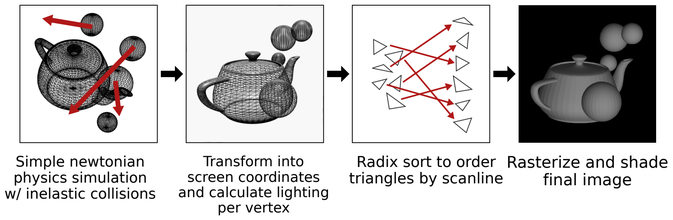
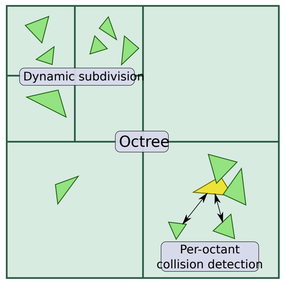
Mapping Computer Graphics Applications to a Maven Vector-Thread Core

Alex Bishara, Richard Xia

Overview

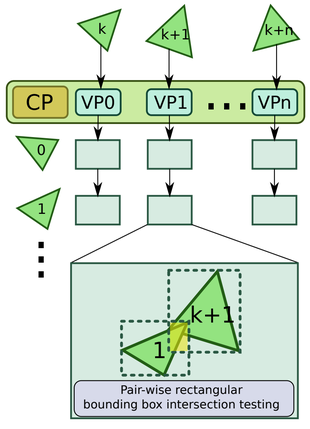


Physics Engine



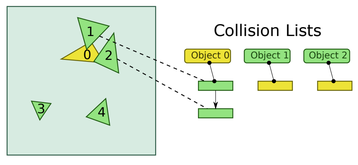
Octree Body Division

- Allows for parallel collision detection by assigning each CP an octant
- Dynamically subdivides an octant when the number of bodies exceeds a certain threshold



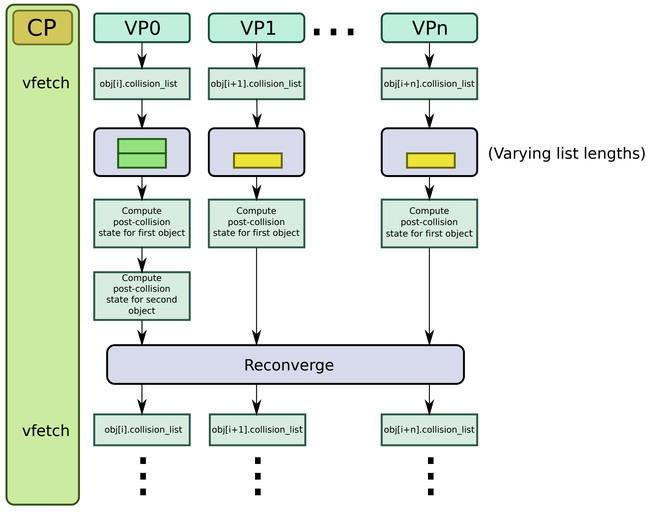
Collision Detection Phase

- Use rectangular bounding boxes to determine collisions
- Stripmine over each object and have each VP compare its own object to a single, fixed one
- Generates a list of colliding bodies for each object



Collision Computation

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



Time Step

- Stripmine over each object and add the impulses computed in the previous step to the object's current state
- Completely data parallel, contains no branches

Graphics Pipeline

Vertex Shader

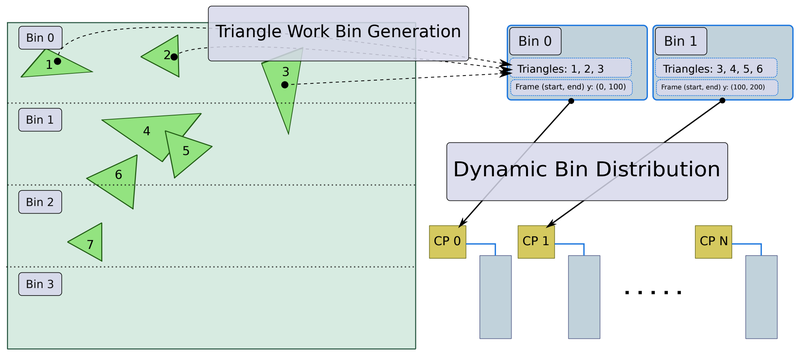
- Stripmine across all triangles in scene
- Perform per vertex lighting
- Transform each triangle to screen coordinates, and assign a face color (flat shading)
- Maintain y_begin, the lowest y-coordinate a triangle appears in the scene

Sort Triangles by y_begin

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

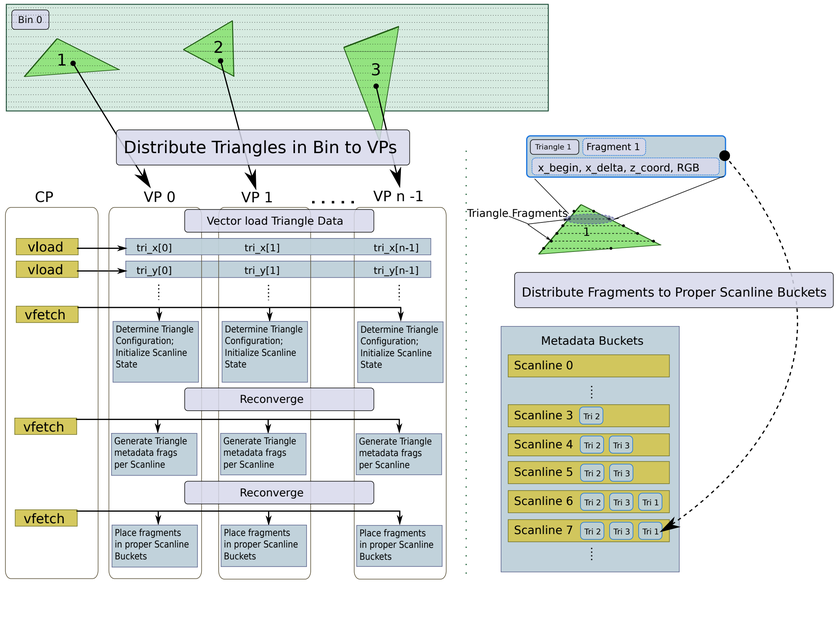
Divide Frame into Bins

- With sorted triangles, easy to split frame into pieces and place triangles into "bins"
- Each core paints an independent section of the frame



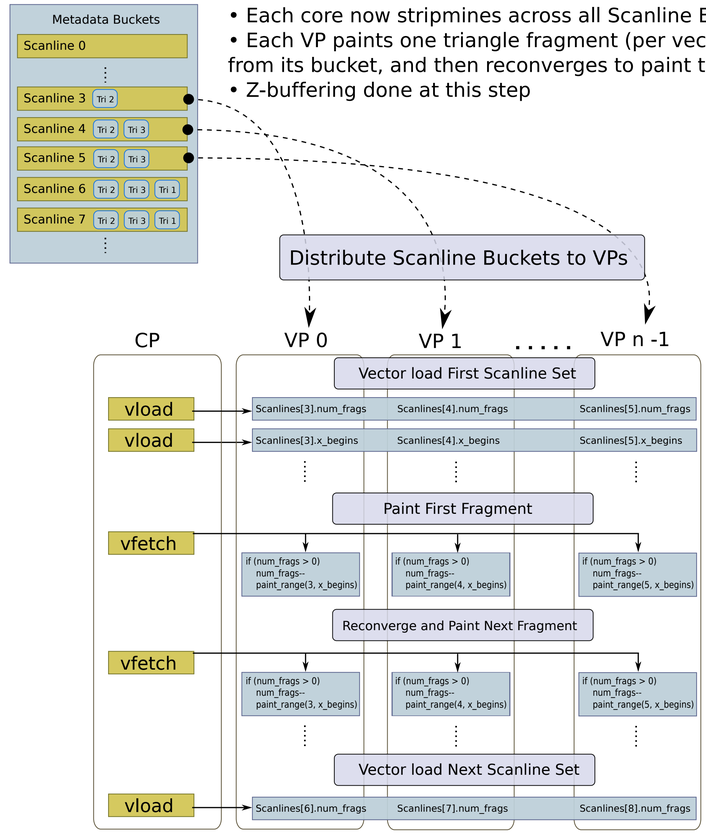
Scan Convert Triangles

- Each core stripmines 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

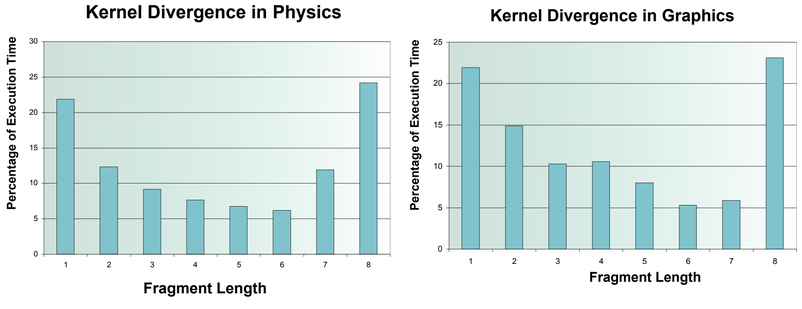


Scanline Rasterizer

- Each core now stripmines 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