Subject: Re: Repainting of DropList
Posted by crydev on Sat, 23 Feb 2019 09:29:44 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I'm sorry, I use a 'virtual' DropList made by somebody on the forum a few years ago. I realized that I should go through the logic I use there, and try to add something to fix it. I use the following (maybe something for the U++ framework if not already in it):

VirtualDropList.h

```
#ifndef _VirtualDropList_VirtualDropList_h_
#define _VirtualDropList_VirtualDropList_h_

NAMESPACE_UPP

class VirtualPopUpTable : public PopUpTable {
public:
 void Virtual() {
  AddRowNumColumn(String()).Accel();
  NoHeader();
  HeaderTab(0).SetMargin(0);
  MouseMoveCursor();
  NoGrid();
  AutoHideSb();
  SetLineCy(Draw::GetStdFontCy());
  NoPopUpEx();
 }
};

class VirtualDropList : public MultiButton {
protected:
 VirtualPopUpTable list;
 int index;
 int dropwidth;
 const Convert *valueconvert;
 const Display *valuedisplay;
 bool dropfocus;
 bool notnull;
 bool alwaysdrop;
 bool usewheel;
public:
 VirtualDropList();
 ~VirtualDropList() { }

 typedef VirtualDropList CLASSNAME;
```

```
typedef MultiButton::Style Style;

Callback WhenDrop;

virtual void MouseWheel(Point p, int zdelta, dword keyflags);
virtual bool Key(dword key, int);

void Drop();
void Sync();
void Cancel();
void Select();
void Change(int q);

void EnableDrop(bool b = true) { MainButton().Enable(b || alwaysdrop); }
int  GetCount() const         { return list.GetCount(); }
void SetCount(int c)          { list.SetCount(c); EnableDrop(c > 0 ? true : false); Sync(); }

void SetData(const Value& data) { SetIndex(data); }
Value GetData() const           { Value value = GetValue(); return notnull && IsNull(value) ?
NotNullError() : value; }
Value GetValue(int i) const     { return valueconvert->Format(i); }
Value GetValue() const          { return index < 0 ? Null : GetValue(index); }
Value operator[](int i) const   { return GetValue(i); }

void SetIndex(int i);
int GetIndex() const { return index; }
void GoBegin()       { if (GetCount()) SetIndex(0); }
void GoEnd()         { if (GetCount()) SetIndex(GetCount() - 1); }

const VirtualPopUpTable& GetList() const { return list; }
VirtualPopUpTable& ListObject()          { return list; }

VirtualDropList& SetDropLines(int d)                { list.SetDropLines(d); return *this; }
VirtualDropList& SetValueConvert(const Convert& cv);
VirtualDropList& SetConvert(const Convert& cv);
VirtualDropList& SetDisplay(int i, const Display& d);
VirtualDropList& SetDisplay(const Display& d);
VirtualDropList& SetLineCy(int lcy)                 { list.SetLineCy(lcy); return *this; }
VirtualDropList& SetDisplay(const Display& d, int lcy);
VirtualDropList& ValueDisplay(const Display& d);
VirtualDropList& DropFocus(bool b = true)           { dropfocus = b; return *this; }
VirtualDropList& NoDropFocus()                      { return DropFocus(false); }
VirtualDropList& AlwaysDrop(bool e = true);
VirtualDropList& SetStyle(const Style& s)           { MultiButton::SetStyle(s); return *this; }
VirtualDropList& NotNull(bool b = true)             { notnull = b; return *this; }
VirtualDropList& DropWidth(int w)                   { dropwidth = w; return *this; }
VirtualDropList& DropWidthZ(int w)                  { dropwidth = HorzLayoutZoom(w); return *this; }
VirtualDropList& Wheel(bool b = true)               { usewheel = b; return *this; }
```

```
 VirtualDropList& NoWheel()                    { return Wheel(false); }

 VirtualDropList& SetScrollBarStyle(const ScrollBar::Style& s) { list.SetScrollBarStyle(s); return
*this; }
};

END_UPP_NAMESPACE

#endif
```

VirtualDropList.cpp

```
#include <CtrlLib/CtrlLib.h>
#include "VirtualDropList.h"

NAMESPACE_UPP

VirtualDropList::VirtualDropList() : index(-1), valueconvert(&NoConvert()), valuedisplay(NULL),
 dropfocus(false), notnull(false), alwaysdrop(false), dropwidth(0), usewheel(true)
{
 AddButton().Main().WhenPush = THISBACK(Drop);
 NoInitFocus();
 EnableDrop(false);
 list.Virtual();
 list.WhenCancel = THISBACK(Cancel);
 list.WhenSelect = THISBACK(Select);
}

void VirtualDropList::MouseWheel(Point p, int zdelta, dword keyflags)
{
 if (usewheel)
  Change(zdelta < 0 ? 1 : -1);
}

bool VirtualDropList::Key(dword key, int)
{
 if (IsReadOnly())
  return false;
 switch(key) {
 case K_ALT_DOWN:
  Drop();
  break;
 case K_DOWN:
 case K_RIGHT:
  Change(1);
  break;
```

```
 case K_UP:
 case K_LEFT:
  Change(-1);
  break;
 default:
  if (key >= 32 && key < 65536) {
   bool b = list.Key(key, 1);
   int c = list.GetCursor();
   if (c >= 0 && c < list.GetCount())
    Select();
   return b;
  }
  return false;
 }
 return true;
}

void VirtualDropList::Drop()
{
 if (IsReadOnly())
  return;
 if(dropfocus)
  SetFocus();
 WhenDrop();
 list.SetCursor(index);
 list.PopUp(this, dropwidth);
}

void VirtualDropList::Sync() {
 const Display& d = valuedisplay ? *valuedisplay : index >= 0 ? list.GetDisplay(index, 0)
                                                  : list.GetDisplay(0);
 MultiButton::SetDisplay(d);
 MultiButton::SetValueCy(list.GetLineCy());
 if (index >= 0)
  Set(valueconvert->Format(index));
}

void VirtualDropList::Cancel()
{
 if (dropfocus)
  SetFocus();
 Sync();
}

void VirtualDropList::Select()
{
 int c = list.GetCursor();
 if (c >= 0)
```

```
  index = c;
 if (dropfocus)
  SetFocus();
 Sync();
 UpdateAction();
}

void VirtualDropList::Change(int q)
{
 int count = list.GetCount();
 if (count == 0)
  return;
 int c = index + q;
 if (c < 0)
  c = 0;
 else if (c >= count)
  c = count - 1;
 SetIndex(c);
}

void VirtualDropList::SetIndex(int i)
{
 if (i == index)
  return;
 index = i;
 Update();
 Sync();
}

VirtualDropList& VirtualDropList::SetValueConvert(const Convert& cv)
{
 valueconvert = &cv;
 Sync();
 return *this;
}

VirtualDropList& VirtualDropList::SetConvert(const Convert& cv)
{
 list.ColumnAt(0).SetConvert(cv);
 return SetValueConvert(cv);
}

VirtualDropList& VirtualDropList::SetDisplay(int i, const Display& d)
{
 list.SetDisplay(i, 0, d);
 Sync();
 return *this;
}
```

```
VirtualDropList& VirtualDropList::SetDisplay(const Display& d)
{
 list.ColumnAt(0).SetDisplay(d);
 Sync();
 return *this;
}

VirtualDropList& VirtualDropList::SetDisplay(const Display& d, int lcy)
{
 SetDisplay(d);
 SetLineCy(lcy);
 Sync();
 return *this;
}

VirtualDropList& VirtualDropList::ValueDisplay(const Display& d)
{
 valuedisplay = &d;
 Sync();
 return *this;
}

VirtualDropList& VirtualDropList::AlwaysDrop(bool e)
{
 alwaysdrop = e;
 if (e)
  EnableDrop();
 return *this;
}

END_UPP_NAMESPACE
```

I use the VirtualDropList::SetCount(c) function to set the virtual element count, and the function looked as follows before:

```
void SetCount(int c)          { list.SetCount(c); EnableDrop(c > 0 ? true : false); }
```

I added a call to Sync(); at the end of this function, and now the redrawing works as expected. :)

Thanks for your reply! You set me thinking about what is actually under the hood of the control. :)

crydev