



Opening Mobile Phones to Multimodal Interaction - The Multimodal Hub Approach

Nilo Menezes (Multitel ASBL)

Sophia Antipolis, France, November 19, 2008

Agenda



- Our case
- The OI Kernel
- Multimodal Applications
- The Mobile Barrier
- The Multimodal Hub
- Multimodal Middleware Protocol
- Game example
- Open Source
- New standard
- Conclusions

Our case

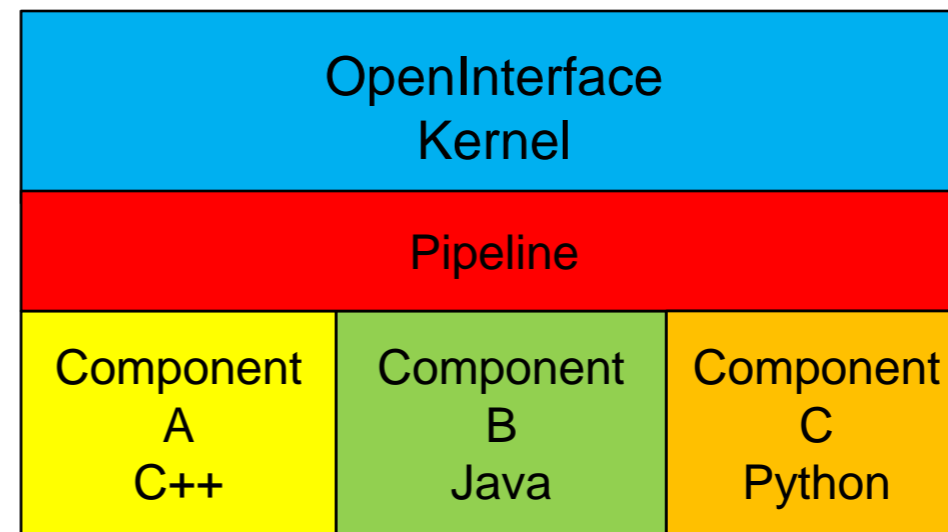


- In the OpenInterface project, we needed to validate Multimodal components in mobile phones
- Our existing solution, the OpenInterface Kernel was not able to run or even to be ported to mobile phones
- A first intermediary solution was built, but it still need a link to a computer

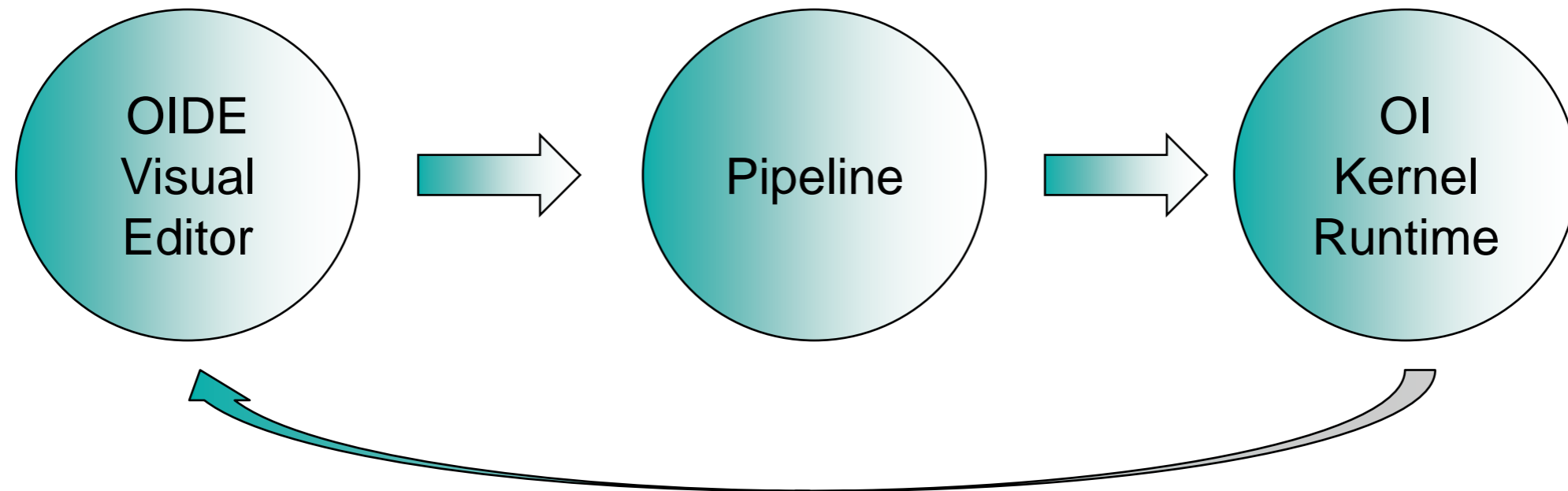
The OI Kernel



- Supports components written in Java, C++, Python, Matlab and .net.
- Runs on Linux and Windows
- Components integrated as libraries
- Pipeline is built to glue all components in a new application



Multimodal Applications

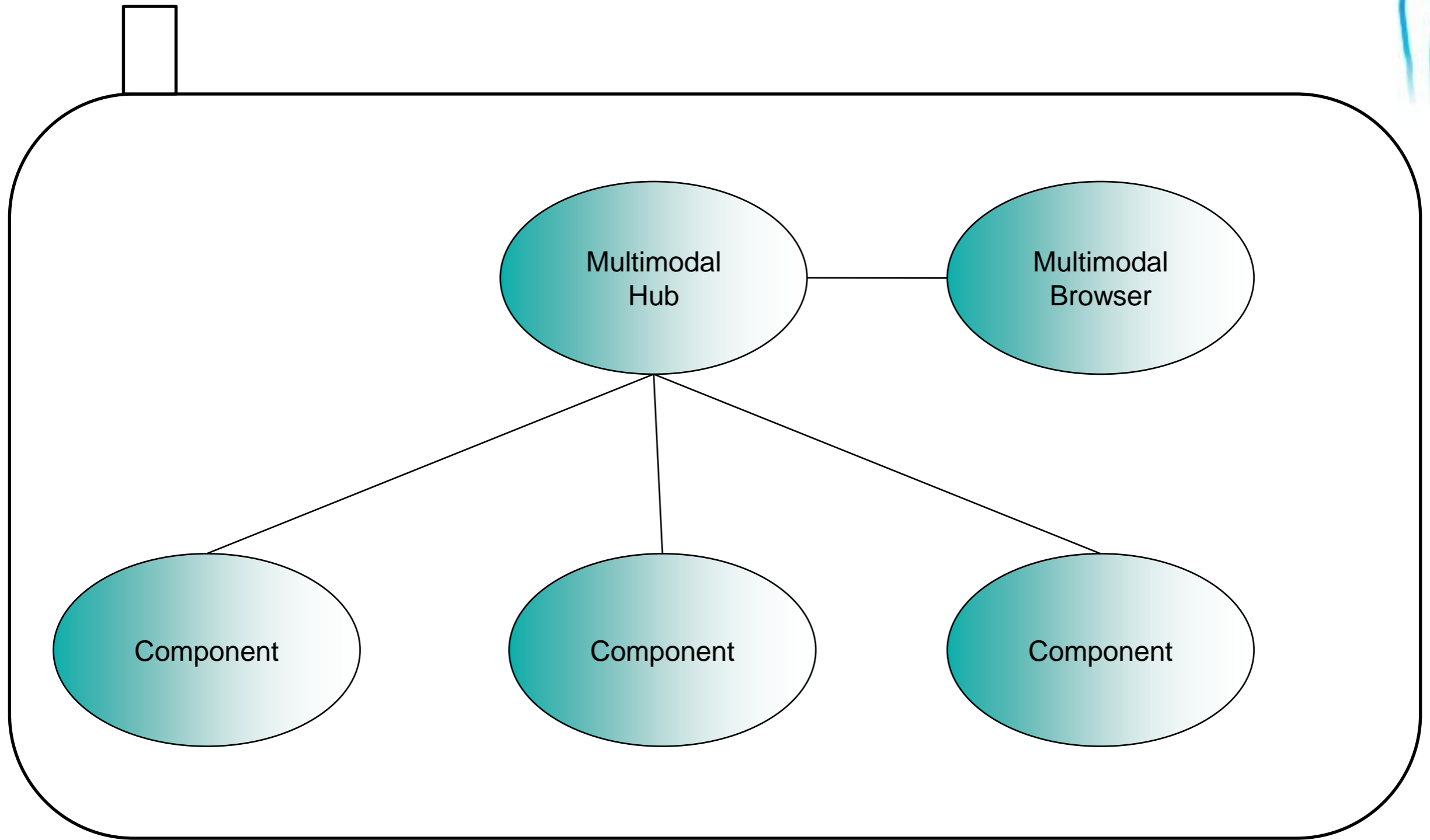


The Mobile Barrier



- Low resources (CPU and Memory)
- No support for external libraries (applications are installed in separate slots, only system libraries are shared)
- No capability to start effectively external applications
- Very difficult integration of components written in different programming languages

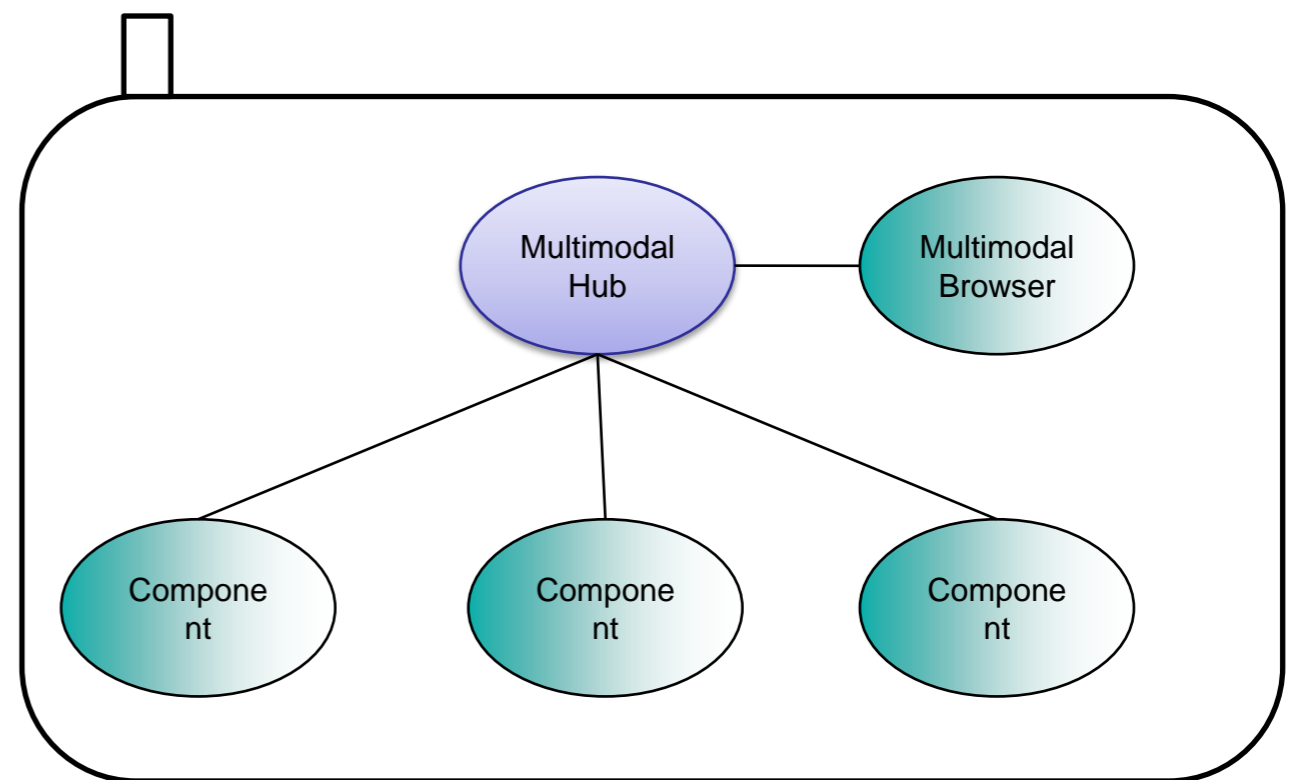
The Multimodal Hub



Multimodal Hub



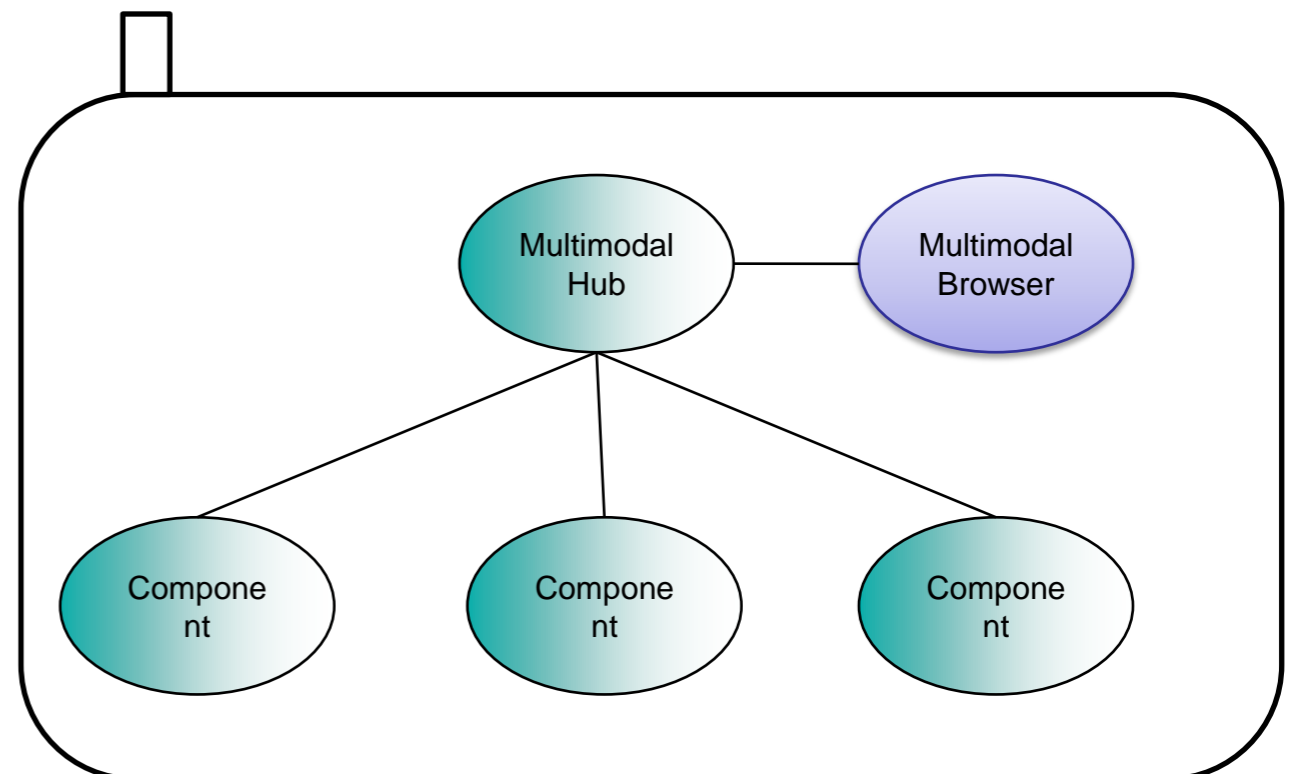
- Central connection point for all components
- Able to route messages between components based on publish/subscribe mechanism
- Uses the Multimodal Middleware Protocol
- Can use multiple network protocols
- Multidevice



Multimodal Browser



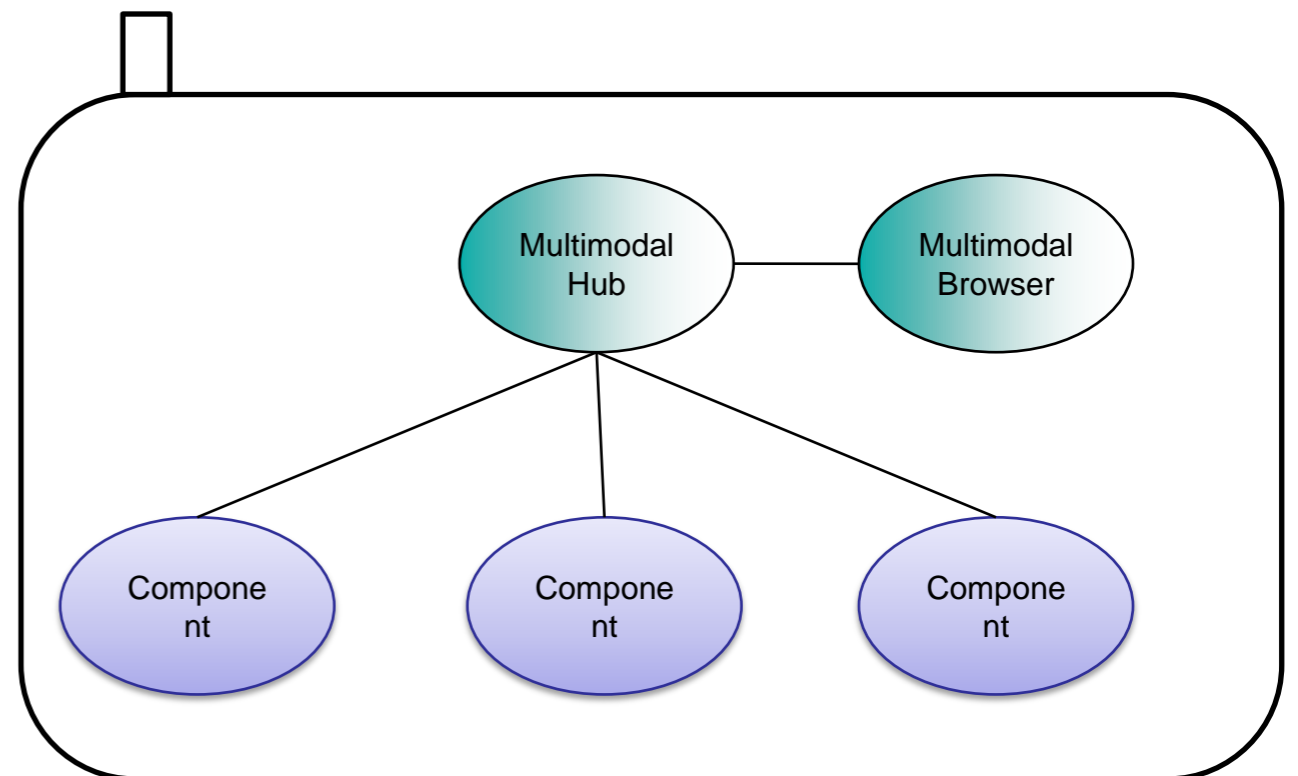
- Visual Interface to the Multimodal Hub
- Can be seen as a special component
- Or even attached to the Hub in the same application
- Enable user configuration
- Supports run time changes without restarting the component network



Components



- Connect to the Multimodal Hub
- Produce events
- Consume events
- Can be written in any language
- Use the Multimodal Middleware Protocol
- Translate
- Compose
- Filter
- Log and Inspect



Multimodal Middleware Protocol



- Defines a cross platform data serialization mechanism
- Extensible to support new messages
- Events described using XML, but exchanged in binary format for performance reasons
- Uses the OI Repository to store new event formats
- Enables message routing between components using three fields: UCID, DeviceID, EventID.
- Support standard rules for event routing

Multimodal Middleware Protocol



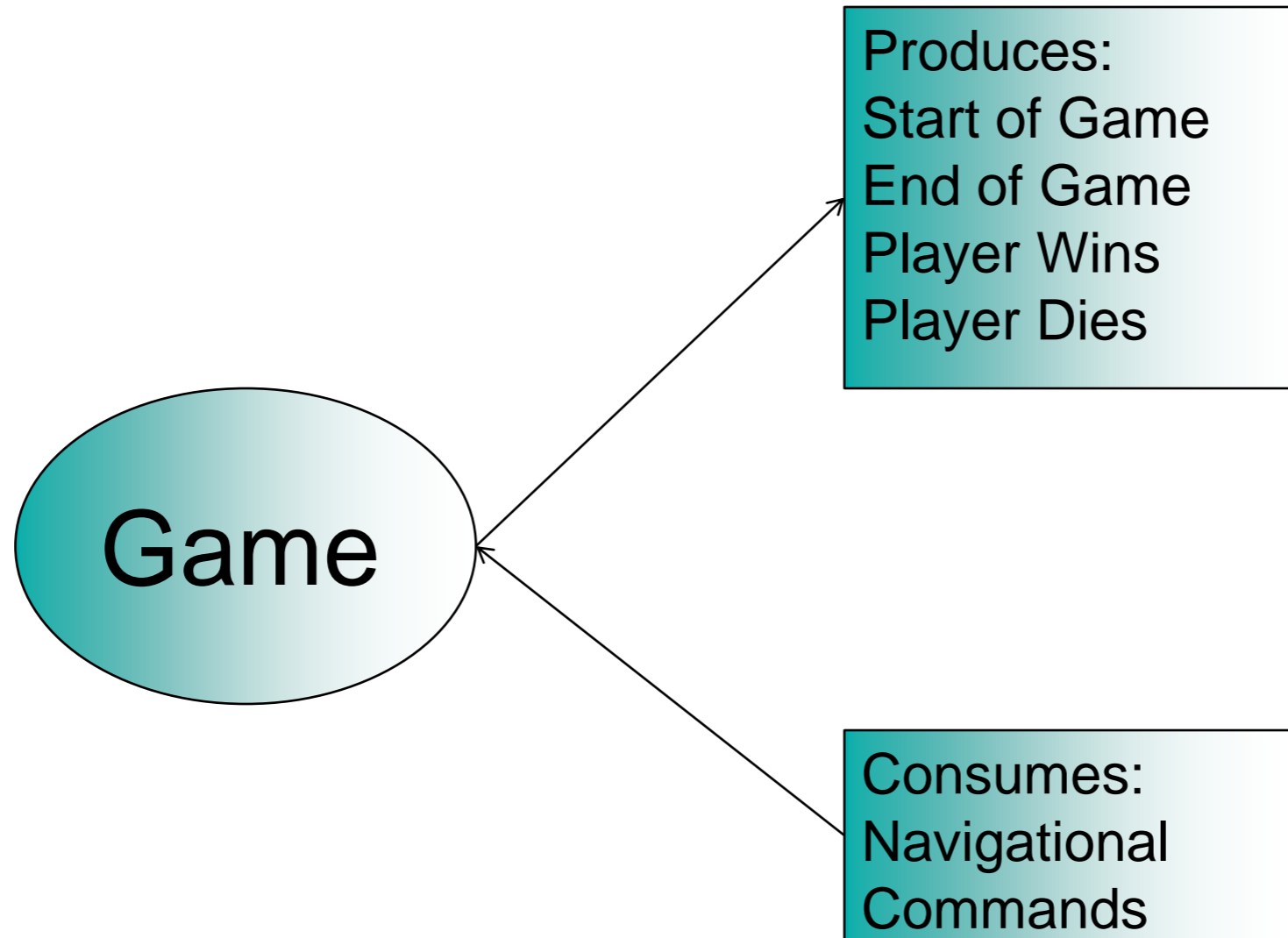
- All components connect to the Multimodal Hub
- Once connected, they announce their list of produced and consumed events.
- The same component can consume or produce any number of events
- The Hub store these lists and try to automatically connect components based on a simple match algorithm
- Sophisticated operations can be made using a configuration file, enabling even direct connections between components
- Uses the best protocol for the service, does not re-specify existing protocols.

Multimodal Middleware Protocol

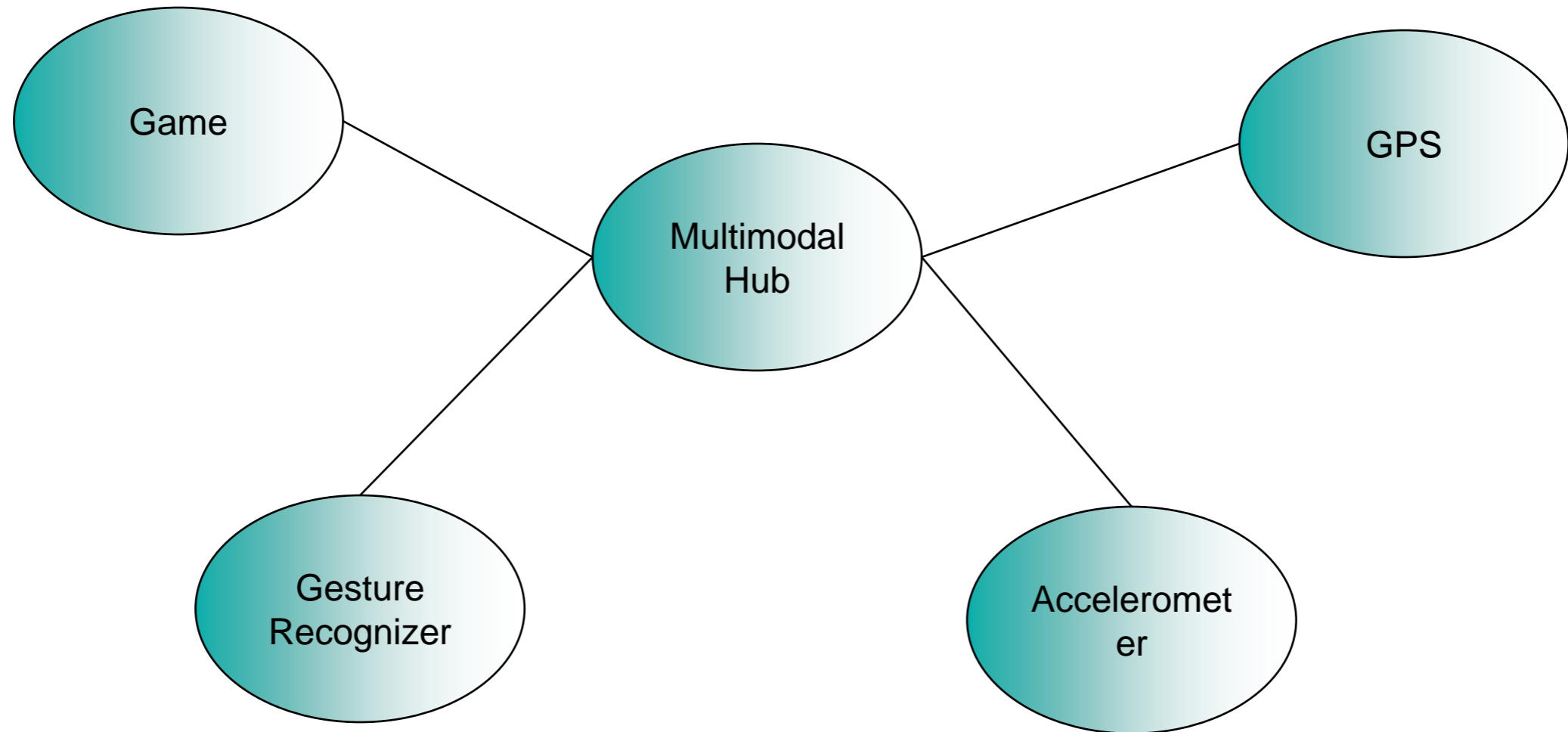


- Lightweight
- Suitable for Mobile and Embedded devices
- Cross platform
- As simple as possible
- Logic is implemented using components
- Multidevice
- Provides a standard interface between components, allowing component reuse, exchange and experimentation

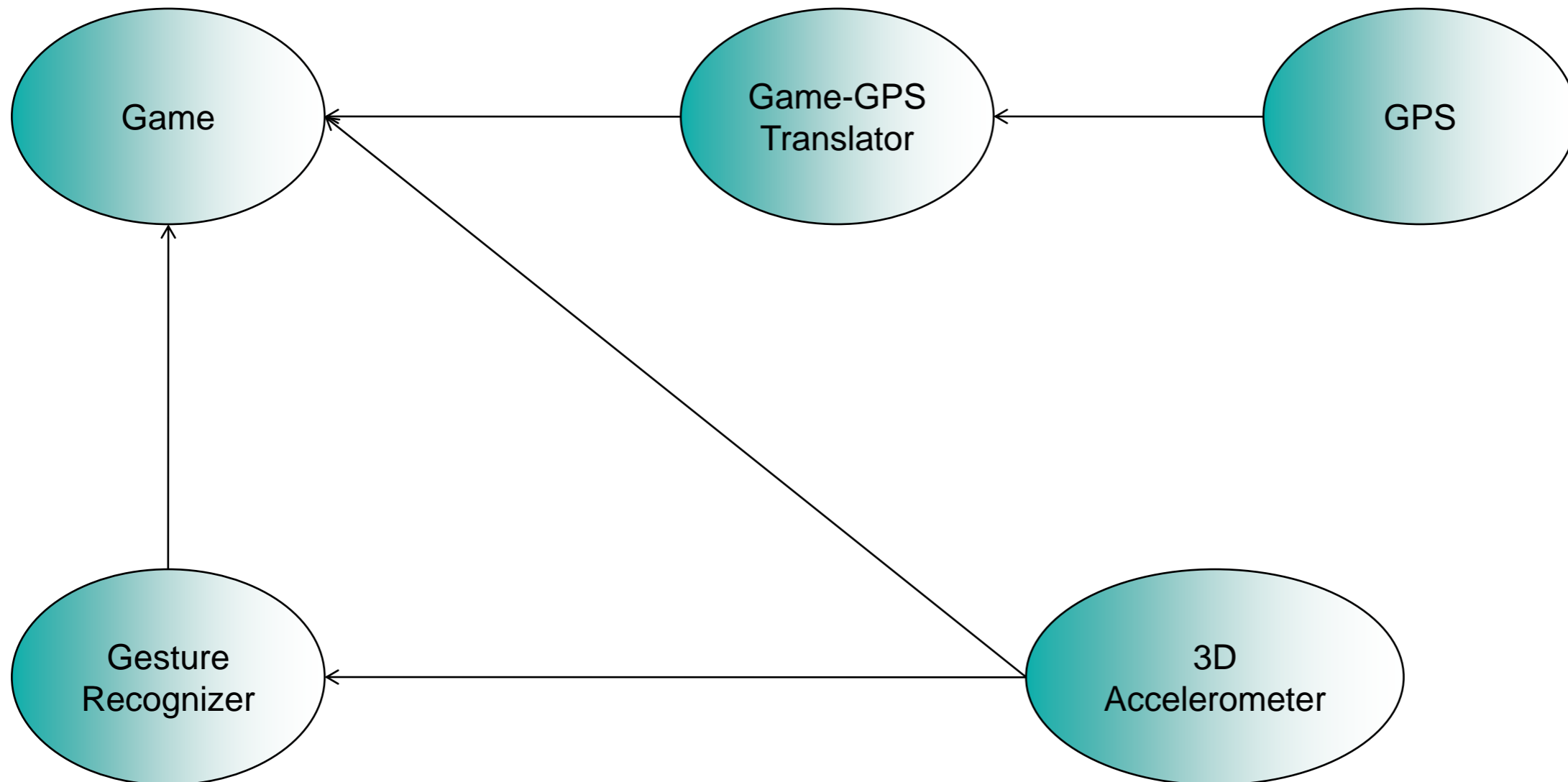
Game Example



Game Example



Game Example



Open Source



- Python Multimodal Hub:
 - LGPL License
 - Hosted at SourceForge.net
 - Implemented in the Python programming language
 - Ready to run on Windows, Linux and Mac OS X

New Standards



- To extend and improve the actual proposition
- To support new modalities
- To provide end users with configurable multimodality
- To enable multimodal components and applications

Conclusions



- A real standard is made with adoption
- Open to extend and modify the protocol
- A standard can create opportunity to the development of new components
- End user configurations will enable new configurations and uses for existing modalities
- Adaptation to user needs
- User scriptable components can enhance multimodal experience

Thank you!