Exemplo 4 - Animações

Objetivos:

- O objetivo é exemplificar o uso de eventos da classe **`user`** para implementar animações.

Comportamento Esperado:

Durante a execução, um corredor atravessa a tela de um lado ao outro, até alcançar a linha de chegada.

Execução:

- Baixe o código do exemplo **aqui**.
- Execute a aplicação e observe o comportamento do corredor.

Considerações:

Precisamos implementar animações com o que a API de Lua para TV Digital oferece. Repare que não podemos simplesmente criar um *loop* dentro do tratador de eventos para animar o corredor, pois já vimos que assim, enquanto o corredor estivesse sendo animado, nenhum outro evento seria tratado. O leitor pode ter reparado que não há nenhum suporte a *multi-threading*, portanto, não podemos simplesmente *disparar* o corredor em uma nova *thread*.

A solução é através de dois recursos ainda não explorados: a função **uptime** e a classe de eventos **`user`**.

A idéia é postar eventos da classe *user* enquanto o corredor não alcança a linha de chegada. Assim, o tratador calcula quanto tempo decorreu a cada chegada de evento e baseado nisso calcula quanto o corredor deve se mover.

Vamos direto ao código do tratador de eventos:

```
function handler (evt)
    -- a animacao comeca no *start* e eh realimentada por eventos da classe *user*
    if (evt.class == 'ncl' and evt.type == 'presentation' and evt.action == 'start') or
    (evt.class == 'user') then
        local now = event.uptime()

        -- movimenta o corredor caso tempo ja tenha passado
        if evt.time then
            local dt = now - evt.time
            runner.x = runner.x + dt/50
        end

        -- caso nao tenha chegado a linha de chegada, continua dando ciclos a animacao
        if runner.x < END then
            event.post('in', { class='user', time=now })
        end

        -- muda o frame do corredor a cada 5 pixels
        runner.frame = math.floor(runner.x/5) % 2

        redraw()
    end
end
```

O primeiro *if* filtra apenas o evento de *start* e eventos da classe *user* que serão usados para realimentar a animação.

A linha seguinte chama a função **uptime** que retorna o tempo passado desde o início da execução do NCLua.

No caso de um evento da classe *user*, o campo `evt.time` terá o tempo do último ciclo da animação. Em cima dele é calculada a diferença para o tempo atual e a posição do corredor é atualizada.

A posição do corredor é então testada e caso este ainda não tenha alcançado a linha de chegada, um novo evento da classe *user* é postado com o tempo atual.

A animação alterna dois quadros do corredor para dar a sensação de que as suas pernas que o fazem se mexer. O quadro corrente é calculado de forma a se alternar entre `0` e `1` a cada 5 pixels de movimento.

No fim do tratador a função de redesenho é chamada, cuja definição é a que segue:

```
function redraw ()
```

```
    -- fundo
    canvas:attrColor('black')
    canvas:drawRect('fill', 0,0, dx,dy)

    -- linha de largada e chegada
    canvas:attrColor('white')
    canvas:drawRect('fill', INI,0, 8,dy)
    canvas:drawRect('fill', END,0, 8,dy)

    -- corredor
    local dx2 = runner.dx/2
    runner.img:attrCrop(runner.frame*dx2,0, dx2,runner.dy)
    canvas:compose(runner.x, runner.y, runner.img)
    canvas:flush()
end
```

O fundo é desenhado em preto, em seguida, as linhas de largada e chegada em branco. Por fim a imagem do corredor é desenhada sobre o canvas. A função **compose** pode receber como parâmetros partes da imagem que devem ser compostas sobre o canvas. No exemplo apenas uma das duas metades da imagem é desenhada a cada 5 pixels.

Repare que a imagem do corredor é uma tira com os quadros da animação.

Até aqui é importante que o leitor esteja familiarizado com os seguintes conceitos:

- classe de eventos *user*
- uptime
- função de animação
- a função compose

Exercícios:

- Colocar mais um frame na imagem para suavizar o movimento do boneco.
- Incluir um corredor de outra cor abaixo do atual.
- Randomizar as velocidades dos corredores.
- Exibir algo em NCL de acordo com o vencedor da corrida.
- Utilizar co-rotinas de Lua para implementar as animações.