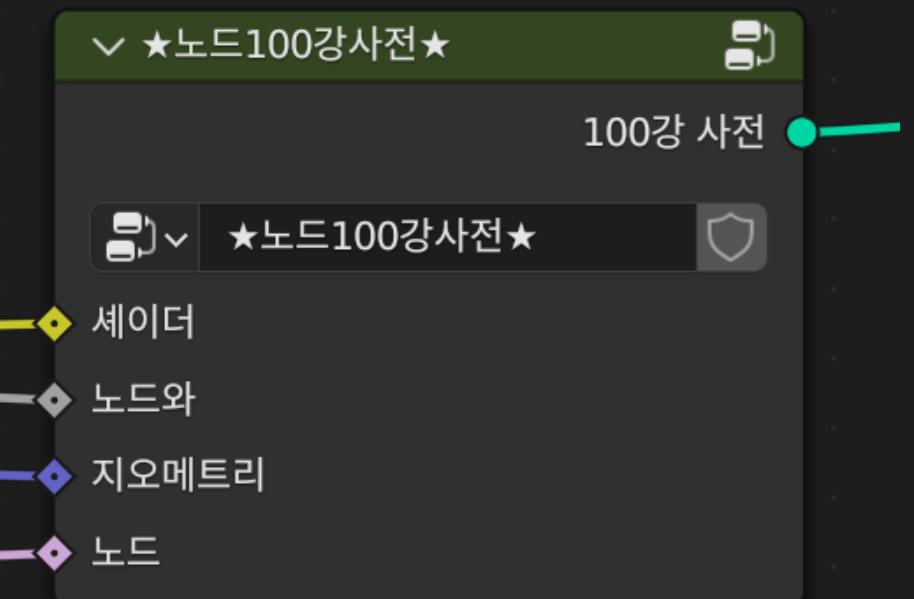


001 WELCOME

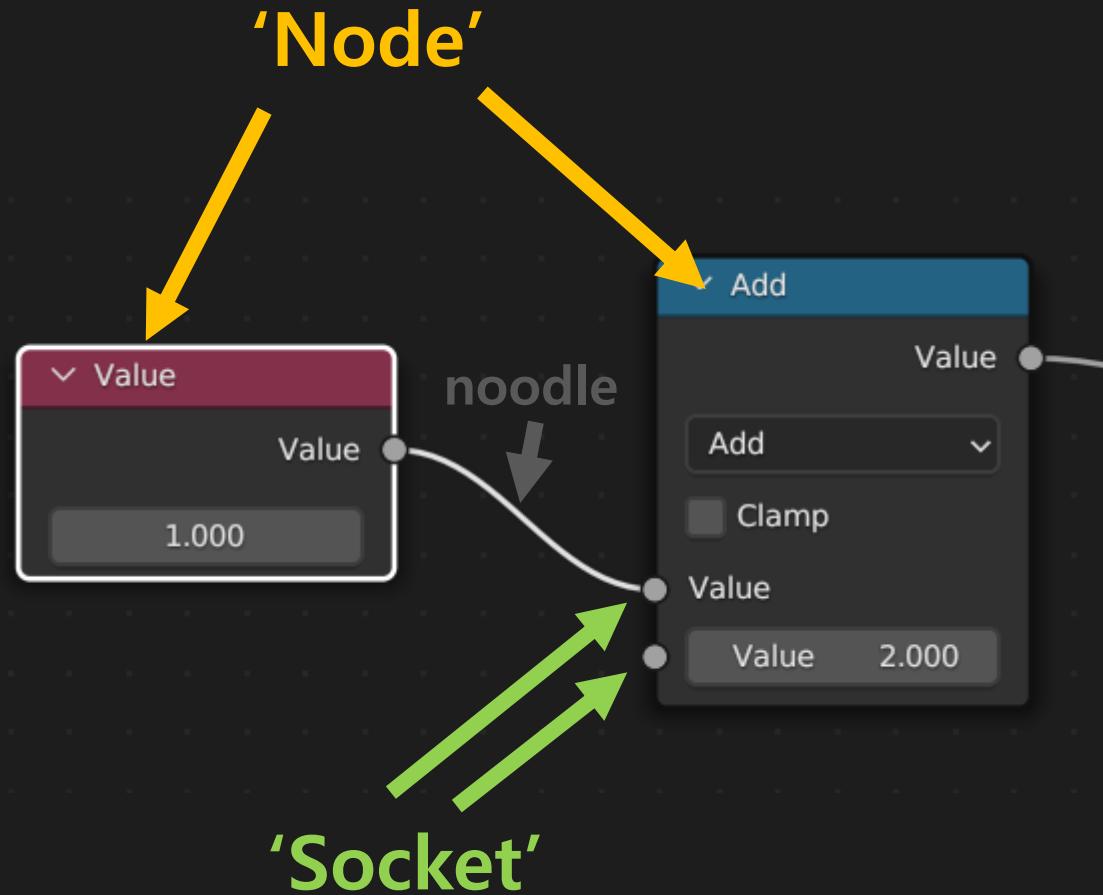
Node 기본개념

강의를 듣기 전 준비할 것들

- 워크스페이스 세팅
- 강의 자료 설명



Node?



이 클래스에서 다루는 내용

셰이더 노드와 지오메트리 노드의 이해와 응용

Shader Nodes



Geometry Nodes

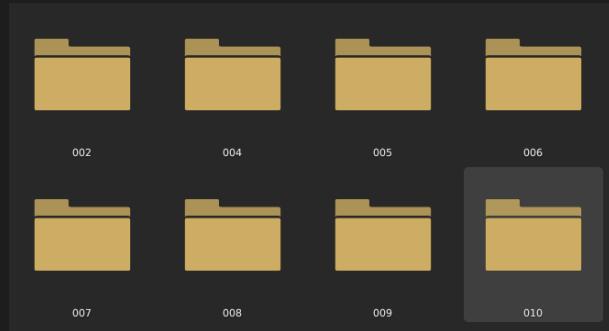


모두 합쳐서..



클래스를 효율적으로 활용하기 위해서

강의파일(.BLEND, 이미지..)



강의PDF

클래스를 효율적으로 활용하기 위해서

강의파일

프리젠테이션

클래스 지도

<https://discord.gg/ttkFHKgV3q>

클래스 지도

| 001 | 오리엔테이션 | 주요 사용 노드 |
|---------|---|--|
| WELCOME | 강의 소개 강의를 듣기 위한 준비 - 배전 - 메트 온, Quick Favorites - 워크스페이스 설정 제공되는 파일 열기 | - |
| 002 | 세이더 노드 기본 조작과 기초 사용법 | 세이더 에디터, 미티리얼의 구조와 기본 흐름 소켓의 종류와 연결 여러 가지 세이더 |
| 003 | 텍스쳐 작업 | Texture Coordinate Image Texture Separate XYZ, Combine XYZ |

디스코드 커뮤니티



Coloso Blender Nodes

100

<https://discord.gg/ttkFHKgV3q>

준비(1)

강의를 제작중인 컴퓨터 사양

I7 4790K / RTX3070 8GB / 16G DDR4 / WINDOWS 10

버전

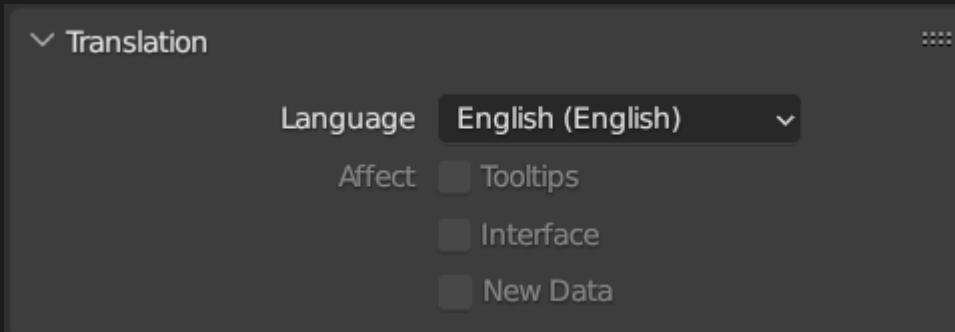


블렌더 3.5 (혹은 그 이상)

시뮬레이션 노드만 3.6으로 진행.

3.5 이후의 버전이 출시되었을 때,
만약 강의를 따라갈 수 없는 치명적인 변경점이 있다면
추후에 알려드리겠습니다.

언어



준비(2)

다음의 Addon은 켜져 있다고 가정합니다

- ▶ Node: Node Wrangler 
- ▶ Import-Export: Import Images as Planes 

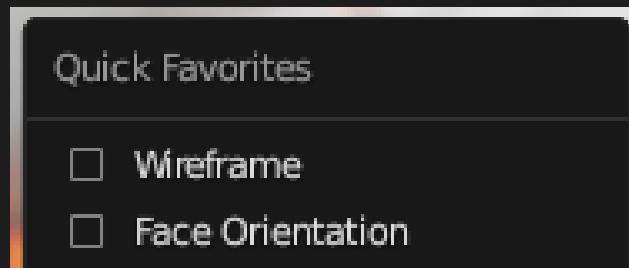
그 외 제가 사용하는 애드온들

(이 강의에서는 자주 사용하지 않습니다)

- ▶ Interface: Copy Attributes Menu 
- ▶ Object: Bool Tool 
- ▶ Mesh: LoopTools 

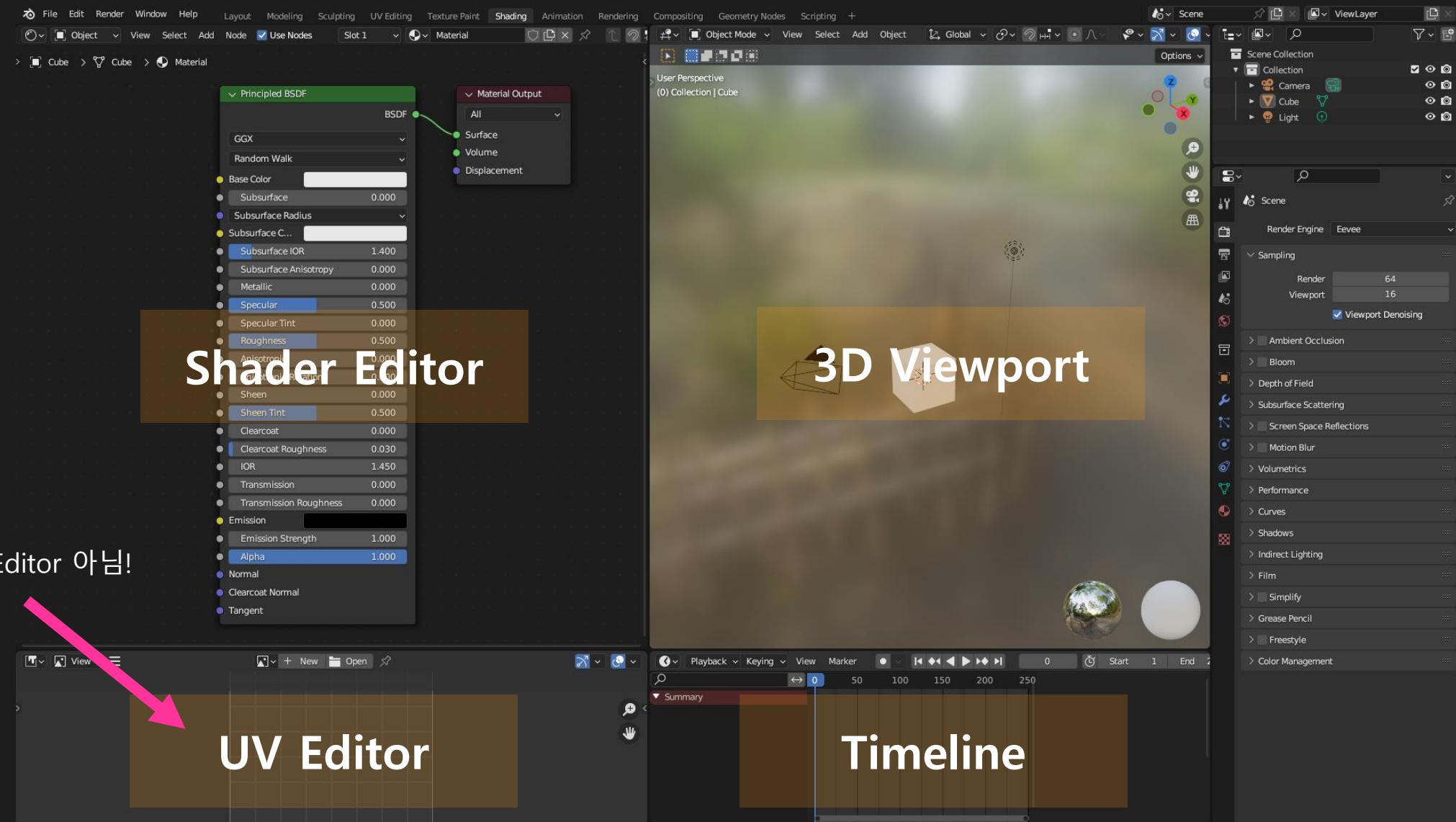
이외의 애드온을 사용할 때는 강의중에 알려드리겠습니다.

Quick Favorites 설정

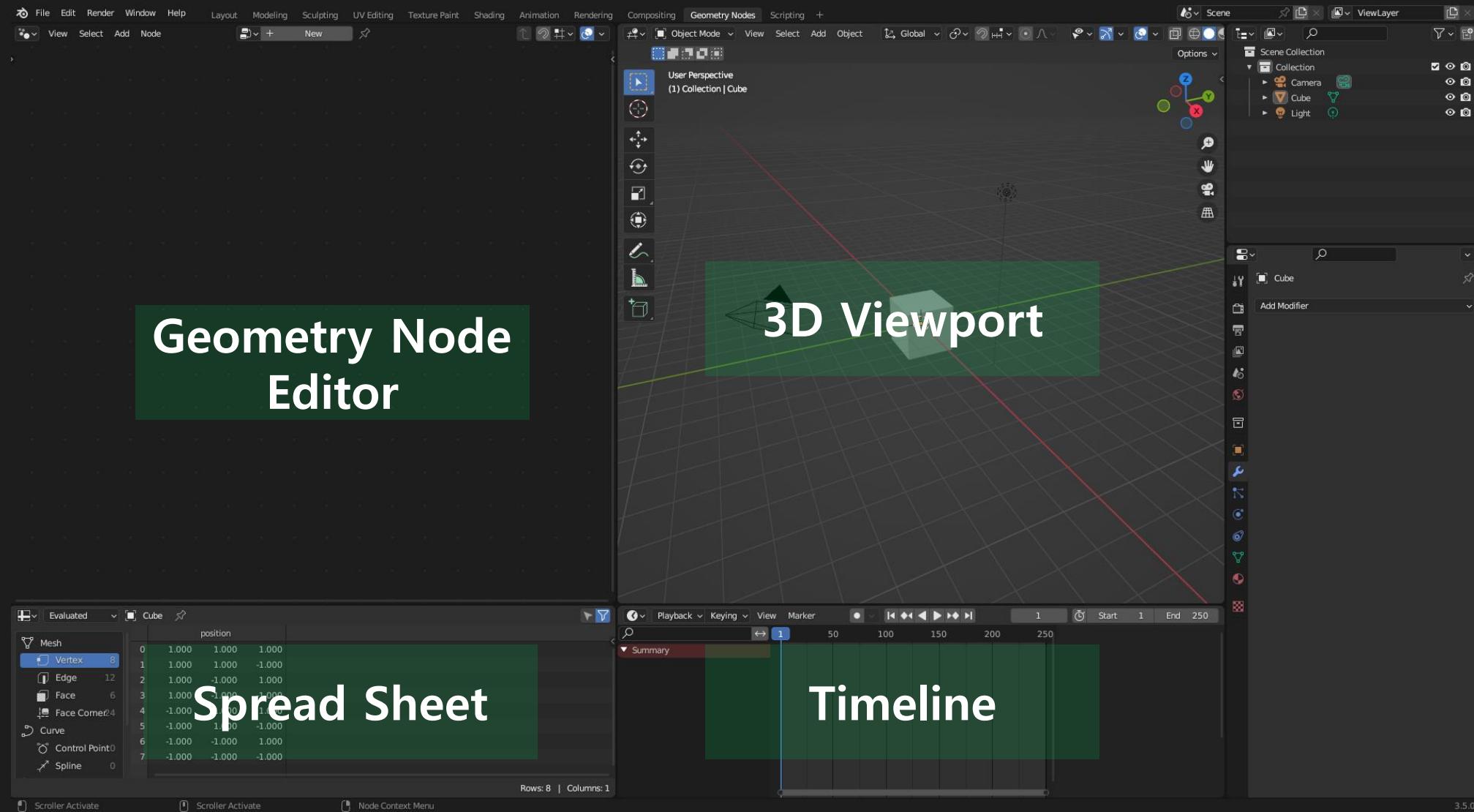


Object Mode와 Edit Mode 모두
Wireframe과 Face Orientation 설정 합니다.

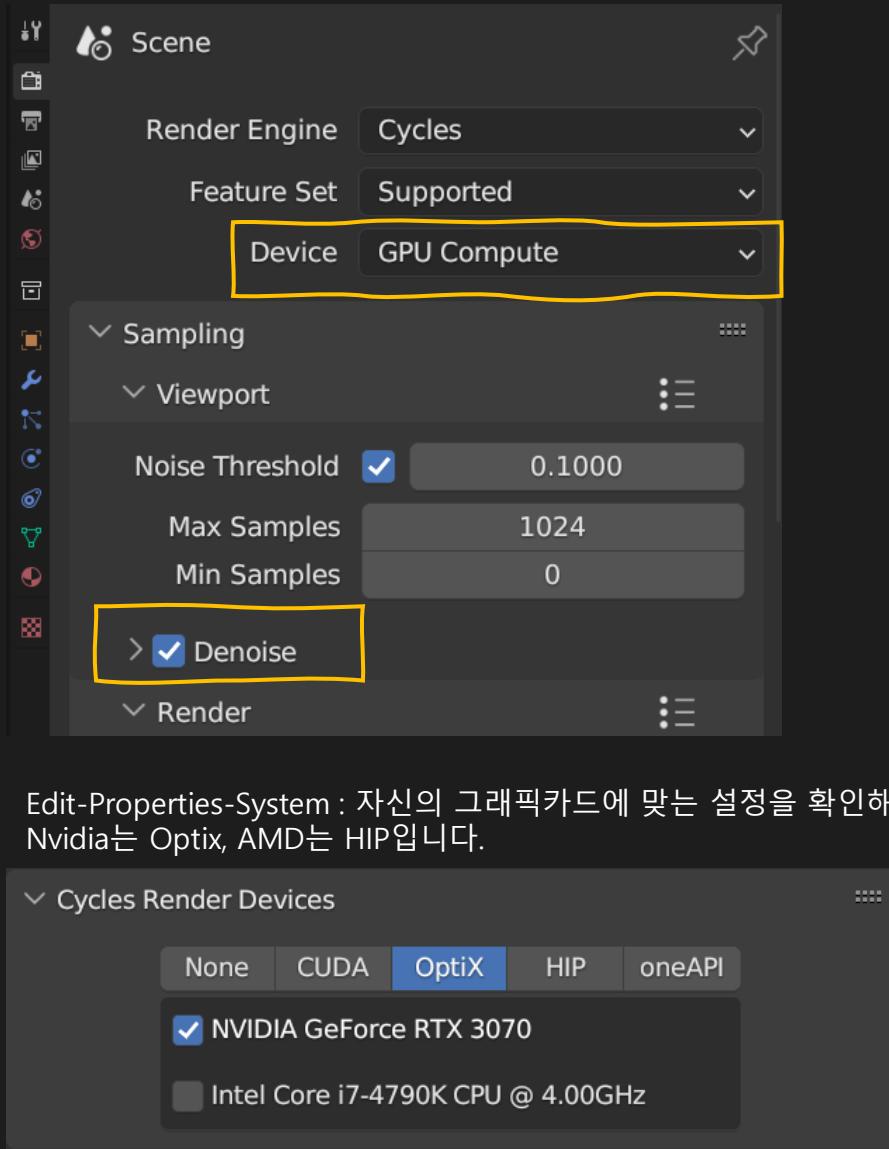
준비(3) 워크스페이스 설정-Shading



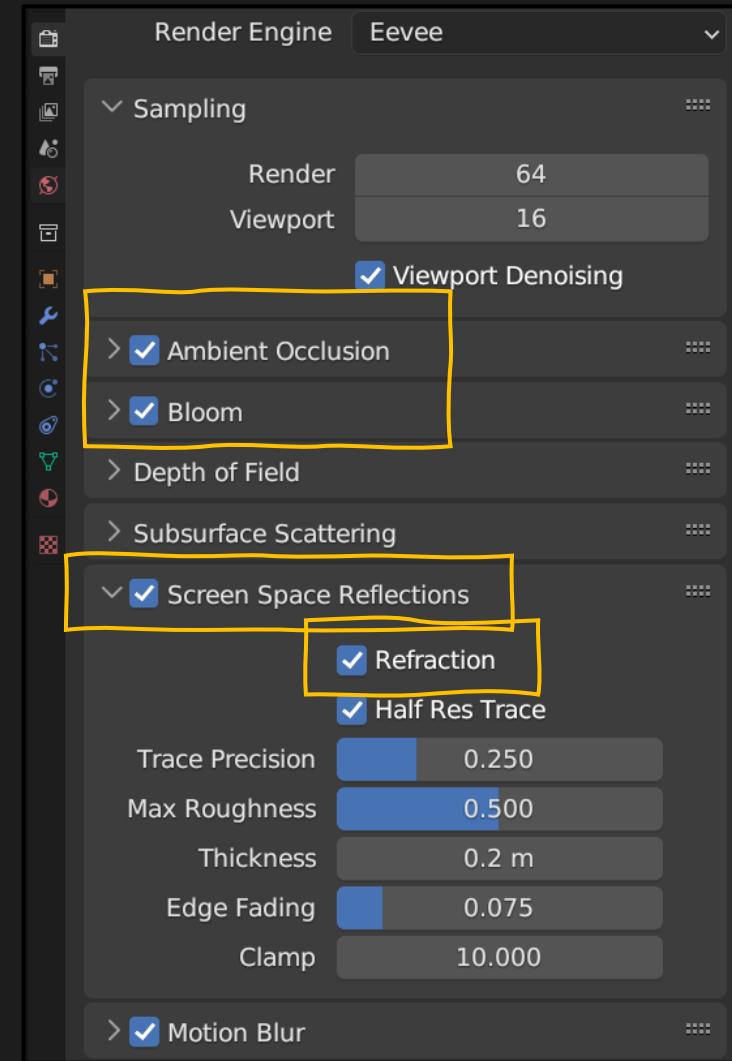
준비(3) 워크스페이스 설정-Geometry Nodes



준비(4) 그외 설정들

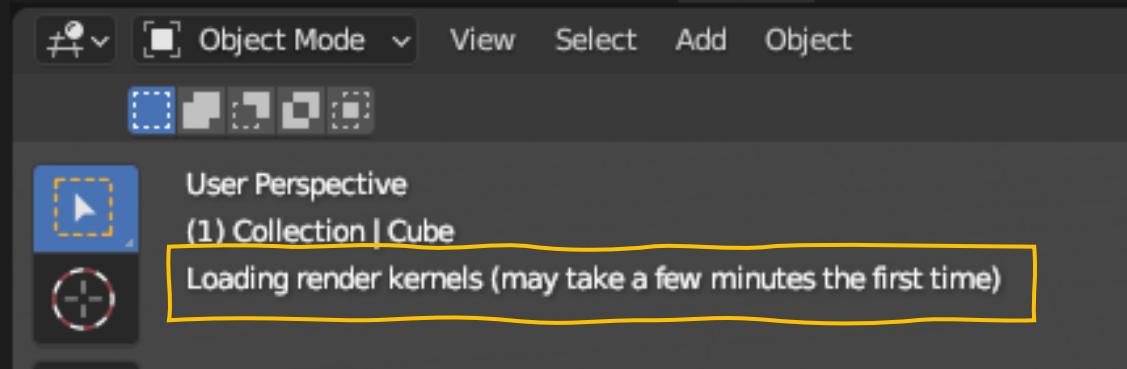


Edit-Properties-System : 자신의 그래픽카드에 맞는 설정을 확인해 주세요.
Nvidia는 Optix, AMD는 HIP입니다.



변경 후 File – Defaults – Save Startup File

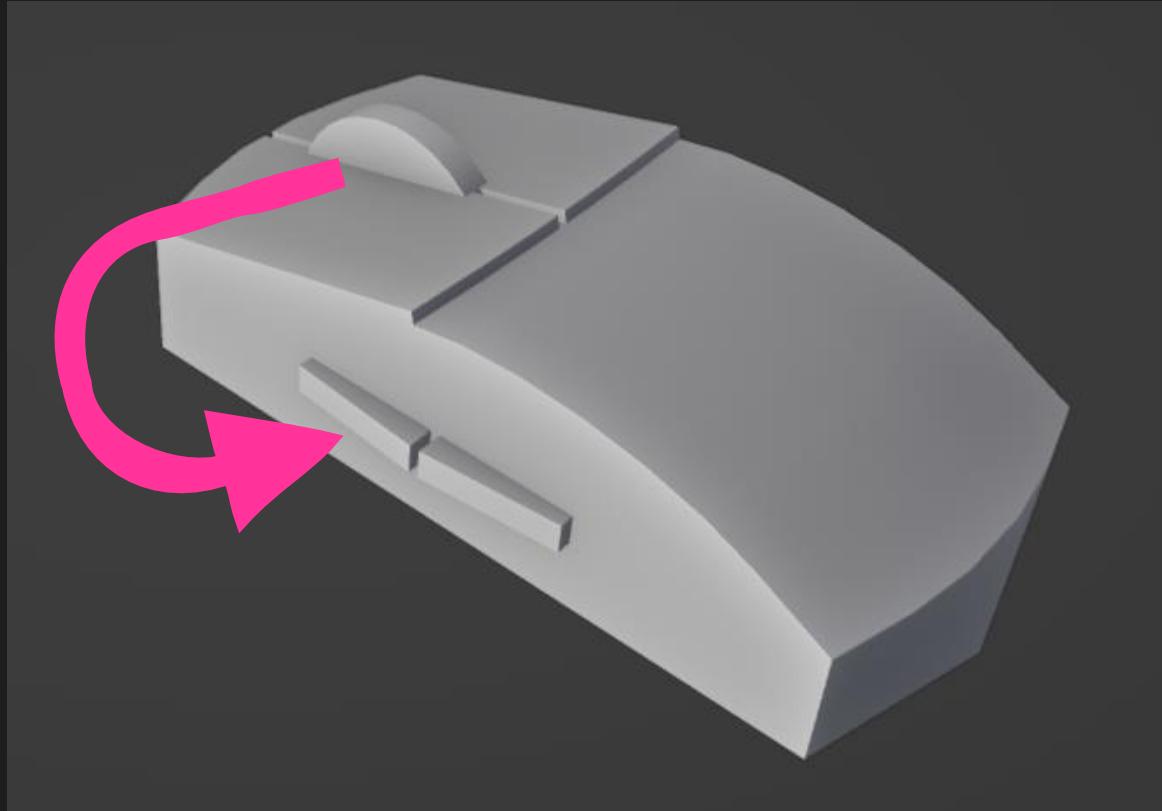
준비(5) 그외 확인사항



처음 설치 혹은 버전 업그레이드시, Cycles 렌더에 'Loading render kernels'라는 메시지와 함께 렌더링이 멈출 때가 있습니다. 몇 분, 길게는 수십 분까지 시간이 소요될 수 있는데, 주로 특정 세이더 노드를 처음 연결할 때 표시됩니다.

처음 설치 후 2번에서 3번정도 이 메시지가 표시될 수 있는데, 차분히 기다려 주시면 정상적으로 진행됩니다.

마우스 설정?



Viewport 이동(회전) 기본값은 **마우스 휠** 버튼입니다만, 누르기 불편할 수 있습니다.
저는 마우스 **사이드버튼**에 휠 버튼을 할당하여 사용중입니다.

기본조작

Object 모드

| 조작키 | 설명 |
|------------------|---------------------|
| 마우스 휠 드래그 | 화면 회전 |
| Shift+마우스 휠 드래그 | 화면 이동 |
| 마우스 휠 | 화면 확대/축소 |
| T | Tool (좌측 도구창) |
| N | Context (우측 설정창) |
| A, Alt+A | 전체 선택, 선택 취소 |
| Shift+A | 오브젝트 생성 |
| Shift+D | 오브젝트 복제 |
| Ctrl+Z | 실행 취소 |
| Ctrl+Shift+Z | 실행 취소의 취소! (=다시 실행) |
| X | 오브젝트 삭제 |
| G | 이동 |
| R (RR) | 회전 (트랙볼 회전) |
| S | 오브젝트 확대, 축소 |
| G,R,S 상태에서 X,Y,Z | X,Y,Z 축을 따라 이동 |

Edit 모드

| 조작키 | 설명 |
|-----------------------|---------------------|
| Tab | 오브젝트- 에딧 모드 토글 |
| Shift+A | 메쉬 생성 |
| 1,2,3 | 차례대로 점,선,면 선택모드 |
| X | 삭제 메뉴 |
| Ctrl+X | Dissolve Vertices |
| G,R,S | 이동, 회전, 확대축소 |
| G 상태에서 Shift+,Z | Z 축을 '제외하고' 이동 |
| Ctrl+R | Loop Cut |
| Ctrl+R 후 마우스휠 | Loop Cut 개수 |
| E | Extrude |
| Alt+E | Extude 메뉴 창 |
| I (I,I) | Inset (모드 변경) |
| Ctrl+B (Ctrl+Shift+B) | Bevel (Vertex mode) |
| Shift 누르고 클릭 | 다중 선택 |
| Ctrl 누르고 클릭 | 최단거리 선택 |

기본조작

카메라 설정

| 조작키 | 설명 |
|-----------------------|---------------|
| numpad 0 | 카메라 뷰로 전환 |
| Ctrl + Alt + numpad 0 | 현재 뷰로 카메라를 설정 |

강의중 블렌더의 기본조작은 설명드리지 않습니다.

조작에 익숙치 않으시다면 표를 확인하고 한번쯤 연습해보는 것을 추천드립니다.

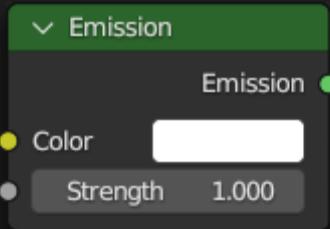
002강 셰이더 노드 기본 조작과 기초 사용법

셰이더 에디터, 머티리얼의 구조

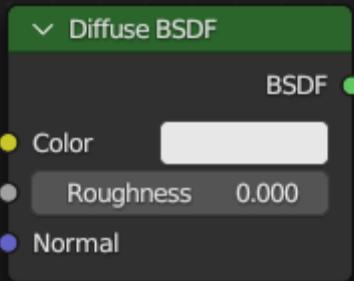
여러가지 셰이더

셰이더 노드의 연결

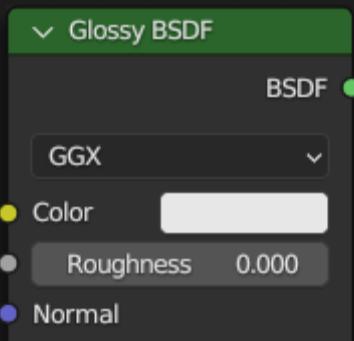
Shaders 여러가지 셰이더



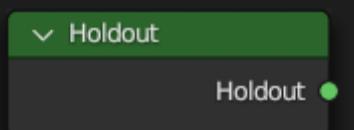
입력받은 Color (색, 이미지..) 를 밝기만 변경하여 그대로 내보냅니다.
발광체나 보조 조명의 용도로 사용합니다



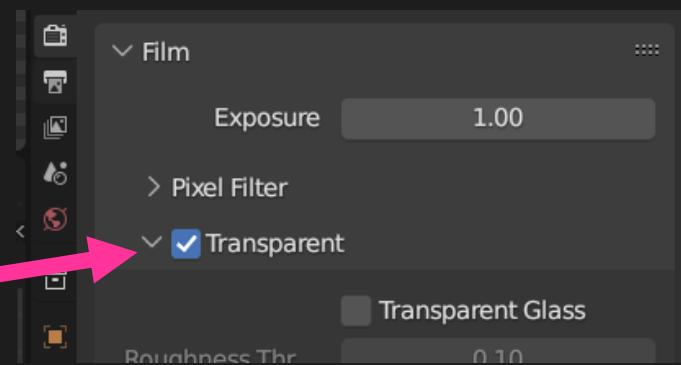
거친 표면을 만듭니다.
(Roughness 수치를 올리면 거친 표면에서 '더 거친' 표면이 됩니다. 이 수치는 cycles에서만 작동합니다.)



매끈한 금속 느낌이 나는 셰이더입니다.



해당 재질을 사용한 부분의 '렌더 이미지가' 투명해지는 신기한 셰이더.
※참고 : 배경을 투명하게 하고 싶을때는 렌더 설정의 Film - Transparent를 켭니다.



소켓의 종류



Float

실수 (ex : 1, -1, 0.1, -0.31418....)



Color

RGB 색 데이터 (ex : 빨간색 : (1,0,0), 파란색 : (0,0,1)....)



Vector

XYZ 벡터 데이터 (ex : 위쪽 방향 (0,0,1), 오른쪽 방향 (1,0,0)....)

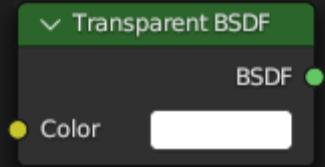


Shader

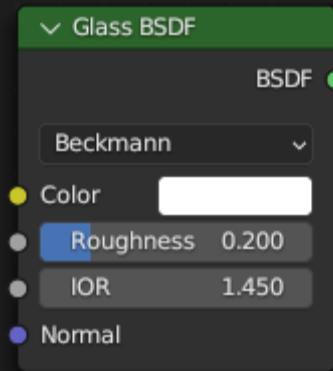
쉐이더 데이터.

Shaders

어떤 셰이더는 Eevee에서 사용할 때 설정이 필요합니다

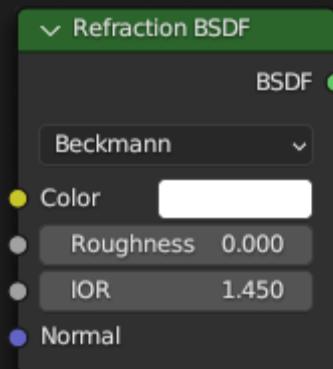


투명을 표현하는 셰이더.
Eevee에서는 옵션의 Blend mode를 Alpha xxx로 설정해야 작동합니다.

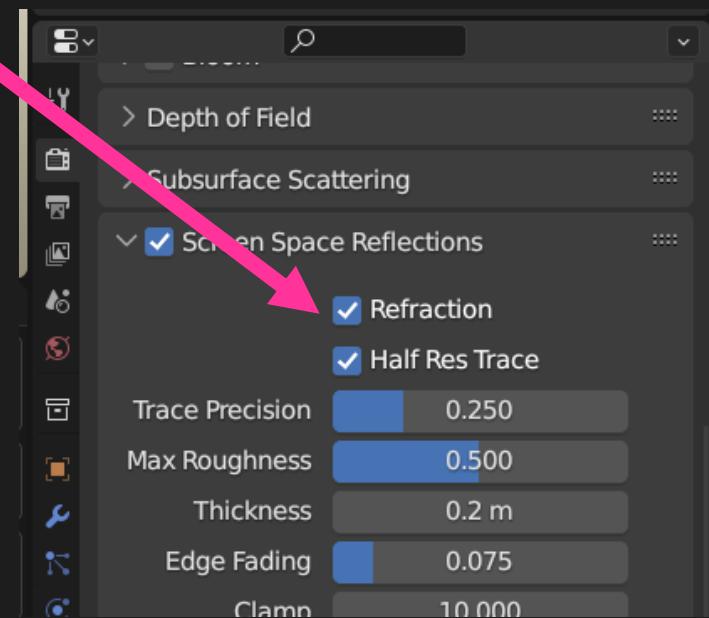
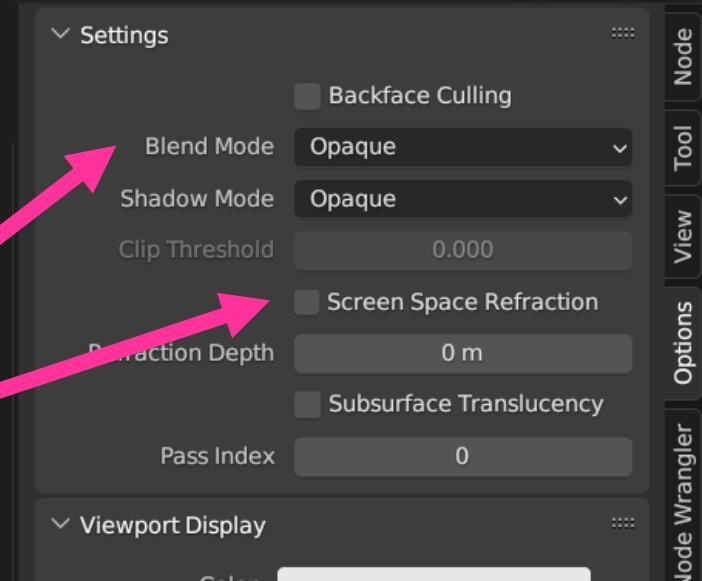


유리를 표현하는 셰이더.
Eevee에서는 Screen Space Refraction을 켜야 작동합니다.

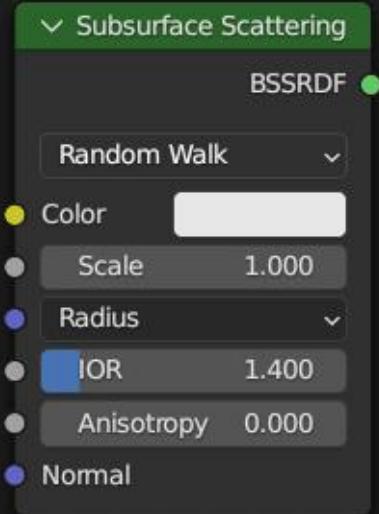
Roughness는 표면이 아니라 내부의 거칠기를 조절합니다.
IOR은 굴절률을 조절합니다.



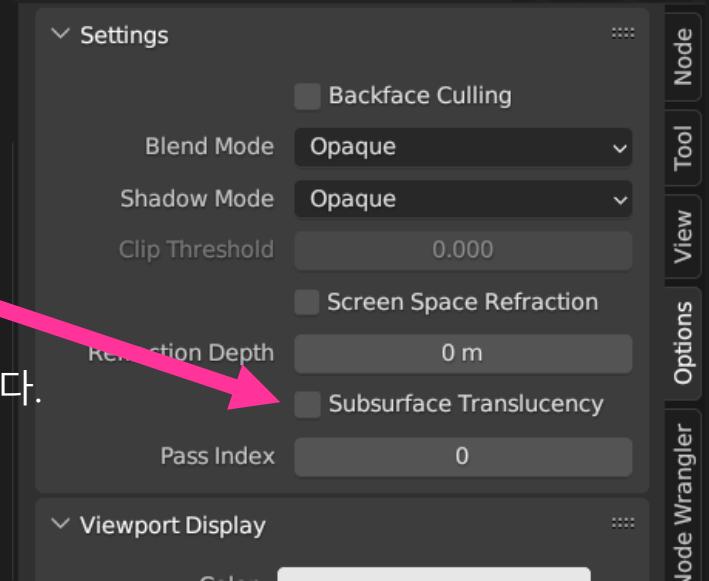
Glass BSDF와 비슷하지만, '굴절'만 표현합니다.
표면에서 반사는 일어나지 않습니다.



Shaders 여러가지 셰이더

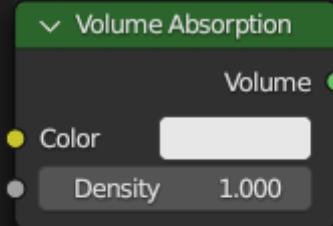


피부와 같이 미묘하게 투과하는 재질을 만들 때 사용하는 셰이더.
Eevee에서는 Subsurface Transcluency를 켜야 작동합니다.



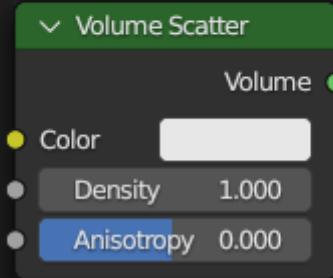
거친 표면에서 일어나는 투과를 표현하는 셰이더.
투과되는 깊이나, 거칠기를 조절하는 옵션이 없음에 유의.

Shaders 여러가지 셰이더



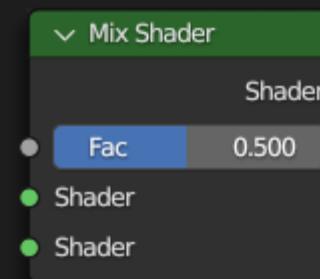
거리, 깊이에 따라 어두워지는 볼륨 효과를 만듭니다.

※ Volume 노드는 Material Output의 Volume소켓에 꽂습니다.



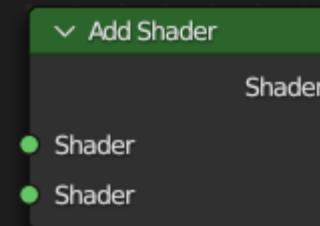
빛이 지나면서 산란되어 눈에 보이는 효과를 만듭니다.

외부 조명의 각도와 위치에 따라 효과가 극대화됩니다.



입력받은 두 셰이더를 합성합니다!

Factor 값이 작을수록 위쪽 셰이더, 클수록 아래쪽 셰이더를 더 많이 섞습니다.

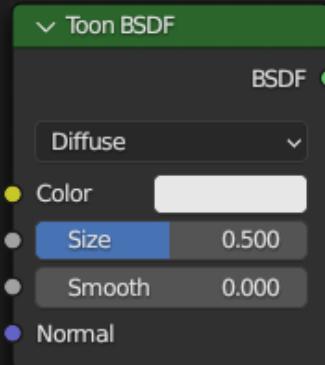


입력받은 두 셰이더를 더합니다.

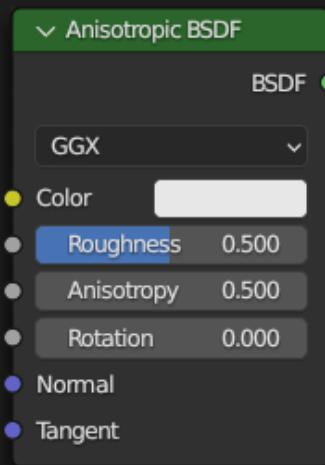
포토샵의 add 블렌딩모드와 비슷합니다.
자세한 설명은 비디오 텍스쳐 강의에서.

Shaders

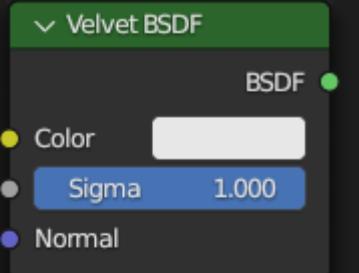
Cycles에서만 작동하는 셰이더도 있습니다



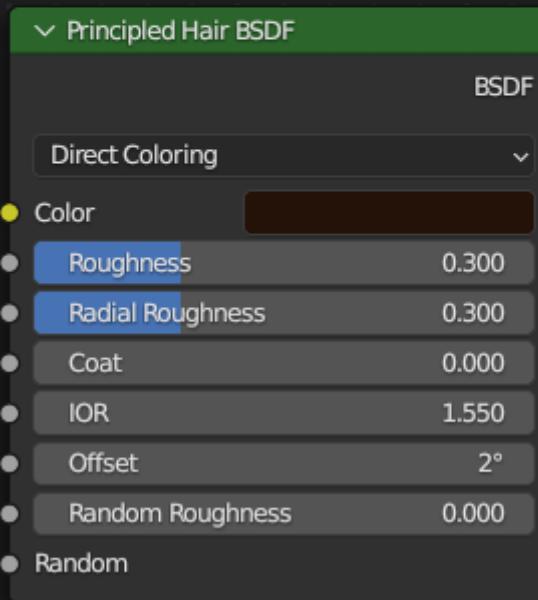
만화와 같은 표현을 해주는 셰이더.
이와 같은 효과는 Eevee가 더 잘 처리하므로
잘 쓰이지는 않습니다.



Brushed Metal과 같은, 특정 방향으로
거칠기가 있는 표면을 표현합니다.
방향을 바꾸고 싶으면 Tangent 소켓에
추가적인 방향을 입력해야 합니다.



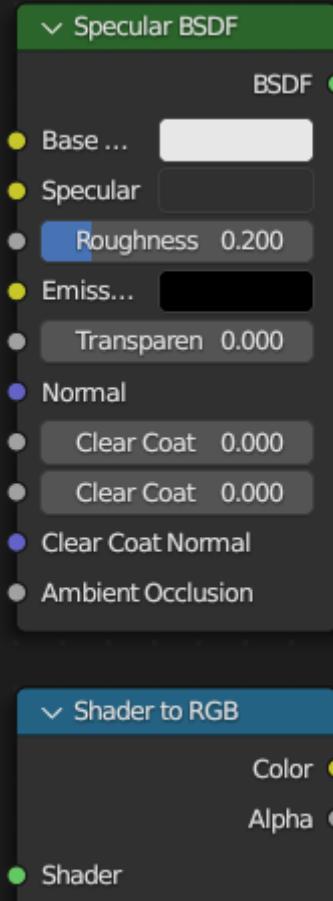
천 재질의 가장자리를 표현해 줍니다.
기술적으로 오래되어서, 잘 쓰이지는 않습니다.



Hair Particles
(혹은 최신버전의 Hair Curves)
에 적용하기 위한 머리카락 재질입니다.
자세한 이야기는 지오메트리 노드의 헤어 강의 참고.

Shaders

Eevee에서만 작동하는 셰이더도 있습니다

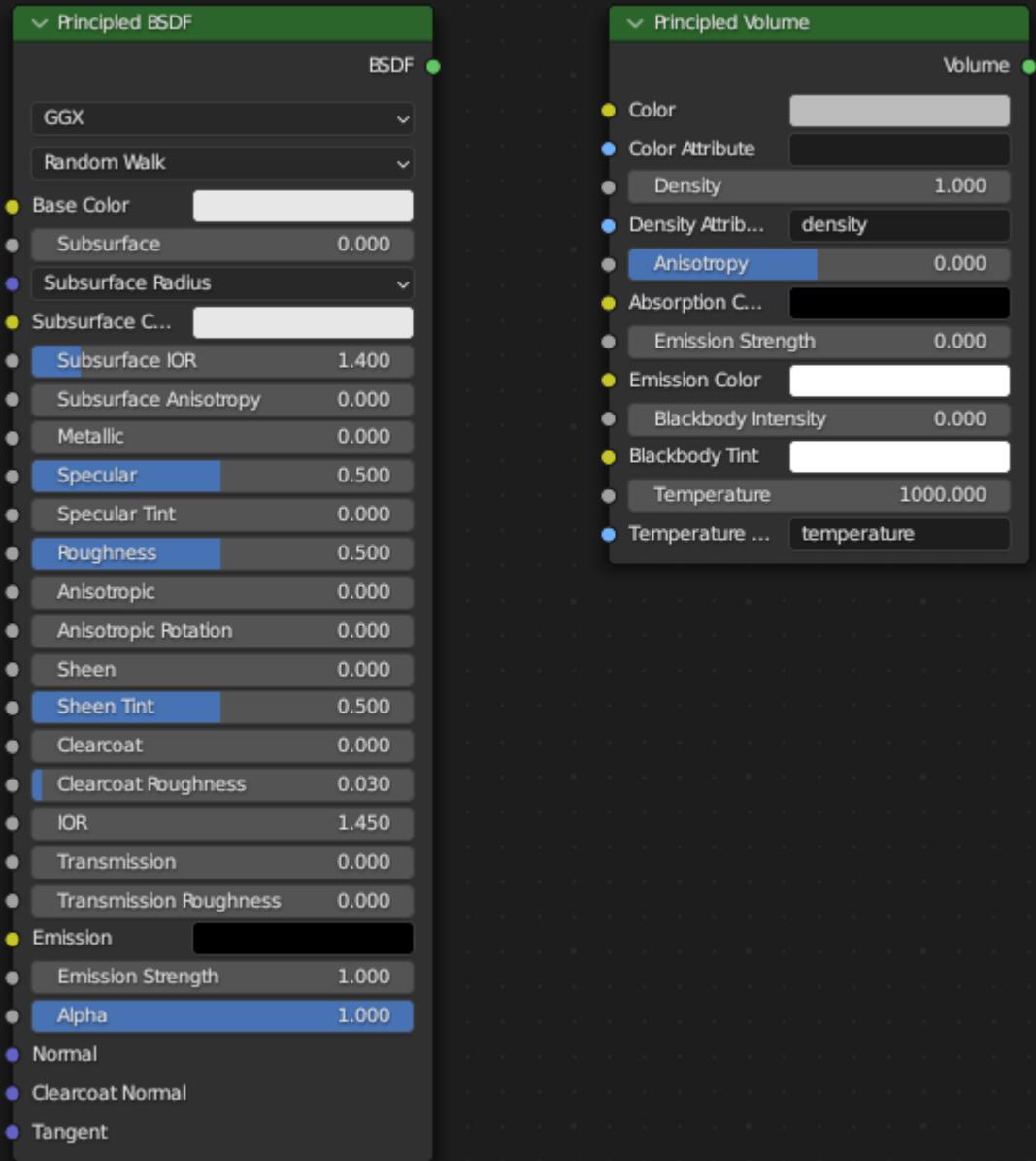


표면색과 반사되는 색을 분리해서 지정할 수 있습니다.

이것은 셰이더가 아니고
셰이더를 이미지로 변환시켜 주는 특이한 노드입니다. (NPR 강의 참고)

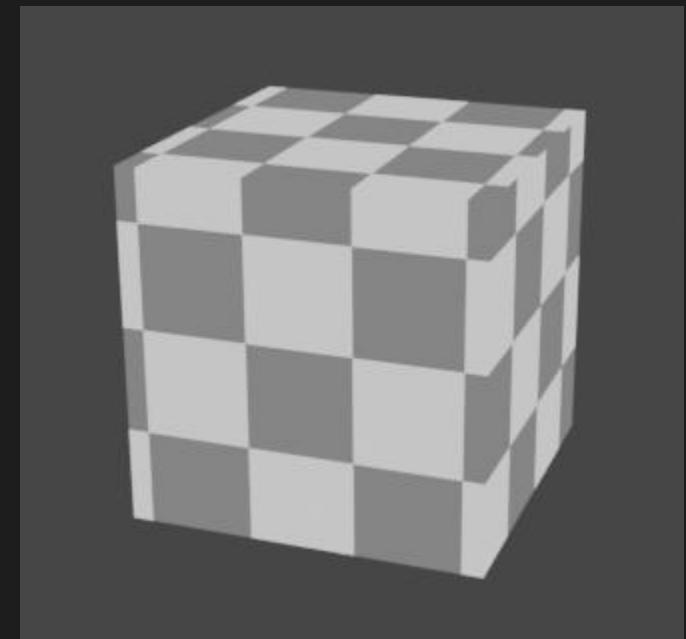
Principled Shader

(거의) 모든 것을 통합한 셰이더.



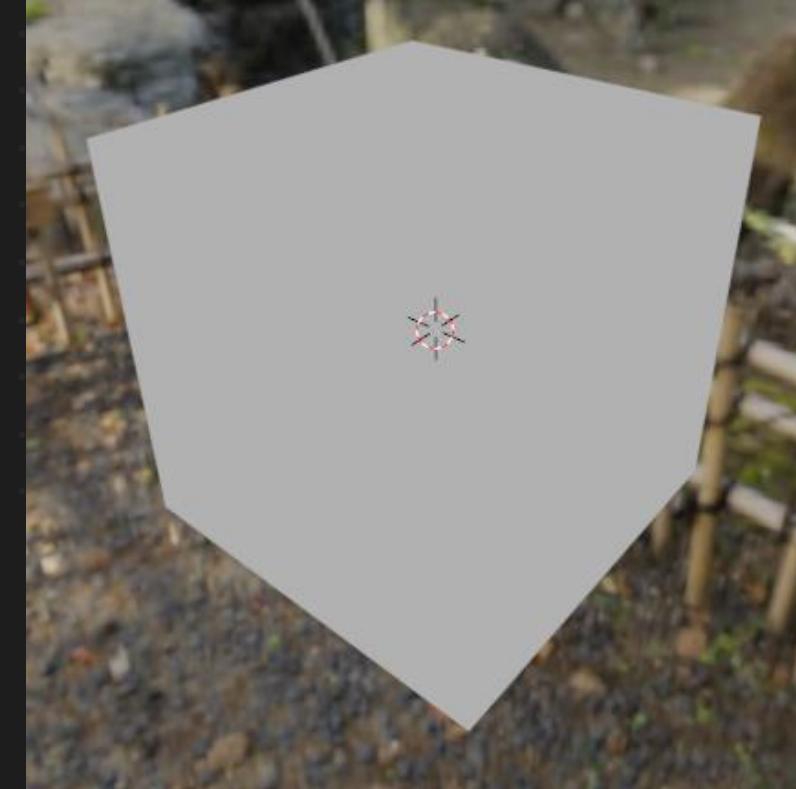
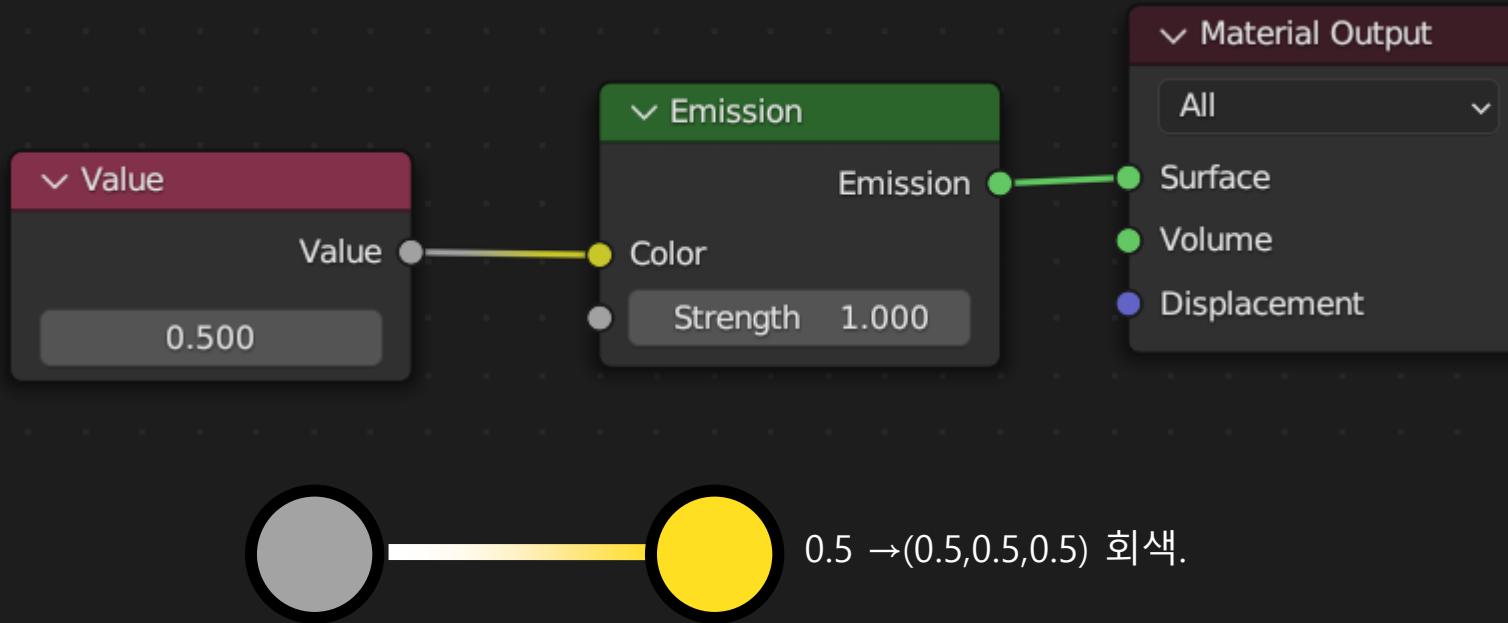
기본 연결 :

Texture – Shader – Material Output



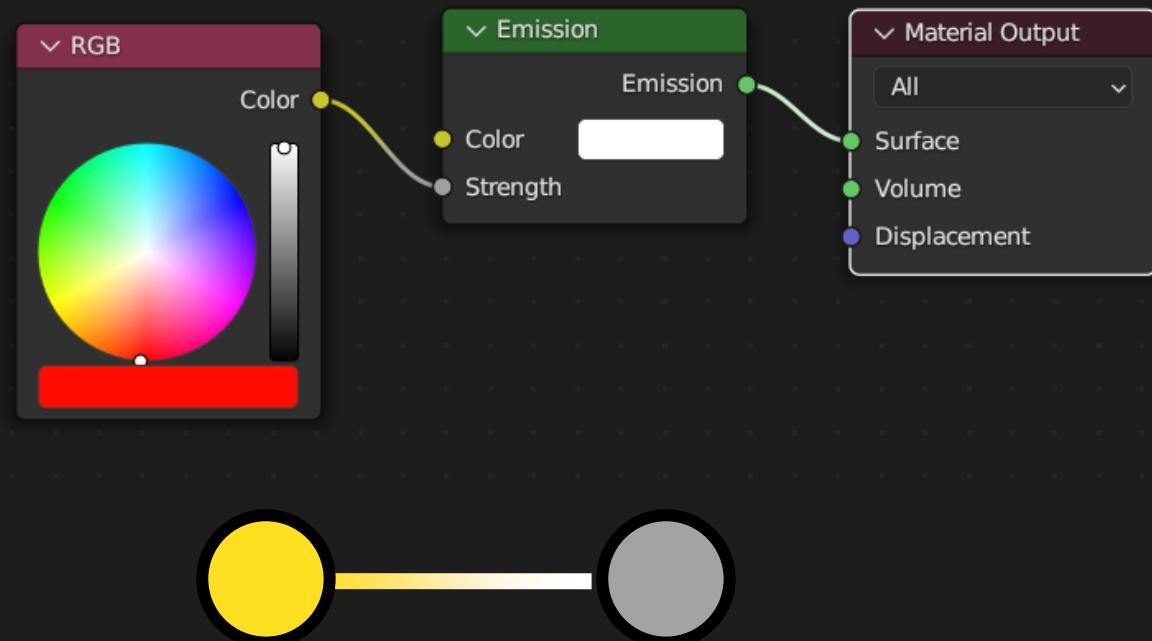
Float을 Color에 꽂으면?

Float값을 컬러에 꽂으면, RGB각각에 동일한 값이 입력되어, 무채색이 출력됩니다.

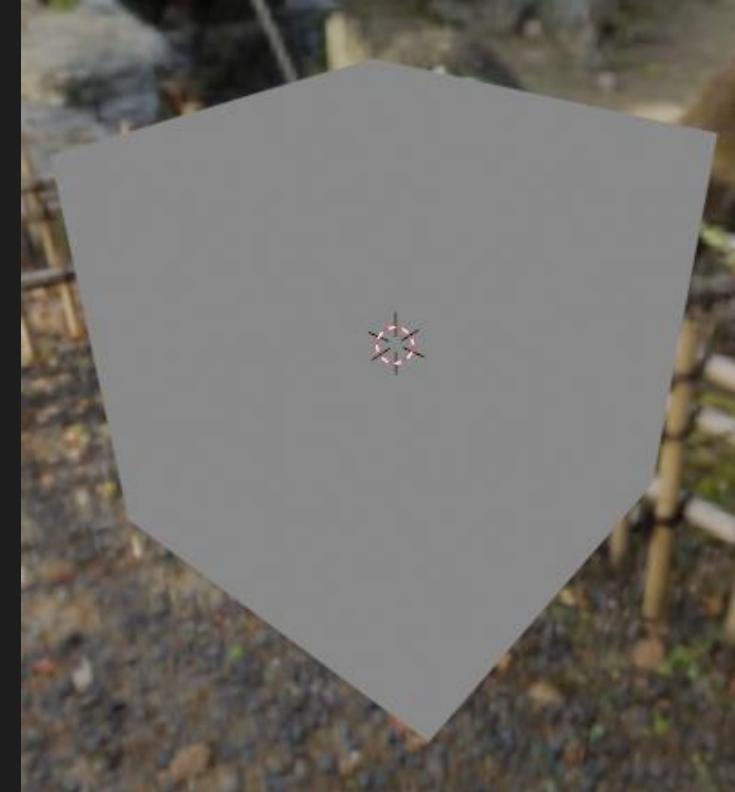


Color를 Float에 꽂으면?

Color를 Float에 꽂으면, Color의 '밝기' 가 출력됩니다.

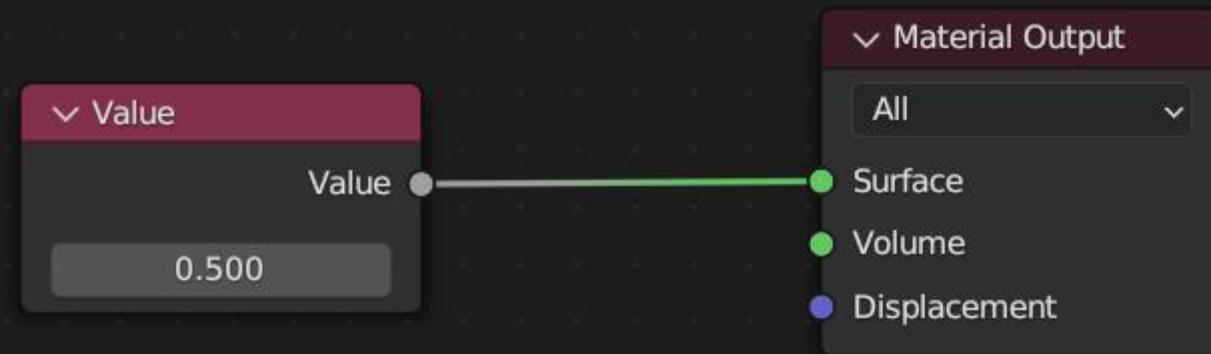


$(1,0,0) \rightarrow$ 빨간색의 '밝기' = 0.2126..



소켓 미리보기

Material Output 한정으로, 어떤 데이터타입이라도 꽂을 수 있습니다.
Emission의 Color에 꽂은 것과 같은 효과입니다.



003강 Vector

Texture좌표

- 이미지가 3차원 표면에 입혀지는 원리
- UV, Generated, Object 좌표의 의미와 특징
- Image Texture의 매핑 방식

Normal

- Normal Map

2차원 이미지를 어떻게 3차원에 입힐 것인가?

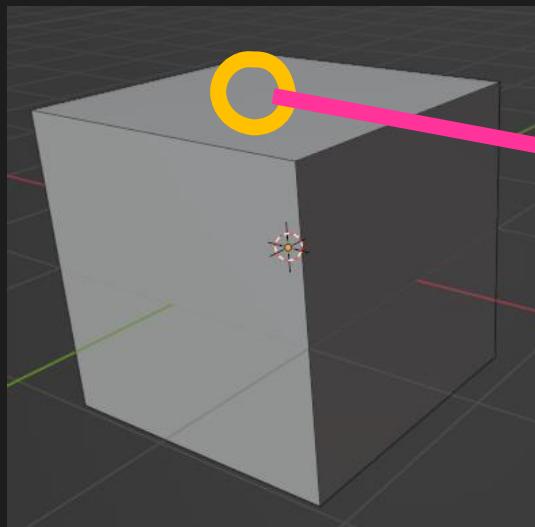
UV

전개도와 비슷한 원리

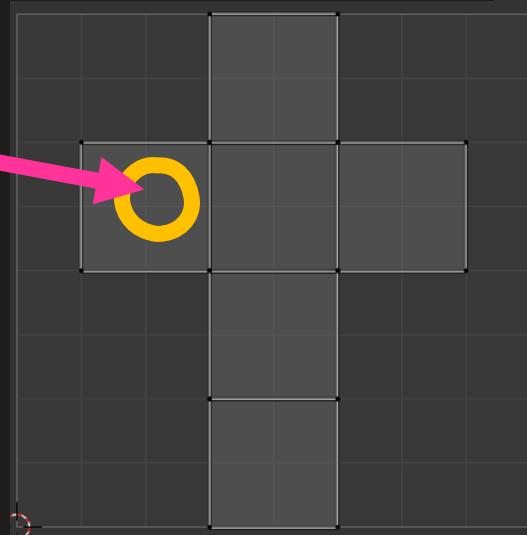


2차원 전개도가 어떻게 3차원에 연결되는가?

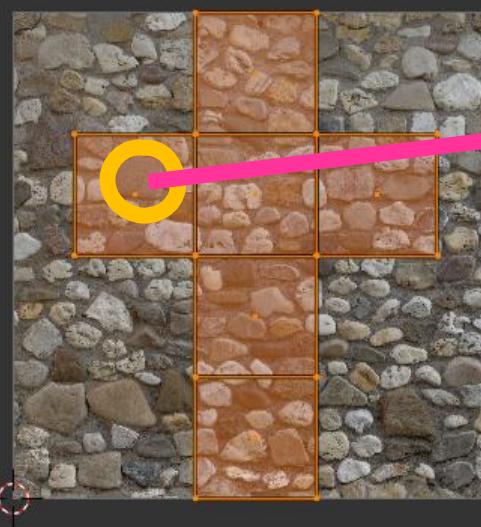
UV맵이 만드는 위치를 3차원의 위치에 대응시킨다.



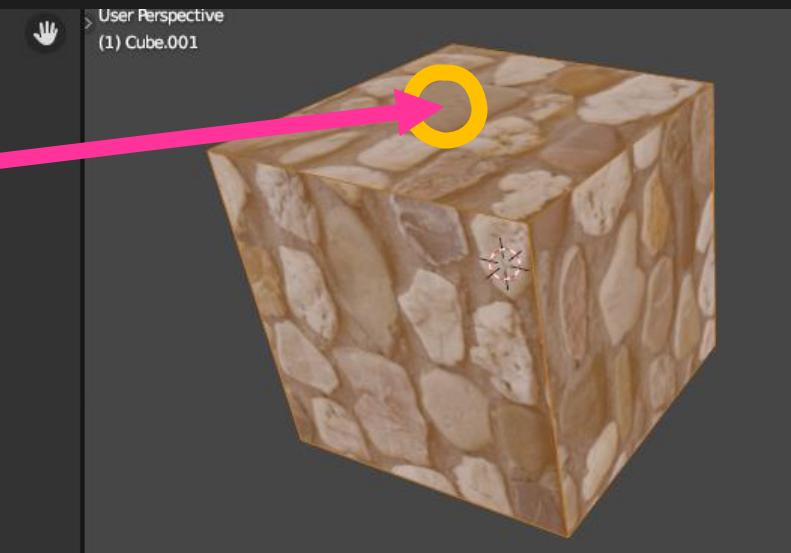
이 점을..



2차원 평면의 점과
대응시킵니다.



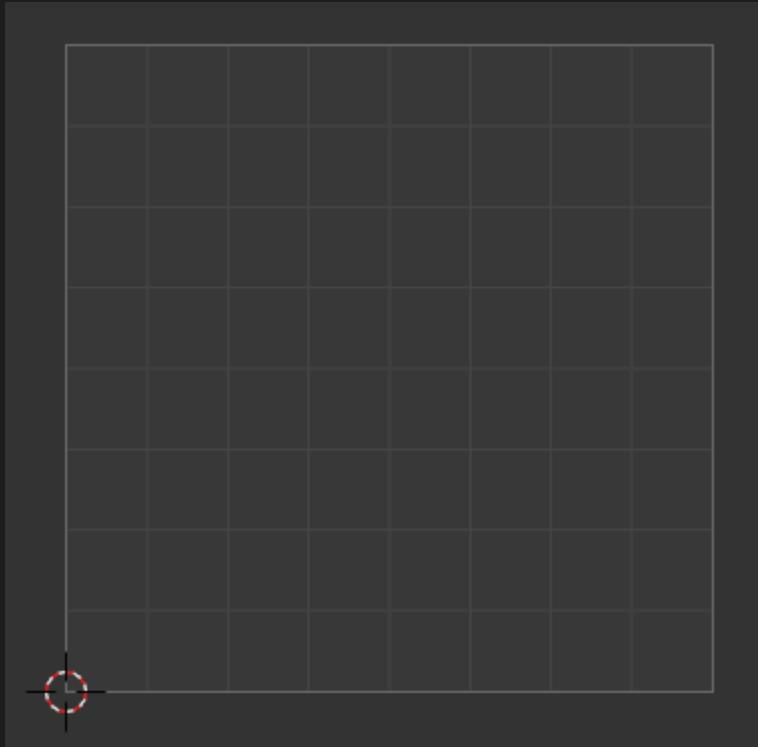
그 지점의 이미지
색을 읽습니다.



그 색을 표면에
나타냅니다.

2차원 전개도가 어떻게 3차원에 연결되는가?

1. UV좌표

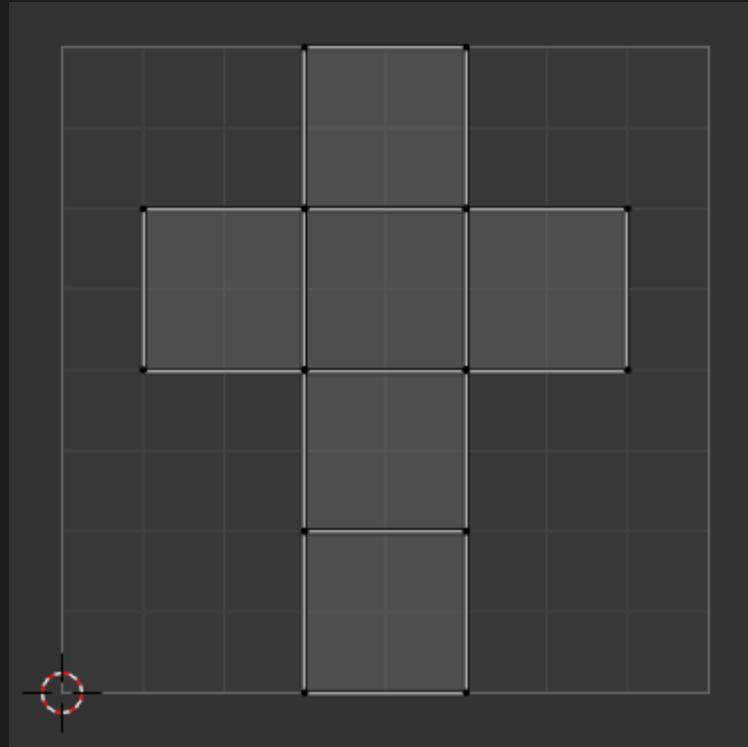


UV좌표는 이미지가 사용하는 2차원 좌표입니다.
이것은 XYZ축이 만드는 3차원 공간과는 구분되기 때문에, X,Y,Z를 사용하지 않고
새로운 알파벳인 U,V로 표시합니다.

가로 축이 U, 세로 축이 V이고, 가로 세로 방향으로 0에서 1 사이의 값을 가집니다.

2차원 전개도가 어떻게 3차원에 연결되는가?

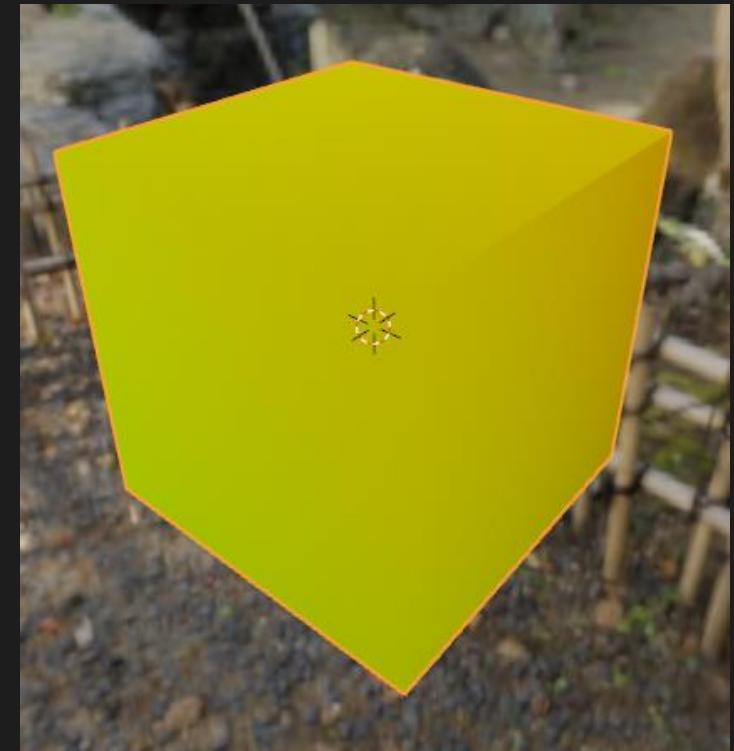
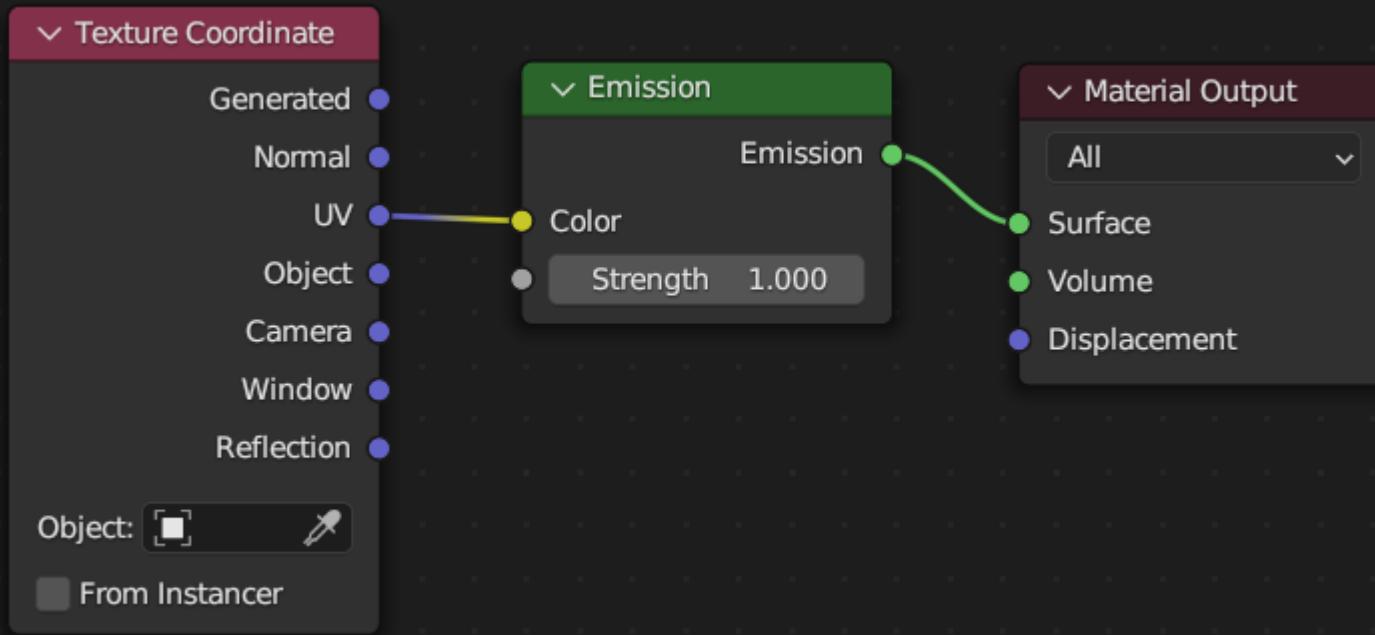
2. UV unwrap



3차원 오브젝트 각각의 면을 UV의 어디에 위치시킬 것인지 결정합니다.

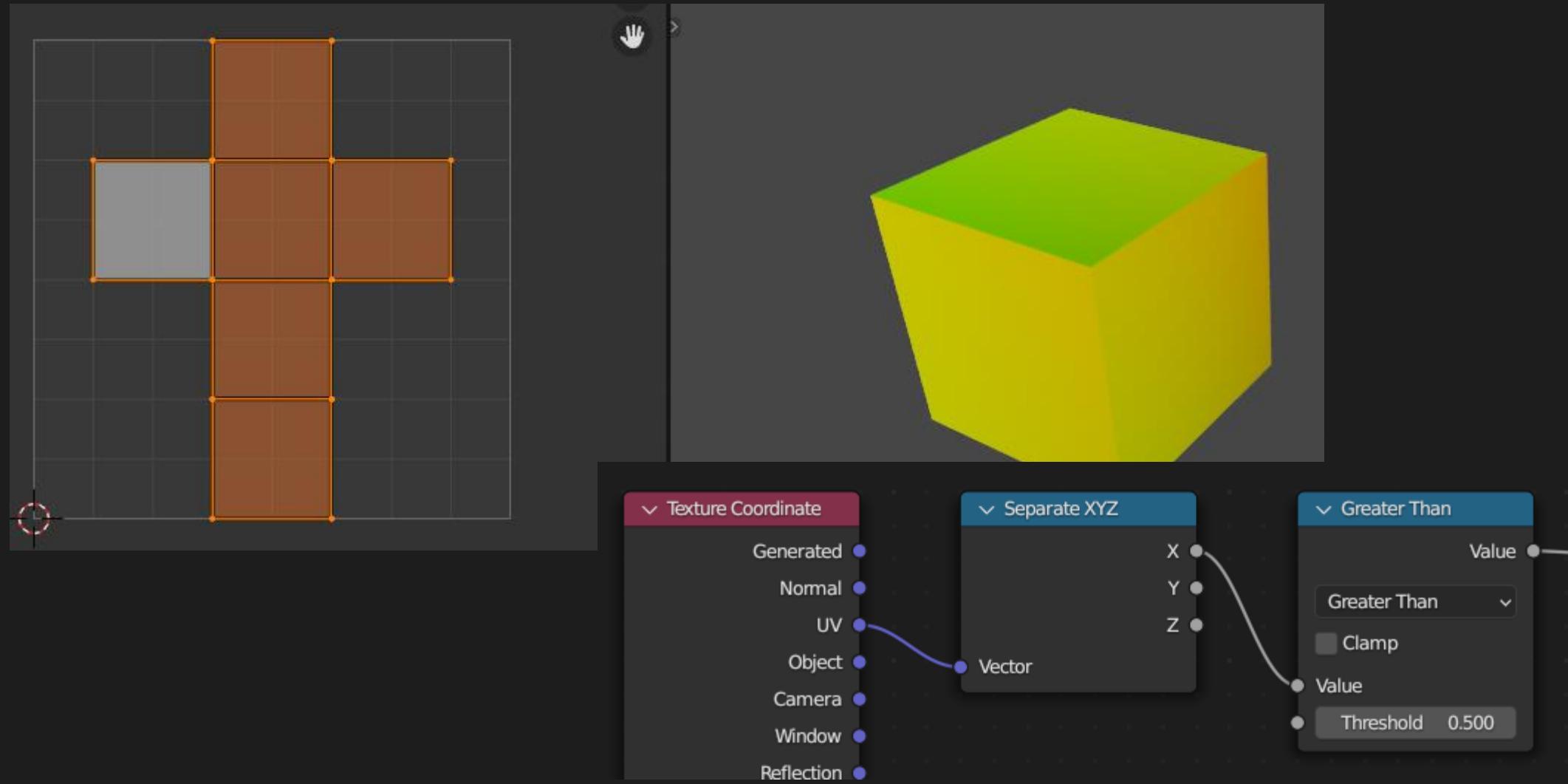
Vector의 미리보기

Vector값을 출력에 꽂으면, X,Y,Z 각각이 R,G,B에 대응됩니다.
즉, X축은 빨간색, Y축은 초록색, Z축은 파란색을 나타냅니다.



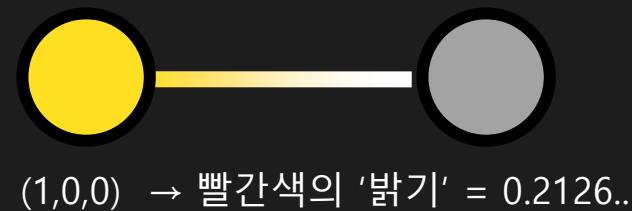
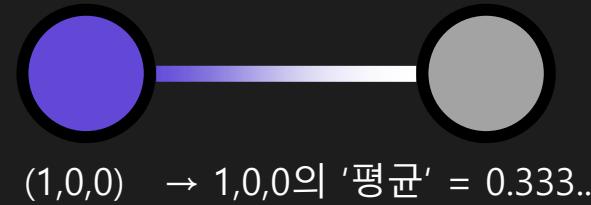
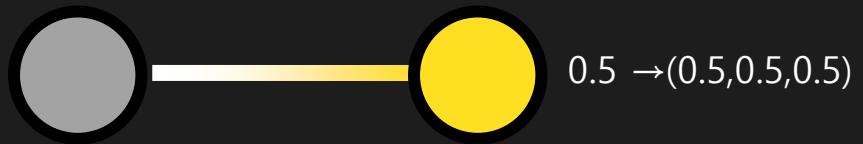
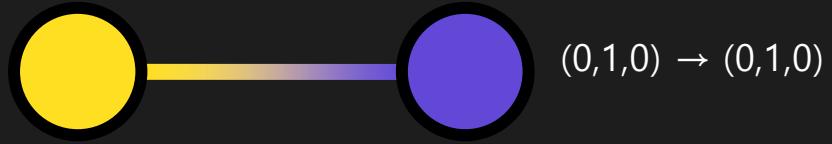
UV 좌표의 색(?)

UV는 빨간색에서 초록색 사이, 주로 노란색을 띕니다. (why?) Separate XYZ와 Math 노드를 사용해 봅시다.



소켓 호환 정리

외우실 필요는 없습니다. 합당한 방식으로 호환된다는 정도만 기억해 주세요.



2차원 전개도가 어떻게 3차원에 연결되는가?

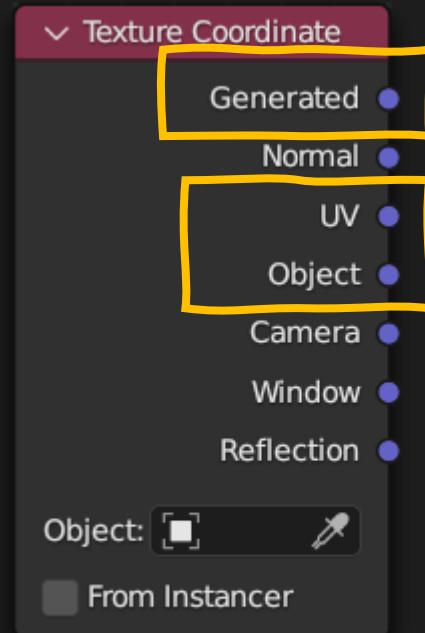
3. Image Texture



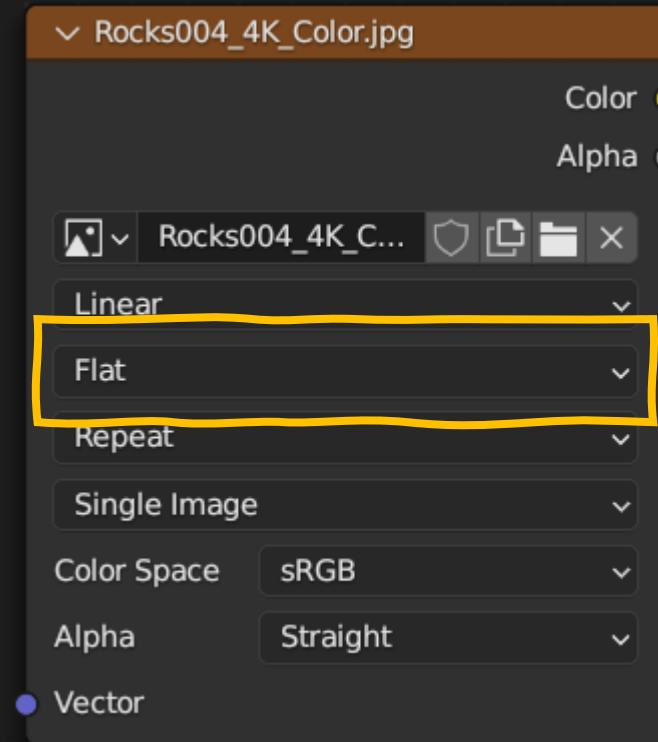
이미지는 UV 좌표에 놓입니다.
이 때, 이미지 해상도에 관계없이, 가로 세로가 0에서 1 사이에 맞춰집니다.
따라서 가로 세로 크기가 같지 않은 이미지는 좌표 위의 비율이 틀어지게 됩니다.

좌표와 맵핑 방식

텍스쳐 좌표 : Generated, UV, Object



맵핑 방식 : Flat, Box, Sphere, Tube

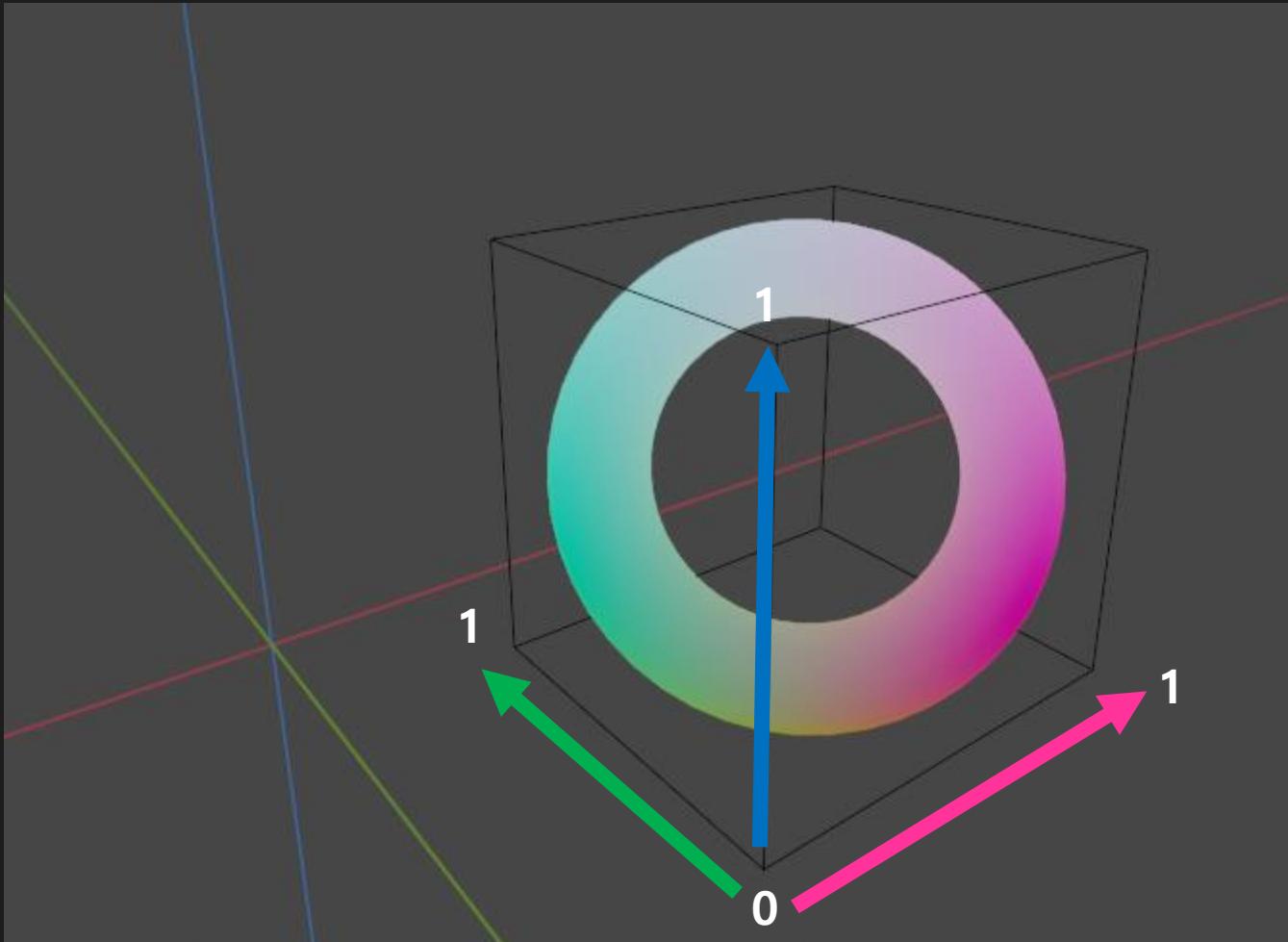


※Camera, Window 좌표는 직접 사용해 보세요

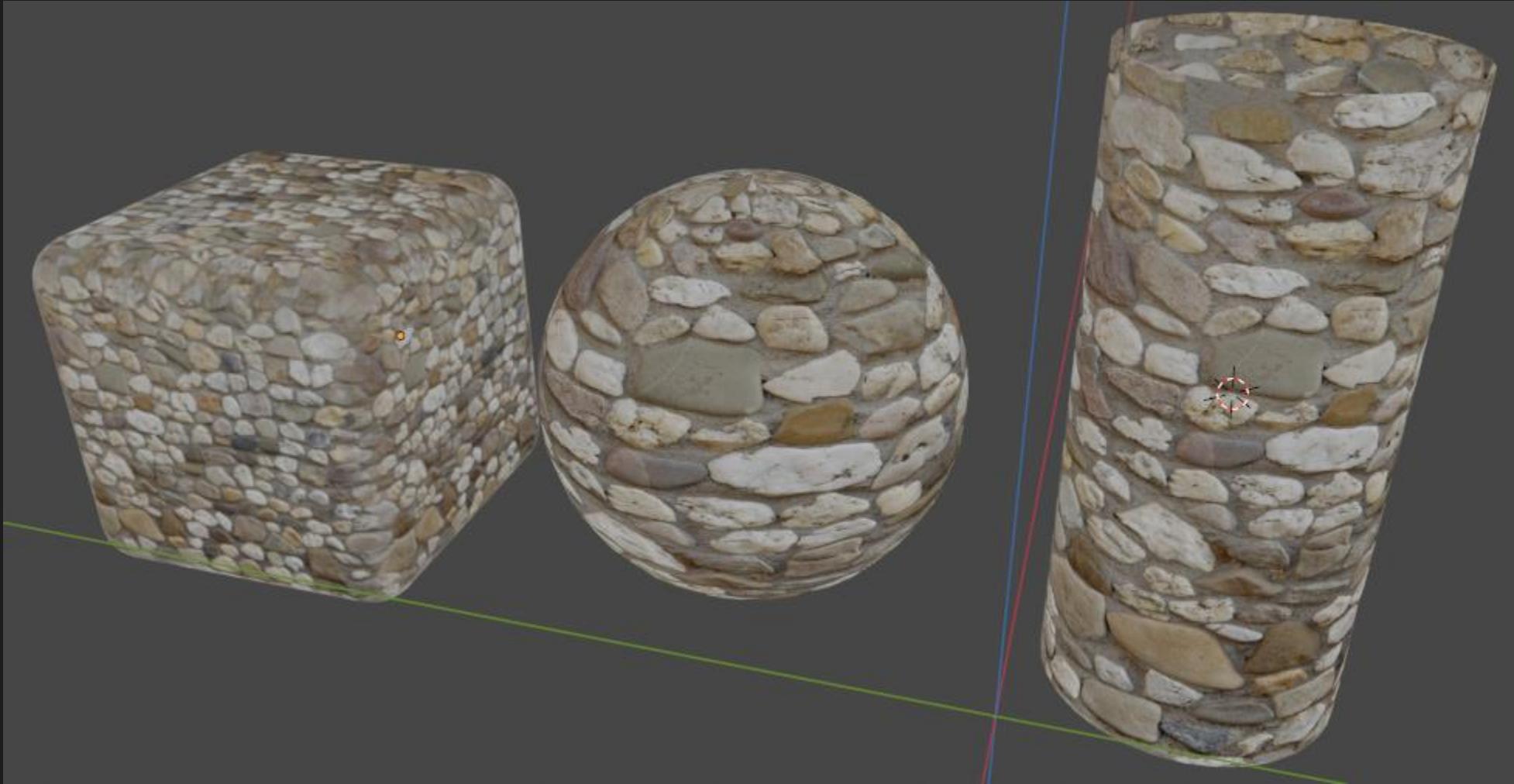
※Reflection은 Environment 텍스쳐를 사용해야 작동합니다. (NPR 강의 참고)

물체를 감싸는 3차원 Generated 좌표

Bounding Box의 가로, 세로, 높이를 0에서 1사이의 값으로 지정



Generated 좌표를 위한 Box, Sphere, Tube 매핑



오리진에서부터 뻗어나가는 Object 좌표

오브젝트의 오리진을 중심으로, X,Y,Z 방향으로 뻗어나갑니다.



Generated vs Object

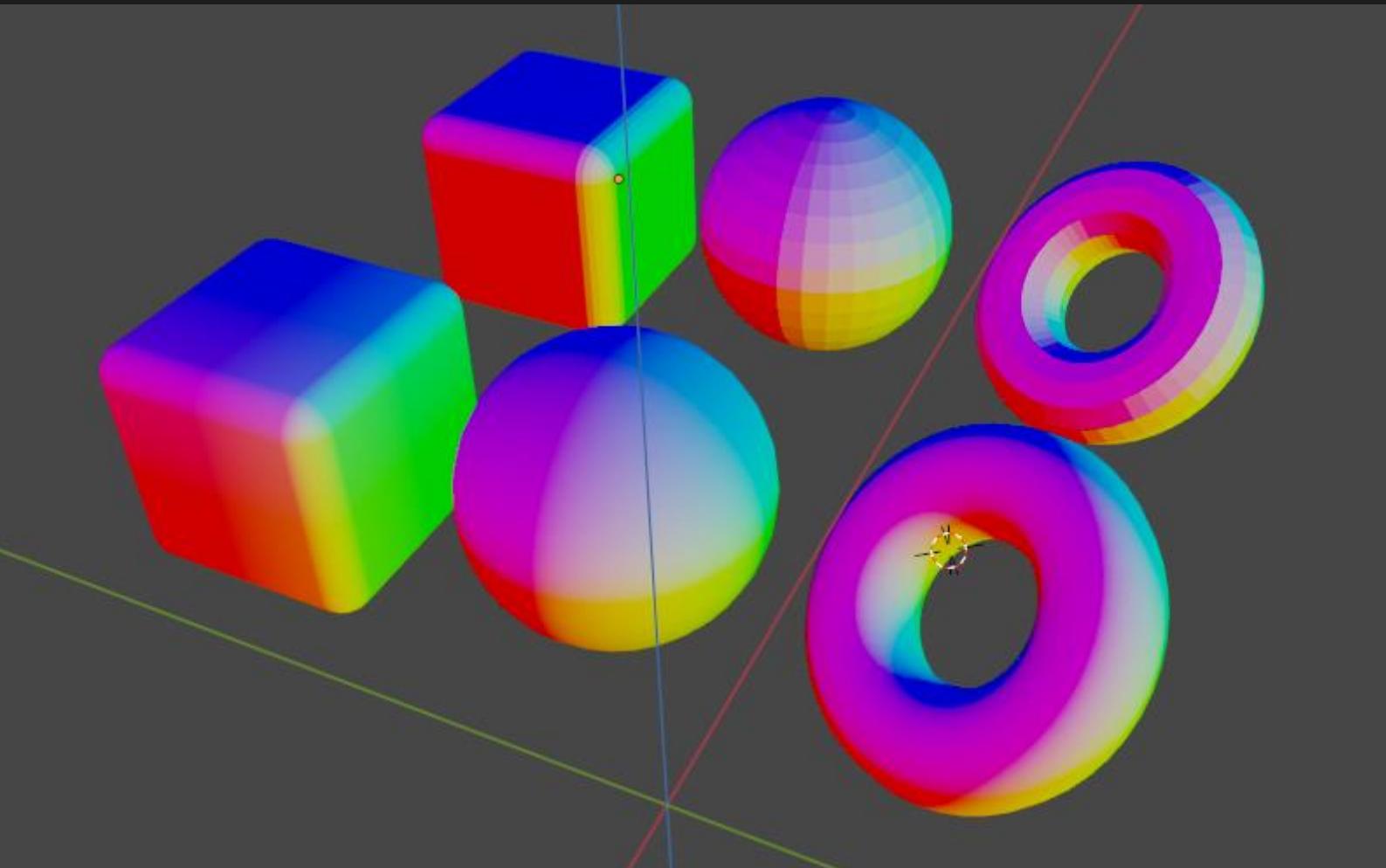
| | Generated | Object |
|--------------|---|------------------------------|
| 중심의 위치 | 바운딩박스 모서리 | 오리진 |
| 스케일 | 물체의 크기에 따라 찌그러짐 | 오브젝트 스케일을 따름 |
| 변형(modifier) | 변형되기 전 원본좌표를 유지 (Geometry node 제외) | 물체가 변형되면 텍스쳐가 이동할 수 있음 |

Normal

물체 표면이 바깥을 향하는 '방향'

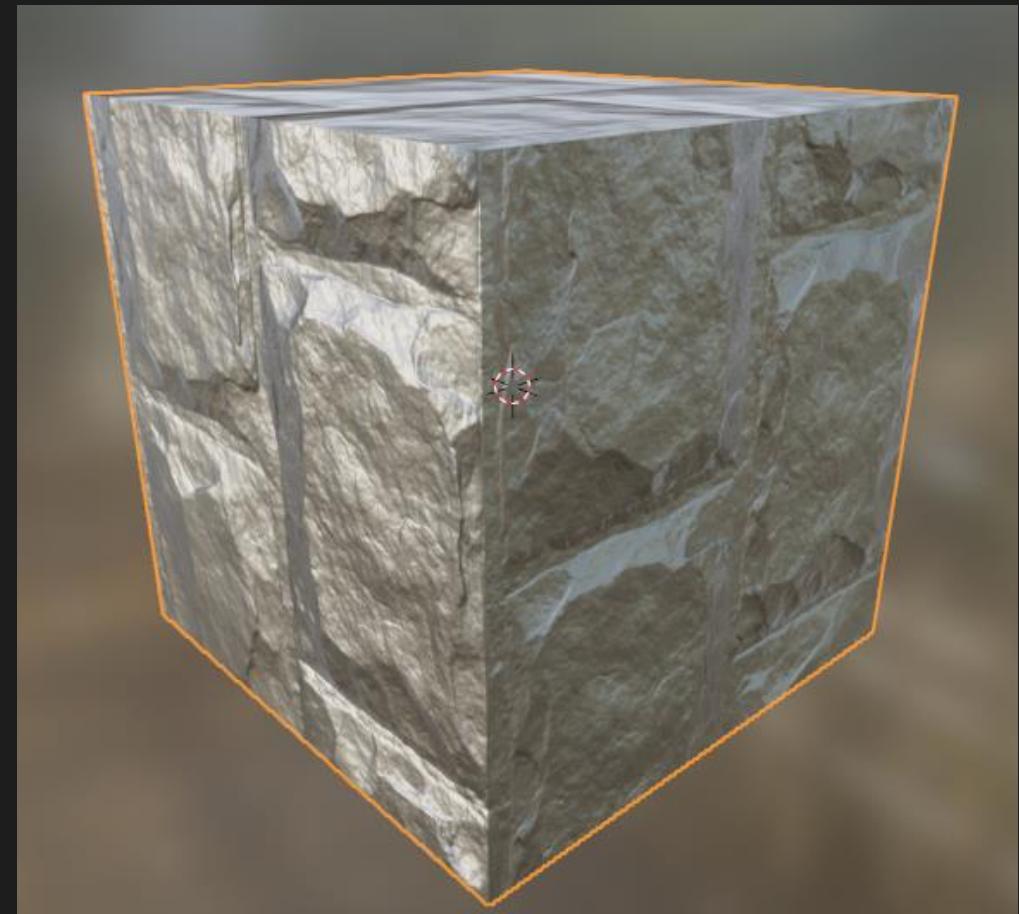
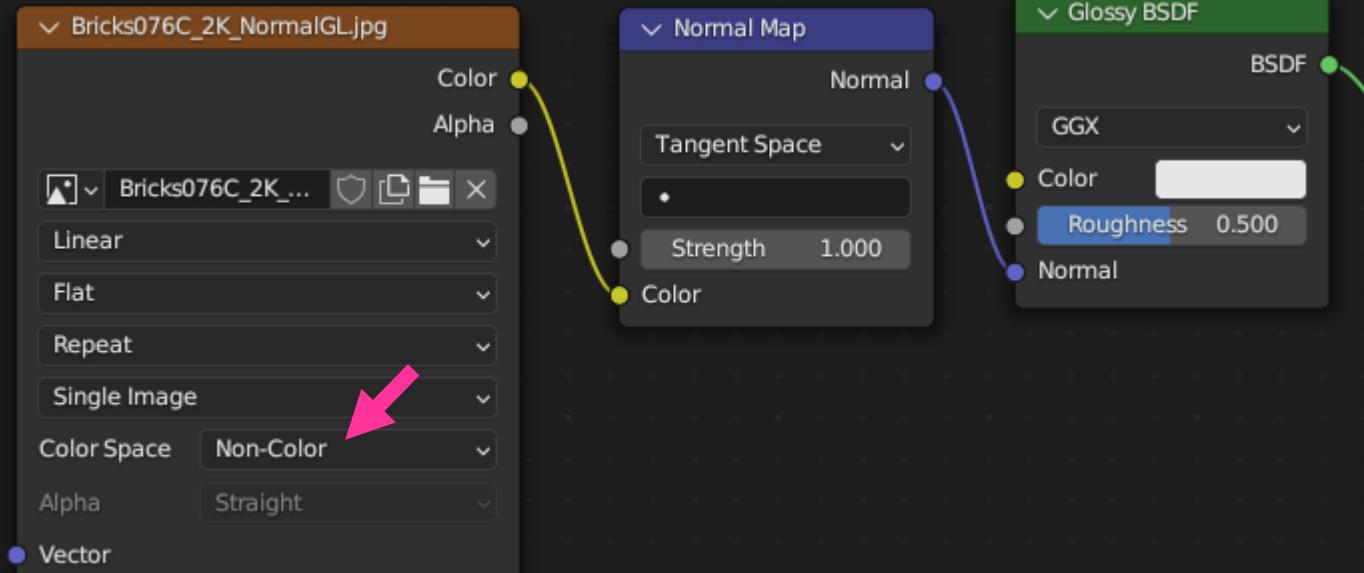
- 항상 크기는 1입니다.
- Shade smooth vs Shade flat
- Weighted Normal

- 실제 면의 방향과 일치할 필요가 없고, 그것이 중요한 포인트 중 하나입니다.



Normal Map

-Bump 노드와 Normal Map으로 변형시킬 수 있습니다.



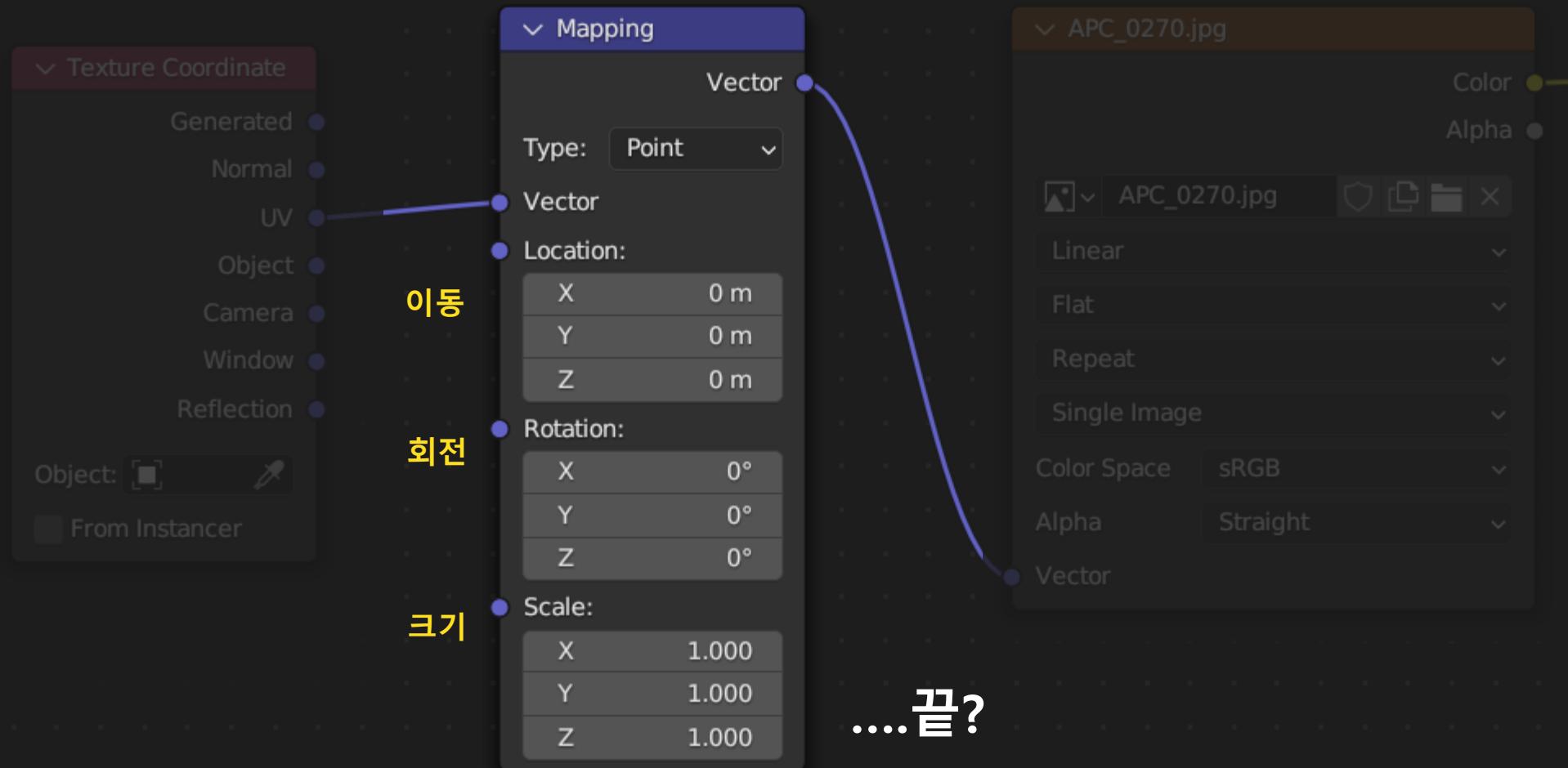
004강 매핑 노드와 좌표의 컨트롤

텍스쳐 좌표 이동, 회전, 스케일
좌표 이동을 이용한 노드 애니메이션
Mix 노드 소개

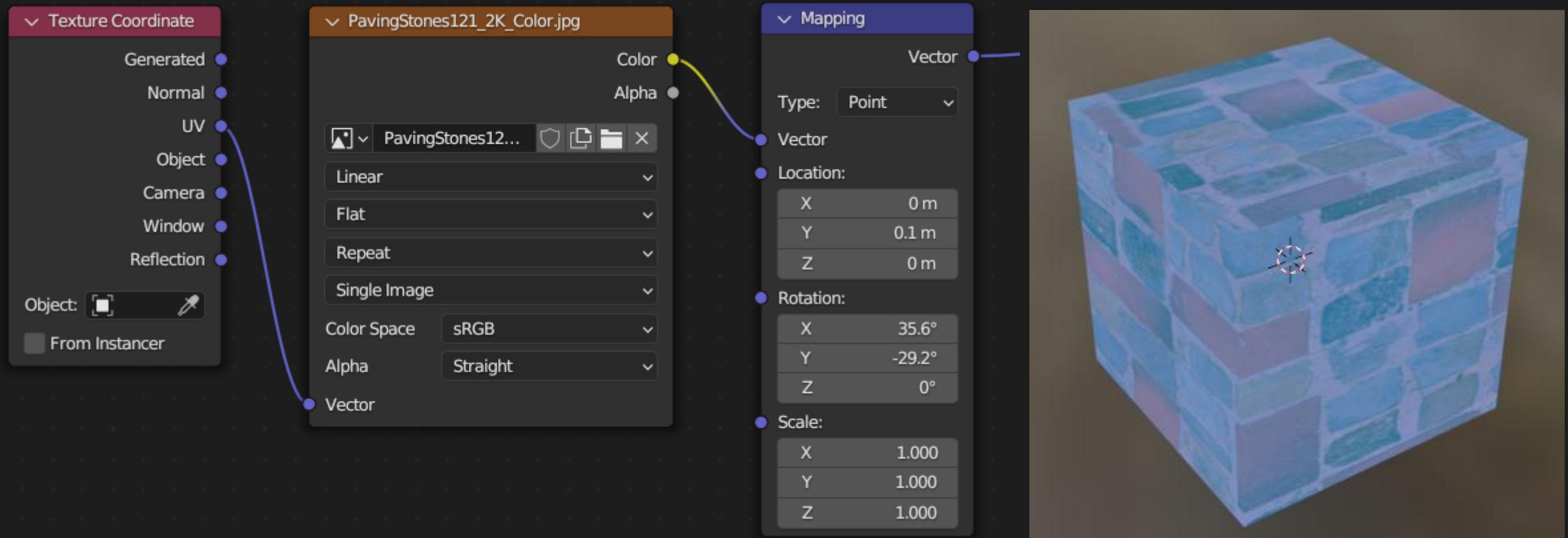


Mapping 노드

좌표를 움직입니다

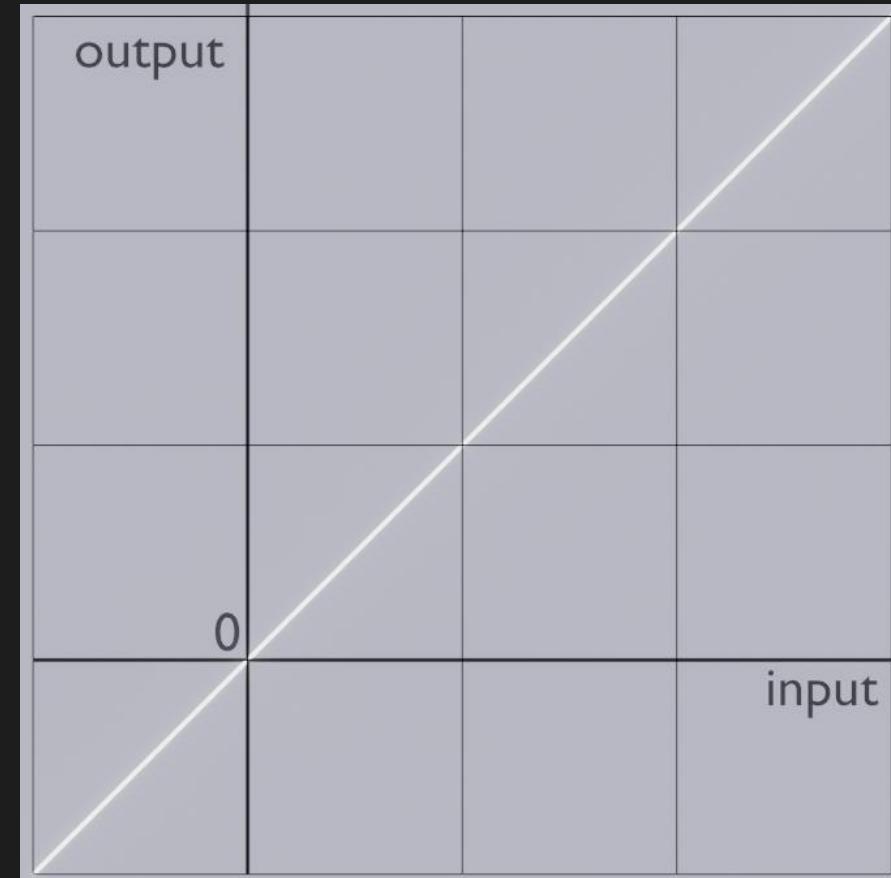
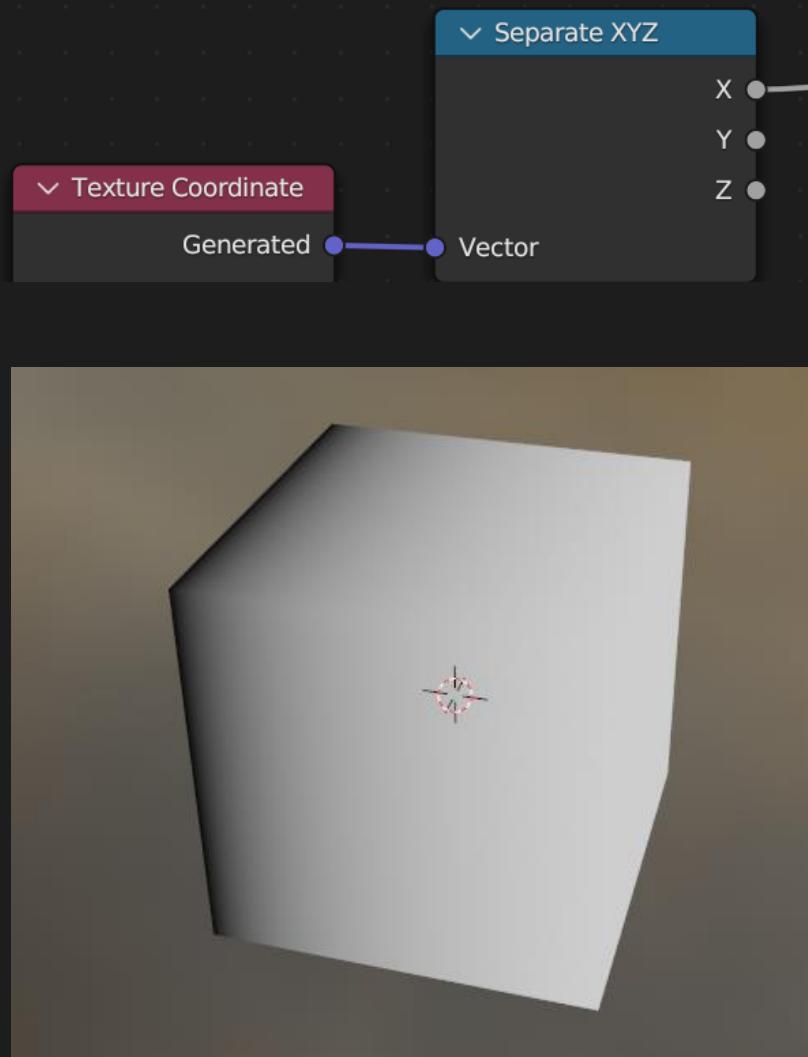


Mapping 노드



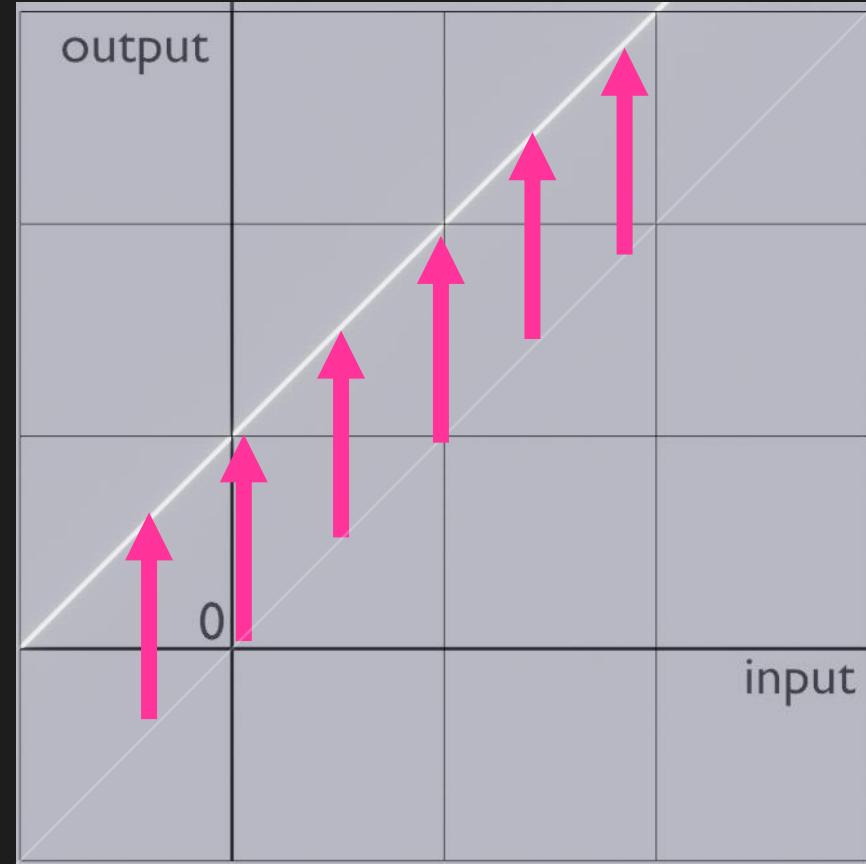
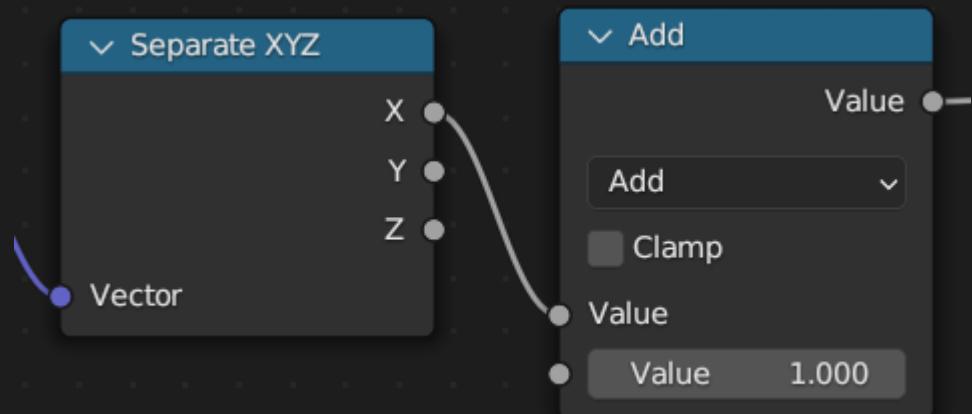
???

이동의 원리



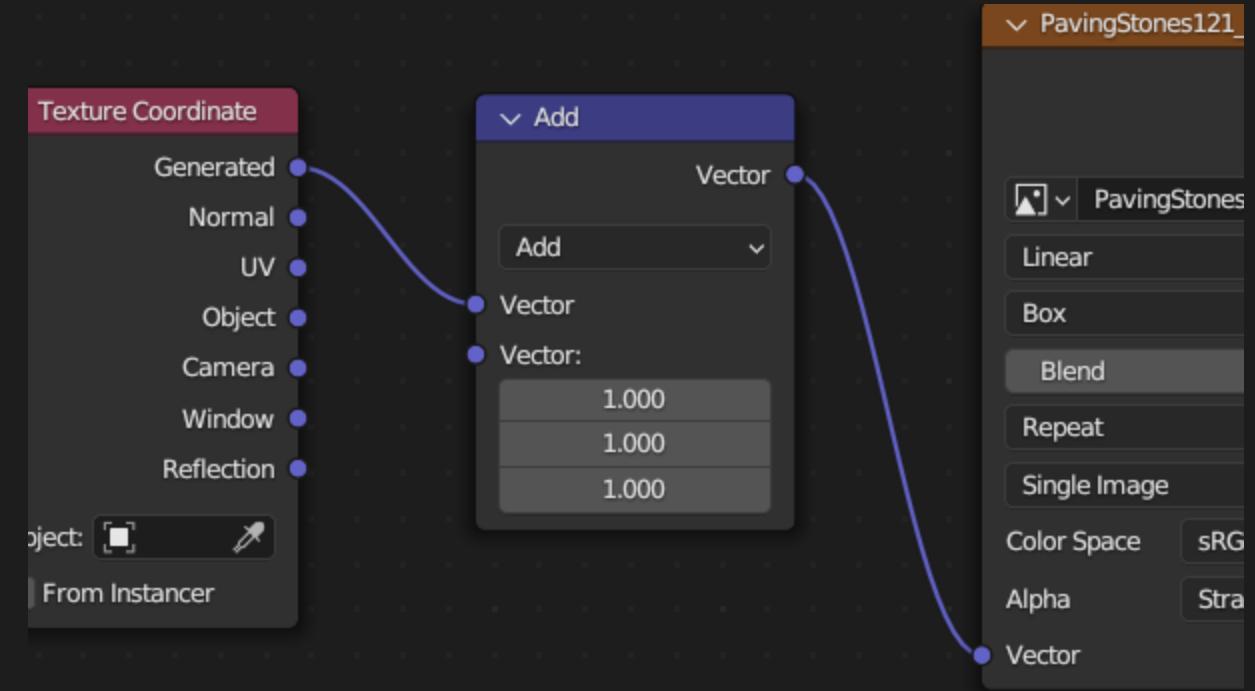
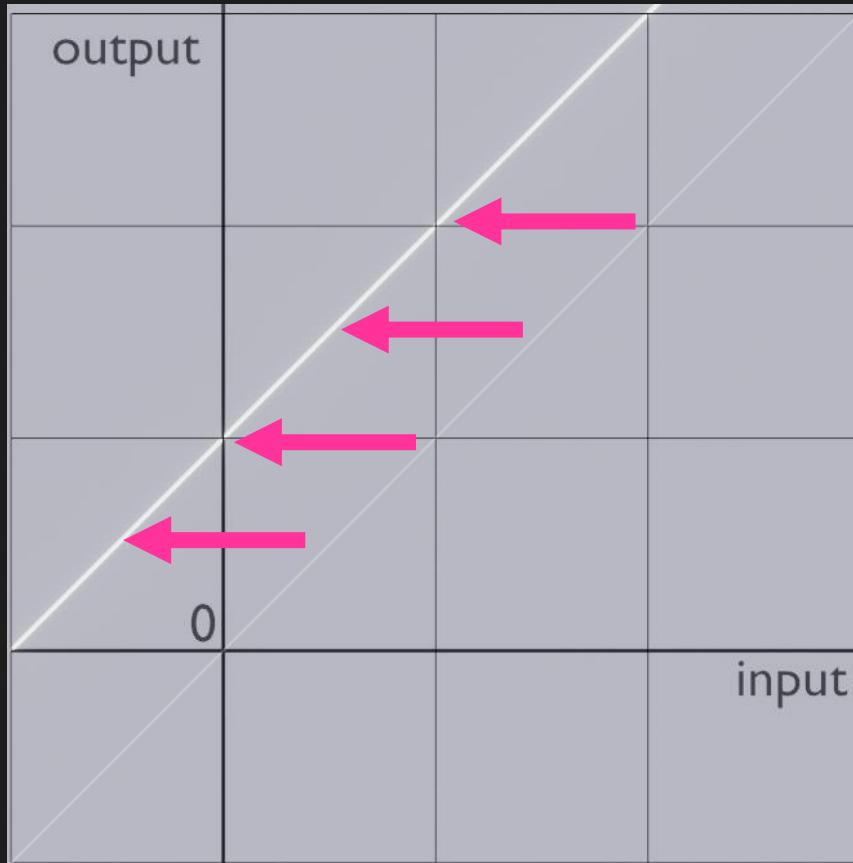
이동의 원리

각각의 점에 1을 더한다는 건..



이동의 원리

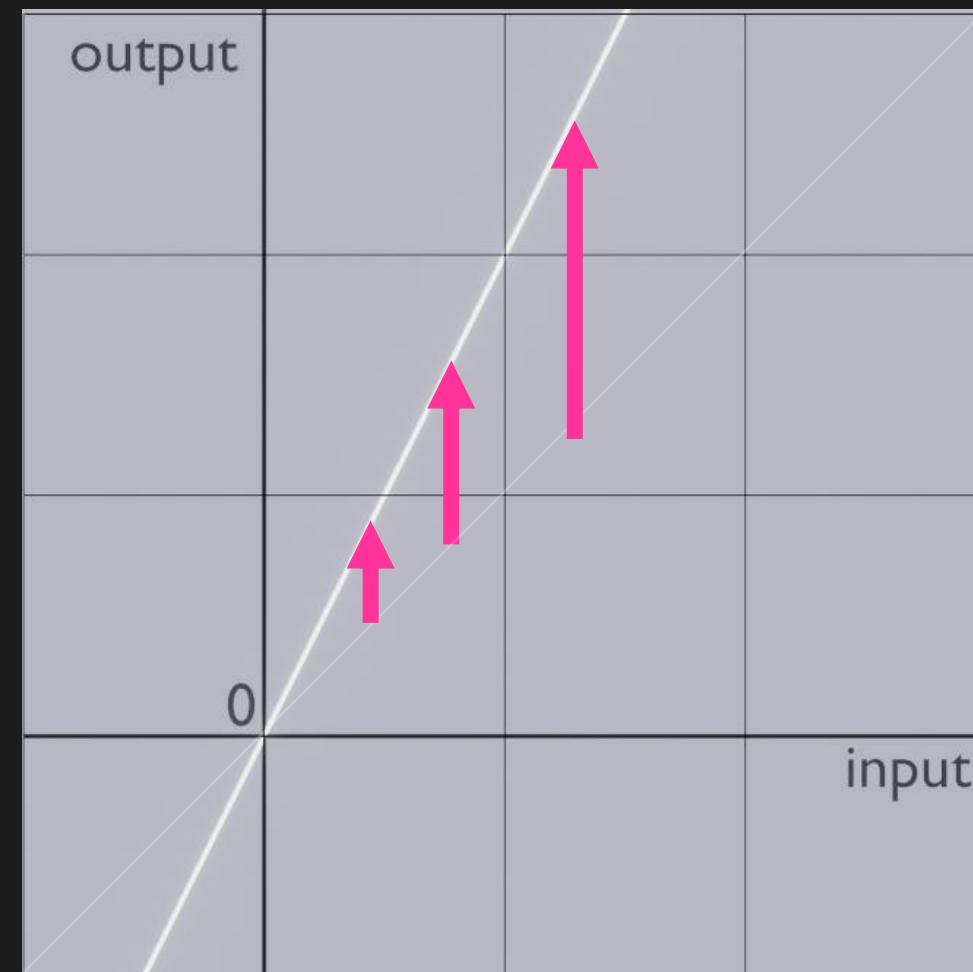
그래프를 왼쪽으로 1 이동시키는 효과입니다.



3차원은 머릿속에서 상상하기 힘들지만,
2차원일 때와 같은 원리입니다.

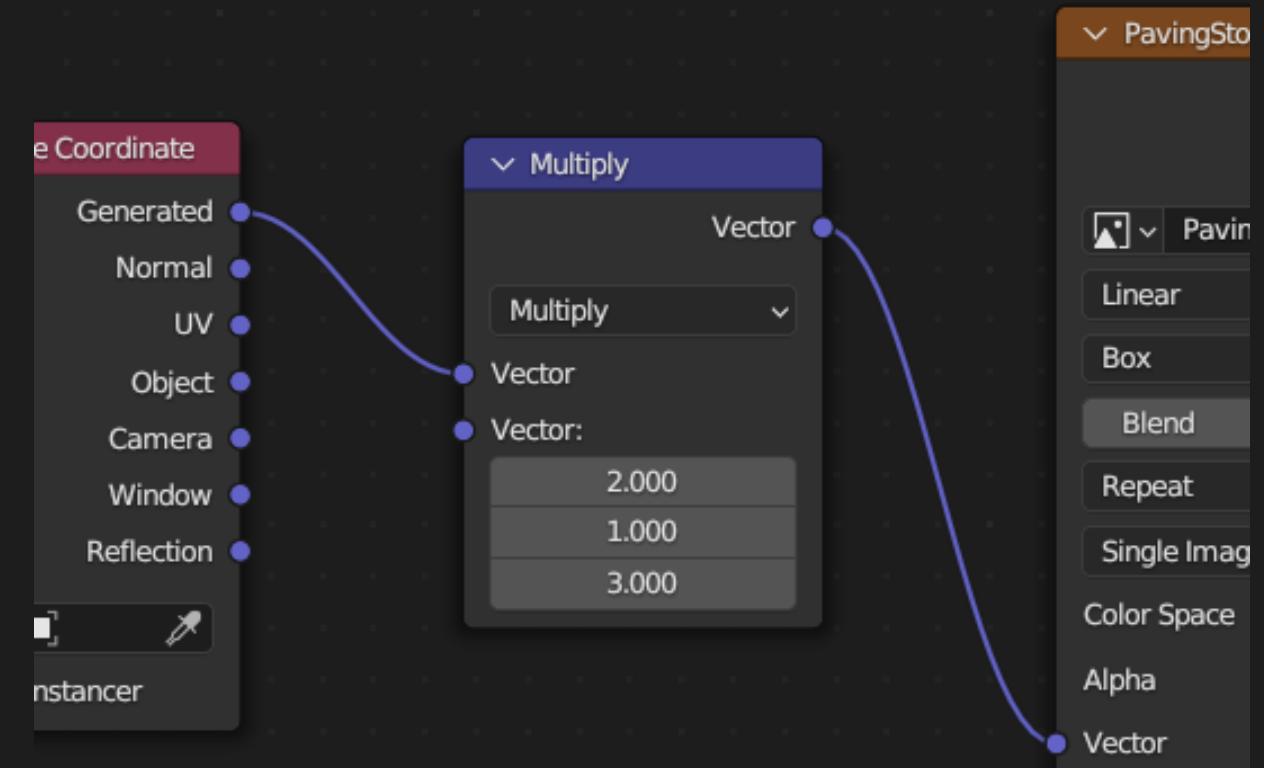
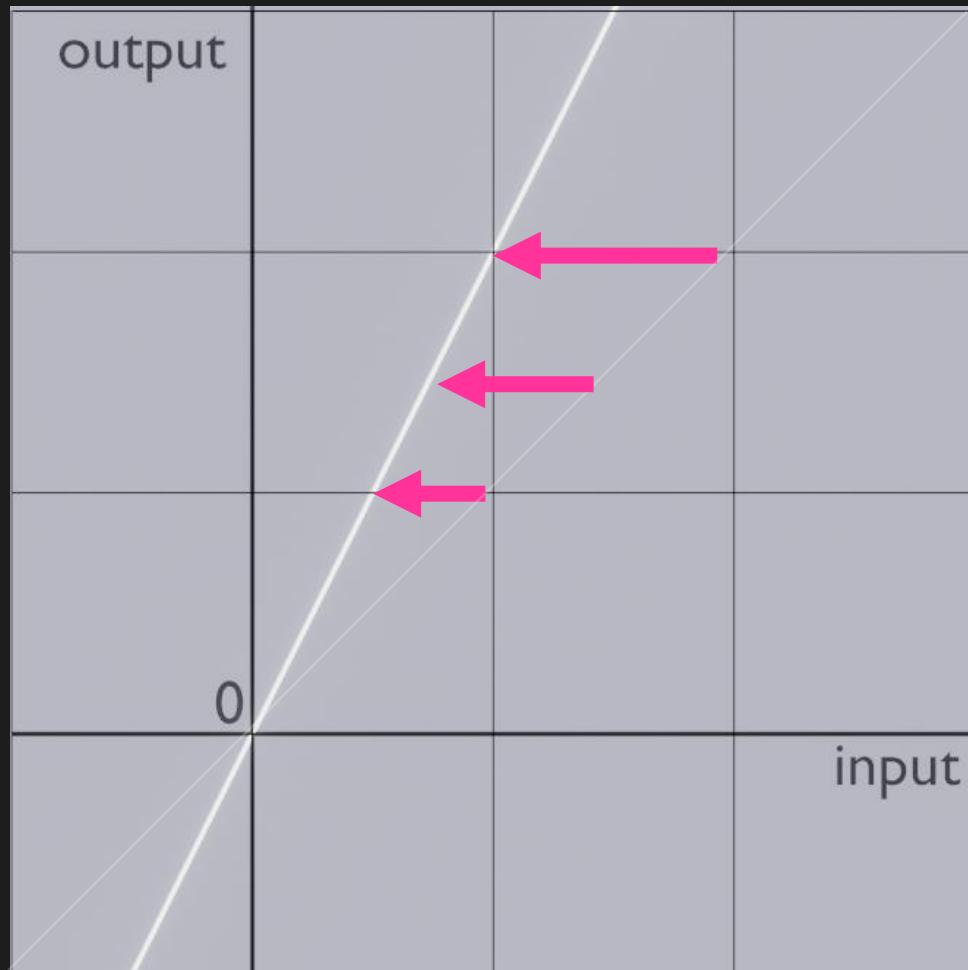
확대/축소의 원리

2를 곱해서 그래프의 키를 키우는 건..



확대/축소의 원리

그래프를 짜부러트리는 효과와 같습니다.



큰 값을 곱하면, 변화가 빨라집니다.
즉, 스케일이 축소됩니다.

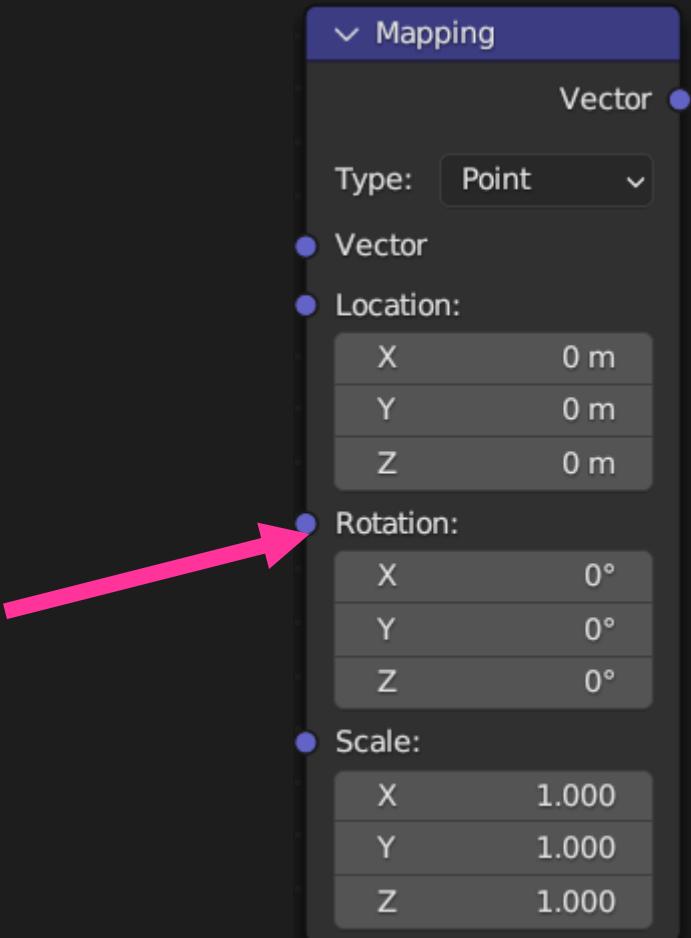
회전의 방정식?

$$x \rightarrow x \cos \theta - y \sin \theta$$

$$y \rightarrow x \sin \theta + y \cos \theta$$

(이건 2차원 회전.

3차원 회전은 더 복잡합니다)



체크포인트

-셰이더 노드의 연산은 각 점마다 개별적으로 이루어집니다.

즉, 어떤 점의 데이터를 다른 점에 전달하는 것은 불가능합니다!

-덧셈과 곱셈이 갖는 의미는 좌표 개념을 넘어서는 개념입니다.

즉, 덧셈이 가지는 이동의 의미, 곱셈이 가지는 확대/축소의 의미는 범용적으로 활용 가능합니다.
상황에 따라 '이동' 이 가지는 뜻이 바뀔 뿐입니다.

회전과 Box Mapping

Box Mapping은 Bounding Box를 기준으로 이루어집니다.
그런데 mapping 노드로 텍스쳐 좌표를 회전시켜도 Bounding Box는 회전하지 않습니다.
이 괴리 때문에 회전만큼은 2차원 이미지의 Box Mapping에서는 잘 작동하지 않습니다.

Mix Shader



두 Shader를 섞습니다.

Factor가 작을수록 위쪽, 클수록 아래쪽이 나옵니다.

005강 Mix Node 사용법

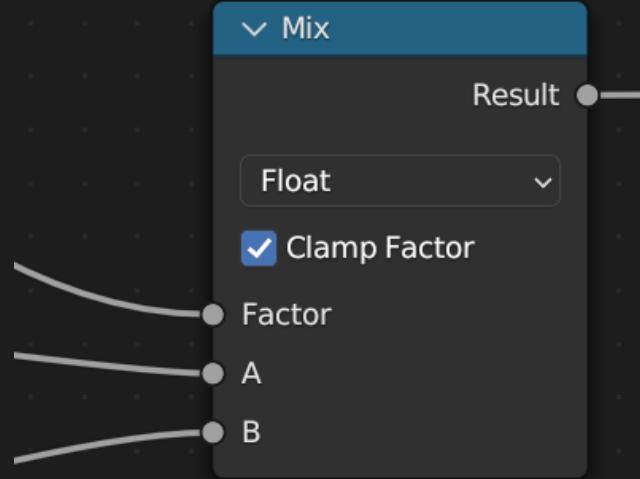
셰이더 노드 단축키와 제스쳐
Mix 노드의 각 블렌딩 모드 사용법

노드 에디터의 단축키와 제스쳐

| 단축키와 제스쳐 | 설명 |
|------------------------|---------------|
| N | Context 메뉴 |
| Shift+A | 노드 생성 메뉴 |
| Ctrl+X | 연결을 끊지 않고 삭제 |
| Alt+드래그 | 연결을 끊지 않고 빼내기 |
| m | 노드 끄기 |
| Shift+D | 노드 복제 |
| Ctrl+Shift+D | 연결을 유지한 채로 복제 |
| Ctrl+Shift+클릭 | 뷰어로 보기 |
| Ctrl+T | Texture 연결 |
| Ctrl+마우스 오른쪽 버튼 드래그 | 연결 끊기 |
| Ctrl+Alt+마우스 오른쪽 드래그 | 연결 '끄기' |
| Shift+마우스 오른쪽 드래그 | 매듭 만들기 |
| Alt+마우스 오른쪽 드래그 | Lazy Connect |
| Shift+Alt+마우스 오른쪽 드래그 | Connect |
| Ctrl+Shift+마우스 오른쪽 드래그 | Lazy Mix |

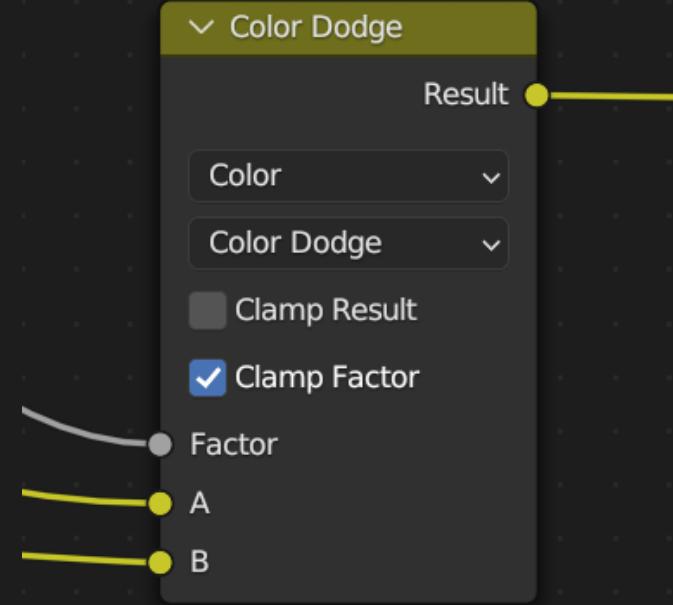
| 단축키와 제스쳐 | 설명 |
|------------------|---|
| | Ctrl+G 그룹 만들기 |
| | Ctrl+J 프레임 만들기 |
| | Alt+P 노드 프레임에서 내보내기 |
| | F2 이름 지정 |
| 단축키는 아니지만 유용한 것들 | |
| | Ctrl+C, Ctrl+V 값의 복사/붙여넣기는 값을 드래그하지 않고, 마우스를 옮려놓는 것만으로 가능합니다. |
| | 수치 입력에 +,-,*,/ 수치 입력에서 계산 가능. |
| | 수치 입력에 pi, e, tau 파이, 자연상수 입력 가능. |
| | 슬라이더를 넘어선 입력 직접 입력하면 슬라이더를 넘어선 값을 입력할 수 있습니다. |
| | 상단메뉴 View - Auto Offset 자동으로 거리를 넓혀주는 기능 On/Off |
| | 상단 메뉴 자석버튼 격자에 맞춰 이동 |

Mix 노드 Linear Interpolation, 혹은 Layer Blending Mode



A 와 B 사이의 가중평균, 혹은 Linear Interpolation. (Lerp)

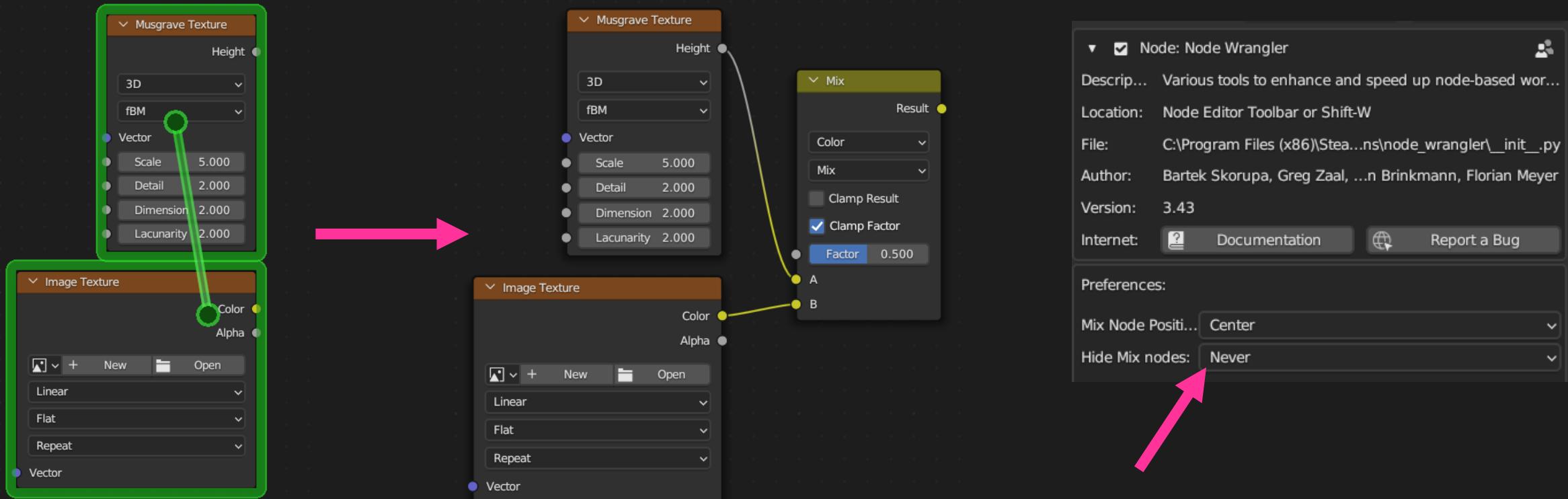
$$(1 - Fac)A + Fac \cdot B$$



Color 모드의 경우 블렌딩 모드도 지원.

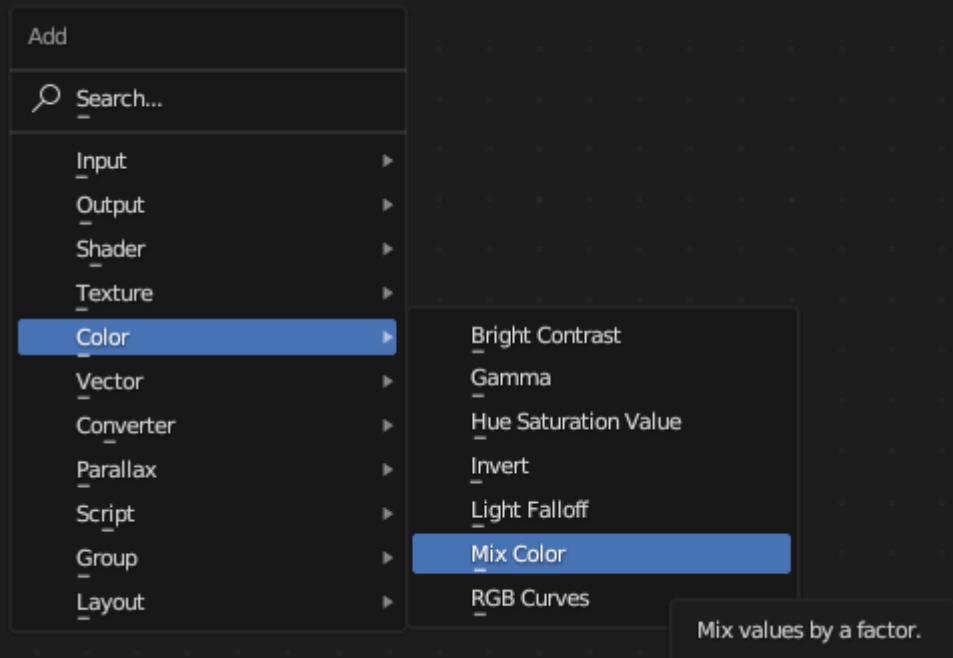
Mix 노드 빠른 접근

Lazy Mix : Ctrl+Shift+마우스 오른쪽 버튼 드래그

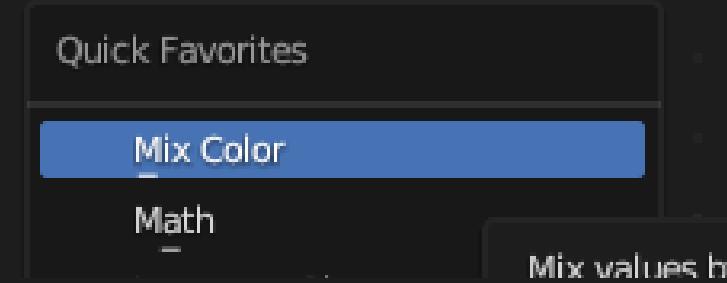


Mix 노드 빠른 접근

Mix Color는 Mix노드를 Color모드 상태로 생성합니다.



.....어느 방법도 불편하다면 Quick Favorites에 넣습니다.

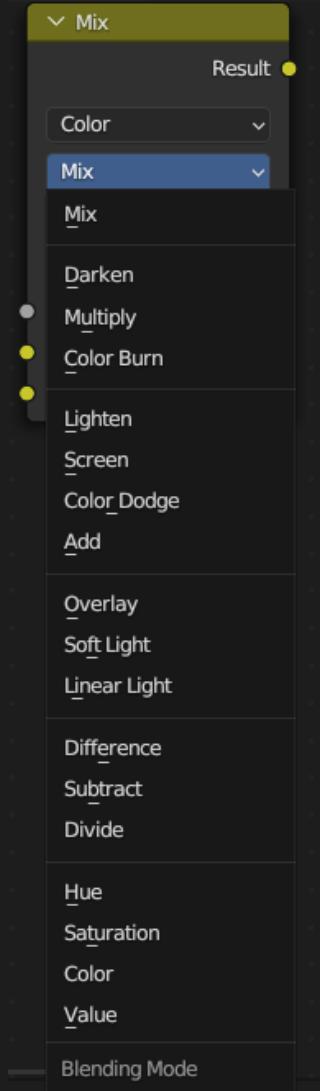


Mix 노드

The table below provides a detailed overview of the blending modes available in the Mix node, categorized by their visual effect.

| | Blending Mode | 식 | 설명 |
|-----|---------------|---|--|
| 어둡게 | Darken | $\min(A, B)$ | 둘 중 어두운 색을 출력합니다. |
| | Multiply | $A \times B$ | A위에 B의 톤을 올린 것처럼 어두워집니다. |
| | Color Burn | $1 - (1 - A) / B$ | B를 이용하여 A를 어둡게 태웁니다. 명도는 낮아지고 채도는 높아집니다. |
| 밝게 | Lighten | $\max(A, B)$ | 둘중 밝은 색을 출력합니다. |
| | Screen | $1 - (1 - B)(1 - A)$ | Multiply 와 반대로, A위에 B를 스크린으로 쓴 것처럼 밝아집니다. |
| | Color Dodge | $A / (1 - B)$ | B를 이용하여 A를 밝게 비춥니다. 명도는 높아지고 채도는 낮아집니다. |
| 대조 | Add | $A + B$ | A와 B의 단순 합. 결과적으로 밝아집니다. |
| | Overlay | ($A \geq 0.5$ 일 때) $1 - 2(1 - A)(1 - B)$ ($A < 0.5$ 일 때) $2AB$ | A가 밝을 때는 Screen, 어두울때는 Multiply와 비슷하게 작동합니다. |
| | Soft Light | $(1 - 2B)A^2 + 2BA$ | Overlay보다 부드럽게 섞어줍니다. |
| | Linear Light | $A + 2B - 1$ | A에 B의 범위를 확장시켜($=2B-1$) 더합니다. Overlay보다 강하게 섞어줍니다. |

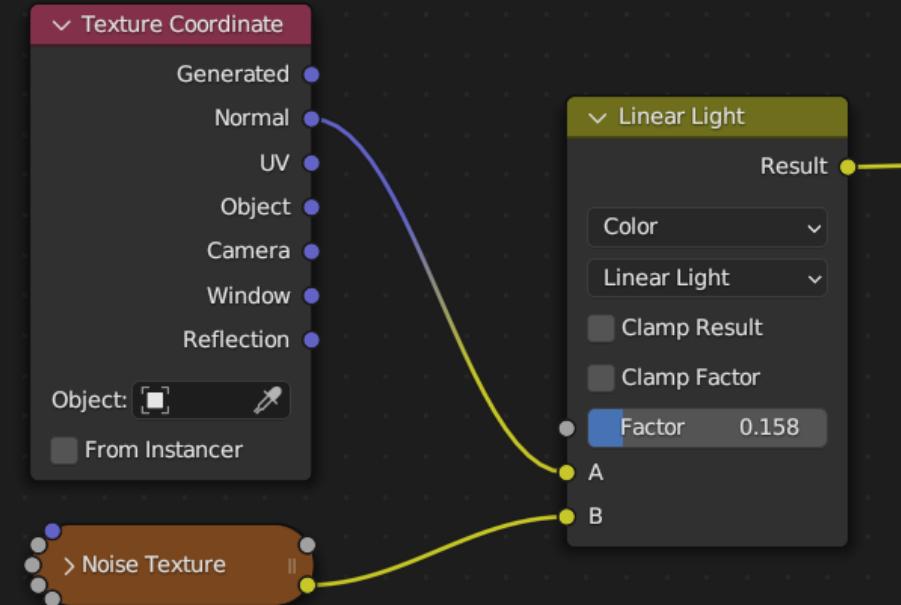
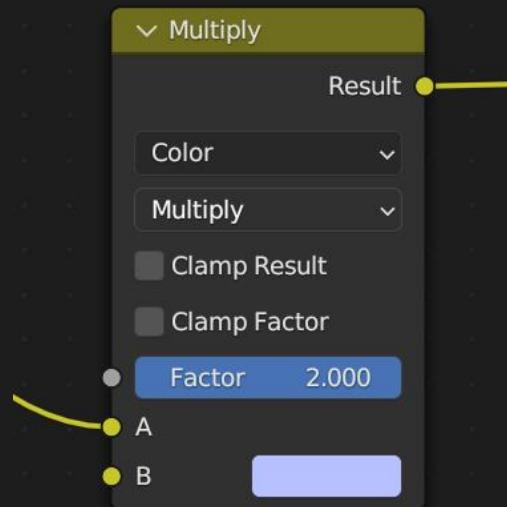
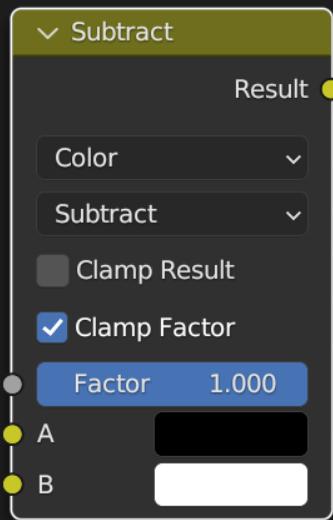
Mix 노드



| | Blending Mode | 식 | 설명 |
|----|---------------|-----------------------|---|
| 반전 | Difference | $ A-B $ | A와 B의 차이 (양수) |
| | Subtract | $A-B$ | A와 B의 차이. |
| | Exclusion | $A+B-2AB$ | A를 B만큼 반전 |
| | Divide | A/B | A에서 B를 나눕니다. Multiply 와 '수학적으로' 반대로 작동합니다. Multiply 가 이미지에 툰을 입혔다면, Divide 는 이미지의 툰을 제거합니다. |
| 색채 | Hue | $HSV(H(B),S(A),V(A))$ | A의 이미지에 B의 색상을 사용합니다. |
| | Saturation | $HSV(H(A),S(B),V(A))$ | A의 이미지에 B의 채도를 사용합니다. |
| | Color | $HSV(H(B),S(B),V(A))$ | A의 이미지에 B의 색상과 채도를 사용합니다. |
| | Value | $HSV(H(A),S(A),V(B))$ | A의 이미지에 B의 명도를 사용합니다. |

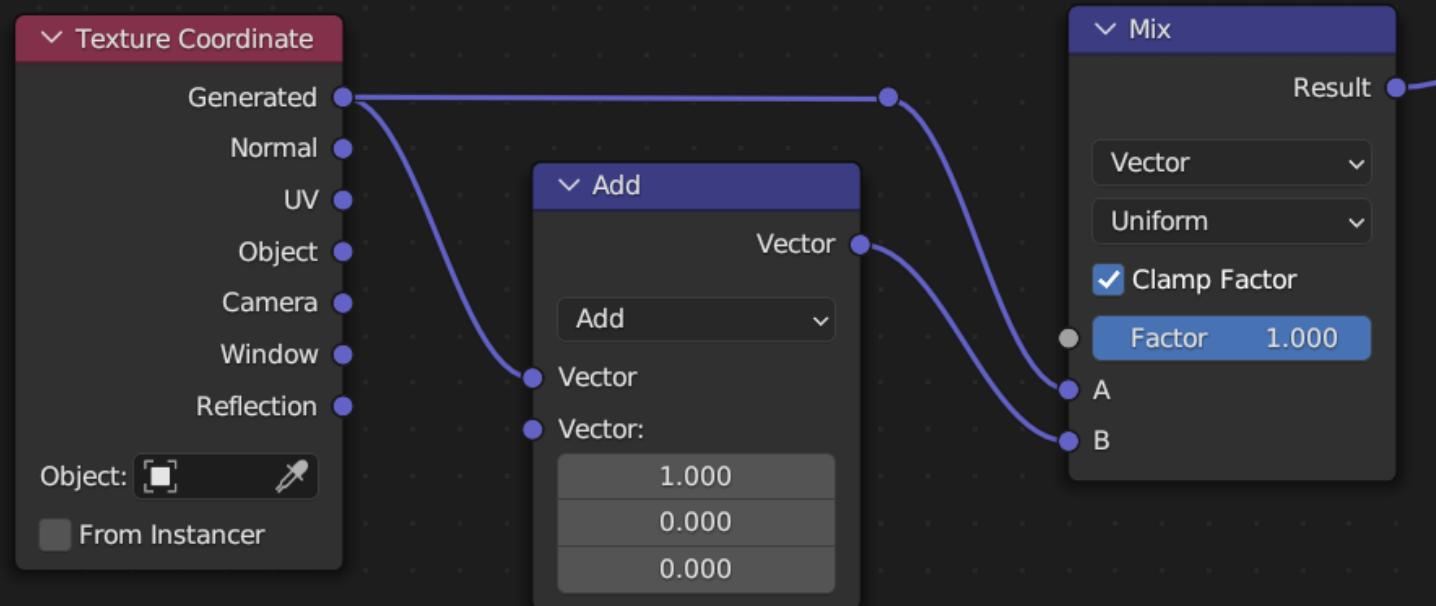
이것은 일러스트 프로그램이 아닙니다

더 자유롭습니다.



※ 범위를 초과하는 결과를 원치 않으면 Clamp를 활용합니다.

이것은 일러스트 프로그램이 아닙니다



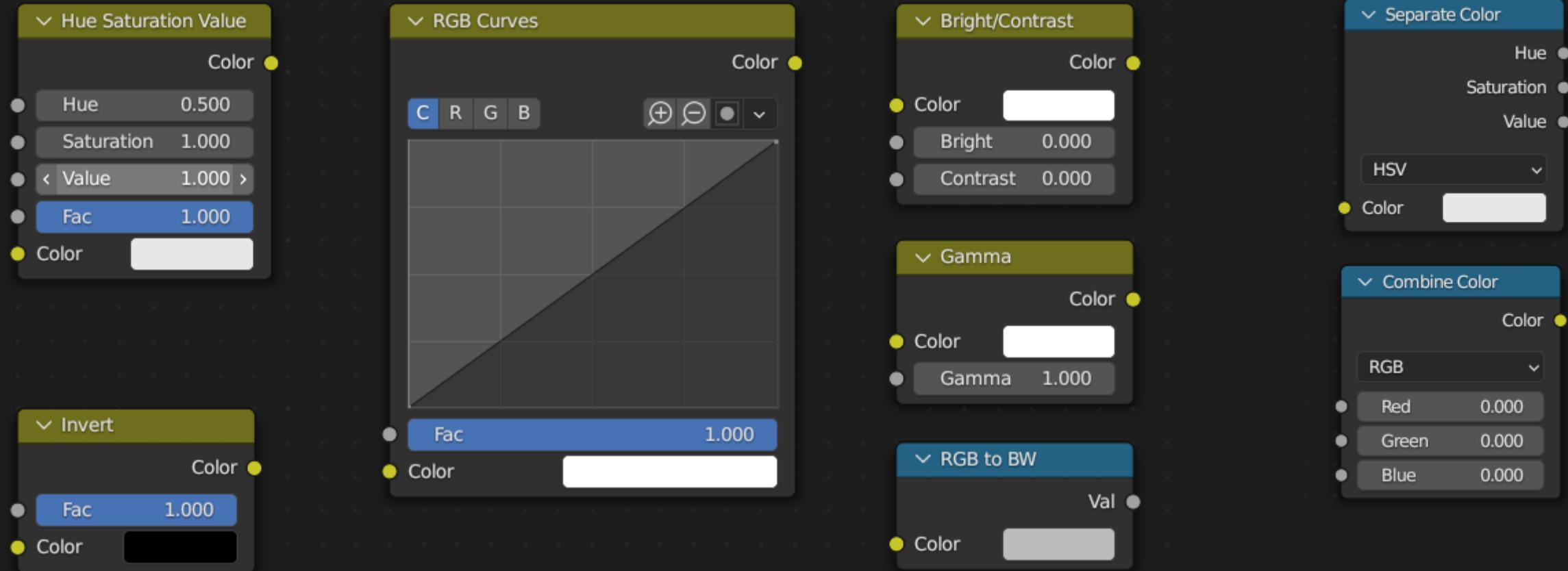
결과를 꼭 이미지로 상상할 필요는 없습니다.

006강 이미지의 편집

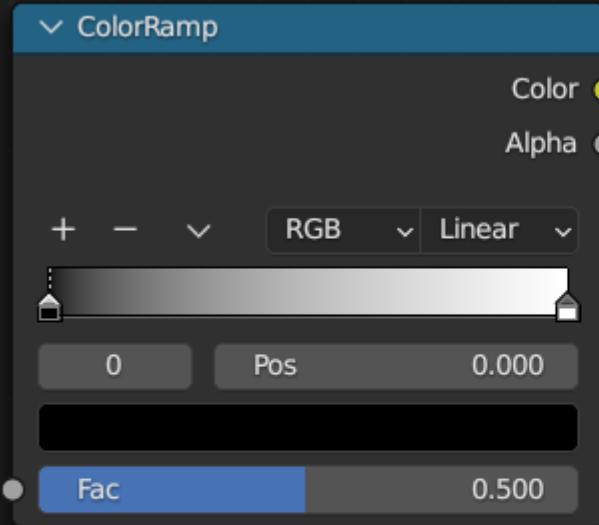
포토샵처럼 이미지를 수정하는 방법
감마 보정, Filmic Filter에 대한 이해
노드를 이용하여 빛 바랜 사진 만들기



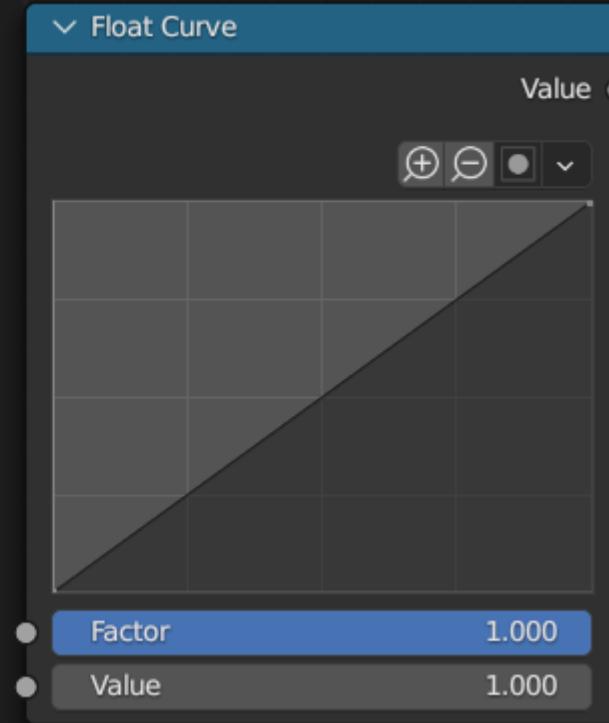
포토샵은 아니지만



ColorRamp

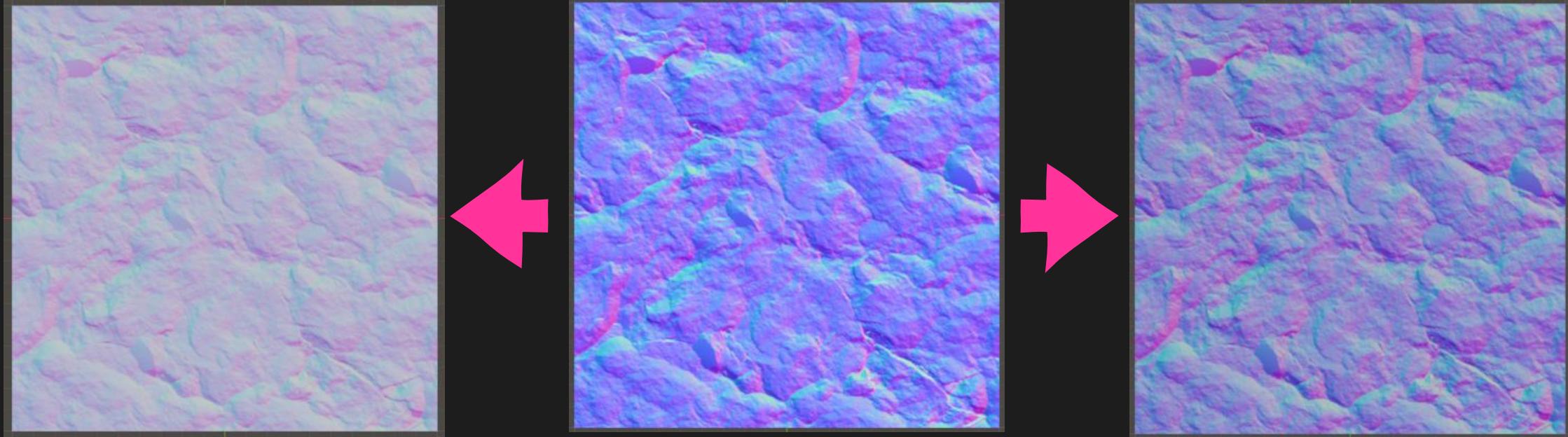


입력받은 명도(Factor)에 따라 색상을 대응시킵니다.
기본 보간 방법은 Linear 이지만 B-Spline을 사용하면
더 부드러운 변화도 가능합니다.
(이는 Procedural Texture 섹션에서 자세히 다루게 됩니다.)



흑백 이미지만 필요하다면 Float Curve 도 같은 역할이지만..
※입력값은 Factor가 아니라 Value에 꽂습니다.

색상 문제

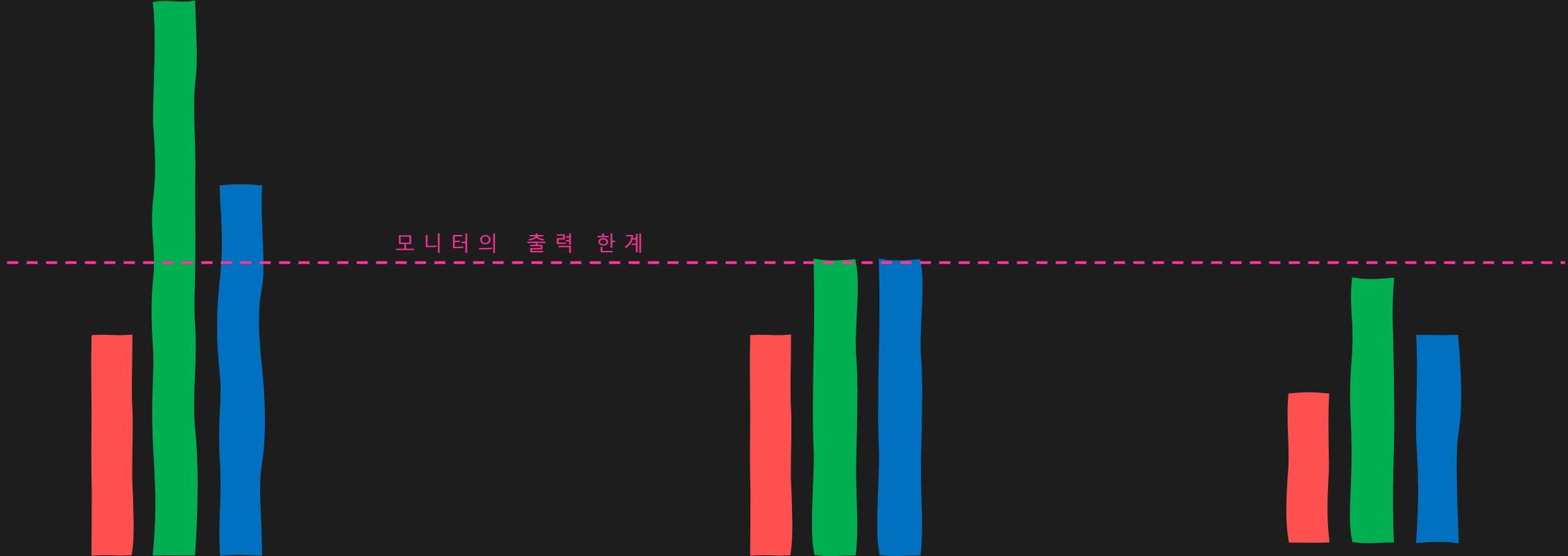


감마 보정

sRGB

Filmic Filter

Filmic View Transform



블렌더가 이미지를 만들 때,
밝기의 제약 없이 색을 생성합니다.

그대로 출력한다면, 1을 넘는 값을
구분할 수 없습니다.

자연스러운 이미지를 위해선,
전체적인 색상 변형이 필요

Filmic View Transform

View Transform 적용 전



View Transform 적용 후

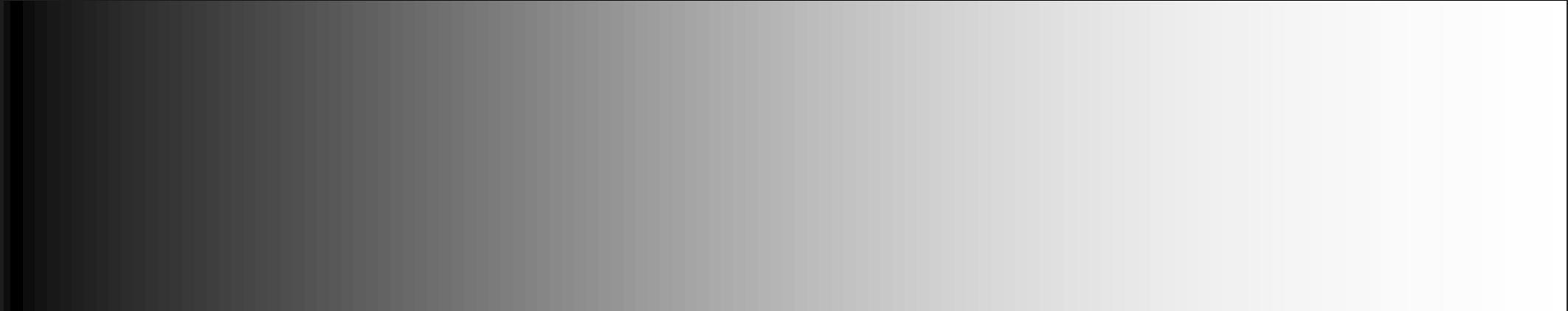


그에 따라, 0에서 1 사이의 밝기를 가진 이미지도 색상이 바뀌어,
미묘하지만 원본과 달라집니다. (흐리거나 탁하게 느껴질 수 있음.)

감마 보정

인간의 눈은 어두운 부분에 더 민감합니다.

이미지를 저장할 때도 어두운 부분의 정보를 더 많이 저장합니다.



그래서 이미지를 화면에 표시할 때는 저장한 값을 그대로 보여주지 않고 어떤 연산을 거쳐서 표시합니다.
그것이 감마보정입니다.

감마 보정



이미지 원본 데이터

감마보정 (2.2제곱)

되돌리고 싶으면,
1/2.2제곱을 해줍니다.

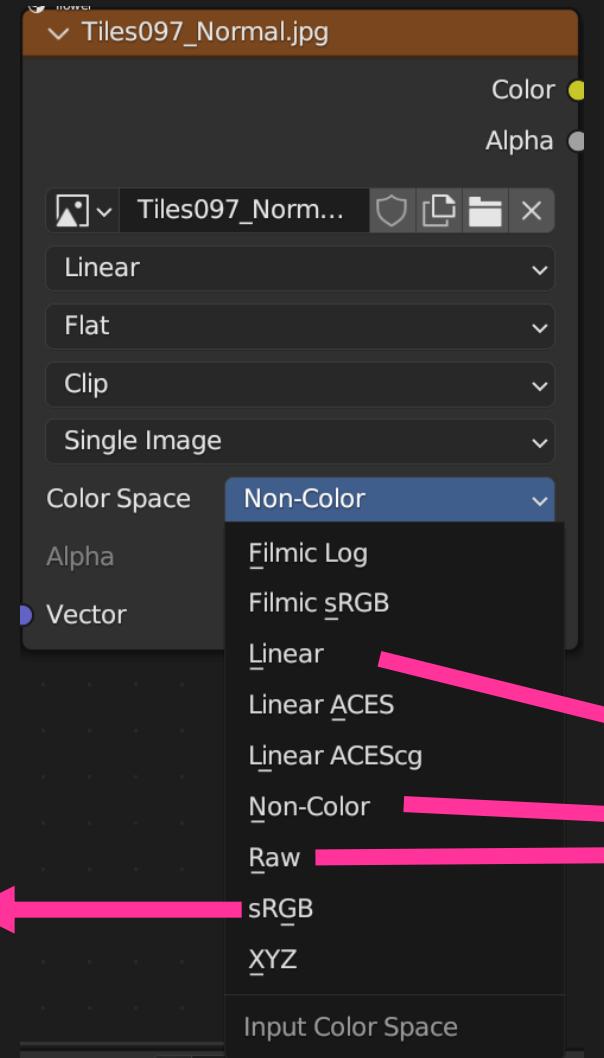


일반적인 모니터의 출력 이미지



원본 데이터를 그대로 색상에 대입한 것

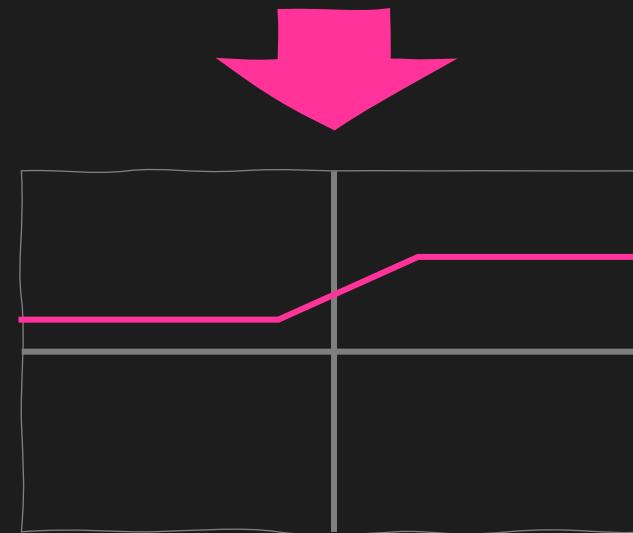
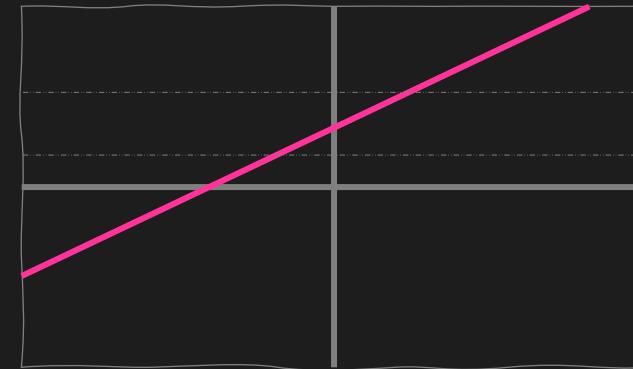
감마 보정



Clipping 문제

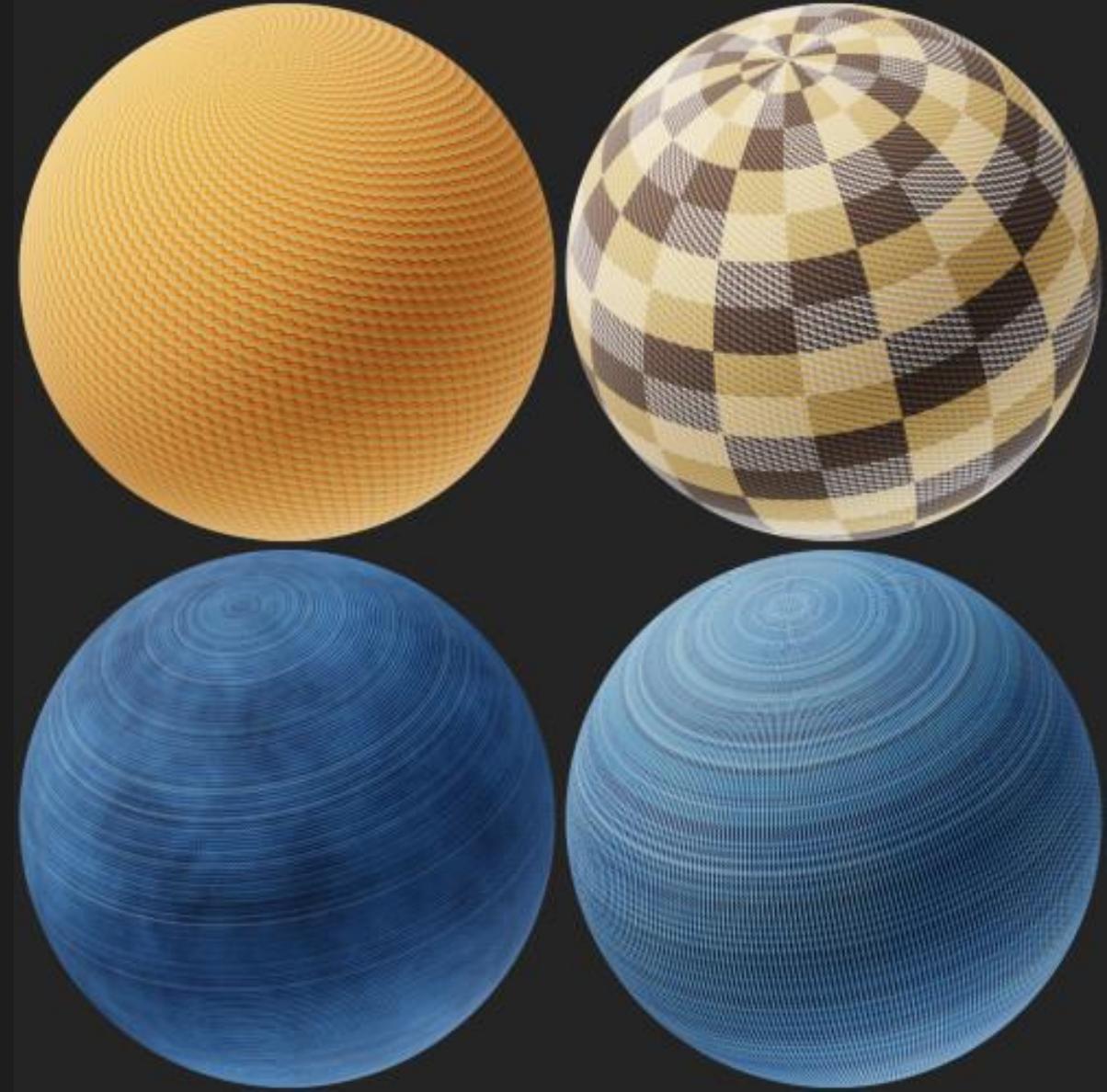
일반적으로, 연산 결과는 최대 최소값에 의해 제한되지 않습니다.
필요에 따라 직접 컨트롤합니다.

The image shows a node editor interface with two nodes: a 'Mix' node on the left and a 'Clamp' node on the right. The 'Mix' node has a 'Color' output and a 'Mix' input. It also has parameters for 'Factor' (set to 0.500), 'A' (yellow dot), and 'B' (yellow dot). A pink arrow points to the 'Clamp Result' checkbox, which is checked. The 'Clamp' node has a 'Min Max' dropdown set to 'Value'. Its parameters are 'Value' (1.000), 'Min' (0.000), and 'Max' (1.000).



007강 내장 텍스쳐

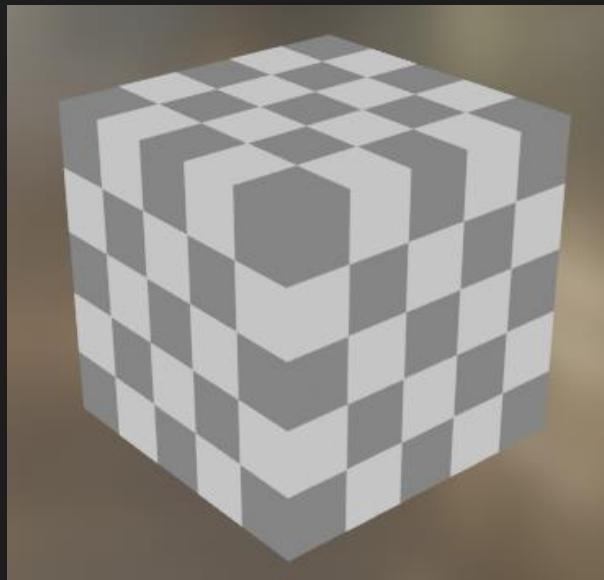
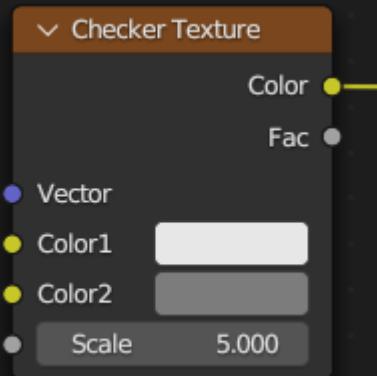
- 블렌더에 내장된 절차적 텍스쳐들
- 천 재질을 만들어봅시다



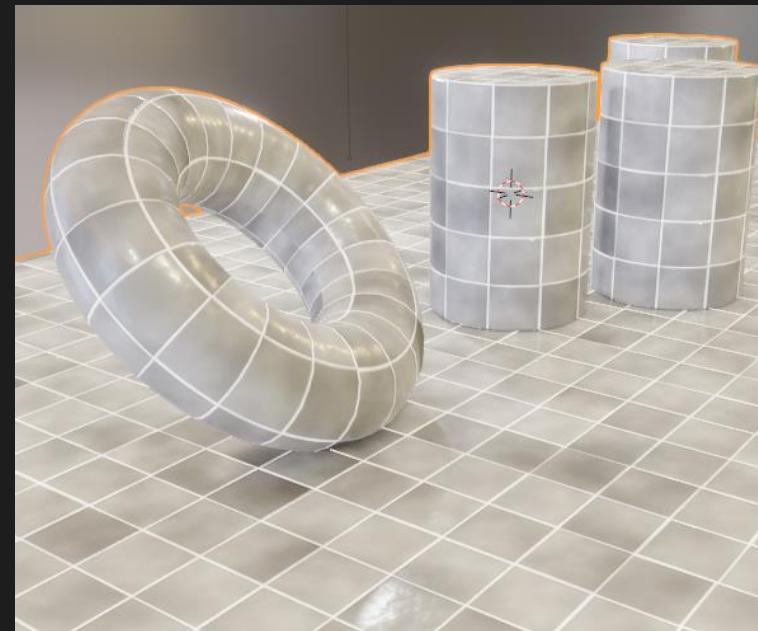
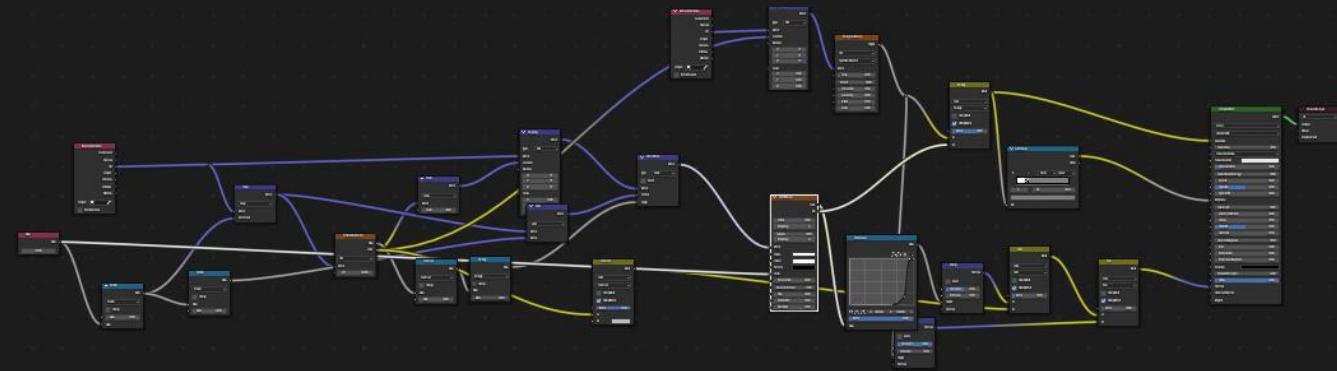
Procedural Texture

- 수학적 방법으로 생성된 텍스쳐

블렌더 기본 텍스쳐들



...혹은 그것들을 연결하여 만든 텍스쳐

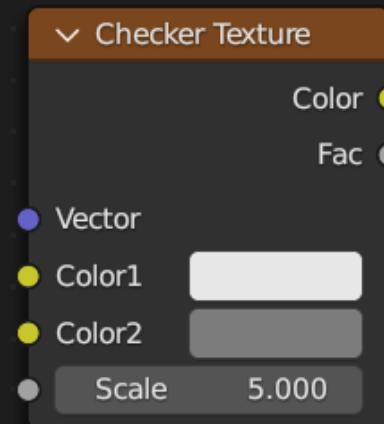


Brick Texture, Checker Texture



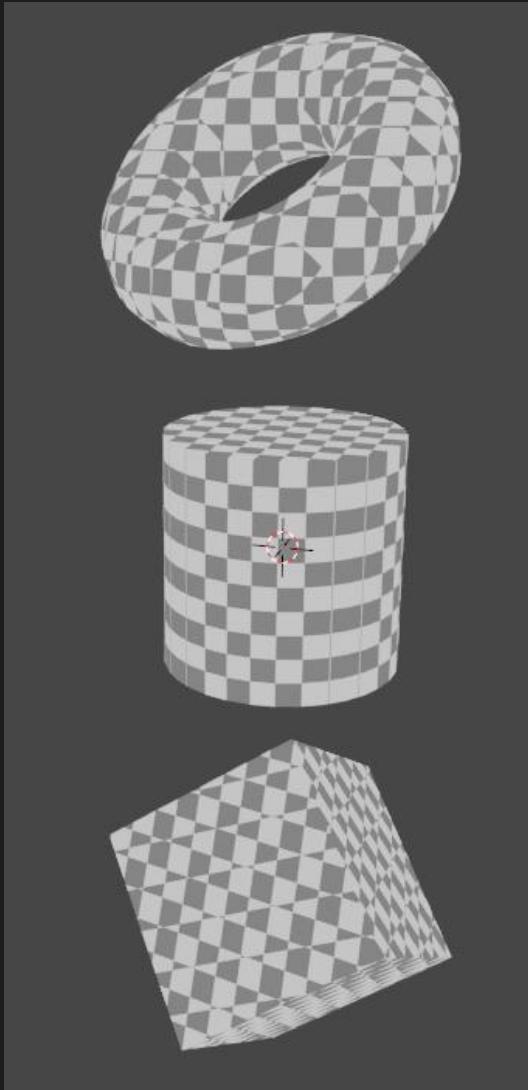
벽돌 무늬를 만듭니다.

Offset, Squash로 패턴을 조절할 수 있습니다.



체크 무늬를 만듭니다.

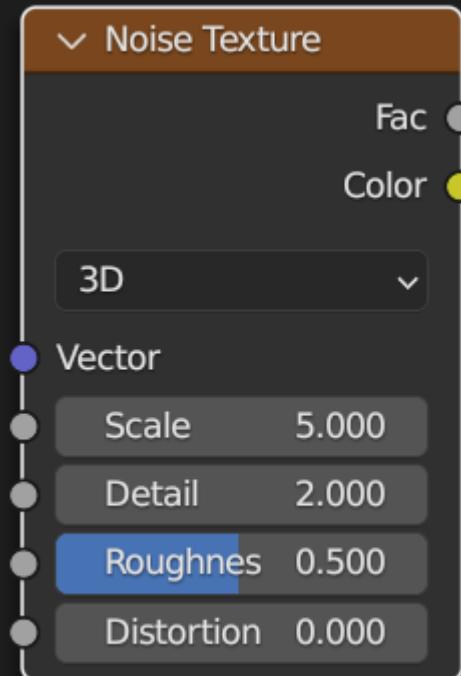
이것은 3차원 텍스쳐입니다



좌표의 3차원을 모두 사용합니다

3차원 공간에 연속적으로 존재하기 때문에,
UV – 이미지텍스쳐 사용시 생기는 Seam이 발생하지 않습니다
※하지만 필요에 따라 2차원으로도 사용할 수 있습니다.

Noise Texture

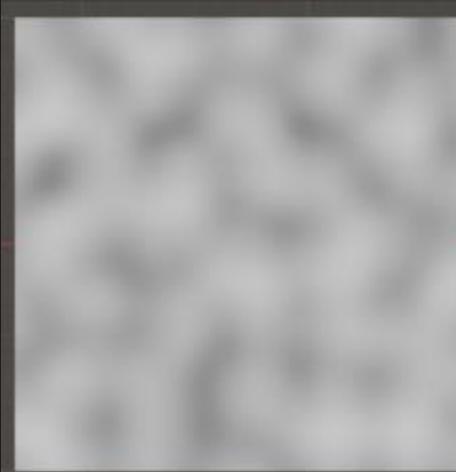


Detail : 1씩 올라갈 때마다 복잡도가 2배씩 증가합니다.

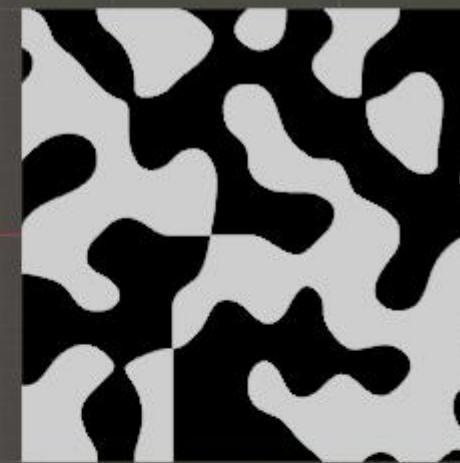
Roughness : 작은 노이즈 패턴을 얼마나 섞을 지 결정합니다.
0이 되면 Detail을 무시합니다.

텍스쳐를 눈으로 확인하는 여러 방법

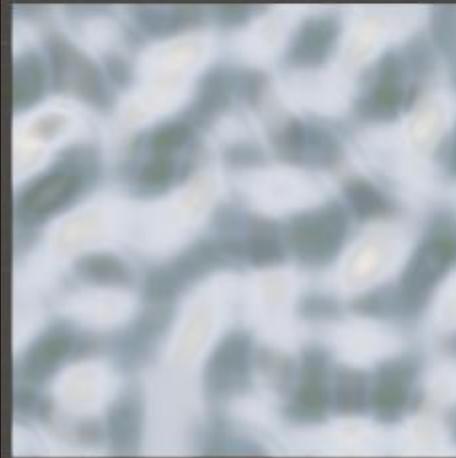
이미지 미리보기



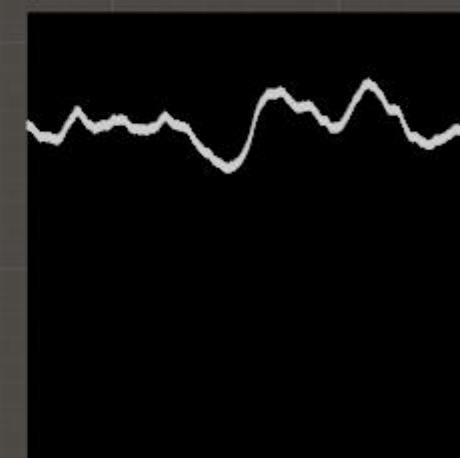
Greater than



Bump

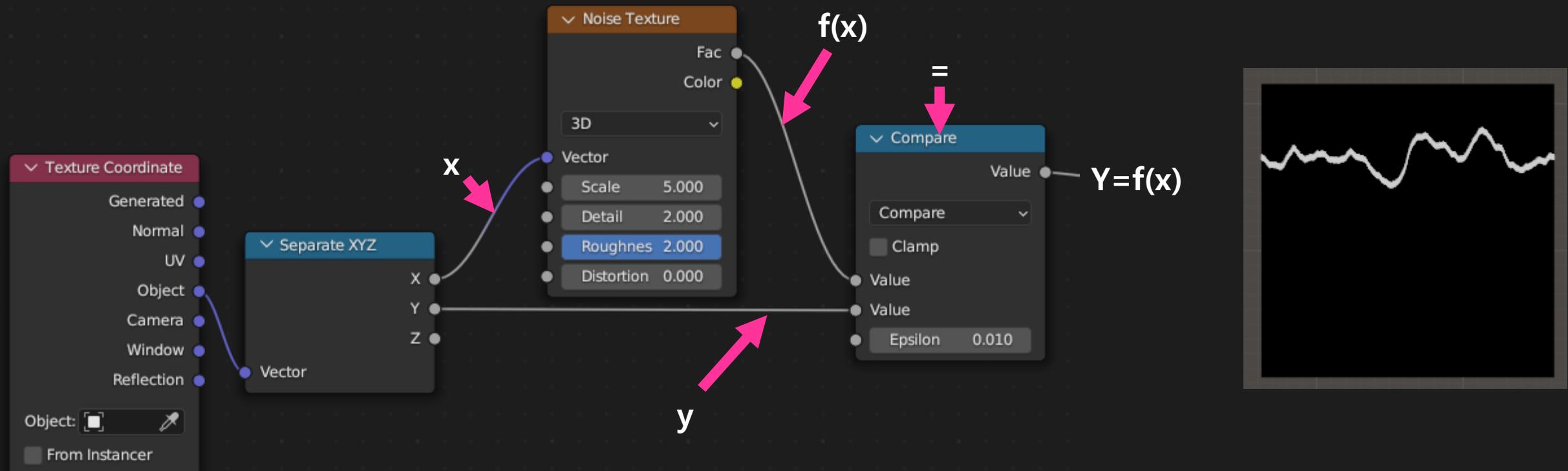


2차원 그래프

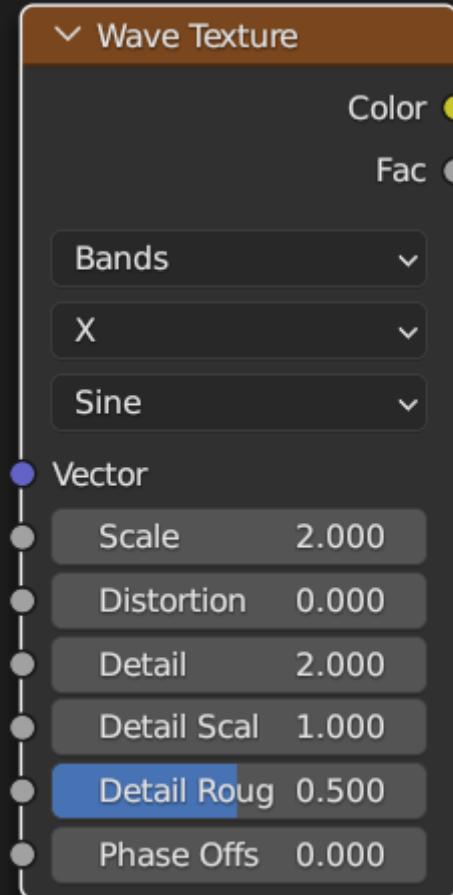


텍스쳐를 눈으로 확인하는 여러 방법

블렌더에서 $y=f(x)$ 를 만들면 2차원 그래프를 볼 수 있습니다.

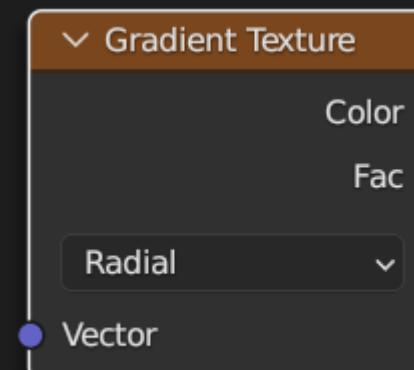


Wave Texture, Gradient Texture



주기함수를 생성합니다.
※범위는 항상 0에서 1 사이입니다.

Detail, Detail Scale, Detail Roughness 모두
Distortion이 만드는 노이즈에 대한 수치입니다.



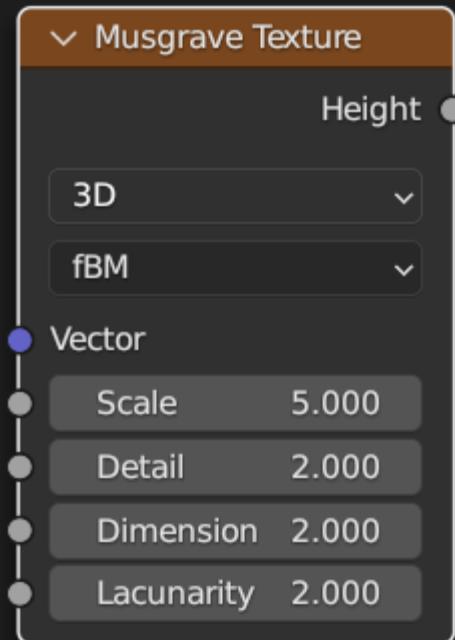
여러 모양의 그라데이션을 생성합니다.

※범위는 0에서 1 사이로 클리핑됩니다.

※Spherical, Quadratic Sphere, Radial의 중심은
(0,0,0)입니다. 따라서 Object 좌표를 사용할 때 제대로
나타납니다.

Musgrave Texture (1)

노이즈 텍스쳐와 비슷하지만..



노이즈와 달리 출력값이 1차원입니다.

범위는 '대체로' -1에서 1 사이이지만 그 이상으로도 넘어갑니다
경우에 따라 Map range 노드를 활용하는 편이 좋습니다

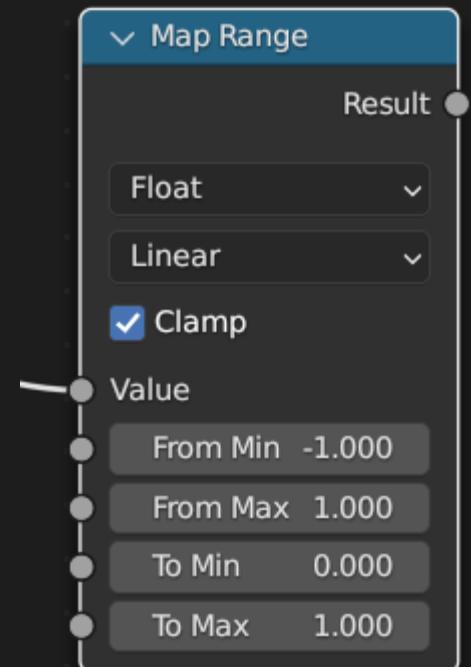
Detail은 Noise texture에서와 같은 뜻이지만,
노이즈 '옥타브'를 조절하는 방식이 다릅니다.

Dimension : 옥타브 간 진폭 차이를 조절합니다.
정확한 공식은

$$Amplitude = \frac{1}{Lacunarity^{Octave \times Dimension}}$$

입니다만..

Lacunarity : 옥타브 간 스케일 차이를 조절합니다.



Musgrave Texture (2) 효과적인 조합들

fBM



Multifractal



Hybrid
Multifractal



Dimension 1.4, Lacunarity 2.0

Dimension 1.4, Lacunarity 2.0

Dimension 0.1, Lacunarity 3.0
Offset 0.45, Gain 0.63

Rigid Multifractal



Hetero Terrain



Dimension 0.1, Lacunarity 3.0
Offset 0.5, Gain 5.0

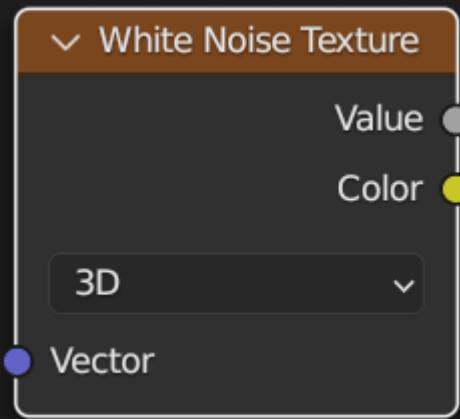
Dimension 0.1, Lacunarity 3.0
Offset -0.06 Gain 5.0

Dimension 1.0, Lacunarity 3.0
Offset 0.3

Dimension 1.0, Lacunarity 3.0
Offset -0.3

White Noise Texture

이게..텍스쳐?



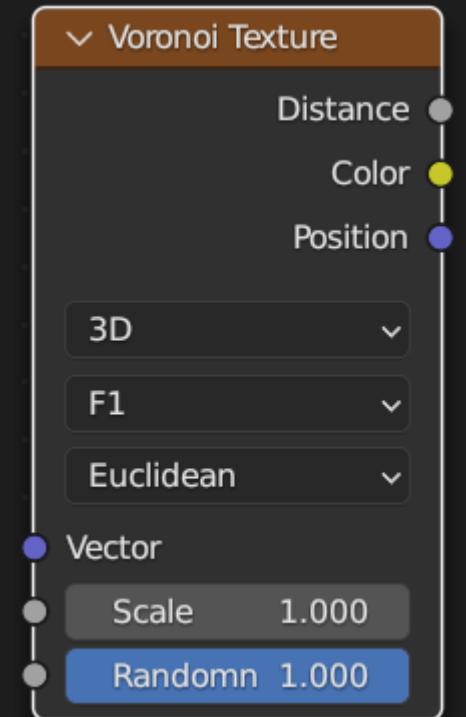
입력값에 따라 0에서 1 사이의 랜덤한 숫자를 내보냅니다.

※입력값이 비슷해도 완전히 다른 값을 내놓습니다.

예컨대 입력값 1.1과 1.101에 대하여, 각각 0.943, 0.792 의 완전히 다른 값을 내놓습니다.
오로지 같은 입력값에 대해서만 같은 출력값을 내놓습니다.



Voronoi Texture



Distance : 공간에 균일하게 분포한 점으로부터의 거리를 나타냅니다.

가장 가까운 점이 무엇이냐에 따라서 경계가 만들어집니다. 그렇게 생긴 구역마다 임의의 색을 칠한 것이 **Color** 출력입니다.

Position은 각 점의 위치값을 출력합니다.

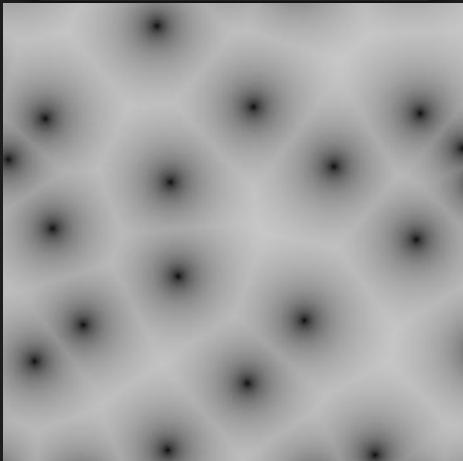
일반적으로 가장 가까운 점까지의 거리를 나타내지만 (F1) 아래와 같이 옵션을 바꿀 수 있습니다.

F2 : '두번째로' 가까운 점으로부터의 거리를 출력합니다.

Smooth F1 : 구역의 경계가 부드러워집니다.

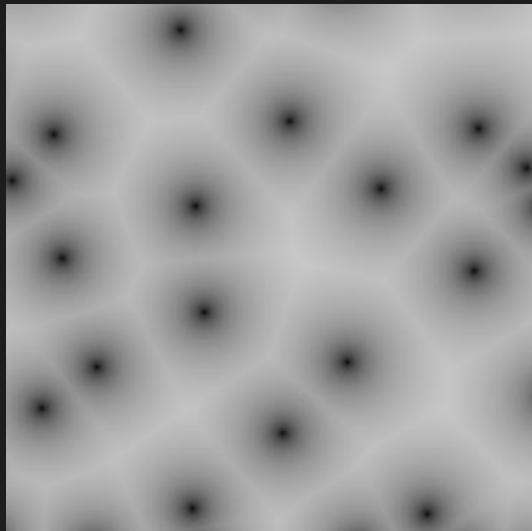
Distance to Edge : 점까지의 거리가 아니라 '경계로부터' 의 거리를 잡니다.

N Sphere Radius : 정의상으로는 점을 중심으로 경계에 내접하는 구의 반지름입니다.
쉽게 말해서 각 구역마다, 점에서 경계까지의 최소거리를 출력합니다.
잘 쓰이지는 않습니다.

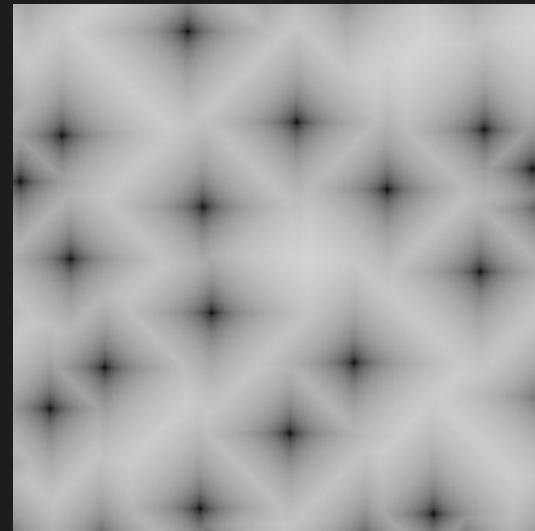


Voronoi Texture (2)

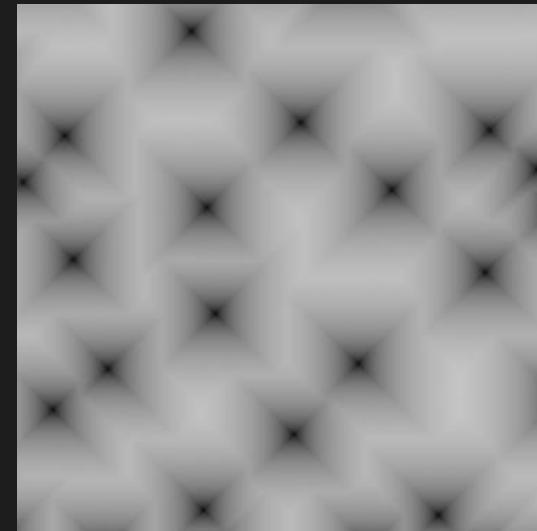
Euclidean, Manhattan, Chebyshev, Mincowski : 거리를 재는 방법을 바꿉니다



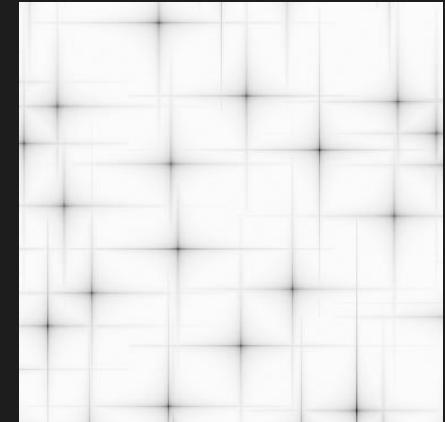
Euclidean



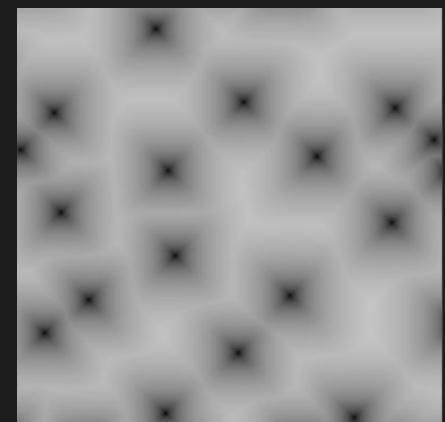
Manhattan



Chebyshev

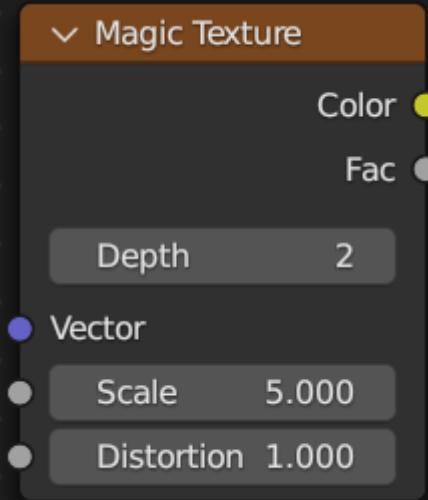


Mincowski ($e=0.3$)



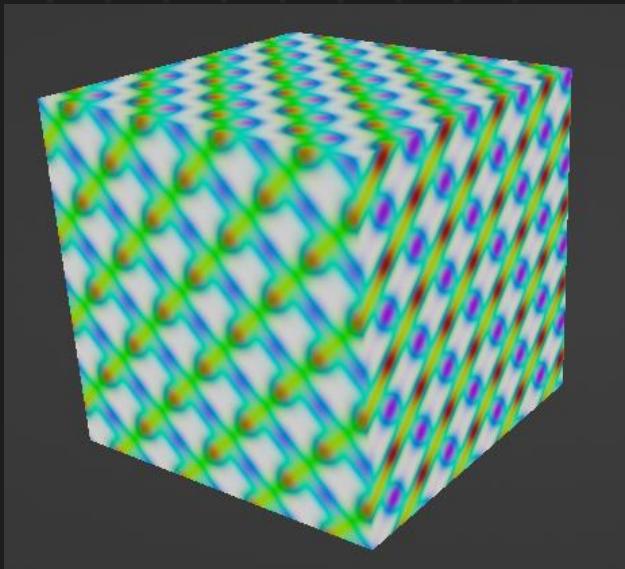
Mincowski ($e=10$)

Magic Texture



싸이키델릭한 텍스쳐 (공식 매뉴얼 표현)

많이 쓰이지는 않지만, 규칙적이면서 단조롭지 않은 패턴을 만들 때 유용합니다.



008강 이미지를 통한 재질 만들기 (1)

기본적인 이미지 텍스쳐 사용법



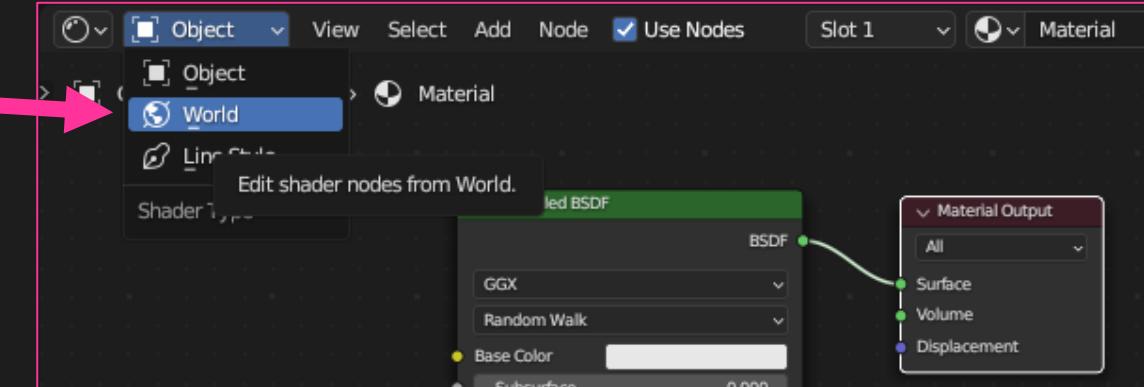
무료 텍스쳐

<https://ambientcg.com/>

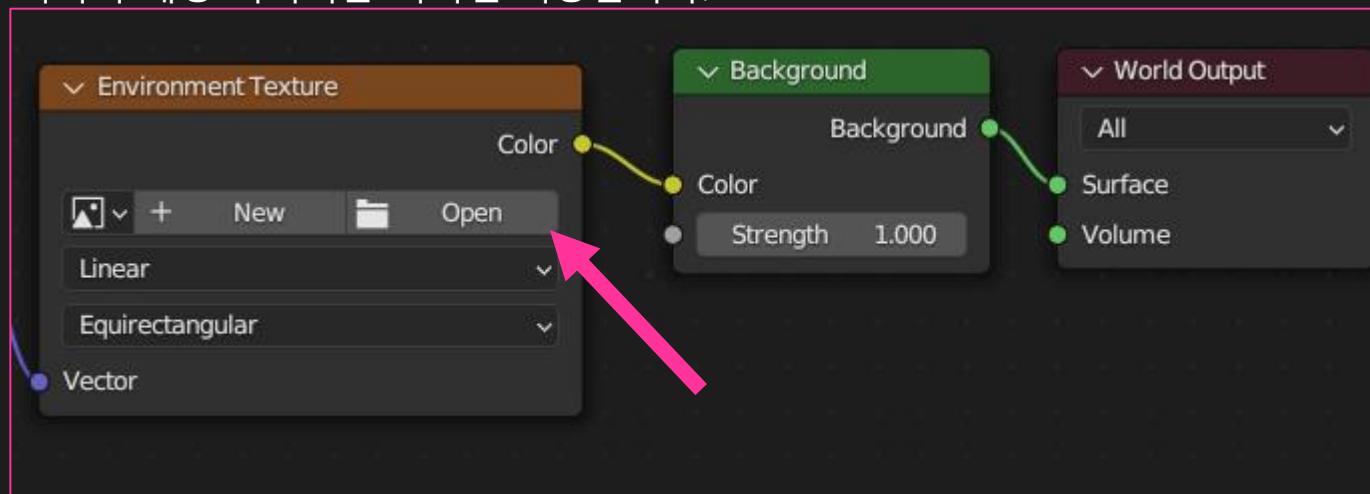
<https://www.3dassets.one/>

배경

셰이더에디터의 왼쪽 상단에서 Object를 World로 바꿔 주시면 배경을 설정하실 수 있습니다.



Background 노드를 클릭하시고, 이미지 텍스쳐 노드를 만드는 단축키 Ctrl+T를 누르시면, Environment Texture가 생성됩니다. 여기서 배경 이미지를 여시면 적용됩니다.



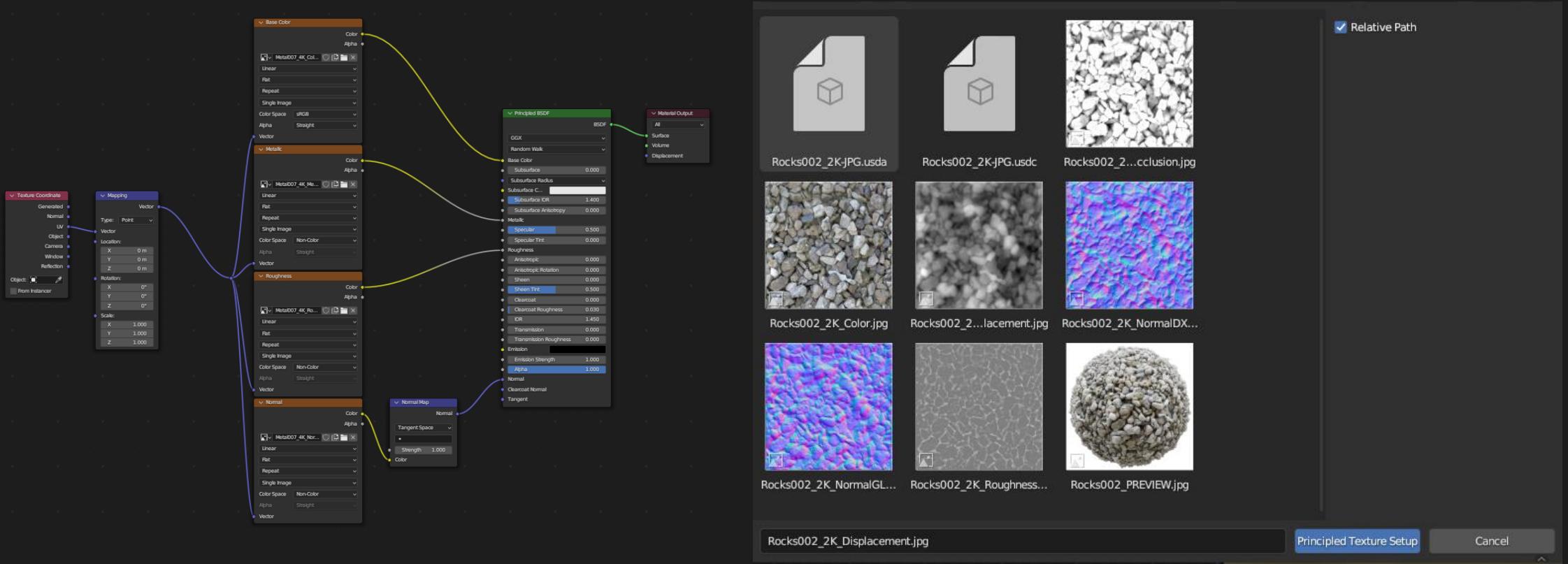
다운로드 받은 재질

Base Color, Metallic, Roughness, Normal 텍스쳐를 가져와서 이름에 맞게 꽂아주면 됩니다.

-Normal은 Normal Map을 거쳐 연결합니다.

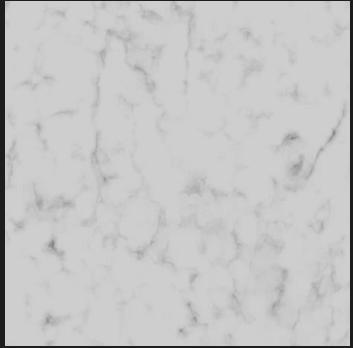
-베이스컬러를 제외하고는 모두 Non-Color로 바꿔줍니다.

만약 Principled Bsdf에서 **Ctrl+Shift+T**를 누르면,
여러 개의 텍스쳐를 한번에 열어 자동으로 연결할 수 있어 편리합니다.



다운로드 받은 재질

다음은 필요에 따라 사용합니다.



Ambient Occlusion

텍스쳐의 구석이나 틈새를 어둡게 만들어주는 텍스쳐입니다.
필요시 Base Color와 섞어줍니다. (Mix Color의 Multiply 사용)

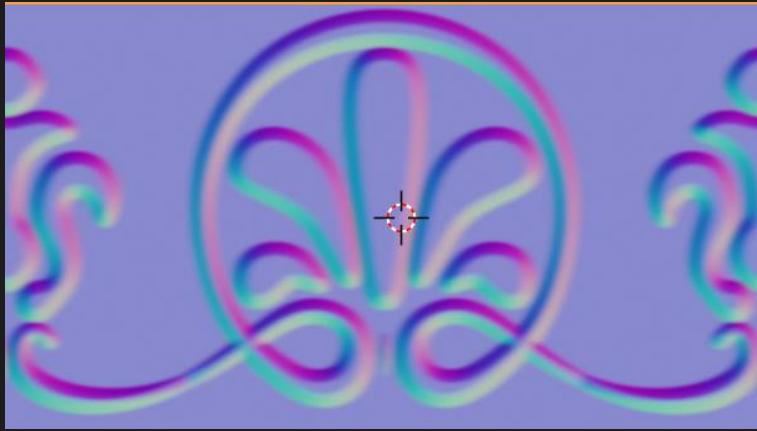


Displacement

높이를 만들어 주기 위한 텍스쳐입니다.
Displacement 노드의 Height에 연결한 뒤 Material Output에 꽂아줍니다.

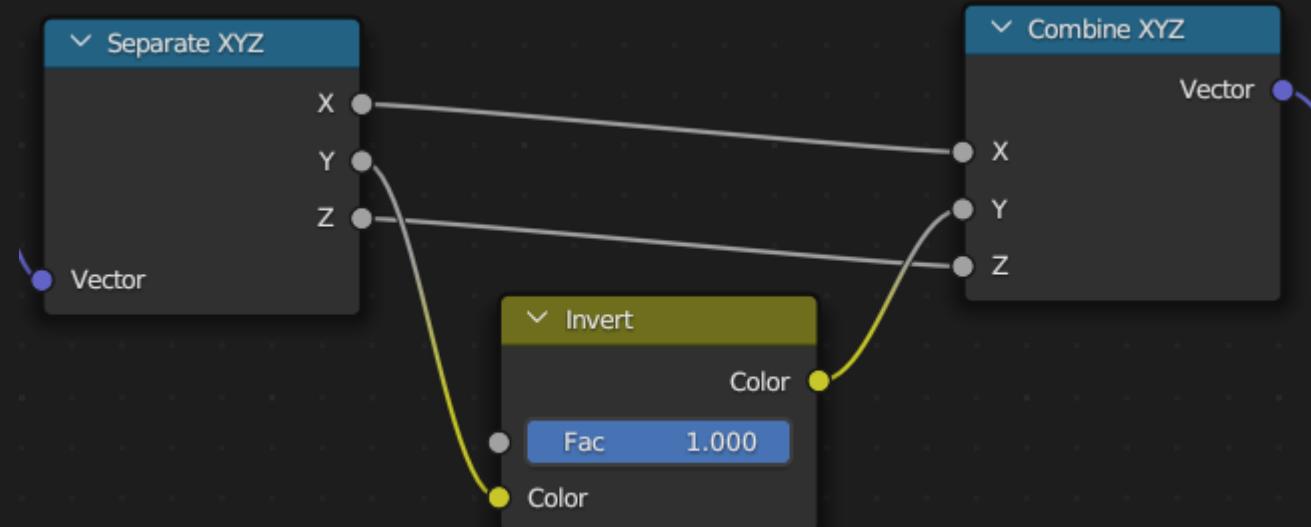
※ 현재 시점에서는 적용 전후의 차이가 없을 것입니다. 자세한 내용은 14강을 참고.

DirectX vs OpenGL



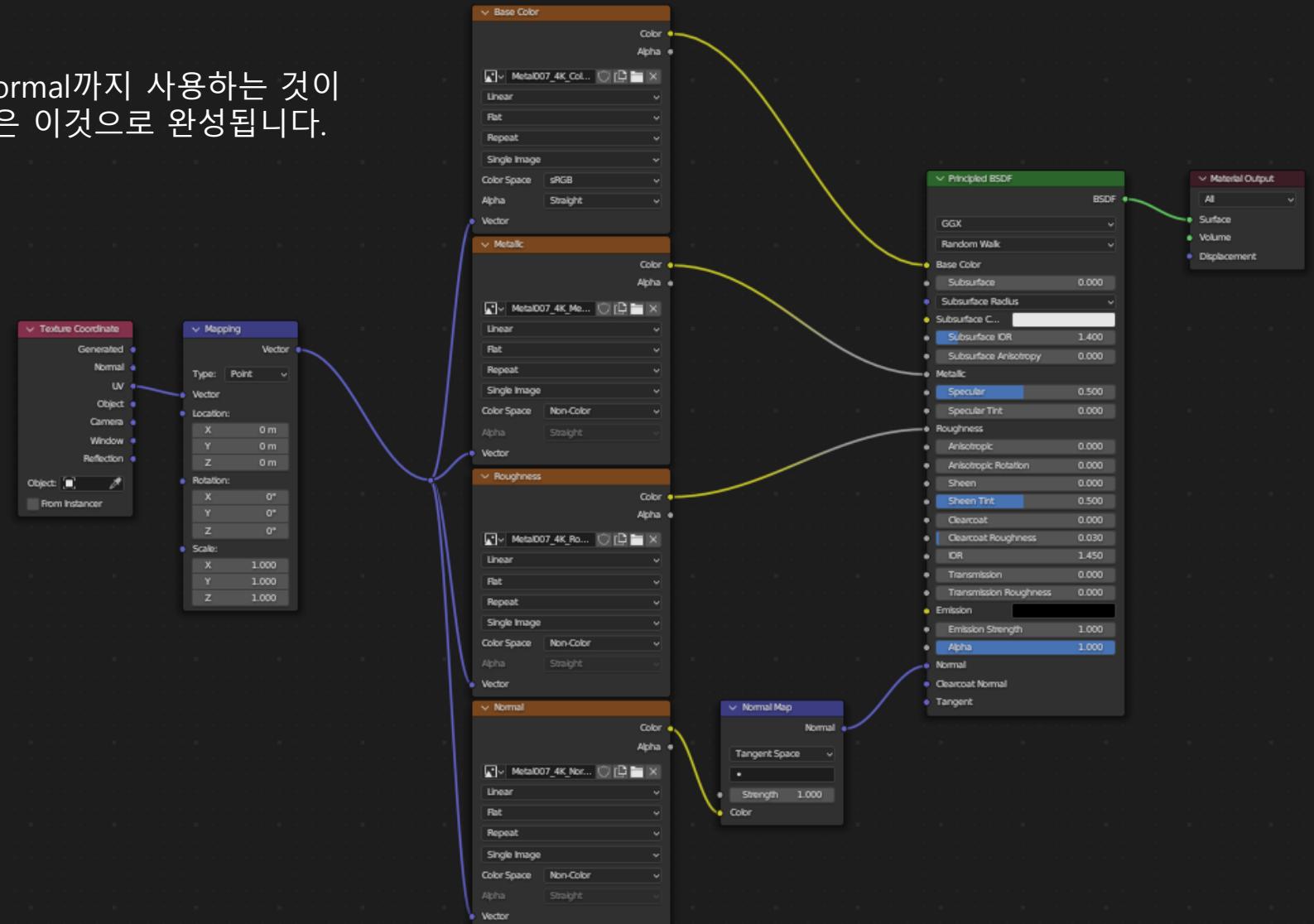
Normal Map은 DirectX와 OpenGL의 두 방식이 존재합니다.
블렌더는 OpenGL방식을 사용합니다.

불가피하게 DirectX 방식으로 만들어진 Normal Map을 사용해야 할 경우,
아래와 같이 Y좌표를 뒤집어야 합니다.



연결(1) 일반적인 연결

Base Color, (Metallic), Roughness, Normal까지 사용하는 것이 기본적인 연결입니다. 대부분의 재질은 이것으로 완성됩니다.



연결(2) 추가적인 재질 설정

앞에서의 4가지 연결로 표현할 수 없는 재질들은 추가적인 설정이 필요합니다.

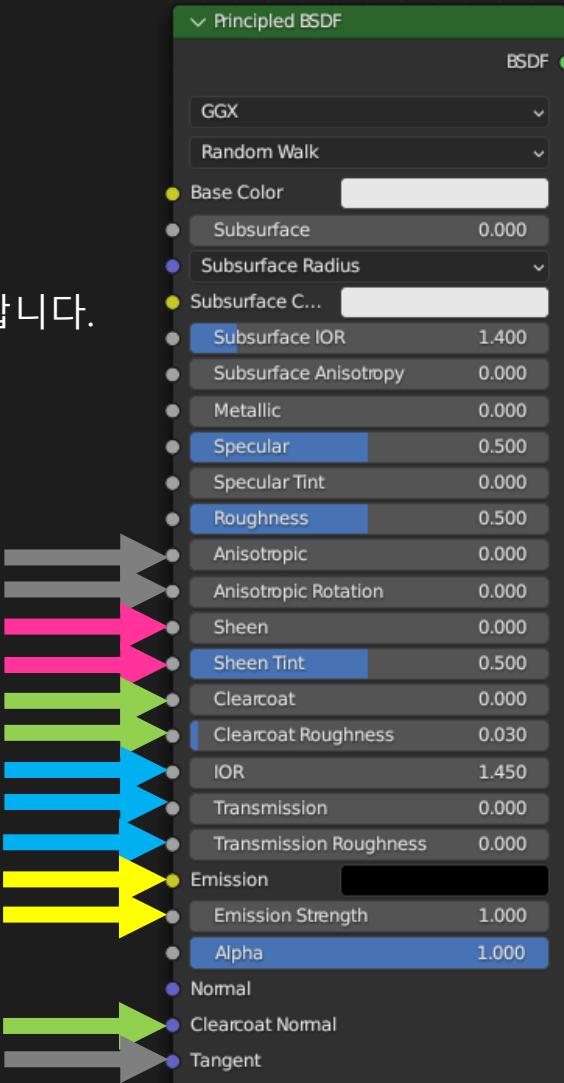
부드러운 천 재질 : Sheen 슬라이더를 올립니다.

표면이 코팅된 재질 : Clearcoat

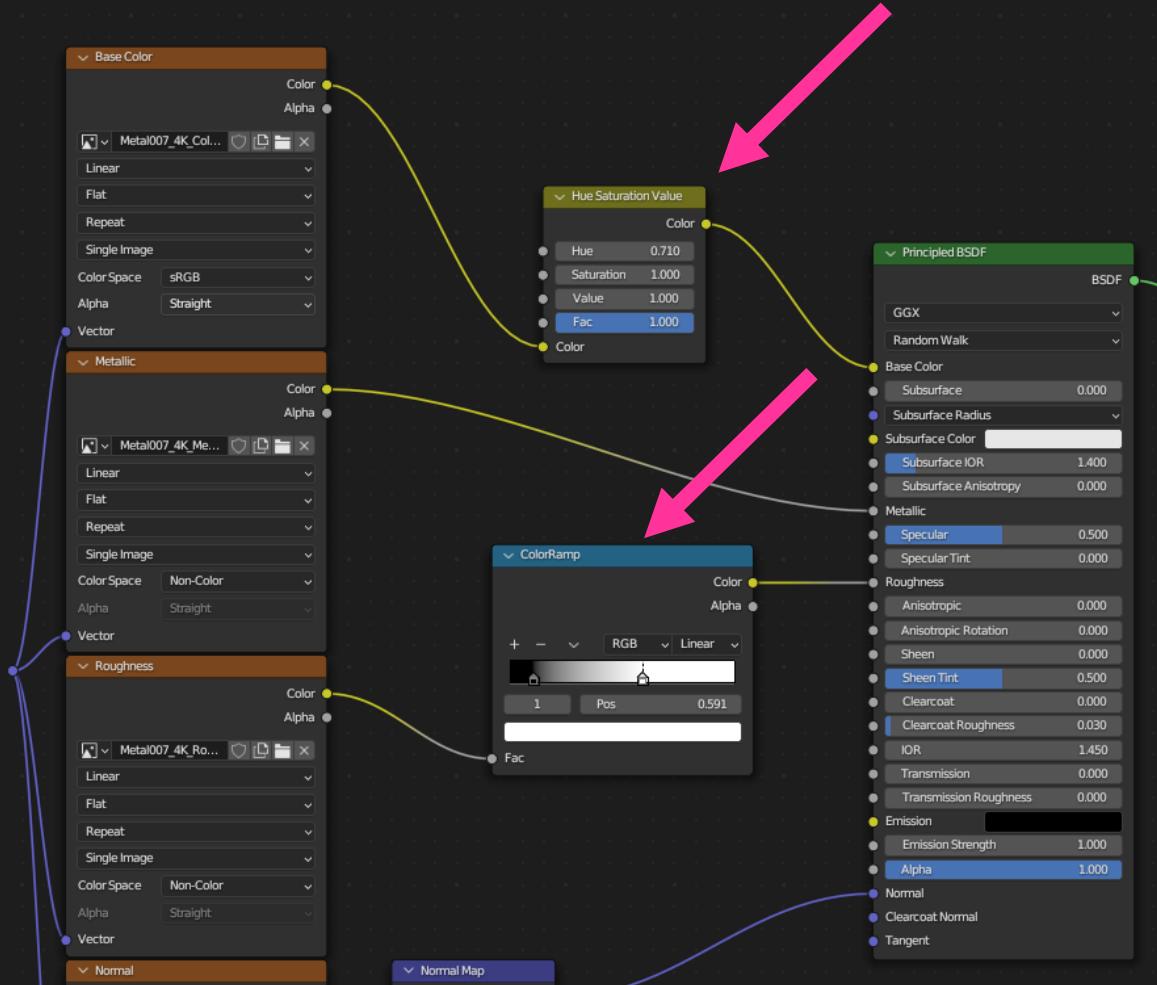
유리 : Transmission

발광하는 조명 등 : Emission

Brushed Metal : Anisotropic



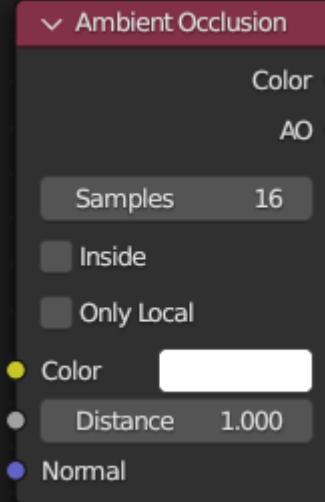
연결(3) 텍스쳐 보정과 추가효과



텍스쳐의 색을 바꾸거나,
특히 Roughness를 뚜렷이 강조하기 위해 텍스쳐를 보정합니다.

구석진 곳이나 외곽선을 강조하기 위해
Ambient Occlusion, Bevel을 사용할 수도 있습니다.

Ambient Occlusion, Bevel

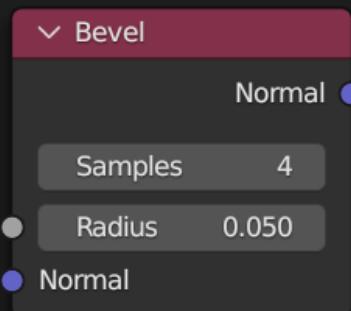


지오메트리를 바탕으로 Ambient Occlusion을 만들어주는 노드.

Inside : 기능이 거꾸로 작동합니다! 즉, 튀어나온 부분을 어둡게 합니다.

Only Local : 단일 오브젝트 내에서만 계산합니다. 이 옵션은 Cycles에서만 작동합니다.

Distance : 앰비언트 오클루전 효과가 미치는 범위를 지정합니다.



Bevel : 입력받은 노멀에 일종의 흐림 효과를 주어 모서리를 부드럽게 만들어 줍니다. 이 노드는 Cycles에서만 작동합니다.

Radius : 흐림 효과의 강도를 조절합니다.

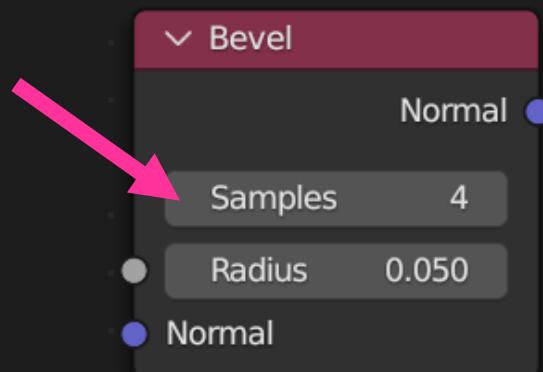
Bevel과 Ambient Occlusion 사용시 유의점

AO와 Bevel은 컴퓨터가 연산을 반복하여 만들어냅니다. 이를 조절하는 것이 **Samples**입니다.

연산 횟수가 적을수록 거친 결과가 만들어집니다.

연산 횟수가 많을수록 정확하고 부드러워지지만, 렌더링 시간이 늘어납니다.

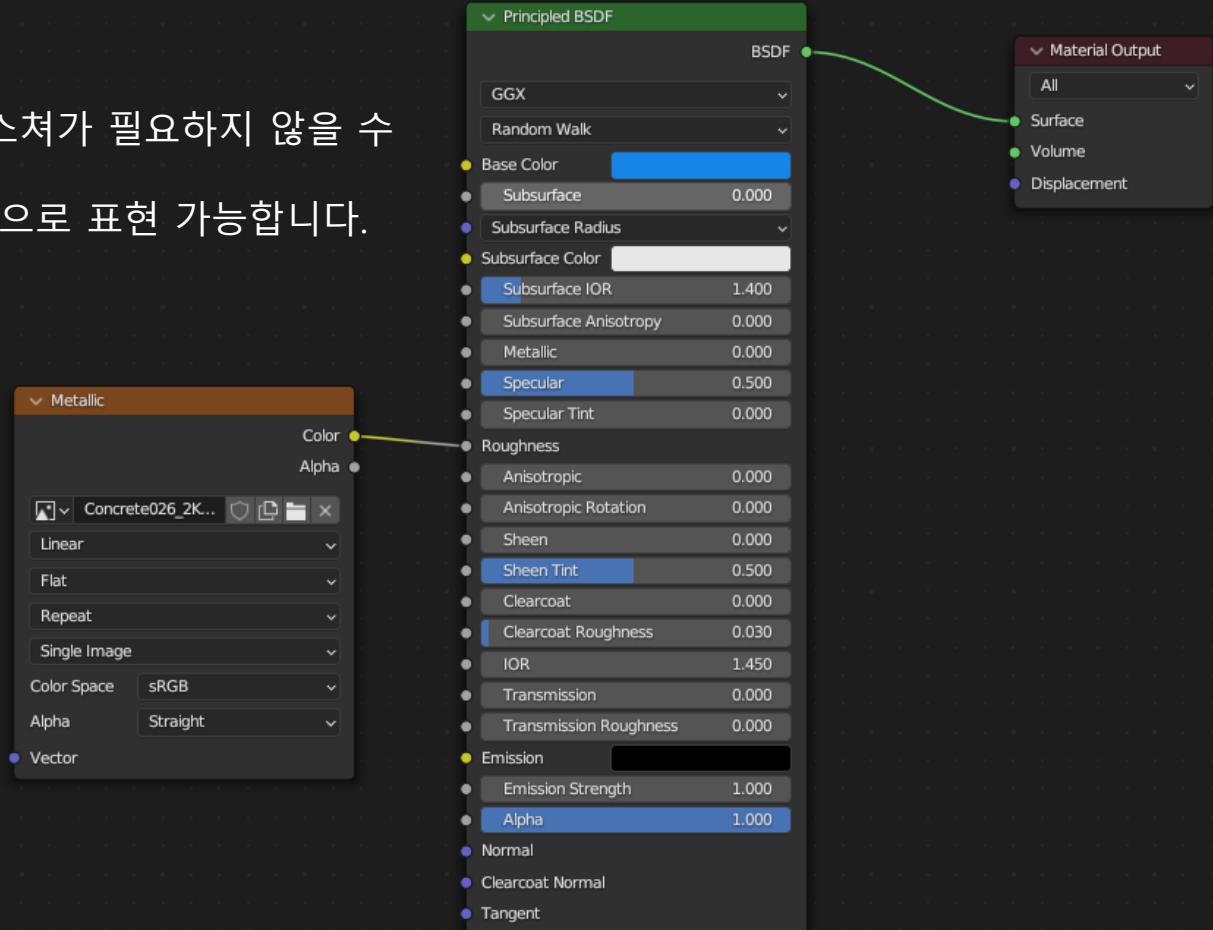
※AO와 Bevel의 Samples 수치는 렌더링 옵션의 Sample과는 별개의 수치입니다.
(물론 렌더링 샘플에도 영향을 받습니다.)



연결(4) 필요에 따라 일부만 연결할 수도 있습니다.

페인트 등으로 표면에 단색을 칠했다면 BaseColor는 텍스쳐가 필요하지 않을 수도 있습니다.

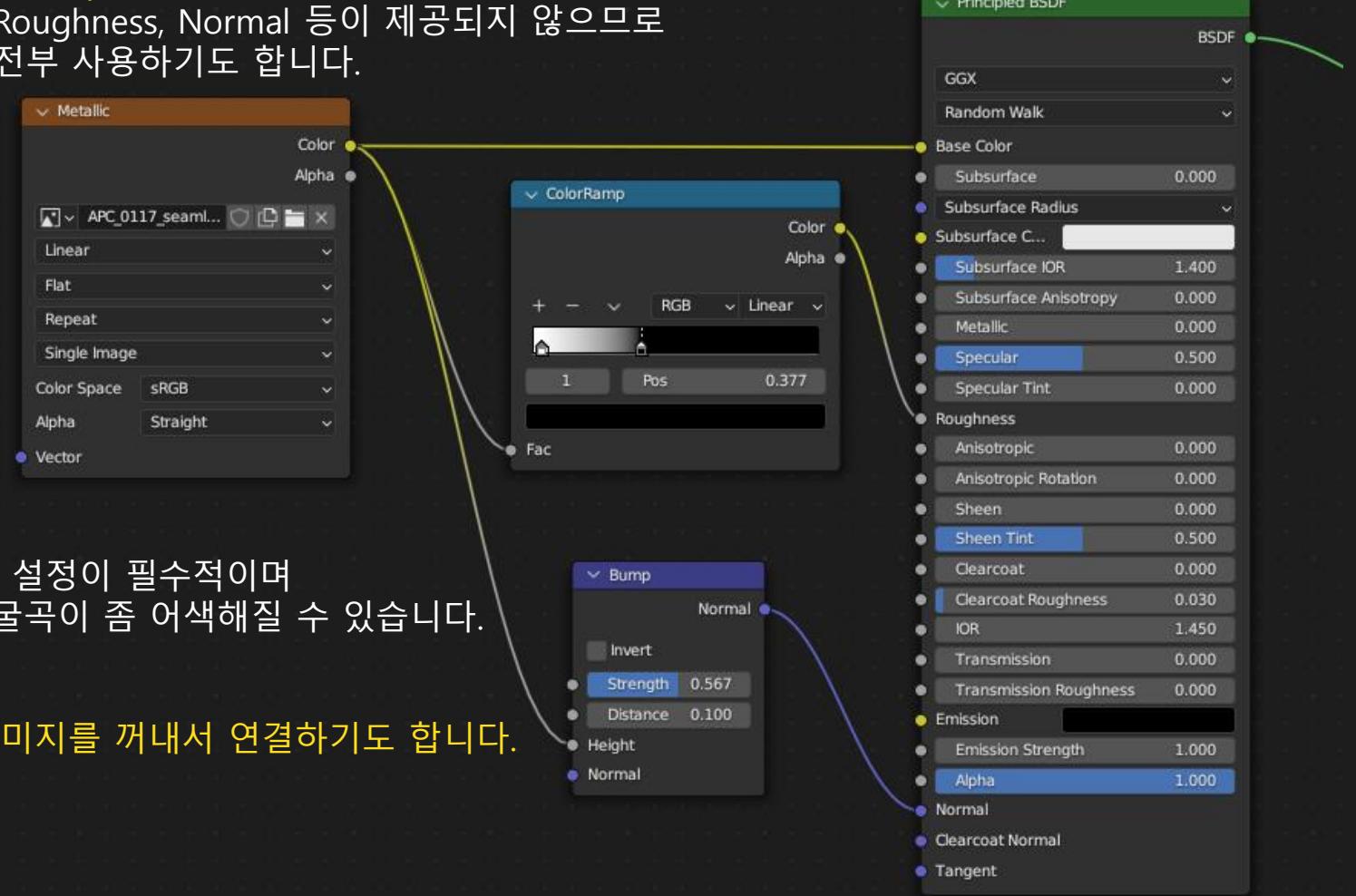
이런 표면들은 Roughness나 Normal만 연결해도 효과적으로 표현 가능합니다.



연결(5) 이미지를 통한 재질 만들기

하나의 이미지를 Color, (Metallic), Roughness, Bump 에 모두 사용하는 방법입니다.

재질이 아닌 일반적인 텍스쳐를 사용할 때는 Roughness, Normal 등이 제공되지 않으므로
이럴 때는 이렇게 하나의 이미지를 나머지에 전부 사용하기도 합니다.



결과는 생각보다 나쁘지 않지만

- Roughness는 ColorRamp등의 추가적인 설정이 필수적이며
- Bump를 통한 노멀맵 생성으로 표면의 굴곡이 좀 어색해질 수 있습니다.

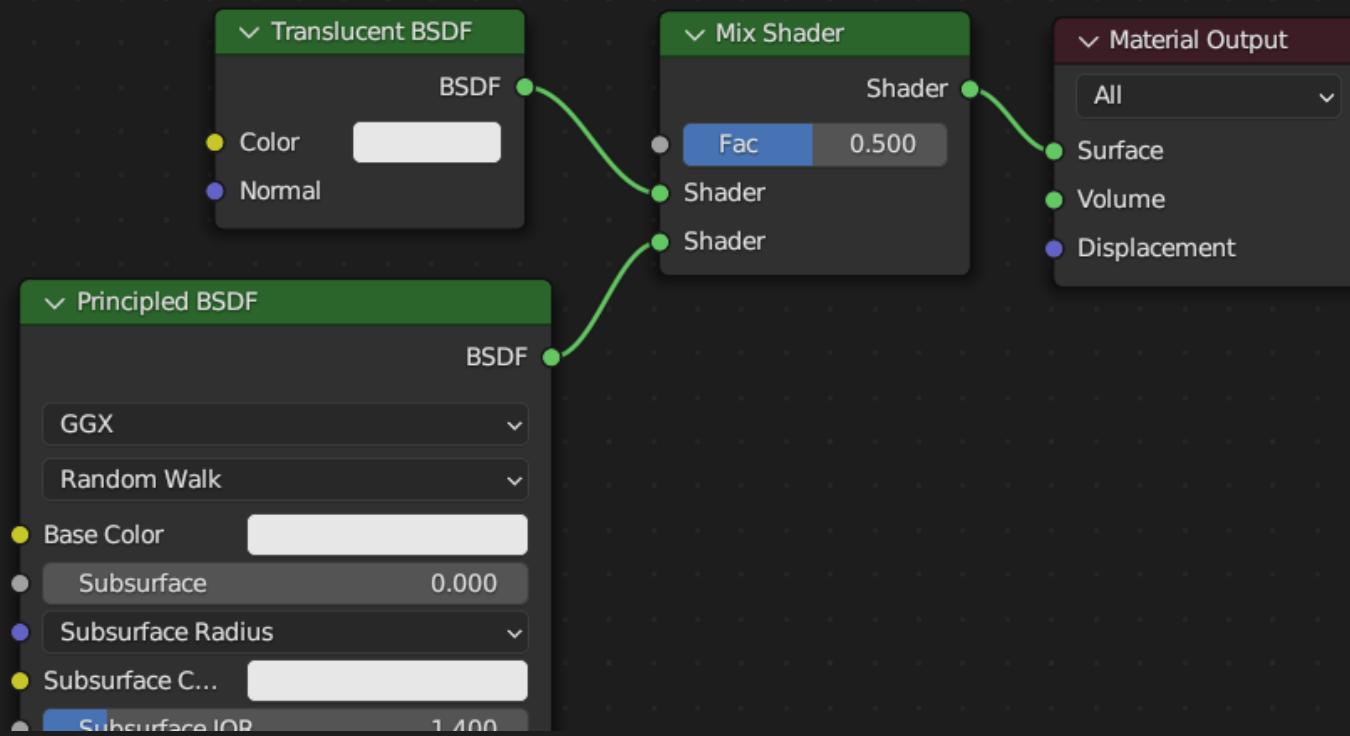
그런 경우 Roughness와 Bump에 또 다른 이미지를 꺼내서 연결하기도 합니다.

Translucent

투과하는 재질

커튼이나 종이와 같이 빛이 투과하는 재질을 만들 때 사용합니다.
Subsurface를 쓸 수도 있지만, 이쪽은 연산이 무겁고 두께가 있을 때만 작동합니다.

Translucent는 Principled BSDF에 없으므로 Mix Shader나 Add Shader로 섞어서 사용합니다.



009강 이미지를 통한 재질 만들기 (2)

외부에서 가져온 소재를 사용하기



다운로드받은 오브젝트 사용하기

Polyhaven

<https://polyhaven.com/>

Hdri 배경과 블렌더 머티리얼, 오브젝트 에셋을 제공합니다.

다운로드받은 오브젝트 사용하기

3D스캔



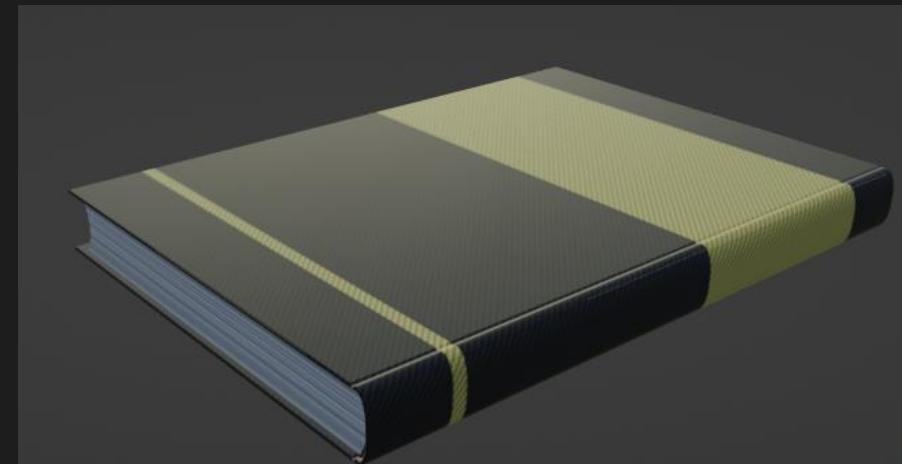
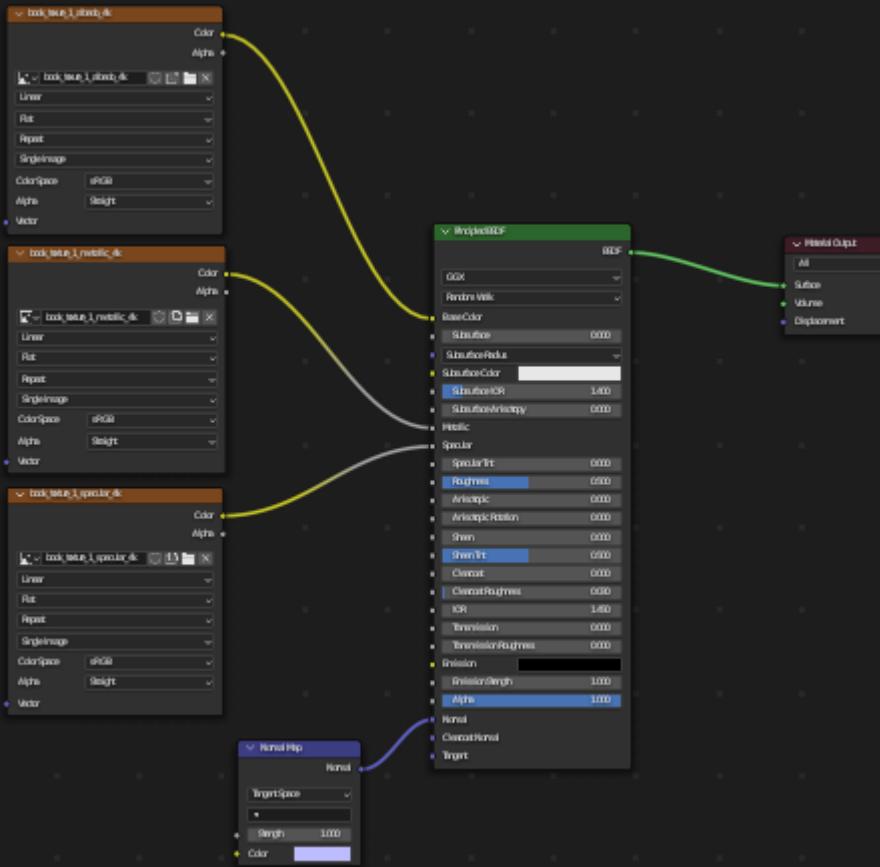
<https://sketchfab.com/> 에, 박물관에서 제공한 CC0 파일들이 있습니다.

3D 스캔 모델은 보통 폴리곤 수가 너무 많기 때문에 조정이 필요합니다.

다운로드받은 오브젝트 사용하기

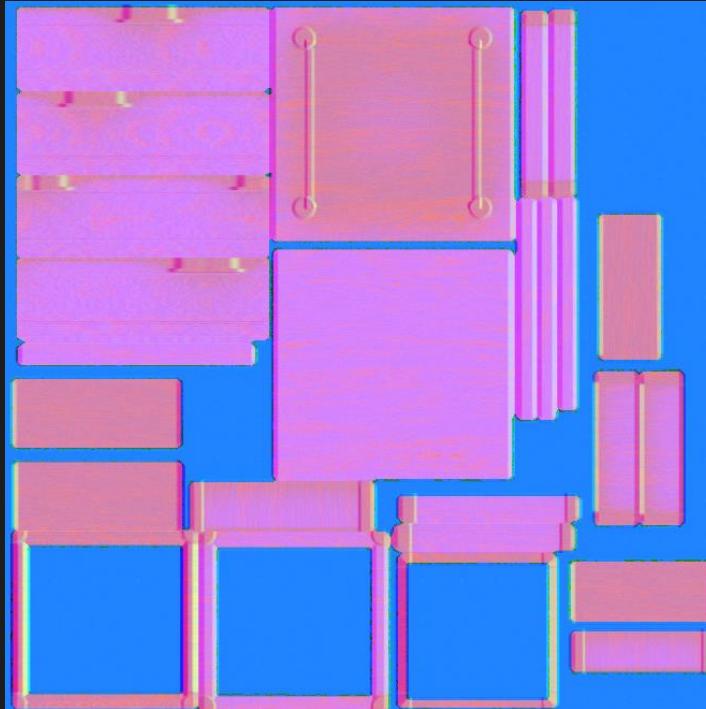
연결이 나쁜 파일들

블렌더의 세이더 시스템은 Fbx, Obj와 완벽히 호환되는 것이 아닙니다.

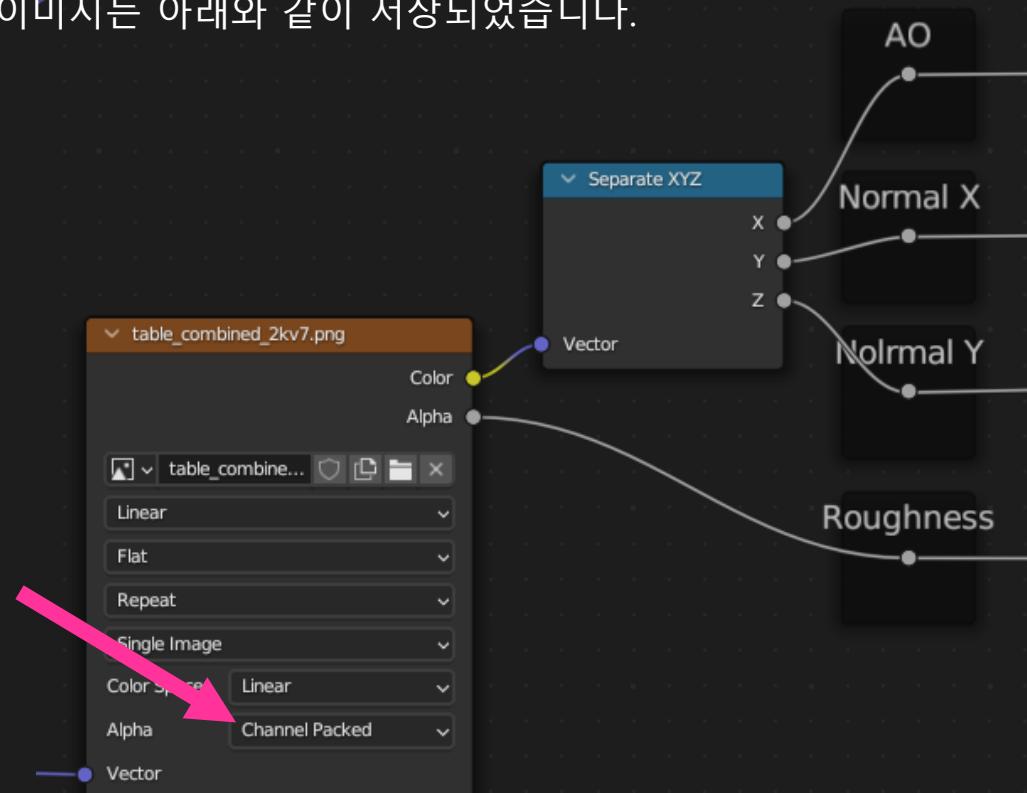


다운로드받은 오브젝트 사용하기

Channel Packed Image



Roughness, Ambient Occlusion, Normal 등을 이미지의 RGBA 채널에 저장한 이미지입니다.
저장 순서에 일반적인 규칙은 없습니다.
강의에서 사용한 이미지는 아래와 같이 저장되었습니다.

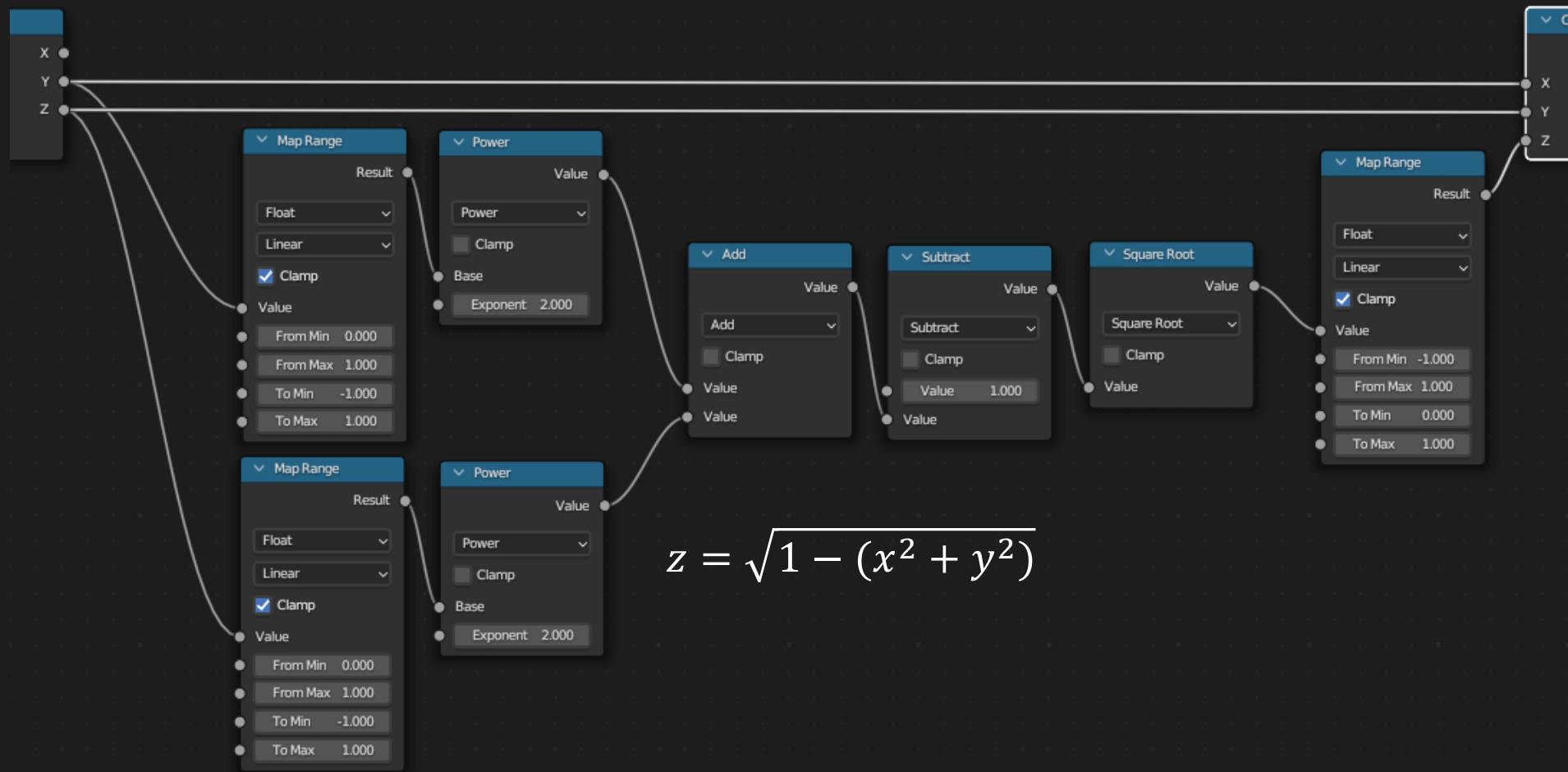


다운로드받은 오브젝트 사용하기

Normal Z 좌표값을 얻는 법

Channel Packed image는 대개 Normal의 두개 좌표만 저장합니다.

Normal은 항상 크기가 1이라는 사실을 이용하면 나머지 축을 통해 z좌표를 계산해서 얻을 수 있습니다.



010강 단조로움을 피하는 방법

반복에 의한 단조로움을 피하는 방법

오브젝트별 반복 피하기

Object Info 노드를 이용한 랜덤 간판 만들기



단조로움

연결부위는 안 보여도..



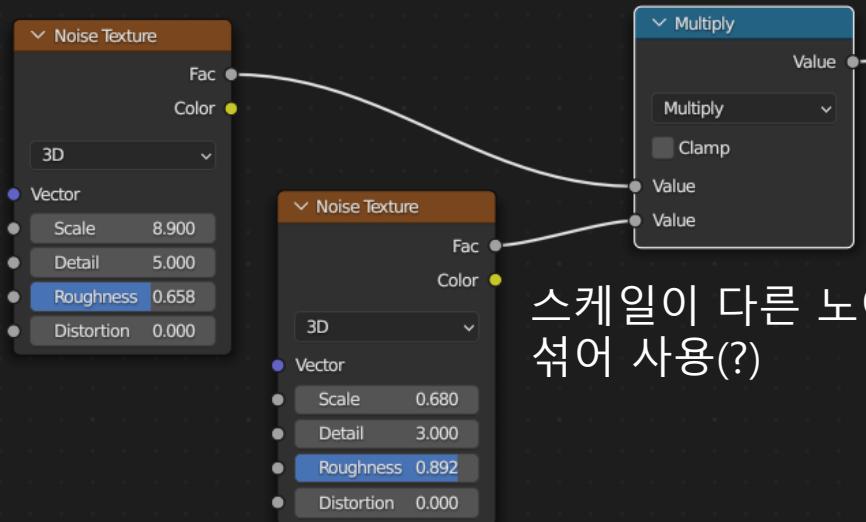
단조로움을 피하기 위해서

노이즈 섞기

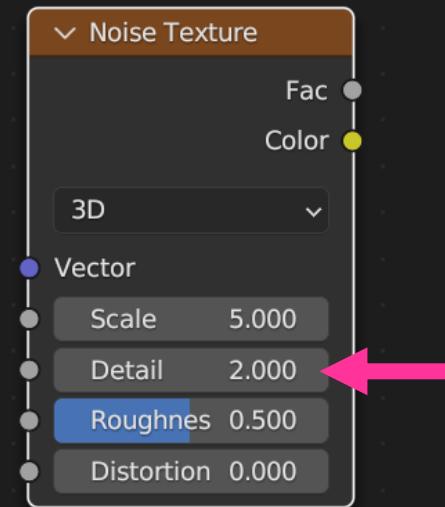


단조로움을 피하기 위해서

노이즈의 노이즈(?)



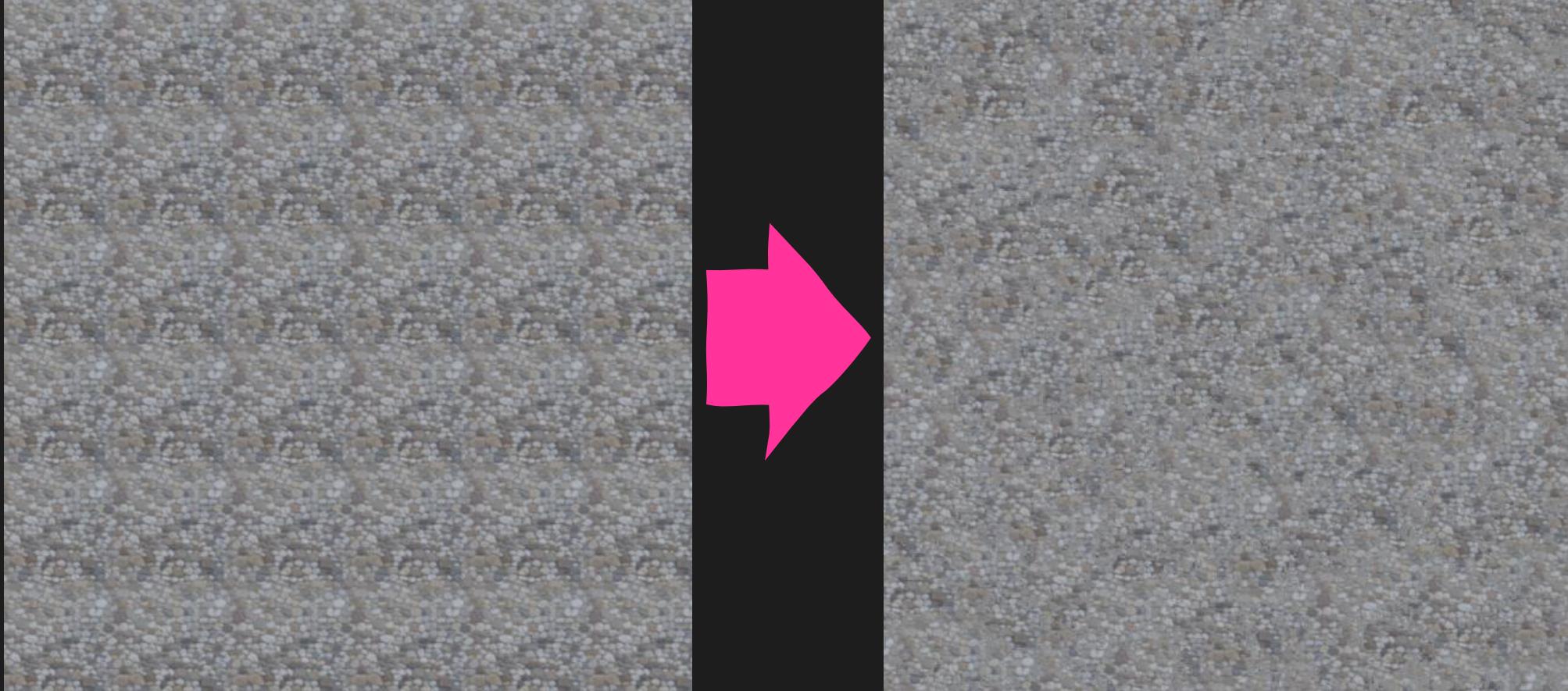
스케일이 다른 노이즈를
섞어 사용(?)



이미 여러 개의 노이즈가
섞인 상태입니다.

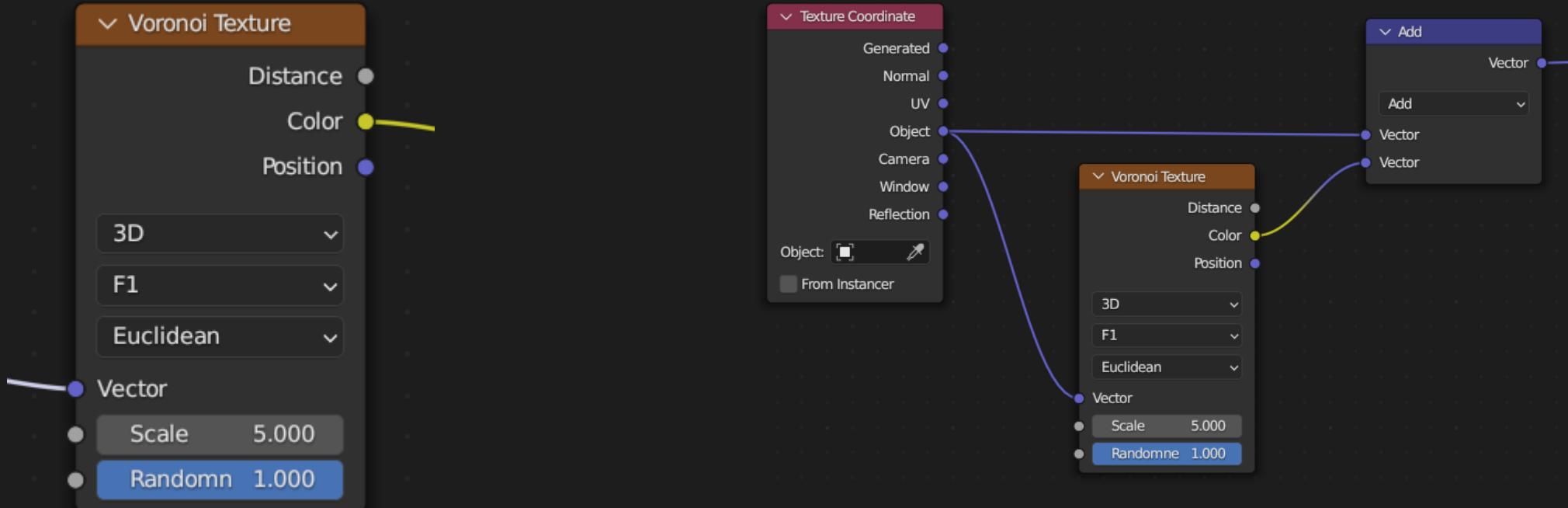
보로노이 랜덤좌표

원본 텍스쳐의 느낌 그대로 반복만 피할 순 없을까요?



보로노이 랜덤좌표(2)

궁금하신 분들을 위한 자세한 설명

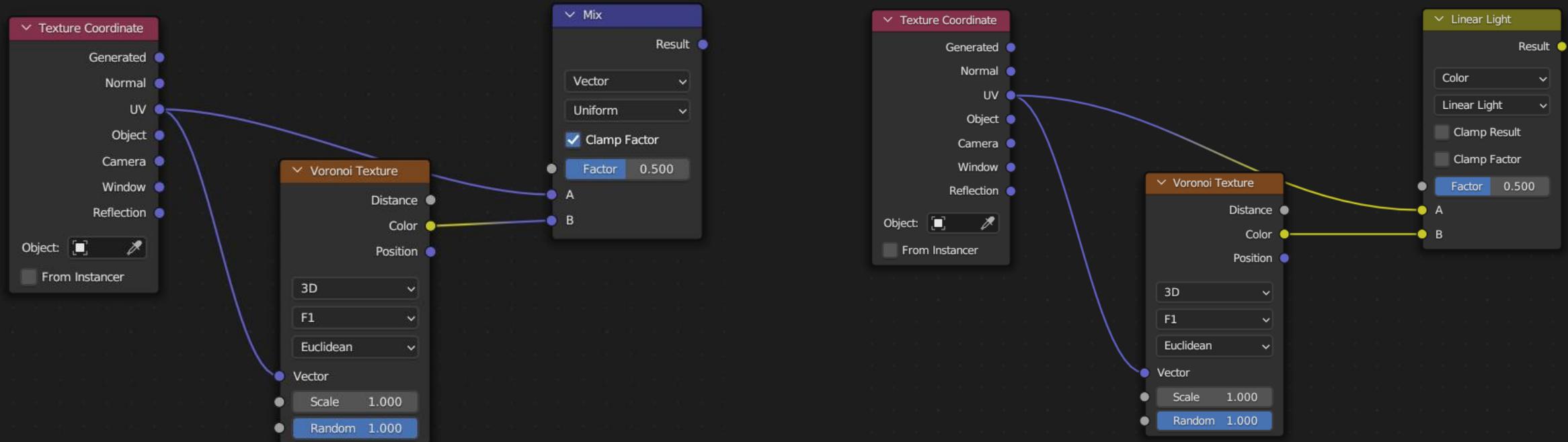


보로노이 텍스쳐는 Color라는,
셀마다 랜덤값을 출력하는 소켓이 있습니다.

만약 텍스쳐 좌표에 더한다면,
셀마다 좌표가 랜덤으로 어긋나게 될 것입니다.

보로노이 랜덤좌표(3)

궁금하신 분들을 위한 자세한 설명

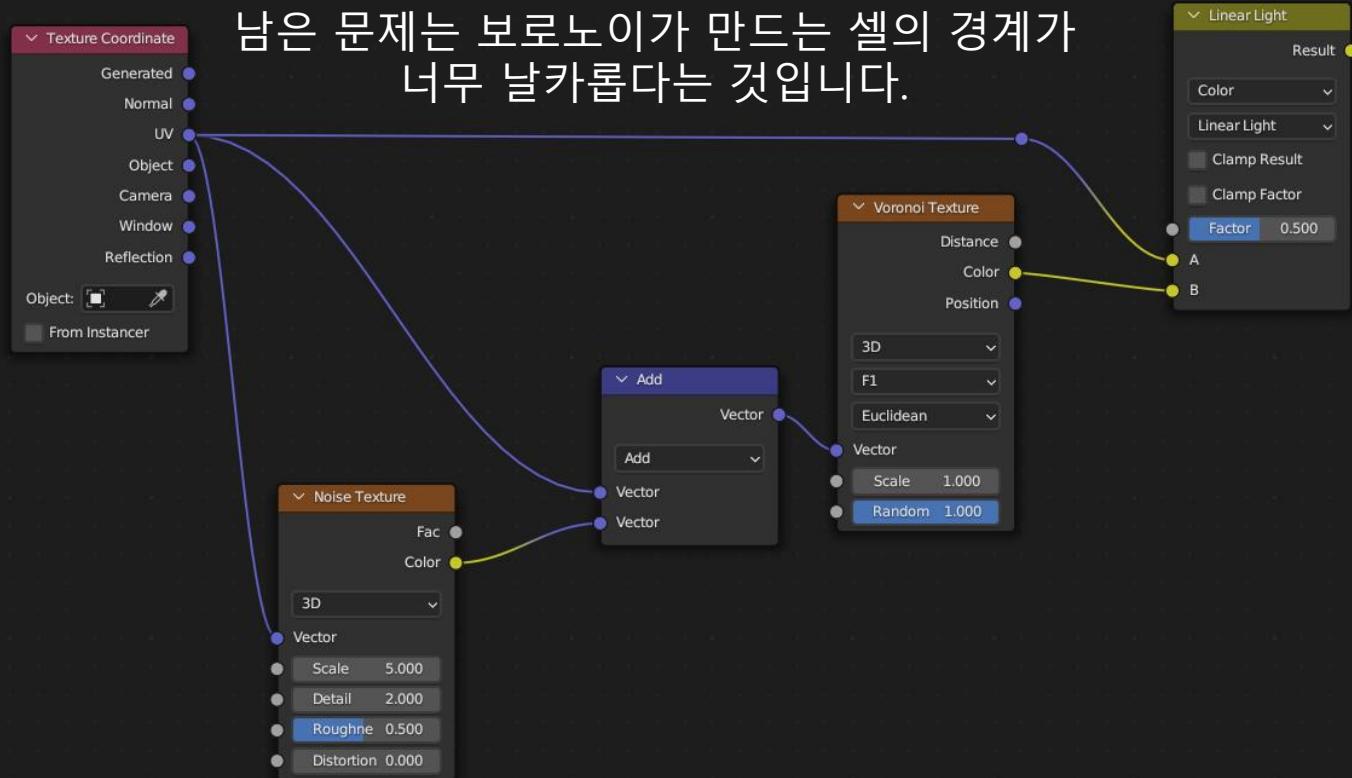


Vector Math 대신 Mix노드를 쓰면
보로노이 텍스쳐를 섞는 정도를 조절할 수 있습니다.

더 나아가, Linear light를 쓰면
-- 방향 양쪽으로 이동시킬 수 있습니다.

보로노이 랜덤좌표(4)

궁금하신 분들을 위한 자세한 설명

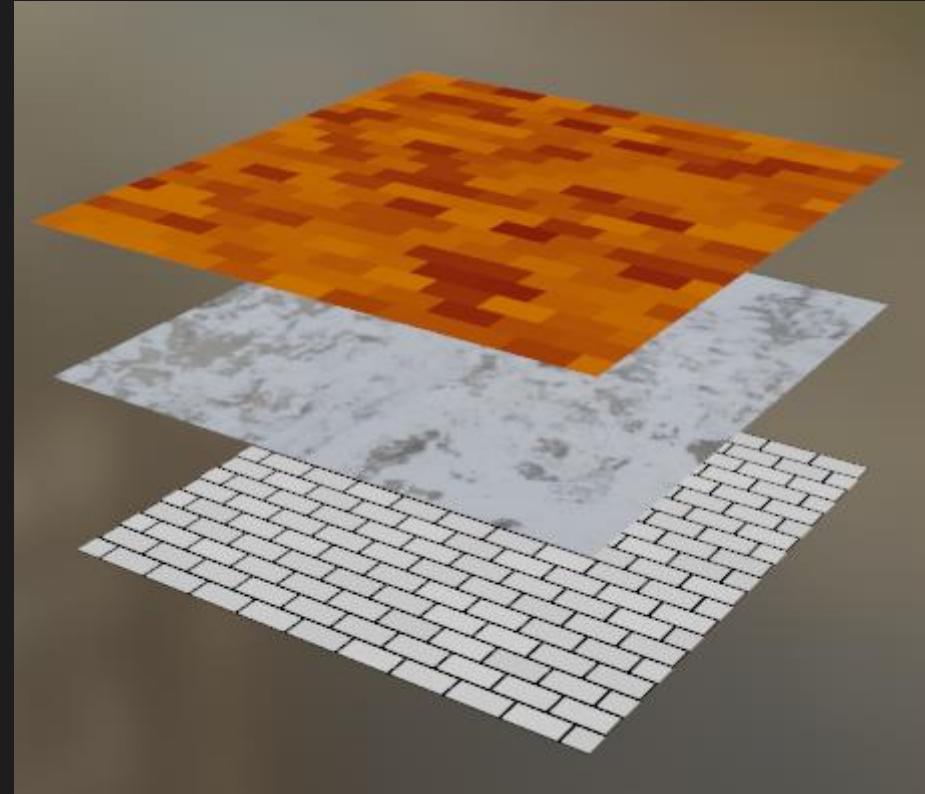


남은 문제는 보로노이가 만드는 셀의 경계가
너무 날카롭다는 것입니다.

그래서.. 보로노이의 좌표에 노이즈를 더해서 경계를 일그러트렸습니다.
이때도 Vector Math 대신 Mix노드를 쓰는게 낫고, 또한 Linear light를 쓰면....

벽돌이나 타일의 경우

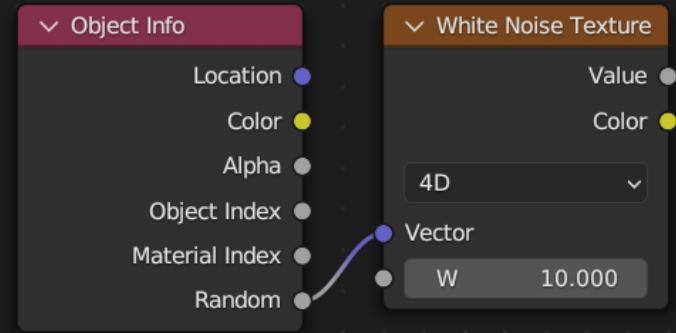
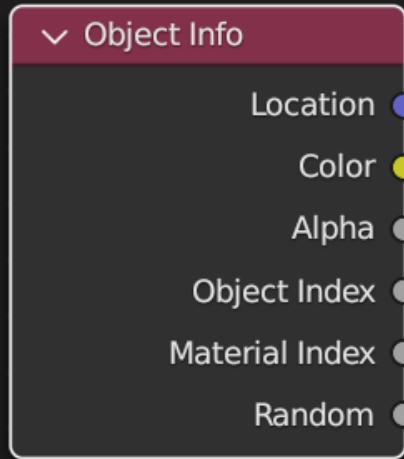
구성요소를 분리할 수 있다면....



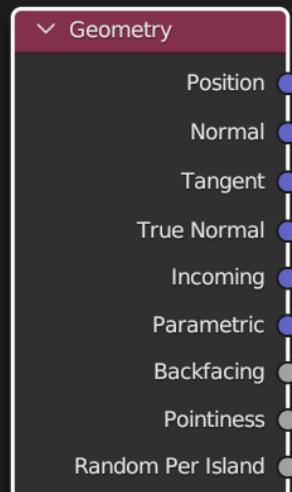
※ 자세한 사항은 16~19강을 참고해주세요.

오브젝트 랜덤

오브젝트마다 텍스쳐를 바꾸고 싶으면



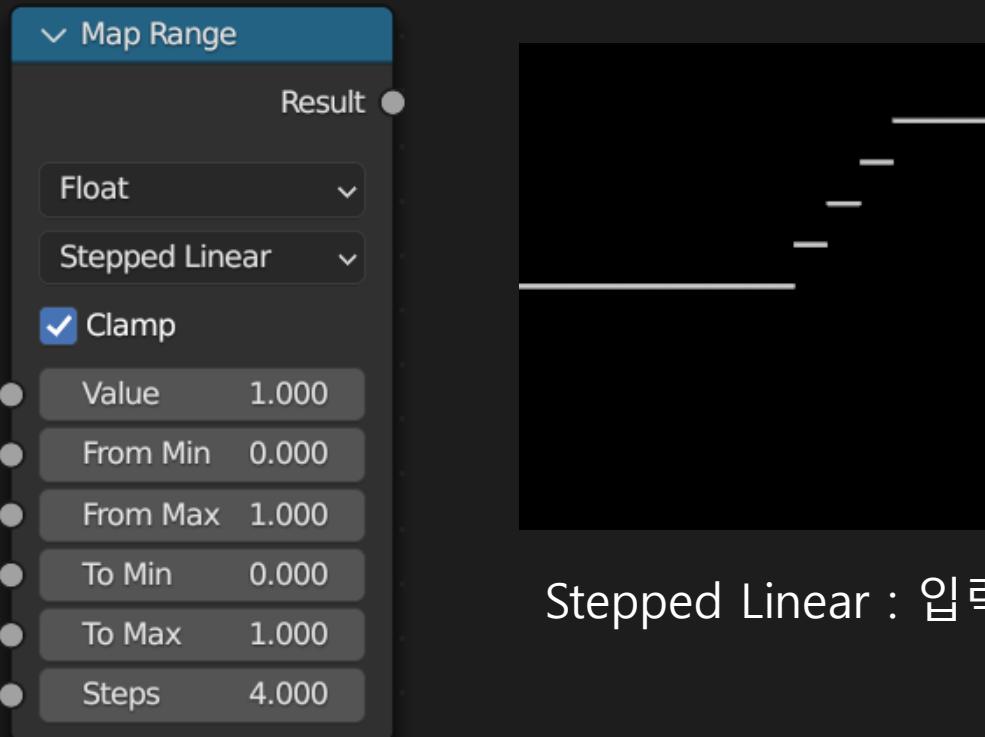
Random : 오브젝트마다 랜덤값을 출력합니다.
랜덤값은 흑백이지만 **White Noise Texture**와 함께 사용하면 랜덤 컬러를 만들 수 있습니다.



만약 뎅어리가 오브젝트로 분리되어 있지 않고, 하나의 오브젝트로 묶여 있다면 Random Per Island 를 사용할 수도 있습니다. (이 소켓은 Cycles에서만 작동합니다.)

Stepped Linear

Map range 노드의 숨겨진 기능



Stepped Linear : 입력값을 몇 단계로 나눌 수 있습니다.

011강 투명한 재질

유리 사용시의 유의점 (Cycles / Eevee)

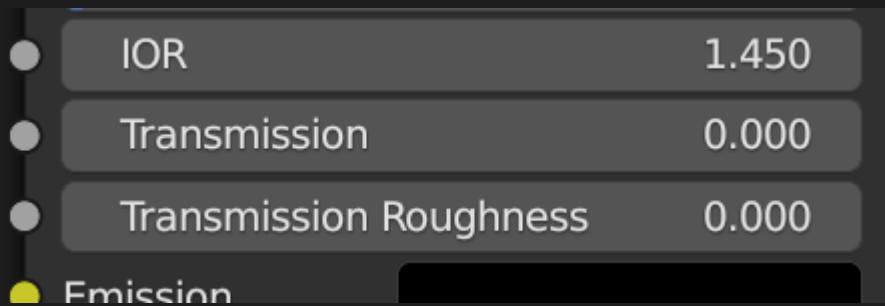
Caustics

Eevee에서의 투명 설정

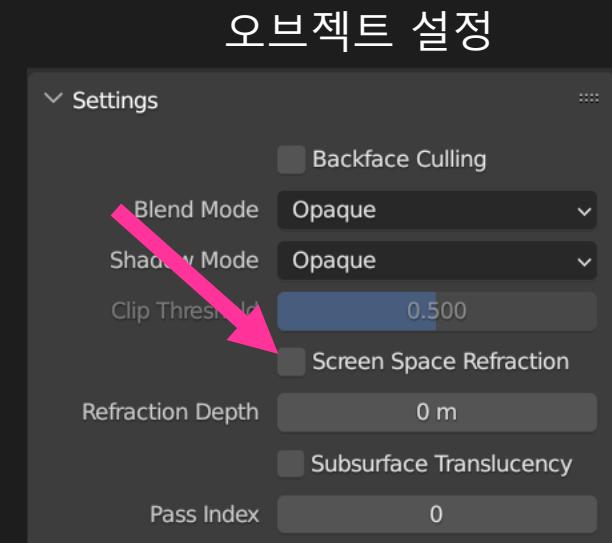
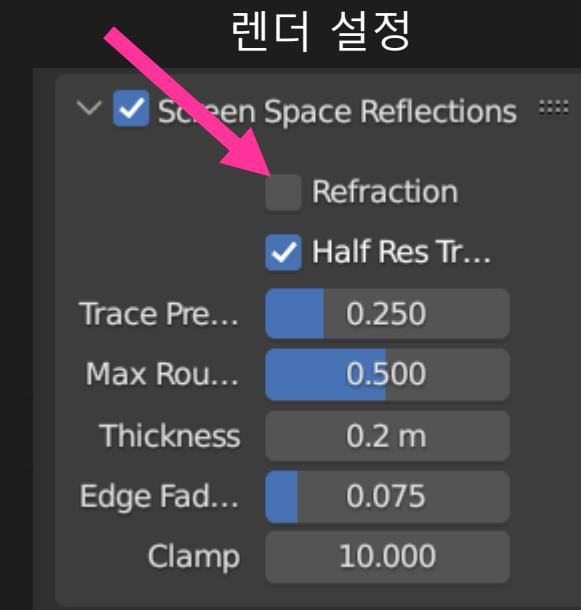


유리 재질, (반)투명 재질

Transmission을 켜는 순간 유리가 됩니다.
IOR(굴절률)을 조절하여 재질의 차이를 만들어낼 수 있습니다.



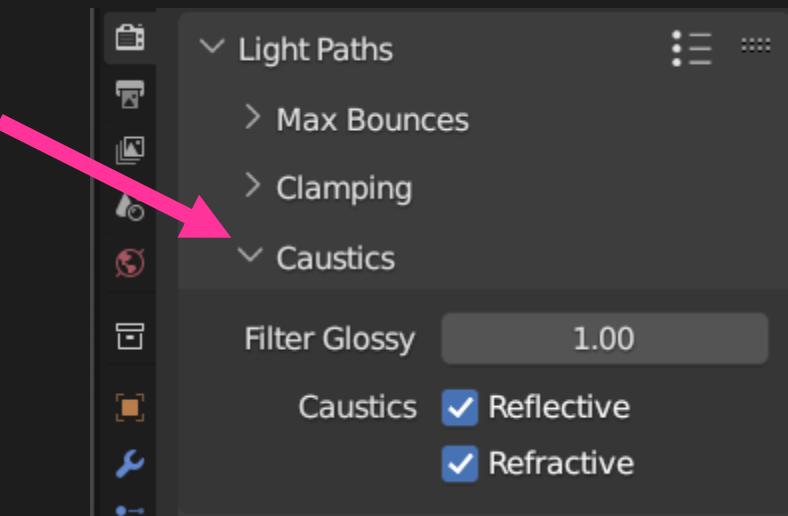
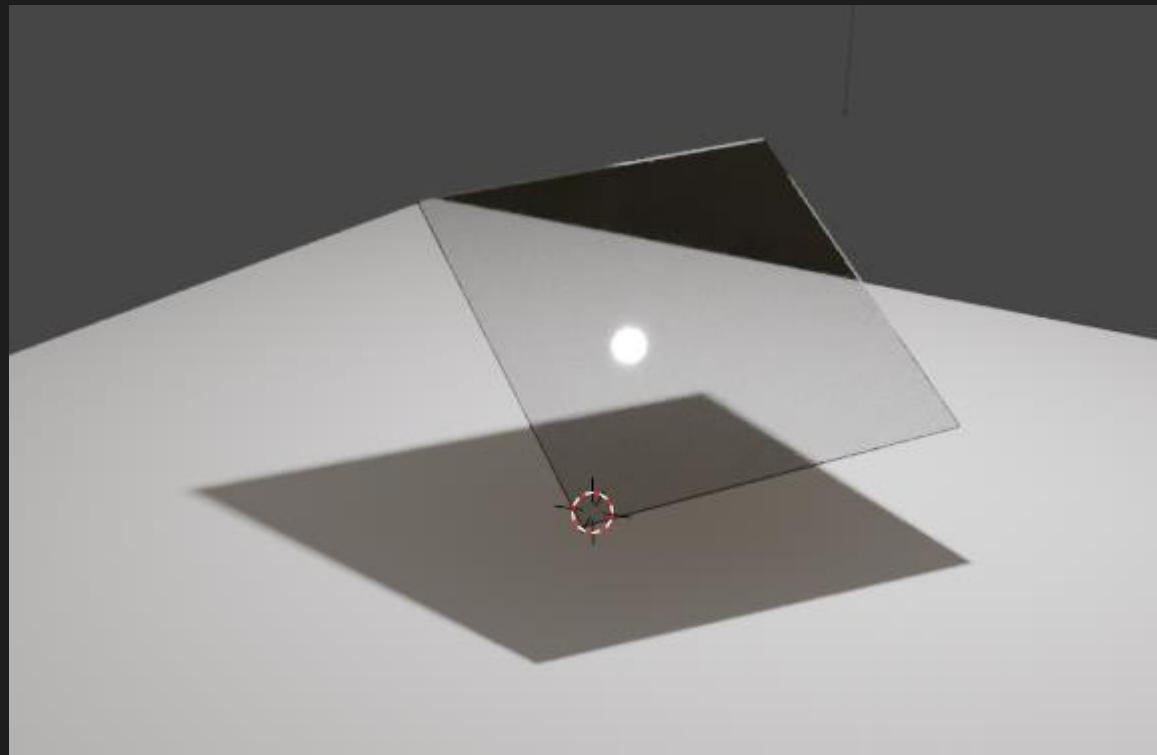
Eevee는 설정 2개를 켜주어야 작동합니다.



Cycles는 더 자연스럽지만...

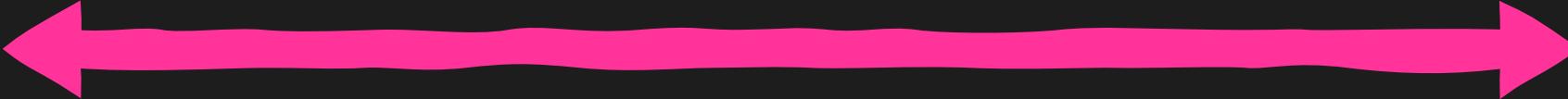
유리재질은 생각보다 다루기 까다롭습니다

이는 사이클이 렌더링을 하는 원리와 연관됩니다



Cycles는 더 자연스럽지만...

투과성



유리창



그냥 유리컵



빛의 굴절

나는 유리!



유리창 != 유리



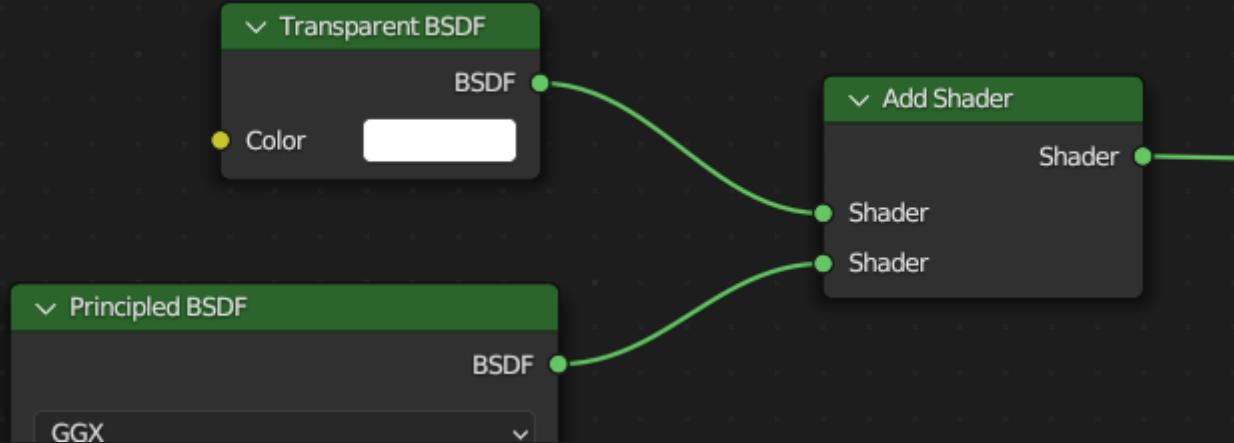
굴절 + 반사 (= Transmission)

VS

투과 + 반사 (Transparent + ??)

Add Shader

Transparent BSDF 를 '더하기'

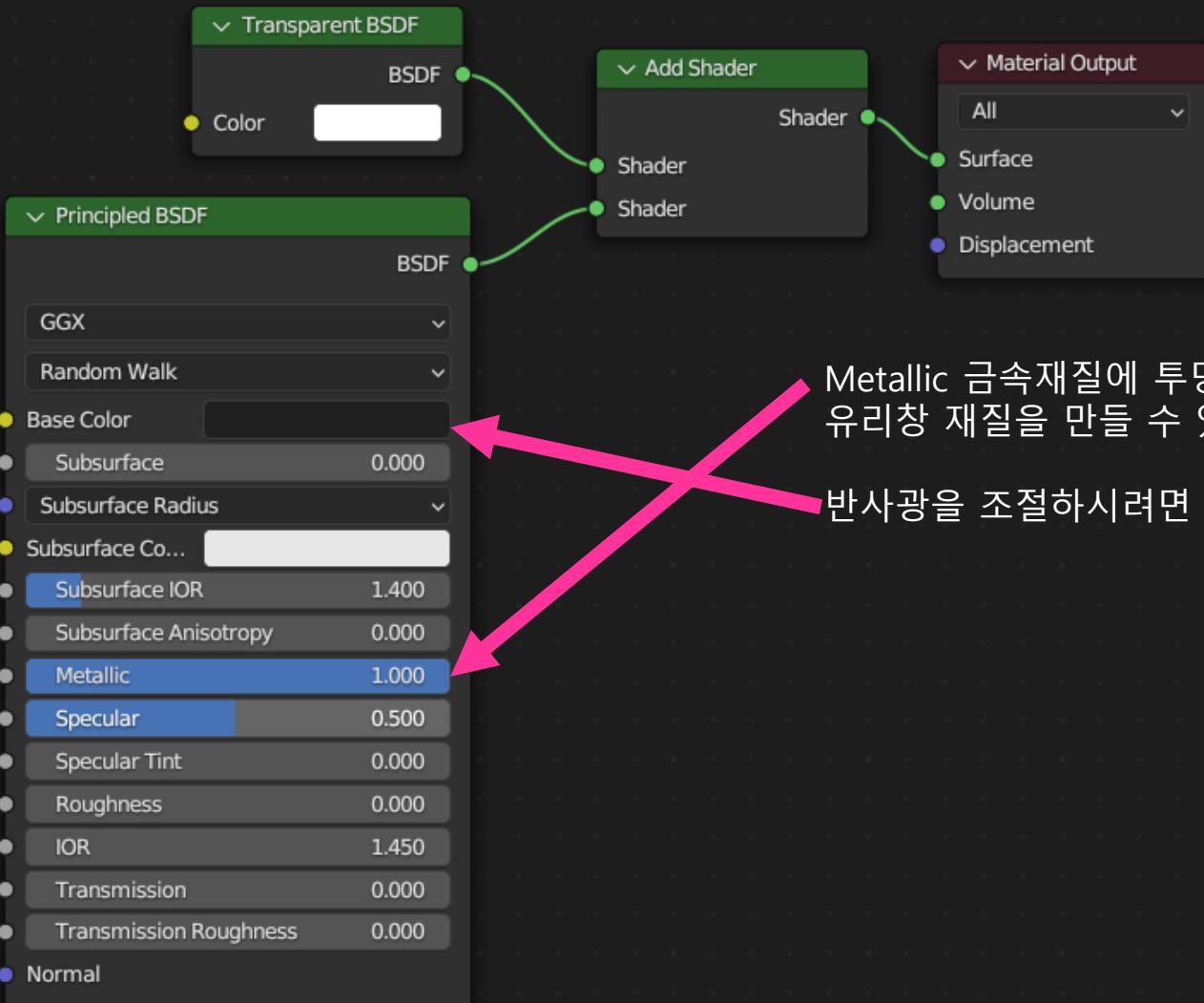


투명을 더한다는게 대체 어떤 의미인가 싶지만..
두 셰이더가 만들어내는 이미지를 더한다는 느낌으로 작동합니다.

그렇기 때문에 오브젝트 뒤쪽은 그대로 비치면서,
(transparent에 의한 효과)

+
반사광 같은 효과가 그 위에 입혀집니다.
(Principled BSDF에 의한 효과)

유리창 재질



Metallic 금속재질에 투명을 더하면, 투과된 배경에 반사광이 더해진 유리창 재질을 만들 수 있습니다.
반사광을 조절하시려면 Base Color의 밝기를 바꾸시면 됩니다.



Alpha_XXX

Eevee에서의 투명 설정

Opaque

Opaque : 불투명한 재질일 때

Alpha Clip

Alpha blend : 가장 평범한 투명 재질.

Alpha Hashed

Alpha Clip : Threshold를 기준으로 완전 투명/ 완전 불투명을 나눕니다.

Alpha Blend

Alpha hashed : 투명을 만들기 위해 샘플을 사용합니다.

이 방법은 다음 페이지에서 볼 수 있는 Alpha blend 문제가 없지만, 샘플이 많이 필요합니다.

Blend Mode

Alpha Blend가 가장 잘 작동하지만

Show Backface를 켜면..



Alpha Blend는 카메라 방향으로 오브젝트가 늘어서 있는 순서에 따라 차례대로 이미지를 합성합니다.
단일 오브젝트 내의 여러 면들은 **순서**가 헷갈리게 되어, 왼쪽처럼 이상하게 출력됩니다.

따라서 오브젝트가 한쪽 면만 있는 Plane이 아니라면, 보통 Show Backface는 끄고 사용하게 됩니다.

Caustics

유리가 빛을 받으면, 일종의 렌즈 역할을 해서 빛을 모으거나 퍼트립니다.



진짜 Caustics

블렌더 3.4부터 Caustics를 표현할 수 있습니다.

다음의 설정이 완료되어야 합니다 :

1. 렌더 엔진은 Cycles
2. Transmission 재질을 사용
3. 조명은 옵션에서 'Cast Shadow'를 켜줍니다.
4. 유리는 'Cast Shadow Caustics'
5. 바닥은 'Receive Shadow Caustics'를 켭니다.

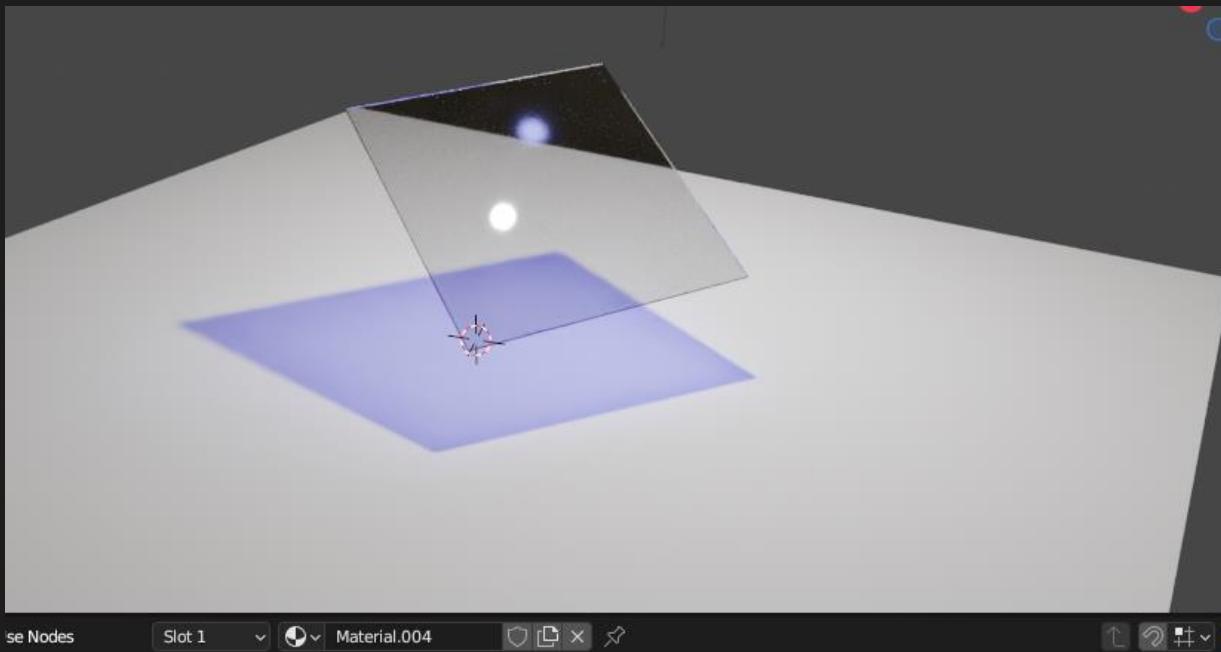
여기까지 설정하였다면 바닥에 커스틱스 무늬가 생길 것입니다.



The image shows the Blender Properties panel on the right side of the interface. It includes sections for Light, Shading, and Caustics.

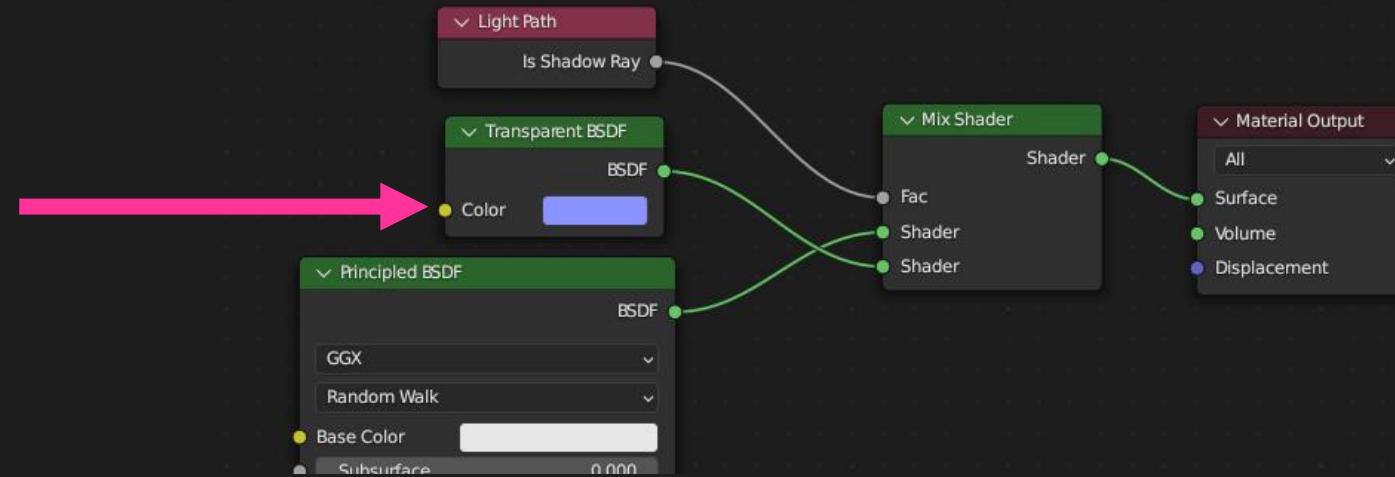
- Light Section:** Shows a Point light source with settings: Color (white), Power (1000 W), Radius (0.1 m), and Max Bounces (1024). Three checkboxes are checked: Cast Shadow, Multiple Importance, and Shadow Caustics. A pink arrow points from the 'Cast Shadow Caustics' checkbox here to the corresponding setting in the Caustics section.
- Shading Section:** Contains Motion Paths, Motion Blur, and a Shading sub-section with options for Shadow Terminator, Fast GI Approximation, and Caustics. Under Caustics, the 'Cast Shadow Caustics' checkbox is checked, and the 'Receive Shadow Caustics' checkbox is unchecked. Two pink arrows point from the checked 'Cast Shadow Caustics' checkboxes in both the Light and Shading sections to these respective checkboxes in the Caustics sub-section.

가짜 Caustics



Scene Nodes Slot 1 Material.004

투명 재질에 텍스쳐를 꽂으면
독특한 효과를 만드실 수 있습니다.



012강 비디오 텍스쳐

비디오(혹은 Image Sequence)
를 이용한 불과 연기 효과



비디오 텍스쳐

<https://pixabay.com/videos>

<https://www.pexels.com>

<https://pixabay.com/videos/fire-flame-heat-burn-power-warm-6766/>

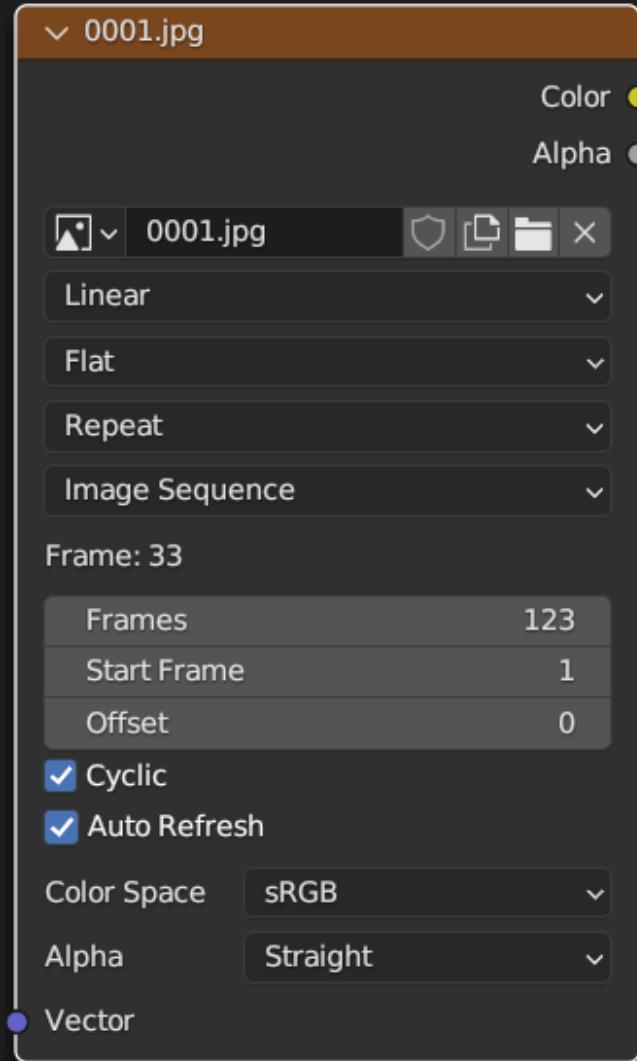
<https://www.pexels.com/ko-kr/video/8943207/>

<https://www.pexels.com/ko-kr/video/8943205/>

페이지가 열리지 않는 경우 다른 비디오를 검색해서 다운로드 받으시거나,
구하실 수 없다면 강의자료에 첨부된 동영상을 사용해주세요.

비디오 텍스쳐

Movie 혹은 Image Sequence



Frames : 비디오의 전체 프레임수를 입력합니다.

Start Frame : 비디오가 현재 씬의 몇번째 프레임부터 재생될지 선택합니다.

Offset : 비디오가 어느 프레임부터 시작할지에 대한 오프셋을 입력합니다.

EX) 총 13 프레임인 비디오 파일을,

Frames 13

Start Frame 10

Offset 5으로 입력 :

| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Scene | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| Video | | | | | | | | | | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | |

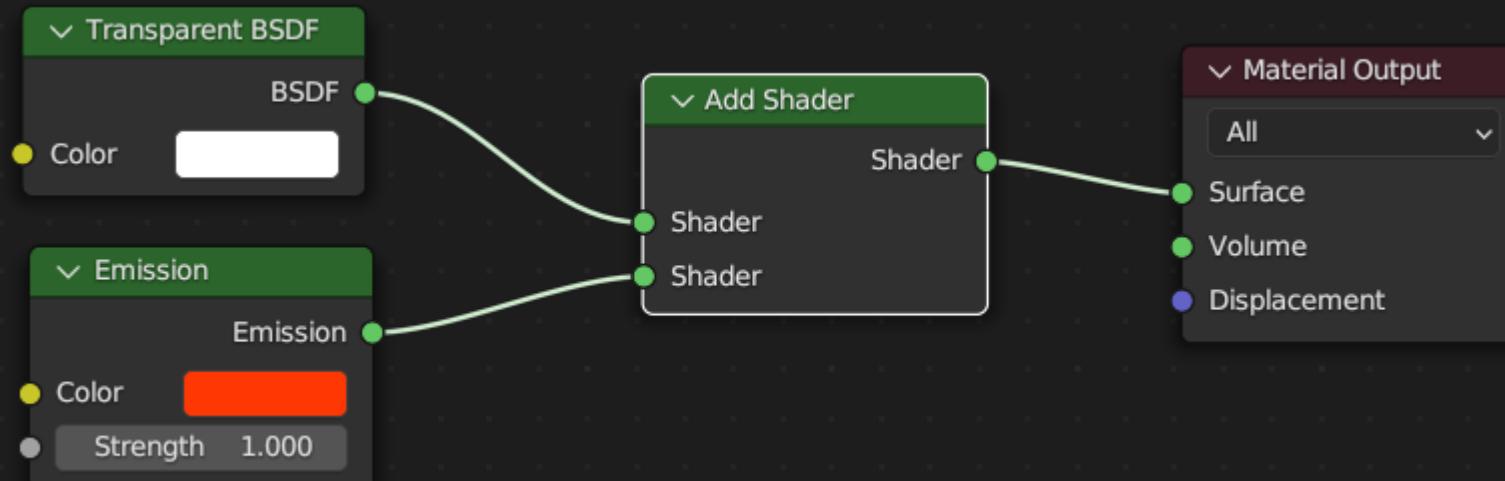
▲ 여기부터 비디오의 13프레임을 읽음-----

Cyclic : 비디오를 전체 프레임에 걸쳐 반복합니다.

Auto Refresh : 비디오의 재생을 뷰포트에서 확인합니다.

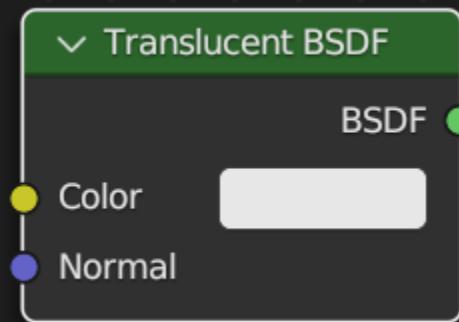
Add Shader

투명과 Emission을 섞으면, 뒷 배경은 그대로 비치면서, (Transparent)
밝게 빛나는 효과가 그 위에 추가됩니다. (Emission)



Translucent

오브젝트 뒤쪽의 밝기를 투영합니다



Subsurface Scattering과 비슷하게, 투과하는 재질이 만들어집니다.
(원리상으로 Subsurface보다는 매우 거친 유리와 더 비슷합니다.)

연산이 더 가볍고, 한쪽면만 있어도 작동합니다.

오직 뒤쪽에서 온 빛만을 반영하므로, 앞쪽 빛에 영향을 받지 않습니다.
따라서 일반적인 재질 (principled나, diffuse)과 섞어서 사용합니다.

013강 텍스쳐를 믹스하는 여러가지 방법들

Normal과 Dot Product

여러개의 UVMap 사용하기

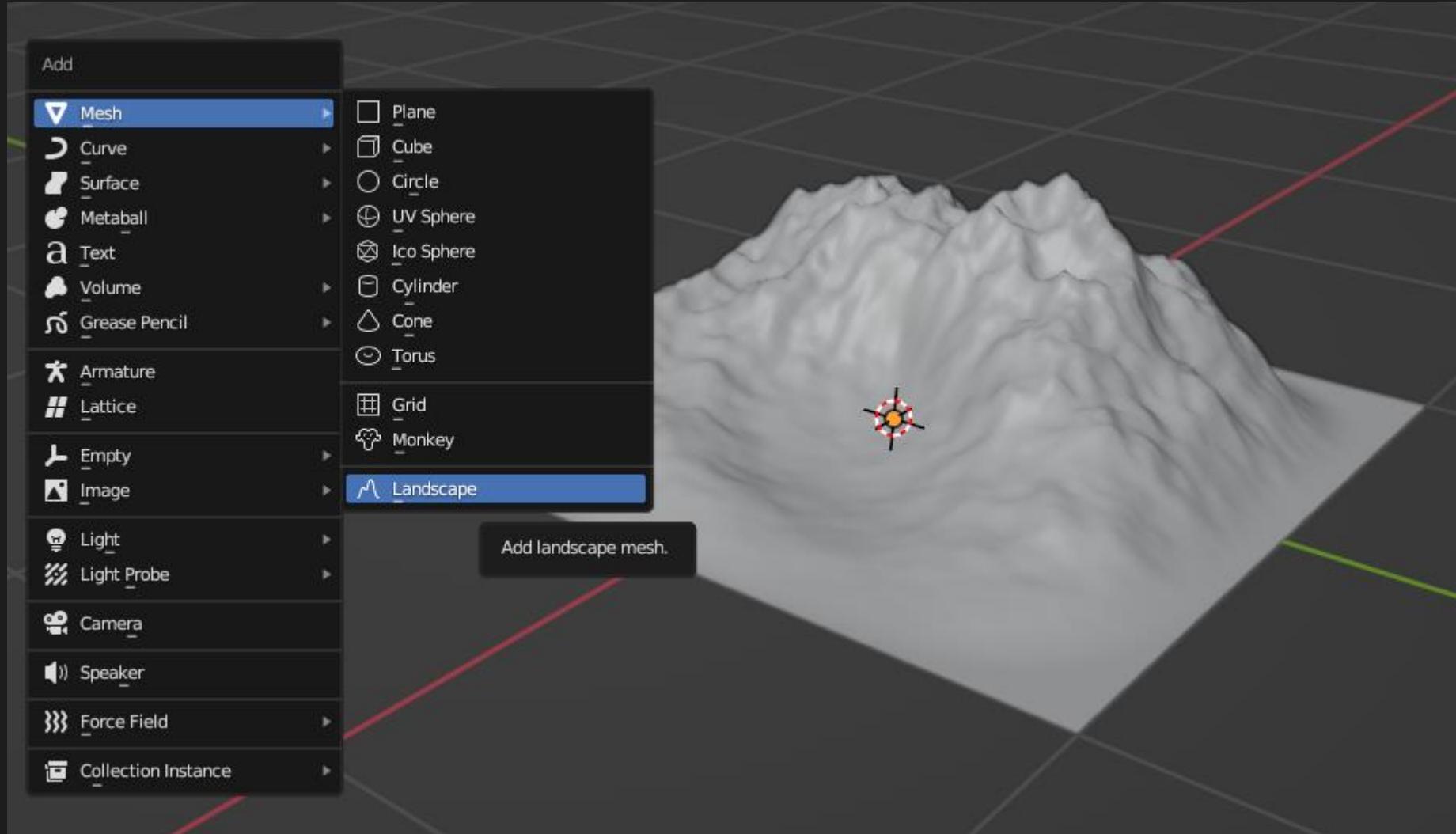
Vertex Color



A.N.T.Landscape

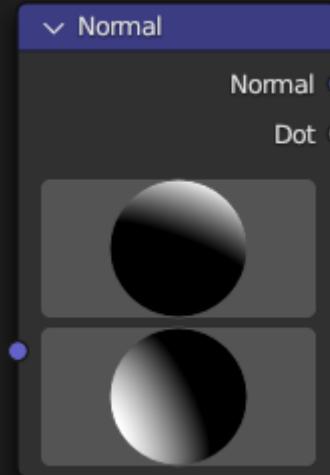
Musgrave Texture를 이용하여 지형을 생성하는 내장 애드온입니다.

Displace modifier를 사용하는 것보다 좀 더 편리합니다.



Normal 방향 컨트롤

Normal node



- 입력받은 벡터와의 dot product를 계산합니다.
- 구는 위쪽에서 바라본 것입니다. 정면 방향이 z축, 오른쪽이 x축, 위쪽이 y축입니다.
- Dot : 입력값과의 Dot product (= Inner Product, 내적) 을 계산합니다.

Dot Product(내적)

내적은 벡터의 곱셈중 하나입니다.

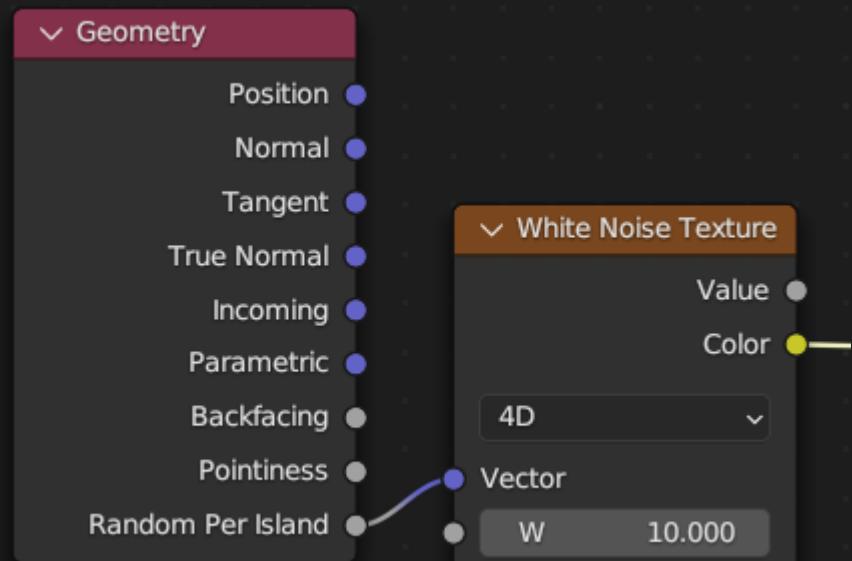
- 방향이 같으면 두 벡터의 크기곱과 같고, 방향이 반대이면 크기곱에 음수 부호를 붙인 것과 같습니다.
- 곱하는 벡터가 서로 수직이면 0이 됩니다.

Random Per Island (Cycles Only)

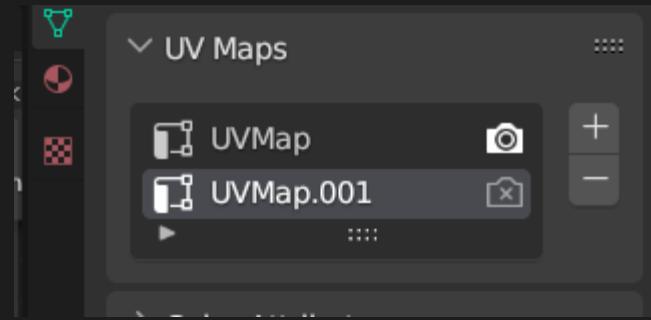
Geometry 노드의 Random Per Island는 분리되어 있는 폴리곤마다 랜덤 컬러를 출력합니다.

Object Info 노드의 Random소켓과 비슷하지만, 하나의 오브젝트 내에도 작동한다는 점이 다릅니다.

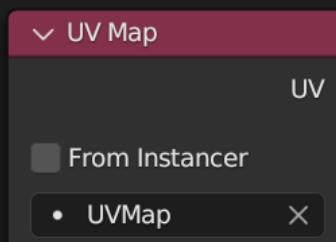
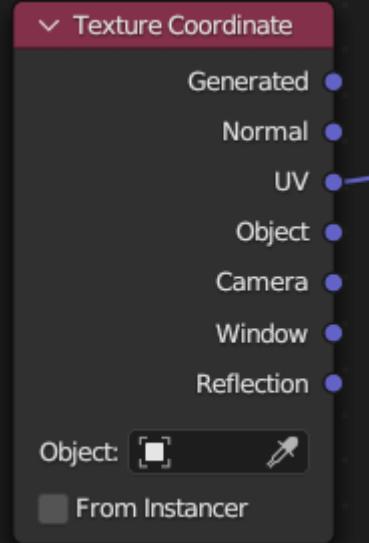
출력값은 흑백이므로 Object Info 노드처럼 White Noise Texture와 함께 사용하면 유용합니다.



다중 UVMap



오브젝트는 여러 개의 UV맵을 가질 수 있습니다.

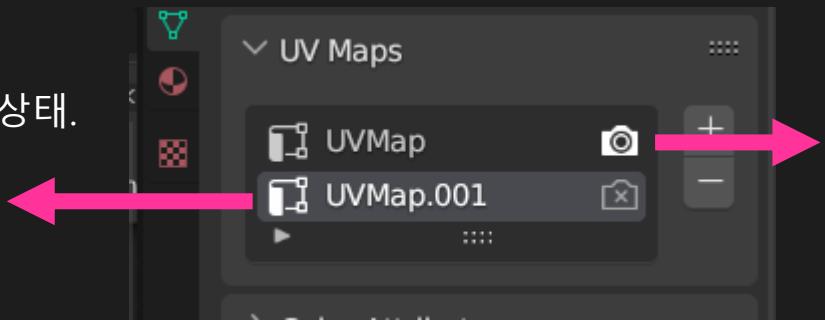


UVMap의 'Select' 와 'Active'

Select

UVMap을 클릭하여 밝게 하이라이트 된 상태.

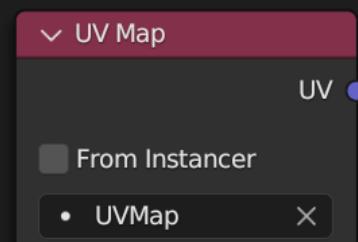
- UV에디터 창에선 이게 보입니다
- 솔리드 모드에서의 텍스쳐 매핑
- Bake할 때도 이걸 사용합니다



Active

오른쪽의 카메라 버튼을 누른 UV

- Texture Coordinate의 UV 소켓은 이걸 사용합니다
- Image Texture 노드가 기본으로 사용하는 UV입니다

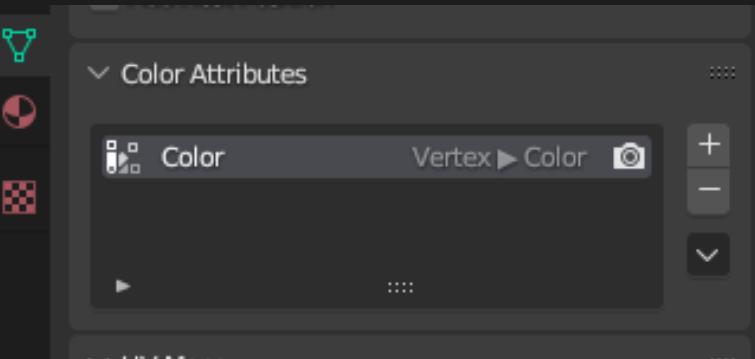


직접 UV를 고르고 싶을 때는 UVMap 노드를 사용합니다.

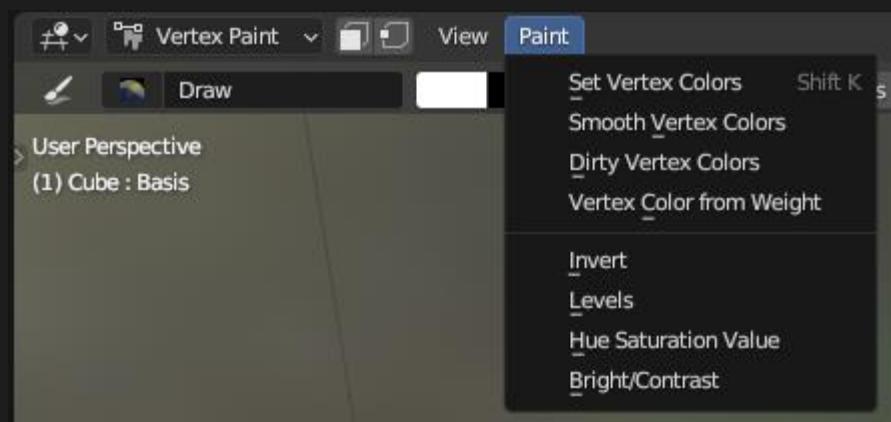
Color Attribute (= Vertex Color)

Vertex에 색깔을 지정합니다

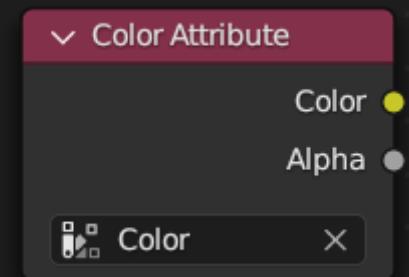
각각의 Vertex에 색깔을 지정할 수 있습니다.



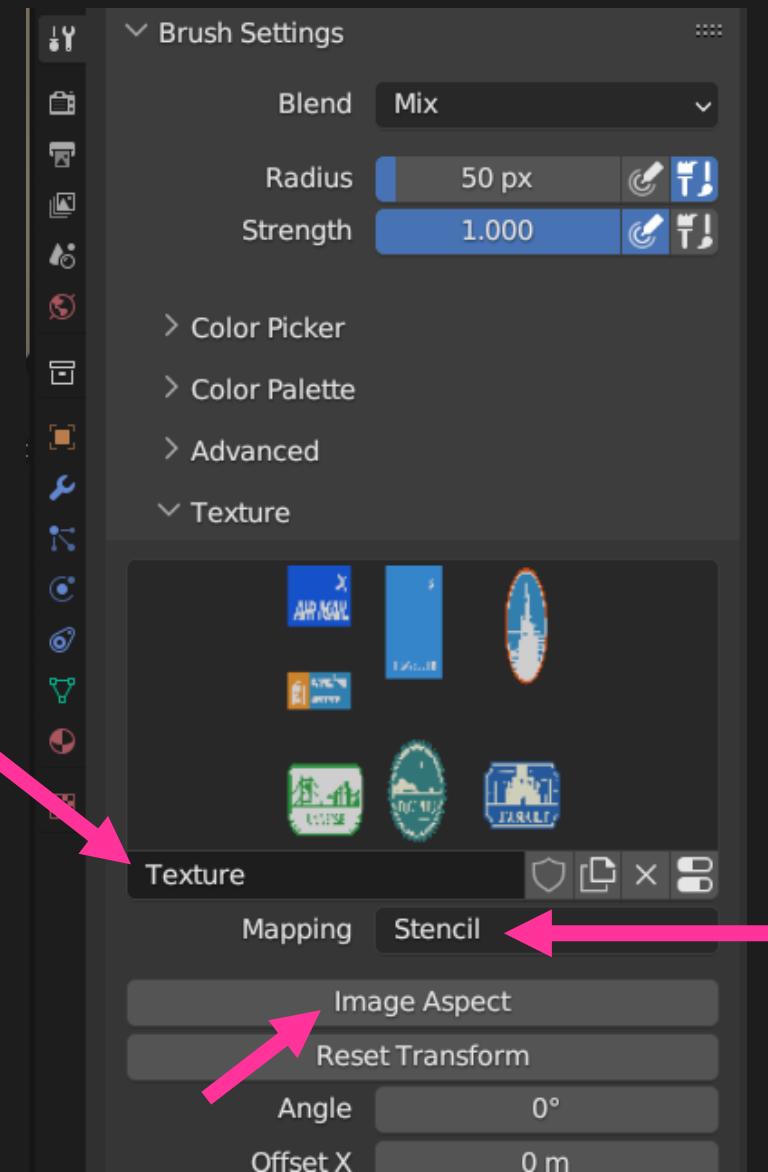
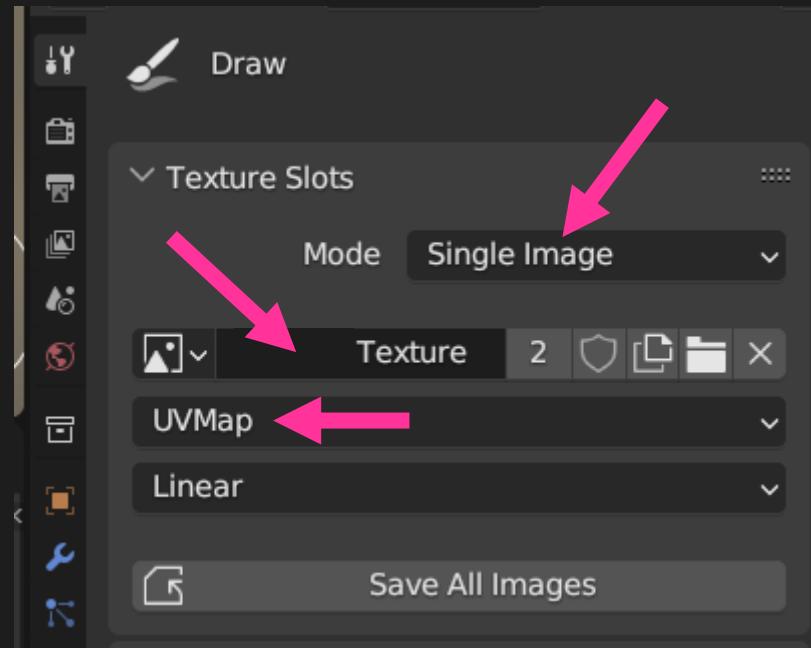
Vertex Paint에서,
Paint – Vertex Color from Weight로
Vertex Group을 Vertex Color로 가져올 수 있습니다.



Color Attribute 노드를 통해
셰이더로 가져올 수 있습니다.



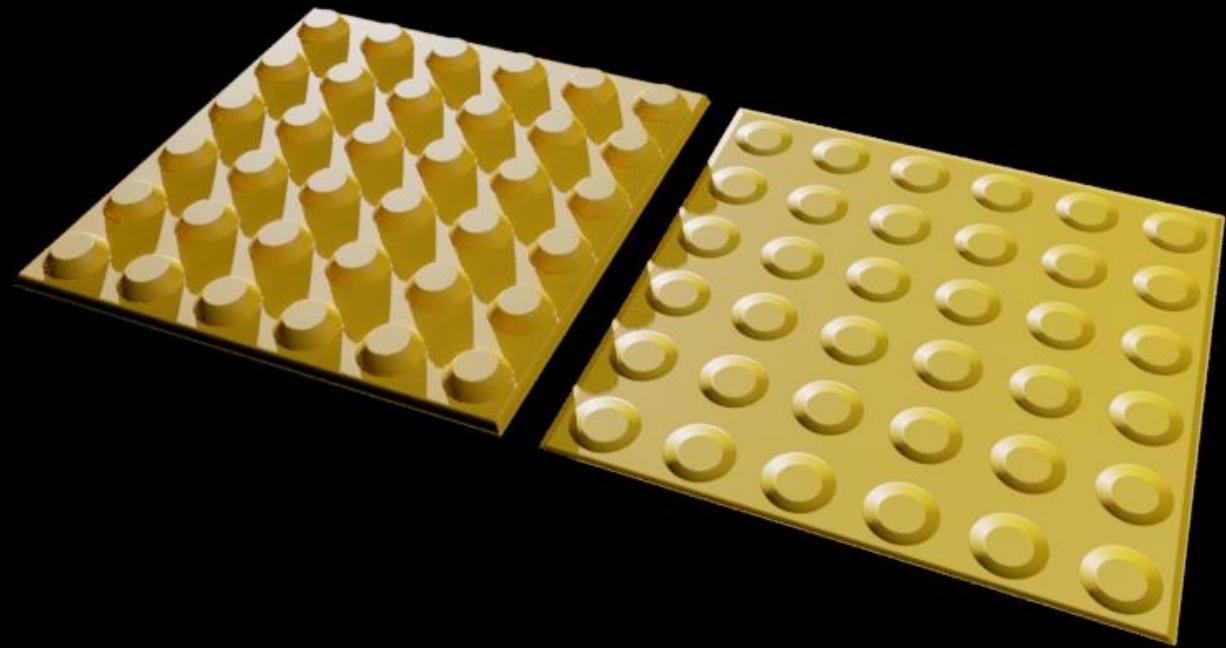
Texture Paint (Stencil)



014강 튀어나오는 재질

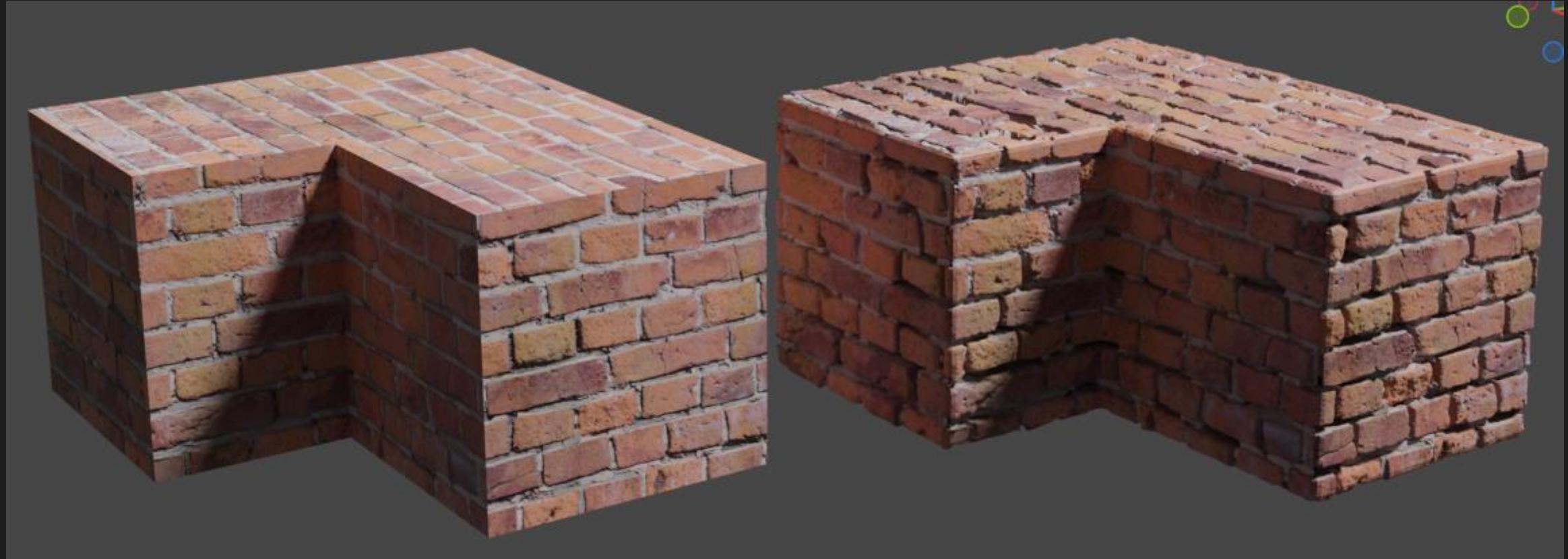
Height Map (Displacement)의 의미

Cycles와 Eevee에서의 Displacement 사용법



벽돌은 왜 어색할까요?

매끈한 표면의 한계



돌출이 필요한 경우

표면 자체의 특징

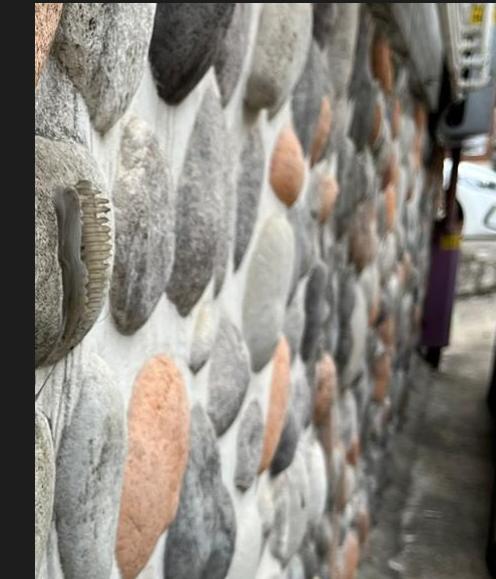
표면 자체가 평면 위에 있지 않은 경우
노멀맵과 범프맵만으로는 표현이 어렵습니다.



보는 각도에 따라

보는 각도에 따라서 돌출을 표현하지 않아도 문제되지 않을 수 있습니다.

하지만 낮은 각도에서 바라보게 되면 문제가 달라집니다.



Displacement

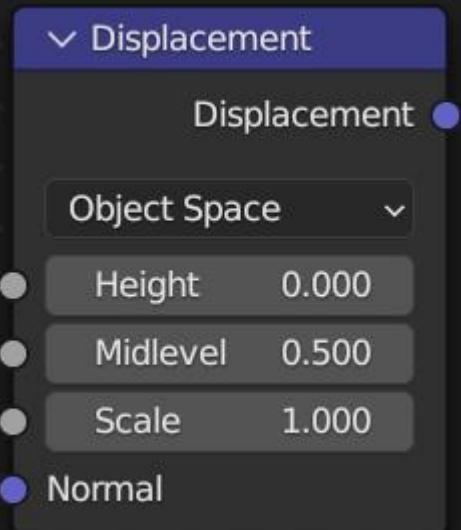
높이를 나타내는 이미지



Displacement 이미지를 Displacement 노드의 Height에 꽂아서 오브젝트의 높낮이를 나타낼 수 있습니다.

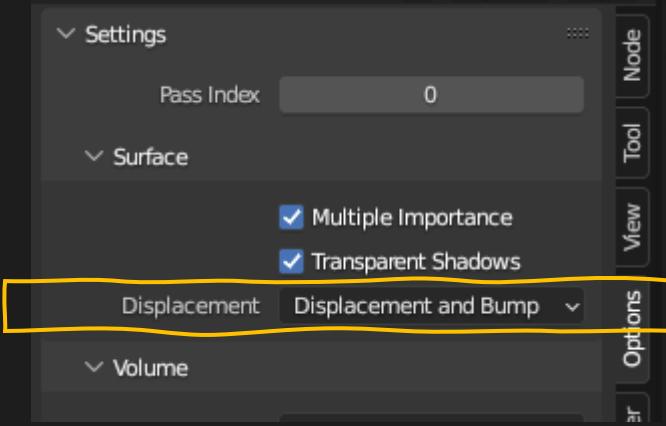
Midlevel은 기준위치로, 이 값보다 크면 밖으로 튀어나오고, 이 값보다 작으면 안으로 들어가게 만듭니다.

Displacement는 Eevee에서 동작하지 않고, Cycles에서도 바로 작동하지 않습니다.
다음 페이지에서 작동방법을 확인해 주세요.



Displacement

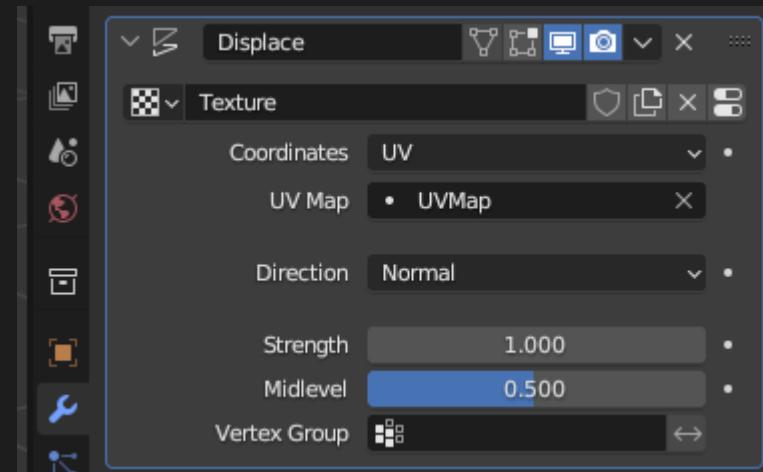
Cycles 설정 : 재질 옵션의 Displacement를 켜 주어야 합니다.



Displacement only : Displacement 맵이 오로지 물리적인 높낮이만을 만들어줍니다.

Displacement and Bump : Displacement가 Bump 효과를 추가로 생성하여 디테일을 더해줍니다.

Eevee 설정 : Eevee는 Displacement 노드가 작동하지 않습니다. (3.5 기준)



대신, Displace modifier를 사용합니다.

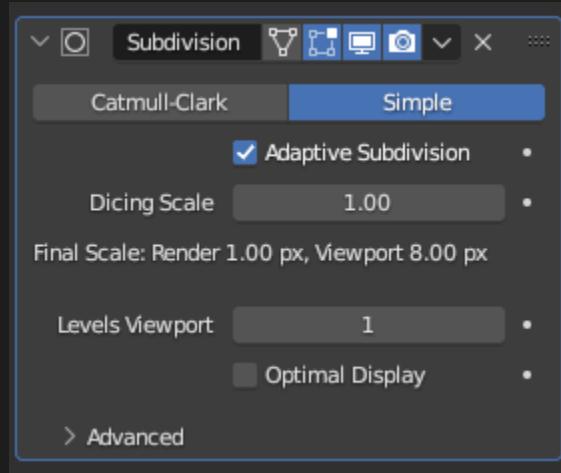
사용하는 텍스쳐를 Texture 설정에서 새로 생성 후 입력하고, 올바른 좌표를 선택해 줍니다.

Eevee의 Displacement는 셰이더 노드 방식이 아니므로, 원치 않는 결과가 나올 수 있습니다.

UVMAP을 사용하는 이미지텍스쳐가 아닌 경우 사용에 주의.

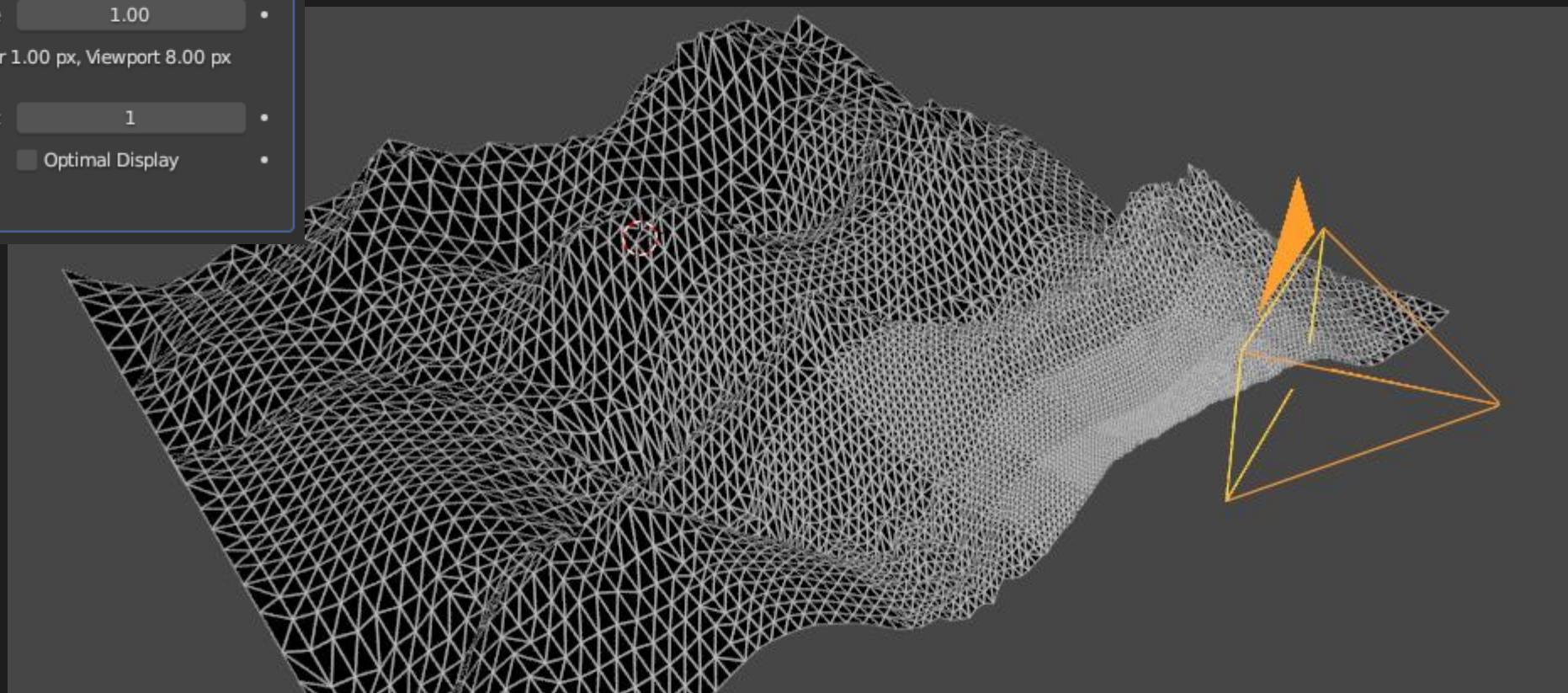
Adaptive Subdivision

거리에 따라 섭디비전을 조절합니다

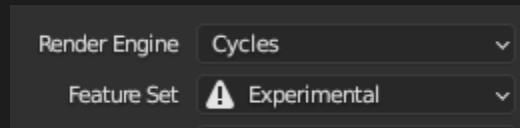


Dicing Scale : 카메라에서 보이는 면 하나가 얼마나 클 지 **픽셀 단위로** 입력합니다.

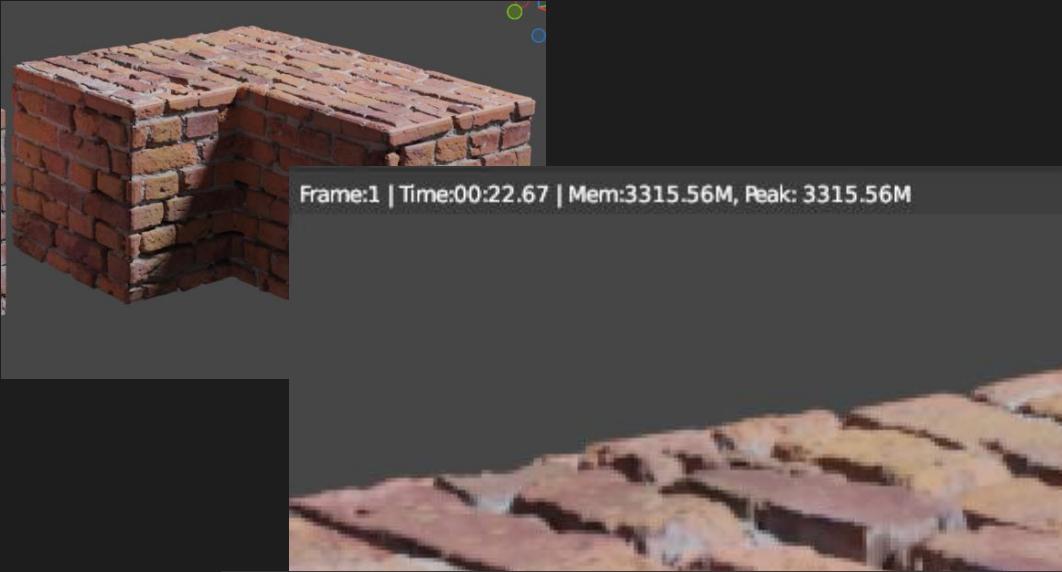
따라서, 면이 얼마나 나눠질지는 해상도에도 영향을 받습니다.



※ Adaptive Subdivision을 켜려면,
Feature Set이 Experimental이어야 합니다.



Displace의 문제점



메모리가 많이 필요합니다.

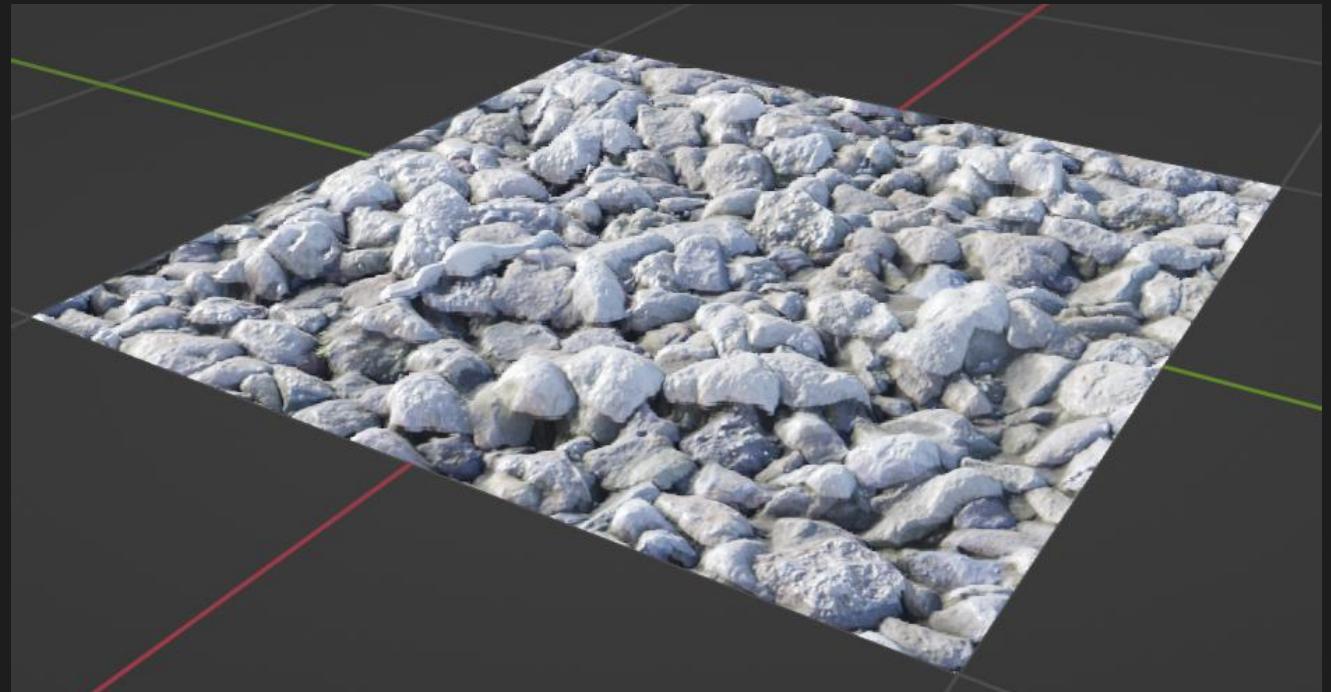
| Mem:3315.56M, Peak: 3315.56M

Parallax Occlusion Mapping

좌표를 왜곡시켜 마치 튀어나온것처럼 보이게 하는 방법입니다. 실제 메쉬를 생성하지 않으므로 Displacement보다 메모리가 훨씬 적게 필요합니다.
블렌더에 공식적으로 추가된 기능은 아닙니다. (3.5기준)

여러가지 관련 애드온이 존재하므로 블렌더마켓 등에서 검색해보세요.

※공식 기능으로 추가될 계획이 있는 것으로 알고 있습니다만, 그것이 언제일지는 알려지지 않았습니다.



015강 Procedural Texture 입문

Procedural Texture의 장점과 단점
기본 수학 지식 훑어보기

Procedural Texture의 장점과 단점

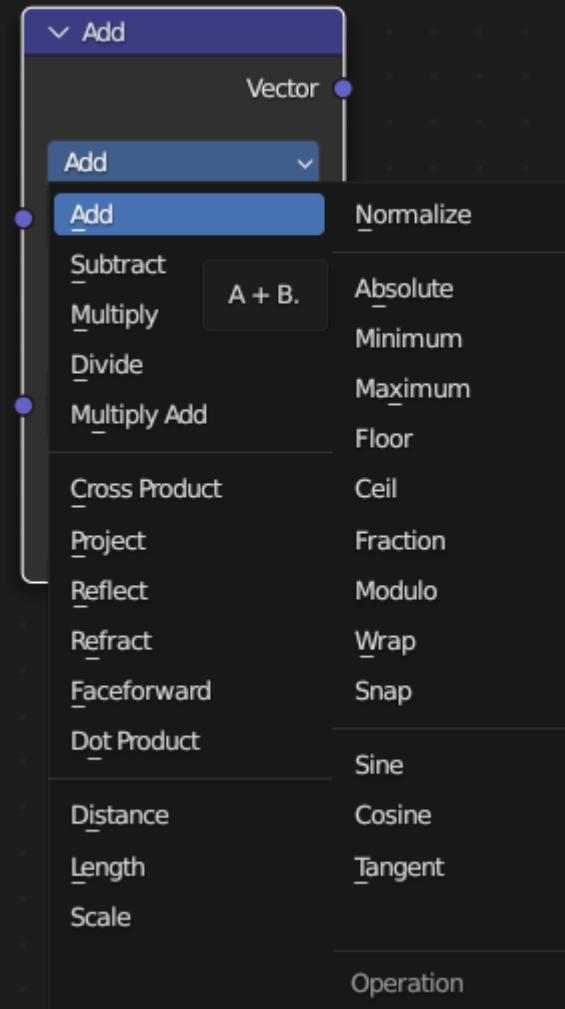
| | Image | Procedural |
|--------|-------|------------|
| 해상도 | ▼ | ▲ |
| 제작 난이도 | (▼) | (▲) |
| 메모리 | ▲ | ▼ |
| 연산 | ▼ | ▲ |

MATH NODE

The screenshot shows a node-based interface with a sidebar on the left and a main panel on the right. The sidebar has a tree view with 'Add' selected, which has a child node 'Value'. Below this, a dropdown menu is open, showing the 'Add' option as selected. The main panel is titled 'Add' and contains five tabs: Functions, Comparison, Rounding, Trigonometric, and Conversion. The 'Functions' tab is active and displays a list of mathematical operations:

| Functions | Comparison | Rounding | Trigonometric | Conversion |
|---------------------|----------------|-----------|--------------------|------------|
| Add | Minimum | Round | Sine | To Radians |
| Subtract | Maximum | Floor | Cosine | To Degrees |
| Multiply | Less Than | Ceil | Tangent | |
| Divide | Greater Than | Truncate | Arcsine | |
| Multiply Add | Sign | Fraction | Arccosine | |
| Power | Compare | Modulo | Arctangent | |
| Logarithm | Smooth Minimum | Wrap | Arctan2 | |
| Square Root | Smooth Maximum | Snap | Hyperbolic Sine | |
| Inverse Square Root | | Ping-Pong | Hyperbolic Cosine | |
| Absolute | | | Hyperbolic Tangent | |
| Exponent | | | | |

VECTOR MATH NODE



수식 사용

노드의 숫자 입력창에서 기본적인 계산을 할 수 있습니다.
예를 들어 1/3을 입력하면 자동으로 0.3333... 이 입력됩니다.

상수

pi : 원주율 3.141..

e : 자연상수 2.718...

tau : 2π = 6.283...

함수

기본적인 사칙연산 : +, -, *, /

거듭제곱 : **

루트 : sqrt(x)

삼각함수 sin(x), cos(x), tan(x) asin(x), acos(x), atan(x), atan2(x,y).....

지수함수 exp(x)

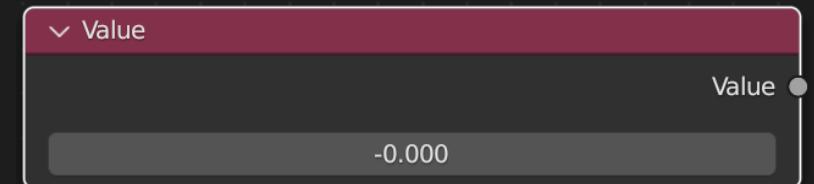
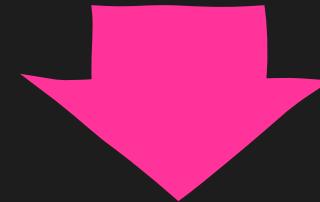
라디안 - 육십분법 변환 degrees(), radians()

최대, 최소 max(x,y,z,.....) min(x,y,z,.....)

등등..



```
3*sin(radians(30))**2+cos(pi/3)**2-log(e)
```



```
-0.000
```

MATH NODE

Add

덧셈은 이동의 성질을 가지고 있습니다.

이미지는 0에서 1 사이의 값을 가지기 때문에, 이때 덧셈은 항상 양의 방향의 이동 = 밝아지는 성질을 가집니다.

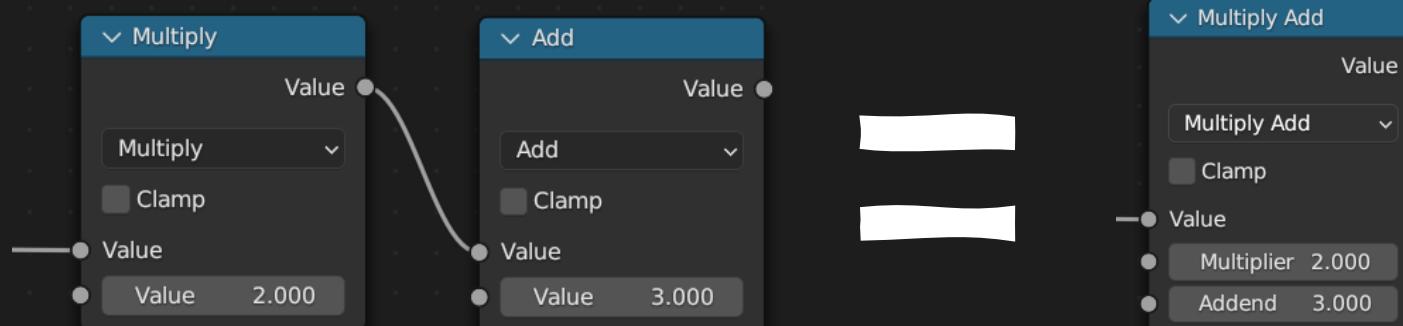
Multiply

곱셈은 확대/축소의 성질을 가지고 있습니다.

이미지는 0에서 1 사이의 값을 가지기 때문에, 이때 곱셈은 항상 소수점의 곱셈 = 어두워지는 성질을 가집니다.

Multiply Add

곱셈과 덧셈을 한번에 처리합니다.



※Multiply Add는 곱셈을 먼저 함에 유의하세요.

MATH NODE

Subtract

블렌더에서 대부분의 값의 범위는 0에서 1 사이이기 때문에, Subtract를 이용하여 $1-x$ 를 해주면 Invert와 동일한 효과를 얻을 수 있습니다.

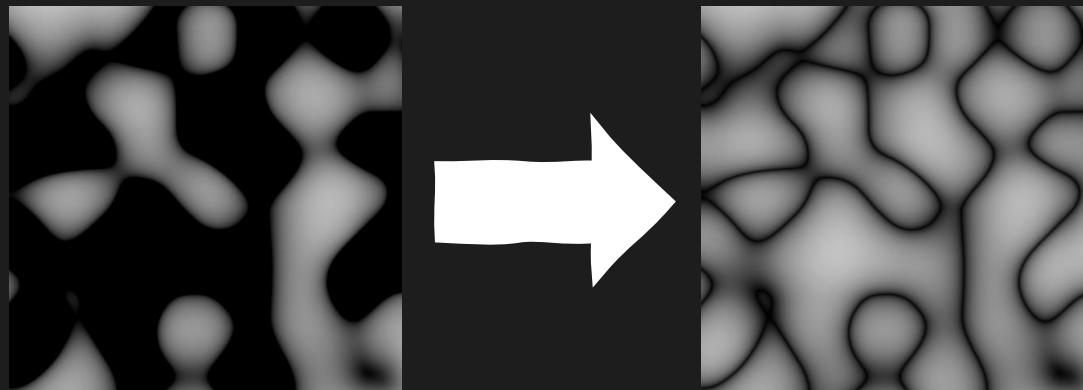
Logarithm

로그를 계산합니다.

Square Root

제곱근 ($=0.5$ 제곱) 을 계산합니다.

Absolute 음수의 부호를 바꾸어 양수로 만듭니다.



Inverse Square Root

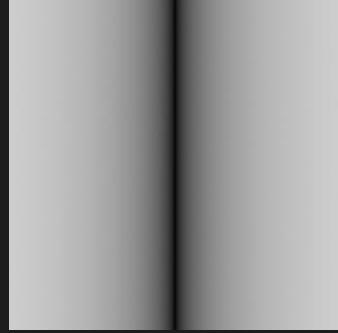
제곱근의 역수를 계산합니다.

Exponent

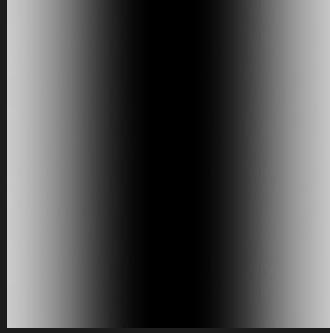
밑이 2인 지수함수를 계산합니다.

MATH NODE

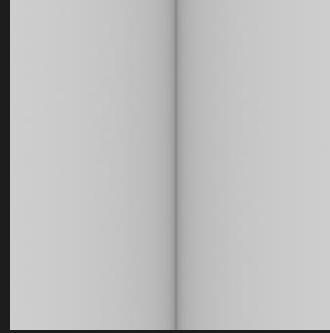
Power 거듭제곱입니다.
0~1사이의 값에 대한 경향성(밝은 쪽, 어두운 쪽) 을 결정합니다.



원본



4제곱



$\frac{1}{4} = 0.25$ 제곱

Power의 exponent연결

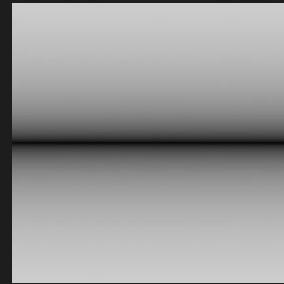
MATH NODE

Minimum, Smooth Minimum

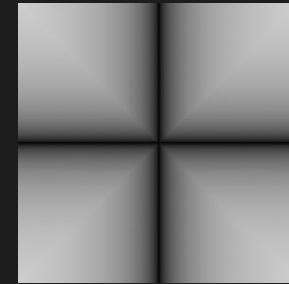
입력받은 두 값중 작은 쪽을 내보냅니다. Smooth Minimum 은 두 값 사이의 경계를 더 부드럽게 만들어줍니다.



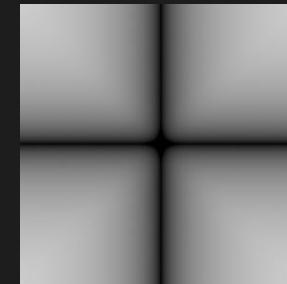
A



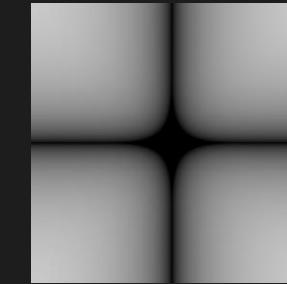
B



Minimum (A,B)



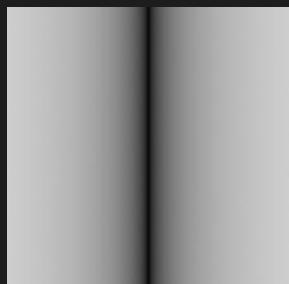
Smooth Minimum
Distance 0.2



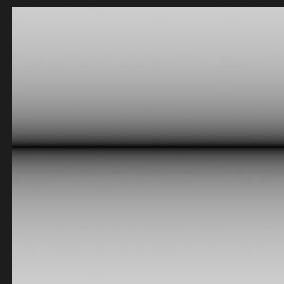
Smooth Minimum
Distance 0.5

Maximum, Smooth Maximum

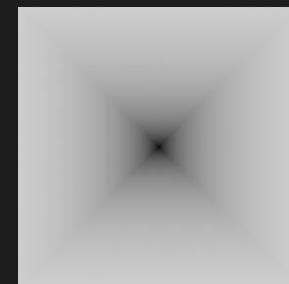
입력받은 두 값중 큰 쪽을 내보냅니다.



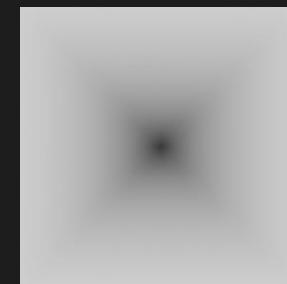
A



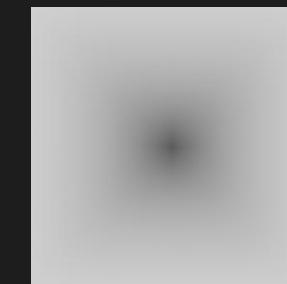
B



Maximum (A,B)



Smooth Maximum
Distance 0.2

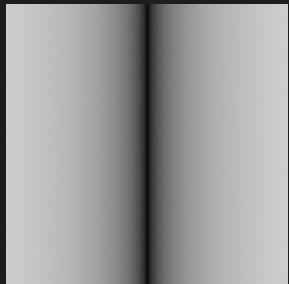


Smooth Maximum
Distance 0.5

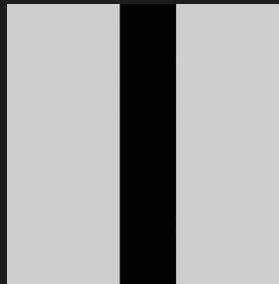
MATH NODE

Less Than, Greater Than : <, >

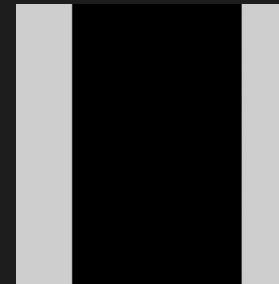
Greater than : Threshold 기준으로 큰 쪽을 1, 작은 쪽을 0으로 나타냅니다. Less than은 그 반대로 작동합니다.



원본



Greater than
Threshold 0.2

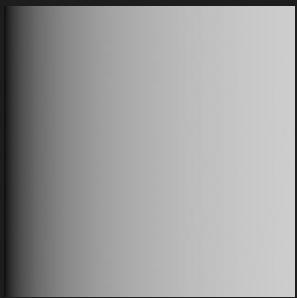


Greater than
Threshold 0.6

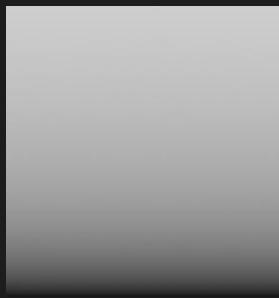
Compare : =

입력받은 두 값이 같은지 판단합니다.

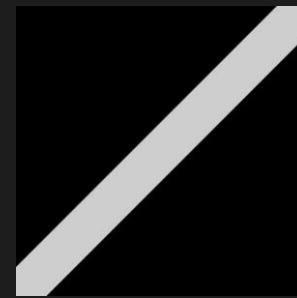
'정확히 같은지' 판단한다면 많은 경우 표현이 곤란하므로, epsilon을 이용하여 '얼마나 비슷한지'를 판단합니다.



X



Y



Y=X (epsilon 0.1)

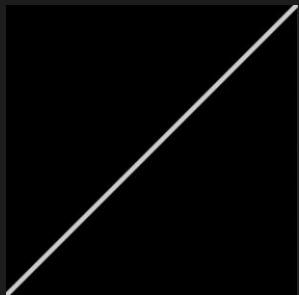
Sign

양수는 1, 음수는 -1을 내보냅니다.

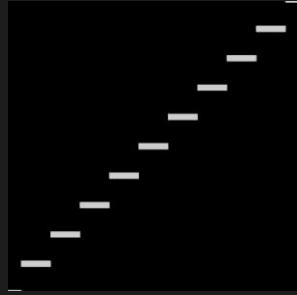
MATH NODE

Round, Floor, Ceil

차례대로 반올림, 내림, 올림입니다.
기준에 따라 정수 부분을 출력합니다.



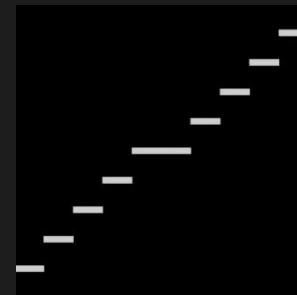
$Y=X$



$Y=\text{Round}(X)$

Truncate

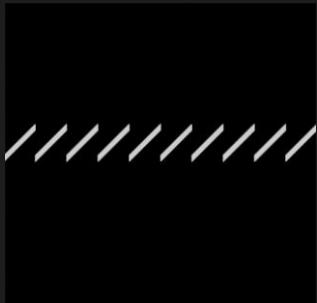
입력받은 값의 정수 부분을 출력합니다.
만약 음수라면, 예컨대 -1.2라면 -1을 출력합니다.
그에 따라 그래프의 모양은..



양수 범위에서는 Floor와 같습니다.

Fraction

입력받은 값의 소수 부분을 출력합니다.
만약 음수라면, 예컨대 -1.2라면 -2+0.8로 보아 0.8을 출력합니다.



그렇기에 Truncate+Fraction 은 – 정수부분+소수부분 이니까 – 원래 숫자가 나올 것 같지만 그렇지 않습니다.
만약 Truncate처럼, -1.2의 소수부분을 -0.2로 인식하고 싶다면 Modulo를 사용하세요!

MATH NODE

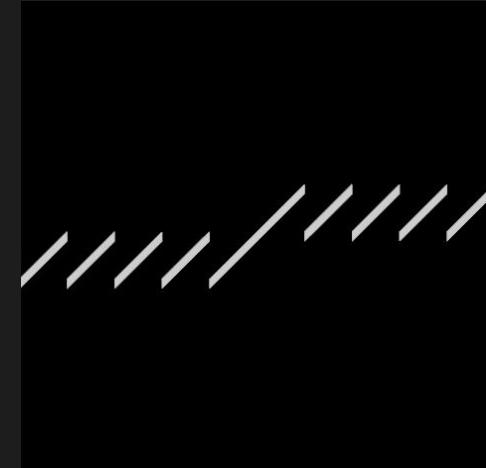
Modulo

Modulo(A,B)는 A를 B로 나눈 나머지입니다.

예컨대, Modulo(5.3,2) = 5.3을 2로 나눈 나머지 = 1.3입니다.

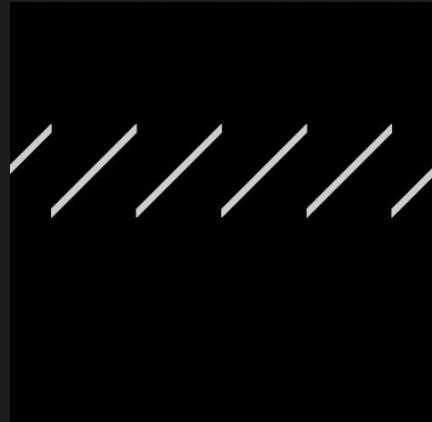
음수일 때 계산에 유의하세요! 피젯수(dividend)가 음수일 때의 작동방식은,
예컨대 Modulo(-5,4)라면 -5를 4로 나눈 나머지 : $-5 = 4 \times (-1) + (-1)$ 로 -1로 계산합니다.

만약 음수일 때도 양수일때와 같은 규칙을 갖게 하고 싶으시면, Wrap을 이용합니다.



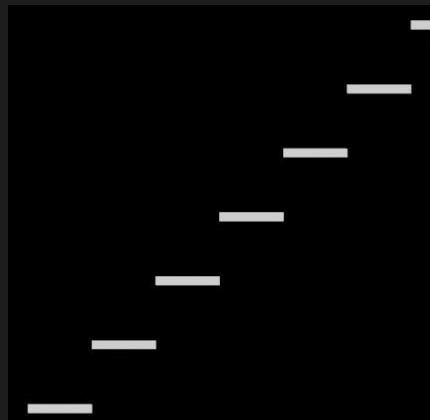
MATH NODE

Wrap

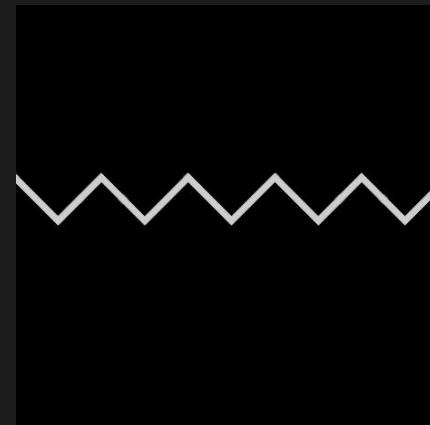


Wrap ($A, \text{max}, \text{min}$)은 A 를 $\text{min} \sim \text{max}$ 사이에서 반복하게 만듭니다.
만약 min 을 0으로 둔다면 Modulo와 같습니다.
다만, 음수일 때도 같은 규칙을 갖는다는 점은 Modulo와 다릅니다.

Snap Snap은 Floor와 비슷하지만 increment에 따라 내림의 범위를 정할 수 있습니다.



Ping-Pong Ping pong은 입력받은 값을 왔다갔다 반복하는 주기함수로 만듭니다.



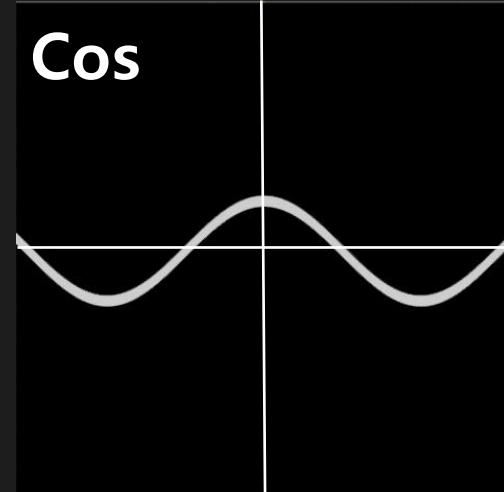
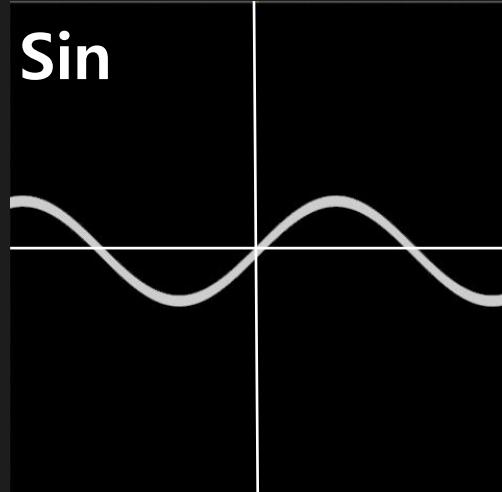
기준은 Scale이며, Scale까지 올라갔다 내려감을 반복하므로 주기는 Scale의 두배입니다.

예컨대, Scale = 0.5라면 0.5까지 올라갔다 내려가므로 주기는 1입니다.

MATH NODE

Sine, Cosine

삼각함수는 기하학적인 의미도 중요합니다만, 일단 그래프 모양만 확인하겠습니다.

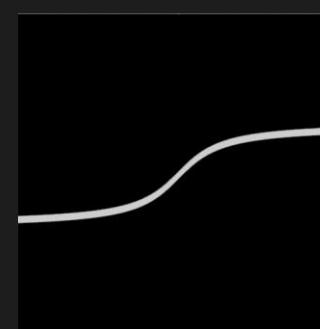
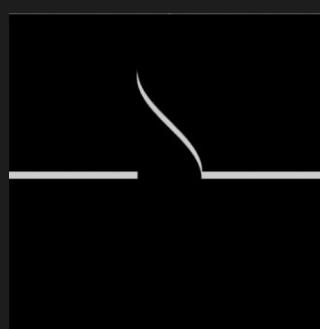
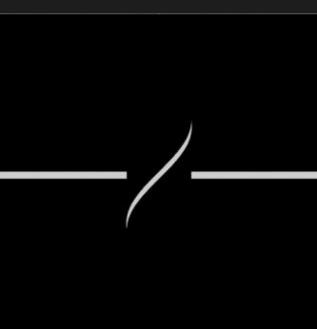
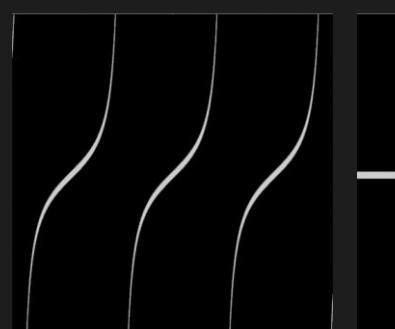


사인과 코사인은 $2\pi = 6.283\dots$ 마다 반복되는 주기함수이며, -1에서 1 사이에 놓입니다.

Tangent, Arcsine,

Arcsine,

Arccosine, Arctangent

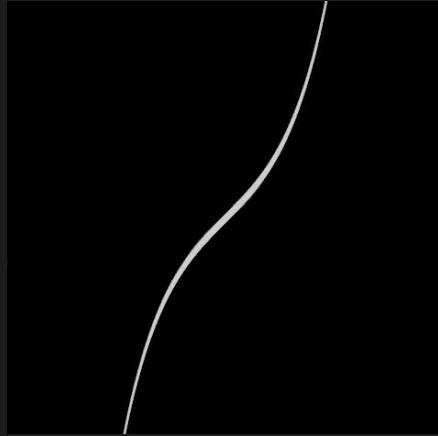


Atan2

Atan2 (y, x)는 y/x 의 역탄젠트 각을 출력합니다.

MATH NODE

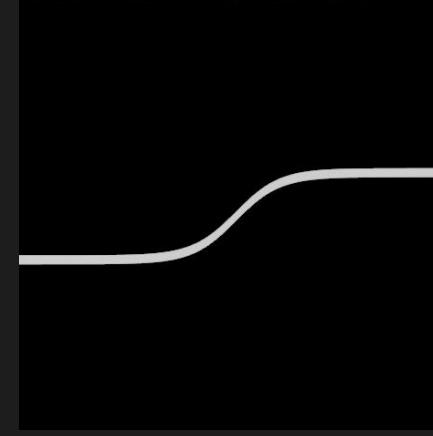
sinh



cosh



tanh



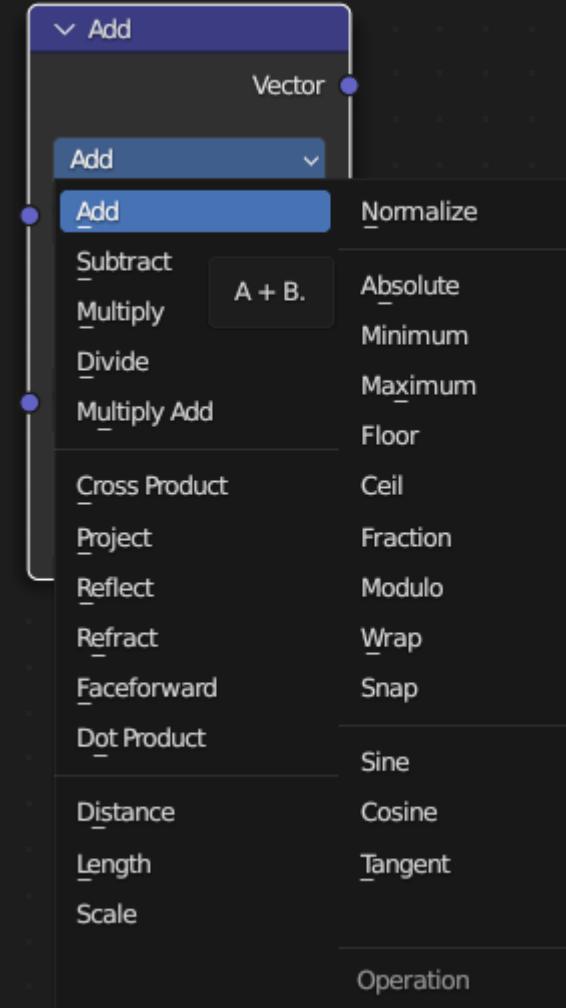
To Radians, To Degrees

육십분법과 호도법을 변환합니다. 예컨대 $\text{To radians}(180) = \pi$, $\text{To degrees } (\pi/3) = 60$ 입니다.

VECTOR MATH NODE

같은 이름의 연산은 물론 Vector Math 에서도 동일하게 작동합니다.

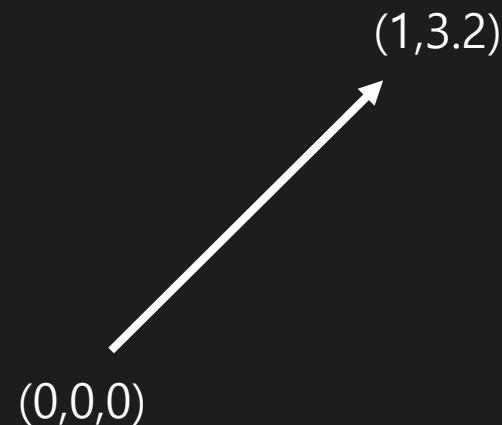
Multiply는 각 성분을 개별적으로 곱할 수 있고,
Scale은 모든 성분에 하나의 값을 한번에 곱할 수 있습니다.



VECTOR

Vector에 대한 개념은 두 가지 다른 방식으로 생각할 수 있습니다.

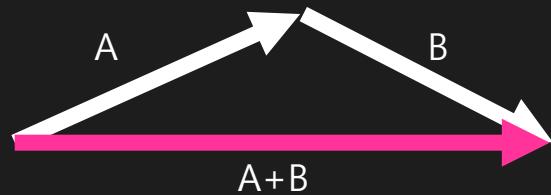
1. 단지 숫자 3개를 묶어놓은 것 : 벡터의 표현에 숫자 3개 이상은 필요하지 않습니다.
벡터 자체에는 첫번째, 두번째, 세번째 숫자가 각각 무엇을 의미하는지 언급되어 있지 않습니다.
그렇기 때문에, 좌표에 꽂으면 위치로, 컬러에 꽂으면 색으로 인식합니다.
2. 원점으로부터 뻗어나온 방향 : 예컨대 $(1,3,2)$ 라는 것은 x축 위의 한 점으로 볼 수도 있지만, 원점으로부터 $(1,3,2)$ 까지 이은
화살표로 생각할 수도 있습니다. 이런 개념은 유용할 때도 있고, 그렇지 않을 수도 있습니다.



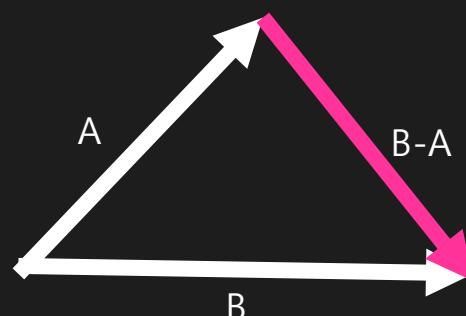
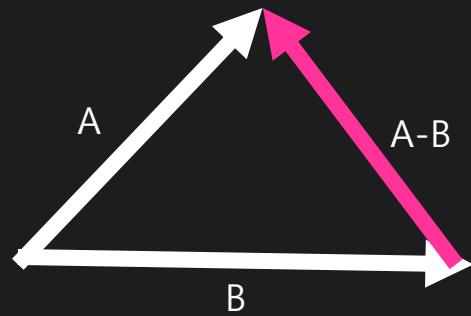
VECTOR

벡터를 방향으로 인지한다면, 다음의 기하학적 의미가 생깁니다.

1. 덧셈 : 두 벡터의 시작점과 끝점을 이은 것이 됩니다.



2. 뺄셈 : 두 벡터의 끝점을 이은 것이 됩니다.



※ 각 성분끼리 곱하거나 나눈 값은 기하학적 의미를 찾기 힘듭니다.

VECTOR MATH NODE

Distance, Length

Distance 는 두 점 사이의 거리, Length는 단일 벡터의 크기 (=화살표의 길이) 를 의미합니다.



Normalize

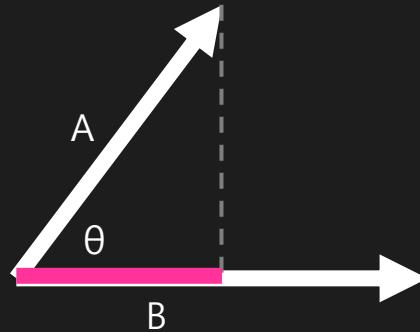
(벡터의 방향만 생각하기 위해,) 크기를 1로 만듭니다.



VECTOR MATH NODE

Dot Product

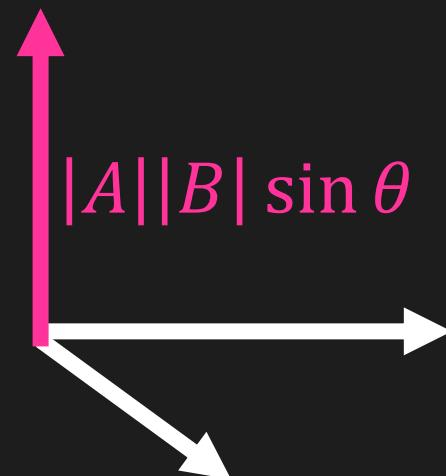
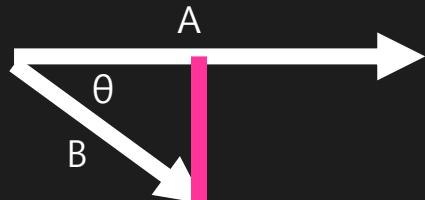
.....결과적으로, 입력받은 두 벡터가 얼마나 방향이 다른지 알게 해 줍니다.



$$|A||B| \cos \theta$$

Cross Product

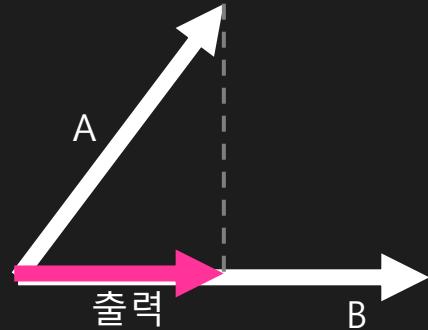
.....결과적으로, 입력받은 두 벡터와 수직인 벡터가 만들어집니다.



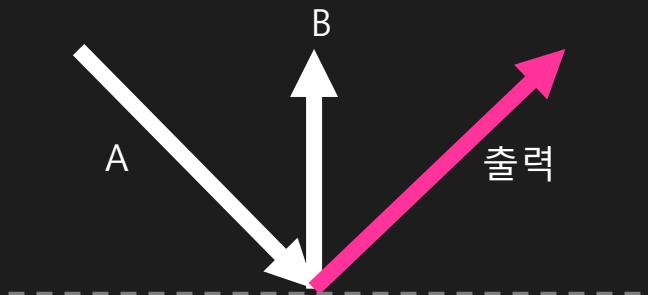
$$|A||B| \sin \theta$$

VECTOR MATH NODE

Project

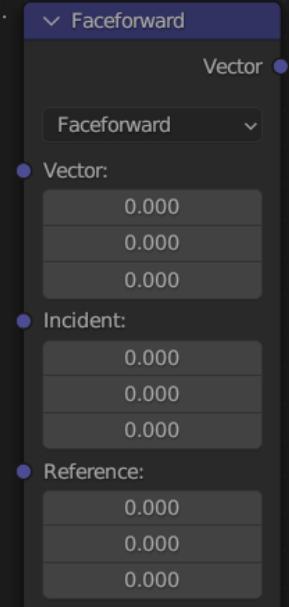


Reflect

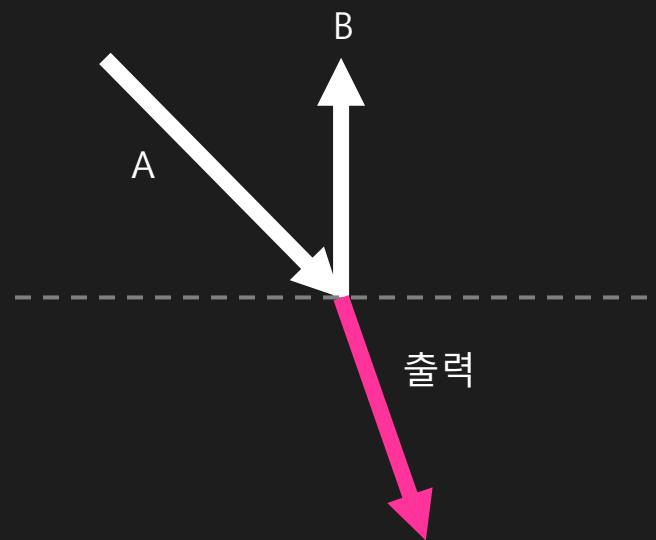


Faceforward

Incident가 Reference가 만드는 면을 바라보는 방향이면, Vector를 그대로 출력, 바라보는 방향이 아니라면, Vector에 -1을 곱해 방향을 바꿉니다.



Refract



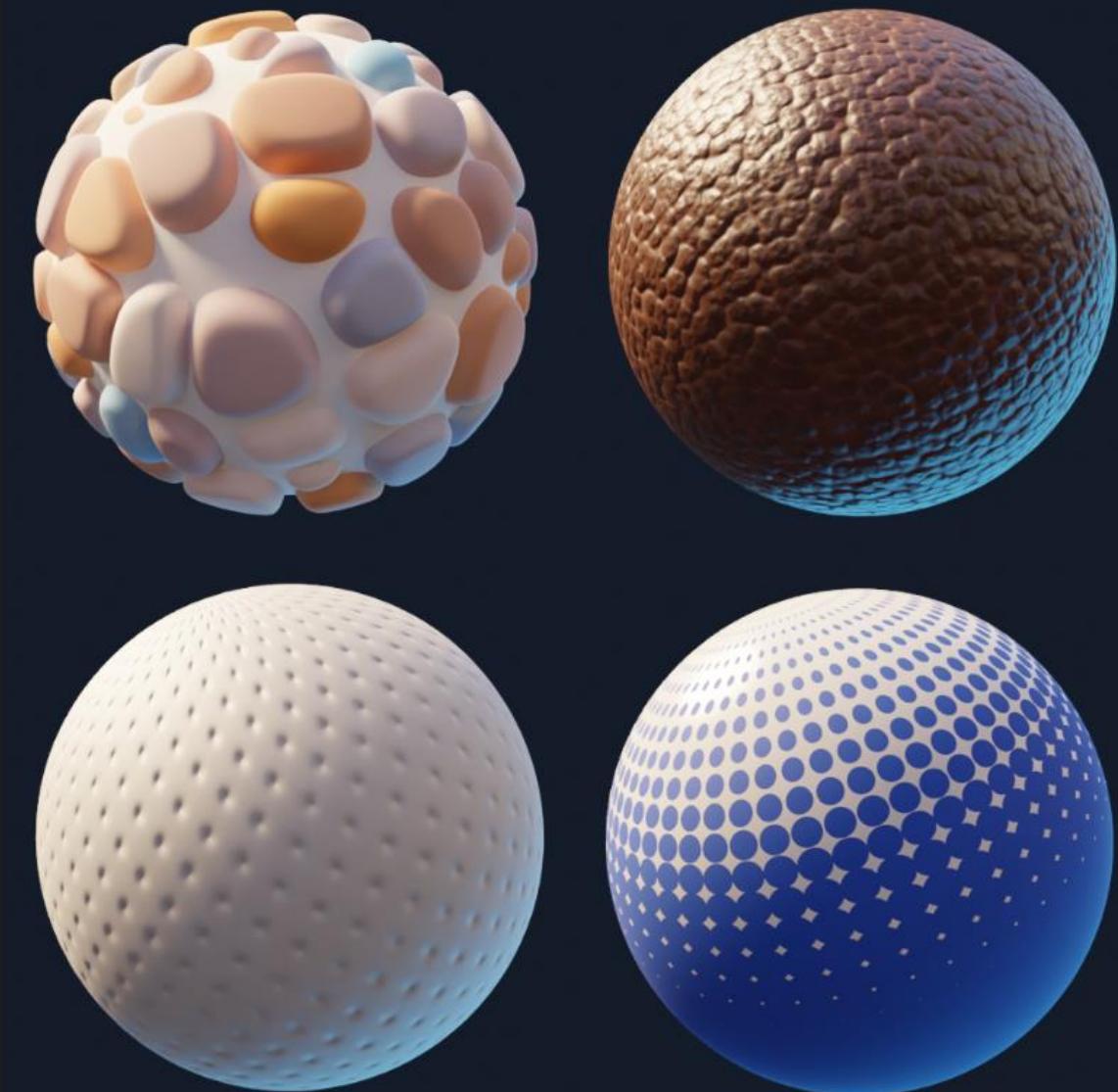
Refract의 IOR은 밀(Dense)한 매질 기준입니다.
왼쪽과 같이 되기 위해서는 IOR이 1보다
작아야 됩니다.

016강 Procedural Texture 기본

Voronoi를 이용한 무늬

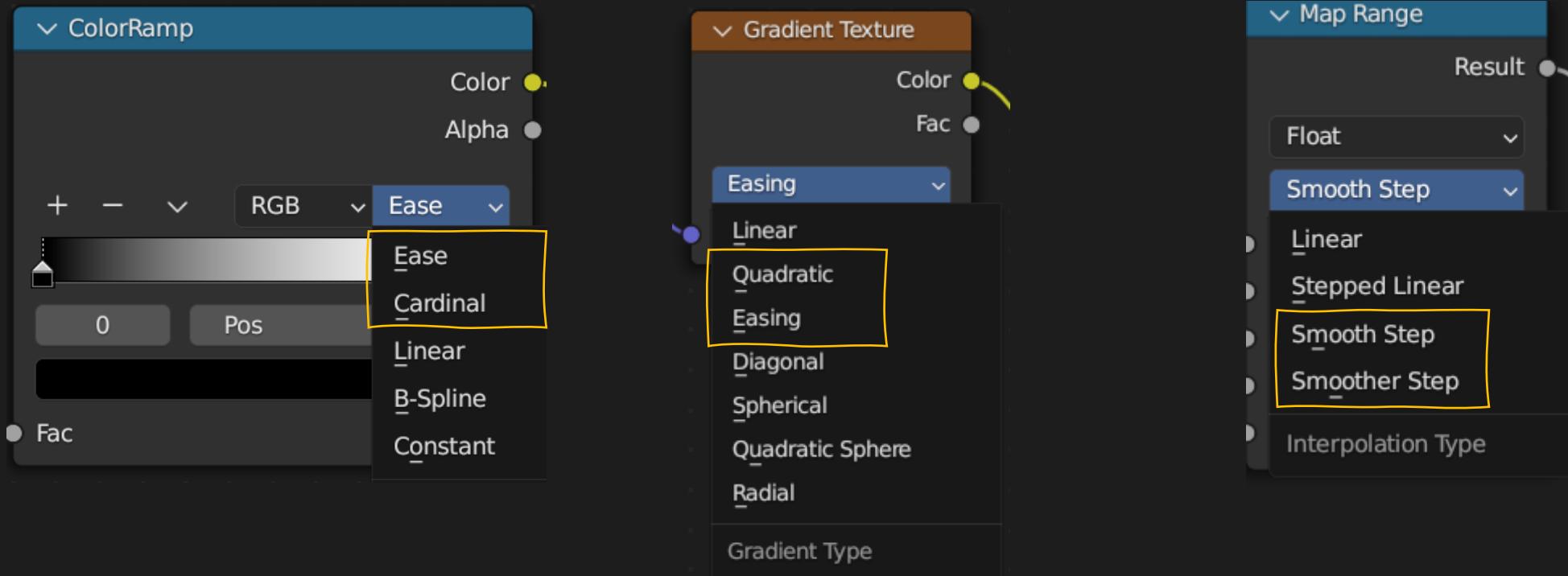
Bump Map 사용 시의 유의점

그래프를 부드럽게 만드는 법
(ColorRamp, Gradient, Map range 의 숨겨진 기능)



그래프를 부드럽게 만드는 법

ColorRamp / Gradient Texture / Map range



사용하기 편한 것으로 골라 씁니다.

Bump vs Normal Map

왜 이미지 텍스쳐엔 범프맵 대신 노멀맵이 제공될까요?

1. '높이' 가 가지는 모호함

범프맵은 밝기로 '높이' 를 나타냅니다.

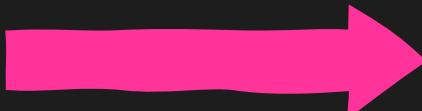
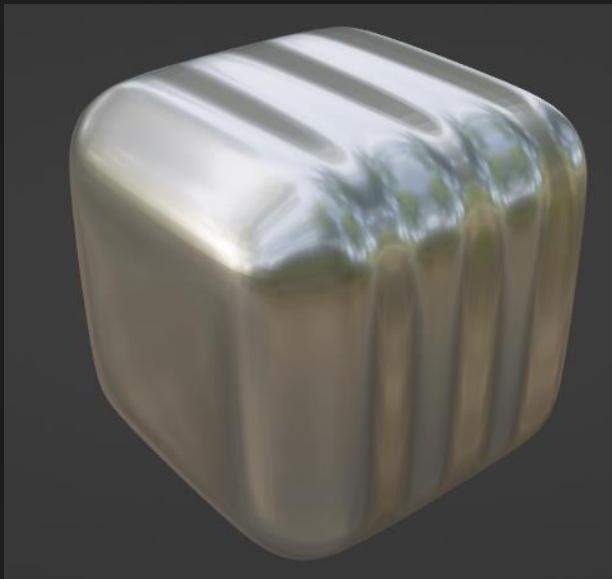
하지만 이 방식으로는 얼마나 '기울어' 있을지가 물체 크기에 따라 달라집니다!

그것은 왜일까요? 예를 들어, 범프맵 그라디언트가 1m에 걸쳐있다고 해 봅시다.

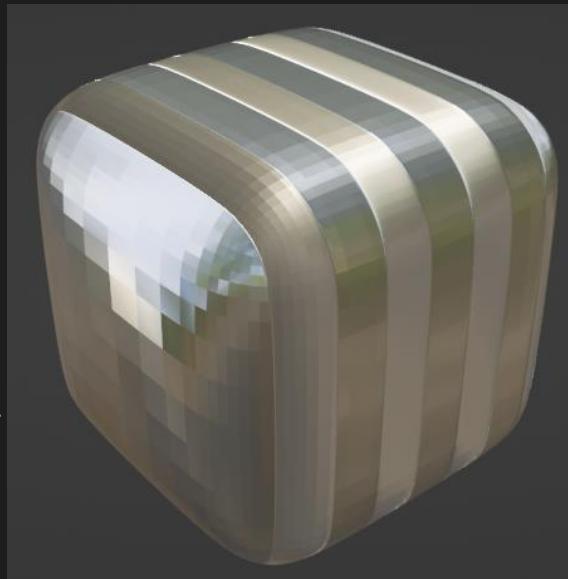
그리고 가장 어두운 부분과 가장 밝은 부분의 높이 차이를 5cm라고 설정했다고 쳐 봅시다.

지금은 잘 작동하겠지만, 만약 오브젝트가 1000분의 1만큼 작아졌다면 어떨까요?

범프맵은 여전히 높이 차이를 5cm로 표현하려 할 것입니다. 그에 따라 기울기는 엄청나게 높아집니다.



같은 셋팅으로,
크기만 1000분의 1로 줄였습니다.



Bump vs Normal Map

왜 이미지 텍스쳐엔 범프맵 대신 노멀맵이 제공될까요?

2. 이미지 포맷의 한계

일반적인 이미지는 8Bit의 정보를 담고 있습니다. RGB각 채널별로 $2^8 = 256$ 가지의 색을 만들 수 있습니다.

따라서 RGB 각 채널마다 256가지의 색 : $256 \times 256 \times 256 = 1,600\text{만가지}$ 의 색을 만들 수 있습니다.

하지만 범프맵은 흑백의 이미지이므로, 256가지의 명도 변화만 가능합니다.

256단계는 부드러운 이미지 변화를 만들기엔 부족합니다.



Bump vs Normal Map

그럼에도 범프맵이 가지는 이점은

- 직관적입니다.

노멀맵의 색을 직접 만들어내기는 힘듭니다.

Procedural한 방법으로 생성한 범프맵은 8비트의 한계를 가지고 있지 않으므로, 1번 문제는 사라집니다.
2번 문제만 적절히 컨트롤하면, 좋은 도구가 될 수 있습니다.

- 나중에 노멀맵으로 만들 수도 있습니다.

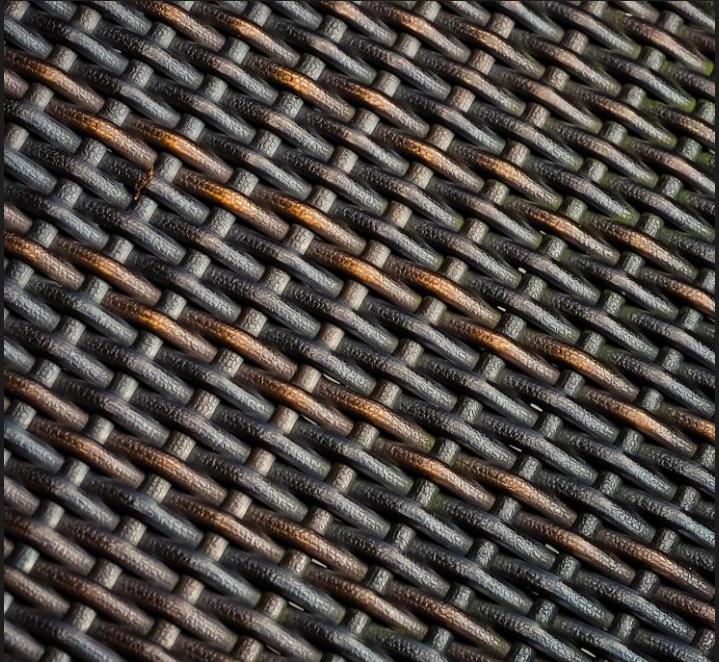
범프맵을 노멀맵으로 만들어주는 프로그램들이 있으며, 웹페이지에서도 가능합니다. (height to normal로 검색해 보세요)
아니면 블렌더 내에서 가능하게 해주는 무료 애드온도 존재합니다.

017강 Procedural Texture - 직물

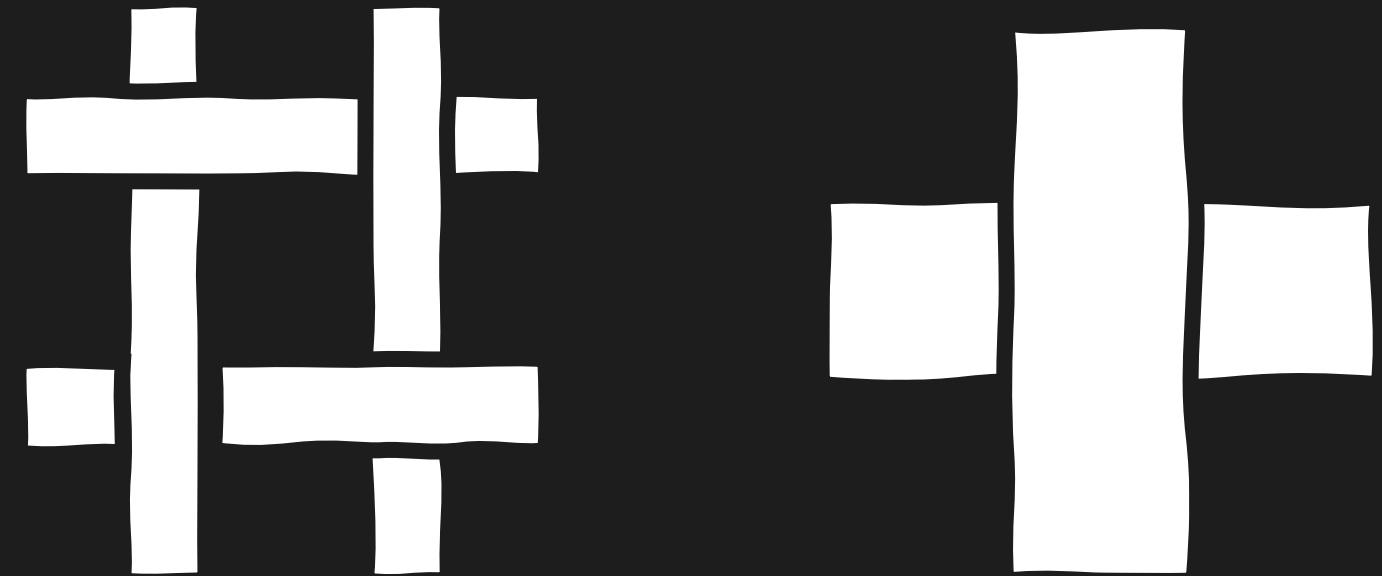


기본 아이디어

패턴을 조개봅시다



모든 디테일을 한번에 고려하기는 어렵습니다.

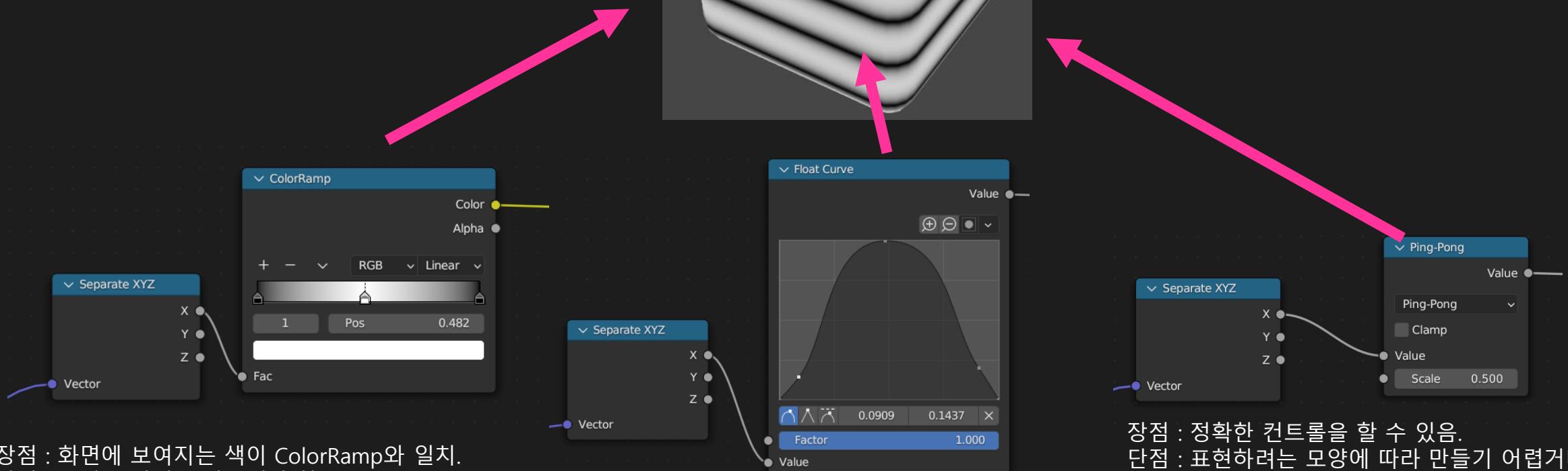
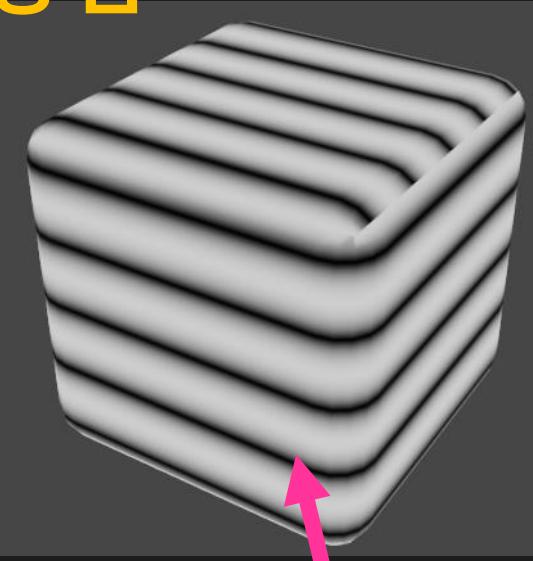


구조를 단순화해서 생각해봅시다.

일단, 두 선이 겹치는 것만 생각하겠습니다.

높낮이를 표현하는 3가지 방법

ColorRamp
vs Curve
Vs Math node

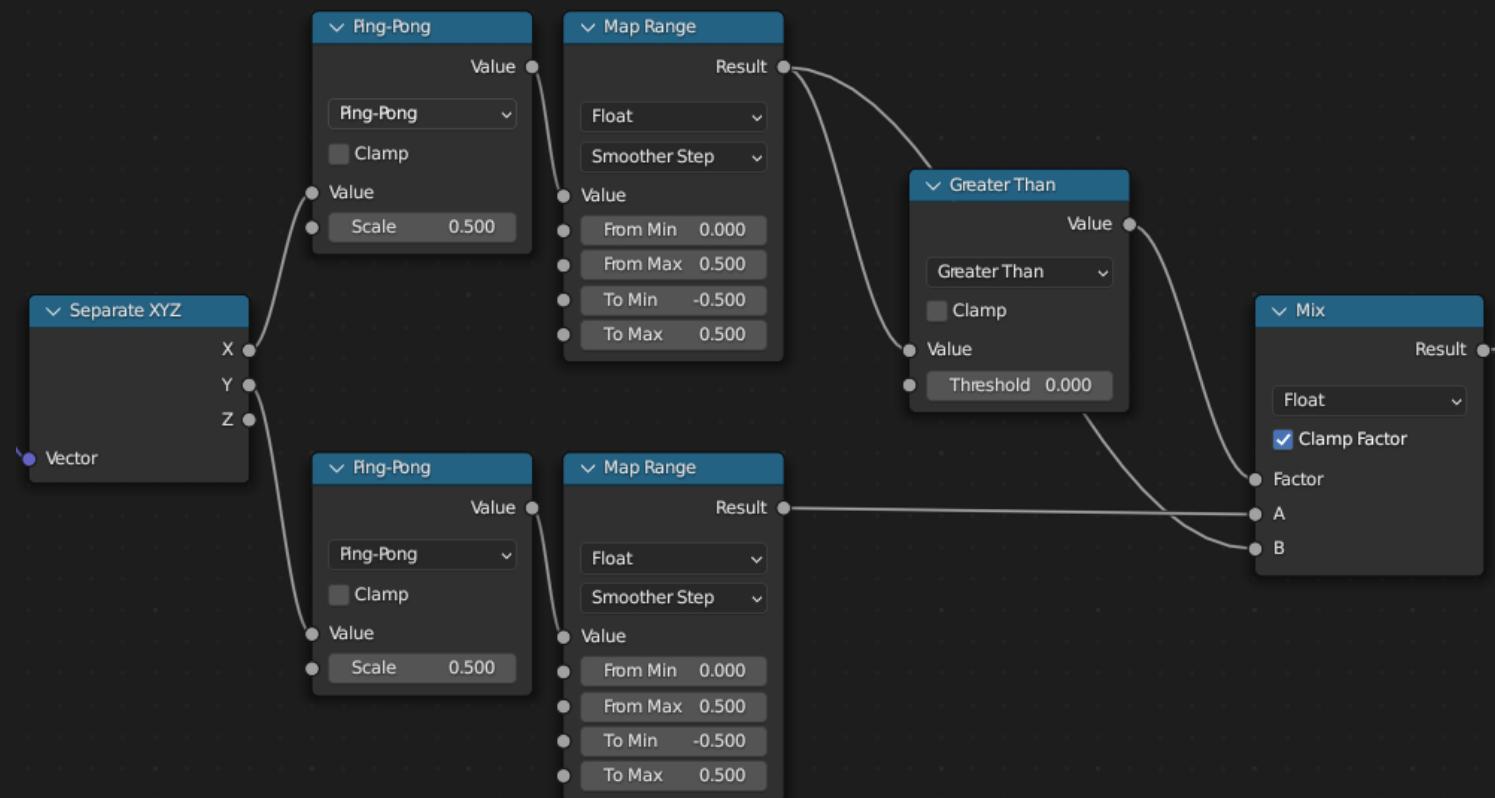
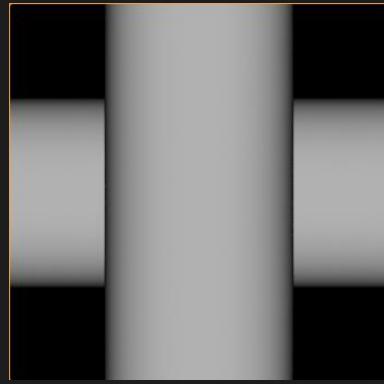
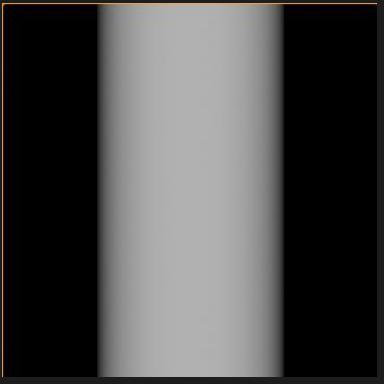
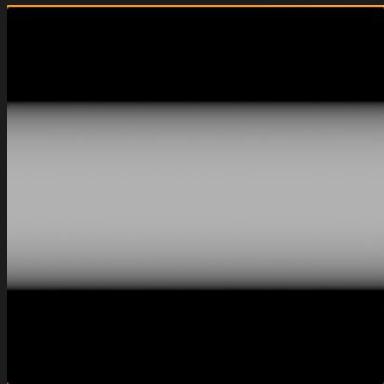
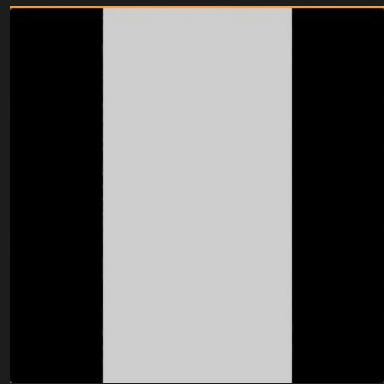


장점 : 화면에 보여지는 색이 ColorRamp와 일치.
단점 : 높이를 밝기로 가늠해야 함

장점 : 만들어지는 높이를 직접 확인 가능.
단점 : 컨트롤이 좋지 않음.
크기가 큼 (치명적)

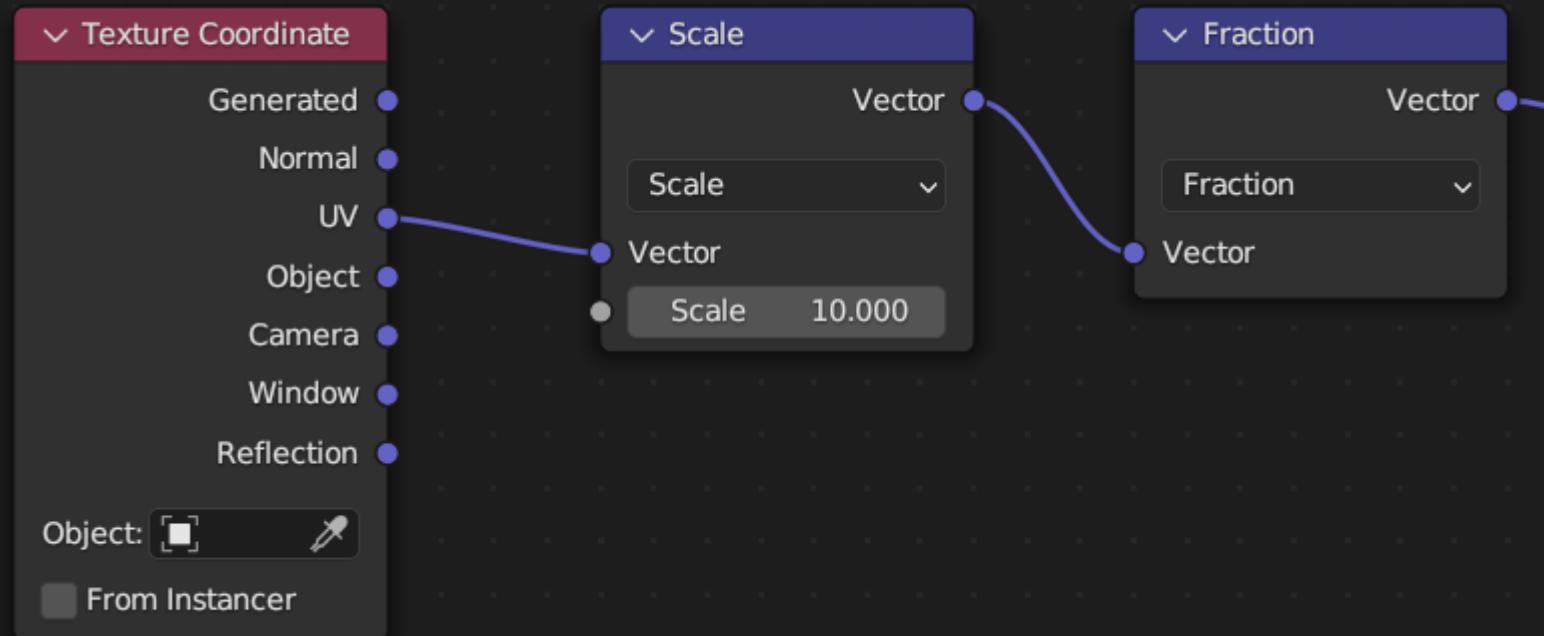
장점 : 정확한 컨트롤을 할 수 있음.
단점 : 표현하려는 모양에 따라 만들기 어렵거나
불가능할 수 있음

Mix

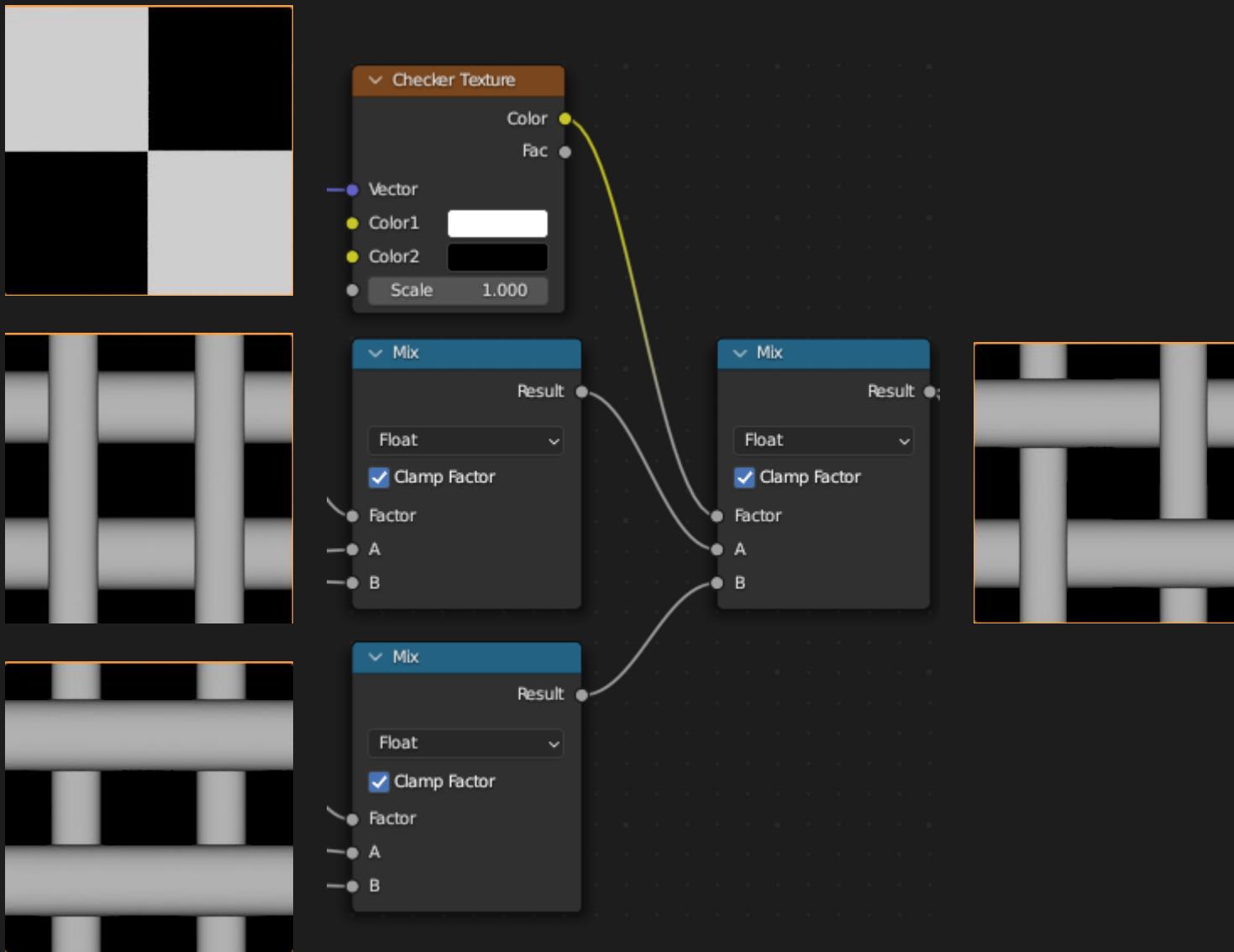


격자무늬 좌표

0에서 1 사이가 번갈아가며 나타나면 격자가 됩니다.

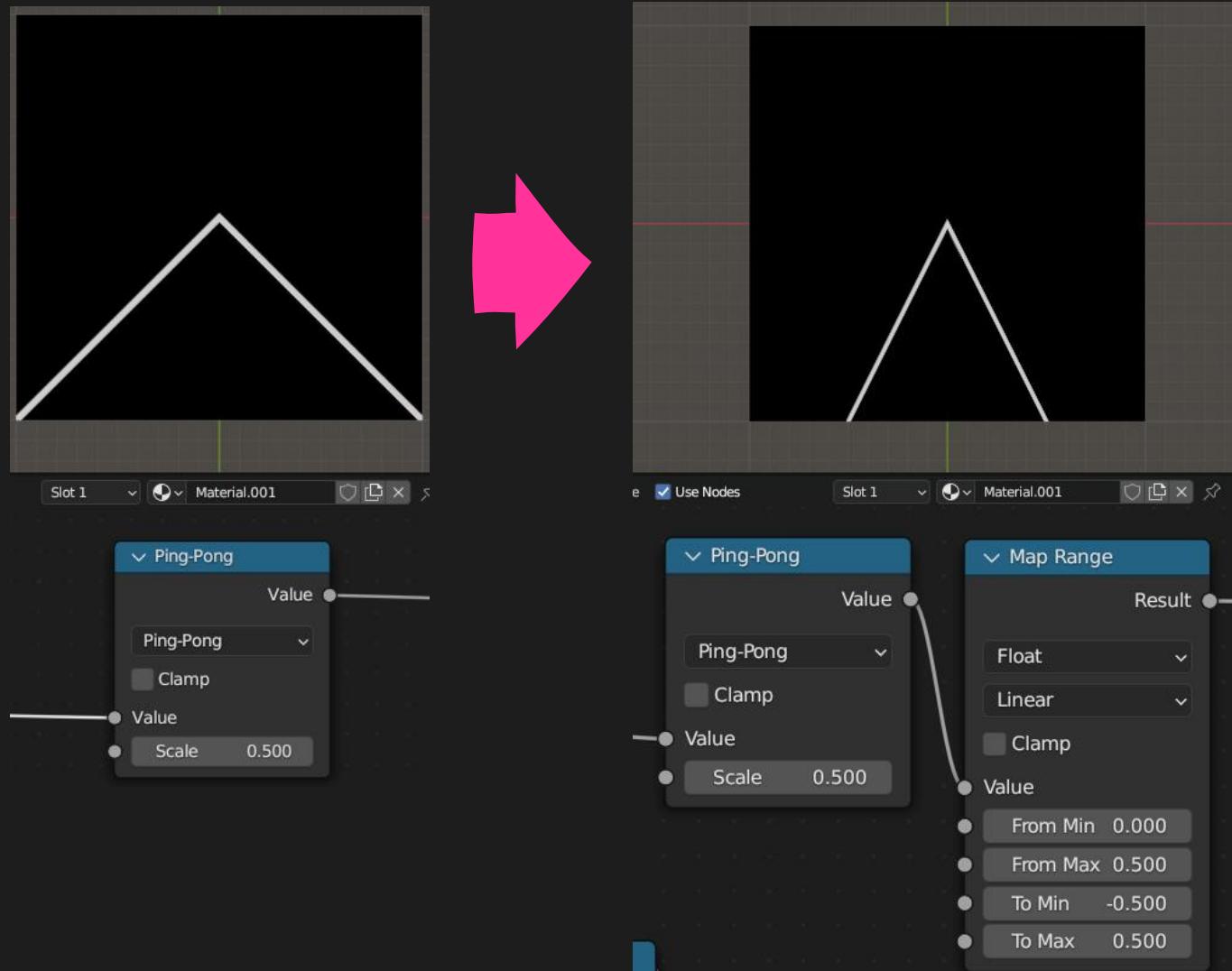


체커 무늬를 활용하기



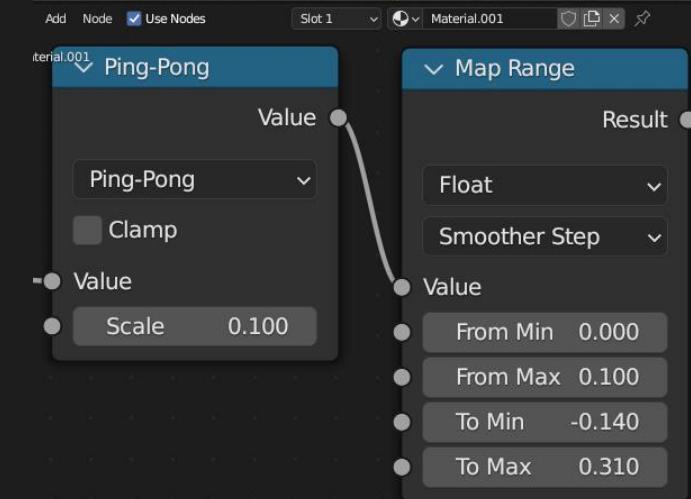
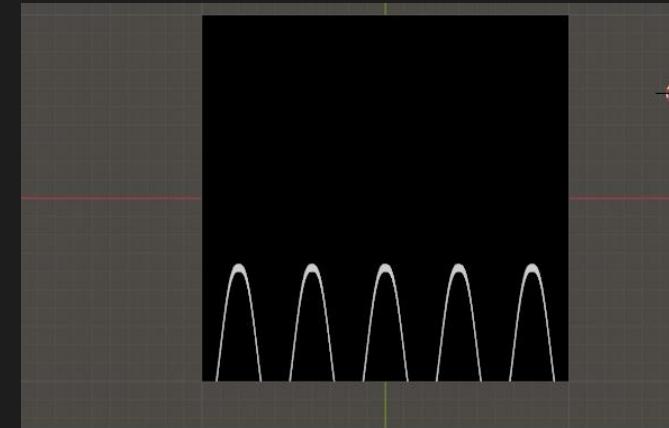
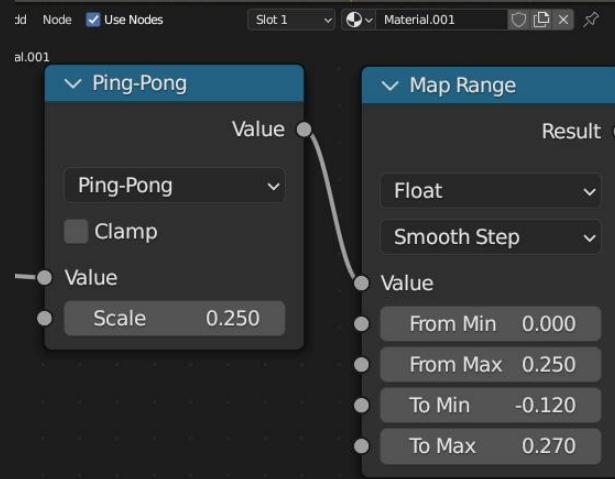
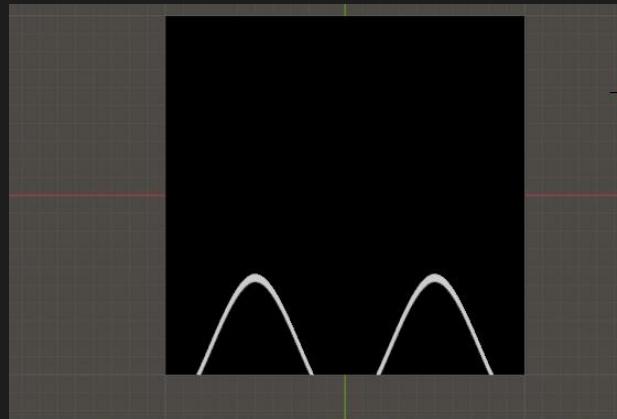
무늬를 그래프로 만들어 봅시다

Feat. Ping Pong node



무늬를 그래프로 만들어 봅시다

여러 바리에이션이 가능합니다.



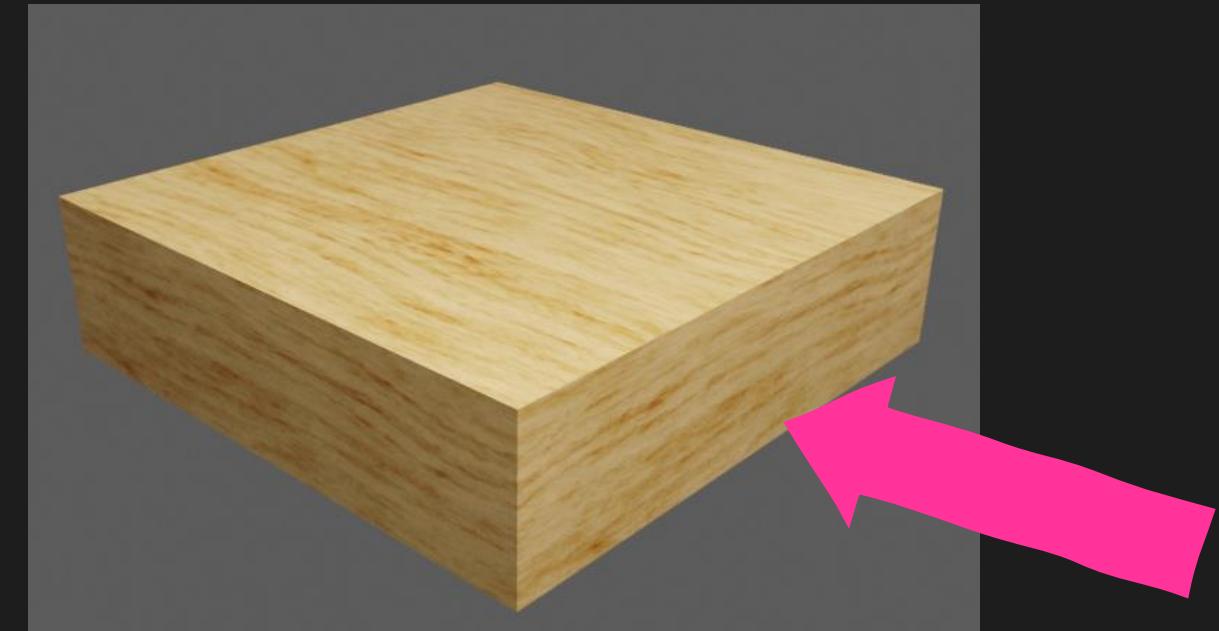
018강 Procedural Texture - 목재



2D 텍스쳐의 한계

아무리 잘 만든 이미지라도 2차원의 데이터이므로, 3차원 좌표의 변화에 따라 달라질 수는 없습니다.

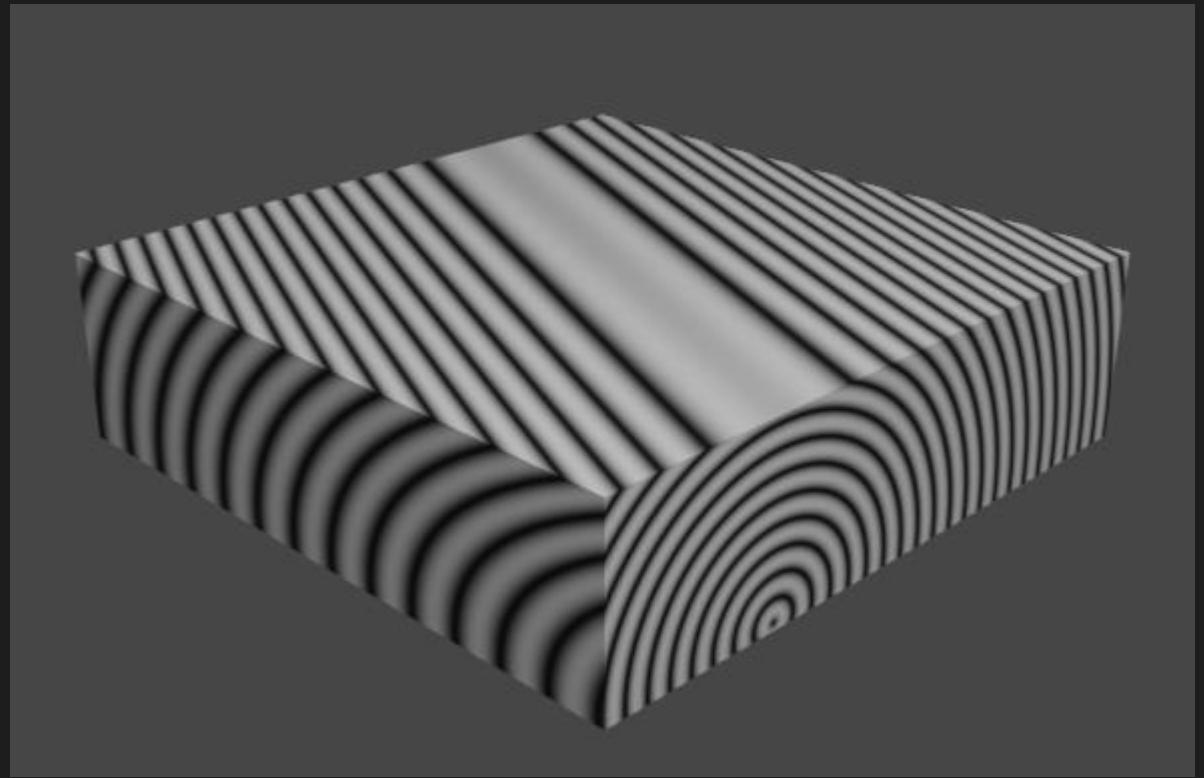
목재를 자른 단면은 이미지텍스쳐로는 표현할 수 없습니다.
~~(또다른 텍스쳐를 이용하면 가능하지만...)~~



3D 텍스쳐

3차원 좌표를 모두 사용한다면 이 문제를 해결할 수 있습니다.
즉 목재의 표면과 단면을 한번에 표현할 수 있습니다.

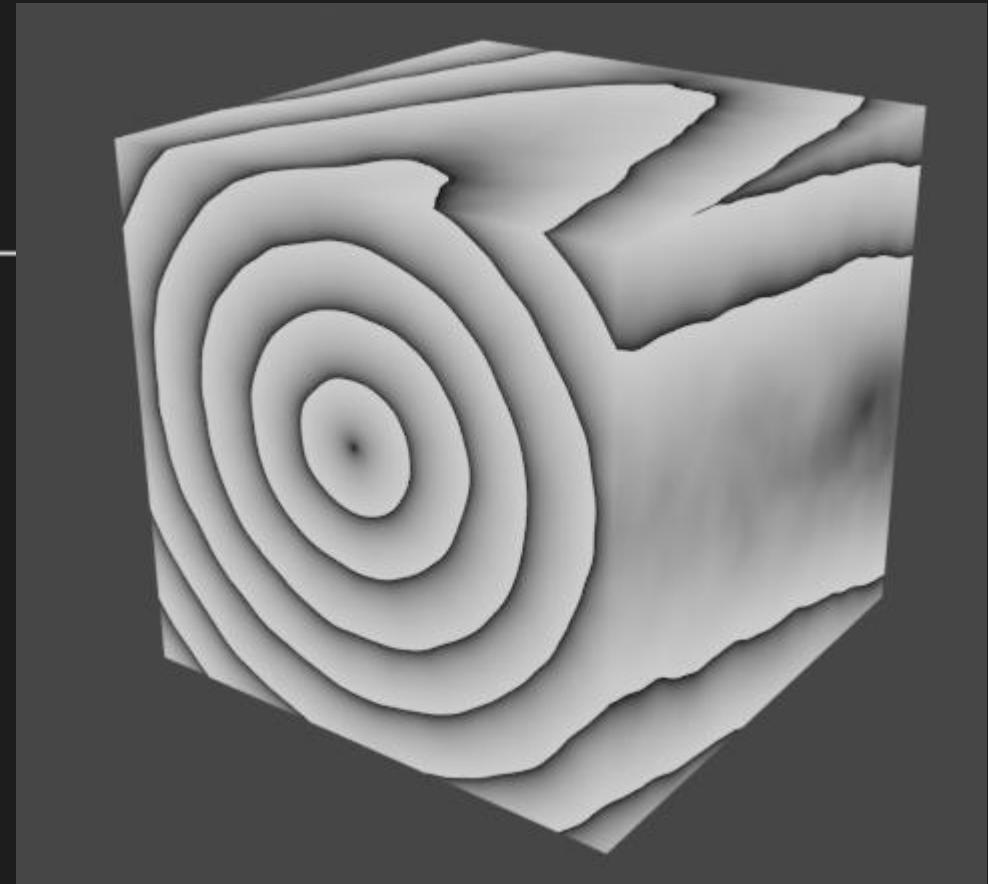
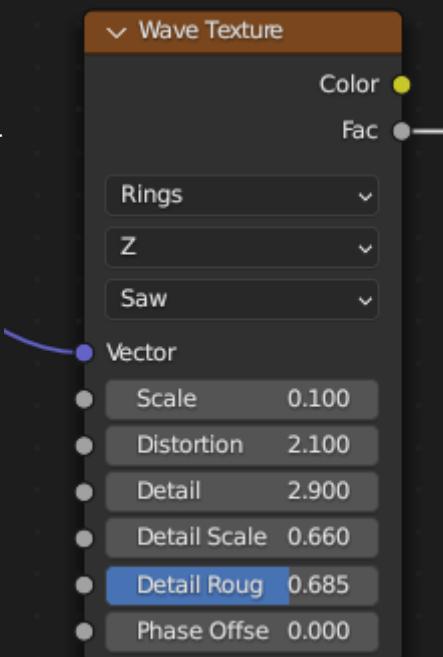
물론 한계는 존재합니다. 3차원 좌표가 필요하기 때문에 UV를
이용할 수 없고, 따라서 이미지를 고정하기 힘들 수 있습니다.



목재는 나무를 자른 것.

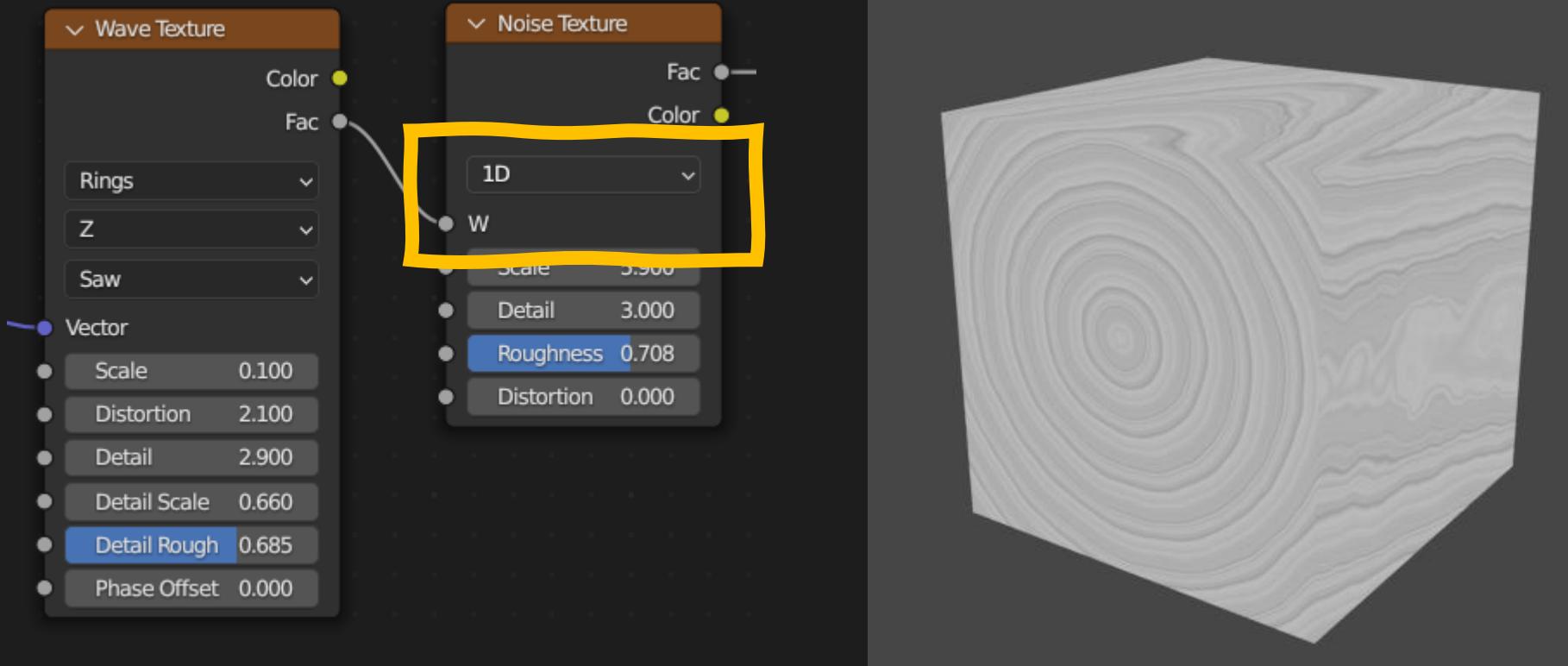
Wave Texture

Wave Texture가 많은 일을 해 줍니다.
한쪽 축 방향으로 동심원을 그리고, 노이즈를 통한
왜곡까지 한번에 처리합니다.



1D 텍스쳐

웨이브 텍스쳐로 생성한 그라데이션을 더 복잡한 형태로 만들어 줍니다.
흑백 그라데이션은 1차원 정보이므로, 1D 노이즈를 사용할 수 있습니다.

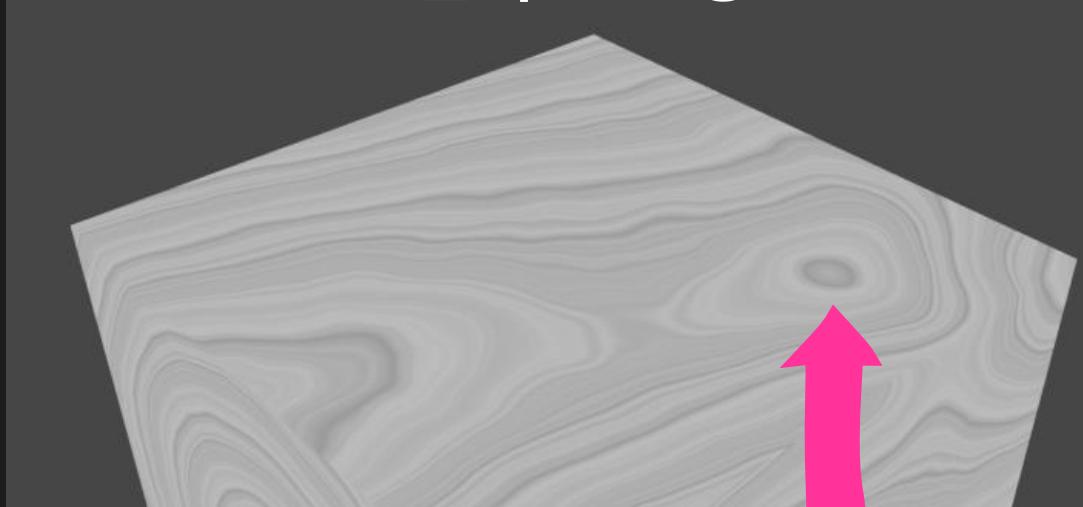


옹이

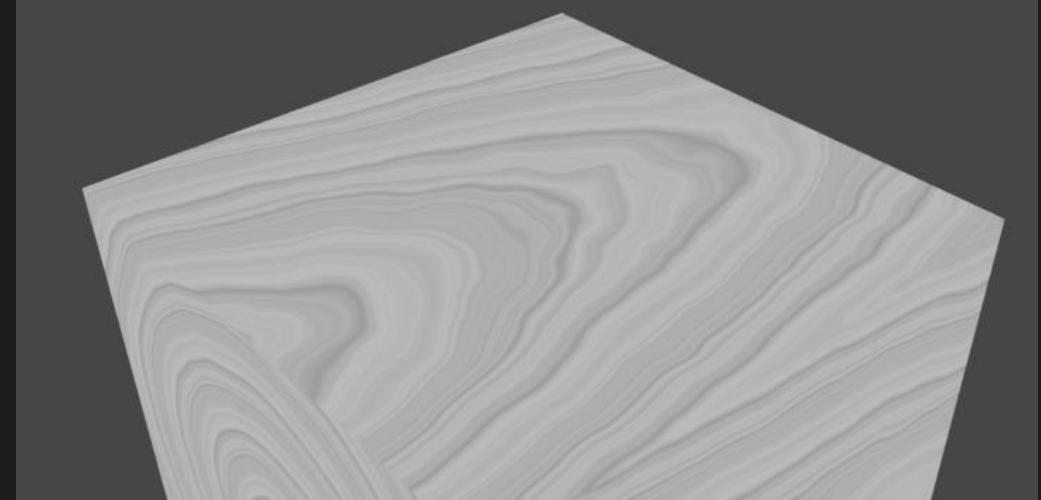
나무는 원기둥이 아니므로

가운데 중심을 제외하고도 뻗어나가는 가지가 만들어내는 옹이가 존재합니다.

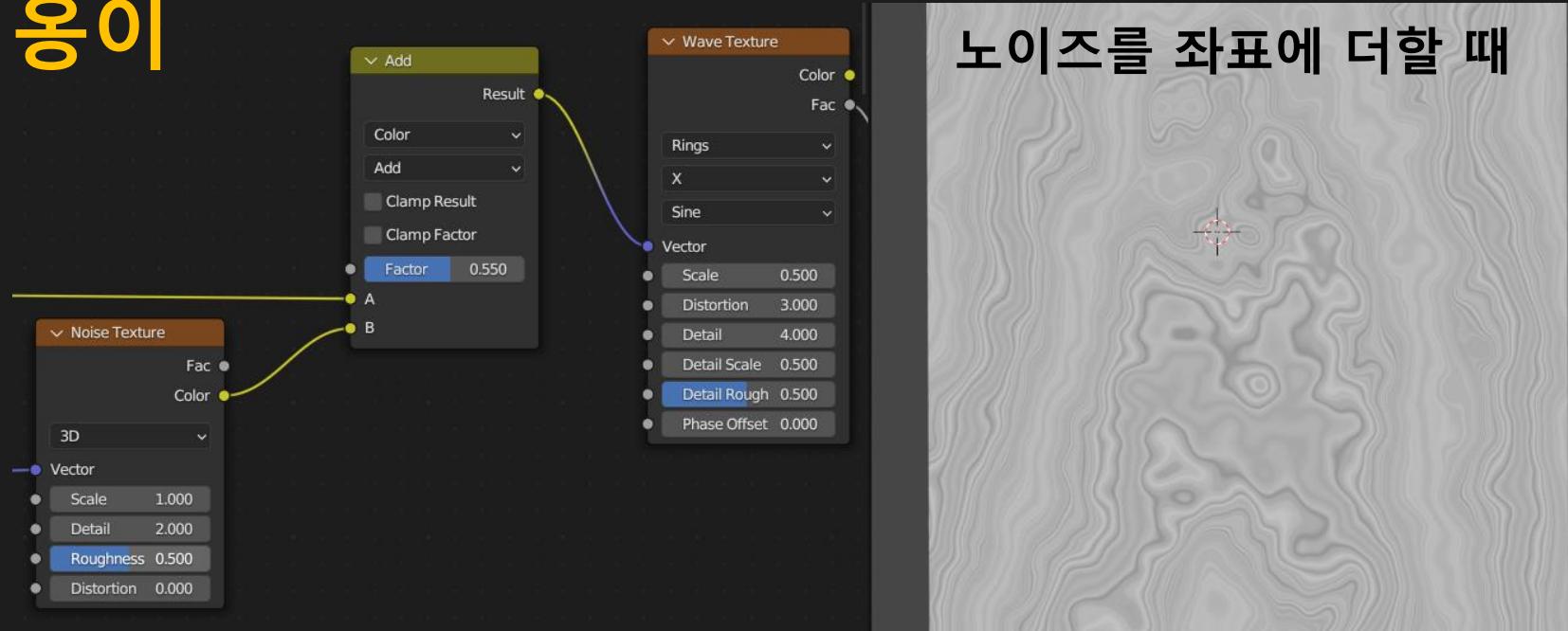
원하는 모양



현재상태

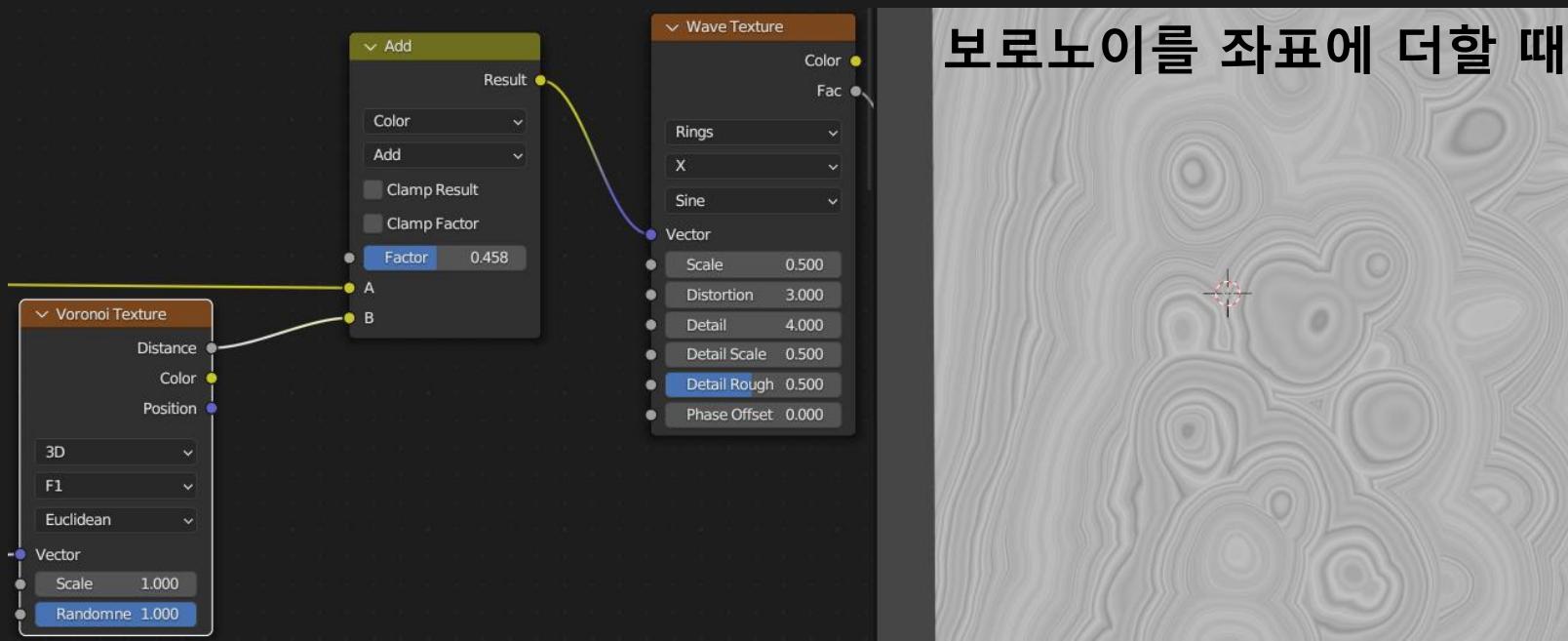


옹이



노이즈를 좌표에 더할 때

좌표 왜곡으로 보로노이를 사용하면 원형의 왜곡이 일어납니다.

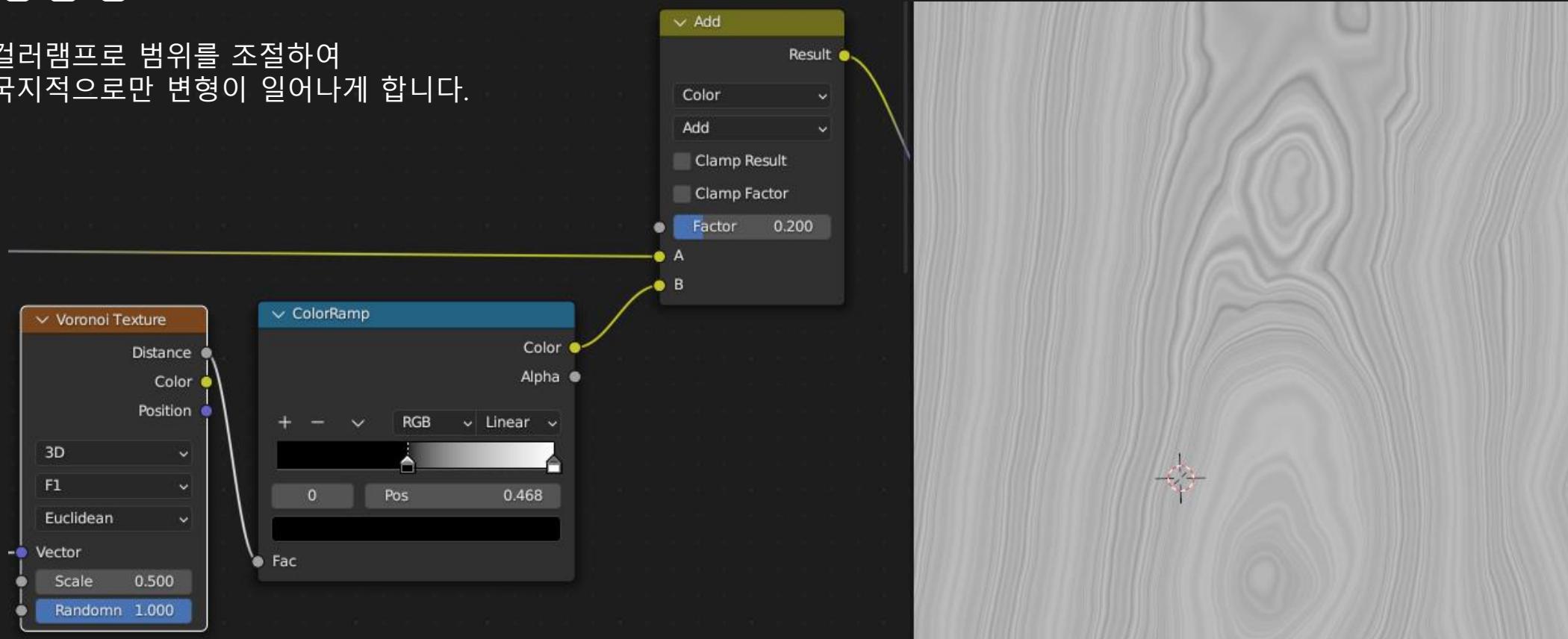


보로노이를 좌표에 더할 때

옹이

경향성

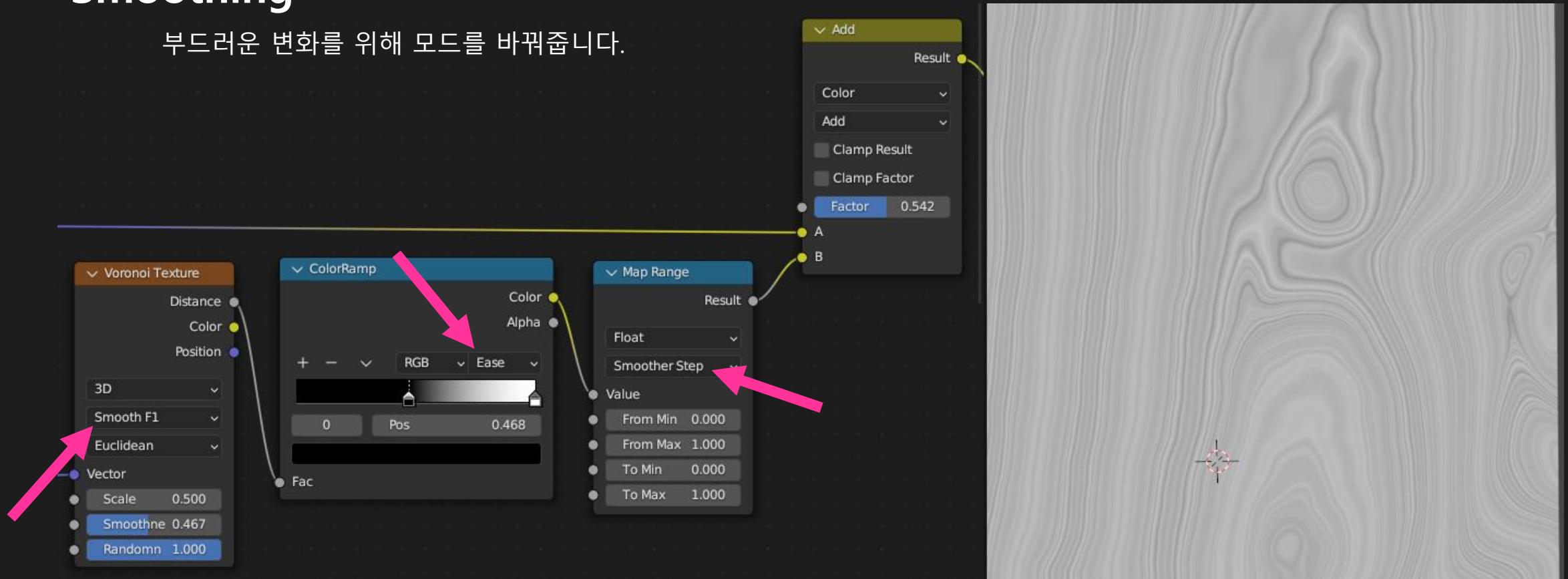
컬러램프로 범위를 조절하여
국지적으로만 변형이 일어나게 합니다.



옹이

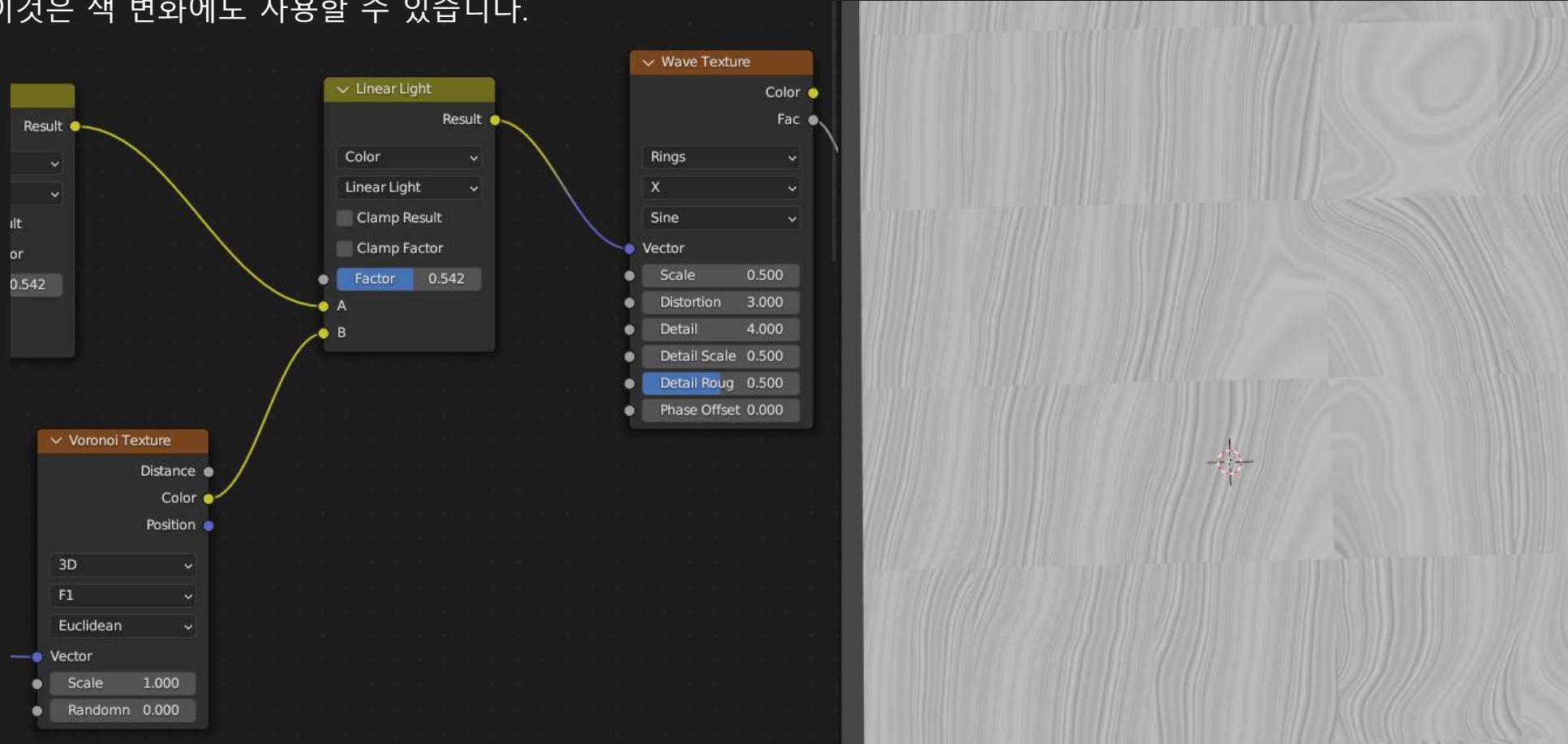
Smoothing

부드러운 변화를 위해 모드를 바꿔줍니다.



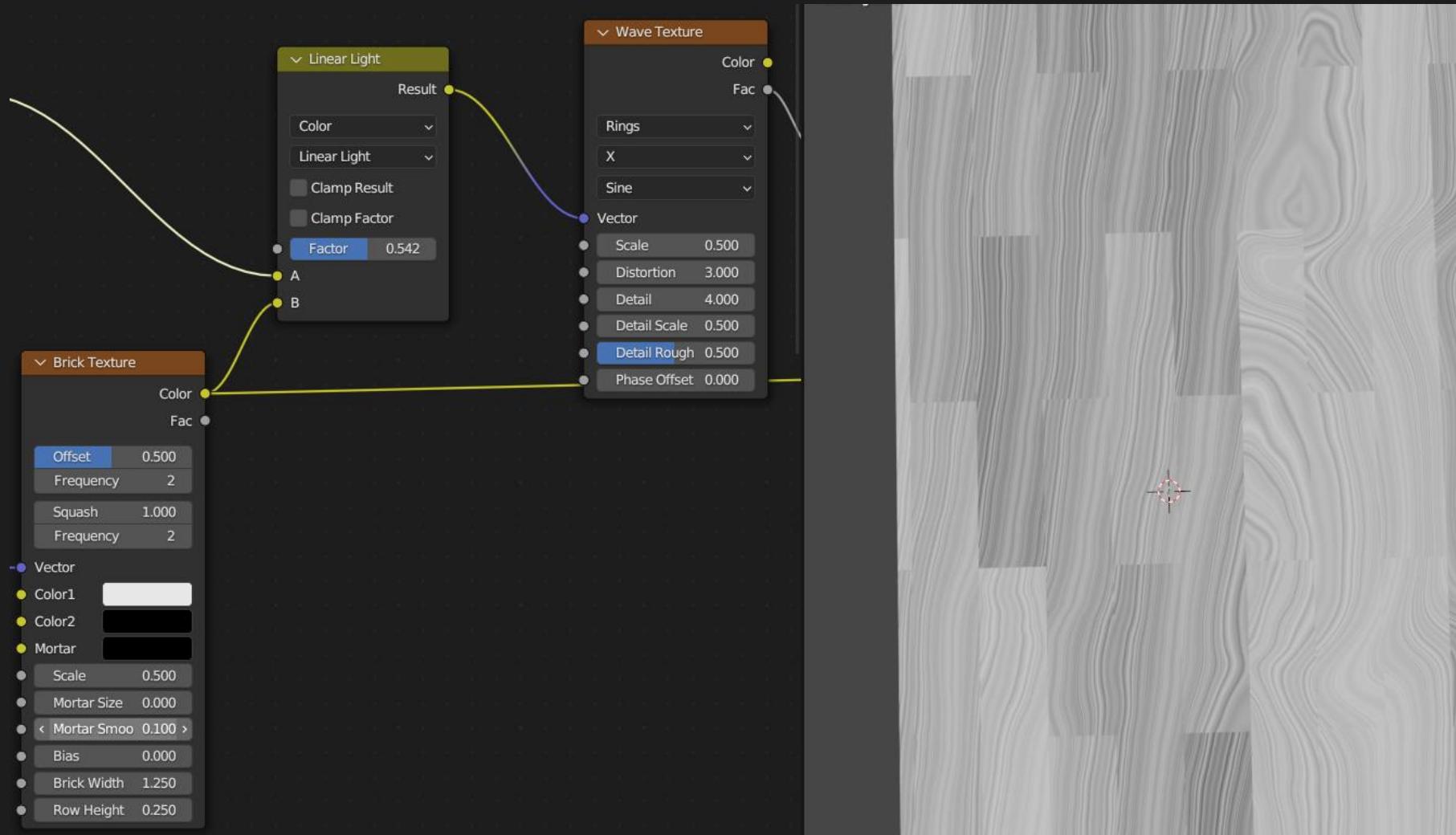
집성목

일반적인 목재는 나무를 잘라서 이어붙여 만들어집니다.
나무 무늬를 격자마다 어긋나게 해 봅시다.
이것은 색 변화에도 사용할 수 있습니다.



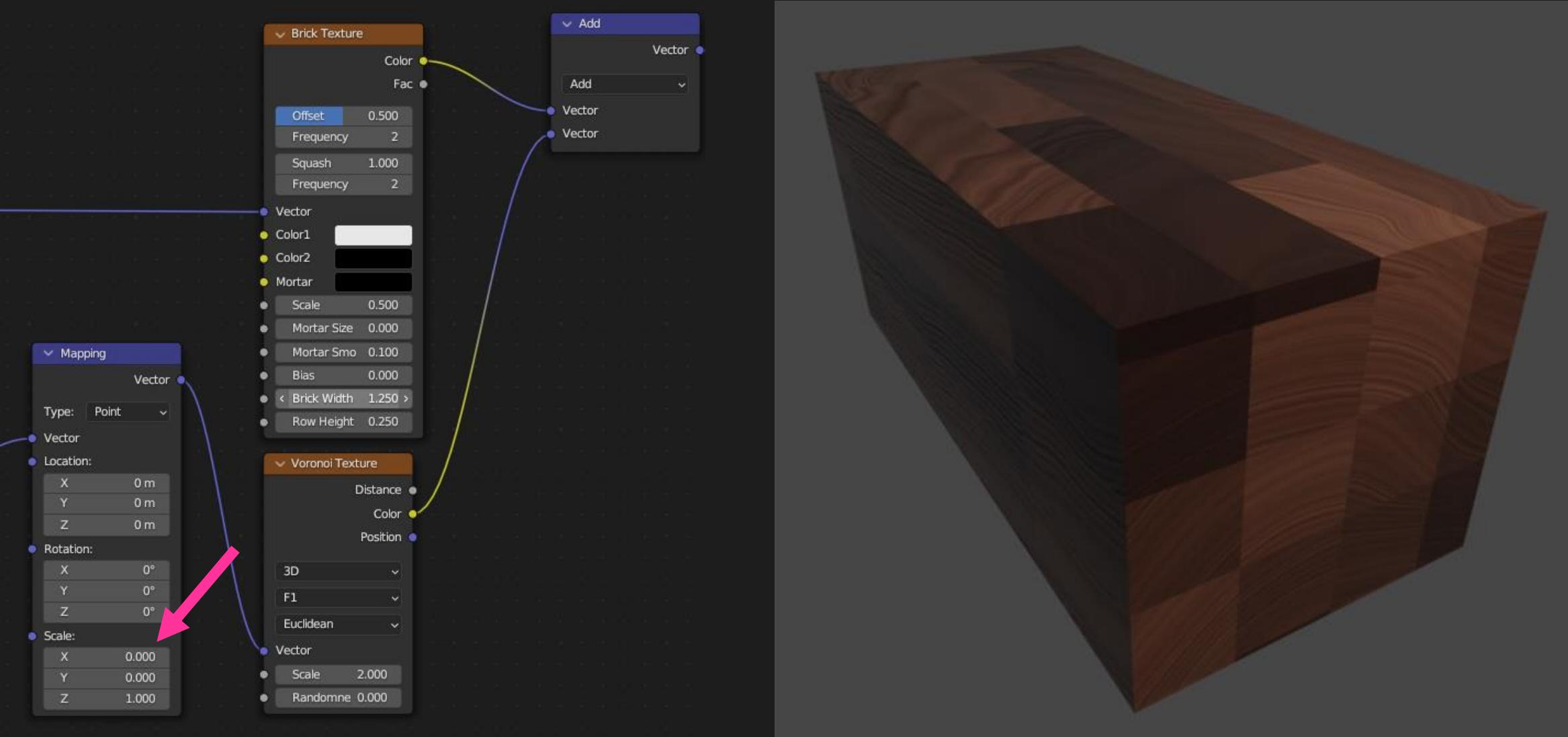
집성목

어긋나게 이어붙이려면 보로노이 말고 브릭텍스쳐를 이용합니다.
다만 브릭 텍스쳐는 2차원이므로, 가로, 세로, 높이를 동시에 자를 수는 없습니다.



집성목

브릭 텍스쳐와 보로노이를 모두 이용하면 모든 방향으로 나무를 이어붙일 수 있습니다.



스크래치

목재의 러프니스는 나무 무늬보다는 표면의 특성에 결정될 때가 많습니다.

바니쉬 등으로 마감을 한 후 오랫동안 사용을 하면서 벗겨진 칠 등이 목재 자체의 재질보다 더 강조될 수 있습니다.

이 때 clearcoat roughness에 스크래치 이미지를 사용한다면 그럴듯한 느낌을 만들 수 있습니다.

더 이상 procedural이 아니긴 하지만, 상당히 효과적입니다.



019강 Procedural Texture - 타일

블럭과 타일의 두께감 표현하기

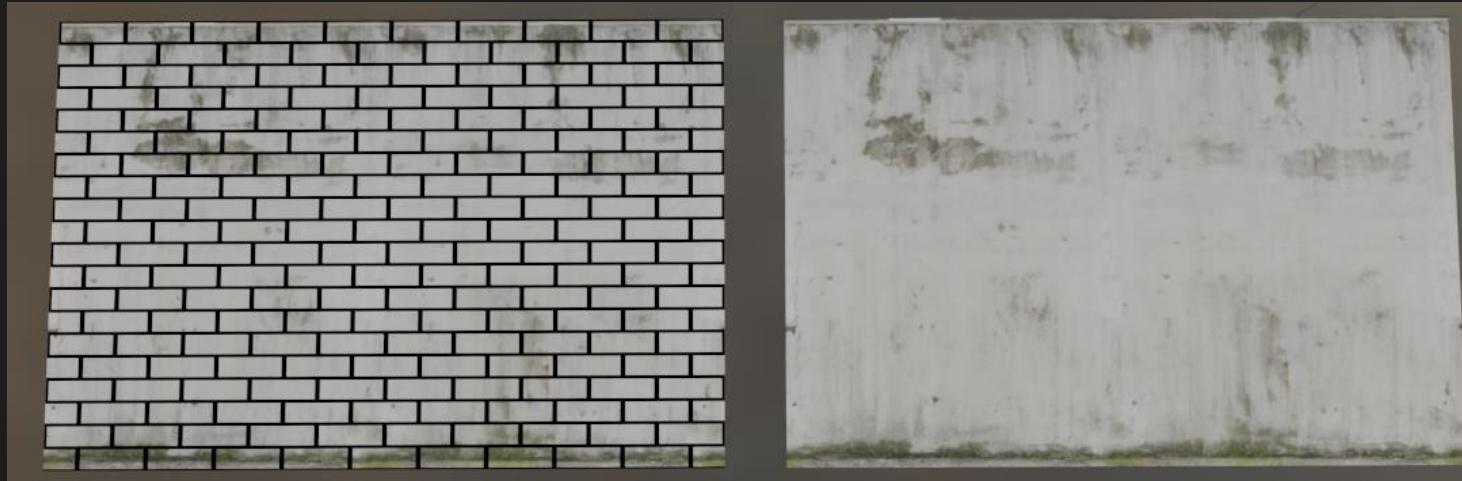
엇갈려 쌓인 모양의 좌표 만들기

가로세로축의 높이정보를 섞는 법



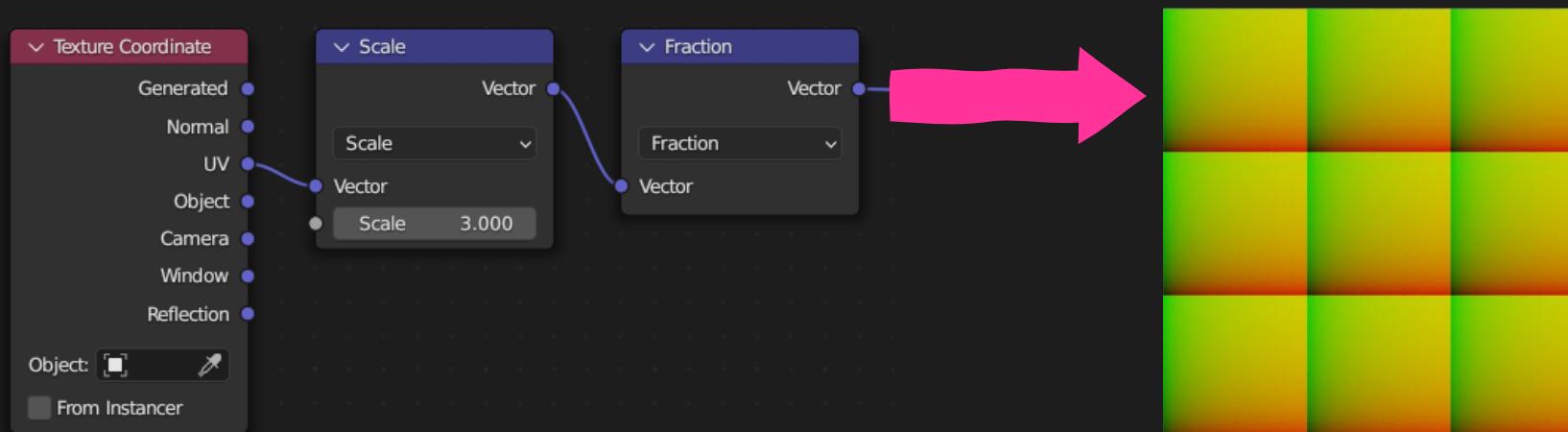
브릭 텍스쳐의 문제점

브릭 텍스쳐는 다양한 패턴을 만들기는 좋지만, 텍스쳐에 블럭 무늬를 얹을 뿐입니다.



무엇이 필요한가

우리는 무늬를 얹는 것을 넘어서서, 각 무늬 안쪽의 좌표를 컨트롤하고 싶습니다.

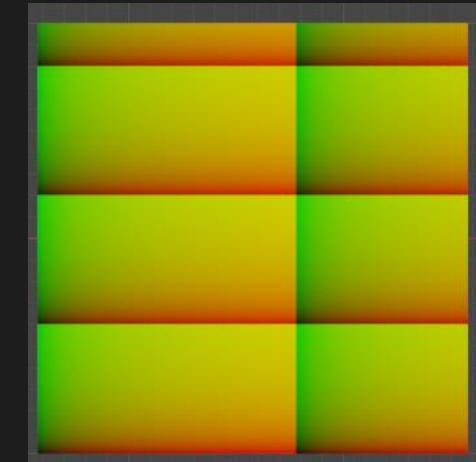
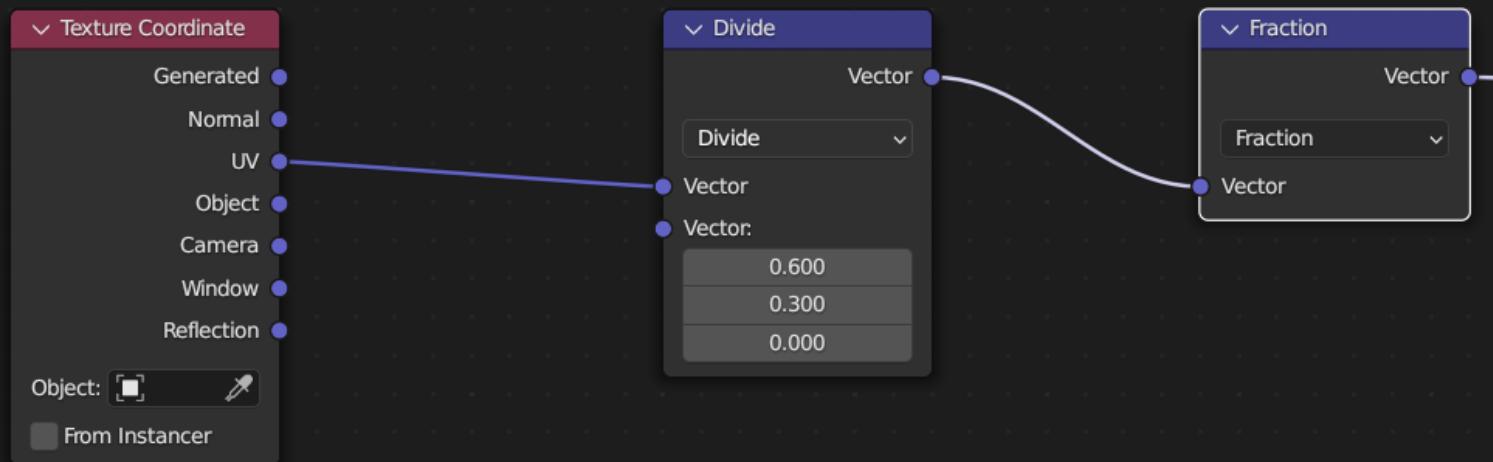


격자의 가로 세로

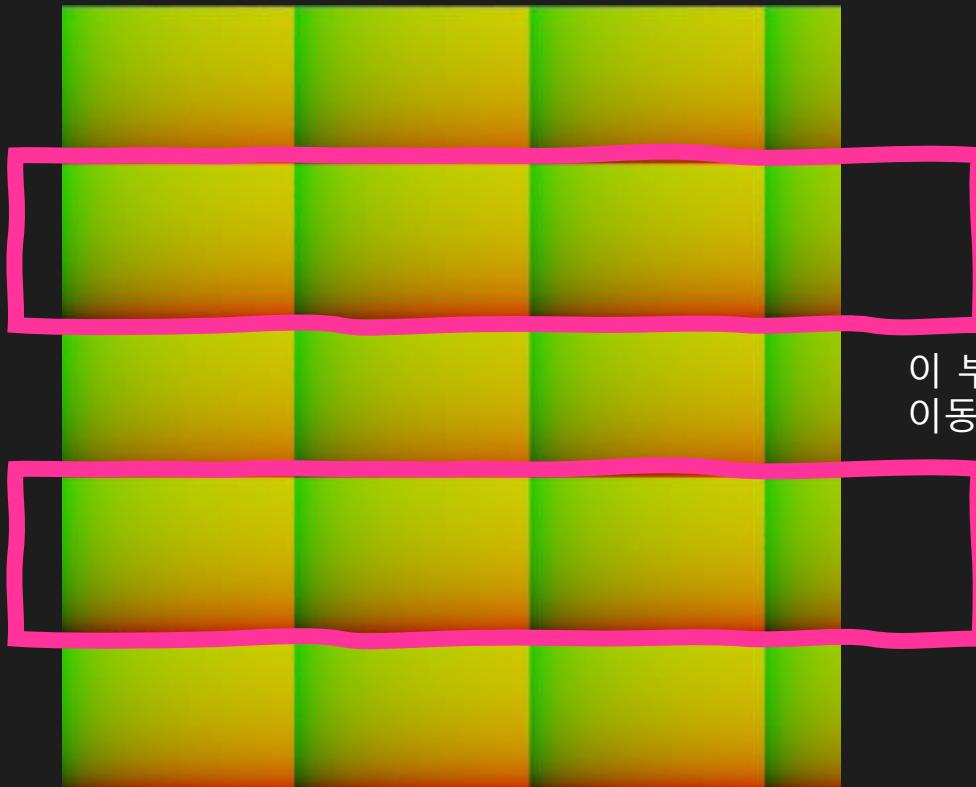
스케일을 2배로 올리면, 격자는 절반으로 작아집니다.
스케일을 3배로 올리면, 격자는 1/3로 줄어들죠.

Multiply 가 아니고 Divide를 한다면 어떨까요?
스케일을 2로 '나누면', 이는 0.5를 곱한 것과 같으므로, 격자는 2배로 커집니다.

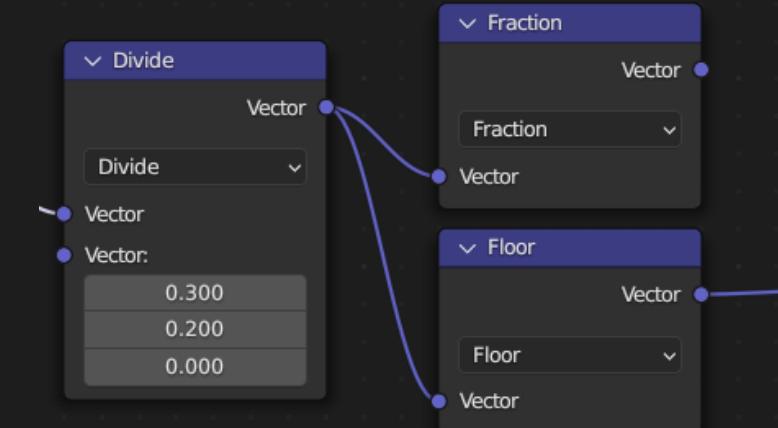
이렇게 곱셈이 아니라 나눗셈을 이용하면 입력값이 길이와 연관됩니다.



엇갈려 쌓기- 정석

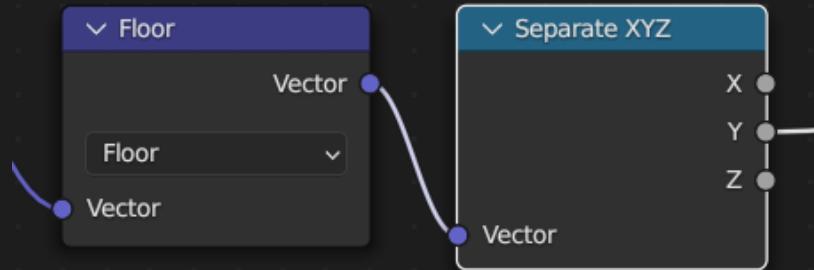


이 부분만 왼쪽으로
이동시키고 싶습니다.



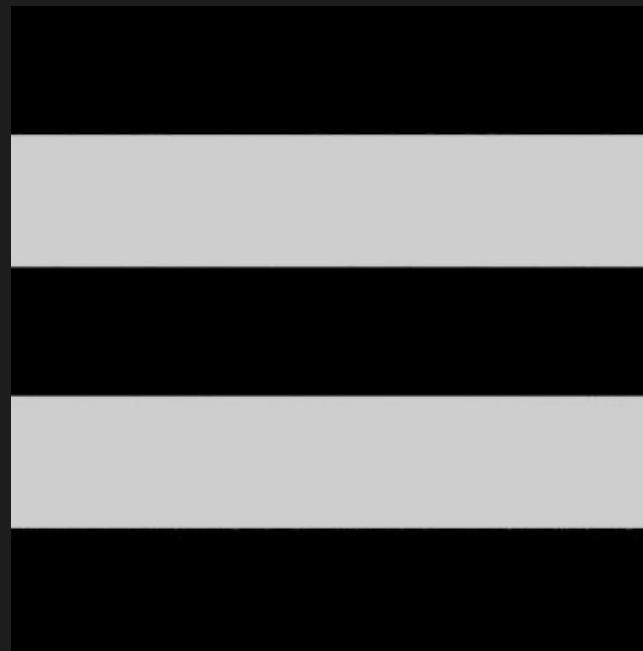
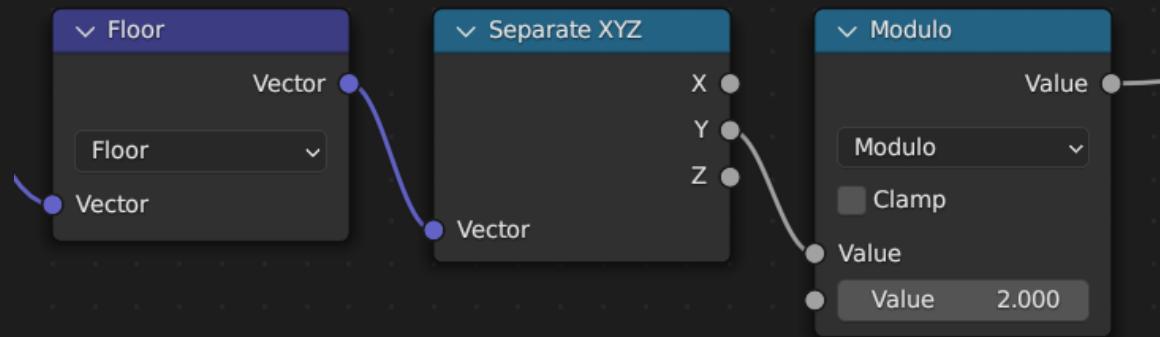
Floor 노드를 활용합니다.

엇갈려 쌓기 - 정석



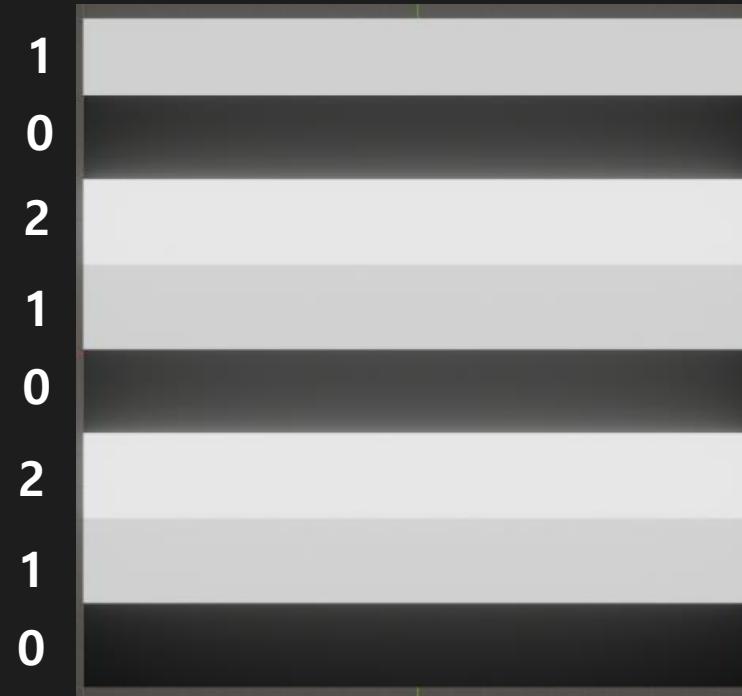
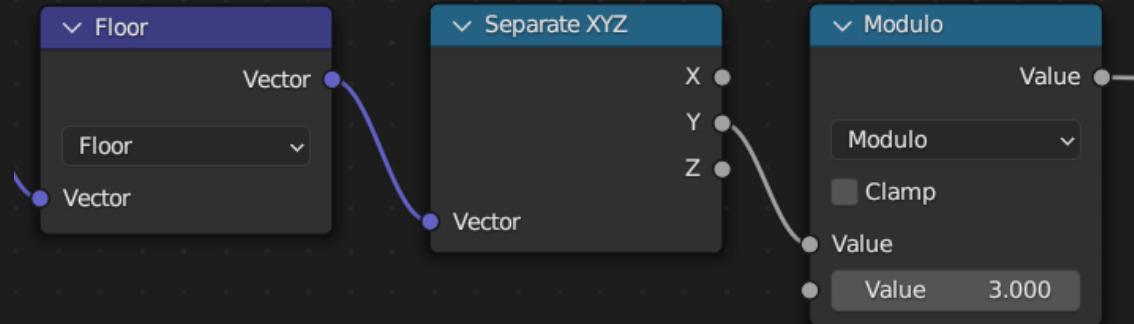
1,3,5....번째 행만 표시할 수는 없을까요?

엇갈려 쌓기 - 정석

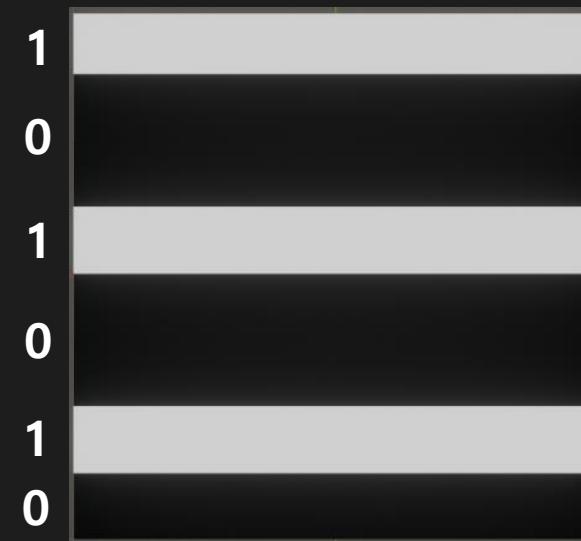
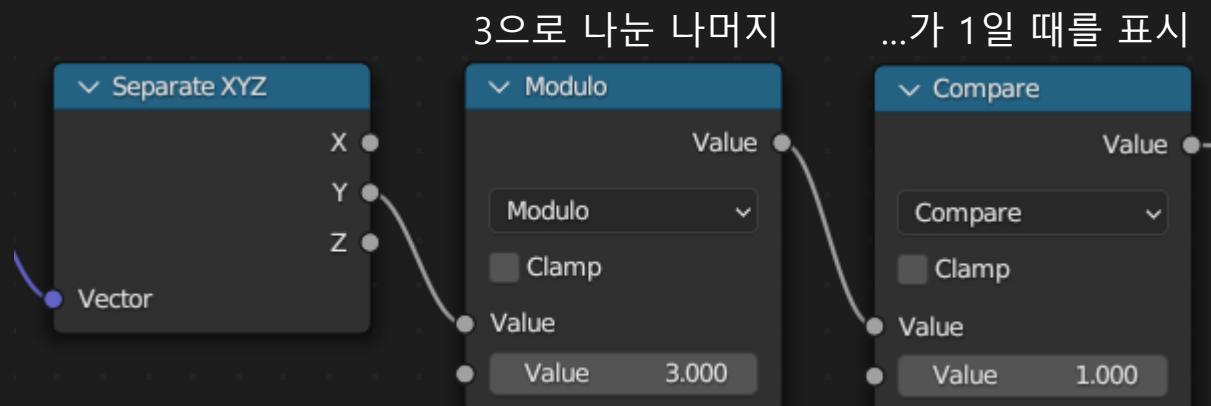


2로 나눈 나머지를 표시하면 됩니다.

엇갈려 쌓기 - 나누기 3의 경우

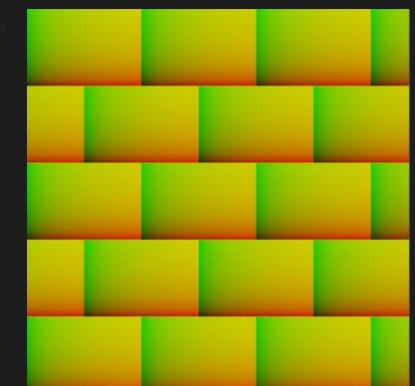
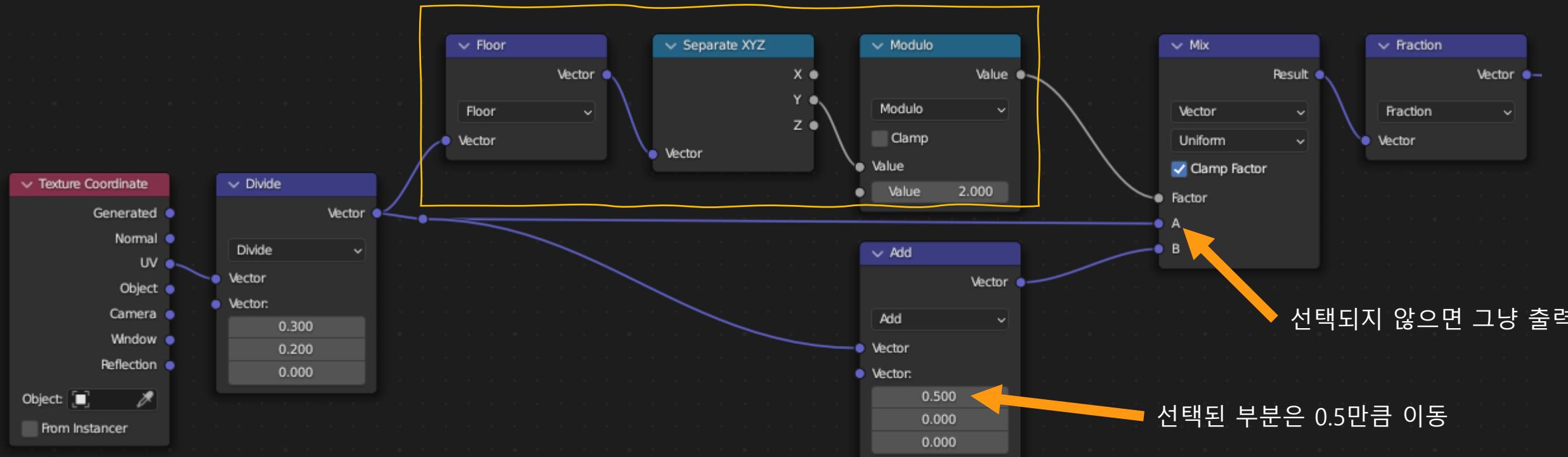


2가 아니고 3으로 나눈 경우에는 0과 1만 출력되지 않습니다.
3으로 나눈 나머지는 0,1,2의 3가지가 될 수 있습니다!
이 때는 Compare를 이용하여 값을 골라줍니다.

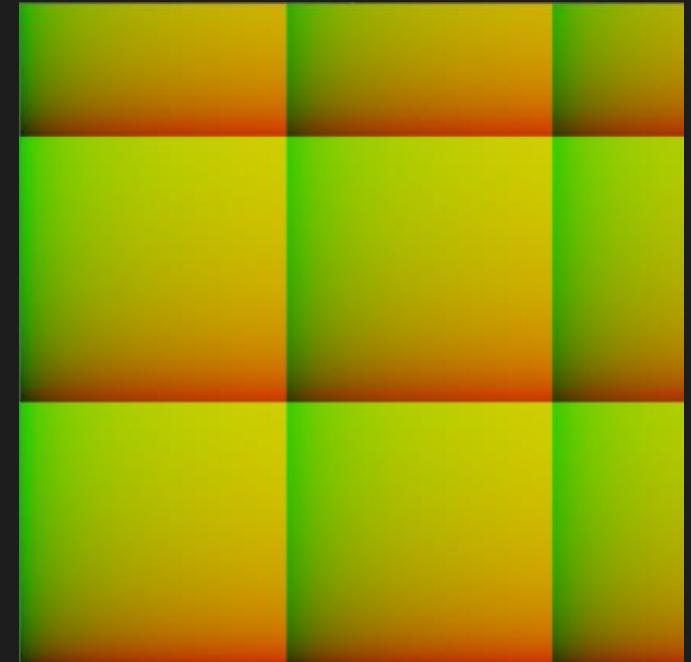
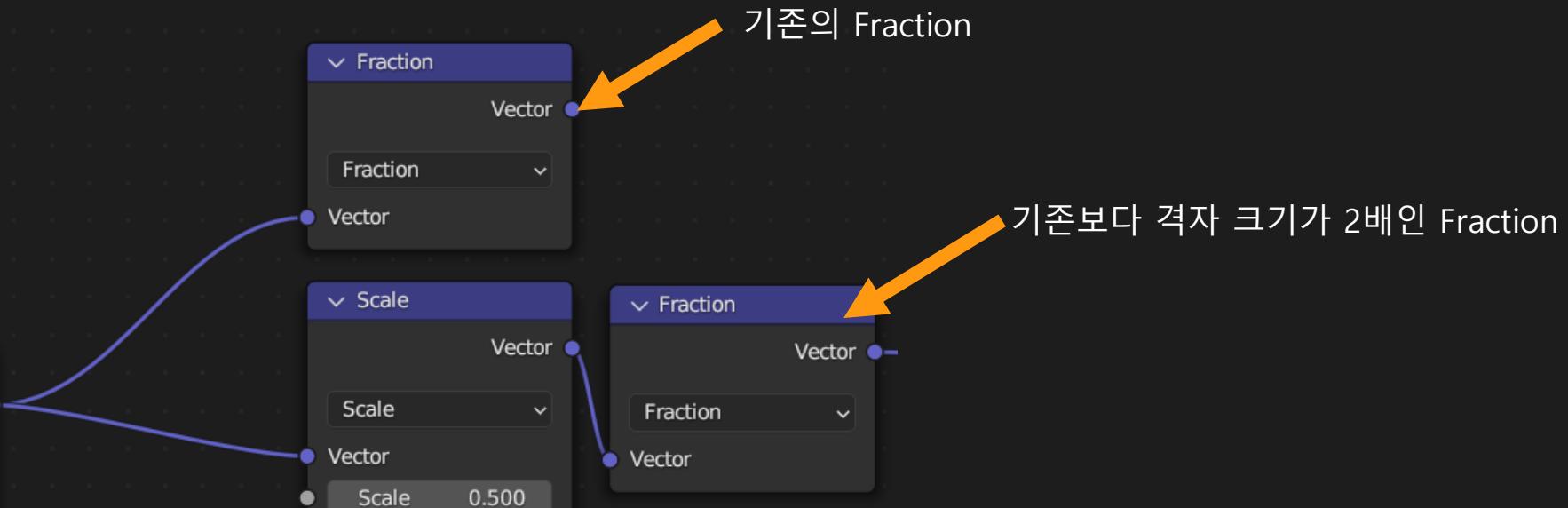


엇갈려 쌓기

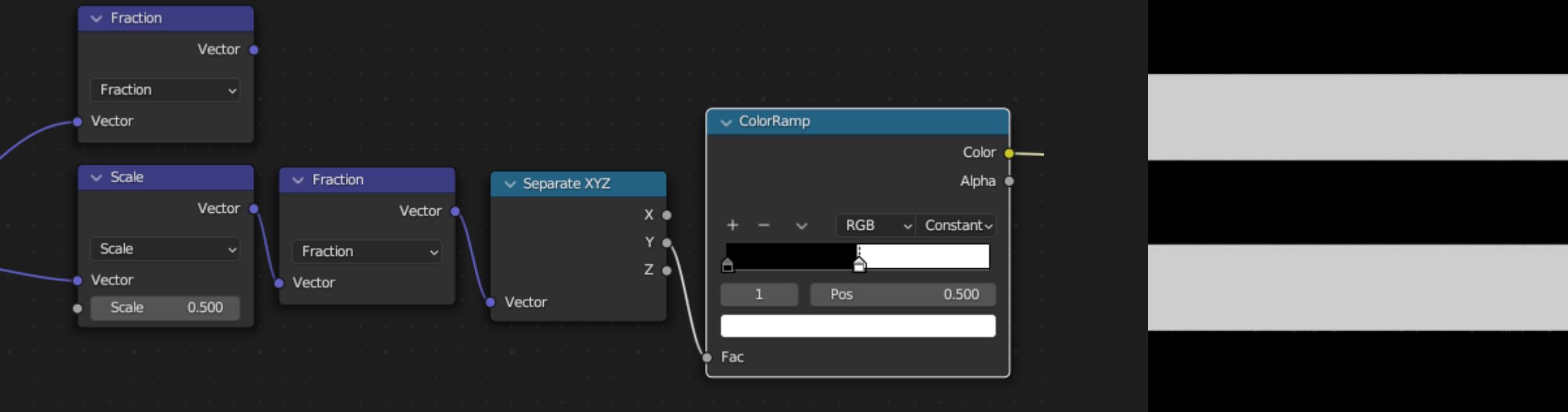
아까 만든 마스크



엇갈려 쌓기- 다른 방법 (1)

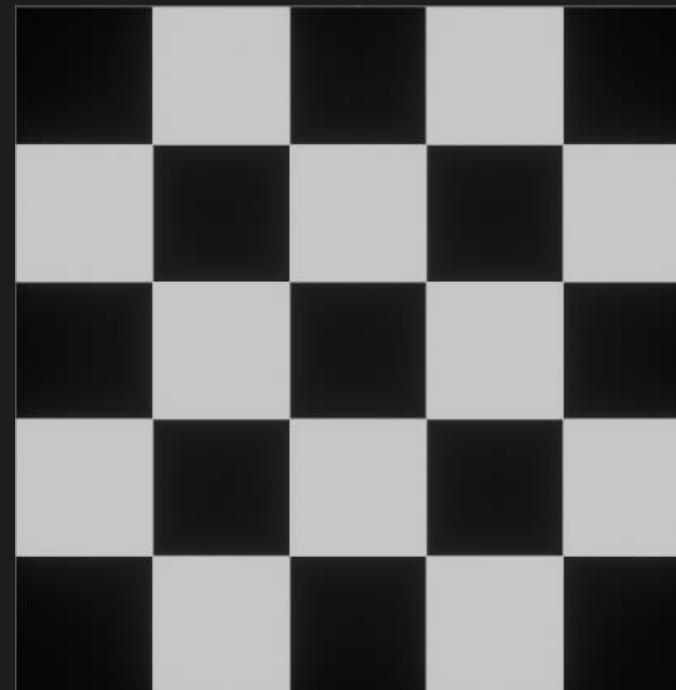
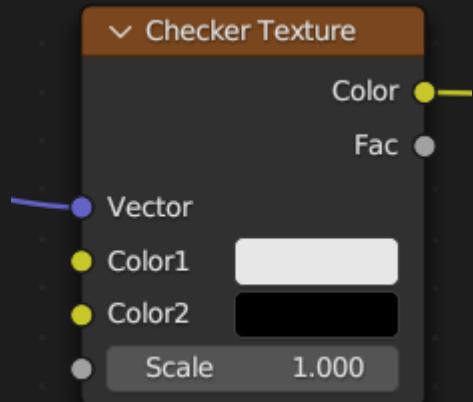


엇갈려 쌓기- 다른 방법 (1)



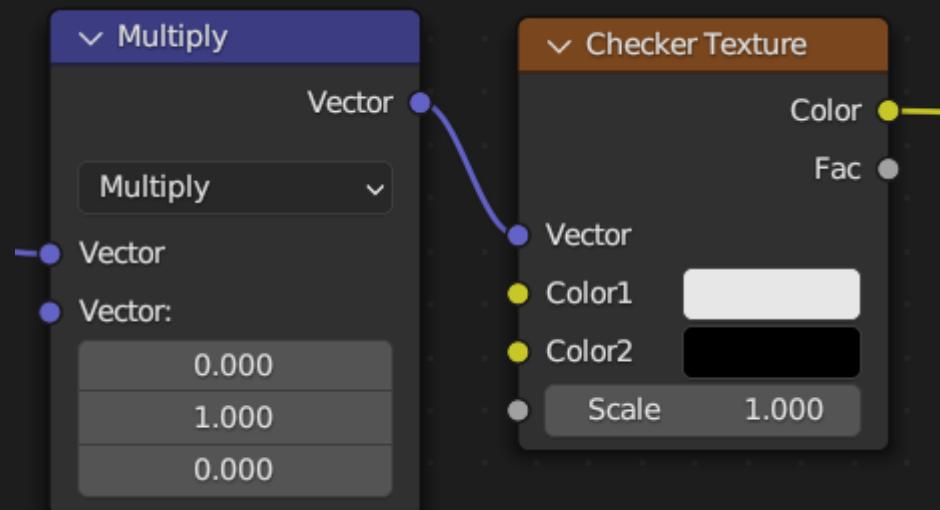
엇갈려 쌓기- 다른 방법 (2)

Checker Texture는 그 자체로 엉갈린 무늬입니다.



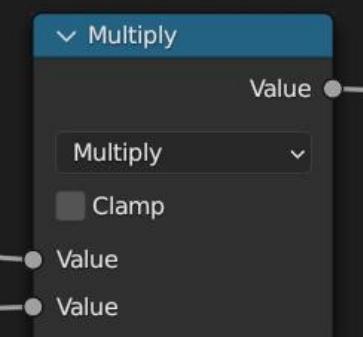
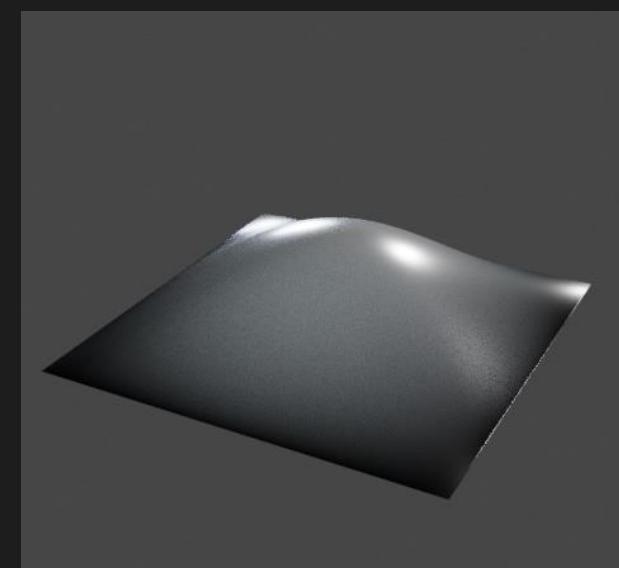
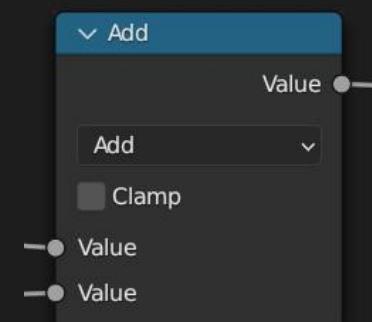
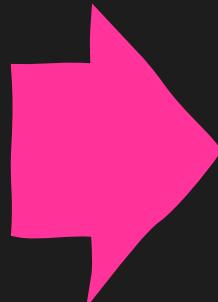
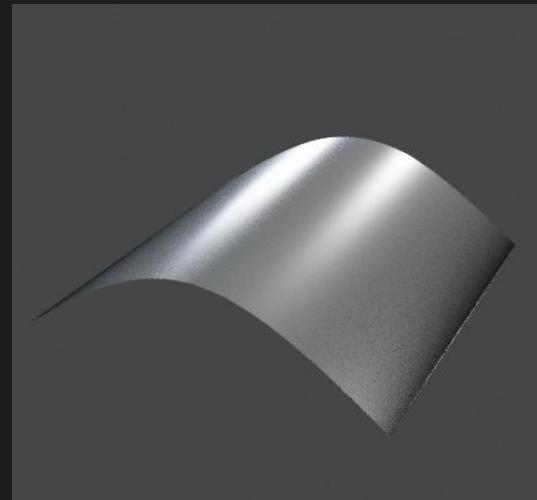
엇갈려 쌓기- 다른 방법 (2)

X 스케일을 0으로 만들어 한쪽으로만 늘어나게 만듭니다.



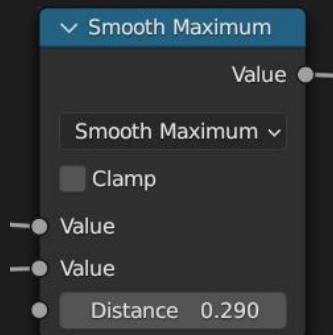
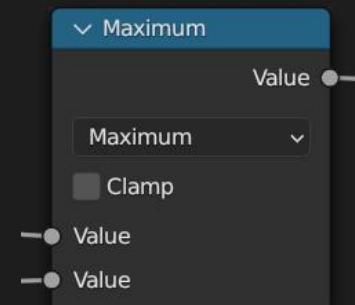
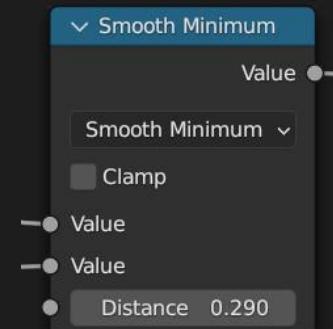
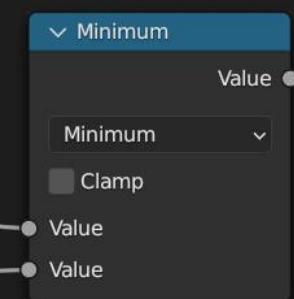
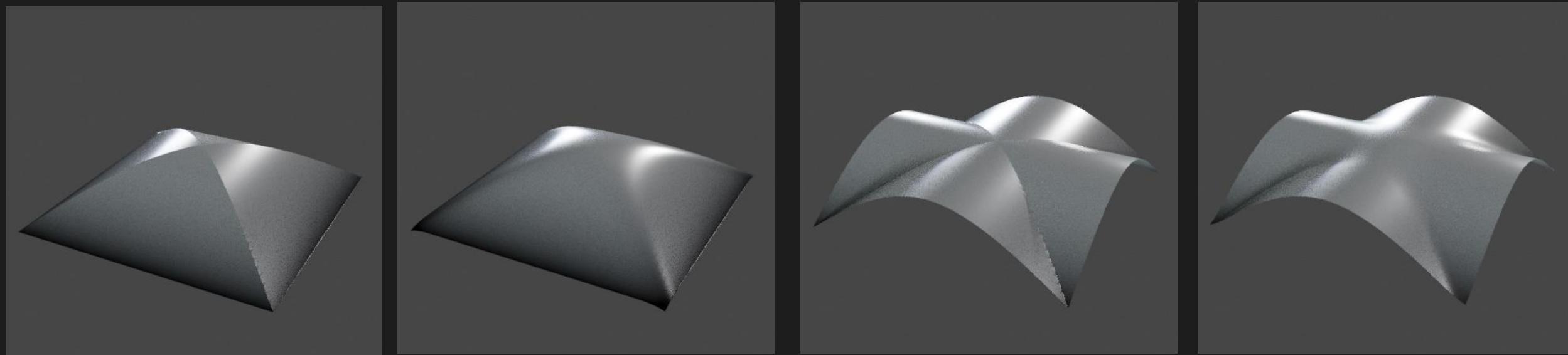
두 그라데이션이 만날 때

Math 노드를 이용한 그라데이션 합성

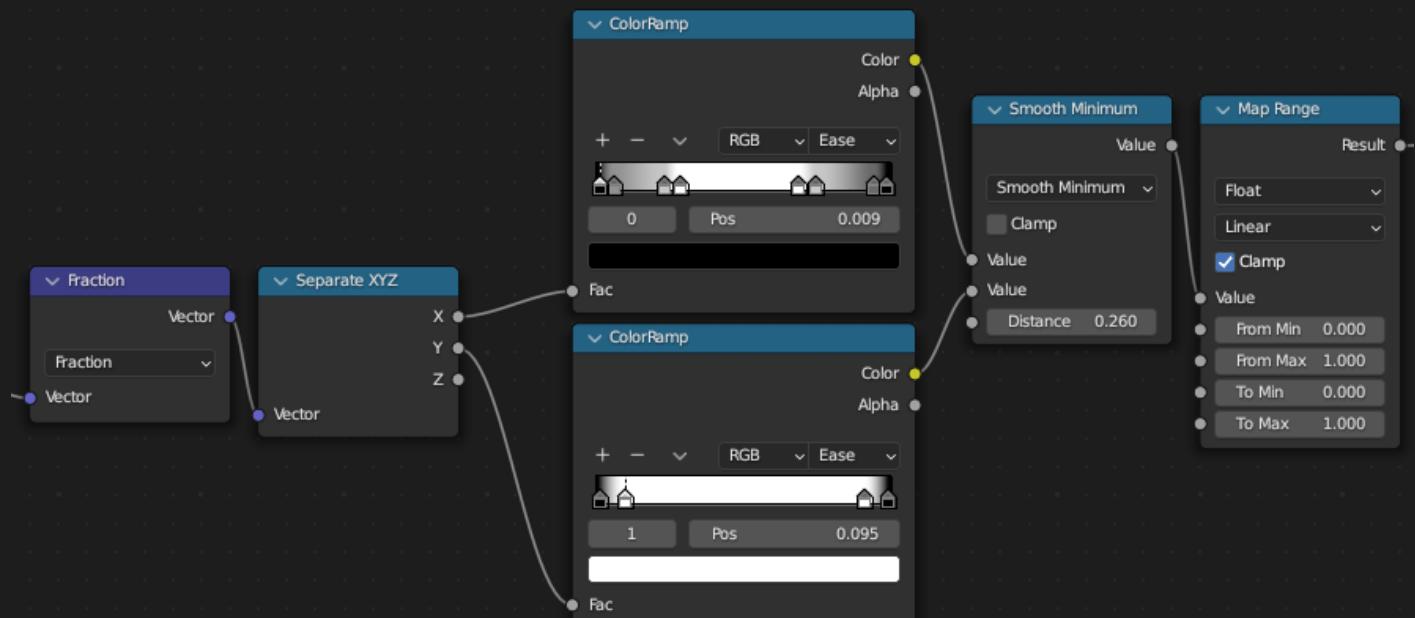
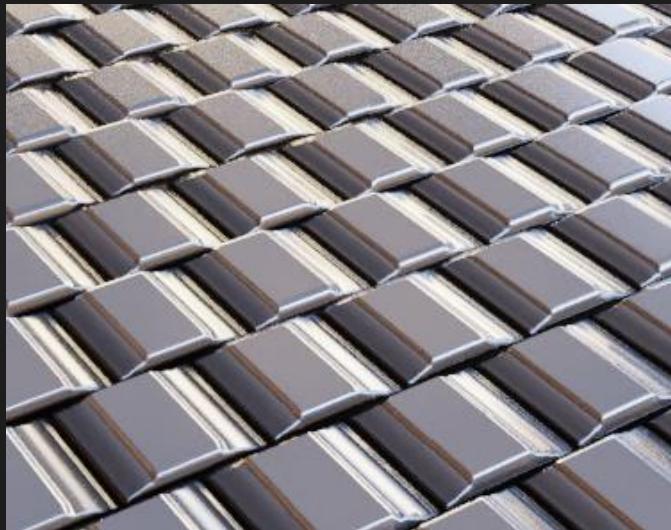
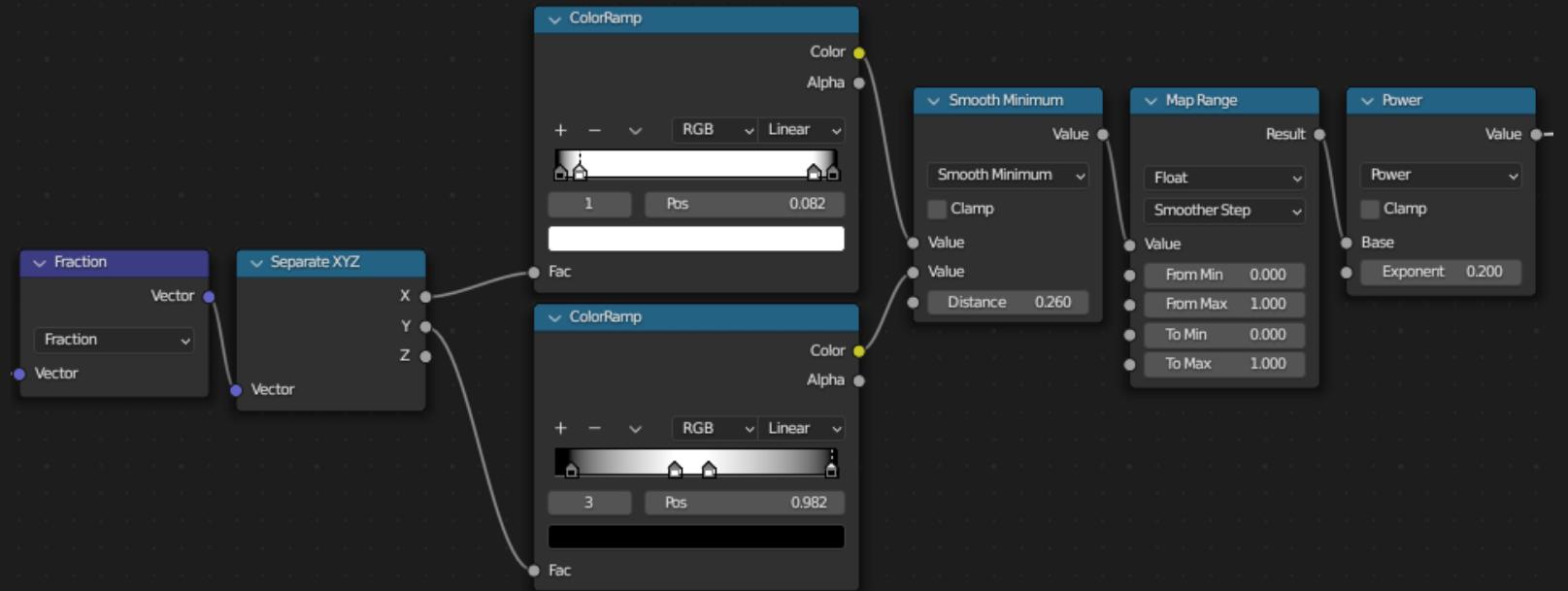


두 그라데이션이 만날 때

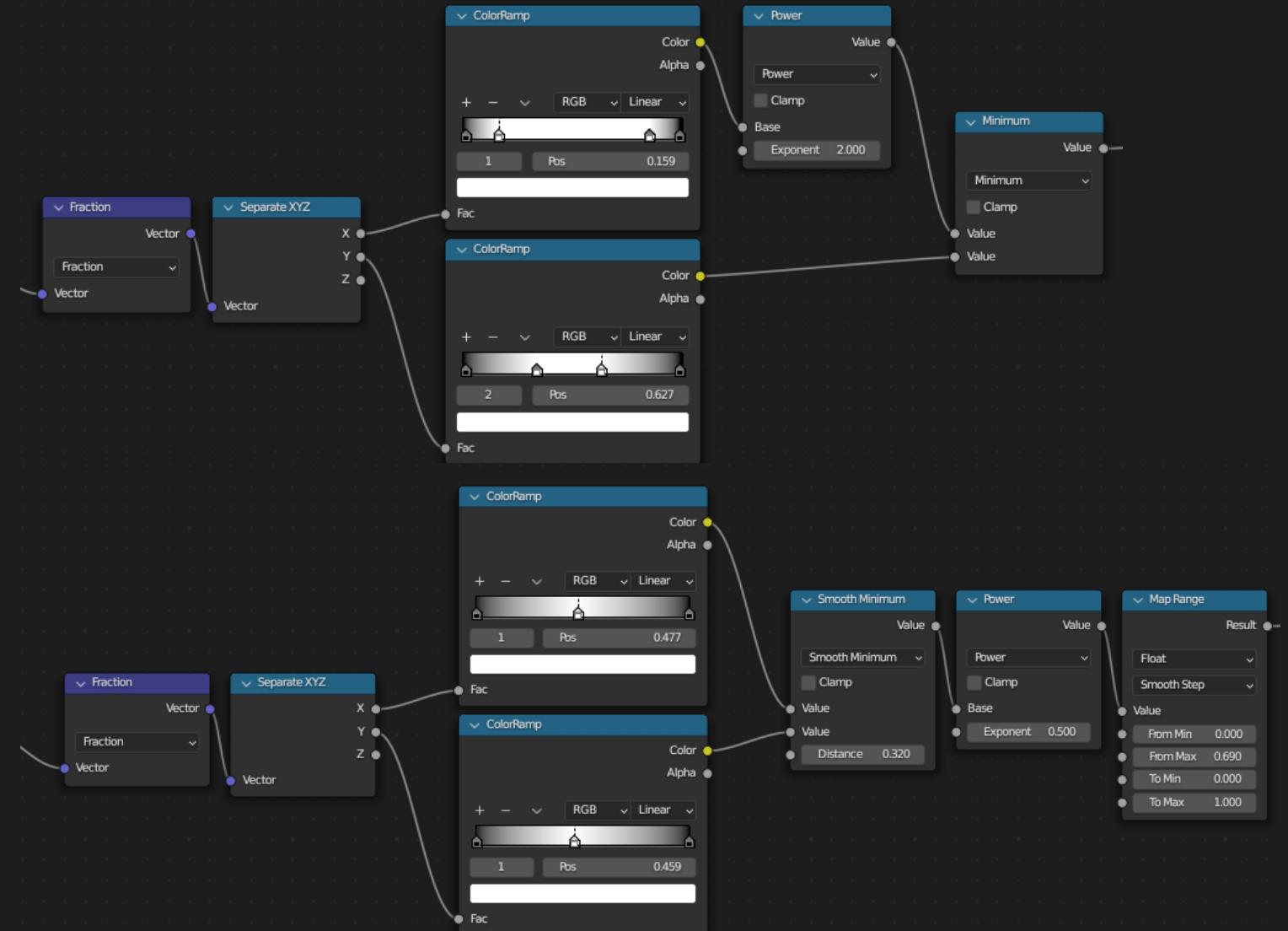
Math 노드를 이용한 그라데이션 합성



프리셋들



프리셋들



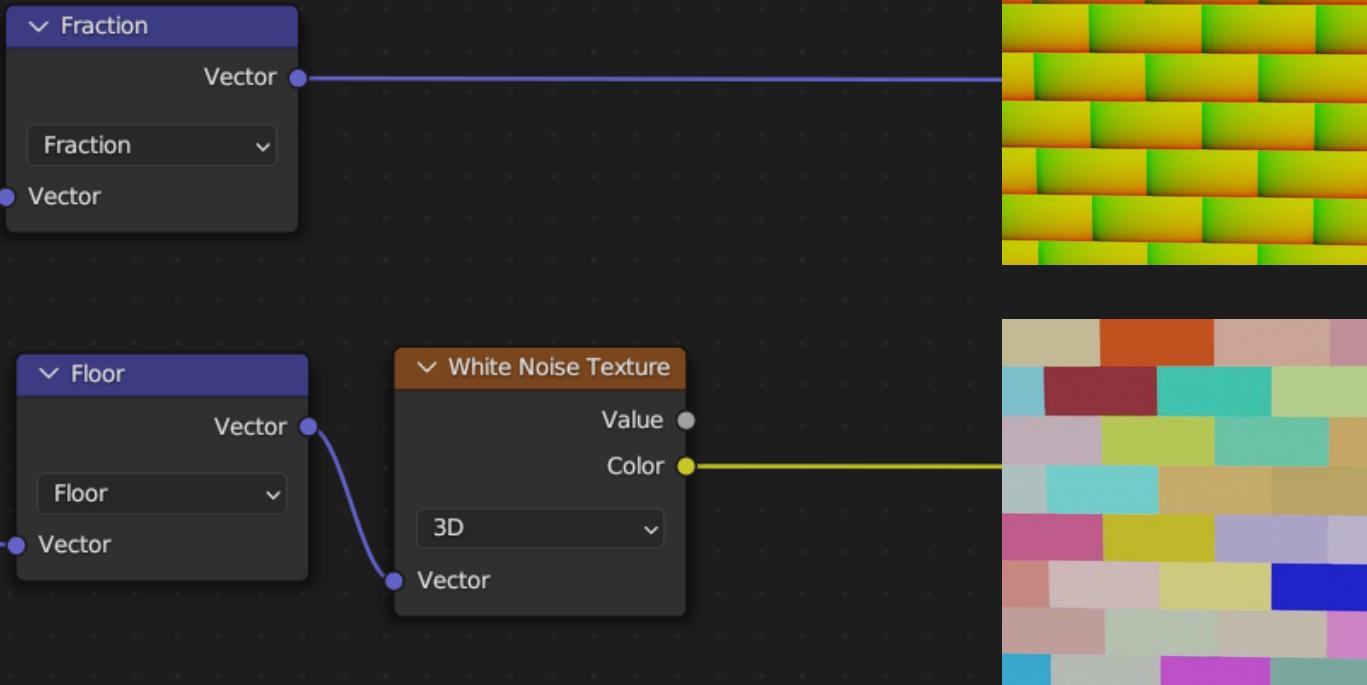
블럭별 랜덤

Fraction으로 좌표를 나눈 연결에, 대신 Floor를 연결해봅시다.

Fraction이 0,1,2,3... 의 좌표를 0에서 1로 반복시킨다면,

Floor는 '내림'을 통해 좌표를 정수값으로 만들어줍니다.

이를 이용하여 블럭별 랜덤값을 만들 수 있습니다.



이러한 랜덤값은 색깔로 이용하는 것 뿐만 아니라
좌표에 Linear Light로 살짝 더해주어 벽돌을 무작위로 움직여 주는 등,
여러가지 방법으로 활용할 수 있습니다.

Semi – Procedural Texture

이미지와 Procedural 한쪽만 사용할 필요는 없습니다.

이 장에서 이야기한 내용의 핵심은 블록 모양으로 좌표를 분배하는 방법입니다.
오른쪽 이미지는 여기에 이미지 텍스쳐를 추가해서, 랜덤
어떻게 사용할지는 여러분에게 달렸습니다.



020강 Procedural Texture - 문양

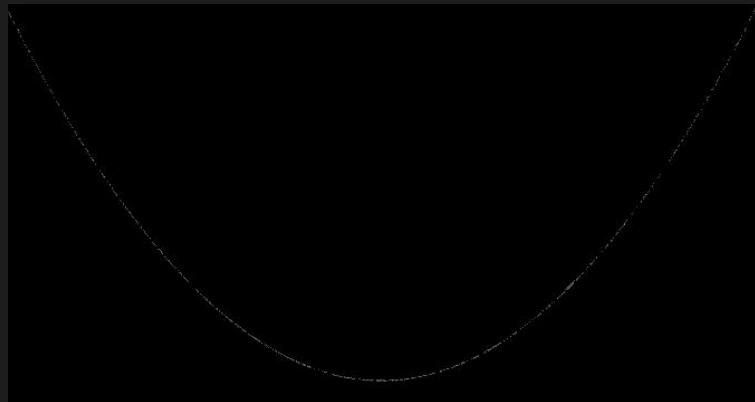
수학 그래프를 문양으로 사용하는 법



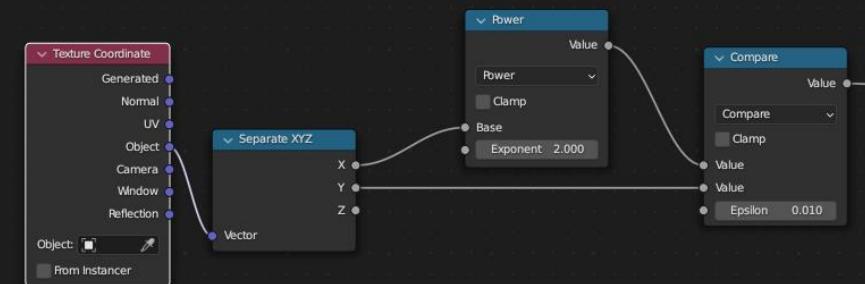
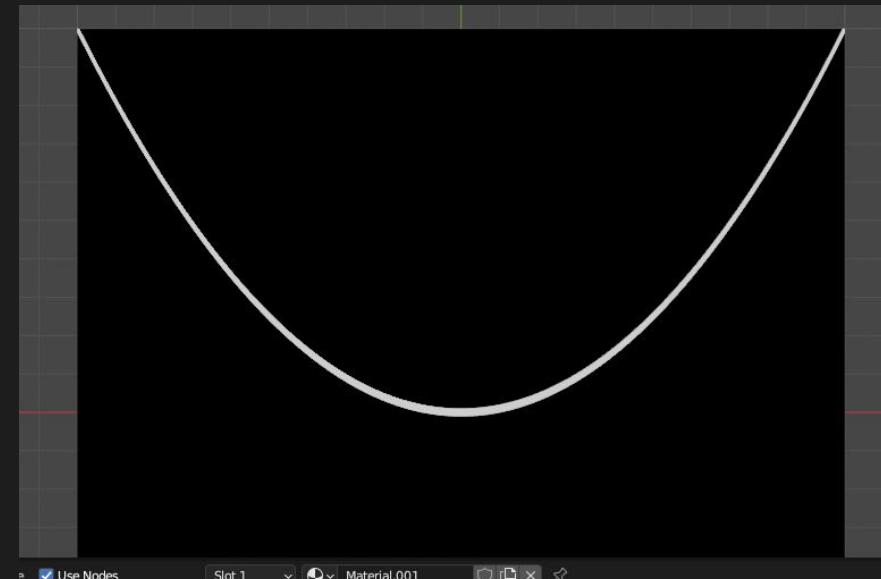
그래프

그래프를 시각화하는 법

우리는 학교에서 $y = x^2$ 을 그려본 적이 있습니다.
그런데 $y = x^2$ 를 정확히 만족하는 점만 찍으면
화면상에 잘 나타나지 않습니다..



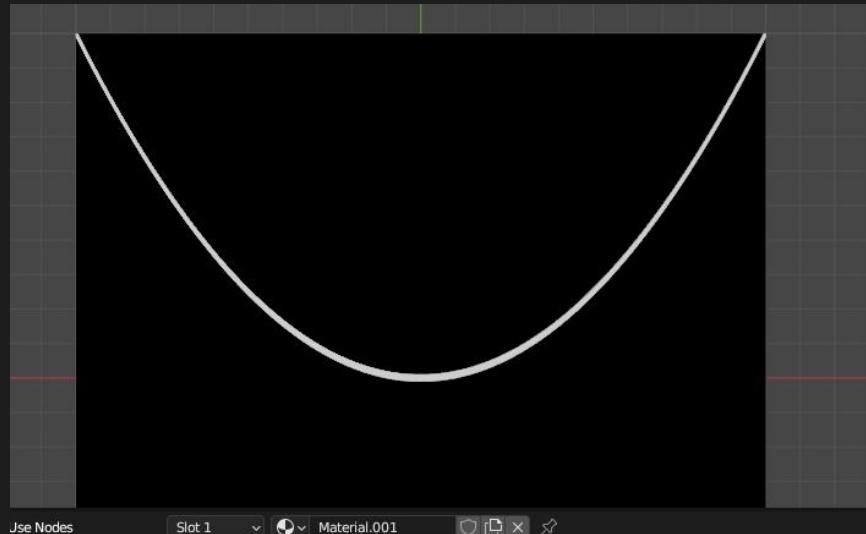
앞에서 Compare노드를 이용해서 그래프를
근사하게 만족하는 값들까지 표현해보는 적이 있습니다.



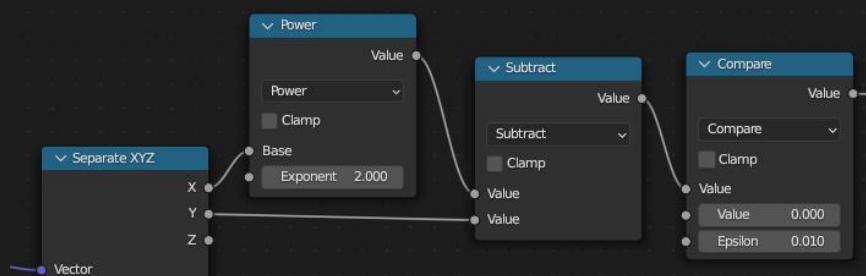
그래프

그래프를 시각화하는 법

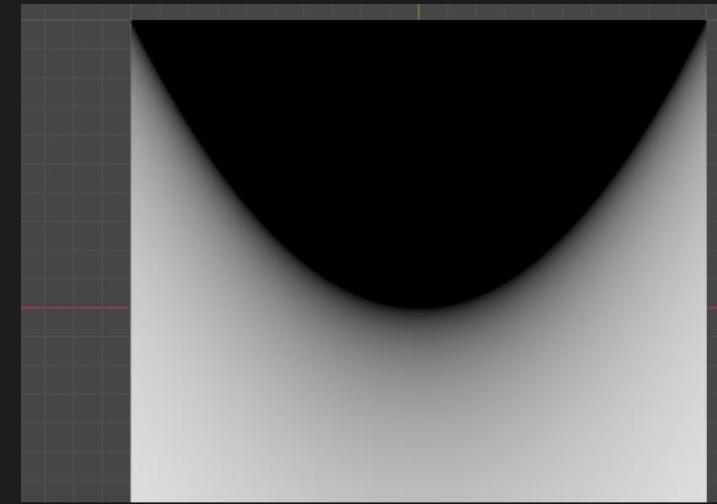
$y = x^2$ 이라는 것은, $x^2 - y = 0$ 과 같습니다.



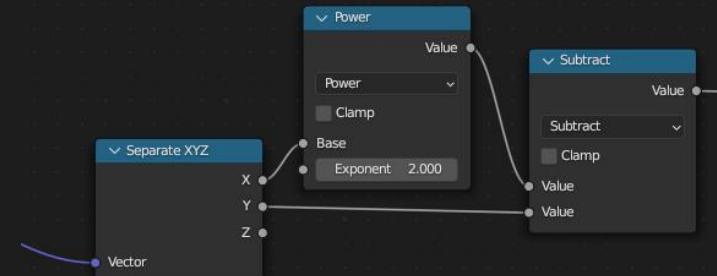
Use Nodes Slot 1 Material.001



좌표의 나머지 부분에서 $x^2 - y$ 는 무슨 값을 가질까요?



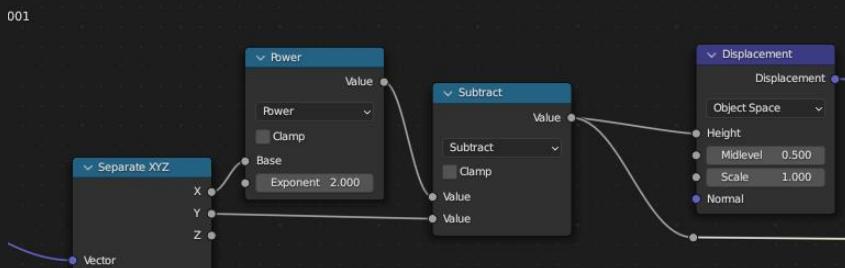
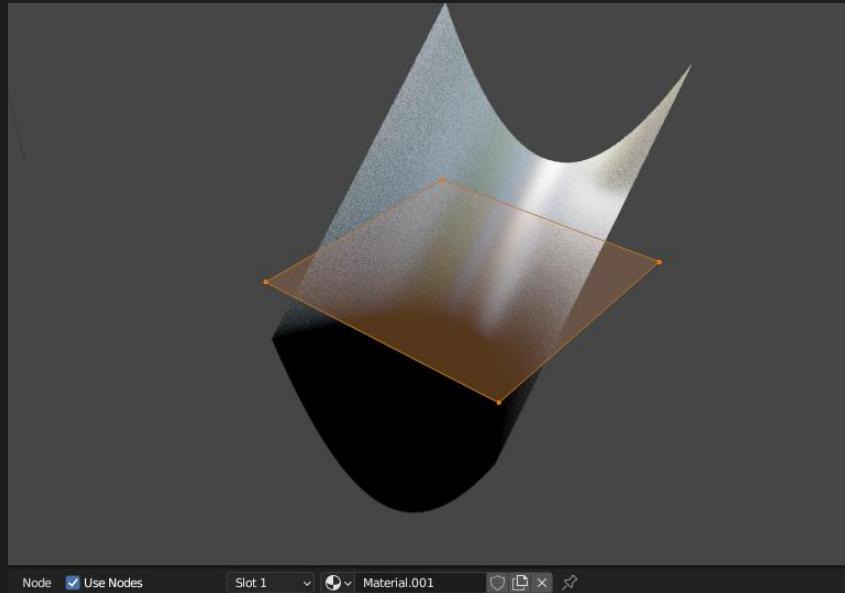
Use Nodes Slot 1 Material.001



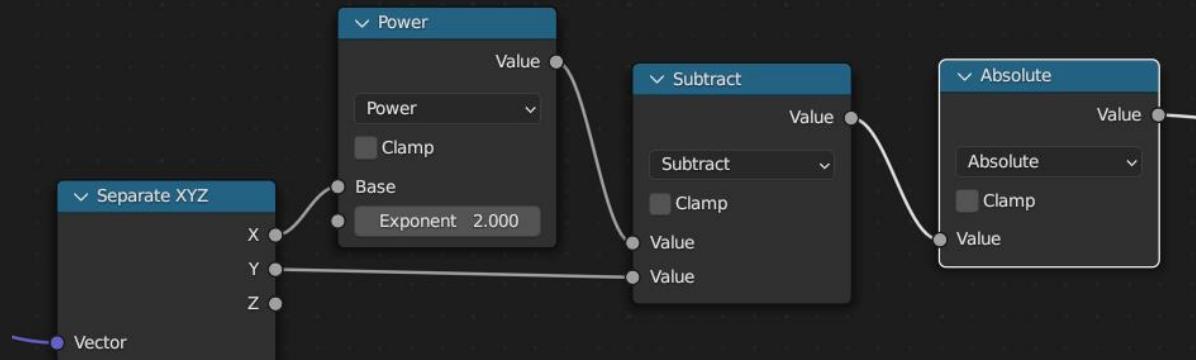
그래프

그래프를 시각화하는 법

Displacement를 이용해 보았습니다.



Absolute 노드를 꽂아봅시다.

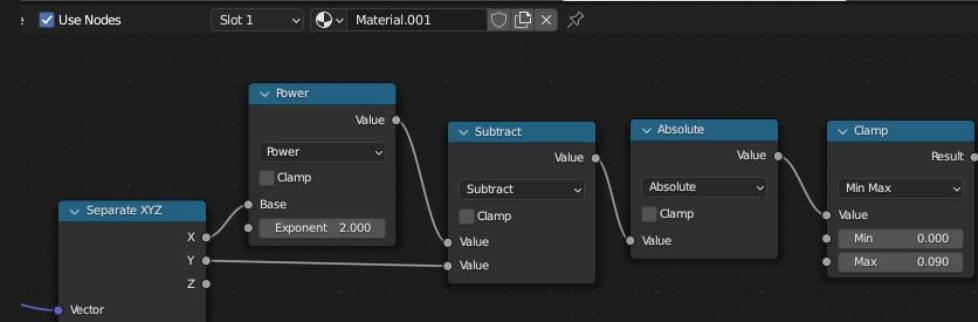
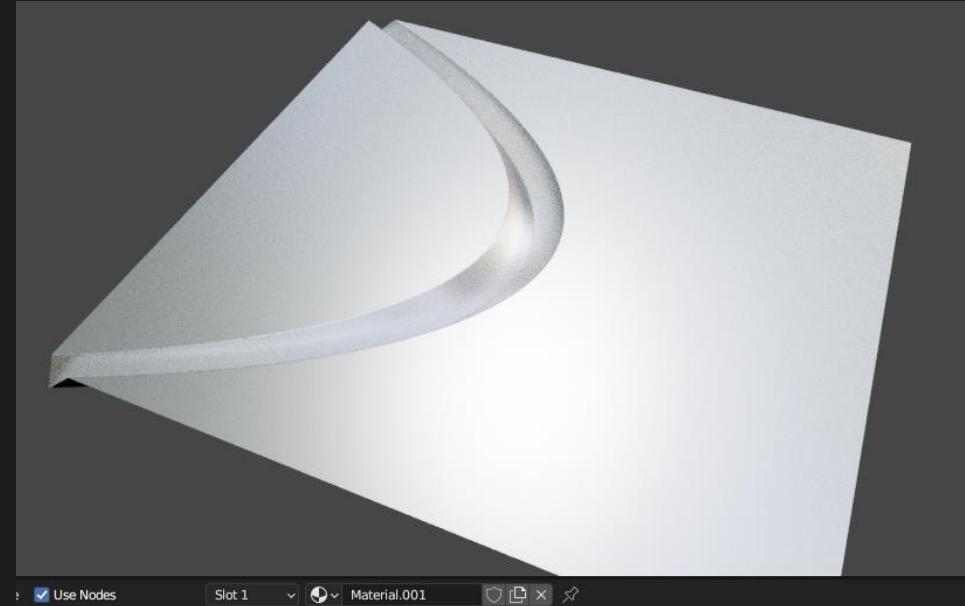
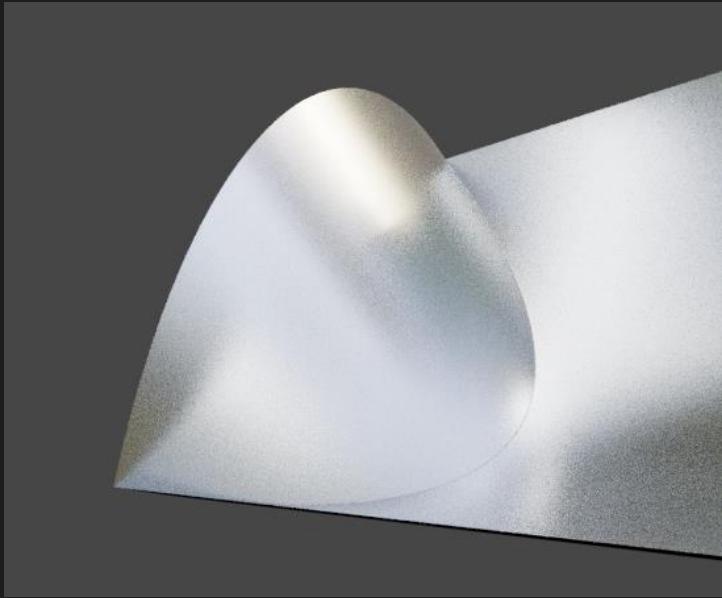


그래프

그래프를 시각화하는 법

Clamp 노드 등을 연결하여 범위를 제한해 봅시다.

포물선으로 파여진 모양새입니다.



정리하면,

$y=f(x)$ 의 함수를 활용하는 법은 :

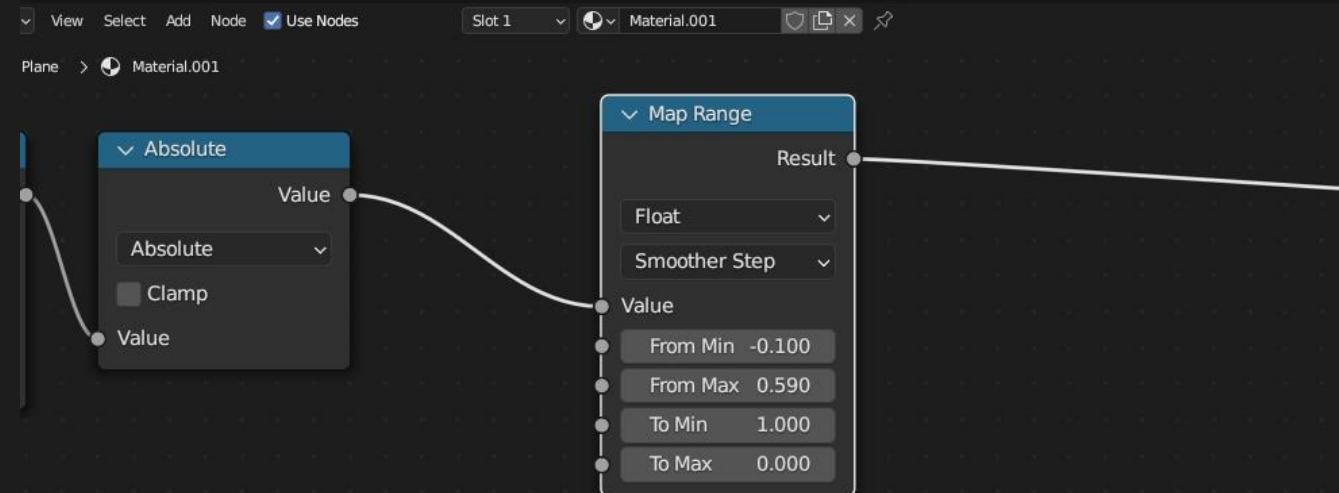
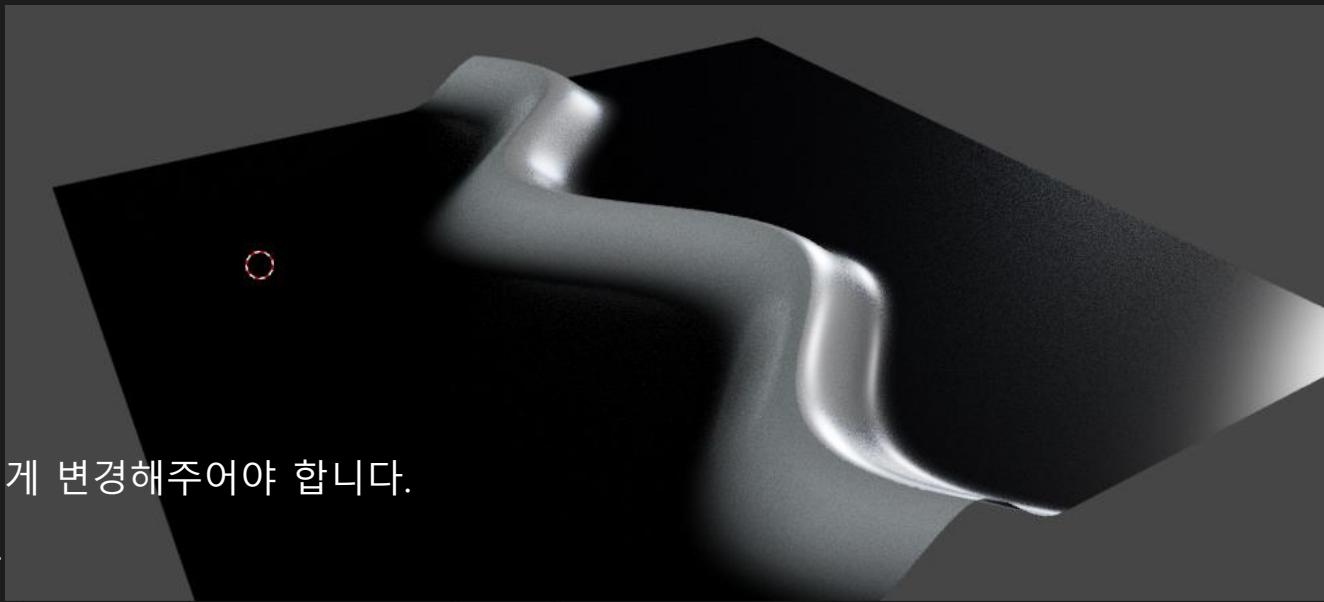
1. 모두 이항하여 $y-f(x)$ 혹은 $f(x)-y$ 의 식을 만들고,
2. 그것에 absolute를 붙인다.
3. 마지막으로 취향껏 범위를 조절한다.

예시

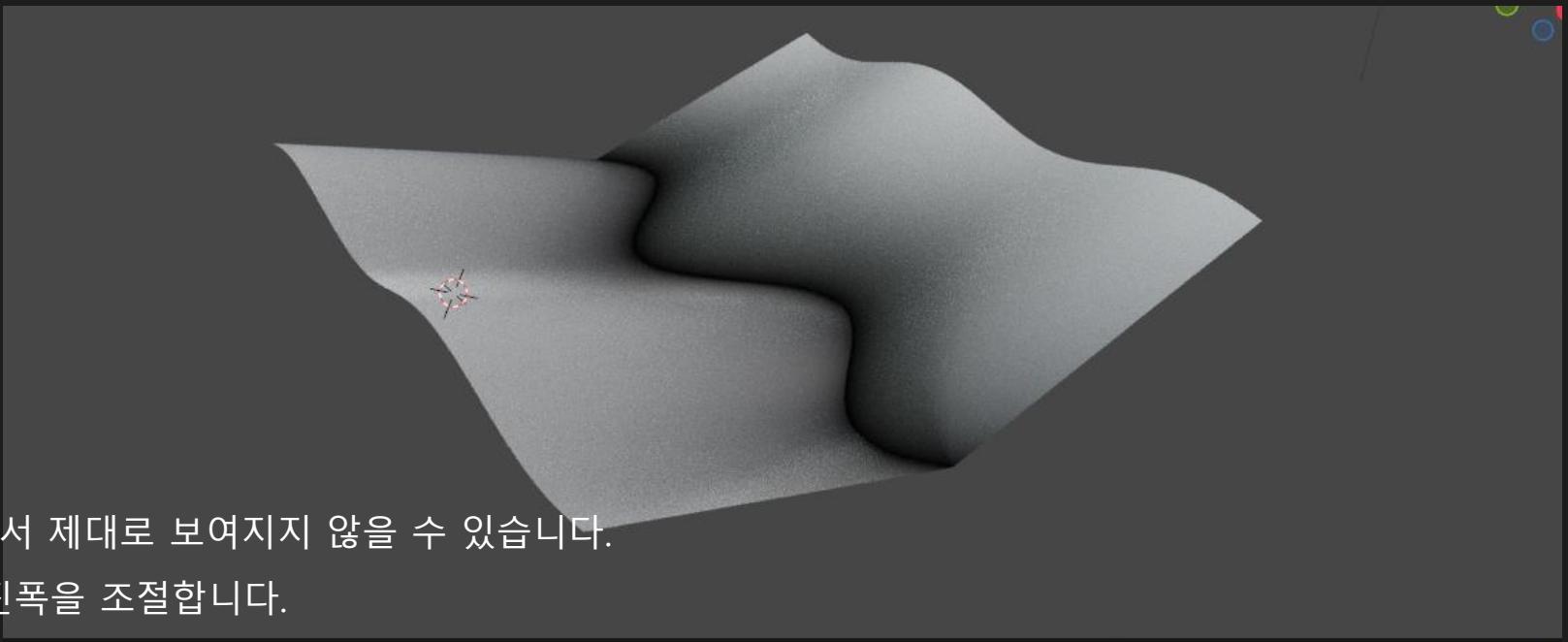
범위 제한

Absolute를 꽂은 뒤에는, 범위를 제한하고 굴곡도 부드럽게 변경해주어야 합니다.

Map range 노드를 이용하면 한번에 처리할 수 있습니다.



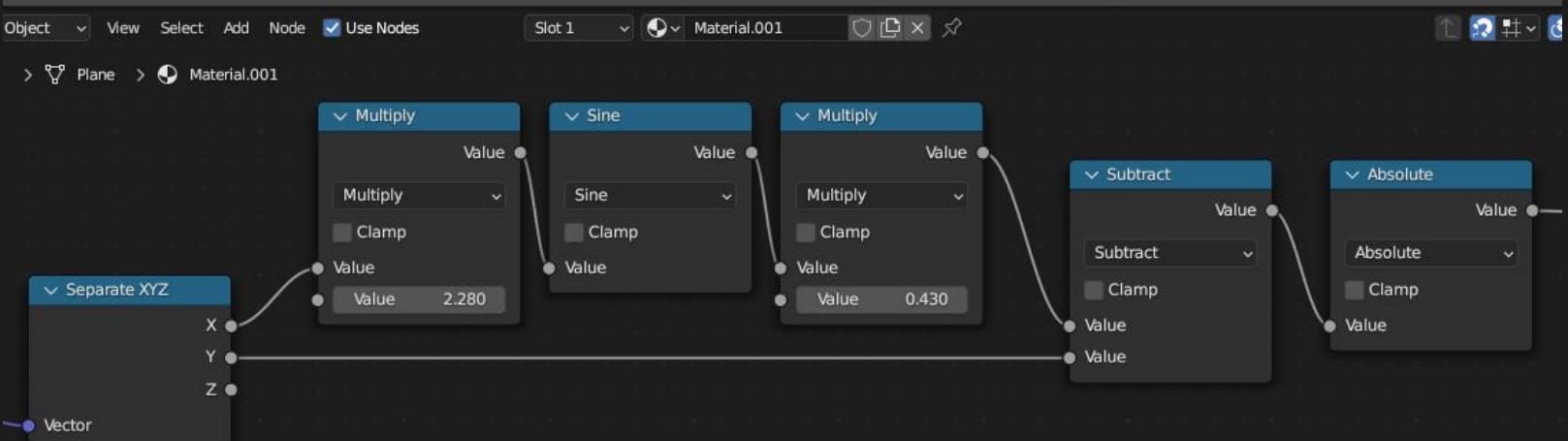
예시



삼각함수

삼각함수는 주기가 길기 때문에, 좁은 범위에서 제대로 보여지지 않을 수 있습니다.

삼각함수의 앞뒤로 multiply를 붙여 주기와 진폭을 조절합니다.



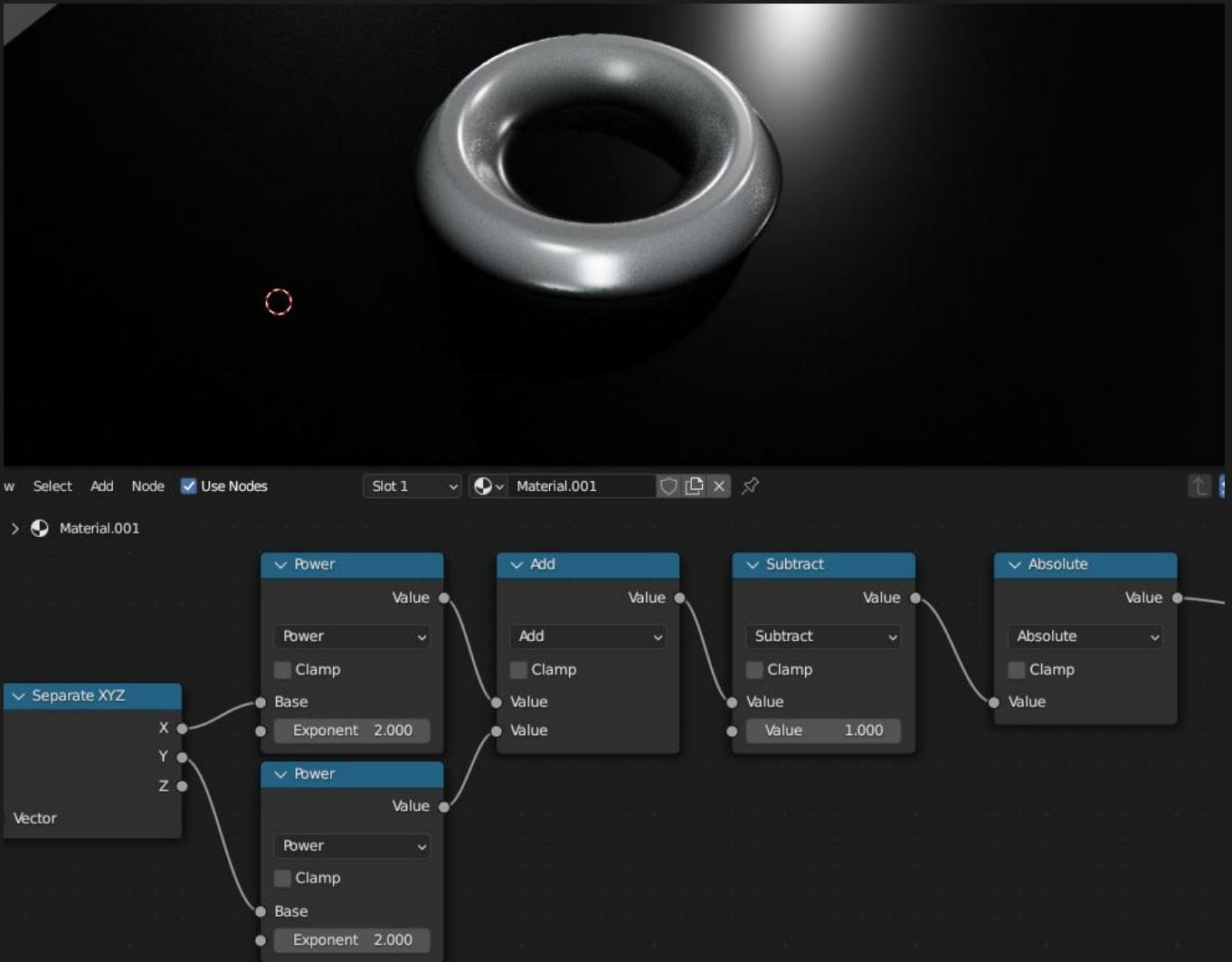
응용

원의 방정식

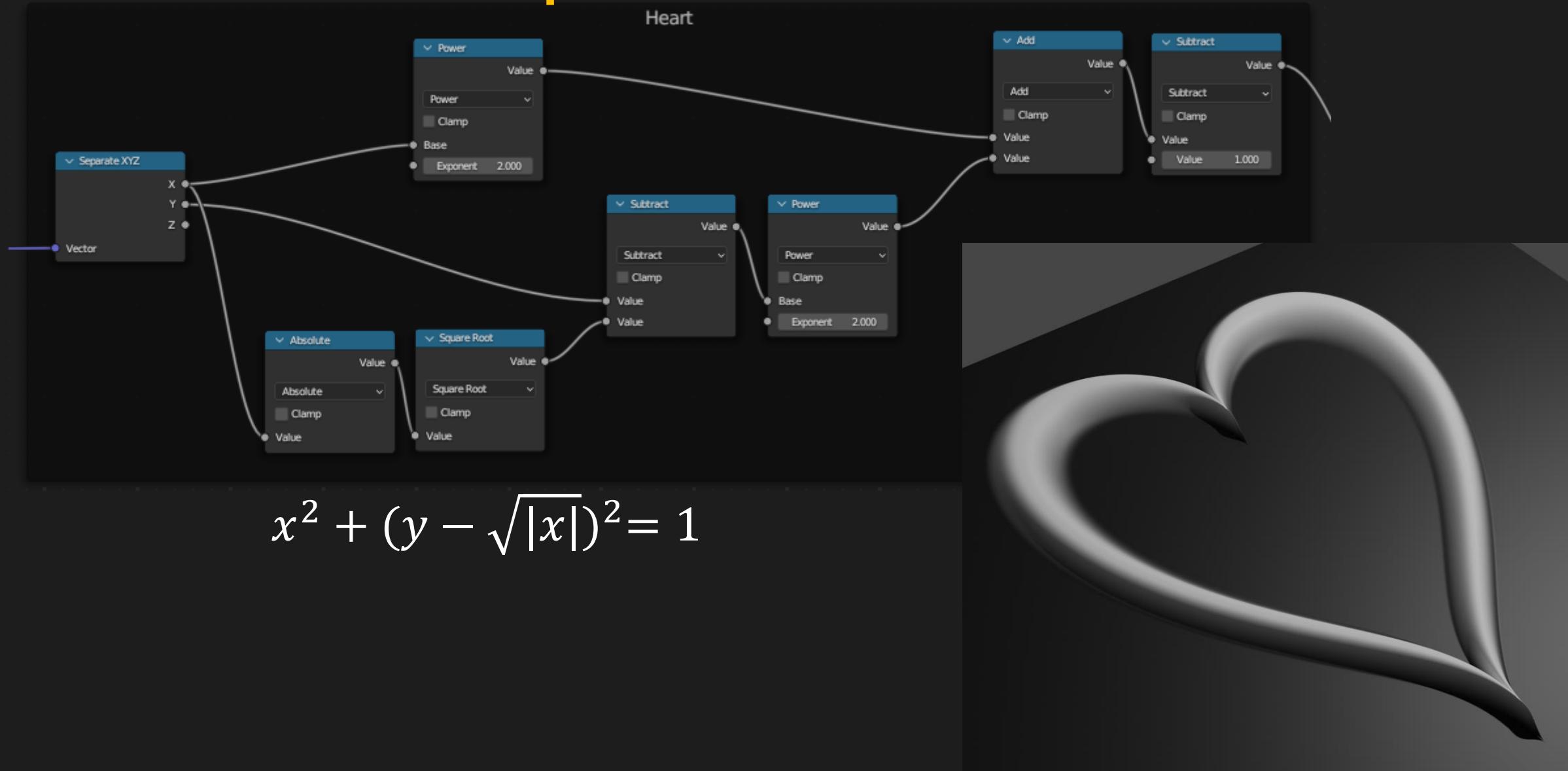
$Y=f(x)$ 꼴이 아니어도 비슷한 원리로 만들 수 있습니다.

$x^2 + y^2 = 1$ 인 원의 방정식을

$x^2 + y^2 - 1$ 로 바꾸면 됩니다.



하트의 Parametric Equation



021-022강 Advanced Procedural Texture – Ornament



Nodevember

매년 11월 Procedural Shading / Modeling 기술을 한계까지 사용하여 주어진 주제를 완성하는 이벤트가 열립니다.

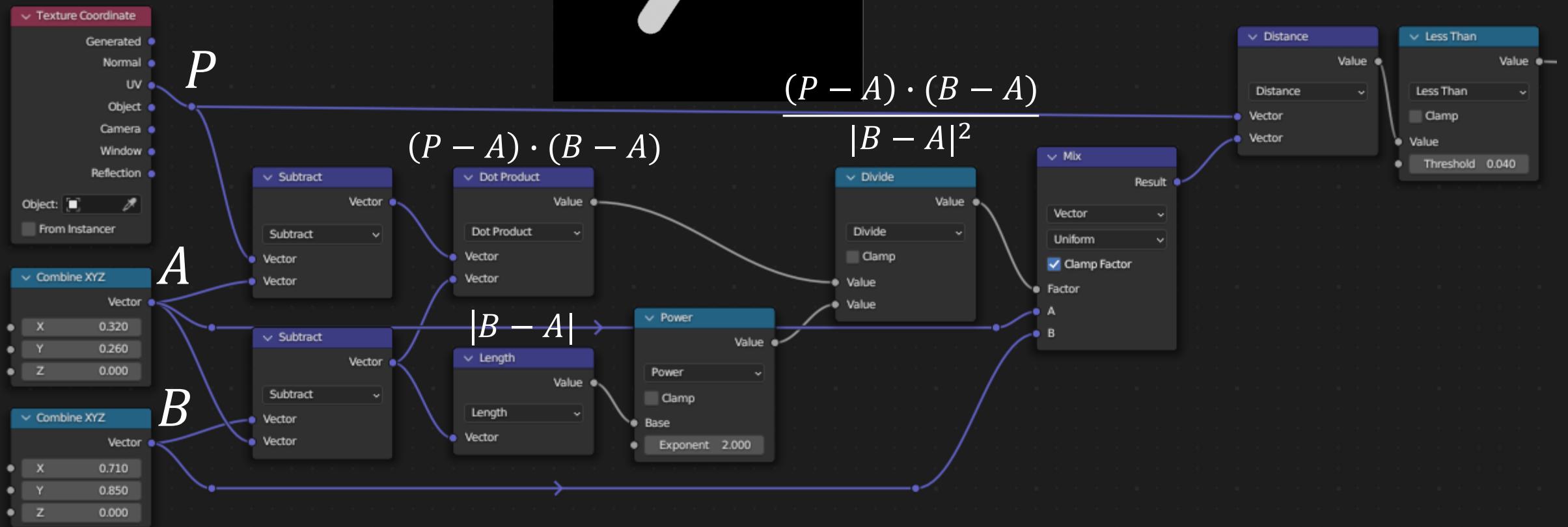
규칙 :

1. 이미지 텍스쳐 사용 불가능.
2. 외부 데이터 불러오기 불가능.

Ornament



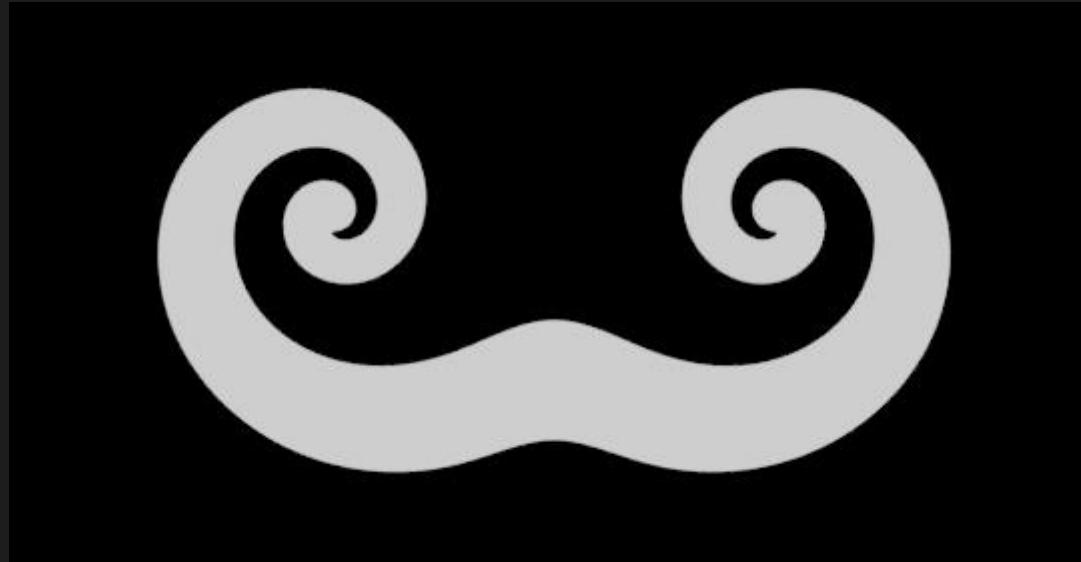
'선분'을 그리는 방법?



준비

타협점 찾기

1. 직접 형태를 만들지 않습니다.
2. 하나의 형태에만 집중합니다.
3. 대체할 수 있는 비슷한 모양을 찾습니다.



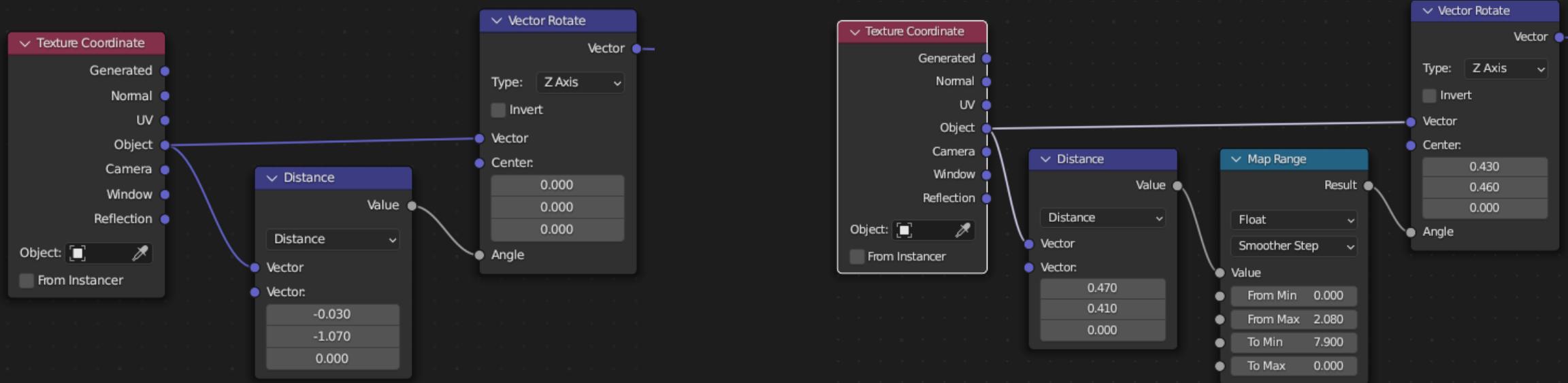
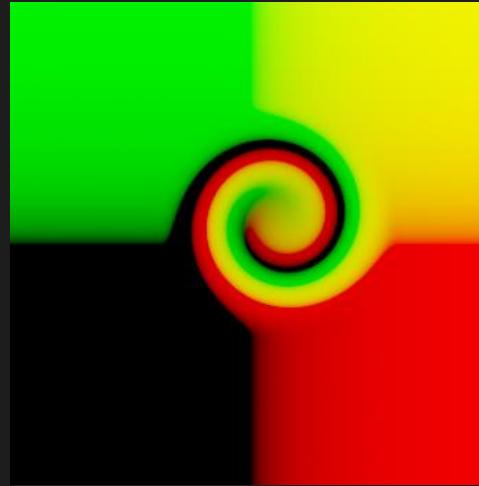
우선 이 소용돌이 형태에 집중합니다.



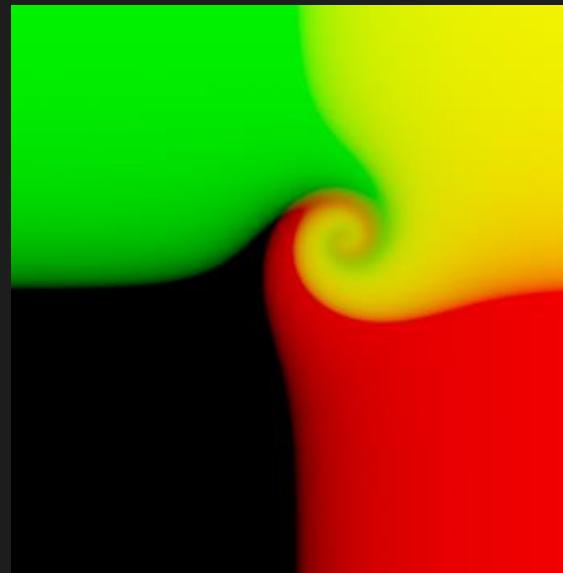
한 쪽만 만들어봅시다.

Spiral

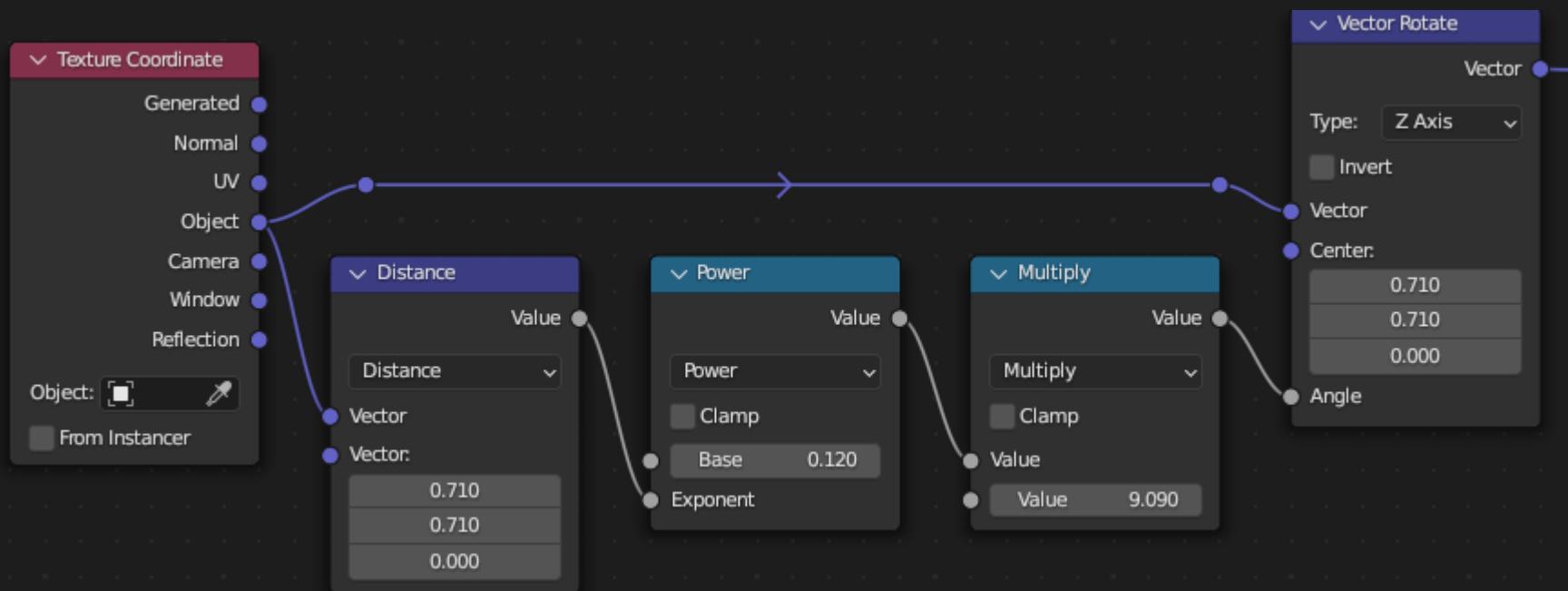
좌표 자체를 회전시킵니다



Spiral

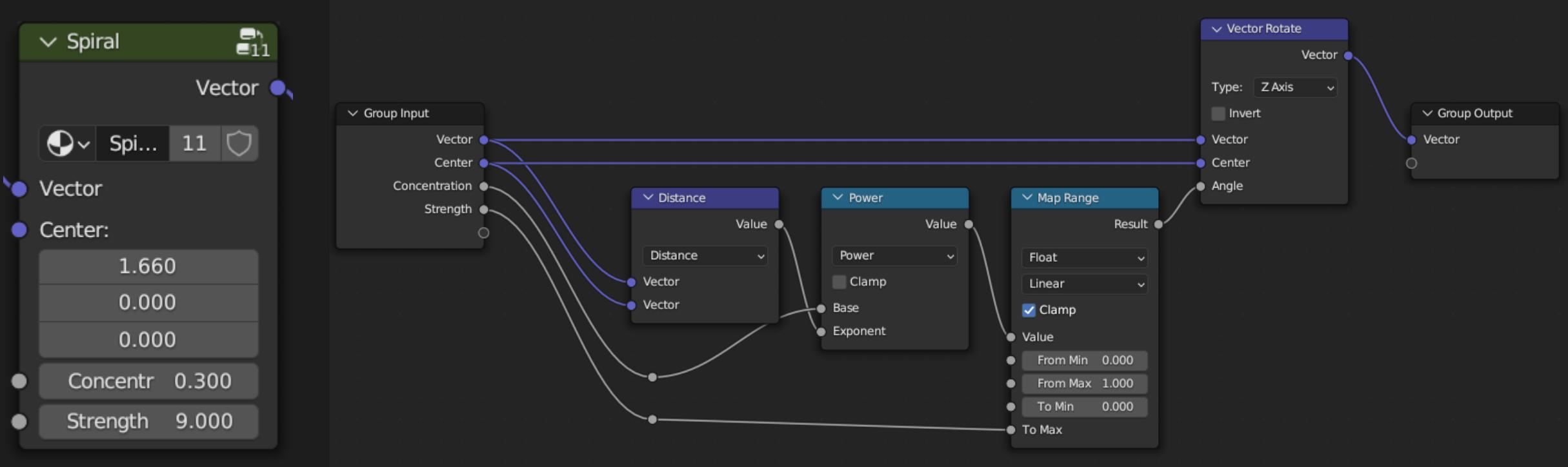


Power 노드(exponent 연결)를 이용하면 영향력을 부드럽게 제한시킬 수 있습니다.



Spiral Nodegroup

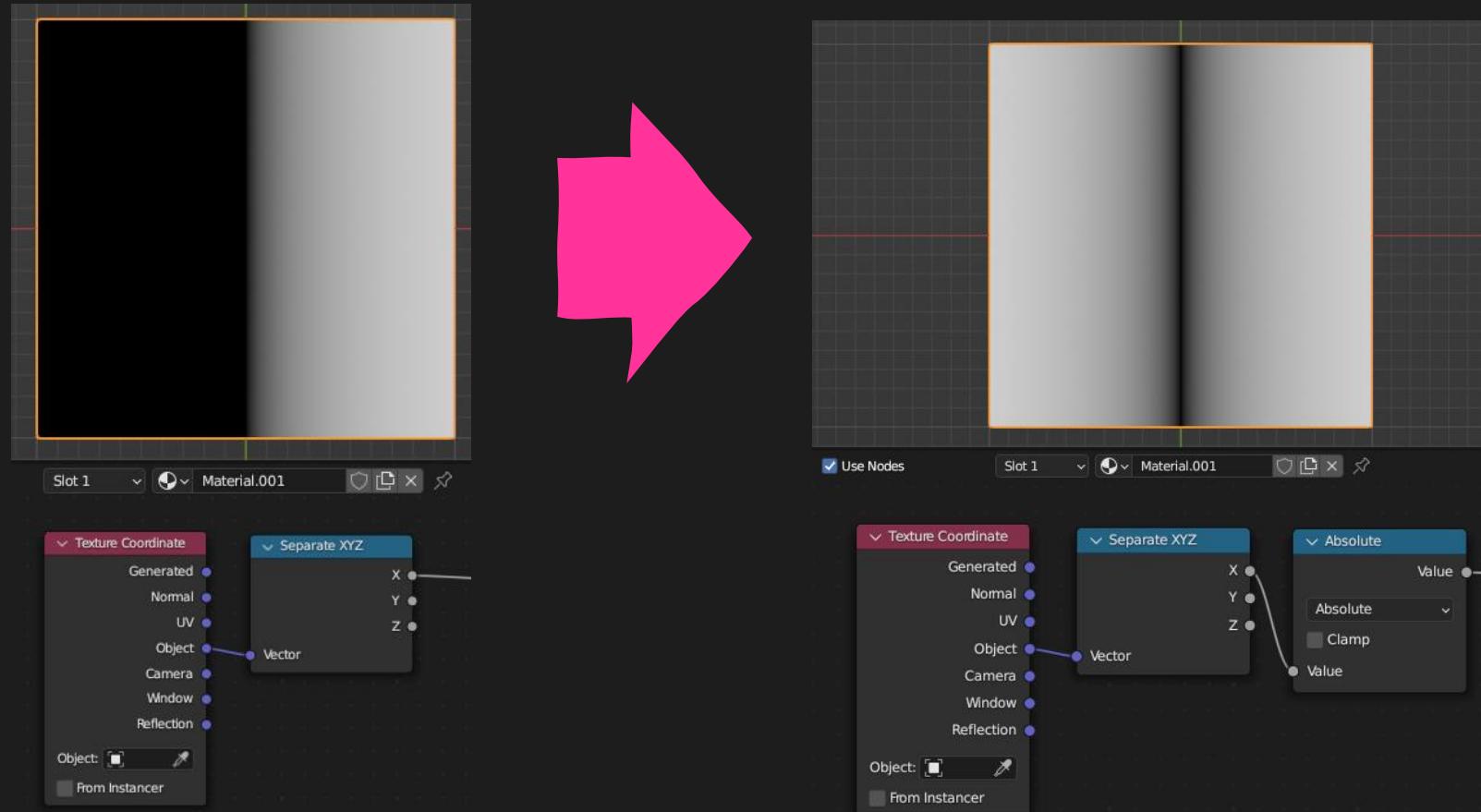
좌표를 소용돌이치게 만드는 부분만 따로 노드그룹으로 묶어 관리해 봅시다.
연속으로 연결하면 여러 개의 무늬도 만들 수 있습니다.



Mirror

거울대칭

이미지를 좌우대칭 시킬 수는 없으므로, 좌표를 이용해야 합니다.
즉, 왼쪽 좌표와 오른쪽 좌표를 좌우대칭으로 만들어야 합니다.

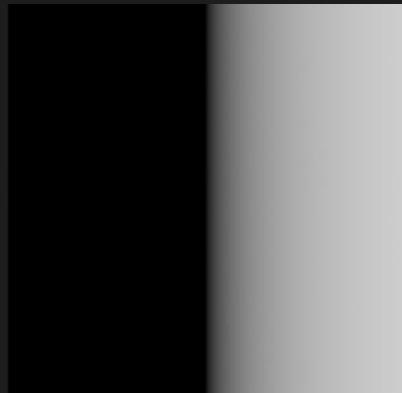


Mirror

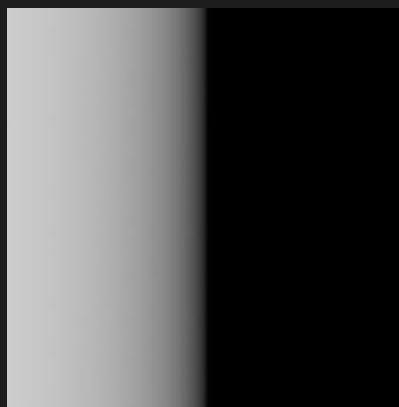
부드러운 거울대칭

경계를 부드럽게 만들고 싶지만 부드러운 absolute는 없습니다..

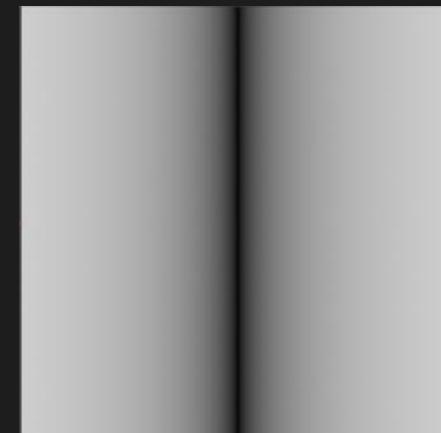
Smooth를 조절할 수 있는 연산은 minimum, Maximum 뿐입니다. 둘중 하나를 이용해야 합니다.



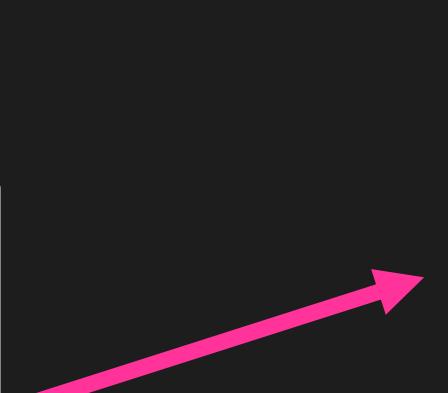
원본 x좌표



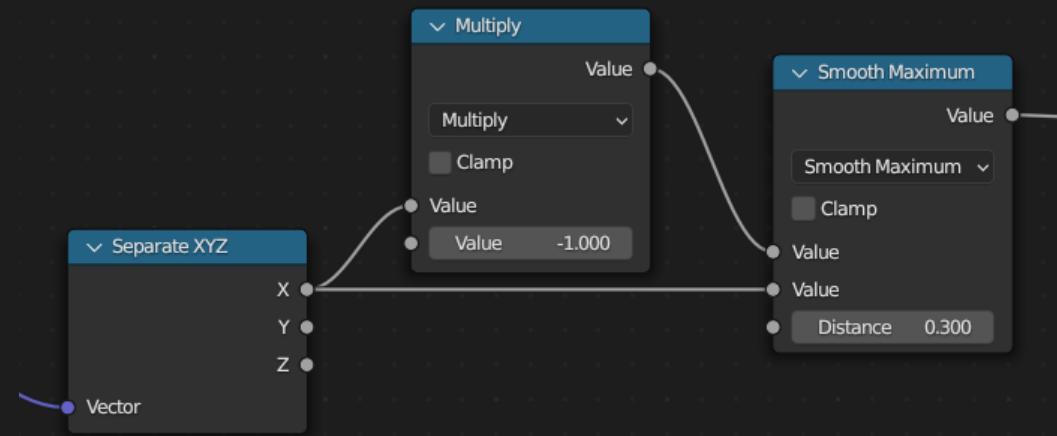
Multiply -1



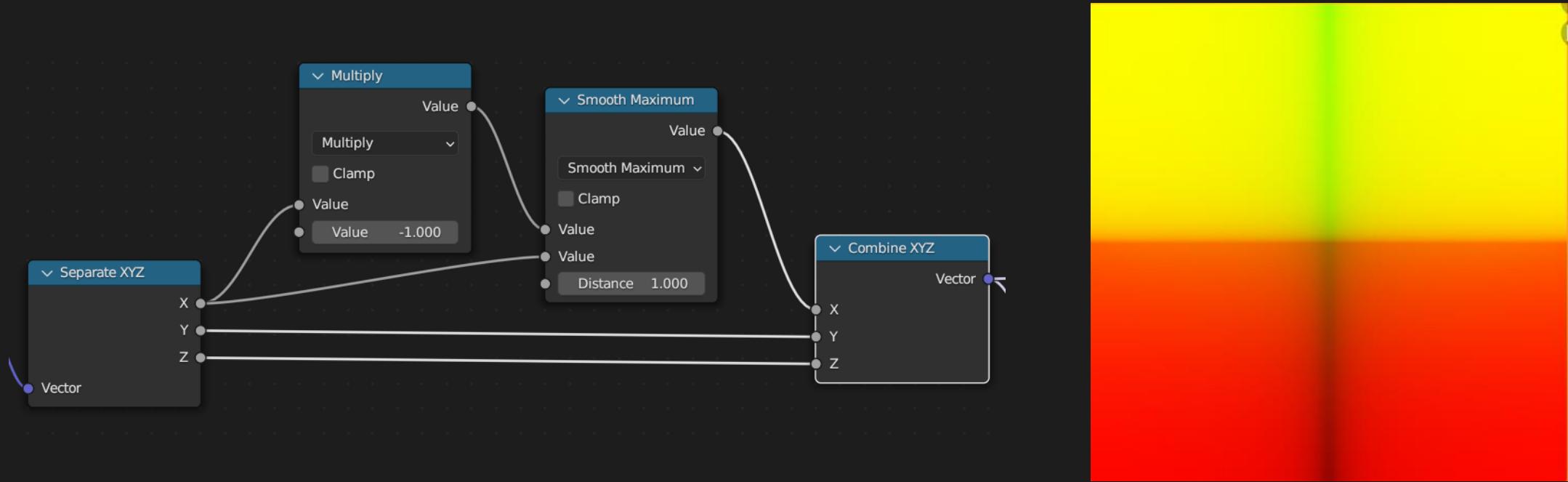
Maximum



Smooth Maximum



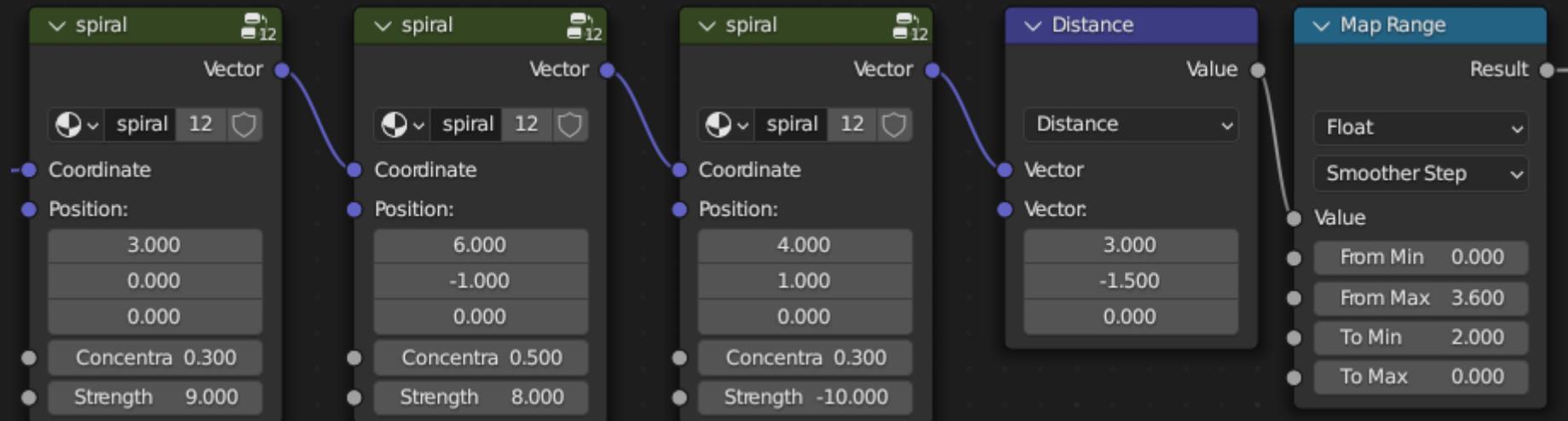
Mirror



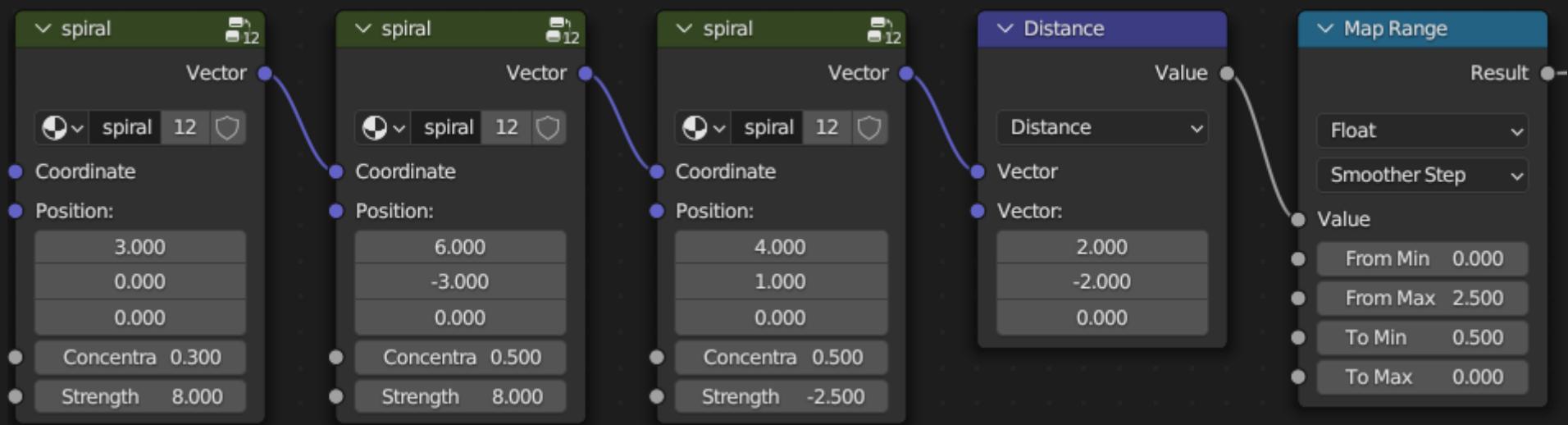
예시

-10

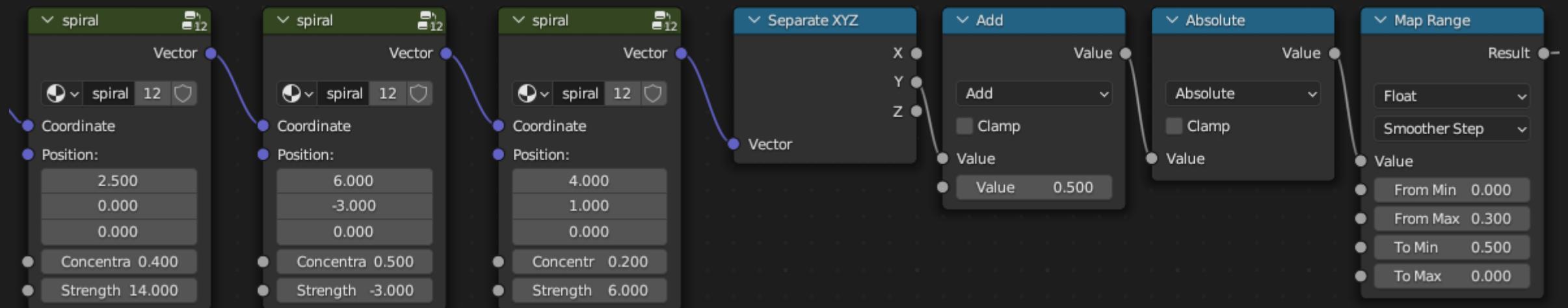
+10



예시



예시



023-024강 Advanced Procedural Texture – Vector Displacement

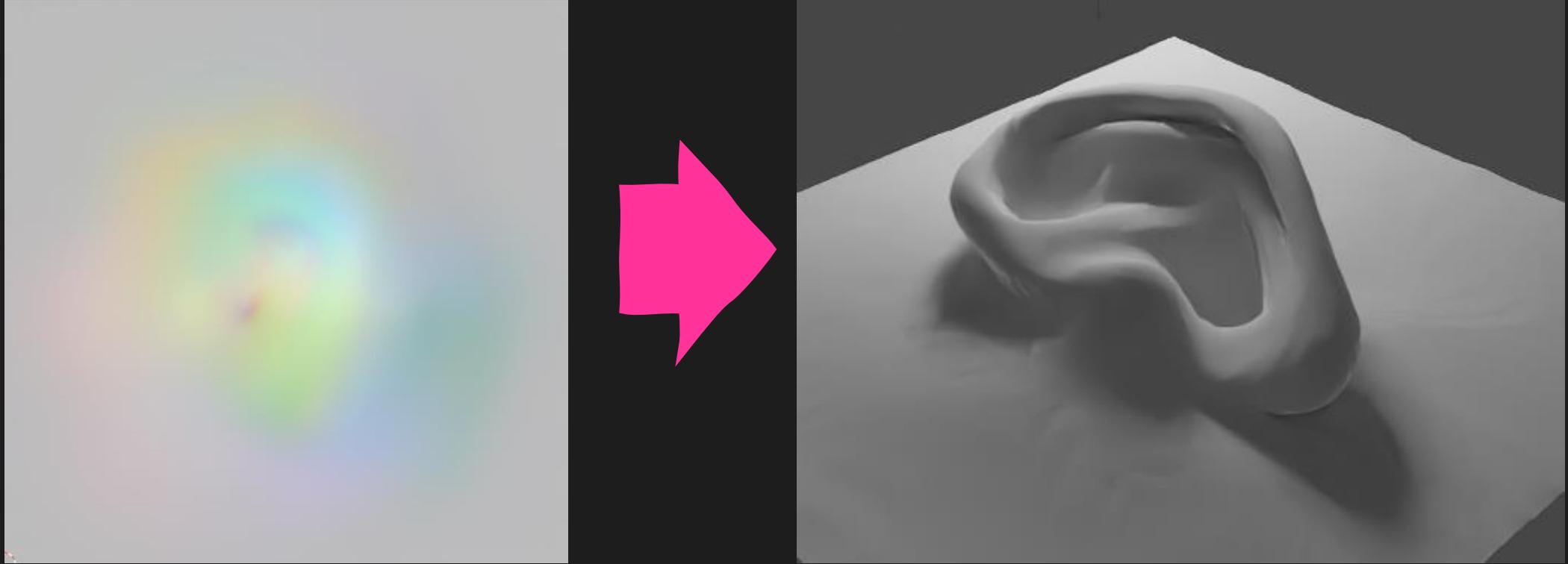


Vector Displacement

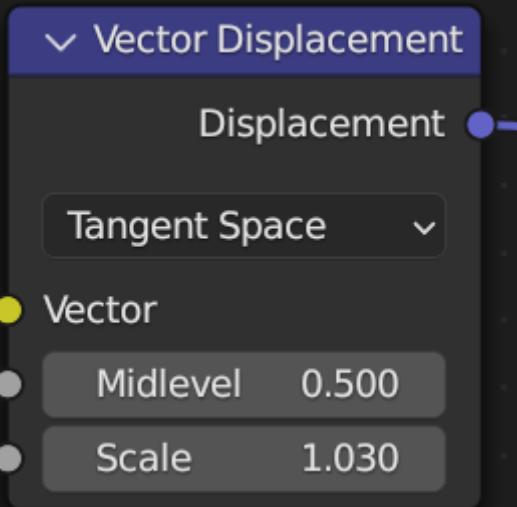
소개

Vector Displacement는 높이가 아닌 '방향을 통해 displacement를 만듭니다.

위아래 뿐만 아니라 여러 방향으로 이동할 수 있으므로, 일반적인 displacement에서 만들기 불가능한 표면의 겹침을 구현할 수 있습니다.

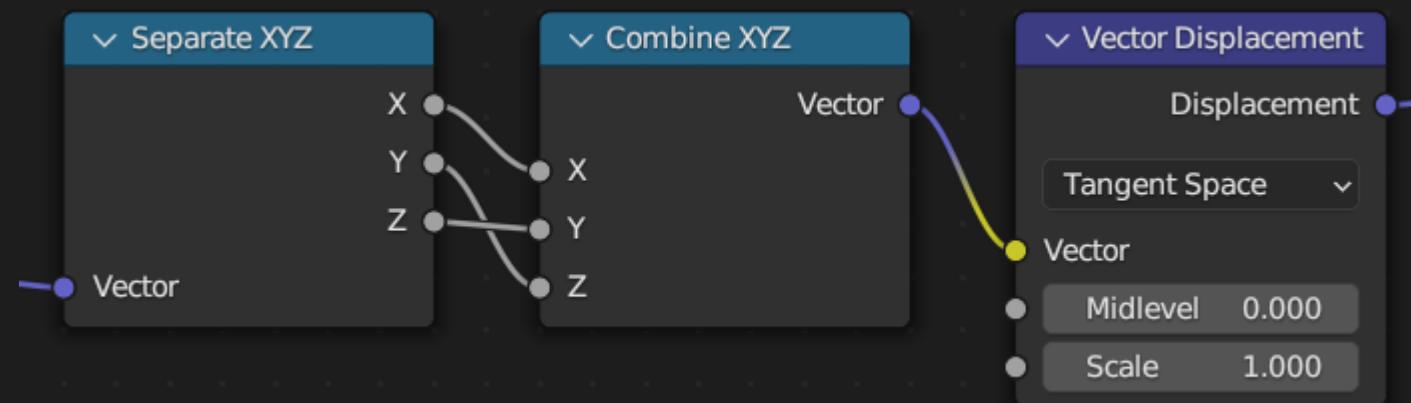


Vector Displacement node



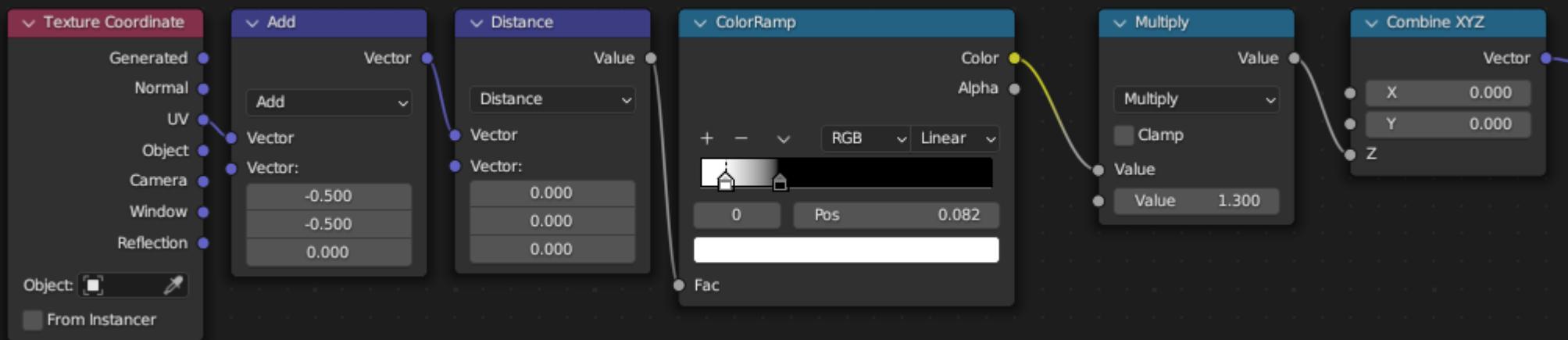
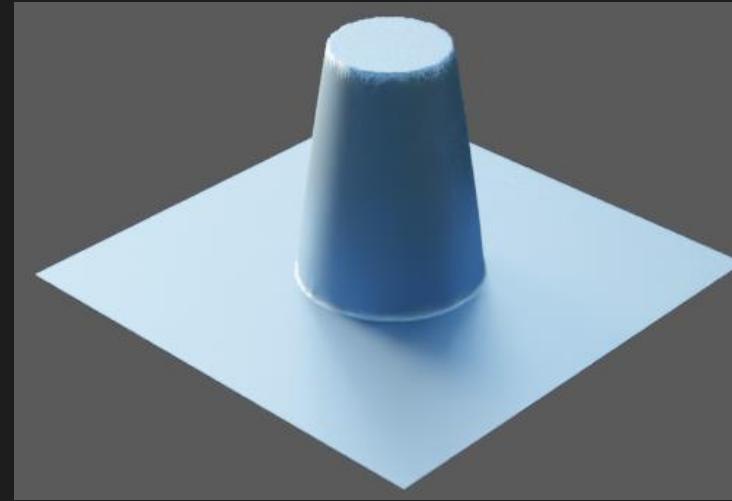
Tangent Space는 표면의 노멀 방향으로 왜곡을 일으킵니다.
Tangent Space에서, 표면이 향하는 바깥 방향은 y축입니다.

(일반적으로 이것을 신경쓸 일은 없지만, 우리는 이미지텍스쳐를 이용하는 것이 아니기 때문에 중요한 부분입니다)
아래와 같이 y,z축을 바꿔놓으면 조금 익숙해집니다.



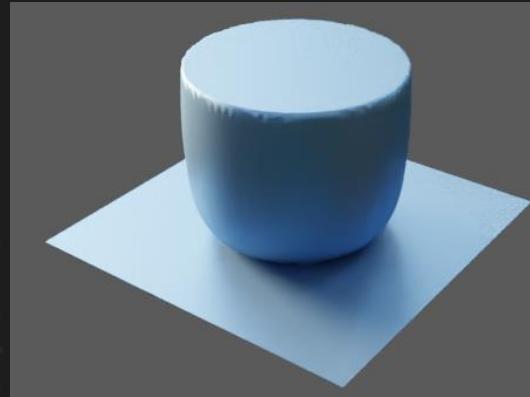
Vector Displacement

1단계 : 컵의 형태

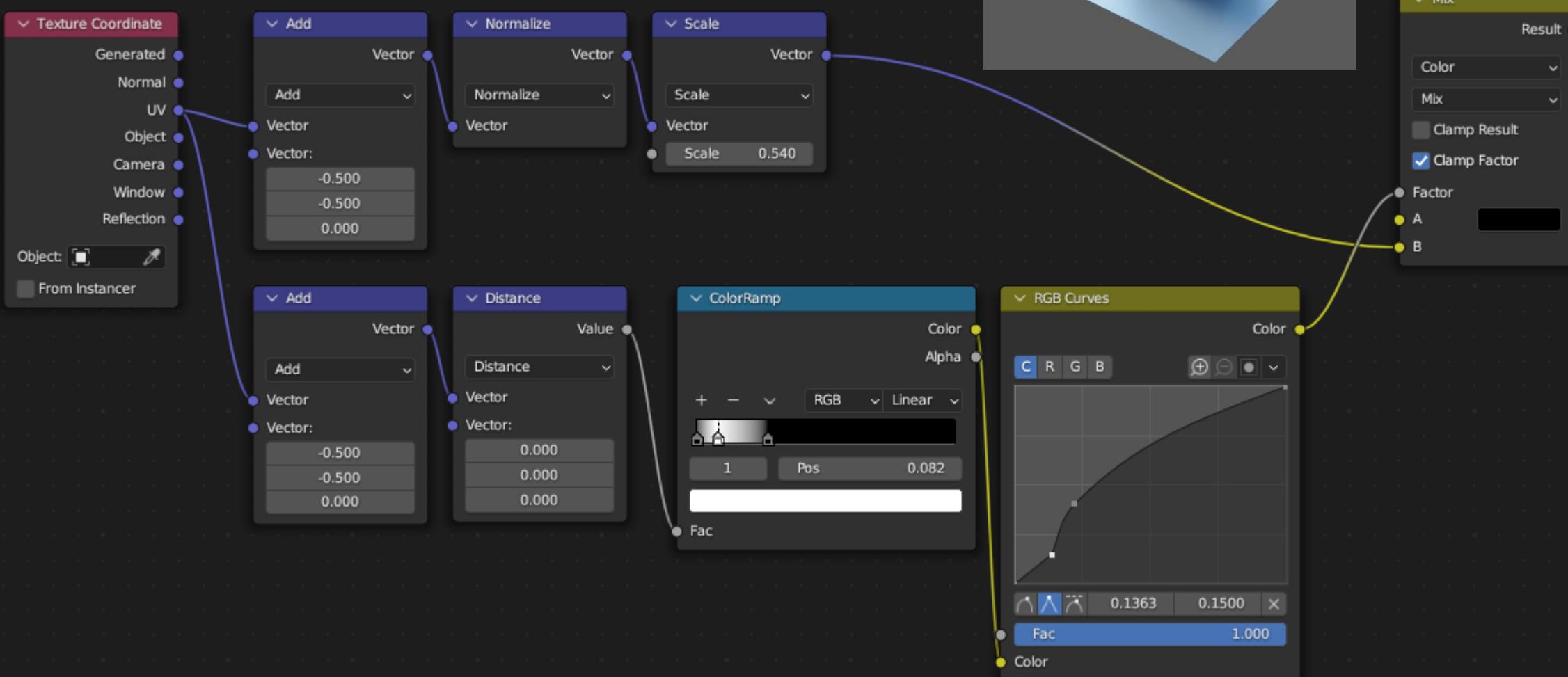


Vector Displacement

1단계 : 컵의 형태(2)



Vector Math (Add)로 연결

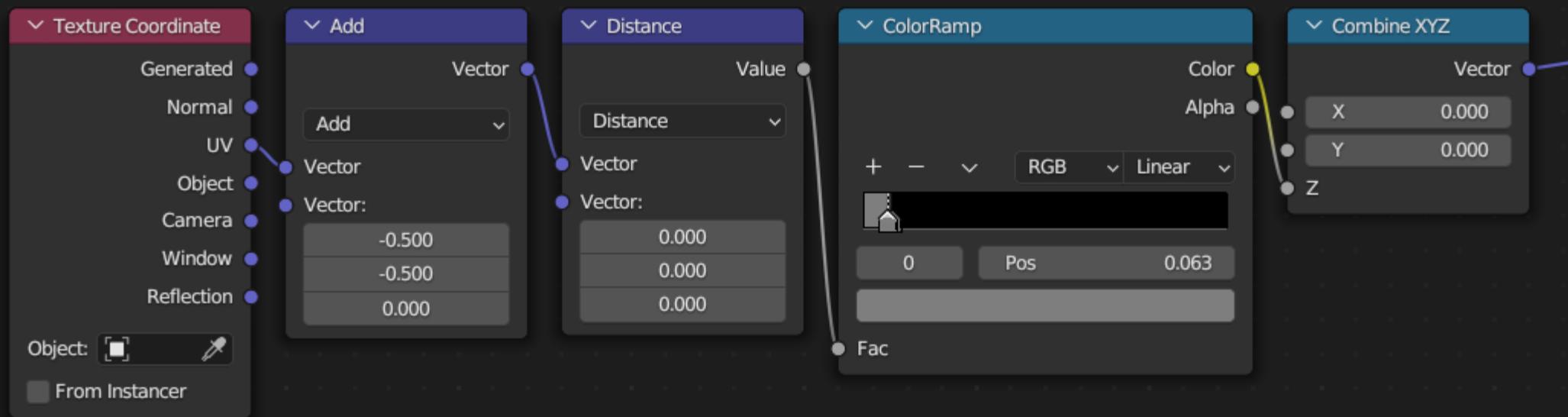


Vector Displacement

2단계 : 안으로 밀어넣기



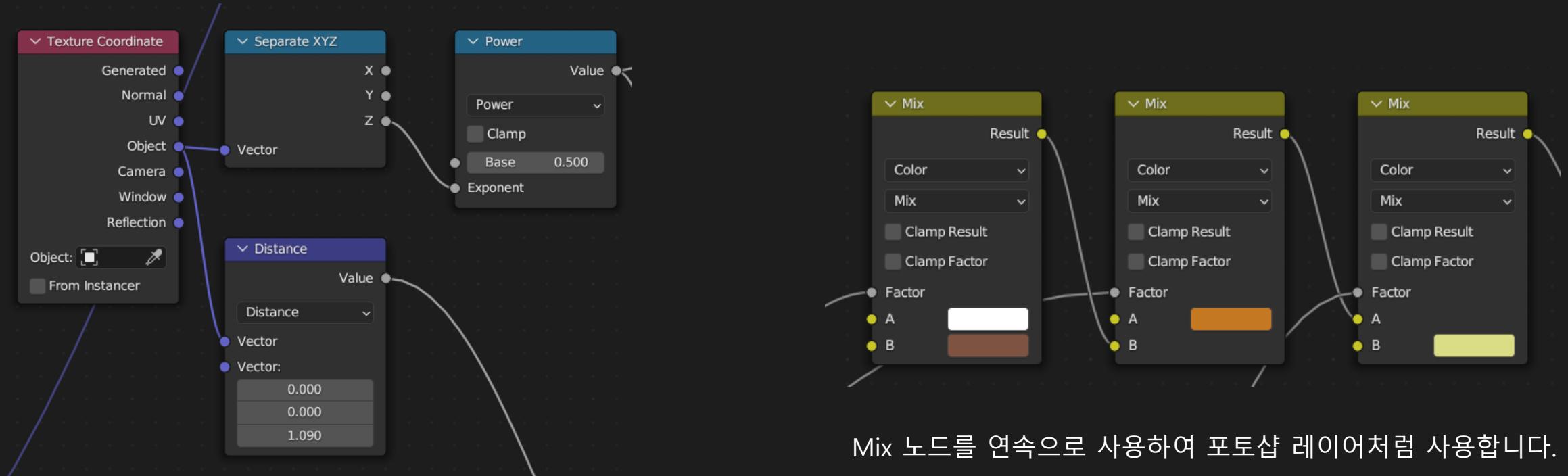
Vector Math (Subtract)로 연결



Vector Displacement

4단계 : 텍스쳐링

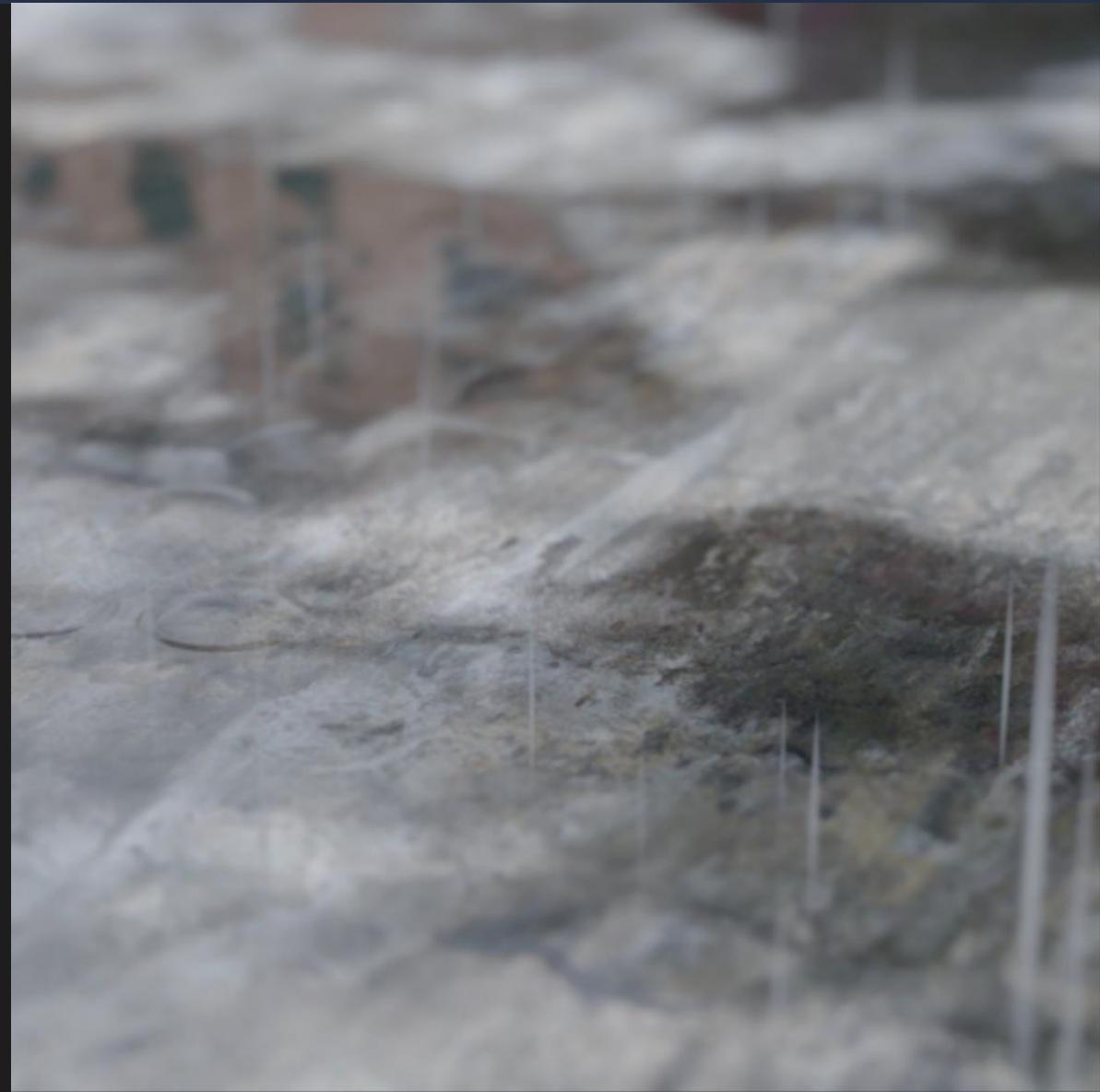
UV를 사용해도 되지만, Displacement는 물리적으로 메쉬를 뛰어나오게 하므로 Object 3차원 좌표를 사용할 수 있습니다.



025강 Procedural Texture – 셰이더 노드 애니메이션

좌표 이동을 이용한 애니메이션

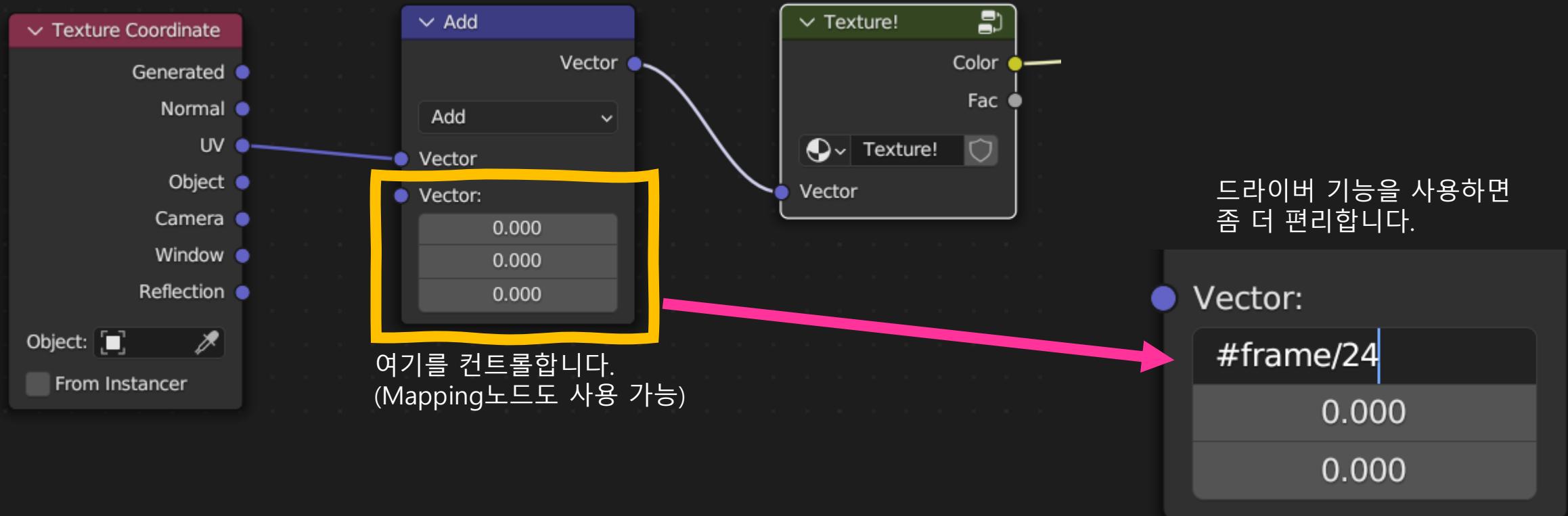
- 불
- 잔잔한 물
- 파도
- 고여있는 물
- 비와 빗방울



Procedural Texture Animation

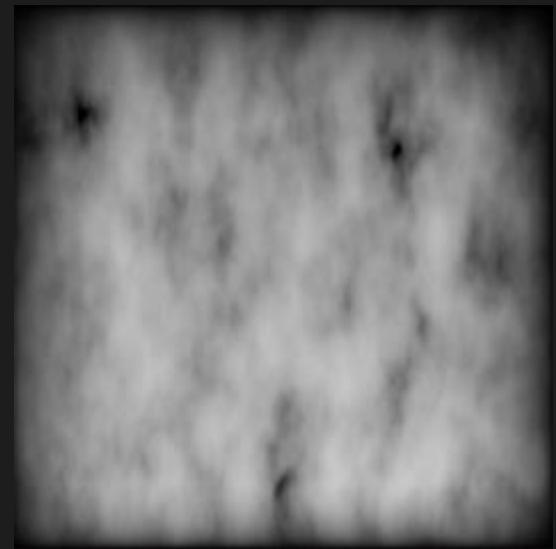
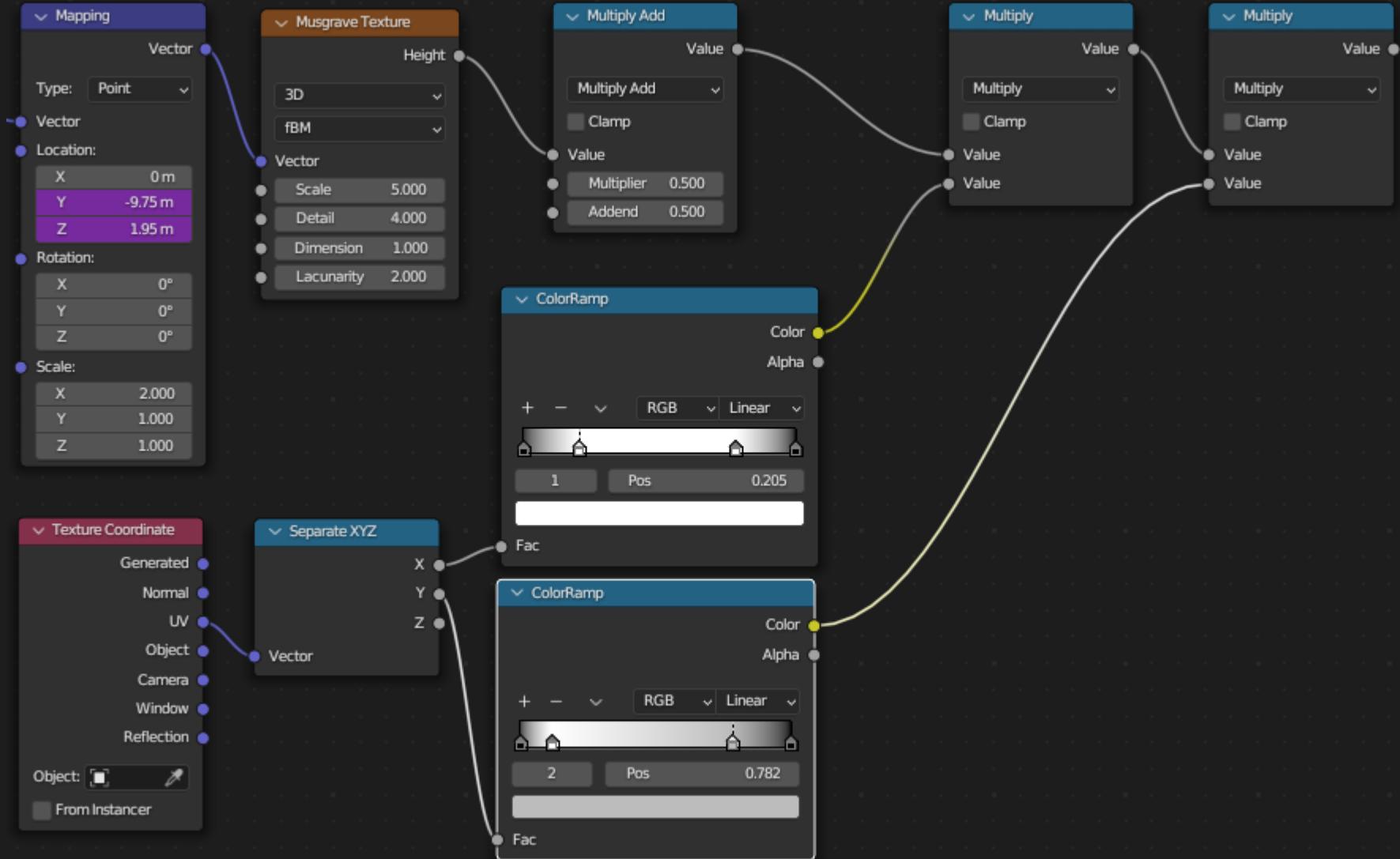
좌표의 이동을 통한 애니메이션

좌표를 프레임에 따라 이동시키면 애니메이션을 만들 수 있습니다.



물

간단한 노이즈로도 그럴듯한 효과를 만들 수 있습니다.



Colorramp 사용

잔잔한 물

Wave Texture를 사용하면 편리합니다.
물의 흐름이 전혀 없으면 부자연스럽습니다. 천천히라도 어느 한쪽으로 흐르는 방향을 지정해주는 편이 좋습니다.

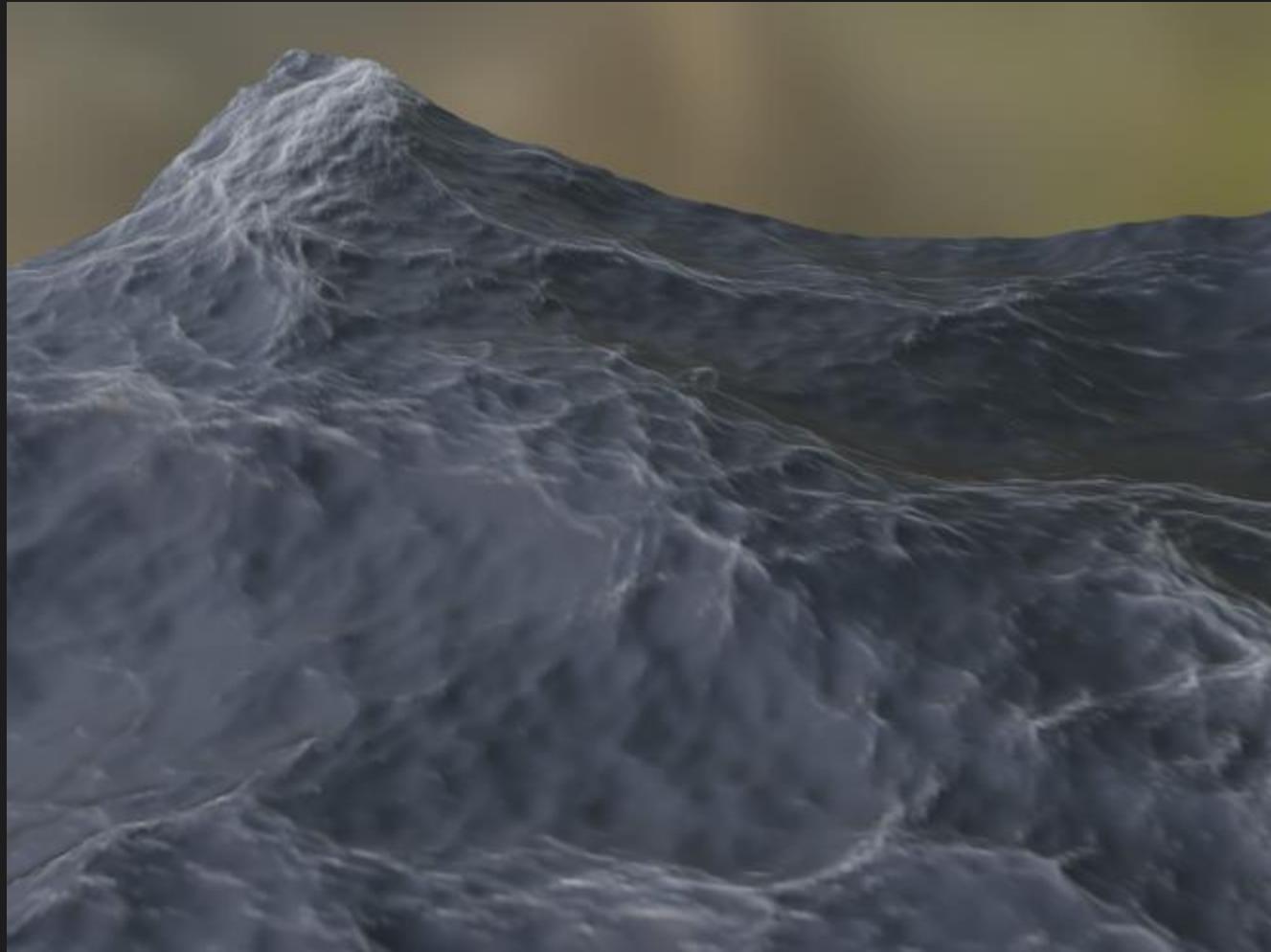


Transmission도 잘 작동합니다만, 물 아래쪽이 없다면 Metallic을 이용합니다.

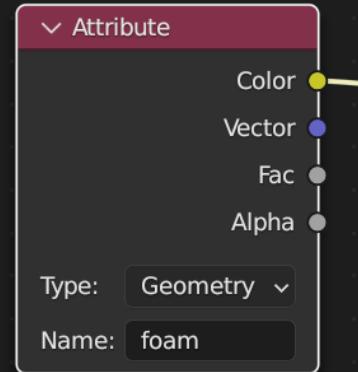
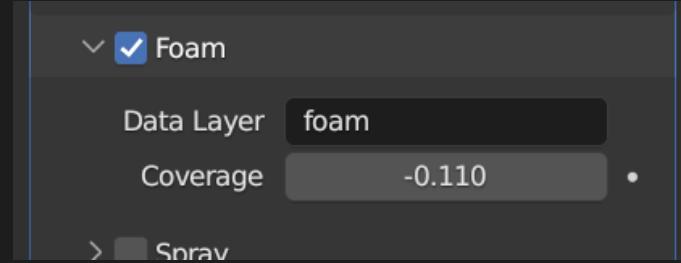
파도

Ocean 모디파이어를 사용합니다.

Ocean modifier는 파도치는 물결을 만들어냅니다.



Foam을 사용하면 거품 영역을 세이더에서 불러올 수 있습니다.



Foam은 로우폴리에서 계단현상이 발생합니다만, 폴리곤 수를 늘리지 말고 foam 출력에 노이즈를 섞어 사용하는 쪽이 최적화에 도움이 됩니다.

고여있는 물

믹스 노드의 응용



This screenshot shows the node setup for the initial water effect. It starts with a "Single Image" texture node (APC_0117_5c1...). Its "Color" output is connected to the "Color" input of a "ColorRamp" node. The "Fac" (Factor) input of the "ColorRamp" node is connected to the "Fac" (Factor) output of a "Noise Texture" node. The "Alpha" output of the "ColorRamp" node is connected to the "Color" input of a "Multiply" node. The "Alpha" input of the "Multiply" node is connected to the "Alpha" output of the "Single Image" texture node. The "Result" output of the "Multiply" node is the final output. A "Vector" input is also present but not connected.

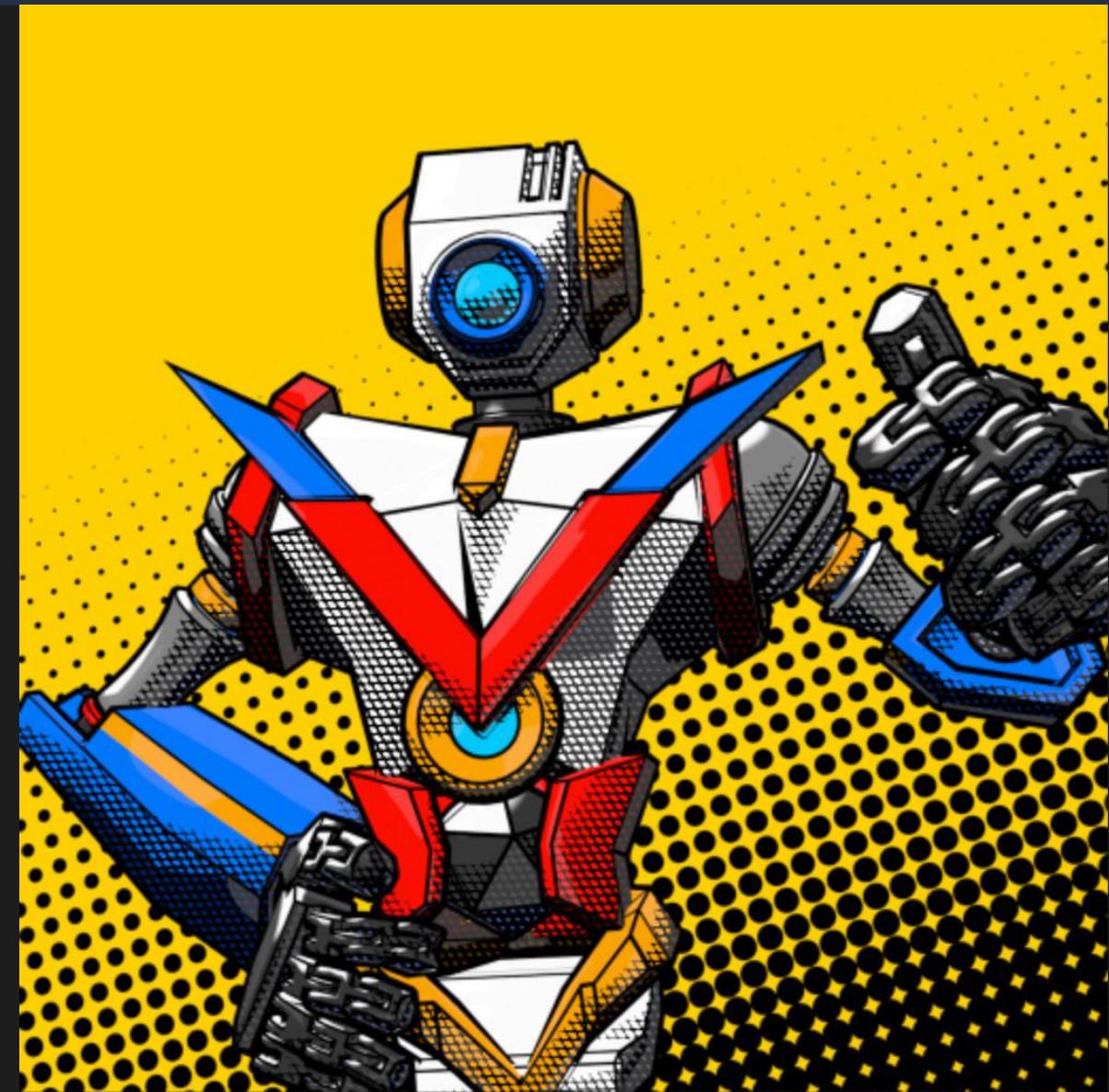


This screenshot shows the expanded node setup. It includes a "Linear Light" node with a "Noise Texture" node as its "Color" input. The "Fac" output of the "Noise Texture" node is connected to the "Fac" input of a "ColorRamp" node. The "Color" output of the "ColorRamp" node is connected to the "Color" input of a "Multiply" node. The "Alpha" output of the "ColorRamp" node is connected to the "Alpha" input of the "Multiply" node. The "Result" output of the "Multiply" node is the final output. A "Vector" input is also present but not connected. The "Linear Light" node is highlighted with a pink box.

026강 NPR : Non-Photorealistic Rendering

카툰 렌더링

Eevee에서 실사가 아닌 재질을 만들기



NPR : Non-Photorealistic Rendering

'Stylized'

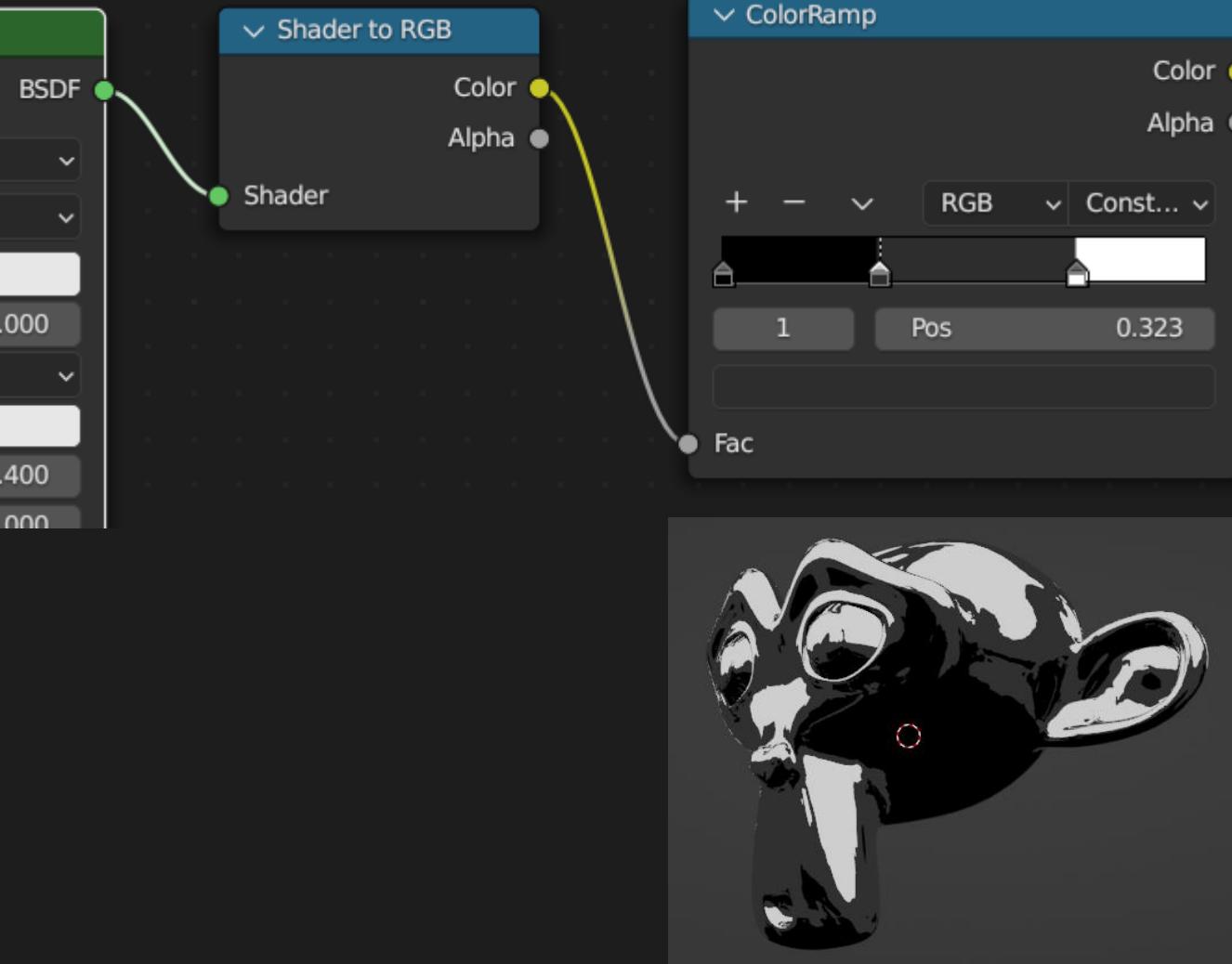
NPR은 카툰, 일러스트레이션과 같은 스타일을 지향하는 렌더링 방식입니다.

NPR은 그 특성상 추구하는 형태나 접근 방식이 무궁무진하므로 여기서 모두 다룰수는 없지만, 기본적인 접근법과 주로 사용하는 노드 중심으로 살펴보겠습니다.



셀 셰이딩과 음영 단계

효과적인 명도 표현을 위해서



가장 흔한 연결은 Shader to RGB – ColorRamp 조합입니다.

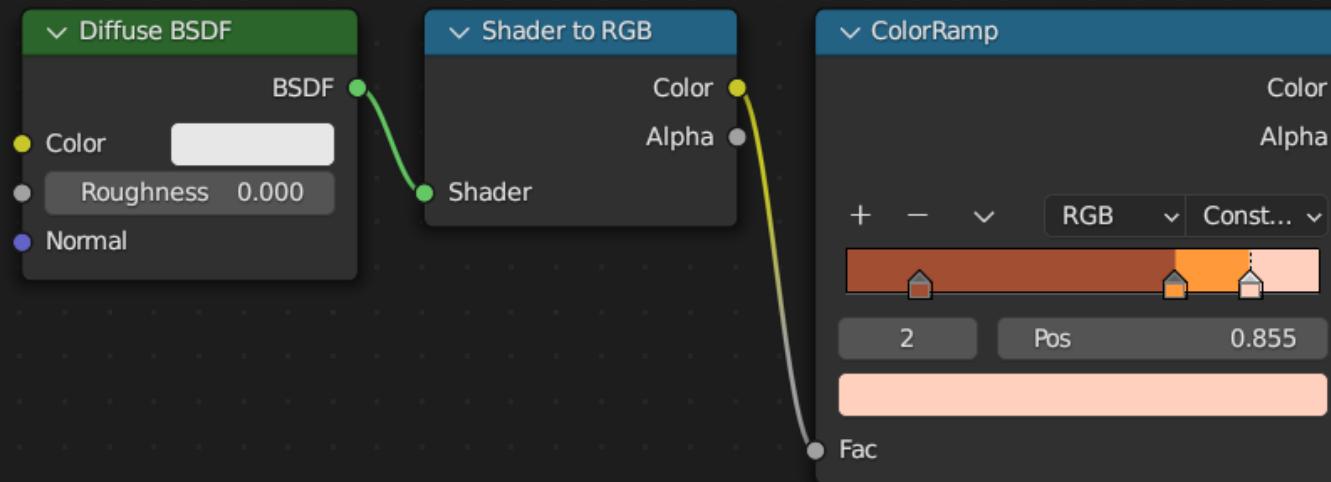
그런데 이 연결은, 경우에 따라 왼쪽처럼
울퉁불퉁하게 표현될 수 있습니다..

물론 이런 이미지를 의도할 수도 있겠습니다만

보통 셀 셰이딩은 경계가 물체의 형태를 따라 만들어질 때
더 효과적입니다.

셀 셰이딩과 음영 단계

효과적인 명도 표현을 위해서



ColorRamp를 통한 색 구분은 날카로운 편이 좋습니다.

점진적으로 부드럽게 증가하는 경우,
마치 금속과 같은 느낌으로 번들거립니다.

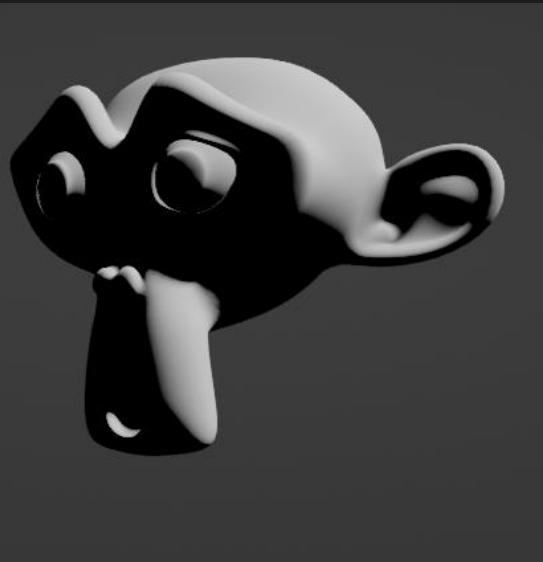
그러나 좌측처럼 단계를 단순화시키면 표면의 디테일이 제대로
드러나지 않아 답답한 느낌이 납니다.

NPR에서는 이런 부족한 디테일을 색 구분, 외곽선, 텍스쳐로
메꾸게 됩니다.



가짜 음영

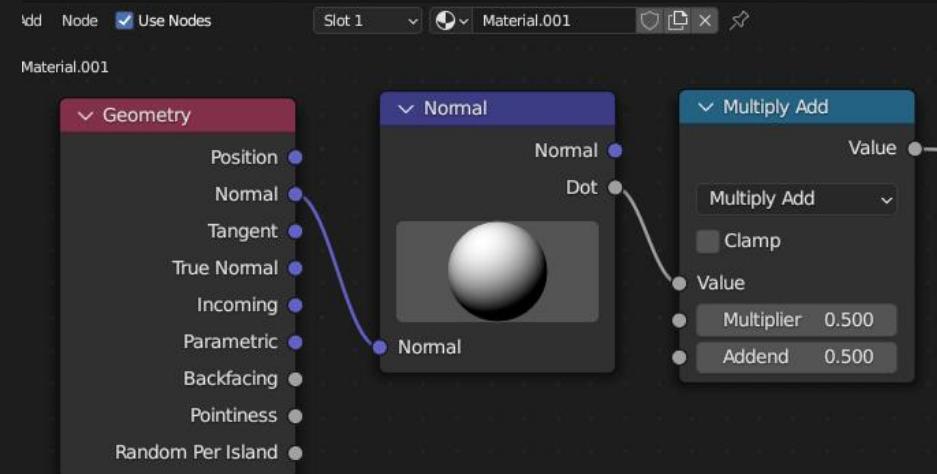
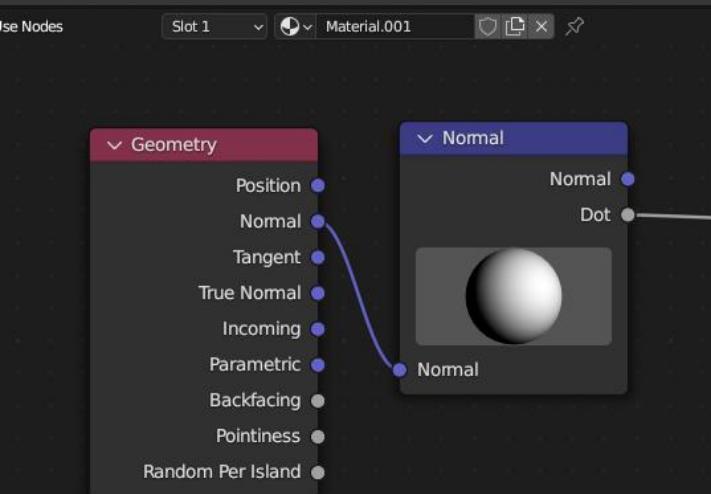
Normal node



라이팅 효과가 아닌 물체 표면의 각도만 필요하다면 13강에서처럼 Normal 노드를 이용할 수 있습니다.

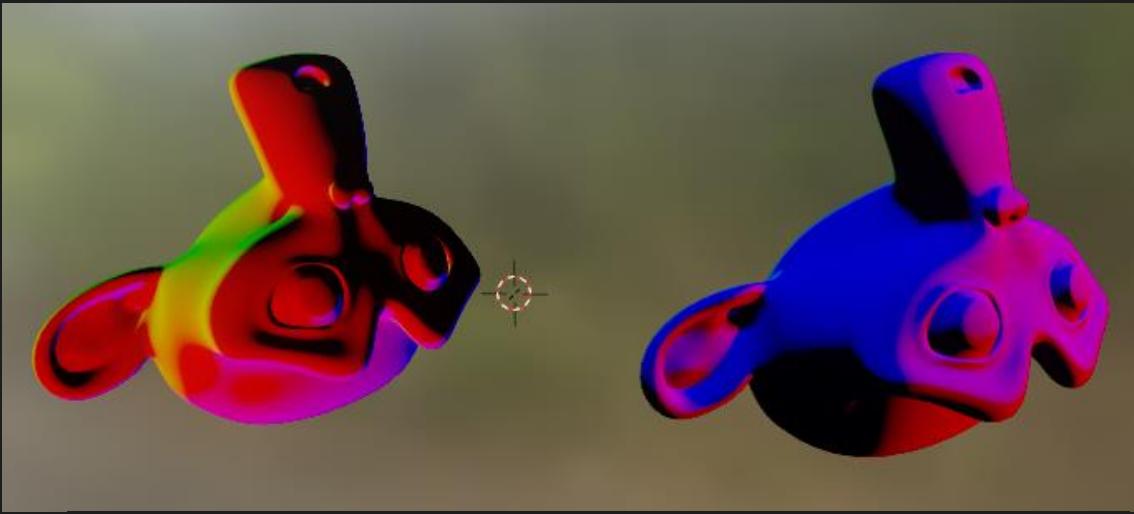
Dot 출력은 -1에서 1의 범위를 갖기 때문에, 우측처럼 범위를 0에서 1로 변경하면 표면 전체에 걸쳐 부드러운 명도 단계를 얻습니다.

이것을 [컬러램프](#)에 연결하여 모든 영역을 컨트롤할 수 있습니다.



여담입니다만 2000년대 초반 게임에서 사용되던 방식 중 하나입니다.

두 종류의 Normal



Texture Coordinate

- Generated
- Normal
- UV
- Object
- Camera
- Window
- Reflection

Object:

From Instancer

Geometry

- Position
- Normal
- Tangent
- True Normal
- Incoming
- Parametric
- Backfacing
- Pointiness

Random Per Island

Texture Coordinate의 Normal은 Object Space를 따라갑니다.
즉, 오브젝트가 회전했을 때 노멀도 같이 회전합니다.

한편, Geometry의 Normal은 World Space에 속하므로
오브젝트가 회전해도 노멀은 방향을 유지합니다.

Texture Coordinate의 Normal은 오브젝트를 따라
같이 움직이고 싶을 때 사용하고,

Geometry의 Normal은 오브젝트가 회전해도
계속 같은 방향을 가리키고 싶을 때 사용합니다.

※ Shader의 Normal은
World Space Normal입니다.

Glossy BSDF

BSDF

GGX

Color

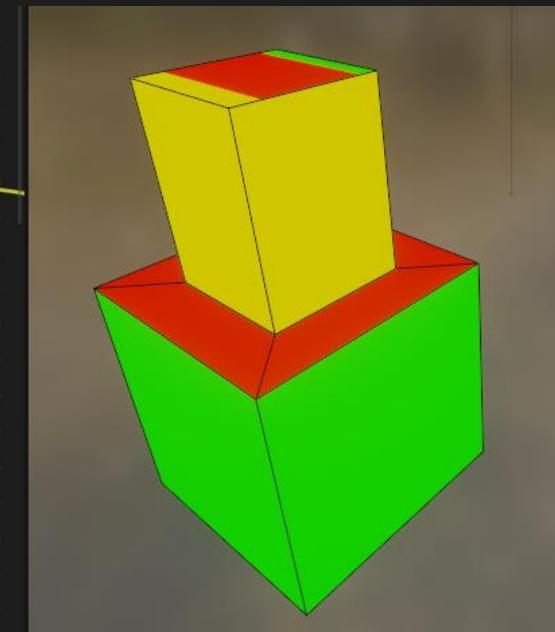
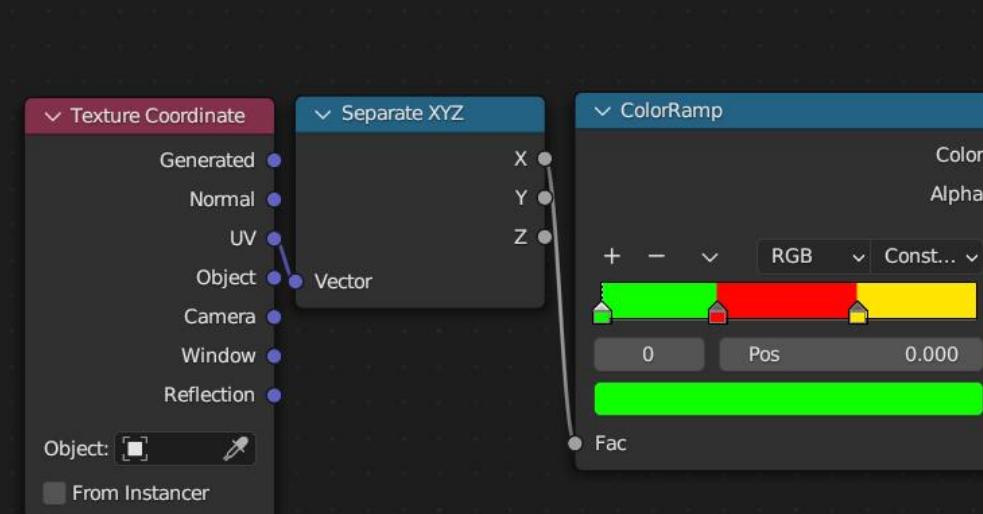
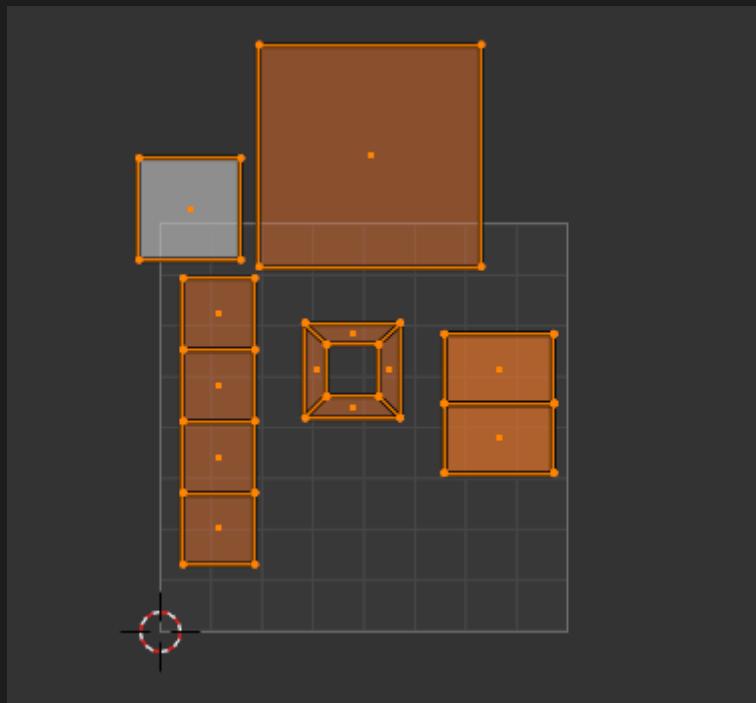
Roughness 0.500

Normal

색 구분

하나의 머티리얼에서 색을 구분시키기

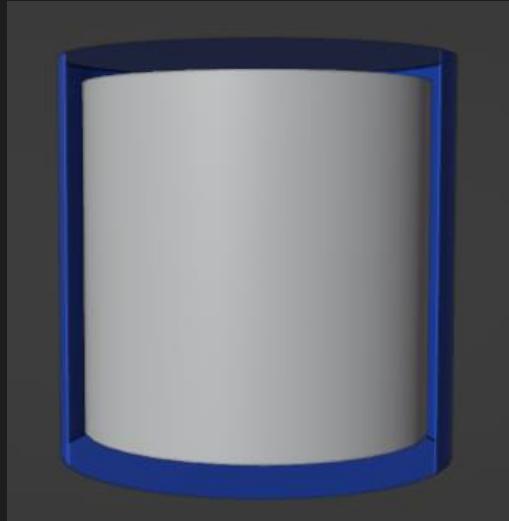
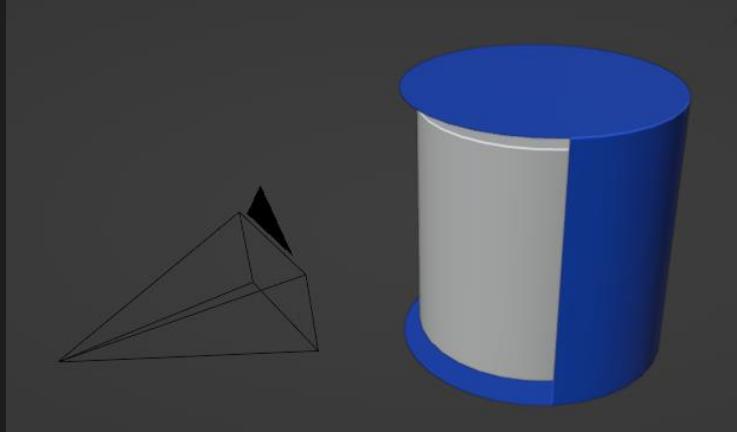
UV좌표를 이용하면 이미지없이 표면의 색을 여러 개로 구분할 수 있습니다.
아래와 같이 연결하시고 각 면의 UV위치를 조절하세요.



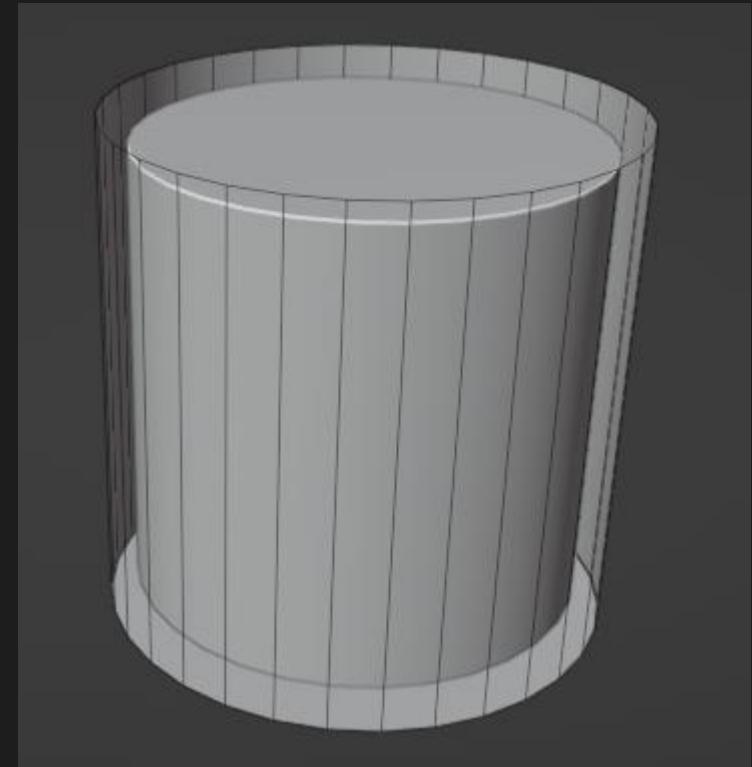
만약 UV를 이미 다른곳에서 사용하신다면 UV를 두개를 만드세요. (13강 참고)

외곽선(1) Backface Culling

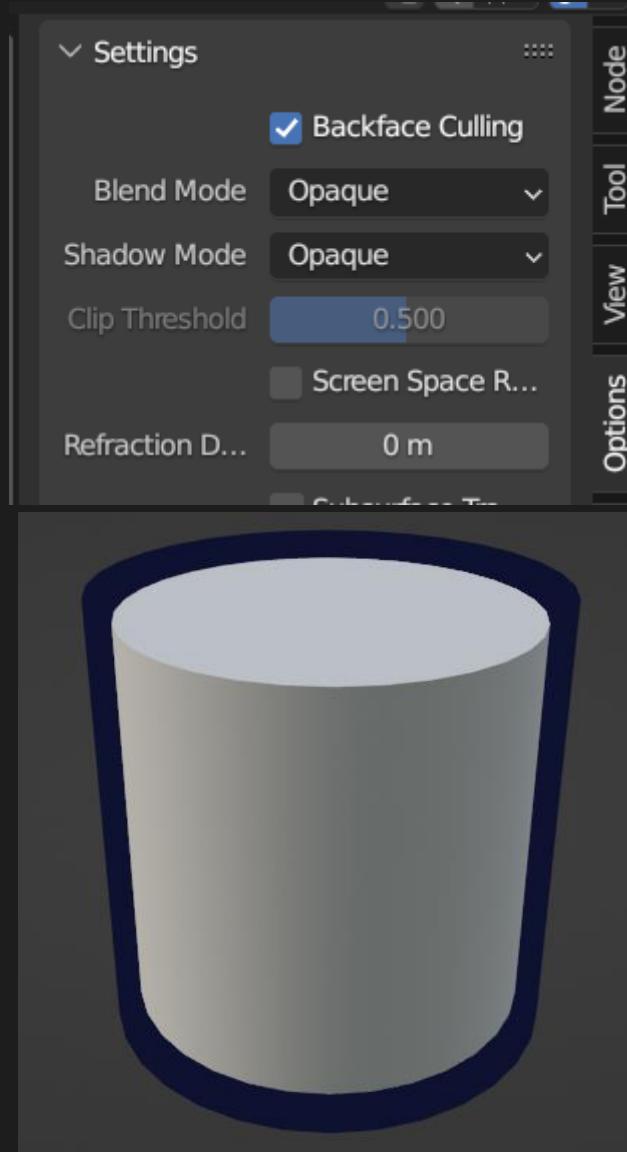
1. 만약 오브젝트를 감싸는 좀 더 큰 오브젝트를 만들어서,
카메라 방향으로 구멍을 뚫는다면 마치 외곽선처럼 보일 것입니다.



2. 한편, Backface Culling은 물체의 뒷면을 투명하게 만드는 기능입니다.
노멀 방향을 반대로 뒤집는다면, Backface Culling은 거꾸로 작용하여
물체의 앞면을 투명하게 만들어, 결과적으로 아래와 같이
물체의 뒷면만 보이게 만들 것입니다.
이것이 '자동으로 카메라 방향으로 구멍을 뚫어주는' 효과가 됩니다.

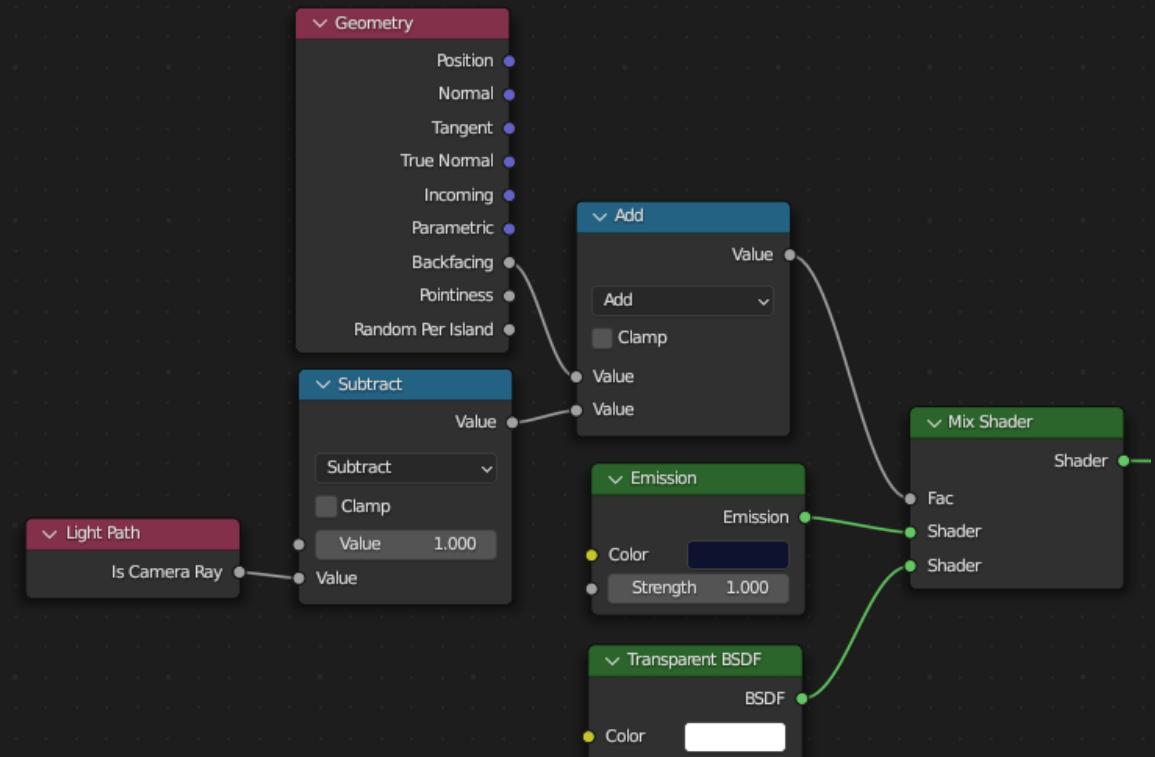


외곽선(1) Backface Culling

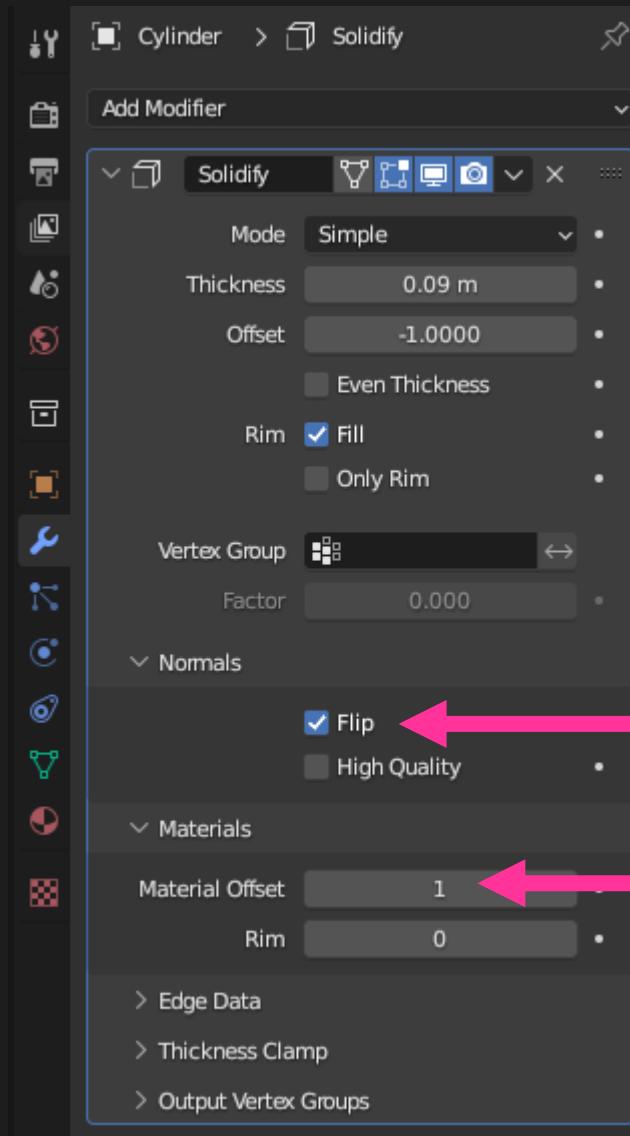


Eevee에서는 렌더 옵션에서 Backface Culling을 켜주면 작동합니다.

Cycles에서는 조금 복잡해집니다.. (자세한 사항은 28강 참고)
하지만 NPR은 보통 Cycles에서 구현하지 않습니다.



외곽선(1) Backface Culling



오브젝트를 감싼 외곽선이 **자동으로** 만들어지게 하려면 Solidify를 이용합니다.

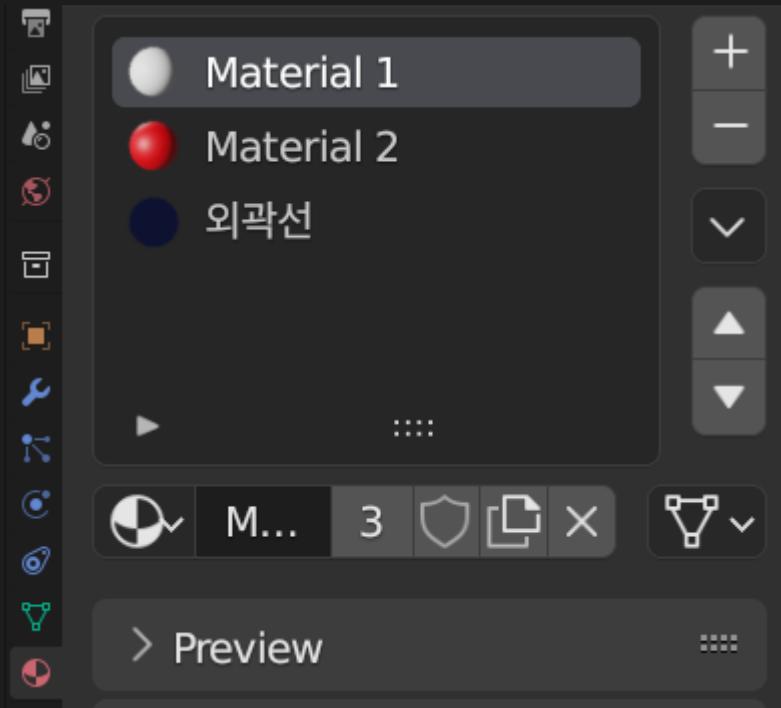


Flip으로 바깥의 면을 뒤집습니다.



Material Offset으로 Material을 선택합니다. (뒷장 참고)

외곽선(1) Backface Culling



Solidiy의 Material Offset은 조금 특이한 방식으로 재질을 선택합니다.

Offset이라는 이름처럼, 원래 재질에서 n번째 떨어진 재질을 선택합니다.

앞에서 Material Offset이 1이었으므로,

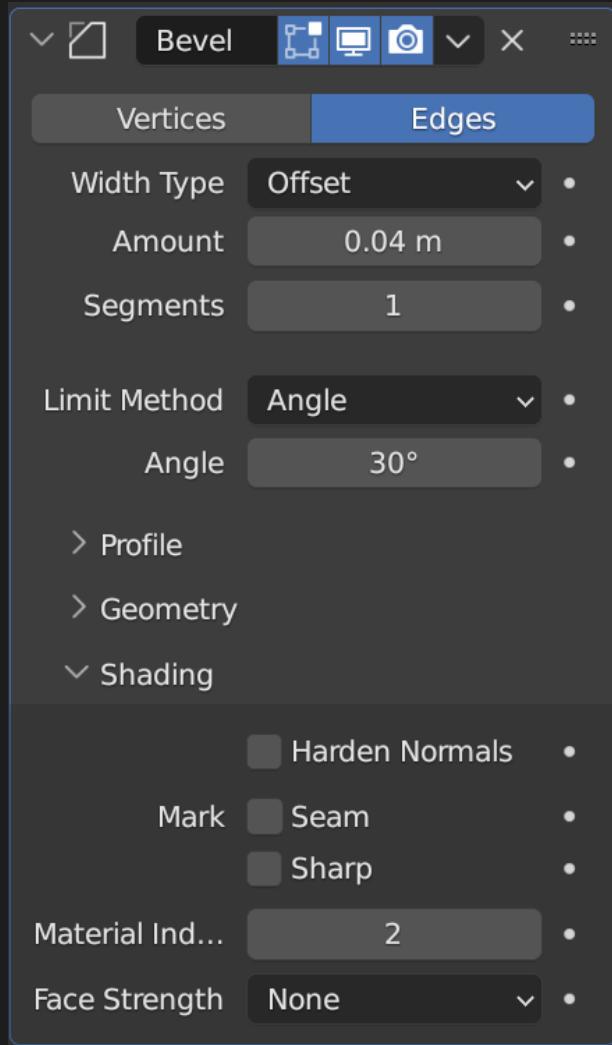
Material 1이었던 재질은 한칸 아래인 **Material 2**,

Material 2였던 재질은 그 한칸 아래인 **외곽선**이 됩니다.

혼란을 막기 위해 **외곽선** 재질을 가장 마지막에 놓고,

Material offset을 매우 큰 숫자 (예컨대 1000) 으로 두면 반드시 맨 아래 재질이 선택되므로 편리합니다.

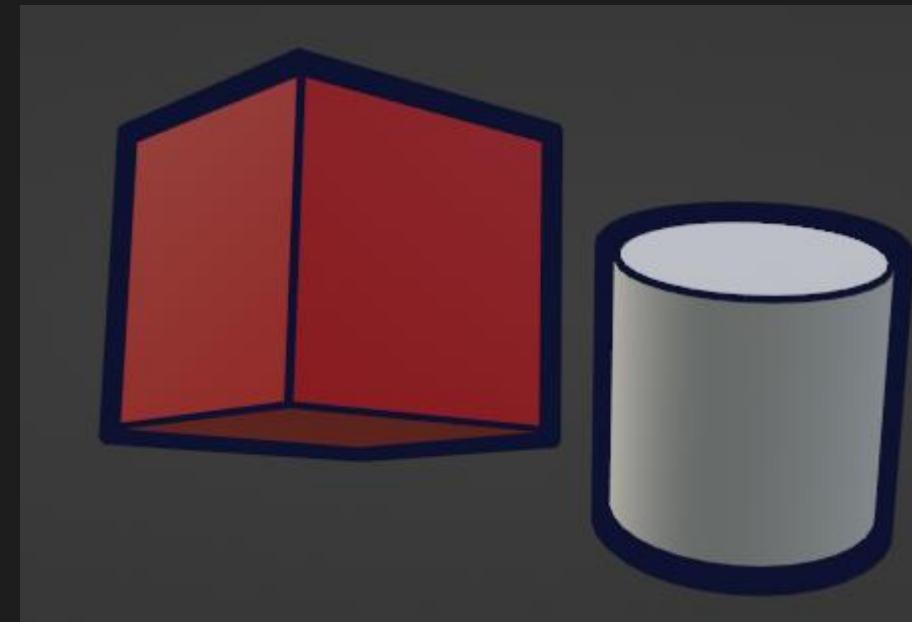
외곽선(2) Bevel



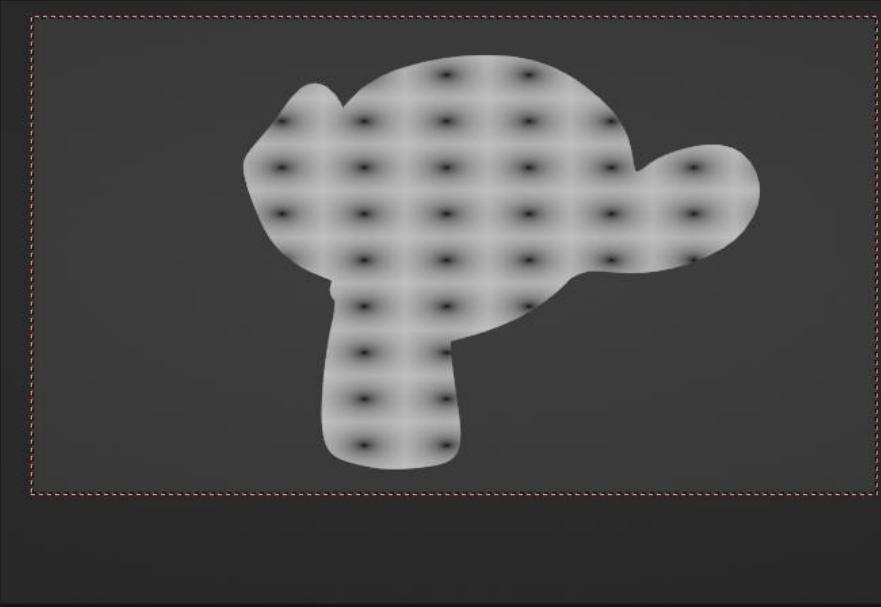
Bevel 모디파이어를 사용하여 추가적으로 외곽선을 만들 수 있습니다.

Bevel은 Solidify와 달리 Material 'index'를 사용합니다. 이는 Material 슬롯의 재질을 순서대로 불러옵니다.

방식은 다르지만 이 역시 매우 큰 값으로 두면 반드시 마지막 재질을 선택하므로, 이렇게 Solidify와 같은 재질을 선택하게 만들 수 있습니다.



Window 좌표



Window 좌표는 현재 보고 있는 화면을 나타냅니다.

카메라 모드에서 카메라 왼쪽 아래부터 오른쪽 위 끝까지를 x,y축에 대응시킵니다.

뷰포트 상태라면 '현재 뷰포트의 끝부터 끝까지' 가 좌표가 됩니다.

(뷰포트 크기에 따라 좌표가 바뀝니다!)

혼란을 막기 위해 window좌표를 쓸 때는 가급적 카메라 모드를 사용하며, 해상도 비율에 따라 미리 스케일을 조절해 둡니다.

가짜 광택

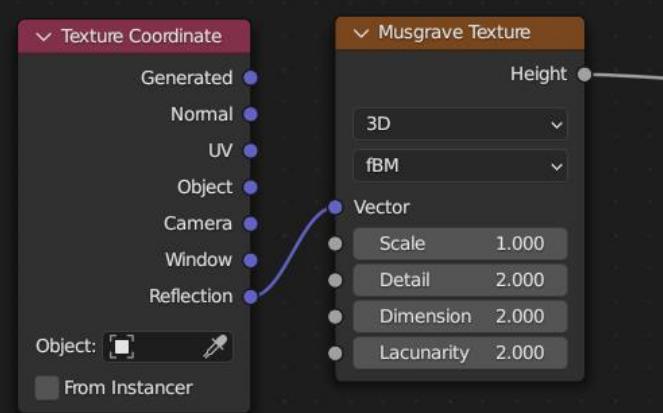
Reflection 좌표



Reflection 좌표는 화면이 움직임에 따라 변화하는 반사를 나타낼 수 있습니다.

Reflection좌표의 원리는 '배경 좌표를 반사한 것' 이므로, 만약 이미지 텍스쳐를 꽂는다면 배경으로 쓰일 수 있는 이미지여야 합니다.

즉 Hdri 같은 배경 텍스쳐를 Environment Texture 노드에 꽂아 사용합니다.



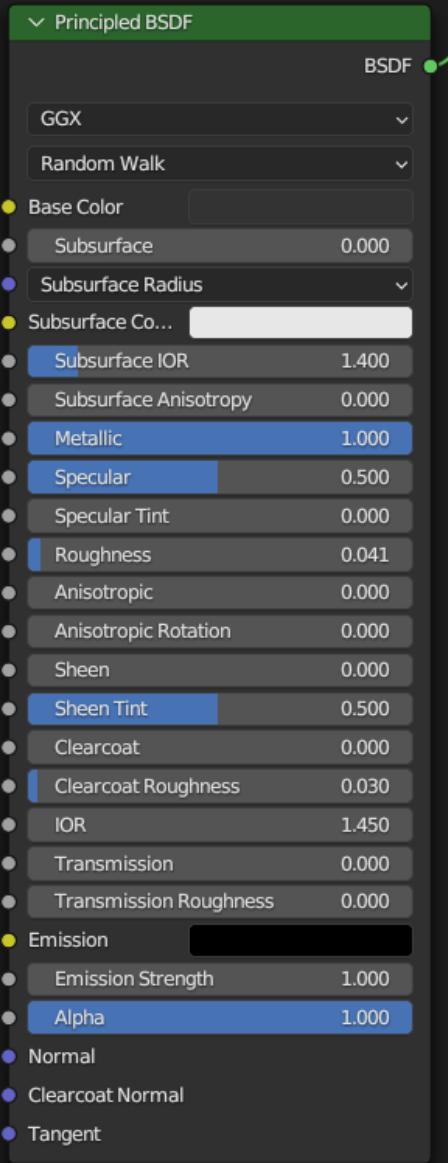
027강 특수한 재질

Principled BSDF로 만들 수 없는 특수한 재질들

- 형광/홀로그램
- 반투명 플라스틱

반사와 굴절을 분리해서 컨트롤하는 법





표현하기 힘든 재질들

일상 생활에서 흔히 보이는 재질과 컴퓨터로 만들기 쉬운 재질은 서로 다릅니다.

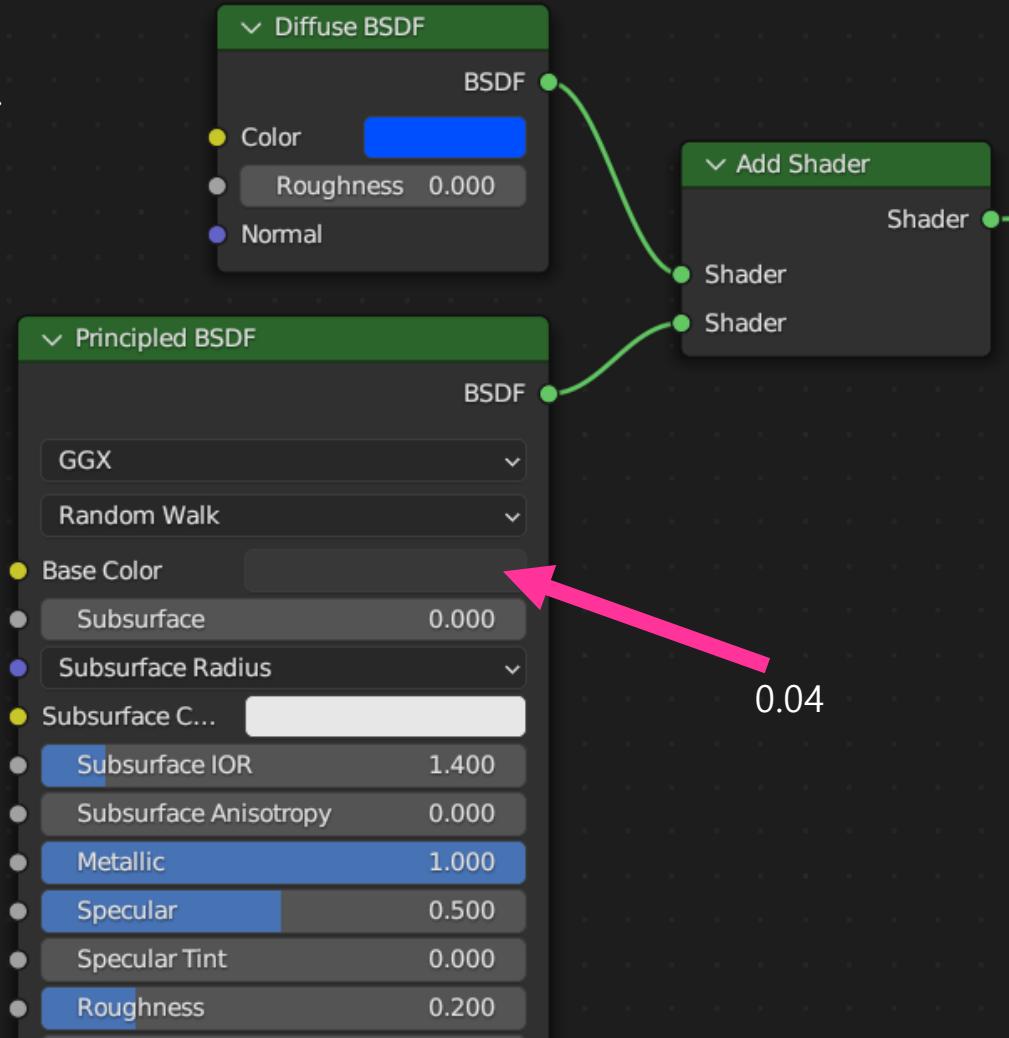
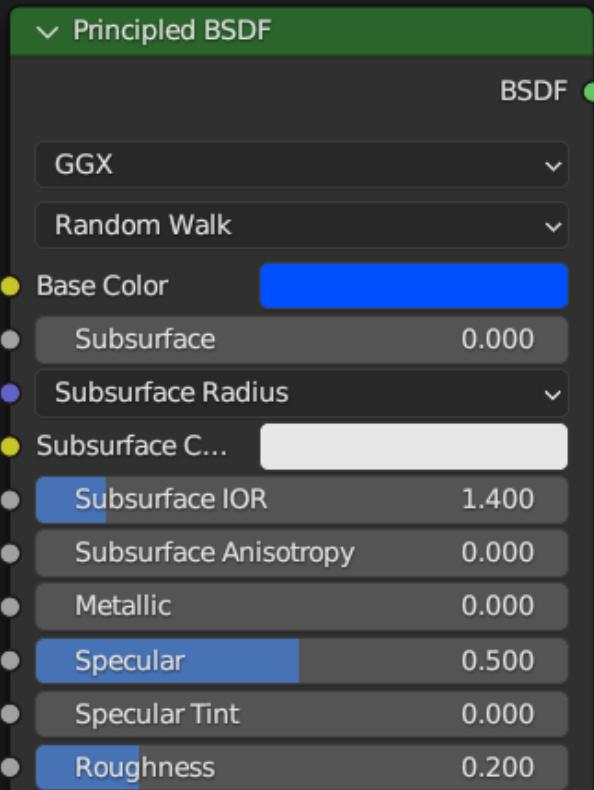
- 형광?
- 홀로그램?
 - 물체 색과 반사색의 분리
 - 각도에 따라 바뀌는 색
- 반투명 플라스틱?



구성요소 분리 (1)

불투명 재질의 구조

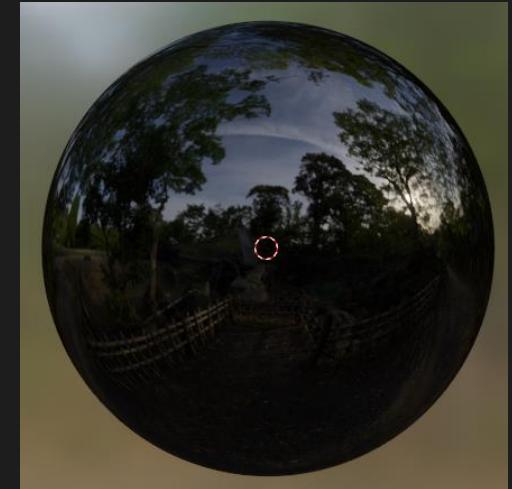
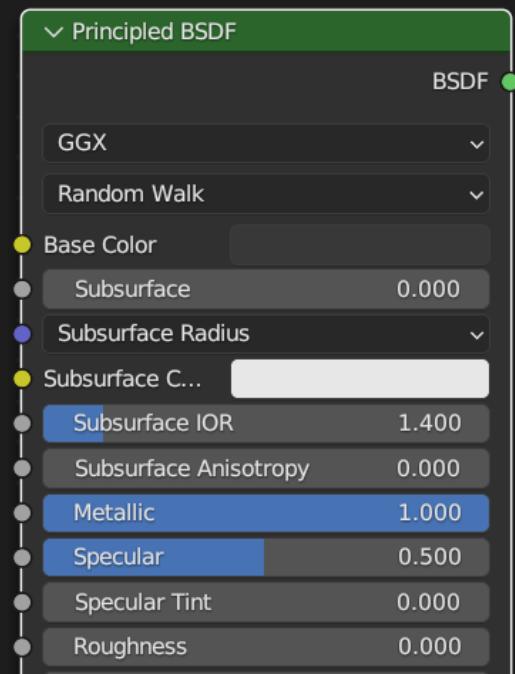
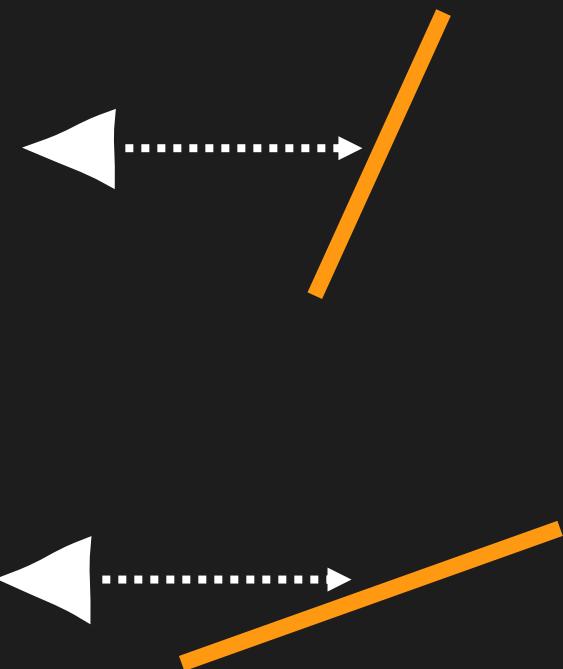
물체의 광택은, 기본색에
순수한 반사인 Metallic을 더한 것과 같습니다.



구성요소 분리 (1)

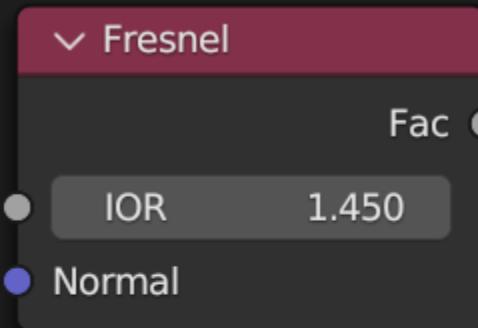
반사의 구조

Metallic은 모든 표면이 동일하게 반사되지 않습니다.
가장자리로 갈수록 더 반사도가 높아집니다.
정확히는, 면을 바라보는 각도에 따라 반사도가 달라집니다.



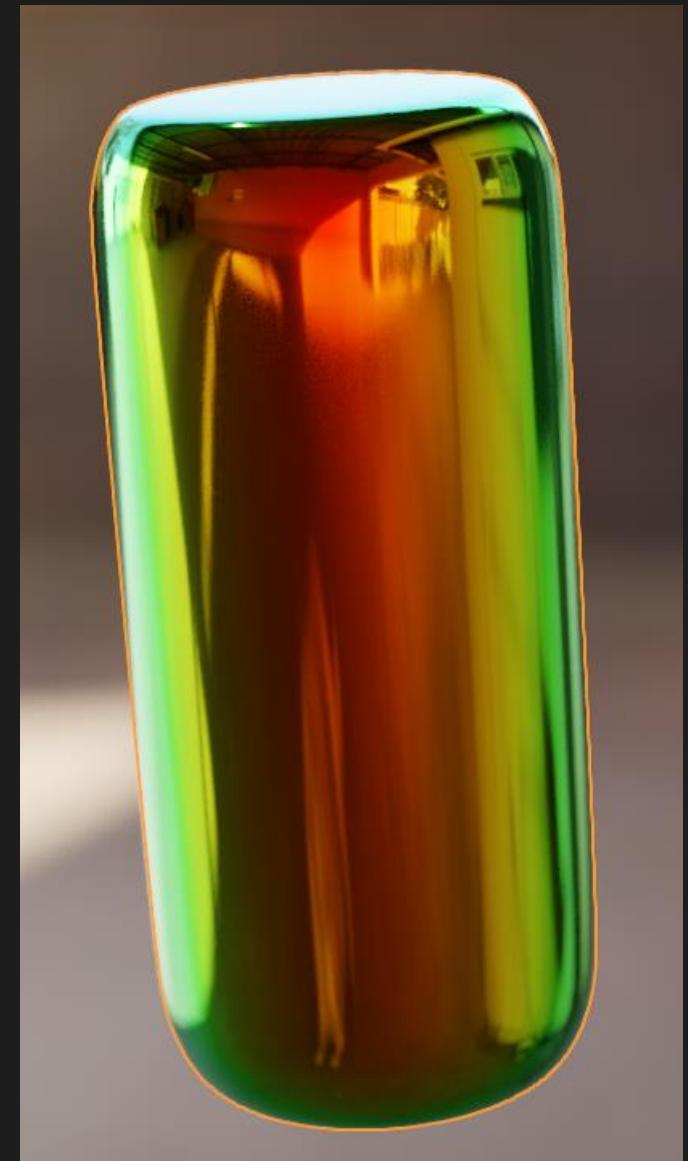
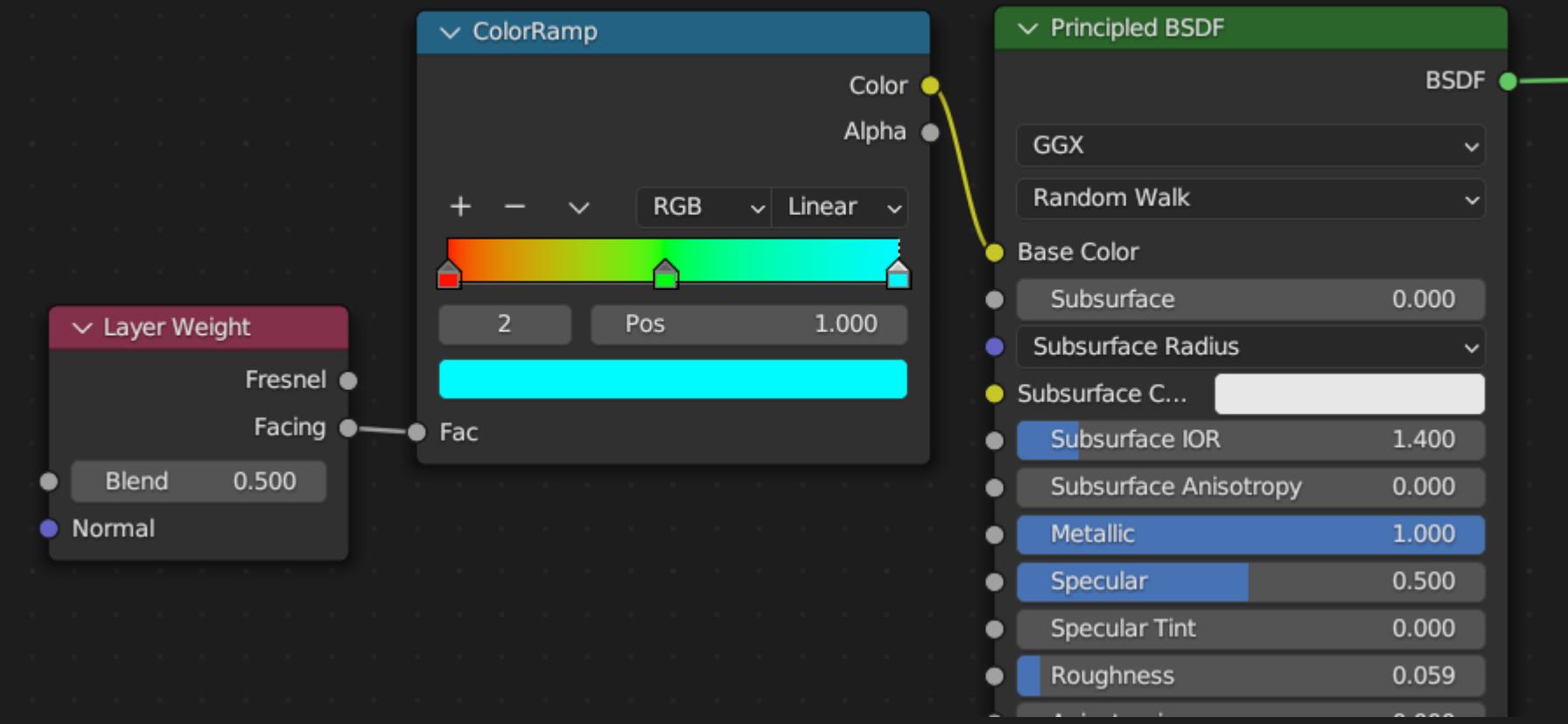
Fresnel / Layer Weight

이것을 만들어주는 노드가 Fresnel과 Layer Weight 입니다.

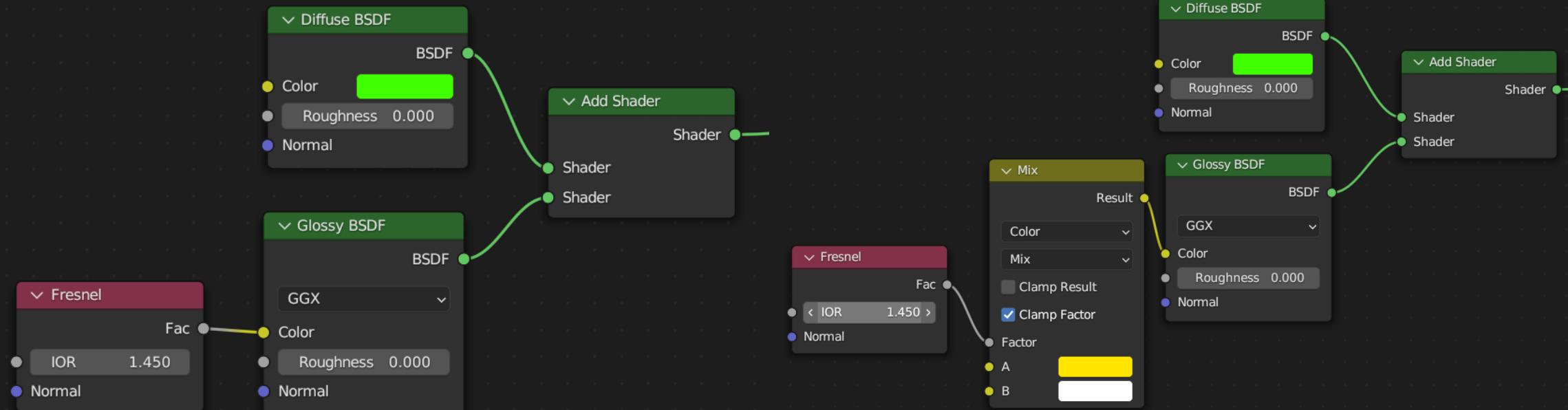


홀로그램

색이 바뀌는 재질

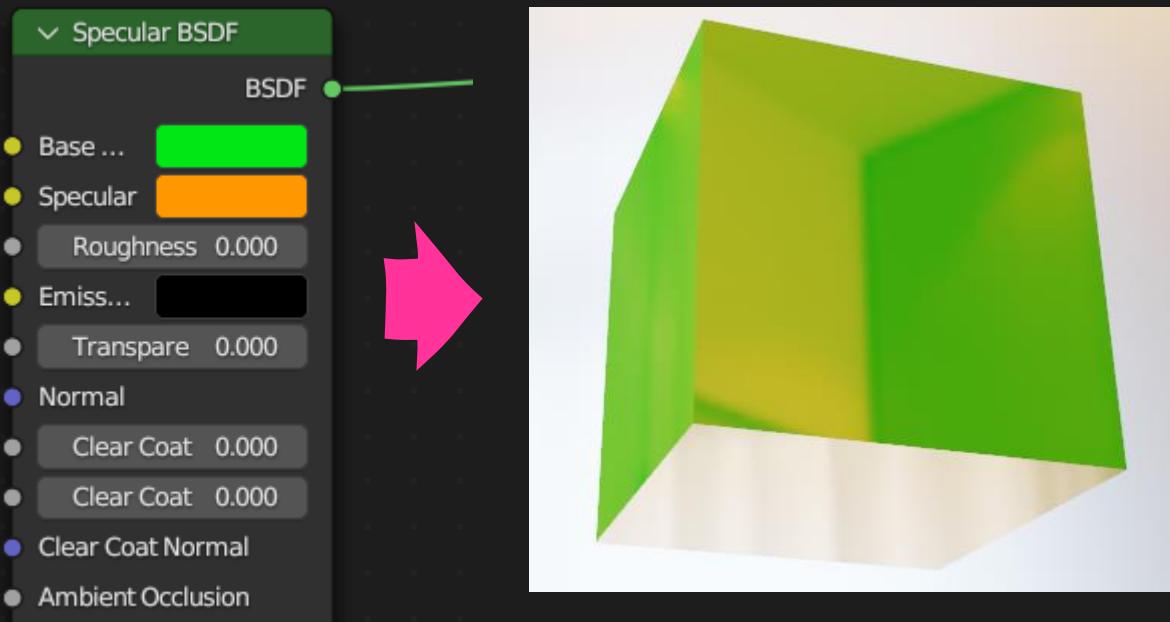


구성요소 분리 (1)



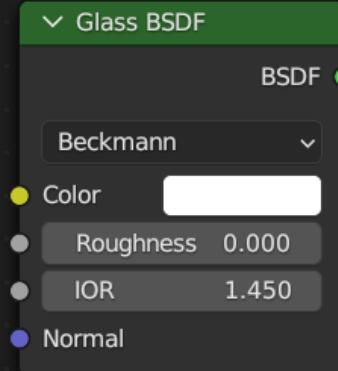
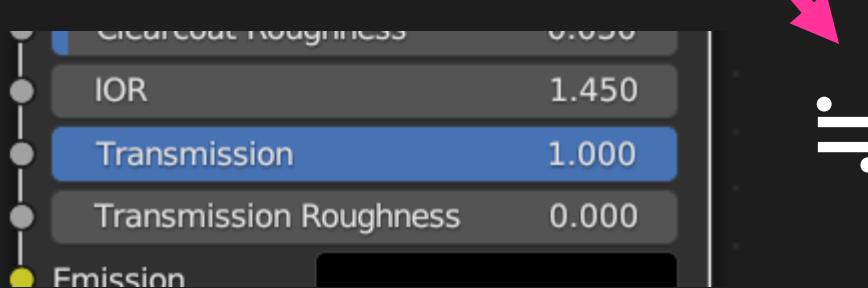
Specular BSDF

Specular BSDF를 사용하면 반사색을 간단하게 분리할 수 있습니다만, Eevee에서만 작동합니다.

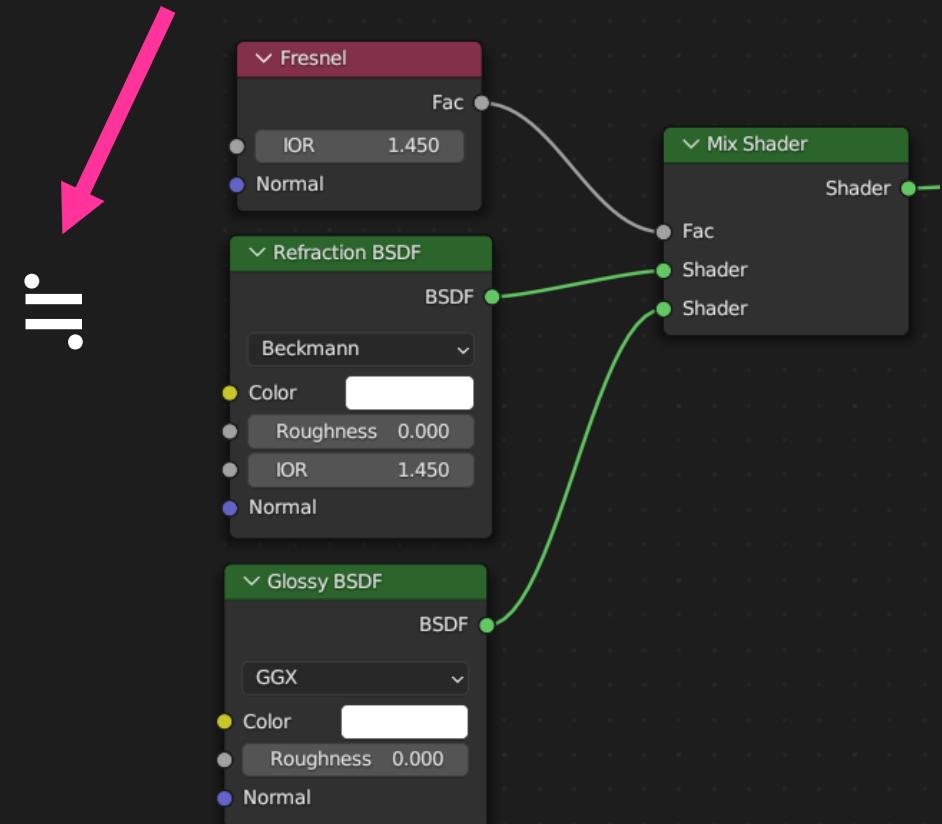


구성요소 분리 (2)

Principled BSDF에서 Trasmission을 올리면 유리 재질이 됩니다.
이것은 Glass BSDF와 같습니다.

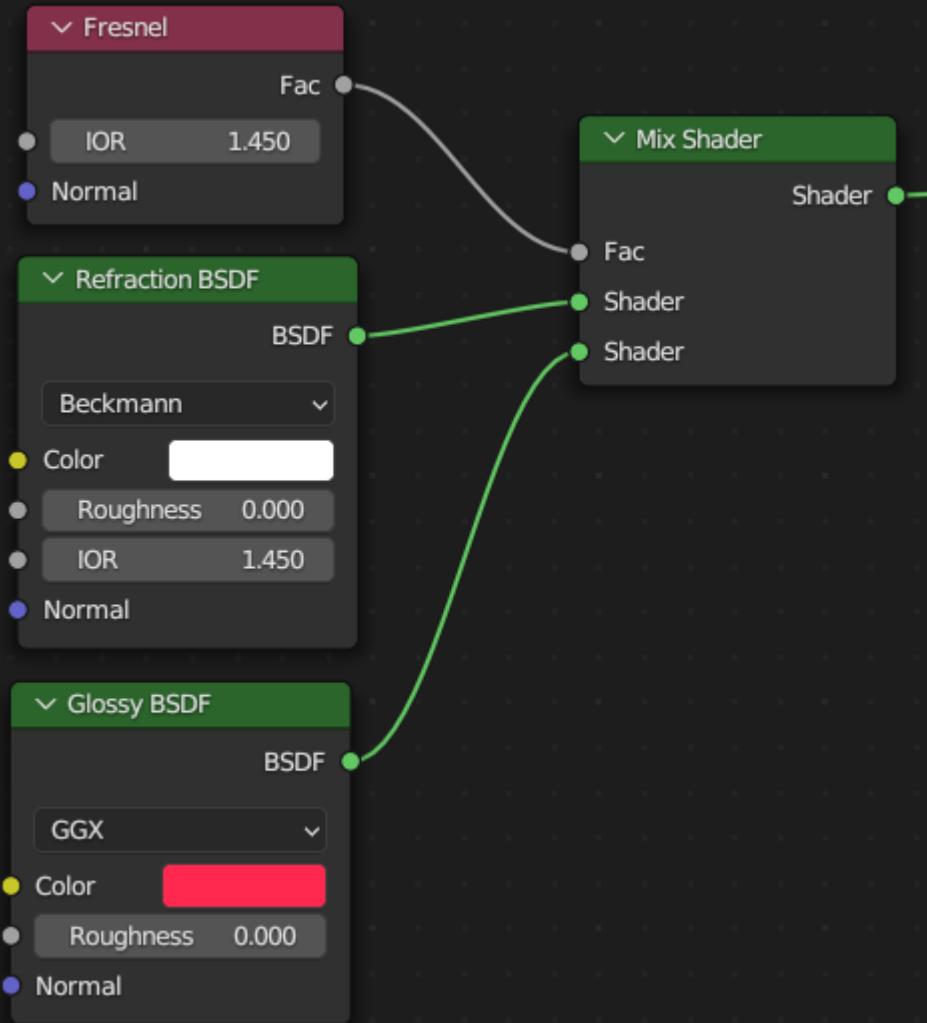


유리는 Glossy (반사) 와 Refraction (굴절)의 결합입니다.



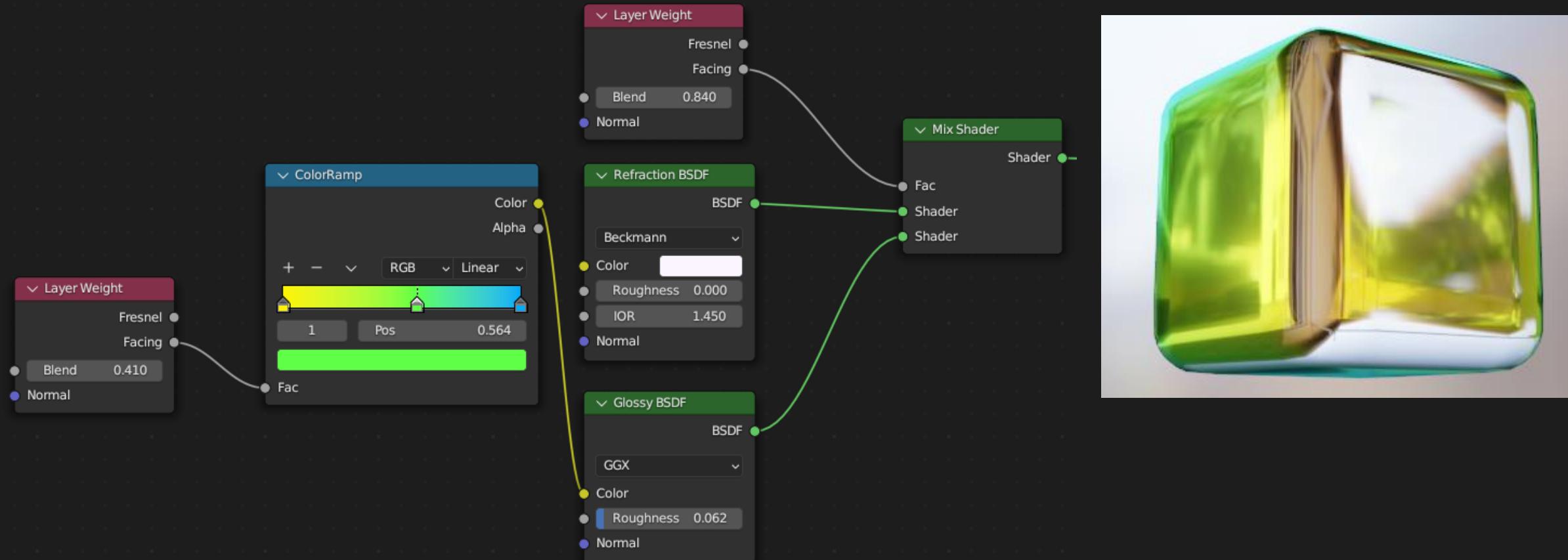
※프레넬 값에 따라 반사와 굴절이 결정되므로,
Add shader 가 아니라 Mix shader를 사용합니다.

구성요소 분리 (2)



홀로그램

색이 바뀌는 재질

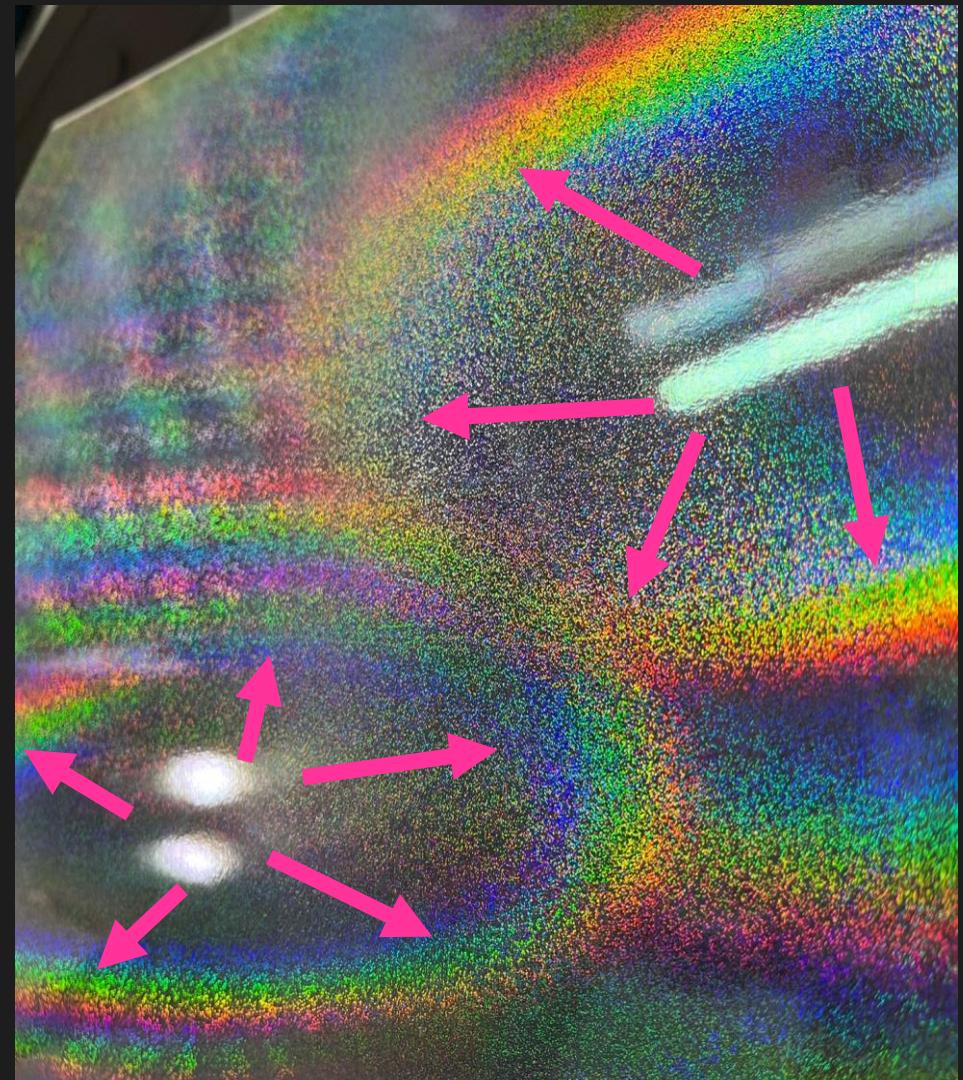


※보는 각도에 따라 그라데이션을 만드는 것이 아니라 이미지 텍스쳐를 교체한다면, 렌티큘러 이미지도 만들 수 있습니다.

홀로그램?

홀로그램 종이중에는 무지개색으로 반사가 일어나는 것도 있습니다.
이 무지개 무늬는 광원과 표면이 이루는 각도에 영향을 받는데
(Thin-film interference 와 흡사한 원리로 알고 있습니다.)
안타깝게도 블렌더에서 이를 컨트롤할 방법은 없습니다.

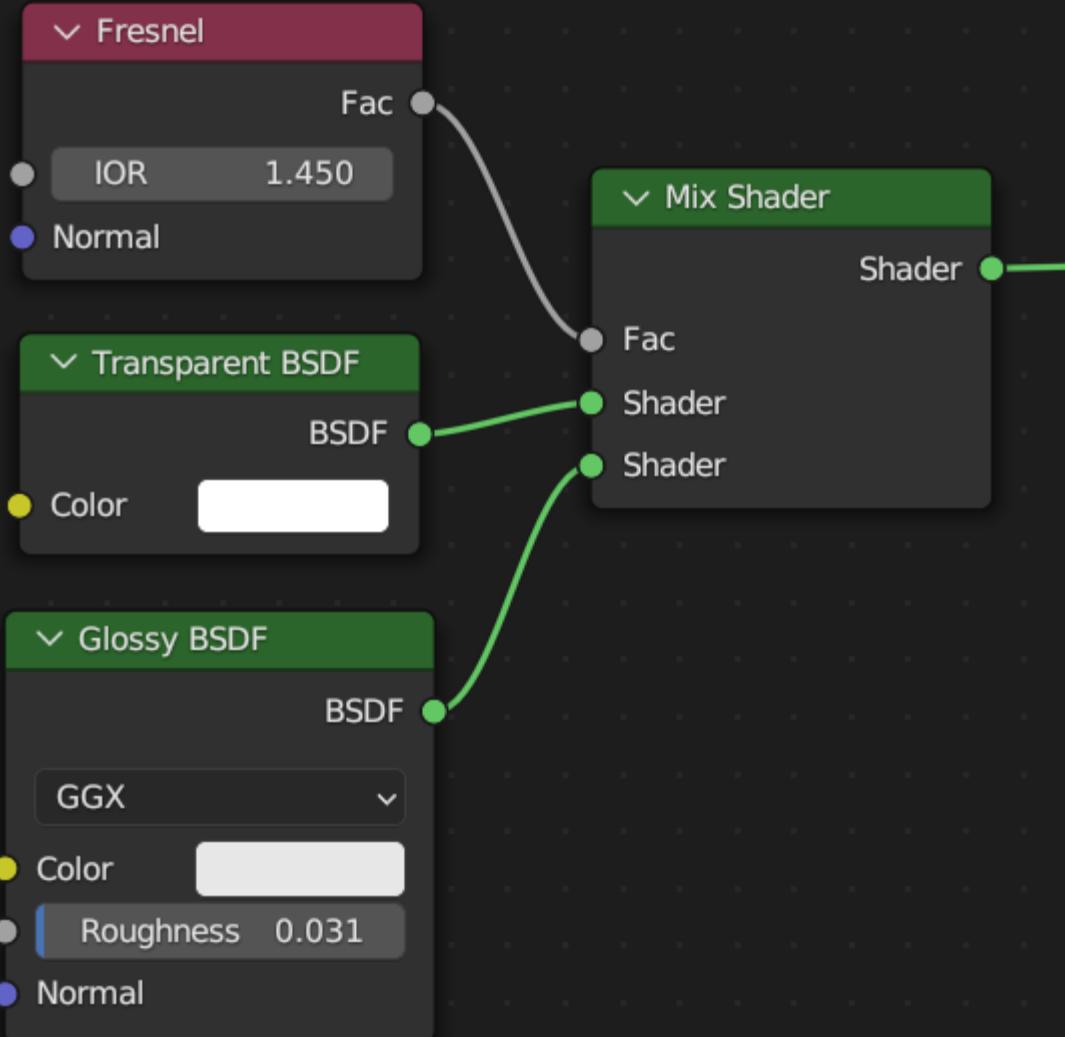
(26강 NPR에서 해본 것처럼 Normal의 내적값에
무지개색 ColorRamp를 연결하면 비슷하게 만들 수는 있습니다.)



뒤집힌 Normal과 Fresnel

만약 Fresnel을 Refraction이 아니라 Transparent와 같이 쓰면, 아래와 같이 이상하게 행동합니다.

이것은 노멀의 뒷면에서 Fresnel이 우리가 생각했던 것과 다르게 작동하기 때문입니다.



Schlick's approximation

Fresnel은 반사 뿐만 아니라 굴절과도 관련됩니다.

면의 노멀 방향이 뒤집혀 있다면, Fresnel은 안에서 밖으로 나가는 굴절을 표현하기 위해 이상한 모양이 됩니다.

(Snell's Window / <https://www.youtube.com/watch?v=mKX0ao5R5fE>)

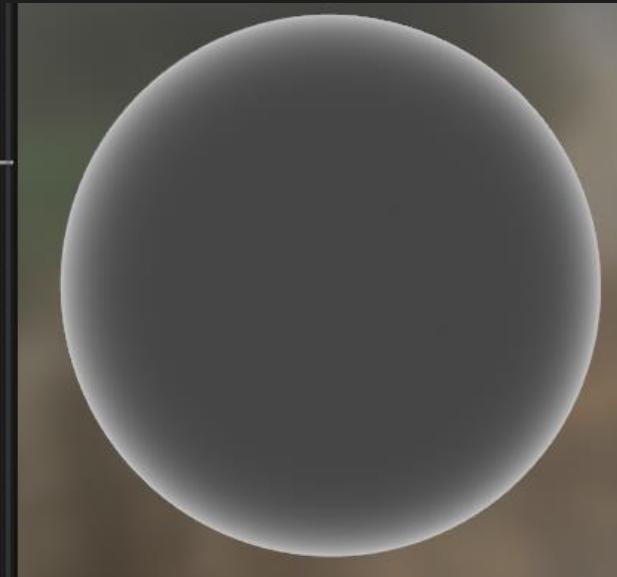
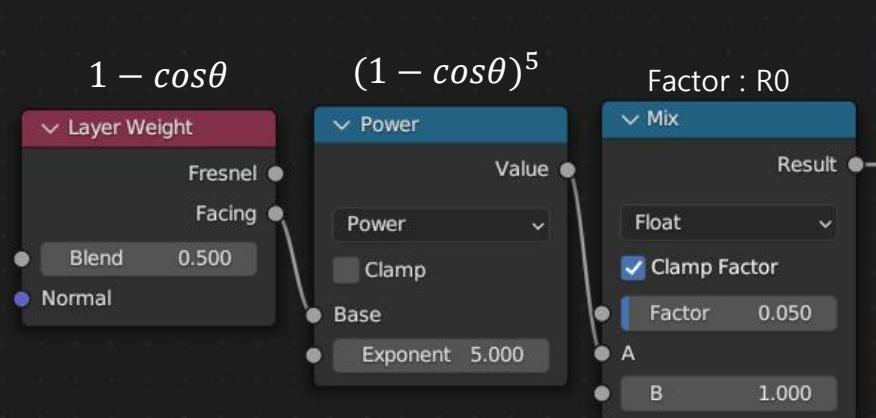
굴절이 아닌 투명에서 Fresnel을 사용하면, 뒷면에서의 Fresnel 모양이 보이게 됩니다.

노멀 방향과 상관없이 앞면 모양이 나오길 바란다면 Fresnel 말고 Facing을 사용합니다.

Facing을 Fresnel과 동일한 형태로 만들고 싶을 땐 Schlick's approximation을 사용합니다.

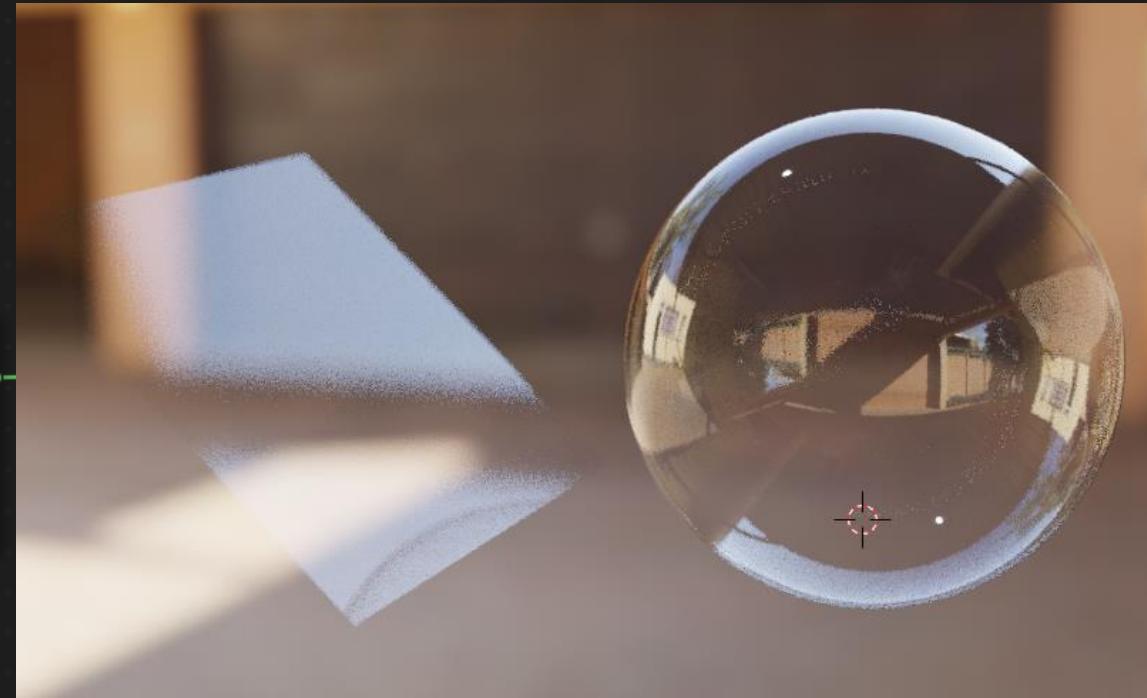
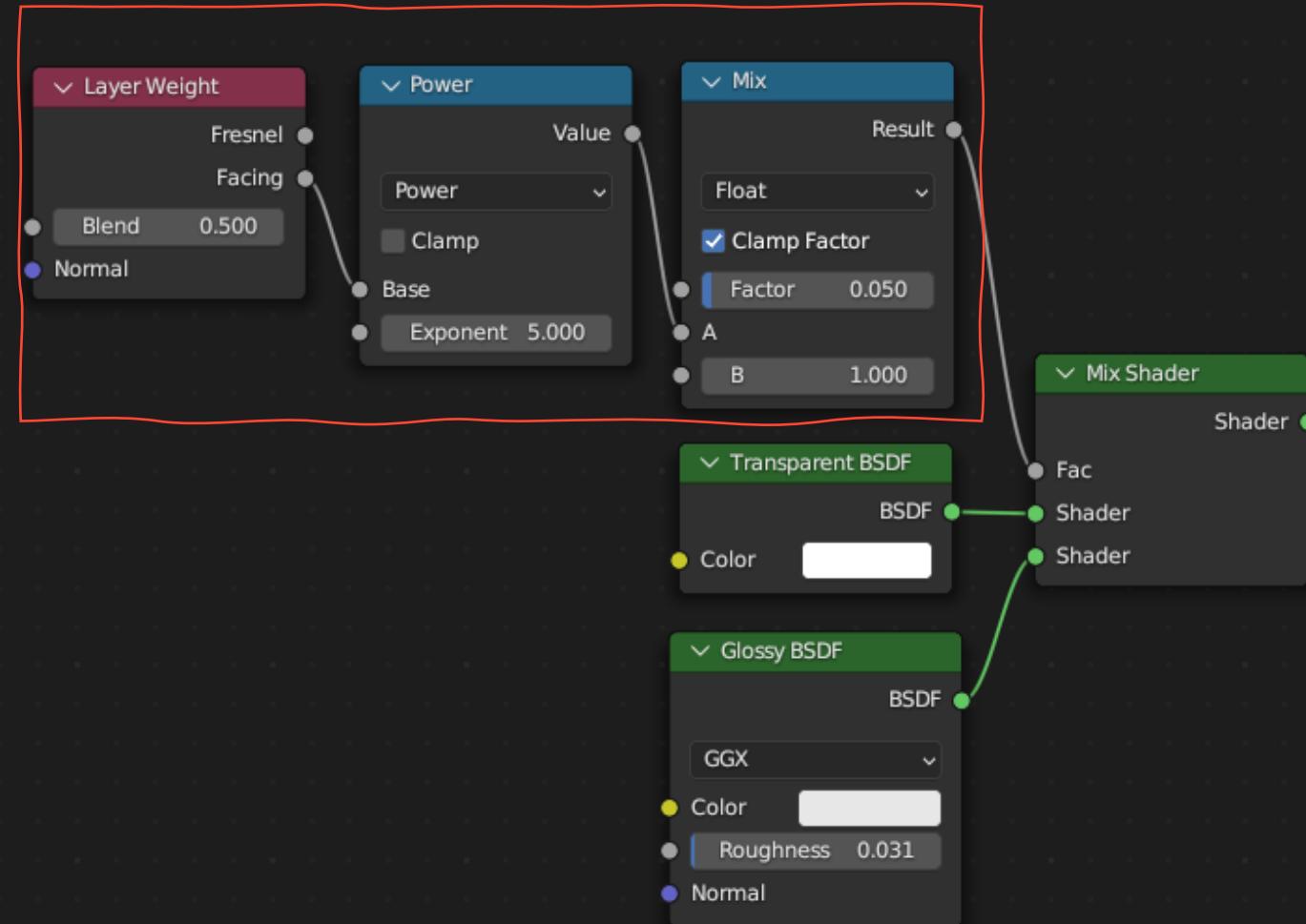
$$R_0 + (1 - R_0) (1 - \cos\theta)^5$$

이 식에서 facing은 $1 - \cos\theta$ 입니다.



Schlick's approximation

≈ Fresnel

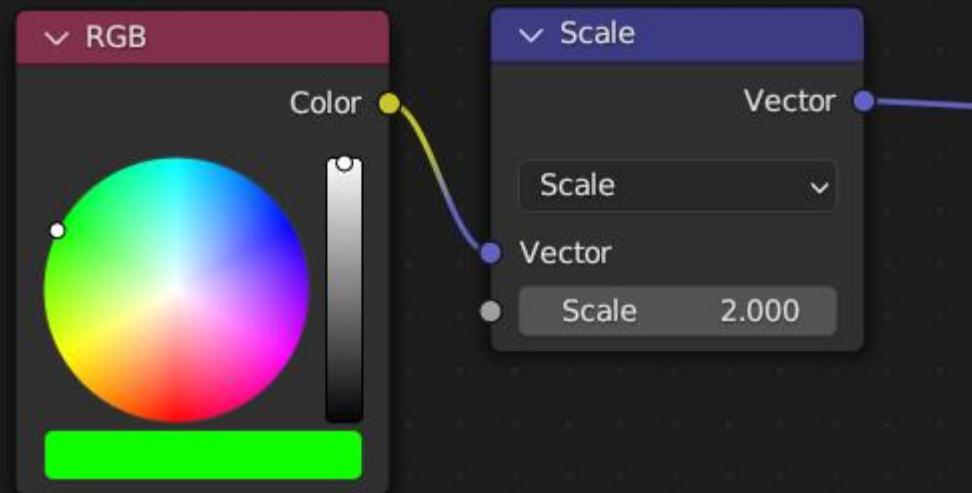


형광색

형광은 실제로 빛을 방출하기 때문에 일반적인 색보다 더 밝아야 합니다.

Emission을 사용해도 밝아지지만, 외부 환경과 상호작용하지 않습니다. (야광이라면 가능하겠지만..)

아래와 같이 Scale을 이용하여 밝기를 높이면, 주변 환경에 따라 밝기가 바뀌는 밝은 색이 나옵니다.



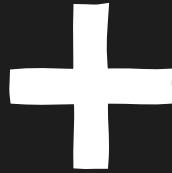
반투명 플라스틱

플라스틱의 IOR은 유리와 매우 흡사하지만, 실제로 표현되는 색은 많이 다릅니다.

그것은 플라스틱은 두께에 따라 불투명해지고, 그 양상이 유리와 다르기 때문입니다.

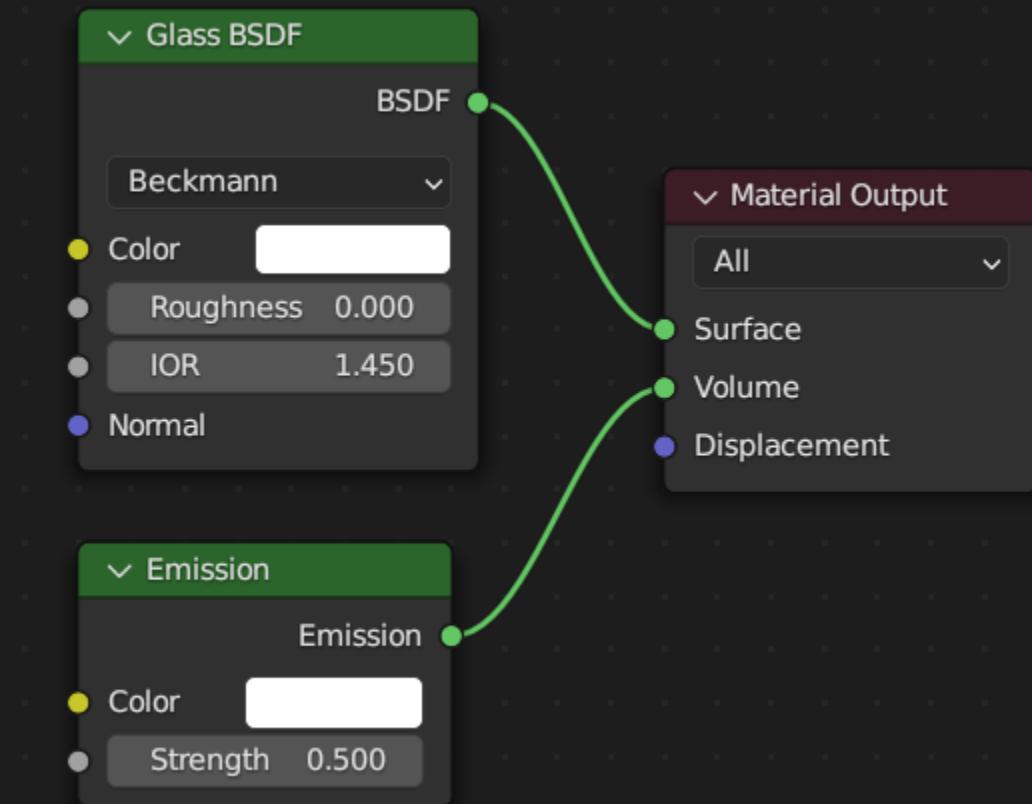
반투명 플라스틱을 만들기 위해서는 유리에 **불투명한 투과 재질 (Translucent)**을 섞되, 팩터를 ‘**두께에 따라**’ 조절해야 합니다.

※ fresnel로 섞어도 그럴듯합니다만, 두께 반영이 되지 않으므로 뭔가 부족한 느낌이 듭니다.



반투명 플라스틱

'두께에 따라' 를 가장 쉽게 해결하는 방법은 볼륨 셰이더를 사용하는 것입니다.
하지만 이 경우 Emission만 사용할 수 있으므로 완벽한 해결책은 아닙니다.



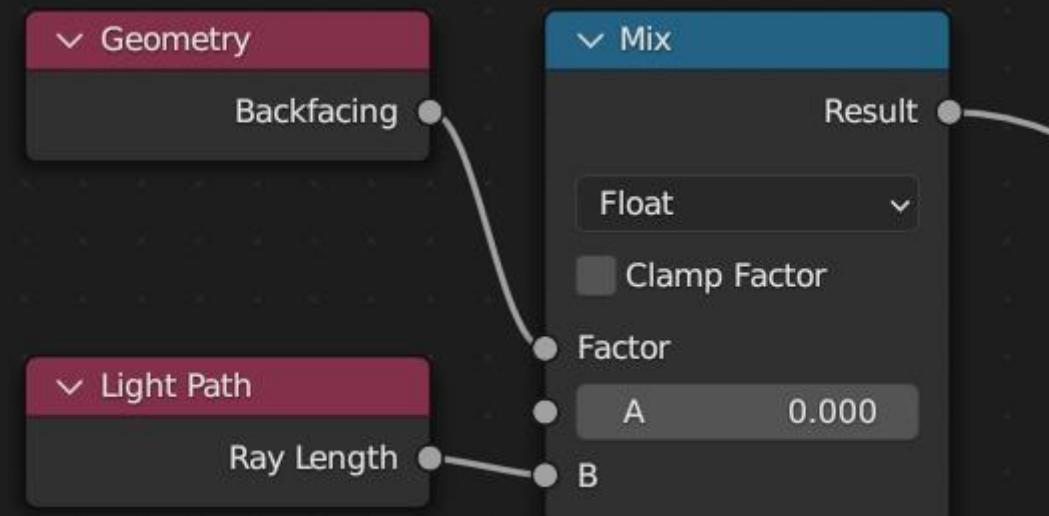
반투명 플라스틱

Light Path의 Ray Length로 물체의 두께를 간접적으로 잴 수 있습니다.

두꺼운 부분이라면 Ray Length도 길 것이기 때문입니다.

하지만 Ray Length는 안쪽과 바깥쪽을 구분하지 않으므로, Backfacing을 이용합니다.

아래와 같이 뒷면일때만 Ray Length가 나오고, 앞면일 때는 0이 나오게 해서 길이를 재지 않도록 만듭니다.



반투명 플라스틱

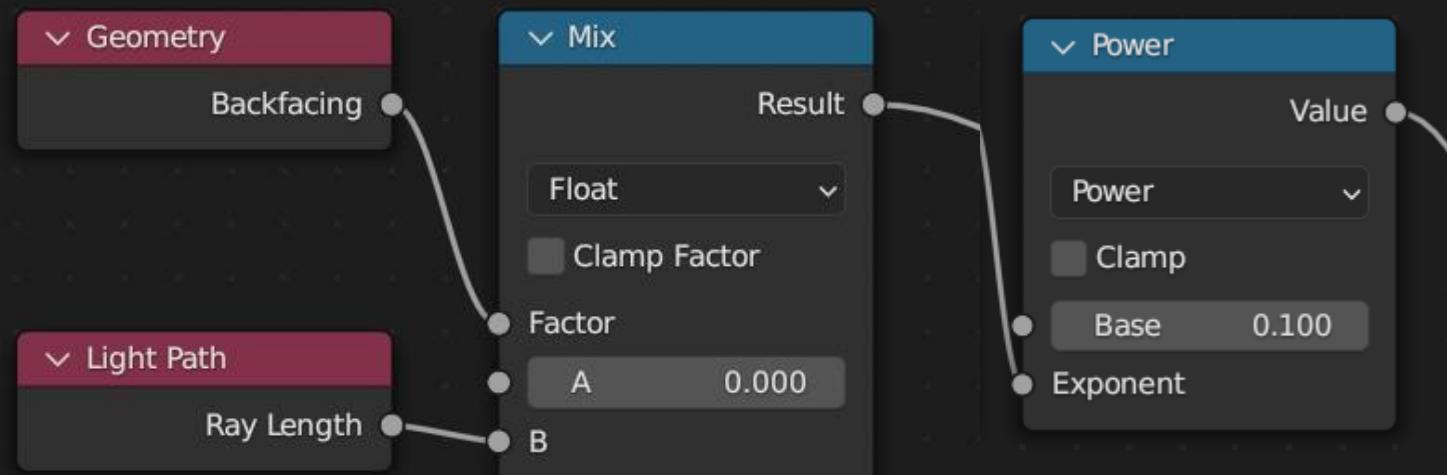
이제, 얻은 두께를 어떻게 사용할지 생각해야 합니다.

두께라는 연속적인 길이를 생각하면 어려우므로, 셀로판지를 한겹씩 겹치는 상황을 생각해 봅시다.

셀로판지를 하나씩 겹칠 때마다, 투과도는 %로 줄어들 것입니다.

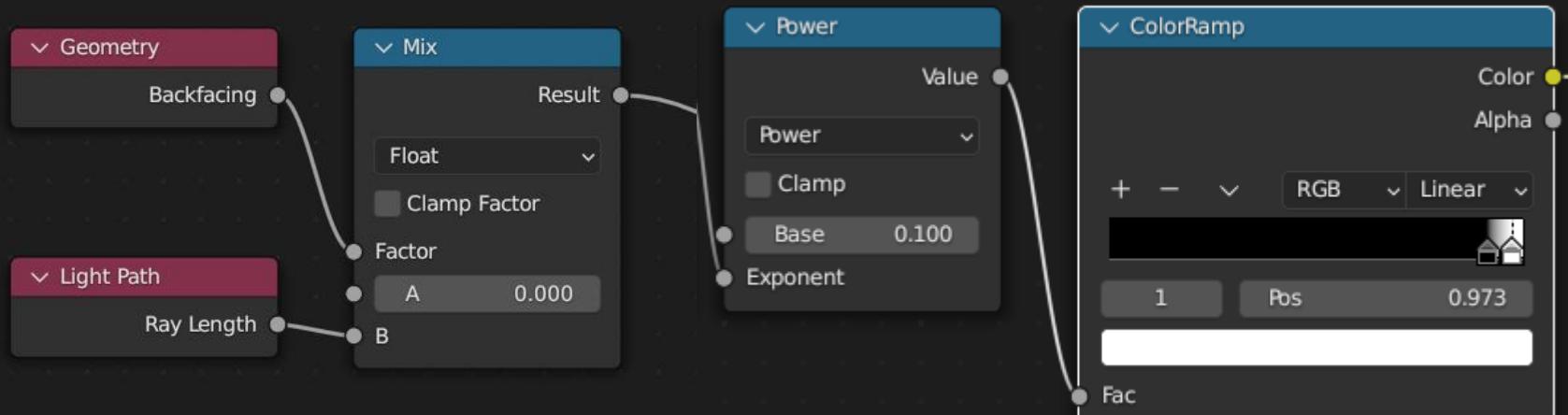
예컨대 투과도 90%의 셀로판지라면, 두 장 겹치면 $0.9 * 0.9 = 0.81$ 로 81%의 투과도가 될 것입니다.

반복해서 곱해나가는 연산 = 거듭제곱 = power의 exponent 연결입니다.



반투명 플라스틱

마지막으로 ColorRamp를 이용해 세기를 조절합니다.
여기부터는 수학적으로 엄밀하지 않지만..



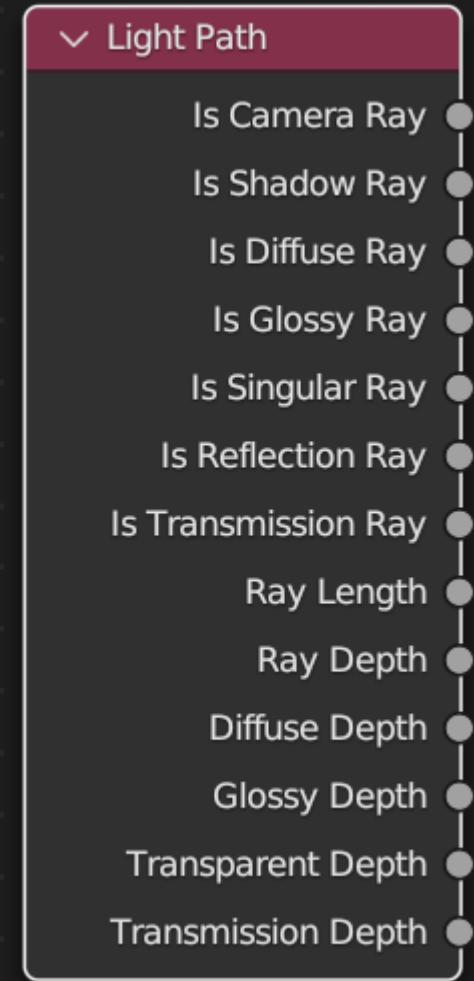
반투명 플라스틱



028강 Light Path

Cycles 엔진이 이미지를 만드는 원리

Light Path 노드

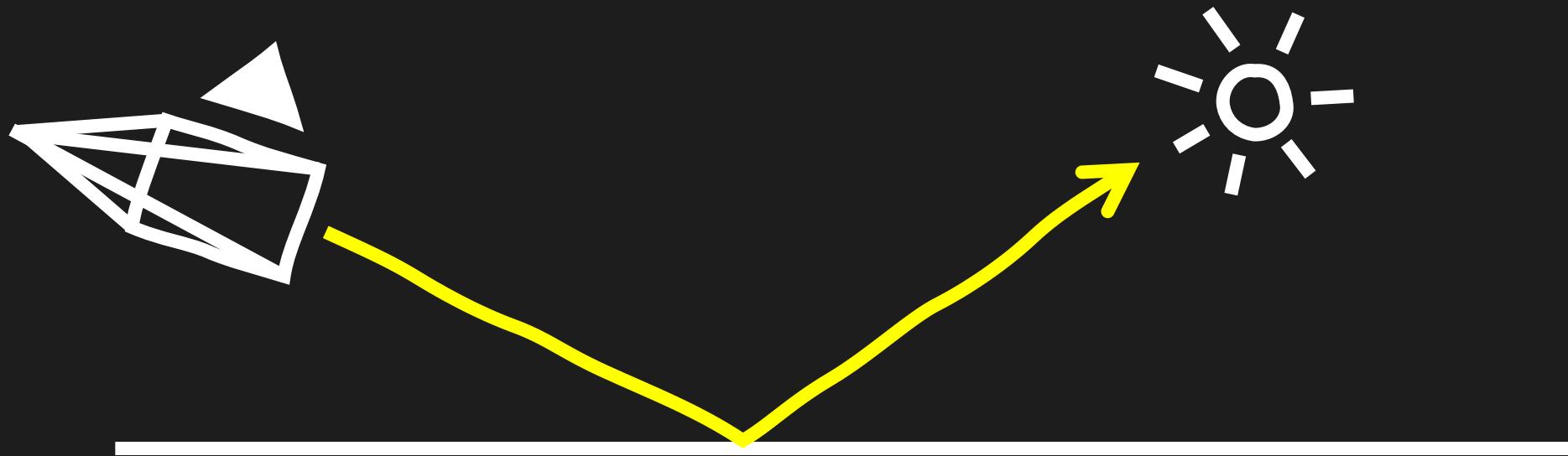


Cycles의 기본 원리

광선은 카메라에서 나옵니다

Cycles를 비롯한 대부분의 Path tracing은 카메라에서 Ray를 쏘서 광원에 닿을 때까지를 연산합니다.

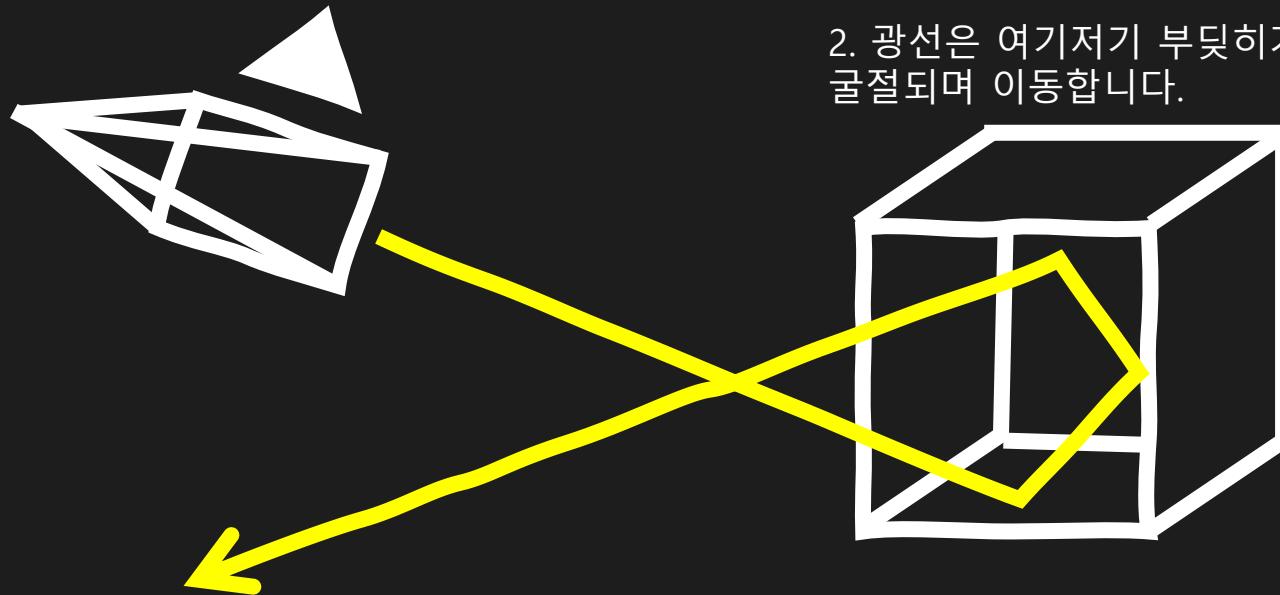
- 연산은 어딘가에 닿을 때마다 일어납니다.
- 표면에 닿아서 일어난 연산은 표면 뿐만 아니라 그 전 단계에도 영향을 미칩니다.



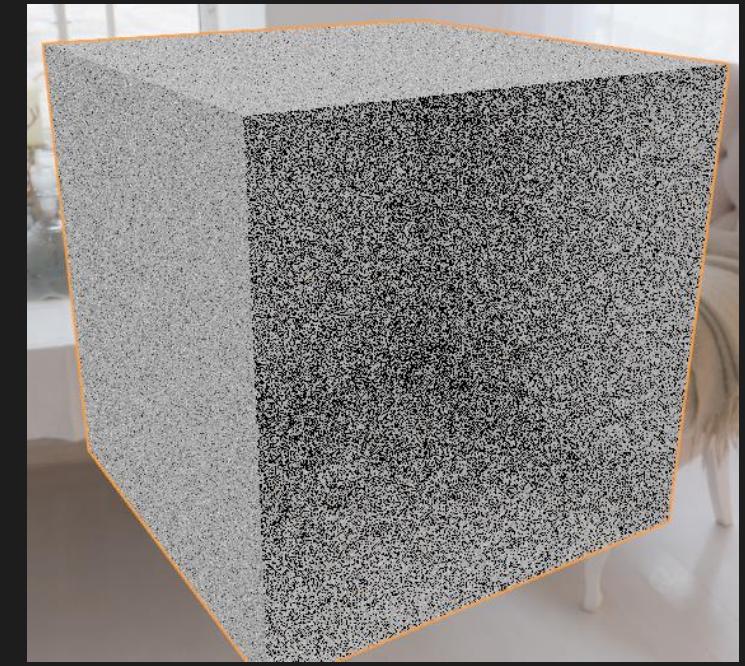
Cycles의 기본 원리

연산의 기본 흐름

1. 광선이 카메라에서 출발합니다.



2. 광선은 여기저기 부딪히거나,
굴절되며 이동합니다.



3. 광원에 닿으면 연산은 종료됩니다.

연산이 종료되면 그 결과를 처음 쏘아올린
방향(위치)에 기록합니다.

연산이 종료되는 시점은

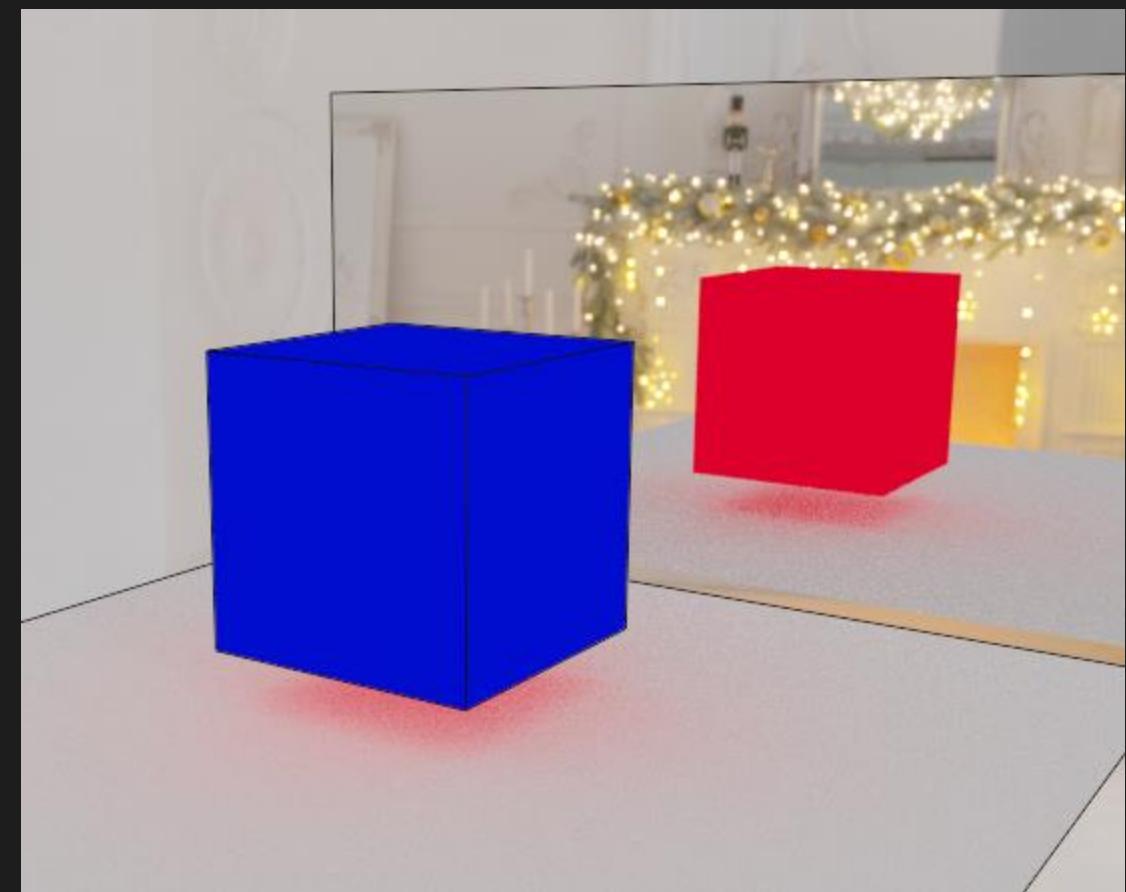
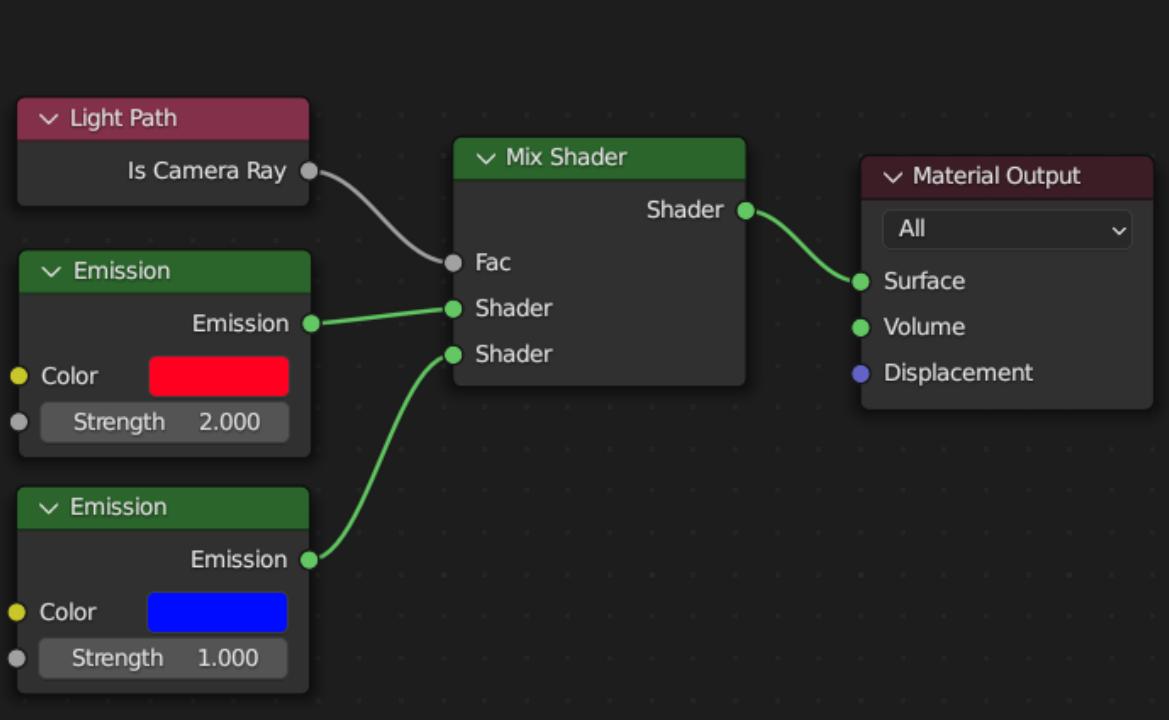
- Light Object에 닿거나
- Emission 재질에 닿거나
- 배경에 닿거나
- 연산 횟수를 초과하는 경우가 있습니다.

Light Path

Camera ray

카메라에서 막 출발했을 때의 광선을 Camera Ray라고 합니다.

무언가와 상호작용한 시점부터 Camera ray가 아닙니다. (투명 재질 제외)



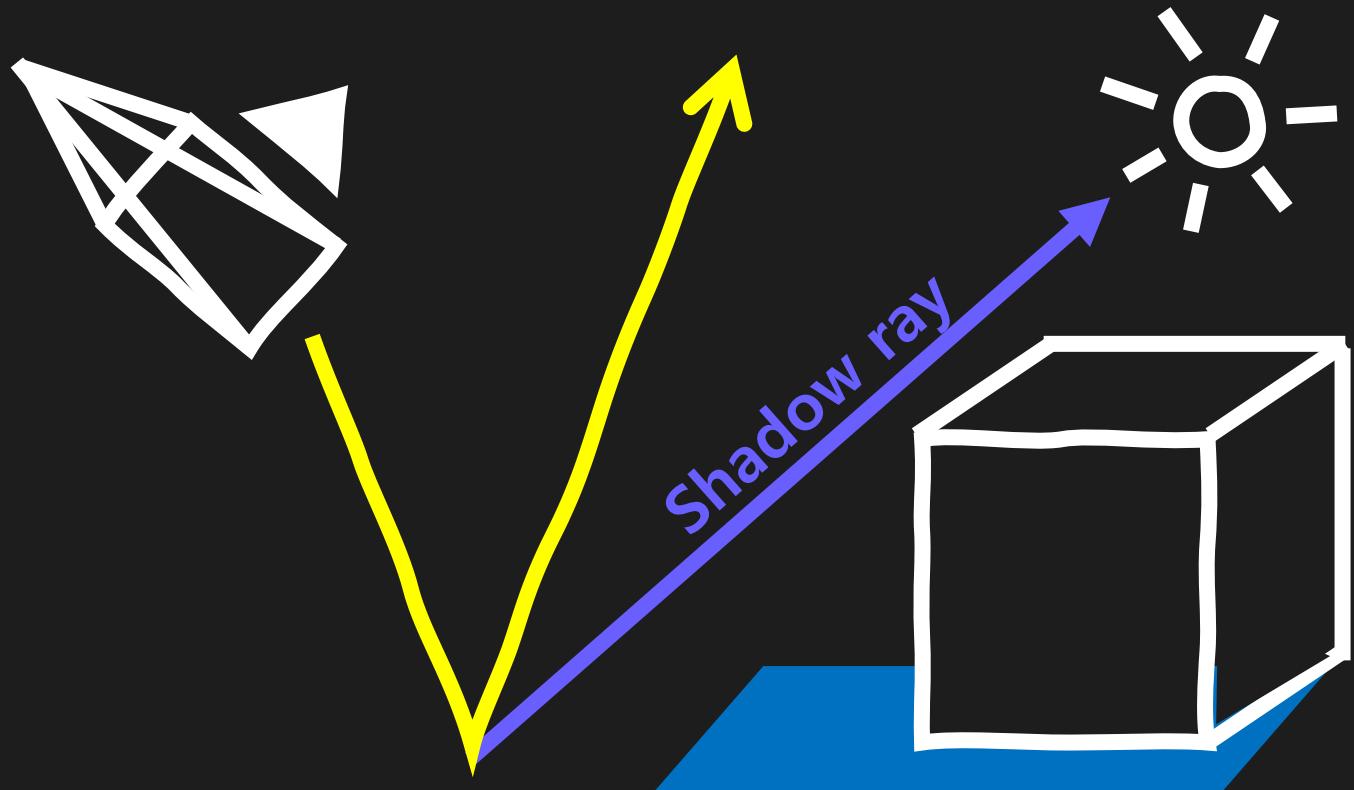
오브젝트에 적용하여, 카메라로 볼 때만 파란색으로 만들었습니다.
그림자나 비치는 모습 등 나머지 연산에선 빨간색으로 작동합니다.

Light Path

Shadow ray

바운스할 때마다, 반사/굴절의 원리와 상관없이, 광원을 향해 진행하는 광선이 존재합니다.
이렇게 분리된 광선을 Shadow Ray라고 합니다.

- Shadow Ray는 다시 반사되거나 하지 않고, 오로지 광원에 닿는지/아닌지만 판단합니다.
- 광원에 닿으면 해당 부분을 밝게, 닿지 않으면 어둡게 표시하는 역할을 합니다. 즉, 그림자를 만듭니다.



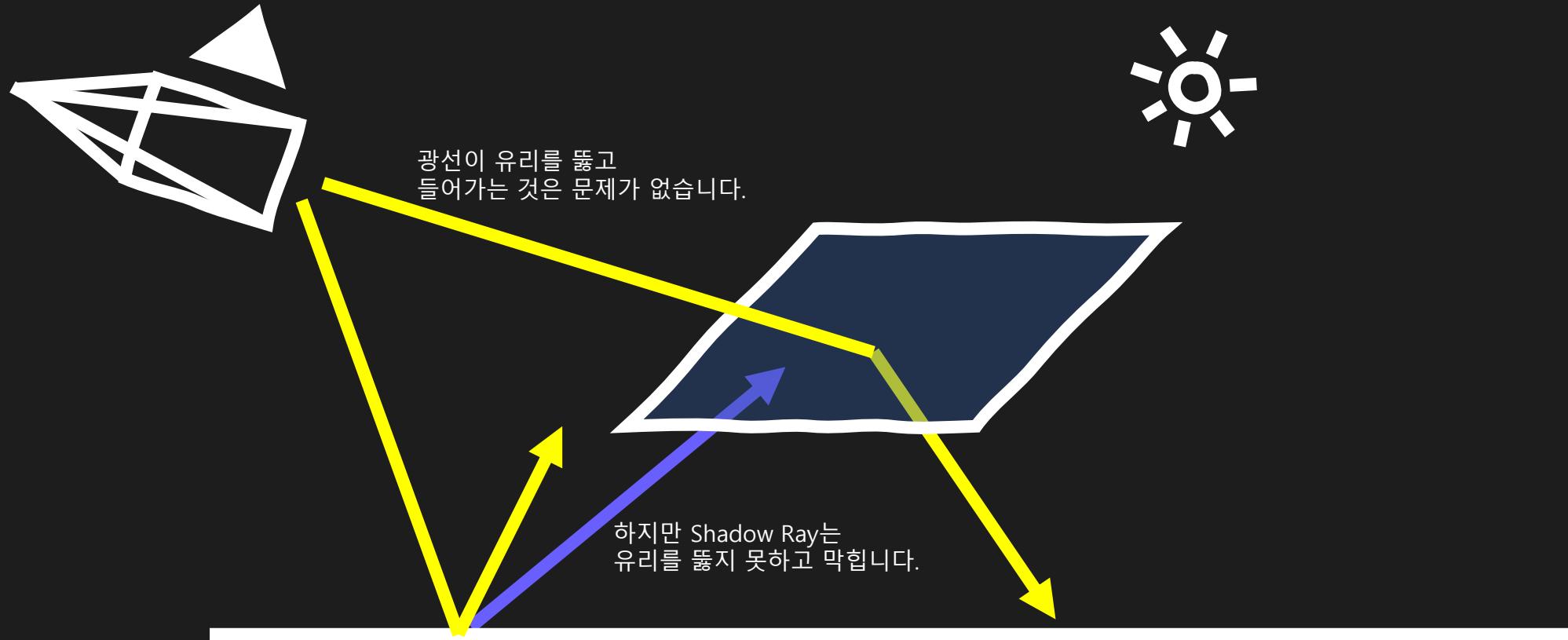
Light Path

Shadow ray

Shadow Ray는 오로지 투명 재질 (Transparent) 만 통과할 수 있습니다.

10강에서 알아본 유리의 문제점이 여기에 있습니다.

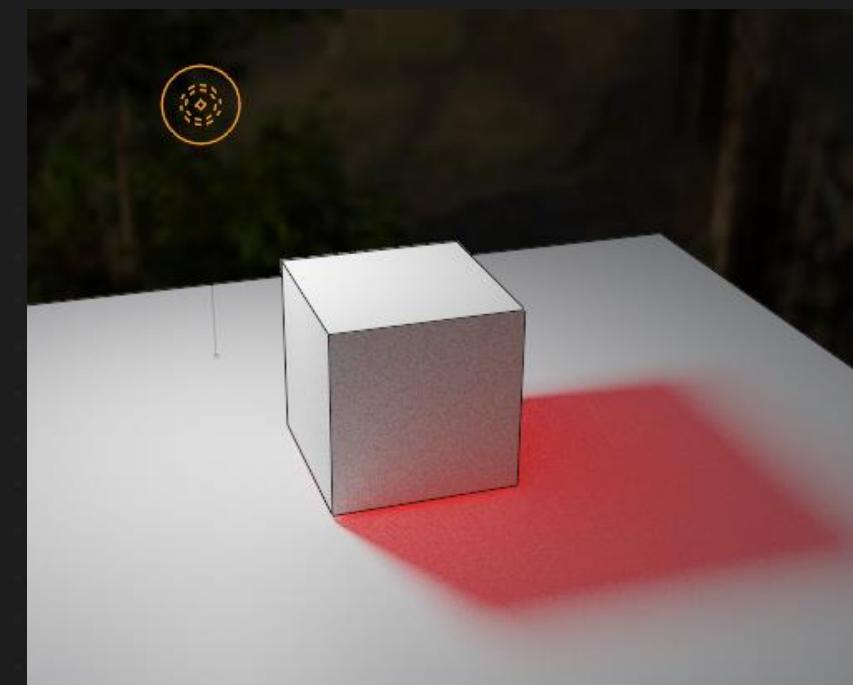
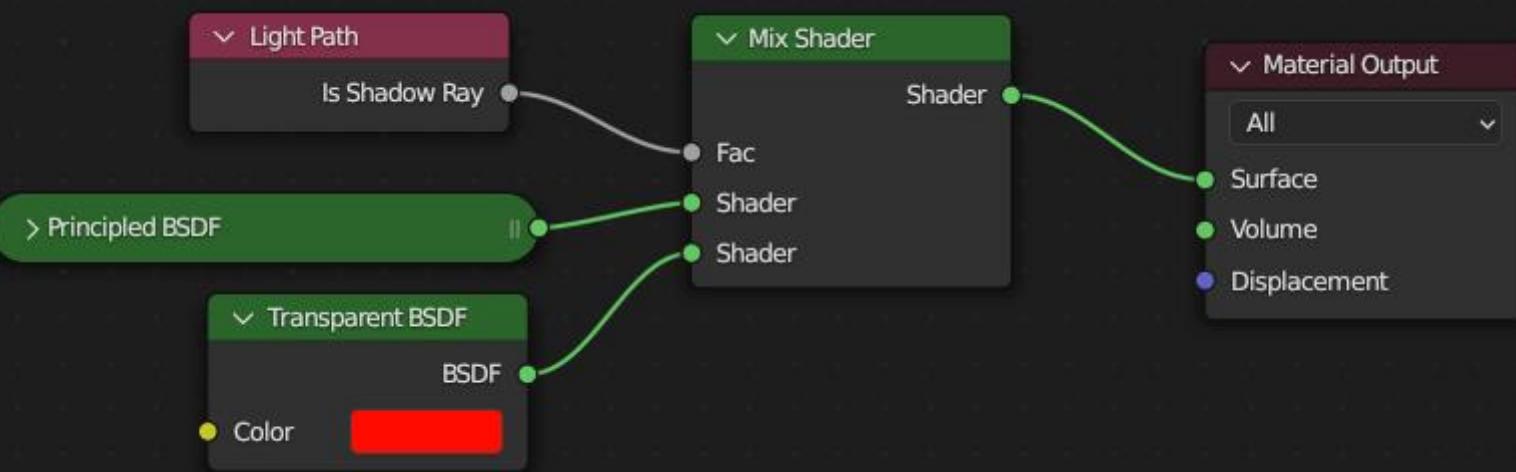
유리는 transparent가 아니기 때문에 Shadow Ray가 통과할 수 없고, 그에 따라 그림자가 생깁니다.



그림자 색을 바꾸려면?

그림자는 바닥에 생기지만, 그림자가 생기는 물체 쪽을 아래처럼 바꾸어야 합니다.

1. 카메라에서 나온 광선이, 바닥에 닿아 Shadow Ray가 분리됩니다.
2. Shadow Ray는 바닥에서 광원을 향해 날아가다 물체에 닿습니다.
3. 물체의 셰이더 연결은 아래와 같아서, Shadow Ray는 통과해서 광원에 닿습니다.
4. 해당 계산 결과는 처음 광선이 향했던 곳 : 바닥에 적용됩니다.

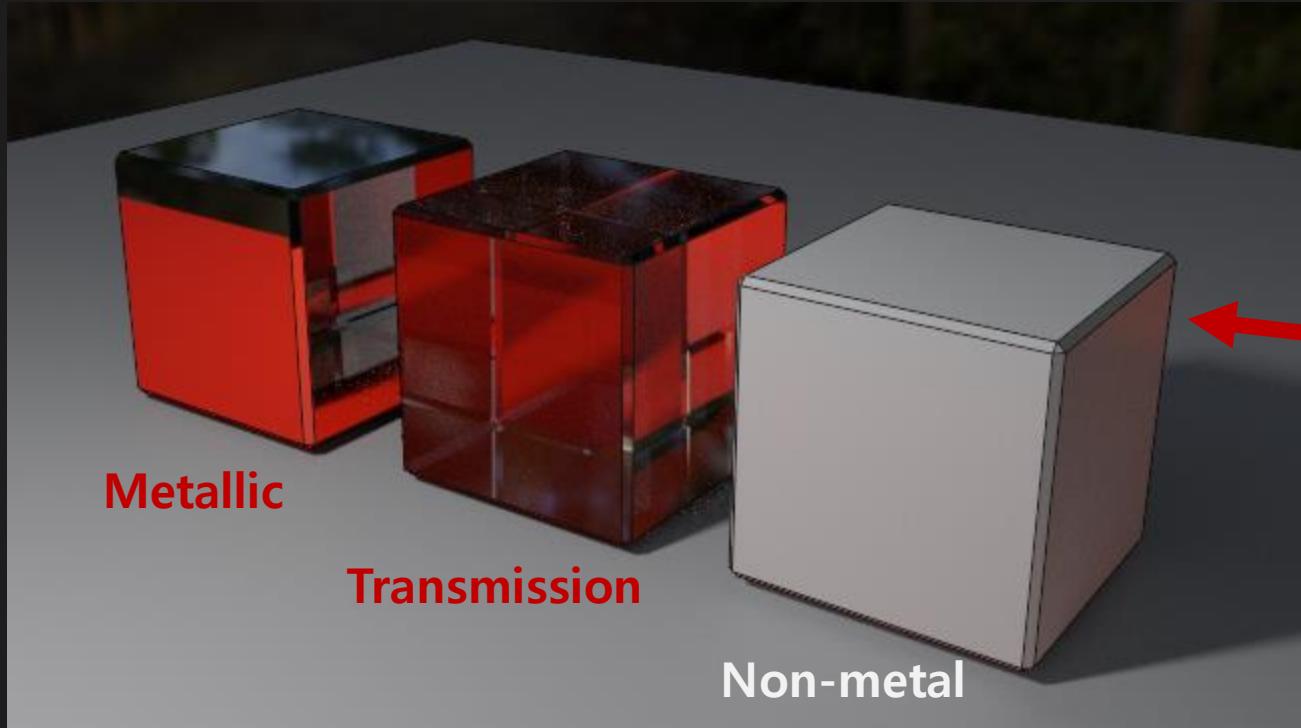


Light Path

Diffuse vs Glossy

Diffuse에 반사된 광선을 Diffuse Ray

Glossy에 반사된 광선을 Glossy Ray라고 합니다.



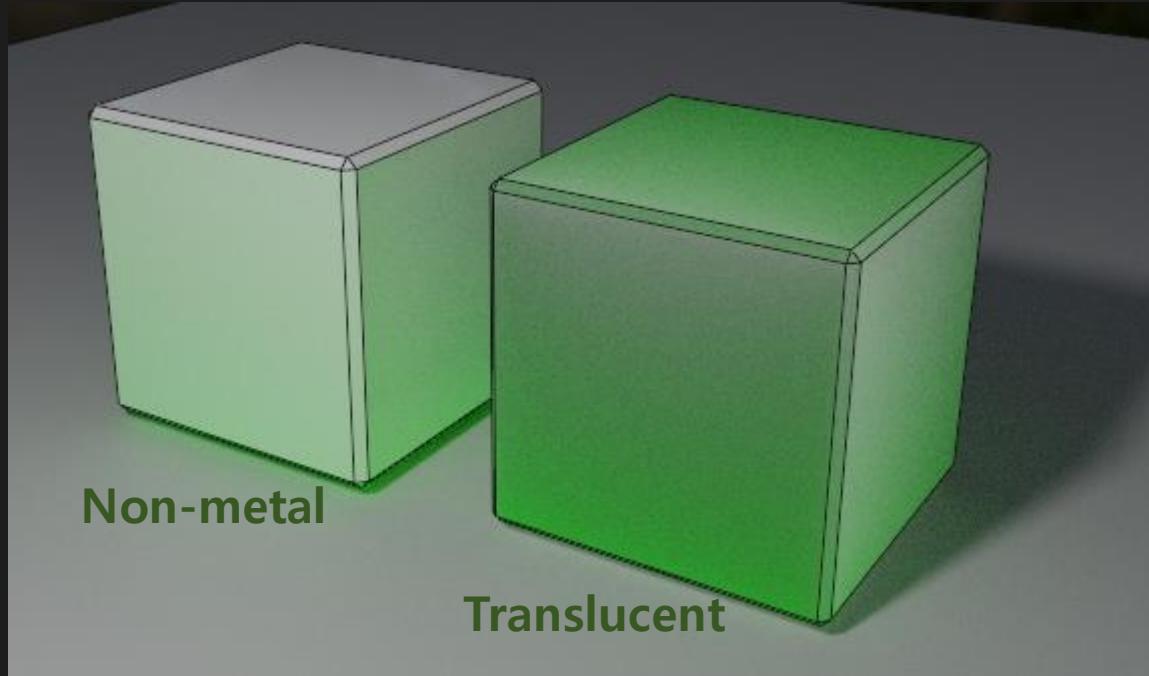
Glossy의 종류 : Metallic , Transmission, Specular

바닥을 'Glossy일 때 빨간색' 으로 했을때의 결과입니다.

- Glossy에는 유리재질도 포함됩니다.
Principled BSDF 의 Metallic을 꺼도, 광택이 있는 부분은 Glossy입니다.

Light Path

Diffuse vs Glossy



Diffuse의 종류 : Non-Metal, Translucent

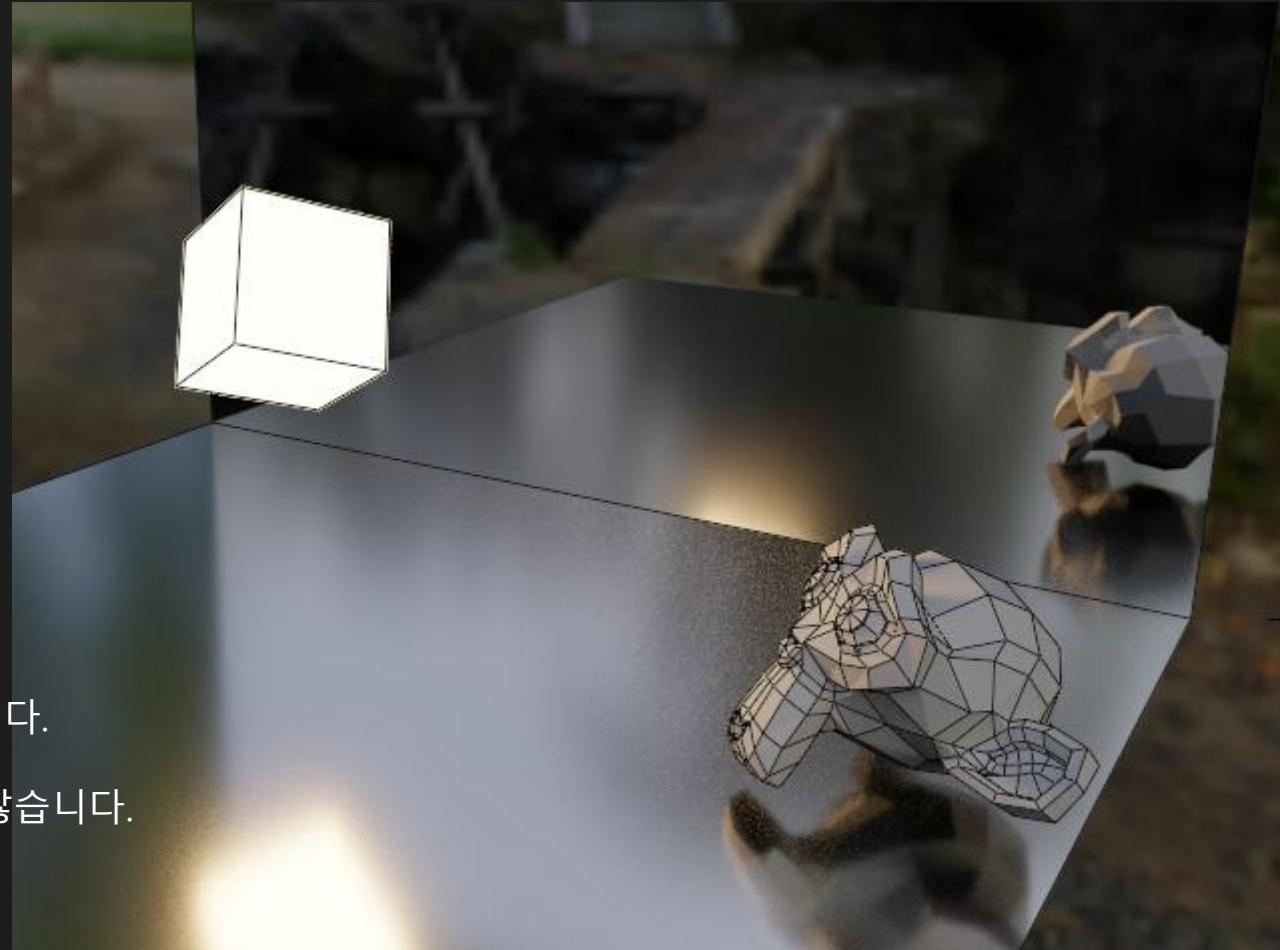
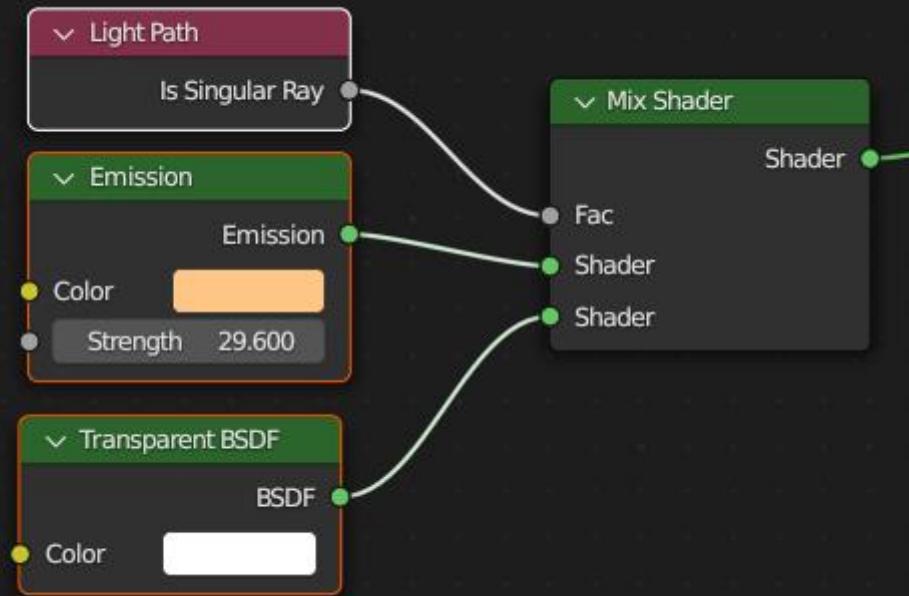
바닥을 'Diffuse일 때 초록색'으로 했을 때의 결과입니다.

- Diffuse에는 Translucent도 포함됩니다.

Light Path

Singular

러프니스 0으로 반사된 광선을 Singular Ray라고 부릅니다.



박스를 Emission으로 사용하되, Singular일 때 투명으로 둔 예시입니다.

바닥은 박스를 반사하지만, 러프니스 0인 정면의 거울은 반사하지 않습니다.

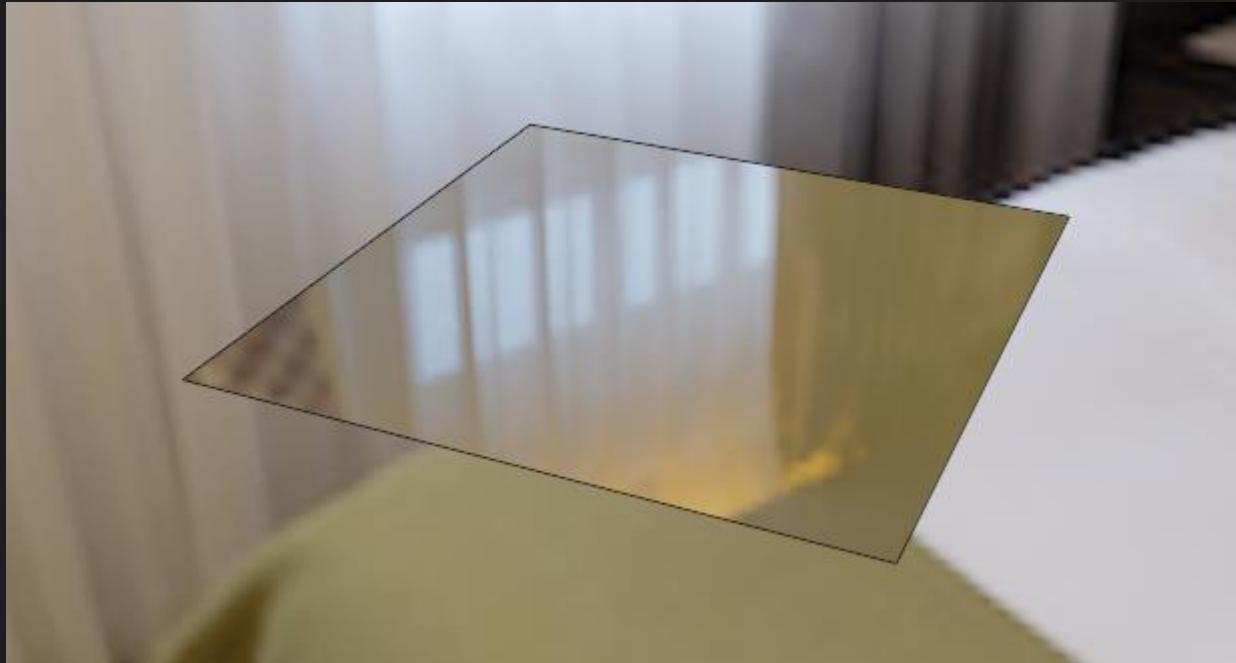
※조명에는 사용할 수 없습니다. 자세한 것은 30강 참고.

Light Path Transmission

굴절된 광선을 Transmission Ray 라고 부릅니다.

굴절 공식은 현실과 같지만, 3D에서 굴절은 매질에 의해 생기는 것이 아닙니다.

하나의 굴절 면만 존재하면 됩니다.



블렌더에선 하나의 면만 있어도 굴절이 일어납니다.

앞쪽 면을 소한 매질, 뒤쪽 면을 밀한 매질로 보고 계산합니다.

그에 따라 노멀이 뒤집히면 바깥 전체가 물속에 있는 것처럼 작동하기 때문에, 스넬의 창이 생기고 부자연스러워집니다.

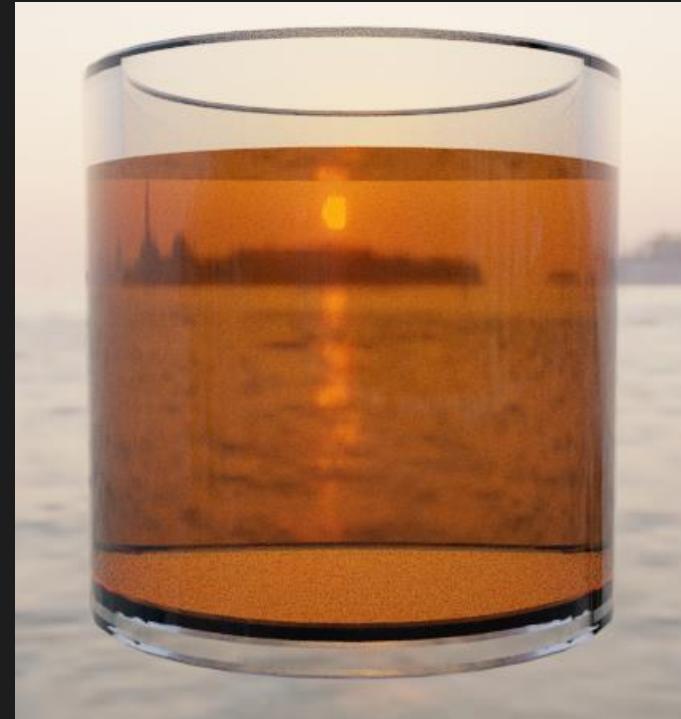
※Eevee의 경우, 여러 면이 겹쳐도 한번의 굴절만 일어납니다.
Eevee에서 유리로 창문을 만들면 안되는 이유입니다.

Light Path Transmission

연산의 특성때문에 생기는 문제가 하나 있습니다.



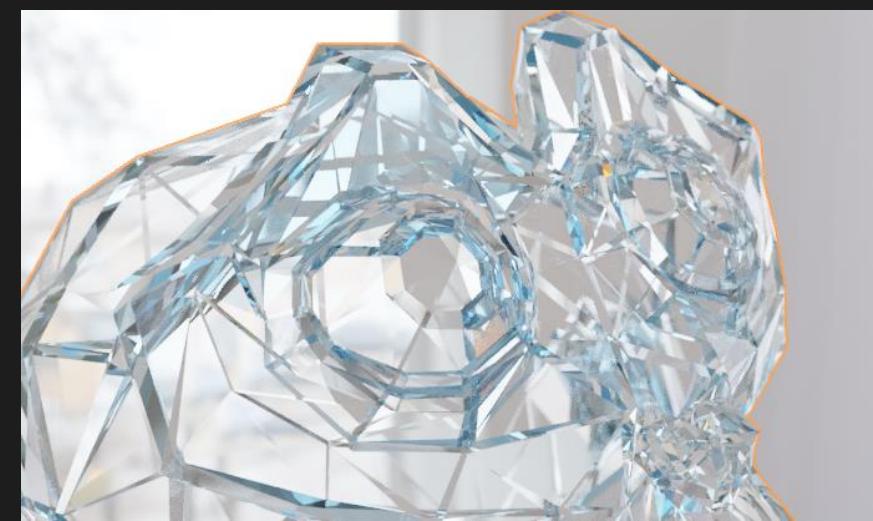
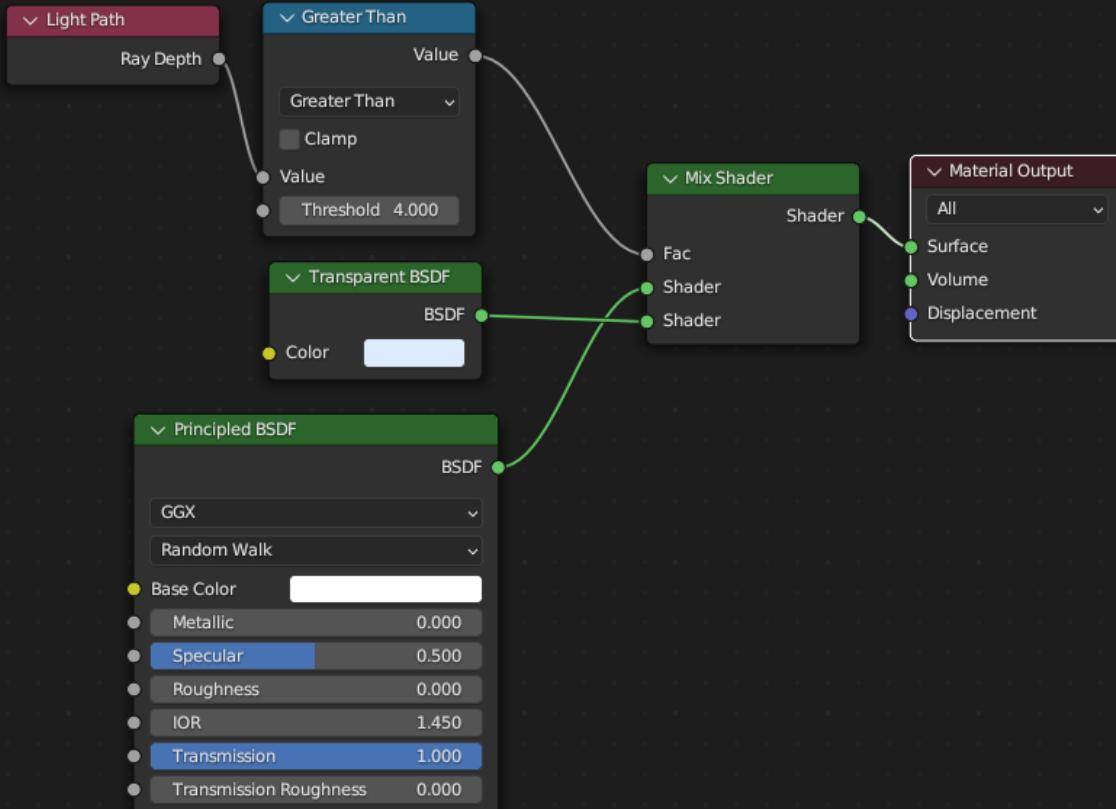
만약 유리컵 안에 물 오브젝트가 존재한다면..



물 오브젝트가 컵보다 약간 커서 면이 겹쳐지면 이렇게 됩니다.
현실은 이쪽에 더 가깝습니다.

유리가 너무 어두울 때

유리 내부에서 전반사가 일어나 빛이 제대로 빠져나오지 못하면, 위쪽 이미지처럼 부분적으로 어두워집니다. 이것은 현실에서도 일어나는 일이지만, 만약 너무 과하게 어둡다면 Depth가 충분히 클 때 Transparent로 바꿔 통과시켜 주면 됩니다. Transparent의 색을 조절하면 유리다운 색으로 만들어줄 수도 있습니다.



Light Path

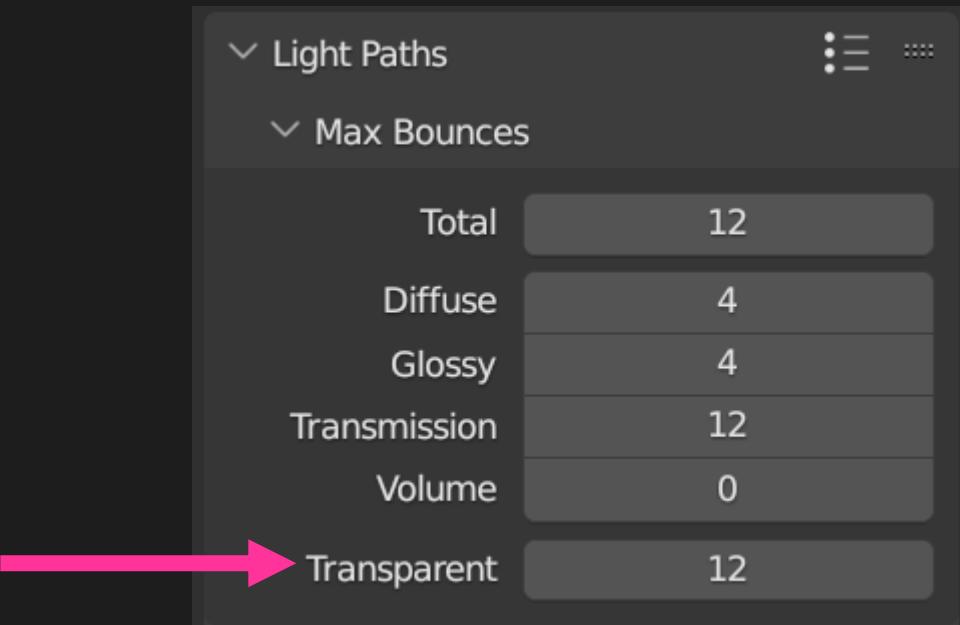
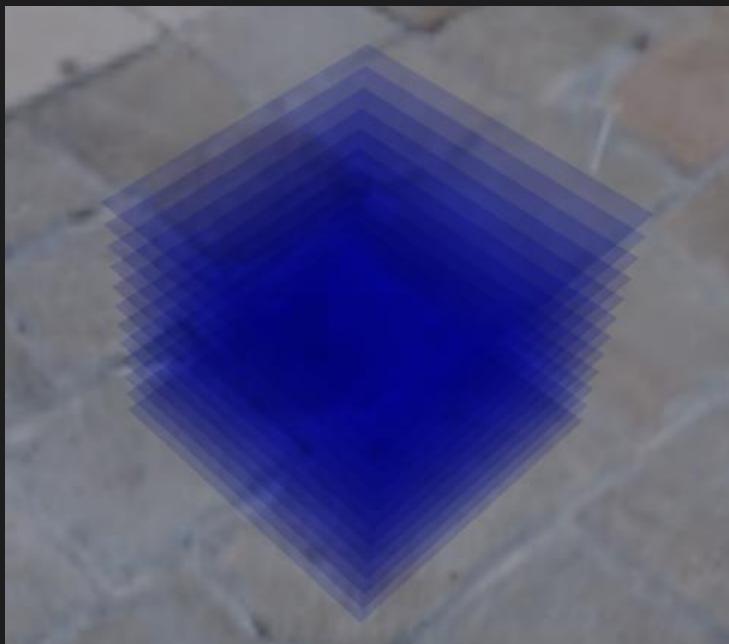
Transparent

Transparent특성은 오로지 Transparent BSDF에 의해서만 생깁니다. (유리재질 안됨!)

-Transparent는 Shadow ray가 통과할 수 있는 유일한 재질이며,

-바운스 횟수를 셀 때도 따로 셹니다.

-Camera Ray가 Transparent를 통과해도 Camera Ray의 특성이 변하지 않습니다.

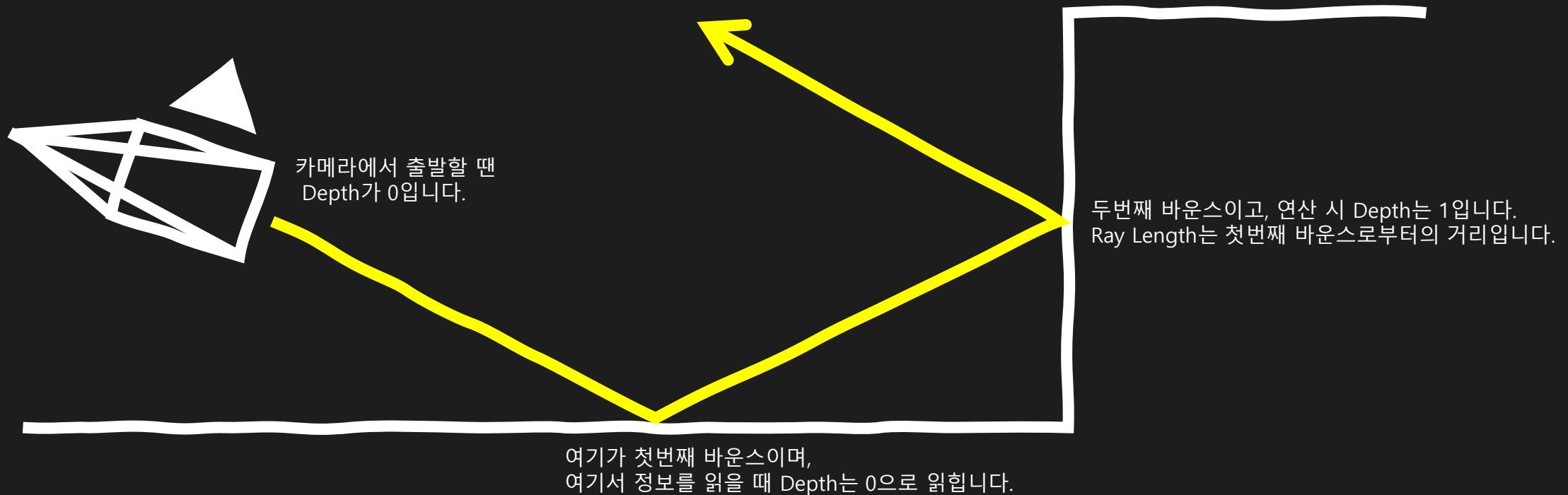


Light Path

Ray Depth, Length

Ray Depth는 지금까지 바운스한 횟수를 의미합니다. (그러므로, Camera Ray는 Depth가 0입니다.)

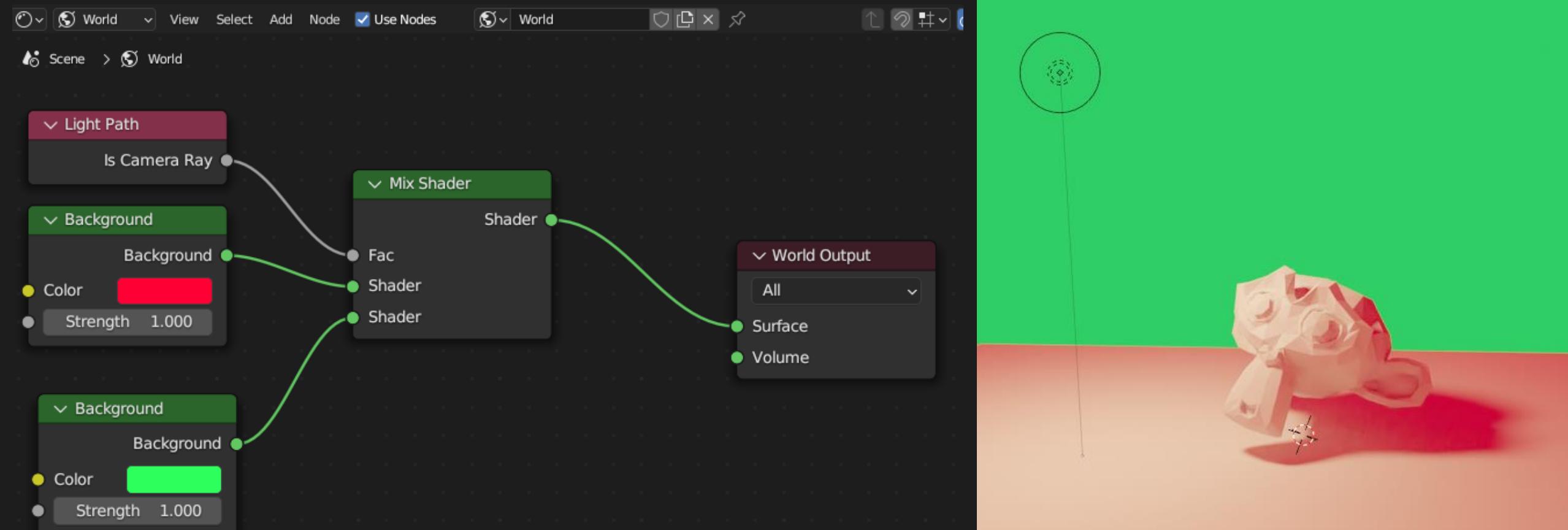
Ray Length는 마지막 바운스로부터의 거리를 의미합니다. (전체 거리가 아님에 유의하세요!)



예시 – 배경에서의 활용

배경 분리

Camera Ray를 이용해서, 눈으로 볼 때의 배경과 상호작용하는 배경색을 분리시킬 수 있습니다.
HDRi를 쓰고 싶은데, 눈으로 볼 땐 단색으로 하고 싶을 때 자주 사용하는 방법입니다.

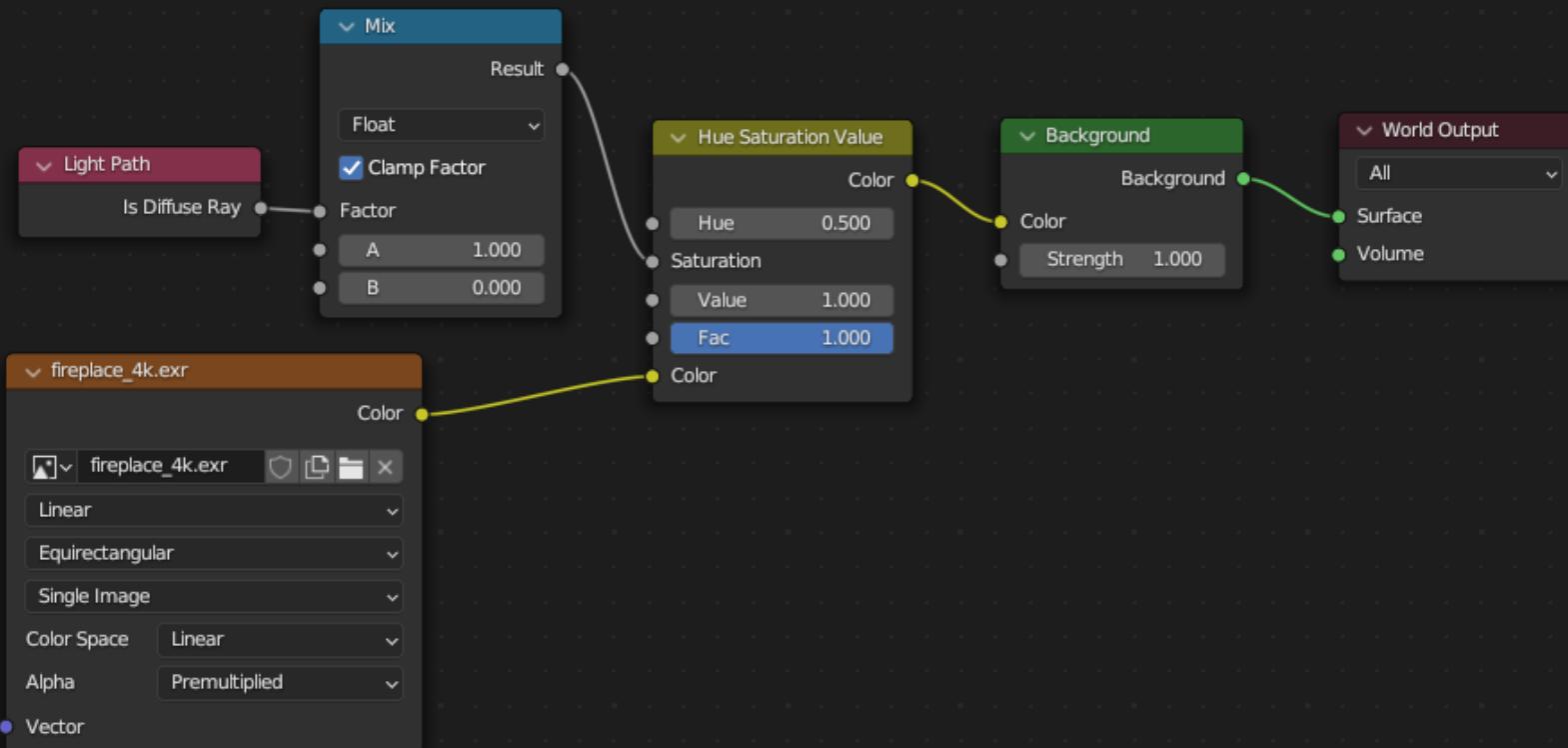


예시 – 배경에서의 활용

물체의 기본색 강화

Diffuse는 물체의 기본색 역할을 합니다.

만약 Diffuse일 때 **배경의 채도를 낮춘다면**, 배경 광원 효과와 반사효과는 그대로 사용하면서 물체 기본색을 살릴 수 있습니다.



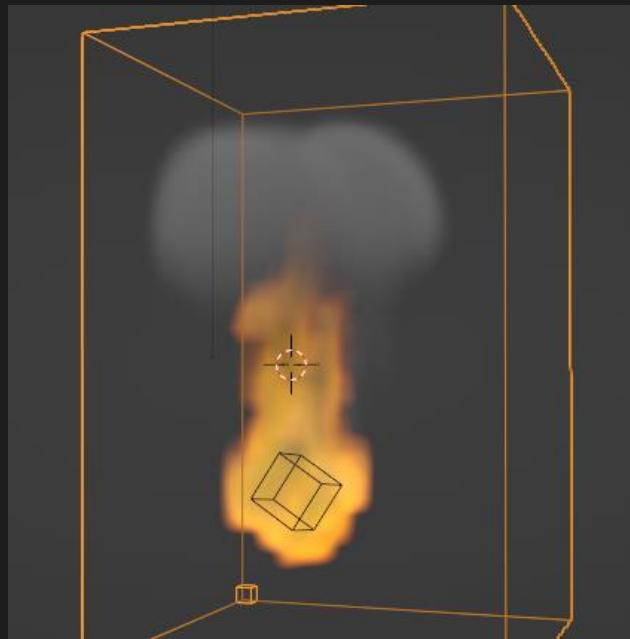
029강 Volume

Mesh Volume에 대한 개념
volume으로 장면과 물체에 깊이감을 만들기



Volumetrics

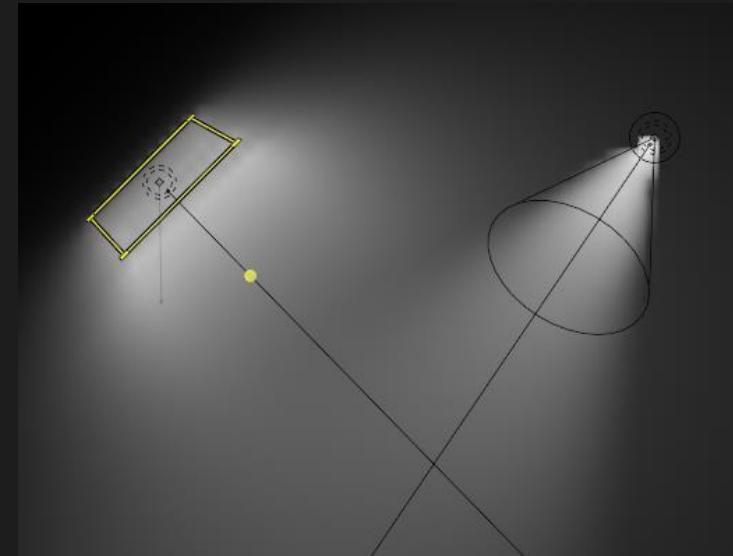
플루이드 시뮬레이션



오브젝트 볼륨 (Mesh Volume)

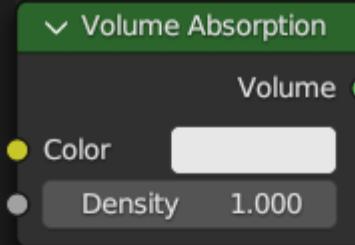


배경 볼륨

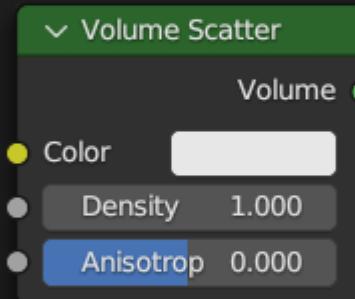


Volume Shader

Volume Scatter vs Volume Absortion

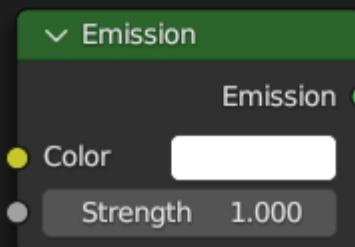


Volume Absorption : 지나간 거리에 따라 빛을 흡수합니다.
검은색으로 설정하면 모든 색의 빛을 흡수하고,
흰색으로 설정하면 빛을 흡수하지 않습니다.(작동하지 않습니다)



Volume Scatter : 볼륨을 지나가는 빛을 여러 방향으로 산란시킵니다.
흰색으로 설정하면 모든 색의 빛을 산란시키고, 검은색으로 설정하면 작동하지 않습니다.

Anisotropy : 산란할 방향을 지정합니다.
기본값 0 은 모든 방향으로 산란시키고, 1은 정방향, -1은 역방향으로 작동합니다.

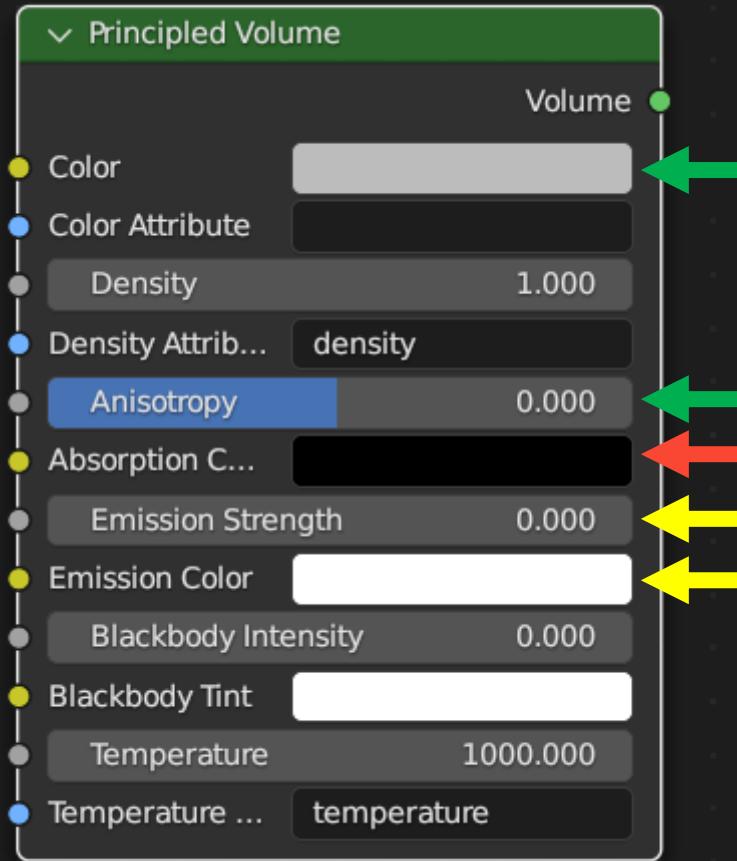


볼륨 셰이더로 Emission도 사용할 수 있습니다.

※Emission 외의 다른 셰이더들이 볼륨과 호환되는 것은 아닙니다. Emission은 특수하게 Surface와 Volume에 개별적으로 작동하도록 만들어져 있습니다.

Volume Shader

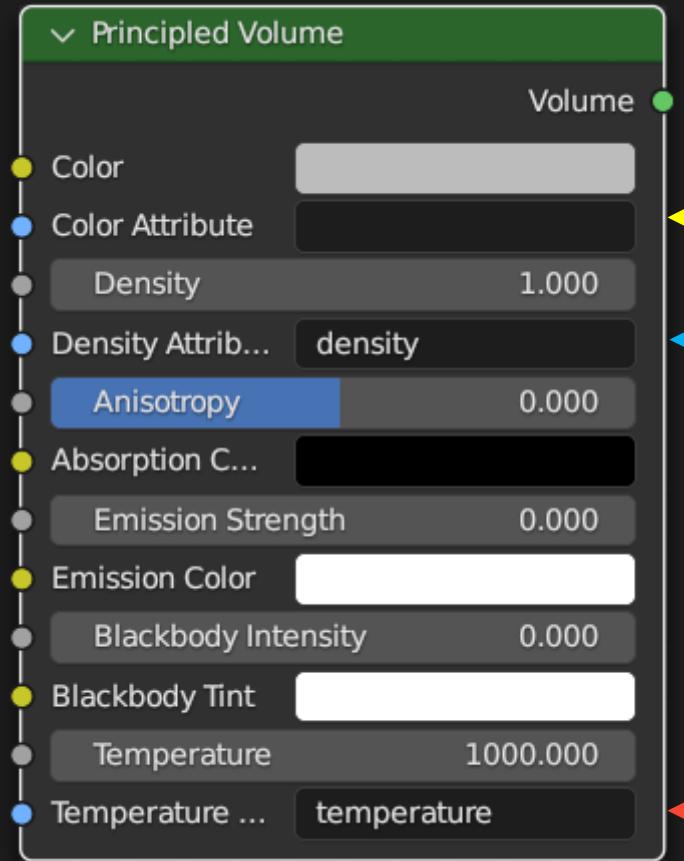
Principled Volume



앞에서 알아본 세가지를 합친 셰이더입니다.
Scatter, Absorption, Emission 수치를 확인해 보세요.

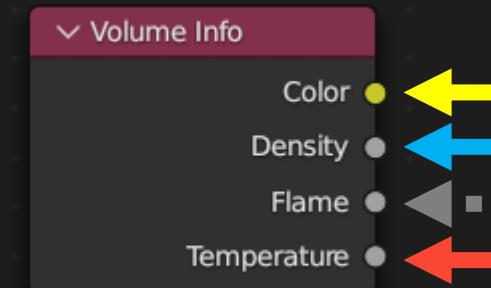
Volume Shader

Principled Volume



Principled Volume에는 시뮬레이션에서 만들어진 볼륨의 색, 밀도, 온도를 받아오는 창이 있습니다.
각각의 창에 알맞은 이름을 적어넣으면, 자동으로 시뮬레이션의 수치를 받아옵니다.

이 수치는 Volume Info 노드를 이용하여 받아올 수도 있습니다.

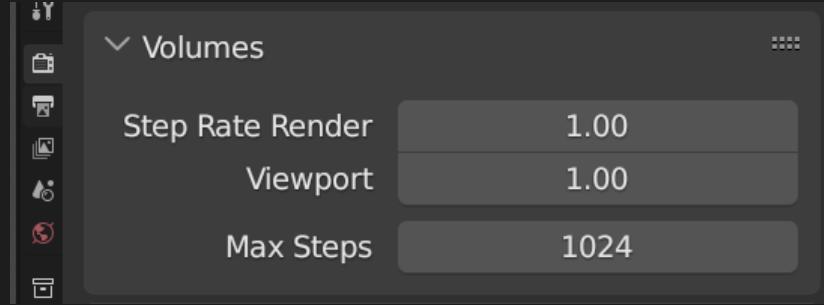


Principled Volume은 Temperature에 의해서
자동으로 불꽃 형태가 만들어지므로, Flame은 쓰지 않아도 됩니다

Temperature Attribute는 위의 Temperature 값과 곱해집니다.
만약 Temperature Attribute가 2라면 2000도의 온도가 됩니다.

렌더 세팅

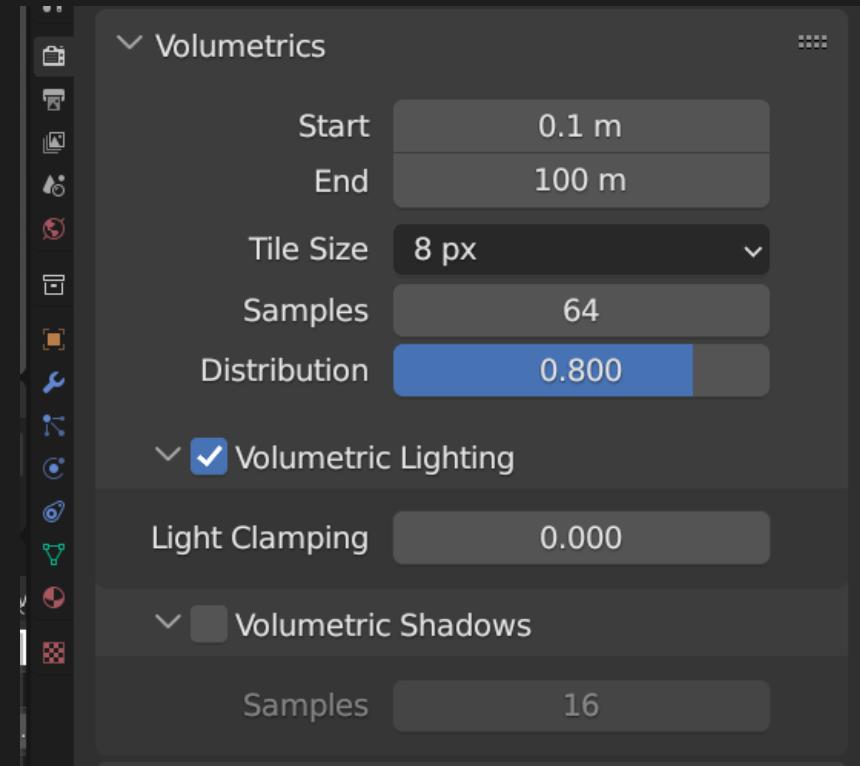
Cycles



Step Rate : 광선이 얼마나 움직일 때마다 연산이 일어날 지 결정합니다. 작은 Step Rate는 퍼포먼스를 저하시킵니다.

Max Steps : 연산이 최대 몇 번 일어날지 결정합니다.

Eevee

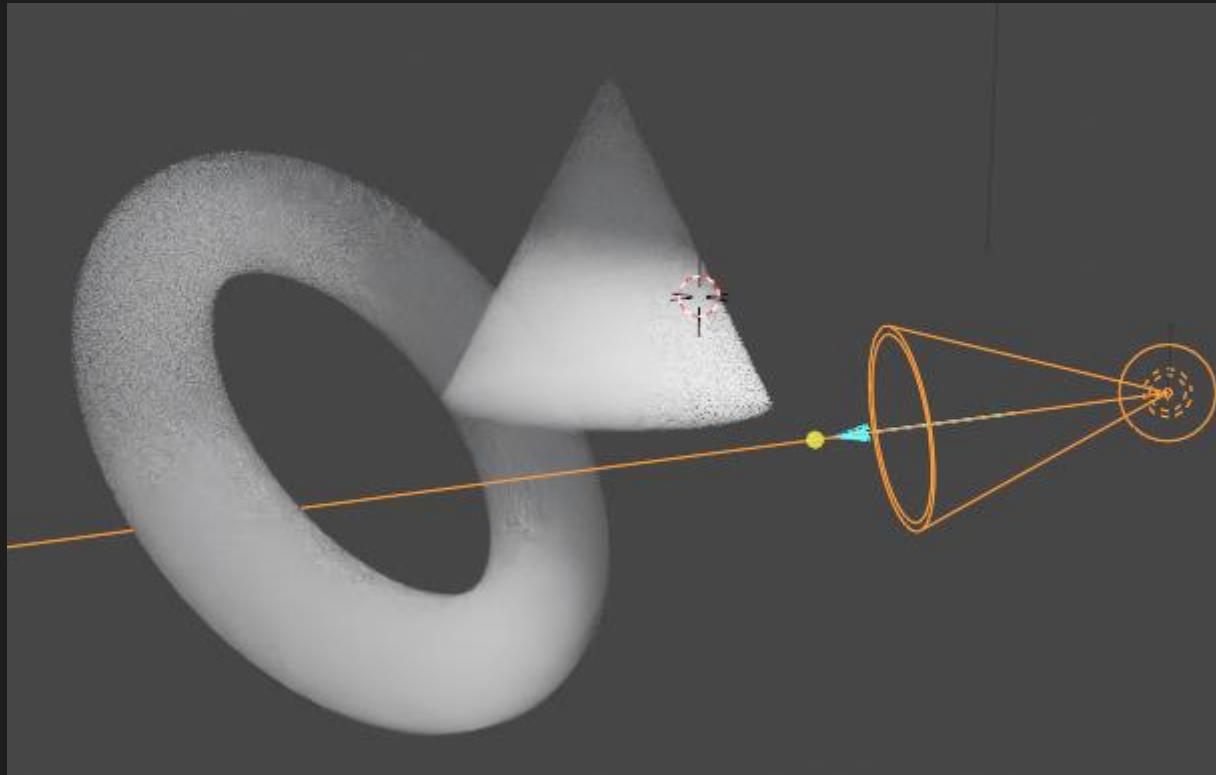


Eevee는 Ray Marching을 이용합니다.
쉽게 표현하자면, 여러 장의 이미지를 겹치는 방식으로 볼륨을 만들어냅니다.
Start – End : 볼륨이 표현될 카메라 거리
Tile size : 이미지 한장의 해상도
Samples : 이미지를 몇 장 사용할 것인지.
Distribution : 이미지를 카메라 거리에 따라 분배합니다.

오브젝트 볼륨

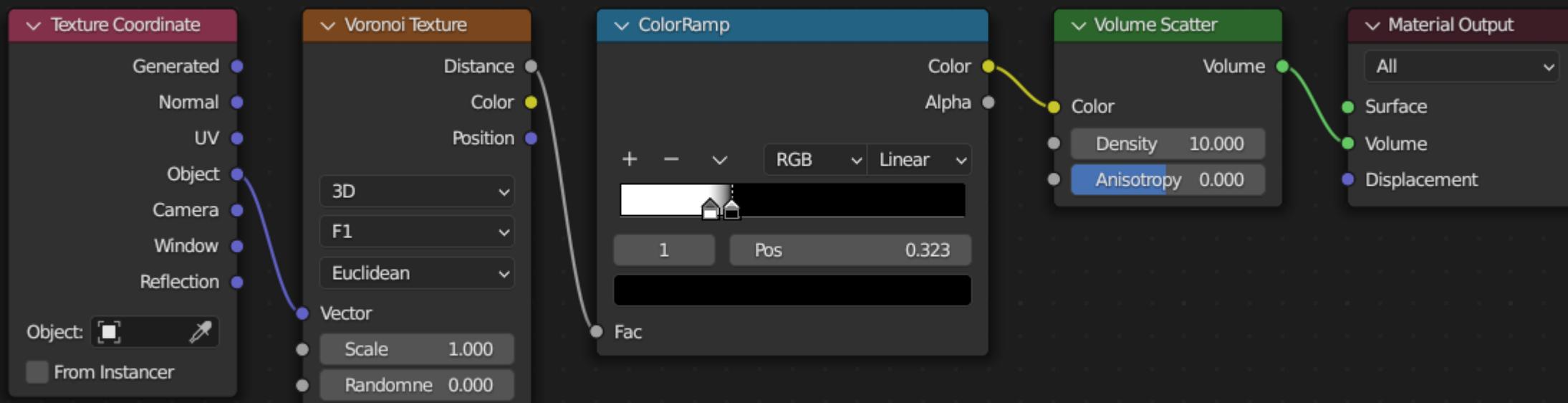
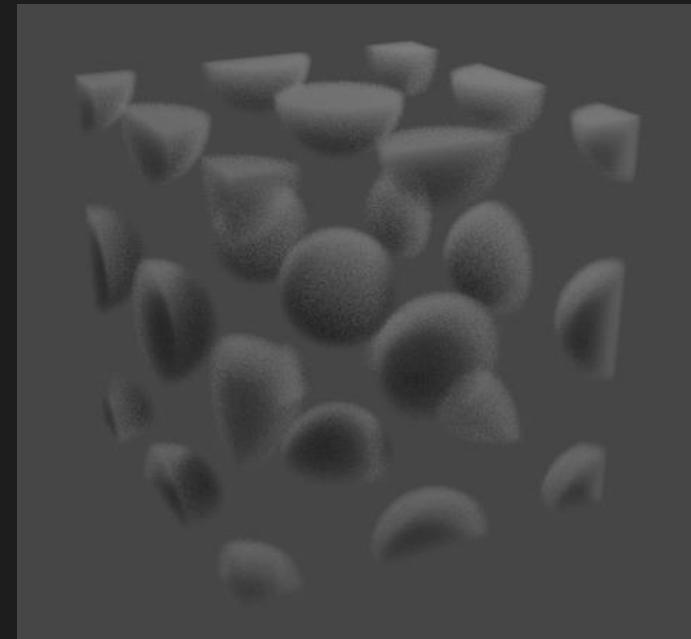
오브젝트에 볼륨 셰이더를 사용하면, 오브젝트 모양으로 볼륨이 생성됩니다.

단 Eevee는 오브젝트 모양을 따라가지 못하고 bounding box 형태로 볼륨을 만듭니다.



볼륨과 텍스쳐

Texture Coordinate의 3차원 좌표들은 볼륨에도 잘 대응됩니다. 이를 통해 여러가지 3차원 텍스쳐들을 볼륨에 사용할 수 있으며, Eevee에서도 문제없이 작동합니다.

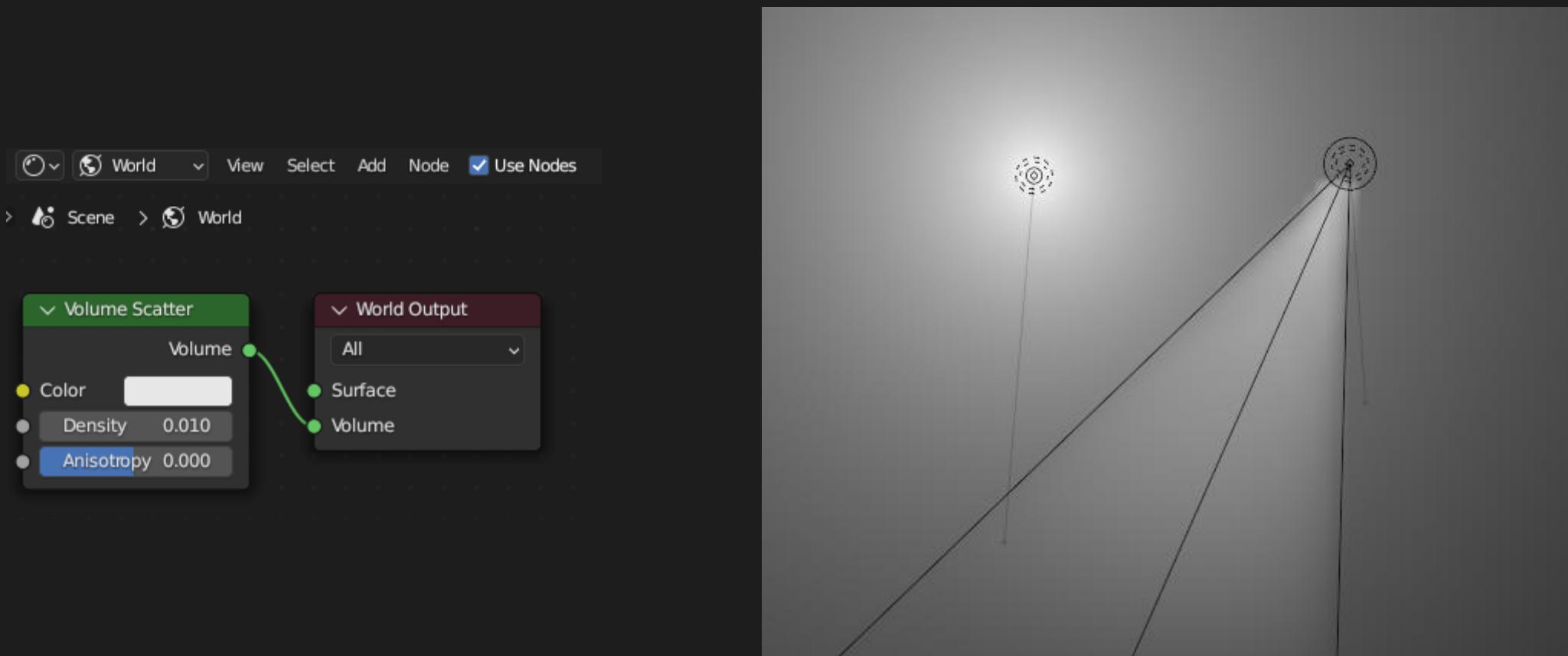


배경 볼륨

배경에 통째로 볼륨메트릭 셰이더를 사용할 수 있습니다.

볼륨 효과가 무한히 먼 곳까지 적용되기 때문에 Hdri가 제대로 보이지 않고,
Cycles의 경우엔 연산 속도가 극도로 느려질 수 있음에 유의하세요.

※Light Path 노드를 이용하여 Camera ray일 때만 밀도있게 하면 좀 더 빠른 연산을 할 수 있습니다.



030강 배경과 조명에서의 노드 사용

배경과 조명에서 사용하는 노드

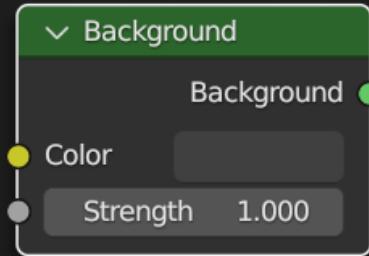
조명 좌표 변환

월드 좌표 변환으로 움직이는 하늘 만들기

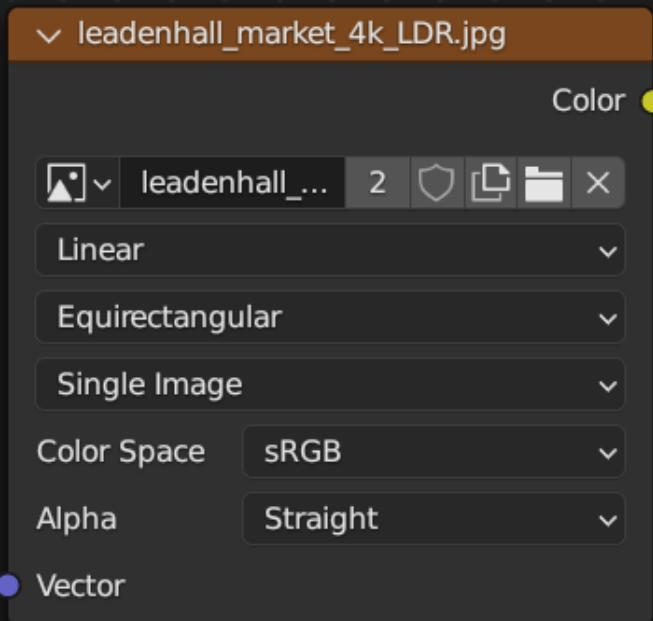


배경에서 쓸 수 있는 노드

배경에서 사용할 수 있는 노드가 몇 가지 있습니다.



Background : Emission과 비슷하게 작동합니다.
입력받은 이미지의 밝기를 조절합니다.

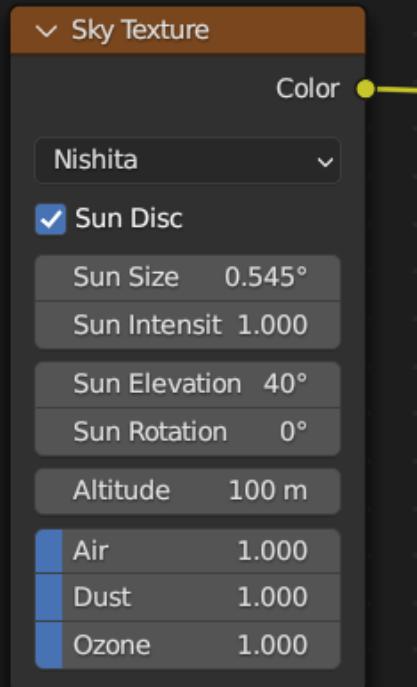


Environment Texture :
이미지 텍스쳐 노드와 비슷하지만, 입력받는 좌표와 사용하는 이미지의 종류 모두 다릅니다.

기본값으로 Equirectangular 이미지를 받지만, Mirror Ball 이미지도 사용 가능합니다.

배경에서 쓸 수 있는 노드

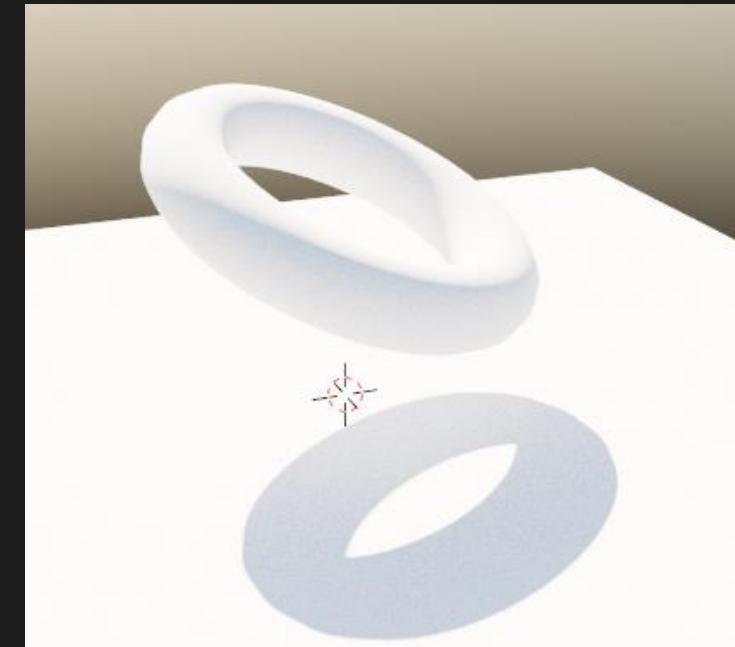
Sky Texture



하늘 배경을 생성합니다.

기본값은 Nishita이며, Preetham 등, 이전 버전도 선택 가능하지만 추천하지 않습니다.

Eevee에서도 작동하지만 태양은 생기지 않습니다.
(Eevee에서는 Hdri를 통한 그림자가 생기지 않으므로 어차피 소용 없음.)



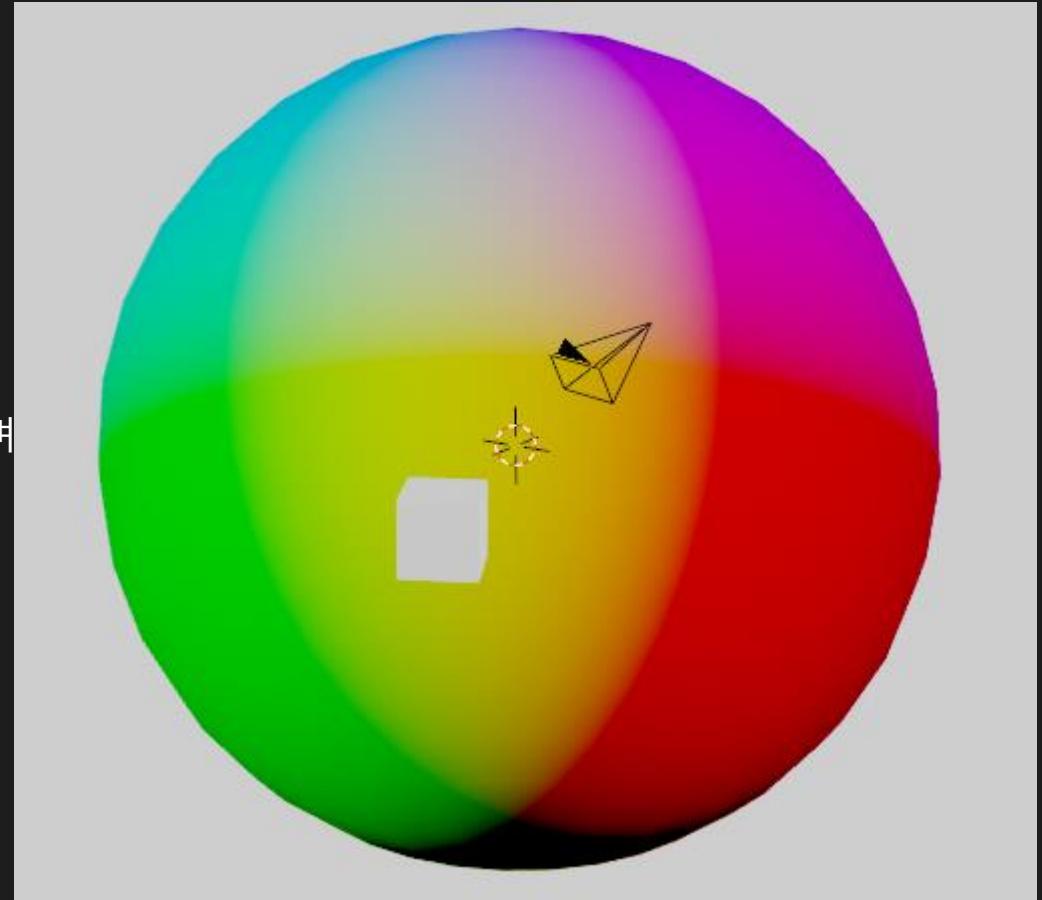
배경 좌표

일반적으로 사용하는 배경 좌표는 방향만 존재하는 출력입니다.

구면 위의 점을 나타낸 것과 비슷하여, 크기는 항상 1입니다.

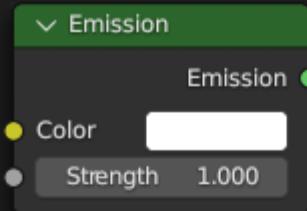
이러한 특수성 때문에 일반적인 image Texture 노드는 정상적으로 작동하지 않으며

Environment texture를 써야 좌표를 제대로 해석해 이미지를 배경에 매핑합니다.



조명에서 쓸 수 있는 노드 (Cycles only)

Cycles일 때만 조명에서 노드를 사용할 수 있습니다.

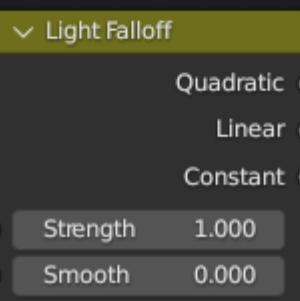


Emission : 조명에서는 Emission만 사용할 수 있습니다.



Wavelength, Blackbody : 노드 자체는 조명이 아니어도 사용할 수 있지만, 그 특성상 조명에서 많이 유용합니다.

Wavelength는 파장에 따른 색을, Blackbody는 온도에 따른 색을 출력합니다.



Light Falloff : 일반적으로 거리 제곱에 반비례하는 빛의 밝기 특성을 변경합니다.

Quadratic : 빛의 세기가 거리 제곱에 반비례합니다. (이는 Light Falloff을 사용하지 않았을 때의 기본값입니다.)

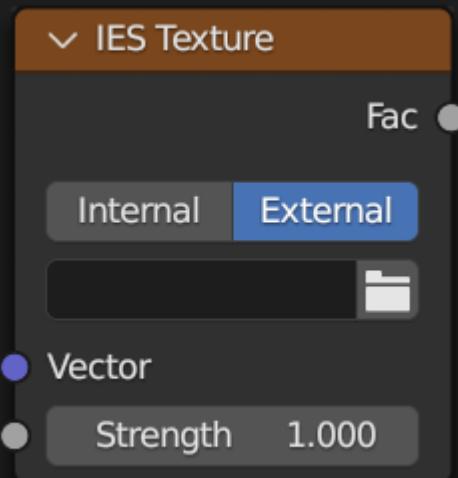
Linear : 빛의 세기가 거리에 반비례합니다. 빛은 더 멀리 퍼집니다.

Constant : 빛의 세기가 거리에 영향을 받지 않습니다. (이는 Sun light의 기본값입니다.)

Smooth : 가까운 거리에서의 밝기를 줄입니다.

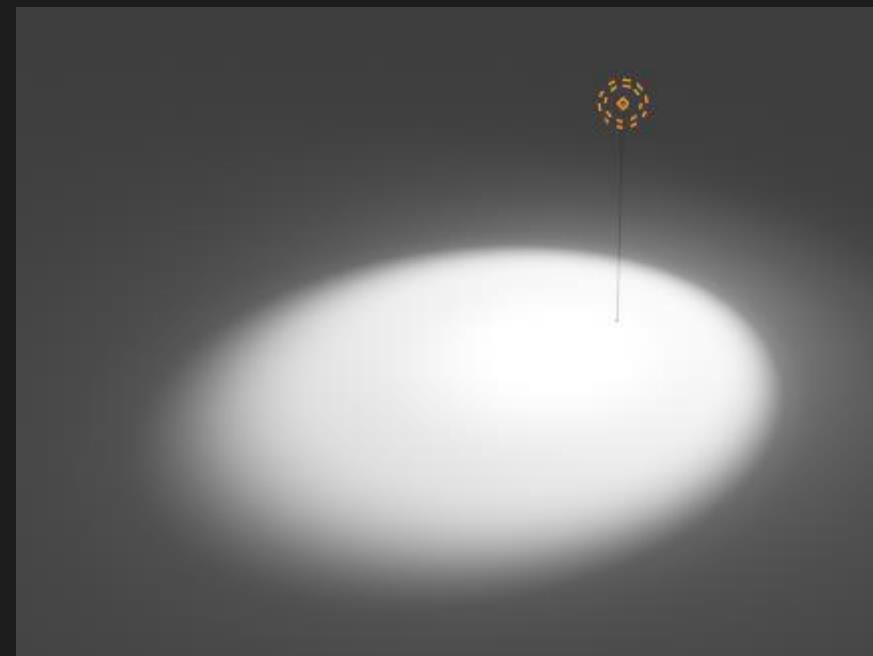
IES Texture

<https://ieslibrary.com/>



조명이 방출하는 빛의 분포 형태를 담고 있는
IES 파일을 사용할 수 있습니다.

조명의 종류에 따라 어떻게 빛이 퍼지는지를 나타내며
스팟 조명을 선택하지 않아도 자동으로 한쪽으로만 빛을 내보냅니다.

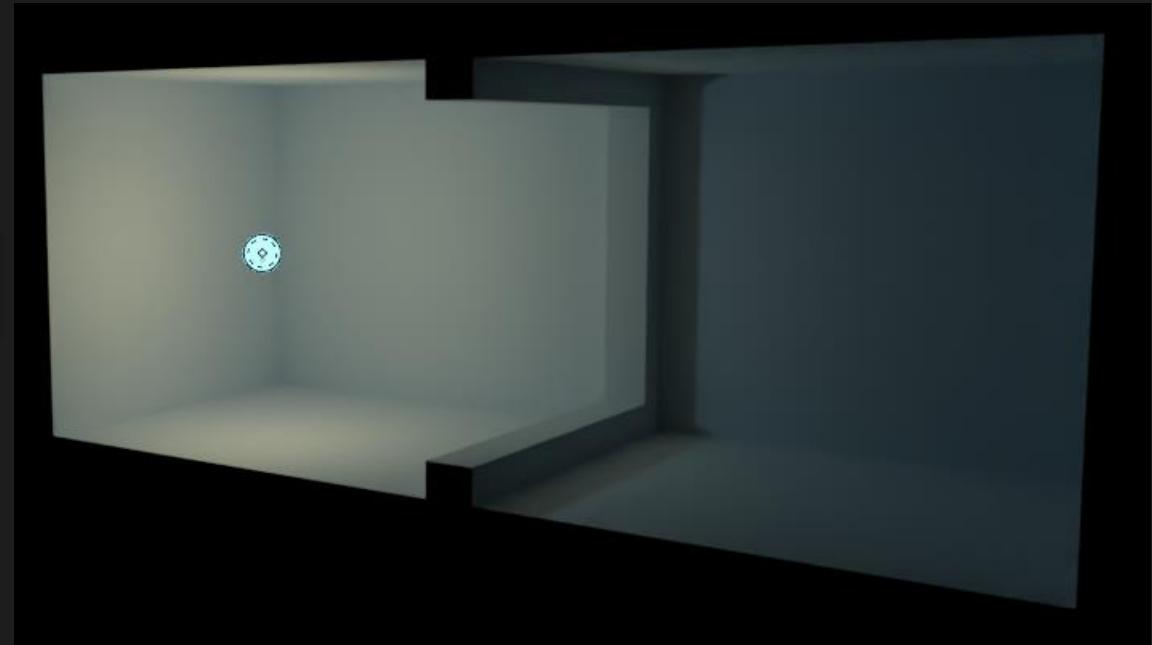
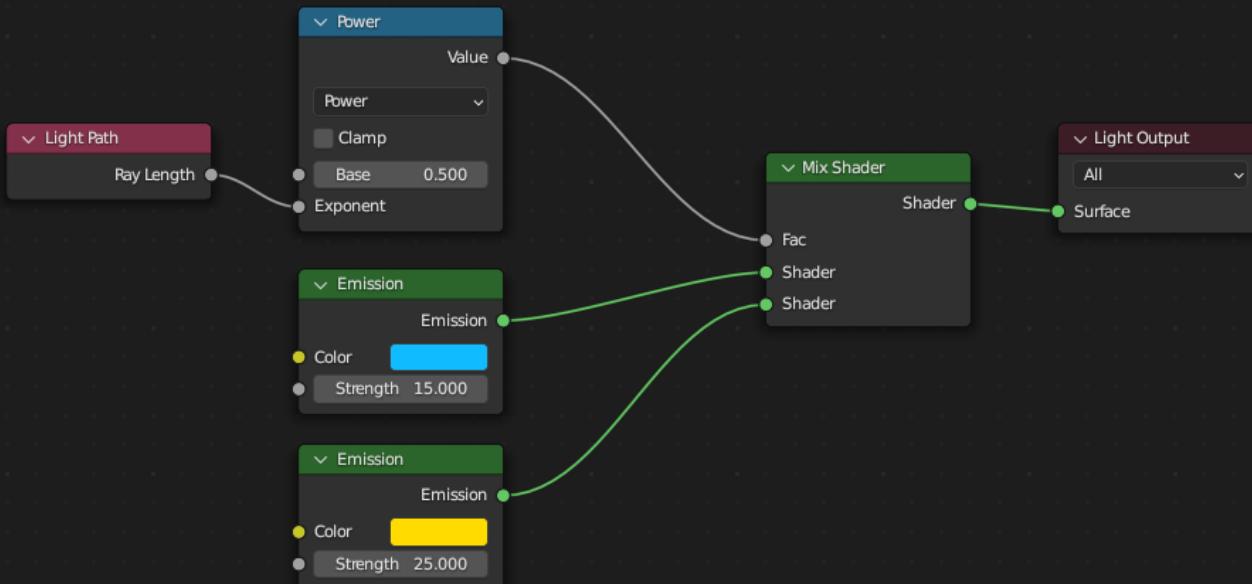


조명에서의 Light Path

배경에서는 Ray Length를 제외하곤 자유롭게 쓸 수 있는 반면,

조명에서는 Ray Length, Ray depth만 정상 작동합니다.

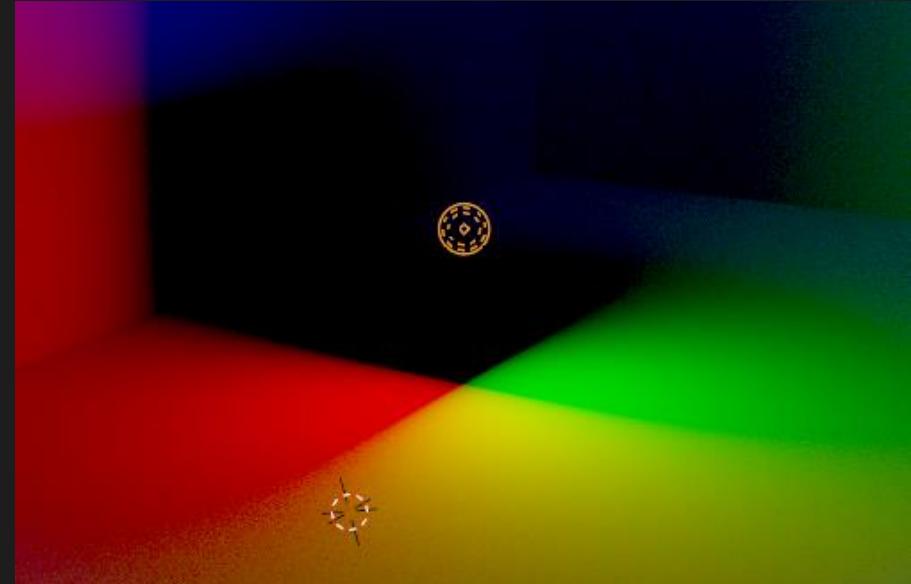
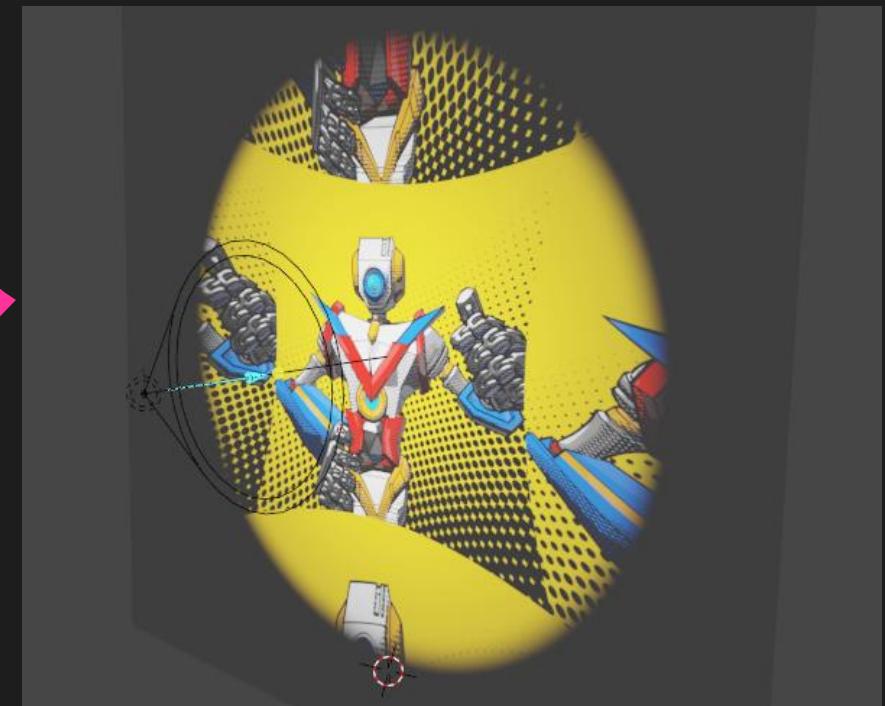
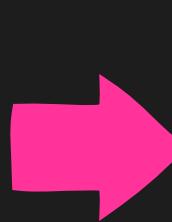
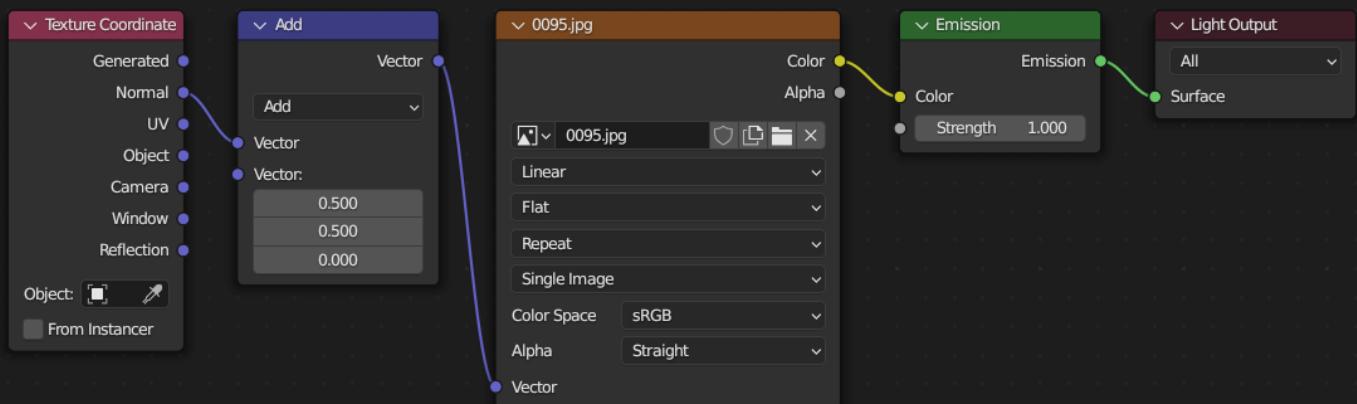
Ray Length를 이용할 때는 Power의 Exponent 연결이 도움됩니다.



조명의 텍스쳐좌표

조명도 배경처럼 텍스쳐 좌표가 방향을 출력합니다.
(따라서, hdri를 꽂으면 잘 작동합니다..)

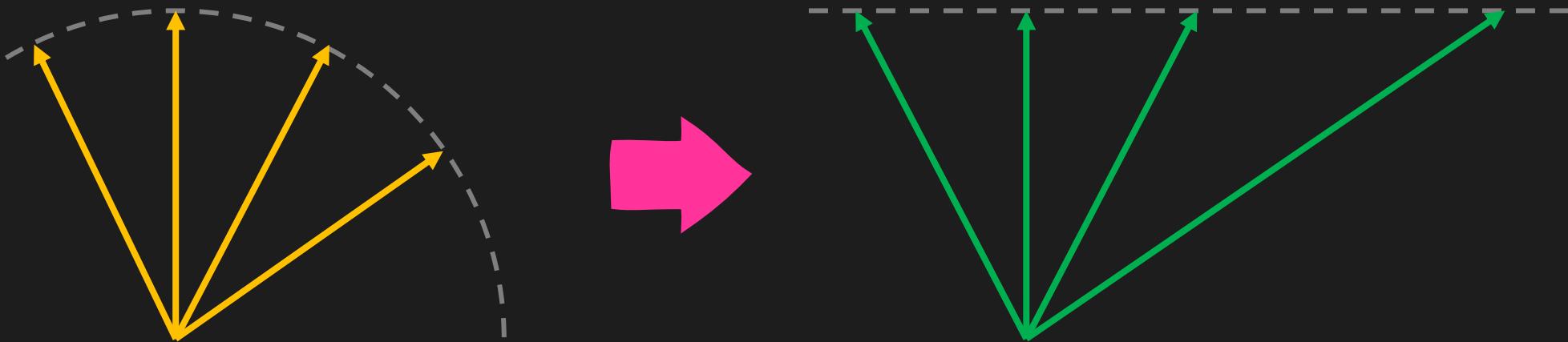
이 출력은 좁은 범위에서는 보통의 좌표처럼 보이나
범위가 넓어지면 왜곡이 발생합니다.
이를 해결하기 위해서는 좌표를 수정해야 합니다.



좌표 변환

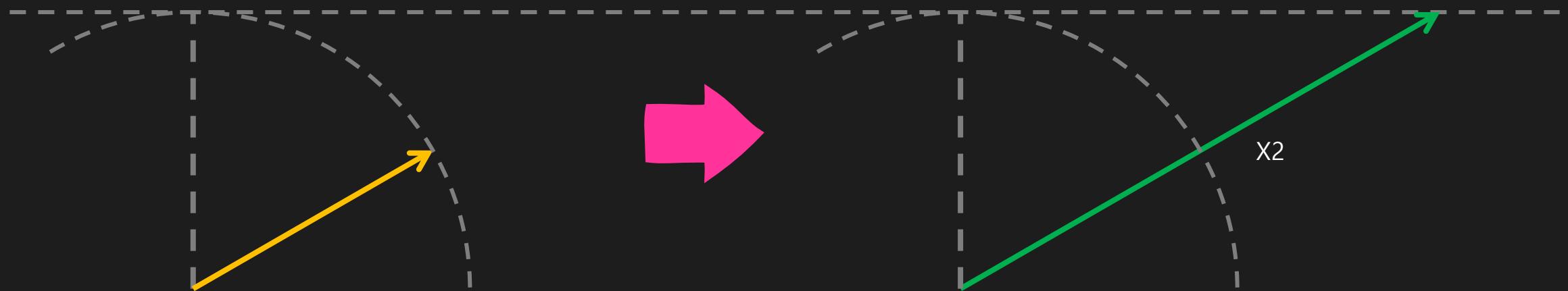
이 부분은 노드 연결만 알아두셔도 괜찮습니다.

우리가 하고자 하는 건 구형 좌표를 평면으로 펼치려는 것입니다.



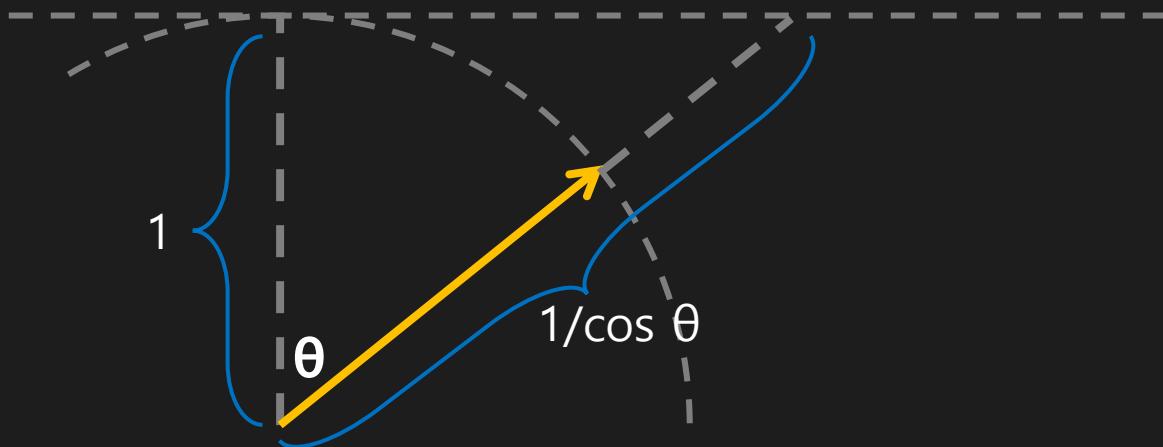
좌표 변환

예를 들어 이 경우 원래 좌표에 2배 정도 곱해줘야 됩니다.



좌표 변환

구체적으로, 각도 θ 만큼 기울어진 벡터는, $1/\cos \theta$ 배 커져야 합니다.

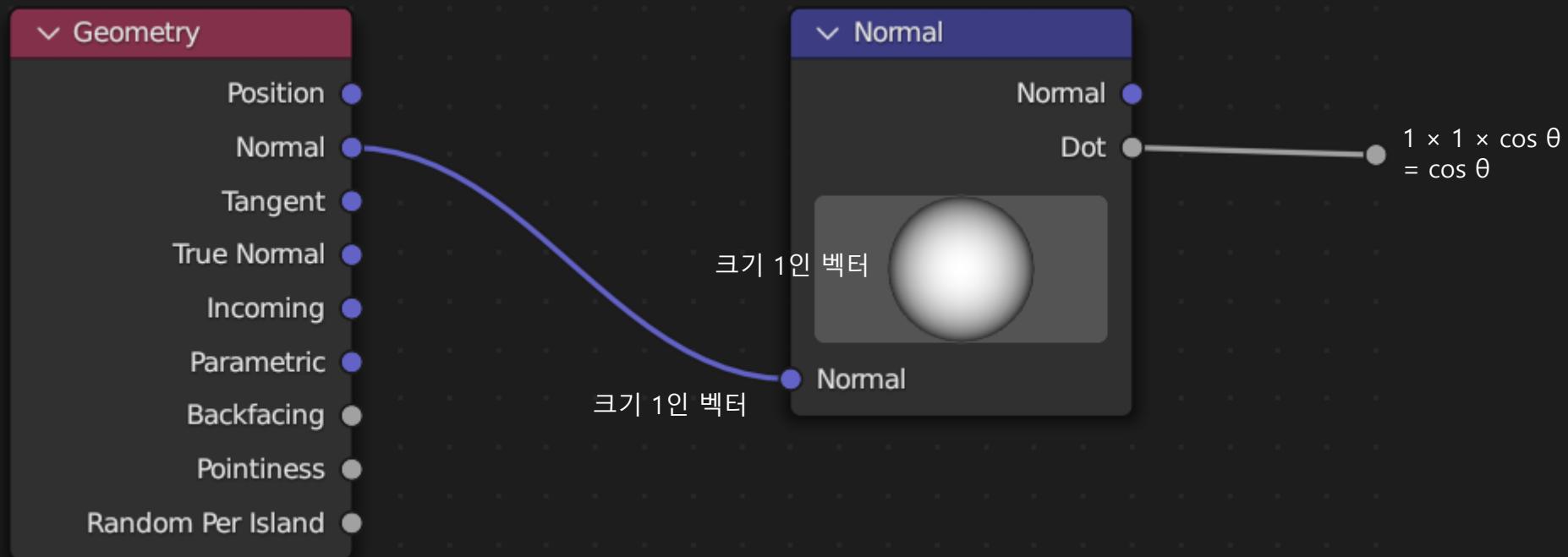


즉 θ 를 구하면 문제를 풀 수 있습니다.

좌표 변환

$\cos \theta$ 는 벡터의 내적에서 구할 수 있습니다. 내적은 두 벡터의 크기에 코사인을 곱한 값이기 때문입니다.

그런데 우리의 계산에서 사용하는 벡터는 모두 크기 1이므로, 내적 그 자체로 코사인 값이 됩니다.



좌표 변환

최종 연결은 다음과 같습니다.

※배경 좌표도 같은 원리이므로, 이와 동일하게 사용할 수 있습니다!

