MotionBlur10 Sample

⊟ Collapse All

This sample implements motion blur using the geometry shader to extrude fins from the original mesh.

Path

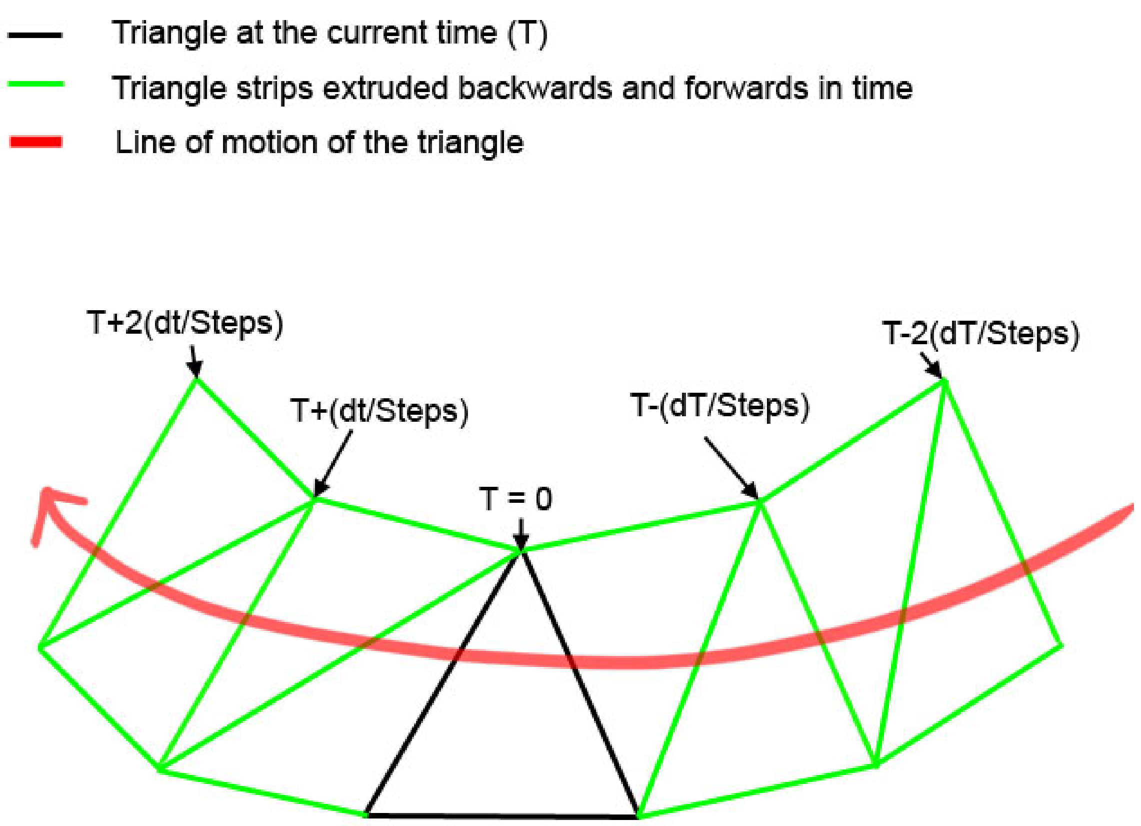
Source	*SDK root*\Samples\C++\Direct3D10\MotionBlur10
Executable	*SDK root*\Samples\C++\Direct3D10\Bin\x86 or x64\MotionBlur10.exe

Sample_Overview

The MotionBlur10 sample implements motion blur using a geometric approach combined with order independent transparency through the use of AlphaToCoverage and with anisotropic texture filtering.

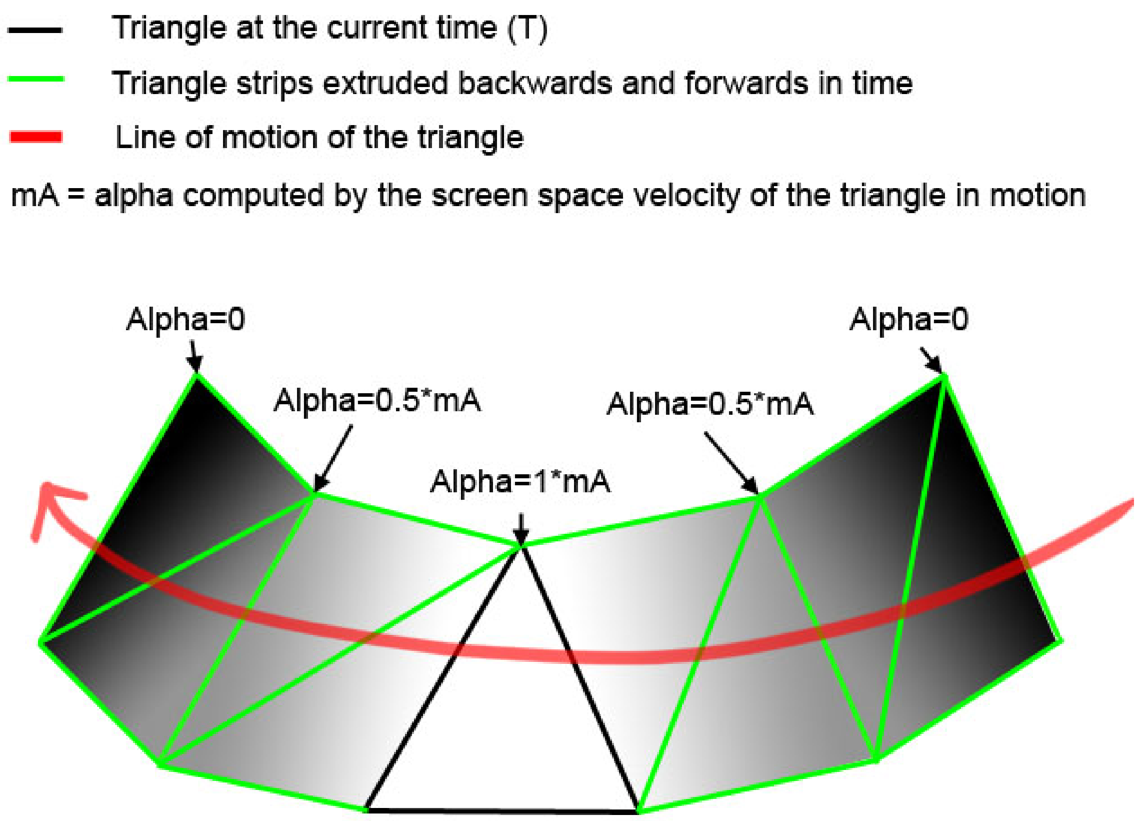
Extruding Geometry

Geometry is extruded along the path of motion. A triangle strip is extruded backwards and forwards in time such that the middle triangle is always at the current time. This is done by passing in an array of matrices each containing the transformation information for specific points in time. Namely, N equally spaced points of time ahead and N equally spaced points of time behind. The sample handles this in the application code by calculating the current time and then determining how far to go backwards in time based upon the length of time that the fictional "shutter" will stay open. The middle time in this array of matrices is where the object will look most "solid". This method allows the motion blur effect to be independent of frame rate. The only dependency is that the application must track or guess the locations of the objects at times in the past. Normally this is just a few fractions of a second into the past, but for dramatic effect, applications could blur over several seconds.


Triangle at the current time (T)
Triangle strips extruded backwards and forwards in time
Line of motion of the triangle
T+2(dt/Steps)
T+(dt/Steps)
T = 0
T-(dT/Steps)
T-2(dT/Steps)

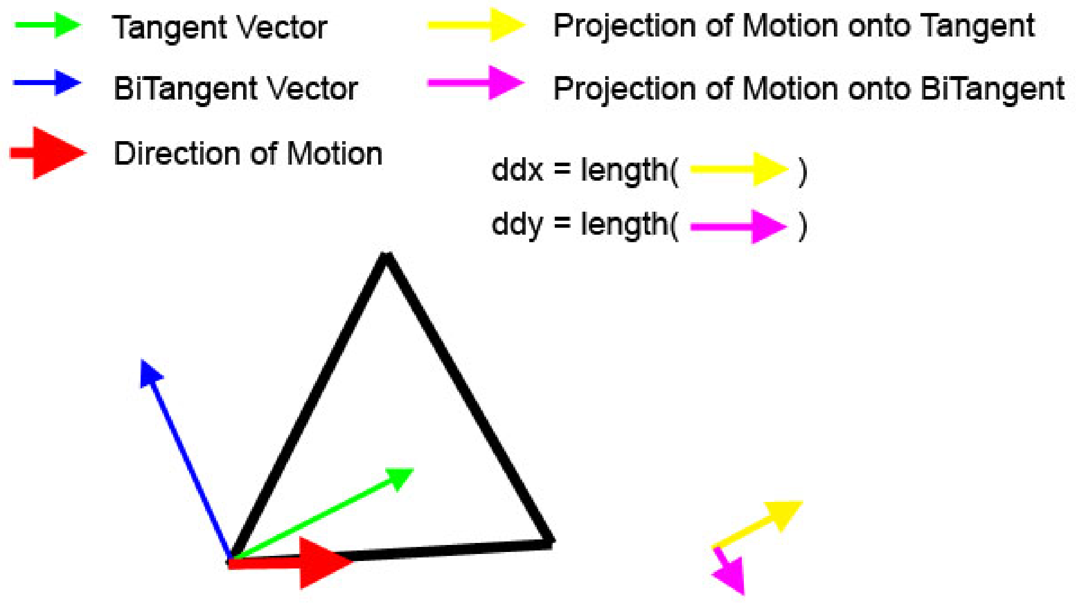
Alpha Blending

As each section of the triangle strip is extruded, it is given an alpha value based upon how much time has elapsed between the time represented by this section of the strip and the current time (the triangle in the middle of the forwards and backwards strip). Because this information is not transmitted back to the CPU, we do not have the option of using the CPU to sort the triangles for accurate alpha blending. Therefore, the sample uses AlphaToCoverage to fake order independent transparency. Please see the Instancing10 sample for a more detailed description of transparency using AlphaToCoverage.


Triangle at the current time (T)
Triangle strips extruded backwards and forwards in time
Line of motion of the triangle
mA = alpha computed by the screen space velocity of the triangle in motion
Alpha=0
Alpha=0
Alpha=0.5*mA
Alpha=0.5*mA
Alpha=1*mA

Texture Space Motion Blur

The final trick to achieving the motion blur illusion is to blur the texture in the direction of motion when sampling. This is achieved by enabling anisotropic filtering and carefully controlling the ddx and ddy parameters to SampleGrad. These can be found by transforming the vector indicating the direction of motion into the texture space for that particular triangle.



```
    // Find the projection of our motion into our tangent/texture space
    Output.Aniso.y = max( 0.0001, abs( g_fTextureSmear*dot( clipTangent, clipMotionDir ) ) );
    Output.Aniso.x = max( 0.0001, abs( g_fTextureSmear*dot( clipBiTangent, clipMotionDir ) ) );
```

Note that the g_fTextureSmear variable can be used to control how much the texture will stretch along the direction of motion. Setting this value too high can result in the anisotropic filter accessing part of the texture completely unrelated to the current texture coordinate.