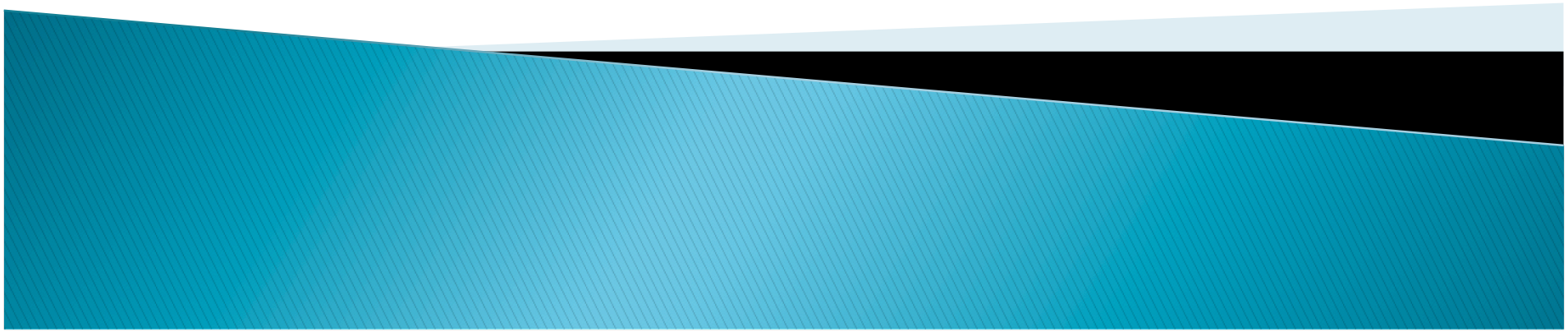
15-462 Project 1

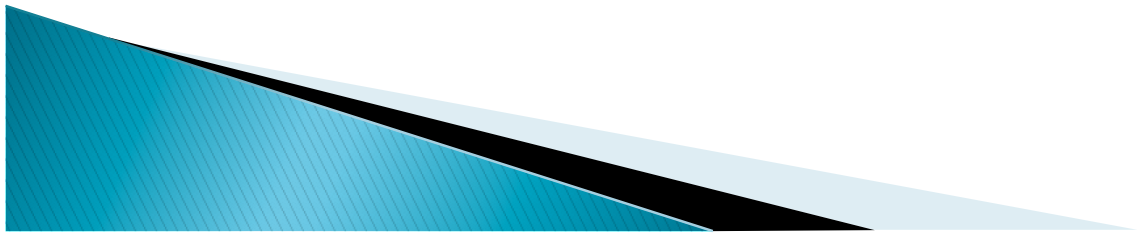
Eric Butler

January 20, 2009



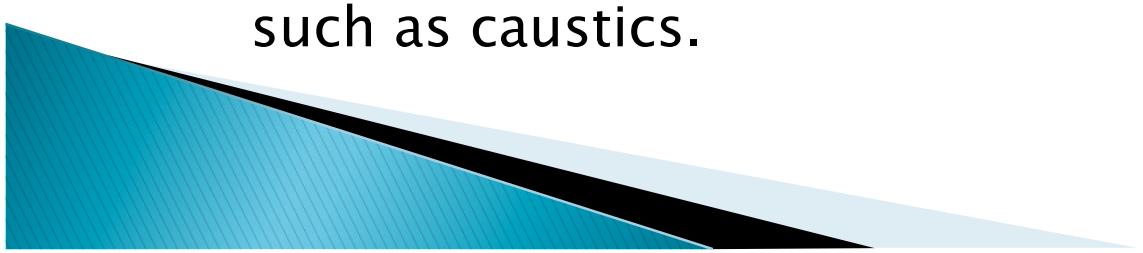
Overview

- Overview of the 4 projects
- Description of project 1
- Introduction to starter code

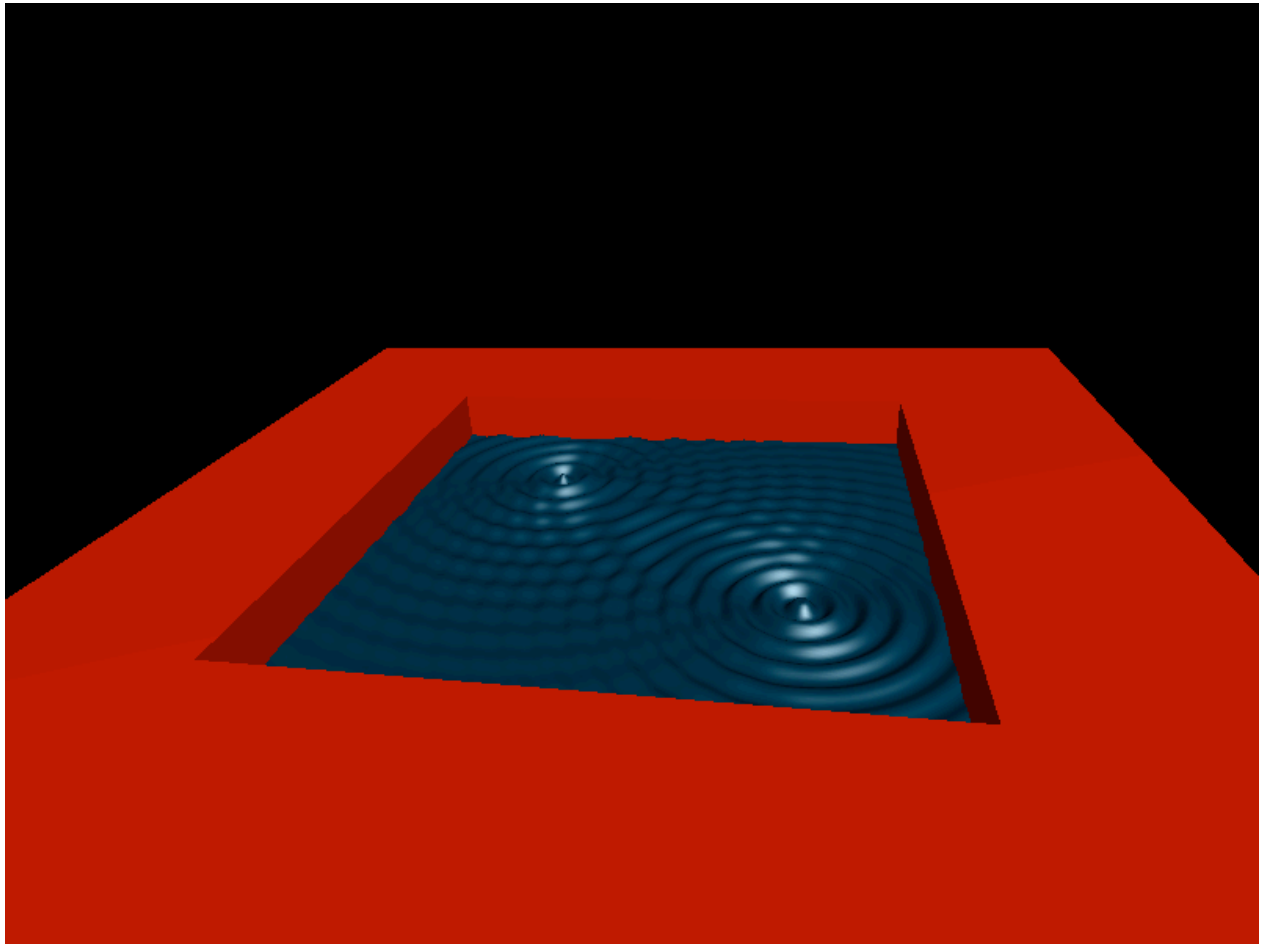


Project Overview

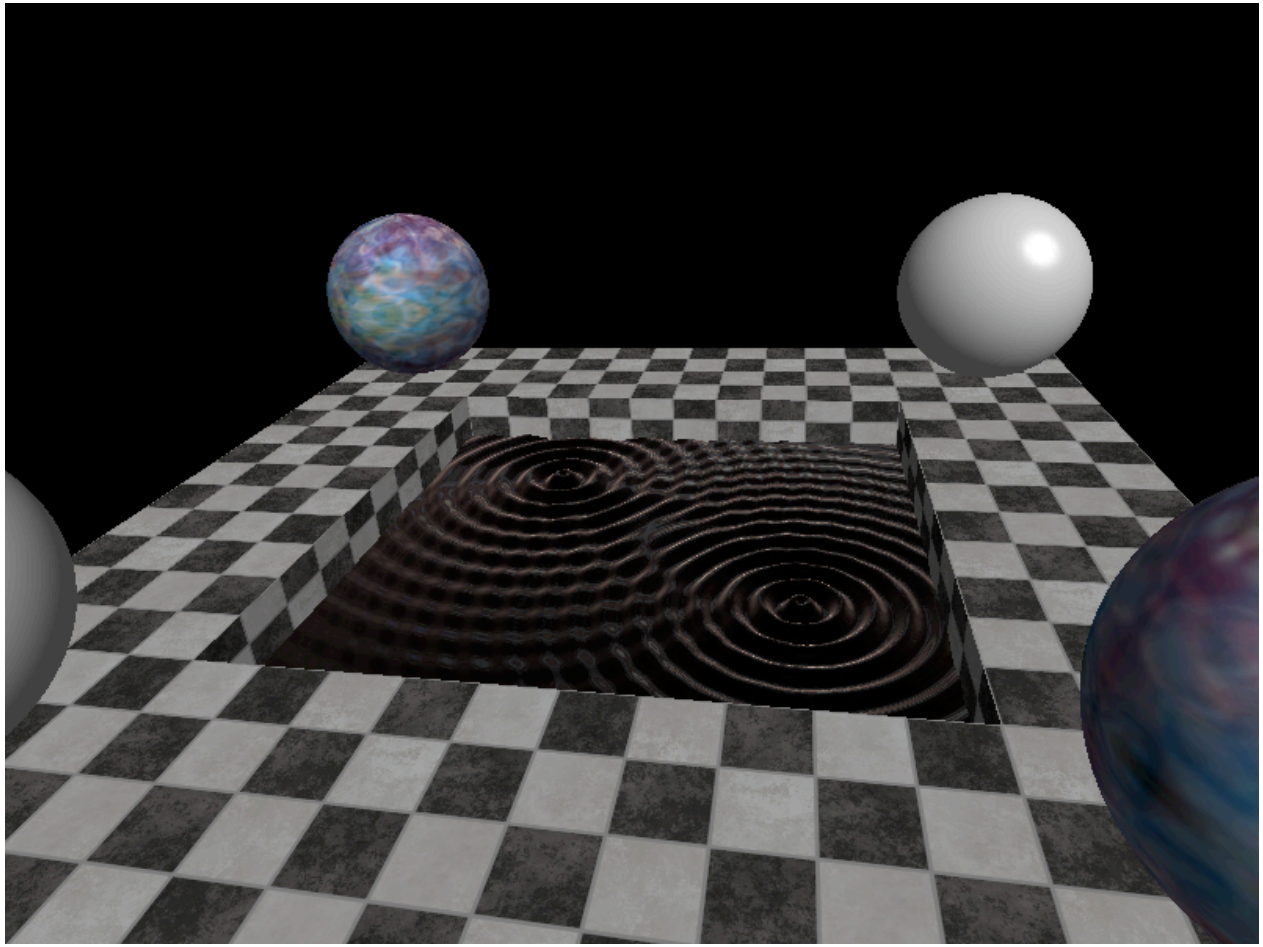
- Project 1: Basic Open GL
  - Render geometries using OpenGL.
  - Starts today!
- Project 2: GPU Programming
  - Change rendering pipeline with shaders to achieve effects such as bump mapping and the Fresnel effect.
- Project 3: Raytracing
  - Use backwards raytracing to render the scene, including relfections and refractions.
- Project 4: Photon Mapping
  - Add photon mapping to the raytracer to achieve effects such as caustics.



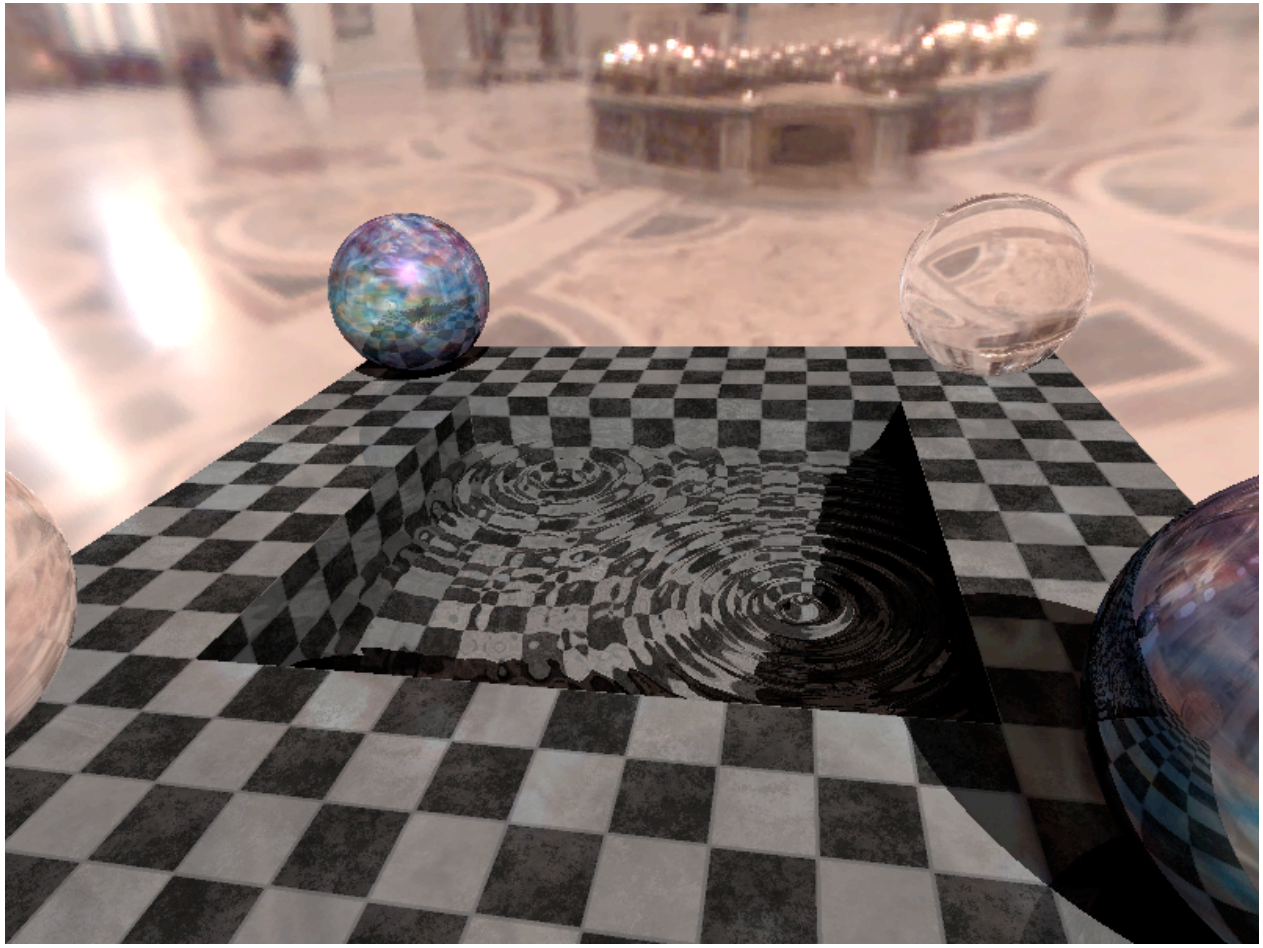
Project 1: Basic OpenGL



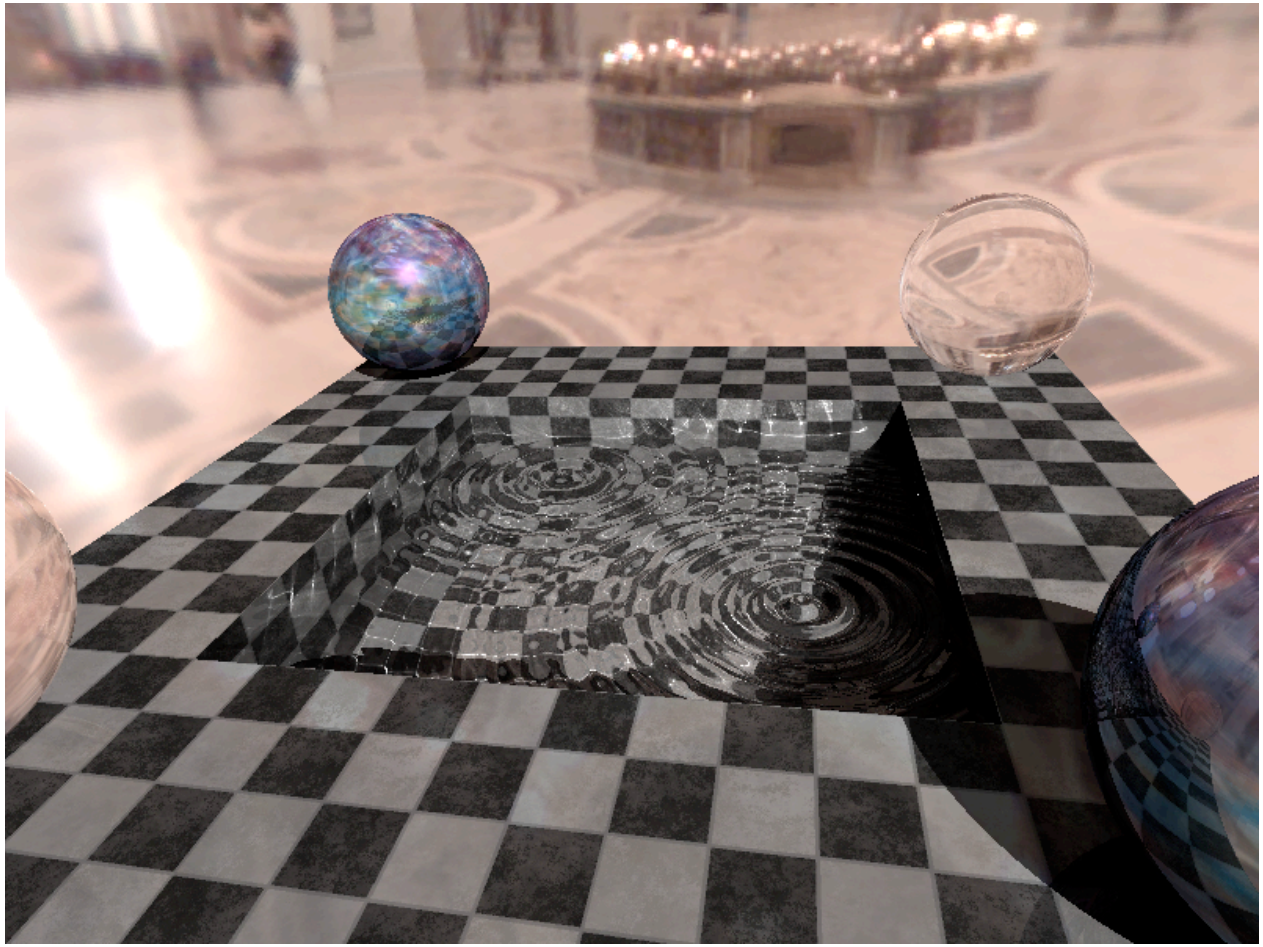
Project 2: GPU Programming



Project 3: Raytracing

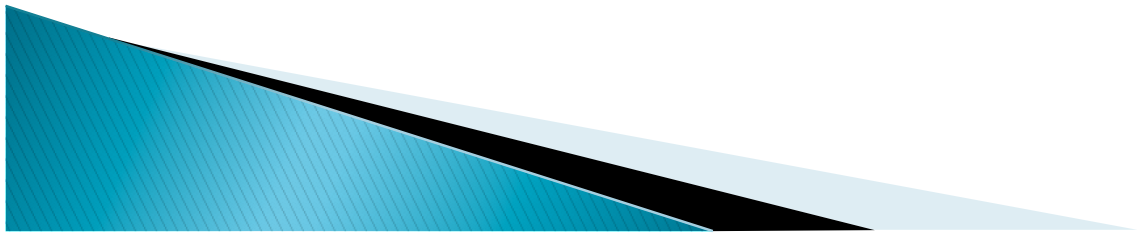


Project 4: Photon Mapping



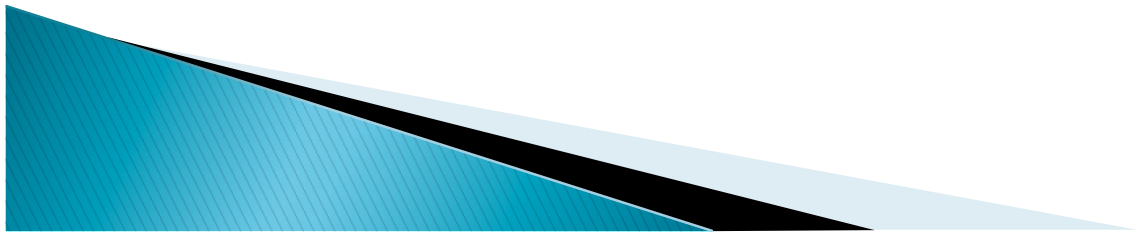
Project 1: Basic OpenGL

- Covers using OpenGL for:
  - Camera transformations
  - Object transformations
  - Blinn-Phong Shading
  - Model Rendering
- Project requirements are to render 3 geometries using OpenGL with basic lighting
  - Sphere, Triangle, Water Surface
  - Geometries are contained in a Scene object which describes where and how to render them.



The Handout Code

- What we give you:
  - Code to open and window using GLUT.
  - Code to move the camera using the input devices.
  - Classes that describe the scene and its geometries, lights, and camera.
  - Math classes such as vectors, quaternions, and matrices.
  - Code to manage application state and GLUT callbacks.
  - Code to load/save PNG images and take screenshots.
- What you give us:
  - Implementation of prj_render in project.cpp
  - Implementation the draw function for each geometry.

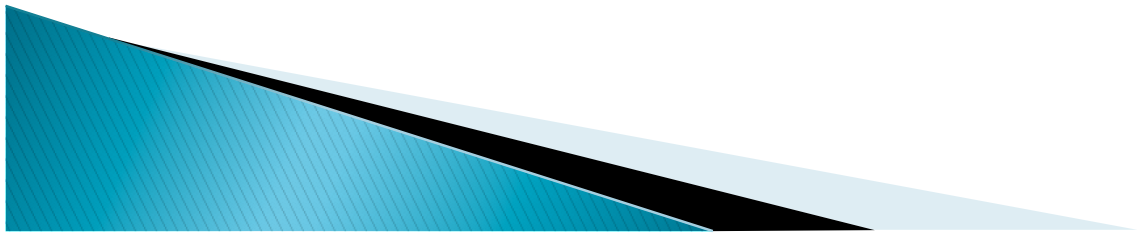


The Scene Class

- All relevant classes defined in scene.h.
  - Scene, Camera, Light, Material, Geometry, UpdatableGeometry
- Scenes contain:
  - A camera
  - A list of geometries
  - A list of lights
  - A list of materials
  - A list of shader effects (ignore until p2)
  - Global information such as ambient light
- Geometries contain:
  - A position, orientation, and scale
  - A material
  - A shader effect (ignore until p2)
  - A virtual draw function (you implement this)

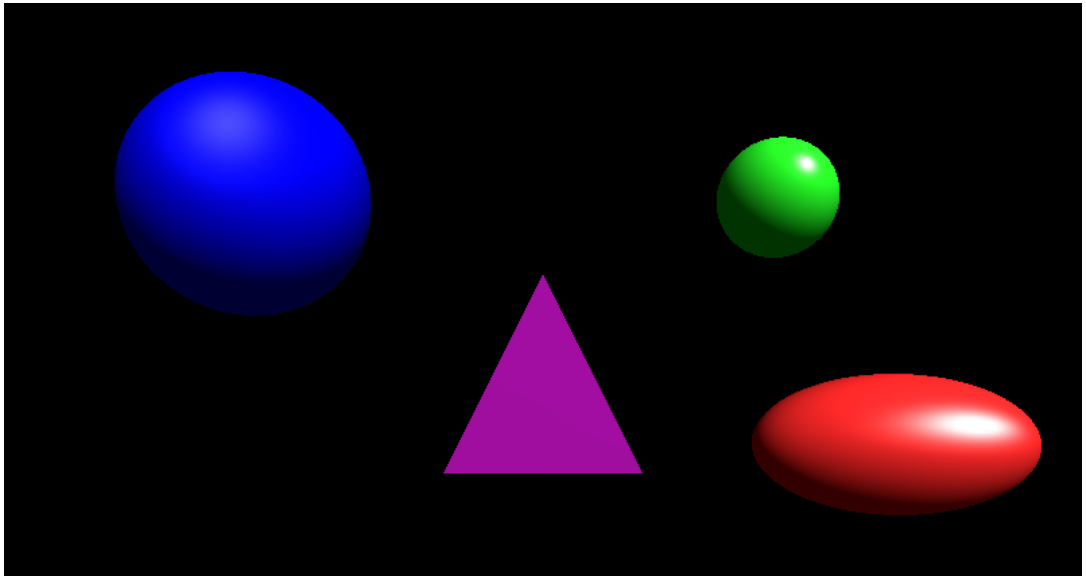
Lighting and Viewing the Scene

- Camera class defines the camera's position and viewing properties.
  - Use properties such as field-of-view and near/far plane to set the projection matrix.
  - Use properties such as the position, direction, and up vectors to set the modelview matrix.
  - The starter code moves the camera for you. All you need to do is use the accessors to set the correct position each frame.
- Material class defines the properties needed to shade an object using Blinn-Phong shading.
  - Ambient
  - Diffuse
  - Phong Specular
  - Phong Shininess
- Light class defines a light's position and color.

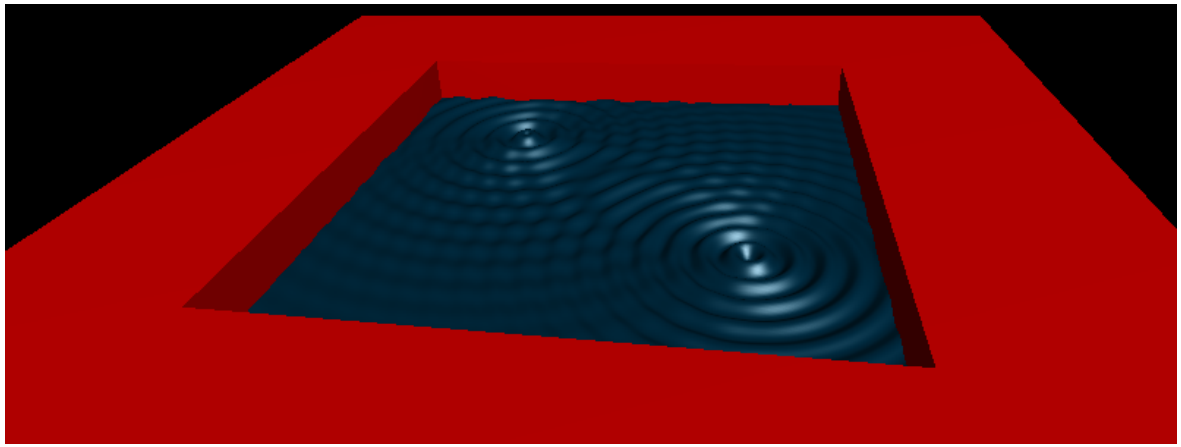


Rendering the Geometries

- Sphere and Triangle
  - Rather straightforward: set the material properties and transformation matrices, then draw the shapes.



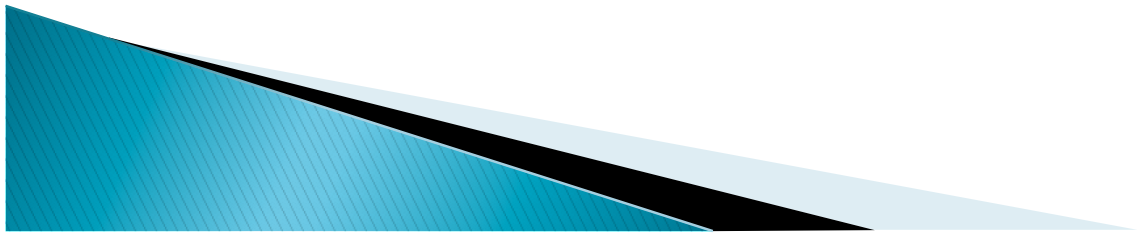
Rendering the Geometries



- Water Surface
  - We give you a function, WaterSurface::get_height, that returns the y-coordinate (height) of the surface at a given x, z, and time, in the local coordinate space.
  - You create a heightmap using that function.
  - Then you create a triangle mesh using that heightmap.
  - Then you compute per-vertex normals for that mesh.
  - Then you render the mesh using OpenGL.

Scene Loading

- There is no actual “scene loader.”
  - TAs aren’t really as good at coding as the professor claims: we never got around to making a file format for scenes.
- You have to manually construct scenes in the code.
  - Yeah, it kind of sucks, but you’ll live.
  - All scene loading goes through the ldr_load_scene function.
  - Each scene has an integer id, you load that scene by using the command line option “./project -s <scene num>”.
- Staff provides several built-in scenes for each project (in staffldr.cpp).
  - For p1, we provide 2 scenes, scene numbers 0 and 1.
- Students are encouraged to make their own as well (in ldr.cpp).



Project Source File Overview

- Files you will likely be editing:
  - project.cpp: main update/render code
  - geom/sphere.cpp, geom/triangle.cpp, geom/watersurface.cpp (and corresponding headers): geometry classes
  - ldr.cpp: scene loader, add your custom scenes
- Files you should definitely have a look at:
  - scene.h: scene-related classes (e.g. Geometry, Camera, Light)
  - project.h: general project header
  - vec/462math.h: general math typedefs and helper functions
  - vec/vec.h: 2d, 3d, 4d vector classes
  - vec/quat.h: quaternion class
  - vec/mat.h: 3d, 4d matrix classes
- Other files:
  - app.cpp: main entry point; GLUT code, application state code
  - gui.cpp: GLUT interface code
  - staffldr.cpp: staff scene loader
  - imageio.h : PNG write/read functions