Reactive Programs

1 March 2023
Where are we?
Where are we?
Traffic-light world
All traffic lights are the same size and position on the screen.

What distinguishes them?

Asking this helps us think about data
All traffic lights are the same size and position on the screen.

How do we get from one to the other?

Asking this helps us think about functions. We need more(?), but this is a good start. i.e. not just how, but when.
Data

Data definition
Examples
Template

Functions

Signature
Docstring
Examples
Body
Data definition

Examples
Template

Functions

Signature
Docstring
Examples
Body
data TrafficLight:
   ...
end
data TrafficLight:
  | green
  | yellow
  | red
end
data TrafficLight:
  | green
  | yellow
  | red
end

TL-GREEN = green
TL-YELLOW = yellow
TL-RED = red

For this data definition, the examples are so trivial we can skip them, but you saw in the pipeline lab how helpful it can be to have examples when you have a lot of possibilities!
data TrafficLight:
  | green
  | yellow
  | red
end
<table>
<thead>
<tr>
<th>Data definition</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Signature</td>
</tr>
<tr>
<td>Template</td>
<td>Docstring</td>
</tr>
<tr>
<td>Examples</td>
<td>Examples</td>
</tr>
<tr>
<td></td>
<td>Body</td>
</tr>
</tbody>
</table>
data TrafficLight:
  | green
  | yellow
  | red
end
data TrafficLight:
  | green
  | yellow
  | red
end

fun trafficlight-fun(tl :: TrafficLight) -> ...:
data TrafficLight:
  | green
  | yellow
  | red
end

fun trafficlight_fun(tl :: TrafficLight) -> ...
  doc: "TrafficLight template"
  cases (TrafficLight) tl:
    | green => ...
    | yellow => ...
    | red => ...
  end
where:
  trafficlight_fun(green) is ...
  trafficlight_fun(yellow) is ...
  trafficlight_fun(red) is ...
end
As we saw* last class, Pyret has a mechanism for supporting interactive programs, called a reactor. It allows us to model the real word**

To use it, first write

```
include reactors
```

*Well, we didn’t *actually* see it, but just go with it 😊
**Not the television show!
reactor:
  init: \textit{initial-state}
  to-draw: \textit{draw-function}
  \textbf{event-type}: event-function
end

\textbf{Event-type} allows us to ask \texttt{“when?”} do we want to call the \textit{draw-function}. The 4\textsuperscript{th} dimension exists!
reactor:
  init: initial-state
to-draw: draw-function
event-type: event-function
end

“reactor?” this allows us our program to react to events. What kind of events? Something like…
Less nuclear reactor; more person-that-reacts to something.
reactor puts all the pieces together to start things up and allows the program to react to events...
2.1.11.21 Reactor Expressions

```plaintext
<reactor-expr> ::= reactor : init : <expr> [, <option-name> : <expr>]>* end
<option-name> ::= on-tick
               | on-mouse
               | on-key
               | to-draw
               | stop-when
               | title
               | close-when-stop
               | seconds-per-tick
```

Events!
initial state
some event happens…
next state
next-state

now the current state
some event happens…
next-state

now the current state
some event happens…
next-state

now the current state
reactor:
  init: initial-state,
  to-draw: draw-function,
  event-type: event-function
end
reactor:
  init: green,
  to-draw: draw-function,
    event-type: event-function
end
reactor:
  init: green,
  to-draw: draw-light,
  event-type: event-function
end

We haven’t written this; add it to our wishlist!
reactor:
  init: green,
  to-draw: draw-light,
  on-tick: next-light
end

Another function for the wishlist!
So far…

# TrafficLight data
# - definition
# - examples
# - template

# define reactor

# Wishlist:
# - fun draw-light...
# - fun next-light...
fun draw-light(tl :: TrafficLight) -> Image:

    ...

end
fun draw-light(tl :: TrafficLight) -> Image:
    ...
end

fun next-light(tl :: TrafficLight) -> TrafficLight:
    ...
end
fun **draw-light**(tl :: TrafficLight) -> Image:
    doc: "Draw a circle of the given color, rendering a traffic light"
    ...
end

fun **next-light**(tl :: TrafficLight) -> TrafficLight:
    ...
end
fun **draw-light**(tl :: TrafficLight) -> Image:
   doc: "Draw a circle of the given color, rendering a traffic light"
   ...
end

fun **next-light**(tl :: TrafficLight) -> TrafficLight:
   doc: "Produce the next light in the sequence green, yellow, red"
   ...
end
Data

Data definition
Examples
Template

Functions

Signature
Docstring
Examples
Body
fun draw-light(tl :: TrafficLight) -> Image:
  doc: "Draw a circle of the given color, rendering a traffic light"
  ...
where:
  draw-light(green) is circle(20, "solid", "green")
  draw-light(yellow) is circle(20, "solid", "yellow")
  draw-light(red) is circle(20, "solid", "red")
end

fun next-light(tl :: TrafficLight) -> TrafficLight:
  doc: "Produce the next light in the sequence green, yellow, red"
  ...
end
fun **draw-light**(tl :: TrafficLight) -> Image:
   doc: "Draw a circle of the given color, rendering a traffic light"
   ...
where:
   draw-light(green) is circle(20, "solid", "green")
   draw-light(yellow) is circle(20, "solid", "yellow")
   draw-light(red) is circle(20, "solid", "red")
end

fun **next-light**(tl :: TrafficLight) -> TrafficLight:
   doc: "Produce the next light in the sequence green, yellow, red"
   ...
where:
   next-light(green) is yellow
   next-light(yellow) is red
   next-light(red) is green
end
Data

Data definition
Examples
Template

Functions

Signature
Docstring
Examples
Body
Starter code:

tinyurl.com/2023-03-01-tl-starter
Code:

tinyurl.com/2023-03-01-tl
Screensaver
Code:

tinyurl.com/2023-03-01-bounce
Acknowledgments

This lecture incorporates material from:

W. Daniel Hillis, *The Pattern on the Stone*

Marc Smith, Vassar College

Laney Strange, Northeastern University