Functional reactive animation

Anthony C. Daniels

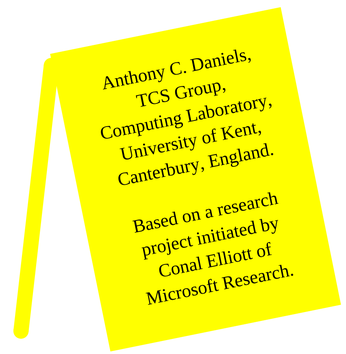
Fran is a language for creating interactive animations. It is implemented as a library in the functional language Haskell. Let's see what it can do!

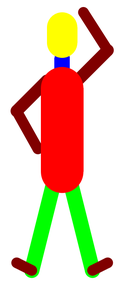
Here is a simple Fran program,

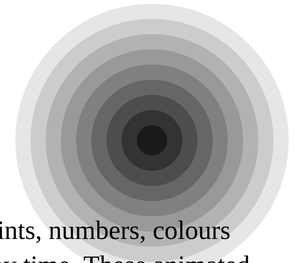
```
move pos circle
  where pos = vector2XY (sin time) 0
```

It describes a circle moving horizontally back and forth.

I hope you have enjoyed this presentation of Fran. This poster was created entirely using Fran, by taking a snapshot in time of a large animation. To the right is the program that produced this poster.





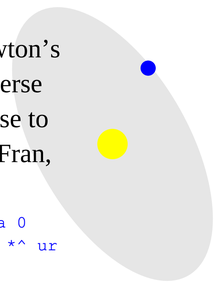


In Fran, all values--such as points, numbers, colours and images--are parameterised by time. These animated values are called behaviours. Animations are constructed by composing behaviours, and the underlying language Haskell can be used to structure your programs. Here is a program that yeilds concentric fading circles,

```
foldl1 over [withColor (gray x) (stretch x circle)
               | x <- [0, 0.1 .. 1.0] ]
```

"foldl1" is a standard Haskell function and "foldl1 over" will overlay a list of images; in this case, gray circles.

We will now demonstrate Newton's celebrated result--that an inverse square law of gravity gives rise to elliptical planetary orbits--in Fran,

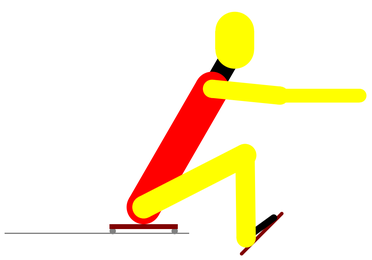


```
orbit = p0 + integral v 0
  where v  = v0 + integral a 0
        a  = (k / (mr ^ 2)) *^ ur
        mr = magnitude r
        ur = unit r
        r  = origin2 -  orbit
anim = move orbit planet `over` sun
```

Again, we see that Fran is capable of solving systems of integral equations that are described by mutually recursive definitions.

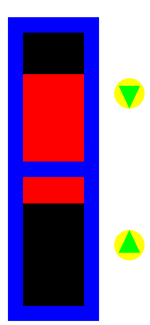
This animation of a rower can be duplicated four times and placed together in a crew as follows,

```
crew = besides fourRowers
         `over`
       cox `over` boat
  where
  fourRowers = replicate 4 rower
```



Fran provides facilities for interactivity via mouse and keyboard input. In this program, the ball follows the mouse as if it were on a spring obeying Hooke's law,

```
followMouse t0 = move p ball
  where p = integral v t0
        v = integral a t0
        a = (mousev t0 - p) - 0.5 *^ v
```

Notice that the definitions for the position (p), velocity (v), and acceleration (a) of the ball are mutually recursive. They describe integral equations which Fran solves numerically.

The combinators for reactivity are powerful enough to describe many complex real-time systems. This is ideal for prototyping systems, such as this lift simulation program.