GPU-based trimming and tessellation of NURBS and T-Spline surfaces

Michael Guthe Ákos Balázs Reinhard Klein*
Universität Bonn, Institute of Computer Science II

1 Introduction

Despite the fact that trimmed NURBS are the standard surface representation in CAD there is no hardware support for directly rendering either trimmed NURBS or T-Spline surfaces. In [Guthe et al. 2005] we proposed a fragment based trimming approach combined with a single pass tessellation on the GPU. To support a wide range of graphics cards, the surfaces are approximated with a rational bi-cubic Bézier patch hierarchy at runtime. For rendering, these hierarchies are traversed every frame, and since typical CAD models can have tens of thousands of surfaces this traversal has to be fast.

2 Implementation

Figure 1 shows the UML diagram for each surface. As the key to an efficient implementation is a compact and cache-friendly data structure for the approximating bi-cubic Bézier patch and cubic trimming curve hierarchies, we describe the novel classes (marked green) in more detail. The hierarchies are organized in binary trees and the root of each is the coarsest approximation of a given trimming loop or surface. To improve cache coherency during the traversal we keep each tree in a single memory block by using an `std::vector` as container, where the root is at index 0. For curves the structure of a node in the tree is shown in Figure 2 top left. The red star denotes a pointer, while the red *i* denotes an index in the vector and the flag distinguishes between leaf nodes (pointer to the original high degree curve) and inner nodes (index of first child). For bi-cubic patches see Figure 2 right. Here the flags determines if the node is a leaf node (pointer to the original high degree patch which it approximates) and the direction in which the patch is to be split if necessary. Figure 2 bottom left shows the additional data necessary for the (bi-)cubic approximation. It has pointers to its approximating binary trees (one for each trimming loop plus one for the bi-cubic patch hierarchy that approximates the surface itself) and to the original NURBS or T-Spline surface class (that provides the `convertToBezier()` method).

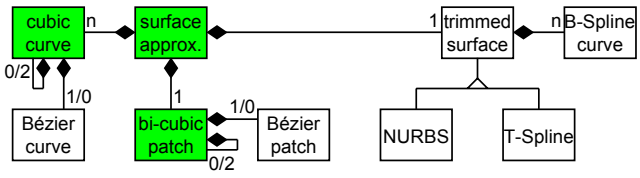

Figure 1: UML diagram for (bi-)cubic approximation.

*e-mail:{guthe,edhellon,rk}@cs.uni-bonn.de

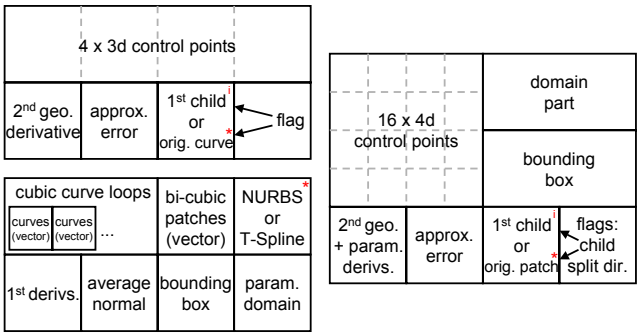

Figure 2: Top left: Node structure for a cubic approximating curve. Bottom left: Data structure for (bi-)cubic surface approximation. Right: Node structure for a bi-cubic approximating Bézier patch.

Since every parent's bounding box must contain its children, we speed up the traversal by storing the intersected planes of each inner node to accelerate view-frustum culling for the children. In order to select an accurate approximation, the maximum object space error must be calculated for each node from the prescribed screen-space error. Since the object space error of the parent node is already known (from the previous traversal step) and cannot be greater than that of the current node (because the bounding box of a node cannot be closer to the viewer than that of the parent), we first check the bi-cubic approximation error against the parent's object space error.

```
calculate weights & derivative weights

evaluate in u-direction (patch → curve)
calculate curve derivative (⇒ v-derivative)
evaluate curve (⇒ position)

evaluate in v-direction (patch → curve)
calculate curve derivative (⇒ u-derivative)

normal = u-derivative × v-derivative
```

Figure 3: Pseudo-code for the patch evaluation shader.

On the GPU, three different shaders are used to: evaluate cubic Bézier curves, evaluate bi-cubic Bézier patches and perform the fragment-based trimming. Since the shader for the patch evaluation is the most complex one, we give the pseudo-code in Figure 3. The full GLSL shader functions are provided in the supplementary material along with the specification of the proposed OpenGL extension.

References

GUTHE, M., BALÁZS, Á., AND KLEIN, R. 2005. GPU-based trimming and tessellation of NURBS and T-Spline surfaces. *ACM Transactions on Graphics 24*, 3 (Aug.), cond. accepted.