Subject: Painter and transformations
Posted by mdelfede on Tue, 22 Jan 2013 17:29:46 GMT
View Forum Message <> Reply to Message

Hi, usually (in many other geometric tools) this code

```
 painter.Translate(-jg.center);
 if(rot)
  painter.Rotate(-jg.angle);
 painter.Translate(dx, dy);

(do something with painter)

 painter.Translate(-dx, .dy);
 if(rot)
  painter.Rotate(jg.angle);
 painter.Translate(jg.center);
```

shiould behave exactly like this one :

```
 Xform2D trsf;
 trsf = Xform2D::Translation(-jg.center.x, -jg.center.y);
 if(rot)
  trsf = trsf * Xform2D::Rotation(-jg.angle);
 trsf = trsf * Xform2D::Translation(dx, dy);

 // translate painter to fit left joint center
 painter.Transform(trsf);

(do something with painter)

 trsf = Inverse(trsf);
 painter.Transform(trsf);
```

But it doesn't.... and I still don't understand how Upp does manage combined transformations.
By now I solved with second way, but I guess it would be by far more intuitive to fix the first one; each transformation should apply to "modified" reference done by former ones.

Anyways, it would be useful to add some member functions to Xform2d, to make possible to do something like this :

```
 Xform2d trsf = Xform2d().Translated(p).Rotated(a).Scaled(x);
```

Max

Subject: Re: Painter and transformations
Posted by mirek on Wed, 23 Jan 2013 11:37:35 GMT
View Forum Message <> Reply to Message

Tried with this testcase:

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

using namespace Upp;

struct PainterTest : public TopWindow {
 virtual void Paint(Draw& w) {
  DrawPainter p(w, GetSize()/*, MODE_NOAA*/);

  p.Clear(White());
  p.Text(0, 0, "A", Roman(50));
  p.Fill(Black());

  p.Translate(100, 100);
  p.Rotate(1);
  p.Fill(Blue());

  p.Translate(200, -200);
  p.Scale(6);
  p.Fill(Magenta());

  p.Scale(1.0 / 6);
  p.Translate(-200, 200);

  p.Rotate(-1);
  p.Fill(Red());

  p.Translate(-100, -100);
  p.Fill(Green());
 }
};

GUI_APP_MAIN
{
 PainterTest().Run();
```

}

All seems ok...

Subject: Re: Painter and transformations
Posted by mdelfede on Wed, 23 Jan 2013 23:23:57 GMT
View Forum Message <> Reply to Message

Hi Mirek, I'll append a sample test case which do the transformation in 2 ways, one with discrete commands (Translate, Rotate) and the other with a transformation matrix.
They shows the same drawing done in both ways in 2 controls aside; they should be identical (imho...) but on first one the image is not there because rotation shift out of control.

The circle is just to show image center point; you can notice that the line is not on left picture, but it's displayed correctly on right one.

If I remove the rotation (in both paths), I get the correct image for boths, as following :

It seems to me that rotation using discrete 'painter.Rotate()' command does the rotation around a wrong point... or am I missing something ?

Max

File Attachments

1) Transformation.png, downloaded 588 times
2) Transformations.zip, downloaded 362 times
3) Transform2.png, downloaded 571 times

Subject: Re: Painter and transformations
Posted by mirek on Thu, 24 Jan 2013 07:07:22 GMT
View Forum Message <> Reply to Message

You are not doing transformations right:

```
Xform2D trsf;
trsf = Xform2D::Translation(-center.x, -center.y);
trsf = trsf * Xform2D::Rotation(angle);
```

```
 trsf = trsf * Xform2D::Translation(sz.cx / 2, sz.cy / 2);
```

You have reversed order of operands there, that is why you are getting different results.

Please refer to

http://en.wikipedia.org/wiki/Transformation_matrix#Composing _and_inverting_transformations

I have spend some time with your example to find out what is going on and it seems to me that everything behaves as expected (apart from the fact that the whole thing is perhaps not doing what you expect

```
void Transformations::PaintImage(PaintingPainter &painter, Color color)
{
 Size sz = imageCtrl.GetSize();
 painter.Rectangle(0, 0, sz.cx, sz.cy);
 painter.Stroke(10, color);

 painter.Circle(0, 0, 50);
 painter.Fill(color);

 Pointf center(2000, 2000);
 painter.Text(2000, 2000, "A", Roman(400));
 painter.Fill(color);
}

void Transformations::Paint1()
{
 Size sz = imageCtrl.GetSize();
 PaintingPainter painter(sz);

 painter.Clear(White());

 painter.Scale(0.1, 0.1);
 painter.Translate(3000, 3000);


 painter.Scale(1, -1);
 painter.Translate(0, -sz.cy);

 PaintImage(painter, Black());

 Pointf center(2000, 2000);
 double angle = -45.0 * M_PI / 180;

 painter.Translate(-center);
```

```
 PaintImage(painter, Red());
 painter.Rotate(-angle);
 PaintImage(painter, Green());
 painter.Translate(sz.cx / 2, sz.cy / 2);
 PaintImage(painter, Blue());

 ImageDraw dw(sz);
 dw.DrawPainting(0, 0, sz.cx, sz.cy, painter);
 imageCtrl.SetImage(dw);
}
```

Subject: Re: Painter and transformations
Posted by mdelfede on Thu, 24 Jan 2013 08:17:27 GMT
View Forum Message <> Reply to Message

Well, the purpose is :

1- Move te origin to 'center' point, aka have point (0,0) at 'center' point

3- Move reference again to be on ImageCtrl center

4- Draw stuffs

5- Undo the transformation

That should (IMHO...) be done with

Translate(-center).Rotate(-angle).Translate(imagecenter)

So, maybe I understand Painter transformations wrong, but I still don't know how to have the thing to what I want
Even more, if I remove the Rotate part it behaves as expected, so I guess that the problem (mine or Painter's one) it's the rotation center point.

BTW, you're right, I'm reversing the product's order in transformation matrices, but the weird is that if I do it right it doesn't work either....

Max

Subject: Re: Painter and transformations
Posted by mdelfede on Thu, 24 Jan 2013 09:06:39 GMT
View Forum Message <> Reply to Message

Well, thanks to Mirek I got te point :

the last translation (used to center drawing on control) happens on a rotated reference, so it's wrong :

```
painter.Translate(-center);
painter.Rotate(angle);
painter.Translate(sz.cx / 2, sz.cy / 2); <--WRONG
```

The weird part is that matrix way worked because it is reversed too, as Mirek explained correctly; so, to achieve the purpose it's enough to reverse transformation order (explanation is quite complex, but it works) :

```
painter.Translate(sz.cx / 2, sz.cy / 2); <--WRONG
painter.Rotate(angle);
painter.Translate(-center);
```

This one does exactly what the matrix version do.

Max