Visualization and Distributed Systems

Niklas Elmqvist <elm@cs.chalmers.se>

Contents

- Introduction
- Collaborative Virtual Environments
- Visualization of Distributed Systems
- Distributed Graphics
- Wearable Computers
- Conclusions

Introduction

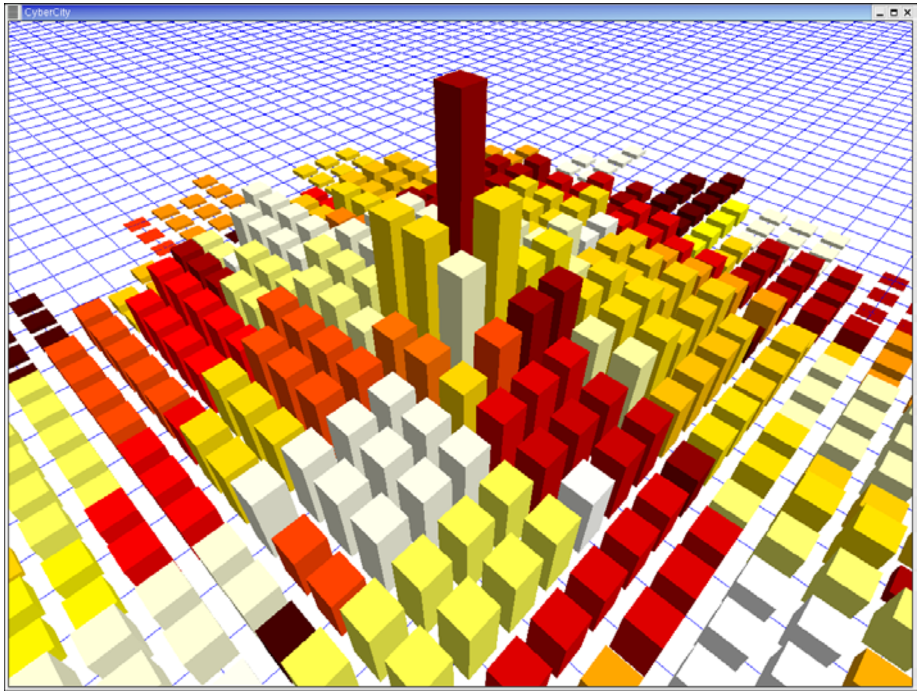
- **Visualization** and **distributed computing** are two fields of research with many potential connections
- Overview of the visualization research of the **Distributed Computing & Systems** group at the department
- We will discuss a selection of projects related to these two areas:
  - Visualizing distributed systems
  - Using distributed systems for visualization

Visualization

Visualization: graphical representation of data to aid in human cognition.

- Visualization is a very large field of research
- Many subfields
  - Information visualization
  - Scientific visualization
  - Data visualization
  - etc.
- Key component: computer graphics
- Key feature: high performance
- **Distributed systems** may be useful here!

Visualization: Example



- **CyberCity** information visualization technique
  - Visualization of multi-dimensional data
  - 3D computer graphics
  - Uses a ”city” metaphor with buildings, blocks, and a downtown area

- This is just one example of the unlimited uses of visualization

Visualization and Distributed Systems

- Why do we want to **combine** visualization and distributed systems?
- Two main reasons:
  1. Visualizing distributed systems
     - Distributed systems may be very complex to design, implement and debug.
     - Visualization can help us to do this easier.
  2. Using distributed systems for visualization
     - Visualization is usually a very CPU-intensive activity
     - We can use distributed systems to improve performance

Collaborative Virtual Environments

Of Quake and Quest

Introduction

- Sometimes it is beneficial to allow geographically distributed people to work (or fight!) together in a **shared environment**
- Online 3D multiplayer games are excellent examples of such environments
  - EverQuest, Dark Age of Camelot, Ultima Online, Quake, etc
- Visualization is an important part of these shared environments (see picture), and so is distributed systems

Scene from Dark Age of Camelot

Definitions

Collaborative Virtual Environment (CVE): general-purpose virtual world shared by participants across a computer network.

- Generalization of online multiplayer games to include any kind of activity
- Each participant is embodied by an **avatar**
  - Graphical entity representing
    - Identity
    - Presence
    - Location
    - Activity
  - Is used to interact with the world
- Convergence of Virtual Reality (VR) and Computer-Supported Cooperative Work (CSCW) research

CVEs and Distributed Computing

- CVEs are by definition distributed systems
- Participants are geographically distributed and connected using some kind of network
- Many interesting challenges:
  - **Scalability**: potentially thousands of users
  - **Interest management**: who shall receive state changes?
  - **Distributed architecture**: client/server or peer-to-peer (multicast vs unicast)
  - **Consistency**: how to ensure that all users have the same view of the world?

Scalability

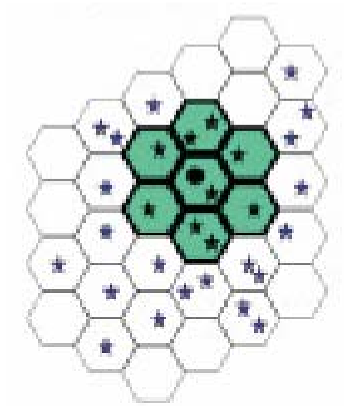
- Modern multiplayer games support up to 20,000 concurrent users (Anarchy Online)
  - Client/server architecture with a large server cluster
- A large-scale CVE may need to support even more users
  - Massive bandwidth requirements
  - Node processing power
  - Most of the users don’t see each other (at the same time)
- **Solution**: Send information only to nodes that are interested in it.

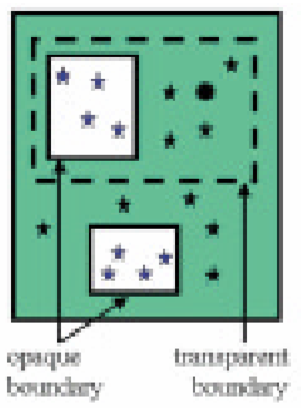
Interest Management

- Scalability problems can be remedied by only sending information to nodes that need it
- Two major approaches:
  1. **Geography-based**: send information to nodes that are in the vicinity of the originating node.
  2. **Interest-based**: send information to nodes that have registered an interest in this type of info.
- The second approach is the more general of the two but more complex to implement

Interest Management: Example

- The NPSNET system uses a network of hexagons
  - Works well for military simulations
  - Newest version of NPSNET has modularized interest management
- The MASSIVE-2 system uses hierarchies of dynamically resizing regions
  - Allows for easy culling of whole subhierarchies
  - Interest-based approach (in some ways)





Distributed Architecture

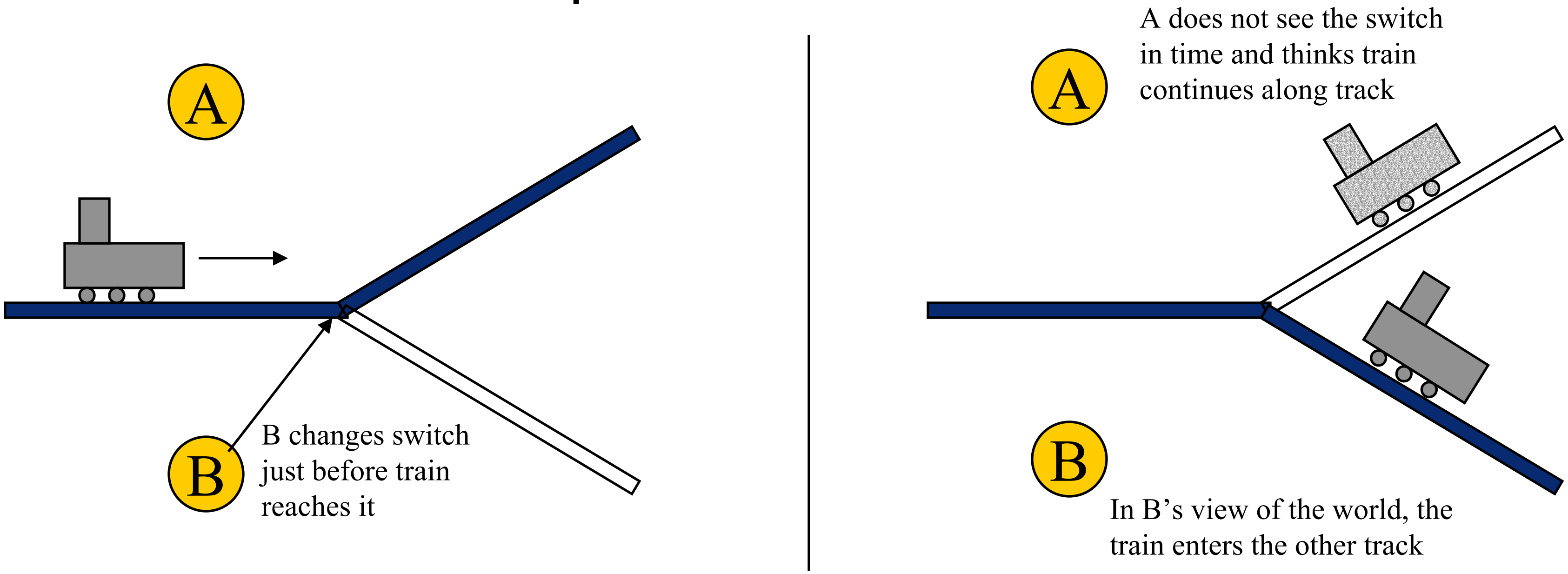
- Finding a suitable architecture is a major challenge
- Three basic architectures exist:
  1. **Client/server**: Clients only participate with a central server, never with each other. Easy, secure, but may scale poorly.
  2. **Peer-to-peer unicast**: Clients communicate with each other. Massive bandwidth but load is shared among all nodes, network delays low.
  3. **Peer-to-peer multicast**: Clients communicate with each other using multicast mechanisms (IP multicast). Bandwidth-efficient but not generally available

To Serve Or Not

- Client/server architecture
  - Advantages:
    - Easy to implement
    - Secure (one authorization point)
    - Reasonably efficient
    - Consistency is simple
  - Disadvantages:
    - Will not scale indefinitely
    - Vulnerable to faults
    - Inflexible
- Peer-to-peer architecture
  - Advantages:
    - Fault-tolerant
    - Bandwidth-efficient (if using multicast)
    - Dynamically reconfigurable
    - Scales well
  - Disadvantages:
    - Complex to implement
    - Consistency is hard
    - Hard to make secure

Consistency

- How do we ensure that all peers have the same **consistent** view of the shared world?
- Classic example: train switch



DCS and CVEs

- The DCS group is exploring the field of CVEs
- We want to build a system (or a prototype) with the following characteristics
  - Reconfigurable consistency (with consistency guarantees)
  - General area-of-interest management scheme
  - Peer-to-peer multicast architecture
  - General-purpose framework that can both be used for 3D worlds as well as collaborative editors

Visualization of Distributed Systems

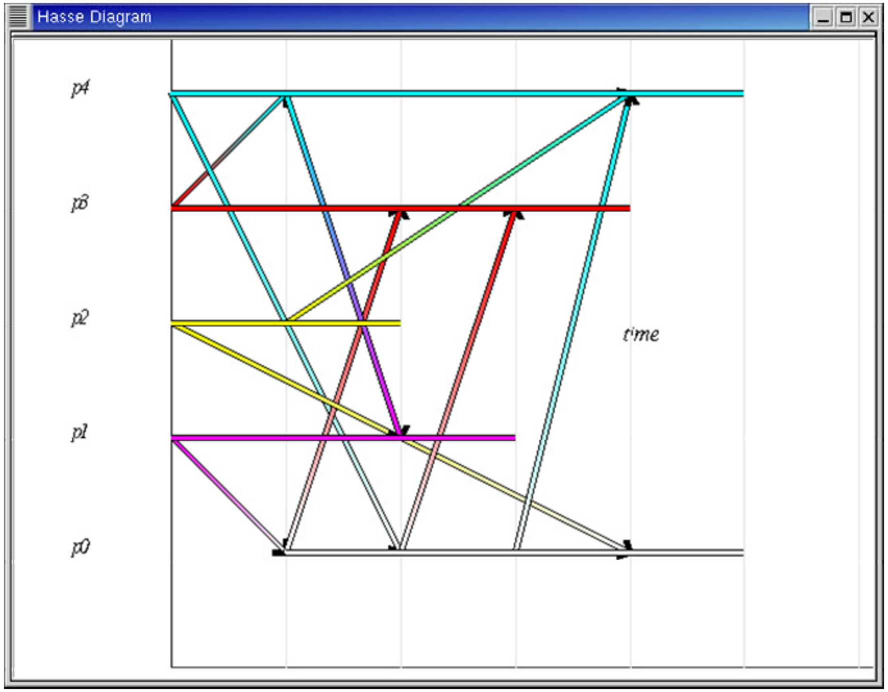
Growing Squares and CausalViz

Introduction

- Distributed systems are often complex to design, build, and debug
- Visualization may be able to help these activities
- Trace files of messages sent in a distributed system are useful diagnostic tools
  - Traces can be extremly large and hard to overview
  - The distributed system may be large and/or complex
- We need a visualization technique that takes system traces as input and gives a useful graphic representation as output

Hasse Diagrams

- **Hasse Diagrams** can be used for this purpose
  - Each process/node has a lifeline in a time-space diagram
  - Messages are represented as arrows between lifelines
  - This visualization quickly becomes messy

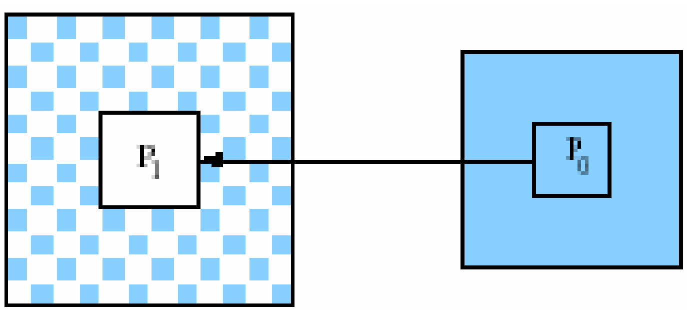


Growing Squares

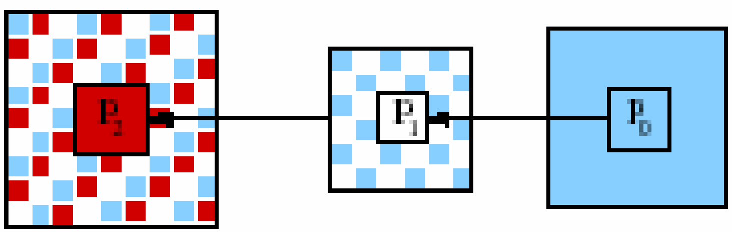
- The **Growing Squares** visualization technique was designed to solve this problem
- **Metaphor**: pools of color spreading on a piece of paper
  - Each pool is a process/node
  - Pools grow in size over time
  - Messages from one pool to another will add color to the destination pool
  - The influence of a single pool is easy to see just by studying its color composition
  - The history of the growing process is animated

Growing Squares (2)

- Message passing
  - Process p0 (blue) sends a message to process p1 (white)
  - Process p1 will now have both blue and white in it

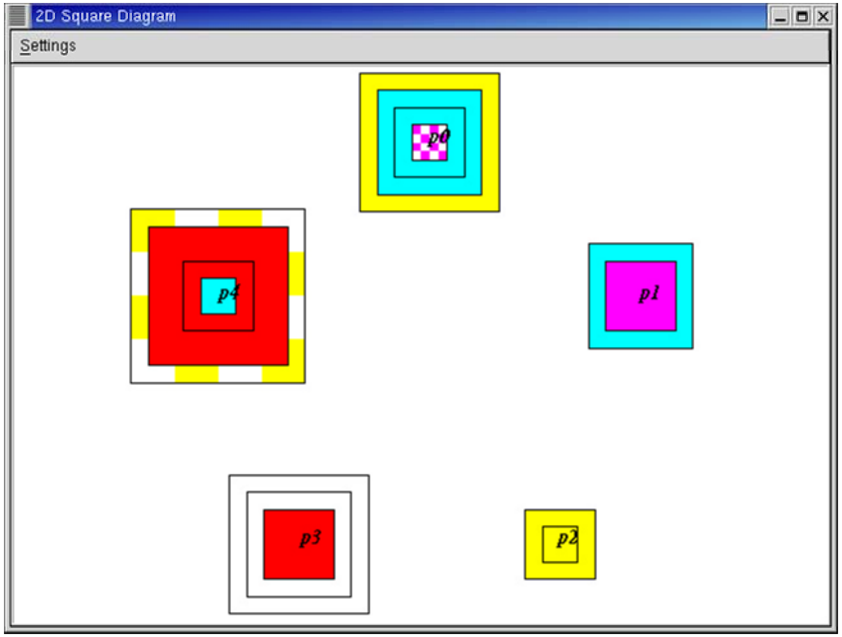


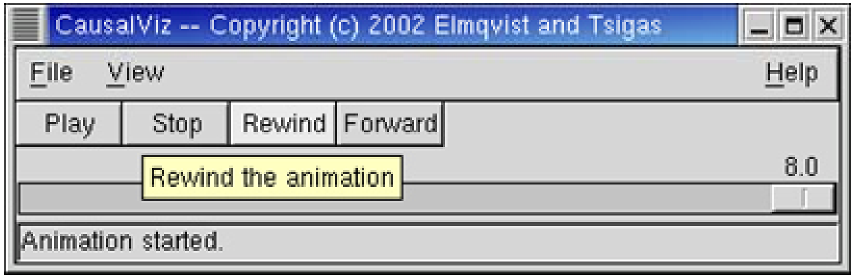
- Transitivity
  - Process p0 (blue) sends a message to process p1 (white)
  - Then, process p1 sends a message to p2 (red)
  - Transitivity is clearly visible in p2



Growing Squares: Example

- 5-process distributed system at $t = t_{end}$
  - Colors at each process shows influences
  - Animated history
  - Scales to larger systems



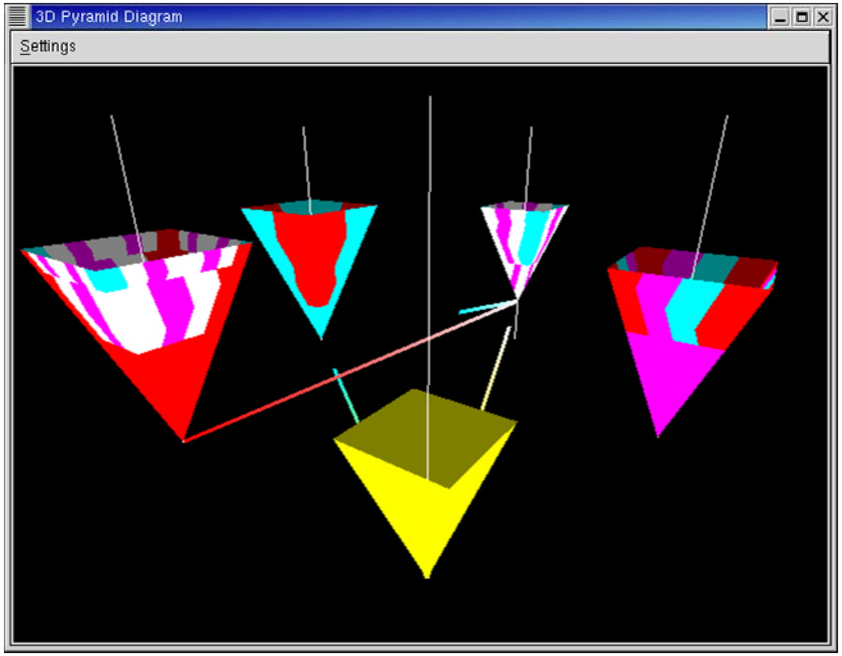


User Study and Results

- We have performed a user study comparing Hasse diagrams to Growing Squares
  - New method was faster and more efficient
  - Users preferred the new method over Hasse diagrams
- Things to be improved
  - Layout of processes (circular, grid, geographical)
  - Color assignment (colors too close)
  - Alternative visualizations?
- Work has been submitted to CHI 2003

Future Work

- 3D graphics is useful
  - Extensions to the original technique (i.e. Growing Pyramids)
  - More complex adaptions that make use of the 3rd dimension
- Work in progress
  - Require test subjects!
  - Sign up if you want to help



Distributed Graphics

3Dwm: The Three-Dimensional Workspace Manager

Introduction

- Some visualization applications have very high requirements
  - Virtual Reality
  - Computer-Supported Cooperative Work (CSCW)
  - High-end Computer-Aided Design (CAD)
- Networked high-performance workstations are becoming available
  - Visualization applications that were previously impractical are now feasible
- We need a framework to construct applications making use of heterogeneous set of platforms

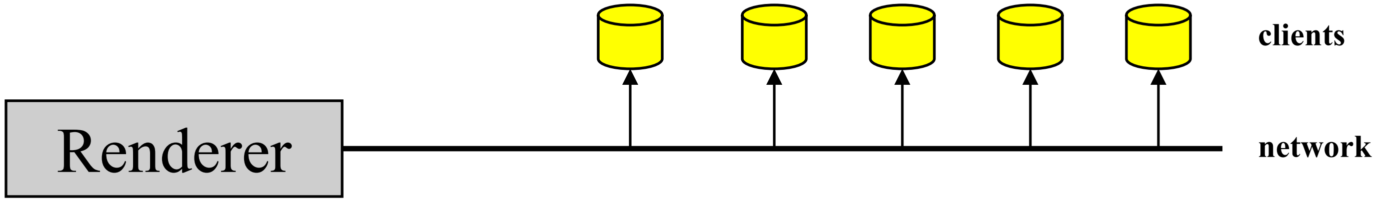
Definitions

Distributed graphics: systems for distributing shared graphical state of multi-display/multi-user, distributed, interactive applications. [MacIntyre *et al* 1998]

- Many excellent stand-alone 3D libraries exist
  - Example: Java3D, Performer, Inventor, VRML, etc
- Existing VR/CSCW platforms require ”dual databases” for graphics vs application state
- We are looking for a unified way to build distributed graphics applications

Basic Concepts

- Two basic types:
  1. **Multi-display**: multiple views of a single graphical database
  2. **Multi-source**: the graphical database is distributed and/or modified by multiple clients
- Hybrids are possible where both the database is distributed and rendered by different nodes



Example: multi-source distributed graphics application

Distributed Graphics Platforms

- Several distributed graphics platforms exist
  - **Repo3D**: general-purpose, object-oriented library for distributed 3D applications
  - **WireGL/Chromium**: distributed OpenGL, useful for distributing rendering across workstations
  - **VR Juggler**: Virtual Reality application framework, includes distributed graphics primitives
  - **MR Toolkit**: library for distributed graphics (mainly using ”crude” IPC primitives)
  - **DIVE**: CVE platform supporting mechanisms for distributed graphics

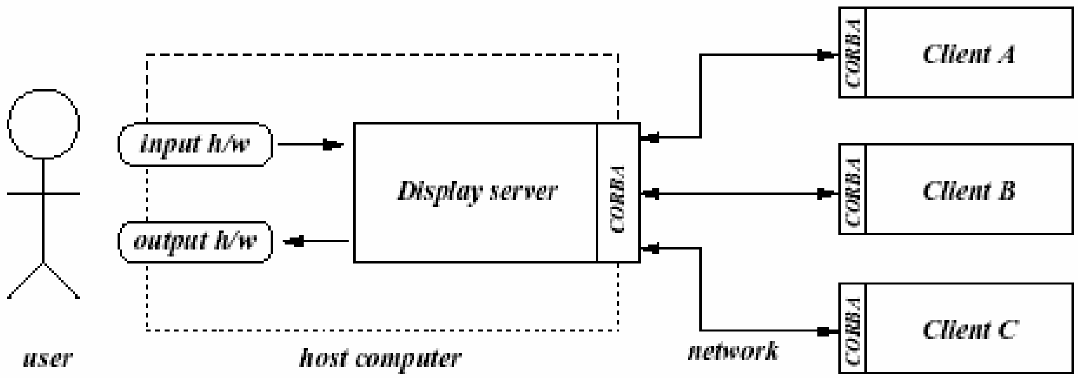
3Dwm

- The **3Dwm** (Three-Dimensional Workspace Manager) is an application framework for distributed 3D graphics
  - Initiated in 1999 by Niklas Elmqvist and Robert Karlsson
  - Supports all kinds of 3D platforms (VR, AR, etc)
- Contains a run-time component for managing concurrent workspaces and applications

Definition: a *workspace* is a special environment in which the cost structure of the needed materials is tuned to the work process using them [Card, Robertson, Mackinlay].

3Dwm: Basic Idea

- Single-user client/server interface system
  - Multi-source/single-display application platform
  - Each client maintains its own part of the graphical database
  - Display server contains the rendering hardware
  - Clients on the network
  - CORBA is used for communication

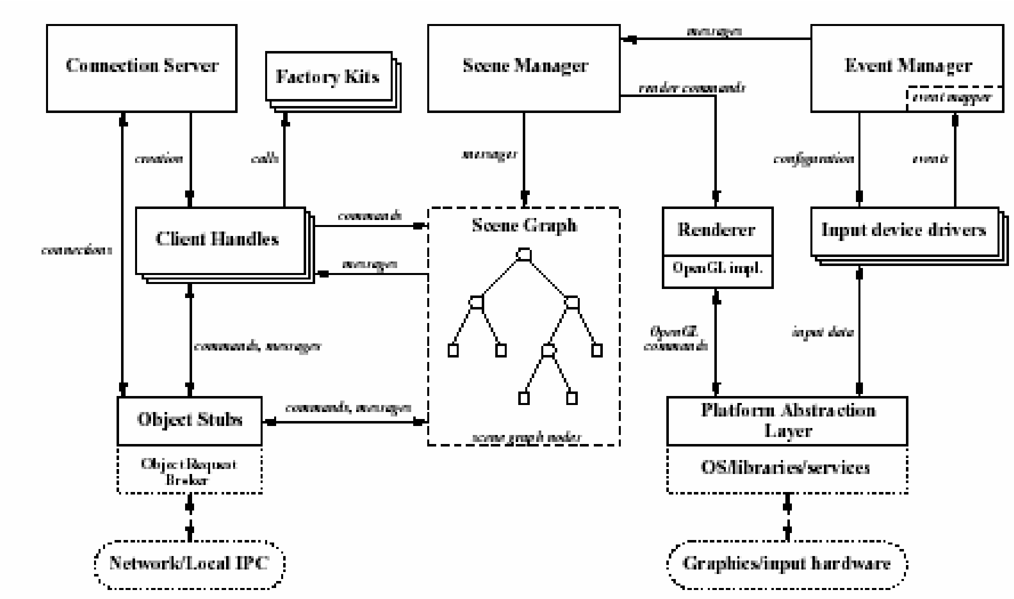


3Dwm: 3D User Interfaces

- 3Dwm is a platform for building applications with *3D user interfaces*
- 3D user interfaces extends conventional graphical 2D interfaces
  - Interfaces with ”volume”
  - Makes use of human spatial perception
  - Well-suited for Virtual and Augmented Reality
  - **Example**: immersive 3D modeller, simulation, 3D web browser, etc

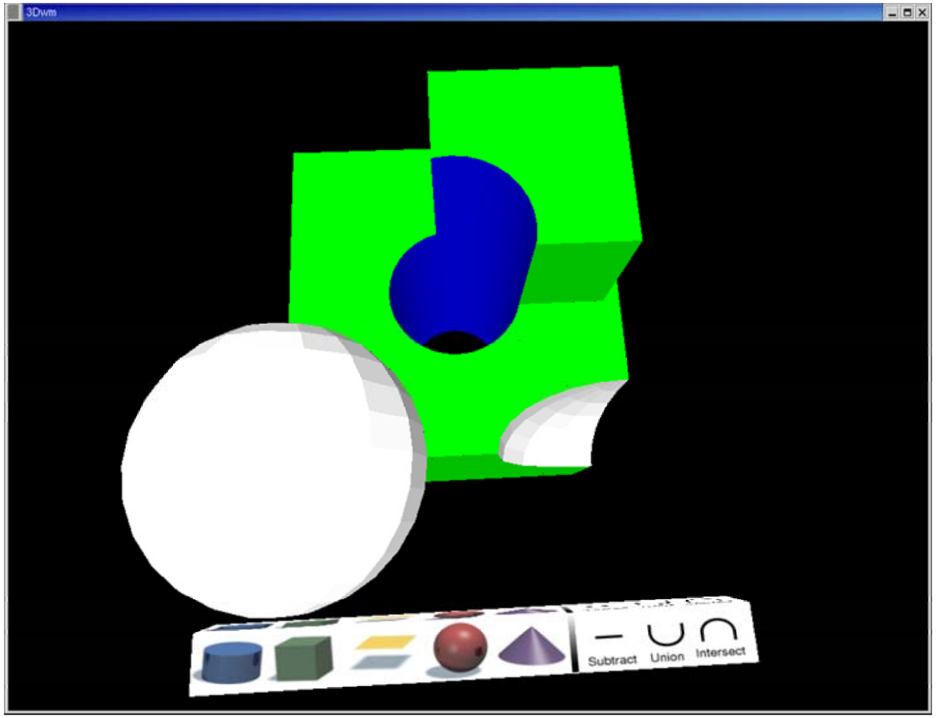
3Dwm: System Architecture

- Layered system
- Important components:
  - **Distributed 3D scene graph**: abstract scene description
  - **Platform abstraction layer**: portable hardware/OS interface
  - **Renderer**: support various 3D APIs such as OpenGL and Direct3D



3Dwm: Example

- Immersive CSG modeller
  - 3D model is created by combining 3D primitives
  - Supports spheres, quads, cylinders, cones, planes, etc
  - Very simple toolbar interface
  - The 3D space is the ”workshop”
  - Can coexist with other 3Dwm applications



3Dwm: Future Work

- See http://www.3dwm.org for more information on 3Dwm
- Many future research directions
  - 3D session management
  - 3D widget toolkit
  - Information visualization
  - CVE integration
  - Augmented Reality on wearable computers

Wearable Computers

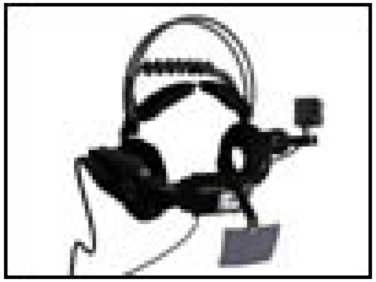
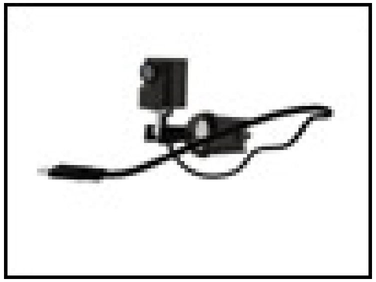
Truly Mobile Computing and Distributed Systems

Introduction

- **Wearable computers** are special computers that can be *worn*, like clothing
- Natural way to carry a computer
- Pioneered by Steve Mann of MIT in the 1980s
- **Basic feature**: personal imaging system (often via a head-mounted display)

Wearable Equipment

- Often x86-based computer
- Typical wearable:
  - Head-mounted display
  - Digital camera
  - Wireless communication
  - One-handed keyboard
  - Touch screen
  - Battery
- **Example**: Xybernaut Mobile Assistant IV (next slide)

Example: Xybernaut MA IV

Augmented Reality

- Augmented Reality: the next killer app?
  - Computer-generated images superimposed on the *real* world (VR on a virtual world)
  - 3D visualization important
  - **Example**: directions to a shop drawn on your normal vision
- Wearables are especially well suited for *Augmented Reality*
  - Wear them in everyday life
  - Head-mounted display (goggles) very useful
  - Head-tracking prominent

Augmented Reality: Example

- Many uses in everyday life
  - Tourist information in a new city
  - Soldiers with integrated order and navigation system
  - Operation support systems

DCS and Wearables/AR

- The DCS group has 2 Xybernaut MA IV wearables
  - One unit used by a D3 student project group
- We are conducting research in the following areas
  - Turning the 3Dwm system into an Augmented Reality platform
  - Wearables as mobile entities on a wireless LAN
  - Using 3Dwm and wearables in CVEs

Conclusions

Conclusions

- The fields of visualization and distributed systems have many interconnections
  - Mutual benefit
    - Visualization of distributed systems
    - Visualization using distributed systems
  - Interesting problems and possibilities
- Much work remains to be done
  - Integrating 3Dwm with CVEs
  - Integrating 3Dwm with wearables
  - Integrating visualization with 3Dwm
  - etc...