A Complete Guide to WebGPU and WebGL Technologies

No comments

For rendering 2D graphics and interactive 3D graphics on the Web, WebGL (Web Graphics Library) is the new standard. WebGPU-technology enables data-parallel computation on the web and high-performance 3D graphics. This article discusses the differences between WebGPU and WebGL and how to hire the best WebGPU and WebGL Website Developers to build your 3D website.

WebGL-Technology

The WebGL (web graphics library) is an application programming interface (API) for implementing interactive web graphics in JavaScript. No external plugins are required to run GPU-accelerated graphics within an HTML canvas. WebGL is a set of functions that draw vector elements on a browser screen using the client's graphics processor (GPU). Its uniqueness lies in its ability to render visual elements of high quality and complexity. Indeed, no HTML or CSS method can achieve a similar impact. The WebGL technology, developed by KhronosGroup, is a direct descendant of OpenGL ES, used for 3D visualizations in games and VR. WebGL also uses for 3D web design, interactive games, physics simulations, and data visualization.

WebGL Pros and Cons

A complete understanding of the pros and cons of different technologies is essential when selecting the right tech

stack for your needs.

WebGL Pros:

WebGL has the following essential advantages:

1. Most desktop and mobile browsers support it natively and for free.

2. The technology is relatively fast in terms of speed and compatibility with HTML.

3. You don't need to install anything on your device to use the WebGL API.

4. To create outstanding 3D web experiences, you can leverage external libraries.

WebGL Cons:

WebGL has the following cons:

1. Native WebGL programming has a steep learning curve

2. Despite the vendors' claims, it has security vulnerabilities

3. It can cause browser crashes due to plugin incompatibility, etc.

If you want cross-platform and cross-browser functionality, seamless interaction with all the elements of HTML, and seamless integration with WebGL, then WebGL is the way to go.

WebGL Libraries

Web developers can build web products using a variety of WebGL frameworks and libraries. Applying prewritten elements can speed up the development of web products significantly.

Unity WebGL

If you're searching for WebGL resources, Unity WebGL will probably be one of the first libraries you'll find. Unity WebGL allows web developers to interact directly with the browser's JavaScript engine. Unity WebGL offers several methods

for doing this, including calling Javascript or C functions from Unity scripts or even sending data to Unity scripts from the browser's JavaScript. Most major desktop browsers currently support Unity WebGL content. Mobile devices are not yet supported.

Three.js

In Three.js, you can find prewritten JavaScript elements that can be used for WebGL. In addition to effects, objects, cameras, scenes, materials, shaders, and other utilities, there are still manuals in development. Still, a large developer community on GitHub hosts forums and discussions.

Babylon.js

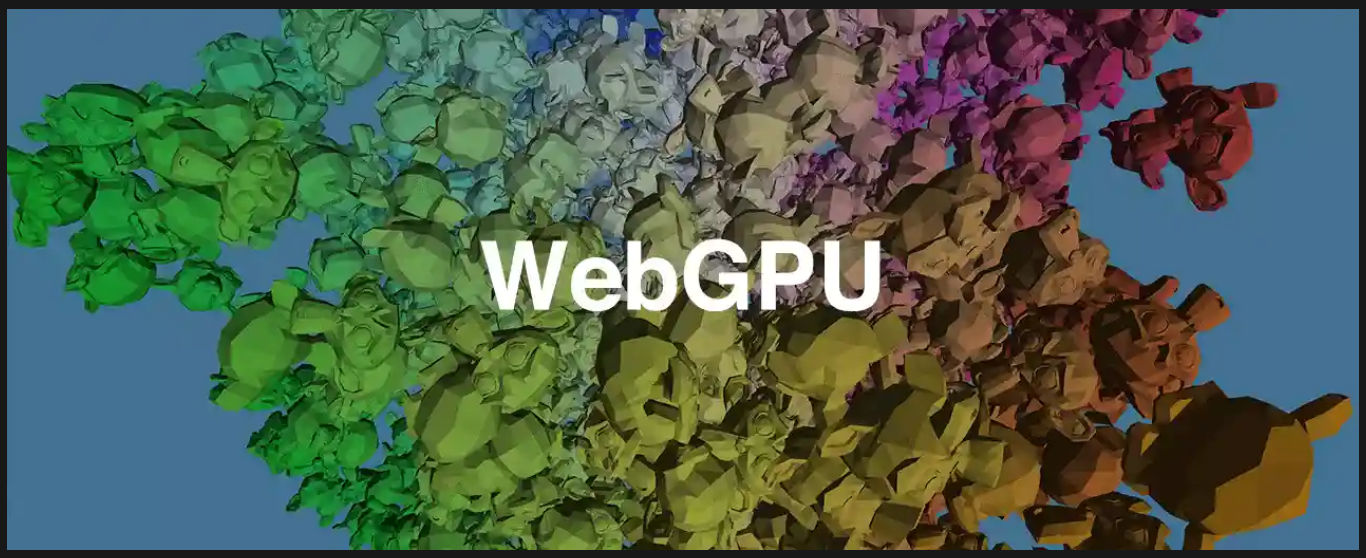
One of the most popular open-source libraries, Babylon.js, is a browser-based 3D graphics engine that is written in Typescript. However, it can be interpreted natively by all browsers that support WebGL and HTML5. In addition to gaming and crime data visualization, Babylon.js has several other applications, such as fashion, healthcare education, and military training.

A-Frame

It is intended for displaying 3D and AR/VR elements in a browser and AR/VR experiences. A-Frame is a VR plugin that includes animations, geometries, cursors, controls, etc. The A-Frame library generates code for VR headsets, as well as desktops and mobiles. This makes A-Frame a perfect library for making browser games. It is also available on GitHub.

WebGPU-Technology

With WebGPU-technology, you can perform rendering and computation operations on a graphics processing unit (GPU) using modern graphics capabilities, specifically Direct3D 12, Metal, and Vulkan. This goal is similar to the WebGL family of APIs, but WebGPU-technology gives you access to more advanced features of GPUs. WebGPU-technology provides first-class support for general computations on the GPU. In contrast, WebGL is primarily used to draw images but can be repurposed for other computations with great effort. Developers can now try WebGPU in Chrome after four years of development in the W3C's "GPU for the Web" Community Group.



WebGPU Advantages:

WebGPU-technology provides several benefits (some of which are shared with other new low-level graphics APIs):

1. There is actually a cleaner, simpler, and easier-to-understand API.

2. With applications, rendering happens much more precisely, and performance can be optimized more.

3. It is also much more powerful and brings new capabilities that were previously impossible or difficult to achieve with WebGL, such as managing command buffers.

4. When using WebGPU-technology, the browser can perform most of the validation and security checks in advance rather than during rendering, reducing the browser's load.

5. Doing this removes a lot of complexity and overhead from the graphics driver. Now it can be much simpler and smaller, just forwarding calls to the hardware. This also reduces the need for notoriously horrible graphics driver bugs.

The result is good for everyone - developers like us have easier time writing applications, hardware manufacturers have an easier time manufacturing products, and gamers benefit from better performance.

There's one downside, however: everything has to be rewritten to take advantage of this. A rewrite of Construct's renderer to use WebGPU isn't the only problem - it affects the entire graphics stack, including browser rendering code, graphics driver code, and possibly even parts of the operating system. Moreover, that's after all the hard specification work, which determines how the API should be designed to work across a wide range of low-level graphics APIs. These new technologies will take time to become widely available since a lot of work will be required across the industry.

WebGL Vs. WebGPU: Similarities and Differences

Similarities

There are many similarities between WebGL and WebGPU-technology. Both allow you to run small functions on the GPU. WebGL has vertex and fragment shaders, while WebGPU has the same as compute shaders. WebGL uses GLSL to shade. WebGPU-technology uses WGSL. Although they are different languages, their concepts are mostly similar.

There are attributes in both APIs that allow you to specify data pulled from buffers and fed into vertex shaders every iteration. Uniforms will enable you to specify values that are shared across multiple shader iterations. There are variations in both APIs, allowing data to be passed from a vertex shader to a fragment shader and interpolated between values computed by the vertex shader during rasterization. It's possible to provide 2D or 3D data and sample it using both APIs (filter multiple pixels into a single value) using textures and samplers. Both APIs offer textures as render targets. And both have a bunch of settings for blending pixels, depth buffers, stencil buffers, etc.

Differences

stateful API

The largest difference between WebGL and WebGPU is that WebGL is a stateful API. By that, I mean that there are many global states in WebGL. Such as what textures are bound, which buffers are bound, what program is running, and what the blending, depth, and stencil settings are. Various API functions like gl.bindBuffer, gl. enable, gl.blendFunc, etc... are used to set those states, and they remain that way until changed.

WebGPU, on the other hand, does not have much global state. Instead, it uses the concept of a pipeline and a render pass, which together effectively contain the majority of the global WebGL state: the texture, the attribute, the buffer, and all the other settings. The default values apply to any settings you do not set. A pipeline cannot be modified. Instead, you create it; then, it becomes immutable. The render passes have a state, but that state is specific to the render pass. if you want different settings, you will need to create another pipeline.

The Canvas

The canvas is managed by WebGL. When you create a WebGL context, you select antialias, preserveDrawingBuffer, stencil, depth, and alpha, then WebGL handles the canvas itself. You just need to set canvas.width and canvas.height.

WebGPU requires you to create all of those buffers on your own. You create the canvas's drawing buffer (with or without alpha) and the depth buffer (with or without stencil buffer). To resize, you must delete the old ones and create new ones. Due to this, unlike WebGL, you can render multiple canvasses with one WebGPU-technology device.

Lack of mipmap generation in WebGPU.

If you want mips for your textures, you must generate them yourself in WebGL. WebGPU has no such function. If you wish to generate mips for your textures, you have to call gl.generateMipmap.

WebGPU requires samplers

For WebGL1, samplers did not exist, or to put it another way, samplers were handled by WebGL internally. For WebGL2, samplers were optional. and for webGPU-technology, samplers are necessary.

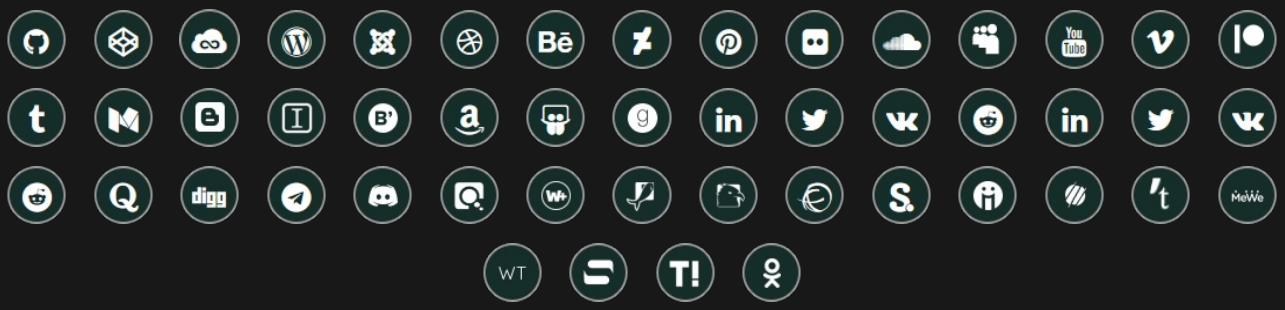
Conclusion

While many product development agencies are offering WebGL expertise, the number is quite limited. Web developers who are knowledgeable and skilled in WebGL are in high demand. It makes sense to look for reputable product development firms that have done successful web projects and have proven expertise in creating 3D canvas graphics. With Arashtad, we leverage WebGL as one of the leading-edge technologies for extended reality solutions. In addition to technical expertise, we also offer full-cycle support for your software applications.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.



Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP | NEWSLETTER | RSS FEED