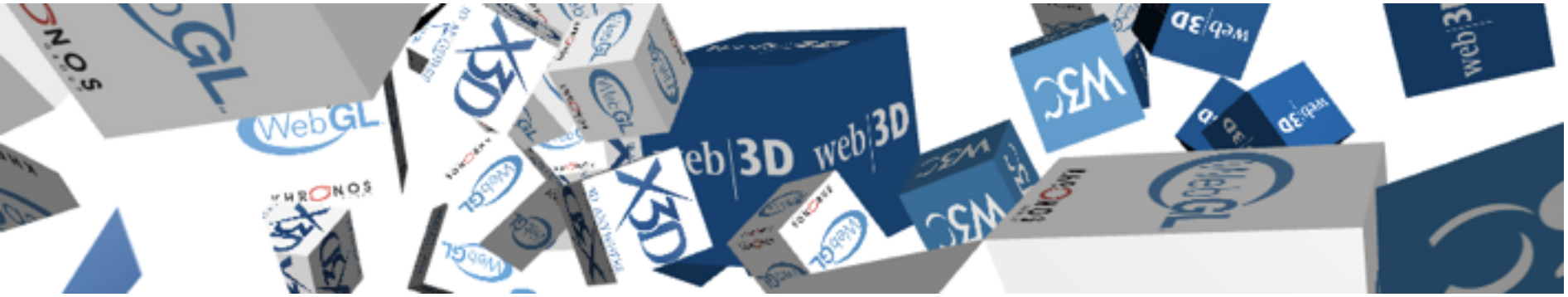
X3DOM

Getting declarative (X)3D into HTML



WebGL BOF, Siggraph 2010

Johannes Behr & Yvonne Jung

Virtual and Augmented Reality Group,
Fraunhofer IGD, Darmstadt, Germany
johannes.behr@igd.fraunhofer.de

Fraunhofer IGD

TECHNISCHE UNIVERSITÄT DARMSTADT

TU Graz

Qualitätsmanagement zertifiziert nach DIN ISO 9001:2000

Motivation

“Future of web3D” Panel 2008 (Web3D symposium)

Vladimir Vukicevic presented **Canvas3D/WebGL**

Pro:

User-agent service => **Plugin-free** approach

OpenGL(-ES) proved itself as **excellent Graphics API**

Con:

Efficiency: Spend too many (battery) resources to manage your scene?

Concepts: HTML Developer has to deal with GLSL and 4x4 matrices.

Metadata: Index and search “content” on WebGL-Apps?

HTML5 Specification:

*12.2 **Declarative** 3D scenes*

*Embedding 3D imagery into XHTML documents is the domain of X3D, or technologies based on **X3D** that are namespace aware.*

Idea: Declarative (X)3D in HTML

Embed a live scenegraph in the DOM

```
<!DOCTYPE html >
<html >
    <body>
        <h1>Hello X3DOM World</h1>
            <x3d xmlns='…' profile='HTML' >
                    <scene>
                            <shape>
                                    <box></box>
                            </shape>
                    </scene>
            </x3d>
    </body>
</html>
```

HTML/DOM Profile

Reduce X3D to 3D visualization component for HTML5

General Goal:

⇒ Utilizes **HTML/JS/CSS** for **scripting** and **interaction**

⇒ **Reduced complexity** and **implementation effort**

⇒ Reduces X3DOM to visualization component for 3D like SVG for 2D

⇒2 Profiles: HTML and HTML-Tiny/WebSG

"**HTML**"-Profile (Extends "Interchange" Profile): **~ 80 nodes**

Full **runtime** with anim., navigation and **asynchronous data** fetching

No X3D-Script, Proto, High-Level Sensor-nodes

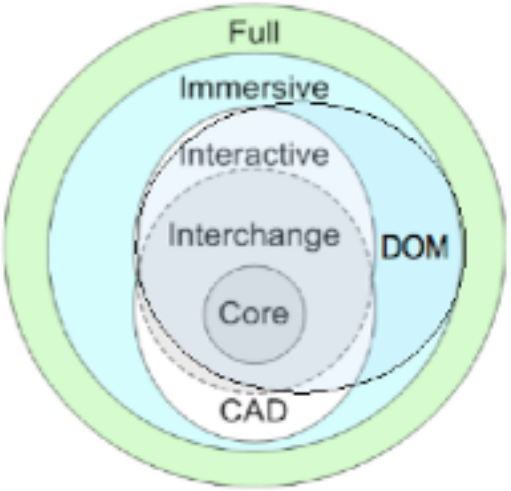
Declarative and explicit shader material

"**HTML-Tiny**" or "**WebSG**"-Profile: **~ 15 nodes**

No Runtime at all: Just **redraw on changes**

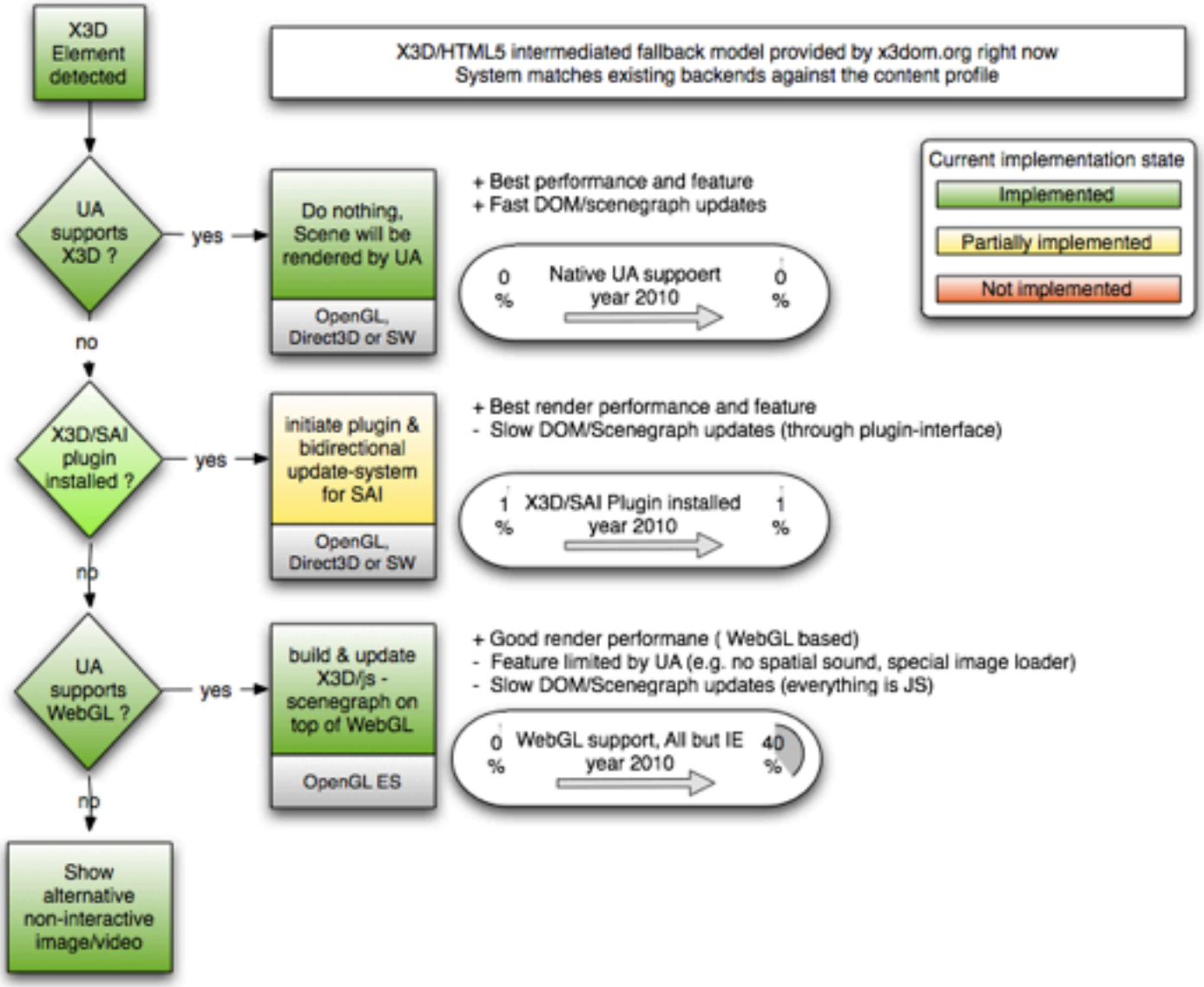
Generic **<mesh>** node without vertex semantics

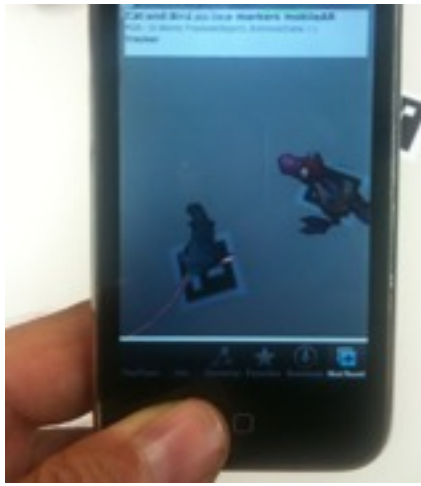
Only explicit shader material



Implementation

JS-Layer: Supports native, X3D/SAI-Plugin or WebGL





x3dom.org/x3dom/release/x3dom.js

JavaScript-based implementation

```
<!DOCTYPE html >
<html >
    <head>
        <link rel="stylesheet" type="text/css" href="x3dom.css" >
        <script type="text/javascript" src="x3dom.js"></script>
    </head>
    <body>
      <h1>HTML5 Hello World</h1>
        <x3d xmlns="…" profile='…' backend='…' >
            <scene>
                …
            </scene>
        </x3d>
    </body>
</html>
```

DOM Manipulation

Node appending and removal

HTML/X3D code:

```
    …
    <group id='root'></group>
    …
```

HTML-Script to add nodes:

```
trans = document.createElement('Transform');
trans.setAttribute('translation', '1 2 3' );
document.getElementById('root').appendChild(trans);
```

HTML-Script to remove nodes:

```
document.getElementById('root').removeChild(trans);
```

DOM Manipulation

Field updates with setAttribute() or SAI-Field interfaces

HTML/X3D code:

```
…
<material id='mat'></material>
…
<coordinate id='coord' point='5.6 3 87, 8.8 8.4 3.2, …' ></coordinate>
…
```

Generic HTML-Script with **setAttribute()**: also useful for libs like **jQuery**

```
document.getElementByid('mat').setAttribute('diffuseColor','red');
```

HTML-Script using SAI-Field interface: X3D JS-binding for more efficiency

```
var saiField = document.getElementById('coord).getField('point');
saiField[4711].x = 0.815;
```

HTML Events

User Interaction through DOM Events

Supports interaction with standard HTML-Events. Supports **ancient** (Netscape2) and **addEventListener()** interfaces.

```
<x3d xmlns=”…">
<Scene>
  <Shape>
    <Appearance>
      <Material id=‘mat’ diffuseColor=‘red’ />
    </Appearance>
    <Box onclick=“document.getElementById(‘mat’).diffuseColor=‘green’” />
  </Shape>
</Scene>
</x3d>
```

DOM Manipulation

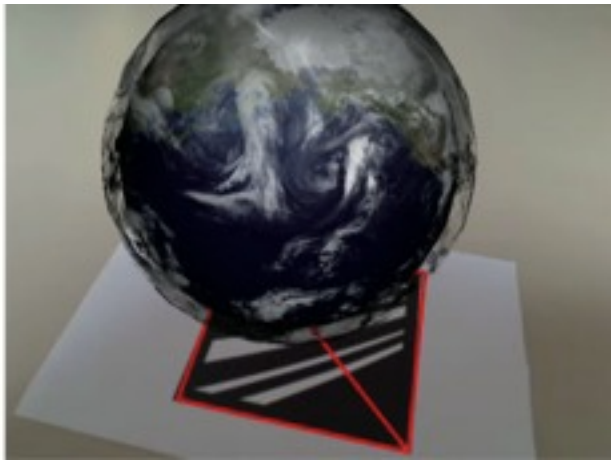
CSS 3D Transforms and CSS Animation

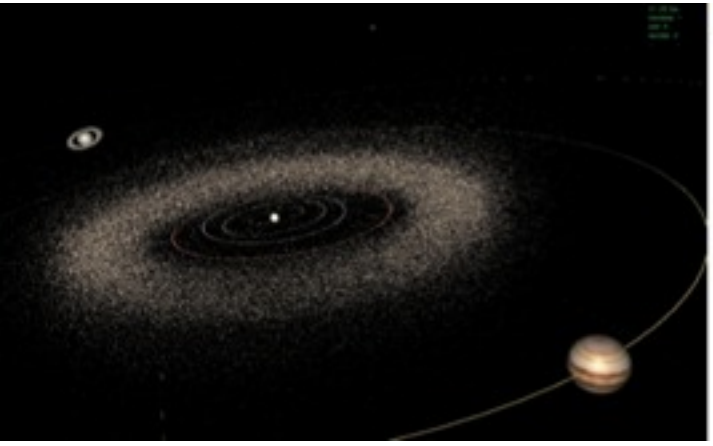
CSS 3D Transforms Module Level 3; W3C Draft

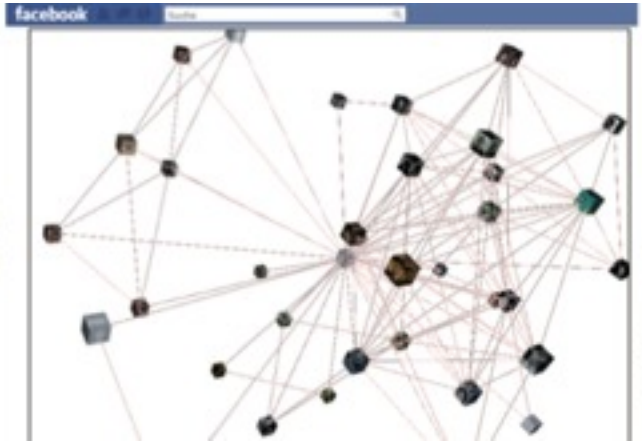
Utilized to transform and update **<transform>** nodes

```
<style type="text/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);">
    …
```

Applications

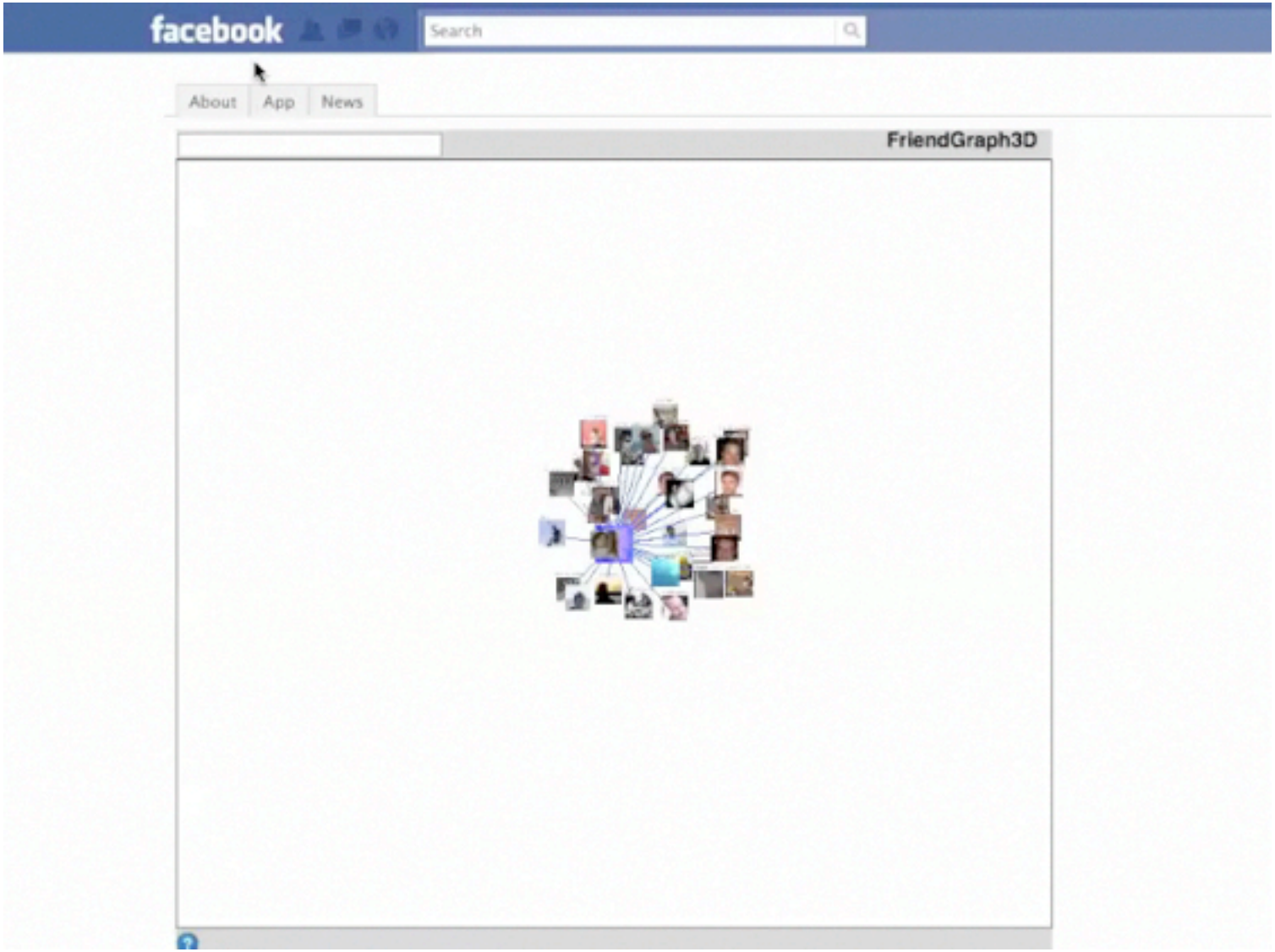






Application show-case

Facebook Friendgraph in 3D



Application show-case

Thematic data on climate change in cities

Conclusions – UserAgent

General:

JS performance is sufficient for a large number of applications.

(Almost) consistent support for **DOM Level 2 mutation** event types.

Missing:

CSS 3D Transform and CSS Animation support in all UA

information about the original image-format in Image object.

Spatial-Audio API.

WebGL:

WebGL performance and features are very impressive today.

Missing:

Support in all released UAs (especially on mobile devices)

depth/floating-point texture (also for FBO)

picking, shadow, multipass

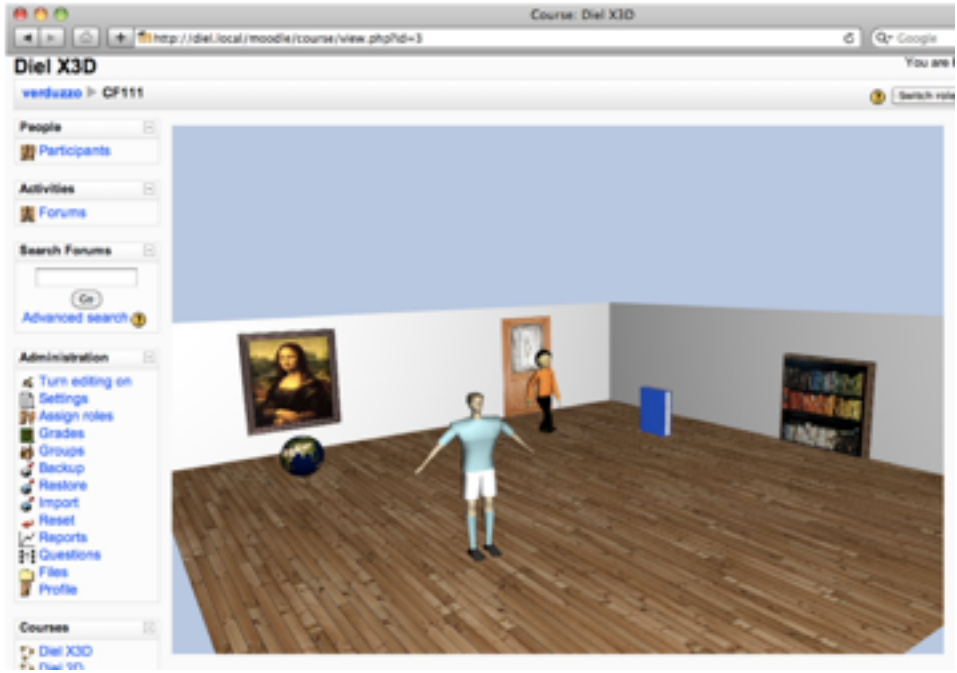
HDR-image loader

Conclusions – X3DOM

X3DOM (pronounced X-Freedom) is an experimental open source framework and runtime to support the **ongoing discussion** in the **Web3D and W3C communities** how an integration of HTML5 and **declarative 3D** content could look like.

Targeted Application Area:

- **Declarative content** design
- Builds on an **HTML5** layer
- Application concepts map well to generic **scenegraph**



Thanks!
Questions?

System:

www.x3dom.org