A Prototype for a Graphics Library Utility

To facilitate the design of drivers for the proposed architecture, we must develop a Graphics Library Utility (GLU). The primitives of the GLU are strips, fans, meshes and indexed meshes. Current rendering methods are based on triangle databases (strips, fans, meshes) resulting from offline tessellation via specialized tools. These tools tessellate the patch databases and ensure that there are no cracks between the resulting triangle databases. The tools use some form of zippering. The triangle databases are then streamed over the AGP bus into the GPU. There is no need for any coherency between the strips, fans, etc., since they are, by definition, coherent (there are no T-joints between them). The net result is that the GPU does not need any information about the entire database of triangles, which can be quite huge. Thus, the GPUs can process virtually infinite triangle databases. Referring to Figure 1 in a strip, the first patch will contribute sixteen vertices, and each successive patch will contribute only twelve vertices because four vertices are shared with the previous patch. Of the sixteen vertices of

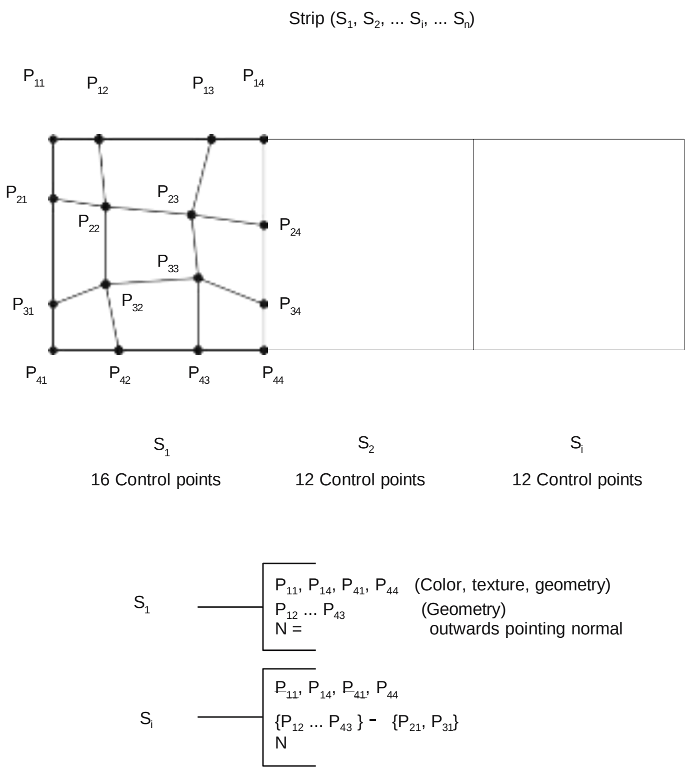
Strip (S1, S2, ... Si, ... Sn)
P11
P12
P13
P14
P21
P22
P23
P24
P31
P32
P33
P34
P41
P42
P43
P44
S1
16 Control points
S2
12 Control points
Si
12 Control points
S1
P11, P14, P41, P44 (Color, texture, geometry)
P12 ... P43 (Geometry)
N = outwards pointing normal
Si
~~P11~~, P14, ~~P41~~, P44
{P12 ... P43 } - {P21, P31}
N

Fig. 1. Strip

Mesh (S_{11}, S_{12}, ... S_{1N}, ... S_{21}, ... S_{2N}, ... S_{M1}, ... S_{MN})

S_{M1} 12 Control Points	S_{M2} 9		S_{Mi} 9		S_{MN} 9
S_{21} 12 Control Points	S_{22} 9 Control Points		S_{2i} 9		S_{2N} 9
S_{11} 16 Control Points	S_{12} 12 Control Points		S_{1i} 12		S_{1N} 12

Fig. 2. Mesh

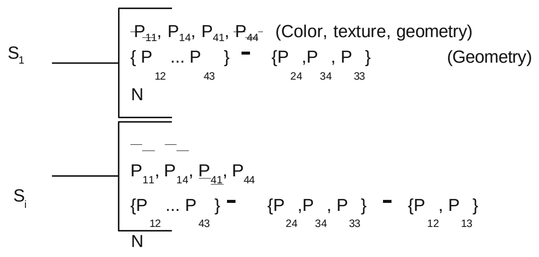
S_1
P_{11}, P_{14}, P_{41}, P_{44} (Color, texture, geometry)
$\{P_{12} \ldots P_{43}\} - \{P_{24}, P_{34}, P_{33}\}$ (Geometry)
N
S_i
P_{11}, P_{14}, P_{41}, P_{44}
$\{P_{12} \ldots P_{43}\} - \{P_{24}, P_{34}, P_{33}\} - \{P_{12}, P_{13}\}$
N

Fig. 3. Fan

the first patch, S_1, there will only be four vertices (namely, the corners P_{11}, P_{14}, P_{41}, P_{44}) that will have color and texture attributes; the remaining twelve vertices will have only geometry attributes. Of the twelve vertices of each successive patch, S_i, in the strip, there will only be two vertices (namely, P_{14} and P_{44}) that will have color and texture attributes. It is this reduction in the number of vertices that will have color and texture attributes that accounts for the reduction of the memory footprint and for the reduction of the bus bandwidth necessary for transmitting the primitive from the CPU to the rendering engine (GPU) over the AGP bus. Further compression is achieved because a patch is expanded into potentially many triangles by the Tessellator Unit inside the GPU. Referring to Figure 2, in a mesh, the anchor patch, S_{11} has sixteen vertices, all the patches in the horizontal and vertical strips attached to S_{11} have twelve vertices and all the other patches have nine vertices. Each patch has an outward pointing normal. Referring to Figure 3, each patch has only three boundary curves, the fourth boundary having collapsed to the center of the fan. The first patch in the fan enumeration has eleven vertices; each successive patch has eight vertices. The vertex P_{11}, which is listed first in the fan definition, is the center of the fan and has color and texture attributes in addition to geometric attributes. The first patch, S_1, has two vertices with color and texture attributes, namely P_{41} and P_{14}; the remaining nine vertices have only geometric attributes. Each successive patch, Si, has only one vertex with all the attributes.

The meshed curved patch data structures introduced above are designed to replace the triangle data structures used in the conventional architectures. Within one patch strip, the edge database must be retained for zippering reasons but no information needs to be stored between two abutting patches. If two patches bounding two separate surfaces share an edge curve, they share the same control points and they will share the same tessellation. By doing so we ensure the absence of cracks between patches that belong to data structures that have been dispatched independently and thus our method scales the exactly the same way the traditional triangle based method does.