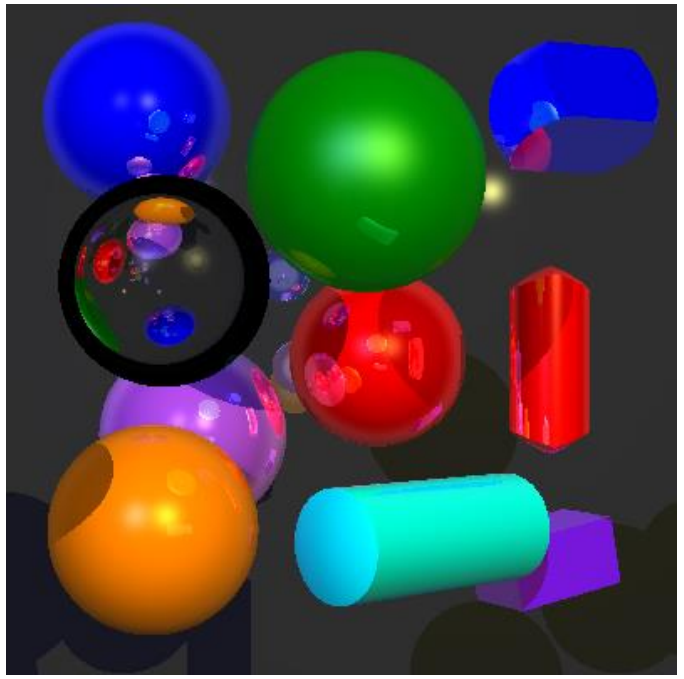
Lab Sessions

Photorealistic Rendering (Advanced Computer Graphics)

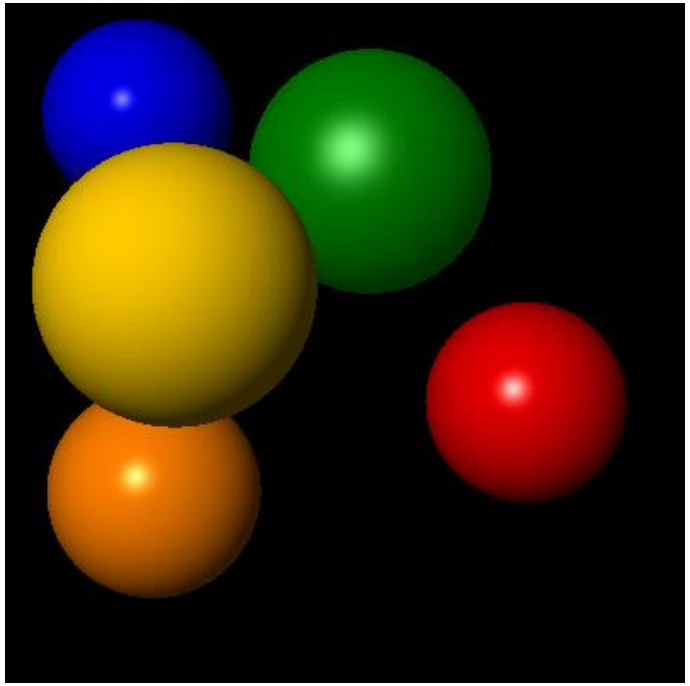
Tobias Isenberg

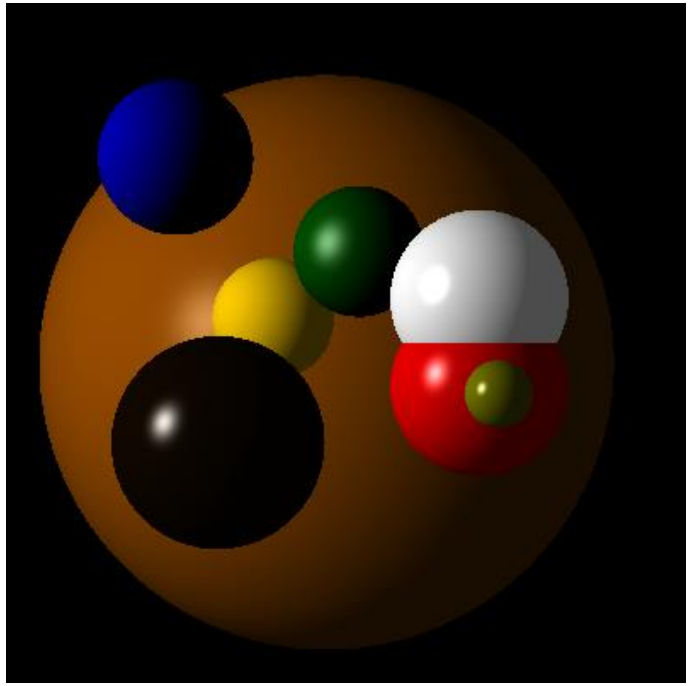
informatics mathematics Inria



Thanks for sending the assignments early

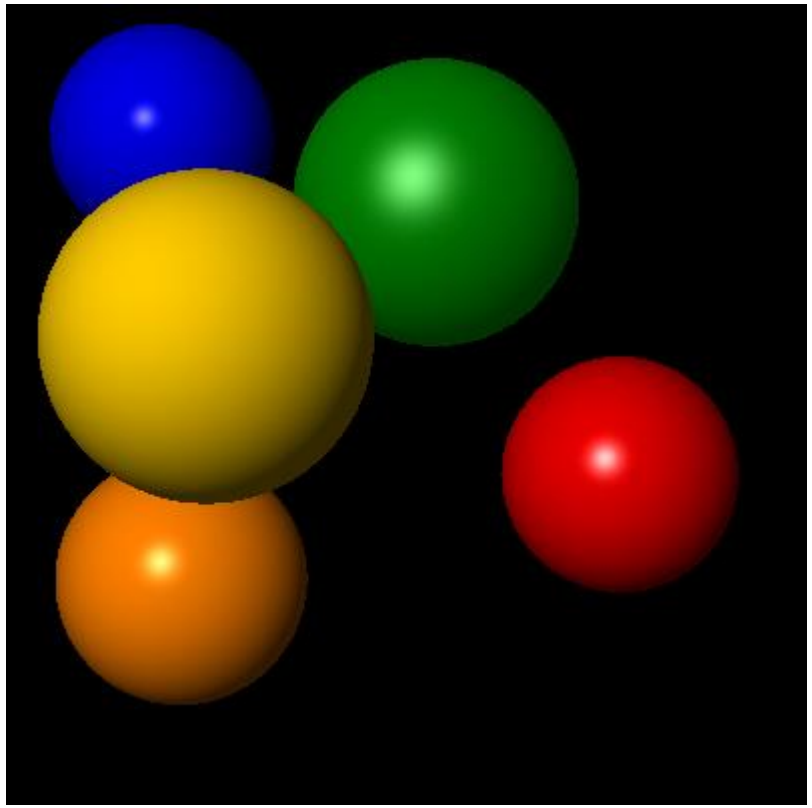
- most with correct results
- got assignments from 3/4 groups, all followed instructions!
- the `runme.bat` should not overwrite the images in `results/`

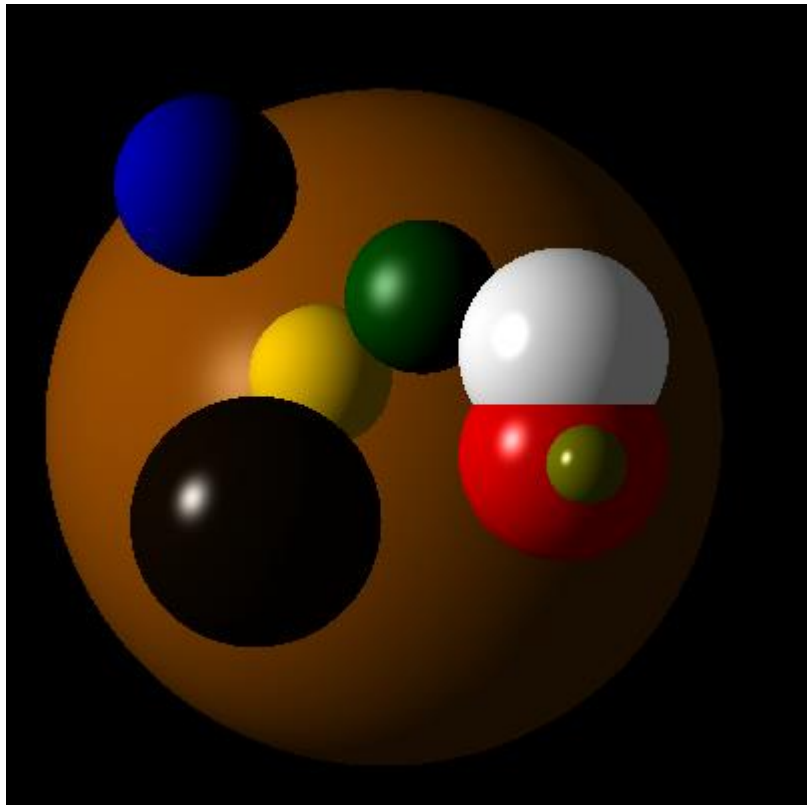




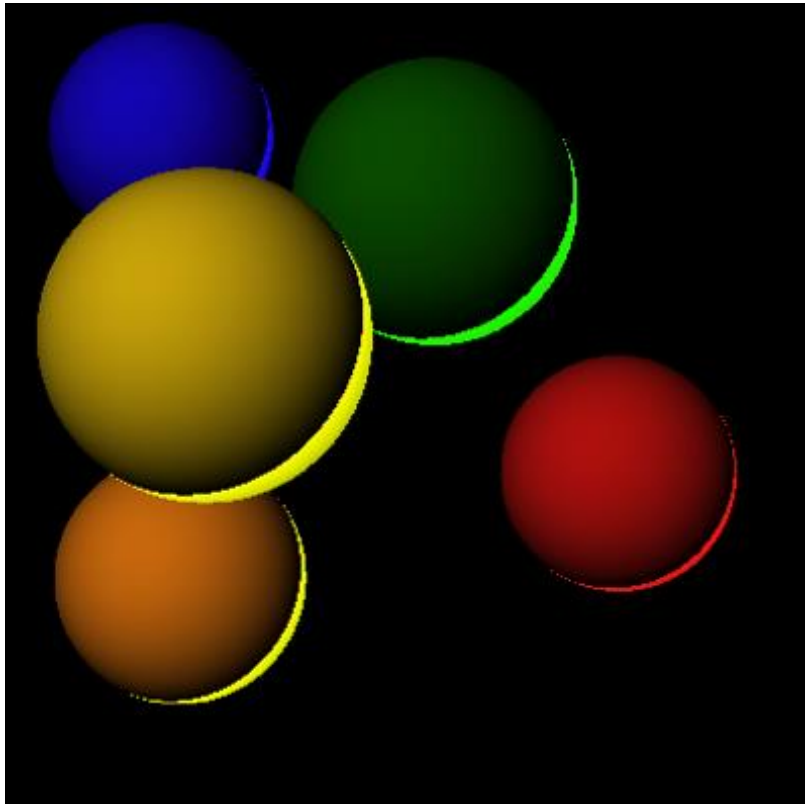
Problems

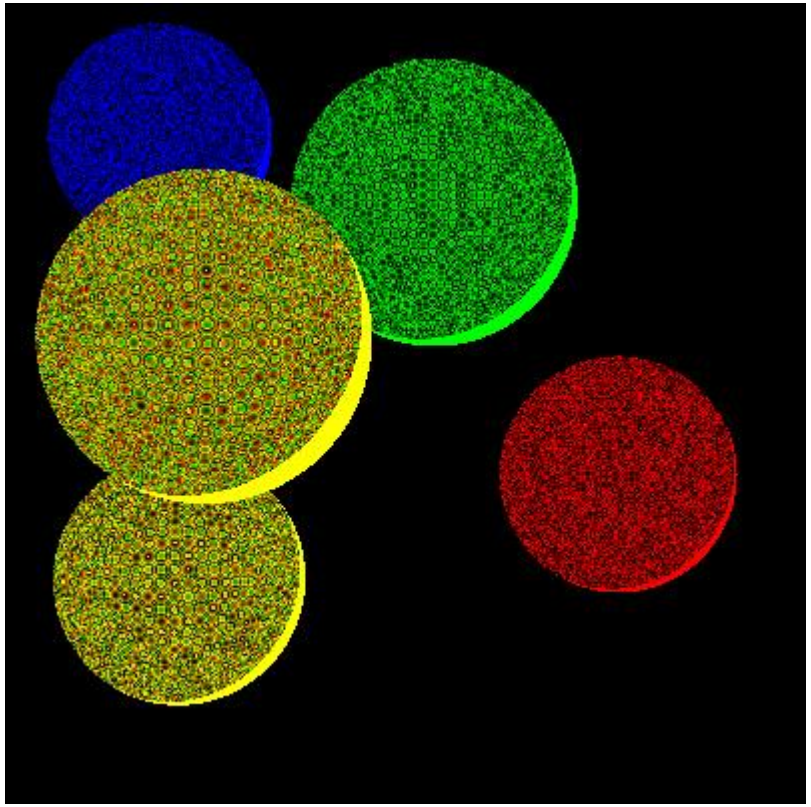
- specular reflection does not use object color, only light color



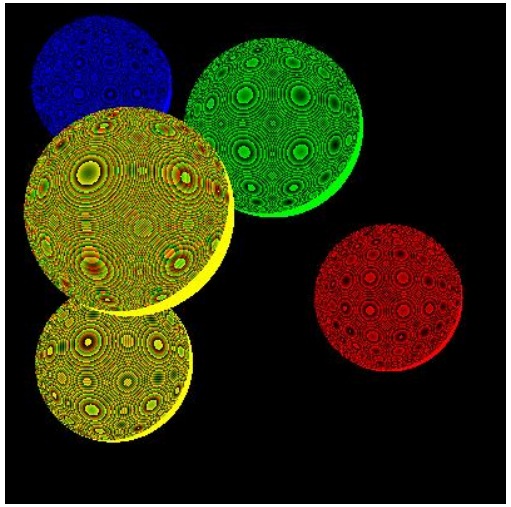
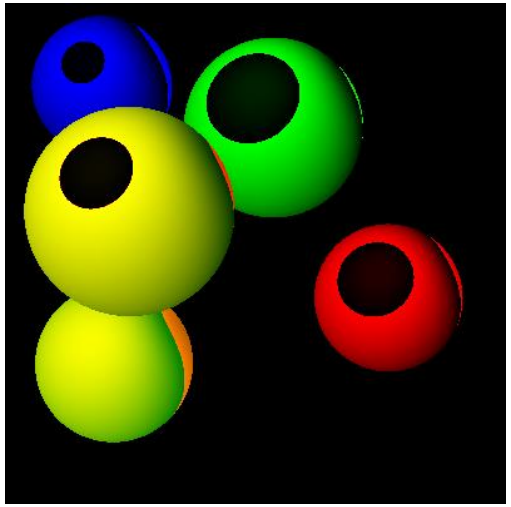
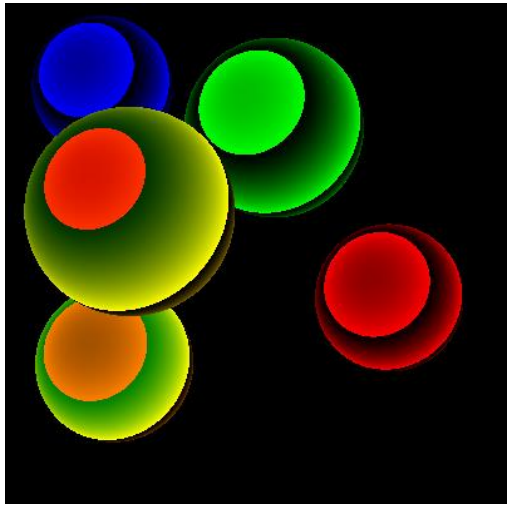
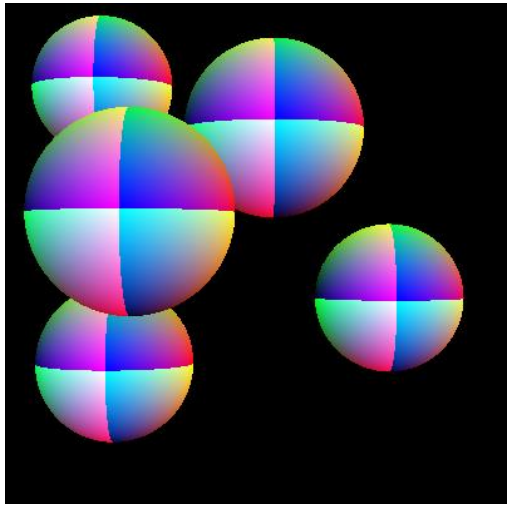
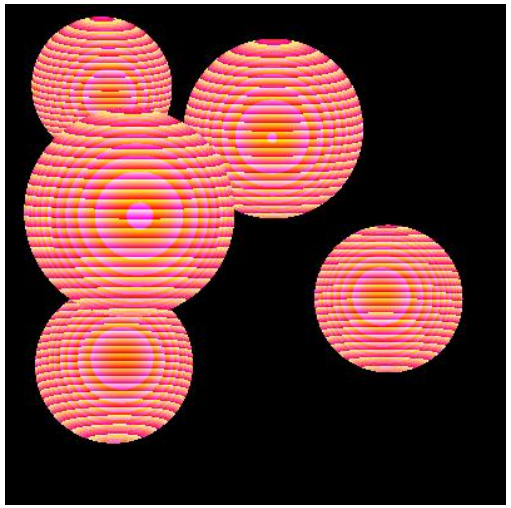
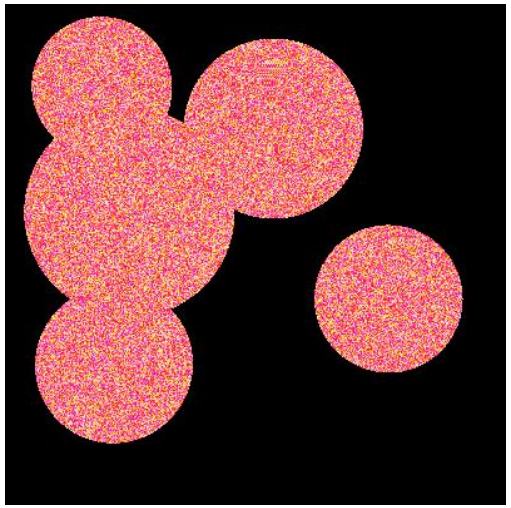
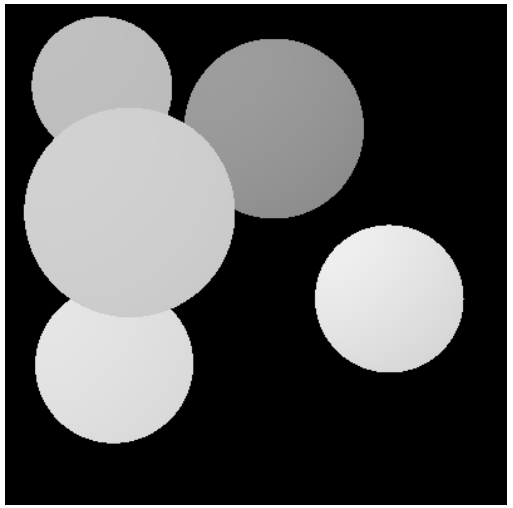
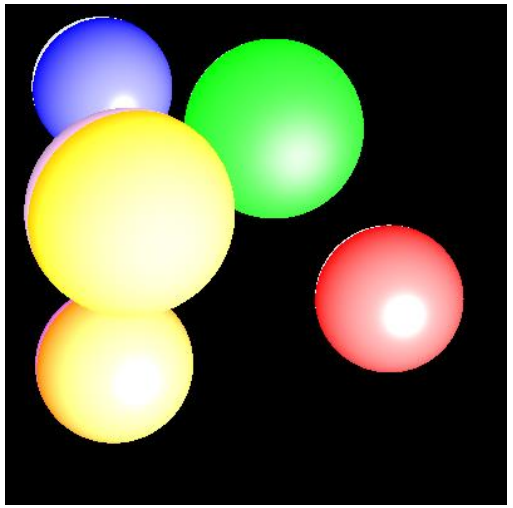


First assignment reminder: Errors

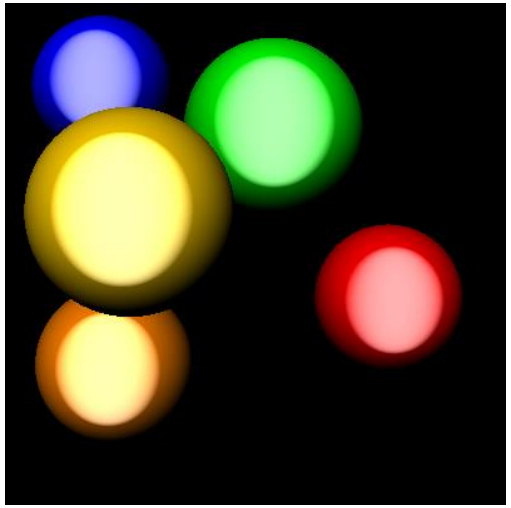
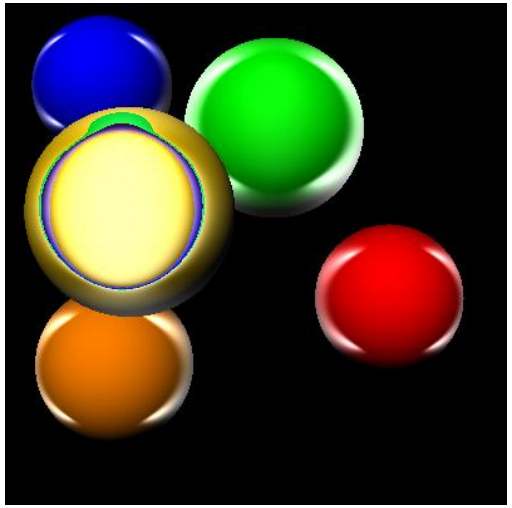
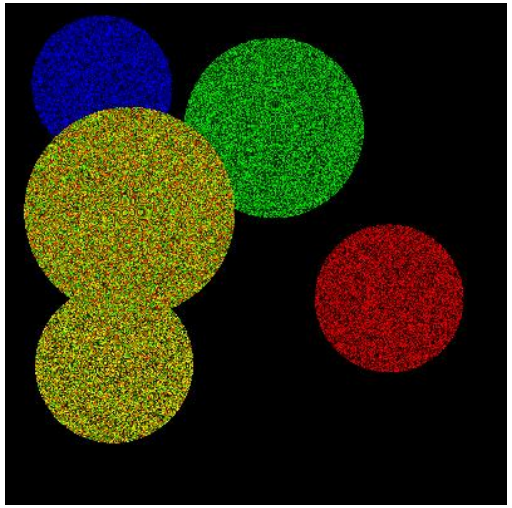
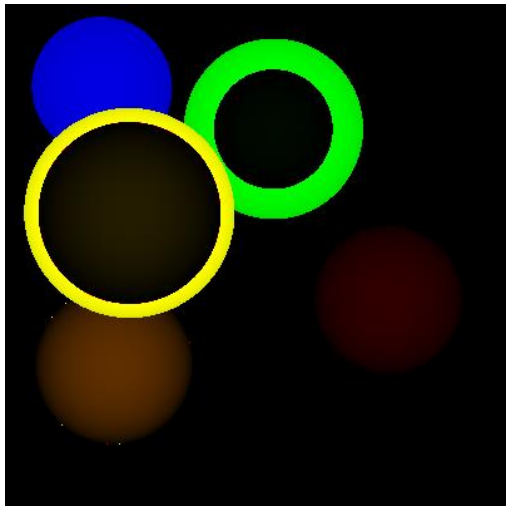
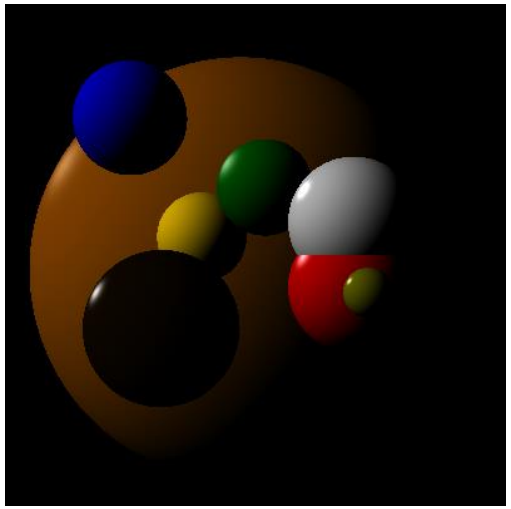
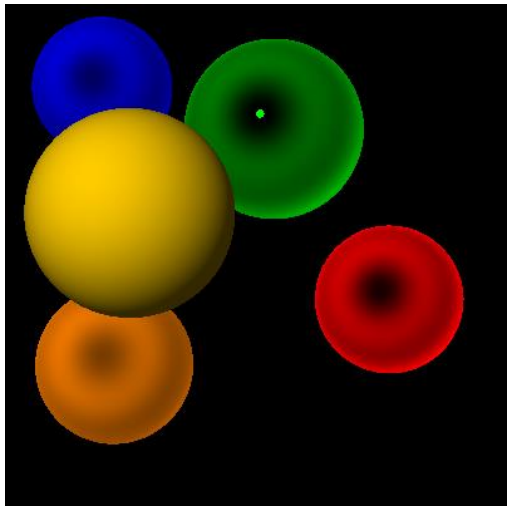
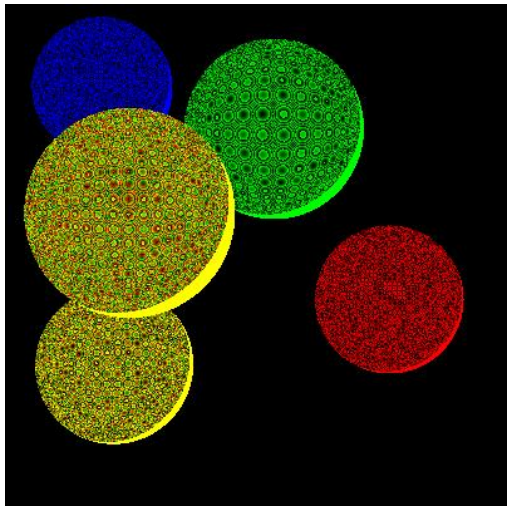




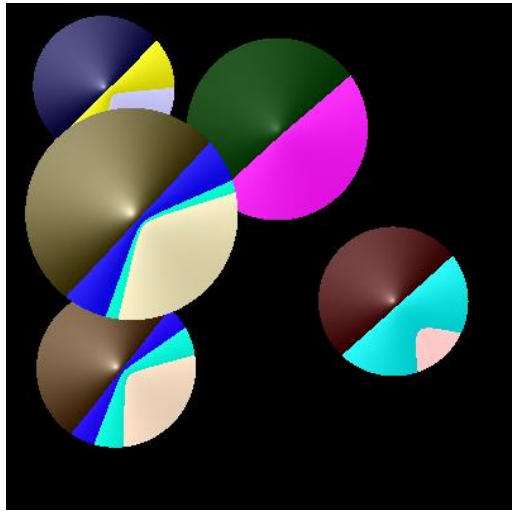
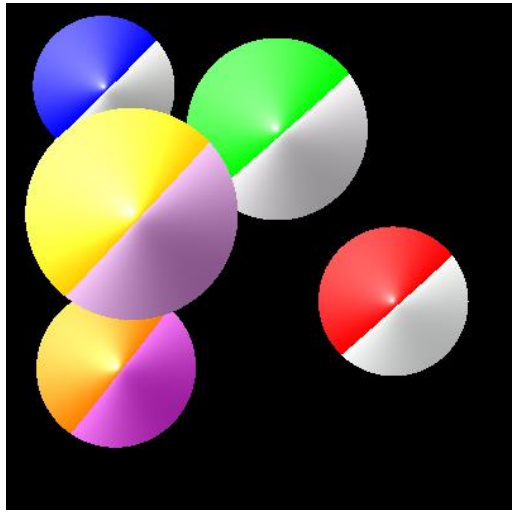
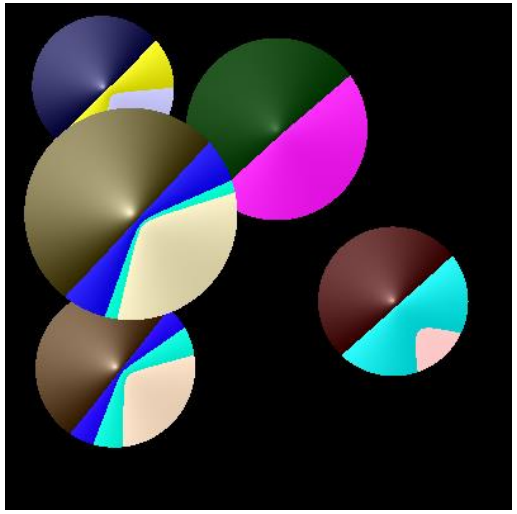
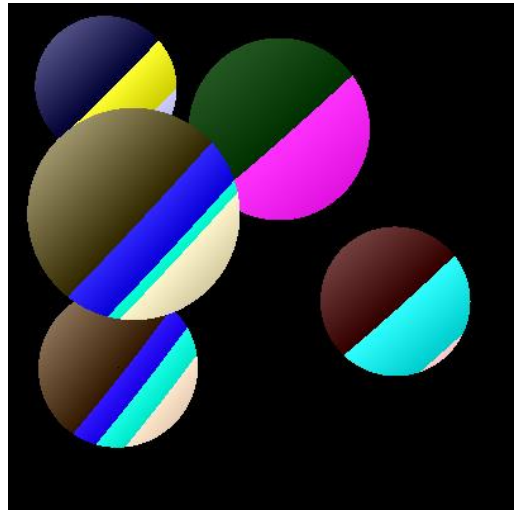
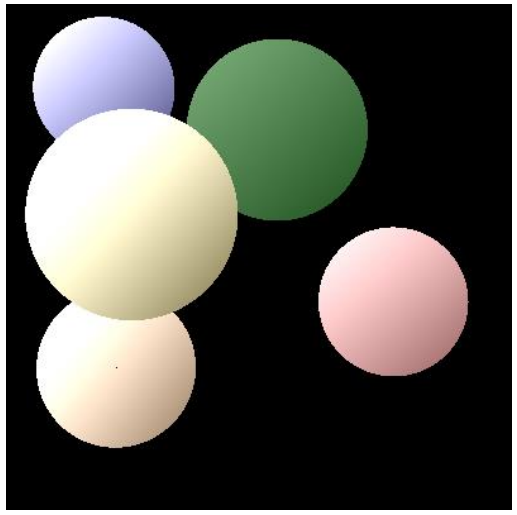
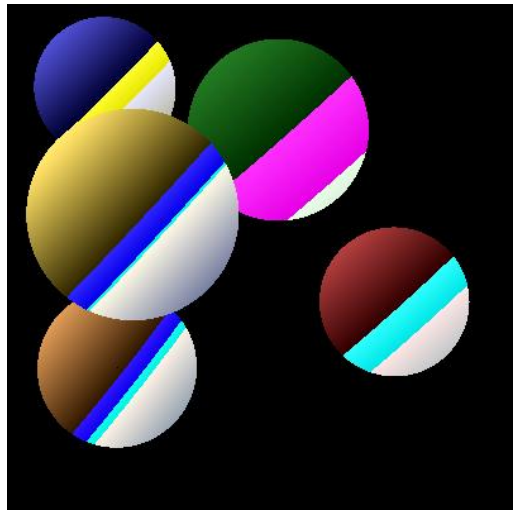
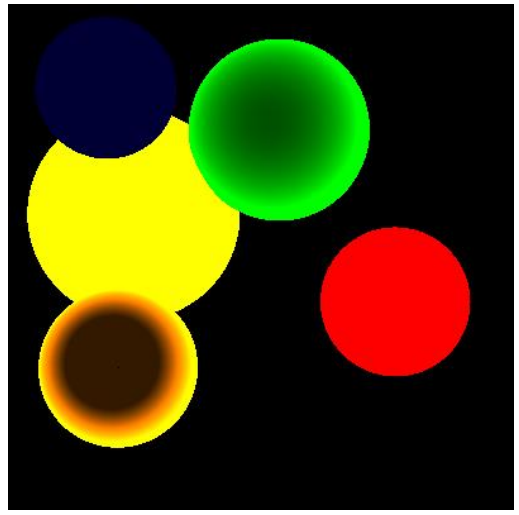
Other interesting mistakes (prev. years)

Other interesting mistakes (prev. years)

Other interesting mistakes (prev. years)

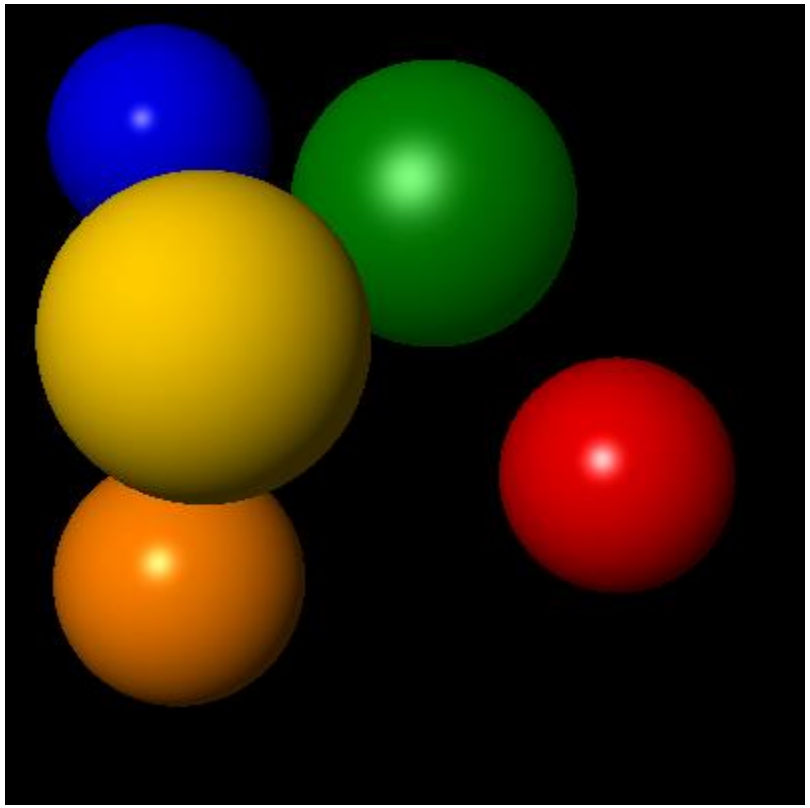








Some notes coding

- compare with examples on webpage
- use material properties from the scene file
- think about efficiency in the raytracer
  - stop calculations as early as possible
  - do not compute elements more than once (store them)
  - think about math, sometimes several alternatives
- rendering speed in general
  - use RELEASE compilation
  - use parallel computing: very easy using OpenMP

Compiler differences

- 0.5-pixel offset to the bottom-left (likely rounding behavior)

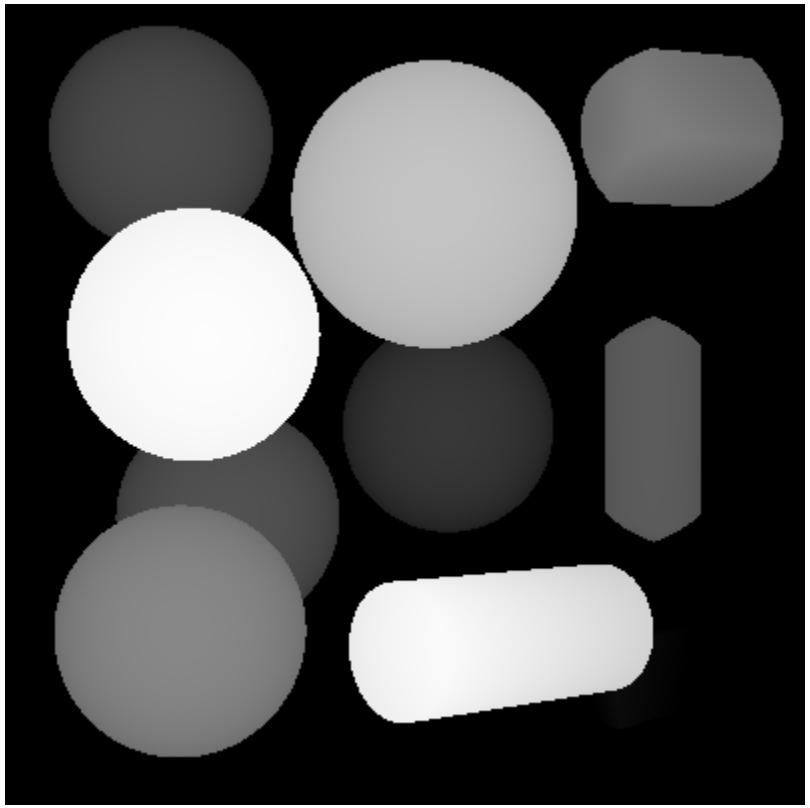


Efficiency in the raytracer

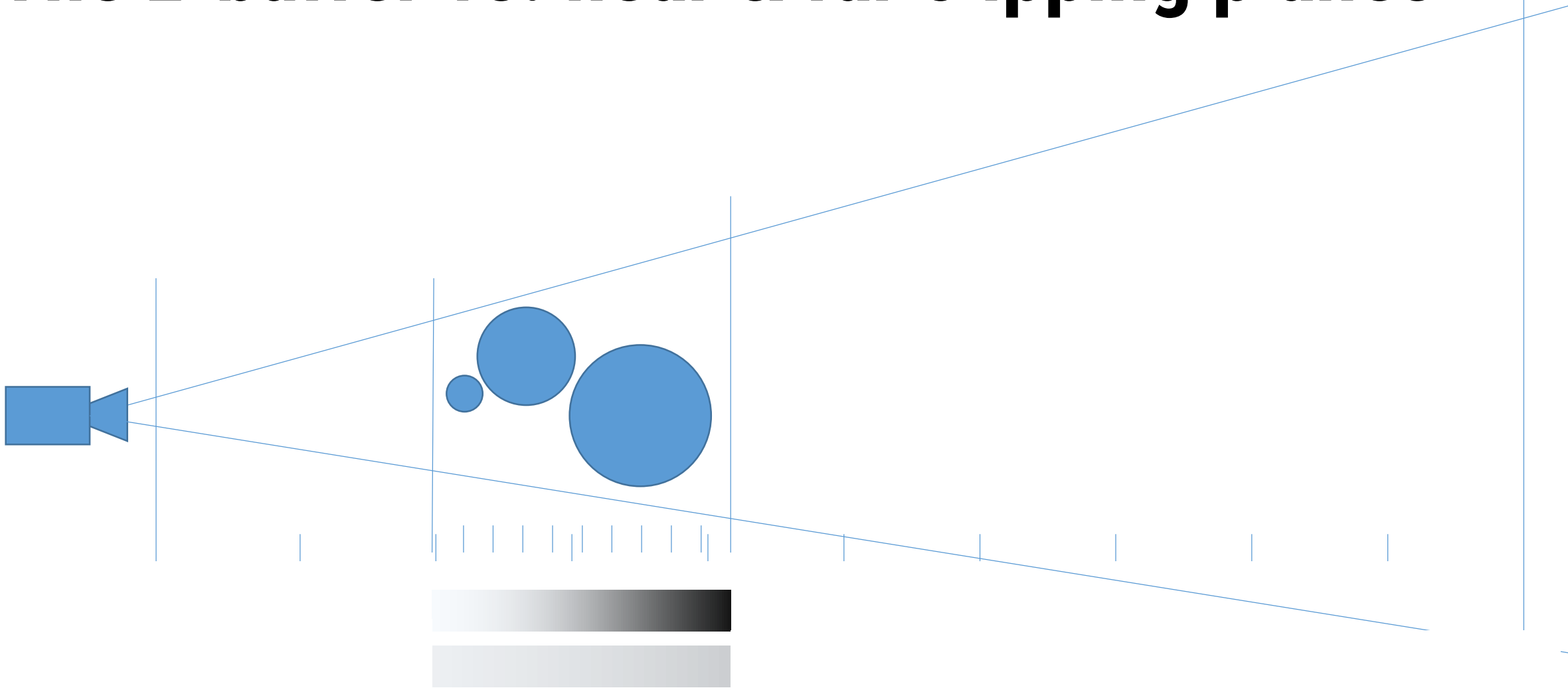
```
if (OC.dot(OC) - OC.dot(ray.D)*OC.dot(ray.D) > pow(r,2)){
        return Hit::NO_HIT();
    }
double t = OC.dot(ray.D) - sqrt(pow(r, 2) - (OC.dot(OC) - OC.dot(ray.D)*OC.dot(ray.D)));
```

Raytracer: assignment 2

- create z-buffer image
- gray-scale to encode depth
- need to define near & far **distance**
- make configurable in YAML file

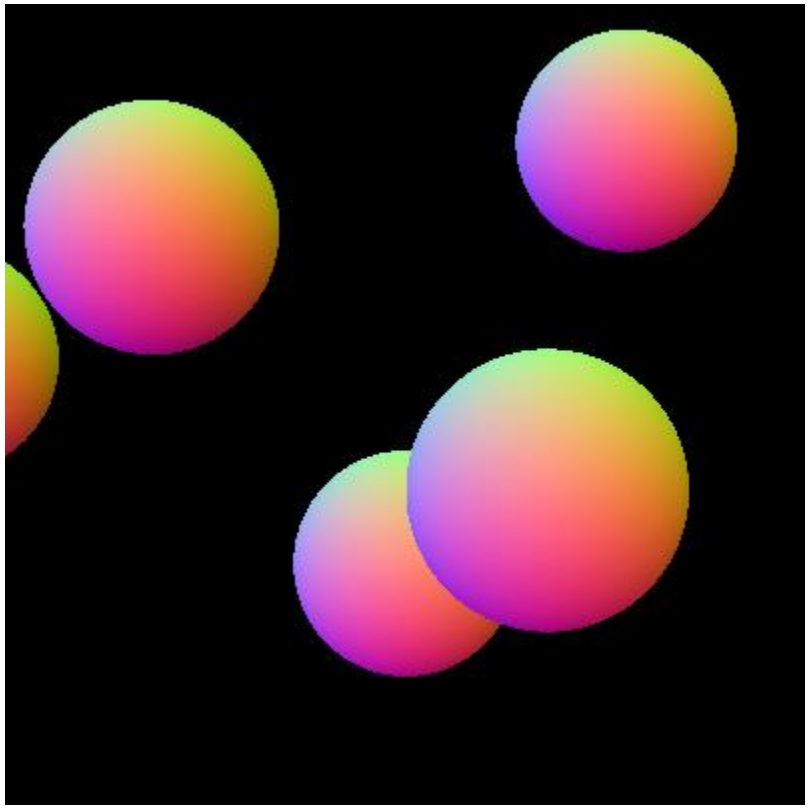


The z-buffer vs. near & far clipping planes



Raytracer: assignment 2

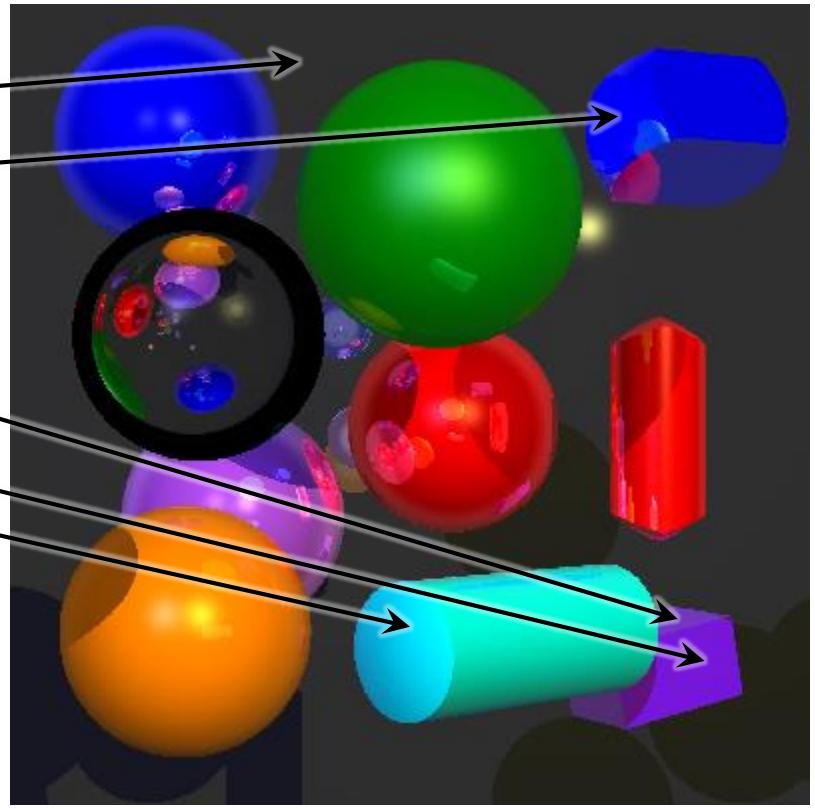
- normal buffer: visual representation of normals on the object's surfaces
- map [-1, 1] to the range of possible colors
- make configurable in YAML file



(different eye position and view direction)

Raytracer: assignment 2

- implement one more type of geometry:
  - quad
  - plane
  - box
  - cylinder
  - cone
  - triangle
  - torus
- to be added for each new geometry:
  - reading the parameters
  - intersection calculation
  - normal calculation



- make configurable in YAML file

Adding to your raytracer: extend YAML file

```
Objects:
- type: sphere
  position: [90,320,100]
  radius: 50
  material: # blue
    color: [0.0,0.0,1.0]
    ka: 0.2
    kd: 0.7
    ks: 0.5
    n: 64
```

```
Output: 1 # 0 = regular, 1 = normal, 2 = zbuffer
Objects:
- type: cylinder
  position: [90,320,100]
  orientation: [1, 0, 0]
  radius: 50
  height: 70
  material: # blue
    # ... existing color and parameters
    albedo: [0.0,0.0,1.0]
    roughness: 0.2
    metalness : 0.2
```

Internships & PhDs @

/

Many Possible Topic Domains