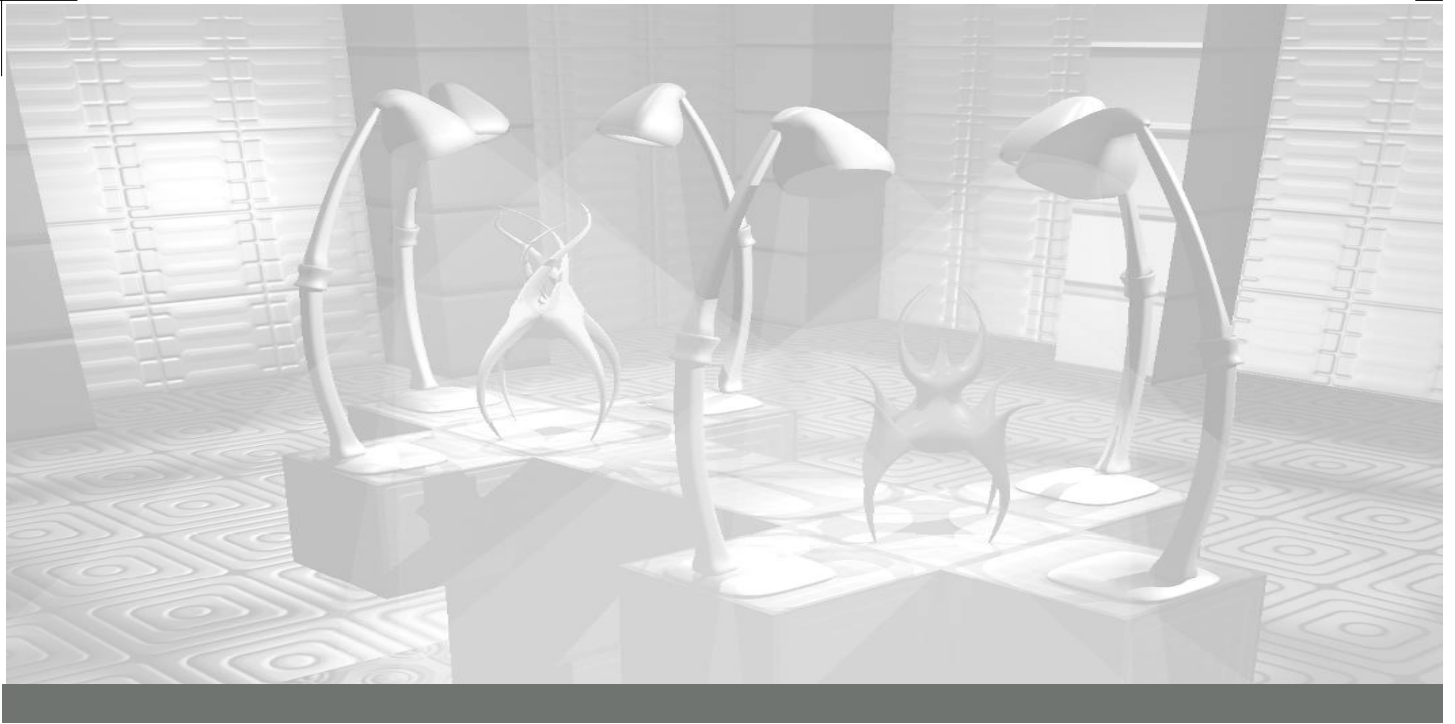


1



1.

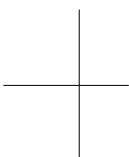
Dean Calver

「Direct3D ShaderX: &」
. DirectX 9 (stream)
가 .

DirectX 9

1.1 가
가

3.0 , 가



가

DirectX 8

가 , 가 () 가

DirectX 8

(D3DDEVCAPS2_STREAMOFFSET

), DirectX 9

DirectX 7

, FVF

, DirectX 9

, D3DDEVCAPS2_VERTXELEMENTSCANSHARESTREAMOFFSET 가

DWORD (4

)

DirectX 9

가

가



No te

DirectX 9

2.0

가

2.0

(ATI

).



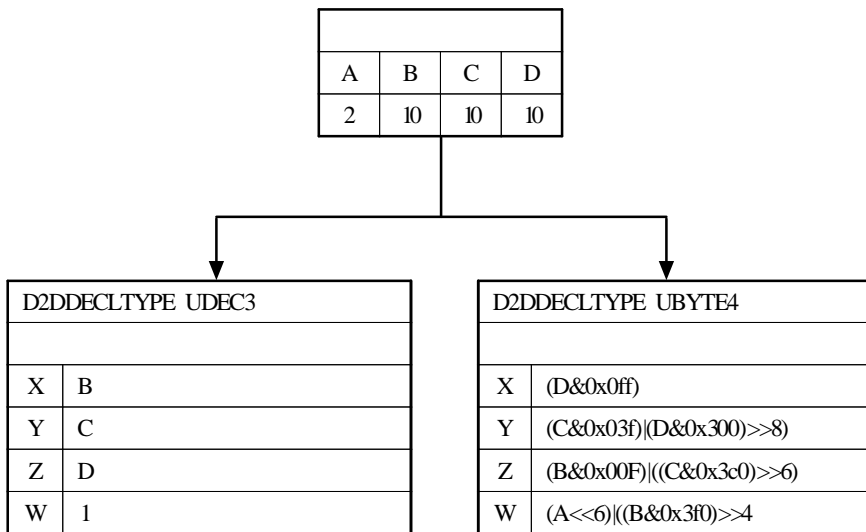
DirectX 8 가 (signed),
(unsigned), (normalized),

D3DCOLOR	4	32	8	[0,1]	N	a
UBYTE4	4	32	8	[0,255]	Y	
UBYTE4N	4	32	8	[0,1]	Y	
UDEC3	3	32	10	[0,1024]	Y	b
DEC3N	3	32	10	[-1,1]	Y	b
SHORT2	2	32	16	[-32768,32767]	N	
SHORT4	4	64	16	[-32768,32767]	N	
USHORT2N	2	32	16	[0,1]	Y	
USHORT4N	4	64	16	[0,1]	Y	
SHORT2N	2	32	16	[-1,1]	Y	
SHORT4N	4	64	16	[-6.55e4,6.55e4]	Y	
FLOA16_2	2	32	16	[-6.55e4,6.55e4]	Y	c
FLOAT16_4	4	64	16	[-3.48e38,3.48e38]	Y	c
FLOAT1	1	32	32	[-3.48e38,3.48e38]	N	d
FLOAT2	2	64	32	[-3.48e38,3.48e38]	N	d
FLOAT3	3	96	32	[-3.48e38,3.48e38]	N	d
FLOAT4	4	128	32	[-3.48e38,3.48e38]	N	d

- a) D3DCOLOR . ARGB RGBA가 .
- b) 2
- c) float 16 OpenEXR , nVidia PIXAR가 . D3DXFLOAT 16
(OpenEXR SDK).
- d) float C IEEE754 .

(DEC3 32). D3DCAPS9
 .DeclType , D3DTCAPS_datatype
 , D3DDECLTYPE_datatype (datatype
 가).

DEC3N UDEC3 , ()
 , , 가)
 (가)
).



[1-1]

UDEC3 ()

```

NORMAL0
NORMAL1          UBYTE4
                UDEC3
D3DVERTEXELEMENT9 decl[] =

```

```

{
    //          , ' ' UDEC3
    { 0,          //
      0,          //
      D3DDECLTYPE_UDEC3, //
      D3DDECLMETHOD_DEFAULT, //
      D3DDECLUSAGE_NORMAL, // (
      0          // (n
    },
    //          ,          UBYTE4
    { 0,          //
      0,          //
      D3DDECLTYPE_UBYTE4, //
      D3DDECLMETHOD_DEFAULT, //
      D3DDECLUSAGE_NORMAL, // (
      1          // (n
    },
    D3DDECL_END()
};

```

, 2⁶(64) (floor)

, 0 3

(mov

).

```

struct VS_INPUT
{
    float4 normal : NORMAL0,
    float4 enc2Bit : NORMAL1
};

void main( VS_INPUT input )
{

```

```

//
float3 normal = input.normal;
// 2 (0-3)
float two_bits = floor(input.enc2Bit.w / 64.0);
}

```

DEC3N 10, 10, 12 가 (Direct3D ShaderX 가 “ ” (eigen), (dominant) 가 (64 (SHORT4) 32 z

```

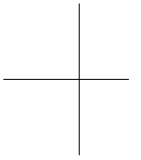
struct VS_INPUT
{
    float4 position : POSITION0,
    float4 enc2Bit : POSITION1
};

void main( VS_INPUT input )
{
    // 10, 10, 10
    float3 cpos = input.position;
    // 2 (0-3)
    float two_bits = floor(input.enc2Bit.w / 64.0);
    // 0-1 가
    cpos.z = (cpos.z + two_bits) * 0.25;
}

```

```
//
float4 pos = mul ( float4(cpos,1), InvCompressionTransform );
}
```

Tom Forsyth가 “ (subdivision surfaces) ” 가 , (tessellation) 가 , CPU가 , (()). 가 , 가 (barycentric) , 가 가 . : . : , . 가 가 . Mike Dogget Tom Forsyth [2]. , Tom



Forsyth
[1].

ID
가 Direct3D ShaderX
가

CPU

가 (FLOAT 1)
가 . 8
, UBYTE4가
가

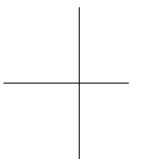


Note

, DirectX 9
GPU (가). Mike Doggett OpenGL DirectX 9
to-vertex-array GPU (uber-buffer) render-

(pre - filter ring)

가
가 ,
() ()
) 가 .



UBYTE4

가).

가 가 (/

(barycentric) (, (areal))
 ()
 가
 ()

(k i j 가) .

가 가 .

```
//          HLSL
float3 VertexPos[3 * NUMBASE_TRIANGLE];

void main(float3 vertexStream : POSITION0)
{
    float i = vertexStream.x;
    float j = vertexStream.y
    float k = 1.0 - i - j;
    float baseIndex = vertexStream.z * 256; //
    float3 pos =      i*VertexPos[ (baseIndex*3) + 0 ] +
                    j*VertexPos[ (baseIndex*3) + 1 ] +
                    k*VertexPos[ (baseIndex*3) + 2 ];
}
```

N-

N- (PN [3]) (bicubic) . N- 가 ,
 , 가 ,
 , 가 ,
 CPU GPU , PC
 CPU
 가 (N- ,
 39 18 가), CPU 가 .

```
float3 VertexPos[3 * NUMBASE_TRIANGLE];
float3 VertexNormals[3 * NUMBASE_TRIANGLE];

//
float3 b300,b030,b003, b210,b120,b021, b201,b102,b012, b111;
```

```

float3 n200, n020, n002;
void generateControlPointsWithNearNormals(float baseIndex);
{
    float3 v0 = VertexPos[ (baseIndex*3) + 0 ];
    float3 v1 = VertexPos[ (baseIndex*3) + 1 ];
    float3 v2 = VertexPos[ (baseIndex*3) + 2 ];
    float3 n0 = VertexNormal [ (baseIndex*3) + 0 ];
    float3 n1 = VertexNormal [ (baseIndex*3) + 1 ];
    float3 n2 = VertexNormal [ (baseIndex*3) + 2 ];
    //
    //      CD          ATI      PN      [3]
    float3 edge = v1 - v0;
    // E - (E.N)N
    float3 tangent1 = edge;
    float tmpf = dot( tangent1, n0 );
    tangent1 -= n0 * tmpf;
    b210 = v0 + (tangent1 * rcp3);
}

void evaluatePatchNearNormal(float i, float j, out float3 pos, out float3
norm)
{
    float k = 1 - i - j;
    float k2 = k * k;
    float k3 = k2 * k;
    float i2 = i * i;
    float i3 = i2 * i;
    float j2 = j * j;
    float j3 = j2 * j;

    //
    pos = (b300*k3) + (b030*u3) + (b003*v3) +
          (b210*3*k2*i) + (b120*3*k*i2) + (b201*3*k2*j) +
          (b021*3*i2*j) + (b102*3*k*j2) + (b012*3*i2*j) +
          (b111*6*k*i*j);

    //
    norm = (w * n200) + (i * n020) + (j * n002);
}

void main(float3 vertexStream : POSITION0)
{

```



```

};

void main( VS_IN input )
{
    float i = input.barycentric.x;
    float j = input.barycentric.y;
    float k = 1.0 - i - j;
    float i0 = input.indices.x * 256;
    float i1 = input.indices.y * 256;
    float i2 = input.indices.z * 256;

    float3 pos = i*VertexPos[i0] + j*VertexPos[i1] + k*VertexPos[i2];
    float3 normal = i*VertexNormal[i0] + j*VertexNormal[i1] + k*VertexNormal[i2];
    float2 uv = i*VertexUV[i0] + j*VertexUV[i1] + k*VertexUV[i2];

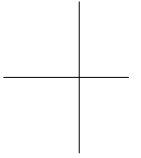
    normal = normalize( normal );
    pos = pos + input.displacement * normal * MAX_DISPLACEMENT_HEIGHT;
}

```

$[0, 1]$
 $\frac{1}{256}$ (가)가 $\frac{1}{256}$ (가)가
 $\frac{1}{8}$ (가)가 $\frac{1}{8}$ (가)가
 $\frac{1}{3}$ (가)가

(perturbed)

가



가 . ,
 , .
 , 가 ()
) , ()
) .
 , 가 가
 , “ ”
 , ([2]
), 가 .

[1] Forsyth, Tom, Displacement Mapping, 『ShaderX²: Shader Programming Tips & Tricks with DirectX 9』, Wolfgang Engel , Wordware Publishing, Inc., 2004, pp. 73-86.
 1 “7. ” .

[2] Doggett, Mike Tom Forsyth, Displacement Mapping, GDC 2003.

[3] Vlachos, A. J. Peters, C. Boyd, J. Mitchell, Curved PN Triangles,
<http://www.ati.com/developer/CurvedPNTriangles.pdf>.

