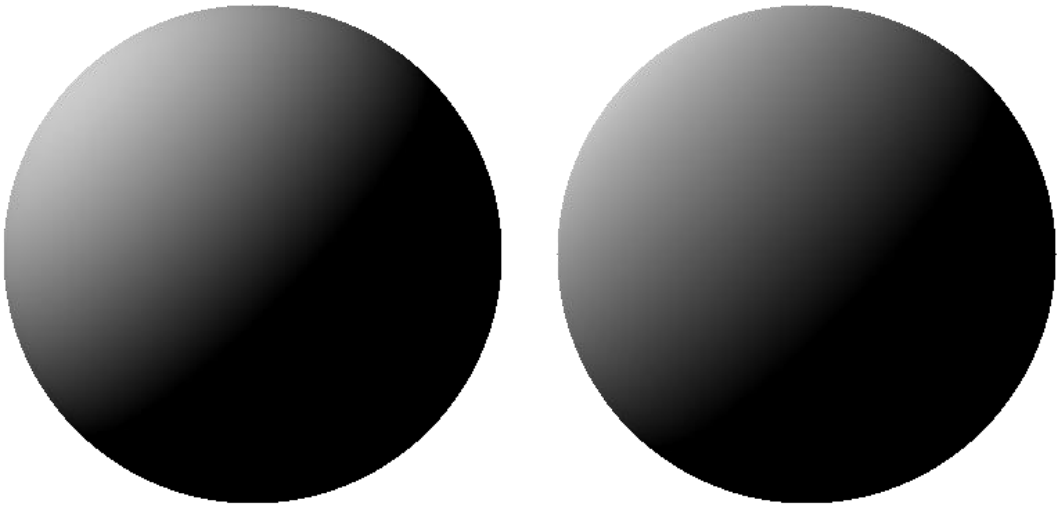
Homework1

Part1

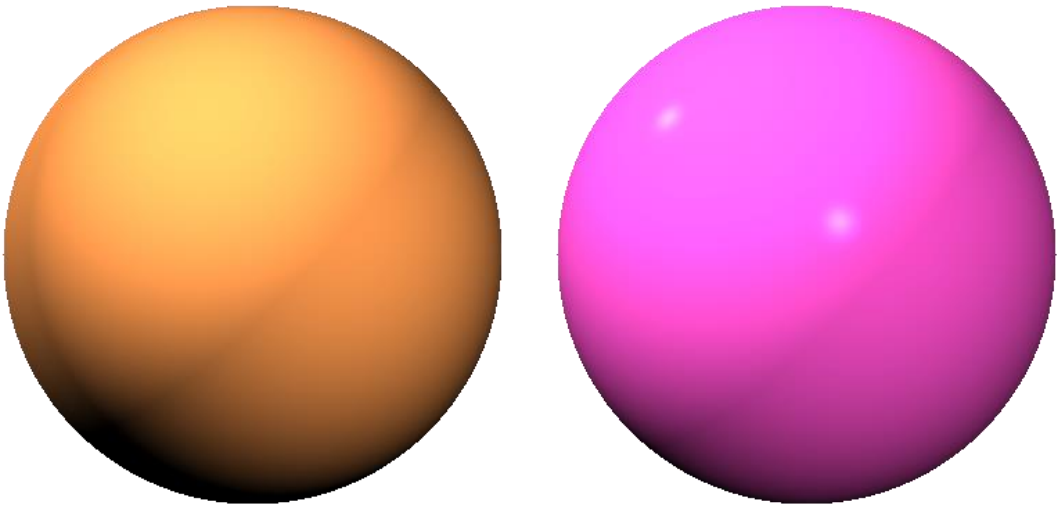
In this part, I modified the given code to support multiple spheres in general position. To achieve that, I added the center and radius of sphere. Since we already know the ray direction, the intersect is easy to calculate. As the Picture1 shows, the light source is on the left of spheres and uses only Lambertian Shading.



Picture1 Ray Tracing Spheres

Part2

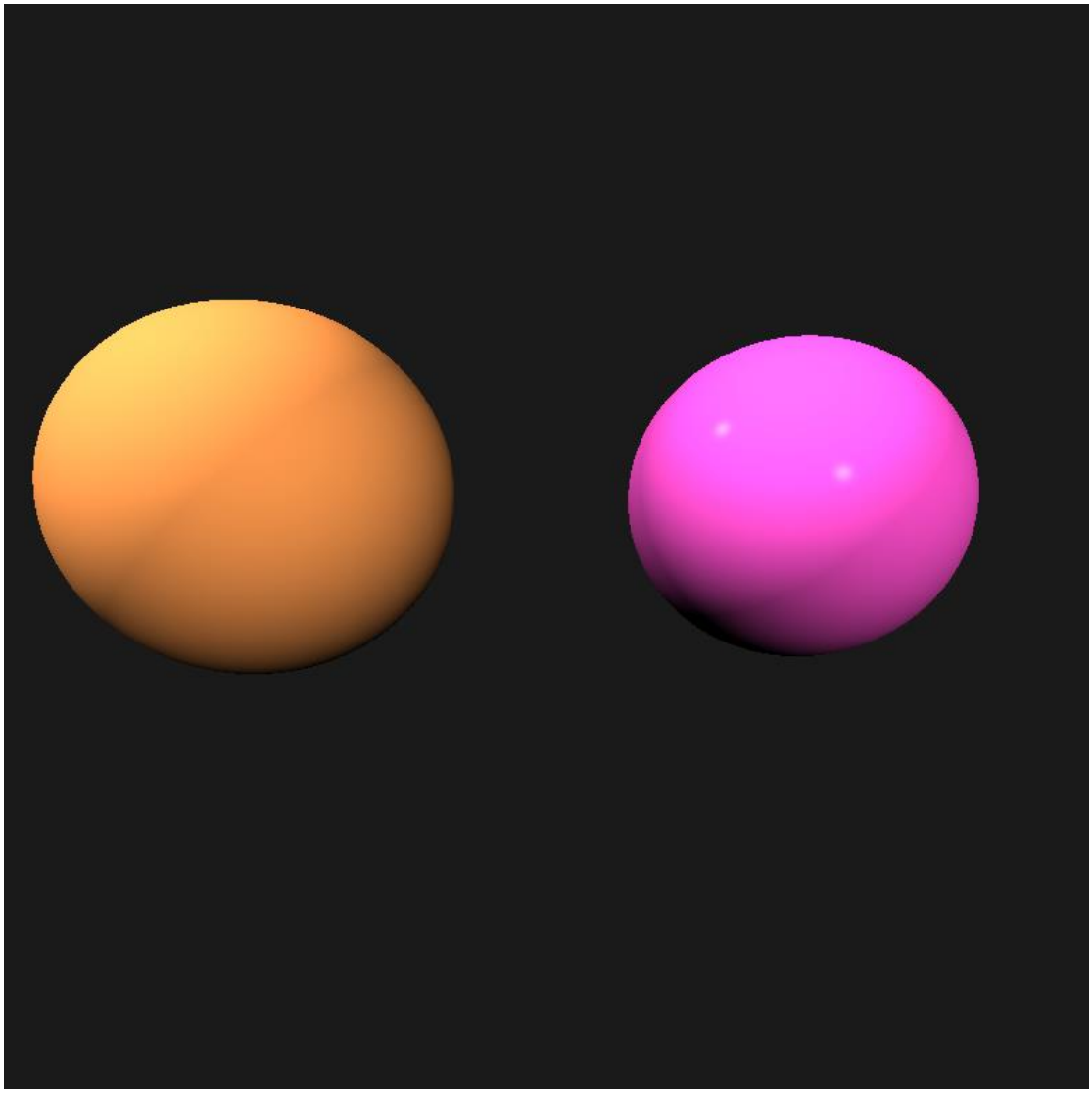
This part adds a shading to make the picture realistic. The left sphere is set to be purely diffuse while the other one is specular. The highlight of the sphere on the right is the specular effect.



Picture2 Shading

Part3

This part uses perspective projection rather than orthographic. The different between them is that the ray direction is no longer parallel with each other in the perspective projection, thus the object looks bigger if getting closer to the view point. As the Picture3 shows, the right sphere is father than the left sphere, so it looks smaller than the left sphere.

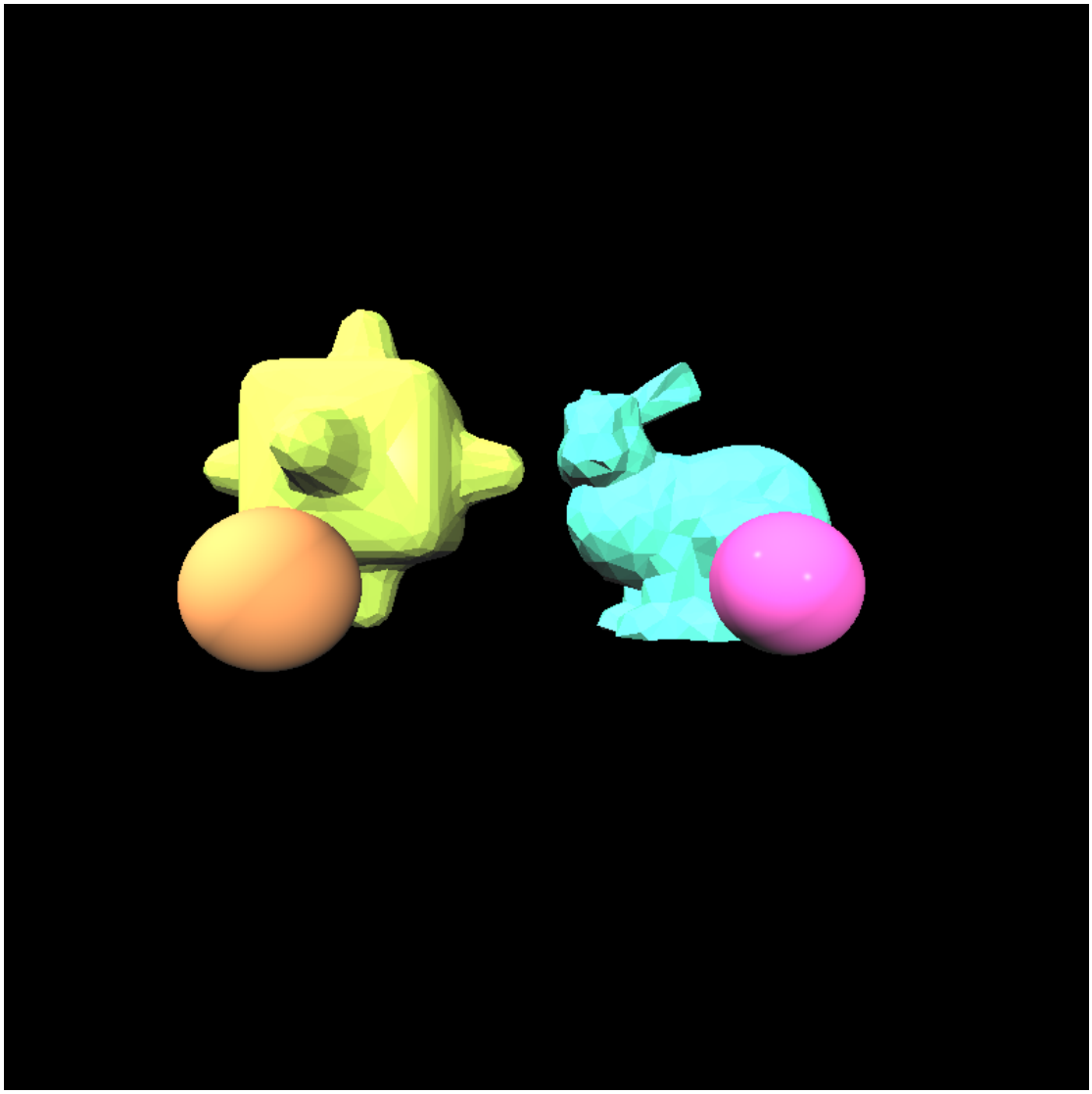


Picture3 Perspective Projection

Part4

This part loads two meshes in off format and adds them to the previous part. These two objects look like a cube and a rabbit respectively. To achieve that, I set two Eigen arrays V and F to store the face information and vertice information in memory, and then I need to calculate each intersection between ray and triangle and sphere. Most importantly, remember to open the release mode to accelerate the speed of execution, otherwise it will be very very slow to run the program.

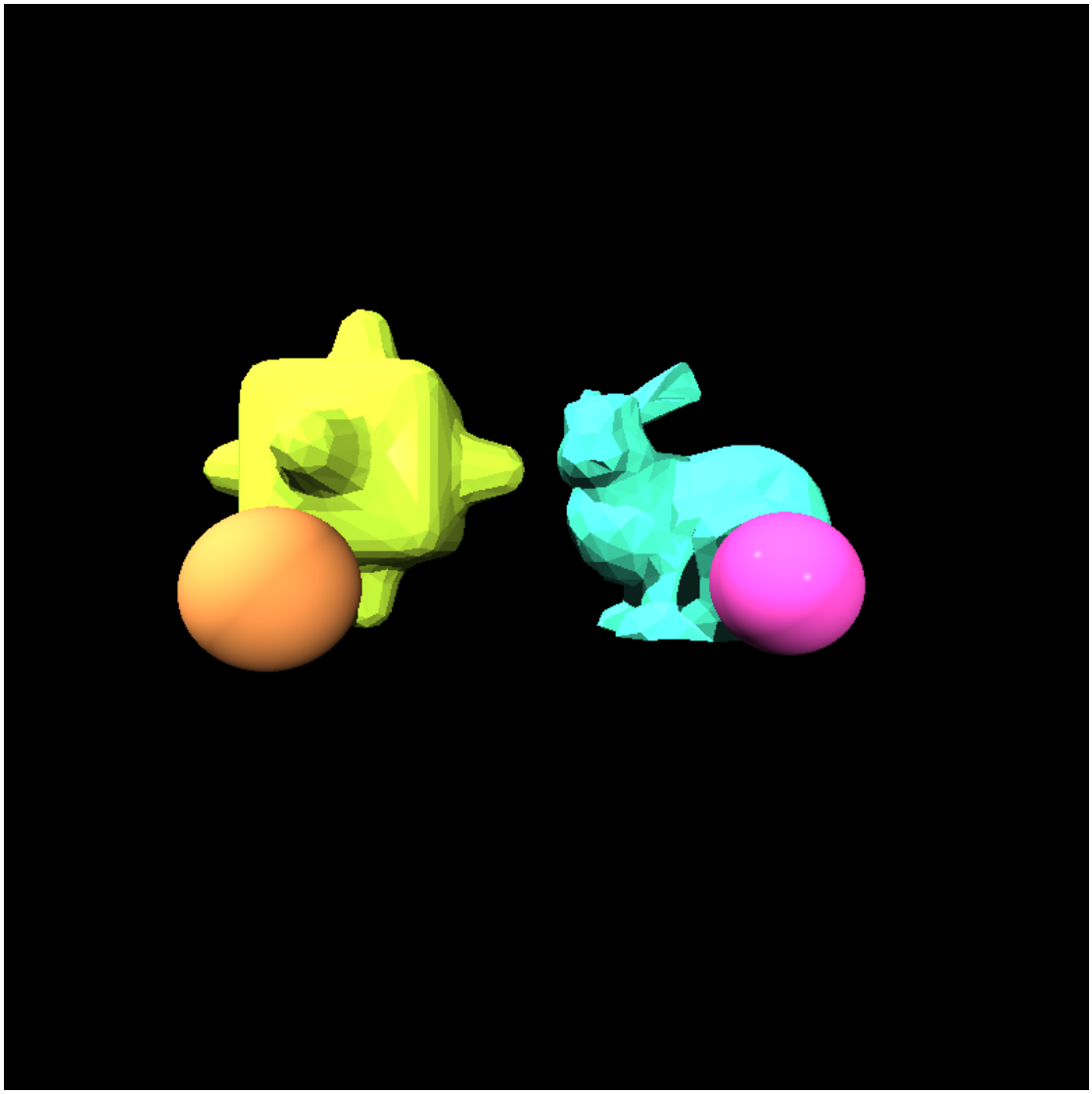
```
cmake -DCMAKE_BUILD_TYPE=Release ../
```



Picture4 Load meshes

Part5

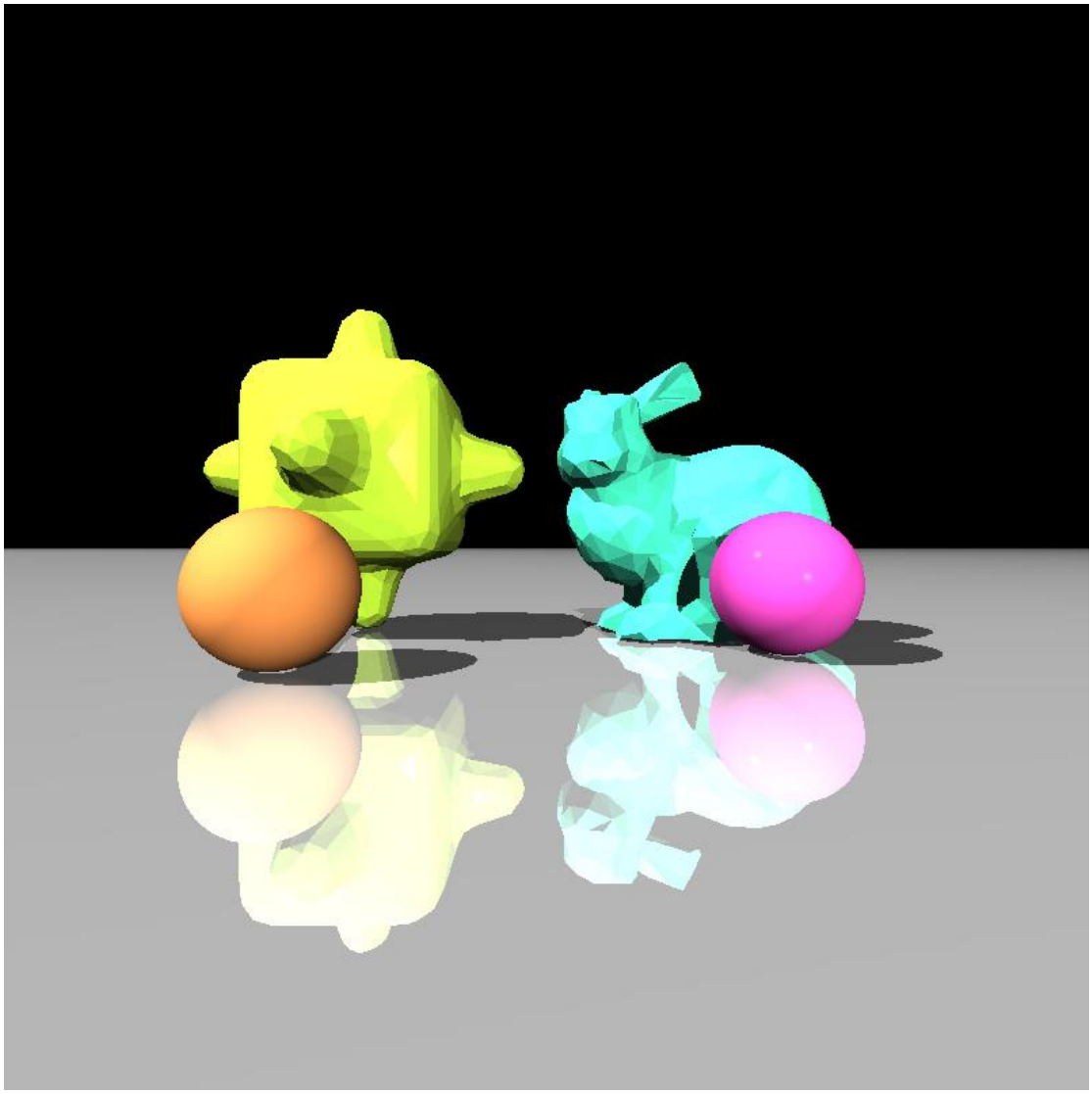
This part adds shadow to the previous picture. To achieve that, for each pixel, I need to judge whether the path to light source from intersection is blocked. As Picture5 shows, the right sphere gives a shadow to the rabbit because part of it was blocked by the sphere.



Picture5 Shadows

Part6

This part adds reflection to the previous picture, thus a floor is needed to reflect the light. For each pixel, calculate the intersection and then render it with shadow. After that, if the ray hit floor, then calculate the reflect light and judge whether it hit other objects again. If it does, then repeat the procedure of calculate the intersection and then render it with shadow.



Picture6 Reflection

Part7

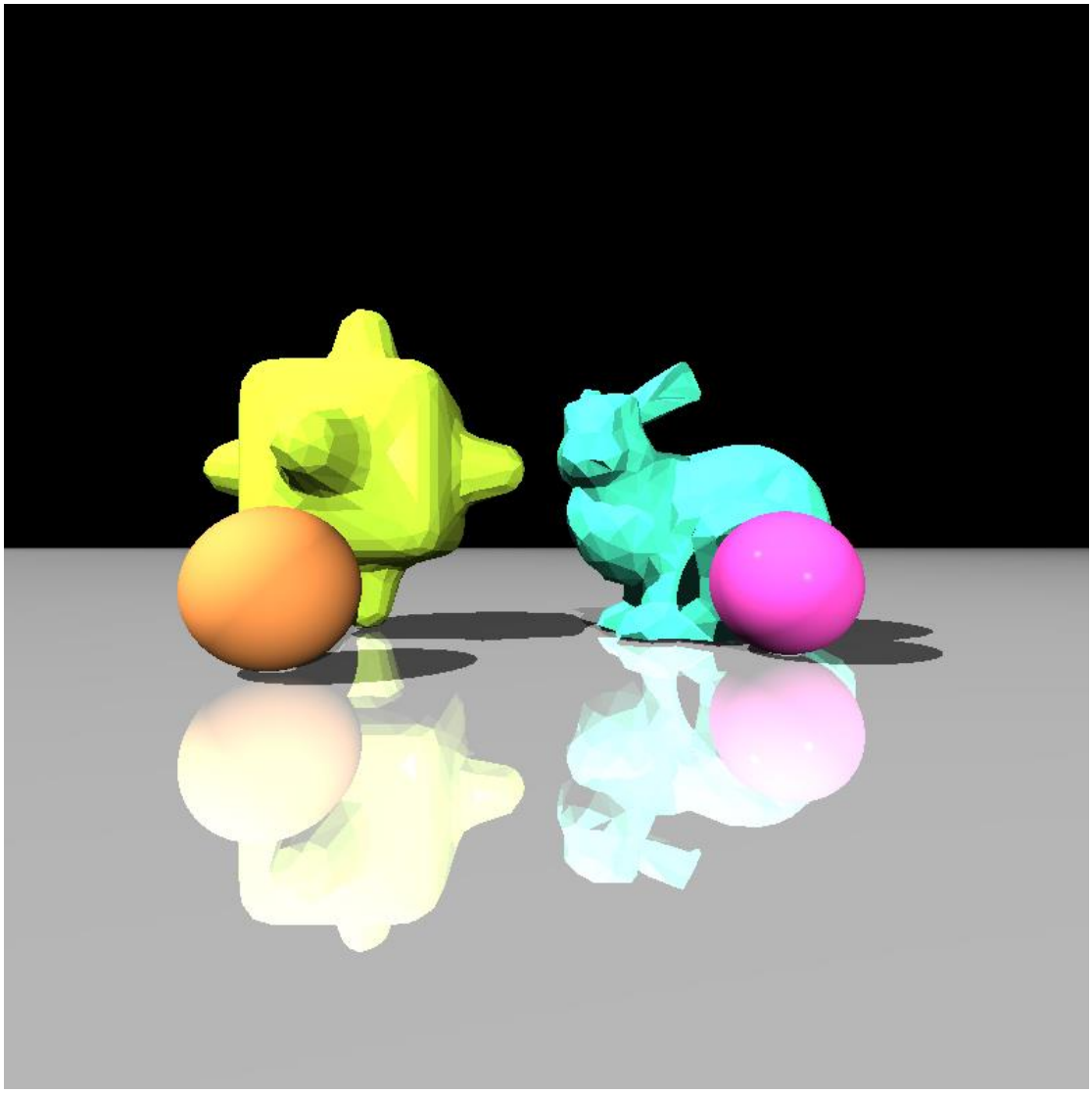
This part adds parallel computing to each pixels with Intel TBB. In my program, for each columns in the picture, I use the function Parallel_for(), and then they can be executed independently by the different cores in CPU.

To use tbb successfully, first clone the Tbb from Github, and then make install the lib to your system, in order to use the libtbb.so, we need to modify the cmake file. Add the following commands to your cmake file.

```
#Link library, make sure tbb installed in /usr/lib
set(LINK_DIR /usr/lib)
link_directories(${LINK_DIR})
link_libraries(tbb)
#Link a third-party library to an executable file
target_link_libraries(${PROJECT_NAME}_bin tbb)
```

And then go to the build file, use the command

```
cmake -DCMAKE_BUILD_TYPE=Release ../
```



Part8

This part renders multiple frames while moving the position of camera and of light. I set a 3*3 rotation matrix, and multiply it by the view point and screen point so that it seems like the camera is moving in 3D coordinate system. I add a gif into the project directory.