To3DSun API

Manual d'usuari

Autor : Albert Mas Baixeras
e-mail : albertm@ima.udg.es

Índex

1.- Introducció

Aquest document descriu com emprar la llibreria d'interfície (API) del To3DSun. Aquesta llibreria ha estat desenvolupada per oferir la connexió necessària a qualsevol usuari que la utilitzi, amb les funcionalitats del To3DSun. Per a més informació sobre el To3DSun es recomana consultar el seu manual d'usuari.

Els objectius de la llibreria són oferir accés a :

- Execució del To3DSun.
- Extracció de les dades generades per el To3DSun.
- Exportació de les dades generades juntament amb les modificades per l'usuari als formats VRML i P3D.

La llibreria ha estat desenvolupada amb C++ sobre la plataforma Windows 2000/XP, la mateixa emprada per el desenvolupament del To3DSun. La interfície gràfica emprada ha estat el Qt.

2.- Requisits mínims

No existeixen requisits mínims per la utilització de la llibreria, excepte els imposats per el software necessari per la seva utilització. Aquest software és :

- Visual Studio (VC++) v6.0
  - No és del tot necessari, però aquesta documentació basa la seva utilització, instal·lació i configuració amb aquest software. Queda en mans del usuari la utilització d'altres entorns i eines de compilació.
- Qt v2.3
- OpenGL v1.0
  - Poden emprar-se llibreries compatibles GL, sempre i quan també ho siguin amb Qt, ja que és el Qt qui proporciona directament els recursos GL

Pel que fa a hardware, és recomanable disposar d'una tarja gràfica OpenGL, amb capacitats bàsiques, com *Depth Buffer*, *Alpha Buffer*, i *Stencil Buffer* de 8 bits com a mínim. La resta de requisits hardware existeixen en funció dels imposats per el software abans comentat.

3.- Continguts

La distribució disposa dels següents continguts :

- *include/*
  Conté els fitxers de capçalera (.h) :

- *To3DMain.h*
  És la capçalera principal i cal incloure-la en qualsevol punt on s'accedeix a les funcionalitats bàsiques del *To3DSun*
- *ExportData.h*
  És la capçalera que dóna accés a l'extracció de les dades generades per el *To3DSun*.
- *VRMLInterface.h*
  És la capçalera que dóna accés a l'exportació de fitxers VRML
- *P3DExportWrapper.h, libp3d/, libP3DExportDLL/*
  Aquesta capçalera juntament amb el contingut dels directoris especificats donen accés a l'exportació de fitxers P3D.

- *lib/*
  Conté les llibreries (.lib). Cal emprar ambdós fitxers existents : *To3DSun.lib* i *P3Dexporterdll.lib*

- *bin/*
  Conté els executables i llibreries dinàmiques (.dll).
  Per a utilitzar la llibreria cal emprar les següents llibreries també : *To3DQT.dll*, *P3DExporterDLL.dll , oncrpc.dll*. La resta de llibreries es poden obviar, ja que estan destinades a usos no contemplats en aquesta documentació.
  Els executables inclosos són petites aplicacions d'exemple que utilitzen aquesta llibreria, i que es comentaran més endavant.
  Finalment, cal mencionar que el directori *bin/Icones/* cal que es trobi en el mateix directori d'execució de l'aplicació que utilitzi la llibreria. Altrament, no es visualitzarien correctament les icones de l'aplicació.

- *extras/*
  Conté un fitxer de capçalera especial, el *Interficie.h*, el qual es te que incloure a cada projecte que empri el To3DSun.

4.- Creant un nou projecte

A continuació es descriu com configurar el *Visual Studio* per treballar amb aquesta llibreria. Altres plataformes de desenvolupament poden ser igualment emprades, modificant els paràmetres corresponents, els quals tan sols fan referència a inclusió de noves rutes i fitxers per la compilació i muntatge. Alhora, l'usuari és lliure de modificar els següents passos segons ell mateix cregui convenient a efectes de millorar eficiències o simplificar el procés. El que es mostra aquí és tan sols una guia de totes les possibles.

Els passos a seguir són els següents :

- Crear dos noves variables d'entorn :

`$(QTDIR)` -> Ha d'especificar la ruta actual del sistema on està instal·lat el Qt.

`$(TO3DSUN)`-> Ha d'especificar la ruta actual del sistema on està instal·lat el To3DSun.

- Crear un nou projecte. El tipus de projecte sol ser irrellevant, tot i que és útil crear una aplicació simple i desvinculada del sistema MFC de Windows
- Modificar les opcions del projecte accedint al menú *Project/Settings*.
    - Especificar el directori de treball. Cal que l'executable creat pugui trobar les DLL necessàries, així com el directori *Icones/*.

    Debug -> Working Directory -> `$(TO3DSUN)/bin`

    - Especificar les directives del preprocessador.

    C/C++ -> Preprocessor definitions -> `QT_DLL,UNICODE, QT_THREAD_SUPPORT`

    * Aquestes directives són necessàries per la utilització del Qt

    - Especificar els directoris alternatius de capçaleres

    C/C++ ->Additional Include Directories ->

    ```
    $(QTDIR)\include,
    $(TO3DSUN)\include,
    $(TO3DSUN)\include\libP3DexportDLL\include\,
    $(TO3DSUN)\include\libp3d\
    ```

    - Especificar les llibreries emprades

    Link -> Object/Library Modules ->

    ```
    $(QTDIR)\lib\qt-mt230nc.lib
    $(QTDIR)\lib\qtmain.lib
    $(QTDIR)\lib\qt-mt230nc.lib
    $(QTDIR)\lib\qtmain.lib
    $(TO3DSUN)\lib\To3DQT.lib
    $(TO3DSUN)\lib\P3DExporterDLL.lib
    ```

    Link -> Ignore libraries -> `libc`

    - Opcionalment, es pot especificar que es copií el executable resultant al directori *bin/* de la llibreria. En aquest cas, cal canviar també el camí de l'executable

    Post-Build Step -> `copy Debug\test.exe $(TO3DSUN)\bin\test.exe`
    Debug -> Executable for Debug session -> `$(TO3DSUN)\bin\test.exe`

- Especificar al sistema les dues noves variables de ruta, `$(QTDIR)` i `$(TO3DSUN)`. Cal emprar les eines que disposa el sistema operatiu per

modificar aquestes variables d'entorn. És possible que requereixi reiniciar el *Visual Studio* per que tinguin efecte.

- Afegir al projecte el fitxer `Extras/Interficie.h`

- Modificar les propietats de compilació del fitxer

  Settings -> Outputs `-> $(InputDir)\moc_$(InputName).cpp`
  Settings -> Commands ->
  `%qtdir%\bin\moc.exe "$(InputDir)\$(InputName).h" -o`

- Compilar tot el projecte, obviant els errors que apareixen.

- Afegir al projecte el nou fitxer creat : `moc_Interficie.cpp`

Qualsevol d'aquestes configuracions no ha d'implicar la supressió d'altres ja existents per defecte.

Algunes de les configuracions fan referència tan sols a les compilacions en mode *Debug*. Per les compilacions en mode *Release* es recomana emprar les mateixes configuracions.

5.- Utilitzar la llibreria

5.1.- Execució del To3DSun

La llibreria que s'ofereix aglutina tota l'aplicació To3DSun, incloent per tant la possibilitat d'executar l'aplicació. De fet, caldrà executar l'aplicació si es volen adquirir les dades 3D generades i/o exportar VRML o P3D, ja que es requereix d'un procés interactiu per part de l'usuari.

El To3DSun s'executa sota la interfície gràfica Qt. Es recomana consultar la referència d'aquesta interfície per obtenir-ne mes detalls. La seva execució depèn d'un marc principal, anomenat *QMainWindow*. Per tant, es pot visualitzar l'aplicació de forma independent o associada a un *frame* d'una altra aplicació. Per executar-la de forma independent tan sols cal associar un objecte de tipus *To3DMain* a un objecte Qt que faci d'aplicació o *frame.* El següent exemple mostra com executar l'aplicació :

```
#include <To3DMain.h>
#include <qapplication.h>
#include <qstatusbar.h>

int main(int argc, char **argv )
{
    QApplication a(argc, argv);

    To3DMain *t3s = new To3DMain(NULL);

    t3s->setCaption("To3D Test");

    t3s->connect(t3s, SIGNAL(message(const QString&, int)), t3s->statusBar(),
SLOT(message(const QString&, int)));

    a.setMainWidget(t3s);

    t3s->show();

    return a.exec();
}
```

En aquest cas es cedeix tot el control de l'execució al To3DSun, i quan aquest finalitza, finalitza també l'aplicació.

Tan sols s'ha d'utilitzar la capçalera *To3DMain.h* que ofereix la llibreria.

Aquest exemple es pot trobar en el exemple *To3DTest* de la distribució.

5.2.- Extracció de les dades 3D generades

Un cop s'ha generat el model 3D pot interessar extreure les dades obtingudes per analitzar-les i treballar-hi més còmodament des de l'aplicació del usuari.

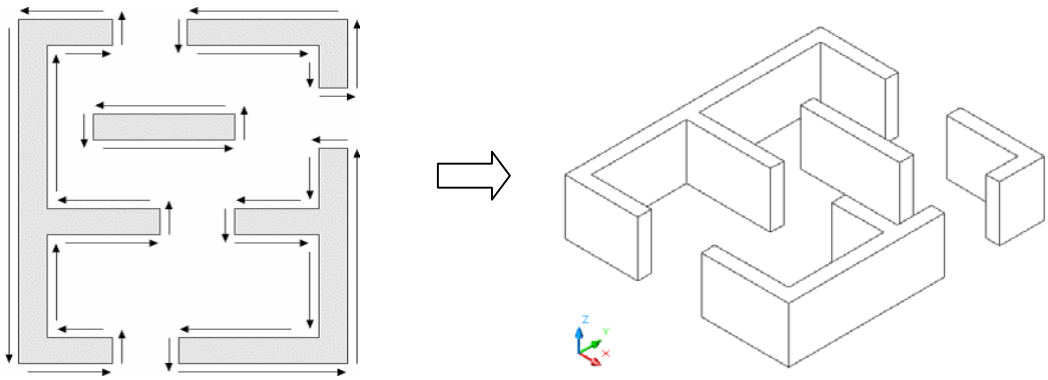
5.2.1.- Estructura de dades

Existeix una estructura que permet recollir aquestes dades en un format entenedor i escalable. En l'annex inclòs en la distribució es pot observar la distribució de classes i dades membre d'aquesta estructura (veure documentació HTML).

A continuació es comentarà el contingut de les dades de cada estructura :

- Parets

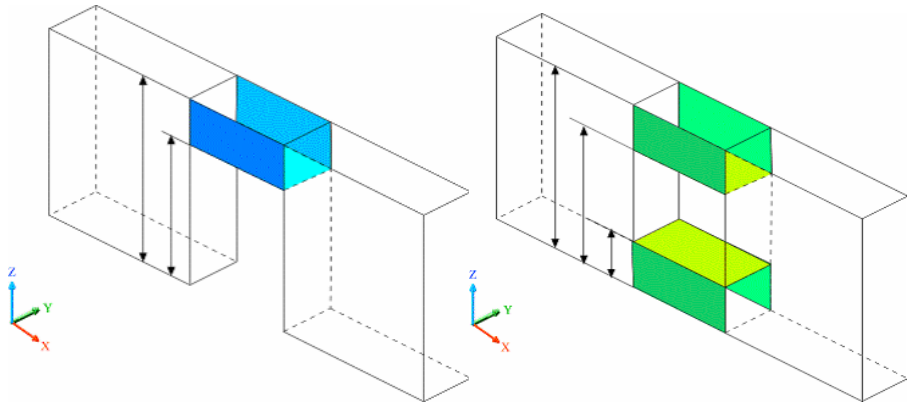
  Les parets es defineixen com un conjunt de polilínies tancades, on cadascuna forma un polígon irregular còncau i orientat positivament. Cadascuna de les línies de cada polilínia és la base per crear un pla perpendicular amb una certa alçada, formant un costat de la paret. Gràcies a que els polígons estan orientats positivament es pot especificar fàcilment l'orientació correcta d'aquests plans.



Les parets es representen amb polígons irregulars còncaus orientats positivament i extruits

- Portes i Finestres

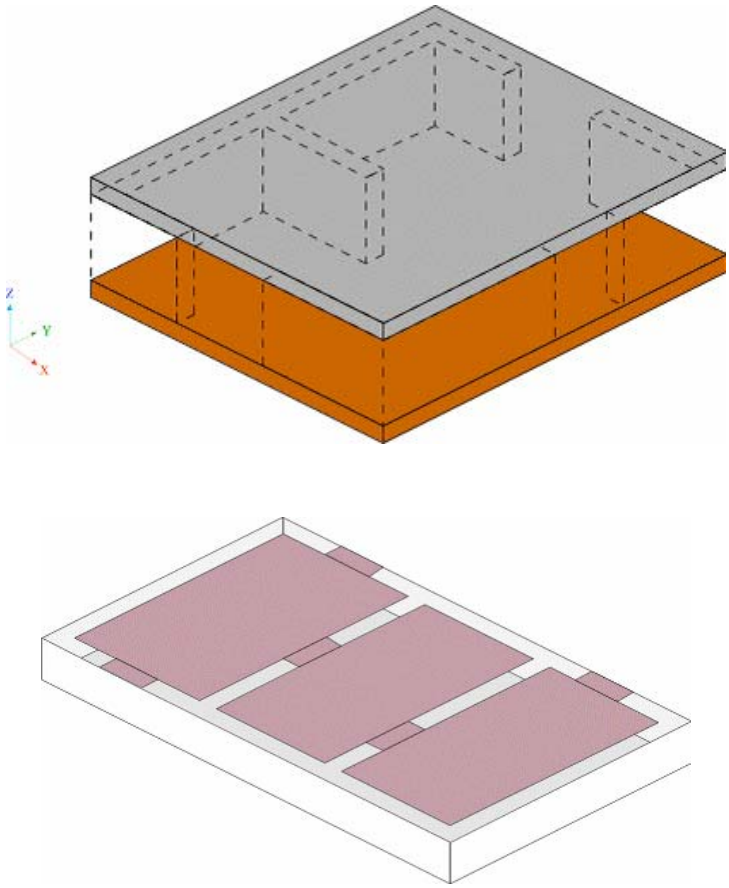
  Les portes i finestres es defineixen amb quatre punts, que simbolitzen la seva base. Llavors, amb dos alçades i l'alçada de les parets es poden construir els plans a partir de la combinació dels quatre vèrtexs de la base. L'orientació i ordre d'aquests vèrtexs és la següent :



A l'esquerra la configuració d'una porta, i a la dreta la d'una finestra

- Terra i Sostre

  El terra i sostre es defineixen com un conjunt de polígons que donen forma a la seva base, generant un petit prisma paral·lel a les parets. En el cas més senzill, tan sols hi haurà un polígon que definirà el envolupant de la planta. Altrament, hi haurà un conjunt de polígons que definiran l'interior de les estances de la planta. En aquest cas, es te que generar igualment un prisma amb el supòsit que tenim un sol polígon envolupant.



A dalt el cas bàsic amb polígons simples. A sota, el cas complex, amb polígons per a cada estança

- Elements estructurals

  Un element estructural no és res mes que un polígon que s'extrueix en la direcció del sostre i que disposa de dos alçades : una des del terra, i l'altre com a longitud d'extrusió.

Un cas particular contemplat és el dels blocs 2D. Els blocs 2D són aquells símbols especificats com a tal i llegits del fitxer DXF d'entrada. S'ha considerat que podrien ser d'utilitat i s'han inclòs en aquesta estructura, contenint tan sols la posició, escalat, rotació i nom (etiqueta).

Finalment, existeix una estructura que conté totes les dades de tipus genèric que no s'han pogut incloure en les altres, anomenada *ExportMisc*. Aquesta conté el nombre de plantes i funcions que permeten saber les alçades de cada planta o les alçades globals, tenint en compte també els gruixos dels terres i sostres.

5.2.2.- Polígons 3D

En la secció anterior es detalla com construir cada objecte a partir de dades lògiques, en 2D i mitjançant extrusions. És possible que l'usuari desitgi construir el model 3D sense necessitat de generar un a un tots els polígons 3D. És per això que cada classe de la estructura abans mencionada disposa de dos membres :

- *m_QuadFaces*
  Llista de polígons quadrilàters. Cada polígon disposa de quatre vèrtexs en una seqüència orientada positivament.
- *m_TriFaces*
  Llista de polígons triangulars. Cada polígon disposa de tres vèrtexs en una seqüència orientada positivament.

Aquestes llistes es generen en el procés d'extracció de les dades que es veu a continuació.

5.2.3.- Extracció

Per extreure el codi cal que el To3DSun estigui executant-se, ja que al tancar la seva finestra principal es perden totes les dades. A continuació es mostra un exemple per extreure aquestes dades :

```
ExportData dat;
ExportDataSet set;

dat.GetData(&set, m_To3D);    // m_To3D és un punter al objecte To3DMain
```

Per donar més flexibilitat al usuari, s'ha evitat la possibilitat de modificar les dades internes al To3DSun. O sigui, es poden extreure, però no es poden tornar a incloure. Així s'evita restringir l'estructura de dades a un format molt específic en funció del funcionament intern de la llibreria, alhora que aïlla la llibreria de possibles futurs canvis interns.

5.3.- Exportació a VRML

Utilitzar les funcions del To3DSun per exportar en format VRML pot portar a certa confusió entre les dades pròpies del To3DSun i les possibles dades modificades que es volen exportar. Per barrejar els dos tipus de dades a exportar s'ha dissenyat un sistema que permet a l'usuari definir què vol exportar i com, ja sigui utilitzant les ja existents funcions d'exportació del To3DSun o les seves pròpies.

5.3.1.- Contenidor i interfícies

La classe *VRMLExportWrapper* és una classe contenidor que inclou dos components. Per una banda hi ha l'exportador intern del To3DSun, el qual queda invisible al usuari i es crea i accedeix de forma automàtica. Per altra banda hi ha una interfície, *VRMLInterface*, la qual conté cadascuna de les funcions d'exportació en funció del tipus d'objecte a exportar. Aquesta classe cal que sigui especificada abans de començar l'exportació.

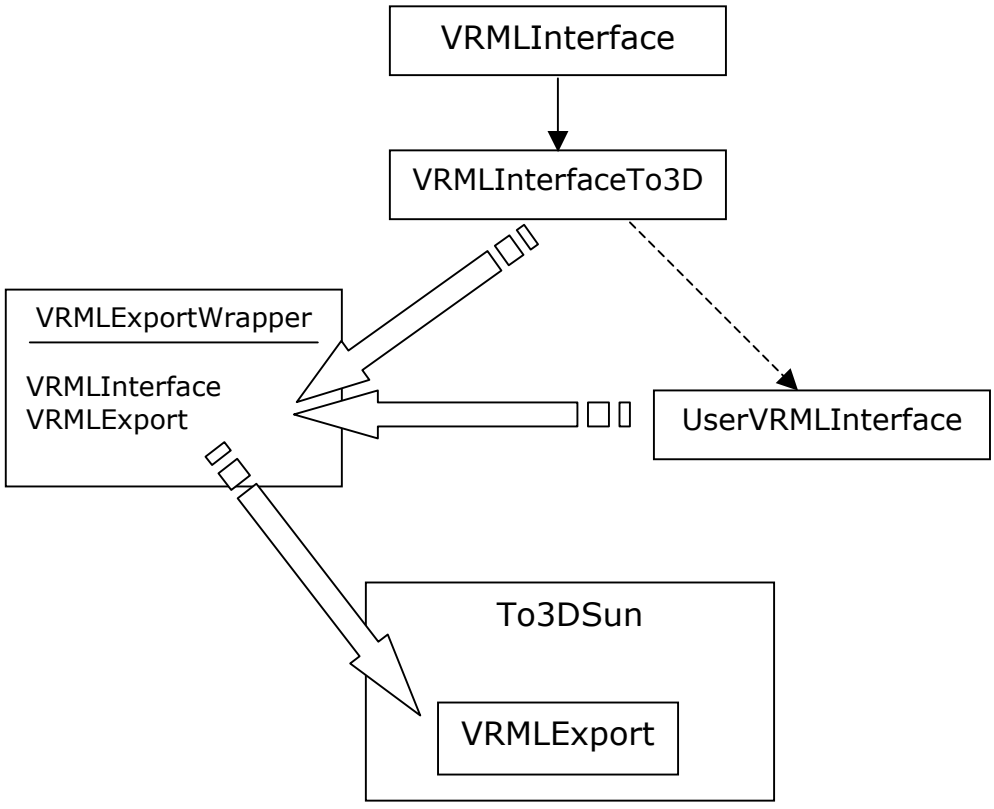
Tal i com es pot observar en la documentació HTML inclosa en la distribució, la classe *VRMLInterface* disposa de funcions virtuals pures. Això vol dir que l'usuari està obligat a derivar una classe d'aquesta que implementi cadascuna de les funcions. Per això existeix la classe *VRMLInterfaceTo3D* derivada. Aquesta implementa totes les funcions, tal i que s'executen les funcions internes d'exportació del To3DSun.

Si l'usuari vol utilitzar totes les funcions d'exportació del To3DSun sense modificar-les, tan sols cal que especifiqui a la classe *VRMLExportWrapper* la interfície *VRMLInterfaceTo3D*. Si l'usuari vol modificar les funcions d'exportació ha de derivar una nova classe de *VRMLInterfaceTo3D* i implementar les funcions desitjades. Així la resta de funcions seguiran executant el codi d'exportació intern del To3DSun.

Un dels motius per que l'usuari ha de voler modificar les funcions d'exportació és la modificació de les dades exportades. Per exemple, si l'usuari vol adquirir les dades de les parets 3D generades, modificar-les i exportar-les, tan sols haurà de crear una interfície que implementi la funció d'exportar les parets.

Una altra possible necessitat és la d'exportació de nous objectes. Per això existeixen les funcions *ExportExtras*, les quals per defecte no exporten res, però que permeten al usuari exportar nous tipus d'objectes.

Finalment, i de forma obvia, les classes interfície necessiten accedir a les dades que es vol exportar. Per això, és necessari especificar-les abans d'assignar-les al contenidor. Cal veure que aquesta necessitat tan sols existeix si l'usuari requereix de les seves pròpies dades. Altrament, el propi sistema accedeix de forma automàtica a les dades internes del To3DSun.



5.3.2.- Estructura del VRML

No és objectiu d'aquest document detallar la sintaxi del format VRML, motiu per el qual s'emplaça al usuari a la consulta de la documentació adequada. És fonamental però especificar l'estructura dels documents de sortida. Aquesta estructura és important ja que permet mantenir coherència en la disposició de les dades en documents extensos.

L'estructura és la següent :

- Capçalera
- Definició de la camera
- Plantes de l'edifici

  Es defineixen etiquetes per a cadascuna amb el següent format : *floor 1, floor 2, ... floor n*.

  Cada planta disposa dels següents ítems i respectives etiquetes :

  - Parets (*Walls*)
  - Portes (*Doors*)
  - Finestres (*Windows*)
  - Sostre (*Ceiling*)
  - Terra (*Ground*)
  - Mobles (*Units*)
  - Lluminàries (*Lights*)
  - Elements estructurals (*Structural*)

    Aquests elements poden ser pilars, envans, etc.

En base a aquesta estructura, les funcions de la interfície demanen com a paràmetre el numero de planta, ja que seran executades tantes vegades com plantes hi hagin. Un cas especial és el de les funcions *ExportExtras*. Hi ha dos versions, una que requereix el numero de planta i l'altre no. La que requereix el numero de planta s'executa al final de cada planta, i per tant qualsevol cosa que s'escrigui en el fitxer de sortida apareixerà darrera l'etiqueta *Structural*. La que no requereix numero de planta s'executa al final de l'escriptura de totes les plantes.

5.3.3.- Exemple

A continuació es mostren exemples d'utilització. Aquests es poden trobar en l'exemple To3DTestGUI que es pot trobar amb la distribució.

El següent codi és l'encarregat d'instanciar l'exportador i assignar-li la interfície i dades a exportar :

```
ExportData dat;
ExportDataSet set;
TestInterface *vint;                // Classe interfície derivada per l'usuari
VRMLExportWrapper *wrap;

if (m_To3D == NULL) return;         // m_To3D és un punter al objecte To3DMain

dat.GetData(&set, m_To3D);          // Extracció de les dades

wrap = new VRMLExportWrapper(m_To3D);

vint = new TestInterface(wrap);
vint->m_Data = &set;                // Assignació de les dades a la interfície

wrap->SetInterface(vint);           // Assignació de la interfície

wrap->Export("TestGUI.wrl\0");      // Exportació
```

La classe d'exemple *TestInterface* implementa les funcions d'exportació de les parets i les dos funcions d'exportacions extres. En l'exportació de les parets s'ha modificat la seva alçada, duplicant-la. En les exportacions extres, s'exporten petits cubs centrats a cada planta, i un cub que engloba totes les plantes.

Les implementacions dels exemples es poden observar en el projecte To3DTestGUI.

És important observar com per escriure en el fitxer de sortida s'utilitza la funció *VRMLExportWrapper::Write(int file, char *, unsigned int size)*. És molt important emprar aquesta funció enlloc d'intentar escriure directament sobre el *file descriptor* del fitxer, ja que el context d'execució és diferent entre la llibreria i l'aplicació del usuari, i per tant, els *file descriptors* seran diferents. La utilització d'aquesta funció evita aquest problema.

Finalment, tan sols resta comentar que cal incloure la capçalera *VRMLInterface.h* per accedir a aquestes classes.

5.4.- Exportació a P3D

De la mateixa forma que amb el format VRML, no és intenció d'aquest document especificar la sintaxi d'aquest format. L'estructura del fitxer de sortida no disposa de seccions com el VRML, però si que requereix d'una mínima agrupació que permeten classificar alguns tipus d'objectes. Abans però de mostrar aquesta estructura, cal emplaçar al usuari a consultar els fitxers, estructures i classes dedicades a l'exportació al format P3D i que s'inclouen amb la llibreria. Això és necessari degut a que les necessitats d'estructuració no són plenament dependents del To3DSun, i es mantenen de forma separada al seu funcionament intern.

5.4.1.- Contenidor i interfícies

La idea és similar a la del contenidor i interfícies per exportar VRML. En aquest cas la classe que exporta s'anomena *P3DExportWrapper*, i la interfície base *P3DInterface*. La implementació que permet executar les funcions internes d'exportació i que cal derivar per canviar-les s'anomena *P3DInterfaceTo3D*. Es pot consultar la informació HTML annexa amb la distribució.

5.4.2.- Estructura del P3D

Tal i com s'ha comentat, no existeix una estructura pròpia en el fitxer, dividida per seccions o zones. L'estructura es realitza a nivell d'objecte i agrupacions d'objectes.
Es disposa d'una llista d'objectes, la qual a posteriori, i de forma totalment interna, és emprada per l'exportació. Quan l'usuari implementa una funció d'exportació, tan sols té que afegir els objectes modificats a la llista sense preocupar-se de com exportar-los.

Cada tipus d'objecte es té que definir de la següent forma :

- Parets
  Cada conjunt de parets consecutius es defineix en un *P3DWall* (consultar els objectes i la seva estructura directament en la llibreria exportadora P3D). Cada *P3DWall* conté una llista de *P3DSurfaces*, que representen cada paret. Cada *P3DSurface* disposa d'un *P3DFace*, que defineix els vèrtexs de les parets.

- Resta d'objectes
  Per a cada objecte es defineix en un *P3DWall*. Cada *P3DWall* conté una llista per a totes les superfícies que formen l'objecte, i es defineix en un *P3DSurface*. Cada *P3DSurface* disposa d'un *P3DFace* que defineix els vèrtexs de cada superfície.

De fet, es pot observar com l'estructura de les parets i la resta d'objectes és molt similar.

5.4.3.- Exemple

A continuació es mostra un exemple per executar la exportació a P3D lleugerament modificada i que es pot trobar en l'exemple To3DTestGUI de la distribució. Les modificacions realitzades són les mateixes que en l'exemple anterior d'exportació a P3D.

A continuació es mostra el codi emprat per instanciar l'exportador, la interfície, extreure les dades i exportar a P3D :

```
ExportData dat;
ExportDataSet set;
TestInterfaceP3D *vint;              // Classe interfície derivada per l'usuari
P3DExportWrapper *wrap;

if (m_To3D == NULL) return;          // m_To3D és un punter al objecte To3DMain

dat.GetData(&set, m_To3D);           // Extracció de les dades

wrap = new P3DExportWrapper(m_To3D);

vint = new TestInterfaceP3D(wrap);
vint->m_Data = &set;                 // Assignació de les dades a la interfície

wrap->SetInterface(vint);            // Assignació de la interfície

wrap->Export("TestGUI.p3d\0");       // Exportació
```

La classe *TestInterfaceP3D* és la interfície derivada de *P3DInterfaceTo3D* que s'ha implementat per l'exemple. Les modificacions en l'exportació són les mateixes que es realitzen en la exportació a VRML.