Accessing HTTP Interfaces within X3D Script Nodes

Manuel Olbrich*
Fraunhofer IGD

Abstract

X3D supports a variety of media types to be used in 3D scenes, like images, videos or other X3D models. A scene can dynamically load and replace this media during runtime, but since there is no way to communicate directly with outside sources like a server, all data sources need to be known in advance. This problem is usually solved by using interfaces like SAI, which allow external applications to modify the current scene. But this solution makes it necessary to set up all the communication via SAI and have the external application communicate with a server. In this paper, we will show how XMLHttpRequest, an object common in web browsers, can be used to handle the communication from within the X3D Browser. We will show how well this approach fits into the X3D environment and how easy this can be implemented in an X3D Browser. Afterwards, some examples will show the benefits in real applications and how easy this solution is to use.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Languages I.3.6 [Computer Graphics]: Methodology and Techniques—Standards

Keywords: X3D, ECMAScript, XMLHttpRequest, REST, Ajax

1 Introduction

Changing media in a X3D scene is literally as easy as writing the new url into the right field. This makes it easy for designers to create interactive scenes, that use different textures and models according to different user inputs. This works for scenarios, where the media urls are known upfront. It fails when the data depends on user inputs which must be interpreted by a server, or on data that is not available when the scene was loaded. A related problem is data generated by the user inside a X3D scene. There is no easy interface to store this data outside the scene.

A common solution to this is to use an external application to do the server communication. X3D supports this via SAI, which allows applications to directly access and modify data in the X3D browser. This SAI setup can easy get very complex, with listeners for changed fields and the implementation of a server communication.

An other way to get around this problem involves Java-based `Script` nodes, which can be allowed to bypass security and access the hard drive and network. The downsides of this method are relatively complex code in an additional programming language and modifications to the Java settings.

Both solutions decrease the portability of the X3D applications, since the target environment must be able to run the SAI-using application, or offer the modification of java security settings. This is complicated between different desktop operation systems, and next to impossible in the growing mobile sector. Therefore a solution that works from inside the X3D browser increases portability and simplifies deployment of these applications.

The XMLHttpRequest object, which can be found in every major web browser's JavaScript implementation, allows websites to access data on the server without loading a new website. This concept is often called Ajax, and can be found in nearly all modern web applications. Also data from the current page, like user inputs, can be communicated to a server without leaving that page, like a form submit would. Figure 1 shows in the "static" window the traditional approach where a page is requested and delivered to the web browser. The "Ajax" window starts with the same procedure, but the delivered page contains JavaScript code, which uses the XMLHttpRequest object to interact with the server without the need to load a new page.

This allows interactions which where next to impossible with the static approach, like offering suggestions while the user is entering text in a search field. Other examples would be a live chat in a webpage or collaborative editing.With these possibilities in a X3D browser, dynamic scenes can directly communicate with a server. This enables authors to build complex interactive scenes without depending on complex and often incompatible X3D browser extensions.

Native 3D support in web browsers with WebGL [Khronos WebGL Working Group 2011] opened the way for JavaScript based runtimes like X3DOM [X3DOM 2011]. This approach has the advantage of delivering X3D content to the user without depending on an installed X3D browser. With the X3D browser running in the web browser's JavaScript engine, native XMLHttpRequest support is available without any extra effort. Simple X3D applications which need to communicate with web servers can already be implemented in this environment and show the advantages of this approach.

2 Related Work

Combining the communication skills of XMLHttpRequest with X3D [Web3D Consortium 2008] scenes is not a entirely new idea. Other attempts exist, but they don't integrate XMLHttpRequest with the X3D browser, but use interfaces to communicate with a web browser. The web browser is running a JavaScript file, which is using the web browsers XMLHttpRequest implementation to communicate with the web and the previously mentioned interface to communicate back to the X3D browser. In contrast, our implementation integrates XMLHttpRequest directly into the X3D browser and makes it usable in Script nodes.

AjaX3D [Parisi 2006] controlled an X3D browser via its SAI [Web3D Consortium 2010] interface from inside a web browser, which natively provides the XMLHttpRequest object in its JavaScript implementation. To make use of the browsers implementation, the communication logic has to be in the websites javascript, which also needs to control and listen to the X3D scene. This approach has disadvantages, some of them are caused by the dependency of a X3D running as an applet inside a web browser, which is especially problematic for mobile or clustered setups. An other dis-

*e-mail: manuel.olbrich@igd.fraunhofer.de

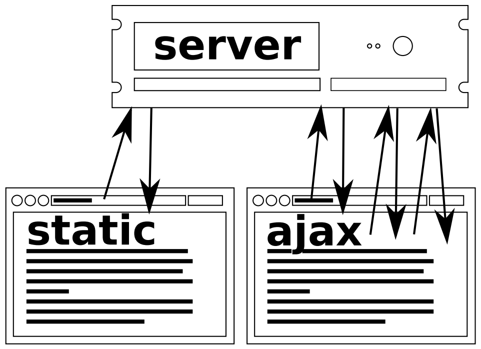

Figure 1: *Static page delivery vs dynamic pages which use Ajax technology to communicate with the web server*

advantage is the increased complexity of the setup, which decreases portability of the applications.

Another pattern to achieve connectivity from inside X3D scenes are Java-based Script nodes. By default, java applications running in their virtual machine have no access to resources like hard disk or network. Java security restrictions can be lowered or even disabled, which can be used to enable access to disk and network. This depends on the user for disabling the security, or a X3D browser without security restrictions for the Java runtime.

Other approaches like the interfacing of an X3D scene via an HTTP interface[Behr et al. 2004] can be used to access and change information in the scene, but are not fitted for initializing communication from within the scene, since there is no way of sending a message to an http client without an request.

3 Introducing XMLHttpRequest

XMLHttpRequest is available in all major web browsers, like Mozillas Firefox [1]. It allows HTTP client operations from inside a website, and is therefore one of the enabling technologies for the Web2.0. Even if the name indicates XML as data encoding, the object can be used for all kinds of data, like X3D or JSON. The XMLHttpRequest specification is available at the W3C website [W3C 2010].

A great advantage of adapting this standardized object from the web browser environment instead of creating an X3D specific one is the extensive available documentation. Since the XMLHttpRequest object is one of the key elements in most Web2.0 applications, many tutorials and proven patterns are available on the web.

3.1 Implementation

XMLHttpRequest isn't a native part of the ECMAScript language [ECMA international 2008]. It is an standardized extension [W3C 2010], which provides access to to the browser's HTTP client capabilities. Therefore it is not natively available in common ECMAScript engines like Google V8[2] or Mozillas SpiderMonkey[3] and needs to be implemented in the browser.

To make the XMLHttpRequest object available inside X3D Script nodes, the object itself has to be added to the JavaScript engine. The engines mentioned above provide extensive documentation on this topic. Writing code for network connections is often error-prone and hard to debug, but since X3D browsers already need to implement an HTTP client to fetch data, most of the functionality is already available. An alternative is to use a proven library like libcurl [Haxx 2010], which is already used in may applications. It offers client implementations for HTTP, but also for other protocols like FTP or SCP and includes SSL support.

[1]XHR in Firefox https://developer.mozilla.org/en/xmlhttprequest
[2]http://code.google.com/apis/v8
[3]https://developer.mozilla.org/en/SpiderMonkey

3.2 Basic Usage

XMLHttpRequest can be used in nearly every part of JavaScript to get data from a web server, or send data back. Asynchronous requests with callback functions can be used to minimize interruption to the user. The following example shows a simple request with XMLHttpRequest.

```
xhr = new XMLHttpRequest();
xhr.open('GET','http://localhost/test.txt');
xhr.send();
print(xhr.responseText);
```

The `XMLHttpRequest` object is used to fetch the file `test.txt` from the local webserver. The `open()` method prepares the request, and the `send()` method executes it. `responseText` contains the response body. The XMLHttpRequest specification also defines methods to access other information about the transaction, like headers or status codes.

3.3 Types of Data

There is no real restriction in datatypes which can be used with XMLHttpRequest, but usually text based formats are the most interesting for the use in `Script` nodes. Especially easy to handle are formats that can be directly interpreted by JavaScript, like XML or JSON [4]. Most of the webservice-APIs support at least one of them. The following is an example of JSON:

```
{  "type":"Material ",
   "diffuseColor":"0.1 0.1 0.9",
   "transparency":0.5 }
```

The JSON segment above describes a JavaScript object with 3 attributes: `type` and `diffuseColor` are strings and `transparency` is a number. This can be translated into an real JavaScript object with JavaScripts `JSON.parse()` or `eval()`.

3.4 Asynchronous mode

Communication with servers over the Internet takes time, and even requests over a local network can easily take enough time to be noticeable in a real-time 3D application. The asynchronous mode of XMLHttpRequest can help in this case. By providing a callback function with your request, your X3D browser can use the time to render some frames until the response arrives.

3.5 HTTP, Servers and Services

With HTTP as the transmission protocol, web servers like Apache, lighttpd[5] or Microsoft's ISS[6] are the first type of data sources that come to mind. They can be used to serve static data directly from the server's filesystem. The use of languages like PHP, Python, Perl or (in the case of ISS) ASP can be used to deliver dynamic content back to the client. That includes query results from databases, or even data gathered by request to other servers.

[4]JavaScript Object Notation
[5]http://www.lighttpd.net
[6]Internet Information Server http://www.iss.net

In the same way data can be stored, using HTTP's POST or PUT requests or as parameters of a GET request. PUT or POST are designed to carry the data in the request body, while the GET request usually has no body.

3.5.1 WebDAV

WebDAV[7] uses HTTP like a filesystem. The GET request, which is also used to access webpages with a web browser, is used to read files. Since the protocol is exactly the same, you can point your web browser to a WebDAV resource to download or view a file. In addition to this, WebDAV uses the PUT and DELETE requests to write and delete files. These can also be used with XMLHttpRequest to store and manage information. Depending on the used file format, this can also be used to exchange information between the X3D application and an external application. The following example shows the creation of new file named `new.x3d` with XMLHttpRequest.

```
xhr = new XMLHttpRequest();
xhr.open('PUT','http://localhost/new.x3d');
xhr.send("<X3D><Scene><Box/></Scene></X3D>");
print(xhr.status);
```

The structure is similar to the earlier example, but this time the request type is `PUT` and the `send()` method gets the request body as a parameter. In case of WebDAV, this request body will be the content of the new file. Another difference is in the print statement, where this time the response status code is printed. In case of success, it will be 200.

3.5.2 REST

REST (Representational State Transfer) [Fielding 2000] is an architecture pattern which describes how to address resources in an HTTP-based service. The perhaps most interesting constraint of REST for X3D application is that it is designed to be stateless, which enables authors to access data without a complicated session setup. REST is used by many services that are based upon HTTP. Especially web APIs for web applications adapt it to offer a structured interface to their services. Many mobile applications which act as an interface to a web service use a so-called RESTful architecture to communicate with the application servers. Popular web applications like Flickr, Twitter, Facebook[8] or Reddit allow access to the service with a RESTful API. These APIs usually use JSON or XML to deliver text-based content and image formats like JPEG or PNG for pictures. That means the requests and responses can be implemented without much overhead in ECMAScript and API calls which return images can directly be used as the URL for an `ImageTexture` node.

APIs for public web services have often some restrictions to prevent misuse of the API. The Flickr API [9] for example requires for some requests types (usually those which provide high resolution data or consume much time on the server) an identification key, which can be obtained for free. This way the service providers can easily lock a key if the application misbehaves. Other API functions that act like a user (like uploading photos or posting status updates) need the client to do user authentication with special API calls.

REST interfaces are also interesting with a 3D application as server. [Schiefer et al. 2010] describes the addition of a REST interface to OpenSG. Interfaces like this can be used to control a 3D application with a (mobile) web browser. In the context of this paper, this interface could be used to have a X3D application as an interface which controls another X3D application. An HTTP interface for the InstantReality X3D browser is described in [Behr et al. 2004], that can already be used to implement such setups.

3.5.3 CouchDB

Another example for a RESTful API is CouchDB [Apache Software Foundation 2011]. It is a lightwight database management system which is accessed via an RESTful HTTP API. Even the management interface, called Futon, is an internal website which uses the API calls to access and manipulate the databases. All data inside a database is stored and accessed via JavaScript Object Notation (JSON), which is optimal for JavaScript based applications, like dynamic websites or X3D scenes. CouchDB is a potent data storage for X3D applications, especially if they involve some kind of authoring or share live data with other applications. An example will be provided later in section 4.2.

3.5.4 Special Purpose Server

Since HTTP is a widespread and easy to use transfer protocol, many libraries which implement servers are available in most high-level languages. This makes it quite easy to implement your own server to use with the XMLHttpRequest object. Especially Python has proved to be useful for this, because it is easy to implement HTTP server functionality, and all the available libraries allow easy preparation of data, access to a wide range of devices and services. These kind of servers can be used to access resources which are usually out of reach for X3D applications.

3.6 Extending XMLHttpRequest for X3D

The XMLHttpRequest object on its own offers many uses in the X3D Script nodes. Every field type in X3D has a string representation, which allows easy representation in XML or JSON, but some of them are better represented in other forms. `SFImages` string representation consists of some dimensional information followed by the pixel data, but usually textures are stored in compressed image formats rather than this representation. Since ImageTexture nodes can use web resources, there is no problem for loading from HTTP sources like WebDAV. Writing in compressed image formats is not directly possible.

The easiest way around this problem is a conversion routine to translate `SFImage` field data into image formats like PNG or JPEG. This could be implemented separately from the involved objects, or either in the `SFImage` or `XMLHttpReport` objects.

For our reference implementation in the InstantReality framework [InstantReality 2011], the XMLHttpRequest object received an additional send() method which gets a second parameter to select the target format mimetype. With this setup, sending compressed SFImage data works almost like sending the uncompressed data. This can be used to store images as files on WebDAV, or to upload them to web services that work with image data, like Flickr or Facebook.

The implementation of this format conversion as a method of the SFImage object could be beneficial because it can be used without the XMLHttpRequest object. Also, this would be similar to the HTML5 Canvas object[10], which has methods to output its data as PNG or JPEG.

[7]WebDAV http://webdav.org/specs/

[8]Facebook API https://developers.facebook.com/docs/reference/api/

[9]Flickr API http://www.flickr.com/services/api/

[10]HTML5 - The canvas element http://www.w3.org/TR/html5/the-canvas-element.html

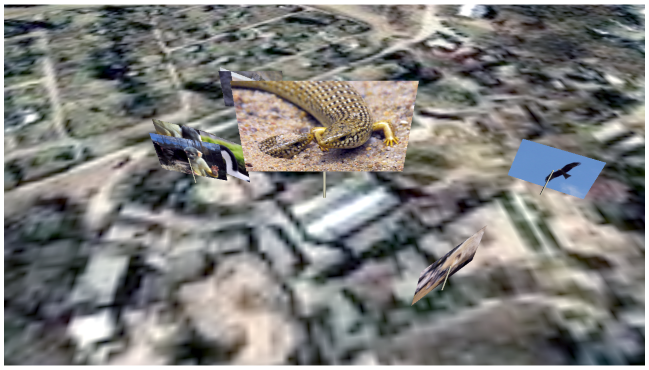

Figure 2: *Georeferenced images from the flickr web api presented on a virtual globe*

4 Applications

We have implemented the XMLHttpRequest in the current 2.0 release of the Instant Reality framework, and it has been used successfully in different applications and examples. A simple Example that shows how to read RSS is available in the example section at instantreality.org. The following section shows different use cases and if appropriate, some lines of code to show how it is used in the X3D context.

4.1 Presenting geolocated Images from the web

This example takes georeferenced images from the image hosting platform flickr and presents them on a virtual globe. The positioning of these images is realised with X3Ds Geospatial component. With the XMLHttpRequest objekt, the api server is queried for images with GPS data around an previously selectet spot on the globe.

The following code is part of the `Script` node which is responsible for the communication with the api server. The requests are simplified for readability, but show the whole process from getting a list of of images around the selected spot to retrieving the position for each individual image.

```
xhr.open("http://flickr.com/?method=search&\
  lat=49.8743&lon=8.6602&radius=3");
xhr.send();
picList=xhr.responseXML.photos;
for(var i in picList){
  xhr.open("http://flickr.com/?method=\
    geo.getLocation&photoid="+piclist[i].@id);
  xhr.send();
  var picPos = xhr.responseXML.location;
  var pos = new SFVec3d(picPos.@lat,picPos.@long, 0.0);}
```

Over time, the application keeps looking for new images from the selected area and adds them to the current set of images in the 3D scene. Figure 2 shows the application displaying images around a zoo. There is also an video attached to this submission which shows the application in action.

4.2 Store Data in a DB or web API

As mentioned before, CouchDB is a database management system, that uses HTTP for client connections and JSON as a data format. The combination between an XMLHttpRequest enabled X3D browser and CouchDB is currently used in an collaborative system and enables users in different locations to create annotations (spatially bound notes with additional information, like text and media) and share them with the other connected users.

The following example takes some simplified data for an annotation and stores it into the database. Reading from the database is similar to the example in section 3.2. Some more examples to the use of CouchDB can be found in the tutorial on XMLHttpRequest[11] in the InstantReality framework.

```
var newAnnotation = '{ "text":"Interesting spot", \
   "position":"4 1 2",  "orientation ":"0 1 0 1.234"}';
xhr.open('PUT','http://localhost:5984/annotations/a123')
xhr.send(newAnnotation);
```

5 Conclusions

The XMLHttpRequest object has shown its possibilities in the rise of the Web2.0. The adoption of this object into the X3D browser doesn't only allow Ajax operations from inside X3D worlds, but also offers solutions for common problems like data storage. Data created inside a X3D application can be written to a local or remote server and reused in a different session or applications. Even exchange and synchronization between different simultaneous sessions is possible. The usability of this extension was shown with several examples.

References

Apache Software Foundation, 2011. The Apache CouchDB Project. http://couchdb.apache.org/.

Behr, J., Dähne, P., and Roth, M. 2004. Utilizing x3d for immersive environments. In *Web3D 2004 Symposium*, Web3D '04, 71–78.

ECMA international, 2008. Standard ecma-262 - ecmascript language specification. http://www.ecma-international.org/publications/standards/Ecma-262.htm.

Fielding, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis.

Haxx, 2010. libcurl - the multiprotocol file transfer library. http://curl.haxx.se/libcurl/.

InstantReality, 2011. Javascript - xmlhttprequest. http://doc.instantreality.org/apidocs/scripting/javascript/class XMLHttpRequest.html.

Khronos WebGL Working Group, 2011. Webgl specification. https://www.khronos.org/registry/webgl/specs/1.0/.

Parisi, T., 2006. Ajax3d: The open platform for rich 3d web applications. (original site is offline.) http://replay.waybackmachine.org/20090207131321/ http://ajax3d.org/whitepaper/.

Schiefer, A., Berndt, R., Ullrich, T., Settgast, V., and Fellner, D. W. 2010. Service-oriented scene graph manipulation. Web3D '10, 55–62.

W3C, 2010. Xmlhttprequest - w3c candidate recommendation 3 august 2010. http://www.w3.org/TR/2010/CR-XMLHttpRequest-20100803/.

Web3D Consortium, 2008. X3D. http://www.web3d.org/x3d/.

Web3D Consortium, 2010. Scene access interface(sai). http://web3d.org/x3d/specifications/ISO-IEC-19775-2.2-X3D-SceneAccessInterface/.

X3DOM, 2011. X3DOM - a DOM-based HTML5/ X3D integration model. http://x3dom.org.

[11]http://doc.instantreality.org/tutorial/http-communication-in-ecmascript-with-xmlhttprequest/