Subject: GridCtrl performance
Posted by crydev on Mon, 22 Apr 2013 14:36:59 GMT
View Forum Message <> Reply to Message

Hello,

I am working on a memory scanner project. I use a GridCtrl to visualize the results from a scan to the user. However, I notice that the ArrayCtrl, as well as the GridCtrl are very slow when it comes to adding ~300.000+ items at once. I have 8 threads scanning the memory so the scan is done very fast. The adding and removing of the items from the list however, is slow.

Is there a way to make it faster? Why is it so slow?

Thanks,

Crydev

Subject: Re: GridCtrl performance
Posted by BioBytes on Mon, 22 Apr 2013 19:43:35 GMT
View Forum Message <> Reply to Message

Hello Crydev,

Did you try SetVirtualCount method in ArrayCtrl ?

Regards

Biobytes

Subject: Re: GridCtrl performance
Posted by crydev on Mon, 22 Apr 2013 20:25:14 GMT
View Forum Message <> Reply to Message

BioBytes wrote on Mon, 22 April 2013 21:43Hello Crydev,

Did you try SetVirtualCount method in ArrayCtrl ?

Regards

Biobytes

Yes I tried using it but unfortunately it didn't solve my problem. If I could safely use the multithreaded structure to fill up my GridCtrl it would speed up very much but because it is GUI it isn't possible.

Subject: Re: GridCtrl performance
Posted by unodgs on Mon, 22 Apr 2013 20:50:41 GMT
View Forum Message <> Reply to Message

GridCtrl uses Vector<Vector<Item>> internally. So inserting is as fast as Vector implementation is. You should observe dramatic speed improvement if you compile your app in release mode (but I assume you did). If you insert many rows at once you can put insertion part between grid.Ready(false) and grid.Ready(true).
You can also add 300.000 empty rows to the grid in the beginning and simply clear and set particular cells. This way you would avoid removing and inserting rows over and over again.
ArrayCtrl and virtual rows should solve your problem. It seems like you did something wrong, but without the real code it's hard to tell.

Subject: Re: GridCtrl performance
Posted by crydev on Thu, 25 Apr 2013 09:15:48 GMT
View Forum Message <> Reply to Message

I see. I tried doing the posted things but no luck, unfortunately.

Is there a more skinned or lightweight list-like control that I can use for this? I'm not sure about how I can customize a control in a way that it will act faster in my situation.

Another question that pops up to me: Is it possible to link the values displayed in the Array/GridCtrl as pointers to the original data? I think I am saving a lot of data redundantly, which is not nessecary ofcourse.

Subject: Re: GridCtrl performance
Posted by Sender Ghost on Thu, 25 Apr 2013 14:47:23 GMT
View Forum Message <> Reply to Message

Hello.
crydev wrote on Mon, 22 April 2013 22:25BioBytes wrote on Mon, 22 April 2013 21:43Hello Crydev,

Did you try SetVirtualCount method in ArrayCtrl ?

Regards

Biobytes

Yes I tried using it but unfortunately it didn't solve my problem. If I could safely use the multithreaded structure to fill up my GridCtrl it would speed up very much but because it is GUI it isn't possible.
I don't know how you tried it, but try again.

Below is additional example of VirtualArray reference application:
Toggle Spoiler

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

using namespace Upp;

int count = 0x100000;
Vector<String> data;

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

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

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

String FormatData(int index)
{
 return NFormat("Data #%d", index + 1);
}

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;
```

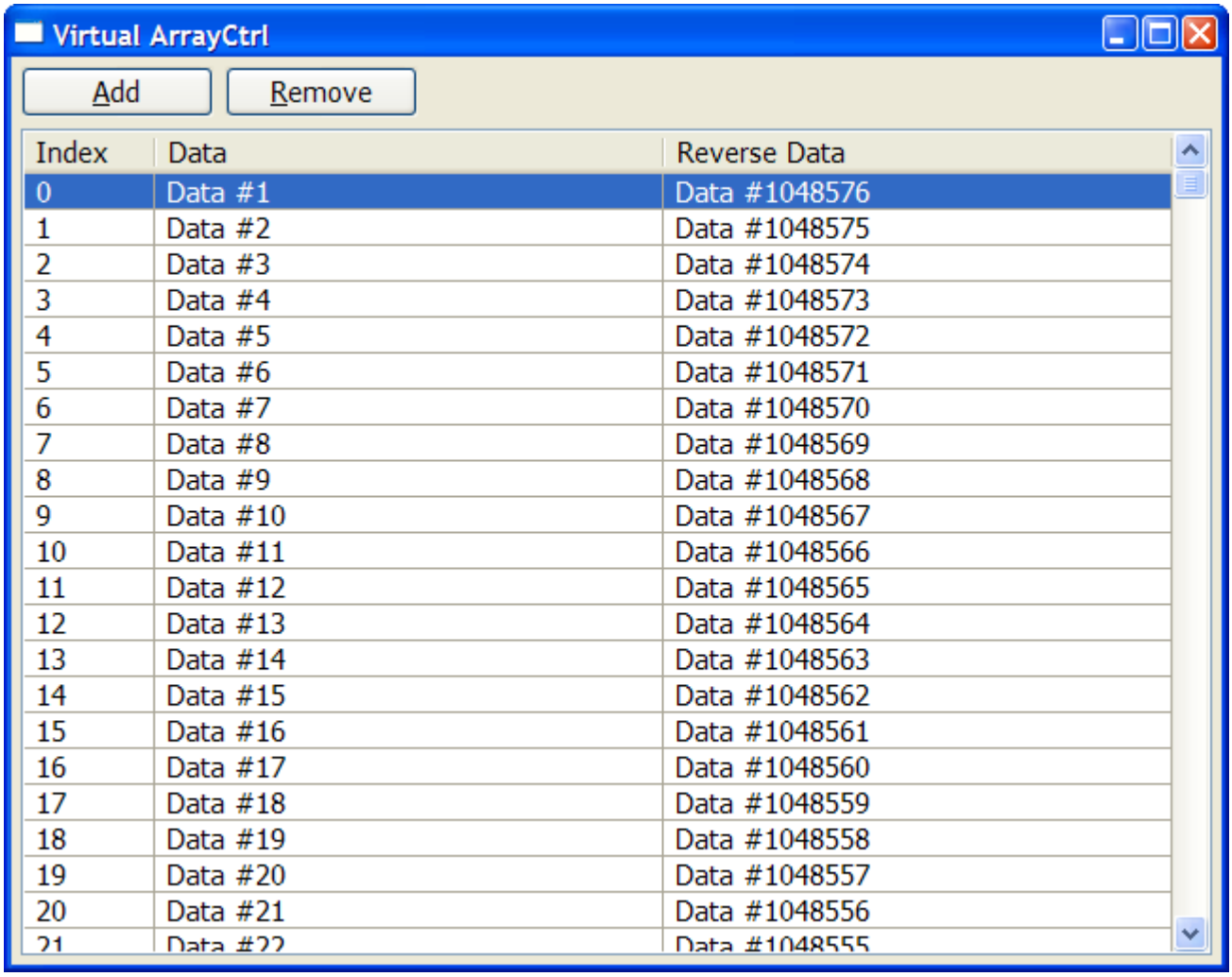
```
void App::OnAdd()
{
 const int prevCount = count;
 count += addition;
 data.SetCount(count);

 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);

 HeaderCtrl::Column& hc = list.AddRowNumColumn("Index").HeaderTab();
 const int textWidth = GetTextSize(AsString(count), Draw::GetStdFont()).cx,
  columnWidth = textWidth + hc.GetMargin() * 2 + 1;
 hc.Fixed(columnWidth);

 list.AddRowNumColumn("Data", 50).SetConvert(Single<NumberToText<GetIndexData> >());
 list.AddRowNumColumn("Reverse Data",
50).SetConvert(Single<NumberToText<GetReverseData> >());
 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();
}
```

Where instead of Vector<String> you could use your "faster" container, filled by parallel, if you want. Or even try to show real time (or cached) results, calculated by some index, without using any container.

File Attachments

1) VirtualArrayCtrl.png, downloaded 767 times



Subject: Re: GridCtrl performance
Posted by crydev on Thu, 25 Apr 2013 15:49:00 GMT
View Forum Message <> Reply to Message

Thanks a lot Sender Ghost, you cleared it up for me, I see how it works. However, 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
};
```

Thanks in advance!

Crydev

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).

Subject: Re: GridCtrl performance
Posted by crydev on Fri, 26 Apr 2013 11:33:14 GMT
View Forum Message <> Reply to Message

Thanks Sender Ghost. I got it to work now and it is at least 10x faster. As others may be able to get the same problem as I have, I'll post my solution here.

My problem was mainly the use of templated structs that could have 7 different types, that also needed a different cast.

```
template <String (GetData) (SearchResult& instance, const int index)>
struct MemoryBlockValueConvert : public Convert
{
 virtual Value Format(const Value& q) const
 {
  return GetData(mMemoryScanner->GetSearchResult(), int(q));
 }
};

String GetAddress(SearchResult& instance, const int index)
{
 return FormatIntHexUpper(instance.mFirstMillionResults[index]->Address, 0);
}

String GetValue(SearchResult& instance, const int index)
{
 switch (mMemoryScanner->GetSearchResult().GetResultType())
 {
  case VALUETYPE_BYTE:
   return IntStr(((MemoryBlock<Byte>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_2BYTE:
   return IntStr(((MemoryBlock<short>*)instance.mFirstMillionResults[index])->Buffer);
   break;
```

```
  case VALUETYPE_4BYTE:
   return IntStr(((MemoryBlock<int>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_8BYTE:
   return IntStr64(((MemoryBlock<__int64>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_FLOAT:
   return DblStr(((MemoryBlock<float>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_DOUBLE:
   return DblStr(((MemoryBlock<double>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_STRING:
   return ((MemoryBlock<String>*)instance.mFirstMillionResults[index])->Buffer;
   break;
  default:
   return STRING_EMPTY;
   break;
 }
}
```

The rest is the same as Sender Ghost posted. Just call:
SetVirtualCount(mFirstMillionResults.GetCount()) to add all the items to the ArrayCtrl.

Thanks again for helping me out!
