Multithreading/Parallel Processing

CS 35L
Spring 2018 - Lab 3

Assignment 7 Reminder
Beaglebone Wireless

For assignment 7, you will need a Seeed Studio BeagleBone Green Wireless Development Board

We'll be using them **next week**!

See the specs for assignment 7 for details: https://web.cs.ucla.edu/classes/spring18/cs35L/assign/assign7.html

Parallelization

- **Parallelization** is the practice of accelerating a program by running multiple sections simultaneously
- **Process forking** allows for a process to split into multiple subprocesses that run simultaneously
  - Switching between processes (context switching) on the CPU is expensive
  - Inter-process signalling is difficult (eg. pipes)
- **Multithreading** is an efficient type of parallelization
  - **Thread switches are less expensive**
  - Inter-thread signalling is easy via **shared data**
  - Need **synchronization** among threads accessing the same data
    - e.g. Mutex.lock(), Mutex.unlock()

Pthread API

```
#include <pthread.h>
```

- int **pthread_create**(pthread_t *thread,
  const pthread_attr_t *attr,void*
  (*thread_function) (void*), void *arg);
  - Returns 0 on success, otherwise returns non-zero number
- void **pthread_exit**(void *retval);
- int **pthread_join**(pthread_t thread, void **retval);
  - thread: thread ID of thread to wait on
  - retval: the exit status of the target thread is stored in the location pointed to by *retval
    - Pass in NULL if no status is needed
  - Returns 0 on success, otherwise returns non zero error number

Simple Example

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

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

int main() {
    pthread_t threads[NUM_THREADS];
    int ret, t;
    for(t = 0; t < NUM_THREADS; t++) {
        ret = pthread_create(&threads[t], NULL,
                             PrintHello, (void*)t);
        // check return value
    }
    for(t = 0; t < NUM_THREADS; t++) {
        ret =  pthread_join(threads[t], NULL);
        // check return value
    }
}
```

Race Conditions

Execution order of threads is non-deterministic

Race Condition:

`Total = Total + val1`	`Total = Total - val2`

What value does Total end with?

Solution: Mutexes for synchronization

Mutex Example (w/o mutexes)

```
#include <pthread.h> …

const int nthreads = 5;
pthread_t tid[nthreads];

int counter;

void* doSomeThing(void *arg) {

    counter = counter + (int)arg;

}

int main() {
    int i;
     counter = 0;

     for (i = 1; i <= nthreads; ++i)
        pthread_create(&(tid[i]), NULL, &doSomeThing, i);
     for (i = 1; i <= nthreads; ++i)
        pthread_join(tid[i], NULL);

     printf("Counter: %d\n", counter);
     return 0;
}
```

Mutex Example (w/ mutexes)

```
#include <pthread.h> …

const int nthreads = 5;
pthread_t tid[nthreads];
pthread_mutex_t lock;
int counter;

void* doSomeThing(void *arg) {
    pthread_mutex_lock(&lock);
    counter = counter + (int)arg;
    pthread_mutex_unlock(&lock);
}

int main() {
    int i;
     counter = 0;
    pthread_mutex_init(&lock, NULL);
     for (i = 1; i <= nthreads; ++i)
        pthread_create(&(tid[i]), NULL, &doSomeThing, i);
     for (i = 1; i <= nthreads; ++i)
        pthread_join(tid[i], NULL);
    pthread_mutex_destroy(&lock);
     printf("Counter: %d\n", counter);
     return 0;
}
```

Deadlock

Deadlock:

`mutex1.lock();`	`mutex2.lock();`
`mutex2.lock();`	`mutex1.lock();`

What happens if each thread is waiting on a resource that is locked by another?

Solutions

- Ignore (simple to implement, but unsafe)
- Detect (slightly complicated): directed graph cycle checking
- Prevent (very complicated): wait-for-graphs, banker’s algorithm, etc.

SIMD vs MIMD

- Multiple Instruction Multiple Data (MIMD)
  - Performs multiple actions on any number of data pieces simultaneously.
  - Standard CPU multithreading (eg. pthread)
- Single Instruction Multiple Data (SIMD)
  - Performs the same action on multiple pieces of data simultaneously.
  - Best for algorithms with little data interaction.
  - Typical of most modern parallel specialized hardware, including GPUs (CUDA).

Homework 6

- Download the single-threaded raytracer 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

Homework 6

- Build a multi-threaded version of Ray tracer
- Modify “main.c” & “Makefile”
  - Include <pthread.h> 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”
  - See next slide on how to convert ppm

Ray-tracing

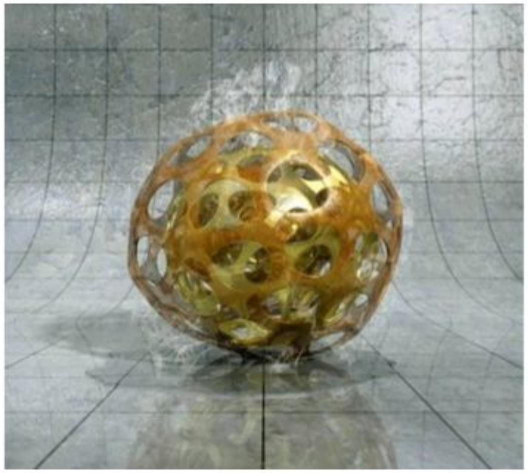


Image Source: POV Ray, Hall of Fame hof.povray.org

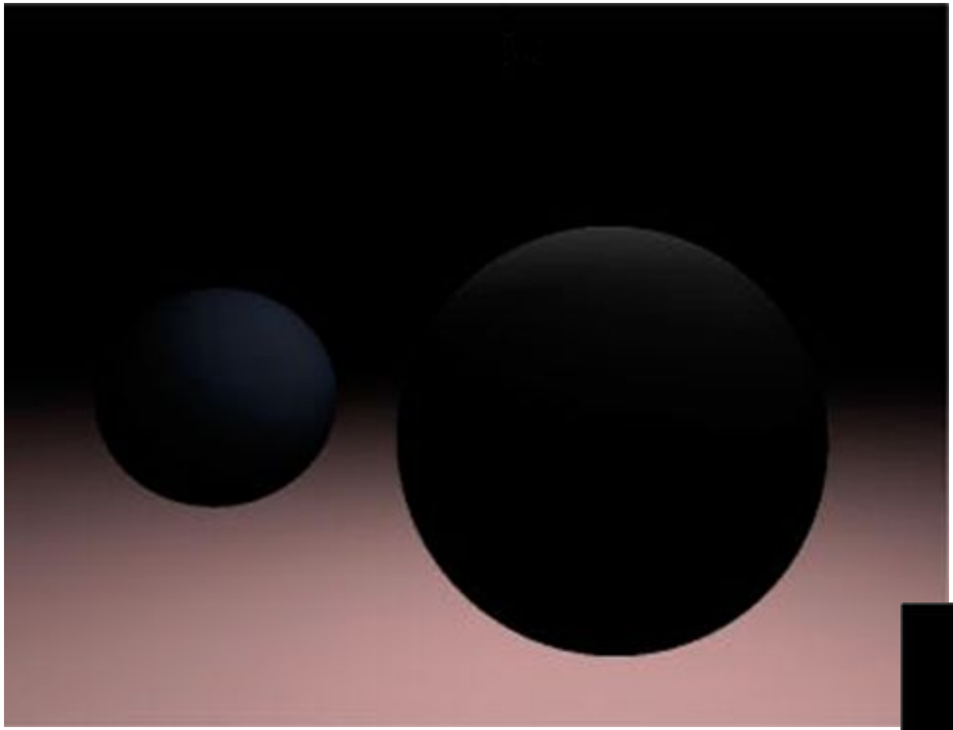
Motivation

Siggraph 2017 technical papers

Siggraph Asia 2017 technical papers

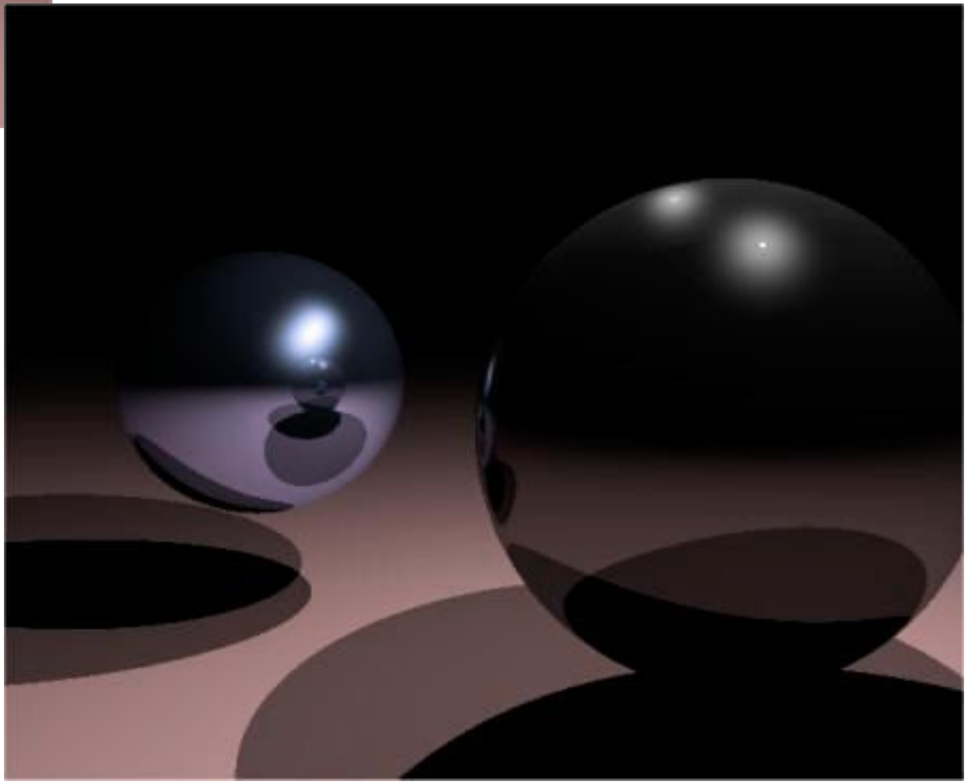
Ray-Tracing

- **Powerful rendering technique in Computer Graphics**
- **Yields high quality rendering**
  - Suited for scenes with complex light interactions
  - Visually realistic
  - Trace the path of light in the scene
- **Computationally expensive**
  - Not suited for real-time rendering (e.g. games)
  - Suited for rendering high quality pictures (e.g. movies)
- **Embarrassingly parallel**
  - Good candidate for **multi-threading**
  - Threads need **not synchronize** with each other, because each thread works on a different pixel



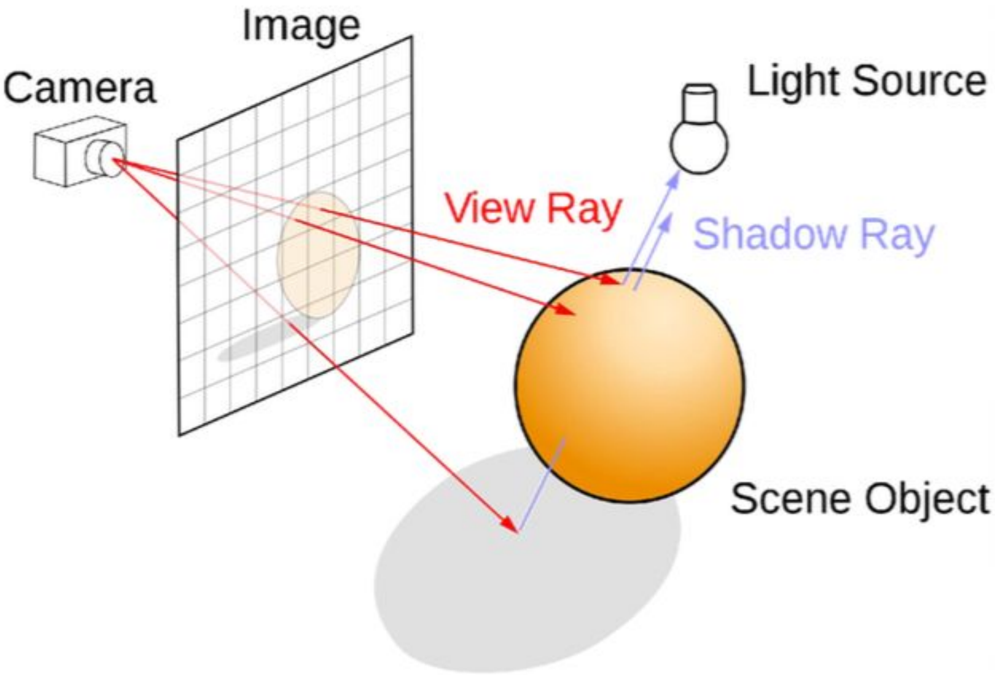
Without ray tracing

With ray tracing



Ray-tracing

- Trace the path of a ray from the eye
  - **One ray per pixel** in the view window
  - The color of the ray is the color of the corresponding pixel
- Check for **intersection** of ray with scene objects.
- **Lighting**
  - **Flat shading** – The whole object has uniform brightness
  - **Lambertian shading** – Cosine of angle between surface normal and light direction



Viewing a ppm file

- How to view a ppm file?
  - ppmtojpeg
    - ppmtojpeg - is more lightweight than gimp. If you don't already have it, you can download ppmtojpeg as part of the Netpbm package here (windows,linux,mac)
    - This program comes with many Linux distributions as well as with Cygwin for Windows; it is also installed on the SEAS Unix machines.
      - ppmtojpeg input-file.ppm > output-file.jpg
  - Gimp:
    - sudo apt-get install gimp (Ubuntu)
    - www.gimp.org - or install on your computer (windows,linux,mac)
      - scp the file to your local folder to view it
        - » conversion tutorial with gimp
      - X forwarding (lnxsrv)
        - » gimp 1-test.ppm

1-test.ppm

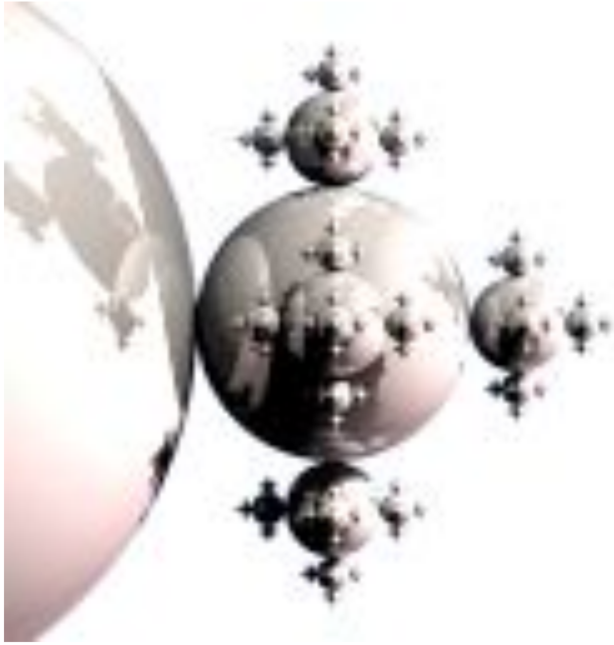
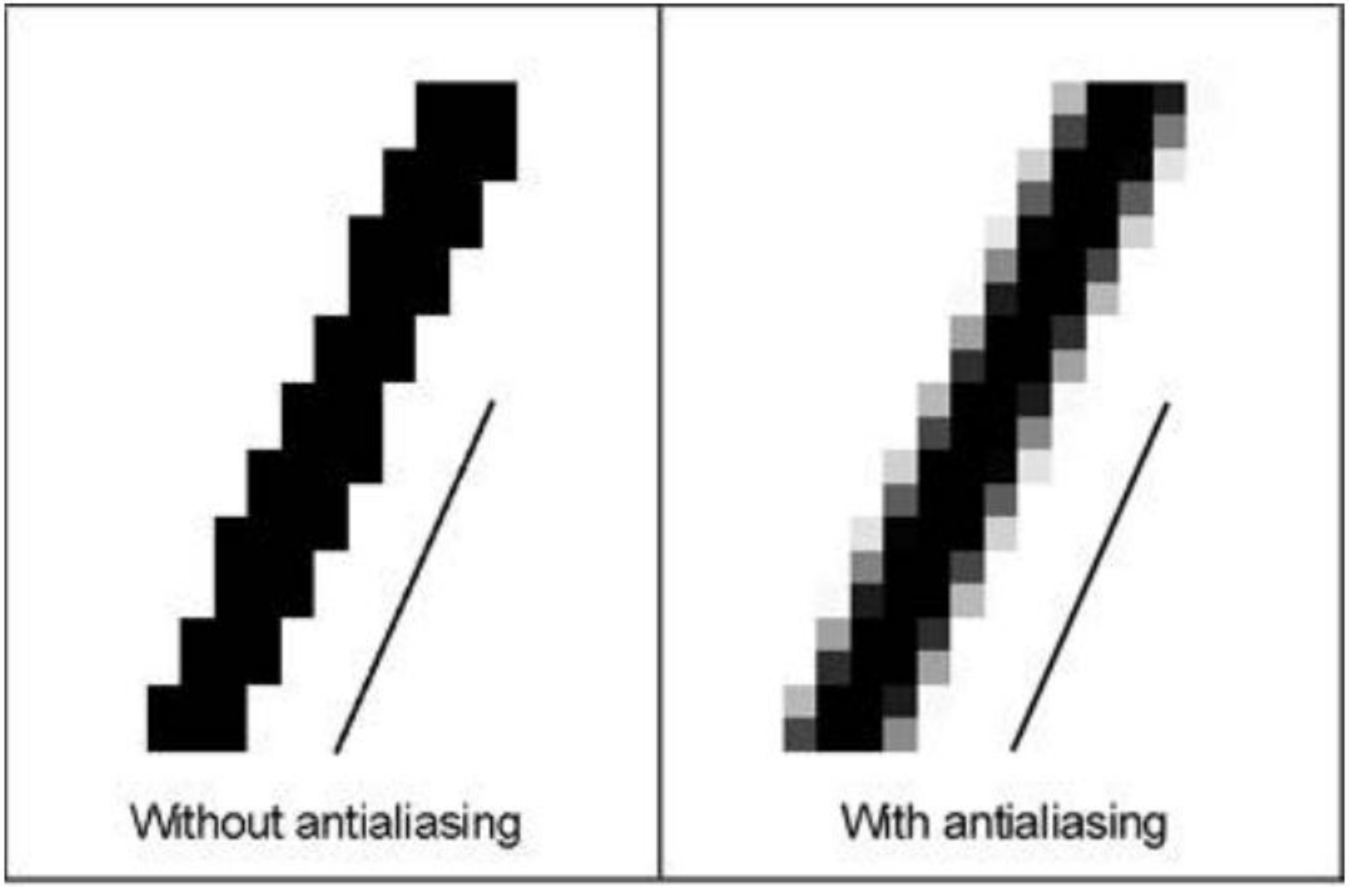


Figure. 1-test.ppm

Homework 8 - antialiasing



Assignment 6 Hints

- GNU sort has default parallelization options, see the official docs.
- You shouldn’t need mutexes for the homework portion.
- Make sure you read Guangyu’s guidelines on piazza.