Comparch Final on GPUs

Abstract:
In this project our goal was to learn about the circumstances around and at what point exactly the need arose for a separate computational architecture to handle intensive graphics processing. We planned to study the architectural similarities and different design constraints of a GPU and CPU, and make sense of common GPU technologies and jargon not easily understood by the layman.

We decided to take on this project because we were inspired by the learning structure of CompArch this semester and wanted to create a spinoff miniseries that focused on the architecture of GPUs instead of CPUs. We intended to make this presentation lesson be a continuation of sorts for the last few lessons in CompArch. Because we left off on pipelining and MIPS architecture in class, we saw it as fitting to pick up from this point and look at how parallel processing takes the lessons from pipelining to further optimize processing speeds. From here we decided to then look at the historic branching off of GPUs as these pieces of hardware specced further into the processing horsepower skilltree than the CPU.One could look at the links in our sources section to find the resources that helped us develop our presentation.

There is much that we included from our sources but at the same time we only scratched the surface on what there is to know about GPUs. Platforms like OpenGL, WebGL, DirectX, Vulcan, etc. were not even discussed and looking into that would be especially interesting. Additionally we all would be open to offering additional insight to those wanting to build upon our two day presentation.

Documentation:
Presentation: 2 days worth of content, 2 activities.

- Day 1
    - *Slides* - discuss CPU purpose, basics of parallel compute, graphics pipeline, basic GPU hardware architecture, and connections.
    - *Activity* - have class break up into teams of 3-4 for 25 minutes and implement custom hardware for particular stages of the graphics pipeline.
- Day 2
    - *Slides* - discuss GPU capabilities for non graphics computing, expand on the GPU parallel compute architecture (CUDA in particular), relate to CUDA and OpenGL and finish with some performance examples
    - *Activity* - It's 2010 and you need to come up with arguments as to why OpenCL or CUDA is superior for the current technical demands of the time. You will not know which side you will be representing so come up with arguments and counter arguments for each side.

Learnings

Adi - I learned a lot about the motivation behind GPU architecture, parallel computing, and the basics of the graphics pipeline. It's really neat to understand how hardware used to fit a 'set in stone' graphics pipeline 20-30 years ago, but today, graphics and rendering pipelines are super customizable and sit on top of essentially programmable hardware that is optimized for parallel computation. I learned a lot about the graphics pipeline and got a nice taster/ intro to how 3D models / scenes are rendered.

Tolu - I have always been aware of the phrase "necessity is the mother of innovation", but in learning about the evolution of GPUs and some other processing platforms I came away with a newfound level of regard for how true this quote is. I really liked the way that the lessons in CompArch were structured this semester and wanted to use that framework to engage in an exercise of my own understanding: teaching a lesson on a topic that I researched. With the skills that I have now acquired from this project I am now much more comfortable in my ability to teach somebody something cool about COmputer Architecture and even show at Expo.

Patrick - I learnt a lot of the complex GPU pipeline basics. It was interesting to see from a graphics rendering perspective the number of shaders and other stages required to render a final output, and how to process evolved from a specific hardware for each stage, to a unified programmable hardware. It also strengthened my understanding of parallel core computing, by working with threads across multiple cores, to calculate necessary parallel calculations for graphics rendering, since the GPU would be dealing with millions of space coordinates each second, and would have to apply a wide variety of transformations in very little time I came away with a very solid foundation of the GPU workflow as a result, adding new words to my dictionary