SDK White Paper

# Matrix Palette Skinning

An Example

WP-01407-001-v01
July 2004

# Matrix Palette Skinning Example

Bone based animation is a good way to add believable animation to your engine without the overhead of vertex animation (morph targets). This example implements a simple bone-based animation technique that can be used on any PS/VS1.1 class hardware. On more advanced hardware this technique can be extended to support more bones and more complicated lighting.

This paper assumes that the reader is familiar with the concepts of bone-based animation, and shader programming.

This sample was written to run under DirectX9.0b on hardware that supports PS/VS1.1.

Bryan Dudash
bdudash@nvidia.com

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050

## Discussion

Bone based animation is a simple concept in which vertices on a model are associated with “bones” that are animated in space. The vertices blend the delta transformations of a group of bones to obtain a final transformation matrix. Implementing the transformation of the vertices based on the bone matrices on hardware is called “hardware skinning”, or “matrix palette skinning”. In order to animate a model using bone animation, you need to have the bone positions per frame, and the bone weights per vertex. The bone matrices are stored in the vertex shader constant registers and accessed by a vertex shader which in turn transforms each model vertex to its correct position. The bone weights per vertex are passed down on a per vertex basis in the vertex stream.

| Data | Use | How it is passed down |
|---|---|---|
| Per Frame Bone Transforms | Up to 4 used per vertex to perturb it’s position, orientation, and/or scale | Vertex Shader Constants |
| Per vertex Bone indices | Used to identify an offset into the bone matrix list(matrix palette) for up to 4 bones per vertex | Embedded per vertex in the vertex stream |
| Per vertex Bone weights | Controls how much each of the up to 4 bones weight the transform of the vertex | Embedded per vertex in the vertex stream |

Table 1. Data required for hardware skinning

# Implementation Notes

This example implements hardware skinning via HLSL which simplifies implementation and makes it as easy to follow as pseudo-code. Please refer to the .FX file with the sample source code for the details on the implementation. Some implementation details to note are:

1. The normal is also transformed. However, you only use the primary bone as variations based on the remaining bones are usually undetectable. This makes the normal subtly incorrect, but usually this cannot be noticed. This is done for performance reasons.

2. This sample implements simple per vertex diffuse lighting by taking the dot product of the transformed normal and the light vector. The user may of course implement a user-defined lighting equation.

### Notice

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

### Trademarks

### Copyright

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com