Réalité Virtuelle et Interaction (RVI)
Spécialité Informatique, 5ème année 2025 - 2026

TP WebXR

Introduction avec Aframe

Documentations
- Cours : https://webxr.ldavid.fr
- WebXR : https://developer.mozilla.org/fr/docs/Web/API/WebXR_Device_API
- Aframe : https://aframe.io/docs/1.7.0/introduction/
- Three.js :
  - https://threejs.org/docs/
  - https://threejs.org/manual/#en/installation
- Javascript Cheatsheet :
  - https://syntaxsimplified.com/cheatsheet/Javascript/javascript.html
  - https://quickref.me/javascript.html

Objectif
- Découvrir et prendre en main WebXR via Aframe
- Utiliser diverses documentations
- Réaliser sa première application WebXR

Pré-requis
- Base de programmation
  - idéalement des connaissances en javascript
- Connaissances 3D (vecteurs, matrices, etc.)

Matériel
- un casque XR compatible avec WebXR (ex: Pico, Quest, …)
  - Idéalement 6dof avec manettes 6dof

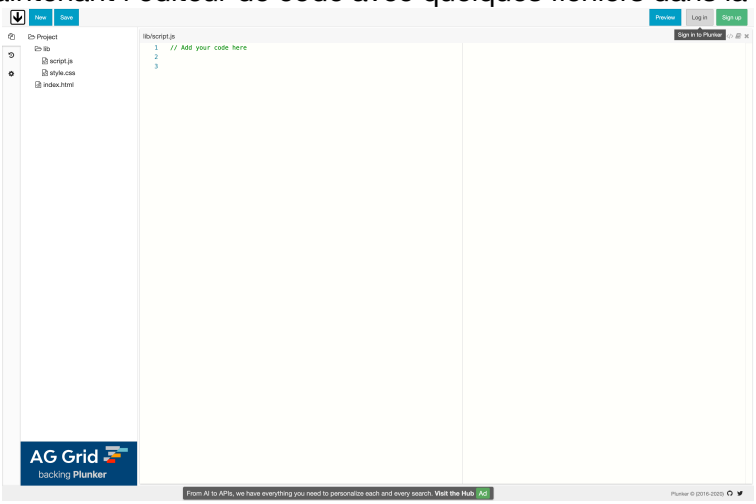
Rendu noté
- Une archive zip nommée “ET5-TP-WebXR-2526-<nom>-<prenom>” contenant les différents exercices (3, 4, 5, 6) comme différents fichiers HTML.

Contact
Ludovic DAVID - ludovic.david@inria.fr

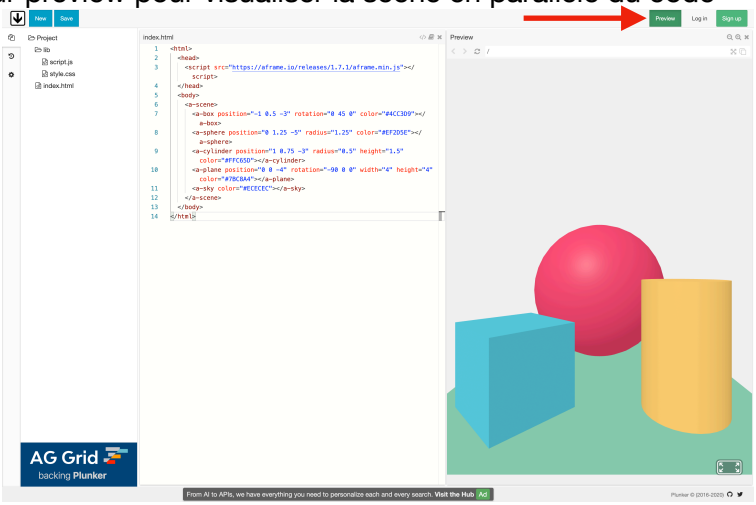
1. Mise en place

Pour éviter d'avoir à installer quoi que ce soit pendant le TP, nous allons utiliser un éditeur de code en ligne qui permet d'héberger des sites web statique sans même avoir à s'inscrire !

- Rendez vous sur https://plnkr.co/
- Cliquer sur *New*
- Sélectionner le type de projet *Default* (an empty html, css and javascript template)
- Vous visualiser maintenant l'éditeur de code avec quelques fichiers dans la hiérarchie



- Ensuite, copier le code de l'introduction d'aframe et coller le dans l'index.html
- Cliquez ensuite sur preview pour visualiser la scène en parallèle du code



- **Avant d'oublier, sauvegardez le projet en cliquant sur *Save :* cela vous évitera de tout perdre au rechargement de la page, donc faites le régulièrement. De plus, conservez bien l'URL pour ne pas perdre votre travail (ex: bookmark, copier dans une note, …).**
- NOTE : vous pouvez changer le thème de l'éditeur dans les paramètres de Plunker…
- Supprimez le dossier *lib*
- **Testez le téléchargement** en cliquant droit sur *Projet*
  - cliquez sur *Download as …*
  - *Nommez votre fichier comme montrer en page 1*

ET5-TP-WebXR-2526-<nom>-<prenom>.zip

2. Accéder l'application sur le casque

Avant de continuer à développer l'application, assurez vous que vous pouvez y accéder dans le casque

- Pour cela, cliquer sur *Open the preview in a separate window*

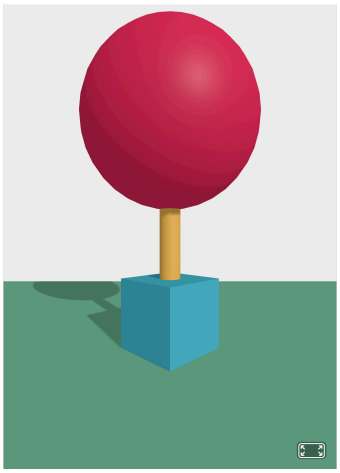
- Une nouvelle page va s'ouvrir avec une URL comme
  https://run.plnkr.co/preview/cmhz0vfy400062e6tw5hhvvup/
- Ouvrez l'url de votre preview dans le navigateur de votre casque
  - **TIPS** (*risque fort de ne pas fonctionner dans la salle de TP + risque de conflit car vous êtes tous sur le même réseau… mais facilite la vie dans d'autres cadres*)
    - si vous êtes sur le même réseau, utilisez hmd.link
    - aller sur le site sur votre PC et votre casque
    - vérifiez que le même identifiant en bas de page s'affiche de chaque côté
      (ex: 

)
    - ajoutez le lien sur votre PC et vous n'aurez qu'à cliquer dessus dans votre casque !

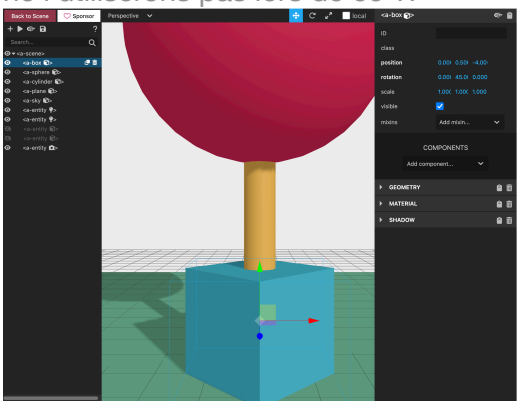
Si tout c'est bien déroulé, vous devriez pouvoir entrer dans votre scène immersive sur votre casque en cliquant sur le bouton VR ou AR qui s'affiche en bas à droite.

Sinon, contactez l'encadrant du TP.

3. Familiarisez vous avec Aframe

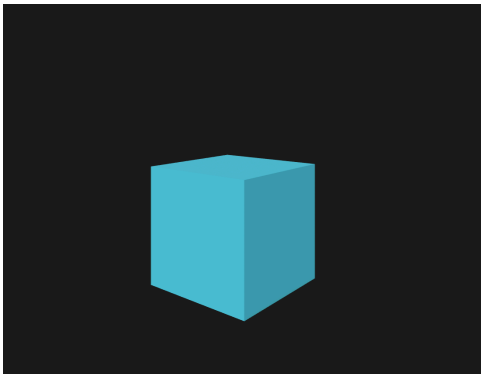


- Parcourez la documentation brièvement (~5min)
  - https://aframe.io/docs/1.7.0/introduction/
  - https://aframe.io/docs/1.7.0/introduction/html-and-primitives.html
- Reproduisez la scène ci-dessus en changeant les *positions*, *orientations* et *attributs (width, radius, …)* des objets.
  - N'oubliez pas d'ajouter les ombrages (https://aframe.io/docs/1.7.0/components/shadow.html#main)
  - **TIPS**
    - Aframe dispose d'un *inspecteur :* https://aframe.io/docs/1.7.0/introduction/visual-inspector-and-dev-tools.html. De la même manière que vous pouvez inspecter vos pages web, aframe permet de le faire pour les scènes 3d. Pour l'ouvrir, appuyez sur Shift+Alt+i (ou Shift+Option+i sur Mac). Cela peut vous permettre plus facilement de disposer vos objets dans votre scène.
    - [ATTENTION] ce n'est pas un éditeur, donc toutes les modifications seront perdues au prochain rechargement de la page !!! Si vous souhaitez copier des valeurs de l'inspecteur dans votre code, n'oubliez pas de désactiver l'option *refresh the preview automatically* dans les paramètres de Plunker avant. Vous pouvez évidemment activer de nouveau le paramètre par la suite si vous le souhaitez.
    - [NOTE] Il existe un moyen de conserver les modifications via aframe-watcher, mais nous ne l'utiliserons pas lors de ce TP
    -



- **Une fois la scène terminée, copiez son contenu dans un nouveau fichier nommé "*3.html*"**
  - (Pour créer un nouveau fichier, cliquez droit sur *Project* puis cliquer sur *Create file…*

4. Votre premier composant



À partir de maintenant, créer un fichier html pour chaque exercice. Ici “4.html”.

Lors de l’exercice, vous pouvez d’abord écrire le code dans l’index.html pour ne pas avoir à modifier l’URL dans votre casque et avoir la preview dans Plunker.

Cependant, à chaque fin d’exercice, copier-coller votre code dans le fichier correspondant.

- Lisez en diagonal la documentation sur l’écriture d’un composant
  - https://aframe.io/docs/1.7.0/introduction/entity-component-system.html
  - https://aframe.io/docs/1.7.0/introduction/writing-a-component.html
- Dans cet exercice, repartez d’une scène vide avec juste un cube coloré sur fond #1b1b1b
  - https://aframe.io/docs/1.7.0/components/background.html#main
- Codez votre premier composant qui permet de faire tourner sur lui-même autour de l’axe y l’objet auquel il est attaché.
- Pour ce faire, commencez par ajouter un script directement dans le fichier html pour définir le composant.

```
<script>
    AFRAME.registerComponent('auto-rotate', {
        // …
    });
</script>
```

- Ensuite, trouvez la fonction qui est appelée à chaque frame dans le composant
- Dans cette fonction, mettez à jour la rotation sur l’axe y en passant directement par l’object3D three.js de l’élément
  - **TIPS**
    - https://threejs.org/manual/#en/creating-a-scene
    - Les rotations dans threejs sont en radians
- Ajoutez maintenant un paramètre *speed* (avec une valeur par défaut décente) pour faire varier la vitesse de rotation
- Mettez ensuite trois cubes de couleurs différentes dans votre scène et faites les tourner à des vitesses différentes
- Idéalement, utiliser le shortcut Alt+Shift+F (ou l’icône) pour formatter votre code proprement avant de passer à l’exercice suivant et de sauvegarder.
- **COPIEZ LE CONTENU DANS “*4.html*”**

5. Un peu d'interaction

- Lisez brièvement la document sur l'interaction et les contrôleurs
  - https://aframe.io/docs/1.7.0/introduction/interactions-and-controllers.html
- Ajoutez des contrôles : manettes ou mains selon votre appareil et vos préférences
- Vérifiez que vos contrôles s'affichent dans le casque
- Codez un composant qui permet à un objet de suivre un autre objet[1]
- Créer de multiples objets (commencez par 5) qui se suivent les uns les autres (comme un train)
- Faites que l'objet de tête suive un de vos contrôles

Pour faciliter les tests et évitez de mettre le casque trop régulièrement pendant le développement, nous allons ajouter IWER: https://meta-quest.github.io/immersive-web-emulation-runtime/

"Immersive Web Emulation Runtime 🔓 Unlock WebXR Emulation Everywhere"

Pour ce faire, nous allons quelque peu changer le chargement des librairies.
Nous allons utiliser une *importmap* pour décrire aux modules où aller chercher les librairies :

```
<script type="importmap">
  {
    "imports": {
      "iwer": "https://unpkg.com/iwer@1.1.1/build/iwer.module.js",
      "@iwer/devui": "https://unpkg.com/@iwer/devui@0.2.1/build/iwer-devui.module.js",
      "three": "https://cdn.jsdelivr.net/npm/three@0.181.1/build/three.module.js",
      "aframe": "https://unpkg.com/aframe@1.7.1/dist/aframe-v1.7.1.module.min.js"
    }
  }
</script>
```

Puis, utiliser des *scripts* de type *module* pour importer ce dont nous avons besoin

```
<script type="module">
  import { XRDevice, metaQuest3 } from 'iwer';
  import { DevUI } from '@iwer/devui';

  const xrDevice = new XRDevice(metaQuest3);
  xrDevice.installRuntime();

  // Initialize the DevUI with the configured XR device
  const devui = new DevUI(xrDevice);
</script>
```

Pour rappel, dans votre structure HTML, ces éléments doivent impérativement se placer dans la balise <head>, et surtout **avant tout usage d'aframe** car lors de son initialisation, celui-ci va chercher à savoir si le navigateur supporte WebXR. Hors IWER émule le support WebXR. Il doit donc être déclaré avant pour qu'Aframe pense que WebXR est présent et qu'il puisse utiliser IWER.

Maintenant, au lieu d'utiliser `<script src="https://aframe.io/releases/1.7.1/aframe.min.js"></script>` et directement **AFRAME** dans le code, nous allons l'importer comme module.

```
<script type="module">
      import AFRAME from 'aframe';

      // ...
</script>
```

[1] https://aframe.io/docs/1.7.0/introduction/writing-a-component.html#example-follow-component

Vous pouvez maintenant voir qu'au lieu du bouton *fullscreen* en bas à gauche, votre application propose désormais le bouton VR, même sur desktop. Lorsque vous cliquez dessus, l'émulation IWER s'active et vous pouvez alors vérifier que vos objets suivent votre contrôle.

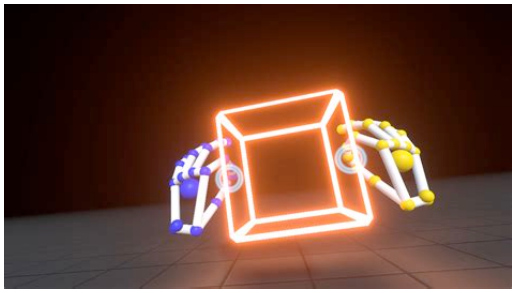
Pour éviter qu'IWER ne se lance dans votre casque, vous pouvez encapsuler le code de création du device et de la DevUI dans un if qui vérifie si "xr" n'est pas présent dans le navigateur.

NOTE :
Plunker semble avoir des difficultés à faire de la coloration syntaxique pour les scripts de type module…

TIPS :
- Adaptez la taille de vos objets (par exemple des cubes de 2cm)
- Si parfois la hiérarchie aframe semble simple, beaucoup de choses de passe de manière invisible avec three.js en dessous. Par example, si vous utilisez les mains comme contrôle (*hand-tracking-controls*), l'entité elle-même ne change pas de position. Aframe encourage grandement à aller lire le code source pour mieux savoir ce qu'il se passe, par exemple : https://github.com/aframevr/aframe/blob/v1.7.0/src/components/hand-tracking-controls.js
  - On peut alors voir que certaines positions sont accessibles comme *this.wristObject3D* ou *this.indexTipPosition* ;
  - Souvenez aussi qu'il est possible d'accéder à d'autres composants, comme par exemple element.components['hand-tracking-controls'].wristObject3D
- N'hésitez pas à ajouter des valeurs par défaut aux paramètres de vos composants…
- Vous utilisez plusieurs les mêmes composants dans vos entités ? Jetez un coup d'oeil aux *Mixins :* https://aframe.io/docs/1.7.0/core/mixins.html (dans l'esprit des prefabs Unity)

COPIER LE CONTENU DANS 5.html avant de passer à la suite

6. Redo Leap Motion Blocks



- Créer une copie de *Leap Motion Blocks* :
  - L'utilisateur peut utiliser ses mains ou des manettes pour créer des cubes lorsqu'il *pinch* ou *triggerdown* les deux contrôles simultanément.
  - Le cube doit se situer entre les contrôleurs
  - Le cube se dimensionne en fonction de l'écart entre les contrôleurs (mains ou manettes).
  - Le cube doit avoir une taille minimale (par exemple 2,5cm).
  - Le cube doit s'orienter sur l'axe y selon la disposition des contrôleurs.
  - L'opacité du cube doit être < 1 pendant la création (tant que l'utilisateur *pinch/triggerdown*).
    - Une fois le geste finalisé, l'opacité du cube = 1
  - Les instances de cubes doivent pouvoir être manipulées (*grab*)
  - Lorsque le geste est relâché, le cube doit tombé selon la physique (aframe-physics-system)
  - Le *grab* doit être compatible avec la physique[2]

BONUS :
- Détruire les objets trop éloignés (notamment ceux qui tombent à l'infini)
- Deux paumes orientées vers le haut devant l'utilisateur pour supprimer la gravité, orientées vers le bas pour rétablir la gravité normale
- Ajouter 3 boutons à droite de la paume gauche orientée vers le haut afin de sélectionner d'autres primitives
- Ajouter des capacités de lancer (donner une vitesse/accélération lors d'un *grabended*.
- Ajouter la physique à la main (afin de pouvoir déplacer les objets en les poussants)
- Ajouter des effets visuels via des shaders (par exemple sur la création du cube)
- Ajouter des sons subtils (lors du pincement, lors de la création, lors d'une collision)

COPIER LE CONTENU DANS 6.html avant de rendre le TP

- POUR LE RENDU -
TÉLÉCHARGEZ VOTRE PROJET SOUS FORMAT ZIP
ET NOMMEZ LE COMME INDIQUÉ
ET5-TP-WebXR-2526-<nom>-<prenom>.zip

AVANT DE LE SOUMETTRE, VÉRIFIEZ SON CONTENU

[2] TIP: Rien ne vous empêche de copier un composant aframe déjà existant pour modifier le code afin de gérer certains cas non prévus, comme la physique