Observatoire astronomique
de Strasbourg

Rapport de Stage

Création d'une palette d'outils et interprétation de données astronomiques en 3D

Thibault BOUCHARD

Année Universitaire 2015/2016

Maître de Stage : André Schaaff

Remerciement

Tout d'abord, je tiens à remercier toutes les personnes qui ont contribué, de près ou de loin, par leur aide et assistance, à la réalisation de ce stage de deuxième année.

Mes remerciements iront tout d'abord à Mr. André Schaaff, ingénieur de recherche à l'Observatoire de Strasbourg pour m'avoir accordé sa confiance, m'avoir accueilli au sein de la structure, pour le temps qu'il m'a consacré tout au long du stage en répondant à toutes mes interrogations et pour son aide précieuse à l'élaboration du rapport de stage.

Je tiens à remercier aussi Hervé Wozniak, directeur de l'observatoire et Mark Allen, directeur du CDS pour m'avoir accueilli au sein de l'observatoire de Strasbourg.

J'aimerais aussi remercier Jérôme Desroziers, stagiaire à l'Observatoire à qui j'ai pris la suite de l'application de visualisation 3D puis employé en job d'été pour son aide, ses nombreux conseils et pour tout le temps qu'il a consacré afin de m'expliquer le fonctionnement de l'application.

Je remercie aussi toutes les personnes avec qui j'ai travaillé sur l'application ainsi que tous les stagiaires et l'ensemble du personnel de l'observatoire pour leur accueil et pour leur aide afin de mener ce stage à bien.

Je voudrais aussi remercier tous mes proches pour la relecture de ce rapport.

Table des matières

Chapitre 1

Indroduction

Lors de ma deuxième année à l'ENSIIE Strasbourg, j'ai eu l'occasion de réaliser mon stage de dix semaines au sein de l'observatoire de Strasbourg.

Le but du stage était d'ajouter une palette d'outils à une application de visualisation 3D de données astronomiques. En effet, l'observatoire développe depuis plus d'un an une application Web qui permet de visualiser directement dans le navigateur et en trois dimensions des données astronomiques (comme des nuages de particules par exemple, cf Figure 1.1). J'ai donc développé une palette d'outils permettant l'interprétation de ces données ainsi que quelques outils (comme un calcul de distance entre deux points par exemple). Cette palette d'outils et ces outils devaient être robustes et capables d'être appliqués à des volumes importants de données.

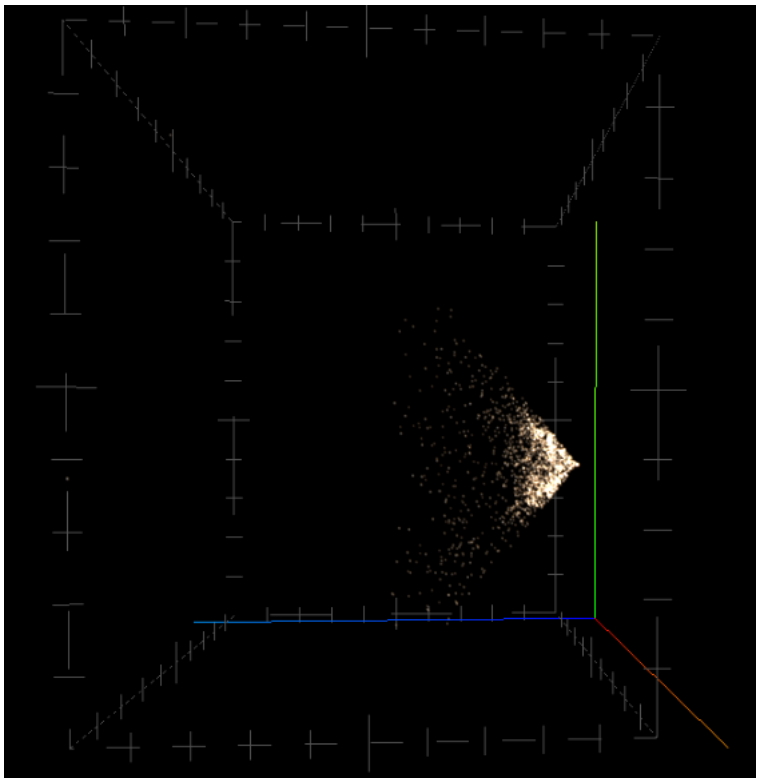


FIGURE 1.1 – Exemple de nuage de points visualisé

J'ai donc choisi de participer à ce projet car la problématique de la programmation 3D me plaît et je trouve très intéressant d'appliquer la visualisation 3D au domaine scientifique. De plus, ce sujet pouvait m'apporter de nouvelles connaissances dans le domaine de la programmation web que je connaissais assez peu jusqu'à lors.

Dans la suite du rapport, dans un premier temps, nous décrirons l'observatoire. Ensuite, nous aborderons le déroulement du stage en insistant sur mes réalisations. Enfin, nous dresserons un bilan de ce stage.

Chapitre 2

L'Observatoire de Strasbourg

2.1 Activités

L'Observatoire de Strasbourg est un établissement de recherche et d'enseignement axé sur l'astronomie. L'Observatoire fournit plusieurs activités :

- Une activité de diffusion des connaissances et de vulgarisation scientifique.
- Une activité d'enseignement pour des masters et des licences scientifique ainsi qu'une préparation à l'agrégation et au CAPES.
- Une activité de recherche organisée autour de trois équipes :
  - L'équipe Galaxies qui mène des études sur la formation des galaxies ainsi que sur la population stellaire des galaxies. Les recherches sont menées plus spécifiquement sur la Voie Lactée (notre galaxie) ainsi que sur les galaxies proches.
  - L'équipe de recherche Hautes Énergies s'intéresse à la physique des astres compacts (par exemple les étoiles à neutrons), aux sources émettrices de rayons X ainsi qu'aux noyaux de galaxies.
  - Le centre de données astronomique de Strasbourg (CDS) développe un ensemble de services permettant un accès libre à des données astronomique. Ces services comprennent notamment trois logiciels :

    VizieR : C'est un service de base de données sur lequel on peux effectuer des requêtes sur un ou plusieurs catalogues astronomiques.

    Simbad : C'est aussi un service de base de données qui lui est centré sur les astres (étoiles, nuages de gaz ...). On effectue de requête sur cette base de données en se basant sur le nom ou sur des propriétés physiques des astres. Il y a environ 8300000 objets répertoriés dans Simbad ainsi que 23000000 d'identifiants associés à ces objets.

    Aladin : C'est un atlas interactif qui permet de visualiser la voûte céleste avec des images provenant de différentes observations astronomique.

2.2 Raison Sociale, Statut Juridique et Actionnariat

L'Observatoire de Strasbourg est un observatoire des sciences de l'univers de l'Institut National des Sciences de l'Univers (INSU) ainsi qu'une unité mixte de recherche scientifique du Centre national de la recherche scientifique (CNRS) et de l'Université de Strasbourg.

2.3 Historique

1673 : Création du premier observatoire de Strasbourg sur l'une des tours de l'enceinte de la ville.

1828 : Création du second observatoire de Strasbourg sur le toit des bâtiments de l'Académie.

1881 : Inauguration de l'actuel observatoire.

1972 : Création du CDS (sous le nom *Centre de Données Stellaires* puis devient en 1982 le *Centre de Données astronomiques de Strasbourg*).

1981 : Création du planétarium dans les locaux de l'observatoire.

2008 : Le CDS est labellisé "Très Grande Infrastructure de Recherche".

Chapitre 3

Objectifs du Stage

3.1 Sujet du Stage

Depuis plus d'un an, une application de visualisation 3D sur navigateur web sans plugin est en développement. Cette application a pour but de visualiser en trois dimensions des nuages de particules ainsi que de faire une analyse des données astronomiques (comme des nuages de particules par exemple). L'application est pour le moment utilisée par des astronomes et thésards de l'observatoire pour visualiser facilement de "petits" nuages de particules (d'au maximum quelques millions de points). Or ces astronomes et thésards ont besoin d'outils pour interpréter et analyser les données chargées dans l'application, en effet l'application ne permet que d'afficher des données dans le navigateur mais ne donnait que peu de moyen d'interpréter les données (une capture d'écran de l'application avant mon stage est donnée Figure 3.1). Le but de ce stage était donc de créer une palette d'outils ainsi que l'ajout d'outils dans l'application permettant l'interprétation de données. Une vue général de l'application est donnée Figure 3.2.

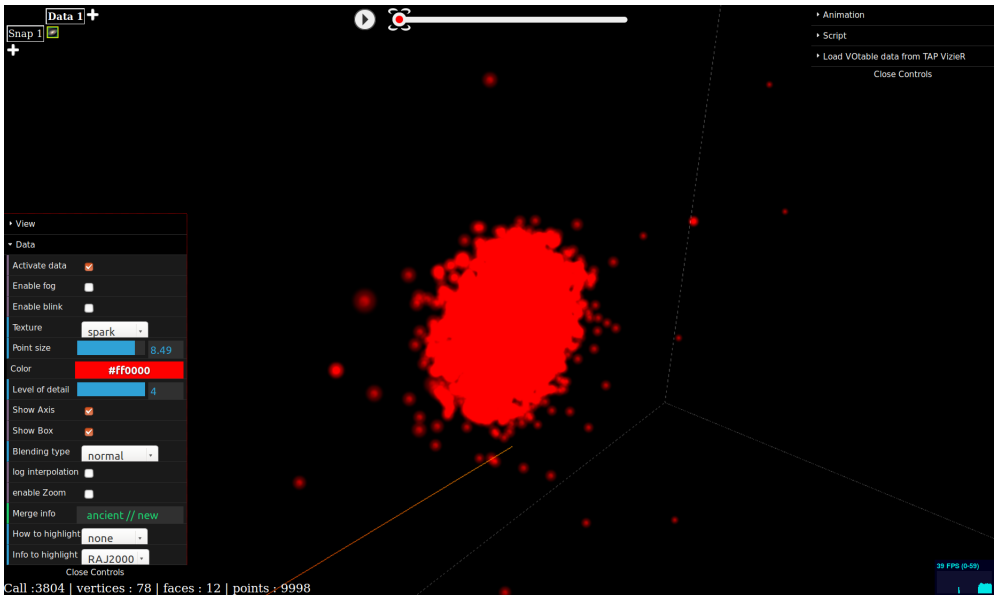


FIGURE 3.1 – Vue générale de l'application avant mon stage

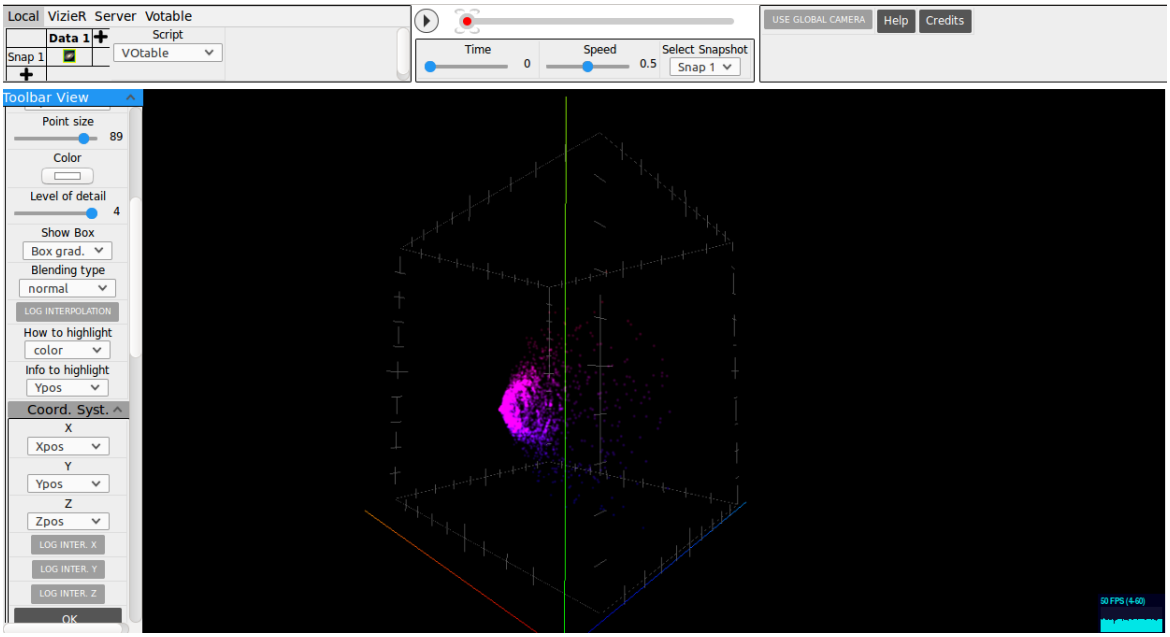


FIGURE 3.2 – Vue générale de l'application à la fin du stage

3.2 Demandes et Contraintes du Projet

Lors de ce projet, de nombreuses contraintes ont été fixées. J'ai donc dû respecter ces points suivants :

— Programmer toutes mes modifications en langages Web (c'est-à-dire en HTML5, CSS3 et JavaScript) ainsi qu'avec la bibliothèque THREE.js. En effet, l'application est programmée dans ces langages et donc les modifications doivent être aussi apportées dans ces langages.

— Les modifications doivent être compatibles avec les navigateurs internet les plus utilisés, c'est-à-dire avec Mozilla Firefox, Safari, Google Chrome et Microsoft Edge (ou Internet Explorer).

— S'adapter à l'architecture de l'application. En effet, quatre stagiaires ont déjà participé à l'élaboration de l'application avant le début de mon travail, le cœur de l'application était donc déjà construit et j'ai donc adapté toutes mes modifications en fonction du code déjà existant.

— Veiller à la performance et à la robustesse de mon travail, en effet, un gros travail a été mené par les anciens stagiaires au niveau des performances pour que l'application puisse fonctionner sur de plus gros volumes de données dans un simple navigateur web sans problème majeur d'affichage, de ralentissement, de crash du navigateur...

— Réaliser une palette d'outils robuste. Cette palette devra être petite, redimensionnable en fonction de la taille de l'écran (c'est-à-dire que la palette doit s'adapter à toutes les tailles d'écran d'ordinateur et doit se redimensionner si la fenêtre change de taille). De plus, la palette doit pouvoir être cachée entièrement afin d'avoir un mode plein-écran de l'application (c'est-à-dire une fenêtre où n'est présente que la scène 3D). De plus, il fallait une possibilité de cacher une partie des outils et même une partie de la palette afin d'éviter l'encombrement de l'écran et pour cacher les

outils non utiles à l'utilisateur.

— Pouvoir ajouter facilement par le biais de la programmation des outils à la palette ainsi que comprendre facilement l'architecture de la palette afin de pouvoir créer de nouveaux types d'outils.

— Avoir une organisation intuitive de la palette et des outils afin de garantir une facilité d'utilisation pour l'utilisateur.

— Travailler en équipe avec Nicolas Adam et Jérôme Desroziers sur l'application. Cela impliquait d'avoir une bonne communication au sein de l'équipe au sujet des modifications apportées par chacun au jour le jour pour éviter les conflits dans le code source au sein du GitLab de l'application, pour adapter des modifications en fonction des besoins de chacun (par exemple ajouter des outils dans la palette pour qu'un autre membre de l'équipe puisse continuer ses modifications), pour échanger des conseils à sujet du langage de programmation et des modifications apportées ainsi que pour signaler les bugs.

3.3 Critères de Validation du Projet

Mon travail a été jugé en fonction du respect des critères décrits ci-dessus. Mes modifications devaient être entièrement utilisables et devaient être facilement maintenables dans l'application pour la suite du développement de l'application.

3.4 Moyens Mis à Disposition du Stagiaire

J'ai travaillé avec une machine sous Ubuntu avec l'environnement de programmation Webstorm. Un accès à l'intranet du CDS m'a été donné. J'y accédais aux pages Wiki des stagiaires (qui sont des sortes de journaux de bord où les stagiaires disent au jour le jour ce qu'ils ont fait et les difficultés rencontrées) ainsi qu'aux rapports de stage des anciens stagiaires qui ont travaillé sur l'application. J'ai eu aussi accès au GitLab de l'application sur lequel j'ai travaillé en collaboration avec Nicolas Adam et Jérôme Desroziers.

3.5 Planification du Projet

Mes travaux ont été planifiés au début de mon stage en cinq parties, voici les principaux travaux que j'avais prévu de réaliser (décrites ci-dessous par ordre chronologique) :

— Découverte et prise en main de l'environnement de programmation et des technologies utilisées. Analyse du code-source et de l'architecture du projet. Réunion et échange avec les principaux acteurs du projet au sujet de l'application et de ses axes de développement.

— État de l'art au sujet de l'implémentation d'une palette de fonctionnalités et d'outils dans l'application.

— Création de la base de la palette d'outils et des outils de base.

— Migration petit à petit des fonctionnalités de l'application jusqu'à ce que toutes les fonctionnalités présentes dans l'application soient dans la nouvelle palette d'outils.

— Création de nouveaux outils et fonctionnalités dans l'application.

Voici la chronologie du stage, semaine par semaine de ce qui a été effectivement effectuée lors du stage :

- **Semaine 1 :**
  - Découverte et prise en main de l'environnement de programmation et des technologies utilisées. Analyse du code-source et de l'architecture du projet. Réunion et échange avec les principaux acteurs du projet au sujet de l'application et de ses axes de développement.
  - Début du développement de l'affichage des informations des particules, cela n'a pas nécessité une modification de la palette d'outils.
- **Semaine 2 :**
  - Fin du développement de l'affichage des informations des particules.
  - Recherche au sujet de l'implémentation d'une palette de fonctionnalités et d'outils dans l'application.
- **Semaine 3 :**
  - Création de la base de la palette d'outils et des principaux outils.
- **Semaine 4-5-6 :**
  - Refactorisation du code-source de l'application au niveau de l'interface graphique afin de pouvoir implémenter facilement et proprement la palette d'outils.
  - Création du design de la palette d'outils.
  - Migration progressive des fonctionnalités de l'application jusqu'à ce que toutes les fonctionnalités présentes dans l'application soient dans la nouvelle palette d'outils.
- **Semaine 7-8-9 :**
  - Création de nouveaux outils et fonctionnalités dans l'application.
  - Débogage de la palette d'outils et de l'application en générale.
  - Rédaction de la documentation technique de l'interface graphique de l'application et du fonctionnement de la palette d'outils (cf Annexe A).
- **Semaine 10 :**
  - Rédaction du rapport de stage.

Chapitre 4

Réalisation du Projet

4.1 Analyse des Demandes et des Contraintes

L'analyse des demandes et des contraintes du projet s'est effectuée principalement à partir des indications données tout à long du stage.

4.2 Conception Générale

Mes modifications ont porté essentiellement sur l'interface graphique de l'application, j'ai modifié tous les éléments d'affichage de l'application afin de rendre l'interface plus ergonomique, plus cohérente et plus intuitive.

4.2.1 Affichage des informations d'une particule

Pour la plupart des données chargées dans l'application, des informations sont associées (comme par exemple des positions dans l'espace, des énergies associées à ces particules, des caractéristiques physiques ...). Il est donc intéressant d'afficher ces données dans l'application pour les utilisateurs.

Avant le début de mon stage, pour afficher ces informations, il fallait cliquer sur une particule et les informations s'affichaient en bas à gauche de l'écran et la particule cliquée se colorait dans la couleur négative de la couleur d'origine de l'étoile (c'est-à-dire que si l'étoile était blanche, l'étoile une fois sélectionnée se colorait en noir par exemple) (cf Figure 4.1. Dès que l'on re-cliquait sur la scène 3D, ces informations disparaissaient aussitôt.

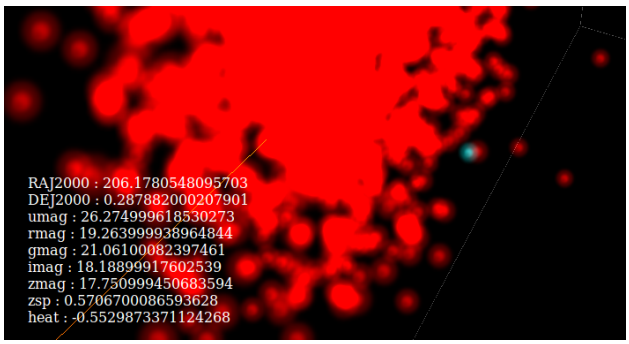


FIGURE 4.1 – Affichage des informations d'une particule (coloré en bleu) avant

Cette solution ne me paru pas adaptée. En effet, le fait de colorer les étoiles avec leur négatif faisait tout simplement disparaître la plupart du temps l'étoile, en effet, comme l'on ne colore que très rarement les étoiles en une autre couleur que le blanc, quand on sélectionnait une étoile, elle se colorait en noir or le noir étant la couleur du fond, l'étoile disparaissait et donc il était presque impossible de voir l'étoile en même temps que les informations.

J'ai donc décidé de rendre cette affichage plus ergonomique. Pour ce faire, l'affichage de ces informations se fait par le biais d'une bulle (un peu comme une bulle de dialogue de bande-dessiné) (cf Figure 4.2). Cette bulle est pointée sur l'étoile associée aux informations (ce qui fait qu'on n'a pas besoin de colorer l'étoile en question en négatif et donc l'étoile est toujours visible). De plus, cette bulle reste visible a l'écran tant que l'utilisateur n'a pas fait un clic "simple" dans la scène 3D, c'est-à-dire que l'utilisateur peut se déplacer dans la vue avec le clavier et avec la souris en faisant un clic du type "glisser-déposer" sans que la bulle ne s'efface ce qui permet de visualiser la position de la particule facilement. La bulle s'efface aussi en cas d'animation des étoiles (il est possible dans l'application de faire des animations pour voir le déplacement des particules entre deux jeux de données) et en cas de changement des données en général (en effet, quand on charge dans l'application des données d'observations (au format *VOTable*), ces données sont redimensionnées afin d'être contenues dans un cube de taille 1 x 1 x 1 qui correspond à la scène 3D, or, quand on charge une nouvelle fois une donnée de type *VOTable*, toutes les données précédemment chargées sont encore une fois redimensionnées afin que l'affichage des données soit toujours cohérent et soit toujours contenu dans un cube 1 x 1 x 1).

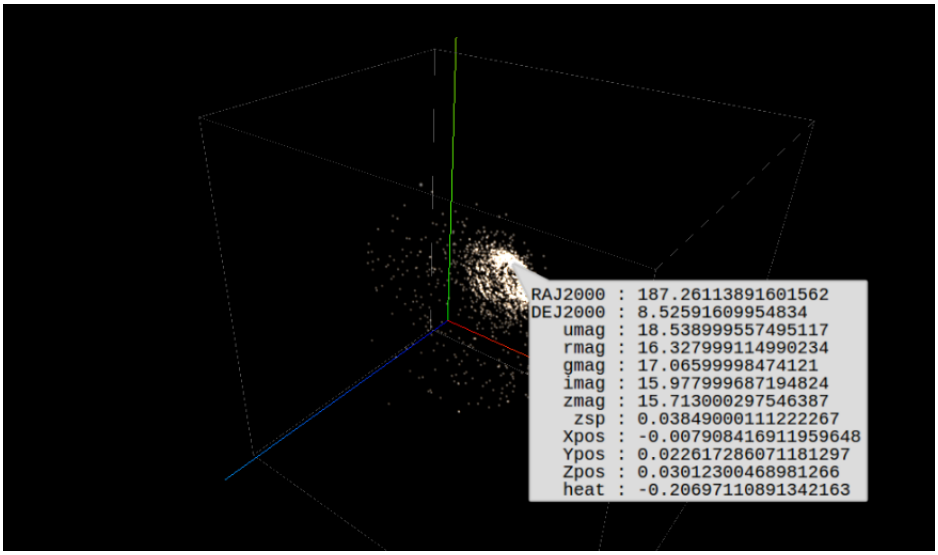


FIGURE 4.2 – Affichage des informations d'une particule

4.2.2 Création de la palette d'outils

État de l'Interface avant mon Stage

La palette d'outils est primordiale pour le fonctionnement de l'application. En effet, elle contient tous les outils de l'application qui servent à charger et à agir sur les données affichées dans la scène 3D. Il est donc très important d'avoir une interface claire, cohérente et facile à utiliser.

Avant le début de mon stage, une palette d'outils sommaire était présente (cf Figure 4.3). Cette interface utilisait la bibliothèque dat.GUI qui permet de créer des barres d'outils. L'interface utilisait aussi quelques éléments HTML qui servent à charger les données (en haut à gauche), à lancer des animations (en haut au centre) et à afficher des informations de débogage (en bas à gauche).

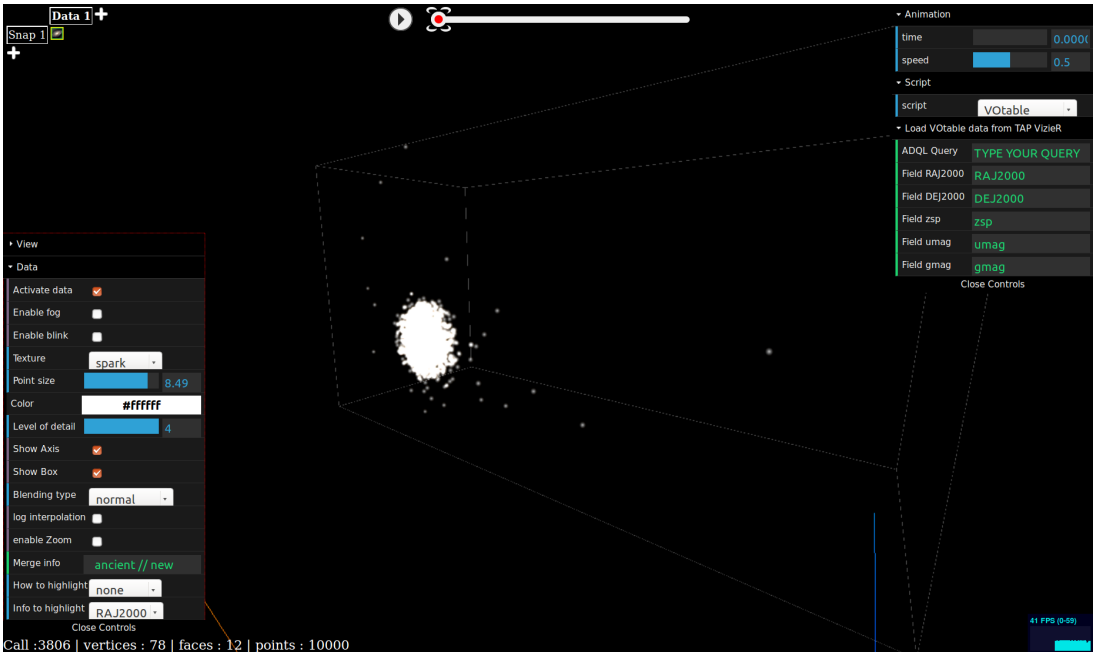


FIGURE 4.3 – Interface de l'application avant mon stage

Cet interface n'était pas suffisante, en effet, elle avait plusieurs problèmes :

— Les barres d'outils dat.GUI n'étaient pas adaptées à l'application :
  — Ces barres d'outils sont conçues pour agir sur de petites applications avec peu d'outils or l'application en ayant déjà beaucoup et étant vouée à en avoir de plus en plus ce qui allait conduire rapidement à un encombrement de l'interface de l'application.
  — Ce type de barre d'outils ne permettait pas d'afficher des valeurs (comme des valeurs de mesure par exemple) et ne permettait pas l'ajout de nouveaux types d'outils intéressants ou plus élaborés pour l'application.
  — dat.GUI ne comporte aucune fonction permettant de modifier l'état d'un outil par la programmation (par exemple cocher une checkbox par la programmation). Cela était donc assez embêtant, en effet, il est possible de sélectionner

dans l'application une donnée courante, c'est-à-dire un jeu de données sur lequel on agit directement (comme changer la couleur de ces données), or certains outils dépendent directement des paramètres de la donnée (comme la couleur) et comme on peut changer la donnée courante, les outils doivent se mettre à jour pour afficher les paramètres de la nouvelle donnée courante, ce qui est impossible nativement avec dat.GUI. Les anciens stagiaires ont réussi à contourner ce problème en modifiant directement le code HTML des outils mais cette solution n'était ni propre, ni satisfaisante car elle rendait difficile l'ajout de nouveaux outils.

  - dat.GUI ne permet pas de faire des barres d'outils horizontaux. Il est donc impossible de remplir l'écran horizontalement.

- L'organisation de l'ancienne interface n'est pas bonne ce qui pose des problèmes d'ergonomie. Par exemple, les outils qui agissent sur le chargement des données (comme choisir le script de chargement des données) sont présents en haut à droite, mélangés avec des outils agissant sur l'animation des données ... Tout cela en sachant que l'outil de chargement de données principal se situe en haut à gauche.

Organisation de la Nouvelle Palette d'Outils

L'organisation de cette palette est primordiale car une bonne organisation de l'interface graphique permet d'utiliser facilement et intuitivement l'application et de palier aux problèmes d'encombrement de l'interface quand beaucoup d'outils sont présents. Avant de commencer l'élaboration de la nouvelle palette d'outils, j'ai créé une maquette qui permet de se donner une idée de l'organisation générale de l'application (cf Figure 4.4).

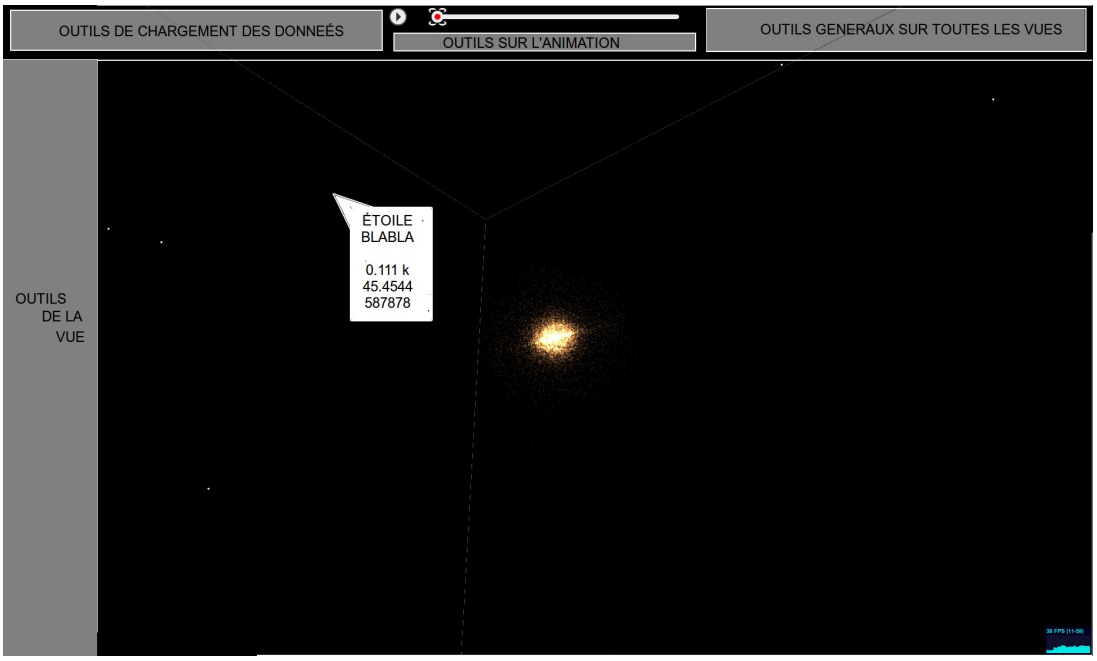


FIGURE 4.4 – Maquette de l'interface de l'application

L'interface est organisée comme suit :

Outils de chargement de données : Cette barre d'outils comporte tous les outils liés au chargement de données (script utilisé pour le chargement, choix du chargement depuis internet ou en local ...).

Outils pour l'animation : Cette barre d'outils comporte tous les outils liés à l'animation de données (vitesse de l'animation, temps de l'animation ...)

Outils généraux sur toutes les vues : Cette barre d'outils comporte les outils généraux de toute l'application (aide, crédits ...)

Outils de la vue : Cette barre d'outils comporte les outils agissant sur une et une seule vue de l'application. En effet, l'application peut comporter deux vues (en écran splité), dans ce cas, il y aura deux barres d'outils, une par vue (cf Figure 4.5)

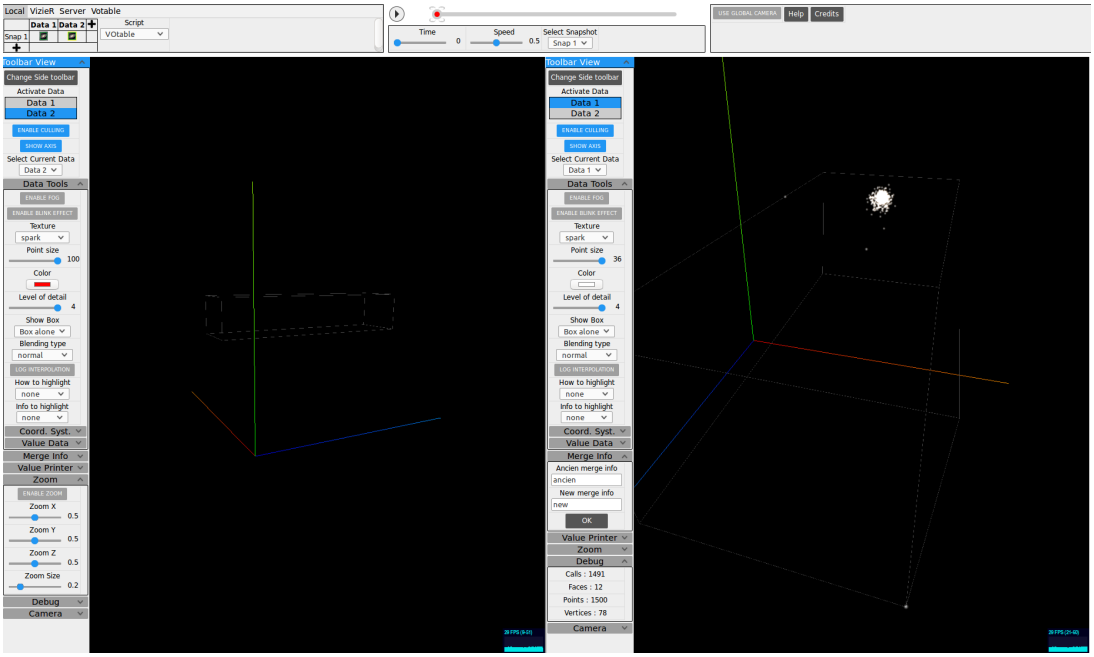


FIGURE 4.5 – Application avec deux vues

Technologie de la Nouvelle Palette d'Outils

A cause de tous les problèmes listés précédemment, j'ai décidé d'enlever complètement dat.GUI de l'application et de développer une bibliothèque créant des barres d'outils dans une application JavaScript qui répondent à tous les besoins et contraintes définis ci-dessus car je n'ai pas trouvé de bibliothèque JavaScript remplissant toutes ces conditions, en effet, je n'ai trouvé que dat.GUI qui était capable de créer des barres d'outils. A partir de ce moment là, deux choix se sont offerts à moi pour savoir quelle technologie utiliser pour la création de cette bibliothèque :

— Utiliser les langages HTML, CSS et JavaScript et créer une bibliothèque de palette d'outils sur le modèle de dat.GUI respectant tous les besoins et contraintes de l'application.

— Utiliser la bibliothèque THREE.js et créer une interface graphique comme dans beaucoup de jeux vidéos actuels, c'est-à-dire en plaquant une texture au premier plan de l'écran (qui contient la vue des outils) et créer un événement pour chaque outils.

La première solution me parue très vite la meilleure, en effet, la plupart des outils simples (comme des checkboxs, des zones de saisies de texte) existent déjà en HTML et donc il est inutile de recréer tous ces outils "à la main" (ce qui génère un gain de temps et de

performance notable). De plus, en utilisant le langage CSS, on peut gérer les problèmes d'encombrement simplement en créant des barres de défilement si trop d'outils sont affichés dans une zone donnée ou en pouvant mettre des éléments l'un à coté de l'autre horizontalement comme verticalement. La première solution est donc en tout point la meilleure pour résoudre les problématiques de l'application.

Spécification de la Bibliothèque de la Nouvelle Palette d'Outils

Voici les caractéristiques de la nouvelle palette d'outils :

— Cette palette est séparée en barres d'outils qui contiennent des outils ou des conteneurs (comme des onglets, des dossiers ...) qui contiennent eux-mêmes les outils.

— Les barres d'outils ont une dimension fixe (soit en pixel, soit en pourcentage d'occupation de l'écran). Si trop d'outils sont présents dans une barre d'outils, alors une barre de défilement se met en place ce qui permet de mettre en théorie une infinité d'outils sans avoir un encombrement de l'interface graphique.

— Les barres d'outils peuvent être cachées entièrement.

— Les barres d'outils ont un mode qui permet de les contracter et rétracter en cliquant sur un bouton. La figure 4.6 représente une barre d'outils affichée et la figure 4.7 représente une barre d'outils contractée.

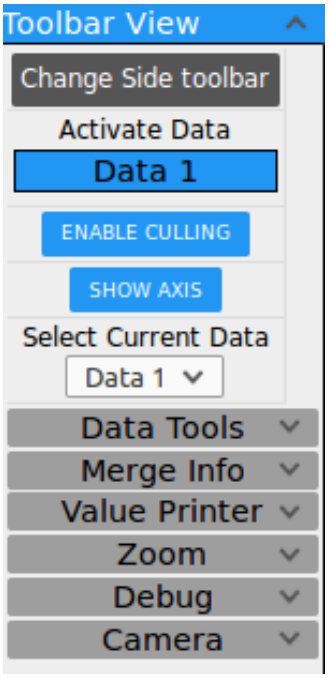


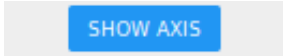
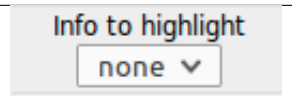
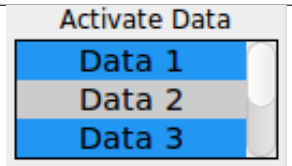
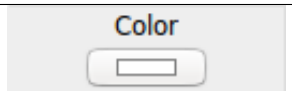
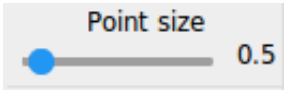
FIGURE 4.6 – Barre d'outils totalement affichée

FIGURE 4.7 – Barre d'outils contractée

— Si une barre d'outils est plus grande en hauteur qu'en largeur, alors les outils s'agencent verticalement les uns sur les autres. Par contre, si une barre d'outils

est plus grande en largeur qu'en hauteur, alors les outils s'agencent horizontalement les uns à coté des autres.

— Voici les types d'outils utilisables :

Descriptions	Outils
Saisie de texte	
Bouton simple	
Bouton ON/OFF : Ce bouton possède deux états : Un état activé (ON) et un état inactif (OFF)	
Menu déroulant : Cet outil permet de sélectionner un item dans une liste	
Outil de sélection : Cet outil permet de sélectionner un ou plusieurs items dans une liste	
Sélection de couleur	
Sélection d'un nombre : Permet de sélectionner un nombre entre une valeur minimale et maximale avec un pas particulier	
Afficheur de valeur : Permet d'afficher un nombre	

— Une barre d'outils peut contenir des conteneurs contenant eux-mêmes des outils (cela permet d'organiser et réunir les outils d'une même barre d'outils par thème et de cacher les outils qui ne sont pas utilisés par l'utilisateur), voici ces conteneurs :

Onglets : Les onglets permettent de classer thématiquement des outils et de cacher ceux qui sont des autres thèmes (cf Figure 4.8).

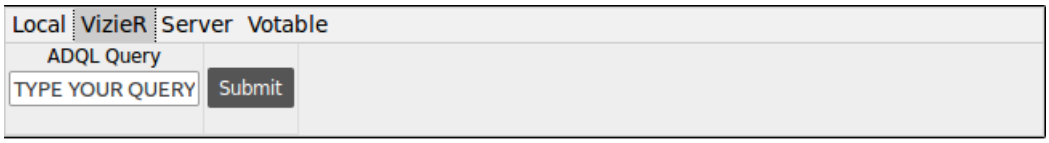


FIGURE 4.8 – Exemple d'onglet

Dossier : Un dossier contient les outils d'un même thème pour organiser les outils dans une barre d'outils. Ces dossiers peuvent être rétractés (cf Figure 4.10) et contractés afin de rendre plus clair la barre d'outils (cf Figure 4.9).

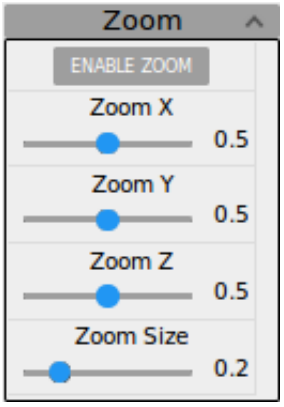


FIGURE 4.9 – Exemple de dossier

FIGURE 4.10 – Exemple de dossier contracté

4.2.3 Migration des outils dans la nouvelle palette d'outils

Migration de la barre d'outils associée à une vue

Une fois la bibliothèque créée et incluse dans le projet de l'application, il fallut inclure petit à petit les outils de l'ancienne palette dans la nouvelle. En effet, il fallut premièrement inclure dans l'application une barre d'outils (j'ai commencé par la barre d'outils associée à une vue, c'est-à-dire la barre d'outils sur le coté gauche de la vue). Une fois la barre d'outils incluse, pour chaque outil de l'interface dat.GUI, j'ai créé l'outil correspondant dans la nouvelle barre d'outils. Une fois cela effectué, j'ai créé des événements pour chaque nouvel outil et migrer les événements des anciens outils vers les nouveaux. Ensuite, j'ai migré le fait que les outils peuvent être modifiés par la programmation. Une fois tout les événements migrés, j'ai pu supprimer l'ancienne barre d'outils. Enfin, j'ai réorganisé les outils dans la barre d'outils (comme par exemple mettre tous les outils associés à une et une seule données comme la couleur des particules, la taille des particules, leurs textures ...) afin de les grouper dans des dossiers.

De plus, l'application comporte un outil de zoom qui une fois activé crée une vue dite "esclave". La vue "esclave" contient un zoom d'une zone sélectionnée dans la vue "maître" (cf Figure 4.11 où la vue maître est à droite et la vue esclave est à gauche). Or quand l'outil de zoom est activé, certains outils ne sont pas présents dans la barre d'outils de la vue esclave (entourée en orange dans la figure 4.11). Ces outils contiennent les outils de zoom et les outils agissant sur les dimensions et la position de l'outil de sélection dans la vue maître, or, il est impossible dans une vue esclave de rappeler l'outil de zoom (et donc de créer un outil de sélection), ces outils sont donc inutilisables dans la vue esclave, il ne doivent donc ne pas être affichés.

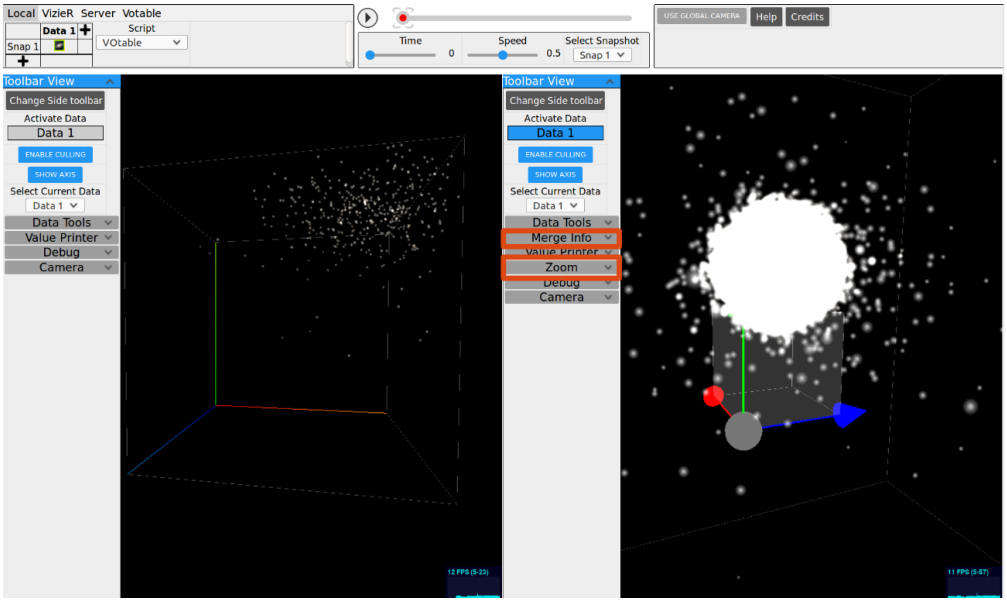


FIGURE 4.11 – Outils de zoom

Migration de l'outil Global Caméra

Il existe un outil qui permet si l'application est en mode deux vues splités (Figure 4.5) de prendre le contrôle des deux vues en même temps (les deux vues se déplacent en même temps avec la même caméra). Cet outil étant présent dans la barre d'outil associée à une vue, or, cet outil n'était pas intuitif car il fallait activer cet outil dans chacune des deux vues pour utiliser leurs fonctionnalités (cf Figure 4.12) ... Ce qui est un peu compliqué. J'ai donc migré cet outil dans les outils généraux (la barre d'outils en haut à droite) afin de rendre cet fonctionnalité facile d'utilisation (il suffit maintenant que d'un clic pour activé la fonctionnalité, cf Figure 4.13).

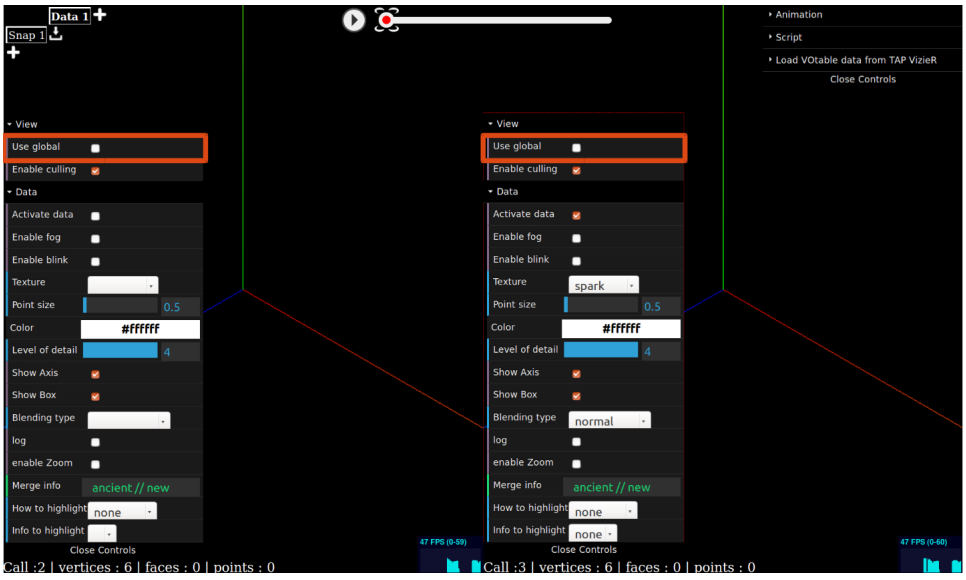


FIGURE 4.12 – Nouveaux outils de contrôle des deux vues

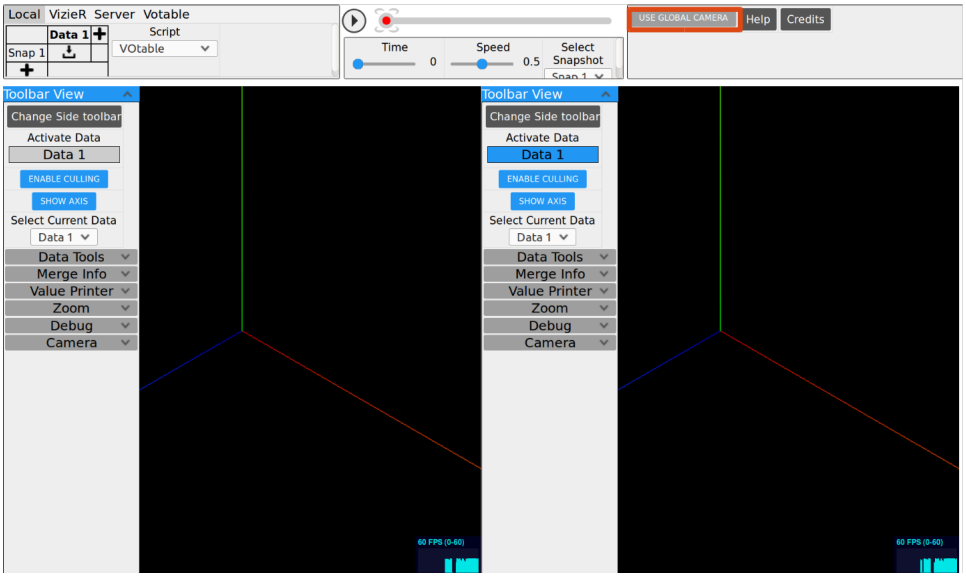


FIGURE 4.13 – Outils de contrôle des deux vues actuellement

Migration de l'outil "Mergeinfo"

Dans la version de l'application d'origine, un champ nommé *MergeInfo* était présent (cf 4.14) Ce champ servait à modifier le nom d'un attribut pour données VOTable. En effet, il est possible qu'un même attribut ait un nom différent dans deux données VOTable. Cet outil servait donc à convertir le nom d'un attribut en un nouveau nom afin que si on utilise deux données VOTable, ces deux attributs aient le même nom. Or l'utilisation de cet outil était difficile, en effet, il fallait écrire dans le champ une chaîne de caractères de la forme "*Ancien Nom//Nouveau Nom*" ... ce qui pouvait occasionner des erreurs et des difficultés d'utilisation. J'ai donc crée un dossier *Merge Info* qui contient un champ *Ancien merge info*, un champ *New merge info* ainsi qu'un bouton de confirmation (cf 4.15) ce qui rend l'utilisation de cette fonctionnalité bien plus aisée.

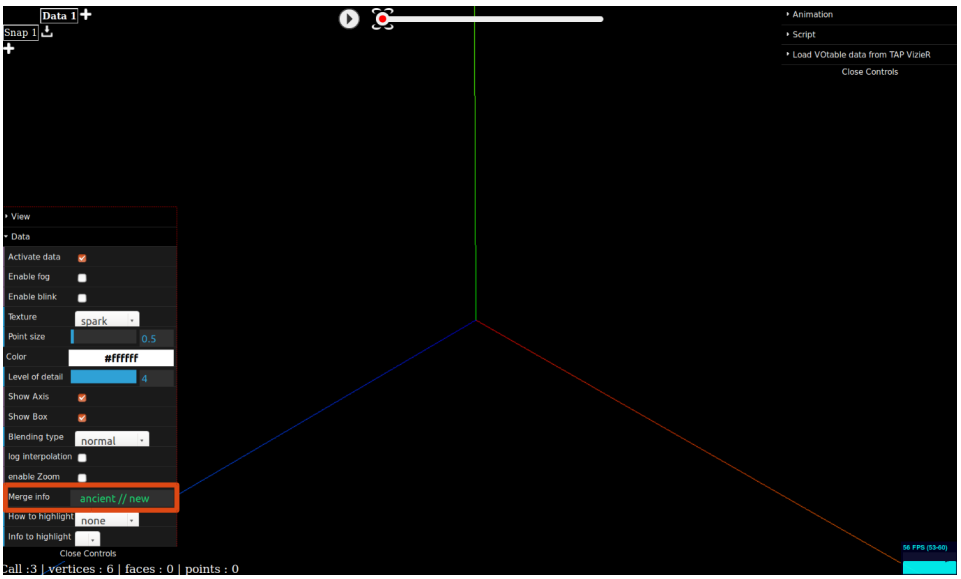


FIGURE 4.14 – Champ Merge info dans l'ancienne interface

FIGURE 4.15 – Dossier Merge Info

Migration du reste des outils

Une fois la migration des outils précédemment cités effectuée, il fallut effectuer la migration des outils de chargement des données et d'animation.

Outils de chargement des données : Ces outils étaient les plus difficiles à organiser, en effet, l'application possédait à l'origine le moyen d'importer des données (depuis le disque dur ou depuis le serveur VizieR) et de plus, le format de données VOTable peut nécessiter l'initialisation dans l'application du nom des champs utilisés dans les données à charger or, ce format de données est utilisé aussi bien en local que depuis le serveur VizieR. J'ai donc choisi d'organiser tous ces outils avec des onglets thématiques, c'est-à-dire qu'il y a un onglet par moyen de chargement (en local ou depuis VizieR) et un onglet pour les données VOTable (cf Figure 4.16). *NB : L'onglet Server a été ajouté après la migration des outils.* Pour le chargement de données en local, j'ai choisi de garder le tableau de

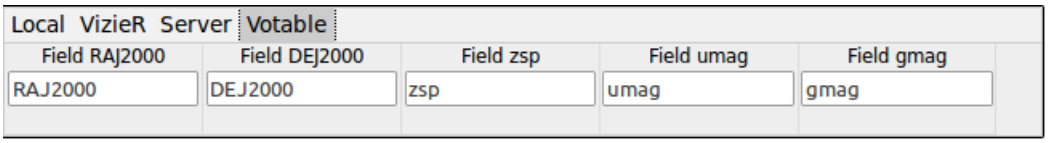


FIGURE 4.16 – Barre d'outils de chargement des données dans l'onglet VOTable

chargement des données qui était à l'origine tout en haut à gauche et de le mettre dans l'onglet Local (cf Figure 4.17).

FIGURE 4.17 – Onglet Local

Outils d'animation : J'ai choisi de garder la barre d'animation (qui est en haut au centre de l'interface) car elle donne pleinement satisfaction. J'ai donc migré de l'interface dat.GUI l'outil de sélection du temps et de la vitesse d'animation. J'ai effectué la migration sur le même principe que pour la barre d'outils des vues.

Informations de débogage : Ces informations étaient à l'origine affichées en bas à gauche de l'écran sous la barre d'outils (cf Figure 4.18). Pour des raisons esthétiques et d'encombrement de l'interface, ces informations sont maintenant affichées dans un dossier dans la barre d'outils de la vue (cf Figure 4.19).

FIGURE 4.18 – Information de débogage avant

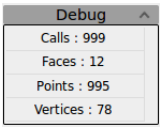


FIGURE 4.19 – Information de débogage après

4.2.4 Outils ajoutés à l'application

Une fois tous les outils migrés dans la nouvelle palette d'outils, j'ai pu ajouter des outils à la palette, ces outils sont issues des demandes et des besoins des utilisateurs de l'application ainsi que d'André Schaaff, Nicolas Adam et Jérôme Desroziers. Voici les outils ajoutés :

— Une bouton d'aide et de crédits (mise en place dans la barre d'outils généraux). Le bouton d'aide ouvre une fenêtre contenant les contrôles de l'application (cf Figure 4.20). Le bouton des crédits affiche les noms des contributeur de l'application dans une fenêtre.

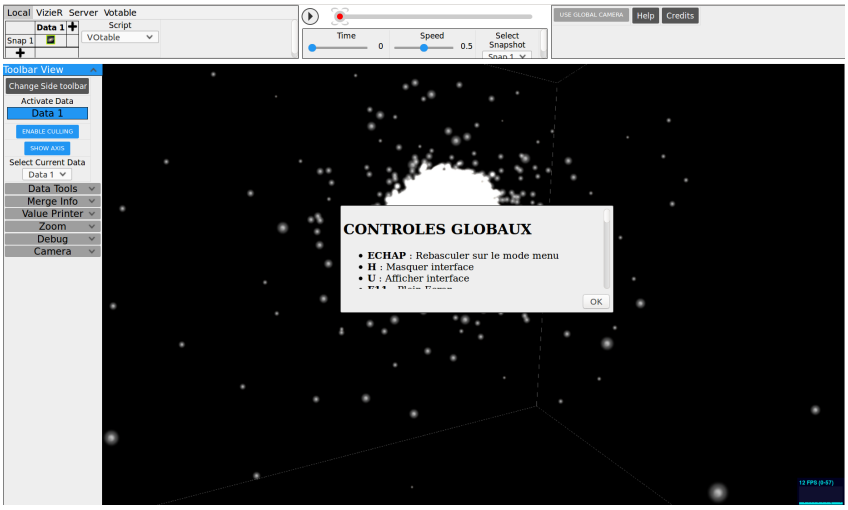


FIGURE 4.20 – Fenêtre d'aide

— Création d'un onglet Server dans la barre d'outils de chargement des données. Cet onglet a été ajouté pour lancer le chargement de données depuis un serveur dans l'application (Stage de Nicolas Adam) (cf Figure 4.21).

— Affichage de la position et de l'angle de la caméra dans un dossier contenu dans la barre d'outils de la vue. De plus, j'ai ajouté la possibilité de sauvegarder des positions de caméra et de pouvoir revenir à ces positions précédemment sauvegardées (cf Figure 4.22).

— J'ai ajouté la possibilité de créer des segments entre deux (ou plusieurs) particules. Pour créer un segment entre deux particules, il suffit de cliquer successivement sur

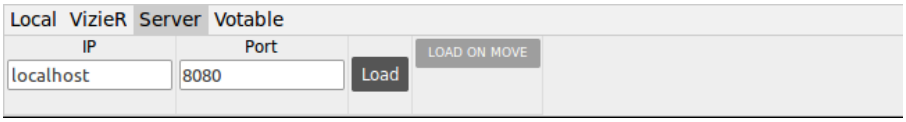


FIGURE 4.21 – Onglet Server

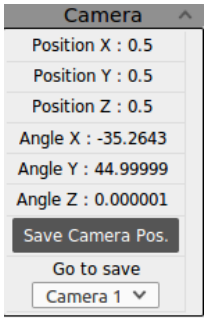


FIGURE 4.22 – Dossier Camera

les deux étoiles. Il est possible de créer plus de segments en cliquant sur d'autres étoiles en maintenant la touche SHIFT. De plus, j'ai ajouté un outil qui permet d'avoir la longueur de ces segments, ce qui permet d'avoir la distance entre deux étoiles dans l'unité des données chargées (cf Figure 4.23).

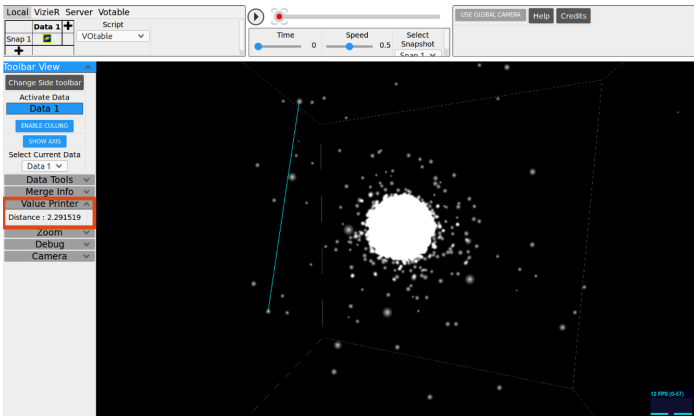


FIGURE 4.23 – Distance entre deux étoiles

— J'ai ajouté à l'outil de zoom des outils de réglages (pour plus de précision au sujet de cet outil, relire le second paragraphe de la partie 4.2.3). En effet, avant mon stage, les dimensions et la position de l'outil de zoom ne pouvait qu'être réglées en déplaçant l'outil de zoom dans la scène 3D par le biais de "cliquer-déposer" avec la souris. Or, on ne pouvait donc pas régler avec précision la position et la dimension de l'outil de zoom (ce qui rend difficile la visualisation d'une zone précise dans une jeu de données). J'ai donc ajouté des outils de type jauqe (cf Figure 4.24) qui permettent de sélectionner et d'afficher la position dans l'espace du centre de l'outil de zone et qui permet de sélectionner et visualiser la dimension de l'outil de zone.

— Jérôme Desroziers a développé des outils permettant de modifier le système de coor-

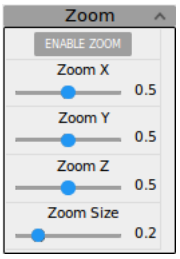


FIGURE 4.24 – Dossier de l'outil de zoom

données. En effet, les données contiennent les positions des particules ainsi que des propriétés physiques (comme des énergies ou des vitesses). Jérôme a donc développé un système permettant d'associer un axe à l'un de ces champs (par exemple, d'avoir sur l'axe X l'énergie des particules). Par défaut, on affiche selon l'axe X la position en X des particules, selon l'axe Y la position en Y des particules et selon l'axe Z la position en Z des particules. Sur la demande de Nicolas Deparis (thésard à l'observatoire), j'ai ajouté le moyen de mettre un axe à l'échelle logarithmique (il est très utile en science, particulièrement en physique d'avoir des échelles logarithmiques pour visualisé des phénomènes). On appelle échelle logarithmique l'application d'une fonction tel que :

$$f(x) = \begin{cases} log(x) & \text{si x} > 0 \\ 0 & \text{si x} = 0 \\ -log(-x) & \text{si x} < 0 \end{cases} \tag{4.1}$$

J'ai donc du ajouté un outil de choix entre coordonnées normales et logarithmiques pour chaque axes (cf Figure 4.25).

FIGURE 4.25 – Outils de changement de coordonnées

— J'ai ajouté la possibilité d'afficher sur la boîte englobante des données des graduations et des grilles sur chaque faces de la boîte. Avant mon stage, il était déjà possible d'afficher ou non les boîtes englobantes. A la demande de Nicolas Deparis, j'ai ajouté la possibilité d'afficher dix graduations sur chaque arêtes des boîtes englobantes afin de donner un point de repère au niveau de la répartition des particules dans la boîte englobante (cf figure 4.26). J'ai aussi trouvé intéressant d'ajouter la possibilité d'afficher une grille 10 x 10 sur chaque face des boîtes englobantes (cf figure 4.27). Le type de boîte englobante est choisi par le biais d'un menu déroulant. De plus, l'échelle des graduations (c'est-à-dire la longueur réelle d'une graduation) selon les

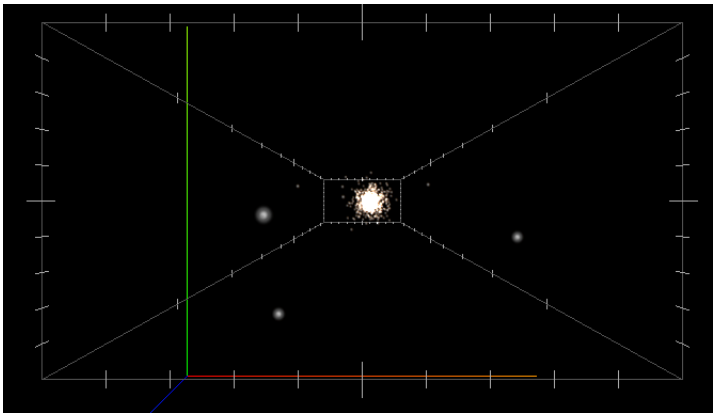


FIGURE 4.26 – Graduation sur la boîte englobante

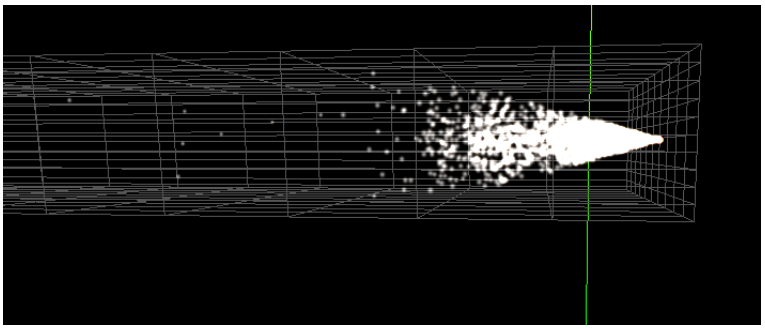


FIGURE 4.27 – Grille sur la boîte englobante

axes X, Y et Z sont affichées dans la section *Value Data* de la barre d'outils de la vue (cf figure 4.28) Cette section est incluse dans la section spécifique à la donnée courante car comme la boîte englobante est spécifique à une donnée, l'échelle de la boîte englobante est spécifique à une donnée.

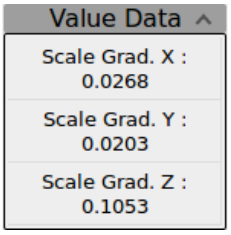


FIGURE 4.28 – Échelles de la boîte englobante de la donnée courante

4.3 Conception Détaillée

4.3.1 Affichage des informations d'une particule

L'affichage des informations d'une particule ce fait par le biais d'un objet 3D dans la scène THREE.js. Plus précisément, cet objet est un sprite (ou billboard), c'est-à-dire un objet plat et toujours parallèle au plan focal de la caméra. La texture de ce sprite

est générée par le biais d'un canvas HTML (dans lequel on écrit les informations de la particule ainsi que le contour des informations) qui sert à créer une image de format JPEG. Cette image JPEG est ensuite chargée par le biais de THREE.js en temps que texture qui sera elle-même affichée dans le sprite.

Les informations affichées se doivent d'avoir toujours la même taille pour le confort d'utilisation de l'application, quelque soit la position de la particule et quelque soit la distance entre la caméra et la particule. Or, ces informations étant affichées dans sprite avec une position fixée à la position de la particule, j'ai dû mettre en place en système de redimensionnement du sprite en fonction de la distance entre la caméra et la particule afin d'avoir la taille des informations à l'écran constante, quelque soit les déplacements de caméra dans la scène et la position de la particule. L'échelle d'affichage de ce sprite est mise à jour à chaque rendu de la scène THREE.js en fonction de la distance entre le sprite et la camera avec la formule suivante (avec S la position cartésienne du sprite et C la position cartésienne de la caméra) :

$$echelle = ||S - C|| \tag{4.2}$$

4.3.2 Création de la palette d'outils

Esthétique de la barre d'outils

L'esthétique d'une application est essentielle pour le confort et la facilité d'utilisation de l'application par l'utilisateur. Cette esthétique est traduite par une cohérence graphique des outils (en utilisant par exemple un nombre réduit de couleurs qui sont dans l'application des dégradées de gris et des dégradées de bleu). De plus, tous les outils de l'application ont une cohérence structurelle, c'est-à-dire que la grande majorité des outils, ces outils sont contenus dans un rectangle avec en haut un titre et en bas l'outil en lui-même. Pour une minorité des outils (les boutons et les "switchs"), le libellé de l'outil est écrit dans le bouton ou dans le switch.

Le design des outils est inspiré du design *Material* utilisé par exemple sur les sites de l'entreprise Google ou sur les systèmes Android. Je me suis inspiré de ce design pour créer le style et la forme des outils car ce desiqn est très répandu sur les plateformes mobiles et web et donc cela permet à l'utilisateur d'avoir une prise en main facile des outils car il en a déjà que leurs sont similaires.

Le style de ces outils a été fait en pure CSS3 sans bibliothèque externe (comme Bootstrap) car je n'ai trouvé aucune bibliothèque CSS pouvant servir à styliser des outils.

4.3.3 Migration des outils

Réorganisation du code-source

L'ajout de la palette d'outils a nécessité une réorganisation du code. En effet, avant mon stage, le code-source mélangeait l'affichage des outils en eux-même et les algorithmes utilisés par les outils. Cette solution n'était pas satisfaisante car cela apportait une confusion dans le code, la code source était donc peu claire. C'est pourquoi (en même temps que j'ai fait la migration de l'ancienne palette d'outils vers la nouvelle) j'ai choisi de séparer la partie affichage des outils et interaction avec l'utilisateur de la partie algorithmique des outils. Plus précisément, j'ai mis tout ce qui est au sujet de la création des barre

d'outils ainsi que les événements associés dans un fichier JavaScript et les algorithmes des outils dans un autre fichier. Par exemple, pour la barre d'outils agissant sur une vue de l'application, j'ai mis la partie affichage et événement de cette barre d'outils dans le fichier **GUIView.js** et les algorithmes associés à ces outils dans le fichier **View.js** (ce fichier agit aussi sur la création d'une vue et sur les données affichées dans cette vue).

4.4 Validation

La validation du projet s'est effectuée tout au long du stage. Mes travaux ont été suivis par André Schaaff régulièrement (au moins une fois par semaine) afin de vérifier si mon travail respectait les exigences demandées et ainsi valider les modifications apportées à l'application au fur et à mesure de l'avancée de mes travaux. De plus, à la fin du stage, une réunion à été effectuée avec entre autre Nicolas Deparis, thésard et l'un des principaux utilisateurs de l'application. Lors de cette réunion, j'ai présenté toutes les nouvelles fonctionnalités de l'application.

Une documentation technique (présente dans l'annexe de ce rapport) présente tous les outils implémentés dans l'application et indique comment ajouter des outils dans les barres d'outils afin que mon travail puisse être repris facilement dans la suite du projet.

De plus, ce rapport et la documentation sont ajoutés dans le dépôt GitLab avec les autres rapports de stage et dans l'intranet afin qu'ils soient facilement accessibles aux développeurs suivants.

4.5 Résultats

La palette d'outils ainsi que les outils ajoutés fonctionnent. André Schaaff ainsi que les thésards sont satisfaits des modifications effectuées sur l'application. De plus, ces modifications respectent les demandes et contraintes énoncées auparavant.

Chapitre 5

Conclusion

L'ajout d'une palette d'outils à l'application de visualisation 3D a donc été mené à bien, les modifications effectuées sur l'application sont fonctionnelles. De plus, les modifications respectent les demandes et contraintes énoncées tout au long du stage par les différents acteurs du projet. L'application est donc en état fonctionnelle et est d'ores et déjà utilisée à l'observatoire pour des travaux scientifique.

Le développement de l'application étant voué à continuer, voici deux axes de développement possible :

— Un axe de développement majeur est l'amélioration de l'utilisation de l'application sur Google Cardboard (c'est-à-dire comment piloter l'application sur Cardboard sans manette). En ayant réfléchit un peu à la question, je pense que le contrôle complet de l'application par la Cardboard est très complexe (mais pas impossible). Le plus simple me semble de faire un pilotage de l'application Cardboard par une application principale sur PC gérée par une personne tierce (comme un animateur). L'utilisateur pourrait avoir un contrôle limité de l'application (comme tourner la tête pour voir les particules autour de lui-même) et avoir accès par le biais de l'animateur à toutes les fonctionnalités de l'application.

— Un autre axe de développement serait de faire une refactorisation complète de l'application. En effet, on a remarqué avec Jérôme Desroziers et Nicolas Adam que le code source de l'application n'est pas forcement très claire et que comprendre et modifier certaines parties du code comme le chargement des données ou l'utilisation des données dans l'application est très difficile. Une refactorisation de ces parties du code source pourrait permettre un gain de lisibilité et de compréhension dans le code source et ainsi permettre de rendre les modifications de l'application beaucoup plus simple.

Ce stage m'a permit de découvrir le monde du développement d'application dans un contexte de recherche scientifique et plus particulièrement dans le domaine de l'astronomie. J'ai donc pu découvrir le fonctionnement d'un observatoire astronomique ainsi que le fonctionnement d'un laboratoire de recherche du point de vue d'un informaticien et ainsi connaître le rôle de plus en plus important de l'informatique et du développement informatique dans le domaine de la recherche. De plus, ce stage m'a permit de travailler en équipe avec un même dépôt de sources avec toute les contraintes que cela implique. Chaque membre de l'équipe ont donc dû acquérir au fur et à mesure du stage une bonne coordination, une connaissance des travaux des autres membres de l'équipe ainsi qu'une bonne communication. Tout cela m'a permit d'acquérir des automatismes qui me seront très utiles dans ma vie professionnelle.

Chapitre 6

Glossaire

THREE.js : Bibliothèque JavaScript pour créer des scènes 3D en WebGL, CSS3D et SVG dans un navigateur web sans plugin.
`https://fr.wikipedia.org/wiki/Three.js`

Webstorm : Environnement de développement crée par JetBrains qui prend en charge les langages Web (HTML, CSS, JavaScript, Node.Js ...). A noter qu'une licence pour étudiant est disponible gratuitement.
`https://www.jetbrains.com/webstorm/`

dat.GUI : Bibliothèque qui permet de créer une interface graphique servant à modifier des valeurs de variables JavaScript.
`https://workshop.chromeexperiments.com/examples/gui/#1--Basic-Usage`

Design Material : Type de design utilisé par Google et sur Android.
`https://material.google.com/`

Table des figures

Annexe A

Documentation Technique de l'interface graphique

Cette documentation a été écrite par moi-même et elle décrit le rôle de chaque outils de l'application ainsi que le fonctionnement de chaque type d'outils afin de pouvoir en ajouter facilement de nouveaux.

Documentation Technique des Barres d'Outils de l'Application de Visualisation 3D

Thibault Bouchard

11 août 2016

Table des matières

Chapitre 1

Introduction

Au début du développement de l'application, on utilisait les barres d'outils dat.GUI. Or, ce type de barre d'outils qui peut être suffisant pour des applications de faible envergure ne s'est pas avéré suffisant pour l'application de visualisation 3D. En effet, l'application nécessite un grand nombre d'outils ainsi qu'une bonne ergonomie (c'est-à-dire une meilleur organisation des outils) afin de rendre l'utilisation de l'application assez simple. De plus, Dat.GUI ne répondait pas aux besoins techniques de l'application telle que de modifier l'état d'un bouton depuis le code (par exemple, cocher ou décocher une checkbox), en effet, dat.GUI ne permet que de modifier l'état d'un bouton par l'utilisateur.

Ce document décrira d'une part les fonctionnalités des outils présents dans l'application et d'autre part une description technique des outils afin de savoir par exemple comment ajouter un outil dans une barre d'outils.

Chapitre 2

Description des barres d'outils dans l'application

2.1 Vue d'ensemble de l'interface de l'application

Voici une vue d'ensemble de l'application :

FIGURE 2.1 – Vue d'ensemble de l'interface de l'application

Voici à quoi correspond chaque zone de l'interface :

A : Zone contenant les barres d'outils agissant sur toute l'application.

B : Zone contenant les barres d'outils agissant sur une vue.

1 : Barre d'outils s'occupant du chargement des données dans l'application.

2 : Barre d'outils modifiant les paramètres d'animation des données.

3 : Barre d'outils contenant les outils généraux de l'application.

4 : Outil s'occupant de l'animation des données.

2.1.1 Zone contenant les barres d'outils agissant sur toute l'application

Barre d'outils s'occupant du chargement des données dans l'application

Cette barre d'outils est découpée en 4 onglets, voici une description de ces onglets :

Onglet "Local" : L'onglet local contient les outils de chargement des données depuis le disque dur, il contient deux outils :

- L'outil de chargement des données qui permet de charger une donnée dans un slot *data* et *snapshot* particulier. Il est possible d'ajouter un slot *data* en cliquant sur le signe plus de la ligne où il y a écrit *Data* et il est possible d'ajouter un slot *snapshot* en cliquant sur le signe plus de la colonne où il y a écrit *snapshot*. Pour allouer une donnée dans le slot *[Data i ; Snapshot j]*, il suffit de cliquer sur le symbole dans la colonne *Data i* et la ligne *Snapshot j*. Une fois la donnée chargée, il est possible d'afficher le nom du fichier chargé en laissant le curseur de la souris sur le slot de la donnée.
- L'outil de sélection de script permet de sélectionner le script qui va lire les données que l'on va charger en local après le sélection de ce script.

FIGURE 2.2 – Vue de l'onglet "Local"

Onglet "VizieR" : L'onglet VizieR contient les outils de chargement des données depuis VizieR, il contient deux outils :

- L'outil **ADQL Query** permettant d'écrire une requête VizieR.
- L'outil **Submit** permettant d'envoyer la requête VizieR.

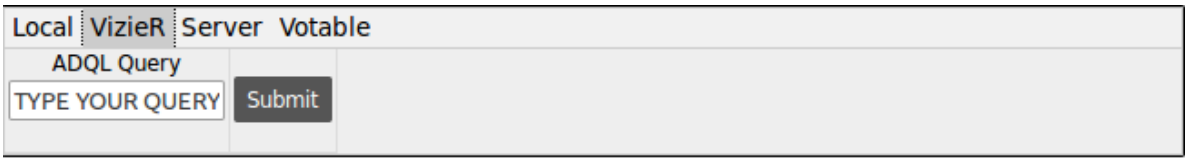


FIGURE 2.3 – Vue de l'onglet "VizieR"

Onglet "Server" : L'onglet Server contient les outils de chargement des données depuis un serveur, il contient trois outils :

- L'outil **IP** permet d'écrire l'IP du serveur (*ATTENTION : Pour le moment (Aout 2016), ce champ n'est pas utilisé*).
- L'outil **Port** permet d'envoyer le port du serveur (*ATTENTION : Pour le moment (Aout 2016), ce champ n'est pas utilisé*).
- L'outil **Load** permet de charger les données depuis le serveur.
- L'outil **Load on Move** permet de dire si oui ou non on veut charger depuis le serveur les données quand la caméra bouge.

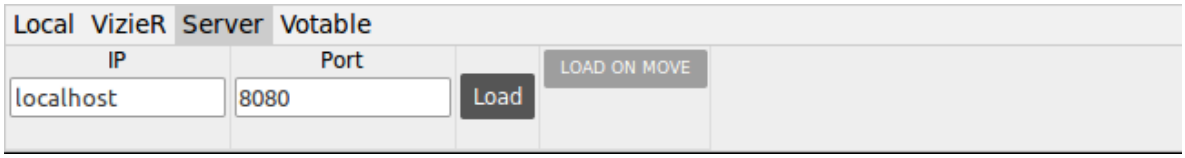


FIGURE 2.4 – Vue de l'onglet "Server"

Onglet "VoTable" : L'onglet VoTable contient les outils de chargement des données VoTable, il contient cinq outils :

- L'outil **Field RAJ2000** permettant de modifier la dénomination du champ *RAJ2000* pour le chargement des données VoTable.
- L'outil **Field DEJ2000** permettant de modifier la dénomination du champ *DEJ2000* pour le chargement des données VoTable.
- L'outil **Field zsp** permettant de modifier la dénomination du champ *zsp* pour le chargement des données VoTable.
- L'outil **Field umag** permettant de modifier la dénomination du champ *umag* pour le chargement des données VoTable.
- L'outil **Field gmag** permettant de modifier la dénomination du champ *gmag* pour le chargement des données VoTable.

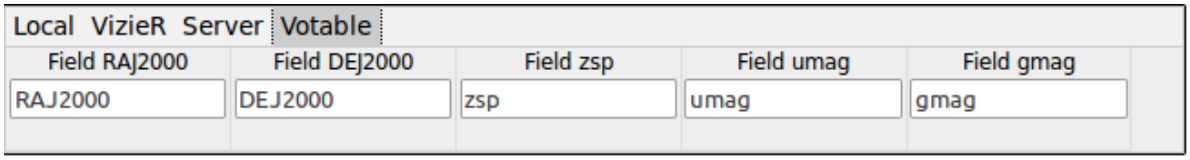


FIGURE 2.5 – Vue de l'onglet "VoTable"

Barre d'outils modifiant les paramètres d'animation des données

Cette barre d'outils permet de modifier les paramètres d'animation des données. Elle contient trois outils :

- L'outil **Time** qui permet de fixer l'état du temps de l'animation.
- L'outil **Speed** qui permet de fixer la vitesse de l'animation.
- L'outil **Select Snapshot** qui permet de fixer quelle snapshot afficher.

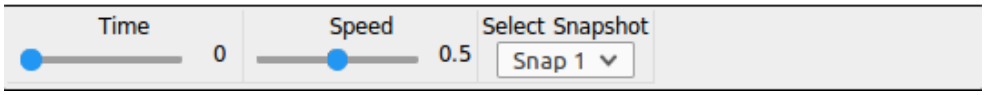


FIGURE 2.6 – Vue de la barre d'outils d'animation

Barre d'outils contenant les outils généraux de l'application

Cette barre d'outils contient les outils généraux de l'application. Elle contient trois outils :

- L'outil **Use Global Camera** qui permet de pouvoir contrôler les caméras de toutes les vues en mode de vue Zoom ou Multivue en même temps.
- L'outil **Help** qui permet d'afficher une aide. Cette aide s'affiche dans une fenêtre au centre de l'application.
- L'outil **Credits** qui permet d'afficher les noms des contributeurs de l'application.

FIGURE 2.7 – Vue de la barre d'outils général

Outil s'occupant de l'animation des données

Cet outil permet de lancer une animation des données quand celle-ci comporte plusieurs snapshots. Le bouton **Play** permet de lancer une animation. Cliquer sur la barre grise permet d'aller à un temps de l'animation précis. Pour plus de détail au sujet de cette barre, il faut se référer au rapport de stage de *Pierre Lespingal* et *Nicolas Buecher*.

FIGURE 2.8 – Vue de l'outil d'animation

2.1.2 Zone contenant les barres d'outils agissant sur une vue

Cette barre d'outils agit sur une vue. Si l'application utilise le zoom ou si elle est en mode multivue, il y aura une barre d'outils pour chaque vue. Il est possible de cacher ces barres d'outils en cliquant en haut de celle-ci sur

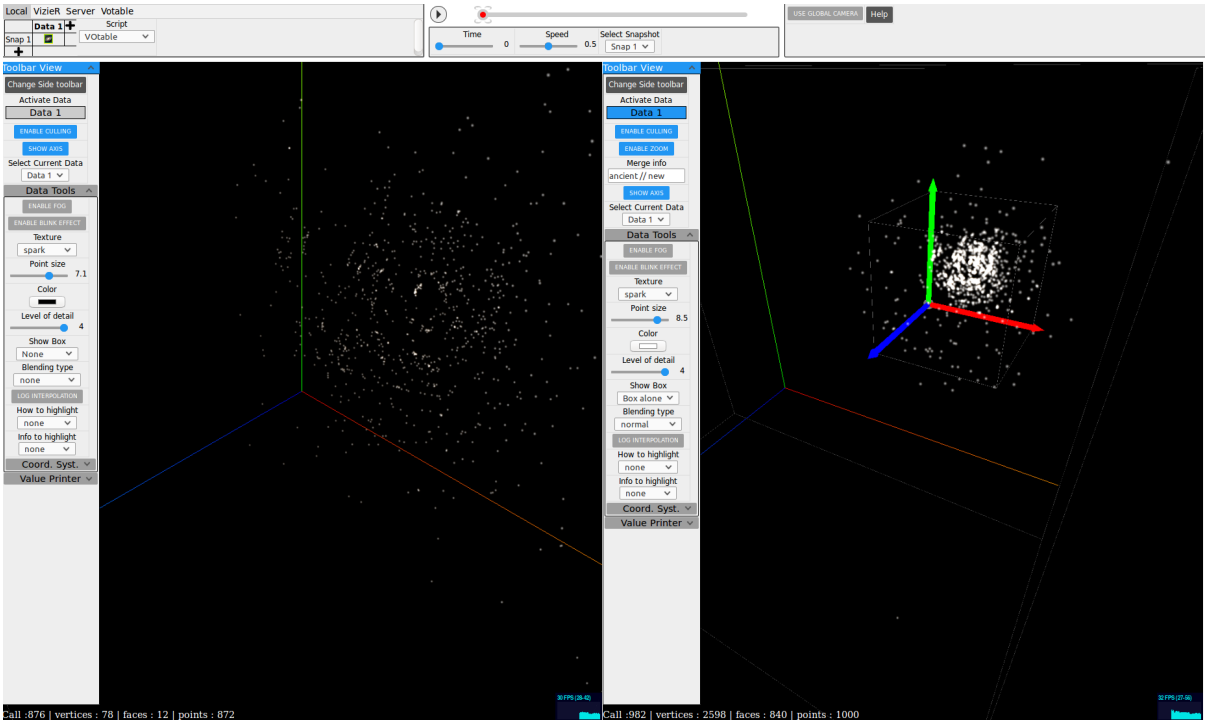


FIGURE 2.9 – L'application en mode de zoom (l'interface est pratiquement la mode en mode multivue)

Toolbar View. Une fois caché, un nouveau clic sur celle-ci permet de les ré-afficher.

Ces barres d'outils sont découpées en plusieurs parties :

Outils généraux de la vue

Ces outils se situent en haut de la barre d'outils. Ils sont toujours affichés dans la barre d'outils. Voici une description de ces outils :

- L'outil **Change Side Toolbar** permet de changer la position de la barre d'outils, quand la barre d'outils est à gauche, un clic permet de mettre la barre d'outils à droite et si la barre d'outils est à droite, un clic permet de mettre la barre d'outils à gauche.
- L'outil **Activate Data** permet de choisir les datas à afficher dans la vue. Toutes les datas sont affichées dans l'outil, pour afficher une data, il suffit que la couleur de cette data dans l'outil soit bleue. Si la data n'est pas affichée, alors sa couleur dans l'outil sera grise.
- L'outil **Enable Culling** sert à utiliser le frustum culling
- L'outil **Show Axis** permet d'afficher ou non les axes dans la vue.
- L'outil **Select Current Data** permet de sélectionner la data courante de la vue, c'est-à-dire la data sur laquelle les outils spécifique à une data de la barre d'outils agit.

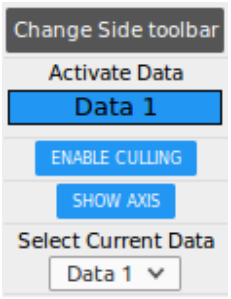


FIGURE 2.10 – Outils généraux d'une vue

Outils spécifiques à une Data

Ces outils ce situent dans une section nommée **Data Tools**. Il est possible de ne pas afficher ces outils en cliquant sur le titre de la section et de les ré-afficher en effectuant la même action. Ces outils agissent spécifiquement sur une data. Il est possible de choisir la data courante en la choisissant dans l'outil **Select Current Data**. Quand on modifie la data courante, les outils de cette section se mettent à jour (en effet, on garde en mémoire l'état de ces outils pour chaque data). Voici les outils qui composent cette section :

- L'outil **Enable Fog** permet d'activer le brouillard (Plus une particule est loin, moins elle sera visible).
- L'outil **Enable Blink Effect** permet de faire scintiller les particules.
- L'outil **Texture** permet de choisir la texture des particules.
- L'outil **Point Size** permet de modifier la taille des particules.
- L'outil **Color** permet de modifier la couleur des particules.
- L'outil **Level of Detail** permet de modifier le niveau de détail affiché dans la vue.
- L'outil **Show Box** permet de choisir le type de boite englobante de la data. Il est possible de ne pas afficher de boite, d'afficher une boite englobante simple, d'afficher une boite englobante avec 10 graduations sur chaque arête et d'afficher une boite englobante avec une grille 10 x 10 sur les faces de la boite.
- L'outil **Blending Type** permet de choisir le type de blending de la data.
- L'outil **Log Interpolation** permet d'avoir une interpolation logarithmique des couleurs quand on met en valeur un attribut d'une donnée.
- L'outil **How to Highlight** permet de choisir comment mettre en valeur un attribut de la data courante.
- L'outil **Info to Highlight** permet de choisir quel attribut de la data courante mettre en valeur.
- La section **Coord. Syst.** permet d'agir sur le système de coordonnées de la data courante. On peut donc afficher la data le long d'un axe en fonction d'un attribut de la data (d'où les champs **X**,**Y** et **Z**. Le bouton **OK** permet d'appliquer les modifications du système de coordonnées sur la data courante. Les champs **Log Inter. X**, **Log Inter. Y** et **Log Inter. Z** permettent de choisir si on veux avoir des coordonnées sur une échelle logarithmique ou non selon les axes X,Y et Z. On appelle *échelle logarithmique* l'application d'un

fonction tel que :

$$f(x) = \begin{cases} log(x) & \text{si x} > 0 \\ 0 & \text{si x} = 0 \\ -log(-x) & \text{si x} < 0 \end{cases} \tag{2.1}$$

— La section **Value Data** permet de voir des données appartenant à la vue. On affiche les échelles des données le long des axes X,Y,Z d'un graduation de la boite englobante de la donnée. Ces échelles sont calculées en fonction des données réelles (des données chargés en mémoire) et non des données affichées dans la vue (qui ont été normalisées afin d'être contenues dans un cube de 1 x 1 x 1).

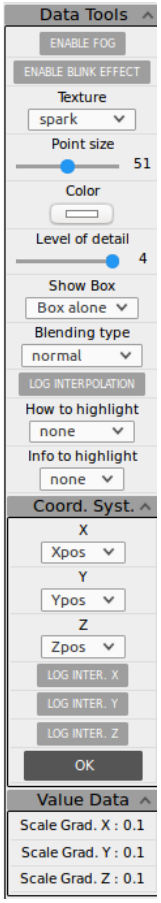


FIGURE 2.11 – Outils spécifique à une data

Outils de changement de nom d'attribut dans les données

Les outils présents dans cette section servent à modifier le nom d'un attribut pour données VoTable. En effet, il est possible qu'un même attribut ait un nom différent dans deux données VoTable. Cette section d'outils sert donc à convertir le nom d'un attribut en un nouveau nom afin que si on utilise deux données VoTable, ces deux attributs aient le même nom (et donc par exemple, on peut utiliser l'outil *Info to Highlight* sur les deux données et non sur qu'une seule donnée) (pour plus d'info, cf Rapport de stage de Jérôme Desroziers p27-28-29). *(ATTENTION : Si la vue est elle-même la vue de l'outil de zoom, cet outil ne sera pas affiché).*

Voici une description des trois outils de cet section :
— L'outil **Ancien Merge Info** est le nom de l'attribut à remplacer.
— L'outil **New Merge Info** est le nouveau nom de l'attribut.
— L'outil **OK** exécute les modifications de nom d'attribut.

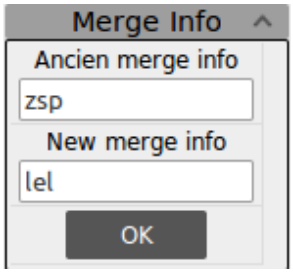


FIGURE 2.12 – Outils de changement de nom d'attribut dans les données

Outils d'affichage

Cette section contient les outils d'affichage de valeur de l'application. Voici les outils associés à cette section :

— L'afficheur **Distance** qui affiche la distance entre deux particules sélectionnés dans la vue. Lors de la sélection (avec la souris) des particules, il est possible en appuyant sur le clavier sur **SHIFT** d'ajouter des particules à la sélection. Dans ce cas, la valeur affichée et la distance de chemin entre tous le points sélectionnés. La distance affiché est calculé en fonction des données réelles (des données chargés en mémoire) et non des données affichées dans la vue (qui ont été normalisé afin d'être contenu dans un cube de 1 x 1 x 1). *(ATTENTION : La valeur affiché n'est pas très précise (avec des tests, elle n'est précise qu'à 5 ou 6 décimales), il ne faut donc pas utiliser cet outil pour avoir des distances exacts, par contre, cet outil peut être utilisé pour avoir des ordres de grandeur ou pour comparer des distances.)*

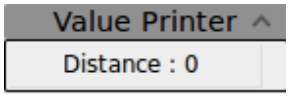


FIGURE 2.13 – Outils d'affichage de valeur d'une vue

Outils de zoom

Cette section contient les paramètres de l'outil de zoom. *(ATTENTION : Si la vue est elle-même la vue de l'outil de zoom, ces outils ne seront pas affichés).*

— L'outil **Enable Zoom** permet d'utiliser l'outil de zoom
— Les outils **Zoom X**, **Zoom Y** et **Zoom Z** permettent de régler la position du centre de l'outil de zoom.
— L'outil **Zoom Size** permet de régler la taille de l'outil de zoom.

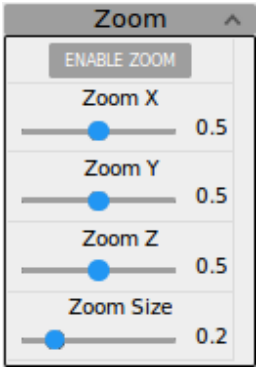


FIGURE 2.14 – Outils de zoom d'une vue

Outils d'affichage des infos de debug

Cette section contient un affichage des données de debug, c'est-à-dire le nombre de primitives affiché dans la scène THREE.js.

— L'afficheur **Calls** correspond au nombre d'appel de la fonction THREE.js *drawCalls*.
— L'afficheur **Faces** correspond au nombre de polygones affichés dans la scène THREE.js.
— L'afficheur **Points** correspond au nombre de particules affichées dans la scène THREE.js.
— L'afficheur **Vertices** correspond au nombre d'appel de vertices contenus dans la scène THREE.js.

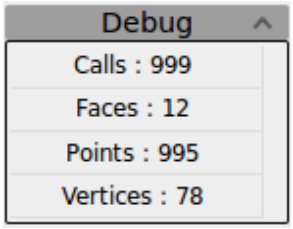


FIGURE 2.15 – Outils d'affichage des infos de debug

Outils d'affichage des infos de la caméra

— Les afficheurs **Position X**, **Position Y** et **Position Z** correspondent à la position de la caméra dans la scène THREE.js.
— Les afficheurs **Angle X**, **Angle Y** et **Angle Z** correspondent aux angles en degré de la caméra autour des axes X,Y et Z dans la scène THREE.js.
— Le bouton *Save Camera Pos.* sauvegarde la position courante de la caméra.
— L'outil **Go to Save** permet de revenir à une position de la caméra précédemment sauvegarder.

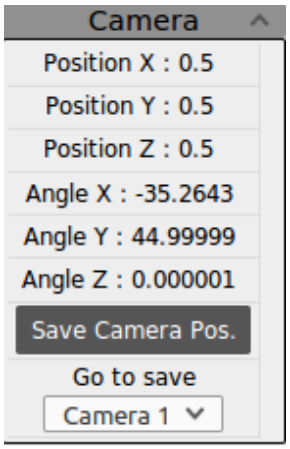


FIGURE 2.16 – Outils d'affichage des infos de la caméra

Chapitre 3

Description technique des barres d'outils et des outils en eux-même

3.1 Introduction

La barre d'outils et les outils associés sont écrits comme le reste de l'application en Javascript sans bibliothèque extérieur. Ce code Javascript, présent dans le dossier *js/Toolbar* génère une fois utilisé dans l'application le code HTML et CSS des barres d'outils et des outils en eux-mêmes. Cela permet, grâce à des appels de fonctions d'ajouter des outils aux barres d'outils facilement.

Voici une succincte description des fichiers présents dans le dossier *js/Toolbar* :

buttonTool.js : Fichier contenant l'outil bouton.

caseTool.js : Fichier contenant l'outil de sélection de données dans les vues.

colorTool.js : Fichier contenant l'outil de choix de couleur.

elementTool.js : *(Classe Abstraite)* Fichier contenant l'outil de base (Classe mère de tous les outils).

folderTool.js : Fichier contenant l'outil qui contient des outils dans la barre d'outils de la vue.

inputTool.js : *(Classe Abstraite)* Fichier contenant l'outil basé sur la balise HTML <input> (Classe mère des outils contenant la balise <input>).

listTool.js : Fichier contenant l'outil menu déroulant.

printTool.js : Fichier contenant l'outil d'affichage de nombre.

rangeTool.js : Fichier contenant l'outil de saisie de nombre par le biais d'une jauge.

switchTool.js : Fichier contenant l'outil bouton ON/OFF.

tabContainer.js : Fichier contenant l'outil contenant les onglets.

tabTool.js : Fichier contenant l'outil de création d'onglet.

textTool.js : Fichier contenant l'outil de saisie de texte.

toolbar.js : Fichier contenant la barre d'outils et contenant aussi le CSS de tous les outils et de la barre d'outils.

3.2 Description technique

Pour chaque outil et élément de la barre d'outils seront décrits un plan de la structure du DOM HTML de ces outils ainsi qu'un exemple d'utilisation.

3.2.1 Barre d'outils

La barre d'outils est contenue dans le fichier *toolbar.js*. Elle est faite de telle sorte qu'elle puisse contenir beaucoup d'outils tout en gardant une dimension fixe. Si il n'est pas possible d'afficher tous les outils en même temps dans la barre d'outils, un ascenseur se met en place (à droite pour les barres d'outils verticales et en bas pour les barres d'outils horizontales). La classe à utiliser pour intégrer une barre d'outils est la classe *Toolbar*.

Description de l'HTML

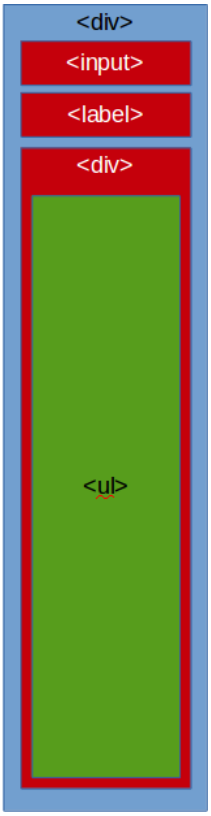


FIGURE 3.1 – Schéma HTML de la Toolbar

— La balise <**div**> (nommé dans la classe Toolbar *this.toolbar_volet*) en bleu contient tous les éléments de la barre d'outils.
— La balise <**input**> (nommé dans la classe Toolbar *this.toolbar_button*) est le bouton pour ouvrir/fermer la barre d'outils.
— La balise <**label**> (nommé dans la classe Toolbar *this.toolbar_button_label*) est le label associé au bouton précèdent.
— La balise <**div**> (nommé dans la classe Toolbar *this.toolbar_main*) en rouge est un container de barre d'outils.
— La balise <**ul**> (nommé dans la classe Toolbar *this.toolbar_contener*) contient tous les outils de la barre d'outils.

Utilisation

Pour ajouter une barre d'outils dans une application, il suffit d'appeler le constructeur de la classe *Toolbar* et ensuite de paramétrer l'objet grâce aux méthodes présentes dans la classe *Toolbar* (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

Créons et affichons une barre d'outils et ensuite la paramétrer :

```
1 var toolbar = new Toolbar('tdata',parentHTML,200,90); //Creation de la barre d'outil avec l'id "
      tdata", comme parent "parentHTML" et comme largeur 200 pixel et hauteur 90 pixel
2 toolbar.addButtonClose(false); //On enleve le bouton qui permet d'ouvrir et fermer la barre d'
      outils
3 toolbar.largeurInPourcent(35); //On definit la largeur de la barre d'outil a 35% de la fenetre
4 toolbar.setVertical(false); //La barre d'outil est horizontale
5 toolbar.noPadding(true); //On ne met pas de marge interieur a la barre d'outils
6 toolbar.noScroll(true); //On n'affiche pas le scroll de la barre d'outils
```

3.2.2 Outils abstraits (ElementTool et InputTool)

ElementTool

L'outil abstrait *ElementTool* est la base à tous les outils, c'est-à-dire que tous les outils héritent de cet outil abstrait. Cet outil est contenue dans une balise <li>, en effet, lors de l'ajout d'un outil dans la barre d'outils, le container de la barre d'outils étant une balise <ul>, la balise <li> est parfaite pour contenir l'outil.

Voici un schéma HTML de l'outil *ElementTool* en lui-même :

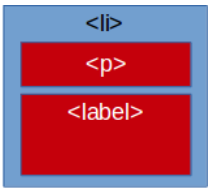


FIGURE 3.2 – Schéma de ElementTool

— La balise <**li**> (nommé dans la classe ElementTool *this.main*) contient l'outil en lui-même.
— La balise <**p**> (nommé dans la classe ElementTool *this.label*) contient le nom de l'outil.
— La balise <**label**> (nommé dans la classe ElementTool *this.element*) contient l'intérieur de l'outil.

InputTool

L'outil abstrait *InputTool* est un outil qui hérite directement de l'outil abstrait *ElementTool*. Cet outil contient une balise <input> en plus de l'HTML de l'outil *ElementTool* (en effet, la balise <input> est utilisée dans bon nombre d'outil).

Voici un schéma HTML de l'outil *InputTool* en lui-même :

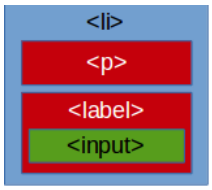


FIGURE 3.3 – Schéma de InputTool

— La balise <**li**> (nommé dans la classe InputTool *this.main*) contient l'outil en lui-même.
— La balise <**p**> (nommé dans la classe InputTool *this.label*) contient le nom de l'outil.
— La balise <**label**> (nommé dans la classe InputTool *this.element*) contient l'intérieur de l'outil.
— La balise <**input**> (nommé dans la classe InputTool *this.input*) contient la zone de texte.

3.2.3 Outils de saisie de texte (TextTool)

L'outil *TextTool* sert à saisir du texte. Cet outil est crée par la classe *TextTool* (qui hérite de *InputTool*).

FIGURE 3.4 – L'outil TextTool

Description de l'HTML

L'HTML est le même que l'outil InputTool.

Utilisation

Pour ajouter cet outil dans une barre d'outils, il suffit d'appeler le constructeur de la classe *TextTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
1 var texttool = new TextTool("NOM","Id",Valeur_Initiale);
2 texttool.addEventListener("change",function (e){/*Evenement*/});
3 toolbar.add(texttool);
```

3.2.4 Outils de création d'onglet (TabTool et TabContainerTool)

L'outil de création d'onglet permet de créer des containers stockant des outils. Cet outil utilise deux classes (*TabContainerTool* et *TabTool*). En effet, la classe *TabContainerTool* sert à contenir les onglets en eux-mêmes et la classe *TabTool* représente les onglets en eux-même.

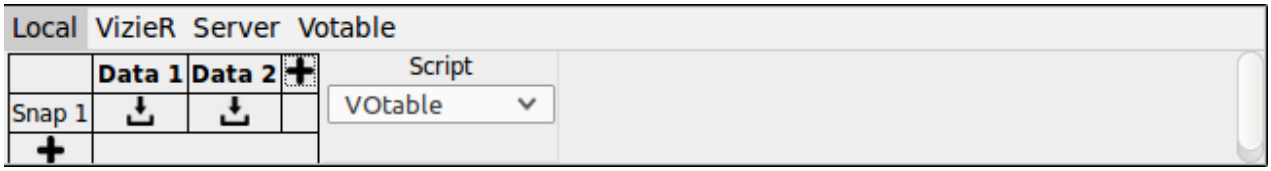


FIGURE 3.5 – L'outil onglet

Description de l'HTML

Classe TabContainerTool : L'outil *TabContainerTool* sert à contenir les onglets. Voici un schéma du conteneur sans onglet :

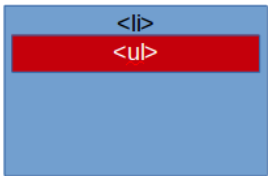


FIGURE 3.6 – Schéma de TabContainerTool sans onglet

- La balise <**li**> (nommé dans la classe TabContainerTool *this.main*) contient l'outil en lui-même.
- La balise <**ul**> (nommé dans la classe TabContainerTool *this.contener_label*) contient la liste des labels associés aux onglets.

Classe TabTool : L'outil *TabContainerTool* est l'onglet en lui-même. Voici un schéma de l'onglet :

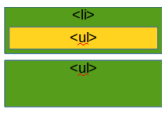


FIGURE 3.7 – Schéma de TabTool seul

- La balise <**li**> (nommé dans la classe TabTool *this.label*) contient le nom et le lien de l'onglet.

— La balise <**a**> contient lien de l'onglet (c'est grâce à ce lien que l'on peut ouvrir un onglet, en effet, ce lien est attaché à un événement de type *onClick* qui permet d'ouvrir l'onglet et de fermer tous les autres onglets du conteneur d'onglet père).
— La balise <**ul**> (nommé dans la classe TabTool *this.content*) contient la liste de outils inclus dans l'onglet.

Ajout d'un onglet : Lors de l'ajout d'un onglet dans le conteneur d'onglet, deux actions se produisent :
— La balise <li> (qui est le label de l'onglet) de l'objet TabTool est inclus dans la balise <ul> de l'objet TabContainerTool (qui contient tous les labels des onglets).
— La balise <ul> (qui contient les outils de l'onglet) de l'objet TabTool est inclus dans la balise <li> de l'objet TabContainerTool.

Voici donc la schéma complet HTML de l'outil *TabContainerTool* avec un onglet à l'intérieur :

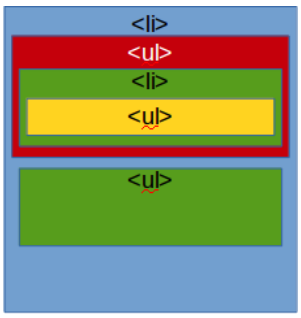


FIGURE 3.8 – Schéma d'outil *TabContainerTool* avec un onglet à l'intérieur

Utilisation

Pour ajouter cet outil dans une barre d'outils, il suffit d'appeler le constructeur de la classe *TabContainerTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter deux onglets au conteneur d'onglet ainsi qu'un outil dans le premier onglet (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
1  var onglet_conteneur = new TabContainerTool('Id');
2  toolbar.add(onglet_conteneur);
3
4  var onglet1 = new TabTool("Nom onglet 1",'id onglet 1',65); //Ici, 65 est la hauteur en pixel du
       conteneur d'outils de l'onglet
5  onglet_conteneur.addTab(onglet1);
6
7  var onglet2 = new TabTool("Nom onglet 2",'id onglet 2',65); //Ici, 65 est la hauteur en pixel du
       conteneur d'outils de l'onglet
8  onglet_conteneur.addTab(onglet2);
9
10 var outils = new ButtonTool("Bouton","id_bouton");
11 onglet1.add(outils); //On ajoute l'outils au 1er onglet
```

3.2.5 Outils bouton ON/OFF (SwitchTool)

L'outil *SwitchTool* sert à activer ou désactiver un paramètre. Cet outil est crée par la classe *SwitchTool* (qui hérite de *ElementTool*).

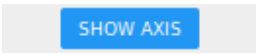


FIGURE 3.9 – L'outil SwitchTool en mode activé

Description de l'HTML

L'outil *SwitchTool* est basé sur l'outil *ElementTool*. Même si l'outil contient une balise <input>, il n'est pas basé sur *InputTool* car la construction de cette balise est assez spéciale (elle se fait dans la classe *SwitchTool* avec un appel de la fonction Javascript *innerHTML*.

Voici un schéma HTML de l'outil *SwitchTool* en lui-même :

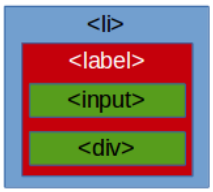


FIGURE 3.10 – Schéma de SwitchTool

— La balise <**li**> (nommé dans la classe SwitchTool *this.main*) contient l'outil en lui-même.
— La balise <**label**> (nommé dans la classe SwitchTool *this.element*) contient l'intérieur de l'outil.
— La balise <**input**> contient une checkbox (qui n'est pas affiché en temps que tel dans l'application).
— La balise <**div**> (nommé dans la classe SwitchTool *this.div*) contient le nom de l'outil.

Utilisation

Pour ajouter cet outil dans une barre d'outils, il suffit d'appeler le constructeur de la classe *SwitchTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
1 var switch = new SwitchTool("NOM",'Id');
2 switch.addEventListener('click',function(e) {/* Evenement */});
3 toolbar.add(switch);
```

3.2.6 Outils de jauge (RangeTool)

L'outil *RangeTool* sert à sélectionner un nombre entre une valeur maximum et minimum. Cet outil est crée par la classe *RangeTool* (qui hérite de *InputTool*). Il affiche à droite du sélecteur la valeur sélectionnée.

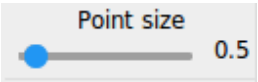


FIGURE 3.11 – L'outil RangeTool

Description de l'HTML

L'outil *RangeTool* est basé sur l'outil *InputTool*.

Voici un schéma HTML de l'outil *RangeTool* en lui-même :
— La balise <**li**> (nommé dans la classe RangeTool *this.main*) contient l'outil en lui-même.
— La balise <**p**> (nommé dans la classe RangeTool *this.label*) contient le nom de l'outil.
— La balise <**label**> (nommé dans la classe RangeTool *this.element*) contient l'intérieur de l'outil.
— La balise <**input**> (nommé dans la classe RangeTool *this.input*) l'outil range.
— La balise <**p**> (nommé dans la classe RangeTool *this.print_value*) contient l'affichage de la valeur sélectionné.

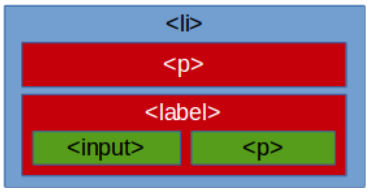


FIGURE 3.12 – Schéma de RangeTool

Utilisation

Pour ajouter cet outil dans une barre d'outils, il suffit d'appeler le constructeur de la classe *RangeTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
1 var range = new RangeTool("NOM",'Id',0,1,0.01,0.5); //ici, 0 represente la borne min, 1 la borne
      max, 0.01 le pas et 0.5 la valeur initiale du range
2 range.addEventListener('click',function(e) {/* Evenement */});
3 toolbar.add(range);
```

3.2.7 Outils d'affichage de nombre (PrintTool)

L'outil *PrintTool* sert à afficher un nombre. Cet outil est crée par la classe *PrintTool* (qui hérite de *ElementToolbar*).

FIGURE 3.13 – L'outil PrintTool

Description de l'HTML

L'outil *PrintTool* est basé sur l'outil *ElementTool*.

Voici un schéma HTML de l'outil *PrintTool* en lui-même :

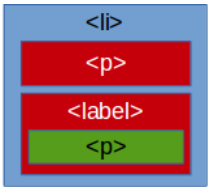


FIGURE 3.14 – Schéma de PrintTool

— La balise <**li**> (nommé dans la classe PrintTool *this.main*) contient l'outil en lui-même.
— La balise <**p**> en rouge (nommé dans la classe PrintTool *this.label*) contient le nom de l'outil.
— La balise <**label**> (nommé dans la classe PrintTool *this.element*) contient l'intérieur de l'outil.
— La balise <**p**> en vert (nommé dans la classe PrintTool *this.printer*) contient l'affichage de la valeur sélectionné.

Utilisation

Pour ajouter cet outil dans une barre d'outils, il suffit d'appeler le constructeur de la classe *PrintTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
var print = new PrintTool("Nom",'Id',12); //On affiche le nombre avec 12 caracteres
print.addPrint(1); //Ajouter 1 au compteur (on a donc 1 qui est affiche)
print.addPrint(2); //Ajouter 2 au compteur (on a donc 3 qui est affiche)
print.reset(); //Reset le compteur (on a donc 0 qui est affiche)
toolbar.add(print);
```

3.2.8 Outils de menu déroulant (ListTool)

L'outil *ListTool* sert à afficher un menu déroulant . Cet outil est crée par la classe *ListTool* (qui hérite de *ElementToolbar*).

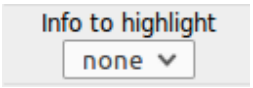


FIGURE 3.15 – L'outil ListTool

Description de l'HTML

L'outil *ListTool* est basé sur l'outil *ElementTool*.

Voici un schéma HTML de l'outil *ListTool* en lui-même :

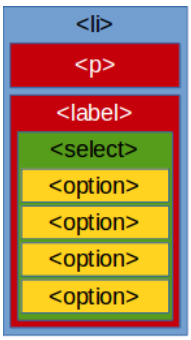


FIGURE 3.16 – Schéma de ListTool

— La balise <**li**> (nommé dans la classe ListTool *this.main*) contient l'outil en lui-même.
— La balise <**p**> (nommé dans la classe ListTool *this.label*) contient le nom de l'outil.
— La balise <**label**> (nommé dans la classe ListTool *this.element*) contient l'intérieur de l'outil.
— La balise <**select**> (nommé dans la classe ListTool *this.list*) contient les options de la liste déroulante.
— Les balises <**option**> contiennent sont les options du menu.

Utilisation

Pour ajouter cet outil dans une barre d'outils, il suffit d'appeler le constructeur de la classe *ListTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
var list = new ListTool("Nom","Id");
list.addEventListener('change',function(e) {/* Evenement */});
list.addList("Option 1",0); //Option 1 est le nom affiche et 0 est l'id de l'element
list.addList("Option 2",1);
list.addList("Option 3",2);
list.addList("Option 4",3);
toolbar.add(list);
```

3.2.9 Conteneur d'outils (FolderTool)

L'outil *FolderTool* sert à contenir des outils dans un même espace. Il est possible, en cliquant sur le libellé de l'outil de caché le contenue d'un *FolderTool* (et de le ré-afficher par la même action). Cet outil est crée par la classe *FolderTool*.

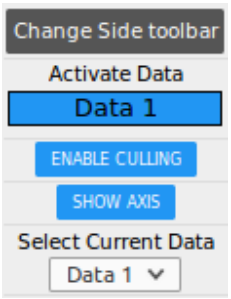


FIGURE 3.17 – L'outil FolderTool

Description de l'HTML

Voici un schéma HTML de l'outil *FolderTool* en lui-même :

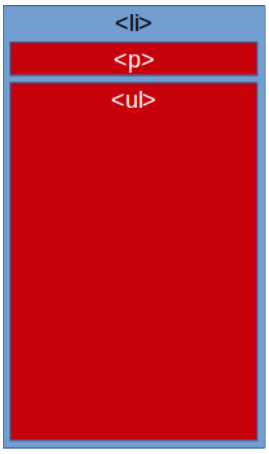


FIGURE 3.18 – Schéma de FolderTool

— La balise <**li**> (nommé dans la classe FolderTool *this.main*) contient l'outil en lui-même.
— La balise <**p**> (nommé dans la classe FolderTool *this.label*) contient le nom de l'outil.
— La balise <**ul**> (nommé dans la classe FolderTool *this.element*) est le conteneur qui va contenir les outils qui seront à l'intérieur.

Utilisation

Pour ajouter cet outil dans une barre d'outil, il suffit d'appeler le constructeur de la classe *FolderTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un outil dans le conteneur. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
1 var folder = new FolderTool("NOM","Id");
2 toolbar.add(folder);
3
4 var color = new ColorTool("NOM","Id");
5 folder.add(color); //On ajoute l'outil color dans le conteneur
```

3.2.10 Outils de sélection de couleur (ColorTool)

L'outil *ColorTool* sert à saisir une couleur. Cet outil est crée par la classe *ColorTool* (qui hérite de *InputTool*).

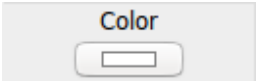


FIGURE 3.19 – L'outil ColorTool

Description de l'HTML

L'HTML est le même que l'outil InputTool.

Utilisation

Pour ajouter cet outil dans une barre d'outil, il suffit d'appeler le constructeur de la classe *ColorTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
1 var color = new ColorTool("NOM","Id");
2 color.addEventListener("change",function (e){/*Evenement*/});
3 toolbar.add(color);
```

3.2.11 Outils de sélection de données (CaseTool)

L'outil *CaseTool* permet d'activer ou désactiver des données dans une vue. Cet outil est crée par la classe *CaseTool* (qui hérite de *ElementTool*).

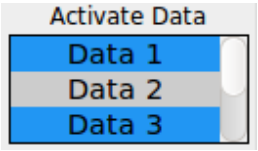


FIGURE 3.20 – L'outil CaseTool

Description de l'HTML

L'outil *CaseTool* est basé sur l'outil *ElementTool*.

Voici un schéma HTML de l'outil *CaseTool* en lui-même :
— La balise <**li**> (nommé dans la classe CaseTool *this.main*) contient l'outil en lui-même.
— La balise <**p**> (nommé dans la classe CaseTool *this.label*) contient le nom de l'outil.
— La balise <**label**> (nommé dans la classe CaseTool *this.element*) contient l'intérieur de l'outil.
— Les balises <**label**> en vert contiennent les cases à sélectionner ou désélectionner.
— Les balises <**input**> contiennent les checkboxs sélectionnés ou désélectionnés.
— Les balises <**div**> contiennent le nom des cases.

Utilisation

Pour ajouter cet outil dans une barre d'outil, il suffit d'appeler le constructeur de la classe *CaseTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

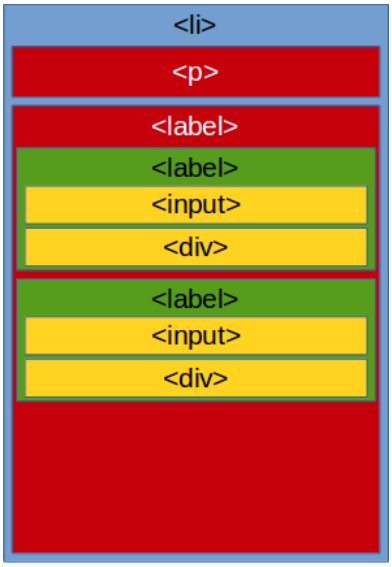


FIGURE 3.21 – Schéma de CaseTool

```
1 var caseTool = new CaseTool("NOM","Id");
2 caseTool.addCase("Case 1",1,false); //Case 1 est ici le nom, 1 l'id et false signifie que la case
      n'est pas selectionnee.
3 caseTool.addCase("Case 2",2,true); //Case 2 est ici le nom, 2 l'id et false signifie que la case
      est selectionnee.
4 caseTool.addEventListener("change",function (e){/*Evenement*/});
5 toolbar.add(caseTool);
```

3.2.12 Outils bouton (ButtonTool)

L'outil *ButtonTool* est un simple bouton. Cet outil est crée par la classe *CaseTool* (qui hérite de *Input*).

FIGURE 3.22 – L'outil ButtonTool

Description de l'HTML

L'outil *ButtonTool* est basé sur l'outil *InputTool*.

Voici un schéma HTML de l'outil *ButtonTool* en lui-même :

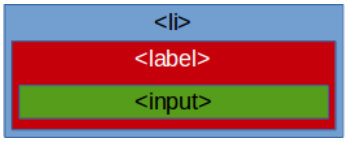


FIGURE 3.23 – Schéma de ButtonTool

— La balise <**li**> (nommé dans la classe ButtonTool *this.main*) contient l'outil en lui-même.
— La balise <**label**> (nommé dans la classe ButtonTool *this.label*) contient le nom de l'outil.
— La balise <**input**> (nommé dans la classe ButtonTool *this.folder*) contient le bouton en lui-même.

Utilisation

Pour ajouter cet outil dans une barre d'outil, il suffit d'appeler le constructeur de la classe *ButtonTool* et d'ajouter l'outil à une barre d'outils. De plus, dans notre exemple, on va ajouter un événement qui permettra d'interagir avec l'application. (Pour plus de détails au sujet de ces méthodes, il faut se référer à la Javadoc de l'application).

```
1 var button = new ButtonTool("NOM","Id");
2 button.addEventListener("click",function (e){/*Evenement*/});
3 toolbar.add(button);
```

Chapitre 4

A Savoir

— Pour avoir plus de détail au sujet des outils décrit dans la première partie ainsi qu'au fonctionnement de l'application, merci de se référer au rapport de stage de *Arnaud Steinmetz*,*Pierre Lespingual*, *Nicolas Buecher*, *Jérôme Desroziers*, *Nicolas Adam* et *Thibault Bouchard*.
— Pour le choix de couleur (ColorTool), cet outil ne marche pas sur les navigateurs Internet Explorer et Microsoft Edge. Sur ces navigateur, une zone de saisie de texte est affichée.
— L'application a été testée majoritairement sur Mozilla Firefox. L'application fonctionne aussi sur Safari.
— Sur les navigateurs Internet Explorer et Microsoft Edge, des erreurs Three.js sont lancées. De plus, il est impossible de charger des données VoTable (Erreur dans le fichier *votable.js*). Pour le choix de couleur (ColorTool), cet outil ne marche pas sur ces navigateurs, une zone de saisie de texte est affichée où il faut écrire la couleur sous forme hexadécimal (par exemple *#ffffff*).