Controles Compostos

A interação entre usuário e programa é feita a partir de **controles**. Menus, Grids e Dialogs são exemplos de controles. Um controle representa uma **área retangular** onde serão apresentados dados para o usuário, estes controles podem ser chamados também de **Containers**.

O Symbian oferece **APIs** para alguns controles como grids, menus, dialogs, formulários, listas, etc. Porém programadores podem criar seus próprios controles utilizando a classe CCoeControl

Exemplos de controles do Symbian OS

CAknGrid - Grids
CEikDialog - Dialogs
CEikEdwin - Editores de texto

Controles simples e compostos

Controles podem ser **simples** ou **compostos**. Controles compostos englobam **vários** controles simples, sendo o controle principal responsável pelos outros chamado de **Pai**. É responsabilidade do controle "pai" gerênciar seus filhos através de **ponteiros** ,**re-passar eventos** e definir qual área da tela eles irão utilizar. A principal vantagem em se utilizar controles compostos é que o manuseio de outros controles se dará de forma mais **fácil**. Para fazer a sua implementação é necessário sobrecarregar duas funções da classe CCoeControl, **CountComponentControl** e **ComponentControl**:

```
TInt CountComponentControl() const;
```

Essa função retornará a **quantidade** de controles que o controle Pai "gerência".

```
CCoeControl* ComponentControl(TInt aIndex) const;
```

Os controles filhos serão controlados através desta função, o argumento aIndex será utilizado para **referenciar** os ponteiros para os controles filhos.

Nota: Em geral controles pais **não implementam** a função Draw(), quando o fazem é apenas para fazer a chamada a função Clear() do contexto gráfico para não inicializar o programa com "lixo" na tela.

Os controles filhos serão responsáveis por **exibir** os dados do programa para o usuário, cada controle filho terá (ou não) um espaço na tela que irá utilizar para exibir seus dados. Este espaço nada mais é do que um **retângulo** criado pelo controle Pai e passado como referência para seus filhos durante instâncialização dos mesmos.

A função SizeChanged() deve ser implementada no controle pai definindo o retângulo que os controles irão utilizar.

Controles que possuem janela e que não possuem uma janela

Controles são responsáveis por realizar a **interação** entre usuário -> programa, porém o fazem através de **janelas**. Em geral aplicações utilizam apenas uma janela, e é responsabilidade dos controles cria-la ou utilizar uma de outro controle. Controles que criam janelas são conhecidos como **Window Owning**, aqueles que não o fazem são conhecidos como **Non Window Owning**. Como janelas utilizam **recursos**, tratando-se de controles compostos em geral **apenas** o controle pai cria a janela e seus filhos fazem uso desta através da função SetContainerWindowL() que pode receber como argumento o controle que criou a janela. Isto garante uma **econômia** de recursos.

A criação/definição de uma janela se da no ConstructL de cada controle, a criação de uma janela é feita a partir da função:

```
CreateWindowL()
```

Quando um controle não criar uma janela, deverá ser definido que ele irá utilizar a janela de outro controle, essa definição é feita a partir da função:

(C) Copyright Nokia 2014. All rights reserved.

```
SetContainerWindowL()
```

Existem diversas possibilidades de argumentos, porém a que passa o controle pai como argumento é a mais interessante em se tratar de controles compostos.

```
virtual void SetContainerWindowL(const CCoeControl& aContainer);
```

Artigos relacionados

Controles

Exemplo

Controles Compostos - Exemplo

Leitura complementar

Symbian OS - Creating Custom Controls (EN)

(C) Copyright Nokia 2014. All rights reserved.