renderman 에서 사용하는 랜더링 씬 기술사양인 Rib 형식을 랜더링 할 수 있으며, Renderman Shading Language(RSL)을 사용하여 Shader를 작성할 수 있습니다. Renderman 과 같이 랜더링 모드로서 Reyes / PathTrace 모드가 동시에 내장되어 있으므로 랜더링 씬의 상황에 따라 선택적으로 사용할 수 있는 것도 특징입니다. 3Delight for Maya / 3Delight for SoftImage 로 Maya / Softimage 를 지원하며, 3Delight Pro 인 경우에는 양쪽 소프트웨어를 동시에 지원합니다. 독립적인 뷰어 idisplay 를 제공함으로 Maya / SoftImage 가 없더라도 쾌적하게 랜더링 테스트 / 셰이더 개발이 가능한 것이 큰 특징이라고 볼 수 있겠습니다.

5-2. 인스톨 가이드

5-2-1. Windows

a. 이하의 사이트에 접속합니다.

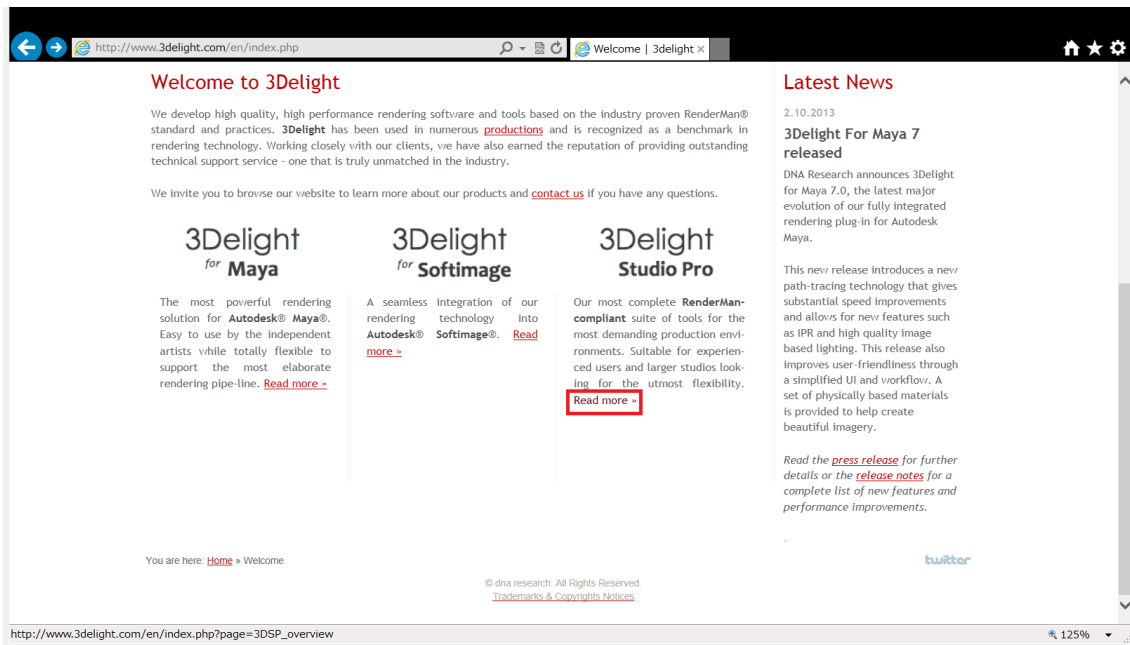
<http://www.3delight.com>



[그림 23] 3Delight 사이트 접속 화면

b. 하단의 3Delight Studio Pro 의 Read more 부분을 클릭합니다.

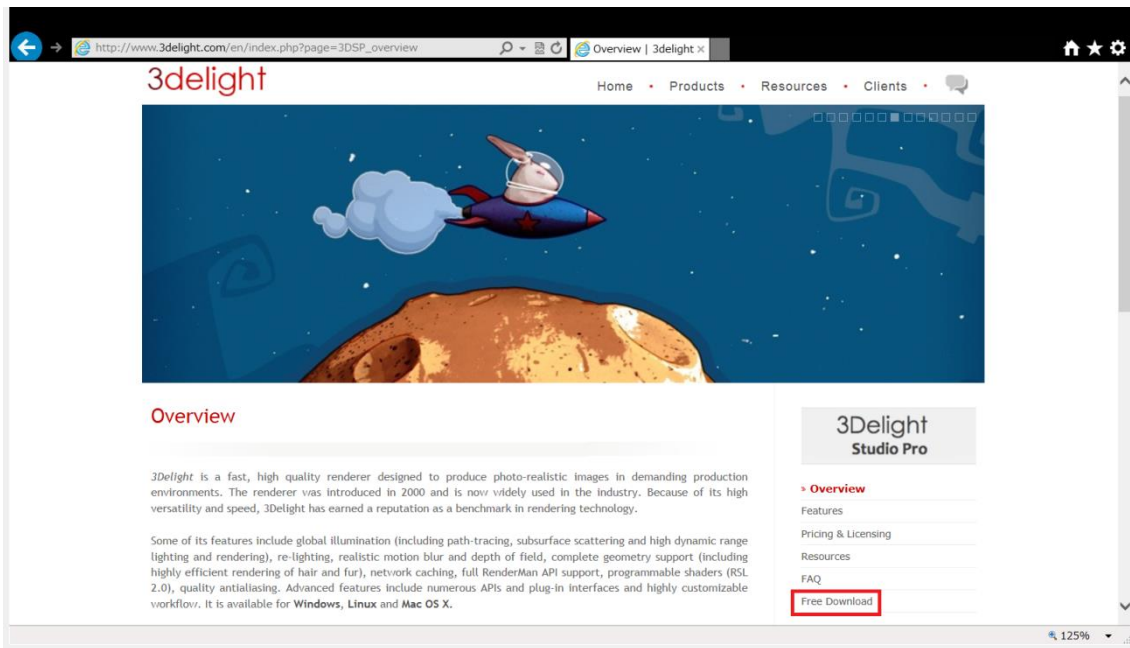
※ 붉은 사각형으로 표시한 부분입니다.



[그림 24] 3Delight 사이트 접속 화면

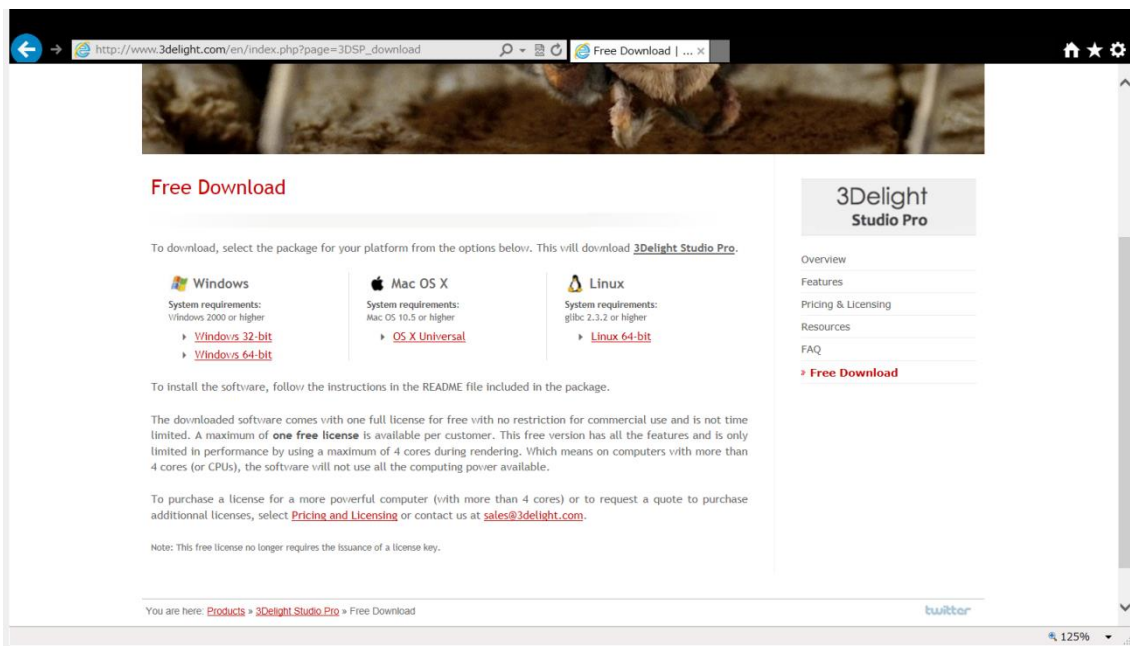
c. 우측 하단의 Free Download 를 클릭합니다.

※ 붉은 사각형으로 표시한 부분입니다.



[그림 25] 3Delight 사이트 접속 화면

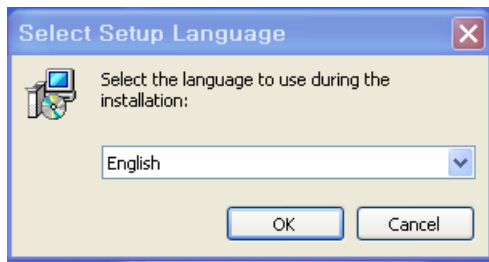
d. 좌측 하단에서 자신의 환경에 해당하는 프로그램을 다운로드 합니다.



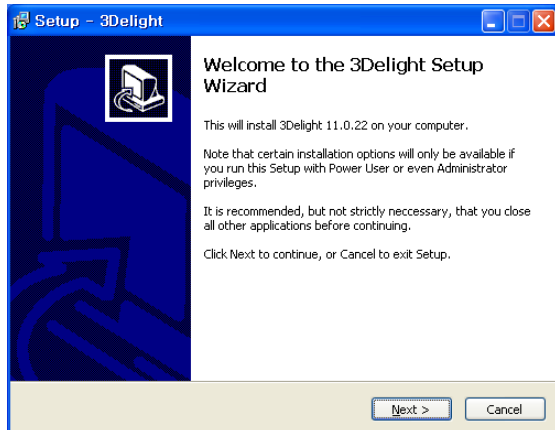
[그림 26] 3Delight 사이트 접속 화면

e. 다운로드 한 프로그램을 실행하여 이하의 과정을 거쳐 인스톨 합니다.

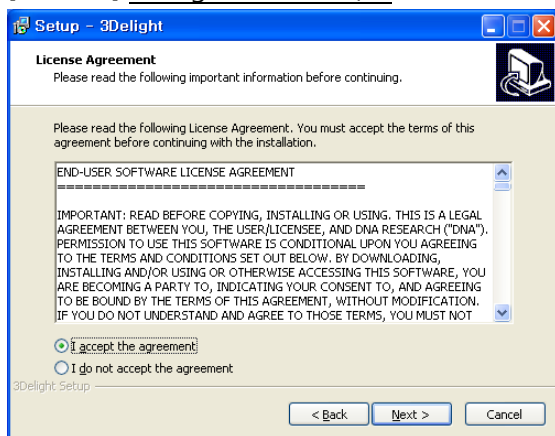
※ 수회에 걸쳐 OK / Next 를 클릭하게 됩니다. 기본적으로 Default 설정을 사용합니다.



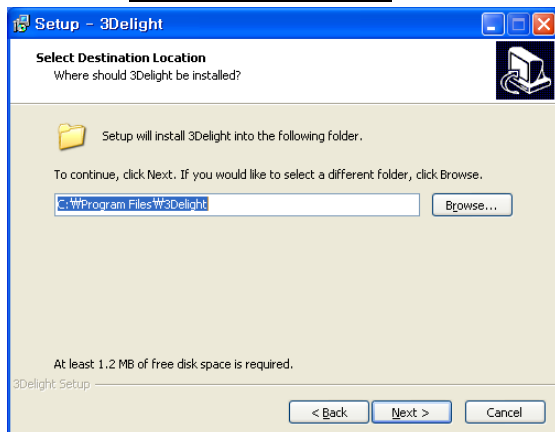
[그림 27] 3Delight 인스톨 화면



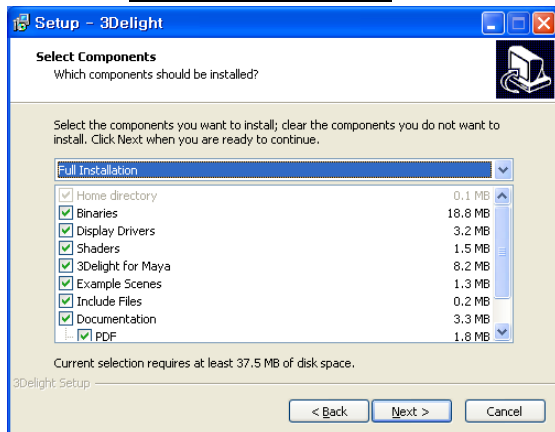
[그림 28] 3Delight 인스톨 화면



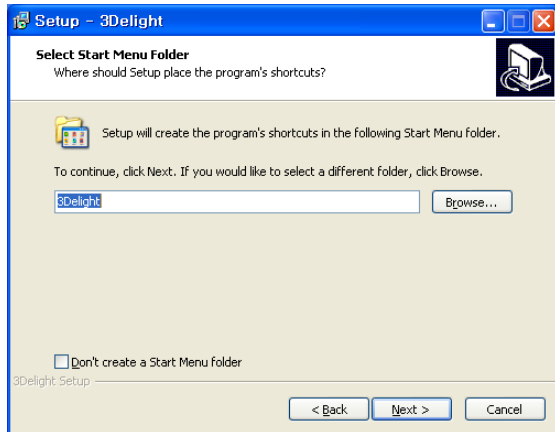
[그림 29] 3Delight 인스톨 화면



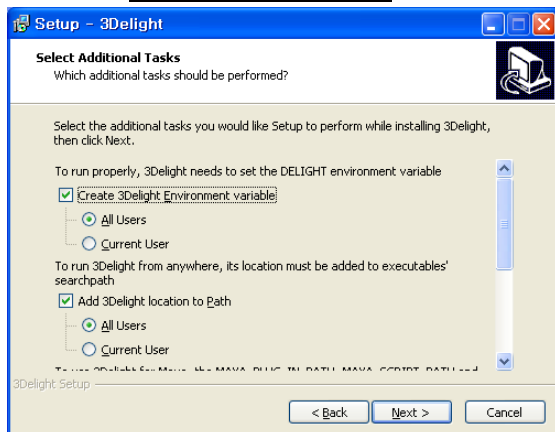
[그림 30] 3Delight 인스톨 화면



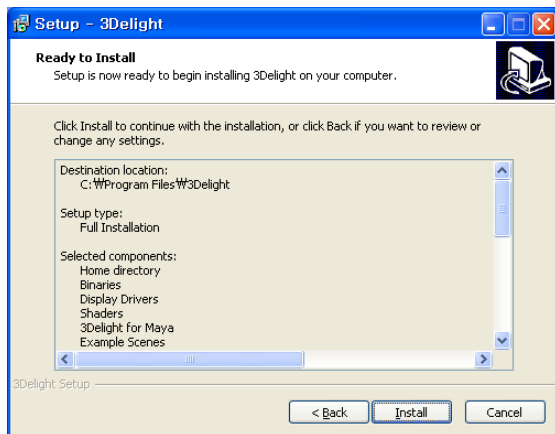
[그림 31] 3Delight 인스톨 화면



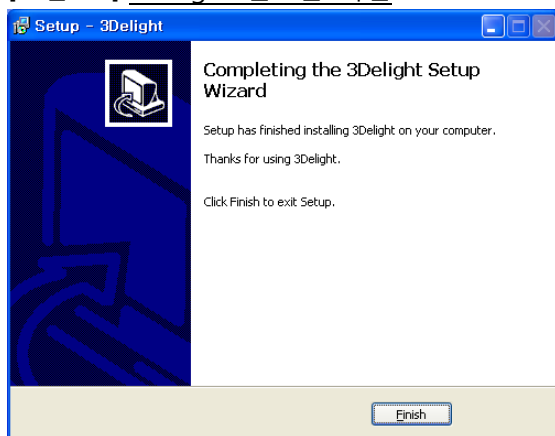
[그림 32] 3Delight 인스톨 화면



[그림 33] 3Delight 인스톨 화면



[그림 34] 3Delight 인스톨 화면



[그림 35] 3Delight 인스톨 화면

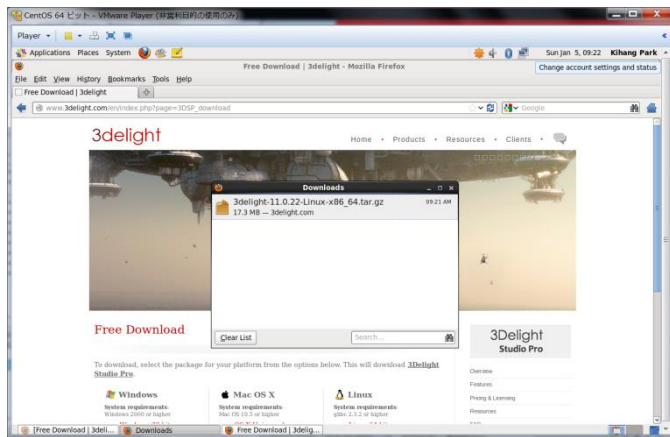
5-2-2. Linux

--- 참고 ---

이하의 과정으로 Linux 에 3Delight 를 인스톨 하도록 하겠습니다. Linux 의 기본 명령어를 사용하므로 만약 각 명령어에 관하여 세부사항을 확인하시고 싶은 경우에는 인터넷에서 Linux 명령어 설명을 추가로 참고 하시기 바랍니다.

--- 참고 끝 ---

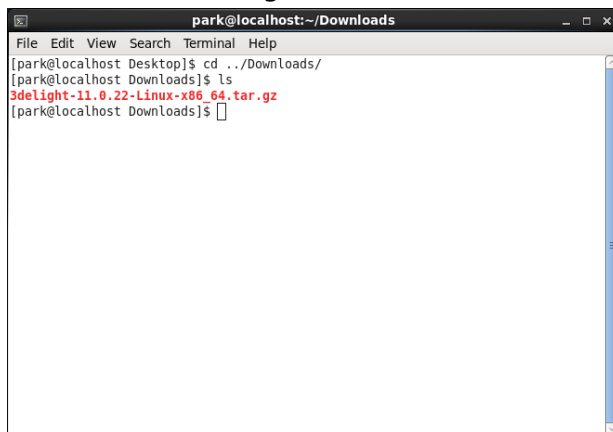
- a. VMWare Player 의 Linux 에 로그인하여 상기의 Windows 와 동일한 요령으로 Linux 용 3Delight 를 다운로드 합니다.



[그림 36] 3Delight 사이트 접속 화면

b. 배경화면에서 오른쪽 클릭 - Open in Terminal 을 선택하여 Console 을 엽니다.

c. 다운로드한 3Delight 파일을 확인합니다.



[그림 37] 다운로드 파일 확인 화면

--- 참고 ---

상황에 따라서 약간 차이가 있을 수 있겠습니다만, 제 경우에 실제 입력한 내용은 이합니다.

```
cd ../Downloads
```

```
ls
```

--- 참고 끝 ---

d. 인스톨을 위하여 루트 권한을 취득합니다.

※ 패스워드는 OS 인스톨 시에 지정한 암호를 입력합니다.



[그림 38] 루트 권한 취득 화면

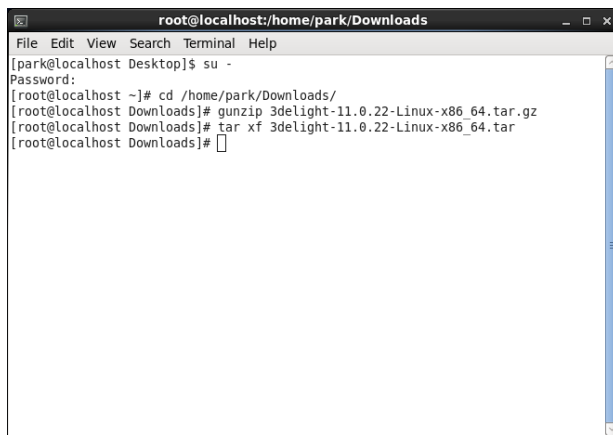
--- 참고 ---

이상의 진행에서 실제 입력한 내용은 이하입니다.

```
su -
```

--- 참고 끝 ---

e. 다운로드 한 압축파일을 전개합니다.



[그림 39] 압축파일 전개 화면

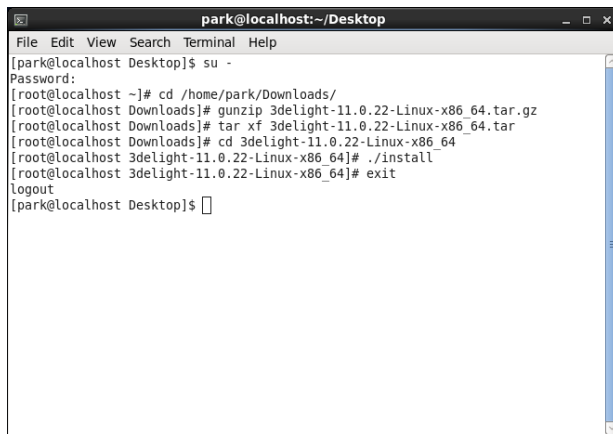
--- 참고 ---

이상의 진행에서 실제 입력한 내용은 이하입니다.

```
cd /home/<user name>/Downloads/  
gunzip 3delight-<version>-Linux-x86_64.tar.gz  
tar xf 3delight-<version>-Linux-x86_64.tar
```

--- 참고 끝 ---

f. 3Delight 인스톨을 위하여, 상기의 작업으로 압축파일을 전개한 디렉토리 위치의 install 파일을 실행하도록 합니다.



```
park@localhost:~/Desktop
File Edit View Search Terminal Help
[park@localhost Desktop]$ su -
Password:
[root@localhost ~]# cd /home/park/Downloads/
[root@localhost Downloads]# gunzip 3delight-11.0.22-Linux-x86_64.tar.gz
[root@localhost Downloads]# tar xf 3delight-11.0.22-Linux-x86_64.tar
[root@localhost Downloads]# cd 3delight-11.0.22-Linux-x86_64
[root@localhost 3delight-11.0.22-Linux-x86_64]# ./install
[root@localhost 3delight-11.0.22-Linux-x86_64]# exit
logout
[park@localhost Desktop]$
```

[그림 40] 3Delight 인스톨 화면

--- 참고 ---

이상의 진행에서 실제 입력한 내용은 이하입니다.

```
cd 3delight-<version>-Linux-x86_64
```

```
./install
```

```
exit
```

--- 참고 끝 ---

5-3. 기본 개발 환경 구축

본 챕터에서의 3Delight 를 이용한 개발 환경은 셰이더 개발을 주 목적으로 상정하고 있습니다. 기본 인스톨 환경을 확인하기 위하여 각 OS 에서 이하의 사항을 테스트 해보도록 하겠습니다.

가. 렌더링 환경 테스트

나. Rib 파일 렌더링 테스트

다. RSL Shader 컴파일 테스트

5-3-1. windows

5-3-1-1. 렌더링환경 테스트

Default 설정으로 인스톨한 경우, 3Delight 프로그램은 이하에 설치됩니다.

C:\Program Files\3Delight

--- 참고 ---

본 챕터 에서 인스톨한 3Delight Pro 의 경우에 해당합니다. 3Delight for Maya 의 경우,

Default 설정인 경우 C:\Program Files\W3Delight For Maya 에 인스톨 됩니다.

--- 참고 끝 ---

그 중에서도 본 챕터 에서 주로 사용하게 되는 프로그램은 이하의 3 개의 프로그램이 되겠습니다.

c:\Program Files\W3Delight\bin\wi-display.exe : 렌더링 이미지 뷰어

c:\Program Files\W3Delight\bin\renderdl.exe : 렌더러

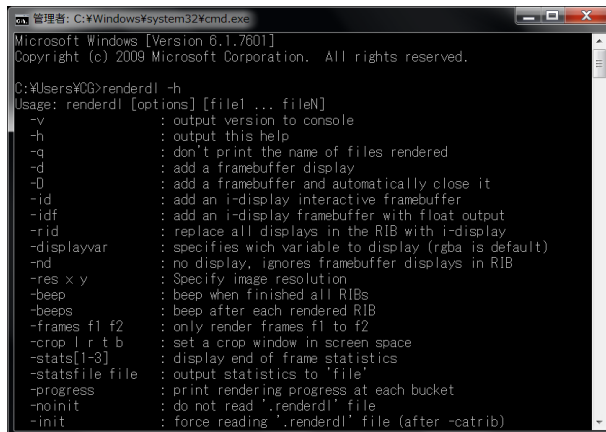
c:\Program Files\W3Delight\bin\shaderdl.exe : 셰이더 컴파일러

이 3 개의 프로그램은 이미 인스톨 시에 환경변수에 패스가 등록되어 있으므로, cmd shell 에서 바로 사용이 가능합니다.

확인을 위하여 cmd shell 에서 이하를 입력합니다.

renderdl -h

renderdl 에서 사용할 수 있는 옵션이 표시됩니다.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

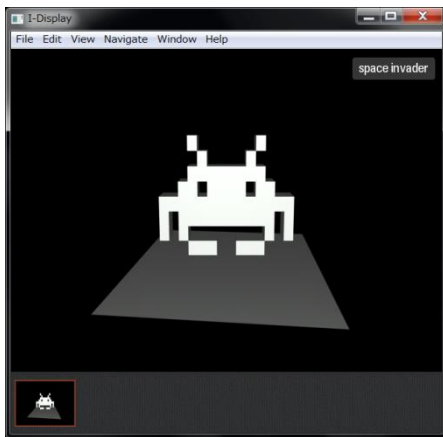
C:\Users\YG>renderdl -h
Usage: renderdl [options] [file1 ... fileN]
-v          : output version to console
-h          : output this help
-q          : don't print the name of files rendered
-d          : add a framebuffer display
-D          : add a framebuffer and automatically close it
-id         : add an i-display interactive framebuffer
-idf        : add an i-display framebuffer with float output
-rid        : replace all displays in the RIB with i-display
-displayvar : specifies wich variable to display (rgba is default)
-nd         : no display, ignores framebuffer displays in RIB
-res x y    : Specify image resolution
-beep       : beep when finished all RIBs
-beeps      : beep after each rendered RIB
-frames f1 f2 : only render frames f1 to f2
-crop l r t b : set a crop window in screen space
-stats[1-3]  : display end of frame statistics
-statsfile file : output statistics to 'file'
-progress    : print rendering progress at each bucket
-noinit     : do not read '.renderdl' file
-init       : force reading '.renderdl' file (after -catrib)
```

[그림 41] renderdl 옵션 출력 화면

renderdl 의 경우, 손쉽게 렌더링 테스트가 가능한 옵션이 이하와 같이 존재합니다.

renderdl -test

실행하여 이하의 이미지가 i-display 에 표시되면 본인의 렌더링 환경이 정상임을 확인 할 수 있습니다.



[그림 42] renderdl 렌더링 테스트

5-3-2. Rib 파일 렌더링

Rib 파일은 Renderman 사양을 따르는 렌더러가 사용가능한 입력파일입니다. Rib 파일은 3D 공간상에 위치하는 카메라, 오브젝트 등 렌더링에 필요한 여러가지 정보를 포함하며, ASCII 포맷으로 되어 있습니다.

정상적인 사용이라면 Maya / SoftImage 등 모델링이 가능한 프로그램을 사용하여 Rib 을 작성 / 렌더링하는 것이 정석이지만, 모델링 가능한 프로그램의 사용은 본 챕터의 범위를 많이 벗어나므로 간단한 Rib 파일을 직접 수동으로 작성하여 렌더링 해보도록 하겠습니다.

--- 참고 ---

Houdini의 경우, 3Delight 사용을 활성화 하는 것으로 Rib 파일이 출력 가능하게 됩니다. 다만 Houdini Apprentice 에서는 Rib 파일 출력이 불가능한 관계로, 본 챕터에서는 수동으로 Rib 파일을 작성하는 방법과 3Delight 에서 제공하는 샘플 Rib 파일을 사용하는 방법으로 진행하도록 하겠습니다.

--- 참고 끝 ---

우선 렌더링 테스트를 위한 Rib 파일을 작성하도록 합니다. 이하의 내용을 텍스트 에디터를 이용하여 sphere.rib 이라는 파일명으로 세이브합니다.

```
Display "test.idisplay" "idisplay" "rgba"
```

```
Format 400 400 1
```

```
WorldBegin
```

```
    Translate 0 0 3
```

```
    Color 0 1 1
```

```
    Sphere 1 -1 1 360
```

WorldEnd

--- 참고 ---

본래 rib 파일은 수동으로 작성하는 것이 아닌 모델링이 가능한 툴을 사용하여 직접 Export 하는 것이 기본입니다. 챕터의 취지를 벗어나지 않는 한에서 각 라인의 의미를 간단히 설명하겠습니다. 만약 Rib 파일의 키워드에 관하여 더 알고 싶으신 부분이 있으시다면 3Delight 의 유저 매뉴얼, 혹은 Renderman Interface Specification(<http://renderman.pixar.com/view/rispec>)을 참고해 주십시오.

Display "test.idisplay" "idisplay" "rgba" :

Display 키워드 이후에 <파일명><출력방법><데이터종류>를 지정함으로써 렌더링 출력방법을 지정합니다.

만약 idisplay가 아닌 sphere.tif라는 이미지 파일로 출력하고 싶은 경우에는 이하와 같이 수정함 으로서 이미지파일로 출력가능합니다.

Display "sphere.tif" "file" "rgba"

Format 400 400 1 :

Format 키워드 이후에 <가로 화소수><세로 화소수><가로 세로 화소수의 비율>을 지정함으로써 출력 이미지를 형태를 지정합니다.

WorldBegin, WorldEnd :

3D 정보내용을 포함하는 블록지정

Translate 0 0 3 :

Translate 키워드 이후에 <x><y><z>를 지정함으로써 이하의 모델을 지정한 좌표로 평행이동

Color 0 1 1 :

Color 키워드 이후에 <r><g>를 지정함으로써 표면 색상을 지정한 성분으로 변경

Sphere 1 -1 1 360 :

Sphere 키워드 이후에 <반경><구의 상부한계점><구의 하부한계점><z 축을 기점으로 한 구의 표시각도>

--- 참고 끝 ---

--- 참고 ---

Houdini 를 사용하여 Rib 을 작성하는 방법이 있겠습니다만, Houdini Apprentice 는 Rib 을 출력하는 것을 지원하지 않으므로 유의하시기 바랍니다. 다른 방법으로, 학생이라면 Maya의 Academic 버전(무료 테스트 버전)을 사용하는 것도 하나의 방법이 되겠습니다.

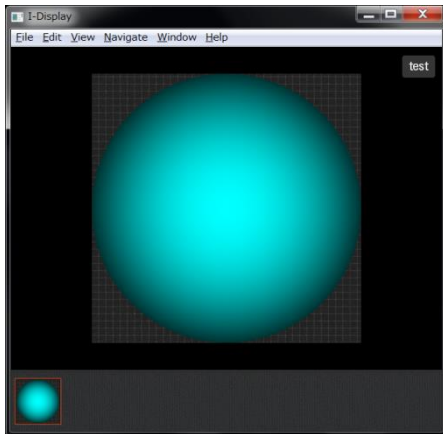
<http://www.autodesk.com/education/student-software>

--- 참고 끝 ---

이 파일을 렌더링하기 위하여 cmd shell 에서 이하를 입력합니다.

```
renderdl sphere.rib
```

이하의 출력이미지가 자동으로 i-display 에 표시됩니다.



[그림 43] renderdl rib 렌더링 테스트

--- 참고 ---

다른 방법으로, **윈도우즈 익스플로러**에서 해당 rib 파일을 **직접** 더블 클릭하는 것으로도 같은 효과를 얻을 수 있습니다.

--- 참고 끝 ---

--- 참고 ---

Renderman 사양을 따르는 렌더러를 사용하는 큰 이유중의 하나는 인터넷에 이미 많은 정보가 오픈 되어 있다는 점이 되겠습니다. Rib 의 파일도 물론 많은 정보가 이미 오픈되어 있습니다.

테스트에 사용한 rib 파일은 가장 간단한 기본적인 사항만을 포함한 rib 파일이 되겠습니다. 반면 보통 실제로 사용되는 rib 파일은 훨씬 복잡하고 많은 사항을 포함하고 있습니다. 이러한 rib 을 테스트 하고 싶은 경우에는 인터넷에서 **좀더 복잡한** 샘플 rib 파일을 습득하여 테스트해 보는 것도 흥미 있는 시도가 되겠습니다.

--- 참고 끝 ---

5-3-3. RSL Shader 컴파일

상기의 작업으로 **본인의 환경이 렌더링** 가능한 상태인 것을 확인할 수 있었습니다. 이제는 Shader 작성이 가능한 개발 환경을 확인하도록 하겠습니다. 상기 5-3-2 에서 작업한 Rib 파일에는 이하의 기술이 있었습니다.

Color 0 1 1

Sphere 1 -1 1 360


Sphere 의 색상을 지정한 수치 0 1 1 는 Default 셰이더, Lambert 의 DiffuseColor 에 적용되어 랜더링 된 경우에 해당합니다. 이번에는 Default 셰이더가 아닌, 직접 작성한 셰이더를 Sphere 에 적용하여 랜더링 해보고자 합니다.

Renderman / 3Delight 모두 셰이더 **작성에 있어서** 같은 언어 RSL 을 사용합니다. 기본적으로 RSL **사양에 맞추어** 셰이더 파일을 **작성, 컴파일하여** Rib 파일에서 **해당 셰이더 파일을 지정함으로써 랜더링에 사용되게** 됩니다. Shader 컴파일이 가능한지를 확인하기 위하여, 가장 **간단한 Shader** 를 작성하여 방금전에 작성한 Sphere 에 적용해보도록 하겠습니다.

우선 셰이더 컴파일을 위하여 사용하는 실행파일 shaderdl 가 이용가능한지 확인하기 위하여 cmd shell 에서 이하를 입력합니다.

shaderdl -h

이하와 같이 shaderdl 에서 사용할 수 있는 옵션이 표시됩니다.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\QG>shaderdl -h
Usage: shaderdl [options] [source_files]
  -o <name>           : Override default output shader name
  -d <name>           : Specifies an output directory for compiled shader(s)

  --strict            : Strict SL syntax
  --int               : Generate ".sdl" file as byte-code (default)
  --embed-source      : Embeds source code in ".sdl" file. Such a file
                        may be used as a source for future recompilation
  --recompile-sdl     : Use this option to update a sdl file to a newer version

  --message-log       : Send warnings and errors to .log file
  --no                : No optimizations
  --o1                : Small optimizations
  --o2 or -o2         : Aggressive optimizations (default)
  --o3                : Very aggressive optimisations
  --w or -w0          : No warnings
  --w1                : Log important warnings only (default)
  --w2                : Log all warnings
  --all-errors        : Log all errors, disabling error and warning filter
  --no-dso            : Do not search into DSO directories for undefined functions
```

[그림 44] shaderdl help 출력결과

문제없이 확인이 가능하면 우선 이하의 내용을 작성하여, red.sl 이라는 이름으로 보존합니다.

```
surface red( ) {
    Ci = color(1, 0, 0);
    Oi = 1;
}
```

--- 참고 ---

본 Shader 의 내용에 대하여도 챕터의 범위를 넘지 않는 한에서 간단한 해설을 하도록

하겠습니다.

```
surface red( ) :
```

```
    surface 키워드로 셰이더 타입을 surface 로 지정합니다.
```

```
    Ci = color(1, 0, 0); :
```

```
    Surface 셰이더의 color 사전 예약어 Ci 에 붉은 색을 직접 지정합니다.
```

```
    Oi = 1; :
```

```
    Surface 셰이더의 opacity 사전 예약어 Oi 에 1 을 직접 지정합니다.
```

--- 참고 끝 ---

이 셰이더 파일 red.sl 은 3Delight 에서 제공하는 shaderdl 실행파일을 사용하여 컴파일 가능합니다. 상기의 보존한 red.sl 파일을 이하의 커맨드 실행으로 컴파일 합니다.

```
shaderdl red.sl
```

실행결과로서 red.sdl 파일이 생성됩니다.

--- 참고 ---

shaderdl 실행파일이 환경변수에 등록되어 있으므로 **윈도우 익스플로러**에서 red.sl 파일을 더블 클릭하는 것으로도 같은 효과를 얻을 수 있습니다.

--- 참고 끝 ---

이제 이렇게 생성된 셰이더를 Sphere 에 적용하기 위하여 상기 과정에서 작성했던 sphere.rib 파일을 이하와 같이 수정합니다.

```
Display "test.idisplay" "idisplay" "rgba"
```

```
Format 800 800 1
```

```
WorldBegin
```

```
    Translate 0 0 2
```

```
    Surface "<red.sdl 이 위치한 패스>/red"
```

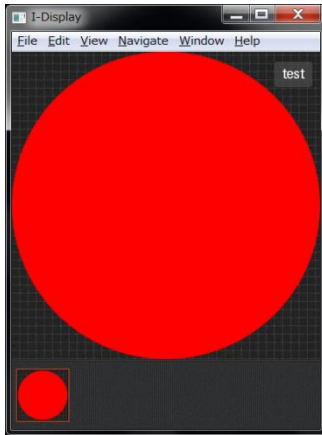
```
    Sphere 1 -1 1 360
```

```
WorldEnd
```

수정된 Rib 파일을 렌더링하기 위하여 다시 한번 cmd shell 에서 이하를 입력합니다.

```
renderdl sphere.rib
```

이하의 출력이미지가 자동으로 i-display 에 표시됩니다.



[그림 45] renderdl rib 렌더링 테스트

문제없이 상기의 렌더링 화면을 확인 할 수 있다면 작성한 셰이더가 정상적으로 적용되었음을 알 수 있습니다.

이상의 작업으로 기본적인 환경을 확인 하였습니다. 물론 여러 가지 **실제 업무 레벨의** 개발을 하기에는 추가적인 설정등이 필요 하겠지만, 기본적으로 셰이더 개발은 이상의 테스트 환경을 베이스로 진행할 수 있습니다.

5-3-2. Linux

상기의 window 부분에서 이미 동일한 내용을 진행하였으므로 상이한 부분을 중심으로 간략하게 설명하도록 하겠습니다. 또한 이번에는 테스트도 기본적으로 3Delight 에서 제공하는 샘플파일을 사용하여 확인하도록 하겠습니다.

개발환경을 확인하기에 앞서, 가장 먼저 해야할 일은 환경변수를 적용하는 것입니다. Windows 의 경우, 인스톨을 함으로서 자동으로 환경변수에 모든 설정이 적용되지만, Linux 의 경우 현재의 기본설정의 조건에서는 수동으로 적용하여야 합니다.

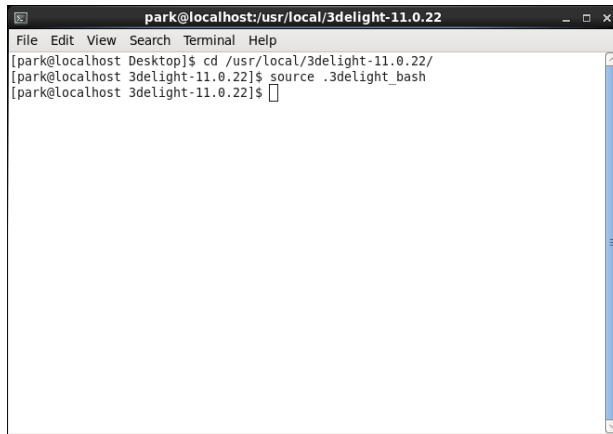
--- 참고 ---

항상 사용하는 프로그램인 경우, 환경설정을 매번 수동으로 적용하는 것은 번거로우므로 로그인 할 때 자동으로 적용되도록 설정하는 경우가 많습니다. Linux 의 환경설정에 관하여 찾아보시면 여러가지 정보를 얻을 수 있으므로 시도해 보시는 것도 좋을 수 있습니다.

--- 참고 끝 ---

5-3-2-1. 환경변수 적용

3Delight 를 인스톨한 디렉토리로 이동하여 환경파일을 source 하도록 합니다. 필요한 환경변수를 한번에 적용시킬 수 있습니다.



[그림 46] 환경파일 source

--- 참고 ---

이동 디렉토리는 Default 설정으로 인스톨 한 경우를 상정하고 있습니다.

필자의 환경에서 실제로 사용한 명령어는 이하와 같습니다.

```
cd /usr/local/<3delight version>/
source .3delight_bash
```

--- 참고 끝 ---

환경파일을 source 하고난 후에는 windows 에서와 동일하게 어느 위치에서나 renderdl, shaderdl 을 사용할 수 있게 됩니다.

--- 참고 ---

환경변수의 적용은 Linux 환경에서 기본적인 사항이 됩니다. 대개의 경우, 이용 프로그램을 기동시키는 기동 스크립트에서 필요한 환경변수를 적용시키는 방법이 이용됩니다.

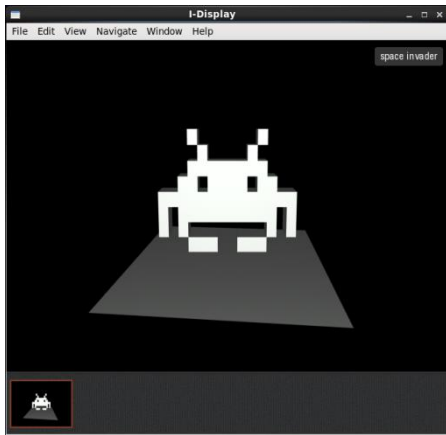
--- 참고 끝 ---

5-3-2-2. 랜더링 환경 테스트

환경파일을 source 한 console 에서 이하의 command 를 실행합니다.

```
renderdl -test
```

이하의 랜더링 화면을 확인 할 수 있다면 성공입니다.



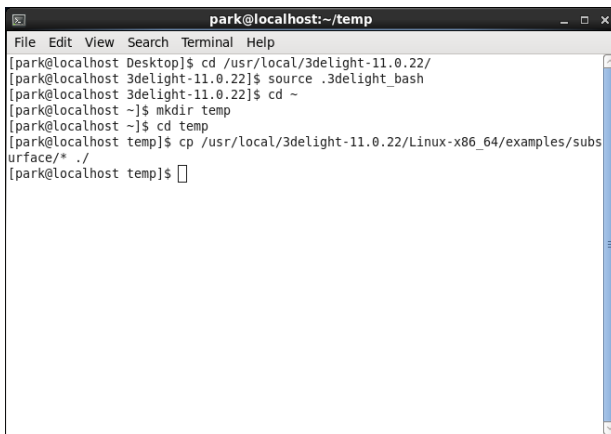
[그림 47] 렌더링 테스트 화면

5-3-2-3. RSL Shader 컴파일 테스트

- a. 원하는 위치에 3Delight 에서 제공하는 subsurface 샘플파일을 복사합니다.

샘플파일은 이하의 위치에 있습니다.

/usr/local/3delight-11.0.22/Linux-x86_64/examples/subsurface/



[그림 48] 3Delight 샘플 복사 화면

--- 참고 ---

지정된 위치는 11.0.22 의 64 비트 버전을 기본 설정으로 인스톨한 경우에 해당합니다. 다른 버전을 다른 위치에 인스톨한 경우에는 해당 위치에서 찾으시기 바랍니다.

필자의 경우에 실제로 입력한 명령어는 이하입니다.

cd ~

mkdir temp

cd temp

```
cp /usr/local/<3delight version>/Linux-x86/examples/subsurface/* ./
```

--- 참고 끝 ---

b. 이하의 실행으로 simple.sl 셰이더를 컴파일 합니다.

```
shaderdl simple.sl
```

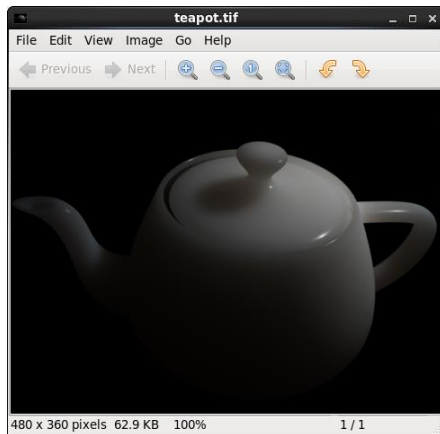
동일한 디렉토리에 simple.sdl 파일이 생성되면 성공입니다.

5-3-2-4. Rib 렌더링 테스트

a. 상기의 과정으로 복사된 teapot.rib 을 이하의 실행으로 렌더링합니다.

```
renderdl teapot.rib
```

동일한 디렉토리에 teapot.tif 가 생성되면 성공입니다. 이하의 실행으로 이미지 파일을 확인할 수 있습니다.



[그림 49] rib 렌더링 테스트 화면

6. RSL 을 이용한 PlasmaShader 개발

이상으로 기본환경을 테스트 해 보았습니다. 이상의 정보로도 본인의 RSL 셰이더를 작성, 컴파일하여 Rib 에서 사용하는 것은 가능합니다. 다만 실제로 간단한 개발과정의 예를 확인하는 것이 더욱 이해에 용이하다고 판단되어, 이 부분에서는 간단한 PlasmaShader 개발과정을 확인해 보도록 하겠습니다.

셰이더 개발과정의 하나의 예로서 게임이나 애니메이션등에서 흔히 볼 수 있는

Plasma 현상을 셰이더로 재현하여 보려고 합니다. 단 몇줄의 코드만으로 유사한 현상을 재현할 수 있기 때문에 흥미유발을 위한 좋은 시도라고 생각되어 이 주제를 선정하였습니다. 물론 실제 업무에서 사용가능한 셰이더인 경우에는 이제부터 작성하는 제한적인 셰이더가 아닌 매우 **여러가지** 부분을 **추가적으로** 고려하여야 하겠지만 여기에서는 가능한 간단하게 셰이더 개발을 체험하는 것을 목표로 합니다.

--- 참고 ---

이 곳에서 사용하는 PlasmaShader 는 이하의 사이트에 실린 내용을 알기 쉽게 일부 간략화한 내용입니다. 원래의 셰이더는 몇가지 이펙트를 더 가지고 있지만 기본 베이스는 **본 챕터에서** 작성하는 내용이 되겠습니다.

<http://www.renderman.org/RMR/Shaders/RudyCShaders/RudyCplasmaball.sl>

--- 참고 끝 ---

6-1. Reference 조사

우선 어떠한 셰이더를 작성할 것인지 자료를 조사해 보겠습니다. Google 의 이미지 검색에서 Plasma 를 검색하여 이하와 같은 결과를 얻었습니다.



[그림 50] plasma 키워드 검색 화면

대강 이미지는 떠오릅니다만, 실제로 어떤 것을 주제로 삼을지는 여러가지를 고려해야 하겠습니다. 금번에는 작성하려고 하는 것이 Surface 셰이더 입니다. 대상 오브젝트의 표면의 질감을 표현하는 셰이더가 되므로 구의 표현에서 안쪽으로 뺏어 나가는 표현은

Surface 셰이더만으로는 힘든 작업이 될 것입니다. 따라서 구의 표면에 빛줄기가 나타나 있는 붉은 사각형으로 표시된 구체와 유사한 셰이더를 작성하도록 하겠습니다.

6-2. 셰이더 제작 준비

일반적으로 마야 등의 모델링 툴을 사용하여 작업을 하지만, 본 챕터에서는 유저가 모델링 툴을 보유하지 않은 것으로 가정하고 있습니다. 따라서 번거롭기는 하지만 이하의 과정을 통하여 셰이더 제작 준비를 하도록 합니다.

6-2-1. Rib 파일 준비

원하는 위치에 3Delight 에서 제공하는 샘플파일을 복사, 준비합니다. 3Delight 에서 제공하는 샘플파일은 이하의 Rib 파일을 사용하도록 합니다.

```
c:\Program Files\3Delight\examples\subsurface\teapot.rib
```

6-2-2. Rib 파일 수정

6-2-1 에서 복사한 teapot.rib 파일은 같은 위치에 동봉된 셰이더 simple.sl 을 사용하도록 지정되어 있습니다. 이제부터 작성할 셰이더를 렌더링 이미지에서 확인하기 위해서는 새로운 셰이더를 사용하도록 Rib 파일을 수정할 필요가 있습니다. 수정할 부분은 이하의 '>>'가 표시된 3 부분에 해당합니다.

```
# an example showing off 3Delight's subsurface scattering features.
```

```
# Aghiles Kheffache. <Tue Jan 13 13:59:33 EST 2004>
```

```
Display "teapot.tif" "file" "rgb"
```

```
>> Display "teapot.idisplay" "idisplay" "rgba"
```

```
--- 참고 ---
```

실행시 idisplay 로 직접 출력하도록 합니다.

```
--- 참고 끝 ---
```

```
PixelSamples 4 4
```

```
...
```

```
Format 480 360 1
```

>> Format 800 600 1

--- 참고 ---

렌더링 해당도를 수정합니다. 확인하기 편하기 위함이므로 본인이 편하신 대로 수정하시면 됩니다.

--- 참고 끝 ---

...

Surface "simple" "float Ks" 0.7 "float roughness" 0.02

>> Surface "plasmaShader"

--- 참고 ---

금번 수정에서의 메인 입니다. simple 셰이더를 사용하여 렌더링하는 원래의 부분을 수정하여 이제부터 작성하는 plasmaShader 를 이용하도록 합니다.

--- 참고 끝 ---

6-2-3. PlasmaShader 의 작성

이제부터 PlasmaShader 를 작성하겠습니다. 각 셰이더의 코드내용에 관하여는 본 챕터의 취지를 벗어나지 않는 한에서 간단한 해설을 참고부분에 기재하였습니다. 다만 이곳에서 설명하는 해설내용은 어디까지나 간단히 테스트하기에 필요한 내용만을 기재하였으므로 사용하는 함수와 셰이더 구조에 관한 좀더 상세한 해설에 관하여는 Renderman, 3Delight 사이트를 참고하시기 바랍니다.

--- 참고 ---

대개의 rsl 함수에 관한 정보를 이하에서 확인 할 수 있습니다.

<http://renderman.pixar.com/resources/current/rps/rslFunctions.html>

--- 참고 끝 ---

a. 단색 셰이더

가장 처음에 작성할 셰이더는 붉은 색의 단색 셰이더 입니다. 이하의 코드를 기입한 파일 plasmaShader.sl 을 작성합니다.

```
surface plasmaShader (
    )
{
    Ci = (1, 0, 0);
```

```

    Oi = Os;
}

```

※ 이후에는 상기의 셰이더 부분에서 메인부분을 중심으로 수정하여 진행합니다.

--- 참고 ---

$C_i = (1, 0, 0)$: output color 를 붉은 색으로 지정합니다.

$O_i = O_s$: alpha 값을 Rib 에서 지정한 값 그대로 사용합니다. 현재 사용중인 Rib 에는 특별히 지정되어 있지 않으므로 기본값 1 이 사용됩니다.

--- 참고 끝 ---

작성한 셰이더를 cmd shell 에서 이하의 실행으로 컴파일합니다.

--- 참고 ---

윈도우에서 개발하는 경우 plasmaShader.sl 의 더블 클릭으로도 컴파일 가능합니다.

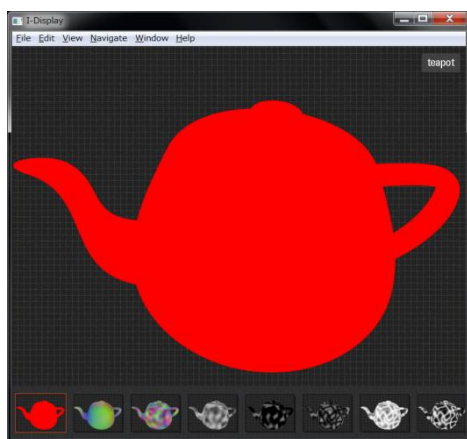
--- 참고 끝 ---

```
shaderdl plasmaShader.sl
```

plasmaShader.sdl 이 생성된 것을 확인한 뒤, cmd shell 에서 이하의 실행으로 렌더링 합니다.

```
renderdl teapot.rib
```

이하와 같이 렌더링 되었다면 문제없이 성공한 것입니다.



[그림 51] rib 렌더링 화면

b. Noise 효과 추가

셰이더 메인 부분, 색상을 지정했던 부분을 노이즈 함수로 변경 합니다.

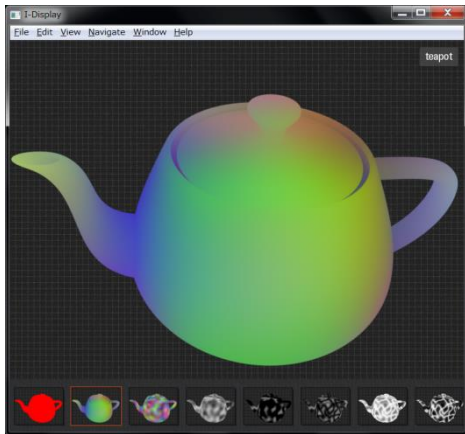
```
{  
    Ci = noise(transform("shader", P));  
    Oi = Os;  
}
```

--- 참고 ---

`Ci = noise(transform("shader", P));` : 예약변수 P (현재 위치)를 Shader 공간으로 변환한 뒤, 그 값을 기본으로 노이즈 함수에 입력, 결과값을 그대로 렌더링 결과로 출력합니다.

--- 참고 끝 ---

이상의 수정을 렌더링 한 결과로 이하의 렌더링 결과를 얻을 수 있다면 성공입니다.



[그림 52] rib 렌더링 화면

c. Frequency 증가

상기의 결과로 얻은 노이즈 이미지는 변화율이 너무 적은 관계로 좀더 결과값의 변화율을 높이기 위하여 노이즈의 Frequency 를 높이도록 하겠습니다.

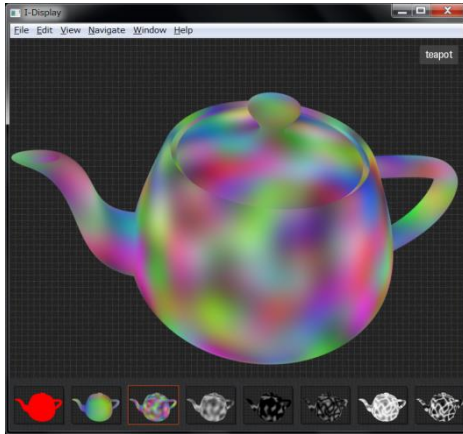
```
{  
    Ci = noise(transform("shader", P * 4.0));  
    Oi = Os;  
}
```

--- 참고 ---

`Ci = noise(transform("shader", P * 4.0));` : 입력치 P 의 변화율을 4 배 증가 시킴으로서 노이즈 결과값의 변화율을 높일 수 있습니다.

--- 참고 끝 ---

이상의 수정의 렌더링 결과는 이하입니다.



[그림 53] rib 렌더링 화면

d. 단색화

최종적으로 사용할 plasma 효과는 단색으로 표현이 되므로 노이즈 효과를 단색으로 변환합니다.

```
{  
    float c = noise(transform("shader", P) * 4.0);  
    Ci = color(c, c, c);  
    Oi = Os;  
}
```

--- 참고 ---

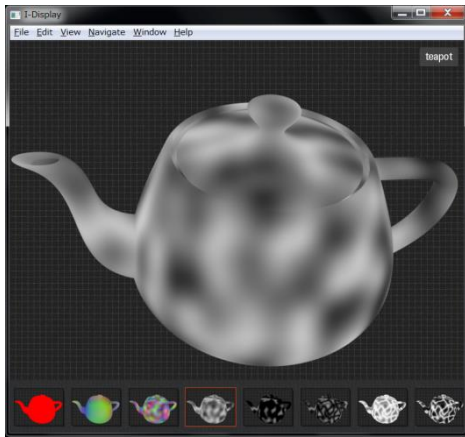
`float c = noise(transform("shader", P) * 4.0);` : noise 함수는 결과값으로 color / float 모두 반환 가능합니다. 상세한 해설은 이하를 참고하십시오.

<http://renderman.pixar.com/resources/current/rps/rslFunctions.html>

`Ci = color(c, c, c);` : r g b 값을 모두 같은 수치로 통일 함으로서 단색으로 변환 합니다.

--- 참고 끝 ---

렌더링 결과는 이하와 같습니다.



[그림 54] rib 렌더링 화면

e. 최대 최소값 조정

현재의 Noise 함수의 결과값으로 0~1 의 값이 반환됩니다. 이 부분의 최대 최소값을 조정합니다. 0~1 의 결과값을 -1~1 으로 조정함으로서 전체적인 밝기를 조정합니다.

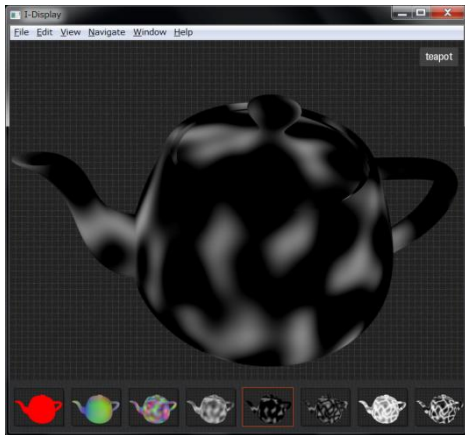
```
{
    float c = noise(transform("shader", P) * 4.0);
    float c2 = c * 2 - 1;
    Ci = color(c2, c2, c2);
    Oi = Os;
}
```

--- 참고 ---

float c2 = c * 2 - 1; : 0~1 이 얻어진 noise 의 결과값을 -1~1 으로 조정합니다.

--- 참고 끝 ---

이하, 렌더링 결과입니다.



[그림 55] rib 렌더링 화면

f. 절대값

어두운 부분, 즉 $-1 \sim 0$ 부분에 대하여 절대값을 취함으로써 0에 가까운 부분이 좁은 라인으로서 나타나도록 조정합니다.

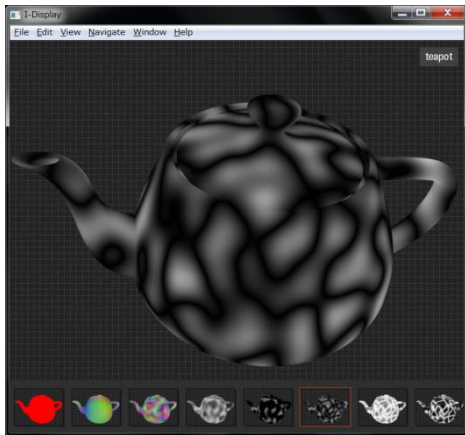
```
{
    float c = noise(transform("shader", P) * 4.0);
    float c2 = abs(c * 2 - 1);
    Ci = color(c2, c2, c2);
    Oi = Os;
}
```

--- 참고 ---

$\text{float c2} = \text{abs}(c * 2 - 1);$: 음수 부분에 대하여 절대값을 취합니다.

--- 참고 끝 ---

렌더링 결과는 이하와 같습니다.



[그림 56] rib 렌더링 화면

g. 반전

상기의 결과는 라인 자체가 검은 색으로 나타나는 상태입니다. 목적하는 결과는 라인 자체가 발광하는 형태이므로 전체 색을 반전 시키겠습니다.

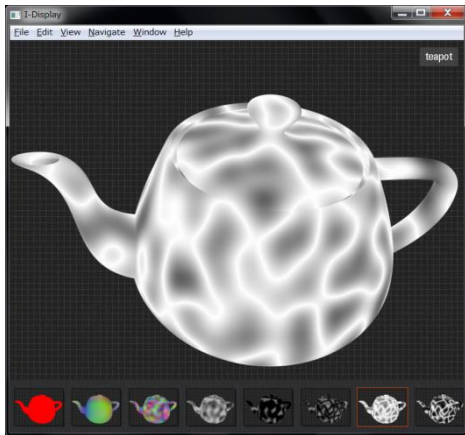
```
{
    float c = noise(transform("shader", P) * 4.0);
    float c2 = 1 - abs(c * 2 - 1);
    Ci = color(c2, c2, c2);
    Oi = Os;
}
```

--- 참고 ---

float c2 = 1 - abs(c * 2 - 1); : 결과값을 1 에서 뺌으로서 색상을 반전 시킵니다.

--- 참고 끝 ---

이하, 렌더링 결과입니다.



[그림 57] rib 렌더링 화면

h. 라인 조임

이로서 발광하는 라인은 작성이 가능하였습디만, 전체적으로 라인이 너무 퍼져있는 느낌이 들고 있습니다. 이하의 수정으로 라인을 좀 더 조여보도록 하겠습니다.

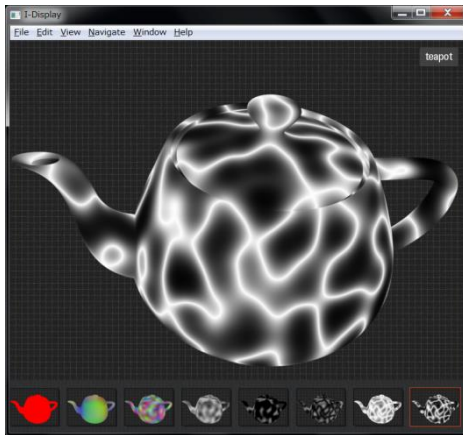
```
{
    float c = noise(transform("shader", P) * 4.0);
    float c2 = 1 - abs(c * 2 - 1);
    float c3 = pow (c2, 4);
    Ci = color(c3, c3, c3);
    Oi = Os;
}
```

--- 참고 ---

`float c3 = pow (c2, 4);` : 결과 값을 4 승함으로서 1 이하의 결과값을 더욱 큰 폭으로 줄이도록 합니다.

--- 참고 끝 ---

렌더링 결과값은 이하와 같습니다.



[그림 58] rib 렌더링 화면

i. 기본 색 추가

라인 이외의 색이 현재 검은 색으로 되어 있으므로 원하는 색상으로 변환합니다. 여러 가지 방법이 있겠지만, 여기에서는 파란 색으로 표시하기 위하여 일정 수치 이하의 r, g 값을 잘라내는 방법으로 수정하였습니다. 또한 Alpha 결과값 O_i 에 대하여 r, g, b 결과값을 적용함으로써 투명도를 부여하였습니다.

```
{
    float c = noise(transform("shader", P) * 4.0);
    float c2 = 1 - abs(c * 2 - 1);
    float c3 = pow(c2, 4);
    Ci = color(smoothstep(0.3, 1, c3), smoothstep(0.3, 1, c3), c3 + 0.3);
    Oi = Os * c3;
}
```

--- 참고 ---

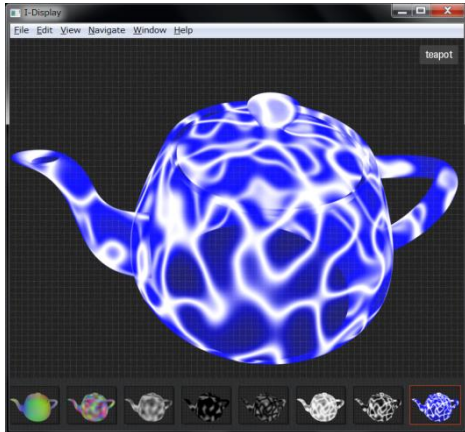
$C_i = \text{color}(\text{smoothstep}(0.3, 1, c3), \text{smoothstep}(0.3, 1, c3), c3 + 0.3)$; : r, g 값에 대하여 smoothstep 함수를 이용하는 방법으로 0.3 이하 값을 0 으로 잘라냅니다. 반면 b 값에 대하여는 무조건 0.3 을 더 함으로서 모든 부분에서 b 값은 0.3 이상을 가지도록 조정합니다. Smoothstep 에 관하여 좀더 상세한 설명을 원하시면 이하의 페이지를 참고 하십시오.

<http://renderman.pixar.com/resources/current/rps/rslFunctions.html>

$O_i = O_s * c3$; : 플라스마 효과 이외의 부분에 대하여 Alpha 값을 낮추는 방법으로 투명한 효과를 부여합니다.

--- 참고 끝 ---

렌더링 결과는 이하가 됩니다.



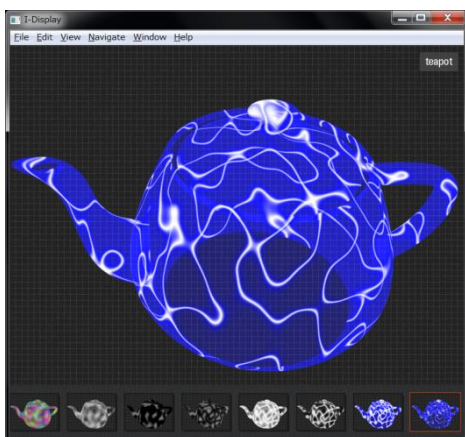
[그림 59] rib 렌더링 화면

j. 라인 추가 조임

이미 원하는 렌더링이 되었을 수 도 있습니다만, 원래 목적하던 이미지와 비교하여 라인이 아직도 퍼져 있는 경향이 있으므로 좀더 조이도록 하겠습니다.

```
{
    float c = noise(transform("shader", P) * 4.0);
    float c2 = 1 - abs(c * 2 - 1);
    float c3 = pow (c2, 20);
    Ci = color(smoothstep(0.3, 1, c3), smoothstep(0.3, 1, c3), c3 + 0.3);
    Oi = Os * c3;
}
```

이하가 최종 렌더링 결과가 되겠습니다.



[그림 60] rib 렌더링 화면

이상으로 plasmaShader 의 작성을 끝마치도록 하겠습니다. 물론 이것이 실제로 사용되기 위한 최종상태로서는 많이 **부족합니다**. 우선 현재 상태로서는 셰이더를 패러미터로 조정하는 것이 불가능합니다. Rib 에서 입력한 패러미터를 받아들여서 변수로서 처리하는 부분이 필요합니다. 다음으로 플라즈마의 라인이 흰색으로 발광하는 형태로 고정되어 있습니다. 발광라인에 색을 부여하는 것도 필요할 수 있습니다. 또한 프레넬 효과와 같이 오브젝트의 외곽부분에 관하여 발광효과를 증가시키는 부분이 필요한 경우도 있을 수 있습니다. 이러한 여러 가지 추가요소에 관하여 **본인이 이것저것 생각해 시도해** 보는 것도 흥미로운 시도가 될 수 있습니다.

7. 3Delight 와 Houdini 의 관계에 대하여

본 원고는 셰이더 개발 입문자에게 3Delight 라는 툴을 활용한 RSL 개발환경을 구축하는 것을 기본 목적으로 하고 있습니다만, 추가적으로 3Delight 와 Houdini 와의 관계에 관하여 간략하게나마 정리하고자 합니다.

3Delight 와 Houdini 의 관계를 알아 보기 이전에 우선 CG 스튜디오 Pipeline 에서의 Houdini 의 위치에 관해 일반적인 사항을 확인해 두는 것이 좋을 듯 합니다. 물론 '일반적인 사항'이라고 해도 CG 스튜디오의 Pipeline 은 각 회사 마다 모두 다른 관계로 일률적으로 말할 수 없는 부분이 많은 것이 사실입니다.

--- 참고 ---

이러한 경향은 메이커가 제공하는 그대로의 툴을 Customize 하여 사용하는 사내 개발팀이 있는 대규모 작업 환경의 경우에 더욱 두드러진다고 볼 수 있습니다.

--- 참고 끝 ---

다만 CG 스튜디오에서 Houdini 는 일정분야, 특히 VFX 분야에 특화되어 사용되는 경우가 많이 보여집니다. 이러한 경향은 모델링 / 애니메이션작업의 경우 테크니컬 성향이 강한 Houdini 보다는 좀더 디자이너 친화적인 Maya 와 같은 툴이 주로 사용되기 때문이라고 볼 수 있습니다.

--- 참고 ---

만약 목표로 하는 것이 Houdini 의 파티클 이펙트 최종 렌더링 뿐이라 하더라도, 그러한 이펙트를 생성하기 위해서는 여러가지 요소가 필요하게 됩니다. 예를 들어 캐릭터가 특정 건물을 파괴하는 이펙트를 작업한다고 하면, 실제 이벤트가 발생하게 되는 배경건물과 캐릭터의 지오메트리가 필요하게 됩니다. 물론 캐릭터, 배경 건물을 Houdini 에서 작업하는 것이 불가능한 작업이 되지는 않겠지만, 효율적인 작업이 되기는 어려우므로 디자이너가 직관적으로 조작할 수 있는 Maya 등의 툴을 사용하여 지오메트리를 작업, Houdini 로 데이터를 전달하는 것이 일반적인 경우가 되겠습니다.

최근 대두되고 있는 Alembic 포맷이 활성화 되기 이전에는 이러한 경우 Obj, Fbx 등으로 데이터를 분할시켜 Houdini 로 전달하는 경우가 많았습니다만, Alembic 포맷이 업계의 공통 포맷으로 자리를 잡게 되면서 Alembic 포맷을 사용한 데이터 전달이 메인 수단으로 부각되고 있다고 볼 수 있겠습니다.

--- 참고 끝---

이러한 경우, 상류 작업에서 작업한 데이터를 Houdini 로 어떻게 가지고 갈 것인가가 자연스럽게 부각되는 부분이 되고, 이것이 각 CG 스튜디오 Pipeline 의 특색으로 나타나게 되겠습니다. 하지만 이러한 각기 다른 Pipeline 의 공통부분으로서, Houdini 는 Pipeline 전체를 관통하는 경우보다는 상류작업을 담당하는 다른 톨과 데이터를 연계하여 사용되는 것이 일반적인 사용법이 되며, 이 부분이 각 CG 스튜디오 Houdini Pipeline 에서 항상 고민하는 부분이 되겠습니다.

--- 참고 ---

이러한 이유에서 Houdini 사례 세미나의 질의 / 응답 코너에서는 항상 '기타 톨과의 데이터 연동은 어떻게 했습니까?' 라는 질문이 나옵니다.

--- 참고 끝 ---

이러한 Pipeline 상에서의 Houdini 의 특수한 위치를 고려해 볼 때, Houdini 를 이용한 렌더링의 특수한 상황이 부각되게 됩니다. 단순히 생각하여 볼 때, Houdini 가 기본적으로 제공하는 Mantra 를 사용하는 것이 가장 타당하다고 생각될 수 있지만, 그 이전에 우선 고려해야 하는 것이 '과연 Houdini 에서 최종 렌더링 이미지를 작성할 것인가' 가 되겠습니다.

만약 Houdini 에서 작업한 결과물이 상류작업의 데이터와 강하게 연결되어 있는 경우, 즉 라이팅 환경이라든지 셰이딩환경이 상류작업 데이터와 밀접하게 연관되어 있는 상황에서는 Houdini 에서 해당 환경을 재현한 뒤 렌더링 하는 것보다 작업 데이터를 BGeo / Alembic 등으로 베이킹하여 상류작업 톨로 반환, 렌더링하는 경우가 작업이 편한 경우가 있을 수 있습니다. 반면 파티클 이펙트 등과 같이 Houdini 상에서 작업한 결과가 상류작업 환경과 특별히 연관되어 있지 않은 경우, 작업 데이터를 상류작업 톨로 반환하기 보다는 Houdini 에서 직접 렌더링 후 그 결과를 기타 렌더링 결과물과 합성하는 방식을 택하는 것이 더 효율적인 작업이 될 것 입니다. Houdini 에서 렌더링 결과물을 작성하는 후자의 경우, 사용가능한 메이저 Renderer 는 이하가 있습니다.

--- 참고 ---

실제로 사용가능한 Renderer 는 Mentalray 를 포함하여 추가적으로 더 존재 합니다만, 버전업에 지속적으로 대응하고 있는 메이저급 Renderer 로서는 상기의 Renderer 가 일반적인 리스트가 되겠습니다.

--- 참고 끝 ---

a. Mantra

b. Renderman

c. 3Delight

작업에 사용하는 Renderer 를 선택하는 일률적인 기준, 즉 해당 작업에 가장 적절한 Renderer 를 판단하는 절대적인 기준을 제시하는 것은 매우 어려운 작업입니다. 이는 작업 씬 환경, 그리고 때로는 작업 씬 환경을 넘어서는 너무나 많은 변수가 있기 때문입니다. 해당 작업 씬의 구성 상황은 물론, 해당 작업자의 보유 스킬, 더 나아가 해당 스튜디오에서 사용가능한 보유 라이선스 수 등 작업 내 / 외적인 모든 상황이 Renderer 의 선택에 영향을 끼치게 된다고 볼 수 있겠습니다. 이러한 여러가지 조건에서 작업 외적인 부분을 무시할 때, 3Delight 를 Houdini 에서 사용하게 되는 경우는 이하의 경우가 가장 일반적인 이유가 될 것이라고 생각합니다.

a. 상류작업의 Renderer 가 3Delight 인 경우

b. Scanline 렌더링이 필요하다고 판단되는 경우

Pipeline 전체를 3Delight 로 통일한 경우, Houdini 에서 작업한 결과를 동일한 Renderer 로 통일하는 것은 여러가지 면에서 메리트를 가지는 선택지가 될 수 있습니다. 또한 Houdini 의 기본 Renderer 인 Mantra 를 이용하는 경우 Path-Tracing 만이 사용가능 함으로, 작업상황에 따라서 Path-Tracing / Scanline 을 선택적으로 사용하고 싶은 작업 환경인 경우에도 유용한 선택지가 될 수 있습니다.

--- 참고 ---

3Delight 를 Maya 에서 사용하는 경우, Maya 용 3Delight 플러그인을 경유한 렌더링과 Standalone 타입을 사용한 렌더링이 가능합니다만, Houdini 의 경우, Houdini 용 3Delight 플러그인이 존재하지 않는 관계로 Standalone 타입을 이용한 렌더링만이 가능합니다. 다음으로 Houdini 에서 3Delight 를 사용하여 렌더링하는 경우, 기존에 이미 작성한 RSL 코드를 Houdini 가 인식가능한 OTL 로 변환가능한 툴(sdl2otl.py, slo2otl.py)이 제공됩니다.

--- 참고 끝 ---

실제로 3Delight 의 개발사는 Houdini 용 플러그인을 별도로 제공하고 있지 않은 상황입니다. 그럼에도 불구하고 Houdini 개발사인 SideEffect 측은 3Delight 를 Houdini 내에서 사용가능한 오픈 상태로 제공하고 있습니다. 이런 Houdini 의 구성은 3Delight 를 Houdini 파이프라인의 일부로 구성할 수 있는 가능성을 최대한 열어두고 있다고 보여집니다. 이러한 배려는 이미 기술한 바와 같이 VFX 이외의 상류작업에서 Houdini 이외의 보다 일반적인 3D 소프트웨어를 사용하는 경우가 일반적인 상황에서, 최종 Renderer 를 Houdini 만이 사용가능한 Mantra 로 통일하는 것은 매우 어려운 상황일 수 있기 때문이라고 보여집니다. Mantra 와 같은 Path-Tracer 로서는 렌더링에 너무나 부하가 걸리는 씬 구성이라든지, 상류과정을 포함한 파이프라인이 Rib 파일 조작을 기본으로 구성되어 있는 경우, 또는 VFX 이외의 섹션이

사용하는 렌더러가 3Delight 로 고정되어 있는 경우등 여러 가지 상황에서 이러한 Houdini 의 시뮬레이션 + 외부 렌더러의 구성이 보다 원만한 파이프라인을 구성하기 위해 선택 가능한 방법이 될 것 입니다.

--- 참고 ---

실제로 3Delight 홈페이지의 FAQ 페이지에는 Houdini 에서 3Delight 를 사용할 수 있는지를 묻는 질문에 대한 대답이 실려 있습니다.

http://www.3delight.com/en/index.php?page=3delight_faq#010