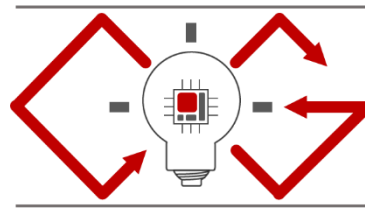
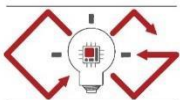


Programming Assignment 4

2023 Computer Graphics



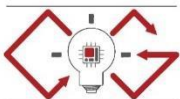
Computer Graphics
Laboratory



Computer Graphics
Laboratory

Submission

- Deadline: 23:59:59, Sunday, June 11th , 2023 (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 ignored. (I will use github server timestamp instead)



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
- Your final grade will be “F”
 - Copy

Commenting Commit ID

master 1 branch 0 tags

Go to file Add file <> Code

ychanu test1	444ea6f 3 hours ago	2 commits
doc	Initial commit	4 hours ago
extern	Initial commit	4 hours ago
src	test1	3 hours ago
.gitignore	Initial commit	4 hours ago
.gitmodules	Initial commit	4 hours ago
CMakeLists.txt	Initial commit	4 hours ago
README.md	Initial commit	4 hours ago
report.md	Initial commit	4 hours ago

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

<> Code Issues 1 Pull requests Actions Projects Security Insights

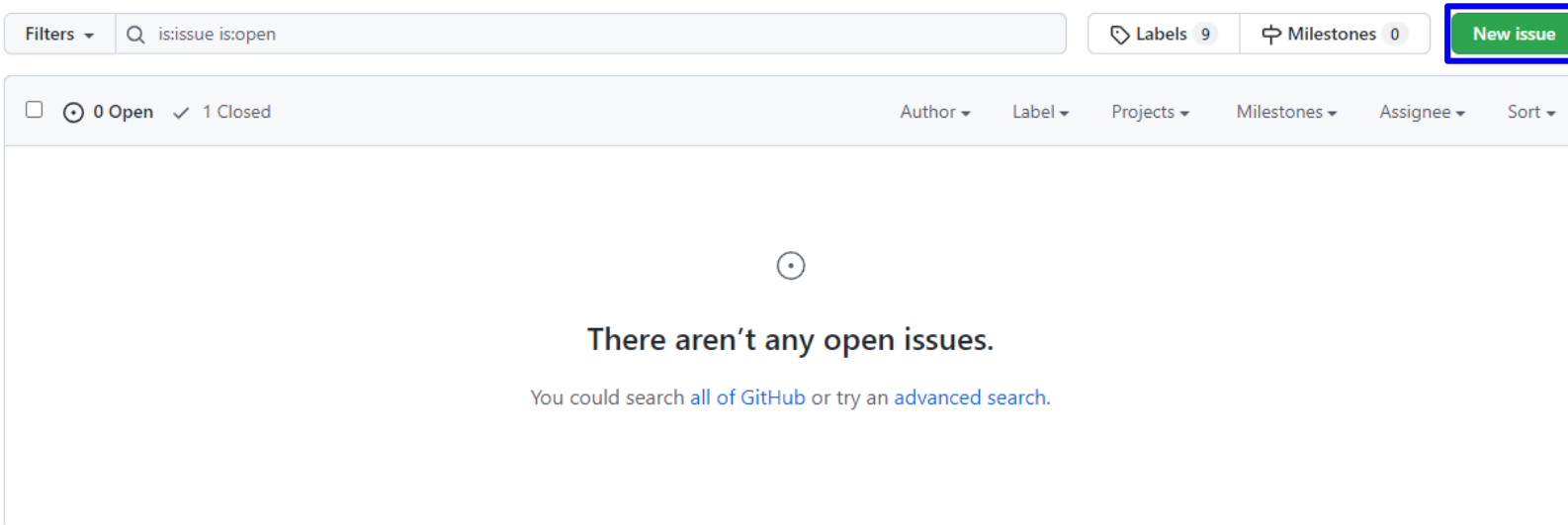
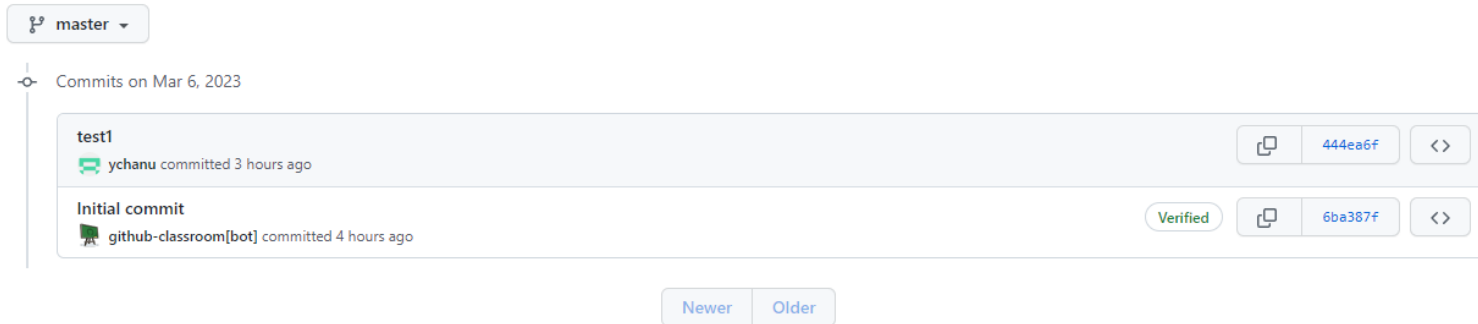
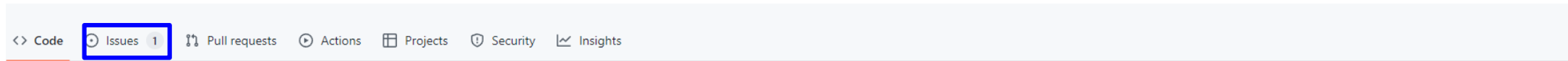
master

Commits on Mar 6, 2023

test1	ychanu committed 3 hours ago	444ea6f	<>
Initial commit	github-classroom[bot] committed 4 hours ago	Verified 6ba387f	<>

Newer Older

Commenting Commit ID



1. Go to issue tab
2. Click "New issue"

Commenting Commit ID

Final Submit

Write Preview

H B I @

444ea6f55b20b559dd59070a0e1fd60a2127e2ee

Attach files by dragging & dropping, selecting or pasting them.

Styling with Markdown is supported

Submit new issue

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).



Paste your latest commit id (Ctrl+v)

Computer Graphics Laboratory

Click "Submit new issue"

Task List

1. Materials (12 Points)

- Lambertian, Metal, Dielectric, Area light (Emissive material)
- Implement **scatter** function in each material class (material.cpp)

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

Illumination

- Rendering equation [Kajiya 86] (another form: [Immel and Cohen 86])

- $$L_s(\mathbf{x}, \omega_o) = \int_{\text{all } \mathbf{x}'} \frac{\rho(\omega_i, \omega_o) L_s(\mathbf{x}', \mathbf{x} - \mathbf{x}') v(\mathbf{x}, \mathbf{x}') \cos\theta_i \cos\theta'}{\|\mathbf{x} - \mathbf{x}'\|^2} dA$$

It would be difficult to implement a complete rendering equation (Optional)

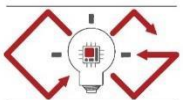
- Phong illumination model

- $$I = \sum_{i=1}^{\# \text{ of lights}} \{L_a^i k_a + L_d^i k_d \max(0, \mathbf{n} \cdot \mathbf{l}^i) + L_s^i k_s \max(0, \mathbf{r}^i \cdot \mathbf{v})^s\}$$

- Whitted illumination model

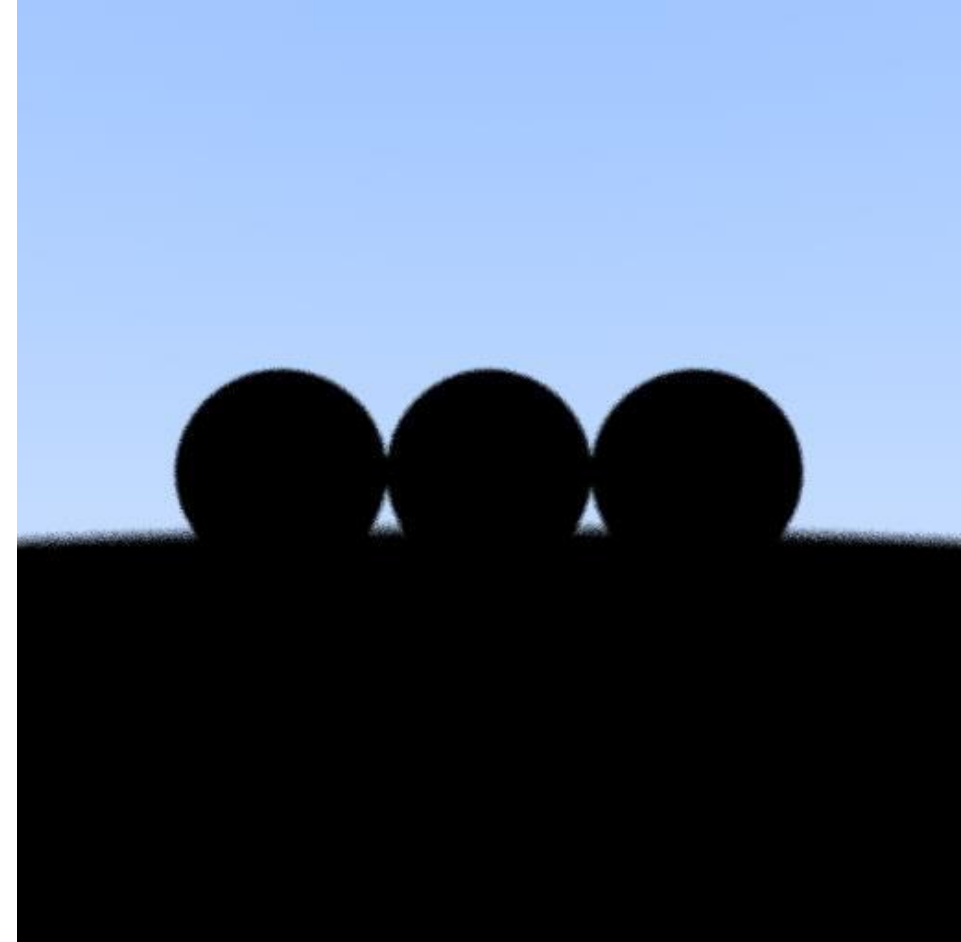
- $$I = \sum_{i=1}^{\# \text{ of lights}} \{L_a^i k_a + L_d^i k_d \max(0, \mathbf{n} \cdot \mathbf{l}^i)\} + k_s S + k_t T$$

As long as we can clearly distinguish the properties of the materials in PA 4, any illumination model is fine.



Initial Appearance

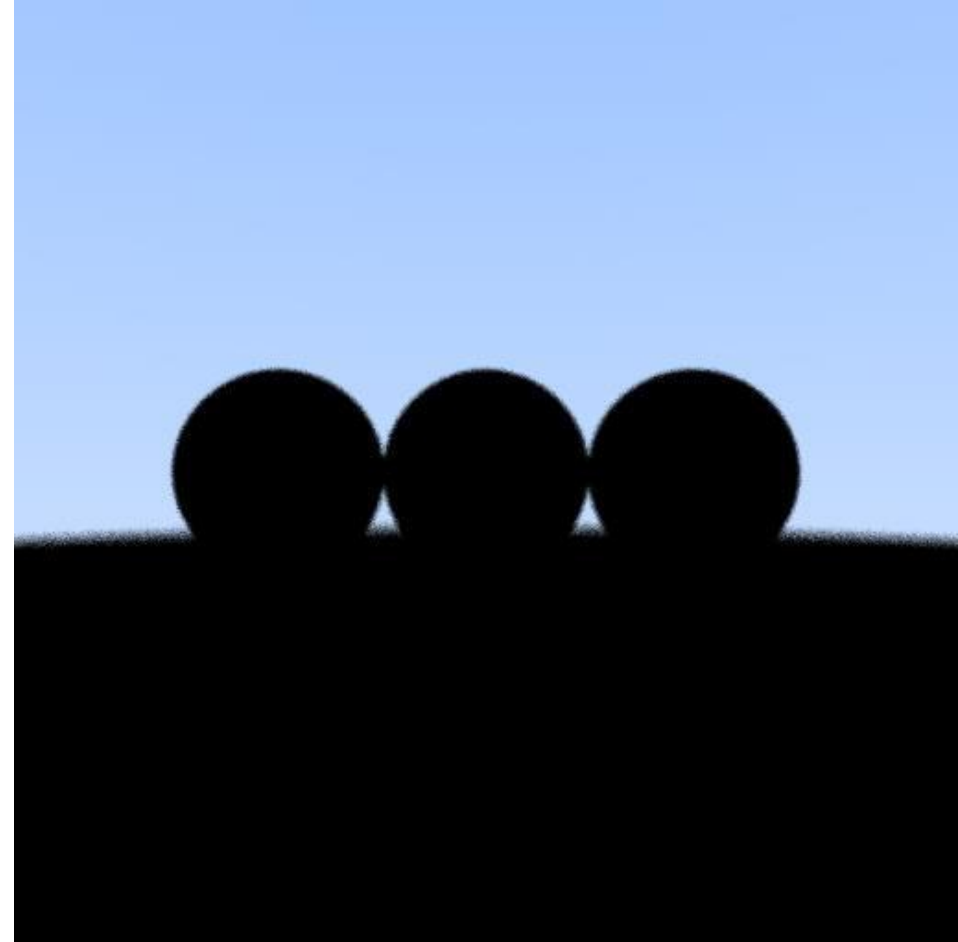
- Skeleton code: Neon renderer
 - Unlike OpenGL project, the result will be png file.
- output: *.png



Initial Appearance

- If it takes too much time to render the scene, test your initial code with low resolution.
- And render with high resolution for your final results. (in report)

```
int main(int argc, char *argv[]) {  
    // int nx = 512;  
    // int ny = 512;  
    int nx = 128;  
    int ny = 128;  
}
```



Initial Appearance (test.cpp)

```
// Factory function for simple test scene
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);

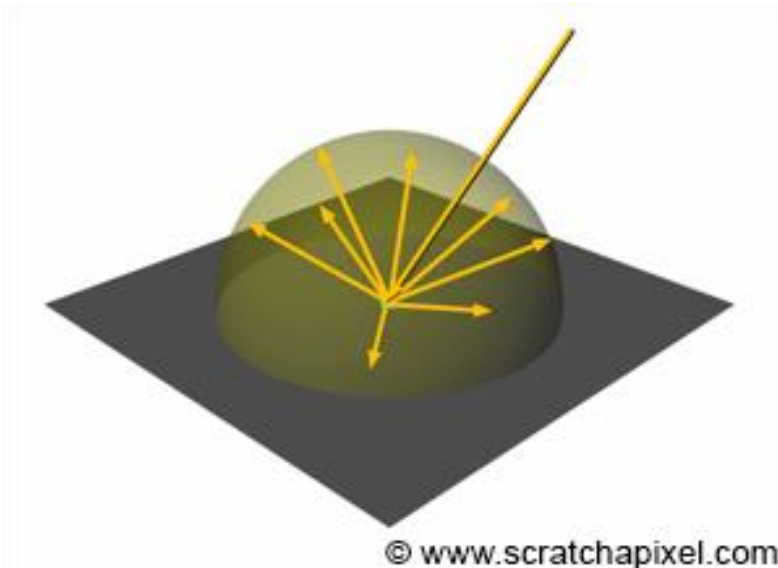
    // Assemble the scene
    std::shared_ptr<ne::Scene> scene = std::make_shared<ne::Scene>();
    scene->add(s1);
    scene->add(s2);
    scene->add(s3);
    scene->add(s4);
    scene->add(s5);

    return scene;
}
```

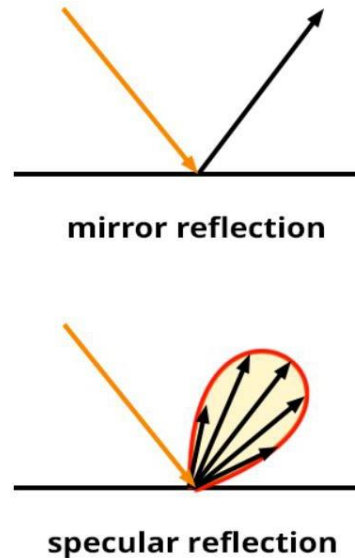
- S1: center circle
- S2: ground
- S3: right circle
- S4: left circle
- S5: light

Materials

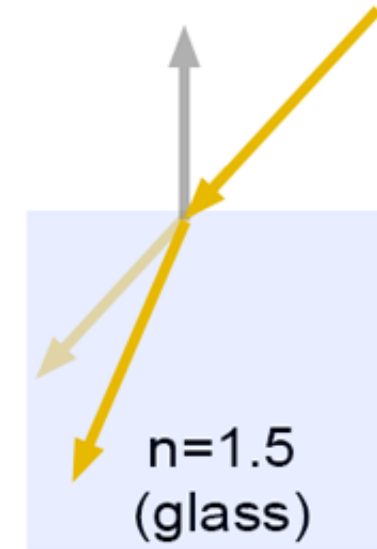
See **scatter** method in each material class



**Lambertian
(diffuse)**

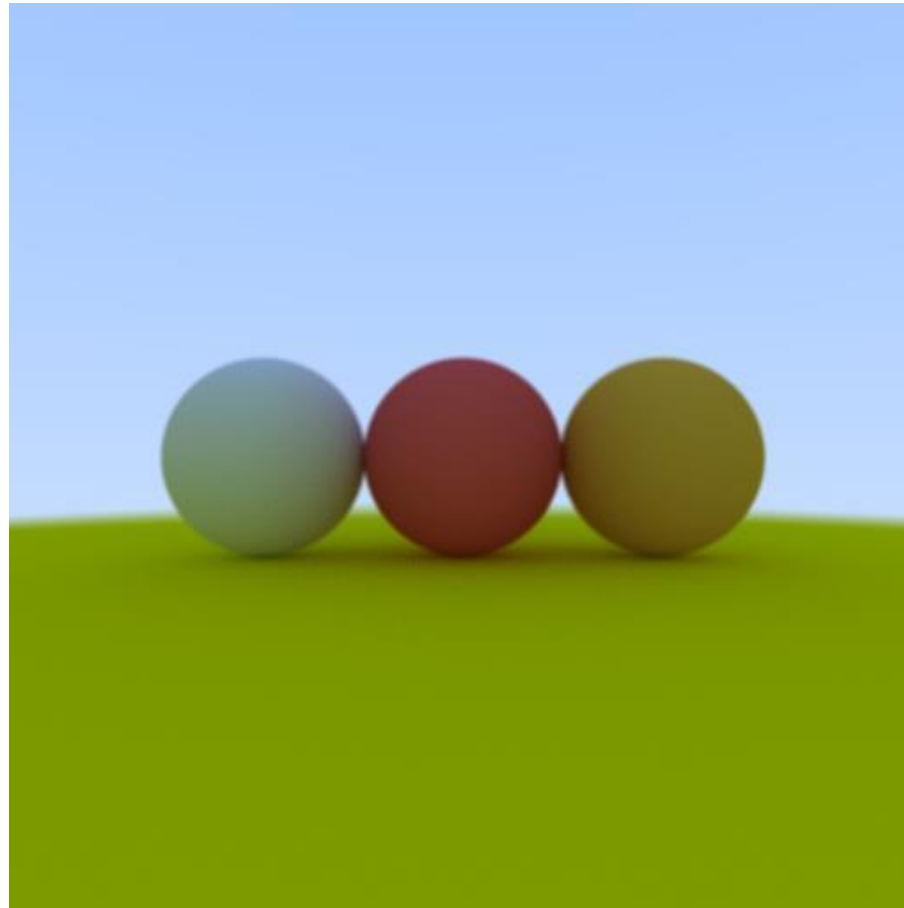


**Metal
(mirror with randomness)**



Dielectric

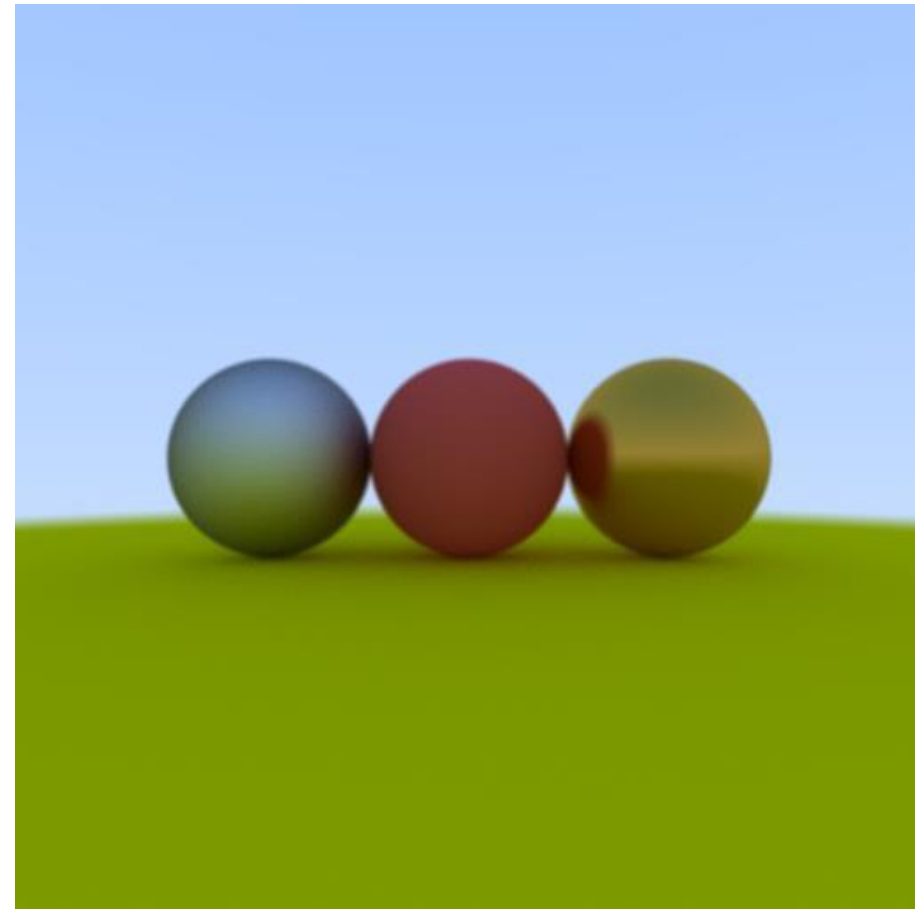
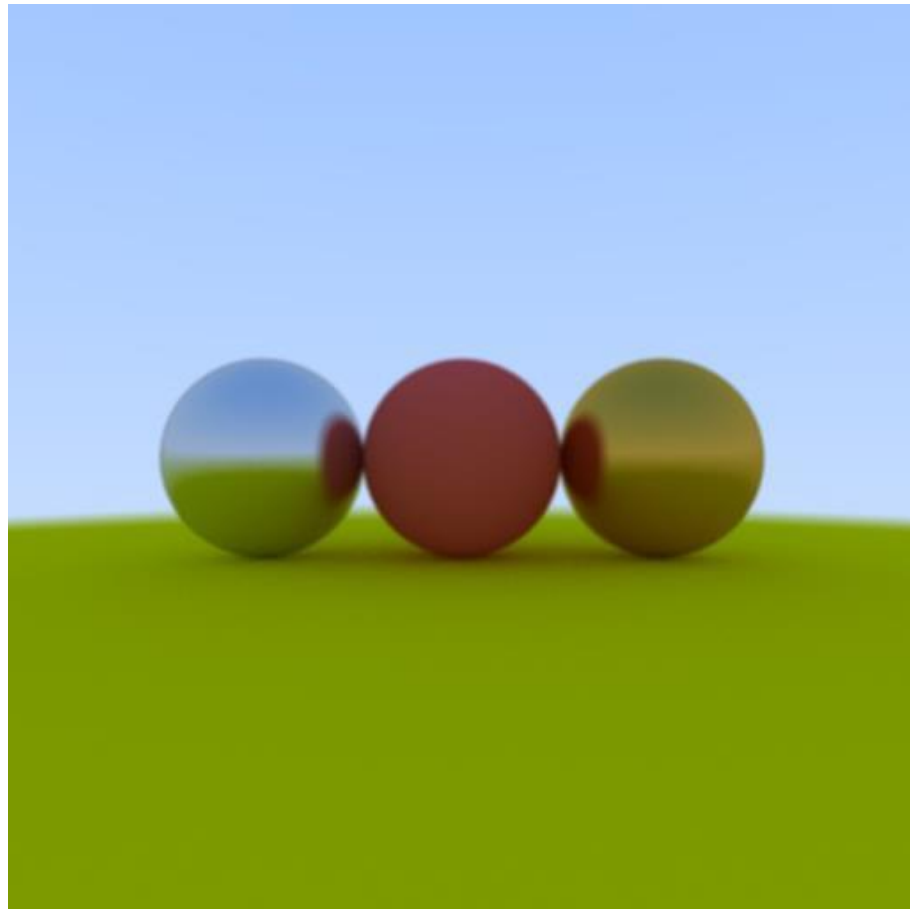
Materials



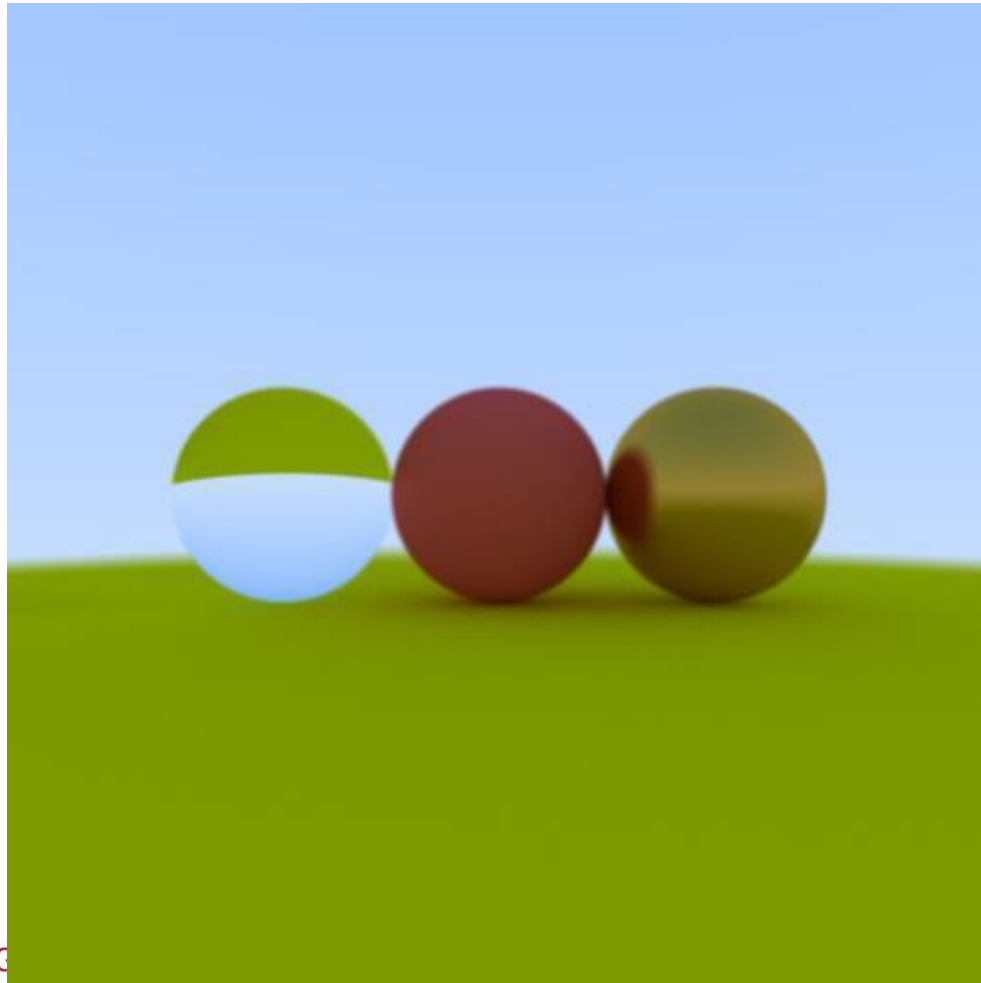
Lambertian (diffuse)

Materials

- Perfect mirror vs metal (mirror with randomness)



Materials



dielectric material

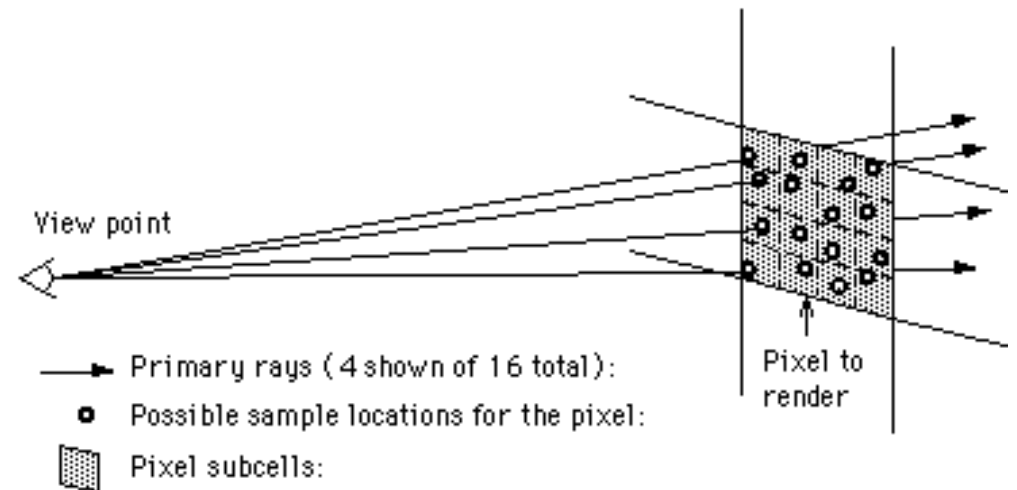
Materials

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



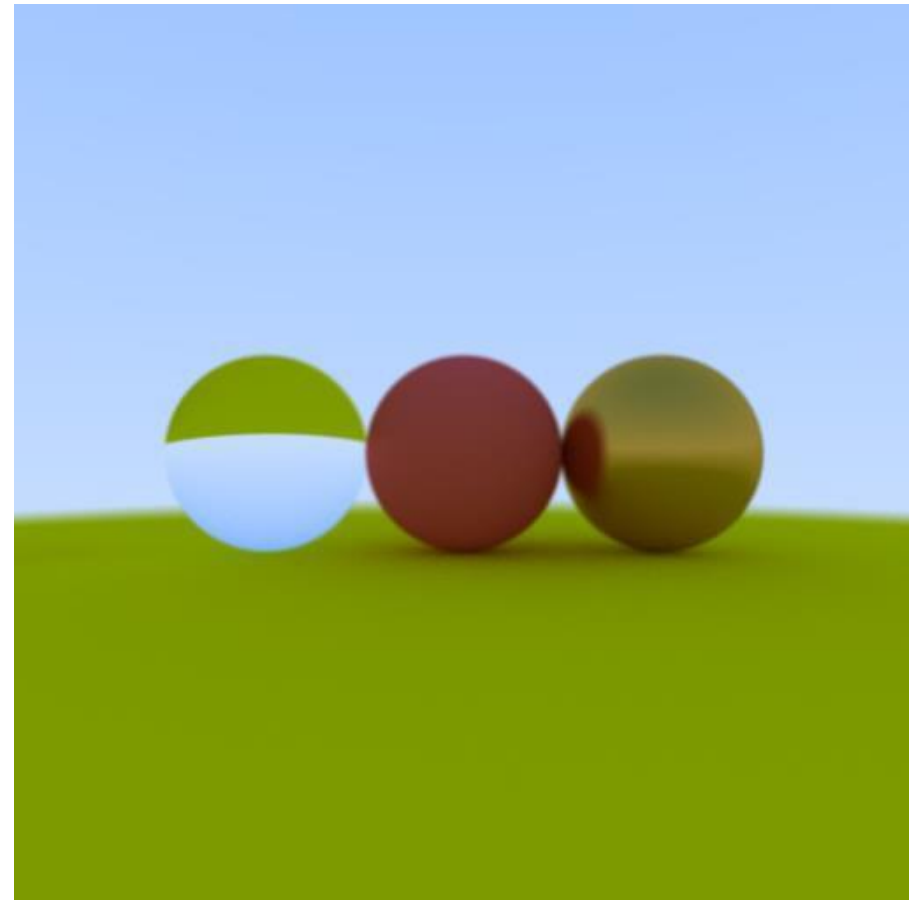
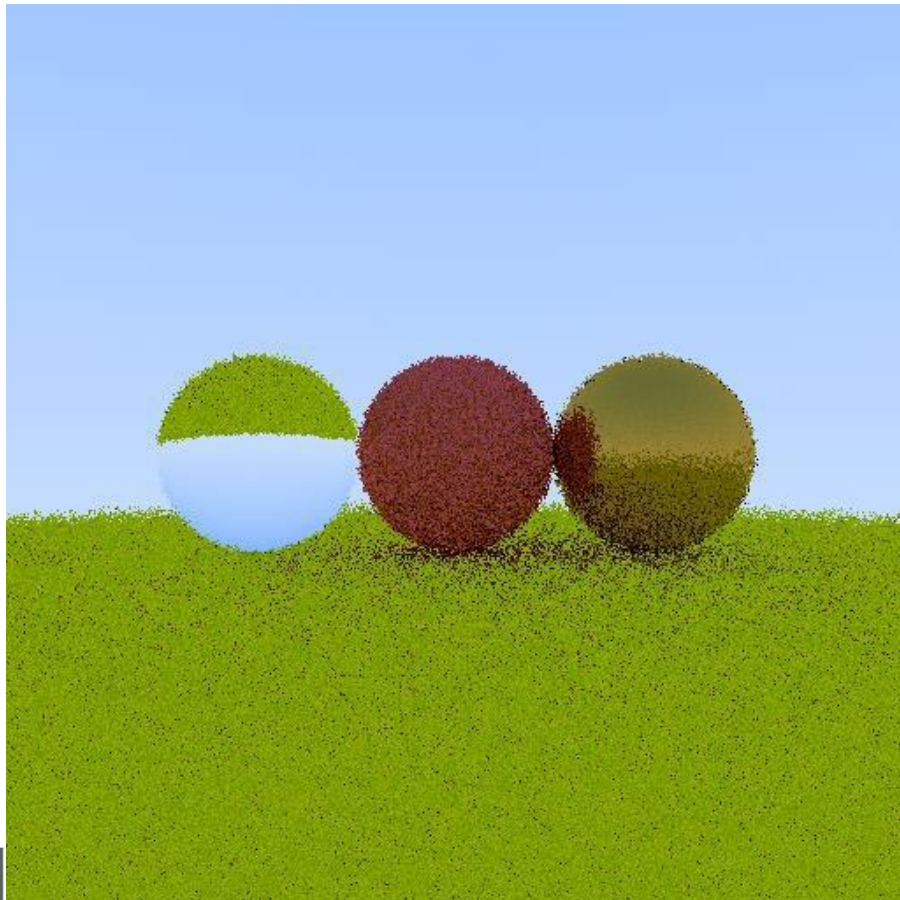
Antialiasing

- 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.



Antialiasing

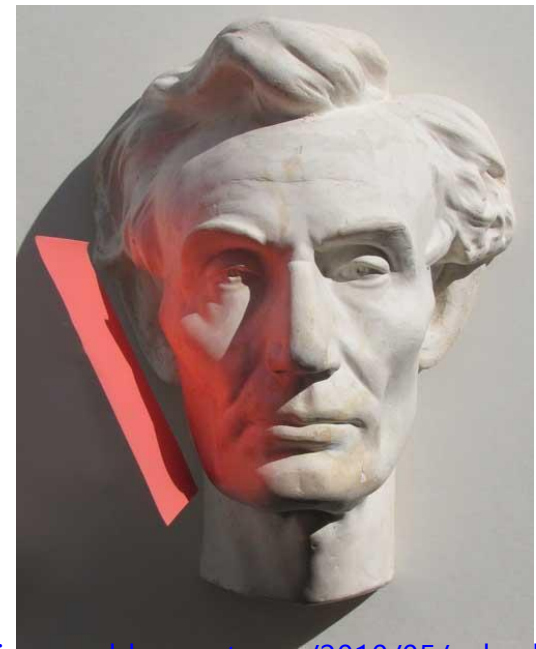
- 1spp vs 1024 spp (samples per pixel)



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

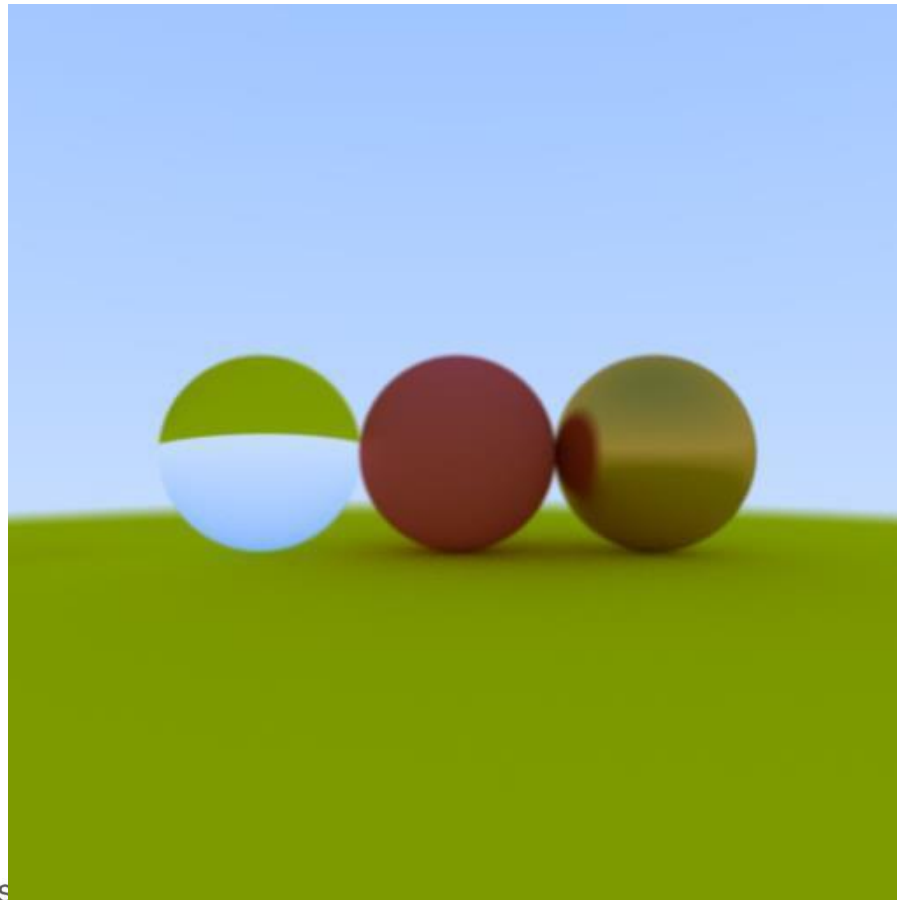
<https://www.pinterest.co.kr/pin/362117626263103458/>



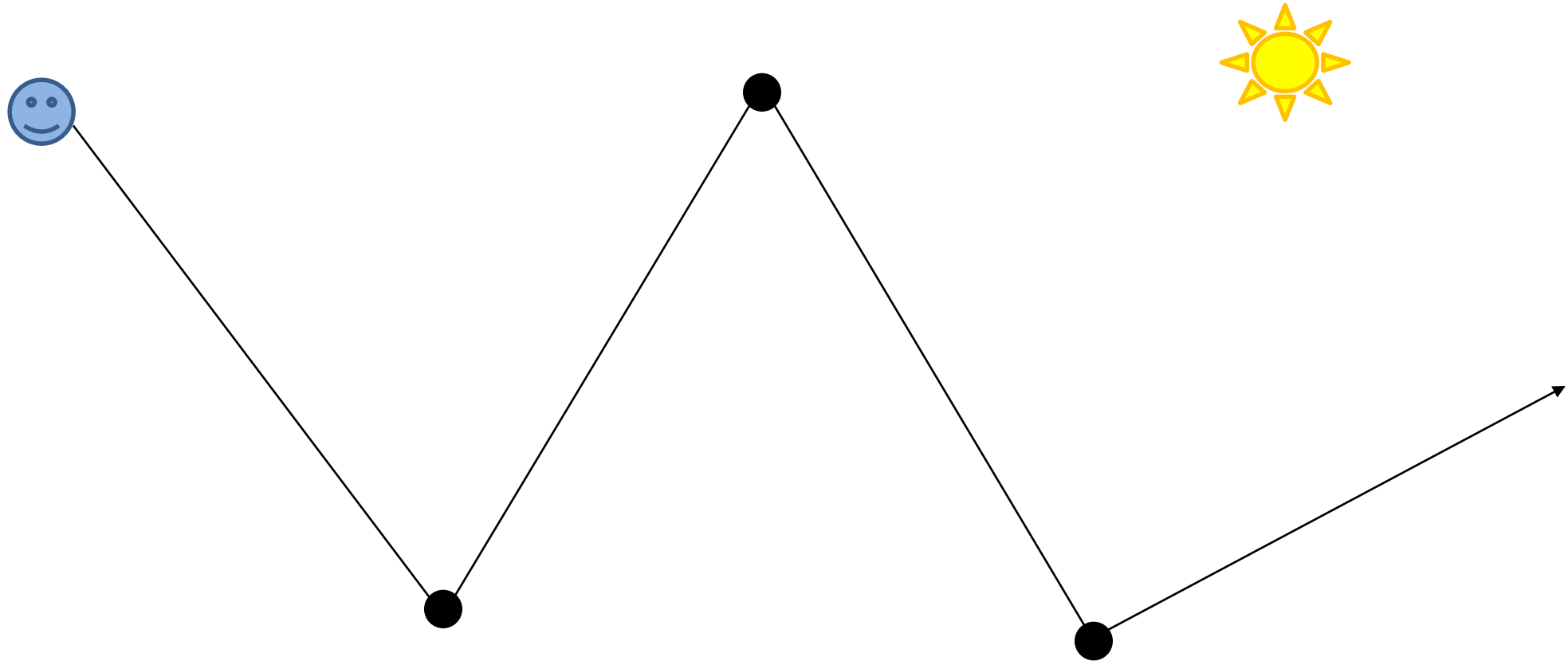
<http://gurneyjourney.blogspot.com/2010/05/color-bleeding.html>

Indirect Lighting

- See red color bleeding under the red sphere

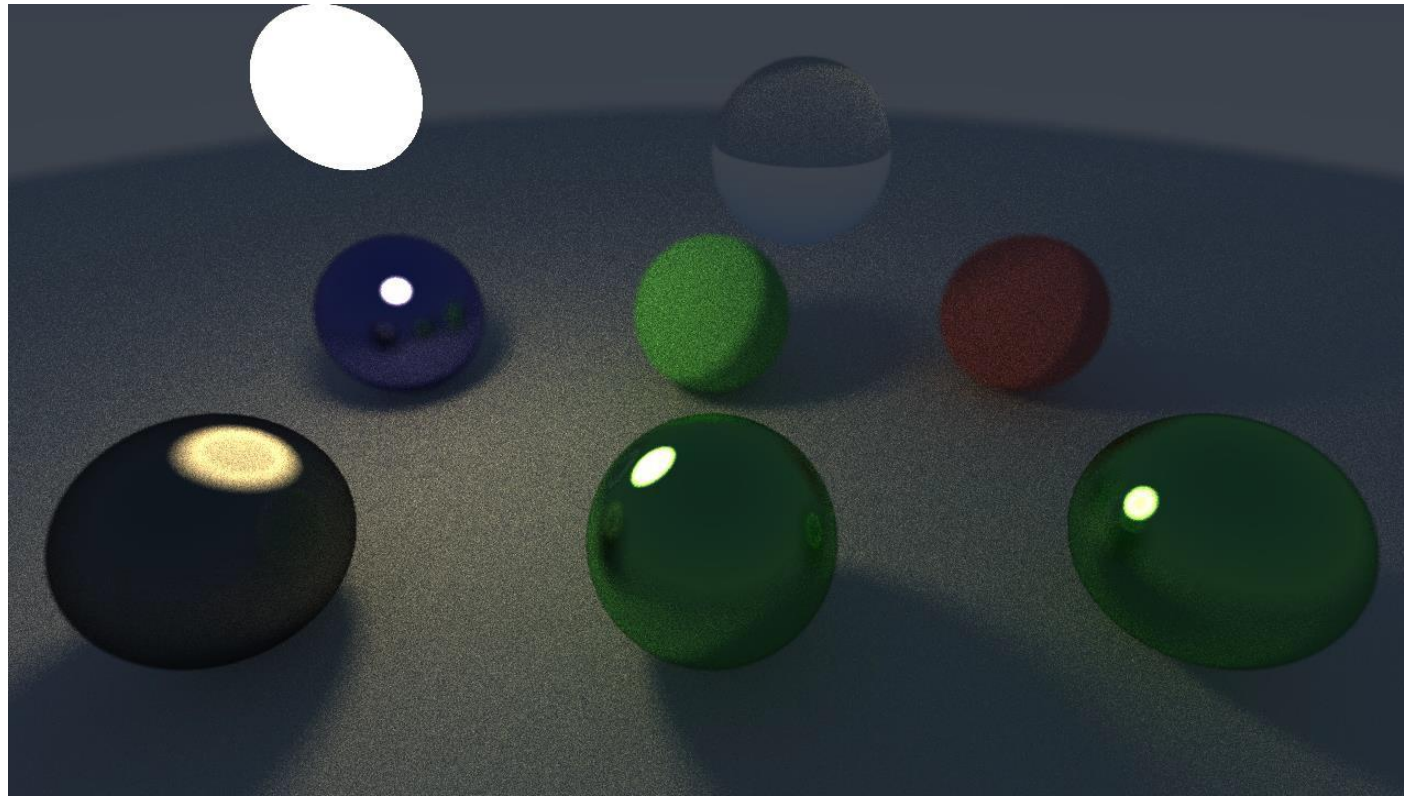


Indirect Lighting

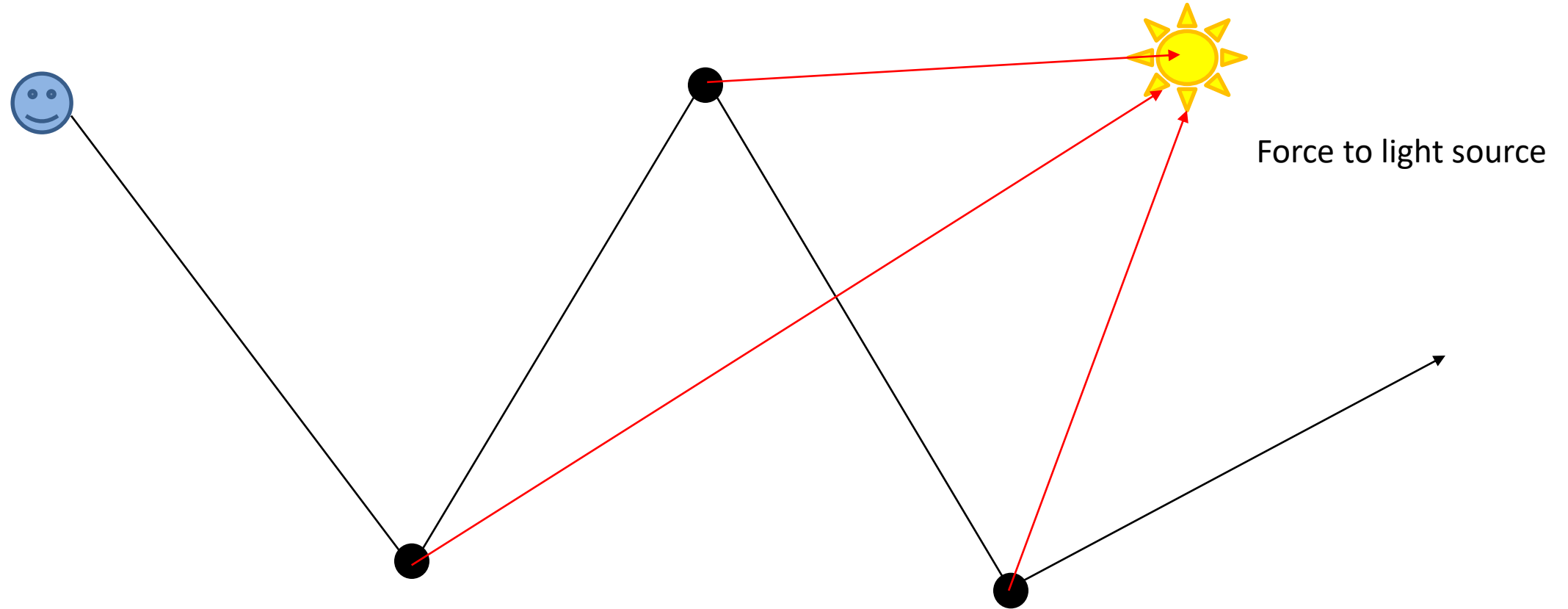


Direct Light Sampling

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



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.

```
// create scene
// std::shared_ptr<ne::Scene> scene = testScene1();
std::shared_ptr<ne::Scene> scene = testScene2();
```

```
// 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;
}
```

```
// 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]);
    }

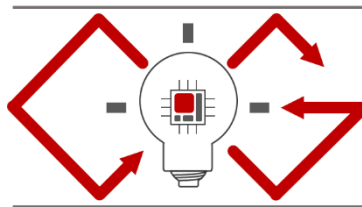
    return scene;
}
```

PA4 Link

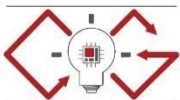
1. Login to github
2. Go to following link – <https://classroom.github.com/a/hjtDlcZd>
3. Accept the assignment

Additional Materials for PA

2023 Computer Graphics



Computer Graphics
Laboratory



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>

