Subject: Re: how to add runtime StaticText and hook mouse events for it
Posted by fudadmin on Mon, 22 May 2006 08:39:47 GMT
View Forum Message <> Reply to Message

ok, my experimental drag&drop control....

```
#include <CtrlLib/CtrlLib.h>
#define LLOG(x)

class DragDropCtrl : public Button
{
private:
 Point       start, last, pdelta, pstartdelta;
 LogPos      pos;
 Rect        RC_last, RC_next, RC_start;
 dword       last_key;
 char        state;
 enum { OFF, ON, POSSIBLE, HIDDEN };

public:
 DragDropCtrl();

 // drag & drop interface
 virtual bool Push(Point pt, dword keyflags);
 virtual void Drag(Point pt, Point last, Point next, dword keyflags);
 virtual void Drop(Point pt, Point end, dword keyflags);
 virtual void Click(Point pt, dword keyflags);
 static const int DBLCLK = 0x40000000;

 virtual void DragRect(const Rect& last, const Rect& next, dword keyflags);
 virtual void DropRect(const Rect& rc, dword keyflags);

 void        DragShow(bool _show = true);
 void        DragHide()                  { DragShow(false); }
 void        DragStop(bool accept, dword keyflags);
 void        DragStop(bool accept = false) { DragStop(accept, last_key); }

 bool        IsDragging() const          { return state == ON || state == HIDDEN; }
 bool        IsPushed() const            { return state != OFF; }

 // control overrides
 virtual void LeftDown(Point pt, dword keyflags);
 virtual void LeftDouble(Point pt, dword keyflags);
 virtual void LeftUp(Point pt, dword keyflags);
 virtual void MouseMove(Point pt, dword keyflags);
 virtual bool Key(dword key, int repcnt);

};
```

```
//=============helpers start=================
inline Rect SortRect(Point p1, Point p2)
{
 return Rect(min(p1.x, p2.x), min(p1.y, p2.y), max(p1.x, p2.x) + 1, max(p1.y, p2.y) + 1);
}

bool IsDragDistance(Point pt1, Point pt2)
{
#ifdef PLATFORM_WIN32
 return tabs(pt1.x - pt2.x) >= GetSystemMetrics(SM_CXDRAG)
  || tabs(pt1.y - pt2.y) >= GetSystemMetrics(SM_CYDRAG);
#endif
#ifdef PLATFORM_POSIX
 enum { CXDRAG = 4, CYDRAG = 4 };
 // todo? are there any CXDRAG / CYDRAG system metrics in LINUX?
 return tabs(pt1.x - pt2.x) >= CXDRAG || tabs(pt1.y - pt2.y) >= CYDRAG;
#endif
}

//what does this do?
int GetRectDragMask(Rect rc, Point pt, int tolerance)
{
 Point center = rc.CenterPoint();
 int m = (tabs(rc.left   - pt.x) <= tolerance ?  1 : 0)
  |   (tabs(rc.top    - pt.y) <= tolerance ?  2 : 0)
  |   (tabs(rc.right  - pt.x) <= tolerance ?  4 : 0)
  |   (tabs(rc.bottom - pt.y) <= tolerance ?  8 : 0);
 if(m & 5)
  if((m & 10) || tabs(center.y - pt.y) <= tolerance)
   return m;
 if(m & 10)
  if((m & 5) || tabs(center.x - pt.x) <= tolerance)
   return m;
 return 0;
}

//==========helpers end====================

DragDropCtrl::DragDropCtrl()
: state(0)
{
}

bool DragDropCtrl::Push(Point pt, dword keyflags)
{
 return true;
```

```
}

void DragDropCtrl::DragRect(const Rect& last, const Rect& next, dword keyflags)
{
 ViewDraw draw(this);
 DrawDragRect(draw, last, next, draw.GetClip(), 1, Yellow(), NULL);
}

void DragDropCtrl::LeftDown(Point pt, dword keyflags)
{
 SetWantFocus();
 LLOG("DragDropCtrl::LeftDown -> " << pt << ", keyflags " << FormatIntHex(keyflags));
 if(Push(pt, last_key = keyflags))
 { // begin drag & drop
  state = POSSIBLE;
  start = last = pt;
  SetCapture();
  pstartdelta=GetRect().TopLeft()-start; //aris
  RC_start=GetRect();
//  SetLabel("leftdown "+AsString(pstartdelta));

 }

}

void DragDropCtrl::Drag(Point pt, Point last, Point next, dword keyflags)
{
 Rect rc_last = Null, rc_next = Null;
 if(!IsNull(last))
  rc_last = RectSort(pt, last);
 if(!IsNull(next))
  rc_next = RectSort(pt, next);
 if(rc_last != rc_next){
  DragRect(rc_last, rc_next, keyflags);
  pdelta=next-last; //aris

 Rect newrc= RC_start+next-pt;// + pdelta;//pdelta;
 SetLabel("DRAG last-next:"+AsString(pstartdelta));
  SetRect(newrc);
  RC_start=GetRect();
 }
}

void DragDropCtrl::Drop(Point pt, Point end, dword keyflags)
{
 DropRect(RectSort(pt, end), keyflags);
}
```

```
void DragDropCtrl::Click(Point pt, dword keyflags)
{
 // no-op, should be implemented in derived class
 //PromptOK("click");
// DragShow(); //aris was not
}


void DragDropCtrl::DropRect(const Rect& rc, dword keyflags)
{
 // no-op, should be implemented in derived class
}

void DragDropCtrl::DragShow(bool _show)
{
 if(_show && state == HIDDEN) {
  Drag(start, Null, last, last_key);
  state = ON;
 }
 if(!_show && state == ON) {
  Drag(start, last, Null, last_key);
  state = HIDDEN;
 }

}


void DragDropCtrl::LeftDouble(Point pt, dword keyflags)
{
 SetWantFocus();
 Click(pt, keyflags | DBLCLK);
}

//
void DragDropCtrl::DragStop(bool accept, dword keyflags)
{
 ReleaseCapture();
 DragHide();
 if(state == HIDDEN && accept)
  Drop(start, last, last_key = keyflags);
 else if(state == POSSIBLE && accept)
  Click(start, last_key = keyflags);
 state = OFF;
// SetLabel("dragstop "+AsString(start)+" "+AsString(last));
}

//should we call overriden???
void DragDropCtrl::LeftUp(Point pt, dword keyflags)
```

```
{
 LLOG("DragDropCtrl::LeftUp -> " << pt);
 DragStop(true, keyflags);
}

void DragDropCtrl::MouseMove(Point pt, dword keyflags)
{
 LLOG("DragDropCtrl::MouseMove -> " << pt);
 if(keyflags != last_key)
  DragHide();
 if(state == POSSIBLE && IsDragDistance(pt, start))
 {
//  SetLabel("possible "+AsString(start)+" "+AsString(last));

  state = ON;
  Drag(start, Null, last = pt, last_key = keyflags);
 }
 else if(state == ON || state == HIDDEN)
 {
// SetLabel("ON "+AsString(start)+" "+AsString(last));

  Point plast = (state == ON ? last : Point(Null));
  last = pt;
  last_key = keyflags;
  state = ON;
  Drag(start, plast, pt, last_key);

 }
}

//not important
bool DragDropCtrl::Key(dword key, int repcnt)
{
 if(key == K_ESCAPE)
 {
  DragStop(false);
  return true;
 }
 return Ctrl::Key(key, repcnt);
}
//========================================

class App : public TopWindow {
 DragDropCtrl dr;
public:
    typedef App CLASSNAME;
    App();
};
```

```
App::App() {
 dr.SetRect(150,100,300,200);
    Add(dr);
// dr.Color(SRed());
 //   btn1.WhenAction = THISBACK(testback);

    Sizeable().Zoomable();
    Title("DragDropCtrl");
}


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