Google

Path to native spherical video

David Dorwin
Google Chrome Media & VR

Near Term: App-Driven Presentation Using WebVR

Current state

- Non-rectangular media formats (including projections) are not standardized
- Native <video> presentation is not standardized and the path is unclear

Current solution

- <video> ➡ WebGL ➡ WebVR
- App determines and applies the projection
  - Adaptive use cases may require WebGL extensions for sync (e.g. crbug.com/639174)

In the near term, libraries will abstract this for developers

- VR View: github.com/googlevr/vrview
  - Preview of 2.0 (with WebVR support): github.com/googlevr/vrview/tree/2.0-pre1
  - Supports headsets as well as magic window

Native Prerequisite: Standardized Media Metadata

Projection and 3D information should be specified in the container.

- These are properties of the media (like resolution, bit depth, etc.)
- Media can be understood without accompanying app or external metadata
- Avoids app and media getting out of sync
- Avoids duplicate standardization efforts around projection specification
- Codec independent

Active proposal for ISO BMFF (MP4) and WebM:

- github.com/google/spatial-media/blob/master/docs/spherical-video-v2-rfc.md
- Includes custom projection using mesh projection

Recommended: Developing VR Media Technologies talk at Demuxed 2016

Native Presentation

Base Assumptions

- Native support for spherical rendering is desirable for latency, performance, DRM, or other reasons.
- Metadata has been standardized.
- The browser processes the metadata and handles projection.
- Must support all clients and users, including those with and without headsets.
    - For example, present spherical video in a rectangular “magic window” on traditional screens.

Simplest Approach:
Browser provides UX to select spherical rendering

Concerns about this approach:

- Only enables a limited video-only experience with default controls
  - No custom controls, custom thumbnails, text overlays, etc.
- Therefore, the value over the WebVR solution is questionable
- Cannot be exposed via custom controls without new JS API(s)
  - Thus, it requires W3C spec additions.

W3C Recommendation track seems unlikely given the limitations

More Complete Approach: Spherical/Immersive DOM

Given a spherical DOM presentation, spherical <video> is like any other element.

- Does not require new video-specific APIs.

To render spherical video, set the <video> dimensions to 100%.

Custom controls, custom thumbnails, text overlays, etc. are all supported.

This will take longer given the dependency, but it is much more powerful.

Related Topics

Supporting Encrypted Media

All solutions: Highest robustness requires platform/HW support for secure textures

WebVR solution: Requires user agent and/or platform support for opaque and/or tainted video textures

- To prevent readback via JS or GL

Native solution: May only requires support for final composition and distortion

Applies to both spherical video and 2D video in VR (e.g. virtual theater)

Media Capabilities

Spherical and 3D video require powerful decoders (i.e. 4K and greater) and may use nonstandard dimensions (i.e. left-right).

The ability for apps to detect client capabilities will be more important than ever.

The Media Capabilities API is being incubated

- Proposal currently includes width, height, bitrate, and framerate
- Get involved!