Collision Computation
- When all VPs exhaust their lists, they reconverge and are assigned the next set list until it reaches the end of its list.
- Each VP runs computes the resulting impulse from an element of the collision list.

Vertex Shader
- Striplime across all triangles in scene.
- Perform per vertex lighting.
- Transform each triangle to screen coordinates, and assign a face color (flat shading).
- Maintain y_begin, the lowest y-coordinate a triangle appears in the scene.

Sort Triangles by y_begin
- Sort-middle rendering approach.
- Use a standard vectorized radix sort with y_begins as the keys.

Divide Frame into Bins
- With sorted triangles, easy to split frame into pieces and place triangles into “bins.”
- Each core paints an independent section of the frame.

Scan Convert Triangles
- Each core striplimes across all triangles in its bin.
- Each VP determines triangle orientation and initializes state.
- For each scanline a triangle intersects, a “fragment” is created with the triangle metadata: beginning x-coordinate, length of fragment, color value, etc.
- Each fragment is placed in the proper Scanline “Bucket” which hold all the triangle fragments which intersect the scanline.

Metadata Buckets
- Each core now striplimes across all Scanline Buckets.
- Each VP paints one triangle fragment (per vector fetch) from its bucket, and then reconverges to paint the next.
- Z-buffering done at this step.

Early Results

Challenges
- Finding ways to break up application kernels to force reconvergence among Virtual Processors.
- Trying to write divergent kernels which can exploit the vector memory system as opposed to using explicit VP loads and stores, but still allow for fragments to branch separate ways.
- Finding ways to always exploit the two dimensions of parallelism: one along different Control Processors, and the other along the respective Virtual Processors.