THE XML3D ARCHITECTURE

Kristian Sons, Felix Klein, Jan Sutter, Philipp Slusallek

HTML5

XML3D is an extension to HTML5 that allows describing interactive 3D graphics in any Web page.

XML3D

The XML3D scene description [1] is embedded inside HTML using the <xml3d> tag which is fully styleable with CSS.

DEFINITION AREA

The <defs> area is adopted from SVG: It defines content to be referenced and reused later.

HTML ELEMENTS

XML3D reuses existing HTML element and concepts wherever possible.

GENERIC DATA

XML3D has a generic approach to data. Users can define named and typed data entries that can be used e.g. as mesh attributes, material parameters, or with Xflow. The combined data entries defined by the <data> tag can be reused, specialized and composed from multiple sources. The entries defined in <data> elements can be mapped directly to GPU buffers in WebGL.

HIERARCHY

<group> elements define the scene hierarchy in the DOM tree. Materials and transformations can be applied to entire groups hierarchically.

TRANSFORMATIONS

XML3D provides three ways to define transformations: Via (i) CSS, (ii) by referencing a <transform> element, or (iii) through an Xflow graph that defines a data entry named 'transform'.

ASSET INSTANCING

Assets can be externally defined and instanced multiple times within a scene [5]. Each instance can be specialized by overriding values from the instancing document. Assets can reference other assets recursively while maintaining full configurability.

LIGHTS

Lights are first level objects in the scene hierarchy. Light definitions can be reused referencing a light shader.

VIEWS

Views define camera parameters within the scene hierarchy. The currently active view is defined via the 'activeView' attribute of the <xml3d> element.

```
<!doctype html>
<html lang="en">
<head>
  <title>The XML3D Architecture</title>
  <script type="text/javascript" src="../../script/xml3d.js"></script>
</head>
<body>
<div id="content">
  <h1>Ciccio and Suzanne</h1>

  <xml3d activeView="#defaultView" style="width: 600px; height: 400px;">
    <defs>

      <script id="customMaterial" type="text/javascript">
        function shade(env) {
          var ks = env.specularColor || new Vec(0.8);
          var tx = env.texcoord.mul(200.0);
          var sweep = Math.sin(env.time * .25) * tx.x() * 0.005;
          var px = tx.x() + tx.y() * (1.75 * sweep + 1.0);
          var modAmount = (px / 48) % 3;

          var tints0 = new Vec3(1, .5, .2);
          if (env.tints0 && env.tints0.sample2D) {
            tints0 = env.tints0.sample2D(env.texcoord).rgb()
          }
          var tints1 = env.tint1 || new Vec3(.3, .8, .4);
          var tints2 = env.tint2 || new Vec3(.3, .6, 1);
          var tint = modAmount > 2 ? tints2 : (modAmount > 1) ? tints1 : tints0;
          return new Shade().diffuse(tint.mul(Math.fract(modAmount)), env.normal)
                            .ward(ks, env.normal, env.tangent.xyz(), 0.4, 0.1);
        }
      </script>

      <material id="myMaterial" script="#customMaterial">
        <float name="ambientIntensity">0.1</float>
        <float3 name="specularColor">0.5 0.5 0.5</float3>
        <float name="time">0.5</float>
        <texture name="tints0" wraps="repeat" wraps="repeat">
          <img src="../textures/wood.jpg"/>
        </texture>
      </material>

      <lightshader id="light1" script="urn:xml3d:lightshader:spot">
        <float3 name="intensity">1 1 1</float3>
        <float3 name="attenuation">1 0 0.0008</float3>
        <bool name="castShadow">true</bool>
      </lightshader>

      <data id="mesh_data" compute="tangent = xflow.calculateTangents(index, position, normal, texcoord)">
        <data src="../../resources/meshes/suzanne.blst"></data>
      </data>

      <transform id="t_Suzanne" translation="-17 2 15" rotation="1 0 0 -2.2" scale="4 4 4"></transform>

    </defs>

    <group style="material: url(#myMaterial)">
      <mesh type="triangles" transform="#t_Suzanne" onclick="alert('Hi, my name is Suzanne!');">
        <data src="#mesh_data"></data>
        <float3 name="tints1">0.8 0.0 0.2</float3>
      </mesh>
      <group>

        <model src="../../resources/assets/robots/ciccio.xml#asset">
          <assetdata name="animation">
            <float name="key">0.5</float>
          </assetdata>
        </model>

      </group>
    </group>
    <group style="transform:  translate3d(-20px, 40px, 0px) rotateX(-90deg) rotateY(-30deg);">
      <light shader="#light1"></light>
    </group>
    <view id="defaultView"></view>

  </xml3d>
  <script type="text/javascript">
    // Keyframe animation with jQuery
    $("xml3d").on("framedrawn", function (e) {
      var key = computeKeyframe(e);
      $('*[name=key]').text(key);
    });
  </script>
</div>
</body>
</html>
```

SEE ME

RUNNING

POLYFILL

Just add a single script to your web page to enable XML3D in the browser. The Polyfill implementation [3] uses WebGL and JavaScript to emulate native XML3D support.

SHADE.JS

With shade.js [6], developers can write portable materials using JavaScript. It is renderer agnostic, adaptive and compiles to GLSL for forward and deferred rendering via OpenGL but can also compile to e.g. OSL for ray tracing and global illumination.

REFERENCES

In XML3D all resources can be referenced by a URI. A resource is predefined (e.g. often used light and material models), defined within the same document or in an external resource.

XFLOW

Xflow [2] is a declarative approach to define composable data resources as well as processing on this data via a dataflow graph. It can be used for geometry processing (e.g. skinning and morphing), animation, image processing (e.g. postprocessing and AR), and others. Its declarative approach allows mapping computations to the GPU and other parallel processors via different supported APIs (e.g. WebGL, WebCL, ParallelJS, SIMD.js).

BLAST

BLAST [4] is a novel transmission format for binary resources. It is streamable and offers a flexible approach to compression based on a code-on-demand approach.

EVENT ATTRIBUTES

Similar to HTML, XML3D defines events for elements in the scene graph like starting a JavaScript when the user clicks on an element.

SPECIALIZATION

Specialization of reusable resources is a general concept in XML3D that is achieved with partial overrides. Assets, materials, meshes and data in general can be specialized.

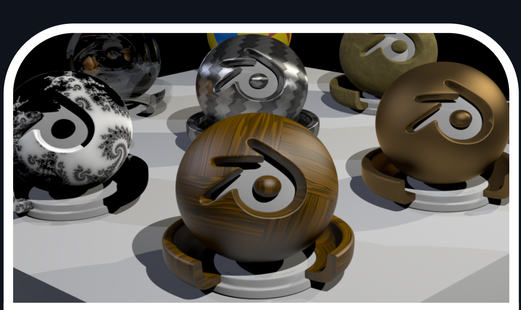
CSS

CSS can be used to style XML3D elements, e.g. to define transformations on groups using CSS3 3D transforms.

JAVASCRIPT

JavaScript can modify the web page - and thus the 3D scene - arbitrarily by modifying the DOM. Any of the numerous DOM libraries can be used, e.g. jQuery.

Renderer-Independent



Programmable Materials

Dynamic Scenes

Large Scenes

[1] XML3D: Interactive 3D Graphics for the Web
Kristian Sons; Felix Klein; Dmitri Rubinstein; Sergiy Byelozyorov; Philipp Slusallek
In: Proceedings of the 15th International Conference on Web 3D Technology, Web3D 2010

[2] Xflow: Declarative Data Processing for the Web
Felix Klein; Kristian Sons; Stefan John; Dmitri Rubinstein; Sergiy Byelozyorov; Philipp Slusallek
In: Proceedings of the 17th International Conference on Web 3D Technology, Web3D 2012

[3] xml3d.js: Architecture of a Polyfill Implementation of XML3D
Kristian Sons; Christian Schlinkmann; Felix Klein; Dmitri Rubinstein; Philipp Slusallek
In: Proceedings of the 6th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2013

[4] Blast - A Binary Large Structured Transmission Format for the Web
Jan Sutter; Kristian Sons; Philipp Slusallek
In: Proceedings of the 19th International Conference on Web 3D Technology, Web3D 2014

[5] Configurable Instances of 3D Models for Declarative 3D in the Web
Felix Klein; Torsten Spieldenner; Kristian Sons; Philipp Slusallek
In: Proceedings of the 19th International Conference on Web 3D Technology, Web3D 2014

[6] shade.js: Adaptive Material Descriptions
Kristian Sons; Felix Klein; Jan Sutter; Philipp Slusallek
In: Computer Graphics Forum (Proceedings of Pacific Graphics), 33(7), 2014

SIGGRAPH2015 Xroads of Discovery · DFKI · intel VISUAL COMPUTING INSTITUTE · FI-WARE · FIcontent · DFG · Bundesministerium für Bildung und Forschung · SEVENTH FRAMEWORK PROGRAMME