Exemplo 3 - Gráficos e Controle Remoto

Descrição:

Um jogo simples onde o usuário deve mover o macaco até a banana. Quando isso ocorre é exibido um botão indicando o fim do jogo.

Objetivos:

- Introduzir a classe de eventos **`'key'`**.
- Introduzir o uso do pacote gráfico **`canvas`**.

Execução:

- Baixe o código do exemplo **aqui**.
- Execute o exemplo e movimente o macaco para todos os lados. Observe que o macaco pode sair da tela e rodear a banana.

Considerações:

Esse exemplo deixa um pouco de lado a integração NCL-Lua e mostra uma aplicação mais auto-contida em que a comunicação com o documento NCL ocorre apenas no início da exibição do documento e no fim do jogo.

O jogo é iniciado assim que o documento é carregado:

```
<port id="entryPoint" component="lua"/>
```

Quando o macaco alcança a banana, o NCLua sinaliza o início da âncora *fim* e o documento exibe o botão de vitória:

```
<link xconnector="onBeginStart">
    <bind role="onBegin" component="lua" interface="fim"/>
    <bind role="start" component="win"/>
</link>
```

Nos exemplos anteriores, o código da aplicação estava concentrado no documento NCL, agora vemos que praticamente todo o código está dentro do NCLua.

As primeiras linhas do código NCLua criam objetos para representar o macaco e a banana:

```
local img = canvas:new('monkey.png')
local dx, dy = img:attrSize()
local monkey = { img=img, x=10, y=10, dx=dx, dy=dy }
```

Já na primeira linha de código há uma referência para o pacote **canvas**, com o qual todas as operações gráficas são efetuadas. A chamada a **canvas:new** do trecho acima carrega a imagem `monkey.png` que é guardada na variável `img`. A última linha cria uma tabela que guarda as propriedades do macaco: sua imagem, posição (em `x`,`y`) e tamanho (em `dx`,`dy`). Esse uso de tabelas é similar ao de objetos em linguagens orientadas a objetos e é bastante recorrente em Lua. O código para representar a banana é bastante parecido ao trecho acima.

A variável `canvas` referencia a região NCL destinada ao NCLua. Isso significa que a variável `canvas` terá valor igual a `nil` em um NCLua cujo documento NCL correspondente não definiu uma região para ele.

Em seguida é definida a função de redesenho que será chamada a cada movimento feito pelo macaco:

```
function redraw ()
    canvas:attrColor('black')
    canvas:drawRect('fill', 0,0, canvas:attrSize())
    canvas:compose(banana.x, banana.y, banana.img)
    canvas:compose(monkey.x, monkey.y, monkey.img)
```

```
    canvas:flush()
end
```

Nessa função, usando o método **canvas:drawRect**, é desenhado um retângulo que pinta a tela toda de preto (cor configurada anteriormente). As duas linhas seguintes utilizam o método **canvas:compose** para desenhar a banana e o macaco sobre o canvas. Por fim o canvas é atualizado com uma chamada à **canvas:flush**.

A função `collide`, definida em seguida, possui a assinatura:

```
function collide (A, B)
```

Essa função recebe dois objetos tais como o macaco e a banana e retorna se ambos estão se superpondo. Utilizando a mesma estrutura em tabela, novos personagens podem ser incluídos no jogo e usados pela mesma função de colisão.

Repare que até aqui apenas foram definidos os objetos (macaco e banana) e funções (`redraw` e `collide`), mais uma vez, o *script* é apenas um inicializador e ações são tomadas somente em resposta a eventos.

Por fim é definida a função de tratamento de eventos onde toda a ação do jogo acontece. Estamos interessados em tratar eventos de pressionamento de tecla, sendo assim, a função deve filtrá-los:

```
if (evt.class == 'key') and (evt.type == 'press') then
    ...
end
```

O tratador de teclas, então, testa o valor da tecla e altera a posição do macaco de acordo:

```
if evt.key == 'CURSOR_UP' then
    monkey.y = monkey.y - 10
elseif evt.key == 'CURSOR_DOWN' then
    monkey.y = monkey.y + 10
elseif evt.key == 'CURSOR_LEFT' then
    monkey.x = monkey.x - 10
elseif evt.key == 'CURSOR_RIGHT' then
    monkey.x = monkey.x + 10
end
```

Um evento de tecla carrega a informação `type` que indica se o botão está sendo pressionado ou soltado; e `key` que indica o valor da tecla em questão.

Após o movimento do macaco, a função se houve colisão com a banana:

```
if collide(monkey, banana) then
    event.post {
        class  = 'ncl',
        type   = 'presentation',
        area   = 'fim',
        action = 'start',
    }
    IGNORE = true
end
```

Caso haja a colisão, o NCLua sinaliza ao formatador que sua âncora *fim* teve início e marca um *flag* (`IGNORE=true`) para que eventos futuros sejam ignorados.

Resumo:

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

- Tabelas Lua como objetos.
- Representação do pacote gráfico e região NCL com a variável `canvas`.
- Código de inicialização de um NCLua.

Exercícios:

- Incluir na cena uma barreira que o macaco não consiga atravessar.
- Após a âncora de fim, ignorar os eventos futuros através de uma ação de pausa comandada pelo NCL.