```
//--------------------------------------------------------------------------
// Additional material for the paper
//    "A Hybrid Physical/Device-Space Approach for Spatio-Temporally
//     Coherent Interactive Texture Advection on Curved Surfaces"
//
// Excerpt of the HLSL/fx files for the GPU programs

string EffectDescription = "advection in hybrid physical / device space";

//--------------------------------------------------------------------------
// Parameters and constants
// Textures:
texture g_tFlowField2D;
texture g_tCoordinates;
// Vector parameters
float4  g_vStepSize;            // Integration step size
// GPU state variables
float4x4 g_mMVP;                    // Model * View * Projection matrix

//--------------------------------------------------------------------------
// Samplers
// Bilinear lookup in image-space flow field
sampler FlowField2dSMP = sampler_state
{
    texture = <g_tFlowField2D>;
    AddressU  = CLAMP;
    AddressV  = CLAMP;
    MIPFILTER = NONE;
    MINFILTER = LINEAR;
    MAGFILTER = LINEAR;
};
// Nearest-neighbor sampling for current coordinates
sampler CoordinatesSMP = sampler_state
{
    texture = <g_tCoordinates>;
    AddressU  = CLAMP;
    AddressV  = CLAMP;
    MIPFILTER = NONE;
    MINFILTER = POINT;
    MAGFILTER = POINT;
};

//--------------------------------------------------------------------------
// Vertex shader output for domain-filling quad
struct VS_OUTPUT_Domain
{
    float4 Position       : POSITION;   // vertex position
    float2 TextureCoord   : TEXCOORD0;  // tex coords
};

//--------------------------------------------------------------------------
// Common pixel shader output
struct PS_OUTPUT_IntegrateTwoPhase
{
    float4 RGBA     : COLOR0;
};

//--------------------------------------------------------------------------
// Integrate and accumulate
//    coords tex: xyz = initial object space coords
//                w    = measure for choosing noise frequency
//    property tex: xy=0

PS_OUTPUT_IntegrateTwoPhase IntegrateCoordPS ( VS_OUTPUT_Domain In )
{
    PS_OUTPUT_IntegrateTwoPhase Output;
    // Lookup current coordinate texture with nearest-neighbor sampling
    float4 coordsOld = tex2D(CoordinatesSMP, In.TextureCoord);

    // Integration step:
    // (1) Transform from model space coords into device/image space
    float4 coordsProj;
    float4 coordsOrig = coordsOld;
    coordsOrig.w = 1.0f;       // Fix w component which originally holds the noise
    frequency measure
```

```
    coordsProj = mul(coordsOrig, g_mMVP);
    coordsProj = coordsProj / coordsProj.w;
    // (2) Get velocity from 2D (image-space) field
    coordsProj.x = 0.5f * coordsProj.x + 0.5f;
    coordsProj.y = -0.5f * coordsProj.y + 0.5f;
    float3 velocity =  tex2D(FlowField2dSMP, (float2) coordsProj) - 0.5f;
    // (3) Compute new model-space position according to Euler integration
    float3 coordsNew = coordsOld - velocity * g_vStepSize.x * coordsOld.w;

    // Output new coords:
    Output.RGBA = float4 (coordsNew, coordsOld.w);    // Propagate the original coordsOld.w↙
     value

    return Output;
}
```