ET++ Application: PolyDocApp

Prepared by: Rana Sen

Introduction

This application shows how to handle several document types within one application. As illustrative examples, a document with text and another document with graphic items have been used. The user can copy a graphic item from the latter and paste it on the text document. Files with ASCII text can also be pasted on the text document. Other functionalities include:

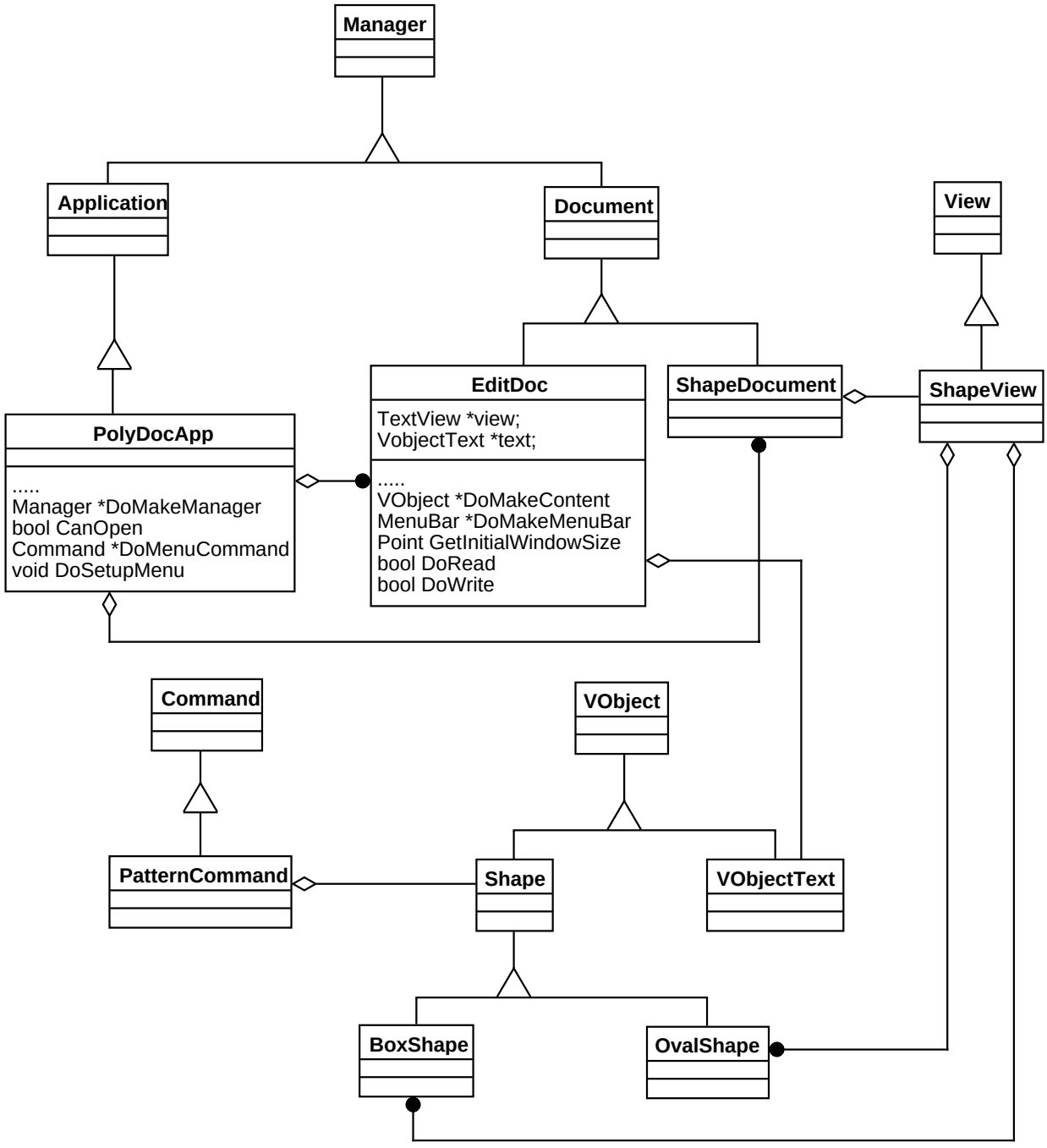
- Opening other compatible documents.
- Performing available operations between documents, like cutting and pasting, importing text files, etc.
- Saving an edited document.

Snap Shots

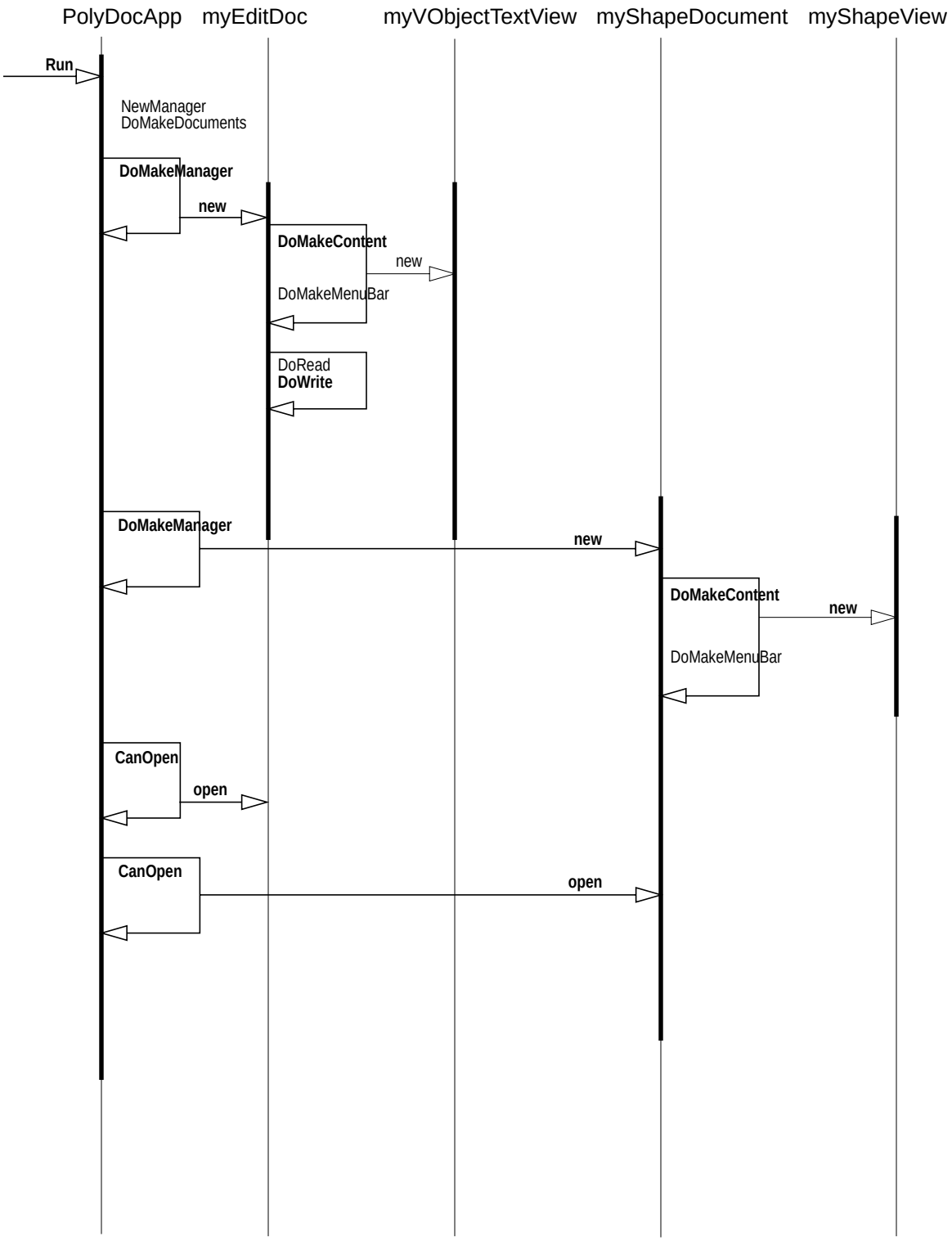
(a)
View of the text document window and the initial graphics window.

(b)
View of window showing an edited document: the oval shape has been copied to the text document. Another text file also has been added.

Object Model Diagram



How it all happens (I think)!



Source Code

PolyDocApp.C

```
#include “ET++.h”
//#include “DialogItems.h”
#include “Data.h”

#include “EditDoc.h”
#include “ShapeDocument.h”
#include “PolyCmdNo.h”

//---- PolyDocApp ----------------------------------------------------------

class PolyDocApp: public Application {
public:
    MetaDef(PolyDocApp);
    PolyDocApp(int argc, char **argv) : Application(argc, argv, cVobDocType)
{ }
    Manager *DoMakeManager(Symbol);
    bool CanOpen(Data *data);

    Command *DoMenuCommand(int);
    void DoSetupMenu(Menu *m);
};

NewMetaImpl0(PolyDocApp,Application);

void PolyDocApp::DoSetupMenu(Menu *m)
{
    Application::DoSetupMenu(m);
    m->EnableItem(cNEWSHAPES);
}

Command *PolyDocApp::DoMenuCommand(int cmd)
{
    if (cmd == cNEWSHAPES)
NewManager(cDocTypeShapes);
    else
Application::DoMenuCommand(cmd);
    return gNoChanges;
}
```

Creating new documents

```
Manager *PolyDocApp::DoMakeManager(Symbol type)
{
    if (type == cDocTypeShapes)
return new ShapeDocument;
    if (type == cVobDocType)
return new EditDoc;
    return 0;
}

bool PolyDocApp::CanOpen(Data *data)
{
    return (data->Type() == cVobDocType) || (data->Type() == cDocTypeShapes);
}

//---- main ----------------------------------------------------------------
```

```
main(int argc, char *argv[])
{
    return PolyDocApp(argc, argv).Run();
}
```

EditDoc.C

```
#include “ET++.h”

#include “VObjectTView.h”
#include “VObjectText.h”
#include “EditDoc.h”
#include “Data.h”
#include “About.h”
#include “PolyCmdNo.h”

char *cVobDocType= “VOBTEXT”;

//---- EditDoc -----------------------------------------------------------------

NewMetaImpl(EditDoc,Document, (TP(view), TP(text)));

EditDoc::EditDoc() : Document(cVobDocType)
{
    text= new VObjectText(1024, new_Font(eFontTimes, 12));
    text->SetFString(aboutMsg);
}
```

Initial text doc

```
EditDoc::~EditDoc()
{
    SafeDelete(view);
    SafeDelete(text);
}

VObject *EditDoc::DoMakeContent()
{
    view= new VObjectTextView(this, Rectangle(380, cFit), text);
    SetFirstHandler(view);
    return new Scroller(view);
}

MenuBar *EditDoc::DoMakeMenuBar()
{
    MenuBar *mb= Document::DoMakeMenuBar();

    Menu *m= mb->FindMenu(cFILEMENU);
    m->ReplaceItem(cNEW, “New Text”);
    m->InsertItemAfter(cNEW, “New Shapes”, cNEWSHAPES);

    mb->AddMenu(TextView::MakeMenu(cFONTMENU));
    mb->AddMenu(TextView::MakeMenu(cSTYLEMENU));
    mb->AddMenu(TextView::MakeMenu(cSIZEMENU));
    mb->AddMenu(TextView::MakeMenu(cFORMATMENU));
    return mb;
}
```

Augmenting menu

```
Point EditDoc::GetInitialWindowSize()
{
    return Point(420, 300);
}

bool EditDoc::DoRead(IStream &s, Data *data)
{
    Document::DoRead(s, data);
    VObjectText *t= 0;
    s >> t;
    if (t) {
text= t;
text->SetView(view);
Text *oldtext= view->SetText(text);
SafeDelete(oldtext);
return TRUE;
    }
    return FALSE;
}

bool EditDoc::DoWrite(OStream &s, Data *d)
{
    Document::DoWrite(s, d);
    s << text;
    return TRUE;
}
```

ShapeDocument.C

```
#include “ET++.h”
#include “Alert.h”
#include “ShapeDocument.h”
#include “ShapeView.h”
#include “PolyCmdNo.h”

//---- ShapeDocument ------------------------------------------------------------

Symbol cDocTypeShapes(“TWOSHAPES”);

NewMetaImpl(ShapeDocument,Document, (TP(view)));

ShapeDocument::ShapeDocument() : Document(cDocTypeShapes)
{
}

ShapeDocument::~ShapeDocument()
{
    SafeDelete(view);
}

bool ShapeDocument::DoWrite(OStream &s, Data *data)
{
    Document::DoWrite(s, data);
    view->PrintOn(s);
    return TRUE;
}
```

```
bool ShapeDocument::DoRead(IStream &from, Data *data)
{
    Document::DoRead(from, data);
    view->ReadFrom(from);
    return TRUE;
}

VObject *ShapeDocument::DoMakeContent()
{
    view= new ShapeView(this, Point(600));
    SetFirstHandler(view);
    return new Splitter(view);
}
```

Augmenting menu

```
MenuBar *ShapeDocument::DoMakeMenuBar()
{
    MenuBar *mb= Document::DoMakeMenuBar();

    Menu *m= mb->FindMenu(cFILEMENU);
    m->ReplaceItem(cNEW, "New Text");
    m->InsertItemAfter(cNEW, "New Shapes", cNEWSHAPES);

    mb->AddMenu(ShapeView::MakePatternMenu());
    return mb;
}

Point ShapeDocument::GetInitialWindowSize()
{
    return Point(400, 400);
}
```

ShapeView.C

```
#include "ShapeView.h"
#include "ShapeDocument.h"
#include "ShapeCommands.h"
#include "Shapes.h"

#include "Class.h"
#include "Menu.h"
#include "ClipBoard.h"
#include "OrdColl.h"
#include "PolyCmdNo.h"

//---- PatternMenuItem -------------------------------------------------------

NewMetaImpl(PatternMenuItem,VObject, (TP(ink)));

PatternMenuItem::PatternMenuItem(int id, Ink *p) : VObject(id)
{
    ink= p;
    SetExtent(Point(50, 20));
}

void PatternMenuItem::Draw(Rectangle)
{
    GrPaintRect(contentRect.Inset(3), ink);
```

```
}

//---- ShapeView -----------------------------------------------------------

OrdCollection *ShapeView::palette;
int ShapeView::refcnt;

NewMetaImpl(ShapeView,View, (TP(shapes)));
```

Default shapes

```
ShapeView::ShapeView(Document *d, Point ext) : View(d, ext)
{
    Shape *sp;

    RefPalette();

    shapes= new OrdCollection;
    sp= new OvalShape(Rectangle(50, 50, 80, 80), (Ink*)palette->At(8));
    shapes->Add(sp);
    sp= new BoxShape(Rectangle(100, 100, 100, 100), (Ink*)palette->At(3));
    shapes->Add(sp);
    sp= new OvalShape(Rectangle(150, 150, 100, 100), (Ink*)palette->At(4));
    shapes->Add(sp);
    sp= new BoxShape(Rectangle(200, 200, 80, 120), (Ink*)palette->At(5));
    shapes->Add(sp);
    sp= new OvalShape(Rectangle(250, 250, 80, 120), (Ink*)palette->At(7));
    shapes->Add(sp);
    shapes->ForEach(VObject,SetContainer)(this);

    selection= 0;
}

ShapeView::~ShapeView()
{
    if (shapes) {
shapes->FreeAll();
SafeDelete(shapes);
    }
    UnRefPalette();
}

void ShapeView::RefPalette()
{
    if (refcnt == 0) {
palette= new OrdCollection;
palette->Add(ePatWhite);
palette->Add(new_Grey(0.75));
palette->Add(new_Grey(0.5));
palette->Add(new_Grey(0.25));
palette->Add(ePatBlack);
palette->Add(ePat00);
palette->Add(ePat01);
palette->Add(ePat02);
palette->Add(new_RGBColor(255, 0, 0));
palette->Add(new_RGBColor(0, 255, 0));
palette->Add(new_RGBColor(0, 0, 255));
    }
    refcnt++;
}
```

```
void ShapeView::UnRefPalette()
{
    refcnt--;
    if (refcnt <= 0)
SafeDelete(palette);
}

void ShapeView::Draw(Rectangle r)
{
    shapes->ForEach(VObject,Draw)(r);
    if (! gPrinting && selection != 0)
selection->Highlight(On);
}

Command *ShapeView::DoLeftButtonDownCommand(Point p, Token t, int cl)
{
    Iter previous(shapes->MakeReversedIterator());
    Shape *shape, *s= 0;

    while (shape= (Shape*) previous()) {
if (shape->ContainsPoint(p)) {
    s= shape;
    break;
}
    }

    if (s) {
if (t.Flags & eFlgCntlKey)
    return s->GetStretcher();
return s->GetMover();
    }
    return View::DoLeftButtonDownCommand(p, t, cl);
}

IStream &ShapeView::ReadFrom(IStream &s)
{
    selection= 0;
    shapes->FreeAll();
    SafeDelete(shapes);
    s >> shapes;
    shapes->ForEach(VObject,SetContainer)(this);
    ForceRedraw();
    return s;
}

OStream &ShapeView::PrintOn(OStream &s)
{
    return s << shapes NL;
}
```

Copy graphic to clipboard

```
Command *ShapeView::DoMenuCommand(int cmd)
{
    if (cmd == cCOPY) {
gClipBoard->SelectionToClipboard(shapes->At(0));
return gNoChanges;
    }
    if (cmd >= cSETINK && cmd <= cSETINK+palette->Size())
```

```
return new PatternCommand((Shape*)shapes->At(0),
    (Ink*)palette->At(cmd-cSETINK));
    return View::DoMenuCommand(cmd);
}
```

```
Menu *ShapeView::MakePatternMenu()
{
    Iter next(palette);
    Ink *ink;
    Menu *m= new Menu(“Pattern”, FALSE);
    int i= 0;
    while (ink= (Ink*) next()) {
m->Append(new PatternMenuItem(cSETINK+i, ink));
i++;
    }
    return m;
}
```

Pattern menu

```
//---- clipboard

bool ShapeView::HasSelection()
{
    return TRUE;
}

Command *ShapeView::PasteData(Data *data)
{
    VObject *vop= (VObject*) data->AsObject(Meta(VObject));
    if (vop)
shapes->Add(vop);
    return gNoChanges;
}

bool ShapeView::CanPaste(Data *data)
{
    return data->CanConvert(Meta(VObject));
}
```