- # Class: Image Synthesis
  - ## Course Description
    - **From course site:**
      - **Description:** The course will cover mathematical and artistic foundations of image synthesis. The topics include but not limited to Overview of Computer Graphics, Introduction to Ray Tracing, Illumination of spheres and planes, Illumination of cones and cylinders and shadow determination, Texture mapping planes and spheres. Ray tracing polygons, Specular reflection, Transparency/refraction, Sampling theorem and distributed ray tracing, Volume rendering implicits, Volume rendering fractals, Image based Lighting, Backward ray tracing and caustics, Photon Mapping and radiosity.
      - **Goals:** To teach mathematical, artistic and computational principles of image synthesis. This course will provide a thorough grounding in the state of the art in 3D rendering and shading within the context of computer graphics, and digital effects. It is designed to prepare students to (1) understand existing 3D rendering systems; (2) write their own software for synthesizing images; and (3) undertake creative work and research involving 3D rendering Students read, discuss, and are tested on hand-out material, and complete a series of exercises on the computer.
      - **Learning Outcomes:** Upon completion of this course, students will know the state of the art in 3D rendering and shading methods of computer graphics and visualization. They will understand existing systems and they will be able to write their own 3D rendering software to obtain digital images and they will be able to undertake creative work and research involving creation of 3D digital images.
      - **Programming Projects:** Class projects will involve programming and making use of graphics libraries. Work may be done on any computer using any programming language such as Java, C++ or Processing. It is also allowed to use a wide variety of graphics libraries such as OpenGL and the OpenGL interface API GLUT (OpenGL Utility Toolkit), GLSL or CUDA. On the other hand, high level operations that are key the learning objectives of the this class such as object and ray intersections are not allowed to use if they are provided by the specific graphics library or programming language you use.
    - Reference Link

- # Final Project Ray Tracer Implementation
  - ## Project Description
    - This program is the summation of a semester's worth of class assignments, each building up to the final product, a ray tracing rendering program. In computer graphics, ray tracing is an approach for modeling how light travels within a 3D modeled scene, influencing how said scene is rendered. Ray tracing simulates optical effects such as reflection, refraction, depth of field, motion blur, caustics, and ambient occlusion.
    - The final version of my ray tracing renderer implements the following:
      - Diffuse shading
      - Specular reflection
      - Refraction and Fresnel
      - Shadow Casting
      - 3D Camera manipulation and ray casting
      - Texture, bump, and normal mapping.
      - 3D object rendering
      - Animation
    - The particular version of my ray tracing renderer renders out 3 spheres and a plane. Each object is textured mapped and reflects a foreground plane with lighting controlled by way of pixel values from a read image.
    - **Contents**:
      - Code for the final iteration of the ray tracer can be found in the subdirectory, *RayTracerCode.*
      - Images and videos produced from the ray tracer throughout the semester can be found in the subdirectory, "Ray_Tracer_Generated_Images".
  - ## Language
    - C++
  - ## Operation
    - Run the command "make -f makefile" to compile the program.
    - To run the program, type in the command line, "./rayTracerFinal <source.ppm>"
      - <source.ppm> can be any .ppm file stored in the subdirectory of RayTracerCode/ppmFolder