Designing 3D CVE is a complex task

- Address 3D interaction and immersion issues
  - Adaptation to 3D graphics API, to various displays (desktop, tablets, CAVE, ...)
- Address collaborative issues:
  - Distribution, synchronization, consistency maintenance of the shared data

To meet all these requirements, we have to merge:

- HCI and CSCW results about independence to graphics 2D API
- 3D CVE results about collaborative issues
Plan / Schedule

- Related work
  - Distributed architectures for 3D CVE
  - HCI and CSCW software architectural models

- The PAC-C3D model
  - Independence to 3D graphics
  - Adaptation to distribution modes

- PAC-C3D main features
  - Drive the same abstraction with different input devices
  - Visualize the same abstraction with different 3D API
  - Delegate behavior to abstraction and other presentations
  - Make interoperability possible between heterogeneous 3D viewers
  - Maintain consistency between Virtual and Physical world

- Conclusion and future work
Related work

3D CVE distribution modes

- 3D CVE architectures: duplicated model
3D CVE architectures: centralized model
Related work

3D CVE distribution modes

- 3D CVE architectures: hybrid model
Related work
HCI architectural models

➢ HCI models
CSCW models

- [Ellis, Wainer 1994] ontological, coordination and user-interface models
- Clover concept, PAC* [Calvary, Coutaz, Nigay 1997]
CSCW models

- Dewan's model [Dewan 1999], Clover model [Laurillau, Nigay 2002]

Related work
CSCW architectural models
The need: help for designing 3D CVE

- That should not rely too much on the 3D graphics API
- That should not be too dependent from the real displays
- That should not be too dependent from the network issues

First solutions:

- Static choice of the global distribution mode of the virtual environment
  - With no explicit separation between network and graphic features
  - With no possibility to change the distribution mode
- HCI software architectural models
  - That do not impose a strong enough separation with 3D graphics
  - That do not locate precisely the place where to put the collaboration management

These two research domains must be merged more efficiently

- To be more dynamic
- To provide explicit implementations of HCI/CSCW models
We propose a new interpretation of the PAC model: PAC-C3D

PAC + explicit interfaces for better independence between components
We propose a new interpretation of the PAC model: PAC-C3D

- PAC + explicit management of distribution policies
The PAC-C3D model
Managing duplicated architecture

➢ PAC-C3D and duplicated architecture
PAC-C3D and centralized architecture

The PAC-C3D model
Managing centralized architecture
The PAC-C3D model
Managing hybrid architecture

➢ PAC-C3D and hybrid architecture

Node 1

Node 2

Node 3

C/PDP

P

C/PDP

A

C/RDP

P

C/PDP

P

1: set

2: set

3: set

4: get

5: update

6: update

7: update

8: update

1: set

2: set

3: set

4: get

5: update

6: update

7: update

8: update
The PAC-C3D model
Creating PAC virtual objects

- PAC-C3D and AbstractFactory for creation of PAC components

- Then the controller may send messages to other nodes
  - According to its distribution policy...
The PAC-C3D model
Benefits of the PAC separation

➢ PAC-C3D encourages devices abstraction

2DPointer/3DRay

- setP
- getPO
- updatePO
- 3DPick
- do3DPick

Java3D Visualizer

- setPO
- updatePO
- 3DPick
- do3DPick

Wiimote

- (x,y) mouse move Event and/or (z) Wheel mouse event
- (x,y) Infra-red event
- button event

➢ Allows to drive virtual interaction tools with any physical device...
The PAC-C3D model
Coupling physics engines

➢ PAC-C3D and Physics engines

Server

Node 1

Node 2

➢ Physics engines are considered as active presentation components
PAC-C3D current implementation in the Collaviz framework:

- **Abstractions:**
  - ✔️ Java

- **Controls:**
  - × Java
  - × tcp or http/https communication layer
  - × 3 distribution policies
    - • The policy can change dynamically, at run-time
    - • Each virtual object can choose its own distribution policy (fine grain granularity)

- **Presentations:**
  - × Java3D
  - × jReality
  - × jMonkey (not yet fully operational)
  - × Aviatrix3D (for the Collaviz generic client)
  - × jBullet (for Physics)
To design a new kind of shared object:

- **Abstraction:**
  - ✔ Inherit from the Abstraction of an existing Collaviz object

- **Control:**
  - ✗ Inherit from the control of the corresponding Collaviz object (5%)
  - ✔ Reuse the control of an existing object (95%)
    - Standard behavior of inherited controllers ensure the maintain of consistency between Abstraction and Presentation, and between associated distant controllers

- **Presentation:**
  - ✗ Inherit from the presentation of the corresponding Collaviz object (5%)
  - ✔ Reuse the presentation of an existing object (95%)
    - Most of the shared objects have similar presentation features

- ✔ If needed, new Java interfaces for these new PAC components
To integrate a new 3D graphics API:

- Design new presentations for Collaviz objects
  - WorldObject, SharedObject, SupportedObject
  - Lights (DirectionalLight, SpotLight, PointLight, AmbientLight)
  - Camera (VirtualCamera, HeadTrackingCamera)
  - Other specialized objects (Plane, CuttingPlane, ...)

- Implement the AbstractFactory in charge of the creation of the objects
  - PXXX_ObjectManager
➢ To describe a CVE

✔ Describe the collaborative sessions
  ✗ The participants  ✔ The shared objects

✔ Describe the content of the shared environment
  ✗ The objects
    • Their name, their owner, theirs access mode (who is allowed to do what...)
    • Their presentation features (geometry name, sound, ...)
    • Their distribution policy

✔ Describe the environment of the user
  ✗ Which session he wants to join (session name, user name and password)
  ✗ Which interaction tools he is using
    • 3D cursor, hand, ray, ...
  ✗ What is his physical environment made of
  ✗ Which devices drive his interaction tools
    • Mouse, Wimote, ARTracking, ...
  ✗ Which kind of 3D rendering he is using
    • Java3D, jReality, ...
PAC-C3D:

- Evolution of the PAC model
  - Explicit interfaces between components
  - Very small dependency on 3D graphics API
  - Control components are also in charge of the collaboration
- Deals with different distribution modes
  - Through the control components

Tangible results through our Collaviz implementation:

- Ability to build “abstract” virtual worlds
  - With quite standardized control components
  - With dynamical choice of the distribution mode, for each virtual object
  - With wide choice of 3D graphics visualization
- Ability to provide abstract navigation and interaction tools
  - That can be driven by any kind of physical devices, even by a 2D GUI
- Ability to share a virtual world between heterogeneous 3D viewers
  - With possibility to exchange services between them!
Future work

➢ Take into account “active presentations” more efficiently
   ✔ Work in progress...

➢ Couple PAC-C3D objects with other kinds of “engines”
   ✔ Artificial intelligence behavior libraries

➢ Propose a kind of abstract scene-graph to describe CVE
   ✔ Should cover the most common 3D features such as X3D or Collada
   ✔ Should describe the distribution mode of each object
   ✔ Should deal with interactive and collaborative capabilities for virtual objects
      ✗ Kind of interaction for each object, how many users at the same time, ...
   ✔ Should manage access rights for interaction and collaboration
      ✗ Who can interact, who can join or interrupt an interaction...
   ✔ Should describe the interaction tools capabilities
   ✔ Should describe the physical environment of the users
      ✗ To embed it within the virtual environment, to prevent collisions, to improve collaboration, ...
➢ OpenSG as a new presentation for Collaviz
  ✔ Same kind of integration than Java3D, jReality, jMonkey
  ✔ With a Java – C++ binding

➢ SOFA as a new physics engine for Collaviz
  ✔ Same kind of integration than jBullet
  ✔ With a Java – C++ binding

➢ Collaviz as a backbone between OpenSG and SOFA
  ✔ In the same way that Collaviz makes links Java3D, jReality and jBullet

➢ Replace Collaviz by a C++ backbone if needed
  ✔ To be more efficient than the Java backbone
  ✔ Still respecting the PAC-C3D model

➢ Propose a standardized description of a CVE
  ✔ Through X3D or Collada Extensions