

Clément FERRERE
Victor MOMMALIER
Clara PONCET
Lucile VELUT

RAPPORT DE PROJET TUTORÉ

VISITE VIRTUELLE DE L'IUT
GÉNÉRATEUR DE PANORAMA

DUT Informatique Clermont-Ferrand – 2^{ème} année

AUTORISATION À DIFFUSER SUR L'INTRANET DE L'IUT

Nous autorisons la diffusion de notre rapport sur l'intranet de l'IUT.

REMERCIEMENTS

Nous remercions notre tuteur de projet M. SALVA pour son accompagnement et le prêt de matériel tout au long du projet ainsi que l'équipe enseignante du DUT Informatique.

SOMMAIRE

| | |
|---|-----------|
| AUTORISATION À DIFFUSER SUR L'INTRANET DE L'IUT..... | 2 |
| REMERCIEMENTS | 3 |
| SOMMAIRE | 4 |
| INTRODUCTION..... | 5 |
| DÉVELOPPEMENT..... | 6 |
| I. PRESENTATION DU PROJET | 6 |
| II. GESTION DE PROJET..... | 7 |
| a. <i>Présentation du WBS</i> | 7 |
| b. <i>Gantt prévisionnel</i> | 8 |
| c. <i>Gantt réel et analyse des écarts</i> | 9 |
| III. PRESENTATION A-FRAME | 10 |
| IV. PANORAMA | 11 |
| a. <i>Objectifs</i> | 11 |
| b. <i>Panorama</i> | 11 |
| c. <i>La carte</i> | 16 |
| d. <i>Animations et améliorations ergonomiques</i> | 17 |
| V. GENERATEUR | 21 |
| a. <i>Objectifs</i> | 21 |
| b. <i>Interfaces utilisateurs</i> | 22 |
| c. <i>Diagramme de classe</i> | 25 |
| d. <i>Edition des scènes</i> | 26 |
| e. <i>Sauvegarde et génération HTML</i> | 31 |
| BILAN TECHNIQUE..... | 33 |
| CONCLUSION | 34 |
| RÉSUMÉ EN ANGLAIS | 35 |
| BIBLIOGRAPHIE | 36 |
| LEXIQUE | 37 |
| ANNEXES..... | 38 |

INTRODUCTION

Notre projet tutoré de deuxième année de DUT Informatique consiste en la réalisation d'une visite virtuelle de l'IUT Informatique de Clermont-Ferrand ainsi que d'un générateur de panorama en ligne. Il a été réalisé sur une durée de cinq mois, du 9 novembre au 29 mars.

Notre équipe est composée de Clément Ferrere, Enzo Mazella, Victor Mommaliier, Clara Poncet et Lucile Velut. Ce projet a été proposé et supervisé par notre tuteur M. Salva, professeur de l'IUT.

Le projet est composé de deux parties et mettra en avant les panoramas sous forme d'applications web. Un panorama est défini comme un enchaînement de scènes composées de photos à 360° utilisées pour recréer un environnement virtuel autour de l'utilisateur. Les photos étant en 360°, elles couvrent l'intégralité du champ de vision de l'utilisateur même lorsque celui-ci se tourne ou se déplace. Ces scènes seront accompagnées de panneaux informatifs, de points de navigation permettant de circuler d'une scène à l'autre, et d'une carte interactive facilitant les déplacements et apportant une vue d'ensemble du panorama. Ce projet a été réalisé grâce au framework¹ A-Frame² qui permet de créer du contenu web en réalité virtuelle.

La première partie de notre projet est une visite immersive de l'IUT. Ce panorama apporte des informations sur l'IUT et les points de navigations permettent de se déplacer d'une scène à l'autre afin de visiter l'intégralité du département informatique comme si l'on s'y trouvait.

La seconde partie est une application web de génération de panorama. Ce site internet permet à n'importe qui ayant des photos à 360° de créer son propre panorama en arrangeant les scènes les unes par rapport aux autres et en ajoutant des éléments de navigation et d'information. Le panorama résultat est ensuite téléchargeable et peut être visualisé grâce à un serveur web.

Notre but est donc de créer deux outils ergonomiques, un pour la visite de l'IUT et l'autre pour un public plus large, qui offrent des solutions concrètes de visite à distance tout en offrant une expérience immersive et utilisable sur trois plateformes : casque de réalité virtuelle, navigateur sur ordinateur et sur téléphone et casque de RV¹ pour téléphone. Les visualisations sur téléphones utilisent le gyroscope intégré de l'appareil pour reproduire le déplacement physique dans le panorama.

Dans ce rapport, nous allons détailler notre travail au cours de ces cinq mois. Tout d'abord, nous aborderons la gestion de notre projet, puis nous présenteront l'élément technologique clé de ce projet : le framework A-Frame. Nous détaillerons ensuite le panorama de l'IUT, puis le générateur de panorama avant de conclure.

¹ Ces termes sont définis dans le lexique en fin de rapport

² Une présentation détaillée d'A-Frame est disponible en page 10

DÉVELOPPEMENT

I. Présentation du projet

a. L'environnement technique

Le projet ayant déjà été proposé à un autre groupe d'étudiants il y a quelques années, les photos à 360° du département informatique et du bloc central du site de Clermont-Ferrand de l'IUT nous ont été fournies. Ces photos ont été réalisées grâce à un appareil photo à 360° composé de 2 objectifs *fisheye*³ qui produit des photos rectangulaires qui, une fois mise sous forme d'une sphère, recréent une vue immersive de l'environnement photographié. Ces photos ont été compressées afin de réduire la taille importante des fichiers qui aurait fortement ralenti les transferts.

Pour mettre en scène ces photos en RV, nous avons utilisé le framework A-Frame qui est un framework HTML³ permettant de créer des environnements virtuels. La visite virtuelle de l'IUT a ainsi été réalisée majoritairement avec le langage HTML avec des parties en JavaScript³, notamment pour les animations.

Pour créer le générateur, nous avons choisi le langage PHP³ pour réaliser le back-end³ de notre application web et les langages HTML et CSS³ pour la partie front-end³. En revanche, une partie importante du site repose sur du JavaScript, notamment l'édition des scènes.

b. Les objectifs

Notre principal objectif commun aux deux parties de notre projet était de créer des visites virtuelles compatibles sur trois plateformes : casques RV, ordinateurs et téléphones. Le générateur n'est utilisable que sur ordinateur mais c'est son résultat qui doit être compatible sur ces trois plateformes.

Tous ces objectifs ont été définis dans le cahier des charges au début de la réalisation de notre projet.

Les objectifs spécifiques à la visite de l'IUT sont les suivants :

- Visualiser des scènes de l'IUT en 360°
- Se déplacer d'une scène à l'autre grâce aux points de navigation
- S'informer grâce à des panneaux informatifs
- Accéder à la carte représentant une vue d'ensemble de l'IUT
- Naviguer rapidement en cliquant sur des lieux figurants sur la carte

Ceux pour le générateur de panorama sont :

- Charger des photos 360°
- Editer chaque scène :
 - Définir le point de vue, la position de la caméra
 - Fixer les points de navigation
 - Ajouter des panneaux informatifs
- Créer et éditer la carte
- Sauvegarder le panorama sous forme d'une application web

Tous les objectifs de la visite de l'IUT s'appliquent également au résultat du générateur.

³ Ces termes sont définis dans le lexique en fin de rapport

Grâce au panorama de l'IUT, nous avons souhaité créer une visite immersive s'adressant à n'importe qui désirant visiter le département informatique de l'IUT de Clermont-Ferrand. Cela pourrait par exemple être un futur étudiant ne pouvant se rendre sur place. Nous souhaitons pouvoir le proposer comme un outil supplémentaire lors des portes ouvertes qui ont dû se dérouler à distance cette année.

Avec le générateur, nous souhaitons viser un public beaucoup plus large. Nous pourrions imaginer notre site internet utilisé par un particulier souhaitant créer une visite virtuelle d'un bien immobilier qu'il souhaite vendre. La seule condition d'utilisation est de posséder des photos à 360°.

II. Gestion de projet

a. Présentation du WBS

Notre WBS se scinde en trois parties distinctes : une première sur le panorama original de l'iut, une seconde sur notre générateur et la dernière partie correspond à la documentation du projet tel que le rapport et la soutenance. Le WBS dans son intégralité est disponible en annexe en fin de rapport.

Dans la première partie, il y a tout d'abord l'analyse du panorama pour pouvoir commencer au bon endroit et ne pas se lancer dans l'inconnu. Nous avons ensuite programmé la base du panorama, puis la création de la carte et finalement nous avons dû vérifier si notre panorama était totalement fonctionnel sur les trois plateformes demandées : un ordinateur, un smartphone et un casque de réalité virtuelle (type Oculus rift).

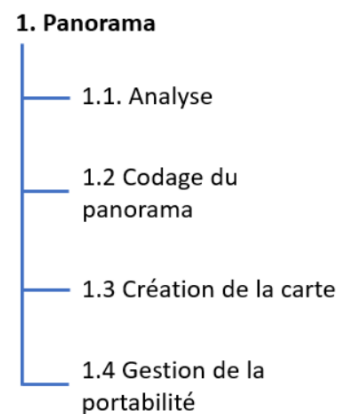


Figure 1.a : première partie du WBS

Pour le générateur, nous avons également commencé par l'analyse pour définir les cas d'utilisations ainsi que les différentes classes objets dans deux diagrammes. Nous avons ensuite codé le générateur avec les différentes étapes que sont le formulaire pour rentrer les photos, l'ajout des points de navigation ainsi que des panneaux informatifs puis nous avons intégré la possibilité que l'utilisateur rajoute une carte correspondante à son panorama. Pour finir le générateur, il a fallu rajouter une sauvegarde et une génération pour que le résultat soit consultable sur les mêmes plateformes que le panorama de l'IUT.

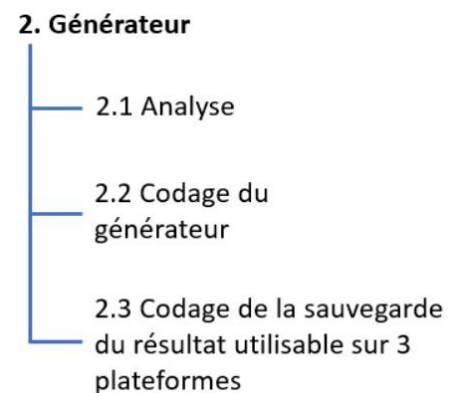


Figure 1.b : deuxième partie du WBS

Concernant la partie sur le rapport et la soutenance de notre projet, nous avons décidé dans un premier temps qu'il nous fallait rédiger les documents prévisionnels comme ce WBS ou un Gantt qui a été séparé en deux parties : une première pour la période 2 de notre année et une deuxième pour la période 3. Nous avons également prévu la préparation des soutenances, et pour finir la rédaction du rapport.

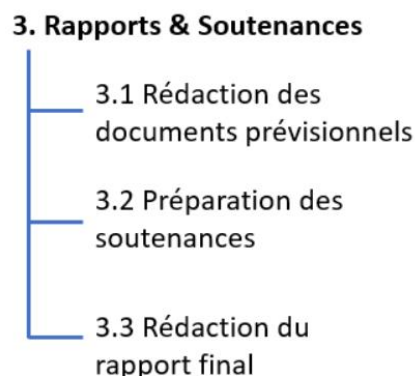


Figure 1.c : troisième partie du WBS

b. Gantt prévisionnel

Suite au cours de gestion de projet qui se déroulait en parallèle du projet tutoré, nous avons fait un Gantt prévisionnel afin de nous organiser et de répartir le travail plus efficacement. Le diagramme de Gantt est disponible en deux parties dans les annexes en fin de rapport.

Nous avons commencé par mettre en place les tâches les plus élémentaires comme faire un exemple d'utilisation du framework A-Frame pour pouvoir le prendre en main et comprendre son fonctionnement, puis faire l'analyse du panorama. Nous avons ensuite commencé la programmation des points de navigations ainsi que des panneaux informatifs avec une partie de l'équipe tandis que l'autre partie qui s'occupait de la création d'une carte pour faciliter la navigation au sein du panorama. Cette première partie a été terminée en janvier.

| | | |
|---|------------|------------|
| Analyse du panorama (Clara-Lucile) | 23/11/2020 | 06/12/2020 |
| Prise en Main du Framework A-Frame | 09/11/2020 | 22/11/2020 |
| Codage des scènes du panorama (Victor-Enzo) | 23/11/2020 | 06/12/2020 |
| Codage des points de navigation (Victor-Enzo-...) | 30/11/2020 | 22/12/2020 |
| Ajout des panneaux informatifs (Lucile) | 07/12/2020 | 22/12/2020 |
| Analyse ergonomique de la carte du panorama ... | 23/11/2020 | 29/11/2020 |
| Codage de la carte du panorama (Clément-Enzo) | 30/11/2020 | 10/01/2021 |

Figure 2.a : gantt prévisionnel du panorama

Après la fin de notre deuxième période, nous avons repris notre Gantt afin de le compléter pour la partie sur le générateur. Dans un premier temps, il a fallu faire une analyse des besoins pour celui-ci. Nous avons ensuite décidé de créer deux équipes : une qui se chargera de la sauvegarde en JSON ainsi que de la génération en HTML, et la seconde équipe réalisera le formulaire de soumission des photos ainsi que le JavaScript pour pouvoir rentrer de nouveaux points de navigation et panneaux informatifs. Ces étapes devaient se finir le 14 mars.

| | | |
|--|------------|------------|
| Analyse du générateur (Clara-Lucile) | 07/12/2... | 10/01/2021 |
| Codage du formulaire permettant l'ajout des pho... | 04/01/2... | 10/01/2021 |
| Codage de l'ajout des points de navigation | 11/01/2... | 31/01/2021 |
| Codage de l'ajout de panneaux informatifs | 11/01/2... | 31/01/2021 |
| Codage de la création de la carte | 25/01/2... | 14/03/2021 |

Figure 2.b : gantt prévisionnel du générateur

Nous avons fini par les étapes 3 du WBS, telle que la rédaction du rapport ainsi que la préparation à la soutenance finale de 25 minutes.

c. Gantt réel et analyse des écarts

Nous allons maintenant passer à l'analyse de notre Gantt réel. Pour la première partie, nous avons fait une analyse poussée grâce à nos cours de gestion de projet. Vous retrouverez tous ces documents en annexe. Ce que nous avons pu conclure de notre première partie de projet c'est que l'on avait fait de bonnes prédictions car nous avons suivi le Gantt parfaitement et n'avons pas eu de retard.

À l'inverse, pour la seconde partie, nous savions moins comment nous y prendre. Nous avons tenté de faire un Gantt (cf. ci-dessus) afin de prévoir quelle tâche nous allions faire et à quel moment. Mais nous pensions avoir fini le panorama alors qu'il restait des erreurs sur celui-ci. Nous avons donc décidé de nous séparer en deux équipes : une qui allait se concentrer sur la création du générateur (Lucile, Clara, Victor) et une autre sur les erreurs à corriger sur le Panorama de l'IUT (Clément et Enzo). C'est à ce moment-là que nous avons appris la démission d'Enzo. Clément s'est donc retrouvé seul sur le panorama, ce qui nous a ralenti car nous pensions que cette équipe qui travaillait sur le panorama pourrait par la suite nous rejoindre pour nous aider sur le générateur. Nous avons donc pris une semaine de retard sur le Gantt prévisionnel.

Les étapes qui étaient prévues à la fin du projet étaient la sauvegarde en cours du panorama ainsi que la génération du fichier HTML. Nous avons donc choisi de nous concentrer sur la génération car cela nous semblait plus intéressant à finir que la sauvegarde. Comme dit plus haut nous devons finir le projet le 14 mars, or nous avons pris une semaine de retard donc nous n'avons pas encore fini notre projet. Il nous reste la génération du fichier HTML. Nous pensons par ailleurs ne pas avoir assez de temps pour mettre en place la sauvegarde en cours d'édition.

Nous pouvons en conclure que la gestion et l'appréhension du projet se sont mieux passées pour le panorama de l'IUT que pour le générateur. Cela peut être dû à plusieurs facteurs : en cette troisième période nous n'avons pas eu de cours de gestion de projet. Nous avons donc peut-être délaissé la gestion, sans prendre en compte les conséquences, pour prioriser la technique pure afin de finir le projet. De plus, dans le contexte actuel, nous n'avons jamais eu l'occasion de travailler en présentiel et malgré les moyens mis en place tels que les réunions le lundi matin ainsi qu'une conversation de groupe sur une plateforme de messagerie, il y a eu un manque de communication qui a pu mener à un retard.

III. Présentation A-Frame

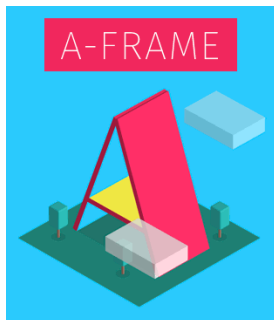


Figure 3 : logo A-Frame

A-frame est un framework qui fonctionne avec le langage de programmation JavaScript mais dont l'utilisation se fait principalement en HTML. La première version est rendue disponible en décembre 2015. Il est open-source, cela veut dire que l'auteur autorise toute personne à l'utiliser gratuitement et à y apporter des modifications.

Conçu à l'origine au sein de Mozilla et maintenant maintenu par les co-créateurs d'A-Frame au sein de Supermedium, A-Frame a été développé pour être un moyen simple mais puissant de développer du contenu RV. A-Frame est devenu l'une des plus grandes communautés de réalité virtuelle.

Basé sur de l'OpenGL, il permet de faire facilement des rendus 3D sur des plates-formes web. Il est plus particulièrement fait pour de la réalité virtuelle que l'on nomme WebVR. Il est plus facile d'utilisation qu'OpenGL. La documentation est accessible sur le site du framework. De plus, A-Frame prend en charge la plupart des casques RV tels que Vive, Rift, Windows Mixed Reality, Daydream, GearVR, Cardboard, Oculus Go et peut même être utilisé pour la réalité augmentée.

L'utilisation de ce framework est très simple. En effet, il suffit de partir d'une page HTML vierge et de la compléter comme une page normale. Pour pouvoir ajouter les fonctionnalités du framework, il suffit de l'insérer comme un script JavaScript :

```
<title>Panorama</title>
<script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
<script src="https://unpkg.com/aframe-slice9-component/dist/aframe-slice9-component.min.js"></script>
<script src="https://unpkg.com/aframe-look-at-component@0.5.1/dist/aframe-look-at-component.min.js"></script>
```

Figure 4 : code de l'intégration des scripts JS

Nous l'avons utilisé pour développer un panorama et un générateur mais ce framework permet de créer différentes solutions tel que des jeux-vidéos sur serveur web ou des applications web.

IV. Panorama

a. Objectifs

Afin de mieux définir les objectifs de la visite virtuelle de l'IUT, nous avons réalisé un diagramme de cas d'utilisation de notre application web.

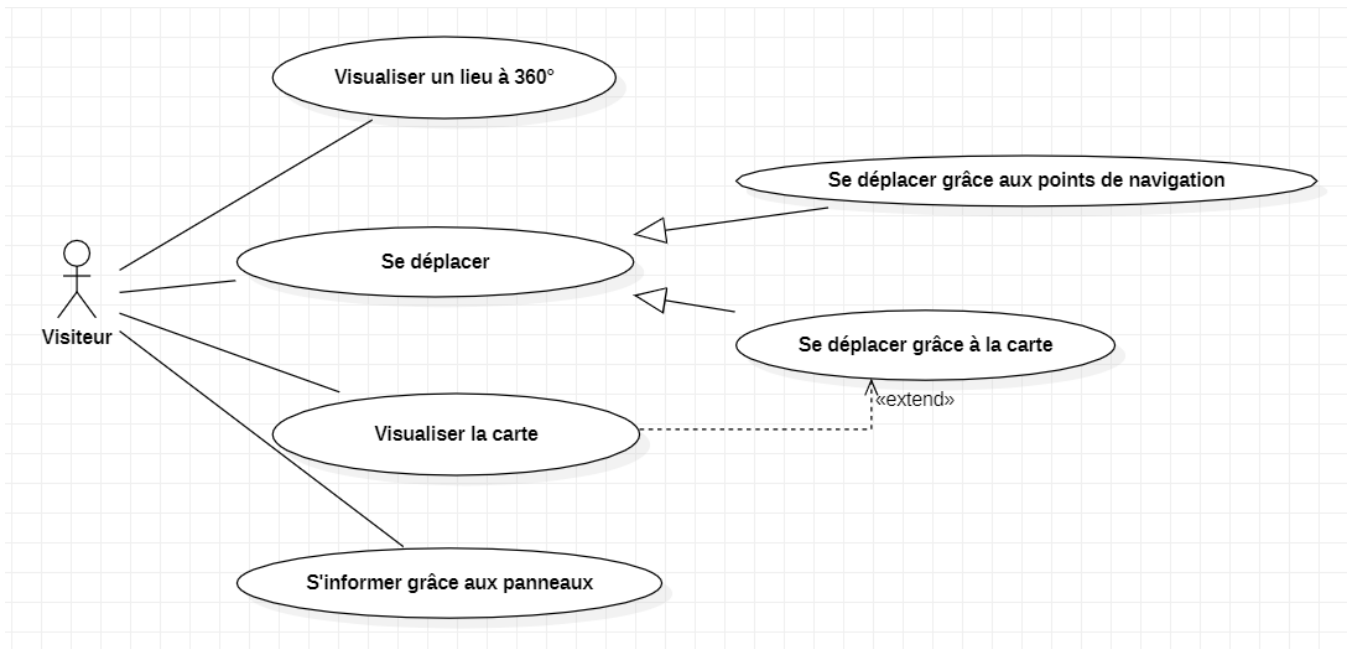


Figure 5 : diagramme de cas d'utilisation du panorama

Sur ce diagramme, nous pouvons voir que l'utilisateur peut visualiser un lieu à 360° et s'y déplacer, soit grâce aux points de navigation soit en visualisant la carte qui doit être accessible à partir de chaque scène de la visite.

Le panorama doit également comporter des panneaux informatifs qui indiquent à l'utilisateur où il se trouve et vers où il peut se diriger.

b. Panorama

Le panorama est composé d'une scène constituée de plusieurs hotspot. Un hotspot est un « groupe » qui est composé de points de navigation, de panneaux et de cartes.

Nous allons aborder et détailler la partie scène dans un premier temps, puis nous définirons plus précisément la composition d'un hotspot par la suite.

1. Création des groupes

L'aspect création des groupes correspond à la partie encadrée dans le diagramme objet du panorama (voir la figure n°6.a ci-dessous).


```

<a-assets>

```

Figure 9 : code des images mises en asset

La scène contient également une entity qui représente la caméra, et une autre entity qui représente le curseur.

Maintenant que les termes de la partie gauche du diagramme objet du panorama (figure n°6) ont été définis, nous pouvons aborder la constitution des groupes « hotspot », ce qui correspond à l'encadré suivant :

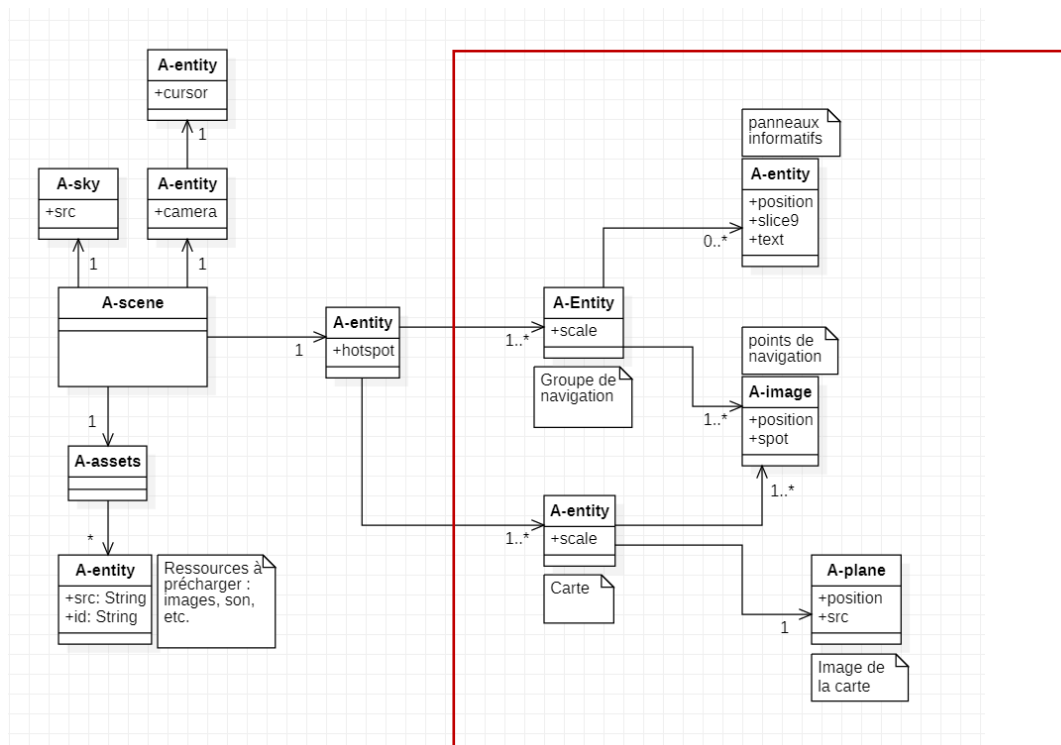


Figure 6.b : deuxième partie diagramme de classe du panorama

Les groupes (hotspot) sont constitués de balises⁴ <a-entity>, qui contiennent, comme expliqué précédemment, les panneaux, les points de navigation et la carte. Voici un exemple de code complet pour le groupe qui a pour id « group-coinGauche » :

⁴ Ce terme est défini dans le lexique en fin de rapport

```

<a-entity id="group-coinGauche" scale="0 0 0">
  <!--panneaux-->
  <a-entity slice9="width: 5; height: 3; left: 20; right: 43; top: 20; bottom: 43;src: tooltip.png"
    text="value:Entree arriere batiment informatique;wrap-count:15; width:5; align:center" ;
    look-at="#cam" position="10.350 1.684 -3.803"></a-entity>

  <a-entity slice9="width: 4; height: 2; left: 20; right: 43; top: 20; bottom: 43;src: tooltip.png"
    text="value:Departement biologie ;wrap-count:15; width:5; align:center" ;
    look-at="#cam" position="-10.8 0.163 -4.246"></a-entity>
  <!--points de navigation-->
  <a-image spot="linkto:#croisementChemin;spotgroup:group-croisementChemin"
    position="2.4 -0.17 15.36" src="#fleche" look-at="#cam"></a-image>
  <a-image spot="linkto:#exterieurGauche;spotgroup:group-exterieurGauche"
    position="13 -0.5 -1.6" src="#fleche" look-at="#cam"></a-image>
  <!--carte-->
  <a-image spot="linkto:#fondBlanc;spotgroup:group-cartePrincipale"
    position="-6 0 -5.6" src="#map" look-at="#cam"></a-image>
</a-entity>

```

Figure 10 : code d'un groupe complet

Nous allons maintenant voir en détail ce code, et comment sont codés ces trois éléments.

2. La navigation

La navigation entre les groupes est rendue possible grâce à l'utilisation d'un script JS préexistant. Ce script crée deux attributs : *hotspot* et *spot* (ce dernier ayant des paramètres *linkto* et *spotgroup*) qui vont être ajoutés à des composants de notre panorama. Le hotspot décrit précédemment dans la description du diagramme objet du panorama est donc une simple entité A-Frame à laquelle on a rajouté ce nouvel attribut *hotspot*.

Contenus dans ce hotspot se trouvent tous les groupes de navigation (c'est-à-dire les scènes du panorama) qui ont un attribut *scale* qui a la valeur « 1 1 1 » pour la scène en cours et la valeur « 0 0 0 » pour toutes les autres. Ainsi, un seul des groupes de navigation contenus dans le hotspot n'est visible puisque les autres ont tous une taille nulle.

Chaque groupe a ensuite un ou plusieurs points de navigation, représenté par une image. Ces points de navigation ont un attribut *spot* qui va permettre d'indiquer quelle est leur destination. Le paramètre *linkto* indique l'identifiant de l'image de la nouvelle scène et le paramètre *spotgroup* précise le groupe de navigation de destination.

```

AFRAME.registerComponent('hotspots',{
AFRAME.registerComponent('spot',{
  schema:{
    linkto:{type:"string",default:""},
    spotgroup:{type:"string",default:""}
  },
<a-entity id="spots" hotspots>
  <a-entity id="group-A12" scale="1 1 1">
    <a-image spot="linkto:#fondBlanc;spotgroup:group-fondBlanc"
      position="-1 -3 -6" src="#map" look-at="#cam">
    </a-image>

```

Figure 11 : extraits d'un panorama et du script JS permettant la navigation

Lorsque le script JS détecte un clic sur un point de navigation (après que le curseur a été pointé sur le point pendant 2 secondes), il exécute une fonction qui remplace la source de l'image de la skybox (l'image de fond du panorama) par l'image précisée dans le paramètre *linkto* du point de navigation. Ensuite, la taille de la scène courante est réduite à 0 (elle devient invisible) et la taille de la nouvelle scène est passée à 1, elle devient donc visible ainsi que tous les éléments qu'elle contient.

Le point de navigation est caractérisé par trois autres attributs :

- « position » : position de l'image dans l'espace 3D.
- « src » : image du point de navigation. Le # sert à appeler l'identifiant d'une image qui a auparavant été déclaré dans les assets (voir figure n°9). Nous avons opté pour l'image d'une flèche pour que la fonction de l'image soit explicite.
- « look-at » : oriente l'image. Ici, en mettant la camera, l'image sera toujours face à l'utilisateur.

3. Les panneaux

Les panneaux informatifs orientent l'utilisateur tout au long de la visite virtuelle.

```

<a-entity slice9="width: 5; height: 3; left: 20; right: 43; top: 20; bottom: 43;src: tooltip.png"
  text="value:Entree arriere batiment informatique;wrap-count:15; width:5; align:center" ;
  look-at="#cam" position="10.350 1.684 -3.803"></a-entity>

```

Figure 12 : code d'un panneau

Nous avons fait plusieurs essais pour rendre le panneau informatif le plus ergonomique possible. D'abord sous la forme d'une balise `<a-plane>`, nous avons ensuite choisi d'utiliser un composant déjà existant pour jouer sur la transparence de l'élément.

Ce composant se trouve dans l'attribut « *slice9* », où est spécifié la photo de ce panneau et ses réglages de taille.

Le panneau a ensuite, tout comme le point de navigation un attribut « look-at » et un attribut position. Il a également un attribut « text » où se trouve le texte du panneau en question.

c. La carte

La carte propose une navigation plus directe d'un point à l'autre de l'IUT. Naviguer de groupe en groupe peut devenir fastidieux si notre objectif est d'atteindre un groupe à l'opposé de celui où nous nous trouvons. Par conséquent, nous mettons à disposition un groupe dédié à de longues navigations, qui est une carte générale de l'IUT vu du ciel, accessible depuis n'importe quel endroit dans le panorama via un point de navigation avec l'icône suivante :



Figure 13 : icône de la carte

Comme la carte est un groupe, nous retrouvons les mêmes principes dans sa création que pour les groupes de navigation qui montrent l'IUT.

Nous préchargeons donc des images et des icônes dans les <a-assets> :

```
<a-assets timeout="30000">

  <!-- Cartes -->
  
  
  
  
  
  
  
  <a-sound id="boup" src="son/BOUP.mp3"></a-sound>

</a-assets>
```

Figure 14 : code des images préchargées

Pour les utiliser, nous passons par l'attribut *src* d'une entité :

```
<a-plane position="-1.89574 1.6 -1.96425" src="#cartePrincipale"
  look-at="#cam" height="4" width="6"
  material="" geometry="" rotation="0 43.98317825991033 0">
</a-plane>
```

Figure 15 : code du plan représentant l'IUT vu du ciel

Nous disposons également de points de navigation qui renvoient vers différents points-clés du panorama :


```

<a-image height="0.480" width="0.300" link="href:exterieurParking.html; on:click" look-at="#cam"
position="-0.058 1.125 -3.527" src="#LogoJaune" sound="src:#boup; on: click"
color="#FFFFFF"
></a-image>
<a-image height="0.480" width="0.300" link="href:batCentralR.html; on:click" look-at="#cam"
position="-1.102 2.144 -2.342" src="#LogoVert" sound="src:#boup; on: click"
color="#FFFFFF"
></a-image>
<a-image height="0.480" width="0.300" link="href:exterieurInfoGauche.html; on:click" look-at="#cam"
position="-0.86567 2.98873 -2.84189" src="#LogoJaune" sound="src:#boup; on: click"
color="#FFFFFF"
></a-image>
<a-image height="0.480" width="0.300" link="href:exterieurInfoMilieu.html; on:click" look-at="#cam"
position="-0.63693 2.00661 -3.01714" src="#LogoJaune" sound="src:#boup; on: click"

```

Figure 16 : code des points de navigation sur la carte

La carte fonctionne donc comme les autres groupes de navigation. Un <a-sky> définit le fond de la scène, des entités ont pour source des images préchargées, montrant une vue globale de l'IUT, et permettant d'aller vers un autre groupe éloigné beaucoup plus rapidement.

d. Animations et améliorations ergonomiques

Le panorama tel que nous vous l'avons décrit jusqu'à présent était fonctionnel à la fin du mois de janvier, mais il était largement perfectible, aussi bien d'un point de vue technique qu'ergonomique. Nous allons donc expliquer quelles sont les modifications que nous avons apportées au panorama depuis, ainsi que les raisons qui nous ont poussé à les faire.

1. Animations sur les icônes

Le panorama était très statique et paraissait un peu ancien voire austère, ce qui rentre en contradiction avec la capacité de rendu 3D que nous proposait A-Frame. Nous voulions cependant éviter l'ajout d'éléments superflus pour garder une bonne lisibilité. Nous avons donc choisi d'animer les icônes des points de navigation. Ceci facilite la compréhension de la navigation au travers du panorama tout en apportant du dynamisme.

```

<a-image height="0.480" width="0.300" link="href:batCentral.html; on:click" look-at="#cam"
position="-0.852 1.275 -2.810" src="#LogoVert" sound="src:#boup; on: click"
color="#FFFFFF"
animation__ColorMouseEnter="property: color; to: #0CE2B8; startEvents: mouseenter; dur: 300"
animation__ColorMouseLeave="property: color; to: #FFFFFF; startEvents: mouseleave; dur: 300"
animation__ScaleMouseEnter="property: scale; to: 1.2 1.2 1.2; startEvents: mouseenter; dur: 300"
animation__ScaleMouseLeave="property: scale; to: 1 1 1; startEvents: mouseleave; dur: 300"></a-image>

```

Figure 17 : code des animations d'une icône de la carte

Dans cet exemple, nous définissons 4 animations sur un point de navigation. Nous changeons la taille et la couleur lorsque nous passons au-dessus de l'image. A l'inverse, la couleur et la taille redeviennent normales lorsque nous enlevons la souris de l'image.

Nous avons appliqué des animations comme celles-ci à tous les points de navigation. De même, les points amenant à la carte ont l'animation de changement de taille, mais pas celle de la couleur.

2. Problème de chargement, découpage de panorama

Lors d'une tentative de déploiement par notre tuteur, nous avons rencontré un problème auquel nous n'avions pas pensé auparavant : le chargement du panorama.

En effet, nous avons environ 70Mo de données à charger en une seule fois, soit un temps de chargement de plus de 10 secondes pour près de 40% des français. Nous devons donc trouver des solutions afin de réduire ce délai de chargement, pouvant être de plusieurs minutes pour les plus faibles connexions.

La première solution a été de compresser les images de l'IUT. Nous avons pu réduire la taille totale des données à environ 23Mo (soit 32% de la taille initiale) en compressant les fichiers. Nous n'avons pas pu les compresser davantage car la qualité de l'image aurait été trop diminuée. Bien que cette solution ait été efficace, elle restait insuffisante.

La deuxième solution choisie a été de séparer le panorama en plusieurs fichiers HTML, permettant ainsi de séparer les temps de chargement et donc de les réduire grandement. Nous avons donc séparé le panorama en 5 « zones » principales :

- Bâtiment Central
- Extérieur
- Bâtiment Informatique RDC
- Bâtiment Informatique 1^{er} étage
- Carte

Ainsi, un utilisateur voulant voir le bâtiment informatique n'aura pas besoin de charger les <a-assets> du bâtiment central par exemple.

Cependant, la séparation du panorama s'est avérée plus complexe que prévue à cause de la navigation, les transitions entre les zones ne correspondaient pas toujours à ce qui était attendu. Prenons un exemple (voir figure 18 ci-après).

Le premier groupe (cercle jaune) de la « zone extérieure » est sur le parking de l'IUT devant le bâtiment informatique (en vert). Si vous entrez dans le bâtiment central (en rouge), et que plus tard vous ressortez vers la « zone extérieure », alors vous vous téléporterez directement au parking de l'IUT devant le bâtiment informatique (cercle jaune) alors que vous devriez être devant le bâtiment central (cercle bleu).

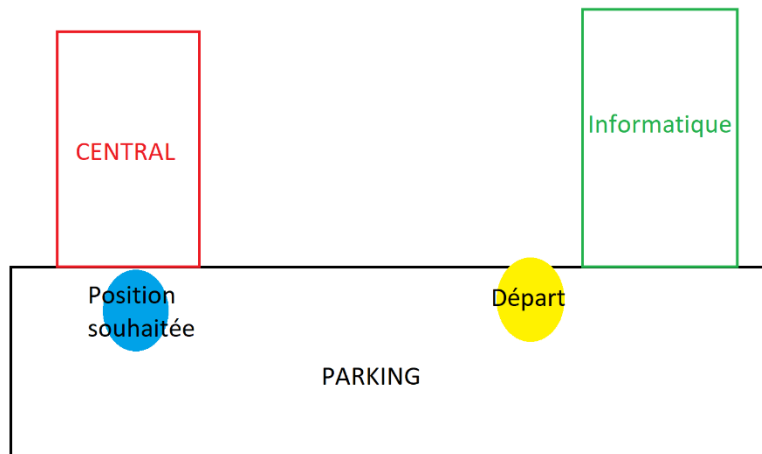


Figure 18 : schéma d'un exemple de problème de navigation

Afin de résoudre cette nouvelle contrainte, nous avons dû créer plusieurs fichiers HTML, correspondant aux différentes possibilités d'entrées et de sorties entre les zones.

Voici donc la liste de tous les fichiers HTML que nous avons créé pour palier à ce problème :

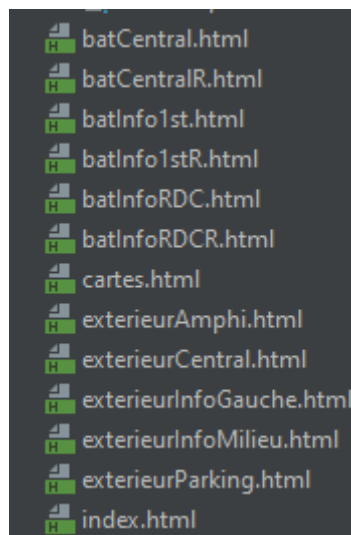


Figure 19 : liste des fichiers HTML

Les duplicatas des fichiers HTML sont presque identiques, seuls l'ordre de chargement des données ainsi que leur groupe de départ diffèrent des uns aux autres.

Nous avons résolu le problème de chargement ainsi, augmentant nettement la complexité de notre code. Une bonne solution alternative aurait été de transformer chaque groupe en un fichier HTML. Nous aurions pu encore plus séparer les temps de chargement, et totalement éviter le dernier problème évoqué.

3. Autres améliorations : son, écran de chargement, ...

Nous avons procédé à d'autres modifications plus légères de qualité de vie et de dynamisation du panorama.

Nous avons ajouté un son de transition entre chaque groupe au sein d'une même zone. Pour faire ceci, nous chargeons un fichier mp3 dans les <a-assets>, et nous venons lier ce fichier ainsi que l'évènement « click », pour déclencher le son au moment voulu (voir figure n°19)

```
<a-image spot="linkto: #parking; spotgroup: group-parking"  
position="-3.72751 0.89279 -6.31928" src="#fleche"  
sound="src:#boup; on: click" color="#FFFFFF"
```

Figure 19 : code de l'utilisation du son

Ensuite, nous avons modifié l'écran de chargement pour que ses couleurs correspondent à notre générateur de panorama, que nous détaillerons plus tard. Nous forçons également le rendu graphique de la scène au bout de 30 secondes, même si le navigateur n'a pas téléchargé toutes les données nécessaires, afin d'éviter de provoquer l'agacement ou l'incompréhension de l'utilisateur.

```
<a-scene id="notreScene" loading-screen="dotsColor:#FED700; backgroundColor: #303030">  
<a-assets timeout="30000">
```

Figure 20 : code du chargement

L'attribut loading-screen de l'<a-scene> altère la couleur de l'écran de chargement, et l'attribut timeout des <a-assets> force le rendu pendant la suite du téléchargement des données.

Pour finir, nous avons ajouté un petit tutoriel, qui est notre index.html. Dans ce fichier, on y retrouve un panneau qui explique brièvement le fonctionnement de notre panorama, afin de permettre aux utilisateurs débutants une prise en main rapide et simple du site.

V. Générateur

a. Objectifs

La première tâche à réaliser pour l'élaboration du générateur a été l'analyse de l'application. Nous avons commencé par élaborer le diagramme de cas d'utilisation, pour décrire les objectifs à atteindre sur ce projet.

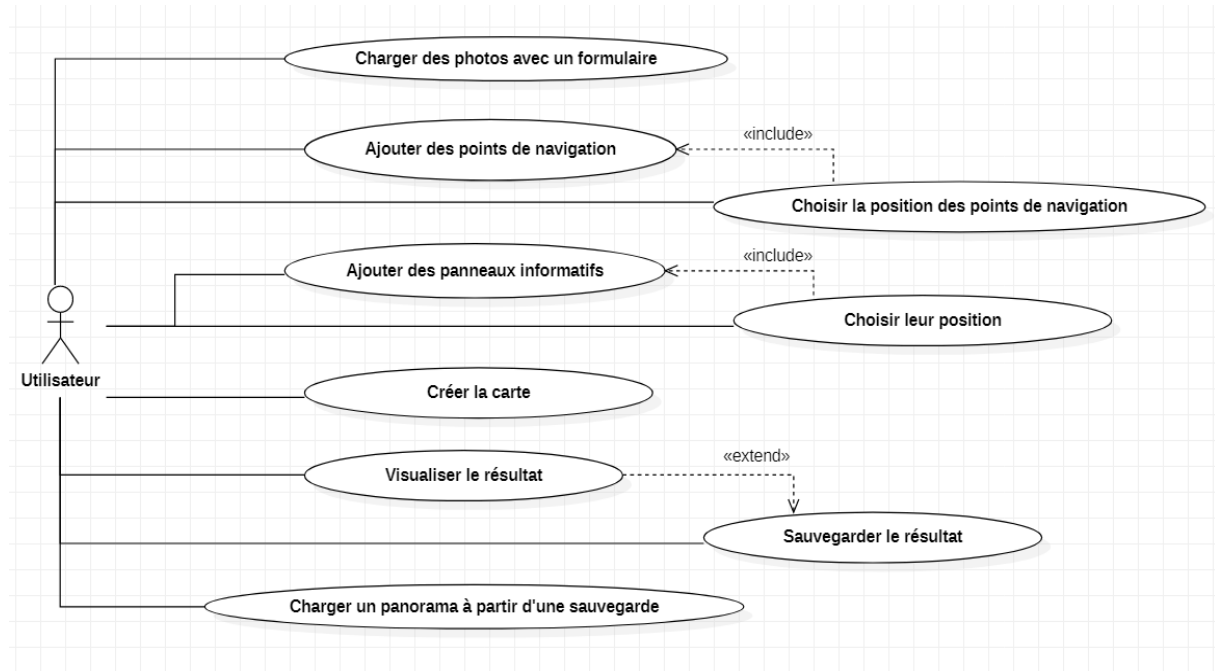


Figure 21 : diagramme de cas d'utilisation du générateur

Les objectifs pour le générateur peuvent être regroupés en 4 aspects principaux :

- Uploader : l'utilisateur doit être capable de charger ses photos 360° à partir de son navigateur.
- Editer : après avoir chargé ses photos, l'utilisateur va pouvoir placer ses propres panneaux et points de navigation, ainsi qu'une carte qu'il devra éditer.
- Visualiser : il peut visualiser ses scènes 360° et naviguer à l'intérieur de son panorama.
- Sauvegarder : pour finir, il peut sauvegarder son panorama sous forme de fichier HTML pour le conserver.

Pour comprendre et mieux définir l'enchaînement des actions de l'utilisateur, nous avons également conçu le diagramme d'activité du générateur, qui permet de visualiser d'avantage le processus de création de son propre panorama.

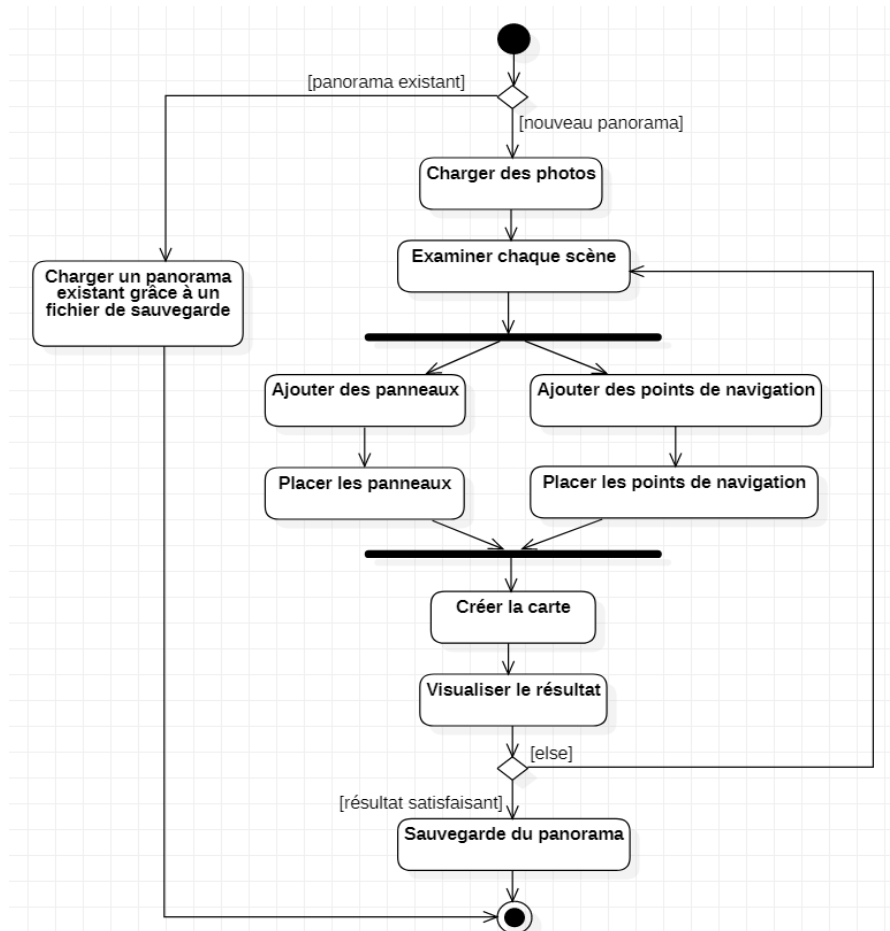


Figure 22 : diagramme d'activité du générateur

b. Interfaces utilisateurs

L'interface utilisateur de l'application, c'est à dire le site web, a été conçu et réfléchi pour être ergonomique et facile à prendre en main pour les utilisateurs. Pour cela, nous avons réalisé des maquettes, des tutoriels et définit une charte graphique pour le site.

Au niveau de l'accueil du site web, il y a une barre de navigation qui permet de retrouver les différentes rubriques de la page. On peut donc accéder à ces rubriques en cliquant sur leurs intitulés, mais aussi en scrollant la page d'accueil.

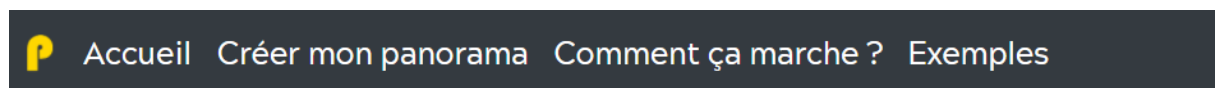


Figure 23.a : extrait du site web, barre de navigation

La première rubrique, intitulé « Créer mon panorama » est composée de quelques phrases explicatives, puis d'un bouton pour accéder aux formulaires de création d'un générateur.



Figure 23.b : extrait du site web, rubrique « Créer mon panorama »

La deuxième rubrique, « Comment ça marche ? » reprend les grandes lignes du processus de création de panorama.



Figure 23.b : extrait du site web, rubrique « Comment ça marche ? »

Puis pour finir la page, la dernière rubrique est celle nommée « Exemples ». Elle montre un exemple de panorama déjà créé grâce au site web dans un carrousel dynamique de photos.



Figure 23.c : extrait du site web, rubrique « Exemples »

Pour simplifier la prise en main de l'outil par les utilisateurs, nous avons également réalisé un tutoriel, disponible au début du processus de création d'un panorama.

TUTORIEL : CRÉER MA VISITE

Pour placer une flèche :

J'appuie sur la touche **F** 1
de mon clavier

2 Je la positionne grâce aux
flèches de mon clavier

F ↑ ↓ ← →

Pour placer un panneau :

J'appuie sur la touche **P** 3
de mon clavier

4 Je la positionne grâce aux
flèches de mon clavier

↑ ↓ ← →

Pour passer à la photo suivante, j'appuie sur la flèche de mon clavier

Figure 23.d : extrait du site web, tutoriel pour les utilisateurs

c. Diagramme de classe

Tout d'abord nous avons réalisé une analyse pour pouvoir faire un diagramme de classe et ainsi mieux comprendre comment notre projet de générateur allait être structuré :

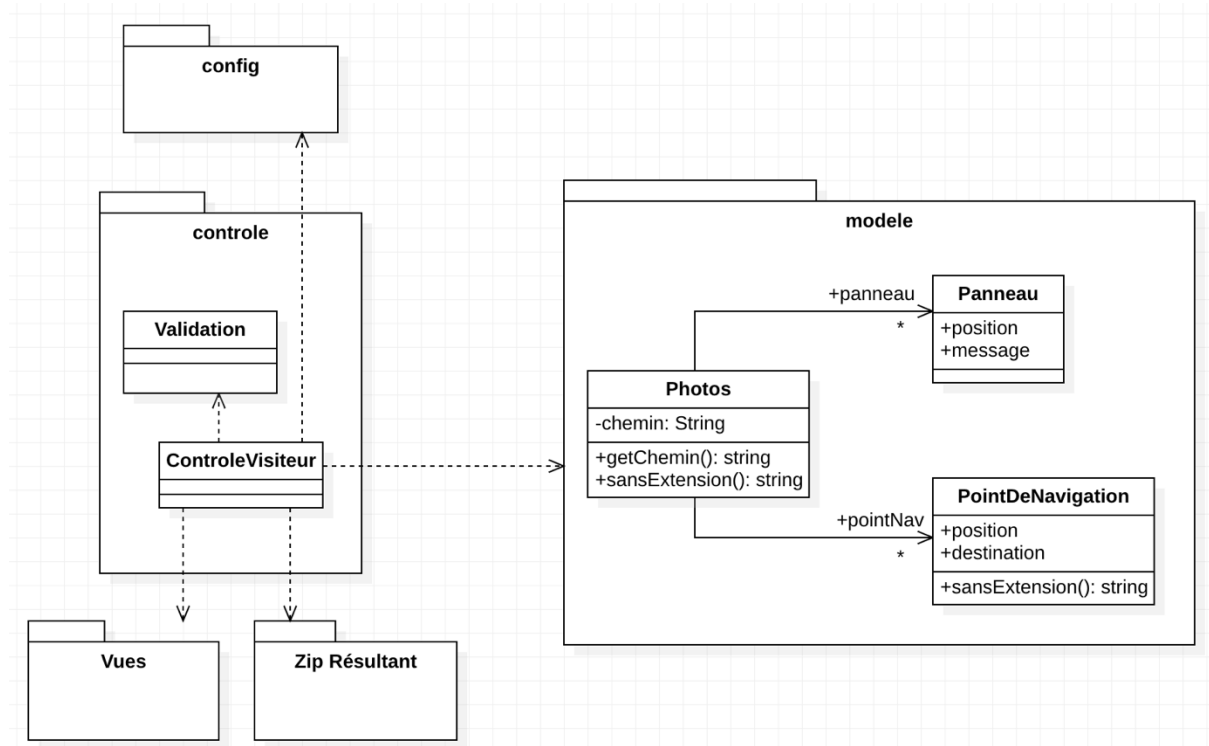


Figure 24 : Diagramme de classe du Générateur

Nous avons un modèle simple qui contient une classe Photos qui comporte elle-même deux autres classes qui sont Panneau et PointDeNavigation. On utilise un fichier de configuration afin de créer des variables globales. Nous avons créé un unique contrôleur pour l'ensemble du générateur car il n'y a pas d'acteurs différents mais seulement le visiteur. Le contrôleur possède une classe validation pour valider les actions et nettoyer une variable qu'on lui passe. Et finalement il y a un dossier contenant les différentes vues de notre site.

Nous avons décidé de rendre persistante la liste des images en utilisant une variable de session⁵ qui possède une liste d'objets photos. Chacun de ces objets a pour attribut une liste de panneaux et une liste de points de navigation ainsi qu'une chaîne de caractères correspondant à leur chemin dans l'arborescence de fichiers pour être facilement accessibles et utilisables.

```
private string $chemin;  
public array $panneau = [];  
public array $pointNav = [];
```

Figure 25 : code des attributs de l'objet Photo

⁵ Ce terme est défini dans le lexique en fin de rapport.

Nous avons seulement besoin de ces trois éléments pour pouvoir générer un fichier de format HTML. Un panorama est en effet un assemblage de photos entre elles. Il faut donc savoir quelle photo mène à quelle autre photo, ce qui correspond aux points de navigation. De plus, on peut rajouter des informations sur un panneau pour pouvoir décrire la photo sur laquelle on se trouve. C'est pourquoi notre modèle n'est composé que de ces trois éléments.

Le nom du panorama qui a été précisé par l'utilisateur dans le formulaire de chargement des photos est également sauvegardé en session. Ce nom est ensuite réutilisé pour nommer le dossier compressé contenant le résultat afin qu'il soit clairement identifiable par l'utilisateur.

d. Edition des scènes

1. Créer une scène interactive

Dans un premier temps, l'utilisateur peut charger ses photos. Grâce à un formulaire, le générateur récupère ces photos et les copie dans un fichier *photosUpload* ainsi que dans une variable de session.

```
if($cpt == $total_fichier_upload){
    $dir_nom = __DIR__."/../photosUpload";
    $dir = opendir($dir_nom) or die('Erreur de listage : le répertoire n\'existe pas');
    $lesPhotos = [];

    while(false !== ($element = readdir($dir)) {
        if($element != '.' && $element != '..') {
            $cheminPhoto = "";
            $cheminPhoto .= $element;
            $photo = new Photos($cheminPhoto);
            $lesPhotos [] = $photo;
        }
    }
    $_SESSION['photos']=$lesPhotos;

    $_SESSION['titre']=$nomProjet;
```

Figure 26 : code du chargement des photos de l'utilisateur

Chaque fichier du dossier *photosUpload* est ainsi transformé en objet Photos avant d'être ajouté à une collection de photos qui sera sauvegardée en session.

Après avoir validé le formulaire de chargement des photos, l'utilisateur se voit proposer un second formulaire où il choisit quelle sera la première photo traitée. La photo choisie est ensuite déplacée en tête de la liste des photos puis est ouverte grâce à un fichier HTML intégrant des balises A-Frame.

L'utilisateur peut ainsi visualiser sa photo comme dans un panorama.

```

<a-scene id="notreScene">
  <a-assets>
    
  </a-assets>

  <a-sky id="skybox" src="#photo1"></a-sky>

```

Figure 27 : code de l'affichage de la photo en cours d'édition

On envoie le chemin de la photo à la vue, qui l'utilise comme `<a-sky>` de la vue, c'est-à-dire comme image de fond de la sphère virtuelle créée par A-Frame.

Lorsque l'édition de cette première photo est terminée, il peut passer à la suivante en appuyant sur une touche de son clavier. Cela déclenche l'appel d'une fonction JavaScript qui va sauvegarder tous les éléments de la scène en cours dans un formulaire qui va ensuite être récupéré en back-end.

Lorsque la sauvegarde est terminée, le générateur affiche la photo suivante en parcourant simplement la liste des photos sauvegardées en session grâce à une variable `index` elle aussi sauvegardée en session.

```

$lesPhotos[$indexPhoto-1] = $photoTerminee;

if($indexPhoto==count($lesPhotos)){
  unset($_SESSION['compteur']);
  $this->formulaireAjoutPhotoCarte();
  return;
}

$photoEnCours=$lesPhotos[$indexPhoto];
$_SESSION['compteur'] = $indexPhoto + 1;
$_SESSION['photos'] = $lesPhotos;

require($chemin.$lesVues['debutpano']);

```

Figure 28 : code du parcours des photos à éditer

Si la photo qui vient d'être éditée (ici `$photoTerminee`) est la dernière, alors l'utilisateur passe à l'étape suivante, l'ajout de la carte (détaillée dans la partie 3. Création de la carte ci-dessous). Sinon, la même vue est rappelée avec la photo suivante qui peut être éditée à son tour.

2. Générer des éléments grâce au clavier

Lors de l'édition d'une scène, l'utilisateur peut ajouter sur celle-ci des panneaux informatifs ainsi que des points de navigation. Pour cela, nous avons intégré à la vue d'édition des scènes un fichier

JavaScript qui écoute les événements envoyés par le clavier. Si l'utilisateur appuie sur certaines touches, certaines actions se produisent.

```
window.addEventListener( type: 'keydown', listener: (event : KeyboardEvent ) => {  
  switch(event.key) {  
    case 'p' :...  
    case 'n' :...  
    case 'a' :...  
    case 'h' :...  
    case 'ArrowLeft' :...  
    case 'ArrowRight' :...  
    case 'j' :...  
  }  
}
```

Figure 29 : code de la récupération de l'action de l'utilisateur

Ainsi, pour ajouter un panneau informatif, l'utilisateur appuie sur la touche **P** et une fenêtre apparaît à l'écran en lui demandant d'indiquer le texte qui sera présent sur le panneau. Pour un point de navigation, un appuie sur la touche **N** ouvre une fenêtre permettant d'indiquer le nom de l'image de destination.

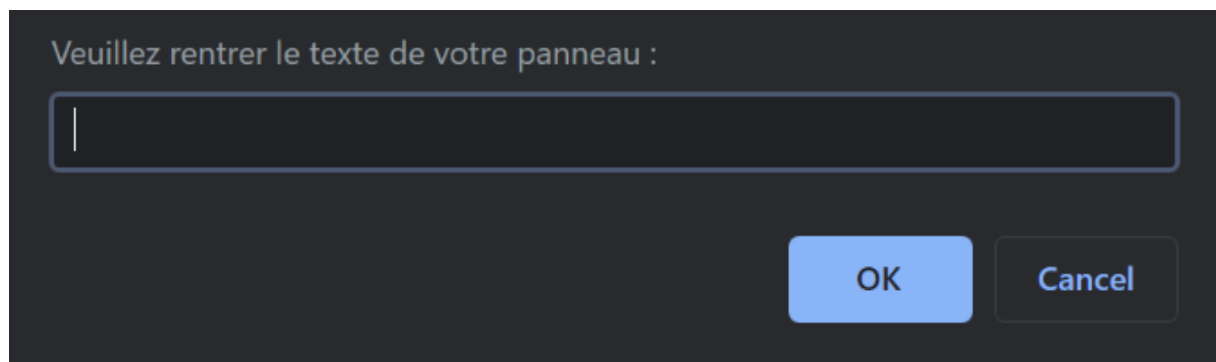


Figure 30.a : formulaire d'ajout d'un panneau

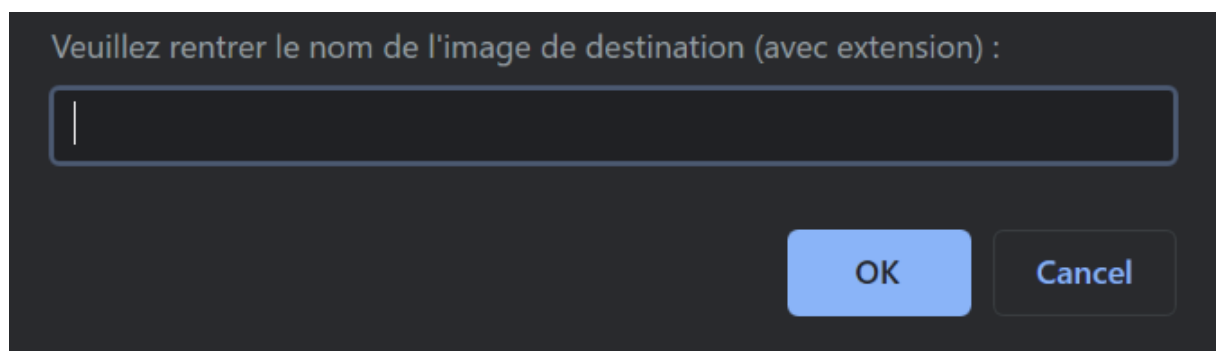


Figure 30.b : formulaire d'ajout d'un point de navigation

Après avoir validé le texte, l'élément apparaît à l'écran et l'utilisateur peut alors le déplacer.

Sur les scènes créées avec A-Frame, chaque élément peut être disposé selon trois axes comme le montre la figure suivante :

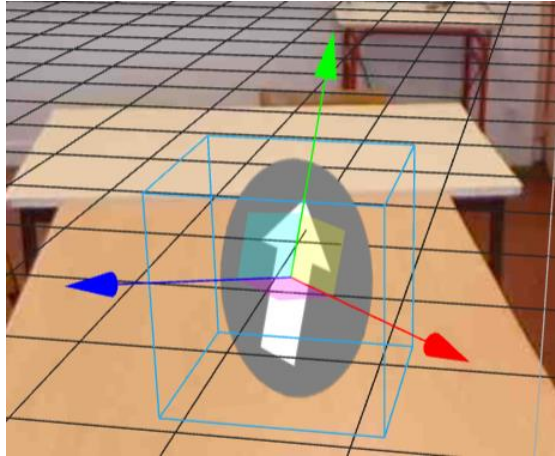


Figure 31 : visualisation des axes d'un élément

Afin de simplifier le processus du placement d'un élément, nous avons décidé de ne permettre à l'utilisateur de déplacer l'élément que sur un seul axe à la fois. Avec les flèches de son clavier, l'utilisateur déplace son point ou son panneau selon un axe, puis peut changer d'axe avec la touche **A**. En appuyant plusieurs fois sur cette touche, l'utilisateur peut changer indéfiniment l'axe en cours d'utilisation pour pouvoir perfectionner la position de son élément.

La rotation des éléments n'est pas modifiable puisque chaque panneau ou point de navigation sur le panorama final sera par défaut orienté pour regarder la caméra (c'est-à-dire l'utilisateur).

Une fois que l'utilisateur a terminé d'ajouter et de placer tous les éléments, il peut déclencher la sauvegarde de la scène avec un appui sur la touche **J** de son clavier. Cela va appeler la fonction de sauvegarde du script JS qui lit chaque élément composant la scène et inscrit les éléments importants dans un formulaire. Pour un panneau, le générateur sauvegarde son texte et sa position. Pour un point de navigation, c'est le nom de son image de destination et sa position qui sont sauvegardés.

Pour ce faire, chaque information est ajoutée comme saisie dans un formulaire.

```

case 'j' : //SAUVEGARDE DES ELEMENTS CREES
    let nbElements = 0;
    form=document.getElementById( elementId: "notreFormulaire");
    let panneaux = Array.from(document.getElementsByClassName( className: "panneau"));
    let points = Array.from(document.getElementsByClassName( className: "point"));
    lesElements = panneaux.concat(points);
    let count = lesElements.length;
    for (let i=0; i<count; i++) {
        sauvegarde(lesElements[i],nbElements);
        nbElements += 3;
    }
    let info = document.getElementById( elementId: "nbElements");
    info.setAttribute( qualifiedName: "value",nbElements.toString());
    form.submit();

```

Figure 32 : code de sauvegarde des éléments

Dans un premier temps, on récupère tous les enfants de la scène étant de classe « panneau » ou « point » puis on les sauvegarde un à un.

```

if (item.className === "panneau") {
    input.setAttribute(qualifiedName: "value", item.getAttribute(qualifiedName: "text").value);
    input2.setAttribute(qualifiedName: "value", value: "panneau");
}
else {
    input.setAttribute(qualifiedName: "value", item.getAttribute(qualifiedName: "id"));
    input2.setAttribute(qualifiedName: "value", value: "point");
}
input.setAttribute(qualifiedName: "name", value: "item" + (nb + 1));
form.appendChild(input);
form.appendChild(input2);
input = document.createElement(tagName: "input");
input.setAttribute(qualifiedName: "type", value: "text");
coord = item.getAttribute(qualifiedName: "position");
coord = coord["x"] + " " + coord["y"] + " " + coord["z"];
input.setAttribute(qualifiedName: "value", coord);
input.setAttribute(qualifiedName: "name", value: "item" + (nb + 2));
form.appendChild(input);

```

Figure 33 : code de sauvegarde des attributs d'un élément

Lors de la sauvegarde, on enregistre le texte pour un panneau et le nom de l'image de destination (ici passée en identifiant du point de navigation afin que l'utilisateur ne puisse pas circuler de scène en scène lors de l'édition) ainsi que la position de chaque élément et un nom s'incrémentant pour chaque donnée pour permettre la récupération des éléments par la suite.

Une fois le formulaire envoyé par le script JS, il est récupéré côté back-end.

```

if ($type == 'panneau') {
    $panneau = new Panneau($valeur, $coord);
    $photoTerminee->panneau[] = $panneau;
}
elseif ($type == 'point') {
    $point = new PointDeNavigation($valeur, $coord);
    $photoTerminee->pointNav[] = $point;
}

```

Figure 34 : code de récupération des attributs des éléments

Une boucle permet ensuite de parcourir chaque élément et de créer un objet correspondant au type de l'élément (panneau ou point de navigation) pour les ajouter aux attributs de la photo qui vient d'être éditée. Ainsi, tous les éléments créés lors de l'édition de la scène et envoyés via le formulaire sont sauvegardés en session car contenus dans l'objet photo.

3. Création de la carte

Une fois chaque scène éditée, l'utilisateur est invité à déposer la photo de sa carte dans un nouveau formulaire. Le fichier est à nouveau enregistré dans le dossier *photosUpload* avec les autres photos et un objet Photos est créé dont les attributs permettront de sauvegarder les éléments de la carte. Cet objet est également sauvegardé en session mais dans une variable à part afin de pouvoir être traité différemment des autres photos lors de la sauvegarde.

```
$carte = new Photos($fileName);  
$_SESSION['carte'] = $carte;
```

Figure 35 : code de création et sauvegarde de l'objet carte

La carte est ensuite affichée comme un plan vertical sur une scène vide afin de ne pas déformer la carte qui n'est pas une photo à 360°. L'utilisateur peut ensuite ajouter des points de navigations comme lors de l'édition des scènes et les placer à sa guise.

Une fois tous les points placés, l'utilisateur peut sauvegarder la carte, dont les points seront sauvegardés de la même manière que pour une scène, et termine ainsi son panorama.

L'icône permettant d'accéder à la carte n'est pas ajoutée lors de l'édition des scènes mais est créée lors de la génération du résultat avec une position fixe qui est la même sur chaque scène.

e. Sauvegarde et génération HTML

1. Sauvegarde au milieu de la création

Comme précisé dans notre WBS, nous avons prévu de pouvoir faire une sauvegarde au milieu de l'édition d'un panorama pour pouvoir permettre à un utilisateur de sauvegarder sa progression et de revenir dessus plus tard.

Pour commencer, il a fallu décider de la méthode à adapter et du type de fichier que nous allons utiliser. Nous avons choisi la sauvegarde en JSON. L'utilisateur pourra donc récupérer un fichier JSON avec le modèle sauvegardé à l'intérieur. En attendant, on laisse les photos en sauvegarde sur le serveur. Pour reprendre l'édition d'un panorama, il suffira de rentrer dans un formulaire le fichier JSON et après un simple parcours on pourra récupérer la liste de photo ainsi que tous les attributs qui leur étaient associées. Ensuite l'édition reprend comme s'il n'y avait pas eu d'interruption.

Malheureusement nous n'avons pas trouvé le temps de programmer et de mettre en place cette fonctionnalité dans le générateur. Nous avons préféré nous concentrer sur la partie génération à la fin du panorama comme indiqué dans l'analyse des écarts.

2. Génération d'un fichier HTML

La génération du fichier HTML est réalisée après la sauvegarde de la carte. Pour ce faire, nous avons analysé la structure de notre fichier HTML de la visite de l'IUT pour comprendre quels paramètres changeaient selon le panorama et lesquels étaient présents dans tous les cas. Nous avons remarqué que la caméra et l'entête du fichier n'avaient pas besoin d'être modifiées. Nous avons ensuite récupéré les éléments propres au panorama sauvegardés en session. Par la suite, nous avons créé plusieurs variables qui correspondent aux différentes parties du fichier HTML. En premier il y a l'« entête », qui contient les balises HTML obligatoirement présentes en début de fichier et les appels aux différents scripts JS nécessaires au fonctionnement du panorama. Puis dans une variable « assets » nous plaçons les différentes images grâce à une boucle qui parcourt la liste des photos. On rajoute ensuite la variable « camera » ainsi que la « skybox ».

```
$sauvegarde = $entete . $asset . $skyBox . $camera . $lesGroupes . $fin;  
  
file_put_contents( filename: "index.html", $sauvegarde);
```

Figure 36 : extrait du code de la génération du html résultat

Ensuite, nous avons inséré dans une variable « lesGroupes » les différentes entités d'A-frame qui composent une image avec des attributs du type : « <a-entity id= group » qui est l'entité associée à une image. À l'intérieur de cette boucle, on parcourt la liste des points de navigation de la photo ainsi que ses panneaux informatifs pour ajouter à chaque photo les éléments qui lui sont rattachés.

Pour finir, on crée une archive (un dossier compressé) en format ZIP qui contient les icônes et script JS commun à tous les panoramas, toutes les photos du panorama, la carte ainsi que le fichier index.html généré par notre application web.

```
$zip = new ZipArchive();  
$ret = $zip->open( filename: Validation::val_texte($_SESSION['titre']).'.zip',  
    flags: ZipArchive::CREATE | ZipArchive::OVERWRITE);  
...  
$options = array('add_path' => 'photos/', 'remove_all_path' => TRUE);  
$zip->addGlob( pattern: './photosUpload/*.{png,PNG,jpg,JPG}', flags: GLOB_BRACE, $options);
```

Figure 37 : extrait du code de la création du ZIP

Le dossier est ensuite téléchargé par le client et la session est réinitialisée. Si l'utilisateur veut utiliser son fichier, il lui suffira de lancer le fichier index.html sur son propre serveur web.

BILAN TECHNIQUE

Nos réalisations sont deux sites web, le panorama de l'IUT et le générateur de panorama, utilisant le framework A-Frame. Le panorama de l'IUT ainsi que ceux générés par notre générateur sont compatibles avec les systèmes de Réalité Virtuelle sur ordinateur et sur mobile. Bien que nos sites soient fonctionnels, de nombreuses améliorations sont encore possibles.

a. Améliorations possibles du panorama de l'IUT

Nous avons vu que nous avons déjà apporté de nombreuses améliorations au panorama depuis le mois de janvier, cependant il reste des points à améliorer :

- La navigation n'est pas très intuitive à cause d'un problème d'orientation des images, lorsque nous passons d'un groupe à l'autre, nous ne gardons pas la même orientation dans l'espace, ce qui peut être perturbant et contre-intuitif lors de la visite, ce problème peut être résolu en modifiant les images pour qu'elles aient toutes la même origine.
- Il aurait été judicieux de transformer chaque groupe en un fichier HTML. Nous aurions pu encore plus séparer les temps de chargement, et totalement éviter le problème de duplication des fichiers html.
- L'utilisation des manettes d'un casque VR pourrait être un ajout intéressant afin de proposer plus de diversité dans la façon de naviguer dans le panorama.

b. Améliorations possibles du générateur

Notre générateur de panorama pourrait avoir de nouvelles fonctionnalités telles que :

- La sauvegarde en cours de création du panorama en JSON afin de pouvoir recharger des panoramas non terminés et leur apporter des modifications.
- La visualisation d'un panorama créé sur ce site.

Ces améliorations n'ont pas été faites principalement par manque de temps, ces modifications étant toutes chronophages.

CONCLUSION

Ce travail a été pour nous l'occasion de travailler dans une équipe beaucoup plus nombreuse que ce dont on avait l'habitude à l'IUT mais qui se rapproche plus de la taille des équipes que l'on peut retrouver en entreprise. Nous avons dû faire un gros travail sur la répartition des tâches et la communication à l'intérieur du groupe, ce qui était nouveau pour nous.

Cela nous a également permis de prendre conscience de l'importance de la gestion de projet, il est indispensable d'avoir établi des prévisionnels avec le temps de travail prévu pour mener à terme le projet dans les délais imposés. La notion de compréhension du besoin et de satisfaction du client a également été l'une des compétences que nous avons acquises durant ce travail. Nous avons fait de nombreuses réunions durant lesquelles nous avons pris des notes pour nous adapter au mieux par rapport aux remarques que le client (ici notre tuteur) avait sur le travail présenté.

Nous avons aussi appris à nous documenter par nous-même, et l'apprentissage autonome de nouveaux langages et de nouvelles technologies est une part essentielle du travail d'un développeur, car ses compétences doivent régulièrement être mises à niveau.

Pour conclure, ce projet nous a apporté une expérience enrichissante et de nombreux apprentissages qui nous ont permis d'acquérir des compétences nécessaires et indispensables dans le monde professionnel : la communication, l'organisation et la rigueur.

RÉSUMÉ EN ANGLAIS

Our project is part of our two-year university diploma in Computer Science at the University Institute of Technology (IUT), UCA in Clermont-Ferrand and consists of two parts. On one hand, a virtual visit of the IUT of Clermont-Ferrand and on the other hand, an online generator allowing people to create their own panorama.

Our team is composed of five second-year students: Clément Ferrere, Enzo Mazella, Victor Mommaliier, Clara Poncet and Lucile Velut, and was supervised by our teacher Mr. Salva. Enzo left our team in January after resigning from the IUT.

The visit of the IUT is an immersive experience where anyone can walk through the computer science department as if they were there. It also has informative signs to indicate where the user is and where they can go. We also added navigation arrows allowing the user to go from one scene to the other. In addition, the panorama has a map which offers a global view of the IUT and allows the visitor to jump faster from one point to another.

To enhance the ergonomic aspects of the visit, we added animations and sound to the panorama and made improvements to reduce the loading time as the photographs of the IUT were large files.

Whereas the IUT visit is mainly aimed towards prospective students, we hope to attract a wider audience with our generator. We wanted it to be accessible and easy to use so that anyone having 360° photographs could use it to create their own VR panorama.

By using the generator, the user can upload their photographs and edit each scene. They can add signs and arrows for each photograph, and then add a map which will offer a wider view of their whole panorama and allow easy navigation.

Once each photograph has been edited and the map has been created, the user can save their panorama by downloading a zipped file containing an HTML panorama and all the necessary photographs and icons.

The generator is functional but could be improved with further functionalities, such as the upload of a panorama to visualize it or the possibility to save a work-in-progress panorama to finish it later.

The interactive visit has been made using the framework A-Frame, which is an open-source HTML framework designed to allow web applications to easily offer a VR experience. We also used A-Frame for the edit view of the generator as well as the result. The rest of the generator has been made using PHP and JavaScript.

BIBLIOGRAPHIE

Documentation A-Frame :

<https://aframe.io/docs/1.0.0/introduction>

Tutoriels et exemple de panorama A-Frame :

<https://caseyee.github.io/aframe-360-tour>

<https://ambiguous-diploma.glitch.me>

<https://medium.com/designerrs/how-to-create-a-virtual-tour-using-a-frame-164941fea573>

Documentation PHP :

<https://www.php.net/docs.php>

Cours de PHP :

Cours de Programmation Web Côté Serveur – DUT 2^{ème} Année – M. Salva

Documentation JavaScript :

<https://developer.mozilla.org/fr/docs/Web/JavaScript>

LEXIQUE

- RV : Réalité virtuelle, technologie qui consiste à simuler la présence de l'utilisateur dans un autre lieu créé virtuellement.
- framework : désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (source : wikipedia)
- PHP : Hypertext Preprocessor, langage de programmation orienté objet principalement utilisé pour produire des pages web dynamiques en utilisant un serveur HTTP
- fish-eye : Un objectif fish-eye est un objectif photographique ayant pour particularité une distance focale très courte et donc un angle de champ très grand, jusqu'à 180° dans la diagonale, voire dans toute l'image.
- HTML : HyperText Markup Language, langage de balisage conçu pour réaliser des pages web.
- CSS : Cascading Style Sheets, ou feuilles de style en cascade, langage informatique décrivant la présentation des documents HTML et donc l'apparence des pages web.
- Javascript (JS) : langage de programmation de scripts utilisé pour réaliser des pages web interactives.
- back-end : ou arrière-plan, terme informatique désignant l'étage d'accès aux données auquel l'utilisateur n'a pas accès.
- front-end : ou frontal, terme informatique désignant ce que l'utilisateur perçoit du programme.
- balise : suite de caractères délimitant une section de code ou marquant une position précise.
- session / variable de session : une session est lancée à chaque démarrage d'une application web. Elle permet de stocker des informations propres à l'utilisateur sur le serveur à l'aide de variables de sessions qui peuvent être utilisées à tout moment tant que la session est lancée.

ANNEXES

1. WBS

1. Panorama

- 1.1. Analyse
 - 1.1.1 Diagramme de cas d'utilisation
 - 1.1.2 Diagramme d'objets
- 1.2 Codage du panorama
 - 1.2.1 Codage des scènes
 - 1.2.2 Codage des points de navigation
 - 1.2.3 Ajout des panneaux informatifs
- 1.3 Création de la carte
 - 1.3.1 Analyse ergonomique
 - 1.3.2 Codage de la carte
- 1.4 Gestion de la portabilité

2. Générateur

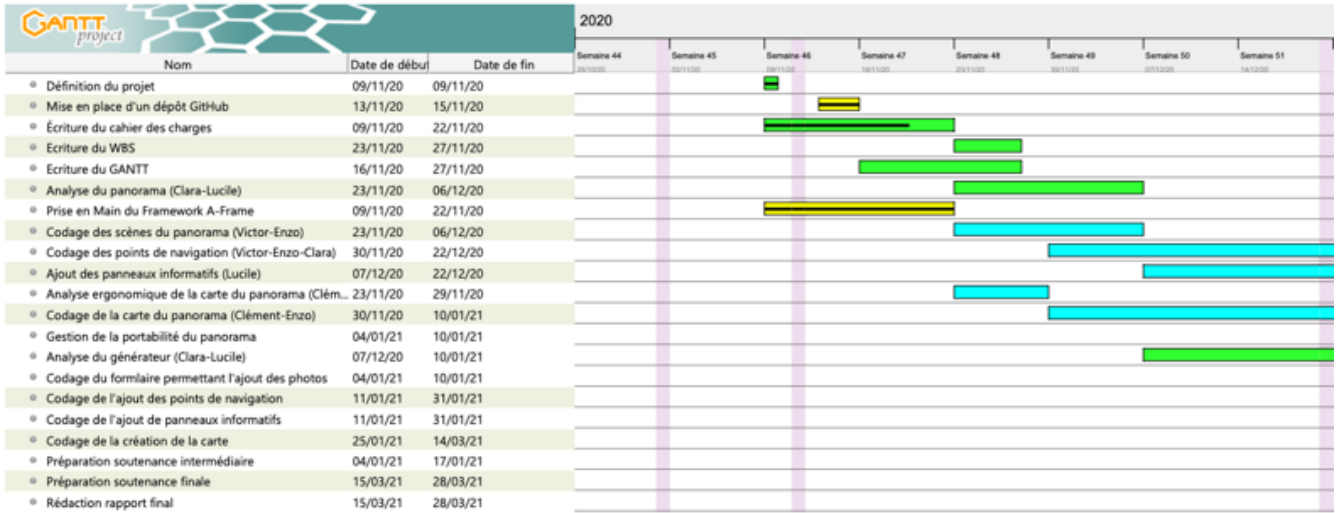
- 2.1 Analyse
 - 2.1.1 Diagramme de cas d'utilisation
- 2.2 Codage du générateur
 - 2.2.1 Codage du formulaire pour le chargement des photos
 - 2.2.2 Codage de l'ajout des points de navigation
 - 2.2.3 Codage de l'ajout de panneaux informatifs
 - 2.2.4 Codage de la création de la carte
- 2.3 Codage de la sauvegarde du résultat utilisable sur 3 plateformes

3. Rapports & Soutenances

- 3.1 Rédaction des documents prévisionnels
 - 3.1.1 Ecriture du cahier des charges
 - 3.1.2 Ecriture du WBS
 - 3.1.3 Ecriture du GANTT
- 3.2 Préparation des soutenances
 - 3.2.1 Préparation de la soutenance du 19 janvier
 - 3.2.2 Préparation de la soutenance du 29 mars
- 3.3 Rédaction du rapport finale

2. Gantt prévisionnel

Diagramme de Gantt



3. Gantt réel

| Nom | Date de début | Date de fin |
|--|---------------|-------------|
| Définition du projet | 09/11/2020 | 09/11/2020 |
| Écriture du cahier des charges | 09/11/2020 | 17/11/2020 |
| Mise en place d'un dépôt GitHub | 13/11/2020 | 15/11/2020 |
| Prise en Main du Framework A-Frame | 09/11/2020 | 22/11/2020 |
| Analyse du panorama | 23/11/2020 | 29/11/2020 |
| Analyse ergonomique de la carte du panorama | 23/11/2020 | 29/11/2020 |
| Ecriture du WBS | 23/11/2020 | 27/11/2020 |
| Ecriture du GANTT | 16/11/2020 | 27/11/2020 |
| Codage des scènes du panorama | 23/11/2020 | 29/11/2020 |
| Ajout des panneaux informatifs | 07/12/2020 | 13/12/2020 |
| Codage des points de navigation | 02/12/2020 | 18/12/2020 |
| Analyse du générateur | 23/12/2020 | 14/01/2021 |
| Codage de la carte du panorama | 30/11/2020 | 22/12/2020 |
| Gestion de la portabilité du panorama | 26/11/2020 | 04/12/2020 |
| Création des vues du Générateur | 22/12/2020 | 10/01/2021 |
| Préparation soutenance intermédiaire | 04/01/2021 | 10/01/2021 |
| Codage du formulaire permettant l'ajout des photos | 22/01/2021 | 11/02/2021 |
| Codage de l'ajout des points de navigation | 22/01/2021 | 11/02/2021 |
| Codage de l'ajout de panneaux informatifs | 12/02/2021 | 03/03/2021 |
| Codage de la création de la carte | 12/02/2021 | 03/03/2021 |
| Amélioration du Panorama | 18/01/2021 | 03/03/2021 |
| Codage de la sauvegarde et de la génération | 01/03/2021 | 18/03/2021 |
| Rédaction rapport final | 15/03/2021 | 28/03/2021 |
| Préparation soutenance finale | 15/03/2021 | 28/03/2021 |

