IUT du Limousin / Département MMI / BUT2
SAE 4.DWeb-DI.02 | Concevoir un dispositif interactif

Compétences ciblées

Exprimer un message avec les médias numériques pour informer et communiquer
Développer pour le web et les médias numériques

Apprentissages critiques

AC23.02
Définir une iconographie (illustrations, photographies, vidéos)
AC23.04
Imaginer, écrire et scénariser en vue d'une communication multimédia ou transmédia
AC23.06
Élaborer et produire des animations, des designs sonores, des effets spéciaux, de la visualisation de données ou de la 3D
AC24.03
Intégrer, produire ou développer des interactions riches ou des dispositifs interactifs

Ressources mobilisées et combinées

R4.DWeb-DI.01 Anglais
R4.DWeb-DI.05 Création et design interactif
R4.DWeb-DI.06 Développement front

Objectifs et problématique professionnelle

Objectifs : combiner les ressources liées aux compétences développer et exprimer pour développer une application interactive avec un objectif de promotion d'un produit, de divertissement ou simplement artistique.

En tant que développeurs juniors, les étudiants doivent concevoir et développer une application en répondant à la question : Comment développer une application interactive plaçant les utilisateurs au centre du dispositif ?

Descriptif générique

La SAE est l'occasion d'expérimenter différentes technologies ou méthodes de programmation : réalité virtuelle, dispositifs interactifs ou immersifs, jeux vidéo 2D ou 3D. Les étudiants mobilisent des compétences et ressources techniques aussi bien que créatives et veillent à l'ergonomie et à la qualité esthétique des productions.

1. Organisation de la SAé

Cette SAé est à réaliser en deux temps :

- d'abord un travail en trinôme (4 groupes, composition fixée par les enseignants) sur 3 semaines
- ensuite un travail individuel pour implémenter des améliorations

Groupe 1	Cakir	Huzeyfe
	Lochis	Ethan
	Kamara	Suzanne
Groupe 2	Hujol	Celena
	Viroulaud	Anna
	Zbalah	Addem
Groupe 3	Reich	Malo
	Pain	Vivien
	Mishcherikova	Alina
Groupe 4	Lippler	Manon
	Paugnat	Jean
	Gader	Wahel

Vous pouvez retrouver le déroulement de la SAé ci-dessous. Attention les séances prévues en autonomie n'apparaissent pas mais l'idée est bien que vous devez travailler sur cette SAé dès que vous avez un moment de libre.

Semaine 1	19/01/2026	16:00-17:30	SPRINGINSFELD Denis	
	20/01/2026	08:00-09:30	CRESPIN Benoit	
	21/01/2026	08:00-09:30	MORA Frédéric	
	23/01/2026			Livrable #1
Semaine 2	26/01/2026	10:30-12:00	CRESPIN Benoit	
	26/01/2026	16:00-17:30	SPRINGINSFELD Denis	
	27/01/2026	15:30-17:00	MORA Frédéric	
	30/01/2026			Livrable #2
Semaine 3	02/02/2026	10:30-12:00	CRESPIN Benoit	
	03/02/2026	15:30-17:00	MORA Frédéric	
	04/02/2026	08:00-09:30	SPRINGINSFELD Denis	
	06/02/2026			Livrable #3 - Fin du travail en trinôme
Semaine 4	09/02/2026	14:00-15:30	CRESPIN Benoit	Démo
	10/02/2026	08:00-09:30	SPRINGINSFELD Denis	
	10/02/2026	09:30-11:00	MORA Frédéric	

	13/02/2026			Livrable #4
Semaine 5	23/02/2026	08:00-10:00	CRESPIN Benoit	
	25/02/2026	08:00-10:00	SPRINGINSFELD Denis	
			ADAMCZYK Natacha	Préparation oral
			ADAMCZYK Natacha	Préparation oral
	27/02/2026	14h		Soutenance finale
	06/03/2026	*13:30-15:30*	*LAVEFVE Valérie*	*(séance portfolio)*

La thématique 2025/26 pour cette SAé est la **production d'une application XR (réalité mixte) dans un casque Quest 3 avec les technologies AFrame et WebXR**. Le choix de l'appli elle-même est libre (jeu, appli éducative, etc), mais devra être validé par les enseignants. Vous devez réfléchir aussi à l'**impact numérique de votre application**, grâce à par exemple des modèles "low-poly" (https://poly.pizza/), ce qui a également comme intérêt d'éviter de longs temps de chargement.

Le but est de fournir une appli web et des livrables de qualité professionnelle, **entièrement en anglais**. Vous devez néanmoins rester réaliste et viser des fonctionnalités que vous vous sentez capables de terminer dans le temps imparti. La conduite du projet, votre répartition du travail, l'utilisation de Github, etc. feront partie de l'évaluation finale.

Pour les semaines 1, 2 et 3 vous serez en trinôme. Vous partirez ensuite chacun sur vos propres améliorations (ou une refonte complète) sur les semaines 4 et 5.

Chaque groupe doit créer et utiliser pour le développement un dépôt Github à partager avec :
- benoit.crespin@unilim.fr
- frederic.mora@unilim.fr
- denis.springinsfeld@unilim.fr

Les livrables seront à déposer sur Community :
https://community-iut.unilim.fr/course/view.php?id=3528

Pour le prêt de casques VR : amelin.chanteloup@unilim.fr (studio audiovisuel)

2. Livrables

#1
Le premier livrable est un document de deux pages maximum en PDF qui présente (en anglais) :
- votre idée d'application, les interactions envisagées, etc
- un Gantt qui décrit les tâches prévues, leur durée, à qui elles sont affectées, etc (qui devrait avoir déjà commencé)
- les liens vers votre dépôt Github et vers les premières maquettes testables en ligne si vous en avez déjà

#2
Le second livrable doit présenter la première version de l'application avec des captures d'écran et des explications, et donner l'avancée par rapport aux objectifs initiaux. Présentez ensuite les améliorations envisagées pour la suite et le planning de travail prévu pour les jours restants. En

conclusion, chaque membre du groupe doit expliquer ce sur quoi il a principalement contribué. Partagez les liens vers la page qui héberge votre application et vers une capture vidéo prise avec le screencast Meta.

#3
Pour le dernier livrable en trinôme vous devez décrire la version finale et une section dédiée à la gestion de projet : état du planning final par rapport au planning initial, tâches prévues mais non terminées, améliorations que vous auriez pu implémenter avec plus de temps, estimation du temps total passé sur le projet, explications pour chaque membre des tâches auxquelles il a contribué. Redonnez les liens vers la page qui héberge votre application et vers une ou plusieurs captures vidéo prises avec le screencast Meta.

#4
PDF de deux pages présentant votre avancement : quelles améliorations envisagées, résultats déjà obtenus, etc. Vous pouvez repartir du projet existant mené en trinôme ou de zéro.

#5
Lien vers un site web complet réalisé **individuellement**, hébergé sur Github ou autre, qui présente votre application finale avec des captures vidéos, donne des détails sur le fonctionnement avec des petits exemples de code illustrant des fonctionnalités-clés. L'application devra bien sûr être accessible depuis ce site, qui contiendra également un lien vers le dépôt Github pour le code source et toutes les ressources consultées sur le web pour aboutir au résultat final. Pendant la présentation orale vous utiliserez ce site comme support pour montrer le code et les vidéos : **15 minutes en anglais** avc une présentation commune au début par l'ensemble du trinôme, puis chacun montre ses améliorations.

Votre expérimentation sera appréciée comme d'habitude comme suit :

- Convaincant et votre base d'évaluation sera 15.
  C'est convaincant si le résultat est de qualité professionnelle en tout point. Si des défauts existent, ils sont mineurs et aisément rectifiables.

- Mitigé et votre base d'évaluation sera 10.
  C'est mitigé si le résultat est intéressant mais avec des défauts majeurs qui ne sont pas acceptables dans une optique professionnelle. Un défaut majeur reste rectifiable mais demandera un travail significatif.

- Insuffisant et votre base d'évaluation sera 5.
  C'est insuffisant si le résultat n'est tout simplement pas utilisable et/ou ne répond pas à la demande. Les défauts sont alors critiques, trop de mauvais choix ou d'erreurs ont été faites pour envisager rectifier le tir sans reprendre le projet de zéro.

Cette base d'évaluation sera ensuite modulée en appréciant séparément le niveau d'acquisition de tous les apprentissages critiques (voir au début du document) **et l'implication mesurée à travers votre activité sur Github au cours de la SAé**.

3. Réalité étendue (XR)

La **réalité étendue (XR)** est un terme générique qui englobe toutes les technologies immersives fusionnant les mondes réel et virtuel. Dans un casque VR, la XR combine trois expériences principales : la **réalité virtuelle (VR)** qui vous plonge dans un environnement 100% numérique, la **réalité augmentée (AR)** qui superpose des éléments virtuels sur votre environnement réel grâce aux

caméras du casque, et la **réalité mixte (MR)** qui permet aux objets virtuels d'interagir avec votre espace physique. Ces technologies transforment votre casque en une porte vers des expériences variées : du gaming immersif aux visites virtuelles, en passant par des applications professionnelles comme la formation ou la collaboration à distance, offrant ainsi un continuum d'expériences entre le monde réel et virtuel.

Pour la SAé **c'est à vous de choisir quelle application XR vous souhaitez développer**, en utilisant obligatoirement les technologies A-Frame et WebXR.

La librairie A-Frame permet le développement d'applications web orientées VR (entre autres), et vous devrez faire face à de nombreux challenges : utiliser des modèles 3D, des images, ajouter de l'interactivité, combiner A-Frame et WebXR, etc. La librairie A-Frame est un peu une version "simplifiée" de Threejs que vous découvrez dans un autre cours, avec de nombreux points communs. Vous trouverez quelques exemples en annexe.

WebXR est l'API web qui permet de créer et d'exécuter des expériences XR directement dans un navigateur, sans nécessiter d'installation de logiciels ou de plugins. Concrètement, WebXR permet aux développeurs d'utiliser JavaScript pour accéder aux fonctionnalités de votre casque VR : suivi de position et d'orientation, gestion des contrôleurs, affichage stéréoscopique pour chaque œil, et détection de l'environnement. L'avantage majeur est la compatibilité : une seule application WebXR fonctionne sur tous les casques compatibles (Meta Quest, HoloLens, etc.) et peut même s'adapter aux smartphones pour l'AR, sans avoir à développer une version spécifique pour chaque plateforme même si ce n'est pas toujours aussi simple : **dans votre cas partez du principe que vous développez pour un Quest 3**.

Exemples en JS "pur" :

- Mixed Reality Support in Browser
- WebXR - Samples

Voici maintenant une version du dernier exemple qui combine A-Frame et WebXR :

- vidéo : https://mediaserver.unilim.fr/videos/13012026-171936/
- code : https://github.com/BenoitCrespin/SAE4.DWeb-DI.02-XR/

Pour tester le code en local :

- mettez le PC et le casque sur un même réseau Wifi (mais pas eduroam, plutôt votre téléphone)
- sur le PC lancez "python3 ./serveur.py", qui va vous afficher l'URL à utiliser
- repassez sur le casque et lancez le navigateur internet avec cette URL pour tester
- le réseau local est approprié aussi pour la "mise en miroir"

L'autre possibilité est bien sûr de mettre votre code sur un serveur (type Github Pages ou autre).

Pour la SAé le but est de manipuler A-Frame et WebXR, mais dans le cadre d'une application "pro" vous pouvez aussi réfléchir à d'autres aspects. Si vous créez une application qui permet de décorer virtuellement une pièce par exemple, on peut imaginer pouvoir sauvegarder ses créations, voir les créations d'autres utilisateurs, etc : il ne s'agit pas ici de 3D ou de VR/XR mais d'aspects plus "classiques" en dév web qui peuvent également être très utiles.

Annexe : A-Frame pour la VR “classique”

Vous pouvez trouver ci-dessous des bouts de code qui seront montrés lors de la première séance.

Exemple 1 - Bases de A-Frame

Une scène A-Frame peut être définie comme une simple liste de primitives définies avec des balises HTML spécifiques.

```
<!-- https://aframe.io/docs/1.5.0/introduction/html-and-primitives.html -->
<html>
 <head>
   <script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
 </head>
 <body>
   <a-scene>
     <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
     <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
     <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5"
color="#FFC65D"></a-cylinder>
     <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4"
color="#7BC8A4"></a-plane>
     <a-sky color="#ECECEC"></a-sky>
   </a-scene>
 </body>
</html>
```

Vous pouvez tester cette scène en local dans un navigateur sur votre PC, mais pour pouvoir la tester dans le casque il est nécessaire que le code soit hébergé sur un site externe (ie Github ou autre). Ensuite vous pouvez lancer le navigateur dans le casque pour accéder à votre site et utiliser le mode “VR”.

Exemple 2 - Une scène plus complexe

Avec des librairies additionnelles il est possible, toujours avec des balises HTML, de rajouter des animations, des effets de pluie, etc.

```
<!-- https://aframe.io/docs/1.5.0/introduction/entity-component-system.html
-->
<html>
 <head>
   <title>Community Components Example</title>
   <meta name="description" content="Community Components Example">
   <script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
```

```
   <script
src="https://cdn.jsdelivr.net/gh/c-frame/aframe-particle-system-component@a5a
8449/dist/aframe-particle-system-component.js"></script>
   <script
src="https://cdn.jsdelivr.net/npm/aframe-simple-sun-sky@^1.2.2/simple-sun-sky
.js"></script>
   <script
src="https://cdn.jsdelivr.net/gh/c-frame/aframe-extras@d5f3f8/dist/aframe-ext
ras.min.js"></script>
 </head>
 <body>
   <a-scene>
       <!-- https://github.com/IdeaSpaceVR/aframe-particle-system-component
-->
     <a-entity id="rain" particle-system="preset: snow; color: #24CAFF;
particleCount: 5000"></a-entity>
     <a-entity id="sphere" geometry="primitive: sphere"
               material="color: #EFEFEF; shader: flat"
               position="0 0.15 -5"
               light="type: point; intensity: 5"
               animation="property: position; easing: easeInOutQuad; dir:
alternate; dur: 1000; to: 0 -0.10 -5; loop: true"></a-entity>
     <a-entity id="ocean" ocean="density: 20; width: 50; depth: 50; speed: 4"
               material="color: #9CE3F9; opacity: 0.75; metalness: 0;
roughness: 1"
               rotation="-90 0 0"></a-entity>
     <a-simple-sun-sky sun-position="1 0.4 0"></a-simple-sun-sky>
     <a-entity id="light" light="type: ambient; color: #888"></a-entity>
   </a-scene>
 </body>
</html>
```

Exemple 3 - Interactions avec la souris ou les contrôleurs VR

L'intérêt principal de A-Frame réside dans le fait que les entités définies dans le code HTML peuvent être "enrichies" avec du code Javascript. On peut ainsi déterminer par exemple si un objet est cliqué avec la souris (dans un navigateur sur PC) ou avec un contrôleur (dans le casque VR, ici un Oculus).

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Simple VR Scene</title>
   <script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
```

```
</head>
<body>
   <a-scene cursor="rayOrigin: mouse">
       <a-box position="0 1.6 -4" color="#4CC3D9" scale="1 1 1" id="myBox"
class="collidable"></a-box>
       <a-sphere id="mySphere" position="2 1.6 -6" radius="0.5"
color="#FF0000"></a-sphere>
       <a-entity id="leftController" oculus-touch-controls="hand:
left"></a-entity>
       <a-entity id="rightController" oculus-touch-controls="hand: right">
           <a-entity raycaster="showLine:true;objects: .collidable"
cursor></a-entity>
       </a-entity>
       <script>
           document.querySelector('#myBox').addEventListener('click',
function () {
               var box = document.querySelector('#myBox');
               box.setAttribute('color', '#' + Math.floor(Math.random() *
16777215).toString(16));
           });

document.querySelector('#rightController').addEventListener('triggerdown',
function () {
               var s = document.querySelector("#mySphere");
               s.setAttribute('color', '#' + Math.floor(Math.random() *
16777215).toString(16));
           });
       </script>
   </a-scene>
</body>
</html>
```

Exemple 4 - Animations et ajout d'objet dynamique

En réalité, on peut pratiquement tout faire en Javascript, par exemple ajouter des éléments à la scène A-Frame et/ou une boucle d'animation comme dans Threejs.

```
<!--
https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_on_the_web/Build
ing_up_a_basic_demo_with_A-Frame -->
<html>
<head>
   <script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
</head>
<body>
```

```
    <a-scene>
        <a-sky color="#DDDDDD"></a-sky>
        <a-light type="directional" color="#FFF" intensity="0.5" position="-1
1 2"></a-light>
        <a-light type="ambient" color="#FFF"></a-light>
        <a-camera position="0 1 4" cursor-visible="true" cursor-scale="2"
cursor-color="#0095DD"
            cursor-opacity="0.5"></a-camera>
        <a-box position="-1 1.6 -5" animation="property: position; to: 1 8
-10; dur: 2000; easing: linear; loop: true"
            color="tomato"></a-box>
        <a-entity rotation="0 0 0" animation="property: rotation; to: 0 360 0;
loop: true; dur: 10000">
            <a-sphere position="5 0 0" color="mediumseagreen"></a-sphere>
        </a-entity>
        <script>
            var scene = document.querySelector('a-scene');
            var cylinder = document.createElement('a-cylinder');
            cylinder.setAttribute('color', '#FF9500');
            cylinder.setAttribute('height', '2');
            cylinder.setAttribute('radius', '0.75');
            cylinder.setAttribute('position', '3 1 0');
            scene.appendChild(cylinder);
            var t = 0;
            function render() {
                t += 0.01;
                requestAnimationFrame(render);
                cylinder.setAttribute('position', '3 ' + (Math.sin(t * 2) + 1)
+ ' 0');
            }
            render();
        </script>
    </a-scene>
</body>
</html>
```

A vous de jouer, gardez à l'esprit que la qualité du code et la modularité (*ie* le découpage en différents fichiers) seront bien sûr prises en compte dans l'évaluation.