CS 35L

Week 8

TA: Tomer Weiss
Feb-23-2016

goo.gl/ZChcrJ

Slides

Announcements

- Student presentations today:
  - Michael: Mining Social Media Can Help Disaster Response Efforts
  - Gauri: UW brain implant could help paralyzed limbs move again
  - Yujing: Enabling human-robot rescue team

  web.cs.ucla.edu/classes/winter16/cs35L/assign/assign10.html

- Next week:
  - Write your topic here
  - Not registering you topic beforehand may result in rescheduling of your presentation
  - For reference on presentation, grading, please refer to this rubric.

Multithreading/Parallel Processing

Week 8

Multithreading reminder

- Multithreads is an efficient way to **parallelize** tasks
- **Thread switches are less expensive** compared to process switches (context switching)
- Inter-thread communication is easy, via **shared global** data
- Need **synchronization** among threads accessing same data
  - e.g. Mutex.lock(), Mutex.unlock()

Ray-tracing

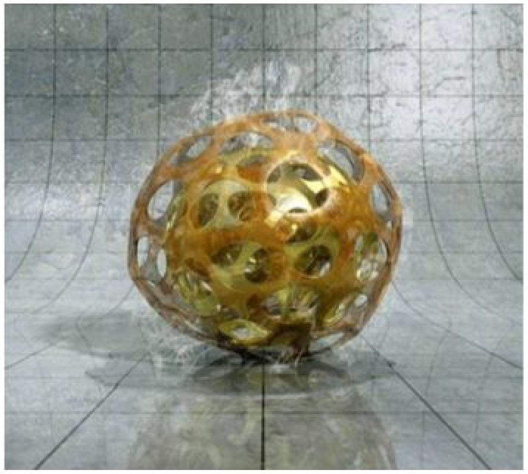


Image Source: POV Ray, Hall of Fame hof.povray.org

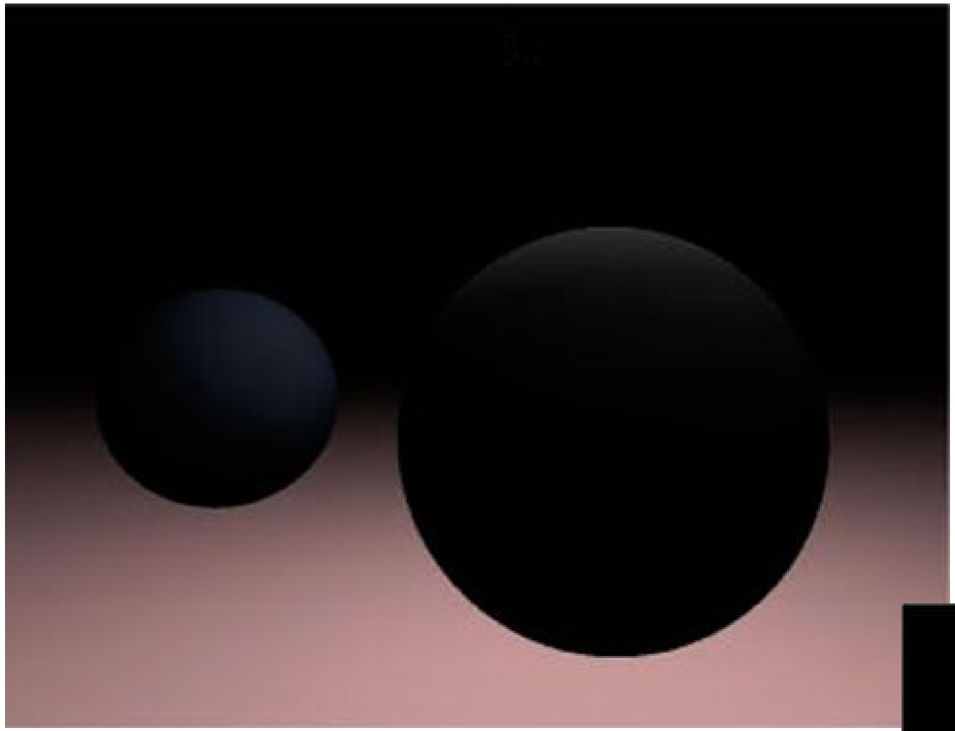
Motivation

Siggraph 2015 technical papers

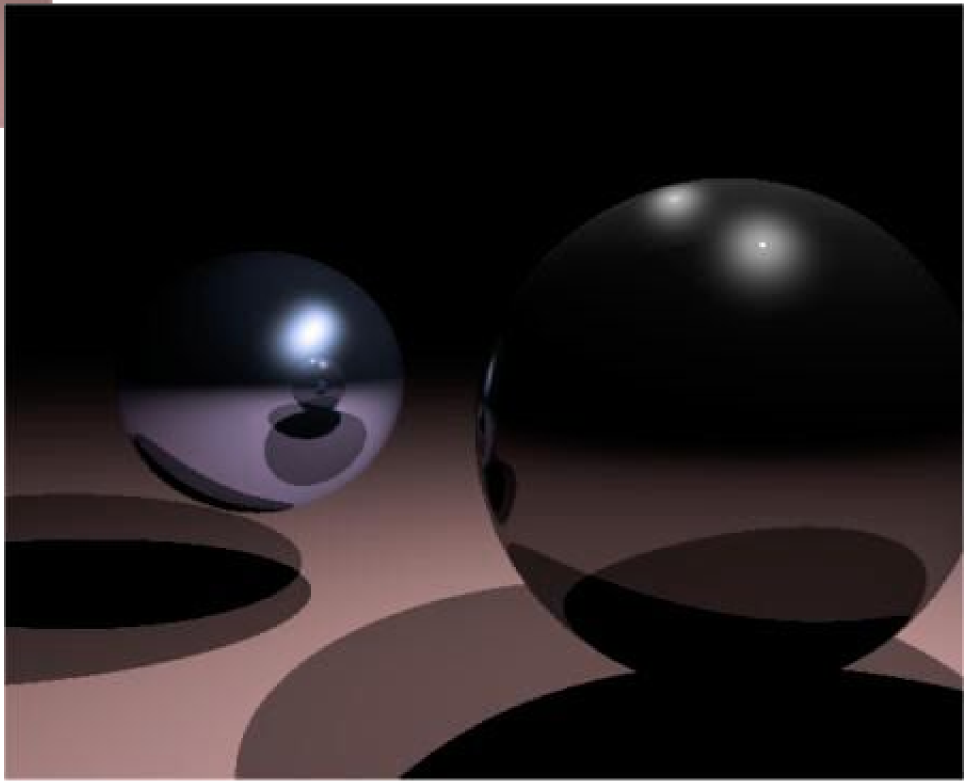
Siggraph Asia 2015 technical papers

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



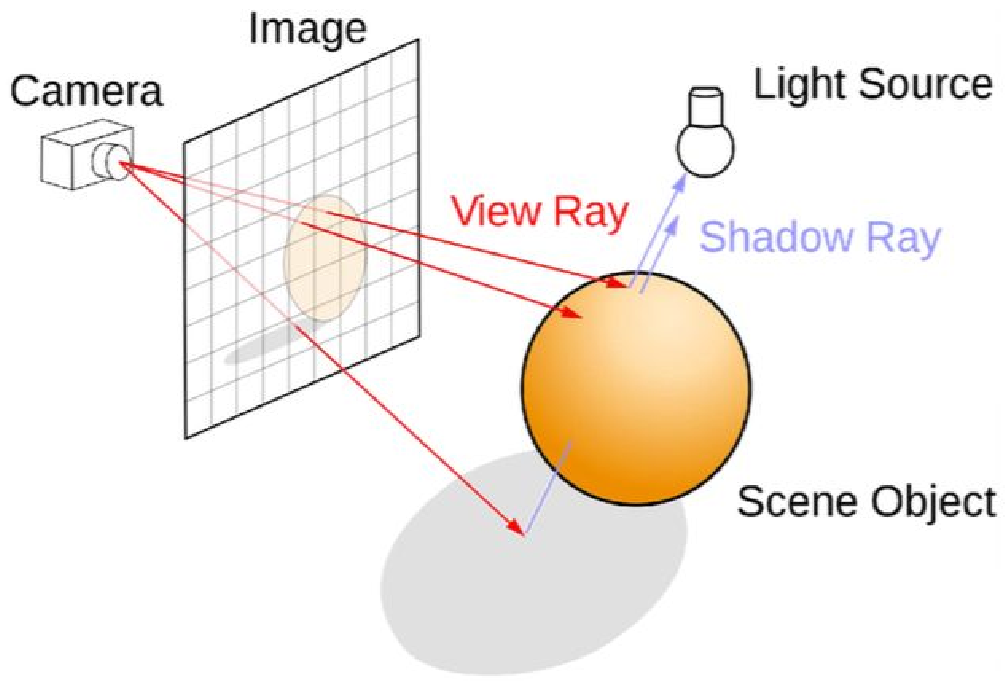
Without ray tracing



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



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

```c
#include<pthread.h>  //Compile the following code as - gcc main.c -lpthread
#include<stdio.h>
void* ThreadFunction(void *arg) {
  long tID = (long)arg;
  printf("Inside thread function with ID = %ld\n", tID); pthread_exit(0);}

int main(int argc, char *argv[]) {
  const int nthreads = 5; pthread_t threadID[nthreads]; long t;
  for(t = 0; t < nthreads; ++t) {
    int rs = pthread_create(&threadID[t], 0, ThreadFunction, (void*)t);
    if(rs) {
      fprintf(stderr, "Error creating thread\n");
      return -1; }}
  printf("Main thread finished creating threads\n");
  for(t = 0; t < nthreads; ++t) {
    void *retVal;
    int rs = pthread_join(threadID[t], &retVal);
    if(rs) {
      fprintf(stderr, "Error joining thread\n");
      return -1;
  }}
  printf("Main thread finished execution!\n");
  return 0; }
```

Pthread API

pthread_join Example

```
#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);
                // check return value }

         for(t = 0; t < NUM_THREADS; t++) {
                ret =  pthread_join(threads[t], NULL);
                // check return value }
}
```

Homework 8

- 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 8

- 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”
    - sudo apt-get install gimp (Ubuntu)
    - X forwarding (lnxsrv)
    - gimp 1-test.ppm

1-test.ppm

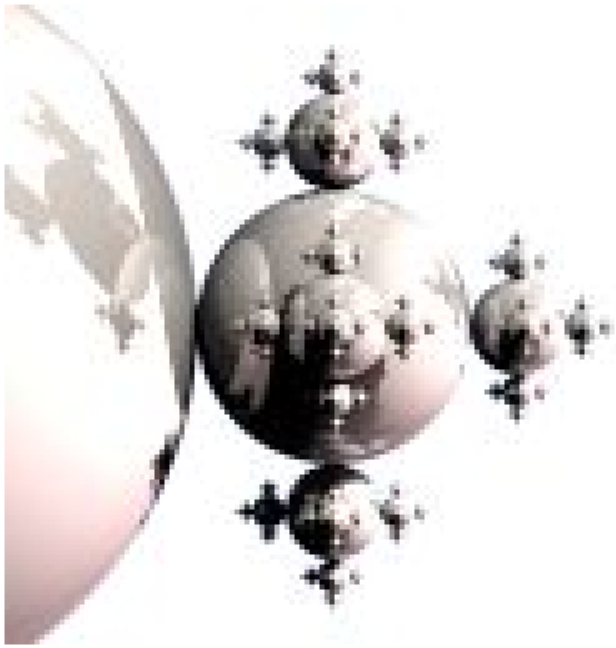
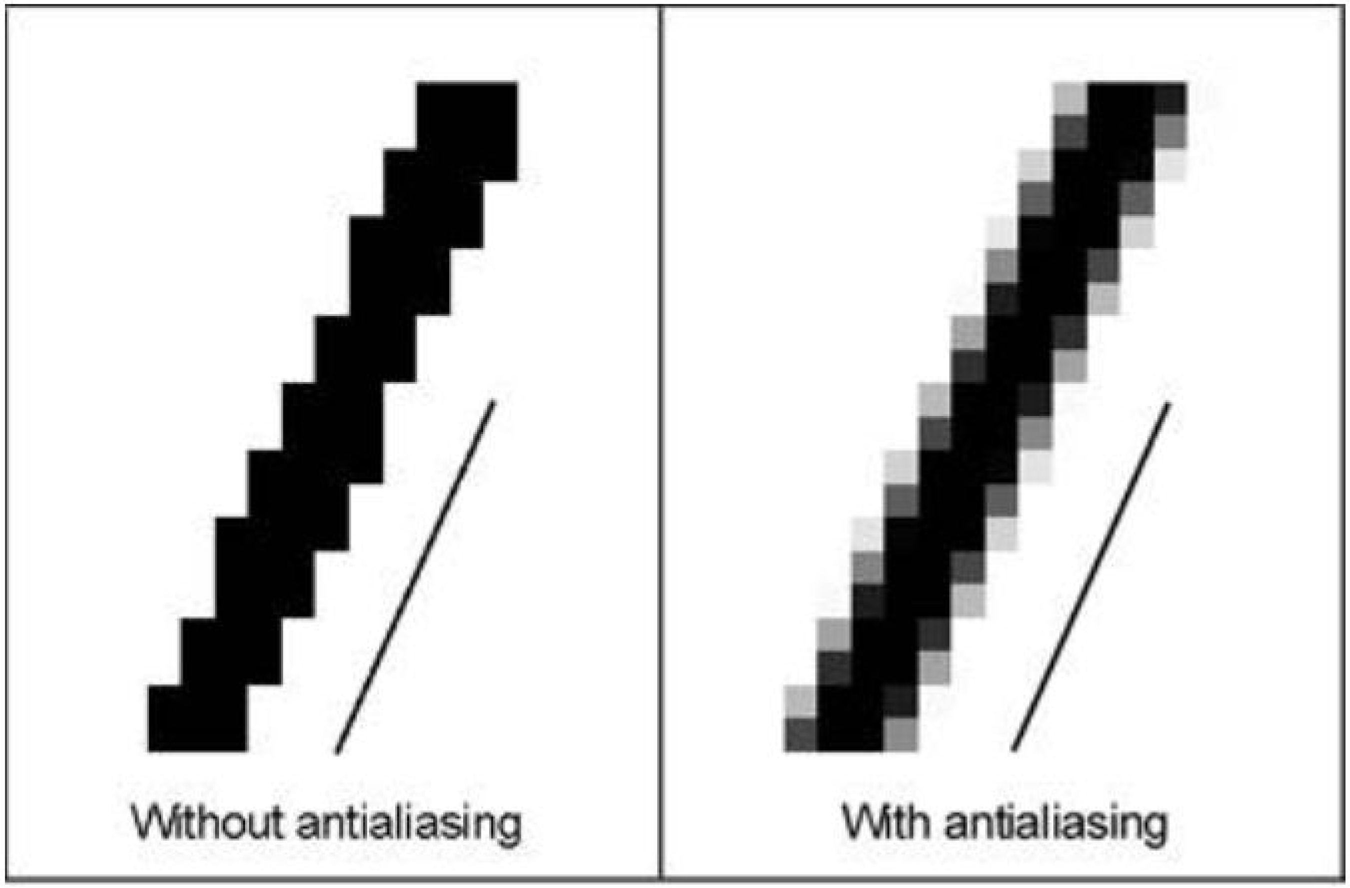


Figure. 1-test.ppm

Homework 8 - antialiasing



Lab

web.cs.ucla.edu/classes/winter16/cs35L/assign/assign8.html