Welcome to
CS 413 Introduction to
Ray and Vector Graphics

Instructor: Joel Castellanos
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/
Farris Engineering Center: 319

Course Resources

Textbook: Ray Tracing from the Ground Up
by Kevin Suffern

Blackboard Learn: https://learn.unm.edu/
- Assignement Drop-box
- Discussions
- Grades

Class website: http://cs.unm.edu/~joel/cs413/
- Syllabus
- Projects
- Lecture Notes
- Readings Assignments
- Source Code
Structure of the Course

- Studio: Elementally, we will spend the full semester building a single, well featured ray tracer.
- Stages the ray tracer total to 70% of course grade.
- No exams
- Class time (30% of course grade):
  - Lecture,
  - Discussion of Reading,
  - Show & Tell,
  - Code review.

Language and Platform

- The textbook (and class) examples are all in C++:
  - STL (Standard Template Library – this is multiplatform).
  - wxWidgets application framework (Windows and Linux).
- You may use the language of your choice. Using anything other than C++ will be more work since you will need to write and/or find language specific versions of the provided utility code. This will be taken into account during grading and some requirements will be relaxed.
- You may choose your target platform – BUT you must be able to demo that platform in class or in a location at UNM (i.e. Dome at ARTS Lab).
Assignment 1 & 2

For Wednesday, Aug 20:
- Read Chapters 1, 2 (skim and reference later) and 3.
- Be prepared to discuss solutions to questions 3.1, 3.2, 3.3 & 3.4, 3.5 & 3.6.

For Friday, Aug 22: "Bare-Bones Ray Tracer”.
- If not using wxWidgets, then it is fully enough to get the single red sphere displayed in a simple materialless, lightingless, orthographic projection straight down the z-axis.
- If using wxWidgets, then get the supplied code working with an additional 3 menu options:
  a) Exercise 3.1 (single, simple orthographic sphere)
  b) Exercise 3.2 (part a, with $z_w = 100.0$)
  c) Exercise 3.3 (sphere with different center, radius & color).

3D Graphics Coordinate System

Throughout this course, the usual way of specifying the location of a point in 3D space is as Cartesian coordinates using an ordered triplet, $(x, y, z)$, of floating point numbers (float or double).
Ray Tracing versus Ray Casting

1) Define some objects.
2) Specify a material for each object.
3) Define a window whose surface is covered with pixels.
4) For each pixel:
   a) From the center of each pixel, cast a ray towards the objects.
   b) Compute the nearest (if any) hit point of the ray with the objects.
   c) If the ray hits an object:
      Use object's material and the lights to compute pixel color.
   d) else:
      Set the pixel color to black.

Simple Ray Casting Ray Tracer
### Four Common Types of Rays

**Primary Rays** start at the centers of pixels for parallel viewing and at the camera for perspective viewing.

**Secondary Rays** are reflected and transmitted rays that start on object surfaces.

**Shadow Rays** are used for shading and start at object surfaces.

**Light Rays** start at the lights and are used to simulate some aspects of global illumination.

Chapter 3 covers only Primary Rays and only those starting at the centers of pixels.

---

### wxWidgets: Getting Debug Output

```c++
wxMessageOutputDebug().Printf("Hello=%d\n", 5);
```

Place where Frame is created and menu items are added:

```c++
xraytracer.cpp
wxraytracerFrame::wxraytracerFrame
```

Place where objects are added to scene:

```c++
BuildShadedObjects.cpp World::build(void)
```

I added options to:

```c++
World::build_Joel_x(void)
```
wxWigets: Getting Debug Output

```c
wxMessageOutputDebug().Printf("Hello=%d\n", 5);
```

Finding your way around the code: