This assignment is to be completed in a group

Assignment Three
A simple ray tracer

Software Design and Programming and
Software and Programming III

Spring Term 2016

Contents

This assignment is to be completed in groups of no less than three and no more than four members. You may *self select* your group but we reserve the right to alter the membership (obviously before you start work on the problem).

1 Preamble

This assignment adds to your experience in dealing with an existing code base. The problem is to be tackled in groups, as outlined above and on the website. T he sample code is provided in Scala but you can convert it to Java if you wish (I strongly suggest you don't).

The solution to the problem will involve the use of an actors library and the functional approach to programming. Again, this is most easily achieved using Scala but you can use the functional features of Java 8 if you wish.

The sample code mentioned in the text can be found on the module repository under the `RayTracer` project. We provide a (brief) introductory document on ray tracing but you don't actually need to understand the process at more than a high level abstraction as the coursework is more about concurrency and distributed processing.

1.1 Learning objectives

- Experience extending an existing code base.
- The use of the actors programming paradigm and an appropriate library (Akka[1]).
- Further practice with the functional programming paradigm.
- The experience of working in groups and using a shared codebase.

1.2 Collaboration policy and academic integrity

You should carry out this assignment in groups; how you decompose the problem into individual tasks is up to the group members. It is against the rules for one person to do most of the programming on this assignment and it is required that all members of the group understand the codebase. We reserve the right to use individual vivas to evaluate the group submission.

With the exception of your group, you may not look at anyone elses code, in any form, or show your code to anyone else in any form (except the teaching staff). This includes having a public repository for your code on `github` or `bitbucket` (make it a private repo and share it only with your group); this is a fundamental requirement of this coursework assignment.

1.3 Getting help

If you dont know where to start, if you dont understand testing, if you are lost, etc., please see someone immediately — an instructor, or a TA. Do not wait. The forums are there to help so please do use them. A little in-person help can do wonders. See the module Moodle page for contact information.

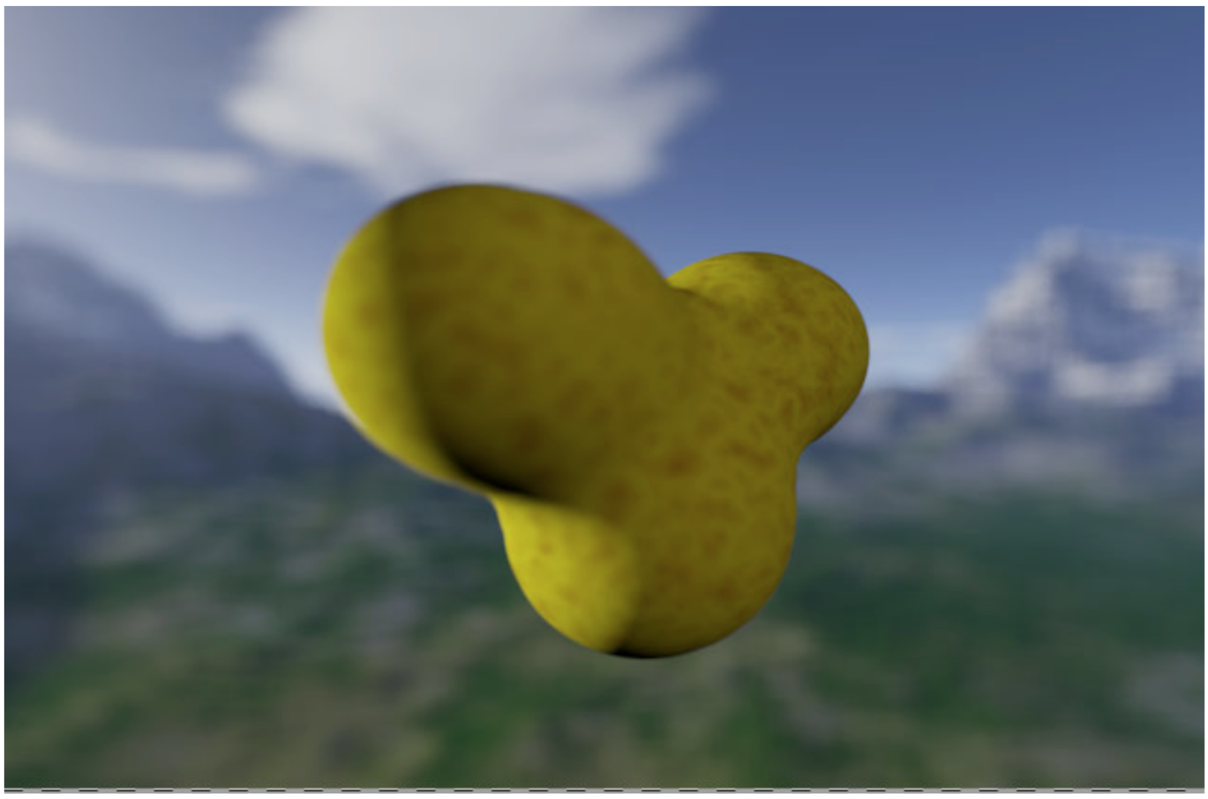
1.4 Background — Actors

"Actors are very lightweight concurrent entities. They process messages asynchronously using an event-driven receive loop. Pattern matching against messages is a convenient way to express an actor's behaviour. They raise the abstraction level and make it much easier to write, test, understand and maintain concurrent and/or distributed systems. You focus on how the messages flow in the systeminstead of low level primitives like threads, locks and socket I/O."

[1] `http://akka.io`

2 The problem

A 3D scene is represented as a set of objects. A ray tracer works by casting rays from a virtual camera onto the scene, computing intersections with objects in the scene. The rays are cast through an invisible plane (the *view window*) in the scene that represents the generated image. The view window can be thought of as a grid, where each square represents a pixel of the generated image.



When a ray hits an object, the ray is reflected and refracted, producing secondary rays which may intersect other objects. All of these rays contribute to the final colour computed for the pixel. For instance, a ray through a given pixel might intersect with a reflective red object and the reflected ray may in turn intersect a blue object. The resulting colour of the pixel will be *purple-ish*[2].

2.1 The input file format

To run the code provided in the repository you will need an input file; a sample file is provided in the file named `input.dat`.

The input file consists of a sequence of directives, one per line. The `sphere` directive defines a sphere by the (x, y, z) position of its centre, its radius, its (r, g, b) colour (all between 0 and 1), and its reflectivity (also between 0 and 1). All numbers are floating point numbers. The `light` directive defines a point light source by its (x, y, z) position and its (r, g, b) colour.

For example, the following defines three spheres and two lights:

[2]For further information on ray tracing see the following article link (`http://www.ics.uci.edu/~gopi/CS211B/RayTracing tutorial.pdf`)

```
sphere 0 0 2 .25 .8 .9 0 .8
sphere -.2 .15 1 .25 .8 0 0 .8
sphere .35 0 1 .25 0 0 .7 .8
light 1 1 1 1 1 1
light -1 -1 1.5 1 1 1
```

- The first sphere is at (0,0,2), has radius 0.25, colour (0.8,0.9,0) greenish-yellowish, and reflectivity 0.8.
- The second light is at (1,1,1.5) and has colour (1,1,1) white.
- There must be at least one sphere and one light.
- The camera is at the origin (0,0,0) and shoots rays in the positive z direction (0,0,1).

2.2 The provided code

The provided code should compile *out-of-the-box* using the command line but you are welcome to use an IDE.

To run the code:

```
scala -cp . Trace input.dat output.png
```

The code writes a `PNG` image file, which you should be able to view using a web browser or other image viewer.

2.3 Your task

Since the colour of each pixel is computed independently, ray tracers are an excellent candidate application for concurrency as one can compute pixels in parallel. Your task is to parallelise a sequential ray tracer using an appropriate actor library, e.g., the Scala AKKA framework.

In theory, the ray tracing for each pixel can be done concurrently but in reality, since our code will run on 24 processors, we can reduce the overhead by parallelising each row of the image. Create an actor that computes the colour of each pixel in a row and sends the colour to a coordinator actor.

Implement a coordinator actor by modifying the existing `Coordinator` object appropriately. This should collects the colours of each pixel sent as messages by the tracer actors. The coordinator also has a pr int method that outputs the PNG file.

In the provided sequential version, the `print` method is called from `render` but this will not work for the concurrent solution as `render` will spawn several actors (perhaps via `traceImage`) and might call print before the image has been completed. You will need to move the call to `print` to a more appropriate place.

You should not need to introduce additional actors beyond the ones described above but you can if you wish.

3 Submission

Your repository will be cloned at the appropriate due date and time.

Credits

. . . I've forgotten where this comes from. . . — I'll add the credits in the next version.