Interactive Computer Graphics: Coursework

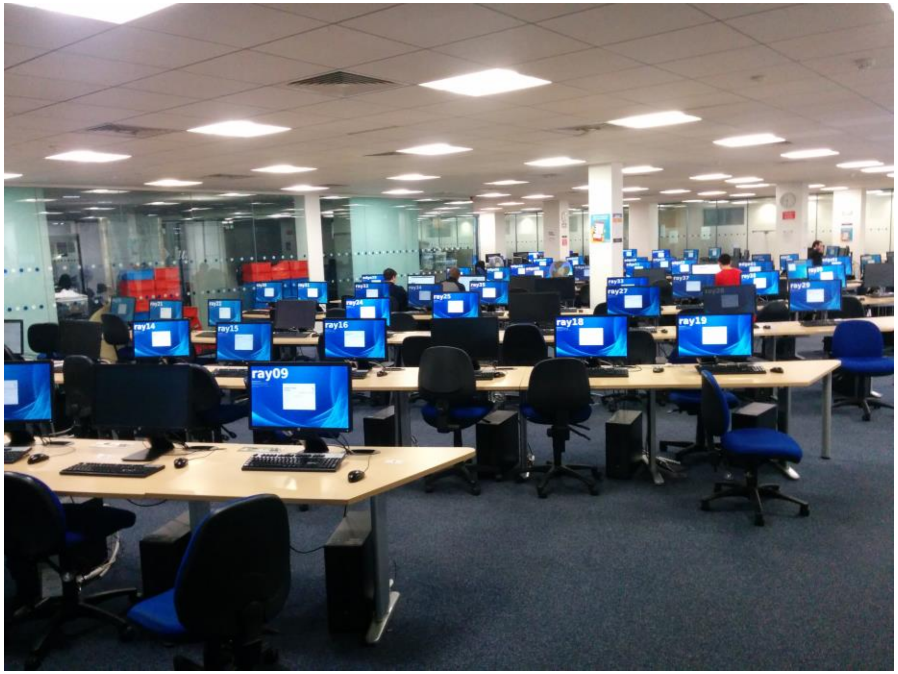
framework and tasks

Logistics

- 5 tasks
  - One per week
  - Last task: 3 weeks
- Description and framework already available for all exercises, but
- Necessary content in lecture per week
  - Sample at the end of the module also via git
- Submission electronically via CATE!
- Lab machines: **all except Corona and Texel**

Labs

- Week 3: Tuesday 12-13
- Week 4: Tuesday 12-13
- Week 5: Tuesday 12-13
- Week 6: Tuesday 12-13
- Week 7: Tuesday 12-13
- Week 8: Tuesday 12-13



weighting

- **Practicals (assessed)**:

1. Framework and Basic interaction **(5%)**
2. Illumination and Shading **(15%)**
3. Generating Primitives **(15%)**
4. Texture & Render to Texture **(25%)**
5. Simple GPU ray tracing **(30% + 10%)**

Getting the framwork

- Open terminal and go to your home directory
- Enter:

```
git clone
 https://gitlab.doc.ic.ac.uk/bka
 inz/cgcoursework2015.git
```

Using the framwork

- Supported on Ubuntu lab machines
- Windows and Mac **no support by us!**

```
cd cgcoursework
cmake .
make
./cgExercise01
```

Update the framework

- Update the framework every time before you start a new task!
- Open terminal and go to your home directory

```
cd cgcoursework2015
git add mainXX.cpp shaderXX..
git commit -m 'Exercise xx`
```

(local commit of your most recent exercise)

```
git pull
```

- Don’t push anything (you don’t have access to push files to the repository – read only)

Framework

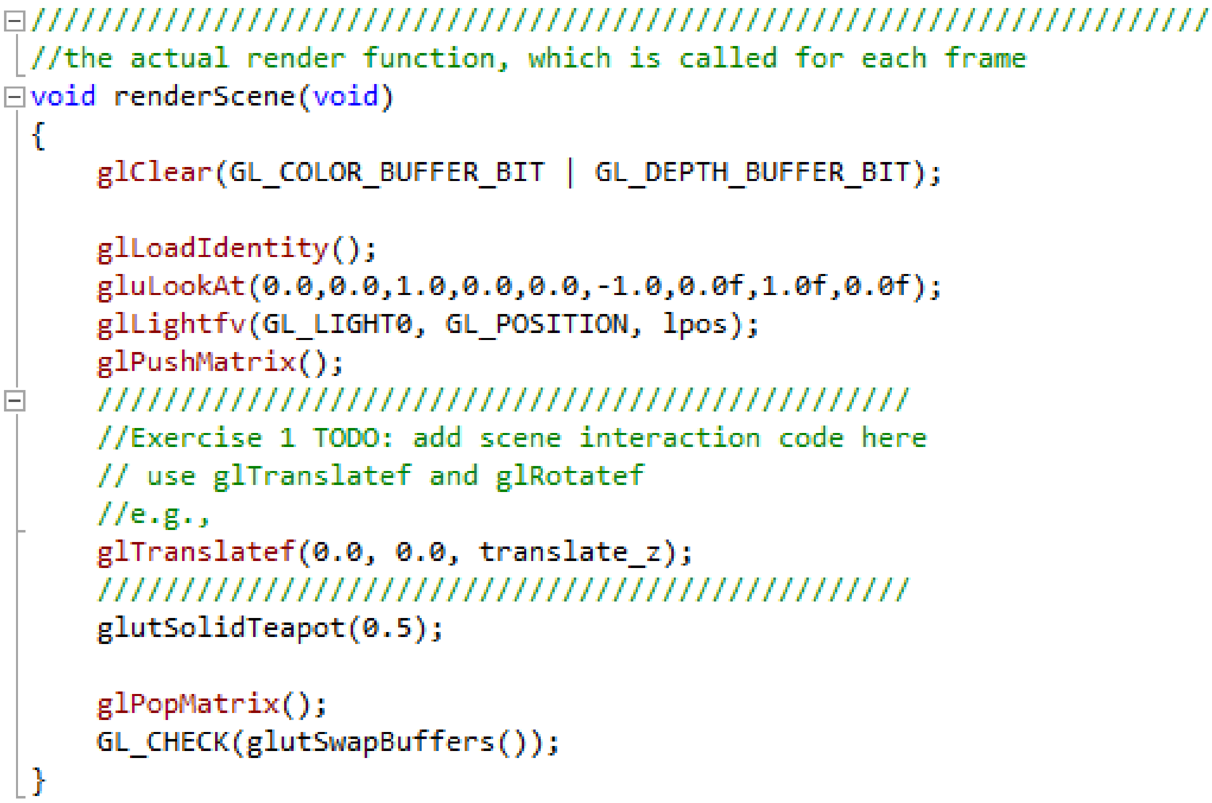
- CMakeLists.txt
- mainXX.cpp
- shaderXX.vert
- shaderXX.geom
- shaderXX.frag

- XX = Exercise Number

- Submission: files as noted in CATE + the generated .png file!

Framework

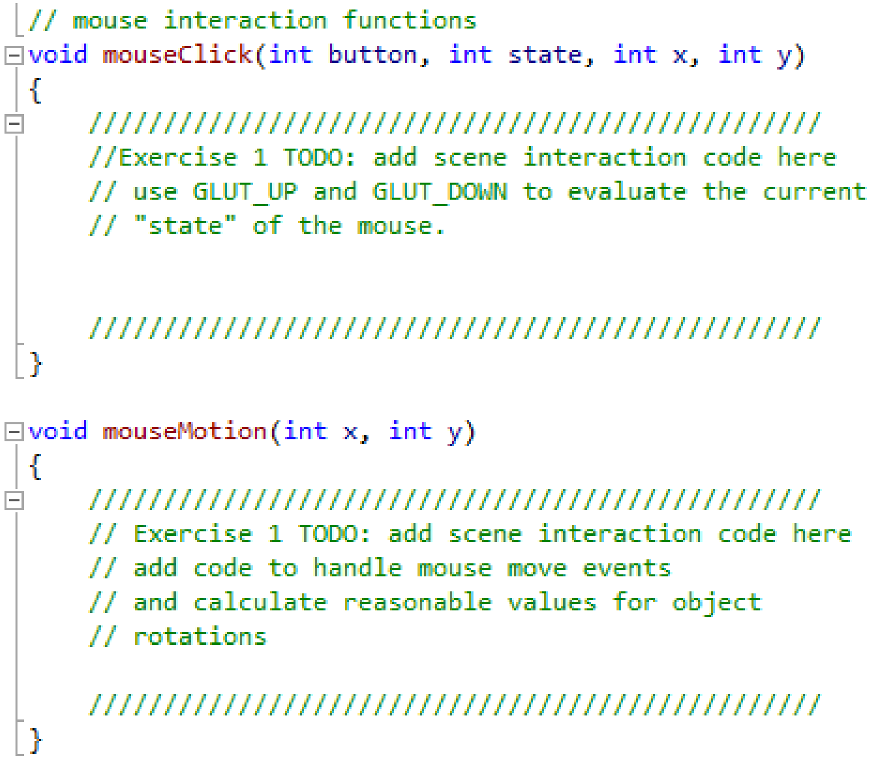
- Description and Code hooks mark areas where you should add your code. Example Ex.1, main.cpp



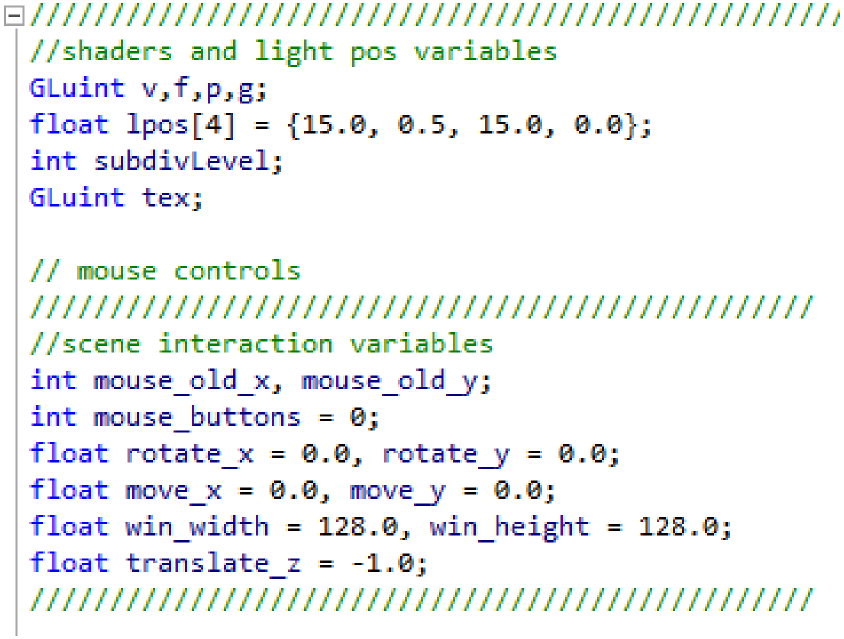
```
///////////////////////////////////////////////////////////////////////
//the actual render function, which is called for each frame
void renderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();
    gluLookAt(0.0,0.0,1.0,0.0,0.0,-1.0,0.0f,1.0f,0.0f);
    glLightfv(GL_LIGHT0, GL_POSITION, lpos);
    glPushMatrix();
    /////////////////////////////////////////////////
    //Exercise 1 TODO: add scene interaction code here
    // use glTranslatef and glRotatef
    //e.g.,
    glTranslatef(0.0, 0.0, translate_z);
    /////////////////////////////////////////////////
    glutSolidTeapot(0.5);

    glPopMatrix();
    GL_CHECK(glutSwapBuffers());
}
```

Exercise 1

- Simple scene interaction
- Use global variables to catch mouse state

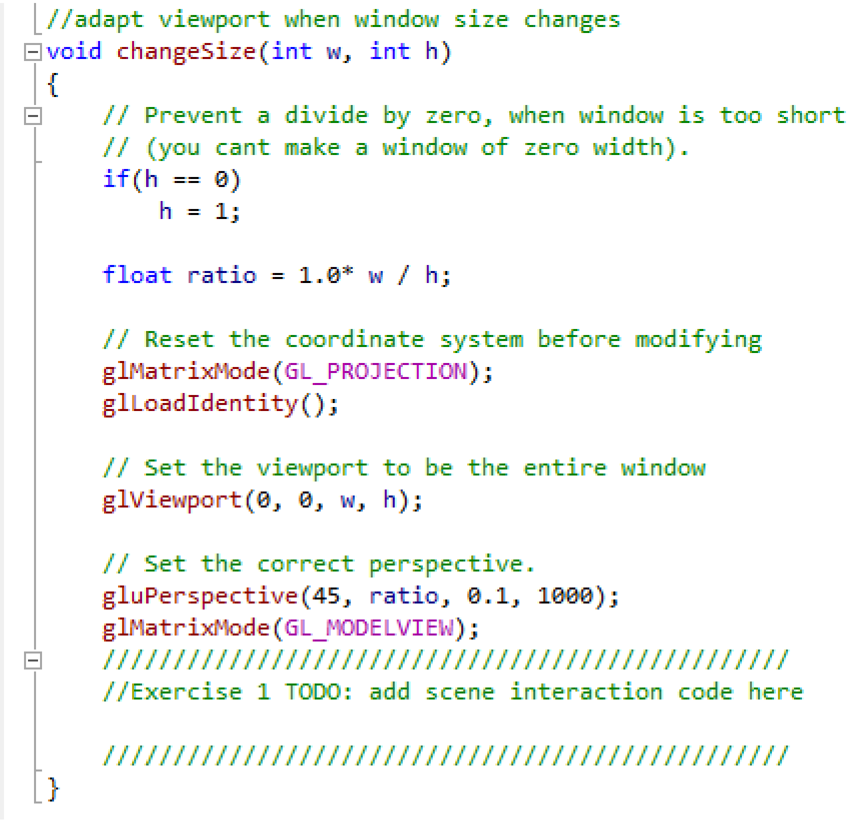


```
// mouse interaction functions
void mouseClick(int button, int state, int x, int y)
{
    //////////////////////////////////////////////////
    //Exercise 1 TODO: add scene interaction code here
    // use GLUT_UP and GLUT_DOWN to evaluate the current
    // "state" of the mouse.


    //////////////////////////////////////////////////
}

void mouseMotion(int x, int y)
{
    //////////////////////////////////////////////////
    // Exercise 1 TODO: add scene interaction code here
    // add code to handle mouse move events
    // and calculate reasonable values for object
    // rotations

    //////////////////////////////////////////////////
}
```



```
///////////////////////////////////////////////////
//shaders and light pos variables
GLuint v,f,p,g;
float lpos[4] = {15.0, 0.5, 15.0, 0.0};
int subdivLevel;
GLuint tex;

// mouse controls
/////////////////////////////////////////////////
//scene interaction variables
int mouse_old_x, mouse_old_y;
int mouse_buttons = 0;
float rotate_x = 0.0, rotate_y = 0.0;
float move_x = 0.0, move_y = 0.0;
float win_width = 128.0, win_height = 128.0;
float translate_z = -1.0;
/////////////////////////////////////////////////
```

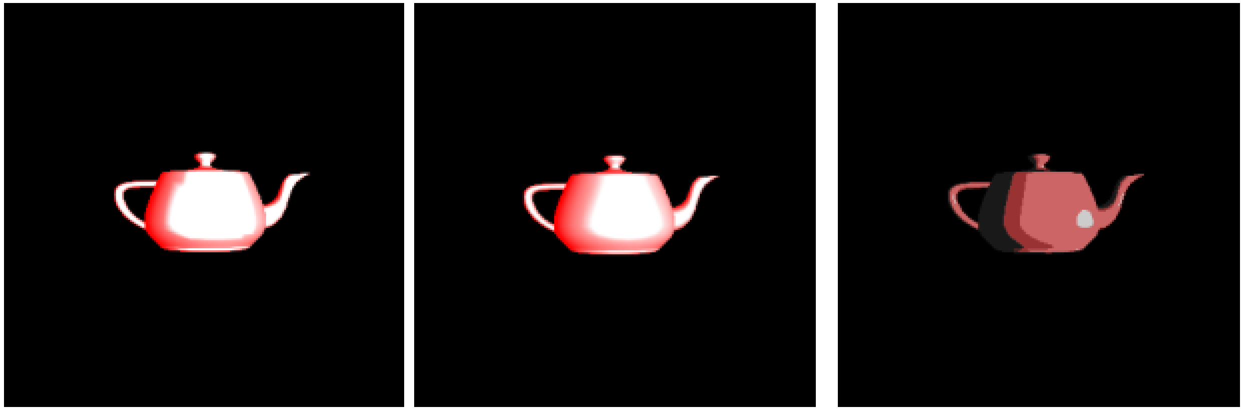
Exercise 1



```
//adapt viewport when window size changes
void changeSize(int w, int h)
{
    // Prevent a divide by zero, when window is too short
    // (you cant make a window of zero width).
    if(h == 0)
        h = 1;

    float ratio = 1.0* w / h;

    // Reset the coordinate system before modifying
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Set the viewport to be the entire window
    glViewport(0, 0, w, h);

    // Set the correct perspective.
    gluPerspective(45, ratio, 0.1, 1000);
    glMatrixMode(GL_MODELVIEW);
    ///////////////////////////////////////////////
    //Exercise 1 TODO: add scene interaction code here

    ///////////////////////////////////////////////
}
```

Exercise 2

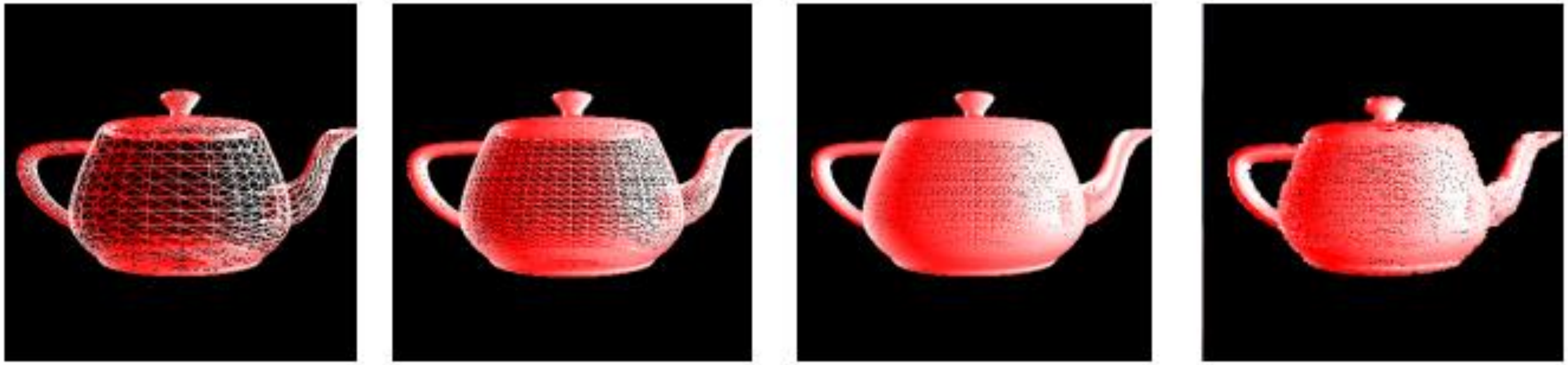
- Illumination
- From now on: working with shaders
  - – shader02.vert
  - – shader02.geom
  - – shader02.frag
  - – (main02.cpp)



(a) Gouraud shading (b) Phong shading (c) Toon shading

Exercise 3

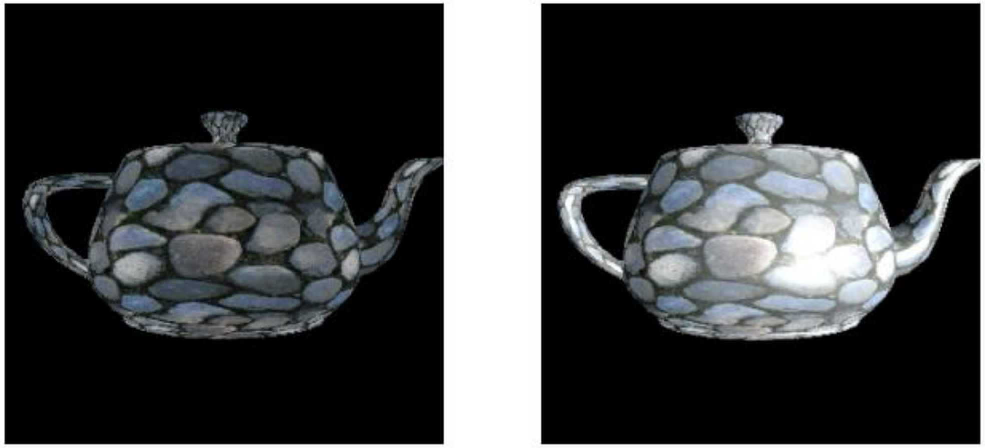
- Generating Primitives
  - shader03.vert
  - shader03.geom
  - shader03.frag
  - (main03.cpp)



(a) Level 0 (b) Level 1 (c) Level 2 (d) Level 2 Bonus

Exercise 4a

- Generating Texture
  - shader04a.vert
  - shader04a.geom
  - shader04a.frag
  - (main04a.cpp) – changing the texture file name if you want



(a) texture only

(b) texture and Phong

Figure 5: Textured and Phong shaded teapot.

Exercise 4b

- Render to Texture
  - shader04b.vert
  - ~~shader04b.geom~~
  - shader04b.frag
  - (main04b.cpp)

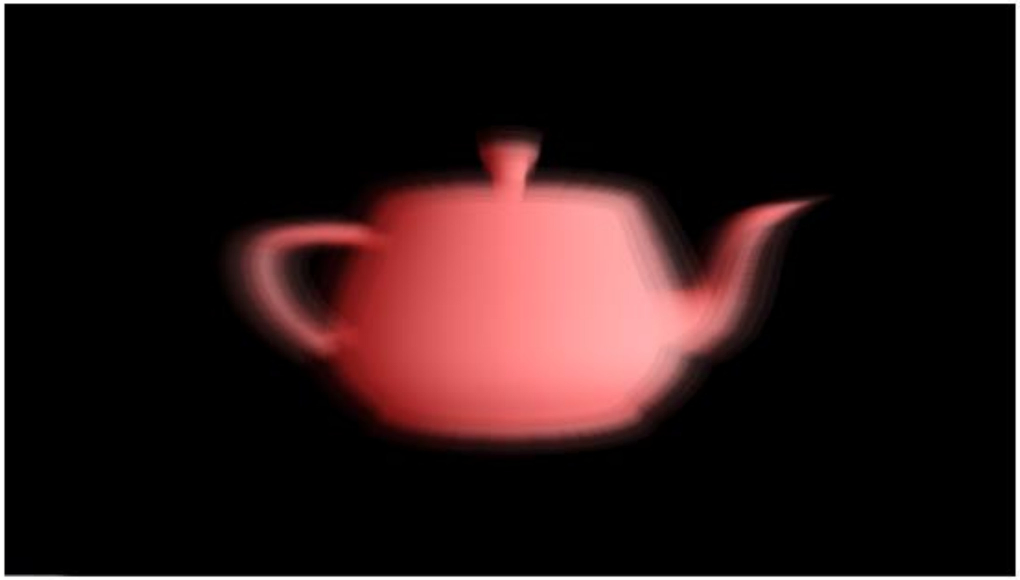


Figure 6: Very simple blur effect.

Exercise 5

- Simple ray tracing of geometric objects
  - shader05.vert
  - shader05.frag
  - (main05.cpp)

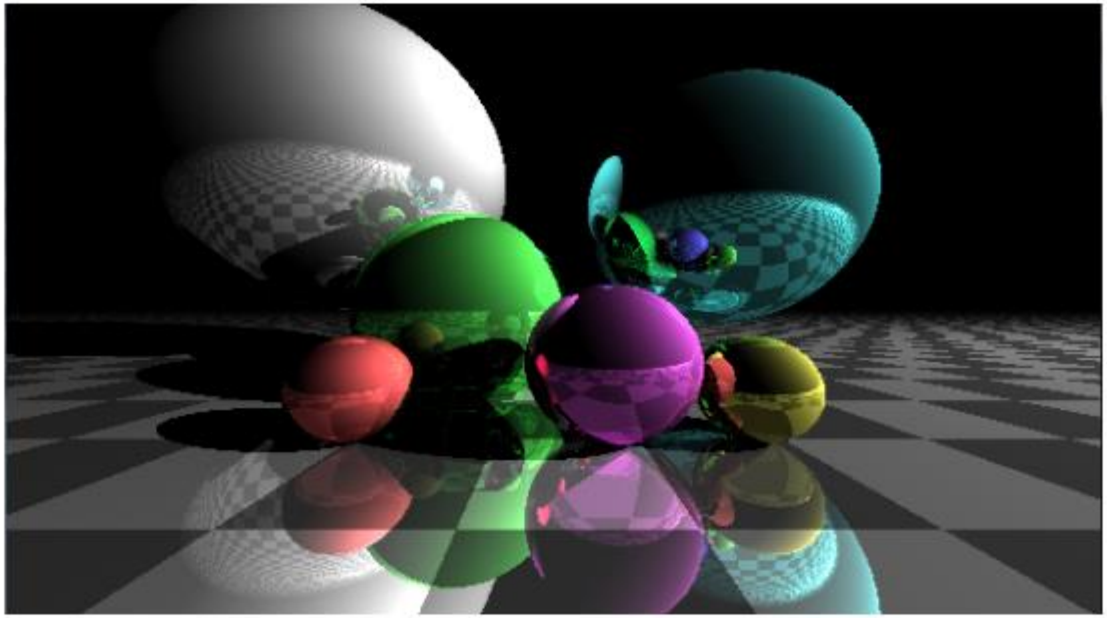


Figure 7: Result from geometric ray tracing.

Exercise 5

- 3 weeks:
  - You get 75% of the points for this task if you implement

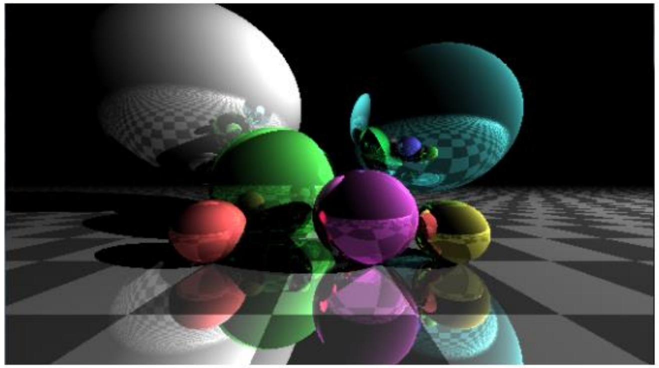


Figure 7: Result from geometric ray tracing.

  - You get another 25% for implementing any other sensible ray tracing effect: soft shadows, refractions, new objects, caustics, etc..

Questions: b.kainz@imperial.ac.uk

Have fun!