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

QUI ?
Cette SAé est à réaliser en trinôme (5 groupes)

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

23 JANV. 2024, MAR. 13:30 à 15:00	SAÉ 4.DWeb-DI.02 TD G3 CRESPIN Benoit, BUT2-G3 R02	Phase de recherche et de tests
24 JANV. 2024, MER. 14:00 à 17:30	SAÉ 4.DWeb-DI.02 Autonomie G3 , BUT2-G3	Phase de recherche et de tests
31 JANV. 2024, MER. 08:30 à 10:00	SAÉ 4.DWeb-DI.02 TD G3 CRESPIN Benoit, BUT2-G3 R01	Présentation et validation du projet envisagé => Document à rédiger
7 FÉVR. 2024, MER. 14:00 à 15:30	SAÉ 4.DWeb-DI.02 TD G3 CRESPIN Benoit, BUT2-G3 R01	Développement itération 1
16 FÉVR. 2024, VEN. 13:30 à 15:30	SAÉ 4.DWeb-DI.02 TD G3 CRESPIN Benoit, BUT2-G3 R01	Présentation itération 1 et fonctionnalités prévues pour l'itération 2 => Document à rédiger
19 FÉVR. 2024, LUN. 09:00 à 11:00	SAÉ 4.DWeb-DI.02 TD G3 SPRINGINSFELD Denis, BUT2-G3 R02	Développement itération 2
20 FÉVR. 2024, MAR. 09:00 à 11:00	SAÉ 4.DWeb-DI.02 TD G3 MORA Frédéric, BUT2-G3 R01	Développement itération 2
21 FÉVR. 2024, MER. 09:00 à 11:00	SAÉ 4.DWeb-DI.02 TD G3 PORRO Heinich, BUT2-G3 R04	Développement itération 2
23 FÉVR. 2024, VEN. 14:00 à 16:00	SAÉ 4.DWeb-DI.02 TD G3 CRESPIN Benoit, MORA Frédéric, PORRO Heinich, SPRINGINSFELD Denis, BUT2-G3 R04	Présentation finale => Site web à réaliser
12 MARS 2024, MAR. 14:00 à 15:30	SAÉ 4.DWeb-DI.02 TP G3 LAVEFVE Valérie, CRESPIN Benoit, BUT2-G3 R04	Ne compte pas pour cette SAé

QUOI ?
La finalité de cette SAé est : **Promouvoir un geste éco-responsable à travers un petit jeu**. A vous de choisir et de définir des principes de jeu basés sur cette thématique, et si vous cherchez de l'inspiration :

- https://www.durable.com/10-gestes-eco-responsables-a-adopter-au-quotidien/
- https://www.economie.gouv.fr/mission-innovation/sensibilisation-aux-ecogestes
- …

Des contraintes complémentaires concernent la technologie à utiliser. En effet votre jeu devra être **codé avec la librairie A-Frame et jouable sur navigateur ou sur casque VR.** Vous devez réfléchir aussi à l'**impact numérique de votre application**, en utilisant par exemple des modèles "low-poly" (https://poly.pizza/) et des extensions comme Green-IT:
https://www.francenum.gouv.fr/guides-et-conseils/pilotage-de-lentreprise/numerique-durable/evaluer-lempreinte-environnementale-de

COMMENT ?
Chaque groupe doit créer et utiliser pour le développement un dépôt Github à partager avec les 4 intervenants :

- benoit.crespin@unilim.fr
- frederic.mora@unilim.fr
- heinich.porro@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 (HTC Vive Focus) et/ou l'acquisition de photos 360°, etc. :
amelin.chanteloup@unilim.fr (studio audiovisuel)

2. Livrables

A fournir avant le 31 janvier 8h30
Le premier livrable est un document d'une page maximum qui présente votre idée de jeu, le lien avec la thématique proposée, quelles seront les interactions, etc.
Fournir aussi les liens vers votre dépôt Github et vers la maquette du jeu.

A fournir avant le 16 février 13h30
Le second livrable doit présenter la première version du jeu avec des captures d'écran et des explications, et donner l'avancée par rapport aux objectifs initiaux. Présentez ensuite les améliorations envisagées pour la version 2 et le planning de travail prévu.
En conclusion, chaque membre du groupe doit expliquer ce sur quoi il a principalement contribué et ce sur quoi il pense principalement contribuer pour la suite du projet.

A fournir avant le 23 février 14h
A l'issue de cette SAé, vous devrez avoir produit **un site web complet**, hébergé sur le serveur mmi, github ou autre, qui présente le ou les éco-gestes que vous avez choisi de traiter, explique en quoi votre jeu permet de s'y relier et quelles sont les règles, donne des détails sur le fonctionnement du jeu et comment il est codé (avec des petits exemples de code illustrant des fonctionnalités-clés de votre application). Le jeu devra bien sûr être accessible depuis ce site, qui contiendra également un

lien vers le dépôt github et toutes les sources consultées sur le web pour aboutir au résultat final. Pendant la présentation vous montrerez le site, le code… et le jeu (prévoyez un casque VR, mais ça doit marcher aussi dans un navigateur).

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. A-Frame

Pour cette année 2023/24 nous vous imposons d'utiliser la librairie A-Frame, qui simplifie le développement d'applications web orientées VR (entre autres) même si vous devrez faire face à de nombreux challenges pour créer un "vrai" jeu : utiliser des modèles 3D, des images, ajouter de l'interactivité, etc. La librairie A-Frame se base largement sur Threejs, que vous découvrez dans un autre cours, avec de nombreux points communs.

Quelques liens à consulter :

- https://aframe.io/
- https://aframe.io/docs/1.5.0/introduction/
- https://blog.glitch.com/post/an-intro-to-webvr
- https://aframe.io/examples/showcase/modelviewer/
- https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_on_the_web/Building_up_a_basic_demo_with_A-Frame
- https://www.youtube.com/watch?v=t5Hou5QsRiE&ab_channel=CodeChangers
- … (Google, ChatGPT, etc)

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 se résumer à 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 avec l'extension LiveServer. Néanmoins pour pouvoir la tester dans le casque il est nécessaire que le code soit hébergé sur un site accessible depuis le réseau (ie le serveur MMI ou autre). Ensuite vous pouvez lancer le navigateur Vive Browser 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).

```
<!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" vive-controls="hand: left"></a-entity>
       <a-entity id="rightController" vive-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 !