Measurement and performance analysis of graphics hardware

Sugeerth Murugesan

1 February 2014

Graphics operations mainly consists of two components:, geometry and rasterization. The geometry/vertex operations are mainly characterized by lighting, transformation, and vertex operations, while fragment operations consists of rasterization, textring and so forth. The overall graphics performance is a combined effort of the fragment and the vertex operations. Provided that one of the stage is slow , it would affect the overall rendering process of the graphics pipeline.

This document contains our results for the investigation of the crossover point between the geometry and the rasterization stages of the pipeline.We use WesBench as our benchmark program to make the characterization under different conditions. The different impts

1 Crossover between the Geometry and Rasterization/Fill operations

As we discussed, the performance of the overall graphics rendering is determined by the interdependent stages- fragment and vertex. Our tests measures the performance of stages relative to the other. Theoretically, as the fragment operations become larger, the operations done by the vertex stage drop as they are dominated by the rasterization/fill operations.

The tests determine a point at which there is a crossover between the vertex operations and the fill rate by varying the triangle size. The crossover point between the geometry stage and the fill operations. We analyze the performance of the geometry rate as number of vertices processed per second and the fill rate as the number of fragments per second. The platform that we run our tests are:

Graphics Card: NVIDIA GeForce GT 420M/PCIe using 304.88 NVIDIA version.

Processor: Intel Core i5-460M

Version: 4.2.0 NVIDIA

Screen: 1366x768

To answer the question regarding the crossover point between the geometry stage and the rasterization we hold the lighting, texture and the buffer size as

constants and vary the triangle area (pixels) to observe the patterns in performance. Basically, we disable all the non-rasterization fragment operations. We perform the test with varying triangle area sizes are: 1,2,4,8,16,32,64,128,256,512,1024,2048, and 4096 pixels. The results suggest that the crossover between the geometry

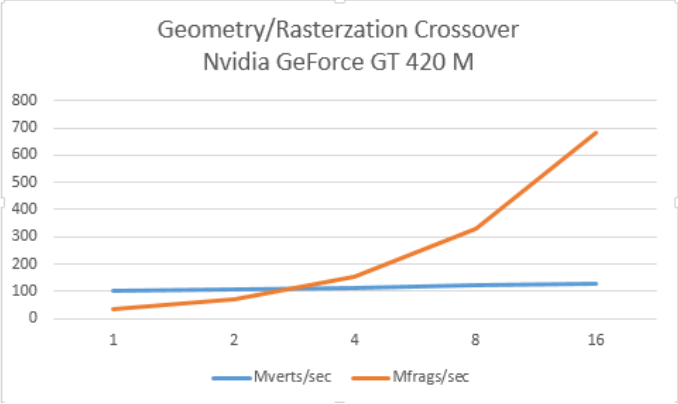

Figure 1: X-axis:Triangle Area,Y-axis:Performance, comparision of the geometry adn rasterization performance as a function of triangle area size

and the rasterization occurs on the GPU as the triangle area is raised from 2 pixel to 4 pixels. This range corresponds where the fill rate increases and crosses over the geometry rate. The decrease in the geometry rate is more profound between the 16 pixels to 32 pixels. We provide a graph which states the comparison of the geometry and rasterization perfomance as a function of triangle area size. The geometry rate is high for the small-area triangles then falls off at roughly 16 pixels.

This behaviour seen as the fragment operations become more dominant in the rendering process. The cost to draw a triangle is dominated by fragment operations. The vertex rate drops as the rasterization stage becomes overwhelmed by the incoming geometry portion. In other words the capacity of the rasterization process cannot hold the workload from the vertex stage.

2 Geometry vs Rasterization with various parameters

In this section we explore the various aspects of tests where we tweak the parameters such as the lighting, texturing, and the type of triangles sent to the rasterizer. We use OpenGLs default lighting environment

2.1 Lighting Enabed

In the figure 2, we send disjoint triangles of varying size from the vertex stage to the rasterizer. The following graph plots between the lit and the unlit version of the two stages.

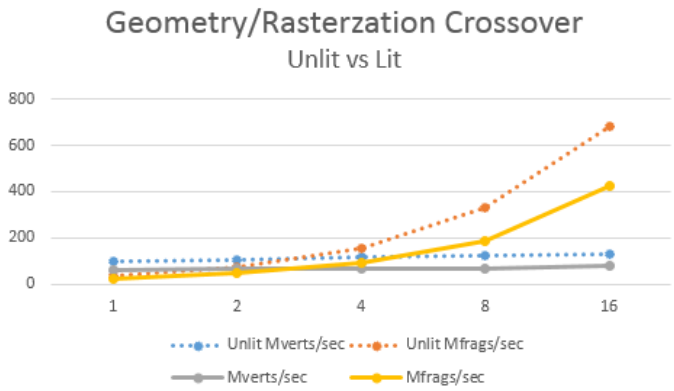

Figure 2: X-axis:Triangle area,Y-axis:performance, The extra work that is calculated for the lit triangles result in a decreased performance than the unlit version of the triangles

We find that when lighting is enabled the vertex rate is less than the rate when lighting is disabled. This is because there is more work needs to be done per vertex to give to the rasterizer stage. Again, as the vertex rate is lower than for the unlit case, the crossover point moves towards 4 pixel triangle area from 2pixel . Also, in the lit case we are sending more data down the rasterizer stage, hence, the decrease in performance in the vertex rate.

2.2 Texture Enabled

The texture is computed by WesBench using checkerboard pattern. In the figure 3, the results show the comparision between textured and untextured triangles. We find that the relative vertex rate of the untextured triangles are greater than the textured triangles. This is because the textured triangles need mapping from the texture matrix to the plain triangles, these require computations from both the vertex and the fragment stage of the graphics pipeline.Also, for this reason the crossover point of the untextured triangles are about the same for the textured triangles. The calculations are intense, the performance of the fragment rate also drops.

2.3 Texture Enabled and Lighting Enabled

As we saw in the sections earlier that the lighting operation takes a toll on the vertex stage and the texture operation takes a toll on the rasterizer stage,

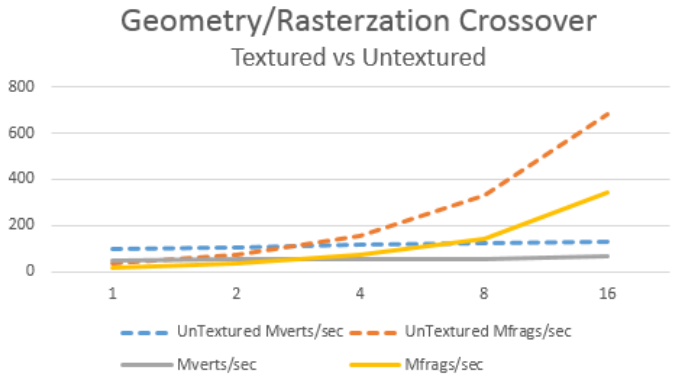

Figure 3: The crossover points of the rasterizer and the geometry stage are fond to be somwhat similiar. The performance rate of the vertex has dropped.

doing both simultaneously imposes a work on both the stages. Although, the work done in the vertex stage outweighs the rasterizer stage. There is a greater amount of work per vertex in the lit and textured case than in the unlit case. The more amount of work in the vertex stage is evident from the "moved" crossover point between the geometry and rasterization stage. The crossover stage has moved towards 4 triangle area pixel in the lit and textured case. The "moved" crossover stage indicates that the rasterizaer stage is waiting until 4 triangle area size to receive the geometry from the vertex stage.

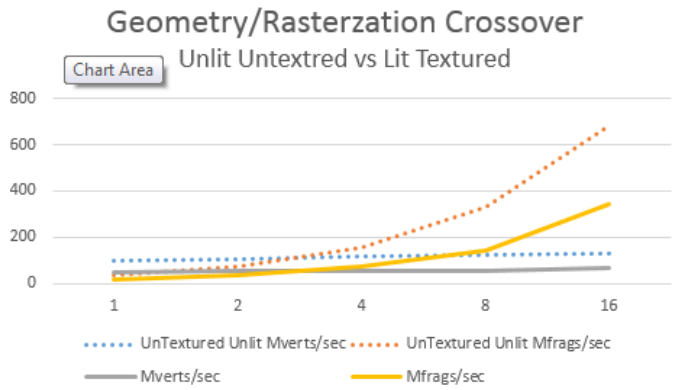

Figure 4: The crossover points of the rasterizer and the geometry stage have moved towards 4 triangle size. The performance drop in the lit and textred triangle is profound than the unlit and untextured version.

2.4 Disjoint triangles vs triangle strips

The relative crossover points for disjoint and triangle strips. The disjoint triangles indicate that the vertex stage sends 3 different vertices to the rasterization stage, while the triangle strips indicate that strips are shared between triangles. In the disjoint case, more amount of work is done on the vertex stage hence the rasterization stage has to wait unitll 16 to 32 pixel area in the dataset. As a result, the crossover happens only in the 16 to 32 pixel area stage. While in the strips, the vertex stage has less amount of vertices to send to the rasterization stage. Also, there is less amount of work being done on the vertex stage. This results in an early crossover of the geometry and the rasteization stage.

3 Batch-size

At small batch-sizes when it is equal to the total geometry sent in one call, the CPU is interface-limited. We mainaly used vertex rates as the total geometry sent to the GPU. If the vertex calls are limited, the relative occupancy are also limited.Sending too few vertices hurts the performance as the call that goes to the GPU.When the buffer is equal to the total number of vertices sent then there would not be an exess or a defiency of occupancy. By varying the buffer limit, the right triangle size can be obtained with a good curve.

4 Texture tests

The size of the texture has impact on the performance. The test was run over disjoint triangles with an area size of 2 pixels. In the Figure we see that the performance is unaffected by the change in textre size untill a certain limit. As a result, an optimal texture occurs when the texel is below have less than or equal to 512 x 512 texels. Beyond the size, the frame rate decreases as there is less texture that can be fit into the cache. we also conclude that the cache size is 512 KB.

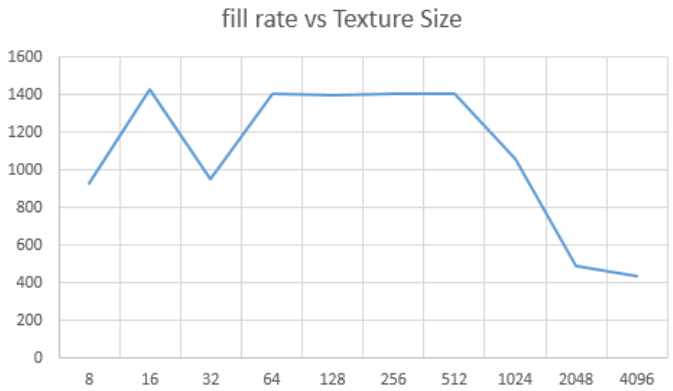

Figure 5: The performance is unaffected unitll 512 KB. After 512 KB, performance degrades.