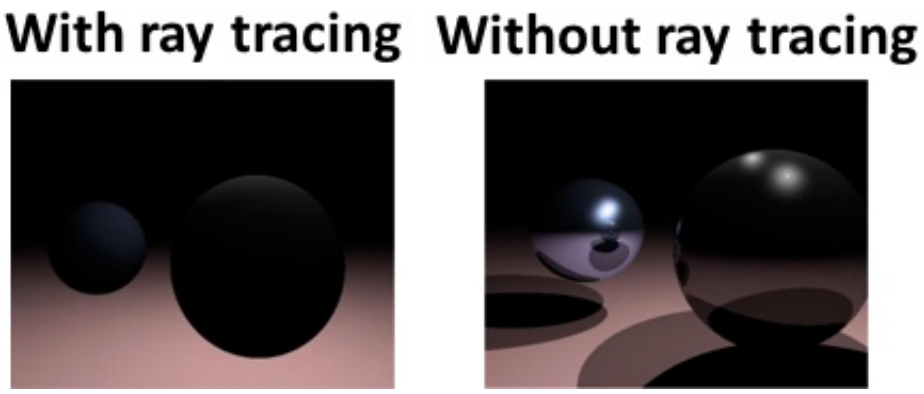
Week 8: Multithreaded Programming

Ray Tracing

- advanced computer graphics technique for rendering 3D images
- mimics the propagation of light through objects
- simulates the effects of a single light ray as it's reflected or absorbed by objects in the images



Computational Resources

- ray tracing produces a very high degree of visual realism at a high cost
- algorithm is computationally intensive
    - good candidate for multithreading (**embarrassingly parallel**)

Basic pthread Functions

1. `pthread_create`: creates a new thread within a process
2. `pthread_join`: waits for another thread to terminate
3. `pthread_equal`: compares thread ids to see if they refer to the same thread
4. `pthread_self`: returns the id of the calling thread
5. `pthread_exit`: terminates the currently running thread

`pthread_create`

- **function**: creates a new thread and makes it executable
- can be called any number of times from anywhere within code
- return value
    - **success**: 0

- **failure**: error #

`pthread_join`

- **function**: makes originating thread wait for the completion of all its spawned thread's tasks
- without join, the originated thread would exit as soon as it complete its job
  - a spawned thread can get aborted even if it is in the middle of its chore
- return value:
  - **success**: 0
  - **failure**: error number

Arguments

- `int pthread_join(pthread_t tid, void **status);`
  - **tid**: thread ID of thread to wait on
  - **status**: the exit status of the target thread is stored in the location pointed to b *status
    - pass in NULL if status unneeded

```c
#include <pthread.h>
#define NUM_THREADS 5

void *PrintHello(void *thread_ num) {
    printf("\n%d: Hello World!\n", (int) thread_num); }

int main() {
    pthread_t threads[NUM_THREADS];
    int ret, t;
    for(t = 0; t < NUM_THREADS; t++) {
        printf("Creating thread %d\n", t);
        ret = pthread_create(&threads[t], NULL, PrintHello, (void *) t);
        if(ret) {
            printf("Error creating thread. Error code is %d\n", ret");
            exit(-1); }
    }

    for(t = 0; t < NUM_THREADS; t++) {
        ret =  pthread_join(threads[t], NULL);
        if(ret) {
            printf("Error joining thread. Error code is %d\n", ret");
            exit(-1); }
    }
}
```

Homework 8

- download the single-threaded ray tracer implementation
- run it to get output image
- multithread ray tracing
  - modify main.c and Makefile

- run the multithreaded version and compare resulting image with single-threaded one

- built a multi-threaded version of Ray tracer
- modify "`main.c`" and "`Makefile`"
    - `#include <pthread.c>` in "main.c"
    - use "`pthread_create`" & "`pthread_join`" in `main.c`
    - link with `-lpthread` flag (LDLIBS target)
- **`make clean check`**
    - outputs "1-test.ppm"
    - can see "1-test.ppm"
        - sudo apt-get install gimp (Ubuntu)
        - X forwarding (lnxsrv)
        - gimp 1-test.ppm