Subject: Re: GridCtrl performance
Posted by Sender Ghost on Thu, 25 Apr 2013 16:53:21 GMT
View Forum Message <> Reply to Message

crydev wrote on Thu, 25 April 2013 17:49However, I'm having trouble implementing my own data in it. Say I have the following data, how could this be properly implemented using the virtual ArrayCtrl?

```
template <class T>
struct MemData
{
 int address; // column address
 T value; // column value
};
```

Depends from the type of T. If T is String, then modified version of example application might look like follows:
Toggle Spoiler

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

using namespace Upp;

int count = 0x100000;

template <class T>
struct MemData : Moveable<MemData<T> >
{
 int address; // column address
 T value; // column value
};

typedef MemData<String> StringMemData;
Vector<StringMemData> data;

template <String (GetData) (int index)>
struct NumberToText : public Convert {
 virtual Value Format(const Value& q) const {
  return GetData(int(q));
 }
};

String GetAddress(int index)
{
 ASSERT(index >= 0 && index < data.GetCount());
 return Format64Hex(data[index].address);
```

```
}

String GetIndexValue(int index)
{
 ASSERT(index >= 0 && index < data.GetCount());
 return data[index].value;
}

String GetReverseValue(int index)
{
 ASSERT(index >= 0 && index < data.GetCount());
 return data[count - index - 1].value;
}

StringMemData FormatData(int index)
{
 StringMemData d;
 d.address = index;
 d.value = NFormat("Data #%d", index + 1);

 return d;
}

class App : public TopWindow {
public:
 typedef App CLASSNAME;
 App();

 // Ctrls
 ArrayCtrl list;
 Button btnAdd, btnRemove;
 // Events
 void OnAdd();
 void OnRemove();
};

const int addition = count / 2;

void App::OnAdd()
{
 const int prevCount = count;
 count += addition;
 data.SetCount(count);

 for (int i = prevCount; i < count; ++i)
  data[i] = FormatData(i);

 list.SetVirtualCount(count);
```

```
}

void App::OnRemove()
{
 count -= addition;
 if (count < 0)
  count = 0;

 data.SetCount(count);
 list.SetVirtualCount(count);
}

App::App()
{
 Title("Virtual ArrayCtrl");
 Sizeable().Zoomable();
 const Size sz(640, 480);
 SetRect(sz); SetMinSize(sz);

 btnAdd.SetLabel("Add") <<= THISBACK(OnAdd);
 btnRemove.SetLabel("Remove") <<= THISBACK(OnRemove);

 list.AddRowNumColumn("Address", 12).SetConvert(Single<NumberToText<GetAddress> >());
 list.AddRowNumColumn("Value", 44).SetConvert(Single<NumberToText<GetIndexValue> >());
 list.AddRowNumColumn("Reverse Value",
44).SetConvert(Single<NumberToText<GetReverseValue> >());
 list.SetVirtualCount(count);

 const int offset = 75;
 Add(btnAdd.TopPosZ(4, 20).LeftPosZ(4, offset));
 Add(btnRemove.TopPosZ(4, 20).LeftPosZ(offset + 8, offset));
 Add(list.HSizePosZ(4, 4).VSizePosZ(28, 4));
}

GUI_APP_MAIN
{
 Ctrl::GlobalBackPaint();

 data.SetCount(count);
 {
  Progress p;
  p.SetText("Preparing the data..");
  for (int i = 0; i < count; ++i) {
   data[i] = FormatData(i);
   p.Set(i, count);
  }
 }
```

```
 App app;
 app.Run();
}
```

If you need to use another types, then just change the function declaration of NumberToText template, but return compatible types for ArrayCtrl values (which is Value).
