



Sharing One World: Multi-Client applications with XML3D

Tutorial on Mixed Reality for the Web
Web3D '14, Vancouver

Torsten Spieldenner

German Research Center for Artificial Intelligence (DFKI)

August 10th 2014

Recap



We have seen so far:

- XML3D for interactive 3D graphics in browser
- 3D Assets for configurable scene content
- Efficient content delivery

Question:

- How to *Share* a 3D-scene among multiple users?
- Possible use-cases: Gaming, Social Platforms, Training, Collaborative Work ...

Synchronization GE



(From: *FI-Ware OpenSpecification*)

Goal:

- Common requirement of many multi-user environments: store some *World* on a server
- World state needs to be *synchronized* to connected client applications
- Large scale applications: *Distribution* over several servers, e.g. by dynamic spatial partitioning

Synchronization GE



Special Requirement:

- Target platform: [Web-browser](#)
- Link [XML3D-Asset-Instances](#) to entities in virtual environment

How to get there?

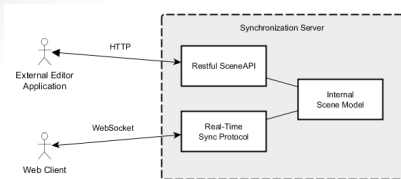


Synchronization GE



Synchronization:

- Lightweight server application
- SceneAPI and SyncProtocol access internal scene for modification
- GE provides sample implementation of server and JavaScript implementation of sync library



Entity-Component-Attribute Model



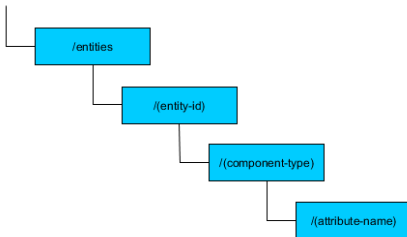
World Representation in Synchronization GE:

- Generic, non-domain specific data model
- World objects: *Entities* without any data
- Entities can be equipped with *Components*
- Components are containers for typed *Attributes*

RESTful Scene API



http://{serverRoot}:{serverPort}



RESTful Scene API:

- Resource-oriented API accessed by HTTP
- Query and manipulate scene objects and scene structure
- Used for low-frequent updates

Realtime Synchronization

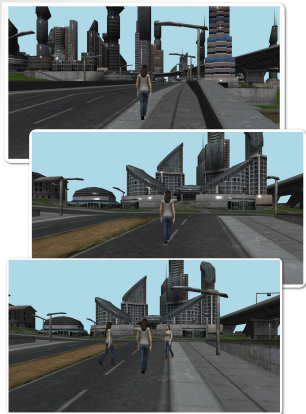


Real-time interaction operates in *phases*:

1. Establish connection to server via WebSocket
2. Receive *intial scene state* from server
3. Continuously receive and send updates to the server
 - Create and delete *entities*
 - Create and change *components and attributes* of entities
 - Send an *entity action*
4. *Disconnect* from server

Synchronization FIVES

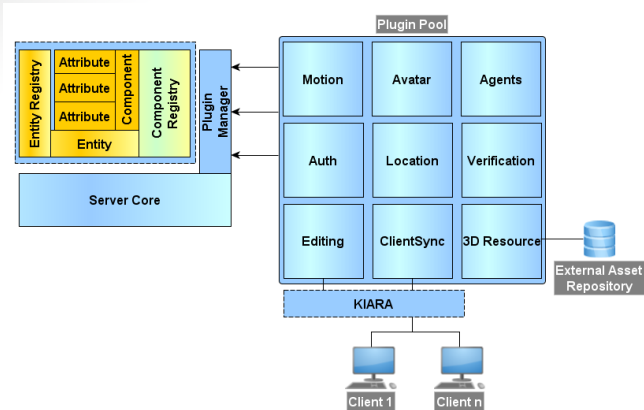
GEi Proposal:



FiVES (*Flexible Virtual Environment Server*):

- Synchronization server implemented in C#
- Slim server core with generic world model
- Domain-specific features can be added via *Plugins*
- Web client implemented in *JavaScript*, based on *XML3D*

FiVES General Architecture



ECA Model in FiVES



Adding Data to FiVES Entities:

- New Components are registered by *Plugins*
- *ComponentRegistry* keeps track what components are available
- *ComponentDefinitions* define the set of and types of attributes
- Once registered, components can be accessed on every entity

Example:

```

1 ComponentDefinition p = new ComponentDefinition("position");
2 position.AddAttribute<float> ("x", 0f);
3 position.AddAttribute<float> ("y", 0f);
4 position.AddAttribute<float> ("z", 0f);
5 ComponentRegistry.Instance.Register(p);
6 // Accesss attribute on entity:
7 float x = (float)entity["position"]["x"];

```

Synchronizing FiVES worlds



Synchronizing Data:

- Data entirely contained in Attributes → Synchronize *Attribute values*
 - Attribute value synchronization implemented in *ClientSync* Plugin:
 - Plugin subscribes to *ChangedAttribute* events of entities
 - On change, generate *UpdateInfo* object that contains all information about changes
 - Send update info to clients to apply the changes on their side
 - Queue updates to improve performance
 - Components and Attributes are synchronized as soon as they are registered
- **No need to implement synchronization for custom Plugin**

KIARA - Advanced Middleware GE



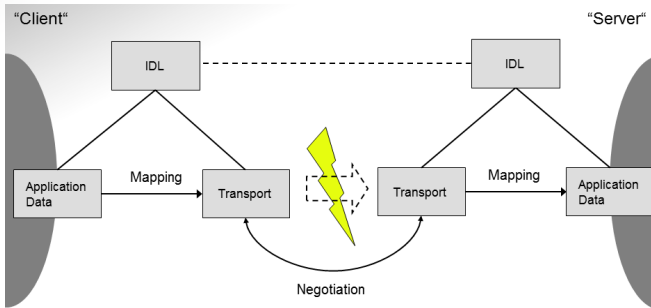
KIARA API:

- *Uses native data types of application*
 - Declarative API to declare types
 - Manual declaration or generated *automatically (Clang preprocessor) / Reflection*
- *IDL for interface definitions*
 - IDL defines types and annotations of remote application
 - Parsed by IDL parser at runtime
- *Multi-protocol support*
 - Selects protocol best suitable for connection
 - Extension mechanism to add protocols
- Implemented in *C/C++, C#, Java, JavaScript*

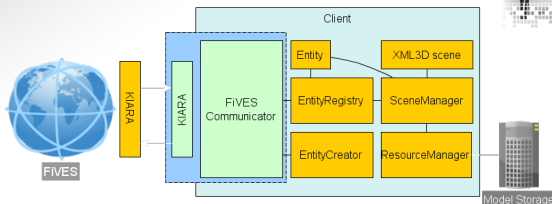
Design Principle



KIARA - Advanced MiWi GE:



FiVES Demo Client



Client Design:

- Maintains scene state in JavaScript objects
- *SceneManager* translates ECA to XML3D representation
- *ResourceManager* loads externally stored models in XML3D format
- Synchronization of client with server state via *KIARA*

Synchronized Asset Instances



Model-View-Presenter Pattern:

- *Model*: Entity in ECA format

```
1 var e = {  
2   "id" : "ba5290f3",  
3   "components" : [  
4     "renderable" : {  
5       "uri": "/resources/model.xml"  
6     }  
7   ]  
8 }
```

- *View*: XML3D model with configurable nodes

```
1 <model id="ba5290f3" src = "/resources/model.xml" />
```

- *Presenter*: In our case: Scene-Manager (Passive View)

Synchronized Asset Instances



Establishing a connection to the server:

1. Client connects to FIVES server
2. Requests list of KIARA services provided by the server
3. Wrap KIARA services to JavaScript functions
4. Call services with local data types

Synchronized Asset Instances



Send Updates to Server:

- Wrap service JavaScript function (*Example: update entity position*)

```
1 updateEntityPosition = connection  
2   .generateFuncWrapper("location.updatePosition");
```

- Call Function with local data types:

```
1 updateEntityPosition(guid, position);
```

Synchronized Asset Instances



Handle updates from server:

- Apply received updates to model:

```
1 updatedEntity.updateAttribute(update.componentName ,  
2     update.attributeName ,  
3     update.value );
```

- Apply updates to view:

```
1 var transform = entity.getXml3dTransformElement();  
2 transform.translation.set  
3     (entity.components.location.position);
```

Demo!



Summary



Synchronization GE:

- Provides *Generic World Model* to represent many variations of Virtual Environments
- *Modular* to allow easy adaption to domain specific use-cases
- *REST SceneAPI* and *RealTime Sync Protocol* to transmit scene updates to clients

FiVES GE Implementation:

- Employs *ECA Model* for scene description
- Implements *Plugin System* for modular extendability
- Uses *KIARA Service Interface* for Real Time Synchronization
- REST API under construction