



BREW 2005
conference

Reusable C++ Objects in BREW®

Paul L. Anderson
Author & Lecturer
Anderson Software Group, Inc.

June 2, 2005

Why C++?

- **Benefits**

- Data abstraction and encapsulation
- C++ has better type checking than C
- Good for OOP and reusability

- **What C++ Do You Need to Know?**

- Classes
- Constructors and destructors
- Class member functions
- Inheritance
- Static member functions
- Scope operator (::)
- Operators new and delete
- Inline functions
- Default arguments

C++ with BREW

- **Classes**
 - Encapsulate access to BREW APIs
 - Define controlled interface for BREW APIs
- **Class Constructors and Destructors**
 - Constructor acquires BREW resources
 - Destructor releases BREW resources
- **Class Member Functions**
 - Defines interface to BREW API calls
 - Hides internal complexities of BREW API

Class Example

```
class MyObject {
private:
    IShell *pIShell;           // BREW resource
    . . .
public:
    MyObject(IShell *ps) {     // constructor
        // acquire BREW resource...
    }
    ~MyObject() {             // destructor
        // release BREW resource...
    }
    // member functions here...
    bool doSomething(int data) {
        // call BREW API to perform service
    }
    . . .
};
```

C++ with BREW

- **Inheritance**
 - Defines “Is-A” relationship between classes
 - Interfaces to BREW framework
- **Static Member Functions**
 - Class functions called without an object
 - Provides linkage with BREW startup code
- **Scope Operator (::)**
 - Identify member functions outside class definitions
 - Register BREW callbacks for startup code

Class Example

```
class MyApp : public AEEApplet {
private:
    IShell      *pIShell;      // BREW resource
    MyObject    *pObject;      // user defined object
    . . .
public:
    . . .
    // static member functions
    static void freeAppData(MyApp *pMyApp);
    static bool InitAppData(MyApp *pMyApp);
};

// class static member function definitions
bool MyApp::InitAppData(MyApp *pMyApp) {
    . . .
}

void MyApp::freeAppData(MyApp *pMyApp) {
    . . .
}
```

C++ with BREW

- **Operators new and delete**

- Creates C++ objects on heap
- Must overload to call BREW helper functions

```
inline void *operator new(size_t size) {  
    return MALLOC(size);  
}  
inline void operator delete(void *ptr) {  
    FREE(ptr);  
}
```

- **Invocation**

```
MyObject *pObject = new MyObject(plShell);  
...  
delete pObject ;
```

C++ with BREW

- **Default arguments**

- Provides default behaviors for C++ functions, constructors, and member functions
- Useful to set defaults for BREW APIs

- **Example**

```
void draw(IDisplay *piDisplay, int width = 20) {  
    ...  
}
```

- **Invocations**

```
draw(piDisplay);           // width is 20  
draw(piDisplay, 10);      // width is 10
```

Design Approach

- **Similar to BREW Application Wizard**
- **“Boilerplate” C++ Files**
 - Header File
`MyApp.h`
 - Source File
`MyApp.cpp`
- **User C++ Files**
 - Header File(s)
`MyObject.h`
 - Source File(s)
`MyObject.cpp`

MyApp.h

```
#ifndef _MYAPP_H__
#define _MYAPP_H__
// MyApp.h - header file for C++ App

#include "AEEModGen.h"      // Module interface definitions
#include "AEEAppGen.h"     // Applet interface definitions
#include "AEEShell.h"      // Shell interface definitions
#include "AEEStdLib.h"     // Helper functions

// include object file headers here...
#include "MyObject.h"

// overload new and delete operators
inline void *operator new(size_t size) {
    return MALLOC(size);
}
inline void operator delete(void *ptr) {
    FREE(ptr);
}
```

MyApp.h (continued)

```
class MyApp : public AEEApplet {
private:
    AEEDeviceInfo DeviceInfo;        // device info
    IDisplay      *pIDisplay;
    IShell       *pIShell;
    // add any Interface or Object pointers here...
    MyObject     *pObject;
protected:
    // event and message handlers from a C applet
    bool         OnAppInitData();
    void         OnAppfreeData();
    bool         OnEvent(AEEEvent eCode, uint16 wParam,
                        uint32 dwParam);
public:
    static bool  HandleEvent(MyApp *pMyApp,
                            AEEEvent eCode, uint16 wParam, uint32 dwParam);
    static void  freeAppData(MyApp *pMyApp);
    static bool  InitAppData(MyApp *pMyApp);
};
#endif
```

MyApp.cpp

```
#include "MyApp.h"
#include "MyApp.bid"
extern "C"
int AEEClsCreateInstance(AEECLSID ClsId, IShell *pIShell,
                        IModule *po, void **ppObj) {
    *ppObj = NULL;
    if (ClsId == AEECLSID_MYAPP) { // Create App
        if (AEEApplet_New(sizeof(MyApp), ClsId, pIShell, po,
                        (IApplet**)ppObj,
                        // pass pointers to static member functions
                        (AEEHANDLER)MyApp::HandleEvent,
                        (PFNFREEAPPDATA)MyApp::freeAppData)) {
            // Initialize applet data
            if (MyApp::InitAppData((MyApp*)*ppObj)) {
                return AEE_SUCCESS;
            } else {
                IAPPLET_Release((IApplet*)*ppObj);
                return EFAILED;
            }
        } // end AEEApplet_New
    }
    return EFAILED;
}
```

MyApp.cpp (continued)

```
// class static member function definitions

bool MyApp::InitAppData(MyApp *pMyApp) {
    return pMyApp->OnAppInitData();
}

void MyApp::freeAppData(MyApp *pMyApp) {
    pMyApp->OnAppfreeData();
}

bool MyApp::HandleEvent(MyApp *pMyApp, AEEEvent eCode,
                        uint16 wParam, uint32 dwParam) {
    return pMyApp->OnEvent(eCode, wParam, dwParam);
}
```

MyApp.cpp (continued)

```
// class static member function definitions

bool MyApp::OnAppInitData() {
    // initialize Display and Shell variables
    pIDisplay = m_pIDisplay;
    pIShell = m_pIShell;

    // Get the device information for this handset
    DeviceInfo.wStructSize = sizeof(DeviceInfo);
    ISHELL_GetDeviceInfo(pIShell, &DeviceInfo);

    // Insert code here for creating objects here...
    pObject = new MyObject(pIShell);
    // check for failures, return success if ok
    return true;
}

void MyApp::OnAppfreeData() {
    // Insert code for deleting objects here...
    delete pObject;
}
```

MyApp.cpp (continued)

```
// class static member function definitions

bool MyApp::OnEvent(AEEEvent eCode, uint16 wParam,
                   uint32 dwParam) {
    switch (eCode) {
        // App is told it is starting up
        case EVT_APP_START:
            // Add your code here...
            return true;

        // App is told it is exiting
        case EVT_APP_STOP:
            // Add your code here...
            return true;

        // other events here...

    }
    return false;
}
```

MyObject.h

```
#ifndef _MYOBJECT_H__
#define _MYOBJECT_H__
// MyObject.h - header file for object

// include files here for interfaces...
#include "AEEStdLib.h"      // Helper Functions

class MyObject {
private:
    // instance variables here
    IShell *pIShell;
    . . .
public:
    MyObject(IShell *ps) {
        // constructor code here...
    }
    ~MyObject() {
        // destructor code here...
    }
    // member functions here...
};
#endif
```

SoundPlayer Object

- **Objectives**

- Encapsulate BREW ISoundPlayer API
- Provide OO interface
- Integrate into C++ Boilerplate Files
- Make reusable

- **Example**

- Play MP3
- Boilerplate Files

 - `PlayMP3.h`

 - `PlayMP3.cpp`

- Object Files

 - `SoundPlayer.h`

SoundPlayer Object

- **First Version**

- Create with existing ISoundPlayer resource
- No error checking

- **Example**

```
SoundPlayer *pSoundPlayer =  
    new SoundPlayer("music.mp3",  
                    plShell,  
                    plSoundPlayer);
```

SoundPlayer.h

```
#ifndef __SOUNDPLAYER_H__
#define __SOUNDPLAYER_H__
#include "AEESound.h"
class SoundPlayer {
private:
    char          *filename;    // MP3 filename
    IShell        *pIShell;
    ISoundPlayer  *pISoundPlayer;
public:
    SoundPlayer(char *fname, IShell *ps, ISoundPlayer *psp) {
        filename = fname;    pIShell = ps;
        pISoundPlayer = psp;

        AEESoundPlayerInfo playerInfo;
        playerInfo.eInput = SDT_FILE;
        playerInfo.pData = filename;    // MP3 file to play
        ISOUNDPLAYER_SetInfo(pISoundPlayer, &playerInfo );
    }
    ~SoundPlayer() {
        ISOUNDPLAYER_Stop(pISoundPlayer);
    }
}
```

SoundPlayer.h (continued)

```
char *getFileName() { return filename; }

void start() {
    ISOUNDPLAYER_Play(pISoundPlayer);
}

void stop() {
    ISOUNDPLAYER_Stop(pISoundPlayer);
}

void suspend() {
    ISOUNDPLAYER_Pause(pISoundPlayer);
}

void resume() {
    ISOUNDPLAYER_Resume(pISoundPlayer);
}
};
#endif
```

PlayMP3.h

```
#ifndef _PLAYMP3_H__
#define _PLAYMP3_H__
// PlayMP3.h - header file for C++ App
#include "AEEModGen.h" // Module interface definitions
#include "AEEAppGen.h" // Applet interface definitions
#include "AEEShell.h" // Shell interface definitions
#include "AEEStdLib.h" // Helper functions
#include "SoundPlayer.h"
. . . .
class MyApp : public AEEApplet {
private:
    AEEDeviceInfo DeviceInfo; // device info
    IDisplay *pIDisplay;
    IShell *pIShell;
    ISoundPlayer *pISoundPlayer; // SoundPlayer interface
    SoundPlayer *pSoundPlayer; // SoundPlayer object
. . . .
};
#endif
```

PlayMP3.cpp

```
// PlayMP3.cpp - play an MP3 file
#include "PlayMP3.h"
#include "PlayMP3.bid"
extern "C"
int AEEClsCreateInstance(AEECLSID ClsId, IShell *pIShell,
                        IModule *po, void **ppObj) {
    *ppObj = NULL;
    if (ClsId == AEECLSID_PLAYMP3) { // Create App
        if (AEEApplet_New(sizeof(MyApp), ClsId, pIShell, po,
            . . . .
            // pass pointers to static member functions
            (AEEHANDLER)MyApp::HandleEvent,
            (PFNFREEAPPDATA)MyApp::freeAppData)) {
            // Initialize applet data
            if (MyApp::InitAppData((MyApp*)*ppObj)) {
                . . . .
            }
        } // end AEEApplet_New
    }
    return EFAILED;
}
```

PlayMP3.cpp (continued)

```
// class static member function definitions

bool MyApp::InitAppData(MyApp *pMyApp) {
    return pMyApp->OnAppInitData();
}

void MyApp::freeAppData(MyApp *pMyApp) {
    pMyApp->OnAppfreeData();
}

bool MyApp::HandleEvent(MyApp *pMyApp, AEEEvent eCode,
                        uint16 wParam, uint32 dwParam) {
    return pMyApp->OnEvent(eCode, wParam, dwParam);
}
```

PlayMP3.cpp (continued)

```
bool MyApp::OnAppInitData() {
    . . . .
    // open SoundPlayer Interface
    if (ISHELL_CreateInstance(pIShell, AEECLSID_SOUNDPLAYER,
        (void **)&pISoundPlayer) != SUCCESS) {
        pISoundPlayer = NULL;
        return false;
    }
    // create SoundPlayer object to play MP3 file
    pSoundPlayer = new SoundPlayer("piano.mp3",
        pIShell, pISoundPlayer);
    if (pSoundPlayer == NULL)
        return false;
    return true;
}

void MyApp::OnAppfreeData() {
    delete pSoundPlayer;
    if (pISoundPlayer != NULL) {
        ISOUNDPLAYER_Release(pISoundPlayer);
        pISoundPlayer = NULL;
    }
}
```

PlayMP3.cpp (continued)

```
bool MyApp::OnEvent(AEEEvent eCode, uint16 wParam,
                    uint32 dwParam) {
    switch (eCode) {
        case EVT_APP_START:
            pSoundPlayer->start();
            return true;
        case EVT_APP_STOP:
            pSoundPlayer->stop();
            return true;
        case EVT_APP_SUSPEND:
            pSoundPlayer->suspend();
            return true;
        case EVT_APP_RESUME:
            pSoundPlayer->resume();
            return true;
        . . . .
    }
    return false;
}
```

SoundPlayer Object

- **Second Version**

- Manages ISoundPlayer resource if not supplied
- Checks for errors
- More reusable

- **Invocations**

```
SoundPlayer *pSoundPlayer =  
    new SoundPlayer("music.mp3",  
                    pIShell,  
                    pISoundPlayer);
```

```
SoundPlayer *pSoundPlayer =  
    new SoundPlayer("music.mp3", pIShell);
```

SoundPlayer.h

```
class SoundPlayer {
private:
    char          *filename;        // MP3 filename
    bool          own_ISound;
    IShell       *pIShell;
    ISoundPlayer *pISoundPlayer;
public:
    SoundPlayer(char *fname,
                IShell *ps, ISoundPlayer *psp = NULL) {
        filename = fname;
        pIShell = ps;
        pISoundPlayer = psp;
        own_ISound = false;
        if (pISoundPlayer == NULL) {
            if (ISHELL_CreateInstance(pIShell,
                AEECLSID_SOUNDPLAYER,
                (void **)&pISoundPlayer) == SUCCESS)
                own_ISound = true;
            else
                return;
        }
    }
};
```

SoundPlayer.h (continued)

```
// SoundPlayer constructor
AEESoundPlayerInfo playerInfo;
playerInfo.eInput = SDT_FILE;
playerInfo.pData = filename; // MP3 file to play
if (ISOUNDPLAYER_SetInfo(pISoundPlayer,
    &playerInfo) != AEE_SUCCESS)
    pISoundPlayer = NULL;
}

bool objectBuilt() {
    return (pISoundPlayer) ? true : false;
}

~SoundPlayer() {
    ISOUNDPLAYER_Stop(pISoundPlayer);
    if (own_ISound)
        ISOUNDPLAYER_Release(pISoundPlayer);
}
```

SoundPlayer.h (continued)

```
char *getFileName() { return filename; }

void start() {
    ISOUNDPLAYER_Play(pISoundPlayer);
}

void stop() {
    ISOUNDPLAYER_Stop(pISoundPlayer);
}

void suspend() {
    ISOUNDPLAYER_Pause(pISoundPlayer);
}

void resume() {
    ISOUNDPLAYER_Resume(pISoundPlayer);
}
};
```

PlayMP3.h

```
#ifndef _PLAYMP3_H__
#define _PLAYMP3_H__
// PlayMP3.h - header file for C++ App
#include "AEEModGen.h" // Module interface definitions
#include "AEEAppGen.h" // Applet interface definitions
#include "AEEShell.h" // Shell interface definitions
#include "AEEStdLib.h" // Helper functions
#include "SoundPlayer.h"
. . . .
class MyApp : public AEEApplet {
private:
    AEEDeviceInfo DeviceInfo; // device info
    IDisplay *pIDisplay;
    IShell *pIShell;
    SoundPlayer *pSoundPlayer; // SoundPlayer object
. . . .
};
#endif
```

PlayMP3.cpp

```
bool MyApp::OnAppInitData() {
    . . .
    // create SoundPlayer object to play MP3 file
    pSoundPlayer = new SoundPlayer("piano.mp3", pIShell);
    if (pSoundPlayer == NULL || !pSoundPlayer->objectBuilt())
        return false;
    // if no failures, then return success
    return true;
}

void MyApp::OnAppfreeData() {
    if (pSoundPlayer && pSoundPlayer->objectBuilt())
        delete pSoundPlayer;
}
```

File Objects

- **Objectives**

- Encapsulate BREW IFileMgr and IFile APIs
- Provide OO interface for reading/writing
- Integrate into C++ Boilerplate Files
- Make reusable

- **Design Approach**

- File class for generic file handling
- Separate wrapper classes to read/write
- Error handling
- Objects release file resources

File Object Classes

- **File Class**

- Encapsulates IFileMgr API
- Call with file name, invoke methods

```
File *myFile = new File("somefile", pIShell);  
myFile->rename("otherfile");  
if (myFile->exists())  
    myFile->remove();
```

- **FileRead Class**

- Encapsulates IFile API
- Wrapper for File class
- Opens file for reading

```
FileRead *input = new FileRead(myFile);  
input->read(buf, 0, nbytes);
```

File Object Classes

- **FileWrite Class**

- Encapsulates IFile API
- Wrapper for File class
- Opens file for writing or appending

```
FileWrite *output = new FileWrite (myFile);
```

```
output->write(buf, 0, nbytes);
```

```
FileWrite *log = new FileWrite (myFile, _OFM_APPEND);
```

- **Example**

- Copy Files
- Boilerplate Files

```
MyCopy.h
```

```
MyCopy.cpp
```

- Object Files

```
File.h
```

MyCopy.h

```
class MyApp : public AEEApplet {
private:
    AEEDeviceInfo DeviceInfo;           // device info
    IDisplay      *pIDisplay;
    IShell       *pIShell;
    char          buffer[BUFSIZE];
    File          *source, *dest;
    FileRead     *input;
    FileWrite     *output;
protected:
    bool OnAppInitData();
    void OnAppfreeData();
    bool OnEvent(AEEEvent eCode, uint16 wParam,
                uint32 dwParam);
public:
    static bool HandleEvent(MyApp *pMyApp, AEEEvent eCode,
                            uint16 wParam, uint32 dwParam);
    static void freeAppData(MyApp *pMyApp);
    static bool InitAppData(MyApp *pMyApp);
    void DrawText(const char *pcText);
};
```

MyCopy.cpp

```
bool MyApp::OnAppInitData() {  
    . . . .  
    source = dest = input = output = NULL;  
    // allocate source and dest File objects  
    source = new File("news.txt", pIShell);  
    if (source == NULL || !source->objectBuilt()) {  
        return false;  
    }  
    dest = new File("newscopy.txt", pIShell);  
    if (dest == NULL || !dest->objectBuilt()) {  
        return false;  
    }  
    // if no failures, then return success  
    return true;  
}
```

```
void MyApp::OnAppfreeData() {  
    // delete FileRead, FileWrite, File objects  
    delete input;    delete output;  
    delete source;  delete dest;  
}
```

MyCopy.cpp (continued)

```
bool MyApp::OnEvent(AEEEvent eCode, uint16 wParam,
                    uint32 dwParam) {
    switch (eCode) {
        case EVT_APP_START:
            // open read file object
            input = new FileRead(source);
            if (input == NULL || !input->canRead()) {
                DrawText("input file error");
                return true;
            }
            // if dest file exists, remove it
            if (dest->exists())
                dest->remove();

            // open write file object
            output = new FileWrite(dest);
            if (output == NULL || !output->canWrite()) {
                DrawText("output file error");
                return true;
            }
    }
}
```

MyCopy.cpp (continued)

```
// still in EVT_APP_START

// use object buffer to copy files
int nbytes;
do {
    nbytes = input->read(buffer, 0, BUFSIZE);
    output->write(buffer, 0, nbytes);
} while (nbytes == BUFSIZE);
return true;
// other event handlers
. . . .
}
return false;
}
```

File.h

```
#include "AEEFile.h"
#include "AEEStdLib.h"
class File {
private:
    IShell    *pIShell;
    IFileMgr  *pIFileMgr;
    char      *filename;
public:
    File(const char *fname, IShell *ps) {
        filename = (char *)MALLOC(STRLEN(fname)+1);
        STRCPY(filename, fname);    pIShell = ps;

        // open File Manager
        if (ISHELL_CreateInstance(pIShell, AEECLSID_FILEMGR,
            (void **)&pIFileMgr) != SUCCESS)
            pIFileMgr = NULL;
    }
    ~File() {
        if (pIFileMgr != NULL)
            IFILEMGR_Release(pIFileMgr);
        FREE(filename);
    }
}
```

File.h (continued)

```
IFileMgr *getIFileMgr() { return pIFileMgr; }

const char *getFileName() { return filename; }

bool objectBuilt() {
    return (pIFileMgr) ? true : false;
}

bool rename(const char *newname) {
    return IFILEMGR_Rename(pIFileMgr,
        filename, newname) == SUCCESS;
}

bool exists() {
    return IFILEMGR_Test(pIFileMgr, filename) == SUCCESS;
}

bool remove() {
    return IFILEMGR_Remove(pIFileMgr, filename) == SUCCESS;
}
};
```

File.h (continued)

```
class FileRead {
private:
    IFile *pIFile;
public:
    FileRead(File *fread, AEEOpenFileMode mode = _OFM_READ) {
        pIFile = IFILEMGR_OpenFile(fread->getIFileMgr(),
            fread->getFileName(), mode);
    }
    ~FileRead() {
        if (pIFile != NULL)
            IFILE_Release(pIFile);
    }
    bool canRead() {
        return (pIFile) ? true : false;
    }
    IFile *getIFile() { return pIFile; }
    int read(char *buf, int offset, int nbytes) {
        return IFILE_Read(pIFile, buf+offset, nbytes);
    }
};
```

File.h (continued)

```
class FileWrite {
private:
    IFile *pIFile;
public:
    FileWrite(File *fwrite, AEEOpenFileMode mode=_OFM_CREATE) {
        pIFile = IFILEMGR_OpenFile(fwrite->getIFileMgr(),
            fwrite->getFileName(), mode);
    }
    ~FileWrite() {
        if (pIFile != NULL)
            IFILE_Release(pIFile);
    }
    bool canWrite() {
        return (pIFile) ? true : false;
    }
    IFile *getIFile() { return pIFile; }
    int write(char *buf, int offset, int nbytes) {
        return IFILE_Write(pIFile, buf+offset, nbytes);
    }
};
```

Summary

- **Why C++ with BREW?**
 - C++ is a better C
 - Stronger type checking
 - More useful features
- **OO Design Approach**
 - Use Boilerplate approach
 - Develop C++ class libraries
- **Pros and Cons**
 - C++ objects are reusable
 - Overhead in execution time and code space