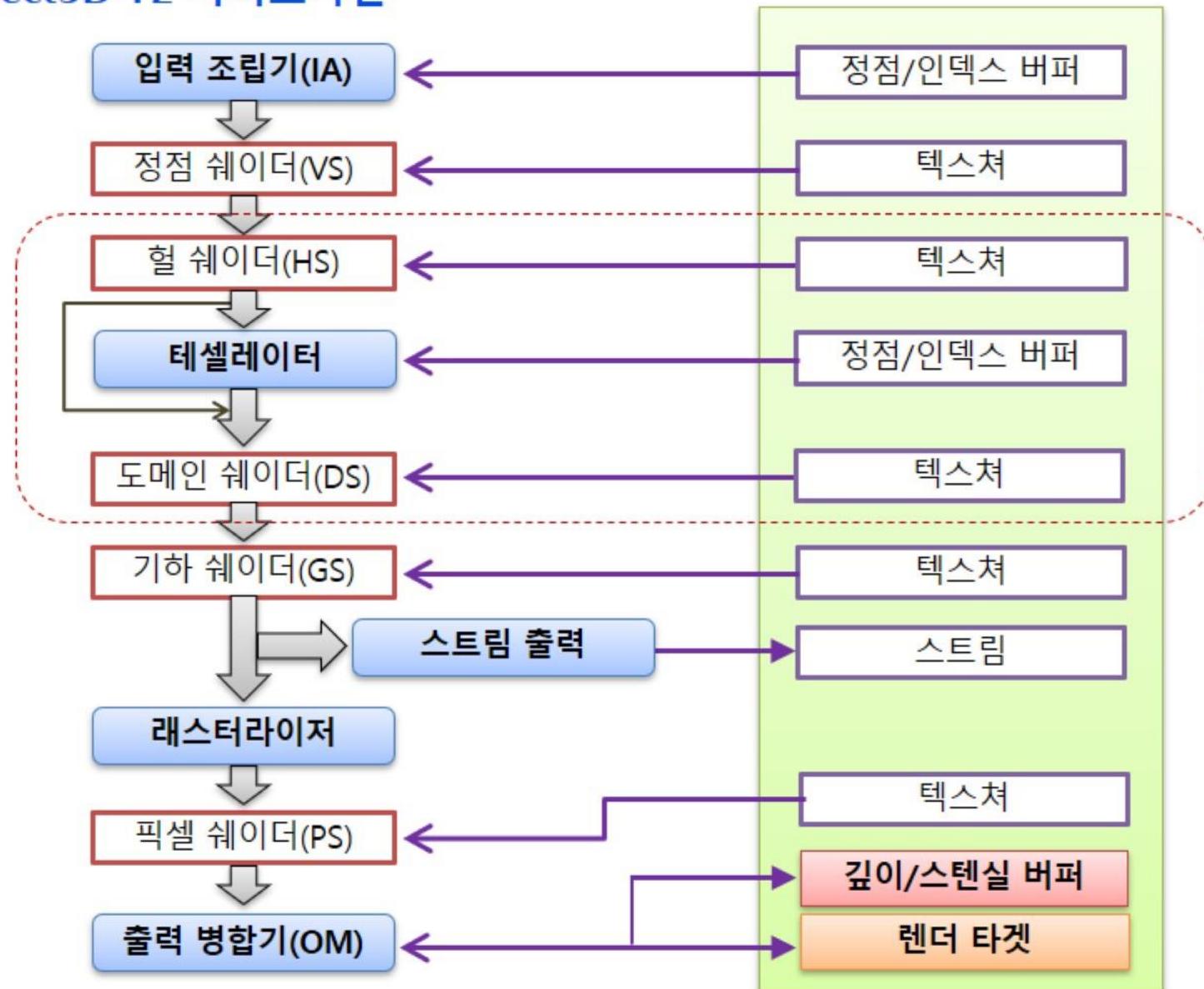


Game Programming with DirectX

Direct3D Graphics Pipeline (Tessellation)

Direct3D 파이프라인

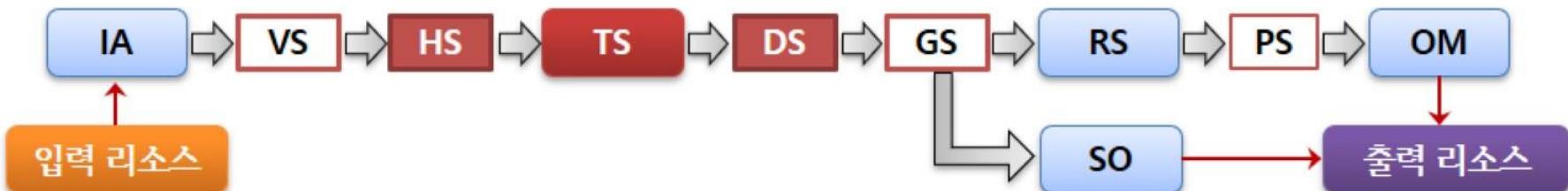
- Direct3D 12 파이프라인



Direct3D 파이프라인

• 테셀레이션(Tessellation)

- Direct3D 12는 GPU에서 테셀레이션을 위한 3가지 단계를 제공
 - 쉐이더 모델 5(hs_5_1, ds_5_1)
 - 헐-쉐이더(Hull-Shader) 단계
입력 제어점으로 부터 출력 제어점과 패치 상수를 생성
 - 테셀레이션(Tessellation) 단계
도메인(사각형, 삼각형, 선분)을 작고 많은 프리미티브로 분할
 - 도메인-쉐이더(Domain-Shader) 단계
분할된 점의 정점 위치를 계산
- 테셀레이션은 적은 다각형으로 구성된 표면을 아주 세밀한 프리미티브로 바꿈
 - 메모리와 대역폭(Bandwidth)를 줄임
 - 동적 LOD(Dynamic LOD)
 - 덜 세밀한(더 적은 다각형으로 구성된) 모델에서 계산을 수행할 수 있음
블렌딩, 모핑(Morphing), 충돌 검사를 위한 물리 계산 등
- 테셀레이션은 사각형 패치(Patch), 삼각형 패치 또는 등치선(Isoline)으로 부터 생성된 표면에서 더욱 세밀한 표면을 GPU에서 계산함
- 고차(High-Ordered) 표면을 근사하기 위하여 각 패치는 테셀레이션 인자를 사용하여 삼각형, 점 또는 선분들로 분할됨



Direct3D 파이프라인

▪ 헬-쉐이더(Hull-Shader) 단계

- 헬-쉐이더는 입력 패치(Patch)마다 한번씩 호출됨
- 입력 제어점(Control Point)을 출력 제어점으로 변환(Transform)



- 제어점 단계
각 입력 제어점에 대하여 한 번 실행
- 패치 상수 단계
에지 테셀레이션 인자와 패치 상수를 생성하기 위하여 패치마다 한 번 실행
- 쉐이더 입력: 1 ~ 32개의 제어점
- 쉐이더 출력: 1 ~ 32개의 제어점
헬-쉐이더의 출력 제어점은 도메인-쉐이더 단계에서 사용할 수 있음
- 테셀레이션 인자(Factor)는 각 패치를 얼마만큼 분할하는 가를 결정
테셀레이션 인자는 도메인-쉐이더와 테셀레이션 단계에서 사용할 수 있음
- 쉐이더는 테셀레이션 단계에서 필요한 상태를 선언함
제어점의 개수, 패치 면의 유형, 테셀레이션을 할 때 사용할 분할의 유형
- 에지 테셀레이션 팩터를 0 또는 음수, NaN로 설정하면 패치는 컬링됨
결과적으로 테셀레이터 단계와 도메인-쉐이더 단계는 실행되지 않을 것임
컬링되는 패치에 대한 출력은 생성되지 않음

Direct3D 파이프라인

▪ 테셀레이터(Tessellator) 단계

- 테셀레이터 단계는 고정 파이프라인 단계임
- 헐-쉐이더를 파이프라인에 연결하면 테셀레이터 단계가 자동적으로 초기화
테셀레이터 단계에 상태 객체를 설정하지 않음(헐-쉐이더의 설정을 그대로 사용)
- 테셀레이터 단계는 하나의 도메인(사각형, 삼각형, 선분)을 여러 개의 작은 삼각형, 점, 또는 선분들로 분할하는 것이 목적임
- 테셀레이터는 정규화된 좌표계(0.0~1.0)를 사용
- 테셀레이터는 패치(Patch)마다 한번 실행됨
테셀레이션 인자: 도메인을 얼마나 세밀하게 분할하는 가를 설정
분할의 유형: 하나의 패치를 분할하는 알고리즘을 명시

Fractional_odd	테셀레이션 인자 범위: [1..63] 홀수
Fractional_even	테셀레이션 인자 범위: [2..64] 짝수
Integer	테셀레이션 인자 범위: [1..64] 정수
Pow2	테셀레이션 인자 범위: [1..64] 2의 거듭제곱

- 테셀레이터는 $uv[w]$ 좌표와 표면 토텔로지(Topology)를 출력
 - 단계 1
32-비트 부동-소수점 연산을 사용하여 테셀레이션 인자를 처리
 - 단계 2
분할의 유형에 따라 점 또는 토텔로지 리스트를 생성
고정-소수점 연산으로 16-비트 소수점을 사용

Direct3D 파이프라인

▪ 도메인-쉐이더(Domain-Shader) 단계

- 도메인 쉐이더는 헐 쉐이더의 출력 제어점과 uv 좌표를 사용하여 표면 메쉬를 생성
- 출력 패치에서 분할된 하나의 점의 정점 위치를 계산
- 도메인 쉐이더는 테셀레이터 단계에서 생성한 출력 좌표마다 한번 호출됨
- 도메인 쉐이더는 헐-쉐이더 단계의 출력 제어점을 사용함
- 도메인 쉐이더는 하나의 정점을 출력함(SV_Position 시멘틱을 포함해야 함)
- 도메인 쉐이더의 입력은 헐-쉐이더 출력
 출력 제어점, 패치 상수 데이터, 테셀레이션 인자
 테셀레이션 인자는 원래의 값 뿐 아니라 테셀레이터가 사용한 값을 포함



- 테셀레이션이 활성화되면 인접성을 갖는 프리미티브를 입력으로 사용하는 기하 쉐이더는 사용할 수 없음
- 테셀레이션을 비활성화하려면 헐-쉐이더와 도메인 쉐이더를 NULL로 설정
- 테셀레이션이 활성화되면 입력-조립 단계의 입력은 패치(Patch) 데이터이어야 함
 IA 단계에 삼각형이 아닌 여러 개(1~32)의 제어점을 포함하는 패치를 전달

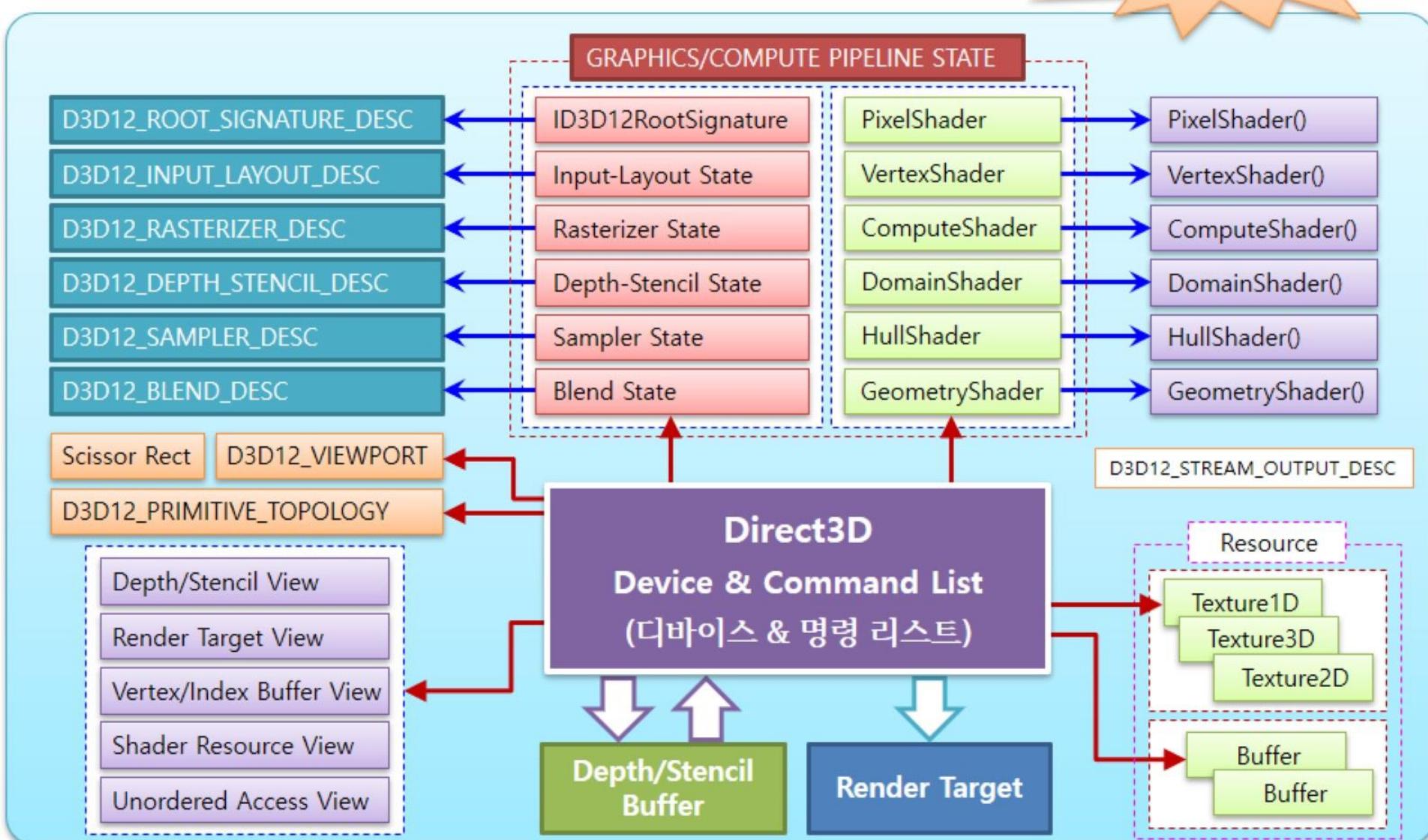
D3D_PRIMITIVE_TOPOLOGY_1(32)_CONTROL_POINT_PATCHLIST
 삼각형은 3개의 제어점을 가진 삼각형 패치로 간주할 수 있음

Direct3D 파이프라인

- Direct3D 디바이스

- Direct3D 디바이스는 **상태 기계(State Machine)**이다.

Set & Draw



Direct3D 파이프라인

- **ID3D12Device**

- 가상 어댑터(Virtual Adapter)를 나타내는 인터페이스

CheckFeatureSupport

CopyDescriptors	CopyDescriptorsSimple	
CreateCommandAllocator	CreateCommandList	CreateCommandQueue
CreateCommandSignature	CreateCommittedResource	CreateComputePipelineState
CreateConstantBufferView	CreateDepthStencilView	CreateDescriptorHeap
CreateFence	CreateGraphicsPipelineState	CreateHeap
CreatePlacedResource	CreateQueryHeap	CreateRenderTargetView
CreateReservedResource	CreateRootSignature	CreateSampler
CreateShaderResourceView	CreateSharedHandle	CreateUnorderedAccessView
Evict		
GetAdapterLuid	GetCopyableFootprints	GetCustomHeapProperties
GetDeviceRemovedReason	GetNodeCount	GetDescriptorHandleIncrementSize
GetResourceAllocationInfo	GetResourceTiling	
MakeResident		
OpenSharedHandle	OpenSharedHandleByName	
SetStablePowerState		

Direct3D 파이프라인

- **ID3D12GraphicsCommandList 인터페이스**

BeginEvent	BeginQuery	
ClearDepthStencilView	ClearRenderTargetView	
ClearState	ClearUnorderedAccessViewFloat	ClearUnorderedAccessViewUint
Close		
CopyBufferRegion	CopyResource	CopyTextureRegion
CopyTiles		
DiscardResource		
Dispatch		
DrawIndexedInstanced	DrawInstanced	
EndEvent	EndQuery	
ExecuteBundle	ExecuteIndirect	
IASetIndexBuffer	IASetPrimitiveTopology	IASetVertexBuffers
OMSetBlendFactor	OMSetRenderTargets	OMSetStencilRef
Reset		
ResolveQueryData	ResolveSubresource	
ResourceBarrier		
RSSetScissorRects	RSSetViewports	SetComputeRoot32BitConstant
SetComputeRoot32BitConstants	SetComputeRootConstantBufferView	
SetComputeRootDescriptorTable	SetComputeRootShaderResourceView	
SetComputeRootSignature	SetComputeRootUnorderedAccessView	
SetDescriptorHeaps		
SetGraphicsRoot32BitConstant	SetGraphicsRoot32BitConstants	SetGraphicsRootConstantBufferView
SetGraphicsRootDescriptorTable	SetGraphicsRootShaderResourceView	
SetGraphicsRootSignature	SetGraphicsRootUnorderedAccessView	
SetMarker	SetPipelineState	SetPredication
SOSetTargets		

Direct3D 파이프라인

- 그래픽 파이프라인 상태

```
typedef struct D3D12_GRAPHICS_PIPELINE_STATE_DESC {  
    ID3D12RootSignature *pRootSignature;  
    D3D12_SHADER_BYTECODE VS;  
    D3D12_SHADER_BYTECODE PS;  
    D3D12_SHADER_BYTECODE DS;  
    D3D12_SHADER_BYTECODE HS;  
    D3D12_SHADER_BYTECODE GS;  
    D3D12_STREAM_OUTPUT_DESC StreamOutput;  
    D3D12_BLEND_DESC BlendState;  
    UINT SampleMask;  
    D3D12_RASTERIZER_DESC Rasterizer;  
    D3D12_DEPTH_STENCIL_DESC DepthStencil;  
    D3D12_INPUT_LAYOUT_DESC InputLayout;  
    D3D12_INDEX_BUFFER_STRIP_CUT_VALUE INDEX_STRIP_CutValue;  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE PrimitiveTopology;  
    UINT NumRenderTargets;  
    DXGI_FORMAT RTVFormats[8];  
    DXGI_FORMAT DSVFormat;  
    DXGI_SAMPLE_DESC SampleDesc;  
    UINT NodeMask;  
    D3D12_CACHED_PIPELINE_STATE CachedPSO;  
    D3D12_PIPELINE_STATE_FLAGS Flags;  
} D3D12_GRAPHICS_PIPELINE_STATE_DESC;
```

```
typedef struct D3D12_SHADER_BYTECODE {  
    void *pShaderBytecode;  
    SIZE_T BytecodeLength;  
} D3D12_SHADER_BYTECODE;
```

```
typedef struct D3D12_RASTERIZER_DESC {  
    D3D12_FILL_MODE FillMode;  
    D3D12_CULL_MODE CullMode;  
    BOOL FrontCounterClockwise;  
    INT DepthBias;  
    FLOAT DepthBiasScale;  
    FLOAT SlopeScaleBias;  
    BOOL DepthClipEnable;  
    BOOL MultisampleEnable;  
    BOOL AntialiasedLineEnable;  
    UINT ForcedSampleCount;  
    D3D12_CONSERVATIVE_RASTERIZATION_MODE ConservativeRaster;  
} D3D12_RASTERIZER_DESC;
```

```
typedef struct D3D12_DEPTH_STENCIL_DESC {  
    BOOL DepthEnable;  
    D3D12_DEPTH_WRITE_MASK DepthWriteMask;  
    D3D12_COMPARISON_FUNC DepthFunc;  
    BOOL StencilEnable;  
    UINT8 StencilReadMask;  
    UINT8 StencilWriteMask;  
    D3D12_DEPTH_STENCILOP_DESC FrontFace;  
    D3D12_DEPTH_STENCILOP_DESC BackFace;  
} D3D12_DEPTH_STENCIL_DESC;
```

```
typedef enum D3D12_PIPELINE_STATE_FLAGS {  
    D3D12_PIPELINE_STATE_FLAG_NONE,  
    D3D12_PIPELINE_STATE_FLAG_TOOL_DEBUG  
} D3D12_PIPELINE_STATE_FLAGS;
```



```
typedef enum D3D12_PRIMITIVE_TOPOLOGY_TYPE {  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE_UNDEFINED,  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE_TRIANGLE,  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE_TRIANGLE_STRIP,  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE_TRIANGLE_FAN,  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE_LINE,  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE_LINE_STRIP,  
    D3D12_PRIMITIVE_TOPOLOGY_TYPE_PATCH  
} D3D12_PRIMITIVE_TOPOLOGY_TYPE;
```

테셀레이션(Tessellation)

• 헐 쉐이더(Hull Shader) 작성

- 헐 쉐이더는 두 개의 함수(쉐이더 함수, 패치 상수 함수)로 작성
 - ① 헐 쉐이더 입력 및 출력 제어점을 정의
 - ② 출력 패치 상수 데이터를 정의
 - ③ 패치 상수 함수(Constant Hull Shader)를 정의
 - ④ 헐 쉐이더 함수(Control Point Hull Shader)를 정의

```
HS_CONSTANT_OUTPUT HSConstant(InputPatch<VS_OUTPUT, 32> input, uint PatchID : SV_PrimitiveID)
```

```
{  
    HS_CONSTANT_OUTPUT output;  
    ...  
    return(output);  
}
```

```
[domain("quad")]  
[partitioning("integer")]  
[outputtopology("triangle_cw")]  
[outputcontrolpoints(16)]  
[patchconstantfunc("HSConstant")]  
BEZIER_CONTROL_POINT HSBezier(InputPatch<VS_OUTPUT, 32> input, uint i : SV_OutputControlPointID,  
uint patchID : SV_PrimitiveID)  
{  
    BEZIER_CONTROL_POINT output;  
    ...  
    return(output);  
}
```

```
struct HS_CONSTANT_OUTPUT  
{  
    float fTessEdges[4] : SV_TessFactor;  
    float fTessInsides[2] : SV_InsideTessFactor;  
    float3 vTangent[4] : TANGENT;  
    float2 vUV[4] : TEXCOORD;  
    float3 vTanUCorner[4] : TANUCORNER;  
    float3 vTanVCorner[4] : TANVCORNER;  
    float4 vCWts : TANWEIGHTS;  
};
```

```
struct VS_OUTPUT  
{  
    float3 vPosition : WORLDPOS;  
    float2 vUV : TEXCOORD0;  
    float3 vTangent : TANGENT;  
};
```

테셀레이션(Tessellation)

• 헐 쉐이더(Hull Shader) 작성

- ① 헐 쉐이더 입력 및 출력 제어점을 정의

```
struct VS_OUTPUT
{
    float3 vPosition : POSITION;
    float2 vUV : TEXCOORD0;
    float3 vTangent : TANGENT;
};
```

```
struct BEZIER_CONTROL_POINT
{
    float3 vPosition : BEZIERPOS;
};
```

- ② 출력 패치 상수 데이터를 정의

상수 헐 쉐이더(Constant Hull Shader)는 입력 패치마다 실행
테셀레이션 인자(Tessellation Factor)를 출력해야 함

```
struct HS_CONSTANT_OUTPUT
{
    float fTessEdges[4] : SV_TessFactor;
    float fTessInsides[2] : SV_InsideTessFactor;

    float3 vTangent[4] : TANGENT;
    float2 vUV[4] : TEXCOORD;
    float3 vTanUCorner[4] : TANUCORNER;
    float3 vTanVCorner[4] : TANVCORNER;
    float4 vCWts : TANWEIGHTS;
};
```

4개의 제어점을 갖는 사각형 패치

사각형(Quad)은 4개의 변을 가짐
4개의 변을 어떻게 나눌 것인가를 정의
테셀레이터는 이러한 테셀레이션 인자를 사용

D3D_PRIMITIVE_TOPOLOGY_4_CONTROL_POINT_PATCHLIST
pd3dCommandList->DrawInstanced(4, 1, 0, 0);

테셀레이션(Tessellation)

• 시스템 생성값(System Value Semantics)

SV_TessFactor	패치의 각 에지에 대한 테셀레이션의 양을 정의 사각형 또는 삼각형 패치를 사용하는 헐 쉐이더의 출력에 사용 도메인 쉐이더의 입력에 사용(패치-상수 데이터 시그너쳐 일치) 테셀레이션 인자들은 배열로 선언되어야 함, 팩킹될 수 없음 테셀레이션 인자들의 나열 순서는 항상 시계방향(CW) 사각형 패치의 시작은 왼쪽 에지부터, 삼각형 패치의 시작은 두 번째 에지부터 Direct3D 12 하드웨어가 지원하는 최대 테셀레이션 인자는 64 모든 테셀레이션 인자가 0이면 그 패치는 파이프라인에서 처리되지 않음	float[4]	사각형 패치
		float[3]	삼각형 패치
		float[2]	등치선
SV_InsideTessFactor	패치 표면에 대한 테셀레이션의 양을 정의 이 값은 헐 쉐이더의 패치 상수 함수에서 정의되어야 함 테셀레이션 인자들은 배열로 선언되어야 함, 팩킹될 수 없음 사각형 또는 삼각형 패치를 사용하는 헐 쉐이더의 출력에 사용 도메인 쉐이더의 입력에 사용(패치-상수 데이터 시그너쳐 일치)	float[2]	사각형 패치
		float	삼각형 패치
SV_OutputControlPointID	헐 쉐이더에서 현재 사용하는 출력 제어점의 인덱스를 정의		
SV_DomainLocation	현재 도메인 점의 헐에서의 위치를 정의	float2	사각형 패치
		float3	삼각형 패치
		float2	등치선

테셀레이션 인자는 일반적으로 카메라와의 거리, 화면 픽셀의 개수, 다각형의 방향, 표면의 거칠기에 따라 결정

테셀레이션(Tessellation)

- 헐 쉐이더(Hull Shader) 작성

- ③ 패치 상수 함수를 정의

```
HS_CONSTANT_OUTPUT HSConstant(InputPatch<VS_OUTPUT, 3> input, uint nPatchID : SV_PrimitiveID)
{
    float3 vCenter = (input[0].positionW + input[1].positionW + input[2].positionW) / 3.0f;
    float fDistanceToCamera = distance(vCenter, gvCameraPosition);

    float fMin = 10.0f, fMax = 100.0f;
    float fTessFactor = 6.0f * saturate((fMax - fDistanceToCamera) / (fMax - fMin));

    HS_CONSTANT_OUTPUT output;
    output.fTessEdges[0] = fTessFactor;
    output.fTessEdges[1] = fTessFactor;
    output.fTessEdges[2] = fTessFactor;
    output.fTessInsides[0] = fTessFactor;

    return(output);
}
```

```
struct VS_OUTPUT
{
    float3 position : SV_POSITION;
    float3 positionW : POSITION;
};
```

```
struct HS_CONSTANT_OUTPUT
{
    float fTessEdges[3] : SV_TessFactor;
    float fTessInsides[1] : SV_InsideTessFactor;
};
```

InputPatch	입력 제어점 배열, [] 연산자
OutputPatch	출력 제어점 배열, [] 연산자

카메라와 삼각형 패치 사이의 거리에 따라 테셀레이션 인자를 설정

카메라까지의 거리가 fMin 보다 작거나 같으면 테셀레이션 인자가 최대(가장 많은 삼각형들로 테셀레이션이 됨)

카메라까지의 거리가 fMax 보다 크면 테셀레이션 인자가 음수(테셀레이션이 되지 않음)

카메라까지의 거리가 fMin과 fMax 사이면 테셀레이션 인자가 거리에 반비례

테셀레이션(Tessellation)

• 헐 쉐이더(Hull Shader) 작성

- ④ 헐 쉐이더 함수(제어점 헐 쉐이더: Control Point Hull Shader)를 정의
헐 쉐이더는 각 출력 제어점에 대하여 한 번씩 호출이 됨

```
[domain("tri")]
[partitioning("integer")]
[outputtopology("triangle_cw")]
[outputcontrolpoints(3)]
[patchconstantfunc("HSConstant")]
[maxtessfactor(64.0f)]
HS_OUTPUT HS(InputPatch<VS_OUTPUT, 3> input, uint i : SV_OutputControlPointID,
uint patchID : SV_PrimitiveID)
{
    HS_OUTPUT output;
    output.position = input[i].position;
    return(output);
}
```

```
struct VS_OUTPUT
{
    float3 position : SV_POSITION;
    float3 positionW : POSITION;
};
```

```
struct HS_OUTPUT
{
    float3 position : POSITION;
};
```

헐 쉐이더 함수는 패치의 모든 제어점을 입력 받음
헐 쉐이더 함수는 매번 하나의 새로운 제어점(정점)을 출력함
입력 제어점은 정점 쉐이더의 출력
출력할 제어점의 인덱스는 SV_OutputControlPointID 시멘틱으로 입력

domain	패치의 유형	tri, quad, isoline
partitioning	분할 방법(테셀레이션 인자의 사용 방법)	integer, fractional_even, fractional_odd, pow2
outputtopology	출력 프리미티브 유형	line, triangle_cw, triangle_ccw
outputcontrolpoints	출력 제어점의 개수	출력 제어점의 개수 만큼 헐 쉐이더 함수가 호출됨
patchconstantfunc	패치 상수 함수 이름	
maxtessfactor	최대 테셀레이션 인자	Direct3D 12 하드웨어가 지원하는 최대값은 64

테셀레이션(Tessellation)

• 헐 쉐이더(Hull Shader) 작성

- ④ 헐 쉐이더 함수(제어점 헐 쉐이더: Control Point Hull Shader)를 정의

partitioning

integer	테셀레이션 인자의 값보다 크거나 같은 가장 작은 자연수 [1..64] 3.0→3, 3.25→4, 3.75→4, 4.25→5, 4.75→5
fractional_odd	테셀레이션 인자의 값보다 크거나 같은 가장 작은 홀수 [1, 3, 5, ..., 63] 3.0→3, 3.25→5, 3.75→5, 4.25→5, 4.75→5, 5→5 소수부분만큼 슬라이드 테셀레이션 인자가 $f \in [3.0, 5.0]$ 일 때 $\{(f-3.0)/(5.0-3.0)*100\}$ 의 비율로 슬라이드
fractional_even	테셀레이션 인자의 값보다 크거나 같은 가장 작은 짝수 [2, 4, 6, ..., 64] 3.0→4, 3.25→4, 3.75→4, 4.25→6, 4.75→6, 5→6 소수부분만큼 슬라이드 테셀레이션 인자가 $f \in [4.0, 6.0]$ 일 때 $\{(f-4.0)/(6.0-4.0)*100\}$ 의 비율로 슬라이드
pow2	테셀레이션 인자의 값보다 크거나 같은 가장 작은 2^n 형태의 자연수 [1, 2, 4, 8, ..., 64] 3.0→4, 3.25→4, 3.75→4, 4.25→8, 4.75→8, 5→8

outputtopology

triangle_cw,	분할로 생성되는 삼각형 프리미티브(시계 방향 와인딩 순서)
triangle_ccw	분할로 생성되는 삼각형 프리미티브(반시계 방향 와인딩 순서)
line	선분 프리미티브

테셀레이션(Tessellation)

• 헐 쉐이더(Hull Shader) 작성

③ 패치 상수 함수(상수 헐 쉐이더: Constant Hull Shader)를 정의

```
HS_CONSTANT_OUTPUT HSConstant(InputPatch<VS_OUTPUT, 4> input, uint PatchID : SV_PrimitiveID)
```

```
{
```

```
    HS_CONSTANT_OUTPUT output;  
    output.fTessEdges[0] = 3; //왼쪽 에지  
    output.fTessEdges[1] = 3; //위쪽 에지  
    output.fTessEdges[2] = 3; //오른쪽 에지  
    output.fTessEdges[3] = 3; //아래쪽 에지
```

```
    output.fTessInsides[0] = 3; //u-축(열의 개수)  
    output.fTessInsides[1] = 3; //v-축(행의 개수)
```

```
...
```

```
    return(output);
```

```
}
```

```
struct HS_CONSTANT_OUTPUT
```

```
{
```

```
    float fTessEdges[4] : SV_TessFactor;  
    float fTessInsides[2] : SV_InsideTessFactor;
```

```
    float3 vTangent[4] : TANGENT;  
    float2 vUV[4] : TEXCOORD;  
    float3 vTanUCorner[4] : TANUCORNER;  
    float3 vTanVCorner[4] : TANVCORNER;  
    float4 vCWts : TANWEIGHTS;
```

```
};
```

SV_TessFactor

float[4]

사각형 패치

float[3]

삼각형 패치

float[2]

등차선

SV_InsideTessFactor

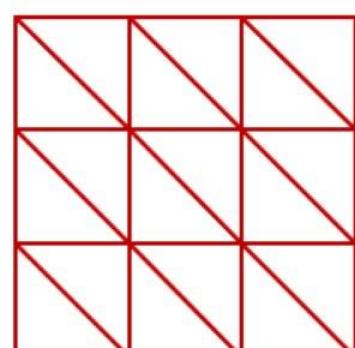
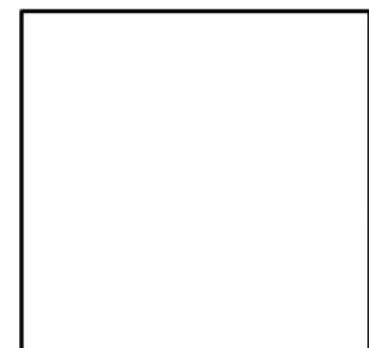
float[2]

사각형 패치

float

삼각형 패치

패치 상수 함수는 패치의 모든 제어점을 입력 받음
제어점은 정점 쉐이더의 출력
패치의 인덱스는 SV_PrimitiveID 시멘틱으로 입력
패치 상수 함수의 출력은 도메인 쉐이더로 전달됨



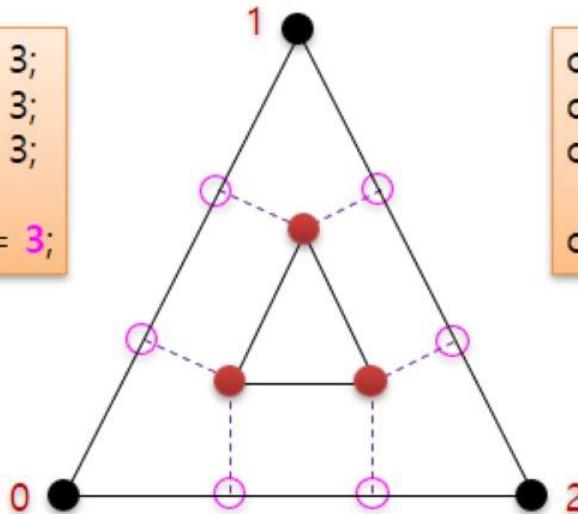
테셀레이션(Tessellation)

• 헐 쉐이더(Hull Shader) 작성

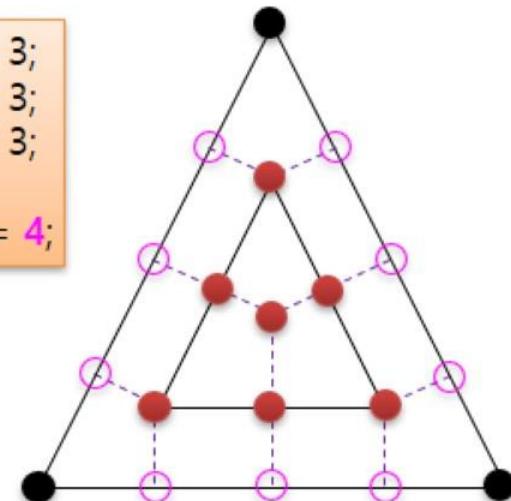
③ 패치 상수 함수를 정의

삼각형 패치(도메인)에 대한 테셀레이션 인자 설정의 예

```
output.fTessEdges[0] = 3;  
output.fTessEdges[1] = 3;  
output.fTessEdges[2] = 3;  
  
output.fTessInsides[0] = 3;
```



```
output.fTessEdges[0] = 3;  
output.fTessEdges[1] = 3;  
output.fTessEdges[2] = 3;  
  
output.fTessInsides[0] = 4;
```



삼각형의 세 변을 에지 테셀레이션 인자로 분할하는 것은 각 변을 분할 하는 것임

삼각형의 세 변을 내부 테셀레이션 인자로 분할함

삼각형의 각 꼭지점을 지나는 두 변에서 같은 순서에 해당하는(이웃하는) 분할된 두 점을 선택함

선택된 점을 지나는 각 변에 수직인 선분을 찾음

수직인 두 선분의 교점을 찾음(이 교점이 삼각형의 내부 꼭지점이 됨)

내부 꼭지점들을 연결하면 내부 삼각형이 됨(내부 삼각형에 대하여 같은 방법으로 내부를 분할함)

내부 삼각형의 변이 수직인 교점을 갖지 않으면 분할을 종료함

내부 삼각형의 내부에 교점이 한 개이면 분할을 종료함

내부 삼각형의 개수는 내부 테셀레이션 인자의 절반임(예: 4→2, 3→1)

내부 테셀레이션(분할)이 수행되면 삼각형 내부에 교점이 적어도 하나 존재하게 됨

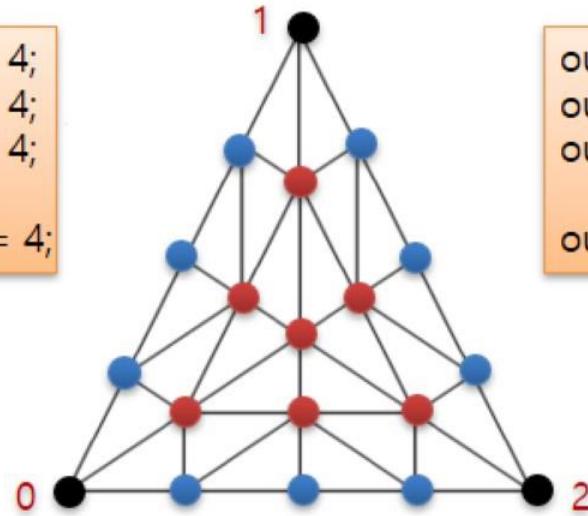
테셀레이션(Tessellation)

- 헐 쉐이더(Hull Shader) 작성

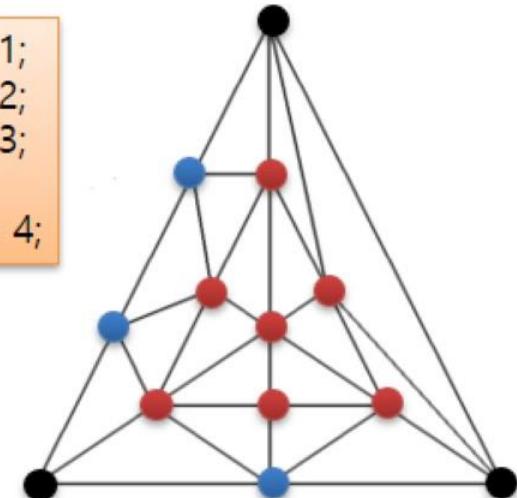
③ 패치 상수 함수를 정의

삼각형 패치(도메인)에 대한 테셀레이션 인자 설정의 예

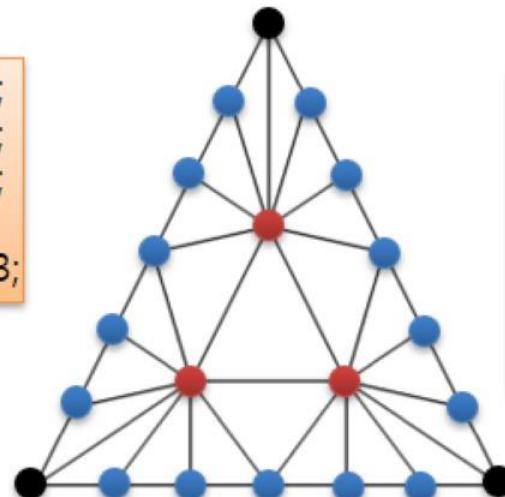
```
output.fTessEdges[0] = 4;  
output.fTessEdges[1] = 4;  
output.fTessEdges[2] = 4;  
  
output.fTessInsides[0] = 4;
```



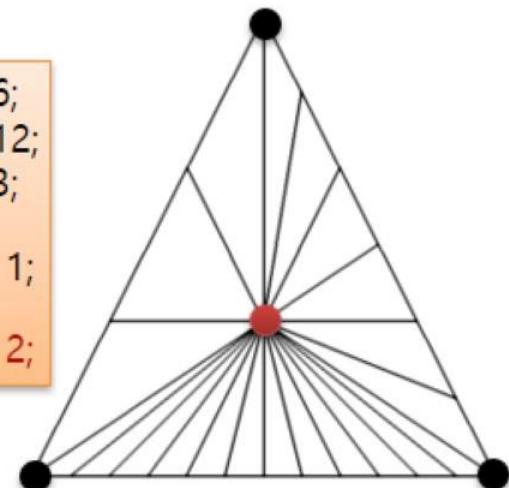
```
output.fTessEdges[0] = 1;  
output.fTessEdges[1] = 2;  
output.fTessEdges[2] = 3;  
  
output.fTessInsides[0] = 4;
```



```
output.fTessEdges[0] = 6;  
output.fTessEdges[1] = 6;  
output.fTessEdges[2] = 6;  
  
output.fTessInsides[0] = 3;
```



```
output.fTessEdges[0] = 6;  
output.fTessEdges[1] = 12;  
output.fTessEdges[2] = 3;  
  
output.fTessInsides[0] = 1;  
  
output.fTessInsides[0] = 2;
```

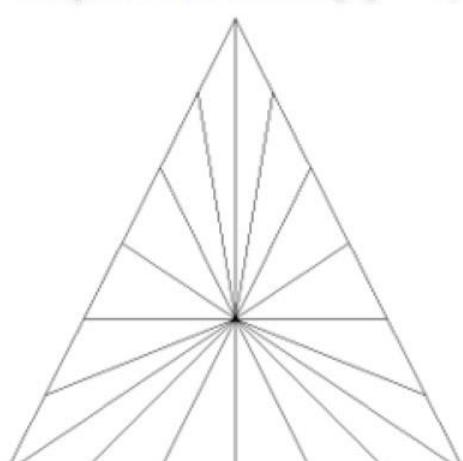


• 헬 쉐이더(Hull Shader) 작성

③ 패치 상수 함수를 정의

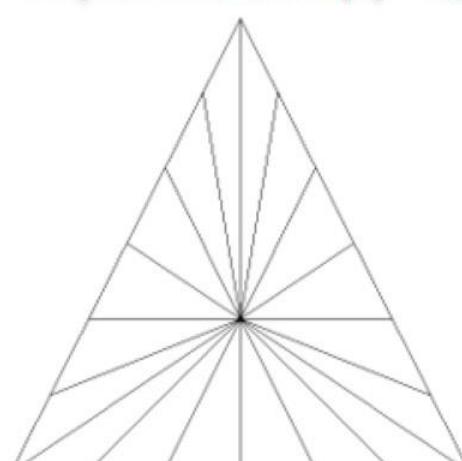
삼각형 패치에 대한 테셀레이션 인자 설정의 예

`output.fTessInsides[0] = 1;`

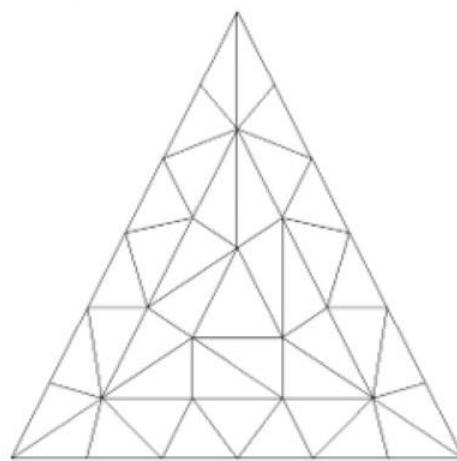
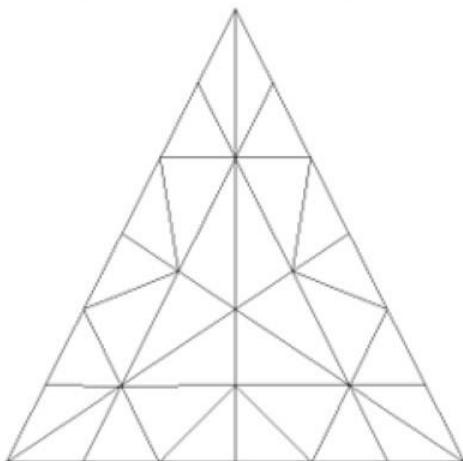


`output.fTessInsides[0] = 2;`

`output.fTessInsides[0] = 2;`

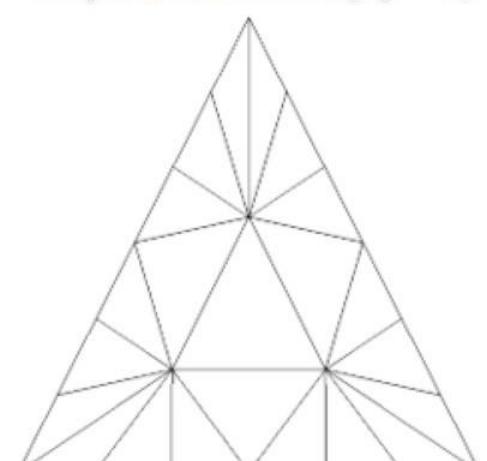


`output.fTessInsides[0] = 5;`

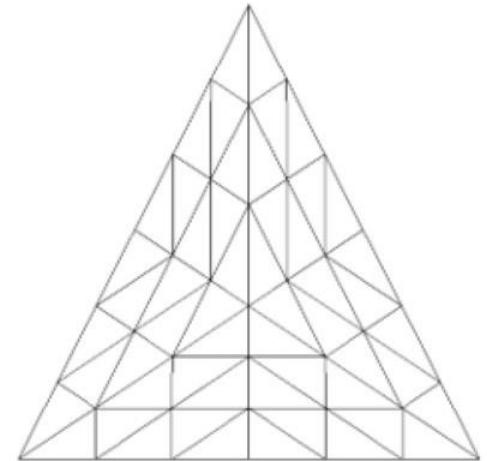


`output.fTessEdges[0] = 6;`
`output.fTessEdges[1] = 6;`
`output.fTessEdges[2] = 6;`

`output.fTessInsides[0] = 3;`



`output.fTessInsides[0] = 6;`

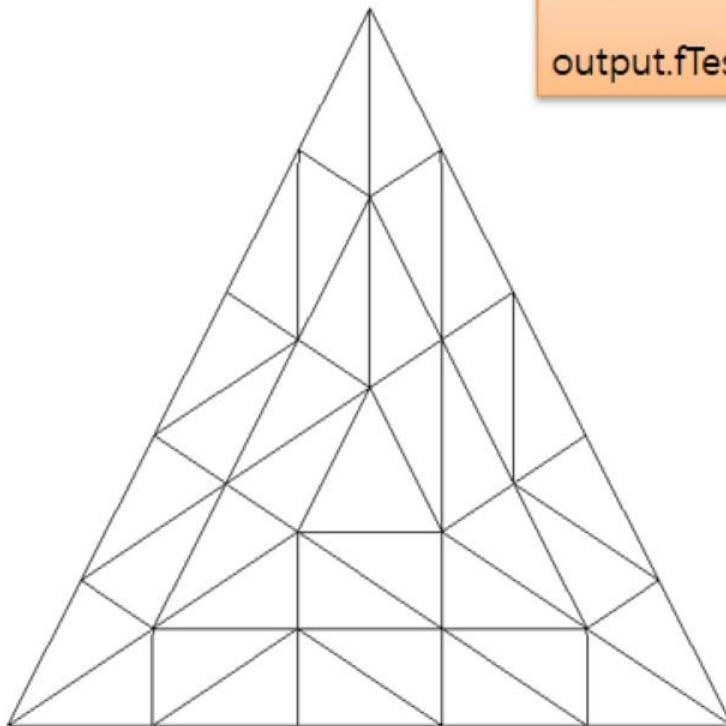


테셀레이션(Tessellation)

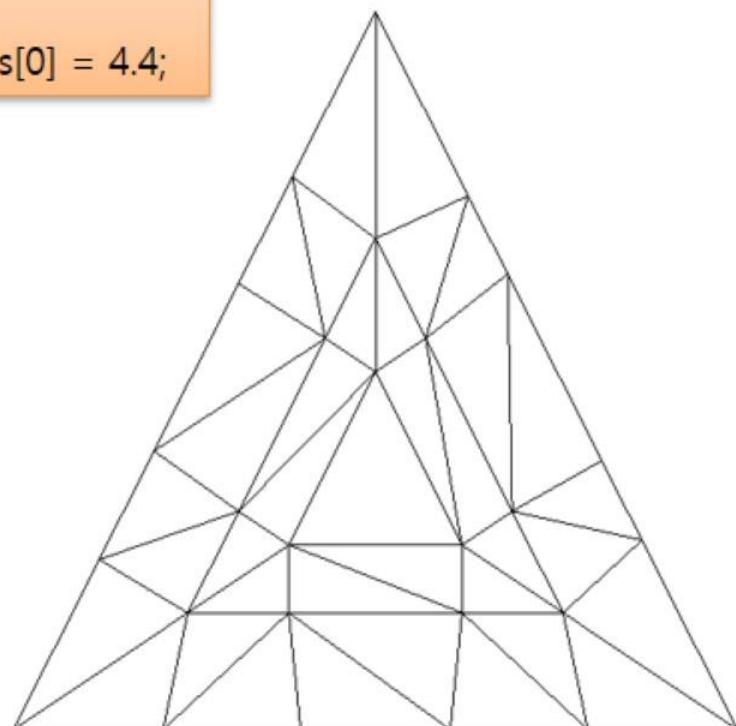
- 헐 쉐이더(Hull Shader) 작성

- ③ 패치 상수 함수를 정의

```
output.fTessEdges[0] = 4.9;  
output.fTessEdges[1] = 4.5;  
output.fTessEdges[2] = 4.1;  
  
output.fTessInsides[0] = 4.4;
```



[partitioning("integer")]



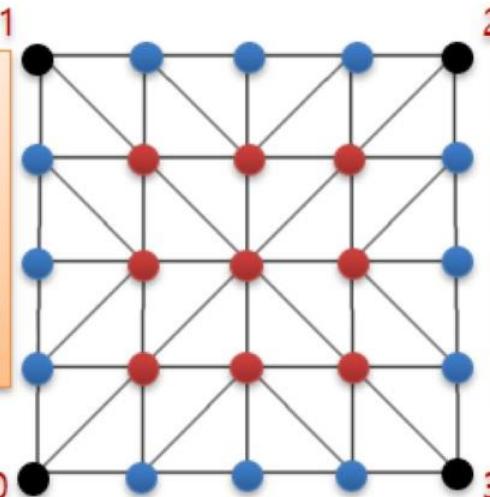
[partitioning("fractional_odd")]

테셀레이션(Tessellation)

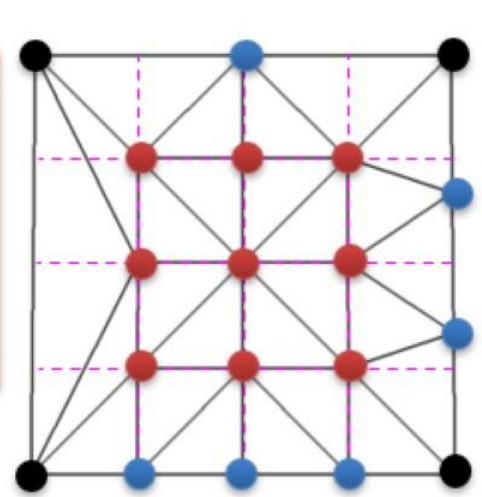
- 헐 쉐이더(Hull Shader) 작성

- ③ 패치 상수 함수를 정의

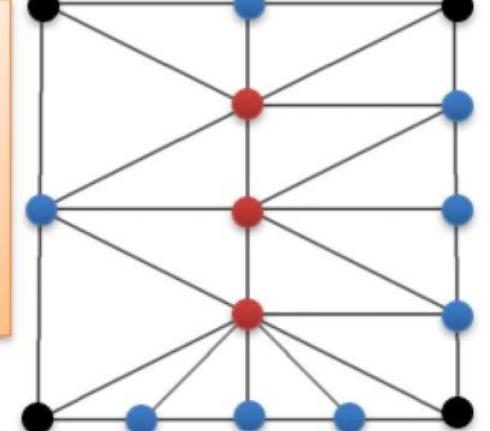
- 사각형 패치(도메인)에 대한 테셀레이션 인자 설정의 예



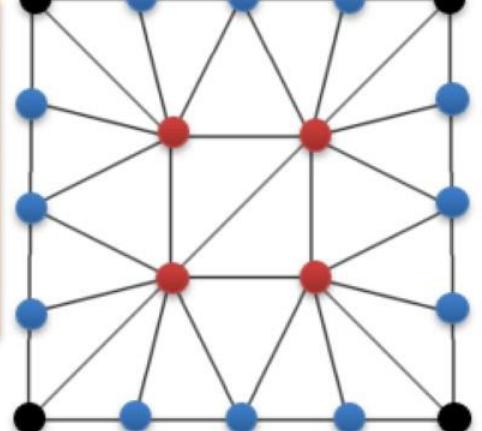
```
output.fTessEdges[0] = 4;  
output.fTessEdges[1] = 4;  
output.fTessEdges[2] = 4;  
output.fTessEdges[3] = 4;  
  
output.fTessInsides[0] = 4;  
output.fTessInsides[1] = 4;
```



```
output.fTessEdges[0] = 1;  
output.fTessEdges[1] = 2;  
output.fTessEdges[2] = 3;  
output.fTessEdges[3] = 4;  
  
output.fTessInsides[0] = 4;  
output.fTessInsides[1] = 4;
```



```
output.fTessEdges[0] = 2;  
output.fTessEdges[1] = 2;  
output.fTessEdges[2] = 4;  
output.fTessEdges[3] = 4;  
  
output.fTessInsides[0] = 2;  
output.fTessInsides[1] = 4;
```

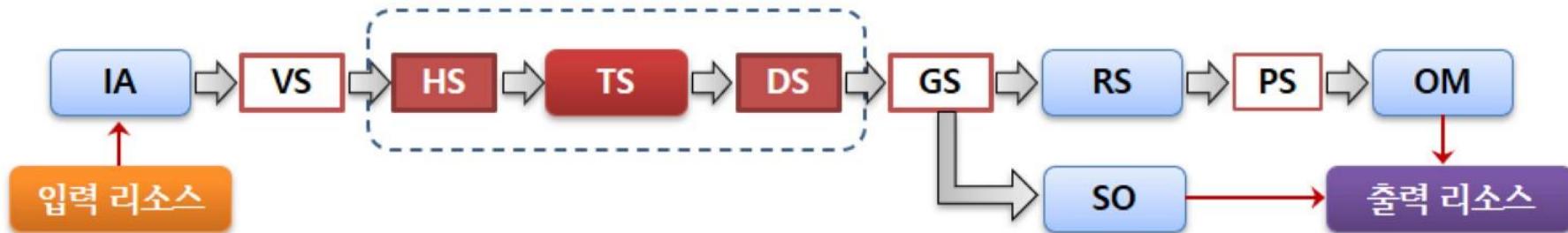


```
output.fTessEdges[0] = 4;  
output.fTessEdges[1] = 4;  
output.fTessEdges[2] = 4;  
output.fTessEdges[3] = 4;  
  
output.fTessInsides[0] = 3;  
output.fTessInsides[1] = 3;
```

테셀레이션(Tessellation)

• 도메인 쉐이더(Domain Shader)

- 도메인 쉐이더는 테셀레이터가 생성한 각 정점에 대하여 한번씩 실행됨



테셀레이션 단계가 없다면?

테셀레이터는 새로운 정점과 삼각형들을 생성함

이러한 정점들에 대한 변환 과정이 필요(정점 쉐이더의 역할을 수행하는 과정이 필요)

도메인 쉐이더의 출력을 텍셀 쉐이더로 전달하려면 SV_Position 시멘틱을 포함해야 함

```
struct POINT
{
    float3 position : POSITION;
};
```

```
struct DS_OUTPUT
{
    float4 position : SV_POSITION;
};
```

```
[domain("quad")]
DS_OUTPUT DS(HS_CONSTANT input, float2 uv : SV_DomainLocation, OutputPatch<POINT, 16> patch)
{
    DS_OUTPUT output;
    ...
    return(output);
}
```

```
struct HS_CONSTANT
{
    float fTessEdges[4] : SV_TessFactor;
    float fTessInsides[2] : SV_InsideTessFactor;
};
```

테셀레이션(Tessellation)

• 도메인 쉐이더(Domain Shader)

- 도메인 쉐이더는 테셀레이터가 생성한 각 정점에 대한 변환을 처리해야 함
- 도메인 쉐이더는 헐 쉐이더의 출력과 테셀레이터가 생성한 정점을 입력 받음
 헐 쉐이더의 출력은 상수 데이터와 출력 패치(1~32개의 제어점)
- 테셀레이터가 생성한 정점의 위치는 출력 패치의 헐에 대한 중심 좌표계로 주어짐

SV_DomainLocation

현재 도메인 점의 헐에서의 위치를 정의

float2

사각형 패치

float3

삼각형 패치

```
[domain("quad")]
DS_OUTPUT DS(HS_CONSTANT input, float2 uv : SV_DomainLocation, OutputPatch<HS_OUTPUT, 4> patch)
{
    DS_OUTPUT output = (DS_OUTPUT)0;
    float3 v1 = lerp(patch[0].position, patch[1].position, uv.x);
    float3 v2 = lerp(patch[2].position, patch[3].position, uv.x);
    float3 position = lerp(v1, v2, uv.y);

    matrix mtxWorldViewProjection = mul(gmtxWorld, gmtxView);
    mtxWorldViewProjection = mul(mtxWorldViewProjection, gmtxProjection);
    output.position = mul(float4(position, 1.0f), mtxWorldViewProjection);

    return(output);
}

position.y = 0.3f*(position.z*sin(position.x) + position.x*cos(position.z));
```

struct HS_CONSTANT

{

 float fTessEdges[4] : **SV_TessFactor**;

 float fTessInsides[2] : **SV_InsideTessFactor**;

};

struct HS_OUTPUT

{

 float3 position : POSITION;

};

struct DS_OUTPUT

{

 float4 position : **SV_POSITION**;

};

테셀레이션(Tessellation)

• 파라메터 곡선(Parametric Curves)

- 면(Face)은 에지(Edge)로 둘러싸여 있음, 각 에지는 선분 또는 곡선임
- 하나의 면은 표면(Surface)의 일부분(즉 표면의 패치(Patch))
- 파라메터 곡선
 - $F: [0, 1] \rightarrow (f(u), g(u), h(u))$
 - 함수 f, g, h, F 의 정의역(Domain)이 $[0, 1]$ 일 필요는 없으나 일반적으로 $[0, 1]$ 로 제한
 - $F(u)$ 는 0과 1사이의 실수 u 를 공간의 한 점으로 매핑
 - 함수 f, g, h 는 다항 함수로 가정

- 파라메터 곡선의 예
 - 선분(Line Segment)

$$B + uD, B = (b_1, b_2, b_3), D = (d_1, d_2, d_3), u \in [0, 1]$$

$$f(u) = b_1 + ud_1, g(u) = b_2 + ud_2, h(u) = b_3 + ud_3$$

F 는 $u \in [0, 1]$ 을 B 와 $B+D$ 사이의 선분 위의 한 점으로 매핑

- 원(Circle)

$$f(u) = a + r * \cos(2\pi * u), g(u) = b + r * \sin(2\pi * u), u \in [0, 1] \rightarrow [0, 2\pi]$$

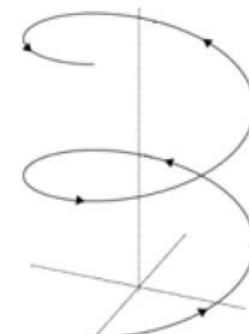
$$(f(u) - a)^2 + (g(u) - b)^2 = r^2$$

F 는 $u \in [0, 1]$ 을 중심이 (a, b) , 반지름이 r 인 원 위의 한 점으로 매핑

- 나선(Circular Helix)

$$f(u) = a * \cos(u), g(u) = a * \sin(u), h(u) = b * u$$

F 는 $u \in [0, 1]$ 을 시작점이 $(a, 0, 0)$, 끝점이 $(a, 0, b * 4\pi)$ 인 나선으로 매핑
원기둥의 표면을 둘러싸는 나선



테셀레이션(Tessellation)

- 베지어 곡선(Bezier Curve)

- 베지어 곡선(Bezier Curve)
 - 파라메터 곡선(Parametric Curve)
 - P_0, \dots, P_n 의 제어점(Control Point)들의 집합으로 정의
 - P_0 와 P_n 는 곡선의 시작점과 끝점
 - n : 차수(Order)

- 선형($n = 1$) 베지어 곡선

$$P(t) = P_0 + t(P_1 - P_0) = (1 - t)P_0 + tP_1, t \in [0, 1]$$

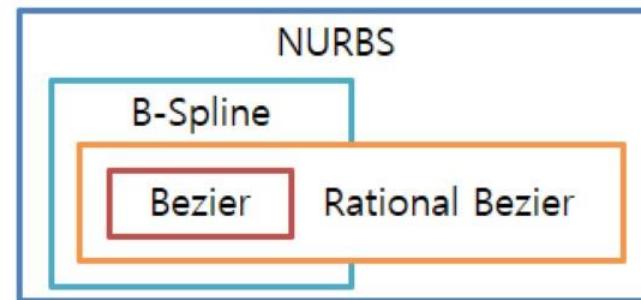
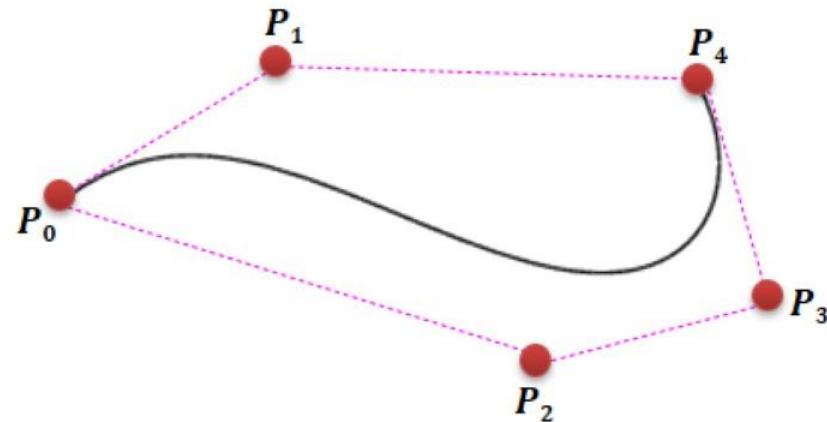
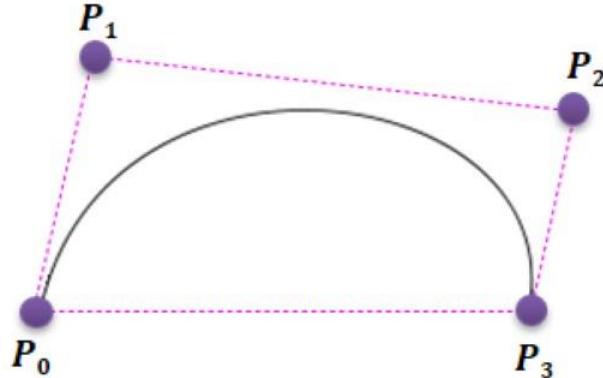
- 이차($n = 2$) 베지어 곡선

$$\begin{aligned} P(t) &= (1 - t)[(1 - t)P_0 + tP_1] + t[(1 - t)P_1 + tP_2], t \in [0, 1] \\ &= (1 - t)^2 P_0 + 2(1 - t)tP_1 + t^2 P_2, t \in [0, 1] \end{aligned}$$

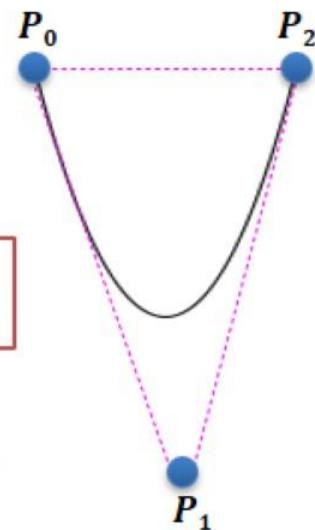
- n 차 베지어 곡선

$$P(t) = \sum_{i=0}^n B_{n,i}(t)P_i, \quad B_{n,i}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

- 컨벡스 헬(Convex Hull)



$${}_n C_r = \frac{n!}{r!(n-r)!}$$



$$(a + b)^1 = a + b$$

$$(a + b)^2 = a^2 + 2ab + b^2$$

$$(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

$$(a + b)^4 = a^4 + \dots + b^4$$

$$(a + b)^n = a^n + \dots + b^n$$

$$(a + b)^n = \sum_{r=0}^n {}_n C_r a^{n-r} b^r$$

$$= {}_n C_0 a^n b^0 + {}_n C_1 a^{n-1} b^1 + {}_n C_2 a^{n-2} b^2 + \dots + {}_n C_r a^{n-r} b^r + \dots + {}_n C_n a^0 b^n$$

$${}_n C_r = \frac{n!}{r! (n-r)!}$$

$$((1-t) + t)^n = \sum_{r=0}^n {}_n C_r (1-t)^{n-r} t^r$$

$$B_{n,i}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

테셀레이션(Tessellation)

- 베지어 곡선(Bezier Curve)

- 이차($n=2$, Quadratic) 베지어 곡선

$P_0, P_1, P_2 \rightarrow \{ P_0, P_1 \text{을 선형보간}(P_0(t)), P_1, P_2 \text{을 선형보간}(P_1(t)) \}$

$$P_0(t) = (1-t)P_0 + tP_1$$

$$P_1(t) = (1-t)P_1 + tP_2$$

$$\begin{aligned} P(t) &= (1-t)P_0(t) + tP_1(t) = (1-t)[(1-t)P_0 + tP_1] + t[(1-t)P_1 + tP_2] \\ &= (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2 \end{aligned}$$

- 삼차($n=3$, Cubic) 베지어 곡선

$P_0, P_1, P_2, P_3 \rightarrow \{ P_0, P_1 \text{을 선형보간}, P_1, P_2 \text{을 선형보간}, P_2, P_3 \text{을 선형보간} \}$

$$P_0(t) = (1-t)P_0 + tP_1$$

$$P_1(t) = (1-t)P_1 + tP_2$$

$$P_2(t) = (1-t)P_2 + tP_3$$

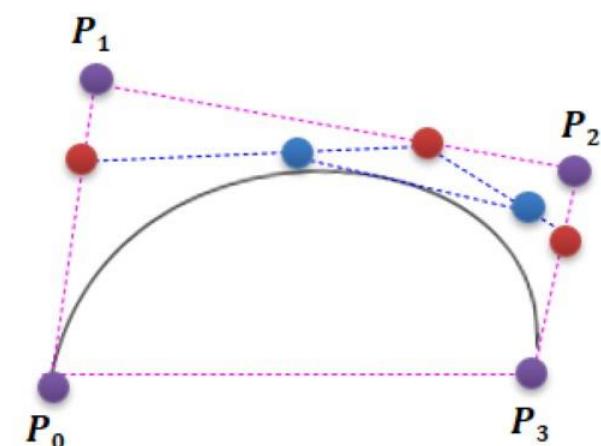
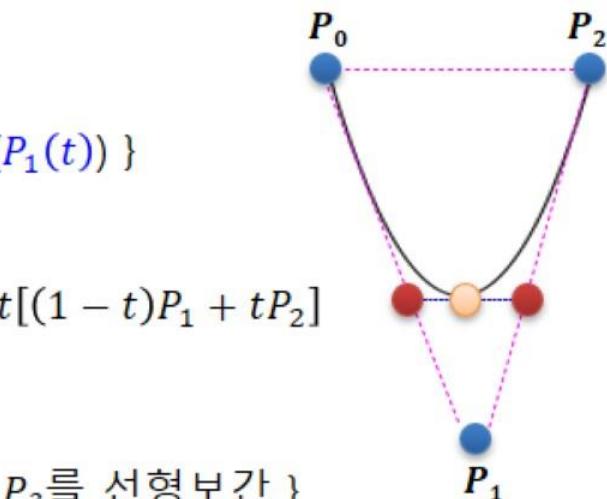
$$\begin{aligned} P(t) &= (1-t)[(1-t)P_0(t) + tP_1(t)] + t[(1-t)P_1(t) + tP_2(t)] \\ &= (1-t)^2 P_0(t) + 2(1-t)tP_1(t) + t^2 P_2(t) \\ &= (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3 \end{aligned}$$

- n -차 베지어 곡선

$$P(t) = \sum_{i=0}^n B_{n,i}(t)P_i$$

Bernstein Basis Function: $B_{n,i}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$

$n = 3, P(t) = B_{3,0}(t)P_0 + B_{3,1}(t)P_1 + B_{3,2}(t)P_2 + B_{3,3}(t)P_3$



테셀레이션(Tessellation)

- 헤르밋 곡선(Hermite Curve)

- 삼차($n=3$, Cubic) 베지어 곡선

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

$$P'(t) = -3(1-t)^2 P_0 + \{3(1-t)^2 - 6t(1-t)\} P_1 + \{6t(1-t) - 3t^2\} P_2 + 3t^2 P_3$$

$$P'(0) = 3(P_1 - P_0) \quad P_1 = P_0 + \frac{1}{3} P'(0)$$

$$P'(1) = 3(P_3 - P_2) \quad P_2 = P_3 - \frac{1}{3} P'(1)$$

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

$$= (1-t)^3 P_0 + 3(1-t)^2 t \left\{ P_0 + \frac{1}{3} P'(0) \right\} + 3(1-t)t^2 \left\{ P_3 - \frac{1}{3} P'(1) \right\} + t^3 P_3$$

$$= (1-3t^2)P_0 + 3(1-t)^2 t \{P_0 + \frac{1}{3} P'(0)\} + 3(1-t)t^2 \{P_3 - \frac{1}{3} P'(1)\} + t^3 P_3$$

$$= (1-3t^2 + 2t^3)P_0 + (1-t)^2 t P'(0) + (3t^2 - 2t^3)P_3 - t^2(1-t) P'(1)$$

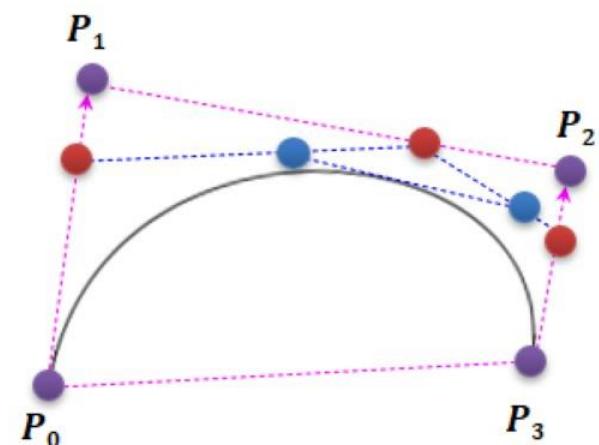
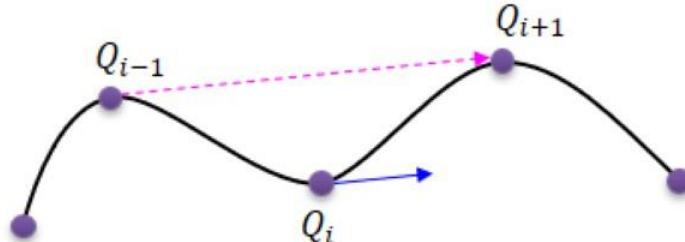
- 스플라인(Spline) 곡선

$\{C_i\}$: 연결된 베지어 또는 헤르밋 곡선 $C_i = [Q_i, Q_{i+1}]$

- Catmull-Rom 스플라인

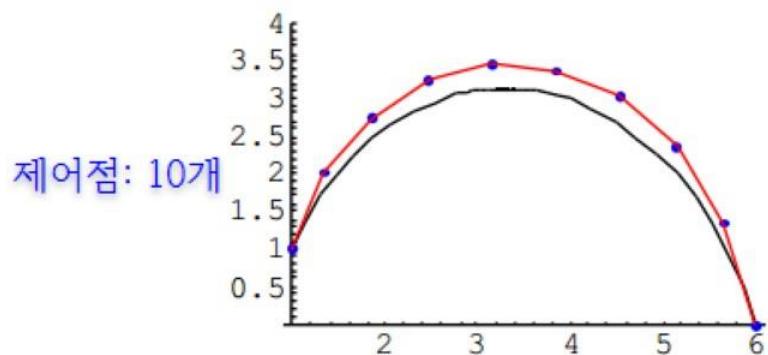
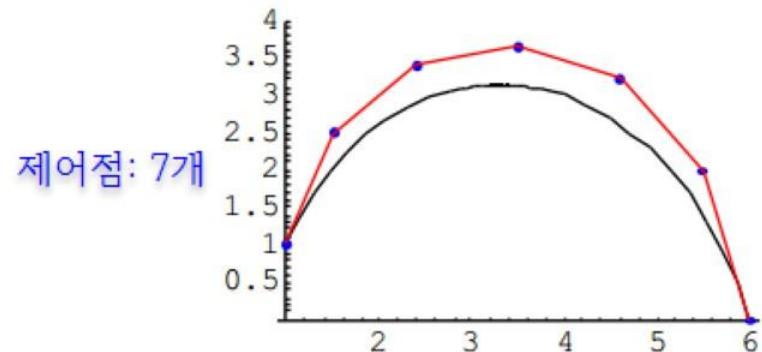
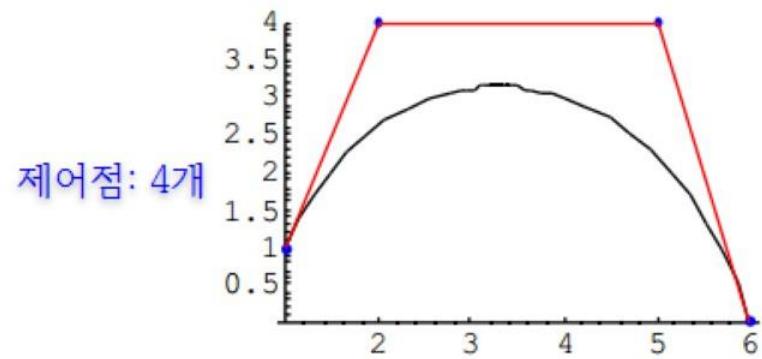
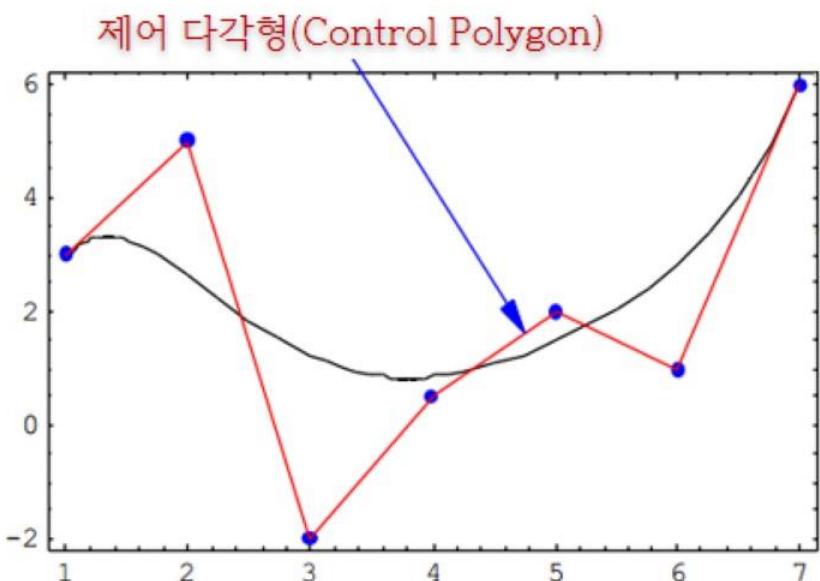
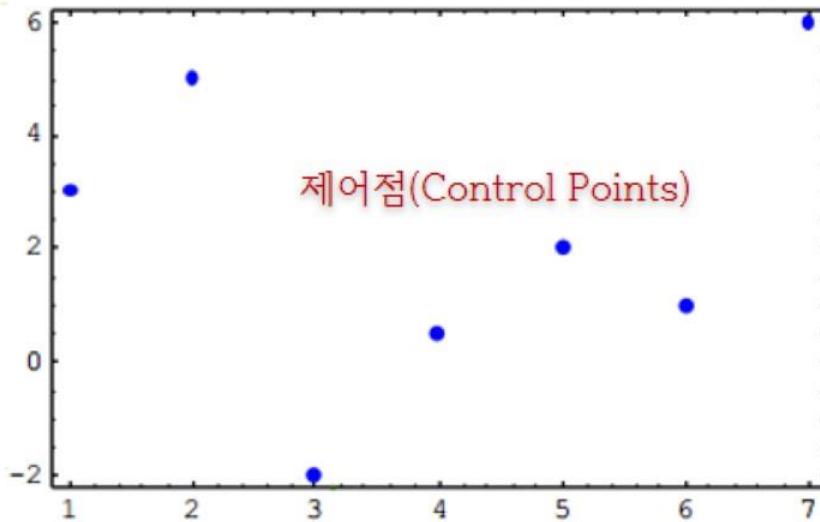
$$T_i = k * (Q_{i+1} - Q_{i-1})$$

k : Q_i 에서 곡선이 휘는 정도



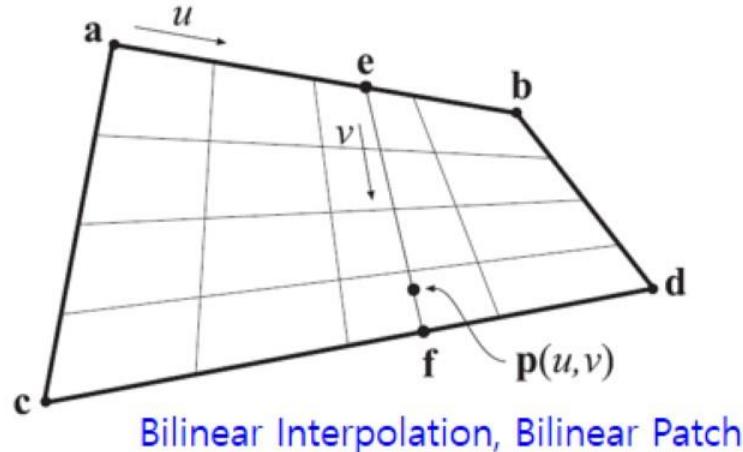
테셀레이션(Tessellation)

- 베지어 곡선(Bezier Curve)



테셀레이션(Tessellation)

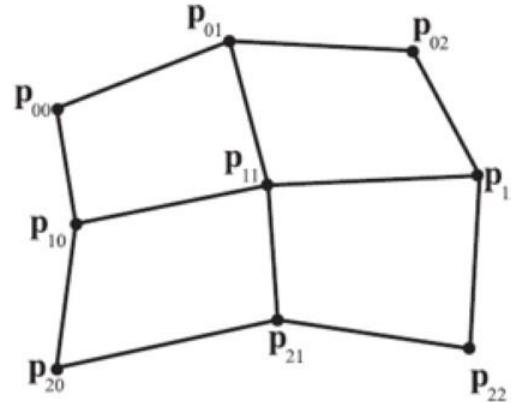
- 베지어 곡면(Bezier Surface)



$$e(u) = (1 - u)\mathbf{a} + u\mathbf{b}, u \in [0, 1]$$

$$f(u) = (1 - u)\mathbf{c} + u\mathbf{d}, u \in [0, 1]$$

정의역(Domain): $(u, v) \in [0, 1] \times [0, 1]$



$$p(u, v) = (1 - v)\mathbf{e}(u) + v\mathbf{f}(u), v \in [0, 1]$$

$$= (1 - v)[(1 - u)\mathbf{a} + u\mathbf{b}] + v[(1 - u)\mathbf{c} + u\mathbf{d}]$$

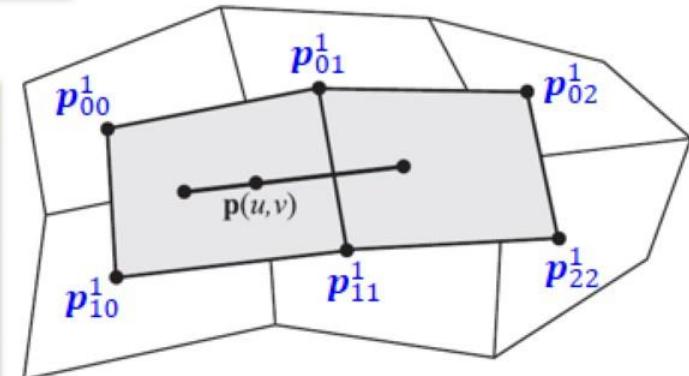
$$= (1 - v)(1 - u)\mathbf{a} + (1 - v)u\mathbf{b} + v(1 - u)\mathbf{c} + vu\mathbf{d}$$

$$p_0^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{00} + (1 - v)u\mathbf{p}_{01} + v(1 - u)\mathbf{p}_{10} + vu\mathbf{p}_{11}$$

$$p_0^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{01} + (1 - v)u\mathbf{p}_{02} + v(1 - u)\mathbf{p}_{11} + vu\mathbf{p}_{12}$$

$$p_1^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{10} + (1 - v)u\mathbf{p}_{11} + v(1 - u)\mathbf{p}_{20} + vu\mathbf{p}_{21}$$

$$p_1^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{11} + (1 - v)u\mathbf{p}_{12} + v(1 - u)\mathbf{p}_{21} + vu\mathbf{p}_{22}$$



$$p(u, v) = (1 - u)^2(1 - v)^2\mathbf{p}_{00} + 2u(1 - u)(1 - v)^2\mathbf{p}_{01} + u^2(1 - v)^2\mathbf{p}_{02} + 2(1 - u)^2v(1 - v)\mathbf{p}_{10}$$

$$+ 4uv(1 - u)(1 - v)\mathbf{p}_{11} + 2u^2v(1 - v)\mathbf{p}_{12} + (1 - u)^2v^2\mathbf{p}_{20} + 2u(1 - u)v^2\mathbf{p}_{21} + u^2v^2\mathbf{p}_{22}$$

테셀레이션(Tessellation)

- 베지어 곡면(Bezier Surface)

$$\mathbf{p}(u, v) = (1 - v)\mathbf{e}(u) + v\mathbf{f}(u), v \in [0, 1]$$

$$= (1 - v)[(1 - u)\mathbf{a} + u\mathbf{b}] + v[(1 - u)\mathbf{c} + u\mathbf{d}]$$

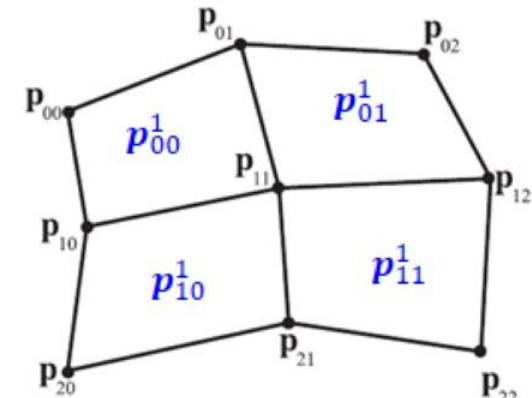
$$= (1 - v)(1 - u)\mathbf{a} + (1 - v)u\mathbf{b} + v(1 - u)\mathbf{c} + vu\mathbf{d}$$

$$\mathbf{p}_{00}^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{00} + (1 - v)u\mathbf{p}_{01} + v(1 - u)\mathbf{p}_{10} + vu\mathbf{p}_{11}$$

$$\mathbf{p}_{01}^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{01} + (1 - v)u\mathbf{p}_{02} + v(1 - u)\mathbf{p}_{11} + vu\mathbf{p}_{12}$$

$$\mathbf{p}_{10}^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{10} + (1 - v)u\mathbf{p}_{11} + v(1 - u)\mathbf{p}_{20} + vu\mathbf{p}_{21}$$

$$\mathbf{p}_{11}^1(u, v) = (1 - v)(1 - u)\mathbf{p}_{11} + (1 - v)u\mathbf{p}_{12} + v(1 - u)\mathbf{p}_{21} + vu\mathbf{p}_{22}$$



Biquadratic Interpolation, Biquadratic Patch

$$\mathbf{p}^2(u, v) = (1 - v)(1 - u)\mathbf{p}_{00}^1 + (1 - v)u\mathbf{p}_{01}^1 + v(1 - u)\mathbf{p}_{10}^1 + vu\mathbf{p}_{11}^1$$

$$\mathbf{p}^2(u, v) = (1 - v)(1 - u)\{(1 - v)(1 - u)\mathbf{p}_{00} + (1 - v)u\mathbf{p}_{01} + v(1 - u)\mathbf{p}_{10} + vu\mathbf{p}_{11}\}$$

$$+ (1 - v)u\{(1 - v)(1 - u)\mathbf{p}_{01} + (1 - v)u\mathbf{p}_{02} + v(1 - u)\mathbf{p}_{11} + vu\mathbf{p}_{12}\}$$

$$+ v(1 - u)\{(1 - v)(1 - u)\mathbf{p}_{10} + (1 - v)u\mathbf{p}_{11} + v(1 - u)\mathbf{p}_{20} + vu\mathbf{p}_{21}\}$$

$$+ vu\{(1 - v)(1 - u)\mathbf{p}_{11} + (1 - v)u\mathbf{p}_{12} + v(1 - u)\mathbf{p}_{21} + vu\mathbf{p}_{22}\}$$

$$= (1 - u)^2(1 - v)^2\mathbf{p}_{00} + u(1 - u)(1 - v)^2\mathbf{p}_{01} + (1 - u)^2v(1 - v)\mathbf{p}_{10} + u(1 - u)v(1 - v)\mathbf{p}_{11}$$

$$+ u(1 - u)(1 - v)^2\mathbf{p}_{01} + u^2(1 - v)^2\mathbf{p}_{02} + u(1 - u)v(1 - v)\mathbf{p}_{11} + u^2v(1 - v)\mathbf{p}_{12}$$

$$+ (1 - u)^2v(1 - v)\mathbf{p}_{10} + u(1 - u)v(1 - v)\mathbf{p}_{11} + (1 - u)^2v^2\mathbf{p}_{20} + u(1 - u)v^2\mathbf{p}_{21}$$

$$+ u(1 - u)v(1 - v)\mathbf{p}_{11} + u^2v(1 - v)\mathbf{p}_{12} + u(1 - u)v^2\mathbf{p}_{21} + u^2v^2\mathbf{p}_{22}\}$$

$$\mathbf{p}^2(u, v) = (1 - u)^2(1 - v)^2\mathbf{p}_{00} + 2u(1 - u)(1 - v)^2\mathbf{p}_{01} + u^2(1 - v)^2\mathbf{p}_{02} + 2(1 - u)^2v(1 - v)\mathbf{p}_{10}$$

$$+ 4uv(1 - u)(1 - v)\mathbf{p}_{11} + 2u^2v(1 - v)\mathbf{p}_{11} + (1 - u)^2v^2\mathbf{p}_{20} + 2u(1 - u)v^2\mathbf{p}_{21} + u^2v^2\mathbf{p}_{22}$$

테셀레이션(Tessellation)

- 베지어 곡면(Bezier Surface)

- $P = \{ \{ P_{00}, P_{01}, \dots, P_{0n} \}, \{ P_{10}, P_{11}, \dots, P_{1n} \}, \dots, \{ P_{m0}, P_{m1}, \dots, P_{mn} \} \}$
- 베지어 곡면: $S(u, v)$

$$S(u, v) = \sum_{i=0}^m \sum_{j=0}^n B_{m,i}(v) B_{n,j}(u) P_{ij}$$

$$(u, v) \in [0, 1] \times [0, 1]$$

$$\sum_{i=0}^m \sum_{j=0}^n B_{m,i}(v) B_{n,j}(u) = 1$$

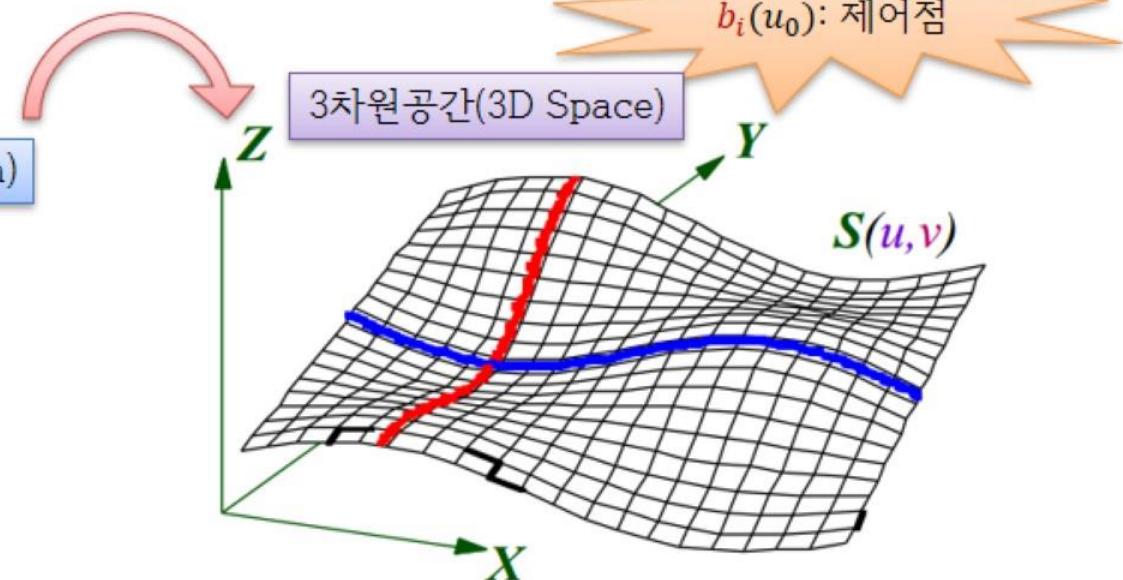
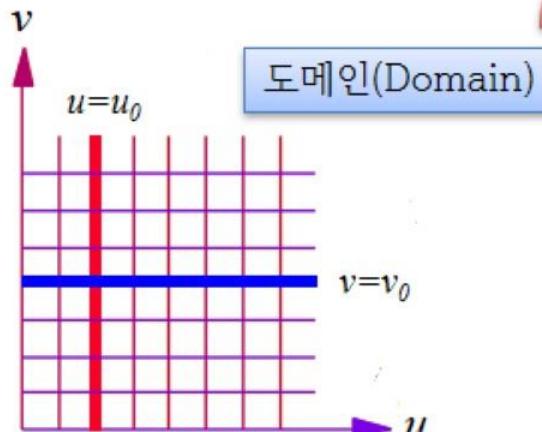
$$P(u) = \sum_{j=0}^n B_{n,j}(u) P_j$$

$$B_{n,j}(u) = \frac{n!}{j!(n-j)!} u^j (1-u)^{n-j}$$

$$S(u_0, v) = \sum_{i=0}^m \sum_{j=0}^n B_{m,i}(v) B_{n,j}(u_0) P_{ij} = \sum_{i=0}^m b_i B_{m,i}(v), \quad b_i = \sum_{j=0}^n B_{n,j}(u_0) P_{ij}$$

$$S(u, v_0) = \sum_{i=0}^m \sum_{j=0}^n B_{m,i}(v_0) B_{n,j}(u) P_{ij} = \sum_{j=0}^n c_j B_{n,j}(u), \quad c_j = \sum_{i=0}^m B_{m,i}(v_0) P_{ij}$$

$b_i(u_0)$: 제어점

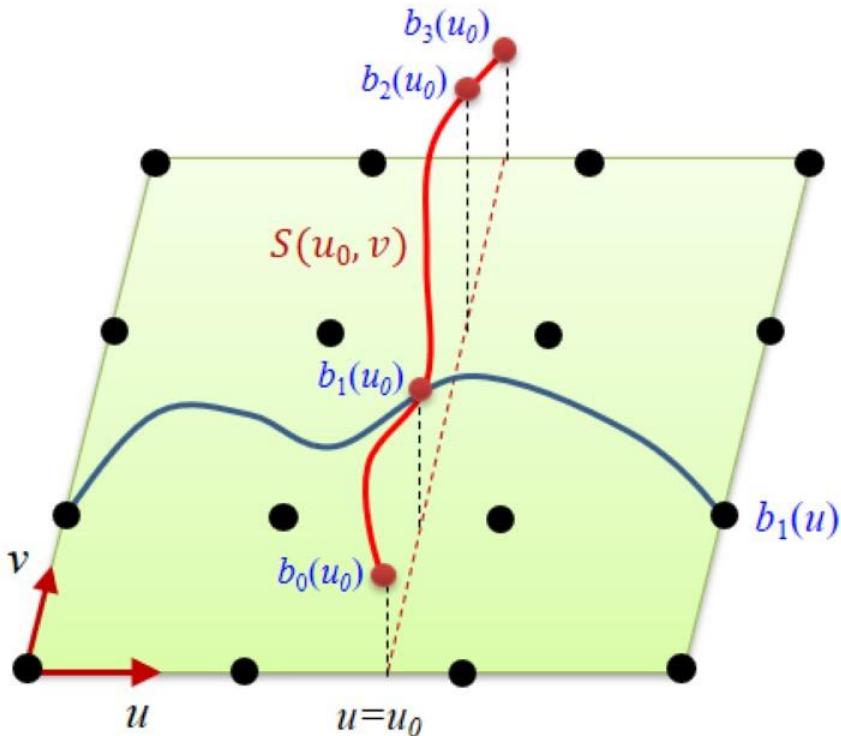


테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)

 - 4x4 제어점으로 구성된 패치

 - $S(u_0, v) = \sum_{i=0}^3 \sum_{j=0}^3 B_{3,i}(v) B_{3,j}(u_0) P_{ij} = \sum_{i=0}^3 b_i B_{3,i}(v), \quad b_i(u_0) = \sum_{j=0}^3 B_{3,j}(u_0) P_{ij}$
 - $S(u, v_0) = \sum_{i=0}^3 \sum_{j=0}^3 B_{3,i}(v_0) B_{3,j}(u) P_{ij} = \sum_{j=0}^3 c_j B_{3,j}(u), \quad c_j(v_0) = \sum_{i=0}^3 B_{3,i}(v_0) P_{ij}$



$$b_i(u) = \sum_{j=0}^n B_{n,j}(u) P_{ij}$$

$$S(u, v) = \sum_{i=0}^m \sum_{j=0}^n B_{m,i}(v) B_{n,j}(u) P_{ij} = \sum_{i=0}^m B_{m,i}(v) b_i(u)$$

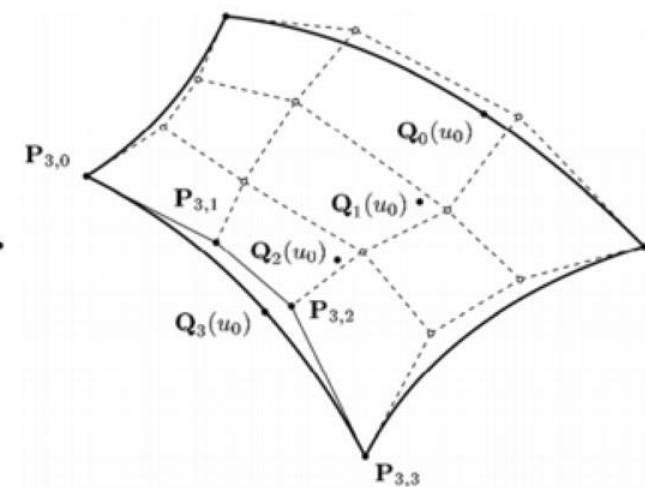
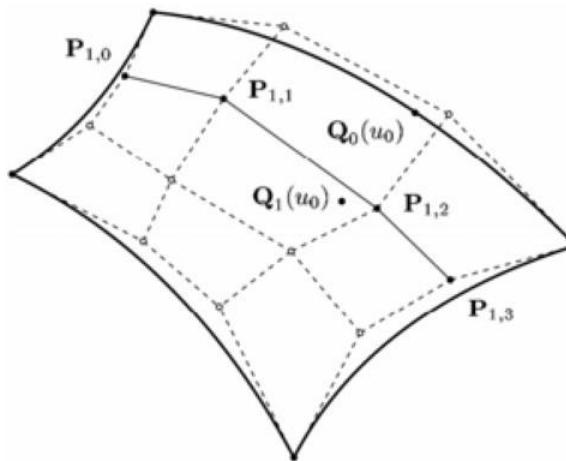
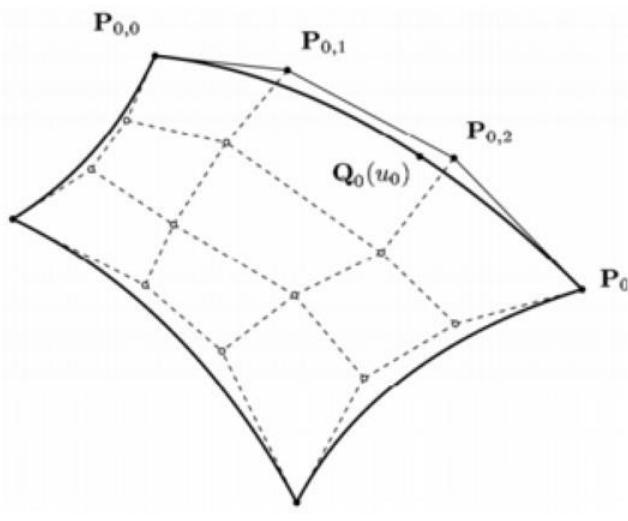
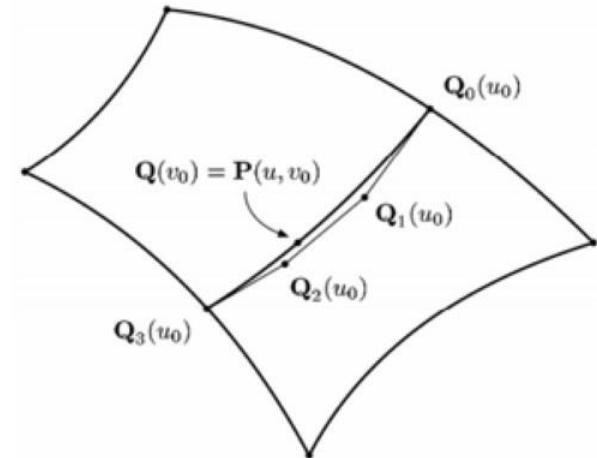
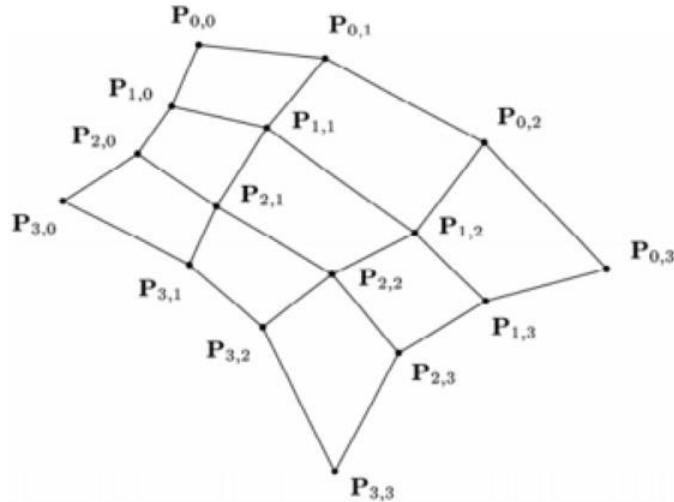
$b_i(u) = \sum_{j=0}^3 B_{3,j}(u) P_{ij}$: i -번째 행의 베지어 곡선

$S(u_0, v) = Q(v)$
 $Q(v) = \sum_{i=0}^3 B_{3,i}(v) b_i$: $b_i(u_0)$ 열의 베지어 곡선

$$S(u, v) = \sum_{i=0}^3 B_{3,i}(v) b_i(u), \quad b_i(u) = \sum_{j=0}^3 B_{3,j}(u) P_{ij}$$

테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)



테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)

- 4x4 제어점으로 구성된 패치

$$b_0(u) = \sum_{j=0}^3 B_{3,j}(u)P_{0j} = B_{3,0}(u)P_{00} + B_{3,1}(u)P_{01} + B_{3,2}(u)P_{02} + B_{3,3}(u)P_{03}$$

$$b_1(u) = \sum_{j=0}^3 B_{3,j}(u)P_{1j} = B_{3,0}(u)P_{10} + B_{3,1}(u)P_{11} + B_{3,2}(u)P_{12} + B_{3,3}(u)P_{13}$$

$$b_2(u) = \sum_{j=0}^3 B_{3,j}(u)P_{2j} = B_{3,0}(u)P_{20} + B_{3,1}(u)P_{21} + B_{3,2}(u)P_{22} + B_{3,3}(u)P_{23}$$

$$b_3(u) = \sum_{j=0}^3 B_{3,j}(u)P_{3j} = B_{3,0}(u)P_{30} + B_{3,1}(u)P_{31} + B_{3,2}(u)P_{32} + B_{3,3}(u)P_{33}$$

$$S(u, v) = \sum_{i=0}^3 B_{3,i}(v)b_i(u)$$
$$b_i(u) = \sum_{j=0}^3 B_{3,j}(u)P_{ij}$$

$$S(u, v) = \sum_{i=0}^3 B_{3,i}(v)b_i(u) = B_{3,0}(v)b_0(u) + B_{3,1}(v)b_1(u) + B_{3,2}(v)b_2(u) + B_{3,3}(v)b_3(u)$$
$$= B_{3,0}(v)(B_{3,0}(u)P_{00} + B_{3,1}(u)P_{01} + B_{3,2}(u)P_{02} + B_{3,3}(u)P_{03})$$
$$+ B_{3,1}(v)(B_{3,0}(u)P_{10} + B_{3,1}(u)P_{11} + B_{3,2}(u)P_{12} + B_{3,3}(u)P_{13})$$
$$+ B_{3,2}(v)(B_{3,0}(u)P_{20} + B_{3,1}(u)P_{21} + B_{3,2}(u)P_{22} + B_{3,3}(u)P_{23})$$
$$+ B_{3,3}(v)(B_{3,0}(u)P_{30} + B_{3,1}(u)P_{31} + B_{3,2}(u)P_{32} + B_{3,3}(u)P_{33})$$

테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)

- 4x4 제어점으로 구성된 패치

```
HS_CONSTANT HSConstant(InputPatch<VS_OUTPUT, 16> patch, uint nPatchID : SV_PrimitiveID)
{
    HS_CONSTANT output;
    output.fTessEdges[0] = 15;
    output.fTessEdges[1] = 15;
    output.fTessEdges[2] = 15;
    output.fTessEdges[3] = 15;
    output.fTessInsides[0] = 15;
    output.fTessInsides[1] = 15;
    return(output);
}

[domain("quad")]
[partitioning("integer")]
[outputtopology("triangle_cw")]
[outputcontrolpoints(16)]
[patchconstantfunc("HSConstant")]
[maxtessfactor(64.0f)]
HS_OUTPUT HS(InputPatch<VS_OUTPUT, 16> p, uint i : SV_OutputControlPointID, uint nID : SV_PrimitiveID)
{
    HS_OUTPUT output;
    output.pos = p[i].pos;
    return(output);
}
```

```
struct VS_INPUT
{
    float3 pos : POSITION;
};
```

```
struct VS_OUTPUT
{
    float3 pos : POSITION;
};
```

```
VS_OUTPUT VS(VS_INPUT input)
{
    VS_OUTPUT output;
    output.pos = input.pos;
    return(output);
}
```

```
struct HS_CONSTANT
{
    float fTessEdges[4] : SV_TessFactor;
    float fTessInsides[2] : SV_InsideTessFactor;
};
```

테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)

- 4x4 제어점으로 구성된 패치

```
float4 BernsteinCoefficient(float t)
```

```
{
    float tInv = 1.0f - t;
```

```
return(float4(tInv * tInv * tInv, 3.0f * t * tInv * tInv, 3.0f * t * t * t * tInv, t * t * t));
```

$$B_{3,0}(t) = (1-t)^3, B_{3,1}(t) = 3(1-t)^2t, B_{3,2}(t) = 3(1-t)t^2, B_{3,3}(t) = t^3$$

$$P(t) = \sum_{i=0}^n B_{n,i}(t)P_i, \quad B_{n,i}(t) = \frac{n!}{i!(n-i)!}t^i(1-t)^{n-i}$$

$$P(t) = (1-t)^3P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3$$

```
float3 CubicBezierSum(OutputPatch<HS_OUTPUT, 16> patch, float4 u, float4 v)
```

```
struct HS_OUTPUT
{
    float3 pos : POSITION;
};
```

```
{
    float3 sum = float3(0.0f, 0.0f, 0.0f);
```

```
sum = v.x * (u.x*patch[0].pos + u.y*patch[1].pos + u.z*patch[2].pos + u.w*patch[3].pos);
```

```
sum += v.y * (u.x*patch[4].pos + u.y*patch[5].pos + u.z*patch[6].pos + u.w*patch[7].pos);
```

```
sum += v.z * (u.x*patch[8].pos + u.y*patch[9].pos + u.z*patch[10].pos + u.w*patch[11].pos);
```

```
sum += v.w * (u.x*patch[12].pos + u.y*patch[13].pos + u.z*patch[14].pos + u.w*patch[15].pos);
```

```
return(sum);
```

$$b_0(u) = \sum_{j=0}^3 B_{3,j}(u)P_{0j} = B_{3,0}(u)P_{00} + B_{3,1}(u)P_{01} + B_{3,2}(u)P_{02} + B_{3,3}(u)P_{03}$$

$$b_1(u) = \sum_{j=0}^3 B_{3,j}(u)P_{1j} = B_{3,0}(u)P_{10} + B_{3,1}(u)P_{11} + B_{3,2}(u)P_{12} + B_{3,3}(u)P_{13}$$

$$b_2(u) = \sum_{j=0}^3 B_{3,j}(u)P_{2j} = B_{3,0}(u)P_{20} + B_{3,1}(u)P_{21} + B_{3,2}(u)P_{22} + B_{3,3}(u)P_{23}$$

$$b_3(u) = \sum_{j=0}^3 B_{3,j}(u)P_{3j} = B_{3,0}(u)P_{30} + B_{3,1}(u)P_{31} + B_{3,2}(u)P_{32} + B_{3,3}(u)P_{33}$$

$B_{3,0}(u)$	$B_{3,0}(v)$
$B_{3,1}(u)$	$B_{3,1}(v)$
$B_{3,2}(u)$	$B_{3,2}(v)$
$B_{3,3}(u)$	$B_{3,3}(v)$

$$S(u, v) = \sum_{i=0}^3 B_{3,i}(v)b_i(u) = B_{3,0}(v)b_0(u) + B_{3,1}(v)b_1(u) + B_{3,2}(v)b_2(u) + B_{3,3}(v)b_3(u)$$

테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)

- 4x4 제어점으로 구성된 패치

```
[domain("quad")]
DS_OUTPUT DS(HS_CONSTANT input, float2 uv : SV_DomainLocation, OutputPatch<HS_OUTPUT, 16> patch)
{
    DS_OUTPUT output;
    float4 uB = BernsteinCoefficient(uv.x);
    float4 vB = BernsteinCoefficient(uv.y);
    float3 position = CubicBezierSum(patch, uB, vB);

    matrix mtxWorldViewProjection = mul(gmtxWorld, gmtxView);
    mtxWorldViewProjection = mul(mtxWorldViewProjection, gmtxProjection);
    output.pos = mul(float4(position, 1.0f), mtxWorldViewProjection);

    return(output);
}
```

```
struct DS_OUTPUT
{
    float4 pos : SV_POSITION;
};
```

```
struct HS_OUTPUT
{
    float3 pos : POSITION;
};
```

```
struct HS_CONSTANT
{
    float fTessEdges[4] : SV_TessFactor;
    float fTessInsides[2] : SV_InsideTessFactor;
};
```

D3D_PRIMITIVE_TOPOLOGY_16_CONTROL_POINT_PATCHLIST

```
D3D12_INPUT_ELEMENT_DESC d3dInputLayoutDesc[1] =
{ "POSITION", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, 0, D3D12_INPUT_PER_VERTEX_DATA, 0 }
};
```

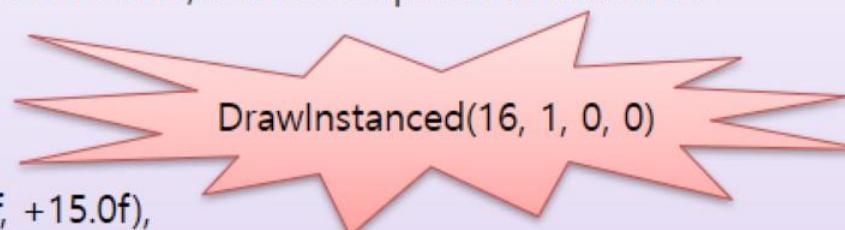
테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)

- 4x4 제어점으로 구성된 패치

```
void CQuadBezierMesh::CQuadBezierMesh(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList *pd3dCommandList)
{
    XMFLOAT3 pVertices[16] =
    {
        XMFLOAT3(-10.0f, -20.0f, +15.0f), XMFLOAT3(-5.0f, 0.0f, +15.0f),
        XMFLOAT3(+5.0f, 0.0f, +15.0f), XMFLOAT3(+10.0f, 0.0f, +15.0f),
        XMFLOAT3(-15.0f, 0.0f, +5.0f), XMFLOAT3(-5.0f, 0.0f, +5.0f),
        XMFLOAT3(+5.0f, +30.0f, +5.0f), XMFLOAT3(+15.0f, 0.0f, +5.0f),
        XMFLOAT3(-15.0f, 0.0f, -5.0f), XMFLOAT3(-5.0f, 0.0f, -5.0f),
        XMFLOAT3(+5.0f, 0.0f, -5.0f), XMFLOAT3(+15.0f, 0.0f, -5.0f),
        XMFLOAT3(-10.0f, +20.0f, -15.0f), XMFLOAT3(-5.0f, 0.0f, -15.0f),
        XMFLOAT3(+5.0f, 0.0f, -15.0f), XMFLOAT3(+25.0f, +20.0f, -15.0f)
    };
    m_nVertices = 16;
    m_nStride = sizeof(XMFLOAT3);
    m_d3dPrimitiveTopology = D3D_PRIMITIVE_TOPOLOGY_16_CONTROL_POINT_PATCHLIST;
    m_pd3dVertexBuffer = ::CreateBufferResource(pd3dDevice, pd3dCommandList, pVertices, m_nStride * m_nVertices, D3D12_HEAP_TYPE_DEFAULT, D3D12_RESOURCE_STATE_VERTEX_AND_CONSTANT_BUFFER, &m_pd3dVertexUploadBuffer);

    m_d3dVertexBufferView.BufferLocation = m_pd3dVertexBuffer->GetGPUVirtualAddress();
    m_d3dVertexBufferView.StrideInBytes = m_nStride;
    m_d3dVertexBufferView.SizeInBytes = m_nStride * m_nVertices;
}
```

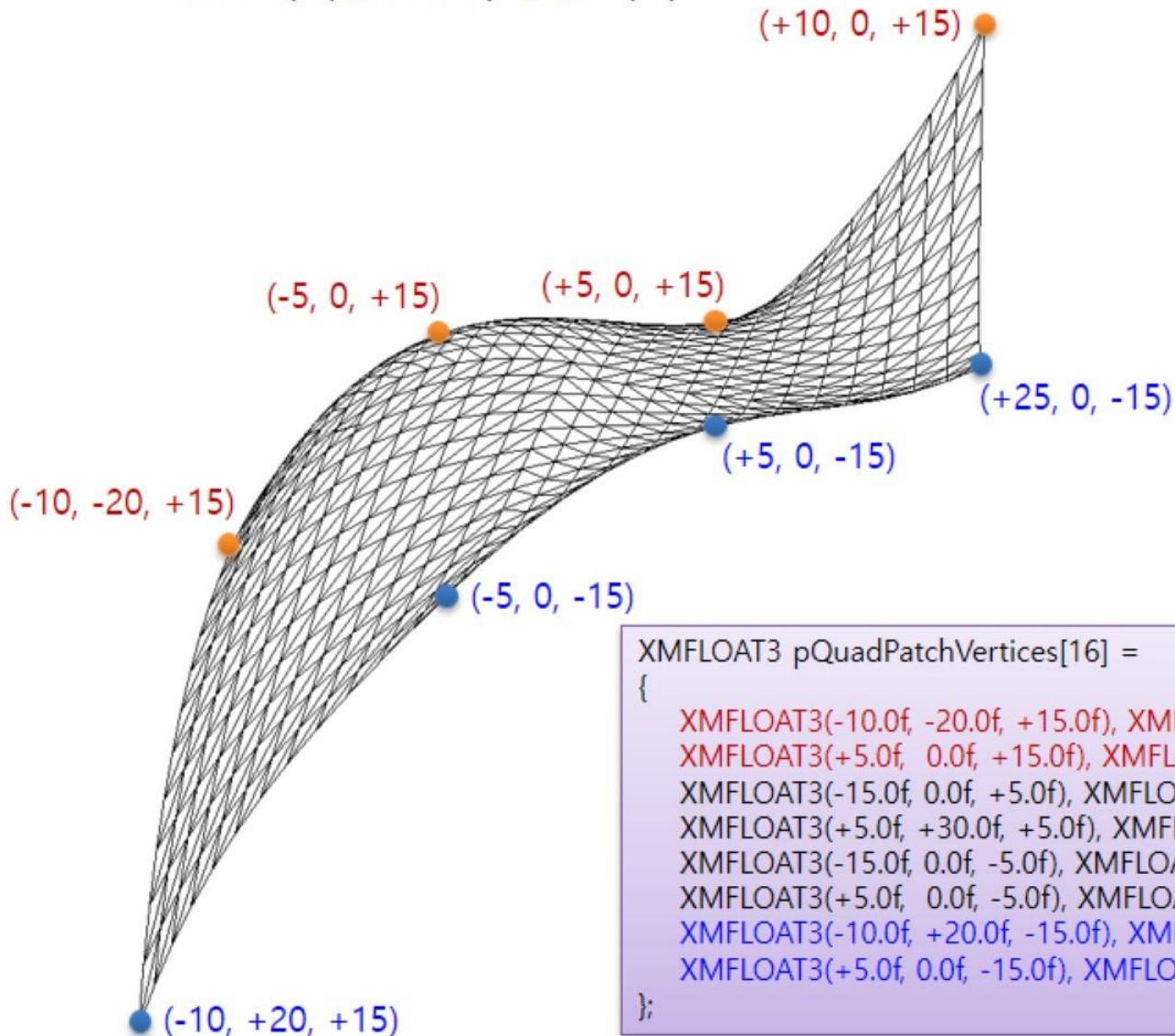


ID3D12Resource *m_pd3dQuadPatch;

테셀레이션(Tessellation)

- 큐빅 베지어 곡면(Cubic Bezier Surface)

- 4x4 제어점으로 구성된 패치



```
HS_CONSTANT output;  
output.fTessEdges[0] = 25;  
output.fTessEdges[1] = 25;  
output.fTessEdges[2] = 25;  
output.fTessEdges[3] = 25;  
  
output.fTessInsides[0] = 25;  
output.fTessInsides[1] = 25;
```

```
XMFLOAT3 pQuadPatchVertices[16] =  
{  
    XMFLOAT3(-10.0f, -20.0f, +15.0f), XMFLOAT3(-5.0f, 0.0f, +15.0f),  
    XMFLOAT3(+5.0f, 0.0f, +15.0f), XMFLOAT3(+10.0f, 0.0f, +15.0f),  
    XMFLOAT3(-15.0f, 0.0f, +5.0f), XMFLOAT3(-5.0f, 0.0f, +5.0f),  
    XMFLOAT3(+5.0f, +30.0f, +5.0f), XMFLOAT3(+15.0f, 0.0f, +5.0f),  
    XMFLOAT3(-15.0f, 0.0f, -5.0f), XMFLOAT3(-5.0f, 0.0f, -5.0f),  
    XMFLOAT3(+5.0f, 0.0f, -5.0f), XMFLOAT3(+15.0f, 0.0f, -5.0f),  
    XMFLOAT3(-10.0f, +20.0f, -15.0f), XMFLOAT3(-5.0f, 0.0f, -15.0f),  
    XMFLOAT3(+5.0f, 0.0f, -15.0f), XMFLOAT3(+25.0f, +20.0f, -15.0f)  
};
```

테셀레이션(Tessellation)

- 지형(Terrain)

```
Texture2D gtxtHeightMap : register(t2);
```

```
VS_OUTPUT VS(VS_INPUT input)
{
    VS_OUTPUT output;
    output.posW = input.posL;
    output.posW.y = gtxtHeightMap.SampleLevel(gssHeightMap, input.uv, 0).r;
    output.uv = input.uv;
    return(output);
}
```

```
struct VS_INPUT
{
    float3 posL : POSITION;
    float2 uv : TEXCOORD0;
};
```

```
struct VS_OUTPUT
{
    float3 posW : POSITION;
    float2 uv : TEXCOORD0;
};
```

```
[domain("quad")]
[partitioning("fractional_even")]
[outputtopology("triangle_cw")]
[outputcontrolpoints(4)]
[patchconstantfunc("ConstantHS")]
[maxtessfactor(64.0f)]
```

```
HS_OUTPUT HS(InputPatch<VS_OUTPUT, 4> input, uint i : SV_OutputControlPointID)
{
    HS_OUTPUT output;
    output.posW = input[i].posW;
    output.uv = input[i].uv;
    return(output);
}
```

```
struct HS_OUTPUT
{
    float3 posW : POSITION;
    float2 uv : TEXCOORD0;
};
```

테셀레이션(Tessellation)

- 지형

```
HSC_OUTPUT ConstantHS(InputPatch<VS_OUTPUT, 4> input, uint nPatchID : SV_PrimitiveID)
```

```
{
```

```
    HSC_OUTPUT output;
```

```
    float3 e0 = 0.5f * (input[0].posW + input[2].posW);  
    float3 e1 = 0.5f * (input[0].posW + input[1].posW);  
    float3 e2 = 0.5f * (input[1].posW + input[3].posW);  
    float3 e3 = 0.5f * (input[2].posW + input[3].posW);
```

```
    output.fTessEdges[0] = CalculateTessFactor(e0);  
    output.fTessEdges[1] = CalculateTessFactor(e1);  
    output.fTessEdges[2] = CalculateTessFactor(e2);  
    output.fTessEdges[3] = CalculateTessFactor(e3);
```

```
    float3 c = 0.25f * (input[0].posW + input[1].posW + input[2].posW + input[3].posW);  
    output.fTessInsides[0] = CalculateTessFactor(c);  
    output.fTessInsides[1] = output.fTessInsides[0];
```

```
    return(output);
```

```
}
```

```
float CalculateTessFactor(float3 p)
```

```
{
```

```
    float fDistToCamera = distance(p, gvCameraPosition);  
    float s = saturate((fDistToCamera - 10.0f) / (200.0f - 10.0f));  
    return(pow(2, lerp(20.0f, 4.0f, s)));
```

```
}
```

```
struct HSC_OUTPUT
```

```
{
```

```
    float fTessEdges[4] : SV_TessFactor;  
    float fTessInsides[2] : SV_InsideTessFactor;  
};
```

테셀레이션(Tessellation)

- 지형

```
struct DS_OUTPUT
{
    float4 posH : SV_POSITION;
    float3 posW : POSITION;
    float2 uv : TEXCOORD0;
    float2 uvDetail : TEXCOORD1;
};
```

```
struct HS_OUTPUT
{
    float3 posW : POSITION;
    float2 uv : TEXCOORD0;
};
```

```
struct HSC_OUTPUT
{
    float fTessEdges[4] : SV_TessFactor;
    float fTessInsides[2] : SV_InsideTessFactor;
};
```

```
[domain("quad")]
DS_OUTPUT DS(HSC_OUTPUT input, float2 uv : SV_DomainLocation, OutputPatch<HS_OUTPUT, 4> quad)
{
    DS_OUTPUT output;

    output.posW = lerp(lerp(quad[0].posW, quad[1].posW, uv.x), lerp(quad[2].posW, quad[3].posW, uv.x), uv.y);
    output.uv = lerp(lerp(quad[0].uv, quad[1].uv, uv.x), lerp(quad[2].uv, quad[3].uv, uv.x), uv.y);
    output.uvDetail = output.uv * gvTextureScale;
    float2 gvTextureScale = 50.0f;

    output.posW.y = gtxtHeightMap.SampleLevel(gssHeightMap, output.uv, 0).r;
    output.posH = mul(float4(output.posW, 1.0f), gmtxViewProjection);

    return(output);
}
```

테셀레이션(Tessellation)

- 지형

```
struct DS_OUTPUT
{
    float4 posH : SV_POSITION;
    float3 posW : POSITION;
    float2 uv : TEXCOORD0;
    float2 uvDetail : TEXCOORD1;
};
```

```
float gTexelCellSpaceU;
float gTexelCellSpaceV;
float gWorldCellSpace;
float2 gvTextureScale = 50.0f;
```

```
float4 PS(DS_OUTPUT input) : SV_Target
{
    float2 leftTex = input.uv + float2(-gTexelCellSpaceU, 0.0f);
    float2 rightTex = input.uv + float2(gTexelCellSpaceU, 0.0f);
    float2 bottomTex = input.uv + float2(0.0f, gTexelCellSpaceV);
    float2 topTex = input.uv + float2(0.0f, -gTexelCellSpaceV);

    float leftY = gtxtHeightMap.SampleLevel(gssHeightMap, leftTex, 0).r;
    float rightY = gtxtHeightMap.SampleLevel(gssHeightMap, rightTex, 0).r;
    float bottomY = gtxtHeightMap.SampleLevel(gssHeightMap, bottomTex, 0).r;
    float topY = gtxtHeightMap.SampleLevel(gssHeightMap, topTex, 0).r;

    float3 tangent = normalize(float3(2.0f * gWorldCellSpace, rightY - leftY, 0.0f));
    float3 bitangent = normalize(float3(0.0f, bottomY - topY, -2.0f * gWorldCellSpace));
    float3 normalW = cross(tangent, bitangent);
    ...
}
```

테셀레이션(Tessellation)

- 퍼거슨 삼차 곡선(Ferguson's Parametric Cubic Curves)

- 제어점: P_0, P_1 , 접선의 기울기: P'_0, P'_1

접선의 기울기 P'_0, P'_1 을 가진 P_0, P_1 를 지나는 큐빅(3차) 파라메터 곡선을 정의

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$P(0) = a_0$$

$$P(1) = a_0 + a_1 + a_2 + a_3$$

$$P'(0) = a_1$$

$$P'(1) = a_1 + 2a_2 + 3a_3$$

$$a_0 = P(0)$$

$$a_1 = P'(0)$$

$$a_2 = 3\{P(1) - P(0)\} - 2P'(0) - P'(1)$$

$$a_3 = 2\{P(0) - P(1)\} + P'(0) + P'(1)$$

$$0 \leq t \leq 1$$

$$P(0) = P_0$$

$$P(1) = P_1$$

$$P'(0) = P'_0$$

$$P'(1) = P'_1$$

$$\begin{aligned} P(t) &= P(0) + P'(0)t + \{3\{P(1) - P(0)\} - 2P'(0) - P'(1)\}t^2 + \{2\{P(0) - P(1)\} + P'(0) + P'(1)\}t^3 \\ &= (1 - 3t^2 + 2t^3)P(0) + (3t^2 - 2t^3)P(1) + (t - 2t^2 + t^3)P'(0) + (-t^2 + t^3)P'(1) \end{aligned}$$

$$P(t) = (1 \ t \ t^2 \ t^3) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{bmatrix} = \begin{bmatrix} 1 - 3t^2 + 2t^3 \\ 3t^2 - 2t^3 \\ t - 2t^2 + t^3 \\ -t^2 + t^3 \end{bmatrix}^T \begin{bmatrix} P(0) \\ P(1) \\ P'(0) \\ P'(1) \end{bmatrix}$$

테셀레이션(Tessellation)

- Catmull-Rom 스플라인 곡선

- 제어점: $\{P_0, P_1, \dots, P_n\}$

모든 제어점을 지나는 3차 파라메터 곡선을 정의

P_i 와 P_{i+1} 의 두 개의 제어점 사이를 3차 곡선으로 표현

각 제어점에서 이전 점과 다음 점을 연결한 선분을 접선으로 사용하여 곡선을 정의

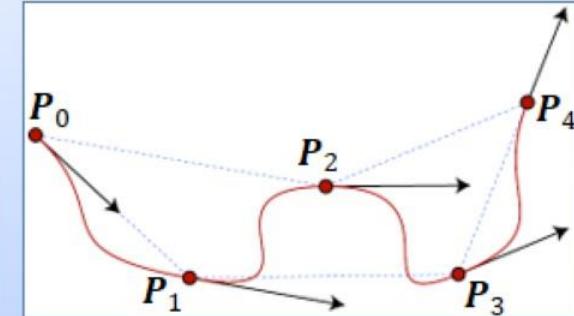
제어점 P_i 의 접선: $\frac{1}{2}(P_{i+1} - P_{i-1})$, 제어점 P_{i+1} 의 접선: $\frac{1}{2}(P_{i+2} - P_i)$

접선의 기울기 P'_i, P'_{i+1} 을 가진 P_i, P_{i+1} 를 지나는 큐빅(3차) 파라메터 곡선

퍼거슨 삼차 곡선을 사용하여 표현할 수 있음

P'_i, P'_{i+1} 을 표현하려면 P_{i-1}, P_{i+2} 이 필요함(제어점: $P_{i-1}, P_i, P_{i+1}, P_{i+2}$)

$$\begin{aligned}
 P(t) &= (1 \ t \ t^2 \ t^3) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ \frac{1}{2}(P_{i+1} - P_{i-1}) \\ \frac{1}{2}(P_{i+2} - P_i) \end{bmatrix} \\
 &= (1 \ t \ t^2 \ t^3) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix} \\
 &= \frac{1}{2} (1 \ t \ t^2 \ t^3) \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}t + t^2 - \frac{1}{2}t^3 \\ 1 - \frac{5}{2}t^2 + \frac{3}{2}t^3 \\ \frac{1}{2}t + 2t^2 - \frac{3}{2}t^3 \\ -\frac{1}{2}t^2 + \frac{1}{2}t^3 \end{bmatrix}^T \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}
 \end{aligned}$$



$0 \leq t \leq 1$

테셀레이션(Tessellation)

- Catmull-Rom 스플라인 곡선

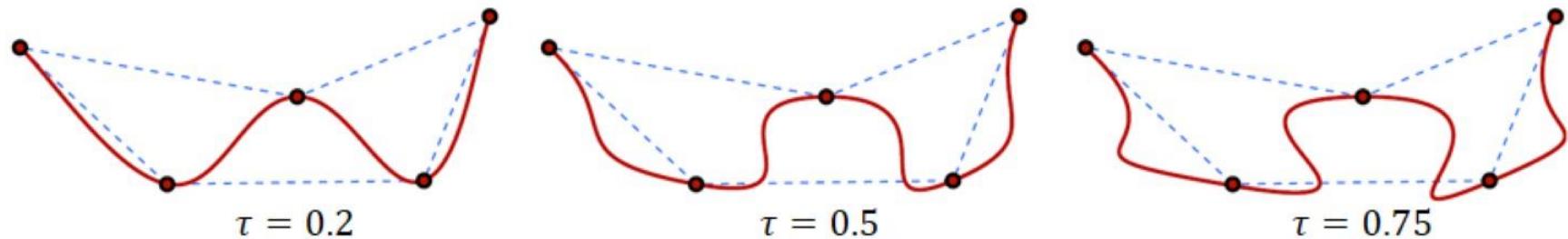
- 제어점: $\{P_0, P_1, \dots, P_n\}$

텐션(Tension): 제어점에서 곡선이 휘는 정도(τ)

일반적으로 $\frac{1}{2}$ 을 사용

제어점 P_i 의 접선: $\tau(P_{i+1} - P_{i-1})$

제어점 P_{i+1} 의 접선: $\tau(P_{i+2} - P_i)$



$$P(t) = (1 \ t \ t^2 \ t^3) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ \tau(P_{i+1} - P_{i-1}) \\ \tau(P_{i+2} - P_i) \end{bmatrix}$$

$$= (1 \ t \ t^2 \ t^3) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\tau & 0 & \tau & 0 \\ 0 & -\tau & 0 & \tau \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

$0 \leq t \leq 1$

$$= (1 \ t \ t^2 \ t^3) \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix} = \begin{bmatrix} -\tau t + 2\tau t^2 - \tau t^3 \\ 1 + (\tau - 3)t^2 + (2 - \tau)t^3 \\ \tau t + (3 - 2\tau)t^2 + (\tau - 2)t^3 \\ -\tau t^2 + \tau t^3 \end{bmatrix}^T \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

테셀레이션(Tessellation)

- 헤르밋 삼차 곡선(Hermite Parametric Cubic Curves)

- 제어점: $\{P_0, P_1, \dots, P_n\}$
- 제어점: P_k, P_{k+1} , 접선의 기울기: P'_k, P'_{k+1}
접선의 기울기 P'_k, P'_{k+1} 을 가진 P_k, P_{k+1} 를 지나는 큐빅 파라메터 곡선($0 \leq t \leq 1$)
$$P(t) = (1 - 3t^2 + 2t^3)P_k + (3t^2 - 2t^3)P_{k+1} + (t - 2t^2 + t^3)P'_k + (-t^2 + t^3)P'_{k+1}$$
$$P(t) = h_{00}(t)P_k + h_{01}(t)P_{k+1} + h_{10}(t)P'_k + h_{11}(t)P'_{k+1}$$

- 구간 (P_k, P_{k+1}) 의 임의의 점 p

$$t = \frac{(p - P_k)}{(P_{k+1} - P_k)}$$

$$P(p) = h_{00}(t)P_k + h_{01}(t)P_{k+1} + h_{10}(t)(P_{k+1} - P_k)P'_k + h_{11}(t)(P_{k+1} - P_k)P'_{k+1}$$

$h_{00}(t)$	$(1 - 3t^2 + 2t^3)$	$B_0(t) + B_1(t)$
$h_{01}(t)$	$(3t^2 - 2t^3)$	$B_3(t) + B_2(t)$
$h_{10}(t)$	$(t - 2t^2 + t^3)$	$\frac{1}{3}B_1(t)$
$h_{11}(t)$	$(-t^2 + t^3)$	$-\frac{1}{3}B_2(t)$

$$B_k(t) = \binom{3}{k} t^k (1-t)^{3-k}$$