OpenGraph Library (SDK)

Quick Guide

Computations & Graphics, Inc.
Highlands Ranch, CO 80130, USA
Email: info@cg-inc.com
Web: www.cg-inc.com

OpenGraph Library

Computations & Graphics, Inc. (CGI) OpenGraph Library is a powerful 2D and 3D visualization and charting software tool. It is built on the industrial strength OpenGL and makes it easy for ordinary software programmer to add impressive and interactive 2D or 3D graphics in their applications without the need to learn the complex OpenGL API. This library is especially suitable for use in building scientific, engineering and financial software applications on Windows platform.

The following are the main features:

- Capability to draw different objects such as points, lines, triangles and quads.
- Capability to draw texts including Unicode texts.
- Automatic support for zooming, panning, rotating 3D models using mouse events.
- Support graphic selections on drawing objects.
- Support both 3D objects as well as 2D screen objects.
- Support transparency.
- Supports both native C++ language
- Support .NET 4.x, 4.5x, 4.6x, 4.7x, and 4.8 using as C# and VB.NET.
- Supports Unicode and Non-Unicode.
- Supports x64 and x86 platform architectures.
- Supports Visual Studio 2015 and up including Visual Studio 2022
- Ability to save rendered image to a file
- Available in both binary library and source code form.
- Very easy to learn. The library comes with a variety of examples illustrating the uses of the library including Win32 Windows, MFC Dialog, MFC Document-View, WinForms, and WPF Form.
- Reasonably priced and royalty-free on binary redistribution.

C++ Interface

The C++ library comes with a native Windows DLLs called OpenGraphLib.dll and a library file called OpenGraphLib.lib for linking to your projects. The interface includes the following header files: OpenGraphLib.h, OpenGraphDefines.h and OpenGraph.h.

Different versions of OpenGraphLib.lib and OpenGraphLib.dll are provided for your configuration needs on x64 and x86 CPU platforms in Unicode and non-Unicode. You should link your projects to the correct version of the lib file. Make sure you also copy the correct version of the DLL to your executable directory.

The OpenGraphLib.h is a standard DLL include header which makes exporting from a DLL simpler.

```
#ifdef OPENGRAPHLIB_EXPORTS
#define OPENGRAPHLIB_API __declspec(dllexport)
#else
#define OPENGRAPHLIB_API __declspec(dllimport)
#endif
```

The OpenGraphDefines.h defines a few input and output data structures. The following are two examples.

```
struct OPENGRAPHLIB_API CPointf
{
    CPointf(double x = 0, double y = 0, double z = 0) {
        xyz[X] = x;
        xyz[Y] = y;
        xyz[Z] = z;
    }
    double xyz[3];

};

const int MAX_UID_SIZE = 20;
struct OPENGRAPHLIB_API CIdentity
{
    CIdentity()
    {
        uid[0] = _T('\0');
        id = 0;
    }

    TCHAR uid[MAX_UID_SIZE];
    int id;
```

```
};
```

The OpenGraph.h is the main header that defines the library interface.

```
class OPENGRAPHLIB_API COpenGraph
{
public:
    COpenGraph(void);
    virtual ~COpenGraph(void);


public:
    enum {modeNONE = -1,modeTRACK, modeSELECT, modeZOOM, modeREALPAN, modeREALZOOM, modeROTATE };

private:
    enum {GRID_LIST = 0, NODE_LIST, LINE_LIST, TRIANGLE_LIST, QUAD_LIST, TEXT_LIST,
          WINDOW_NODE_LIST, WINDOW_LINE_LIST, WINDOW_TRIANGLE_LIST, WINDOW_QUAD_LIST,
                WINDOW_TEXT_LIST, END_LIST};

public:
    bool Initialize(HWND hWnd);
    void SetMode(int mode);
    void SetWindowSize(int cx, int cy);
    void SetBackgroundColor(COLORREF bgColor);
    void Render();
    void ForceRender();

    BOOL OnLButtonDown(RECT& rect, UINT nFlags, POINT point);
    void OnMouseMove(UINT nFlags, POINT point);
    void ZoomMouseWheel(UINT nFlags, short zDelta, POINT point);
    HCURSOR GetModeCursor();
    void OnSetCursor();
    bool SaveImage(LPCTSTR fileName, bool bUseWhiteBackground);

    void Pan(double fTx, double fTy);
    void Rotate(double fRx, double fRy, double fRz);
    void ZoomExtentView();
    void SetFrontView();
    void SetBackView();
    void SetLeftView();
    void SetRightView();
    void SetTopView();
    void SetBottomView();
    void SetIsometricView();

    bool TrackRect(HWND hWnd, POINT point);
    RECT GetTrackRect()const; // un-normalized
    CPointf GetProjectedPoint(CPointf pt);

    void Clear();
```

```
void SelectAll();
void UnSelectAll();
void ReverseSelectAll();

// note: client is responsible to delete pIdentity pointer in the following functions
void GetSelected(CIdentity*& pIdentity, int& nCount);
void GetSelectedNodes(CIdentity*& pIdentity, int& nNodes);
void GetSelectedLines(CIdentity*& pIdentity, int& nLines);
void GetSelectedTriangles(CIdentity*& pIdentity, int& nTriangles);
void GetSelectedQuads(CIdentity*& pIdentity, int& nQuads);
void GetSelectedWindowNodes(CIdentity*& pIdentity, int& nNodes);
void GetSelectedWindowLines(CIdentity*& pIdentity, int& nLines);
void GetSelectedWindowTriangles(CIdentity*& pIdentity, int& nTriangles);
void GetSelectedWindowQuads(CIdentity*& pIdentity, int& nQuads);

CIdentity AddNode(int id,CPointf pt, int nSize, COLORREF color,COLORREF colorSelected,
                          int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0);
void AddNodes(CIdentity* pIdentity, int* pId, CPointf* pPt, int nNodes, int nSize, COLORREF color,COLORREF colorSelected,
                  int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
CIdentity AddLine(int id,CPointf pt1,CPointf pt2, int nThickness, COLORREF color,COLORREF colorSelected,
                          int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
void AddLines(CIdentity* pIdentity, int* pId, CPointf* pPt1,CPointf* pPt2, int nLines, int nThickness, COLORREF color,COLORREF colorSelected,
                          int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
CIdentity AddTriangle(int id,CPointf pt1,CPointf pt2,CPointf pt3, COLORREF color,COLORREF colorSelected, float alpha=1.0f,
                          int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
void AddTriangles(CIdentity* pIdentity,int* pId, CPointf* pPt1,CPointf* pPt2, CPointf* pPt3, int nTriangles, COLORREF color,COLORREF
                  colorSelected,float alpha = 1.0f, int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
CIdentity AddQuad(int id,CPointf pt1,CPointf pt2,CPointf pt3,CPointf pt4, COLORREF color,COLORREF olorSelected, float alpha=1.0f,
                          int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
void AddQuads(CIdentity* pIdentity,int* pId, CPointf* pPt1,CPointf* pPt2, CPointf* pPt3,CPointf* pPt4, int nQuads, COLORREF color,
                  COLORREF colorSelected,float alpha = 1.0f, int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
int AddFont(const TCHAR *typeface, int height, int weight = 0, DWORD italic = 0);
CIdentity AddText(int id, int nFontId, CPointf pt, LPCTSTR text, COLORREF color,COLORREF colorSelected,
                  int nAlignment = ALIGN_BOTTOM|ALIGN_LEFT, int nStatus = STATUS_UNSELECTED);


CIdentity AddWindowNode(int id,CPointf pt, int nSize, COLORREF color,COLORREF colorSelected, int nStatus = STATUS_UNSELECTED,
                          LPCTSTR tag = 0, LPCTSTR userData = 0);
void AddWindowNodes(CIdentity* pIdentity, int* pId, CPointf* pPt, int nNodes, int nSize, COLORREF color, COLORREF colorSelected,
                          int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
CIdentity AddWindowLine(int id,CPointf pt1,CPointf pt2, int nThickness, COLORREF color,COLORREF colorSelected,
                                  int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
void AddWindowLines(CIdentity* pIdentity, int* pId, CPointf* pPt1,CPointf* pPt2, int nLines, int nThickness, COLORREF color,
                          COLORREF colorSelected, int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
CIdentity AddWindowTriangle(int id,CPointf pt1,CPointf pt2,CPointf pt3, COLORREF color,COLORREF colorSelected, float alpha=1.0f,
                                  int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
void AddWindowTriangles(CIdentity* pIdentity,int* pId, CPointf* pPt1,CPointf* pPt2, CPointf* pPt3, int nTriangles, COLORREF color,
                          COLORREF colorSelected,float alpha = 1.0f, int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
CIdentity AddWindowQuad(int id,CPointf pt1,CPointf pt2,CPointf pt3,CPointf pt4, COLORREF color, COLORREF colorSelected, float alpha=1.0f,
                          int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
```

```
    void AddWindowQuads(CIdentity* pIdentity,int* pId, CPointf* pPt1,CPointf* pPt2, CPointf* pPt3,CPointf* pPt4, int nQuads,
                            COLORREF color,COLORREF colorSelected,float alpha = 1.0f,
                            int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0 );
    CIdentity AddWindowText(int id, int nFontId, CPointf pt, LPCTSTR text, COLORREF color,COLORREF colorSelected,
                            int nAlignment = ALIGN_BOTTOM|ALIGN_LEFT, int nStatus = STATUS_UNSELECTED);

};
```

As you can see, the interface is pretty lean. It does not require you to have deep knowledge about 3D graphics programming. The interface allows you to draw 2D and 3D objects such as points (also called nodes in the library), lines, triangles, quads and texts. The main difference between 2D and 3D objects is that 2D objects will have their coordinates in Windows pixel and their positions will not be affected by zooming, panning or rotating. Each object except texts may be assigned two different colors for selected and unselected states. All objects except texts can be selected using mouse to the end user feedback, thus making this library applicable and useful in many fields.

The following are the different drawing modes.

modeNONE: mouse movement will not trigger any action.

modeTRACK: a rubber-band rectangle will be drawing as mouse is pressed down and moved.

modeSELECT: objects will be selected or unselected by mouse clicking or mouse dragging.

modeZOOM: window zooming occurs by mouse dragging

modeREALPAN: objects will be panned by mouse moving.

modeREALZOOM: not used for now

modeROTATE: objects will be rotated by mouse dragging.

You can set the view position by calling Pan() and Rotate() functions.

```
    void Pan(double fTx, double fTy);
    void Rotate(double fRx, double fRy, double fRz);
```

Several preset view positions are available in the following functions.

```
     void SetFrontView();
    void SetBackView();
    void SetLeftView();
    void SetRightView();
    void SetTopView();
    void SetBottomView();
     void SetIsometricView();
```

The ZoomExtentView() will reset the view so all objects can be seen in the window.

3D Graphic objects may be added by calling the proper Add functions. For example, call AddNode() to add a 3D point (or node).

```
CIdentity AddNode(int id,CPointf pt, int nSize, COLORREF color,COLORREF colorSelected,
                        int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0);
```

The status can be STATUS_UNSELECTED or STATUS_SELECTED as defined in the following:

```
const int STATUS_UNSELECTED = 0x00000000L;
const int STATUS_SELECTED = 0x00000001L;
```

An identity as defined below is returned for each object being added. The uid is unique id that the library generated for each object. The id is the one that user-defined identifier that may or may not be unique.

```
const int MAX_UID_SIZE = 20;
struct OPENGRAPHLIB_API CIdentity
{
    TCHAR uid[MAX_UID_SIZE];
    int id;
};
```

Each object may have two colors associated: one for unselected state and one for selected state. Currently, the library does not make use of tag and userData passed in but may make use of them in the future.

For triangles and quads, an alpha value between 0-1 may be passed in, which defines the opacity of the objects. For completely opaque object, use alpha value 1.0. For completely transparent object, use alpha value of 0.0.

2D Graphic objects may be added by calling the proper AddWindow functions. 2D objects are defined by their pixel positions on the screen and are not affected by zooming, panning or rotating. For example, call AddWindowNode() to add a 2D point (or node).

```
CIdentity AddWindowNode(int id,CPointf pt, int nSize, COLORREF color,COLORREF colorSelected,
                                int nStatus = STATUS_UNSELECTED, LPCTSTR tag = 0, LPCTSTR userData = 0);
```

You may retrieve all selected objects by calling the following function.

```
void GetSelected(CIdentity*& pIdentity, int& nCount);
```

You may distinguish different objects by inspecting the uid prefixes:

N: for 3D nodes
L: for 3D lines
TRI: for 3D triangles
QUAD: for 3D quads
WN: for 2D nodes
WL: for 2D lines
WTRI: for 2D triangles
WQUAD: for 2D quads

All objects will be erased by calling the Clear() function. Rendered image may be saved to a file by calling SaveImage() function.

The best way to learn how you can use these data structures and interface functions is to study the examples that come with the library. OpenGraphWin32Client is a general Win32 Windows application in which the main window is the graphics window. OpenGraphDialogClient is a MFC dialog-based application that uses a control on the dialog as the graphics window. OpenGraphMFCClient is a MFC document-view application in which the view displays the graphics.

The following is a general guide lines to setup the library:

1. Bring the interface to your code

```
#include "OpenGraph.h"
using namespace NSOpenGraphLib;
```

2. Instantiate the COpenGraph object.

```
COpenGraph m_openGraph;
```

3. Initialize the COpenGraph object right after the window (on which graphics will be displayed) is created.

```
m_openGraph.Initialize(hWnd);
```

4. Handle WM_ERASEBKGND Windows message properly so OpenGL will take care of the drawing. This will prevent the flickering of graphics when the window is being resized. For MFC application, it is as simple as return TRUE in OnEraseBkgnd() handler.

```
BOOL COpenGraphMFCClientView::OnEraseBkgnd(CDC* pDC)
{
    return TRUE;    // this will prevent flickering of the window when resizing
}
```

5. Handle WM_SIZE Windows message so we can setup the graphics window size. It is also a good idea to remember the window size, especially if we want to draw 2D objects on screen. For example:

```
void COpenGraphMFCClientView::OnSize(UINT nType, int cx, int cy)
{
    m_nCx = cx;
    m_nCy = cy;
    m_openGraph.SetWindowSize(cx, cy);
}
```

6. Handle WM_LBUTTONDOWN and WM_MOUSEMOVE windows message so we can handle different graphics modes such as selection, zooming, panning or rotating automatically.

```
void COpenGraphMFCClientView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CRect rect;
    m_openGraph.OnLButtonDown(rect, nFlags, point);
}

void COpenGraphMFCClientView::OnMouseMove(UINT nFlags, CPoint point)
{
    m_openGraph.OnMouseMove(nFlags, point);
}
```

7. Handle the WM_MOUSEWHEEL windows message if we want automatic zooming with mouse wheeling

```
BOOL COpenGraphMFCClientView::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
{
    m_openGraph.ZoomMouseWheel(nFlags, zDelta, pt);
    return CView::OnMouseWheel(nFlags, zDelta, pt);
}
```

8. Handle cursor changes for different graphics mode. For MFC view, call the interface function OnSetCursor() if we want the library to handle cursor changes automatically as shown below:

```
BOOL COpenGraphMFCClientView::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
    m_openGraph.OnSetCursor();
    return TRUE;
}
```

In Win32 Windows application, we may do the similar in MouseMove event handler like the following:

```
case WM_MOUSEMOVE:
{
    m_openGraph.OnSetCursor();
}
 break;
```

9. Add various graphics objects by calling appropriate Add functions

```
m_openGraph.AddLine(99999, CPointf(-2, 0, 0), CPointf(2, 0, 0), 15, RGB(255, 255, 0), RGB(0, 255, 255),
                                0, NULL, 0);
m_openGraph.AddText(99999, 0, (CPointf(-2, 0, 0) + CPointf(2, 0, 0)) / 2.0, _T("99999"), RGB(125, 123, 0),

                                    RGB(0, 123, 0));
```

10. Render the graphics in WM_PAINT windows message handler. For example:

```
void COpenGraphMFCClientView::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    m_openGraph.Render();
}
```

Please note if the graphics window is a control on the dialog, we generally need to have a custom control to handle the appropriate Windows messages mentioned above. You can refer to the class COpenGLWnd class in the OpenGraphDialogClient demo project for details.

.NET Interface

The .NET interface is a managed class assembly DLL called OpenGraphLib_Cli.dll. It is written in C++/CLI and makes calls to the native Windows DLL called OpenGraphLib.dll (refer to C++ Interface for details). The actual procedure to use this .NET interface is identical to C++ interface, except syntax differences. Different versions of OpenGraphLib_Cli.dll are provided for your configuration needs in x64 and x86 CPU platforms for .NET version 4.0, 4.5x, 4.5x, 4.6x, 4.7x, and 4.8. Make sure your project has reference to the correct version of OpenGraphLib_Cli.dll. Also make sure a copy of the corresponding native DLL OpenGraphLib.dll is placed in the executable directory.

Interface input and output data structures are reference types in NSOpenGraphLib_Cli namespace. For example:

```
public class CColor_Cli
{
    public byte R;
    public byte G;
    public byte B;

    public CColor_Cli(byte r, byte g, byte b)
    {
        this.R = r;
        this.G = g;
        this.B = b;
    }

    public CColor_Cli()
    {
        this.B = (byte)0;
        this.G = (byte)0;
        this.R = (byte)0;
    }

    public static void Copy(CColor_Cli colorDest, CColor_Cli colorOrg)
    {
    }
}

public enum CDirection
{
    X,
    Y,
    Z,
    OX,
    OY,
    OZ,
}
```

```
public class CIdentity_Cli
{
    public string uid;
    public int id;

    public CIdentity_Cli()
    {
        this.id = 0;
        this.uid = (string)null;
    }

    public static void Copy(CIdentity_Cli idDest, CIdentity_Cli idOrg)
    {
        idDest.uid = idOrg.uid;
        idDest.id = idOrg.id;
    }
}

public class CPointf_Cli
{
    public double X;
    public double Y;
    public double Z;

    public CPointf_Cli(double x, double y, double z)
    {
        this.X = x;
        this.Y = y;
        this.Z = z;
    }

    public CPointf_Cli()
    {
        this.Z = 0.0;
        this.Y = 0.0;
        this.X = 0.0;
    }

    public static void Copy(CPointf_Cli ptDest, CPointf_Cli ptOrg)
    {
        ptDest.X = ptOrg.X;
        ptDest.Y = ptOrg.Y;
        ptDest.Z = ptOrg.Z;
    }
}

public class CPoint_Cli
{
    public int X;
    public int Y;
```

```
    public CPoint_Cli(int x, int y)
    {
        this.X = x;
        this.Y = y;
    }

    public CPoint_Cli()
    {
        this.Y = 0;
        this.X = 0;
    }

    public static void Copy(CPoint_Cli ptDest, CPoint_Cli ptOrg)
    {
        ptDest.X = ptOrg.X;
        ptDest.Y = ptOrg.Y;
    }
}

public class CRect_Cli
{
    public int left;
    public int right;
    public int top;
    public int bottom;

    public CRect_Cli(int l, int r, int t, int b)
    {
        this.left = l;
        this.right = r;
        this.top = t;
        this.bottom = b;
    }

    public CRect_Cli()
    {
        this.bottom = 0;
        this.top = 0;
        this.right = 0;
        this.left = 0;
    }

    public static void Copy(CRect_Cli rectDest, CRect_Cli rectOrg)
    {
        rectDest.left = rectOrg.left;
        rectDest.right = rectOrg.right;
        rectDest.top = rectOrg.top;
        rectDest.bottom = rectOrg.bottom;
    }
}
```

```
public class CStatus_Cli
{
    public static int STATUS_UNSELECTED;
    public static int STATUS_SELECTED;
    public static int STATUS_FREEZED = 2;

    static CStatus_Cli()
    {
        CStatus_Cli.STATUS_SELECTED = 1;
        CStatus_Cli.STATUS_UNSELECTED = 0;
    }
}

public class CTextAlignment_Cli
{
    public static int ALIGN_LEFT;
    public static int ALIGN_CENTER;
    public static int ALIGN_RIGHT;
    public static int ALIGN_TOP;
    public static int ALIGN_BOTTOM;
    public static int ABOVE;
    public static int BELOW;
    public static int ALIGN_TOP2;
    public static int BELOW2;
    public static int BELOW3;
    public static int BELOW4 = 1024;

    static CTextAlignment_Cli()
    {
        CTextAlignment_Cli.BELOW3 = 512;
        CTextAlignment_Cli.BELOW2 = 256;
        CTextAlignment_Cli.ALIGN_TOP2 = 128;
        CTextAlignment_Cli.BELOW = 64;
        CTextAlignment_Cli.ABOVE = 32;
        CTextAlignment_Cli.ALIGN_BOTTOM = 16;
        CTextAlignment_Cli.ALIGN_TOP = 8;
        CTextAlignment_Cli.ALIGN_RIGHT = 4;
        CTextAlignment_Cli.ALIGN_CENTER = 2;
        CTextAlignment_Cli.ALIGN_LEFT = 1;
    }
}
```

The COpenGraph_Cli class in NSOpenGraphLib_Cli namespace provide the main library interface.

```
public class COpenGraph_Cli
{
    public COpenGraph_Cli();

    public bool Initialize(IntPtr hWndPtr);
    public void SetMode(COpenGraph_Cli.Mode mode);
    public void SetWindowSize(int cx, int cy);
    public void SetBackgroundColor(CColor_Cli bgColor);
    public void Render();
    public void ForceRender();
    public void ShowAxes([MarshalAs(UnmanagedType.U1)] bool bShow);
    public int OnLButtonDown(ref CRect_Cli rect, uint nFlags, CPoint_Cli point);
    public void OnMouseMove(uint nFlags, CPoint_Cli point);
    public void ZoomMouseWheel(uint nFlags, short zDelta, CPoint_Cli point);
    public IntPtr GetModeCursor();
    public void OnSetCursor();
    public bool SaveImage(string fileName, [MarshalAs(UnmanagedType.U1)] bool bUseWhiteBackground);

    public void Pan(double fTx, double fTy);
    public void Rotate(double fRx, double fRy, double fRz);
    public void ZoomExtentView();

    public void SetFrontView();
    public void SetBackView();
    public void SetLeftView();
    public void SetRightView();
    public void SetTopView();
    public void SetBottomView();
    public void SetIsometricView();

    public bool TrackRect(IntPtr hWndPtr, CPoint_Cli point);
    public CRect_Cli GetTrackRect();
    public CPointf_Cli GetProjectedPoint(CPointf_Cli pt);
    public void Clear();

    public void SelectAll();
    public void UnSelectAll();
    public void ReverseSelectAll();
    public void GetSelected(ref List<CIdentity_Cli> list);
    public void GetSelectedNodes(ref List<CIdentity_Cli> list);
    public void GetSelectedLines(ref List<CIdentity_Cli> list);
    public void GetSelectedTriangles(ref List<CIdentity_Cli> list);
    public void GetSelectedQuads(ref List<CIdentity_Cli> list);
    public void GetSelectedWindowNodes(ref List<CIdentity_Cli> list);
    public void GetSelectedWindowLines(ref List<CIdentity_Cli> list);
```

```
public void GetSelectedWindowTriangles(ref List<CIdentity_Cli> list);
public void GetSelectedWindowQuads(ref List<CIdentity_Cli> list);

public CIdentity_Cli AddNode(int id, CPointf_Cli pt, int nSize, CColor_Cli color, CColor_Cli colorSelected,
                             int nStatus, string tag, string userData);
public void AddNodes(ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt, int nSize,
                     CColor_Cli color, CColor_Cli colorSelected, int nStatus, string tag, string userData);
public CIdentity_Cli AddLine(int id, CPointf_Cli pt1, CPointf_Cli pt2, int nThickness, CColor_Cli color, CColor_Cli colorSelected,
                             int nStatus, string tag, string userData);
public void AddLines(ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt1, List<CPointf_Cli> listPt2,
                     int nThickness, CColor_Cli color, CColor_Cli colorSelected, int nStatus, string tag, string userData);
public CIdentity_Cli AddTriangle(int id,  CPointf_Cli pt1, CPointf_Cli pt2, CPointf_Cli pt3, CColor_Cli color, CColor_Cli colorSelected,
                                 float alpha, int nStatus, string tag, string userData);
public void AddTriangles( ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt1, List<CPointf_Cli> listPt2,
                             List<CPointf_Cli> listPt3, CColor_Cli color, CColor_Cli colorSelected, float alpha, int nStatus,
                             string tag, string userData);
public CIdentity_Cli AddQuad(int id, CPointf_Cli pt1, CPointf_Cli pt2, CPointf_Cli pt3, CPointf_Cli pt4, CColor_Cli color,
                             CColor_Cli colorSelected, float alpha, int nStatus, string tag, string userData);
public void AddQuads(ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt1, List<CPointf_Cli> listPt2,
                     List<CPointf_Cli> listPt3, List<CPointf_Cli> listPt4, CColor_Cli color, CColor_Cli colorSelected,
                     float alpha, int nStatus, string tag, string userData);

public int AddFont(string typeface, int height, int weight, uint italic);
public CIdentity_Cli AddText(int id, int nFontId, CPointf_Cli pt, string text, CColor_Cli color, CColor_Cli colorSelected,
                             int nAlignment, int nStatus);

public CIdentity_Cli AddWindowNode(int id, CPointf_Cli pt, int nSize, CColor_Cli color, CColor_Cli colorSelected, int nStatus,
                                   string tag, string userData);
public void AddWindowNodes(ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt, int nSize,
                           CColor_Cli color, CColor_Cli colorSelected, int nStatus, string tag, string userData);
public CIdentity_Cli AddWindowLine(int id, CPointf_Cli pt1, CPointf_Cli pt2, int nThickness, CColor_Cli color, CColor_Cli colorSelected,
                                   int nStatus, string tag, string userData);
public void AddWindowLines(ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt1, List<CPointf_Cli> listPt2,
                            int nThickness, CColor_Cli color, CColor_Cli colorSelected, int nStatus, string tag, string userData);
public CIdentity_Cli AddWindowTriangle(int id, CPointf_Cli pt1, CPointf_Cli pt2, CPointf_Cli pt3, CColor_Cli color,
                                       CColor_Cli colorSelected, float alpha, int nStatus, string tag, string userData);
public void AddWindowTriangles(ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt1,
                               List<CPointf_Cli> listPt2, List<CPointf_Cli> listPt3, CColor_Cli color, CColor_Cli colorSelected,
                               float alpha, int nStatus, string tag, string userData);
public CIdentity_Cli AddWindowQuad(int id, CPointf_Cli pt1, CPointf_Cli pt2, CPointf_Cli pt3, CPointf_Cli pt4,
                                   CColor_Cli color, CColor_Cli colorSelected, float alpha, int nStatus, string tag, string userData);
public void AddWindowQuads(ref List<CIdentity_Cli> listIdentity, List<int> listId, List<CPointf_Cli> listPt1, List<CPointf_Cli> listPt2,
                           List<CPointf_Cli> listPt3, List<CPointf_Cli> listPt4, CColor_Cli color, CColor_Cli colorSelected,
                           float alpha, int nStatus,  string tag, string userData);

public CIdentity_Cli AddWindowText( int id, int nFontId, CPointf_Cli pt, string text, CColor_Cli color, CColor_Cli colorSelected,
                                   int nAlignment, int nStatus);


public enum Mode
{
```

```
            modeNONE = -1, // 0xFFFFFFFF
            modeTRACK = 0,
            modeSELECT = 1,
            modeZOOM = 2,
            modeREALPAN = 3,
            modeREALZOOM = 4,
            modeROTATE = 5,
        }

        public enum ButtonFlag
        {
            flagMK_LBUTTON = 1,
            flagMK_RBUTTON = 2,
            flagMK_SHIFT = 4,
            flagMK_CONTROL = 8,
            flagMK_MBUTTON = 16, // 0x00000010
        }
    }
```

As you can see, the interface is pretty lean. It does not require you to have deep knowledge about 3D graphics programming. The interface allows you to draw 2D and 3D objects such as points (also called nodes in the library), lines, triangles, quads and texts. The main difference between 2D and 3D objects is that 2D objects will have their coordinates in Windows pixel and their positions will not be affected by zooming, panning or rotating. Each object except texts may be assigned two different colors for selected and unselected states. All objects except texts can be selected using mouse to the end user feedback, thus making this library applicable and useful in many fields.

The following are the different drawing modes.

modeNONE: mouse movement will not trigger any action.

modeTRACK: a rubber-band rectangle will be drawing as mouse is pressed down and moved.

modeSELECT: objects will be selected or unselected by mouse clicking or mouse dragging.

modeZOOM: window zooming occurs by mouse dragging

modeREALPAN: objects will be panned by mouse moving.

modeREALZOOM: not used for now

modeROTATE: objects will be rotated by mouse dragging.

You can set the view position by calling Pan() and Rotate() functions.

```
void Pan(double fTx, double fTy);
void Rotate(double fRx, double fRy, double fRz);
```

Several preset view positions are available in the following functions.

```
void SetFrontView();
void SetBackView();
void SetLeftView();
void SetRightView();
void SetTopView();
void SetBottomView();
void SetIsometricView();
```

The ZoomExtentView() will reset the view so all objects can be seen in the window.

3D Graphic objects may be added by calling the proper Add functions. For example, call AddNode() to add a 3D point (or node).

```
public CIdentity_Cli AddNode(int id, CPointf_Cli pt, int nSize, CColor_Cli color, CColor_Cli colorSelected,
                             int nStatus, string tag, string userData);
```

The status can be STATUS_UNSELECTED or STATUS_SELECTED as defined in the following:

```
const int STATUS_UNSELECTED = 0x00000000L;
const int STATUS_SELECTED = 0x00000001L;
```

An identity as defined below is returned for each object being added. The uid is unique id that the library generated for each object. The id is the one that user-defined identifier that may or may not be unique.

```
public class CIdentity_Cli
{
    public string uid;
    public int id;
}
```

Each object may have two colors associated: one for unselected state and one for selected state. Currently, the library does not make use of tag and userData passed in but may make use of them in the future.

For triangles and quads, an alpha value between 0-1 may be passed in, which defines the opacity of the objects. For completely opaque object, use alpha value 1.0. For completely transparent object, use alpha value of 0.0.

2D Graphic objects may be added by calling the proper AddWindow functions. 2D objects are defined by their pixel positions on the screen and are not affected by zooming, panning or rotating. For example, call AddWindowNode() to add a 2D point (or node).

```
public CIdentity_Cli AddWindowNode(int id, CPointf_Cli pt, int nSize, CColor_Cli color, CColor_Cli colorSelected, int nStatus,
                                   string tag, string userData);
```

You may retrieve all selected objects by calling the following function.

```
public void GetSelected(ref List<CIdentity_Cli> list);
```

You may distinguish different objects by inspecting the uid prefixes:

N: for 3D nodes
L: for 3D lines
TRI: for 3D triangles
QUAD: for 3D quads
WN: for 2D nodes
WL: for 2D lines
WTRI: for 2D triangles
WQUAD: for 2D quads

All objects will be erased by calling the Clear() function. Rendered image may be saved to a file by calling SaveImage() function.

Please note if the graphics window is a control on the dialog, we generally need to have a user control to handle the appropriate Windows messages mentioned above. You can refer to the class OpenGLUserControl user control in OpenGraphWinformDialogClient demo project for details.

For WPF form, you can use the same WinForms user control inside the WPF WindowsFormsHost control. For example:

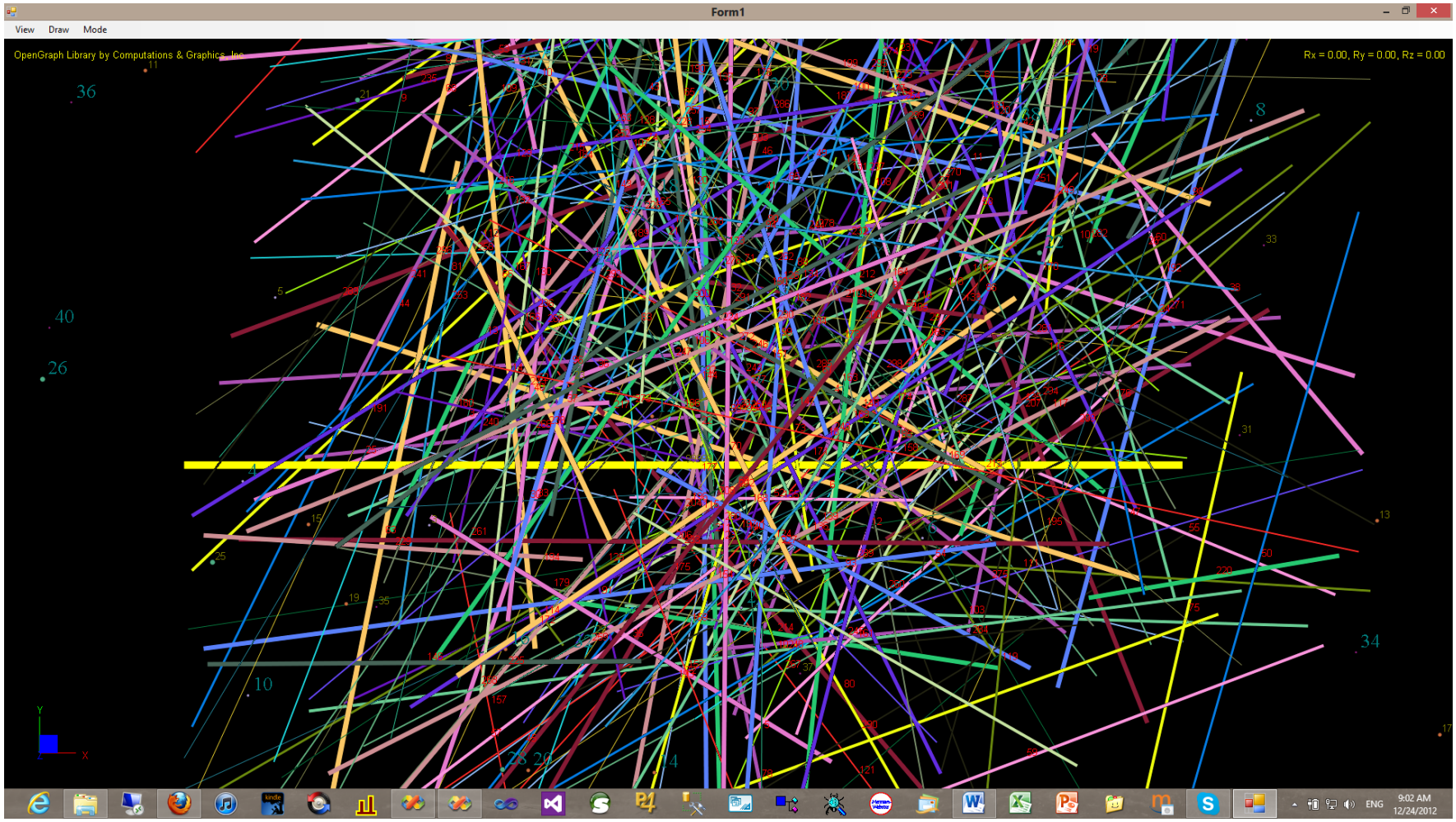
```
<Window x:Class="OpenGraphWpfClient.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:wf="clr-namespace:OpenGraphWpfClient"
        Title="MainWindow" Height="520" Width="934" SizeChanged="Window_SizeChanged">
    <Grid>
        <WindowsFormsHost Margin="182,49,40,31" Name="windowsFormsHost1">
            <wf:OpenGLUserControl Name="openGLUserControl"></wf:OpenGLUserControl>
        </WindowsFormsHost>
    </Grid>
</Window>
```

```
    public partial class MainWindow : Window
    {
        COpenGraph_Cli _openGraph_Cli = new COpenGraph_Cli();
        private int m_nCx = 0;
        private int m_nCy = 0;
        private bool m_bShowAxes = true;

        public MainWindow()
        {
            InitializeComponent();

            OpenGLUserControl = windowsFormsHost1.Child as OpenGLUserControl;
            openGLUserControl.Load_CallBack = this.Delegate_Load;
            openGLUserControl.MouseWheel_CallBack = this.Delegate_MouseWheel;
            openGLUserControl.MouseMove_CallBack = this.Delegate_MouseMove;
            openGLUserControl.MouseDown_CallBack = this.Delegate_MouseDown;
            openGLUserControl.Paint_CallBack = this.Delegate_Paint;
            openGLUserControl.Resize_CallBack = this.Delegate_Resize;
        }

        private void Window_SizeChanged(object sender, SizeChangedEventArgs e)
        {
            OpenGLUserControl = windowsFormsHost1.Child as OpenGLUserControl;
            _openGraph_Cli.Initialize(openGLUserControl.Handle);
            m_nCx = openGLUserControl.Width;
            m_nCy = openGLUserControl.Height;
            _openGraph_Cli.SetWindowSize(m_nCx, m_nCy);
```

```
            _openGraph_Cli.Render();
        }
    }
```

For more details, you can refer to the OpenGraphWpfClient demo project.

The following image is produced by OpenGraphWinformClient demo project. The complete main form code in the OpenGraphWinformClient demo project is shown below.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NSOpenGraphLib_Cli;

namespace OpenGraphWinformClient
{
    public partial class Form1 : Form
    {
        COpenGraph_Cli _openGraph_Cli = new COpenGraph_Cli();
        private int m_nCx = 0;
        private int m_nCy = 0;

        public Form1()
        {
            InitializeComponent();
            this.MouseWheel += new MouseEventHandler(Form1_MouseWheel);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            _openGraph_Cli.Initialize(this.Handle);
            m_nCx = this.ClientSize.Width;
            m_nCy = this.ClientSize.Height - menuStrip1.Height;
            _openGraph_Cli.SetWindowSize(m_nCx, m_nCy);
            _openGraph_Cli.Render();
            _openGraph_Cli.SetMode(COpenGraph_Cli.Mode.modeNONE);

        }

        private void Form1_MouseWheel(object sender, MouseEventArgs e)
        {
            CPoint_Cli pt = new CPoint_Cli();
            Point ptTemp = new Point();
            ptTemp.X = e.X;
            ptTemp.Y = e.Y;
            Point screenPt = PointToScreen(ptTemp);
            pt.X = screenPt.X;
            pt.Y = screenPt.Y;
            if (_openGraph_Cli != null)
            {
                // jxu -- we expect screen coordinate here
                _openGraph_Cli.ZoomMouseWheel(0, (short)(e.Delta / 120), pt);
            }

        }
```

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    _openGraph_Cli.OnSetCursor();
    if (e.Button == MouseButtons.Left)
    {
        CRect_Cli rect = new CRect_Cli();
        CPoint_Cli pt = new CPoint_Cli();
        pt.X = e.X;
        pt.Y = e.Y;

        uint nFlags = (int)COpenGraph_Cli.ButtonFlag.flagMK_LBUTTON;
        if (_openGraph_Cli != null)
        {
            _openGraph_Cli.OnMouseMove(nFlags, pt);
        }
    }
}

private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        CRect_Cli rect = new CRect_Cli();
        CPoint_Cli pt = new CPoint_Cli();
        pt.X = e.X;
        pt.Y = e.Y;
        uint nFlags = 0;
        if (_openGraph_Cli != null)
        {
            _openGraph_Cli.OnLButtonDown(ref rect, nFlags, pt);
        }
    }
}

private void Form1_Paint(object sender, PaintEventArgs e)
{
    if (_openGraph_Cli != null)
    {
        _openGraph_Cli.Render();
    }
}

private void Form1_Resize(object sender, EventArgs e)
{
    if (_openGraph_Cli != null)
    {
        m_nCx = this.ClientSize.Width;
        m_nCy = this.ClientSize.Height - menuStrip1.Height;
        _openGraph_Cli.SetWindowSize(m_nCx, m_nCy);
    }
```

```
}


// view ---------------------------------------------

private void selectAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SelectAll();
    _openGraph_Cli.Render();
}

private void unselectAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.UnSelectAll();
    _openGraph_Cli.Render();
}

private void reverseSelectToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.ReverseSelectAll();
    _openGraph_Cli.Render();
}

private void showSelectedToolStripMenuItem_Click(object sender, EventArgs e)
{
    List<CIdentity_Cli> list = new List<CIdentity_Cli>();
    _openGraph_Cli.GetSelected(ref list);

    StringBuilder sb = new StringBuilder();
    sb.Append("The following entities are selected:\n");
    for (int i = 0; i < list.Count; i++)
    {
        sb.Append(string.Format("[{0}, {1}]; ", list[i].uid, list[i].id));
    }
    MessageBox.Show(sb.ToString());
}

private void viewFrontToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SetFrontView();
}

private void viewIsometricToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SetIsometricView();
}

private void resetViewToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.ZoomExtentView();
}
```

```
private void projectPointToolStripMenuItem_Click(object sender, EventArgs e)
{
    CPointf_Cli pt = new CPointf_Cli(0, 0, 0);
    CPointf_Cli ptProjected = _openGraph_Cli.GetProjectedPoint(pt);
    string s = string.Format("projected point for (0, 0, 0) is: {0}, {1}, {2}", ptProjected.X,
                                    ptProjected.Y, ptProjected.Z);
    MessageBox.Show(s);
}

// draw ----------------------------------------------------
private void drawNodesToolStripMenuItem_Click(object sender, EventArgs e)
{
    int fontId = _openGraph_Cli.AddFont("Times New Roman", 30, 0, 0);
    Random random = new Random();

    const int nNodes = 400;
    const int nBatches = 10;
    const int nNodesPerBatch = nNodes / nBatches;
    for (int j = 0; j < nBatches; j++)
    {
        CColor_Cli color = new CColor_Cli((byte)(random.Next(0, 256)), (byte)(random.Next(0, 256)),
                                    (byte)(random.Next(0, 256)));
        CColor_Cli colorSelected = new CColor_Cli(255, 0, 0);
        List<CPointf_Cli> pPt = new List<CPointf_Cli>();
        List<int> pId = new List<int>();
        int nSize = (random.Next(0, 256) % 8);
        string strTag;
        strTag = string.Format("Tag_{0}", j);
        for (int i = 0; i < nNodesPerBatch; i++)
        {
            double x = ((double)random.Next(0, 256) / 256 - 0.5) * 4;
            double y = ((double)random.Next(0, 256) / 256 - 0.5) * 4;
            double z = ((double)random.Next(0, 256) / 256 - 0.5) * 4;
            string strText;
            int id = j * nNodesPerBatch + i + 1;
            strText = string.Format("{0}", id);
            if (i % 2 == 0)
            {
                _openGraph_Cli.AddText(id, 0, new CPointf_Cli(x, y, z), strText, new CColor_Cli(125, 123, 0),
                                    colorSelected,
                                    CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM,
                                    CStatus_Cli.STATUS_UNSELECTED);
            }
            else
            {
                _openGraph_Cli.AddText(id, fontId, new CPointf_Cli(x, y, z), strText,
                                                new CColor_Cli(0, 123, 123), colorSelected,
                                            CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM,
                                                CStatus_Cli.STATUS_UNSELECTED);
            }
```

```
            pId.Add(id);
            pPt.Add(new CPointf_Cli(x, y, z));
        }
        List<CIdentity_Cli> pIdentity = new List<CIdentity_Cli>();
        // for performance reason, you should send in your data in batches
        _openGraph_Cli.AddNodes(ref pIdentity, pId, pPt, nSize, color, colorSelected,
                                    CStatus_Cli.STATUS_UNSELECTED, strTag, null);
    }


    _openGraph_Cli.AddNode(99999, new CPointf_Cli(), 15, new CColor_Cli(255, 255, 0),
                               new CColor_Cli(0, 255, 255), 0, null, null);
    _openGraph_Cli.AddText(99999, 0, new CPointf_Cli(), "99999", new CColor_Cli(125, 123, 0), new CColor_Cli(0, 255, 255),
                               CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM, CStatus_Cli.STATUS_UNSELECTED);


    _openGraph_Cli.Render();
}

private void drawLinesToolStripMenuItem_Click(object sender, EventArgs e)
{
    Random random = new Random();

    const int nNodes = 300;
    const int nBatches = 30;
    const int nLinesPerBatch = nNodes / nBatches;
    for (int j = 0; j < nBatches; j++)
    {
        CColor_Cli color = new CColor_Cli(((byte)random.Next(0, 256)), ((byte)random.Next(0, 256)),
                                                ((byte)random.Next(0, 256)));
        CColor_Cli colorSelected = new CColor_Cli(255, 0, 0);
        List<CPointf_Cli> pPt1 = new List<CPointf_Cli>();
        List<CPointf_Cli> pPt2 = new List<CPointf_Cli>();
        List<int> pId = new List<int>();
        int nSize = (random.Next(0, 256) % 8);
        string strTag;
        strTag = string.Format("Tag_{0}", j);
        for (int i = 0; i < nLinesPerBatch; i++)
        {
            double x1 = ((double)random.Next(0, 256) / 256 - 0.5) * 4;
            double y1 = ((double)random.Next(0, 256) / 256 - 0.5) * 4;
            double z1 = ((double)random.Next(0, 256) / 256 - 0.5) * 4;
            double x2 = ((double)random.Next(0, 256) / 256 - 0.3) * 4;
            double y2 = ((double)random.Next(0, 256) / 256 - 0.2) * 4;
            double z2 = ((double)random.Next(0, 256) / 256 - 0.4) * 4;
            int id = j * nLinesPerBatch + i + 1;
            pId.Add(id);
            pPt1.Add(new CPointf_Cli(x1, y1, z1));
            pPt2.Add(new CPointf_Cli(x2, y2, z2));

            string strText;
```

```
            strText = string.Format("{0}", id);
            CPointf_Cli pos = new CPointf_Cli((pPt1[i].X + pPt2[i].X) / 2, (pPt1[i].Y + pPt2[i].Y) / 2,
                                              (pPt1[i].Z + pPt2[i].Z) / 2);
            _openGraph_Cli.AddText(id, 0, pos, strText, new CColor_Cli(255, 0, 0), colorSelected,
                                   CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM,
                                         CStatus_Cli.STATUS_UNSELECTED);
        }
        List<CIdentity_Cli> pIdentity = new List<CIdentity_Cli>();
        // for performance reason, you should send in your data in batches
        _openGraph_Cli.AddLines(ref pIdentity, pId, pPt1, pPt2, nSize, color, colorSelected, 0, strTag, null);
    }

    _openGraph_Cli.AddLine(99999, new CPointf_Cli(-2, 0, 0), new CPointf_Cli(2, 0, 0), 15, new CColor_Cli(255, 255, 0),
                           new CColor_Cli(0, 255, 255), 0, null, null);
    _openGraph_Cli.AddText(99999, 0, new CPointf_Cli(0, 0, 0), "99999", new CColor_Cli(125, 123, 0),
                                 new CColor_Cli(0, 123, 0),
                           CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM, CStatus_Cli.STATUS_UNSELECTED);

    _openGraph_Cli.Render();
}

private void drawTrianglesToolStripMenuItem_Click(object sender, EventArgs e)
{
    Random random = new Random();
    CPointf_Cli[] pt = { new CPointf_Cli(0.5, 0, -0.5), new CPointf_Cli(0.5, 0, 0.5),
                           new CPointf_Cli(-0.5, 0, 0.5), new CPointf_Cli(0, 2, 0) };
    CColor_Cli colorSelected = new CColor_Cli(255, 255, 0);
    string strTag = "";

    _openGraph_Cli.AddTriangle(1, pt[0], pt[1], pt[2], new CColor_Cli(123, 0, 123), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddTriangle(2, pt[0], pt[1], pt[3], new CColor_Cli(65, 0, 0), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddTriangle(3, pt[0], pt[3], pt[2], new CColor_Cli(123, 0, 0), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddTriangle(4, pt[2], pt[3], pt[1], new CColor_Cli(225, 0, 0), colorSelected, 1, 0, strTag, null);

    // two transparent triangle
    List<CPointf_Cli> pt1 = new List<CPointf_Cli>();
    pt1.Add(new CPointf_Cli(-1.5, -1.5, 1));
    pt1.Add(new CPointf_Cli(-1.5, -1.5, 1));

    List<CPointf_Cli> pt2 = new List<CPointf_Cli>();
    pt2.Add(new CPointf_Cli(1.2, 1.5, 1));
    pt2.Add(new CPointf_Cli(1.2, 1.5, 1));

    List<CPointf_Cli> pt3 = new List<CPointf_Cli>();
    pt3.Add(new CPointf_Cli(-2, 2.2, 1));
    pt3.Add(new CPointf_Cli(2, 0.2, 1));

    List<int> ids = new List<int>();
    ids.Add(5);
    ids.Add(6);
```

```
    List<CIdentity_Cli> identity = new List<CIdentity_Cli>();
    _openGraph_Cli.AddTriangles(ref identity, ids, pt1, pt2, pt3, new CColor_Cli(0, 0, 255), colorSelected,
                                    0.5f, 0, strTag, null);
    _openGraph_Cli.AddText(5, 0, new CPointf_Cli((-1.5 + 1.2 + -2) / 3, (-1.5 + 1.5 + 2.2)/3, (1.0 + 1 + 1)/3),
                               "5", new CColor_Cli(125, 123, 0), colorSelected,
                              CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM, CStatus_Cli.STATUS_UNSELECTED);
    _openGraph_Cli.AddText(6, 0, new CPointf_Cli((-1.5 + 1.2 + 2) / 3.0, (-1.5 + 1.5 + 0.2)/3.0, (1 + 1 + 1)/3.0),
                               "6", new CColor_Cli(125, 123, 0), colorSelected,
                              CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM, CStatus_Cli.STATUS_UNSELECTED);

    _openGraph_Cli.Render();
}


private void BoxPts(ref CPointf_Cli[] pt, CPointf_Cli ptCenter, double fDeltaX, double fDeltaY, double fDeltaZ)
{
    pt[0].X = ptCenter.X + fDeltaX / 2.0;
    pt[0].Y = ptCenter.Y + fDeltaY / 2.0;
    pt[0].Z = ptCenter.Z + fDeltaZ / 2.0;
    CPointf_Cli.Copy(pt[1], pt[0]);    // jxu: import, CPointf_Cli is reference type
    pt[1].X = ptCenter.X - fDeltaX / 2.0;
    CPointf_Cli.Copy(pt[2], pt[1]);
    pt[2].Y = ptCenter.Y - fDeltaY / 2.0;
    CPointf_Cli.Copy(pt[3], pt[2]);
    pt[3].X = pt[0].X;

    for (int i = 0; i < 4; i++)
    {
        CPointf_Cli.Copy(pt[i + 4], pt[i]);
        pt[i + 4].Z = ptCenter.Z - fDeltaZ / 2.0;
    }

}

private void drawQuadsToolStripMenuItem_Click(object sender, EventArgs e)
{
    CColor_Cli colorSelected = new CColor_Cli(123, 0, 255);
    string strTag = "";
    CPointf_Cli[] pt = new CPointf_Cli[8];
    for (int i = 0; i < 8; i++ )
    {
        pt[i] = new CPointf_Cli(0, 0, 0);
    }
    CColor_Cli color = new CColor_Cli((byte)(0.8 * 255), 0, 0);
    BoxPts(ref pt, new CPointf_Cli(0, 0, 0), 0.5, 2, 1);
    _openGraph_Cli.AddQuad(1, pt[1], pt[0], pt[4], pt[5], new CColor_Cli(65, 0, 0), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddQuad(2, pt[3], pt[2], pt[6], pt[7], new CColor_Cli(65, 0, 0), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddQuad(3, pt[2], pt[1], pt[5], pt[6], new CColor_Cli(123, 0, 0), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddQuad(4, pt[0], pt[3], pt[7], pt[4], new CColor_Cli(123, 0, 0), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddQuad(5, pt[0], pt[1], pt[2], pt[3], new CColor_Cli(225, 0, 0), colorSelected, 1, 0, strTag, null);
    _openGraph_Cli.AddQuad(6, pt[7], pt[6], pt[5], pt[4], new CColor_Cli(225, 0, 0), colorSelected, 1, 0, strTag, null);
```

```
    List<CPointf_Cli> pt1 = new List<CPointf_Cli>();
    pt1.Add(new CPointf_Cli(-1.5, -1.5, 10));
    pt1.Add(new CPointf_Cli(0, -1.5, 10));

    List<CPointf_Cli> pt2 = new List<CPointf_Cli>();
    pt2.Add(new CPointf_Cli(0, -1.5, 10));
    pt2.Add(new CPointf_Cli(1.5, -1.5, 10));

    List<CPointf_Cli> pt3 = new List<CPointf_Cli>();
    pt3.Add(new CPointf_Cli(0, 1.5, 10));
    pt3.Add(new CPointf_Cli(1.5, 1.5, 10));

    List<CPointf_Cli> pt4 = new List<CPointf_Cli>();
    pt4.Add(new CPointf_Cli(-1.5, 1.5, 10));
    pt4.Add(new CPointf_Cli(0, 1.5, 10));

    List<int> ids = new List<int>();
    ids.Add(7);
    ids.Add(8);
    List<CIdentity_Cli> identity = new List<CIdentity_Cli>();
    _openGraph_Cli.AddQuads(ref identity, ids, pt1, pt2, pt3, pt4, new CColor_Cli(0, 0, 255), colorSelected,
                            0.5f, 0, strTag, null);
    _openGraph_Cli.AddText(5, 0, new CPointf_Cli(-0.75, 0, 10), "7", new CColor_Cli(125, 123, 0), colorSelected,
                           CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM, CStatus_Cli.STATUS_UNSELECTED);
    _openGraph_Cli.AddText(5, 0, new CPointf_Cli(0.75, 0, 10), "8", new CColor_Cli(125, 123, 0), colorSelected,
                           CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM, CStatus_Cli.STATUS_UNSELECTED);

    _openGraph_Cli.Render();
}

private void drawToolStripMenuItem1_Click(object sender, EventArgs e)
{
    Random random = new Random();
    int fontId = _openGraph_Cli.AddFont("Times New Roman", 30, 0, 0);

    const int nNodes = 40;
    const int nBatches = 4;
    const int nNodesPerBatch = nNodes / nBatches;
    for (int j = 0; j < nBatches; j++)
    {
        CColor_Cli color = new CColor_Cli(((byte)random.Next(0, 256)), ((byte)random.Next(0, 256)),
                                          ((byte)random.Next(0, 256)));
        CColor_Cli colorSelected = new CColor_Cli(255, 0, 0);
        List<CPointf_Cli> pPt = new List<CPointf_Cli>();
        List<int> pId = new List<int>();
        int nSize = (random.Next(0, 256) % 8);
        string strTag;
        strTag = string.Format("Tag_{0}", j);
        for (int i = 0; i < nNodesPerBatch; i++)
        {
```

```
            double x = random.Next(1, 3000) % m_nCx;
            double y = random.Next(1, 3000) % m_nCy;
            double z = 0;
            string strText;
            int id = j * nNodesPerBatch + i + 1;
            strText = string.Format("{0}", id);

            if (i % 2 == 0)
            {
                _openGraph_Cli.AddWindowText(id, 0, new CPointf_Cli(x, y, z), strText,
                                                    new CColor_Cli(125, 123, 0), colorSelected,
                                        CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM,
                                                CStatus_Cli.STATUS_UNSELECTED);
            }
            else
            {
                _openGraph_Cli.AddWindowText(id, fontId, new CPointf_Cli(x, y, z), strText,
                                                    new CColor_Cli(0, 123, 123), colorSelected,
                                                CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM,
                                                    CStatus_Cli.STATUS_UNSELECTED);
            }
            pId.Add(j * nNodesPerBatch + i + 1);
            pPt.Add(new CPointf_Cli(x, y, z));
        }
        List<CIdentity_Cli> pIdentity = new List<CIdentity_Cli>();

        // for performance reason, you should send in your data in batches
        _openGraph_Cli.AddWindowNodes(ref pIdentity, pId, pPt, nSize, color, colorSelected, 0, strTag, null);
    }

    _openGraph_Cli.AddWindowNode(99999, new CPointf_Cli(m_nCx / 2, m_nCy / 2, 0), 10,
                                    new CColor_Cli(255, 255, 0), new CColor_Cli(0, 255, 255), 0, null, null);
    _openGraph_Cli.AddWindowText(99999, 0, new CPointf_Cli(m_nCx / 2, m_nCy / 2, 0), "99999",
                                    new CColor_Cli(255, 0, 0), new CColor_Cli(0, 255, 0),
                                    CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM,
                                    CStatus_Cli.STATUS_UNSELECTED);

    _openGraph_Cli.Render();
}

private void drawWindowLinesToolStripMenuItem_Click(object sender, EventArgs e)
{
    Random random = new Random();
    int fontId = _openGraph_Cli.AddFont("Times New Roman", 30, 0, 0);

    const int nNodes = 20;
    const int nBatches = 10;
    const int nLinesPerBatch = nNodes / nBatches;
    for (int j = 0; j < nBatches; j++)
    {
        CColor_Cli color = new CColor_Cli(((byte)random.Next(0, 256)), ((byte)random.Next(0, 256)),
```

```
                                            ((byte)random.Next(0, 256)));
        CColor_Cli colorSelected = new CColor_Cli(255, 0, 0);
        List<CPointf_Cli> pPt1 = new List<CPointf_Cli>();
        List<CPointf_Cli> pPt2 = new List<CPointf_Cli>();
        List<int> pId = new List<int>();
        int nSize = (random.Next(0, 256) % 8);
        string strTag;
        strTag = string.Format("Tag_{0}", j);
        for (int i = 0; i < nLinesPerBatch; i++)
        {
            double x1 = random.Next(1, 3000) % m_nCx;
            double y1 = random.Next(1, 3000) % m_nCy;
            double z1 = 0;
            double x2 = Math.Abs(random.Next(1, 3000) % m_nCx - 300);
            double y2 = Math.Abs(random.Next(1, 3000) % m_nCy - 20); ;
            double z2 = 0;
            int id = j * nLinesPerBatch + i + 1;
            pId.Add(id);
            pPt1.Add(new CPointf_Cli(x1, y1, z1));
            pPt2.Add(new CPointf_Cli(x2, y2, z2));
            string strText;
            strText = string.Format("{0}", id);
            CPointf_Cli pos = new CPointf_Cli((pPt1[i].X + pPt2[i].X) / 2, (pPt1[i].Y + pPt2[i].Y) / 2,
                                              (pPt1[i].Z + pPt2[i].Z) / 2);
            _openGraph_Cli.AddWindowText(id, 0, pos, strText, new CColor_Cli(125, 123, 0), colorSelected,
                                         CTextAlignment_Cli.ALIGN_LEFT | CTextAlignment_Cli.ALIGN_BOTTOM,
                                               CStatus_Cli.STATUS_UNSELECTED);
        }
        List<CIdentity_Cli> pIdentity = new List<CIdentity_Cli>();
        // for performance reason, you should send in your data in batches
        _openGraph_Cli.AddWindowLines(ref pIdentity, pId, pPt1, pPt2, nSize, color, colorSelected, 0, strTag, null);
    }

    // draw the boundary line
    List<CPointf_Cli> pt = new List<CPointf_Cli>();
    pt.Add(new CPointf_Cli(5, 5, 0));
    pt.Add(new CPointf_Cli(m_nCx - 10, 5, 0));
    pt.Add(new CPointf_Cli(m_nCx - 10, m_nCy - 10, 0));
    pt.Add(new CPointf_Cli(5, m_nCy - 10, 0));
    _openGraph_Cli.AddWindowLine(1, pt[0], pt[1], 3, new CColor_Cli(255, 0, 0), new CColor_Cli(255, 255, 0), 0, "", null);
    _openGraph_Cli.AddWindowLine(2, pt[1], pt[2], 3, new CColor_Cli(255, 0, 0), new CColor_Cli(255, 255, 0), 0, "", null);
    _openGraph_Cli.AddWindowLine(3, pt[2], pt[3], 3, new CColor_Cli(255, 0, 0), new CColor_Cli(255, 255, 0), 0, "", null);
    _openGraph_Cli.AddWindowLine(4, pt[3], pt[0], 3, new CColor_Cli(255, 0, 0), new CColor_Cli(255, 255, 0), 0, "", null);

    _openGraph_Cli.Render();
}

private void drawWindowTrianglesToolStripMenuItem_Click(object sender, EventArgs e)
{
    CPointf_Cli[] pt = {new CPointf_Cli(m_nCx / 8, m_nCy / 8, 0), new CPointf_Cli(m_nCx / 1.5, m_nCy / 8, 0),
                        new CPointf_Cli(m_nCx /2, m_nCy / 2, 0)};
```

```
    CColor_Cli colorSelected = new CColor_Cli(255, 255, 0);
    string strTag = "";
    _openGraph_Cli.AddWindowTriangle(1, pt[0], pt[1], pt[2], new CColor_Cli(123, 0, 123), colorSelected,1,
                                            0, strTag, null);

    List<CPointf_Cli> pt1  = new List<CPointf_Cli>();
    pt1.Add(new CPointf_Cli(m_nCx / 4, m_nCy / 1.1, 0));
    pt1.Add(new CPointf_Cli(m_nCx / 1.5, m_nCy / 1.1, 0));

    List<CPointf_Cli> pt2 = new List<CPointf_Cli>();
    pt2.Add(new CPointf_Cli(m_nCx / 2, m_nCy / 5, 0));
    pt2.Add(new CPointf_Cli(m_nCx / 4, m_nCy / 4, 0));

    List<CPointf_Cli> pt3 = new List<CPointf_Cli>();
    pt3.Add(new CPointf_Cli(10, 10, 0));
    pt3.Add(new CPointf_Cli(m_nCx / 3, m_nCy / 8, 0));
    List<CIdentity_Cli> pIdentity = new List<CIdentity_Cli>();
    List<int> pId = new List<int>();
    pId.Add(2);
    pId.Add(3);
    _openGraph_Cli.AddWindowTriangles(ref pIdentity, pId, pt1, pt2, pt3, new CColor_Cli(123, 123, 123),
                                            colorSelected,0.5f, 0, strTag, null);
    _openGraph_Cli.Render();
}

private void drawWindowQuadsToolStripMenuItem_Click(object sender, EventArgs e)
{
    CPointf_Cli[] pt = {new CPointf_Cli(25, 25, 0), new CPointf_Cli(m_nCx - 25, 25, 0),
                        new CPointf_Cli(m_nCx - 25, m_nCy - 25, 0), new CPointf_Cli(25, m_nCy - 25, 0)};
    CColor_Cli colorSelected = new CColor_Cli(255, 255, 0);
    string strTag = "";
    _openGraph_Cli.AddWindowQuad(1, pt[0], pt[1], pt[2], pt[3], new CColor_Cli(123, 0, 123), colorSelected,
                                    0.5f, 0, strTag, null);

    List<CPointf_Cli> pt1 = new List<CPointf_Cli>();
    pt1.Add(new CPointf_Cli(50, 50, 0));
    pt1.Add(new CPointf_Cli(m_nCx*2 / 3, 50, 0));

    List<CPointf_Cli> pt2 = new List<CPointf_Cli>();
    pt2.Add(new CPointf_Cli(m_nCx / 3,50, 0));
    pt2.Add(new CPointf_Cli(m_nCx -50, 50, 0));

    List<CPointf_Cli> pt3 = new List<CPointf_Cli>();
    pt3.Add(new CPointf_Cli(m_nCx / 3,m_nCy - 50, 0));
    pt3.Add(new CPointf_Cli(m_nCx -50, m_nCy - 50, 0));

    List<CPointf_Cli> pt4 = new List<CPointf_Cli>();
    pt4.Add(new CPointf_Cli(50,m_nCy - 50, 0));
    pt4.Add(new CPointf_Cli(m_nCx*2 / 3, m_nCy - 50, 0));

    List<CIdentity_Cli> pIdentity = new List<CIdentity_Cli>();
```

```
    List<int> pId = new List<int>();
    pId.Add(2);
    pId.Add(3);
    _openGraph_Cli.AddWindowQuads(ref pIdentity, pId, pt1, pt2, pt3, pt4, new CColor_Cli(123, 123, 123),
                                  colorSelected,0.5f, 0, strTag, null);
    _openGraph_Cli.Render();
}

private void clearToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.Clear();
    _openGraph_Cli.Render();
}

private void saveImageToolStripMenuItem_Click(object sender, EventArgs e)
{
    bool bUseWhiteBackground = false;
    // use bmp for highest quality of picture
    const string fileName = "c:\\temp\\test_winform.bmp";
    _openGraph_Cli.SaveImage(fileName, bUseWhiteBackground);
    MessageBox.Show(fileName);
}

// mode ----------------------------------------

private void defaultToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SetMode(COpenGraph_Cli.Mode.modeNONE);
}

private void trackToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SetMode(COpenGraph_Cli.Mode.modeTRACK);
}

private void selectToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SetMode(COpenGraph_Cli.Mode.modeSELECT);
}

private void panToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SetMode(COpenGraph_Cli.Mode.modeREALPAN);
}

private void rotateToolStripMenuItem_Click(object sender, EventArgs e)
{
    _openGraph_Cli.SetMode(COpenGraph_Cli.Mode.modeROTATE);
}

private void windowZoomToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
            _openGraph_Cli.SetMode(COpenGraph_Cli.Mode.modeZOOM);
        }

        private void pixelPanToolStripMenuItem_Click(object sender, EventArgs e)
        {
            _openGraph_Cli.Pan(20, 10); // change the steps here to suit your needs
        }

    }

}
```