DECLARATIVE AR AND IMAGE PROCESSING ON THE WEB WITH XFLOW

Felix Klein, Dmitri Rubinstein, Kristian Sons, Farshad Einabadi, Stephan Herhut, Philipp Slusallek
MOTIVATION
THE WEB IS READY FOR AR

- Fast JavaScript
- WebGL, WebCL upcoming
- getUserMedia, WebRTC
- Geolocation, Orientation, Motion

> Problem: Usability
NEW WEB IS POWERFUL AND COMPLICATED

**HTML**

Declarative, High-Level

```
<div>
  <p>
    This is some declarative HTML
  </p>
  <img src="blub.png" alt="with an image too" >
  <a href="someWhereElse.html" >
    ...and a famous link.
  </a>
</div>
```

Pretty straight forward!

**WEBGL**

Imperative, Low-Level

```
var canvas =
  document.getElementById("cvs");
initGL(canvas);
initShaders();
initBuffers();

gl clearColor(0.0, 0.0, 0.0, 1.0);
gl enable(gl.DEPTH_TEST);
drawScene();
```

Pretty difficult...

> Plz make 3D **simpler** for Web developers kthxbye.
DECLARATIVE 3D FOR THE WEB

- W3C Community Group
- Goals
  - Extension to HTML5 for 3D Content
  - Integrated with DOM, CSS etc.
  - Accessible to Web developers
- Evaluation Platform (polyfills)
  - X3DOM - X3D inside the DOM
  - XML3D & Xflow
XML3D

An extension to HTML5 for 3D graphics. Presented at Web3D 2010

```html
<html xmlns="http://www.w3.org/1999/xhtml">
  <!-- ... -->
</html>

<body>
  <!-- ... -->
  <xml3d xmlns="http://www.xml3d.org/2009/xml3d">
    <!-- ... -->
    <group shader="shaders.xml#xml3dTex" >
      <mesh src="cube.json" />
    </group>
  </xml3d>
</body>
</html>
```

- Key features
  - Declarative 3D inside of Web document
  - Minimal extension to HTML
  - Generic core - matching modern GPUs
XFLOW

Declarative Data Processing based on Dataflows. Presented at Web3D 2012

```xml
<data id="wave" compute="(position, normal) = xflow.mywave(pos, norm, str, len, phase)">
  <float name="str">0.01</float>
  <float name="len">40.0</float>
  <float name="phase">0.0</float>
  <data compute="(pos, norm, texcoord, index) = xflow.mygrid(size)">
    <int name="size">50</int>
  </data>
</data>
```

• Key features
  ▪ Declare Dataflows inside the Web document
  ▪ Dataflows execution parallelized / mapped on GPU
  ▪ Generic Design - reusable operators for processing
GOAL

- Extend XML3D & Xflow
  - Support for images processing
  - Support for augmented reality
- Go for a minimal extension
IMAGE PROCESSING

What do we need for image processing with Xflow?
XFLOW FOR REGULAR MESHES

- Use compute attribute
  - access typed arrays (e.g. with float data)
  - output typed arrays
XFLOW WITH IMAGES

- Use texture element to pass images or videos to Xflow operators
  - Xflow operator generates new image
  - Output images can have arbitrary size
  - Default: output image same size as input image

```xml
<!-- Process image -->
data compute="grayImage = xflip.grayscale(image)"
  <texture name="image">
    <img src="someImage.png" />
  </texture>
</data>

<!-- Process videos -->
data compute="grayImage = xflip.grayscale(image)"
  <texture name="image">
    <video src="someVideo.avi" autoplay />
  </texture>
</data>
```
CONNECT PROCESSED IMAGES TO 3D

- Used processed image as surface texture
  - Link processed images to shader
  - Link shader to group
USE PROCESSED IMAGE WITHOUT 3D: XFLIP

```xml
<!-- Process image -->
<xflip id="ipData" compute="output = xflip.grayscale(input)"
     compute="output = xflip.grayscale(input)"
>
    <texture name="input" >
        <img src="someImage.png" />
    </texture>
</xflip>

<!-- Display processed images in HTML -->
<h3>Input:</h3>
<xflipImg src="#ipData" srcName="input" />
```

- Runs independently of XML3D
- Display processed images with xflip element
  - Exactly like regular images
EXAMPLE: IMAGE PROCESSING
AUGMENTED REALITY
THE ACTUAL XFLOW CODE

Based on JSARToolKit

Output

- transforms: world-space transformation for each marker
- visibilities: visibility flag for each marker
- perspective: matrix for intrinsic camera transformation

```html
<data id="arBase" compute="transforms, visibilities, perspective = xflar.detect(arvideo, markers, threshold)">
  <texture name="arvideo">
    <video autoplay="true" src="ar_marker.ogg"></video>
  </texture>
  <int name="markers">22 64</int>
  <int name="threshold" >110</int>
</data>
```
UPDATE INTRINSIC CAMERA PARAMETERS

- Refer xflow element via `perspective` attribute
- Expected to contain `perspective` value of type `float4x4`
UPDATE POSITION OF GEOMETRY

<!--AR Data (as before) -->
<data id="arBase" compute="transforms,visibilities,perspective= ..."> ...
</data>

<!--Extract transform for specific marker -->
<data id="obj1Xfm" compute="transform = xflow.select(index, transforms)">
 <int name="index">0</int>
 <data src="#arBase"/>
</data>

<!--Apply transform to a group node containing geometry-->
<group transform="#obj1Xfm">
 <!-- Geometry placed relative to Marker 0 -->
</group>
UPDATE VISIBILITY OF MESH

<!--AR Data (as before) -->
<data id="arBase" compute="transforms,visibilities,perspective= ...">...
</data>

<!--Extract visibility for specific marker -->
<data id="obj1Vis" compute="visibility =
    xflow.select(index, visibilities)" >
    <int name="index">0</int>
    <data src="#arBase"/>
</data>

<!--Assign visibility flag to shader-->
<shader id="obj1Shader" script="urn:xml3d:shader:phongvs" >
    <float3 name="diffuseColor">1.0 0.4 0.2</float3>
    <float name="ambientIntensity">0.2</float>
    <data src="#obj1Vis"/>
</shader>
FIRST RESULTS FOR AR

- Basic AR application
  - Fiducial Marker Detection
  - Based on JSARToolkit
- Fully Declarative
  - No additional JavaScript
AR IMPROVEMENTS
PROBLEM#1: VIDEO AND 3D NOT SYNCHRONIZED

- We simply overlap a video with an xml3d element
  - Two videos: one in 2D layout, one in 3D scene
  - 3D content delayed by marker algorithm + rendering
WORKAROUND: DRAW VIDEO INTO CANVAS

- Use canvas instead of video
- Draw video frame currently processed by Xflow
- Draw it right before rendering
  - Drawing in canvas works only via JavaScript
  - So far, we have only declarative content

> Need JavaScript integrated with Xflow and Rendering
INTRODUCTION: DATA OBSERVERS

- Designed after DOM MutationObservers
- Efficiently integrated
  - Responses between dataflow computation and rendering
- Great way to efficiently integrate scripts for more flexibility

```javascript
var observer = new XML3DDDataObserver(function (records, observer) {
    var arvideo = records[0].result.getValue("arvideo");
    if (arvideo)
        drawCanvas(canvasElement, arvideo);
});
observer.observe(dataElement, { names: ["arvideo"] });
```
PROBLEM #2: FIXED THRESHOLD

- JSARToolKit expects threshold
  - Used to pre-process image into black/white version
- Fixed threshold only works for certain lighting conditions
  - Add a way to automatically adapt threshold to lighting conditions
• Reuse generic Xflow image processing operators
  ▪ Grayscale image, compute histogram
• Use new xfloar.getOtsuThreshold
  ▪ compute threshold from histogram
RESULT: USING OTSU THRESHOLD COMPUTATION
PARALLELIZATION
PARALLELIZATION WITH RIVERTRAIL

- Parallel JavaScript API by Intel Labs (RiverTrail project)
  - Data-parallelism with regular JavaScript functions
  - Available as Firefox Plug-in
  - Also first native support in Firefox Beta
- Effectively for image processing
  - Process each pixel in parallel
## PERFORMANCE RESULTS

<table>
<thead>
<tr>
<th>Xflow Operator</th>
<th>Time in ms</th>
<th>Speedup Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sequential</td>
<td>Parallel</td>
</tr>
<tr>
<td>Convolution 3x3</td>
<td>236</td>
<td>52</td>
</tr>
<tr>
<td>Convolution 5x5</td>
<td>502</td>
<td>89</td>
</tr>
<tr>
<td>Convolution 7x7</td>
<td>880</td>
<td>141</td>
</tr>
<tr>
<td>Convolution 9x9</td>
<td>1420</td>
<td>218</td>
</tr>
</tbody>
</table>

Image size: 896x512 pixel; Processor: Intel Core i7-2670QM
EXAMPLES
JUMPING TEAPOTS!
I HEARD YOU LIKE WEBCAM STREAMS...
CONTRIBUTION

• Support for Image Processing and AR
  ▪ For Declarative 3D for the Web
• Minimal extension to existing standard
  ▪ 0 new nodes
  ▪ For AR: 2 Xflow operators (+3 otsu)
  ▪ Several new connections for Xflow (reusable)

> Generic Design is awesome!
FUTURE WORK

- Further decompose `xflar.detect` into smaller operators
  - Option to replace individual parts of the detection algorithm
- Implement Post-Processing
  - A better way to combine video with rendered content
- Further optimize Xflow
  - Merge image-processing operators
  - Execution with parallel JavaScript
THANKS TO OUR SUPPORTERS

- Intel Visual Computing Institute
- Verve EU Project
- FI-CONTENT EU Project
  - Part of Future Internet PPP program
THANKS!

QUESTIONS?