UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

ARQUITECTURA PARA APLICACIONES GRÁFICAS REMOTAS E INTERACTIVAS EN DISPOSITIVOS MÓVILES

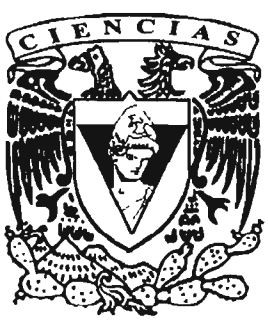
T E S I S

QUE PARA OBTENER EL TÍTULO DE:

MATEMÁTICO CON ORIENTACION EN CIENCIAS DE LA COMPUTACION

P R E S E N T A :

JOSÉ LUIS PEREA GUZMÁN



FACULTAD DE CIENCIAS
UNAM

DIRECTORA DE TESIS: MAT. MARÍA CONCEPCIÓN ANA LUISA SOLÍS GONZÁLEZ COSÍO

2005

VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

"Arquitectura para aplicaciones gráficas remotas e interactivas en dispositivos móviles"

realizado por PEREA GUZMAN JOSE LUIS con número de cuenta 08816253-5

quién cubrió los créditos de la carrera de MATEMATICAS CON ORIENTACION EN CIENCIAS DE LA COMPUTACION

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	Mat. María Concepción Ana Luisa Solís González Cosío
Propietario	Dra. Hanna Oktaba
Propietario	Dra. Amparo López Gaona
Suplente	M. en C. Miguel Miranda Miranda
Suplente	M. en C. Jorge Luis Ortega Arjona

Consejo Departamental de Matemáticas

M. en C. Alejandro Bravo Mojica

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMATICAS

Agradecimientos

A mis padres: Maria Agustina y Bernardino†, quienes me dieron la vida y se esforzaron por darme educación.

A mi hermano Bernardino, por su apoyo incondicional y sus palabras de aliento.

A mis maestros, que contribuyeron a mi formación.

A mi directora de tesis Ana Luisa Solís, a quien agradezco la paciencia y dedicación al dirigir este trabajo.

A mis sinodales, cuyas observaciones ayudaron a enriquecer este trabajo.

Índice

Capítulo 1
Introducción

En los últimos años las tecnologías de la información y comunicaciones se han desarrollado rápidamente. El uso de computadoras y el acceso a recursos de información se han vuelto parte necesaria en la vida de muchas personas. Hay numerosos equipos y aplicaciones que permiten realizar las más diversas tareas y obtener grandes volúmenes de información de todo tipo. Sin embargo, el acceso a estos recursos se restringía a localidades fijas. La necesidad de las personas para tener acceso a estos recursos a pesar de estar en tránsito o lejos de sus equipos resultó en el desarrollo del computo móvil.

Muchos de estos equipos móviles permiten realizar tareas equivalentes a sus contrapartes fijas: hojas de cálculo, procesadores de texto, navegadores de Internet, juegos, bases de datos son comunes en varios de estos equipos portátiles. Aunque las prestaciones de estas aplicaciones están un tanto limitadas por las capacidades del hardware de los propios equipos, tienen un nivel que es aceptable para los usuarios.

1.1 Asistentes Personales Digitales

Los Asistentes Personales Digitales o PDA (Personal Digital Assistant) son equipos de cómputo móvil cuyo tamaño permite que quepan en la palma de la mano con capacidades limitadas pero similares a equipos de escritorio y laptops. Existen también los equipos denominados smartphones, teléfonos celulares con capacidades de cómputo integradas que además tienen acceso a redes públicas. La tendencia es fusionar ambos conceptos en un solo dispositivo.

El uso de estos equipos actualmente va más allá de las funciones de una agenda y directorio telefónico. Existen equivalentes para las aplicaciones más populares, como los procesadores de texto u hojas de cálculo. También hay aplicaciones para el entretenimiento, como reproductores de audio y video de diversos formatos. Sin embargo, en el área de las imágenes y gráficas generadas por computadora (algunas muy complejas) no existen muchas aplicaciones.

OpenGL es un estándar ampliamente utilizado para generar imágenes tridimensionales en varias plataformas de cómputo. Sin embargo, el estándar para equipos PDA's, llamado OpenGL ES, está por implementarse [KG04]. Este estándar permitirá la portabilidad de aplicaciones gráficas hacia estos dispositivos, pero la calidad de las imágenes estará limitada por la capacidad de procesamiento y almacenamiento de cada equipo, sobre todo si se trata de producirlas en tiempo real.

En este trabajo se propone la construcción de una aplicación que permita visualizar modelos gráficos complejos, ubicados en equipos fijos, en un dispositivo portátil, además de poder interaccionar en forma remota.

1.2 Objetivos del proyecto

Este trabajo se enfoca en el estudio de técnicas y herramientas para el desarrollo de una aplicación gráfica remota en sistemas de cómputo móvil. Los objetivos son:

- Revisar las técnicas y herramientas para el desarrollo de aplicaciones de computo móvil.
- Revisar las arquitecturas de red disponibles.
- Revisar las arquitecturas gráficas disponibles.
- Desarrollar una aplicación móvil para visualizar y controlar modelos gráficos de manera remota.

1.3 Alcance del proyecto

- Definición de las herramientas para el desarrollo de una aplicación tipo cliente servidor para un dispositivo móvil.
- Definición de la arquitectura de red necesaria para la aplicación.
- Definición de la arquitectura gráfica necesaria para la aplicación.
- Diseño y construcción de una aplicación de software prototipo.

1.4 Organización del trabajo

El primer capítulo hace la presentación con el objeto de introducir al tema y a la aplicación que serán estudiados en los siguientes capítulos.

El segundo capítulo presenta el cómputo móvil. Se revisa la evolución histórica de los dispositivos portátiles. Se revisan las principales plataformas de software y las herramientas de desarrollo disponibles para estos equipos.

En el tercer capítulo se revisa la arquitectura gráfica disponible en los equipos móviles. También se revisa la arquitectura de las redes inalámbricas que permiten la conexión con otros equipos. Además, se revisan algunos trabajos ya realizados sobre aplicaciones gráficas para estos dispositivos.

En el cuarto capítulo se describe la arquitectura, el diseño y la construcción de la aplicación prototipo en un sistema cliente-servidor. También se mencionan algunas características del funcionamiento de esta aplicación.

En el último capítulo se presentan las conclusiones y ejemplos de posibles aplicaciones en las que puede ser utilizado el prototipo desarrollado en este trabajo. También se mencionan las líneas de desarrollo que se pueden seguir para complementar este trabajo.

Capítulo 2
Dispositivos y Herramientas para Cómputo Móvil

Cómputo móvil se refiere a los equipos de cómputo portátiles y sus usuarios. Tales dispositivos tienen diversas formas y tamaños, así como una amplia capacidad de cómputo. Pueden conectarse a otros equipos por medio de diversas tecnologías de red. Pueden ir desde computadoras laptop de propósito general a dispositivos organizadores de datos. Pueden ser stand-alone, de conexión intermitente o de conexión permanente mediante redes inalámbricas. Aunque existe una gran variedad de combinaciones, lo que caracteriza al cómputo móvil es la asociación entre el dispositivo portátil y su usuario móvil. Este aspecto plantea problemas nuevos y ofrece nuevas oportunidades en el desarrollo de aplicaciones.

Una computadora móvil se asocia con un ambiente de recursos limitados. Aunque estas limitantes se notan menos conforme mejora la tecnología. La portabilidad de una computadora móvil inducirá, en mayor o menor grado, limitantes si es comparada con una computadora de escritorio. Por ejemplo, las computadoras móviles operadas por baterías tienden a tener limitantes de energía con respecto a sus contrapartes fijas.

2.1 Dispositivos portátiles

La evolución de los equipos portátiles se ha dado al menos desde dos vertientes: por un lado en las computadoras personales o PC's hacia las laptops, y por el otro, de las calculadoras digitales hacia los organizadores personales o PDA's. Recientemente ambas tecnologías tienden a converger, por lo que los dispositivos actuales heredan características de ambas tendencias. Existen PDA's que parecen mas una pequeña laptop, y hay laptops que parecen una PDA de mayor tamaño (Tablet PC's).

Las PC's fueron un primer intento en llevar el cómputo fuera de los edificios que contenían las grandes computadoras. Al igual que los equipos móviles actuales, inicialmente tenían limitantes, esencialmente en capacidad de almacenamiento y de procesamiento. Aun así, seguían siendo equipos estáticos que permanecían en casas, oficinas o escuelas. Uno de los primeros intentos por tener un equipo móvil fue realizado por Compaq, en 1983, con su

modelo Compaq Portable (ver figura 2.1) [HPH05]. Contaba con dos unidades de disco de 5 y ¼ de pulgada y un pequeño monitor monocromático de 9 pulgadas. Contenida en una caja liviana, el teclado funcionaba como una tapa que protegía el monitor y las unidades de disco. Se le podía cargar con cierta facilidad y podía ser usada en lugares donde hubiera una toma de corriente eléctrica. En 1984 IBM presentó al mercado un modelo portátil similar llamado IBM Portable PC, model 5155.

Figura 2.1 Compaq Portable

Figura 2.2 HP-110

En el mismo año apareció la primera laptop: la HP-110 (ver figura 2.2). Este modelo introdujo el uso de monitores LCD, baterías y el diseño en forma de libreta. La tapa contiene el monitor y la base, el teclado. Tiene las mismas prestaciones que las computadoras de escritorio, pero la batería integrada permite trabajar sin la necesidad de una toma de energía todo el tiempo. Dos años después IBM presenta su modelo 5140 Convertible PC, en la que agregó la opción de ocultar la pantalla LCD para usar un monitor convencional. El diseño de una laptop está orientado a ser compatible en hardware y software con los equipos de escritorio. Los modelos más recientes de laptops actualmente también son conocidos como notebooks (ver figura 2.3).

Fig. 2.3 Notebook

Una variante más reciente de estos equipos son las Tablet PC's (ver figura 2.4). Son del mismo tamaño que una laptop, con un monitor sensible al tacto que funciona como dispositivo de entrada. En algunos modelos el teclado se esconde detrás de la pantalla y en otros no existe. Muchos de estos modelos tienen accesorios que permiten instalarlas como computadoras de escritorio en un lugar fijo: se colocan sobre un soporte que contiene las conexiones con los periféricos y que a la vez recarga las baterías. Cuando no se utiliza el teclado la interacción del usuario con el equipo se logra mediante trazos de una pluma sobre la pantalla.

Fig. 2.4 Tablet PC

La evolución de los Asistentes Personales Digitales (o PDA por sus siglas en inglés) es diferente. La miniaturización y la reducción de los precios de procesadores y memorias aumento su disponibilidad, permitiendo mejorar las calculadoras personales. Inicialmente se aprovecharon las mejoras tecnológicas para agregarles la capacidad de almacenar y ejecutar programas. Posteriormente se les agregó agenda y directorio telefónico. Manejaban

pequeñas cantidades de información, pero era suficiente para un usuario promedio. Uno de estos primeros equipos fue la SHARP PC-1500 (ver figura 2.5). Fue creada en 1980, y permitía almacenar programas, conectar módulos de memoria y algunos equipos periféricos como una impresora. La PSION Organiser I (ver figura 2.5) fue creada en 1984. Al igual que la SHARP se podía programar, pero además tenía la funcionalidad de guardar datos personales. La versión mejorada PSION Organiser II introdujo el uso de más de una línea de texto, lo que derivó posteriormente en las amplias pantallas de las calculadoras modernas que inclusive pueden realizar gráficas de los datos que contienen.

Figura 2.5 SHARP PC-1500 y PSION Organiser

Desde 1992, Apple Computer concibió la idea de construir un dispositivo portátil que no requiriera teclado. La información se capturaría en la propia pantalla. La idea fue implementada y en 1993: surgió la Newton Message Pad, el primer dispositivo orientado principalmente al manejo de información personal [KR03] (ver figura 2.6). Su diseño es la base para todos los dispositivos portátiles sin teclado (o touch screen) que existen actualmente. Con este modelo también se introduce el concepto de la pluma, necesaria en estos equipos. Inicialmente, el reconocimiento del texto escrito en la pantalla se hizo mediante un diccionario de palabras. En versiones mejoradas se cambio este por un algoritmo de reconocimiento de caracteres, que es como funcionan actualmente los dispositivos sin teclado.

Figura 2.6 Newton Message Pad

La compañía Palm inició como un proveedor de sistemas de captura de información touch screen, también llamado sistema graffiti, para otras compañías fabricantes de PDA's. En 1995 inició la fabricación de sus propios dispositivos PDA [PO05]. Algunos modelos se muestran en la figura 2.7. Su primer organizador, el Palm Pilot-1000, era muy sencillo y limitado, pero su pequeño tamaño, su sistema de entrada que es una combinación del sistema graffiti y de botones reales, acompañada de un precio razonable, lo hizo muy popular. Las funciones principales de estos modelos es llevar una agenda, directorio, reloj, notas personales y calculadora. Posteriormente se les agregó correo electrónico. Existen muchos modelos pero el diseño es esencialmente el mismo. Estos equipos funcionan con el sistema operativo PalmOS que fue diseñado especialmente para ellos.

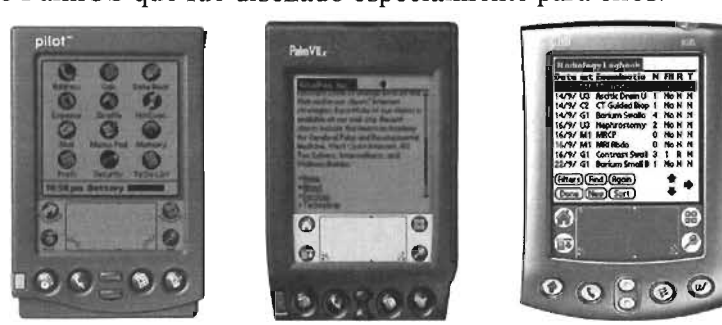


Figura 2.7 Varios modelos de Palm

Existe otro conjunto de equipos personales, son los basados en el sistema operativo Windows CE de Microsoft. Estos son las Pocket PC's y las Hanheld PC's. Las primeras son similares a los dispositivos tipo Palm, sin teclado. Las segundas son de tamaño un poco mayor y tienen teclado. En la figura 2.8 se muestran algunos modelos. La diferencia entre los dispositivos basados en PalmOS y Windows CE es que los segundos fueron concebidos desde el principio como computadoras de propósito general.

Figura 2.8 Equipos Windows CE: NEC Mobile Gear II, Casio Cassiopeia FIVA y HP iPaq

A diferencia de los equipos Palm, los equipos que funcionan con Windows CE son muy variados en procesadores (Hitachi, NEC, MIPS, Motorola, ARM, Intel, entre otros) y fabricantes (Casio, HP, Compaq entre otros). Los primeros sistemas hasta la versión 2.0 de Windows CE eran monocromáticos. A partir de la versión 2.1 en adelante, soportan color.

Por último, hay que señalar que existe otro conjunto de equipos, los teléfonos celulares, también conocidos como SmartPhones. Algunos de estos trabajan con el sistema operativo Symbian. La compañía Symbian fue fundada en 1998 por Ericsson, Nokia, Motorola y Psion. Pronto otras compañías celulares comenzaron a utilizar este sistema operativo: Panasonic en 1999, Sony en el 2000, Fujitsu en el 2001, Siemens en el 2002, Samsung en el 2003 y Arima y LG Electronics en el 2004. Algunos modelos se aprecian en la figura 2.9 [SOP05].

Figura 2.9 Diversos modelos de teléfonos celulares basados en el sistema operativo Symbian

El sistema operativo Symbian está diseñado específicamente para estos dispositivos, que, de manera similar a las PDA's, administran la información personal del usuario, contienen directorio telefónico, agenda, notas, correo electrónico, y además, soporte para aplicaciones adicionales a las que se entregan con el sistema operativo. A diferencia de los equipos Palm o Windows CE, el tamaño de la pantalla es muy variado: desde 128x32 píxeles (en modo monocromático) hasta 640x200 píxeles (con 65,000 colores). Opcionalmente, pueden incluir: un teclado qwerty (normalmente vienen equipados con el teclado del teléfono), una pantalla sensible al tacto, doble pantalla, la posibilidad de conexión a una computadora de escritorio o ranuras de expansión de memoria.

2.2 Arquitecturas

Como se mencionó anteriormente, existen 3 plataformas principales de sistemas operativos para dispositivos portátiles: Palm OS, Windows CE y Symbian OS.

La diferencia entre estos sistemas es el origen de los mismos. Los sistemas Palm provienen de organizadores personales diseñados para ser sencillos, intuitivos y compactos. Las Pocket PC, son versiones limitadas de una PC, y por lo tanto, una computadora de propósito general. Los equipos basados en Symbian están orientados a las capacidades de comunicación de los teléfonos celulares, pero están al mismo tiempo abiertos al desarrollo de aplicaciones.

Con el sistema operativo Palm OS 4.1 se introdujeron nuevas capacidades, como sistema de archivos, conexión con teléfonos y manejo de color para imágenes y videos. Salvo estas adiciones, el concepto Palm se ha mantenido: deber ser usable, móvil y resolviendo una cantidad limitada de tareas. El núcleo del sistema fue desarrollado por Kadak Products Ltd [KPL05], quien vendió la licencia a Palm Computing. Una característica de este sistema operativo es la estricta adherencia a una cierta plataforma, tipo de procesador, tamaño de la memoria, pantalla, etc. Estas limitaciones de hardware, sin embargo, son las que hacen poderoso al sistema. Como fue desarrollado para una sola plataforma, las computadoras Palm son más rápidas en capacidad de procesamiento comparadas con las equivalentes con otros sistemas operativos.

Las primeras versiones de Palm OS se desarrollaron para procesadores Motorola de 16 MHz, con 128 KB de RAM y 512 KB de ROM. Todas las aplicaciones de software se proporcionan en la memoria ROM junto con el sistema operativo. Se pueden agregar aplicaciones adicionales a la memoria RAM y se ejecutan desde ahí. Por mucho tiempo las pantallas eran monocromáticas. Las versiones más recientes soportan 65,000 colores. La resolución de la pantalla es de 160 x 160.

Las más recientes versiones de Palm OS [PSD05], Garnet (Palm OS 5) y Cobalt (Palm OS 6), están diseñadas para los dispositivos equipados con procesadores ARM, que también es utilizado en las Pocket PC's. Para mantener la compatibilidad con las aplicaciones de versiones anteriores de sus equipos, Palm incluye PACE (Palm OS Application Compatibility Environment) en las nuevas versiones de su sistema operativo.

Windows CE es el sistema operativo desarrollado por Microsoft para los dispositivos portátiles Handheld PC y Pocket PC (inicialmente conocidas como dispositivos del tamaño de una Palm) [WCE05]. La diferencia principal entre estos dispositivos es el tamaño de la pantalla (480x240 para las Hanheld contra 320x240 para las Pocket PC). Las primeras versiones de este sistema operativo se asemejan a los sistemas operativos Windows 95 y Windows 98. La versión 4, también llamada Windows CE.net se asemeja en la interfaz a Windows XP.

Aunque similar a los sistemas operativos de Microsoft para PC, Windows CE no es una versión reducida de estos. Fue construido desde cero y tomando en cuenta las limitaciones de los equipos. Por esto, algunas funciones del sistema operativo Windows no existen en Windows CE. La portabilidad de las aplicaciones resultan limitadas sobre todo en prestaciones gráficas.

El equipo contiene el sistema operativo y un conjunto de programas. Entre estos destaca una suite de Office con aplicaciones compatibles a sus contrapartes en equipos de escritorio (Pocket Excel, Pocket Word, etc) residentes en memoria ROM. Se pueden agregar aplicaciones adicionales que se almacenan junto con los archivos de datos en memoria RAM. El equipo permite administrar la memoria de tal modo que se puede determinar cuánto se utilizará para el sistema de archivos y cuánto para la ejecución de los programas.

Symbian es el sistema operativo adoptado por muchos de los fabricantes de teléfonos celulares (SmartPhones). Funciona en una gran variedad de equipos [Me03]. La versión más reciente del sistema, Symbian OS v8.0, viene acompañado de mejoras gráficas. Trabaja con las cámaras digitales integradas a los equipos. Maneja diversas tecnologías de comunicación inalámbricas, tanto remotas como locales. El desarrollo de aplicaciones se realiza principalmente en Java y C++.

Respecto a las comunicaciones, soporta redes personales, locales y celulares . Está diseñado para los procesadores ARM y Xscale [SDH05]. En el tema multimedia, este sistema soporta los formatos de audio WAV, AU, RAW, PCM. Soporta codecs para otros formatos. Permite la reproducción y grabación de video, así como procesamiento en paralelo para procesar streams de datos, como en el formato MPEG. También tiene soporte para varios formatos de imagen por medio de plug-ins para JPEG, GIF, BMP, PNG, entre otros. En la parte gráfica, se pueden desarrollar aplicaciones que trabajen directamente con el GDI o con acceso directo a la pantalla. Es el primero en adoptar el estándar OpenGL ES para manejo de imágenes tridimensionales.

Las capacidades gráficas de estos equipos son muy variadas. Las pantallas de mayor capacidad se encuentran alrededor de 640x200 píxeles y 65,000 colores. Las aplicaciones incluidas junto con el sistema operativo residentes en ROM son básicamente, hoja de calculo, procesador de palabras, manejador de presentaciones, reproductor de video, navegador de Internet, agenda, directorio telefónico, administrador de archivos y algunos juegos.

2.3 Conectividad

Los primeros dispositivos eran prácticamente stand-alone, es decir, desconectados de otros equipos. En los mejores casos había forma de transferir datos por medios magnéticos o por memorias intercambiables. Los dispositivos Palm inicialmente sincronizaban datos entre el equipo portátil y una PC mediante un cable por puerto serial (RS-232C). Esta forma de conexión existe actualmente por medio de puertos seriales USB y se conoce como conexión de respaldo.

Poco tiempo después se implementaron las primeras comunicaciones inalámbricas por medio de puertos de luz infrarroja o IrDA. Esto permitió que dos equipos móviles pudieran intercambiar información sin depender de equipos fijos.

Actualmente los dispositivos PDA, además, pueden conectase mediante radiotransmisores a redes inalámbrica locales (WLAN) y redes locales personales (Bluetooth). Los SmartPhones tienen además la capacidad de conexión a las redes públicas celulares (GPRS/GSM). En el capítulo 3 se revisan estas tecnologías.

2.4 Herramientas de desarrollo

La siguiente tabla muestra las herramientas más representativas para el desarrollo de aplicaciones en dispositivos portátiles.

Nombre de la aplicación	Palm	Win CE	Symbian	Tipo de Lenguaje
Palm SDK	*			C++
Code Warrior	*		*	C++
CASL	*	*		Pascal
AppForge Crossfire	*	*	*	VB
Windows CE SDK		*		C++ / VB
eMbedded C++		*		C++
eMbedded Visual Basic		*		VB
.NET Compact Framework		*		C# / VB
Symbian SDK			*	C++ / Java
Borland C++ BuilderX Mobile Edition			*	C++

Palm SDK

Este es uno de los primeros ambientes de desarrollo para los equipos Palm [PSD05]. Este software fue desarrollado por Palm Computing y está orientado a los programadores de C y C++.

Code Warrior

Este también es uno de los ambientes más antiguos en el desarrollo de aplicaciones para PDA's . Es un compilador de C y C++, inicialmente solo para Palm [CW05]. Actualmente

también está disponible para equipos con sistema operativo Symbian que estén basados en procesadores ARM. Se puede instalar en equipos con sistema operativo MAC, Windows y Linux.

Compact Application Solution Language CASL

CASL (Compact Application Solution Language) es un lenguaje similar al Pascal [CS05]. La interfaz de desarrollo es del tipo RAD (Rapid Application Development) que permite crear rápidamente formas y controles visuales, acceso a bases de datos y comunicaciones. Se puede instalar en equipos con sistema operativo MAC y Windows. Se pueden crear aplicaciones tanto para Palm como para Windows CE.

AppForge Crossfire

Este producto se ha vuelto popular por ser totalmente compatible con Visual Basic 6 y Visual Studio NET [CFA05]. Permite crear aplicaciones para la mayoría de los equipos móviles basados en Palm OS y Windows CE. Además, permite el desarrollo de aplicaciones para equipos Nokia serie 60 basados en Symbian.

Windows CE SDK

Es el ambiente de desarrollo creado por Microsoft para los equipos con sistema operativo Windows CE [WAD05]. Permite la construcción de aplicaciones en Visual Basic y C++.

eMbedded Visual C++ y eMbedded Visual Basic

Son los compiladores de C++ y Visual Basic respectivamente que proporciona Microsoft para desarrollar aplicaciones del sistema operativo Windows CE. El ambiente de trabajo es similar a las versiones de compiladores para equipos de escritorio. Los compiladores pueden crear archivos ejecutables para los diversos procesadores que contienen los equipos Pocket PC y Handheld PC. Incluyen un emulador del sistema operativo Windows CE para realizar pruebas.

.NET Compact Framework

Es el ambiente de desarrollo diseñado por Microsoft tanto para equipos móviles (PDA's, Tablet PC's y teléfonos celulares) como equipos fijos. Para los equipos basados en Windows CE se pretende sustituir a los compiladores de C++ y Visual Basic en sus versiones "eMbedded" por compiladores de C# y Visual Basic.NET, pretendiendo que sean compiladores independientes de la arquitectura.

Symbian SDK

Este es el ambiente de desarrollo que provee la compañía Symbian para la construcción de aplicaciones [SDH05]. Permite el desarrollo en C++ que es el lenguaje nativo del sistema operativo. Tiene soporte para aplicaciones creadas en Java a partir la versión 5.

Borland C++ BuilderX Mobile Edition

Es el compilador de C++ desarrollado por Borland para los equipos basados en Symbian [BM05]. El ambiente de desarrollo RAD es similar a los ambientes de desarrollo de otros productos como Delphi. Contiene numerosos componentes visuales y no visuales precompilados que facilitan la construcción de las aplicaciones. Este compilador puede crear archivos ejecutables para un buen número de equipos celulares, incluye un emulador del sistema operativo Symbian para realizar pruebas.

Capítulo 3
Comunicación Inalámbrica y Aplicaciones para Cómputo Móvil

Las recientes mejoras en las capacidades gráficas en los equipos PDA, de la capacidad de los procesadores y de las distintas posibilidades para conectarlos tanto con equipos fijos como con otros dispositivos móviles, han permitido el desarrollo de aplicaciones más allá de la agenda personal que caracterizaba estos dispositivos. Los reproductores de video y los juegos son las primeras aplicaciones en utilizar las mejoras tecnológicas, aunque no las únicas que pueden aprovechar la movilidad del dispositivo.

3.1 Tecnologías de comunicaciones inalámbricas

El inicio de los servicios inalámbricos nos lleva al siglo XIX, cuando Guillermo Marconi comenzó a experimentar con ondas de radio (ondas Hertzianas) [EH00]. En 1896 tuvo éxito en producir y detectar ondas de radio a grandes distancias. En 1901 las señales se pudieron enviar a través del Atlántico y en 1905 se transmitieron los primeros mensajes en código Morse.

La tecnología inalámbrica eventualmente progresó como una herramienta útil para los militares. Durante la Segunda Guerra Mundial, el ejército de los Estados Unidos usó por primera vez las señales de radio para enviar datos. Los militares configuraron señales inalámbricas para transmitir los mensajes en forma encriptada, lo que hacía un acceso sin autorización casi imposible. Esta tecnología fue utilizada extensivamente por los Estados Unidos y sus aliados.

Wi-Fi

Un verdadero avance en la tecnología inalámbrica se dio con la implementación de las WLAN y su posterior uso en los dispositivos móviles. La primera WLAN o red inalámbrica de área local, llamada ALOHANET, fue implementada en 1971 por la Universidad de Hawai para conectar siete equipos de cómputo que se comunicaban mediante una topología

de estrella bidireccional, que abarcaba cuatro islas Hawaianas, con una computadora central ubicada en la isla de Oahu.

Para poder utilizar ampliamente las redes WLAN se requirió un estándar que el Institute of Electrical and Electronics Engineers (IEEE) definió en 1997, llamado IEEE 802.11 [sMC04]. En 1999 aparecieron el IEEE 802.11a y el IEEE 802.11b, seguidos en el 2003 por el IEEE 802.11g y en el 2004 por el IEEE 802.11i.

El estándar original opera en la banda de radiofrecuencia (RF) cercana a los 2.4 GHz. Provee tasas de transferencia de datos de 1 Mbps a 2 Mbps.

El estándar IEEE 802.11a opera en la banda de 5.8 GHz, con tasas de transferencia de datos de 5 Mbps y 54 Mbps. El estándar IEEE 802.11b opera en la banda de 2.4 GHz, con una tasa de transferencia de datos de 11 Mbps. El estándar IEEE 802.11b es conocido como Wi-Fi (Wireless Fidelity).

El estándar IEEE 802.11g combina características de los estándares IEEE 802.11a y IEEE 802.11b. Opera en la banda de 2.4 Ghz al igual que el IEEE 802.11b, por lo que los dispositivos del estándar 802.11g pueden conectarse con equipos 802.11b a una tasa de transferencia de datos de 11 Mbps. Del IEEE 802.11a utiliza el método OFDM (Orthogonal Frequency Division Multiplexing) para obtener tasas de transferencia de datos de 54 Mbps.

El estándar IEEE 802.11i, define nuevos métodos mejorados de encripción de datos tanto para las redes 802.11a como para las 802.11b (Incluye Wi-Fi y 802.11g).

Existen otras versiones del estándar, normalmente adecuaciones:

- El estándar IEEE 802.11d es similar al 802.11b. Es para los países donde no se puede utilizar la banda de los 2.4 Ghz.
- El IEEE 802.11e define reglas de prioridad para la transmisión, considerando si ésta es de datos, voz o video.
- El estándar IEEE 802.11j agrega extensiones al estándar 802.11a para cumplir con disposiciones legales del gobierno japonés, así como para utilizar la banda de los 4.9 Ghz.
- El estándar IEEE 802.11k propone reglas para el uso de los canales de radiofrecuencia.

Los dispositivos PDA fueron, inicialmente, provistos de radio transmisores IEEE 802.11 por medio de tarjetas que se conectaban en puertos de expansión. Actualmente estos transmisores forman parte integral del hardware de los equipos más recientes.

Seguridad Inalámbrica

Las redes inalámbricas presentan problemas de seguridad que no existen en las redes de cables como Ethernet. Esencialmente se necesita evitar la conexión de usuarios no autorizados y mantener la privacidad de la información.

En la capa física existen dos tipos de esquemas de modulación para lograr una radiocomunicación segura: "Direct Sequence Spread Spectrum" (DSSS) y "Frequency Hopping Spread Spectrum" (FHSS) [WLA99] [We04]. Ambos fueron diseñados por los militares y se les considera confiables, seguros e íntegros.

FHSS trabaja separando la banda de frecuencia disponible en varios canales. La frecuencia de la transmisión cambia de canal de una manera pseudo aleatoria que es conocida por el emisor y el receptor. Esto se logra mediante "Gaussian Frequency Shift Keying" (GFSK), (ver figura 3.1). La ventaja de FHSS es que permite la coexistencia de múltiples redes sobre el mismo espacio físico.

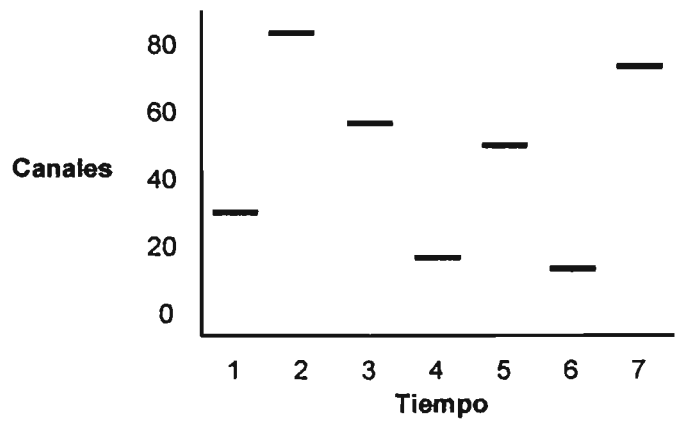


Figura 3.1 FHSS

DSSS funciona de una manera diferente. En este esquema se combina el flujo de datos con un código digital de alta velocidad. Cada bit de datos es mapeado a un patrón de bits conocido sólo por el emisor y el receptor. Este patrón es conocido como "chipping code". Este código es una secuencia aleatoria de señales altas y bajas que significan el bit actual. El "chipping code" es invertido para representar el bit opuesto de la secuencia de datos (ver figura 3.2).

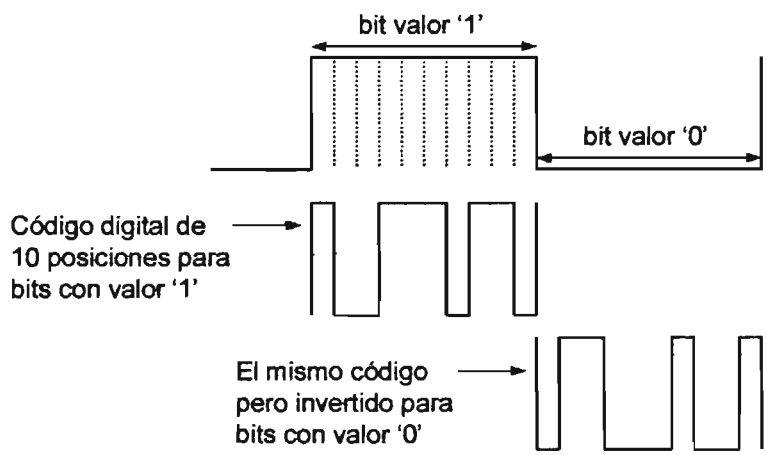


Figura 3.2 DSSS

En la capa de datos, existen tres métodos de autentificación y cifrado de información [MT03]: el primero es el sistema abierto; el segundo es de llave compartida; y un tercero que funciona tanto con llave compartida como con identificación individual:

- En el sistema abierto cualquier equipo puede conectarse a la red sin necesidad de contraseñas de acceso. No hay encriptado de datos.
- WEP (Wired Equivalent Privacy). En este método todos los equipos se configuran con una llave compartida, la cual consiste en un conjunto de números hexadecimales. Las llaves pueden ser de 64 y 128 bits de longitud. El encriptado de datos es opcional y se realiza mediante ICV (Integrity Check Value). Este valor se encripta con en el propio método WEP.
- WPA (Wi-Fi Protected Access). En este método se puede utilizar un servidor denominado RADIUS para autentificar cada equipo que se conecta a la red o utilizar una llave compartida PSK (Pre Shared Key) de manera similar al método WEP. A diferencia de este, la llave se constituye con una cadena de caracteres de longitud variable. El encriptado de datos es obligatorio y se realiza mediante TKIP (Temporal Key Integrity Protocol) cuyo algoritmo provee mayor seguridad que el método WEP.

Redes Celulares

Las redes Wi-Fi están limitadas a que los dispositivos estén relativamente cerca unos de otros, en promedio unos 300 metros. Para los casos en que no hay conexión de red Wi-Fi o se está lejos de la red local, se pueden utilizar las redes de teléfonos celulares. Estas redes ofrecen básicamente servicio de voz, transmisión de datos y acceso a Internet.

La primera generación de redes celulares fue analógica. El desarrollo de la tecnología digital para la segunda generación permitió el desarrollo de sistemas más eficientes que las reemplazaron. Los sistemas celulares TDMA y CDMA, del mercado norteamericano, ofrecen tasas de transferencia de datos de 9.6 kbps y 14.4 kbps respectivamente. El estándar europeo GSM (Global System for Mobile Communications) ofrece tasas de 14.4 kbps en las frecuencias de 900 MHz y 1800 MHz. En Japón el sistema NTT-DoCoMo ofrece tasas de 9.6 kbps.

Los sistemas celulares de segunda generación mejorados para alcanzar mayores tasas de datos se les denomina como Sistemas 2.5G.

En Europa hay tres variaciones 2.5G del sistema GSM: el sistema GPRS (General Packet Radio System) que provee tasas de hasta 100 kbps y es el que muchas redes europeas han implementado, así como los sistemas HSCSD (High Speed Circuit Switched Data) con una tasa de 230.4 kbps y EDGE (Enhanced Data rates for Global Evolution) con una tasa de 384 kbps, menos utilizados por su alto costo y consumo de recursos.

En Norteamérica los sistemas 2.5G siguen una evolución similar a los europeos, aunque un poco menos avanzados respecto al sistema GSM. Para TDMA las evoluciones son: TDMA+ con una tasa de 43.2 kbps y TDMA EDGE, análogo a la versión GSM, con una

tasa de 384 kbps. Para CDMA las actualizaciones son: CdmaOne95B con una tasa de 64 kbps y Cdma2000 con tasa de 144 kbps.

La tercera generación de sistemas celulares (3G) responde a la necesidad de servicios de datos de alta velocidad para tecnologías inalámbricas. Actualmente coexisten tres tecnologías de tercera generación: UMTS (Europa), Cdma2000 (Estados Unidos) y WCDMA (Japón). Operan en las bandas de 1900-1980, 2010-2025 y 2110-2170 MHz respectivamente. Se espera que ofrezcan tasas de hasta 2 Mbps cerca de la estación base y de 384 kbps en zonas urbanas.

Bluetooth

En 1995 Ericsson lanzó Bluetooth, un estándar para permitir la comunicación entre dispositivos a corta distancia en forma inalámbrica. En 1998 el estándar fue introducido al mercado por un consorcio de desarrolladores de telecomunicaciones, formado por Ericsson, Nokia, IBM, Intel y Toshiba. Bluetooth es la respuesta para tecnologías infrarrojas, las cuales son incapaces de penetrar obstáculos. La banda de operación está en los 2.4 GHz, la cual está dentro de los espectros libres de licencia, así estos dispositivos pueden ser utilizados libremente en muchos países. Esta es la misma banda que es utilizada por los dispositivos IEEE 802.11, resultando en una incompatibilidad entre los dos sistemas. Inicialmente Bluetooth ofrecía tasas de transferencia de 722 kbps con cobertura de 10 metros. Actualmente ofrece 1 Mbps de transferencia y cobertura de 50 metros. Se espera que la tasa se incremente a 10 Mbps y la cobertura a 100 metros en los próximos años.

La tecnología Bluetooth tiene la ventaja de estar siempre conectada y no requiere la intervención directa del usuario para establecer la conexión. Está implementada, gracias a su bajo consumo de energía, en diversos dispositivos. Entre éstos, los teléfonos celulares, permitiendo compartir la conexión a redes públicas celulares con otros dispositivos como PDA's o laptops.

La conveniencia del uso de Bluetooth en los teléfonos celulares para compartir sus conexiones a redes públicas con dispositivos PDA, en conjunto con las mejoras tecnológicas en el tamaño de los aparatos telefónicos, ha evolucionado de tal forma, que el teléfono celular es considerado parte integral del hardware de algunos dispositivos portátiles de reciente manufactura.

ZigBee

ZigBee es un nuevo estándar para construir redes locales, básicamente para ser usado de manera similar a sensores entre dispositivos [Pw04], transmitiendo datos de manera periódica y respondiendo al estímulo de otros transmisores. Al igual que Wi-Fi y Bluetooth opera en la banda de los 2.4Ghz pero consume menos energía, reduciendo el alcance (10 metros) y la velocidad de transmisión de datos (250 kbps). El estándar IEEE 802.15.4 define esta tecnología. ZigBee fue conocido inicialmente como PURLnet, RF-Lite, FireFly y Home RF Lite.

3.2 Tecnologías para generación de imágenes gráficas en PDA's

Los primeros equipos portátiles tenían capacidades gráficas limitadas, y equipados con una pantalla en blanco y negro. Con distintas resoluciones dependiendo de la arquitectura. No hubo mejoras más allá de las funciones básicas para dibujar objetos 2D por medio del GDI (Graphics Device Interface) hasta que aparecieron las primeras pantallas a color y posteriormente con la aparición de las primeras cámaras de video digital.

DirectX

Las primer mejora de las capacidades gráficas en equipos basados en Windows CE se dio con la implementación de su librería gráfica DirectX en la versión 3 del sistema operativo. Las escasas funciones disponibles en esta versión básicamente dan acceso directo a la pantalla, por medio de DirectDraw. Esto para lograr un aumento el número de cuadros por segundo que se pueden dibujar, orientado a aplicaciones como reproductores de video y juegos.

A partir de la versión 4, se implementan las primeras funciones de la librería Direct3D para el modelado de objetos tridimensionales. En la figura 3.3 se muestra la integración de la arquitectura gráfica DirectX en la versión 5 del sistema operativo Windows CE.

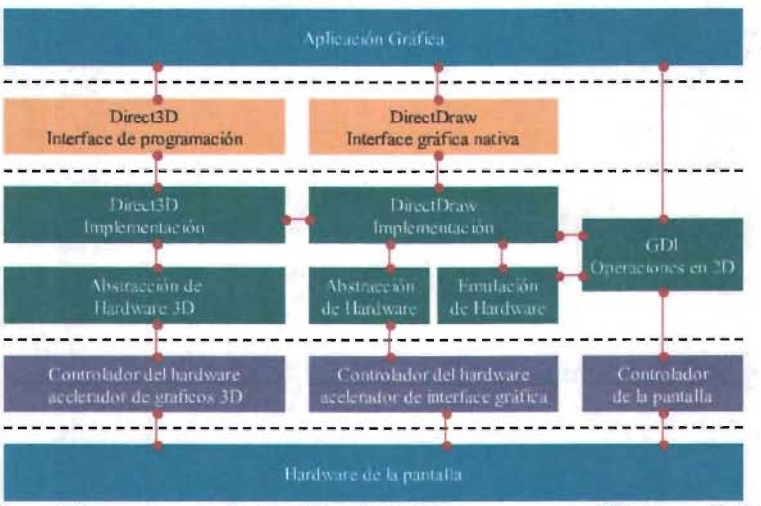


Figura 3.3 Arquitectura Grafica DirectX del sistema operativo Windows CE 5.0

OpenGL ES

OpenGL (Open Graphics Library), una librería grafica desarrollada inicialmente por Silicon Graphics, es ahora uno de los estándares más ámpliamente utilizados en el desarrollo de aplicaciones de aceleración gráfica tridimensional. Está disponible en diversas arquitecturas de cómputo. El estándar para ambientes móviles se denomina OpenGL ES (Open Graphics Library for Embbeded Systems).

En la arquitectura de Windows CE, al menos hasta la versión 5, no existe una implementación de OpenGL respaldada por los fabricantes de hardware y del sistema operativo. Existe una implementación independiente de OpenGL para Pocket PC, que se conoce como PocketGL. Esta librería fue creada por Pierre Leroy [Le02], y fue construida en C++ antes de la aparición del estándar OpenGL ES. Soporta un subconjunto de OpenGL, por lo que ofrece algunas funciones de manejo de texturas y polígonos suficientes para el desarrollo de imágenes de calidad aceptable considerando la capacidad del hardware del equipo. Funciona en equipos basados en procesadores ARM y MIPS. Esta librería es ámpliamente utilizada gracias a la portabilidad de aplicaciones desarrolladas para otras arquitecturas. En la figura 3.4 se muestra una aplicación que utiliza esta arquitectura.

Fig. 3.4 Aplicación basada en PocketGL. La imagen de la izquierda es el modelo para la aplicación. La imagen de la derecha es la aplicación funcionando en el dispositivo

En los equipos basados en Symbian, al igual que en las Pocket PC's, las funciones gráficas se limitaban a funciones de operaciones de 2D por medio del GDI y a funciones de acceso directo a la pantalla mediante EGL (Embbeded Graphics Library). A partir de la versión 8.0 de Symbian [SS04], se ha adoptado OpenGL ES como la solución de hardware-software para aceleración gráfica tridimensional. La figura 3.5 muestra la integración de la arquitectura gráfica OpenGL en la versión 8.0 de este sistema operativo.

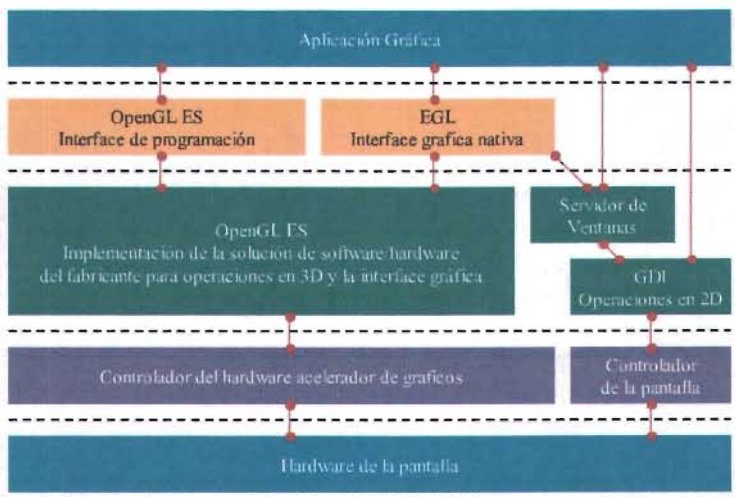


Figura 3.5 Arquitectura gráfica OpenGL ES del sistema operativo Symbian 8.0

3.3 Trabajos en arquitectura gráfica

Aunque los equipos portátiles actuales contienen soluciones de software-hardware que permiten el modelado de imágenes tridimensionales, las aplicaciones gráficas de alto realismo o de visualización de grandes y complejos modelos, requieren hardware especializado que aún no está disponible. Este tipo de aplicaciones son importantes para diversas disciplinas como: medicina, simulación, CAD y realidad virtual, entre otras.

La solución más comúnmente adoptada ha sido distribuir el trabajo de cómputo entre un sistema servidor con hardware especializado que procesa las imágenes y clientes que las visualizan. Está no es una idea nueva. Actualmente existen sistemas que permiten la visualización remota de modelos gráficos para estaciones de trabajo que no tienen suficiente capacidad de hardware para procesarlos. El proyecto Chromium es uno de estos sistemas. En dispositivos móviles ya existen algunas aplicaciones de visualización gráfica remota basadas en arquitectura cliente-servidor.

Chromium

Chromium [Chr02], un proyecto de la Universidad de Stanford, se diseñó como alternativa a los servidores gráficos con múltiples procesadores que trabajan en paralelo. Consiste en la construcción de clusters de equipos de menor capacidad donde se distribuye el trabajo de construir la imagen. Inicialmente conocido como Proyecto WireGL, es una librería que tiene la misma interfaz que OpenGL, pero los comandos no son ejecutados en el equipo donde se encuentra la aplicación, sino que éstos son enviados por red a los equipos servidores que calculan la imagen en paralelo.

La arquitectura del sistema Chromium es cliente-servidor. Los nodos clientes generan paquetes de comandos OpenGL, y los servidores manejan las conexiones e interpretan los mensajes que son ejecutados en el hardware local. Los fragmentos de las imágenes calculadas son enviados a un servidor que se encarga de formar la imagen final que será enviada al cliente.

Existen al menos tres formas de distribuir el cálculo de la imagen:

- Por regiones: la imagen es dividida en tantas regiones como la cantidad de procesadores disponibles, normalmente en múltiplos de 2. Este es el método más utilizado.
- Por objetos: la base de datos del modelo tridimensional es dividido en tantas partes como la cantidad de procesadores disponibles.
- Por colores: cada servidor calcula un componente de color. Para imágenes RGB se utilizan 3 servidores y para imágenes RGBA se necesitan 4 servidores.

Arquitectura Grafica Remota Acelerada para PDA's

Desarrollado en el Politécnico de Torino, el proyecto An Accelerated Remote Graphics Arquitecture for PDAs (ARGAP) extiende las capacidades del proyecto Chromium hacia

un dispositivo portátil [LZSFM03]. Consiste en crear un puente entre el nodo cliente del cluster que calcula la imagen y la PDA, permitiendo tener acceso a imágenes tridimensionales de alto realismo en dispositivos portátiles. En la arquitectura propuesta, el dispositivo portátil es una interfaz de visualización de un sistema donde los recursos de hardware están centralizados en el cluster y son los encargados de calcular la imagen.

La arquitectura del proyecto ARGAP es cliente-servidor. Se establece una conexión TCP inalámbrica en donde el cliente envía paquetes de datos que representan eventos de navegación en la PDA. El servidor recibe los paquetes de eventos y los transforma en comandos de OpenGL. Estos comandos son enviados al cluster de procesadores que calculan la imagen. La imagen resultante es devuelta al servidor, el cual la envía en un paquete de datos al cliente. La arquitectura genérica cliente-servidor propuesta en este proyecto es capaz de controlar cualquier modelo basado en OpenGL a distancia y puede ser construida tanto en equipos portátiles como de escritorio. En la figura 3.6 se muestra una prueba de uso de la aplicación.

Figura 3.6 Visualización de un modelo molecular en la aplicación del proyecto ARGAP

Realidad Aumentada Móvil

Realidad Aumentada es una herramienta que modifica la percepción del ambiente al usuario, agregando objetos generados por computadora a la escena que está viendo. Los objetos pueden ser calculados a partir de la sincronización de un modelo virtual con el ambiente real o por medio del reconocimiento de objetos directamente en la imagen. La combinación de la información puede ser utilizada en aplicaciones tales como guías, manuales, etc.

La aparición de dispositivos móviles (SmartPhones y PDA's) con cámaras de video digital, ya sea que se puedan conectar al equipo o estén integradas al mismo, permite el desarrollo de aplicaciones de realidad aumentada aprovechando la posibilidad de tener la cámara de captura de video y la interfaz del usuario en un mismo dispositivo programable.

Figura 3.7 Prueba de uso del prototipo del proyecto "A Touring Machine"

Mucho del trabajo en Realidad Aumentada Móvil se ha desarrollado desde 1997 en la Universidad de Columbia mediante el proyecto "A Touring Machine" [FMHW97], un prototipo de un sistema de realidad aumentada móvil para ambientes urbanos. En este proyecto se utiliza un dispositivo portátil como interfaz para el usuario, y una

computadora portátil que se lleva a la espalda, la cual calcula la imagen de video de los objetos. Esta imagen se despliega en unos anteojos cuya pantalla transparente permite ver al mismo tiempo el mundo real. En la figura 3.7 se muestra el prototipo. La conexión inalámbrica entre los equipos y las radio bases en el campus se realiza mediante radio modems NCR Wave LAN de 2 Mbps.

Este proyecto utiliza sensores para sincronizar el ambiente real con un modelo del virtual. La ubicación del usuario se obtiene mediante un sistema GPS. La orientación se obtiene mediante un magnetómetro para determinar la orientación de la cabeza respecto al campo magnético de la tierra y un inclinómetro de dos ejes que utiliza la gravedad para determinar el ángulo de inclinación respecto a la horizontal. La energía se obtiene de un cinturón de baterías recargables de Niquel-Cadmio.

También se han realizado trabajos de investigación donde se utilizan técnicas de reconocimiento de objetos. Ejemplo de esto, es el trabajo realizado por Daniel Wagner y Dieter Schmalstieg de la Universidad de Tecnología de Viena [WS03]. Evaluaron el desarrollo de aplicaciones de realidad aumentada para dispositivos portátiles, en particular PDA's, utilizando el hardware que se encuentra disponible comercialmente. En su proyecto migraron uno de los módulos que forman parte de la herramienta ARToolKit (Augmented Reality Toolkit) hacia una Pocket PC modelo Compaq iPaq basada en Windows CE 4.0 y equipada con una cámara digital Compact Flash. En la figura 3.8 se muestra una prueba de la implementación en un prototipo.

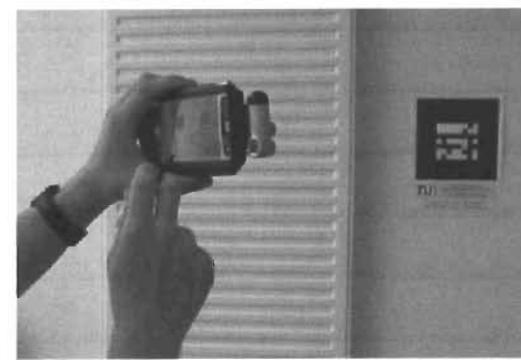


Figura 3.8 Prueba de uso de la implementación prototipo del tracking de ARToolkit en una PDA

El módulo de ARToolKit que fue migrado realiza la tarea del tracking de marcadores, consistente en leer las imágenes de la cámara de video y ubicar las posiciones donde se agregarán los objetos virtuales a la escena. En las pruebas se observó que es factible leer pocos cuadros, unos 7 u 8 cuadros por segundo. Por esta causa, el trabajo de investigación concluyó que el resto del proceso, el modelado de los objetos y la composición de la escena, deben ser realizados por un equipo servidor.

Figura 3.9 Prueba de uso de PocketHouseAR, la cámara esta montada sobre el HDM. En el monitor se observa lo que aparece en el HDM y en la PDA

PocketHouseAR, una herramienta diseñada por Gerald Binder de la Universidad de Ciencias Aplicadas de Austria [Bi04], considera el equipo móvil sólo como medio de navegación portátil del usuario y no como interfaz de visualización. En este proyecto se utiliza la técnica de reconocimiento de

objetos. La cámara se considera como un dispositivo independiente conectado directamente al servidor, que calcula la escena. La imagen calculada es enviada a un tercer dispositivo para su visualización, que puede ser un monitor, un proyector o un dispositivo para la cabeza conocido como HDM (Head Mounted Display). En la figura 3.9 se muestra una prueba del prototipo del proyecto.

Cabe mencionar que aunque los teléfonos celulares o SmartPhones tienen la suficiente capacidad de hardware y software para realizar tareas de Realidad Aumentada, aún no es factible realizar una implementación que funcione dentro de parámetros aceptables [HO03]. Esto debido a que la conectividad de la mayoría de estos equipos por ahora está basada en redes GSM/GPRS, cuyas tasas de transferencia de datos aún son bajas comparadas con las redes Wi-Fi. Este es el factor más importante para la calidad de la aplicación, pues de esta variable depende la cantidad de cuadros de video que se pueden transferir en una arquitectura cliente-servidor.

PVHA Asistente Personal Humano Virtual

Este proyecto desarrollado en Suiza [GVT03] plantea un cambio en la interfaz de los equipos portátiles, actualmente basada en ventanas e iconos, que viene heredada de los equipos de escritorio. En este proyecto se plantea que la interfaz apropiada para un asistente personal debería perecerse más a un ser humano dentro de la computadora. Para esto se plantea que es necesario que la interfaz incluya la capacidad de realizar análisis y síntesis tanto de audio como de video en combinación con técnicas de inteligencia artificial, así como un apropiado manejo de diálogos y gestos.

El objetivo del proyecto fue sintetizar un modelo tridimensional de un ser humano en una PDA, como se muestra en la figura 3.10. La arquitectura de este proyecto es cliente-servidor en tres componentes: la interfaz en el dispositivo móvil, un servidor de mensajes intermediario y un servidor que contiene el ambiente virtual.

Figura 3.10 Asistente Personal Humano Virtual en una PDA iPaq

A diferencia del proyecto ARGAP, donde todo el cálculo de la imagen se realiza en el servidor, en este proyecto se utilizan las capacidades gráficas del cliente para dibujar el modelo tridimensional. El manejo de la cámara y la manipulación de los objetos de la escena se hace directamente en el equipo portátil. Para esto, el servidor envía a la PDA los datos de la escena en formato MPEG-4.

El formato MPEG-4 permite enviar la descripción del modelo, los parámetros de deformación y los parámetros de animación. La información del modelo está en términos de la jerarquía de objetos, de las uniones entre éstos y de segmentos (partes del

cuerpo como cabeza, piernas, etc.). Los parámetros de deformación son tablas que contienen la información necesaria para producir las deformaciones anatómicas en las uniones de los objetos para aumentar el realismo. Los parámetros para realizar la animación se envían en términos de ángulos de rotación para las uniones.

Capítulo 4
Desarrollo de la aplicación

En este capítulo se analiza y construye una solución genérica que permita visualizar y controlar imágenes calculadas remotamente en otro equipo. El cálculo de la imagen se realiza en un equipo de escritorio y la imagen es enviada a la PDA mediante una conexión de red. En el lado de la PDA, el usuario puede explorar la escena. Aunque existen diversos medios para el desarrollo de aplicaciones gráficas para dispositivos portátiles, la capacidad de generar y controlar imágenes tridimensionales de alta calidad aún está lejos de las capacidades de estos dispositivos.

4.1 Arquitectura

La problemática para una aplicación gráfica local en equipos portátiles radica en que no es posible manipular modelos de imágenes tridimensionales complejos o de gran cantidad de datos dada la limitada capacidad del PDA.

Por lo tanto, la propuesta de este trabajo es construir un sistema de software basado en el paradigma cliente-servidor, donde el dispositivo portátil (el cliente) cumpla la función de interfaz final al usuario, dejando los datos del modelo y el cálculo de la imagen a otro dispositivo (el servidor) con mejores prestaciones de hardware.

Las ventajas de esta arquitectura son:

- Seguridad en la información. La base de datos necesaria para construir la imagen no se transmite a los clientes.
- Colaboración. Varios clientes pueden trabajar en la misma imagen.

Las desventajas:

- Posibles problemas de comunicación entre los dispositivos, tanto la conexión física como la capacidad de transmisión de datos.
- Las restricciones de cada dispositivo para la implementación de la aplicación, es decir, cada arquitectura requiere una adaptación del software.

Para esta arquitectura se requiere que las aplicaciones a construir cumplan con las siguientes características mínimas:

Para la aplicación del servidor
- Conexión de red.
- Recursos para construir y manipular la imagen.

Para la aplicación del cliente
- Conexión de red.
- Medio para visualizar la imagen en el dispositivo.
- Medio para recibir instrucciones del usuario.

La arquitectura genérica de la propuesta se muestra en la figura 4.1. La aplicación servidor, implementada para un equipo de escritorio, calcula la escena. Al mismo tiempo, una aplicación cliente, construida para un dispositivo portátil, permite la visualización y el control de la imagen.

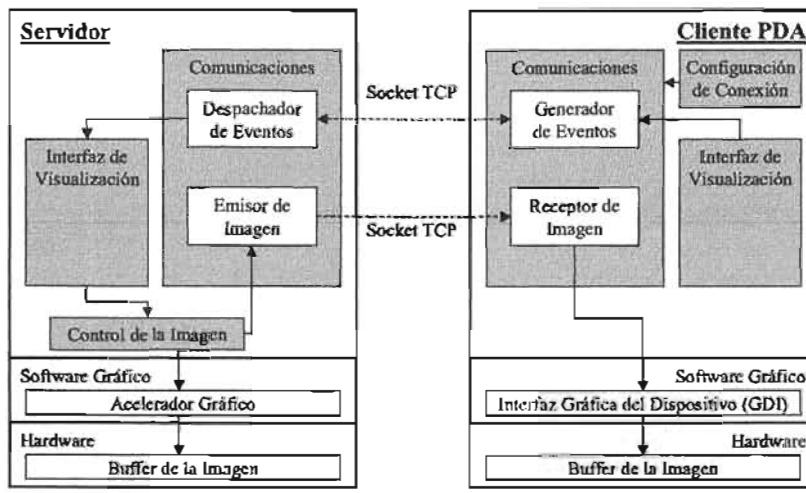


Figura 4.1 Arquitectura cliente-servidor propuesta

En esta arquitectura un usuario en el servidor también puede modificar la escena por medio de la interfaz de visualización. Además, es independiente del software acelerador de gráficos empleado para calcular la escena. En particular podría controlar imágenes creadas por otros métodos.

Aplicación Cliente

El módulo de control de las comunicaciones establece dos conexiones TCP inalámbricas con el servidor. La primera conexión se utiliza para enviar los eventos generados por el módulo de la interfaz de visualización y recibir datos de la escena desde el servidor. La segunda conexión se utiliza sólo para recibir los datos de la imagen.

La interfaz de visualización controla la interacción del usuario, muestra la imagen que se recibe del servidor y supervisa los eventos generados por el uso de la pluma o los botones del equipo. La interfaz contiene varios controles alrededor de la imagen que permiten al usuario manejar remotamente las funcionalidades del servidor, así como conocer algunas variables de la escena.

Aplicación Servidor

El módulo de comunicaciones controla la conexión así como la recepción y transmisión de paquetes de datos desde y hacia el cliente. Despacha los eventos recibidos a la interfaz de visualización y transmite la imagen al cliente.

La interfaz de visualización traduce los eventos que se reciben del cliente y los aplica a la escena, proporciona una interfaz donde el operador del servidor puede visualizar la escena y tiene controles para modificarla. Este módulo contiene el acceso a los datos que se van a transmitir al cliente.

El modelado de la imagen tridimensional está dividido en dos partes: la primera denominada Control de la Imagen es la encargada de crear la escena y de manipularla, también contiene el acceso al bitmap de la escena que se enviará al cliente; la segunda parte es el Software Acelerador de Gráficos. En esta implementación se utilizó OpenGL, el cual da el acceso al hardware del equipo.

4.2 Características de los equipos utilizados

El dispositivo donde se desarrolló la aplicación cliente es una HP iPAQ rx3100 PocketPC, equipada con el sistema operativo Windows Mobile 2003 Second Edition (Windows CE versión 4). El procesador es Samsung S3C2440 de 400Mhz compatible con procesadores ARM y Xscale. La pantalla es TFT de16 bits (65536 colores) con resolución de 240x320 píxeles (3.5 pulgadas), soporta visualización vertical y horizontal. El dispositivo está equipado con 56MB de memoria RAM y se pueden agregar tarjetas de memoria SD para aumentar su capacidad de almacenamiento.

La interacción del usuario se realiza tocando la pantalla con una pluma, por medio de un teclado virtual cuya distribución es similar a la que se encuentra en los de equipos de escritorio. Además, cuenta con botones de dirección y cuatro botones programables.

Respecto a las capacidades de comunicación, incluye un radiotransmisor Wi-Fi integrado, compatible con el estándar IEEE 802.11g/ 802.11b para redes WLAN; un radiotransmisor Bluetooth integrado, además de un puerto infrarrojo. El equipo es compatible con los estándares de seguridad WEP y WPA-PKS/TKIP para redes WLAN.

La aplicación del servidor se desarrolló para un equipo PC de escritorio, basado en sistema operativo Windows. El equipo de prueba cuenta con sistema operativo Windows 2000, procesador Pentium III de 700Mhz con 128MB de memoria RAM y equipado con una tarjeta aceleradora de gráficos ATI Radeon 9200 SE AGP de 400MHz y 64MB de memoria RAM.

Para conectar los equipos se utilizó un router inalámbrico marca Belkin con capacidad de hasta 4 puertos Ethernet para conectar equipos fijos y 100 puertos inalámbricos Wi-Fi compatibles con los estándares IEEE 802.11b y IEEE 802.11g con capacidad de transferencia de 11 Mbps para 802.11b y 54 Mbps para 802.11b, alcance de 500 metros en espacios abiertos, con soporte para protocolo TCP/IP. En el aspecto de seguridad incluye un firewall y soporta los estándares de seguridad WEP con llaves de 64 y 128 bits así como WPA-PSK con técnica de encripción de datos TKIP. Una fotografía del equipo se muestra en la figura 4.2.

Figura 4.2 Router Inalámbrico

4.3 Herramientas de desarrollo

Para la aplicación cliente se utilizó el compilador eMbedded Visual C++ 4.0 (EVC), actualizado con el Service Pack (SP) 2 y el Standard Software Development Kit (SDK) para Windows CE.net 4.2 como ambiente de desarrollo, ambos de Microsoft. Se escogió esta plataforma nativa de desarrollo para aprovechar las capacidades del equipo y por las facilidades del ambiente de desarrollo. En la figura 4.3 se muestra el ambiente típico de desarrollo de esta herramienta.

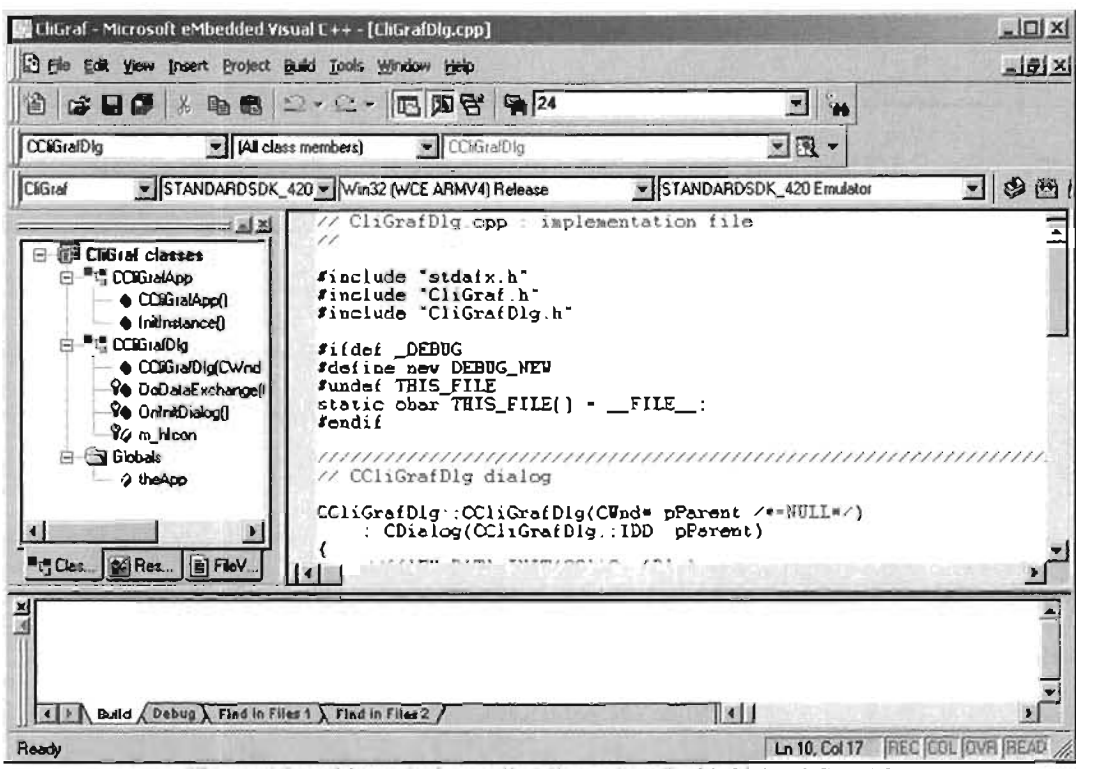


Figura 4.3 Ambiente de desarrollo Microsoft eMbedded Visual C++ 4.0

El servidor se construyó utilizando el compilador Borland C++ Builder 5.0 (BCB) para desarrollar aplicaciones basadas en el sistema operativo Windows de 32 bits. Se escogió esta plataforma para aprovechar mejor los recursos del equipo y por la facilidad para construir la aplicación al tener un ambiente RAD muy eficiente. En la figura 4.4 se muestra el ambiente típico de desarrollo de esta herramienta.

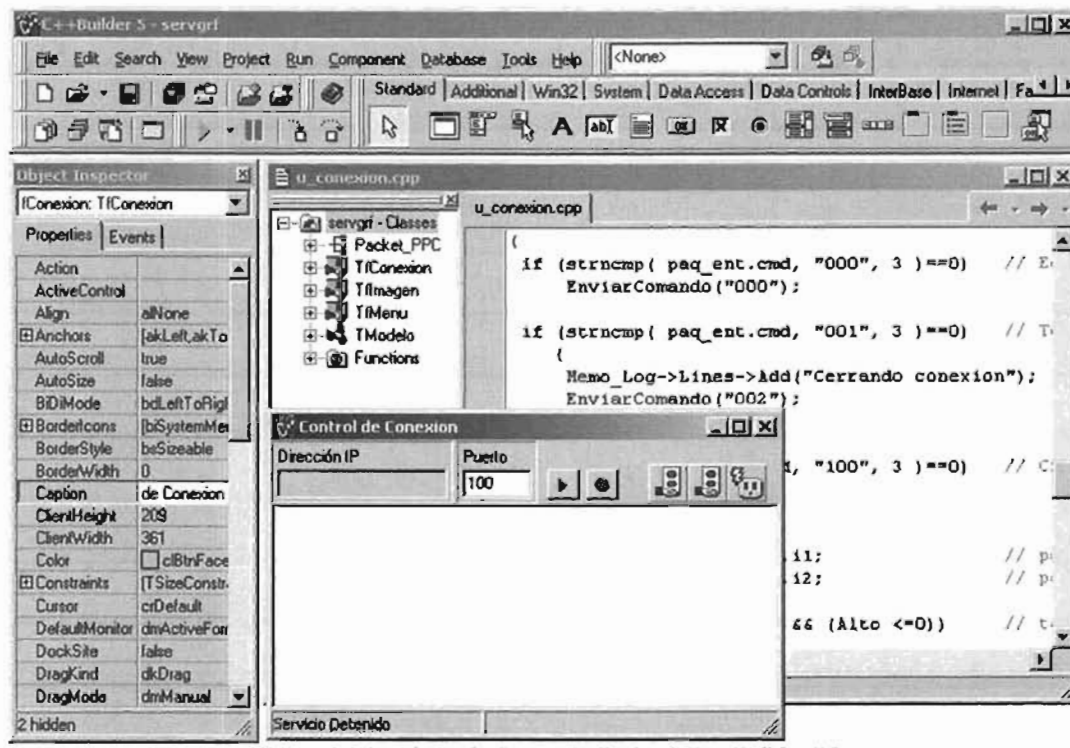


Figura 4.4 Ambiente de desarrollo Borland C++ Builder 5.0

4.4 Requerimientos

Se consideran cinco casos de uso en los requerimientos funcionales de la aplicación. Los diagramas de los casos de uso se encuentran en el anexo A. Para la aplicación servidor, dos casos: Iniciar la interfaz e Iniciar Servicio. Para la aplicación cliente, tres casos: Configuración de la conexión, Iniciar la aplicación y Navegar en la escena.

Servidor: Iniciar la Interfaz de Visualización

Al crearse y mostrarse la ventana de la interfaz de visualización de la imagen debe inicializarse también la librería gráfica. Los siguiente pasos son: la creación de los objetos de la escena, calcular la imagen y mostrarla en la ventana.

Servidor: Iniciar Servicio de Comunicaciones

El usuario podrá modificar el valor del puerto que el servidor utilizará para trasmitir los datos al cliente antes de poner en marcha el servicio de transmisión de imágenes.

Cliente: Configuración de la Conexión

Considerando que la información necesaria para lograr la conexión con el servidor no se modifica frecuentemente y que el espacio en la interfaz del equipo portátil es reducida, se plantea que estos datos sean capturados y registrados en el sistema mediante una aplicación de configuración y que la aplicación principal tome estos datos para conectarse automáticamente.

Cliente: Inicializar la Aplicación y Comunicaciones

Al inicio de ejecución, además de la inicialización de las instancias que se utilizarán, la aplicación cliente se conectará automáticamente con el servidor, utilizando los datos capturados en la aplicación de configuración, y obtendrá la primera imagen de la escena.

Cliente: Navegación en la Interfaz de Visualización

Durante el proceso de navegación se envían eventos que modifican la escena al servidor. La imagen de la escena modificada debe ser actualizada tanto en el servidor como en el cliente. Este proceso se dispara por el uso de la pluma en el área de la imagen o por presionar los botones de dirección del equipo.

4.5 Desarrollo del Cliente

Aplicación de configuración

El prototipo de la aplicación para configurar la conexión con el servidor se muestra en la figura 4.5, se denomina Configuración Cliente. La aplicación consiste en una sola ventana con dos campos: uno es para capturar la dirección IP y el otro es para capturar el valor del puerto con el que se conectará al servidor. Contiene dos botones: Prueba y Guardar. El primero realiza una prueba de conexión con el servidor utilizando los datos capturados. El segundo almacena la información en las variables dir_ip y puerto en la llave \\HKEY_CURRENT_USER\CliGraf en la base del registro del equipo (Registry) para que puedan ser utilizados por la aplicación principal al conectarse automáticamente con el servidor.

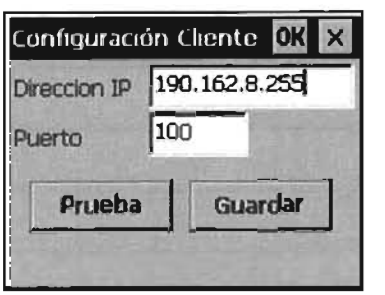


Figura 4.5 Configuración de la conexión del cliente

Aplicación principal

La aplicación principal se denomina Cliente Gráfico. Un prototipo de la interfaz se muestra en la figura 4.6. Es una ventana dividida en tres secciones:

- La región izquierda de la ventana es el área donde se muestra la imagen que proviene del servidor. En esta área el usuario puede interactuar con la escena usando la pluma.

- En la sección derecha se encuentran tres botones que representan las acciones que el usuario puede realizar sobre la escena: mover, rotar y acercar/alejar.
- En la parte inferior se muestra la posición de la cámara en coordenadas x, y, z. Contiene una línea de estatus que muestra el estado de la conexión y que también se utiliza para mostrar mensajes de error.

El usuario también puede interactuar con la escena utilizando los botones de dirección que provee el equipo. El botón central de estos botones de dirección se utiliza para cambiar la selección de la acción sobre la escena.

Figura 4.6 Aplicación cliente para la visualización de la escena

Interfaz de visualización e interacción

En el desarrollo de la interfaz se consideran dos aspectos: el manejo de la imagen de la escena y la interacción del usuario por medio de la pluma y los botones del equipo.

Para la imagen de la escena es necesario construir un z-buffer para evitar dibujar directamente en pantalla. Para esto se crea un bitmap de manera dinámica. Los recursos son reservados en memoria al construir la clase y liberados al cerrar la ventana de la aplicación. En el siguiente código se muestra como se asignan los recursos:

```
// Declaración de la estructura del bitmap
BITMAPINFO bInfo;
bInfo.bmiHeader.biBitCount = 24;                    // RGB
bInfo.bmiHeader.biClrImportant = 0;
bInfo.bmiHeader.biClrUsed = 0;
bInfo.bmiHeader.biCompression = 0;
bInfo.bmiHeader.biHeight = wpAlto;
bInfo.bmiHeader.biPlanes = 1;
bInfo.bmiHeader.biSize = 40;
bInfo.bmiHeader.biSizeImage = wpAncho*wpAlto*3; // ancho*alto*3 bytes
bInfo.bmiHeader.biWidth = wpAncho;
bInfo.bmiHeader.biXPelsPerMeter = 3780;
bInfo.bmiHeader.biYPelsPerMeter = 3780;
bInfo.bmiColors[0].rgbBlue = 0;                     // fondo negro
bInfo.bmiColors[0].rgbGreen = 0;
bInfo.bmiColors[0].rgbRed = 0;
bInfo.bmiColors[0].rgbReserved = 0;

// Asignar recursos para el bitmap
HDC hDC = ::GetDC(NULL);
zBuff = CreateDIBSection( hDC, &bInfo, DIB_RGB_COLORS,
                          (void**)&pBuffer, NULL, 0 );
::ReleaseDC(NULL, hDC);
// pBuffer apunta a la zona de memoria donde se pueden manipular
// directamente los pixeles
```

Mediante este código se obtienen al mismo tiempo una variable para el bitmap y un apuntador al buffer de la imagen.

Para mostrar la escena en pantalla es necesario agregar código en el evento OnPaint de la ventana, donde se utiliza la variable zBuff definida en la asignación de recursos para el bitmap.

```
// Evento para actualizar el contenido de la pantalla
void CCliGrafDlg::OnPaint()
{
  CPaintDC dc(this);  // variable de contexto para el área de dibujo
  Dibujar_Img( &dc ); // método que dibuja el bitmap de la escena
}

// Dibujar el contenido del bitmap en la ventana
void CCliGrafDlg::Dibujar_Img( CDC* pDC )
{
  CDC memDC; // zona de trabajo

  // asignar recursos para la zona de trabajo
  if (!memDC.CreateCompatibleDC(pDC))
     return;

  // utilizar el bitmap del z-buffer
  HBITMAP m_hOldBitmap = (HBITMAP)::SelectObject( memDC.GetSafeHdc(),
                                                  zBuff);

  // copiar el contenido del bitmap a la ventana
  pDC->BitBlt( area_dib.left, area_dib.top, wpAncho, wpAlto, &memDC,
               0, 0, SRCCOPY);

  // liberar recursos de la zona de trabajo
  ::SelectObject(memDC.GetSafeHdc(), m_hOldBitmap);
  memDC.DeleteDC();
}
```

Para la interacción del usuario, se definen los métodos OnLButtonDown y OnLButtonUp para capturar la interacción de la pluma con la pantalla. En el primer método se captura la posición de la pluma cuando ésta toca la pantalla. El segundo captura la posición cuando ésta se retira de la pantalla. Además, este segundo método determina la dirección del movimiento relativo de la pluma utilizando la información del primer método.

En el método OnLButtonUp se envía el evento al servidor. El paquete del evento se conforma con el comando que identifica la acción (mover, rotar o acercar), la última posición de la pluma y la dirección de movimiento relativa de la pluma en la interfaz. La información de la posición y movimiento se envían como porcentajes del área de la imagen. En el siguiente código se muestran los métodos para estos eventos, la estructura de datos del paquete se define a detalle más adelante:

```
// Evento en que la pluma toca la pantalla
void CCliGrafDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
```

```
  // procesar solo si esta dentro del área de dibujo
  if ((point.y >= area_dib.top)  && (point.y <= area_dib.bottom) &&
     (point.x >= area_dib.left) && (point.x <= area_dib.right))
  {
    pluma_x = point.x;
    pluma_y = point.y;
  }
  CDialog::OnLButtonDown(nFlags, point);
}

// Evento en que la pluma se retira de la pantalla
// Enviar mensaje del evento al servidor
void CCliGrafDlg::OnLButtonUp(UINT nFlags, CPoint point)
{
  // procesar solo si esta dentro del área de dibujo
  if ((point.y >= area_dib.top)  && (point.y <= area_dib.bottom) &&
      (point.x >= area_dib.left) && (point.x <= area_dib.right) &&
      (pluma_x !=-1))
    {
      // la posición y el desplazamiento respecto a la posición donde
      // toco la pantalla se envían como porcentaje de las dimensiones
      // del área de la escena
      paq_cmd_sal.flags[0] = 0;     // bandera de navegación por pluma

      // Posición
      paq_cmd_sal.i1 = ((pluma_x - area_dib.left)*100) / wpAncho;
      paq_cmd_sal.i2 = ((area_dib.bottom - pluma_y)*100) / wpAlto;

      // Desplazamiento
      paq_cmd_sal.i3 = ((pluma_x - point.x)*100) / wpAncho;
      paq_cmd_sal.i4 = ((point.y - pluma_y)*100) / wpAlto;

      // Enviar la acción que se encuentra seleccionada
      EnviarCmd( accion_sel );
    }

    pluma_x = pluma_y = -1;
    CDialog::OnLButtonUp(nFlags, point);
}
```

La interacción por medio de los botones de navegación se controla mediante el método OnKeyUp. El paquete del evento es similar al evento de la pluma. La diferencia está en que se envía el botón presionado en lugar de las coordenadas de la pluma. Para diferenciar un evento generado por un botón de uno generado por la pluma, el valor de la bandera del método de navegación es 0 para la pluma y 1 para los botones. El evento del botón central de navegación no se envía al servidor. Es procesado en la interfaz para cambiar el botón de acción seleccionada (Mover – Rotar – Acercar – Mover).

Comunicaciones

Para la comunicación con el servidor se utilizan sockets TCP/IP. Para esto, se crea la clase CDeCeSocket derivada de la clase CCeSocket que proporciona el compilador EVC. En esta clase derivada se redefine el evento OnReceive para dar el control a la aplicación que permita procesar los paquetes de datos que se reciben.

La conexión con el servidor se realiza mediante el método Conectar, que construye dos instancias de la clase CDCeSocket: una para el puerto donde se transmiten los eventos y otra para la transmisión de las imágenes.

El valor de la dirección IP del servidor y del puerto para la transmisión de paquetes de eventos se leen de la base de registro del equipo en la llave utilizada en la aplicación de Configuración del Cliente. El valor del puerto para la transmisión de datos de la imagen es el valor del puerto de eventos más uno. Así, si el puerto de eventos es el 100, el puerto de datos de la imagen será el 101.

El evento OnReceive del socket de eventos recibe paquetes de eventos y los pasa al método Proc_Msg_EC (Procesar Mensaje de Evento/Comunicación) que los interpreta. El paquete está definido en la estructura Packet_EC como sigue:

```
struct Packet_EC                // Estructura de los paquetes de eventos
{                               // byte : dato             : tamaño
  unsigned char cmd[3];         //    0 : comando          :    3 bytes
  unsigned char flags[5];       //    3 : banderas         :    5 bytes
  int  i1, i2, i3, i4,          //    8 : parametros       :   16 bytes
       tam_buff;                //   24 : long. del buffer :    4 bytes
  unsigned char datos[996];     //   28 : buffer de datos  :  996 bytes
};                                                 // total : 1024 bytes
```

El evento OnReceive del socket de datos recibe paquete de datos de la imagen y los pasa al método Proc_Msg_DI (Procesar Mensaje de Datos de la Imagen). El paquete está definido en la estructura Packet_DI como sigue:

```
struct Packet_DI                // Estructura de los paquetes de datos
{                               // byte : dato             : tamaño
  unsigned char cmd[3];         //    0 : comando          :     3 bytes
  unsigned char flag;           //    1 : bandera          :     1 byte
  int desp,                     //    4 : desplazamiento   :     4 bytes
      tam_buff;                 //    8 : long. del buffer :     4 bytes
  unsigned char datos[58000]; //   12 : buffer de datos  : 58000 bytes
};                                                    // : 58012 bytes
```

El tamaño máximo del buffer de datos de la imagen de esta estructura es independiente del tamaño de la imagen. Si el tamaño del buffer de la imagen es mayor que el tamaño de la estructura del paquete, el buffer de la imagen se transmite en varios paquetes. Para una imagen de 160x120 píxeles RGB sin comprimir (que es la implementada en este proyecto) se requieren 57,600 bytes, por lo que esta estructura es suficiente para enviar los datos en un solo paquete.

En el método Proc_Msg_DI se reciben los paquetes de datos de la imagen, se copia el contenido de estos al z-buffer de la imagen y se llama al método para actualizar el contenido de la pantalla.

La cantidad de bytes que se reciben en el buffer del socket es variable, entre 4,000 y 32,000 bytes. Dado que el tamaño del paquete de datos es mayor que estos valores, normalmente

se generan varios eventos OnReceive para el socket de datos. En cada evento se recibe un fragmento del paquete. Por esto, es necesario modificar un poco el código del método Proc_Msg_DI para tomar en cuenta esta situación. Este sería el código suponiendo que el paquete se recibe en un solo evento:

```
void CCliGrafDlg::Proc_Msg_DI()
{
  // leer los datos del buffer de la imagen del socket
  nRead = cnx_di->Receive(&paq_datos, 57612 );

  // copiar los datos al z-buffer de la imagen
  if (strncmp( (char*)buff_read, "201", 3)==0) {
     memcpy( pBuffer, paq_datos.datos, 57600 );
     // ... llamar a los métodos para actualizar la pantalla
  }
}
```

La adaptación del código del método Proc_Msg_DI para recibir el paquete en fragmentos es como sigue:

```
void CCliGrafDlg::Proc_Msg_DI()
{
  // leer los datos del buffer del socket
  nRead = cnx_di->Receive(buff_read, 4096 );

  // recepción del primer fragmento del paquete
  if (strncmp( (char*)buff_read, "201", 3)==0) {
     memcpy( &paq_datos, buff_read, nRead );
     nBytes = nRead;
  }

  // recepción de los fragmentos restantes
  if ((strncmp( (char*)buff_read, "201", 3)!=0) && (nBytes > 0)) {
     memcpy( (&paq_datos)+nBytes, buff_read, nRead );
    nBytes += nRead;
  }

  // si se ha completado la lectura del paquete
  // copiar los datos al z-buffer y dibujar la escena en pantalla
  if (nBytes == 57608) {
     memcpy( pBuffer, paq_datos.datos, 57600 );
     nBytes = 0;

     // llamar a los métodos para actualizar la pantalla
     InvalidateRect(inv_area_dib, FALSE);
     UpdateWindow();
  }
}
```

4.6 Desarrollo del Servidor

El prototipo de la interfaz del servidor se muestra en la figura 4.7. La aplicación está constituida por cuatro módulos, tres de estos son ventanas:

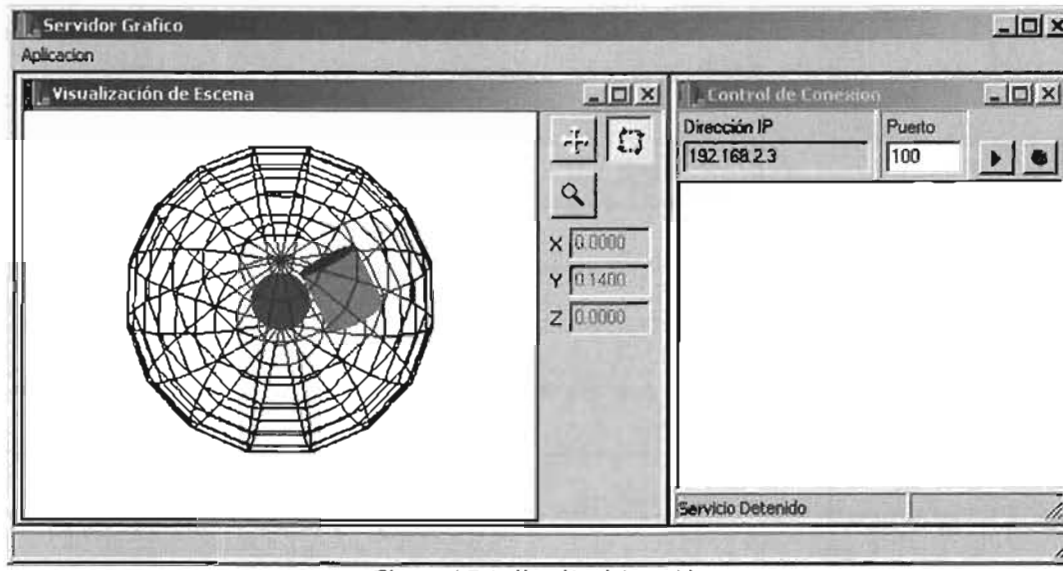


Figura 4.7 Aplicación del servidor

- La ventana del menú principal. Establece el área de trabajo para la aplicación y los menús para abrir las ventanas de la aplicación.
- La ventana de Visualización de Escena. Contiene la interfaz al usuario. En esta ventana se muestra la imagen del modelo y aparecen los controles para que el operador del servidor pueda modificarla. En la clase que define esta ventana se encuentran los métodos para aplicar las transformaciones a la escena y para obtener la imagen que se enviará al cliente.
- La ventana de Control de Conexión. En este módulo se controla la transmisión de paquetes de datos con el cliente, se reciben los eventos y se transmiten las imágenes. Contiene botones para iniciar y terminar el servicio, en el área de mensajes aparecen informes de la actividad del cliente.
- El módulo Modelo. Utiliza los métodos de la librería gráfica OpenGL para definir y transformar la escena. En este módulo se forma la imagen que se transfiere al cliente.

El usuario de la aplicación del servidor puede modificar la escena utilizando el ratón y las teclas de dirección del teclado.

El menú Aplicación contiene tres opciones: Imagen, Conexión y Salir. La opción Imagen abre la ventana de Visualización de Escena; la opción Conexión abre la ventana de Control de Conexión; y la opción Salir termina la ejecución de la aplicación.

Modelado de la Escena

El modelado de la escena se encuentra encapsulado en la clase TModelo que contiene los métodos para construirla y modificarla, mediante el uso de la librería grafica OpenGL. En el siguiente código se muestra la declaración de la clase:

```
class TModelo
{          // declaración de instancias y métodos para
 private:  // construir y manipular la escena
    HGLRC hRC;  // Recurso para dibujar la escena, debe ser proporcionado
                // por la clase de la interfaz de visualización

    // Datos de la posición de la cámara
    float latitud, longitud, latinc, longinc,
          x_desp, y_desp, z_desp, x_desp_inc, y_desp_inc, z_desp_inc;

    GLvoid CrearObjetos();   // crear los objetos de la escena

    // calcular posición de la camara
    void CalcCamPos( void );

 public: // interfaz pública

    // Métodos de acceso al controlador para dibujar la escena
    void AsignarRC( HGLRC rc );  // Asignar valor
    HGLRC ghRC( void );          // Leer valor

    float PosX( void );     // Información de la posición de la cámara
    float PosY( void );
    float PosZ( void );

    // Inicializar librería grafica
    void InicializarGL( int ancho, int alto );

    // Método para cambiar el tamaño de la imagen
    void CambTamImagen( int ancho, int alto );

    // Método para dibujar la escena
    void DibEscena();

    // Método para construir un bitmap de la escena
    void LeerBitmap( int ancho, int alto, unsigned char *buff );

    // Modificar la posición de la cámara (desplazamiento)
    void MoverCamara( float dx, float dy, float dz );

    // Modificar la posición de la cámara (orientación)
    void RotarCamara( float dlon, float dlat );
};
```

La interfaz pública de esta clase es independiente de la biblioteca grafica utilizada. Esto permite que se pueda implementar esta clase utilizando otros medios para calcular la imagen de la escena, como DirectX.

Para utilizar una instancia de esta clase deben proporcionarse los recursos para dibujar la escena en la interfaz, además de las dimensiones iniciales de esta.

El método InicializarGL realiza tres tareas: inicializar la librería gráfica con las dimensiones iniciales de la interfaz; construir los objetos de la escena; y calcular la primera imagen.

La posición de la cámara esta representada con las variables: x_desp, y_desp y z_desp (x, y, z), la orientación con: latitud y longitud. Las unidades de movimiento de traslación están en las variables: x_desp_inc, y_desp_inc, z_desp_inc (dx, dy, dz) y las unidades para la rotación en: latinc, longinc. La posición de la cámara se modifica con el método MoverCamara. La orientación se modifica con el método RotarCamara.

El método CambTamImagen cambia las dimensiones de la imagen de la escena. Este método se utiliza cuando cambia el tamaño de la interfaz en el servidor y para calcular la imagen en las dimensiones de la interfaz del cliente.

Las imágenes de la escena son calculadas con el método DibEscena, por medio los métodos de la librería gráfica considerando la posición de la cámara.

Los bitmaps que se envían al cliente se calculan con el método LeerBitmap. Para esto deben proporcionarse las dimensiones de la imagen y un apuntador a una zona en memoria donde se guardará la información.

Interfaz de Visualización

La interfaz de visualización está definida en la clase TfImagen que controla la ventana Visualización de Escena. Las funciones de esta clase son crear la escena, navegar por ésta y proporcionar los datos de la imagen que se enviarán al cliente.

Un usuario en el servidor puede también navegar por la escena como lo haría desde la aplicación remota. Para esto, la interfaz contiene los mismos controles que tiene la aplicación cliente, los botones de acción sobre la escena (Mover – Rotar – Acercar/Alejar) y los campos que indican la posición de la cámara. La navegación puede realizarse con el ratón o con las teclas de dirección.

La navegación por la escena se realiza mediante el método Procesar. Este método modifica la posición de la cámara del modelo de la escena en función del comando proporcionado y ejecuta los métodos del modelado que calculan la nueva escena la dibujan en la interfaz. En el siguiente código se muestra la implementación de este método:

```
// Procesar un evento de navegación en la escena
void __fastcall TfImagen::Procesar( int ev_nav, int ft, int i1, int i2,
                                    int i3, int i4 )
{
 // Acción Mover
 if (ev_nav == NV_MOVER) {
     if (ft==0) // si es navegación por pluma o ratón
         Modelo.MoverCamara( i3/100.0, -i4/100.0, 0.0 );
     else { // navegación por botones
         if (i1 == BT_IZQ) Modelo.MoverCamara( -0.1,  0.0, 0.0 );
         if (i1 == BT_DER) Modelo.MoverCamara(  0.1,  0.0, 0.0 );
         if (i1 == BT_ARR) Modelo.MoverCamara(  0.0,  0.1, 0.0 );
         if (i1 == BT_ABJ) Modelo.MoverCamara(  0.0, -0.1, 0.0 );
        }
    }
```

```
// Acción Rotar
if (ev_nav == NV_ROTAR) {
    if (ft==0) // si es navegación por pluma o ratón
        Modelo.RotarCamara( -i3/100.0, -i4/100.0 );
    else { // navegación por botones
        if (i1 == BT_IZQ) Modelo.RotarCamara( -0.1,  0.0 );
        if (i1 == BT_DER) Modelo.RotarCamara(  0.1,  0.0 );
        if (i1 == BT_ARR) Modelo.RotarCamara(  0.0, -0.1 );
        if (i1 == BT_ABJ) Modelo.RotarCamara(  0.0,  0.1 );
       }
   }

// Acción de Acercar/Alejar
if (ev_nav == NV_ZOOM) {
    if (ft==0) // si es navegación por pluma o ratón
        Modelo.MoverCamara( 0.0, 0.0, i4/100.0 );
    else { // navegación por botones
        if (i1 == BT_ARR) Modelo.MoverCamara( 0.0, 0.0, -0.1 );
        if (i1 == BT_ABJ) Modelo.MoverCamara( 0.0, 0.0,  0.1 );
       }
   }

Modelo.DibEscena();  // Calcular la nueva imagen de la escena
SwapBuffers(ghDC);   // mostrar la imagen en pantalla
}
```

Los eventos de navegación locales (ratón o teclas de dirección) se aplican en la escena mediante el mismo método que se construye para procesar los eventos que provienen del cliente. En el siguiente código se muestra como se procesan los eventos del ratón, cuya implementación es similar a la presentada para los eventos de la pluma en la aplicación cliente:

```
// Navegación en la escena por medio del ratón
// detectar posición inicial
void __fastcall TfImagen::FormMouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
 if (Button != mbLeft) return; // procesar solo mensajes
                               // del botón izquierdo
 x_pos = X;
 y_pos = Y;
}

// Navegación en la escena por medio del ratón, detectar posición final
// Generar evento de navegación sobre la escena
void __fastcall TfImagen::FormMouseUp(TObject *Sender,
                                      TMouseButton Button,
                                      TShiftState Shift, int X, int Y)
{
 if (Button != mbLeft) return; // procesar solo mensajes
                               // del botón izquierdo

 int desp_x = X - x_pos,  // calcular el desplazamiento
     desp_y = Y - y_pos;
```

```
    // generar evento de navegación
    Procesar( id_ctl_sel, 0, X*100/(ClientWidth - Panel1->Width),
              Y*100/ClientHeight,
              desp_x*100/(ClientWidth - Panel1->Width),
              desp_y*100/ClientHeight);
}
```

Para tener acceso al bitmap de la imagen que se enviará al cliente se define el método CargarBitmap, que utiliza el método LeerBitmap de la clase TModelo y ajusta los datos para que puedan ser procesados correctamente por el cliente. La implementación se muestra en el siguiente código:

```
// Cargar el bitmap del modelo en el buffer de transferencia
void __fastcall TfImagen::CargarBitmap( int ancho, int alto,
                                        unsigned char *buff )
{
 Modelo.CambTamImagen( ancho, alto); // cambiar al tamaño solicitado
 Modelo.DibEscena();                 // dibujar la escena

 // Copiar los pixeles de la imagen al buffer
 Modelo.LeerBitmap( ancho, alto, buff );

 // recuperar el tamaño original
 Modelo.CambTamImagen( ClientWidth - Panel1->Width, ClientHeight);

 // el buffer tiene los bytes de los pixeles organizados como RGB
 // cambiar el orden a BGR que es el formato del bitmap
 for (int i=0; i<ancho*alto; i++) {
      unsigned char aux = buff[i*3];
      buff[i*3] = buff[i*3+2];
      buff[i*3+2] = aux;
     }
}
```

Comunicaciones

Para las comunicaciones se define la clase TfConexion. Esta clase controla la ventana de Control de Conexión, que contiene los controles para iniciar y terminar el servicio que presta la aplicación.

Se utiliza una instancia de la clase TPowerSocket para obtener la dirección IP del servidor al inicializar la clase. Esta dirección es la que se captura en la ventana de configuración del cliente. El campo Puerto es el que define cual será el puerto de eventos. Este corresponde con el capturado en la ventana de configuración del cliente.

La clase utilizada para construir los sockets en el servidor es TServerSocket. Al igual que en el cliente se construyen dos instancias: una para eventos y otra para datos. Sólo en la definición del socket de eventos se construye el método para el evento OnRead que responde a los mensajes que se reciben del cliente.

Se construyen dos instancias de la estructura Packet_EC: una para enviar y otra para recibir paquetes de eventos. También se construye una instancia de la estructura Packet_DI para la transmisión de las imágenes.

El evento OnRead del socket de eventos se construye para recibir los paquetes del cliente y los pasa al método ProcesarCmd que los interpreta. El evento puede ser de dos tipos: de acción sobre la escena o de comunicación.

Los eventos de acción sobre la escena se envían al método Procesar de la ventana de Visualización de Imagen que los interpreta y aplica. Los eventos de comunicación son procesados por el propio método ProcesarCmd de la clase TfConexion. En el siguiente código se muestra la implementación de este método:

```
// Procesar un comando recibido por el puerto de eventos
void __fastcall TfConexion::ProcesarCmd( void )
{
 // Eco de reconocimiento de la conexión
 if (strncmp( paq_cmd_ent.cmd, "000", 3 )==0)
     EnviarComando("000");

 // Comando para finalizar la conexión
 if (strncmp( paq_cmd_ent.cmd, "001", 3 )==0)
    {
     Memo_Log->Lines->Add("Cerrando conexión");
     EnviarComando("002"); // Aviso de conocimiento del evento
    }

 // Definición del tamaño de la imagen en el cliente
 // Crear el buffer de datos para la transferencia de la imagen
 if (strncmp( paq_cmd_ent.cmd, "100", 3 )==0)
    {
     if (fImagen)
        {
         Ancho = paq_cmd_ent.i1;            // parámetro 1: Ancho
         Alto  = paq_cmd_ent.i2;            // parámetro 2: Alto

         if ((Ancho <=0) && (Alto <=0))     // tamaño invalido
             EnviarComando("901");
         else {
             Memo_Log->Lines->Add("100 Datos de ventana cliente");

             tam_buffer = Ancho * Alto * 3;
             bmpBuffer = new unsigned char[tam_buffer];
             EnviarComando("200");          // buffer de imagen creado
            }
        }
     else
        EnviarComando("900"); //Error: falta la interfaz en el servidor
    }

 // Solicitud de la primera imagen
 if (strncmp( paq_cmd_ent.cmd, "101", 3 )==0)
     EnviarImagen();
```

```
// Eventos de navegación, procesar y enviar imagen al cliente
if (paq_cmd_ent.cmd[0] == '3')
    {
     char s_cmd[4];
     strncpy(s_cmd, paq_cmd_ent.cmd,3);

     // Se envía el evento a la interfaz para procesarlo
     fImagen->Procesar( atoi( s_cmd ), paq_cmd_ent.flags[0],
                        paq_cmd_ent.i1, paq_cmd_ent.i2,
                        paq_cmd_ent.i3, paq_cmd_ent.i4 );
     EnviarImagen();
    }
}
```

Para enviar los paquetes de datos al cliente se definen los métodos EnviarComando y EnviarImagen. Para enviar paquetes de eventos, bajo la estructura Packet_EC, y datos de la imagen, bajo la estructura Packet_DI, respectivamente.

El método EnviarComando se ejecuta como respuesta a cada paquete de un evento que envía el cliente, exceptuando los de navegación. La respuesta a los eventos de navegación se realiza mediante el método EnviarImagen, que también se ejecuta cuando se envía la primera imagen al cliente al inicio de la conexión.

El método EnviarImagen ejecuta el método CargarBitmap de la clase de la ventana de Visualización de Escena para llenar el buffer de la imagen en el paquete de datos de la imagen antes de enviarlo al cliente. También se envía la posición de la cámara, que se obtiene del método CargarPosCam, por el puerto de eventos. El siguiente código muestra la implementación de este método:

```
// Enviar la imagen al cliente
void __fastcall TfConexion::EnviarImagen( void )
{
 if (tam_buffer == 0) {      // verificar si esta definido el bitmap
    {
     EnviarComando("902");  // no hay bitmap de datos definido
     return;
    }

 // Bytes por enviar
 int nBytes = min( tam_buffer-desp, TAM_BUFF_DI );

 // Se envía el primer fragmento de la imagen, obtener el bitmap
 // y enviar posición de la cámara
 if (desp == 0)
    {
     fImagen->CargarPosCam( paq_cmd_sal.i1, paq_cmd_sal.i2,
                            paq_cmd_sal.i3 );
     EnviarComando("400");  // Posición de la cámara
     fImagen->CargarBitmap( Ancho, Alto, bmpBuffer );
    }

 // copiar datos al paquete en función de la sección del bitmap
 // que se enviará
```

```
    memcpy( paq_datos.cmd, "201", 3 );
    memcpy( paq_datos.datos, bmpBuffer+desp, nBytes );

    paq_datos.desp = desp;              // posición en el bitmap
    paq_datos.tam_buff = nBytes;        // bytes que se envían

    Serv_DI->Socket->Connections[0]->SendBuf( &paq_datos, nBytes+12 );
  }
```

4.7 Pruebas

Para probar la aplicación prototipo se construyó un modelo gráfico básico en la aplicación servidor. Las pruebas consistieron en verificar que los eventos de navegación se ejecutarán apropiadamente y que la imagen llegara completa al cliente. Se verificó la ocurrencia de las excepciones de los casos de usos.

Latencia

En las pruebas de uso se encontró que la aplicación es capaz de desplegar 9 cuadros por segundo en promedio, es decir, unos 110 ms por cuadro. Considerando que la velocidad de la red Wi-Fi utilizada es de 11 Mbps o 1,455,872 bytes por segundo. El tiempo para transmitir 57,600 bytes a esta velocidad es de 0.03956 seg, 40 ms aproximadamente. Para determinar los 70 ms restantes se hicieron algunas modificaciones al cliente.

Para verificar este tiempo se modificó el método de recepción de datos en el cliente (Proc_Msg_DI). Se creó un método para solicitar 5,760,000 bytes de información al servidor (equivalente a 100 cuadros). El tiempo observado fue de 5 segundos, por lo que el tiempo promedio para 57,600 bytes es de 50 ms. Esta diferencia de 10 ms resulta por tres causas: la información es fragmentada en paquetes de menor tamaño; cada paquete de datos aumenta de tamaño debido a que el protocolo agrega información como: tamaño, dirección de destino, etc.; y el tiempo que ocupa el método en el cliente para procesar los paquetes.

Para determinar el tiempo de procesamiento en el servidor se creo un método para solicitar 100 cuadros de imagen al servidor. Se modificó el método de recepción de datos en el cliente (Proc_Msg_DI) para que no llenara el buffer de la imagen. El tiempo observado fue de 7 segundos, 70 ms por cuadro. Dado que el tiempo para transmitir los datos (incluyendo el tiempo de recepción de los fragmentos) es de 50 ms, el tiempo para este proceso es de 20 ms.

Por último, se modifico el método de recepción de datos en el cliente (Proc_Msg_DI) para permitir que llenara el buffer de la imagen y actualizara la pantalla. El tiempo observado para los 100 cuadros fue de 10 segundos, 100 ms. Descontando el tiempo del proceso anterior resultan 30 ms para este proceso.

Los últimos 10 ms que faltan para completar los 70 ms analizados se consideran como el tiempo que ocupa la aplicación para: capturar un evento en la interfaz de visualización, enviar el evento al servidor, y que el servidor aplique el evento en la escena antes de enviar la nueva escena.

El siguiente cuadro es el resumen de los tiempos observados con que contribuye cada uno de los distintos procesos de la aplicación en la generación de un cuadro de imagen.

Tiempo	Proceso
10 ms	Capturar el evento y enviarlo al servidor
20 ms	Procesamiento en el servidor
40 ms	Transmisión de la imagen por la red (57600 bytes)
10 ms	Recepción del paquete de datos de la imagen
30 ms	Llenar el buffer de la imagen y actualizar la pantalla

Funcionamiento

En esta sección se muestran algunas imágenes de la aplicación prototipo en funcionamiento durante las pruebas.

El primer paso es configurar la conexión del cliente con el servidor. Para esto se captura la dirección IP y el puerto mediante la aplicación Configuración Cliente. Los valores deberán ser los que aparecen en la ventana de Control de Conexión del servidor, como se muestra en la figura 4.8.

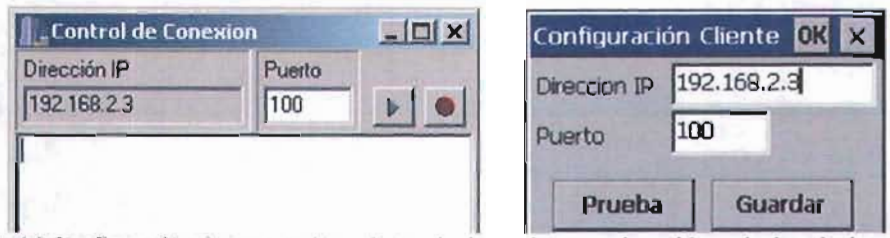


Figura 4.8 Configuración de la conexión, a la izquierda los datos en el servidor, a la derecha los valores capturados en la aplicación de configuración del cliente

Una vez capturados estos datos y con el servidor en funcionamiento, se ejecuta la aplicación en el dispositivo portátil. La conexión se realiza automáticamente y aparece la primera imagen del modelo. La interacción con la escena se realiza mediante la pluma en el área de la imagen o por medio de los botones de navegación que tiene el equipo. El tipo de acción realizada depende de la acción seleccionada en la sección derecha de la aplicación. En la figura 4.9 se muestra la aplicación en funcionamiento.

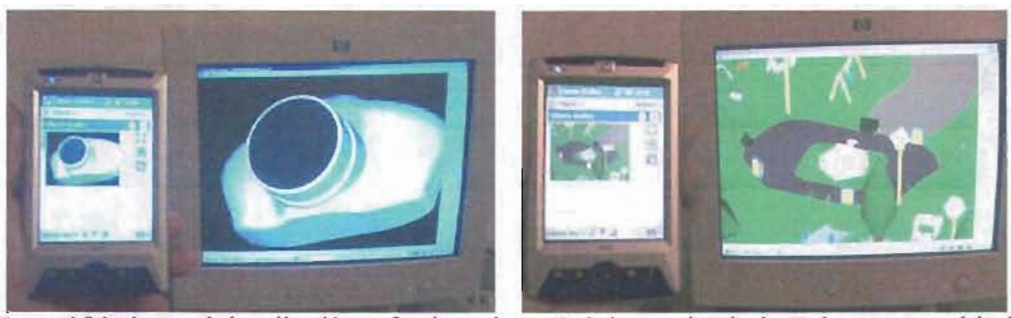


Figura 4.9 Imágenes de la aplicación en funcionamiento. En la imagen izquierda se observa un modelo de una cámara fotográfica. En la imagen de la derecha se observa un modelo de un parque

La aplicación es independiente de las dimensiones de la imagen en la interfaz del cliente como se muestra en la figura 4.10.

Figura 4.10 Imágenes de la aplicación visualizando el modelo de prueba con distintas dimensiones, de izquierda a derecha, 160x120, 160x60 y 80x120 pixeles, respectivamente.

Conclusiones

El objetivo de este trabajo era analizar y describir objetivamente el potencial del cómputo móvil aplicado a la visualización remota de imágenes generadas por computadora. Los esfuerzos se dirigieron a utilizar los equipos, dispositivos y herramientas de desarrollo comercialmente disponibles.

En lo que concierne a los objetivos específicos se puede decir que todos fueron atendidos. Se mostró una visión del estado actual del cómputo móvil, de las redes inalámbricas y de las capacidades graficas de los equipos, donde el desarrollo de aplicaciones aún no es el ideal comparado con aplicaciones en equipos fijos.

Fue desarrollado un prototipo de una aplicación para visualizar y controlar imágenes calculadas remotamente. El desarrollo de la aplicación demostró no ser más compleja que si se hubiera hecho para un equipo fijo, salvo algunos detalles muy puntuales propios de la naturaleza del dispositivo portátil. Esto gracias a que estos dispositivos se diseñan de tal forma que funcionen de manera similar a sus contrapartes fijas.

En lo que se refiere a herramientas de desarrollo para este tipo de aplicaciones, el mercado tiene diversas propuestas tecnológicas. Las grandes compañías han hecho esfuerzos para llevar las ventajas de sus productos diseñados para grandes equipos hacia los dispositivos portátiles. Una señal a tomar en cuenta respecto a la visión que tienen estas compañías respecto al cómputo móvil.

En capacidad de procesamiento, los equipos móviles son equivalentes a los equipos de escritorio de hace unos ocho años y la brecha se reduce continuamente. También los costos de estos dispositivos se han estado reduciendo, por lo que en los próximos años estarán al alcance de muchas personas.

El entretenimiento es el área que más se ha explotado, tanto en aplicaciones de contenido local como de acceso a contenido remoto, tales como juegos o reproductores de audio y video. Desde hace algún tiempo ya es factible tener acceso a bases de datos remotas y ya existen algunas aplicaciones de servicios de información. El uso de la PDA como herramienta en el salón de clases, así como la educación a distancia [MHV02], son áreas que han comenzado a explorarse. Las posibilidades de uso de estos equipos son muy variadas gracias a la movilidad.

La tendencia hacia integrar redes caseras, considerando a los equipos de escritorio como los servidores de estas redes para interconectar y controlar los diversos dispositivos de la casa sin importar donde este el usuario, dejará el cómputo personal en los dispositivos portátiles. Es de esperarse que la mayoría de las aplicaciones disponibles para equipos de escritorio estén disponibles para ambientes móviles a corto plazo. Es por esto, que diversas empresas apoyan el desarrollo de redes de bajo costo y bajo consumo de energía como Bluetooth y ZigBee.

Las posibles aplicaciones que se pueden construir aprovechando la movilidad de estos dispositivos son muy variadas. Algunos de los proyectos que pueden construirse utilizando una aplicación gráfica remota como la desarrollada en este trabajo son:

- Consulta de bases de datos de modelos tridimensionales. Una aplicación de este tipo se puede usar en el campos como la ingeniería, la industria o la medicina, permitiendo comparar modelos virtuales tridimensionales, almacenados en un servidor, con los objetos reales en el sitio donde se encuentren.
- En medicina, además de visualizar modelos virtuales, se tendría acceso a imágenes generadas por equipos especializados, como los de resonancia magnética, sin tener que estar en el lugar donde se generan, permitiendo hacer diagnósticos a distancia.
- Como aplicación comercial, se pueden crear vendedores personalizados, cuyas funciones pueden ir, desde mostrar las ofertas del día, hasta ayudar a buscar un producto especifico, apoyándose en tecnologías como Bluetooth y ZigBee para ubicar al cliente y los productos.
- En la educación, es factible la visualización de modelos virtuales y gráficas complejas en el salón de clases. En la educación a distancia, se tendría acceso a este tipo de imágenes sin necesidad de tener un equipo especializado para calcularlas. También pueden crearse guías personalizados para museos o sitios arqueológicos donde no es deseable instalar cables.
- En sitios como aeropuertos, estaciones de tren o autobuses, es posible proporcionar mapas de las instalaciones a los usuarios, sin que tengan que descargarlos de alguna base de datos.

Líneas Futuras

Para ampliar las capacidades de la aplicación se recomienda continuar con los siguientes proyectos:

1. Modificar los módulos de transmisión y recepción de datos, agregando métodos de compresión y descompresión de datos, para poder enviar las imágenes a través de redes celulares, cuyas tasas de transferencias son más bajas.
2. Agregar métodos de encripción de datos y manejo de contraseñas de acceso, para utilizar la aplicación en redes públicas de forma segura.
3. Transmitir audio y video, considerar también la posibilidad de que el cliente pueda enviar este tipo de información hacia el servidor.

Bibliografía

[**Bi04**] Gerald Binder, "PocketHouseAR - An Approach to Use a Pocket PC as Interaction Tool for Augmented Reality", Media Technology and Design, Upper Austria University of Applied Sciences, Hagenberg College of Information Technology, Austria, 2004

[**BM05**] "Borland Mobile", http://www.borland.com/mobile/, Borland Software Corporation

[**CFA05**] "CrossFire", http://www.appforge.com/products/enterprise/crossfire/, AppForge Inc

[**Chr02**] "Chromium", http://sourceforge.net/projects/chromium, sourceforge.net, 2002

[**CS05**] "CASL", http://www.caslsoft.com/, Feras Information Technologies, CASLsoft

[**CW05**] "CodeWarrior Technology", http://www.metrowerks.com/mw/default.htm, Metrowerks

[**EH00**] "Marconi, Guglielmo", "Radiocomunicación", Enciclopedia Hispánica, Volúmenes 9 y 12, Barsa International Publishers Inc., Estados Unidos, 2000

[**FMHW97**] Steven Feiner, Blair MacIntyre, Tobias Höllerer, Anthony Webster, "The Touring Machine", http://www1.cs.columbia.edu/graphics/projects/mars/touring.html, Columbia University, Computer Graphics and User Interfaces Lab, International Symposyum on Wearable Computing, Cambridge, MA, 13-14, octubre 1997, páginas 74–81

[**GVT03**] Mario Gutiérrez, Frederic Vexo, Daniel Thalmann, "Controlling Virtual Humans Using PDAs", Virtual Reality Lab (VRlab), EPFL, Suiza, Conference on Multimedia Modeling, Taipei, Taiwan, 2003

[HO03] Anders Henrysson, Mark Ollila, "Augmented Reality on Smartphones", IEEE 2nd International Augmented Reality Toolkit Workshop, 27-28, octubre 2003

[HPH05] "HP History and Facts", http://www.hp.com/hpinfo/abouthp/histnfacts/, Hewlett-Packard Development Company, L.P., 2005

[KG04] Khronos Group, "OpenGL ES Overview", http://www.khronos.org/opengles/index.html, 2004

[KPL05] "KADAK", http://www.kadak.com/, Kadak Products Ltd

[KR03] Vadim Kalinin, Vladimir Rafalovich, "Palm & Pocket PC Programming", A-List Publishing, marzo 2003

[Le02] Pierre Leroy, "PocketGL" http://pierrel5.free.fr/PocketGLb/ReadMeFirst.htm, 2002

[LZSFM03] Fabrizio Lamberti, Claudio Zunino, Andrea Sanna, Antonino Fiume, Marco Maniezzo, "An accelerated remote graphics architecture for PDAS", DAUIN - Politecnico di Torino, Proceeding of the eighth international conference on 3D Web technology, Saint Malo, Francia, 2003, ACM Press New York, NY, USA

[Me03] David Mery, Technology Outreach, Symbian Ltd, "Why is a different operating system needed?", http://www.symbian.com/technology/why-diff-os.html, revisión: 2.4, octubre 2003

[MT03] "Wi-Fi Protected Access (WPA) Overview", http://www.microsoft.com/technet/community/columns/cableguy/cg0303.mspx, The Cable Guy, Microsoft TechNet, marzo 2003

[MHV02] M. A. Moreno Rocha, E. E. Hernández Rueda, M. A. Villarroel Salgueiro, "Incorporando Dispositivos PDA a la Educación a Distancia", Universidad Tecnológica de la Mixteca, Oaxaca, México, Depto. Informática-Universidad de Valladolid, Campus Miguel Delibes, Valladolid, España, 2002

[PO05] "Palm Historical Timeline", http://www.palmone.com/us/company/corporate/timeline.html, palmOne Inc

[PSD05] "Palm Source Developers", http://www.palmsource.com/developers/, palmSource Inc.

[Pw04] Palowireless Pty Ltd, "What is ZigBee", http://www.palowireless.com/zigbee/whatis.asp, 2004.

[SDH05] "Symbian Developer Home", http://www.symbian.com/developer/index.html, Symbian Ltd.

[sMC04] "802.xx Fast Reference", http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci992311,00.html, searchMobileComputing.com, octubre 2004

[SOP05] "Symbian OS Phones", http://www.symbian.com/phones/index.html, Symbian Ltd

[SS04] Sander Siezen, "Symbian OS Version 8.0 functional description", http://www.symbian.com/technology/symbos-v8x-det.html, revisión 2.1, febrero 2004

[WAD05] "Windows CE Application Development", http://msdn.microsoft.com/embedded/usewinemb/ce/appdev/default.aspx, Microsoft Corporation

[WCE05] "Windows CE.net", http://msdn.microsoft.com/embedded/prevver/ce.net/, Microsoft Corporation

[We04] John C. Welch, "Wireless Networking and the AirPort", http://www.mactech.com/articles/mactech/Vol.15/15.12/WirelessNetworking/, MacTech, Vol. 15, Issue 12, 2004

[WLA99] WLANA, Wireless LAN Association, "Introduction to Wireless LAN", http://www.wlana.org/learn/intro.pdf, 1999.

[WS03] Daniel Wagner, Dieter Schmalstieg, "ARToolKit on the PocketPC Platform", Interactive Media Systems Group, Institute for Software Technology and Interactive Systems, Vienna University of Technology, Austria, 2003

– Guan-Ming Su, Min Wu, K. J. Ray Liu, "ENEE408G Multimedia Signal Processing Network Programming Manual, Using Microsoft Visual Studio.NET C++ And eMbedded Visual C++ 3.0", Department of Electrical and Computer Engineering, University of Maryland at College park, diciembre 2003

Acrónimos

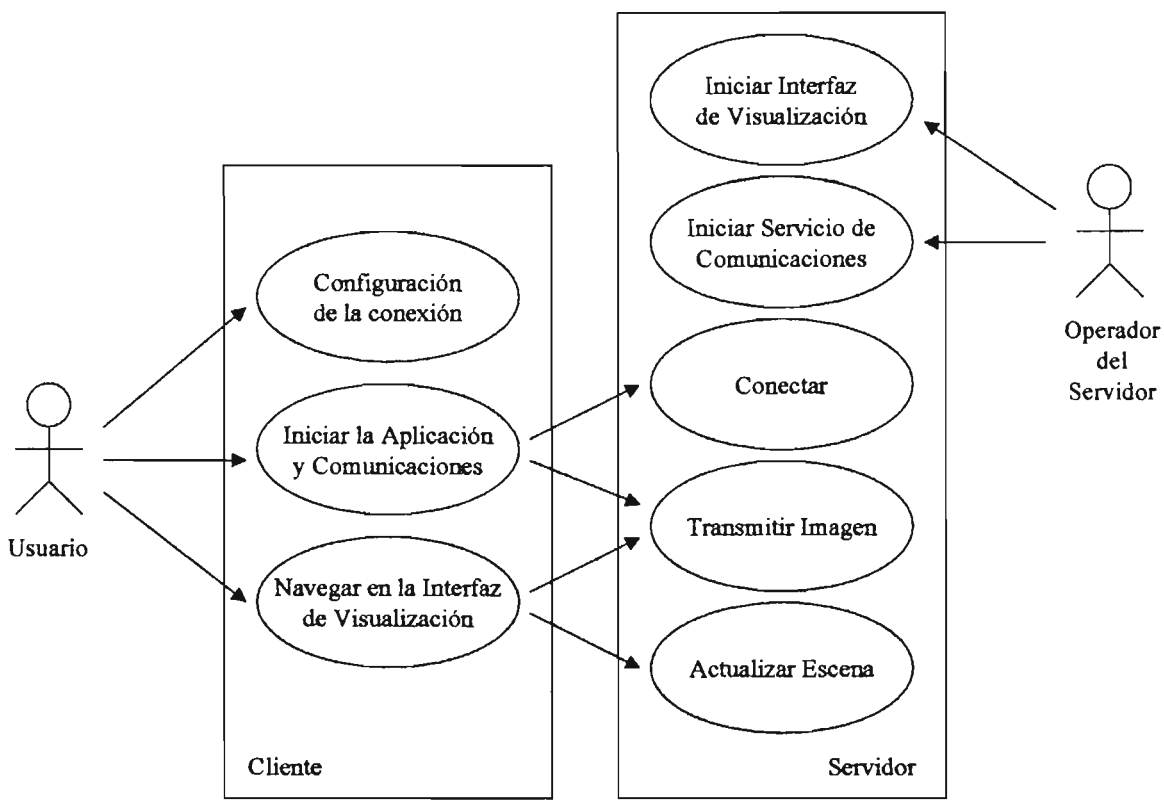
BCB	Borland C++ Builder
CDMA	Code Division Multiple Access
DSSS	Direct Sequence Spread Spectrum
EDGE	Enhanced Data rates for Global Evolution
EGL	Embbeded Graphics Library
EVC	eMbedded Visual C++
FHSS	Frequency Hopping Spread Spectrum
GDI	Graphics Device Interface
GFSK	Gaussian Frequency Shift Keying
GPRS	General Packet Radio System
GSM	Global System for Mobile Communications
HAL	Hardware Abtraction Layer
HSCSD	High Speed Circuit Switched Data
ICV	Integrity Check Value
IEEE	Institute of Electrical and Electronics Engineers
IrDA	Infrared Data Association
LAN	Local Area Network
OFDM	Orthogonal Frequency Division Multiplexing
OpenGL	Open Graphics Library
OpenGL ES	Open Graphics Library for Embbeded Systems
PDA	Personal Digital Assistant
PSK	Pre Shared Key
RAD	Rapid Application Development
RF	Radio Frequency
TDMA	Time Division Multiple Access
TKIP	Temporal Key Integrity Protocol
UMTS	Universal Mobile Telecomunications System
WCDMA	Wideband Code Division Multiple Access
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Fidelity
WLAN	Wireless LAN
WPA	Wi-Fi Protected Access

Anexo A: Documentación

A.1 Diagramas de casos de uso

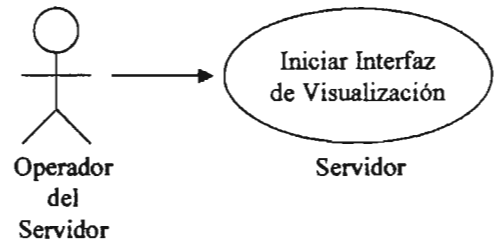
Diagramas de caso de uso que describen los requisitos funcionales de la aplicación.

Diagrama general



Caso de uso: Iniciar la Interfaz de Visualización

Actor: Operador del servidor



Descripción: El operador del servidor abre la ventana de la interfaz para visualización de la imagen

Flujo:

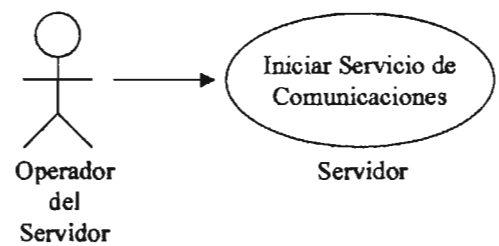
Usuario	Servidor	Excepción
1. Abre la ventana de interfaz de visualización	2. Inicializar librería gráfica	E1
	3. Crear objetos de la escena	
	4. Calcular imagen de la escena	
	5. Desplegar imagen de la escena en la ventana de interfaz de visualización	

Excepciones:

Id	Descripción	Acción
E1	Error de inicialización de librería gráfica	Reiniciar aplicación

Caso de uso: Iniciar Servicio de Comunicaciones

Actor: Operador del servidor



Descripción: El operador del servidor inicia el servicio de transmisión de imágenes.

Flujo:

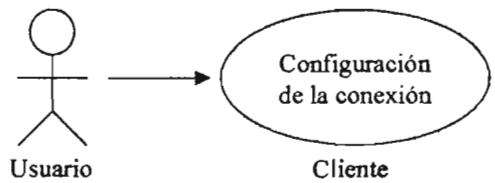
Usuario	Servidor	Excepción
1. Abre la ventana de control de conexión.	2. Obtiene la dirección IP del equipo	
	3. Muestra la dirección IP del equipo	
4. Captura valor del puerto		E1
5. Presiona el botón de inicio de servicio	6. Abre el puerto y queda en espera de la conexión del cliente.	

Excepciones:

Id	Descripción	Acción
E1	El usuario no captura valor del puerto	Usar el valor del puerto por omisión

Caso de uso: Configuración de la conexión

Actor: Usuario de la aplicación cliente



Descripción: El usuario captura los datos necesarios para conectarse con el servidor

Flujo:

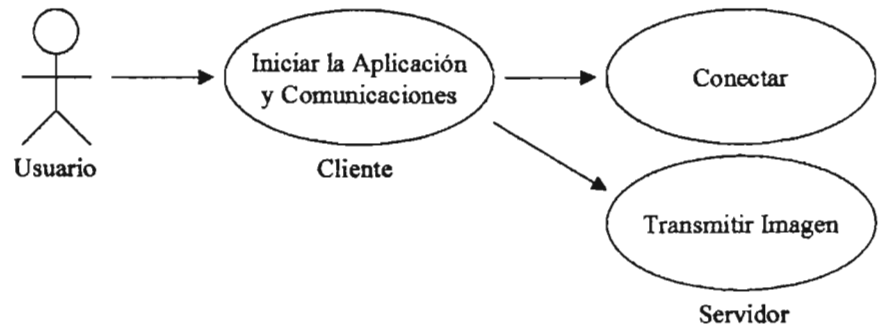
Usuario	Cliente	Excepción
1. Captura datos para la conexión	2. Registra datos	E1

Excepciones:

Id	Descripción	Acción
E1	Información incorrecta	Volver a capturar la información

Caso de uso: Iniciar la Aplicación y Comunicaciones

Actor: Usuario de la aplicación cliente



Descripción: El usuario inicia la aplicación cliente. La aplicación cliente se conecta con el servidor y obtiene la primera imagen de la escena.

Flujo:

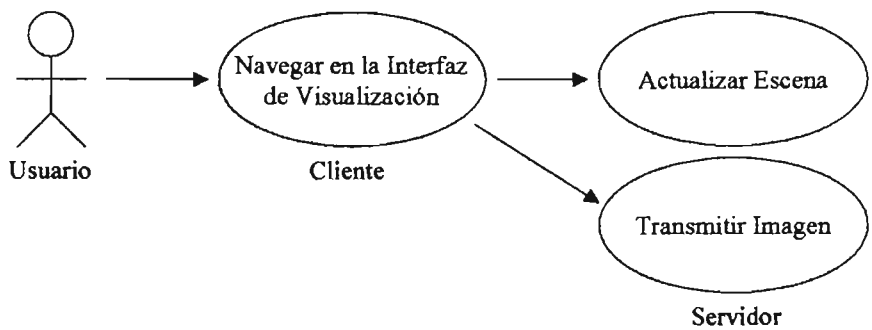
Usuario	Cliente	Servidor	Excepción
1. Inicia la aplicación Cliente Gráfico	2. Inicializa buffer de la imagen		
	3. Inicializa conexión		E1
	4. Conecta con el servidor	5. Acepta conexión	E2
	6. Envía comando ECO	7. Responde al comando	E3
	8. Envía tamaño de la interfaz	9. Inicializa buffer de datos de transferencia	E4, E5
		10. Envía comando de inicialización terminada	
	11. Envía solicitud de actualización de imagen	12. Recibe solicitud de actualización de imagen	
		13. Calcula imagen de la escena	
		14. Envía imagen al cliente	
	15. Recibe imagen		
	16. Muestra imagen en la interfaz		

Excepciones:

Id	Descripción	Acción
E1	No hay datos capturados de la dirección del servidor	- Capturar datos de la conexión - Reiniciar aplicación cliente
E2	No se logra la conexión porque el servidor no esta funcionando, esta fuera de servicio o la dirección es incorrecta	- Verificar datos de la conexión - Verificar el servidor - Reiniciar aplicación cliente
E3	El servidor no responde el comando ECO	Reiniciar aplicación cliente
E4	Los valores del tamaño de la interfaz son incorrectos	Reiniciar aplicación cliente
E5	La ventana de Interfaz del servidor no se encuentra inicializada para transmitir datos de la imagen	Reiniciar aplicación cliente

Caso de uso: Navegación en la Interfaz de Visualización

Actor: Usuario de la aplicación cliente



Descripción: El usuario navega por la escena, la aplicación cliente envía el evento al servidor que actualiza la imagen y la envía al cliente como respuesta al evento.

Flujo:

Usuario	Cliente	Servidor	Excepción
1. Navega por la escena	2. Envía el evento al servidor	3. Recibe el evento	E1
		4. Aplica el evento en la escena	
		5. Calcula imagen de la escena	
		6. Envía imagen al cliente	
	7. Recibe imagen		
	8. Muestra imagen en la interfaz		

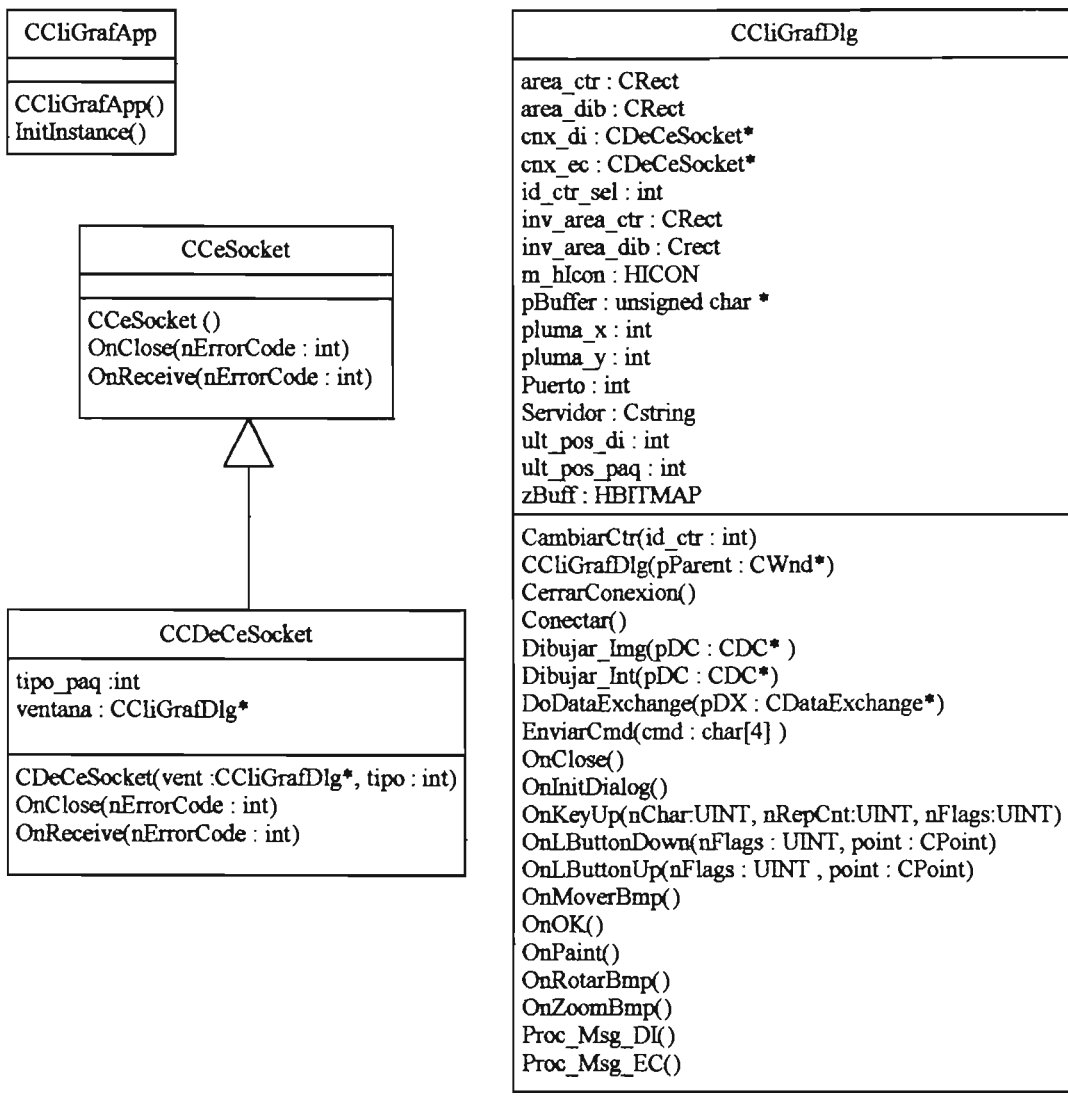
Excepciones:

Id	Descripción	Acción
E1	El evento no es reconocido por el servidor	Fin del proceso

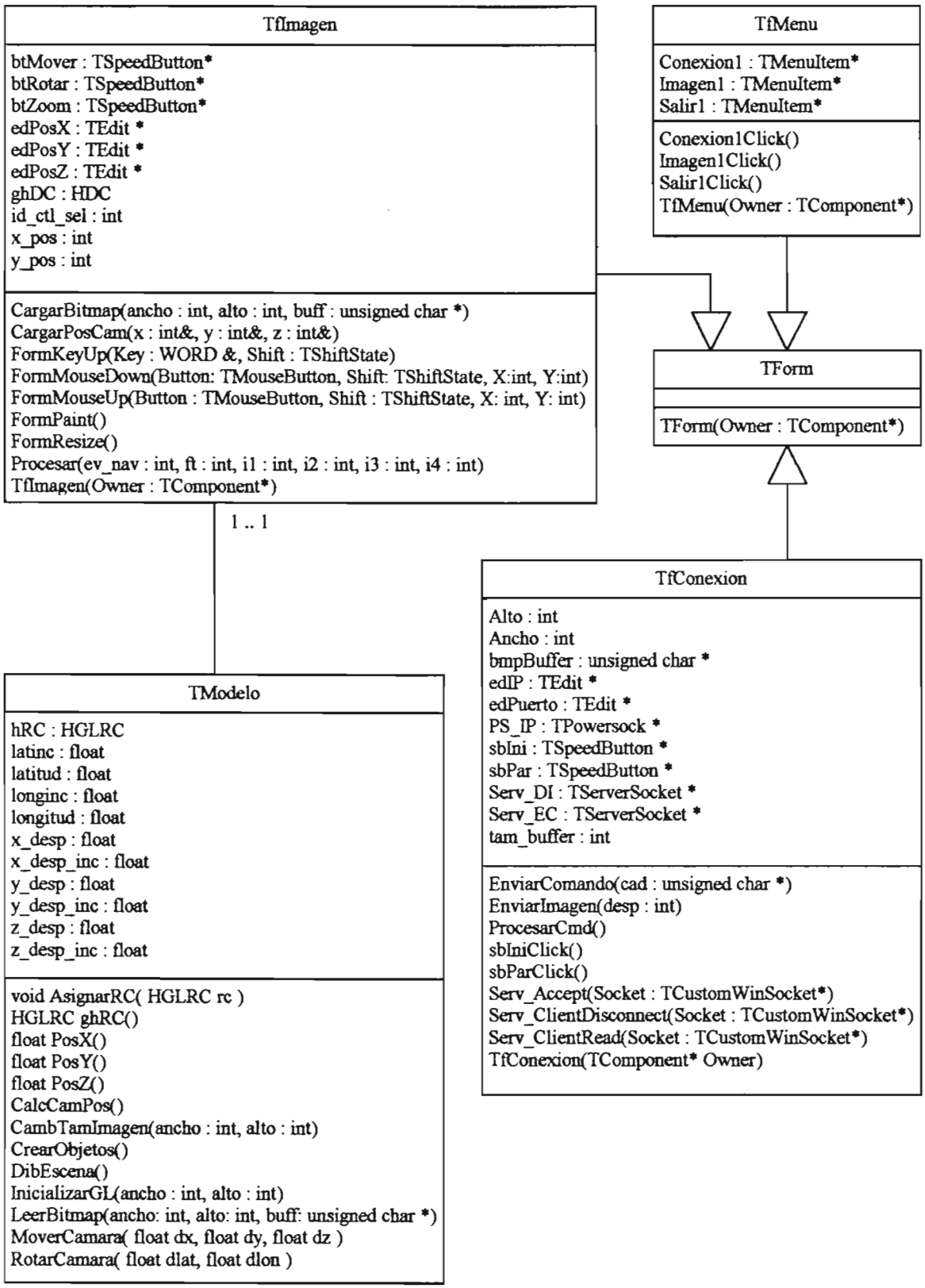
A.2 Diagramas de clases

Diagramas de las clases que conforman la aplicación.

Aplicación Cliente



Aplicación Servidor



A.3 Descripción de los paquetes de información

Estructura de los paquetes de eventos

Posición (bytes)	Parámetros	Descripción	Tipo	Longitud (bytes)
0	Comando	Comando que se envía en el paquete	unsigned char	3
3	Banderas	Banderas del comando	unsigned char	5
8	Parámetro 1	Parámetro 1 del comando	integer 32 bits	4
12	Parámetro 2	Parámetro 2 del comando	integer 32 bits	4
16	Parámetro 3	Parámetro 3 del comando	integer 32 bits	4
20	Parámetro 4	Parámetro 4 del comando	integer 32 bits	4
24	Tamaño Buffer	Longitud del buffer de datos adicionales del comando	integer 32 bits	4
28	Buffer de Datos	Buffer de datos adicionales del comando	unsigned char	996

Tamaño máximo 1,024 bytes, tamaño mínimo 28 bytes.

Estos paquetes pueden ser enviados por el servidor y por el cliente a través del puerto de eventos.

Comandos

000 – ECO
Banderas: No
Parámetros: No
Buffer: No

Este comando se utiliza para establecer el inicio de las comunicaciones. El servidor responde con el mismo comando. También puede ser utilizado para probar la conexión.

001 – Aviso de fin de conexión
Banderas: No
Parámetros: No
Buffer: No

El cliente envía este comando para avisar que va a cerrar la conexión

Nota: El cliente espera que el servidor envíe el comando 002 para cerrar la conexión.

002 – Aceptación de fin de conexión
Banderas: No
Parámetros: No
Buffer: No

El servidor envía este comando en respuesta al comando 001 para reconocer que el cliente va a cerrar la conexión.

100 – Descripción del área de la imagen del cliente
Banderas: No
Parámetros:
1 – Ancho del área de la imagen en píxeles
2 – Alto del área de la imagen del cliente en píxeles
Buffer: No

El cliente envía las dimensiones de la imagen que tiene en la interfaz. Estos datos se utilizan para construir el buffer de datos que se enviará en cada petición de imagen.

Nota: El servidor responderá con el comando 200 si el buffer fue creado correctamente o con los comandos de error 900 y 901 en caso contrario.

101 – Solicitud de imagen
Banderas: No
Parámetros: No
Buffer: No

El cliente envía este comando para solicitar la imagen de la escena actual. Este comando se utiliza normalmente para pedir la primera imagen.

Notas
El servidor enviará la escena por el puerto de datos de la imagen con el comando 201. El servidor también enviará un comando 400 por el puerto de eventos. El servidor puede enviar el comando 902 en caso de error.

Si el tamaño del buffer de la imagen es mayor que el buffer de datos, se envían los primeros 58,000 bytes de la imagen. Los bytes faltantes se deberán solicitar con el comando 102.

102 – Solicitud de fragmento de imagen
Banderas: No
Parámetros:
1 – Desplazamiento en el bitmap del fragmento
Buffer: No

Este comando complementa el comando 101. Cuando el buffer de la imagen es mayor que el buffer de datos, se envía este comando para solicitar los bytes faltantes para completar el buffer de la imagen. El parámetro 1 indica los bytes que se han recibido y corresponde con la posición desde la que se desea recibir el siguiente fragmento.

Notas
El servidor enviará la escena por el puerto de datos de la imagen con el comando 201. El servidor puede enviar el comando 902 en caso de error.

Si el valor del parámetro 1 es mayor que el tamaño del buffer de la imagen, el servidor no enviará los datos de la imagen ni comando de error.

Si la longitud del fragmento (calculada desde la posición que se solicita hasta el final del buffer de la imagen) es mayor que la longitud buffer del paquete, se enviarán 58,000 bytes a partir de la posición solicitada. Se deberán enviar más solicitudes mediante este comando hasta llenar el buffer de la imagen en el cliente.

200 – Buffer para imagen del cliente creado
Banderas: No
Parámetros: No
Buffer: No

El servidor envía este comando en respuesta al comando 100 para avisar al cliente que el buffer de datos de la imagen fue creado satisfactoriamente.

300 – Evento Mover
301 – Evento Rotar
302 – Evento Acercar/Alejar

Banderas:
1 – Tipo de control utilizado, posibles valores: [0, 1]
0 – Dispositivo señalador (Ratón o Pluma)
1 – Botones de navegación
Parámetros:
Si el valor de la bandera 1 = 0
1 – Posición Horizontal, posibles valores: [0 a 100]
2 – Posición Vertical, posibles valores: [0 a 100]
3 – Desplazamiento Horizontal, posibles valores: [0 a 100]
4 – Desplazamiento Vertical, posibles valores: [0 a 100]
Si el valor de la bandera 1 = 1
1 – Botón utilizado, posibles valores: [0 a 3]
0 – Botón Izquierdo
1 – Botón Arriba
2 – Botón Derecha
3 – Botón Abajo
Buffer: No

Estos comandos son enviados por el cliente como eventos de navegación en la interfaz.

Notas
El servidor enviará la escena por el puerto de datos de la imagen con el comando 201. El servidor también enviará un comando 400 por el puerto de eventos. El servidor puede enviar el comando 902 en caso de error.

El valor para la posición y desplazamiento del dispositivo señalador es la proporción respecto a la dimensiones de la imagen, de 0 a 100 por ciento. La posición (0,0) es la esquina inferior izquierda.

400 – Posición de la cámara
Banderas: No
1 – Posición en el eje X * 100.
2 – Posición en el eje Y * 100.
3 – Posición en el eje Z * 100.
Buffer: No

El servidor envía este comando para actualizar la información de la posición de la cámara en la interfaz del cliente como respuesta a los comandos 101, 300, 301 y 302. El valor debe dividirse entre 100 para obtener la posición en unidades de los objetos de la escena.

900 – Error: Modelo grafico no inicializado
Banderas: No
Parámetros: No
Buffer: No

Este mensaje de error se envía cuando aún no se ha inicializado la escena y se envía una petición para crear el buffer de datos de la imagen. Este mensaje aparece como respuesta al comando 100.

901 – Error: Datos del área de la imagen del cliente inválidos
Banderas: No
Parámetros: No
Buffer: No

Este mensaje de error se envía cuando el cliente proporciona valores incorrectos (negativos o cero) en las dimensiones. Este mensaje aparece como respuesta al comando 100.

902 – Error: No se ha creado el buffer de datos de la imagen
Banderas: No
Parámetros: No
Buffer: No

Este mensaje de error se envía cuando el servidor no pudo crear el buffer de datos de la imagen. Este mensaje aparece como respuesta a los comandos 101, 102, 300, 301 y 302.

Estructura de los paquetes de datos de la imagen

Posición (bytes)	Parámetros	Descripción	Tipo	Longitud (bytes)
0	Comando	Comando que se envía en el paquete	unsigned char	3
1	Bandera	Bandera del comando	unsigned char	1
4	Desplazamiento	Desplazamiento en memoria del bitmap	integer 32 bits	4
8	Tamaño Buffer	Longitud del buffer de datos adicionales del comando	integer 32 bits	4
12	Buffer de Datos	Buffer de datos adicionales del comando	unsigned char	58000

Tamaño máximo 58,012 bytes, tamaño mínimo 12 bytes.

Estos paquetes pueden ser enviados sólo por el servidor.

Comandos

201 – Datos de la imagen
Bandera: No
Desplazamiento: Si
Tamaño del Buffer: Si, posibles valores: [1 a 58,000]
Buffer: Si, información de la imagen.

Este comando es enviado por el servidor en el puerto de datos de la imagen como respuesta a los comandos 101, 102, 300, 301 y 302. En el buffer se envían los bytes de la imagen desde la posición de Desplazamiento, con la longitud indicada en el parámetro "Tamaño Buffer"

Nota

El valor del desplazamiento normalmente es cero excepto en la respuesta al comando 102, para este comando se envía la posición del bitmap solicitada.