

2

Introduction à DirectX 8.1

2

Introduction à DirectX 8.1

2.1	Présentation et installation du SDK DirectX 8.1 pour C++	37
2.2	Les outils offerts par DirectX 8.1	41
2.3	Préparer un projet multimédia avec Microsoft Visual C++	44
2.4	Création d'une application	46
2.5	Les bibliothèques et les fichiers à inclure dans le projet	56
2.6	Notions de base de la programmation DirectX 8.1 en C++	58
2.7	Architecture de DirectX 8.1	65
2.8	Conclusion	68

DirectX 8.1 est un kit de développement logiciel (SDK) élaboré par Microsoft Corporation. Le but principal est de réaliser des applications multimédias de très haute performance. Ce logiciel s'exécute dans l'environnement Windows, et nécessite un outil de développement intégrant les langages C++ ou Visual Basic.

Le SDK DirectX 8.1 est composé d'un ensemble d'interfaces de bas niveau, intégrant de nombreuses techniques pour le traitement d'images en deux ou trois dimensions, pour la gestion d'effets sonores et de la musique, mais également pour la mise en œuvre de protocoles de communication en réseau, et la reconnaissance de périphériques divers, tels que les claviers, les souris, les joysticks.

Ce chapitre est consacré à la présentation et à l'installation du SDK DirectX 8.1 sur le poste de travail du programmeur. Il décrit également l'ensemble de ses composants, les bibliothèques et les fichiers nécessaires pour la création d'un projet multimédia dans l'environnement Microsoft Visual C++ 6.0. Les programmeurs et les concepteurs d'applications multimédias familiers d'une version antérieure du logiciel pourront découvrir les nouveaux apports de la version DirectX 8.1.

2.1. Présentation et installation du SDK DirectX 8.1 pour C++

Microsoft DirectX 8.1 est un ensemble d'interfaces de programmation de bas niveau (API), dont l'objectif principal est la création de jeux vidéo, mais également d'autres types d'applications multimédias à hautes performances.

Les interfaces proposées par DirectX 8.1 sont de bas niveau, car elles permettent d'accéder directement aux composants matériels de l'ordinateur, occultant les fonctionnalités du système d'exploitation Windows. En effet, la gestion du matériel réalisée par Windows est relativement lente, et complètement insatisfaisante en terme d'efficacité pour des applications multimédias. Ainsi, DirectX 8.1 définit un ensemble de composants dédiés aux tâches de gestion du matériel, et capables de mettre en œuvre ses caractéristiques afin d'accélérer les traitements, pour offrir une performance optimale à l'exécution de l'application multimédia.

Ces composants ont la charge de détecter les matériaux suivants, présents dans votre machine :

1. La carte graphique, ses résolutions graphiques (taille de l'écran, nombre de couleurs utilisés), et ses capacités d'accélération 2D et 3D.

38 Chapitre 2 - Introduction à DirectX 8.1

2. La carte son, afin de déterminer ses pilotes et le niveau d'accélération lors de la lecture de sons, de musiques, ou dans la mise en œuvre d'effets sonores.
3. Les périphériques d'entrée tels que la souris, le clavier et le joystick, utilisables pour interagir avec l'application multimédia. DirectX 8.1 détecte également certaines caractéristiques essentielles pour tirer le meilleur parti des fonctionnalités liées aux périphériques d'entrées, telles que les boutons de la souris, la rétroaction du joystick (*force feedback*), etc.
4. Les cartes réseaux et les modems, pour permettre de communiquer avec des amis sur un réseau local, ou par Internet, via un protocole de communication approprié.

Ce livre vous permettra de découvrir la manière d'intégrer les composants DirectX 8.1 dans votre application, et d'exploiter le meilleur de votre matériel, pour époustoufler vos amis ou collègues par la rapidité et l'efficacité de votre application multimédia.

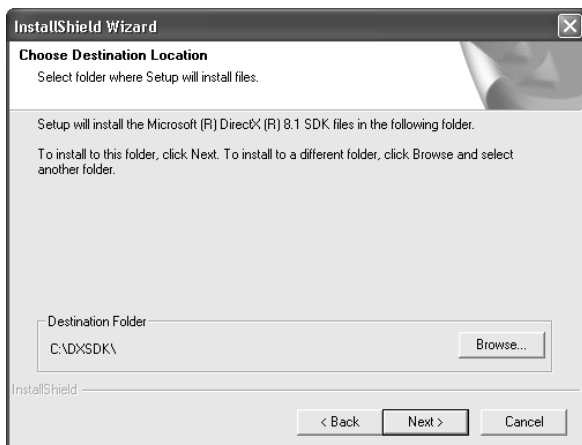
Pour développer une application multimédia avec DirectX 8.1, vous devez commencer par vous le procurer. Il vous est proposé soit gratuitement, par un téléchargement sur le site de Microsoft Corporation, soit sous forme payante, en commandant le CD contenant le SDK. Le téléchargement du SDK est possible uniquement si vous possédez une connexion Internet à haut débit. En effet, le Kit DirectX 8.1 a une taille d'environ 173 Mo. Les temps de téléchargement sont rédhibitoires pour ceux qui disposent d'un modem au débit modeste de 56 Kbits/seconde !

Le SDK DirectX 8.1 peut être téléchargé sur le site de Microsoft Corporation, à l'adresse suivante <http://www.microsoft.com/downloads> ; cliquez simplement sur le lien *DirectX 8.1*.

Le SDK DirectX 8.1 consiste en un fichier compressé et auto-extractible. Il suffit de double-cliquer dessus, puis de sélectionner un répertoire cible, pour décompresser cette archive (environ 225 Mo sur votre disque dur).

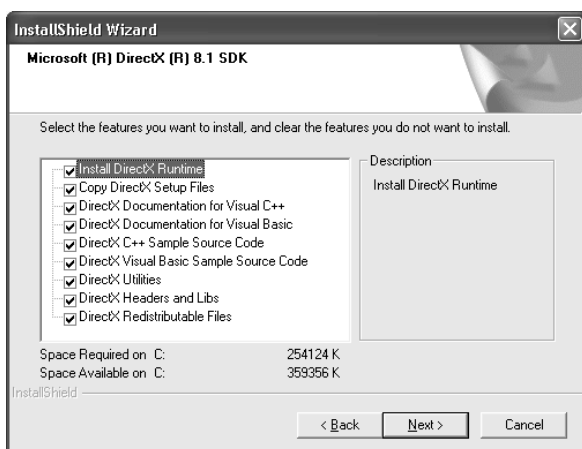
Le fichier *setup.exe* est situé dans le répertoire principal de l'archive. Il permet de débiter l'installation du SDK DirectX 8.1, de prendre connaissance des termes de la licence d'utilisation et d'exploitation de DirectX 8.1, et de sélectionner un répertoire de destination.

Figure 2-1 :
*Sélectionner le
répertoire cible
pour l'installation
de DirectX 8*



Cette figure permet de choisir, sur le disque dur, le répertoire où sera installé le SDK DirectX 8.1. Le bouton **Browse** provoque l'ouverture d'une boîte de dialogue, permettant de spécifier un emplacement particulier ; par défaut, le répertoire est `C:\DXSDK`. Vérifiez que le disque dur de la machine dispose de suffisamment d'espace pour installer le SDK. Lorsque le répertoire est choisi, cliquez sur le bouton suivant (**Next**).

Figure 2-2 :
*Choix des
composants
DirectX 8 à
installer*



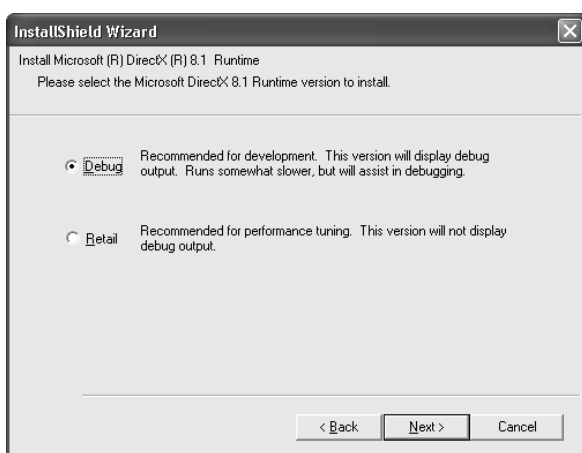
Cette figure permet de sélectionner les composants à installer, tel que le cœur du système DirectX 8.1, ses bibliothèques et ses fichiers d'en-têtes, qui sont obligatoires pour le développement avec l'environnement Microsoft Visual C++ 6.0. Le SDK DirectX 8.1 propose des utilitaires, pour déterminer notamment les caractéristiques matérielles de votre machine, ainsi que des exemples d'applications DirectX avec

40 Chapitre 2 - Introduction à DirectX 8.1

leurs codes sources. Il est également possible d'installer les fichiers d'aide (en anglais). Il faut sélectionner les composants DirectX 8.1 Runtime, les exemples fournis avec le SDK, les utilitaires de configurations de DirectX 8.1 et la documentation.

Lorsque les composants sont sélectionnés, et que le chemin d'installation est défini, cliquez sur le bouton **Next**, pour poursuivre le processus d'installation du SDK avec la fenêtre suivante :

Figure 2-3 :
*Type
d'installation
DirectX 8*



Cette fenêtre permet de sélectionner le type de Runtime qui assurera l'exécution des applications DirectX. Le mode Commercial (*Retail*) assure une performance optimale, et n'est à préconiser que lorsque l'application est achevée, et prête à être commercialisée. Le mode débogage (*Debug*) est nettement moins performant ; mais il permet de déterminer et d'obtenir des informations sur les dysfonctionnements lors du processus de développement et des procédures tests de l'application.

Dans un premier temps, il est intéressant de sélectionner le mode *Debug*, pour le développement d'une application multimédia puis, lorsque celle-ci est finalisée, d'utiliser le mode *Retail* pour optimiser sa performance et la commercialiser.

Cliquez sur le bouton suivant (**Next**) pour engager le processus d'installation. À l'issue de cette procédure, le système installe le Runtime DirectX 8.1, mis en œuvre par le système d'exploitation pour exécuter les applications DirectX. Windows XP est livré avec le Runtime DirectX 8.1. Cette version sera remplacée par la version de débogage, dans le cas où cette option a été sélectionnée. Au terme de l'installation du SDK DirectX 8.1, le système redémarre, pour intégrer les modifications et prendre en compte cette nouvelle application.

**SDK DirectX 8.1**

Dans la suite de ce livre, nous considérerons que le SDK DirectX 8.1 est installé dans le répertoire *C:\DXSDK*.

2

2.2. Les outils offerts par DirectX 8.1

Le SDK DirectX 8.1 propose des outils intéressants lors du développement d'applications, et lorsque celles-ci rencontrent certains problèmes de fonctionnement.

Les outils principaux intégrés dans le SDK DirectX 8.1 sont :

- DirectX Caps Viewer ;
- DirectX Control Panel ;
- DirectX Diagnostic.

DirectX Caps Viewer

L'outil Microsoft DirectX Caps Viewer permet d'énumérer les composants DirectX qui sont installés dans votre système, et de contrôler les périphériques qui leurs sont associés, ainsi que leurs caractéristiques.

Le fichier exécutable de l'outil Caps Viewer *DXCapsViewer.exe* est situé dans le répertoire *Bin\DXUtils*, dans le dossier principal d'installation du SDK.

L'outil de visualisation des caractéristiques DirectX est composé de deux parties complémentaires, et se manie de la même manière que l'Explorateur Windows.

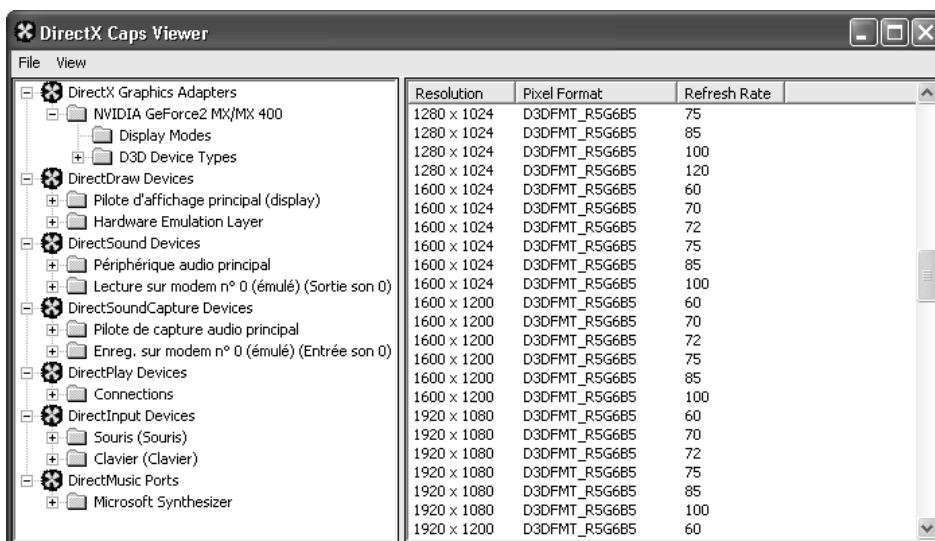
La partie gauche indique la liste des composants DirectX sous une forme arborescente, qu'il est possible de parcourir en double-cliquant sur les dossiers. Chaque dossier regroupe les périphériques pris en charge par le composant DirectX. La sélection d'un dossier permet d'afficher le détail des caractéristiques du composant, et de la configuration des périphériques dans la partie droite.

L'illustration ci-dessous montre l'exploration des informations relatives au composant DirectX Graphics, chargé de la gestion du périphérique d'affichage (carte vidéo : *GeForce 2 MX*). Dans cet exemple, le dossier *Display Modes* est

42 Chapitre 2 - Introduction à DirectX 8.1

sélectionné dans la partie gauche, provoquant l'affichage de toutes les résolutions reconnues par ce type de carte graphique.

Figure 2-4 :
*Liste des
périphériques
gérés par DirectX*
8



DirectX Control Panel

L'outil Microsoft DirectX Control Panel est installé par défaut avec le SDK DirectX 8.1, pour examiner et modifier les propriétés des composants DirectX. Cet utilitaire contient des onglets pour chaque composant DirectX. Il assure les fonctionnalités suivantes :

1. Obtenir des informations sur la version des composants DirectX.
2. Modifier le niveau et les propriétés de débogage, pour aider à comprendre les dysfonctionnements de l'application durant les processus de développement, de tests et d'exploitation.
3. Visualiser les pilotes spécifiques et les informations concernant le matériel pris en charge par les composants DirectX.
4. Basculer entre les versions de débogage et finales des bibliothèques pour les composants Microsoft Direct3D, DirectMusic et DirectInput.

Cet outil est installé dans le Panneau de configuration de l'environnement Windows ; il est aisément identifiable à l'icône *DirectX*.

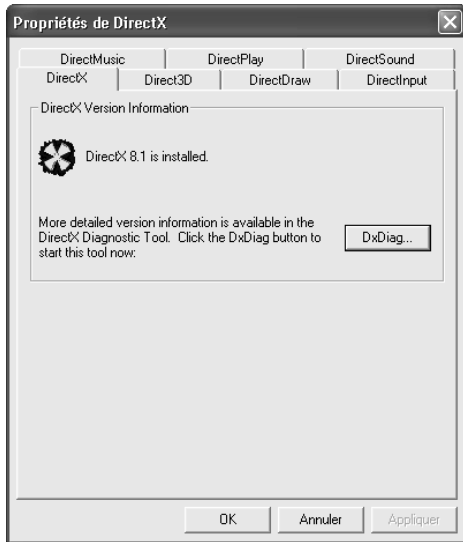


Figure 2-5 :
*Propriétés
DirectX 8*

2

Chaque onglet de la boîte de dialogue permet d'obtenir des informations concernant les composants DirectX 8.1 et les périphériques qu'ils gèrent. L'onglet **Direct3D** informe le développeur sur les capacités 3D du matériel et sur le choix de l'environnement d'exécution des applications (débogage, finale) ; l'onglet **DirectDraw** permet de connaître les caractéristiques 2D de la carte vidéo et les pilotes qui sont installés. L'onglet **DirectInput** permet de configurer les options de débogage concernant les périphériques pris en charge par le système. Les onglets **DirectSound** et **DirectMusic** permettent d'obtenir des informations sur les périphériques multimédias, et notamment sur la carte son utilisée pour l'enregistrement et la lecture de fichiers musicaux. L'onglet **DirectPlay** permet de connaître la liste des services réseaux pris en charge par DirectX pour réaliser les communications dans une application client-serveur.

DirectX Diagnostic

L'outil de diagnostic Microsoft DirectX permet d'obtenir des informations sur le système et sur les composants DirectX installés. Il offre une série de tests pour chaque composant de DirectX, pour s'assurer de son bon fonctionnement. La version de l'outil installé avec le SDK DirectX autorise les développeurs à communiquer des problèmes directement à l'équipe conceptrice du Kit.

Cet outil est composé de plusieurs onglets, dont le but est de fournir des informations sur la mémoire, le type de processeur, les fichiers DirectX installés

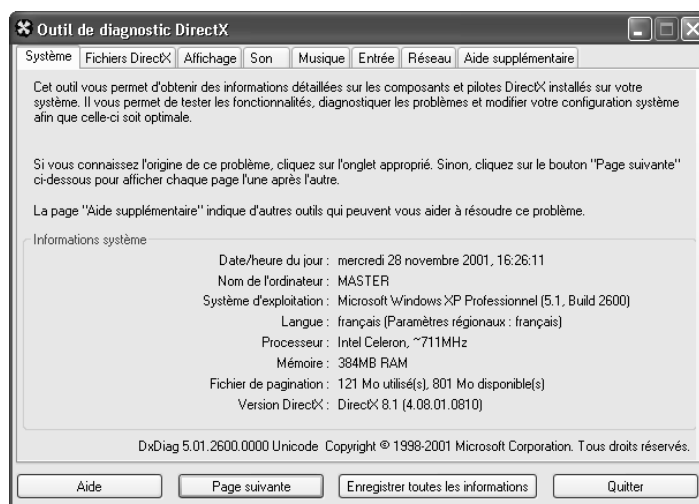
44 Chapitre 2 - Introduction à DirectX 8.1

dans le système, les caractéristiques de la carte graphique, de la carte son, des périphériques d'entrées/sorties, et des composants réseaux.

L'outil de diagnostic est accessible à partir de l'onglet principal de l'outil DirectX Control Panel.

L'illustration ci-dessous montre les informations relatives au périphérique d'affichage, et propose quelques tests, pour vérifier le bon fonctionnement du système.

Figure 2-6 :
*Diagnostics des
composants
DirectX 8*



La section suivante est consacrée à la découverte de l'environnement Microsoft Visual C++ 6.0, pour préparer correctement la création d'une application multimédia, en intégrant les librairies et les fichiers DirectX 8.1.

2.3. Préparer un projet multimédia avec Microsoft Visual C++

Microsoft Visual C++ est un environnement de développement pour les applications dans le langage C++, fonctionnant sous le système d'exploitation Windows.

Visual C++ 6.0 est conçu pour rendre le développement d'applications plus productif, en proposant un ensemble d'outils, développés autour d'un système riche en fonctionnalités, allant du compilateur au débogueur. L'environnement intégré de

développement Visual C++ 6.0 permet aux développeurs d'écrire du code selon les dernières technologies et les standards les plus récents, tels qu'Internet. Le code produit par cet outil est réutilisable dans d'autres projets, extensible, compréhensible, capable d'accéder à des données externes à l'application, telles que des bases de données.

Microsoft Visual C++ 6.0 est uniquement disponible en version payante, auprès d'un distributeur ou de Microsoft Corporation. Il constitue toutefois l'outil nécessaire et indispensable pour le développement de programmes en C++.

Visual C++ 6.0 est disponible selon trois éditions :

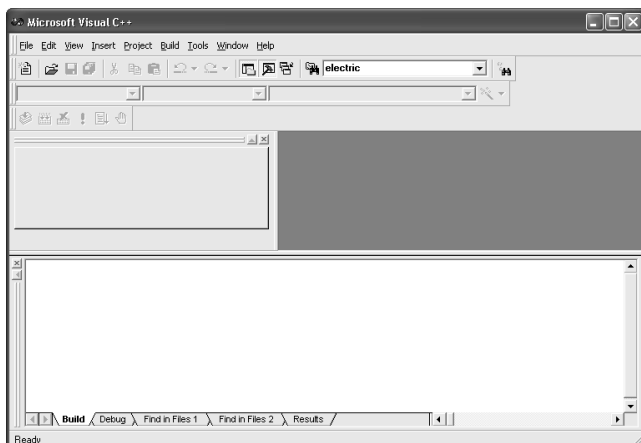
1. Visual C++, Standard Edition. Une version de Visual C++ idéale pour les familiers des environnements de développement tels que Visual Basic, ou pour les connaisseurs des langages C/C++, désirant développer sur la plateforme Windows.
2. Visual C++, Professional Edition. Il propose aux développeurs expérimentés l'ensemble des outils et des fonctionnalités pour la réalisation d'applications très performantes dans le langage C++. Il inclut un compilateur optimisant le code de l'application, des outils visuels pour l'accès à des bases de données professionnelles, et une nouvelle version d'un installateur (InstallShield) pour distribuer facilement vos applications. Les développeurs ont la possibilité d'exploiter la richesse des Microsoft Foundation Classes (MFC), qui est une librairie de composants très populaires, ainsi que les Active Template Library (ATL), permettant l'élaboration de composants réutilisables.
3. Visual C++, Enterprise Edition. Cette édition permet aux développeurs professionnels de développer des applications rapides et distribuées en utilisant des solutions multi-tiers. Tous les éléments de l'édition Professionnelle sont inclus dans l'édition Enterprise : les outils visuels de bases de données, le SQL distant, le débogage, la création de procédures stockées dans un système de gestion de bases de données, le support du système Oracle, et des outils d'entreprises tels que : Microsoft SQL Server 6.5, Visual SourceSafe, Visual Modeler, Microsoft Transaction Server, SNA Server, AS/400 data access drivers, Internet Information Server 4.0, etc.

Ce bref descriptif des différentes éditions de Microsoft Visual C++ 6.0 vous permettra de sélectionner le meilleur produit pour vos besoins spécifiques.

L'environnement de développement C++ est présenté par la figure suivante :

46 Chapitre 2 - Introduction à DirectX 8.1

Figure 2-7 :
*Environnement
de développement
Microsoft Visual
C++ 6.0*

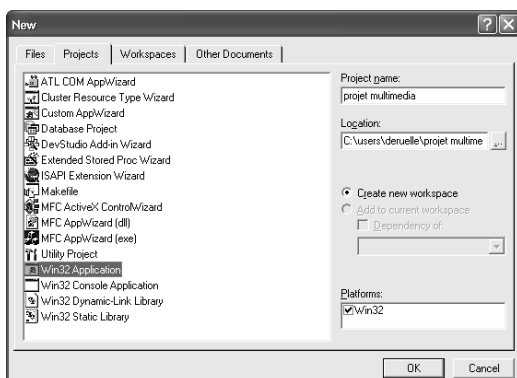


L'environnement de développement C++ permet de réaliser les tâches courantes de gestion de projets, appelées également makefile. La partie supérieure de Microsoft Visual C++ est composée de menus qui sont expliqués dans les parties suivantes. Le but de ces sections est de familiariser le programmeur avec l'environnement Microsoft Visual C++, et de lui fournir une vue des fonctionnalités ainsi que leur emplacement. Cela permet également de préparer un projet multimédia, et de l'enrichir par des applications concrètes au fur et à mesure du parcours des chapitres de l'ouvrage.

2.4. Création d'une application

Le menu **File** et le sous-menu **New** permettent de créer un nouveau projet, comme présenté dans la figure suivante :

Figure 2-8 :
*Création d'un
projet multimédia*



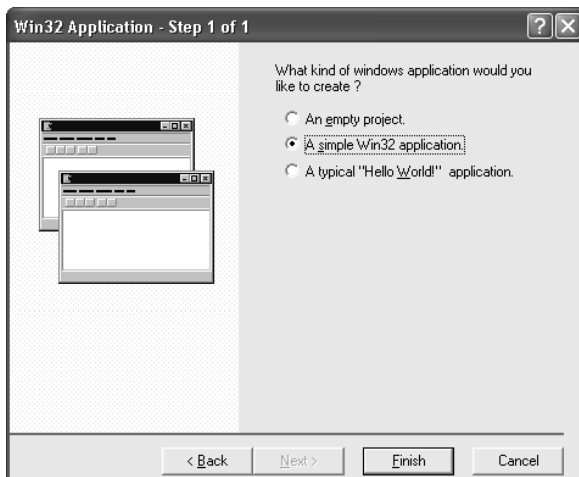
L'élaboration d'une application multimédia nécessite la sélection du type de projet *Win32 Application* dans la partie centrale de la fenêtre. Il est alors obligatoire de saisir le nom du projet, dans la zone de saisie *Project name*, située en haut à gauche de la fenêtre. Le projet sera automatiquement créé dans le répertoire *C:\APPLI\projet multimédia*.

L'environnement Microsoft Visual C++ vous demande de sélectionner le type de projet désiré, parmi le choix suivant :

- Un projet vide, qui ne contiendra aucun fichier.
- Une application Win32, qui contiendra uniquement le point d'entrée de l'application.
- Une application exemple de type *HelloWorld*, qui est généralement le premier programme que l'on crée lorsqu'on débute dans la programmation C++.

Dans le cadre de notre projet, nous choisirons une application Win32, à laquelle nous ajouterons par la suite les fichiers de notre application multimédia.

Figure 2-9 :
Sélection du type
d'application à
créer



Le bouton **Finish** permet de confirmer la création du projet, pour se trouver confronté avec l'interface de l'environnement Microsoft Visual C++ 6.0.

La validation effective du projet engendre la création d'un fichier de projet Microsoft Visual C++ 6.0, possédant une extension *DSP*. Dans le cadre de notre exemple, ce fichier est intitulé *projet multimédia.DSP*, et se trouve par exemple dans le répertoire *C:\APPLI\projet multimédia*. Les modèles de démonstration proposés dans le SDK DirectX possèdent également ce type de fichier. L'ouverture

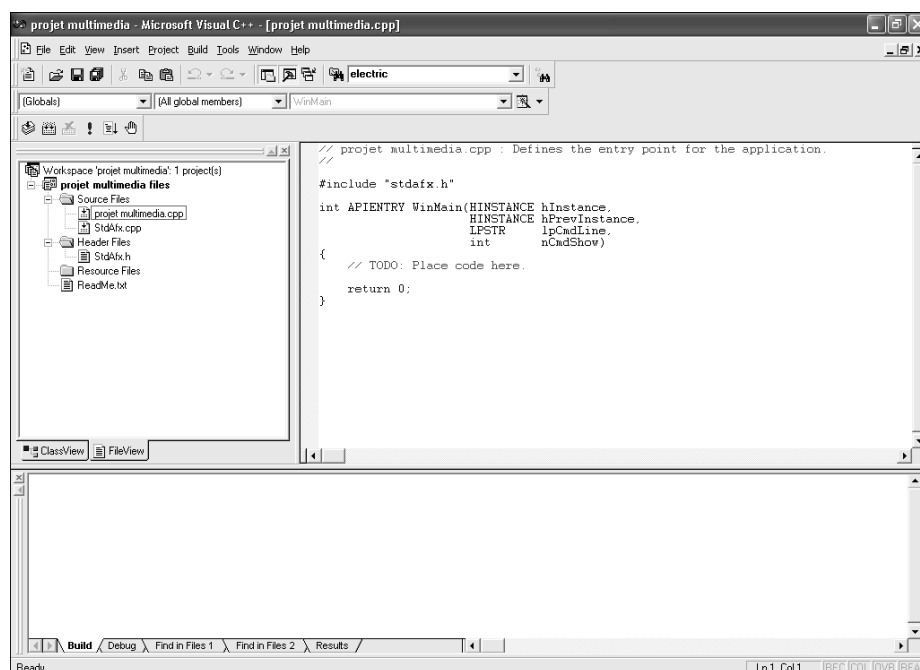
48 Chapitre 2 - Introduction à DirectX 8.1

d'un fichier *DSP* permet à l'environnement de connaître la structure de l'application, en termes de fichiers sources et en-têtes, de ressources et de bibliothèques.

L'interface est composée de 3 parties :

1. *Le gestionnaire de projet*, qui permet de visualiser les classes, les objets, les méthodes, les fonctions définies dans l'application et, également, l'organisation des fichiers, répartis en sources, contenant le code du programme, et en-têtes, définissant des éléments partagés par plusieurs fichiers de l'application.
2. *L'éditeur de code* occupe la partie centrale de l'environnement, et contient les lignes de programmes de notre unique fichier. Ces lignes représentent le point d'entrée du programme, qui est la première fonction qui sera traitée lors de l'exécution de l'application. Il s'agit de la fonction `WinMain`.
3. La troisième partie contiendra les *messages* de l'environnement concernant les détails de compilation, les erreurs éventuelles, les messages d'avertissement, et les informations de débogage.

Figure 2-10 :
*Génération du squelette
des programmes de
l'application multimédia*



Le projet relatif à l'application multimédia est désormais prêt. Il n'y a plus aucun obstacle à votre création. Votre génie artistique n'a plus qu'à s'exprimer pour révolutionner le monde du multimédia. Quelques paramétrages préalables s'imposent cependant.

Dans les sections suivantes, nous étudions les menus de l'environnement Microsoft Visual C++, pour jouer d'une vue globale de ses fonctionnalités et des outils de configuration de projets.

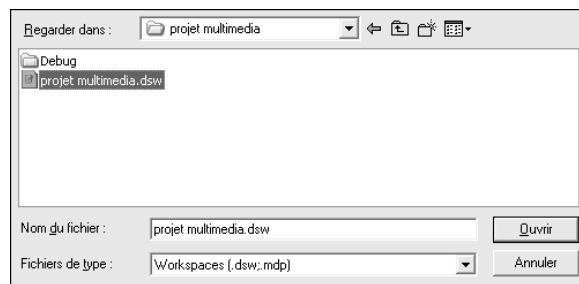
2

Le menu File

Le menu **File** est composé de plusieurs sous-menus pour la gestion de projet C++ :

- Le sous-menu **New** permet de créer un nouveau projet C++, ou des fichiers contenant la définition des classes de l'application.
- Le sous-menu **Open** permet d'ouvrir des fichiers C++, et de les modifier dans l'éditeur de code. Il faut remarquer que les fichiers ouverts de cette manière ne font pas forcément partie d'un projet C++.
- Le sous-menu **Close** ferme le fichier C++, qui est en cours d'édition.
- Le sous-menu **Open Workspace** permet d'ouvrir un projet C++, contenant de multiples fichiers définissant les programmes de l'application. Un Workspace consiste en la définition d'un environnement de travail, qui contient un ensemble de projets ou de fichiers C++ constituant l'application. Le Workspace permet de créer ou de modifier des fichiers C++. Il permet également de visualiser les fichiers dans une interface graphique, décrite dans la section *Menu View*. Un projet C++ est caractérisé par l'extension *.dsp*, et peut être directement chargé dans un Workspace. Le fichier Workspace est caractérisé par l'extension *.dsw*. Ces fichiers peuvent être chargés dans l'environnement Microsoft Visual C++ 6.0 en double-cliquant dessus dans l'Explorateur Windows. Les sous-menus **Save Workspace** et **Close Workspace** permettent, respectivement, de sauvegarder toutes les modifications effectuées au sein du projet, et de fermer le projet courant.

Figure 2-11 :
*Workspace du projet
multimédia*



- Les sous-menus **Save** et **Save All** permettent d'enregistrer les modifications d'un fichier ou toutes celles qui ont été réalisées sur un ensemble de fichiers

50 Chapitre 2 - Introduction à DirectX 8.1

du projet. Cela permet de s'assurer que toutes les modifications sont prises en compte pour la génération du fichier exécutable de l'application.

- Le sous-menu **Exit** quitte l'environnement de développement Microsoft Visual C++, en demandant au préalable au programmeur de confirmer toutes les modifications effectuées sur les fichiers, qui n'ont pas été sauvegardées par le biais des sous-menus **Save** et **Save All**.

Le menu Edit

Le menu **Edit** offre au programmeur des outils de gestion du texte dans les programmes, tels que les fonctionnalités **Undo/Redo**, pour annuler des modifications ou les restituer, ou encore **Cut**, **Copy**, **Paste** pour couper ou copier une partie de texte d'un programme et le coller à un autre endroit. Le menu **Edit** permet également de rechercher un ensemble de mots dans le texte d'un ou de plusieurs programmes C++, par le biais des sous-menus **Find** ou **Find in files**.

Le menu View

Le menu **View** est principalement utilisé pour visualiser le Workspace courant. Ce workspace regroupe les fichiers d'une application multimédia en quatre catégories :

- Les fichiers sources, qui contiennent le code des programmes et les définitions de méthodes. Ils ont une extension *.cpp* ou *.c*.
- Les fichiers d'en-têtes, qui définissent les structures des données du programme, telles que les structures (*struct*), les déclarations de classes (*class*), les variables globales partagées par plusieurs programmes. Ces fichiers ont généralement une extension *.h*.
- Les fichiers de ressources, qui décrivent les éléments des interfaces graphiques et les images de l'application. Ils portent l'extension *.rc*.
- Les autres fichiers, qui sont mis en vrac, peuvent être des fichiers textes, les bibliothèques utilisées par l'application, etc. Les bibliothèques contiennent des fonctionnalités, des services et leurs codes associés, directement utilisables dans les programmes de l'application. Elles peuvent être vues comme des archives de programmes qui sont réutilisables dans d'autres projets. Notamment, les fonctionnalités offertes par l'API DirectX 8.1 sont regroupées en bibliothèques (bibliothèques), et sont invoquées par le programmeur dans son application.

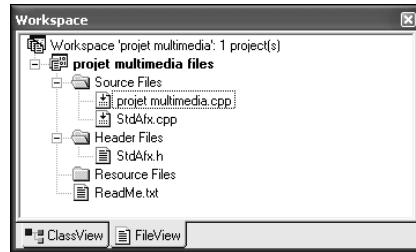


Figure 2-12 :
*Liste des fichiers du
Workspace*

2

Le menu Insert

Le menu **Insert** permet d'insérer des éléments de programme dans une application. Ceux-ci peuvent être de nouvelles classes (**New Class**), de nouvelles interfaces graphiques (**New Form**), de nouvelles ressources (**New Resource**).

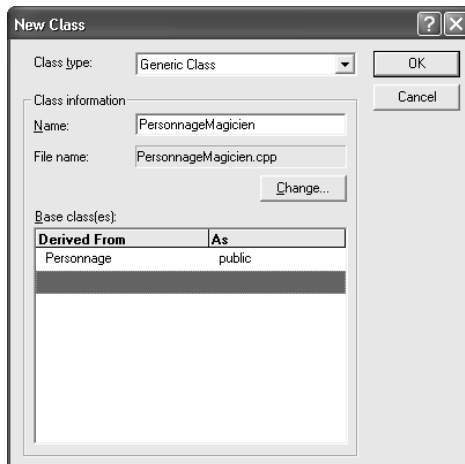


Figure 2-13 :
*Insérer une classe
dans le projet*

Les nouvelles classes sont créées à l'aide de l'interface graphique de création de classe.

Par exemple, nous allons créer deux classes représentant les caractéristiques d'un personnage d'un jeu vidéo. Dans l'interface graphique, nous saisissons le nom de la classe : `Personnage`. Le système crée alors deux fichiers. Le premier est nommé `Personnage.cpp`, et contient le code source des méthodes de la classe `Personnage`. Plus concrètement, les méthodes de cette classe devront définir et implanter le comportement d'un personnage. Le second fichier est nommé `Personnage.h`, et contient la définition de la classe `Personnage` suivante :

```
class Personnage
```

52 Chapitre 2 - Introduction à DirectX 8.1

```
{
public:
    Personnage ();
    virtual ~Personnage ();
};
```

Nous effectuons la même opération avec la seconde classe `PersonnageMagicien`, qui représente les caractéristiques et les comportements des magiciens dans un jeu vidéo. Nous ajoutons simplement la classe `Personnage` dans la partie de Base class - Derived From, pour indiquer que notre magicien est un personnage classique, possédant certaines caractéristiques supplémentaires. Ce mécanisme permet de générer directement un lien d'héritage entre les classes `PersonnageMagicien` et `Personnage`.

```
#include "Personnage.h"

class PersonnageMagicien : public Personnage
{
public:
    PersonnageMagicien();
    virtual ~PersonnageMagicien();
};
```

Les ressources qu'il est possible d'insérer dans une application sont, par exemple, des images (*bitmaps*), des styles de pointeurs de souris, des boîtes de dialogues pour afficher un message, un avertissement ou une erreur, en cours d'exécution.

Ces fichiers ressources sont représentés par des icônes (fichiers images), qu'il est possible d'utiliser dans l'application et, notamment, dans les fenêtres et les boîtes de dialogues créées par l'application. Un éditeur d'images est intégré dans Microsoft Visual C++ 6.0 pour dessiner ces icônes et les sauvegarder dans le projet.

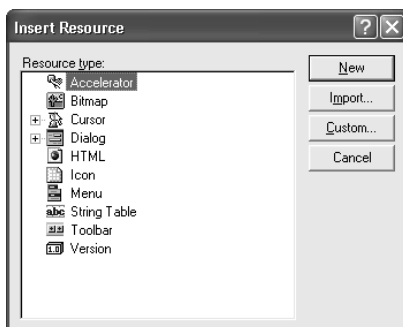
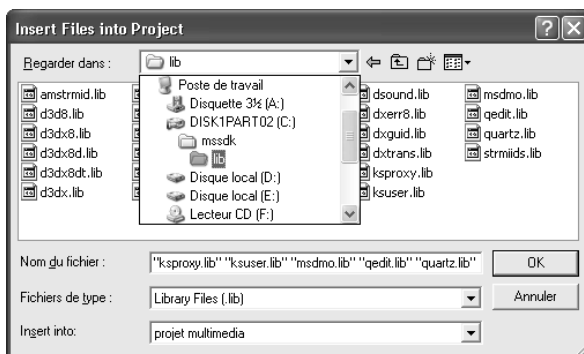


Figure 2-14 :
*Sélection d'un
type de ressources
dans le projet
multimédia*

Le menu Project (Project)

Le menu **Project** permet de gérer les fichiers de l'application. Ce menu comporte des sous-menus très utiles pour ajouter des fichiers à l'application multimédia. Le sous-menu **Add To Project - Files** nous permettra d'ajouter, par exemple, les bibliothèques du DSK DirectX 8.1, nécessaires à l'élaboration de l'application multimédia. Ce sous-menu propose une boîte de sélection des fichiers à insérer dans le projet. Il est nécessaire d'indiquer correctement le type des fichiers à intégrer. Cela est réalisé en choisissant dans la partie *Type*, matérialisée dans une liste déroulante, la ligne *Library Files* ; positionnez-vous ensuite dans le sous-répertoire *Lib* du répertoire contenant le SDK DirectX 8.1 (soit *C:\DXSDK*) ; sélectionnez ensuite les bibliothèques utiles à la mise en œuvre du projet. Le choix des bibliothèques sera détaillé dans chaque chapitre, suivant le type d'application que le programmeur souhaitera créer. Par ailleurs, la boîte de dialogue de sélection de fichier permet d'inclure des fichiers C++ (sources et à en-têtes), des pages web (HTML, ASP, SHTML, etc.), des fichiers ressources, du code pour la connexion et l'exécution de requêtes SQL sur une base de données, des images, des contrôles ActiveX, des bibliothèques dynamiques (DLL), etc.

Figure 2-15 :
*Ajouts des
bibliothèques DirectX
8.1 dans le projet
multimédia*



Le menu **Project** permet de spécifier la configuration du projet multimédia, au niveau des fichiers de codes sources et d'en-têtes, ainsi que des fichiers inclus dans le projet, tels que les bibliothèques. La fenêtre de configuration de projet comporte plusieurs onglets permettant de sélectionner :

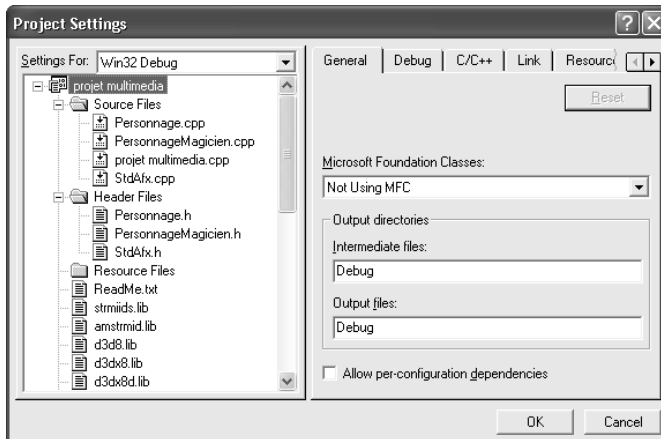
- Les options concernant l'utilisation des bibliothèques de classes Microsoft (*MFC: Microsoft Foundation Classes*) ;
- Le répertoire contenant les fichiers compilés de l'application (*Debug* ou *Release*) ;
- Les options de compilation C++ , telles que les optimisations ;

54 Chapitre 2 - Introduction à DirectX 8.1

- Les informations de débogage, qui offrent un moyen de traquer les dysfonctionnements dans l'application en cours de développement.

L'onglet **Link** permet une configuration de la création du fichier exécutable de l'application, en spécifiant le nom de celui-ci et des bibliothèques nécessaires à sa réalisation.

Figure 2-16 :
*Configuration du
projet multimédia*



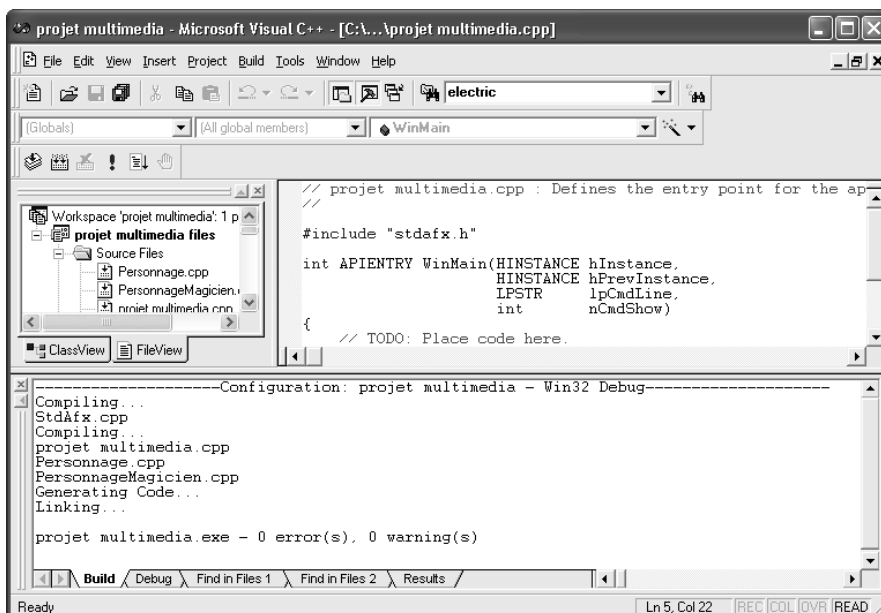
Le sous-menu **Insert Project Into Workspace** du menu **Project** permet également d'inclure le projet en cours dans un autre Workspace, afin de définir un projet plus général et d'en faire une architecture en différents sous-projets. Cela assure une meilleure maintenance de l'application, et la réutilisation de plusieurs projets pour la construction d'un nouveau. Par exemple, il est tout à fait concevable de définir différents projets, tels que *GestionRéseau*, *GestionMusique*, *GestionMoteur3D*, chacun réalisant des tâches et des fonctionnalités spécifiques indépendantes les unes des autres, puis de les inclure dans un Workspace plus général, consistant en la réalisation d'un jeu vidéo 3D multijoueur via Internet.

Le menu Construire (Build)

Le menu **Build** permet de compiler les fichiers de l'application et de créer le fichier exécutable. La compilation est un processus qui consiste à lire les fichiers sources créés par le programmeur, pour les traduire dans un langage compréhensible par la machine. Les fichiers résultant de ce processus sont appelés des fichiers objets, portant l'extension *.obj*. Ils sont ensuite regroupés avec les bibliothèques statiques, d'extension *.lib*, et les bibliothèques dynamiques, d'extension *.dll*, pour produire un fichier exécutable (*.exe*). Ce processus de regroupement des fichiers est communément appelé la phase d'édition des liens (*linkage*).

Le sous-menu **Compile** permet de compiler le fichier en cours d'édition dans l'environnement Microsoft Visual C++, pour générer le fichier objet correspondant. Il est nécessaire de compiler chaque fichier de l'application pour procéder à la phase de *linkage* du fichier exécutable. Toutefois, il est possible de réaliser en une seule action la compilation de tous les fichiers de l'application et le *linkage* produisant le fichier exécutable, par le biais du sous-menu **Build**. La figure suivante montre le résultat de la construction du fichier exécutable (*Build*), dans la fenêtre de messages de l'environnement. Les messages générés précisent généralement les opérations réalisées, les messages d'avertissement et les erreurs éventuelles décelées lors des phases de compilation et de linkage. Un double clic sur un avertissement ou sur une erreur montre directement au programmeur la ligne de code concernée, dans l'éditeur.

Figure 2-17 :
Résultat de la
phase de
construction de
l'application
multimédia



Le menu Outils (Tools)

Le menu **Tools** offre un ensemble de sous-menus constitués d'outils complémentaires et intégrés dans l'environnement Microsoft Visual C++. L'outil *Register Control* permet notamment de spécifier les informations d'enregistrement d'une librairie dynamique (*DLL*) ou d'un composant ActiveX (*OCX*) au niveau du système. Ces composants seront alors directement utilisables par un ou plusieurs fichiers exécutables. L'outil *Visual Component Manager* permet de disposer d'un référentiel visuel des composants d'application utilisables dans l'environnement

Visual Studio. L'outil *Error Lookup* permet d'obtenir des informations sur un code d'erreur retourné par l'application, lors d'un dysfonctionnement. L'outil *Spy++* permet d'obtenir une vue graphique des processus et des threads du système (les programmes en cours d'exécution), des fenêtres et des messages circulant dans le système Microsoft Windows. L'outil *Install Shield Wizard* permet de créer une archive des fichiers du projet, qui sera utilisée pour installer et configurer l'application sur un système client.

Les menus **Window** et **Help** concernent respectivement la gestion des fenêtres, dans l'environnement Microsoft Visual C++, et la recherche d'informations sur les fonctionnalités de celui-ci ou sur le détail des composants standard en œuvre dans l'application. Par composant, nous entendons des classes, des méthodes, des fonctions, etc. La sélection d'un composant et l'appui sur la touche **F1** permet d'obtenir les informations associées, et offre ainsi un mécanisme de raccourci à l'aide. Ces informations sont disponibles uniquement si les modules MSDN Visual Studio sont installés dans le système.

2.5. Les bibliothèques et les fichiers à inclure dans le projet

Dans la section précédente, nous avons créé notre premier projet, appelé projet `multimédia.DSP`. Ce fichier décrit les fichiers sources de l'application, les ressources du projet, et sa configuration en terme de paramètres du compilateur et de l'éditeur de liens (*linker*).

Pour utiliser les composants DirectX, il est nécessaire d'effectuer quelques paramétrages de l'environnement, pour s'assurer que Microsoft Visual C++ saura accéder aux composants DirectX 8.1, au niveau de la compilation et du *linkage*.

Le paramétrage du compilateur est nécessaire pour que Microsoft Visual C++ dispose d'une connaissance claire et précise de la structure des composants DirectX 8.1. Cette connaissance est disséminée dans une multitude de fichiers, d'extensions *.h* et *.lib*. Les premiers sont les fichiers d'en-têtes, et sont censés décrire la structure fonctionnelle des composants DirectX, en précisant quels sont les traitements qu'ils sont capables d'effectuer (liste de méthodes). Cette description est relativement obscure pour les développeurs débutants, mais également pour certains confirmés ! Toutefois, il est important de se rassurer, en se disant que l'environnement Visual C++ est capable de comprendre cette structure, et de puiser d'avantages d'informations dans les seconds types de fichiers, appelés les bibliothèques, qui contiennent les détails de chaque composant, au niveau de la réalisation des traitements.

Il faut indiquer à Microsoft Visual C++ 6.0 le répertoire dans lequel il doit chercher les fichiers constituant la description des composants DirectX, c'est-à-dire le répertoire contenant les fichiers d'en-têtes. Ce répertoire doit figurer en premier dans la liste de recherche ; sinon, il provoquera des erreurs lors de la compilation de l'application. Pour vérifier et corriger l'ordre de recherche dans l'inclusion des fichiers d'en-têtes, il faut choisir le menu **Options** à partir du menu principal **Tools**, et sélectionner l'onglet **Directories**. La boîte de dialogue suivante apparaît à l'écran :

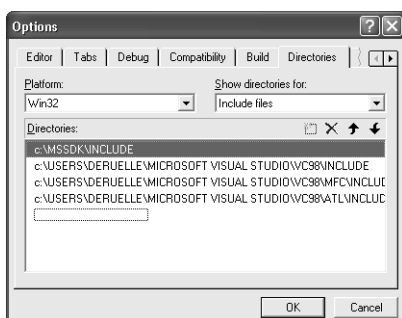


Figure 2-18 :
*Configurer le
projet pour
l'accès aux
fichiers DirectX*

Le premier chemin indique le répertoire qui contient les fichiers d'en-tête DirectX. Le chemin par défaut dans lequel ils sont installés est *C:\DXSDK\Include*. S'il n'est pas mentionné dans la liste, il faut l'ajouter en cliquant dans le rectangle vide, en bas de la liste des chemins, puis le déplacer vers le haut à l'aide des flèches présentes dans la boîte de dialogue.

L'environnement dispose alors d'une connaissance sur les fonctionnalités des composants DirectX ; il faut maintenant lui indiquer le répertoire où sont situés les fichiers contenant les détails d'implémentation de ces fonctionnalités.

À partir de la boîte de dialogue précédente, il faut choisir les *Library files* dans la section *Show directories for*. Microsoft Visual C++ indique la liste des chemins pour la recherche des librairies. Le premier chemin doit contenir les libraires fournies par le SDK DirectX 8.1. Le chemin par défaut est *C:\DXSDK\Lib*.

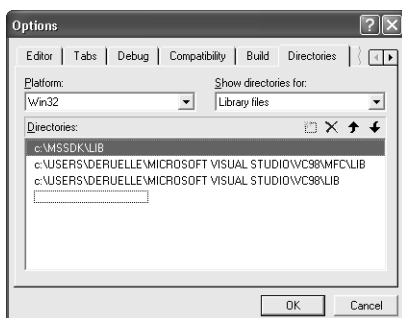


Figure 2-19 :
*Configurer le
projet pour
l'accès aux
librairies DirectX*

Il est désormais nécessaire d'inclure les bibliothèques DirectX 8.1 dans notre projet, pour importer les composants et leurs fonctionnalités dans notre application multimédia. Cette tâche est réalisée par le menu **Project** et l'option *Add to Project -> Files*, puis par la sélection des bibliothèques DirectX 8.1 contenues dans le répertoire par défaut *C:\DXSDK\Lib*. Une autre solution consiste à les ajouter dans les paramètres de l'application, accessibles par le menu **Project** et l'option *Settings*, puis en sélectionnant l'onglet correspondant aux paramètres du linker, et en ajoutant les bibliothèques manuellement.

2.6. Notions de base de la programmation DirectX 8.1 en C++

De nombreux développements d'applications DirectX dans les langages C/C++ impliquent la connaissance et l'utilisation de techniques de programmation liées à l'environnement Windows. Certains de ces aspects sont obscurs pour bon nombre de programmeurs. Cette section propose un bref rappel sur le modèle de programmation objet *COM*, et l'utilisation de fonctions *Callback*, qui sont nécessaires pour développer une application multimédia.

Le modèle de composants COM

Le modèle de composant objet (COM) est un modèle de programmation orienté objet, fréquemment par les concepteurs d'applications sous Windows, et particulièrement dans DirectX, du fait de sa compatibilité avec le cœur de ce dernier, dont les composants sont des objets COM.

Il apparaît alors nécessaire de comprendre les principes de base du modèle de composants COM et des techniques de programmation qui lui sont associées. Il faut cependant savoir que COM a la solide réputation d'être difficile et très complexe ; c'est pourquoi nous allons exposer clairement ses concepts, en les vulgarisant du plus que nous pouvons.

Il existe deux manières de programmer en exploitant le modèle COM :

1. La première consiste à mettre en œuvre des objets COM existants. Le procédé est aussi simple que de manipuler des objets C++. En effet, il suffit de récupérer le composant COM, à l'aide du système Windows, sous la forme d'un objet C++, puis d'exploiter les méthodes qu'il propose.

2. La seconde consiste à implémenter ses propres objets COM. Cela est beaucoup plus difficile, et ne se résume pas à définir une simple classe et les liens d'instanciations pour créer les objets.

La plupart des applications fondées sur DirectX 8.1 se contentent d'exploiter les ressources d'objets COM existants. Les développeurs n'ont alors aucun besoin de connaître parfaitement le modèle de composants COM.

2

Les objets COM peuvent être vus comme des boîtes noires, utilisées par les applications pour réaliser certaines tâches. Ils sont souvent implémentés sous la forme de bibliothèques dynamiques, connues sous le nom de DLL. Une application interagit avec des objets COM de la même manière qu'avec des objets C++, aux quelques différences suivantes près :

- Les objets COM appliquent strictement l'encapsulation, tout comme les objets C++. Il n'est pas possible de créer simplement un objet et d'invoquer des méthodes publiques. Les méthodes publiques d'un objet COM sont groupées dans une ou plusieurs **interfaces**. Pour appliquer une méthode, il est nécessaire de créer l'objet et d'obtenir l'interface appropriée pour lui. Une interface contient typiquement un ensemble de méthodes permettant d'accéder à une caractéristique particulière de l'objet. Par exemple, l'interface `IDirect3DCubeTexture8` contient les méthodes qui autorisent la manipulation des textures d'un cube. Les méthodes qui ne font pas partie de l'interface ne sont pas accessibles.
- Les objets COM ne sont pas créés de la même manière que des objets C++. Il existe plusieurs façons de créer un objet COM, mais toutes impliquent des techniques spécifiques. L'API de Microsoft DirectX 8.1 propose une variété de méthodes et de fonctions pour la création automatique des objets DirectX, via le modèle COM.
- Le contrôle de la durée de la vie d'un objet COM dans l'application nécessite également l'utilisation des techniques spécifiques à COM.
- Les objets COM peuvent être écrits et rendus accessibles dans divers langages, tels que C++ et Visual Basic. Il n'est pas nécessaire de disposer du code source d'un objet COM pour le faire fonctionner. Il faut uniquement connaître son interface (la liste de ses méthodes).

Il est important de comprendre la distinction entre les objets et les interfaces. Les deux termes ne sont pas interchangeables. Un objet peut exposer plusieurs interfaces. Par exemple, tous les objets exposent l'interface `IUnknown`, mais ils possèdent généralement une interface supplémentaire, spécifiant les véritables fonctionnalités de l'objet. Pour utiliser une méthode particulière de l'objet, il est nécessaire d'obtenir la bonne interface, qui la contient. La même interface peut être

60 Chapitre 2 - Introduction à DirectX 8.1

exploitée par plus d'un objet. Une interface est un groupement de méthodes qui réalisent un ensemble d'opérations. La définition de l'interface spécifie uniquement la signature de la méthode. L'implémentation effectuée par la classe de l'objet consiste à définir le corps de la méthode spécifiée dans l'interface. L'objet COM est chargé d'implémenter les méthodes, en respectant la spécification donnée par l'interface, c'est-à-dire qu'elle doit conserver les mêmes types de retours, noms et listes de paramètres.

Toutes les méthodes des objets COM retournent une valeur indiquant si le traitement s'est déroulé correctement ou si une erreur s'est produite. Le type de la valeur retournée est `HRESULT`. Suivant les types de composants DirectX et leurs méthodes associées, les valeurs retournées sont sensiblement différentes. Par convention, les valeurs représentant un résultat correct sont préfixées par `S_`, tandis que les valeurs de retour erronées sont préfixées par `E_`. Par exemple, les deux valeurs les plus courantes dans les programmes sont `S_OK` et `E_FAIL`, qui indiquent respectivement un succès ou une erreur.

Le programme suivant montre le traitement d'une fonction, et la gestion de son code de retour :

```
HRESULT hr;
Hr = initialiserApplication();
if(hr == E_FAIL)
{
    // l'application n'est pas initialisée correctement
    // nous quittons l'application.
    exit(-1);
}
else
{
    // l'application est initialisée correctement,
    // nous pouvons continuer l'exécution du programme.
}
```

Le programme précédent n'est pas vraiment correct, dans la mesure où les méthodes des composants DirectX retournent souvent des codes d'erreurs différents de `E_FAIL`, tels que `D3DERR_INVALIDCALL`, pour spécifier que l'appel à la méthode est invalide, ou encore `E_OUTOFMEMORY`, lorsque le système n'a plus suffisamment de mémoire pour exécuter la méthode ou créer des objets. Dans ce cas, le programme considérera que les codes renvoyés indiquent un succès de l'exécution de la méthode. En effet, toutes valeurs différentes de `E_FAIL` retournées par la méthode valideront le test, et exécuteront alors la partie définie dans la section `else`. Deux fonctions spécifiques, appelées `macros`, permettent de résoudre cette incohérence :

1. La macro `SUCCEEDED` permet de répondre à la question suivante : la méthode s'est-elle correctement exécutée ? Cette macro répondra `TRUE` (vrai, oui) pour indiquer un succès ou `FALSE` (faux, non) en cas d'erreur.
2. La macro `FAILED` est l'inverse de `SUCCEEDED` ; elle permet de répondre à la question suivante : y a-t-il eu une erreur lors de l'exécution de la méthode ? Cette macro répondra `TRUE` (vrai, oui) pour indiquer une erreur ou `FALSE` (faux, non) si la méthode s'est exécutée correctement.

Le programme précédent peut alors être corrigé de la manière suivante :

```
HRESULT hr;
Hr = initialiserApplication();
if( FAILED(hr) )
{
    // l'application n'est pas initialisée correctement
    // nous quittons l'application.
    exit(-1);
}
else
{
    // l'application est initialisée correctement,
    // nous pouvons continuer l'exécution du programme.
}
```

Le modèle COM spécifie que la définition d'une interface ne doit pas changer lorsqu'elle a été rendue publique, afin de ne pas générer des conflits ou des dysfonctionnements lors de l'exécution des applications. Par exemple, il n'est pas possible d'ajouter une méthode (une fonctionnalité) à une interface existante. Pour pallier cette contrainte, il est nécessaire de créer une nouvelle interface. Toutefois, il est d'usage que celle-ci conserve toutes les méthodes de l'ancienne interface, en plus des nouvelles, afin de garantir une compatibilité avec les versions antérieures, et d'assurer la pérennité des applications.

Les différentes générations d'une interface se traduisent, dans DirectX 8.1, par des versions régulières révisées du SDK, et de l'environnement d'exécution (Runtime) des applications multimédias. Ainsi, les applications anciennes continuent de s'exécuter normalement, alors que celles développées à l'aide de la dernière version peuvent tirer avantage des caractéristiques des nouvelles interfaces, et ainsi bénéficier des dernières innovations technologiques.

La création d'un objet COM

Il existe plusieurs techniques pour créer un objet COM : par la fonction `CoCreateInstance` ou une fonction DirectX. Toutefois, il faut veiller à initialiser COM, afin de le prévenir que nous allons créer des objets COM. Cette initialisation

62 Chapitre 2 - Introduction à DirectX 8.1

est réalisée à l'aide de la fonction `CoInitialize` ; à l'inverse, lorsque l'application termine son exécution, elle doit prévenir COM de détruire les objets DirectX, pour libérer les ressources qu'ils occupent. La destruction des objets COM est réalisée à l'aide de la fonction `CoUninitialize`. Les appels à ces deux fonctions interviennent, respectivement, au tout début et à la fin du programme, ou à tout autre endroit provoquant une fin plus ou moins brutale de l'application.

Le programme ci-dessous illustre la création et la destruction des objets COM DirectX, notamment celles d'un composant `DirectMusic` faisant partie de `DirectX Audio`.

```
int APIENTRY WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow )
{HRESULT hr;

    // Initialiser COM
    CoInitialize(NULL);

    // Créer un objet DirectX pour la gestion de la musique
    hr = CoCreateInstance( CLSID_DirectMusicLoader, NULL, CLSCTX_INPROC,
IID_IDirectMusicLoader8, (void**) )
    if (FAILED( hr ))
    {   MessageBox( NULL, TEXT("Impossible de créer l'objet DirectX. "),
TEXT("Exemple"), MB_OK | MB_ICONERROR );
    }

    // lire des morceaux de musiques
    // demander à COM de détruire les objets COM
    CoUninitialize();
    // fin de l'application
    return 0;
}
```

La création d'un composant DirectX, en exploitant les fonctions prévues à cet effet, est une tâche plus facile. Il suffit de fournir la variable représentant le composant dans le dernier paramètre de la méthode, pour que celle-ci s'occupe de son initialisation.

Dans l'exemple suivant, nous créons un objet COM `Direct3D`, représentant les fonctionnalités d'une caret graphique présente dans la machine. Cet objet est appelé `g_pd3dDevice` ; il est passé à la méthode `CreateDevice` pour effectuer son instanciation dans le modèle COM. Les autres paramètres seront explicités ultérieurement, dans le chapitre consacré à `Direct Graphics`.

```
IDirect3DDevice8 *g_pd3dDevice = NULL;
...
if( FAILED( g_pd3D->CreateDevice(D3DADAPTER_DEFAULT,
3DDEVTYPE_HAL,
```

```
return E_FAIL;

hWnd,
D3DCREATE_SOFTWARE_VERTEXPROCESSING,
&d3dpp,
&g_pd3dDevice )))
```

Les fonctions Callback

Une fonction *callback* est essentiellement un gestionnaire d'événements qui est implémenté par une application, et invoqué par le système Windows. Les applications Microsoft Windows implémentent de multiples fonctions callback, chacune étant désignée pour un ensemble particulier d'événements. Lorsqu'un événement survient, le système notifie l'application en exécutant la fonction callback appropriée. Cette dernière possède généralement un paramètre de type liste, que le système peut utiliser pour communiquer à l'application de plus amples informations concernant l'événement survenu. L'exemple le plus simple de fonction callback est la *procedure window*. Cette fonction est mise en œuvre par le système pour transférer les messages Windows aux applications qui s'exécutent, et réaliser des actions suivant la nature de chaque message. Les messages sont généralement associés à des actions de l'utilisateur, effectuées dans l'interface graphique de l'application (cliquer sur un bouton, par exemple).

Microsoft DirectX utilise les fonctions callback pour plusieurs raisons. Par exemple, considérons un système qui prend en charge de multiples périphériques d'entrée. Le composant Microsoft `DirectInput` représente chaque périphérique (souris, joystick, etc.) par un objet le décrivant, et qui contient les détails de ses caractéristiques. L'application devra inévitablement énumérer les périphériques d'entrée disponibles, et examiner les objets afférents, afin de trouver celui qui correspond à des critères spécifiés par l'utilisateur. Par exemple, un critère pourrait être défini de telle sorte que le périphérique (joystick) doit gérer les forces de rétroaction. Cette énumération est effectuée dans l'application en implémentant la fonction callback, appelée `DIEnumDeviceObjectsCallback`, spécifique aux composants `DirectInput`. Chaque composant DirectX 8.1 dispose de fonctions callback pour la gestion d'événements qui lui sont propres. Nous verrons dans chaque chapitre les fonctions callback à implémenter, dans le cadre du développement d'une application réelle, illustrant la mise en œuvre de chaque type de composant DirectX.

Dans notre exemple, le processus d'énumération des périphériques sera effectué lors de l'appel à la méthode `EnumObjects`, définie dans l'interface `IDirectInputDevice8`, et en lui fournissant en paramètre la fonction callback à exécuter. Le système invoquera cette dernière une unique fois pour chaque


```
// Les détails d'implémentation de la fonction
BOOL CALLBACK EnumAxesCallback( const DIDEVICEOBJECTINSTANCE* pdidoi,
                                VOID* pContext )
{
    // Traiter les informations passés par DirectInput
    // dans les deux paramètres
}
```

2.7. Architecture de DirectX 8.1

DirectX 8.1 est un ensemble de composants répartis et implémentés dans des bibliothèques. Ces composants sont des objets C++ spéciaux, appelés objets COM, qui interagissent avec les composants matériels de la machine et les éléments du programme. La figure suivante montre l'architecture des composants DirectX.

Le module Components représente les objets COM DirectX, qui sont fondés sur les modules DirectX Media et DirectX Foundation.

Le module DirectX Media comprend la définition des composants DirectX, pour simplifier le développement de l'application en proposant des outils qui accélèrent la création et la lecture de contenus multimédias, et facilitant l'intégration de tels éléments.

Le module DirectX Foundation fournit des composants internes et des mécanismes spécifiques, pour s'assurer que les applications développées fonctionneront sur n'importe quel ordinateur sous Windows, tout en s'assurant qu'elles tireront parti de tous les avantages du système. Ce module est alors proche des configurations matérielles et réseaux.

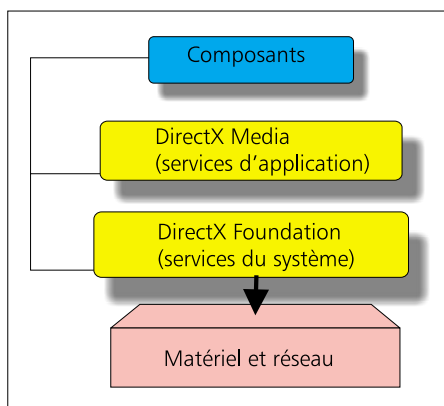


Figure 2-20 :
Présentation de
l'architecture
DirectX 8

66 Chapitre 2 - Introduction à DirectX 8.1

Cette architecture fournit aux développeurs de nouvelles opportunités de créativité et d'innovation, en leur permettant de se focaliser sur la création, et en les exonérant de questions relatives à l'identification de la carte son ou du processeur graphique installé dans l'ordinateur de l'utilisateur. Comme DirectX a été conçu pour prendre en charge de futures innovations dans les domaines matériel et logiciel, les développeurs et consommateurs peuvent être confiants et rassurés ; alors que la technologie évolue, ils vont continuer à profiter de performances optimales de leurs applications.

Les composants de DirectX 8.1

Microsoft DirectX 8.1 est découpé en six composants, chacun possédant ses propres caractéristiques, et une conception adaptée pour des traitements spécifiques.

1. DirectX Graphics combine les composants Microsoft DirectDraw et Microsoft Direct3D des versions antérieures de DirectX dans une unique API, qui peut être utilisée pour tous types de développements graphiques. Le composant inclut la librairie d'utilitaires `Direct3DX`, qui simplifie de nombreuses tâches de programmation graphique.
2. DirectX Audio combine les composants Microsoft DirectSound et Microsoft DirectMusic développés dans les versions précédentes de DirectX, permettant de programmer des applications incluant du son et de la musique.
3. Microsoft DirectInput permet un support d'une multitude de périphériques d'entrée, et notamment des joysticks à force rétroactive (technologie force feedback).
4. Microsoft DirectPlay assure la prise en charge d'applications réseau suivant le modèle client-serveur. Cette fonctionnalité est intéressante dans le cadre du développement de jeux réseaux multijoueurs sur Internet.
5. Microsoft DirectShow offre des fonctionnalités pour la capture et la lecture de flux de données multimédias avec de hautes performances. Un exemple d'application DirectShow, que nous étudierons, sera consacré à la lecture de film DVD.
6. Microsoft DirectSetup est une API simple, permettant l'installation des composants DirectX sur une machine. Il s'agit d'un outil de déploiement de DirectX 8.1.

Apports de la nouvelle version de l'API

2

Le SDK DirectX 8.1 a subi de nombreuses évolutions en profondeur au niveau de ses interfaces de programmation (API). L'une des plus marquantes concerne le passage de la version DirectX 7 à la version DirectX 8.1, car elle a nécessité une refonte en profondeur de la conception du système. De ce fait, de nombreux développeurs de jeux vidéo et de moteurs 3D doivent redéfinir l'architecture de leurs applications, et effectuer un nouveau développement en considérant les modifications et les évolutions majeurs introduits dans la version DirectX 8.1.

Cette section nous montre les apports de la nouvelle version de DirectX 8.1, concernant chaque composant.

Apport dans DirectX Graphics : intégration de DirectDraw et Direct3D

Les versions précédentes de DirectX définissaient deux composants distincts :

1. DirectDraw, pour la gestion des graphiques en deux dimensions ;
2. Direct3D, pour les univers virtuels en 3D dimensions.

Microsoft DirectDraw et Microsoft Direct3D ont fusionné dans un nouveau composant, appelé **DirectX Graphics**. Les API ont été mises à jour, pour rendre plus simple l'utilisation du graphisme dans les applications multimédias et également pour prendre en charge les dernières fonctionnalités des cartes graphiques performantes existant sur le marché et les nouvelles technologies 3D.

Apport dans DirectX Audio : intégration de DirectMusic et DirectSound

Microsoft DirectMusic et Microsoft DirectSound sont plus fortement couplés dans cette nouvelle version de DirectX. Les fichiers son Wave (extensions *.wav*) ou les fichiers de ressources musicales peuvent être chargés par le composant chargeur DirectMusic. Ces fichiers sont lus par l'interprète du composant DirectMusic, prenant en compte les notes de synchronisation MIDI. Ces différents composants seront expliqués plus en détail dans le chapitre consacré à Direct Audio.

Apport dans DirectInput

Le composant Microsoft DirectInput possède une nouvelle caractéristique majeure : la correspondance d'actions (*l'action mapping* en anglais). La correspondance d'actions permet d'établir une connexion entre des actions en entrée et des périphériques d'entrée, qui ne dépendent pas forcément de l'existence d'objets périphériques particuliers. Cela simplifie le traitement des informations du périphérique, et réduit le besoin de disposer de pilotes pour les périphériques standard de jeu, ou pour des périphériques d'entrée exotiques (inconnus pour tous), et proposent des interfaces utilisateurs de configurations personnalisées dans les jeux.

Apport dans DirectPlay

Le composant DirectPlay a été mis à jour, pour offrir de nouvelles caractéristiques et améliorer sa facilité d'intégration dans des applications réseaux. Ainsi, il est plus facile à manipuler, et permet d'accroître les fonctionnalités de l'application multimédia ; en particulier, DirectPlay intègre désormais la communication vocale entre des joueurs ou des participants à un jeu ou à un forum de discussion.

Apport dans DirectShow

Le composant DirectShow est complètement nouveau dans le SDK DirectX 8.1. En effet, les versions précédentes des SDK DirectX ne contenaient pas le composant DirectShow. Ce dernier était une application à part entière, qu'il était nécessaire d'intégrer dans l'application multimédia pour la lecture de flux de données multimédias. Cela lui permet notamment de lire des flux vidéo et son présents sur un DVD, ou encore de la musique dans le format MP3.

2.8. Conclusion

Ce chapitre a présenté une description de DirectX 8.1, abordant de manière didactique l'installation du SDK DirectX 8.1. Ce processus est primordial pour disposer des outils adéquats, des bibliothèques définissant les composants DirectX et de la documentation afférente. Nous avons également détaillé les caractéristiques des outils DirectX, ainsi que leurs fonctionnalités.

Cela vous a permis de vous familiariser avec Microsoft Visual C++, de créer un projet et de configurer cet environnement de développement pour prendre en compte les bibliothèques et les composants DirectX.

Finalement, nous avons introduit l'ensemble des composants DirectX, et montré leurs évolutions par rapport aux anciennes versions du logiciel. Vous maîtrisez maintenant assez ces composants pour commencer à appréhender les concepts qu'ils représentent. Nous avons remarqué que plusieurs d'entre eux ont été fusionnés pour permettre une plus grande simplicité de programmation et assurer de meilleures performances. Par ailleurs, le composant DirectShow est désormais intégré à DirectX et accroît les capacités d'une application pour lire de la musique MP3 et des films ou animations sur un support DVD.

