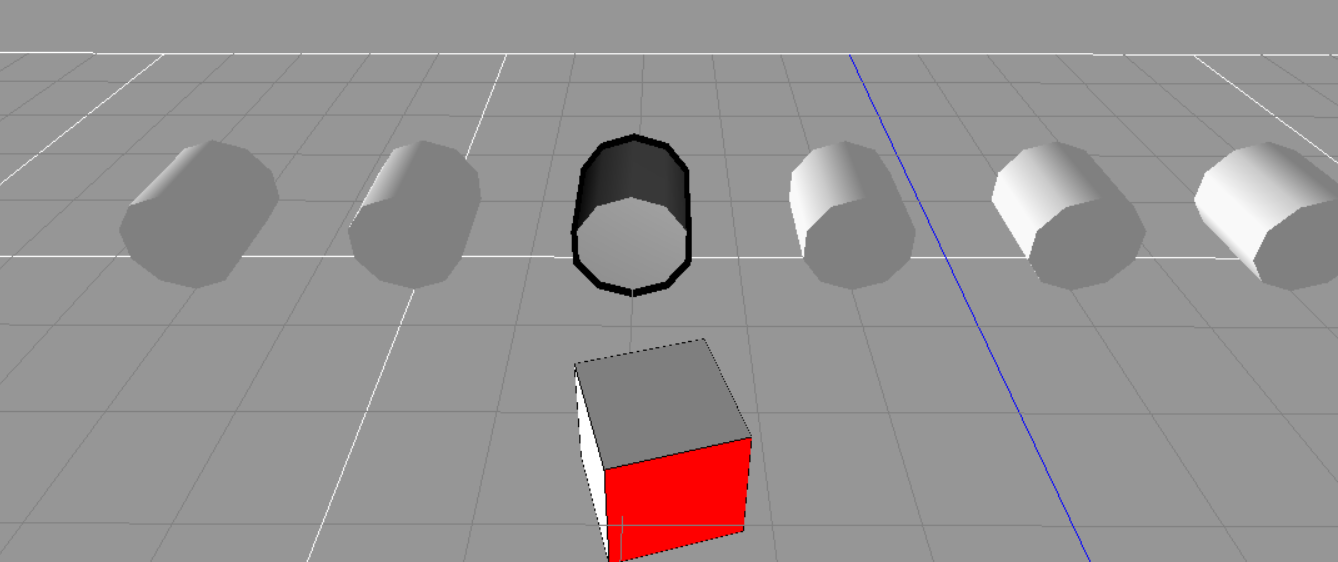
[ 게임 그래픽 프로그래밍 ]

**셰이더 언어 이해 및 코드 작성**

인하대학교 미래인재개발원 문화컨텐츠개발

김예슬

**셰이더가 적용된 스크린 샷**



**적용된 셰이더 코드**

float4x4 matViewProjection;

float4 OutlineColor;

float OutlineWidth;

struct VS\_INPUT

{

float4 Position : POSITION;

};

struct VS\_OUTPUT

{

float4 Position : POSITION;

float4 Color : COLOR;

};

float4 CreateOutLine(float4 vertex, float outLine)

{

float4x4 scaleMat;

scaleMat[0][0] = 1.0f + outLine;

scaleMat[0][1] = 0.0f;

scaleMat[0][2] = 0.0f;

scaleMat[0][3] = 0.0f;

scaleMat[1][0] = 0.0f;

scaleMat[1][1] = 1.0f + outLine;

scaleMat[1][2] = 0.0f;

scaleMat[1][3] = 0.0f;

scaleMat[2][0] = 0.0f;

scaleMat[2][1] = 0.0f;

scaleMat[2][2] = 1.0f + outLine;

scaleMat[2][3] = 0.0f;

scaleMat[3][0] = 0.0f;

scaleMat[3][1] = 0.0f;

scaleMat[3][2] = 0.0f;

scaleMat[3][3] = 1.0f;

return mul(scaleMat, vertex);

}

VS\_OUTPUT VS( VS\_INPUT Input )

{

VS\_OUTPUT Output;

Output.Position = mul(CreateOutLine(Input.Position, OutlineWidth), matViewProjection);

// Output.Position = UnityObjectToClipPos(CreateOutline(v.vertex, \_Outline));

Output.Color = OutlineColor;

return Output;

}

struct PS\_INPUT

{

float4 Color : COLOR0;

};

float4 PS(PS\_INPUT Input) : COLOR0

{

return Input.Color;

}

//--------------------------------------------------------------//

// Pass 1

//--------------------------------------------------------------//

// float4x4 gWorldMatrix;

// float4x4 gViewMatrix;

// float4x4 gProjectionMatrix;

// // >> 컬러 쉐이더

float4x4 gWorldMatrix;

float4x4 gViewMatrix;

float4x4 gProjectionMatrix;

float4 gWorldLightPos;

float4 gWorldCameraPos : ViewPosition;

// >> 라이트 쉐이더

struct VS\_INPUT2

{

float4 mPosition : POSITION0;

float3 mNormal : NORMAL;

};

struct VS\_OUTPUT2

{

// float4 Position : POSITION;

// // >> 컬러 쉐이더

float4 mPosition : POSITION0;

float3 mDiffuse : TEXCOORD1;

float3 mViewDir : TEXCOORD2;

float3 mReflection : TEXCOORD3;

// >> 라이트 쉐이더

};

VS\_OUTPUT2 VS2( VS\_INPUT2 Input )

{

// VS\_OUTPUT2 Output;

// Output.Position = mul(Input.Position, gWorldMatrix);

// Output.Position = mul(Output.Position, gViewMatrix);

// Output.Position = mul(Output.Position, gProjectionMatrix);

//

// return Output;

// // >> 컬러 쉐이더

VS\_OUTPUT2 Output;

Output.mPosition = mul(Input.mPosition, gWorldMatrix);

float3 lightDir = Output.mPosition.xyz - gWorldLightPos.xyz;

lightDir = normalize(lightDir);

float3 viewDir = normalize(Output.mPosition.xyz - gWorldCameraPos.xyz);

Output.mViewDir = viewDir;

Output.mPosition = mul(Output.mPosition, gViewMatrix);

Output.mPosition = mul(Output.mPosition, gProjectionMatrix);

float3 worldNormal = mul(Input.mNormal, (float3x3)gWorldMatrix);

worldNormal = normalize(worldNormal);

Output.mDiffuse = dot(-lightDir, worldNormal);

Output.mReflection = reflect(lightDir, worldNormal);

return Output;

// >> 라이트 쉐이더

/\*VS\_OUTPUT2 Output;

Output.Position = mul(Input.Position, matViewProjection);

return Output;\*/

// >> 원본

}

float4 SurfaceColor;

float4 gLightColor;

struct PS\_INPUT2

{

float3 mDiffuse : TEXCOORD1;

float3 mViewDir : TEXCOORD2;

float3 mReflection : TEXCOORD3;

// >> 라이트 쉐이더

};

float4 PS2(PS\_INPUT2 Input) : COLOR

{

//return SurfaceColor;

//// >> 컬러 쉐이더

float3 diffuse = gLightColor.rgb \* SurfaceColor.rgb \* saturate(Input.mDiffuse);

float3 refelction = normalize(Input.mReflection);

float3 viewDir = normalize(Input.mViewDir);

float3 specular = 0;

if (diffuse.x > 0)

{

specular = saturate(dot(refelction, -viewDir));

specular = pow(specular, 20.0f);

specular \*= SurfaceColor.rgb \* gLightColor.rgb;

}

float3 ambient = float3(0.1f, 0.1f, 0.1f) \* SurfaceColor;

return float4(ambient + diffuse + specular , 1.0f);

// >> 라이트 쉐이더

}

//--------------------------------------------------------------//

// Technique Section for Default\_DirectX\_Effect

//--------------------------------------------------------------//

technique Default\_DirectX\_Effect

{

pass Pass\_0

{

VertexShader = compile vs\_2\_0 VS();

PixelShader = compile ps\_2\_0 PS();

}

pass Pass\_1

{

VertexShader = compile vs\_2\_0 VS2();

PixelShader = compile ps\_2\_0 PS2();

}

}