Programming Assignment 4

2022 Computer Graphics

Computer Graphics
Laboratory

Submission

- Deadline: 23:59:59, June 14th , 2022 (KST, +0900)
  - Github server clock

- To submit your assignment, you **must** do two things. **Both of them must be done BEFORE deadline.**

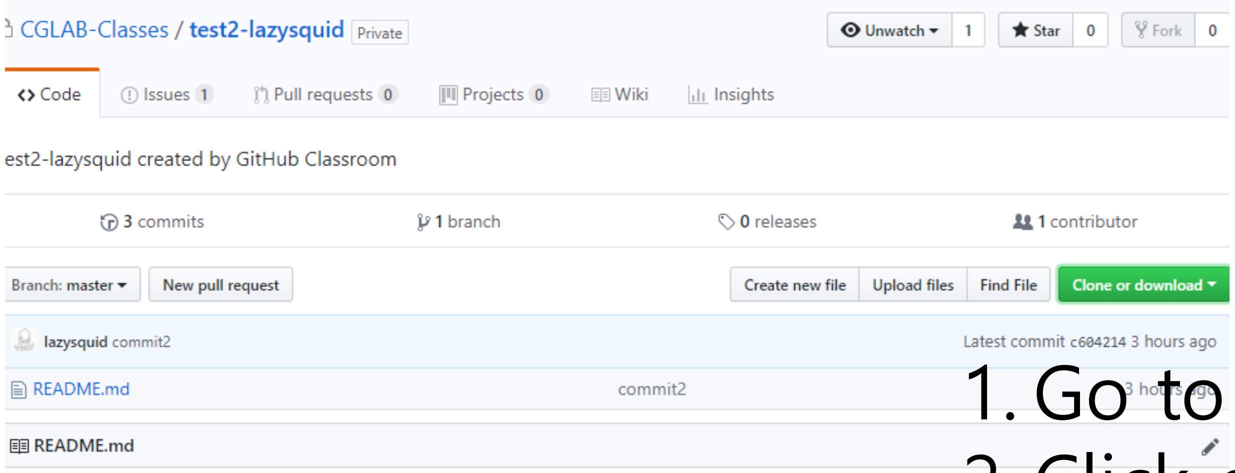
1. You should push your commit to your assignment repo before deadline.

   -Obviously, e- mail submission is not accepted

2. You should comment the last commit (before deadline) id (SHA-1 hash) in github issue board. (See next slide)

- The last commit BEFORE dead line will be considered as submitted assignment.
  - Github server will track this for me.
  - Timestamp in your commit (local time) will be igrnoed. (I will use github server timestamp instead)

Commenting Commit ID 1/2



CGLAB-Classes / test2-lazysquid Private

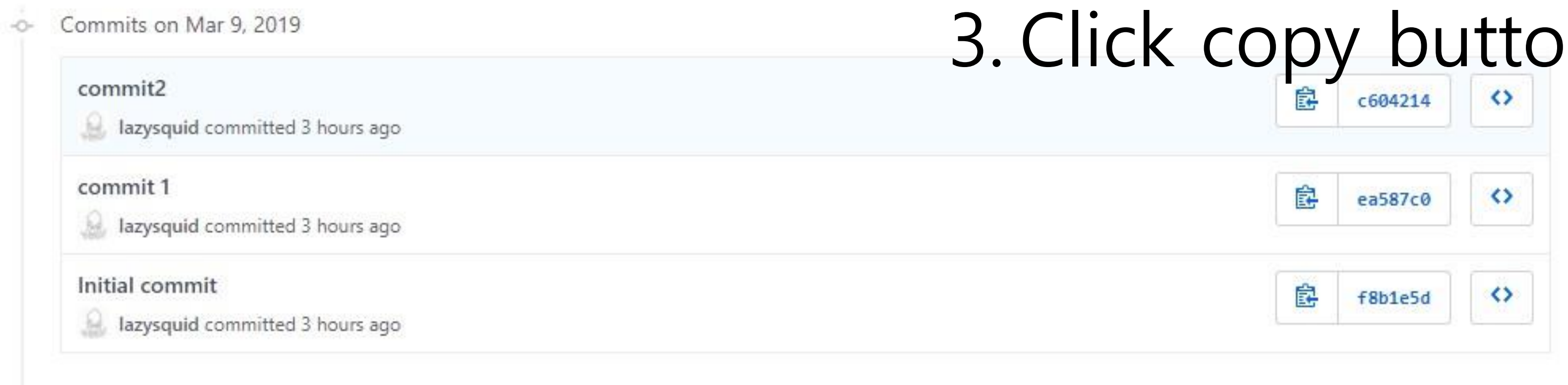
est2-lazysquid created by GitHub Classroom

3 commits | 1 branch | 0 releases | 1 contributor

lazysquid commit2 Latest commit c604214 3 hours ago

README.md commit2 3 hours ago

README.md



Commits on Mar 9, 2019

commit2
lazysquid committed 3 hours ago c604214

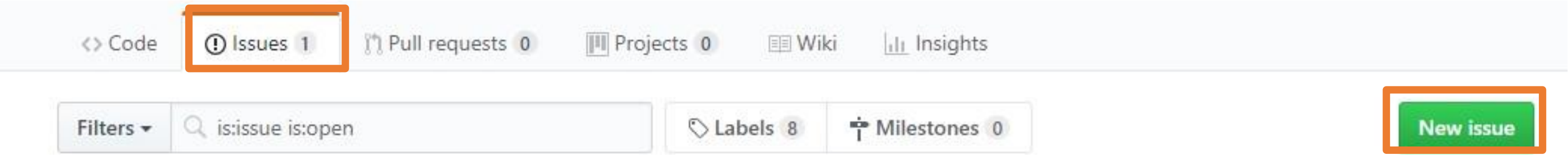
commit 1
lazysquid committed 3 hours ago ea587c0

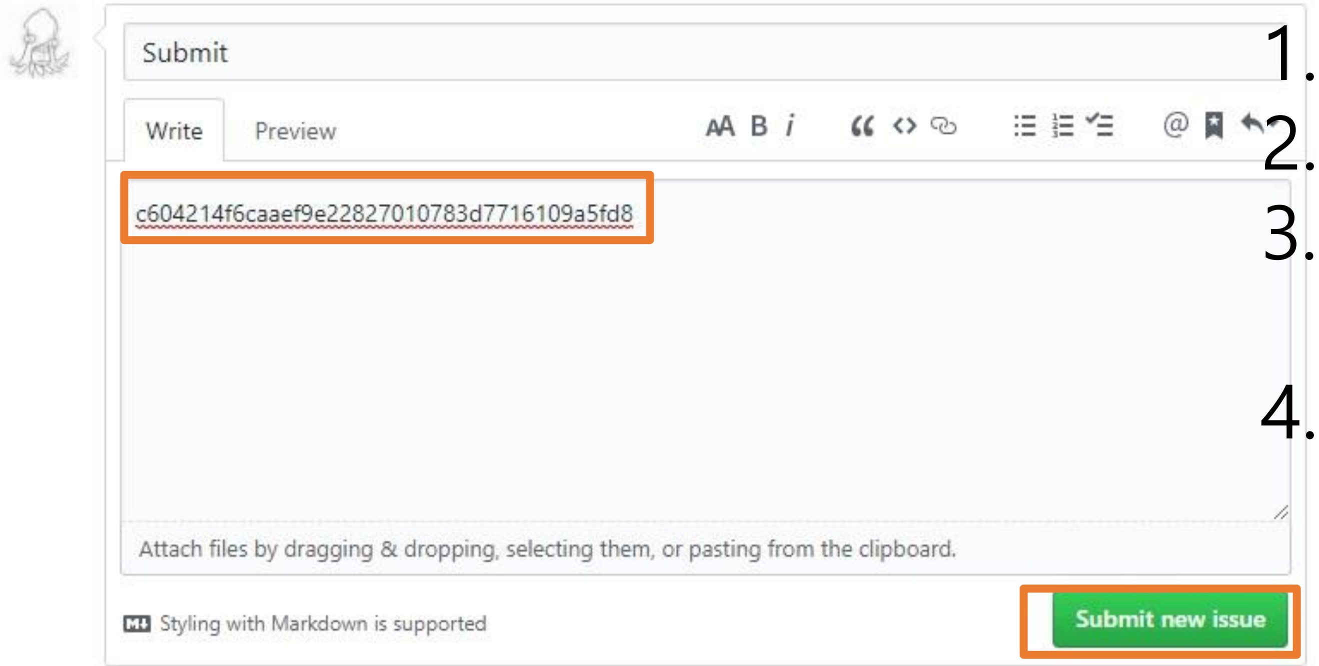
Initial commit
lazysquid committed 3 hours ago f8b1e5d

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

G I S T Computer Graphics Laboratory

Commenting Commit ID 2/2




Submit

c604214f6caaef9e22827010783d7716109a5fd8

Write
Preview
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Submit new issue

1. Go to issue tab
2. Click "new issue"
3. Paste your lastest commit id (Ctrl-v)
4. Click "submit new isse"

Policy

- In the following cases, your grade for this PA will be 0
  - Late submission (Late push before deadline or Late last commit id comment on issue board)
  - Build/execution failure
  - Making public of your assignment repository
  - If you tried to push your commit with force option(Tried to change history of remote server)

- Your final grade will be “F”
  - Copy

Task List

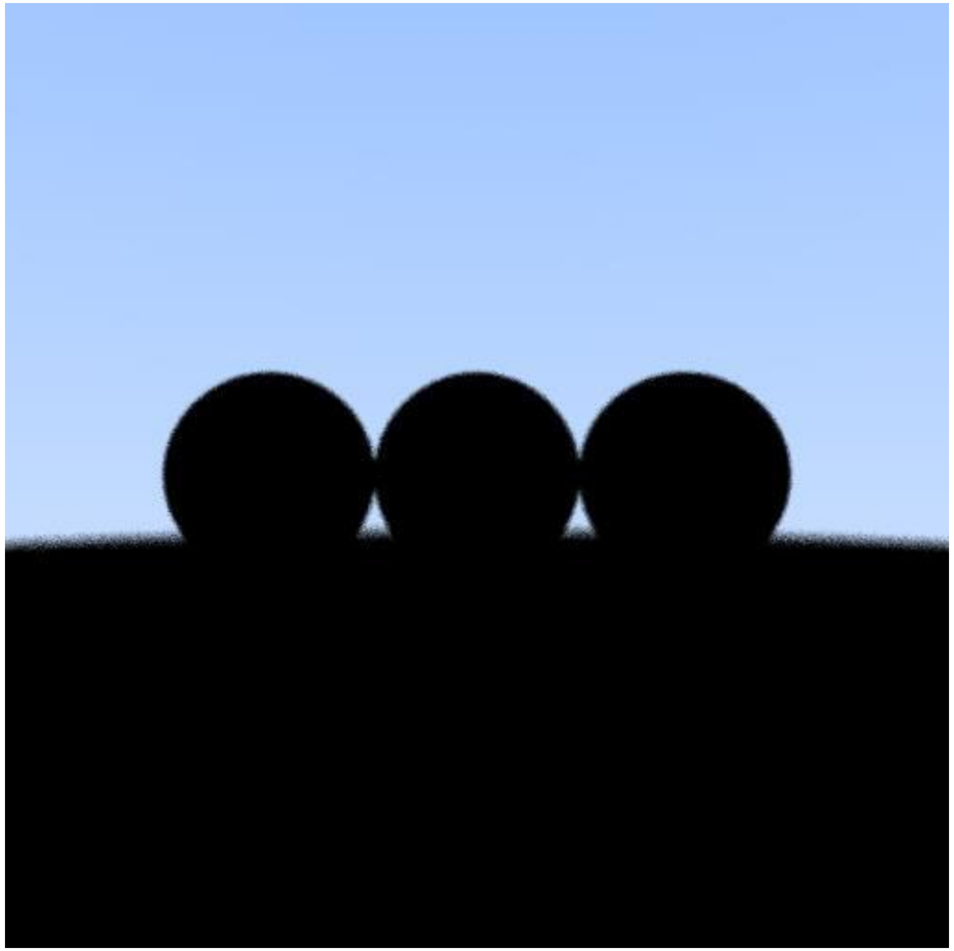
1. Materials (12 Points)
   - Lambertian, Metal, Dielectric, Area light (Emissive material)
   - Implement **scatter** function in each material class
2. Antialiasing (6 Points)
3. Indirect lighting (6 Points)
   - Multiple bounces, depth > 10
4. Direct light sampling (6 Points)
5. Report (10 Points)
   - For this time, you need to write your report in detail.
   - Add teaser image whenever you add new features (e.g. complete your task) and explain about it

Initial Appearance

- Skeleton code: Neon renderer

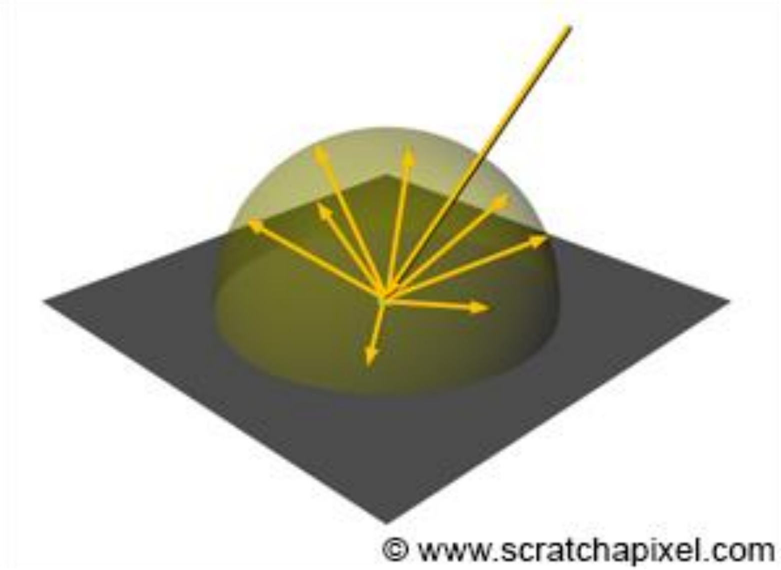
- Unlike OpenGL project,
  the result will be png file.

- output: *.png

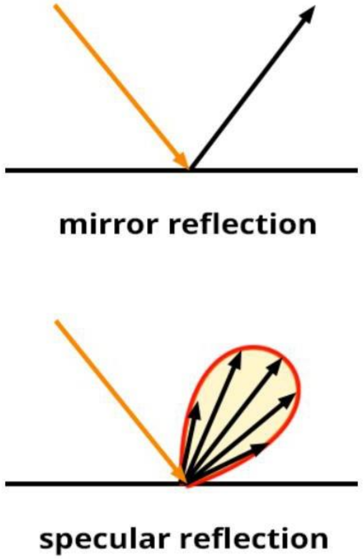


Materials

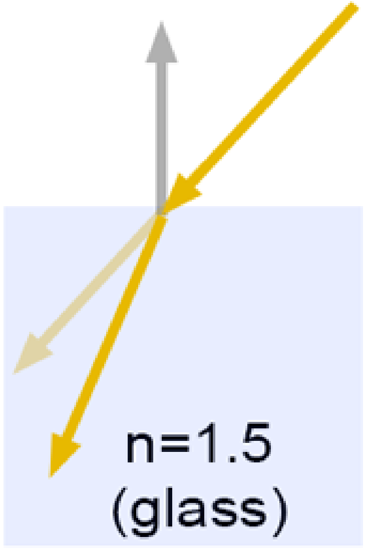
See **scatter** method in each material class



Lambertian
(diffuse)



Metal
(mirror with randomness)

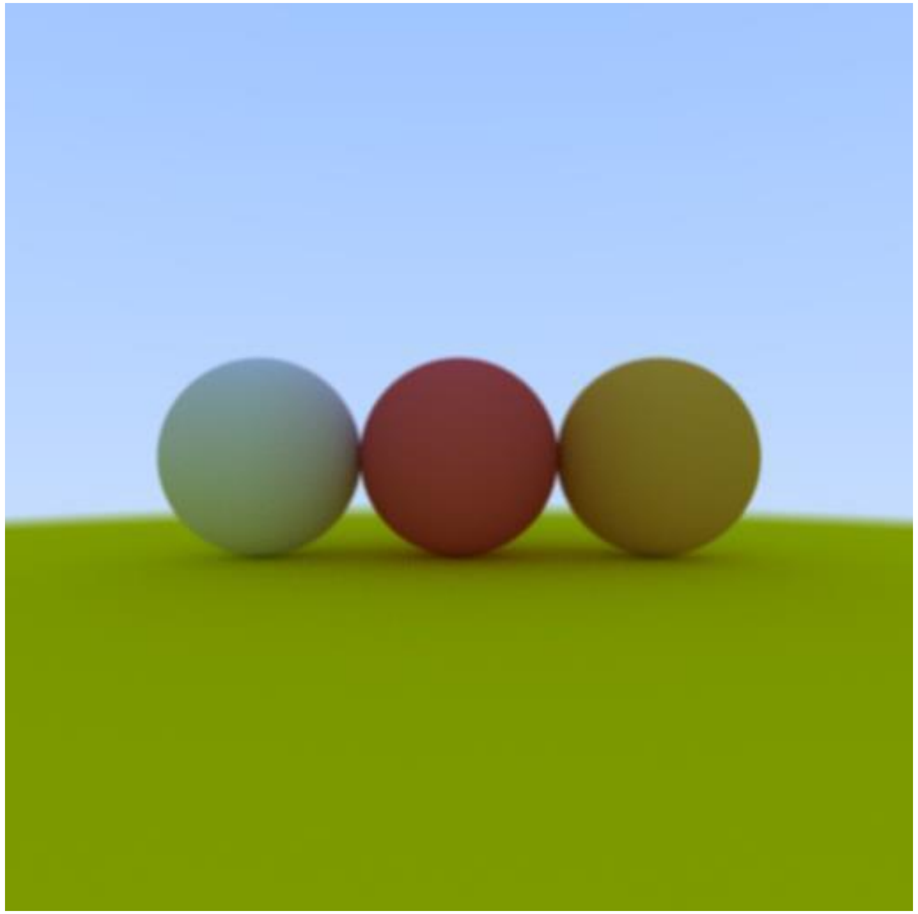


Dielectric

Computer Graphics Laboratory

https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading

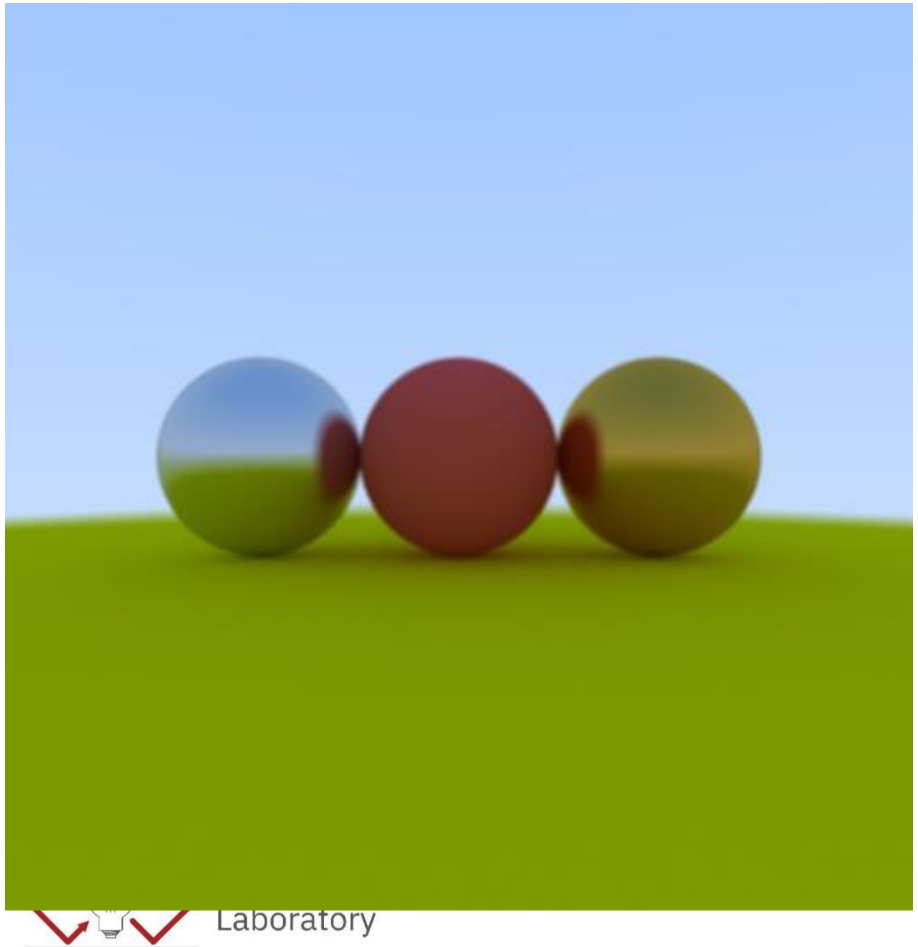
Materials

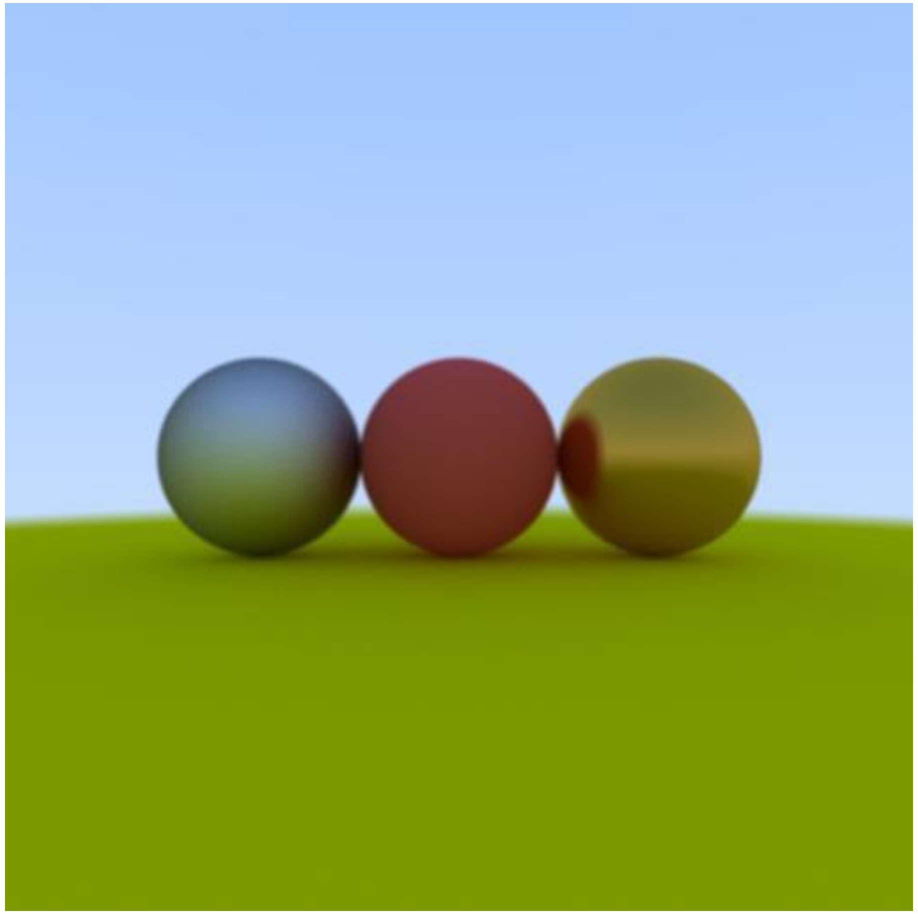


Lambertian (diffuse)

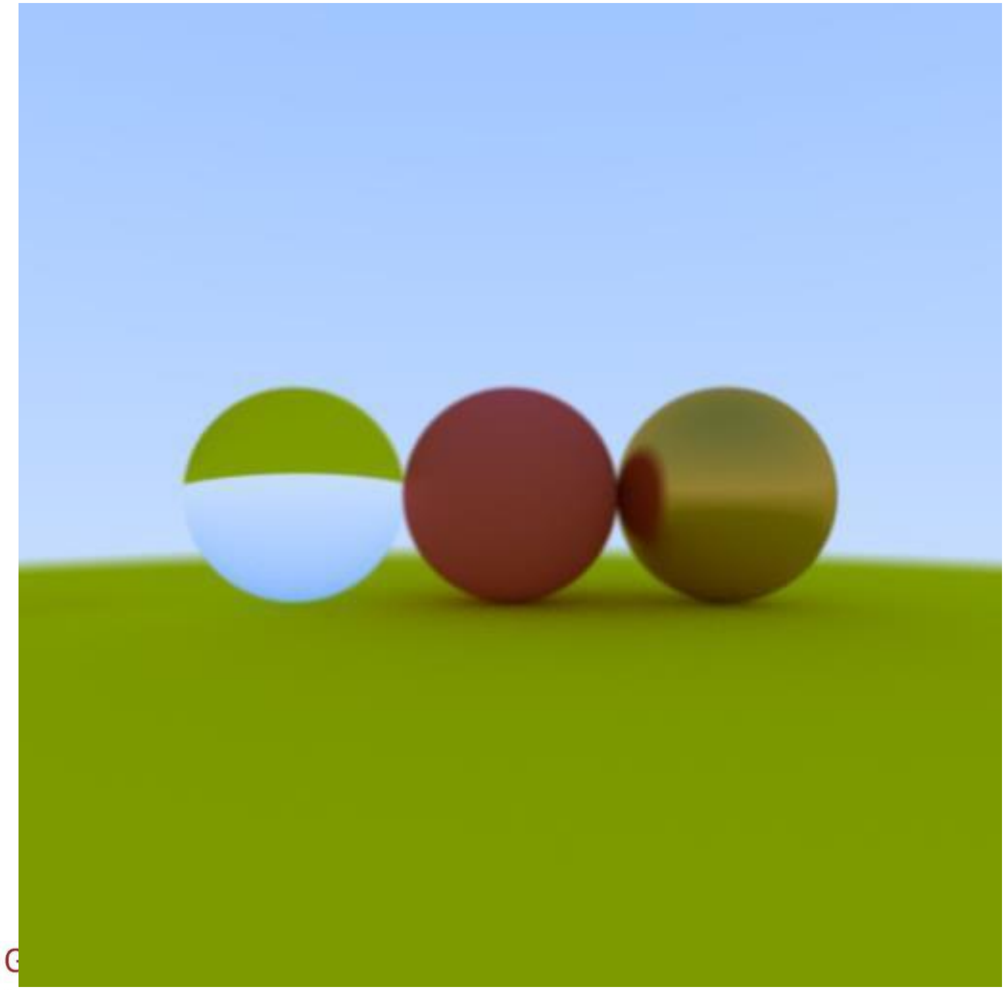
Materials

- Perfect mirror vs metal (mirror with randomness)





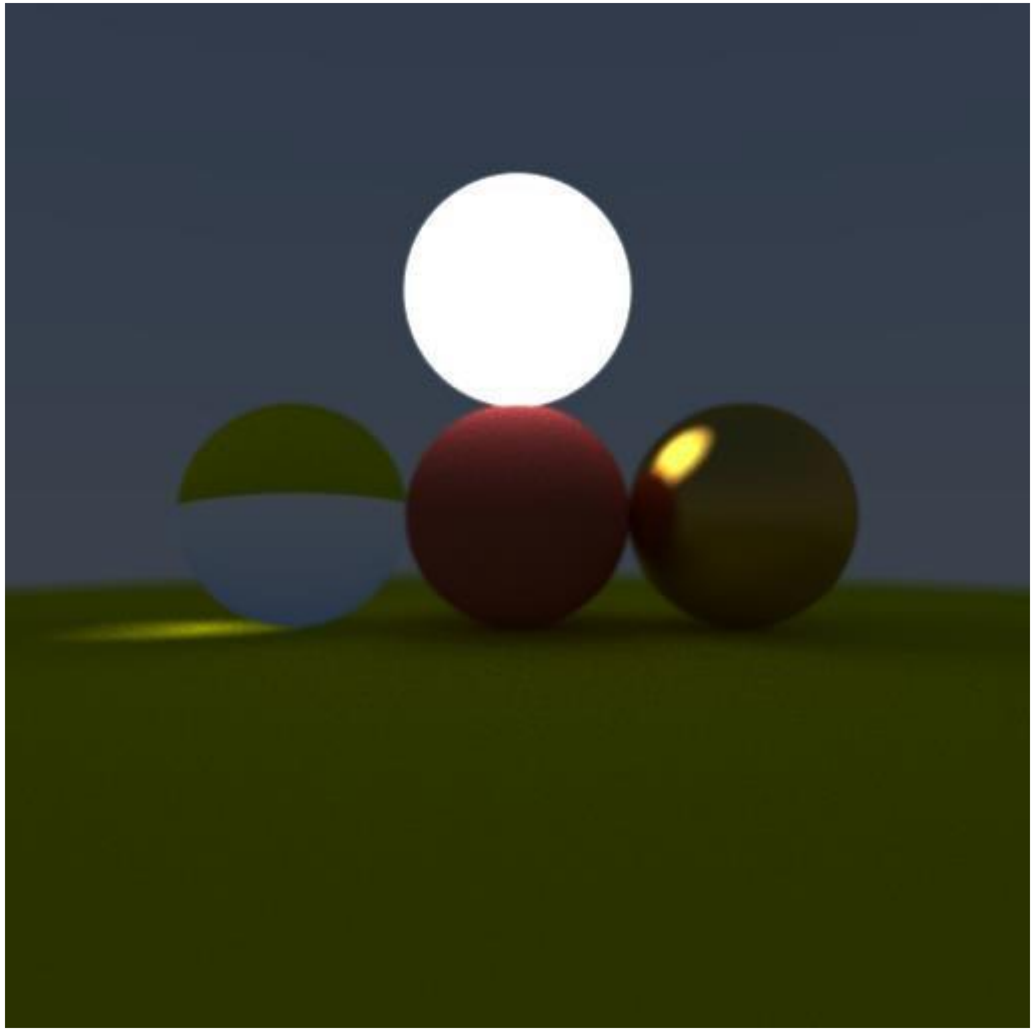
Materials



dielectric material

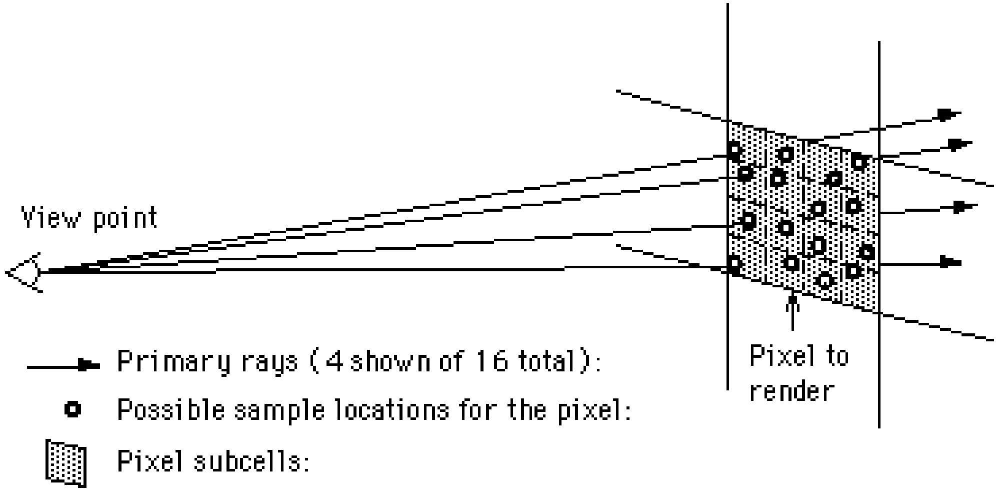
Materials

- Light ball
- Perfect glass ball
- Perfect diffuse ball
- Glossy metal



Antialiasing

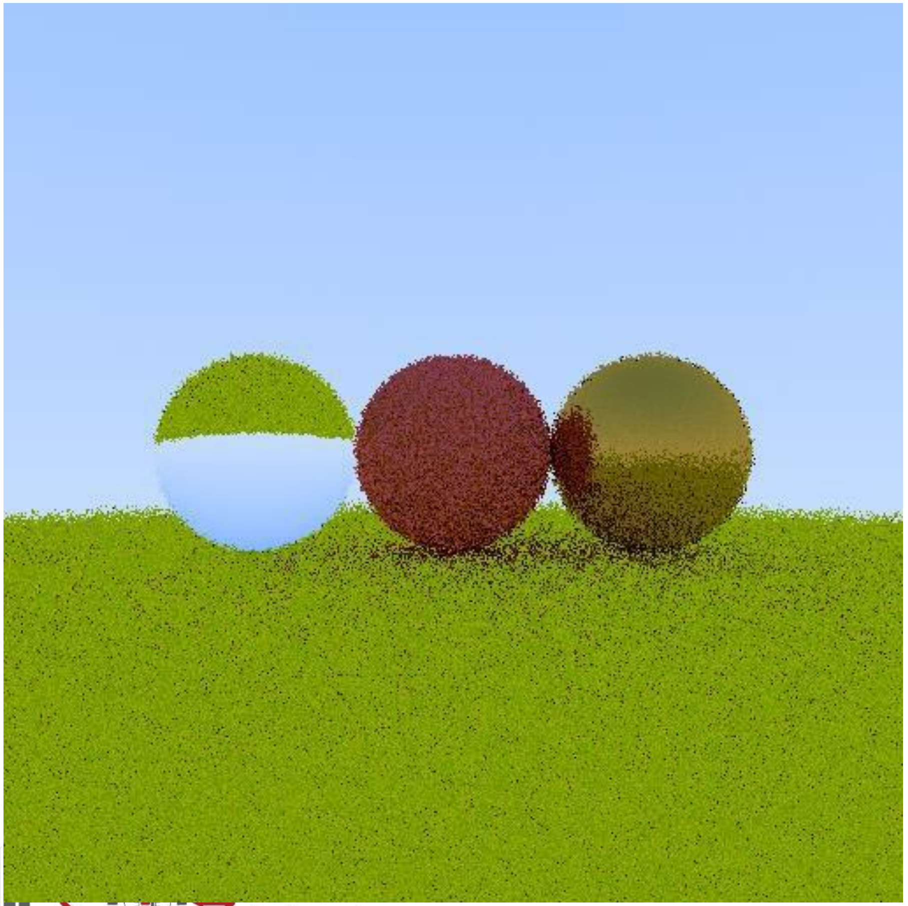
- Shoot multiple rays per pixel
- Final color will be average of those ray colors
- You can control this in rendering loop which is in main function.

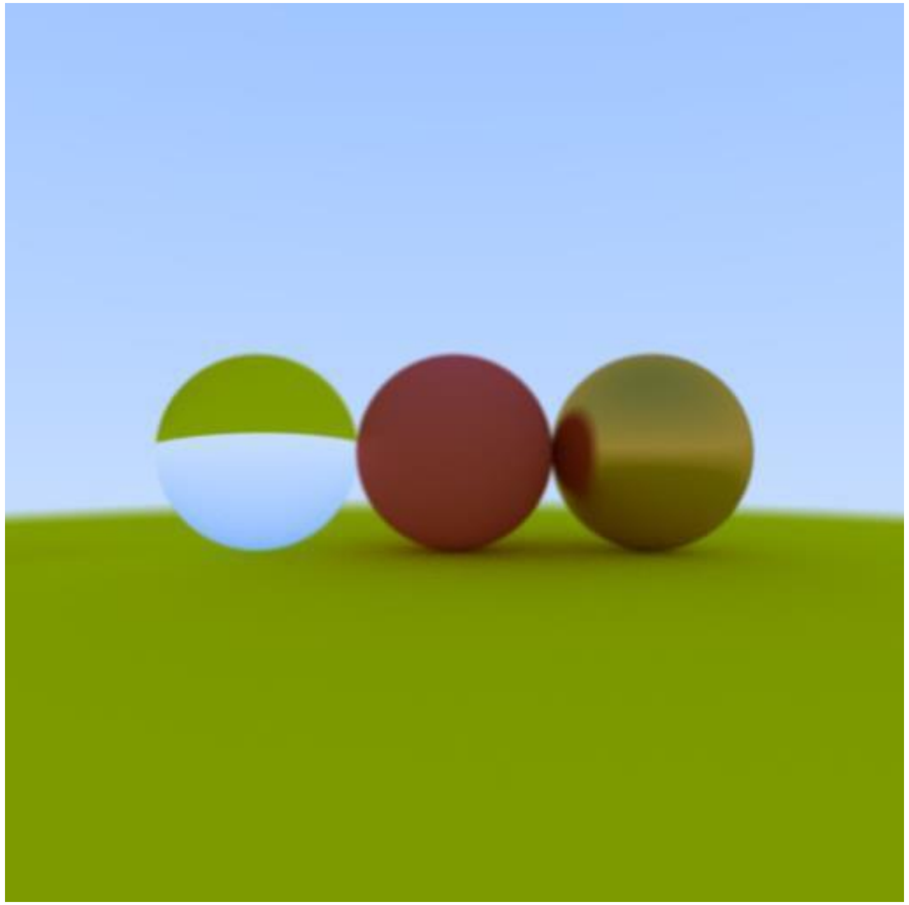

View point
Primary rays (4 shown of 16 total):
Possible sample locations for the pixel:
Pixel subcells:
Pixel to render

http://www.cs.montana.edu/~halla/cs525/intro.html

Antialiasing

- 1spp vs 1024 spp (samples per pixel)

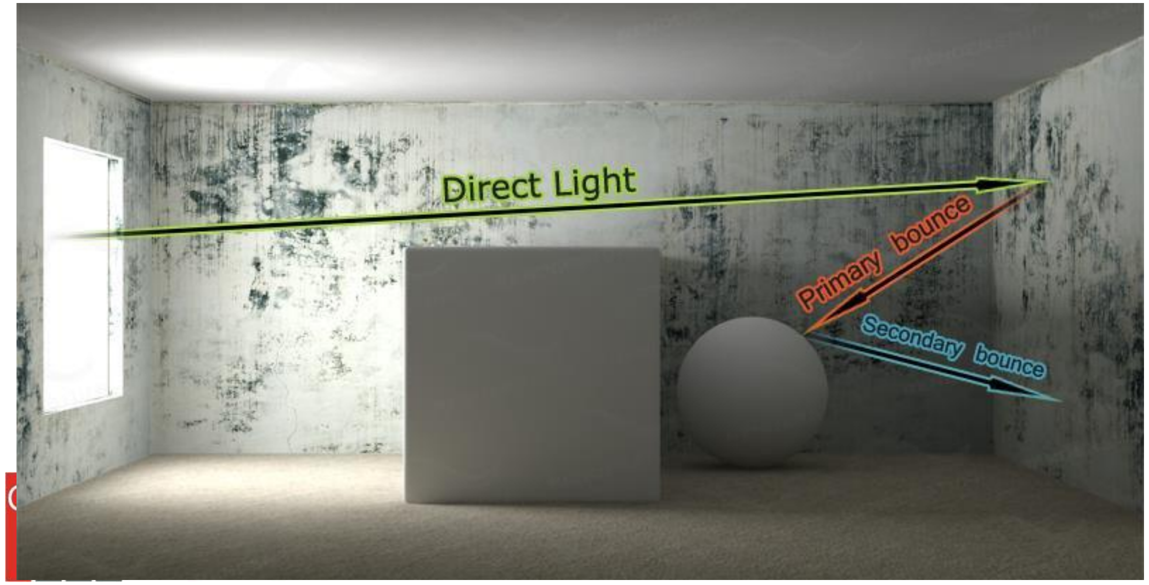


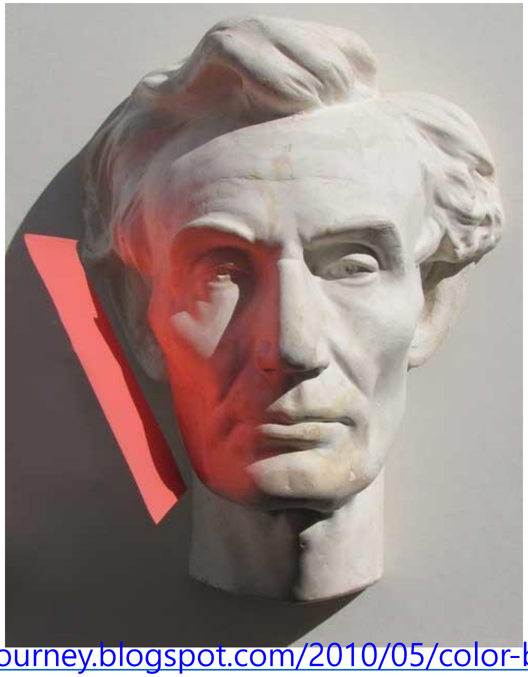


Indirect Lighting

- Simulate multiple bounce of light.
- You can see color bleeding (diffusive interreflections) after this!
- See **integrate** method in **integrator** class to control this behavior

https://www.pinterest.co.kr/pin/362117626263103458/

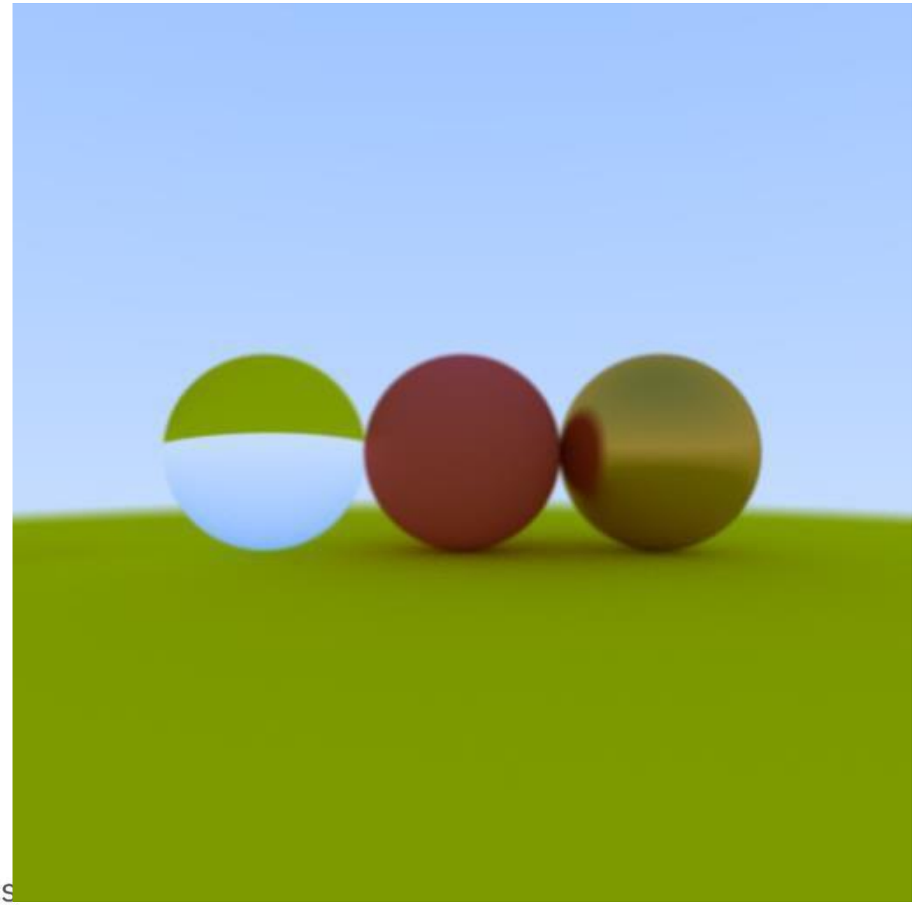




http://gurneyjourney.blogspot.com/2010/05/color-bleeding.html

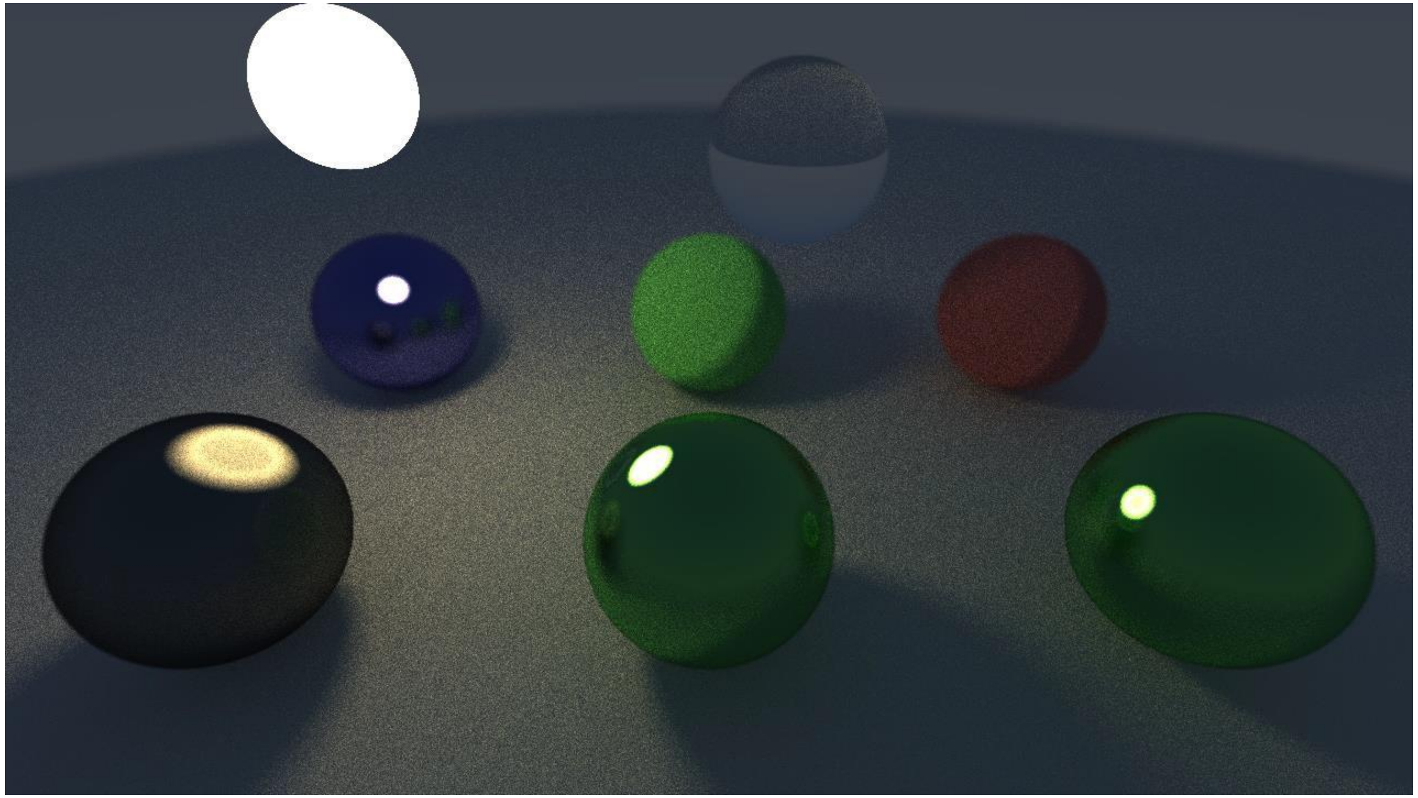
Indirect Lighting

- See red color bleeding under the red sphere



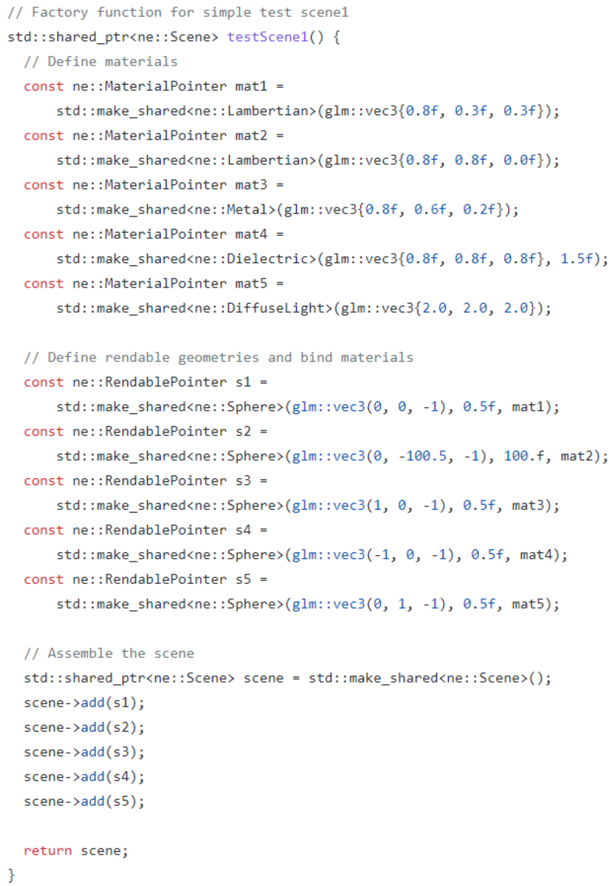
Direct Light Sampling

- You can remove these noises if you are using direct light sampling



Other Information

- There are two given scenes, testScene1 and testScene2.
  - Check test.cpp and test.hpp in neon-sandbox
  - The scene can be chosen in main.cpp.

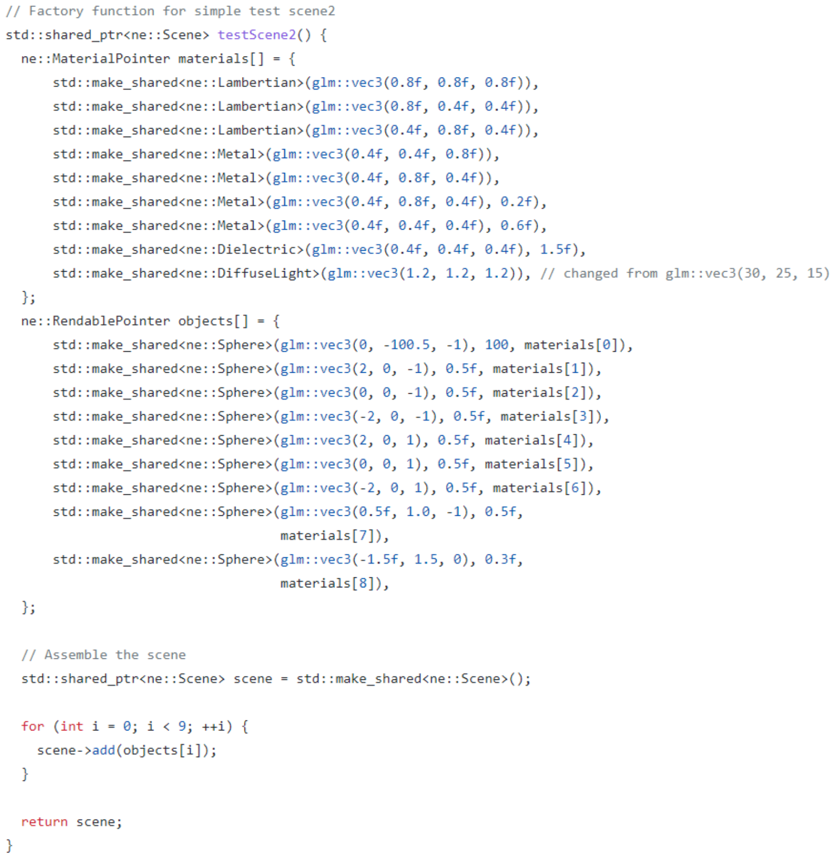


```cpp
// Factory function for simple test scene1
std::shared_ptr<ne::Scene> testScene1() {
  // Define materials
  const ne::MaterialPointer mat1 =
      std::make_shared<ne::Lambertian>(glm::vec3{0.8f, 0.3f, 0.3f});
  const ne::MaterialPointer mat2 =
      std::make_shared<ne::Lambertian>(glm::vec3{0.8f, 0.8f, 0.0f});
  const ne::MaterialPointer mat3 =
      std::make_shared<ne::Metal>(glm::vec3{0.8f, 0.6f, 0.2f});
  const ne::MaterialPointer mat4 =
      std::make_shared<ne::Dielectric>(glm::vec3{0.8f, 0.8f, 0.8f}, 1.5f);
  const ne::MaterialPointer mat5 =
      std::make_shared<ne::DiffuseLight>(glm::vec3{2.0, 2.0, 2.0});

  // Define rendable geometries and bind materials
  const ne::RendablePointer s1 =
      std::make_shared<ne::Sphere>(glm::vec3(0, 0, -1), 0.5f, mat1);
  const ne::RendablePointer s2 =
      std::make_shared<ne::Sphere>(glm::vec3(0, -100.5, -1), 100.f, mat2);
  const ne::RendablePointer s3 =
      std::make_shared<ne::Sphere>(glm::vec3(1, 0, -1), 0.5f, mat3);
  const ne::RendablePointer s4 =
      std::make_shared<ne::Sphere>(glm::vec3(-1, 0, -1), 0.5f, mat4);
  const ne::RendablePointer s5 =
      std::make_shared<ne::Sphere>(glm::vec3(0, 1, -1), 0.5f, mat5);

  // Assemble the scene
  std::shared_ptr<ne::Scene> scene = std::make_shared<ne::Scene>();
  scene->add(s1);
  scene->add(s2);
  scene->add(s3);
  scene->add(s4);
  scene->add(s5);

  return scene;
}
```



```cpp
// Factory function for simple test scene2
std::shared_ptr<ne::Scene> testScene2() {
  ne::MaterialPointer materials[] = {
      std::make_shared<ne::Lambertian>(glm::vec3(0.8f, 0.8f, 0.8f)),
      std::make_shared<ne::Lambertian>(glm::vec3(0.8f, 0.4f, 0.4f)),
      std::make_shared<ne::Lambertian>(glm::vec3(0.4f, 0.8f, 0.4f)),
      std::make_shared<ne::Metal>(glm::vec3(0.4f, 0.4f, 0.8f)),
      std::make_shared<ne::Metal>(glm::vec3(0.4f, 0.8f, 0.4f)),
      std::make_shared<ne::Metal>(glm::vec3(0.4f, 0.8f, 0.4f), 0.2f),
      std::make_shared<ne::Metal>(glm::vec3(0.4f, 0.4f, 0.4f), 0.6f),
      std::make_shared<ne::Dielectric>(glm::vec3(0.4f, 0.4f, 0.4f), 1.5f),
      std::make_shared<ne::DiffuseLight>(glm::vec3(1.2, 1.2, 1.2)), // changed from glm::vec3(30, 25, 15)
  };
  ne::RendablePointer objects[] = {
      std::make_shared<ne::Sphere>(glm::vec3(0, -100.5, -1), 100, materials[0]),
      std::make_shared<ne::Sphere>(glm::vec3(2, 0, -1), 0.5f, materials[1]),
      std::make_shared<ne::Sphere>(glm::vec3(0, 0, -1), 0.5f, materials[2]),
      std::make_shared<ne::Sphere>(glm::vec3(-2, 0, -1), 0.5f, materials[3]),
      std::make_shared<ne::Sphere>(glm::vec3(2, 0, 1), 0.5f, materials[4]),
      std::make_shared<ne::Sphere>(glm::vec3(0, 0, 1), 0.5f, materials[5]),
      std::make_shared<ne::Sphere>(glm::vec3(-2, 0, 1), 0.5f, materials[6]),
      std::make_shared<ne::Sphere>(glm::vec3(0.5f, 1.0, -1), 0.5f,
                                   materials[7]),
      std::make_shared<ne::Sphere>(glm::vec3(-1.5f, 1.5, 0), 0.3f,
                                   materials[8]),
  };

  // Assemble the scene
  std::shared_ptr<ne::Scene> scene = std::make_shared<ne::Scene>();

  for (int i = 0; i < 9; ++i) {
    scene->add(objects[i]);
  }

  return scene;
}
```

Computer Graphics
Laboratory

PA4 Link

1. Login to github

2. Go to following link – https://classroom.github.com/a/cMcFtX-e

3. Accept the assignment

Additional Materials for PA

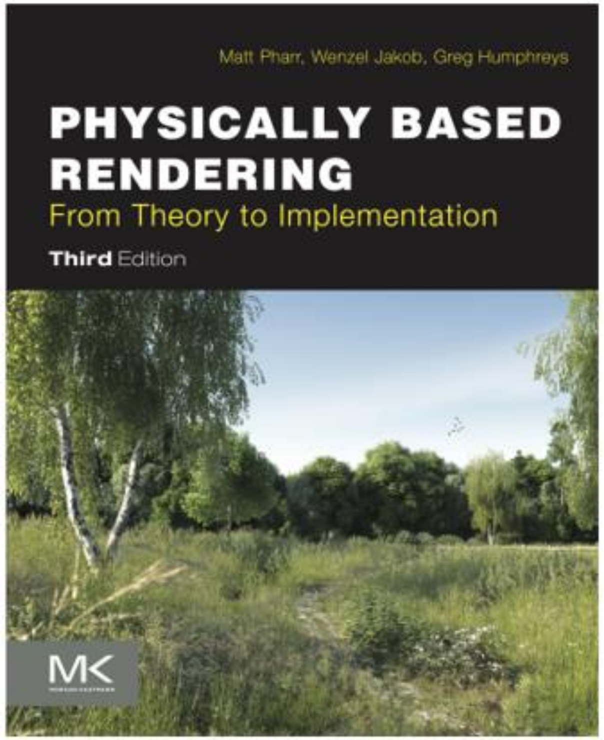
2022 Computer Graphics

Computer Graphics
Laboratory

Additional Materials

If you are having difficulty with this programming assignment, please check the materials as well:

- Physically Based Rendering: From Theory to Implementation
  - Book: there are a lot of books in GIST library
  - E-book: https://www.pbr-book.org/
  - Code: https://github.com/mmp/pbrt-v3


Matt Pharr, Wenzel Jakob, Greg Humphreys
PHYSICALLY BASED RENDERING
From Theory to Implementation
Third Edition
MK