Institut Sabadell

Generalitat de Catalunya

Departament d'Ensenyament

Unió Europea

Fons Social Europeu

CICLE FORMATIU GRAU SUPERIOR

DESENVOLUPAMENT D'APLICACIONS MULTIPLATAFORMA

Manual d'usuari

Curs: 2012-2013

Memòria Presentada per:

Alberto Lopez Sanchez

Ruben Bagan Benavides

Sabadell, 31 de Maig de 2013

Índex

Requisits de sistema:

El següent enllaç serveix per descarregar el Microsoft DirectX SDK de la pagina oficial: http://www.microsoft.com/en-us/download/details.aspx?id=6812

- Supported Operating System
  - Windows 7, Windows Server 2003, Windows Server 2008, Windows Vista, Windows XP
- És important tindre una gràfica que suporti Microsoft DirectX 10 mínim.

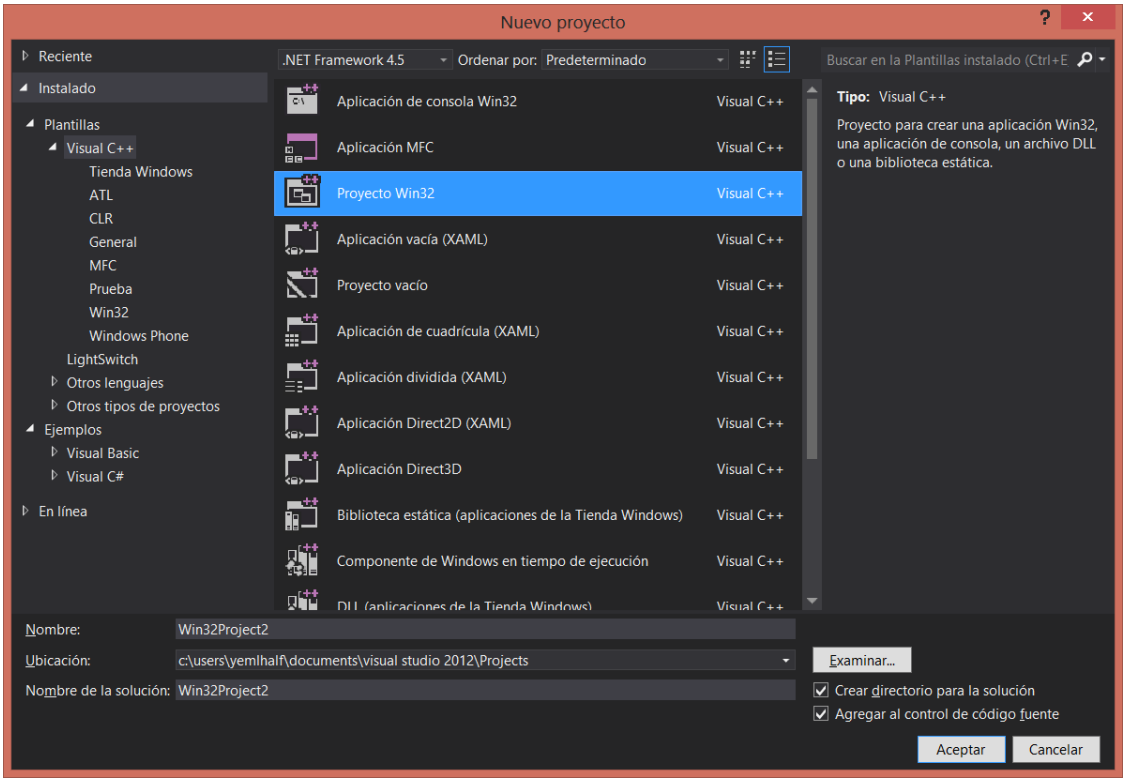
Demo project setup on Visual Studio 2012

S'assumeix que s'ha instal·lat de forma correcta la ultima versió del Microsoft DirectX SDK per poder començar a crear i utilitzar les demos. Si no es segueixen bé tots els passos a l'hora de compilar possiblement no s'obtindrà el resultat desitjat.

Crear a Win32 Project

Primer inicia el Microsoft Visual Studio 2012 i al dirigeix-te al menú **Archivo > Nuevo > Proyecto**.

S'obrirà un quadre de diàleg. Selecciona **Visual C++ > Win32** de la secció de plantilles de Visual C++ a la part esquerra. En la part dreta selecciona un projecte de tipus Win32 Project. Després dona un nom al projecte i la ubicació on vols guardar-lo.



Al seguir tots aquests passos s'obrirà un nou quadre de diàleg. En la part esquerra del quadre dins de la secció **Tipo de aplicación** tenim varies opcions, els tutorials estan realitzats triant la opció de **Aplicación de Consola** per obtenir una consola CMD on poder visualitzar sortides de text. En cas que no és volgués una terminal podem triar **Aplicación para Windows** però s'ha d'afegir el següent codi al fitxer on està ubicat el main.

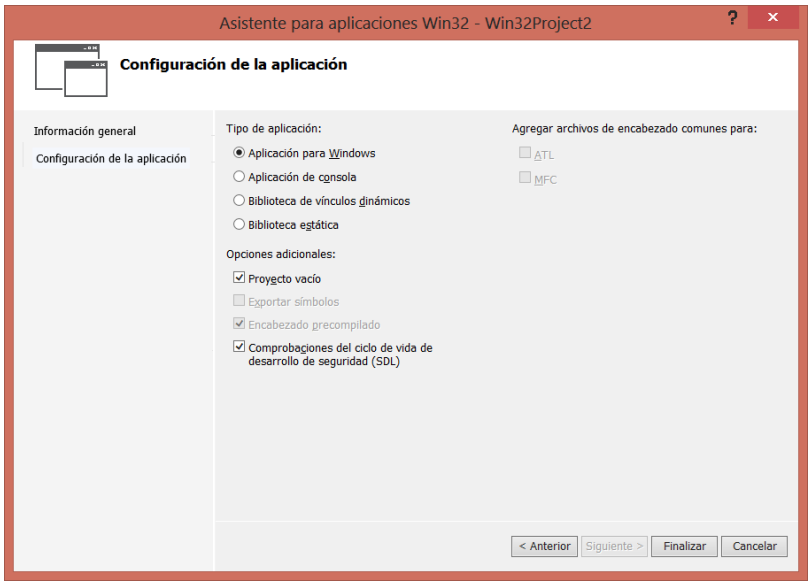
Substituir el tradicional int main () per el següent codi:

```
int CALLBACK WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{

}
```

Exemple de fitxer:

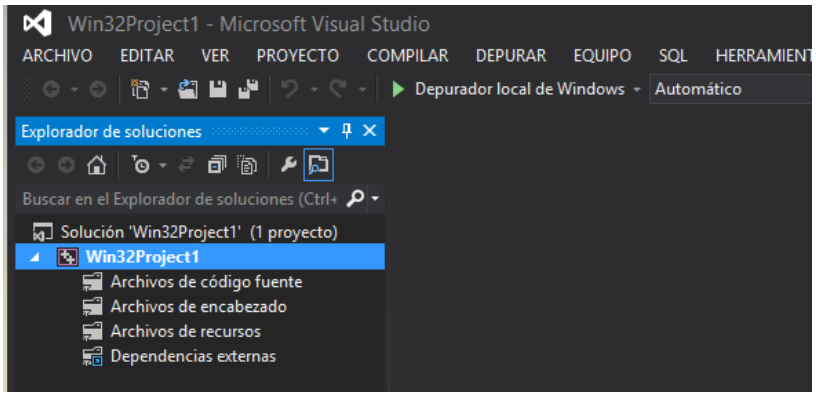
```
#include "Dementia.h"
#include "Window.h"
#include <iostream>

int CALLBACK WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{
    Dementia::Window window(1,800,600, L"UserDemo1", false, true);
    window.init();

    while (window.run(NULL))
    {
    }

    return 0;
}
```

La següent secció és **Opciones Addicionales** i aquí seleccionarem **Proyecto Vacio**. Un cop marcades totes les opcions finalitzarem l'assistent.



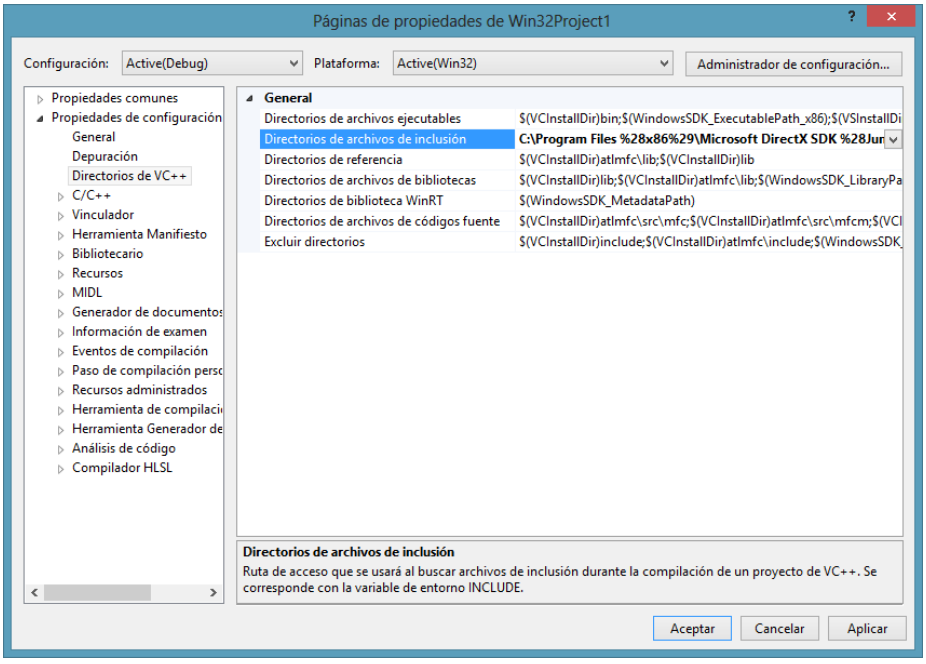
Vincular les Microsoft DirectX i Dementia Libraries

Un cop tenim el projecte creat farem clic dret sobre ell (al projecte no a la solució) en aquest cas té de nom Win32Project1, i dins del total d'opcions que tenim triarem la opció de **Propiedades.**



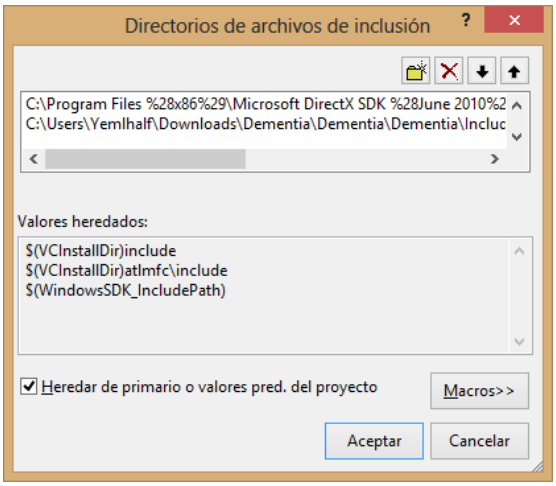
Al fer-ho se'ns obrirà les propietats del projecte. Tenim que configurar-lo per tal que agafi les llibreries de Dementia. Per això desplegarem del panell lateral esquerra la secció de **Propiedades de configuración** i dins d'aquí seleccionar l'opció **Directorios de VC++.**

Apareixerà una sèrie de registres on tenen uns valors per defecte, però en el nostre cas necessitem afegir nous directoris, farem clic sobre el registre **Directorios de archivos de inclusión** i al extrem dret de tot ens permetrà a traves d'un icona afegir nous elements.

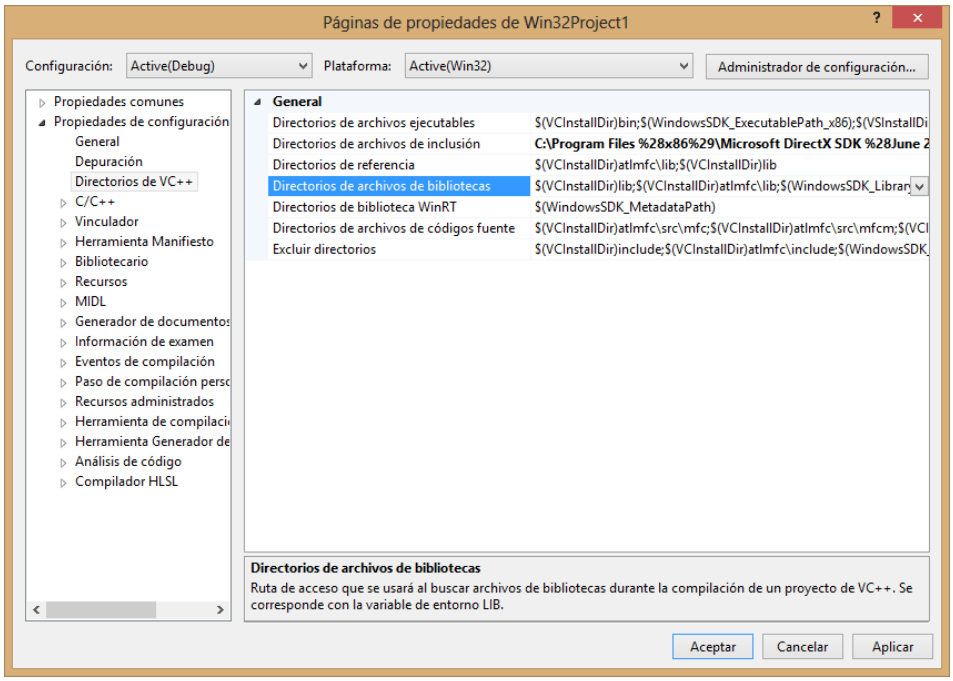


Al entrar en les propietats de directoris d'inclusió tindrem que incloure dos directoris. El primer de tots fa referencia al Microsoft DirectX SDK que està ubicat on hem especificat en la instal·lació d'aquest. Per agregar una nova ruta s'ha de fer clic sobre el símbol d'una carpeta i en aquest cas dirigir-nos C:\[Ubicació de l'instalació]\Microsoft DirectX SDK (June 2010)\Include (Exemple: C:\Program Files (x86)\Microsoft DirectX SDK (June 2010)\Include).

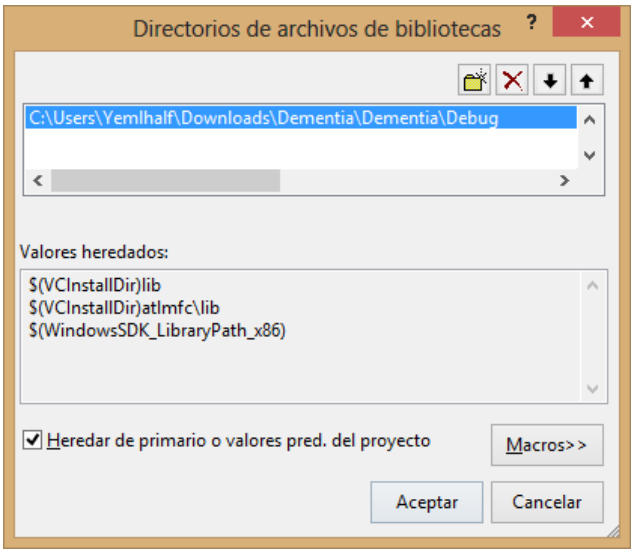
El següent directori farem referencia a la carpeta Include del projecte Dementia proporcionat en el DVD i és \Dementia\Dementia\Include. L'ordre d'inclusió és important, primer la Microsoft DirectX SDK i després la Dementia.



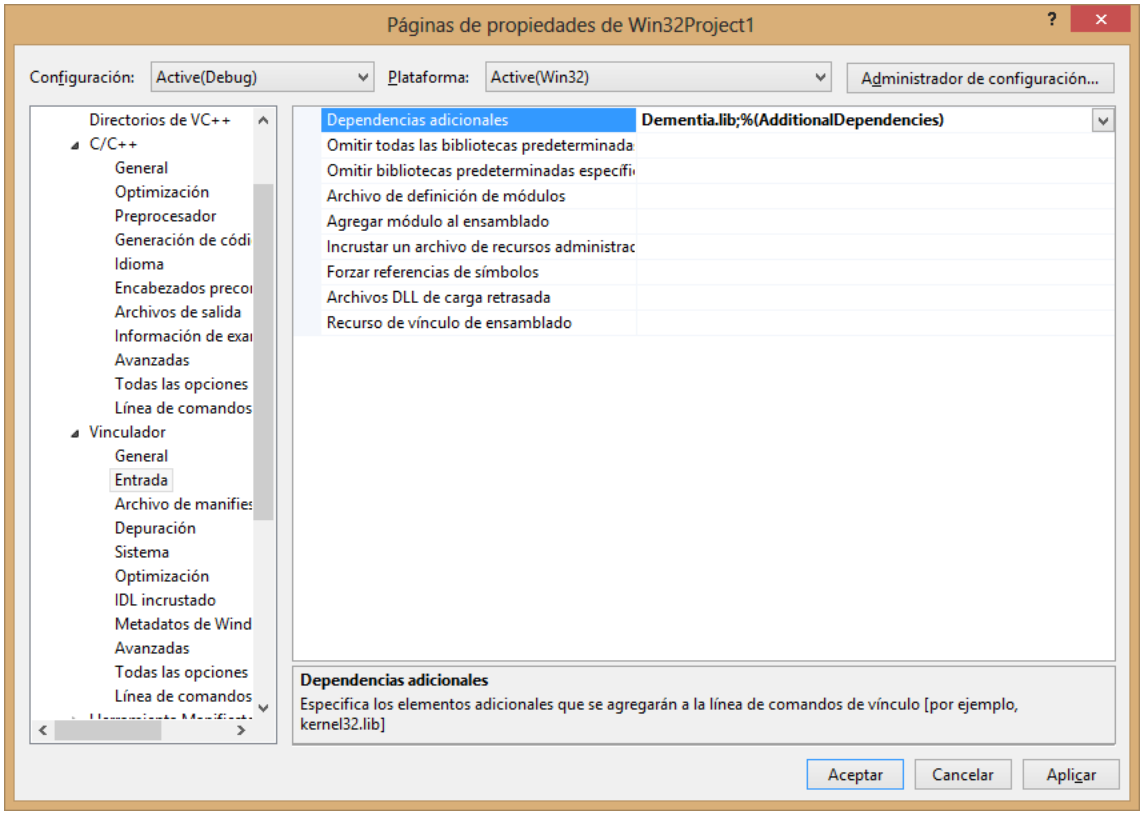
Finalment afegirem la biblioteca de Dementia al projecte, per això tornarem al menú de **Directorios de VC++** i en el registre de **Directorios de archivos de bibliotecas** afegirem un nou paràmetre com hem fet anteriorment.



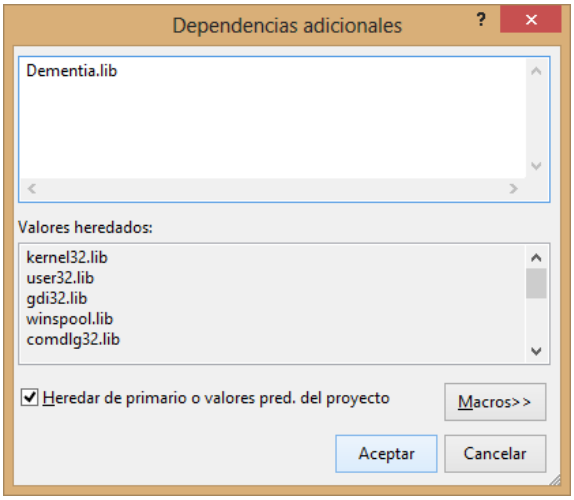
Les biblioteques de Dementia es troben en el projecte en la carpeta Dementia\Debug.



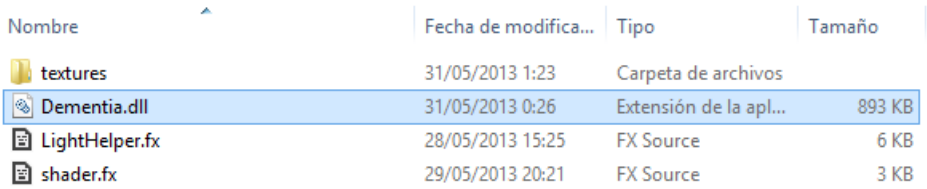
El següent pas en la configuració del projecte continuarem desplegant el apartat de **Vinculador** que és troba en el panell esquerra i entrarem a l'apartat de **Entrada** i en el panell lateral dreta seleccionarem el registre **Dependencias Adicionales** per afegir una nova dependència.



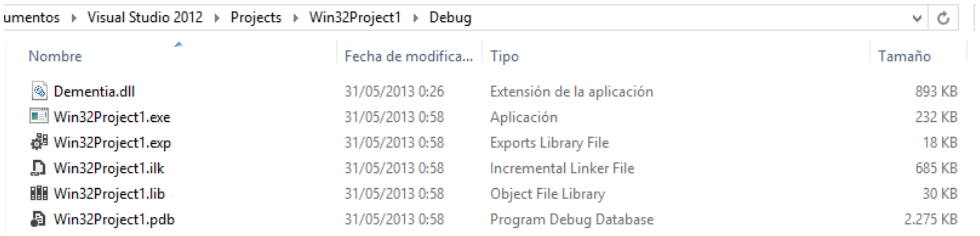
La dependència que tenim que afegir és la llibreria anomenada Dementia.lib (la tenim que escriure a mà).



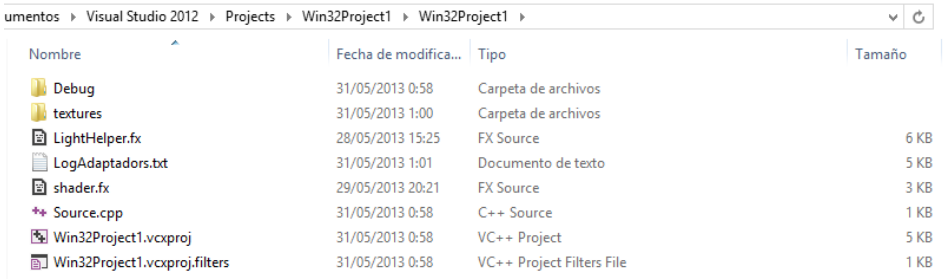
Un cop realitzat tots aquests passos és important prémer el botó de Aplicar sinó els canvis no es guardaran. Per finalitzar el procés de configuració s'han de copiar els següents arxius. Primer de tot la llibreria anomenada Dementia.dll que ubicada en el carpeta de Common del projecte.



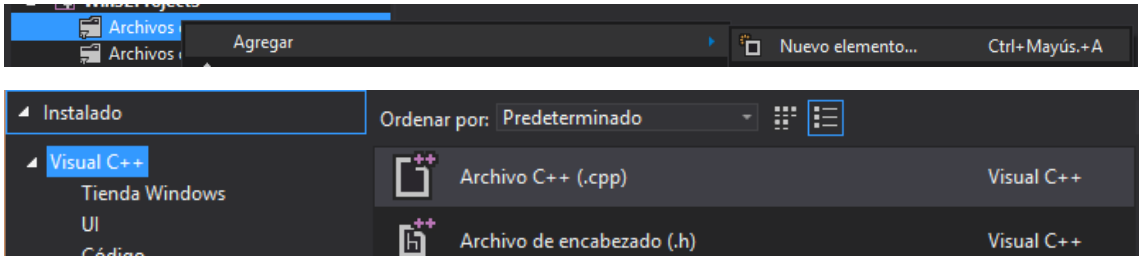
La ruta on s'ha de copiar la llibreria és en el projecte que hem creat en la carpeta Debug (i.e C:\Users\[Nom Usuari]\Documents\Visual Studio 2012\Projects\[Nom Projecte]\Debug. Sinó existeix la carpeta l'hem de crear.



El següents arxius són les textures i fitxers shaders que ha de tindre Dementia per defecte per poder visualitzar el contingut. Aquests components es troben també en la carpeta Common del projecte i els copiarem dins de la carpeta del projecte Win32Project1/Win32Project1 (i.e C:\Users\[Nom usuari]\Documents\Visual Studio 2012\Projects\[Nom Projecte]\[Nom Projecte]).



Ara ja podem crear un nou fitxer en el projecte a traves del Microsoft Visual Studio 2012, per crear-lo és tan fàcil com fer clic sobre la carpeta **Archivos de codigo fuente** i **Agregar > Nuevo Elemento** de tots els elements serà un **Archivo C++ (.cpp)**. Possem el nom i fem clic al botó **Agregar.**



Ara ja podem afegir el següent codi i tindria que executar-se sense cap incident sí hem realitzat tots els passos de forma correcta:

```
#include "Dementia.h"
#include "Window.h"
#include <iostream>

int main()
{
    Dementia::Window window(1,800,600, L"UserDemo1", false, true);
    window.init();

    while (window.run(NULL))
    {
    }

    return 0;
}
```

Nota: Els avisos de compilació que donà són normals ja que sobreescrivim noms de la API gràfica.

Tutorial 1: Crear la primera finestra

```
#include "Dementia.h"   //Incloem la capçalera base del framework.
//Necessari per poder utilitzar el framework.
#include "Window.h"     //Incloem el capçalera per poder utilitzar una
//finestra.

#include <iostream>     //Incloem el capçalera de entrada sortida
//estàndard, per poder imprimir per consola.

int main(){

    /* Creem un objecte Window anomenat window en el que li passem la
    ID de la finestra (1), l'ample (800), la alçada(600), el títol de
    la finestra, desactivem el vsync(false) i la posem en mode
    finestra(true) si fos false, seria a pantalla completa.*/
    Dementia::Window window(1, 800, 600, L"UserDemo1", false, true);

      /* Inicialitzem la finestra (internament crea tots els objectes
      necessaris).*/
    window.init();

    /* Indiquem que mentres la finestra s'estigui executant faci el
    codi que fiquem aquí dins. El mètode run, cada cop que s'executa
    renderitza un frame (si activem el vsync això no canvia, ignora
    les crides fins que toca renderitza un frame).*/
    while( window.run(NULL) ){

        // Codi de la aplicació en execució.

    }

    return 0;   //Fi del programa
}
```

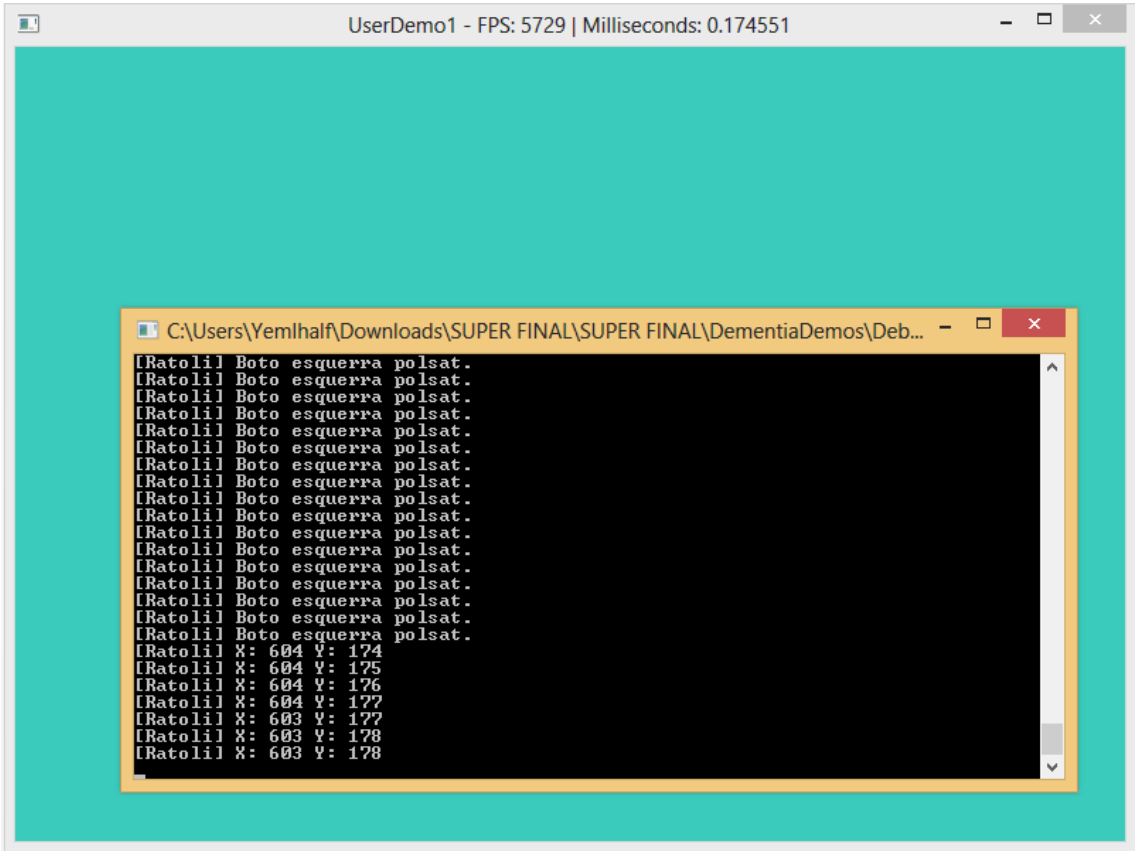
Aquest és el resultat d'executar el codi anterior:

Tutorial 2: Gestionar entrada i sortida

```
#include "Dementia.h"   //Incloem el capçalera base del framework.
//Necessari per poder utilitzar el framework.
#include "Window.h"     //Incloem el capçalera per poder utilitzar
//una finestra.

#include <iostream>     //Incloem el capçalera de entrada sortida
//estàndard, per poder imprimir per consola.

int main(){

    /* Creem un objecte Window anomenat window en el que li passem la
    ID de la finestra (1), l'ample (800), la alçada(600), el títol de
    la finestra, desactivem el vsync(false) i la posem en mode
    finestra(true) si fos false, seria a pantalla completa.*/
    Dementia::Window window(1, 800, 600, L"UserDemo2", false, true);

    /* Obtenim del input manager d'aquesta finestra el ratolí que
    afecta a aquesta finestra. */
    Dementia::Mouse* mouse = window.getInputManager()->getMouse();

    /* Obtenim del input manager d'aquesta finestra el teclat que
    afecta aquesta finestra. */
    Dementia::Keyboard* keyboard = window.getInputManager()-
    >getKeyboard();

    /* Inicialitzem la finestra (internament crea tots els objectes
    necessaris). */
    window.init();

    /* Indiquem que mentres la finestra s'estigui executant faci el
    codi que fiquem aquí dins. El mètode run, cada cop que s'executa
    renderitza un frame (si activem el vsync això no canvia, ignora
    les crides fins que toca renderitza un frame).*/
    while( window.run(NULL) ){

        /* Comprobem si des de la ultima vegada que vam fer un
        getPosition ha canviat de posició el ratolí*/
        if(mouse->isMoved() )
        {
            /* Obtenim la posició en la que es troba el ratolí i la
            guardem en una variable especial per guardar posicions.*/
            XMFLOAT2 posicio = mouse->getPosition();

            // Mostrem per pantalla la posició on es troba el ratolí.
            std::cout << "[Ratoli] X: " << posicio.x << " Y: " <<
            posicio.y << std::endl;
        }

        // Preguntem al ratolí si esta polsat el boto esquerra.
        if(mouse->isButtonPressed(Dementia::Mouse::Left) )
        {
            /* Mostrem per pantalla que el boto esquerra del ratolí
            esta polsat.*/
            std::cout << "[Ratoli] Boto esquerra polsat." <<
            std::endl;
        }
```

```
        /* Com es asíncron podem preguntar també si esta polsat el
        boto dret a la vegada*/
        if( mouse->isButtonPressed(Dementia::Mouse::Right) )
        {
            /* Mostrem per pantalla que el boto dret del ratolí esta
            polsat.*/
            std::cout << "[Ratoli] Boto dret polsat." << std::endl;
        }

        // Preguntem al teclat si la tecla "A" esta polsada
        if(keyboard->isKeyPressed('A') )
        {
            // Mostrem per pantalla que la tecla "A" esta polsada.
            std::cout << "[Teclat] Tecla polsada: A" << std::endl;
        }

        // Preguntem al teclat si la tecla "D" esta polsada
        if(keyboard->isKeyPressed('D') )
        {
            // Mostrem per pantalla que la tecla "D" esta polsada.
            std::cout << "[Teclat] Tecla polsada: D" << std::endl;
        }

        // Preguntem al teclat si la tecla "S" esta polsada
        if(keyboard->isKeyPressed('S') )
        {
            // Mostrem per pantalla que la tecla "S" esta polsada.
            std::cout << "[Teclat] Tecla polsada: S" << std::endl;
        }

        // Preguntem al teclat si la tecla "W" esta polsada
        if(keyboard->isKeyPressed('W') )
        {
            // Mostrem per pantalla que la tecla "W" esta polsada.
            std::cout << "[Teclat] Tecla polsada: W" << std::endl;
        }

        // Preguntem al teclat si la tecla Enter esta polsada
        if(keyboard->isKeyPressed(VK_RETURN) )
        {
            // Mostrem per pantalla que la tecla Enter esta polsada
            std::cout << "[Teclat] Tecla polsada: ENTER" << std::endl;
        }
    }

    return 0;   //Fi del programa
}
```

Aquest és el resultat d'executar el codi anterior.



Tutorial 3: Crear una figura geomètrica

```
#include "Dementia.h"   //Incloem el capçalera base del framework.
//Necessari per poder utilitzar el framework.
#include "Window.h"     //Incloem el capçalera per poder utilitzar
//una finestra.
#include "GeometryFactory.h"

#include <iostream>     //Incloem el capçalera de entrada sortida
//estàndard, per poder imprimir per consola.

int main(){
    Dementia::Window window(1, 800, 600, L"UserDemo1", false, true);

    /* Fem la inicialitzacio de la finestra abans de res, per tenir
    tot preparat. */
    window.init();

    Dementia::Mouse* mouse = window.getInputManager()->getMouse();
    Dementia::Keyboard* keyboard = window.getInputManager()-
    >getKeyboard();

    /* Creem una fabrica de geometries, com a paràmetre li passem la
    finestra per a que sàpiga per quina finestra son. */
    Dementia::GeometryFactory geometryFactory(&window);

    // Creem una geometria buida box, que es on després la omplirem.
    Dementia::Geometry box;

    /* Li demanem a la fabrica de geometries que ens creí una caixa de
    1.0f de ample, 1.0f de alçada i 1.0f de fons, i ens la guarda a la
    variable box, que es la geometria que hem creat abans.*/
    geometryFactory.CreateBox(5.0f, 5.0f, 5.0f, box);

    /* Li demanem a la finestra que ens doni el seu manegador de
    escena.*/
    Dementia::SceneManager* sceneMgr = window.getSceneManager();

    /* Li demanem al manegador de escena que ens creí un mesh amb la
    geometria creada anteriorment assignada. Això es així perquè com a
    mínim un mesh ha de tenir una geometria, però podria tenir també
    textura i shader. */
    Dementia::Mesh* meshBox = sceneMgr->createMesh(&box);

    /* Obtenim el node arrel que es a partir de on s'ha de
    desenvolupar la escena.*/
    Dementia::Node* rootSceneNode = sceneMgr->getRootSceneNode();

    /* Li indiquem al node arrel que te la entitat meshBox associada a
    ell, d'aquesta manera fem que surti per pantalla. */
    rootSceneNode->setEntity(meshBox);

    while( window.run(NULL) ){

        //User code

    }

    return 0;   //Fi del programa
}
```

Aquest és el resultat d'executar el codi anterior.

Tutorial 4: Aplicar textures

```
/* ------------------------------------------------------------------
------------
 Aquet tutorial explica com aplicar textures
 ------------------------------------------------------------------
--------- */

#include "Dementia.h"   //Incloem el capçalera base del framework.
//Necessari per poder utilitzar el framework.
#include "Window.h"     //Incloem el capçalera per poder utilitzar
//una finestra.
#include "GeometryFactory.h"

#include <iostream>     //Incloem el capçalera de entrada sortida
//estàndard, per poder imprimir per consola.

int main(){
    Dementia::Window window(1, 800, 600, L"UserDemo1", false, true);

    /* Fem la inicialitzacio de la finestra abans de res, per tenir
    tot preparat. */
    window.init();

    Dementia::Mouse* mouse = window.getInputManager()->getMouse();
    Dementia::Keyboard* keyboard = window.getInputManager()-
>getKeyboard();

    /* Creem una fabrica de geometries, com a paràmetre li passem la
    finestra per a que sàpiga per quina finestra son. */
    Dementia::GeometryFactory geometryFactory(&window);

    // Creem una geometria buida box, que es on després la omplirem.
    Dementia::Geometry box;

    /* Li demanem a la fabrica de geometries que ens creí una caixa de
    1.0f de ample, 1.0f de alçada i 1.0f de fons, i ens la guarda a la
    variable box, que es la geometria que hem creat abans. */
    geometryFactory.CreateBox(5.0f, 5.0f, 5.0f, box);

    /* Li demanem a la finestra que ens doni el seu manegador de
    escena. */
    Dementia::SceneManager* sceneMgr = window.getSceneManager();

    // Creem una textura a partir de un fitxer de textura existent.
    Dementia::Texture* wood = sceneMgr-
>createTexture(L"textures/woodDirectX.dds");

    /* Li demanem al manegador de escena que ens creí un mesh amb la
    geometria creada anteriorment assignada. Això es així perquè com a
    mínim un mesh ha de tenir una geometria, però podria tenir també
    textura i shader. */
    Dementia::Mesh* meshBox = sceneMgr->createMesh(&box);

    // Li apliquem la textura al mesh
    meshBox->setTexture(wood);

    /* Obtenim el node arrel que es a partir de on s'ha de
    desenvolupar la escena. */
    Dementia::Node* rootSceneNode = sceneMgr->getRootSceneNode();
```

```
    /* Li indiquem al node arrel que te la entitat meshBox associada a
    ell, d'aquesta manera fem que surti per pantalla.*/
    rootSceneNode->setEntity(meshBox);


    while( window.run(NULL) ){

        //User code

    }

    return 0;    //Fi del programa
}
```

Aquest és el resultat d'executar el codi anterior.

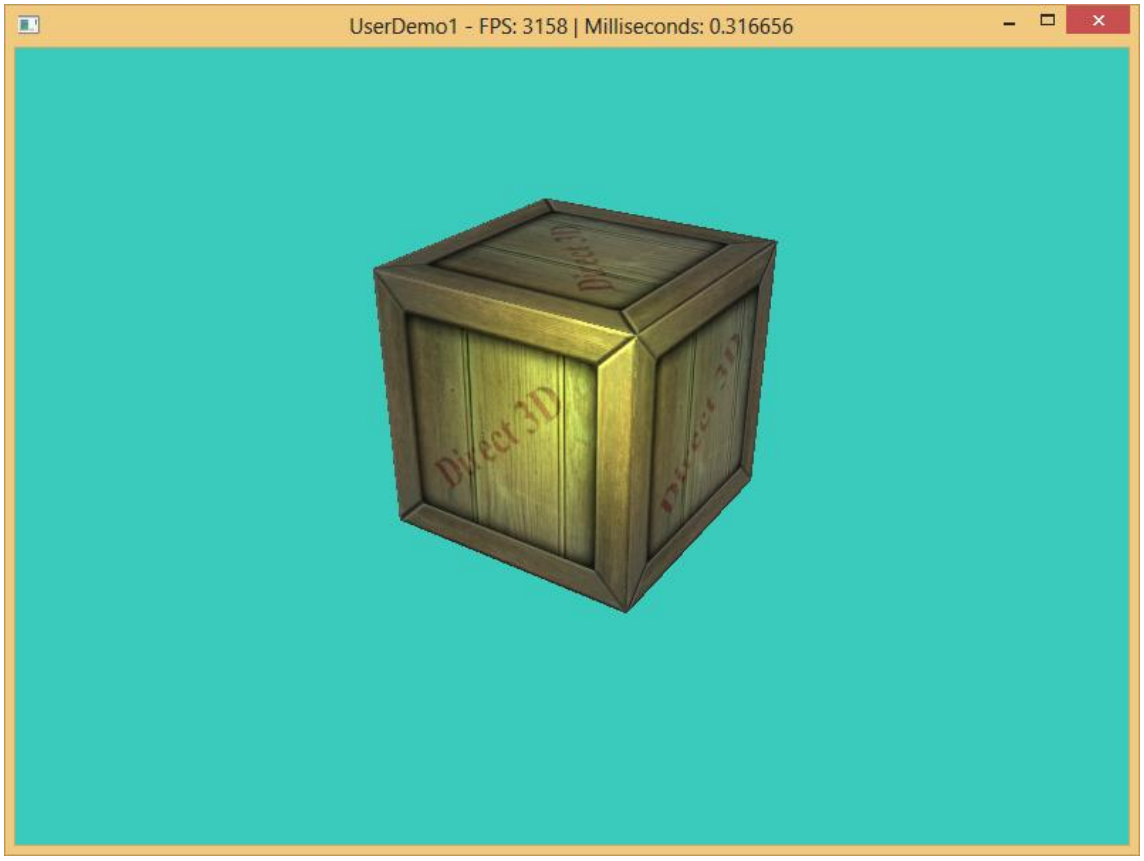
Tutorial 5: Crear una càmera en primera persona

```
/* ------------------------------------------------------------------
------------
 Aquet tutorial explica com utilitzar la classe Camera per crear una
càmera lliure.
------------------------------------------------------------------
--------*/

#include "Dementia.h"   //Incloem el capçalera base del framework.
//Necessari per poder utilitzar el framework.
#include "Window.h"     //Incloem el capçalera per poder utilitzar
//una finestra.
#include "GeometryFactory.h"

#include <iostream>     //Incloem el capçalera de entrada sortida
//estàndard, per poder imprimir per consola.

int main(){
    Dementia::Window window(1, 800, 600, L"UserDemo1", false, true);

    /* Fem la inicialitzacio de la finestra abans de res, per tenir
    tot preparat. */
    window.init();

    Dementia::Mouse* mouse = window.getInputManager()->getMouse();
    Dementia::Keyboard* keyboard = window.getInputManager()-
>getKeyboard();

    /* Obtenim el game timer per després obtenir el delta entre els
    frames. */
    Dementia::GameTimer* gameTimer = window.getGameTimer();

    // Obtenim la càmera associada a la finestra.
    Dementia::Camera* camera = window.getCamera();

    /* Creem una fabrica de geometries, com a paràmetre li passem la
    finestra per a que sàpiga per quina finestra son. */
    Dementia::GeometryFactory geometryFactory(&window);

    // Creem una geometria buida box, que es on després la omplirem.
    Dementia::Geometry box;

    /* Li demanem a la fabrica de geometries que ens creí una caixa de
    1.0f de ample, 1.0f de alçada i 1.0f de fons, i ens la guarda a la
    variable box, que es la geometria que hem creat abans. */
    geometryFactory.CreateBox(5.0f, 5.0f, 5.0f, box);

    /* Li demanem a la finestra que ens doni el seu manegador de
    escena. */
    Dementia::SceneManager* sceneMgr = window.getSceneManager();

    // Creem una textura a partir de un fitxer de textura existent.
    Dementia::Texture* wood = sceneMgr-
>createTexture(L"textures/woodDirectX.dds");
```

```
    /* Li demanem al manegador de escena que ens creí un mesh amb la
    geometria creada anteriorment assignada. Això es així perquè com a
    mínim un mesh ha de tenir una geometria, però podria tenir també
    textura i shader. */
    Dementia::Mesh* meshBox = sceneMgr->createMesh(&box);

    // Li apliquem la textura al mesh
    meshBox->setTexture(wood);

    /* Obtenim el node arrel que es a partir de on s'ha de
    desenvolupar la escena. */
    Dementia::Node* rootSceneNode = sceneMgr->getRootSceneNode();

    /* Li indiquem al node arrel que te la entitat meshBox associada a
    ell, d'aquesta manera fem que surti per pantalla. */
    rootSceneNode->setEntity(meshBox);

    /* Creem una variable de posició per emmagatzemar la posició
    actual del ratolí */
    XMFLOAT2 pos;

    /* Creem una variable que guarda la posició del ratolí en la
    imatge anterior. */
    XMFLOAT2 prevMousePos;

    /* Creem una variable per emmagatzemar el delta de temps entre la
    imatge anterior i la actual. */
    float dt = 0.0f;

    /* Inicialitzem la posició anterior del ratolí al centre de la
finestra. */
    prevMousePos.x = window.getWidth()/2;
    prevMousePos.y = window.getHeight()/2;

    while( window.run(NULL) ){

        /* Mourem la càmera quan l'usuari premi el boto esquerra del
        ratolí.*/
        if(mouse->isButtonPressed(mouse->Left)){

            /* Si el ratolí ha canviat de posició des de la ultima
            vegada que es va cridar al mètode getPosition. */
            if(mouse->isMoved()){

                // Ens guardem la nova posició per treballar amb ella.
                pos = mouse->getPosition();

                  /* Convertim la diferencia entre la posició anterior
                  i la actual en un angle que després convertim a
                  radians per dir-li a la càmera quant a de rotar.*/

                // Angle del pla X
                float dx = XMConvertToRadians(0.25f *
static_cast<float>(pos.x - prevMousePos.x) );

                // Angle del pla Y
                float dy = XMConvertToRadians(0.25f *
static_cast<float>(pos.y - prevMousePos.y) );
```

```
                // Rotem el eix Y el angle del pla X.
                camera->RotateY(dx);

                // Rotem el eix X el angle del pla Y.
                camera->Pitch(dy);

                /* Igualem la posició anterior a la posició actual del
                ratolí.*/
                prevMousePos = pos;
            }
        }

        /* Obtenim el delta de temps entre el frame anterior i el
        actual. */
        dt = gameTimer->getDeltaTime();

        // Comprobem si la tecla W esta polsada
        if(keyboard->isKeyPressed('W') ){
            /* Li diem a la càmera que simuli que el usuari a caminat
            cap endavant 10 unitats, però les multipliquem per el delta
            perquè així aconseguim que avanci el mateix independent-
            ment del processador si s'executa mes vegades per segon o
            menys. */
            camera->Walk(10.0f*dt);
        }

        // Comprobem si la tecla A esta polsada
        if(keyboard->isKeyPressed('A') ){
            /* Li diem a la càmera que simuli que es desplaça de costat
            10 unitats multiplicades per el delta.*/
            camera->Strafe(-10.0f*dt);
        }
        if(keyboard->isKeyPressed('S') ){
            camera->Walk(-10.0f*dt);
        }
        if(keyboard->isKeyPressed('D') ){
            camera->Strafe(10.0f*dt);
        }
        camera->UpdateViewMatrix();
    }

    return 0;   //Fi del programa
}
```

Aquest és el resultat d'executar el codi anterior. Per poder navegar per el escenari mantenim polsat el boto esquerra del ratolí i amb el moviment del ratolí podem moure la camara. Per desplaçar-nos per el escenari utilitzarem las tecles W per moure cap endavant, A moviment esquerra, D per moviment cap a la Dreta i finalment S enredera.



Tutorial 6: Utilitzar el sistema de nodes

```
/*-----------------------------------------------------------------------
-----------
Aquet tutorial explica com utilitzar la classe Camera per crear una
càmera
lliure.
-----------------------------------------------------------------------
--------*/

#include "Dementia.h"   //Incloem el capçalera base del framework.
//Necessari per poder utilitzar el framework.
#include "Window.h"     //Incloem el capçalera per poder utilitzar
//una finestra.
#include "GeometryFactory.h"

#include <iostream>     //Incloem el capçalera de entrada sortida
//estàndard, per poder imprimir per consola.

int main(){
    Dementia::Window window(1, 800, 600, L"UserDemo1", false, true);

    /* Fem la inicialitzacio de la finestra abans de res, per tenir
    tot preparat. */
    window.init();

    Dementia::Mouse* mouse = window.getInputManager()->getMouse();
    Dementia::Keyboard* keyboard = window.getInputManager()-
>getKeyboard();

    /* Obtenim el game timer per després obtenir el delta entre els
    frames.*/
    Dementia::GameTimer* gameTimer = window.getGameTimer();

    // Obtenim la càmera associada a la finestra.
    Dementia::Camera* camera = window.getCamera();

    /* Creem una fabrica de geometries, com a paràmetre li passem la
    finestra per a que sàpiga per quina finestra son.*/
    Dementia::GeometryFactory geometryFactory(&window);

    // Creem una geometria buida box, que es on després la omplirem.
    Dementia::Geometry box;

    // Creem una altra geometria.
    Dementia::Geometry sphere;

    // Creem una altra geometria,
    Dementia::Geometry cylinder;

    /* Li demanem a la fabrica de geometries que ens creí una caixa de
    1.0f de ample, 1.0f de alçada i 1.0f de fons, i ens la guarda a la
    variable box, que es la geometria que hem creat abans. */
    geometryFactory.CreateBox(1.0f, 1.0f, 1.0f, box);

    /* Li demanem a la fabrica de geometries que ens creí un cilindre
    de 3.0f de radi de base, 1.0f de radi de tap, 5.0f de llargada, de
    20 divisions, tant horitzontals com verticals, ens la guarda a la
    geometria cylinder que hem creat abans.*/
```

```
    geometryFactory.CreateCylinder(1.5f, 1.0f, 5.0f, 20, 20,
cylinder);

    /* Li demanem a la fabrica de geometries que ens creí una esfera
    de 5.0f de radi i de 20 divisions tant horitzontals com verticals.
    Ens la guarda a la geometria sphere que hem creat abans. */
    geometryFactory.CreateSphere(1.0f, 20, 20, sphere);

    /* Li demanem a la finestra que ens doni el seu manegador de
    escena. */
    Dementia::SceneManager* sceneMgr = window.getSceneManager();

    // Canvien el color de fons.
    window.setBackgroundColor(XMVectorSet(0.0f, 0.00784f, 0.2f,
1.0f));

    // Creem una textura a partir de un fitxer de textura existent.
    Dementia::Texture* wood = sceneMgr-
>createTexture(L"textures/woodDirectX.dds");

    /* Li demanem al manegador de escena que ens creí un mesh amb la
    geometria creada anteriorment assignada. Això es així perquè com a
    mínim un mesh ha de tenir una geometria, però podria tenir també
    textura i shader. */
    Dementia::Mesh* meshBox = sceneMgr->createMesh(&box);

    /* Li demanem al manegador de escena que ens creí un mesh amb la
    geometria cylinder assignada. */
    Dementia::Mesh* meshCylinder = sceneMgr->createMesh(&cylinder);

    /* Li demanem al manegador de escena que ens creí un mesh amb la
    geometria sphere assignada. */
    Dementia::Mesh* meshSphere = sceneMgr->createMesh(&sphere);

    // Li apliquem la textura al mesh
    meshBox->setTexture(wood);

    /* Obtenim el node arrel que es a partir de on s'ha de
    desenvolupar la escena. */
    Dementia::Node* rootSceneNode = sceneMgr->getRootSceneNode();

    // Creem un nou node per al cilindre.
    Dementia::Node* nodeCylinder = sceneMgr->createNode(
        XMVectorSet(3.0f, 0.0f, 0.0f, 0.0f),    // Indiquem la posició
on es troba el node (X,Y,Z,W)
        XMVectorSet(1.0f, 1.0f, 1.0f, 1.0f),    // Indiquem la escala
del node, per no alterar la escala ha de ser 1 (X,Y,Z,W)
        XMVectorSet(0.0f, 0.0f, 0.0f, 0.0f)     // Indiquem la rotació
del node en angle de 360 (X,Y,Z,W)
    );

    // Creem un nou node per a la esfera.
    Dementia::Node* nodeSphere = sceneMgr->createNode(
        XMVectorSet(-3.0f, 0.0f, 0.0f, 0.0f),   // Indiquem la posició
on es troba el node (X,Y,Z,W)
        XMVectorSet(1.0f, 1.0f, 1.0f, 1.0f),    // Indiquem la escala
del node, per no alterar la escala ha de ser 1 (X,Y,Z,W)
        XMVectorSet(0.0f, 0.0f, 0.0f, 0.0f)     // Indiquem la rotació
del node en angle de 360 (X,Y,Z,W)
    );
```

```
    /* Li indiquem al node arrel que te la entitat meshBox associada a
    ell, d'aquesta manera fem que surti per pantalla. */
    rootSceneNode->setEntity(meshBox);

    /* Li indiquem al node cylinder que te la entitat meshCylinder
    associada a ell. */
    nodeCylinder->setEntity(meshCylinder);

    /* Li indiquem al node sphere que te la entitat sphere associada a
    ell. */
    nodeSphere->setEntity(meshSphere);

    rootSceneNode->addChild(nodeCylinder);
    rootSceneNode->addChild(nodeSphere);

    /* Creem una variable de posició per emmagatzemar la posició
    actual del ratolí */
    XMFLOAT2 pos;

    /* Creem una variable que guarda la posició del ratolí en la
    imatge anterior.*/
    XMFLOAT2 prevMousePos;

    /* Creem una variable per emmagatzemar el delta de temps entre la
    imatge anterior i la actual. */
    float dt = 0.0f;

      /* Inicialitzem la posició anterior del ratolí al centre de la
      finestra.*/
    prevMousePos.x = window.getWidth()/2;
    prevMousePos.y = window.getHeight()/2;

    /* La variable on guardarem la rotació dels objectes que farem a
    la escena */
    float rotation = 0.0f;

    while( window.run(NULL) ){

        // Afegim una mica de rotació.
        rotation = rotation+(1.0f*dt);

        // Canviem la rotació amb els valors nous.
        nodeCylinder->setLocalRotation(XMVectorSet(rotation, 2.0f,
rotation, 0.0f));
        // Canviem la posició amb els valors nous.
        nodeCylinder->setLocalPosition(XMVectorSet(rotation, 1.0f,
rotation, 0.0f));

        if(rotation > 8){ rotation = 0; }

        /* Mourem la càmera quan l'usuari premi el boto esquerra del
        ratolí. */
        if(mouse->isButtonPressed(mouse->Left)){

            /* Si el ratolí ha canviat de posició des de la ultima
            vegada que es va cridar al mètode getPosition. */
            if(mouse->isMoved()){

                // Ens guardem la nova posició per treballar amb ella.
                pos = mouse->getPosition();
```

```
                    /* Convertim la diferencia entre la posició anterior
                    i la actual en un angle que després convertim a
                    radians per dir-li a la càmera quant a de rotar. */

                // Angle del pla X
                float dx = XMConvertToRadians(0.25f *
static_cast<float>(pos.x - prevMousePos.x) );

                // Angle del pla Y
                float dy = XMConvertToRadians(0.25f *
static_cast<float>(pos.y - prevMousePos.y) );

                // Rotem el eix Y el angle del pla X.
                camera->RotateY(dx);

                // Rotem el eix X el angle del pla Y.
                camera->Pitch(dy);

                // Igualem la posició anterior a la posició actual del
ratolí.
                prevMousePos = pos;
            }
        }

        /* Obtenim el delta de temps entre el frame anterior i el
        actual. */
        dt = gameTimer->getDeltaTime();

        // Comprobem si la tecla W esta polsada
        if(keyboard->isKeyPressed('W') ){
            /* Li diem a la càmera que simuli que el usuari a caminat
            cap endavant 10 unitats, però les multipliquem per el delta
            perquè així aconseguim que avanç el mateix independent-ment
            del processador si s'executa mes vegades per segon o menys.
            */
            camera->Walk(10.0f*dt);
        }

        // Comprobem si la tecla A esta polsada
        if(keyboard->isKeyPressed('A') ){
            /* Li diem a la càmera que simuli que es desplaci de costat
            10 unitats multiplicades per el delta. */
            camera->Strafe(-10.0f*dt);
        }
        if(keyboard->isKeyPressed('S') ){
            camera->Walk(-10.0f*dt);
        }
        if(keyboard->isKeyPressed('D') ){
            camera->Strafe(10.0f*dt);
        }
        camera->UpdateViewMatrix();
    }

    return 0;   //Fi del programa
}
```

Aquest és el resultat d'executar el codi anterior.