Summary

I attempted to produce a quick way to add raytraced images in parallel. Using the Arnold renderer, one can take the light samples at each pixel and combine them together in order to quickly produce new rendered images.

Background

Raytracing is a common rendering technique for many graphics applications. Since it can be difficult to see how different lighting conditions can change a scene, many renderings have to be made in order to judge which lighting conditions best convey the artists' intent. Even the best raytracers can be slow if scenes are complex, so constantly rendering scenes can be time consuming. Fortunately, there is an easier way to create rendered images. Pixel values are calculated by aggregating light intensities through each pixel. Because of the additive nature of this, we can simply add images together to get new lighting combinations. So, if we have two images with two different lighting combinations, adding these images together will be the same as rendering an image with the combined lighting arrangement.

Of course, adding up pixels is easily parallelizable. All pixels should take approximately the same amount of work so work scheduling shouldn’t be a problem. Also, because we only have to visit data once, caching presents minimal challenges. Any prefetching should be able to get the picture’s array values into the cache for easy access.
The tricky part is learning the intricacies of Arnold and how to use its API to get the required values. Since we can’t naively use the typical 8 bit standard for pixels, we need to get the raw light sample data before we clamp it into the ordinary format.

Approach

First, I would’ve figured out how to program an output driver to write out all the sample values for an image. This would’ve allowed me to generate many sample images to use in testing. Next, I would have to write a program to parallelize over all these values and add up the samples. I would also have to use Arnold’s API to clamp the values to 8 bits and use a different output driver to store that result into a more common file format.

To continue from here, I would’ve been able to write some extra commands for my program which could allow for custom rendering parameters. This could involve specifying certain scenes and light source combinations to render and combine.

Results

Ordinarily, I would have some output images of my results to compare the accuracy of my added images with images rendered with all the same light sources. Then I would’ve had some graphs to show how much faster it is to just add images instead of rendering so many. Unfortunately, I was not able to complete my project. I was far too busy this semester and just wasn't able to handle the workload I took on overall.