Program 4

Hardware Ray Tracing

Due: 11:59:59 PM, March 24, 2014

Update your samples directory for an example solution path tracer (`samples/src/pathtracer`). Feel free to use this as your own solution, or just use it as a reference for adding to your own ray tracer.

This assignment is mostly about analysis. In this assignment you will implement multi-sampling and path tracing, or simply use the provided example to do your analysis.

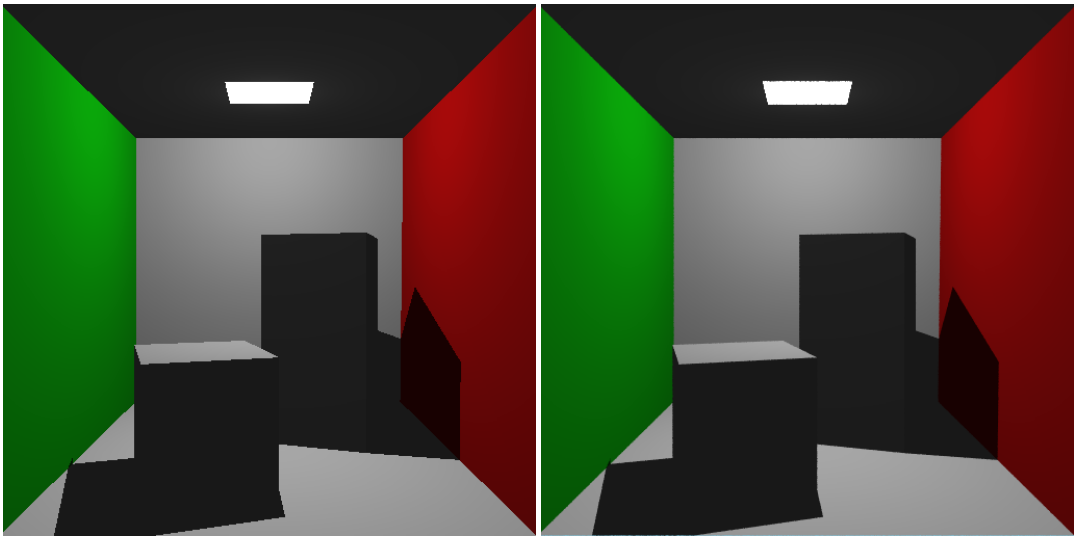
See Lecture 15 slides for more information.

1. **Part 1** (10 points): Multi-sampling

   Implement pixel multi-sampling with a simple box filter for anti-aliasing. This means, for every pixel, instead of casting a ray through the center of the pixel, cast it through a random point in the area of the pixel, and do this `N` times for `N` number of samples per pixel. A box filter is simply an average. Every sample contributes energy with equal weight to the final color of the pixel (divide total color by `num_samples`). This will require random numbers. To get a random number between 0 and 1, use the intrinsic `trax_rand()` with no arguments, as in:

   ```
   float offset = trax_rand();
   ```

   To verify your sampling is working correctly, render the Cornell scene with 20 samples per pixel, and compare it to 1 sample per pixel. Your results should look something like the following:

   On the left is 1 sample per pixel, on the right is 20 samples per pixel. There is a noticeable difference in the edges of the images, specifically on the shadow of the large box. Note that at 512x512, this will take a minute to render at 20 samples per pixel, even in run_rt.

   The TRaX memory loader supports an option for specifying the number of samples, if you don't want to hard-code it. Supply the argument `--num-samples N`, which will set the value located in memory at: `loadi(TRAX_NUM_SAMPLES)` to `N`.

2. **Part 2** (10 points): Path tracer

   Implement Kajiya style path tracing for global illumination in your ray tracer. See Lecture 15, or the provided path tracer sample for more information.

   For this assignment, remember to turn off the ambient term! The only light added to your image should be directly from the light source, and from the indirect global illumination rays.

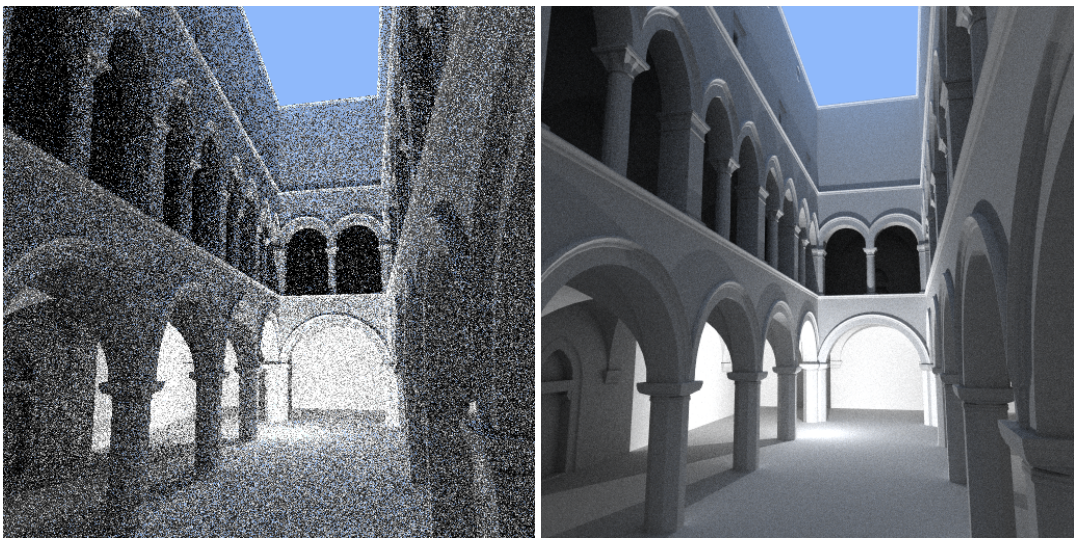
   The `samples/scenes` directory has been updated with the Sponza Atrium model.

   Note that there is no material file for sponza.obj. TRaX will automatically load a default material and assign it to every triangle, so you shouldn't need any special code to handle this. The background color is [0.561f, 0.729f, 0.988f].

   Render the Sponza scene using path tracing with at least 20 samples per pixel and a max ray depth of 4 in run_rt. You may use more samples per pixel as you desire. Note that the sample path tracer considers a ray depth of 1 to be only primary rays with no indirect illumination rays. A depth of 2 will be primary rays plus one bounce indirect illumination. Shadow rays do not count towards the ray depth.

   The TRaX memory loader supports an option for specifying the maximum ray depth, if you don't want to hard-code it. Supply the argument `--ray-depth N`, which will set the value located in memory at: `loadi(TRAX_RAY_DEPTH)` to `N`.

   Before you start your final rendering, make sure your shading looks reasonable first by rendering with fewer samples. As a reference, below is the scene rendered with one sample per pixel (left) and 100 samples per pixel (right).



   If your image is brighter than this, you probably forgot to turn off the ambient term.

   When you are convinced your path tracer is correct (or that you are using the provided one correctly), make sure you can render a low resolution, low sample version in `simtrax`.

3. **Analysis** (60 points)

   For all experiments below, do not disable usimm or any of the caches.

   (a) Full TRaX Chip

   Starting with your best $performace/area$ TM configuration found in assignment 3, tile multiple TMs on to a chip for a total of 2560 thread processors, arranged however you desire.

   Recommended: You are free to use the below recommendation as your chip configuration, or try to improve on it for better $performace/area$ for your path tracer:

   - 32 thread-procs per TM
   - 20 TMs connected to an L2
   - 4 L2s
   - 256KB L2s with 16 banks each
   - The L1 configuration you determined in assignment 3

   Recall that the number of thread processors is $num\text{–}thread\text{–}procs * num\text{–}TMs * num\text{–}l2s$.

   (b) Ray Coherence and DRAM

   For the experiments below, render the conference scene at 128x128 resolution. Use either the recommendation from the previous section, or your own optimized chip configuration.

   Primary (camera) and point-light shadow rays are both fairly "coherent", as they have either a common origin or destination. All of these rays will likely access a similar small portion of the total scene data. Path tracing introduces indirect illumination rays which have, by nature, random directions and uncommon origins, and are thus "incoherent". These incoherent rays can potentially access any part of the scene data in an unpredictable way.

   To measure the effect of incoherent rays on performance we will attempt to control the level of incoherence by increasing the bounce depth of indirect illumination rays.

   i. For ray depths 1 through 4, holding samples per pixel at 1, plot:
      - Average DRAM read latency. This is the average of the reported read latencies for all DRAM channels. It is important to average all channels, as there can be high variance.
      - Average row buffer hit rate. This is the average "Read Page Hit Rate" for all DRAM channels. It is important to average all channels, as there can be high variance.

   ii. For 1 through 4 samples per pixel, holding ray depth at 1 (no indirect illumination rays), plot the same data as above.
      The trends in the plots from part (i) may not reveal ray incoherence alone. Since increasing the ray depth also increases the total number of rays, this may affect read latencies, e.g., if the read queues get full. To address this we will also measure the affects that increasing the *n*umber of rays has, without increasing incoherence. Increasing the number of samples per pixel should roughly give us the desired effect. Admittedly, the number of rays will not correspond exactly with the experiments from part (i), but they should be close enough to see any related trends. Furthermore, more samples will actually *i*ncrease the ray coherence slightly, but the trends we see should still be revealing.

   For the experiments above, this is an example of the command line arguments used:

```
--num-thread-procs 32
--config-file <path to your config file determined in part (a)>
--num-TMs 20
--num-l2s 4
--num-icaches 2
--num-icache-banks 16
--load-assembly <path to your path tracer>
--simulation-threads <max of 8 recommended on a 4 or 8-thread machine>
--ray-depth <varies from 1 to 4>
--num-samples <varies from 1 to 4>
```

4. **TRaX Output** (10 points)

   Provide the simtrax output running your program on each of the 8 data points from the above analysis.

5. **Code listing (only if you used your own code, you do not need to hand in the sample path tracer)**

   Link your source code to your web page (preferably .tar).

6. **Creative Image** (10 points):

   Generate any image of your choice that shows off the effects of global illumination using path tracing.

What to turn in:

By midnight on the due date, you should send e-mail to teach-cs6958@list.eng.utah.edu with the following information:

1. **URL:** A pointer to a web page containing the following information:
   (a) Creative image
   (b) Link to source code
   (c) Analysis (see above)
   (d) TRaX output (see above)
2. **Time required:** How many hours did it take you to complete this assignment?
3. **Difficulty of assignment:** Was the assignment difficult or not? Feel free to expound or to be brief.

You will not be graded on these last two items. They will be used to help improve the class in future assignments and in future years.