Lab 11: Ray Tracer extensions (part III)

3D Computer Graphics

Introduction

No new jar file is provided for this Lab. The idea is that you keep on working on your version of the rendering framework which you obtained after finishing the exercises of Lab 10. However, it is strongly advised that you make a new fresh copy of the project of Lab 10 and rename this copy to 3DCG_Lab11. This will make it easier for you to look again at the work you did in each Lab when you study for the exam later on.

The aim of this Lab is twofold.

Firstly, we will extend our rendering framework so that a larger variety of complex shapes can be created. Last Lab, we started adding support for boolean operators by implementing the difference operator. This Lab, we will implement the intersection operator.

Secondly, we will attempt to alleviate the "jaggies" visible in the final image as a result of aliasing. Hereto, we will implement one anti-aliasing technique: supersampling.

These two features will slow down the rendering even more. However, remember that you can easily decrease the rendering time of your graphical application by reducing the size of the image on the screen (e.g. during the testing phase). Changing the `canvas.width` and `canvas.height` values in the configuration file of your graphical application to 200 and 150, respectively, will reduce the computing time by a factor 16. At the end, after you have fully tested your code, you can always set these parameters back to their original value to get a higher-resolution image.

CSG: The intersection operator

Exercise 1

a) Create a new file `simpleScene4.sdl` in the `resources` folder.

b) This scene should have the following properties:

- white background,
- one light source with RGB colour $(0.8, 0.8, 0.8)$ and position $(-5, 5, 5)$,
- two cylinders with an ambient colour $(0.25, 0.05, 0.05)$ and diffuse colour $(1, 0.2, 0.2)$.
  * The first cylinder is the result of applying two transformations to the generic cylinder:
    · first a translation by one fourth in the x and y-direction and subsequently
    · a rotation by 45^o around the x-axis in counterclockwise order if one looks in the direction of the x-axis.

    The mesh data of the generic cylinder can be found in the file `cylinder1.txt`.
  * The second cylinder is the result of applying two transformations to the generic cylinder:
    · first a translation by minus one fourth in the x and y-direction and subsequently
    · a rotation by 45^o around the x-axis in counterclockwise order if one looks in the direction of the x-axis.

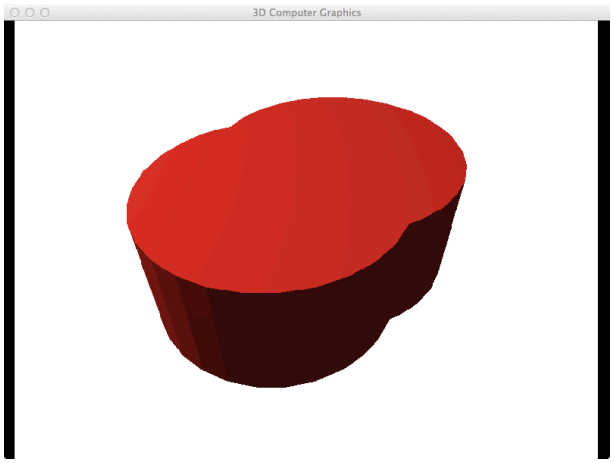
    The mesh data of the generic cylinder can be found in the file `cylinder1.txt`.

c) Create a new package `apps.app7`.

d) Create a new graphical application (`App7`) in this package.

e) Configure this graphical application as follows:
   - The scene to be rendered is described in the `simpleScene4.sdl` file.
   - The width and height of the canvas are 800 and 600 pixels, respectively.
   - The eye of the camera is located at the position $(0, 0, 4)$.
   - The camera is aimed at $(0, 0.2, 0)$.
   - The upwards vector of the camera is in the direction of the y-axis.
   - The worldwindow has a width and height of $4/3$ and 1, respectively, and is located 1 unit in front of the camera.
   - We want to render the scene without shadows or reflections.

f) Run `App7` and make sure you get the image as shown below on your screen.



Exercise 2

a) Add the word "intersection" to the file `simpleScene4.sdl` so that the intersection of the first and second cylinder will be rendered later on.

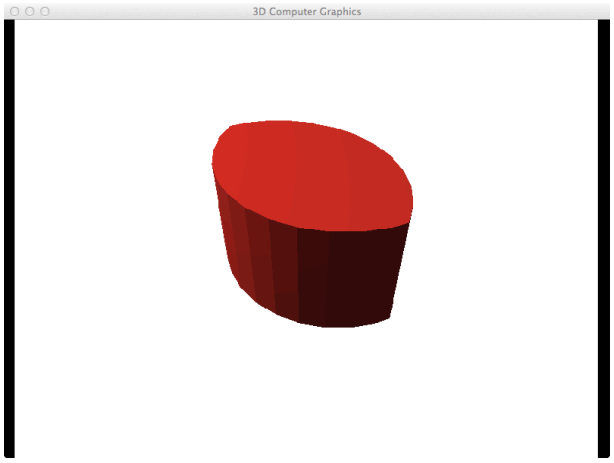
b) In order to render this scene, you will need to implement the intersection method of a new `IntersectionBoolean` class. Implement this method based on the guidelines below.

- Open the slides of Lab 10.
- Study slide 10 to 17 again. All this information is still valid for the intersection operator.
- Study slide 18. `Comb` should now be the intersection of the left and right shape. What are the t-values of the intersection points between the ray and `comb`? Consider how the `combInside` values should change in the diagram in the lower left corner of the slide. How can you compute `combInside` if you know `leftInside` and `rightInside`?
- Consider which changes have to be made to the pseudocode listed on slide 19 based on your answers to the previous questions.
- Study slide 20 en 21. Figure out how these slides should change in case of an intersection operator. If one copies the appropriate `HitInfo` objects of `leftInter` and `rightInter` to `combInter`, which data should be changed?

- Slide 22 mentions three changes in changelist A which have to be carried out if the difference operator is applied. Check for each of these three changes whether they have to be carried out in case of an intersection operator. Use your answer to the previous question as a guide.
- Reconsider the questions on slide 23. Are the answers different for the intersection operator? Do you have to adapt changeset B?
- Reconsider the question on slide 24. Is the answer different for the intersection operator? Do you have to adapt changeset C?

c) Make ALL other changes to the rendering framework which are necessary to visualize `simpleScene4.sdl`.

d) Test your changes by running `App7`. Make sure you get the image shown below as result.



Exercise 3

a) Change "intersection" into "difference" in `simpleScene4.sdl`.

b) Can you predict how this will change the image?

c) Verify your answer by running `App7` again.

d) Can you explain why you obtain this result?

e) Assume you would apply a scaling transformation by 1.1 units in the z-axis just before drawing the second cylinder (the one closest to the camera). Do you think this is a solution to your problem?

f) Verify your answer by running `App7` again.

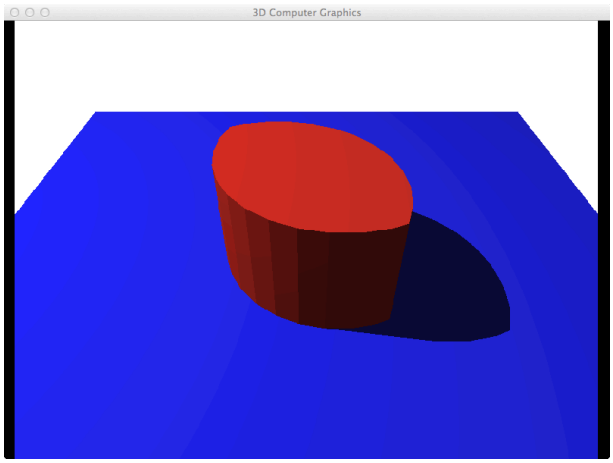
g) Can you explain why you obtain this result?

h) Make sure you get the expected image.

Exercise 4

a) Undo the changes you made to `simpleScene4.sdl` in the previous exercise so that you obtain the figure shown on the previous page again.

b) Adapt `simpleScene4.sdl` and your rendering framework so that running `App7` results in the following image.



Anti-aliasing: supersampling

Exercise 5

a) Make sure `app6.cfg` contains the following data:

```
scene.file = resources/buckyball2.sdl

canvas.width = 600
canvas.height = 450

camera.eye.x = 0
camera.eye.y = 0.5
camera.eye.z = 4

camera.look.x = 0.5
camera.look.y = 0
camera.look.z = 0

camera.up.x = 0
camera.up.y = 1
camera.up.z = 0
```

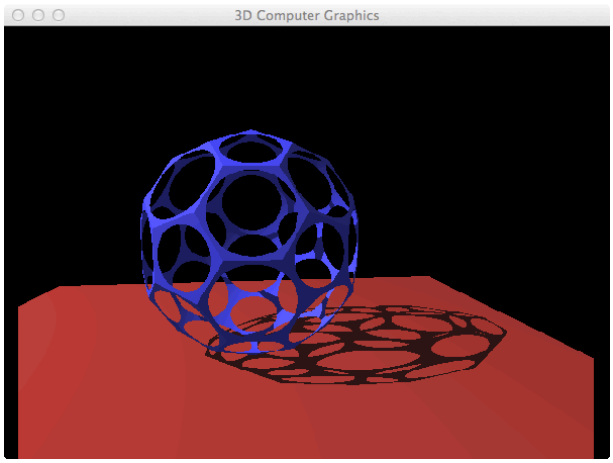
```
camera.worldwindow.distance = 1
camera.worldwindow.width = 1.33333333f
camera.worldwindow.height = 1

raytrace.mode = shadow
```

b) Make sure `buckyball2.sdl` contains the following data:

```
background 0 0 0
light -10 10 2 0.9 0.9 0.9
ambient 0.2 0.1 0.1
diffuse 0.8 0.2 0.2
reflectivity 0
push
rotate -90 1 0 0
translate 0 0 -1
scale 3 3 1
square
pop
ambient 0.15 0.15 0.4
diffuse 0.3 0.3 1
reflectivity 0.2
difference
mesh resources/buckyball.txt
scale 0.99 0.99 0.99
sphere
```

c) Run `App6`. You should obtain the figure shown below.



d) Notice the "jaggies" at the edges of the square, buckyball and shadow. These visual artifacts are most visually distracting at the edges of the shadow. Your next task is to alleviate this aliasing problem by implementing the supersampling method as explained in the slides of this Lab. Your implementation should meet the following requirements:

1. A user should be allowed to configure a graphical application with

a new key `raytrace.supersampling` whose value should be an integer number.

2. The meaning of this new key is as follows:
   - `raytrace.supersampling = 0` means no supersampling should be applied. This corresponds to the upper case on slide 12 of this Lab.
   - If the user sets `raytrace.supersampling` to an integer larger than 0, supersampling should be applied. For example,
     * If `raytrace.supersampling = 1`, two samples should be taken along the side of a pixel. This corresponds to the middle case on slide 12 of this Lab.
     * If `raytrace.supersampling = 2`, three samples should be taken along the side of a pixel. This corresponds to the lower case on slide 12 of this Lab.
     * And so on ...

   Identify the three classes in the rendering framework which have to be adapted to implement supersampling in this way. Know what you are doing!

- Test your work by running `App6` with supersampling turned on.
- Compare the resulting images for supersampling values equal to 0, 1 and 2. Is a supersampling value of 1 enough to remove the visual artifacts or is a sampling value of 2 or more necessary?