Interface Development and Design

WebXR

Efrei Paris

2025 - 2026

daniel.mai@efrei.fr

Sommaire

Web3D

OpenGL : la fondation des graphiques 3D modernes

- **OpenGL** (**Open G**raphics **L**ibrary) : une API open-source standardisée utilisée pour le rendu des graphiques 2D et 3D

WebGL : la technologie de base pour les graphiques 3D dans le navigateur

- **WebGL** (**Web G**raphics **L**ibrary) : une API JavaScript qui permet de rendre des graphiques 3D dans n'importe quel navigateur compatible, sans l'utilisation de plugins
- **WebGL** s'appuie sur **OpenGL ES (E**mbedded **S**ystem), une version de **OpenGL** adaptée aux appareils mobiles et aux navigateurs web. Cela permet d'exploiter les capacités de la carte graphique (GPU) directement via le navigateur pour obtenir des performances proches des jeux vidéo natifs

Microsoft® DirectX®

OpenGL™

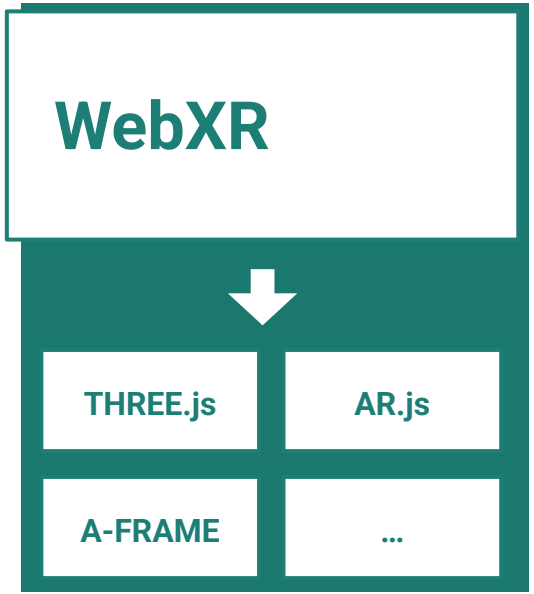
Vulkan®

KHRONOS® GROUP

Web3D

- **Three.js** : une bibliothèque JavaScript permettant de créer et de rendre des scènes 3D dans le navigateur à l'aide de **WebGL**

- **A-Frame** : un framework open-source basé sur HTML, conçu pour créer des expériences **WebXR** immersives telles que la réalité virtuelle (VR) et la réalité augmentée (AR)
- Construit sur **Three.js**, A-Frame permet de manipuler des objets 3D, la lumière, la caméra et les interactions utilisateurs via de simples **balises HTML**, simplifiant ainsi le développement pour les développeurs web



A-Frame et JavaScript : qui fait quoi ?

A-Frame : gestion de l'affichage 3D

- **A-Frame** principalement utilisé pour gérer **l'affichage 3D** et **les interactions XR** (réalité virtuelle et augmentée). Il simplifie le processus de création d'une scène 3D en utilisant des balises HTML pour définir les objets, les caméras, les lumières, et même les interactions de base avec l'utilisateur
- A-Frame offre également une série de **composants préconstruits** qui permettent d'ajouter facilement des fonctionnalités comme le contrôle des mouvements ou l'intégration VR

JavaScript : gestion de la logique du jeu

- **Le contrôle des mouvements** : par exemple, calculer la vitesse de déplacement, les trajectoires, ou les rotations d'un personnage.
- **Gestion des collisions et de la physique** : si besoin de systèmes physiques avancés (comme les moteurs physiques, la gravité, velocity) >> des bibliothèques JavaScript additionnelles, telles que *aframe-physics-system*
- **Interaction utilisateur et gameplay** : les événements de jeu, comme la collecte d'objets, la gestion des scores ou des conditions de victoire, doivent être implémentés en JavaScript

https://aframe.io/docs/1.6.0/introduction/

`requestAnimationFrame();`
son importance dans les jeux sur navigateur

Synchronisation avec le taux de rafraîchissement

- **`requestAnimationFrame`** aligne l'animation avec le **taux de rafraîchissement de l'écran** (généralement 60 FPS), assurant des mouvements fluides et naturels.
- S'adapte automatiquement aux écrans à haute fréquence (ex : 120Hz, 144Hz)

Efficacité et performance

- Permet une meilleure **gestion des ressources** : suspend l'animation lorsque l'onglet est inactif, économisant ainsi de la batterie et des ressources CPU
- Assure une **utilisation optimale du GPU** pour des animations régulières, contrairement à **`setTimeout`** ou **`setInterval`**

requestAnimationFrame();
son importance dans les jeux web

Gestion du temps (`deltaTime`)

- Calculer **`delta Time`** (le temps écoulé entre deux frames) garantit que le jeu reste fluide, même lorsque le FPS fluctue
- Le mouvement d'un objet ou d'un personnage reste cohérent, que le jeu tourne à 30 FPS ou 60 FPS

```
let lastTime = 0;
function gameLoop(currentTime) {
    let deltaTime = (currentTime - lastTime) / 1000;
    lastTime = currentTime;
    // Mise à jour basée sur le temps écoulé
    updateGameLogic(deltaTime);
    // Prochaine frame
    requestAnimationFrame(gameLoop);
}
requestAnimationFrame(gameLoop);
```

Événements JS
keydown et **keyup**

```
let keys = {};
document.addEventListener('keydown', function(event) {
        keys[event.code] = true;
});
document.addEventListener('keyup', function(event) {
        keys[event.code] = false;
});
```

```
function gameLoop() {
        if (keys["KeyW"]) {
                console.log("Avancer");
        }
        if (keys["KeyA"]) {
                console.log("Aller à gauche");
        }
        if (keys["KeyS"]) {
                console.log("Reculer");
        }
        if (keys["KeyD"]) {
                console.log("Aller à droite");
        }
        requestAnimationFrame(gameLoop);
}
```

https://www.w3schools.com/jsref/obj_keyboardevent.asp

A-FRAME : Skybox

```
<a-scene>
    <a-assets>
        <img id="sky"
    src="sky.png">
    </a-assets>
    <a-sky
    src="#sky"></a-sky>
</a-scene>
```

equirectangular image

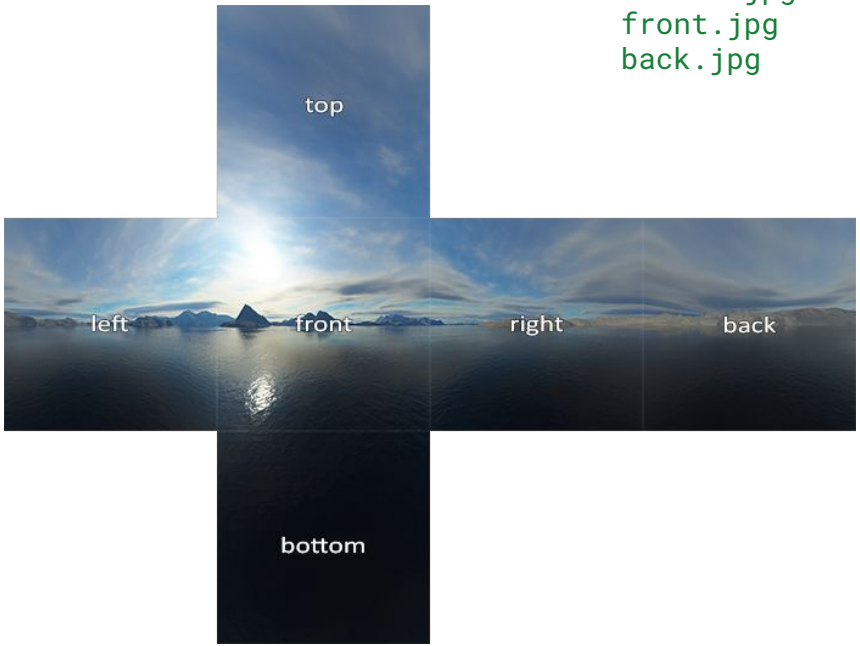
https://aframe.io/docs/1.6.0/primitives/a-sky.html

A-FRAME : cubemap

```
<a-scene>
 <a-assets>
   <!-- Cubemap asset -->
   <a-cubemap id="reflection">
     <img src="cubemap/right.jpg">
     <img src="cubemap/left.jpg">
     <img src="cubemap/top.jpg">
     <img src="cubemap/bottom.jpg">
     <img src="cubemap/front.jpg">
     <img src="cubemap/back.jpg">
   </a-cubemap>
 </a-assets>

 <!-- Sphere with reflection. -->
 <a-sphere position="0 1 -2"
           material="envMap:#reflection;
metalness:1.0; roughness:0.0">
 </a-sphere>
</a-scene>
```

```
right.jpg
left.jpg
top.jpg
bottom.jpg
front.jpg
back.jpg
```



https://aframe.io/docs/1.6.0/primitives/a-cubemap.html

A-FRAME caméra orbit

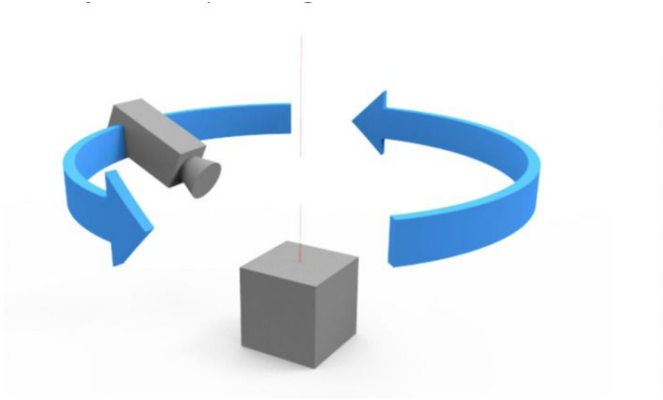
```
<!-- Caméra Orbit autour du cube →

<a-entity id="camera-rig" position="0 0 0">

        <a-entity camera look-controls position="0 0 15">

        </a-entity>

 </a-entity>
```



A-Frame Physics System

- Un module permettant d'ajouter une **simulation physique réaliste** à une scène A-Frame, en utilisant des moteurs de physique comme **Cannon.js** ou **Ammo.js**
- **n'a pas été mis à jour depuis plusieurs années** (environ 4 ans)
- Erreur typique : **"THREE.Geometry is not a constructor"**
    - **Three.js** a supprimé **`THREE.Geometry`** dans à partir de la version **r125**, préférant l'usage de **`BufferGeometry`** pour des raisons de performance

```
<script src="https://aframe.io/releases/0.9.2/aframe.min.js"></script>

<script src="https://cdn.rawgit.com/donmccurdy/aframe-physics-system/v3.3.0/dist/aframe-physics-system.min.js"></script>
```

https://github.com/n5ro/aframe-physics-system
fork : https://github.com/c-frame/aframe-physics-system

Exécuter HTML de GitHub sur le navigateur

- URL
  - https://github.com/{user_name}/{repo_name}/blob/master/{file_name}
  - https://github.com/{user_name}/{repo_name}/blob/main/{file_name}
- Run
  - https://rawgit.com/{user_name}/{repo_name}/master/{file_name}
  - https://rawgit.com/{user_name}/{repo_name}/main/{file_name}