Games CG meeting Notes

Mar 25, 2025 - LoCo CG proposal

Agenda

See calendar entry and check the draft LoCo CG charter.

Attendance

- François Daoust
- Tom Greenaway
- Dominique Hazael-Massieux
- Noël Meudec
- Andrzrej Mazur
- Gaëtan Faramaz
- Florin Ciornei
- Erik Dubbelboer
- Andrew Gildfind
- Koen Bollen
- Adam Scott

Minutes

- Tom: We discussed in past meetings the possibility to bring some kind of common format or API to bring simplicity to how a game developer can distribute a game across portals. How the game loads initially, how it interacts with ads platforms. Goal is for different

stakeholders to work together on a common set of functionalities that all portals currently do in their own way. We discussed that here, followed up with chairs and François who provided input from a W3C perspective. The idea would be to create a new Community Group. We'd like to raise awareness about this idea, get feedback on the draft charter.

- Tom: This group has quite a few members. We want to be able to continue to host discussions as we've done so far. Spin off group would focus on the API itself. Please check the draft charter, which highlights the scope of the work and the out of scope of the work.
- Erik: How did you come with the out of scope list?
- Tom: We just tried to come up with a minimal set of things, but if you have strong opinions about some of these, we can discuss.
- Erik: I would love to see storage management at some point.
- Tom: Yes, that would be my top one as well, the others are more complicated than they seem.
- Andrew: In my world, these things are separated because they touch on integration with the distribution platform and then Gamesnacks framework.
- Tom: Right. Perhaps the storage management could still be considered.
- Dom: Commenting on the scope, the reason to limit the scope is to make sure that organizations feel ok to join the group, especially for topics that are patent sensitive (such as payments). For things that are not as sensitive, such as a get/set method, you could leave it in scope and consider it low priority. It's always a pain to extend a charter, the argument I would use is more to look at how it might affect participation.
- Tom: Makes sense.
- Tom: Looking at what's in scope. Namespacing is for integration within portals. Bidirectional versioning is to make sure that updates can be detected and propagated. Need a way to identify the main DOM element that contains the game. The core meat comes next with the hooks to trigger events and register callbacks. Then ways to manage ads. And then hooks for audio would be nice. Do people feel that some of them do not fit?
- Erik: I'm not sure what the DOM element would be in practice. Iframe?
- Tom: That's also the purpose of creating a CG, to figure these details out.
- Andrew: Context is that it may be hard to know if a game is running on a page. If we know there's a game, by having someone somewhere declaring that this is a game, then we can built on that.
- Tom: On the call today, I see Poki, Crazy Games, GameSnacks. About 3 people represented.
- Erik: Also Adam from Godot. I guess this is relevant for game engines as well.
- Tom: How do people feel about the idea?
- Koen: Happy to join as advocate from a game developer perspective.
- Adam: In an ideal world, it would be interesting for engines to package a game once. Common events that come with games on web sites. What may be missing is how we package the game. Do we want a common way to package the game? Or is it too difficult and we only rely on the SDK/API approach? Having metadata to be able to declare the game, what it is, etc.

- Tom: I think many game portals expect some kind of zip files with structured folders. Does that process deviate quite a bit between different portals? We could try to standardize that, but the core goal is to try to simplify things that are different across portals.
- Koen: It seems very simple, but there's a lot of differences. Is the entry point in the root directory or not? Etc. It could benefit from having a standardized form, some sort of manifest.
- Tom: The Open Mini Games proposal was kind of around that. I'm not against that. But more creating new features than solving a pain. We can make guidelines perhaps in the API to recommend a root index.html file.
- Koen: There's also the encoding.
- Andrew: Packaging is only interesting if the game is very static. Open Mini Games is very good for the runtime model. Another use case is knowing the static components for the game for portals. I'm not sure standardizing that part is useful, games will likely be generating content on the fly in the future.
- Tom: Who's the target audience we're trying to make things simpler for? For Open Mini Games, it was more around discoverability. Trying to take a step back, first variation I see is on events and API, with bugs creeping from these variations, I'm thinking that's the right initial scope.
- Dom: The question to ask about the scope is what is the set of features that can demonstrate that this can work. Once that's done, then expanding will be easier as people will then realize that "it works". Is the set of features identified sufficient to solve a real problem? Does it solve enough that people would adopt it?
- Andrew: If you expand scope, it will have to be layered. With an AdSense hat on, I'm really trying to solve the problem of ads. If you add storage, it would have to be optional, because AdSense does not want to have to do anything with your data. Some of these features are disjoint from a platform perspective. In GameSnack, there are building blocks, some of the functions we cannot expose.
- Tom: Another question to consider. 3 portals here. At the moment, with the current draft, would you be interested in building this proposal?
- Erik: I would definitely be interested. I cannot commit on behalf of Poki, but would personally be interested.
- Tom: When you look at it, do you see something that might be problematic or missing? On top of your head?
- Erik: Not necessarily. From a platform perspective, standardization is something we need, but it would be better if game engines all implemented the same thing.
- Adam: I agree with what Erik is saying here. I'm still skeptical about the implementations and the limits that we are already setting. How to export to specific platforms. If there were a common way to distribute games to portals, I think we should concentrate on that. You declare when you load, start, render ads, then you decide what the export looks like. If you're an indie game maker, you may have your own engine implementing the API and then you give that to a game portal and things will work.
- Tom: To clarify, what you say is the purpose and goal from my perspective. I don't want portals to communicate between each other. I want portals to accept a common API and

set of events used in games. Having consensus between the portals that this thing would have value and that they'd be ok with implementing and maintaining it. Maybe something to add to the charter is ways to extend with API methods.

- Gaëtan: As a platform, we'd be interested in participating in that. It would be easier for devs to implement the stuff that we rely on. For example, it's hard to draw a line between resources that need to load when the game starts vs. resources that get streamed. You need a firstload event which is something that we ask games to implement. As a platform, it would be really great to have such features that declare the events.
- Tom: Some sort of manifest then?
- Gaëtan: Yes, events, play/pause stuff, standardizing this would help.
- Tom: We need to come to an agreement on the scope.
- Gaëtan: Apart from the DOM element, everything makes sense to me.
- Tom: Do people feel strongly about adding things? I'm feeling storage management is low hanging fruit. Also index.html as entry point could be in the initial scope. [missed].
- Andrew: Ideally, we want everything standardized. Current LoCo proposal is focused on runtime. That's good. Then there's integration with platform, high scores and the like… runtime, vs. how you set things up.
- Koen: Storage management is good to add. Otherwise, you don't introduce something new that would favor adoption by developers. That's a struggle for many of them, and would provide incentive.
- Adam: Storage management, I'm not that keen on tackling this first. What would be possible to tell developers? Not a browser API. Hooks to manage ad monetization may be a lot to handle at first pass. Do we want the users to get feedback from the host? Currently, it could work with the normal triggers. Maybe adstarted and adended. Otherwise, level change. Not necessarily related to ad monetization, more "can I continue the game?". I may be wrong.
- Erik: I think all the platforms have some form of monetization in their SDK. One of the issues is when you say to the platform "you can play an ad now" but then you have no idea on whether an ad gets shown. Standardizing that would be great.
- Andrew: A lof ot the motivation in the original pitch is that ad management can be very complicated. We tried to remove the complexity in GameSnacks. The purpose of this API is to make it as easy as possible. The host is then responsible for the complexity. "Please, this is an opportunity to place an ad", and then that's it. The idea is to remove all the logic out of the game into the surrounding environment. Sometimes, it's going to be a simple environment, or a full-fledged one.
- Tom: My vision for this is that game developers need to be able to make money. The ad monetization piece is crucial for the happiness of the game developer.
- Adam: I'm not against the hook for ad monetization. But the description goes much further than what we're discussing here. The essence of my comment was more about that.
- Andrew: That's a great point. We wouldn't want testing for ad availability in practice, that's pushing too much logic in games. The design principle that I try to push is the minimal set of hooks for ad interstition. The absolute minimum to put in the game. And then the ad network can handle the complexity.

- Tom: Can we trim this part, then?
- Erik: One of the things that could be added is how could the platform keep their functionality that's not in scope of this work? Should there be extension points?
- Tom: Some mechanism to do that would be good. Either part of the API, or the portal just exposes their own SDK object. That's something to tackle in this new CG indeed.
- Andrew: We try to align semantically internally between different platforms. If there were a W3C standard, this would help me with convergence. Back in the day, OpenGL was good at handling extensions. Industry practice converges at some point and that's usually a good sign. It's a bit like operating systems. Being able to launch a command, we don't even have that. We should be pragmatic and start with a first chunk. Then we can do it again.
- Tom: I made suggested changes in the draft charter. May I accept them? Plan for extensibility and simplification of ad monetization stuff. Storage management also. What about layering?
- Dom: It seems to me that this could be discussed in the CG itself, no need to formalize it in the charter, I think.
- Tom: Not sure about the DOM element. I guess there's no risk leaving it in here.
- Dom: That's correct.
- Tom: And it sounds like people here would be happy to participate.
- Francois: If people have further comments, please try to come up with concrete text for the draft charter. Leave the document open for review for about a week.
- Tom: Thanks, everyone!

Nov 26, 2024

Agenda

- GameSnacks
- Open discussion around monetisation, discovery and distribution

Attendance

- Andrew Gildfind, PM For GameSnacks and Adsense for Games at Google
- Erik Dubbelboer from Poki
- Noel Meudec from Meta
- Florin from independent studio
- Beckett LeClair from ??? (digital rights organisation focusing on young people)
- Richard Davey from Phaser
- Andrzej Mazur from Enclave Games / js13kGames
- Raf founder of Crazy Games
- Koen
- Bjorn
- Tom Greenaway from Google AI / Games DevRel

Gamesnacks

- Andrew introduction: ads for web games and also distribution for games
- Web games reminds Andrew of the 90's and likes that. Graphics and simplicity of gameplay.
- Background on web games at Google
- Monetisation:
    - AdSense
    - Ad Manager
- Distribution:
    - Gamesnacks
    - YouTube Playables
- WeChat is more mature than us
- Today's focus will be on GameSnacks
- Defining HTML5 games (not web specifically)
    - Good for low end devices
    - Fragmented and complex
    - A post app store ecosystem
- In the space, we see a transition to a streaming model for games
- Last year or two, lots of competition in the app store space. Microsoft making moves around this area.

- Explosion in distribution channels
    - YT playables
    - Netflix
    - Mobile OEMs
    - Microsoft Edge as well
- Authoring with AI is impacting this area of development
    - And it's accelerating
    - In China, maybe they're 3 to 5 years ahead of us. Web instant games revenue is picking up.
- Video demo of Microsoft Edge embedding games
- Explanation of GameSnacks and the ad model
- Some challenges:
    - Transition to streaming
    - Big catalogs of games
    - Lack of standardization → fragmentation
    - Can we standardize the distribution
- Content needs to be designed a bit around the ad and monetisation model
-

Questions:

- Who here has used the GameSnacks API?
- OMG. would that be valuable to bring this back into the conversation?
- Who here feels that Revshare is a complicated problem?

Koen: like the idea of how easy it would be if it was standardized

- Ajag: it's not exported via the schema at the moment

Erik: like the idea of a standardized SDK. There are parts already in common.

- But feels a bit like xkcd comic: standards + 1
- Each platform has its own functionality
- Standardized API would still have Q/A problems
- Maybe for one platform the triggers would be different. Perhaps touch on one versus something else.
- Ajag: some of the differences would be policy based as well?
- Erik: yes, and that's probably harder.

Raf: fully agree with Erik, it's very hard to get right.

- There's a lot of nuances.
- Difficult to capture in an 'easy spec'
- The IMA is very complex
- Developers just want "show ads"
- Platforms love the control it gives over the UX
- Reliability and robustness

- There are so many bad video ads out there. If no progress is being made, you want to be able to kill it. Etc.
- If we abstract away from the IMA SDK, some of those nuances might be lost.
- Not sure how all this works with… Header bidding and specifically video header bidding

Ajag: that makes total sense. But if games gave you a distributor API
- E.g. stop sound, start sound.
- Not advocating against advanced monetisation
- Not aiming to couple it heavily for GS and ads. We have designed it to be easily decoupled.

Raf: GamePush is an example of an aggregating library

Anzrej: as a developer, all the platforms do it differently and it's a huge pain
- Standardizing and a schema sound great.
- Let's create an OMG portal and promote developers who do this too

"Making a single game isn't too hard but making a business that scales with more games is difficult"

Erik: bizdev teams might be against opening up heavily for platforms that are already succeeding because then developers might too easily migrate away from their platforms.

Andrzej:
- At the peak I had my game added to 10-15 portals with different SDKs and 5-10 of them had fluctuations to their SDKs more frequently than they should

Richard Davey:
- They have 30 - 40 folders for a single game to deal with all the different SDKs/platforms.
- It happens frequently.

Ajag: the pain of all this prevents the web from growing into something that can rival app store games

Ajag: health of the web would be better if this was less fragile for web games.

April 30th, 2024

Agenda

https://www.w3.org/events/meetings/d6e36359-b4c8-49e6-a849-2c7f1a027141/

Attendance

- Erik Dubbelboer
- François Daoust
- Björn Ritzl
- Andrzej Mazur
- Koen Bollen
- Adam Scott
- Gaëtan Faramaz
- Rafael
- Richard Davey
- Noël Meudec
- Russell Kay

GDC

See the recorded presentation and transcript.

- GDConf
- Erik: Tech lead at Poki. Gib GDC for Poki this year because that was the first time we actually had a booth. Main takeaway is that last year people were wondering what the web was for games, this year much more interest. App store, Steam, they're all getting very saturated. The web is making a "come-back" in that regard, as a new way to distribute games. Also people don't want to install games on mobiles anymore, so lots of interest for web games.
- Noel: I was happily surprised by Poki's booth. I tried to convince people to develop Instant Games as well, of course :) I also tried to get people interested in VR but that's an entirely different level of investment. It's true that people don't want to install games anymore. Tired of ad "spam". The frictionless experience of web games is a key feature. Also the login experience. Much better to be a in a logged in state once and for all.
- Adam: I was at the Godot booth. Fabio presented Godot last time. In the meantime, we switched role, I took over as web lead. It was interesting to be at GDC to get that perspective. Astonished by how the web has grown for games. Before, I was looking at the web as another way to export the game, with some caveats (e.g., thread support), with platforms such as Poki that allowed you to play with that frictionless experience.

Now, there's such a demand for games that play as soon as possible. It blew up my mind. Looking forward to making the Godot engine better for the web.

- Björn: There representing Defold, invited as partners to Poki booth, with a focus on integration with Poki. One thing that surprised me was the number of people coming to the booth and not knowing that the web was still a thing for games. Still think about it from a "Flash" old time perspective. A lot of people still don't realize that playing games on the web and on the phone can provide a very good interface. Example of a person who came, sat to give a web game a try, and stayed 15mn playing the game. Interested on whether people exchanged with Discord people?
- Richard: Discord opened up things a little bit. You can join an activity, and start an iframe that can run a game. Not straightforward, but hooked into the social voice features of Discord. I think it's going to be massive. We've done two tests. Some restrictions for now (20 people max) but temporary for sure. Once lifted, things will be huge, allowing to create games with massive captive audience.
- Erik: Wondering about Rafael, involved in big talk with Unity.
- Rafael: 8th GDC for me. A lot of it happens outside of the expo show. We caught up with partners that bring games to the web. A lot of big companies expressed interest in web. We talk to everyone as game platform: publishers, service providers, developers, etc. Personal highlight: more people wanted to talk to me than to Unity at the end of the talk! Show was good overall. It took me a month to follow up with everybody. A lot of movement, and a lot of content. The moment that we have the same quality of content as on mobile and other platforms, things become a no-brainer for game providers: it's easier to play games on the web!
- Erik: 25 game developers who develop games on Poki were present at the booth. I think it was nice to have them actually present their experience. I can give a "sales pitch", but it's much better to have actual people come and share practical experience, how much money they make, etc.
- Erik: Unity layed off a number of people last year, and I note that the web team was totally unaffected. This is interesting to note. May be due to Meta partnering with Unity on web features.
- Noel: I talked with a bunch of French developers. They looked happy with the money they were making as indie developers. Apparently, enough to run their studio. Very pleased to hear such stories.
- Adam: Second year for Godot at GDC, I think. First time for me. Repeating myself, this really changed my perspective on web games. Seeing the Poki booth. I think even LinkedIn wanted to see games. Once you have web platform at your disposal, you can embed anything, and you can load an infinite number of games in a frictionless, instantaneous way. This made me realize that there are lots of people that are willing to play web games. In Godot, we're using software mixin, but that's limited in single thread. When I created the threadless version of Godot for the web, I went through that. I'm now trying to integrate more directly, e.g., Web Audio. I'm super happy to do that. And I'm feeling the pressure: people want this. A few years ago, you could "play" the game, but that was more a demo. You then needed to download the game. That's no longer the approach that people want to follow.

- Noel: I stopped by the Godot booth as well and had the chance to talk with Fabio. Fabio was super excited to show me the animated robot face that was featured on the booth and that you could program directly through a computer.
- Erik: So many people come to the booth to get merch. We had socks, etc. We had to hide them because people were rushing for them.
- Adam: Same. We had a lot of merch too.
- Noel: Reminds me of past experiences where people would just ask for a pen without even asking a question.
- Erik: I thought crypto and NFT would be gone, replaced by generative AI, but I was suprised that they are still around.
- Rafael: Crypto is still hype, indeed. Sometimes, different players have different mindsets.
- Andrzej: There is a subset of regular web 2.0 developers trying to go into web3. In the jams I organize, 10 games out of 200 would typically attempt to use web3 features. Some scam, but some interesting use games. I hope there will be more focusing on the technologies themselves. There is some hope for web3 but you have to ignore all the noise. It shouldn't be forced on anyone. Those who participate in those challenges benefit from that.
- Erik: By their booth size, there seems to be money in it, still!
- Erik: Another observation, Unity booth used to be huge, this year just the creators lounge.
- Noel: Re. generative AI, what did you see about it? I only saw one example related to a company allowing to generate game artefacts such as swords. Demo did not work well due to network issues though.
- Erik: I did not have much time to tour GDC in practice.
- Andrzej: This space will likely look very different in one year from now. Some of the products that were released this or last month may have stopped to exist by next GDC but others will have replaced them. In the media, there's a fear that AI will replace all the jobs. It's interesting to see that, for game developers, it's more about switching to a different set of tools, as there's a lot of work to be done to harness generate AI. AI in making games is just a different set of tools that will help developers. I hope there will be some cool stories to share.
- Richard: What's the biggest questions/hurdles that come back when it comes to publishing games on the web?
- Erik: One of them is Unity games. Of course, Unity has web export, but the game is far from optimized for the platform, starting with size (200MB won't work on the web). Investing the time in optimizing the build for the web.
- Richard: Promoting the game as well?
- Erik: That happens as well.
- Rafael: The distribution has been broken for quite some time on the web. No high quality places where developers feel the trust, with tools, SDKs. It's a matter of people getting an understanding of the opportunity, so that they may ponder whether it's worth the investment. When people see the value of it, things change. The more money we can bring to the web, the more content you'll see.

- Erik: How about in-app purchases? That's a question that keeps coming back. We don't have that feature in Poki for instance.
- Andrzej: Different experiments. Web Monetization API was an example. The fact that you can promote a game with a single link is a powerful feature, even though that's also a problem. Where is the Steam for web games? That's a question I keep hearing. We want the freedom, but we want the big place where everybody goes to get our game. Trying to experiment with different models.
- Francois: wondering about interoperability of the platforms where game developers can publish games on the web.
- Andrzej: From a developer perspective, that's needed. You need to adjust things for each and every platform. Different SDKs, different information to provide.
- Adam: From a game engine too, that's interesting. If there was a standard for porting games to the Web, that would be great. "Game web platform"
- Russel: Something we'd be interested in as well. From our side (GameMaker), we're seeing a big uptake for web exports for game jams since Christmas. That's good, I had always been disappointed by the figures for HTML5 exports.
- Adam: Godot is super used in game jams. It's all about to play a game quickly. Super useful for game jammers to remove the friction.
- Andrzej: There are game jams specifically for web games :) Worth promoting them!
- Erik: Maybe we should try to get other game platforms in there. itch.io? Y8? Etc.
- Andrzej: That would be great!
- Rafael: Yes, some fragmentation. Arkadium comes to mind as well.
- Adam: Maybe it's about saying "we want to hear from you".
- Andrzej: Thanks so much for all the input! The future for web games seems bright, I can't wait to see how this will all look like next year!
- Adam: Long live the web!

January 16th, 2024

Agenda

https://www.w3.org/events/meetings/7bb4dcbb-7027-4c78-a9bd-17ffc76831d3/

Attendance

- François Daoust
- Erik Dubbelboer
- Noel Meudec
- Andrzej Mazur
- Koen Bollen
- Florin Ciornei
- Georg Zoeller
- Fabio Alessandrelli

Gamedev.js survey 2023 results (Andrzej Mazur)

- Results not yet published but about to be!
- Survey was quite long: 35 questions on a variety of topics, including happiness of developers.
- 3rd yearly addition
- Only ran over two weeks, but almost 500 responses received.
- Compared to previous years, questions and answers were opened on GitHub, leading to 12 suggestions to amend the survey. That's nice!
- Pretty decent representation around the world, ~80 countries represented.
- We asked about technologies. Interesting to see WebGPU rising. WebAssembly also got more interest from people.
- The AI topic was added to this survey. AI tools already have 22% of answers to "Where are you getting your graphic design assets from?". 8-9% for audio assets.
- The monetization question: <$1k was the most chosen option last year. We added 0 this year, chosen by 60% of respondents. Lots of game developers don't make any money at all. Perhaps students.
- Overall happiness seems similar to last year, slightly better. Those are pretty happy developers.
- Report will be published at: https://gamedevjs.com/survey/2023
- Feel free to use the results!

Use of generative AI to create games (Georg Zoeller -)

See the recorded presentation and transcript.

Q&A

- If you're not doing anything with AI right now, at some point, you'll be too late. When is the right time to start?
    - That's a good point. You shouldn't rush into adoption, but still "play" with the technology. Adoption may lead to all sorts of scary outcomes. The system may not be built for what you're trying to achieve. Human judgment seems still required and fundamental in most contexts, except perhaps games where we love to be loose and edgy.
    - Not only "don't adopt and die", it may also be "adopt and die", as you may lose your employees in the process. Some balance to be found.
    - No one has the muscles to run a company in a system that is so unstable.
    - Need experts to guide you on where AI may be useful, when applied to a specific topic.
    - Business risks: if you're based on Stable Diffusion, what happens if one of the courts cases in the US jeopardizes things
- Fabio: The main thing that is unexplored is verifiability. As developers, we tend to want to verify things, write determinic things. It may be hard. What you get out of LLMs is more probabilistic. Regarding the example of translation, how do you verify that? How do I know that the translation is written in a way that I cannot be sued for (e.g., does not contain sexual or illegal content)?
    - Regarding the engineering, the technology is indeed written as a "daemon". It has all the properties of a daemon, including the fact that it lies. All the exploration we're doing is trying to frame the daemon. We give the daemon to consumers because we can't run it in business systems yet. End users are the ones who verify the results. They are the ones who check the outcome of Copilot. For translation, GPT-4 is phenomenal with English, not so much with other languages.
    - GPT-4 has all of GitHub history. When we look at scaling, the logical thing would do draw a line between versions, and make previsions based on this. OpenAI's thesis is that it's game over at 5. 1 and 2 were not that impressive. The jump in capability came around 3, and then 4 really showed us how to make a difference. We're good at predicting how long things take usually, but for this one, we don't really know what we're going to discover as we add more parameters. I see 30% increase in developers throughput since ChatGPT came out.
    - Be careful, be humble about making statements. We discovered a second way of building things, e.g., drawing things. Old AI had confidence, could say "bicycle,

95% sure". New AI is impressive but is always 100% sure it is correct. When we get the confidence back, things will be better.

- Will OpenAI be overtaken by something else?
    - Very hard to make predictions with AI!
    - Proprietary AI and local AI.
    - Convinced that the next Apple device will embed an LLM.
    - 5 large companies in the world that have capabilities. Only 1 company not in the US. Question is more: what is the top N?
    - Open source AI is probably going to be progressing.
    - More important to look at what is the right AI for which job.

More from Georg: Relax. Don't worry about AI Tech Adoption (December 2023).

November 28th 2023

Agenda

https://www.w3.org/events/meetings/0f82134f-2f07-4e82-b668-3bdf7927e59d/

Attendance

- François Daoust
- Andrzej Mazur
- Koen Bollen
- Adam Scott
- Gaëtan Faramaz
- Florin Ciornei
- Erik Dubbelboer
- Fabio Alessandrelli
- Raf Mertens
- Bjorn Ritzl
- Noël Meudec
- Rachel Yager

Web port challenges in Godot 4

See the recorded presentation and transcript.

Q&A

- Footprint? How small can I get an empty Godot project for WASM to be?

    - You can trim a lot of things that you don't need. For instance, text support can be quite large in Godot but can be removed. The uncompressed size should be around 20MB. It used to be 12MB uncompressed. Compressed with Brotli, around 8MB. And then your content is your content.
- Any work on improving that? Probably still a lot of code that is not being used.
    - Godot is a general purpose engine. You can recompile without 3D, physics, text support. But if you want to provide your users with something that is complete, there won't be much to remove. Godot is open source, so recompilation is easy.
- Features are on by default?
    - Yes. The problem is that you need to recompile the engine. We provide the full build.
- SharedArrayBuffer not going to work with ads in the foreseeable future. You said you're working on bringing it back to single-thread. Are you looking into multi-thread without relying on SharedArrayBuffer, e.g. relying on postMessage?
    - We're already doing that, e.g., for audio.
    - Godot is designed to work on any platforms. All platforms support threads. I'm not sure how much we can change the internal behavior.
    - EMScripten essentially creates Web workers for us, this is based on SharedArrayBuffer indeed. We could perhaps see how to have it use postMessage.
    - It's really hard to do without rewriting the engine, requires JavaScript specific code, web-specific code.
- Any idea of the timeline for both the reintroduction of single-threaded for the web and support for WebGPU
    - Next iteration for single-threaded. That's the plan (but not a strict milestone), a priority task for Adam though.
    - For threads, we use a system internally with a pool. It should be easy to adjust to single-threading.
    - The problem with WebGPU is that it's not a standard yet. It's still changing. With the standard changing daily. It can work on certain versions of browsers on certain platforms for now, but things are flacky. For a project that has a limited amount of support, it seems a bit of a waste to focus on that now. WebGPU/WGSL are Editor Drafts for now. Planning to work on it when specificatoins reach Candidate Recommendations and start to ship across browsers
- Any specific stats about how many people moved to v4 compared to v3? Any active work on trying to convince people to move?
    - We don't really know numbers. We have active companies that invest on Godot, part of the board, and they themselves use Godot v3 because they target mobile and some features aren't there yet. The new Godot v4 are the goal, but the modernization of the engine broke some things, e.g., switch to WebGL2.
    - New projects are done with Godot v4. People who need to target the Web may still stick to Godot v3. Split depending on the target.

- Example of Game Jam where people were initially excited about v4 but then rolled back due to issue.
    - If you can set COOP/COEP up, things should be mostly fine.
    - Support in itch.io being added
- Some of the specific challenges on Web technologies (WebGPU, WASM, WebXR, COOP/COEP) you mention would be - or would have been - very useful input to standardization groups that develop them. Contributing to standardization groups can be time consuming. Could it have helped avoid some of the challenges that you faced?
    - Godot has grown from a small open source project that follows existing technologies. Reaching a level where the project might also benefit from influencing the ecosystem.
    - Problems with WebSockets raised last time (back in 2021) are still current. Trying to show that they can be solved by browser vendors by creating a proof of concept.

June 27th 2023

Agenda

https://www.w3.org/events/meetings/cb63e8a4-af75-43f9-8728-3b8f1f79921b

Attendance

- Björn Ritzl (Defold)
- François Daoust (W3C)
- Andrzej Mazur (Enclave Games)
- Paul Gadi (Outplay Games)
- Koen Bollen (Poki)
- Noël Meudec (Meta)
- Erik Dubbelboer (Poki)
- Florin Ciorney (Crazy Games)
- Richard Davey (Phaser)

Defold game engine

Björn presents slides. See recorded talk with transcript

Q&A

Erik: How many web devs write in C vs. stick to Lua?

Björn: Basically all of them write Lua. They use C++ under the hoods but don't need to write C++ code. Also Typescript to Lua is being used.
Björn: About the survey, we surveyed our community, 15000 active developers (we don't really know the actual number), with a couple of hundreds who responded.
Koen: Are you thinking about supporting WebRTC for cross-communication purpose?
Björn: I don't know. Isn't WebTransport going to be better?
Koen: Well, it's different. WebRTC can be used for peer-to-peer communication. WebTransport is exclusively for client/server communications.
Björn: We have some partnership with HeroicLabs (Nakama backend service) to develop a multi-player game.
François: Wondering about the positive "start with web" story you told.
Björn: Yes, we hear two sides of the story, of course. For the big players, they start with the web. For mobile-only games, the Web is unavoidable. For smaller players, it's almost impossible to publish a game on your own on mobile these days. You need a publisher these days.
Andrzej: Tools to export a game to specific platforms are getting better and better, but it seems there are still struggles to do it. Example of vampire game, which started with Phaser but had to switch to Unity to target consoles.
Rich: For Phaser, I would say that 80% of users are web developers willing to develop a game. For Defold, more game developers?
Björn: Probably, yes. Different mindset. Melsoft is mobile-only.
Paul: Outside of deployment platforms, wondering whether games can be successful on the web
Björn: I think so. We hear a lot of good stories from developers releasing games on Poki for instance. If the game is successful, there is an opportunity to release to other platforms to make the game more profitable. Some devs want to see their games on their game console, because that looks like a life achievement, regardless of whether that's profitable.

Updated goals for the group

Noël: Discussed with Tom, Andrzej and François about updated goals for the group. We came up with 3 updated key goals, which we'd like feedback on. First one is "Listen to web game developers". Basically what we've been doing through inviting presentations. "Listen to web game publishers". "Reach out to non-web game people", which is something we have not been doing much until now. These are the three main directions that we would like to explore. This is something that we would like to build with you. Any comment?
Erik: My first thought is that it's a lot. Maybe we should pick one or two of these things and focus on that.
Andrzej: We started from 10 goals ;)
Koen: The first two goals feel very passive. And then what? If you just listen, that could be a waste.
Andrzej: We have "feeding back" as subgoal. That's a way to create next steps. We can assist with connecting with other groups. It is indeed important to be more active.
Tom: Making connections is something that we've been doing. The third goal seems harder to me. We've been doing good things on first two goals. Perhaps time to focus on the third one?

Rich: Do we have enough people? Much more diverse conversation when we had more people in the calls.
Andrzej: Idea is to reach out to more folks with these goals. We have ~90 people in the group. Are they not interested anymore?
Rich: Just wondering what may scare them away. “W3C” group?
Andzrej: Nothing prevents people from joining the group, for sure. Maybe we need a marketing campaign to encourage people to join and become active. I could see that happening through our respective communities.
Erik: We could reach out with some template text
Björn: I would definitely use such a text if it existed.
Noël: What kind of things could be attractive to new audiences? It’s hard to bring new people. Maybe we’re “unwelcoming” because we’re a small community. What kind of text. What kind of appeal could we use to attract new people?
Rich: They probably just don’t know about it, to be honest. As long as people would benefit from it.
Erik: As it’s a W3C group, I don’t think that “casual” game developers would be interested in joining. More technical ones, most likely.
Paul: I found the group because I was interested in how to make web games better. Something to look at: people who are not only into building core game engines, but willing to improve their games. Some people use three.js for instance.
Noël: I’ve been talking with people who are trying to make audio APIs to make life easier to developers. Could be good candidates. I’m hearing that we should come up with a short intro that clarifies the goals of the group and make it appealing for developers we’d like to see become active. We’ll discuss with Tom, Andrzej and François and get back to the group.
Andrzej: This feels like a group that can solve strategic issues, interoperability problems. We can somehow address that.
Florin: Could we make some Google Forms so that people can submit issues perhaps?
Erik: W3C group means standards for me. I have been thinking about a save game API. What if this group could develop such an API? The API could use local or cloud storage. What if this group made that?
François: Just noting that the charter puts actual technical development as out of scope for now. Charter can be changed, of course, it does have some IPR implications. Today, the group can certainly raise the need for a save game API and start discussing technical directions!

April 18th 2023

Agenda

https://www.w3.org/events/meetings/0a6e24f0-e593-4aca-8c9f-a0ded7cd5cdf

Attendance

- Richard Davey
- François Daoust
- Koen Bollen
- Andzrej Mazur
- Sandy Aggarwal
- Rachel Yager

Welcome Andrzej as co-chair

Francois: Andrzej is joining as co-chair to help Tom and Noël. Welcome!
Andrzej: Hoping to see more attendance in the group

Introductions

Sandy: Day job is work at a bank. Involved in fintech. Games off and on. I've also been involved in open source, blockchain, DLT. My focus is not blockchain first on everything though! Trying to assess where blockchains and gaming can fit together. Working on identity and pay.
Koen: Game developer, backend engineer at Poki, web games platform.
Richard: Phaser creator, used in games.
Rachel: W3C evangelist promoting activities. Also looking at educational aspects of gaming.
Andzrej: Indie game developer running a tiny studio. Also running the JS13K competition (running right now!) Trying to move the Games CG forward with the new c
Francois: Media & Entertainment champion at W3C. Gaming pushes the envelope of the web platform, interesting to find gaps and explore solutions.

MESIG - Gaming Subgroup

See the recorded video and transcript.

Q&A

Koen: Regarding the game that you're trying to prototype. Piece of advice I'd have is to pick up an existing game otherwise you'll just spend time on the actual game and not on the parts that you'd like to explore.
Sandy: Good point. I agree. Any game that we could use as a starting point?
Andzrej: I think that we could reach out to some communities. E.g. games submitted as entries to JS13K are open source and people are building actual games on top of them. I could imagine extending such open source games.
… One comment to the presentation itself: you kind of mentioned that it's not blockchain specific. Even from the blockchain point of view, it's interesting to see that there is the holy grail

of interoperability: even if blockchain is standardized, there are many of them and they are not interoperable. Something that would work no matter the blockchain that gets used, would be an interesting area to explore.
Sandy: If you could reach to folks, that would be great! The focus here is really not on the blockchain. Blockchain may help with identity providing. But game data and the like absolutely do not the blockchain. The only places where blockchain could perhaps be useful would be as identity stores, so that we could welcome people from other games and platforms. E.g. id.me: 100 million users, powered by a blockchain. The blockchain is essentially just a database, shared and public one.
Andrzej: Presenting this as a bounty for developers to try seems sound. Presenting the process of how you would pick up developers would be great, so that we can advertize that in a blog post, e.g. around JS13K.
… As long as expectations are clear, that sounds good.
Sandy: I shared the link to mentees program. We got some candidates already, although not necessarily from a pure game developer background. Also, main outcome is a research paper.
Andzrej: That could align with Google's season of docs. Program to document open source project. I did that a few years ago. We created a game with a tutorial. I think they have specific timeframes though. Google is trying to connect open source developers with people who are good at documentation. This could be a great fit. See blog post I wrote.
Sandy: OK, I appreciate the help. I could also use some help with using tools, e.g. Unity.
Richard: I cannot help with Unity
Andrzej: There are developers in the jam that are used to Unity and Unreal game engines. Feel free to join the Discord. I don't know someone specifically that I can connect you with but we can talk about me inviting you to the game discord.
Sandy: Also in touch with people used to Unreal who might be able to help.
Rachel: I note I'm trying to setup a W3C chapter. I could invite you to lead a discussion forum and ask the web community that comes there.
Sandy: Excellent community here in the Games CG because it is focused in games. At Hyperledger, some people are more looking into cryptogaming. We're trying to keep clear of this. My hope is that we can come up with recommendations that can be utilized and hopefully eventually make their ways into web standards.

What's next with W3C Games CG

Andrzej: Asking for help to promote future meetings
… Also, it would be awesome to have a presentation on the latest version of Phaser, and your ideas about Phaser 4.
Richard: That sounds good. It may be interesting to look into where the web is going rather than on Phaser specifically, so more focused on technologies.

September 6th 2022 meeting

Agenda

https://www.w3.org/events/meetings/243c41c3-0326-40c2-b3ad-5fe640dddb11

Attendance

- François Daoust (W3C)
- Koen Bollen (Poki)
- Andrzej Mazur (Enclave Games)
- Tom Greenaway (Google)
- Noël Meudec (Meta)
- Rachel Yager

Open web games directory

Tom presents slides:

- Idea is to create a directory, to be managed as a community, both in terms of how it's built and what it contains.
- Reason for this is linked to discoverability. Indexing web games is hard. We feel that a directory could create a system for others in the community to query it and search certain types of games.
- Another factor is that games are more visual than text, so automated indexing is harder by definition. A community managed directory could help.
- Part of the problem is agreeing on a common format to describe games.
- More specifically, this could be an open source project based on a standard format (slightly more complex than the one I proposed previously), with a crawler and an API endpoint
- In terms of standard format, we think starting from the web game schema, but also it would be nice to review the open mini games format.
- Reviewing architectural diagram: games get submitted, cron job crawls the queue of submitted URLs and fills the directory. Then an API for searching and filtering. Probably the API would be JSON-based or something like that.
- Ultimately, that shouldn't be a huge amount of work.
- Some possible future directions, e.g. metacritic
- Some of the challenges include:
    - ways to deal with games that only show up when users are authenticated, e.g. Facebook games. Maybe if the metadata was still served to unauthenticated users, things would just work.
    - how to prevent abuse. Submitted stuff that aren't games for instance

    - Also how do we trust who adds to the list? Do we need to create a review process?
    - Where do we start?
    - And of course who pays for the server?

Discussion

- Andrzej:
    - This is close to heart. Discoverability & monetization are top key topics for HTML games.
    - Monetization is mostly adverts with web monetization and web3 APIS changing the statu quo.
    - For discovery, making games is much easier now. Ideal would be a Google-like search result along with a video to present the game. Ideal for games developers and end users looking for games.
    - Perfect scenario would also have a games tab in the search result.
    - No issue with submitting games to JS13K competition through GitHub, so GitHub feels like the easiest way to start. Developers should be familiar with it already.
    - A dev named leereilly created a list at https://github.com/leereilly/games. This could be a starting point, or not.
    - The list "Awesome web games" also comes to mind.
    - The biggest value of the list would be for game developers themselves. If I submit my own game, I would like to keep the directory clear of spam. I would likely want to be involved in the review process to make sure that this remains the case. Will help ensure that the right games are in the directory.
    - No specific opinion on the format, just need one schema. No specific requirement comes to mind.
- Tom:
    - Just to clarify, open source is for the directory itself, games do not need to be open source.
    - The OMG proposal is trying to add 3 pieces of metadata including pwa manifest, game manifest and resources manifest to improve the experience. I don't think that you need to complete all 3 of these to be in the directory. Partial compliance seems enough for the purpose of building an index
- Koen:
    - I may be biased with my Poki hat
    - Interesting perspective. I think it could be useful to integrate with Poki but also with itch.io. An automated way to submit games for instance?
    - Speaking from a Poki perspective, all our games are valid, reviewed.
- Andrzej:
    - That makes sense. Portals could submit data themselves. Or there could be a mechanism to ingest content from portals. Need to find mechanisms to identify games and keep systems in sync.
- Koen:

    - Another thought is that this could perhaps be handled as a pure GitHub thing, with PR being used to submit games, actions to crawl, etc.
    - Metadata is likely not going to change often.
    - This would mean no real need for a real server.
    - The data may not be fairly big.
- Rachel:
    - I second the GitHub approach. It's cost effective and will reach a broader audience
- Koen:
    - It would make things more transparent as well
    - One final thought I had on the matter is discoverability and quality. Remembering when Steam went open for everyone to upload their games. Result is lower-quality.
- Tom:
    - If we integrate MetaCritic, then we may need a database.
- Andrzej:
    - We don't have to have everything in one place. E.g. rating could be done on the side. The voting app could be a separate thing as well, while the catalog remains on GitHub. If we need different benefits from different parts of the system, it could be separated.

Challenges

- Logged in status:
    - Direction to serve metadata when the user is unauthenticated seems a good direction.
- Preventing abuse:
    - Noel:
        - For Instant Games, we choose carefully who can submit games. Having the first round of vetting developers prevents spam from random people. We sometimes ran in crazy situations with 500 submissions per day per developer.
        - Growing reviewers is a good idea, we should select the first ones carefully. Choosing people who can review submitted games in any case is important.
    - Andrzej:
        - I agree that it can bring a lot of spam.
        - If we communicate quality guidelines right from the start, that would help
    - Noel:
        - We could seed the directory with games from the portals we mentioned that already run quality checks.
    - Andrzej:
        - Yes, that would get us the "verified developers" who could be interested in having a main open directory of web games devoid of spam submissions.

- Tom:
  - I think it would still be useful to enable another path of game submission that does not go through the portals. I would be a little bit wary to see us gating the community too much. I want to make sure that there is room for the little person. Indexing the open as much as possible.
- Rachel:
  - Another path. As a W3C evangelist, one thing I'm trying to drive is education. Some game developers are young. I reach out to universities, schools. I do think that the big portals are important but that is another way of driving new developers to the directory.
- Andrzej:
  - The mechanism to submit a game directly is important but I'd like to emphasize that we should clearly communicate that people should not expect that a team of 50 people will jump on submissions and review them. Maybe we will have time but maybe it will take time. For instance, it could take weeks instead of hours. Many folks could misunderstand this, so it is important to communicate properly.
- Noel:
  - +1 to this, the scalability would be horrible if it's total randomness. I like the idea of growing the community through trusted links
- Tom:
  - I don't disagree but I think we can find a middle ground, e.g. where a trusted developer can invite other developers, with restrictions including max number of submissions per week, etc.
- Noel:
  - That has been my daily job. I just want to make it clear that these problems are real.
- Tom:
  - One of the advantages here is that we could leverage the community. Difference is we're low on resources, but possibly strong on the community.
- Noel:
  - Could be that only the games played by the community end up in the directory.
  - That is meant to be positive. If we're strong on the community, let's leverage this!
- Tom:
  - I'm not trying to constrain the solution right now, just want to make sure that we find a path that works
- Andrzej:
  - It feels that the Open web games directory could be the Steam for web games. The fact that it would be open by default, then if we can position it as the source of truth, I'm confident the community would sit on our side.
- Koen:

        - IMDB is a better comparison, I think. Steam is closer to Poki in that you can play the games.
    - Rachel:
        - Thinking about other criteria, linked to a game jam / educational concept for web standards: accessibility, audio, game design itself, visual.
    - Noel:
        - Having metadata on what accessibility support a game has is certainly a good idea.

Next steps

- Noel: Wondering about next steps
- Tom: Another meeting where we define what constitutes an MVP, and split up the work within the group
- Andrzej: I can try to get leereilly to join the next call
- Tom: That sounds cool! Another part of the work that can move forward in parallel is the schema itself. Just reaching agreement on the schema. Then how we're going to design. And then the vetting process.
- Koen: Also the guidelines will be tough. I will also definitely check in with my colleagues at Poki.
- Noel: For guidelines, we can review those at Instant Games. Fun fact is that we had a hard time defining what a game actually is, e.g. are quizzes games? Maybe best is to align on existing guidelines where possible.
- Rachel: Gamification is a big thing. Wondering whether to include it or not.
- Tom: OK, we'll schedule a follow-up call.

July 6th 2022 meeting

Agenda

https://www.w3.org/events/meetings/cc13b701-b00a-44c6-8f6e-831f3ee0ed97

Attendance

- François Daoust (W3C)
- Noël Meudec (Meta)
- Tom Greenaway (Google)
- Andrzej Mazur (Enclave Games)
- Koen Bollen (Poki)
- Luke Stapley (Cocos)
- Huabin Ling (Cocos)
- Wanfang (Cocos)
- Erik Dubbelboer (Poki)

Cocos

The Advancing Web Gaming To New Heights with Cocos slides (PDF, 13.5MB) presented by Huabin are available.

Some notes taken during the presentation:

- Huabin (a.k.a panda) presents Advancing Web Gaming to New heights with Cocos
- Huabin is technical director of Cocos
- Cocos has been a well-known 2D game engine on mobiles. Engine has evolved since then.
- Cocos Creator has a new UI in v3.6 (not released yet). Showing scene imported from FBX.
- Workflow is built around the web. Game can be previewed within the browser and deployed within a native app. No need to wait for a build process. Web workflow is recommended.
- TypeScript is used for game logic.
- Editor is based on Electron, also using web technologies.
- Example of successful games:
    - Top War, published to native platforms and the web. Merge gameplay is lightweight and fun. 100 millions downloads with $20m monthly income.
    - Ubisoft Nano, 10 multiplayer party games on web, deployed to 188 countries, 4s loading time in browsers, 10mb package size.

    - Three Kingdoms. Graphics are really good. Cross-platform game. All are using the web platform to attract players. Experience is the same regardless of the platform.
- Demo of a complex rendering pipeline:
    - https://preview.cocos.com/lake/
    - Uses a number of advanced techniques, including TAA
    - Consumes a lot of GPU, runs smoothly on high-end devices. Can also run on mobile devices

Engine architecture

- At the bottom: GFX, Render Graph to organize all rendering activities, Render Pipeline. This pipeline will be introduced in 3.6 (to be released in August 2022).
- The green part is written in both C++ and TypeScript. When engine is running in the browser, it uses the TypeScript engine.
- At the top: TypeScript components. More and more of the modules are being moved to native for better performances.
- GFX backend wrapper abstracts away all different platforms (Web with WebGL, WebGPU in the future, but also Metal, OpenGL ES 3.0 and Vulkan). WebGPU part is already implemented.
- Using upward adaptation for older backends: command buffers, pipeline objects, etc.
- Also care about backward compatibility: games need to run on as many devices as possible. For example, GPU instancing uses a WebGL1 extension, and built-in in WebGL2, etc.
- When float texture is not available, we will pack floats to TGBA8 textures. It's quite complex, but means the application does not need to care about the differences in the backend layer.
- Showing a GPU based skeletal animation, with different materials, transparent helmets. Good performance on low-end devices.

Engine features and recent updates

- Engine is designed fully physically based. Lighting, material, cameras
- Lighting: environmental convolution map, and mipmaps.
- Shadows: Cascade shadow map has been introduced in 3.6. They all support instancing batch.
- Material: Separated lighting and surface descriptions. No need to modify every part of the shaders. Shaders use GLSL 300 ES. Uniforms get preserved and can be edited in the keyframe editing tool.
- Smart material conversion is being introduced as well. Internally, everything is physically based so converted to PBR.
- End of last year, Cocos introduced a marionette animation system.

Our Journey to WebGPU

- Started work two years ago.
- Running demo in July 2021 but result has not been integrated in the engine
- Now working on second version of WebGPU implementation.
- First WebGPU demo available online.
- The initial implementation is the classic web-way. gfx-wgpu interfaces with WebGPU. But we've improved that by introducing WebAssembly glue code. This may create some initial performance loss. Reason is that in the future, Cocos envisions that all C++ code will be compiled to WebAssembly, instead of maintaining two codebases: better performances, and allows to reuse more advanced code that has been produced in C++.
- Question is whether there could be a way to pilot the WebGPU API through the C++ code compiled to WebAssembly. That would be ideal.
- Huabin shows the full shader compilation graph. SPIR-V is used for WebGPU backend

Design principles

- Performance is at the core of everything. Games are among the heaviest applications in the world.
- We want to make cross platform development as easy and seamless as possible. High priority in our engine design.
- We also want to maximize accessibility. That's also one of the reasons why we are embracing web-based platforms because they can be accessed by everyone, including older people, children, people with disabilities.
- We want to adapt to modern graphics and computing power to the web. Using compute shaders for instance, but ES compute shader are not properly supported on the Web. WebGPU would solve that.
- Also, we'd like to reduce the engine maintenance cost. That's the reason why we like the WebAssembly approach to save a lot of men-power and energy.

Questions & Answers

- Koen: How big is the team working on Cocos Creator? What is the download footprint of an empty project
    - Huabin: Company is 300 people. Development team is less than 100. A lot of games of our overseas developers publish games on Poki. Smallest package should be around 2MB.
- Francois: GPU for the Web groups are keen on feedback. WebAssembly proposals are underway to interact with Web APIs more directly. Wondering whether you keep a close eye on progress?
    - Huabin: Part of the GPU for the Web Community Group and tracking progress. We were building our renderGraph architecture so we couldn’t allocate resources much, but the second half of the year we’ll try to be more active.

- Andrzej: Cool examples of big games. Do you have any data on how many devs are using the engine? AAA vs Indies?
    - Huabin: I show fancy demos in presentations, but the dev partition is more toward mid-core games and indie devs. We haven't been working on 3D technologies for as long as others such as Unity. Only been 6 years. Still young. We want cross-platform and accessibility to provide added value that other engines may not focus on.
- Francois: Do you suffer from fragmentation of WebViews?
    - Huabin: Struggled a lot in the past. Even today, there are more and more platforms to deal with. E.g. WeChat Mini games use web technologies but a different private runtime. It makes it different from browsers. For graphics, we're using a common wrapper to hide away differences but that comes with a cost. Other parts like file system, downloads, asset management, etc. For that part, we are lucky in the sense that we started with a web mindset so always come with a web perspective. It's easier to port web features to native rather than having to port native approaches to the web. For instance, for file system, we will create a virtual file system. It's a way for us to handle it.
- Tom: Are you in touch with Corentin on WebGPU?
    - Huabin: Not directly but we follow discussions in the GPU for the Web CG

JS13K competition

- 11th edition
- 13 August - 13 September
- Please share!
- Consider competing, it's fun!
- Koen: We'll make sure that it gets advertised on our Discord

March 29th 2022 workshop meeting

Agenda

https://www.w3.org/events/meetings/302eaf19-f4a2-4bc1-8733-17e808b5cd1d
"What is the future of NFTs in gaming? Is the web a suitable platform for this technology? Can it be a form of monetisation of web games?"

Attendance

- Andrzej Mazur (Game developer, games jam organizer)
- François Daoust (W3C)
- Paul Gadi (OP Games)
- Noël Meudec (Meta, working on Instant Games)
- Richard Davey (Phaser)
- Rachel Yager (W3C Evangelist)

Blockchain and NFTs in web games

Context

- Summary of monetization workshop:
    - We discussed monetization topics in HTML games, ending on blockchain topics for each of them. Decided to leave it for today and focus on other ways to monetize: ads, patreon, different ways of supporting creators, in-app purchases.
    - We went through app packetization as well
- Would be good to have game publishers in attendance.
- Richard reflects on experience with packaging a web game and distributing on Steam. As long as the game is good, getting millions of players is doable, regardless of the technology.
- Phaser and NFT:
    - Building a game so it works on the blockchain (e.g. tokens that users can buy in the game)
    - Projects built on the web with a game UI with NFT at the heart
    - Seeing them used everywhere.
    - Web3 game => Web3 platform. Fascinating things going on.
- Paul reflects on his experience with developing games. Looked at blockchains with CryptoKitties. Skeptical at first with "web3", but seeing many successful projects. Created NFT project and shared funds to open source projects. Good community.

- On Instant Games, Meta is not looking into blockchain technologies yet. Other teams in Meta are. Only monetization mechanisms so far: in-app ads and in-app purchases.

Success stories:

- Example: CryptoKitties.
- Different business model. Lot of blockchain games.
- Example of project created with 10 000 NFTs to start with. A lot of it is also fundraising to raise a few thousands dollars.
- In terms of actual games: Axie Infinity?, Aurory.io, Sipher.xyz.

Understanding monetization aspects

- It shifts the model from developing the game and getting the money through in-app purchases, to getting the money and developing the game.
- For ads, people pay to place ad banners. For in-app purchases, you expect something in return. How is money generated with blockchain games?
    - NFTs can be sold initially through crypto. This gives actual money to develop the game. New financial primitives.
    - Tokens in the game itself, that gets value because it exists on a public blockchain (e.g. Axie Infinity).
    - Pretty crazy what you can do with these.
- NFTs don't have value in themselves, they get value because people attribute value to them. Like art.
- NFTs are like doing Patreon with new technology. Buying something as a token of appreciation that I support this developer/creator. I may not be expecting anything else from this token. I could use Patreon and pay monthly fees. Physical proof of my donation.
- Outside of the blockchain "proof", what makes NFT different from paying with usual money? People may be paying for the proof (example of giving money to someone vs. giving money to someone while recording a Tik Tok video of it). You may be getting additional value. With verifiable ownership, you can offer something specific, premium features. Not locked through services like Patreon. You can build your own rules on top of that.
- Ability to trade tokens without going through the initial service/game.

Demographics of people attracted to it?

- Complex technology, hard to understand.
- Successful projects are massively community-driven. Pulling people from Twitter, Telegram channels. Young demographics. The technical challenges are high, e.g. setting a secure wallet. Speed bumps all the way. But once you're into it, the community will stick to it, trading tokens from game to game.

- The young generation grew up buying virtual items. Massive range of people gettgin into it, including non-technology savvy people. Different mentality.
- Easy to envision the younger generation creating crypto wallets rather than bank accounts in the future, because more direct, and less interest in fiat money.

What about regulations?

- Risk of creating Ponzi schemes. Need to understand what is ok and not.
- True that there are massive amounts of scam. Side effect of the lack of the regulation. It's a matter of time before "taxmen" come knocking. Big actors will likely jump into it once regulations kick in.

Technical perspective

- Blockchain is not only money. More about decentralization. Technology is proving itself. Game developers can leverage the infrastructure, without having to worry about setting trading platforms, wallets, transparency, etc.
- Technical gaps?
    - "MetaMask" (Chrome extension) Most people paying on the desktop. Could W3C expose a wallet? Brave has a wallet built-in by default. Opera as well.
    - Web Monetization API
    - Remove the need for the in-between sessions, let the browser handle the intermediary steps.
    - Integration with IPFS (Protocol Labs): putting tokens in decentralized storage. Resolving IPFS links.
- One way to look into wallets is to realize that popular games have wallets already, they are just not exposed and cannot be managed publicly by users.

February 22nd Meeting

Agenda

https://www.w3.org/events/meetings/43010bf1-547f-4c0d-b61b-cadeaca4b475

Attendance

- Francois Daoust (W3C)
- Andrzej Mazur (Enclave Games)
- Noel Meudec (Meta)
- Song Xu (China Mobile)
- Huaqi Shan
- Fabio Alessandrelli (Godot engine)
- Erik Dubbelboer (Poki)
- Dan Druta (AT&T)
- Sudeep Divakaran (Intel)
- Eric Siow (Intel)
- Dominique Hazaël-Massieux (W3C)
- Louay Bassbouss (Fraunhofer FOKUS)

Games & Networks

Web & Networks IG overview

- Mission: explore solutions for web application to leverage network capabilities
- Song presents Web & Networks IG slides
- Workstreams:
    - Edge computing for the Web: initial research on use cases for web applications, and approach towards creating a "split-browser" model
    - Network quality monitoring and prediction, to adapt to network conditions.
    - Network emulation browser tools and trace formats
- Lately, working with Alibaba and ByteDance on challenges for offloading to leverage the computing power of the cloud. 3 approaches considered (send the WASM to the server, server fetches the WASM, browser provides a set of native APIs)
- Games discussed as use case for offloading. IG would like to gather requirements.

Games & Networks

- Fabio believes that edge computing is of more interest for large studios. Smaller players generally target less infrastructure. Main problem for networking in games is the API and the fact that they are subpar compared to native. HTTP, server-sent events, Websockets, lack of support for custom headers. Fetch is better but no way to cancel the request. Recovering from network failures is critical. WebRTC gives more flexibility but requires more infrastructure. WebTransport solves this but drops P2P. None of the APIs are complete. Need to fix the APIs before going further.
- Dom wonders whether the gaps have been captured somewhere. It seems to me that fetch has abort capabilities now. WebRTC is very actively looking at shaping the next version, notably the data channel part. Gaps are welcome!
- ACTION: Fabio to compile a list of gaps.
- Erik reports that Poki is developing an open-source WebRTC library for developers so that they can leverage WebRTC without having to handle all of its complexity. Most Poki games, and most mobile games in general, have limited computing needs in practice, so need for offloading is not yet a priority. Cloud gaming platforms may need such a feature more.
- Andrzej: Way many more developers focusing on single player games than on multiplayer games. Multiplayer is still too complicated. The barrier of entry is too big for indie game developers. Some still build multiplayer games, but usually complain about the technology that they need to use.
- Dan: Different perspective. Two models in the past: caches and CDNs. Caches tend to be more on the client side. CDNs act on behalf of content producers. You can think of offloading the same way. The argument against infrastructure for indie game developers is equivalent to CDNs. But it could still be useful to look into setting up an analogy for caches for offloading. I agree that, for most web games, offloading does not make a lot of sense, especially if it requires more infrastructure.
- Erik: We should keep in mind that AAA games could come to the Web in the future. Example of photo-realistic rendering of 3D models. About multiplayer games, the biggest bottleneck is that multiplayer is difficult in itself. Managing game state is difficult. Replicating state is hard. No real help from Web technologies. We probably need some more guidelines and common libraries.
- Eric: Looking from a business perspective. How to create a sustainable community and attract other players. How to build a healthy ecosystem with a wide range of stakeholders vested in the developments?
- Sudeep: From a network perspective, I'm assuming that the challenges for AAA and indie games are going to be the same. Are the requirements from a networking perspective different?
- Erik: It's slightly different. Bigger studios will use WebRTC because it's much more real-time, while most indie developers will use Websockets because that is the easy way.
- Fabio: Different kind of games. Some games require much more resources. Massive multiplayer games come to mind. Even First Person Shooter games. There is also

another level with 2D, casual, simple 3D games. Designing 3D games is much harder. 3D games have stricter requirements in terms of managing the state, and thus on real-time networking technologies. Simpler games can make do without such strong constraints.

- Sudeep: Round-trip latency, e.g. "mouse event" round-trip latency, should be the same regardless of the game. You probably have some sort of latency criteria that need to be met
- Fabio: Any fast-paced game will indeed have requirements on latency.
- Sudeep: From a gaming perspective, how do you take that into account? Measure network properties? Dynamic? Or will the game simply not work with higher latency?
- Fabio: Different techniques. If latency is too high, some games simply won't work (or will work but will be unplayable). The usual technique is to determine what things you want to see synchronized across players. You send raw events to the server, server calculates everything, and sends back objects that need to be synchronized. Client interpolates between these states. You might lose some packets but you don't care. Past packets are discarded. New packets are taken into account with interpolation to maintain an average latency level. When you have physics, you try to predict upcoming state. MMO require more optimized approaches and are more events-based.
- Erik: Another use case at Poki. We have some games that are local multiplayer games (players on the same keyboard). We have made some tests where a player plays the game and the other player gets a stream of the game and sends controls to the first player. That's an interesting approach to handling multiplayer in that you don't need to manage state across devices. Steam does that with "Remote play together".
- Eric: Has Google shared any needs with their Stadia platform?
- Noel: Tom is not around today. I think you should reach out to him directly.
- Dom: One lesson I take from this discussion is that there are clearly intersections between gaming needs and networks, be them from indie games or AAA games. We, the Web and Networks IG, are very interested in keeping track of such needs.
- Erik: Curious whether Web and Networks IG has monthly calls?
- Dom: Usually, we have monthly meetings.
- Sudeep: Meeting coming up next week, on edge computing and edge offload.
- Sudeep: We didn't touch on cloud gaming per se. Split model. Can that be enabled through Web APIs? How can the architecture look like? Edge can mean a machine sitting at home. Not only relevant for games. Also for metaverse worlds.
- Noel: We don't have cloud game providers in the CG for now.

January 25th Meeting

Agenda

https://www.w3.org/events/meetings/939b832f-11d6-40f1-bb60-c1922e31cc0c

Attendance

- Koen Bollen
- Francois Daoust
- Andrzej Mazur
- Fuqiao Xue
- Bjorn Ritzl
- Tom Greenaway
- Fabien Alessandrelli

Next generation Monetisation

- Brainstorming happens in a slide deck
- Koen shares https://www.youtube.com/watch?v=jAZapFzpP64 on how TikTok makes money
-

November 30th Meeting

Agenda

Attendance

- Francois Daoust
- Tom Greenaway
- Noel Meudec
- Kang Seonghoon
- Andrzej Mazur
- Koen Bollen

RoadRoller

- Developed for JS13K competition, which has 13KB entries. See post-mortem blog post
- Looking at 2020 winner, typical uncompressed limit is ~30-40KB
- Plenty of ways to compress content. But better compression needs larger compressor, so mostly useless in practice
- Managed to pack the compressor in ~700 bytes
- The details are only useful for data compression geeks.
- Up to 15% smaller than zip/deflate when you include the compressor (20% if you don't)
- Mostly useful for JS13K competitors
- Joke is that this made the 2021 version of the competition JS15K because this compression method allowed people to put a couple of additional KB in their games, e.g. a second game level.

Q&A

- Motivation to develop RoadRoller?
    - Kang shows different entries that he made in various contests. Not sure what to do for JS13K in terms of game, so decided to focus on data compression stuff.
- Andrzej notes that RoadRoller proved essential to the JS13K contest. Many folks were building extra features and were glad to be able to include them in the final build.
- Why develop RoadRoller in JavaScript?
    - Initially done in Python to produce compressed JavaScript. But it was too slow.
    - JS version proved to be much faster, even though it was a direct port of the Python code. Compressor was later re-written to be much faster in any case.
- Did you have any browser-specific issue? Or did it work perfectly across browser engines?
    - RoadRoller only uses regular ES6 features. One technical issue linked to accuracy of statistical functions in JavaScript.
    - If you're using V8 or SpiderMonkey, no problem.
    - Also much faster to switch to WebAssembly in browsers.
    - RoadRoller is a meta compressor that produces compression algorithms.
- Where can you find more room for optimizing? You mentioned an extra 1KB?
    - RoadRoller takes inspiration from past compressors in the demoscene.
    - Leveraging Squishy and cmix, optimized in starlix, could help optimize things further. cmix can compress typical JS13K to 8KB with the drawback that compression/decompression takes ~2 minutes!
    - All of that is speculative work, though
- Wishlist for 2022
    - Integrate better with typical bundlers used in JS13K
    - Produce optimal ZIP files, as doable for PNG.
    - Current JS13K developers typically bundle their ZIP separately and use RoadRoller at the very end of the pipeline to compress things further. Integrating RoadRoller earlier would allow to be optimized.

- Overall, lots happening in data compression. Kang believes we're in a data compression renaissance with specific compression techniques depending on the type of content being compressed (images, audio, video, etc.)
- Kang explains the Pareto frontier. Vertical is compression ratio, horizontal is compression bitrate. Different groups of compression techniques. Slow compressions that provide very good compression levels, etc.
- To bring a new compression format to the Web, the main problem is compatibility. WebAssembly could be used of course. In CSS, the CSS Painting API could be used for images. Not sure about audio & video. Eventually, you need a hardware decompressor.
- Francois mentions audio/video codecs, with WebAssembly being the only way to add more support for now, also due to patents
- RoadRoller can be applied to larger codebase but it will be very slow. 1 hour to compress 1MB of JavaScript, less than a minute to decompress. Could be made faster, but then we would lose on compression ratio, so it would be less interesting.

October 2021 TPAC Meeting

26 October 2021

Agenda

https://www.w3.org/events/meetings/dcf599f4-549c-4a8a-81a5-6437429bac3c

Attendance

- Alexandre (missing family name) from Boku
- Andrzej Mazur from Enclave Games
- Barbara Hochgesang from Intel
- Björn Ritzl from Defold
- David Tisserand from Ubisoft
- Dirk (missing family name)
- Fabio Alessandrelli from Godot
- Jeff Young from OCLC
- Karen Myers from W3C
- Kazumasa Okabe from Yahoo! Japan
- Martin Alvarez from W3C
- Richard Davey from Phaser
- Richard Lea from Rakuten
- Yves Lafond from W3C
- Xiaoqian Wu from W3C

Introduction

Noël gave a quick introduction and talked about:

- What the community group has done in 2021
    - Many guests: Pixi.js, Phaser, Coil, Unity, the BBC, Tencent, Construct, Godot, GDevelop, Chrome and more.
    - Two workshops:
        - "What holds back web gaming?"
        - "How could the web platform enable web games to be more viral?"
- Where do we see things going in the future?
    - A few words about where we are with Instant Games and Cloud Games and our intention to grow
    - Challenges we need to tackle together (always the same):
        - Monetization

    - Discovery

Group Discussion

Topic: "What do people imagine will change Web Games in the next year?"

(Note: Noël was both hosting and taking notes, so the transcript is a very summarized version of the actual conversation, meant to capture the essence of what was said):

- Noel: Are there any new techs or things you are particularly excited about and would like to talk about today?
- Andrzej: Blockchain in games.
- Fabio: WebGL and Web Assembly
- Barbara: AI/ML - Super resolution
- Karen: Monetization. Question for the group: what is the primary demographic for HTML5 games?
- Björn: Most of the games are free, monetized by ads and in-app purchases.
- Fabio: Similar to native apps.
- Andrzej: The audience is not very young. People who have access to payment methods. The average player is around 30. A lot of nostalgia.
- *Barbara (question in the chat): I have questions on AI/ML. How could it improve the experience? How ML could protect audiences (for example: minorities)*
- Fabio: Multiplayer through webrtc is growing but still a minority of the games.
- Andrzej: Multiplayer games are much harder to make, thus more rare. Both have a place on the market.
- Barbara: Are there gaps in the api offering to make multiplayer games?
- Fabio: Cross platform playing is desired. Webrtc implementation is a monster. The main issue is technological: implementation is hard.
- Richard (Phaser): Multiplayer games cost money.
- Fabio: Full peer to peer mesh helps reducing the cost but is prone to cheating (player hosting the game).
- Bjorn: We use websockets because webrtc is too complicated.
- Barbara: What AI feature do you need?
- Andrzej: Not needed for small businesses. It would be a better match for AAA games.
- Richard (Phaser): Same here, 0 request for it. Maybe need dev education.
- Fabio: regarding super résolution: it would be good to apply on canvas. It feels more like a browser feature than a game feature.
- Noel: How about 5G?
- Richard (Phaser): No developer is really looking forward to that. It is just convenience.
- Fabio: Low latency will be good but outside of that, no change.
- Richard (Phaser): Side benefits, such as streaming while playing.
- Barbara: How are you installing your game? PWA?

- Richard (Phaser): No install in general. Some games do it through HSS and PWA.
- Andrzej: PWA has benefits, but devs are slowly dropping it or doing with no effort. Sad to see because the tech is good.
- Fabio: Sad Firefox killed it.
- Dirk: Is anyone famliar with webassembly?
- Fabio: we are using it (Emscripten). It is working well. Interface with javascript.
- Richard (Phaser): We are trying to not use it. Not very open source friendly. It makes no sense in our context. It is not necessarily faster than javascript. Now JS is so fast.
- Yves: A word about latency. Webtransport is websockets ++. It is in progress, we might want to look into it for gaming.
- Barbara: Any word about webcodec?
- Group answer: No one is really using it for games at the moment.
- Barbara: Stadia will use webcodec and webtransport.
- Fabio: For cross platform engines it doesn't fit because it is only web, not native friendly.
- Noel: How about webgpu?
- Richard (Phaser): Everyone is playing around with it. Not mainstream yet.
- Fabio: In the future yes, but not at the moment.
- Noel: Does anyone want to talk about monetization today?
- Andrzej: How about Rafiki? The Web monetization API is not going to take if there is a need to set bank accounts and all. Rafiki should greatly improve the UX for players. It is being worked on by Coil.
- Barbara: is China a hot spot now?
- Xiaoqian: Chinese devs are not doing much on web. They are doing it in Cloud Game though.
- Richard (Rakuten): Game making in Japan is very Japanese. Visual novels for experience. One or two button gameplay (branching in stories). It might not be a good fit for the global market.
- Fabio: Godot get many contributions from China. Not sure if it is for web or not.
- Karen: How about educational games? Any innovation? VS entertainment?
- Andrzej: I know a studio focused on that. The pandemic might have impacted that area (in a good way?).
- Richard (Phaser): it was and is still huge.
- Fabio: I have seen a game that teaches coding from a game (Godot coding)
- Richard (Phaser): Many initiatives to learn programming through gaming (hourofcode.com)
- Richard (Rakuten): Nintendo lab.

September 2021 Meeting

Agenda

https://www.w3.org/events/meetings/ca95868f-2da0-42a2-a633-169d6a68a424/edit#agenda

Attendance

- Erik Dubbelboer
- Noel Meudec
- Francois Daoust
- Fabio Alessandrelli
- Richard Davey
- Clément Pasteau
- Florian Rival
- Andrzej Mazur
- Koen Bollen

Short JS13K report

Andrzej shares updates of decentralization and monetization categories during JS13K.
Clone of Quake hit the news.

GDevelop

- Florian: involved in GDevelop since the beginning
- Clément: joined development recently from Web and mobile development
- GDevelop: trying to build something that is "no-code", so not trying to compete to other game engines that require coding (such as Unity). Started ~10 years ago as an open-source project. Project is on GitHub.
- A few buzz words:
    - game engine based on PixiJS.
    - Interface build with React, good for building and reusing components
    - Started as C++. Not everything has been moved to JavaScript, some code still compiled to WebAssembly.
- Clément shows a few screen shots of the UI. Events are on the right: conditions and actions. In other words, when something happens, I want this to happen.
- The interface is using React. This is good to build a component-based architecture. Also good for an open-source project to define boundaries between components. It also makes it easier to specify the tools that can be used within the engine.

- Florian: This has been instrumental to building not-only simple games but also full-fledged and scalable games.
- There was no typing initially. Code slowly moved to Flow then TypeScript. Still a mix of the two. Typing strongly helps in the open source world.
- About the C++ part, core architecture was initially developed in C++. Remaining C++ code gets compiled through Emscripten to WebAssembly. If someone wants to contribute, they need to learn a bit of C++, which is a bit of work, but remains accessible.
- Mostly for the physics engine. But also considering WebAssembly for other parts.
- Started with PixiJS because it was useful at the time and performant. It is less useful now. It is great to see support for WebGL2 and we're looking forward to WebGPU in the future.
- We had some issue in iOS, which I'll talk about later
- Pain points:
    - First obvious thing: no way to store something and be sure that it will remain there. OS or browser can always delete things, you're never sure that resources will be preserved. The new File System API seems like a good solution. We want more browsers to support this. A major paint point
    - Consistent performance: It's very easy to fall out of the "happy path". Some warnings are surfaced on some debuggers, but overall you're never too sure so you need to be cautious. There are tools that can help such as Spector.js, but there is room for improvement
    - Garbage collection: It used to be worse, but same thing, there is no easy way to measure the impact or understand what triggers a GC.
      => One idea: a "super strict"/"never slow" JS mode, perhaps?
    - Issues with iOS, and every new iOS release. iOS is too constrained, forcing us to remain "old school". Something that we need to be careful about.
    - Monetization on the Web: For now, it seems that ads are the only solution. There should be ways to do something else, but not sure what can work. Been looking at Web Monetization, but not sure whether this is getting traction. Currently, best solution is to package Web apps in a native shell (e.g. using Cordova) and sell them on app stores.
    - Discovery on Web games: Hard to get noticed. Indie stores can help. Hard to do "remarketing" with limited push capabilities. Again back to wrapping Web apps in native apps. Some companies are promoting HTML5 games through.
- Web technologies allow the engine to run almost everywhere. Just a bit of a shame that we cannot go beyond some limits and need to package apps in a native application.

Q&A

Fabio: Same concerns in Godot for the Web. For Emscripten, are you using the minimal runtime or additional libraries e.g. for network I/O?
Florian: Only the minimal runtime. Really just about compiling C++ code. Some issues with miscompilation at the time when Emscripten was compiling to asm.js, but mostly everything is straightforward now. A bit hard to create binding between C++ and JS though, mainly from a

memory management perspective. If there was some garbage collection on the Web Assembly side, that would be great.
Fabio: Thanks for sharing. Our issues were mostly related to additional libraries.
Florian: I never wanted to go too far because support on iOS was uncertain.

Erik: Export to WebAssembly or JavaScript?
Florian: Most games compiled to JavaScript. Collision engine may be reworked in the future in WebAssembly.
Erik: Size of games?
Florian: Core is ~100kb or so. Every game is standalone, although we could perhaps envision hosting in the future. We don't want to constrain people. There is a way to export a bundled version hosted on GDevelop, but it's very barebone.
Erik: An "export to Poki" feature could be interesting :)
Florian: We can certainly discuss this.

Koen: About file storage, game editor or game assets?
Florian: It was more about the editor. This is why we consider the Web version to still be a demo.
Koen: Have you looked at the "persist" API proposal?
Florian: Not yet.

Noel: Top bets to monetize outside of ads?
Florian: I'm not sure. We could look at magic words like blockchain or NFTs, but not clear. We're not targeting developers so we need to make things as easy as possible.

Andrzej: Way to include only parts of the engine in a build?
Florian: That is already the case by default. If you don't use the physics engine, it won't be included for instance. Granularity could be improved but looks good enough, I thinkg

Noel: Experiment with PWA?
Florian: Less important for us given the packaging step

Richard: Working full-time on GDevelop?
Florian: Fairly new, but yet. We have multiple plans to cover the costs. Because we have an audience which are not developers, we believe that we can offer more services for them. We'll see.

Francois: "super strict"/"never slow" JS mode?
Florian: If the JS could throw an error when something needed to be de-optimized, that would be good.
Francois: seems like how asm.js started. Some related ideas: H5ES that Tencent shared with this group some time ago. Also some TV framework used to create UIs on constrained devices that restrict Web APIs and use canvas-only approaches.

Erik: Is it possible to code if you want or all declarative through the events feature you presented?
Florian: Yes, it is flexible. If you want to, you can inject JavaScript in the blocks that run. We see some people write JS.
Erik: It sounds a lot like Flash used to be. Why isn't GDevelop as big as Flash?
Florian: Maybe we're too late… or too early. The ability to use something that is one level higher than a low level game engine is slowly making its way.
[Some discussion on promoting GDevelop. Name might change in the future.]

Andrzej reports on his wife's happy experience with GDevelop. Shows the ability to develop games without necessarily having coding skills.

Richard: Do you make use of Web workers at the moment?
Florian: Not really for the time being. We are not doing it because users are not asking for this kind of performance yet.

Andrzej: How hard is it to add new Pixi features when they are released?
Florian: Updating is mostly painless.
Clément: More a fix than anything else. It's quite easy to create a game that tests all features.

Discussion on an "export to Phaser 4" feature: possible, as GDevelop is not in the business of developing a game engine.

Francois mentions that he expects local storage issues to be raised during the upcoming Workshop on Professional Media Production on the Web, which could perhaps help with giving a gentle push to browser vendors to do something about it.

W3C Web Games Workshop #2 Virality

Jul 27, 2021

Agenda

- 5mn - Introduction
- 25mn - Split into groups of ~5 people for brainstorming, adding to a big group google slide deck
- 5-10mn - Return as a whole group. A person from each group presents as we walk through the group deck.
- 15mn - Discuss as one large group
- 5-10mn - Try to identify “actionable items”

Attendance

- Noel Meudec - Facebook
- Andrzej Mazur - Js13kGames
- Koen Bollen - Poki Developer Relations
- Erik Dubbelboer - Poki Developer Relations
- Sean Sechang Ahn - Facebook Games Partnerships
- Tom Greenaway - Chrome
- Shufen Lee - Facebook Games Partnerships

Brainstorm / Minutes Deck

https://docs.google.com/presentation/d/13Y_SqQjh7drgc2-kJBSJOyo6VdtQnNE31UyIEijsK8g/edit#slide=id.p

W3C Web Games Meeting June 2021

22 June 2021

Agenda

- Fabio Alessandrelli from Godot will give a talk on Godot's history with HTML5, challenges and future (10 min talk + 5 min Q/A)
- Uchi Uchibeke from Coil will give us the latest news on their work and non-ad alternative monetisation on the web in general (10 min talk + 5 min Q/A)
- Mat Groves from Goodboy Digital will talk about Pixi's history and future (10 min talk + 5 min Q/A)
- Open discussion (15 min)

Attendance

- Mat Groves
- Francois Daoust
- Erik Dubbelboer
- Uchi Uchibeke
- Tom Greenaway
- Andrzej Mazur
- Richard Davey
- Ashley Gullen
- Jianjun Zhu
- Fabio Alessandrelli
- Noel Meudec

Godot - Fabio Alessandrelli

- Fabio: Maintainer of the HTML5 for Godot
- F: Godot is a 2D and 3D game engine
- Powerful built in editor
- Tiny 30 MB download for the editor
- Programmer friendly
- Runs everywhere: native and Web, support for XR. Open license (MIT)
- Follow recommendations from Khronos Group and W3C
- Development is community driven
- Inclusion is important for us: want to support as many languages and communities as possible

- Aims to be easy to learn and use
- Support for more human spoken languages in the world
- 1200+ contributors
- 70+ core contributors
- Godot on the Web
    - Always available, use emscripten, initially through asm.js, now using WebAssembly + WebGL 1 and 2
    - Footprint is ~3MB compressed.
    - Web technologies include WebGL, WebXR, WebSockets/WebRTC as native engine class, allowing support across native and web, persistent storage (IndexedDB), and various APIs.
    - We want to support PWA out of the box with offline support. To improve loading after first load thanks to caching.
    - We want to improve integration with apps
    - Also improve the usability of the Web editor
    - Longer term: WebGPU support
- Some limitations: actions must be run during user inputs (e.g. mouse capture), networking is subject to CORS.
- “SecureContext requirements”
    - many APIs are being broken by security measures?
- Standards / UA issues:
    - Performances
    - Audio quality
    - Main loop issues when page is unfocused (throttling, no animation)
    - No IME support (canvas removed from spec)
    - Clipboard is broken
    - Constantly breaking APIs(gamepads, webaudio, threads, fetch, etc)
- Showing a demo of the Web editor
    - Can’t export a version of the game directly from the web editor, but they’re working on it.
- Showing a game example from a platform called game of the month that hosts Godot games.

Questions

- Secure Context issue? In some cases, you want to try a multiplayer game from different devices and you need to install it to servers instead of simply being able to use a local server.
- Do you have a lot of developers using the HTML5 backend?
    - The HTML5 backend is really used. More games done that way than using the native platform but we don't really have numbers.
- Gameofthemonth: gotm.io
- Performances vs Unity?

    - Don't have performance measurements per se
    - For export size, decent compression is 3MB for the engine, and then you have the assets.
- IS there a way to integrate Godot games with front-ends?
    - From Godot, you can access JavaScript main global context. You can code in JS from Godot if you want, using React or whatever framework you fancy.
- How do you handle changed browser APIs? Multiple versions of the Godot code and detect browser or does it just fail?
    - We hope that SharedArrayBuffer will soon be supported everywhere, basically two versions for now, to be done manually

Web monetization - Uchi Uchibeke

- Will focus on how Web monetization may be applied to Web games
- Web monetization is there to prevent large orgs from spying on customers to provide personalized ads, and rather create a context where money flows from end user to content provider.
- Wallet + Payment Pointer (ILP) + Html meta tag or profile settings
- All of this is made possible through interledger
    - Universal
    - Flexible
    - Simple
- We're planning a Web Monetization workshop
    - Goal is to explore spec changes for the W3C draft proposal
    - Some topics that are explored: whether to use <meta> or <script>, web monetization multipass
    - If websites should know if user can make payment. How informed the user and sites can be.
    - Also want to explore integration in immersive experiences natively.
    - July 28th
    - You may participate by joining the workshop to showcase your game or project, for instance to explore how we can better support game engines such as Godot.
    - You may also participate by joining the showcase session and architecture discussion, one of them being chaired by Marcos.
    - We got a lot of feedback on the spec, there's an opportunity to align the spec with other W3C specs
- Rafiki:
    - All in one solution for interledger wallets
    - Goals
        - Simple payments (pay specific amount)
        - Recurring payments (subscriptions)
        - App-based payments (connecting apps to the wallet, allowing them to make payments)

        - Advanced payments (including streaming payments and other novel use cases)
    - Ideas:
        - Multiple providers
        - Tipping: possibility for one player to send payment to another player.
        - Advanced payments in Web Games extending PixiJS or Godot to include Web monetisation functionalities. E.g. shared revenue model between developer, framework, hosting platform.
    - With Rafiki, it will be possible for any developers to enable sending of funds, to tip or pay friends, send money. For instance, eCommerce platforms would be able to enable users to natively pay through their browser.
    - We're going to launch a test nest by September. You will be able to integrate this in Web games.
- More resources:
    - https://dev.to/coil
    - https://github.com/coilhq

Questions

- How is progress going re. getting this made native in browsers?
    - We have a technical program where we bring people from different communities (e.g. Chrome community) to develop proof-of-concepts. E.g. working with Mozilla community to add to Firefox.
    - Once we have these PoC, we can look at browser developers to support Web monetization. It's a long process.
- How far, roughly, are we from having Web Monetization API as a W3C Draft?
    - Hard to put a timeline and I'm not best positioned for that. Happy to talk about this later.
- How many users, in the wild, are logged into Coil?
    - I don't have the numbers of daily active users.
    - With regards to Web monetized sites, we have some numbers that I'll look into and share.
- How does interledger process payments? You mentioned not needing a bank account? Do you register to a provider (Paypal-like) and then that is used as underlying wallet?
    - Currently, we have two wallet providers. With the Rafiki project, we're looking to not just depend on these wallet providers, but also allow people to develop their wallets.
    - You do need to register to a provider. Hopefully, this will change in the future.

Pixi - Mat Groves

- Open source 2D JS based rendering engine, born 7 years ago. Started by playing with a Canvas. Worked fine, so we built Pixi.

- Only 2D, focused on graphics rendering. Not a full game engine, not meant to compete with other game engines.
- Not 100% focused on games, designed for any complex rendering scene (could be used for web apps or a map interface for example)
- Balance performance and flexibility
- API kept as simple as possible. No crazy WebGL commands, hidden from the developers. API has not changed that much since we started. Backwards compatible, mostly.
- Future
    - Probably going to drop the Canvas renderer and focus on WebGL. We're thinking of creating an optional module for the Canvas renderer instead.
    - Streamlining the API
    - Documentation
        - Want to make it easier for developers to add custom graphics features. These features are added to the engine but not documented well enough.
    - WebGPU
        - Really lovely API, really cool to see it coming.
- Challenges:
    - WebGPU - have found some little issues so far
        - Great TypeScript typing out there
        - Somethings not 100% backwards compatible
        - Shaders for example, even if we use GLSL.
    - Debugging GPU
        - Something that's missing from WebGPU:
            - Shader inspection. Here are all the uniforms, etc.
            - Can be done with the source of the shader but it's more accurate / better when you can do it on the GPU itself
        - The WebGL inspector plugin was great but it broke
        - The BabylonJS guys created a great inspector tool
        - But now with WebGPU, we are in the dark again. Having a GPU equivalent to the CPU inspection tool would be fantastic.
    - Fragmentation
        - Number of IF statements can be different on different devices/browsers
        - No way to detect this.
        - We use a giant IF statement in a shader to detect/adapt. Should be a "no no" but it works very well for us.
    - Multithreading:
        - Making multithreaded renderers easier would be great. WebGPU is enabling that to some extent.

Questions

- Will Pixi try to fallback from WebGPU to WebGPU?
    - The dream is that we'll fallback

    - Just import what you need when you need. Request WebGL or WebGPU depending on what's available
- 2D legacy?
    - About making a clean cut and completely moving it to a plugin
    - Tends to slow things down. E.g. 30% boost by refreshing code.
    - Tom mentions experience with some Chromebook laptops needing canvas2D for Pixi for Google.io. There are still cases, but they should really be edge cases.
- Is WebGPU compatible with the current Pixi API?
    - High level: yes
    - Mid level: no
    - I'm currently learning how to map things. A few changes will be needed probably but we should be able to make it work.
- Who is still using canvas2d? Could it not just be completely removed?
    - Kindle Fire?
- Have you seen any performance gains from WebGPU when working mostly in 2D?
    - Honestly, I haven't gone far enough yet. It's all about the switching of state. I think we're going to see benefits if you have graphs of objects and the like.

Next workshop

We'll run another workshop next month. Noël will send a survey to the group soon.

W3C Web Games Workshop #1

11 May 2021

Agenda

"The web platform has come a long way since the days of Flash. Megahits such as Candy Crush and Clash Royale or indie games such as Among Us are completely achievable on the web today. So… why don't more game developers build games for the web? In other words, what holds back web gaming?"

https://www.w3.org/events/meetings/6159072f-6319-4431-83eb-eb4ca4bddf39

Attendance

- Andrzej Mazur
- Francois Daoust
- Fabio Alessandrelli
- Jianlin Qiu (audio issues?)
- Koen Bollen
- Paul Gadi
- Noël Meudec
- Tom Greenaway

Transcript - by Noël

Points made by participants

Koen Bollen (Poki)

1- Developers don't know what is possible to achieve on web
2- Real time multiple is hard (WebRTC might help, but it is not mainstream yet)
3- The code open for everybody to steal

Fabio (Godot)

4- The API are different from more classic game development (for example: loops don't work the same way) . It makes it hard to port.
5- API are not as stable as they used to be in the past (old games are breaking)

Paul (OP Games)

6- Tooling is not at the same level as native game development
7- Loss of flash made it harder to get on the web, and nothing is at the same UX level. What could replace it?

Andrzej Mazur (js13k game jam and more)

8- Non-technical side: discoverability and monetization. How to make money and reach an audience?
9- The ecosystem needs an equivalent to Steam. There are many publishers but no central place.
10- On a positive note: technically, it is much better than years ago.

Q&A and free chat

Question from **Francois:** how are native games doing multiplayer games? Technical answers by Koen and Fabio (note: unfortunately, a bit too fast for me to take notes)
Fabio: UDP is not directly accessible from Web. WebRTC helps but it is complicated.
Francois: Developers could use Web Transport (client to server) but there is a need for peer-to-peer, and WebRTC is better for that.
Fabio: adding another protocol (Web Transport) might bring more complexity.
(Fabio also raised concerns about API that could stop working after some years, as he experienced it in the past)
(Note we tried to explore points #1, #9 and #7 a bit deeper)

1- **Tom:** There are not enough success stories of games made in HTML5.
Koen: web games are not “cool”, and there not enough high quality games. There is an image problem with web games in general.
Andrzej: we have seen impressive tech demos, but no huge hits following those. And agreed, not enough success stories.
Koen: cross-play would be helpful/motivating (need for a success story here too).
Andrzej: some games were ported to native. Having the same game on Steam brought credibility (woaw effect).
Fabio: tools are super important. Developers are making games, not specifically web games. They just want to be able to distribute anywhere.

9- **Koen:** Poki aims to become the Steam of web. Selling games (like 10$) is not common on web. How do web developer make money?

About monetization: Poki looked at Coil.

Andrzej: monetizing with ads is the only thing that works at the moment. But we are pushing web monetization hard. It is difficult to make it happen (people are not convinced yet). We really need to find alternatives to ads.
Tom: Youtube monetizes with ads, but also with sponsorships and patreons. Can we learn from it?
Andrzej: All the options already exist, but advertising is still 95% of the revenue. The problem with patreon: it is served by a unique company, everyone needs to go through that website and all. Side note: Coil is experimenting with tipping.
Paul: There is no standard for in-app purchase.
Koen: At Poki, we don't want to have to handle our own wallet.
Andrzej: Coil, with the tipping tech, is working on in-app purchase.
Noel: alternatively, what do you think of bitcoin mining while playing as a way to monetize?
Andrzej/Koen: It was tried before but got abused. Maybe it needs refinement.
Koen: There might be a problem of transparency. Asking for money is more transparent by default.
Andrzej: But it is great to bring it up, we need original ideas to challenge the possible ways to monetize.
Koen: We already have examples of working alternatives. Twitch works! They make money from people even if nobody must pay. Huge fans are paying.
Andrzej: Patreon model for the web also has a huge potential. We need to find ways to make it work.
Fabio: The platform itch.io is quite used and monetize well (to pay the developer directly is possible, redirection to the native app is possible, etc.)
Andrzej: yes, itch.io is experimenting a lot (including with web monetization). An asset marketplace also available.
Koen: it feels more targeted toward developers than toward consumers.

7- **Paul:** Before there were a lot of creators using Flash. They are not successful anymore. Why?
Flash was used by artists, that was helping getting more people to make games.
Koen: Web is not seen as a gaming platform, it does not attract "real" developers.
Fabio: When in reality it is just another platform. Game engines should invest on it. People want to make games, but they don't know the tricks of the web.
Koen: It seems Unity/Unreal don't see it as an opportunity.
Fabio: on a side note, 99% of games on Steam are made with game engines.
Paul: so are game engines the problem?
Fabio: it is true that the resulting applications run much worse than native ones.
Koen: In Unity, using"export to web" often results in something not workable.
Fabio: From the game engine side, it is hard because there are a lot of limitations than come with web.
Andrzej: the unity team explained that there are ongoing efforts to make the export better, plus unity tiny is being improved.
Tom: They expected Unity and Tiny Unity tech to converge. Nowot is not clear where it is going, especially project tiny. The main use case for it is playable ads.

It is far from what it needs to be.
Koen: At Poki, we rarely see Unity games perform well unless they are big IP.
Fabio: Godot has a 12Mb footprint on web, but it still too big. Then loading assets usually adds something like 300Mb, it becomes super heavy.
Koen: games on web load assets on demand, which is not the case in traditional game development.
Fabio: True. It is a different tech, making it hard to port. Example: access to cache, etc.
Tom: Game streaming is a brilliant solution for games. But will it succeed?
Koen: The main problem is input delay. It is difficult to avoid it.

Conclusion

Tom: It was a great time. Next time we should try to get more actionable outputs though.
Francois: having a marketing campaign to raise awareness could be a great idea (see screenshot shared at the top of this email).

Some notes - by Francois

Some notes taken by Francois during the call. Covers the same topics as the transcript.

- **Awareness**: game developers do not necessarily realize what they can do with Web technologies:
    - Need to celebrate achievements, a marketing campaign "I can't believe it's not native!":
        - Some kind of hero app
        - More examples of good 3D games
        - More success stories.
        - Games that exist on Steam (because that's where hard gamers are) and also on the Web.
    - Image issue. Web games aren't seen as "cool" to start with.
    - Many demos in the past, not many huge hits after that.
- Hard to achieve **realtime multiplayer communication**
    - In native, developers are used to dealing with UDP packets
    - WebRTC is complex.
    - WebTransport might help, but 1) yet another new protocol and 2) client/server only whereas P2P scenarios exist.
- **Need to write things differently on the Web**. Async example: Clipboard API. Why is it async? Makes it difficult. No sync versions.
- **APIs on the Web no longer as stable as in the past**. Example of the ScriptProcessorNode in the Web Audio API. All games made with that API will need to be updated or trashed as the feature has been deprecated. Shouldn't push developers to adopt a technology that is not stable yet. (or the Gamepad API that now requires SecureContext).

- **Authoring/Debugging tools**. Lots of tooling for Unity, Unreal. Copy and paste is easy. Lot of programming involved on the Web. Tools UX is not on par with Flash.
    - Where are the creators on the Web?
    - Flash was more targeted at artists, designers.
    - Lots of people are not JS/WASM experts
    - Majority of games made with a game engine
    - Web exports are not very good for now. Devs need to invest time to make game run on the Web. Not clear whether technologies are converging (example of project Tiny for Unity)
    - Asset management. Starting point in native is often that assets have already been downloaded.
- **Discoverability and monetization**. URL is fantastic in theory, but hard to point many people to it in practice. No central place where everyone can go. No Steam for the Web. How do I reach the audience and make money? Publishers competing with each other
    - Poki and itch.io are examples of "Steam for the Web". Give initial set of users.
    - People not used to paying for Web games.
    - Alternatives to ads and sales: Web Monetization? But chicken and egg issue as no one uses that for now.
    - Could sponsored web games be an alternative?
    - Alternatives can force lock-in to a specific vendor, e.g. patreons.
    - Ads are still 95% of the revenue
    - In-app purchases? c.f. Digital Goods API.
    - Crypto money mining => bad UX. But perhaps it was badly implemented. Won't work well on mobile devices.
    - Utopia? Example of Twitch streamers that make money. Huge fans pay to watch streamers, others watch for free. Shift in culture. Pay as you wish.
    - itch.io exploring a number of monetization options.
- **Native & Web compatibility**: want to use the same code throughout, e.g. for multi-device communication.
- **Some limitations of the Web platform**. Hard to access the virtual keyboard for instance. Not a lot of involvement from game engines to fill these gaps.

W3C Web Games Meeting Apr 2021

15 April 2021

Summary

The group discussed the following topics:

- **Anthony Bowker** from Unity gave a presentation about Unity's current HTML5 efforts, mentioning:
    - How to generate a web-based version of Dragon Crashers that can be embedded in a canvas element
    - The compilation workflow that runs under the hoods, and the IL2CPP toolchain in particular
    - The roadmap for web features, including focus on smaller download size and memory usage to target mobile devices, and integration with sensors
    - Project Tiny, powered by DOTS, which offers a lower footprint alternative to the HTML5 export
- **Bernard Aboba** from Microsoft, **Paul Adenot** from Mozilla, **Chris Cunningham** from Google presented WebCodecs.
    - What the specification enables in terms of precise control over decoded audio and/or video frames
    - Two main envisioned use cases for games: progressive decoding and zero-copy transfer of decoded audio data
    - For audio, WebCodecs provides a memory-footprint-friendly and lower-level approach to decodeAudioData. WebCodecs does not handle demuxing but community-maintained demuxers in JS/WASM are envisioned.
- **Andrzej Mazur** from Enclave Games shared news about the Gamedev.js survey and jam
    - Survey got >400 responses from 72 countries.
    - Main takeaways: 40% of responses from solo developers. 22% haven't released any Web game yet. 58% checked the "earn less than $1k per year" box, but positive responses for the happiness question overall!
    - Jam session started 2 years before the call, with web Monetization and decentralized as main categories
    - The Media Working Group welcomes feedback on WebCodecs regarding suitability and specific needs in web game on the WebCodecs repo.

The group also discussed the possibility to organize "workshops" during interleaving bi-monthly calls, where participants can join and brainstorm on a specific topic related to web games. We'll try a first workshop on 11 May at 8am UTC. Please save the date, more on that soon!

Agenda

Unity Presentation - Anthony Bowker

See video and transcript.

Presentation notes taken in real-time:

- Unity: Film, animation, architecture, engineering and… games
- With Unity, goal is to build once, and deploy anywhere across consoles, etc. and the Web!
- Web is important to us. 2015 debut for web for Unity.
- I'll show how to create a simple 2D game within the editor
- Walkthrough of Dragon Crashers template/example:
    - First, change the target platform to be WebGL. Then import assets.
    - You can play directly within the editor but also build for the Web
    - That will pop the game open on the destination browser
    - Official sample project that was released recently. It showcases most of the relevant features. Scrolling, shapes, etc.
    - Available free on the Web for anyone to try.
- Designed to be hosted anywhere.
- Anyone can provide their own Web server provided right WASM mime-type is set
- Also many places where you can upload your games, e.g. play.unity.com
- The Unity WebGL target has a basic template. All you need to do is give it a <canvas> element, and then embed in any page.
- Unity is a native C++ game engine, running C# game script
- Different compile steps. On desktop platforms, you can also use the .Net core.
- C# compiler takes source to intermediary language, then we do very aggressive stripping of bytes the game does not use. Very important on the Web. Then we can build a native binary.
- For the Web, we use the Emscripten toolchain.
- That gives us near-native speeds on the Web.
- We've been contributing to the Emscripten project for many years.
    - Recent contribution: Uniform binding. Allows you to lock your layouts in the shaders you're using.
    - IL2CPP toolchain allows us to bring the C# and C++ content to the web.
- Roadmap:
    - Now a public roadmap on the website.
    - Focused on this time on support for smaller download / memory usage to target mobile devices: compressed audio in memory, mobile-compressed texture formats.
    - Integration with the gyroscope, sensors, webcam as well.
    - In addition, faster build times.

    - Currently, we're targeting WebGL, but plan it to also support WebGPU when it becomes available.
- Project Tiny:
    - Upcoming modular runtime powered by DOTS (Data-Oriented Technology Stack)
    - Project Tiny is currently in development. Preview available, but not yet production ready
    - Mobile form factor right out of the box.

Questions:

- What's the link for the public roadmap?
    - New Roadmap: https://unity3d.com/roadmap/unity-platform
    - Can solicit feedback from the userbase.
    - Suggest features, etc
- Thoughts on Tiny vs HTML5 export?
    - Do you expect them to converge?
        - We don't see them as replacing HTML5 export
        - Want to support all types of projects.
        - Tiny is still growing in features/capabilities.
        - HTML5 export has all the capabilities. Rich
    - How does that really work from a third-party / plugin community? Do they support both?
        - It depends on what the plugin does. Today, all plugins are based on the editor because Tiny is not yet out.
        - The developer will have a choice on whether to use the object-oriented approach or DOTS. It will be a design decision as to which road they go.
    - Roadmap? Challenges?
- Unity games are usually quite a lot bigger than normal JS games. Is Tiny the answer to lower file sizes and faster loading times or are you still working on improving this for the normal Unity builds as well?
    - Tiny was created for these super fast loading times. For small games, or interactive ad types of games, that's the right thing to use.
    - Of course, we want to improve build sizes throughout.
- **Mobile browser support was in the "under consideration" section. Is there any timeframe on when mobile browser will properly be supported?**
    - It's in the under consideration section in the roadmap because we haven't announced when it will be supported. Watch this space!
- Between HTML5 export and Dots… you mentioned it's a design decision, I'm curious what drives that?

WebCodecs - Bernard Aboba, Paul Adenot, Chris Cunningham

See video and transcript.

Presentation notes taken in real-time:

- New low-level interfaces to codecs
  - E.g. Mediacodec on android, ffmpeg
- Motivations:
  - Example of existing use case: Streaming that uses codecs
  - Transport could be adaptive streaming over HTTP, WebSockets, WebTransport, or an RTCDataChannel
  - The browser receives it and decodes it. Sends mouse/keyboard/controller feedback to the server.
  - Cloud Game: low latency MSE is used.
- Web codecs fit in:
  - Substitutes MSE. Works with any transport. None of the other blocks change
- Benefits?
  - Transport agnostic.
  - Low latency
  - WebCodecs leverages the GPU which MSE currently does not
  - In scenario where encoding is done on the client, WebCodecs enables access to further encoding options such as AVC.
  - DRM are not currently supported
- You manage the painting of the decoded frames by painting to the canvas yourself

- You call decode as much as you want. When inputs are ready, callback gets called
- Audio decoding for game engines
    - Lots of issues over the year
    - No progress
    - Don’t know how long an asset will take to decode.
    - Need to do everything in one go.
    - 3 minute soundtrack, every needs to be done ahead of time.
    - WebCodecs is the opposite. Very little assumptions. Supposed to let you do whatever you may already do in native, based on codecs supported by the browser
    - Integrates well with SharedArrayBuffer, ArrayBuffer or AudioBuffer
- Use case: Progressive decoding
    - You want perfect sync but you don't want to decode everything at once
    - Very limited amount of data in memory at all times
    - That's going to use workers, very low latency
- Use case: zero-copy transfer of decoded audio data
    - You'd rather minimize the amount of memory use and transfer buffers instead
    - You'll postMessage buffers but that won't create copies
- That's only two use cases that we think are going to be specifically useful for games.
- WebCodecs is integrating with other specifications that provide frames from audio/video tracks.

Questions:

- Time frame on when it will lend?
    - Chrome planning to ship this over summer. Very aggressive timeline, so we'll see.
    - Chrome origin trials available
    - Turning the question around: when do you plan to play with it?
- Will WebCodecs support container formats like WebM?
    - Containers were mentioned in Bernard's talk.
    - WebCodecs is not a demuxing API. We're seeing requests to support muxing/demuxing but that is not currently being pursued. That may change in the future

    - Demuxing can be done in JS with zero-copy
    - In fact, we are now shipping in-process WASM demuxers in Firefox, and it's absolutely dominated by memory fetches, no CPU usage struggle
- And is Safari on board?
    - They said it's interesting. Hard to tell.
- Ashley: It might be a bit of a hurdle to switch from decodeAudioData to WebCodecs, since decodeAudioData also demuxes
    - Yes, it's a change. You will immediately see the benefits in terms of memory footprint. decodeAudioData is certainly not going away, so no rush moving away. Plus it demuxes. We need something lower level because higher level doesn't suit game engine developers.
    - People have been using custom containers or no containers as well.
    - We are hoping that we'll have good quality JS-land/WASM-land demuxers maintained by the community.
    - Muxers is more important, I believe, because broken muxers mean you'll be disseminating bad content on the Web. There's an issue on this.

Gamedev.js survey and jam - Andrzej Mazur

See video and transcript.

Presentation notes taken in real-time:

- Survey ran early 2021
    - 437 responses, I was hoping for 100. The survey itself had 26 questions. 72 countries.
- Results: gamedevjs.com/survey/2021
- Many interesting takeaways: 40% of responses were from solo developers. 22% of all the answers responded that they haven't released any Web game yet. 58% checked the "earn less than $1k per year" box, which is frightening.
- Positive responses in the happiness question though.
- Gamedev.js jam:
    - Started 2 days ago, run through the next 11 days
    - Web Monetization and decentralized are the main categories
    - First one was tested last year.
    - Decentralized is a new topic.
    - Itch.io platform is being used, over 400 participants already
    - I encourage everyone to join
- Decentralized games: Do we want to discuss the topic of decentralization in games? Blockchain, crypto, NFT. Interest is growing in this space.
    - Tom: I think that's an interesting game. We could invite someone for next call.

"Workshops" during interleaving bi-monthly calls

- Idea is to hold a call focused on one specific topic, discuss during the call, e.g. in small groups, brainstorm and come back before the end of the call to exchange.
- Examples:
    - How could the Web platform help games become more viral?
    - How can we protect the IP of web games? Web games are quite easy to copy and paste.
- Crypto seems like a good theme for that too.
- Would all 20 people on this call join such a workshop call? Please come back to me.

Questions:

- What types of ways are people copying games out of curiosity?
    - It's easy to copy and adjust the source code, changing assets. There are some workarounds. I wonder whether there is more that can be done on the Web
- Ultimate goal of the brainstorming sessions?
    - Understand needs and turn into an actual proposal for browser vendors
- Suggestions:
    - WebTransport

Schedule:

- 6:10 – hard cut, move to questions
- 6:16 – transition off questions, move to survey lookback
- 6:20 – end survey, open discussion (remaining Unity questions, workshop discussion)
- 6:30 – hard stop

Attendance

- Francois Daoust
- Chris Cunningham
- Tom Greenaway
- Anthony Bowker
- Koen Bollen
- Richard Davey
- Erik Dubbelboer
- Paul Adenot
- Fintyuan
- Bernard's iPad
- kalai
- Ashley Gullen

- Björn Ritzl
- Shouqun Liu
- Andrzej Mazur
- Xiaoqian Wu
- Paul Adenot
- Jianjun Zhu
- Kemal Ahmed (he/him)
- toretto
- linxiaosong
- Paul_OPGames
- 姚晓鹏
- Jun JIang
- Binh Bui
- Nick Burris
- songshourui.null

W3C Web Games Meeting Feb 2021

09 February 2021

Summary

See summary email.

Agenda

See agenda sent to the mailing-list

- Tencent: Alicia Nie presenting on H5 ES
    - 10 minutes preso
    - 5 minutes q/a
- Google: Andre Bandarra presenting on WebViews, Chrome Custom Tabs and Trusted Web Applications
    - 10 minutes preso
    - 5 minutes q/a
- BBC: Jacob Clark presenting on BBC Children's & Education HTML5 framework
    - 10 minutes preso
    - 5 minutes q/a
- Open discussion

Attendance

- Tom Greenaway (Google)
- Noel Meudec (Facebook)
- Francois Daoust (W3C)
- Andre Bandarra (Google)
- Alicia Nie (Tencent)
- Andrzej Mazur (Enclave Games)
- Björn Ritzl (Defold)
- Jacob Clark (BBC)
- Richard Davey
- Russell Kay
- Siney Pang（Tencent）
- Dominique Hazael-Massieux (W3C)
- Yajun Zhang (Tencent)
- Erik Dubbelboer
- Matthew Atkinson

- Hu Song
- Xiaoqian Wu
- Harry Hao Wang

H5 ES - Alicia Nie, Siney Pang

- Started at 9:03
- Example of live game streaming over WeChat. Hundred of millions of pageviews
- Live broadcasting or short video platforms like TikTok or Douyu
- “It’s clear that game platforms and social platforms need to work together”
    - “That’s why H5 platforms need to exist”
- We wanted to minimise GPU and memory consumption
- We wanted the UX to be consistent with native experiences
- We remove unnecessary features to make H5 ES (Embedded Systems) lighter.
- And we add extra code to make the experience coordinate better with the UI
- The H5 ES Standard, attempts to define a subset of features and requirements:
    - Only specific tags allowed
    - Technical requirements: multiple window panels, CSS selectors, animations, etc.
    - Multi-touch
- A subset of CSS features to force developers to take a more performant approach.
- RenderTarget is not allowed.
- Consistent FPS of 30/60/90/120 fps
    - Render overhead must be under 2ms
- Nested layers: Game UI, H5 UI, Game UI
    - Touch interaction need to penetrate through layers, which cannot be done in regular browsers
- Sharing of resources: page should be able to use existing resources in host environments, fonts, images, etc. Also with extended CSS properties that are not available in regular browsers.
- Basic implementation should be <4MB
- Code size < 4MB
- Runtime memory <2MB
- GPU rendering frame time < 2MS
- Remove eval JS
- “Not a sub-set of H5” (Worth clarifying?)
- React and Vue are supported (fully?)
- Network flow control requirements:
    - Controlled by a whitelist in the configuration
    - Sandboxing of the network that the game can access
    - What is the purpose?
        - JS APIs can call engine APIs, so need to control JS in a sandbox

Questions:

- Can you explain more about the multi-window process isolation?

    - Siney: In the game, each Window is a UI Panel, and each Window corresponds to an independent JS VM.
- Do you allow developers to specify an FPS and lock the FPS of the canvas?
    - Is it technically similar to “request animation frame”?
- Any benchmark that compares performance with WebViews, Trusted Web Activities? Similar to accelerated mobile pages in a way. Is there an intrinsic benefit that can be proved? Or can a WebView or Trusted Web Activity be used to run the engine?
    - Performances of the UI will be almost as good as the engine itself.
    - Performances as good as native engines
    - Tom: Would be great if benchmarks could be shared
    - Siney: Webview hasn't take the running efficiency in embedded systems into account; webview cannot reuse the resources and can't coordinate with native game UI; Last but not least, the memory consumption is extremely huge, usually over 50 MegaBytes in a simple page; And this kind of memory consumption is not acceptable especially game is running at full capacity itself.
- Bjorn: I guess there were some suggested features such as resource sharing of assets, using shaders on divs(?), click through of input that aren't supported using a traditional WebView?
    - Siney: Yes, you can using shader on a DIV.
    - Bjorn: Want to learn more about use cases where this is really needed. When is a traditional WebView not enough?
- Richard Davey: Is the H5 ES Standard published anywhere for developers to see?
    - Siney: No.
- Andre: I'm wondering what the supported set of APIs compares to something like https://cobalt.dev/ ?
- Richard Davey suggested it sounded similar to ScaleForm.
    - Siney: Yes, using H5 standart to implement an embedded UI system.

WebViews, Chrome Custom Tabs, Trusted Web Applications - Andre Bandarra

- Start at: 9:26
- See presentation slides
- Working with things related to WebView since 2015.
- WebView:
    - Lots of use cases: news apps, games, GMail, ads (most of the native ads on Android are actually powered by WebViews)
    - Widely available, since version 1. From a device perspective, pretty much available everywhere.
    - Full access to Android APIs (can communicate with Java or native code via the JS2Native bridge)

    - The cookie jar is specific to each application and separate from the cookies of the web browser and any webpages in the browser.
    - The WebView does not have access to the whole set of Web platform APIs. That's on purpose. Most of these APIs would not make sense. Feature detection may not work as expected. You may end up with code that says that a feature is detected, but the WebView actually does not support it in the end.
    - WebViews are fragmented depending on the version of Android. Updatability issues with initial versions, switch to Chrome after Lollipop. Updatability mechanism has changed over versions, merged with Chrome or separated. About 90% of devices should have an updatable WebView.
- Custom Tabs:
    - Before, let's talk about In-app browsers, using WebView. Given lack of support for some APIs, you will likely run into issues. Custom Tabs are an answer to this.
    - CCT cannot run in full-screen.
    - Powered by Chrome, doesn't run in the application process. Looks like it is in the same app, but it actually runs in Chrome.
    - Supported by most browsers on Android: Chrome, Firefox, Brave, Edge…
    - Shared storage between the origin of the page you're connecting to across both regular tabs in the browser and tabs displayed via a CCT in any app.
- Trusted Web Activities:
    - Very similar to CCT but one important UX difference: it can be full-screen.
    - An APK that uses Trusted Web Activities must include an asset verification file that proves it is verified by the origin it attempts to load. This is why it's trusted and why we can trust showing it in full-screen with the access to the shared storage feature.
    - If the verification fails, the TWA reverts to a CCT (i.e. it shows the URL bar at the top of the screen. Because it's not trusted.)
- Digital Goods API + Google Play Billing
-

Questions:

- Erik: In one of your slides it looked like Chrome Custom Tab is much faster than normal Chrome, why is that?
    - Custom tabs has an API that can tell Chrome to pre-render the page.
- Richard: If Chrome on the device updates, does the version Custom Tabs use change with it?
    - It does. Custom tabs depends on the underlying browser. The developer should use the user's chosen browser if it supports custom tabs.
- Russel: does WebGL work for WebView? Custom Tabs?
    - Yes
- Tom: Custom Tabs and TWA cannot integrate with native UIs?
    - That's right. [Andre going through example of WebViews integrated in native apps]

- Russel: can you install custom functions in either of these scenarios so JavaScript could call app level code?
    - Custom tabs have an API. Not enabled in TWA. Concern that developers may be tempted to use native APIs.
- Russel: or could you just use a protocol handler for this?
    - Andre: to communicate between a Custom Tab/ Trusted Web Activity and the native app, Intents using custom schemas work.
- Francois: Any plan to converge WebView and TWA?
    - Andre: there are conversations around something like WebView that would have better support for web platform APIs, but nothing concrete so far.
- Erik: Do you know if Apple has any interest in implementing Custom Tabs or will it always be an Android thing?
    - Andre: Apple has Safari View Controller, which is roughly the same thing as Custom Tabs. There's not equivalent to Trusted Web Activity

BBC Universal App Platform - Jacob Clark

- Start: 9:42
- Build lots of interactive game experiences
- Ship across multiple platforms:
    - Web and in app stores
- Shipping 350 games
- Constantly in the top 5 apps our category in the app stores
- Platform approach to offline support, push notes, rewards, saved game data
- Custom game engines embedded within the apps
- Lots of custom proprietary code to ensure storage, persistence, etc.
- This ended up being a pain to maintain and explain to developers who had to learn something slightly different. It was super expensive to duplicate a huge number of functionality.
- We decided to migrate to Progressive Web Apps instead. We tried to push the limits of this and see how far we could go.
- For iOS and Android, we needed some wrapping framework to target the application stores.
- We didn’t use Trusted Web Activities because there is “no durable storage on Android” (with TWAs?)
- We still use some custom bits for runtime downloads and durable offline storage.
- Custom “game manifest” spec:
    - Asked S3 for this file and precached the files so they could be run offline.
- PWA gives us the ability to build applications that previously required native components
- Sharing code between Web and "App"
- PWA + Capacitor + React = UAP
- Baseplates contain the code, building on React components, wrapped to native through Capacitor.js

- What we wish was better:
    - Inconsistent durable storage
        - Being able to control when storage is evicted and ensuring it can stay permanent (android especially)
    - Disparity between iOS and Android
        - Haptic feedback
        - PWA support
    - Native apps are hard
    - Lack of consistent browser styles. We could less internally if there were more consistency.
    - Inconsistencies between browsers and devices. Ex: Chromebook.
- Showing example of a game running across devices, including Web, Tesla browser

Questions:

- Is durable storage the only blocker for you supporting TWAs?
    - Effectively yes. If we could reliably store the games for TWAs.
- Follow on: If the files could be bundled inside the app, would that be good?
    - In theory this would be great from the technical perspective but from a business perspective, if the initial download bloats in size then that might deter users from downloading the game: i.e. negative impact to total downloads.
- Francois: For durable storage, would the Storage Buckets proposal address the issue?
- Andre: I'd love to understand better the durable storage requirement and how it differs from persistent storage? (https://web.dev/persistent-storage/).
    - The most immediate requirement: 15 games with 40MB each. Lots of devices ship older versions of WebViews.

Possible convergences between WebView or TWAs?

- Andre: WebLayer proposal
    - Would give the best of both worlds: a flexible UI, webview like, but have all the platform APIs.
    - Would not have shared storage (nice feature when user experience starts on the Web before moving to the "native" app, but probably something that games can make do without)
- Andre:
    - YouTube uses this: https://cobalt.dev/

Tom: UI in canvas or UI in HTML on top?
Russell: in the actual game for portability.
Bjorn: Depends on what you want to do. FAQ for a game would probably be built completely in HTML. Rest of the UI may be using a canvas-based approach.

Tom: Note idea for further call times is to pick the time that fits speakers best.

W3C Web Games Meeting Dec 2020

15 December 2020

Agenda

- Scirra: Ashley Gullen presenting on Construct game engine / editor
    - 10 minutes preso
    - 5 minutes q/a
- Google: Peter Conn presenting on Install API proposal
    - 10 minutes preso
    - 5 minutes q/a
- Google: Tom Greenaway presenting on CDS Adventure
    - 10 minutes preso
    - 5 minutes q/a
- Open Q/A
    - 10 minutes
    - Subtopic: monetization (Andrzej and Danyao?)

Attendance

- Tom Greenaway (Google)
- Noel Meudec (Facebook)
- Kasper Mol (Poki)
- Chris Needham (BBC)
- Jacob Clark (BBC)
- Lars Knudsen
- Nick Burris (Google - Web Payments)
- Richard Davey
- Ashley Gullen (Scirra)
- Andrzej Mazur (Enclave Games)
- Danyao Wang (Google - Web Payments, Digital Goods API)
- Francois Daoust (W3C)
- Peter Conn (Google - Web Install API)
- Pratyush Sinha (Google - Web Install API)
- Binh Bui (BBC)
- Fernando Campos

Construct - Ashley Gullen

- BLock-base Not aimed at technical people
- Technology work:
    - WebGPU replacing WebGL. It's highly experimental, and a complete API. It's still close enough to WebGL that we managed to use WebGPU in the backend without having to make changes to the frontend. Much faster. Exciting to see this taken up by Web browsers
    - OffscreenCanvas. The game engine can run entirely in a separate worker now. OffscreenCanvas is the key piece, but lots of other pieces to the puzzle. Input events passed through postMessage() to worker.
    - Some challenges: Requesting full-screen, can’t be done from a worker for example
    - Lots of APIs are missing for web workers:
        - Web audio
        - Video
        - WebRTC
        - Fullscreen
        - Sharing
    - Lots of exchanges needed between worker and main thread to workaround these limitations.
    - A lot of APIs are limited to user gestures. If you postMessage, you're no longer under user activation. Chrome is working in User Activation v2 to fix this. Would be good to see that in other browsers too.
    - All the other browsers are adding support for web workers but not solving _this_ part (i.e. not borrowing the idea from Chrome for User Activation v2)
    - Web Audio support in Web workers would be great.
    - SVG rendering is a popular request. But hard to do in WebGL. You have to rasterize to texture. If somehow, the browser could somehow do SVG in WebGL, that would be good. Not sure how that could work in practice.
- Platform needs:
    - Cordova / WebView. The store becomes the host, which is often overlooked but very useful.
    - Digital Goods API is ongoing. That is good news. One of the reasons why we have to use Cordova in practice for now. If the solution requires a backend, that is going to affect a number of developers of Construct.
    - Showing the AdMob prompt for European users (GDPR). iOS 14 has a separate prompt for the ID for Advertisers. For all these reasons, it seems better to focus on in-app purchases using the Digital Goods API than on ads.
    - Trusted Web Activity is interesting although not fully had time to exploit that so far. It still requires Web hosting.
    - Cordova will probably still be needed for the foreseeable future.
    - Want the store to still host the files for you.
    - And for there to still be an escape hatch for remaining gaps.

Questions:

- How much faster is WebGPU vs WebGL?
    - 3 different strategies for uploading buffers to the gpu
    - Especially faster in difficult cases
- Limitations for the backend
    - A third party service could solve that perhaps. The store takes 30% of the developer's money. Perhaps it would only be fair for it to handle that for the developer.
    - Tom: Part of the problem is technical. Linked to Trusted Web Activity.
- Web Audio has audio worklets and there were discussions on Audio Device Client. Which direction would work?
    - Most important bit is to be able to call the API from a worker. Accessing AudioContext.
- How long do you imagine until WebGPU is suitable for live use?
    - (Ashley) Best to ask browser developers/spec writers - I'd guess probably a year away
- Do you think the ideal end situation is to run Construct 3 completely in a worker? How do you currently handle browsers that do not support offscreencanvas?
    - (Ashley) For performance I think it's best to run all code in a worker. If the browser doesn't support that it just falls back to running entirely in the DOM. We also use a MessageChannel in the DOM so we can keep the same postMessage code.

Web Install API - Peter Conn

Start time: 17:55 - 18:05 + q/a 5 minutes – end 18:10

- Web Install API allows web site to request installation of another website
- https://github.com/PEConn/web-install-explainer
- Two main use cases: Web app directory. And single page for a vendor of web apps.
- Better discovery, democratizing installation on the Web.
- Also split apps per domain, e.g. for storage.
- There are some concerns. We need to gate the installation. Protect the user from the catalogue, from the target website (we want to make sure we're consistent), and the target website from the catalogue (want to avoid fraud! malicious websites. Maybe an allowlist would help here). Require the website to be served over HTTPS and to have a Service Worker.
- UX examples: seamless installation onto the user's home screen. The app goes fullscreen and so the user doesn't see the URL of the site. A browser prompts the user to install. A "try before you buy" feature lets the user return to the store easily if the user doesn't like the website. It's to allow the user to explore a store catalog.
- Open questions: Should the directory know if the user went through with the installation? Should the directory know which apps are already installed? Concerns around fingerprinting if the directory can query for installed games/experiences

- Pratyush: We're excited about this for web games. Game stream platforms support web apps, more are coming.
- Showcasing possible UX for games installation: seamless install, try before you buy.
- What else could we offer for gaming? Payments could be an example,
- Peter: Digital Goods API. This allows a website running in a TWA to query details from a store such as prices. The Payments Request API would go through the Play Store to get payments through play billing. If the site installer has its own payments system setup, the browser could transparently ensure that that store's payment method is always used, if the app was installed from there.
- Pratyush: We want to hear your feedback on this, and anything else.
- Digital Goods API explainer: https://github.com/WICG/digital-goods/blob/master/explainer.md

Questions:

- Francois: Any views from people who already run a game store? Is this going in the right direction?
    - Kasper: We're investigating PWA. One hurdle we have is a problem installing PWAs from our main user facing products as our games are running eg in an iframe on a different domain - can't install a game running from a subdomain on the main domain.
    - Challenge with installing a PWA that is running inside an iframe
- Explainer:
    - https://github.com/PEConn/web-install-explainer (has link to slides)
    - Recording of a previous talk from TPAC 2020: https://www.w3.org/2020/10/TPAC/breakout-schedule.html#web-install
    - Peter's BlinkOn talk on Install API: https://www.youtube.com/watch?v=RdaZhX7T6to

CDS Adventure - Tom Greenaway

- Project I worked on in the past few months and launched last week during Chrome Developer Summit, which was entirely virtual this year.
- Re-imagine physical conferences in a virtual way, ended up thinking about it in terms of games. Games mapped needs pretty naturally, especially with a real-time mulitplayer game.
- I used Construct to do a quick prototype in about a week. Really useful to explore ideas I wanted to implement: bumping into people/things to interact.
- For the final project, we wanted to open source it and re-implemented things from scratch: Docker, node.js for the backend, redis for Websockets, socket.io (although some issues with performance on the server) and Firestore.

- We had a login server, moderation server, docker running on Compute Engine. Quite complicated in the end, but worked well.
- I think the UI came up very nicely. It does not feel like a HTML and CSS "page". It looks like a real game.
- We used pixi.js: pixi-layers to handle z-sorting. Object pooling and was surprised for that not to be supported by default.
- Feedback has been positive. Nice experience on the Web.
- https://goo.gle/cds-adventure

Questions

- How many people played the game?
    - No figures we can share yet. But we designed for 1000 users per server
    - Based on the real user interaction, we think 1 server could handle 2000 simultaneous users
- Why did you go with React?
    - Mostly because the Web agency (Set Snail) was quite familiar with it.
    - Worked really well.

Web Monetization - Andrzej Mazur + Danyao

- Coil is working on some requested features: being able to track payment from specific pages (right now, you need different payment pointers). They are also working on other features.
- Grant for the Web is planning a games specific CfP next year. It may not happen, but if it does, it would be great!
- That's it for a short update.
- Danyao:
    - Work on the Web Payments team in Chrome, work closely with Peter on Digital Goods API. Game developer community seems an interesting one, although I note I'm not a gamer.
    - Encouraging to see Ashley mentioning the Digital Goods API going in the right direction. Web developer needs a way to monetize their game. We just want to bring that primitive to the Web as well. It's more than just payments. Asset management, etc. It's more than about money.
    - Not enough people on the Web are talking about this. If we can spread the word and fine-tune the proposal accordingly, that would be great! If you are interested in trying out, we're eager to hear your feedback and move things forward.
    - Please reach out!

Questions

- What's the timeline for the APIs?
    - Digital Goods API will be available in TWAs in Chrome 88
- Does the Digital Goods API work in the Android WebView in a Cordova app?
    - Currently it doesn't
    - And we don't have a timeline for it either.
    - WebView is more dated architecture compared to Chrome, so technical challenges
    - Ashley: If you already have a Cordova app, it would be much easier to switch the API instead of going through TWA.
    - Tom: Theoretically, I suppose that people could make a polyfill in Cordova.

Games CG - TPAC 2020
Meeting notes

20 October 2020

Agenda

See email sent to public mailing-list.

- Welcome from Tom/Noël (5 minutes)
    - Talk from Noël + Q/A (20 minutes)
    - Talk from Richard + Q/A (20 minutes)
    - Talk from Andrzej + Q/A (20 minutes)
    - Talk from Tom + Q/A (20 minutes)
- Group Discussion (30 minutes)
    - What are the concrete materials that we want to get out of our group long term?
    - Vision or plan for how we see working with other groups?
    - Guidance on how other people can help us?
    - How can we track things as a whole group?
    - Can we create missing topics and reach out to people?
- Wrap-up (5 minutes)

Attendance

- Noël Meudec (Facebook)
- Tom Greenaway (Google)
- François Daoust (W3C)
- Andrzej Mazur (Enclave Games / js13kGames)
- Kasper mol (Poki)
- Erik Dubbelboer (Poki)
- Ashley Gullen (Construct engine / Scirra Ltd)
- Björn Ritzl (Defold)
- Martin Alvarez (Fundacion CTIC)
- Yajun Zhang (Tencent)
- Binh Bui (BBC)
- Geoff Gustafson (Intel)
- Lei Zhai (Intel)
- Chris Needham (BBC)
- Takio Yamaoka (Yahoo Japan)
- Richard Davey (Phaser)
- Johnathan Chetwynd

- Alexandre Gouaillard (CoSMo Software Consulting)
- Geun-Hyung Kim (Gooroomee Co)
- Wenli Zhang
- Matthew Atkinson (The Paciello Group)
- Florian Rival
- Alicia Nie (Tencent)
- Harry Wang (Tencent)
- Xiaoqian Wu (W3C)
- Lars Knudsen

Time Keeping

Topic	Time Actually Started
Welcome from Tom/Noël (5 minutes)	12:00 CET
• Talk from Noël + Q/A (20 minutes)	12:05 CET
• Talk from Richard + Q/A (20 minutes) Phaser, Engines deck, video	12:21 CET Until 12:41 or 12:45
• Talk from Andrzej + Q/A (20 minutes) Monetization deck, video	12:43 Q/A: 12:53 Until 13:05
• Talk from Tom + Q/A (20 minutes) Discovery deck	13:05
Group Discussion (30 minutes)	13:
Wrap-up (5 minutes)	13:

Welcome

Tom: Welcome. Track support for games on the Web. Both local games and cloud gaming.

Noël on Facebook Instant Games

Noël: Intro: Platform engineer in Facebook Instant Games. I was on the jury of the last js13k (awesome event!). I've been in touch with a number of developers and game engine makers, related to instant games.
… I'll give you a very short intro about Instant Games, HTML5 runtime for games in Facebook. It started with an Emoji, the basketball emoji. Someone thought it would be great to launch a basketball game when you click on the emoji.
… That led to instant Games launch in 2016. Added monetization in 2017 and made it an open platform in 2018.
… Some features: When players are playing, no need to log in since the player is already logged in in Facebook. Unique idea across games. Also access to friends.
… Posting to share async
… We have bots, so you can do some story telling or remind people of events.
… Leaderboards API allows to maintain scores. Backend managed by Facebook.
… Matchmaking allows to find friends.
… Cross promotion allows you to switch to any of your other games at any time

… Homescreen shortcuts make it easy to add your game to the home screen.
… Ads include rewarded video and interstitial ads.
… We also support in-app purchases through a set of APIs. Not available on iOS but available on the Web and on Android.
… And of course, we have analytics.
… A few words on monetization, we only support interstitial and rewarded videos.
… You start by preloading the ads, and then display them later on.
… How the platform works in general: every platform will have a container which will contain the Web view, which will embed the game. The game has the SDK that connects to the platform.
… We're using WebView on embedded devices, and iframe on the Web. postMessage for communication.
… On iOS, we're using a message handler to enable communications.
… The API is different on Android but that works exactly the same way otherwise.
… On Desktop, we're simply using postMessage.
… [shows slide with key message loops]
… Check the links on the slides

[Richard Davey] Are there any stats you can share re: game successes? (play counts etc)
Noël: I cannot share stats today. As soon as they are available, I'd be happy to share. The public blog will contain stats.

[Björn Ritzl] What are the latest trends in terms of Instant Games? Popular games?
Noël: I don't have a list of popular games right now, but I'll have a look

[Erik Dubbelboer] Are there many users/games using in-game purchases?
Noël: The limitation is that it's not available in iOS. The most common way to monetize right now is using rewarded videos. We do have success stories with IAP on Android and on the Web. Not the main way to make money.

[Ashley Gullen - Construct/Scirra] Google are experimenting with a standardised API for in-app purchases (the Digital Goods API). Could Instant Games support that to improve the use of cross-platform IAP code?
Noël: That's a very good question. I should escalate to the dev team. Maybe there is something that instant games could do for standardized IAP.

[Binh Bui - BBC Games] Is the add to desktop functionality done via PWA? If so do you manage service workers centrally? (e.g. add to homescreen)
Noël: To be honest, I don't know exactly how it works. I'll check and report later.

Richard on Phaser and web dev feedback

Slides and a pre-recorded version of Richard's talk with a full transcript are available.

Richard: I run PhotonStorm and have been developing Phaser in the last few years. I want to talk about that and about pain points from Web game developers.
… Back in 2012, there was no choice for a game engine on the Web, so I had to build one.
… Very strong community and ecosystem around it.
… I work full time on it, thanks to the community.
… It's been used to create thousands of games. That never ceases to amaze me.
… Phaser has evolved a lot in the past 7 years. The Web was a different platform back then!
… For games, we could use canvas and CSS.
… Canvas support is vitally important.
… The way that developers create games has also changed. These days, we need to support a wider variety of workflows.
… Evolution of Phaser roughly matches the evolution of browsers.
… Platform evolves fast, and long gone are the days when developers can develop a game and be done with it.
… I've been collecting feedback from game developers. Platform, tooling and monetization. I'd like to quote some of the feedback we got.
… [going through quotes, check slides]
… Top concerns are an unstable platform. Inconsistencies. While the web is based on standards, game developers realize that it is fragile, and unstable since it keeps evolving. That makes developers a little bit wary. Building a game is not like building a Web site. APIs may be the same but that's where things start to differ.
… Even tools like Unity and other libraries cannot help hide discrepancies and evolution of the platform.
… The tooling is problematic too. Some of them like visual tools, which can be found (Playcanvas, Construct). Debugging is an entirely different story. Dev tools, while great, are focusing on web sites, not on the needs of game developers (like shaders), which makes sense. Logging data or inspecting value at runtime is hard.
… When you combine an unstable platform with poor debugging tools it gets complicated very quickly.
… Making money from their web games is a big problem.
… Steam and App Store help for native games. For the web, the lack of a central discovery area undermines web games.
… Ironically the platform which enables anyone to make a game that can run anywhere is not good at helping anyone find those games.
… Games being played on the web are much more casual actually.
… Developers are not targeting the web ultimately because there is no way for them to get money from it. There are initiatives taking place to improve the monetization story, focusing on other problems will not be enough to bring developers to the web.

[Tom Greenaway - Google] About dev tools and debugging. What you're saying is that they are focused on HTML/CSS, not focused on games
Richard: Not an instant snapshot. Dev tools are not good at throwing a bunch of stuff without pausing the game. Different problem space, maybe that requires different solutions.

Tom: Dev tools have matured a lot, but you're making a good point. I've had debug situations with different canvas elements, some of them hidden, and I'm trying to see the graphics context. When it comes to shaders, that's even more tricky. Would a way to inspect everything that goes to canvas, WebGL, offscreen canvas be good?

[Ashley Gullen - Construct/Scirra] Do you find graphics driver issues are a problem in Phaser? They seem to be the cause of many graphics problems that come up in Construct

Richard: Yes and no. Tools will get in the way. There are tools like NVidia ones that manage to do that without affecting performance too much. Our shaders are basic so not a real problem. But then there are things such as audio codec strings not supported on some devices. There are lots of APIs to focus on.

[Erik Dubbelboer] Do you think browsers should play a role in helping with the monetisation?
Yes I think so.
Equivalent, mobile purchases for IAP.
The phone is helping with the purchases.

Tom: There is a Web Payments group and a Web monetization.

[larsgk] Have you considered how to best utilize Screen Fold API?
No, not yet.

See the angle that the device is folded at?
Sort of thing that could be in the realm of plugins.

[Noël - Facebook Instant Games] What are current web gaming platforms missing to become the Steam of HTML5 gaming?
Tom: I suppose discoverability and monetization are the key ones.
Richard: Yes, also the distribution. It's ironic since the Web is about sharing.
… It's easy for the Web to break, which is unfortunate. Compared to Steam, where the game will basically work forever.

[Francois] Are pain points documented somewhere? Would it be useful to collect them?
Tom: That's a good question. You probably discussed that with browser vendors and companies.
Richard: Yes, it's hard to identify e.g. audio issues.

Andrzej on Web Monetization

Slides are available, see [Introduction to Web Monetization in HTML5 Games](#).
A [pre-recorded version of Andrzej's talk](#) is available too!

Andrzej: I would like to talk about Web Monetization specifically in HTML5 games. Some of us met in the workshop on Web games last year. I was giving an indie perspective on web games. I was comparing 2011 vs 2019, because we had had a previous games workshop in 2011. We had 20-30 games total back in 2011.
… It evolved over time. Many smaller pain points appeared in 2019. Two issues that I noticed the most were discoverability and monetization.
… Interestingly enough, Coil was present at the workshop and approached me about the Web Monetization API. That API really clicked for me and I was happy to dig into it.
… In the following JS13KGames, we added a Web monetization API category. People were eager to try that. The overall sentiment was that people were very excited about that possibility to literally make money by having people visit your game.
… We had a MozFest Arcade event later in UK. It was really cool to see people getting some money in real-time.
… That was also the time when the Grant for the Web effort was announced. $100 million grant. Coil, Mozilla, Creative Commons.
… It was good to see that different folks got some grant for Web games, including Defold
… I was really happy to see that Phaser also got some of it.
… Getting some momentum.
… So what is Web Monetization? A proposed JS API which allows the creation of a payment stream from the user to the application developer. To enable it, you need to add a <meta> tag that points at a payment holder. If you do that, you'll start getting some money.
… Currently, you can create a wallet through uphold and Gatehub. More companies will offer wallets in the future.
… And then there's Coil who is the first company to offer membership. Paying subscribers. Right now, you can pay $5 / month and have access to all monetized content.
… How does that work? The app can check through JS that monetization has started. We can detect that monetization starts through an event handler. From there on, you know you get money, and decide what extract content you may want to propose. You have full control over what you are doing and offering.
… Another great thing is that you can ditch adverts and offer a better way to monetize content. You don't need to worry about privacy of users, spying of users, data selling.
… Compared to the "classic publisher" scenario, which is complicated, with changing code, Web Monetization is just 1 line of HTML and 2 lines of JS. In terms of income, you're being paid instantly in real time, even if it's small streams of money.
… We finally have the "native" alternative to other options. I really believe that this can work.
… I'm going to invest my time in exploring possibilities and options.

[Erik Dubbellboer] Are there any statistics on how many people are using monetization?
Andrzej: One person, I think, collects some info about that. A few thousands of web sites so far. Super early for now, but we see the growth.

[Kasper Pol] I am aware of this twitter bot that shows a graph of web monetized sites, that's not users and I'm not sure how accurate it is.

[Richard Davey] Do you think being linked to wallets and crypto currencies will "scare" away regular people?
Andrzej: That's a good question. It doesn't have to be a crypto currency. This is amenable to any currency. Totally independent from someone who is sending the money to someone who is receiving. I don't have to worry about it, I will receive the amount in my own currency.
… The wallet can be used to send the money to the bank account.
… Some people may be scared away, but the whole thing is done with the Interledger protocol, nothing to do with crypto currencies.
… Again, automatically converted

[Björn Ritzl] But it doesn't have to be a crypto currency though? We get paid in SEK (Swedish Krona).

[Erik Dubbelboer] Have browsers shown any interest in implementing this API instead of users having to install a plugin?
Andrzej: There is already a proof of concept in the Puma browser that implements the Web Monetization API by default. That's a mobile browser. For desktop, you still have to install different plugins. There was an effort to have native support in Firefox Reality in VR headsets, and I would love to see browser vendors picking that up, when the specification becomes a W3C standard.

[Noël Meudec] Any concern about fraud, money laundering, etc?
Andrzej: I don't think so. People may find a way to bypass that. The API is so privacy-focused that, other than the sender and the receiver, you cannot find any information. I bet there could be something around it, like not directly related to the Web Monetization API.
… It's still early version. Until now, it was only sending a fixed amount of money per second. There are more options that are being explored.

Tom on Web Games Discovery

Slides are available, see [Web Game Discovery].

Tom: Games lead for the Games dev relation team at Google. Before Google, I developed games on native platforms such as Duet. I know what it takes to succeed with games. I have a crush on Web games.
… I'll touch a bit on cloud gaming, but starting with focus on HTML5 games.
… What are the biggest barriers to bringing more games to the Web? I asked people around and got some technical feedback. Asking more broadly, feedback was "discovery" and monetization.
… A lot of platforms are using HTML5 to ship games.
… I think that there are things that we can do to improve the discovery the of games, but not only. We can look at what these platforms improve and see whether we can do that in the open.

It's also about rediscovery, equivalent of installability (where the web platform can do a good job at providing an equivalent). Also load time and offline access as well.
… Can we align on the best practices for web games for discovery?
… I collected 3 different standards (well, 2 + 1 that could be reorganized). First one is schema.org web game metadata. That's the metadata that gets sent when you share something on a platform.
… Another angle to enhance for rediscovery and installability is to double-down on PWA and add to homescreen concepts.
… Lastly, load time can be really improved if we pre-fetch game resources, which would enable offline access. File manifest and Web packaging are promising there.
… I'm hoping that we can find a balance between the open web and game platforms.
… I just published a small proposal on GitHub, called Open Mini Games (OMG)
… That URL should be available right now.
… As a bonus topic, I'd like to talk a bit about cloud gaming. Fun fact is that we noticed that there has been a peak of usage of the Gamepad API during lockdown. More companies are entering the cloud gaming area. Can we also improve the UX of finding games across these streaming platforms?
… I think that the Web games metadata could highlight that it runs server-side.
… We could then provide a catalog of streamable games. The metadata could include a URL for a demo URL, and whether a subscription is required or not.

[Kasper - Poki] On the loading stuff. With JS development, we moved to bundling and that's actually detrimental sometimes to loading time. If we put game engines through CDN for instance, they would be downloaded only once
Tom: There is plenty of potential to have a CDN of popular game engines and libraries
… You could control which resources you need and download and leverage CDN.
Kasper: Modern JS development works nicely with importing modules. Bundling makes it easier to use.

[larsgk] It might be a bit too specific to add game details in manifest - but perhaps there is room for extensions if it could be made a bit more generic (and needed across other types of apps, perhaps)
Tom: With the PWA manifest, I wasn't proposing to extend that. I'm thinking on extending schema.org. I created an issue around that. We can continue to discuss that on the OMG repo as well.

[Ashley Gullen] It sounds like there might be a bit of a chicken-and-egg problem getting people to use the schema - i.e. until some big service makes use of it, there might not be much value to adding it. Do you have any thoughts on how to get around that?
Tom: It is a chicken-and-egg issue. That's way it seems preferable to focus on standard. We could start with PWA elements. An important point about schema.org is that many game platforms implement it, but they implement it differently. Some people say that their game is an app. Some say that their operating system is the browser. The properties are loosely defined. Standardizing that would be good.

[Andrzej Mazur] Would talking to search engine dev teams help expose the info from Schema (of specific games implementing them) in the results, and ultimately help with discoverability?
Tom: I can communicate with the search team. I already touched on that in previous question.

[Binh Bui] Really cool stuff - very interested in things like the open mini games format. Installability/Add to homescreen is of particular interest. Do we know how receptive the audience is to this journey? Will casual gamers use it?
Tom: Add to homescreen will help move the needle, but it's not perfect. Different platforms can handle that differently. Casual gamers use that less than other types of gamers perhaps.

[Lei Zhai] On Cloud Gaming part, how will WebGame CG collaborate with other technology support groups including WebMedia WG/WebRTC WG/WebTransport WG? Mainly on Game Discovery part?
Tom: I don't think that's super specific to me, so let's tackle that in the open Q&A

[Kasper Mol] Also about the schema, is just having more metadata the complete solution? Because even if for example a search engine knows everything about a game, the game still needs to build SEO value from somewhere in order to get surfaced. So marketing is still a necessity.
Tom: There's finding the search result, and there's presenting the search result.

Group discussion

On Cloud Gaming part, how will WebGame CG collaborate with other technology support groups including WebMedia WG/WebRTC WG/WebTransport WG? Mainly on Game Discovery part?
Francois: depends on what the group wants to do. “It starts with you!”

We should collect things perhaps.
If you have a problem, you need to document the problem where it starts and the need.

Better than just an individual coming forward with a problem.

Breakout session on cloud gaming.
Lei Zhai: makes sense to focus on discovery.

Noel: It could be on a forum hosted by Richard:
https://www.html5gamedevs.com/forum/40-web-gaming-standards/

[Kasper Mol] I would really like to understand the discoverability problem more, because that keeps coming up as the big thing.

For example I'd be interested to know if there's a difference in the discoverability challenge for web game devs compared to for example a steam dev. A steam dev also has a discoverability problem, but I don't think we would blame the platform, we would blame their lack of marketing + a saturated market.
Tom: That's a good point. There's just more that can be done through a Web browser. There are better ways to put something forward to explain what is this thing that the developers have put forward.
… Platforms like Steam have features that result in better user experience. If there were standard metadata, platforms would gain confidence that they can find these game and add them.
… Platforms may not support all features.

Richard Davey: Steam has wish lists, user reviews, recommendations, sharing games between friends, achievements (hats!) etc - not just the library, so you definitely still need marketing, but wish lists convert to sales, for example, which we don't have on the web (you work for Poki, right? so I guess you've a lot of experience of providing services like those for players?) - I think this is what devs mean when they say they struggle with discoverability

Kasper: Features such as recommendations. I want to challenge the challenge.

Ashley: Is Apple going to be on board? That is a good question. Is there anything we can do to get them to participate in the group?

Francois: I'll try to reach out to contacts there. Don't have "game" profiles though.

What are the concrete materials that we want to get out of our group long term?
Tom: I think one long term goal is to have a clearer relationship with Apple. If anyone can help, that would be great.

Lars Knudsen: Maybe John Davis would know (potential entry point for apple). Maybe it could be an idea to contribute to a 'game dev' section on MDN, where many devs (indie and pro) go for info.

3 possible concrete deliverables:
- Stability and debugging. Dev tools.
- Discovery
- Monetization

"Maybe it could be an idea to contribute to a 'game dev' section on MDN, where many devs (indie and pro) go for info"

Mailing list can be good for announcements.
But for discussion we can use the HTML5 gamedev forum.

“We want people to feel free to post about their pain.”
Problem of the discoverability of the issues for the community

GitHub can be useful to track specific points.

[Ashley Gullen] On platform quality/bugs, I could make a list of bugs (across browsers) that I've filed in the past relating to improving consistency and general DX, and I'm sure many others have their own filed issues too - not sure if coming up with a list like that would help?
Tom: We could also use something like a Google spreadsheet to coordinate pain points, issues, and people. Once I have feedback, I can for instance take it internally and push for coordinated action around Web games.

[Lars Knudsen] Also - like the Fugu tracker - a (meta) tracker could be made for APIs that care relevant for games (there will be a lot of overlap)

Francois: Offer to maintain the roadmap on Web technologies for games that I had prepared for the workshop

Noël: I'd like some way to track what happens in the community. Someone should volunteer to track this.

Tom: I was also going to suggest that we have a recurring call for this CG.
Noël: That seems good. Much more engaging to see faces.
Tom: How about a monthly call?

Tom: On the discovery side, I'm happy to tackle that in the HTML5 dev forum. It would be good for others to volunteer for other topics.

Tom: I can coordinate a spreadsheet of the technical issues.
Ashley: happy to work on managing a spreadsheet of issues with Tom.

[Lars Knudsen] I may not have full time to drive it - but would be happy to help with sparring/going though all relevant APIs (Fugu and more) + drag in Kenneth Christiansen (Generic Sensors, PWA, NFC, devices & sensors, ...)

Noel: We should have a tracker for our actions/tasks. And iterate on that each month.

Andrzej: Happy to help with monetization. One thing to do is to check with Coil about the standardization status of the API.

Tom: What is special about game monetization vs. other areas. Maybe we could dig into this more deeply in our next monthly call.

Richard: Is a valid action trying to get more people from other platforms here? For example someone from the Chromium WebGPU team, or WeChat, etc (sorry if you are here already!) but it feels like education would be a useful goal? I would love to hear their thoughts in future meetings

Francois: Happy to reach out to people. I note Tencent was present today, at least at the beginning of the call. Inviting people from companies of interest to come and present can be a good to engage them in the work.

[Discussion on monthly. Idea is to do bimonthly calls. Next call in December]

How can we track things as a whole group?

Task tracker, monthly sync.

[unanswered questions below]

Can we create missing topics and reach out to people?

Vision or plan for how we see working with other groups?

Guidance on how other people can help us?