

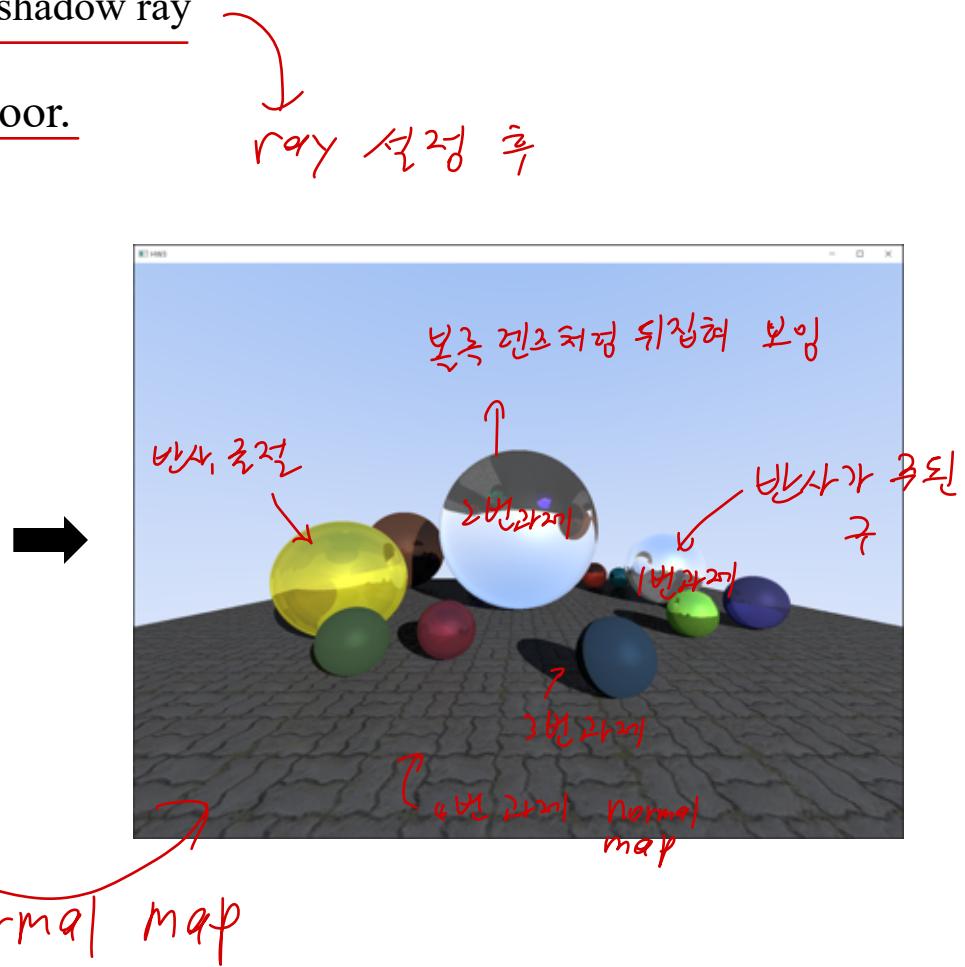
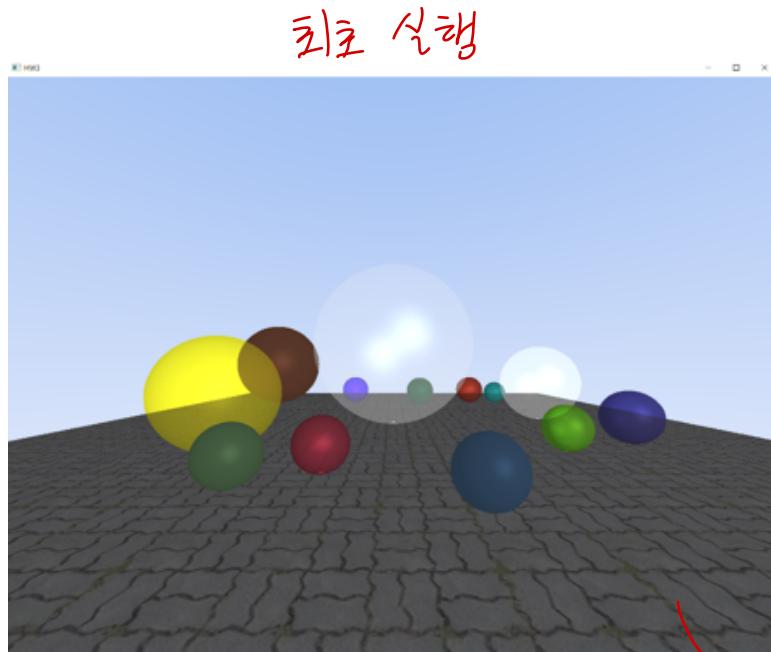
# Homework 3

---

COSE331 Computer Graphics

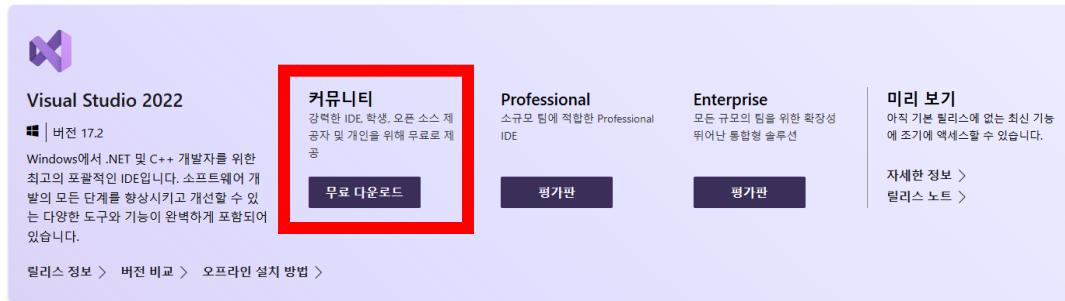
# Goal

- Render spheres of various materials using ray tracing technique.
  - Calculate reflection ray, refraction ray, shadow ray
- Construct a normal map to shade the floor.



# Visual Studio

- <https://visualstudio.microsoft.com/ko/downloads/>
- Visual Studio for C++ is not run on macOS. We highly recommend using Windows device for HW3, contact TAs if you need help.



The screenshot shows the Visual Studio 2022 download page in Korean. It features three main product options: Community, Professional, and Enterprise. The 'Community' section is highlighted with a red box. The 'Professional' and 'Enterprise' sections are shown below it. The 'Community' section includes a '무료 다운로드' (Free Download) button. The 'Professional' and 'Enterprise' sections also have '평가판' (Evaluation) buttons. To the right, there is a sidebar with a '미리 보기' (Preview) section and a 'Community' section. Below the main content, there are separate sections for 'Mac용 Visual Studio 2022' and 'Visual Studio Code'.

Visual Studio 2022

Community

Professional

Enterprise

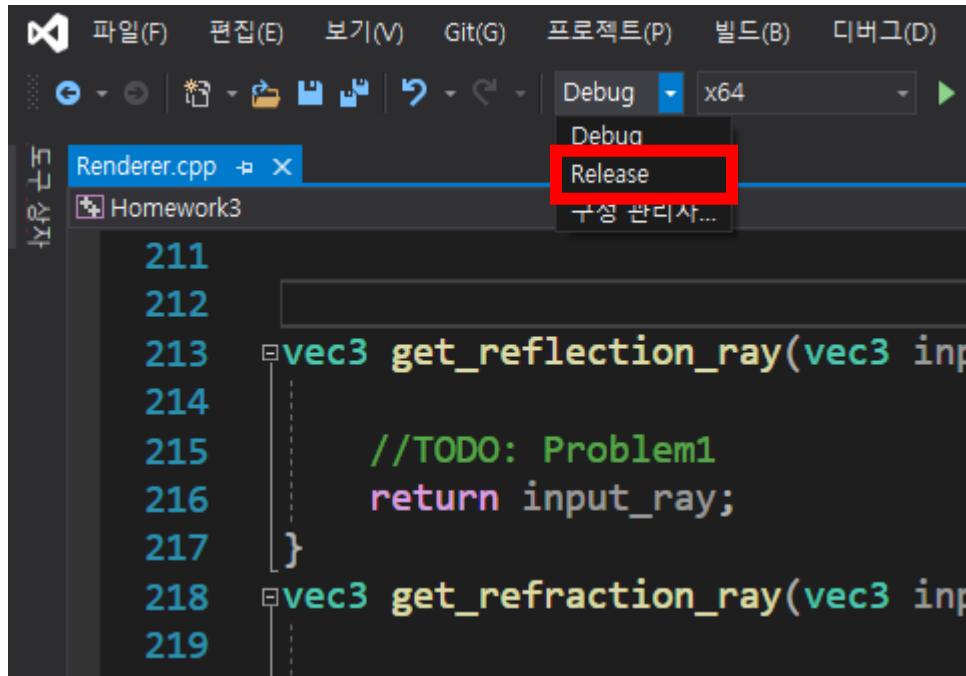
미리 보기

Mac용 Visual Studio 2022

Visual Studio Code

# Visual Studio

- Open the VS solution file (Homework3.sln)
- Change the build configuration to Release



A screenshot of the Visual Studio IDE interface. The top navigation bar includes '파일(F)', '편집(E)', '보기(V)', 'Git(G)', '프로젝트(P)', '빌드(B)', and '디버그(D)'. Below the navigation bar is a toolbar with various icons. The main window shows a code editor with the file 'Renderer.cpp' open. The code contains several lines of C++ code, including function definitions for reflection and refraction rays. A dropdown menu for 'Build Configuration' is open, showing options 'Debug' and 'Release'. The 'Release' option is highlighted with a red box. At the bottom of the screen, there is a status bar with some text and icons.

```
211
212
213     vec3 get_reflection_ray(vec3 inp
214
215         //TODO: Problem1
216         return input_ray;
217     }
218     vec3 get_refraction_ray(vec3 inp
219
```

# Ray Tracing

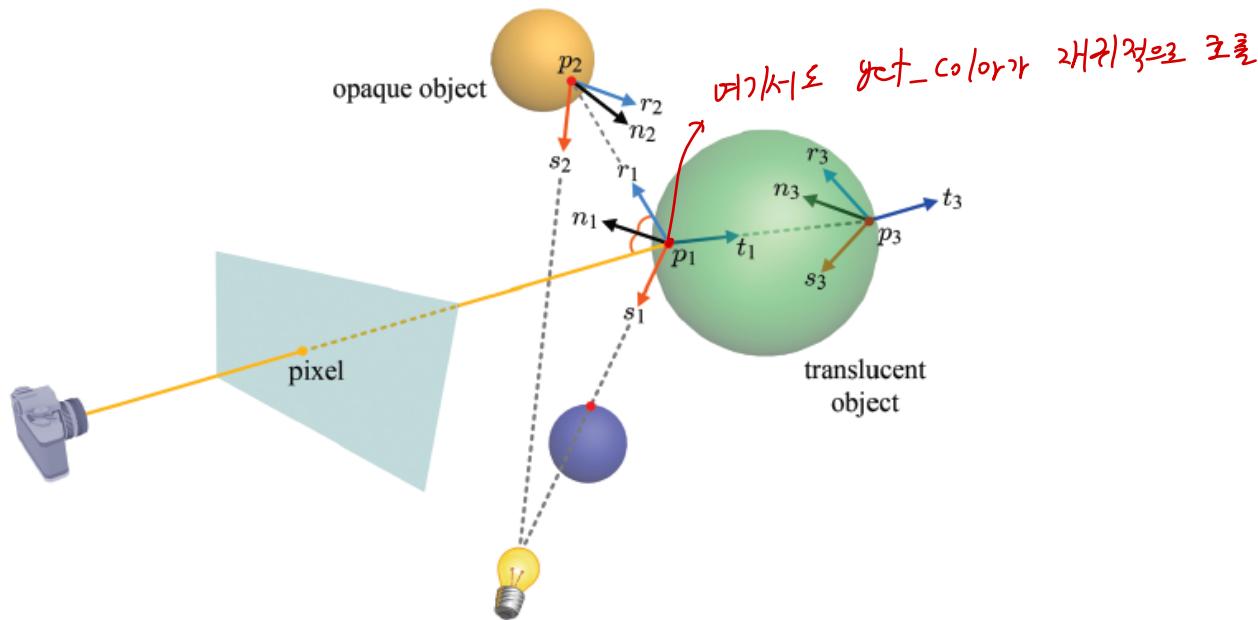
depth == 0 되면 get\_color 안 함  
몇 번까지 뒤가는지.

camera에서 나가는 빛의 방향

```
■ vec3 get_color(int depth, vec3 eye, vec3 ray)
```

- Sums three ray colors: reflection ray, refraction ray, shadow ray.
- Then the result is multiplied piecewise with the shaded color, making a color bias.

(The code for this part is provided.)



# Problems

---

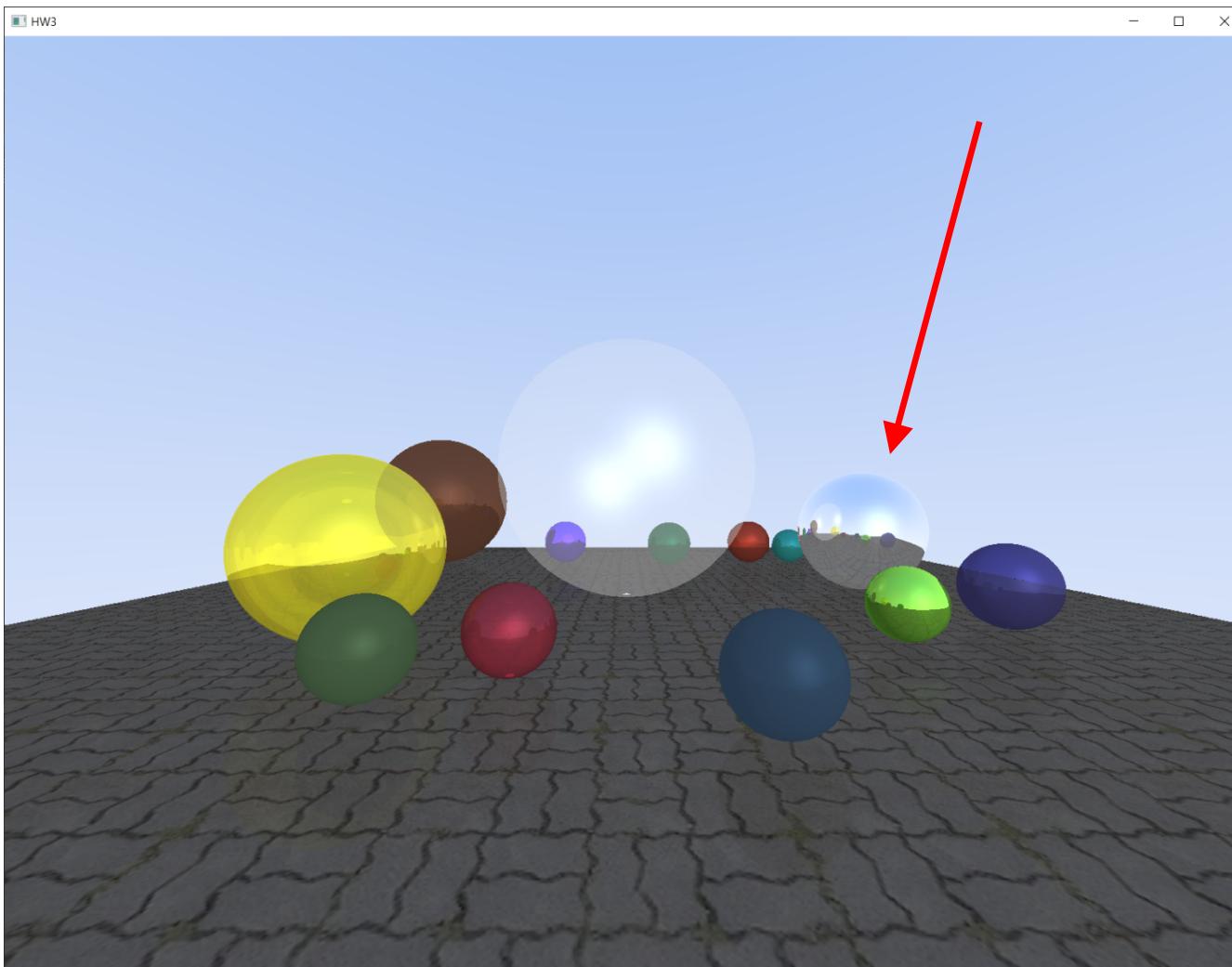
- Fill in some empty functions in Renderer.cpp file.

- `vec3 get_reflection_ray(vec3 input_ray, HitData hit)`
- `vec3 get_refraction_ray(vec3 input_ray, HitData hit, Material m)`
- `bool is_lighted(vec3 eye)` : *shadow가 되었는지*
- `void construct_normal_map(img_height, img_normal)`

# Problem 1

- ray의 방향  
빛이 물체의 어느 위치에서  
증을 했는지, normal  
등등
- `vec3 get_reflection_ray(vec3 input_ray, HitData hit)`
    - Calculate the reflection ray.
    - Primary ray incident data is stored in HitData. (see page 16)
    - The returned vector must be normalized. 정사진 벡터.

# Problem 1



# Problem 2

- `vec3 get_refraction_ray(vec3 input_ray, HitData hit, Material m)`

- Calculate the refraction ray.
- The returned vector must be normalized.

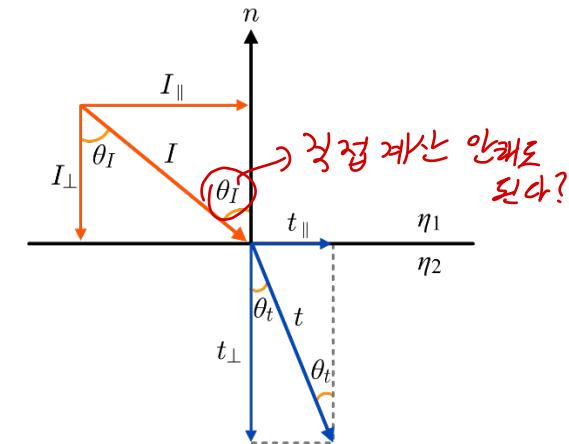
구조를 n 사용.

$m.n$  으로 가중용

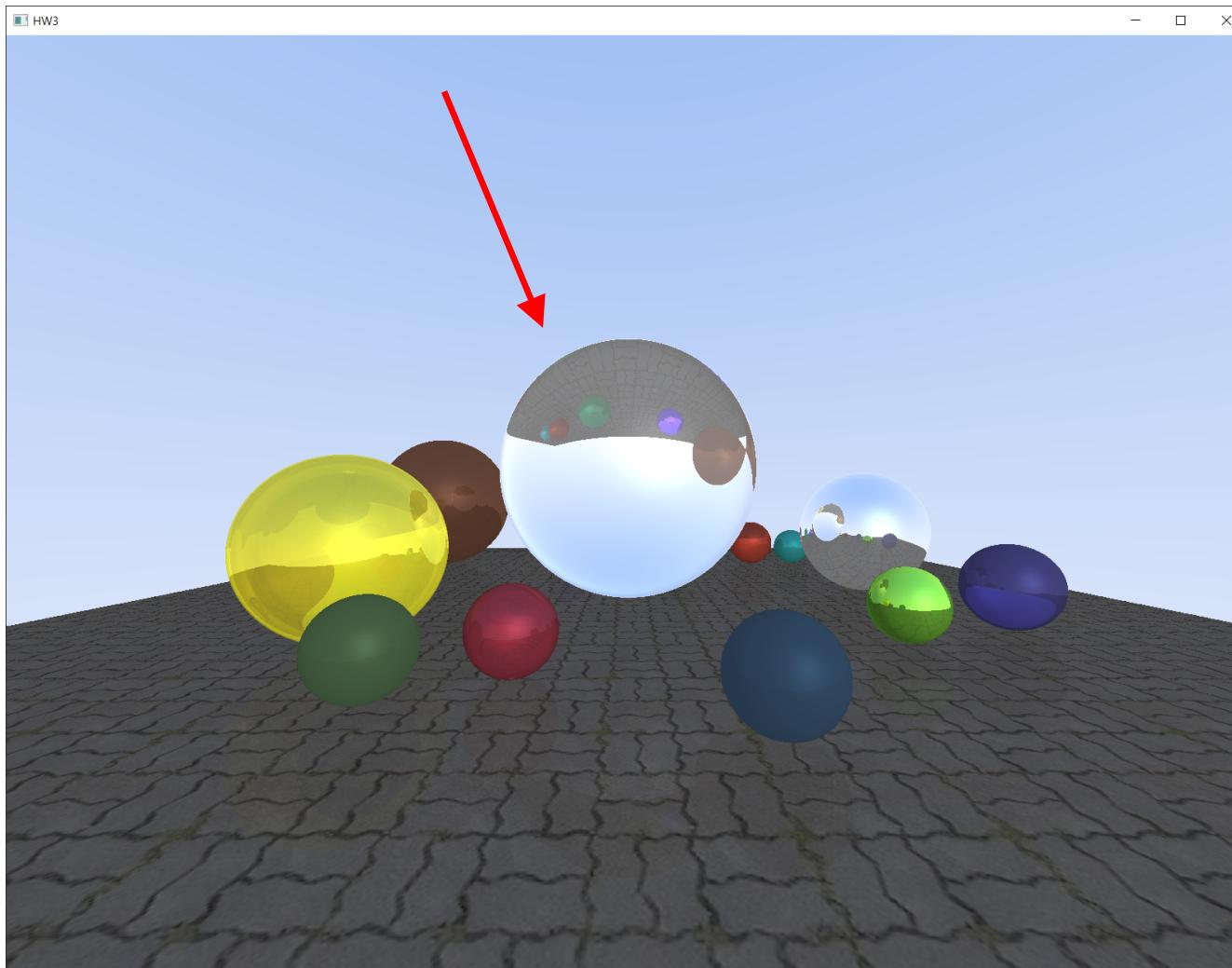
- Snell's Law

- $\eta_1 \sin \theta_I = \eta_2 \sin \theta_t$
- $\eta$ : refractive index,  $\theta$ : angle between light ray and normal.
- Refractive index of empty space is 1.
- Hint: Separate the ray into parallel and normal terms.
- Note:  $\eta_1$  may not always be 1! (use `hit.is_front`)

텅빈 공간의 구조들은 |



# Problem 2

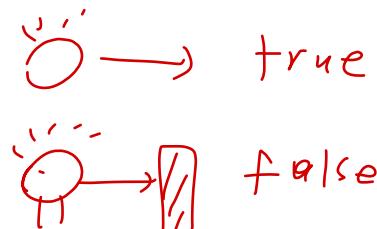


# Problem 3

- bool is\_lighted(vec3 eye)

- Determine if the surface is in shadow.

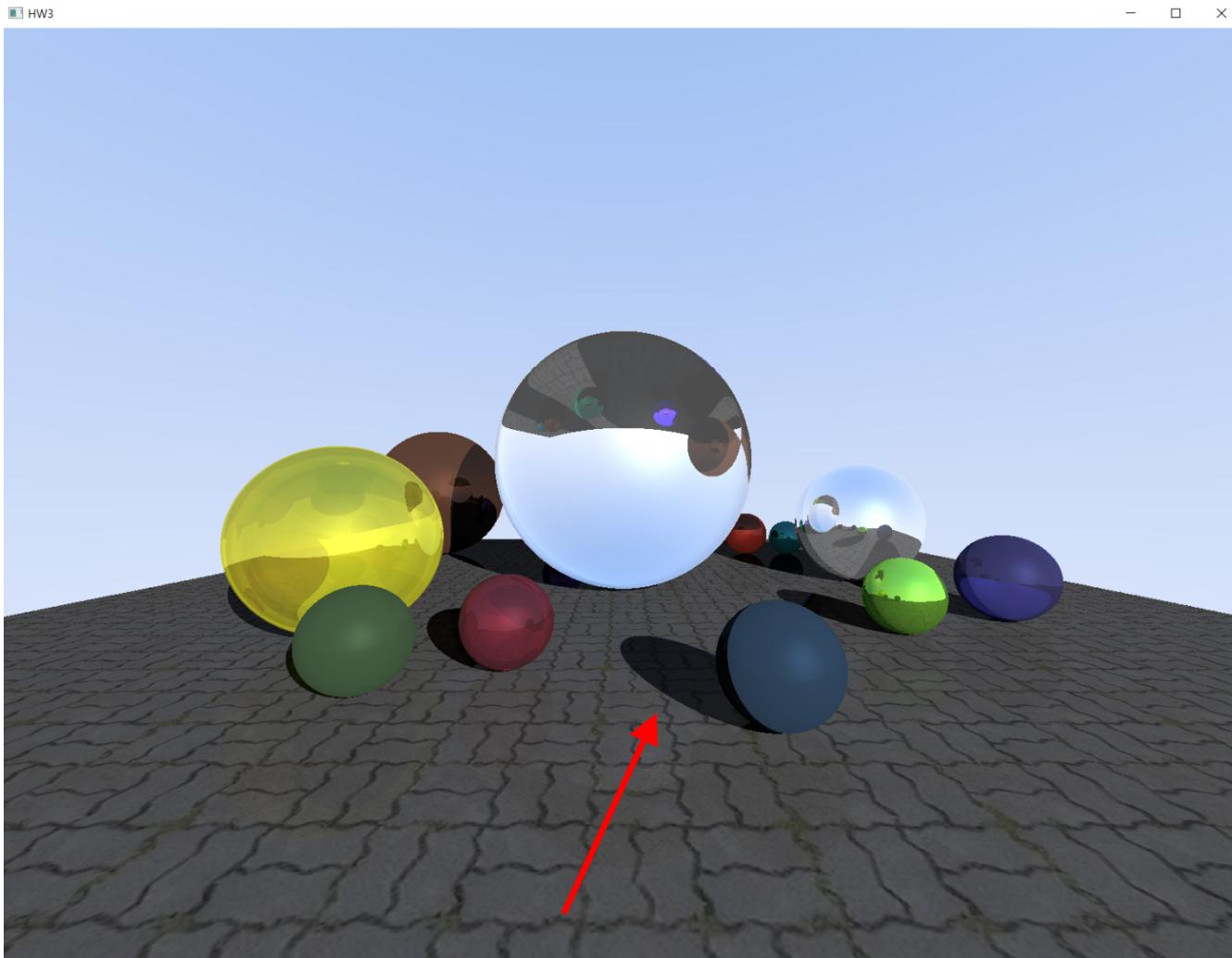
- Use the function get\_closest\_hit to see if the shadow ray hit an object.



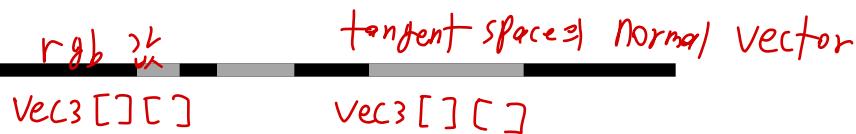
shadow ray가 빛에게 부딪히는지.

light 위치는 전역 변수로 설정되어있음

# Problem 3

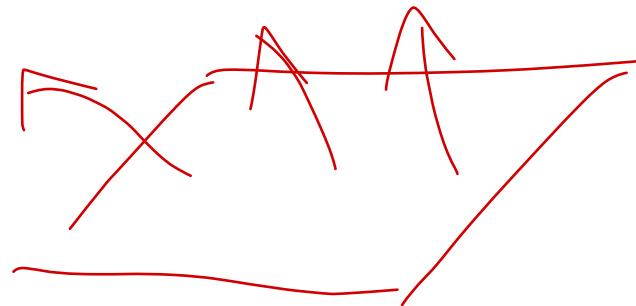


# Problem 4

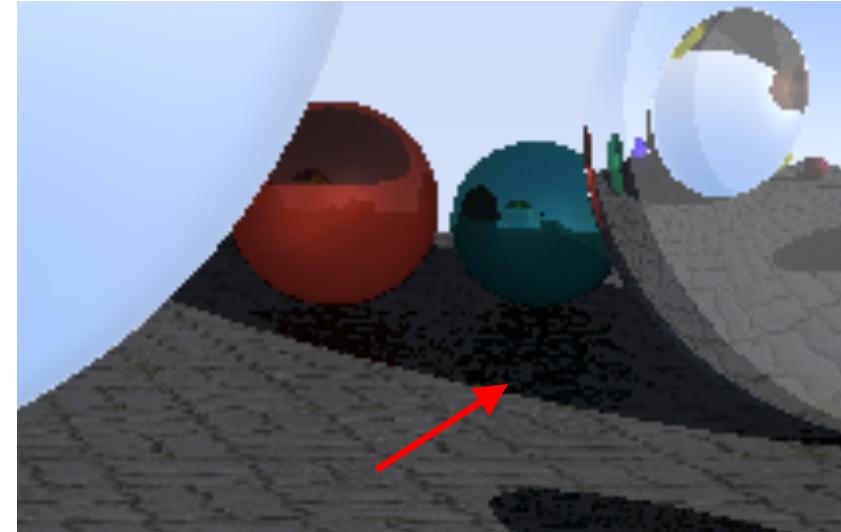
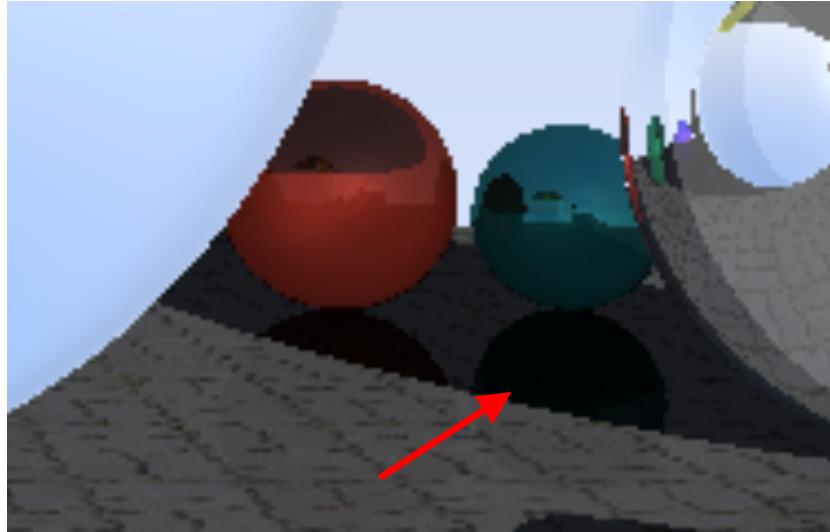


- `void construct_normal_map(img_height, img_normal)`

- Construct a normal map from height map.
- Height map, a grayscale image, is loaded from ./res/height\_map.png to `img_height` as RGB vector. Note that the RGB values are scaled between 0 and 1.
- Save the normal map in `img_normal`.

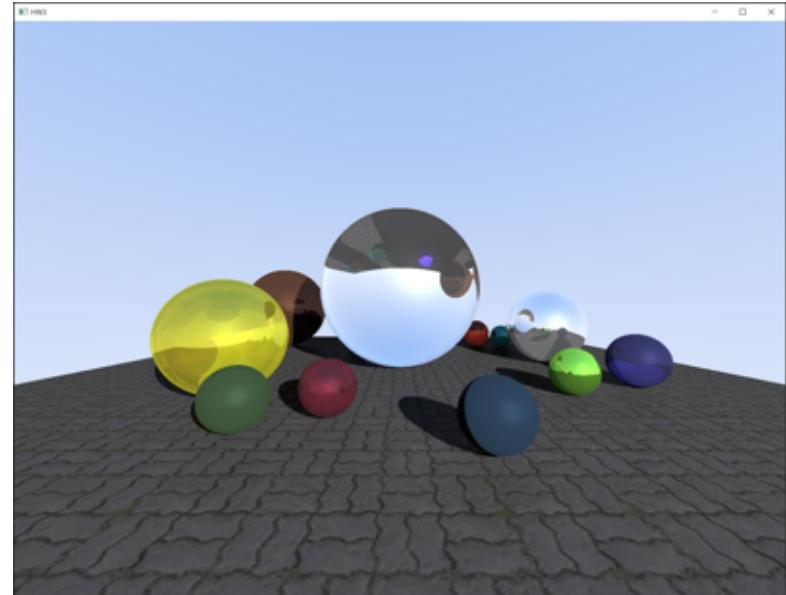
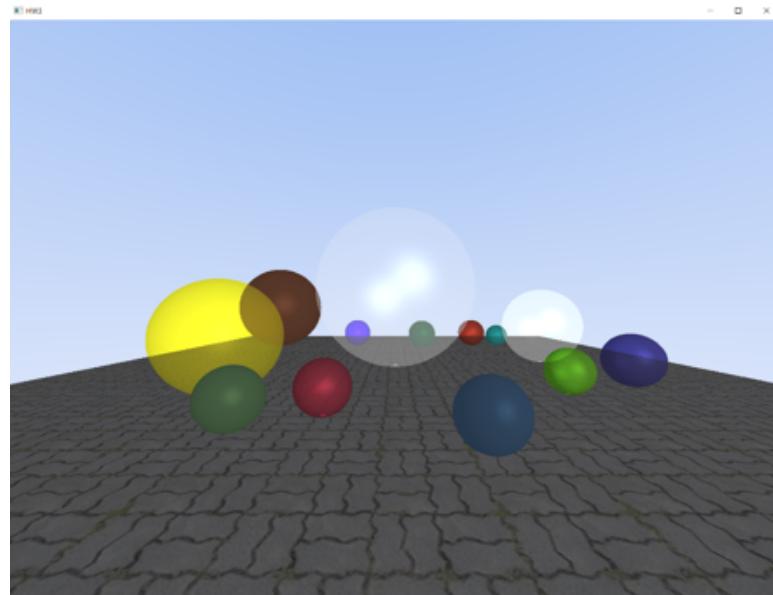


# Problem 4



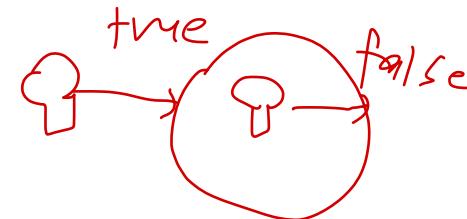
- You can modify `WINDOW_WIDTH` and `WINDOW_HEIGHT` to get a higher resolution image.

# Result



# struct HitData

- `bool is_hit`
  - False if the ray hit nothing
- `vec3 position` light 가 종종 한 위치 (World Space)
  - Position of the closest hit
- `vec3 normal`
  - Normal vector of the hit surface
- `float t`
  - Distance from the ray origin to the hit position
- `bool is_front`
  - True if the ray hit the front face of the object, false when it hit the back face
- `int object_index`
  - ID of the hit object; 0~12 for spheres, 100 for floor



# struct material

- vec3 color
- float diffuse, ambient, specular
  - Variables used in Phong shading
- float n *屈折率*
  - Refractive index of the material (see page 9)
- float reflection, refraction *사용하지 않음*
  - Specular reflectance coefficient and transmission coefficient of the material

Phong lighting (사용하지 않음)

# Submission

---

- Deadline
  - June 13 (Mon) 14:00
- Submission files
  - Rename the file Renderer.cpp to {student\_id}\_{name}.cpp
- **Please follow the submission format!!**
- Submission to Blackboard
- Contact
  - TA email: [2022.CG.TA@gmail.com](mailto:2022.CG.TA@gmail.com)