1. DirectX8® Framework

1.1. A cosa serve il framework

Framework come **Libreria**

Il termine framework letteralmente significa cornice di lavoro: essenzialmente è un insieme di funzioni che svolgono tutte le operazioni comuni necessarie a realizzare una applicazione DirectX8. Questa "libreria" è stata realizzata con lo scopo di rendere più facile realizzare esempi di codice, evitando di riscrivere ogni volta tutto il codice relativo all'inizializzazione, creazione delle devices, etc. etc.. Si preferisce quindi scrivere il codice comune a tutte le applicazioni 3D una sola volta e riutilizzarlo in seguito. Lo scopo del framework è quello di costruire una interfaccia flessibile e **standard** per la produzione di esempi, la Microsoft infatti incoraggia così tanto il suo utilizzo, che anche l'SDK di case come la **ATI Technologies** contiene una versione del framework basata su quello Microsoft, con piccole aggiunte specifiche per il loro hardware. È quindi molto importante capire il funzionamento del framework, almeno nelle sue linee generali, per potere apprezzare al meglio gli esempi presenti nei vari SDK che si trovano in rete.

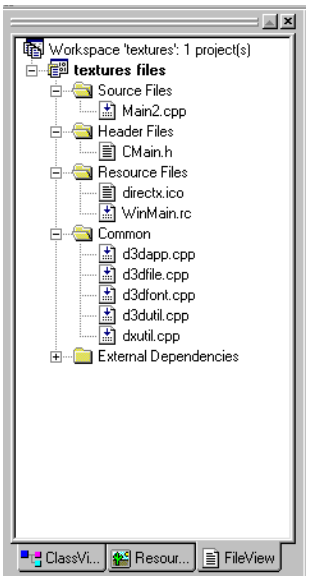
Utilizzare il framework per **esempi di codice**

Il framework stesso è inoltre un completissimo esempio sull'enumerazione e sulla gestione delle devices DirectX8 e sulle funzioni generali che deve svolgere una **window procedure** per gestire un ambiente DirectX.

1.2. Struttura del framework

Framework e **C++**

Il framework è stato strutturato per essere utilizzato con il **C++,** perché con la possibilità di gestire funzioni virtuali ed ereditarietà, si raggiunge il duplice obiettivo di facilità di utilizzo ed efficienza. In generale quindi il framework è costituito da una classe principale che viene ereditata dalla propria applicazione e di cui è possibile ridefinire una serie di metodi che costituiscono l'**interfaccia** del framework stesso.



Come prima cosa è necessario sapere quali file contengono il codice del framework in modo da poterli includere nei propri progetti, la directory cercata è **C:\mssdk\samples\Multimedia\Common**[1], che a sua volta contiene due sottocartelle in cui si trovano gli header ed i file sorgente. Tutti gli esempi dell'SDK si riferiscono al codice contenuto in questa cartella, all'interno dei progetti degli esempi dell'SDK i file vengono infatti identificati mediante il loro percorso relativo. Per non andare ad interferire con gli esempi presenti si consiglia quindi di copiare fisicamente i file del framework in una sottocartella del vostro progetto. È bene inoltre separare i file in sotto gruppi per ottenere una migliore organizzazione del progetto stesso. Un esempio è rappresentato nella figura a fianco.

Dalla figura si evince subito che il codice relativo all'esempio è costituito solamente dal file **Main2.cpp**; i cinque file d3dapp.cpp, d3dfile.cpp, etc. sono invece quelli necessari al corretto funzionamento del framework (in realtà non sempre si utilizzano le funzioni di tutti e cinque). Il più importante di essi è il **d3dapp.cpp** che contiene la definizione della classe **CD3DApplication**, che gestisce tutto il framework; gli altri contengono invece funzioni di utilità generale impiegate anche dalla CD3Dapplication stessa. È necessario fornire inoltre un file di risorse che contenga l'icona delle applicazioni DirectX e la DialogBox per gestire il cambio di risoluzione e la possibilità di mettere in full screen; per fare questo basta copiare da uno qualsiasi degli esempi presenti i file **winmain.rc** e **directx.ico** nella cartella del proprio progetto e poi includerli nel progetto stesso. Come ultima cosa, nella finestra delle opzioni del progetto, è necessario specificare nelle impostazioni del preprocessore la cartella in cui sono stati posizionati gli header file del framework.

[1]La directory indicata è valida si è istallato l'SDK nella directory di default, altrimenti si sostituisca C:\mssdk con la cartella specificata al momento dell'istallazione.