Advanced Programming (I00032) 2016
Railway Control

Assignment 7

Goal

In this assignment you will make a larger program using the `iTask` and SVG DSL's. This should give you more experience in the use of these DSL's. Please pay some attention to the rationale behind the design of these languages, what works convenient, and the DSL problems you encounter.

There is a small example of SVG on Blackboard and we will explain it in the tutorial. Most language elements of SVG contain a brief explanation near their definition in the `dcl` module, use the IDE (or `grep`) to find these definitions. This is very similar to `iTask`. For many libraries `https://cloogle.org/` is very useful.

Part of your final mark

To reward your efforts in making this larger assignment it can contribute to your final mark of this course. When your mark for the exam is at least 5, the final mark can be improved by averaging with this exercise. The weight of this exercise will be 25%. The marks will only be averaged if this improves your result.

1 A Rail Way Control System

In this assignment you will make a railway control system. This task based system has different tasks. It is not required that all of these tasks can be performed concurrently.

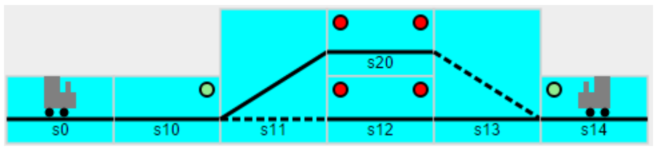
1.1 Track Designer

The first task is the design of the railway system. Our railway system consists of two different elements: *sections* and *points*.

A section is a straight piece of rail. Each section has a label. In our maps it will always run from left to right. The ends of a section are called `left` and `right`. At both ends of a section there is optionally a signal. Sections can be used by trains in both directions. In the layout depicted below `s0`, `s10`, `s12`, 14, and `s20` are sections.

A point can switch between connecting different sections. In the figure below point `s11` connects `s10` and `s20`. After a switch of state the point connects `s10` and `s12`. The end of a point that can connect to two different elements is called the `stem`. The other ends are called `plus` and `minus`. The track between `stem` and `plus` is the main path through the point. There are no signals on a point.

The map of a railway can look like:



There can be trains on elements of the railway. In our system the trains occupy always exactly one element. If desired this can be simplified such that trains are always on a section and passes points in zero time.

The layout of railway systems can in general be pretty complicated graphs. We propose to simplify this to a rectangular grid were sections occupy one cell. Points occupy two cells above each other, one for the main track and one for the branching track. Note that there are for different directions for the branching track: NE, SE, SW and NW.

The track designer can make a collection of these elements. An image of the layout has to be generated automatically from this collection of rail elements. It can look like the figure above, but you can design your own format. Such an image can be made in HTML (like the tic-tac-toe example from the basic-examples), but using the SVG is much easier. Hence, we recommend using SVG.

A simple representation for these elements can be (you are free to use anything else):

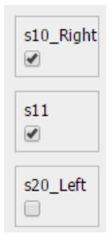
```
:: Section =
  { sLabel       :: String
  , sPosition    :: Position
  , sLeftSignal  :: Bool
  , sRightSignal :: Bool
  }
:: Point =
  { pLabel       :: String
  , pPosition    :: Position
  , pOrientation :: Orientation
  }
:: Position = {x :: Int, y :: Int}
:: Orientation = NE | SE | SW | NW
```

1.2 Track Controller

For the track controller the track design is fixed. The task of the controller is to set the signals and points in such a way that all trains can move safely to their destination element. Typically the destination will be the other end of the layout.

In order to control the signals and points the controller can click on their representation on the image. Clicking a signal or point will toggle it. In addition the system generates a list of checkboxes to toggle the signals and points. The state of these checkboxes should change based on the clicks on the map. Use the `onclick` attribute of images to toggle signals and points.

This list of checkboxes should be derived from the list of elements. A fragment of the checkboxes generated for the layout above can look like:



s10_Right ✓

s11 ✓

s20_Left ☐

The controller also sets the initial position and destination of trains.

1.3 Train Driver

Finally there is one driver for each train. The driver can move her train to the subsequent element of the railway. A good train driver will wait until a signal on the end of the current section turns green.

1.4 Miscellaneous

It is not necessary to implement collision detection, ignoring red signals and other violations of a well behaving railway system. This is of course allowed.

It is convenient to write one program that contains the tasks for the different users. During startup the user selects her role in the system. By their nature the controller and the driver work concurrently. You may assume that the designer will not change the railway during its operation.

Deadline

To give your time to make this exercise there will be no lecture next week. The two weeks after that are a break for exams. There is no exam for this course in those weeks. The deadline for this exercise is November 1, 2016, 23:59h.

Hand in the code, and a very brief description of the main decisions made in the implementation and some screen shots of your system. This should make it easier to mark your work. It is recommended to make this assignment with one partner, more than one partners is not allowed.

Please contact us when you encounter problems.