Tecnologías Multimedia, Curso 10/11 - Práctica 5

Desarrollo de aplicaciones multimedia (I)

Programación

Objetivo:

El objetivo de esta práctica es conocer las posibilidades que ofrecen para el desarrollo de aplicaciones multimedia mediante programación dos de las arquitecturas multimedia más difundidas, como son la *DirectShow* de *Microsoft* y la *Java Media Framework* de *Sun*. Para ello se utilizarán sus entornos de desarrollo y se trabajará sobre sencillas aplicaciones de ejemplo.

Desarrollo:

- **Programación con *DirectShow***
  - Descargar e instalar DirectX SDK
  - Acceder a través del enlace de la *web* de asignatura y también de *Inicio-Programas-Microsoft DirectX 9.0 SDK-DirectX Documentation for C++* a la exhaustiva documentación sobre la arquitectura *DirectX*. Bucear un poco en ella para darse cuenta de todas las posibilidades de programación

  - **Construcción de *Players* con *DirectShow***
    - Acceder a través de *Inicio-Programas-Microsoft DirectX 9.0 SDK-DirectX Utilities* al programa *GraphEdit* , que sirve para realizar prototipos de aplicaciones basadas en *DirectShow* de forma gráfica
    - Utilizar la opción *Graph-Insert Filters* y analizar todos los posibles filtros con los que construir nuestro gráfico de filtros
    - Insertar un filtro de entrada desde fichero mediante la opción *DirectShow Filters-File Source (Async.)*. Seleccionar el archivo de vídeo que se quiere asociar al filtro de entrada. Pinchar con el botón derecho del ratón sobre el *pin* de salida del filtro seleccionado y elegir la opción *Render Pin*. Analizar toda la cadena de filtros que es necesaria para poder visualizar el vídeo
    - Seleccionar el botón *Play* del menú y comprobar que ocurre lo esperado. Minimizar el programa *GraphEdit* para utilizarlo de nuevo más adelante
    - Ahora vamos a construir un reproductor básico mediante programación. El código fuente Player1.cpp corresponde a un ejemplo incluido en la documentación de *DirectShow*. Las ordenes de compilación y enlazado necesarias para obtener el ejecutable están en el archivo de comandos generaP1, que hay que guardar sin extensión
    - Descargar los archivos, editar el código fuente y buscar la línea de programa donde se especifica el archivo de vídeo a renderizar. Descargar ahora el citado archivo lake.mpg y asegurarse de que se guarda con esa extensión
    - Descargar e instalar el cliente de *telnet* seguro SSHSecureShellClient. Acceder mediante el a la máquina *mizar.edv.uniovi.es*, donde está instalado el entorno *Microsoft VisualStudio .NET 2003*, el cual incluye todas las herramientas necesarias para desarrollar en *C/C++*. Una vez conectado (preguntar los datos de acceso al profesor), crea una carpeta de nombre igual a tu número de matricula y trabaja allí en lo sucesivo
    - Abrir una sesión *ftp* desde el mismo cliente y transferir a una carpeta en la máquina remota los 2 archivos del primer ejemplo. En la máquina remota, dar permiso de ejecución al fichero de comandos mediante "*chmod +x generaP1"*. Ejecutar el archivo para generar el ejecutable del reproductor. Transferir el ejecutable a nuestra máquina y para comprobar el funcionamiento abrir una ventana de comandos mediante *Inicio-*

Programas-Accesorios-Simbolo de sistema, ir a la carpeta correspondiente y ejecutar "Player1" (el vídeo debe encontrarse en la misma carpeta)

- Repetir los pasos anteriores con el segundo programa de ejemplo, capaz de ejecutar simultáneamente hasta 5 archivos de audio/vídeo que le pasemos como parámetros. Los archivos necesarios en este caso para generar el ejecutable son Player2.cpp y generaP2
- Descargar en la carpeta del ejemplo más archivos multimedia: mortadelo.avi, chinese_dance.wav, piano.mp3 y también uno de los vídeo grabados por ti de la tele. Para comprobar el funcionamiento ejecutar "*Player2 archivo1 archivo2 ...*"
- Repetir los pasos anteriores con el tercer programa de ejemplo, el cual sincroniza un vídeo con una banda sonora independiente mediante una gestión de eventos adecuada. Los archivos necesarios en este caso para generar el ejecutable son Player3.cpp y generaP3. Comprueba el funcionamiento de este último ejemplo mediante el comando *"Player3 piano.mp3 lake.mpg"*
- Repetir los pasos habituales con el cuarto programa de ejemplo, un reproductor *windows*. Los archivos necesarios en este caso para generar el ejecutable son WinPlayer.cpp, WinPlayer.rc, resource.h y generaWP

- **Captura y compresión con *DirectShow***
  - Vamos ahora a capturar vídeo. Conectar la cámara *web* al PC y comprobar que funciona (con *VirtualDub* por ejemplo)
  - Volver al programa *GraphEdit*. Seleccionar un nuevo gráfico mediante la opción *File-New* e insertar ahora el filtro correspondiente a la cámara dentro de *Video Capture Sources*. Pinchar con el botón derecho del ratón sobre el pin de salida de captura del filtro seleccionado y elegir la opción *Render Pin*. Analizar toda la cadena de filtros que es necesaria para poder visualizar el vídeo capturado
  - Seleccionar el botón *Play* del menú y comprobar que ocurre lo esperado. Minimizar el programa *GraphEdit* para utilizarlo de nuevo más adelante
  - Vamos a generar ahora el ejecutable correspondiente a una de las aplicaciones de captura que viene con la instalación de *DirectX*, cuyo código fuente podemos encontrar en la ruta *Samples/C++/DirectShow/Capture/AMCap* a partir de la carpeta de instalación
  - Transferir a la máquina remota todos los archivos fuente **.cpp, *.rc, *.h, *.jpg y *.ico*. Transferir finalmente el archivo de comandos generaAMC. Realizar los pasos habituales y probar el ejecutable final. Comprobarás que para poder ejecutar la aplicación será necesario que tengas instaladas en *D:\WINDOWS\System32* las *dlls* msvcr71.dll y mfc71d.dll que incorpora *VisualStudio* y que también puedes descargar desde la página *web* de la asignatura
  - Vamos ahora a comprimir vídeo. Volver al programa *GraphEdit* y seleccionar mediante *File-Open Graph* el archivo compresion-xvid.GRF, que corresponde a un gráfico de filtros preparado para comprimir la parte de vídeo del archivo mortadelo.avi. El gráfico necesita que el archivo esté en *C:\tmp* y en esa misma carpeta almacenará el archivo comprimido
  - Seleccionar el botón de *Play* del menú y, una vez comprimido el archivo, comprobar tanto su tamaño como su correcta reproducción
  - Vamos a generar ahora el ejecutable correspondiente a una de las aplicaciones de compresión que viene con la instalación de *DirectX*, cuyo código fuente podemos encontrar en la ruta *Samples/C++/DirectShow/Editing/CompressView* a partir de la carpeta de instalación
  - Transferir a la máquina remota todos los archivos fuente **.cpp, *.rc y *.h y *.jpg*, además de la carpeta *Res*. Transferir también los archivos *dshowutil.cpp* y *dshowutil.h* de la ruta *Samples/C++/DirectShow/Common*. Transferir finalmente el archivo de comandos generaCV. Realizar los pasos habituales y probar el ejecutable final

- Acceder a través de *Inicio-Programas-Microsoft DirectX 9.0 SDK-DirectX Sample Browser* a la colección de ejemplos basados en DirectX para comprobar sus múltiples posibilidades

- **Programación con *Java Media Framework (JMF)***
  - Preparación del entorno *Java Media Framework*
    - Descargar e instalar sucesivamente el entorno Java JRTE, la plataforma Java J2 SDK y el sistema de desarrollo JMF SDK, seleccionando para los dos últimos como carpetas de instalación unas que cuelguen directamente de la unidad C (p.e. ***C:\j2sdk*** *y* ***C:\jmf)***
    - Actualizar las variables de entorno del sistema para poder trabajar desde una ventana de comandos. Para ello ir a *Panel de Control-Sistema-Opciones Avanzadas-Variables de entorno-Variables del sistema* y añadir a la variable *Path* la cadena "***;C:\j2sdk\bin***"

  - Acceder a través de *Inicio-Programas-Java Media Framework* a la utilidad *JMFRegistry*. Bucear un poco por ella para darse cuenta de las posibilidades de configuración de *JMF*, así como de los tipos soportados, las librerías instaladas y los dispositivos de captura y *plug-ins* registrados

  - **Construcción de *Players* con *JMF***
    - Descargar el código SimpleAudioPlayer.java, correspondiente a un primer ejemplo de reproductor solo de audio desde la línea de comandos. Abrir una ventana de comandos, compilarlo mediante *"javac SimpleAudioPlayer.java"* y ejecutar la clase generada mediante *"java SimpleAudioPlayer archivo_audio"*. Probar el funcionamiento con archivos de audio de diferentes tipos
    - Descargar el código MediaPlayerFrame.java, correspondiente a un reproductor de audio/vídeo desde la línea de comandos. Abrir una ventana de comandos, compilarlo y ejecutar la clase generada para comprobar su funcionamiento con archivos de audio y vídeo de diferentes tipos
    - Descargar el código PlayerApplet.java, correspondiente a un *applet* para reproducción de audio/vídeo desde un navegador. Abrir una ventana de comandos, compilarlo y ejecutar la clase generada para comprobar su funcionamiento mediante *"appletviewer prueba.html"*. A continuación, comprobar su funcionamiento desde los navegadores habituales incorporando el código disponible en el archivo de prueba anterior en cualquier punto de la página *web* que se quiera. Si hay problemas con la visualización del *applet* desde los navegadores, asegurarse de que JMF se ha instalado estrictamente después de instalar el entorno Java.

  - **Captura y compresión con *JMF***
    - Descargar el código JMF.java y CaptureDeviceDialog.java, correspondiente a un ejemplo de aplicación de captura de vídeo. Abrir una ventana de comandos, compilarlo mediante *"javac *.java"* y ejecutar la clase *JMF* generada. Comprobar su funcionamiento con la cámara web, para lo cual será necesario registrar previamente el nuevo dispositivo de captura con la herramienta *JMFRegistry*
    - Descargar el código Java Media Applications, correspondiente a las aplicaciones *JMF Registry* (ya analizada) y *JMStudio* (programa demostrador de *JMF*, que permite la captura, compresión y reproducción de medios). Abrir una ventana de comandos, editar el archivo *mk-jmapps.bat* utilizado para generar las aplicaciones y escribir los valores adecuados en las variables *JAVA HOME* y *JMFHOME*. Guardar el archivo, ejecutarlo y comprobar el funcionamiento de las aplicaciones generadas, en especial el de *JMStudio*, capturando de la cámara *web* y comprimiendo archivos de vídeo.