How to use Slur and Riddle

Slur lives in /u0/chip/microcosm/Slur/Slur on Moth.
Riddle lives in /u0/chip/microcosm/riddle/riddle also on Moth.

Here's what they do:

Slur takes .reg files as generated by Red and turns them into .rdl files suitable for input into Riddle. It now understands certain basic things such as how to handle doors properly and not to make regions be dark.

Riddle takes .rdl files and turns them into grody Genesis files suitable for uploading to Quantum for Janet to use. She runs these through Genesis which turns them into database entries, resulting in actual regions in the world.

Information on Slur:

Slur takes a bunch of arguments on the command line that it uses to build the region descriptor for Riddle. The general form of the command is:

```
/u0/chip/microcosm/slur/slur redfile.reg [ -o outputFileName ]
        [ -d facingDirection ] [ -r regionName] [ -s southNeighbor ]
        [ -n northNeighbor ] [ -e eastNeighbor ] [ -w westNeighbor ]
        [ -c classDataFile ]
```

where brackets ('[' and ']') indicate what's optional, as usual.

'redfile.reg' is the .reg file as output by Red.

It is traditional to name the output file 'something.rdl'. If it is not given the output goes to standard out.

The region name, the facing direction, and the neighbors can be almost anything: they are substituted verbatim into the Riddle file at the appropriate spot.

The classDataFile is the class.dat file used to generate the .reg file with Red. If not given it simply defaults to 'class.dat'.

The details of Riddle's input format are documented elsewhere, but there's not really much you need to know about it. What you have to do is put together a couple of command scripts to handle everything.

If you look at the directory /u0/chip/microcosm/realms you will see that it has a number of subdirectories, each of which corresponds to one of the realms we have created or have started to create. Each of these subdirectories in turn has a particular structure which I have adopted as conventional. I recommend that you follow the same format and just add new subdirectories here. Eventually we will move it out of my account and into some common space.

Each of these subdirectories has the following stuff in it:

class.dat	A class.dat file as output by Muddle. Slur requires this to decipher the .reg files. It had better match the class.dat file that Red used to produce the .reg files too, else bad bad bad things will happen.
regdir	A subdirectory holding all the .reg files awaiting processing thru Slur.
rdldir	A subdirectory holding all the .rdl files

	corresponding to all the .reg files, i.e., the output from Slur awaiting processing thru Riddle.
Slur_script	A shell script to run Slur. This will be explained in more detail below.
foo_regions.rdl	A Riddle "include" file that defines the values of all the regions. This too will be explained. "foo" of course should be the realm name.
foo.rdl	The master Riddle file to make it all go.
ofile	The final output from Riddle (i.e., the input to Genesis).

The Slur script needs to be set up to generate all the regions in the realm. Following the above conventions, my Slur scripts all consist of an entry for each region in the realm. Each entry should have the form:

```
echo regionName
/u0/chip/microcosm/Slur/Slur regdir/regionName.reg \
        -o rdldir/regionName.rdl \
        -d FACE_DIR \
        -r regionName \
        -e eastNeighbor \
        -w westNeighbor \
        -s southNeighbor \
        -n northNeighbor
```

The echo command is just so that the script can generate an output that tells you how it is doing (sometimes it can take a while if you're doing a lot of regions).

'regionName' should have whatever the name of the region is substituted for it. I always follow the convention of keeping the region name, the .rdl file name and the .reg file name consistent. I.e., we have region 'foobar' and the files 'foobar.reg' and 'foobar.rdl'.

'FACE_DIR' should be one of the four values FACE_EAST, FACE_WEST, FACE_SOUTH or FACE_NORTH. This tells the orientation of the region, and is interpreted as the direction you are facing when you are standing on the view-point edge of the region (what is represented by the thick gray line on the maps). For example, if the top of the map is west and a box on the map has a grey bar on the right-hand edge, you want FACE_SOUTH.

The various neighbors should be the names of the neighbor regions, if any, in the given directions. Use the same names that you use for the Slur script entries for *those* regions, i.e., what you specify after the '-r' option for them. If the region has no neighbor in a particular direction, don't give it an entry for that direction. Slur will use the default value that indicates no neighbor.

Put all these entries in a file, make the file executable, and execute it as a shell script from the relevant realm's directory (i.e., the common parent directory to 'rdldir' and 'regdir').

The main thing that makes the Slur script tricky is that it is the place where you specify the connectivity between the regions. Be very careful. It's tedious and easy to mess up. This is the primary thing that Gru is being designed to get rid of.

Running the Slur script will turn make .rdl files in 'rdldir' for all the .reg

files in 'regdir' (presuming that you set everything up right). If Slur crashes it probably means you have a class.dat file that doesn't match a .reg file.

Once you have the .rdl files you need to run Riddle. To do this you need to set up two files. The first (called 'foo_regions.rdl' in the above description) assigns the global ID numbers to all the regions. We need to assign these manually, unfortunately. Here is what the 'foo_regions.rdl' file should look like:

```
regionName1     = 5000;
regionName2     = 5001;
regionName3     = 5002;

etc.
```

In other words, a series of lines of the form:

<name> = <number>;

where all the <name>s are (EXACTLY) the names of the regions you created in the Slur script and the <number>s are some set of global ID numbers that haven't been used yet. Currently Populopolis runs from 1000 to 1200+, Dnalsi runs from 2000 to 2057 (so far), and some of the junk regions have values less than 1000. You just have to pick a starting point and start counting. I would advise beginning a new realm at some multiple of 1000.

All these lines do is assign values to these symbols so you only have to keep track of the numbers in one place.

The second file you have to set up for Riddle (the 'foo.rdl' file described above) consists of the following:

```
include "/u0/chip/microcosm/riddle/beta_objects.rdl";
include "foo_regions.rdl";

include "rdldir/regionName1.rdl";
include "rdldir/regionName2.rdl";
include "rdldir/regionName3.rdl";

etc.
```

In other words: include the 'beta_objects.rdl' file from the Riddle directory, then include the 'foo_regions.rdl' file that you just created followed by all the .rdl files in 'rdldir'.

The 'beta_objects.rdl' file defines all the classes of objects for Riddle. You can look at it if you want. It's a little bizarre as it has grown rather than being well planned.

Once you have the 'foo.rdl' file generated, you can crank it through Riddle with the command:

```
/u0/chip/microcosm/riddle/riddle -o ofile foo.rdl
```

where 'ofile' is the name of the output file you want. This final output file is the end product for us. Give it an appropriate name and put it in the directory /u0/chip/realms on Kessel (not Moth), then tell Janet to go snarf it up and run it through Genesis. Once she's done this you have to get into the host database with Twiddle and connect it to the rest of the world manually.

Then login as an avatar and walk around. If you are lucky you’re done!

Whew!