SIMULADOS & TUTORIAIS

TUTORIAIS FLASH

O Componente TileList

O Componente TileList

Introdução

O componente **TileList** consiste de uma lista composta de linhas e colunas cujo conteúdo é proveniente de uma fonte de dados que poderá ser definida tanto em tempo de projeto quanto em tempo de execução através do ActionScript. Um item refere-se a uma unidade de informação que é armazenada em uma célula do **TileList**. Um item, possui tipicamente uma propriedade **label** e uma propriedade **source**. A propriedade **label** identificará o conteúdo a ser mostrado na célula e a propriedade **source** fornecerá um valor para esse conteúdo.
Você poderá criar uma instância de uma matriz ou recuperar uma de um servidor qualquer. O componente **TileList** possui métodos que criam sua própria fonte de dados, como por exemplo, os métodos **addItem()** e **removeItem()**. Se nenhum dado externo for fornecido para a lista, esses métodos criarão uma instância de fonte de dados automaticamente, que será mostrada utilizando-se o parâmetro **dataProvider** da instância (ex: **List.dataProvider**).

Interação do usuário com o TileList

Um **TileList** exibe cada célula usando um **Sprite** que implementa a interface **ICellRenderer**. Você poderá especificar essa exibição com a propriedade **cellRenderer**. O **CellRenderer** padrão do **TileList** é o **ImageCell**, que mostrará uma imagem (classe, bitmap, instância ou URL) e um rótulo opcional. O rótulo é uma simples linha que sempre é alinhada à base da célula. Um **TileList** só poderá ser rolado em uma direção, horizontal ou verticalmente, nunca nas duas direções simultaneamente.

Quando uma instância de um componente **TileList** estiver com o foco, você poderá utilizar as seguintes teclas para acessar seus itens internos.

Tecla	Descrição
Up e Down	Permite mover o conteúdo de uma linha para cima ou para baixo. Se a propriedade **allowMultipleSelection** estiver configurada para **true**, você poderá utilizar essas teclas em combinação com a tecla **Shift** para selecionar múltiplas células.
Left e Right	Permite mover uma coluna para a esquerda ou para a direita em uma linha. Se a propriedade **allowMultipleSelection** estiver configurada para **true**, você poderá utilizar essas teclas em combinação com a tecla **Shift** para selecionar múltiplas células.
Home	Seleciona a primeira célula de um **TileList**. Se a propriedade **allowMultipleSelection** estiver configurada para **true**, mantendo a tecla **Shift** pressionada em combinação com essa tecla serão selecionadas todas as células a partir da célula atual até a primeira.
End	Seleciona a última célula de um **TileList**. Se a propriedade **allowMultipleSelection** estiver configurada para **true**, mantendo a tecla **Shift** pressionada em combinação com essa tecla serão selecionadas todas as células a partir da célula atual até a última.
Ctrl	Se a propriedade **allowMultipleSelection** estiver definida para **true**, essa tecla permite-lhe selecionar múltiplas células, independente de qualquer ordem.

Quando você adicionar um componente **TileList** a uma aplicação, você poderá mantê-lo acessível aos leitores de tela adicionando as seguintes linhas em ActionScript:

```
import fl.accessibility.TileListAccImpl;
TileListAccImpl.enableAccessibility();
```

Essa operação deve ser feita apenas uma vez, independente do número de instâncias do componente existentes na aplicação.

Os parâmetros do componente TileList

Para que você possa utilizar os recursos disponíveis para esse componente, você precisará conhecer os parâmetros necessários ao seu correto preenchimento para que os resultados sejam os esperados, conforme mostraremos abaixo:

Parâmetro	Descrição
allowMultipleSelection	É um valor booleano que indica se a lista poderá ter múltiplas seleções ou não. O valor padrão é: **false**.
columnCount	Indica o número de colunas que deverão estar parcialmente visíveis na lista. O valor padrão é: **0**.
columnWidth	Indica a largura que deverá ser aplicada a uma coluna da lista, em pixels. O valor padrão é: **50**.
dataProvider	Define o modelo de dados da lista de itens que deverão ser visualizados.
direction	Estabelece um valor que indica se o componente **TileList** deverá rolar horizontal ou verticalmente. O valor padrão é **horizontal**.
enabled	Esse parâmetro é um valor booleano que indica se o componente deverá estar habilitado ou desabilitado quando a aplicação for executada. O valor padrão é **true**.
horizontalLineScrollSize	Um valor que indica a quantidade de colunas que deverão ser roladas para a esquerda ou para a direita quando uma das setas da barra for clicada. Para esse recurso funcionar é necessário que o parâmetro **direction** seja configurado para **horizontal**. O valor padrão desse parâmetro é **4**.
horizontalPageScrollSize	Um valor que indica a quantidade de páginas que deverão ser roladas para a esquerda ou para a direita quando uma das setas da barra for clicada. Para esse recurso funcionar é necessário que o parâmetro **direction** seja configurado para **horizontal**. O valor padrão desse parâmetro é **0**.
rowCount	Estabelece o número de linhas que deverão estar visíveis parcialmente na lista. O valor padrão é: **0**.
rowHeight	Indica a altura que deverá ser aplicada a cada linha da lista, em pixels. O valor padrão é: **50**.
scrollPolicy	Estabelece o método de rolamento de um componente **TileList**. Poderão ser **auto** (scroll automático, dependendo do conteúdo), **on** (scroll ligado) ou **off** (scroll desligado). O valor padrão é: **auto**.
verticalLineScrollSize	Um valor que indica a quantidade de linhas que deverão ser roladas para cima ou para baixo quando uma das setas da barra for clicada. Para esse recurso funcionar é necessário que o parâmetro **direction** seja configurado para **vertical**. O valor padrão desse parâmetro é **4**.
verticalPageScrollSize	Um valor que indica a quantidade de páginas que deverão ser roladas para cima ou para baixo quando uma das setas da barra for clicada. Para esse recurso funcionar é necessário que o parâmetro **direction** seja configurado para **vertical**. O valor padrão desse parâmetro é **0**.
visible	Esse parâmetro é um valor booleano que indica se o componente deverá aparecer visível ou oculto quando a aplicação for executada. O valor padrão é: **true**.

Cada um desses parâmetros possui uma propriedade equivalente em ActionScript com o mesmo nome. E além desses parâmetros, você poderá também encontrar outras propriedades e métodos para esse componente na linguagem ActionScript que você poderá utilizá-los em tempo de execução e criar aplicações mais robustas, conforme descrevemos a seguir:

As propriedades do componente TileList

São as seguintes as propriedades disponíveis para o componente **TileList**:

Propriedade	Descrição
iconField	É uma string que identifica o item do campo que fornecerá o ícone para o item. O valor padrão é: **null**.
iconFunction	Configura uma função a ser utilizada para obter o ícone para o item. O valor padrão é: **null**.
innerHeight	É um número que informa a altura do conteúdo da área, em pixels. Propriedade somente de leitura.
innerWidth	É um número que informa a largura do conteúdo da área, em pixels. Propriedade somente de leitura.
labelField	É uma string que indica um campo em cada item que conterá um rótulo para cada célula do **TileList**. O valor padrão é: **label**.
labelFunction	É uma função que indica os campos de um item que fornecerá o texto do rótulo para uma célula. O valor padrão é: **null**.
maxHorizontalScrollPosition	É um número que indica a posição máxima da barra de rolagem horizontal do conteúdo corrente, em pixels.
sourceField	É uma string que indica o campo do item que fornecerá o caminho de origem para uma célula do **TileList**.
sourceFunction	É uma função a ser utilizada para obter o caminho de origem para uma célula do **TileList**.

Os métodos do componente TileList

São os seguintes os métodos disponíveis para o componente **TileList**:

Método	Descrição
TileList	Cria uma nova instância do componente **TileList**.
getStyleDefinition	Recupera o estilo de mapa padrão para o componente atual.
itemToLabel	Recupera a string que o exibidor deverá mostrar em um objeto de dado baseado nas propriedade**s labelField** e **labelFunction**.
scrollToIndex	Rola a lista para o item do índice especificado.

Os eventos do componente TileList

Embora o **TileList** não disponha de eventos específicos para execução, você poderá utilizar os eventos disponíveis para o mouse com esse componente, tais como: **Click**, **rollOver**, **rollOut**, entre outros, além de eventos de outras classes disponíveis no pacote **fl.controls**.

Veja como utilizar esses recursos nos exemplos mais adiante neste capítulo e no DVD, que acompanha a presente obra.

Criando aplicações com o componente TileList

Para entender melhor como funciona esse componente, mostraremos a seguir alguns exemplos práticos, inclusive em conjunto com outros componentes.

Exemplo 1

Nesse exemplo mostraremos como usar o componente **TileList** da maneira mais simples possível, ou seja, criando tudo manualmente e configurando os parâmetros através do painel **Component Inspector**. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);

2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **TileList** para o palco e posicione-o no centro. Veja na **Figura 17.1** abaixo a aparência original desse componente no palco:

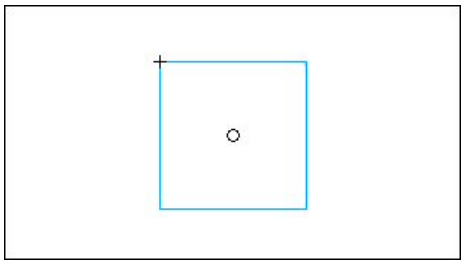

Figura 17.1 – Componente TileList em sua forma original.

3. Nomeie a camada atual para: **componentes**;
4. No painel **Component Inspector**, dê um duplo clique no parâmetro **dataProvider**;
5. Na janela apresentada, preencha-a com os dados mostrados na **Figura 17.2** abaixo:

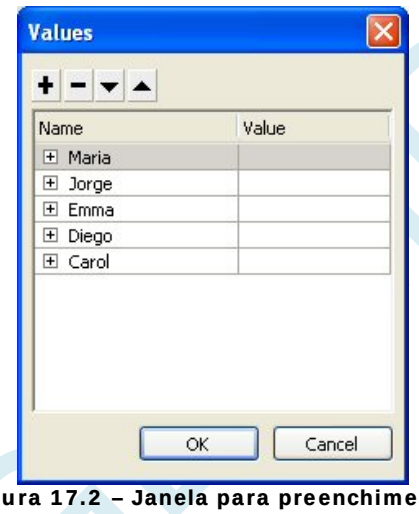

Figura 17.2 – Janela para preenchimento da lista do componente TileList.

6. Feito isso, clique em **OK** para confirmar;
7. Execute a aplicação e confira o resultado com o mostrado na **Figura 17.3** a seguir:

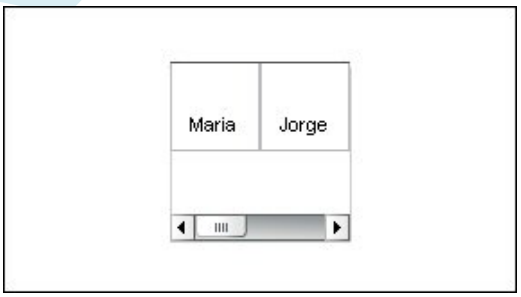

Figura 17.3 – Resultado do exemplo após sua execução.

Observe que a barra de scroll horizontal foi criada automaticamente, tendo em vista que o parâmetro **scrollPolicy** estar configurado para **auto**.

*****

Exemplo 2

Nesse exemplo, embora ainda utilizando o processo manual, alteraremos alguns dos parâmetros do painel **Component Inspector** e veremos como eles se comportam após a execução da aplicação. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **TileList** para o palco;
3. Nomeie a camada atual para: **componentes**;
4. Abra o painel **Component Inspector**, e dê um duplo clique no parâmetro **dataProvider**, e preencha a janela apresentada conforme mostra a **Figura 17.4** a seguir:

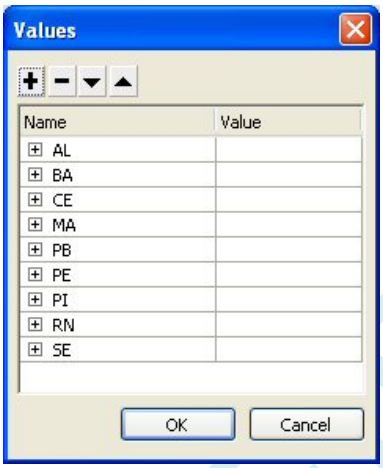

Figura 17.4 – Janela para preenchimento da lista do componente TileList.

5. Clique em **OK** para confirmar;
6. Altere o parâmetro **direction** para: **vertical**;
7. O seu projeto deverá fica igual ao mostrado na **Figura 17.5** abaixo:

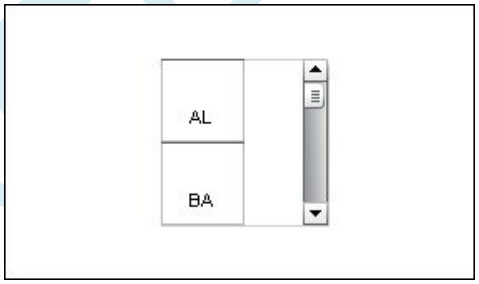

Figura 17.5 – Componente TileList durante o projeto.

8. Altere o parâmetro **columnCount** para: **3**. Esse parâmetro indica o número de colunas que serão visualizadas parcialmente no **TileList**. Embora essa alteração não seja mostrada durante o projeto, ela poderá ser vista após a execução da aplicação;
9. Altere o parâmetro **rowCount** para: **1**. Esse parâmetro indica o número de linhas que serão visualizadas parcialmente no **TileList**. Embora essa alteração não seja mostrada durante o projeto, ela poderá ser vista após a execução da aplicação;
10. Altere o parâmetro **verticalLineScrollSize** para: **50**. Esse parâmetro indica o número de pixels que serão rolados a cada clique na barra de rolagem do **TileList**. Nesse caso, cada linha será rolada totalmente para cada item da lista, e não uma parte de cada linha;
11. Execute a aplicação e confira o resultado com o da **Figura 17.6** abaixo:

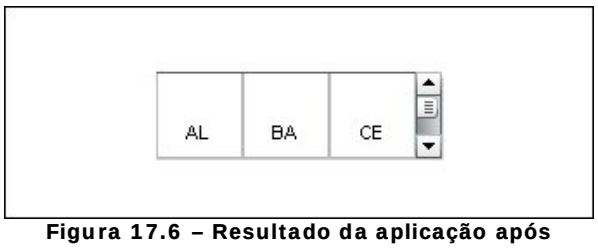

Figura 17.6 – Resultado da aplicação após a sua execução.

Observe que a cada clicada em uma das setas uma linha de informações é mostrada totalmente e não parcialmente. Isso porque o número de pixels de cada linha (**rowHeight**) é igual ao número de pixels da rolagem vertical (**verticalLineScrollSize**), ou seja, **50**.

*****

Exemplo 3

Nesse exemplo criaremos uma aplicação com o componente **TileList**, cuja fonte de dados para o seu preenchimento será proveniente de uma matriz de cores em forma de movie clips. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **TileList** para o palco;
3. Abra o painel **Properties**, e dê um nome para ele: **lista**, por exemplo;
4. Nomeie a camada atual para: **componentes**;
5. Crie uma nova camada e chame-a de: **ações**;
6. Clique no primeiro frame dessa camada e insira o seguinte código:

```
stop();
import fl.data.DataProvider;
import flash.display.DisplayObject;
//
var quadrado:Array = new Array();
//
var cores:Array = new Array(0x00000, 0xFF0000, 0x0000CC, 0x00CC00, 0xFFFF00);
var nomesCores:Array = new Array("Preto", "Vermelho", "Azul", "Verde", "Amarelo");
//
var dp:DataProvider = new DataProvider();
//
for (var i=0; i < cores.length; i++) {
    quadrado [i] = new MovieClip();
    // Desenha uma caixa com a próxima cor da matriz.
    desenhaQuadrado(quadrado[i], cores[i]);
    dp.addItem( {label: nomesCores[i], source: quadrado [i]} );
}
lista.dataProvider = dp;
lista.columnWidth = 110;
lista.rowHeight = 130;
lista.setSize(230,150);
lista.move(150, 150);
lista.setStyle("contentPadding", 5);
//
function desenhaQuadrado(box:MovieClip,color:uint):void {
    box.graphics.beginFill(color, 1.0);
    box.graphics.drawRect(0, 0, 100, 100);
    box.graphics.endFill();
}
```

Vejamos os comentários para esse código:

OBS.: Em todos os exemplos do presente capítulo incluiremos o comando **stop()** na primeira linha do código. Esse comando tem como objetivo interromper a aplicação

quando a mesma for executada, mesmo que a aplicação não possua mais de um frame na sua linha do tempo. Consideramos um bom hábito essa prática. Portanto, não comentaremos mais essa linha nos próximos exemplos.

Nas linhas:

```
import fl.data.DataProvider;
import flash.display.DisplayObject;
```

Importamos as classes necessárias para que possamos utilizar adequadamente os recursos necessários para nossa aplicação.

Na linha seguinte:

```
var quadrado:Array = new Array();
```

Criamos uma instância da classe **Array** e definimos uma matriz vazia para ela, e armazenamos na variável **quadrado**.

Nas linhas abaixo:

```
var cores:Array = new Array(0x00000, 0xFF0000, 0x0000CC, 0x00CC00, 0xFFFF00);
var nomesCores:Array = new Array("Preto", "Vermelho", "Azul", "Verde", "Amarelo");
```

Criamos uma instância de uma matriz chamada **cores**, para armazenar os códigos hexadecimais das cores, e logo em seguida, criamos uma outra instância chamada **nomesCores** para armazenar os nomes dos respectivos códigos das cores.

Na linha:

```
var dp:DataProvider = new DataProvider();
```

Criamos uma instância da classe **DataProvider** utilizando o seu construtor, chamada **dp**, para armazenarmos a fonte de dados que deverá preencher o componente **TileList**.

No bloco de código:

```
for(var i=0; i < cores.length; i++) {
    quadrado [i] = new MovieClip();
    // Desenha uma caixa com a próxima cor da matriz.
    desenhaQuadrado(quadrado [i], colors[i]);
    dp.addItem( {label:nomesCores[i], source: quadrado [i]} );
}
```

Criamos um loop cujo intervalo varia de 0 até o comprimento da matriz **cores**, de forma a realizar as seguintes tarefas:

- A cada passagem do loop, é criado um movie clip e armazenado na variável **quadrado**, declarada anteriormente como uma matriz vazia.
- É chamada a função **desenhaQuadrado** criada mais adiante, que por sua vez desenha uma caixa com 100 pixels de lado e a respectiva cor da matriz **cores**.
- O método **addItem()** do **DataProvider** (**dp**) declarado anteriormente, armazena cada quadrado desenhado através da propriedade **source** com o seu respectivo rótulo da cor proveniente da matriz **nomesCores**.

No bloco de código abaixo:

```
lista.dataProvider = dp;
lista.columnWidth = 110;
lista.rowHeight = 130;
lista.setSize(230,150);
lista.move(150, 150);
```

```
lista.setStyle("contentPadding", 5);
```

Preenchemos o componente **TileList** (instância: **lista**) com o conteúdo armazenado na variável **dp**, através da sua propriedade **dataProvider**. Em seguida definimos:

- A largura de cada coluna (**columnWidth**).
- A altura da linha (**rowHeight**).
- O seu tamanho (**setSize**).
- A sua localização no palco (**move**).
- E finalmente a distância entre a borda da instância do **TileList** e o seu conteúdo (**contentPadding**).

No código a seguir:

```
function desenhaQuadrado(box:MovieClip,color:uint):void {
        box.graphics.beginFill(color, 1.0);
        box.graphics.drawRect(0, 0, 100, 100);
        box.graphics.endFill();
}
```

Criamos uma função **desenhaQuadrado()**, de forma que quando for executada faça o seguinte:

- Define a cor de preenchimento e a espessura da borda do objeto a ser desenhado.
- Desenha o objeto a partir das coordenadas 0,0 e dimensões 100 x 100 pixels.
- Conclui o preenchimento do objeto.

7. Execute a aplicação e confira o resultado com o mostrado na **Figura 17.7** abaixo:

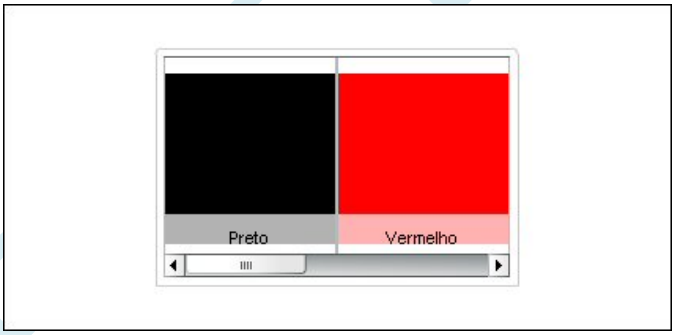

Figura 17.7 – Componente TileList preenchido com caixas coloridas utilizando matrizes para criá-las.

*****

Exemplo 4

Nesse exemplo criaremos um aplicação com o componente **TileList** em tempo de execução usando o ActionScript, cuja fonte de dados para preenchimento são instâncias de vários outros componentes, tais como: **ColorPicker**, **ComboBox**, **NumericStepper** e **CheckBox**, que também serão criados em tempo de execução. Para isso, será criada uma matriz de nome **dp** da classe **DataProvider** contendo os rótulos e os nomes dos respectivos componentes. Além das propriedades e métodos conhecidos, utilizamos também o método **sortItemsOn()** para classificar o conteúdo do **TileList** pelos seus rótulos. Vejamos como isso é feito:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste os seguintes componentes para a biblioteca;

- **Um ComboBox.**
- **Um CheckBox.**
- **Um ColorPicker.**
- **Um NumericStepper.**
- **Um TileList.**

3. Nomeie a camada atual para: **ações**;
4. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
import fl.controls.CheckBox;
import fl.controls.ColorPicker;
import fl.controls.ComboBox;
import fl.controls.NumericStepper;
import fl.controls.TileList;
import fl.data.DataProvider;
//
var meuColorPicker:ColorPicker = new ColorPicker();
var meuComboBox:ComboBox = new ComboBox();
var meuNumericStepper:NumericStepper = new NumericStepper();
var meuCheckBox:CheckBox = new CheckBox();
var meuTileList:TileList = new TileList();
//
var dp:Array = [
   {label:"ColorPicker", source:meuColorPicker},
   {label:"ComboBox", source:meuComboBox},
   {label:"NumericStepper", source:meuNumericStepper},
   {label:"CheckBox", source:meuCheckBox},
];
meuTileList.dataProvider = new DataProvider(dp);
meuTileList.columnWidth = 110;
meuTileList.rowHeight = 100;
meuTileList.setSize(280,130);
meuTileList.move(150, 150);
meuTileList.setStyle("contentPadding", 5);
meuTileList.sortItemsOn("label");
addChild(meuTileList);
```

Vejamos os comentários para esse código:

Nas linhas de código abaixo:

```
import fl.controls.CheckBox;
import fl.controls.ColorPicker;
import fl.controls.ComboBox;
import fl.controls.NumericStepper;
import fl.controls.TileList;
import fl.data.DataProvider;
```

Importamos as classes dos respectivos componentes para que possamos criá-los em tempo de execução.

Nas linhas seguintes:

```
var meuColorPicker:ColorPicker = new ColorPicker();
var meuComboBox:ComboBox = new ComboBox();
var meuNumericStepper:NumericStepper = new NumericStepper();
var meuCheckBox:CheckBox = new CheckBox();
var meuTileList:TileList = new TileList();
```

Criamos novas instâncias de cada um dos componentes que iremos utilizar na nossa aplicação através de seus respectivos construtores.

Nas linhas:

```
var dp:Array = [
   {label:"ColorPicker", source:meuColorPicker},
   {label:"ComboBox", source:meuComboBox},
   {label:"NumericStepper", source:meuNumericStepper},
   {label:"CheckBox", source:meuCheckBox},
];
```

Criamos uma matriz chamada **dp** contendo os rótulos das células que deverão preencher o **TileList** e em seguida o seu respectivo conteúdo através da propriedade **source**.

Nas linhas seguintes:

```
meuTileList.dataProvider = new DataProvider(dp);
meuTileList.columnWidth = 110;
meuTileList.rowHeight = 100;
meuTileList.setSize(280,130);
meuTileList.move(150, 150);
meuTileList.setStyle("contentPadding", 5);
meuTileList.sortItemsOn("label");
```

Criamos uma nova instância da classe **DataProvider** para preenchimento do **TileList** com o conteúdo da matriz **dp** através da propriedade **dataProvider**, e em seguida definimos:

- A largura de cada coluna do **TileList** (**columnWidth**).
- A altura de cada linha (**rowHeight**).
- O tamanho do **TileList** (**setSize**).
- A posição do **TileList** no palco (**move**).
- A distância entre a borda da instância do **TileList** e o seu conteúdo (**contentPadding**).
- A classificação em ordem alfabética do conteúdo do **TileList** considerando o rótulo de cada célula (**sortItemsOn**).

E, finalmente na linha:

```
addChild(meuTileList);
```

Utilizamos o método **addChild()** para inserir o componente **TileList** no palco.

5. Execute a aplicação e confira o resultado com o mostrado na **Figura 17.8** abaixo:

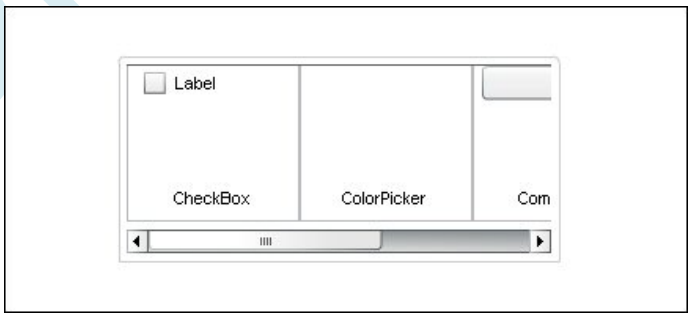

Figura 17.8 – Componente TileList preenchido com outros componentes.

*****

Exemplo 5

Nesse exemplo utilizaremos um componente **TileList** em tempo de execução para armazenar três imagens provenientes de um certo diretório de um servidor remoto (qualquer um de sua preferência). Para isso será necessário colocar essas imagens no devido lugar no servidor. Vejamos como fazer isso:

6. Crie um novo documento (ActionScript 3.0);
7. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **TileList** para a biblioteca;
8. Nomeie a camada atual para: **ações**;
9. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
import fl.controls.TileList;
import fl.controls.ScrollBarDirection;
//
var meuTileList:TileList = new TileList();
//
meuTileList.addItem({label:"FERRARI HAMANN 360", source:
"http://www.fifacupsoftheworld.com/imagens/carro1.png"});
//
meuTileList.addItem({label:"ASTON MARTIN VANTAGE", source:
"http://www.fifacupsoftheworld.com/imagens/carro2.png"});
//
meuTileList.addItem({label:"FERRARI F50", source:
"http://www.fifacupsoftheworld.com/imagens/carro3.png"});
//
meuTileList.direction = ScrollBarDirection.VERTICAL;
meuTileList.columnWidth = 200;
meuTileList.rowHeight = 134;
meuTileList.columnCount = 1;
meuTileList.rowCount = 2;
meuTileList.move(10, 10);
addChild(meuTileList);
```

Vejamos os comentários para esse código:

Nas linhas:

```
import fl.controls.TileList;
import fl.controls.ScrollBarDirection;
```

Importamos as classes necessárias para que possamos utilizar os recursos do componente adequadamente.

Na linha seguinte:

```
var meuTileList:TileList = new TileList();
```

Criamos uma instância do componente **TileList** utilizando o seu respectivo construtor.

Nas linhas abaixo:

```
meuTileList.addItem({label:"FERRARI HAMANN 360", source:
"http://www.fifacupsoftheworld.com/imagens/carro1.png"});
//
meuTileList.addItem({label:"ASTON MARTIN VANTAGE", source:
"http://www.fifacupsoftheworld.com/imagens/carro2.png"});
//
meuTileList.addItem({label:"FERRARI F50", source:
"http://www.fifacupsoftheworld.com/imagens/carro3.png"});
```

Adicionamos três imagens provenientes de um servidor remoto e seus respectivos rótulos ao componente **TileList** (**meuTileList**) através da propriedade **addItem()**.

No bloco de código seguinte:

```
meuTileList.direction = ScrollBarDirection.VERTICAL;
meuTileList.columnWidth = 200;
meuTileList.rowHeight = 134;
meuTileList.columnCount = 1;
meuTileList.rowCount = 2;
meuTileList.move(10, 10);
```

Definimos os seguintes parâmetros para a instância do componente **TileList**:

- A direção da barra de rolagem (**direction**).
- A largura da coluna (**columnWidth**).
- A altura da linha (**rowHeight**).
- O número de colunas (**columnCount**).
- O número de linhas (**rowCount**).
- A sua posição no palco (**move**).

E, finalmente na linha:

```
addChild(meuTileList);
```

Utilizamos o método **addChild()** para inserirmos o componente **TileList** no palco.

10. Execute a aplicação e confira o resultado com o mostrado na **Figura 17.9** abaixo:

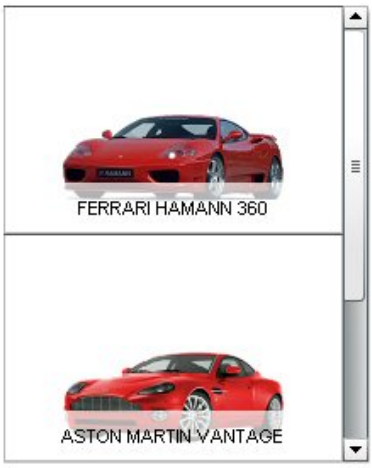

Figura 17.9 – Componente TileList preenchido com imagens provenientes de um servidor remoto.

*****

Exercícios de Fixação

1. Qual das opções abaixo devemos utilizar para criarmos uma instância de um componente **TileList** em tempo de execução. Suponha a instância de nome: **meuTL**?

   a) var meuTL-TileList = new TileList();
   b) var meuTL:TileList = new.TileList();
   c) var meuTL.TileList = newTileList();
   d) var meuTL:TileList = new TileList();
   e) var meuTL:TileList = newTileList();

2. Que propriedade devemos utilizar para determinar o número de colunas que deverão aparecer parcialmente em um componente **TileList**?

a) columnWidth
b) columnView
c) columnCount
d) columnHeight
e) columnPolicy

3. Qual o valor padrão da propriedade **verticalLineScrollSize** de um componente **TileList**?

a) 1
b) 2
c) 3
d) 4
e) 5

4. Qual a propriedade utilizada para definir a altura de cada linha da lista de um componente **TileList**?

a) rowCount
b) rowHeight
c) rowLarge
d) rowVerticalHeight
e) rowWidth

5. Qual a tecla ou conjunto de teclas que devemos utilizar para selecionar a primeira célula de um componente **TileList**?

a) Shift + PageUp
b) Alt + Home
c) End
d) PageUp
e) Home

6. Quais os parâmetros disponíveis na propriedade **scrollPolicy** para estabelecer o método de rolamento de um componente **TileList**?

a) auto, true e false
b) on, true e false
c) auto, true e off
d) auto, on e off
e) automatic, on e false

7. Que parâmetro do **Inspetor de Propriedades** devemos utilizar para preenchermos a lista de um componente **TileList** durante o projeto?

a) labelFields
b) dataFields
c) dataSource
d) dataProvider
e) sourceProvider

8. Qual o valor padrão do parâmetro **columnWidth** de um componente **TileList**?

a) 10
b) 20
c) 30
d) 40
e) 50

9. Qual a propriedade que devemos utilizar para definirmos a quantidade de colunas que deverão ser roladas horizontalmente em um componente **TileList**?

   a) horizontalScrollLineSize
   b) horizontalLineScroll
   c) horizontalLineScrollSize
   d) horizontalSizeLine
   e) horizontalLineSizeScroll

10. Qual o valor padrão do parâmetro **rowHeight** de um componente **TileList**?

   a) 30
   b) 40
   c) 50
   d) 60
   e) 70

*****

Exercícios Propostos

1. Crie uma aplicação utilizando um componente **TileList** em tempo de execução, com duas linhas e duas colunas, de forma que carregue três imagens provenientes de um servidor remoto (qualquer um de sua preferência) quando a mesma for executada. Veja uma sugestão de layout para esse exercício na figura abaixo:

2. Crie uma aplicação utilizando um componente **TileList** em tempo de execução, com duas linhas e duas colunas, de forma que carregue três imagens provenientes de um servidor remoto (qualquer um de sua preferência) quando a mesma for executada. Só que dessa vez as imagens deverão ser carregadas a partir de um script em XML.

3. Crie uma aplicação utilizando um componente **TileList** em tempo de execução, com uma linha e duas colunas, de forma que carregue três imagens provenientes de um servidor remoto (qualquer um de sua preferência) quando a mesma for executada. Veja uma sugestão de layout para esse exercício na figura abaixo:

*****