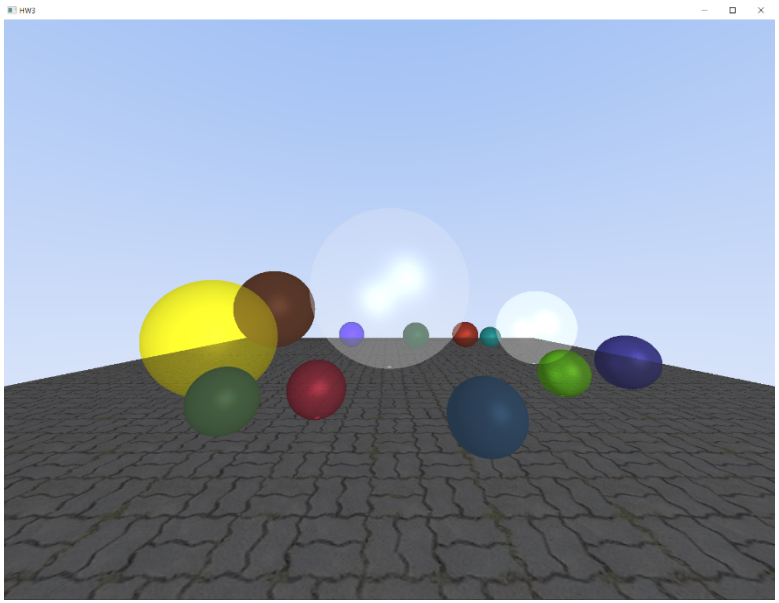
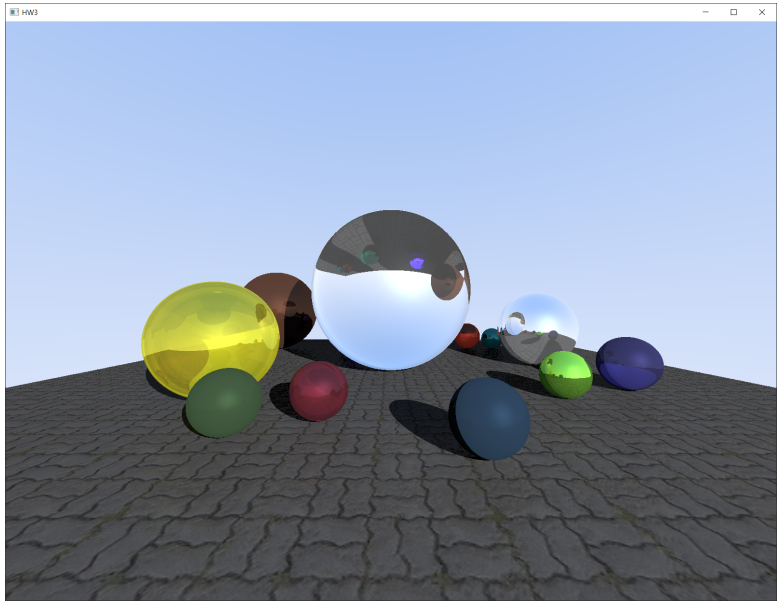
Homework 3

COSE331 Computer Graphics

Goal

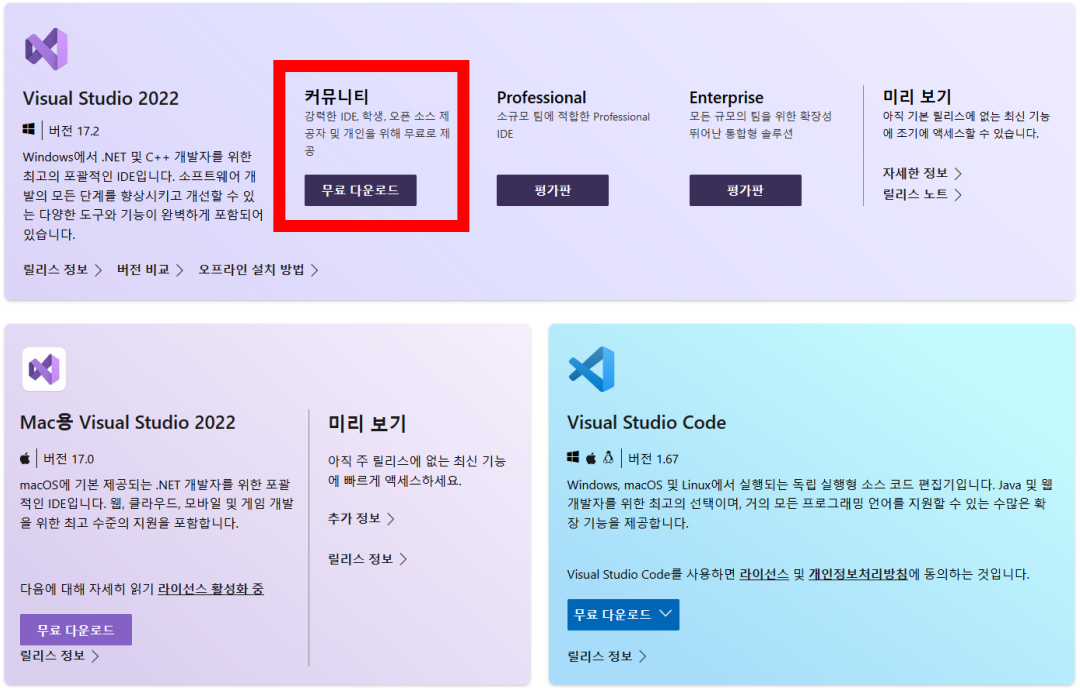
- Render spheres of various materials using ray tracing technique.
  - Calculate reflection ray, refraction ray, shadow ray
- Construct a normal map to shade the floor.





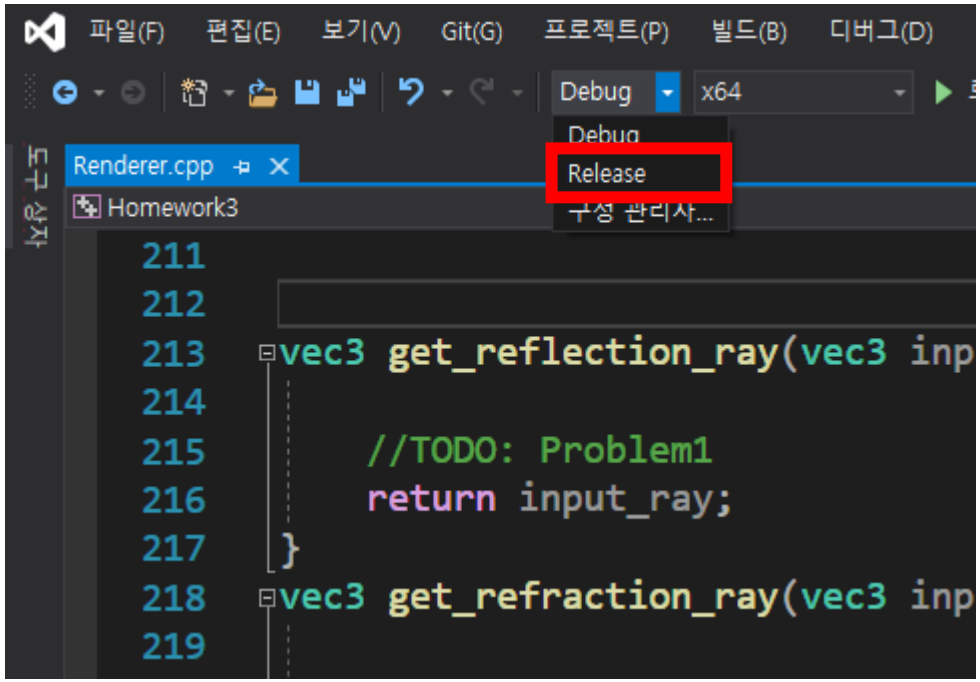
Visual Studio

- 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.



Visual Studio

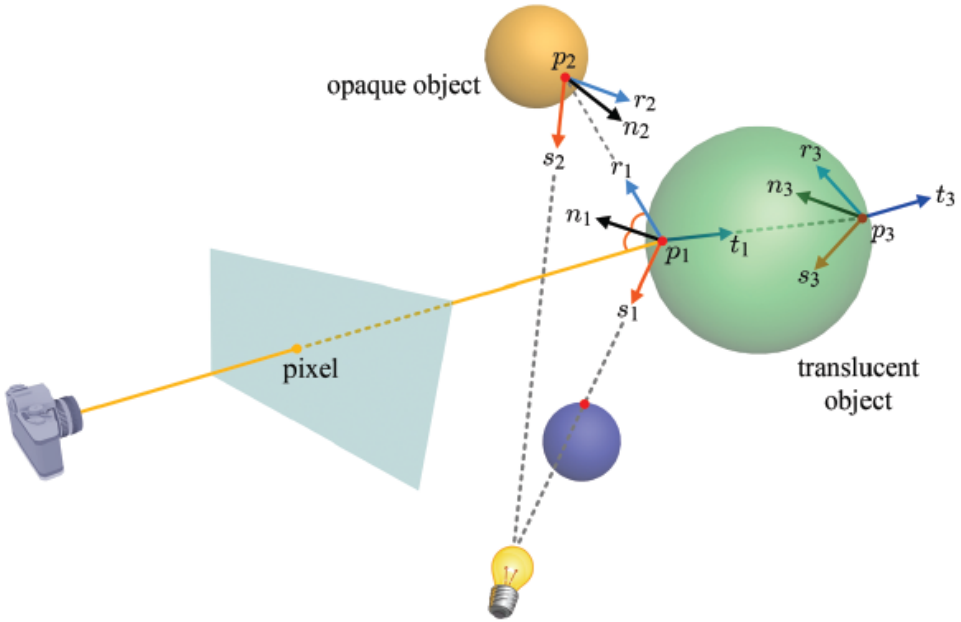
- Open the VS solution file (Homework3.sln)
- Change the build configuration to Release



Ray Tracing

- `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.)



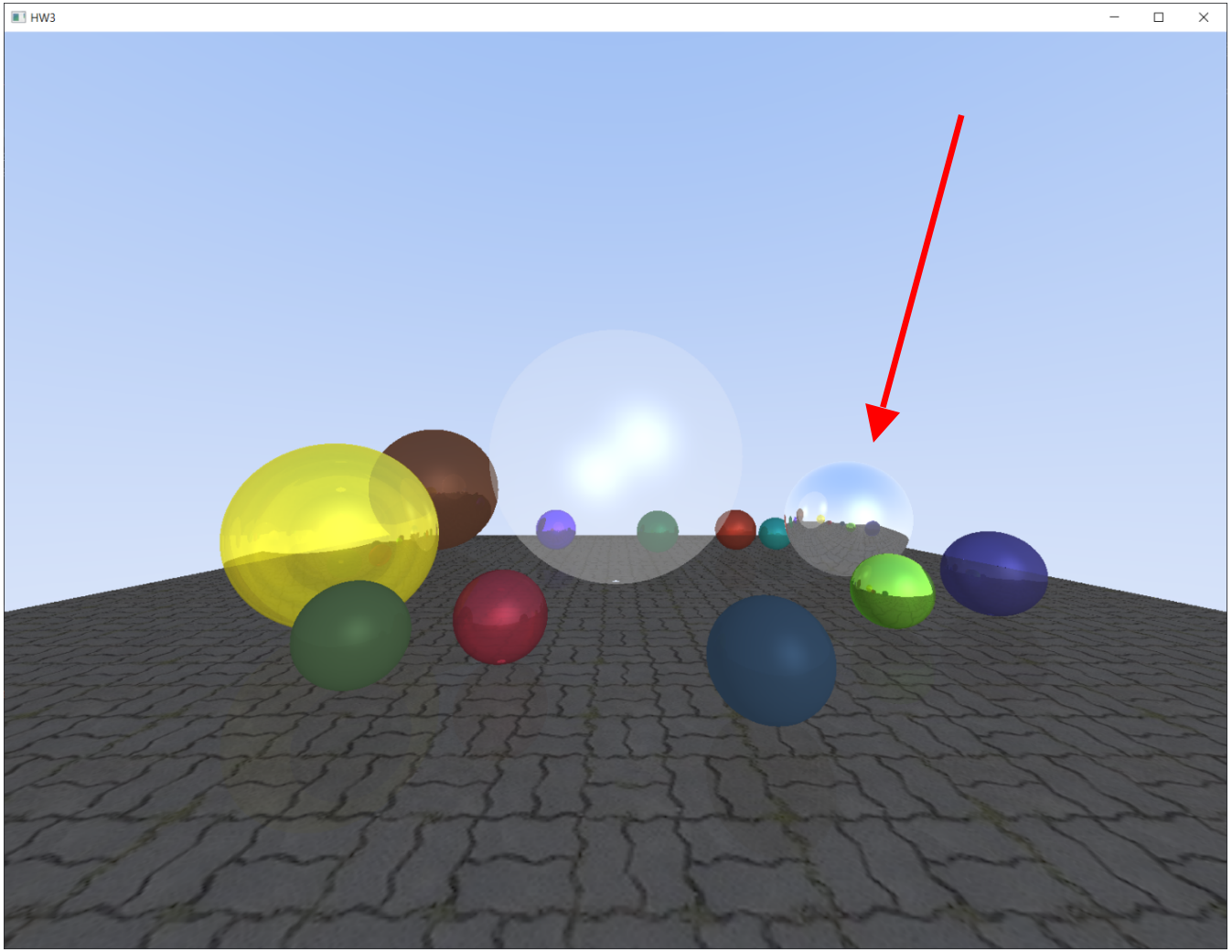
Problems

- 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)`
  - `void construct_normal_map(img_height, img_normal)`

Problem 1

- `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.

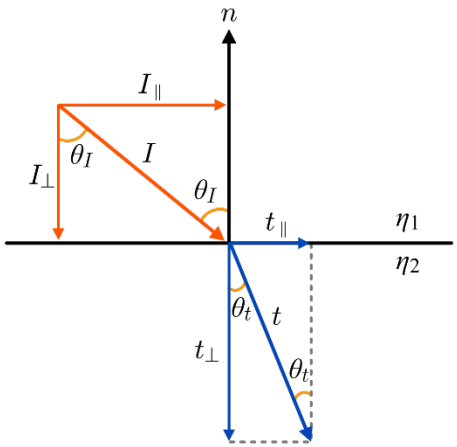
Problem 1



Problem 2

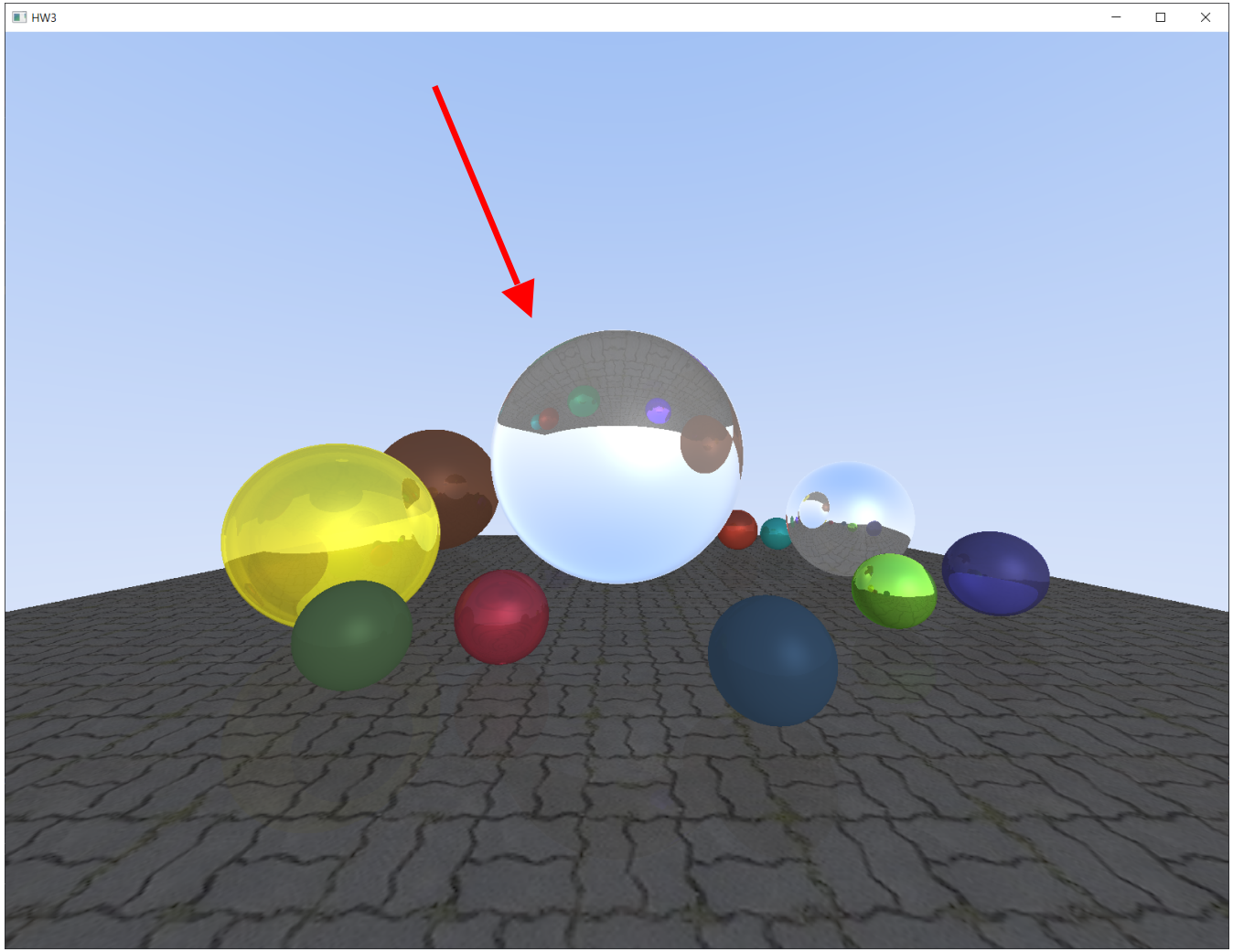
- `vec3 get_refraction_ray(vec3 input_ray, HitData hit, Material m)`
  - Calculate the refraction ray.
  - The returned vector must be normalized.

- Snell's Law
  - $\eta_1 \sin\theta_I = \eta_2 \sin\theta_t$
  - η: refractive index, θ: angle between light ray and normal.
  - Refractive index of empty space is 1.



  - Hint: Separate the ray into parallel and normal terms.
  - Note: η_1 may not always be 1! (use `hit.is_front`)

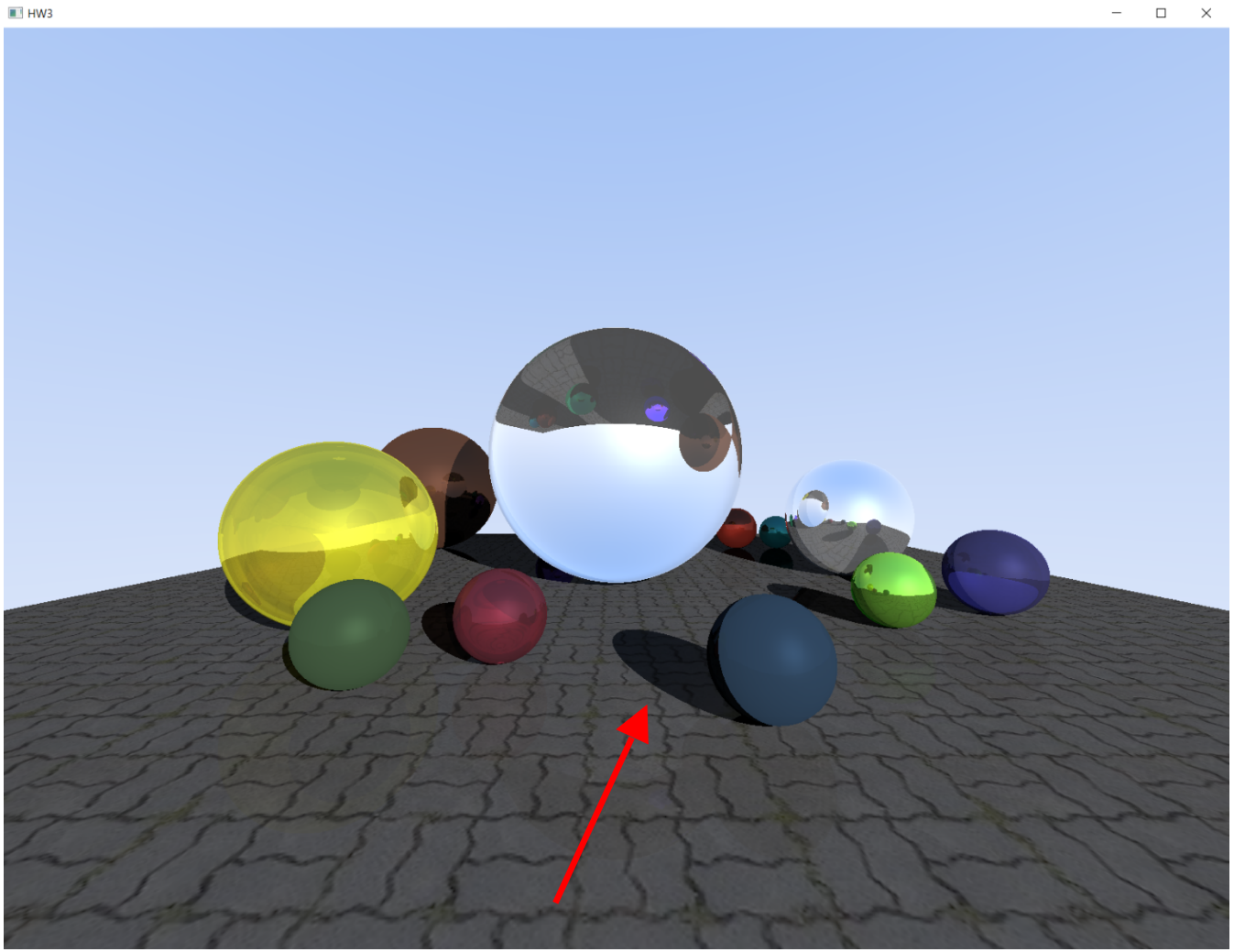
Problem 2



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.

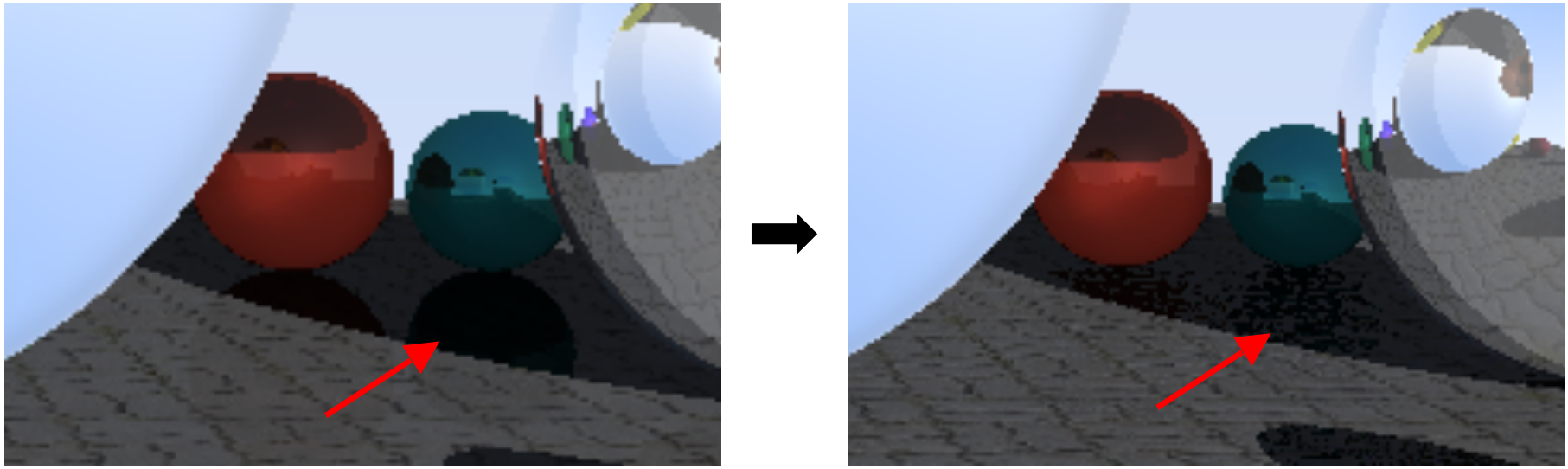
Problem 3



Problem 4

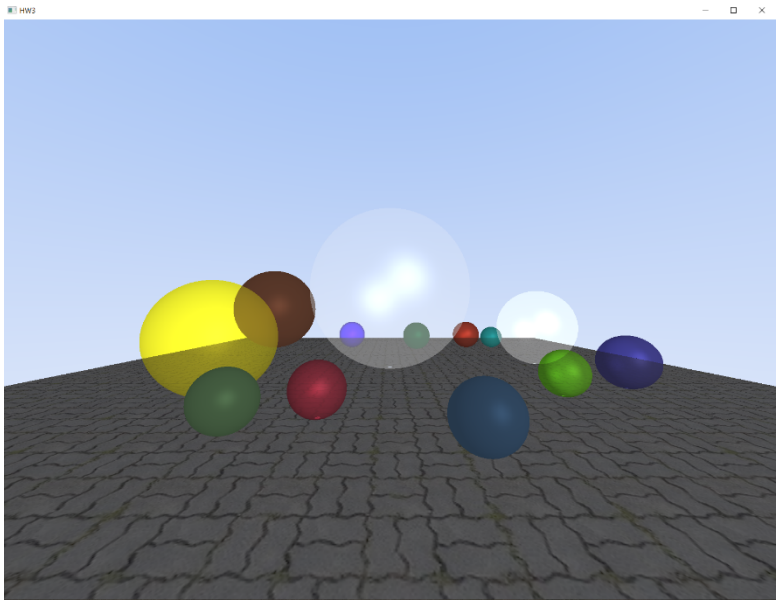
- `void construct_normal_map(img_height, img_normal)`
  - Construct a normal map from height map.
  - Height map, a grayscale image, is loaded from ./res/height_map.png to `img_height` as RGB vector. Note that the RGB values are scaled between 0 and 1.
  - Save the normal map in `img_normal`.

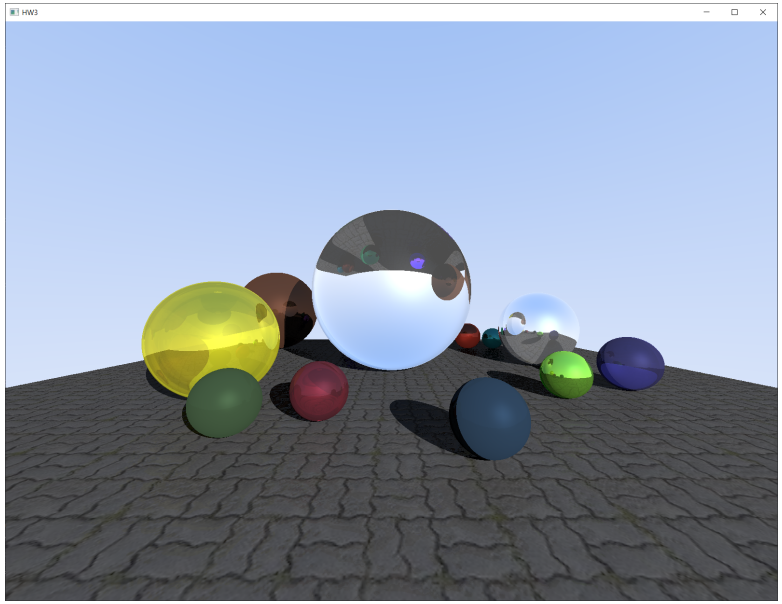
Problem 4



- 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`
  - 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`
  - Refractive index of the material (see page 9)
- `float reflection, refraction`
  - Specular reflectance coefficient and transmission coefficient of the material

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