

Laboratory Exercise 16 (More MFC)

Topics: MFC and Polymorphic Containers

Goals: Upon successful completion of this lab you should be able to:

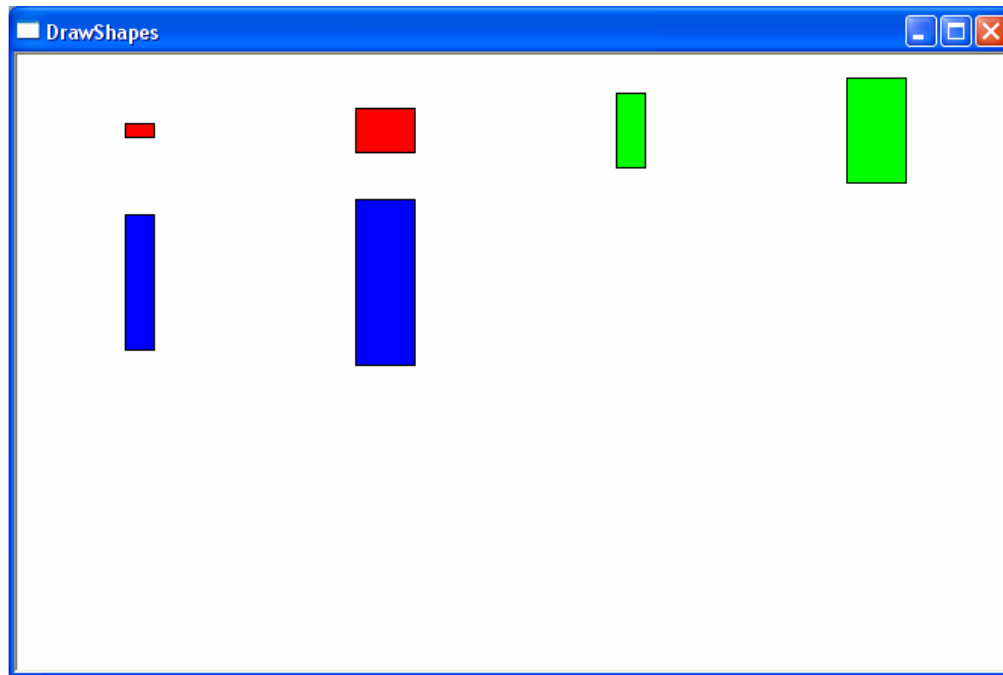
7. Write a simple windows program using the MFC Frame Window and Dialog classes
8. Include a doubly linked list of pointers to polymorphic objects in an MFC program

Related text sections:

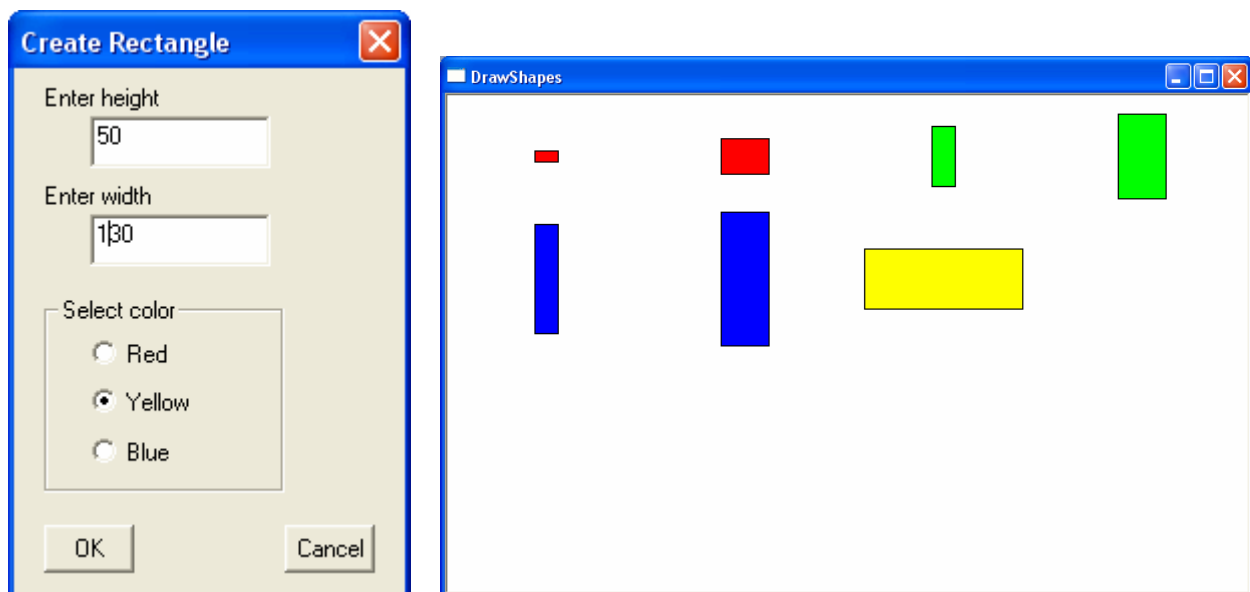
- Chapter 5 (Introduction to MFC)
- Chapter 14
- Chapter 15
- Chapter 17

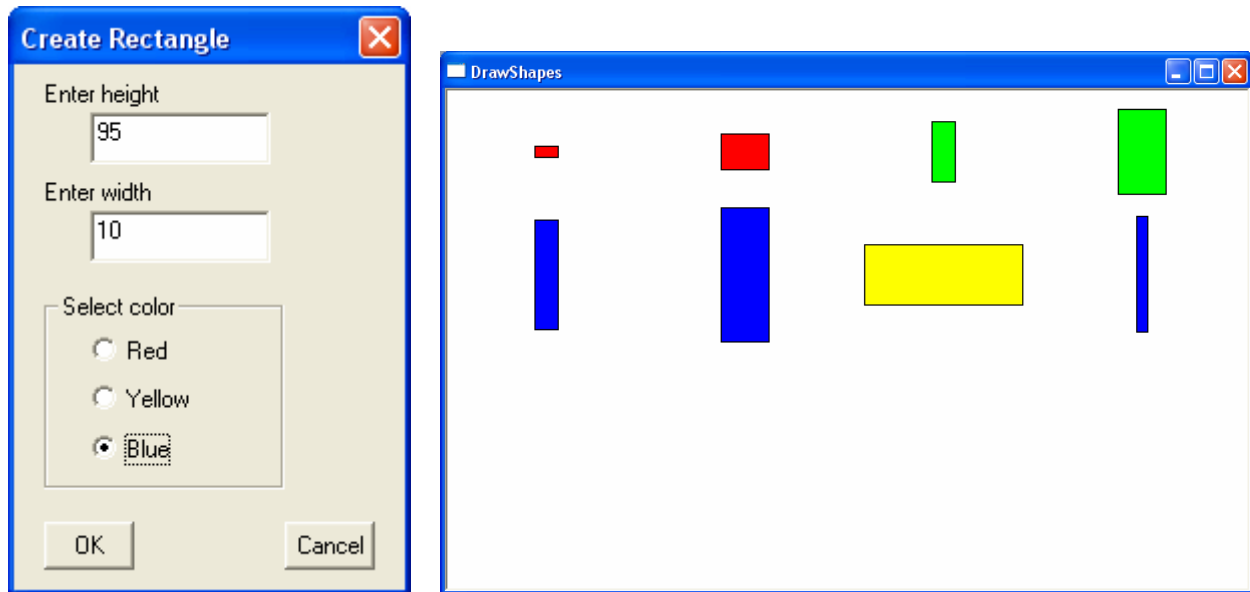
Laboratory Exercise 16 Instructions

23. Create a new Visual C++ application project called Lab16.
24. Copy the header (.h) and implementation (.cpp) files from ~tiawatts/cs215pickup/Lab16 to the folder created for your Lab16 project.
25. Compile and execute the program. You should see:



26. Press any arrow key to popup the "Create Rectangle" dialog box to add more rectangles to the display:





27. Copy your LList2.tmp template file to a new header file called LList2.h in your Lab16 project. Modify the output (<<) prototype as indicated:
- ```
// friend ostream & operator << >> (ostream & outs, const LList2<LT> & L);
friend ostream & operator << (ostream & outs, const LList2<LT> & L);
```
28. Replace the array m\_rectangles in the class CShapesDocument with a linked list:
- ```
CRectangle * m_rectangles [16]; →
LList2 <CRectangle> m_rectangles;
```
29. Modify CShapesDocument.cpp as needed to accommodate the array to linked list replacement in the previous step.
- Each call to the Add function in the class CShapeDocument should use InsertLast to add the new shape to the end of the m_rectangles linked list.
 - The four loop in the Paint function should use an iterator to traverse the m_rectangles linked list.
 - Currently the CShapeDocument Paint function paints the rectangles in a 4x4 grid in the device context window. Only 16 rectangles can be accommodated. Modify the Add function to accept more than 16 rectangles (theoretically it should accept an infinitely many rectangles) and to calculate new values for m_nCols and m_nRows such that if there are N rectangles in the list, m_nCols and m_nRows are each less than or equal to the ceiling of the square root of N, and m_nRows is less than or equal to m_nCols.
30. Compile and execute the modified program. Add more rectangles to the display to test your modified Add and Paint functions.
31. Create a copy of your Polygon class files from Labs 13 and 14. Modify the class names to follow the standards recommended for MFC:
- Polygon → CPolygon
 - Triangle → CTriangle
 - Equilateral → CEquilateral
 - etc.

10. Add a “Paint” function to your CPolygon class. The Paint function for CRectangle is listed below; simplified CPolygon class currently used in Lab 16 is listed at the end of this Lab exercise. Use it as a model.

```
void CRectangle::Paint (CPaintDC & dc, int midX, int midY)
{
    CBrush colorBrush;
    colorBrush.CreateSolidBrush (color);
    CBrush * pBrushSv = dc.SelectObject (&colorBrush);
    dc.Rectangle (midX - side2 / 2, midY - side / 2,
                 midX + side2 / 2, midY + side / 2);
    dc.SelectObject (pBrushSv);
}
```

32. Add 3 new Create Dialogs to your project:

- a. Equilateral Triangle
- b. Right Triangle
- c. Regular Polygon

33. Modify the switch statement in the function OnKeyDown in the class CShapeWin so that each arrow key adds a different shape to the document.

34. Add Paint functions to the classes CEquilateral, CRight, and CRegular.

35. Replace the linked list m_rectangles in CShapeDocument with a linked list of polygons:

```
LList2 <CPolygon> m_polygons;
```

36. Test your modified program. You should be able to add and display a variety of shapes using the arrow keys and the new dialogs.

37. When you are convinced that your program is working correctly, tar and zip your folder into a file called as *yourlastnameL16.tgz* and drop it into the course dropbox..

```

#include <iostream>
using namespace std;

template <class LT> class LList2;

template <class LT> ostream & operator << (ostream & outs, const LList2<LT> & L);

template <class LT>
class LList2
{
private:
    typedef class LNode
    {
        public:
            LNode ();
            LT data;
            LNode * next;
            LNode * prev;
    };

public:
    typedef class Iterator
    {
        public:
            Iterator ();
            Iterator (LNode * NP);
            LT & operator * () const;
            Iterator operator ++ ();
            Iterator operator ++ (int);
            Iterator operator -- ();
            Iterator operator -- (int);
            bool operator == (const Iterator & other) const;
            bool operator != (const Iterator & other) const;
        private:
            LNode * current;
    };
    LList2 ();
    LList2 (const LList2 & other);
    ~LList2 ();
    LList2 & operator = (const LList2 & other);
    bool operator == (const LList2 & other);
    int Size () const;
// friend ostream & operator << >> (ostream & outs, const LList2<LT> & L);
friend ostream & operator << (ostream & outs, const LList2<LT> & L);
    bool InsertFirst (const LT & value);
    bool InsertLast (const LT & value);
    bool DeleteFirst ();
    bool DeleteLast ();
    void Forward (void function (const LT & param));
    void Backward (void function (const LT & param));
    Iterator begin () const;
    Iterator rbegin () const;
    Iterator end () const;
    Iterator rend () const;
    LT & operator [] (const int & index) const;
private:
    LNode * first;
    LNode * last;
    int size;
};

```

```

// File: CShapesApp.h

#include <afxwin.h>
#include "CShapesWin.h"

class CShapesApp : public CWinApp
{
    public:
        BOOL InitInstance ();
};

// File: CShapesApp.cpp

#include <afxwin.h>
#include "CShapesApp.h"

BOOL CShapesApp::InitInstance ()
{
    m_pMainWnd = new CShapesWin();
    m_pMainWnd->ShowWindow (m_nCmdShow);
    m_pMainWnd->UpdateWindow ();

    return TRUE;
}

CShapesApp shapesApp;

// File: CShapesWin.h

#include <afxwin.h>
#include "CShapesDocument.h"

class CShapesWin : public CFrameWnd
{
    public:
        CShapesWin ();
        afx_msg void OnPaint ();
        afx_msg void OnKeyDown( UINT nChar, UINT nRepCnt, UINT nFlags );
    private:
        CShapesDocument m_doc;
        DECLARE_MESSAGE_MAP ()
};

// File: CShapesWin.cpp

#include <afxwin.h>
#include "CShapesWin.h"
#include "CRectDialog.h"
#include "shapeIds.h"

CShapesWin::CShapesWin ()
{
    Create (NULL, "DrawShapes");
}

```

```

afx_msg void CShapesWin::OnPaint ()
{
    CPaintDC dc (this);
    CRect rect;
    GetClientRect (&rect);
    m_doc.Paint (dc, rect);
}

afx_msg void CShapesWin::OnKeyDown( UINT nChar, UINT nRepCnt, UINT nFlags )
{
    switch (nChar)
    {
        case 37: // Left arrow key
        case 38: // Up arrow key
        case 39: // Right arrow key
        case 40: // Down arrow key
            {
                CRectDialog rectDialog;
                if (rectDialog.DoModal() == IDOK)
                    if (m_doc.Add (new CRectangle (rectDialog.m_nHeight,
                                                    rectDialog.m_nWidth, rectDialog.m_Color)) == TRUE)
                        Invalidate (TRUE);
                break;
            }
        default:
            MessageBox ("Key not recognized");
    }
}

BEGIN_MESSAGE_MAP (CShapesWin, CFrameWnd)
    ON_WM_PAINT ()
    ON_WM_KEYDOWN( )
END_MESSAGE_MAP ()

// File:  CShapesDocument.h

#include <afxwin.h>
#include "CPolygon.h"
#include "LList2.h"

const int c_nMax = 16;

class CShapesDocument : public CDocument
{
public:
    CShapesDocument ();
    void Paint (CPaintDC & dc, CRect & winArea);
    BOOL Add (CRectangle * rect);
    CRectangle * m_rectangles [16];
    int m_nRCount;
    int m_nRows;
    int m_nCols;
};

```

```

// File: CShapesDocument.cpp

#include "CShapesDocument.h"

CShapesDocument::CShapesDocument ()
{
    m_nRows = 4;
    m_nCols = 4;

    m_nRCount = 0;
    for (int i = 0; i < c_nMax; i++)
    {
        m_rectangles[i] = NULL;
    }

    // For testing
    m_rectangles [0] = new CRectangle (10, 20, RGB(255,0,0));
    m_rectangles [1] = new CRectangle (30, 40, RGB(255,0,0));
    m_rectangles [2] = new CRectangle (50, 20, RGB(0,255,0));
    m_rectangles [3] = new CRectangle (70, 40, RGB(0,255,0));
    m_rectangles [4] = new CRectangle (90, 20, RGB(0,0,255));
    m_rectangles [5] = new CRectangle (110, 40, RGB(0,0,255));
    m_nRCount = 6;
}

BOOL CShapesDocument::Add (CRectangle * rect)
{
    if (m_nRCount >= c_nMax)
        return FALSE;
    m_rectangles [m_nRCount] = rect;
    m_nRCount++;
    return TRUE;
}

void CShapesDocument::Paint (CPaintDC & dc, CRect & winArea)
{
    int halfRow = winArea.Height() / (2 * m_nRows);
    int halfCol = winArea.Width() / (2 * m_nCols);
    int p = 0;
    for (int r = 0; p < m_nRCount && r < m_nRows; r++)
        for (int c = 0; p < m_nRCount && c < m_nCols; c++)
            m_rectangles[p++]->Paint (dc, (2 * c + 1) * halfCol, (2 * r
+ 1) * halfRow);
}

// File: shapeIds.h

#define IDC_OK          2000
#define IDC_Cancel     2001
#define IDC_Height     2002
#define IDC_Width      2003
#define IDC_Red        2004
#define IDC_Yellow     2005
#define IDC_Blue       2006

```

```

// File: Rectangle.rc

#include <afxres.h>
#include "shapeIds.h"

Rectangle DIALOG 50,50,130,130
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU

CAPTION "Create Rectangle"
{
    LTEXT        "Enter height", IDC_STATIC, 20, 15, 50, 8
    EDITTEXT     IDC_Height, 20, 25, 60, 16
    LTEXT        "Enter width", IDC_STATIC, 20, 45, 50, 8
    EDITTEXT     IDC_Width, 20, 55, 60, 16
    GROUPBOX     "Select color", IDC_STATIC, 20, 75, 50, 50
    AUTORADIOBUTTON "Red", IDC_Red, 30, 60, 50, 16, WS_GROUP
    AUTORADIOBUTTON "Yellow", IDC_Yellow, 30, 70, 50, 16
    AUTORADIOBUTTON "Blue", IDC_Blue, 30, 80, 50, 16
    PUSHBUTTON   "OK", IDC_OK, 20,100,30,15, NOT WS_TABSTOP
    PUSHBUTTON   "Cancel", IDC_Cancel, 80,100,30,15, NOT WS_TABSTOP
}

// File: CRectDialog.h

#include <afxwin.h>

class CRectDialog : public CDialog
{
public:
    CRectDialog ();
    afx_msg void OnOK ();
    afx_msg void OnCancel ();
    int m_nHeight;
    int m_nWidth;
    COLORREF m_Color;
private:
    DECLARE_MESSAGE_MAP ()
};

// File: CRectDialog.cpp

#include "CRectDialog.h"
#include "shapeIds.h"

const int TEXT_MAX = 20;

CRectDialog::CRectDialog () : CDialog ("Rectangle")
{
    m_nHeight = 0;
    m_nWidth = 0;
}

```

```

afx_msg void CRectDialog::OnOK ()
{
    char editText [TEXT_MAX + 1];
    CEdit * heightEdit = (CEdit *) (GetDlgItem (IDC_Height));
    heightEdit->GetWindowText (editText, TEXT_MAX);
    m_nHeight = atoi (editText);
    if (m_nHeight <= 0)
    {
        EndDialog (!IDOK);
        return;
    }
    CEdit * widthEdit = (CEdit *) (GetDlgItem (IDC_Width));
    widthEdit->GetWindowText (editText, TEXT_MAX);
    m_nWidth = atoi (editText);
    if (m_nWidth <= 0)
    {
        EndDialog (!IDOK);
        return;
    }
    int color = GetCheckedRadioButton (IDC_Red, IDC_Blue);
    switch (color)
    {
    case IDC_Red:
        m_Color = RGB (255, 0, 0);
        break;
    case IDC_Yellow:
        m_Color = RGB (255, 255, 0);
        break;
    case IDC_Blue:
        m_Color = RGB (0, 0, 255);
        break;
    default:
        m_Color = RGB (255, 255, 255);
    }
    EndDialog (IDOK);
}

afx_msg void CRectDialog::OnCancel ()
{
    m_nHeight = 0;
    m_nWidth = 0;
    EndDialog (!IDOK);
}

BEGIN_MESSAGE_MAP (CRectDialog, CDialog)
    ON_COMMAND (IDC_OK, OnOK)
    ON_COMMAND (IDC_Cancel, OnCancel)
END_MESSAGE_MAP ()

```

```

// File: CPolygon.h

#ifndef CPolygon_H
#define CPolygon_H

#include <iostream>
#include <afxwin.h>

using namespace std;

class CPolygon
{
public:
    CPolygon ();
    CPolygon (const CPolygon & p);
    ~CPolygon ();
    CPolygon & operator = (const CPolygon & p);
    virtual int Sides () const { return 0; }
    virtual double Area () const { return 0; }
    virtual double Perimeter () const { return 0; }
    friend ostream & operator << (ostream & outs, const CPolygon & P);
    virtual void Paint (CPaintDC & dc, int midX, int midY) {}
protected:
    double side;
    COLORREF color;
private:
};

class CQuadrilateral : public CPolygon
{
public:
    CQuadrilateral ();
    CQuadrilateral (const CQuadrilateral & p);
    ~CQuadrilateral ();
    CQuadrilateral & operator = (const CQuadrilateral & p);
    int Sides () const;
private:
};

class CRectangle : public CQuadrilateral
{
public:
    CRectangle ();
    CRectangle (double L, double W);
    CRectangle (double L, double W, COLORREF C);
    ~CRectangle ();
    CRectangle (const CRectangle & p);
    CRectangle & operator = (const CRectangle & p);
    double Area () const;
    double Perimeter () const;
    void Paint (CPaintDC & dc, int midX, int midY);
private:
    double side2;
};

#endif

```

```

// File: CPolygon.cpp

#include <iostream>
#include <cmath>
#include "CPolygon.h"
using namespace std;

#define M_PI 3.14159

CPolygon::CPolygon ()
{
    side = 0;
    color = RGB (255, 0, 0);
}

CPolygon::CPolygon (const CPolygon & p)
{
    side = p.side;
    color = p.color;
}

CPolygon::~CPolygon ()
{
}

CPolygon & CPolygon::operator = (const CPolygon & p)
{
    side = p.side;
    return * this;
}

ostream & operator << (ostream & outs, const CPolygon & P)
{
    outs << "The area of your " << P.Sides()
        << " sided CPolygon is " << P.Area ()
        << "; its perimeter is " << P.Perimeter() << ".\n";
    return outs;
}

CQuadrilateral::CQuadrilateral ()
{
}

CQuadrilateral::CQuadrilateral (const CQuadrilateral & p)
{
}

CQuadrilateral::~CQuadrilateral ()
{
}

CQuadrilateral & CQuadrilateral::operator = (const CQuadrilateral & p)
{
    return * this;
}

```

```

int CQuadrilateral::Sides () const
{
    return 4;
}

CRectangle::CRectangle ()
{
    side = 0;
}

CRectangle::CRectangle (double L, double W)
{
    side = L;
    side2 = W;
}

CRectangle::CRectangle (double L, double W, COLORREF C)
{
    side = L;
    side2 = W;
    color = C;
}

CRectangle::~~CRectangle ()
{
}

CRectangle::CRectangle (const CRectangle & p)
{
    side = p.side;
    side2 = p.side2;
    color = p.color;
}

CRectangle & CRectangle::operator = (const CRectangle & p)
{
    side = p.side;
    side2 = p.side2;
    color = p.color;
    return * this;
}

double CRectangle::Area () const
{
    // Calculate and return area here
    return side * side2;
}

double CRectangle::Perimeter () const
{
    // Calculate and return perimeter here
    return 2 * side + 2 * side2;
}

```

```
void CRectangle::Paint (CPaintDC & dc, int midX, int midY)
{
    CBrush colorBrush;
    colorBrush.CreateSolidBrush (color);
    CBrush * pBrushSv = dc.SelectObject (&colorBrush);
    dc.Rectangle (midX - side2 / 2, midY - side / 2,
                 midX + side2 / 2, midY + side / 2);
    dc.SelectObject (pBrushSv);
}
```