VCoRE: A web resource oriented architecture for efficient data exchange

Tobias Alexander Franke  Fraunhofer IGD
Volker Settgast  Fraunhofer Austria
Johannes Behr  Fraunhofer IGD
Bruno Raffin  INRIA
"" based Rendering; Using the X3DOM BinaryGeometry Container

Polygon Model
Motivation

Introduction

- Have several demonstrators in desperate need of intercommunication
- Technological advancements
  - WebSockets [W3C 2011]
    - Allow push-like interaction
    - Transfer any kind of data to web apps
  - TypedArrays [Khronos 2012]
    - Binary blobs inside Javascript
    - Single blob can have multiple views
      - Uint16Array
      - ...
Introduction

Previous work

- **SIMNET/DIS/HLA**
  - Used in military simulations
  - DIS has generalized attributes as PDUs
  - HLA has abstract parameters, objects etc.
  - Not a general purpose distribution format

- **Linked Data**
  - W3C effort for “Web of Data”
  - Uses RDF to describe entities
  - Similar to ours, but way too bulky
Introduction

Previous work

- **InstantIO**
  - Data-flow graph
  - Represents devices as **nodes**
  - Nodes can produce/consume data: **slots**
  - Network protocol to transfer data from/to slots

- **RESTful service on OpenSG** [Schiefer et al. 2010]
  - Manipulate nodes and their fields
  - Query nodes as binary blobs instead of text
VCoRE

Basics

- High performance REST interface on top of library/application
- Not bound to internal hierarchy, more generalized/simpler
  - Represent data as simple attributes
  - Represent collection of attributes as elements
  - Collection of elements are pools
  - Attributes and elements have descriptor
VCoRE
Mapping structures

X3D  OpenSG  XML3D  glTF  Instant IO  SOFA  ...
VCoRE

Query string

Websocket or HTTP Request

{ws|http}://host/pool/element/id/{desc|data/attr/{desc|data}}{.type}

Query operation

Fetch descriptor or data of Element

Fetch descriptor or data of Attribute

Name identifying collective of Elements

Identifier of Element

Name of an Attribute

Specify type to deliver data in
Represent all data in an application as **PODs** *mostly known from graphics* (HLSL, Cg, GLSL, OpenSG, X3D, Collada)

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerals</td>
<td>float, double, short, ushort, byte, octet, int, uint</td>
</tr>
<tr>
<td>Vectors</td>
<td>float{n}, double{n}, short{n}, ushort{n}, byte{n}, octet{n}, int{n}, uint{n}</td>
</tr>
<tr>
<td>Matrices</td>
<td>fmat, dmat</td>
</tr>
<tr>
<td>Literals</td>
<td>string, bool</td>
</tr>
<tr>
<td>Images</td>
<td>jpg, png</td>
</tr>
</tbody>
</table>
VCoRE
Attribute

- Attributes can be queried via
  - http://server/poolname/element/id/data/foo

- An attribute has a description
  - http://server/poolname/element/id/data/foo/desc

- Query information about data
  - Name, input format, output format, internal type, length and cardinality

- ... and content
  - http://server/poolname/element/id/data/foo/data
Sample attribute descriptor JSON

http://server/pool/element/foo/data/bar/desc.json

{
    "name" : "position",
    "type" : "float3",
    "length" : "1",
    "cardinality" : "single",
    "input" : [ "float3", "double3", "float", "double" ],
    "output" : [ "float3", "double3" ],
    "version" : 0
}
VCoRE
Sample attribute descriptor JSON

http://localhost/InstantIO/element/TUIO/data/sequenceID/desc.json

{
  "name" : "sequenceID",
  "type" : "uint",
  "length" : "1",
  "cardinality" : "single",
  "input" : [ ],
  "output" : [ "uint", "string" ],
  "version" : 0
}
- Use mimetype to identify query format!
  - `http://server/poolname/element/id/data/foo/data.json`
  - `http://server/poolname/element/id/data/foo/data.fmat`
  - `http://server/poolname/element/id/data/foo/data.float3`
  - `http://server/poolname/element/id/data/foo/data.png`

- For instance:
  - `json/xml`: get textual representation
  - `float3`: internal format needs to be converted to float3 and then sent (for instance cast from double2)
  - **No mimetype**: send data in default/raw native format
  - `png`: Convert data to image first
http://localhost/InstantI/O/element/TUIO/data/sequenceID/desc.xml

<VCData version="" cardinality="single" name="sequenceID" type="uint" length="1" version="0">
  <Input></Input>
  <Output>"uint", "string"</Output>
</VCData>
## VCoRE

### Modifying data via HTTP

<table>
<thead>
<tr>
<th>Resource</th>
<th>GET</th>
<th>PUT</th>
<th>POST</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attribute</strong></td>
<td>- Request JSON descriptor</td>
<td>- Replace value</td>
<td>- Replace value</td>
<td>- Reset value to default state</td>
</tr>
<tr>
<td>(single)</td>
<td>- Retrieve value in given format</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/data/att/desc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/data/att/data(.format)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Attribute</strong></td>
<td>- Request JSON descriptor</td>
<td>- Replace value</td>
<td>- Replace value</td>
<td>- Reset value to default state</td>
</tr>
<tr>
<td>(multi)</td>
<td>- Retrieve entire array in given format</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/data/att/desc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/data/att/data(.format)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Element</strong></td>
<td>- Request JSON descriptor</td>
<td>- Replace entire array</td>
<td>- Replace entire array</td>
<td>- Reset all values to default state</td>
</tr>
<tr>
<td>(single)</td>
<td>- Retrieve Element in given format (application dependent)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/desc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/data(.format)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Element</strong></td>
<td>- Request JSON descriptor</td>
<td>- Reset existing Element</td>
<td>- Reset existing Element or create new Element id</td>
<td>- Remove the Element from the internal representation</td>
</tr>
<tr>
<td>(multi)</td>
<td>- Append new Element</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/desc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/id/data(.format)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Append new Element</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Remove entire Element collection from internal representation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VCoRE

Performance

- Query as text, e.g. JSON/XML/string (slow)
  - http://server/pool/element/id/data/attribute1/data.xml

- Query with XMLHttpRequest as ArrayBuffer
  - http://server/pool/element/id/data/attribute1/data.float3

- Query data as WebSocket
  - ws://server/pool/element/id/data/attribute1/data.float3
  - Allows PUSH model (observer on data change)
Applications usually not just pile of data

Grouping into **Elements** which have

- Attributes (data)
- ID, Type
- “Relations”
  - List of connections of some sort to other elements
  - Not specified on purpose, application-specific!
  - Relations are defined by roles: child, parent, core ...
VCoRE
Element descriptor sample

http://server/pool/element/foo/desc.json

{
    "id" : "identifier",
    "type" : "node",
    "cardinality" : "single",
    "length" : 1,
    "attributes" : [ "bar", "bar1", ... ],
    "relations" : [ "type1", "type2", ... ],
    "input" : [ "format1", "format2", ... ],
    "output" : [ "format1", "format2", ... ]
}
VCoRE

Sample element descriptor JSON

http://localhost/InstantIO/element/TUIO/desc.json

{
  id: "TUIO",
  type: "TUIO",
  relations: [ ],
  ...
}
Sample element descriptor JSON

http://localhost/Avalon/element/mybox/desc.json

```json
{
  id: "box",
  type: "Shape",
  attributes: [ ... ],
  relations: [
    "appearance", "children", "parent", "geometry", ...
  ]
}
```
Multi/Single cardinality

- Cardinality indicates array of elements/attributes
- Length indicates number of elements/attributes
- If cardinality is multi, use index to get element/attribute

http://localhost/Avalon/element/mybox/data/children/0/desc.xml
http://localhost/Avalon/element/mybox/data/children/1/desc.xml
http://localhost/Avalon/element/mybox/data/children/.../desc.xml
What is it good for?

Hardware supported by Instant Reality
What is it good for?

Oculus Rift in Instant Reality
REST interface

Open issues

- Are the default types complete?
- Multithreading and synchronization
  - Multiple clients writing to the same Attribute
- Synchronized access to Attributes
  - Coherency of data sources (e.g. a Leap frame contains hand + fingers belonging together)
- Query multiple Attributes
  - Don’t want to open a websocket to each attribute independently
- More queries other than “element”
Thanks for listening.

Tutorial, demo- and testing-data available at http://www.instantreality.org
Questions?

tobias.franke@igd.fraunhofer.de

@thefranke