

Hi,

I am seeing what appear to be errors in the way that upp under linux handles:

=> Alpha()

=> DrawPolyPolygon()

The alpha function appears to make everything transparent all the time.

The DrawPolyPolygon function appears to not split up the constituent polygons correctly.

Here is my polygon layer drawing function in case I'm doing something wrong but all this code works very nicely in Windows XP.

Nick

```
void PolygonLayer::DrawLayer( Draw& w, // info regarding the screen or drawing context
                             ImageBuffer& ibBk, // image that should be updated by each layer
                             void* pView // pointer to the main top window object
                             )
{
    OpenWind* ptr = (OpenWind*)pView;

    Rect rc = ptr->MPtoLP(m_rc);

    int alpha = (int)(255.0*m_fAlpha + 0.5);

    //check to see if we need to draw at all
    if(ptr->GetClientRect().Intersects(rc))
    {
        if(ptr->DrawAsBackground(this))
        {

            Size sz = ptr->GetClientRect().Size();
            ImageDraw id(sz);
            id.DrawRect(sz,White);
            id.Alpha().DrawRect(sz,Black);

            // every time we draw this we need to convert from MP to LP so its ok to convert from
            // a friendly data structure to a not so friendly one at the same time

            Point* vertices = new Point[GetTotalPoints()];
            int* subpolygon_counts = new int[10000];
            Point pt;
```

```

Color colFill = m_bFill ? m_colFill : Null;
Color colOutline = m_bLine ? m_colLine : Null;
Font font(m_fontLabel.GetFace(),(int)(m_fontLabel.GetHeight()*ptr->GetTextMultiplier()));

for(int k=0;k<m_polys.GetCount();k++)
{
    rc = ptr->MPtoLP(m_polys[k].m_rc);
    if(!rc.Intersects(ptr->GetClientRect()))
        continue; // i.e. dont draw this object - see if next one is in view

    int vertex_count = GetTotalPoints(k);
    int subpolygon_count_count = m_polys[k].m_loops.GetCount();
    int nWidth = m_nWidth;
    uint64 pattern = 0;
    int c=0;

    for(int i=0;i<subpolygon_count_count;i++)
    {
        const int num = m_polys[k].m_loops[i].GetCount();
        subpolygon_counts[i] = num;
        for(int j=0;j<num;j++)
        {
            vertices[c++] = ptr->MPtoLP(m_polys[k].m_loops[i][j]);
        }
    }

    // draw
    id.DrawPolyPolygon(vertices, vertex_count, subpolygon_counts,
        subpolygon_count_count, colFill, nWidth, colOutline, pattern);

    id.Alpha().DrawPolyPolygon(vertices, vertex_count, subpolygon_counts,
        subpolygon_count_count, Color((m_bFill ? alpha : 0),0,0), nWidth,
        Color((m_bLine ? alpha : 0),0,0), pattern);

    if(m_bLabel)
    {
        pt = ptr->MPtoLP(m_polys[k].m_ptCOG);
        String s;
        if(m_bLabelField)
        {
            s = CUtils::EatSpace(m_polys[k].m_valAt[m_nLabelField].ToString());
        }
        else // use object index + 1
        {
            s = Format("%d",k+1);
        }
        Size tsz = GetTextSize(s,font);
        id.DrawText(pt.x-tsz.cx/2,pt.y-tsz.cy/2,s,font,m_colLabel);
    }
}

```

```

        id.Alpha().DrawText(pt.x-tsz.cx/2,pt.y-tsz.cy/2,s,font,Color(255,0,0));
    }
}

delete[] vertices;
delete[] subpolygon_counts;

Image ib(id);
ImageDraw id2(sz);
id2.DrawImage(0,0,sz.cx,sz.cy,ibBk);
id2.DrawImage(0,0,ib);
Image ib2(id2);
ibBk = ib2;
}
}

// always call the base class last
Layer::DrawLayer(w,ibBk,ptr);
}

```