I.3.2. RENDERER PERFORMANCE

I.3.2.1—General Considerations

The Pixel-planes 5 architecture contains a number of "virtual buffers", 128 x 128 pixel arrays of processor-enhanced memories, which are called Renderers. By choosing the Renderer size to be large enough so that most polygons in complex scenes will fit within a given 128 x 128 pixel region of the screen, one Renderer can provide performance almost as high as a full processor per pixel frame buffer, like Pixel-planes 4, but at a fraction of the hardware cost. The full system contains a number of these Renderers, operating on separate streams of polygon input data. The screen is divided into a number of 128 x 128 regions, and the Renderers are dynamically allocated to these regions in such a way that they process nearly equal numbers of polygons, thereby balancing the load.

The performance for rendering a given primitive is given by

$$\text{update time} = N * P_w * T$$

where N is the number of clock cycles required for a Renderer to process one primitive, P_w is the number of primitives which must be processed by the Renderer with the heaviest load (the "worst-case" Renderer), and T is the Renderer clock period, nominally 25 ns. This assumes that Renderers never have to wait for input primitives.

P_w is very complicated to analyze. Its theoretical limits are given by

$$P/B <= P_w <= P * R/B$$

where P is the number of primitives in the scene, R is the number of regions in the screen (or window), and B is the number of Renderers in the system. In the upper limit, all the primitives intersect all the regions, so every Renderer must process every primitive once for each of R/B regions. In the lower limit, each primitive falls within only one region and perfect load balancing is achieved, so that each Renderer processes the same number of primitives. Note that these upper and lower limits differ by a factor of R, which is 80 for 128x128 regions on a 1280 x 1024 screen.

For long (L > S) non-Manhattan vectors, the bounding-box approach will result in many regions unnecessarily processing the vector. For longer vectors, if we assume that we can be more clever than using bounding boxes, so that a vector only passes through at most two regions on any given column of regions, then we get

$$Hv = [1 + (L \cos\phi / S)] * [1 + \tan\phi]$$

where ϕ is the vector angle chosen so that $\phi < 45^{o}$. However, this formula is pessimistic for short vectors, where $(L\sin\phi/S) < \tan\phi$. Note that both formulas give Hv = 1 + (L / S) for horizontal and vertical vectors.

If we assume 50 pixel long vectors oriented at 45° and 128 x 128 regions, then Hv = 1.63. Still assuming perfect load balancing, this gives P_W = 1.63 V / B, where V is the number of vectors in the scene. Assuming 16 Renderers with 25 ns cycle time, we get update time = V * 239 ns.