Vertex Blending under DirectX 7 for the GeForce 256

Sim Dietrich
sim.dietrich@nvidia.com

What is Vertex Blending?

Traditional 3D animation operates on models which consist of a hierarchical series of rigid bodies. This approach to animation produces artifacts, including interpenetration of model sections and gaps between model segments. Vertex blending enables 'skinning' support that addresses these issues.

'Skinning' is a technique that allows vertices to straddle more than one level of the model hierarchy. The triangles that include these shared vertices become stretched between hierarchy levels, thus eliminating gaps and hiding interpenetration. This allows artists a greater level of control over animation. The GeForce 256 supports skinning through 2-matrix vertex blending.

There is a demo of this on our web site.

What's the Math behind Vertex Blending?

When using vertex blending under Direct3D 7, applications set up ***two*** model to world matrices. Rather than just D3DTRANSFORMSTATE_WORLD, there is also a D3DTRANSFORMSTATE_WORLD1.

A Vertex is transformed by *both* matrices, and its final position and normal are determined by a linear interpolation between the results. The weighting factor of the lerp is supplied on a per-vertex basis. The weighting factor is analogous to alpha in a standard alpha blend.

Final Position = VertexPosition * World (Weight) + VertexPosition * World1 (1 - Weight)
Final Normal = VertexNormal * World (Weight) + VertexNormal * World1 (1 - Weight)

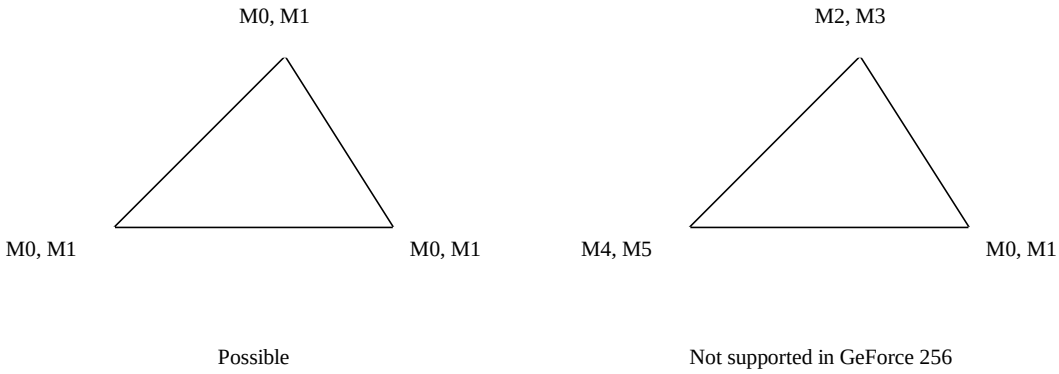
So, as the weight varies from 0 to 1, the position changes from the position given from the World matrix to that given by World1. The normal is lerped similarly. If you are using vertex blending, ensure that D3DRENDERSTATE_NORMALIZENORMALS is set to TRUE.

Vertex Blending is enabled on the GeForce256 by setting D3DRENDERSTATE_VERTEXBLEND to D3DVB_1WEIGHT . When in this state, the GeForce 256 will use both D3DTRANSFORMSTATE_WORLD and D3DTRANSFORMSTATE_WORLD1 in its transform calculations. You must also supply a floating point weight in your FVF format.

Vertex Blending is not available for vertices specified via D3DTLVERTEX.

Per-Triangle Vertex Blending

In Direct3D, blending matrices are applied on a per vertex basis and cannot be changed within a triangle, as shown on the left. The rightmost triangle would require the ability to switch which two matrices were active from one vertex to the next.



The GeForce 256 supports 2 blending matrices per vertex. These are not divisible within a triangle. In Direct3D, blending matrices are applied on a per vertex basis and cannot be changed on a per primitive basis – in other words, you cannot reassign the blending matrices within a Draw*Primitive* call. If you require more sophisticated vertex blending than this, you have several options

1. Use the two most important matrices and discard the others
2. Perform the skinning transform in software. You can still use the rest of the TnL pipeline, though. Pass in 'root model space', world, or view space geometry.

How to use Vertex Blending if starting from scratch

The highest performance use of vertex blending would leverage the GeForce 256 GPU to perform simple parent-child skinning with 2 matrices per triangle. The application would set the two appropriate matrices for each group of triangles that share them and send down the weights as part of the vertex's FVF format. Artists can specify a per-vertex weight, or could have the engine generates automatically based on proximity to each bone.

What to do if your artists can't live with just 2 bones

Certain types of animation are considerably easier for artists to create when allowed the greater flexibility of having more than 2 matrices affect each triangle. Many engines use up to 4 bones per vertex, which can vary within a triangle.

For maximum performance on these models, simply compute the skinning on the CPU and send the vertices down in world or view space. This leverages the TnL pipeline, as well as projection and clipping. Developers have reported a 2x increase in performance over performing all transform, lighting, clipping and projection on the CPU.