X3DOM: New Ways of Presenting 3D CH Content on the Web using Declarative Approaches

Yvonne Jung
Application Examples

Virtual Engineering and Cultural Heritage on the Web
3D Information inside the Web

- Websites have become Web applications

- Increasing interest in 3D for
  - Product presentation
  - Experiencing Cultural Heritage
  - Visualization of abstract information
  - Supporting decision making, e.g. in Virtual Engineering
  - ...

- Enhancing user experience with more sophisticated visualizations
  - Interactive 3D instead of videos
Technology Trends

3D Data and Documents
- 3D Printer
- 3D Scanner
- Augmented/Mixed Reality
- Geo Data

Cloud-based Rendering

Web-based Engineering
- Cloud-based Rendering
- HTML5
- Titanium
- PhoneGap
- Peper NaCl

Convergence of Application Platforms
- Windows
- Chrome OS
- iOS
- Android

Divergence of System Platforms
- Amazon Kindle
- Mac OS
- Windows Phone
- Windows

Touch Table

Web Service Architectures
- Store apps
- Cloud Bases

4
WebGL: OpenGL + GLSL on the Web

- JavaScript Binding for OpenGL ES 2.0 in Web Browser
  - Firefox, Chrome, Safari, Opera, IE11
  - Desktop and mobile systems
- Only GLSL shader-based, no fixed function pipeline
  - No variables from GL state
  - No matrix stack, etc.
- HTML5 `<canvas>` element provides 3D rendering context
  - `gl = canvas.getContext('webgl');`
- API calls via GL object
  - `gl.viewport(0, 0, 400, 300);`
Why 3D Graphics on the Web?

• Native Web Browser integration via WebGL
  – Plugin-free 3D
    • No plugin-specific scripting interface
    • No issues with user permissions, installation, security
  – OS independent, especially on mobile devices
    • Cluttered: Windows Phone, Android, iOS, Symbian...
    • Web Browsers for most devices available

• Browser provides complete deployment structure
  – Eases proliferation of technology and accessibility of content
  – WebGL maintained by non-profit Khronos Group
X3D in a Nutshell

- Open ISO standard for dynamic 3D content
  - Developed by Web3D Consortium (www.web3d.org)
- Extensibility is core feature
  - VRML 2.0: 54 nodes; 1 encoding
  - X3D 3.x: > 200 nodes; 3 encodings
- Organized in Components + Profiles
- Uses Scene-graph to connect nodes
  - Directed acyclic graph (DAG)
    - Tree with one root node and no loops
    - Declarative listing of parameters of interest
- X3D runtime parses X3D scene and renders it
  - Provides simulation capabilities for animation + interaction
  - Traverses scene-graph, updates values, and builds image
  - Behavior defined as changing field values of nodes
X3DOM – Declarative (X)3D in HTML5

• X3DOM := X3D + DOM
• DOM-based integration framework for declarative 3D graphics in HTML5
• Seamless integration of 3D contents in Browser
X3DOM – Declarative (X)3D in HTML5

• Brings together both,
  – declarative content design as known from web design
  – “old-school” imperative approaches as known from game engine development
• Allows utilizing well-known JavaScript and DOM infrastructure for 3D
  – Embed “live” scene-graph in the DOM

```html
<html>
  <body>
    <h1>Hello X3DOM World</h1>
    <x3d>
      <scene>
        <shape>
          <box></box>
        </shape>
      </scene>
    </x3d>
  </body>
</html>
```
X3DOM – Declarative (X)3D in HTML5

Declarative
Scene-graph
Part of HTML document
DOM Integration
CSS / Events

Imperative
Procedural API
Drawing context
Flexible

2D
(Final HTML5 spec)

3D
(No W3C spec yet)

Completes today's graphics technologies
Why Declarative 3D in HTML?

• Integration into HTML document instead of closed systems
  – Integrates well with standard Web techniques like DHTML and Ajax
  – Utilizing standard Web APIs for combining content + user interactions
    • No special APIs necessary (such as in game engines)
    – Interoperability: Write once, run anywhere (web/desktop/mobile)

• Declarative, open, human-readable, wraps low-level graphics
  – Declarative content description
    • No expert knowledge required (OpenGL, mathematics,...)
  – Flexible contents (cultural heritage, industry,...)
  – Unify 2D and 3D media development

• Rapid application development
  – Open architectures, also for authoring, and ease of access
  – Open formats enable automated connection of existing data (e.g., geo-information, Flickr) with 3D content
    • Allows “mash-ups” (i.e., recombination of existing contents)
  – Metadata: index and search “content” on WebGL apps?
Getting started with X3DOM
Short recap of HTML

<html>
  <head>
    <title>Meine 3D-Seite</title>
  </head>
  <body>
    <h1>Hallo X3DOM Welt</h1>
    <p>
      Hier erscheint bald eine Box.
    </p>
  </body>
</html>
First, HTML needs to know about (X)3D

```html
<html>
<head>
  <title>Meine 3D-Seite</title>
  <link rel="stylesheet" type="text/css" href="http://www.x3dom.org/download/dev/x3dom.css">
  <script type="text/javascript" src="http://www.x3dom.org/download/dev/x3dom.js">
  </script>
</head>
...

For full set of nodes: http://www.x3dom.org/download/dev/x3dom-full.js
```
3D only works inside the `<X3D>` tag

... 

```html
<body>
  <h1>Hallo X3DOM Welt</h1>
  <p>
    Hier erscheint bald eine Box.
  </p>
  <x3d width="400" height="300">
  </x3d>
</body>
</html>
```
All 3D objects are children of `<scene>`

... 

```html
<body>
  <h1>Hallo X3DOM Welt</h1>
  `<x3d width="400" height="300">`
    `<scene>`
      `<shape>`
        `<box></box>`
      `</shape>`
    `</scene>`
  `</x3d>`
</body>
</html>

Please note: `<appearance>` omitted for brevity
Every object has a <shape>

...<body>
  <h1>Hallo X3DOM Welt</h1>
  <x3d width="400" height="300">
    <scene>
      <shape>
        <box></box>
      </shape>
    </scene>
  </x3d>
</body>
...and a geometry, like e.g. a `<box>`

... 

```html
<body>
  <h1>Hallo X3DOM Welt</h1>
  <x3d width="400" height="300">
    <scene>
      <shape>
        <box></box>
      </shape>
    </scene>
  </x3d>
</body>
</html>
```
...and an <appearance>

```xml
<x3d width="400" height="300">
  <scene>
    <shape>
      <appearance>
        <material diffuseColor="red"/>
      </appearance>
      <box/>
    </shape>
  </scene>
</x3d>
```
...with a (e.g. red) <material>

<x3d width="400" height="300">
    <scene>
        <shape>
            <appearance>
                <material diffuseColor="red"/>
            </material>
            <box></box>
        </shape>
    </scene>
</x3d>
Materials with specular highlights

```xml
<x3d width="400" height="300">
  <scene>
    <shape>
      <appearance>
        <material diffuseColor="red"
          specularColor="#808080">
        </material>
      </appearance>
      <box></box>
    </shape>
  </scene>
</x3d>
```
Change Background
Colors in (R,G,B) with red/green/blue $\in [0,1]$

```xml
<scene>
  <shape>
    <appearance>
      <material diffuseColor="red"
               specularColor="#808080"/>
    </material>
    </appearance>
    <box></box>
  </shape>
  <background skyColor="0 0 0">
  </background>
</scene>
```
Change Style (now using CSS)

- Change Background (external style possible)
  
  ```xml
  <x3d style="background-color: #00F;">
    <scene>
      ...
    </scene>
  </x3d>
  ```

- Change size of `<x3d>` element to full size
  
  ```xml
  <x3d style="margin:0; padding:0; width:100%; height:100%; border:none;">
    <scene>
      ...
    </scene>
  </x3d>
  ```
Geometric base objects

See screenshot – from left to right:

<sphere radius="1.0"/>
<cylinder radius="1.0" height="2.0"/>
<box size="2.0 2.0 2.0"/>
<cone bottomRadius="1.0" height="2.0"/>
<torus innerRadius="0.5" outerRadius="1.0"/>
<plane size="2.0 2.0" subdivision="1 1"/>
Defining own geometries
Example: simple rectangle with <indexedFaceSet>

```xml
<scene>
  <shape>
    <appearance>
      <material diffuseColor="salmon"/>
    </appearance>
    <indexedFaceSet coordIndex="0 1 2 3 -1">
      <coordinate point="2 2 0, 7 2 0, 7 5 0, 2 5 0"/>
    </indexedFaceSet>
  </shape>
  <viewpoint position="0 0 15"/>
</scene>
```
Defining own geometries
Example: simple rectangle with `<indexedFaceSet>`

```xml
<indexedFaceSet coordIndex="0 1 2 3 -1">
    <coordinate point="2 2 0, 7 2 0, 7 5 0, 2 5 0">
    </coordinate>
</indexedFaceSet>
```

• Important building blocks
  – The vertices of a polygon (here “face”), given as `<coordinate>`
  – The index to a vertex, given as list: “coordIndex”
Defining own geometries
Example: simple rectangle with `<indexedFaceSet>`

```xml
<indexedFaceSet coordIndex="0 1 2 3 -1">
  <coordinate point="2 2 0, 7 2 0, 7 5 0, 2 5 0">
  </coordinate>
</indexedFaceSet>
```

• The end of one polygon and the begin of a new one is marked as “-1” in the index array

• This way arbitrarily complex 3D objects can be created
Defining own geometries
Example: simple rectangle with <indexedFaceSet>

```
<indexedFaceSet coordIndex="0 1 2 3 -1">
  <coordinate point="2 2 0, 7 2 0, 7 5 0, 2 5 0">
  </coordinate>
</indexedFaceSet>
```

• The indices (except “-1”) refer to the array position of a 3D coordinate in <coordinate>

• The coordinates of a certain polygon are listed counterclockwise
Two objects in one scene (?!)

```xml
<scene>
  <shape>
    <appearance>
      <material diffuseColor='red'/>
    </appearance>
    <box/>  
  </shape>
  <shape>
    <appearance>
      <material diffuseColor='blue'/>
    </appearance>
    <sphere/>  
  </shape>
</scene>
```

OK

```xml
<scene>
  <shape>
    <appearance>
      <material diffuseColor='red'/>
    </appearance>
    <box/>
  </shape>
  <shape>
    <appearance>
      <material diffuseColor='blue'/>
    </appearance>
    <sphere/>
  </shape>
</scene>
```

???
Two objects in one scene
Problem: both appear at same position

<scene>
  <shape>
    <appearance></appearance>
    <box></box>
  </shape>
  <shape>
    <appearance></appearance>
    <sphere></sphere>
  </shape>
</scene>
Two objects in one scene
Problem: both appear at same position

<scene>
  <shape>
    <appearance></appearance>
    <box></box>
  </shape>
  <shape>
    <appearance></appearance>
    <sphere></sphere>
  </shape>
</scene>

Reason: 3D objects are usually created in coordinate origin and need to be repositioned afterwards
Excursus: (2D) coordinate systems
Object coordinates in image plane (given by x and y)
Excursus: (3D) coordinate systems
Object coordinates in 3D space; z orthogonal on x, y
Two objects in one scene
Now with translation

<transform translation="-2 0 0">
  <shape>
    <appearance>
      <material diffuseColor="red"></material>
    </appearance>
    <box></box>
  </shape>
</transform>

<transform translation="2 0 0">
  <shape>
    <appearance>
      <material diffuseColor="blue"></material>
    </appearance>
    <sphere></sphere>
  </shape>
</transform>
Two objects in one scene
Now with translation

<transform translation="-2 0 0">
  <shape>
    <appearance>
      <material diffuseColor="red"/>
    </appearance>
    <box/>
  </shape>
</transform>

<transform translation="2 0 0">
  <shape>
    <appearance>
      <material diffuseColor="blue"/>
    </appearance>
    <sphere/>
  </shape>
</transform>
The scene-graph: Grouping and transformations

- 3D elements/nodes are often organized hierarchically
  - Directed acyclic graph
- Starting from root node (<scene> element) all other 3D elements are inserted into the “tree” (scene-graph) as children or siblings
  - Geometry in leaf nodes
  - Allows for re-use of nodes
    - Note: tree ≠ graph
- <group> and <transform> elements help to group and reposition objects

```xml
<transform translation="0 0 0"
  rotation="0 1 0 0" scale="1 1 1">
  ...
</transform>
```
DOM Manipulation
Node appending / removal

HTML/X3D code:
```xml
<group id='root'></group>
```
...

JS script to add nodes:
```javascript
root = document.getElementById('root');
trans = document.createElement('Transform');
trans.setAttribute('rotation', '0 1 0 3.14');
root.appendChild(trans);
```

JS script to remove nodes:
```javascript
root.removeChild(trans);
```

JS script to modify attributes (also useful for libs like jQuery):
```javascript
document.getElementById('mat').setAttribute('diffuseColor', 'red');
```
How do I know, which elements can be combined?

```xml
<Transform translation='4 2 1'>
  <Shape>
    <Appearance>
      <Material diffuseColor='1 1 0'>
        </Material>
    </Appearance>
    <IndexedFaceSet coordIndex='0 1 2 3 -1, 3 2 1 0 -1'>
      <Coordinate point='0 0 0, 1 0 0, 1 1 0, 0 1 0'>
        </Coordinate>
    </IndexedFaceSet>
  </Shape>
</Transform>
```
How do I know, which elements can be combined?

→ Back to the roots: VRML

```
Transform {
    translation 4 2 1
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor 1 1 0
                }
            }
            geometry IndexedFaceSet {
                coord Coordinate {
                    point [ 0 0 0, 1 0 0, 1 1 0, 0 1 0 ]
                }
                coordIndex [ 0 1 2 3 -1, 3 2 1 0 -1 ]
            }
        }
    ]
}
```

http://doc.instantreality.org/tools/x3d_encoding_converter/
defines a coordinate system for its children that is relative to the coordinate systems of its parents and 4.3.6 Standard units and coordinate system for a description of coordinate systems and units.

gNode {
    center 0 0 0 (-∞, ∞)
    children [] [X3DChildNode]
    rotation 0 0 1 0 [-1,1] or (-∞, ∞)
    scale 1 1 1 (-∞, ∞)
    scaleOrientation 0 0 1 0 [-1,1] or (-∞, ∞)
    translation 0 0 0 (-∞, ∞)
}

from the origin of the local coordinate system (0,0,0). The rotation field specifies a rotation of the non-uniform scale of the coordinate system. Scale values may have any value: positive, negative or zero indicates that any child geometry shall not be displayed. The scaleOrientation specifies a scale (to specify scales in arbitrary orientations). The scaleOrientation applies only to the scale islation to the coordinate system.

node, P is transformed into point P’ in its parent’s coordinate system by a series of intermediate ation, where C (center), SR (scaleOrientation), T (translation), R (rotation), and S (scale) are the
Entry points for getting started

Some X3D books:
“X3D: Extensible 3D Graphics for Web Authors”
“The Annotated VRML 97 Reference” (explains concepts)
“Das Einsteigerseminar VRML” (VRML/X3D concepts, German)

X3DOM online documentation and code examples:
http://doc.x3dom.org/ (tutorials and docs)
http://examples.x3dom.org/ (application examples)

More docs and tools:
http://www.instantreality.org/downloads/ (InstantPlayer and aopt)
http://doc.instantreality.org/documentation/getting-started/ (X3D links)
Pushing the limits
DOM holds structure and data
More than 95% are usually unstructured data
Declarative (X)3D in HTML

Problem: Large Datasets

• Real 3D applications lead to huge HTML files
  – DOM holds structure and data
    • More than 95% are usually unstructured data
  – Unpleasant non-interactive user experience
    • Browsers not built to hold Gigabytes of DOM attribute data

• Possible solutions
  – Reference external sub trees (e.g. X3D Inline)
  – Binary XML decompression (e.g. X3DB format)
  – Separate structure from raw data

<table>
<thead>
<tr>
<th>Audio</th>
<th>Video</th>
<th>Images</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP3</td>
<td>H.264</td>
<td>PNG/JPEG</td>
<td>?</td>
</tr>
</tbody>
</table>
Declarative (X)3D in HTML
Problem: Large Datasets

• Separation of structure and data
  – Structure in DOM, attributes in binary containers
• Batching draw calls by merging geometries
  – Subdivision into meshes with (max. $2^{16}$) vertices
• Compact vertex data encoding
  – Improve transmission (over Web and CPU $\rightarrow$ GPU)
  – No decoding, just transfer as-is to GPU memory
• (Client-/server-based approaches)
Data Preparation and Partitioning

Preprocessing steps (in aopt tool):

1. Model optimization
   - Remove duplicate materials
   - Merge meshes with same material into one mesh
     • Flattened graph with only geometry as output
   - \( \rightarrow \) Improve batching of draw calls

2. Large data partitioning
   - Loose KD tree or octree
   - Vertex cache optimization
New Geometry node types

```xml
<binaryGeometry vertexCount='1153083' primType=""TRIANGLES"
    position='19.8 -57. -1.7' size='92.8 159.9 26. 5'
    coord='binGeo/BG0_interleaveBinary.bin#0+24' coordType='Int16'
    normal='binGeo/BG0_interleaveBinary.bin#8+24' normalType='Int16'>
</binaryGeometry>
```

Data transcoding (example with input file „model.ply“)

Without mesh restructuring
aopt -i model.ply -G binGeo/:sac -N model.html

With mesh/graph optimization (cleanup, patching, and binary creation)
aopt -i model.ply -F Scene:"cacheopt(true)" -G binGeo/:sac -N model.html
Vertex Data Encoding

• Compact encoding of vertex attribute data
  • Using 16 bit (pos, texCoord) or 8 (normal, color)
    • Required number of bits for pixel-level accuracy for viewpoint of size $w \times h$, with $N := \max(w, h)$, is: $\lceil \log_2 N \rceil$
  • No expensive CPU-based decoding required

• Optionally normals as spherical coords in pos.w
  • Simplified conversion to Cartesian coords for mobile shaders
  • aopt -i model.x3d -F Scene:”maxtris(20000)” -G binGeo/:sacp -N model.html

• Ensure 32 bit alignment for rendering speed
  • To avoid additional processing on certain GPUs
Vertex Data Encoding

Minimalistic X3DOM Viewer

X3DOM output created with instantReality tool.
Loading objects from external resources

• Please note: web security mechanisms don’t allow file system access
  – Start from local web server (e.g., XAMPP with Apache)
  – ...or just disable security as shown in screenshots
3D application development
Tool chain and asset pipeline

- X3D/VRML open ISO standard
  - Supported by most DCC tools
  - Strong support for X3D/VRML also in mesh processing tools such as MeshLab
- InstantReality includes online / offline converter for X3D/HTML
  - instantreality.org/downloads
  - aopt tutorial on x3dom.org
- X3DOM portal includes growing number of tutorials
  - Blender, 3ds Max, Maya
- Standard JS libs available for dynamics and UI (e.g. jQuery)
• Data transcoding (example with input file `model.ply`)
  
  With mesh/graph optimization; i.e. cleanup, patching, binary creation:
  
  ```
  aopt -i model.ply -F Scene:"cacheopt(true)" -G binGeo/:sa -N model.html
  ```

• Process complicated and result is pure X3D file or very simple web page
Common Implementation Framework
Frontend

http://pipeline.v-must.net
The CIF Workflow Pipeline

Geometry + Scene-graph

Upload Model → Optimization → Transcoding → Application Generation

Python

Output

Preview

Distributable Package

http://pipeline.v-must.net
CIF as Web Service

• Web service framework for generating fully interactive 3D Web applications
  – Simple process from raw 3D-scanned model to Web application that runs on most modern devices
  – Rapidly develop online virtual museum apps or convert even entire collections

• CIF designed as n-tier web service
  – Communication through stateless HTTP protocol
  – No special hardware required: off the shelf machines using Linux

• Powerful template rendering and asset pipeline
  – HTML application authoring detached from conversion / optimization process
  – Sophisticated default templates based on X3DOM: Radiance Scaling, metadata integration, etc.
Open Source Implementation
Available on GitHub

https://github.com/x3dom/pipeline
Radiance Scaling [Vergne et al. 2010]
Optimized scaling function (more robust and better performance on the Web):
\[
\sigma_{\alpha,\gamma}(\kappa, \delta) = \delta \cdot \gamma \cdot \text{sign}(\kappa) \cdot \sqrt{|\kappa|}
\]
Application Templates
Examples
Appearance and Lighting
Light sources in X3DOM
...are part of the <scene>

Directional light

Point light

Spot light

<directionalLight direction='0 0 -1' intensity='1'></directionalLight>
<pointLight location='0 0 0' intensity='1'></pointLight>
<spotLight direction='0 0 -1' location='0 0 0' intensity='1' beamWidth='1.57' cutOffAngle='1.57'></spotLight>
The X3D lighting model (diffuse and specular reflection)

\[ I_{\text{diff}} = \max(0, \vec{N} \cdot \vec{L}) = \max(0, \cos \alpha) \]
\[ I_{\text{spec}} = (\vec{N} \cdot \vec{H})^s = \cos^s \beta \]
\[ \vec{H} = (\vec{L} + \vec{V}) / |\vec{L} + \vec{V}| \]

Final color \( I := \text{ambient material} + \text{diffuse material} \ast (\vec{N} \cdot \vec{L}) + \text{specular material} \ast (\vec{N} \cdot \vec{H})^s \)

For more light sources:
\[ I_{\text{ges}} = a_{\text{glob}} \otimes m_{\text{amb}} + m_{\text{em}} + \sum_k c_{\text{spot}}^k (I_{\text{amb}}^k + d^k (I_{\text{diff}}^k + I_{\text{spec}}^k)) \]
Other rendering effects

Note: like <material> only as child node of <appearance> possible!
Appearance example: a textured box

```xml
<x3d width="500px" height="400px">
  <scene>
    <shape>
      <appearance>
        <imageTexture url="logo.png">
          </imageTexture>
      </appearance>
    <box></box>
    </shape>
  </scene>
</x3d>
```

Interesting alternative – using video as texture:

```xml
<movieTexture url=""foo.mp4","foo.ogv""> </movieTexture>
```
Shadow Mapping

Basics
• Create shadow maps
• Shade scene
Parameterizing shadows

- Shading as post-process, thus transparent to user-defined shaders
  - Drawing of shadows independent from scene rendering
- Additional fields in all light nodes
  - shadowIntensity (default 0), shadowMapSize (default 1024), shadowFilterSize (default 0), shadowOffset (default 0), and, very rarely needed: zNear (default -1), zFar (default -1)
  - Fields in Directional- and SpotLight
    shadowCascades (default 1), shadowSplitOffset (default 0.1), shadowSplitFactor (0 equidistant splitting, 1 logarithmic – default)

HTML: <DirectionalLight ... shadowCascades='1'/>

HTML: <DirectionalLight ... shadowCascades='6'/>
Expressive Rendering

- Radiance Scaling (Vergne et al., 2010) enhances geometric features like tiny engravings on 3D surface
- Method first extracts curvature information from geometry and then adjusts incoming lighting
- Implemented with ComposedShader and RenderedTexture
Expressive Rendering
Interaction and API
HTML Events:
User interaction via DOM Events

```html
<shape onclick="document.getElementById('mat').setAttribute('diffuseColor', 'green');" >
  <appearance>
    <material id="mat" diffuseColor="red">
      </material>
    </appearance>
  <box></box>
</shape>
...
HTML Events:
User interaction via DOM Events - V2

```html
<shape id="box">
  <appearance>
    <material id="mat" diffuseColor="red"></material>
  </appearance>
  <box></box>
</shape>

<script type="text/javascript">
  document.onload = function() {
    document.getElementById('box').addEventListener('click', function() {
      document.getElementById('mat').setAttribute('diffuseColor', 'olive');
    }, false);
  }
</script>
```
Efficient Picking and Interaction

- JS-based SG traversal slow
  - Using „picking buffer“ instead
  - WebGL limitations
    - 8 bit RGBA texture
    - No MRT in FBO
- Single-pass render-buffer
  - Get position, normal and object ID within one step / pass
Efficient Picking and Interaction

• Single-pass render-buffer approach
  • Distance $d$ to camera in RG channel (16 bit)
  • Object- or per-vertex id in BA channel (16 bit)
    • Allows identifying up to 65 535 different objects
• Normal vectors derived using $2 \times 2$ window
  • Important for navigation and interaction
  • Read back 4 values instead of only one
    • Calculate cross product
HTML Events: 3DPickEvent extends DOM MouseEvent

interface 3DPickEvent : MouseEvent {
    readonly attribute float worldX;
    readonly attribute float worldY;
    readonly attribute float worldZ;
    readonly attribute float normalX;
    readonly attribute float normalY;
    readonly attribute float normalZ;

    ...
}

<group onmousemove="updateTrafo(event); " ...
<transform id="trafo"><shape isPickable="false"> ...
</shape></transform>

function updateTrafo(event) {
    var t = document.getElementById('trafo');
    var norm = new x3dom.fields.SFVec3f(event.normalX, event.normalY, event.normalZ);
    var qDir = x3dom.fields.Quaternion.rotateFromTo(new x3dom.fields.SFVec3f(0, 1, 0), norm);
    var rot = qDir.toAxisAngle();
    t.setAttribute('rotation', rot[0].x+' '+rot[0].y+' '+rot[0].z+' '+rot[1]);
    t.setAttribute('translation', event.worldX+' '+event.worldY+' '+event.worldZ);
}
### SFVec3f
- `copy(v)`
- `parse(str)`
- `setValues(that)`
- `at(i)`
- `add(that)`
- `addScaled(that,s)`
- `subtract(that)`
- `negate()`
- `dot(that)`
- `cross(that)`
- `reflect(n)`
- `length()`
- `normalize()`
- `multComponents(that)`
- `multiply(n)`
- `divide(n)`
- `equals(that,eps)`
- `toGL()`
- `toString()`
- `setValueByStr(str)`

### Quaternion
- `parseAxisAngle(str)`
- `axisAngle(axis,a)`
- `rotateFromTo(fromVec,toVec)`
- `multiply(that)`
- `toMatrix()`
- `toAxisAngle()`
- `angle()`
- `setValue(matrix)`
- `dot(that)`
- `add(that)`
- `subtract(that)`
- `setValues(that)`
- `equals(that,eps)`
- `multScalar(s)`
- `normalize(that)`
- `negate()`
- `inverse()`
- `slerp(that,t)`
- `toGL()`
- `toString()`
- `setValueByStr(str)`

### SFCColor
- `parse(str)`
- `colorParse(color)`
- `setValuesParse(color)`
- `equals(that,eps)`
- `add(that)`
- `subtract(that)`
- `multiply(n)`
- `toGL()`
- `toString()`
- `setValueByStr(str)`

### SFCColorRGBA
- `parse(str)`
- `setValues(color)`
- `equals(that,eps)`
- `toGL()`
- `toString()`
- `setValueByStr(str)`
**X3DOM Math Lib**

**SFMatrix4f**
- copy(that)
- identity()
- zeroMatrix()
- translation(vec)
- rotationX(a)
- rotationY(a)
- rotationZ(a)
- scale(vec)
- lookAt(from,at,up)
- parseRotation(str)
- parse(str)
- e0()
- e1()
- e2()
- e3()
- setTranslate(vec)
- setScale(vec)

**Example**

Basic decomposition of x3dom.fields.SFMatrix4f (given as mat)

```javascript
var quat = new x3dom.fields.Quaternion(0, 0, 0, 1);
quat.setValue(mat);
var rotation = quat.toAxisAngle();
var translation = mat.e3();
```

- mult(that)
- multMatrixPnt(vec)
- multMatrixVec(vec)
- multFullMatrixPnt(vec)
- transpose()
- negate()
- multiply(s)
- add(that)
- addScaled(that,s)
- setValues(that)
- setValue(v1,v2,v3,v4)
- toGL()
- at(i,j)
- sqrt()
- normInfinity()
- norm1_3x3()
- normInf_3x3()
- adjointT_3x3()
- equals(that)
- getTransform(translation, rotation, scale, center)
- det3(a1,a2,a3,b1,b2,b3,c1,c2,c3)
- det()
- inverse()
- toString()
- setValueByStr(str)
Navigation
Moving the virtual camera interactively

• Built-in navigation modes
  – Examine, walk, fly, lookat, game, helicopter, none
  – <navigationInfo type="any"></navigationInfo>
  – Abstract behavior dynamically maps to various user inputs: mouse, keys, multi-touch

• Application-specific navigation
  – Use navigation type ‘none’ to disable built-in camera navigation
  – Move camera by updating position and orientation of <viewpoint> node
Multi-touch Navigation

Swipe                 Rotate                 Zoom

(→ Only for navigation mode „examine“)

Please note: special multi-touch events are not yet implemented
Animations
CSS 3D Transforms & CSS Animation

- CSS 3D Transforms Module Level 3; W3C Draft
  - To transform and update `<transform>` nodes (WebKit version)

```css
#trans {
  -webkit-animation: spin 8s infinite linear;
}
@-webkit-keyframes spin {
  from { -webkit-transform: rotateY(0); }
  to { -webkit-transform: rotateY(-360deg); }
}
</style>

...<transform id="trans">
<transform style="-webkit-transform: rotateY(45deg);">
```
Animations
X3D TimeSensor andInterpolator nodes

<scene>
  <transform id="trafo" rotation="0 1 0 0">
    <shape>
      <appearance>
        <material diffuseColor="red">
        </material>
      </appearance>
      <box></box>
    </shape>
  </transform>
  <timeSensor id="ts" loop="true" cycleInterval="2">
  </timeSensor>
  <orientationInterpolator id="oi" key="0.0 0.5 1.0"
    keyValue="0 1 0 0, 0 1 0 3.14, 0 1 0 6.28">
  </orientationInterpolator>
  <ROUTE fromNode='ts' fromField='fraction_changed'
    toNode='oi' toField='set_fraction'></ROUTE>
  <ROUTE fromNode='oi' fromField='value_changed'
    toNode='trafo' toField='set_rotation'></ROUTE>
</scene>

- <timeSensor> "ts" triggers via the first ROUTE <orientationInterpolator> "oi", which provides values for rotation around y-axis (0,1,0)
- Resulting value is then ROUTE 'd to field 'rotation' of <transform> node "trafo", which results in an animation
- Alternatively: X3D Follower nodes for computing transitions at run-time
X3D event flow graph

- Connects fields with routes
  - ROUTE statements connect output of one node field as an input to another node
  - Transports events/values between slots/fields
  - 0-N relation edges/routes per slot/field
    - Fan-in and fan-out
- First introduced in VRML
  - Allows describing behavior without programming
  - Common model in many VR systems (e.g., Avango, Virtools, InstantReality)
X3D event flow graph – example

```xml
<X3D>
  <Scene>
    <Transform translation="-1 0 0">
      <Shape>
        <Appearance>
          <Material diffuseColor="1 0 0" />
        </Appearance>
        <Box DEF="box" size="1 1 1" />
      </Shape>
      <TouchSensor DEF="TS" />
    </Transform>
    <Transform DEF="T" translation="0 0 0">
      <Shape>
        <Appearance>
          <Material diffuseColor="0 0 1" />
        </Appearance>
        <Box USE="box" />
      </Shape>
    </Transform>
    <TimeSensor DEF="TIME" cycleInterval="1.0" loop="false" />
    <PositionInterpolator DEF="PI" key="0.0 0.5 1.0" keyValue="0 0 0, 3 0 0, 0 0 0" />
    <ROUTE fromNode="TS" fromField="touchTime" toNode="TIME" toField="startTime" />
    <ROUTE fromNode="TIME" fromField="fraction_changed" toNode="PI" toField="set_fraction" />
    <ROUTE fromNode="PI" fromField="value_changed" toNode="T" toField="translation" />
  </Scene>
</X3D>
```
HTML Profile (X3D subset)
Basic scene-graph nodes as DOM elements

- Group, transform, lod, switch, billboard
- Inline (loads sub-scenes asynchronously)
- Single and multi-index geometry types
- PointLight, SpotLight, DirectionalLight
- Appearance with flexible shader selection technique (e.g., CommonSurfaceShader)
  - Multipass-Rendering via RenderedTexture
- Follower and interpolator for animation
- Background (skybox background)
- Viewpoint (standard perspective camera)
- Standard navigation (walk, fly, examine...)
- Support for HTML5 media elements
  - Texture: `<img>`, `<video>`, `<canvas>`
  - Sound: `<audio>`
Runtime API

- Provides programmatic live access to system
  - Runtime proxy object attached to each `<x3d>` element
  - Simple example: `x3dElem.runtime.showAll();`
    - Zooms such that all objects are fully visible

- Other useful methods
  - `enter-/exitFrame()`: Called before/after current frame
  - `getActiveBindable(typeName)`: Returns currently active Bindable element of given type (e.g. Viewpoint)
  - `getViewingRay(x, y)`: Returns Ray object through (x, y)
  - `mousePosition(evt)`: Returns position on 2d canvas layer [x, y] for mouse event `evt`
  - `calcPagePos(wx, wy, wz)`: Returns 2d page position for given point [wx, wy, wz] in world coordinates
  - `getSceneBBox()`: Gets min/max of scene bounding box
  - `resetView()`: Resets to initial camera view
Example 1: Interactive Car Configurator

Interaction via standard Web technologies (e.g., JavaScript events etc.)

```html
<img src="felge1_64.jpg" onclick="..." style="..."/>
```

Click on `<img>` element...

```javascript
document.getElementById('body_color').setAttribute("diffuseColor", '#000066');
```

...causes attribute change of `<texture>` url (i.e., other wheel rims appear)
Example 2: Painting Textures of 3D Objects

- `<x3d>` element
  - Part of DOM/HTML document like every other HTML element
  - (JavaScript implementation based on new WebGL API of HTML5 `<canvas>` element)

- HTML5 `<canvas>` element
  - Painted image used as texture on 3D object

- jQuery UI (for User Interface)
  - jQuery JavaScript library: http://jqueryui.com/
Application 1: Web-based 3D modeling

Drag & Drop of primitives

Interactive 3D transformations via mouse and UI

Parameter modifications
Application 2: 3D-Internet Design Review

Interacting with large data

Maximum Visualization

- Whole car incl. modules and parts
- Whole car incl. modules
- Modules with parts
- Only parts
Conclusions (X3DOM)

• Works in most modern Web Browsers without plugins
  – Look & feel comparable to native apps
  – Device independent
• One solution for all kinds of devices
  – Saves development time and money
  – Allows Web developers to directly start coding
• Technology based upon open standards
  – Expectable that app remains accessible in future
• Interactive merging of 3D models with traditional media contents
  – Including smooth 3D interactions
• Short loading time
  – If binary geometry containers are used
Conclusions (Declarative 3D)

- **Development costs**: Web developer vs. graphics expert
- **Adaptability**: Declarative material abstraction allows shading adoption per client hardware (e.g. GLSL, ray-tracing...)
- **Efficiency**: UI events, culling, rendering can be implemented in native code, thus utilizes battery resources efficiently
- **Accessibility**: High level navigation and interaction styles allow very late adaptations for specific use cases
- **Metadata**: Allow indexing and searching content
- **Mash-ups**: Asset reuse in new context
- **Security**: With declarative approach, no plugins or even direct GPU calls necessary (theoretically...)

→ Powerful Abstraction for 3D Web Applications
Special thanks to Nils Michaelis for helping with demo and slides

Part 2: System Development Implementing Web Apps – Example from CH Domain
Visualization of 3D-scanned CH Artifacts
Motivation

• 3D interactive content important building block for Virtual Heritage applications
• Exploration, presentation, analysis, documentation, and reconstruction can be assisted
• Mobility becomes more important and is essential on site
• With various devices synchronization is an issue

R. Scopigno et al., 2011. 3D Models for Cultural Heritage: Beyond Plain Visualization
Requirements

• 3D presentation in Web Browser with integrated use of Web technology
  – No need to install special programs or plugins
• One single app for desktop machines and lightweight mobile devices
  – Performance and interactivity
• Integration of 3D content into DOM tree for easing scene manipulation
  – X3DOM framework
Excursus: jQuery
Mobile JavaScript Frameworks

• Mobile devices use different platforms
• Many problems such as
  – Native apps need to be developed for all systems (portability)
  – Both, desktop applications and Web sites, need to be developed
• Possible solution
  – Develop Web sites such that they run in mobile Browsers with the look and feel of native apps
    • No need to install app
    • “Normal” Web developer can create Web apps
  – JS UI libraries can help to support look and feel of native apps
    • Using library often more efficient

http://www.jqtouch.com/preview/demos/main/#ui
http://jquerymobile.com/demos/1.1.0/docs/lists/lists-nested.html
Mobile JavaScript Frameworks

- jQuery / jQuery Mobile
  - Very powerful
  - Easy to use and easy theming
  - Very well documented with many examples
  - Recommended

- jQTouch
  - Lightweight and performant
  - Bad documentation
  - Nice look and feel
  - Optimized for WebKit
    - Bad support for other Browsers

- Other Frameworks available
  - All aim at providing look & feel of native apps
jQuery

- Include core library
  `<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>`
- jQuery commands start with "jQuery" or "$
- Commands can be chained
  - E.g. `$variable.find("something").each(...)`
- Provides huge number of useful functions
- Wait till the DOM (page) is loaded to execute code
  ```javascript
  $(document).ready(function() {
      // do stuff when DOM is ready
  });
  ```
- Selectors can be used as in CSS
  ```javascript
  $('a'), $('#id'), $('\class')
  ```
- Search descendants of already selected elements
  ```javascript
  $('#list').find('li') - equal to $('\list li')
  ```
jQuery

- Elements can be created with
  ```javascript
  var p = jQuery('<p>' + contentVariable + '</p>);
  group.append(p);
  ```
- Create an element and write attributes
  ```javascript
  var elem = jQuery('<div/>').attr({
    "data-role" : "collapsible",
    "data-collapsed" : "true",
    "data-mini" : "true",
    "data-iconpos" : "right",
    "data-theme" : "b",
    "class" : " metaDataEntry"
  });
  ```
- `attr()` or `prop()` functions can be used to read or write attributes
jQuery

• Events ([http://api.jquery.com/category/events/](http://api.jquery.com/category/events/))
  - `<a href="">Link</a>`
  - `$("a").click(function() {
      alert("Hello world!");
    });`
  - Or: `var callback = function() {
      alert("Hello world!");
    }
    $("a").click(callback);`

• Nice Effects ([http://api.jquery.com/category/effects/](http://api.jquery.com/category/effects/))
  - `$('#someElem').fadeIn();`
  - `$('#someElem').fadeOut();`
  - `$('#someElem').slideIn();`
  - `...`
jQuery UI

- [http://jqueryui.com](http://jqueryui.com) (tutorials: [http://learn.jquery.com](http://learn.jquery.com))
- Useful for creation of GUI's
  - Easy to build
  - Draggable/Resizable Elements
  - Slider, Tabs
  - Autocomplete fields
  -...

![jQuery UI Examples](image.png)
jQuery Mobile

• Builds upon jQuery and jQuery UI
• Include jQuery and jQuery Mobile libs

```html
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.css" />
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.js"></script>
```

http://jquerymobile.com/
jQuery Mobile

• Loads entire Website, not just currently viewed page
  – No loading lags when switching page because of preloading

• Required page setup
  – jQuery Mobile is all about convention over configuration

```html
<body>
  <div data-role="page">
    <div data-role="header">
      <h1>My Title</h1>
    </div><!-- /header-->
    <div data-role="content">
      <p>Hello world</p>
    </div><!-- /content-->
  </div><!-- /page-->
</body>
```
jQuery Mobile

- HTML code has certain commands (or attributes) to tell the framework how to present a section
  - Mostly start with “data-”
  - W3C recommendation for unknown attributes

- Examples

```html
<ul data-role="listview"
    data-inset="true" data-filter="true">
    <li><a href="#">Acura</a></li>
    <li><a href="#">Audi</a></li>
    <li><a href="#">BMW</a></li>
    <li><a href="#">Cadillac</a></li>
    <li><a href="#">Ferrari</a></li>
</ul>

<a href="#" data-role="button"
    data-icon="star">Star button</a>
```
jQuery Mobile

- In dynamic apps HTML is created from XML, JSON or DB
- If HTML is created after DOM is loaded, jQuery Mobile will not convert it automatically
- To force conversion, functions are provided that must be executed after HTML is written into the DOM
- Example: select added element and then convert view
  
  ```javascript
  jQuery('div[data-role=collapsible]').collapsible();
  ```
Ready for Implementation

• We now have...
  – JS framework for look and feel of native apps
  – Web Browser that supports X3DOM

• All further development is just the same for mobile devices and webpages
Concept and Implementation
Hands on

• Download project and open it in editor
  – http://www.x3dom.org/temp/ExampleX3domWebApp.zip
  – Load model (data/Herkules/model-opt.html) and check if it is displayed in Browser

• Getting started with jQuery Mobile
  – Add heading line beneath <div id="rightpane">
    and give it some name
Metadata Integration

<display_date />
</temporal_coverage>
<creation_date Recommended="September 2011" />
</Collection>

<Subject>
- <Record Information>
  <ID Mandatory="" />
  <source />
  <creator Nicola Amico</creator>
  <creation_date Recommended="26/03/2012" />
  <country Recommended="Cyprus" />
  <language Recommended="English (EN)" />
  <rights>STARC - The Cyprus Institute, Department of Antiquities of Cyprus, Archaeological Museum of Nicosia</rights>
</Record Information>

- <Appellation>
  <Name Mandatory="Cruciform figurre" />
  <ID Mandatory="1963-XI-22-9" />
</Appellation>

- <description Mandatory="This anointing flask is made of molded blue glass and depicts a human face. Mold-blown glass vessels appear in the first half of the 1st century A.D. The molds were probably made of clay into which the vessel's details were carved." />

- <temporal_coverage Recommended="Roman period (50 B.C. - 150 A.D.)">
  <start_Date Recommended="" />
  <end_Date Recommended="" />
</temporal_coverage>

- <Type Recommended="Archaeological Object" />
- <Measurement>
  - <perimeter>
    <type />
    <value />
    <unit />
</perimeter>
Data Container

• All data corresponding supplied via
  – JSON
  – XML
  – Database like MySQL or CouchDB

• Data can hold
  – Traditional content such as images and text that shall be displayed for metadata
  – 3D information (e.g. for annotations or camera positions)
  – Application logic
  – …

```json
{
  "title": "xxx",
  "model": "xxx",
  "URL": "folder/name.x3d",
  …
  "meta": {
    "ID": "test_1",
    "Name": "xxx",
    "description": "xxx",
    …
  },
  "annotation": [
    {
      "pos": "48.0 55.5 6.5",
      "text": "Looks like snake",
      "link": "wikipedia.org"
    },
    …
  ]
}
```
Metadata Presentation

- Metadata displayed in lists
- Filtered by typing in search field
  - Hiding wrong results via jQuery
- Since X3DOM integrates model into HTML, it is possible to merge 3D content with textual and/or other information
- By default we just display headings of each metadata
  - By tapping on them they expand
  - Various display modes (full 3D area or all infos extended)
Loading Metadata

- Load Metadata (data/metaData.xml) in your HTML document
- Use jQuery get() function (http://api.jquery.com/jQuery.get/) with parameter “XML” to asynchronously load the file
- Pass same callback function to it and check if file is loaded correctly
  - E.g. with console.log($(xmlData).text());
- Create collapsible list with this data and append it to the prepared div container with the id “metaDataList”
- Helpful functions are
  - find("entry")
  - each(function () {...});
  - jQuery('#metaDataList').append(element);
  - jQuery('<div/>').attr({...});
  - jQuery('<div/>').attr({...});
  - jQuery('div[data-role=collapsible]').collapsible();
    - To be called at the very end!
Annotations

• If many people work together on a model, annotations are great for communication
  – Pass information from one person to another
  – Can be attached to certain position on 3D model
  – Can hold all kinds of information such as author, creation date, text, hyperlinks, images, etc.

• For loading, various data container formats (JSON, XML, ...) are possible

• For persistent storage, it is necessary to use a database (e.g. CouchDB)
  – JavaScript doesn’t allow writing to disc
Annotations

• Look for following X3D element
  `<Group id="annotationMarker" render="true"></Group>`

• Use container `<div class="popup..."></div>` as popup that appears when marker is clicked

• Helpful functions are e.g.:
  − `fadeIn() / fadeOut()`
    Or similar effects
  − `click(function(){})`
    In the JavaScript file
  − ...
Create Annotations

• First user has to activate the annotation creation mode by tapping button

• When tapping on object, position can be saved via:

```javascript
function createAnnotation(event) {
    newAnnot.posX = event.worldX;
    newAnnot.posY = event.worldY;
    newAnnot.posZ = event.worldZ;
    ...
}
```

(don’t forget to register event listener)

• To permanently save data in the form use DB
  – Marker can be generated by first creating an empty `<Group>` with an ID
  – Then attach geometry to it, set saved position and register event to show an annotation popup
Loading 3D Models

- To browse through different objects, list with hyperlinks can be used
  - Each `<a>` tag can contain ID that is equal to the name of the folder containing the object data
  - Alternatively, the ID can be the key for a dictionary that contains the paths to the objects
- `<Inline>` node is used to load an X3D model, when link was clicked
  - Dynamically switch URL and load object with JS
    
    ```javascript
    document.getElementById("inlineID").setAttribute("url", "path/objectID/model.x3d");
    ```
Thank You!

http://www.x3dom.org/

Soon to come: X3DOM 2.0