Université de Bordeaux

Département informatique

Avril - Août 2021

Stage obligatoire

Master 2 - Informatique Image et Son

Ajout de fonctionnalités dans le projet Athena Volumetric Viewer

Rapport de stage

Enseignant référent :
Pierre Bénard

Entreprise:
Thermo Fisher Scientific

Auteur:
Marc Cerutti

Tuteur :
Arnaud-Marie Gallardo

351 Cours de la Libération
33400 Talence

39 Rue d'Armagnac
Quai 8.2, Bât E2
33800 Bordeaux

August 23, 2021

Sommaire

1 Remerciements **3**

2 Préface **4**

3 Contexte **5**

3.1 Entreprise . . . 5

3.1.1 Thermo Fisher Scientific . . . 5

3.1.2 FEI SAS et Coeur de métier . . . 5

3.1.3 Historique . . . 6

3.1.4 Introduction au projet Athena et Volumetric Viewer . . . 7

3.1.5 Déroulement du stage . . . 8

3.2 Sujet . . . 9

3.3 Objectifs pédagogiques . . . 9

4 Conditions de travail **10**

4.1 Équipe . . . 10

4.2 Rythme et méthodologie agile . . . 10

4.3 Matériels & Logiciels . . . 11

5 Existant: Plugin Volumetric Viewer **12**

5.1 Modèle client/serveur . . . 12

5.2 Front-end . . . 12

5.3 Back-end . . . 13

5.4 CI/CD . . . 15

5.5 Tests & Bugs . . . 15

6 Spécifications **17**

6.1 User stories . . . 17

6.1.1 Picking . . . 17

6.1.2 ScaleAxis . . . 17

6.1.3 ScaleAxis contrast . . . 18

6.1.4 ScaleAxis MetaData . . . 18

6.1.5 Colormap legend . . . 18

6.1.6 Distance 2D . . . 19

6.1.7 Angles 2D . . . 19

6.2 Émergence du GUI2D system . . . 20

6.2.1 Point de départ . . . 20

6.2.2 Concept . . . 20

6.2.3 Fonctionnalités proposées . . . 20

6.3 Planning . . . 21

7 Implémentation **22**

7.1 Choix techniques, évolutions et conséquences . . . 22

7.1.1 Outils et bibliothèques . . . 22

7.1.1.1 OpenInventor . . . 22

7.1.1.2 Protobuf . . . 22

7.1.1.3 Angular . . . 22
7.1.1.4 IOLink/IOFormat . . . 23
7.1.1.5 Cypress . . . 23
7.1.1.6 PlantUML . . . 23
7.1.1.7 Linters . . . 24
7.1.1.8 DearImgui . . . 24
7.1.1.9 PaperJS . . . 24
7.1.1.10 Conan . . . 24
7.1.2 Picking . . . 25
7.1.3 Gui2D system . . . 25
7.1.3.1 Méthode de travail . . . 25
7.1.3.2 Architecture . . . 26
7.1.3.3 Conventions & Positions . . . 27
7.1.4 Scale axis . . . 28
7.1.5 Mesures 2D . . . 29
7.1.6 Colormap legend . . . 30
7.2 Tests & bugs . . . 31
7.3 Planning effectif . . . 32
7.4 Améliorations possibles . . . 33
7.4.1 GUI2D System . . . 33
7.4.1.1 Réanalyse du besoin . . . 33
7.4.1.2 Problèmes de rendus . . . 33
7.4.1.3 Textes & Outline . . . 33
7.4.1.4 Interactions avancées . . . 33
7.4.1.5 Problèmes de cache . . . 34
7.4.2 Picking 3D . . . 34
7.5 Concepts appris ou approfondis importants . . . 35
7.5.1 User stories(US) et rôles . . . 35
7.5.2 Tests& bugs . . . 35
7.5.3 Git review, merge requests et rebase . . . 35
7.5.4 Documentation . . . 35
7.5.5 Développement & Architectures . . . 35

8 Bilan **36**

9 Annexes **37**
9.1 Offre originelle de stage . . . 39
9.2 Release Plan 1.14 Connectivity . . . 41
9.3 Exemple hiérarchie complète du Gui2D System . . . 44
9.4 Visualisation mémoire . . . 45
9.5 Cas de Tests - TestRail . . . 46
9.6 Présentation du Gui2D System . . . 58
9.7 Autres . . . 67

1 Remerciements

Je tiens à remercier dans un premier temps, toute l'équipe pédagogique de l'Université de Bordeaux, et surtout **Pierre Bénard** ayant communiqué l'offre de stage, et accompagné pour mon deuxième stage en Master.

Je remercie aussi la bienveillance de tout le pôle de Bordeaux Thermo Fisher Scientific et des personnes ayant travaillé avec moi (4.1), qui m'ont permis que ce stage se passes dans de très bonnes conditions.

Je voudrais spécialement remercier mon tuteur de stage **Arnaud-Marie Gallardo** pour sa patience, son professionnalisme et ses conseils, **Nicolas Pascaud** pour son cours des User Stories (qui serait d'ailleurs excellent dans le cadre d'un cours universitaire), et **Julien Robert** pour toutes ses explications sur la partie test.

Je remercie enfin tout ceux qui ont relu mon rapport et notamment ma famille.

2 Préface

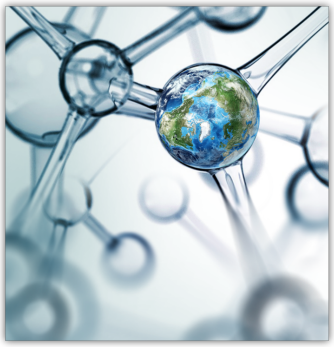
Ce document rend compte du travail accompli lors de mon stage obligatoire de fin de Master Informatique spécialité Image et Son à l'Université de Bordeaux. Il abordera autant le travail effectué, que les difficultés et les améliorations possibles sur celui-ci.

Bien que obligatoire, ce stage est pour moi surtout l'occasion de compléter mes compétences et ma vision du monde professionnel, dont j'ai déjà eu une première expérience l'année dernière en 2020 à SQLI. Ce fut l'occasion de chercher un stage avec une éventuelle insertion professionnelle à la suite.

C'est pour cela que je fus fort intéressé par ce stage à Thermo Fisher Scientific, portant à la fois sur le domaine de la 3D, mais aussi sur l'extension de l'outil du Volumetric Viewer, outil web de visualisation 3D, qui est utilisé par des scientifiques et dont les exigences en termes d'expérience utilisateur sont précis.
Le sujet initial du stage était l'opportunité aussi de sortir de ma zone de confort et d'apprendre tout l'aspect Front-end et communication client-serveur au sein de celui-ci.

La proposition de stage d'origine est disponible en Annexe (9.1).

3 Contexte



3.1 Entreprise

3.1.1 Thermo Fisher Scientific

Thermo Fisher Scientific est une multinationale, leader mondial en termes de services pour la science, avec un revenu annuel de 30 billions de dollars. Sa politique est de proposer à ses clients, souvent issus de la recherche en laboratoire, des solutions pour un monde plus sain, hygiénique et sûr. Thermo Fisher Scientific est là en tant que support à ces personnes et/ou institutions, et propose des solutions et matériels dans des domaines tels que la recherche, les équipements de laboratoire, la biologie moléculaire, les tests cliniques, l'acquisition de donnée (microscopie notamment), le traitement d'images, la reconstruction 3D, l'analyse structurelle et chimique, la pharmacologie, le médical, et aussi la gestion et l'intendance de microscopes.

3.1.2 FEI SAS et Coeur de métier

Le stage s'est donc déroulé au pôle de Bordeaux de FEI SAS, sous filière rattachée à Thermo Fisher Scientific, qui développe et commercialise des solutions d'acquisition, d'analyse et de visualisation d'images 3D (fig.1 et tab.1) issues notamment de la microscopie. Les produits permettent la gestion d'images de grandes tailles, et leur visualisation en 2D, 3D et plus selon les besoins. Les activités couvrent toute la chaîne de valeur, de l'édition à la commercialisation de solutions logicielles, l'installation/déploiement de ceux-ci, la formation, le support hotline, ainsi que le développement de services personnalisés pour des clients.

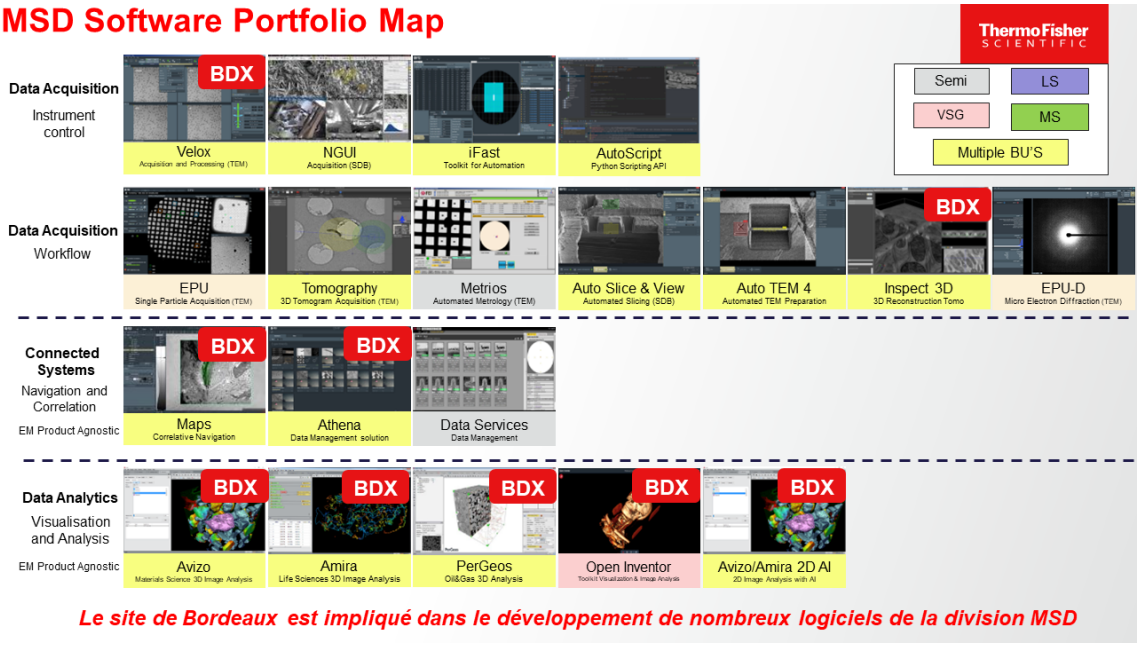


Figure 1: Portfolio des différentes solutions de FEI SAS.

Solutions	Quelques Clients
Technologies de visualisation avancées pour la gestion et la visualisation interactive de gigantesques données sismiques et de réservoir pour l'exploration pétrolière (Open Inventor Oil&Gas)	• Schlumberger (USA et Royaume-Uni) • Halliburton (USA) • Siemens (Allemagne) • Shell (USA et Pays-Bas) • Maptek (Australie) • BGP, CNPC (China)
Solutions d'analyse et de visualisation d'images en Science des matériaux (Avizo)	• University of Manchester (Royaume-Uni) • GE Aviation (USA) • Baker Hughes (Chine, Hongrie, Allemagne) • Unison Engine Components (USA) • Shangai Eastern Science-Tech Machien (Chine) • KAUST (Arabie Saoudite)
Solutions d'analyse et de visualisation d'images en Biologie cellulaire et neurosciences (Amira)	• Max Planck (Allemagne) • Ching-Yeh (Taïwan) • Stryker (Allemagne) • Axial 3D (Royaume-Uni) • Nuohai Life Science (Chine) • National Institutes of Health (USA)
Solutions d'analyse et de visualisation d'images en Digital Rock (PerGeos)	• Techno Info (Royaume-Uni) • Devon Energy (USA) • Petrobras (Brésil) • Stratum Reservoir (USA) • Beijing Run Ze Chuang Xin (Chine) • Colchis Petroconsulting (Chine)

Table 1: Exemple de solutions et clients associés de FEI SAS.

3.1.3 Historique

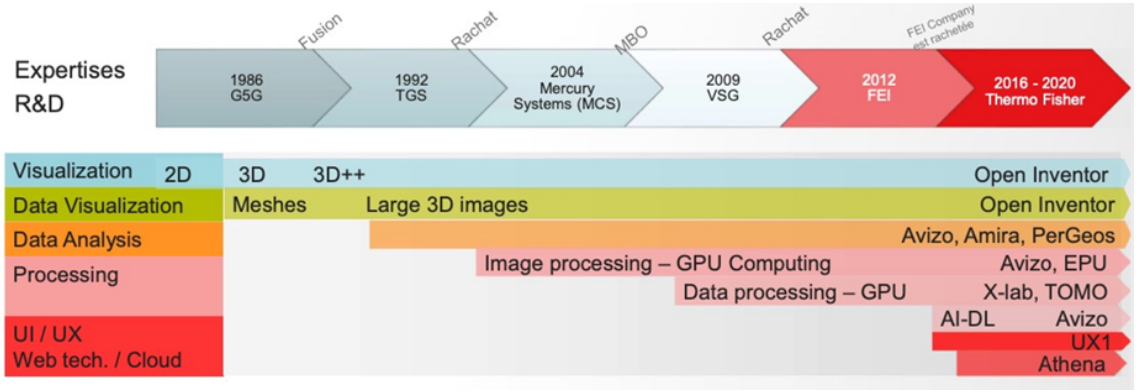


Figure 2: Dates clés de l'évolution du pôle de Bordeaux, de C5G à nos jours.

À la suite de leurs études à l'Université Bordeaux I en 1986, 5 étudiants en mathématiques et informatique créent la société G5G (Graphisme 5e Génération) dédiée à la création de logiciels de visualisation 2D/3D. À cette époque, la 2D et a fortiori la 3D n'était pas étendue, les outils ne permettaient pas d'afficher une image, mais des points et des traits de quelques couleurs tout au plus. Les besoins étaient pourtant là et, l'affichage d'images et en 3D était innovant et challenging. C'est dans les années 90 avec l'essor de nouveaux matériels informatiques, les cartes graphiques et jeux vidéo que la visualisation se démocratise et commence à entrer dans le quotidien. De la maison modélisée à la réalité augmentée, la visualisation 3D est une technologie qui est devenue essentielle, notamment pour les entreprises. L'effectif de l'équipe était d'environ 20 personnes pour un chiffre d'affaire de 1 million de dollars.

En 1995, G5G fusionne avec la société américaine TGS (Template Graphic Software). En accord avec la société Silicon Graphics Inc, TGS devient le développeur/distributeur source de la solution Open Inventor qui devient le produit phare. En 2002, le rapprochement avec la société allemande INDEED permet le développement de la solution Amira comme 1er élargissement du portefeuille de produits.

En 2004, la société est rachetée par l'entreprise américaine Mercury Computer Systems Inc (MCS) qui commercialise alors des cartes d'acquisition d'images équipant les scanners. C'est le début de l'intégration dans le marché des équipements d'imagerie. Au travers d'une opération de rachat, le site devient alors VSG (Visualization Sciences Group) en 2009. Après 25 ans d'existence, l'effectif avait doublé avec environ 43 personnes pour un chiffre d'affaire de 6 millions d'euros.

En 2012, le Groupe américain FEI, qui fabrique des microscopes, souhaite vendre les packages logiciels associés, et rachète donc VSG. Le site bordelais continue le développement de ses solutions logicielles propres Open Inventor, Amira, Avizo, PerGeos, tout en élargissant sa gamme de services avec des logiciels de processing d'image et de données.

En 2016, Le Groupe international Thermo Fisher Scientific, rachète le Groupe FEI. Depuis lors, FEI SAS est associée à la marque Thermo Scientific, dans la division MSD (Material Science Division) en gardant la dénomination VSG pour la Business Unit à l'échelle groupe et en maintenant son nom juridique FEI SAS. Le site commence à intégrer les nouveaux leviers/concepts technologiques comme l'expérience utilisateur (UX), l'interface utilisateur (UI) et technologies cloud en lien avec les évolutions du marché pour faire évoluer les produits et services.

3.1.4 Introduction au projet Athena et Volumetric Viewer

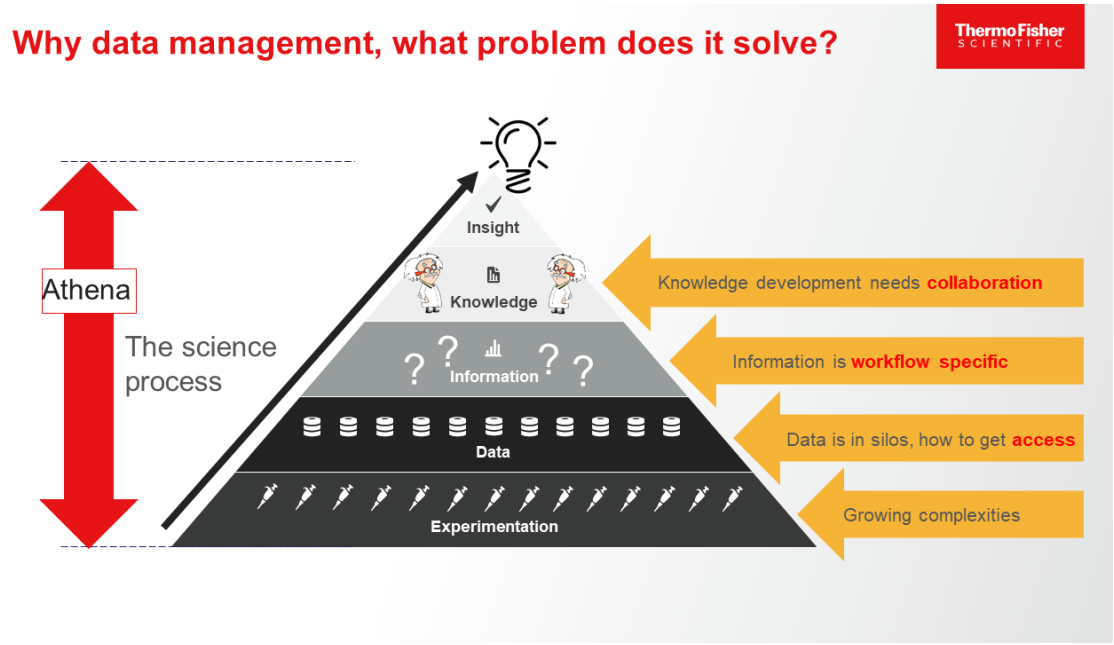


Figure 3: Principe du coeur de métier d'Athena.

Le projet Athena[15] commencé en fin 2017 est un projet de plateforme permettant d'accompagner les scientifiques durant leurs expérimentations et de regrouper toutes les données sur un même serveur, qui serait accessible aussi aux collaborateurs. Le principe est de fournir une plateforme moderne permettant de centraliser et partager les données, connaissances et résultats d'expérimentations. Elle permettrait ainsi d'accompagner les scientifiques tout du long du processus d'expérimentation (fig.4), de l'acquisition jusqu'à l'analyse en passant par le traitement de données.

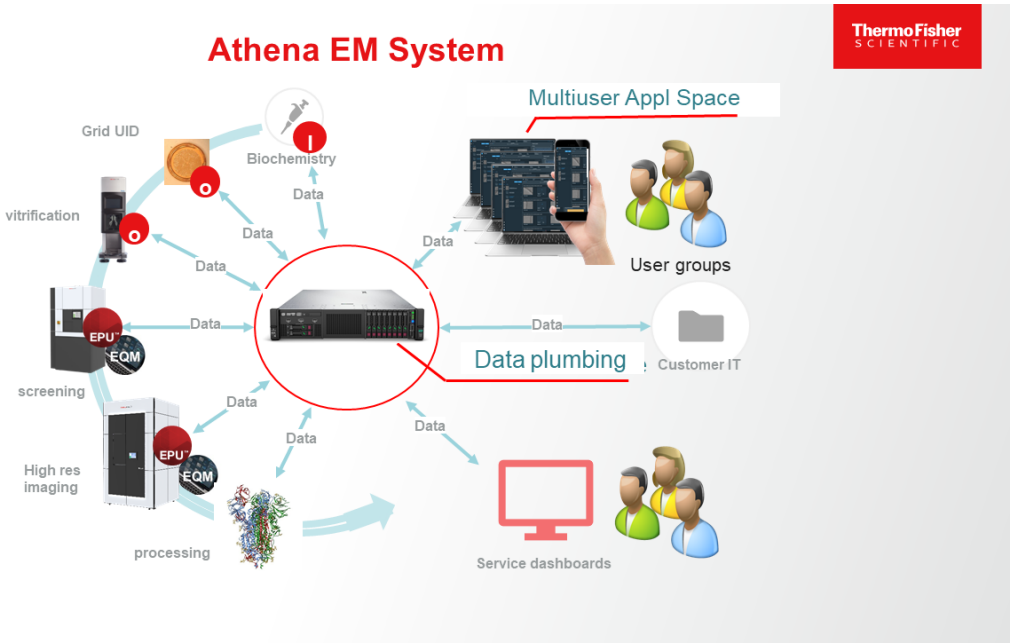


Figure 4: Cas d'utilisation de la plateforme Athena.

Le Volumetric Viewer, ou anciennement appelé 3D Viewer, est donc un service proposé par cette plateforme permettant d'afficher les volumes 3D acquis et reconstruits, qui ont été précédemment mis sur la plateforme.

3.1.5 Déroulement du stage

Étant encore dans la crise sanitaire du COVID-19, le stage s'est déroulé surtout en télétravail, avec 2 jours par semaine autorisées sur site comme défini par les règles sanitaires nationales, du lundi au vendredi de 9h00 à 12h00, et de 14h00 à 18h00.

Je fus rattaché tout d'abord à l'équipe DataConnect du 3 Avril jusqu'au 10 Mai, où peu après une restructuration, je fus finalement intégré à l'équipe Connectivity (fig.5) du projet Athena jusqu'au 31 Août 2021.

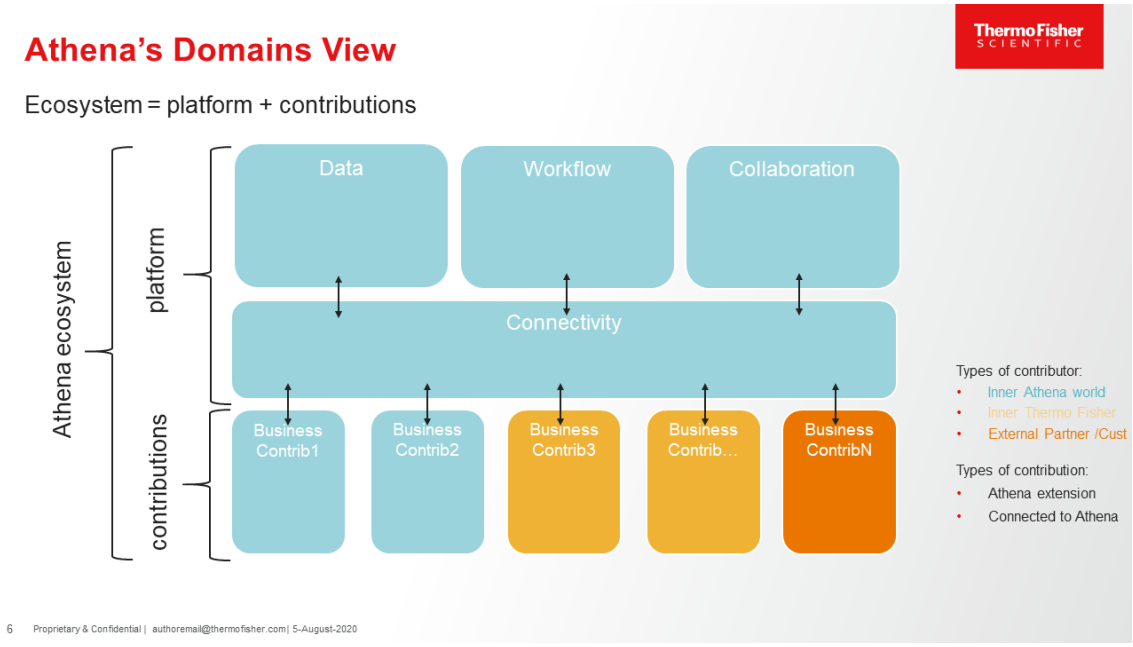


Figure 5: L'écosystème du projet Athena après restructuration avec les collaborateurs.

3.2 Sujet

Dans sa finalité, le but de ce stage est d'étendre les fonctionnalités du Volumetric Viewer (fig.6), et donner de nouveaux outils aux scientifiques pour visualiser et analyser les données 3D tout en restant le plus simple d'utilisation possible. Les fonctionnalités se sont concentrées sur des interfaces et des outils de mesures affichant de plus amples informations sur les collections de données (datasets), sur les vues en 2D. Il existe par contre dans Athena un autre viewer spécialisé pour la 2D, Mirador[11]. L'intérêt d'étendre le Volumic Viewer plutôt que Mirador est parce que il ne permet pas de visualiser des données volumiques.

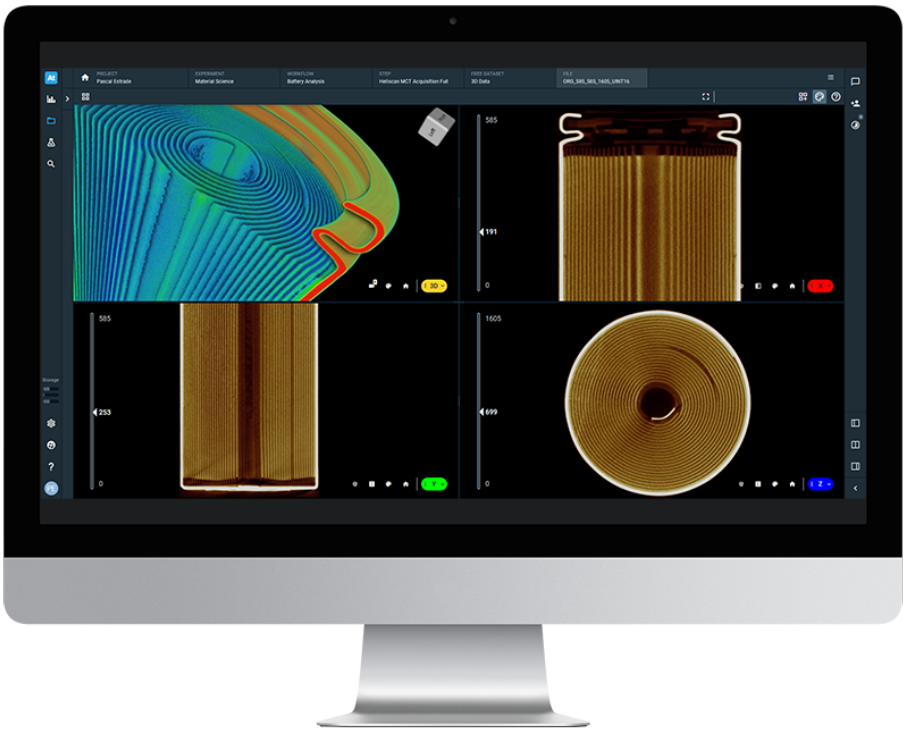


Figure 6: Interface de base du Volumetric Viewer.

3.3 Objectifs pédagogiques

Les objectifs pédagogiques suivis durant ce stage ont été les suivants :

- Apprentissage des technologies utilisées dans le Volumetric Viewer, notamment :
  - OpenInventor
  - Protobuf
  - Angular et TypeScript

  Ainsi que d'autres outils plus variés (voir 4.3 et 7.1.1).
- Approfondissement et application des outils de synthèse 3D.
- Apprentissage de la méthodologie agile Lean et rituels associés (Sprint Planning, Product Backlog Reffinement, Retrospective).
- Apprentissage de la rédaction de user stories et de l'outil Jira.
- Approfondir les outils Git appliqués à un contexte industriel (Review, Fork, Pull request, Rebase, Refactoring).
- Être source de nouvelles idées et innovations, et participer au développement du produit final (réunions, review avec les stakeholders).
- Apprentissage des compétences de coopération et de communication entre plusieurs services.
- Apprentissage et mise en place de procédures de tests appliqués au Front-end.

4 Conditions de travail

4.1 Équipe

Je fus sous la tutelle de **Arnaud-Marie GALLARDO** (développeur logiciel) qui m'a accompagné durant tout le projet. Étant à l'origine de celui-ci et ayant développé l'outil RemoteViz (7.1.1.1), ce fut mon premier interlocuteur en cas de questions sur le Volumetric Viewer. Puis, j'ai travaillé aussi avec **Julien ROBERT** (QA) et **Aurélien BREVET** (architecte logiciel) qui m'épaulaient sur les tâches de test et d'architecture respectivement. Avant la restructuration du service DataConnect, je travaillais également avec **Fréderic BON** et **Thomas PIZON** qui développaient l'outil interne récent IOLink/IOFormat (7.1.1.4). Après la restructuration, je travaillais avec **Nicolas PASCAUD** (Scrum master), **Roland HEMMER** et **Gilles DURAND** (développeurs logiciels) qui furent mes sources de conseils les plus proches concernant la méthodologie et le formalisme. Je travaillais aussi au départ avec **Jean-Paul Digeon**, qui était mon manager et occupait le rôle de Product Owner, mais suite à un congé maladie il fut remplacé par **Emanuel REMIA** courant Mai 2021. J'étais également en collaboration avec l'équipe du logiciel OpenInventor (7.1.1.1) surtout vers la fin du stage pour proposer l'outil du Gui2D system (6.2) au sein de celui-ci, avec notamment **Morgan MATHIAUT** et **Valentin SARTHOU**. Et enfin, je fus en contact avec d'autres services et personnes, comme **Pascal ESTRADE** (chargé de la communication commerciale) qui demandait des retours au collaborateur externe **3DImagination**, ou encore comme **UX1**, service chargé de la création des interfaces et de l'ergonomie de l'application.

4.2 Rythme et méthodologie agile

Il y eu deux périodes qui ont quelques différences quand à la méthode d'application des méthodologies agiles.

Avant la restructuration, les journées étaient rythmées par une réunion courte le matin, le Stand-up, et régulièrement toutes les semaines ont faisait un Product Backlog Reffinenement (PBR) afin de préciser et rédiger les user stories, et toutes les deux semaines, durée d'un sprint, on faisait le point sur l'avancement. Il y avait régulièrement une réunion entre les différents acteurs sur le Volumetric Viewer appelé Synchro d'Athena pour discuter des retours utilisateurs et des avancées. La prise de tâche était plutôt libre, et on utilisait comme support un mélange entre Kanban et Backlog tiré de Scrum assez léger sur Jira.

Après la restructuration, les routines se sont formalisées selon la méthodologie Lean, avec notamment au début de chaque Sprint un "Sprint Planning" où l'on prend les tâches, avec à la fin de ceux-ci un bilan plus approfondi sous forme de Retrospective, et juste après la Review avec les stakeholders où le Volumetric Viewer fut intégré lors des avancements. Le Stand-up est devenu Daily, et les Synchros ont été annulées. La méthodologie Lean ne se limite pas qu'à ça, elle intègre aussi une organisation verticale entres les différents services, avec des représentants et des réunions globales, mais qui ne me concernaient pas.

Mes conditions de travail ont été très stables même durant cette restructuration, car je suis resté avec mon tuteur de stage qui fut réaffecté au service Connectivity, principale raison que je sois rattaché à ce service et non pas Collaboration, dont les responsabilités sont plus proches du travail que j'ai accompli.

4.3 Matériels & Logiciels

En télétravail, mon matériel fut le suivant :

- Ordinateur portable et câbles. Os Windows 10.
  Dell Precision 7550, 15.6", Intel i7, 32GB RAM, 512GB SSD, Nividia Quadro T100
- Casque/micro
- Second écran
- Adaptateur Display port
- 3 livres sur OpenInventor

En présentiel, j'avais à ma disposition un deuxième écran, et j'avais les accès à une machine de tests partagés (disponible aussi en SSH), qui me permettaient de tester sous Linux le projet. Cette machine sinon s'occupait des tests CI/CD.

Je me suis servi de nombreux logiciels et bibliothèques durant ce projet, dont en voici une partie que je ne les présenterai pas d'avantage dans ce rapport, car assez communs. J'ai utilisé des outils de communications tel que la suite Microsoft Office comme Outlook et Teams, des IDE, des outils de développement comme Microsoft Visual Studio (2017), Visual Studio Code, CMake, Git, ainsi que des outils plus généraux, de capture comme ScreenToGif, OBS Studio ou encore ShareX, de traitement d'images comme Gimp, Inkscape, et même un outil pour renommer plus facilement les fichiers, AdvancedRenamer.

Par contre, certains outils méritent que je les détaille ci-dessous, car ils sont impliqués indirectement dans le projet, mais les plus importants sont en 7.1.1.

- Avizo[14]. Solution multi-plate-forme de visualisation et d'analyse de données 3D. Il m'a permis de voir les conventions d'interface au sein de l'entreprise et de faire le composant de ScaleAxis (7.1.4) en partant de leur code des ScaleBars.
- LightImageViewer. Utilitaire interne pour afficher des images 2D et leur metadatas, utilisé durant le développement de IoLink/IoFormat.
- RapidEnvironmentEditor[3]. Utilitaire permettant de changer/sauvegarder rapidement les variables d'environnement sous Windows.
- LockHunter[9]. Utilitaire permettant d'analyser et traquer les processus bloquant la lecture/écriture de fichier.
- Valgrind Massif Visualiseur[21]. Outil permettant de visualiser les données Massif Valgrind sur la consommation et les fuites mémoire. Vous retrouverez en annexe 9.4 des données massif affiché avec un autre visualiseur online.

5 Existant: Plugin Volumetric Viewer

5.1 Modèle client/serveur

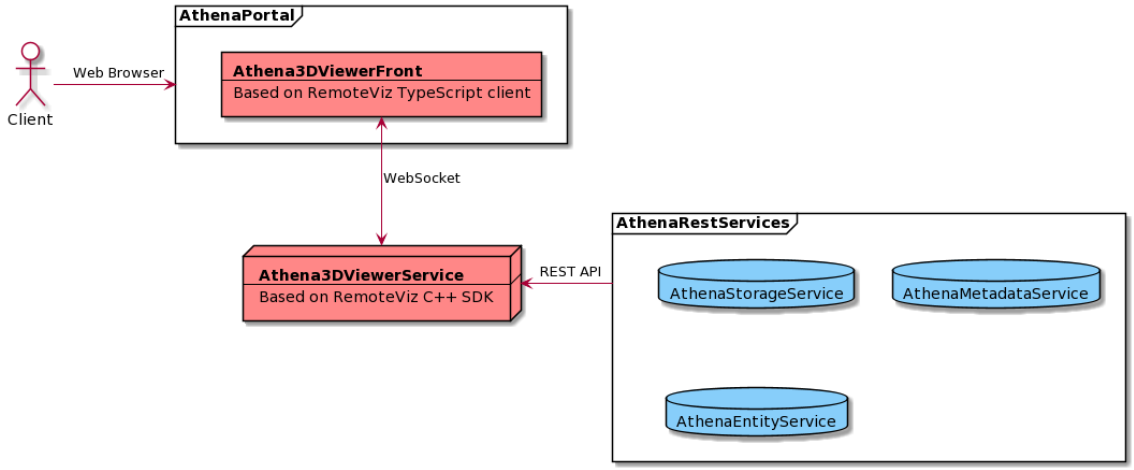


Figure 7: Modèle client/serveur Volumetric Viewer Plugin.

Le Volumetric Viewer permet via une interface web, le Portal d'Athena, d'intéragir et de visualiser des données 3D complexes et lourdes sur des ordinateurs de bureau et tablettes mobiles. Ce n'est pas possible normalement sur des plateformes limitées en puissance, sans CPU et GPU appropriés, mais en séparant le rendu côté Back-end et l'interface côté Front-end, on peut ainsi visualiser des données de 16 Go maximum (limite en général de la RAM, dépendant de la machine), sans pour autant que les interactions et l'ergonomie de la solution soient impactées, c'est-à-dire au minimum 30 FPS avec une bonne connexion.

Il est important de souligner que le service du Volumetric Viewer limite le nombre d'utilisateurs a une seul personne avec un GPU au service associé, ce qui fait que l'on n'a pas à l'heure actuelle de gestion partagée du GPU. Cela simplifie les cas d'utilisations. Athena s'occupe donc de créer et libérer les services suivant les besoins.

La solution se base sur les possibilités qu'offrent OpenInventor[17] avec les modules VolumeViz, permettant l'affichage de volume 3D en voxel, et RemoteViz pour streamer le rendu et envoyer les images ou vidéos au Front (illustré fig.7) (détails 7.1.1.1). Aussi, le Back-end se repose sur les solutions de IoLink et IoFormat (7.1.1.4) qui permettent de charger les données depuis la base via une API Rest. Enfin, le Back-end et le Front-end s'échangent des informations comme les interactions utilisateurs via un protocole Protobuf (7.1.1.2) qui assure l'interactivité de la solution.

5.2 Front-end

La partie Front-end est une bibliothèque Angular en HTML5 qui utilise RemoteViz (7.1.1.1) cité précédemment afin d'afficher les images calculées par le rendu Back-end.

Cependant, pour pouvoir tester et développer le plug-in sans être dépendant du développement de la plateforme Athena et être obligé de le déployer pour tester, le Front-end du Volumetric Viewer est séparé en 2 parties. La première partie **'athena-viewer3d-lib'** est la bibliothèque Angular réellement utilisée, et la seconde **'viewer-mock'** est une version simplifiée du portal permettant de déployer en local pour développer et tester. Ensuite au sein même de la bibliothèque, le projet suit une structure typique de composants Angular (7.1.1.3).

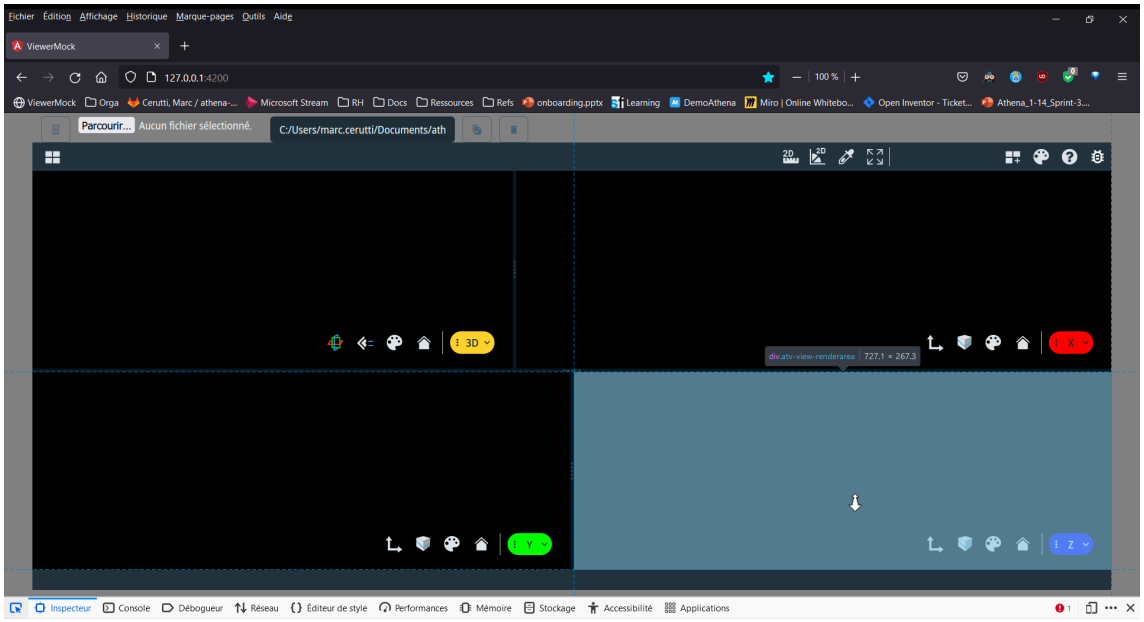


Figure 8: Screenshot du ViewerMock.

L'interface suit la logique suivante : la racine du projet est le ViewerMock qui affiche la bibliothèque (fig.8). Celle-ci contient hiérarchiquement le Viewer, qui lui même contient des Views organisés en Layout. Chaque View contient le RenderArea de RemoteViz qui s'occupe réellement de l'affichage du stream reçu du Back-end, ainsi que de l'échange des interactions utilisateurs entre Back-end et Front-end propre à chaque view. Chaque View peut être soit 2D (OrthoSlices X-Y-Z), soit 3D (Volume), et a son propre état. Une des Views dite "principale" permet de garder active la connexion entre Back-end et Front-end. Le viewer a la responsabilité d'organiser les views.

Hormis ces principaux composants, on a des menus et des panels permettant la personnalisation de l'état du Viewer et des Views, qui possiblement peuvent envoyer des messages au Back-end pour par exemple changer le rendu des Colormaps ou interagir avec la caméra.

En complément de toute cette partie Angular visible, on a les "Services", code gérant des événements et échangeant des données entre les composants, et les "Helpers", code simplifiant la manipulation de certains aspects spécifiques comme les curseurs ou l'affichage scientifique.

On reparlera aussi de la partie test, des environnements et Cypress associé au Front-end en 5.5.

5.3 Back-end

La partie Back-end quant à elle (illustré fig.9), suit cette même logique Viewer/View et a donc dans chaque View un graphe de scène propre OpenInventor (fig.10) correspondant à la ViewScene. On peut ensuite suivant le type d'interface 2D ou 3D accéder à certaines informations par le composant View Back-end qui fait office d'interface. On retrouve aussi cette logique du Viewer Front-end, qui servait au départ qu'à gérer la collection de View.

Les autres composants importants sont le RenderAreaListener, qui reçoit/envoit les messages au front et va modifier la scène en conséquence, et le RawLDMVolumeReaderIoLink, qui s'occupe du chargement des données grâce à IoLink, et des optimisations comme de la visualisation 3D multiscale. Il y a plusieurs formats de fichiers pris en charge, des fichiers seuls comme RAW, MRC, TIFF, DICOM, et des formats multifichiers comme DICOM et TIFF.

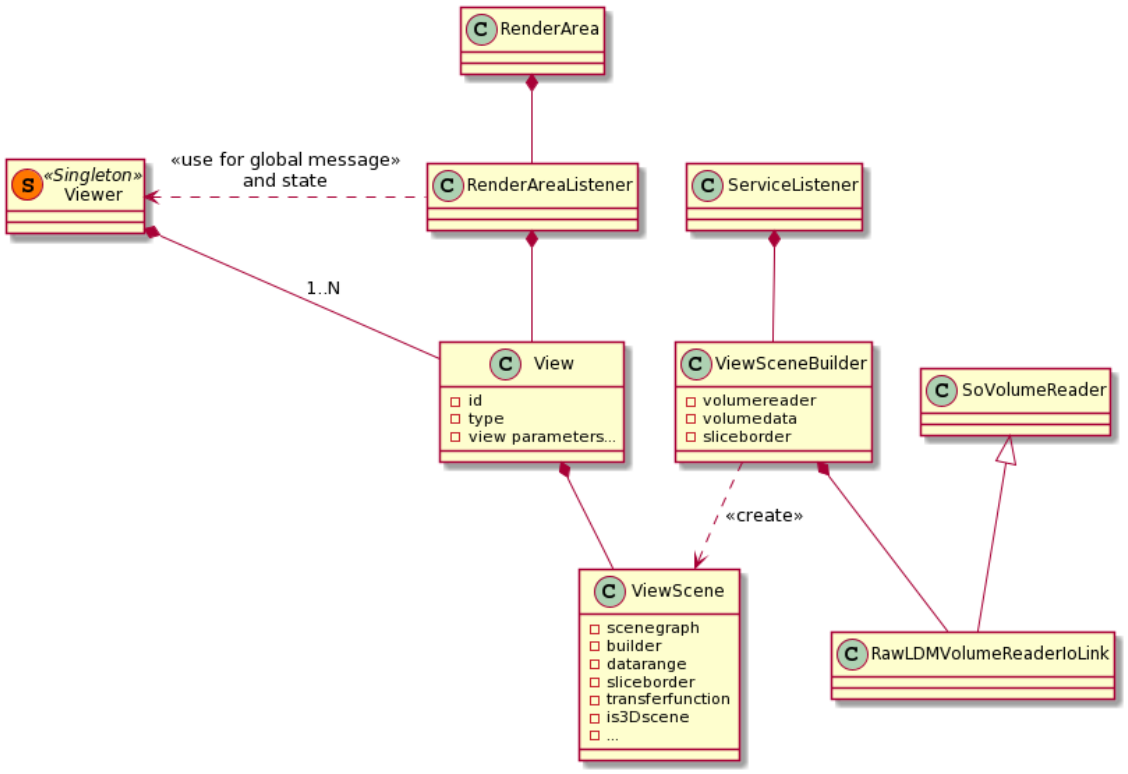


Figure 9: Back-end Volumetric Viewer.

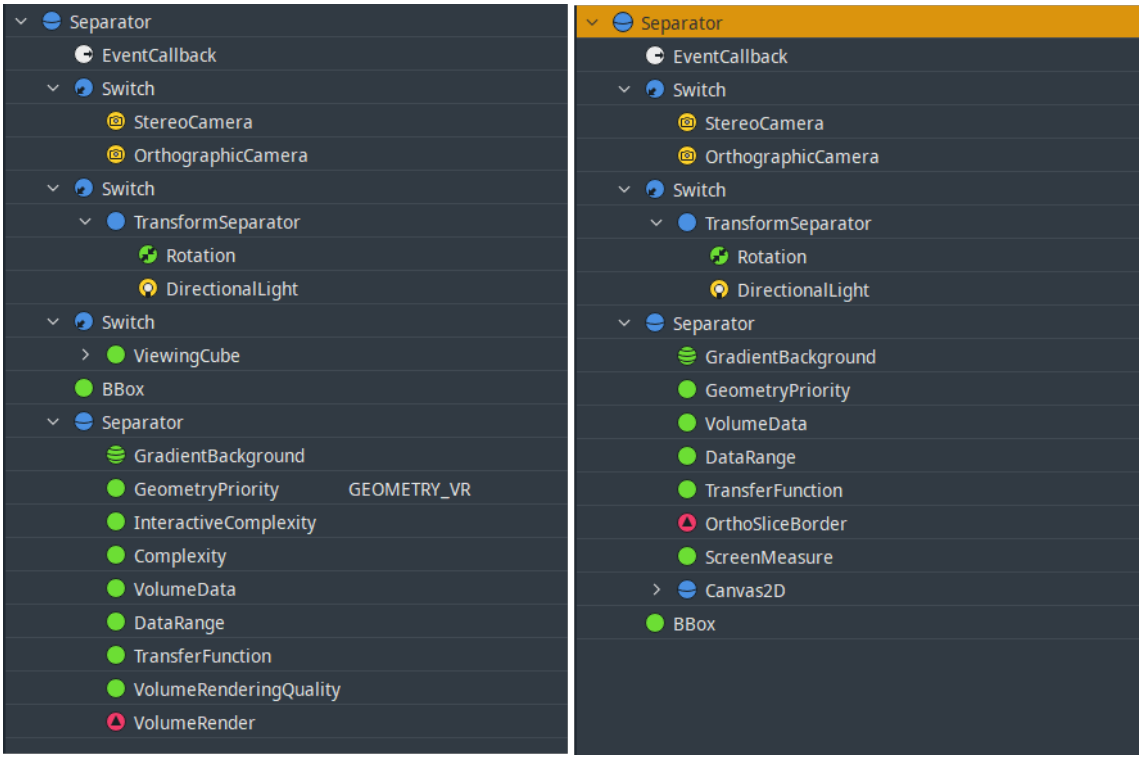


Figure 10: Graph des scènes OpenInventor, 3D à gauche, 2D à droite.

5.4 CI/CD

Le projet a une méthode de travail qui permet l'intégration continue et l'automatisation des tests. Les développeurs du Volumetric Viewer ont chacun leur propre Fork Gitlab du projet, ce qui permet de développer sans souci sur sa version. Une fois que la fonctionnalité est faite, la procédure standard est de faire une merge request sur le repository principal afin de reviewer le code et de lancer des tests automatiques (fig.11) plus approfondis que sur les forks. Dés que c'est validé, c'est le rôle des QA de rajouter/faire des tests complémentaires, en collaboration avec les développeurs, et s'occupent ensuite du versionning et du déploiement de la fonctionnalité. Les packages et les versions du Volumetric Viewer associés utilisent le versionning sémantique Major.Minor.Patch-label.

En raison des automatisations entre Gitlab et Jira, une des contraintes est de nommer les branches de fonctionnalités selon la convention suivante :
ABC(ACRONYME PRÉNOM-NOM 3 LETTRES)/NOM_DE_FONCTIONNALITÉ/ABC_XXX(ATHENA JIRA + NUMÉRO TICKET)
Exemple : MCE/COLORMAP_LEGEND/ATH-350.

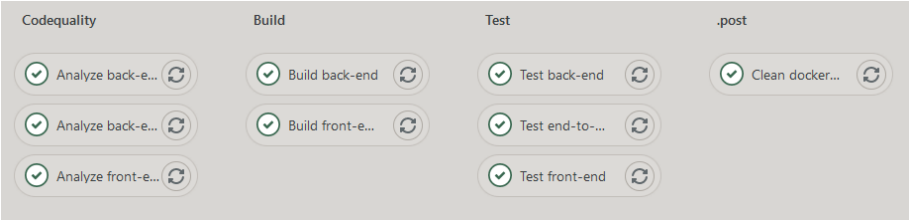


Figure 11: Pipeline Gitlab principal.

5.5 Tests & Bugs

Les tests dans ce projet sont normalement définis dès la rédaction des User Stories qui en plus d'expliquer la fonctionnalité, définissent les cas à tester sous forme de critères d'acceptation (4.2) pour les QA. Ces QA maintiennent la liste des tests grâce à l'outil TestRail[18] (fig.12) où on peut voir dedans s'ils sont manuels ou automatiques, leurs étapes et leurs éléments à vérifier, ainsi que leurs tickets et bugs associés. D'un point de vue développeur, cet outil permet aussi de lire les logs et de déterminer les causes d'échecs pour les tests automatiques.

Il existe plusieurs types de tests. Il y a les automatiques, dont seuls les fichiers RAW sont gérés à l'heure actuelle, et les tests manuels. Il y a les tests unitaires cotés Front-end, et les tests end-to-end. Mais il n'y a pas de tests côté Back-end, car le code dans cette partie vient principalement de OpenInventor et est déjà testé. Au pire des cas ce sont les tests end-to-end qui couvrent les cas où le Back-end ne fonctionne pas (Protobuf ou OpenInventor). Les tests unitaires Front-end sont fait grâce à Angular CLI, et les tests end-to-end grâce à l'outil Cypress (7.1.1.5). Une chose par contre que j'ai trouvé très intéressante pour les tests de non-régression, sont les tests de comparaison entre screenshots d'interface. Ces comparaisons sont par contre sensibles aux différences, comme le calcul est juste un Mean Squared Error sur les deux images, et nécessite un seuil pour ne pas se déclencher sur des faux-positifs. Une situation d'erreur déjà arrivé fut la mise à jour de navigateur, décalant les pixels affichés.

En ce qui concerne les bugs, ils sont recensés et gérés sur le Jira du projet et ont au minimum comme description "Step to reproduce", "Result", "Expected result" pour être clairs et compréhensibles (exemple fig.13).

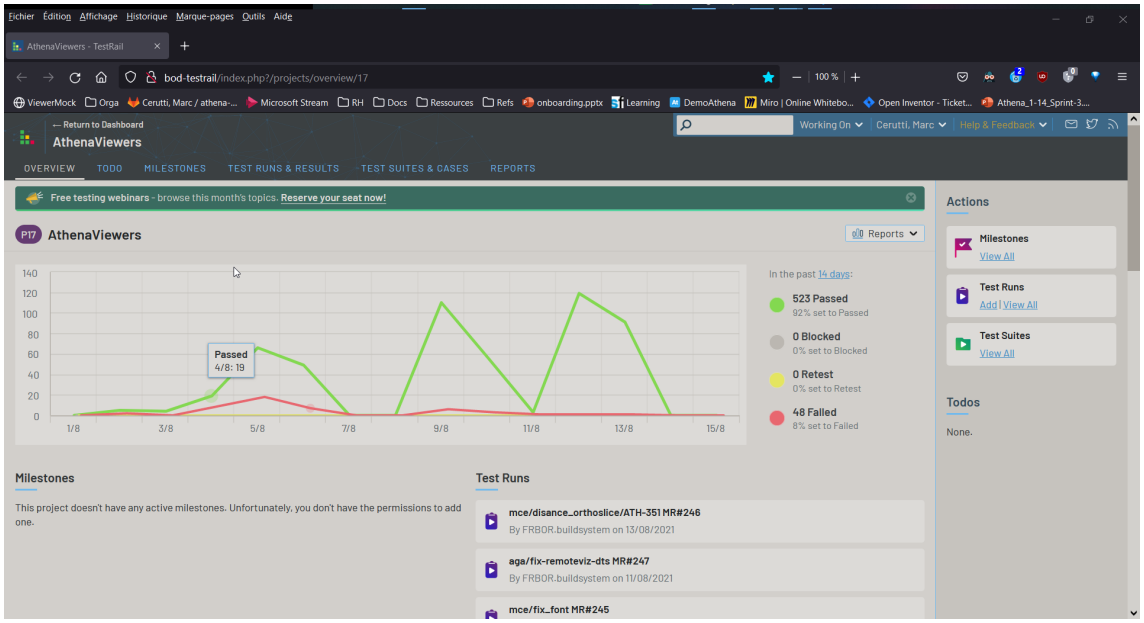


Figure 12: TestRail[18], regroupant les cas de tests et contenant les rapports des pipelines effectués.

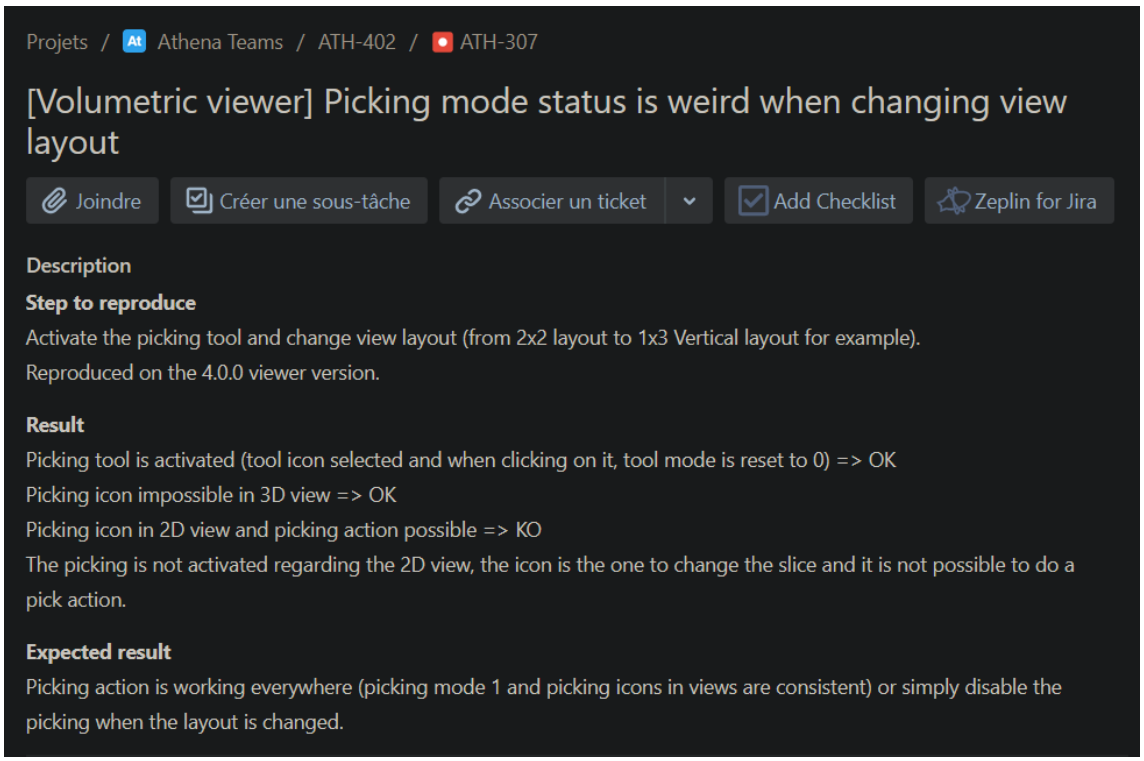


Figure 13: Exemple de bug résolu sur le Picking.

6 Spécifications

6.1 User stories

6.1.1 Picking

Create a picking mechanism in 2D view of the volumetric viewer.

Acceptance criterias :

I am a user (Portal, 3D viewer).

I do:

1. a click on the picking button of the viewer main menu
2. a click on the picking button of the viewer main menu
3. a click on the picking button of the viewer main menu
4. a picking on a 2D view of a volumetric image

I check:

- the picking button is always visible on the common toolbar.
- I obtain a valid value in the status bar, composed of the value and the voxel position.
- I have an error feedback with a precise reason when I enter invalid values.

6.1.2 ScaleAxis

Display a scale axis, to provide information about the scale of the shown data elements.

Acceptance criteria:

I am a user.

I do a click on the scale button of the local 2D view where I want to display the axis.

I check:

- I see the display or removal of the scale axis on the 2D view of a volumetric image, depending on its previous state.
- the size of the object at different zoom levels is coherent with the image data .A test dataset will have to be provided by the development team.
- the position of the object is at the bottom-left corner.
- It rescales with the modification of the view size.

6.1.3 ScaleAxis contrast

Improve the contrast on the scale axis, and include the UX1 feedback on the volumetric viewer.

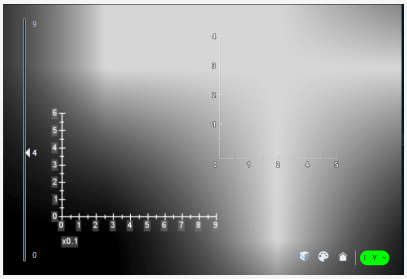
Acceptance criteria:

I am a user

I do a click on the scale button of the local 2D view where I want to display the axis.

I check:

- I can better see white-on-white texts
- it looks like the attached screenshot



6.1.4 ScaleAxis MetaData

From the metadata given by the user, display the base unit of the dataset in the scale axis. It should change based on the actual zoom level. We can create a custom dataset to test Athena Metadata unit.

Acceptance criteria 1 (Athena metadata available):

I am a user

I do a click on the scale button of the local 2D view where I want to display the axis

I check the unit is coherent to dataset metadata based on the zoom level.

Acceptance criteria 2 (no Athena metadata available):

I am a user

I do a click on the scale button of the local 2D view where I want to display the axis

I check the unit is not displayed and we are using voxel position as scale.

6.1.5 Colormap legend

We want to display the colormap legend in the middle top of our view.

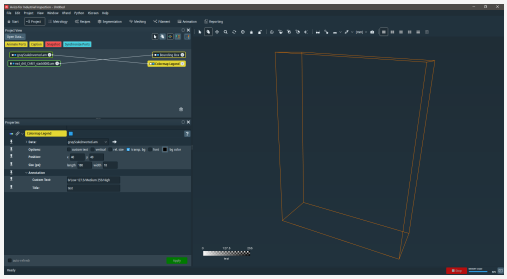
Acceptance criteria:

I am a Volumetric Viewer user

I do a click on a legend button on a specific view

I check that the legend is displayed and:

- have a min / max values
- have the correct colormap displayed
- can be transparent
- is on top / middle of the view



6.1.6 Distance 2D

The measuring tools should be able to measure a distance between two points in a 2D View.

Acceptance criteria:

I am a Volumetric Viewer user

I open a file, and select the ruler tool in the tool bar

I check that the I can measure accurately a distance between two points in a 2D view.

6.1.7 Angles 2D

The measuring tools should be able to measure an angle between two vectors in a 2D View.

Acceptance criteria:

I am a Volumetric Viewer user

I open a file, and select the ruler tool in the tool bar

I check that the I can measure accurately a distance between two points in a 2D view

6.2 Émergence du GUI2D system

6.2.1 Point de départ

Afin de modifier rapidement l'interface suivant les retours d'utilisateurs dans le cas du ScaleAxis et de la la Colormap Legend, il nous fallait un outil qui soit assez flexible et personnalisable à utiliser. Cependant au départ, il n'existait pas de telle solution au sein du projet.

Il y avait plusieurs choix :

- faire du côté Front-end l'affichage et envoyer toutes les informations nécessaires au client,
- implémenter côté Back-end une solution propre au Volumetric Viewer avec OpenInventor (qui n'a pas de système 2D).
- rajouter dans le Back-end une solution third party type ImGui ou autre pour rajouter des rendus d'interface,
- créer une solution générique réutilisable dans OpenInventor.

Chaque solution a ses avantages et ses inconvénients, mais au final celle qui fut choisie est d'implémenter un nouveau système générique faisant office de surcouche dans OpenInventor, le Gui2D system. Cela permettait de proposer ces interfaces à d'autres applications utilisant OpenInventor comme moteur de rendu 3D, de ne pas communiquer trop d'informations par le réseau pour des éléments spécifiques (ColorMap par exemple), et de pouvoir faire des rendus de screenshots à n'importe quelle résolution (et non pas d'après l'interface du client, les tablettes mobiles étant supportées).

6.2.2 Concept

Le Graphical User Interface 2D System (Gui2D) a pour but de faciliter l'affichage d'informations 2D nativement dans le moteur de rendu OpenInventor. C'est un système qui n'existait pas jusqu'à présent.

Cependant ce système n'a pas pour but de remplacer des systèmes d'interfaces comme Qt[7] ou DearIMGUI[12] par exemple, mais il peut être très utile pour du prototypage rapide et afficher des informations spécifiques à la scène OpenInventor sans s'encombrer de callback ou de système de synchronisation entre un système d'interface et OpenInventor. Il se base sur des solutions d'interface comme Unity[19] ou Unreal Engine[1], ainsi que QT[7].

Dans le contexte du Volumetric Viewer, les composants impactés étaient le ScaleAxis et le Colormap Legend. Ce système fut l'objet d'une présentation, disponible en annexe 9.6.

6.2.3 Fonctionnalités proposées

- Positionnement dynamique et responsif avec des ancres.
- Commutatibilité entre placement normalisé relatif (pourcentage de viewport) et fixé (en pixel), toujours par rapport à l'ancre et au parent.
- Pas de transformation de caméra et système de coordonnées fixe en 2D (rendre inaccessible ces composants dans la hiérarchie OpenInventor).
- Afficher au-dessus de la scène (même principe que les SoAnnotation natif OpenInventor)
- Différentes stratégies de taille (fixe en pixel, *minimum-maximum*[0], relatif au parent).
- *Des layouts automatiques contrôlant la position et la taille des enfants avec espacements et marges.*[0]
- *Étendre les fonctionnalités des textes avec des taille de police d'écriture automatique suivant le contenant et les contrôles associés (taille minimale, overflow de texte, wrapping...).*[0]

[0]En italique les fonctionnalités non implémentées au final durant ce stage.

6.3 Planning

Il n'y a pas eu au départ de planning défini, ce n'est qu'après la restructuration qu'il fut mis en place à partir du 22 Juin 2021. Vous pourrez retrouver les photos de celui-ci en annexe (9.2). A cette date, le picking était en validation et le ScaleAxis avait commencé. C'était aussi la période où le Gui2D System devenait concret.

En voici le résumé :

Sprint 0 (28/06)	Sprint 1 (12/07)	Sprint 2(26/07)	Sprint 3(09/08)	*Sprint 4 (23/09)*
ScaleAxis2D Unit - Bug RemoteViz - Gui2D system anchors	Distance mesurement- Angle mesurement	ColorMap Legend - Mesurement Help + User Guide	Gui2D System Position Strategies	Gui2D System Textes basics - Upgrade OpenInventor 10.10

Pour rappel, chaque Sprint correspond à 2 semaines (voire 4.2).

7 Implémentation

7.1 Choix techniques, évolutions et conséquences

7.1.1 Outils et bibliothèques

7.1.1.1 OpenInventor

OpenInventor Toolkit[17] est un moteur de rendu 3D propriétaire pour les professionnels, dérivé de OpenInventor à la base OpenSource, et optimisé pour trois domaines principaux : Medical & Dental, Oil & Gas et Engineering. Son but est de donner une couche d'abstraction à la bibliothèque OpenGl et de permettre de gérer des scènes 3D. On y retrouve plusieurs modules et ceux que j'ai le plus utilisés sont VolumeViz, permettant l'affichage de volume, par opposition aux données surfaciques affichées par l'extension MeshViz, le module Medical spécialisé pour des rendus de fichier Dicom et en analyse médicale, et enfin RemoteViz permettant un affichage interactif d'Open Inventor sur navigateur en transmettant au client un flux vidéo.

L'architecture des scènes OpenInventor se base sur un graph, souvent en arbre, où l'on parcourt les Nodes de la scène en profondeur pour appliquer des effets tels que des matériaux, des shapes, avec un système d'Action (le rendu en étant un type d'action particulière). Lors du parcours de ces actions, les Nodes peuvent influer sur son état, ou State, qui contient des Elements que les futurs Nodes parcourus pourront consulter.

Les Nodes sont composés de Fields, qui abstraient son état et permettent de maintenir le cache de celui-ci, ainsi que la sérialisation et désérialisation des données.

Une liste plus exhaustive des nodes et composants est disponible dans la documentation de OpenInventor [16].

7.1.1.2 Protobuf

Protobuf, ou Protocol buffers[5] est un format de sérialisation qui permet de sérialiser/désérialiser des données structurées avec un langage de description d'interface développée par Google. Il existe plusieurs langages d'implémentation comme à l'origine C++, Java et Python disponible sous une licence libre, et ce format a pour objectif la simplicité et la performance en étant plus léger et plus rapide que le XML.

Dans le volumetric viewer, nous utilisons les implémentations C++ et TypeScript de ce format, afin d'échanger des messages entre le Back-end et le Front-end.

Les messages sont sérialisés sous un format binaire compact, où les champs sont indexés, mais non nommés, ce qui permet une relative rétrocompatibilité entre les versions. On écrit donc suivant la syntaxe Protobuf la description des messages et on génère le code dans les deux versions Front-end et Back-end (C++ et Typescript), que l'on publie ensuite dans sur le serveur Artifactory (serveur de packages qu'on utilise pour le Front-end).

7.1.1.3 Angular

Angular[4] est un framework Web en TypeScript gratuit et open source développé par Google permettant de faire des applications web dites "singlepage", couplé avec un serveur Node.js intégré.

Sa principale caractéristique est que les projets Angular suivent une architecture Modèle-Vue-Contrôleur et hiérarchique en composants, avec souvent pour chaque composant un fichier TypeScript pour la logique, un HTML pour l'affichage, un fichier TypeScript de test, et un fichier SCSS de style, ce qui permet d'être modulaire et de collaborer plus facilement.

7.1.1.4 IOLink/IOFormat

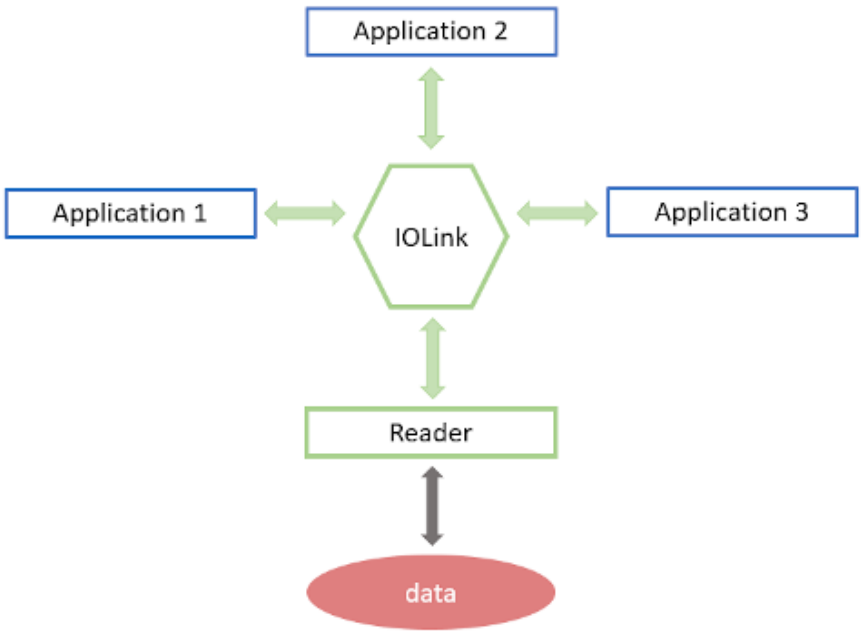


Figure 14: Principe de IOLink.

IoLink est un DataModel qui permet de gérer un problème courant lors de l'utilisation de plusieurs applications qui doivent accéder/échanger/manager les mêmes données. Une implémentation naïve serait de faire un reader pour chaque application, mais cela voudrait dire maintenir autant de reader qu'il y a d'application, et c'est là qu'intervient IoLink (fig.14) et IOFormat, qui l'implémente et le complète en créant des "vues" de datas encodés complexes (fichiers, streams, réseau, etc...).

Hormis la couche existante de chargement des données du Volumetric Viewer, IoLink/IoFormat ont été utilisés notamment pour créer des datasets de tests pour le ScaleAxis et les Mesures 2D.

7.1.1.5 Cypress

Cypress[2] est un outil d'automatisation de test IHM (Interface graphique) open source qui permet de faire des tests dits "End-To-End" vérifiant la bonne intégration des différents composants. Le framework permet notamment d'automatiser et de simuler des événements clavier/souris, et bien sûr de faire des assertions et des tests sur le contenu de l'interface du Volumetric Viewer.

Il fut incontournable lors du développement vu qu'à part manuellement, il y a que peu de possibilités (voir aucune) pour tester automatiquement des fonctionnalités comme le picking ou les mesures.

7.1.1.6 PlantUML

PlantUML[13] est un outil permettant de créer des diagrammes UML à partir de textes. Le principal avantage de cet outil est qu'il est naturellement intégré dans les fichiers Markdown Gitlab et on peut visualiser les diagrammes depuis le repository.

Dans le projet, il est utilisé pour de la documentation avec des diagrammes de classes et de flows, et m'a permis aussi de présenter les composants et l'architecture du Gui2D system.

7.1.1.7 Linters

Il y a plusieurs linters dans projet comme Clang Format[6] pour le Back-end ou eslint[20] pour le Front-end. J'ai utilisé également les outils de formatage de mon IDE VSCode. Je détaille les linters dans une partie à part plutôt que dans 4.3 car j'ai découvert que malgré le fait que l'on puisse s'adapter à plusieurs styles de code et travailler sans problème, ils permettent surtout de clarifier les merges requests, cachant les modifications dues au style de codage. Jusqu'à alors je pensais que c'était plus une question d'esthétique et de confort propre à chacun plutôt que de "réelle" utilité en équipe.

7.1.1.8 DearImgui

DearImGui[12] est un logiciel d'interface graphique en C++ open source, portable et auto-contenu qui s'adapte à n'importe quel moteur de rendu 3D afin de faire des outils avec interfaces flottantes (débuguage, visualisation, UI, etc).

Cet outil fut d'ailleurs utilisé grâce à un projet interne couplé avec Qt[7] et OpenInventor[17] afin de pouvoir modifier en runtime le placement et les paramètres des composants du Gui2D system. Il est d'ailleurs très simple d'utilisation et a permis de maintenir la démo très facilement lors du rajout de composants.

7.1.1.9 PaperJS

Paper.js[8] est un framework open source de dessin vectoriel sur les Canvas en HTML5. Il a permis avec la version TypeScript d'implémenter toute la partie visualisation et calcul lors d'interactions dans les fonctionnalités de mesures sur le Front-end.

7.1.1.10 Conan

Conan[10] un outil open source décentralisé de gestion de packages binaire C++, qui fut utilisé durant mon développement pour gérer les versions de Protobuf[5] et IOFormat/IoLink, mais aussi certaines bibliothèques comme Qt[7] lorsqu'il a fallu tester le Gui2D System.

7.1.2 Picking

Le Picking fut implémenté grâce à OpenInventor et au SoRayPickAction qui était compatible avec les vues 2D (OrthoSlice) et 3D (volumique), et qui est optimisé par le GPU dans le cas des vues 3D. Cette fonctionnalité m'a permis de découvrir le projet, mais aussi de faire la base du système d'outils globaux implémenté dans le Viewer, qui était uniquement pour contenir les vues au départ. Il y a eu par contre des problèmes inattendus en raison de limitations liées au picking mélangé aux fonctionnalités de clipping de volume 3D qui n'était pas encore un cas d'utilisation pris en compte. Il a été donc décidé de ne pas activer le picking dans les vues 3D (7.4.2).

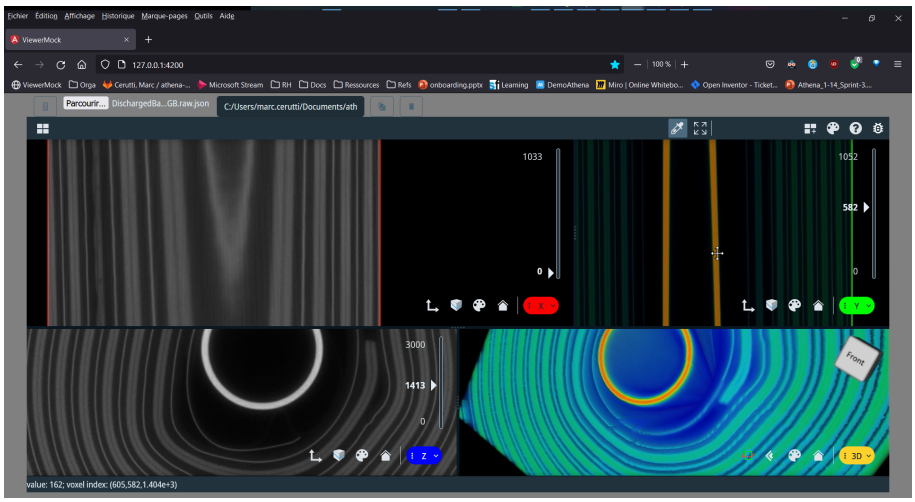


Figure 15: Picking

7.1.3 Gui2D system

7.1.3.1 Méthode de travail

Afin de pouvoir développer plus rapidement ce système, et comme son but est de faire partie de OpenInventor, un nouveau repository Gitlab a été créé pour l'occasion. Utilisant un projet de recherche interne se servant de Qt et DearImgui, toute l'implémentation s'est faite dans un mock (fig.16) et a permis de pouvoir toucher facilement aux différents paramètres des composants pour les tester, sans même avoir toute l'architecture OpenInventor.

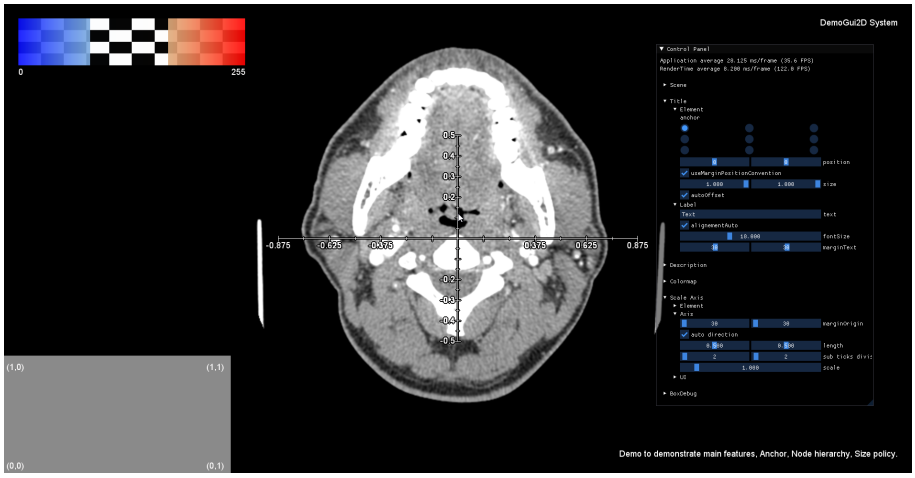


Figure 16: Gui2D - Mock

7.1.3.2 Architecture

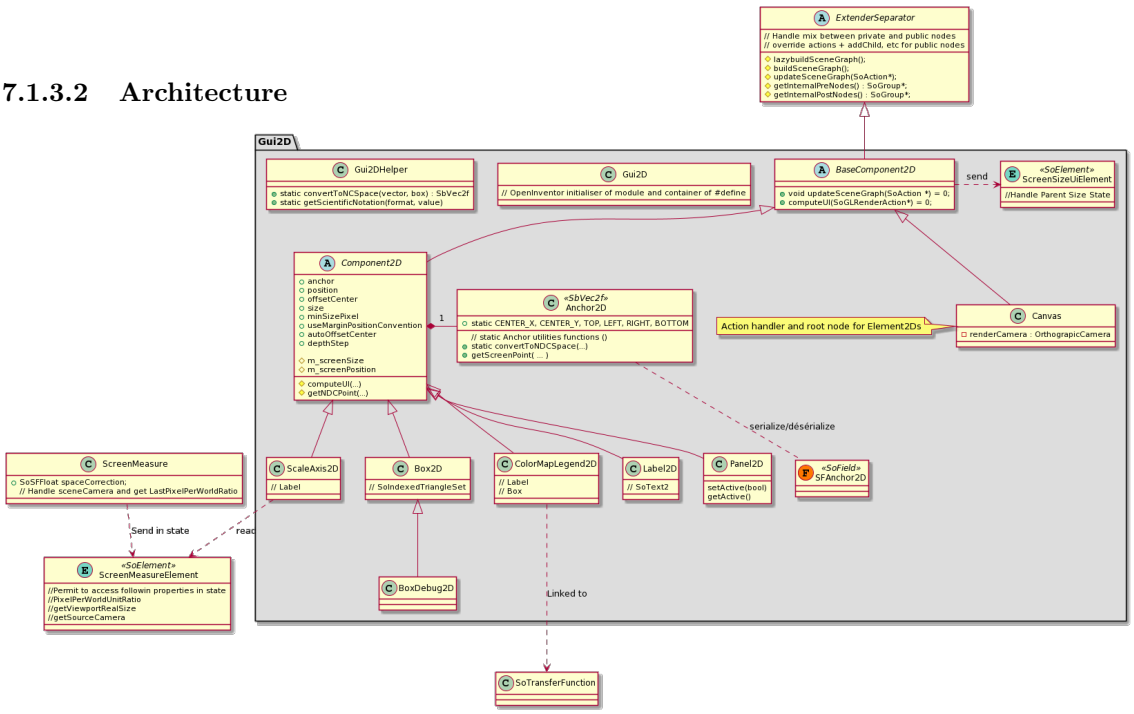


Figure 17: Architecture du Gui2D System.

La plus grande partie du travail a été d'adapter les idées du système à la structure et conventions de OpenInventor [16], notamment pour qu'à terme le système puisse être intégré au sein de l'outil nativement, et pour pouvoir optimiser et préserver le cache des composants. Cela a permis aussi de mettre en place une vue simplifiée pour les utilisateurs de ce système (fig18). Cependant pour ceux qui voudront étendre les composants, il vaudra mieux se référer à la hiérarchie complète utilisée, qui est disponible en annexe 9.3.

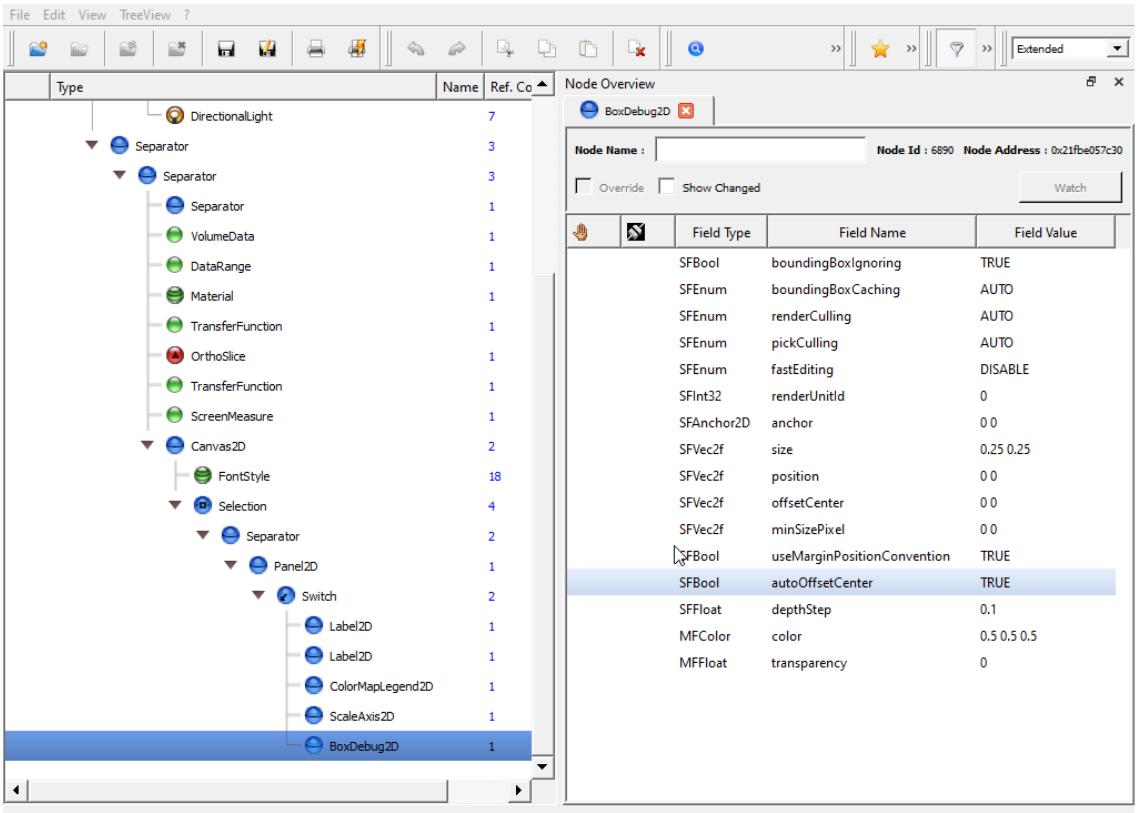


Figure 18: Gui2D - Hiérarchie simplifiée des éléments affichés via IvTune.

La construction des interfaces suit la logique de OpenInventor de base en arbre où les éléments, Component2D, sont représentés abstraitement par des boîtes (position + taille). Le Canvas2D est par contre un Node spécial, du fait qu'il est à la racine du sous graph OpenInventor, responsable de gérer la caméra qui influe sur le système de coordonnées. Le système adopté fut celui du Viewport, avec en bas à gauche (0,0), et en haut à droite (1,1). Cela permet ensuite pour les composants d'avoir un système simple permettant d'utiliser des coordonnées normalisées, ou en pixel si besoin qui sera converti.

7.1.3.3 Conventions & Positions

Les principes de positionnement ont été largement inspiré de logiciels tel que Unreal[1], Unity3D[19] ou encore QT[7], voir le CSS, comme montré sur les figures 19 et 20. Les paramètres prenant des pourcentages peuvent récursivement se référer non pas à l'écran, mais à leur parent. L'implémentation du système s'est basée principalement sur les SoSeparator et les SoTranslation qui ont permis de créer les translations pour chaque composant qui se cumulent entre parent et enfant.

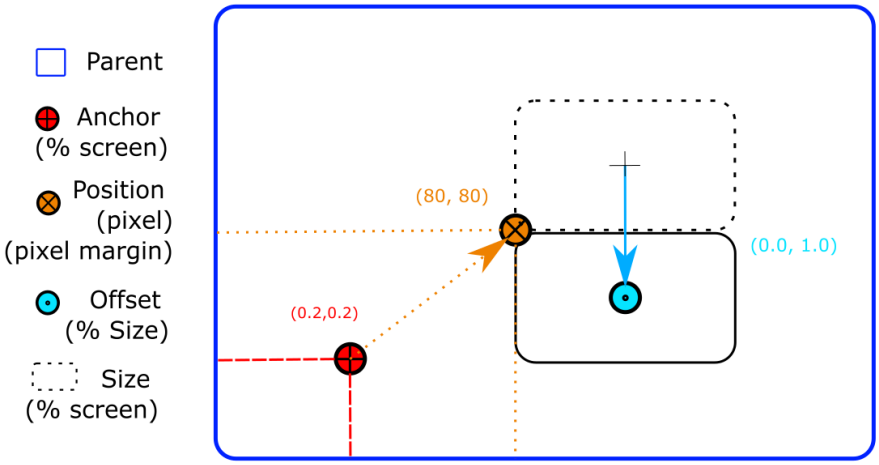


Figure 19: Gui2D - Exemple simple de positionnement position d'élément par rapport à l'écran

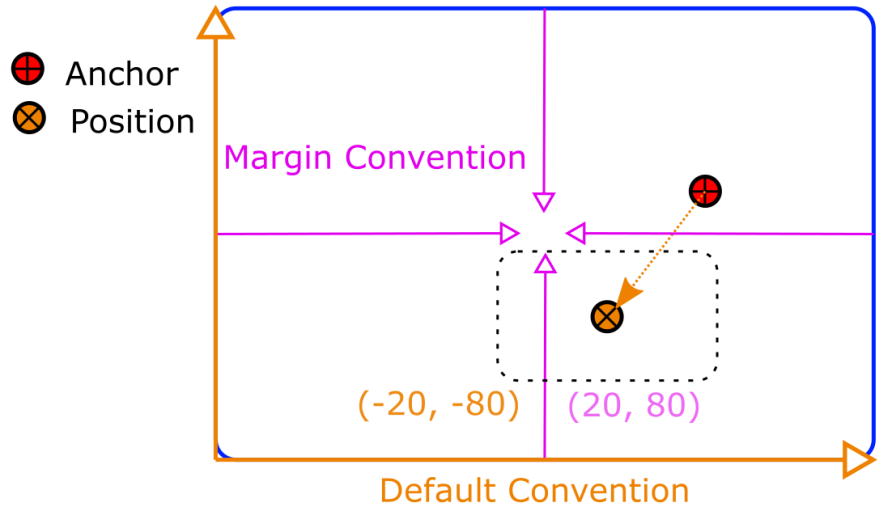


Figure 20: Gui2D - Margin convention

7.1.4 Scale axis

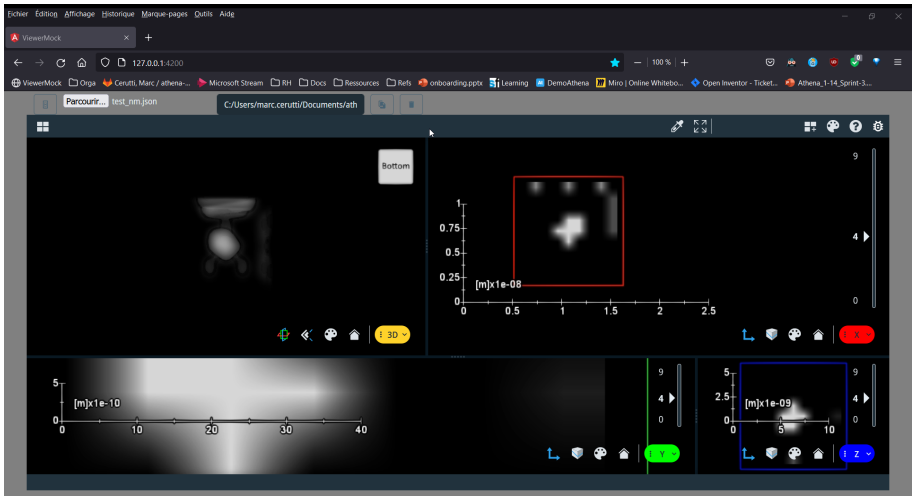


Figure 21: Fonctionnalité du ScaleAxis.

Le POC (Proof of Concept) du composant a été fait avant le Gui2D system et se base sur le code des ScalesBars de Avizo (voir meeting Scale 2D **??**). Il a été nettoyé, rendu compatible OpenInventor, et amélioré tout au long du développement du Gui2D system. L'implémentation final du ScaleAxis (fig.21) est faites en deux noeuds, un reprenant la logique des Component2D du Gui2D Sytem et l'affichage, le ScaleAxis, et un pour les calculs simples mais répétitifs, le ScreenMeasure, réutilisé pour les mesures 2D (7.1.5). On peut adapter le composant selon cinq configurations différentes (fig.22), changeant le sens des axes, l'origine, ainsi que d'autres paramètres permettant de modifier très rapidement le composant.

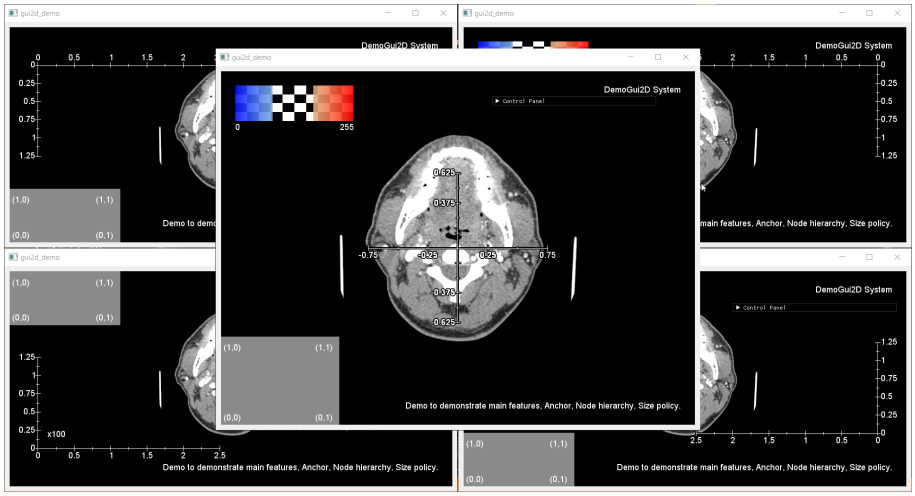


Figure 22: Différentes configurations du ScaleAxis

Il a fallu pour l'occasion changer l'architecture existante sur le Volumetric Viewer afin de pouvoir accéder aux metadatas, qui n'étaient disponibles que dans la partie de chargement des fichiers et le RawLDMVolumeReaderIoLink, les utilisant pour calculer la bounding box. Un utilitaire avec IoLink/IoFormat fut nécessaire afin de construire des datasets pour tester cette fonctionnalité, à partir d'images et de paramètres comme la taille réel voulu. C'est d'ailleurs à ce moment là, en utilisant ces datasets de l'ordre du nanomètre, que l'on a pu se rendre compte d'un bug sur les bounding box (7.2), lié à des limitations sur OpenInventor.

7.1.5 Mesures 2D

Les mesures 2D (fig.23 et 24), comme ne faisant pas partie des légendes comme le ScaleAxis ou les Colormap, il fut décidé de les implémenter sur le Front-end via le framework PaperJS (7.1.1.9), disponible aussi en TypeScript. Les calculs étant déjà fait côté Back-end, il a suffi d'envoyer les données côté Front-end, et de dessiner les vecteurs lors des interactions.

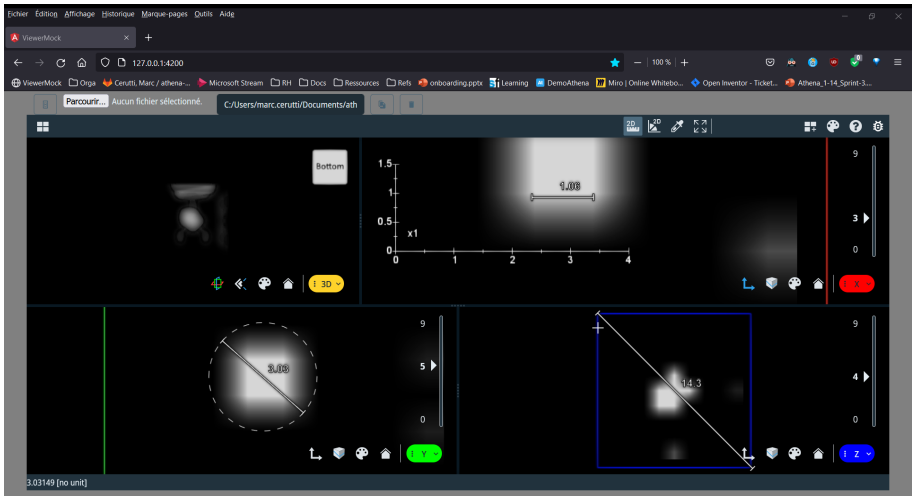


Figure 23: Fonctionnalités de mesure de distance

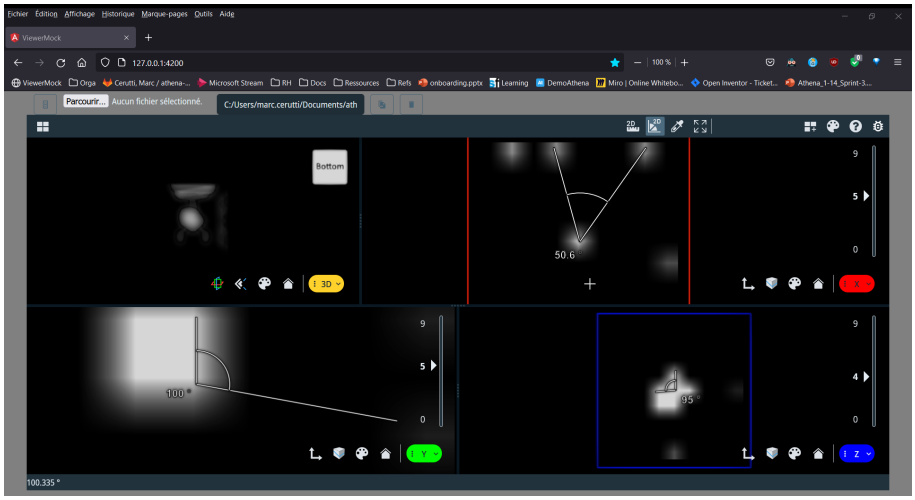


Figure 24: Fonctionnalités de mesure d'angle

La mise en place de cette fonctionnalité a été rendue possible par un composant Angular, DrawAreaComponent, qui s'occupe de la liaison avec les autres composants et l'initialisation des callbacks/liaisons nécessaires, et par une classe abstraite, DrawAreaElement (listing 1), avec une implémentation pour les distances et une pour les angles. DrawAreaElement permet de ce fait ne pas être dépendant de PaperJs, et d'implémenter par "mode d'utilisation" une logique d'intéraction. Les callbacks d'input utilisateur peuvent aussi prévenir la propagation des évènements à RemoteViz en cas de besoin.

```
1
2  //DrawAreaElement
3
4  //...
5
6  export abstract class DrawAreaElement {
7
8    public onMouseDownCallback: ((arg: MouseEvent) => boolean) | undefined;
9    public onMouseUpCallback: ((arg: MouseEvent) => boolean) | undefined;
10   public onMouseMoveCallback: ((arg: MouseEvent) => boolean) | undefined;
11
12   public onTouchStartCallback: ((arg: TouchEvent) => boolean) | undefined;
13   public onTouchEndCallback: ((arg: TouchEvent) => boolean) | undefined;
14   public onTouchMoveCallback: ((arg: TouchEvent) => boolean) | undefined;
15
16   protected readonly element: ElementRef;
17   protected camera: Camera | undefined;
18
19   constructor(element: ElementRef, camera?: Camera) {
20     this.element = element;
21     this.camera = camera;
22   }
23
24   abstract clear(): void;
25
26   abstract resize(size: [number, number]): void;
27
28   // Provide information like pixelPerWorldRatio, unit, etc.
29   abstract updateCamera(camera: Camera): void;
30 }
```

Listing 1: DrawAreaElement contract, implementation with PaperJS permit to display distance and angle.

7.1.6 Colormap legend

Les légendes de Colormap sont disponibles dans le Gui2D System, mais se sont retrouvées en plus basse priorité que les mesures, demandées par des collaborateurs comme l'équipe MicroP. En effet, il y a déjà des moyens de visualiser les colormaps avec des panels, et leur affichage sur les vues serait pour prendre des screenshots de haute résolution à l'avenir. La fonctionnalité a donc de fortes chances de commencer après la remise du rapport et ne sera donc pas abordée.

7.2 Tests & bugs

Les principaux tests automatiques que j'ai rajoutés sont faits grâce à Cypress (7.1.1.5), vous pourrez d'ailleurs retrouver en 9.5 le détails et cas couverts de ceux-ci.

J'ai pu participer aussi à la découverte / résolution de certains bugs. Les principaux ont été notamment lors des tests entre clipping des volumes et picking, ce qui a permis à OpenInventor d'améliorer le fonctionnement entre les deux. Pour l'occasion, j'ai fait un reproducteur pour tester la fonctionnalité prévenant le picking 3D (7.4.2). Un bug aussi dû à l'utilisation de dataset de l'ordre de 10e-9 a permis de mettre en lumière des limitations côté OpenInventor, qui n'avait pas été prévu par le Volumetric Viewer. Cela a permis de prendre en compte l'ordre de grandeur de la taille réelle du dataset lors de la reconstruction des données, ce qui apporte plus de stabilité. Et enfin il y eut aussi un bug lié à remoteviz suite à une division par zéro du fait de rendus trop rapides lors de cas extrêmes de petites fenêtres et il fut corrigé. Bien sûr, il y eut de nombreux bugs que j'ai insérés et corrigés, par exemple un concernant le chargement de police d'écriture incorrect du aux différences d'environnement entre Windows (mon espace de travail), et la machine cible Linux.

Lors de ce problème j'ai eu aussi l'occasion de tester la mémoire et les performances sous Linux du Gui2D System, ce qui a permis de mettre en lumière des fuites mémoire à cause de l'outil OpenInventorStudio, grâce au visualiseur Valgrind Massif. Certains problèmes mis en relief sont toujours d'actualité et seront abordés en 7.4. En résumé, le Gui2D System a une différence d'environ 34 Mb en RAM (avec/sans tous les composants de la démo)(9.4), et tourne à 50 FPS en moyenne sur la démo.

Tests de performances

Configuration :
Windows 10 x64
GPU NVIDIA Quadro T1000
CPU Intel Core I7
32 Go RAM
1920x1080 Resolution Full screen window

Performances cache assured (without text):
57 FPS Application / 528.6 FPS render (mouve mouse only)
55 /400 FPS with continuous interaction.
No redraw without mouse moving.

Performances cache (with text + some security, not working on Volumetric Viewer):
59/500 FPS (mouse move only)
50/89 FPS with continuous integration
No redraw without mouse moving.

Performances no auto-cache (with text):
50/140 FPS (mouse move only)
49/88 FPS with continuous integration
No redraw without mouse moving.

7.3 Planning effectif

Au final selon le planning réalisé (fig.25), la plupart des objectifs en matière de planning pour l'implémentation des fonctionnalités furent tenus, et les estimations ont bien résisté face à des tâches non prévues au début du stage comme le Gui2D system. Cette partie d'ailleurs fut la moins juste en terme d'estimation due à la complexité d'adapter le POC du système en "vrais" noeuds OpenInventor. On peut analyser aussi que le Picking a pris beaucoup de temps avec des retouches, mais ce qui est en soi compréhensible si on enlève le temps d'adaptation aux outils et au projet. Les légendes de Colormaps n'ont aussi pas pu être traité avant le rapport.

Cependant, une chose que j'ai appris rapidement, c'est qu'il est très aisé de faire les prototypes de fonctionnalité complètement fonctionnelle, et c'est les reviews qui prennent le plus de temps, avec les corrections de bugs, les tests, ainsi que les réunions et les échanges avec d'autres équipes comme OpenInventor. Ainsi le temps de "pur" développement est finalement de 20 à 30% d'une fonctionnalité selon l'expérience que j'en tire.

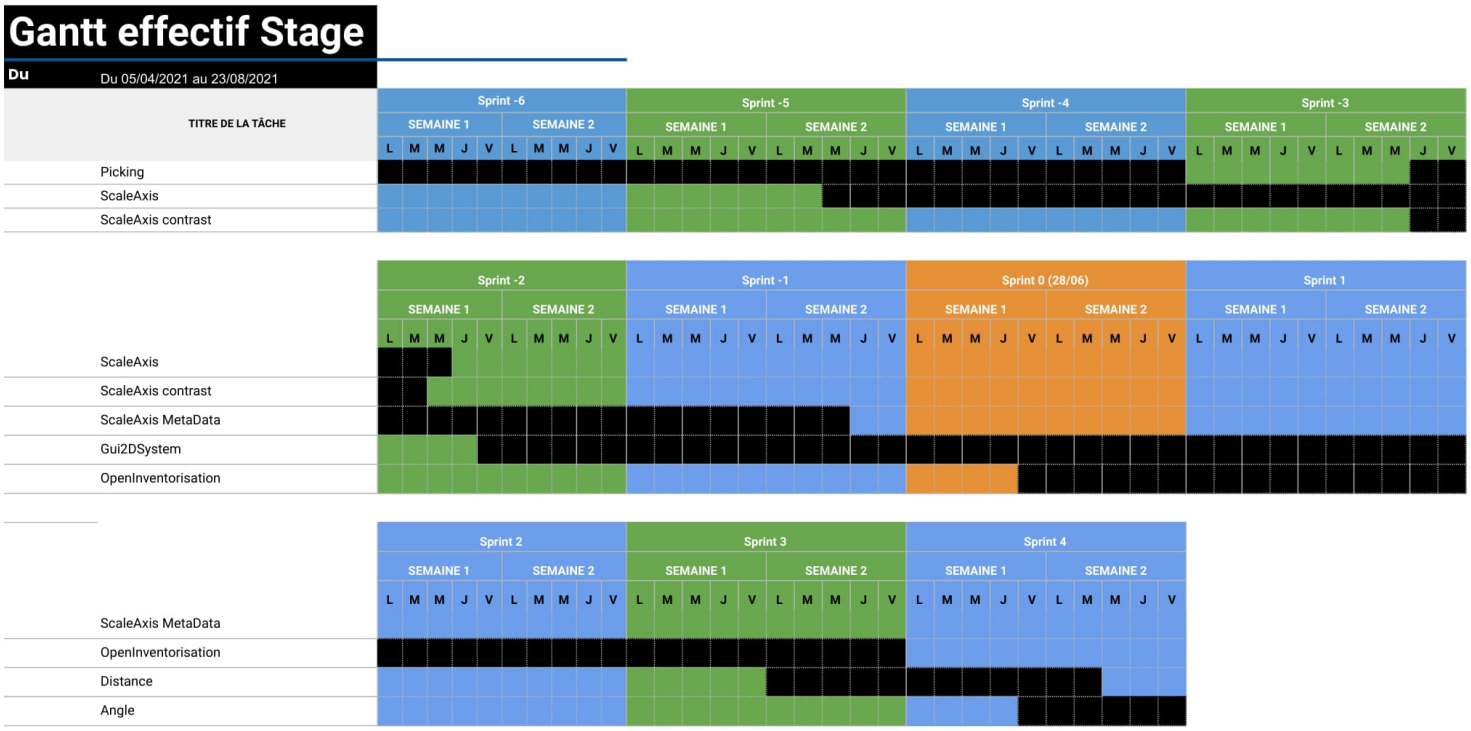


Figure 25: Gantt effectif du projet.

7.4 Améliorations possibles

7.4.1 GUI2D System

7.4.1.1 Réanalyse du besoin

Suite à la création des outils du ScaleAxis et Colormap Legend via Openventor[17] côté Back-end, et celles de mesures grâce à la bibliothèque PaperJs[8] côté Front-end, il est intéressant de faire un retour sur ces deux méthodes afin de créer des nouvelles interfaces pour l'utilisateur. Le bilan est que malgré le fait que OpenInventor soit un outil puissant de l'affichage en 3D, il a fallu beaucoup de temps et de ressources (notamment d'échanges avec l'équipe OpenInventor) pour réaliser la solution du Gui2D system. Même si le but est de servir dans d'autres applications que le Volumetric Viewer, de permettre de prendre des screenshots hautes résolutions et que c'est un tout nouveau système, OpenInventor est très peu adapté à ce genre de solution, surtout si on rajoute des interactions. En comparaison, il a fallu le quart de temps pour faire les mesures grâce à PaperJS du côté Front-end, et surtout avec moins de risque de bugs ultérieur (voire 7.4.1.5). Au final, il aurait été très facile d'envoyer les informations coté Front-end en étendant le protocole Protobuf. Mais seul le long terme et le fait qu'il faudra faire des screenshots hautes résolutions (ou pas), pourront décider de la meilleure.

7.4.1.2 Problèmes de rendus

Il y a eu plusieurs problèmes car OpenInventor est un moteur de rendu 3D très spécialisé. Les éléments (et surtout les textes) sont sujets à des problèmes de profondeur qui ont été contournés en changeant la fonction de test de Depth à "Always". Cela peut cependant devenir problématique à l'avenir si on veut faire des interfaces plus complexes. Cela nécessiterait dans un même temps de faire un mode de rendu spécial comme les SoAnnotation, ce qui permettrait de remplacer le système de HaloHighlightsRenderAction et de SoSelection pour afficher des outlines et optimiser les performances.

7.4.1.3 Textes & Outline

En rapport avec la section précédente, il n'existe pas à l'heure actuelle de solution dans OpenInventor pour faire des contours autour du texte (outlines). L'alternative à cette limitation fut d'utiliser un système à l'origine pour mettre en surbrillance des éléments sélectionnés, les HaloHighlightsRenderAction. Par contre cette solution réalise une nouvelle passe de rendu avec le Depth Buffer qui a été écrit par les éléments sélectionnés. C'est une solution plus coûteuse qu'une solution native, car elle dépend de la résolution de rendu et est très peu flexible pour faire les contours des éléments de l'interface. Implémenter une solution native nécessiterait une réfactorisation, à l'heure actuelle coûteuse dans OpenInventor, mais tout système d'interface futur devrait résoudre ce problème.

7.4.1.4 Interactions avancées

Le Gui2D System a un gros point faible aussi, car il ne permet pas de faire d'interactions avancées comme des boutons, sliders, et autres types d'interfaces. C'était voulu, mais néanmoins le besoin principal de l'équipe OpenInventor concernait surtout cette problématique. Elle mériterait que l'on se penche plus sur les possibilités du Gui2D System pour la résoudre.

7.4.1.5 Problèmes de cache

La solution a encore quelques points à améliorer à cause de la complexité d'OpenInventor pour créer des noeuds optimisés et interactifs avec son système de cache. Même si elle est assez performante pour le Volumetric Viewer, il reste des points obscurs où le cache est impossible à utiliser en raison de ressources mystérieusement désallouées entre deux rendus (notamment les textures sur les textes). Il y a d'ailleurs une dette technique encore à l'heure actuelle sur les textes dans cette même problématique dans OpenInventor.

7.4.2 Picking 3D

Comme précisé précédemment durant le picking, il ne fonctionne pas lorsqu'on veut picker à l'intérieur d'un volume devenu visible par le clipping. Ce cas d'utilisation n'avait pas encore été abordé, et a permis de fournir à l'équipe OpenInventor un reproducer pour cette fonctionnalité. Elle est envisagée pour la nouvelle version d'OpenInventor d'ici Septembre 2021.

7.5 Concepts appris ou approfondis importants

7.5.1 User stories(US) et rôles

Grâce à une formation de Nicolas Pascaud et à la restructuration, j'ai pu voir énormément de choses comme si on commençait un projet. Par exemple, la Definition Of Done (DoD), la Definition Of Ready (DoR), les différents rôles et responsabilités au sein d'une équipe, comment rédiger une User Storie (US). Ces points sont importants à définir, et c'est surtout le "pourquoi?" qui a retenu mon attention, et quelques concepts m'ont marqué : le principe INVEST (Indépendance, Négociable, Valeur, Estimable, Sufisamment petite, Testable) pour définir les US, le Planning poker avec la suite fibonacci pour les estimer, le principe MoSCoW (Must Have, Should Have, Could Have, Wont't Have) pour le tri des tâches du Backlog et enfin les critères d'acceptation (Given-When-Then ou I am, I do, I check) pour la définition des tests.

7.5.2 Tests& bugs

Ce fut l'un de mes premiers projets où j'ai pu faire des tests qui ont eu de réelles conséquences sur la qualité du code. La rédaction et la logique derrière ceux-ci sont donc une expérience inestimable, et j'ai d'ailleurs beaucoup apprécié l'utilisation de Cypress (7.1.1.5) qui permet de les faire facilement.

7.5.3 Git review, merge requests et rebase

Ce projet a significativement réduit le temps passé et mes erreurs commises durant ces opérations, et j'ai appris comment utiliser ces outils au bon moment, pour ne pas tomber dans les extrêmes (un seul commit pour tout versus plusieurs petits commits ne contenant rien). J'avais peu utiliser jusqu'à présent les fonctionnalités de squash et rebase sur Git, mais je pense maintenant que les connaître un minimum est essentiel à long terme sur un projet.

7.5.4 Documentation

Je réalisais déjà de la documentation efficace pour communiquer, mais j'ai approfondi comment la maintenir dans le temps, avec des outils comme PlantUML pour les diagrammes, qui jusqu'à présent étaient trop lourds ou compliqués à mettre à jour dans mes anciens projets.

7.5.5 Développement & Architectures

J'ai pu voir différents logiciels et architectures durant ce stage qui furent très intéressants. J'ai pu compléter mes connaissances avec la logique et les outils du développement Front-end, les méthodes de travail sous Windows, les variables d'environnements, ainsi qu'avec les concepts et décisions derrière l'architecture des logiciels que j'ai utilisés.

8 Bilan

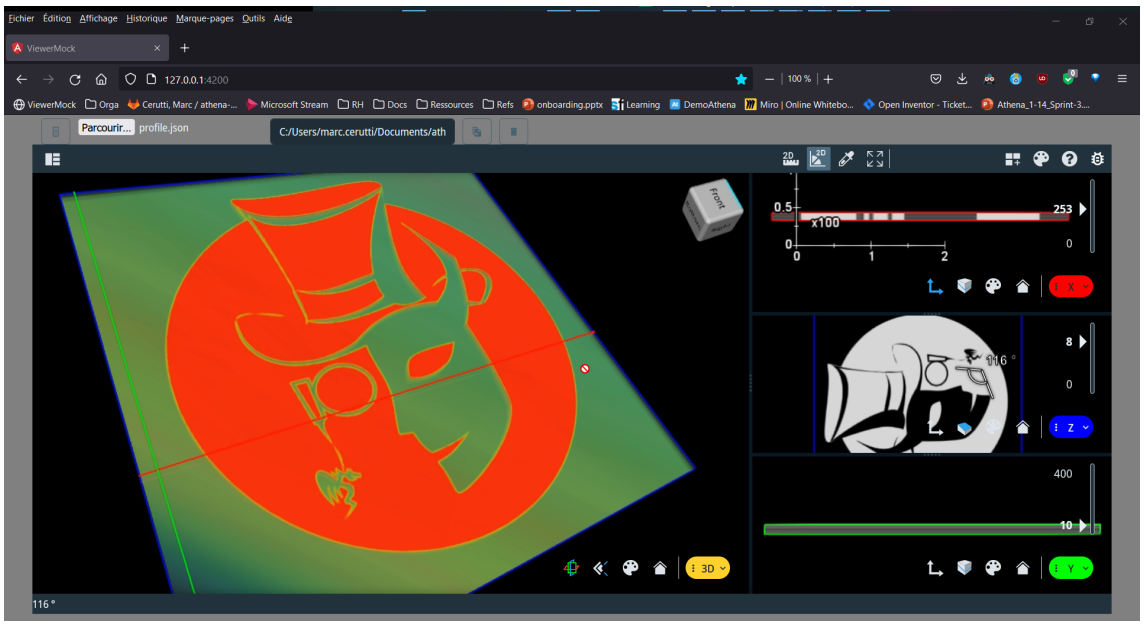
Les objectifs pédagogiques ont été complètement atteints.

J'ai approfondi mes connaissances sur les procédés des méthodes agiles, et sur la mise en place de procédures de tests. Les compétences que j'ai acquis sur la rédaction de User Stories et de bugs m'ont permis d'être plus clair sur l'explication de fonctionnalités et sur leur estimation. Mes connaissances théoriques en synthèse 3D ont été appliquées avec OpenInventor, que j'ai appris à maîtriser avec d'autres technologies. Les processus industriel Git que j'ai approfondi en les mettant en application m'ont permis d'être plus efficace pour la gestion de versions et collaborer sur du code. La conception du Gui2D system fut une excellente expérience, et mon regret est de ne pas pouvoir le continuer jusqu'à son intégration dans OpenInventor, compte-tenu de mon départ à la fin du stage. Les échanges et la collaboration avec plusieurs services et personnes m'ont permis de m'enrichir, de développer mes compétences techniques et de communication en entreprise. Les méthodes acquises de développement Front-end m'ont permis de gérer les contraintes et la qualité du code pour les interfaces web.

J'ai rencontré des difficultés, notamment lorsqu'il a fallu normaliser le Gui2D system, mais grâce à un bon encadrement et des directives claires, j'ai pu accomplir mes objectifs.

J'aurais par contre souhaité approfondir davantage les bases des technologies Front-end, car cela m'aurait facilité la compréhension du code et l'implémentation des solutions. La façon de découper les User Stories que j'ai utilisé peut être encore améliorée car elles étaient trop importantes pour un Sprint, ce qui n'est pas souhaitable en agilité.

En conclusion, ce fut une excellente expérience professionnel qui pourra toujours me servir à l'avenir. Je souhaite une bonne continuation au membres du projet Athena, et j'espère que mon travail servira aux futurs utilisateurs de la plateforme.



9 Annexes

List of Figures

1 Portfolio des différentes solutions de FEI SAS. 5
2 Dates clés de l'évolution du pôle de Bordeaux, de C5G à nos jours. 6
3 Principe du coeur de métier d'Athena. 7
4 Cas d'utilisation de la plateforme Athena. 8
5 L'écosystème du projet Athena après restructuration avec les collaborateurs. . . . 8
6 Interface de base du Volumetric Viewer. 9
7 Modèle client/serveur Volumetric Viewer Plugin. 12
8 Screenshot du ViewerMock. 13
9 Back-end Volumetric Viewer. 14
10 Graph des scènes OpenInventor, 3D à gauche, 2D à droite. 14
11 Pipeline Gitlab principal. 15
12 TestRail[18], regroupant les cas de tests et contenant les rapports des pipelines effectués. 16
13 Exemple de bug résolu sur le Picking. 16
14 Principe de IOLink. 23
15 Picking 25
16 Gui2D - Mock 25
17 Architecture du Gui2D System. 26
18 Gui2D - Hiérarchie simplifiée des éléments affichés via IvTune. 26
19 Gui2D - Exemple simple de positionnement position d'élément par rapport à l'écran 27
20 Gui2D - Margin convention 27
21 Fonctionnalité du ScaleAxis. 28
22 Différentes configurations du ScaleAxis 28
23 Fonctionnalités de mesure de distance 29
24 Fonctionnalités de mesure d'angle 29
25 Gantt effectif du projet. 32

References

[1] Epic Games (1998). **Unreal Engine 4**. https://www.unrealengine.com. Dernier accès 14/08/2021.

[2] Inc. Cypress.io. **Cypress**. https://www.cypress.io/. Dernier accès 19/08/2021.

[3] Oleg Danilov. **Rapid Environment Editor**. https://www.rapidee.com. Dernier accès 18/08/2021.

[4] Google. **Angular**. https://angular.io/. Dernier accès 18/08/2021.

[5] Google. **Protocol Buffers**. https://developers.google.com/protocol-buffers. Dernier accès 18/08/2021.

[6] LLVM Developer Group. **ClangFormat**. https://clang.llvm.org/docs/ClangFormat.html. Dernier accès 19/08/2021.

[7] Qt Group. **Qt Library**. https://www.qt.io/. Dernier accès 18/08/2021.

[8] Jonathan Puckey Jürg Lehni. **PaperJS**. https://paperjs.org/. Dernier accès 18/08/2021.

[9] Crystal Rich Ltd. **LockHunter**. https://lockhunter.com/. Dernier accès 18/08/2021.

[10] JFrog LTD. **Conan**. https://conan.io/. Dernier accès 19/08/2021.

[11] Community of Mirador. **Mirador**. https://projectmirador.org/, https://github.com/ProjectMirador/mirador. Dernier accès 18/08/2021.

[12] Ben Carter Omar Cornut, Rokas Kupstys. **DearImGui**. https://github.com/ocornut/imgui. Dernier accès 19/08/2021.

[13] Arnaud Roques. **PlantUML**. https://plantuml.com/fr/. Dernier accès 19/08/2021.

[14] Thermo Fisher Scientific. **Avizo**. https://www.thermofisher.com/uk/en/home/electron-microscopy/products/software-em-3d-vis/avizo-software.html. Dernier accès 18/08/2021.

[15] ThermoFisher Scientific. **Athena**. https://www.thermofisher.com/uk/en/home/electron-microscopy/products/software-em-3d-vis/imaging-data-management/athena-software-for-core-imaging-facilities.html?cid=2021-vsg-gl-xx-athena_npi-xx-press_release&utm_source=press-release&utm_medium=3rd-party&utm_campaign=2021-vsg-gl-xx-athena_npi-xx-press_release. Dernier accès 14/08/2021.

[16] ThermoFisher Scientific. **Documentation OpenInventor**. Manuel : https://developer.openinventor.com/UserGuides/Oiv10/Inventor_Mentor/index.html, ToolMaker : https://developer.openinventor.com/UserGuides/Oiv10/Inventor_Toolmaker/index.html, SDK : https://developer.openinventor.com/refmans/10.8/RefManCpp/. Dernier accès 19/08/2021.

[17] ThermoFisher Scientific. **OpenInventor**. https://www.openinventor.com/. Dernier accès 14/08/2021.

[18] Gurock Software. **TestrRail**. https://www.gurock.com/testrail/. Dernier accès 19/08/2021.

[19] Unity Technologies. **Unity3D**. https://unity.com/fr. Dernier accès 14/08/2021.

[20] https://eslint.org/team. **ClangFormat**. https://eslint.org/. Dernier accès 19/08/2021.

[21] Milian Wolff. **Massif Visualizer Valgrind**. https://github.com/KDE/massif-visualizer. Dernier accès 18/08/2021.

9.1 Offre originelle de stage

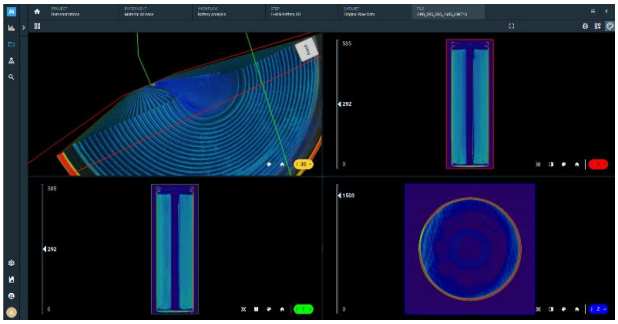
STAGE – Ajout de fonctionnalités dans une solution de visualisation 3D à distance dans un navigateur Web. (H/F)

Lorsque vous faites partie de **Thermo Fisher Scientific**, vous êtes amené(e) à relever des défis et appartenez à une équipe qui valorise les performances, la qualité et l'innovation. Au sein d'une organisation brillante, mondiale et en pleine expansion, vous êtes encouragé(e) à offrir le meilleur de vous-même. Avec un chiffre d'affaires de 18 milliards de dollars et l'investissement le plus important dans le domaine R&D du secteur, nous offrons à notre personnel les ressources et les opportunités dont ils ont besoin pour contribuer considérablement au monde qui les entoure.

Athena est une brique logicielle qui permet la centralisation de données sur une plateforme web, le suivi des workflows, ainsi que la facilitation de l'analyse de données scientifiques.

Les données obtenues par des dispositifs d'acquisition tels que les microscopes à électrons ou les microscopes optiques sont de plus en plus détaillées, il en résulte d'énormes jeux de données 3D. Afin de les visualiser en temps-réel à partir d'un navigateur Web, une application de visualisation (viewer volumétrique) basée sur une architecture client-serveur a été développée.

Pour garantir sa scalabilité, la partie backend basée sur le moteur 3D Open Inventor (C++/OpenGL) est déployée dans une stack Kubernetes. Le frontend se base sur le framework web Angular permettant l'utilisation du langage typé TypeScript.



jobs.thermofisher.com ThermoFisher SCIENTIFIC

Vos missions :

Intégré(e) à une équipe au sein de la R&D travaillant en méthode agile, vous aurez pour objectif d'enrichir le viewer afin de proposer des fonctionnalités supplémentaires répondant aux besoins utilisateur.

Une première étape sera d'ajouter au rendu existant l'affichage d'informations telles que :
- La valeur d'un voxel survolé par la souris de l'utilisateur
- Une légende contenant les informations de colormap, du dataset et d'échelle.

Cela permettra de prendre en main le frontend et le backend du viewer ainsi que le moteur 3D d'Open Inventor.

Dans un deuxième temps, le stagiaire devra effectuer un travail de recherche sur l'expérience utilisateur afin d'intégrer des outils de mesure et d'annotation dans un espace 3D.

Connaissances souhaitées : Architecture client-serveur, compréhension des mathématiques et algorithmes liés à la 3D

Qualités requises :

De formation Universitaires M2 ou école d'ingénieur dernière année, vous cherchez un stage dans un contexte technologique dynamique, motivant et innovant.

Vous disposez de connaissances en génie logiciel ainsi que dans les langages de programmation C++ et TypeScript.

Doté(e) de bonnes capacités relationnelles, vous avez le goût du travail en équipe et vous souhaitez vous investir dans des projets techniques à forte valeur ajoutée.

Stage de fin d'études conventionné de 6 mois à pourvoir au premier semestre 2021.
Gratification de stage : 1200€ brut mensuel + Ticket Restaurant

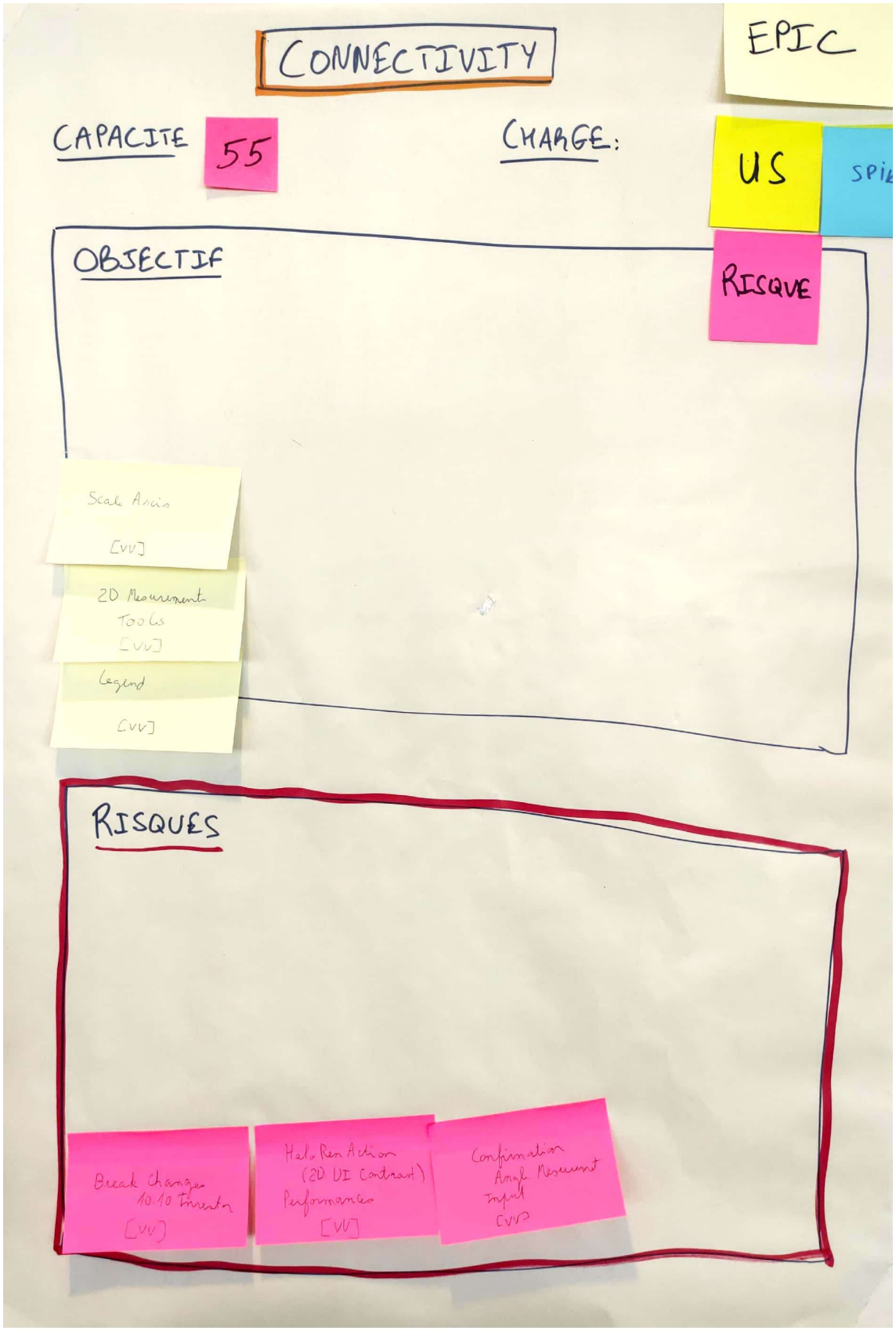
Localisation : Bordeaux (33) proche gare St Jean

Envoyez votre candidature à : Jean-Paul.Digeon@Thermofisher.com

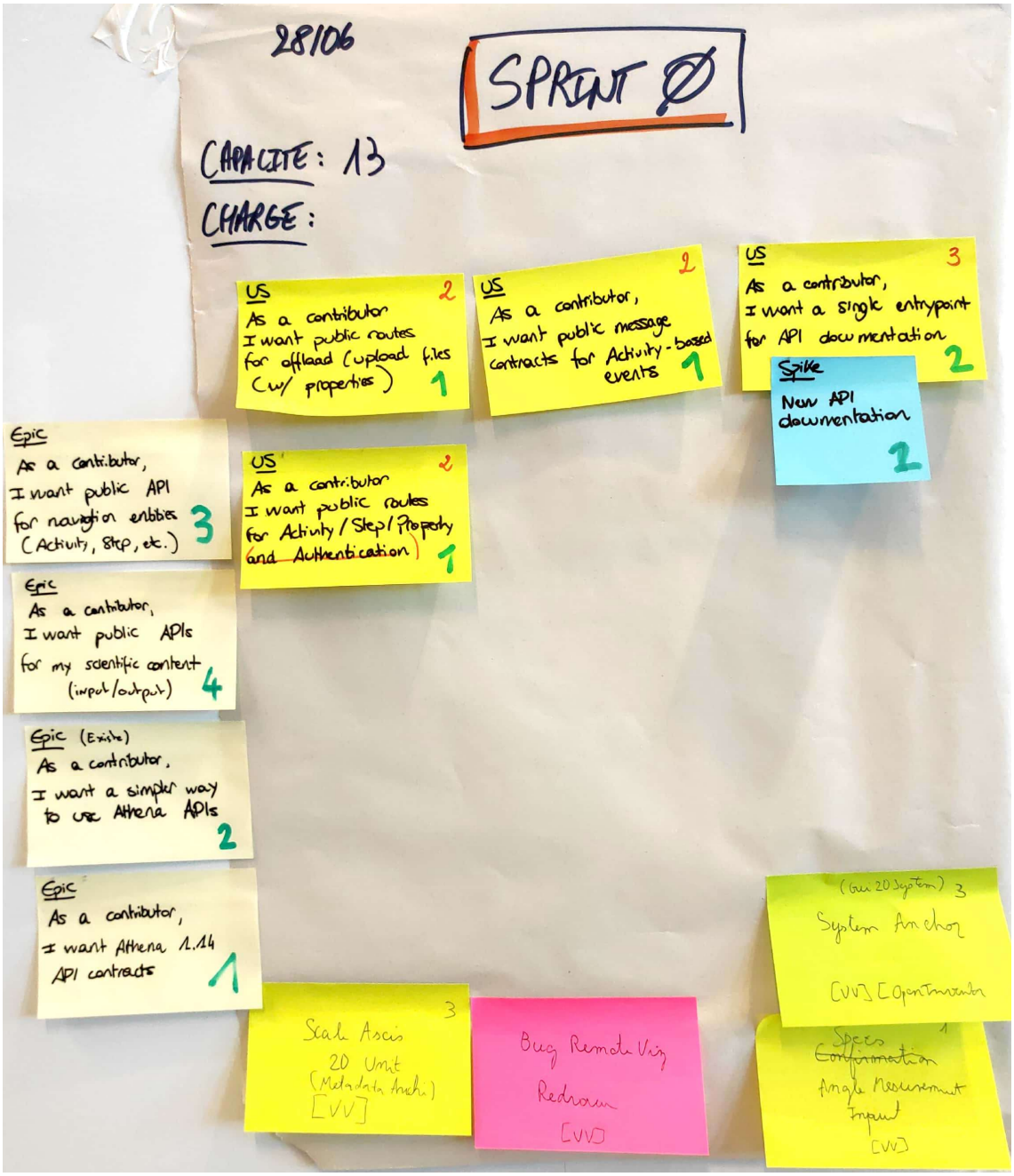
jobs.thermofisher.com

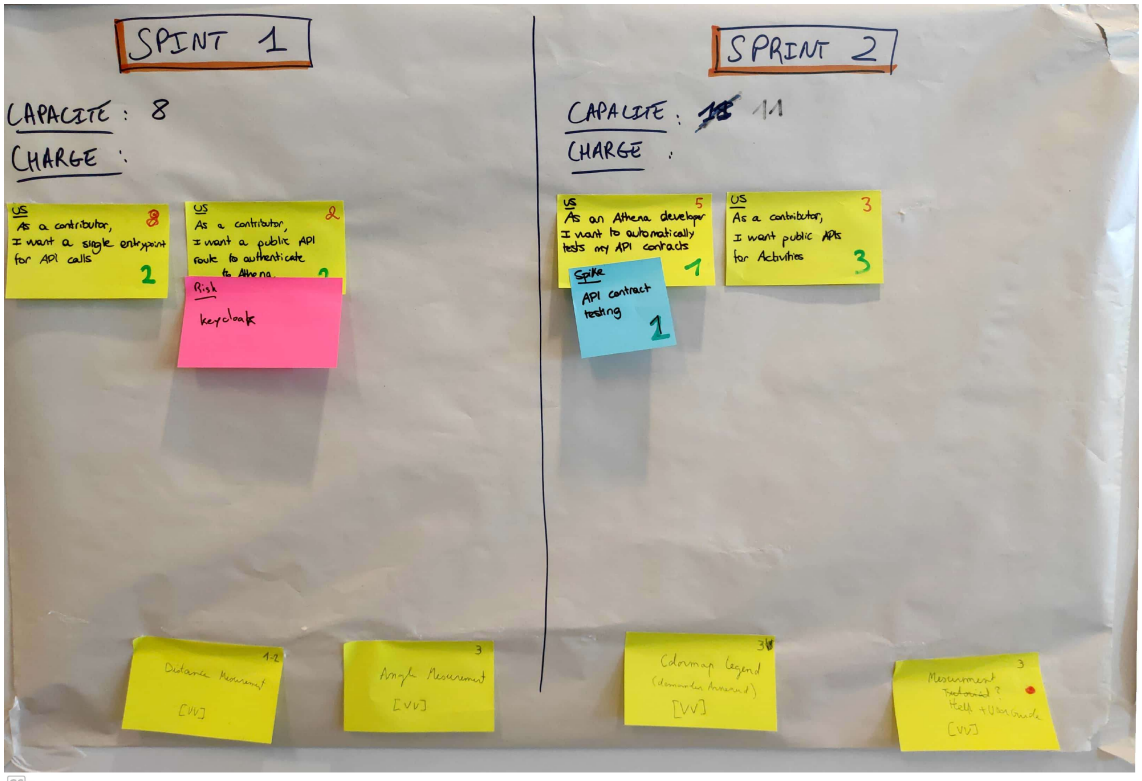
ThermoFisher SCIENTIFIC

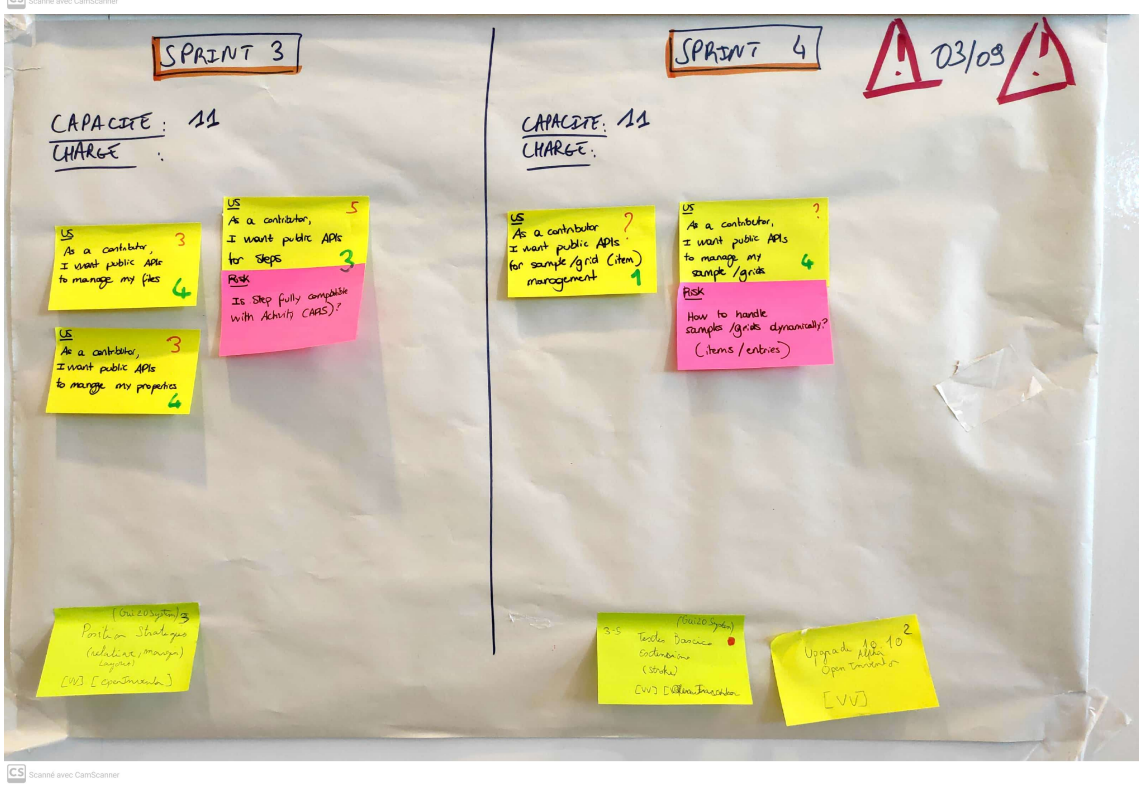
9.2 Release Plan 1.14 Connectivity



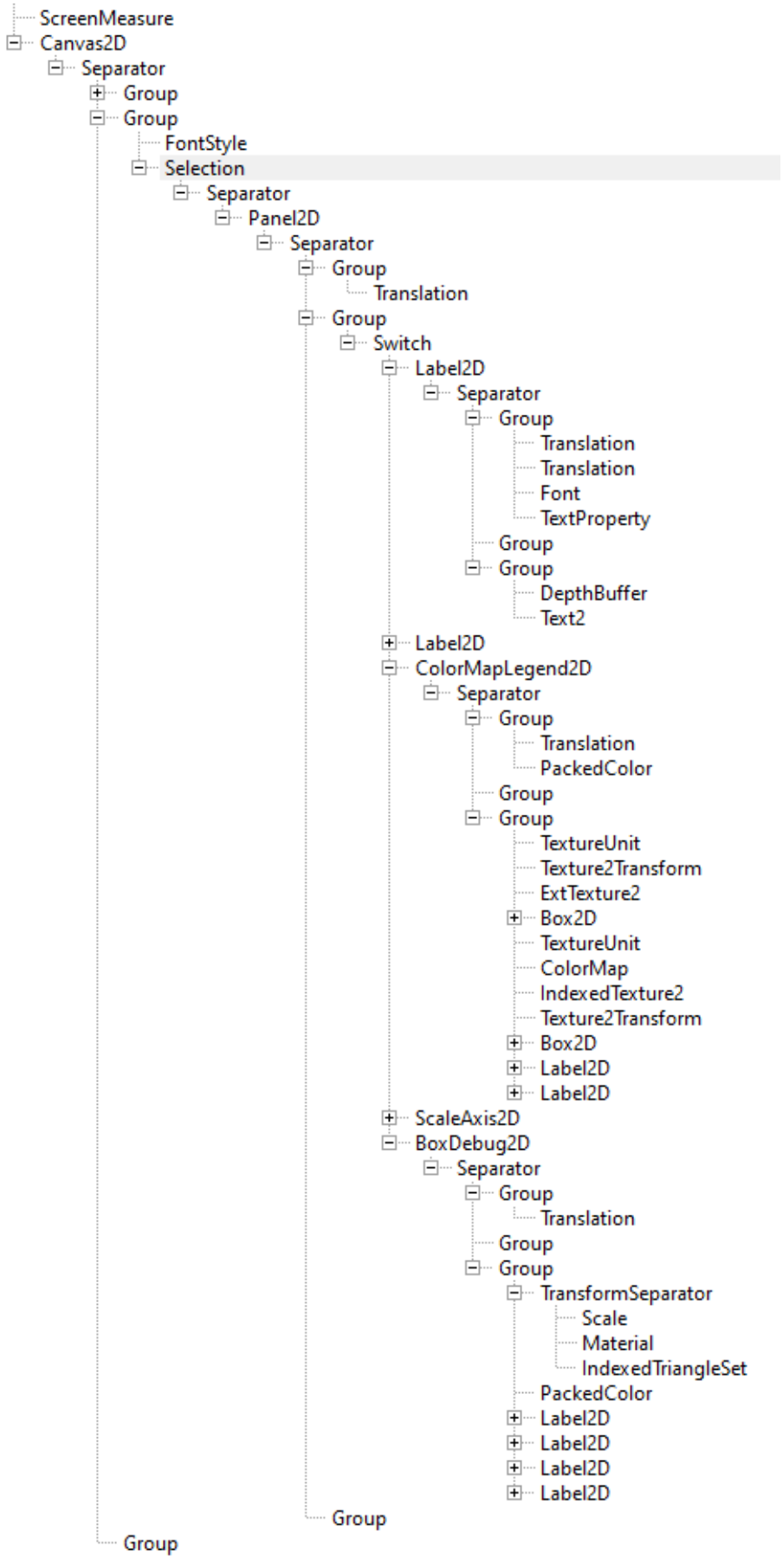
Scanné avec CamScanner


28/06
SPRINT Ø
CAPACITE : 13
CHARGE :
US
As a contributor
I want public routes
for offload (upload files
(w/ properties)
2
1
US
As a contributor,
I want public message
contracts for Activity-based
events
2
1
US
As a contributor,
I want a single entrypoint
for API documentation
3
2
Spike
New API
documentation
2
Epic
As a contributor,
I want public API
for navigation entities
(Activity, Step, etc.)
3
US
As a contributor
I want public routes
for Activity / Step / Property
and Authentication
2
1
Epic
As a contributor,
I want public APIs
for my scientific content
(input/output)
4
Epic (Existe)
As a contributor,
I want a simpler way
to use Athena APIs
2
Epic
As a contributor,
I want Athena 1.14
API contracts
1
3
Scala Ascii
2D Unit
(Metadata tracks)
[VV]
Bug Remote Viz
Redraw
[VV]
(Gui 2D System) 3
System Anchor
[VV] [OpenInventor
1
Specs
Confirmation
Angle Measurement
Input
[VV]
Scanné avec CamScanner


SPINT 1
CAPACITE : 8
CHARGE :
US
As a contributor, I want a single entrypoint for API calls
8
2
US
As a contributor, I want a public API route to authenticate to Athena.
2
Risk
keycloak
SPRINT 2
CAPACITE : 11
CHARGE :
US
As an Athena developer I want to automatically tests my API contracts
5
1
US
As a contributor, I want public APIs for Activities
3
Spike
API contract testing
1
Distance Measurement
1-2
[VV]
Angle Measurement
3
[VV]
Cdvmap legend
3
[VV]
Measurement
3
[VV]

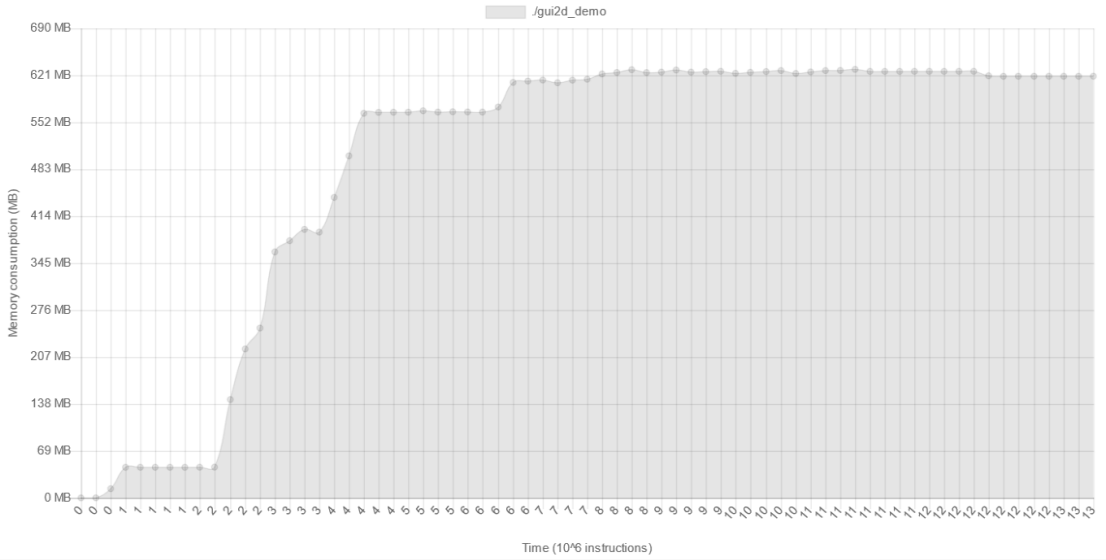

SPRINT 3
CAPACITE : 11
CHARGE :
US
As a contributor, I want public APIs to manage my files
3
4
US
As a contributor, I want public APIs for Steps
5
3
Risk
Is Step fully compatible with Activity (API5)?
US
As a contributor, I want public APIs to manage my properties
3
4
Position Strategies
3
[VV]
SPRINT 4
03/09
CAPACITE : 11
CHARGE :
US
As a contributor I want public APIs for sample/grid (item) management
?
1
US
As a contributor, I want public APIs to manage my sample/grids
?
4
Risk
How to handle samples/grids dynamically? (items/entries)
Tests Basiques
3-5
(Stroke)
[VV]
Upgrade
10
2
Open Traveltor
[VV]

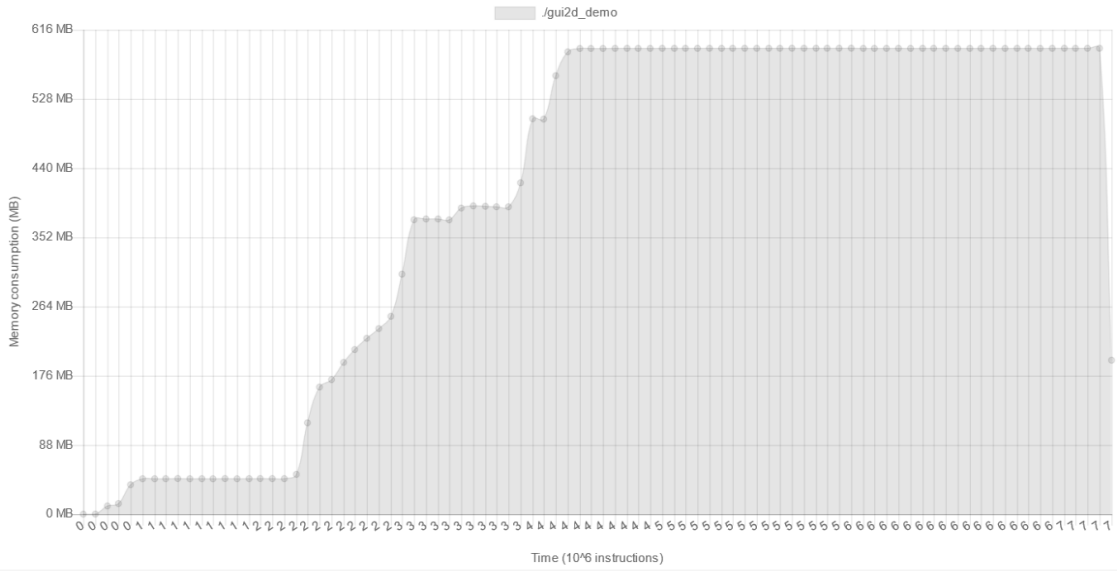
9.3 Exemple hiérarchie complète du Gui2D System



9.4 Visualisation mémoire

Images from http://boutglay.com/massifjs/, First with Gui2D system, the second without (diff 34 Mb).





9.5 Cas de Tests - TestRail

C337725: Picking tool activation

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-306, ATH-149, ATH-307

Test status
Auto

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Activate picking tool** Click on picking tool icon on top bar.	Mouse is in picking state, Picking icon is selected.
2	**Picking reset on layout modification** Activate picking tool and change view layout	Toolmode is reset to Navigation. Mouse picking icons on 3D and 2D are reset (should be rotate on 3D and slice move on 2D)

C330515: 2D view picking

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-306, ATH-149, ATH-307

Test status
Auto

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Valid picking 2D** Pick on the volume on the 2D view	Picking is valid, picked point displayed on the botton bar. Mouse cursor icon is "picking", default crosshair.
2	**Invalid picking 2D** Pick outside the volume on the 2D view	Picking is invalid displayed on the botton bar
3	**Valid picking 2D on small dataset** Load test_nm.json file (ATH-306) Pick on the volume on the 2D view	Picking is valid, picked point displayed on the botton bar

C337728: 3D view picking

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-306, ATH-149, ATH-307

Test status
Auto

Steps

	Step	Expected Result
1	**Disbaled picking 3D** Pick on the volume on the 3D view	Picking is not possible, mouse cursor icon is "disabled", default not-allowed.

C337729: Mouse drag picking

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-306, ATH-149, ATH-307

Test status
Auto

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Valid picking after mouse move** Activate picking, press mouse left, move and release mouse left on a valid picking zone.	Picking is not triggered.
2	**Invalid picking after mouse move** Activate picking, press mouse left, move and release mouse left on an invalid picking zone.	Picking is not triggered.

C339071: Scale axis

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-150, ATH-336, ATH-256, ATH-710, ATH-711

Test status
Manual

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Activate scale axis** Enable scale axis on a 2D view Default dataset.	The scale axis is displayed, no unit is displayed next to the multiplication factor. Scale axis icon is changed to the elected one (blue icon)
2	**Dynamic scale axis** Zoom in or out in the view	The scale axis is redrawn and the scale steps are changed.
3	**Scale axis redraw on drag and drop** Drag and drop the view over an other	The view with the scale axis is well redrawn (ATH-710)
4	**Scale axis redraw on layout modification** Activate scale axis on all 2D views. Switch the views layout.	The scale axis is disabled on all views. The scale axis icons are reset to the disabled one (white icon). (ATH-710)
5	**Scale axis units** Load back\data\test_nm.json	The scale axis is displayed with unit (m) displayed next to the multiplication factor.

C339458: Distance tool activation

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-351, ATH-380

Test status
Auto

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Activate distance tool** Click on distance tool icon on top bar.	Distance icon is selected.
2	**Mouse cursor forbiden on 3D view** Move the mouse on a 3D View	Measure is not possible, mouse cursor icon is "disabled", default not-allowed.
3	**Mouse cursor crosshair on 2D view** Move the mouse on a 2D View	Measure is possible, mouse cursor icon is like picking, default crosshair.
4	**Draw a line on 2D view (no graphic check)** Click and then drag on a 2D View.	Can complete all the interactions without errors, and a distance is displayed on the botton bar.
5	**Distance reset on layout modification** Activate distance tool and change view layout	Toolmode is reset to Navigation. Mouse cursors icons on 3D and 2D are reset (should be rotate on 3D and slice move on 2D)

C339459: Coherent distances

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-351, ATH-380

Test status
Auto

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Sucessives measures** Take a measure after an another (reference) with a determine ratio.	The measures must have the same ratio as the pixel distance.
2	**Sucessives measures after resize** Take a measure, resize the view and do then an another measure with the same pixel distance.	The measures must be equal.
3	**Sucessives measures after translation** Take a measure, translate the view and do then an another measure with the same pixel distance.	The measures must be equal.
4	**Sucessives measures after zoom** Take a measure, zoom/unzoom the view and do then an another measure with the same pixel distance.	The second measure must be less if we dezoom, greater if we zoom.

C339460: Distance values

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-351, ATH-380

Test status
Manual

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Correct values** Loatd test.raw and measure the square at slice 5 on Z view.	Distance must be around 1. Can use the scaleAxis to measure the axes coherency.
2	**Unit correct** Load test_nm.json file. Measure one element.	The power must be 10e-9 with the m unit

C339464: Distance ergonomic exploration

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-351, ATH-380

Test status
Manual

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Contrast line** Draw a line on white background and on black background.	Must remain visible.
2	**Contrast text** Draw a line on white background and on black background.	Text must remain readable.
3	**No overlapp text and lines** Click & drag around the first point.	Text must remain readable and not overlapp the line feedback.
4	**Diameter with shift** Load battery dataset. Measure a circle.	Ergonomie must permit to easily measure a circle.
5	**Cancel distance** Measure and before ending it, press alt during release.	The bottom bar must not be ovverriden and the previous measure must apeared if there was one.
6	**Interaction click & drag exploration**	Must be easy to manipulate.

C339461: Angle activation tool

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-360, ATH-380

Test status
Auto

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Activate distance tool** Click on distance tool icon on top bar.	Angle icon is selected.
2	**Mouse cursor forbiden on 3D view** Move the mouse on a 3D View	Measure is not possible, mouse cursor icon is "disabled", default not-allowed.
3	**Mouse cursor crosshair on 2D view** Move the mouse on a 2D View	Measure is possible, mouse cursor icon is like picking, default crosshair.
4	**Draw a angle on 2D view (no graphic check)** Click & drag two times on a 2D View.	Can complete all the interactions without errors, and a angle is displayed on the botton bar.
5	**Angle reset on layout modification** Activate Angle tool and change view layout	Toolmode is reset to Navigation. Mouse cursors on 3D and 2D are reset (should be rotate on 3D and slice move on 2D)

C339462: Angle values

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-360, ATH-380
Test status			
Auto			

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Squared Angle** Do an angle mesure with a squared angle.	Must be equal to 90 °
2	**Directed angle** Do an angle mesure, and then mesure the opposite angle. Try it on X and Y flip.	Must be equal
3	**Convention on angle offset** Do an angle mesure, less than 90° and greater than 90°.	When the two vectors are equal, angle must be 0° and when they are opposite angle must be 180°
4	**Angle on zoom, translation, resize** Do an angle mesure, and then apply a zoom, translation, resize and each time measure again.	Must be equal

C339463: Angle ergonomic exploration

Type	Priority	Milestone	References
Functionality	Undefined	None	ATH-360, ATH-380

Test status
Manual

Preconditions

The back and the front are running. The page is loaded. A dataset is loaded.

Steps

	Step	Expected Result
1	**Contrast line** Draw a angle on white background and on black background.	Must remain visible and angle direction be clear.
2	**Contrast text** Draw a angle on white background and on black background.	Text must remain readable.
3	**No overlapp text and lines** Click & drag two times and on the second before ended it, turn around the first vector.	Text must remain readable and not overlapp the line feedback, angle must be outside the angle.
4	**Cancel distance** Measure angle and before ending it, press alt during release.	The bottom bar must not be ovverriden and the previous measure must apeared if there was one.
5	**Interaction click & drag exploration**	Must be easy to manipulate.

9.6 Présentation du Gui2D System

WHITE PAGE

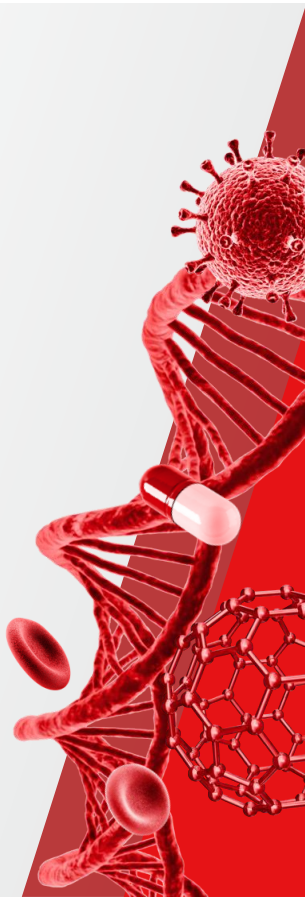
Proposition : Gui2D System

A screen adaptive user interface for Open Inventor

Marc Cerutti

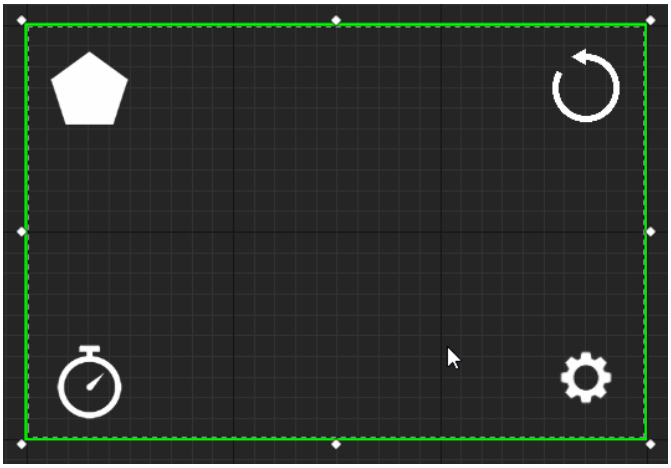
Trainee Developer

06/22/2021



The world leader in serving science

Concept



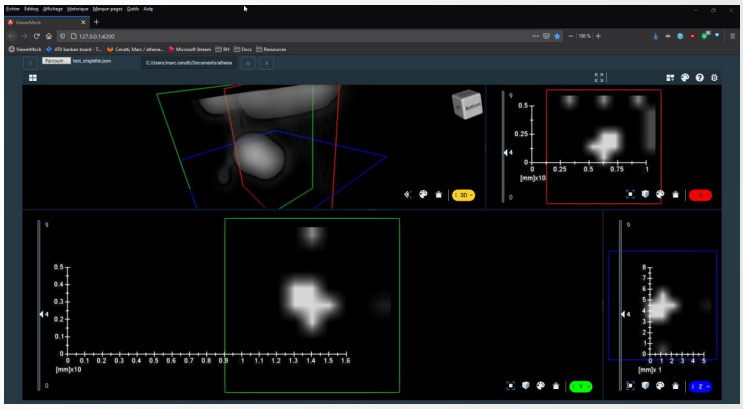
The Gui2D system aim to make easier to display 2D information for the user natively in the Open Inventor 3D engine.

It will not replace a third-party system for interface like QT for example, but it can be rather useful for rapid prototyping, remote screenshot and specific display linked to Open Inventor Nodes.

The main feature is to provide screen-adaptative user interface elements for Text, Axis and Legends (colormap, metadata, title, etc).

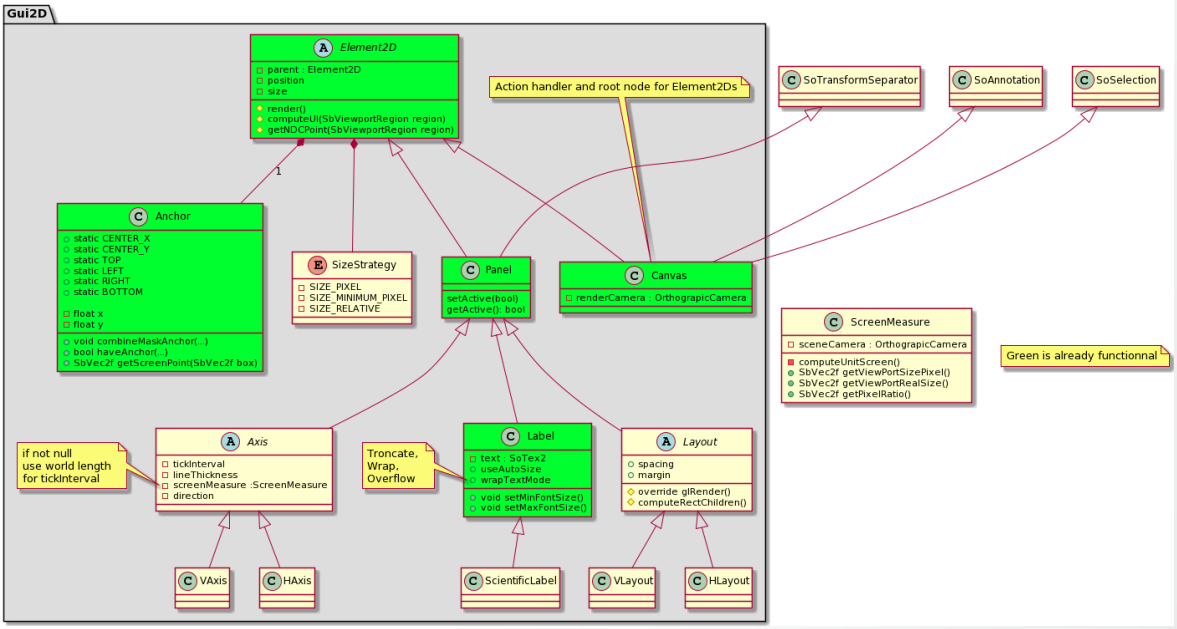
Why ?

- Make prototyping quicker
- Immediate screenshot saving implementation with resolution aimed (usefull in remote context)
- Display informations coupled with the view (colormap legend, camera position, graduate axis,…)
- No third-party dependencies
- Based on already existing Nodes (SoText2, SoGroup,…)
- Basic display system already done (SoAnnotation, SoOrthographicCamera, SoGlRenderAction,…)



Screenshot from Athena 3D viewer showing **ScaleAxis** component, implementing the main ideas for this solution,

Architecture



Features

- Dynamic positioning with **Anchors**
- Commutable placement convention, between relative (normalized) and fixed (pixel), always relative to parent and the **Anchor**
- No camera transformation (belong only to ***Canvas*** Node)
- *SoAnnotation* features (display above scene)
- Different size strategies (fixed in pixel, minimum-maximum, relative)
- Automatic **Layout** controlling children's position and size with spacing and margin
- Extend text functionalities for updating with container size and resolution (minimal size, overflow, wrapping,...)

Main problematics

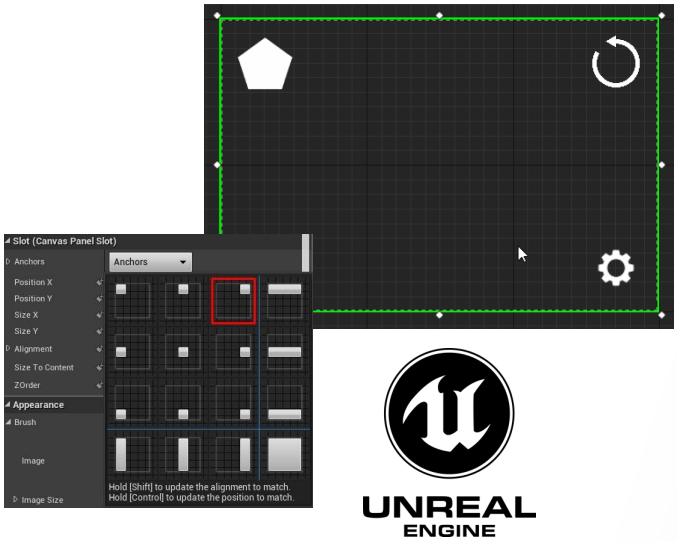
Font size, text size, extensions (wrap, overflow, truncate) and text display (stroke, …)

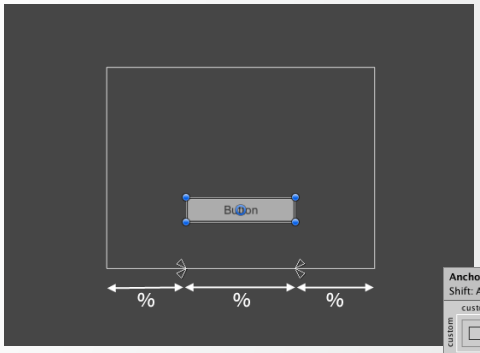
Caching and resize

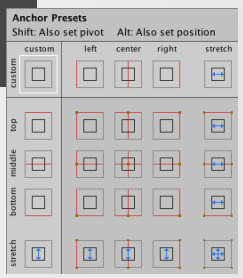
Open Inventor compatibility (new actions, rendering, …)

Wide scope to make generic

References







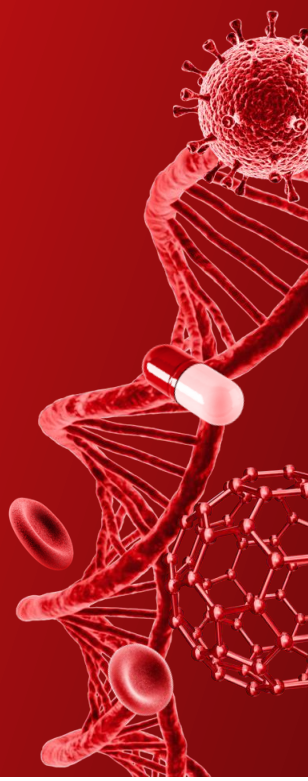
UNREAL ENGINE

unity

Unity3D : https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UIBasicLayout.html

Unreal : https://www.raywenderlich.com/480-unreal-engine-4-ui-tutorial#toc-anchor-007

Questions



9.7 Autres

Concurrent dans le domaine du traitement d'image: Apeer https://www.apeer.com/home, accès le 04/08/2021.