Développement d'une application Web de visualisation et d'analyse de champs électromagnétiques

Rapport de Stage de fin de licence Informatique
parcours Info

Gelles Julien

Tuteurs :
Guillaume Willefert
Francesco De Comité

19 Avril 2022 – 29 Juillet 2022

Entreprise : Luxondes, Avenue Pierre Brossolette, 59280 Armentières

Université : Université de Lille, Campus cité scientifique, 59655 Villeneuve-D'ascq

Remerciements

Mes remerciements vont à :

- **Jean-luc Darroman** et **Jean Rioult**,

  Responsables et fondateurs de l'entreprise Luxondes, pour leur accueil et l'opportunité qu'ils m'ont proposé.

- **Guillaume Willefert**,

  Tuteur entreprise, pour son accompagnement et son aide à travers ce projet, au quotidien.

- **Julien Wyllerman**,

  Ingénieur de recherche au centre de recherche en informatique CRISTAL, pour l'aide occasionnel sur le projet.

- **Francesco de comité**,

  Tuteur universitaire, pour son suivi de mon stage.

- **l'Université de Lille** et **l'équipe encadrant les stages**,

  pour l'encadrement et le suivi des étudiant en stage cette année, dont je fais parti.

Résumé

Ce rapport explique mon implication et ma participation dans le projet de Viewer Web au sein de l'entreprise Luxondes.

Je fais parti d'un groupe initialement de 3 personnes ; mes deux responsables ainsi que mon tuteur, mentionnés plus haut lors des remerciements.

Le projet consiste à concevoir et développer une application permettant d'afficher et d'analyser une représentation des champs électromagnétiques d'un appareil électronique. Tout ceci de la même façon que le Viewer Android sur lequel travaille mon tuteur entreprise.

L'application est conçu d'une part d'une interface web, d'une autre d'un serveur hébergeant la page.
Notre serveur utilise Node.js, un outil libre codé en JavaScript et orientée pour des applications en réseau et particulièrement Express.js, un framework JavaScript qui permet facilement le développement d'un serveur Node.js.

L'objectif à terme est d'avoir une application possédant tous les fonctionnalités demandés par un professionnel du secteur.

Summary

This report explains my involvement and my part in the Web Viewer project at Luxondes company.

I am part of an initial group of 3 people ; my two supervisors and my tutor, mentioned above about thanks.

The project involve designing and developing an application to display and analyse a representation of the electromagnetic fields of an electronic device.
All this in the same way as the Android Viewer on which my company tutor works.

The application is designed on the one hand with a web interface and on the other hand with a server hosting the internet page.
Our server uses Node.js, a free tool coded with JavaScript, oriented for network applications and especially Express.js, a JavaScript framework that easily allows the development of a Node.js server.

The long-term goal is to have an application with all the features that could be requested by a professional in the sector.

Table des matières

1. Introduction

Lors de mon choix d'orientation en études supérieur, j'avais pour ambition de me diriger vers des études me permettant de faire de la modélisation 3D, m'orienter dans un domaine artistique lié à l'informatique.
Je savais que j'aimais l'informatique et trouvais que le parcours d'une licence Informatique plutôt classique n'était pas forcément plus mal non plus.
Je me suis donc dirigé vers cette dernière tout en gardant en tête cet objectif, à terme, d'y lier l'imagerie et l'interaction que ce soit 2D ou 3D.

Dans le cadre de la Licence Informatique nous devons faire un stage d'au moins 3 mois. Cela permet de conclure nos acquis des diverses compétences demandées à l'issue de ces dernières années. C'est donc tout naturellement que je me suis dirigé vers l'entreprise Luxondes et ai accepté l'offre de stage pour concevoir un Viewer.

Le projet se réalise suite à une demande des utilisateurs pour une application Web qui possédera l'avantage de la souris et d'un écran bien plus large que celui d'un téléphone pour l'analyse des données. Le projet se basera sur l'application Android pour les objectifs et les implémentations à faire (cf. figure2, page3).

Dans ce rapport, nous commencerons par présenter le contexte du stage, l'entreprise et leurs activités, ma contribution à l'aide de ce projet pour enfin conclure et faire un bilan de ces derniers mois.

2. Contexte

Luxondes est une petite entreprise du nord de la France créée en 2011 centrée sur le domaine des ondes électromagnétiques. Née de deux experts passionnés de physique et d'innovation, Luxondes imagine, développe et fabrique des outils de mesure et de visualisation réalistes de champs électromagnétiques. Destinées aux mondes de l'industrie, de la recherche et de l'éducation, les solutions Luxondes ont pour vocation de rendre accessibles le traitement et l'analyse des ondes électromagnétiques.

L'entreprise se localise à la Ruche d'entreprise des deux lys à Armentières, un lieu crée pour accueillir et accompagner les créateurs d'entreprises afin de pérenniser leur activité.

Elle se situe par la même occasion au sein de l'Université Gustave Eiffel, Luxondes et cette dernière étant partenaires, non loin du bâtiment M5 et Cristal sur le campus cité scientifique de l'Université de Lille.
C'est par ailleurs à cet endroit que l'on se retrouve régulièrement.

Figure 1 : Evolution de Luxondes jusqu'en 2021

De 2011 à 2021, trois produits majeurs ont fait leur apparition au sein de Luxondes: le gyroscanfield, la dalle RWD et l'EM-Scanphone.

L'objectif étant de répondre aux besoins en CEM (Compatibilité ElectroMagnétique), de localiser la source des rayonnements et de pouvoir comparer deux mesures.

Comme énoncé plus haut, je fais principalement parti d'un groupe de 3 personnes ; mes deux responsables Jean Rioult et Jean-Luc Darroman ainsi que mon tuteur Guillaume Willefert.
Je travaille donc plus particulièrement avec mon tuteur au quotidien.

Le stage proposé était donc le suivant : Développement d'une application Web de visualisation et d'analyse de champs électromagnétiques, le tout en télétravail avec des réunions, plus ou moins hebdomadaires.

A la date actuelle (fin juin 2022), mon stage n'est pas terminé puisqu'il se finira en fin juillet 2022, ce rapport ne fera donc pas mention de l'entièreté du projet, mais seulement de l'avancée qu'il y a jusqu'à maintenant.

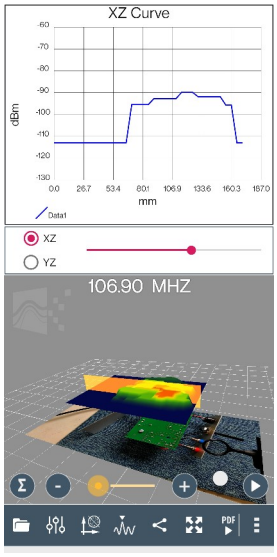

Figure 2 : Viewer Android

3. Contribution

Pour permettre le bon développement du projet, il était nécessaire de sous fragmenter le développement en différentes étapes, qui chacune constitue mon travail dans l'entreprise. Elles sont décrites en détail ci dessous :

a) Three.js

La première étape pour réaliser un Viewer en ligne était, de façon logique de pouvoir travailler dans un environnement permettant de modéliser en 3 dimensions.

De ce fait, j'ai appris à utiliser Three.js, une bibliothèque JavaScript pour créer des scènes 3D dans un navigateur web.
J'ai utilisé cette bibliothèque en premier temps pour représenter un cube puis très rapidement un plan sur lequel j'ai affiché une image.
J'ai par ailleurs décidé d'utiliser la *Nuit étoilée* de Van Gogh pour guise d'exemple.

Les visualisations de ces deux derniers rendus sont toujours observables aux adresses suivantes :
- le cube : https://jsfiddle.net/JulienGelles/gv9ob6ep/13/
- le plan : https://jsfiddle.net/JulienGelles/gv9ob6ep/99/

Ce travail fut très important car il a posé les bases de la modélisation 3D. Cela a notamment permis de connaître les fondamentaux tel que :

- La scène (l.4)*
- Le rendu (l.11-l.13)*
- La caméra (l.6-l.8)*
- Les contrôles (l.15-l.16)*
- Les meshs → objets/ polygones en 3D (l.21-l.25)*

*cf figure 2

```
import * as THREE from 'three';
import { OrbitControls } from 'https://unpkg.com/three/examples/jsm/controls/OrbitControls.js';

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 10000);
camera.position.z = 1;
camera.up.set( 0, 0, 1 );


const renderer = new THREE.WebGLRenderer( { antialias: true } );
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

const controls = new OrbitControls(camera, renderer.domElement);
controls.maxPolarAngle = Math.PI/2-0.01;


const img = 'https://lh6.ggpht.com/HlgucZ0ylJAfZgusynnUwxNIgIpShtNhShF559x3dRXiuy_UdP3UQVLYW6c=s1200';

const geometry = new THREE.PlaneGeometry( 1, 1 );
const texture = new THREE.TextureLoader().load( img );
const material = new THREE.MeshBasicMaterial( { map: texture , side: THREE.DoubleSide} );
const plane = new THREE.Mesh( geometry, material );
scene.add( plane );

const loop = function() {
  requestAnimationFrame(loop);
  renderer.render(scene, camera);
}

loop();
```

Figure 3 : Code de la mise en place des premiers rendus

J’ai par ailleurs effectué un travail non négligeable sur la configuration de la caméra et des contrôles ainsi que ses limitations.

b) Serveur local

Nous souhaitions pouvoir lancer notre script sur une page html hébergée localement. J'ai donc mis en place un serveur local, écoutant sur le port 3000, donc à l'adresse 'localhost:3000'.

Pour se faire j'ai utilisé le framework Express.js, conçu spécifiquement pour le développement d'un serveur Node.js.

Cela à permis de mettre en avant les systèmes d'origines d'accès des données ainsi que le principe des middlwares, étant des fonctions effectuant les tâches suivantes :

- Exécuter tout type de code.
- Apporter des modifications aux objets de demande et de réponse.
- Terminer le cycle de demande-réponse.
- Appeler la fonction middleware suivant dans la pile.

Dans d'autres mots, exécuter des fonctions coté serveur à la suite selon une pile.

Figure 4 : Code de la mise en place du serveur local

Par la même occasion, j'ai crée une simple page web avec un rendu précédemment évoqué.
Ainsi qu'un premier menu, pour le moment sans interaction mais dans une optique de penser à la suite du projet, nécessitant l'utilisation de fichiers.

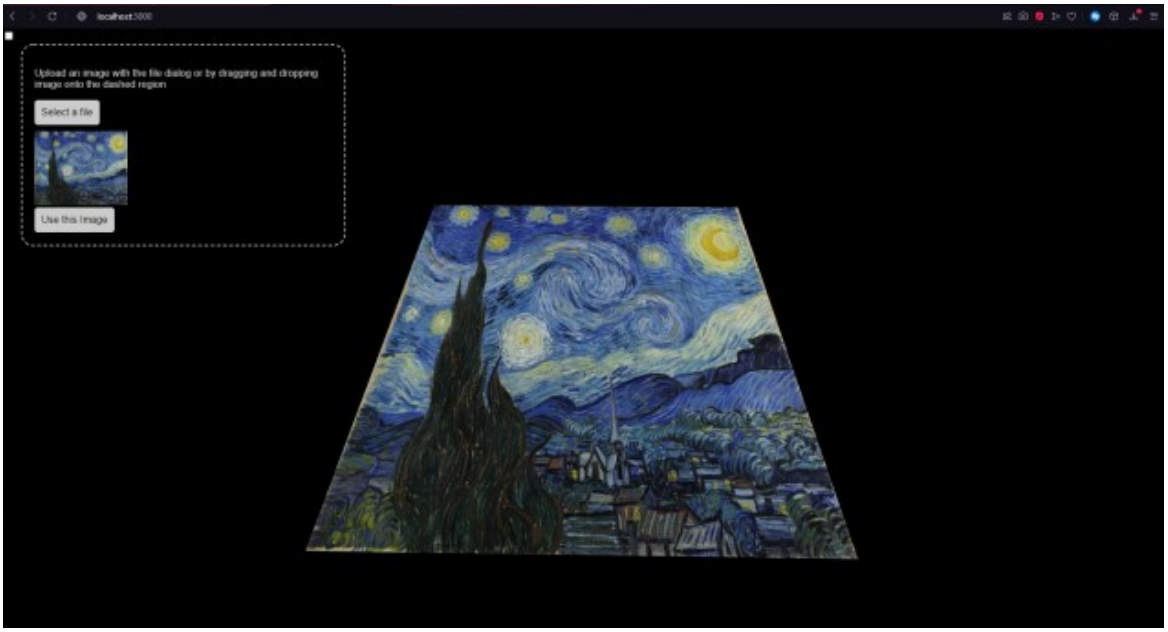

Figure 5 : Aperçu première page web

c) Drag&Drop

Une des étapes importantes dans le travail à effectuer lors de ce projet était un système pour glisser et déposer un fichier.
En effet toute page internet ayant pour fonctionnalité un quelconque traitement de donnée se doit de pouvoir utiliser un système *drag&drop.*

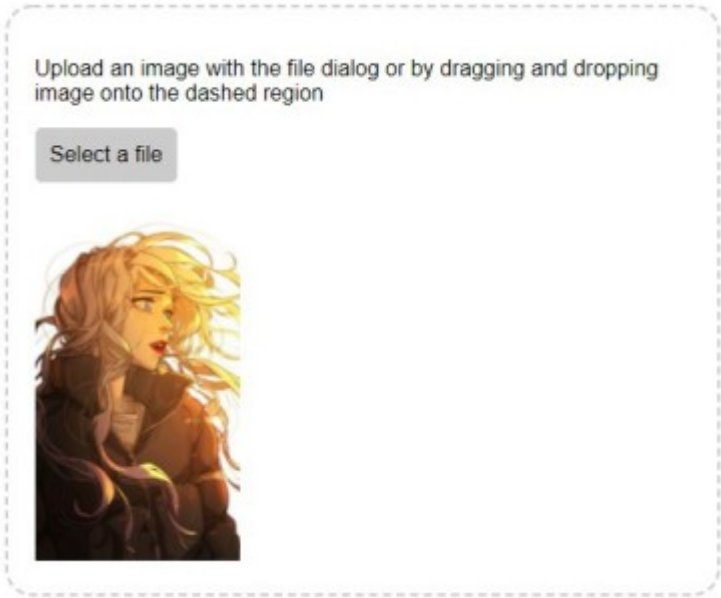

Figure 6 : Aperçu du menu de gestion de fichiers

Pour cela j'ai donc utilisé le menu précédemment crée dans lequel j'ai implémenté un bouton pour aller chercher un fichier depuis le gestionnaire de fichiers, de façon classique. Puis surtout dans lequel j'ai implémenté notre système de drag&drop comme indiqué sur la figure 6.

```
// preventDefault les événements liés au drag & drop
;['dragenter', 'dragover', 'dragleave', 'drop'].forEach(eventName => {
  dropArea.addEventListener(eventName, preventDefaults, false);
  document.body.addEventListener(eventName, preventDefaults, false);
})

// gestion affichage zone de drop
;['dragleave', 'drop'].forEach(eventName => {
  dropArea.addEventListener(eventName, hidden);
})
document.body.addEventListener('dragenter', unhidden);
dropArea.addEventListener('dragenter', unhidden);
```

Figure 7 : Code du drag&drop

En réalité cette fonctionnalité de glisser déposer n'est qu'un jeu avec différents évènements JavaScript. Ici, comme présenté sur la figure 7, il est question de traiter :

- dragenter ; le fait d'entrer avec un fichier dans une zone
- dragover ; le fait de survoler avec un fichier une zone
- dragleave; le fait de quitter le survol de cette zone
- drop; le fait de déposer un fichier sur une zone

C'est le fait de supprimer le comportement par défaut de ces événements et de le remplacer par notre fonctionnalité pour afficher ou non une zone de drag&drop, pour derrière en tirer le fichier déposé, qui crée notre fonctionnalité.

d) Décompression d'une archive

La décompression, ou dézippage d'une archive fut la première grosse fonctionnalité à traiter dans le projet, car s'il on le résume en deux étapes, le projet consiste à :

récupérer les données → afficher les données en 3D

Cela explique donc pourquoi cette étape est super importante.
Néanmoins c'est aussi cette étape qui m'a demandé le plus de temps et de problèmes.

J'ai beaucoup essayer d'utiliser des bibliothèques JavaScript en frontend pour pouvoir simplement extraire les fichiers de l'archive et les utiliser. Un peu comme ce que j'ai pu voir à l'adresse suivante : HTML5 Rar and Zip Archive Viewer | TechSlides

Néanmoins, la page ne faisait pas tout a fait ce que je souhaitais et le code était plutôt complexe. Et ceci pour l'ensemble des bibliothèques que j'ai pu essayer en frontend.
Je me suis dirigé vers des bibliothèques fonctionnant avec Express.js du coté serveur, qui elles fonctionnaient parfaitement et en quelques lignes.

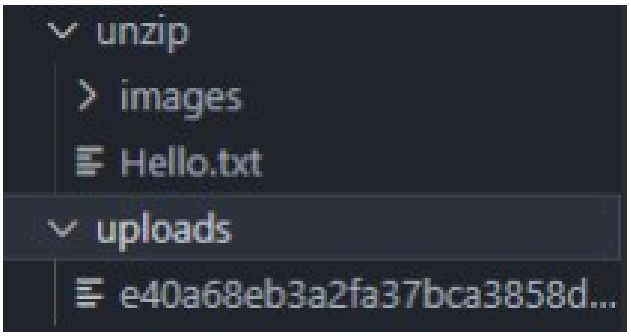

Figure 8 : Aperçu des fichiers décompressés en backend

Bien que le fait de dézipper du coté serveur fut plus simple, cela entraînera un soucis, la transmission des données de la page au serveur, puis du serveur à la page.

e) Transmission et utilisation des données

Cette partie du projet fut le plus gros soucis que j'ai pu rencontrer car transmettre des informations d'une page à un serveur n'utilise pas de bibliothèque, donc n'est pas énormément documenté.

J'ai du faire appel à une aide extérieure, Julien Wyllerman, qui lui s'y connaît très bien et qui m'a expliqué comment gérer le problème.

Figure 9 : Fonction fetch en frontend

Pour se faire il a fallut utiliser une fonction *fetch* coté frontend pour envoyer les données au serveur et une fonction application.post du coté backend pour récupérer, traiter et ensuite renvoyer les données à la page avec le bon url.

La figure 8 nous montre donc cette fonction *fetch*, qui va permettre d'envoyer les données sous la structure *formData*. On remarque ensuite le format d'un middleware avec les *.then((req,res))* permettant le traitement de la réponse ultérieure du serveur.

```
// On traite les requetes POST vers l'URL upload_files,
app.post("/upload_files",
  upload.array("files"), // copie des fichiers par multer dans son répertoire de travail
  uploadFiles);

  function uploadFiles(req, res)
  {
    // fonction de décompression
    unzipRec(req.files, res);
  }

  function unzipRec(files, res, rep=[])
  {
    if(0 == files.length)
      // envoie de la liste d'url en front
      res.json({success:true, files:rep});
    else
    {
      let file = files.shift();
      // dézippage
      decompress(file.path, "public/files").then(unzippedFiles =>
      {
        unzippedFiles.forEach(unzippedFile =>
        {
          rep.push('files/'+unzippedFile.path); // ajout des url à la liste d'url
        });
        unzipRec(files, res, rep); // appel récursif
      });
    }
  }
```

Figure 10 : Fonction post en backend

Du coté du serveur, la fonction *post* va s'occuper de récupérer le *formData* envoyé selon l'url indiqué lors de l'envoi. Elle va simplement dézipper les données dans les dossiers du serveur.

Et enfin renvoyer une réponse à la page, étant une structure *json* contenant l'ensemble des chemins vers les fichiers décompressés.

Hors cette partie ne représente que la transmission des données de la page au serveur et inversement, maintenant il faut pouvoir utiliser les fichiers dézippés en frontend.

```
function loadFiles(urls)
{

  const container = document.getElementById('container');
  // récupére puis traite les url en fonction des types de fichiers
  urls.forEach(url =>
  {
    switch(url.split('.').at(-1))
    {

      case 'jpg': // si image, modifie l'image du menu et du plan
          let img = document.getElementById('img');
          img.src = url;
          break;

      case 'xml':
          let data = JSON.stringify({
              "urls": url
          });
          // envoie de l'url de l'xml pour lecture du fichier coté serveur
          fetch("./upload_xml", {
              method:'POST',
              body: data,
              headers: {
                  'Accept': 'application/json',
                  'Content-Type': 'application/json'
              }
          })
          .then((res) =>
          {
```

Figure 11 : Code utilisation données décompressées

Pour se faire, nous avons récupérer les chemins vers les fichiers, et pour chaque type de fichiers (jpg / xml / dat) nous effectuons une fonction différente.

- Le cas le plus simple est celui de l'image, car il suffit simplement de modifier la source de la balise image.
- Concernant les *xml*, il a fallut, de la même façon que pour le dézippage, avec les méthodes *fetch* et *post*, ainsi que les middlewares, envoyer une requête au serveur pour cette fois-ci lire le fichier et renvoyer son contenu à la page et l'afficher.
- Enfin pour les fichiers *dat*, cela se présente exactement comme les fichiers *xml*, au détail prêt que son affichage n'est pas utile pour le moment.

f) Heightmap

Jusqu'à présent, les fonctionnalités liées aux fichiers ont été traité. Il fut désormais possible de se pencher vers la deuxième partie du projet, c'est à dire afficher les données en 3D.

Je m'étais premièrement penché vers le fait de créer une heightmap avec des cubes, sur lesquels on aurait modifié leurs tailles, leurs hauteurs, positions ainsi que leurs couleurs.
J'ai donc dans un premier temps tester différents rendu, pour m'habituer avec ces cubes.

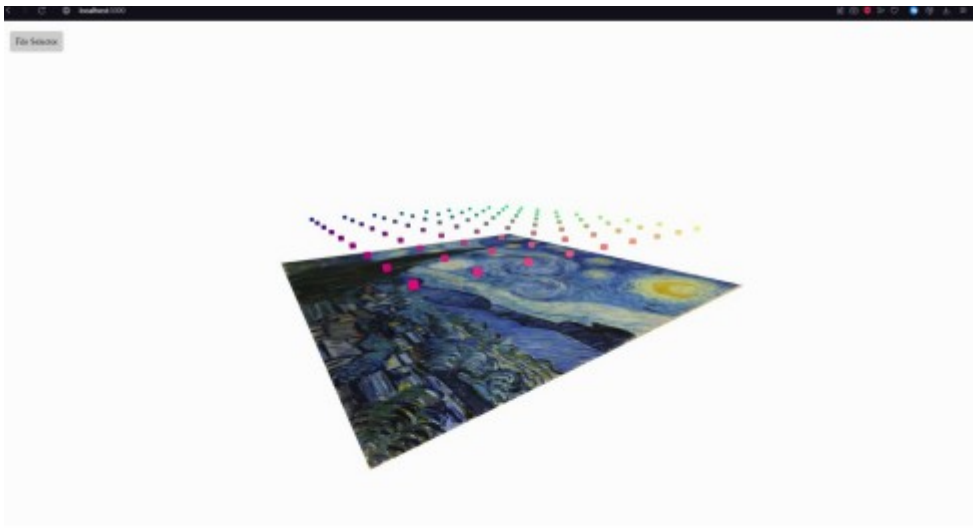

Figure 12 : Premier essai rendu avec cubes

J'ai par la suite trouvé un autre moyen, beaucoup plus convaincant de modéliser une heightmap.
Cette heightmap est en réalité un plan, duquel les points on été étirés selon le canevas en noir et blanc en haut a droite de l'image (cf. figure13).

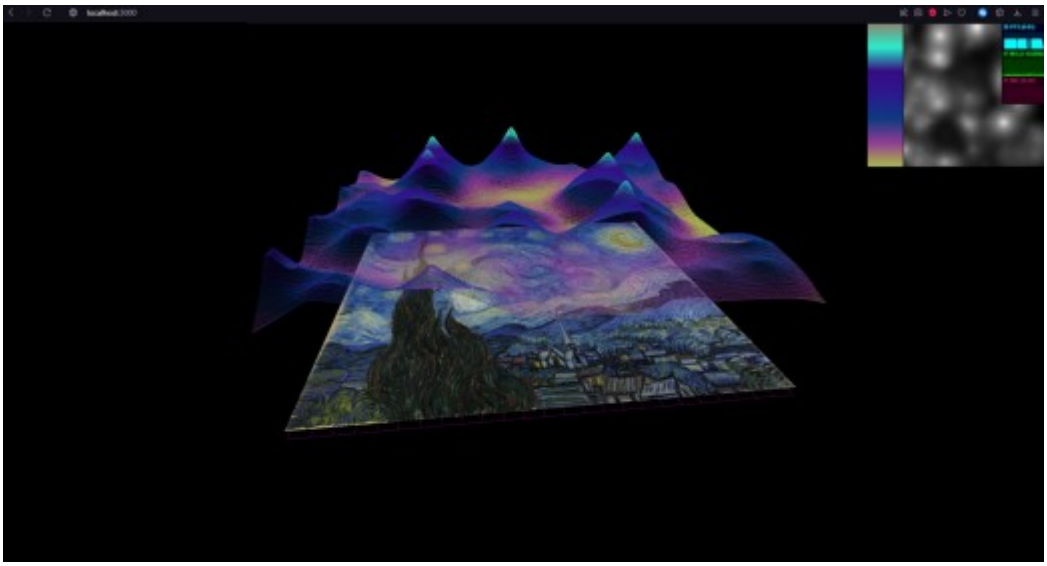

Figure 13 : Première heightmap

De plus aucune archive et globalement rien des travaux concernant les fichiers n'a été implémenté pour simplifier la conception de ce rendu, donc la mire de couleur et le canevas heightmap sont totalement aléatoires et peuvent changer en cliquant dessus.

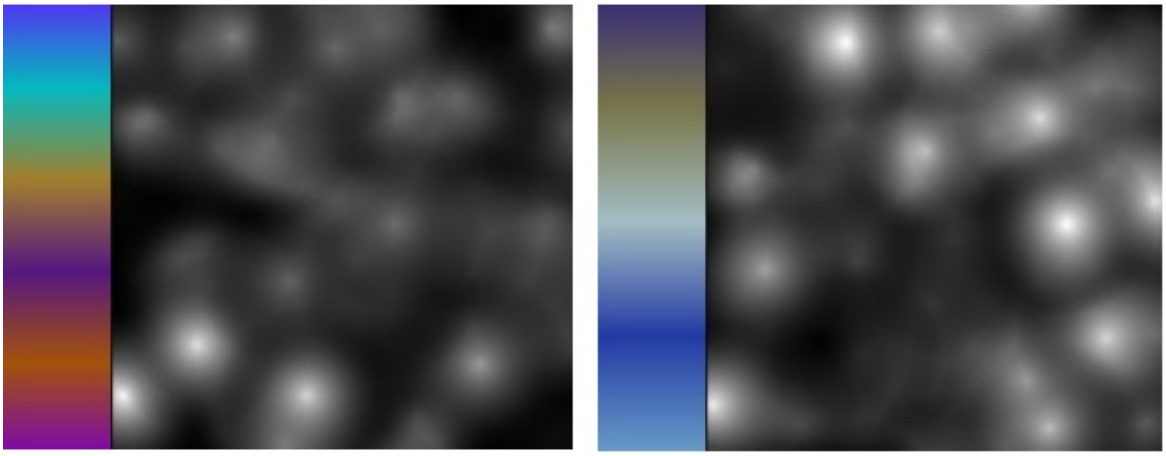

Figure 14 : différentes mires et canevas aléatoire

g) Performances

A l'issue des travaux concernant les heightmap, il a fallut se pencher sur la question de la performance de l'application.

Pour se faire nous avons implémenté une petite fonctionnalité avec Three.js permettant d'afficher des panneaux concernant les statistiques du rendu sur la page.

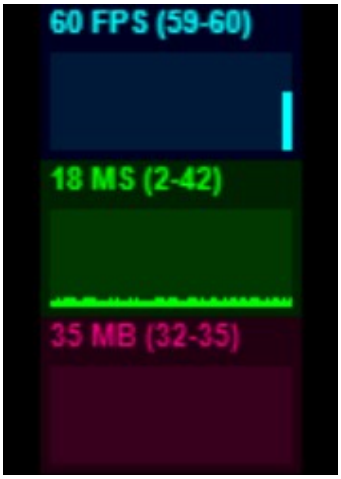

Figure 15 : Affichage stats

Cette question de performance avait donc été premièrement évoqué lors de la représentation des cubes.
Nous avons effectué des tests en augmentant considérablement le nombres de cubes sur le rendu et fusionnant ou non chaque objet du rendu.

Bien que le résulats n'étaient pas très concluant et que nous avons trouvé entre temps trouvé un autre moyen de représenter notre heightmap, ce fut une première approche concernant cette notion de performance, importante à traiter, ou du moins à ne pas oublier.

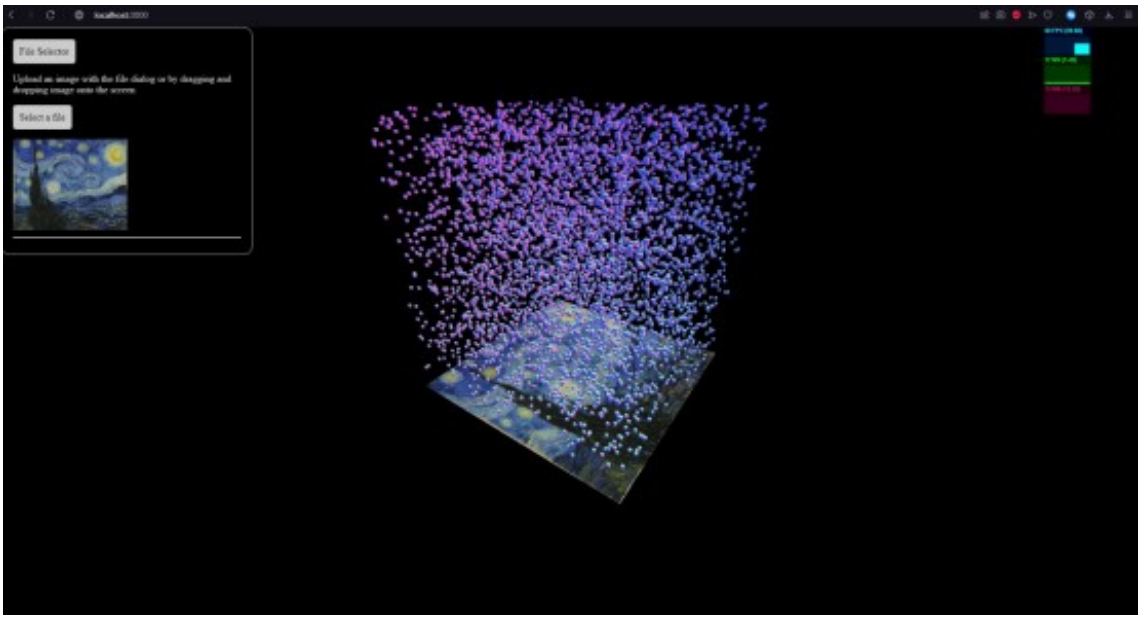

Figure 16 : Test d'optimisation des performances

Pour continuer sur cette notion de performances, il a été remarqué par la suite que la page avait un certain mal à charger et avait des problèmes de latence.

l'utilisation des statistiques de la page nous a permis de voir par la suite des baisses de performances, une chute des fps pour chaque archive uploadé.

De cette façon nous avons pu remarquer des duplication de balises, mais surtout de rendu, ce qui forcément pèse à chaque fois plus lourd sur la page.

Actuellement il existe encore quelques problèmes de latence malgré ces premiers correctifs, l'optimisation est donc encore un sujet à aborder.

h) Travaux futurs

En effet, mon stage se déroulant jusque fin juillet, le projet est encore en cours.

Mais il est déjà possible de prévoir les prochains travaux à effectuer et d'avoir un aperçu du travail à venir :

- Utilisation des données pour afficher une heightmap à chaque fichiers, c'est le travail que je fais en ce moment même (semaine de rendu du rapport) et il déjà possible de voir une représentation avec des mires de couleurs prédéfinis et des données, bien que simplifiés voulues et non plus aléatoires comme auparavant

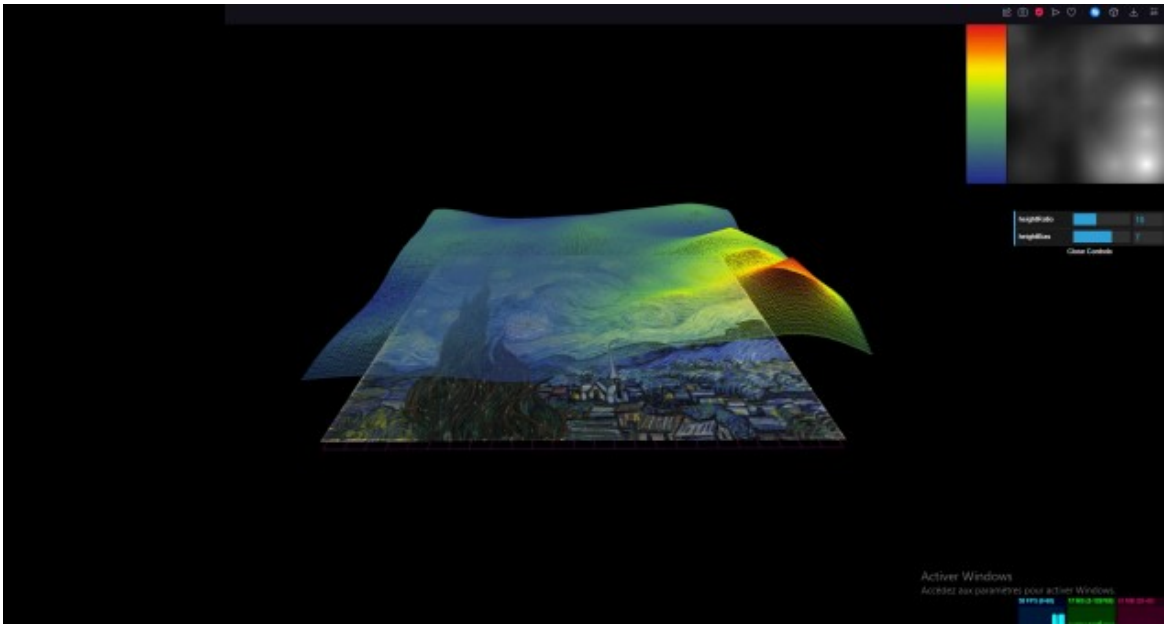

Figure 17 : Rendu d'une heightmap avec .dat allégé

- Modification du rendu en temps réel, pareil ici, ce travail à déjà été commencé. En effet il serait bien de pouvoir modifier l'écart du maillage ainsi que le nombre de point sur le rendu de la heightmap en temps réel, pour le moment seulement l'intensité des valeurs et la hauteur de la map ont été prise en compte avec un petit panneau à cet effet.

Figure 18 : Panneau de modification du rendu

- Comme évoqué précédemment, il va falloir perfectionner les performances de la page, qui pour le moment ne sont pas optimales.

- Représenter le rendu d'une heightmap pour toutes les fréquences des données et non pas seulement de la première

- Créer l'ensemble des fonctionnalités que l'on peut retrouver sur le Viewer Android, telle que pouvoir comparer deux heightmap, avoir accès aux données à la demande ou gérer l'affichage des rendus, transparence par exemple etc.

i) Github

Pour finaliser sur ma contribution, je tiens à informer que plusieurs page Github ont été crée afin de suivre mon avancée sur le projet et de retrouver tous mes travaux, essais et sources.

Voici son lien : GitHub - Luxondes/ViewerWeb

Enfin, bien que mentionné sur les pages Github, voici le lien permettant de suivre le rendu en temps réel du projet : https://viewer-luxondes.com

4. Conclusion

Tout d'abord, je tiens à remercier Luxondes ainsi que mes tuteur pour leur accueil lors de ces mois du stage. Ce projet m'as permis de faire un premier pas vers un domaine qui me tiens particulièrement à cœur, celui de la modélisation 3D et ce qui l'entoure.

Ce rapport a retracé mon travail effectué sur le projet, à travers tous les cheminements de pensées, les problématiques, et solutions que l'on a pu avoir.

J'ai eu une grande satisfaction de mon travail car je suis arrivé dans l'entreprise avec peu de bagages en fin de compte, et j'en repart avec, de mon point de vue, énormément d'expérience et beaucoup d'apprentissage.

Cela m'a conforté dans l'idée de vouloir continuer mes études dans ce domaine, ou du moins d'y consacrer de mon investissement et de mon temps.

Le projet n'est bien sur pas terminé, il me reste 1 mois de travail. D'autant plus que mon tuteur compte reprendre et continuer ce que j'ai pu commencer.
J'espère avoir un minimum répondu aux attentes que l'on pouvait avoir de moi concernant ce projet, et qu'il puisse mener à bien dans le futur de Luxondes.

5. Bibliographie

Autoformation

- https://openclassrooms.com/fr/courses/6175841-apprenez-a-programmer-avec-javascript
- https://www.youtube.com/watch?v=9OJLxDxyNg4
- https://www.youtube.com/watch?v=6q-zt0aQ74U
- https://github.com/sessamekesh/IndigoCS-webgl-tutorials
- https://www.youtube.com/playlist?list=PL2935W76vRNHFpPUuqmLoGCzwx_8eq5yK
- https://glmatrix.net/docs/
- https://openclassrooms.com/fr/courses/5543061-ecrivez-du-javascript-pour-le-web
- https://openclassrooms.com/fr/courses/6390246-passez-au-full-stack-avec-node-js-express-et-mongodb

Three.js

- https://github.com/mrdoob/three.js
- https://threejs.org/docs/
- https://jsfiddle.net/7u84j6kp/

Drag&Drop

- https://www.youtube.com/watch?v=aa18QsK5QZI&t=186s
- https://www.youtube.com/watch?v=6pc9G2ahUuc

Dézippage

- https://stackoverflow.com/questions/3950029/handle-refresh-page-event-with-javascript
- https://stackoverflow.com/questions/18052762/remove-directory-which-is-not-empty
- https://stackoverflow.com/questions/65989416/node-express-req-body-returns-empty-obj
- https://www.youtube.com/watch?v=0gON4MUdJE8

Autres

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://mathjs.org/docs/datatypes/matrices.html
- https://nodejs.org/api/readline.html#readline_example_read_file_stream_line_by_line