VISUALIZACIÓN Y DEFORMACIÓN DE OBJETOS VIRTUALES 3D

ANEXOS

Raúl Andrés Gómez Ruiz
Faber Ramiro Montero Calderón

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Abril de 2012

Contenido

LISTA DE FIGURAS

LISTA DE TABLAS

Anexo A. GUÍA DE INSTALACIÓN

A continuación se muestra una guía para instalar las librerías que requiere la aplicación software 3d para su funcionamiento. Estas librerías se instalarán bajo el entorno de programación Visual Studio 2005 y el sistema operativo Windows 7.

A.1. Instalación de Qt

La última versión de esta librería está disponible en la siguiente dirección: qt.nokia.com/downloads, en la sección “Qt library” seleccionar “Qt libraries 4.8.0 for Windows (VS 2008, 273 MB)” o la versión que esté disponible, al descargar se obtiene el siguiente archivo tipo aplicación qt-win-opensource-4.8.0-vs2008.exe y se procede a instalar en el directorio deseado, por ejemplo en:

C:\Qt\4.8.0

También descargar el archivo ActivePerl-5.14.2.1402-MSWin32-x86-295342.exe en la siguiente página www.activestate.com/downloads del recuadro “ActivePerl Community Edition”, luego instalar.

En el símbolo del sistema de Visual Studio (que se accede desde inicio, todos los programas, Microsoft Visual Studio 2005, Visual Studio Tools y finalmente en Símbolo del sistema de Visual Studio) se ingresan los comandos proporcionados a continuación para configurar y compilar Qt.

En la ventana de comandos aparecerá la siguiente dirección.

C:\Program Files\Microsoft Visual Studio 8\VC>

Se debe ubicar en la dirección donde se instalo Qt, para retroceder directorios se usa el comando “cd..” así como se muestra a continuación.

C:\Program Files\Microsoft Visual Studio 8\VC>cd..

Para acceder a directorios se usa el comando “cd” seguido del nombre del directorio, por ejemplo:

C:\>cd Qt

Luego hay que ubicarse en la dirección donde se instalo Qt, por ejemplo:

C:\Qt\4.8.0>

En esa dirección se ingresa el comando “configure” y se pulsa “o” para usar la edición de código abierto, luego el sistema preguntará por la aceptación de los términos de licencia entonces se pulsa “y” para aceptar, la configuración tarda varios minutos.

Posteriormente al término de la configuración se ingresa el comando “nmake”, este proceso de compilación tarda varias horas, alrededor de 5 horas, dependiendo de la maquina que se esté utilizando.

Después de la compilación se debe añadir la variable Qt en las variables del sistema. Para esto se debe abrir la ventana variables de entorno (inicio, panel de control, sistema y seguridad, sistema, configuración avanzada del sistema y finalmente en el botón variables de entorno), luego en el recuadro “variables del sistema” dar clic en el botón “Nueva” y aparecerá una ventana con dos ítems: “Nombre de la variable” y “valor de la variable”.

En “Nombre de la variable” ingresar Qt y en “valor de la variable” ingresar C:\Qt\4.8.0\bin

A.2. Instalación de VTK

La última versión de esta librería está disponible en la siguiente dirección: www.vtk.org/VTK/resources/software, en este caso la versión de VTK es la 5.8, por tanto ir a la sección “Latest Release (5.8.0)” seleccionar vtk-5.8.0.zip o vtk-5.8.0.tar.gz.

Al término de la descarga se descomprime el archivo y se ubica por ejemplo en:

C:\vtk-5.8.0

Para compilar estas librerías con Visual Studio necesitamos de una herramienta que genere un proyecto en esta plataforma, la herramienta que permite este proceso es CMake.

La última versión de CMake está disponible en la siguiente dirección: www.cmake.org/cmake/resources/software, en la sección “Binary distributions” escoger la plataforma “Windows (Win32 Installer)” seleccionando el archivo tipo aplicación para descargar cmake-2.8.7-win32-x86.exe, luego se instala en la ubicación deseada.

Se ejecuta CMake y en la ventana se deben colocar dos rutas de directorios, la primera es donde se encuentra la fuente y la segunda el destino de compilación, por ejemplo:

Where is the source code: C:\vtk-5.8.0
Where to build the binaries: C:\vtk-build

Luego se da clic en configure y aparecerá una ventana preguntando si desea crear el directorio de destino C:\vtk-build, entonces se da clic en yes. Se escoge la opción Visual Studio 8 2005 con la selección “Use default native compilers” y se da clic en Finish. Al término del proceso donde se verán líneas en rojo se debe seleccionar “Advanced” para acceder a más opciones de configuración.

En las siguientes líneas hacer la respectiva configuración.

VTK_USE_GUISUPPORT: ✓
VTK_USE_QT: ✓

Luego dar clic en Configure hasta que todas las líneas aparezcan en gris, posteriormente se da clic en Generate. Al término de este proceso se habrá generado el código en la ubicación C:\vtk-build.

En la ubicación C:\vtk-build abrir el archivo VTK.sln con Visual Studio, en el “explorador de soluciones” dar clic en ALL_BUILD, luego ir al menú Generar y se selecciona “Generar ALL_BUILD”, con esto se inicia un proceso de compilación. Al terminar la compilación se

procese a instalar VTK de la siguiente manera: En el “explorador de soluciones” dar clic en INSTALL, luego ir al menú Generar y se selecciona “Generar INSTALL”. VTK se instalara por defecto en C:\Program Files\VTK, si en la instalación se generan errores, se debe ejecutar Visual Studio como administrador.

A.3. Instalación de la aplicación software 3D

Copiar la carpeta Cxx en un directorio, por ejemplo:

C:\DefOb3D\Cxx

Se abre CMake y se ingresan los siguientes datos:

Where is the source code: C:\DefOb3D\Cxx
Where to build the binaries: C:\DefOb3D\build

Dar clic en Configure, escoger la opción Visual Studio 8 2005, dar clic nuevamente en Configure hasta que todas las líneas aparezcan en gris y finalmente dar clic en Generate.

Ahora se necesita de las librerías de detección de colisiones VCollide y Rapid, se descomprimen los archivos en una ubicación deseada, por ejemplo:

C:\vcollide-2.01
C:\rapid-2.01

En la ubicación C:\DefOb3D\build abrir el archivo DefOb3D.sln con Visual Studio, en el “explorador de soluciones” dar clic en DefOb3D, luego ir al menú Proyecto, seleccionar propiedades, desplegar “Propiedades de configuración”, desplegar “Vinculador”, seleccionar “Entrada”, en dependencias adicionales ingresar “VCollide.lib RAPID.lib” (Figura A.1) y dar clic en aceptar.

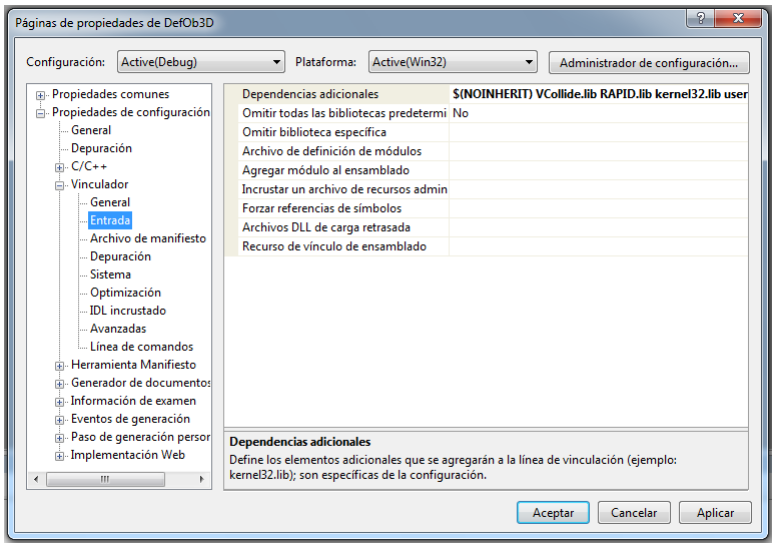


Figura A.1: Configuración VCollide

En el menú “Herramientas” seleccionar “Opciones…” desplegar “Proyectos y soluciones” y luego dar clic en “Directorios de VC++”, en “Archivos de inclusión” agregar las siguientes líneas,

C:\vcollide-2.01\include
C:\rapid-2.01

En “Archivos de biblioteca” agregar,

C:\vcollide-2.01\lib
C:\rapid-2.01

Después en el “explorador de soluciones” dar clic en DefOb3D, luego ir al menú Generar y seleccionar “Generar DefOb3D”, compilando la aplicación.

Posteriormente ir a la ubicación C:\Qt\4.8.0\bin y copiar los archivos “QtCored4.dll y QtGuid4.dll”, después ir a la ubicación C:\DefOb3D\build\debug y pegar estos archivos. Finalmente ejecutar en esta misma ruta el archivo DefOb3D.exe que es el ejecutable de la aplicación software 3D.

Anexo B. MANUAL DE USUARIO

Al ejecutar la aplicación se abrirá la ventana de la Figura B.1 que consta de una barra de menús y una barra de herramientas.

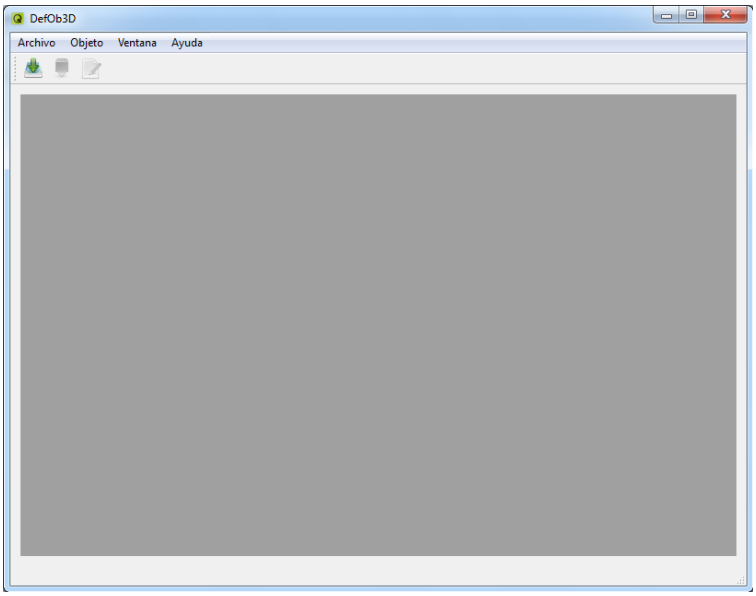


Figura B.1: Interfaz gráfica

La barra de menús consta del menú: Archivo, Objeto, Ventana y ayuda.

En la sección Importar del menú Archivo permite agregar un objeto 3d a la escena en los formatos vtk, obj, vtp o ply.

La sección Convertir del menú Objeto permite convertir el objeto 3d rígido a objeto deformable (Figura B.2).

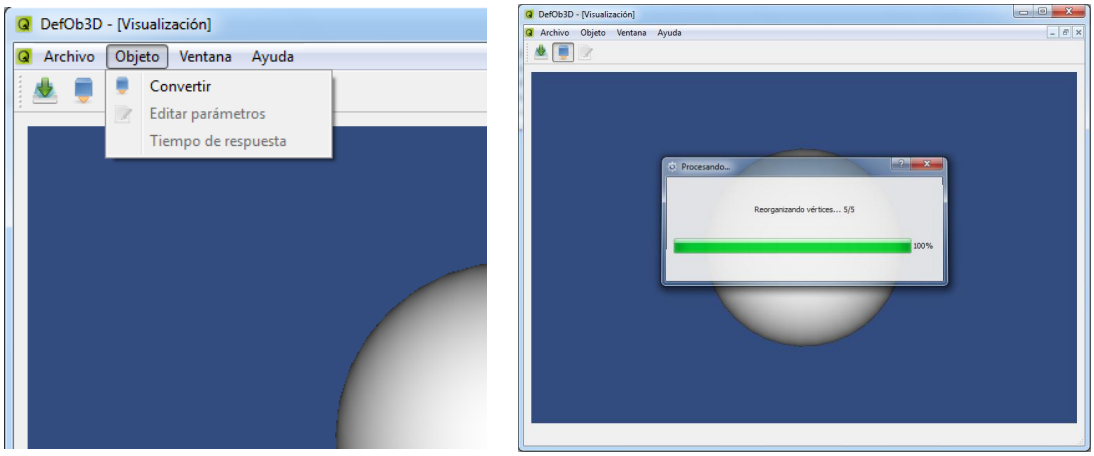


Figura B.2: Acción convertir

Cuando la aplicación termina de convertir el objeto rígido a objeto deformable aparecerá un cursor 3d en forma de lápiz que permitirá deformar el objeto. Para el manejo del cursor

3d se utiliza el movimiento del ratón, el movimiento en profundidad del cursor 3d se realiza manteniendo presionada la tecla barra espaciadora y moviendo el ratón hacia arriba o hacia abajo.

Para realizar una acción de estiramiento del objeto se debe hacer contacto en el punto deseado y luego mantener presionado el botón izquierdo del ratón, que posteriormente haciendo un arrastre del punto de contacto se logra un estiramiento del objeto deformable (Figura B.3).

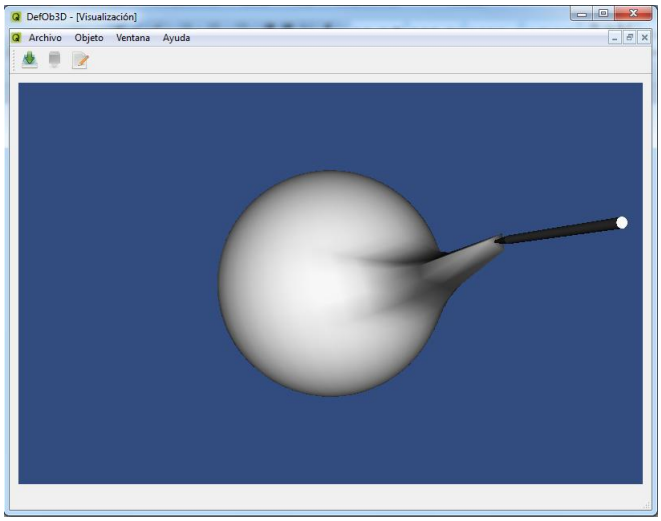


Figura B.3: Estiramiento

La sección Editar parámetros del menú Objeto permite modificar los parámetros del sistema masa resorte amortiguador de la restauración de forma del objeto deformable, con estos parámetros se define la dinámica de restauración del objeto deformable (Figura B.4).

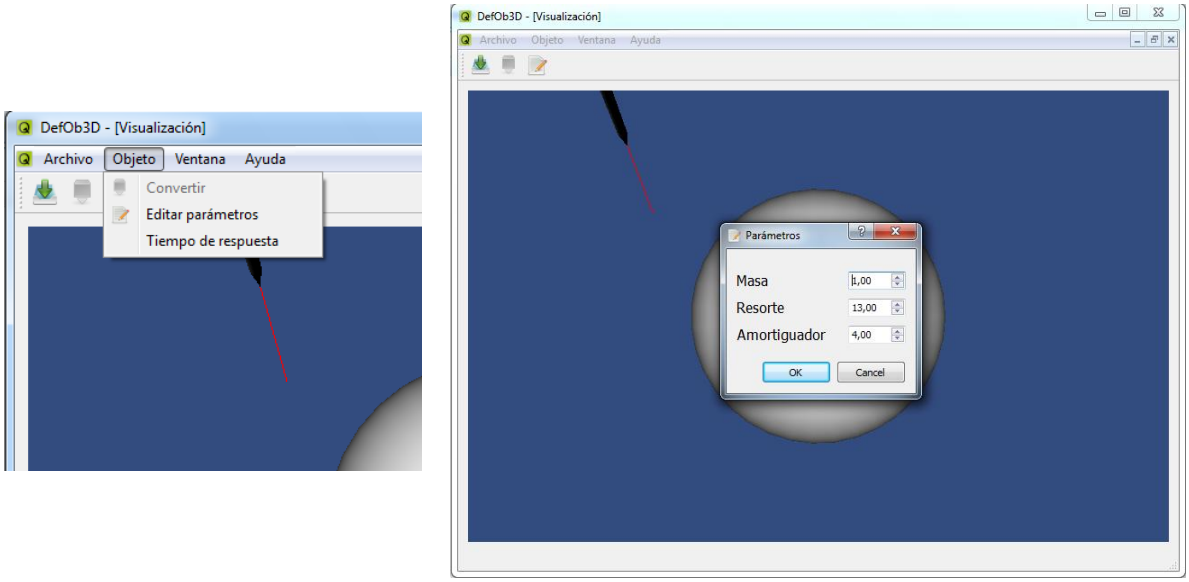


Figura B.4: Parámetros de restauración

La sección Tiempo de respuesta del menú Objeto permite graficar en pantalla la restauración del objeto. La Figura B.5 muestra el intervalo de tiempo que el usuario debe ingresar para realizar la gráfica. La advertencia que muestra la ventana significa que mientras se está graficando la restauración, el cursor 3d no podrá ser usado en el intervalo de tiempo seleccionado.

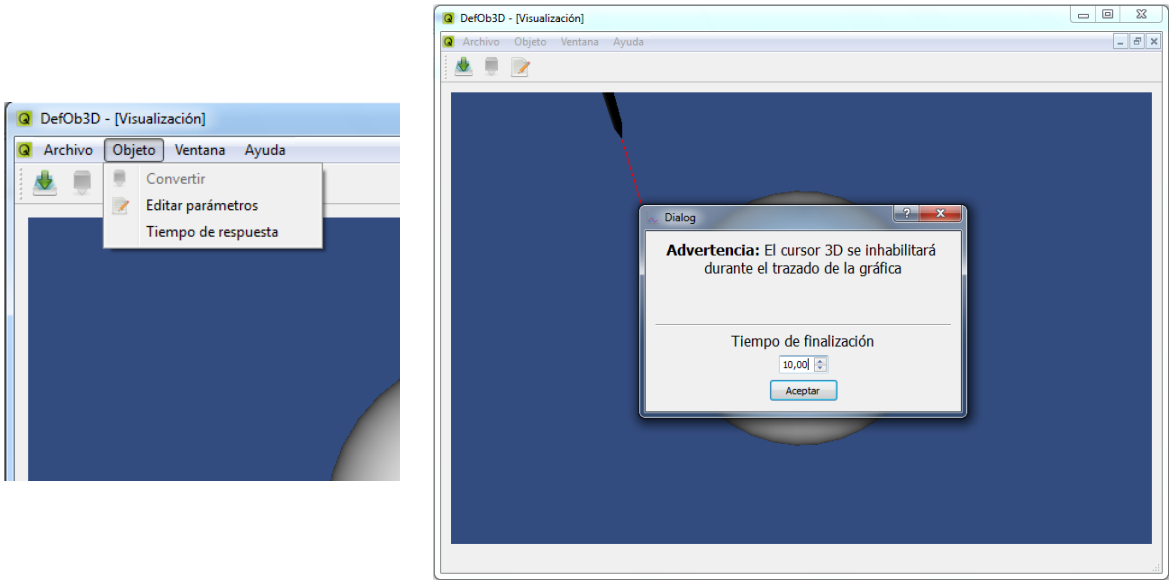


Figura B.5: Tiempo de respuesta

Al dar clic en aceptar aparecerá una ventana como muestra la Figura B.6. En la parte derecha de la ventana se grafica la restauración del objeto. En la parte inferior izquierda de la ventana permite medir los cuadros por segundo (FPS) de la aplicación.

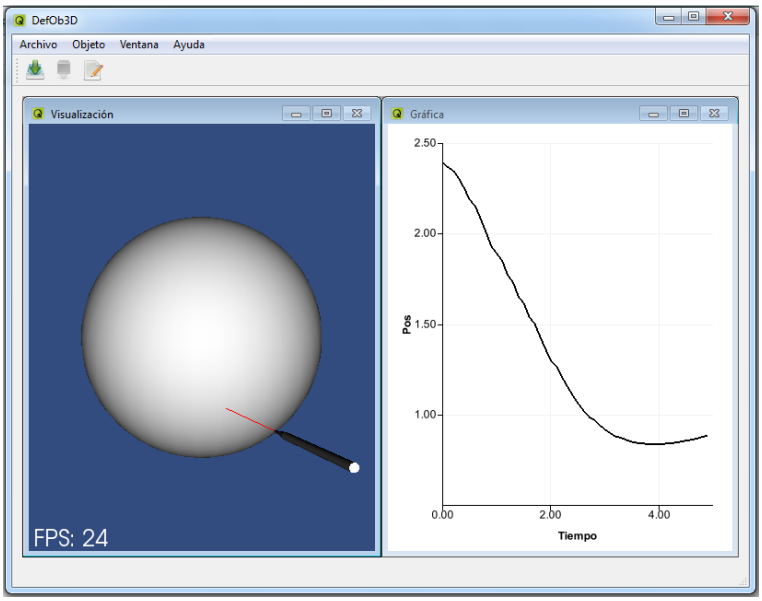


Figura B.6: Gráfica y FPS

Para cerrar la ventana de la gráfica se debe desmarcar el botón “Tiempo de respuesta” del menú Objeto (Figura B.7).

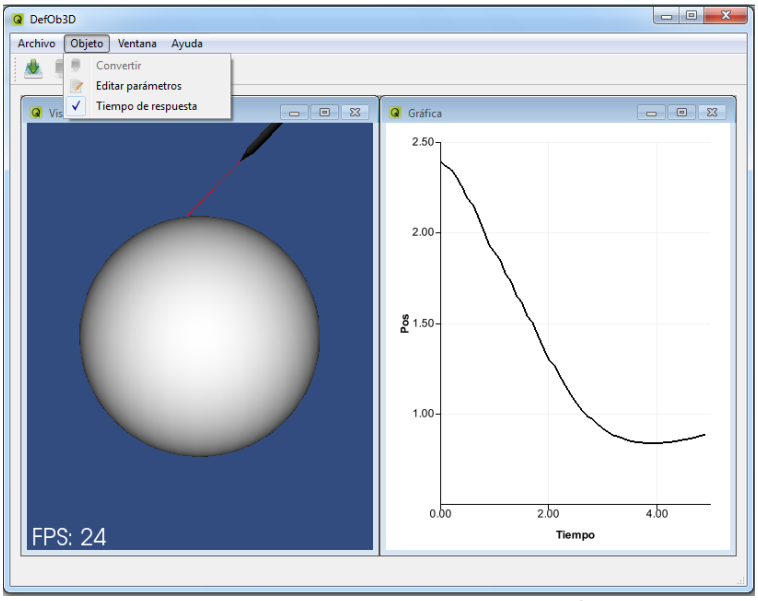


Figura B.7: Cierre de la ventana gráfica

La sección de visualización del menú Ventana permite abrir la ventana del escenario virtual en caso de haberse cerrado.

El menú Ayuda consta del botón “Ayuda de DefOb3D” que contiene información del manejo de profundidad del cursor.

Anexo C. IMPLEMENTACIÓN CON vtkCollisionDetectionFilter

En este anexo se presenta una versión de la aplicación software 3d que utiliza una herramienta de colisiones distinta a VCollide. La implementación de esta aplicación con la clase vtkCollisionDetectionFilter es muy similar a la implementada con VCollide con pequeñas variaciones en la estructura general de la aplicación.

C.1. Clases de la aplicación

Esta versión de la aplicación dentro de las clases base utiliza una herramienta para el manejo de detección colisiones llamada vtkCollisionDetectionFilter (Figura C.1).

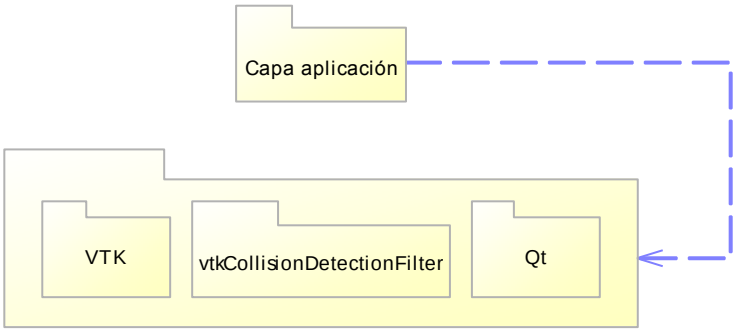


Figura C.1: Diagrama de paquetes

C.2. Estructura general de la aplicación software 3d

La figura C.2 muestra la estructura de la aplicación software con sus respectivos componentes.

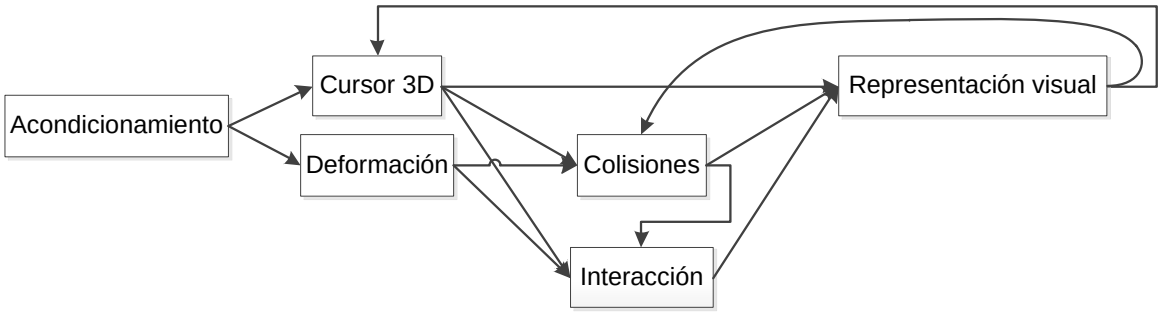


Figura C.2: Componentes de la aplicación software

C.3. Descripción general de la aplicación

La Figura C.3 ilustra el procedimiento general para el funcionamiento de la aplicación. vtkCollisionDetectionFilter evalúa colisiones entre los objetos cursor 3d y el objeto 3d importado, vtkPolyDataMapper genera las primitivas gráficas a partir del puerto de salida de la clase vtkCollisionDetectionFilter.

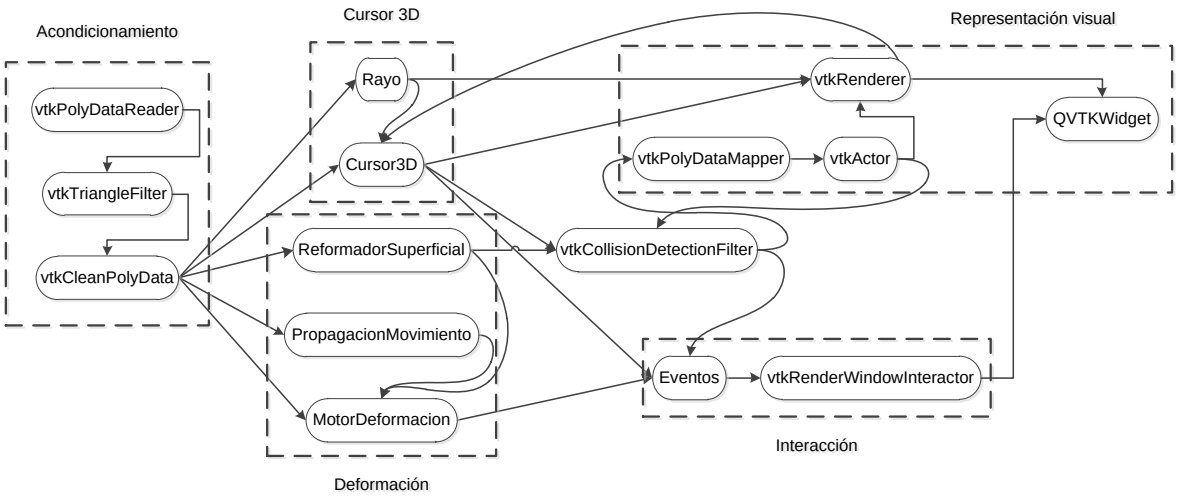


Figura C.3: Estructura general de la aplicación

C.4. Componente colisiones

La clase vtkCollisionDetectionFilter utiliza el método OBBTree que son estructuras de datos para calcular rápidamente una lista de puntos de contacto donde se cruzan dos mallas.

La Figura C.4 muestra los elementos necesarios para la clase vtkCollisionDetectionFilter. Objeto y cursor son los objetos a colisionar, los elementos Matriz indica a la instancia de la clase la posición y orientación de los objetos a colisionar, Número de pares de contacto permite saber si la colisión entre estos objetos se ha dado verificando que este valor sea diferente de cero, Celda de contacto permite identificar el instante y lugar preciso de contacto entregando el valor de identificación de la celda que se contacta al existir una colisión.

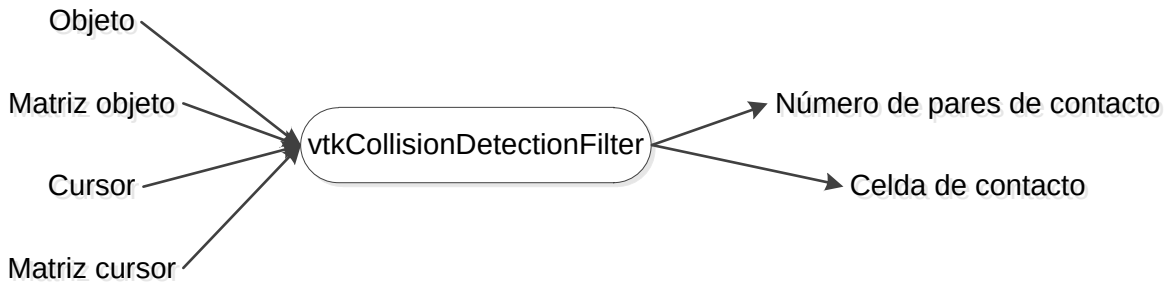


Figura C.4: Clase vtkCollisionDetectionFilter

A continuación se representa en código el esquema de la Figura C.4.

```
colisiones->SetInput( 0, reformador->GetSalida() );
colisiones->SetMatrix( 0, actor->GetMatrix() );
colisiones->SetInput( 1, cursor->GetSalida() );
```

```
colisiones->SetMatrix( 1, cursor->GetMatriz() );
colisiones->GetNumberOfContacts();
colisiones->GetContactCells(0)->GetValue(0);
```

La clase vtkCollisionDetectionFilter se configuró para reportar una sola celda, de manera que este sistema entregará el valor del primer contacto que se registre entre el cursor y el objeto deformable.

C.5. Resultados y pruebas

Las pruebas realizadas se hicieron con las siguientes características hardware.

Procesador: AMD Athlon Dual-Core 1.90 GHz.
Memoria: 2,0 GB
Tarjeta gráfica: GeForce 8200M G 256 MB

Tiempo de restauración

En esta etapa se realizan pruebas con esferas de resoluciones diferentes para ver como se ve afectado el tiempo de respuesta de restauración del objeto. Este indicador nos permite observar como el objeto deformable responde de acuerdo a la resolución del objeto ya que se requieren de mas cálculos para procesar y por ende un retardo de tiempo en la recuperación de forma del objeto.

Se hicieron tres pruebas con una esfera cuya resolución es incrementada por cada ensayo. La tabla C.1 muestra las características de resolución de los modelos de prueba.

En la gráfica de restauración de forma se mide la magnitud del vector de uno de los vértices del triangulo de contacto con respecto al tiempo.

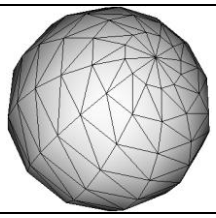
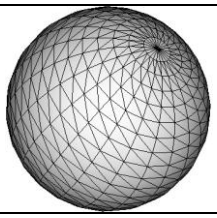
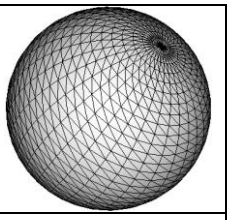
Imagen del modelo			
Modelo	Modelo 1	Modelo 2	Modelo 3
Triángulos	240	1056	2310
Vértices	122	530	1157

Tabla C.1: Modelos de prueba

La figura C.5 muestra el tiempo que le toma al modelo 1 recuperar su forma original.

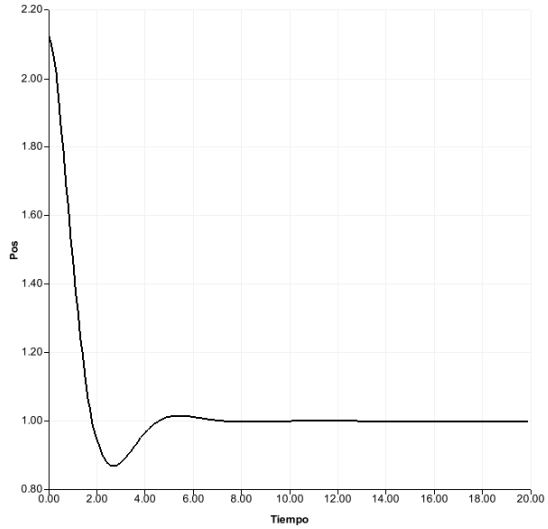


Figura C.5: Tiempo de respuesta modelo 1

La figura C.6 muestra que al modelo 2 le toma alrededor de 14 segundos recuperar su forma original en comparación de los 7 segundos que le toma al modelo 1.

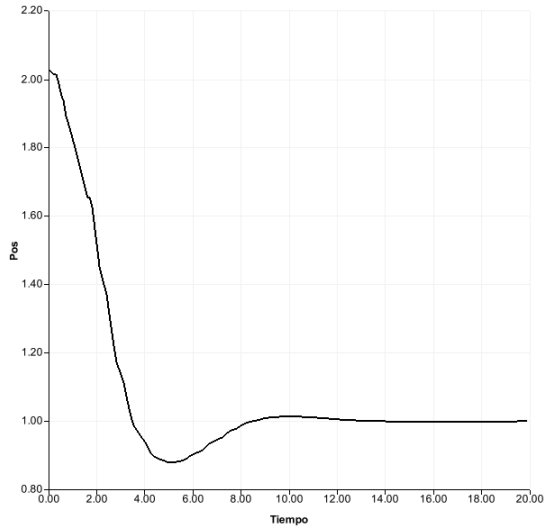


Figura C.6: Tiempo de respuesta modelo 2

Finalmente la figura C.7 muestra que el modelo 3 le toma aproximadamente 18 segundos recuperar su forma original.

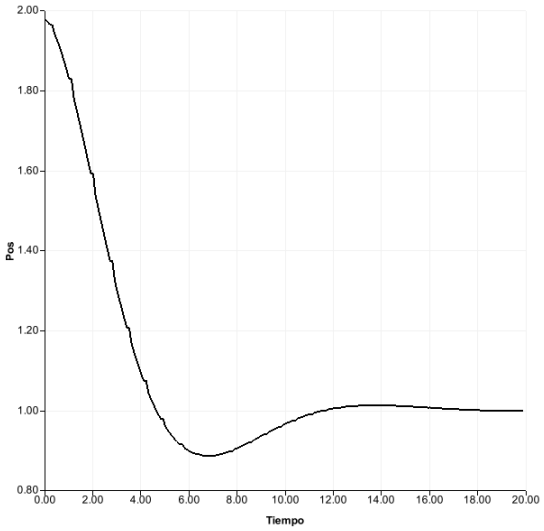


Figura C.7: Tiempo de respuesta modelo 3

Desempeño de visualización

El desempeño de la aplicación se ve afectado en gran parte por la renderización de objetos de alta resolución, en este caso se hacen pruebas de renderización de objetos de diferentes resoluciones para ver como se ve afectada la aplicación. La tabla C.2 muestra que a medida que la resolución en los objetos aumenta los frames disminuyen. La caída de los frames es causa de la actualización de la malla del objeto al renderizar su deformación y también a que la clase vtkCollisionDetectionFilter actualiza toda la malla.

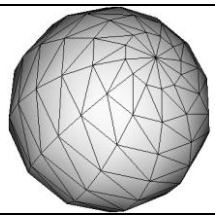
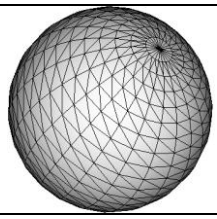
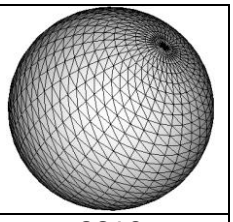
Modelo			
Triángulos	240	1056	2310
Vértices	122	530	1157
FPS	66	13	6

Tabla C.2: Cuadros por segundo según la resolución

Anexo D. EJEMPLOS DE APLICACIONES VTK

En este anexo se muestran pequeñas aplicaciones que permiten observar las funcionalidades base de la aplicación.

D.1. Cursor 3D

Esta aplicación ejecuta el sistema del cursor 3D.

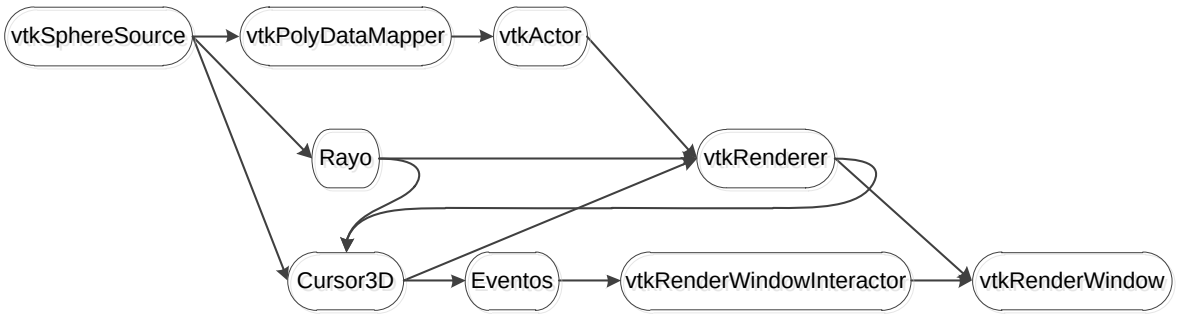


Figura D.1: Estructura de la aplicación cursor 3D

D.2. Colisiones

Esta aplicación ejecuta el sistema de colisiones con la clase vtkCollisionDetectionFilter.

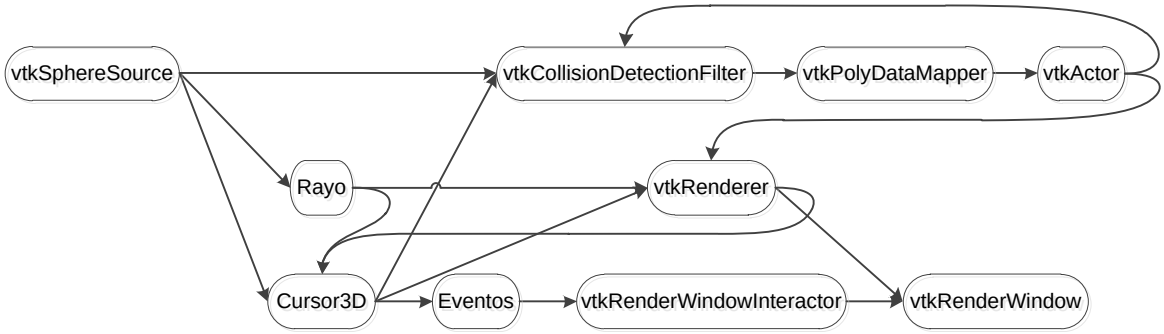


Figura D.2: Estructura de la aplicación colisiones

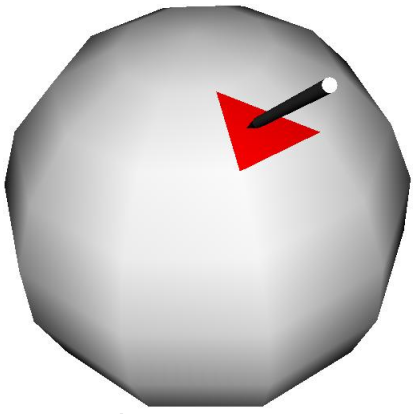


Figura D.3: Colisión entre el cursor 3D y el objeto

D.3. Reformador de superficie

Esta aplicación ejecuta el sistema de reformador de superficie de un objeto.

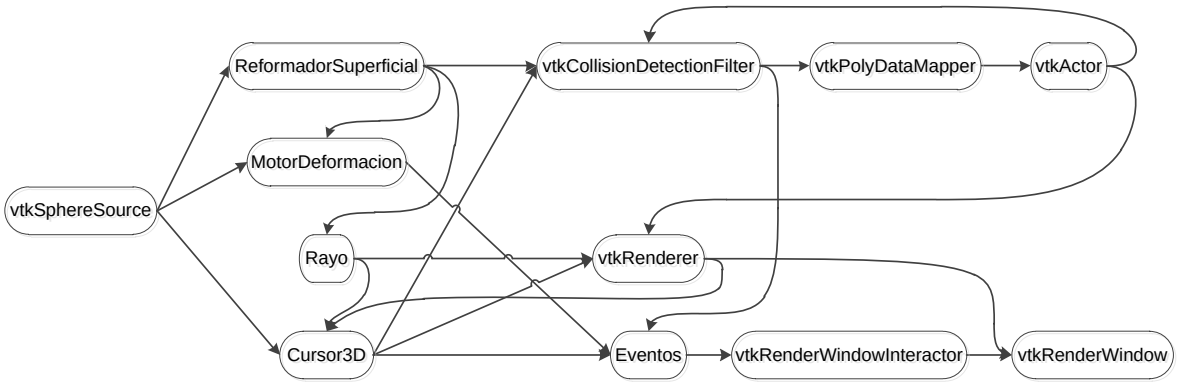


Figura D.4: Estructura de la aplicación reformador

La figura D.5 muestra cómo se modifica la posición de un triángulo en una esfera.

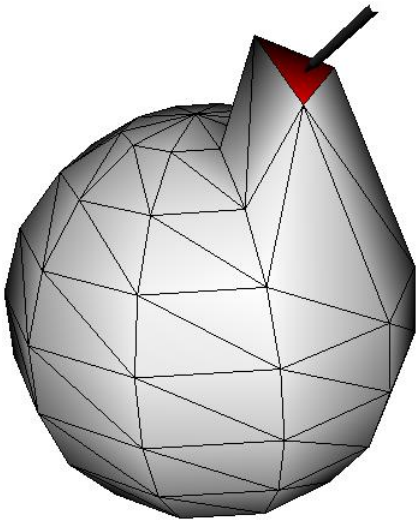


Figura D.5: Modificación de superficie

D.4. Propagación

Esta aplicación ejecuta el sistema de propagación en una malla triangular.

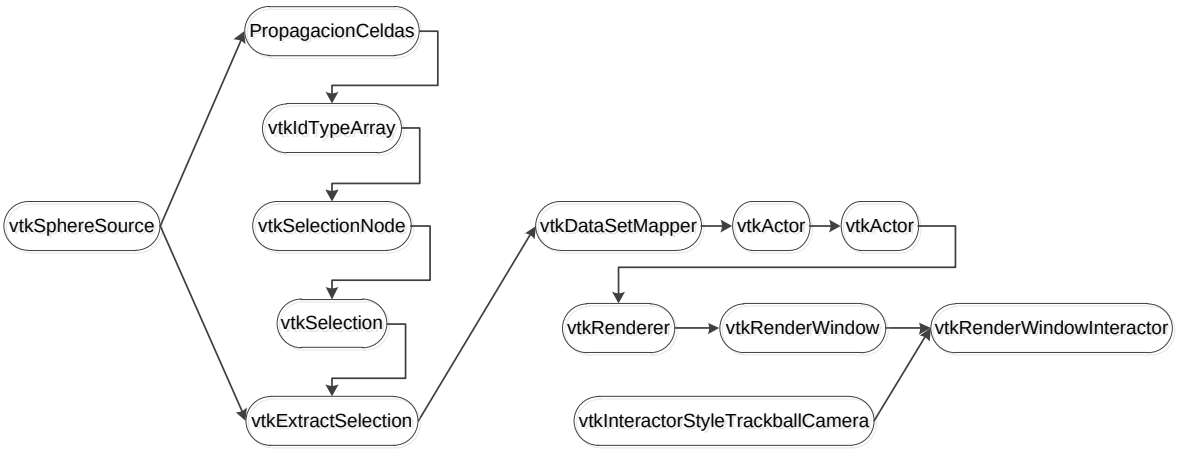


Figura D.6: Estructura de la aplicación propagación

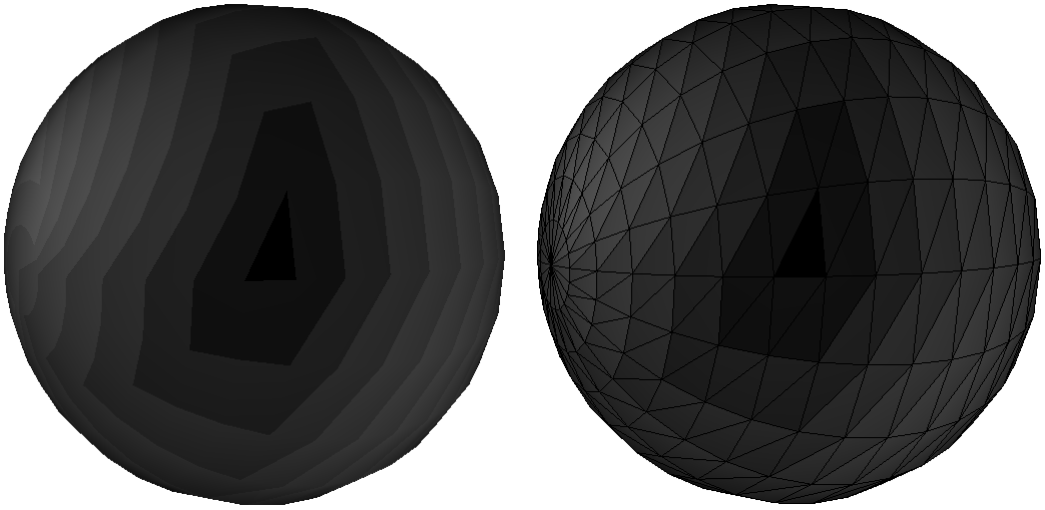


Figura D.7: Propagación a partir de un triángulo

D.5. Uso del timer con Qt

Esta aplicación ejecuta el sistema del timer. El programa implementado consiste en alternar dos colores a un objeto cada 500 milisegundos.

En Qt la clase que permite el uso del timer es “QTimer”.

```
timer = new QTimer;
```

La instancia timer de la clase QTimer envía una acción como señal cada 500 milisegundos al slot denominado “Pulso”.

```
connect(timer, SIGNAL(timeout()), this, SLOT(Pulso()));
```

La configuración del tiempo de muestreo se ilustra a continuación.

```
timer->start(500);
```

El slot Pulso es una función que se ejecuta a partir de la señal del timer, en este caso se ejecutará cada 500 milisegundos.

```
void MainWindow::Pulso()
{
	if(estado == false)
	{
		actor->GetProperty()->SetColor(1.0, 0.0, 0.0);
		estado = true;
	}
	else
	{
		actor->GetProperty()->SetColor(1.0, 1.0, 1.0);
		estado = false;
	}

	ui->widget->update();
}
```

De esta forma la instancia actor de la clase vtkActor cambia el color del objeto (Figura D.8) alternando entre blanco y rojo.

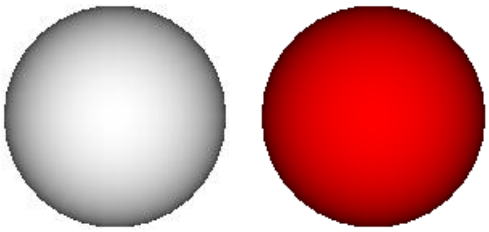


Figura D.8: Permutación de colores con timer

REFERENCIAS BIBLIOGRÁFICAS

[1] Patrick Crawford, "A Novel System For Generating Interactive Context-Preserving Cutaways Of Anatomical Surface Meshes," Ryerson University, Ontario, Tesis de maestría en ciencia 2010.