Subject: Restrict drag&drop to one level
Posted by dolik.rce on Sun, 14 Feb 2010 10:14:33 GMT
View Forum Message <> Reply to Message

Hello!

I've got a little problem here. I want to have a TreeCtrl with drag and drop implemented in such a way, that items from one level can not be inserted into different level. An example to illustrate: + Graphs
  + Graph1
  | + Data1
  | + Data2
  + Graph2
    + Data3I need to allow user to drag any Data into any Graph and disable dropping it into any other level. Also, any Graph must not be dropped anywhere else then top level.

I think this is quite common situation, so there should be easy solution. I'm probably just missing something Bellow is the code I got so far, but it has two problems. First, it doesn't work properly with MultiSelect(). Second, it paints insert marks even in positions where drop is not allowed, which might confuse user. I hope someone can give me some hint how to solve this. The code:

```
#include "CtrlLib/CtrlLib.h"
using namespace Upp;

struct App : TopWindow {
 typedef App CLASSNAME;
 TreeCtrl   tree;

 void DropInsert(int parent, int ii, PasteClip& d){
  if(AcceptInternal<TreeCtrl>(d, "graph")) {
   if(GetLevel(parent)!=0) {d.Reject();return;}
   tree.InsertDrop(parent, ii, d);
   tree.SetFocus();
   LOG("accepted graph");
   return;
  }
  if(AcceptInternal<TreeCtrl>(d, "data")) {
   if(GetLevel(parent)!=1) {d.Reject();return;}
   tree.InsertDrop(parent, ii, d);
   tree.SetFocus();
   LOG("Accepted series");
   return;
  }
 }

 void Drag() {
  int type=GetLevel(tree.GetCursor());
  switch(type) {
  case 1:
```

```
   if(tree.DoDragAndDrop(InternalClip(tree, "graph"),tree.GetDragSample()) == DND_MOVE)
    tree.RemoveSelection();
   break;
  case 2:
   if(tree.DoDragAndDrop(InternalClip(tree, "data"),tree.GetDragSample()) == DND_MOVE)
    tree.RemoveSelection();
   break;
  }
 }

 int GetLevel(int id){
  int i=0;
  while(id!=0){
   id=tree.GetParent(id);
   i++;
  }
  return i;
 }

 App() {
  Add(tree.SizePos());
  tree.SetRoot(Image(), "Graphs");
  for(int i=1;i<3;i++){
   int id=tree.Add(0,Image(),"Graph "+AsString(i));
   for(int j=1;j<3;j++)
    tree.Add(id,Image(),"Data "+AsString(i)+"/"+AsString(j));
  }
  tree.OpenDeep(0);
  tree.WhenDropInsert = THISBACK(DropInsert);
  tree.WhenDrag = THISBACK(Drag);
  tree.MultiSelect();
  Sizeable();
 }
};

GUI_APP_MAIN
{
 App().Run();
}
```

Best regards,
Honza

Subject: Re: Restrict drag&drop to one level
Posted by mrjt on Thu, 01 Apr 2010 11:22:05 GMT
View Forum Message <> Reply to Message

A bit late replying, but it's an interesting situation and as you say someone else may want the same behaviour.

The DnD stuff is quite elegant but unfortunately quite difficult to understand. We can't all be as smart as Mirek

Basically you have to know that Accept<> and AcceptInternal<> serve a dual function. When the user is dragging (ie on MouseMove) the function does the accept but always returns false so that when you use it in an 'if' statement the actual insert code is avoided. Only when actually doing the 'drop' does it ever return true!

Therefor, to add the level check you must reproduce some of the behaviour from the Accept<> function:

```
 void DropInsert(int parent, int ii, PasteClip& d){
  // Check type of drag data, and restrict to level
  if (IsAvailableInternal<TreeCtrl>(d, "graph") && GetLevel(parent) == 0) {
   // Yes we like this data
   d.Accept();
   // If we haven't dropped the data yet (we are still dragging) don't do anything
   if (!d.IsPaste()) return;
                        // The user has dropped it! Do the insert
   tree.InsertDrop(parent, ii, d);
   tree.SetFocus();
   LOG("accepted graph");
   return;
  }
  if (IsAvailableInternal<TreeCtrl>(d, "data") && GetLevel(parent) == 1) {
   d.Accept();
   if (!d.IsPaste()) return;
   tree.InsertDrop(parent, ii, d);
   tree.SetFocus();
   LOG("accepted data");
   return;
  }
 }
```

You could replace the 'd.Accept' and 'if (d.IsPaste())' with:
if (!AcceptInternal<TreeCtrl>(d, "data")) return;
but it's marginally less efficient.

Subject: Re: Restrict drag&drop to one level
Posted by dolik.rce on Thu, 01 Apr 2010 18:55:28 GMT
View Forum Message <> Reply to Message

Hello James,
Thanks for your answer! Better late than never I still didn't solve it, so your hints are more than

welcome. I'll report here when I get time to test it...

Thank you,
Honza

Subject: Re: Restrict drag&drop to one level
Posted by dolik.rce on Thu, 01 Apr 2010 21:59:09 GMT
View Forum Message <> Reply to Message

It works almost perfect. I was missing the IsAvailableInternal<> function, that helps a lot But few problems still remain:

First, as I am using MultiSelect(), the level restriction fails when multiple levels are in selection. I solved this by adding following function as WhenSel callback: void CheckSel(){

```
  Vector<int> sel=tree.GetSel();
  int last=sel.GetCount()-1;
  if (last<1) return;
  if (GetLevel(sel[0])!=GetLevel(sel[last]))
   tree.SelectOne(sel[last],false);
 }
```

It is not perfect solution, but works reasonably with minimal effort.

Another problem is, that if you drag data item to root (or anywhere below the tree), it disappears on drop. I think it is because it just slips through the restriction rules without triggering any of them, but I couldn't find how to prevent that. Any ideas?

Best regards,
Honza

Subject: Re: Restrict drag&drop to one level
Posted by mrjt on Thu, 08 Apr 2010 10:58:12 GMT
View Forum Message <> Reply to Message

I think this selection filter would work better:

```
 void OnSel() {
  Vector<int> sel = tree.GetSel();
  int level = GetLevel(tree.GetCursor());
  for (int i = 0; i < sel.GetCount(); i++)
   if (GetLevel(sel[i]) != level)
    tree.SelectOne(sel[i], false);
 }
```

It works with both Ctrl select and Shift select and preserves the most recently selected.

I don't have thwe problem you describe with draggin to root/the bottom of the tree. Have you tested with the latest SVN?
