講義「Visual C++を用いる画像処理」のための教材制作

松田, 修三 / 森下, 巖 / MATSUDA, Shuzo / MORISHITA, Iwao

(出版者 / Publisher)
法政大学計算科学研究センター
(雑誌名 / Journal or Publication Title)
Bulletin of Computational Science Research Center, Hosei University / 法政大学計算科学研究センター研究報告
(巻 / Volume)
12
(開始ページ / Start Page)
103
(終了ページ / End Page)
108
(発行年 / Year)
1999-03-31
(URL)
https://doi.org/10.15002/00024835

講義「Visual C++を用いる画像処理」のための教材制作

森下　巖　　松田　修三
法政大学工学部システム制御工学科

講義「Visual C++を用いる画像処理」のために制作した教材について報告する。学生自身が画像処理アルゴリズムをプログラミングして処理結果を検討できるように、画像処理の講義にプログラミングの教育を取り入れた。Visual C++には、高レベルの機能を備えていて Windows 上で動作するプログラムを容易に作成できる、本学の学生はだれでもキャンパス内の設備で利用できる、などの利点がある一方、非常に大規模なものであるために、初心者にはとっつきにくいという欠点がある。本文では、初心者にも理解が容易になるように構成した教材の一例を報告する。教材は、プリントして学生に配布する講義ノートと、教室でビデオ・プロジェクターを用いてデモンストレーションするプログラム例で構成されている。この教材は数回使用したが、その教育効果の組織的な評価はまだ試みていない。

1．はじめに

画像処理の教育においては、学生自身が処理アルゴリズムをプログラミングして実行結果を検討する演習が重要である。そこで、画像処理の講義に Visual C++を用いるプログラミングを取り入れることにした。プログラミング言語として Visual C++を採用したのは、高レベルの機能を備えていて Windows 上で動作するプログラムが容易に作成できる、一般に実行速度の速いプログラムが得られる、本学部の学生はだれでもキャンパス内の設備で利用できる、などの利点があるためであるが、一方では、非常に大規模なものであるために、初心者にはとっつきにくいという欠点がある。講義を進める際に、どのような内容をどのような順序で提示するかについて、慎重に検討しなければならない。

本文では、その講義のために制作した教材の一例を報告する。制作したのは、プリントして学生に配布するための講義ノートと、教室においてデモンストレーションするためのプログラム例である。後者では、完成したプログラムを提示して実行させるだけでなく、Visual C++用の開発ツール Microsoft Developer Studio を用いてプログラムを作成していく過程をも示す。

最近では Visual C++ に関する教科書、解説書としてもいろいろのものが出版されているが[1]-[3]、C の初歩を学習しただけの学生でも理解できるレベルのものは少なく、内容が画像処理に重点を置いたものはさらに少ない。今回、新しい教材の制作を試みたのはこのためである。

2．受講学生の特性と講義の方針

対象とする学生は、システム制御工学を専攻する学部４年生であり、センシング、回路、動的システム、制御、信号処理などに強い興味を持つものである。コンピュータの使用法についてはすでに習熟しているが、プログラミングについては C の基礎を学習しただけになっている。また，OS の基本的なメカニズムについて組織的な教育を受けているわけではない。

対象とする学生がソフトウエア工学を専攻するものである場合には、オブジェクト指向プログラミングやイベント駆動プログラミングについて、基礎から系統的に講義していくべきである。しかし、この場合、受講者は応用指向であり、講義の主目的も画像処理手法であるので、理論的な取り扱いはすべて省略し、ともかく記述しやすいプログラム例をつぎつぎに提示することによって、プログラミングに慣れさせていくことにした。画像処理用に限定すれば、この方法でも Visual C++ のプログラムをかなりの程度まで書けるようになると考えるからである。ある程度経験を積んだ後では、言語 C++ の解説書[4]を自習するのも容易になるし、Visual C++ のより高度な技法も徐々にマスターしていけると思う。

Visual C++ については、あまりにブラックボックス化されているので、入門者用には有害であるという意見[5]がある。たしかに、学部１、２年生用に使用するのは不適当である。しかし、卒業研究の手段として使用するのは奨励してよいと思う。実用ソフトウエアを開発するための巨大なツールを使用する経験は、将来どのような仕事に携わるにせよ、役立つはずである。

3．講義の構成

一学期間の講義であるから、内容はごく基本的なものに限定しなければならない。採用したのは下記の構成である。

1) Visual C++とは
2) C++のクラスをどのように利用するか
3) Microsoft Developer Studio (MDS) によるプログラム骨組みの自動生成
4) ビットマップ・カラー画像の表示
5) 線図とテキストの表示
6) ビットマップ画像データの配列変数への格納、処理結果のビットマップ画像への格納と表示
7) ビットマップ画像ファイルの読み出しと書き込み
8) 直列化による変数データのファイルへの書き込みと読み出し
9) 表示ウインドウへのメニュー項目の追加とその項目選択による処理の記述
10) ビットマップ画像の処理例：
濃淡画像化、フィルタリング、ヒストグラム作成
11) ダイアログボックスによるユーザー操作の実現
12) ダイアログボックスによるユーザー入力と結果の表示の実現
13) その他のプログラム例

4．主要内容の説明

4.1　序論部 1)、2)、3)

まず、C と C++と Visual C++との違いを説明する。C++は、記述法は C のものを踏襲しているが、機能的にはクラスと呼ぶ新しい要素を追加して積極的に利用する方式であること、C++自体は汎用性のある言語であるが、Visual C++はマイクロソフト社が提供するクラスの集合（Microsoft Foundation Class Library）を利用してプログラミングしているので、MDS を使用しないことにはプログラムを作成することも修正することもできないものであること、しかし、ウインドウ上で動作するプログラムを作成するには非常に便利であること、など。

つぎに、C++のクラスのもっとも初歩的な使用法を説明する。メンバー変数、メンバー関数、コンストラクターなどによってクラスを定義すること、定義ずみのクラスについてはその一実現例を自由に宣言して使用できること、クラスの定義では宣言部と実装部を別個のファイルとして構成するのが便利であること、など。

第３に、MDS を用いることにより、Windows 上で動作するプログラムの骨組みを自動的に生成できることを説明する。これはデモンストレーションを見せるのがわかりやすい。MDS を起動して一連の操作を行い、一行もコードを書かずに骨組みとなるプログラムを作成し、これをコンパイルし、ビルドし、実行させる。標準的な機能のウインドウが画面に表示され、そのメニューの項目も正しく機能することを見せる。

また、作成されたプログラムには、多数のクラスが使用され、多数のヘッダ・ファイルやソース・ファイルが生成され、それらに意味不明の長いコードが記述されていることを示す。同時に、さしあたりはこの長いコードを読んで理解する必要が全くないことを強調する。

さらに、MDS の画面で Visual C++に関する詳細な情報を見ることができることを示す。Microsoft Foundation Class Library に用意されている膨大な数のクラス、各クラスごとに用意されている多数のメンバー関数、その各メンバー関数の機能、引数、返値などを表示して見せ、かつ、これらを一度に学習して記憶する必要がないことを強調する。英語の単語や構文を少しづつ勉強してきたように、必要に応じて少しづつ記憶していけばよいのである。

4.2　カラー画像、線図、テキストの表示 4)、5)、6)

4.2.1　ビットマップ画像のもっとも原始的な表示法

まず、ディスプレイ表示用のビデオ・ユニットにはいくつかの表示モードがあること、表示モードをツルーカラーとした場合には、１画素のデータを

R 成分、G 成分、B 成分：各８ビット、
予備成分：８ビット、

の合計３２ビットで表現していること、この講義では、プログラミングを簡単にするために、ビデオ・ユニットはツルーカラー・モードに設定されているものと仮定してコードを書くことを説明する。ただし、ビデオ・ユニットによってはツルーカラー・モードを２４ビットで表現するものもあり（予備成分の省略）、この場合にはコードの一部を変更する必要があることを注意しておく。

図１．プログラム BITMAP の実行例

リスト１．画像表示のための追加コード

```
CDC dcMemory;
CBitmap myBitmap;
BITMAP bm;
myBitmap.LoadBitmap(IDB_BITMAP1);
dcMemory.CreateCompatibleDC(pDC);
dcMemory.SelectObject(&myBitmap);
myBitmap.GetBitmap(&bm);
pDC->BitBlt(0, 0,
            bm.bmWidth, bm.bmHeight,
            &dcMemory,
            0, 0,
            SRCCOPY);
```

つぎに、カラー・ビットマップ画像をウインドウに表示するもっとも簡単なプログラム例を提示する。これは、プログラムを実行させると、親ウインドウのなかに１個の子ウインドウが開かれ、そこに用意したビットマップ画像が表示されるものである（図１）。MDS を用いて、このプログラムが以下のステップで作成できることをデモンストレーションする。

1) プロジェクト名を BITMAP としてプログラムの骨組みを自動生成する。
2) MDS のメニュー「挿入」のなかにある項目「リソース」を選択し、表示させたいビットマップ画像をハードディスクの格納位置からインポートさせる。このリソースには、自動的に、
   IDB_BITMAP1
   という名称が付けられる。
3) 自動生成されたファイルの一つ CBITMAPView.cpp のなかに描画のための関数

```
void CBITMAPView::OnDraw(CDC* pDC)
{
    CBITMAPDoc* pDoc=GetDocument();
    ASSERT_VALID(pDoc);
    // TODO この場所に描画コードを追加
}
```

が用意されている。その指定場所にリスト１のコードを記述する。

これで完成であり、コンパイルし、ビルドし、実行させる。マウスで子ウインドウの位置やサイズを自由に変更できること、また、メニュー「ファイル」の項目「新規作成」を選択することにより、同じビットマップ画像の子ウインドウをつぎつぎと生成できること、メニュー「ウインドウ」の項目も正しく機能すること、などを示す。

リスト１のコードは短いものであるが、これについては、かなり詳細に説明しなければならない。

まず、MDS を用いて、プロジェクト名 XXX でプログラムの骨組みを自動生成すると、表示の機能を担当するクラス CView の子クラスである CXXXView が作成され、そのソースファイル XXXView.cpp のなかには描画用の関数 OnDraw が用意されていること、OS はウインドウの表示の更新が要求されるとこの関数を呼ぶこと、したがって、プログラマーが実行させたい描画コードはこの関数の中に追加すればよいことを説明する。

問題は描画コードである。Visual C++では、ディスプレイの表示面やプリンタの印字面のように、描画を出力する対象(「紙面」「キャンパス」のようなもの)を device context(DC)と呼び、これを表現するためのクラス CDC を用意していること、また、その「紙面」「キャンパス」に何かを描くためのツール（「ペン」「ブラシ」のようなもの）を graphics device interface(GDI)と呼び、これを表現するためのクラス、

CPen: 線図ドロー用
CBrush: 領域塗りつぶし用
CFont: 文字印字用
CBitmap: ビットマップ画像ペイント用

を用意していること、さらに、ビットマップ画像の構成を記述するためのストラクト

BITMAP

を用意していることを説明する。

また、MDS で作成されたプログラムにはディスプレイを表現する CDC のインスタンス１個と、プリンタを表現する CDC のインスタンス１個が自動的に用意されていること、そのディスプレイ用のものは、これを指すポインターpDC が引数として関数 OnDraw に与えられていること、ディスプレイに描画するコードはこの引数を用いてを記述すればよいことを説明する。

リスト１の描画コードに用いたクラス、そのインスタンス、メンバー関数、および、ストラクトは表１に示したものである。ビットマップ画像データは大量であるので、これをハードディスクから読み出しながらビデオ・ユニットに転送するのには問題があり、ディスプレイの DC と同一構成の DC をダミーとして主メモリ上に構成し（これが dcMemory)、これにビットマップ画像をペイントし、その内容を関数 BitBlt を用いてビデオ・ユニットにコピーする方式を使用している。

4.2.2 ビットマップ画像の実用的な表示法

上記はもっとも簡単になるプログラミングであるが、その動作には問題がある。それは、関数 OnDraw のなかに

```
CBitmap myBitmap;
myBitmap.LoadBitmap(IDB_BITMAP1);
```

を記述していることである。これは、プログラムの

表１． 描画コードに使用するクラスなど

クラス/ストラクト	インスタンス	メンバー関数
CDC	dcMemory	CreateCompatibleDC
		SelectObject
		BitBlt
Bitmap	myBitmap	LoadBitmap
		GetBitmap
BITMAP	bm	

リスト２． 画像表示のための追加コード、改良版ヘッダー・ファイルとソース・ファイルへの追加

```
// BITMAP2View.h
//
class CBITMAP2View:public CView
{
private:                    // 追加記入
     CBitmap* pMyBitmap; // 追加記入
                            // 以下略

// BITMAP2View.cpp
//
CBITMAP2View::CBITMAP2View()
{
     // 以下２行コンストラクタへ追加記入
     pMyBitmap = new CBitmap;
     pMyBitmap->LoadBitmap(IDB_BITMAP1);
}
CBITMAP2View::~CBITMAP2View()
{
     // 以下１行デストラクタへ追加記入
     delete pMyBitmap;
}
void CBITMAP2View::OnDraw(CDC* pDC)
{
     // 以下６行描画コード追加記入
     CDC dcMemory;
     BITMAP bm;
     dcMemory.CreateCompatibleDC(pDC);
     dcMemory.SelectObject(pMyBitmap);
     pMyBitmap->GetBitmap(&bm);
     pDC->BitBlt(0, 0,
                 bm.bmWidth, bm.bmHeight,
                 &dcMemory,
                 0, 0,
                 SRCCOPY);
}
```

実行中、子ウインドウの描画を更新するたびにハードディスクからビットマップ画像を読み 出すことを意味し、あまりにも非能率となる。そこで、ビットマップ画像データは、プログラムの実行が終了するまで継続して保存されるメモリ領域に格納することにする。それには、リスト２（プロジェクト名 BITMAP2）に示すように、画

像データをスタック上ではなくヒープ領域に格納することにすればよい。

ヒープのメモリ割り当ての要求には、演算子 new を使用し、割り当てたメモリを回収するには、演算子 delete を使用すること、ヘッダー・ファイル CBITMAP2View.h において CBitmap のインスタンスをメンバー変数として宣言すること（ポインター型のものとして宣言）、コンストラクタでこのメンバー変数にメモリ割り当てを要求し、これにビットマップ画像を書き込むこと、デストラクタでこのメモリを回収すること、などを説明する。

4.2.3 線図、テキストの表示

ディスプレイ画面を記述する座標系を説明し、クラス CPen で使用するペンの設定を行い、CDC のメンバー関数 MoveTo(x1, x2)と LineTo(x2, y2)によって線分の描画を行うこと、また、クラス CFont とメンバー関数 CreatePointFont/CreateFont で使用するフォントの設定を行い、CDC のメンバー関数 TextOut によって文字列の表示を行うことを説明する。

4.2.4 画像データの処理と処理結果の表示

表示されているビットマップ画像に処理を加える場合には、画素データを配列に格納して処理計算を実行する。また、処理結果をビットマップ画像として表示させる場合には、配列の内容をビットマップ画像の画素データに格納してそのビットマップ画像を表示させる。

これには、CBitmap のメンバー関数

GetBitmapBits、SetBitmapBits

を使用する。具体的には、クラスのヘッダー・ファイルのなかで

```
CBitmap* pMyBitmap;
BYTE* bmData;  // 画素値データ格納用配列
long size;     // 画素値データのサイズ
```

をメンバー変数として宣言しておき、画素値データを配列に格納するには、

```
BYTE* bmData = new BYTE[size];
pMyBitmap->GetBitmapBits(size, bmData);
```

と記述し、また、逆の操作には、

```
pMyBitmap->SetBitmapBits(size, bmData);
```

と記述する。

また、Visual C++では、Document-View 構成によるプログラミングを推奨していることを説明する。MDS を用いてプロジェクト名 XXX でプログラムの骨組みを生成すると、ビュー・クラス CXXXView と共にドキュメント・クラス CXXXDoc も用意される。ドキュメント・クラスは、プログラムで使用する主要なデータをそのメンバー変数として保存するために使用する。ビュー・クラスはそのデータをディスプレイに表示するために使用する。

この方式を採用すると、表示の際にビュー・クラスからドキュメント・クラスのデータを読み出さなければならないが、これは簡単に実行できる。それは、CXXXView の関数 OnDraw の先頭で CXXXDoc を指すポインター pDoc が与えられているからである。CXXXDoc.h のなかで、CXXXDoc クラスのメンバー変数とメンバー関数を

```
private:
   CBitmap* pMyBitmap;
public:
   CBitmap* GetMyBitmap()
                  {return pMyBitmap;}
```

と宣言しておき、CXXXView.cpp のなかではその関数 OnDraw に、

```
void CXXXView::OnDraw(CDC* pDC)
   {
      CBITMAPDoc* pDoc=GetDocument();
      ASSERT_VALID(pDoc);
      // TODO この場所に描画コードを追加
      CBitmap* pBitmap =
                pDoc->GetMyBitmap();
      // 以下略
   }
```

と記述する。CXXXDoc の pMyBitmap に格納されている画像データが OnDraw で宣言した pBitmap に格納されるので、あとはこの pBitmap を用いて描画を実行することができる。

4.3 ビットマップ画像ファイルの読み出しと書き込み、直列化による変数データのファイルへの書き込みと読み出し 7), 8)

4.3.1 ファイルの読み出し

画像処理ではディスクに格納してあるビットマップ画像を読み出して処理し、処理結果をまたビットマップ画像としてディスクに格納することが多い。そのビットマップ画像はさまざまなファイル形式でディスクに格納されている。この講義では、対象を Microsoft Windows Bitmap ファイル形式のものに限定し、読み出しを行うプログラムと書き込みを行うプログラムを説明する（プロジェクト名は BINARY）。このファイルの読み出し、書き込みを担当するのはドキュメント・クラスである。

読み出しを実行したいとき、プログラムのユーザーはウインドウのメニュー「ファイル」の項目「開く」を選択する。するとダイアログボックスが開かれるので、ユーザーはディレクトリとファイル名を指定して OK ボタンをクリックする。これで指定したファイルがディスクから読み出されてディスプレイに表示されなければならない。Windows は、「ファイル」の「開く」が選択されると、ドキュメント・クラスのメンバー関数である

OnOpenDocument

を実行することになっているので、読み出しのコードはこの関数のなかに記述する。そこで、MDS の ClassWizard を用いて CBINARYDoc.cpp にこの関数を追加する。追加すると下記のコード、

```
BOOL CBINARYDoc::OnOpenDocument
                  (LPCTSTR lpszPathName)
{
   // 読み出しコードを追加記述
   return TRUE;
}
```

が現れる。引数として与えられる lpszPathName は、いま指定したファイルのパスを記述する文字列であり、この引数を使用してファイルを読み出すコードを記述する。

ファイルを表現するクラスは CFile であり、メンバー関数には、Open、Read、Write、Seek、SeekToBegin、SeekToEnd、GetPosition、などがある。

4.3.2 Microsoft Windows Bitmap ファイル形式

この形式のファイルは、ヘッダー部、カラーマップ部

（カラーパレット部)、画素データ部の３部で構成されており、そのヘッダー部には

BITMAPFILEHEADER（１４バイト）

BITMAPINFOHEADER（４０バイト）

が記述されている。まず、これを読み出してビットマップ画像の構成を知り、つぎに、画像のサイズから画素データを格納するための配列のサイズを求め、このサイズの配列を new によって要求し、これにファイルから読み出した画素データを格納する。
れを用いて画素データを読み出す。
書き込みのプログラムについても同様に説明する。

4.3.3 直列化

ドキュメント・クラスのメンバー変数に格納されているデータは、直列化（Serialization）と呼ぶ簡単な方法でファイルに書き込み、また、ファイルから読み出すことができる。これには `CArchive` と呼ぶクラスを使用する。

4.4 画像処理例 9), 10)

ビットマップ・カラー画像をファイルから読み出し、これに、濃淡画像化、フィルタリング、ヒストグラム作成、などの処理を加えることのできるプログラムを作成する。ここで、どの処理を実行させるかはウインドウのメニューに追加した項目で選択させる。MDS を用いるウインドウのメニュー項目の追加法、および、それぞれの項目の処理を指定するコマンドハンドラの記述法を説明する。

一般に、プロジェクトのリソースとしては Bitmap のほかに Menu や Dialog も使用できる。その Menu には親ウインドウ用と子ウインドウ用があり、ここでは子ウインドウ用をクリックして設計用ウインドウを表示させ、メニューの項目を指定していく。

4.5 ダイアログボックスの利用、その他 11), 12), 13)

プログラムの実行中、ユーザーに数値や文字列などのデータをキーボードから入力させたり、あるいは、処理結果の数値や文字列をディスプレイに表示させたい場合がある。それには、リソースの一つとして用意してあるダイアログボックスを利用すればよい。

まず、ダイアログボックス・エディタを起動して、ダイアログボックスの形状やそこに配置する機能要素を選択する。機能要素には、エディットボックス、押しボタン、ラジオボタンなどが利用できる。OK と CANCEL の押しボタンは最初から自動的に用意されている。

つぎに、このダイアログボックス用のクラスを作成する。これには MDS の ClassWizard を使用する。１個のダイアログボックスごとに１個のクラスを作成する。基本クラスには `CDialog` を用いてその子クラスを作成する。名称は、たとえば、`DlgForm` とする。エディットボックスがあればその内容を格納するメンバー変数を用意する。また、ラジオボタンがあればその選択番号を格納するメンバー変数を用意する。

ダイアログボックスを動作させるには、そのダイアログボックスを記述するクラスのインスタンスを宣言し、そのメンバー関数である `DoModal` を実行させる。これでダイアログボックスがディスプレイに表示され、ユーザーが入力を終了した後で押しボタン OK をクリックすると、ダイアログボックスの入力内容がビュー・クラスのメンバー変数に格納される。

5. 講義における教材の使用

講義では、講義ノートとして、毎回、B4 版用紙１枚に２ページをプリントしたもの約１０ページを配布した。全体では約１５０ページになる。

提示したプログラム例は２０プロジェクトである。これはノートブック型 PC とプロジェクターを教室に持ち込んでデモンストレーションした。受講学生は PC を持参していないので、これを見るだけである。ただし、操作の手順はプリントに詳細に記述してあるので、あとで追跡するのは容易と思われる。なお、このプログラム例は Visual C++の Version 4.0 あるいは 5.0 で動作するものである。内容が基本的な機能の範囲にあるので、6.0 でも問題ないと思うが、まだ確認してはいない。

講義は４年生に対するもので、受講者の人数が少なかったこともあり、この教材の教育効果の組織的な評価はまだ試みていない。ただ、筆者の研究室では Visual C++ の使用者が増加してきている。

6. むすび

講義「Visual C++を用いる画像処理」のために制作した教材について報告した。Visual C++は非常に大規模なものであるため、初心者には取っつきにくいという欠点がある。内容を画像処理に限定して、ごく初等的、入門敵なレベルから出発して、卒業研究で利用できる程度までカバーすることを試みたが、なお、今後も改良を続けたいと考えている。

参考文献

[1] Mark Andrews, “Learn Visual C++ Now,” Micrsoft Press, 1996.

[2] 山岡祥, “Visual C++入門,” CQ 出版, 1996.

[3] 田中正造, “Visual C++ 5.0 プログラミング,” ソフトバンク, 1998.

[4] ストラウストラップ著, 斉藤信男他訳, “プログラミング言語 C++,” トッパン, 1993.

[5] 松田洋, 新藤義昭, 椋田実, “プログラミング演習教材ソフトウエア WinTK”, 情報教育方法研究、1-1, 31/36, 1998.

キーワード.

講義教材、Visual C++プログラミング、画像処理、配布プリント、デモンストレーション・プログラム

Summary.

Production of Teaching Materials for a Class on “Image Processing by Using Visual C++”

Iwao Morishita　　Shuzo Matsuda
Department of System Control Engineering, College of Engineering, Hose University

This paper describes an example of teaching materials produced for a class on “Image Processing by Using Microsoft Visual C++.” This language was selected because of its powerfulness for implementing applications to be executed on Windows NT/95/98 machines, and also of its availability in our campus computer facilities. Since Visual C++ is not a compact language and the beginners usually feel considerable difficulties even in understanding the basic techniques, teaching materials must be designed carefully. This paper reports how the lecture series was organized, what contents were distributed to the students as printed lecture notes, and what demonstrations were presented as programming examples at the class room using a video projector. The teaching materials were used two years in a class.

Keywords.

Teaching Materials, Visual C++ Programming, Image Processing, Lecture Note, Demonstration Programs.