SIMULADOS & TUTORIAIS

TUTORIAIS FLASH

O Componente ProgressBar

O Componente ProgressBar

Introdução

O componente **ProgressBar** mostra o progresso de um conteúdo qualquer sendo carregado. Ele é utilizado para mostrar o estado de carga de uma animação ou imagens que deverão ser carregadas em uma aplicação. O processo de carga pode ser determinado ou indeterminado. Uma barra de progresso ***determinate*** é uma representação linear do progresso de uma tarefa, e é usado quando o total do conteúdo a ser carregado é conhecido. Uma barra de progresso ***indeterminate*** é usada quando o total do conteúdo a ser carregado é desconhecido. Você poderá adicionar um componente **Label** para mostrar o progresso de carga do conteúdo que está sendo carregado em forma de percentuais, bytes, etc. Veja na **Figura 11.1** abaixo a aparência de um componente **ProgressBar** quando uma instância é arrastada para o palco.

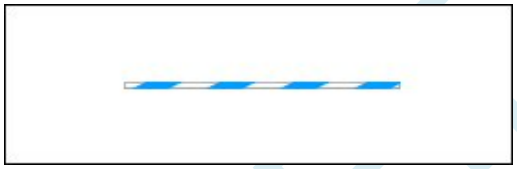


Figura 11.1 – Componente ProgressBar em sua forma original quando arrastado para o palco.

Interação do usuário com o componente ProgressBar

Há três maneiras de se utilizar o componente **ProgressBar**. As formas mais utilizadas são o modo **event** e o modo **polled**. Esses modos especificam o processo de carga que disparam os eventos **progress** e **complete** (modos **event** e **polled**), ou expressam as propriedades **bytesLoaded** e **bytesTotal** (somente modo **polled**). Você também poderá usar o componente **ProgressBar** no modo **manual** configurando as propriedades **maximum**, **minimum** e **value** juntamente com chamadas ao método **ProgressBar.setProgress()**. Você poderá configurar a propriedade **indeterminate** para indicar se o **ProgressBar** deverá apresentar uma barra com faixas diagonais alternadas e uma fonte de tamanho desconhecido (**true**), ou uma barra com um preenchimento sólido e uma fonte de tamanho conhecido (**false**).
O modo de um componente **ProgressBar** poderá ser configurado selecionando a propriedade **mode** no painiel **Component Inspector**, ou ainda usando o ActionScript em tempo de execução.

Os parâmetros do componente ProgressBar

Para utilizar corretamente esse componente é necessário conhecer alguns dos seus parâmetros, que podem ser definidos diretamente no painel **Component Inspector**, conforme mostramos a seguir:

Parâmetro	Descrição
direction	Indica a direção do preenchimento da barra de progresso. Esse valor pode ser **right** ou **left**. O valor padrão é **right**.
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**.
mode	Indica o método a ser utilizado para atualizar a barra de progresso. Esse parâmetro poderá ter um dos seguintes valores: **event**, **polled**

	ou **manual**. O valor padrão é **event**.
source	É uma referência ao conteúdo que deverá ser carregado, e pela qual o **ProgressBar** deverá calcular o progresso dessa operação de carga.
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, que podem ser utilizadas juntamente com outras propriedades, métodos e eventos disponíveis para esse componente permitindo-lhe criar aplicações mais interativas, conforme descrevemos abaixo:

As propriedades do componente ProgressBar

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

Propriedade	Descrição
indeterminate	É um valor booleano que indica o tipo de preenchimento que a barra de progresso deverá usar, e se o conteúdo a ser carregado é conhecido ou desconhecido. O valor padrão é: **true**.
maximum	É um número que informa o valor máximo para a barra de progresso quando a propriedade **ProgressBar**.**mode** estiver configurada para **ProgressBarMode**.**MANUAL**. O valor padrão é: **0**.
minimum	É um número que informa o valor mínimo para a barra de progresso quando a propriedade **ProgressBar**.**mode** estiver configurada para **ProgressBarMode**.**MANUAL**. O valor padrão é: **0**.
percentComplete	Um número entre 0 e 100 que indica o percentual do conteúdo já carregado.
value	É um valor numérico que indica o progresso já realizado em uma operação de carga. Esse valor deve estar entre o valor **minimum** e o valor **maximum**. O valor padrão é: **0**.

Os métodos do componente ProgressBar

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

Método	Descrição
ProgressBar	Cria uma nova instância do componente **ProgressBar**.
getStyleDefinition	Recupera o estilo de mapa padrão para o componente atual.
reset	Cancela o progresso de uma carga para uma nova operação.
setProgress	Configura o estado da barra de progresso para refletir a quantidade de progresso da carga quando for utilizado o modo manual. Esse método possui dois parâmetros: **value** e **maximum**. Ambos são valores numéricos.

Os eventos do componente ProgressBar

Eventos são ações utilizadas pelos componentes para executar tarefas específicas, de acordo com a finalidade de cada situação. Veja a seguir os eventos disponíveis para esse componente e para que servem.

Evento	Descrição
complete	É executado quando a operação de carga for completada.
progress	É executado de acordo com o modo de carga, **event** ou **polled**.

Veja como utilizar esses recursos nos exemplos a seguir, como também no DVD, que acompanha a presente obra.

Criando aplicações com o componente ProgressBar

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

Exemplo 1

Nesse exemplo mostraremos como usar o componente **ProgressBar** para carregar uma imagem local. Para isso, precisaremos de uma imagem qualquer no formato **JPG**, que deverá ser salva no mesmo diretório da aplicação, como também de um componente **UILoader** para armazenar essa imagem e dois componentes **Label** para mostrar o progresso da carga e a carga total. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **ProgressBar** para o palco;
3. No painel **Properties**, crie a seguinte instância para ele: **barra**.
4. Abra o painel **Component Inspector** (**Shift + F7**), informe no parâmetro **source** o nome da instância do destino da imagem: **carregador**. No parâmetro **mode** mantenha o modo **event**;
5. Arraste também um componente **UILoader** para armazenar a imagem. A imagem a ser carregada para esse componente terá as dimensões que você determinar para o componente, mas, obedecendo o formato da imagem original;
6. No painel **Properties**, crie a seguinte instância para ele: **carregador**, e altere as suas dimensões para: **150** x **150** pixels;
7. Agora arraste dois componentes **Label** para o palco, e altere os seguintes parâmetros:

   Label 1:
   Instance Name: progresso
   W (largura): 180
   autoSize: left

   Label 2:
   Instance Name: mensagem
   W (largura): 180
   autoSize: left

8. Nomeie a camada atual para: **componentes**;
9. Crie uma nova camada e chame-a de: **ações**;
10. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
carregador.source = "diego2.jpg";
//
barra.addEventListener(ProgressEvent.PROGRESS, progressoCarga);
//
function progressoCarga (event:ProgressEvent):void {
    progresso.text = event.bytesLoaded + " de " + event.bytesTotal + " bytes carrega-
dos.";
}
//
barra.addEventListener(Event.COMPLETE, cargaCompleta);
//
function cargaCompleta(event:Event):void {
    var nrBytes= carregador.bytesTotal;
    mensagem.text="Carga completada: ("+nrBytes+" bytes).";
}
```

Vejamos os comentários sobre o código:

Na linha:

```
carregador.source = "diego2.jpg";
```

Inicialmente, definimos a imagem a ser carregada automaticamente no componente **UILoader** quando a aplicação for executada, utilizando para isso a propriedade **source**.

No bloco de código a seguir:

```
barra.addEventListener(ProgressEvent.PROGRESS, progressoCarga);
function progressoCarga (event:ProgressEvent):void {
    progresso.text = event.bytesLoaded + " de " + event.bytesTotal + " bytes carregados.";
}
```

Utilizamos o evento **PROGRESS** vinculado à instância (**barra**) do componente **ProgressBar**, de forma que, quando o mesmo for executado, a função **progressoCarga** seja também executada. Em seguida criamos a função **progressoCarga** que por sua vez mostrará na instância (**progresso**) do componente **Label** uma mensagem sobre o progresso da carga dessa imagem. Para isso utilizamos as propriedades **bytesLoaded** e **bytesTotal** do componente **UILoader**.

E finalmente no bloco de código:

```
barra.addEventListener(Event.COMPLETE, cargaCompleta);
function cargaCompleta(event:Event):void {
    var nrBytes=carregador.bytesTotal;
    mensagem.text="Carga completada: ("+nrBytes+" bytes).";
}
```

Utilizamos o evento **COMPLETE** vinculado à instância (**barra**) do componente **ProgressBar**, de forma que, quando o mesmo for executado, a função **cargaCompleta** seja também executada. Em seguida criamos a função **cargaCompleta** que por sua vez armazenará na variável **nrBytes** a quantidade total de bytes da imagem carregada, informação essa proveniente do componente **UILoader** (instância: **carregador**). Depois disso, será mostrada na instância (**mensagem**) do componente **Label** uma mensagem informando que a carga foi completada e a respectiva quantidade total de bytes da imagem carregada.

11. Execute a aplicação e confira o resultado com o mostrado na **Figura 11.2** a seguir:

Figura 11.2 – Componente ProgressBar utilizado conjuntamente com um componente UILoader para acompanhar o progresso da carga de uma imagem.

Você deve ter notado que quase, ou não se percebe a barra de progresso em funcionamento. Isso se deve ao fato de que a imagem desse exemplo é muito pequena e a velocidade de processamento do computador é muito grande, e, além disso, a imagem é local, por isso, é quase imperceptível notar esse detalhe.
Para perceber o processo da barra de progresso, experimente carregar uma imagem bem maior ou então a partir de um servidor remoto.

*****

Exemplo 2

Nesse exemplo utilizaremos um componente **ProgressBar** e um componente **Label** em tempo de execução para mostrar o progresso de uma barra para realizar uma determinada tarefa. À medida que a barra estiver sendo preenchida, será mostrado no **Label** o percentual de carga de um conteúdo qualquer que você poderá utilizar. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **ProgressBar** e um componente **Label** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
import fl.controls.Label;
import fl.controls.ProgressBar;
import fl.controls.ProgressBarMode;
//
var barraProgresso:ProgressBar = new ProgressBar();
barraProgresso.indeterminate = false;
barraProgresso.mode = ProgressBarMode.MANUAL;
barraProgresso.maximum = 100;
barraProgresso.setSize(320, 16);
barraProgresso.move(50, 50);
addChild(barraProgresso);
//
var meuRotulo:Label = new Label();
meuRotulo.text = "";
meuRotulo.autoSize = TextFieldAutoSize.LEFT;
meuRotulo.move(barraProgresso.x, barraProgresso.y + barraProgresso.height);
addChild(meuRotulo);
//
var tempo:Timer = new Timer(100);
tempo.addEventListener(TimerEvent.TIMER, tempoProgresso);
tempo.start();
//
function tempoProgresso(event:TimerEvent):void {
    barraProgresso.setProgress(barraProgresso.value + 1, barraProgresso.maximum);
    if (barraProgresso.percentComplete == barraProgresso.maximum) {
        tempo.stop();
    }
    meuRotulo.text = int(barraProgresso.value) + " de " +
int(barraProgresso.maximum) + " (" + int(barraProgresso.percentComplete) + "%)";
}
```

Vejamos os comentários sobre o código:

Nas linhas:

```
import fl.controls.Label;
import fl.controls.ProgressBar;
import fl.controls.ProgressBarMode;
```

Importamos as classes dos respectivos componentes para que possamos utilizá-los adequadamente em nossa aplicação.

No bloco de código a seguir:

```
var barraProgresso:ProgressBar = new ProgressBar();
barraProgresso.indeterminate = false;
```

```
barraProgresso.mode = ProgressBarMode.MANUAL;
barraProgresso.maximum = 100;
barraProgresso.setSize(320, 16);
barraProgresso.move(50, 50);
```

Criamos uma nova instância do componente **ProgressBar** utilizando o seu respectivo construtor, chamada **barraProgresso**, e em seguida definimos os seguintes parâmetros para ela:

- **indeterminate** – definimos para **false** para indicar que a barra de progresso possui um preenchimento sólido e o tamanho do conteúdo a ser carregado é conhecido.
- **mode** – selecionamos o modo manual para que possamos definir os valores **minimum** e **maximum** do conteúdo a ser carregado, como também poder chamar o método **setProgress()**.
- **maximum** – definimos o valor máximo para **100**.
- **setSize** – definimos o tamanho da barra de progresso (largura e altura).
- **move** – definimos a localização da barra no palco.

Na linha a seguir:

```
addChild(barraProgresso);
```

Inserimos o componente no palco através do método **addChild()**.

No bloco de código seguinte:

```
var meuRotulo:Label = new Label();
meuRotulo.text = "";
meuRotulo.autoSize = TextFieldAutoSize.LEFT;
meuRotulo.move(barraProgresso.x, barraProgresso.y + barraProgresso.height);
```

Criamos uma nova instância do componente **Label** chamada **meuRotulo**, utilizando o seu respectivo construtor, e em seguida definimos os seguintes parâmetros para ela:

- **text** – definimos o rótulo do **Label** como uma string vazia.
- **autoSize** – definimos o alinhamento do conteúdo do componente para **LEFT** (esquerda).
- **move** – definimos a sua posição no palco de forma que a mesma fique com as mesmas coordenadas da barra de progresso juntamente com sua altura.

Na linha a seguir:

```
addChild(meuRotulo);
```

Inserimos a instância do componente **Label** no palco através do método **addChild()**.

No bloco de código a seguir:

```
var tempo:Timer = new Timer(100);
tempo.addEventListener(TimerEvent.TIMER, tempoProgresso);
tempo.start();
```

Criamos uma nova instância da classe **TIMER** chamada **tempo**, através do seu respectivo construtor, para definirmos o tempo (100 milisegundos) para completar o tempo de carga do conteúdo. Em seguida, criamos um evento **TIMER**, vinculado à instância **tempo**, de forma que quando o mesmo for disparado, a função **tempoProgresso** seja executada. Logo após iniciamos a contagem do tempo através da função **start()**.

No bloco de código:

```
function tempoProgresso(event:TimerEvent):void {
    barraProgresso.setProgress(barraProgresso.value + 1, barraProgresso.maximum);
    if (barraProgresso.percentComplete == barraProgresso.maximum) {
            tempo.stop();
    }
    meuRotulo.text = int(barraProgresso.value) + " de " +
int(barraProgresso.maximum) + " (" + int(barraProgresso.percentComplete) + "%)";
}
```

Criamos uma função chamada **tempoProgresso** vinculada ao evento **TIMER**, de forma que quando a mesma for executada, a barra de progresso (**barraProgresso**) comece a ser preenchida acrescida de uma unidade a cada 100 milisegundos. Em seguida, criamos uma condição para que a execução seja interrompida quando o valor máximo (100) estabelecido para a barra for atingido. Também mostramos no componente **Label** o andamento da operação de carga, em valores e em percentuais.

5. Execute a aplicação e confira o resultado com o mostrado na **Figura 11.3** a seguir:

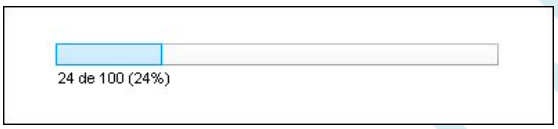


Figura 11.3 – Componente ProgressBar processando a carga de um conteúdo qualquer.

*****

Exemplo 3

Nesse exemplo utilizaremos um componente **ProgressBar**, um componente **Label** e um componente **NumericStepper** que poderá ser utilizado para carregar um conteúdo qualquer, de forma que, à medida que o valor do **NumericStepper** for sendo alterado a barra de progresso acompanhará esse valor percentualmente, e simultaneamente uma mensagem desse percentual será mostrada no componente **Label**. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **Label** , um componente **ProgressBar** e um componente **NumericStepper** para o palco;
3. Nos painéis **Properties** e no **Component Inspector**, altere os seguintes parâmetros para eles:

Label:
Instance Name: meuRotulo
text = “”

ProgressBar:
Instance Name: barraProgresso

NumericStepper:
Instance Name: nStepper
maximum = 100
value = 0

4. Nomeie a camada atual para: **componentes**;
5. Crie uma nova camada e chame-a de: **ações**;
6. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
import fl.controls.ProgressBarDirection;
import fl.controls.ProgressBarMode;
```

```
import flash.events.Event;
//
meuRotulo.text = "Percentual do progresso = 0";
barraProgresso.direction = ProgressBarDirection.RIGHT;
barraProgresso.mode = ProgressBarMode.MANUAL;
barraProgresso.minimum = nStepper.minimum;
barraProgresso.maximum = nStepper.maximum;
barraProgresso.indeterminate = false;
//
nStepper.addEventListener(Event.CHANGE, alteraPercentual);
//
function alteraPercentual(event:Event):void {
    barraProgresso.value = nStepper.value;
    barraProgresso.setProgress(barraProgresso.value, barraProgresso.maximum);
    meuRotulo.text = "Percentual do progresso = " + nStepper.value + "%";
}
```

Vejamos os comentários sobre o código:

Nas linhas:

```
import fl.controls.ProgressBarDirection;
import fl.controls.ProgressBarMode;
import flash.events.Event;
```

Importamos as classes necessárias para que possamos utilizar os componentes adequadamente em nossa aplicação, conforme descritas abaixo:

- A primeira classe nos permite definir a direção da barra de progresso.
- A segunda classe nos permite definir o modo que vamos utilizar.
- A terceira classe nos permite utilizar os respectivos eventos disponíveis para o componente.

Na linha:

```
meuRotulo.text = "Percentual do progresso = 0";
```

Criamos um rótulo para a instância **meuRotulo** do componente **Label** utilizando para isso a propriedade **text**.

No bloco de código:

```
barraProgresso.direction = ProgressBarDirection.RIGHT;
barraProgresso.mode = ProgressBarMode.MANUAL;
barraProgresso.minimum = nStepper.minimum;
barraProgresso.maximum = nStepper.maximum;
barraProgresso.indeterminate = false;
```

Definimos alguns parâmetros para a instância **barraProgresso** conforme segue:

- **direction** – decidimos através da propriedade **RIGHT** que o preenchimento da barra de progresso será a partir da esquerda para a direita.
- **mode** – selecionamos o modo manual para que possamos definir os valores **minimum** e **maximum** do conteúdo a ser carregado, como também poder utilizar o método **setProgress()**.
- **minimum** – decidimos que o valor da propriedade **minimum** da barra deverá ser o mesmo do componente **NumericStepper**, ou seja, 0.
- **maximum** – decidimos que o valor da propriedade **maximum** da barra deverá ser o mesmo do componente **NumericStepper**, ou seja, 100.
- **indeterminate** – definimos para **false** para indicar que a barra de progresso possui um preenchimento sólido e o tamanho do conteúdo a ser carregado é conhecido.

Na linha seguinte:

```
nStepper.addEventListener(Event.CHANGE, alteraPercentual);
```

Utilizamos o evento **CHANGE** vinculado à instância (**nStepper**) do componente **NumericStepper**, de forma que quando o mesmo for executado, a função **alteraPercentual** seja também executada.

No bloco de código a seguir:

```
function alteraPercentual(event:Event):void {
    barraProgresso.value = nStepper.value;
    barraProgresso.setProgress(barraProgresso.value, barraProgresso.maximum);
    meuRotulo.text = "Percentual do progresso = " + nStepper.value + "%";
}
```

Criamos a função **alteraPercentual** vinculada ao evento **CHANGE** do componente **NumericStepper**, de forma que, a cada vez que a mesma for executada, ou seja, o valor do **NumericStepper** for alterado, a barra de progresso seja preenchida com o mesmo valor selecionado no **NumericStepper**, e ao mesmo tempo o andamento dessa operação seja mostrado no label **meuRotulo**.

6. Execute a aplicação e confira o resultado com o mostrado na **Figura 11.4** abaixo a seguir:

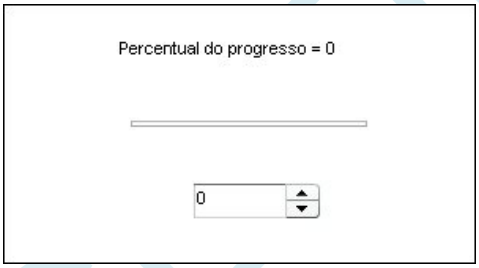


Figura 11.4 – Componente ProgressBar utilizado conjuntamente com um componente NumericStepper e um Label para mostrar o progresso da carga de um conteúdo qualquer, utilizando a forma manual.

Experimente digitar um valor qualquer diretamente no componente **NumericStepper** ou utilizar as teclas quando o mesmo estiver com o foco. Procure utilizar esse exemplo em uma atividade prática.

*****

Exemplo 4

Nesse exemplo utilizaremos um componente **ProgressBar**, e um componente **ScrollPane** em tempo de execução, de forma que, uma imagem deverá ser carregada de um servidor remoto (qualquer um de sua preferência), mas antes da imagem ser carregada completamente a barra de progresso mostrará o percentual já carregado. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **ProgressBar** e um componente **ScrollPane** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
```

```
import fl.containers.ScrollPane;
import fl.controls.ProgressBar;
//
var meuScrollPane:ScrollPane = new ScrollPane();
meuScrollPane.setSize(200, 160);
meuScrollPane.move(100, 100);
meuScrollPane.source = "http://www.fifacupsoftheworld.com/imagens/diego4.jpg";
addChild(meuScrollPane);
//
var barraProgresso:ProgressBar = new ProgressBar();
barraProgresso.width = meuScrollPane.width;
barraProgresso.move(meuScrollPane.x, meuScrollPane.y - barraProgresso.height-10);
barraProgresso.source = meuScrollPane;
addChild(barraProgresso);
```

Vejamos os comentários sobre o código:

Nas linhas:

```
import fl.containers.ScrollPane;
import fl.controls.ProgressBar;
```

Importamos as classes dos respectivos componentes para que possamos utilizá-los adequadamente em nossa aplicação.

No bloco de código a seguir:

```
var meuScrollPane:ScrollPane = new ScrollPane();
meuScrollPane.setSize(200, 160);
meuScrollPane.move(100, 100);
meuScrollPane.source = "http://www.fifacupsoftheworld.com/imagens/diego4.jpg";
```

Criamos uma nova instância do componente **ScrollPane** através do seu respectivo construtor, chamada **meuScrollPane**. Em seguida definimos os seguintes parâmetros para ela:

- **setSize** – definimos o seu tamanho (largura e altura).
- **move** – a sua posição no palco (coordenadas X e Y).
- **source** – a imagem que deverá ser carregada, que neste caso será proveniente de um servidor remoto.

Na linha seguinte:

```
addChild(meuScrollPane);
```

Inserimos a instância do componente **ScrollPane** no palco através do método **addChild()**.

No bloco de código:

```
var barraProgresso:ProgressBar = new ProgressBar();
barraProgresso.width = meuScrollPane.width;
barraProgresso.move(meuScrollPane.x, meuScrollPane.y - barraProgresso.height-10);
barraProgresso.source = meuScrollPane;
```

Criamos uma nova instância do componente **ProgressBar** chamada **barraProgresso** através do seu construtor. Em seguida, definimos os seguintes parâmetros para ela:

- **width** – definimos a sua largura para a mesma largura do **ScrollPane**.
- **move** – definimos que a sua posição no palco deverá ser a mesma do **ScrollPane** menos a altura do próprio componente **ProgressBar** deduzidos 10 pixels.

- **source** – decidimos que a fonte de origem da imagem seria o componente **ScrollPane**.

Na linha seguinte:

addChild(barraProgresso);

Inserimos a instância do componente **ProgressBar** no palco através do método **addChild()**.

7. Execute a aplicação e confira o resultado com o mostrado na **Figura 11.5** abaixo a seguir:

Figura 11.5 – Componente ProgressBar utilizado conjuntamente com um componente ScrollPane para carregar uma imagem de um servidor remoto.

*****

Exemplo 5

Nesse exemplo utilizaremos um componente **ProgressBar**, um componente **ScrollPane** e um componente **Button** em tempo de execução de forma que uma imagem deverá ser carregada somente quando o botão for clicado, e logo em seguida o botão deverá ficar desabilitado. A imagem deverá ser carregada a partir de um servidor remoto (qualquer um de sua preferência). Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **ScrollPane** , um componente **ProgressBar** e um componente **Button** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
import fl.containers.ScrollPane;
import fl.controls.ProgressBar;
import fl.controls.Button;
import flash.events.MouseEvent;
//
var meuBotao:Button = new Button();
meuBotao.setSize(120, 30);
meuBotao.label="Carregar Imagem";
meuBotao.move(110, 240);
addChild(meuBotao);
//
var meuScrollPane:ScrollPane = new ScrollPane();
meuScrollPane.setSize(320, 200);
meuScrollPane.move(10, 30);
var minhaURL="http://www.fifacupsoftheworld.com/imagens/carol1.jpg";
addChild(meuScrollPane);
```

```
//
var barraProgresso:ProgressBar = new ProgressBar();
barraProgresso.width = meuScrollPane.width;
barraProgresso.move(meuScrollPane.x, meuScrollPane.y - barraProgresso.height-10);
barraProgresso.source = meuScrollPane;
addChild(barraProgresso);
//
meuBotao.addEventListener(MouseEvent.CLICK,carregaImagem);
function carregaImagem(event:MouseEvent):void {
    meuScrollPane.source = minhaURL;
    meuBotao.enabled=false;
}
```

Vejamos os comentários sobre o código:

Nas linhas:

```
import fl.containers.ScrollPane;
import fl.controls.ProgressBar;
import fl.controls.Button;
import flash.events.MouseEvent;
```

Importamos as classes dos respectivos componentes para que possamos utilizá-los adequadamente em tempo de execução em nossa aplicação.

No bloco de código a seguir:

```
var meuBotao:Button = new Button();
meuBotao.setSize(120, 30);
meuBotao.label="Carregar Imagem";
meuBotao.move(110, 240);
addChild(meuBotao);
```

Criamos uma nova instância do componente **Button**, chamada **meuBotao**, através do seu respectivo construtor. Logo em seguida definimos o seu tamanho (**setSize**), o seu rótulo (**label**) e a sua posição no palco (**move**). Por fim, inserimos o componente no palco através do método **addChild()**.

No bloco de código:

```
var meuScrollPane:ScrollPane = new ScrollPane();
meuScrollPane.setSize(320, 200);
meuScrollPane.move(10, 30);
var minhaURL="http://www.fifacupsoftheworld.com/imagens/carol1.jpg";
addChild(meuScrollPane);
```

Criamos uma nova instância do componente **ScrollPane**, chamada **meuScrollPane**, através do seu respectivo construtor. Logo em seguida definimos o seu tamanho (**setSize**), a sua posição no palco (**move**) e criamos uma variável (**minhaURL**) para armazenar o endereço de um servidor remoto onde se encontra a imagem que deverá ser carregada. Por fim, inserimos o componente no palco através do método **addChild()**.

No código abaixo:

```
var barraProgresso:ProgressBar = new ProgressBar();
barraProgresso.width = meuScrollPane.width;
barraProgresso.move(meuScrollPane.x, meuScrollPane.y - barraProgresso.height-10);
barraProgresso.source = meuScrollPane;
addChild(barraProgresso);
```

Criamos uma nova instância do componente **ProgressBar**, chamada **barraProgresso**, através do seu respectivo construtor. Logo em seguida definimos a sua largura (**width**)

de forma que seja igual à largura do componente **ScrollPane**, a sua posição no palco (**move**) que deverá ser igual às coordenadas do componente **ScrollPane** deduzidas da altura da própria barra de progresso menos 10 pixels. Depois decidimos que o recipiente que deverá receber a imagem será o **ScrollPane**. Por fim, inserimos o componente no palco através do método **addChild()**.

No bloco de código seguinte:

```
meuBotao.addEventListener(MouseEvent.CLICK,carregaImagem);
function carregaImagem(event:MouseEvent):void {
    meuScrollPane.source = minhaURL;
    meuBotao.enabled=false;
}
```

Utilizamos o evento **CLICK** vinculado à instância (**meuBotao**) do componente **Button**, de forma que quando o mesmo for executado, a função **carregaImagem** seja também executada. Em seguida criamos a função **carregaImagem** que por sua vez carregará a imagem que se encontra no endereço **minhaURL** e ao mesmo tempo o botão ficará desabilitado, evitando a recarga da mesma imagem.

8. Execute a aplicação e confira o resultado com o mostrado na **Figura 11.6** abaixo a seguir:

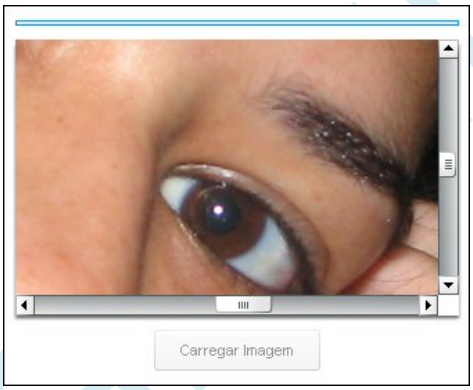


Figura 11.6 – Componente ProgressBar utilizado conjuntamente com um componente ScrollPane e um componente Button para carregar uma imagem e mostrar o seu progresso de carga.

*****

Exercícios de Fixação

1. Quais são as opções disponíveis para o parâmetro **mode** de um componente **ProgressBar**?

   a) manual, auto ou event.
   b) event, auto ou polled.
   c) auto, complete ou manual.
   d) event, polled ou manual.
   e) event, polled ou complete.

2. Quais as opções disponíveis para o parâmetro **direction** de um componente **ProgressBar**?

   a) left ou bottom
   b) true ou false
   c) top ou right
   d) top ou bottom

e) right ou left

3. Quais os valores padrões das propriedades **maximum** e **minimum** respectivamente de um componente **ProgressBar**?

   a) 0 e 0
   b) 0 e 10
   c) 0 e 100
   d) 1 e 10
   e) 1 e 100

4. Que propriedade de um componente **ProgressBar** devemos utilizar para determinar se o conteúdo a ser carregado é conhecido ou não?

   a) determinate
   b) indeterminate
   c) determination
   d) known
   e) unknown

5. Que classe devemos importar para criarmos uma instância de um componente **ProgressBar** em tempo de execução?

   a) flash.control.ProgressBar
   b) fl.control.ProgressBar()
   c) flash.controlsProgressBar
   d) fl.control-ProgressBar
   e) fl.controls.ProgressBar

6. Qual das opções abaixo devemos utilizar para criarmos uma instância de um componente **ProgressBar** de nome **barra**?

   a) var barra:ProgressBar = new ProgressBar(barra);
   b) var barra.ProgressBar = new.ProgressBar;
   c) var barra:ProgressBar = newProgressBar;
   d) var barra:ProgressBar = new ProgressBar;
   e) var barra.ProgressBar = new-ProgressBar;

7. Que evento de um componente **ProgressBar** é utilizado quando o conteúdo é totalmente carregado?

   a) totalBytes
   b) totally
   c) complete
   d) completed
   e) totalComplete

8. Que método é utilizado para cancelar o processo de uma carga de um componente **ProgressBar**?

   a) cancel
   b) cancelled
   c) reset
   d) stop
   e) finish

9. Qual o valor padrão da propriedade **value** de um componente **ProgressBar**?

   a) 0

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

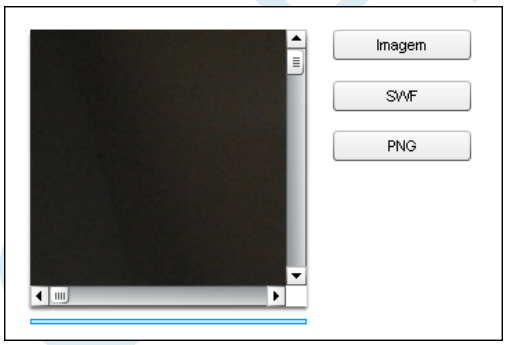
10. Que parâmetro devemos utilizar para vincular um componente **ProgressBar** a um componente **UILoader**?

a) value
b) progress
c) set
d) source
e) mode

*****

Exercícios Propostos

1. Crie uma aplicação utilizando um componente **ProgressBar**, um componente **ScrollPane** e três componentes **Button**, em tempo de execução ou não, de forma que quando cada botão for clicado, sejam carregados no componente **ScrollPane** uma imagem **JPG**, um arquivo **SWF** e uma imagem **PNG** respectivamente, e mostre no componente **ProgressBar** o processo de carga de cada operação. Veja a seguir uma sugestão para o layout desse exemplo:



2. Crie uma aplicação utilizando um componente **ProgressBar**, um componente **Label**, um componente **UILoader** e três componentes **Button**, em tempo de execução ou não, de forma que quando cada botão for clicado carregue uma imagem diferente para o mesmo **UILoader**, e mostre no **Label** o processo de carga de cada operação. As imagens deverão ter tamanhos diferentes e provenientes de um servidor remoto (qualquer um de sua preferência). Os botões após clicados deverão ficar desabilitados. Veja a seguir uma sugestão para o layout desse exemplo:

*****