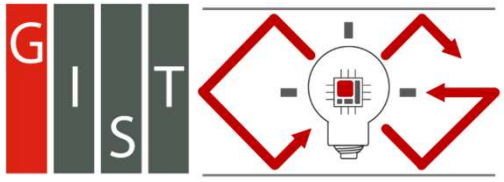
Programming Assignment 4

2026 Computer Graphics



Computer Graphics
Laboratory

2026/05/27

Seongil Kim

Contents

- Notice
  - Submission
  - How to comment commit id
  - Policies
- Assignment Overview
  - Task 1. Naïve Path Tracer
  - Task 2. Path Tracer with Next Event Estimation
  - Task List
- Q & A

Notice

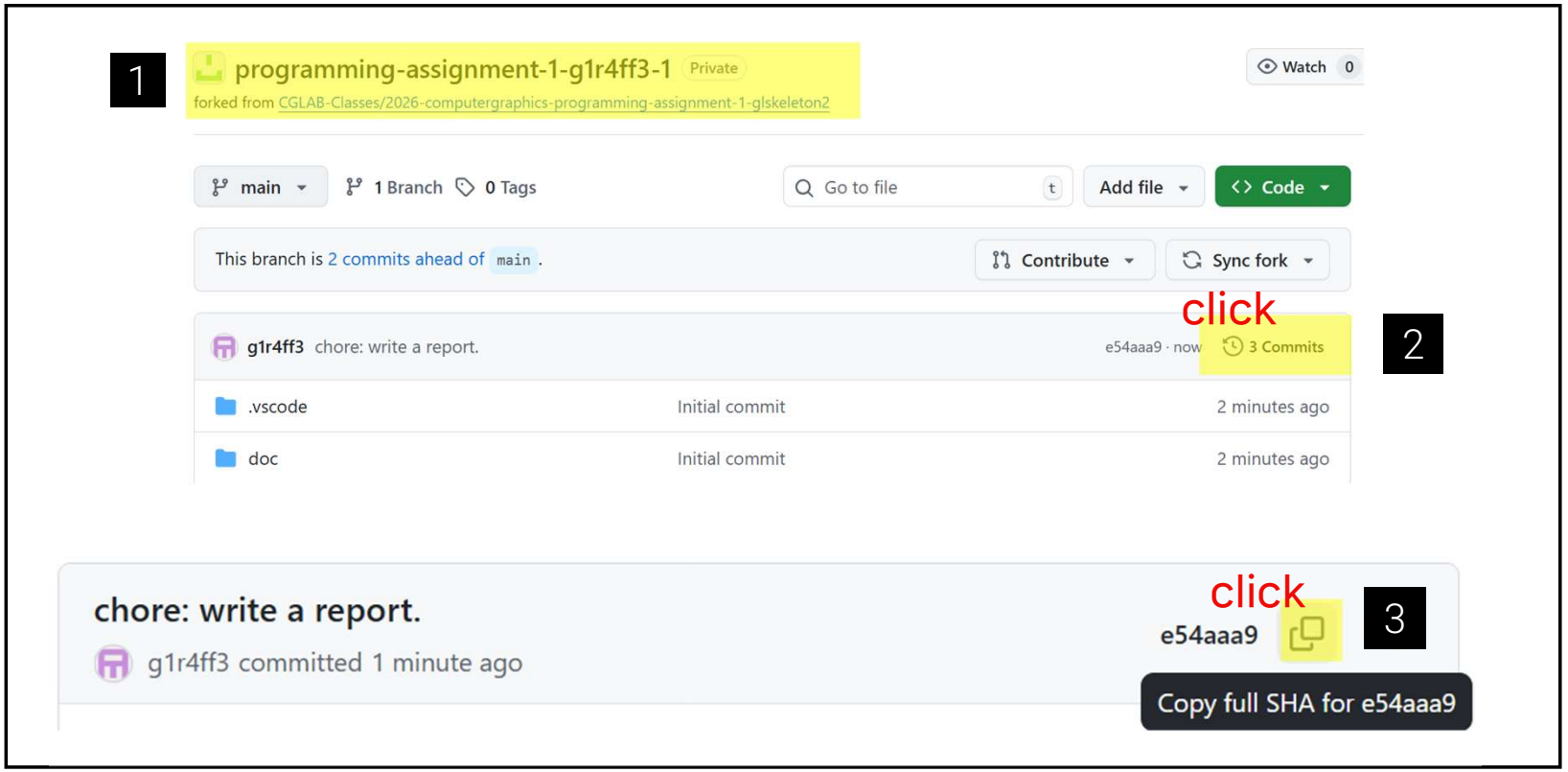
Submission

- Deadline: 23:59:59, Sunday, June 14th, 2026 (KST, +0900)
  - GitHub server clock

- To submit your assignment, two things must be done BEFORE deadline.
  - You must push your commit to your repository
  - You must comment the last commit id (SHA-1 hash) in github issue board
  - You must match username of pushed commits with GitHub username

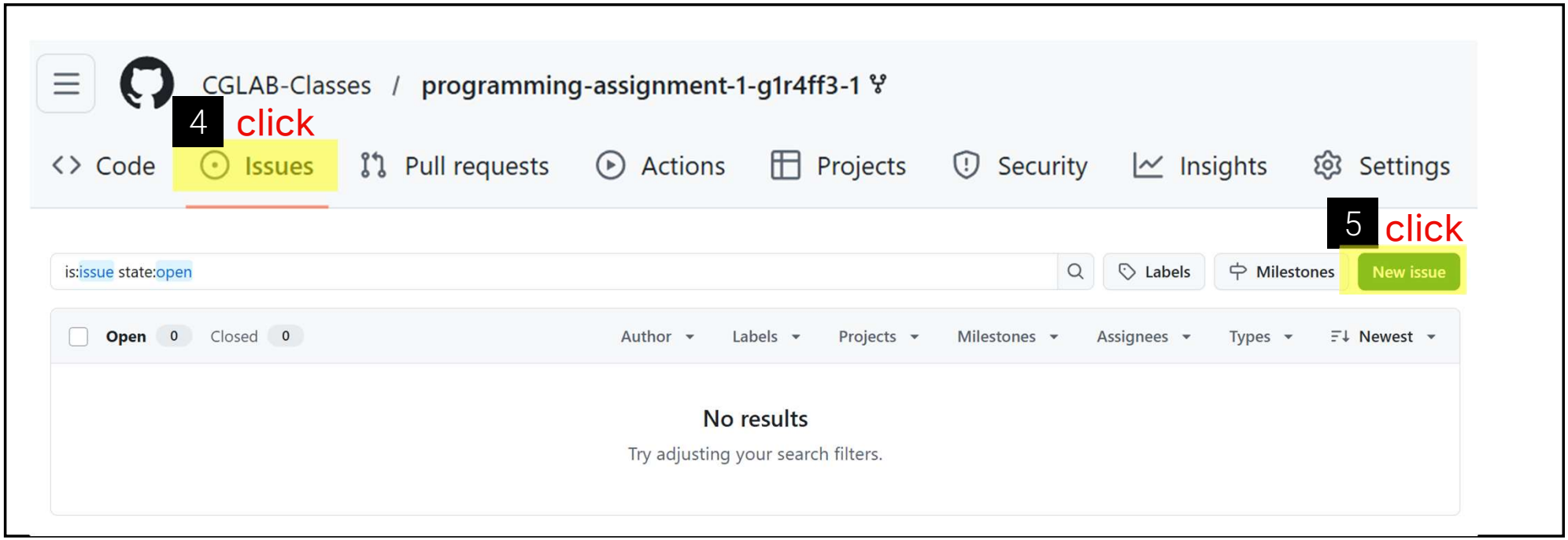
- The last commit in the issue board BEFORE deadline will be considered as submitted assignment.
  - Local timestamp in your commit will be ignored. GitHub server timestamp used instead

How to comment a commit id



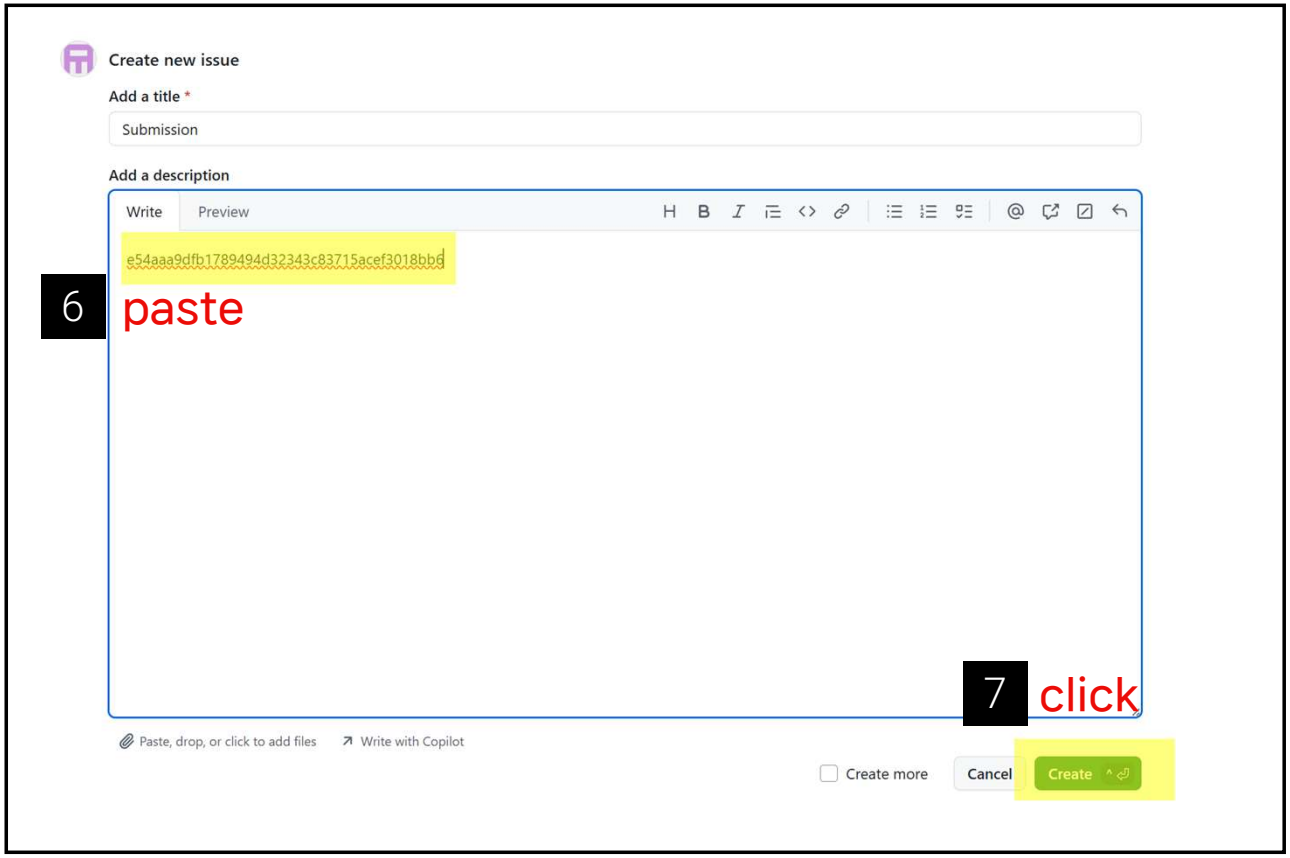
1. Go to your assignment repository
2. Click "Commits"
3. Click copy button of your last commit

How to comment a commit id



4. Go to issues tab
5. Click "New Issue"

How to comment a commit id



6. Paste your latest commit id (ctrl-v)
7. Click "Create"

Policies

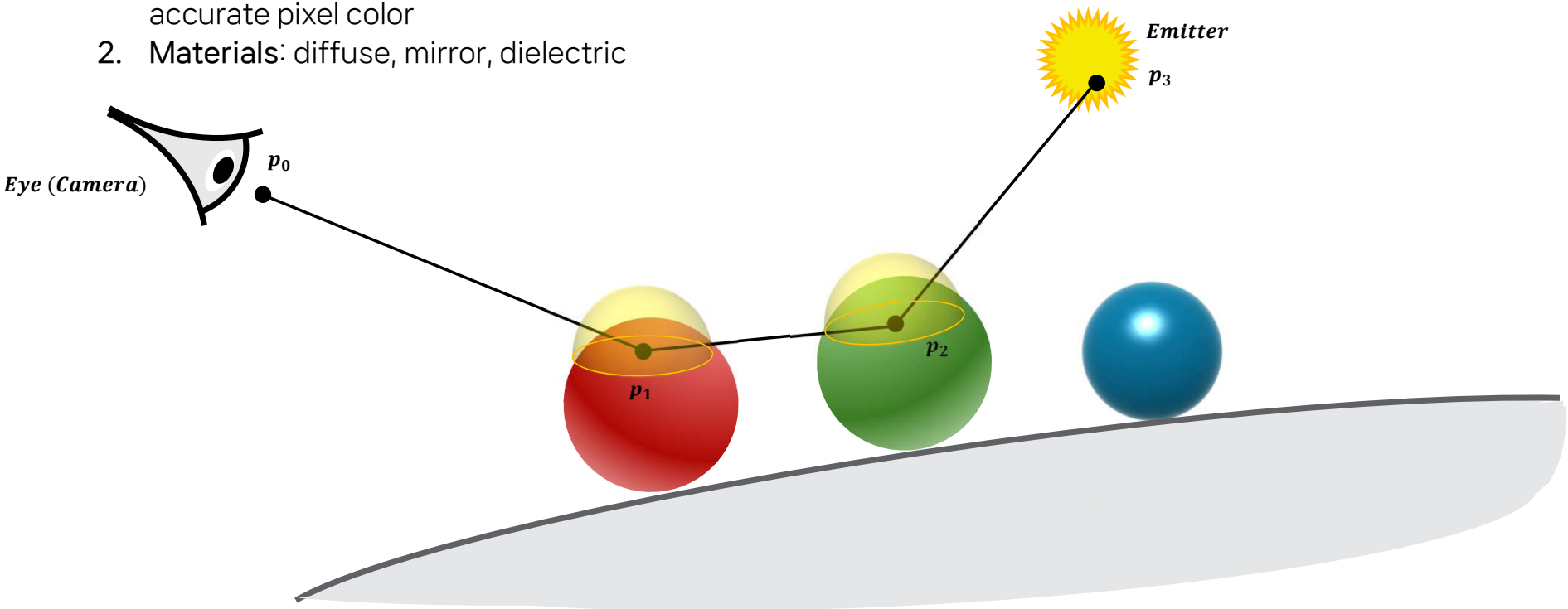
- In following cases, your grade for this PA will be "0"
  - Late submission
    - late push after deadline or late last commit id comment on issue board
  - Build/execution failure
  - Making public of your assignment repository
  - Mismatch of username in a submitted local commit with the submitter's GitHub username
  - API use of a library that gives the solution directly, instead of working your assignment on your own

- Your final grade will be "F"
  - Copy (We will run a copy detector)

Assignment Overview

Task 1. Naïve Path Tracer

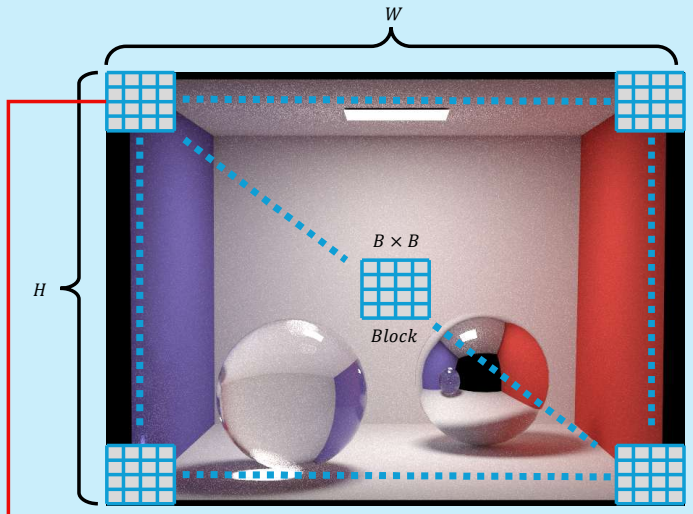
- Path tracer is a ray tracer that constructs "paths" and integrates the rendering equation
  - Refer to the "Global illumination" lecture note in the course page for further details
- Goal: implement a naïve path tracer with only BSDF sampling
- Subtasks
  1. Antialiasing and convergence: more samples per pixel (spp) lead to reduce aliasing and produce a more accurate pixel color
  2. Materials: diffuse, mirror, dielectric



Task 1-1. Antialiasing & Convergence

- Multiple random samples reduce aliasing and lead to convergence of a pixel color
- Goal: implement a block rendering function based on samples per pixel (spp)
  - The rendering framework works in a parallel way by dividing a whole image into multiple blocks

Blockwise rendering



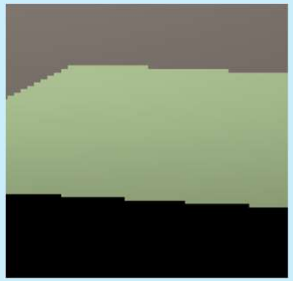

```
static void renderBlock(const Scene *scene, Sampler *sampler, ImageBlock &block)
{
    const Camera *camera = scene->getCamera();
    const Integrator *integrator = scene->getIntegrator();

    Point2i offset = block.getOffset();
    Vector2i size = block.getSize();
    block.clear();

    for (int y = 0; y < size.y(); ++y)
    {
        for (int x = 0; x < size.x(); ++x)
        {
        }
    }
}
```

Worker thread call this function

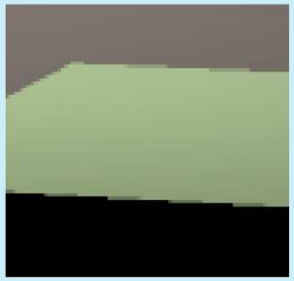
SPP

Antialiasing

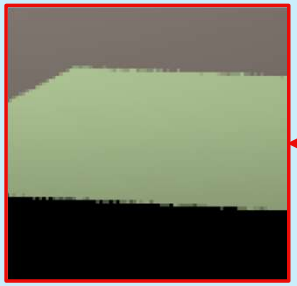
Implement this
(i.e. apply randomness to camera primary rays)



Deterministic sample

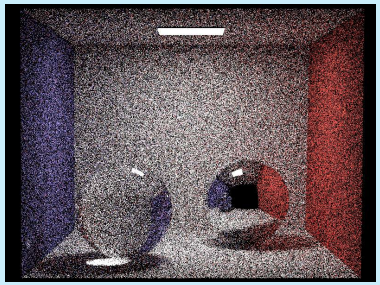


Deterministic Multiple Samples

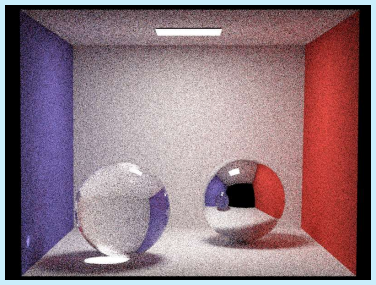


Random Multiple Samples

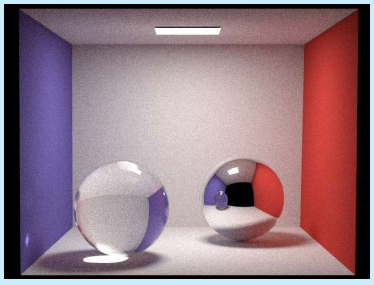
Convergence (rendered by path tracer without NEE)



64



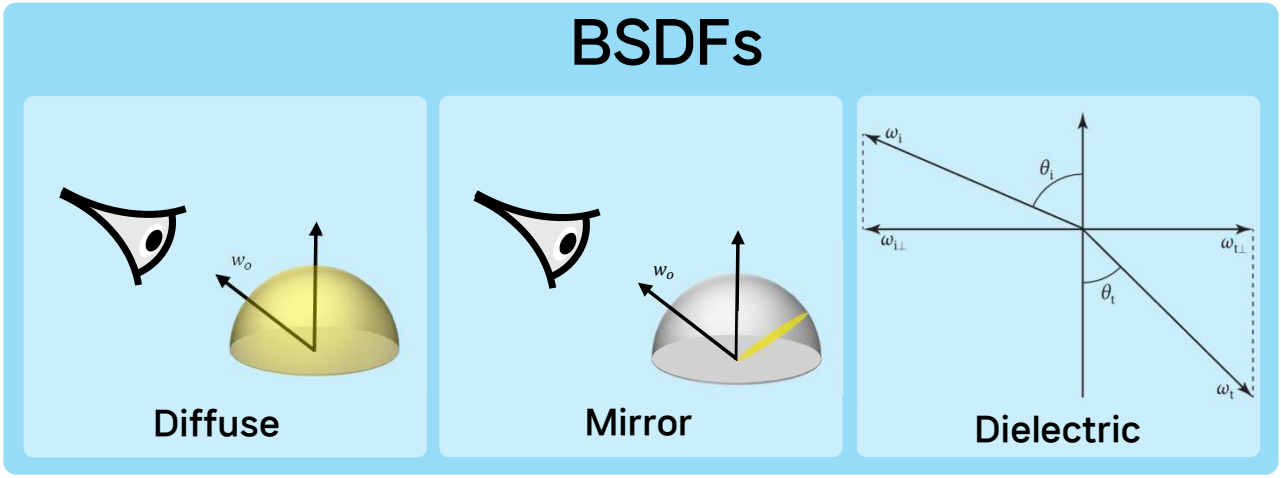
512

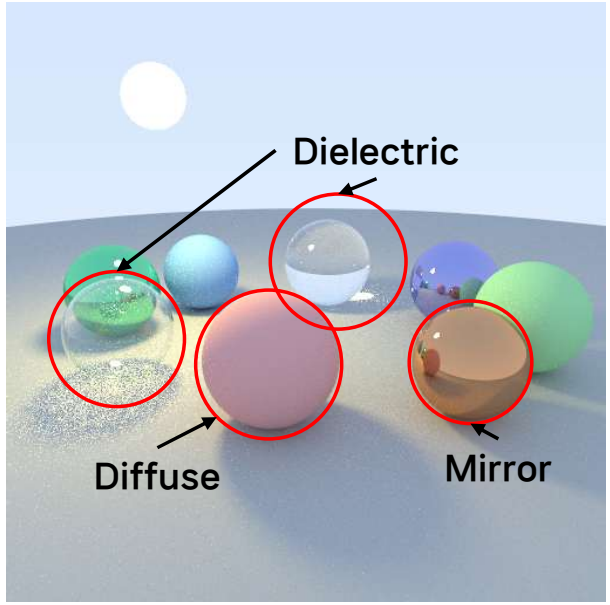


4096

Task 1-2. Materials

- Materials are represented by BSDFs (Bidirectional Scattering Distribution Function)
- Goal: implement diffuse (lambertian surface), mirror (perfect specular reflection), and dielectric (perfect specular transmission) materials
  - Complete **stub methods(sample, eval, pdf)** in those materials
  - At least, **include and use material's elements in test scenes** as member variables
  - **For the dielectric material**, you have to apply Fresnel equation (or Schlick approximation)
  - Refer to Physically Based Rendering: From Theory to Implementation (PBRT) for more details about each material (Reflection Models)

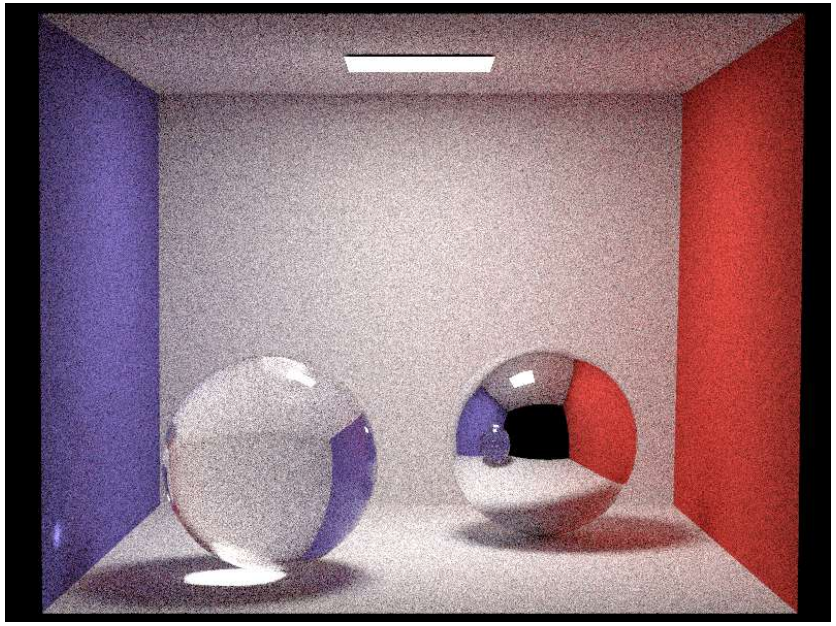




Initial Appearance

- Skeleton code just returns a black image for all provided scenes, but has incomplete framework lacking processing a block of pixels

Task 1 finished

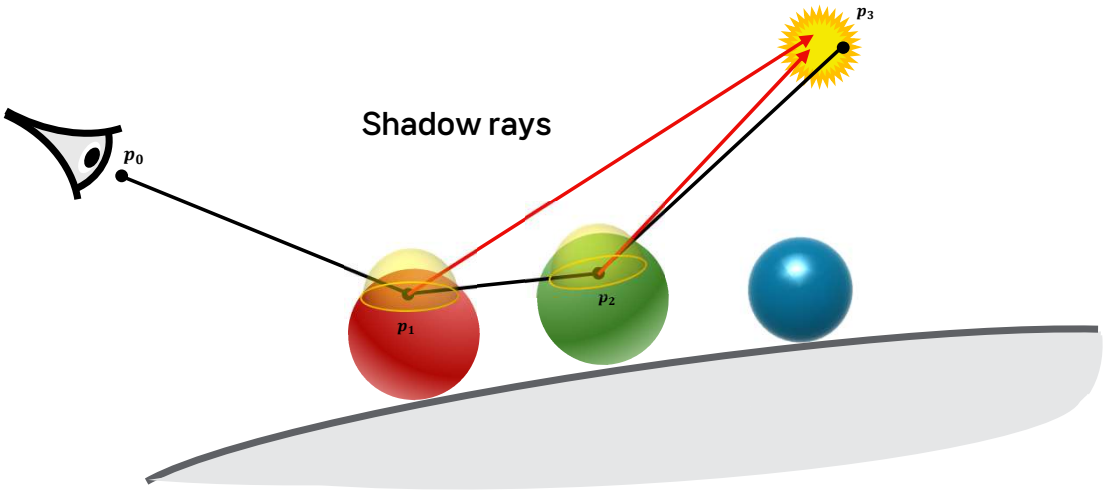


e.g. cbox-sphere

Task 2. Path Tracer with Next Event Estimation

- Too hard to find a light with only BSDF sampling, introducing the emitter (light) sampling will decrease the pixel variance
- Goal: implement the emitter sampling into the L_i function of the rendering equation
  - Consider light contributions from emitters (i.e. direct lighting) well because estimated pixel values could be biased if they are mishandled
  - Multiple emitters must be considered
  - Number of shadow rays is a free parameter, but we recommend to set it 1 due to slow rendering time

Emitter Sampling



The rendering equation

Solid angle measure formulation [1]

$$L(\mathrm{p},\omega_\mathrm{o}) = L_\mathrm{e}(\mathrm{p},\omega_\mathrm{o}) + \int_{\mathrm{S}^2} f\,(\mathrm{p},\omega_\mathrm{o},\omega_\mathrm{i})\,L(t(\mathrm{p},\omega_\mathrm{i}),-\omega_\mathrm{i})\,|\cos\theta_\mathrm{i}|\,\mathrm{d}\omega_\mathrm{i}$$

Area measure formulation [2]

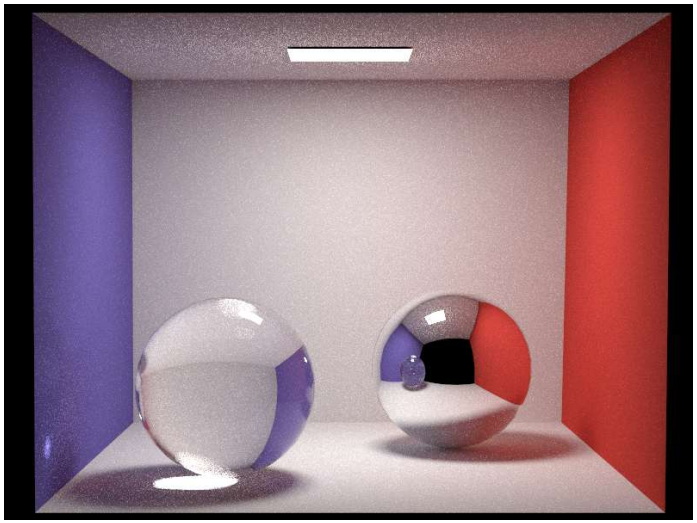
$$P(\bar{\mathrm{p}}_n) = \underbrace{\int_A \int_A \cdots \int_A}_{n-1} L_\mathrm{e}(\mathrm{p}_n \to \mathrm{p}_{n-1}) \times \left(\prod_{i=1}^{n-1} f\,(\mathrm{p}_{i+1} \to \mathrm{p}_i \to \mathrm{p}_{i-1})\,G(\mathrm{p}_{i+1} \leftrightarrow \mathrm{p}_i) \right) \mathrm{d}A(\mathrm{p}_2) \cdots \mathrm{d}A(\mathrm{p}_n).$$

[1] Kajiya, J. T. 1986. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings) 20*, 143–50

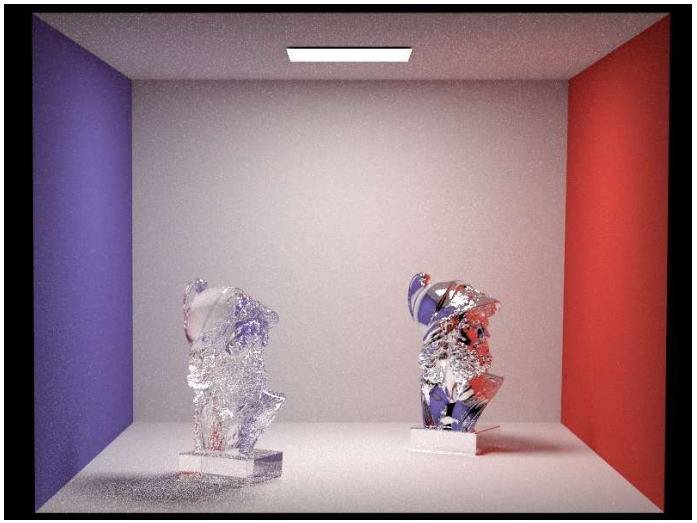
[2] Veach, E. 1997. Robust Monte Carlo methods for light transport simulation. Ph.D. thesis, Stanford University.

Test Scenes

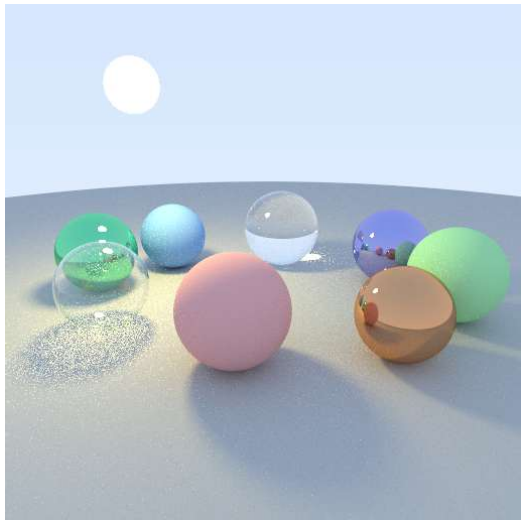
- Three scenes are prepared for your testing
  - cbox-sphere: canonical cornell box scene
  - cbox-ajax **(optional)**: modified cornell box scene in which spheres are replaced with ajax busts
  - bubble: 8 spheres with different materials on the ground
- You can set low spp (e.g. 64) for development because the default spp (1024) requires high computation resources
- You can turn on or off states of nee by specifying nee scene parameter to true or false
  - Don't remove **m_nee** member variable and relevant conditional branches!
- For grading, we will utilize cbox-sphere and bubble scenes, and could try modifying them
  - For example, changing number of emitters, a camera view, objects' material parameters or etc. might be considered



cbox-sphere



cbox-ajax



bubble

Statistical testing

- Special scenes of student's t-test is provided
  - Check your path tracer is well implemented statistically
  - ttest-furance.xml for energy conservation
  - ttest-direct.xml for direct lighting

```
<?xml version="1.0" encoding="utf-8"?>

<!--
  Student's t-test for the `path` integrator on the **furnace test**, a
  classical energy-conservation check.

  A pinhole camera is placed inside a fully enclosed diffuse box whose
  interior emits unit radiance everywhere. By the equilibrium of the
  rendering equation, every interior surface settles to the same outgoing
  radiance L, and at any point

      L = L_e + (ρ/π) ∫ L_i cos θ dω
        = L_e + ρ · L                          (by symmetry, L_i = L)
        ⇒ L = L_e / (1 − ρ).

  Unrolling the recursion gives the geometric series

      L = L_e (1 + ρ + ρ² + ρ³ + … ).

  With `maxDepth = N` set on the integrator the series is truncated after
  N terms, so the path tracer's expected mean is

      L_truncated = L_e · (1 − ρ^N) / (1 − ρ).

  For L_e = 1 and N = 15:
      ρ = 0.5  ⇒  L = 1.99994  (vs. the infinite limit 2.0)
      ρ = 0.8  ⇒  L = 4.82408  (vs. the infinite limit 5.0)

  Failures here usually point at one of: incorrect cosine-weighted
  hemisphere sampling, missing throughput accumulation across bounces,
  or double-counting emission at non-camera vertices when NEE is enabled.
-->

<test type="ttest">
    <string name="references" value="1.99994, 4.82408"/>

    <scene>
        <color name="sky_top" value="0, 0, 0"/>
        <color name="sky_horizon" value="0, 0, 0"/>
        <integrator type="path">
            <integer name="maxDepth" value="15"/>
        </integrator>
        <camera type="perspective">
            <float name="fov" value="10"/>
            <integer name="width" value="1"/>
            <integer name="height" value="1"/>
        </camera>
        <mesh type="obj">
            <string name="filename" value="../meshes/furnace.obj"/>
            <bsdf type="diffuse"><color name="albedo" value="0.5, 0.5, 0.5"/></bsdf>
            <emitter type="area"><color name="radiance" value="1, 1, 1"/></emitter>
        </mesh>
    </scene>
```

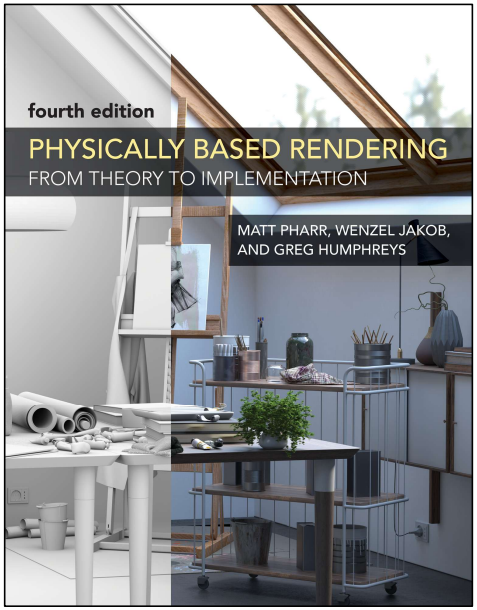
ttest-furance.xml

```
<?xml version="1.0" encoding="utf-8"?>

<!--
  Student's t-test for the `path` integrator on five direct-lighting
  configurations. Each <scene> places a 1×1 pinhole camera looking down
  at a diffuse floor with one differently-shaped triangular area light
  hovering above it; the integrator's mean radiance estimate per scene
  should match the precomputed reference (analytical / heavily-converged)
  to within statistical noise.

  All five tests should pass when next event estimation is implemented
  correctly. Failures here usually point at one of: solid-angle/area-pdf
  conversion, surface-normal orientation on the emitter, or the visibility
  test in the shadow ray.
-->

<test type="ttest">
    <string name="references"
        value="0.0898394, 0.02292, 0.0534198, 0.0205314, 0.26174"/>

    <scene>
        <color name="sky_top" value="0, 0, 0"/>
        <color name="sky_horizon" value="0, 0, 0"/>
        <integrator type="path">
            <integer name="maxDepth" value="15"/>
        </integrator>
        <camera type="perspective">
            <transform name="toWorld">
                <lookat origin="0, 0.01, 0" target="0, 0, 0" up="0, 0, 1"/>
            </transform>
            <float name="fov" value="1e-6"/>
            <integer name="width" value="1"/>
            <integer name="height" value="1"/>
        </camera>
        <mesh type="obj">
            <string name="filename" value="../meshes/floor.obj"/>
            <bsdf type="diffuse"><color name="albedo" value="0.5, 0.5, 0.5"/></bsdf>
        </mesh>
        <mesh type="obj">
            <string name="filename" value="../meshes/polylum1.obj"/>
            <bsdf type="diffuse"><color name="albedo" value="0, 0, 0"/></bsdf>
            <emitter type="area"><color name="radiance" value="1, 1, 1"/></emitter>
        </mesh>
    </scene>
```

ttest-direct.xml

Additional Resource

- If you are having difficulty with this programming assignment, please check the following material as well
- Physically Based Rendering: From Theory to Implementation (3rd/4th edition)
  - Book: there are a lot of books in GIST library
  - E-Book: https://www.pbr-book.org/
  - Code: https://github.com/mmp/pbrt-v3 (v3 is recommended)



Task List

Accept your PA4!
https://classroom.github.com/a/7uKEyNEz

1. Implement a naïve path tracer with only BSDF sampling [15 Points]
   - Use the cbox-sphere test scene to prepare result images for the report [3 Points]
     - At least, set the maximum depth as 8 [1 Points]
     - Port the ray-sphere intersection function from PA3 to this assignment [1 Points]
     - Turn off nee by changing the **nee** scene parameter to **false** for your result images [1 Points]
   - Show antialiasing and convergence as number of samples increases [2 Points]
   - Implement material classes (diffuse, mirror, dielectric) [3 Points]
   - Implement the integration function L_i with only BSDF sampling in the ***nee-off*** branch [7 points]
2. Implement the emitter sampling into the L_i function of the rendering equation [10 Points]
   - Use the bubble test scene to prepare result images for the report [2 Points]
     - At least, set the maximum depth as 8 [1 Points]
     - Turn on nee by changing the **nee** scene parameter to **true** for your result images [1 Points]
   - Introduce the emitter sampling into the L_i in the ***nee-on*** branch[5 Points]
   - Multiple emitters are considered [3 Points]
3. Write a report [10 Points]
   - Describe your submission into a final report concretely; anything is fine, but it should be meaningful and persuasive [10 Points]
     - Prepare minimum materials in the report for your arguments [8 points]
       - **Include rendered images** of test scenes [3 points]
       - **Describe how you implemented** each component of the function for block rendering, materials, L_i function, emitter sampling, multiple emitters, and the rest [5 Points]
     - Comply the following rules [2 points]
       - Write your name, student id, GitHub username
       - Place the report in **docs** directory
       - Admissible file format: **markdown (md), pdf**
4. Challenge [Optional: Not graded]
   - Apply the acceleration data structure to your path tracer

Q & A

- Email
  - Seongil Kim: sngillkym@gm.gist.ac.kr
  - Hyunjin Jung: hjjung0810@gm.gist.ac.kr
- Office: 104 Dasan Bldg