**Department of Computing**

**CS-361: Computer Graphics  
  
Class: BSCS-12ABC & SE12AB**

**Lab 11: Implementing Photon Mapping**

**CLO 2 -** Apply mathematical and algorithmic principles to implement basic computer graphics techniques, such as line drawing and shading.

**CLO 3-** Develop interactive graphics applications using modern graphics APIs such as OpenGL or DirectX.

**CLO 4 -** Design and implement 2D and 3D graphical solutions for real-world problems.

**Date: 22th April 2025**

**Time: 2:00 PM – 4:50 PM**

# Instructor: Dr. Sidra Sutana

# Lab Engineer: Mr. Aftab Farooq

**Name:** Ahmed Mohiuddin Shah

**CMS ID:** 415216

**Section:** BSCS-12-A

**Lab:** 11

**Lab 11: Implementing Photon Mapping**

**Introduction:**Photon mapping is a two-pass global illumination algorithm that simulates the way light interacts with surfaces in a scene. It is widely used to approximate complex lighting phenomena like caustics, soft shadows, and color bleeding. In this lab, students will implement basic photon mapping and explore how it can be extended to more advanced forms such as differentiable photon mapping.

### **Lab Objective:**

The objective of this lab is to introduce students to photon mapping techniques used in global illumination models. Students will explore basic photon mapping, approximate global illumination, and advanced differentiable photon mapping using generalized path gradients.

### **Tools/Software Requirement:**

* **Operating System:**
  + Windows / macOS / Linux (Ubuntu recommended)
* **Development Environment:**
  + **Windows:** [Code::Blocks](http://www.codeblocks.org/) or [Visual Studio](https://visualstudio.microsoft.com/)
  + **macOS:** [Xcode](https://developer.apple.com/xcode/)
  + **Linux:** GCC and g++ compilers
* **Graphics Libraries:**
  + **OpenGL** (built-in on macOS and Linux, available in Windows IDEs)
  + **GLUT** (OpenGL Utility Toolkit)
  + **GLEW** (OpenGL Extension Wrangler Library)
* **Package Manager (for macOS/Linux):**
  + **Homebrew** (macOS): brew install freeglut glew
  + **APT** (Linux): sudo apt-get install freeglut3-dev glew-utils
* **Compilers:**
  + **Windows:** MinGW (for Code::Blocks) or Microsoft C++ Compiler (for Visual Studio)
  + **macOS/Linux:** GCC/G++

### **Prerequisites:**

* Understanding of basic computer graphics pipeline
* Familiarity with ray tracing concepts
* Knowledge of light transport and rendering equation
* Understanding of automatic differentiation and gradient-based optimization

### **Task Description:**

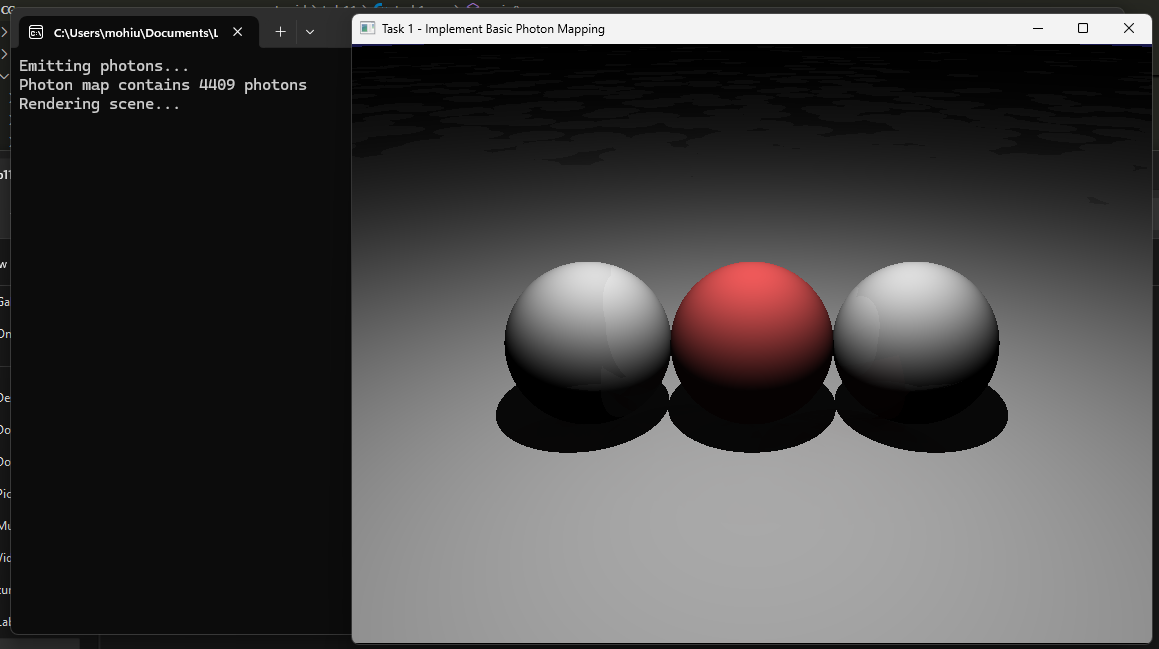
## Task 1: Implement Basic Photon Mapping

1. Write a basic ray tracer capable of:
   * Emitting photons from light sources
   * Storing hit positions in a photon map
   * Estimating radiance at visible points using the photon map
2. Visualize the resulting scene showing caustics or indirect illumination.

**Deliverable:** Submit your code, output images, and a brief report explaining your approach.

The scene consists of simple spheres with diffuse, reflective, and refractive materials. A light source emits photons, which interact with the objects, storing their energy into a photon map. During rendering, the radiance at each pixel is estimated by searching nearby photons within a defined radius. OpenGL is used to display the final image by passing computed color values through shaders. This approach effectively simulates realistic lighting effects like soft shadows, color bleeding, reflections, and caustics.

**Ouptut:**



## 

## Task 2: Approximate Global Illumination

1. Extend the basic implementation to include global illumination approximation:
   * Handle diffuse inter-reflections and soft shadows
   * Tune parameters like photon count, search radius, and density estimation
2. Compare rendering results with and without photon mapping.

**The photon mapping output is more accurate than global illumination method and gives more accurate results.**

**Output**

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

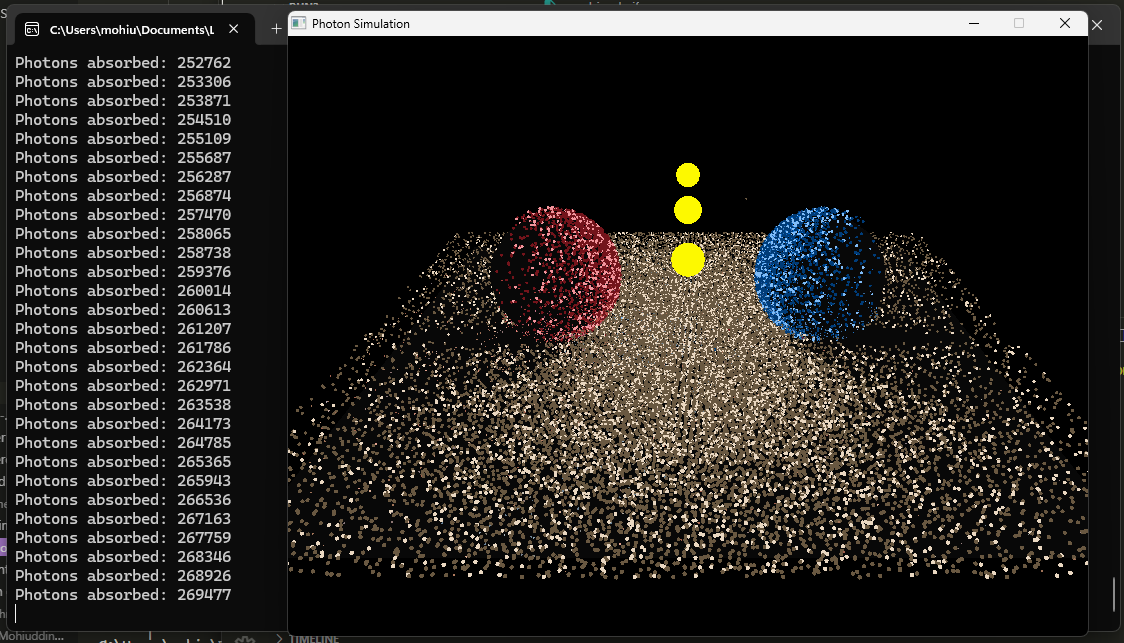
**Deliverable:** Submit side-by-side visual comparisons and an analysis of improvements.

## Task 3: Differentiable Photon Mapping

1. Study the concept of differentiable rendering and its importance in inverse rendering problems.
2. Implement a simplified form of differentiable photon mapping using generalized path gradients.
3. Evaluate the system by optimizing a scene parameter (e.g., light position) to match a reference image.
4. Discuss the challenges and benefits of differentiable photon mapping.

|  |  |
| --- | --- |
| **Differentiable photon mapping = incredibly powerful** — allows optimization, learning, and inverse rendering. |  |
| **But** it's very hard to do properly because of randomness, discrete sampling, and complexity of light paths. |  |
| **Modern approaches** often mix photon mapping with neural networks or "soft" approximations to make it differentiable and stable. |  |

**Output:**



A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer screen

AI-generated content may be incorrect. A screenshot of a computer screen

AI-generated content may be incorrect.

**Deliverable:** Submit source code and test results.

### **Deliverables:**

 Compile a single word document by filling in the solution part and submit this Word file on LMS

 Include screenshots of the program outputs.

 Submit your Lab Word File and code files seperately on submission link.

# Lab Rubrics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lab Rubrics for (Lab-10:Implementing Photon Mapping) | | | | | |
|  | | | | | |
| **Sr.**  **No.** | **Assessment** | **Unacceptable (0 Marks)** | **Does Not Meet Expectations (1/2 Marks)** | **Meets Expectations (3/4 Marks)** | **Exceeds Expectations (5 Marks)** |
| **1** | **Illustrating the basic understanding of semantics and syntax**  **(CLO3, PLO5)** | The student did not submit any work.  OR  The student plagiarized the solution and/or used unfair means. | The student is unable to demonstrate the understanding of syntax of C language and is unable to write an executable code.  The student is not able to understand the structure of a program at all. | The student demonstrates some understanding of syntax of C language and is able to write a code with few errors.  The student is able to understand the structure but still learning the syntax. | The student demonstrates good understanding of syntax of C language and is able to write executable code without help  The student is able to understand the structure and is able to identify problems in the code  when introduced |
| **2** | **Software Tool Usage**  **(CLO4-PLO3)** | The student demonstrates a lack of understanding of tool usage.  Implementation has syntax/semantic/runtime errors, and the student is unable to debug and correct the errors.  The code has inadequate comments and variable names and does not adhere to the coding standards.  No Error handling has been performed.  Documentation is poorly structured. | The student demonstrates some understanding of tool usage.  The codes are correct in terms of their syntax, however, the program output is not always correct in all test cases.  The code has limited comments and inconsistent variable names and may not adhere to the coding standards.  Some Error handling has been performed.  Documentation is adequately structured. | The student demonstrates a good understanding of tool usage.  Furthermore, his/her coding is complete and functional, and the program output is correct in all test cases.  The code has sufficient comments and consistent variable names and reasonably adhere to the coding standards.  Adequate Error handling has been performed.  Documentation is well structured. |