REIHE INFORMATIK
14/94

Effiziente Verarbeitung von multimedialen Datenströmen in Window-Systemen

R. Keller und W. Effelsberg
Universität Mannheim
L15,16
D-68131 Mannheim

Effiziente Verarbeitung von multimedialen Datenströmen in Window-Systemen

Ralf Keller und Wolfgang Effelsberg
Praktische Informatik IV, Universität Mannheim
D-68131 Mannheim
{keller,effelsberg}@pi4.informatik.uni-mannheim.de

Zusammenfassung

Fensterorientierte Oberflächen haben sich auf Workstations aller Leistungsklassen durchgesetzt. Deshalb liegt es nahe, Multimedia-Anwendungen in solche Oberflächen zu integrieren. Dieser Artikel gibt zunächst eine Übersicht über die verschiedenen technischen Möglichkeiten zur Integration von Multimedia-Datenströmen in ein Fenstersystem; die Vor- und Nachteile der einzelnen Ansätze werden gegenübergestellt. Während Lösungen mit Hardware-Unterstützung im allgemeinen schneller sind, sind reine Software-Implementierungen flexibler und portabler. Als ein Beispiel für eine reine Software-Lösung werden Architektur, Implementierung und Leistungsanalyse eines netzwerkfähigen Filmsystems für das X-Window-System ausführlich diskutiert. Es zeigt sich, daß die Übertragung und Darstellung von digitalen Filmen auf modernen Workstations in Hochgeschwindigkeitsnetzen ohne spezielle Hardware in Realzeit möglich ist. Außerdem werden neue Ansätze zur Gestaltung der Mensch-Maschine-Schnittstelle mit multimedialen Komponenten vorgestellt.

1 Einführung

Fensterorientierte Oberflächen haben sich aufgrund ihrer Benutzerfreundlichkeit auf Workstations aller Leistungsklassen durchgesetzt. Gleichzeitig werden in immer stärkerem Maße multimediale Anwendungen entwickelt, die sowohl auf diskrete Medien, wie Text und Grafik, als auch auf kontinuierliche Medien, wie Audio und Video, zugreifen und diese dem Benutzer präsentieren. Deshalb liegt es nahe, eine Integration von Video in Fenstersysteme anzustreben.

Dabei müssen zwei wichtige Bedingungen beachtet werden. Erstens muß dafür gesorgt werden, daß die hohen Anforderungen von Videoströmen in bezug auf Datenrate und Isochronität erfüllt werden. Bei Desktop-Videokonferenzen akzeptiert ein Benutzer in der Regel ein relativ kleines Bildformat mit niedriger Bildwiederholfrequenz, da sich der Bildinhalt nur wenig ändert. Dies gilt jedoch nicht für andere Video-Anwendungen, wie z.B. Video-on-Demand. Zweitens soll der Benutzer mit den kontinuierlichen Medien einfach interagieren können. Meistens werden zur Steuerung von Audio- und Videoströmen Schaltflächen angeboten, die denen von Radios bzw. von Videorekordern gleichen. Diese können mit Maus und Tastatur bedient werden. Neuartige Anwendungen erfordern aber auch neuartige Interaktionsmöglichkeiten. So wie sich viele Computerspiele nur mit einem Joystick optimal bedienen lassen, so lassen sich z.B. computergenerierte virtuelle Welten nur mit neuartigen Mensch-Maschinen-Schnittstellen, wie speziellen Helmen und Handschuhen, „richtig“ bereisen.

Bei unseren Betrachtungen konzentrieren wir uns auf das X-Window-System [15] (im folgenden auch kurz X genannt), den de-facto Standard für Fenstersysteme auf Workstations. Dieses hat gegenüber anderen Fenstersystemen den entscheidenden Vorteil, daß der Quellcode frei verfügbar ist. Somit ist es leicht möglich, Erweiterungen in X einzubringen.

Dieser Artikel gliedert sich wie folgt. Kapitel 2 enthält eine Übersicht über die verschiedenen technischen Möglichkeiten zur Integration von Video-Fenstern in ein Fenstersystem; die Vor-

und Nachteile der einzelnen Ansätze werden gegenübergestellt. Zwei Beispiele für spezielle Video-Hardware werden in Kapitel 3 vorgestellt. Als ein Beispiel für eine reine Software-Lösung werden Architektur, Implementierung und Leistungsanalyse von XMovie in Kapitel 4 ausführlich diskutiert. XMovie ist ein digitales Filmsystem, das an der Universität Mannheim entwickelt wurde. In Kapitel 5 gehen wir kurz auf die Unterstützung von Audio in multimedialen Anwendungen ein. Neue Ansätze zur Interaktion von Benutzern mit Audio und Video werden in Kapitel 6 vorgestellt. Eine Zusammenfassung bildet den Abschluß dieses Artikels.

2 Prinzipielle Möglichkeiten für Video-Fenster

Dem Benutzer einer Workstation sollte ein Video in einem Fenster auf dem Bildschirm angezeigt werden, ohne daß andere Fenster gestört werden. Ein Video-Fenster sollte in ähnlicher Art und Weise wie ein anderes Fenster benutzt werden können. Echte Video-Fenster werden noch nicht von aktuellen Fenstersystemen, wie z.B. dem X-Window-System, unterstützt.

Eine Integration von Video-Fenstern in ein Fenstersystem kann nach einem der drei folgenden prinzipiellen Ansätze durchgeführt werden:

1. Spezielle Video-Hardware wird in die Workstation eingebaut (z.B. nach dem *blue-box*-Verfahren oder zur Echtzeitdigitalisierung von analogen Videoquellen); Beispiele sind Pandora's Box [4], DVI [3] oder typische Video-Karten für den PC. Die Spezialhardware schränkt jedoch die Flexibilität ein und macht die Integration mit anderen Fenstern schwierig. Die Anzahl der offenen Video-Fenster ist beschränkt, und es ist ein spezieller Treiber erforderlich. Der digitale Video-Strom fließt nicht über den Systembus, sondern direkt von der (lokalen) Quelle zum Bildschirm. Eine Verarbeitung des Videostroms im Rechner ist nicht möglich.

2. Die Software des Fenstersystems ermöglicht die Anzeige einzelner digitaler Bilder. Diese Fähigkeit wird dazu verwendet, um Folgen von digitalen Bildern schnell darzustellen; das Fenstersystem stellt eine Folge von Einzelbildern so schnell wie möglich dar. Beispiele für diese Vorgehensweise sind der MPEG-Player der UC Berkeley [13], das INRIA Videokonferenzsystem IVS [16] und nv [1]. Dabei erfolgt die Dekodierung entweder vollständig in Software oder durch Aufruf einer Einzelbilddekodierkarte (z.B. für JPEG [14]). Der Vorteil ist dabei, daß eventuell vorhandene Hardware zur Leistungssteigerung eingesetzt werden kann, ohne daß die Anwendung dies bemerkt. Die Hauptnachteile dieses Ansatzes sind, daß die Synchronisation mit der Realzeit in der Anwendung durchgeführt werden muß, daß die Abspielanwendung und das Fenstersystem auf derselben Maschine laufen müssen, um die höchste Geschwindigkeit zu erreichen und daß das Abspielen mehrerer Filme gleichzeitig einen beträchtlichen Overhead in der Anwendung erzeugt.

3. Video-Fenster werden durch eine Erweiterung in das Fenstersystem integriert. Das Fenstersystem versteht ein Video-Fenster als einen neuen Fenstertyp und kann insbesondere die Realzeitanforderungen optimal berücksichtigen. Auch hier kann durch Aufruf einer eventuell vorhandenen Dekodierkarte eine Leistungssteigerung ohne Veränderung für die Anwendung erreicht werden.

Da wir uns nicht von spezieller Hardware abhängig machen wollen, kommen für uns nur die beiden letzten Ansätze in Frage. In beiden Fällen fließt der digitale Video-Strom über Hauptspeicher und Systembus und kann deshalb vom Rechner bearbeitet werden, wobei der dritte Ansatz die beste Performance für einen Software-Player bietet.

Im Fall des X-Window-System bedeutet dies, daß der X-Server die volle Kontrolle über das Video-Fenster besitzt. Der X-Server kann mehrere Filmaufträge von Klienten gleichzeitig bearbeiten. Für jeden Auftrag liest er einen Strom von Bildern entweder vom Netz oder aus einer Warteschlange im gemeinsamen Speicher.

Erweiterungen des Fenstersystems, hier am Beispiel von X, besitzen eine Reihe von Vorteilen gegenüber dem zweiten Ansatz:

- Der Kommunikationsaufwand zwischen dem X-Client und dem X-Server ist niedriger.
- Der Kode für die Video-Darstellung, d.h. für die Darstellung von Bildfolgen, muß nicht in jede Abspielanwendung integriert werden, sondern ist Bestandteil des X-Servers.
- Der X-Server kann besser an die Hardware der Workstation angepaßt werden, so daß die Leistung und damit die Qualität verbessert wird.
- Die Manipulation des Video-Fensters (Öffnen, Schließen, Zoom, ...) geschieht sowohl für den Anwender als auch für den Programmierer, wie aus anderen Fensteranwendungen gewohnt.

Ein Nachteil des dritten Ansatzes ist allerdings, daß in den Code des X-Serves eingegriffen werden muß, während der zweite Ansatz als reine Client-Anwendung programmiert werden kann.

In den nächsten zwei Kapiteln werden wir zunächst zwei Beispiele für Spezialhardware vorstellen, um dann vertieft auf ein Beispiel für eine reine Softwarelösung einzugehen.

3 Beispiele für hardwaregestützte Lösungen

Zur Analog/Digital-Wandlung von Videoströmen werden spezielle Videokarten benötigt. Die im folgenden beschriebenen Karten dienen zur Digitalisierung von Videoströmen, die dann digital weiterverarbeitet werden (Schneiden, Zoomen, Kodieren, ...).

3.1 Parallax XVideo

Die XVideo-Karte der Firma Parallax ist eine reine Videokarte ohne Audio. Sie ist für Sun SPARCstations[1] und für Workstations der Firma Hewlett Packard verfügbar [12]. Sie kombiniert Kodierung/Dekodierung und Grafikadapter. Die Karte erlaubt je nach Ausstattung bis zu zwei analoge Ein- und Ausgänge, d.h., maximal können zwei verschiedene Videos gleichzeitig digitalisiert und angezeigt werden. Ein (optionaler) Chip der Firma C-Cube Microsystems dient zur JPEG-Kompression von Einzelbildern. Gleichzeitig fungiert die Karte als Echtfarben-Grafikadapter, d.h., analoge Videoströmen können in Echtzeit digitalisiert und in einem X-Fenster angezeigt werden, wobei der digitale Videostrom auf der Karte vom Digitalisierer direkt zum Bildspeicher übertragen wird. Der Zugriff auf diese Karte erfolgt über eine X-Erweiterung der Firma Parallax.

Sowohl komprimierte als auch unkomprimierte Einzelbilder können von der Karte angefordert und auch an die Karte geschickt werden. Die Schnittstelle verarbeitet nur Einzelbilder und keine Videoströme. Die Vorgehensweise bei der Programmierung einer Abspielanwendung entspricht daher dem oben beschriebenen zweiten Ansatz mit Hardware-Unterstützung. Typische Vertreter dieser Anwendungsklasse sind JPEG-basierte Lösungen.

3.2 DEC Sound and Motion J300

Die *Sound and Motion J300*-Karte der Firma DEC ist eine kombinierte Audio- und Videokarte. Sie ist für DEC Alpha AXP verfügbar [2].[2] Die Karte besitzt einen analogen Video-Eingang, einen analogen Video-Ausgang und analoge und digitale Ein- und Ausgänge für Audioströme, d.h., maximal kann nur ein Video digitalisiert und angezeigt werden. Serienmäßig ist ein Chip zur JPEG-Kompression auf der Karte vorhanden. Die Karte fungiert nicht als Grafikadapter; daher muß der digitalisierte Videostrom zur Anzeige in einem X-Fenster mindestens zweimal über den Systembus transportiert werden (von der Karte zur Anwendung und von der Anwendung zum X-Server, der die Grafikkarte verwaltet). Der Zugriff auf diese Karte erfolgt über eine C-Schnittstelle, die unabhängig von der X-Schnittstelle ist. Es können wie bei der XVideo-Karte sowohl JPEG-komprimierte als auch unkomprimierte Einzelbilder von der Karte angefordert und auch an die Karte geschickt werden.

[1] Die SPARCstations verfügen serienmäßig über einen Stereo-Audioprozessor.
[2] Die AXPs verfügen serienmäßig nur über einen Mono-Audioprozessor für Sprache.

4 Beispiel für eine reine Softwarelösung: XMovie

An der Universität Mannheim wird im Rahmen des XMovie-Projekts die digitale Filmübertragung als eine innovative Anwendung in Rechnernetzen untersucht [7]. XMovie definiert eine Architektur und Protokolle für ein verteiltes Multimedia-System, bestehend aus vernetzten heterogenen UNIX-Workstations. Auf Platten gespeicherte Filme können über Hochgeschwindigkeitsnetze übertragen und in X-Fenstern angezeigt werden. Gegenwärtig verwenden wir einen FDDI-Ring zur Übertragung von Filmen. Seit dem Start des XMovie-Projekts wurden zwei Prototypen entwickelt, die jeder für sich eine eigene Variante des in Kapitel 2 beschriebenen dritten Ansatzes zur Integration von Video in Fenstersysteme realisieren. Im folgenden beschreiben wir zunächst generell die Funktionseinheiten von XMovie und die Erweiterungen, die in das X-Window-System eingebracht werden mußten, um dann vertieft auf die Realisierung der Videodarstellung im X-Server einzugehen.

4.1 Funktionseinheiten von XMovie

Die XMovie-Architektur unterscheidet vier Funktionseinheiten, die in beiden Prototypen vorhanden sind: CM-Server, CM-Client, CM-Agent und die Abspielanwendung. Der CM-Server (*Continuous Media*-Server) verwaltet Filme auf Massenspeichern und sendet sie auf Anfrage über das Netz. Der CM-Client empfängt Filme, dekodiert sie falls nötig und übergibt Einzelbilder an den CM-Agent zur Darstellung. Außerdem überwacht er die Isochronität und das Einhalten der Bildrate. Die Abspielanwendung (*Playback Application*) fordert Filme an, öffnet sie und steuert den Filmlauf wie bei einem Videorekorder.

Die Programmierschnittstellen für Anwendungen sind an die von X bekannten Schnittstellen angelehnt, so daß sie leicht zu erlernen und zu benutzen sind. Die Basisobjekte, die der Programmierer benutzt, sind einzelne Ströme, die er auswählen, starten, unterbrechen und beenden kann.

4.2 Einbettung der Erweiterung in X

Um beide Varianten zur Videodarstellung in X einzufügen, mußten jeweils Erweiterungen in der Xlib, im X-Protokoll und im X-Server durchgeführt werden (siehe Abb. 1).

In die Xlib wurden neue Stub-Routinen eingefügt, um die Anfragen an den X-Server zu senden und neuen Antworten zu verarbeiten. Für die Verwaltung der Erweiterung, insbesondere für die neuen Ereignis- und Fehlermeldungen, mußte ebenfalls gesorgt werden.

In das X-Protokoll wurden neue Protokolldateneinheiten (PDUs) eingefügt, die zwischen dem X-Server und einem X-Client, der eine der beiden Erweiterungen anspricht, ausgetauscht werden.

Der größte Teil der Implementierung steckt bei beiden Varianten im X-Server. Hier mußte dafür gesorgt werden, daß die neuen PDUs des X-Protokolls richtig empfangen und verarbeitet bzw. gesendet werden. Außerdem mußte die Verwaltung der Aufträge geregelt und die zur Videoverarbeitung notwendige Funktionalität integriert werden.

4.3 Erweiterungen des X-Servers

Um die Arbeitsweise der beiden Erweiterungen zu verstehen und ihre Vor- und Nachteile einschätzen zu können, muß zunächst die Funktionsweise des X-Servers kurz vorgestellt werden.

4.3.1 Funktionsweise des X-Servers

Das X-Server-Programm besteht im wesentlichen aus zwei ineinander geschachtelten Endlosschleifen, der *Main-* und der *Dispatch-Schleife*. In der Main-Schleife findet die Initialisierung des X-Servers und die Allokation von benötigten Ressourcen statt. Die eigentliche Arbeit des X-Servers wird in der Dispatch-Schleife durchgeführt. Hier wird die Illusion von Multitasking zwischen den Fenstern erzeugt, obwohl der X-Server aus der Sicht des Betriebssystems nur ein Prozeß

ist. Jedesmal, wenn der Prozessor diese Schleife abarbeitet, werden die gesammelten Eingabe-Ereignismeldungen von den Eingabegeräten an die X-Clients gesendet und die Anforderungen der X-Clients verarbeitet, wobei die Verarbeitung von Ereignismeldungen und Anforderungen für verschiedene Clients miteinander verzahnt werden können.

4.3.2 Bildfolgen direkt aus dem Netz

In der als erstes realisierten Variante einer Integration von Video-Fenstern wurden die von einem CM-Server gesendeten Einzelbilder vom X-Server direkt vom Netz gelesen und dargestellt. Der X-Server ist somit gleichzeitig CM-Client und CM-Agent, und die Abspielanwendung steuert den Filmablauf (siehe Abb. 2 und [8]).

In jedem Durchgang der Dispatch-Schleife liest der X-Server für jeden aktiven Auftrag einen Bildteil vom Netz und stellt ihn dar. Es findet also keine Pufferung statt. Die Bilder müssen in Teilen übertragen werden, da nur sehr kleine Bilder in einem Datenpaket übertragen werden können. Der X-Server darf auch nur einen Teil eines Bildes pro Durchgang verarbeiten, da das Warten auf die restlichen Teile den normalen Ablauf des Systems stören würde. Der X-Server kontrolliert die Einhaltung der Anforderungen der einzelnen Filme, insbesondere die Bildrate und die Isochronität.

Die erste Version unterstützt nur ein einziges Filmformat, das nicht dekodiert werden muß. Software-Dekoder wurden daher von uns nicht in den CM-Client (= X-Server) integriert.

4.3.3 Bildfolgen über gemeinsamen Speicher

In der zweiten Variante sind die Funktionseinheiten CM-Client und CM-Agent voneinander getrennt. Der X-Server arbeitet in dieser Version nur als CM-Agent und liest ganze Bilder aus dem mit dem CM-Client gemeinsamen Speicher (*Shared Memory*, SHM) (siehe Abb. 3). Dazu werden im SHM zwei Warteschlangen aus Bildpuffern aufgebaut: eine Warteschlange mit belegten Puffern zur Abarbeitung im X-Server und eine Warteschlange von freien Puffern zur Belegung durch den CM-Client, der nun den digitalen Videostrom vom Netz liest. Eine genauere Beschreibung der Warteschlangen enthält [6].

Durch die Trennung der Funktionseinheiten CM-Client und CM-Agent wird auch die Einbettung von Software-Dekodern in das System erleichtert. Diese können jetzt einfach in den CM-Client integriert werden: der CM-Client liest den komprimierten Videostrom vom Netz und schreibt die dekodierten Einzelbilder in die SHM-Warteschlange.

Eine andere Alternative, nämlich Software-Dekoder in den X-Server einzubetten, hat zwei Nachteile und wurde daher nicht realisiert. Erstens ist die Dekompression in Software sehr aufwendig und würde somit andere X-Clients stören. Zweitens schreitet die Entwicklung auf dem Gebiet der Software-Dekoder sehr rasch voran, und die Dekoder sind in einer vom X-Server getrennten Funktionseinheit leichter auszutauschen.

Die Dekodierung der Einzelbilder eines Videostromes ist in der Regel unterschiedlich aufwendig (z.B. bei den I-, P- und B-Bildern von MPEG [9]); der dadurch erzeugte Jitter kann in der Warteschlange herausgefiltert werden, da CM-Client und CM-Agent unabhängig voneinander arbeiten und der CM-Agent die Einzelbilder mit einer festen Rate aus der Warteschlange liest.

Dadurch ergeben sich zusätzlich zu den bereits in Kapitel 2 genannten zwei weitere Vorteile für die Erweiterung von Fenstersystemen:

- Jitter, der dem CM-Strom bei der Übertragung, Dekodierung oder anderen Funktionen hinzugefügt wurde, kann sehr leicht in der Shared-Memory-Warteschlange entfernt werden.
- Der CM-Client kann durch Überwachung des Füllstandes der Shared-Memory-Warteschlange eine Flußkontrolle durchführen und damit den CM-Server regulieren [6].

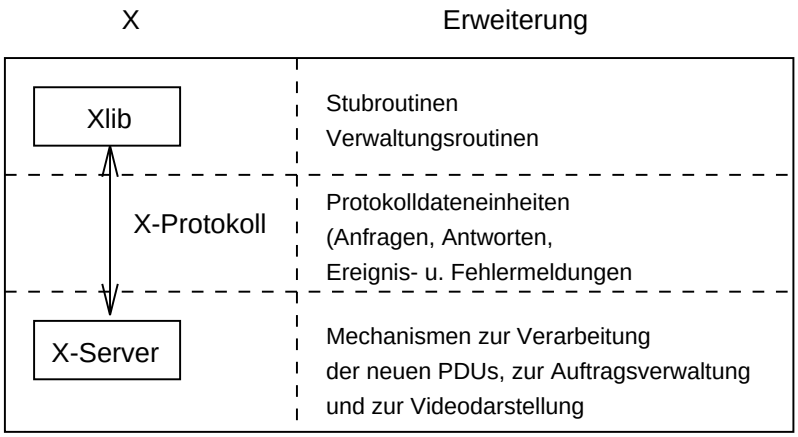

Abbildung 1: Erweiterung von X

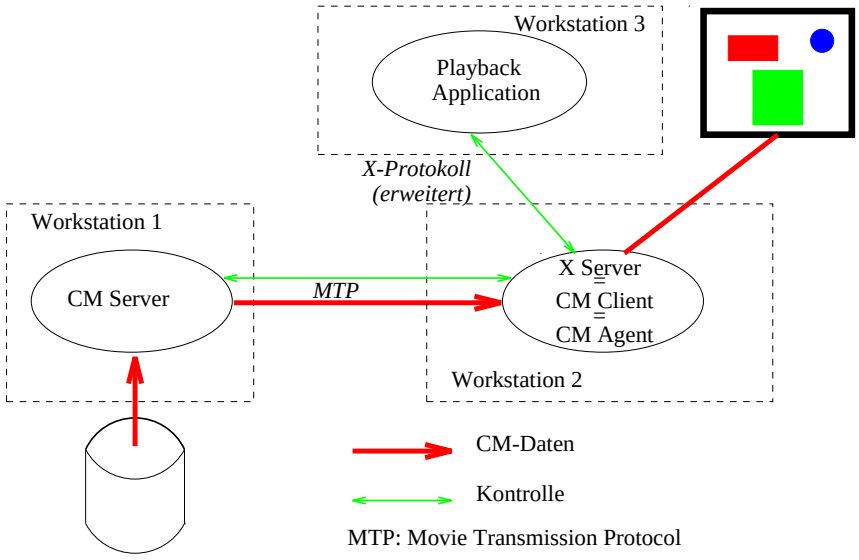

Abbildung 2: Bildfolgen direkt aus dem Netz

4.4 Erfahrungen

Die zuerst realisierte Variante brachte aufgrund ihrer einfachen Struktur sehr schnell akzeptable Ergebnisse.[3] Auf einer SPARCstation 10 erreichten wir bei einer Bildgröße von 160×128 eine Bildwiederholrate von 25 Bildern pro Sekunde. Unsere Messungen haben dabei ergeben, daß die Maschine nicht voll ausgelastet war, die Bildrate sich aber nicht weiter steigern ließ. Durch das Lesen von einzelnen Bildteilen vom Netz muß der X-Server bei 25 Bildern pro Sekunde und bei 4 Paketen pro Bild 100 mal pro Sekunde für jeden Film auf das Netz zugreifen. Dabei wird ein bestimmter Betriebsystemaufruf verwendet, der Ereignisse periodisch abfragt; die dabei bereitgestellte Mindestperiode verhindert ein schnelleres Abspielen.

Der X-Server ist daraufhin optimiert, Grafikoperationen schnell auszuführen. Durch die Bereitstellung von kompletten Bildern im SHM wird der X-Server stark entlastet, da er nun nur noch 25 mal pro Sekunde für einen Auftrag aktiv werden muß und diese Operation selbst bei größeren Formaten sehr schnell ausführt. Erste Messungen haben ergeben, daß auf der SPARCstation bis zu einer Bildgröße von 320×240 noch 100 Bilder pro Sekunde dargestellt werden können; bei 640×480 sind es immer noch 25 Bilder pro Sekunde, d.h., erst bei größere Bildformaten ist nicht mehr die oben erwähnte Betriebssystemroutine der Engpaß, sondern die eigentliche Bilddarstellung.

Weiterhin haben unsere Erfahrungen gezeigt, daß mit unseren X-Erweiterungen für Video-Fenster die Integration von Video als Datentyp in ereignisgesteuerten Anwendungen wesentlich vereinfacht wird. Ereignisgesteuerte Anwendungen wie die *Playback Application*, die die Benutzerschnittstelle und andere höhere Programmfunktionen unterstützen, müssen auf spontane Ereignisse reagieren. Sie werden normalerweise flexibel strukturiert und nicht auf Leistung hin optimiert. Demgegenüber werden mediumgesteuerte Anwendungen, in denen der größte Teil der Verarbeitung abläuft, auf Leistung hin optimiert. Der CM-Client und der CM-Server sind Beispiele für mediumgesteuerte Anwendungen; ihre Kontrollstrukturen sind ganz auf die isochrone Verarbeitung von Video und Audio ausgerichtet, d.h., mit erster Priorität wird das jeweilige Medium zeitgerecht verarbeitet. Die Verarbeitung von Steueranweisungen wird in der Regel verzögert.

5 Audio-Unterstützung

Neben Video ist Audio ein weiteres kontinuierliches Medium, auf das in multimedialen Anwendungen zugegriffen bzw. das verarbeitet wird. Audioströme benötigen zwar sowohl komprimiert als auch unkomprimiert sehr viel weniger Bandbreite als entsprechende Videoströme, dafür sind die Anforderungen an die Isochronität, an die Verzögerung und an die Fehlerhäufigkeit wesentlich härter, da das menschliche Ohr viel empfindlicher auf Störungen reagiert als das Auge.

Zum Abspielen und zum Aufnehmen von Audioströmen wird Spezialhardware zur Analog/Digital-Wandlung benötigt. Diese gehört aber in modernen Workstation zunehmend zur Standardausstattung bzw. ist relativ kostengünstig zu beschaffen. Da die jeweilige Schnittstelle zur Audiohardware in der Regel herstellerspezifisch ist, ist es wünschenswert, ähnlich wie bei der Grafik unter X, mit einem netzwerktransparenten Audiosystem eine einheitliche Schnittstelle für Audioströme zur Verfügung zu haben. Ein Beispiel für ein solches netzwerktransparentes Audiosystem ist AudioFile [10], das bereits für eine große Zahl von Workstations verfügbar ist. Mit AudioFile lassen sich Audioanwendungen viel portabler schreiben, da auf Eigenheiten der Hardware innerhalb der Anwendung nicht mehr eingegangen werden muß.

Im Prinzip muß bei einer separaten Darstellung von Audioströmen das Problem der Synchronisation mit den zugehörigen Videoströmen gelöst werden. Wir sind allerdings der Meinung, daß in den meisten praktischen Anwendungen die Audio- und Videoströme auf dem Quellsystem verschränkt werden und dann gemeinsam übertragen werden. Ein Beispiel sind MPEG-Systemströme [5]. Deshalb muß beim Abspielen auf dem Zielsystem, nach dem Zeitpunkt der Trennung der Audio- und Videoströme, nur noch die Synchronität bei der tatsächlichen Wiedergabe gewährleistet sein; dies ist ein lokales Problem, daß nicht im Kommunikationssubsystem gelöst werden kann.

[3] Die erste Implementierung wurde bereits 1992 auf einer IBM PS/2 Mod. 80 unter AIX durchgeführt.

Erste Erfahrungen mit AudioFile in XMovie zeigten, daß bei der Übertragung und beim Abspielen von Audioströmen in verteilten System mit dem Einsatz von AudioFile eine bessere Qualität als ohne AudioFile erreicht wird. AudioFile kennt die Eigenheiten der verwendeten Audiohardware, z.B. wieviele Samples eines Stromes in der Karte gepuffert werden, und paßt das Abspielen optimal an, so daß es nicht mehr zu störenden Unterbrechungen kommt.

6 Neue Metaphern für die Mensch-Maschine-Schnittstelle

Bisher haben wir Verfahren für die Einbettung von Video- und Audioströmen in traditionelle Benutzeroberflächen beschrieben. Bekanntlich haben sich hier seit Anfang der achtziger Jahre die fensterorientierten Oberflächen durchgesetzt, sowohl auf Workstations als auch auf dem PC. Sie verwenden hochauflösenden Pixel-Grafik, meist in Farbe, und die Maus als Zeigegerät.

Die fensterbasierten Oberflächen haben sich als außerordentlich benutzerfreundlich erwiesen und vielen Computerlaien den Zugang zum Rechner überhaupt erst ermöglicht. Dieser Erfolg ist entscheidend durch das gewählte *Interaktionsparadigma* bedingt: die *Schreibtisch-Oberfläche* (*desktop metaphor*). Jedes Fenster und jedes Symbol auf dem Bildschirm entspricht einem Objekt aus der „Schreibtisch-Welt".

Nun hat aber die Schreibtisch-Welt keine multimedialen Komponenten; weder Audio noch Video passen sich in natürlicher Weise in die Metapher der Schreibtisch-Oberfläche ein. Deshalb erfordert die Integration der kontinuierlichen Medien grundsätzliche Neuüberlegungen zum Interaktionsparadigma. Eine wirklich überzeugende Idee gibt es hierzu noch nicht, aber zwei Denkansätze sind in diesem Zusammenhang erwähnenswert.

6.1 Virtuelle Realität

Computergenerierte virtuelle Welten (*Virtual Reality*, VR) bauen von Anfang an auf den Gesichts- und Gehörsinn des Menschen. Der Benutzer soll sich in der virtuellen Welt ohne „Interaktions-Krücken" wie Tastatur oder Maus bewegen. Dazu trägt er einen VR-Helm, in den die Stereo-Bildschirme und ein Positions- und Richtungsmelder für sechs Freiheitsgrade eingebaut sind [11].

In den heutigen Anwendungen der virtuellen Realität werden computergenerierte Kunstwelten verwendet, beispielsweise in der Innenarchitektur oder in der Flugsimulation. Interaktive Videos, die in der Multimedia-Forschung dominieren, kommen kaum zum Einsatz. Es wäre durchaus interessant, die Techniken der Virtuellen Realität auch für interaktive Multimedia-Systeme zu erproben.

Allerdings sind auch hier noch viele Fragen der Mensch-Maschine-Schnittstelle unbeantwortet. So ist es beispielsweise sehr unnatürlich, während eines dreidimensional dargestellten virtuellen Flugs durch den Grand Canyon durch einen scharf gekrümmten Zeigefinger im Datenhandschuh ein Pull-Down-Menü zu aktivieren, das dann aus heiterem Himmel herabsteigt. Für Video- und Audioströme im VR-Helm ist vollkommen ungeklärt, welche Metapher hier einer natürlichen Interaktion zugrunde gelegt werden sollte.

Leider ist die Technik der Virtuellen Realität mit ungemein lästiger Hardware verbunden: Helm, Datenhandschuh oder Datenanzug, Verkabelung usw. Dem Benutzer wird also stets aufs Neue bewußt gemacht, daß er jetzt mit einem Computer in „natürlicher" Weise interagieren soll. Hierin liegt ein immanenter Widerspruch. Einen radikal anderen Ansatz verfolgen deshalb die Vertreter des *ubiquitous computing*.

6.2 Computer überall und unauffällig

Die Forscher des Xerox Palo Alto Research Center (PARC) haben den Begriff des *ubiquitous computing* geprägt [17]. Sie meinen damit eine unauffällige Präsenz einer Vielzahl von Rechnern in der Umgebung des Benutzers. So wie ein guter Kugelschreiber sich dadurch auszeichnet, daß man ihn beim Schreiben nicht spürt, so sollen die Rechner als Mittel zum Zweck stets bereitstehen, ohne die Aufmerksamkeit des Benutzers zu sehr auf sich zu ziehen.

Eine Verkabelung all dieser Rechner ist natürlich nicht möglich, und so basiert die Kommunikation zwischen den Rechnern auf Funk oder auf Infrarot-Übertragung. Die Rechner können sehr klein und einfach sein, zum Beispiel aktive Namensschilder (*active badges*), die den aktuellen Aufenthaltsort des Trägers an das System melden, oder aufklebbare Etiketten, die zum Beispiel für Bücher verwendet werden und das Auffinden eines Buches irgendwo im Labor ermöglichen. Sie können auch die Größe eines Schreibblocks haben, der auch Audio-Eingabe und -Ausgabe unterstützt; sie dienen als multimediales Kommunikationsendgerät im Bürobereich. Und natürlich können auch Rechner der heute verbreiteten Größenklasse in das System einbezogen werden.

Die Benutzung der Rechner im *ubiquitous computing* erfolgt so natürlich und unauffällig wie möglich. Wenn man auf eine Anzeige schaut, bemerkt man nicht, daß sich dahinter ein Rechner verbirgt, der mit anderen Rechnern vernetzt ist. Das Interaktions-Paradigma ist hier die weitgehende Anpassung von vielen Arten von Computern mit audiovisuellen Komponenten und in den verschiedensten Größen und Erscheinungsformen an die gewohnte Umgebung des Benutzers.

Was nun die Darstellung von Audio und Video in einem solchen Szenario angeht, so sind die größten noch zu lösenden Probleme die zu niedrigen Datenraten auf den Funk- oder Infrarotstrecken und der zu hohe Energieverbrauch der mobilen Geräte. Beim heutigen Stand der Technik lassen sich hochauflösende Videos noch nicht auf Endgeräten des *ubiquitous computing* darstellen.

7 Zusammenfassung

Wir haben zunächst die prinzipiellen Möglichkeiten der Darstellung von Video in Fenstersystemen beschrieben und verglichen. Als zwei Beispiele für hardwaregestützte Lösungen habe wir Video-Karten der Firmen Parallax und Digital Equipment vorgestellt.

Ausführlicher sind wir dann auf Entwurf und Implementierung einer Softwarelösung eingegangen, die das Abspielen von Videos auf Workstations ohne spezielle Hardware ermöglicht. Es ist mit den Workstations der neuesten Generation problemlos möglich, Videos in guter Qualität und in Realzeit mit reinen Software-Playern abzuspielen.

In einem kurzen Kapitel haben wir dann moderne Abspieltechniken für Audio vorgestellt. Die Audio-Abspieler stellen natürlich ihre Ausgabe nicht in Fenstern dar, sie übernehmen aber die Idee der netztransparenten und maschinenunabhängigen Darstellung von digitalen Audio-Datenströmen von Fensteroberflächen wie X.

Ausgehend von ersten Erfahrungen mit Multimedia und den heutigen Fensteroberflächen haben wir argumentiert, daß die Metapher der Schreibtischoberfläche, die den Fenstersystemen zugrunde liegt, für eine wirklich multimediale Interaktion zwischen Mensch und Maschine ungeeignet ist, und wir haben dann die beiden neuen Paradigmen von der Virtuellen Realität und der unauffälligen Rechner überall im Kontext einer neuen Generation von Multimedia-Endgeräten vorgestellt.

Das Gebiet der Multimedia-Systeme ist interdisziplinär, und gerade bei den Benutzeroberflächen ist die Zusammenarbeit von Forschern und Entwicklern aus verschiedenen Wissenschaftsdisziplinen, wie z.B. Informatik, Psychologie, Soziologie, Biophysik und Linguistik, unbedingt erforderlich. Hier bleibt noch viel Arbeit zu tun.

Literatur

[1] Stephen Casner. Are You on the MBone? *IEEE MultiMedia*, 1(2):76–79, Summer 1994.

[2] Digital Equipment Corporation, Maynard, Massachusets. *Sound and Motion J300 – Owner's Guide*, 1993.

[3] James L. Green. The Evolution of DVI System Software. *Communications of the ACM*, 35(1):53–67, January 1992.

[4] Andy Hopper. Pandora – an experimental system for multimedia applications. *ACM Operating Systems Review*, 24(2), April 1990.

[5] Information technology – Coding of Moving Pictures and Associated Audio Information – Part 1: System. International Standard ISO/IEC DIS 13818-1, 1994.

[6] Ralf Keller, Wolfgang Effelsberg, and Bernd Lamparter. Performance Bottlenecks in Digital Movie Systems. In Doug Shepherd, Gordon Blair, Geoff Coulson, Nigel Davies, and Frankie Garcia, editors, *Network and Operating System Support for Digital Audio and Video,* 4th International Workshop, NOSSDAV'93, Lancaster, U.K., November 1993, Lecture Notes in Computer Science 846, pages 161–172. Springer-Verlag Berlin Heidelberg, 1994.

[7] Ralf Keller, Wolfgang Effelsberg, and Bernd Lamparter. XMovie: Architecture and Implementation of a Distributed Movie System. Technical Report TR-94-012, Praktische Informatik IV, Universität Mannheim, September 1994. URL= ftp: //pi4.informatik.uni-mannheim.de /pub/techreports/tr-94-012.ps.gz.

[8] Ralf Keller, Bernd Lamparter, and Wolfgang Effelsberg. Erweiterung von X für digitale Filme. *PIK – Praxis der Informationsverarbeitung und Kommunikation*, 15(4):196–204, October 1992.

[9] Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[10] Thomas M. Levergood, Andrew C. Payne, James Gettys, G. Winfield Treese, and Lawrence C. Stewart. AudioFile: A Network-Transparent System for Distributed Audio Applications. In *Proceedings of USENIX Summer 1993 Technical Conference (Cincinnati, Ohio, USA, June 21–25, 1993)*, pages 219–236. USENIX Association, 1993.

[11] Blair MacIntyre and Steve Feiner. Future Multimedia User Interfaces: Virtual Environments and Ubiquitous Computing. In Ralf Guido Herrtwich, editor, *Proceedings of Dagstuhl Seminar on Multimedia Systems 1994*. Springer Verlag, to appear, 1995.

[12] Parallax Graphics, Inc., Santa Clara, CA. *XVideo – User's Guide and Software Developer's Guide*, 1991.

[13] Ketan Patel, Brian C. Smith, and Lawrence A. Rowe. Performance of a Software MPEG Video Decoder. In *Proceedings of ACM Multimedia 93 (Anaheim, CA, USA, August 1-6, 1993)*, pages 75–82. ACM, New York, 1993.

[14] William B. Pennebaker and Joan L. Mitchell. *JPEG still image data compression standard*. Van Nostrand Reinhold, New York, 1993.

[15] Robert W. Scheifler and James Gettys. *X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD*. Digital Press, 1990. 2nd ed.

[16] Thierry Turletti. H.261 software codec for videoconferencing over the Internet. Technical Report RR-1834, Unité de Recherche INRIA-Sophia Antipolis, January 1993. URL= ftp: //ftp.inria.fr /INRIA/publications/RR/RR-1834.ps.Z.

[17] Mark Weiser. The Computer for the Twenty-First Century. *Scientific American*, 265(3):94–104, September 1991.

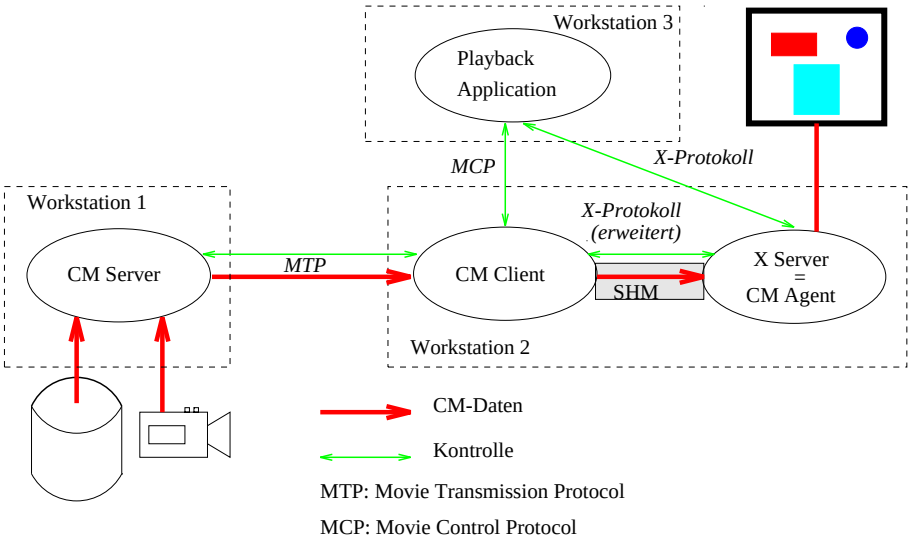

Abbildung 3: Bildfolgen über gemeinsamen Speicher