The Graphics Pipeline:

Dalton Nell

CS 2810

December 4, 2013

Abstract

The graphics pipeline today is focused on being able to render as many polygons as possible as fast as possible. This makes graphic cards quickly obsolete as the amount of processing power of GPU's increase by about a half a gigaflop every year. Game developers have to be aware of the differences in graphic card capabilities so typically their games have polygon and texture budgets that keep them optimized enough to run on a range of hardware. However there is a rendering technique that renders the current graphics pipeline useless. This rendering method uses point cloud data rather than individual polygons. Rendering geometry by traversing the pointcloud in memory does not require any processing power, even if the pointcloud quality or number is increased. Rendering geometry with pointclouds would turn the effeciency of graphics rendering from $O(n^2)$ to a simple O(1) since increasing points in the point cloud would not require any more processing power.

The Graphics Pipeline

Ever since 3D graphics got popular 1990's, graphics hardware have been designed with one goal in mind: to render as many polygons as possible as fast as possible. This leads to graphics processing units not only quickly becoming obsolete, but they are also incredibly expensive. Despite these quick improvements they are still limited. Whenever games are designed they are often given “polygon budgets”. Game worlds and objects are all created out of polygons, usually triangles. Building spheres or trees out of polygons often leave them looking unusually sharp and blocky. To solve this in today's modern pipeline is to simply use more polygons, but due to the polygon budget the game developers can't solve all the sharp and blocky objects in this fasion. This not only limits the artist creating the game objects since they can't go over the polygon budget, but the developers as they need to make the game more efficient with level of detail model switching, binary space partitioning trees, and other complicated algorithms to draw less polygons to the screen. Now polygons aren't the only rendering method that exists. There is a rendering method with speed that isn't affected by the number of objects on screen. Instead of using polygons this rendering method uses volumes also known as voxels or atoms. With use of complicated raytracing techniques you can render an incredible amount of objects realtime with relative ease on normal hardware. This is because instead of looping through every individual object and drawing it to the screen, instead you're looping through uncolored pixels and grabbing what color they should be from the volumetric data. From use of octtrees, quadtrees, and even grids you can get built-in level of detail, frustum culling, and binary space

partitioning. (Kawamura, Sakamoto, & Koyamada, 2010) Although volume based rendering shows incredible potential, it won't be feasible compared to polygonal rendering for years to come.

Speed, it's all that matters in game rendering. Polygonal rendering already has this under its belt. With rendering techniques like imitations, BSP, frustum culling, and many others, with the help of hardware accelleration, it can represent huge game worlds without much effort on huge 1080p resolutions. However volume based rendering isn't as old and certainly isn't as feasible for game development for reasons that will soon be explained, and as such it doesn't have as many rendering techniques or clever algorithms. Volume based rendering currently is really slow. Despite various rendering techniques like empty volume skipping, and early ray termination-- most developers downscale the resolution or don't shoot as many rays in order to get enough frames per second to get a smooth display. Despite this slowness, it doesn't get any slower even if you add more objects. This unique feature of volume based rendering could be exploited to have game worlds with almost limitless detail even down to literally every grain of dirt without affecting the speed of the game. Unfortunately for now, polygonal rendering speed wise is still far more feasible than volume based rendering. Great rendering techniques for volume based rendering are out there (Krüger, Westermann, 2003), but until hardware accelleration exists for it: polygonal rendering for game development wins out on speed.

Detail, most people won't buy a sequal of a game if it doesn't have better “graphics” or detail. Polygonal rendering is pretty flushed in this regard. It has access to normal maps, skeletal animations, specular maps, shadows, and programmable shaders that all combine together to

make realistic models and effects quite easily. Despite polygonal rendering having a limited number of polygons and textures to use, combining all the effects onto a high-polygon, HD-textured, main character can lead to a very realistic and beautiful scene. You can't say this for all the scenes in a polygonal game though because of the polygon budget. You'll always find a low-polygon scene in a game somewhere. Volume based rendering would be quite the opposite. Everything could be as detailed as the developers want, but there's some downsides to the detail of volume rendering. Animation for volume based rendering simply doesn't exist, as least its not feasible to exist currently. For skeletal animation, the developer would have to computate the position of each individual ATOM of a character every frame based on bone weights. Which is a tremendous task for any hardware. Frame by frame animation isn't so out of this world for volume based rendering, but it wouldn't be very memory efficient nor would it be very pretty since it wouldn't match the frame rate. Volume rendering does win out in one aspect though. When rendering actual volumetric objects like smoke or clouds, volume based rendering completely outperforms polygonal rendering (Ikits, Kniss, Lefohn, 2008). Mainly because it is difficult to represent clouds or smoke with polygons. Despite this, polygonal rendering still wins out because it is far easier to get fascinating effects like real-time shadows, reflections, phong, transparency, and others while keeping everything visually consistent and running fast.

Ease of use, nobody will develop with it if it isn't easy to implement. Polygonal rendering has been a standard since the beginning of video games. It has tons of devkits, frameworks, and tools in order to create polygonal worlds with animated objects. Most documented resources are all about how polygonal rendering could be done better, faster, and more beautiful. All game

industries use it and release engine licenses to share the technology. Hardware implemented in modern computers are designed to accellerate polygonal based rendering. Polygonal rendering is by far the easiest to not only develop with, but its also the far easiest to get help with. Volume based rendering has a few articles floating around, but there's no true resources to create a fully fledged game based on volumes with animation, shadows, reflections, etc. This makes polygonal rendering again far more feasible.

Although volume based rendering shows incredible potential, it won't be feasible compared to polygonal rendering for years to come. Volume rendering is already used today for MRI and CAT scanning machines, but its unlikely to see any kind of volume rendering for video games in the forseeable future. Hopefully developers will develop hardware that can handle it with animation, shading, refraction, reflection, and material shaders as it will drasticallly improve video games artistically. Unfortunately with todays hardware, polygonal rendering performs better currently in speed, detail, and ease of use. In the future volume based rendering could be standard.

References

Kawamura, T., Sakamoto, N., & Koyamada, K., (2010). Level-of-Detail Rendering of Large-Scale Irregular Volume Datasets Using Particles. *Journal of Computer Science and Technology*. 25 (5), pp.905-915

Krüger, J., Westermann, R., (2003). Acceleration Techniques for GPU-based Volume Rendering. *IEEE Visualization 2003*. 14 (), pp.38

Ikits, M., Kniss, J., Lefohn, A., (2008). *Volume Rendering Techniques*. [ONLINE] Available at: http://http.developer.nvidia.com/GPUGems/gpugems_ch39.html. [Last Accessed 2 December 2013].