Lab 10: Ray Tracer extensions (part II)

Constructive Solid Geometry

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 9. However, it is strongly advised that you make a new fresh copy of the project of Lab 9 and rename this copy to 3DCG_Lab10. 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 and the following Lab session is to implement boolean operators which allow to create new, more complex 3D objects from simple shapes. In this Lab, we will add support for the difference operator.

Exercise 1

a) Download the `simpleScene3.sdl` file and store it in the `resources` folder.

b) Open this file and study its content. Make sure you understand all commands.

c) Try to picture the scene described in this file.

d) Create a new package `apps.app5`.

e) Create a new graphical application (`App5`) in this package.

f) Configure this graphical application as follows:

- The scene to be rendered is described in the `simpleScene3.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 $(0, 0, 5)$.

- The camera is aimed at the origin.
- 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.

g) Run `App5` and make sure you get the expected image on your screen.

We will use this graphical application to check our support for the difference operator later on.

Exercise 2

In this exercise, we will create new classes to represent boolean objects.

a) Create a new package `geomobj.bool`. All classes related to boolean objects will be put in this package.

b) Create an abstract class `Boolean` in this package which implements the `GeomObj` interface. This class represents a boolean operator.

c) Add two public instance variables `left` and `right` of the type `GeomObj`. These instance variables store the two objects this boolean operator is applied to. Note that these two objects can be boolean operators themselves working on two other objects.

d) Add a constructor to the `Boolean` class which allows to initialize its two instance variables.

e) Create a new class `DifferenceBoolean` which extends the `Boolean` class. This class represents the difference operator.

f) Add a constructor to the `DifferenceBoolean` class which allows to initialize the two instance variables of its superclass.

g) This class should also provide an implementation for the `intersection` and the `hit` method of the `GeomObj` interface. As we want to test our code as quickly as possible, let the `hit` method return false for now.

h) Implement the `intersection` method in the `DifferenceBoolean` class. Use the slides of this Lab as a guide.

Exercise 3

The aim of this exercise is to add boolean operator commands to our scene description language.

a) Open the `simpleScene3.sdl` and add the word "difference" on a new line between the diffuse and push command.

b) The idea is to render the boolean object resulting from taking the difference of the generic sphere and the transformed sphere. Can you picture this object?

c) Adapt the `Token` enum.

Next, we update the `SceneFactory` class. The `createScene` method of the `SceneFactory` class processes all tokens in the sdl file. If a new token is not related to the background, a light source, material or transformation properties, it is assumed to be an object token which is processed by the `createShape` method. Note that we need to add an intermediate layer which checks whether the object token is a boolean operator such as `difference`. If our parser encounters such a token, it should properly read the following two objects on which the difference operator should be applied. Note that these two objects can be preceeded in the sdl file by tokens related to material or transformation properties. Therefore, we need a separate method (`getObject`) to read these two objects.

d) Change the `createShape` method call in the `createScene` method of the `SceneFactory` class into `createGeomObject`. This new method should have the same arguments as the `createShape` method. But do not change the name or implementation of the `createShape` method!

e) Create this new private method `createGeomObject` and implement it according to the pseudocode listed below

```
if(token = difference){
  GeomObj left = getObject(scanner, currMtrl, stack);
  GeomObj right = getObject(scanner, currMtrl, stack);
  return the appropriate boolean object
} else {    // token is a shape token
   call a method to create a shape and return the result
}
```

f) Create a new private method `getObject` with the correct parameters and implement it according to the pseudocode listed below

```
while(the sdl file still contains unprocessed tokens){
  String token = next unprocessed token in the sdl file;
  if (token is not processed as a material token ){
    if(token is not processed as a transformation token){
      // todo: call an appropriate (existing) method
      // todo: and return its result.
    }
  }
}
throw IllegalStateException
```

Exercise 4

a) Check your support for the difference operator by running `App5`. Make sure you get the expected result.

b) When you run `App5`, you get a side view of the boolean object. What is the location of the light source?

c) Animate the camera so that you get a top view of the boolean object. Explain why the resulting image is unrealistic based on your answer to b).

d) Make the necessary changes to make the image more realistic.

e) Check you changes by running `App5` again.

Exercise 5

a) Download the `buckyball2.sdl` file and store it in the `resources` folder.

b) Open this file and study its content. Make sure you understand all commands.

c) Try to picture the scene described in this file.

d) Create a new package `apps.app6`.

e) Download the `app6.cfg` file and store it in this new package.

f) Open this file and study its content. Make sure you understand all commands.

g) Create a new graphical application (`App6`) in this package which uses `app6.cfg` as configuration file.

h) Run `App6` and make sure you get the expected image on your screen.

Exercise 6

Use the guidelines mentioned below to play around with some parameters to check your current rendering framework and enjoy the resulting images. For each guideline mentioned below, try to predict its effect, make the corresponding change and check your prediction by running `App6`.

a) Render `buckyball2.sdl` with shadows and notice the subtle difference.

b) Change the last transformation command in `buckyball2.sdl` into

```
scale 1.1 0.8 1.1
```

Animate the camera upwards to check whether your support for shadows and boolean operators smoothly work together.

c) Change the last transformation command in `buckyball2.sdl` back to

```
scale 0.99 0.99 0.99
```

d) Render `buckyball2.sdl` with reflections turned on and notice the more realistic look compared to the previous image.

d) Change the last transformation command in `buckyball2.sdl` into

```
scale 0.98 0.98 0.98
```

and compare with the previous result.

e) Change the last transformation command in `buckyball2.sdl` into

```
scale 0.999 0.999 0.999
```

and compare with the previous result. Animate the camera. Have a look at the nice shadow patterns which are cast by the boolean object onto the "floor".