ビジュアルプログラミングによるWindows アプリケーションソフトウェアの構築

Construction of an application-software on Microsoft Windows with visual programming

神保　雅人

要 旨　Microsoft Windows 上で動作するGUIアプリケーションソフトウェアを作成するには，通常C++による膨大なプログラミングを必要とする。そこで，ビジュアルプログラミング用の統合開発環境を用いれば，ソフトウェアの開発効率は格段に向上する。本稿ではビジュアル統合開発環境としてBorland C++ Builderを利用し，実用的なソフトウェアの効率的な開発方法を論じる。

１．はじめに

近年，Microsoft Windows 上で動作するGUIアプリケーションソフトウェアの開発効率の向上を図るために，いくつかのビジュアル統合開発環境が製品として市場に提供されている。その中でこれまでのプログラマが慣れ親しんできたC++言語を基盤とするものとして，最も操作性に優れた開発環境がBorland C++ Builder[1)]である。

これは，フォーム上にコンポーネントを貼り付け，イベントハンドラのコードを記述することによって，アプリケーションソフトウェアを作成するビジュアル開発環境のひとつであり，コンポーネントのダブルクリックによってイベントハンドラのヘッダ部が自動生成される等，優れた操作性を持っている。また，アイコン作成のためのイメージエディタも付属した統合開発環境（IDE）であるため，開発に関わる作業は他のツールを起動せずに行うことが可能である。

本稿では，Borland C++ Builderに付属している，ビジュアルコンポーネントライブラリ（VCL；Object Pascalで記述されている）に含まれるコンポーネントの活用により，実用的なソフトウェアが如何にして効率的に開発可能であるかについて，実際にMicrosoft Windows 上で動作するGUIアプリケーションソフトウェアを構築していく課程に沿って論じる。

２．Borland C++ Builder の特徴

２．１．開発環境の概要

Borland C++ Builder は IDE であり，起動直後の画面は図 1 のようになる。メニューバーの左下にはスピードバー，右下にはコンポーネントパレットが配置される。この下に，左側にはオブジェクトのプロパティを設定するためのオブジェクトインスペクタ，右側にはアプリケーションソフトウェアの土台となるフォームが表示される。このフォームの下に一部だけ顔をのぞかせているのが，コードを記述するためのコードエディタで，初期状態ではフォームユニットのソースが表示される。

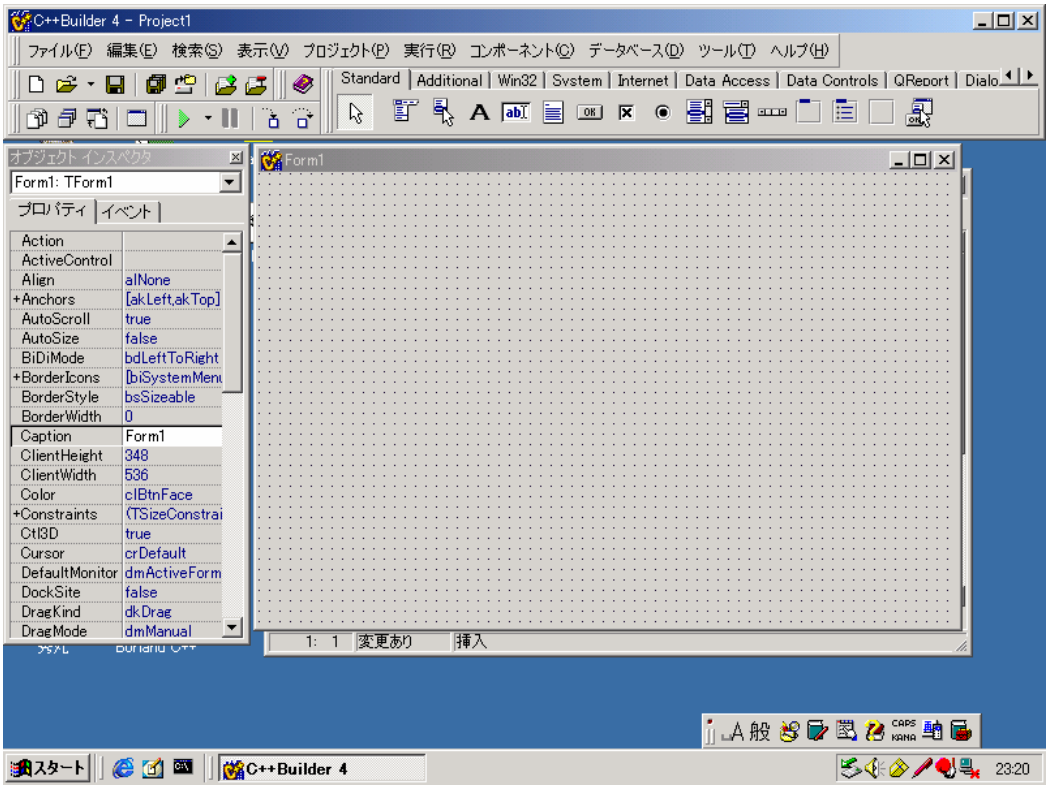

図１．Borland C++ Builder 起動直後の状態

コンポーネントパレットの中から必要なコンポーネントのボタンをマウスで選択し，フォーム上でマウスによる位置指定または範囲指定を行えば，そのコンポーネントが配置される。フォーム上に配置されたコンポーネントを選択すると，そのプロパティがオブジェクトインスペクタに表示され，プロパティの調整が可能となる。

また，フォーム上のコンポーネントにマウスポインタを合わせてダブルクリックを行うと，コードエディタ上のソースに自動的にイベントハンドラ用のメンバ関数の枠組みが自動生成される。その際，クラスの宣言はフォームユニット用のヘッダファイルの中に自動的に記述される。

したがって，開発者の主な仕事は，フォームのデザイン及びイベントハンドラのコードの記述となり，ソフトウェアの開発効率は飛躍的に向上する。

２．２．ユーザインターフェース

Borland C++ Builder で開発可能な Windows アプリケーションソフトウェアには，GUI アプリケーション，コンソールアプリケーション，サービスアプリケーションの 3 種類がある。[2)] このうち，本稿で論じる GUI アプリケーションに関しては，SDI（Single document interface）及び MDI（Multiple document interface）の 2 種類のユーザインターフェースモデルがある。

SDI アプリケーションでは通常，1 個のドキュメントしか開けないのに対して，MDI アプリケーションでは 1 個の親ウィンドウ内で複数個のドキュメントや子ウィンドウを開くことが可能である。後者の作成手順は，複数個のフォームを設計して，親フォームと子フォームとをプロパティで設定していくことで実現できるが，既に用意されている雛型を利用すれば，更に効率の良いものとなる。

MDI アプリケーションの雛型は，メニューバーの「ファイル」→「新規作成」で表示される新規作成ダイアログのプロジェクトページ中に用意されている。これを選択すると，図 2 のように 3 個のフォーム，MainForm（Caption は MDI アプリケーション），MDIChild（Caption は MDI Child），AboutBox（Caption はバージョン情報）が作成される。

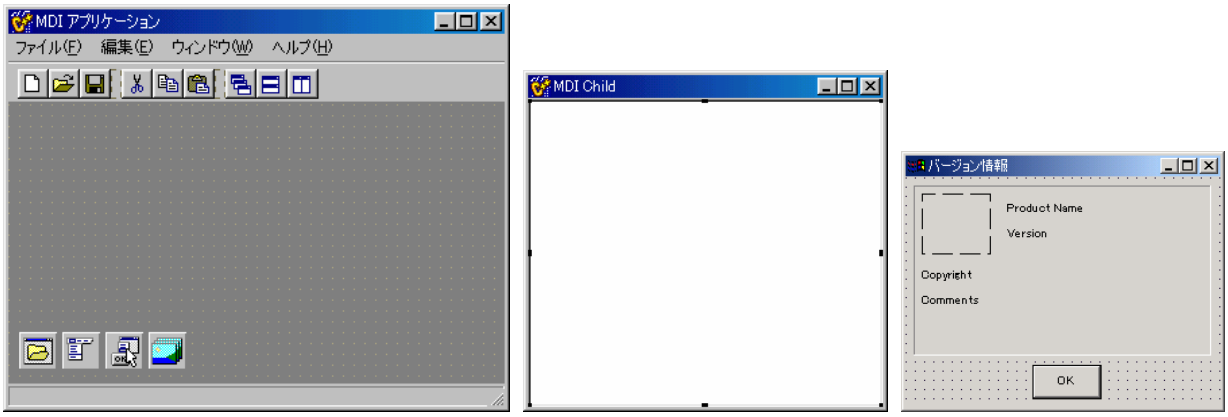

図 2．MDI アプリケーションの雛型

ここで，MainForm 上の左下に配置されているのは，OpenDialog，MainMenu，ActionList，ImageList の各コンポーネントである。また，アクションリストの設定，メニューデザイン，イメージリストへのアイコンの登録，スピードボタンコンポーネントの配置等も済んでいる。開発者は不要なメニュー項目やスピードボタンコンポーネントを削除し，必要なものを付加するだけでよい。

MDIChild フォーム上には，Memo コンポーネントが配置され，これがフォーム全体を占めるように，その Align プロパティが alClient に設定されている。開発者はこのプロパティを変更して，フォーム上に別のコンポーネントを付加することができる。

AboutBox フォーム上には，Panel コンポーネントとその上に配置されたプログラムアイコン用の Image コンポーネント，”Program Name”，”Version”，”Copyright”，”Comments”の Caption が設定された Label コンポーネント，及び”OK”の Caption が設定された Button コンポーネントが配置される。開発者はこれらのプロパティにアイコンの読み込み，または文字入力を行えばよい。

３．Borland C++ Builder による PDF Maker の構築

３．１．PDF Maker の設計

Borland C++ Builder で開発する GUI アプリケーションの一例として，ディジタルカメラで撮影した画像を加工して説明文を付加し，PDF 文書として出力する，”PDF Maker”を構築する。その仕様を次のように定める。

１）ユーザインタフェースモデルは MDI アプリケーションとする。
２）子ウィンドウには，テキスト挿入ボタン及びページの完成ボタンを配置し，イメージ表示用領域と文章作成領域とを設ける。
３）親ウィンドウには，スピードボタンの他に，PDF ファイルへの書き出しボタンを配置する。
４）親ウィンドウで開くファイルの種類は JPEG 形式とし，ファイルを開くと同時に子ウィンドウを開いて，イメージ表示を行う。子ウィンドウには開いた順番にページ番号を付加する。
５）子ウィンドウでページの完成ボタンを押すと，イメージは”outj#*n*.jpg”，文章は”outt#*n*.txt”の名称のファイルとして出力する。(”*n*”はページ番号)
６）親ウィンドウで PDF ファイルへの書き出しボタンを押すと，開かれている子ウィンドの数分の”outj#*n*.jpg”及び”outt#*n*.txt”を読み込んで，PDF ファイルとして出力する。

この仕様に合わせて，MainForm，MDIChild，AboutBox の各フォームの設計を行う。これらはそれぞれ，図 3，図 4，図 5 のようになる。

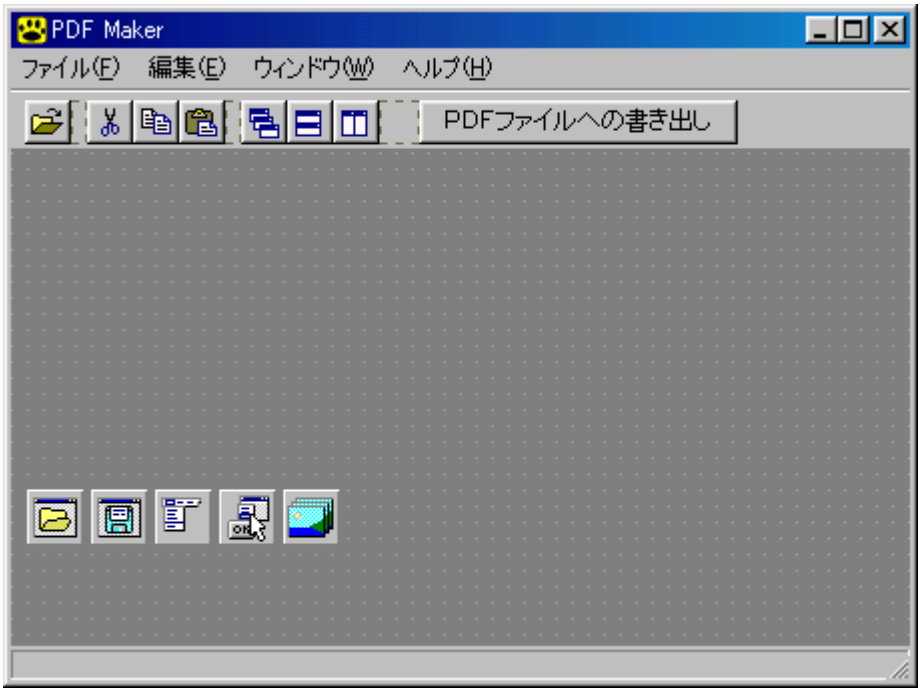

図３．親ウィンドウ用フォーム MainForm の設計

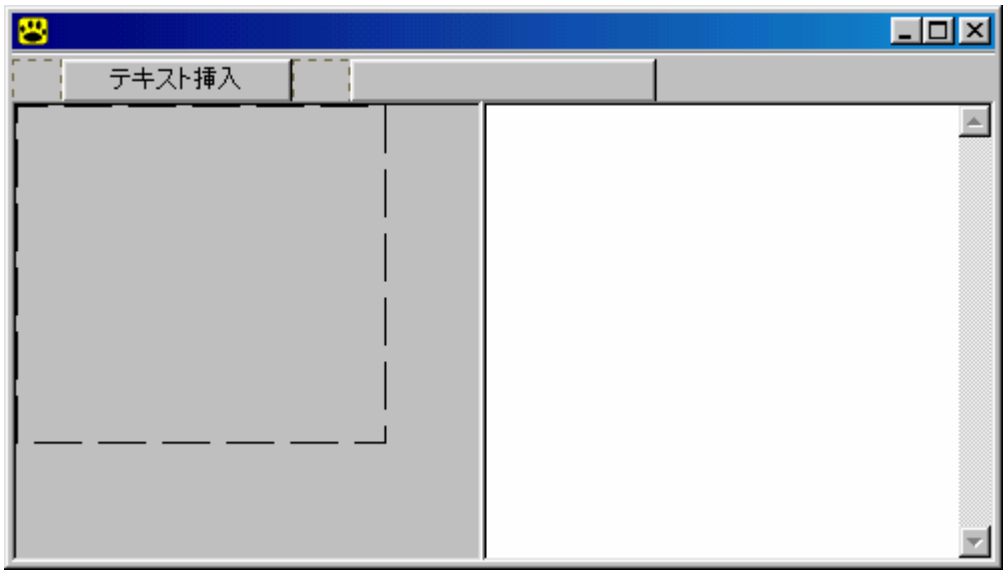

図４．子ウィンドウ用フォーム MDIChild の設計

図５．バージョン情報ウィンドウ用フォーム AboutBox の設計

親ウィンドウ用フォームでは，メニュー及びスピードボタンのうち，新規作成，名前を付けて保存，上書き保存を削除し，PDF ファイルへの書き出しボタン及びその機能を担う SaveDialog コンポーネントを追加する。なお，スピードボタンのうち，切り取り，コピー，貼り付けは子ウィンドウ用フォームの Memo コンポーネントと連携している。

子ウィンドウ用フォームでは，Memo コンポーネントの Width プロパティを 256 に設定した上で，Align プロパティを alRight に設定してフォーム内で右寄せとする。その左側には ScrollBox コンポーネントを配置し，Align プロパティを alClient に設定して残りのフォーム領域全体を占めるようにする。これは，実行時のウィンドウの大きさ変更に応じてサイズを可変とするためである。この中に Image コンポーネント（破線の四角形）を配置する。これにより，子ウィンドウに収まりきらないイメージに対してはスクロールバーが表示される。なお，テキストの挿入ボタンの右に Caption プロパティを空欄にしたボタンを配置するが，このボタンでは，実行時に Caption プロパティに”p．*n* の完成”を書き出すことにする。（”*n*”はページ番号）

バージョン情報ウィンドウ用フォームには，オリジナルアイコンを配置し，必要事項を Label コンポーネントの Caption プロパティに書き込む。なお，コメントに記述している”pdflib. dll”[3] は，このアプリケーションで PDF ファイルへの書き出しの部分に利用するダイナミックリンクライブラリ（DLL）で，詳細は後述する。

イメージファイルの読み込みに関しては，フォームの雛型に配置される OpenDialog コンポーネントをそのまま利用する。Borland C++ Builder にはプレビュー機能付きの OpenPictureDialog コンポーネントも用意されているが，ディジタルカメラで撮影した画像ファイルには通常，番号付けされた命名がなされているのでプレビューは特に必要なく，またメモリーの消費を極力節約するためにも単純な表示の方が有利であるからである。OpenDialog コンポーネントの Filter プロパティは図 6 のように設定する。

図 6． Filter プロパティの設定

３． ２． pdflib.dll の利用

PDFlib[3] は PDF ファイルの作成，PDF ファイルへの文書及びイメージファイルの埋め込みを行うためのライブラリであり，ANSI C，ANSI C++，Java，Perl 等の言語から利用できるバインディングが用意されている。また，イメージファイルとしては TIFF，GIF，PNG，JPEG が取り扱える。[4] このライブラリは http：//www. pdflib. com/pdflib/download/index. html より，様々なプラットフォーム用のバイナリファイルとソースファイルとが入手可能で，前者は商用，後者は非商用のライセンスが設定されている。

ソースファイルから”pdflib. dll”を作成する場合，ソースプログラムは ANSI C 言語で記述され，その C++ラッパーが Microsoft 社製のコンパイラに合わせて設計されているので，Borland C++ Builder で直接コンパイルしても作成できない。ここでは，Visual Studio . Net（3 ヶ月間体験版）中の Visual C++ . Net を利用してコンパイルを行い，”pdflib. dll”を作成する。なお，Borland C++

Builder では，DLL をアプリケーションから呼び出して利用するにあたり，付属するユーティリティプログラム”IMPLIB. EXE”を用いてインポートライブラリを作成する必要がある。

Borland C++ Builder 側の準備として，構築するアプリケーションのソースファイルが置かれたフォルダに，上述の手続きで作成される，” pdflib. dll”，” pdflib. lib”を置き，メニューバーから，「プロジェクト」→「プロジェクトに追加」を選び，” pdflib. lib”をプロジェクトに追加する。また，DLL 中の C 言語で記述された関数を利用するための extern 文をヘッダファイルに記述する。

３．３．PDF Maker のコーディング

PDF Maker の構築に際しては，次の環境を利用した。

１）ハードウェア：	富士通製ノート PC　FMV-BIBLO（5133NA2/W）
	CPU： Pentium 133MHz； RAM： 16+32MB； HDD： 5GB／10GB
２）OS：	Microsoft Windows 2000 Operating System
３）IDE：	Borland C++ Builder 4.0.14.12（Update Pack 2）

コーディングに際しては，自動生成される C++ソースプログラム及びヘッダファイルを利用し，必要なヘッダファイルの#include 文による読み込みの記述と，イベントハンドラの記述とが中心となる。また，上述の DLL 中の関数を利用するためのヘッダファイルは，元になる PDFlib のヘッダファイル”pdflib. h”中の関数プロトタイプ宣言に基づいて，自作する必要がある。

結果として，アプリケーションを構成するソースコードのうち，作者がコード記述を行う必要のあるものは，親ウィンドウ用ソースプログラム”Main. cpp”，子ウィンドウ用ソースプログラム”ChildWin. cpp”，PDFlib 用ヘッダファイル”pdffunc. h”であり，これらを付録として公開する。また，それぞれのソースコードに関しては，アルゴリズム及び留意点をここで論ずる。

①　Main. cpp のコーディング

PDF Maker は MDI アプリケーションであるから，”Main. cpp”（付録 A）には，親フォーム自体のヘッダファイル”Main. h”の他に，子フォーム及びバージョン情報フォームのヘッダファイル”ChildWin. h” 及び”About. h”を読み込む必要があるが，これらについては雛型の段階で#include 文が記述されている。また， VCL コンポーネントを利用するためのヘッダファイル”vcl. h”も読み込まれるようになっている。

Borland C++ Builder によるアプリケーションの開発の際，イメージデータとして JPEG を利用する場合には，標準のヘッダファイルには記述が無く，専用のヘッダファイル”jpeg. h”を読み込まなければならない。[5)] また，上述の” pdflib. dll” 中の関数はこのソースプログラムで利用するので，ヘッダファイル”pdffunc. h”を読み込む。

宣言文"TMainForm *MainForm;"は TMainForm クラスのインスタンス MainForm を作成する。この TMainForm クラスは，"Main.h"の中で"class TMainForm : public TForm"という宣言により，TForm クラスから派生して，クラスを継承している。また，フォーム上に配置されたコンポーネントは，オブジェクトとして TMainForm クラスのデータメンバに含められる。したがって，ここで記述するイベントハンドラはすべてフォームオブジェクトのメソッドとなる。

雛型の段階で，子フォームは自動作成の対象からはずされており，FileOpen1Execute により，ファイルが開かれる際に CreateMDIChild の呼び出しにより作成される。この CreateMDIChild には雛型の段階で，Memo コンポーネントへのテキストの読み込みが記述されているので，Image コンポーネントの Picture プロパティへのイメージの描画に書き換える。

バージョン情報フォームは，雛型の段階で自動作成の対象に入れられているので，メニューバーの「プロジェクト」→「オプション」の選択で表示されるダイアログボックスの「フォーム」タブで，自動作成の対象から外す。これに伴ない，"AboutBox->ShowModal();"と記述されていたイベントハンドラを，実行時に「ヘルプ」→「バージョン情報」の選択により HelpAbout1Execute が呼び出される際に，バージョン情報フォームが作成されるように書き換える。

PDF ファイルへの書き出しボタンを押すと呼び出されるのが，Button1Click である。このイベントハンドラでは，先ず，書き出し用のファイル名を入力させる。その際，「キャンセル」ボタンを押してもデフォルトのファイル名"output.pdf"が適用される仕様を採用し，InputBox を用いる。なお，PDF ファイルのページ数には MDIChildCount で得られる子フォームの数を用いる。

PDF_new 以下，"PDF_"で始まる関数名は，PDFlib の関数である。PDF オブジェクトは PDF_new により生成され，PDF_delete により削除される。また，PDF_open_file により書き出し用のファイルが開かれ，PDF_close で閉じられる。PDF ファイルの各ページは PDF_begin_page から始まり，PDF_end_page で終わる。

ここでは，各ページに仕様にしたがって読み込んだファイルからイメージ，テキストの順で書き出していくが，イメージのページ上の大きさは，横幅が通常は A4 判の 3 分の 2 となるようにスケーリングする。また，縦位置の写真でこの方法では不都合な場合を想定して，イメージを描画した残りが縦方向に A4 判の 3 分の 1 未満となる場合には，高さが A4 判の 3 分の 2 となるようにスケーリングし直す。イメージの書き出しには，PDF_place_image で，PDF オブジェクト及び JPEG ファイルのポインタ，x 座標，y 座標，スケールを指定して行う。

テキストはイメージの下にスペースを空けて，行ごとに書き出す。その際，1 行目は x 座標，y 座標を指定して書き出す関数 PDF_show_xy を用い，2 行目以降はそれに続けて書き出す関数 PDF_continue_text を用いる。なお，PDFlib で用いている x-y 座標は左下を原点とし，x 軸は右方向，y 軸は上方向に向いた設定となっている。

例外処理に関しては，try～catch 文が利用可能であるが，ここでは敢えてこれを用いず，イメー

ジのみのページ，あるいはテキストのみのページも作成できるようにしている。

② ChildWin.cpp のコーディング

子フォームのソースプログラム”ChildWin.cpp”（付録 B）では，親フォームのプロパティである MDIChildCount を参照して利用するために，”Main.h”も読み込む。TMDIChild クラスも TForm クラスから派生している。雛型に付け加えたイベントハンドラは，Button1Click 及び Button2Click である。また，FormCreate では，実行時に開いている子フォームの数を取得し，これを用いて Button2 の Caption を設定する。

Button1Click では，ファイルから読み込んだイメージの左上にテキストを挿入する。この場合，Image コンポーネントから JPEG イメージを編集するために TJPEGImage を用いるが，TJPEGImage では，内部ビットマップイメージへのアクセスができない[5)]ので TBitmap に変換した上で，その Canvas プロパティに TextOut メソッドでテキストを書き出す。このビットマップイメージを再度 JPEG に変換し直して，画面に再描画する。

なお，挿入するテキストは，作者の個人使用を想定して，デフォルト文字列を”Photo by JIMBO Masato”としているが，テキストを挿入したくない場合にも対応できるよう，InputQuery を用いている。これにより，実行時に「キャンセル」ボタンを押せば，何も書き出さないようにできる。また，実行時にテキストボックスの文字列を空にして「OK」ボタンを押した場合にも，何も書き出さないように，空文字列の判定も加えている。

Button2Click では，Image コンポーネントに描画されたイメージ及び Memo コンポーネントに描画されたテキストを，仕様にしたがって書き出す。

③ pdffunc.h のコーディング

Borland C++ Builder で DLL 中の関数を，C 言語風にその関数名で呼び出して利用するには，

```
extern "C" __declspec(dllimport) 戻り値のデータ型 __stdcall 関数名(引数リスト)
```

の構文にしたがって，それぞれの関数を宣言しておく必要がある。”pdffunc.h”（付録 C）では，親フォーム”Main.cpp”で利用している”pfdlib.dll”中の関数をインポートするために，その宣言を記述している。

３．４．PDF Maker の動作確認

PDF Maker を起動すると，先ずは親ウィンドウが現れる。続けてファイルを開くスピードボタン，またはメニューバーから「ファイル」→「開く」を選択した場合，図 7 のような OpenDialog が表示される。

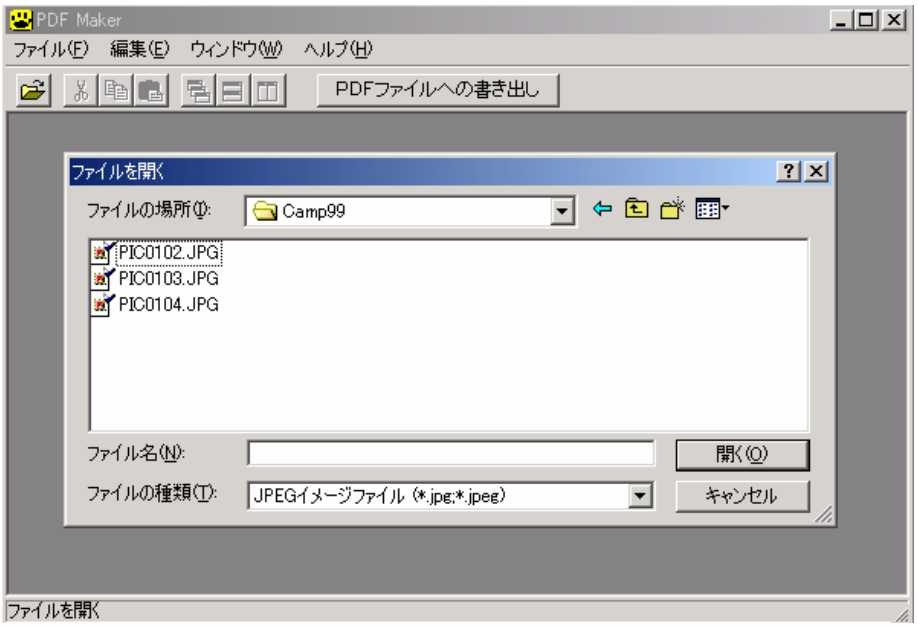

図 7．PDF Maker 起動後，OpenDialog の表示

ここでファイルを選択すると，子ウィンドウが表示され，JPEG イメージが表示される。この子ウィンドウでテキスト挿入ボタンを押すと，図 8 のダイアログが表示される。そのテキストボックスに文字列を入力して「OK」ボタンを押せば，JPEG イメージへのテキスト挿入が行われる。また，子ウィンドウの右側には文章の入力が，直接またはクリップボード経由で行える。図 9 には，1 ページ分の内容を入力し終わった状態を示す。ここで，p. 1 の完成ボタンを押す。

図 8．JPEG イメージへのテキスト挿入ダイアログ

図 9．1 ページ分の入力終了時の状態

上述の操作を 3 ページ分繰り返して，スピードボタンのカスケード表示ボタンを押した状態が図 10 である。ここで，親ウィンドウ上で PDF ファイルへの書き出しボタンを押すと，図 11 のダイアログが表示される。図 12 は，でき上がった PDF ファイルの Acrobat Reader による表示である。

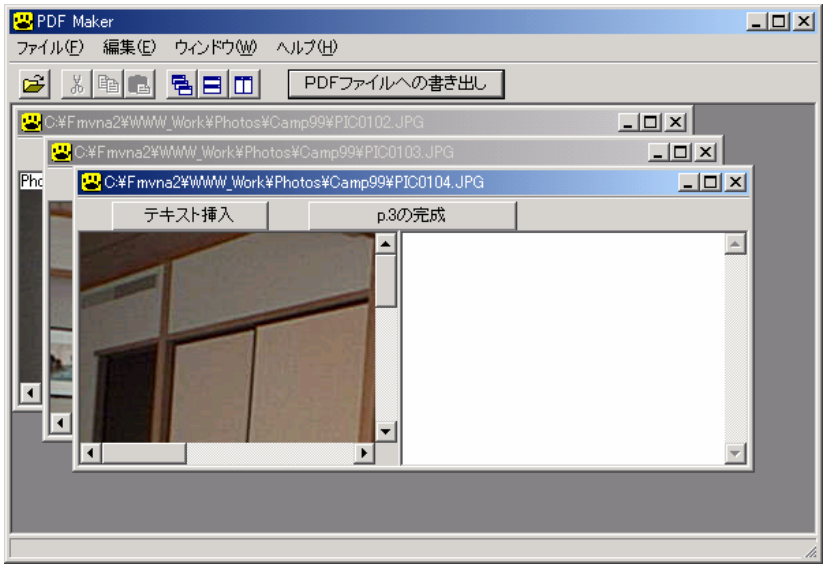

図１０．３ページ分のカスケード表示

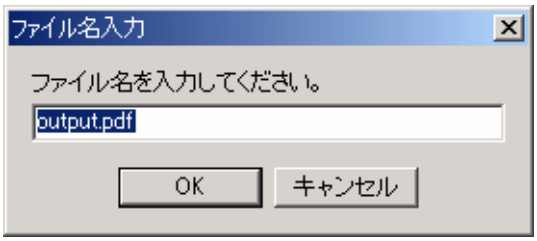

図１１．PDF ファイル名の指定ダイアログ

図１２．PDF ファイルの Acrobat Reader による表示

なお，プロジェクトオプションの設定により，実行形式ファイル pdfmaker. exe のプロパティを開いた際に表示されるバージョン情報を記述できるが，これを表示すると，図 13 のようになる。

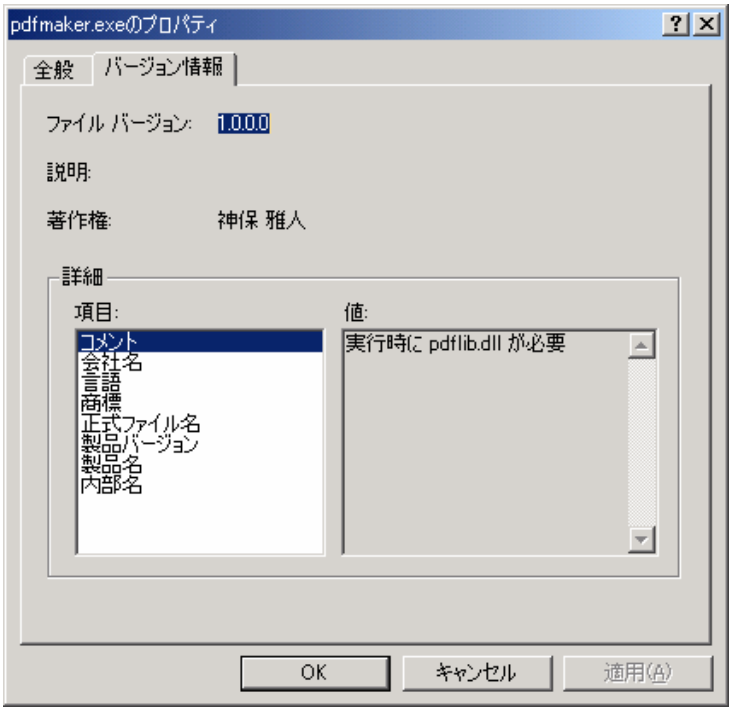

図１３．pdfmaker. exe のプロパティの表示

４．おわりに

本稿では，ビジュアル統合開発環境として操作性に優れた Borland C++ Builder を利用し，実用的な GUI アプリケーションソフトウェアとして，複数の JPEG ファイルを加工したイメージとテキストとから，複数のページから構成される PDF ファイルを作成する，PDF Maker の構築方法を論じた。この構築過程を通じて，GUI アプリケーションソフトウェアの開発効率の向上には，Borland C++ Builder のようなビジュアル統合開発環境が有用であることを例証できた。

また，Borland C++ Builder では，開発したアプリケーションソフトウェアの配布用パッケージを作成するユーティリティも付属している点で，製品化の効率も向上する。ただし，今回構築した PDF Maker では他者の作成したライブラリ PDFlib を利用しているので, 配布用パッケージを作成するとしても，その中に PDFlib を含めて配布することはできない。

なお，筆者の平成 14 年度の 2 年生向けのゼミナールでは，Borland C++ Builder を用いたビジュアルプログラミングをテーマとしたが，簡易ワープロ程度ならば用意に作成できることで，学生の満足度も向上している。今後のプログラミング教育には，以上のような理由からビジュアル統合開発環境が欠かせないものとなりつつある。

参考文献

１）Inprise Corporation，Borland C++ Builder 4 ユーザーズガイド，インプライズ，1999.

２）Inprise Corporation，Borland C++ Builder 4 開発者ガイド，インプライズ，1999.

３）http：//www. pdflib. com/pdflib/index. html

４）PDFlib GmbH and Thomas Merz，*Reference Manual PDFlib － A library for generating PDF on the fly － Version 4.03*，2002.（http：//www. pdflib. com/pdflib/download/index. html より”PDFlib-manual. pdf”として，ダウンロード可能）

５）http：//www. borland. co. jp/qanda/cbuilder/index. html

付 録

Ａ．ソースコード（Main.cpp）

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <jpeg.hpp>
#pragma hdrstop
#include "pdffunc.h"
#include "Main.h"
#include "ChildWin.h"
#include "About.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TMainForm *MainForm;
//---------------------------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent *Owner): TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::CreateMDIChild(String Name)
{
  TMDIChild *Child;
  //--- MDI 子ウィンドウを作成する ----
  Child = new TMDIChild(Application);
  Child->Caption = Name;
  if (FileExists (Name)) {
     Child->Image1->Picture->LoadFromFile(Name);
  }
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FileOpen1Execute(TObject *Sender)
{
  if (OpenDialog->Execute())
    CreateMDIChild(OpenDialog->FileName);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::HelpAbout1Execute(TObject *Sender)
{
  TAboutBox *About = new TAboutBox(this);
  About->ShowModal();
  delete About;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FileExit1Execute(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::Button1Click(TObject *Sender)
{
  #define a4_width      (float) 595.0
  #define a4_height     (float) 842.0
  #define sp_y          (float)  10.0
  #define FONTNAME "HeiseiMin-W3"
  #define ENCODING "90ms-RKSJ-H"
```

```
  int i, n, image, font;
  float scale, x, i_y = a4_height, width, height, y;
  PDF *p;
  AnsiString PDFName, JPEGName, TEXTName;
  PDFName = InputBox("ファイル名入力","ファイル名を入力してください。","output.pdf");
  const char *ppdf = PDFName.c_str();
  if((n = MDIChildCount) > 0) {
    p = PDF_new();
    if (PDF_open_file(p, ppdf) == -1) {
      ShowMessage("PDF ファイルが開けません！");
    }
    else {
      PDF_set_info(p, "Keywords", "image graphics text hypertext");
      PDF_set_info(p, "Subject", "Create PDF documents");
      PDF_set_info(p, "Title", "PDF document");
      PDF_set_info(p, "Creator", "PDF Maker with PDFlib");
      PDF_set_info(p, "Author", "JIMBO Masato");
      for (i=1; i<=n; i++) {
        PDF_begin_page(p, a4_width, a4_height);
        JPEGName = "outj#" + AnsiString(i) + ".jpg";
        const char *pjpg = JPEGName.c_str();
          x = a4_width / 6.0;
          width = a4_width * 2.0 / 3.0;
          height = a4_height * 2.0 / 3.0;
        if ((image = PDF_open_image_file(p, "jpeg", pjpg, "", 0)) == -1)
          ShowMessage("エラー：JPEG ファイルが見当たりません。");
        else {
          scale = width / PDF_get_value(p, "imagewidth", (float) image);
          i_y = a4_height - sp_y - PDF_get_value(p, "imageheight", (float) image) * scale;
          if (i_y < height / 2.0 - sp_y) {
            scale = height / PDF_get_value(p, "imageheight", (float) image);
            i_y = a4_height - sp_y - height;
          }
          PDF_place_image(p, image, x, i_y, scale);
          PDF_close_image(p, image);
        }
        TEXTName = "outt#" + AnsiString(i) + ".txt";
        if (FileExists (TEXTName)) {
          TStringList *TheStr = new TStringList;
          TheStr->LoadFromFile(TEXTName);
          if (TheStr->Count > 0) {
            y = i_y - sp_y * 2.0;
            font = PDF_findfont(p, FONTNAME, ENCODING, 0);
            PDF_setfont(p, font, 12.0);
            PDF_show_xy(p, (TheStr->Strings[0]).c_str(), x, y);
            for (int Index = 1; Index < TheStr->Count; Index++)
              PDF_continue_text(p, (TheStr->Strings[Index]).c_str());
          }
          delete TheStr;
        }
        PDF_end_page(p);
      }
    }
    PDF_close(p);
    PDF_delete(p);
  }
}
//---------------------------------------------------------------------------
```

Ｂ．ソースコード（ChildWin.cpp）

```
//---------------------------------------------------------------------
#include <vcl.h>
#include <jpeg.hpp>
#pragma hdrstop

#include "ChildWin.h"
#include "Main.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"

AnsiString nc;
//---------------------------------------------------------------------
__fastcall TMDIChild::TMDIChild(TComponent *Owner): TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TMDIChild::FormClose(TObject *Sender, TCloseAction &Action)
{
  Action = caFree;
}
//---------------------------------------------------------------------
void __fastcall TMDIChild::Button1Click(TObject *Sender)
{
  TJPEGImage *jpg = new TJPEGImage;
  Graphics::TBitmap *bmp = new Graphics::TBitmap;
  //JPEG File の TJPEGImage への複写
  jpg->Assign( Image1->Picture );
  //Bitmap への変換
  bmp->Assign( jpg );
  //文字列の描画
  AnsiString MemoText = "Photo by JIMBO Masato";
  if (InputQuery("テキスト入力", "文字列を入力してください。", MemoText))
    if ( MemoText != "" ) {
      bmp->Canvas->TextOut(0, 0, MemoText);
    }
  //JPEG に変換し直し
  jpg->Assign( bmp );
  //画面に描画
  Image1->Picture->Assign( jpg );
  delete bmp;
  delete jpg;
}
//---------------------------------------------------------------------------
void __fastcall TMDIChild::FormCreate(TObject *Sender)
{
  nc = AnsiString(MainForm->MDIChildCount);
  Button2->Caption = "p." + nc + "の完成";
}
//---------------------------------------------------------------------------
void __fastcall TMDIChild::Button2Click(TObject *Sender)
{
  Image1->Picture->SaveToFile("outj#" + nc + ".jpg");
  Memo1->Lines->SaveToFile("outt#" + nc + ".txt");
}
//---------------------------------------------------------------------------
```

C. ソースコード (pdffunc.h)

```
typedef struct PDF_s PDF;
extern "C" __declspec(dllimport) PDF * __stdcall PDF_new(void);
extern "C" __declspec(dllimport) int __stdcall PDF_open_file(PDF *p, const char *filename);
extern "C" __declspec(dllimport) void __stdcall PDF_set_info(PDF *p, const char *key, const
char *value);
extern "C" __declspec(dllimport) void __stdcall PDF_begin_page(PDF *p, float width, float
height);
extern "C" __declspec(dllimport) int __stdcall PDF_open_image_file(PDF *p, const char
*imagetype, const char *filename, const char *stringparam, int intparam);
extern "C" __declspec(dllimport) float __stdcall PDF_get_value(PDF *p, const char *key, float
modifier);
extern "C" __declspec(dllimport) void __stdcall PDF_place_image(PDF *p, int image, float x,
float y, float scale);
extern "C" __declspec(dllimport) void __stdcall PDF_close_image(PDF *p, int image);
extern "C" __declspec(dllimport) int __stdcall PDF_findfont(PDF *p, const char *fontname,
const char *encoding, int embed);
extern "C" __declspec(dllimport) void __stdcall PDF_setfont(PDF *p, int font, float
fontsize);
extern "C" __declspec(dllimport) void __stdcall PDF_show_xy(PDF *p, const char *text, float
x, float y);
extern "C" __declspec(dllimport) void __stdcall PDF_continue_text(PDF *p, const char *text);
extern "C" __declspec(dllimport) void __stdcall PDF_end_page(PDF *p);
extern "C" __declspec(dllimport) void __stdcall PDF_close(PDF *p);
extern "C" __declspec(dllimport) void __stdcall PDF_delete(PDF *p);
```