Adding a new embedder to CaGe

This text will walk you through the different steps to add a new embedder to CaGe in a non-invasive way, i.e. without needing to have or update any of the source files of CaGe. At the moment this means that you will use the native embedder that is supplied with CaGe. The native doesn't mean that you can't use Java to write your embedder. This embedder just uses Java Native Interface (JNI) to start up the generators in a separate process. At the moment there is no support for other embedders.

Preparation

The first you will have to do is to turn on the expert mode of CaGe. You can do this by opening the file CaGe.ini that you will find in the directory that contains your CaGe installation. Locate the following line in this file:

```
CaGe.ExpertMode:    false
```

Replace the `false` in this line with `true`. If this property is already set to true, then the expert mode is already turned on and you can proceed immediately. Basically what this expert mode does, is give you the ability to edit the generator and embedder commands for each of the generators prior to starting the generation process. Just proceed to the output panel of any generator after turning on the expert mode. At the top of the panel you will see three text boxes that allow you to see and/or alter the commands for the generator, the 2D embedder and the 3D embedder.

What goes in?

An important thing you need to know before you can write your own embedder is, of course, the format in which you will receive the graphs. Basically what you will receive from CaGe is a file containing the graph in the writegraph format. Depending on you're choice of type of embedding this will be a writegraph3d or writegraph2d file. However in both cases all the coordinates will be zero.
You can also easily see the input you will receive. First choose the generator you want, then proceed to the output panel. Before the embedder commands you add the following command:

```
tee -a WhatGoesIn |
```

So if the original embed command was:

```
embed
```

It will now be:

```
tee -a WhatGoesIn | embed
```

The tee command reads from standard in, and writes its input both to the specified file and standard out. So when the graphs are embedded, the file WhatGoesIn will contain the input that the embedder received from CaGe.

What comes out?

Another important thing is the format you need to supply back to CaGe. Depending on the dimension this will be a writegraph3d or a writegraph2d file. Again you can use the tee command to view an example of this. This time you simply pipe the output of the embedder into tee, e.g.

```
embed | tee -a WhatComesOut
```

The writegraph format

This format originated from Combinatorica, a Mathematica package. It is a plain-text format and easily readable for humans. CaGe always uses this format for embedded graphs.
A file in writegraph format may start with a header, but this is not mandatory. It is however advised to do this. If you have followed the previous steps, you will already have seen that CaGe and its current embedder always include this header. This header always has the following form

```
>>__<<
```

In this header the __ is replaced by the specific format. There are three options for this: `writegraph`, `writegraph2d` and `writegraph3d` depending on the number of coordinates supplied per vertex (respectively 0, 2 and 3).
After the header there is one line for each vertex, and this line contains, separated by white space,

- the vertex number (sequentially numbered, starting with one)
- vertex coordinates,
- the numbers of all vertices adjacent to the current one.

Since we deal with lists of graphs, we define a line containing just the number zero as the separator between two graphs (as zero is not a valid vertex number). When writing graphs to the embedder they will be passed on one-at-a-time. So for the embedder a line containing just the number zero might be seen as the end of the file.

The right place

The final thing you need to know is the correct directory to place your embedders file in. This is determined by the property `CaGe.Generators.Path` in the file CaGe.ini. By default this is set to the directory `Generators` in the directory that contains your CaGe installation and all

the directories in the environment variable `PATH`. It is your own choice whether you add a directory to this list or just move your files to one of the available directories.
Your embedder will normally not be ran in the directory it is placed in, but will be ran in the directory specified by the property `CaGe.Generators.RunDir` in the file CaGe.ini. This is important if, e.g., you want to run an embedder written in Java. The java command will run in the specified directory and so, the classpath will also be resolved from that directory.

Some examples

```
import java.util.Random;
import java.util.Scanner;

/**
 * Reads a graph in writegraph2d or writegraph3d from standard in
and writes a random embedded graph
 * to standard out.
 * @author nvcleemp
 */
public class RandomEmbedder {

    private static final Random RANDOM = new Random();

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String header = in.nextLine().trim();
        int dimension = 0;
        if (header.equals(">>writegraph2d<<")) {
            dimension = 2;
        } else if (header.equals(">>writegraph3d<<")) {
            dimension = 3;
        } else {
            System.err.println("Error: RandomEmbedder needs a valid
header.");
            System.exit(1);
        }
        System.out.println(header);
        while (in.hasNextLine()) {
            String line = in.nextLine().trim();
            if (!line.equals("0")) {
                String[] vertex = line.split("\\s+");
                if (vertex.length < dimension + 1) {
                    System.err.println("Error: Illegal format in
input file.");
                    System.exit(1);
                }
                for (int i = 0; i < dimension; i++) {
                    vertex[1 + i] =
Double.toString(RANDOM.nextDouble()*5);
                }
                for (int i = 0; i < vertex.length; i++) {
                    System.out.print(vertex[i]);
```

```
                        System.out.print(" ");
                    }
                    System.out.println();
                }
            }
            System.out.println("0");
        }
    }
```

If you place this compiled file in the directory Generators in the CaGe installation directory and the run directory is set to the installation directory, then you can use this generator with the following command:

```
java -cp Generators RandomEmbedder
```

The default embedders

CaGe ships with some default embedders. Some are very general, other are more specific. Below you'll find an overview of these embedders.

embed

This is an embedder that is optimized for chemical graphs, but also returns good results for various other graphs. It is written in C. It is capable of embedding both in 2D and in 3D. One of its more interesting options if you want to add your own embedder to CaGe, is that this graph can take an already embedded graph and try to improve that embedding. To do this you use the option `-i` and pass it the parameter `k`, e.g.:

```
embed -i k
```

This tells embed that it should keep the initial embedding of the file it receives. Otherwise embed will try to find a suitable initial embedding. For more information you can use the following command to get a usage message:

```
embed -h
```

cage.embedder.NanoconeEmbedder/ cage.embedder.NanotubeEmbedder

This is an embedder developped specifically for the embedding of nanocones (resp. nanotubes) in 3D. It has been written in Java and is packed in the jar file CaGe.jar. A drawback of this at the moment is that only native embedders are supported. Therefore caGe will start a complete JVM in a separate process each time a graph needs to be embedded with this embedder. This is something that will receive our attention in the future.

These embedders are a good example of how an embedder written in Java can be used in CaGe. e.g. the embedder for nanotube is used with the following embed command:

```
java -cp CaGe.jar cage.embedder.NanotubeEmbedder
```