Octo+ Class Library

Reaching out more than one helping hand.

Contents

Introduction

This package of classes provides an integrated solution to file management problems. All classes are fully MFC compliant and work on both 16 and 32 bit platforms.

These classes are not merely wrappers around the original API-functions, but are a real object oriented representation of the different objects encountered in file management.

All classes are derived from the base class CObject. Dynamic type checking using the IsKindOf function is supported on all classes. The diagnostic features Dump and AssertValid are implemented in every class.

The source code contains for every function clearly stated remarks about their use, the meaning of in/out parameters, the result value and their effect. All public functions are documented in the header file, while the protected and private functions are documented in the CPP-file. If you don't derive from this classes, you normally don't have to look at the implementation itself

The classes are designed with two purposes in mind. First, they can be used as they are. You do not need to derive other classes from it before they are useful. Secondly, they are designed with other uses in mind : so you can easily adjust them to more specific needs if you want to.

Goal

Value for money

We feel this library to be valuable to every C++ programmer using MFC. One single useful class already generates a return on investment, by saving countless hours of programming. We plan to release numerous other classes and therefore offer this library as an annual subscription. Entitling you to every new class that is released for the duration of one year starting at subscription date.

Administration/Maintenance

The Octo+ class library is fully compatible with the latest 16-bit and 32-bit releases of Visual C++ ™ including Visual C++ 1.5x (16-bit), Visual C++ 2.2 (32-bit) and Visual C++ 4.0 (32-bit). The library fills Windows development areas not covered by the Microsoft foundation class library. We will continue administering the classes so that they are not in conflict with newer versions of MFC or operating systems released by Microsoft. You will receive the classes when they become available.

This to avoid that you have to program something that already available. Productivity is our aim.

Full Source

The class library comes in Full Source and is well documented. Please read the license agreement to avoid any misunderstandings.

New contributions

Earn your subscription to the class library by contributing classes. These classes should be of the same quality level as the ones we produced to start this project off. When we add your class to the library you get a free subscription. The subscription is valid for one year. The classes contributed remain in the library until they are made obsolete by an MFC release of Microsoft. If you have a better class than the one available in the library let us know. Unless otherwise requested we mention the name of the author. Be aware that submitting the class entitles us to redistribute it. We only accept classes submitted in full source. Classes will not be edited without your knowledge.

Quality assurance

We will only deliver classes that work, are of true value and are well documented. The library should avoid countless hours spent on searching for a new class or reinventing the wheel. Each class is evaluated by a panel of four C++/MFC experts. It is not our aim to become a junk class library, that consists of a little wrapper around a function call. You know that there is a lot out there, but are unsure about its quality. We would like to become a quality label you can rely on.

Free DefManager

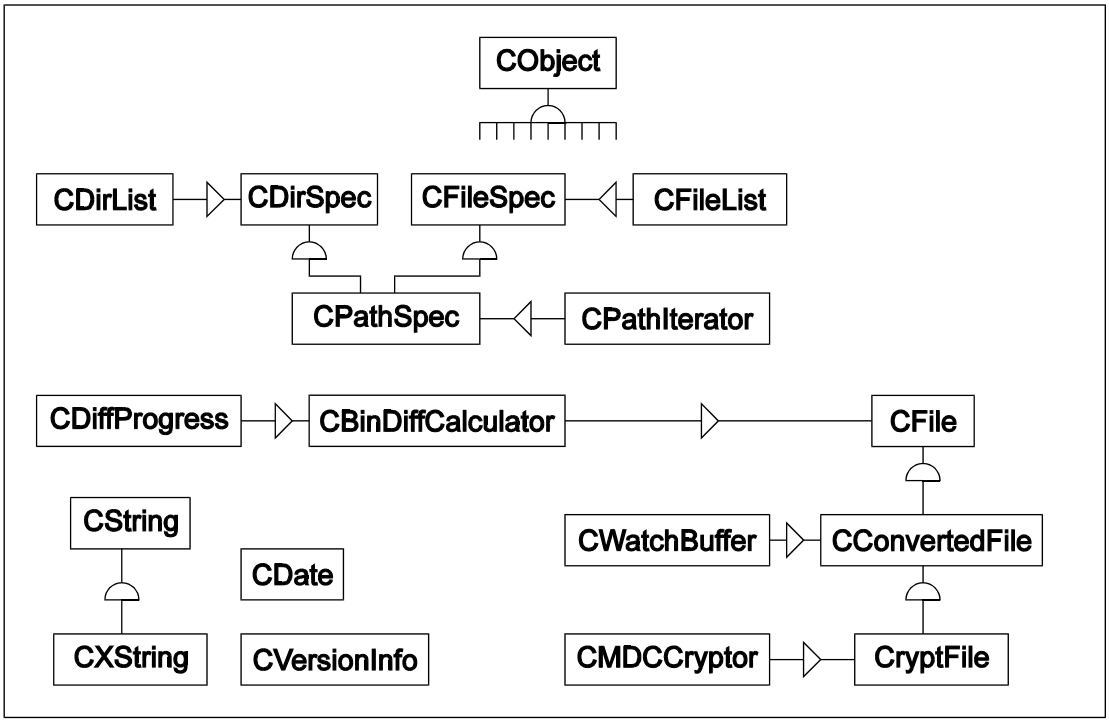
When purchasing the Octo+ class library you receive a free license for a 32bit DefManager. The Ultimate tool to generate DLL's in Visual C++. Value $29.

General structure

The area of these classes can be divided into two parts : file organisation and file manipulation. The first group of classes takes care of file system matters : directory organisation, searching for certain files, copying etc. The second part are classes that actually change the file's contents : encryption and binary difference.

The following drawing gives a general picture of the relationships between the different classes. The notation that is used is that of Coad-Yourdon. Half a circle symbolises inheritance (... is derived from ...) and a triangle means containment (... is part of ...).

Apart from the class CString all classes are directly or indirectly derived from the base class CObject. In order to avoid a clutter in the drawing this relationship is not drawn.



File organisation

File organisation involves all the issues of the directory management on the underlying file system.

The two basic classes that are used for this are **CDirSpec** and **CFileSpec**. They represent the specification of a directory (drive and subdirectory) and of a file (base name and extender). Together they form a **CPathSpec**. E.g. *C:\Windows\256color.bmp* is a path specification. The first part (*C:\Windows*) is the directory spec and the second part (*256color.bmp*) is the file spec. The directory spec consists of a drive spec (*C:*) and a subdirectory (*\Windows*). The file spec contains the base name (*256color*) and the extender (*bmp*). All three classes have a lot of easy access functions to use each part directly.

```
CString sBaseName;
CPathSpec path;
path.SetPath("C:\\Windows\\256color.bmp");
sBaseName = path.GetBaseName();
```

Because CPathSpec inherits from both CDirSpec and CFileSpec all their function can also be applied to a path spec.

Apart from structuring the directory and file specification, different action can be applied on the real underlying files. All these functions start with the prefix *Do*. E.g. DoMakeNew really creates the directory with the given specification.

```
CDirSpec dir("C:\\Temp\\Ours");
dir.DoMakeNew();
```

The previous example creates the directory *C:\Temp\Ours*. If the directory *C:\Temp* does not yet exist it will be created before *C:\Temp\Ours* is created. Other actions that can be performed are :

```
BOOL DoGetCurrentDir();
BOOL DoSetCurrentDir() const;
BOOL DoMakeNew() const;
BOOL DoRemove(BOOL bRecursively = FALSE, BOOL bAlsoRemoveReadOnly = FALSE) const;
CString GetFileSystemType();
BOOL MakeTemp();
BOOL MakeUnique();
BOOL MakeLargestExisting();
BOOL Exists() const;
BOOL IsEmpty() const;
void Empty();
BOOL IsEmptyDir() const;
```

File specifications also contain information like the file size, the attributes and the time specification. These can be accessed using the following functions :

```
CTime GetTime() const;
BOOL SetTime(CTime time);
LONG GetLength() const;
BOOL SetLength(LONG lnLength);
BYTE GetAttributes() const;
BOOL SetAttributes(CFile::Attribute attributes);
BOOL IsEmpty() const;
void Empty();
```

Both directory and file specifications have fully defined comparison operators (==, !=, <=, <, >=, >), empty construction, construction from a const char*, copy constructors and assignment operators.

The functions that actually do something with a file are grouped in the path spec. Some of them are :

```
BOOL DoCopy(CPathSpec destinationPath) const;
BOOL DoMove(CPathSpec destinationPath) const;
BOOL DoRemove(BOOL bIgnoreReadOnly = FALSE) const;
BOOL DoGetInfo();
BOOL DoSetTime();
BOOL DoSetLength();
BOOL DoSetAttributes();
```

The existence of files can be organised using the functions :

```
BOOL MakeTemp(BOOL bCreateEmpty = TRUE, const char* pszPrefix = TEXT("TMP"));
BOOL MakeAbsolute();
BOOL MakeUnique();
BOOL Exists() const;
```

Starting from a path specification you can iterate all the files in the specified directory (using a specific file specification with wild characters, e.g. ***.tmp***). You can also iterate all the directory specifications. These iterations are done using a **CPathIterator** object together with the following member functions of CPathSpec :

```
BOOL GetFirstFile(CPathIterator& Iterator) const;
CFileSpec GetNextFile(CPathIterator& Iterator) const;

BOOL GetFirstDir(CPathIterator& Iterator) const;
CDirSpec GetNextDir(CPathIterator& Iterator) const;
```

If you want to recursively search for a specific file you can use :

```
BOOL DoSearch(CFileSpec fileName, CDirSpec startingDir = CDirSpec(), BOOL bRecursively = FALSE);
```

If you ‘d rather get the full result of a search in one call, so you can handle the iteration yourself, you can use the classes **CDirList** and **CFileList**. This search function gives you a (sorted) array of CDirSpecs or CFileSpecs. This can be interesting to fill a listbox or combobox.

File manipulation

The file manipulation classes convert the contents of a file in another format. This is used for two reasons : to encrypt or decrypt a file and to compute the binary difference between two versions of a file.

Encryption

Encryption can be useful if you want to hide the contents of a file from other persons : For example when you distribute a file via disk or modem, or just to make sure no one else can access your private information.

The **CMDCCryptor** class uses a fast secret-key encryption.. The cipher operates on 256 bit (32 byte) blocks, using a 768 bit (96 byte) key. The only restriction is that the two 48 byte halves of the key should not be the same.

To make the encryption - decryption fully transparent you can use the **CCryptFile** class. This class is derived from CFile, so all the things you can do on a CFile-object apply also to a CCryptFile object. You can read from it, write to it, move the position, change or retrieve the length, flush the buffers etc. The functions ReadString and WriteString which are normally only available in the buffered CStdioFile class are here also available to the non-buffered CFile class.

```
virtual BOOL Open(const char* pszFileName, UINT nOpenFlags, CFileException* pError = NULL);
virtual CFile* Duplicate() const;
virtual DWORD GetPosition() const;
virtual LONG Seek(LONG lOff, UINT nFrom);
virtual DWORD GetLength() const;
virtual void SetLength(DWORD dwNewLen);
virtual UINT Read(void FAR* lpBuf, UINT nCount);
virtual void Write(const void FAR* lpBuf, UINT nCount);
virtual char* ReadString(char* psz, UINT nMax);
virtual void WriteString(const char* psz);
virtual void LockRange(DWORD dwPos, DWORD dwCount);
virtual void UnlockRange(DWORD dwPos, DWORD dwCount);
virtual void Abort();
virtual void Flush();
virtual void Close();
```

Within one file you can even encrypt some parts of the file and leave others untouched. Imagine you organise a lot of information in a file and it contains a large picture. If you don't want the picture to be encrypted you can turn the encryption off, write the picture and afterwards turn it back on. Another use is a small header in the beginning of the file, which states whether the following information is encrypted or not.. The header itself is then never encrypted.

```
BOOL IsConvertEnabled() const;
BOOL EnableConvert(BOOL bEnable = TRUE);
WORD ForceEnableConvert(BOOL bEnable = TRUE);
```

Because encryption works on blocks of 32 bytes, your file size will always be dividable by 32. This is also done fully transparent.

The actual CFile functionality is in fact grouped in the class **CConvertedFile.** This class allows conversion during reading and writing. For doing so it calls the pure virtual functions

```
virtual BOOL ConvertRead(LPCBYTE pOriginal, LPBYTE pConverted) = 0;
virtual BOOL ConvertWrite(LPCBYTE pOriginal, LPBYTE pConverted) = 0;
```

These functions are implemented in the derived class CCryptFile. If you want to provide other conversions apart from encryption during file access, you could derive directly from CConvertedFile. But this is not necessary to use encryption.

Binary Difference

The classes that encapsulated binary differences are able to compute the difference between two versions of a file. Imagine you have an original file and an updated file. Using the **CBinDiffCalculator** you can see what changed from the original to the updated version. This difference can then be written to a third file. Later on you can reproduce the updated version, starting from the original and the difference file.

```
Difference = Update - Original
Update = Original + Difference
```

This can be useful if you want to send an updated version to other people. You do not have the send them the entire new file, but just the differences. These can then be applied to the original file which results in the fully updated version. Normally the difference file is much smaller then the entire updated file.

You can directly supply path specifications to create the difference or updated file

```
void SubtractFiles(LPCTSTR orgFilNam, LPCTSTR derivedFilNam, LPCTSTR diffFilNam);
void AddFiles(LPCTSTR orgFilNam, LPCTSTR derivedFilNam, LPCTSTR diffFilNam);
```

Another possibility is that you create three CFile derived objects, open them and let the CBinDiffCalculator object use them. This creates the possibility to use other CFile based classes : CMemFile, CCryptFile or your own.

```
virtual void SubtractFiles(CFile* pOrgFil, CFile* pDerivedFil, CFile* pDiffFil);
virtual void AddFiles(CFile* pOrgFil, CFile* pDerivedFil, CFile* pDiffFil);
```

While the differences are computed or applied a progressbar bar can visualise how much is already completed. This information is supplied to a CDiffProgress object. By default this is just written to standard output (e.g. a console program). You can derive your own class from it and make another visual indicator

```
void ReplaceProgressBar(CDiffProgress* pProgressBar);
```

If you want to use the class only to AddFiles you can #define BDEXTR as 1, which will make the class more compact.

Utility Classes

Some extra utility classes are also provided.

String and date

The first class is an extended string class **CXString**. This class is mainly used internally within the other classes, but can also be used directly. It can easily check whether a string contains a valid integer or floating point number. Spaces can be removed in front of, behind or in the middle of a string. Etc.

The **CDate** class encapsulates a year - month - day triplet. You can calculate days using :

```
CDate operator+(int deltaDate) const;
const CDate& operator+=(int deltaDate);
CDate operator-(int deltaDate) const;
const CDate& operator-=(int deltaDate);
int operator-(CDate date) const;
```

Or you can compare dates (==, !=, <= , <, >=, <), determine the day of the week. You can check whether a date is valid or really exists. Persistent dates are supported by archiving them using :

```
friend CArchive& AFXAPI operator<<(CArchive& ar, CDate date);
friend CArchive& AFXAPI operator>>(CArchive& ar, CDate& rdate);
```

All dates from the beginning of the Julian calendar (1583) to infinity are supported.

Version information

Most executable files and dynamic link libraries contain version information. Some of it is shown when you retrieve the properties of a file using File Manager or Windows Explorer. To get all the information you can use the **CVersionInfo** class.

By calling the

```
BOOL ReadInfo(const char* pszPathName);
```

function the following information is retrieved :

```
DWORD m_dwSignature;
DWORD m_dwStrucVersion;
DWORD m_dwFileVersionMS;
DWORD m_dwFileVersionLS;
DWORD m_dwProductVersionMS;
DWORD m_dwProductVersionLS;
DWORD m_dwFileFlagsMask;
DWORD m_dwFileFlags;
DWORD m_dwFileOS;
DWORD m_dwFileType;
DWORD m_dwFileSubtype;
DWORD m_dwFileDateMS;
DWORD m_dwFileDateLS;

DWORD m_dwLanguageCountryID;
CString m_sLanguageCountry;

CString m_sComments;
CString m_sCompanyName;
CString m_sFileDescription;
CString m_sFileVersion;
CString m_sInternalName;
CString m_sLegalCopyright;
CString m_sLegalTrademarks;
CString m_sOriginalFilename;
CString m_sPrivateBuild;
CString m_sProductName;
CString m_sProductVersion;
CString m_sSpecialBuild;
```

How to use

The different classes can be divided into clusters. Each cluster can be used individually :

- File Organisation
  - CDirSpec [dir]
  - CFileSpec [file]
  - CPathSpec [path]
  - CDirList [dirlist]
  - CFileLIst [filelist]
  - CPathIter [pathiter]
  - CXString [xstring]
  - File constants [filelmt.h]
- Encryption
  - CConvertedFile [convfile]
  - CCryptFile [crypfile]
  - CMDCCryptor [mdccrypt]
  - CWatchBuffer [watchbuf]
- Binary Difference

- ◊ CBinDiffCalculator [bdifcalc]
- ◊ CDiffProgress [progress]

◊ Date
- ◊ CDate [date]

◊ Version information
- ◊ CVersionInfo [verinfo]

To use them add a complete cluster of classes to your make file and recompile. No Visual C++ features are used in the classes, so every compiler that supports MFC can use the classes.

Registration

The Octo+ Class library is a "shareware program" and some classes of it are provided at no charge to the developer for evaluation. Feel free to share it with your friends, but please do not give it away altered or as part of another system. The essence of "user-supported" software is to provide developers with quality software without high prices, and yet to provide incentive for programmers to continue to develop new products. If you find this Library useful and find you are using classes of the Octo+ Class Library in your products, you must make a registration payment of $100 to Periphere (the people behind the scenes). The $100 registration fee will license one copy for one developer on any one computer at any one time. Upon registration the complete Octo+ Class Library will be eMailed to you. It is considered as an annual subscription fee. New classes and newer versions of existing classes will be eMailed to you.

On line Registration:
You can register on-line at Compuserve and the Complete Octo+ Class Library will be eMailed to the registration account as soon as possible. The process is simple, just !GO SWREG and follow the directions using the following registration numbers:

FMGMT classes registration #8295

Or contact:

Periphere nv
Leuvensesteenweg, 282 bus 1
3190 BOORTMEERBEEK
(CIS-ID: 100117,265)
Tel: (..32)15/52.82.62
Fax: (..32)15/52.82.63

Single User Octo+ Class Source Code License Agreement

IMPORTANT - READ CAREFULLY

By installing this source code package, or exercising your rights to make and use copies of the Software (as may be provided for below), or keeping this package for over 30 days, you agree to be bound by the terms of the Periphere Octo+ class Software License Agreement. If you do not agree to the terms of this Agreement , promptly return this package to the place from which you obtained it within 30 days for a full refund. If you would like specific rights not granted in this agreement, please contact Periphere for more information.

Periphere License Agreement

The enclosed software, including, but not limited to, one or more of the following: source code, object code, dynamic link libraries, static libraries, header files, sample programs, utility programs, Makefiles and scripts, together with the accompanying documentation

(collectively known as the "Software") is owned by Periphere or its suppliers and is protected by US copyright laws and international treaties. Therefore, you must treat the Software like any copyrighted material (e.g., a book or musical recording) except that you may make either (a) one backup copy of the Software solely for backup purposes, or (b) transfer the Software to a hard disk and keep the original copy solely for backup purposes.

Periphere grants to you (one software programmer) the limited right to use only one copy of the Software on a single computer (typically one personal computer) in the manner set forth in this agreement. Each Software Programmer must have his or her own license to use the Software.

Subject to the restrictions contained in this agreement, you **MAY**:

a) Modify (i.e. modify the source code and rebuild) the dynamic link libraries and statically linked libraries and incorporate the modified dynamic link libraries and statically linked libraries into C++ software application products that you develop.
b) Make and distribute copies of the dynamic link libraries and statically linked libraries of the Software as incorporated into C++ software application products that you develop provided that the Software, or other Periphere products, do not constitute a major portion of the value of your product.
c) Solely with the respect to electronic documents, you may make an unlimited number of copies (either in hardcopy or electronic form), provided that such copies shall be used only for internal purposes and are not republished or distributed beyond the licensee's premises.
d) Use and modify the source code version of those portions of the Software that are identified in the documentation as the Sample Code ("SAMPLE CODE"), provided you do not distribute the SAMPLE CODE, or any modified version of the SAMPLE CODE, in source code form.

Notwithstanding any provisions in this agreement to the contrary, you may **NOT:**

a) Distribute in any manner any of the header files, source code, SAMPLE CODE, Makefiles, object modules or independent static libraries of the Software.
b) Use, copy, modify, merge or compile all or any portion of the source code or object code of the Software except as expressly provided in this agreement.
c) Make telecommunication transmittal of the Software.
d) Distribute any portion of the Software or any derivative of any portion of the Software in a software development product or otherwise in competition with Periphere's distribution of the Software.
e) Decompile, disassemble or reverse engineer any object code form of any portion of the Software.
f) Export from the United States any portion of the Software without obtaining prior written consent of Periphere and all applicable export licenses and governmental permits.
g) Rent or lease the Software.
h) Disclose any source codes of the Software to any person or entity.
i) Port the Software to any computer Operating System other than: Windows 3.1, Windows NT, Windows 95 and Macintosh System 7 without the express written consent of Periphere.

The source codes of the Software are valuable assets of Periphere. You agree to keep all source codes of the Software in confidence. You may not transfer or assign the Software or your rights under this agreement.

Limited Warranty

Periphere warrants to you (the original licensee only) that the unaltered Software will substantially perform the functions described in the documentation for a period of 60 days after the date of delivery of the Software to you. Periphere's sole obligation under this warranty shall be limited to using reasonable efforts to correct material, documented, reproducible defects in the unaltered Software that you describe and document to Periphere during the 60-day period. In the event that Periphere fails to correct a material, documented, reproducible defect within a reasonable period, Periphere may, in Periphere's discretion, replace the defective Software, or refund to you the amount that you paid Periphere for the defective Software and cancel this agreement and the licenses granted herein. In such event, you agree to return to Periphere all copies of the Software.

EXCEPT AS SPECIFICALLY PROVIDED IN THE PARAGRAPH IMMEDIATELY ABOVE, PERIPHERE MAKES NO WARRANTY, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL PERIPHERE BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING LOSS OF PROFITS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF PERIPHERE WAS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Periphere will provide you (the original licensee only) with limited technical support by telephone, or by electronic media for a period of 60 days after delivery of the Software to you.

U.S. GOVERNMENT RESTRICTED RIGHTS

The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(l)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software -- Restricted Rights at 48 CFR 52.227-19, as applicable. Contractor/manufacturer is Periphere NV, 3190-Boortmeerbeek, Leuvensesteenweg, 282 bus 1, Belgium. This agreement is governed by Belgium law and the Brussels court.

For more information contact:

Periphere nv
Leuvensesteenweg, 282 bus 1

3190 BOORTMEERBEEK
(CIS-ID: 100117,265)
Tel: (..32)15/52.82.62
Fax: (..32)15/52.82.63

An Octopus has many arms, through this library we hope to become a valuable partner within your development team by reaching out more than one helping hand.