Subject: Modified SliderCtrl
Posted by cdoty on Fri, 02 May 2008 19:35:39 GMT
View Forum Message <> Reply to Message

Here's my modified slider control:

The slider implements:
1) Min and Max values shown at the top/right of the slider.

2) Tick marks below the slider, with a settable tick count and tick size.

3) Value is displayed in a filled rect, whenever the left mouse button is held down on the slider.

Caveats:
1) The vertical code has not been tested.

2) To get it compiled with the current version of the upp source code, the IsVert() and SliderToClient() functions, in SliderCtrl, need to be made public.

3) The font is assumed to be StdFont, I could not find a function to get the current font from a control.

4) The Min and Max values are drawn in the control, because I couldn't put the numbers as static text, and ensure they could be overdrawn with the slider value.

Here's the header:

```
#pragma once

#include <CtrlLib/CtrlLib.h>

using namespace Upp;

#define TICKSTEP 1
#define VALUESPACING 2
#define VALUEOFFSET 4
#define TICKSIZE 8
#define SMALLTICKSIZE 5

struct LabelledSliderCtrl : public SliderCtrl
{
  protected:
 bool DrawValue; // Draw value?
 int TickStep; // Tick step
```

```
 int TickSize; // Size of end tick marks
 int SmallTickSize; // Size of middle tick marks
 int ValueOffset; // Spacing between border and value
 int ValueSpacing; // Distance from control to value rect

  public:
 // Constructor
 LabelledSliderCtrl();

 // Paint
 virtual void Paint(Draw &w);

 // Left down
 virtual void LeftDown(Point pos, dword keyflags);

 // Left up
 virtual void LeftUp(Point pos, dword keyflags);

 // Set tick step
 void SetTickStep(int NewStep);

 // Set tick size
 void SetTickSize(int NewSize);

 // Set small tick step
 void SetSmallTickSize(int NewSize);

 // Set value offset
 void SetValueOffset(int NewOffset);

 // Set value spacing
 void SetValueSpacing(int NewSpacing);
};
```

Here's the source file:

```
#include "LabelledSliderCtrl.h"

LabelledSliderCtrl::LabelledSliderCtrl() :
 DrawValue(false),
 TickStep(TICKSTEP),
 ValueSpacing(VALUESPACING),
 ValueOffset(VALUEOFFSET),
 TickSize(TICKSIZE),
 SmallTickSize(SMALLTICKSIZE)
{
}
```

```
void LabelledSliderCtrl::Paint(Draw &w)
{
 FontInfo Info   = StdFont().Info();
 int  ElementHeight = Info.GetHeight();
 Value  ElementData  = GetData();
 String  DataString  = ElementData.ToString();
 Size  ElementSize  = GetSize();

 SliderCtrl::Paint(w);

 // Draw min. and max. values at the top/right of the slider
 if (true == IsVert())
 {
  int x = (ElementSize.cx - CtrlImg::vthumb().GetSize().cx) / 2 - ValueOffset;

  // Draw the minimum number
  w.DrawText(x, 1, AsString(GetMin()));

  // Draw the maximum number
  w.DrawText(x, ElementSize.cy - ElementHeight - ValueOffset, AsString(GetMax()));
 }

 else
 {
  DWORD ElementWidth = 0;
  String MaxValue  = AsString(GetMax());

  // Draw the minimum number
  w.DrawText(1, ((ElementSize.cy - CtrlImg::hthumb().GetSize().cy) >> 1) - ValueOffset -
   ElementHeight, AsString(GetMin()));

  // Calculate the width of the maximum number, and offset from the right of the control
  for (int Loop = 0; Loop < MaxValue.GetCount(); Loop++)
  {
   ElementWidth += Info[MaxValue[Loop]];
  }

  // Draw the maximum number
  w.DrawText(ElementSize.cx - ElementWidth - 2,
   (ElementSize.cy - CtrlImg::hthumb().GetSize().cy) / 2 - ValueOffset - ElementHeight,
   MaxValue);
 }

 // Draw the tick marks below/left the slider
 if (true == IsVert())
 {
  // Calculate the start of the tick marks
```

```
 int x = CtrlImg::vthumb().GetSize().cx / 2 - 1;
 int y = CtrlImg::vthumb().GetSize().cy / 2 - 1;

 // Draw the first and last tick marks. They are drawn longer than the middle ones
 w.DrawLine(x, y, x, y + TickSize);

 y  = ElementSize.cy - CtrlImg::vthumb().GetSize().cy / 2 - 1;

 w.DrawLine(x, y, x + TickSize, y);

 // Draw the middle tick marks. Needs to be adjusted with a step value.
 for (int Loop = int(GetMin()) + 1; Loop < int(GetMax()); Loop += TickStep)
 {
  y = SliderToClient(Loop) + CtrlImg::vthumb().GetSize().cy / 2;

  w.DrawLine(x + (TickSize - SmallTickSize), y, x + TickSize, y);
 }
}

else
{
 // Calculate the start of the tick marks
 int x = CtrlImg::hthumb().GetSize().cx / 2 - 1;
 int y = ElementSize.cy - CtrlImg::hthumb().GetSize().cy - 1;

 // Draw the first and last tick marks. They are drawn longer than the middle ones
 w.DrawLine(x, y, x, y + TickSize);

 x  = ElementSize.cx - CtrlImg::hthumb().GetSize().cx / 2 - 1;

 w.DrawLine(x, y, x, y + TickSize);

 // Draw the middle tick marks. Needs to be adjusted with a step value.
 for (int Loop = int(GetMin()) + 1; Loop < int(GetMax()); Loop += TickStep)
 {
  x = SliderToClient(Loop) + CtrlImg::hthumb().GetSize().cx / 2;

  w.DrawLine(x, y + (TickSize - SmallTickSize), x, y + TickSize);
 }
}

// Draw the actual value above the slider, enclosed in a rect with a black border.
// The value is only drawn when the mouse button is being held down.
if (true == DrawValue && false == IsNull(GetData()))
{
 int   ElementWidth = 0;
 Rect  BoxRect;
 Rect  TextRect;
```

```
// Calculate the width of the data
for (int Loop = 0; Loop < DataString.GetCount(); Loop++)
{
 ElementWidth += Info[DataString[Loop]];
}

if (true == IsVert())
{
 TextRect.left = ((ElementSize.cx - CtrlImg::vthumb().GetSize().cx) / 2) + ElementWidth -
  ValueOffset;

 TextRect.top = SliderToClient(ElementData);
 TextRect.right = TextRect.left + ElementWidth;
 TextRect.bottom = TextRect.top + ElementHeight;
}

else
{
 TextRect.left = SliderToClient(ElementData);
 TextRect.top = ((ElementSize.cy - CtrlImg::hthumb().GetSize().cy) / 2) -
  ValueOffset - ElementHeight;

 TextRect.right = TextRect.left + ElementWidth;
 TextRect.bottom = TextRect.top + ElementHeight;
}

BoxRect = TextRect;

if (ElementData == GetMin())
{
 if (true == IsVert())
 {
  BoxRect.left = TextRect.left - ValueSpacing;
  BoxRect.right = TextRect.right + ValueSpacing;
 }

 else
 {
  BoxRect.right = TextRect.right + ValueSpacing * 2;

  TextRect.left++;
 }
}

else if (ElementData == GetMax())
{
 if (true == IsVert())
```

```
   {
    BoxRect.left = TextRect.left - ValueSpacing;
    BoxRect.right = TextRect.right + ValueSpacing;
   }

   else
   {
    BoxRect.left = TextRect.left - ValueSpacing * 2;
    BoxRect.right--;

    TextRect.left -= ValueSpacing;
   }
  }

  else
  {
   BoxRect.left = TextRect.left - ValueSpacing;
   BoxRect.right = TextRect.right + ValueSpacing;
  }

  // Draw area inside value rect
  w.DrawRect(BoxRect, WhiteGray());

  // Draw border around value rect
  w.DrawLine(BoxRect.left, BoxRect.top, BoxRect.right, BoxRect.top, Black());
  w.DrawLine(BoxRect.left, BoxRect.top, BoxRect.left, BoxRect.bottom, Black());
  w.DrawLine(BoxRect.right, BoxRect.top, BoxRect.right, BoxRect.bottom, Black());
  w.DrawLine(BoxRect.left, BoxRect.bottom, BoxRect.right + 1, BoxRect.bottom, Black());

  // Draw value
  w.DrawText(TextRect.left, BoxRect.top, DataString);
 }
}

void LabelledSliderCtrl::LeftDown(Point pos, dword keyflags)
{
 SliderCtrl::LeftDown(pos, keyflags);

 // Set a flag to indicate that the value should be drawn
 DrawValue = true;
}

void LabelledSliderCtrl::LeftUp(Point pos, dword keyflags)
{
 SliderCtrl::LeftUp(pos, keyflags);

 // Set a flag to indicate that the value should not be drawn
 DrawValue = false;
```

```
}

void LabelledSliderCtrl::SetTickStep(int NewStep)
{
 TickStep = NewStep;
}

void LabelledSliderCtrl::SetTickSize(int NewSize)
{
 TickSize = NewSize;
}

void LabelledSliderCtrl::SetSmallTickSize(int NewSize)
{
 SmallTickSize = NewSize;
}

void LabelledSliderCtrl::SetValueOffset(int NewOffset)
{
 ValueOffset = NewOffset;
}

void LabelledSliderCtrl::SetValueSpacing(int NewSpacing)
{
 ValueSpacing = NewSpacing;
}
```