Direct Evaluation of NURBS Curves and Surfaces on the GPU

Adarsh Krishnamurthy, Rahul Khardekar, Sara McMains
Computer Aided Design and Manufacturing Laboratory, University of California, Berkeley

Abstract

We present a new method to evaluate and display trimmed NURBS surfaces using the Graphics Processing Unit (GPU). Trimmed NURBS surfaces are currently tessellated into triangles before being sent to the graphics card for display since there is no native hardware support for NURBS. Our method uses a GPU fragment program to evaluate the surface point coordinates of the original NURBS patch directly, from the control points and knot vectors stored as textures in graphics memory. This evaluated surface is trimmed during display using a dynamically generated trim-texture calculated via alpha blending. We obtain rendering speeds at least one order of magnitude faster than evaluation using the CPU.

Algorithm

We use two operations for displaying trimmed NURBS surfaces, one for surface evaluation and the other for trimming. The surface evaluation algorithm is a multi-pass algorithm that uses a fragment program to evaluate the surface point coordinates directly, with no modifications to existing hard-ware. The trimming operation is based on the approach in [Guthe et al. 2005]. In our implementation, the trimming curves are evaluated and then the trim-texture is generated. Finally, while rendering the surface, the actual trimming of the surface is performed using another fragment program.

Curve Evaluation

- Find the knot span in which \( u \) lies, say \( u \in [u_i, u_{i+1}) \)
- Compute the non-zero basis functions \( N_{p,i}^{u_i}(u), \ldots, N_{p,i}^{u_{i+1}}(u) \)
- Multiply the non-zero basis functions with the corresponding control points and sum the results:

\[
N_p^{u}(u) = \begin{cases} 
\frac{u - u_i}{u_{i+1} - u_i} N_{p-1}^{u_i+1}(u) + \frac{u_{i+1} - u}{u_{i+2} - u_{i+1}} N_{p-1}^{u_{i+2}}(u), & 0 \leq u < u_{i+1} \\
1, & u_{i+1} \leq u < u_{i+2} \\
0, & \text{otherwise}
\end{cases}
\]

Basis Function Evaluation

- First-order basis function is the step function common for all evaluation points
- Second-order basis function is computed from the first using a fragment program and directly stored in another texture
- Alternate using two textures to calculate the higher order basis functions from the lower order basis functions
- \( k \)-th order basis function computed in \( k \)-1 passes

![Control points and Basis function values](image1)

![Output during intermediate passes while computing a cubic basis function](image2)

Surface Evaluation

- Locate the lower left corner of the sub-mesh of control points that influence the evaluation point coordinates
- Compute the non-zero basis functions along \( u \) and \( v \) directions
- Multiply the non-zero basis functions with their corresponding control points from the sub-mesh and sum the results:

![Basis Functions and Control Mesh](image3)

![Evaluation Mesh](image4)

Trimming

- A trim-texture is generated on the GPU for each surface
- During rendering, a fragment program discards surface's fragments (pixels) masked by the trim-texture

![Trimming in the parametric domain](image5)

![Trimmed surface](image6)

Results

Sample trimmed NURBS surface models rendered at different LODs:

- Bottom left: graph comparing evaluation time of a single NURBS patch with our GPU algorithm versus standard CPU evaluation.
- Bottom right: graph comparing frame rates for an animated teapot scene containing varying number of trimmed NURBS surfaces (ducks).
- In both, the GPU based evaluation is 30-40 times faster than on the CPU.

![Evaluated Points vs. Evaluation Time](image7)

![Frame Rates vs. Number of NURBS surfaces](image8)