W3C Immersive Web Community Group

September 17th AR Face-to-Face

Action items that came out of the conversations:

Issues to create:

- An issue on the CSS WG to make sure that we're not making conflicting enums for blend mode (**Nell**)
- An Issue about using 2D UI for AR mode (**Brandon**)
- An Issue to explore what AR-lite would need to include separate from hit-testing (**Nell**)
- An Issue to explore coordinate systems based on Max's PR (**Max**)
  Does XRFrameOfReference need to continue to exist
  Feature detection? E.g.
  3DOF vs 6DOF?
  Can I do anchors/hit tests?
  Do I have a floor?
  Do I have a bounded environment?
- An Issue about how semi-transparent real world objects would look (**Kip**)

John will add to the privacy/security explainer that if you create a series of local coordinate systems with geo-alignment you may be able to infer latitude.

Attendees

Nell Waliczek - Amazon
Brandon Jones - Google
Alex Turner - Microsoft

Iker Jamardo - Google
Kip Gilbert - Mozilla
Blair MacIntyre - Mozilla
John Pallett - Google
Gen Mak - Magic Leap
Rik Cabanier - Magic Leap
Jordan Santell - Google
Michael Blix - Samsung
Damon Hernandez - Samsung
Laszlo Gombos - Samsung
Alexis Menard - Intel
Max Rebuschatis - Google
Andrew Hall - Microsoft
Chris Coniglio - North Kingdom
Artem Bolgar - Facebook

Remote:
Leonard Daly - Daly Realism
Tony Hodgson - Brainwaive
Trevor F. Smith - Transmutable
Chris Wilson - Google
Daosheng Mu - Mozilla

Nell:
The wg charter call for responses ended last Friday [37 votes in favor, none against! -CW]. Chris and Ada, Trevor & Brandon and I will communicate how we move work from the CG into the WG and how work will get done.
Getting started a little late, but the topics in the afternoon are a little optional so if we don't get to the afternoon agenda items that's OK.
We're shuffling things around a little, moving the coordinate systems around a bit.
Goes over the agenda items...

WebXR Modes

Brandon:
Summarizing Issue #394 in the WebXR repo.
The first thing that we want to ensure is that everyone in the room agrees that having AR be an explicit mode works for you, your platform, and your users.
Quick straw poll: Does anyone feel like restricting AR to a modal switch in the browser seems problematic?
Q: What's the alternative?
Brandon: The alternative was to allow AR on a phone (not clear how it would work on headsets) you'd have a single inline canvas backed by the AR video feed that could render tracked content on top of if, presented as part of the context of the larger page, so with scrolling and other content.
From a tech perspective it should just work, just by projecting the video into the back of the canvas.
There were security concerns and larger concerns about headsets. What does that mean for AR inline when using a headset browser?
Fundamentally, this option is the difference between letting AR content be positioned in the DOM or do I need to shift into a full screen style mode.
Nell:
Is there an inline mode in this proposal?
Brandon:

There would be no inline mode for AR. But, there could be inline session display, just no AR camera data or environmental data.
So, the inline would be just for non-AR sessions, for example z style head tracking to change the POV using head tracking.
Blair:
Back to the question of what's missing, it's important to couple it with the idea of the DOM overlay. With Argon, the one thing that people wanted to do is do the equivalent of taking "AR" and slide it into the corner to reveal controls. So, apps could show non-overlay UI even while the AR session is still working.
Brandon:
Yes, that is important. Sorry for skipping over bits. It is definitely important to be a way to present DOM content along with it. The exact format of how that works needs iteration and talking through. The easiest mental model would be to implicitly treat the video as full screen and lay the DOM on top of it.
Blair:
It still works, but it's a shift in mindset for the developer.
We never found any apps that couldn't be build this way. I don't think it restricts us in any meaningful way.
Nell:
We also talked about enabling 2D overlays that track against real world positions. We experimented with how that might work, but the rough idea is that given a coordinate system and a point in that system, does that correlate to a point on the screen?
One concern is where the difference between handheld and headset AR occurs when you have these overlays. How do we use existing web tech to build UI that works in both 2D and 3D? What are the implications of overlay when you move into headsets? We;ve batted around ideas, nothing that felt great.
One idea was for headset browsers to take the overlay and represent it on as a virtual slate that the user can hold, but that's a lot for browser developers to implement and can be confusing to web devs.

Another option is that overlays just don’t work in headset AR.
We don’t yet have clear answers, but that question is somewhat orthogonal to whether there’s an AR mode.
Brandon:
Thanks, Nell.
Blair:
I brought up the DOM because full screen in AR on handsets, the requirement of full 3D UIs gets a lot of pushback. Implementing them in GL is a huge burden, so allowing DOM based overlays is necessary for handhelds.
One of the reasons I like the immersive handheld mode is that it should be possible to implement the AR data it more efficiently without it being “in” the DOM.
Brandon:
That comes down to an implementation detail. The current proposal in this Issue, has three modes: inline (no AR, magic window), immersive VR, immersive AR.
The DOM would be either requested as an overlay or have a separate mode for “2D AR” or “3D AR” where the former has an overlay. A lot of people would like to be able to bring 2D UIs into headset AR.
Iker:
Trying to think about possible use cases...
We should understand if there are not a lot of use cases where the page has embedded (inline) content with AR.
For smartphone AR, I could see a use case where the canvas would take up a smaller part of the page, like a thumbnail, or cropping it. An open question of whether people want to use this.
I can see in the future with high rez VR HMDs where you take a browser with you, like a tablet, as a magic window into the real world or some other virtual space. This might be a “crazy” use case.
Brandon:
That sounds like a fun use case! If you’re in a headset with a restricted field of view and looking through a magic window with another restricted field of

view. Maybe you want to peek into a space before allowing it to take over my entire field of view.
This proposal does not banish inline AR forever, we can add one later, but it wouldn't be in there when we start.
Chris:
Let's talk about privacy concerns later.
As a developer, it's like the video element where there are two modes, inline and full screen which the user can toggle on and off. On headsets perhaps you only have fullscreen.
Brandon:
From your perspective, how problematic is it if developers create AR content that should run on anything that is "AR" but they've specified that it only works on a phone screen.
Chris:
That's always happen on the web. They might just develop it for ARKit on iPhone X's, but that shouldn't necessarily inform the API design.
Brandon:
Ok, thanks. At the end of the day we can't stop people using UA strings, but it's a question of how to create the pit of success. So, do we want to funnel people toward creating content that can be used everywhere?
Nell:
At Microsoft we would often see devs hard code to a UA string but if we faked it the content just worked. So, we should do our best to make it harder to make experiences that only work on one platform.
Kip:
It seems there will be local ways to present content in multiple ways. How should the user make those decisions.
Brandon:
You're talking about for example a phone with a wireless headset attached to it. So, the phone could do phone AR or headset AR.
That's a good question, which ties into other questions about whether we need an XRDevice and into session creation, which I'll avoid for now.

I would think that the UA is in the best position to make those decisions. So, if we have a specific mode transition then the UA could help with that. If I hit the “go into AR” button then the UA could offer the user a choice between places to present it. Maybe there are defaults or the user says “remember this decision” but I don’t think we want to push that onto the site because there’s no right decisions.
For VR and AR modes, for example a Pixel phone with a Daydream headset, the AR and VR modes in this proposal creates an explicit split.
In the future we can add the neural implant scenario. :-)
Brandon:
My goal isn’t to solve the API surface today, but to get nodding heads about this general plan and then we can iterate on the API on the way to TPAC.
John Pallett:
Looking at PR #11 in privacy and security repo.
One general point here is that any time the user has data accessed by the site, the user could get into the habit to clicking “OK” and AR mode is one solution.
Another point is that the number of items that the user consents to could be many: camera, object tracking, point data, etc.
It might not be clear that the AR session would be connected to the permissions. So, it’s hard to explain what they’re giving up and how it’s connected to the AR session.
The longevity of the session is also a question..
Consent vs. Notification...
(read the Issue for more details -TFS)
Nell:
This Issue is against the repo whose goal is to create a document that is not WebXR specific, to be used as a checklist to work through for different specs like WebXR Device API. That’s why it’s not in the WebXR repo.
Brandon’s Issue in the WebXR repo is a reference to how to WebXR applies these ideas.
Nell:
I’ll be at the W3C permissions workshop next week, and a lot will be more

clear after that. We'll work with the experts there and then bring that back to this conversation.
John P:
AR mode allows specific consent. (see Issue -TFS)
We looked at something like fullscreen, which has a defined session, duration UI, and instructions. So there are three benefits to AR mode: flexibility, clear scope, and enforcement.
Flexibility: More UX opportunities to present detail info, to ask for consent or send notifications, and to customize across form factors.
Clear scope: It will be clear, like fullscreen mode, that they are entering a specific mode and how to leave the mode.
Enforcement: The UA can enforce the data access and tie it directly to the notification or consent.
Brandon:
One clarification; if you have an inline session (even one that is user action started) the scope of that isn't well defined. When the user sees a camera feed on their page, especially with the phone, it sets off bells that the page is doing something with the data, regardless whether the UA is sharing it. If you have inline content, the page could sneak in a permission for it and then hide the element so the use doesn't know its happening. AR mode makes that more explicit.
Nell:
Related question; the camera API is like that today, as is geolocation. Have your security folks dug into that?
John Pallett:
If you give permission to geolocation it is much less important than camera data. The camera data is sort of the worst case scenario.
If we have geo-alignment, it's an API that's included in the AR mode and it follows the rules of expiring with the session.
We didn't talk through what if the user has given the site a lot of other permissions, do we need to ask for AR mode permission.
Max:
It's interesting that this camera issue exists today.

People are going to do AR on devices without the WebXR API, so I don't know if we want to think about it but if we set up too many barriers when people will just use camera access and their own SLAM to make AR happen without all of the permissions.
Brandon:
I'm very cognizant of this. There is a real risk where the polyfill is more featureful that the actual feature. The page could ask for one permission, camera data, and then is reasonably good.
I don't have solution to that, because it's not a one-time issue but we need to be aware of the risk at every step.
One of the things that we can do to combat that is to provide access to unique features, like transition between displaying on a phone screen and on an AR headset.
Andrew:
I think that AR modality, but by eliminating inline AR we're missing a point. I don't see why we can't have inline, too, like how fullscreen allows devs to make any element full screen. It seems like a balance we can strike.
I can imagine a world where extensions offer value, or multiple iframes that show AR views.
Brandon:
We're actively trying to ask permissions for camera data, so we want a modality where we can show a camera feed and the user doesn't freak out. That's a hard thing to communicate and concerning.
The page could make a label next to it and claim that it has access to the video for blackmail.
The proposal increases the user's confidence in what's going on. Inline AR isn't a completely insurmountable problem, but an explicit mode switch simplifies the problem space and gives us extra options.
We're not saying that inline AR will never work, but it seems more tractable to start here.
Nell:
We're at 10:25 but we'll take

Rik:
What would the UI be for full screen AR mode?
Brandon:
Like fullscreen is today.
In VR, obviously this is a little newer in terms of UX. I've noticed that on all platforms there's some sort of physcial button to jump out of WebVR or back to the home screen, so in many cases there's a natural mechanism. In Chrome, when we go into VR there's a toast that tells the user to press the "app" button to exit WebVR.
Blair:
The uses cases you suggest are useful. I will say that the one reason I really like this proposal is that eventually we'll end up with inline AR but I really want full screen and implementing it efficiently is more important than inline.

--- BIO BREAK ---

Blair:
Paraphrasing Leonard from chat: What if you have a system with a lot of cameras for AR? Also, what if you have an AR device with multiple screens, does AR mode take over all screens?
Brandon:

- Imaging multiple monitors -

That in particular is a good question and similar to multiple display modalities. Probably a user agent decision as it'll be in a better position to offer a solution.
Alex:
One thing about privacy settings. I like the idea of capturing it into modes, but the devil is in the details.
Talking about sensitivity of data and the camera giving you so much. In our headsets, some data is free but camera is an extra thing that the app has to ask for. Only some apps at certain times use it, like streaming a view for a remote user.

Brandon:
Different UAs on different hardware have different affordances. So, a phone is different than a see-through headset in terms of permissions.
Alex:
I like that the fake access to camera, like a fake camcorder, so at some point for certain devices you have to convey camera access.
So, how do we reason through the permissions. Do we have room for devices where camera access is a higher level of access?
Brandon:
Permission elevation is important, the API shape should encourage bundling at one time, but we do need to allow for the page to ask for more later during a session.
Like the app lets you position a table but then later you want to take a photo of the result to share. At that later point the user should be able to increase the permissions.
Nell:
That’s a big topic for the event next week.
Leonard:
I'm not sure I agree with Brandon's position that it's the browser's responsibility and not the spec, since the spec is specifying what should be done.
Blair MacIntyre:
Is this something were the spec could provide advice or best practices? since it seems like different UAs might want to do differen ttihngs
Leonard Daly:
That might be better. My biggest concern is codifying something that states or implies that the content needs to provide the "last" word on what it could do with the data.
Brandon:
I’m being overly flippant about it. Yes, there are few times where the spec talls the UA to provide a user experience or a UI. It does more to provide API shape and determines what will happen. There is room for non-normative recommendations for UA UI.

Alex:
We have to reason about whose job it is to do the composition. For example, where does the app's alpha channel come into play? So if the alpha channel is ignored on some devices but not others, we'll get different results.
Brandon:
This is taken care of in some ways with blend mode; alpha, opaque, and additive. The text for that says what should happen with the alpha channel. So, the blend mode is for not just the GL context but for the camera data, too.
Nell:
We should file an issue on the CSS WG to make sure that we're not making conflicting enums for blend mode.
Alex:
In terms of the AR mode, if the app does want to do its own composition is there a way to allow that?
Brandon:
I think that would need some careful consideration.
Alex:
Right, we should figure that out.
Brandon:
If we provide a way to provide a camera feed as a texture and then you'll get really weird results on a see-through display. I think we want to funnel people into the mode for controlling how it's composited to the DOM but not the environment
Alex:
Great, I was worried we were talking about something else.
Kip:
I've added some details to Issue #398. We need to be careful to differentiate between alpha correct or pre-multiplied blend (I think - TFS) In some cases it's only additive (on see-through) but sometimes the content wants to choose how it wants to blend.

Brandon:
This might have something to do with WebGL work around ???
Taking a moment to recap hallway conversations: Regardless of what we do with an AR mode, it is my opinion that we must allow some way that DOM content appears with it. If we don't then the polyfill wins.
So, to me having some sort of interaction with DOM content is feature critical. I don't know the details, but it's fundamental.
The second thing is it's not entirely clear how that interacts when you're using a headset. I think of it as a compatibility mode. In one case you have something that's really designed for a phone screen, but I'm on a headset so would I prefer to block that content because it wasn't designed for my platform but kind of works? My suspicion is that most headset manufacturers will want to do something like that just to have the most content available. But, I do think the API should include the tools to know when its on a headset and know to do the right thing for that platform.
We'll work through that, hopefully in time for TPAC. For today, do we have a general sense that nobody thinks this is a terrible idea.
Rik:
I don't understand why this two things are intrinsic.
Brandon:
I'm saying that if we take away DOM rendering (and we've seen this in VR) then we take away a large reason to use the web. To me it is critical to provide the option to provide DOM content in a space.
Nell:
The key is that in headset AR mode then the DOM isn't in line with the environment.
Brandon:
There's a path where the device can indicate that it doesn't support DOM overlay. Whether that's a reasonable route is open for discussions.
I'm hesitant to tell browsers that they have to support DOM rendering. It's natural on a phone but not necessarily on a headset. So, there's a balance.
Max:
We should bring up specific use cases.

Rik:
This has been tried so many times before and failed.
Brandon:
Like where?
Rik:
Everyone tries to render DOM elements on a texture.
Brandon:
Yes, it's hard which is why we haven't gone down that path. So, that's why it should be optional and the headsets should be able to refuse.
Nell:
This is not DOM to texture. We are not looking at a general solution. We're looking at something a lot more like the fullscreen API (but isn't exactly it) but the idea is that we need *some* mechanism for a DOM element to act as an overlay. So, there's more browser developer burden on some platforms. But, we want to avoid a whole set of issues around general DOM to texture and we want the overlay to be optional.
Gen Mak:
Isn't having an overlay not immersive? I think the concern where some of the things we want is fully immersive (phone in front of face, ML1) but this is an AR mode that is not immersive.
Blair:
In this context "immersive" means "takes over device" (more like "exclusive")
Gen:
We want to make sure from a dev perspective they want to use the web to make an "inside the web" experience and the other perspective of the "inline" and we want to make sure that we can enable one without cutting off the other.
John Pallett:
I think there's a semantic issue. There is a desire to have an immersive mode and DOM to texture doesn't solve that problem. Having sites have access to DOM element textures is a security problem, but having an API

where DOM elements are composited by the UA into the scene is less of a problem.
Brandon:
Again, I want to stress that we're not presenting API surface. One of the initial proposal for Issue is to have four modes: inline, AR with overlay, immersive VR, & immersive AR. So, we explicitly separated out the modes. And there may be real benefits to doing that for pages deciding what is supported. This is an option.
At the end of the day the UA has to tell the page whether or not it provides overlay.
Nell:
We need to figure out whether we want to support content for handset AR when you're in a headset AR. Is there a path where we want to enable sensible fallbacks for AR with overlays when the user is in a headset?
Brandon:
A browser maker should be able to say "no" to content with overlays but some browsers should be able to offer that as an option. The spec won't force a UA to support overlays.
Kip:
I had similar thoughts about disambiguity of multiple layers of DOM layers so I went through an exercise thinking about a world where we have inline AR, AR with overlay, and immersive AR. It has to do with exclusivity and field of view. So, in a headset you might have two virtual portal views, not just one handset.
So, this proposal fits into that future world.
One question about inline: Would there be a requirement for content that supports inline to (in headsets) support stereoscopy?
Brandon:
Like the z-space devices with the API we could tell the page to render a certain way to work. So, content could say that it will only support a single camera and not head tracking. And there's the question of permissions for head-tracking. For stereo, it is partly a question of performance. Maybe there's an opt out. It's a good question. We probably just need to iterate.

Nell:
And it's a little orthogonal to AR mode or not.
Alex T:
Blair mentioned Argon apps and some of them would work great in a headset showing the UI, but some of the UI is determined by what is behind the UI, like a reticle that changes color based on what's behind it.
Blair:
Argon was weird because people could do 3D CSS transforms and mix DOM and GL content and video. The DOM content is just so easy to create. It performs like crap but who cares?
People would style text with white on one edge and red or black on the other so that it shows up no matter once. The HTML stuff looks way better than UI made with WebGL.
Iker:
In the smartphone AR it makes total sense to have DOM content but in headsets not so much.
Max:
If we can't say that a shopping app will work across handheld and headset mode then we're screwed.
Iker:
Nobody is saying that this isn't possible.
Nell:
Well, we are saying that UAs can choose.
Brandon:
I feel like that's saying that if a shopping app doesn't support handheld AR and Daydream VR then it's a failure. I don't think that's necessarily true. Orgs will release new AR headsets and what percentage of content made today will work on those? 0%.
We shouldn't force webdevs to make everything work all of the time. We should create the pit of success, but there's not much we can do to stop people who only want to make AR.

Max:
Sure, but we do want to support people who want to work across platforms.
Blair:
I don't see how this proposal prevents that. I would envision a shopping app with this proposal would work on a [flat, portal, and immersive - TFS]. I can see making it work in flat mode, then with DOM overlays in full screen, then in headsets that don't support overlays the page could implement it using a GL framework.
Brandon:
Ok, we're at 11:30. This is a great conversation. We're going to let Chris ask his question, then a straw poll, then close the conversation for today.
Chris:
This reminds me of webaudio work, doing spatial content in the audio space. It seems to me that a good way to think about this problem would be to iterate what the device supports and then choose how to present the content. I could just design for one scenario, but I could also implement support for each platform.
Alex:
Here the danger is that mutually exclusive platforms mean that developers can't write content and then have it work across platforms.
Brandon:
If you build everything in WebGL then it just works, but it's a bad story for developers.
Ok, asking for a head nodding straw poll.
Is there anyone in the room concerned about restricting AR to a separate non-inline mode
Nell: Or needs to think about it more?

*** nobody rejects the proposal ***

Brandon:
We will definitely continue to have conversations about that on GitHub and

we won't push anything into the spec that anyone feels inappropriately boxed in by. So, let's keep this level of engagement online about this topic.
Nell:
Let's switch it to two interrelated topics:

- A topic about using 2D UI
- A topic about AR mode

Any concerns with that?
Ok, Brandon will you file a separate Issue?
Brandon:
Yes.
Nell:
Discussion around when to do which topics: hit testing security or break for lunch.
Is anyone too hungry to have this conversation?
Ok, let's switch gears to hit-testing security limitations.

Hit-testing security limitations

Max:
Summary of Issue: how do we prevent people from mapping the environment using hit-testing without the user allowing access to that data.
[See Issue #6 in hit-test repo - TFS]
We are suspicious that it's not possible to prevent the user from mapping the environment without hit-testing limitations.
Blair:
We think it might be necessary to even make it limited to when the use takes a user action.
Nell:
That was in a proposal I made a while ago, linking hit testing to a user's input device.
Max:
Regarding the reticle: if we limited to use action then we'd need a separate path for reticles linked to head or device position.

One concern is if we want to show a ghost version of an object it wouldn't be possible.
Alex:
Potentially, we could go down the path of having … [missed it]
Nell:
There is an option to query off of an input source but it has to be in the middle of a gesture or action.
John Pallett:
There's a lot of detail in the security repo and we could go through a lot of options. But, there is the fundamental question about whether we need to stop people hit-testing.
Nell:
I don't want pages to be able to hit-test to build a map of my environment. But, if it only works if the user has to jump through hoops by waving their arms around then it's less of an issue.
John:
The security team considers this the same as camera access. But again, if the polyfill can just ask for the camera and then the page can reconstruct the environment then why worry about hit-testing.
I'm not sure that limiting this to just user gesture will establish the goal of limiting security problems.
Alex:
This is a detail for a mode we haven't discussed yet.
Brandon:
One of my primary thoughts is that if we can't send out infinite arbitrary hit tests, even if we lack hit testing we provide 6dof testing so a malicious site could say "move your phone to your jewelry box" and the page could determine where it is.
Nell:
Someone could totally create a facebook game to make that happen.
Brandon:
If you want to get a good spatial mapping for malicious intent you'll need to tell the user to sweep the room.

John P:
I can summarize the threat vector document. [See security repo - TFS]
Access to personally IDing info.
Accessing credit card info by seeing the card.
Identifying the user by people in the room.
Room geometry to figure out same room used by two devices.
Geo location inferrance.
Profiling by size of house
User ergonomics like height
Historical info from previous AR sessions could be available to new origins

Possible solutions:
Limit precisions of derived data
Filter data by distance, frustrum, …
Clearing data
Throttling

Chris -
Camera access has the same problems and we don't have the same restrictions on those.
John:
Our security equates camera access with derived data access.
Also in the other direction, if you have derived data you can reconstruct photo like data.
Max:
Do we want to try to have a mode of AR that doesn't equate to camera access. If not, we can stop this conversation.
If so, we need to figure out whether hit-testing is the answer. How do we not nerf hit-testing so it's not useful.
Chris -
Is there a rate limiting approach that eliminates the threat?
Max -

Our security team says that they couldn’t say that hit-testing removes the threat.
??? -
At some point we’re just limiting the developers and the bad actors are just going to jump over the hurdles.
Alex:
If the AR mode is on and all of the derived data is available then there would be no limit to hit testing.
It seems like the question here is whether there is an AR-lite mode where there is some meaningful subset of data that we can expose but is meaningfully different in terms of threat profile.
John P:
If the site puts 3D candy crush to hit test the entire scene.
Max:
Could a page use a user gesture to fire off several different limited things?
Nell:
We’re talking about InputSources not generic user gesture events.
Max:
That includes a smartphone screen touch.
Brandon:
In the candy crush example, if there’s not limit to hit testing only “under” the touch then the page could hit test in any direction on each touch.
Nell:
I was proposing that the InputSource provides an XRRay and that what is used to hit test.
Max:
So, that would mean there was no API for creating an XRRay object.
Brandon:
Or if we didn’t need arbitrary rays, the objects themselves could have an invisible field that marked them as coming from an InputSource.
Gen:
We have two modes, in AR or not. It sounds like people want a different mode with limited access. I’m open to that, but we don’t have that now.

John:
The mechanics of using user gesture. The question is whether the perception of safety is actual safety. The two modes: nothing or everything, seems clear but anything else doesn't.
Max:
I'm hearing that in this limited mode there's some limit on how to generate rays based on InputSource provided XRRay. Is anyone object to that limitation? Do people feel that provides safety?
Alex:
It does feel like the candy crush thing would gather a bunch of data anyway. We could require that the reticle is drawn, but would that express to the user that something weird is going on?
Room: no….
Leonard:
People still fall for Nigerian 419. You can also assume a bit about what is not seen/looked at. And everything that is currently being said.
There are sites (or a collection of sites) that convert from geoposition (if a house) to the income of the owners.
If this level of concern and forethought happened in 1995, would there even be a Web.
Blair:
On the other hand, the bar is raised.
Chris:
As a developer, why wouldn't I always ask for open mode? And, how would I present to the user the difference?
As a user, only 30% of people know what AR is so this will sail right over their heads.
Rik:
People will bail out, scared by the dialog.
Max:
That's an incentive to have a less scary dialog.
Brandon:
I think that there are reasonable things that UAs can do to communicate it,

like telling the user that the page can't see your camera. Some platforms can decide that the user assumes the page has access to AR data, but other UAs might want to have a lesser mode by default and anything further requires the user to give access.
Iker:
It worries me that just one tap really limits the apps. It's fairly complicated to know how to give that permission so I'd like to see more use cases explored.
Max:
The basic use case of furniture placement is very complicated.
Iker:
This hit testing limit makes it very hard.
Kip:
If we're going to come up with some divides to data access it has to be understandable by users. So, perhaps we could solve it at a higher level, like the UA provides its own interface for marking a limited area that the page has access to. That could be more understandable.
We really do need more than one hit test to do things like position furniture.
John:
Re Brandon's point: The Chrome security team simply won't allow geometry access without user consent.
Blair:
Does that include "go into AR mode" or is it a separate thing?
John:
I should say "informed consent" but probably the gate to go into AR mode would include access to geometry.
Blair:
We're probably in the same boat. Any access to sensors will require consent.
Michael:
iOS's quick look goes from page into a quick view without sharing any information.

Nell:
I don’t want to mix that into WebXR. Certainly devs could choose a “just show a model” but WebXR is a step up with interactivity. A declarative mode is different than frame by frame hit-testing.
Michael:
Do we need a light mode?
Alex:
There must be something less than full but what is it? We haven’t got there.
Alexis:
Regarding the selection of a space, it would be hard to guess before the app starts what size to allow.
Kip:
Yes, the sofa case is hard, but for a table top game it would work really well.
Alex:
There could be two buttons: share entire environment, share table top.
Brandon:
Ooof, room setup is my favorite part of the Vive. [sarcasm]
Chris:
If I’m looking at AR-lite and AR-heavy and there’s a dialog for both then I’ll pick heavy since the bounce rate is the same for both.
Gen:
I’ll echo that.
Alex:
On the permissions side on our native app, they know they have a headset so they assume some access. So we have three tiers: head pose and “other bag of data” that includes spatial perception like meshing, and a third tier for camera access.
Nell:
????
Alex:
???

Max:
???
Alex:
It would be convenient for devs to not load a heavy mesh at the lite levels.
Jordan:
There are user media solutions but could we use layers to provide mesh on hololens and other data on other platforms. So, the user can see the shared data itself?
Blair:
It's interesting. I think we need lite mode. Installing an app is intentional, but following a link I don't know what I'll get. On the web if the only way to do AR or VR is to give it all way then it'll fail.
Max:
I think I can finish this discussion. I believe that we've reframed the discussion to "what is lite mode" and hit testing won't help that problem. The question is what is the underlying geometry and how much does the user control and understand that. None of that matters to the hit-test API, so we don't need to change the API surface to solve that underlying problem.
Does everyone agree?
John:
This will also be in the security explainer?
Brandon:
When will we see that document?
John:
First half of this week.
Nell:
So we can bring it to the upcoming meeting.
Max:
Are you comfortable?
Blair:
I have a whole other idea that at the beginning you make a whole scan and that's what the app uses.

Nell:
Let’s make an Issue to explore what AR-lite would need to include, then we can address the hit-testing API
Max:
Great, that’s excellent.

***** LUNCH (DELICIOUS) *****

XRFrameOfReference types (“stage”, “eye-level”, etc)

Process: What are the fundamental substrate coordinate systems that apps care about? Then we’ll figure out the API shape.

Alex: Issue 389 & 386
What it boils down to: What ***experience scale*** (see #389) is the app trying to target? E.g.

- Stage is different than Local in terms of calibration.
- Orientation-only is something where it’s just a 360-degree video player.

Question: Was “head-model” intended to do 360-degree videos w/ no movement?

Brandon: My understanding was that it would include neck modeling. However have heard (e.g. Oculus, others) that some always want a neck model. Implementations haven’t really done neck model yet, it gets zeroed out in some cases.

Alex: Apps w/ position seem to be either (1) Position gets zeroed out, or (2) App wants actual neck motion of the user. Is there anything in the middle where site doesn’t want neck updates.

Nell: Use case, if user is moving may want to do transitions between positions.

Alex: What is the most base use case where I'd need coordinates?

Nell: You do need view matrices. In what coordinate system should those matrices come from?

Alex: Could see a body-locked coordinate system; some hardware devices allow this. Would apps benefit from a body-locked 'orientation-only' coordinate position?

Max: In terms of turning into a view matrix may need to supply an offset matrix to get correct orientation information.

Alex: Underlying systems tend to give view matrices but in AR the real world is the real world so you can't zero it out without decomposing the view matrix.

Nell: Similar to conversations on how to surface pose with inline views, i.e. how can you progressively enhance from touchscreen/mouse on laptop to drive a view matrix, and what does it mean to bridge that - factoring in teleportation concepts - which don't exist in AR - into the poseless session question?

Blair/Brandon: Use cases in AR may include virtual worlds. For example, a tabletop with a holographic view above the table, the world may be shifting inside the space even though the real world is fixed (needs some more consideration).

Nell: Another is a top-down map where you drop into a location. Yup, it's relevant even in AR

Alex: Since it's hard to zero the view matrix, perhaps makes sense to have a frame of reference? And then headset orientation might be useful for UX purposes?

Nell: Is there interest in a substrate coordinate system - which itself may drift, you can't treat global positions as fixed?

Alex: For example, I have two plane anchors. They're far enough apart that they might drift independently. A system *could* treat each as a root, with things connected to them, and a view matrix for each. *OR:* Would you rather have one synthesized coordinate system on top of it?

Brandon: My understanding is that certain systems (e.g. Hololens?) where they system works hard to avoid parent coordinate systems. Has the thinking on that changed?

Alex: If you request coordinates directly they come in the default coordinate system. But in practice, Unity and other engines have a base coordinate system; what is the best thing the platform can do to give meaning to that coordinate system? To some degree the two ideas are orthogonal.

Nell: Any geometry you've found, any anchors, etc. - everything is independently tracked. So on a frame by frame basis you need to request anything rooted off a base node. Eventually you do need one set of view matrices to do drawing and then the ability to render all these fluctuating things. None of that changes!
The real question is whether there's something fundamental thing that we need to track separate of a global coordinate system.

Kip: Only in small projects do you have a single substrate coordinate system. Generally you're only doing FP32 and you have to redefine the coordinate system to avoid errors like skins going inside out, etc. So usually the origin is being redefined all the time

Max: Similar problems in AR - if I'm going from SF to LA then I'll need to change coordinate systems no matter what. Origin will be dynamic betwen

the locations. Doesn't mean we can't have a substrate but does mean we need something localized.

Alex: Every time you create a new frame of reference, you'll get zeroed out and reset and you can reason in the new one.

Kip: Fixed point is the most stable system for large-scale coordinate systems.

Blair: Argon had a local coordinate space with conversions into it from the global coordinate system.

Nell: In such systems having a differentiated frame of reference from a coordinate system really boils down to having a substrate within which you can get view matrices, etc. This drove designs where there were differentiating properties of devices (e.g. some only have world scale, others room scale).

Alex: We've been paving over these in XR with basic frame of references, e.g. room scale, floor location.

Nell: Is there still a need for a frame of reference type? Or can these just be coordinate system instances? Trying to avoid specific APIs, but is there a value to a specific type itself?

Brandon: I don't see value in a frame of reference type specifically. There was a hint to a frame of reference being an intrinsic property of a session; it feels like there'd need to be a way to instantiate (for use cases such as moving around the world); we definitely need something that allows us to reconfigure which of these we're using at any different point and reorient around ourselves. We already have a SetPose event, something from the UA saying that you've been reoriented would be good to make sure the application is staying in a good place.

Blair: When we did this with Argon we only used one coordinate system. The reason is that you can use the same call between two frames of reference to get a coordinate system which made the APIs very clean, and then you can use the anchors to differentiate between things… super useful.

Max: How do you define connection between underlying tracking world, and virtual world? These don't need to be the same thing. If there's a way to tease these apart that'd be good.

Nell: Not sure the underlying tracking system needs to be brought into this. It sounds like there's a canonical 'known of the world' and at every moment we already know the relationships of all the things. Concerned that an API too dependent on a substrate will be disconnected w/ what's actually happening.

Alex: A raw coordinate system might expose what's going on under the hood. May want a different coordinate system for things moving, there are benefits (e.g. precise near user, but doesn't promise to be anchored to one origin). Need to figure out what the promises are for each experience scale and use that.

Max: It's hard to separate this from anchors, because that's how you define relationship between virtual and real world. Need to be able to account for disjoint spaces, and to give that guarantee we're going to need to give app developer access to how the virtual spaces relate to tracking spaces. Part of the guarantee would be that if you use anchors, we'll keep things accurate for you.

Nell: Seems like anchors are just another special case of a coordinate system. What I don't want to lose track of is that there's a behavioral difference between systems. For example, the promise of an anchor is that

you can walk away and come back and it'll be there - on ARCore. But on Windows Mixed Reality with a 'stationary reference frame' might be different, it's stabilizing not anchoring; so things will be stable near you but not necessarily in the same location if you walk away and come back.

Alex: It's possible that a system will learn and improve local stabilization by adjusting things that are far away which may mean that far-away anchors will move on some systems.

Kip: "Stage bounds" seem very tied to VR, can we move some of this into anchors themselves instead of making them a special type of frame or reference? Not all systems will have this.

Alex: There are lots of types of anchors - plane, mesh, etc. - so a stage anchor might make sense. (Note: Trying to avoid calling things 'anchors')

Iker: Is the creation of these automatic?

Alex: The experience types are similar to what's done in Windows, though there are specific implementation guidelines for each experience type. But they're all sources of coordinate types. When you adjust you don't need a new one.

Brandon: The core question - is there a reason for the fundamental split between frame of reference and coordinate system? I don't see a beneficial thing, it's a conceptual difference and the more important aspect of it is how you interact - access and create - with those. None of that is facilitated by having a separate type. Better to differentiate how they are created and managed.

Nell: +1 and also, here's a curveball: How do we think about locatibility loss, e.g. getting view matrices. What's the expectation for developers who are making a world scale coordinate system, is it OK for it to drift and not be

trackable? What are we telling developers they should do when tracking gets lost, what do they fall back to?

Brandon: What do the various platforms do? E.g. SteamVR - if you lose tracking, you're kicked back to metaworld anyway. This is what the platform does for user safety, but clearly not the right answer for an AR headset to block your field of view. :) There's a certain level of platform intervention that we should expect which makes it difficult to have firm guidelines.
Other systems say 'please step back into a tracked area'
Phone AR - if you lose tracking your virtual objects just disappear

Alex: Oculus, PSP - if you tip out of the tracking error it'll drift but return back to you. But more generally if I ask for a transform between two coordinate systems and it isn't available, what should you get? Some engines will drag… others might cut you off… is that something we need to unify?

Alexis: Also multiplayer games - if one pauses, others still need to know what's going on.

Nell: Differentiated types is important. Right now you can get view matrix always. Differentiated objects might help, for example if you say 'you can use any coordinate system you want to get your view matrix' then is there a differentiation between a system tracking failure, and a particular coordinate system not being found? How would we communicate that to developers? Not sure how to square that problem if we have a single type.

Brandon: That feels more content aligned than part of specifics for coordinate systems. If I'm using a bunch of coordinate systems to pin things into the world, and one of the disappears, I'm probably structured where I'm looping through those objects and trying to render them; if I'm missing one, it's probably OK for me to just skip it. However if I'm a VR app

and everything is anchored to stage coordinate system, then it's more problematic… maybe can do this at the API level? But it seems like that sound be an app choice.

Nell: Can it be an arbitrary choice, instead of something concrete like a node, maybe something more abstract?

Alex: Is there anything in the various platforms which prevents you from saying "Where is the view matrix relative to this particular anchor?"

Max: View matrix for the camera is what you use in your native engine to render. Everything else is pinned on anchors, those are the connection points between virtual world and underlying trackables. As long as you're updating and you render from the device, then the only view matrix is the one you're viewing.

Alex: because we're trying to introduce multiple types of coordinate systems, are we setting the apps up to fail

Brandon: to clarify: We have an API shape today that takes just XRCoordinate system. Should that change?

Nell: not quite what we're asking

Brandon: there are two models of rendering things.
1. I have a base coord system, if I have anchored objects, I get their position relative to my baseline, I have a single camera point, I render from that viewpoint
2. Webxr right now has a diff model, instead of having one global coord system, I ask for the place of each anchor relative to camera and render them. There may be a benefit to doing it this way.

So one of the implicit question is whether that model of things is

something we care about, something we want to support… or is everyone going to go with a Unity model?

Nell: The idea that there's one substrate that everyone *has* to handle is what I'm questioning. What is the basic experience I'm trying to build? The initial intent behind frame of reference was to capture the implicit substrate that gets passed in. If we're saying there's a differentiation between room scale and world scale, then when developers get started they are picking one or another… features may depend on the choice (e.g. anchors for world scale)... whatever path we give developers, how do we support that path? There is something intrinsic to tracking getting lost both at world and room scale.

Kip: The reason people try to conform to a single coordinate system in Unity/Unreal, you can mark things as STATIC. That's for performance reasons, but it also means that these objects don't move *relative* to each other; the pre-computed information is guaranteed to be stable. Similarly if we need to know that when we batch things, if we know what's stable relative to other objects, then it's OK. *Key takeaway: It's important for some engines to know where some objects are permanently located relative to each other, you get buckets of things that are anchored separately but they're adjusted together.*

Brandon: Everything that's statically batched together should be assigned to a single coordinate system. But those are current game systems, we have raytracing coming.

Max: I'd conceptualized frame of reference as something you could request, but if we're abstracting anchors as a way to connect virtual and real world objects, the underlying system might need to reframe the world into a system that makes sense. If so then the underlying system would need to update in a way that makes sense and that behavior would be impacted by which experience (World, Seated) is being used.

Alex: Just like you have to update dynamic things, it's on you (the developer) to re-find things and adjust.

Max: This is why when you request an anchor you need to request it within a coordinate system.

Iker: The main thing is, do we agree that we need the concept of a coordinate system to specify different scenarios and semantics like anchors, inputs, etc.? Are frames of references just instances of coordinate systems?

Nell: One approach - everything is a single coordinate system and then there's a union type to get matrices? But is there something more fundamental about tracking and loss?

Alex: Probably don't want to torque the API to prevent bad solutions.

Brandon: Right now we're getting view matrices from a coordinate system. We're expecting that to be in a particular type of stage. But if I simply dropped an anchor where I first stand, and every frame resolves to the original coordinate system?

Nell: Because aiming for better-than-least-bad where you can create stability around what the user sees; the original origin has precision and stability effects. I'm grappling with how to frame my thinking around… is there something intrinsic to what we've called a 'tracking system' or 'substrate' - or a property to a world coordinate system that behaves in its own way?

Brandon: It sounds like if we want to create success for developers we should guide them to one of the user-stable things, and we don't allow

developers to try to render with an anchor as their source of stability (lots of ways to skin the API cat)

Iker: Are we preventing use cases by doing this?

Nell: There's nothing intrinsic from people from doing everything if there's a world coordinate system and a plane anchor that stays stable as you walk around. But is there something intrinsic? We still need a view matrics?

Alex: I'm less worried that people will go out of their way to make an anchor and do a lot of stuff… Stage and Local are both stabilized near the user, it's the World frame of reference that needs anchors.

<< BREAK… oh thank goodness my hands hurt from typing so fast >>

Brandon: Trying to understand differences between WORLD and LOCAL experiences; LOCAL and STAGE seem like they can be combined.
So for WORLD/LOCAL, seems like they can be roughly interchanged in terms of content, i.e. you won't have origins popping up as you go, but the big difference is that if you turn around and go back to the origin, how stable should that be?
e.g. you're in a racing game, you get out of vehicle, you walk away for a bit… getting a drink in the real world… when you come back, the LOCAL coordinate system should be very concerned about making sure the chair is where it was. The WORLD coordinate system is more interested in making sure the environment remains stable relative to itself.
However in this scenario, if I *only* had a WORLD coordinate system then the correct pattern would be to drop an anchor at the cockpit, then the anchoring system would naturally re-align the cockpit with my chair. From that perspective, could probably think of LOCAL as a specific anchoring within a WORLD.

Alex: Right. And EYE LEVEL would be effectively creating a world system then dropping an anchor.

Nell: How does this relate to knowledge of world frame? Is there a model for plane detection? Does it apply to world?

Alex: This is a core difference between ‘seated’ vs. ‘world’ - i.e. if you’re seated in a mech vs. a car it’s a 20 foot difference when you go to the ground, but in a real-world environment there’s actually a floor.

Max: It’s possible that in world you’d get an update to anchors which update them by 1km or more

Brandon: This is where there’s a content decision; a lot of content doesn’t make sense outside of the room. But large-scale anchor updates could result in massive jumps (either in position of the user or objects that aren’t anchored).

Alex: There is value in things that are highly dynamic being ‘loose’ (not anchored) so we can’t assume that an app will anchor everything.

Nell: And, anchors have performance hits; there will need to be a system for anchor batching. Need to be practical, something you can’t practically localize to what you can see is not going to work.

Alex: May be other batch/fetch/pose APIs before we get there.

Nell: May be something we can do - like a world reset or batch API - that we have to solve it eventually.

Alex: Yes - customers saying 1km games are inaccurate when you’re 5mi away then that’s a possibility.

Max: I can write up a summary of this.

Kip: A lot of this can be solved by the content. Just need to make sure we don't make any assumptions.
I.e. If the site recognizes that they're crossing fp32 thresholds then they can drop new anchors.
What we might need is best practices for lerping between coordinate systems; e.g. a cannonball moving long distances may need to build in some correction as it moves between coordinate systems.
Past a certain distance you don't actually need anchors; it's only local.

Chris: Conversation is framed as device as the authority. In a lot of cases the server is the authority (not the device)

Nell: Hoping we can decide whether we need to think **experience** first, and what are the **unique** first experiences that we need, then how that maps to hardware?

Chris: There are clients who want to do both near-field and far-field AR (far-field being tagging things in the world)

Alex: "World" might better be called "Geo-scale" but we can pick the right term

Nell: Framing for discussion of experiences:

- What features are needed for each type of experience?
- Do devices support all types of experiences?
- If a device doesn't, should we fail? Is there progressive fallback?
- Do we need to delineate between experience types?

Jordan: Not sure difference between local and stage

Arteme: Room - you want UX rendered consistently around you, meaning it moves with you as you move to different rooms. Stage - this is harder to do.

Brandon: There's no question that to some degree experiences are 'user-convenient' - you can conceive of a world where you get a local coordinate system, then an offset to the center of the world (if I know one). This is what VR does and developers hate it because there's more matrix math. But a library could fairly easily abstract this away.

Brandon: Favors experience types with fairly clean mapping to use cases (e.g. 360 video = headset orientation)

Alex: If you're 20m away from a Stage and trying to stabilize using a distant location as my origin… you'll get bad results. Whereas if you've picked something with a local origin then it'll work better.

Brandon: There's a flip side to user convenience, i.e. to what degree do we want this system to work as a feature detection system? Do we allow the user to ask 'do you support room scale'? Seems best if we can provide developers with modalities as much as possible, with hints afterwards as to whether things are emulated or not.

Iker: Do we agree that we need several ways of specifying? If so what are the specifics? (Clarification - a set of categories of substrate types)

Nell: Concerns that if we have too many choices then it'll hurt developers.

Kip: Elderly users, learned from a 100-year-old using VR in a swivel chair. ***Categories may have an added benefit to implement accessibility features, for example, allowing a user to choose height even if they're wheelchair bound (allowing them to pick things up even with their height difference)***

John: Feature detection - early AR users definitely want to know whether it's real-world scale (e.g. 6DOF) and not emulated.

Brandon: Pretty much all of the uses we've talked about here have an implicit association to a real-world scale; if you get 1 unit in XR it's 1m in the real world; that breaks down in some cases where motion is restricted, but even then, the head motion should return 1-to-1.

Alex: So if a device can't provide real-world scale, it wouldn't provide position; even in 3DOF if we provide coordinates, neck modeling and so forth should respect that scale; model presentation should be to scale.

Blair: Some of the words we're using have bloated meaning, hard to tell what 'local' and 'stage' mean (Nell: Yup, ideally the terms would be immediately grok'd).

Alexis: Just because you have a 6DOF device you may not want it, for example if you're setting on a train you probably don't want your location to track the train's location. User ability to pick a mode is important in this case.

Brandon: Not sure that apps should control this, similar to accessibility features they may not add them.

Nell: Are the bounded ones all VR and AR is world scale? Inside-out tracking on VR headsets complicates things.

John: Is there a reason someone would do a bounded AR mode?

Max: If all content is anchored, then you can abstract over a lot of the stabilization questions. A lot of this is about simplifying things; if we say 'things are always stable around the device' and then they can use device transforms.

Brandon: In a lot of cases if I have world scale, but I'm using a Vive, the intrinsic properties of keeping things stable around you don't matter so much; it's functionally equivalent to local anyway. I am super-hesitant to say we should make anchors a fundamental primitive right now, but it's not hard to imagine "world+anchor=local" in the near future and for a lot of hardware it'd be functionally equivalent anyway. It's only when we get optically tracked headsets where it's a problem anyway. If we care about those use cases, then "World+Anchor=My behavior" is easier to explain than more environments.

On the others, "Orientation-only" clear cut use cases; "Stage=I know where the floor is" seems valuable too. The others where it's just "Around my head" seem like they bundle into one.

Nell: The concern I have is that there's different behavior on devices that support world scale, and devices that don't. There's a potential path to collapse to two; plus 3DOF devices that don't support world scale at all but could do a standing scale experience? If the developer designs an experience that you have to walk around, and you can't walk around it, then that won't work. Where do we need the splits to be?

Brandon: Still fundamental question - is this a mechanism we want to use for feature detection. As a hypothetical, if you ask for world scale, and get a pose, then it might not be based on sensor data; tricky to communicate in a consistent way (hardware limitations) but the point is that it could be separated from a 'desired' coordinate system.

Max: Ideally we can tease apart a few features that don't actually need to be coupled.

Alex: Do we want to start you at ‘world’ and then developers do extra work to get to ‘standing’ scale? Feels like you should know ahead of time whether your users are sitting or standing.

Nell: How does that play out on a 3DOF device that can’t create anchors?

Max: No-op anchors could work when the user agent knows that the device is 3DOF.

Alex: Right, since there’s no movement the anchor will always remain put.

Nell: But on an inside-out VR system you could create an anchor and would you want it to be emulated or not? No-op would be ideal here?

Alex: Same thing with a backpack and a Hololens system. (Developers would want anchors to work in this case)

Brandon: The biggest risk we’d run is developers who work on an Oculus Go, then put on a system that has more world-scale tracking, it might not be perfect… but how any people are going to do that? If you design your experience for sitting down, then people walk around - users probably don’t expect it to be perfect (example: Roller coaster experience, if you have the ability to get up and leave the roller coaster, it’s probably not going to work)

Alex: Unity does this:

- Stationary World - local world
- Room Scale - bounds
- Disable Position - a bool that means you can’t move around

Summary: There is positive feeling around a coordinate system abstraction; there is some value to having different experiences reviewed, but not clear whether they need to be explicit or not; may have value by avoiding a series of matrix transforms to get what we want.

Today we have 3 frames of reference - stage, eye-level, head-model; orientation-only sounds useful; moving forward local could probably be squished together with stage; also positive feelings about 'world'.

As an API we're already heading in a direction that feels good; need to figure out whether we can reconfigure the experiences and define them more clearly in the spec.

Need issues:

1. What don't we understand, what do we need to close out the topic
2. Does XRFrameOfReference need to continue to exist
    a. Base this on Max' PR
3. Feature detection? E.g.
    a. 3DOF vs 6DOF?
    b. Can I do anchors/hit tests?
    c. Do I have a floor?
    d. Do I have a bounded environment?

<<END OF TOPIC>>

Lightning talk vote tallying
Magic Leap overview of pull-out: 9
Automatic occlusion with the environment: 9
Computer Vision integration: 8
Geo-alignment: 6
Lighting Estimation: 5
Anchors as they related to hit testing: 5
Composed AR stream readouts (e.g. streams, screenshots): 0
Choosing a camera for video-mixed AR: 0
Poseless sessions (not strictly an AR topic): 0
Other RWU topics worth opening a repo for: 2

Computer Vision

Blair - this is time-synchronized access to camera frames related to poses and other sensors. Options:

1. The WebXR Device API already talks to all the necessary sensors/pieces so could do this.
2. Alternatively RTC community (esp. Intel) have already proposed a lot of this, and all that's needed in synchronization. In this mode XR could expose things to RTC.

Differences:

- WebRTC does not give you true device video frames (it can be some other remapping); there have been some proposals to access that data more directly. Similarly cannot access camera location relative to device pose, or device pose that is frame-locked; would need to expose intrinsics for this. But on the upside, MediaDepth Capture extension adds RGBD data.

Comments from issues:

- Would like more detail before repo gets created, including options to compare and contrast
- Seems important to polyfill AR features that aren't implemented in some browsers
- Intel and Mozilla (Blair) will coordinate to get more detail around design

Geo Alignment

Blair - there is an explainer and a draft IDL. The question is how to expose it. We already have geo sensors on the web. But ARCore and ARKit can align the coordinate system with the world automatically. In theory alignment alone doesn't give anything away without location. Could do geo-alignment with Javascript, but the native APIs do a better job as they have access to more precise/timely data including magnetic field skewing. Assuming underlying coordinate system can change, the fact that it's rotating a bit as it gets aligned with the world doesn't change things very

much. Proposal would be to have a way to request it. Then on devices that don't have a compass (e.g. Hololens, Magic Leap) they can say 'no'.
Once you have geo-alignment, doing alignment with geospatial is pretty easy; you can get a report every 1s or so which gives a GPS location and an XR alignment, then you can align them and display geospatial stuff well.

John: Extra permission for GPS data? Want to include this or would it be a different prompt?

Blair: Could alternatively add geolocation to the API and combine into the session consent.

Nell: There's a general question about how to combine permissions that aren't natively supported in XR within a session-based prompt.

Nell: Does this require a world coordinate system, and does this mean every anchor is geo-aligned?

Brandon: If you create a series of local coordinate systems with geo-alignment you may be able to infer latitude. [John to add to privacy/security explainer]

Automatic Occlusion
Nell: If there's an AR lite mode, what would it mean to allow a web developer to get world occlusion without getting mesh data to do the occlusion. Is there a way to explicitly set up a depth buffer.

1. Do we need a way for developers to specify the depth buffer, i.e. an API for supplying depth data? Could be simple like "Use this property of the frame buffer for depth data" and can this be used for the final composition?
2. How would alpha channel information be used?

Nell:

- User agent would be doing the work (not user space)
- Alpha is important because fully opaque objects would occlude the real world
- Not sure how semi-transparent real world objects would look (?)
    - Kip: Please file an issue on this
    - Everyone: Ready about depth peeling

Kip: Other browser use cases might be enabled, e.g. more than one AR thing being presented without the possibility of cross-origin communication.

Nell: I am not a proponent of x-origin rendering in the same scene; would need UA mediation in this case. Could re-evaluate, but this wouldn't be easy to bootstrap. Out of scope for first revision of this API.

Helio / Magic Leap

- Wanted to add 3D content to pages fairly easily. Key goals:
    - Easy
    - Powerful
    - Extensible
    - Open (possibly create a spec. later?)
- There's a library, can use like this:
    - <ml-model style="width: 300px; height:300px;" src="Leaper/RedLeaper.fbx" extractable="true"></ml-model>
- General web platform:
    - Prismatic
        - 3D models, placeable content
        - Standardizing on glTF (only doing FBX for now for legacy reasons)
        - Added bonus features: Transparency, unbounded volumes ("I want a giant f$#@ing whale!"), 3D favicons
    - Prismatic *provides* <ml-model> - it's a deeper level set of APIs
- Links:

- creator.magicleap.com/learn/guides/helio
- magicleaphelio.com/devsamples
- creator.magicleap.com/learn/guides/design-principles

Underlying OS allows boxes created in 3D space
New API allows you to create “prisms” - more boxes - *near* the browser window, can’t be arbitrary. They are only placed relative to browser window. This allows scrolling (object disappears when off page).
With each prism you can build a scene graph with transform, geometry nodes
Interface is simplified so that one box contains one quad; underlying OS API allows you to do more but Prism doesn’t let you.

Extractable attribute does **not** support CSS - once pulled into user’s environment it’s effectively turned off. CSS is used to size element *on the page* also to set depth; when you extract you get a **snapshot** of what it looked like on the page. *Note: They’re considering ways to change things in the real world, but there’s not a great solution right now for this.*

Pulling object out of the page will expand it back and away from the user, but there’s no indication of how big it’ll be. [e.g. pulling out a huge f$#&ing whale may mean that it ends up swimming in your neighbors’ yard]