



XML3D and Xflow

An Overview and Demos

Philipp Slusallek

German Research Center for Artificial Intelligence (DFKI)
Intel Visual Computing Institute
Saarland University

Visual Computing Institute

UNIVERSITÄT DES SAARLANDES

Motivation

- 3D graphics is becoming a commodity
  - **High-Performance Graphics** – even for mobile
  - **3D-Stereo and 3D-Input** – even for consumers
  - **Fast Internet Connections** – even on wireless
- 3D graphics is still a broadcast medium
  - Almost exclusively focused on games (and CAD)
  - Specialized content for specialized engines (and v.v.)
- 3D graphics must be sharable across the Web
  - Must work for non-experts – its a means, not a goal
  - Make it available to millions of Web developers

► **Need to adapt 3D graphics for Web**

Motivation

- Compare to Video Technology
  - Technology had been since the mid 1990ies
    - RealVideo, MPEG, MMX, …
  - But nothing happened for almost a decade
- The Web changed the (Video) World
  - YouTube (2005)
    - Allowed *anyone* to *easily* add video to the *their* Web page
  - Everyone can create, share, experience video
    - Today: 2 billion views per day
    - Revenue of $1.1 Billion (target for 2011)

► **Can we repeat something similar for 3D?**

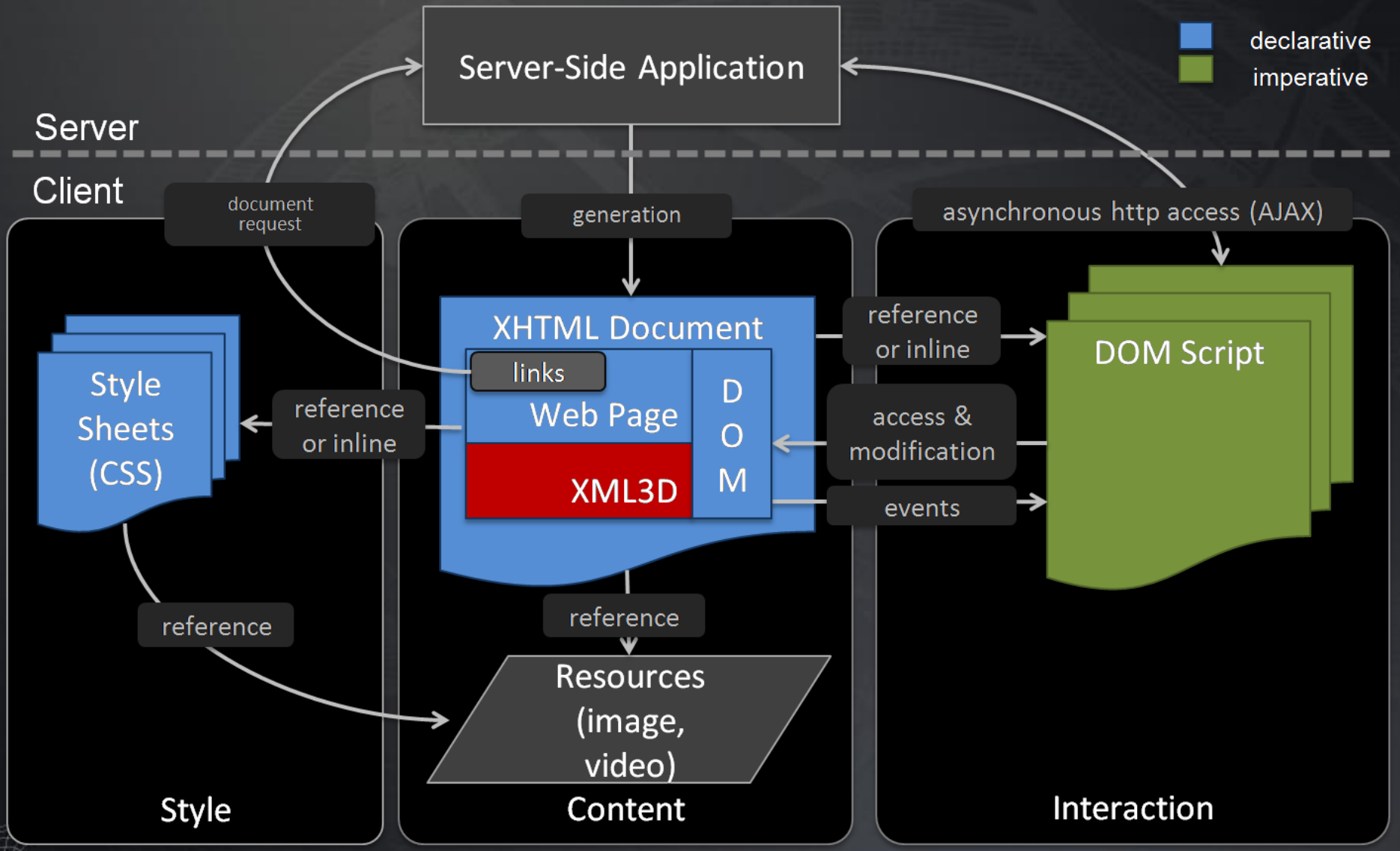
Proposal: Declarative 3D On The Web

- Make it easy to add 3D to Web pages
  - Fully integrate 3D content into HTML5 *documents*
    - Interactive 3D graphics as first class DOM objects
  - Reuse existing Web technology wherever possible
    - Avoid barrier to entry – make Web developers feel at home
    - Do not add new concepts, unless absolutely necessary
- Jump start 3D on the Web
  - Freely provide necessary technology
    - Specification of HTML extensions & standardization
    - Native browser & JS implementations, server side, …
  - Tutorials, examples, hosting, …
  - Joint initiative with research and industry

XML3D in Projects

- EU Future Internet PPP (Architecture Board)
  - Generic Enabler with Disney Research, BBC, Technicolor, Sintef, …
- EU Verve & EU SocialSensor
  - INRIA, Trinity College, several hospitals, …
- EU EIT
  - Most top EU companies & research parters
- BMBF German Spitzencluster (with X3DOM)
  - SAP, Software AG, Fraunhofer, …
- BMBF Ecouss
  - Cray, HLRS Stuttgart, RTT, Beiersdorf, …
- Industry: Caigos & Interactive Software
  - 3D-GIS Extension, IAA Presentation
- Public: Cultural Heritage
  - Museum of Saarlouis, 3D-Kiosk
- Intel Visual Computing Institute & ISTC-VC & Intel
  - Main Project Partner, Links to Stanford, Links to Business Units

Structure of Content: 2D Web



(Re-)use of Web technologies

- HTML / SVG
  - <img>, <video>, <canvas>, <iframe>
    - To define (interactive) textures.
  - <script> for programmable parts
    - Shader, vertex processing …
  - <defs> to define objects reused later
    - As in SVG
  - Event attributes: onmouseover, onclick …
    - Works the same in 3D

(Re-)use of Web technologies

- DOM
  - Event and scripting mechanisms
  - Run time environment
- CSS
  - Separate e.g. geometry from the apperance
  - CSS3 Proposals
    - 3D Transforms, Animations
- Others
  - EXI for external binary data
  - Device API
  - SMIL, XBL

XML3D Elements

- <xml3d>
  - Canvas for XML3D content
  - Embed into (X)HTML
- <defs>
  - Definition of resources
- <group>
  - Structuring and transformation hierarchy
- <transform>
  - Alternative transformation syntax, can be referenced via CSS

```
<body>
<h1>XHTML Document</h1>
<xml3d style="width: 640px; height: 480px"
       xmlns="http://www.xml3d.org/2009/xml3d">
      <!-- ... -->
</xml3d>
</body>
```

```
<xml3d ... >
<defs>
      <transform id="xfm1" translation="3 0 1" />
      <transform id="xfm2" scale="2 2 2" />
</defs>

<group style="transform: url(#xfm1)" >
      <group cass="scaleMe" >
            <mesh ... />
      </group>
      <group style="transform: scale3d(3,1,3)"
            <mesh ... />
      </group>
</group>
</xml3d>
```

XML3D Generic Data

- <data>
  - Collections of typed & named arrays (float int, …)
  - Maps well to HW
    - GPU buffers
    - Arguments to shaders
  - Allows external references
    - E.g. in binary EXI format
  - Supports easy reuse
  - Used also for XFlow

```
<xml3d ... >
<defs>
    <data id="data01">
        <int name="index" >
            0 1 2 0 1 3 1 3 4 ... </int>
        <float3 name="position" >
            0.12 3.1 2.69 ... </float3>
        <float3 name="normal" >
            1 0 0 ...</float3>
    </data>
    <data id="heatData">
        <data src="#data01"/>
        <float name="temperature" >
            32.4 19.2 -42 ...</float>
    </data>
</defs>
<!-- ... -->
</xml3d>
```

XML3D Elements

- <mesh>
  - Format for triangles, triangleStrips etc.
  - Generic mesh attributes
  - Easy to map to hardware buffers
  - Meshes that reference same data source are geometry instances

```
<xml3d ... >
<mesh type="triangles">
      <data id="myData">
            <int name="index" >
                  0 1 2 0 1 3 1 3 4 ... </int>
            <float3 name="position" >
                  0.12 3.1 2.69 ... </float3>
            <float3 name="normal" >
                  1 0 0 ...</float3>
      </data>
</mesh>
<group …>
      <mesh type="triangles" src="#myData"/>
</group>
<!-- ... -->
</xml3d>
```

XML3D Elements

- <shader>
  - Material description
  - Like paint/stroke in SVG
    - But: Arbitrary parameters
  - Use CSS to assign material to geometry
    - References shader script
    - Predefined common shaders (via URN)
  - Portable material descriptions (AnySL)

```
<xml3d ... >
<defs>
      <script id="tex_script"
            type="text/anysl:RSL">
            <!-- ... -->
      </script>

      <shader id=”shader1” script=”#tex_script” >
            <data id=“shaderData”>
                  <texture name=”diffuseTexture”>
                        <img src=”metal.png”/>
                  </texture>
            </data>
      </shader>

      <shader id=“phong”
            script=“urn:xml3d:shader:phong”>
            <data src=“#shaderData”/>
      </shader>
</defs>

<group style=”shader: url(#shader1)”
      <mesh > <!-- ... -> </mesh>
</group>
</xml3d>
```

XML3D Elements

- <view>
  - Specify a 3D viewpoint
  - May be the destination of a link
- <light>
  - Specify lights
  - References light shader
  - Fixed function and programmable

```
<xml3d ... >
        <!-- ... -->
        <group class="animatedView" >
                <view ... />
        </group>
        <light intensity="0.5"/>
</xml3d>
```

Interaction

- DOM Scripting
  - Well known to Web developers
  - Arbitrary scene graph modifications
  - Animations via triggering redraws
- DOM Events
  - Data via attributes
  - 3D specific data

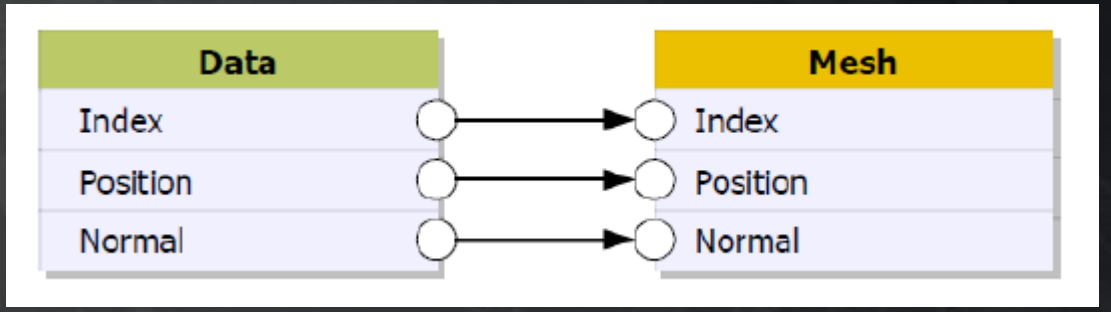
```
<xml3d ... >
<defs>
    <transform id="xfm1" translation="0 0 0" />
</defs>

<group style="transform: url(#transform1)" >
    <mesh type="triangles"
        onclick="alert('Interaction!')">
     ...
    </mesh>
</group>

</xml3d>

<script type="text/javascript" >
    function moveMesh(){
        var xfm = document.getElementById('xfm1');
        xfm.translation.x += 0.1;
    }
    setInterval("moveMesh()", 50);
</script>
```

XML3D

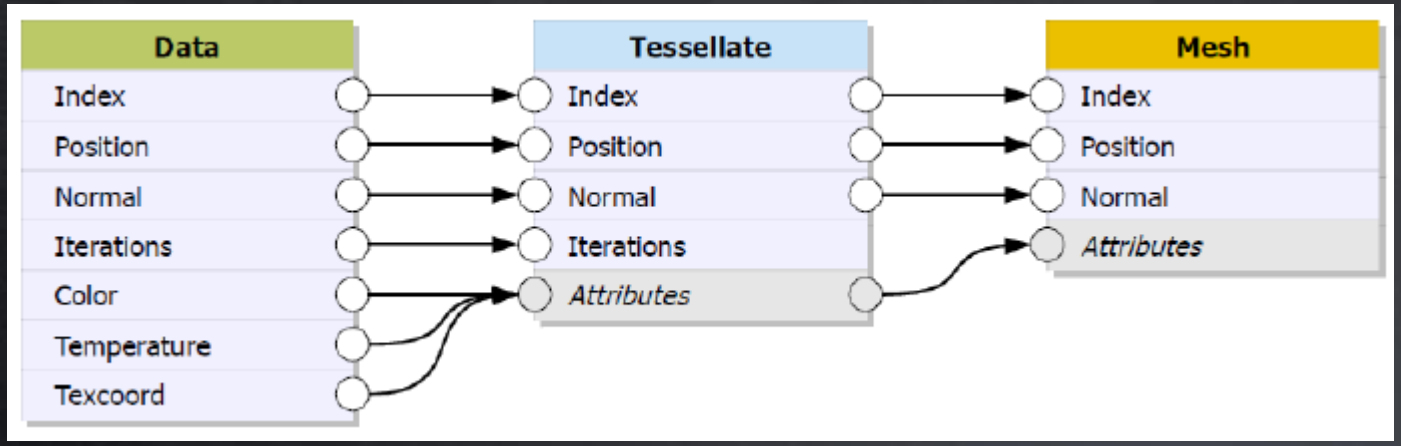
- Interactive 3D graphics
  - Few new elements
  - DOM is scene graph and run time
- Portable Format
  - Portable geometry and material descriptions
- Full support for programmable GPUs
  - Programmable shading (materials, animation, …)
  - HW-oriented data formats
- But what about Animations & Dynamics?
  - Vertex shaders, image (pre/post-)processing

XFlow: Data Processing for XML3D

- Data specification
  - Typically used to provide data for meshes and shaders (surface, lights, geometry, …)



- Simply allow processing elements in the chain
  - Shaders that take data and provides new data



XFlow: Different XFlow Elements

- XFlow Processing Elements
  - Map XML3D data elements to new data sets
- Predefined Fixed-Functions (URN)
  - Geometry shaders (OGL/DX/RT)
    - Morphing, Skeletal Animation, …
  - Post Processing, Tone Mapping, …
- Programmable Elements
  - But which language: Javascript (slow), Plugins (too awkward), CUDA/OpenCL (too HW specific)
  - AnySL: Portable representation & embedded compiler
    - Compiled to x86/SSE, PTX/CUDA, OpenCL, GLSL, …

Beyond Static Geometry

- Different Types of Animations
  - Hierarchical Rigid Motion – OK
  - Morphing – **NOT OK**
  - Skinning – **NOT OK**
  - Tesselation and many more – **NOT OK**
- Different Execution Environments
  - CPU (x86, SSE, AVX, …)
  - GPU (GLSL, OpenCL, CUDA/PTX, others)
  - Avoid having to go back to GPGPU
  - Should make best use of hardware
- Provide Portable Solution

Go for Generic Solutions

- Avoid specializes elements
  - Like <hanim>, <skinning>, <bone>, <morphing>, …
  - No blow up in new concepts
- Generic solution for data parallel processing
  - Like <data script="…" >
  - At least as powerful as GLSL/HLSL/…
  - But platform-independent
- XFlow: Approach for declarative data processing
  - Description of flow graph in HTML/XML3D
    - Composition of data field (easy to re-use and route)
    - Scripts for processing code (where necessary)

A Simple Mesh

```
<mesh type="triangles" >
   <int name="index" >0 1 2 1 2 3 ...</int>
   <float3 name="position" >1.0 2.0 -0.2 ...</float3>
   <float3 name="normal"   >0.0 1.0 0.0 ...</float3>
</mesh>
```

=

```
<mesh type="triangles" >
   <data>
      <int name="index" >0 1 2 1 2 3 ...</int>
      <float3 name="position" >1.0 2.0 -0.2 ...</float3>
      <float3 name="normal"   >0.0 1.0 0.0 ...</float3>
   </data>
</mesh>
```

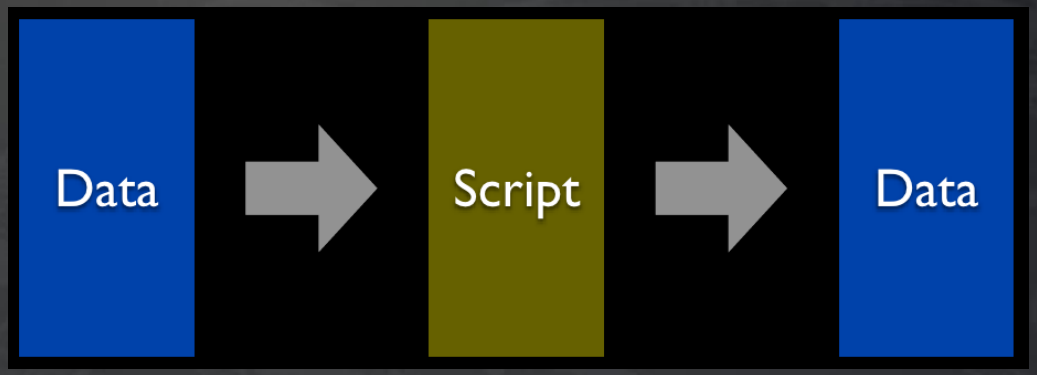
Instantiate with <data>

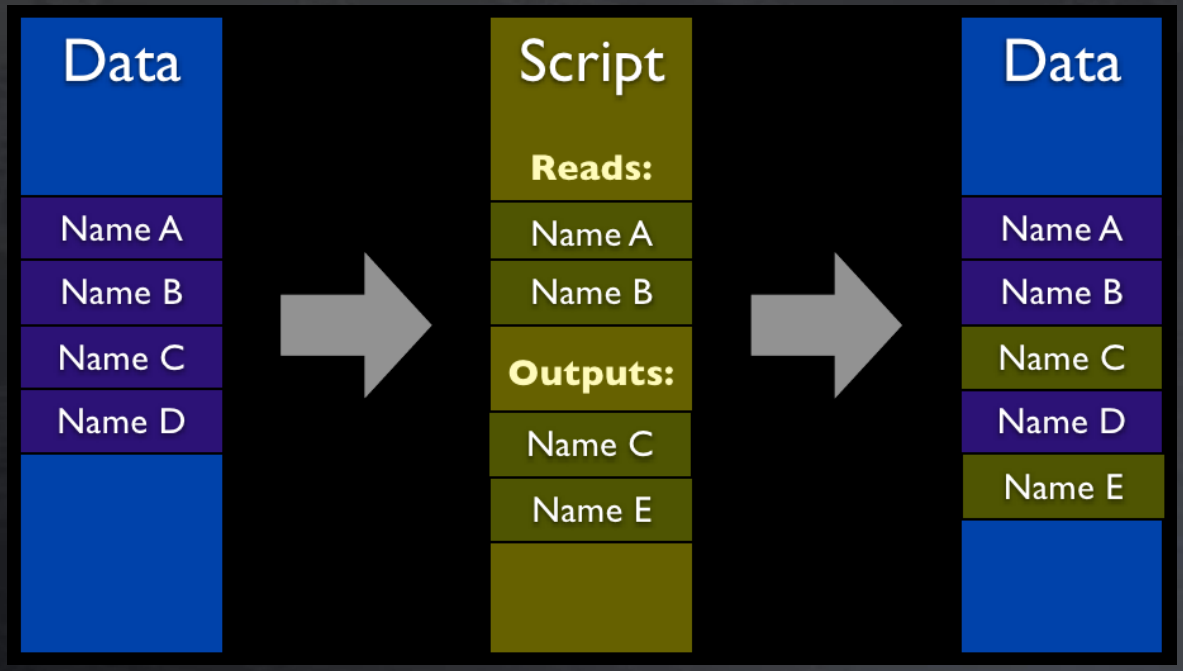
```
<defs>
   <data id=”meshData” >
      <int name=”index” >0 1 2 1 2 3 ...</int>
      <float3 name=”position” >1.0 2.0 -0.2 ...</float3>
      <float3 name=”normal”   >0.0 1.0 0.0 ...</float3>
   </data>
</defs>
...

<mesh type=”triangles” src=”#meshData” />
<group style=”transform: ... “ >
   <mesh type=”triangles” src=”#meshData” />
</group>
```

Partial Re-Use of Data

```
<defs>
   <data id="basicMeshData" >
      <int name="index" >0 1 2 1 2 3 ...</int>
      <float3 name="position" >1.0 2.0 -0.2 ...</float3>
      <float3 name="normal"   >0.0 1.0 0.0 ...</float3>
   </data>
</defs>
...
<mesh type="triangles">
   <data src="#basicMeshData" />
   <float3 name="color" >1.0 0.7 0.1 ...</float>
</mesh>
...
<mesh type="triangles">
   <data src="#basicMeshData" />
   <float3 name="color" >0.2 0.7 0.9 ...</float>
</mesh>
```

Data Scripts

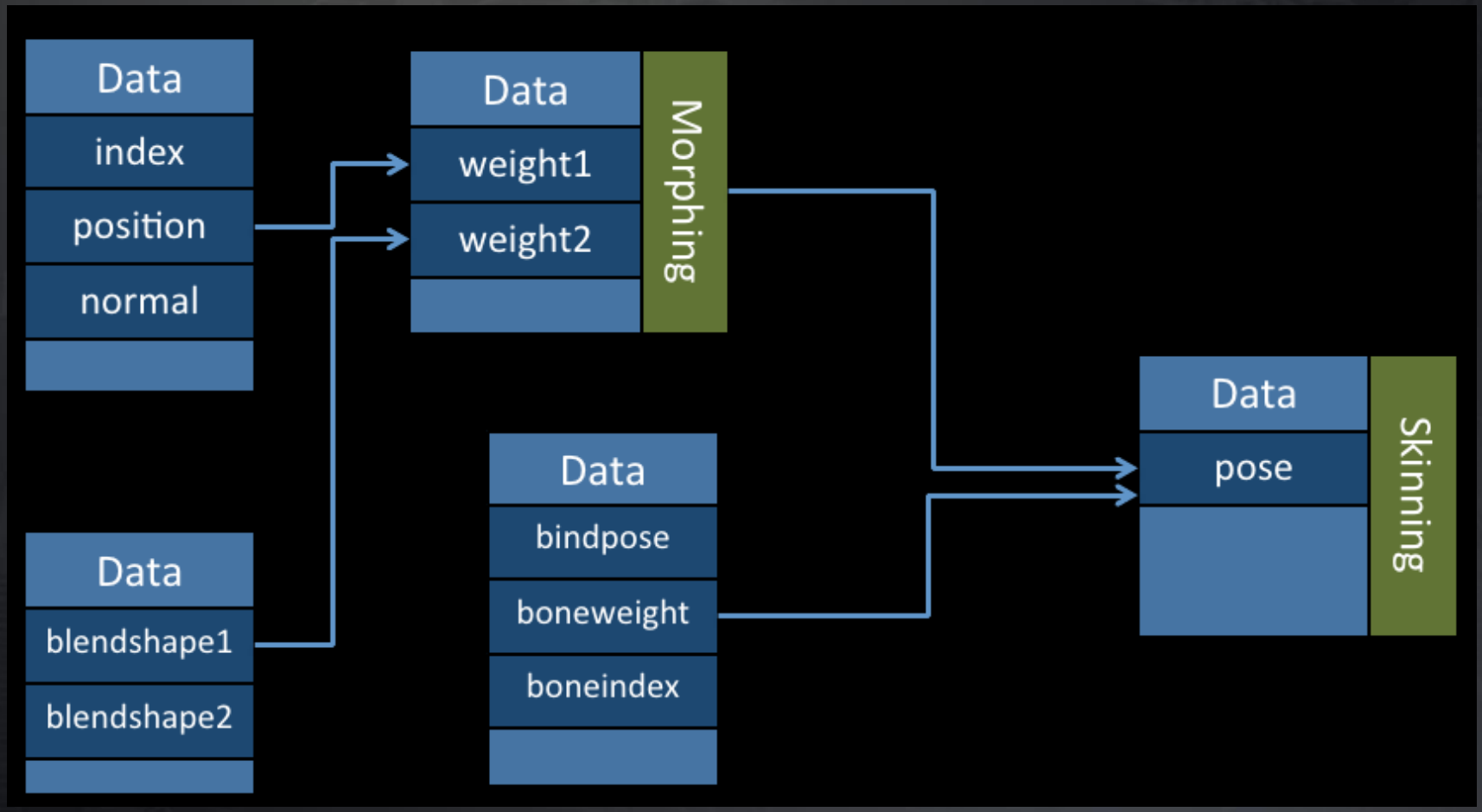




Script Attached to Data

```
<data id="meshData" script="morphing.xxx" >
   <int name="index" >0 1 2 1 2 3 ...</int>
   <float3 name="position" >1.0 2.0 -0.2 ...</float3>
   <float3 name="normal"   >0.0 1.0 0.0 ...</float3>
   <float3 name="blendshape1"   >1.5 2.3 -0.4 ...</float3>
   <float3 name="blendshape2"   >1.2 1.9 0.1 ...</float3>
   <float name="weight1"   >0.3</float3>
   <float name="weight2"   >0.5</float3>
</data>
```

More Complex Data Flow



Compactness of Data Flow Spec

```
<mesh type="triangles" >
   <data script="skinning.xxx" >
      <data script="morphing.xxx" >
        <data src="#meshData" />
        <data src="#blendData" />
        <float3 name="weight1" >0.8</float3>
        <float3 name="weight2" >0.2</float3>
      </data>
      <data src="#skinningData" />
      <float4x4 name="pose">...</float4x4>
   </data>
</mesh>
```

More About Data Scripts

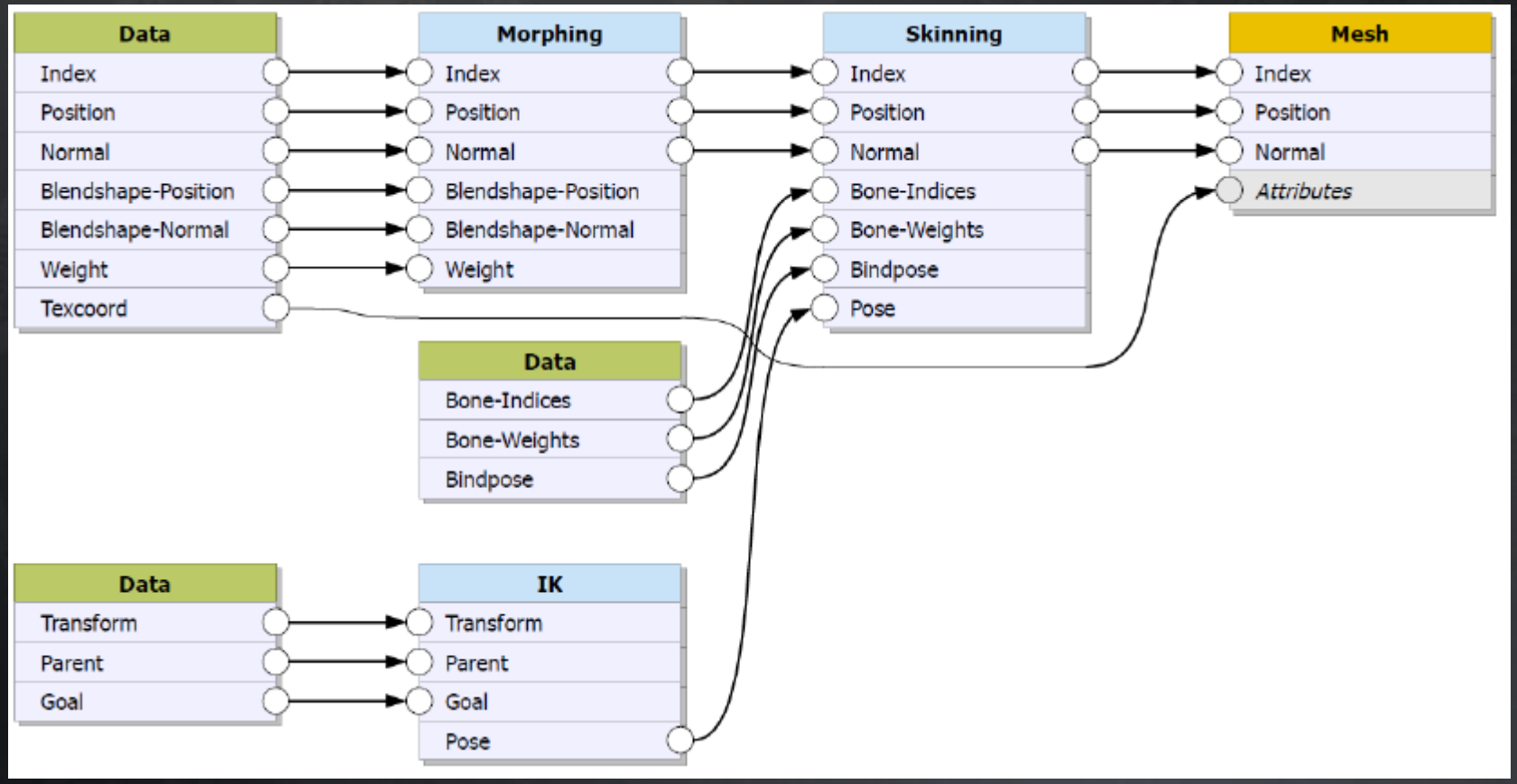
- Platform independent description
  - Use AnySL
  - Safe exposure of parallel computing via OpenCL / CUDA
- Script execution not bound to specific hardware
  - Can be executed on GPU or CPU depending on context and renderer

External Data

- Natural extension to XFlow concept
  - <data src=”external.xxx” />
    - Any format that can be converted to data representation
    - May include Xflow script for decompression, etc.

Example: Character Animation

- All shaders are cross-platform, cross-renderer
  - HW accelerated (CUDA, OpenCL, DXCompute)



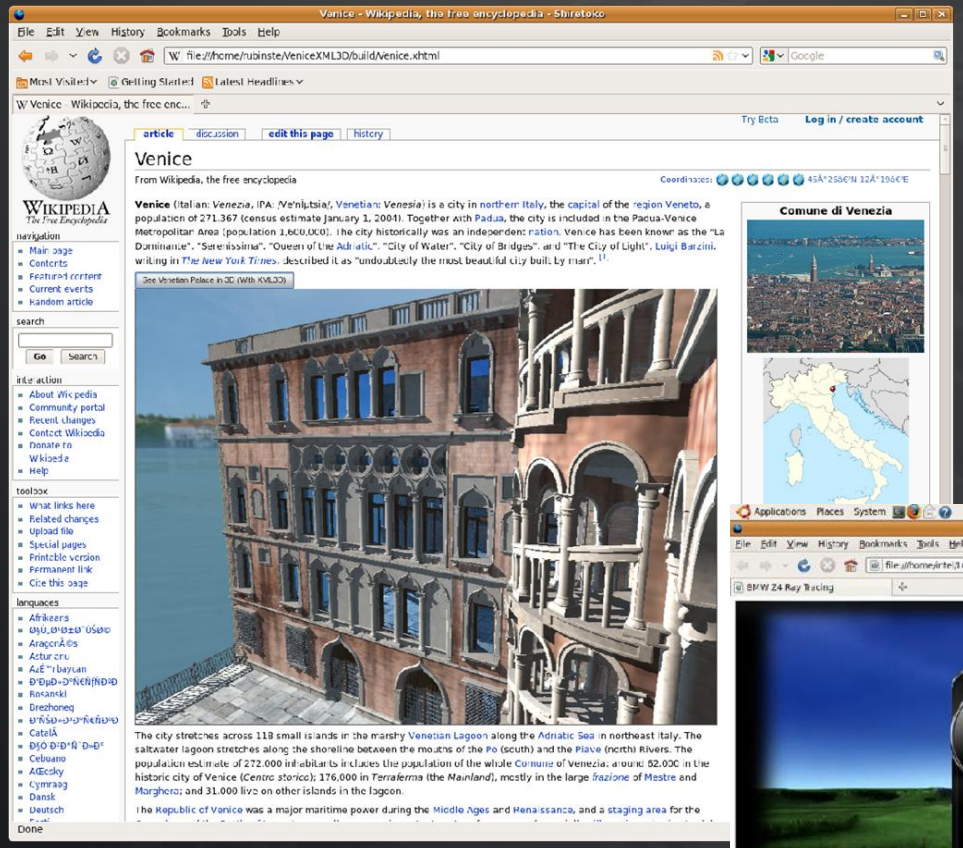
XML3D Implementations

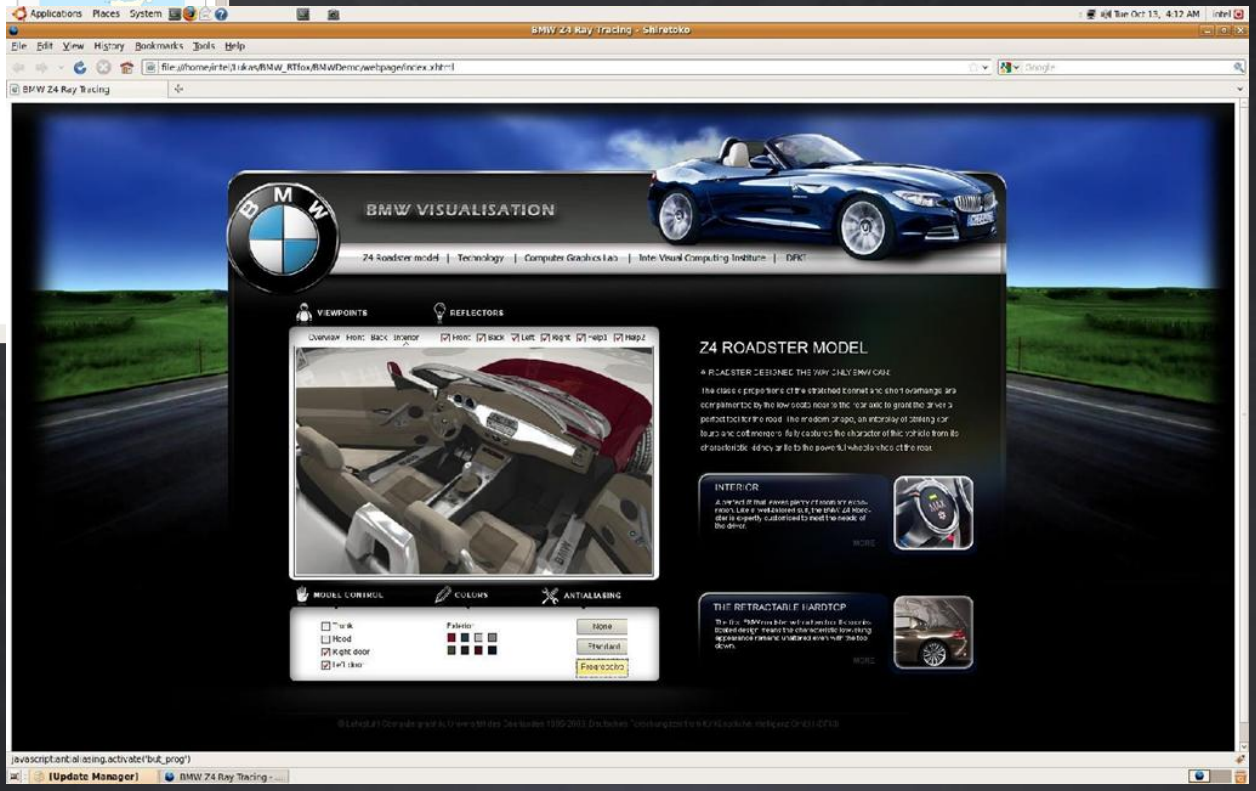
- Native implementations
  - Mozilla Firefox & Google Chromium
  - Renderer:
    - Realtime Ray Tracing & native OpenGL 4.X
- WebGL/JS-based renderer
  - Limitation due to OpenGL ES, CSS, JS
- Standardization at W3C
  - Creation of Incubator/Community group
- Freely available (alpha version):
  http://www.xml3d.org

XML3D Content Pipeline

- Exporters
  - Cinema 4D
  - Blender
- Converters
  - COLLADA
  - X3D / VRML
- Application integration
  - Caigos, BallView, 3D-Kiosk, …
- More to come…

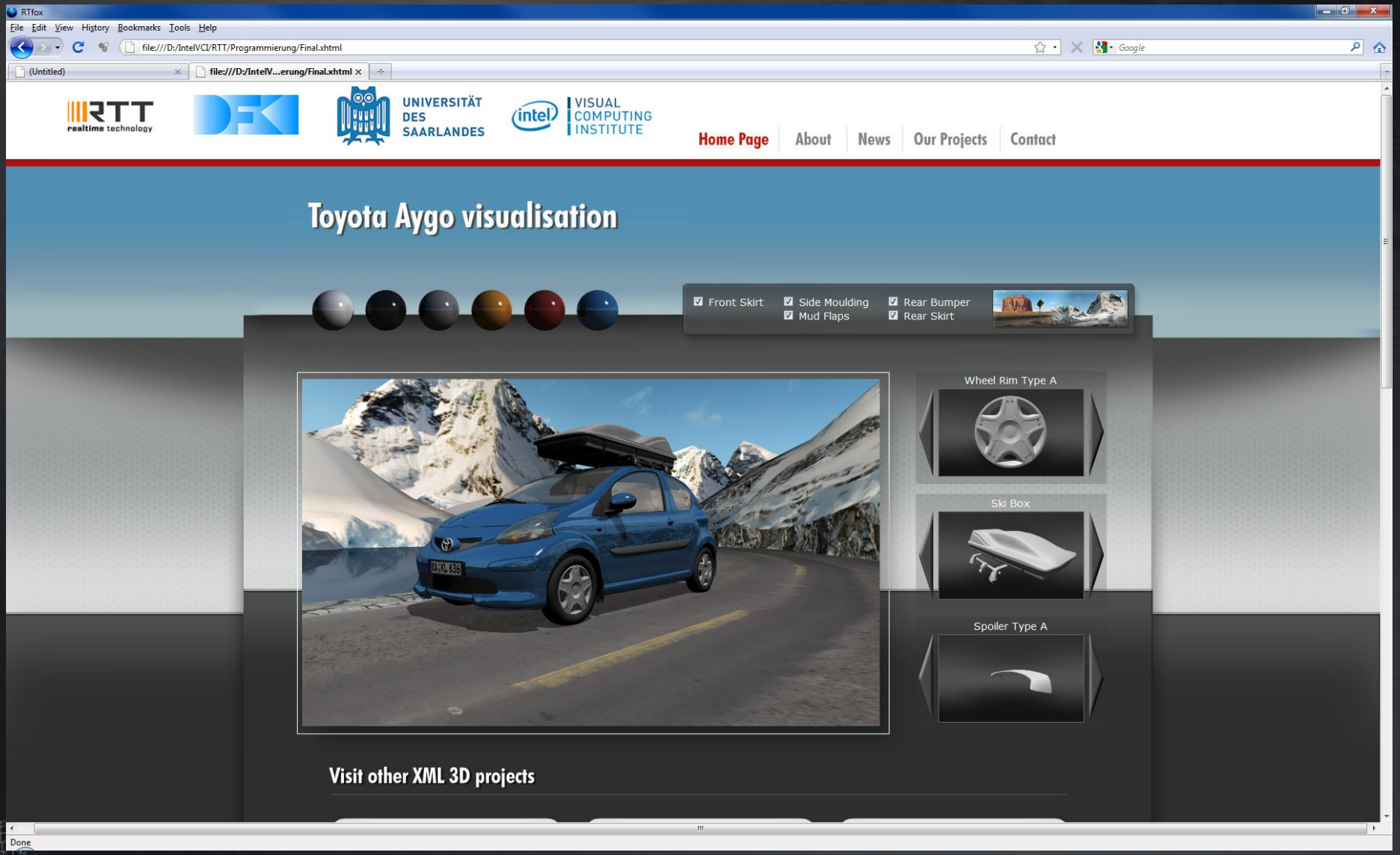
Firefox with XML3D



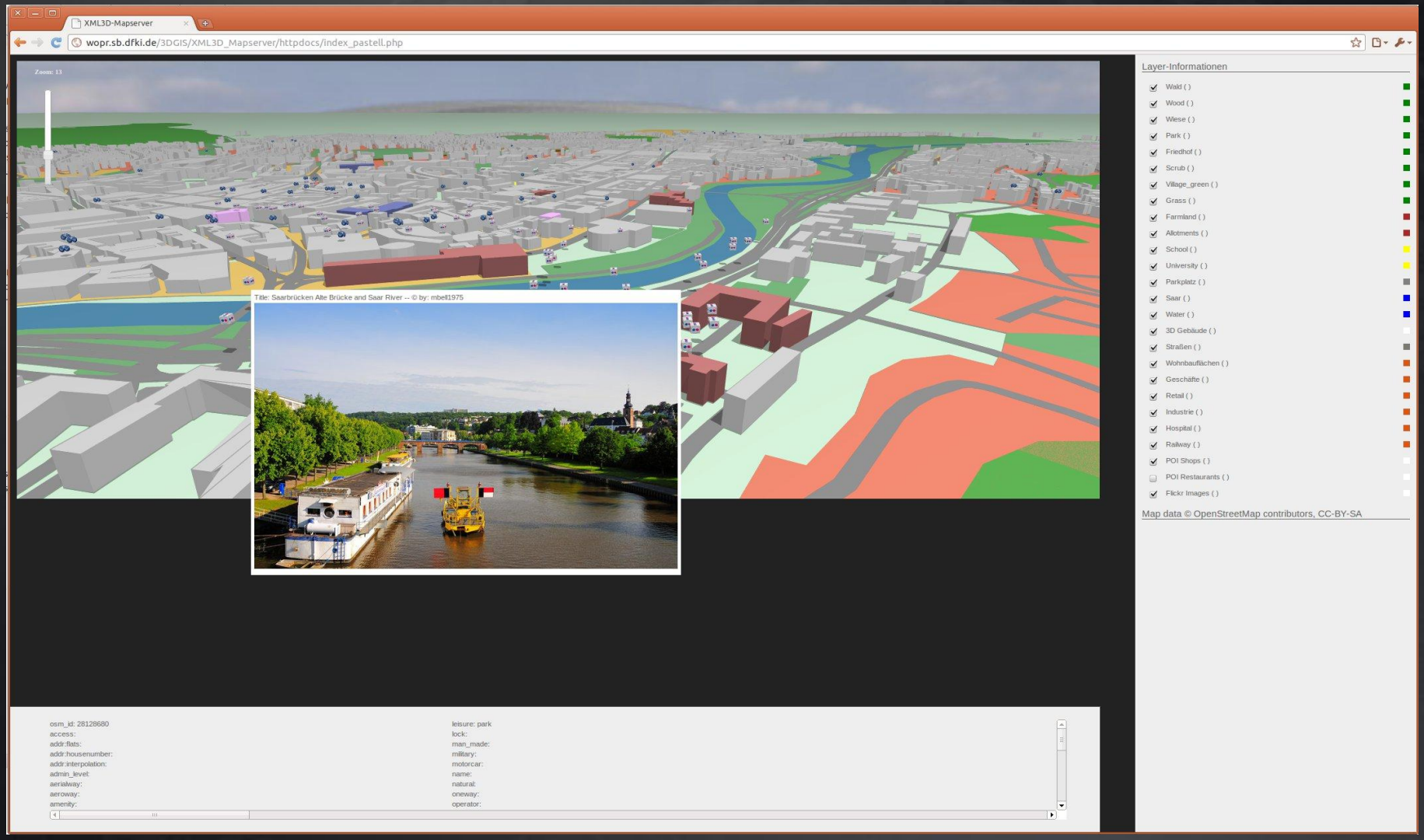


AGENTS and SIMULATED REALITY

XML3D: 3D on the Web



More XML3D Examples



Interactive Web-GIS in 3D from OpenStreetMap & GIS data

In cooperation with Caigos (SME), Dell

AGENTS and SIMULATED REALITY

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

More XML3D Examples

More XML3D Examples

AGENTS and SIMULATED REALITY

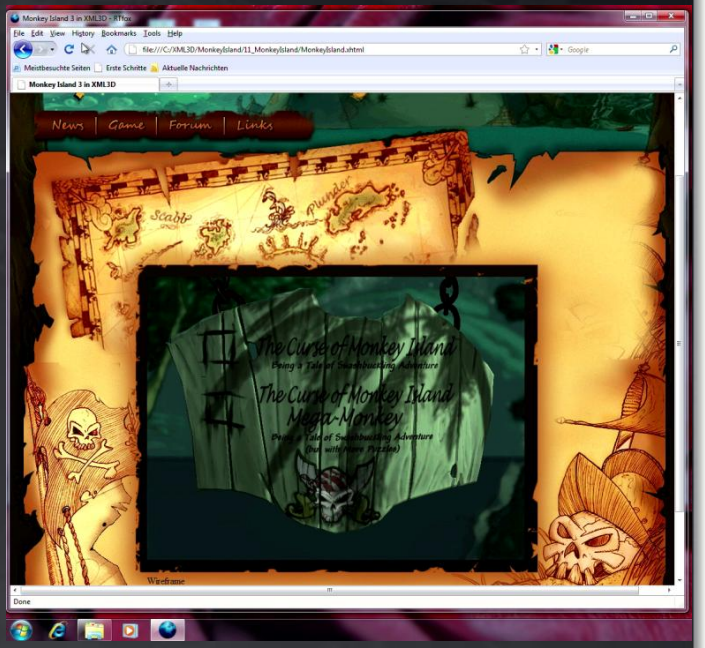
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

More XML3D Examples

More XML3D Examples

XML3D for Cultural Heritage

More XML3D Examples



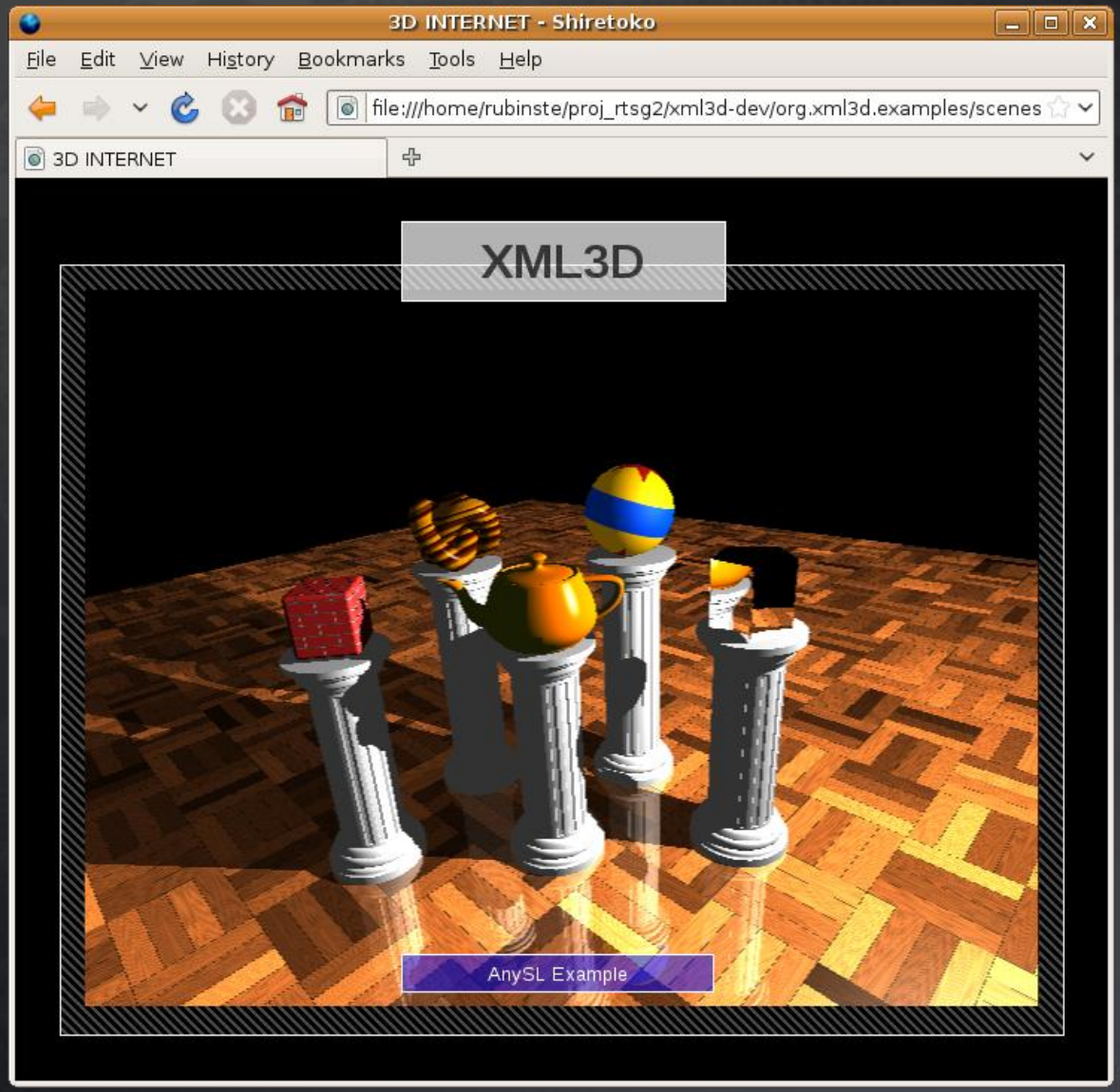
Curse of Monkey Island

3D version of pupolar 2D game

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

More XML3D Examples

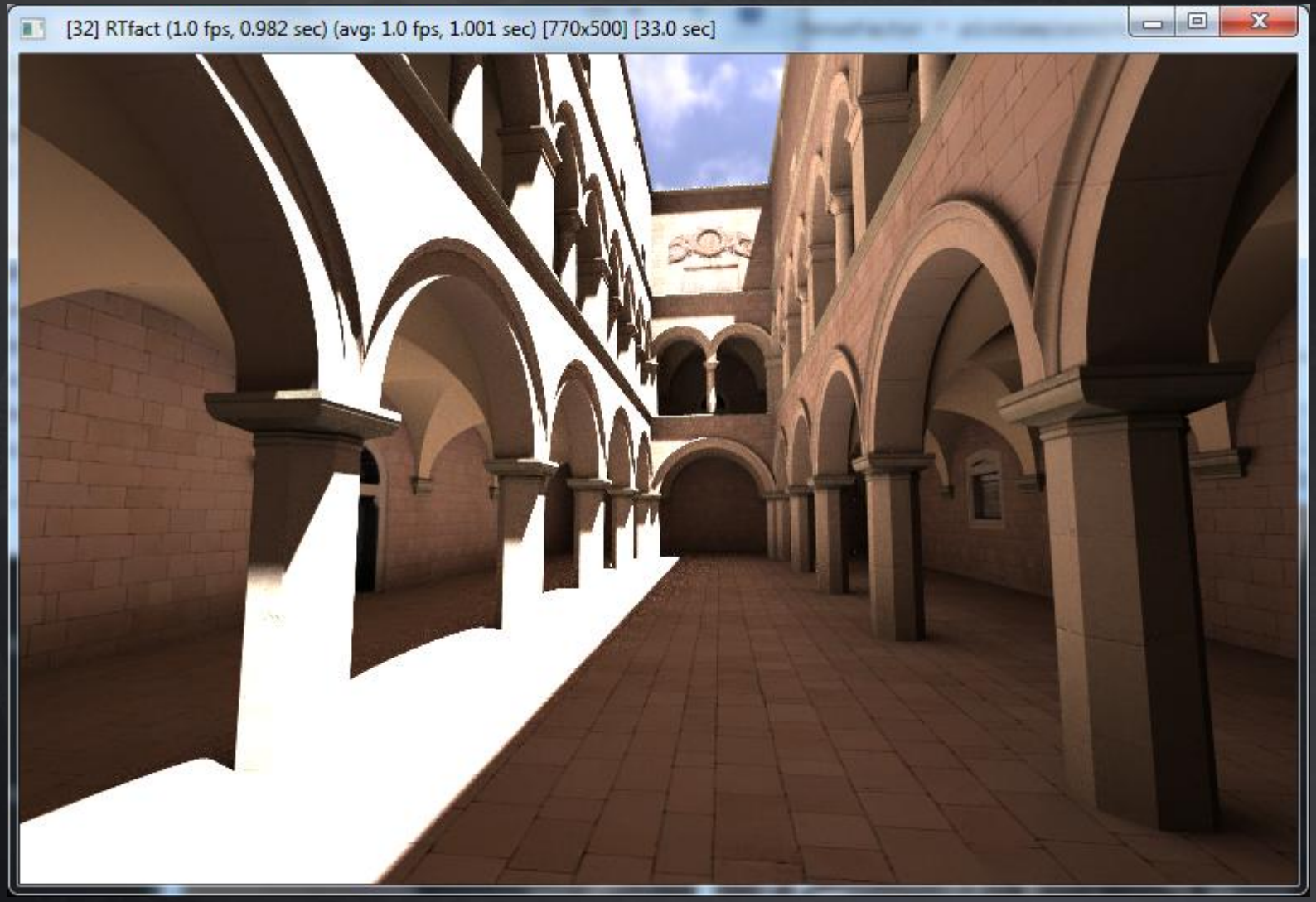
AnySL: Interactive RenderMan in Your Web Browser



AGENTS and SIMULATED REALITY

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Interactive Global Illumination



Interactive Global Illumination

Summary

- Declarative 3D on the Web with XML3D
  - 3D will be a hot topic – particularly on the Web
  - W3C should play major role for *declarative 3D*
  - Upcoming W3C Incubator Group
  - Three fully working prototypes using XML3D
    - Native Mozilla Firefox
    - Native Google Chromium
    - Portable WebGL/Javascript implementation
- Ready to start on your projects today!