Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

Brian Russin

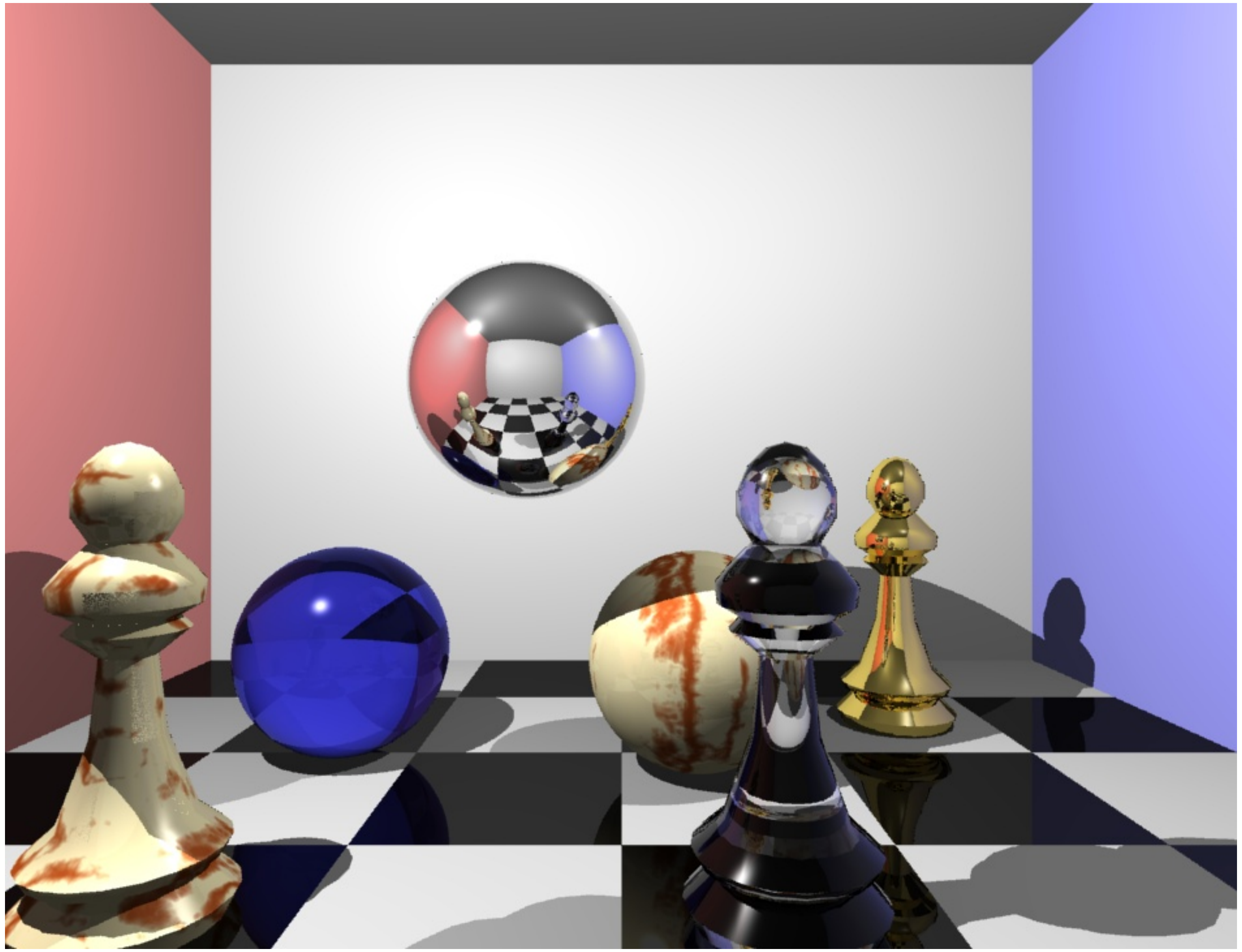
JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Module 1A
Course Introduction

Module 1 Overview

- **Introduction**
- **Picking / Selection**
- **Bounding Volume Types**
  - Forming bounding volumes
- **Ray-Object Intersections**
  - ray/sphere
  - ray/cylinder
  - ray/box
  - ray/triangle
  - ray/polygon

Course Introduction and Outline



Ray Tracing Project. Douglas Paul. 2010

Course Description

- **Advanced rendering topics** in computer graphics
  - extends course 605.667 Computer Graphics
  - larger, more complex graphics applications
- **Topics**
  - surface representations including parametric representations
  - subdivision surfaces, and procedural methods
  - advanced texture mapping techniques and shadow generation
  - programmable shaders
  - advanced rendering techniques
    - ray-tracing
    - radiosity
- **Practical application** using C++ and OpenGL
  - 2 laboratory exercises
  - Final project

Course Materials

- Textbook:
  - **Real-Time Rendering**, Fourth Edition, Akenine-Moller, Haines, Hoffman, Pesce, Iwanicki, Hillaire
    - The third edition is also acceptable, but the fourth edition is better
- Other references:
  - **An Introduction to Ray Tracing**, Glassner
  - **Computer Graphics Using OpenGL**, F. S. Hill and Stephen M. Kelley, 3rd Edition
  - **Real-Time Collision Detection**, Christer Ericson
  - **Interactive Computer Graphics**: A Top-Down Approach Using OpenGL, Sixth Edition, Edward Angel
  - **OpenGL Programming Guide**: The Official Guide to Learning OpenGL, Version 2, Fifth Edition, Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis.
  - **OpenGL Shading Language**, Randi Rost
  - **3D Computer Graphics**, Alan Watt, 3rd Edition
  - **3D Game Engine Design**, David Eberly

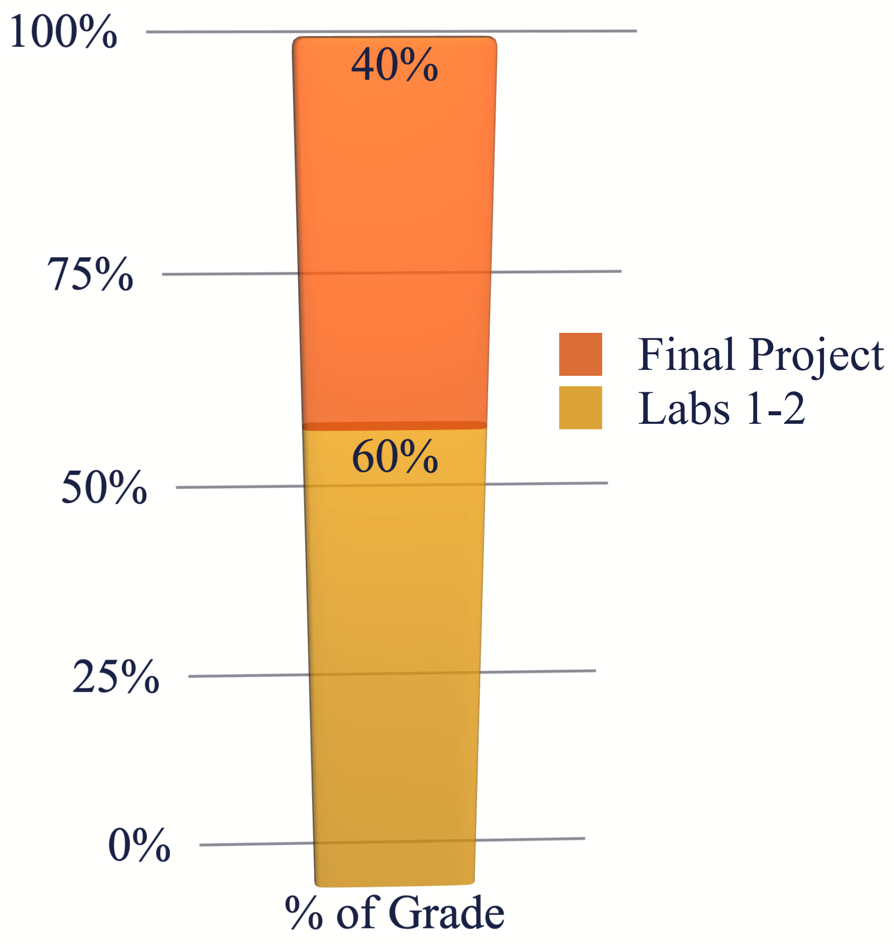
Laboratory

- Most of the work in this course will be LAB work!
- Use PC, MacOS, or Linux, and C++
  - Microsoft Visual C++ 2021
  - Xcode
  - CMake
- Will use OpenGL and SDL
- Extends the framework developed in 605.667 Computer Graphics
  - Geometry library
  - Scene graph library
  - Camera class

Grading Policy

- Laboratory Exercises: **60%**
  - **2** Graded lab exercises
- Final Project Laboratory Project: **40%**

- Labs: **create your own scene**
  - Will be expected to meet a **minimum set of criteria**
- Final project will include a **demonstration**



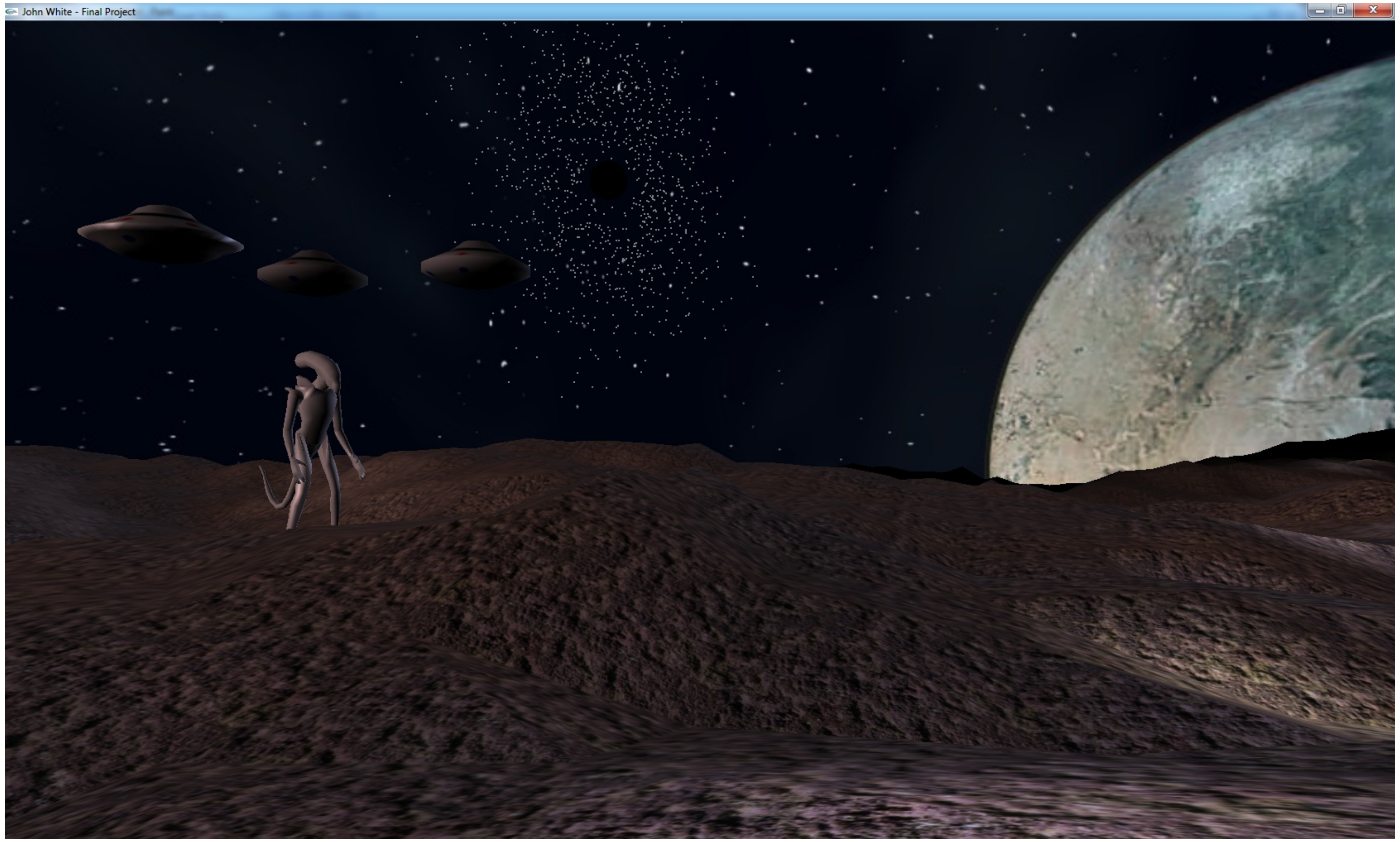
Laboratory Projects

- **Ray tracing**
  - Develop a simple, recursive ray tracer
  - Extend geometry library to support ray-object intersections and bounding volumes
- **Scene graph and advanced modeling**
  - Scene graph extensions
    - View frustum culling, level of detail selection, shader selection
  - Advanced modeling
    - Parametric surfaces
    - Subdivision surfaces
    - Fractal terrain
- **Independent project**
  - Student selected topic
    - Examples include advanced texture methods using programmable shaders, ray-tracer extensions, lighting and shading enhancements
  - Includes a demonstration with short presentation

Grading Laboratory Exercises

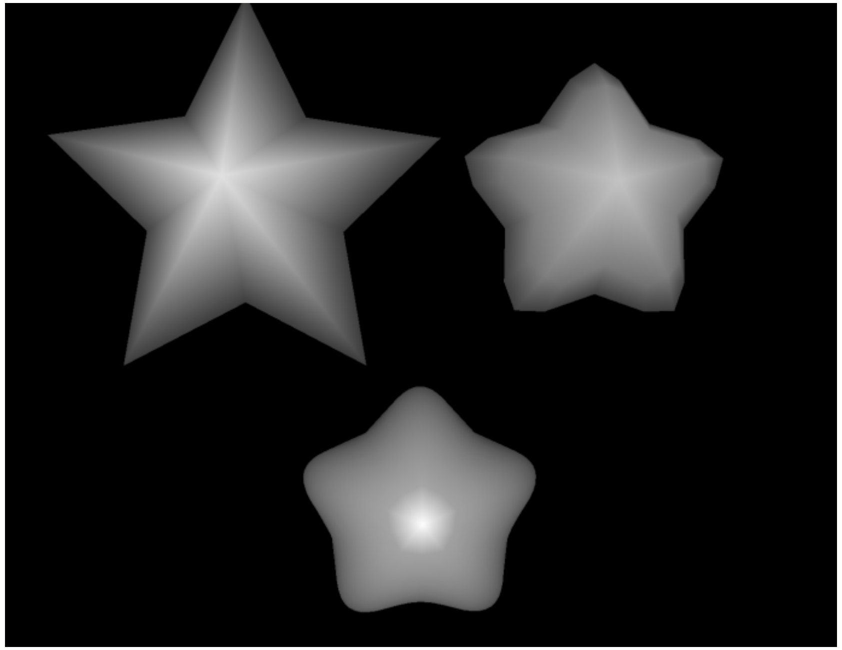
- Will provide grading **guidelines for each project**
- Principle grade is on **correctness of algorithms and resulting display**
- **Efficiency** is important in graphics programming
  - Text often focuses on efficiency
  - Slight deductions for inefficient or complex code
    - Even where correct presentation is achieved
- **Partial credit** given for incomplete portions of the exercise
  - Especially if comments are present explaining your intentions and problems you might have had
- Expect **comments** in the code
  - Function level – short description of the purpose of the method
  - Brief comments on implementation when complexity warrants

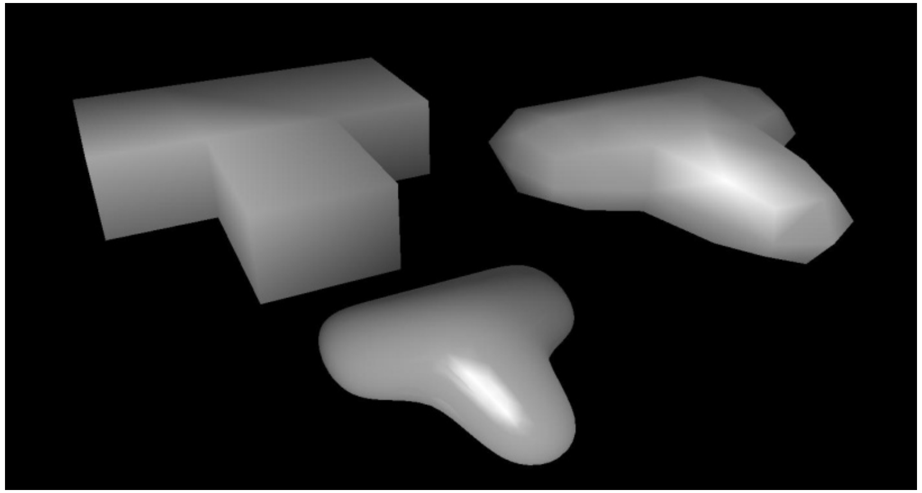
Sample Labs



John White. 2010. Fractal terrain, particle system, 3D model import

Sample Labs





Brian Kohan. 2010. Subdivision surfaces.

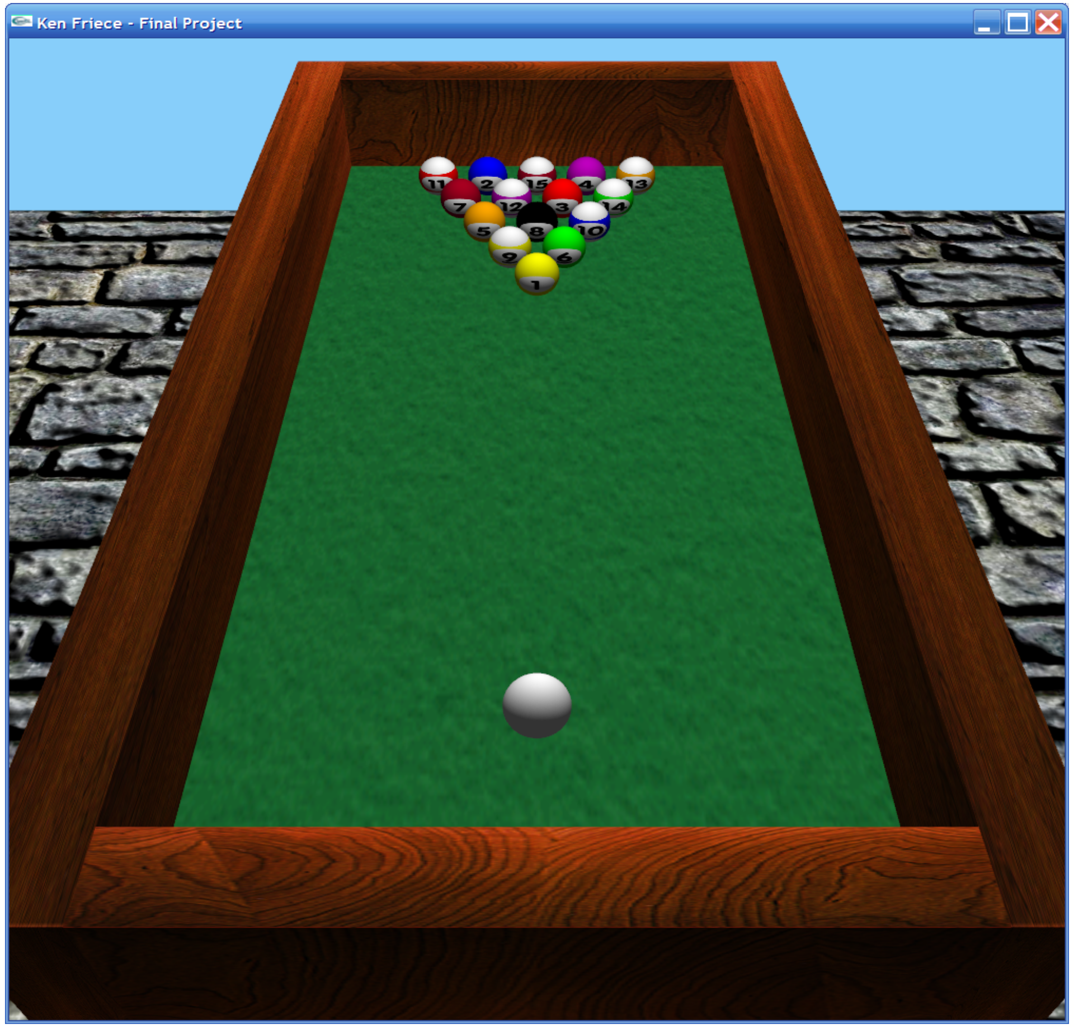
Sample Labs

William Dogan. 2010. Subdivision surfaces.

Sample Labs

Kunai Desai. 2010.
Bump mapping.

Sample Labs



Pool Table

Ken Friece 2006

Incorporates:

Dot-product bump-mapping

Collision detection