Subject: Drag and Drop for TreeCtrl
Posted by nixnixnix on Tue, 24 Apr 2007 18:17:59 GMT
View Forum Message <> Reply to Message

I wasn't brave enough to post this under Drag&Drop as I suspect a flood of people correcting me that it is not "true drag and drop" so am putting it here for people to build on / use / help me improve if you feel like it.

I am using a node of the following type

```
struct TreeOption : Option {
 virtual void LeftDown(Point p, dword keyflags)
 {
  if(p.x<this->GetSize().cy)
   Option::LeftDown(p, keyflags);
  if(this->GetData()==true)
   SetFocus();

  // now send pass on this event ot the parent for possible drag and drop
  Ctrl* pCtrl = this->GetParent();
  Point pt = p + GetRect().TopLeft(); // transform event into parent coordinates
  if(p.x>this->GetSize().cy)
   pCtrl->LeftDown(pt,keyflags);

  Refresh();
 }

};
```

my customised version of the RectTracker and TreeCtrl classes look like this

```
class LayerTracker : public RectTracker
{
public:
 typedef LayerTracker CLASSNAME;

 LayerTracker(Ctrl& ctrl) : RectTracker(ctrl){}
 virtual ~LayerTracker() {}

 Image m_image;

 void DrawRect(Rect rc1,Rect rc2);

 void SetImage(Image img) {m_image = img;}
```

```
};


class LayerTree : public TreeCtrl
{
public:
 typedef LayerTree CLASSNAME;

 LayerTree();
 ~LayerTree();

 virtual void LeftDown(Point p, dword flags);

 bool Drop(int idTarget,int id);

 int GetNodeIDAt(Point p);

 // remember the visible nodes
 Array <Ctrl*> m_pncs;



};



LayerTree::LayerTree()
{

}

LayerTree::~LayerTree()
{


}

void LayerTracker::DrawRect(Rect rc1,Rect rc2)
{
 ViewDraw w(&GetMaster());

 GetMaster().Paint(w);

 LayerTree* pT = (LayerTree*)&GetMaster();

 for(int i=0;i<pT->m_pncs.GetCount();i++)
```

```
 {
  Rect rc = pT->m_pncs[i]->GetRect();
  pT->m_pncs[i]->DrawCtrl(w,rc.left,rc.top);
 }

 w.DrawImage(rc2,m_image);

}

void LayerTree::LeftDown(Point p, dword flags)
{

 // first see if there is a node at this point p
 int i,id=GetNodeIDAt(p);

 if(id<=0)
  return;  // didnt click on a node

 Rect rc = GetNode(id).ctrl->GetRect();

 // draw the node into an image
 // sample the tree ctrls view in this rectangle and make an image
 Size sz(rc.Width(),rc.Height());

 ImageDraw w(sz);

 w.DrawRect(GetSize(), SWhite);
 GetNode(id).ctrl->DrawCtrl(w);

 Image img = w;

 // start local loop and see where it ends
 LayerTracker rt(*this);

 rt.SetImage(img);

 rt.Track(rc,ALIGN_CENTER,ALIGN_CENTER);

 int idTarget = GetNodeIDAt(rt.Get().CenterPoint());

 Drop(idTarget,id);

 // now pass on to default behaviour
 TreeCtrl::LeftDown(p,flags); // hmmm - doesnt appear to work
}

bool LayerTree::Drop(int idTarget,int id)
{
```

```
if(idTarget>=0 && idTarget!=id)
{

 // check that this node can accept the dragged node
 // this code will be application specific


 // your test here - if false return false




 // also check that idTarget is not a child of id
 int parent,child=idTarget;
 do{
  parent = GetParent(child);

  if(parent==id)
  {
   return false;
  }

  child = parent;
 }while(parent!=0);



 // check for children
 // WHAT ABOUT THE CHILDREN!!!



 // add id to idTarget and remove id
 TreeCtrl::Node node = GetNode(id);
 this->Remove(id);
 this->Add(idTarget,node);
 this->Open(idTarget);



}
else if(idTarget<0)
{
 // drop onto root
 TreeCtrl::Node node = GetNode(id);
 this->Remove(id);
 this->Add(0,node);
 this->Open(0);
```

```
 }
}

int LayerTree::GetNodeIDAt(Point p)
{
 // step through all VISIBLE nodes in the tree
 // and see if one of them contains p
 int id,ID = -1;
 int n=GetLineCount(); //all (visible) items
 Ctrl* ptr;
 Rect rc;
 m_pncs.SetCount(0);

 for(int i=0;i<n;i++)
 {
  id = GetItemAtLine(i);
  TreeCtrl::Node node = GetNode(id);
  ptr = node.ctrl;
  if(ptr)
  {
   m_pncs.Add(ptr); // need to remember this so I can repaint it later if I need to

   rc = ptr->GetRect();

   if(rc.Contains(p))
    ID = id;
  }
 }

 return ID;
}
```

I think it all works apart from the open/close behaviour and the issue of what to do with the children of the dragged node. I occasionally get a stranded node but I suspect this will be fixed when I work out what to do with the children.

Please let me know what you think and if you use it and improve it I would appreciate a copy.

Cheers,

Nick

p.s. the layers I refer to are GIS style layers but it should be applicable to any form of info that you want to arrange interactively in a tree view

p.p.s if am spamming the server with my poorly written code I apologise in advance but after all the kind help I received from Mirek on this I felt the overwhelming need to give something back now its (almost) working

---

Subject: Re: Drag and Drop for TreeCtrl
Posted by mrjt on Wed, 25 Apr 2007 15:39:51 GMT
View Forum Message <> Reply to Message

Some code to deal with the children:

```
void MoveChildren(int src, int dest)
{
 while (GetChildCount(src)) {
  int child = GetChild(src, 0);
  TreeCtrl::Node node;

  node = GetNode(child);
  MoveChildren(child, Add(dest, node));
  Remove(child);
 }
}
```

And use it like so (removal of source node must happen after addition of destination - for obvious reasons):

```
    // add id to idTarget and remove id
    TreeCtrl::Node node = GetNode(id);
    MoveChildren(id, this->Add(idTarget,node));
    this->Remove(id);
    this->Open(idTarget);
```

I've got a version of this that doesn't need a RectTracker or a custom Option control that I'll post if you're interested.

James

Subject: Re: Drag and Drop for TreeCtrl
Posted by nixnixnix on Wed, 25 Apr 2007 17:29:35 GMT
View Forum Message <> Reply to Message

Thanks James,

That does the trick nicely . I know I could have done it using LeftDown, MouseMove, and LeftUp but that would have spread code out through other modules in other places and I wanted to keep

this contained so for me the RectTracker is a good way to go. Also, I need a custom node for other reasons as I use it to keep track of and edit data objects in my app.

I would be interested to see your solution though.

The bottom line in my LeftDown function should of course be just after "if(id<=0)" and before the "return" near the top of the function so that the default event processing can take place for opening and closing nodes etc.

Phew! That took me way longer than it should but learnt lots about UPP

Nick
