Distributed Artificial Reality Environment

D.A.R.E

Felix G. Hamza-Lup
Outline

- What is DARE?
- Software packages description
- Communication paradigms
- Maintaining the dynamic shared state
- Interaction methods
- Conclusions & future work
What is DARE?

- Software framework which implements Mixed Reality (MR) and Distributed Systems (DS) paradigms to:
  - improve development time for collaborative MR applications
  - provide a testbed for research in MR and 3D displays (e.g. data distribution, registration, calibration, virtual environment parameters assessment)

- Applications built using this framework:
  - are deployed in the Artificial Reality Center (ARC)
  - span the entire Virtuality Continuum
  - are visualized using HMDs (HMPD)
Software packages description (1)

**Networking package** provides
- a set of classes for fast development of scalable, real-time Mixed Reality applications on a LAN
- a novel paradigm for exchanging information through software objects
- a test-bed for development of real-time synchronization algorithms
- a test-bed in support of advanced coordination schemes for real-time networked environments

**3D Display package**
- provides an interface for three-dimensional displays.
- was designed around, and tested with optical see-through head mounted displays
- it interacts with several well known VR programming languages like OpenGL, SGI Performer and the Virtual Environment Software Sandbox (VESS).
- extensions of the package to account for deformable 3D objects are currently under implementation.

**Sensors package**
- provides a test-bed for interfacing devices that are used within MR environments.
- an application can receive device data without the need of a centralized device server
- currently, the package includes classes for optical trackers
- the package will be capable of providing device information in a variety of formats for registration and calibration in MR.

ODALab, School of Optics, UCF  July, 2003
Software packages description (2)

Calibration package
– provides algorithms for determining the transformations necessary to accurately display virtual objects within an application
– the algorithms include eye-point determination, camera calibration, and optical distortion calculation
– the algorithms can be adapted for use with any two-channel stereoscopic display

Registration package
– provides algorithms for placing real and virtual objects into spatial coincidence (registration)
– provides algorithms for assessing the quality of registration

Assessment package
– includes visual perception tests aimed at assessing virtual environment system parameters
– currently includes a modified Landolt C Visual Acuity tests as well as a perceived contrast test
– will include tests which allow mapping of the contrast sensitivity function for the HMPD display across all spatial frequency channels of the retina.
Software packages description (3)

- The Base package
  - contains algebraic operation for matrices, vectors and quaternions
Communication (1)

- The protocols used for network communication are:
  - TCP/IP
  - UDP/IP

- Communication paradigms:
  - Connectionless (one-to-one) - unreliable/low delay
  - Connection oriented (one-to-one) - reliable/higher delay
  - Mass distribution (one-to-many):
    - broadcast/low delay/bandwidth intensive
    - multicast/low delay/bandwidth conservation
Communication (2) – *IP Multicast*

- Each Receiver joins a group (instead of Sender control)

- Anybody can join any group
  - TV-like ctrl~ tune on any TV station

- **IP Multicast**
  - Implements mcast in IP layer
  - Packets are duplicated ONLY when necessary.
  - Addressing ~ Group Address
  - Receivers to join a group, IGMP
  - Mcast routing protocols for inter-router control
  - Slow deployment, MBONE
Network package classes (1)

- **dareTCP**
  - #MAX_CONN : int = 5
  - #PORT_NUMBER : int = 3339
  - #mSockFd : int
  - clientAddr : char
  - clientInfo : char
  - clientName : char
  - +InitializeServer()
  - +InitializeServer()
  - +InitializeClient()
  - +Accept()
  - +getClientName()
  - +getClientSocketD()
  - +getServerSocketD()
  - +Connect()
  - +Send()
  - +Receive()

- **dareUnicast**
  - #PORT_NUMBER : int = 3338
  - +InitializeServer()
  - +InitializeClient()
  - +getClientSocketD()
  - +getServerSocketD()
  - +Connect()
  - +Send()
  - +Receive()

- **dareCommunication**
  - #mClientSocketID : int
  - #mServerSocketID : int
  - #mSocketClientAddr : char
  - #mSocketServerAddr : char
  - +InitializeServer()
  - +InitializeClient()
  - +Receive()
  - +Send()

- **dareMulticast**
  - #MCAST_PORT : int = 6000
  - #MCAST_GROUP : char = 224.9.9.5
  - -mCastAddr : char
  - -mMcastGroup : char
  - -mTTL : int
  - -mReuse : int
  - -mreq : char
  - +InitializeServer()
  - +InitializeClient()
  - +getServerSocketD()
  - +getMcastGroup()
  - +LeaveMgroup()
  - +JoinMgroup()
  - +setMcastGroup()
  - +Send()
  - +Receive()

- **dareBroadcast**
  - #BCAST_PORT : int = 6200
  - #BCAST_IP : char = 127.255.255.255
  - #mBcastAddr : char
  - +InitializeServer()
  - +InitializeClient()
  - +getServerSockD()
  - +getMcastGroup()
  - +Send()
  - +Receive()
### Network package classes (2)

#### dareControlPackage

- mOX : int
- mOY : int
- mOZ : int
- modelPosition : dareVector
- modelOrientation : dareQuaternion

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>+setModelPosition()</td>
</tr>
<tr>
<td>+setModelOrientation()</td>
</tr>
<tr>
<td>+setModelRotation()</td>
</tr>
<tr>
<td>+setFileName()</td>
</tr>
<tr>
<td>+setMouse()</td>
</tr>
<tr>
<td>+setRotTime()</td>
</tr>
<tr>
<td>+setTransTime()</td>
</tr>
<tr>
<td>+setRotSpeed()</td>
</tr>
<tr>
<td>+setRotQuat()</td>
</tr>
<tr>
<td>+getModelPosition()</td>
</tr>
<tr>
<td>+getModelOrientation()</td>
</tr>
<tr>
<td>+getFileName()</td>
</tr>
<tr>
<td>+getRGBFileName()</td>
</tr>
<tr>
<td>+getRotTime()</td>
</tr>
<tr>
<td>+getTransTime()</td>
</tr>
<tr>
<td>+getTransSpeed()</td>
</tr>
<tr>
<td>+getRotSpeed()</td>
</tr>
<tr>
<td>+getColor()</td>
</tr>
<tr>
<td>+isZoom()</td>
</tr>
<tr>
<td>+isExit()</td>
</tr>
<tr>
<td>+isReset()</td>
</tr>
</tbody>
</table>

#### dareRTT

- MAXPACKAGE : unsigned int = 4096
- MAXHOSTNAMELEN : unsigned int = 64
- MAXWAIT : unsigned int = 10
- rtt : double
- s : int
- packet : unsigned char
- ntransmitted : int
- tz : int
- ident : int
- datalen : int

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>+getRtt()</td>
</tr>
<tr>
<td>+TvSub()</td>
</tr>
<tr>
<td>-pinger()</td>
</tr>
<tr>
<td>-pr_pack()</td>
</tr>
<tr>
<td>-in_cksum()</td>
</tr>
</tbody>
</table>
Maintaining the shared state

• The ControlPackage class allows instantiation of ControlPackage software objects

• The software objects contain information about the shared scene:
  – 3D objects position
  – 3D objects orientation
  – actions type (rotation/translation) applied on the objects
  – actions velocity
  – etc (open for sub-classing and aggregation)

• Objects can be transmitted through LAN using:
  – multicasting, broadcasting employing the client-server paradigm
  – tcp, Unicast(udp) employing peer-to-peer paradigm
Interaction methods (1)

- The definition of interaction differs according to research domain

- Human triggered interaction through a device (sensor) e.g.
  - mouse and a graphical user interface
  - position tracking sensor
  - speech recognition sensor, etc.

- Machine triggered interaction (programmed)
  - simulation loop
  - predetermined avatar behavior
Interaction methods (2) — *human triggered*

- Interaction sequence starts from human action
- Interaction is taken by means of an input device
- Executed interaction = the interaction that occurs within the virtual environment
Interaction methods (3) - *actions*

- **High frequency vs. Low frequency actions**
  - depends on the input device
    (action freq. triggered from a mouse: www.cs.ucf.edu/~fhamza/html/MouseClicking.html)

- **Predictable vs. unpredictable actions**
  - depends on the environment (static vs. dynamic)
  - **direct manipulation** [*Shneiderman, 82*] interfaces
    - Continuous representation of the object of interest
    - Physical actions instead of complex syntax
    - Impact on the object of interest is immediately visible

(!) Direct manipulation interfaces are easy to use, moreover, it also reduces the cognitive load.
Interaction methods - *DARE 3D pointer*

- Small spherical pointer
- Connected to mouse device
- Left/right OZ axis
- Stereoscopic appearance
Conclusions

- DARE is an open framework, freely distributed for research in MR environments and for promoting applications for 3D displays.

- Hopefully DARE will unite the development efforts of the ODALab team members.

- DARE project will be used as reference to pursue research initiatives and to obtain financing for the research initiatives.

- DARE_V1.0_alpha released to ODALab members mid June,03.

- DARE_V1.0_beta released to ODALab members mid July,03.
Future

• DARE_V1.0 will be released soon (we all hope) and can by publicly advertised.

• More applications will start using the framework classes.
Web Page

http://odalab.creol.ucf.edu/dare

suggestions and comments are welcomed