Sound and Audio extensions in X3D v4

Efforts to Improve X3D Audio
1st Approach Spatial Sound in X3D/X3DOM

- “Wrapping” X3DOM around Web Audio API
- Introduction of spatial sound components in the X3DOM framework, based on:
  - X3D specification and
  - Web Audio API
Web Audio API - Structure

- Web Audio API involves handling audio operations inside an audio context (AudioContext node) and has been designed to allow modular routing.

- Particularly, the approach of Web Audio API is based on the concept of audio context, which presents the direction of audio stream flows, between sound nodes (AudioNode).

- A simple, typical workflow for web audio API. This flow includes the creation of audio context. Inside the context:
  - create sources — such as <audio>, oscillator, stream
  - create effects nodes, such as reverb, biquad filter, panner, compressor
  - choose final destination of audio, for example your system speakers
  - connect the sources up to the effects, and the effects to the destination.
Web Audio API – All Registered Nodes

- WebAudioAPI Node
  - AudioSound Node
  - AudioSource Node
  - AudioContext Node
  - AudioNode Node
  - AudioParam Node
  - OscillatorNode Node
  - AudioBuffer Node
    - AudioBufferSourceNode Node
    - MediaElementAudioSourceNode Node
    - MediaStreamAudioSourceNode Node
  - BiquadFilterNode Node
  - ConvolverNode Node
  - Delay Node
  - DynamicsCompressorNode Node
  - GainNode Node
  - WaveShaperNode Node
  - PeriodicWave Node
  - AudioDestinationNode Node
  - MediaStreamAudioDestinationNode Node
  - AnalyserNode Node
  - ChannelSplitterNode Node
  - ChannelMergerNode Node
  - AudioListener Node
  - PannerNode Node
Main Interfaces - Role

- The **AudioContext** interface, which contains an audio signal graph representing connections between AudioNodes.

- The **AudioNode** interface, which represents audio sources, audio outputs, and intermediate processing modules. AudioNodes can be dynamically connected together in a modular fashion. AudioNodes exist in the context of an AudioContext.

- The **AudioBuffer** interface, for working with memory-resident audio assets. These can represent one-shot sounds, or longer audio clips.

- The **BiquadFilterNode** interface is an AudioNode for common low-order filters such as Low Pass, High Pass, Band Pass, Low Shelf, High Shelf, Peaking, Notch, Allpass.

- The **DelayNode** interface introduces a dynamically adjustable variable delay.

- The **GainNode** interface, an AudioNode for explicit gain control. Because inputs to AudioNodes support multiple connections (as a unity-gain summing junction), mixers can be easily built with GainNodes.
Main Interfaces - Role

- The **PannerNode** represents a processing node which positions / spatializes an incoming audio stream in three-dimensional space.

- There is a single listener (AudioListener) attached to the Web Audio API context that can be configured in space through position and orientation. Each source can be passed through a panner node (AudioPannerNode), which spatializes the input audio.

- Based on the relative position of the sources and the listener, the Web Audio API computes the correct gain modifications.
Proposal to Improve X3D Sound Component – Introduction of Web Audio API components
Sound Node (X3D)

```javascript
x3dom.registerNodeType("Sound", "X3DSoundNode", defineClass(x3dom.nodeTypes.X3DSoundNode, function(d) {
    x3dom.nodeTypes.X3DSoundNode.superClass.call(this, d),
    this.addField_SFNode("metadata", x3dom.nodeTypes.X3DMetadataObject),
    this.addField_SFVec3f(d, "direction", 0, 0, 1),
    this.addField_SFFloat(d, "intensity", 1),
    this.addField_SFVec3f(d, "location", 0, 0, 0),
    this.addField_SFFloat(d, "maxBack", 10),
    this.addField_SFFloat(d, "maxFront", 10),
    this.addField_SFFloat(d, "minBack", 1),
    this.addField_SFFloat(d, "minFront", 1),
    this.addField_SFFloat(d, "priority", 0),
    this.addField_SFNode("source", x3dom.nodeTypes.X3DSoundSourceNode),
    this.addField_SFBool(d, "spatialize", 10)

    // New attributes
    // In order to use it in html under the Sound node of X3D
    this.addField_SFNode("transform", x3dom.nodeTypes.Transform),
    this.addField_SFNode("panner", x3dom.nodeTypes.PannerNode),
    this.addField_SFNode("filter", x3dom.nodeTypes.BiquadFilterNode),
    this.addField_SFNode("delay", x3dom.nodeTypes.DelayNode)
})
```
X3DSoundSourceNode Node

X3DSoundSourceNode(X3D)

```
x3dom.registerNodeType("X3DSoundSourceNode", "X3DTimeDependentNode",
defineClass(x3dom.nodeTypes.X3DSoundNode, function(d) {
    x3dom.nodeTypes.X3DSoundSourceNode.superClass.call(this, d),
    this.addField_SFString(d, "description", ""),
    this.addField_SFBool(d, "loop", !1),
    this.addField_SFNNode("metadata", x3dom.nodeTypes.X3DMetadataObject),
    this.addField_SFTime(d, "pauseTime", 0),
    this.addField_SFFloat(d, "pitch", 1),
    this.addField_SFTime(d, "resumeTime", 0),
    this.addField_SFTime(d, "startTime", 0),
    this.addField_SFTime(d, "stopTime", 0)

    // New attributes
    this.addField_MFString(d, "url", []);// From AudioClip X3D Node
    this._audio = document.createElement("audio"); "Microsoft Internet Explorer" != navigator.appName &&
    document.body.appendChild(this._audio)
})
```
AudioContext Node

**AudioContext**: Sound {
  SFNode [in,out] metadata NULL [X3DMetadataObject]
  SFDouble [in, out] latencyHint 0.0 (-\infty, \infty) //optional or
  SFString [in,out] latencyHint “”
  SFFloat [in,out] sampleRate 0 (-\infty, \infty) //optional
}

The `AudioContext()` constructor creates a new `AudioContext` object which represents an audio processing graph built from audio modules linked together, each represented by an AudioNode.

### Syntax

```javascript
var audioCtx = new AudioContext();
var audioCtx = new AudioContext(options);
```

### Parameters

- `latencyHint` [optional]
  - The type of playback and the context will be used for, as a value from the `AudioContextLatencyCategory` enum or a double-precision floating-point value indicating the preferred maximum latency of the context in seconds. The user agent may or may not choose to meet this request; check the value of `AudioContext.baseLatency` to determine the true latency after creating the context.

- `sampleRate` [optional]
  - The sample rate to be used by the `AudioContext`, specified in samples per second. The value may be any value supported by `AudioBuffer`. If not specified, the preferred sample rate for the context’s output device is used by default.
AudioNode: AudioContext {
  SFNode [in,out] metadata NULL [X3DMetadataObject]
  SFInt32 [in,out] numberOfInputs 0
  SFInt32 [in,out] numberOfOutputs 0
  SFString [in,out] channelCount 0
  SFString [in,out] channelCountMode max
  SFString [in,out] channelInterpretation speakers
}
Current Status (1)

• we cloned the Web3D Project in my GitHub account, made the changes in the file `sound.html`
• we finished with a Pull Request
Current Status (2)

- GitHub code of a simple demo "x3domSpatialSoundDemo".
- This demo is a method for the introduction of spatial sound components in the X3DOM framework, based on the above proposed extra sound nodes in X3D for the ISO v4.
<Transform DEF='Audio1' translation='-900 0 -900'>
    <Shape>
        <Appearance>
            <ImageTexture url="images/loudspeaker.png"/>
        </Appearance>
        <box size="100 100 100"/>
    </Shape>
</Transform>

......

<Sound playbackRate='1.0'>
    <Transform USE='Audio1'/>
    <PannerNode position='0 0 0' orientation='1 0 0' velocity='0 0 0' coneInnerAngle='360' coneOuterAngle='360' coneOuterGain='0' distanceModel='inverse' maxDistance='1' panningModel='HRTF' refDistance='1' rolloffFactor='1'></PannerNode>
    <X3DSoundSourceNode loop='true' url='"sound/africa/piano.mp3" "sound/africa/piano.ogg"'/>
</Sound>
Next Steps

- Feedback for the Proposed Approach
- New Demos using more proposed extra sound nodes in X3D (e.g. Filters)