This ray-tracer based on a application developed at the MIT[1], is one of the early project using the OSS library, focus is not on development and improvement of the ray-tracer itself, more on testing and presenting transactional shared memory. All allocations of objects and the image file is made in shared memory and can be influenced with some parameters at start of program.

Installing

1. stand alone

To compile the ray-tracer make sure you've installed glib and the OSS Library correctly. Modify the path to OSS library and sources in makefile.

Now just type make and get started.

2. within OSS

just run the OSS makefile, the ray-tracer will be build automatically.

Running

To start the ray-tracer you only need -i or --address option to specify ip of first host. And -i or --address and -b or --bootstrap option on remote hosts. Make sure connections.conf is present.

Start all nodes which should participate, before starting the tracing process.

To configure the tracing progress, you will be asked some parameters on the first node.

1. Consistency: type 's' for strict and 't' for transactional consistency
2. Number of Nodes
3. Number of access: number of accesses before eot.
4. Pattern: specify one of 'l', 'c', 'p', 'x' or 'm'. 'l' for line by line, 'c' for column by column, 'p' for x partitions, 'x' for every Xth dot, 'm' for matching pages.
5. Scene: one of 1 to 3
6. Rows
7. Columns

When rendering is done, you can give new parameters and render another scene.

1 Tomas Lozano-Perez and Jovan Popovic: “Project 5: Ray Tracing”. http://graphics.lcs.mit.edu/classes/6.837/F01/Project05/project5.html, MIT, 2001.

Scenes

To write own Scenes you have to write a new c-file analogical to SceneDemo1.c. Basically there are 3 Scenes in this project. Scene1 very simple with one sphere in center and a few bowls around and only a few lights, Scene2 very complex with some arrangements of bowls and reflecting walls. Scene3 just for amusing OSS written with bowls.