Subject: Re: A terminal emulator widget for U++
Posted by Oblivion on Fri, 24 Apr 2020 22:36:07 GMT
View Forum Message <> Reply to Message

Hello,

I have pushed the the critical performance fix that I mentioned in my last message.

The main culprit was using a Vector where a BiVector was a much better option. In fact, it seems to be the optimal solution to the scrollback/hsitory buffer problem. Removing elements from the beginning of a Vector, especially when its length > 64K is -not surprisingly- very slow. BiVector solves exactly this problem because a scrollback buffer is mutated only from the either end.

There was also a constant display refresh happening when a page was scrolled, even when a single line scrolled, degrading performance. This is fixed too.

Now Terminal ctrl's scroll speed / throughput can be very fast. (It is Even faster than the above given numbers: ~3.5 secoonds on the same test while acting as a both SSH (using Upp/SSH package) and local terminal. ("time find /usr/share" command was outputting > 300.000 file paths in these tests, with the size of scrollback buffer of each terminal was set to 64 k)

However, you can't see the real difference if you use the provided reference example codes, because they have a very rudimentary event loop.

If you are curious and you want to test the scrolling/throughput performance of Terminal ctrl, you can use the below code: While still rudimentary, it is closer to real world scenarios.

```
#include <Terminal/Terminal.h>
#include <Terminal/PtyProcess.h>

using namespace Upp;

const char *nixshell = "/bin/bash";

struct TerminalExample : TopWindow {
 Terminal term;
 PtyProcess pty;

 TerminalExample()
 {
  SetRect(term.GetStdSize());
  Sizeable().Zoomable().CenterScreen().Add(term.SizePos());
  term.WhenBell   = [=]()               { BeepExclamation(); };
  term.WhenTitle  = [=](String s)       { Title(s);          };
  term.WhenOutput = [=](String s)       { pty.Write(s);      };
  term.WhenResize = [=]()               { pty.SetSize(term.GetPageSize()); };
  term.WhenLink   = [=](const String& s) { PromptOK(s);       };
```

```
  term.InlineImages().Hyperlinks().WindowOps().DynamicColors();
  term.SetHistorySize(65536);
 }

 void Run()
 {
  pty.Start(nixshell, Environment(), GetHomeDirectory());
  OpenMain();
  while(IsOpen() && pty.IsRunning()) {
   int ms = 16;
   String in = pty.Get();
   if(!IsNull(in)) {
    term.WriteUtf8(in);
    int len = in.GetLength();
    if(len >= 1024)
     ms = 1024 * 16 / len; // Scale to workload...
   }
   ProcessEvents();
   Sleep(ms);
  }
 }

};

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

There are other improvements and bug fixes too (mostly on serialization/xmlization/jsonization fronts).

If you have any questions, criticism, bug reports, patches, ideas, let me know.

Best regards,
Oblivion