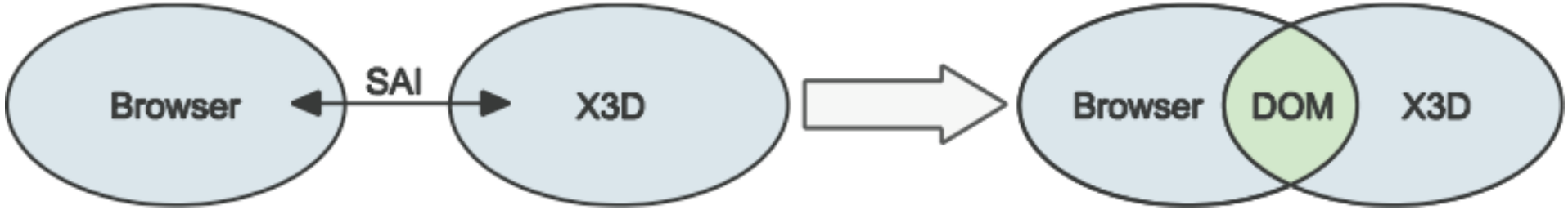
X3DOM

A DOM-based HTML5/ X3D Integration Model



3. Feb. 2010, Clausthal University

Johannes Behr
Peter Eschler, Y. Jung, M. Zöllner,

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

Fraunhofer IGD

TECHNISCHE UNIVERSITÄT DARMSTADT

TU Graz

Qualitätsmanagement zertifiziert nach DIN ISO 9001:2000

Overview

- Introduction and Motivation
- Current State of 3D on the net
- X3DOM Model
  - System Architecture
  - Web Profile
  - DOM integration aspects
  - Implementation
    - Native/Extension
    - SAI/object based
    - O3D based
    - WebGL based
    - Multi-backend Hyprid
- Conclusion and Future Work

Introduction & Motivation

15 years of Web3D technology

Initial hardware and network limitations are gone

- phones render millions of polygons per second
- broadband connection in almost every home

X3D established and solid technology

- Successfully used in various application areas
- But: Very view web application today!

Increasing interest in 3D web technology

- Fat-client based: Second-Life, GoogleEarth, Games (e.g. WOW), …
- Browser based: X3D, O3D, WebGL/Khronos, …

HTML5 group shows interest in 3D technology

- OpenGL (ES) as programming interface
- X3D for declarative content

Current State of 3D on the net

Browser solution – plugin based

General issues:

Installation, security and browser/OS incompatibility

System specific interfaces to access/manipulate the content

Flash (Adobe)

< Version 10: 2D pipeline used for 3D (e.g. Papervision)

>= Version 10: Minimal 3D transformation for 2D elements

Silverlight (Microsoft)

< Version 3: 2D pipeline (there was a 3D pipeline in Avalon/WFC!)

>= Version 3: Minimal 3D transformation for 2D elements

Java, Java3D, JOGL and JavaFX (SUN)

O3D (Google): Javascript based scene-graph API

X3D (ISO, web3d consortium): plugins with SAI interface

MPEG-4 & MPEG-4 Part 11 (ISO, Moving Picture Experts Group)

Current State of 3D on the net

Browser solution – Rendering without plugins

General advantage:

- **No plugin installation issues**
- **Vis./Runtime can be part of the content**

SVG Renderer :

- 3D rendering with 2D pipeline
  - Google chrome experiments / pre3d

CSS Renderer:

- 3D transformation for 2D elements
  - WebKit/Opera extensions

OpenGL based:

- WebGL (plus scene-graph, e.g. C3DL)
- Canvas3D / Opera GL Canvas

Current State of 3D on the net

Native HTML5

- Object/plugin based
  - Model is separated from DOM model
  - Separate data/event model
  - plugin specific scripting interface (e.g. SAI for X3D)
- WebGL
  - Based on Canvas3D (Mozilla)
  - Developed with Khronos group
  - Exposes the OpenGL layer to JavasScript
- 3D scenes (HTML5 specification)
  - *12.2 Declarative 3D scenes*
    Embedding 3D imagery into XHTML documents is the domain of X3D, or technologies bases on X3D that are namespace aware.

X3DOM

A DOM-based HTML5/X3D Integration Model

Allows to embed XML-X3D content inside of every XHTML & HTML page
Uses XML-namespaces to separate X3D content from XHTML content
    => Follows HTML5 declaration
Works with HTML without namespaces but encoding restrictions
X3D content represents a live scene-graph
    Not a single import like the SAI document-import
Provides a single in-place rendering architecture (like e.g. SVG)
Supports updates in both direction
    X3D and DOM events
Presents a declarative interface but no API
    Not a small plugin API but wide content interface
Declaration is independent of runtime implementation style
    Supports native, plugin, or JS+WebGL/O3D implementation
Supports content specific runtime or runtime-extension

DOM Integration Issues

XHTML namespaces: xmlns defines namespace

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
    www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
 <h1>X3D DOM integration and manipulation</h1>
 <x3d:x3d xmlns:x3d="http://www.web3d.org/specifications/
   x3d-3.0.xsd">
   <x3d:Scene>
      <x3d:Shape><x3d:Box x3d:size="4 4 4" /></x3d:Shape>
   </x3d:Scene>
 </x3d:x3d>
</body>
</html>
```

DOM Integration Issues

XHTML namespaces: Default namespaces

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
    www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<!-- All elements within the x3d elements belong to the x3d namespace -->
 <x3d xmlns="http://www.web3d.org/specifications/x3d-3.0.xsd">
   <Scene>
      <Shape><Box size="4 4 4" /></Shape>
   </Scene>
 </x3d>
</body>
</html>
```

DOM Integration Issues

Accessing elements in x3d namespace

```
<x3d xmlns="http://www.web3d.org/specifications/x3d-3.0.xsd">
    <Scene> <Shape><Box size="4 4 4" /></Shape> </Scene>
</x3d>
<script type="text/javascript">
    // The namespace URIs
    var x3d_ns = "http://www.web3d.org/specifications/x3d-3.0.xsd";
    // Get elements using namespaces
    var box = document.getElementsByTagNameNS(x3d_ns, "Box")[0];
    // Edit an attribute of the <Box /> element
    alert(box.getAttributeNS(null, "size"));
    box.setAttributeNS(null, "size", "2 2 2");
    alert(box.getAttributeNS(null, "size"));
</script>
```

DOM Integration Issues

Events from the X3D subsystem

```
<x3d xmlns="http://www.web3d.org/specifications/x3d-3.0.xsd">
      <Scene>
        <Shape><Box size="4 4 4" /></Shape>
            <VisibilitySensor id=”vs" DEF=”vs" size=“4 4 4” />
     </Scene>
</x3d>
<script type="text/javascript">
       var x3d_ns = "http://www.web3d.org/specifications/x3d-3.0.xsd";
        // Get elements using namespaces
        var x3d = document.getElementsByTagNameNS(x3d_ns, "x3d")[0];
        var vs = x3d.getElementsByTagName(“VisiblitySensor")[0];
        vs.addEventListener(”enterTime",
         function() { alert(“There is a Box!"); },  false);
</script>
```

DOM Integration Issues

User Interaction through DOM Events

```
<x3d xmlns="http://www.web3d.org/specifications/x3d-3.0.xsd">
<Scene>
  <Shape>
    <Appearance>
      <Material diffuseColor='1 0 0' DEF='mat' id='mat' />
    </Appearance>
    <Box size="4 4 4"   onclick="document.getElementById
    ('mat').diffuseColor='0 1 0'" />
  </Shape>
</Scene>
</x3d>
```

DOM Integration Issues

HTML5: no ns, lower-case tags and no self-closing tags

```
<!DOCTYPE html >
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
  <h1>X3D DOM integration and manipulation</h1>
  <x3d>
    <scene>
        <shape><box size="4 4 4" ></box></shape>
    </scene>
  </x3d>
</body>
</html>
```

DOM Integration Issues

Open issues

How should we handle HTML5 events and event attributes in general
- e.g. events in X3D and/or node elements ?

Identifying elements
- X3D **DEF** vs. XML **id** and **class**
- **id** and **class** already defined in x3d xsd

Multi-parent x3d-scene-graph relation
- <Group USE='foo' /> replaces the element with a link to 'foo'
- Introduce explicit **< USE />** element ?

X3D elements
- Specific attributes e.g. x, y, width and height, …
- Scene access interface (SAI) on X3D elements

X3D specific JavaScript objects (e.g. to access a specific triangle)

CSS integration: Separation of content and presentation style ?

Content partitioning: **X3D-Inlines** and **X3D-Protos** vs. XML **href** ?
- Alternative: PHP includes: […] include "someCode.php";

X3DOM

Specific Profile: Subset for valid HTML/XHTML tags

Specific X3D-profile for DOM content

- No **Script** nodes
- No **Proto** types
- No **PointingSensor** types
- **Inline** from network component

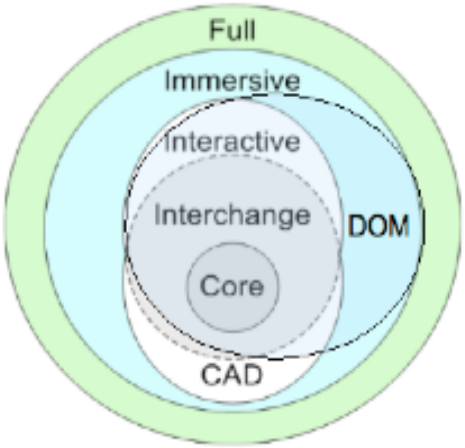
Supports animation for per-frame updates

- **TimeSensor**
- **Interpolator**
- **Follower** (Damper and Chaser)

Reduces complexity

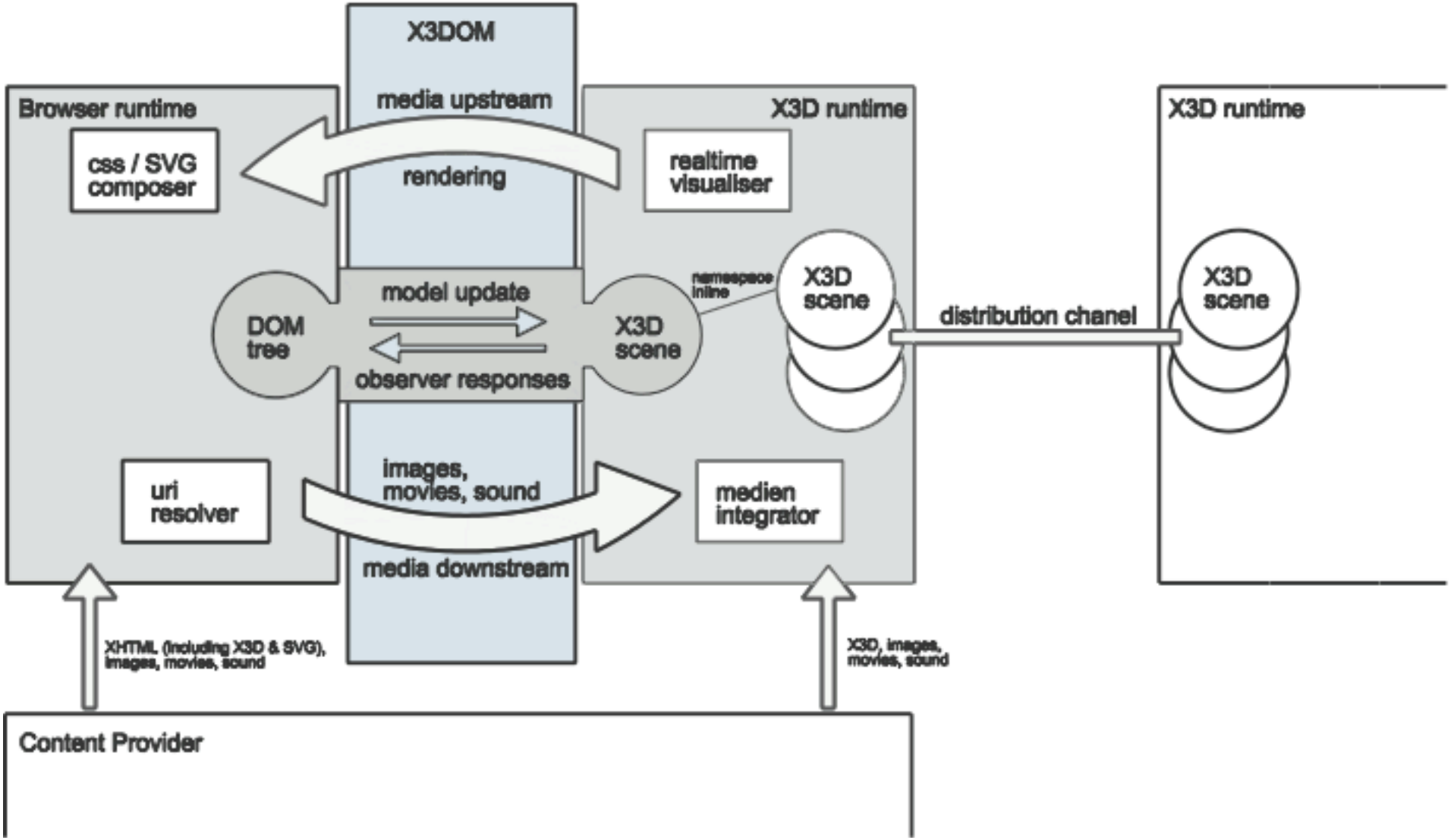
- Eases implementation
- Utilizes xhtml for scripting and distribution

Reduces X3DOM to visualisation component for 3D
like SVG or canvas for 2D



X3DOM

System Architecture / IUA/X3D runtime



Implementation
X3D Runtime for DOM Content

Needs to run the X3D content in-place

Needs to monitor creation/deletion of X3D elements

Needs read/write ACCESS to DOM elements

- Update the X3D graph on DOM changes (e.g. script set)
- Update the DOM element on X3D changes (e.g. animation)

Needs to fetch “Inlined” content

Needs to fetch and download AV-media

- Images, Movie and Sound

Needs to feed the rendered back to browser

Needs to render asynchronously

Implementation

Native/extension based implementation

Needs to monitor creation/deletion of X3D elements

- C/C++ access to DOM elements **browser specific** (e.g. Mozilla ext.)
- ActiveX and NSAPI do not allow to monitor DOM elements

Needs read/write ACCESS to DOM elements

- X3D updates: C++ Observer
- DOM updates: C++ Observer

Fetch “Inlined” content

- Uses browser infrastructure to download DOM document

Needs to fetch and download AV-media

- Uses browser libs to fetch/process Images, Movies and Sound

Pro: Performance, very flexible (e.g. remote rendering)

Con: Browser specific

Implementation

SAI-plugin based implementation

Needs to monitor creation/deletion of X3D elements

- DOM not accessible through plugin-interface
- Needs additional JavaScript wrapper/extension (e.g. jetpack)
  - => creates one plugin/object for every x3d element

Needs read/write ACCESS to DOM elements

- X3D updates: **DOM Mutation Events**
- DOM updates: **SAI callbacks**

Needs to fetch “Inlined” content and AV-media

- Works through X3D runtime

Pro: **Uses standard SAI plugin; high availability**

Con: **Plugin installation issues**

Implementation
O3D based implementation

Needs to monitor creation/deletion of X3D elements

- Needs additional JavaScript wrapper/extension (e.g. jetpack)
  - => creates one O3D context for every x3d element

Needs read/write ACCESS to DOM elements

- X3D/O3D updates: **DOM Mutation Events**
- DOM updates: **javascript callbacks**

Needs to fetch “Inlined” content

- Uses browser infrastructure to download DOM document

Needs to fetch and download AV-media

- Images: O3D-textures; Sound: O3D-Layer; Movie: still open

Pro: **No extra plugin (just O3D), allows content specific runtime**

Con: **Complexity, Needs O3D plugin**

Implementation
WebGL based implementation

Needs to monitor creation/deletion of X3D elements

- Needs additional JavaScript wrapper/extension (e.g. jetpack)
  - => creates one canvas for every x3d element

Needs read/write ACCESS to DOM elements

- X3D updates: **DOM Mutation Events**
- DOM updates: **javascript callbacks**

Needs to fetch “Inlined” content

- Uses browser infrastructure to download DOM document

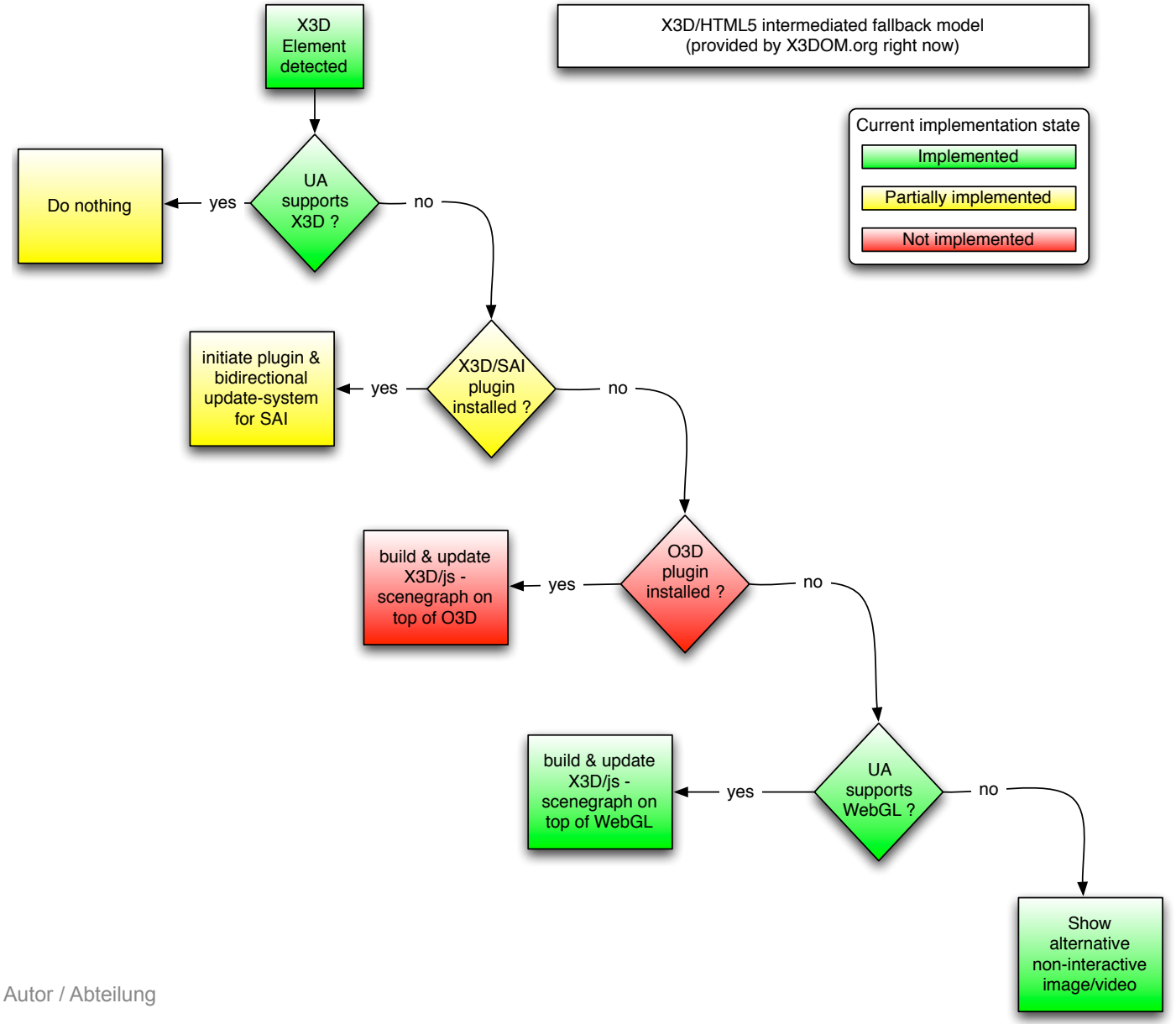
Needs to fetch and download AV-media

- Images: easy, Movie: easy, 3D-Sound: impossible

Pro: **No plugin, allows content specific runtime**

Con: **Performance**

Implementation
Multi-Backend Hybrid: x3dom.org



Implementation
x3dom.org

Open-Source (MIT/GPL)
JavaScript (JS 5-setter for field-updates)
Needs single line per X(HTML)-Page

```
<script type="text/javascript" src="http://
    x3om.org/x3dom/release/x3dom.js" />
```

WebGL-Backend
- Simple - JavaScript – Scenegraph
- Simplified State Model (e.g. field-types)
- One SG-Node-Type per X3D-Node-Type
- N-1 Node relation (DEF/USE)
- OpenGL ES 2.0 Render:
  - No FFP, glsl-shader based
  - Modern shading (e.g. Pixel-lighting)

Conclusion

- DOM-based integration model for X3D and HTML5
- Exploits the current X3D and HTML5 standard
- DOM represents a live X3D scene
  - Read/Write access on scene data
  - Event from/to the X3D runtime
- X3DOM specific X3D-profile
  - Reduces X3D subset to rendering system
  - Eases implementation
- Architecture supports various implementation models
  - Native/Browser, SAI-plugin, O3D or WebGL
- x3dom.org implementation
  - Open-source, JS, WebGL-Backend

Future Work

Standardisation:

- Architecture was presented to the web3d working group
- Accepted as one model to be presented to W3C working group
- Architecture was presented to the W3c/HTML working group (TPAC)
  - Official HTML5 “bug” to integrate X3D
- Developed further through the X3D/HTML5 wiki
  - (**http://www.web3d.org/x3d/wiki/index.php/X3D_and_HTML5**)

Implementation:

- JS-Scenegraph
  - Components and nodes
    - (Follower, Geo-Spatial, Environment-Sensor, CommonShader)
  - Navigation types (e.g fly, walk, look-at)
  - SAI-Field-access
- SAI-Plugin support
- O3D-Backend

Thank you!
Questions?