Université de Bordeaux,
Campus de Talence,
351 cours de la Libération,
33405 Talence

Thermo Fisher
Scientific,
39 rue d'Armagnac,
33000 Bordeaux

IIIF image viewer using WebGPU

Benoît Boidin, Aurélie Casanova, Sébastien Lebreton

Projet de fin d'étude Master Informatique

janvier - mars 2024

Client : Arnaud-Marie Gallardo (Thermo Fisher)

Enseignants responsables de l'UE : Pascal Desbarats, Pierre Bénard

Résumé

Ce projet a pour objectif de développer un viewer d'images hautement résolutives basé sur le framework IIIF [2] [4] [5] et la technologie WebGPU [3][7][11][12] .

La solution proposée s'intègre dans la plateforme Athena[6] de Thermo Fisher Scientific*[1][9]*, offrant ainsi une gestion centralisée des données scientifiques et une interface web de visualisation.

Le résultat attendu est une bibliothèque distribuable via npmjs.com, comprenant un WebComponent[10] simple à intégrer, permettant le chargement, la navigation, le zoom, et l'application de palettes de couleur sur des images 2D, tout en anticipant l'évolution vers WebGPU.

Abstract

This project aims to develop a high-resolution image viewer based on the IIIF framework and WebGPU technology.

The proposed solution integrates with Thermo Fisher Scientific's Athena platform, providing centralized management of scientific data and a web-based visualization interface.

The expected outcome is a distributable library via npmjs.com, including a WebComponent easy to integrate, allowing loading, navigation, zooming, and the application of color palettes to 2D images, while anticipating the transition to WebGPU.

Sommaire

Index des figures

Introduction

L'évolution croissante de la quantité et résolution des données issues d'acquisitions par microscopie électronique*[1] pousse les chercheurs et les entreprises, comme Thermo Fisher Scientific*, à revoir leurs méthodes de stockage et de visualisation de ces données.

Athena* est l'une des solutions apportées par Thermo Fisher Scientific pour répondre à ce besoin. Elle apporte une gestion centralisée des données scientifiques, un système de suivi des workflows* de traitements de ces données et pour finir une interface web de visualisation des résultats.
Afin de proposer une interface standardisée permettant de visualiser et travailler avec des images 2D très haute-résolution, la plateforme Athena se repose sur le framework* IIIF* et principalement l'API* standardisé Image de IIIF.

Parmi les outils permettant de visualiser des images via l'interface IIIF, le plus connu reste OpenSeaDragon [8]. Bien que très performant, la couche de rendu de ce viewer n'est pas basée sur une technologie type WebGL*, ce qui empêche une manipulation fine du rendu, comme par exemple l'application d'une palette de couleur sur les images en niveaux de gris*.

Certains projets proposent une version modifiée d'OpenSeaDragon permettant d'utiliser WebGL dans la couche de rendu mais aucun support officiel n'est actuellement proposé. De plus avec l'arrivée de WebGPU*, considéré comme le successeur de WebGL, il semble intéressant d'en profiter pour explorer une nouvelle piste pour mettre en place le viewer de demain.

[1] les mots annotés d'une astérisque seront définis dans le lexique situé en annexe du rapport

1. État de l'art

1.1. Analyse des Solutions Actuelles

1.1.1. Utilisation d'OpenSeaDragon avec IIIF

L'intégration d'OpenSeaDragon avec le framework IIIF (International Image Interoperability Framework) est une approche courante pour la visualisation d'images numériques. OpenSeaDragon, un viewer d'images open-source, offre une solution robuste pour la présentation et l'exploration d'images 2D à haute résolution via l'API standardisée Image de IIIF.

OpenSeaDragon prend en charge le chargement d'images directement depuis un serveur IIIF en utilisant l'URL de l'image ou du manifeste IIIF. Cette intégration permet une navigation fluide à travers des images très haute résolution, offrant une expérience utilisateur interactive et immersive. Les fonctionnalités de zoom, de panoramique et de navigation sont optimisées pour exploiter les capacités de IIIF, offrant ainsi une visualisation détaillée et précise.

Cependant, l'utilisation d'OpenSeaDragon peut présenter des limitations, notamment en ce qui concerne le rendu basé sur WebGL. Dans le contexte de ce projet, la nécessité d'une manipulation fine du rendu, comme l'application de palettes de couleur sur des images en niveaux de gris, a motivé l'exploration de nouvelles approches, notamment l'utilisation de la technologie WebGPU, pour relever ces défis spécifiques et améliorer davantage la qualité de la visualisation.

1.1.2. Limitations de la couche de rendu non basée sur WebGL

La principale limitation associée à l'utilisation d'une couche de rendu non basée sur WebGL, comme c'est le cas avec OpenSeaDragon, réside dans les capacités graphiques réduites en termes de manipulation fine du rendu. Contrairement à WebGL, qui exploite pleinement la puissance de traitement parallèle du GPU (unité de traitement graphique), une couche de rendu non basée sur WebGL peut présenter des défis spécifiques :

1. Performances limitées : La manipulation d'images très haute résolution, souvent rencontrée dans des applications scientifiques telles que la microscopie électronique*, peut être moins efficace en termes de performances. Les opérations de zoom, de panoramique et d'affichage peuvent être moins réactives.

2. Options de Rendu Limitées : Les fonctionnalités graphiques avancées, telles que l'application de palettes de couleur sur des images en niveaux de gris, peuvent être

restreintes. Cela peut entraîner une limitation dans la représentation visuelle des données scientifiques.

3. Dépendance aux Performances du CPU* : L'utilisation d'une couche de rendu basée sur le CPU plutôt que sur le GPU* peut entraîner une dépendance aux performances du processeur principal, ce qui peut affecter la fluidité de l'expérience utilisateur.

4. Difficulté à Gérer des Images Très Haute Résolution : Les limitations liées à la gestion des images très haute résolution peuvent se traduire par des temps de chargement plus longs et des contraintes sur la manipulation fluide de ces images.

En somme, l'absence de WebGL peut restreindre la capacité d'effectuer des opérations graphiques avancées et d'optimiser les performances, surtout dans le contexte de la visualisation de données scientifiques exigeantes. Cette limitation motive souvent l'exploration de technologies telles que WebGPU pour améliorer ces aspects spécifiques de la visualisation.

1.2. Introduction à WebGPU

1.2.1. Caractéristiques clés de WebGPU

WebGPU, en tant que technologie émergente, présente des caractéristiques clés qui en font une solution prometteuse pour le rendu graphique avancé dans les navigateurs web. Tout d'abord, WebGPU est conçu pour tirer pleinement parti des performances parallèles du GPU, permettant ainsi une accélération significative du rendu graphique. Cette approche offre une réactivité accrue lors de l'affichage d'images très haute résolution, particulièrement pertinent dans des contextes scientifiques tels que la microscopie électronique.

Une autre caractéristique essentielle de WebGPU est son support natif pour les fonctionnalités avancées de shaders graphiques*. Cela permet une personnalisation fine du rendu visuel, répondant aux besoins spécifiques des applications scientifiques, telles que l'application de palettes de couleur sur des images en niveaux de gris. La flexibilité offerte par les shaders dans WebGPU ouvre la voie à des représentations visuelles riches et précises des données.

De plus, WebGPU est conçu pour être plus proche du matériel graphique, offrant ainsi une abstraction plus bas niveau par rapport à WebGL. Cette proximité avec le GPU se traduit par des performances graphiques optimisées et une gestion plus efficace des ressources, améliorant ainsi l'expérience utilisateur, en particulier dans des scénarios exigeants comme la visualisation de données scientifiques.

En résumé, les caractéristiques clés de WebGPU résident dans sa capacité à exploiter efficacement les performances parallèles du GPU, à prendre en charge des fonctionnalités avancées de shaders et à offrir une abstraction plus bas niveau, contribuant ainsi à une visualisation graphique optimale dans un contexte web.

1.2.2. Pertinence par rapport à WebGL pour les images 2D

La pertinence de WebGPU par rapport à WebGL dans le contexte des images 2D réside principalement dans ses capacités améliorées de rendu graphique et de manipulation fine du contenu visuel. Contrairement à WebGL, WebGPU offre une approche plus moderne et puissante, permettant une exploitation plus efficace du GPU pour le rendu graphique. Cette caractéristique est particulièrement cruciale pour les images 2D très haute résolution, où les performances parallèles du GPU peuvent être pleinement exploitées pour assurer une expérience utilisateur fluide et réactive.

Une autre dimension de la pertinence de WebGPU pour les images 2D réside dans sa prise en charge native des fonctionnalités avancées de shaders graphiques. Cette capacité permet une personnalisation fine du rendu visuel, offrant ainsi des options étendues pour l'amélioration de l'esthétique et la représentation précise des détails dans les images 2D. Dans le domaine de la visualisation scientifique, où la précision visuelle est cruciale, cette fonctionnalité représente un avantage significatif par rapport à WebGL.

De plus, WebGPU se distingue par une architecture plus moderne et une abstraction plus bas niveau par rapport à WebGL, favorisant une gestion plus efficace des ressources et des performances optimisées. Ces caractéristiques font de WebGPU une solution pertinente et prometteuse pour le rendu avancé d'images 2D, notamment dans des domaines exigeants tels que la recherche en microscopie électronique.

1.3. Objectifs du Projet

Notre projet vise à développer un prototype fonctionnel de viewer d'images très haute résolution utilisant la technologie WebGPU et le framework IIIF.

Les objectifs spécifiques de ce projet incluent la conception et l'implémentation d'une solution innovante pour répondre aux besoins croissants de visualisation d'images scientifiques. Nous nous sommes fixés pour objectif de fournir une expérience utilisateur optimale en permettant le chargement fluide et la manipulation intuitive d'images de qualité scientifique.

Principales fonctionnalités souhaitées du viewer IIIF* :

- Chargement et affichage d'images complètes via IIIF.
- Chargement et affichage par tuiles* de l'image avec IIIF.
- Chargement et affichage par niveaux de résolution de l'image avec IIIF.
- Fonctionnalités de navigation : déplacement, zoom et dézoom.
- Activation et désactivation d'un shader de palette de couleur sur l'image.
- Affichage d'une mini-carte de la position de la caméra actuelle sur l'image.

1.4. Travaux Connexes

La revue rapide des projets similaires révèle que, bien que plusieurs solutions de visualisation d'images 2D utilisant IIIF soient disponibles, la recherche d'une intégration avec WebGPU demeure limitée. Les projets existants, tels qu'OpenSeaDragon, ont démontré leur efficacité dans la visualisation d'images IIIF, mais la plupart ne bénéficient pas encore officiellement du support de WebGL, entravant ainsi la manipulation fine du rendu.

Certains projets ont entrepris des adaptations d'OpenSeaDragon pour intégrer WebGL dans la couche de rendu, mais ces modifications ne sont souvent pas officiellement supportées. L'absence de solutions établies utilisant WebGPU dans le contexte de IIIF suggère une opportunité pour le projet actuel d'explorer une voie novatrice en adoptant cette technologie émergente, considérée comme le successeur potentiel de WebGL.

La littérature et les forums spécialisés reflètent un intérêt croissant pour l'intégration de WebGPU dans des projets de visualisation graphique. Cependant, il est notable que les projets spécifiques au contexte de IIIF et de la microscopie électronique restent à être pleinement développés. Ainsi, le projet actuel pourrait se positionner en tant que pionnier dans l'exploration de WebGPU au sein du framework IIIF pour la visualisation optimale d'images 2D très haute résolution.

1.5. Conclusion

Les principaux enseignements tirés de cette revue mettent en lumière la nécessité d'évoluer vers des technologies plus avancées pour répondre aux exigences de la visualisation d'images 2D très haute résolution dans le contexte de IIIF. La popularité d'OpenSeaDragon, bien qu'efficace, souligne la limitation de son rendu non basé sur WebGL, entravant ainsi des fonctionnalités avancées telles que l'application de palettes de couleur sur des images en niveaux de gris.

L'exploration de WebGPU émerge comme une réponse pertinente à ces défis, car elle offre une approche moderne, optimisée pour le GPU, ainsi qu'une prise en charge native des fonctionnalités avancées de shaders graphiques. Ces caractéristiques répondent directement aux besoins spécifiques du projet, notamment la manipulation fine du rendu et la création d'une interface optimale pour la visualisation d'images scientifiques très haute résolution.

Le choix de WebGPU est justifié par son potentiel à surmonter les limitations identifiées dans les solutions existantes. En adoptant cette technologie émergente, le projet anticipe également l'avenir, se positionnant comme un précurseur dans l'exploration de WebGPU au sein du framework IIIF. Cette décision est motivée par le besoin de rester à la pointe des avancées technologiques, offrant ainsi une solution de visualisation novatrice et performante pour les applications scientifiques exigeantes. En résumé, les choix technologiques s'alignent sur la recherche d'une solution qui allie efficacité immédiate et préparation pour les évolutions futures de la technologie web.

2. Le développement

2.1. Objectifs du projet

L'objectif de ce projet est de mettre en place un prototype de viewer d'images très haute résolution utilisant la technologie WebGPU et le framework IIIF.

2.2. Les étapes du projet

2.2.1. Prise en main de WebGPU et IIIF

Pour commencer, nous avons suivi le tutoriel officiel de WebGPU, afin de connaître les bases de son fonctionnement. Nous avons appris l'utilisation des balises HTML canvas et celle du pipeline graphique utilisé en Web.

Les prérequis pour l'utilisation de cet outil sont le support de WebGPU par le navigateur, et le succès des requêtes de "adapter" et de "device".
Outre les éléments spécifiques à WebGPU, nous retrouvons dans cet outil des shaders classiques. Ils sont inclus dans le pipeline graphique.

Une fois le pipeline mis en place, il suffit de dessiner les graphiques en accédant aux variables du GPU (ce dernier ne peut pas directement accéder aux variables des autres langages) et en les transmettant au "buffer".

Comme ce tutoriel ne permettait d'afficher que des carrés de couleurs, nous avons dû nous renseigner afin de convertir des images en textures, utilisables par le GPU. Après plusieurs itérations, nous avons été capables d'afficher une image venant d'un dossier local.

Par la suite, nous avons exploré les capacités de IIIF pour les exploiter au mieux. Cette analyse a impliqué la décomposition de l'API en différents paramètres, et la sélection de ceux qui allaient être utiles pour le projet. Voici l'URL type pour la requête :

```url
/cantaloupe/iiif/3/{identifier}/{region}/{size}/{rotation}/{quality}.{format}
```

Figure 1 : requête de chargement de l'image via cantaloupe

Une attention particulière a été portée à l'ordre d'exécution de ces paramètres, afin de garantir le traitement adéquat des images.

Ce tableau permet de résumer le résultat de ces recherches (figure 2).

identifier	region	size	rotation	quality	format
image file path	`full`	`max`	`n`	`color`	`jpg`
	`square`	`^max`	`!n`	`gray`	`tif`
	`x,y,w,h`	`w,`		`bitonal`	`png`
	`pct:x,y,w,h`	`^w,`		`default`	`gif`
		`,h`			`jp2`
		`^,h`			`pdf`
		`pct:n`			`webp`
		`^pct:n`			
		`w,h`			
		`^w,h`			
		`!w,h`			
		`^!w,h`			

Figure 2 : Liste des paramètres des requêtes IIIF.

2.2.1.1. Cantaloupe et docker*

Parallèlement, nous avons mis en place un serveur Cantaloupe*, écrit en Java, qui permet d'utiliser cette API. Nous avons ainsi pu récupérer des images lourdes avec une simple page web (en utilisant pour l'instant le CPU).
Puis, nous avons modifié cette page web pour qu'elle récupère l'image du serveur et qu'elle l'affiche grâce au pipeline WebGPU, comme décrit plus haut.

Notre solution permettait à présent d'utiliser l'API et le GPU.

Afin de correspondre aux standards du marché et faire en sorte que la solution soit portable (nous travaillons tous avec des systèmes d'exploitation différents), nous avons utilisé Docker pour créer un conteneur et faire fonctionner le serveur.
L'exposition du port* 8182 (par défaut pour Cantaloupe) permet de communiquer avec le conteneur.

2.2.1.2. Affichage par niveaux de résolution

L'API de IIIF permet de demander une région spécifique de l'image, dans une définition donnée (voir tableau ci-dessus). Nous avons donc entrepris la création d'un système de tuiles, afin de pouvoir agrandir l'image et adapter la résolution dynamiquement.

Lors de l'affichage initial, l'image est divisée en 4 tuiles d'une résolution de 256 par 256.

Lorsque le client effectue un "zoom", on augmente la quantité de tuiles qui composent l'image en passant à 16, mais elles gardent la même définition, et ainsi de suite. On obtient donc progressivement une image de meilleure qualité, adaptée au niveau de zoom, sans avoir besoin de charger l'entièreté du fichier dès l'ouverture de la page web. Ceci permet une utilisation plus fluide et plus efficace du réseau.

2.2.2. Solution finale

Nous avons donc implémenté un viewer d'images (figure 3) permettant de se déplacer librement sur l'image et de zoomer à l'aide de la souris. Nous avons également intégré un panneau de contrôle pour l'utilisateur à l'aide de TweakPane[13]. Celui-ci permet d'appliquer n'importe quelle palette de couleur à l'image, de montrer les performance du viewer en temps réel grâce à un graphique montrant les FPS, de réinitialiser la vue du viewer et de pouvoir afficher le découpage en tuiles de l'image.

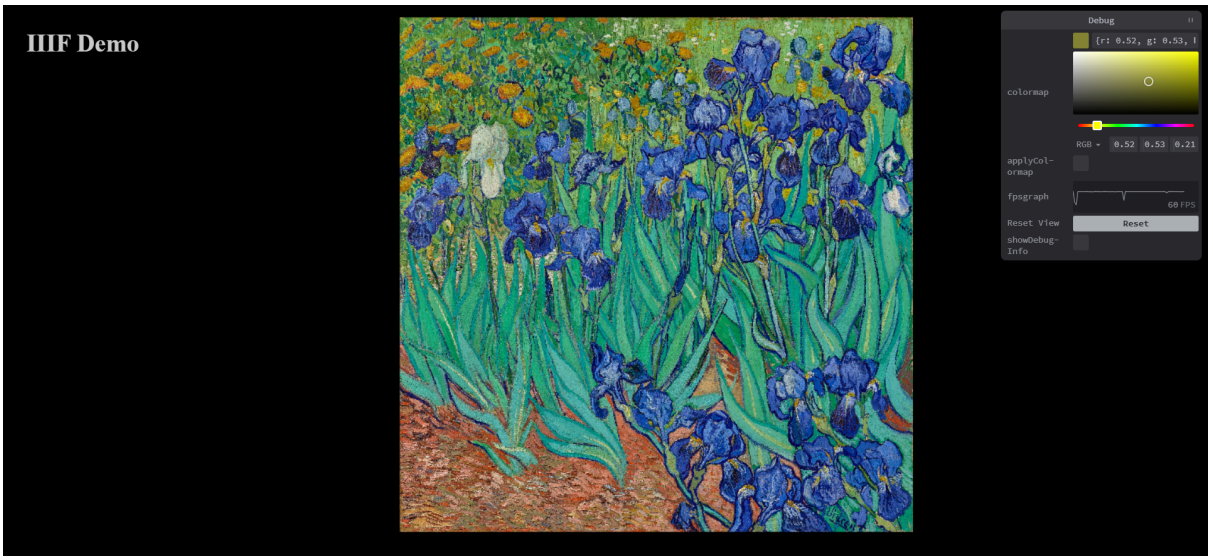


Figure 3 : Viewer d'images et panneau de contrôle

2.2.2.1 Affichage par tuiles

Dans l'optique de proposer au client la manipulation et la navigation dans l'image de manière fluide, nous avons mis en place un système de tuile, chacune de ces tuiles est chargée dynamiquement* et affiche une région de l'image.

Ce qui permet que lorsque l'on zoom sur l'image, le nombre de tuiles est augmenté et la région affichée est mise à jour par chaque tuile afin d'augmenter la résolution de l'image et permettre d'afficher beaucoup plus de détails.

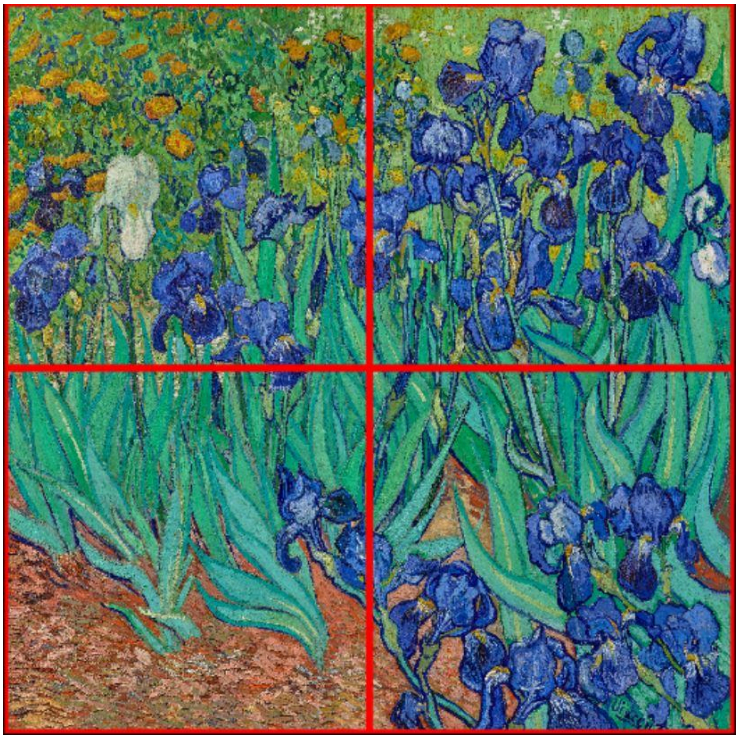


Figure 4 : Premier niveau de résolution du viewer

La figure 4 représente l'image de départ avec une résolution correcte mais peu de détails. L'image est découpée en quatre tuiles, chacune représentant un quart de l'image.

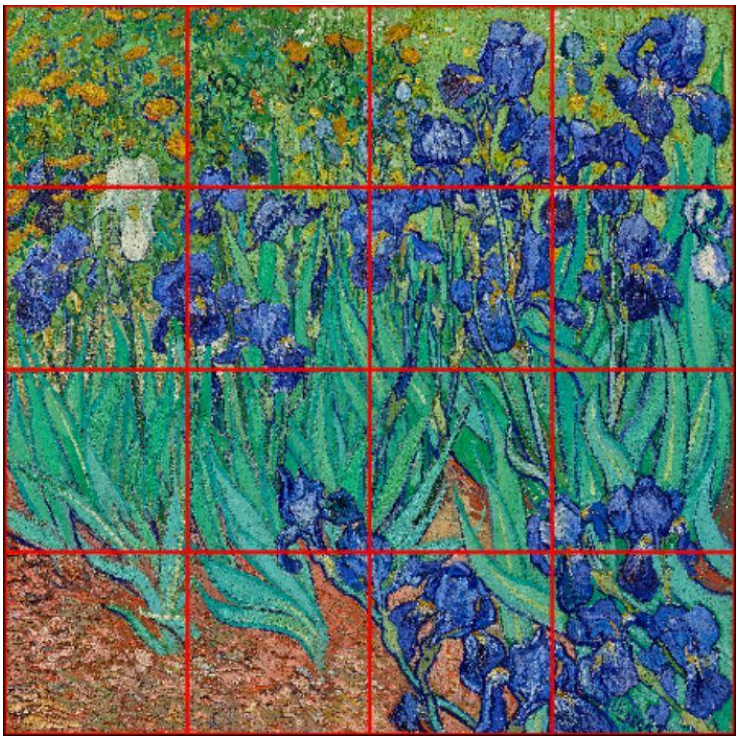
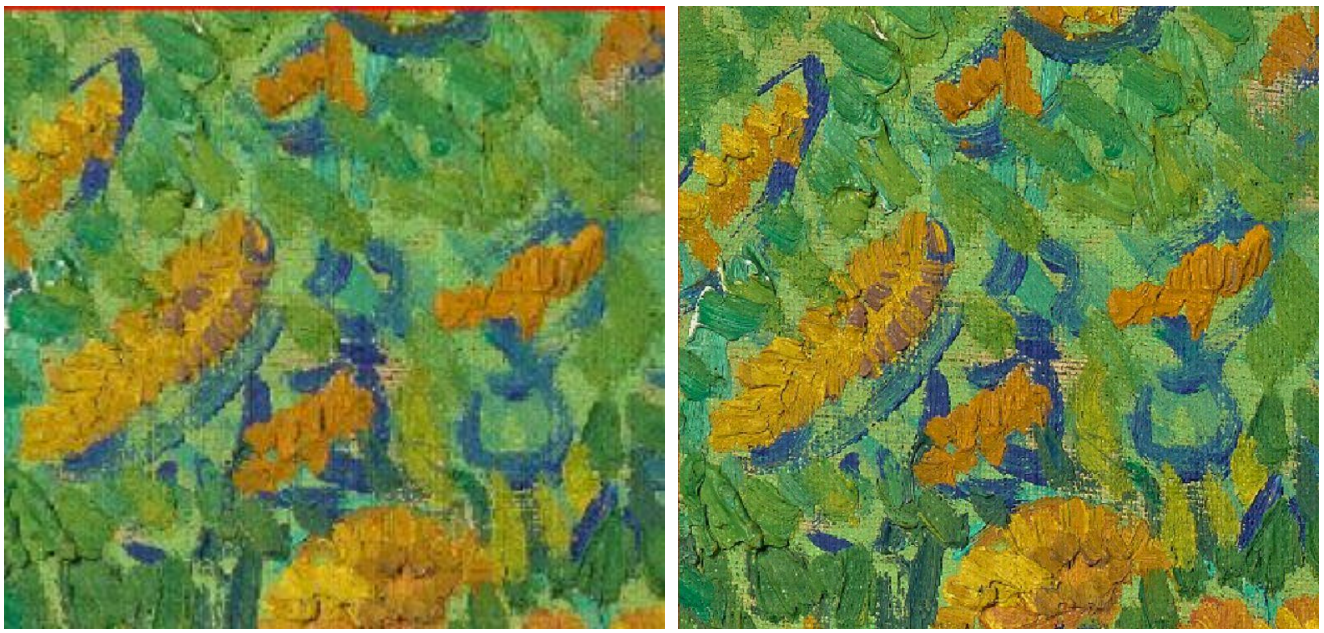


Figure 5 : Deuxième niveau de résolution du viewer

La figure 5 représente le découpage en tuiles de l'image pour le deuxième niveau de résolution, chaque tuile fait toujours la même taille que pour le premier niveau, soit 1024 pixels, mais affiche une plus petite partie de l'image, ici 1/16ème de l'image de départ, cela augmente donc la résolution.



Figures 6 et 7 : Différence de détail entre le premier niveau de résolution (à gauche) et le deuxième (à droite).

La différence entre les figures 6 et 7 est flagrante et démontre bien le niveau de détail qui est permis par le zoom lorsque l'image est découpée en tuiles.

2.2.2.2 Déplacement sur l'image

Le déplacement sur l'image se fait à la souris et au pavé tactile, il suffit juste de maintenir le clic et de se déplacer. On utilise la molette pour le zoom, ce qui met à jour la matrice de projection de la caméra pour créer cet effet de zoom, puis au bout d'un certain lorsque l'utilisateur zoom suffisamment, met à jour le nombre de tuiles et leur résolution.

```
let tilesPerRow = 2 * Math.pow(2,Math.floor(this.zoomLevel/3));
```

Figure 8 : Formule déterminant le nombre de tuiles par lignes et par colonnes.

Pour cela, on incrémente le niveau de zoom et par la suite, lors de la création des tuiles, on applique la formule de la figure 8 afin de savoir de combien de tuiles nous avons besoin par ligne et par colonne.

On a donc toujours une puissance de 2 comme nombre de tuiles et lorsque l'on passe au niveau supérieur chaque tuile est alors divisée en quatres sous-tuiles.
Le nombre de tuiles dans une ligne est le même que celui dans une colonne car nous travaillons uniquement avec des images carrées. Le client aura lui aussi des images carrées provenant d'acquisitions faites avec un microscope électronique.

La division par trois dans le calcul vient du fait que l'on met à jour la résolution tout les trois niveau de zoom (avec la molette par exemple, un niveau de zoom équivaut à un cran de rotation), cela permet de pouvoir continuer à zoomer sur une tuile sans forcément augmenter ou diminuer sa résolution.

Chaque résolution a donc deux états plus ou moins zoomés.

2.2.2.1 Appliquer une palette de couleur

Un des besoins fondamentaux du client était de pouvoir appliquer une palette de couleur à l'image, en effet, les images qui seront manipulées seront issues d'une acquisition au microscope électronique, et donc en niveau de gris.
Pouvoir appliquer une palette de couleur à l'image permettra de pouvoir en faire ressortir des éléments.

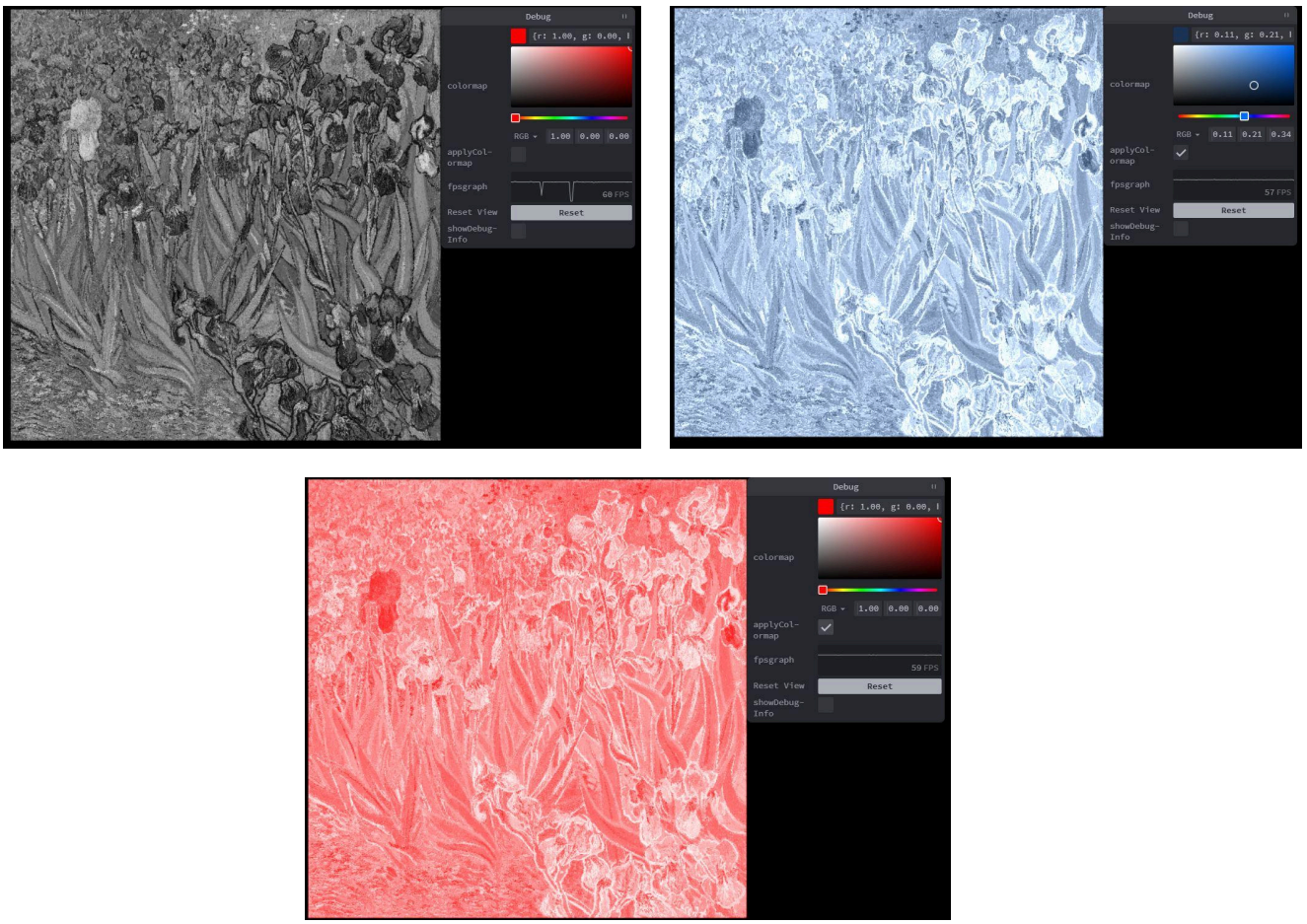


Figure 9, 10 et 11 : Image de base en niveau de gris (à gauche), l'image après l'application d'une palette de couleur bleutée (à droite) , et après l'application d'une palette de couleur rouge (en bas).

Le choix de la palette de couleur se fait grâce au sélecteur situé dans le panneau d'interactions, on peut la choisir directement à la souris en cliquant sur la couleur souhaitée ou remplir manuellement les valeurs RGB* voulues.

On peut ensuite cocher la case "applyColormap" pour appliquer la palette. Celle-ci sera appliquée instantanément et on peut modifier ses valeurs dynamiquement avec un rendu en temps réel, ce qui permet de pouvoir tester différentes nuances et de voir laquelle peut être la meilleure dans chaque situation.

La manière dont la palette appliquée est la suivante, les valeurs RGB de la palette sont utilisées pour les valeurs RGB des pixels, et la valeur en niveau de Gris e l'image est quand à elle utilisé pour le canal alpha* des pixels, par exemple, si un pixel à une valeur en niveau de gris de 0.3, et que la palette sélectionnée est (1.0 , 0.0, 0.0), alors la nouvelle valeur du pixel en RGBA sera (1.0, 0.0 , 0.0, 0.33).

3. Tests

3.1. Chargement

Nous avons vérifié que l'image viewer* chargement correctement les images provenant de diverses sources, notamment des URL IIIF, pour garantir sa capacité à accéder et à afficher des images provenant de différentes origines de manière fluide et fiable.

Nous pouvons nous référer à la figure 3 pour le test avec le premier URL (*https://media.getty.edu/iiif/image/e5d29650-11f8-4897-9540-54a9dd65b04f/full/max/0/default.jpg*). L'image s'ouvre parfaitement.

Pour notre second test, nous avons utilisé une image d'archive différente, voir figure 12 (*https://adore.ugent.be/IIIF/images/archive.ugent.be%3A18E02D0C-35A1-11E1-9038-61883B7C8C91%3ADS.1/full/max/0/default.jpg*).

Figure 12 : Affichage d'une autre image

3.2. Navigation

Nous avons testé les fonctionnalités de navigation, y compris le défilement et le panoramique, ainsi que le zoom, pour nous assurer qu'ils fonctionnent de manière fluide et précise. Cela garantit une expérience utilisateur agréable et réactive lors de l'exploration des images à travers l'interface du viewer.

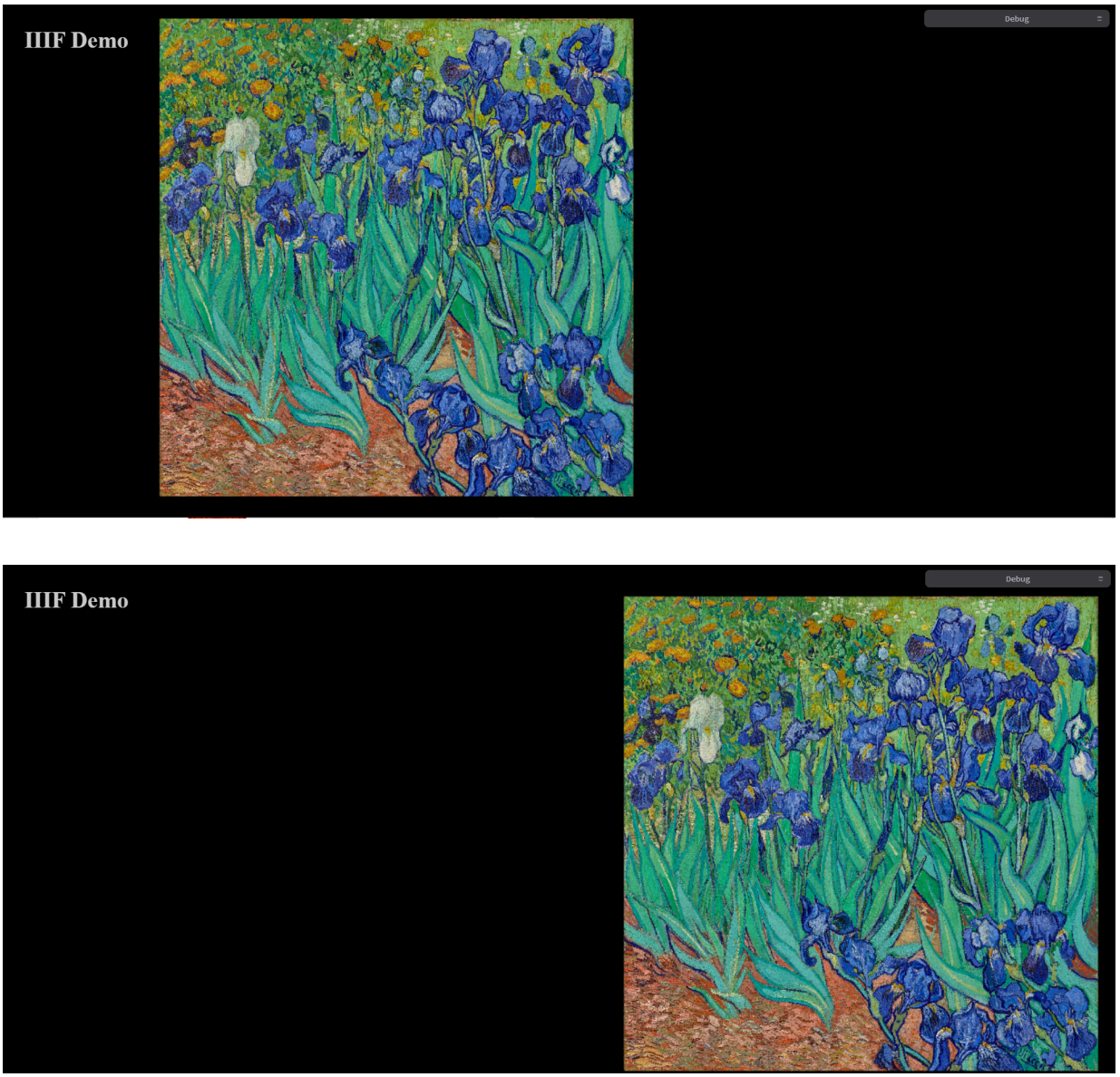


Figure 13 : Déplacement lorsque l'image est dézoomée

Sur la figure 13, on visualise assez facilement le déplacement de l'image sur l'écran. Il est fluide et ne perd pas en qualité.

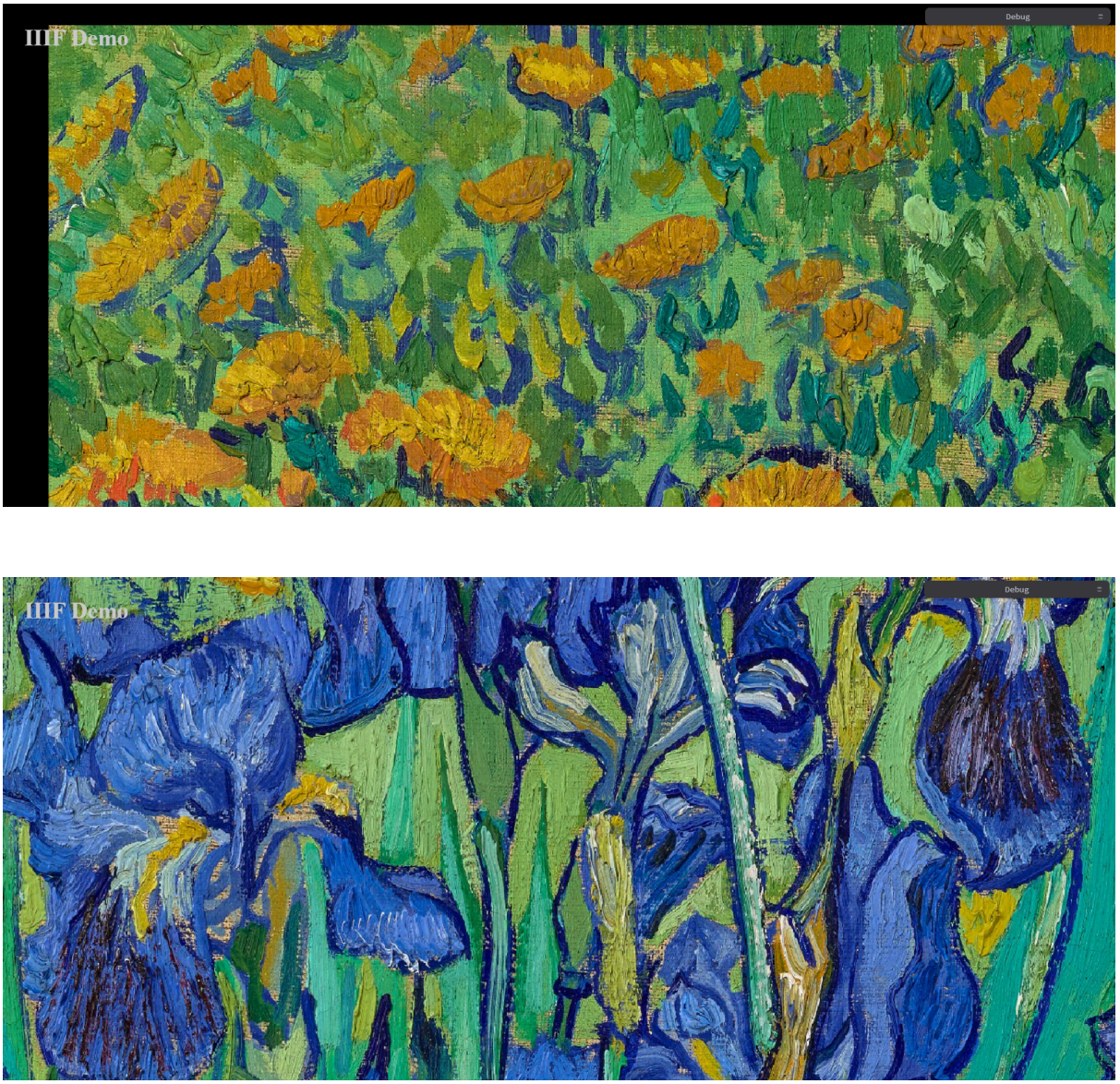


Figure 14 : Déplacement lorsque l'image est zoomée

La figure 14, en plus de démontrer la qualité du zoom, démontre que le déplacement avec image zoomée reste très fiable et fonctionne très bien.

3.3. Rendu et performances

Évaluation de la qualité visuelle des images rendues, de la fluidité des interactions utilisateur et de la vitesse de chargement des images, en particulier pour les images de grande taille.

Figure 15 : Image très zoomée

La figure 15 démontre le niveau de zoom possible. Le fait qu'il soit possible d'aller jusqu'à voir les coups de pinceau sur cette fameuse peinture, montre également à quel point la qualité de l'image rendue n'a pas été altérée.
En ce qui concerne les performances, le viewer peut charger l'image avec n'importe quelle résolution quasi instantanément, alors que sur une page web classique la même image prend une dizaine de secondes à être chargée.

3.4. Compatibilité

Nous avons évalué la qualité visuelle des images rendues, la fluidité des interactions utilisateur et la vitesse de chargement des images, en particulier pour les images de grande taille. Cette évaluation nous a permis de garantir une expérience utilisateur optimale, même avec des images volumineuses, tout en assurant des performances fluides et réactives.

3.5. Intégration

Nous avons vérifié que l'image viewer s'intègre correctement dans la plateforme Athena de Thermo Fisher Scientific, en garantissant sa compatibilité avec les systèmes de gestion de données et les autres composants de l'écosystème. Cette étape garantit une harmonie fonctionnelle avec l'infrastructure existante, assurant ainsi une utilisation fluide et cohérente au sein de l'écosystème d'Athena.

3.6. Accessibilité

Nous avons évalué l'accessibilité de l'image viewer pour les utilisateurs ayant des besoins spécifiques, en nous assurant que l'interface utilisateur est claire et utilisable pour tous. Cette évaluation vise à garantir une expérience inclusive, permettant à tous les utilisateurs, quelle que soit leur situation, d'accéder facilement aux fonctionnalités du viewer et d'interagir efficacement avec les images affichées.

Conclusion

En conclusion de notre projet de développement d'un image viewer IIIF utilisant WebGPU, nous sommes parvenus à concevoir un outil fonctionnel répondant aux besoins croissants de la visualisation d'images très haute résolution dans le domaine scientifique et académique. Ce projet représente une avancée significative dans la manière dont les chercheurs et les professionnels peuvent accéder, manipuler et analyser des images numériques complexes via une interface web interactive.

Grâce à notre travail, nous avons mis en lumière les possibilités offertes par l'intégration de technologies émergentes telles que WebGPU pour améliorer les performances et la qualité visuelle des applications de visualisation d'images. Notre image viewer IIIF ouvre la voie à de nouvelles possibilités pour la recherche en permettant une exploration plus approfondie des données visuelles et en facilitant la collaboration entre les chercheurs.

Pour l'avenir, nous envisageons d'étendre les fonctionnalités de notre image viewer IIIF en intégrant des outils d'analyse d'images avancés, tels que la détection d'objets et l'analyse de données en temps réel. Nous envisageons également d'explorer des possibilités d'intégration avec d'autres technologies émergentes, telles que l'intelligence artificielle et la réalité virtuelle, pour offrir une expérience utilisateur encore plus immersive et enrichissante.

En collaborant avec la communauté académique et les professionnels du domaine, nous sommes convaincus que notre projet continuera à évoluer et à s'améliorer, ouvrant ainsi de nouvelles perspectives pour la visualisation et l'analyse d'images dans divers domaines scientifiques et académiques. Nous sommes impatients de voir comment notre image viewer IIIF contribuera à l'avancement de la recherche et de la découverte dans le futur.

Annexes

Outils utilisés

GitHub

GitHub a été le pilier central de notre processus de développement tout au long de ce projet. En utilisant ses fonctionnalités de gestion de versions, de suivi des problèmes et de collaboration, nous avons pu organiser efficacement notre travail d'équipe, assurant ainsi une coordination transparente et un suivi précis des contributions de chacun. Son intégration fluide avec nos outils de développement a grandement facilité la gestion du code source et des workflows de déploiement.

Cantaloupe

Cantaloupe est un serveur d'imagerie IIIF open-source, performant et extensible, conçu pour stocker, distribuer et manipuler des images numériques à très haute résolution. Son architecture modulaire et sa prise en charge complète du standard IIIF en font un outil idéal pour la visualisation d'images scientifiques via une interface web interactive.

Docker

Docker est une plateforme open-source de virtualisation légère qui permet de créer, de déployer et de gérer des applications de manière efficace et portable. En encapsulant les applications et leurs dépendances dans des conteneurs, Docker offre une solution flexible et répétable pour le développement, le test et le déploiement d'applications, facilitant ainsi la mise en place d'environnements de développement cohérents et la gestion des infrastructures informatiques.

WebGPU

WebGPU est une API graphique moderne conçue pour offrir des performances graphiques élevées et une prise en charge avancée des fonctionnalités graphiques dans les applications web. En exploitant la puissance du GPU, WebGPU permet aux développeurs de créer des applications web avec des graphismes plus réactifs et plus fluides, ouvrant ainsi de nouvelles possibilités pour les jeux, les visualisations de données et d'autres applications graphiques intensives sur le web.

IIIF

IIIF, ou International Image Interoperability Framework, est un ensemble de standards ouverts visant à faciliter l'échange et l'utilisation de contenu image en ligne. Il offre une approche normalisée pour la présentation et la manipulation d'images numériques, permettant ainsi aux utilisateurs d'accéder facilement à des collections d'images provenant de différentes institutions et de les intégrer dans des environnements numériques. IIIF est largement adopté dans les domaines des bibliothèques, des musées, des archives et de l'éducation, offrant une interopérabilité accrue et une expérience utilisateur améliorée pour la visualisation d'images en ligne.

Bibliographie

[1] *Thermo Fisher Scientific*. Consulté le 15 mars 2024, à l'adresse https://fr.wikipedia.org/wiki/Thermo_Fisher_Scientific

[2] *International Image Interoperability Framework*. Consulté le 16 mars 2024, à l'adresse https://fr.wikipedia.org/wiki/International_Image_Interoperability_Framework

[3] Daniel. (2023, 12 avril). *Tout savoir sur WebGPU, la nouvelle API de Google*. Formation Data Science | DataScientest.com. Consulté le 25 mars 2024, à l'adresse https://datascientest.com/tout-savoir-sur-webgpu-la-nouvelle-api-de-google

[4] *Home*. (2024, 14 février). IIIF. Consulté le 25 mars 2024, à l'adresse https://iiif.io/

[5] *Image API 2.0*. (s. d.). Consulté le 25 mars 2024, à l'adresse https://iiif.io/api/image/2.0/

[6] *Imaging Data Management | Athena Software | Thermo Fisher Scientific - IE*. (s. d.). Consulté le 17 mars 2024, à l'adresse https://www.thermofisher.com/uk/en/home/electron-microscopy/products/software-em-3d-vis/imaging-data-management/athena-software-for-core-imaging-facilities.html

[7] Marin, J. (2023, 30 avril). En déployant WebGPU dans Chrome, Google ouvre une nouvelle ère pour les navigateurs web. *www.usine-digitale.fr*. Consulté le 25 mars 2024, à l'adresse https://www.usine-digitale.fr/article/en-deployant-webgpu-dans-chrome-google-ouvre-une-nouvelle-ere-pour-les-navigateurs-web.N2120726

[8] *OpenSeadragon*. (s. d.). Consulté le 17 mars 2024, à l'adresse http://openseadragon.github.io/

[9] *Thermo Fisher Scientific - IE*. (s. d.). Consulté le 17 mars 2024, à l'adresse https://www.thermofisher.com/fr/fr/home.html

[10] *webcomponents.org - Discuss & share web components*. (s. d.). Consulté le 25 mars 2024, à l'adresse https://www.webcomponents.org/introduction

[11] *WebGPU*. (s. d.). W3C. Consulté le 27 mars 2024, à l'adresse https://www.w3.org/TR/webgpu/

[12] *Your first WebGPU app | Google Codelabs*. (s. d.). Google Codelabs. Consulté le 17 mars 2024, à l'adresse https://codelabs.developers.google.com/your-first-webgpu-app#0

[13] *Tweakpane. (s. d.). Consulté le 27 Mars 2024, à l'adresse https://tweakpane.github.io/docs/*

Lexique

A

API (Interface de Programmation d'Applications) : Un ensemble de règles et de protocoles permettant à différents logiciels de communiquer entre eux. 7

Athena : Plateforme de Thermo Fisher Scientific offrant une gestion centralisée des données scientifiques, un suivi des workflows de traitement des données, et une interface web de visualisation des résultats, spécifiquement conçue pour répondre aux besoins de gestion, de stockage et de visualisation des données issues d'acquisitions par microscopie électronique. 7

C

Canal alpha : Une composante d'une image numérique qui stocke les informations de transparence des pixels. Il permet de contrôler l'opacité, facilitant la superposition transparente des images ou la création d'effets de transparence. 20

Cantaloupe : Un serveur d'imagerie IIIF open-source, performant et extensible, conçu pour stocker, distribuer et manipuler des images numériques à très haute résolution. 14

Chargement dynamique : Le processus de chargement de ressources à la demande pendant l'exécution d'un programme, permettant une utilisation efficace des ressources système et une réduction du temps de chargement initial. 15

CPU (Central Processing Unit) : La composante principale d'un ordinateur responsable de l'exécution des instructions et du traitement des données. 9

D

Docker : Une plateforme de virtualisation légère permettant de créer, déployer et gérer des applications dans des conteneurs logiciels. 14

F

Framework : Ensemble de composants structurels servant à créer les fondations d'un logiciel. 7

G

GPU (Graphics Processing Unit) : Une unité de traitement spécialisée dans le rendu graphique et les calculs parallèles, utilisée pour accélérer les tâches graphiques dans les ordinateurs et les appareils électroniques. 8

V

Viewer IIIF : Une application permettant de visualiser des images conformément aux spécifications du standard IIIF, offrant des fonctionnalités avancées telles que le chargement et l'affichage par tuiles ou niveaux de résolution. 11

W

WebComponent : Une technologie web permettant de créer des composants réutilisables et autonomes, encapsulant le code HTML, CSS et JavaScript dans un seul paquet pour une utilisation facile et une meilleure modularité. 3

WebGL (Web Graphics Library) : API JavaScript qui permet de créer des graphiques interactifs en 2D et 3D directement dans un navigateur web. Elle exploite la puissance du GPU de l'ordinateur de l'utilisateur pour accélérer le rendu graphique 7

WebGPU : Une API graphique moderne conçue pour offrir des performances graphiques élevées et une prise en charge avancée des fonctionnalités graphiques dans les applications web. 7

Workflows : Les séquences d'activités organisées et coordonnées pour atteindre un objectif spécifique dans un processus ou un projet. 7