

```

struct vpInput
{
    float4 pos    : POSITION;
    float4 normal : NORMAL;
};

struct fvInput
{
    float4 pos      : POSITION;
    float4 worldPosition : TEXCOORD0;
};

// Vertex Program
fvInput mainVP(vpInput IN
               , uniform float4x4 world : WORLD
               , uniform float4x4 worldViewProjection : WORLDVIEWPROJECTION)
{
    fvInput OUT;
    OUT.pos = mul(worldViewProjection, IN.pos);
    OUT.worldPosition = mul(world, IN.pos);
    return OUT;
}

// Fragment Program
void mainFP(fvInput IN, out float4 color : COLOR)
{
    color = IN.worldPosition;
}

```

Figure 1 - Vertex and fragment shader programs, in Cg language, for generating a position map

```

struct vpInput
{
    float4 pos    : POSITION;
    float2 texCoord : TEXCOORD0;
};

struct fvInput
{
    float4 pos    : POSITION;
    float2 texCoord: TEXCOORD0;
};

// Vertex Program
fvInput mainVP(vpInput IN, uniform float4x4 vp : VIEWPROJECTION)
{
    fvInput OUT;
    OUT.pos = mul(vp, IN.pos);
    OUT.texCoord = IN.texCoord;
    return OUT;
};

// Fragment Program
void mainFP(fvInput IN
    , in uniform sampler2D videoTexture : SAMPLERVIDEOTEXTURE
    , in uniform float4 blendColor
    , in uniform float4x4 view : VIEW
    , in uniform float4x4 videoCameraWorldViewMatrix
    , out float4 color : COLOR )
{
    if (IN.texCoord.x < 0.03 || IN.texCoord.y < 0.03 ||
        IN.texCoord.x > 0.97 || IN.texCoord.y > 0.97)
    {
        color = blendColor;
        return;
    }

    float4 videoColor = tex2D(videoTexture, IN.texCoord);

    float angle = degrees(acos(dot(normalize(view._m20_m21_m22),
normalize(videoCameraWorldViewMatrix._m20_m21_m22))));
    float blendColorWeight = saturate((angle - 30.0f) / 10.0f); // 0.0 at 30 degrees and 1.0
at 40 degrees.
    color = lerp(videoColor, blendColor, blendColorWeight);
};

```

Figure 2 - Vertex and fragment shader programs, in Cg language, for rendering a camera projection mesh

```

struct vpInput
{
    float4 pos    : POSITION;
    float2 panTilt: TEXCOORD0;
};

struct fvInput
{
    float4 pos    : POSITION;
    float2 panTilt : TEXCOORD0;
};

// Vertex Program
fvInput mainVP(vpInput IN
               , uniform float4x4 vp : VIEWPROJECTION)
{
    fvInput OUT;
    OUT.pos = mul(vp, IN.pos);
    OUT.panTilt = IN.panTilt;
    return OUT;
}

// Fragment Program
void mainFP(fvInput IN
            , in uniform float4 blendColor
            , in uniform float4 panTiltRange
            , out float4 color : COLOR)
{
    float panAngle = IN.panTilt.x;
    float tiltAngle = IN.panTilt.y;

    bool inPtzRange = panAngle >= panTiltRange.x &&
                     panAngle <= panTiltRange.y &&
                     tiltAngle >= panTiltRange.z &&
                     tiltAngle <= panTiltRange.w;

    if (inPtzRange)
        color = blendColor;
    else
        color = float4(0, 0, 0, 0);
}

```

Figure 3 - Vertex and fragment shader programs, in Cg language, for rendering the coverage of a PTZ video surveillance camera