Using QTk

The **QTk** module implements a description-based abstraction to help programmers efficiently build graphical user interfaces. Windows are built in a declarative way, expressing the widgets that compose the window along with their geometry. Widgets can be dynamically controlled by handles. Miscellaneous facilities are also provided to facilitate the development of an application's GUI. The **QTk** module can be loaded by:

- From the OPI:
  `declare [QTk] = {Module.link [x-oz://system/wp/QTk.ozf]}`
- In a functor, in the import part:
  `import QTk at x-oz://system/wp/QTk.ozf`

The QTk module is based on using *descriptions* to define user interfaces. The description approach is particularly useful in a symbolic language such as Oz that allows easy and concise creation of data structures. QTk uses record values, which are well supported by Oz, hence, a description is a record. For example, the following code defines a record and references it in D:

```
declare
D = td(button(text:"Show"
              action: proc {PESO}
                         {Show ´Hellow World´}
                      end)
       button(text:"Close"
              action:toplevel#close))
```

The record with label `td` has two fields that themselves contain records defining two buttons that are labelled Show and Close. These buttons are linked to the actions of displaying `Hello World` and closing the window.

Since learning by examples is a good practice, to really see what you can do with QTk, it is good idea to see **The Prototyper Application**. The Prototyper is an application that displays commented Oz code. The Run button executes the code. The user can edit the code and click Run again to see its modifications instantly. The Prototyper proposes a list of topics of interest for QTk. The Prototyper application can be started from the OPI by:

```
declare
[Prototyper] = {Module.link ["x-oz://system/wp/Prototyper.ozf"]}
{Prototyper.run}
```

All basic functionality of QTk and all available widgets are described. Playing with the QTk Prototyper is a nice way of discovering QTk. The code above will display what is shown in Figure 1.

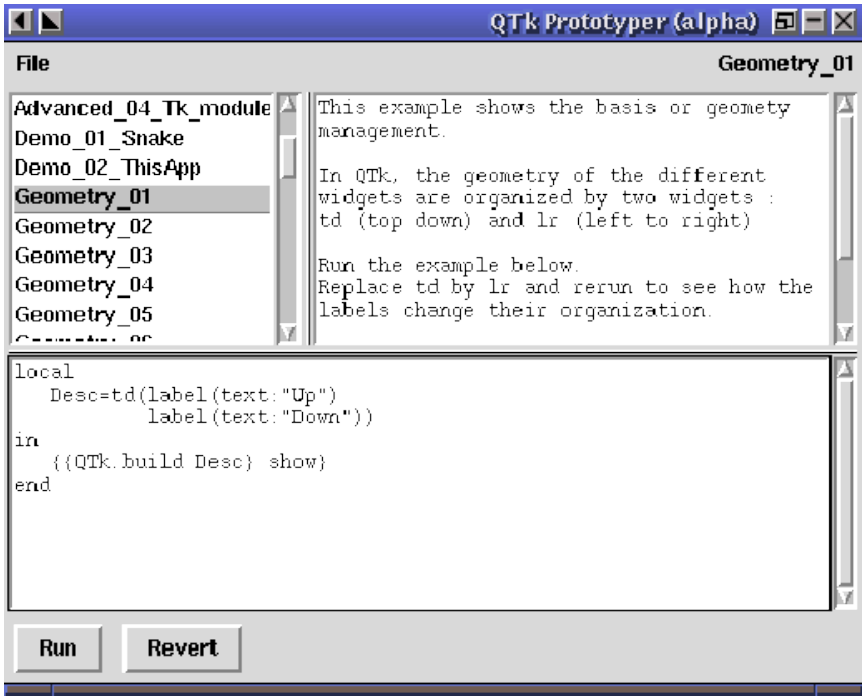

Figure 1: The QTk Prototyper

Have a look at `Demo 04_Canvas` of the Prototyper. It is a good starting point. This example not only gives ideas about building the map, but also updating the field after an agent action.

Example `Demo 01_Snake` gives you a lot of hints about how to implement the bonus of the project. Actually, we are giving already too much help.

The following page for more information about QTk:
`http://www.mozart-oz.org/documentation/mozart-stdlib/wp/qtk/html/index.html`