
Subject: Painter: writes to free blocks detected

Posted by [ManfredHerr](#) on Mon, 19 May 2014 14:31:35 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

when using painter - it's path feature - my application crashes deep in the internals of Painter with the error message given above. I reduced the code as much as possible and have now two functions generating a simple painting: ShowSymbolfulldh and ShowSymbolfulldh1. The first generates a painting that can be drawn without problems. The second gives the identical instructions to PaintingPainter: 1. Move(point) 2. Cubic(point1, point2, point) and 3. Cubic(point1, point2, point). But when drawing the painting the error happens. Here is the code of the CtrlLib - application (wfbd.h and main.cpp):

```
#ifndef _wfbd_wfbd_h
#define _wfbd_wfbd_h

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

using namespace Upp;

#define STATUS_CTRL 0X0002
#define STATUS_MOVETO 0X0100
#define STATUS_LINETO 0X0200
#define STATUS_QUADRB 0X0400
#define STATUS_CUBICB 0X0800
#define STATUS_CONNECT 0X0F00

struct Grapoint: Moveable<Grapoint>{
    int status;
    Pointf p;

    void Serialize(Stream& s)
    {
        s % status % p;
    }

    String ToString() const
    {
        String s;
        s << "status = " << status << ", Point = " << p;
        return s;
    }
    void Xmlize(XmlIO& xml)
    {
        xml
```

```

    ("Status", status)
    ("Point",p)
    ;
}
};

```

```

#define LAYOUTFILE <wfbd/wfbd.lay>
#include <CtrlCore/lay.h>

```

```

class wfbd : public WithwfbdLayout<TopWindow> {
public:
    typedef wfbd CLASSNAME;
    wfbd();
    void Paint(Draw& w);
    Painting ShowSymbolfullhd(Size s,double x,double y);
    Painting ShowSymbolfullhd1(Size s,double x,double y);

    bool symbols_loaded;
};

```

```

#endif

```

```

and

```

```

#include "wfbd.h"

```

```

wfbd::wfbd()
{
    CtrlLayout(*this, "Window title");
}

```

```

void wfbd::Paint(Draw& w)
{
    Size sz = GetSize();
    w.DrawRect(0,0,sz.cx,sz.cy,White());
    if (symbols_loaded)
    {
        Painting paint;
        // paint = ShowSymbol(sz,100,100,"fullhd");
        paint = ShowSymbolfullhd(sz,100,100);
        StoreAsXMLFile(paint,"ShowSymbolfullhd","ShowSymbolfullhd.txt");
        w.DrawPainting(0,0,sz.cx,sz.cy,paint);
        paint = ShowSymbolfullhd1(sz,100,100);
        StoreAsXMLFile(paint,"ShowSymbolfullhd1","ShowSymbolfullhd1.txt");
        w.DrawPainting(0,0,sz.cx,sz.cy,paint);
    }
}

```

```

Painting wfbd::ShowSymbolfullhd(Size s, double x,double y)
{
  PaintingPainter pn(s);
  Pointf p1,p2,pto;

  pn.Translate(x,y);
  pto = Pointf(1.87352941176471, 41.5257352941176);
  pn.Move(pto);
  p1 = Pointf(0.48142640747286, 39.0895550366069);
  p2 = Pointf(8.31200580661449, 34.3912073971219);
  pto = Pointf(9.87812168644282, 37.0014005301691);
  pn.Cubic(p1,p2,pto);
  p1 = Pointf(11.7922633173441, 39.7856065387528);
  p2 = Pointf(3.26563241605655, 44.3099413027013);
  pto = Pointf(1.87352941176471, 41.5257352941176);
  pn.Cubic(p1,p2,pto);
  pn.Fill(Gray());
  pn.Stroke(1,Black());

  return pn.GetResult();
}

```

```

Painting wfbd::ShowSymbolfullhd1(Size s, double x,double y)
{
  Grapoint g;
  Vector<Grapoint> extgrap;
  g.status = 272;
  g.p = Pointf(1.87352941176471, 41.5257352941176);
  extgrap.Add(g);
  g.status = 2 ;
  g.p = Pointf(0.48142640747286, 39.0895550366069);
  extgrap.Add(g);
  g.status = 2;
  g.p = Pointf(8.31200580661449, 34.3912073971219);
  extgrap.Add(g);
  g.status = 2080;
  g.p = Pointf(9.87812168644282, 37.0014005301691);
  extgrap.Add(g);
  g.status = 2;
  g.p = Pointf(11.7922633173441, 39.7856065387528);
  extgrap.Add(g);
  g.status = 2;
  g.p = Pointf(3.26563241605655, 44.3099413027013);
  extgrap.Add(g);
  g.status = 2048;
  g.p = Pointf(1.87352941176471, 41.5257352941176);
}

```

```

extgrap.Add(g);
PaintingPainter pn(s);
double magnify = 1.0;
int n1 = extgrap.GetCount();
int l = 0;
int curstatus =0;
Pointf p1,p2,pto;
bool movewas = false;
pn.Translate(x,y);
for (int i = 0; i < n1 ; i++)
{
DLOG("Point "<<i <<" status " << extgrap[i].status << " : " << magnify * extgrap[i].p);
if (extgrap[i].status & STATUS_CTRL)
{
if (l == 0) p1 = magnify * extgrap[i].p;
else p2 = magnify * extgrap[i].p;
l++;
}
else
{
pto = magnify * extgrap[i].p;
curstatus = extgrap[i].status & STATUS_CONNECT;
switch (curstatus)
{
case STATUS_MOVETO:
DLOG(" pn.Move " <<pto);
pn.Move(pto);
movewas = true;
break;
case STATUS_LINETO:
if (!movewas)
{
DLOG(" pn.Move " <<pto);
pn.Move(pto);
movewas = true;
}
else
{
DLOG(" pn.Line " <<pto);
pn.Line(pto);
}
break;
case STATUS_QUADRB:
if (!movewas)
{
DLOG(" pn.Move " <<pto);
pn.Move(pto);
movewas = true;
}
}
}
}

```

```

    break;
}
if (l)
{
    DLOG(" pn.Quadratic "<<p1<<" , " <<pto);
    pn.Quadratic(p1,pto);
}
else
{
    DLOG(" pn.Quadratic " <<pto);
    pn.Quadratic(pto);
}
break;
case STATUS_CUBICB:
if (!movewas)
{
    DLOG(" pn.Move " <<pto);
    pn.Move(pto);
    movewas = true;
    break;
}
if (l>1)
{
    DLOG(" pn.Cubic " <<p1<<" , " <<p2 <<" , " <<pto);
    pn.Cubic(p1,p2,pto);
}
else if (l>0)
{
    DLOG(" pn.Cubic " <<p1<<" , " <<pto);
    pn.Cubic(p1,pto);
}
else
{
    DLOG(" pn.Move " <<pto);
    pn.Move(pto);
    movewas = true;
}
break;
default:
    DLOG(" pn.Move " <<pto);
    pn.Move(pto);
    movewas = true;
}
l=0;
}
pn.Fill(Gray());
pn.Stroke(1,Black());

```

```
}  
return pn.GetResult();  
}
```

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

The two files generated by the Paint function exhibit that the paintings are different indeed. If I use a separate program to read and display a painting (xmlized) then I see the same behaviour: One can be displayed without problems, the other generates the error Writes to free blocks detected.

BTW, I used the Newbie Corner as I didn't find a forum dedicated to Painter.

Thanks for hints and
Regards

Manfred

Subject: Re: Painter: writes to free blocks detected
Posted by [ManfredHerr](#) on Mon, 19 May 2014 14:42:52 GMT
[View Forum Message](#) <> [Reply to Message](#)

I am using LINUX (Ubuntu 12.04LTS) and have a strange effect with GTK. If I specify NOGTK then the window with the error message "writes to free blocks detected" appears instantly. If I use the default, with GTK, the window appears but remains empty. After an asynchronous break I get the following stack trace:

File Attachments

1) [wfbd_gtkpoll.png](#), downloaded 597 times

```
00-poll at <unknown>:<unknown>
```

```
01-?? at <unknown>:<unknown>
```

```
02-g_main_loop_run at <unknown>:<unknown>
```

```
03-gtk_dialog_run at <unknown>:<unknown>
```

```
04-Upp::Ctrl::PanicMsgBox(title,text) at /home/manfred/upp/uppsrc/CtrlCore/GtkApp.cpp:20
```

```
05-Upp::PanicMessageBox(title,text) at /home/manfred/upp/uppsrc/Core/Util.cpp:29
```

```
06-Upp::Panic(msg) at /home/manfred/upp/uppsrc/Core/Util.cpp:99
```

```
07-Upp::HeapPanic(text,pos,size) at /home/manfred/upp/uppsrc/Core/heaputil.cpp:202
```

```
08-Upp::Heap::DbgFreeCheck(p,size) at /home/manfred/upp/uppsrc/Core/heaputil.cpp:221
```

```
09-Upp::Heap::DbgFreeCheckK(p,k) at /home/manfred/upp/uppsrc/Core/heaputil.cpp:230
```

```
10-Upp::Heap::Allok(this,k) at /home/manfred/upp/uppsrc/Core/sheap.cpp:122
```

```
11-Upp::Heap::Alloc(this,sz) at /home/manfred/upp/uppsrc/Core/sheap.cpp:133
```

```
12-Upp::MemoryAlloc_(sz) at /home/manfred/upp/uppsrc/Core/sheap.cpp:316
```

```
13-Upp::MemoryAlloc(size) at /home/manfred/upp/uppsrc/Core/heapdbg.cpp:88
```

```
14-Upp::MemoryAllocSz(size) at /home/manfred/upp/uppsrc/Core/heapdbg.cpp:109
```

```
15-Upp::VectorReAlloc(v_,newalloc,sizeofT) at /home/manfred/upp/uppsrc/Core/Vcont.cpp:35
```

```
00-poll at <unknown>:<unknown>
```

Subject: Re: Painter: writes to free blocks detected

Posted by [ManfredHerr](#) on Tue, 27 May 2014 13:34:10 GMT

[View Forum Message](#) <> [Reply to Message](#)

Solution for those trying to omit learning the hard way:

In the function creating the crashing painting the Fill and Stroke instructions are one closing bracket (}) to high. So the path is continued after it was filled and stroked. But you have to start a new path after these instructions with a move. Best you enclose each path with Begin() and End().