IUT Nancy Charlemagne
Université de Lorraine
2 ter Boulevard Charlemagne
54052 Nancy Cedex
Dépt. Informatique

Étude et prototypage d'une WebApp HTML5 pour l'astronomie

Rapport de stage DUT Informatique
Observatoire astronomique de Strasbourg

Alexandre WEILER

Promotion 2012

IUT Nancy Charlemagne
Université de Lorraine
2 ter Boulevard Charlemagne
54052 Nancy Cedex
Dépt. Informatique

Étude et prototypage d'une WebApp HTML5 pour l'astronomie

Rapport de stage DUT Informatique

Observatoire astronomique de Strasbourg
11 rue de l'Université
67000 Strasbourg

Alexandre WEILER

Tuteur en entreprise : M. SCHAAFF André

Parrain du stage : M. CRUZ-LARA Samuel

Remerciements

Mes premiers remerciements vont tout naturellement à mon tuteur en entreprise M. André SCHAAFF, ingénieur de recherche au Centre de Données astronomiques de Strasbourg, pour m'avoir encadré pendant ces dix semaines de stage et pour sa grande disponibilité malgré ses responsabilités, pour ses encouragements et précieux conseils qui m'ont permis de mener à bien ce travail ainsi que pour les corrections minutieuses qu'il a portées à ce mémoire.

Je remercie vivement M. Samuel CRUZ-LARA pour avoir été mon responsable enseignant de l'IUT Nancy Charlemagne, pour sa visite à l'Observatoire et pour ses conseils avisés.

Je tiens à remercier chaleureusement M. Thomas BOCH, ingénieur de recherche au Centre de Données astronomiques de Strasbourg, de m'avoir accueilli le premier jour, pour avoir co-encadré mon stage, pour toutes ses explications et pour m'avoir aidé de nombreuses fois.

Je remercie très sincèrement M. Patrick NOURISSIER, chef d'entreprise NETLOR SAS et M. Philippe DOSCH, maître de conférences à l'IUT Nancy Charlemagne, d'avoir pris le temps de répondre à mes questions.

Mes remerciements s'adressent également aux membres de l'Observatoire et plus particulièrement Mme Anaïs OBERTO, M. André SCHAAFF, M. Gilles LANDAIS, M. Laurent MICHEL, Mme Patricia VANNIER et M. Thomas BOCH pour leur accueil chaleureux et leur présentation des services et outils du CDS.

Je remercie aussi mes deux collègues de bureau M. Julien SCHEFFMANN et M. Romain HOUPIN pour la très bonne ambiance.

Table des matières

CHAPITRE 1

INTRODUCTION

1.1 Contexte

Ce stage s'inscrit dans le cadre du devoir de veille technologique du Centre de Données astronomiques de Strasbourg. L'objet de ce travail concerne l'étude des nouvelles technologies et le prototypage d'outils applicatifs destinés à la communauté astronomique. La tendance actuelle est à la virtualisation* des ressources informatiques et à la mobilité. En effet, les logiciels applicatifs nécessitant une installation sur un ordinateur ou un appareil mobile laissent place progressivement aux applications web manipulables directement grâce à un navigateur Internet. L'avantage de ce processus est le gain de ressources informatiques pour les utilisateurs (espace mémoire) et un accès aux outils et aux données depuis n'importe quel terminal. Les exemples sont déjà nombreux et les géants du Web innovent dans ce sens. Nous pouvons citer la suite de logiciels bureautique Google Documents, le service de cartographie Bing Maps de Microsoft ou encore le service de « cloud computing* » iCloud d'Apple. Autant d'applications qui ne nécessitent qu'une connexion à Internet et un navigateur. Le Centre de Données astronomiques de Strasbourg (CDS) souhaite faire de même pour ses outils et services fortement utilisés de par le monde.

1.2 Problématique

Plusieurs problèmes se posent pour parvenir à créer une application web destinée à la communauté astronomique. En effet, les formats de données sont très spécifiques et impliquent des librairies développées spécialement pour l'astronomie. Mais avant tout, les volumes d'informations manipulés dans ce contexte sont gigantesques. Il s'agit ici de plusieurs centaines de téraoctets[1] qu'il faut concilier avec la mémoire limitée de nos appareils informatiques, mobiles ou non.

[1] 1 téraoctet = 10^{12} octets.

1.3 Organisation du document

Ce mémoire se compose de six chapitres. Dans le chapitre 2, nous présenterons l'Observatoire astronomique de Strasbourg afin de décrire le cadre de travail dans lequel ce stage a eu lieu. Nous nous intéresserons ensuite plus particulièrement à l'une de ses composantes, le Centre de Données astronomiques de Strasbourg en citant ses caractéristiques principales ainsi que ses missions et les besoins qui justifient le sujet de ce stage. Puis nous détaillerons précisément la mission qui nous a été confiée et les objectifs fixés.

Dans le chapitre 3, nous dresserons un état de l'art sur les concepts de pixélisation* appliquée à l'astronomie, de standard HTML5 et de représentation 3D au sein des navigateurs web. Nous introduirons ainsi le modèle HEALPix* en expliquant ses particularités et les raisons pour lesquelles il a été développé. Puis nous présenterons le futur standard HTML afin de comprendre comment il va faire évoluer l'Internet et quelles sont les nouvelles fonctionnalités utilisées lors ce stage. Enfin nous révélerons les différentes techniques que nous avons explorées afin de modéliser des volumes en 3D en ligne, sans avoir recours à des « plugins* » mais en exploitant toute la puissance de HTML5 et des technologies associées.

Nous aborderons dans le chapitre 4, les différentes phases de développement du prototype. De la modélisation de la sphère à la représentation multirésolution* de la texture en passant par le contrôle de la mémoire en JavaScript, nous expliquerons notamment pourquoi la sphère est en fait créée sous la forme d'une géométrie particulière, comment nous avons pu permettre de zoomer autant de fois que possible sur les cartes du ciel et enfin de visualiser des centaines de téraoctets d'images depuis n'importe quel terminal sans saturer la mémoire.

Nous ferons un bilan dans le chapitre 5 du support et des performances de notre application selon les différents navigateurs et les différentes plateformes. Nous constaterons ainsi que les résultats sont parfois inattendus. Nous évoquerons ensuite les perspectives du prototype et plus précisément les autres possibilités de HTML5 qui pourraient améliorer les performances ainsi que les fonctionnalités qu'il serait intéressant pour les professionnels d'ajouter.

Finalement, dans le chapitre 6, nous conclurons en reprenant les points abordés et en décrivant les apports tant sur le plan personnel que professionnel de ce stage.

CHAPITRE 2

L'OBSERVATOIRE ASTRONOMIQUE DE STRASBOURG

2.1 Présentation de l'Observatoire

2.1.1 Les lieux

Ce haut lieu scientifique prend ses origines dans la seconde moitié du XIXème siècle, à la fin de la guerre franco-prussienne. L'Allemagne victorieuse de ce combat face aux Français mal préparés, elle récupère les territoires de l'Alsace-Lorraine ainsi que la ville de Metz, conquis par les rois de France durant les trois siècles précédents. Ce rattachement à l'Allemagne est mal vécu par les Strasbourgeois encore traumatisés. Au même titre que Metz, Strasbourg se transforme et retrouve la prospérité grâce à la volonté politique du gouvernement. Ce dernier souhaite faire de la ville une vitrine de l'empire et du savoir-faire allemand. C'est dans son vaste plan d'urbanisation de Strasbourg que l'empereur Guillaume Ier décide d'y installer une université avec un jardin botanique et un observatoire astronomique.

L'édifice construit entre 1876 et 1880 est inauguré en 1881. Il est composé d'une Grande Coupole, d'un bâtiment des salles méridiennes avec deux plus petites coupoles et un bâtiment accueillant des bureaux et des résidences. Ils sont tous les trois reliés par un couloir couvert en forme de « Y ». La Grande Coupole en fer de plus de 9 mètres de diamètre abrite le Grand Réfracteur, une lunette de 7 mètres de long, la plus grande d'Europe à l'époque, aujourd'hui la troisième plus grande de France. Un siècle plus tard, l'Observatoire se dote d'un planétarium en 1981, un nouveau lieu de loisir, de diffusion de la connaissance et un outil pédagogique puissant qui permet de comprendre l'architecture et l'évolution de l'Univers. C'est également un lieu d'exposition et de découverte des plus beaux instruments du patrimoine de l'Observatoire.

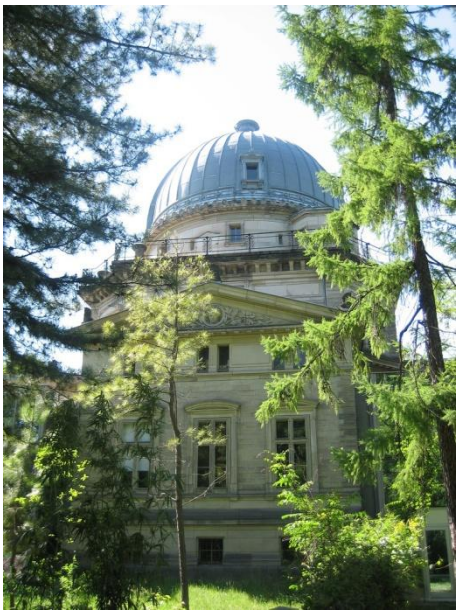


FIG. 1.1 : *Vue de la Grande Coupole.*

2.1.2 Les activités de recherche

La vocation initiale des activités de recherche concernait l'astronomie de position et l'observation des comètes, de météorites et d'étoiles variables. Par la suite les activités se sont étendues à la photométrie de nébuleuses et à l'observation d'étoiles doubles. Durant les années 1970, l'Observatoire développe l'archivage informatique, qui contribuera à la naissance du centre de données stellaires et qui deviendra en 1972 le Centre de Données astronomiques de Strasbourg (CDS). Actuellement les activités de recherche s'articulent autour de trois grandes équipes scientifiques : « Astrophysique des hautes énergies » étudie la physique des astres compacts en fin d'évolution, accrétion, éjection et phénomènes magnétohydrodynamiques*. « Galaxies » se charge des populations stellaires, propriétés chimiques et dynamiques de la Galaxie et des galaxies proches, milieu intergalactique, grandes structures, et dynamique gravitationnelle. Enfin le CDS s'occupe des méthodes de gestion de l'information et de l'exploitation scientifique des grands relevés*. Ce dernier est labellisé depuis 2008 "Très Grande Infrastructure de Recherche" (TGIR) par le Ministère de l'Enseignement Supérieur et de la Recherche.

2.1.3 Un rayonnement international

L'Observatoire est également membre du consortium *Survey Science Center* de la mission XMM-Newton. Cette dernière a pour objectif l'étude des rayons-X afin de mieux comprendre le fonctionnement de l'Univers et des événements violents qui s'y déroulent comme la vie des trous noirs ou les explosions d'étoiles. Mais c'est probablement le CDS qui contribue le plus à la renommée internationale de l'établissement. En effet il participe activement au développement de l'ASOV* et de l'IVOA*, dont nous parlerons ci-dessous, et est à l'initiative de services fortement utilisés par les astronomes et amateurs du monde entier.

2.1.4 Les Observatoires Virtuels

L'Observatoire Virtuel français est une collection d'archives de données interactives et d'outils logiciels utilisant l'Internet afin de bâtir un environnement de recherche scientifique dans lequel des programmes de recherche en astronomie sont conduits. De la même façon qu'un observatoire astronomique réel est un ensemble de télescopes, chacun avec une collection unique d'instruments astronomiques, l'Observatoire Virtuel consiste en un ensemble de centres de données, chacun avec une collection unique de données astronomiques, logiciels et capacités de calcul.

Il existe différents projets d'observatoires virtuels à travers le monde, notamment le projet européen Euro-VO* et le projet national des USA, l'US-VAO*. Ces derniers, ainsi que les

projets similaires d'autres pays comme la Chine, l'Australie, le Canada, le Japon, l'Inde, la Russie et la Corée, se sont associés en 2002 afin de coordonner leurs efforts au sein de l'alliance internationale IVOA avec la mission suivante : « *faciliter la coordination et les collaborations internationales nécessaires au développement et au déploiement d'outils, de systèmes et de structures rendant possible l'utilisation des archives astronomiques comme s'il s'agissait d'un Observatoire Virtuel unique* ».

L'IVOA regroupe ainsi 17 projets nationaux. Son action principale consiste à favoriser l'établissement de standards à travers des groupes de travaux qui suivent un fonctionnement identique à celui du World Wide Web Consortium (document de travail, proposition de recommandation, et finalement recommandation). Deux conférences par an permettent de coordonner les actions des différents groupes (IVOA Interopconferences).

Les groupes de travail de l'IVOA couvrent l'ensemble des besoins de standardisation de la discipline. Des données jusqu'aux applications, ils définissent les formatages des données, des métadonnées, les langages d'interrogation, les protocoles d'échange, etc. Le tableau décrivant les neuf groupes de travail, leur champ de compétences respectif ainsi que l'état des standards proposés est présenté dans l'annexe A.

L'IVOA requiert deux mises en œuvre indépendantes des standards avant de les valider. Le Centre de Données de Strasbourg, fortement impliqué dans l'Observatoire Virtuel participe à de nombreux groupes de travail. Le logiciel Aladin a été régulièrement utilisé pour tester et valider certains de ces standards. De ce fait, Aladin est devenu l'exemple type de «l'outil compatible OV », et un portail reconnu d'accès à l'Observatoire Virtuel.

Il s'agit donc de construire des standards d'échange, des outils d'interrogation, des systèmes d'extraction de l'information, de manière à globaliser les données et plus généralement l'information en astronomie, au niveau international. Dans ce cadre, les principaux centres de données astronomiques internationaux s'efforcent de donner davantage de visibilité à leurs bases de données et développent des descriptions détaillées de leur contenu. Ceci est rendu possible grâce à un immense effort de standardisation aussi bien des données que des méthodes et outils utilisés par les astronomes.

De nombreuses bases de données sont déjà interconnectées dans la communauté astronomique et de multiples applications client/serveur diffusent des données hétérogènes (articles de journaux, catalogues d'observations, images, spectres, etc.). Le Centre de Données

astronomiques de Strasbourg (CDS) y prend une part importante au travers de ses services Aladin, Simbad ou VizieR qui sont utilisés par les astronomes du monde entier. Nous les présenterons dans la partie qui suit.

2.1.5 Les services du CDS

FIG. 2.1 : *Logo d'Aladin.*

Aladin est un logiciel d'astronomie développé par le CDS pour la communauté internationale. Véritable atlas interactif du ciel, il permet aux scientifiques de localiser, accéder, comparer et analyser les données images et catalogues issues de la plupart des serveurs astronomiques (images d'archives, relevés du ciel, catalogues, bases bibliographiques). Ses caractéristiques techniques lui ont permis de devenir l'un des outils clés de la discipline aussi bien pour la préparation de missions d'observations (télescopes spatiaux Hubble et James Webb), que pour la visualisation de données des plus importants centres de données astronomiques européens (ESO, ESAC, CDS), américains (MAST/NASA, NED/NASA), canadiens (CADC)...Il est un véritable intégrateur de données hétérogènes issues des sites et des projets répartis sur toute la planète. Le logiciel entièrement écrit en Java est disponible sur sous forme d'applet sur le site du CDS.

FIG. 2.2 : *Logo de Simbad.*

Simbad est la base de données de référence pour la nomenclature et la bibliographie des objets astronomiques. C'est un service qui permet aux astronomes, à partir du nom d'un objet d'accéder facilement aux informations comme les coordonnées, mesures physiques, données bibliographiques, etc. de plus de 7 millions références astronomiques en dehors du système solaire. Il est aussi possible d'interroger la base en utilisant les coordonnées ou les références d'un objet. Simbad dispose également d'un résolveur de noms qui permet de connaître toutes les nomenclatures de l'objet considéré.

FIG. 2.3 : *Logo de VizieR.*

VizieR est une base de données qui regroupe environ 10 000 catalogues d'objets astronomiques. Ces catalogues sont en fait des tables relevées durant des missions d'observation et ajoutées à VizieR par les documentalistes. Ce service permet à un utilisateur d'accéder de manière homogène à des données hétérogènes de catalogues, de les croiser et de les exporter sous différents formats. Les interrogations sur la base de catalogues peuvent porter sur de multiples critères comme la longueur d'onde avec laquelle les objets ont étés observés ou encore le nom de la mission correspondante.

2.2 L'équipe du CDS

Le Centre de Données astronomiques de Strasbourg est l'une des principales composantes de l'Observatoire. Il offre un accès à des données astronomiques à forte valeur ajoutée au travers de services en lignes et d'applications grâce au travail de ses astronomes, informaticiens et documentalistes. Ils représentent un effectif d'une trentaine de personnes. Ces trois professions sont absolument complémentaires et sont indispensables au bon fonctionnement du centre. En effet, ils forment une chaîne entraînant le processus de collecte et d'enrichissement des données par les documentalistes, de vérification et d'exploitation des données par les astronomes et de développement d'outils et services pour la sauvegarde et l'accès aux données par les informaticiens.

La mission principale de ces derniers consiste à développer et maintenir les services et outils du CDS mais aussi d'assurer la veille technologique. En effet, la technique évolue très rapidement et le CDS tient à identifier les nouveautés prometteuses et à évaluer leur intérêt pour les services en ligne qu'il propose. Il a donc une activité significative de recherche et de développement.

Pour ce faire, les ingénieurs de recherche et les ingénieurs d'étude disposent de nombreux serveurs, de terminaux sous MacOS, Windows 7 et Linux. Afin d'expérimenter le développement mobile, le CDS s'est également doté de smartphones et de tablettes Android et Apple ainsi que d'un moniteur tactile.

2.3 Description du sujet

Le CDS est actuellement dans une phase d'évaluation d'outils mobiles et a déjà développé une application pour Android. Après s'être interrogé sur le bien-fondé à moyen terme des développements natifs, le CDS a proposé un stage comportant une phase exploratoire du nouveau standard web HTML5, le prototypage d'une application en JavaScript pour l'astronomie impliquant la manipulation d'images et enfin une évaluation des performances sur les différentes plateformes et navigateurs.

Cette application a pour ambition de fournir un service correspondant au mode « Allsky » d'Aladin c'est-à-dire la représentation multirésolution d'un relevé donné du ciel sous la forme d'une sphère en 3D. Pour le grand public, il s'agirait d'une expérience se rapprochant de celle proposée par le mode « Ciel » de Google Earth. Ce prototype doit ainsi permettre la découverte et l'exploration du ciel à différents niveaux de résolution et selon différentes longueurs d'onde (visible, infrarouge, rayons-X…).

L'objectif est donc d'utiliser les nouvelles possibilités offertes par HTML5 et WebGL afin de développer un outil multiplateforme* et d'étudier son support par les différents navigateurs et les terminaux mobiles.

Nous ferons dans le partie qui suit, un état de l'art des différents concepts mis en jeu lors de ce stage.

ÉTAT DE L'ART

Dans cette partie, nous traiterons des différents outils, langages et librairies qui préexistent au sujet de ce stage. Nous aborderons notamment la pixélisation HEALPix très répandue dans le domaine de l'astronomie, puis présenterons le standard HTML5 qui nous a permit de développer notre prototype. Enfin, nous étudierons les différentes méthodes permettant la représentation de scènes 3D au sein des navigateurs.

3.1 Pixélisation HEALPix

La pixélisation HEALPix a été proposée afin de répondre au besoin en termes de pixélisation des sondes WMAP et PLANCK, chargées d'étudier le Fond Diffus Cosmologique ou CMB (Cosmic Microwave Background). En effet, ces sondes fournissaient des cartes de plusieurs millions de pixels[2] répartis sur une sphère et il s'agissait de pouvoir traiter correctement ces données. Les autres pixélisations, présentées sur la figure ci-dessous, comme COBE, Quad Cube ou Isocaèdre ne convenaient pas, notamment à cause de l'absence d'iso-latitude* des pixels entraînant une explosion du temps de calcul des coefficients en harmoniques sphériques[3].

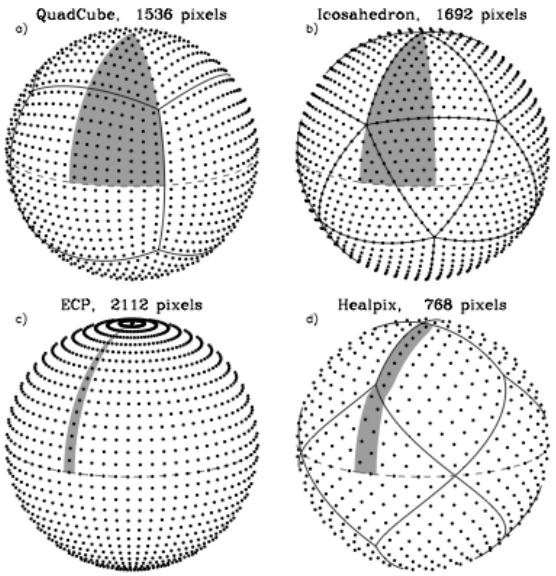


FIG. 3.1 : *Quatre façons différentes de diviser une sphère en sections régulières.*

[2] Dans ce document, il ne s'agit pas du plus petit élément d'une image mais plutôt de l'une des faces composant la sphère HEALPix. Nous emploierons également les termes « face », « dalle » ou encore « tuile ».

[3] Les harmoniques sphériques sont des fonctions mathématiques particulières utilisées en physique mathématique dès qu'intervient la notion d'orientation et de rotation.

HEALPix répond aux exigences des projets WMAP et PLANCK car elle possède trois propriétés fondamentales :

- Hiérarchie de la structure : la sphère est divisée de manière hiérarchique en quadrilatères curvilinéaires*. La pixélisation de plus faible résolution est composée de 12 pixels de base (4 autour de l'équateur et 4 autour des pôles). Pour passer aux résolutions suivantes, il suffit de diviser chaque pixel en quatre. Cependant, tout au long de ce document, nous considérerons que le premier niveau de résolution est le niveau 3 avec 768 faces.
- Les pixels ont la même surface.
- Iso-altitude des pixels : les pixels sont distribués sur des lignes de latitude constante. Cette propriété permet un calcul rapide des coefficients en harmoniques sphériques.

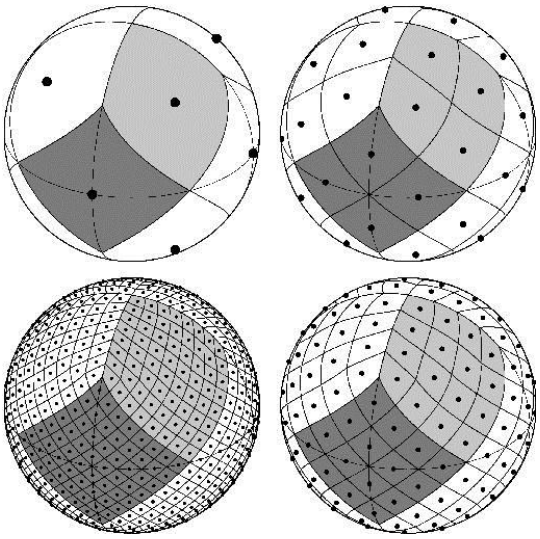


FIG. 3.2 **:** *Différentes échelles de la pixélisation HEALPix. Chaque sphère contient quatre fois plus de faces que la précédente. La sphère de niveau n est donc composée de* $3*4^n$ *faces.*

Cette pixélisation qui permet donc un calcul très rapide (bien qu'approché) des harmoniques sphériques, est très utilisée dans la communauté scientifique et possède de très nombreuses librairies performantes. En outre, elle permet de construire très facilement certaines transformées multi-échelles géométriques comme les curvelets* sur la sphère.

C'est pourquoi l'outil Aladin utilise la librairie HEALPix en Java pour le découpage des images et leur représentation sur une sphère en 3D. Le prototype HTLM5 devait donc manipuler ces images à la répartition très spécifique. L'absence de cette librairie en JavaScript a représenté l'une des principales contraintes de ce stage.

3.2 HTML5 et applications web

FIG. 3.3 : *Logo de HTML5.*

HTML5 est la dernière évolution du standard HTML regroupant un ensemble de technologies : CSS3, WebGL, SVG, File API… Cette nouvelle version fait de plus en plus parler d'elle et des acteurs tels que Google et Apple l'ont d'ores et déjà adoptée : l'un dans son client de messagerie mobile et dans YouTube (via HTML5 vidéo), l'autre pour iAd - la régie publicitaire d'Apple - écrite elle aussi intégralement en HTML5. Dans les faits, ce successeur de HTML 4.01 et de XHTML 1.0 est encore en phase d'adoption, poussé par les membres du consortium WHATWG (Web Hypertext Application Technology Working Group), fondé par des représentants d'Apple, de la Mozilla Foundation et de l'éditeur Opera Software. Les spécifications techniques de HTML5 font encore l'objet de modifications régulières, mais depuis 2009, il accroît de façon notable sa présence dans les navigateurs Safari, Firefox, Opera et Chrome. Avec la croissance soutenue du marché des smartphones (iPhone, Android, Windows Phone), HTML5 apporte des solutions aux problèmes qui bloquaient jusqu'à présent nombre d'innovations sur l'Internet mobile, motivant ainsi son adoption. Les créateurs de HTML5 ont eu à cœur de proposer aux développeurs une suite complète de balises et d'API permettant de concevoir des applications web « rich-media[4] » en s'appuyant sur ce langage, sur CSS3 et sur JavaScript.

Son but avoué est de faciliter le développement d'interfaces utilisateur riches. HTML5 est beaucoup plus orienté applicatif que ses prédécesseurs et veut s'affranchir des plugins pour utiliser au maximum les technologies web natives pour construire des applications interactives.

Le nouveau standard arrive donc avec beaucoup de nouvelles capacités disponibles depuis une API vaste et variée. Il permet désormais de visionner des vidéos sans plugin, d'ajouter des effets visuels aux textes, images, vidéos, d'utiliser des polices non standards, de dessiner en SVG, de faire de la 3D grâce au *canvas**…

Le service du CDS Aladin est déjà accessible depuis Internet grâce à une applet* Java. L'outil est complet, toutes les fonctionnalités sont présentes mais le temps de chargement et le poids

[4] Qui intègre différents médias (sons, vidéos, photos, métadonnée), présentés de manière interactive et temporelle au sein d'une interface de consultation ergonomique.

de l'application dégradent l'expérience utilisateur. De plus, cet applet ne s'inscrit pas dans la politique de développement d'applications mobiles. HTLM5 est donc l'occasion d'expérimenter un nouvel outil plus léger offrant une expérience intuitive et interactive et prêt pour l'utilisation mobile.

3.3 La 3D dans les navigateurs

Les nouvelles technologies introduites par HTML5 permettent de créer des objets en 3D et de les manipuler dans l'espace. Plusieurs possibilités se sont offertes à nous pour représenter notre carte du ciel en 3D. Nous allons présenter les différentes pistes que nous avons explorées, leurs avantages et inconvénients.

3.3.1 CSS3

CSS est à l'origine un langage prévu pour décrire la présentation des documents HTML. Couleur, taille, police d'écriture, disposition et style du texte, CSS permet surtout de donner de la forme au contenu et d'habiller les pages web de couleurs et d'images. Depuis l'introduction de CSS3 porté par HTML5, les possibilités se sont élargies, permettant ainsi de manipuler des dégradés de couleurs, d'ajouter des ombres au texte, des arrondis aux éléments *, des animations jusque-là possibles uniquement par l'intermédiaire de frameworks* JavaScript comme JQuery et bien d'autres choses encore.

Mais c'est la propriété « Transform » de CSS3 qui nous intéresse tout particulièrement. En effet, elle permet d'incliner, d'agrandir ou de rétrécir les éléments d'une page HTML en spécifiant un angle ou une mesure. Mais surtout, elle implante un repère orthonormé qui rend alors possible la rotation et la translation de ces éléments dans un espace tridimensionnel.

FIG. 3.4 : *Exemple d'un slider 3D réalisé avec la propriété « Transform » de CSS3.*

L'avantage des transformations CSS3 est qu'elles bénéficient de l'accélération matérielle, c'est-à-dire que les calculs, normalement pris en charge par le CPU, sont délégués au GPU de la carte graphique. Ceci peut augmenter significativement les performances et peut également réduire la consommation de ressources sur les appareils mobiles dont la puissance de calcul est réduite. Les tests que nous avons pu mener sur iPad avec diverses expérimentations étaient très prometteurs, offrant à chaque fois une expérience fluide et rapide.

Toutefois, les opérations de déplacement, de rotation et d'inclinaison n'étaient pas adaptées au modèle géométrique complexe que nous devions recréer. En effet, nous avons placé 768 éléments de type de telle sorte qu'ils s'ajustent et forment la sphère HEALPix. Mais pour ce faire, au lieu de placer les quatre coins d'une dalle à leurs coordonnées respectives, nous devions placer la dalle à la bonne position, selon la bonne rotation et la bonne inclinaison. De plus, dans une sphère HEALPix, les dalles ne sont pas forcément des parallélogrammes. Or, les transformations CSS3 ne s'appliquant que sur l'ensemble d'une et non pas sur chacun de ses cotés, elles ne permettent pas de les déformer jusqu'à obtenir des quadrilatères quelconques.

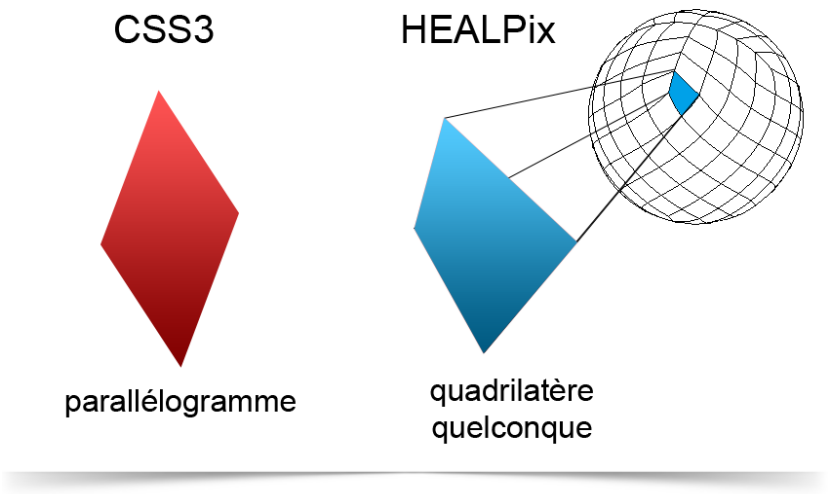


FIG. 3.5 : *Schéma comparant un exemple de parallélogramme obtenu avec CSS3 et un exemple de quadrilatère quelconque présent sur la sphère HEALPix.*

3.3.2 WebGL

WebGL est un standard pour la programmation en 3D avec le navigateur comme plateforme. La spécification finale du standard a vu le jour en 2010 et est définie par le Khronos Group, un consortium qui est aussi en charge de Open GL et Open CL. Il permet de réaliser des animations, des interfaces ou des jeux en 3D fonctionnant à la fois en ligne et hors connexion.

WebGL est une interface entre JavaScript et Open GL ES 2.0, une librairie en code natif qui accède directement au hardware des cartes graphiques. Il existe une version de Open GL pour chaque modèle de carte graphique, qui est généralement installée par le système d'exploitation, quel qu'il soit.

Le rendu se fait dans *canvas,* la surface d'affichage graphique qui fait partie de HTML5 et qui est implémentée par les navigateurs modernes. On peut utiliser WebGL sur d'autres plateformes, si elles utilisent WebKit* notamment. Les avantages de WebGL sont notamment sa rapidité d'exécution, sa puissance de rendu, ses nombreuses possibilités telles que la gestion des ombres et des lumières.

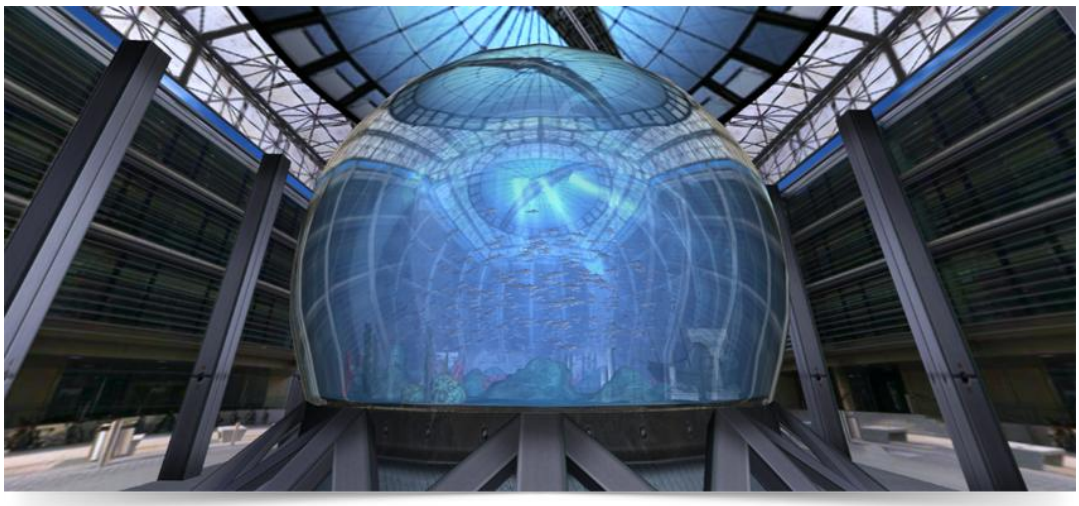


FIG. 3.6 : *Exemple d'une scène 3D réalisée avec WebGL représentant un aquarium futuriste.*

Cependant, la programmation en JavaScript pour WebGL est loin d'être triviale car le processus de rendu est complètement différent de celui du *canvas* 2D HTML5. En effet il utilise des « shaders », programmes informatiques calculant le rendu d'images de synthèses. Ils permettent de décrire l'absorption et la diffusion de la lumière, la texture à utiliser, les réflexions et réfractions, l'ombrage, le déplacement de primitives et des effets post-traitement.

3.3.3 Three.js

Three.js est une librairie JavaScript qui permet de réaliser des rendus 3D au sein même du navigateur. Elle permet de créer des scènes, des cameras, des lumières, des matériaux et bien d'autres choses encore. La force de Three.js est d'être accessible car plus simple d'utilisation que WebGL, s'inscrivant dans la lignée des librairies « Write less, do more[5] ». De plus ce framework nous laisse le choix du moteur de rendu ce qui implique qu'avec le même code, une scène 3D peut être dessinée en utilisant le *canvas* HTML5, WebGL, ou SVG. Ces derniers ayant des syntaxes très différentes, la 3D dans les navigateurs devient plus simple et plus rapide.

En effet, il suffit d'initialiser les trois composantes principales c'est-à-dire une scène, une caméra et un moteur de rendu et d'y ajouter une figure géométrique basique ou un système de particule pour obtenir une première scène en 3D.

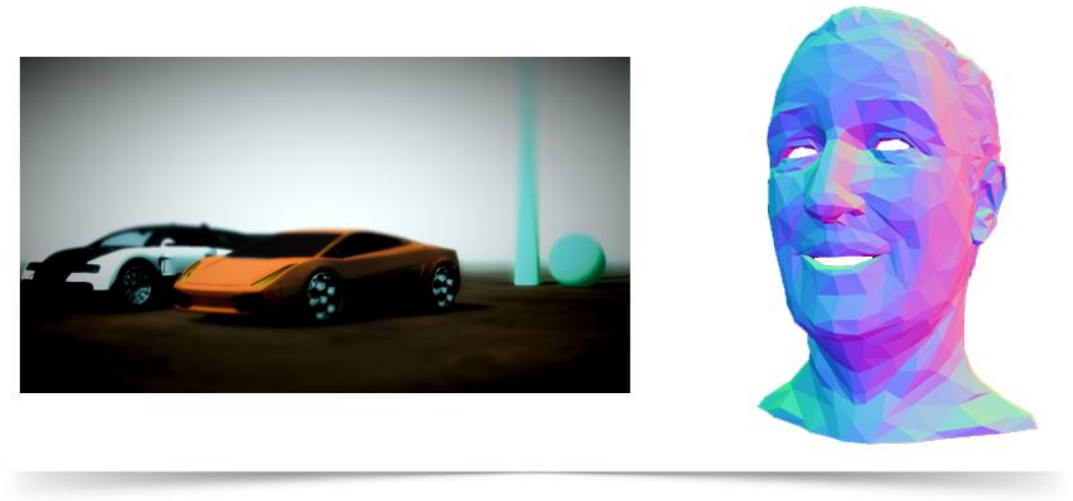


FIG. 3.7 : *Exemples réalisés avec Three.js. A gauche, une simulation de course rendue avec WebGL. A droite, le buste de Walt Disney utilisant directement le canvas HTML5.*

Les exemples de démonstration sont très impressionnants et montrent le potentiel quasi infini de Three.js. Ce projet open source est encore en cours de développement, s'améliorant un peu plus chaque jour. Son point fort est d'être accessible tout en restant très flexible.

[5] « En faire le plus possible en écrivant le moins »

3.3.4 Bilan

Deux tableau comparatifs sont présentés dans l'annexe B. Ils montrent le pourcentage de compatibilité du *canvas* et de WebGL selon les différents navigateurs, y compris dans leur version mobile.

CSS3 fournit des résultats très concluants en matière de rendu. En effet, les exemples 3D que nous avons pu tester étaient très légers au chargement et très fluides à l'exécution. Nous avons même trouvé une expérience de type « skybox[6] » pour la visite interactive et immersive d'un établissement sur iPad. Cependant, les transformations en trois dimensions ne permettent pas une assez grande flexibilité pour se plier aux contraintes de la géométrie HEALPix.

WebGL est une interface très puissante avec des résultats impressionnants. Elle permet de créer des scènes complexes aux effets multiples comme la simulation de fluides ou la réfraction de la lumière. Toutefois, Microsoft ne souhaite pas supporter WebGL sous le motif qu'un accès direct à une librairie système créerait une faille de sécurité. Sans compter que WebGL ne dispose d'un support partiel que sur quelques navigateurs mobiles. En effet Chrome Bêta sur Android le prend partiellement en charge alors que iOS ne le supporte pas du tout.

Three.js permet le rendu de scènes 3D en utilisant le *canvas* HTML5, WebGL ou SVG. Les tests que nous avons menés ont bénéficié d'un support sur la totalité des navigateurs, y compris mobiles, compatibles HTML5 en utilisant le *canvas*. Cette librairie bien que récente, nous permet un développement véritablement multiplateforme et compatible avec les terminaux mobiles. C'est donc cette dernière que nous avons choisie pour l'expérimentation de notre application pour l'astronomie.

Nous allons désormais aborder la phase de prototypage et réalisation de ce stage en décrivant les différentes étapes, problèmes rencontrés et les solutions mises en œuvre.

[6] « Boîte à ciel »

CHAPITRE 4

PROTOTYPAGE ET RÉALISATION

Dans ce chapitre nous allons aborder les différentes étapes de la conception et de la réalisation. L'objectif ici est d'argumenter les choix que nous avons pris et de décrire concrètement leur application, les problèmes que nous avons rencontrés et les solutions mises en œuvre. Nous évoquerons dans un premier temps la représentation volumique de la sphère puis nous expliquerons la modélisation multirésolution des textures. Enfin nous détaillerons les problèmes liés à la gestion de la mémoire en JavaScript.

4.1 Représentation volumique de la sphère

4.1.1 Les modèles

Comme nous l'avons annoncé au début de ce document, le but premier de notre prototype est de créer une représentation complète du ciel sous la forme d'une sphère. Nous allons décrire les différents procédés permettant la visualisation d'environnements 3D que nous avons explorés et justifier nos choix.

4.1.1.1 Skybox

En tant qu'utilisateur, une telle expérience est semblable au mode « Ciel » du logiciel Google Earth. Ce mode offre une vue de la voûte céleste telle que nous pourrions l'observer depuis l'espace. Nous pouvons donc formaliser cette représentation comme étant une caméra à l'intérieur d'un globe dont les parois sont texturées avec les images issues des différents télescopes au sol et spatiaux.

Il s'agit en fait, d'un procédé bien connu surnommé « skybox ». Il permet donc de donner, dans un espace tridimensionnel, l'illusion d'un espace beaucoup plus vaste. Techniquement, il s'agit d'un décor projeté sur les parois intérieures d'un cube dont le centre est une caméra. Cette technique a longtemps été utilisée en infographie 3D et même pour les jeux vidéos tant pour sa simplicité de mise en œuvre que pour son efficacité technique et graphique. En effet une skybox n'est constituée que de six images, une par face du cube dans lequel se trouve la caméra.

C’est cette technique que nous avions pu tester en CSS3 sur iPad. Cependant, deux contraintes se sont opposées à ce modèle. D’une part, les images fournies par le CDS respectent la pixélisation HEALPix. C’est-à-dire qu’il ne s’agit pas de les mettre bout à bout pour obtenir le patron d’un cube avec ses six faces. Pour recréer une texture plate du ciel tout entier, il faudrait refaire un traitement sur les images brutes, récupérées depuis les différentes missions d’observation après qu’elles aient été parfaitement réajustées. Ceci n’était pas envisageable.

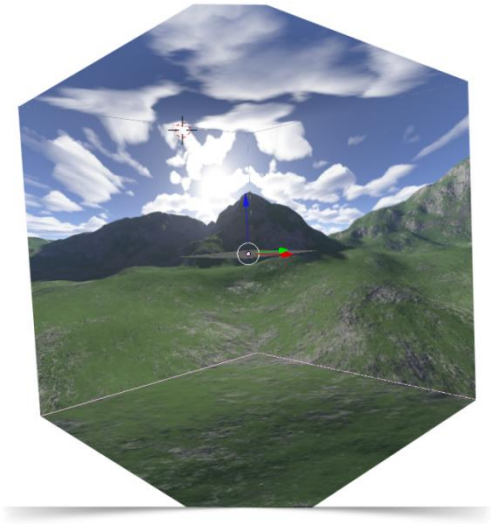


FIG. 4.1 : *Vue en coupe d’une « skybox » émulant un paysage montagneux.*

D’autre part, après concertation avec les membres de l’équipe du CDS, il a été décidé que l’application aurait pour cible les professionnels, et donc les astronomes. Or, tandis que le grand public est habitué à voir le ciel « de l’intérieur », les scientifiques se sont familiarisés avec une vue « de l’extérieur ». Autrement dit, il ne s’agit plus d’émuler en environnement atmosphérique mais plutôt de construire une carte du ciel projetée sur un globe. Cette représentation externe du ciel qui nous englobe naturellement n’est donc pas intuitive pour les novices. Devant permettre l’observation des étoiles « de l’extérieur », le model de la skybox n’est donc plus viable car l’illusion d’un environnement n’est propre qu’à la vue depuis l’intérieur du cube. Il nous faut donc créer une véritable sphère visible aussi bien de l’intérieur que de l’extérieur.

4.1.1.2 La sphère primitive

La librairie Three.js propose nativement des figures géométriques primitives telles que des cubes, des sphères, des plans ou des cylindres. Les sphères disponibles sont partitionnées selon la grille géographique habituelle (longitudes-latitudes). La façon la plus courante de texturer cette géométrie est d’utiliser une image de type « planisphère ». L’avantage de cette technique est qu’il n’y a qu’une seule image à manipuler. Cependant la texture présente des déformations aux pôles. Ces distorsions ne sont pas envisageables pour un tel outil d’observation destiné aux professionnels. C’est, entre autres, pour cette raison que le modèle HEALPix est aussi répandu dans la communauté scientifique.

4.1.1.3 La géométrie particulière

Three.js permet de créer n'importe quelle forme en définissant des points dans l'espace nommés vertex et des faces. Ainsi, il est possible de modéliser un polygone quelconque à partir de vertex. C'est précisément ce dont nous avions besoin car la sphère HEALPix est composée uniquement de quadrilatères.

De plus, la librairie nous autorise à regrouper différentes figures 3D en un seul objet. Ainsi nous avons pu créer 768 faces correspondant aux dalles de la sphère HEALPix, avant de les regrouper en un seul objet « THREE.Geometry ». Chaque face peut alors être texturée indépendamment des autres mais elles pivotent ensemble autour du même centre lorsqu'on effectue une rotation du groupe entier.

Ce modèle offre une flexibilité sans limite dans la mesure où nous positionnons chaque coin de chaque dalle aux coordonnées passées en paramètre. Nous avons donc opté pour cette structure car elle répond le mieux aux contraintes de la pixélisation HEALPix, tant en termes de modélisation géométrique que de texturation.

4.1.2 Mise en œuvre : HEALPix JS solution hybride

L'une des principales contraintes de ce stage est que la librairie HEALPix n'existe pas encore en JavaScript (ce développement fait actuellement l'œuvre d'un autre stage). Il est ainsi impossible de calculer directement en JavaScript les coordonnées de chaque dalle composant la sphère. Il a donc fallu trouver une solution « hybride ».

Nous avions pensé mettre en place un service web permettant de récupérer les coordonnées des quatre coins d'une face, via une requête URL. Ces informations auraient alors été générées « à la volée » grâce à la libraire HEALPix Java installée sur un serveur et renvoyées sous la forme d'un fichier XML ou JSON. Le problème ici tient dans le nombre de requêtes, beaucoup trop important. En effet, pour visualiser la sphère dans sa totalité à la plus basse résolution, il est nécessaire de construire 768 faces et donc de requêter autant de fois le serveur au démarrage de l'application. Sachant qu'à chaque niveau supérieur, le nombre de dalles est multiplié par 4, cette solution n'est pas viable, aussi bien du côté client (avec un faible débit) que du côté serveur (avec plusieurs utilisateurs en simultané).

Nous avons donc opté pour un pré-calcul des coordonnées depuis la librairie HEALPix en Java directement dans un fichier JavaScript. Cependant, il fallait effectuer ces calculs pour

chaque niveau de résolution de la sphère. Certains relevés descendent jusqu'au niveau 20 soit une sphère de plus 13 194 milliards de pixels. Le fichier de coordonnées correspondant pèserait alors dans les 760 téraoctets. Les volumes de données manipulés ici sont gigantesques. Or, cela dépasse largement les possibilités des terminaux, mobiles ou non.

Nous avons alors décidé de séparer les coordonnées sous la forme de fichiers « .position ». Chaque fichier contient un tableau de coordonnées en JavaScript, correspondant aux quatre coins d'une face de la sphère. Ainsi, pour une face donnée, il existe une image correspondante sur le serveur d'images Alasky et un fichier « .position ». Les coordonnées sont ainsi récupérées via Ajax. Nous expliquerons plus tard pourquoi il existe aussi un fichier « .neighbours ».

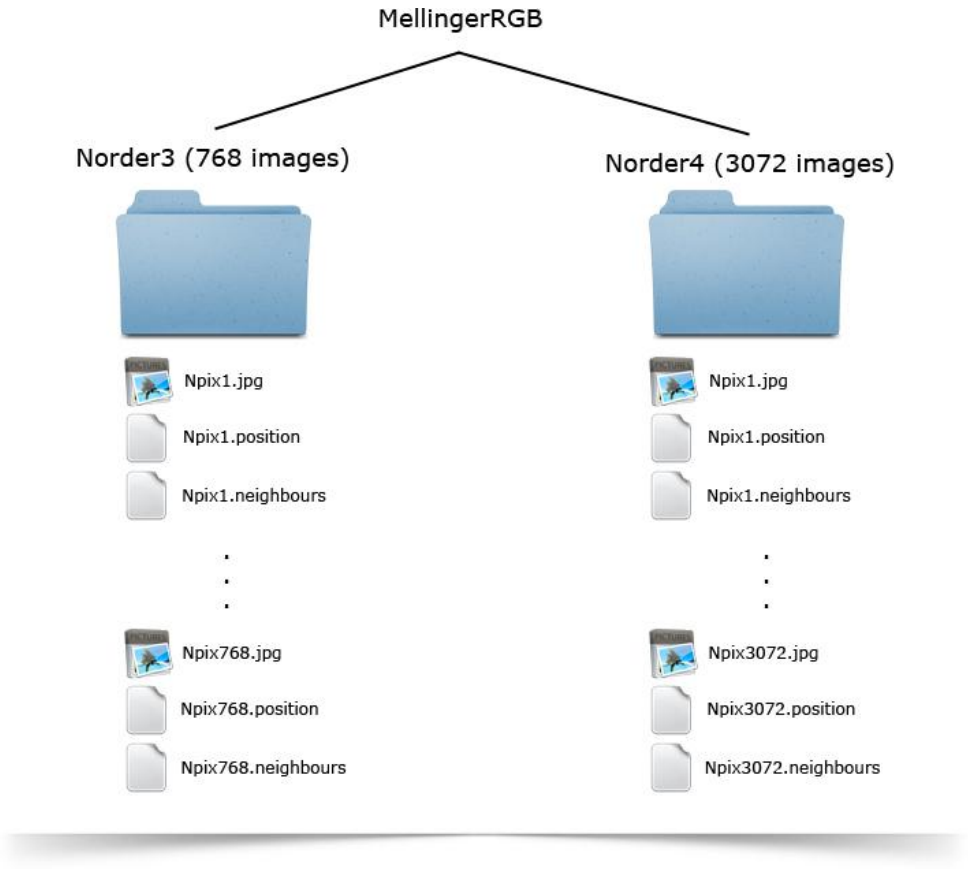


FIG. 4.2 : *Schéma de la structure des dossiers. MellingerRGB est le nom d'un relevé astronomique. Le niveau 3 est habituellement le premier niveau de résolution.*

Cette solution a été décisive pour la gestion de la texture multirésolution et des niveaux de granularité.

4.2 La texture multirésolution

4.2.1 THREE.LOD : Level Of Detail

En modélisation 3D, la résolution des objets a un impact direct sur la mémoire utilisée. En effet, plus le niveau de définition est élevé, plus il y a de faces et de vertex. Dans certains cas, il n'est pas nécessaire d'afficher un objet en haute définition si l'œil humain ne perçoit pas la différence comparé à une résolution inférieure. Par exemple, dans une scène 3D représentant un relief montagneux, il n'est pas utile de multiplier les faces des vallées lointaines dont on ne perçoit pas les détails.

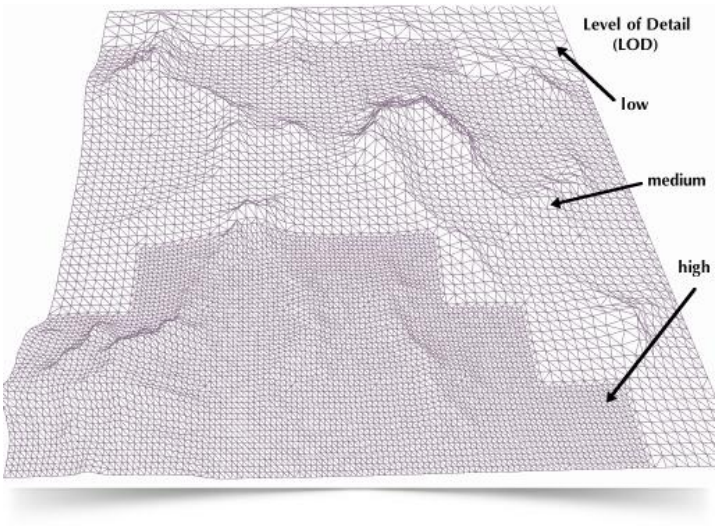


FIG. 4.3 : *Schéma des différents niveaux de détail d'un terrain en 3D. La résolution augmente du haut vers le bas du maillage.*

C'est pourquoi les développeurs de Three.js ont introduit un objet THREE.LOD pour « Level Of Detail[7] ». Il stocke les différents niveaux de résolution d'une géométrie donnée. Dans notre cas, cet objet stockerait dans un tableau une sphère HEALPix de niveau 3 avec 768 dalles, une de niveau 4 avec 3072 dalles, etc. Cependant, cette technique implique la modélisation de toutes les sphères (une pour chaque niveau de résolution), ce qui nécessite plusieurs secondes voire plusieurs minutes de calculs et jusqu'à plusieurs centaines de Mo de données en mémoire.

La mémoire étant un aspect crucial de l'application, il a fallu explorer une nouvelle piste pour gérer les niveaux de détail.

[7] « Niveau De Détail »

4.2.2 Détermination d'un cône de vue

Nous avons vu dans les parties précédentes qu'il était délicat de modéliser entièrement les sphères HEALPix à partir d'un certain niveau de résolution car elles comptent un nombre bien trop important de dalles. Pour allier « limitation de mémoire » et « importante masse de données », nous avons décidé de déterminer un cône de vue. Ceci implique de n'afficher que la section visible de la sphère, c'est-à-dire une lentille couvrant toute la surface de la fenêtre du navigateur.

Nous avons estimé que 25 dalles suffisaient à couvrir cette surface. Ainsi, même au niveau 10 (plus de 12 millions de faces), nous n'affichons que les 25 tuiles visibles par l'utilisateur. Cette astuce résout définitivement les problèmes liés aux volumes de données à charger et aux calculs à effectuer. De cette façon, pour n'importe quel niveau de résolution au-delà du numéro 3 (correspondant à la sphère entière), le navigateur ne manipule que 25 images, pas une de plus.

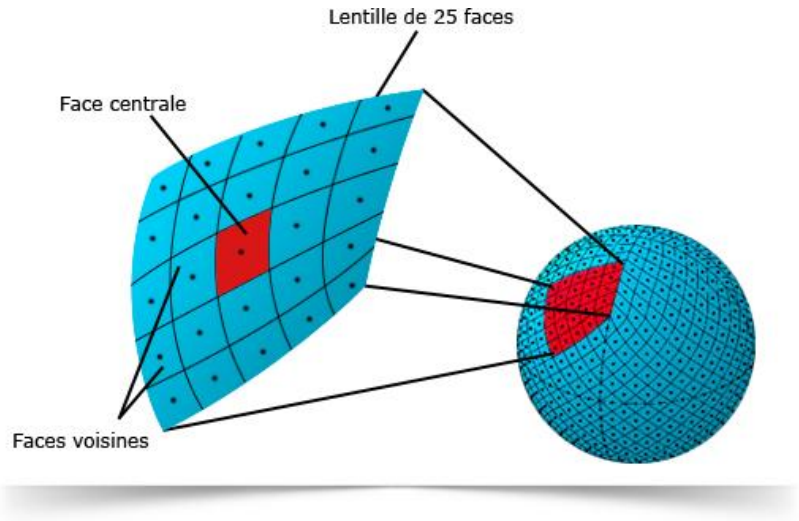


FIG. 4.4 : *Schéma de la restriction de l'affichage de la sphère à une lentille de 25 pixels.*

4.2.3 Création dynamique de la lentille

Ne disposant pas de la librairie HEALPix en JavaScript, nous devions trouver le procédé permettant d'afficher la partie de la sphère visible par l'utilisateur. Nous avons alors compris qu'il nous fallait déterminer la face de la sphère se situant au centre de l'écran et de récupérer les dalles voisines dans la limite des deux premières couronnes, soit un total de 25.

Three.js dispose d'un outil très utilisé dans le rendu d'images de synthèse : le ray tracing ou « lancer de rayon ». Élaboré pour simuler le parcours inverse de la lumière, cette technique permet de reproduire les phénomènes physiques tels que la réflexion et la réfraction. Le ray tracing consiste à lancer un rayon depuis le point de vue (la caméra) dans la scène 3D. Le premier point d'impact du rayon sur un objet définit l'objet concerné, et dans notre cas, la face de la sphère qui se situe au centre de l'écran.

Nous pouvons ainsi déterminer la dalle centrale et donc charger le fichier « indice_dalle_centrale.neighbours ». Ce fichier stocke les indices des 24 faces voisines. Pour chacune d'elles, en plus de la première, nous chargeons le fichier « .position » correspondant et nous la modélisons dynamiquement grâce à ses coordonnées. Nous obtenons donc une section de sphère de 25 dalles.

Le rayon est projeté à chaque rafraîchissement du *canvas*. Si la face centrale n'a pas changé, il ne se passe rien. Si au contraire l'utilisateur a tourné suffisamment la sphère et qu'une nouvelle tuile se retrouve au centre de l'écran, alors la lentille est recrée dynamiquement.

Afin d'optimiser le temps de calcul et le nombre de requêtes, nous avons choisi de ne pas reconstruire à chaque fois toute les faces de la lentille. En effet, deux dalles attenantes partagent plusieurs voisins. Dans le cas du modèle HEALPix, ce nombre est d'environ 20. Recréer ces tuiles déjà existantes aurait été un gaspillage de ressources. C'est pourquoi lorsque la face centrale change d'indice après une rotation suffisante de la sphère, nous ne modélisons que les faces qui n'apparaissaient pas auparavant avant de supprimer les faces qui n'existent plus sur la lentille. Pour ce faire nous comparons deux tableaux, l'un stockant les indices des faces courantes et l'autre ceux des voisins de la nouvelle face au centre. L'intersection des deux tableaux correspond aux tuiles inchangées, le reste du premier tableau est composé des « faces sortantes » à supprimer, le reste du second contient les « faces entrantes » à créer (voir figure ci-dessous).

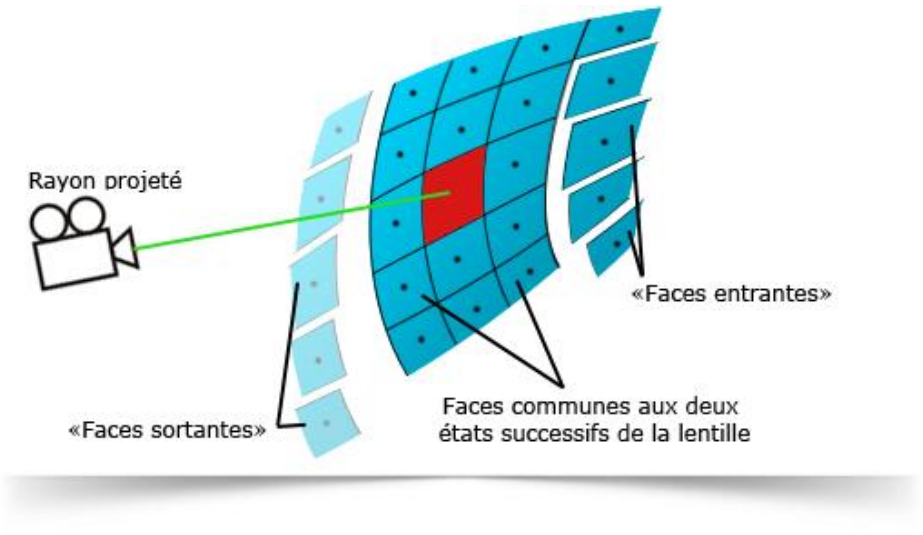


FIG. 4.5 : *Schéma de la création dynamique de la lentille après une rotation suffisante de la sphère.*

Ainsi, entre deux états différents de la lentille, nous n'effectuons des calculs que sur 5 faces environs au lieu des 25 prévues.

4.2.4 Passage aux niveaux de résolution supérieurs

Grâce à la technique du « ray tracing » expliquée précédemment, nous pouvons à chaque instant connaître l'indice de la face au centre de la fenêtre du navigateur. Nous l'appelons la « face centrale courante ».

Nous avons déjà expliqué que l'application doit permettre de zoomer sur une région bien précise du ciel. Pour cela, nous avons défini une échelle de correspondance entre les différents niveaux de résolution HEALPix et une valeur relative au mouvement de la molette de la souris. Lorsque la molette avance, c'est-à-dire que l'utilisateur effectue un « scroll up », cette valeur augmente. De la même manière, elle diminue lors du mouvement inverse ou « scroll down ». En réalité, cette valeur correspond à la coordonnée *z* de la caméra dans la scène 3D ce qui lui permet de s'approcher ou de s'éloigner la sphère. Cela donne donc l'impression de zoomer ou de dézoomer.

Lorsque cette coordonnée *z* atteint un certain seuil, nous augmentons la résolution. Pour ce faire, nous utilisons la hiérarchie de la structure HEALPix. En effet, pour obtenir les quatre dalles du niveau supérieur à partir d'une face donnée, il suffit de multiplier son indice par 4 et d'ajouter une variable *k* allant de 0 à 3. Par exemple, pour la face n°304, les « sous-faces » correspondantes sont (304*4+0) = 1216, (304*4+1) = 1217, (304*4+2) = 1218 et (304*4+3) = 1219. La tuile 1216 devient alors la nouvelle « face centrale courante » et il suffit de

récupérer les fichier « Npix1216.position », « Npix1216.neighbours » et « Npix1216.jpg » dans le bon répertoire et de recréer une nouvelle lentille en suivant le même processus qu'expliqué dans la partie précédente. Nous pouvons préciser que dans le cas inverse, pour passer à la résolution inférieur, il s'agit du même principe. Il suffit juste d'adapter les calculs mathématiques.

4.3 La mémoire JavaScript

Comme évoqué précédemment dans ce document, certains relevés astronomiques descendent jusqu'au niveau 20 de la hiérarchie HEALPix. Le nombre de tuiles et donc d'images correspondantes se compte en milliers de milliards et représente des quantités gigantesques de données. Elles dépassent alors la capacité de stockage de la plupart des ordinateurs et des terminaux mobiles. Nous avons donc expliqué que pour relever ce défi, seules les 25 faces visibles par l'utilisateur sont modélisées. De plus, pour garder le même nombre de tuiles tout en permettant de manipuler la sphère dans sa totalité, les faces de la lentille sont créées dynamiquement de proche en proche.

Cependant nous avons constaté qu'au bout d'un certain temps (variable selon les machines), le navigateur plantait. Nous avons découvert que le système arrêtait l'application car la mémoire dédiée à son processus atteignait rapidement son seuil critique. En effet, la mémoire vive utilisée par l'application pouvait monter jusqu'à plusieurs Go avant que le navigateur ne cesse de fonctionner. Nous étions face à une fuite de mémoire apparente au sein du prototype.

Après une longue investigation et de nombreuses recherches sur la gestion de la mémoire par JavaScript, nous avons pu déterminer la source du problème.

Contrairement au langage C, JavaScript utilise un « ramasse-miettes » (Garbage Collector) pour gérer automatiquement sa mémoire. Le GC recycle la mémoire préalablement allouée lors de l'instanciation de variables, de fonctions, etc. puis inutilisée. Son principe de base est assez simple : déterminer les objets qui ne sont plus utilisés par le programme et récupérer l'espace qu'ils utilisent.

Nous avons donc compris qu'à la génération d'une nouvelle lentille, les dalles supprimées ne l'étaient pas tout à fait dans la mesure où nous constations une accumulation des blocs de mémoire plus jamais réutilisés mais jamais libérés non plus. C'est précisément pour cette raison que nous avons préféré recycler nous-même la mémoire en réutilisant les objets à

supprimer. Ainsi, il n'y a qu'un nombre limité d'objets créés (faces, images, etc.) et la mémoire est mieux contrôlée.

Concrètement lors de la mise à jour de la lentille, au lieu de supprimer les faces sortantes (inutilisées) de la sphère et d'instancier de nouveaux objets pour les faces entrantes, nous remplaçons juste les coordonnées et l'indice des anciennes dalles par les données des nouvelles. De plus, en JavaScript les images sont également manipulées via des objets. De la même façon, plutôt que d'allouer de la mémoire supplémentaire, nous modifions directement la source des textures à recycler en changeant l'attribut src des objets Image.
Toutes ces « astuces » permettent un meilleur contrôle des objets et éviter ainsi les fuites de mémoire.

Chapitre 5

BILAN ET PERSPECTIVES

5.1 Support des navigateurs

Le dernier objectif de ce stage était de rendre compte du support de cette application selon les différents navigateurs et terminaux.

Nous avons tout d'abord testé notre prototype sur les ordinateurs de bureau et portables. Ci-dessous un tableau comparatif des 5 navigateurs les plus répandus.

Chrome de Google se distingue très largement des autres navigateurs. Le temps de chargement au démarrage de l'application est le plus rapide et l'affichage est le plus fluide. Cette réactivité est notamment due à un moteur JavaScript très performant. De plus, l'implantation du *canvas* HTML5 par Google est la plus avancée. Chrome est donc le navigateur qui offre la meilleure expérience.

Firefox 13 de Mozilla est très proche de Chrome en termes de fluidité. Cependant la *canvas* n'est pas aussi bien intégré, ce qui laisse apparaître certains défauts de la sphère HEALPix et notamment les contours des dalles qui ne sont pas parfaitement contiguës. Nous avons également remarqué que le moteur JavaScript de Firefox ne supporte bien aussi bien le changement de la source directement sur l'objet Image lors du recyclage des textures, ce qui générait parfois des bugs se traduisant par des images appliquées aux mauvaises faces. Pour contourner ce problème il a fallu faire quelques modifications mineures lors de la manipulation des textures.

Safari d'Apple dans sa version pour Windows est encore moins réactif que Firefox. L'affichage commence à être « saccadé » ce qui dégrade l'expérience utilisateur. De plus, le *canvas* souffre également d'une moins bonne implantation que sur Chrome en laissant apparaître les mêmes défauts de la sphère que Firefox.

Sans compter que dans sa version Mac, Safari intègre si mal la 3D dans le *canvas* HTML5 qu'un bug d'affichage flagrant déforme complètement les textures de la sphère.

Opera dans sa version 12 offre un rendu bien plus fluide que Safari, rejoignant ainsi Firefox. Cependant, les défauts de la sphère sont largement accentués ce qui dégrade considérablement l'aspect visuel de l'application. Là encore, le *canvas* souffre d'une mauvaise intégration.

Contrairement à nos attentes Internet Explorer 9 donne des résultats tout à fait satisfaisants. Avec une fréquence de rafraîchissement à la hauteur de Firefox, l'application paraît fluide. Les défauts de la sphère sont cependant bien moins perceptibles qu'avec Opéra.

En ce qui concerne les terminaux mobiles, nous avons pu tester notre prototype sur un large éventail de tablettes. Les observations suite à cette étude sont les suivantes :

L'Asus Eepad sous Android 4.0 et avec la version bêta de Chrome donne les meilleurs résultats même s'ils sont loin d'atteindre ceux obtenus sur les ordinateurs de bureau et portables. En effet, il n'y a pas de défaut apparent concernant la représentation de la sphère mais la cadence d'affichage est très faible. Le nombre d'images par seconde étant réduit, le rendu est assez peu fluide mais reste réactif aux mouvements du doigt.

Sur les autres terminaux Android (tablettes et smartphones) ne bénéficiant pas de l'accès à Chrome, le navigateur par défaut affiche correctement la sphère mais la fréquence de rafraîchissement du *canvas* est très réduite (1 image toutes les 2-3 secondes).

Enfin, nous avons observé qu'à l'image de sa version Mac, Safari iOS sur l'iPhone 4, l'iPad, l'iPad 2 et le nouvel iPad n'offrait pas de résultats satisfaisants. Notre application souffrait des mêmes bugs d'affichage et d'une grande lenteur.

Nous pouvons donc conclure que ce type d'application bénéficie d'un support quasi-total sur les navigateurs récents et d'un support partiel sur les terminaux mobiles. Concernant ces derniers, nous préférons utiliser la version bêta de Chrome pour une plus grande réactivité.

5.2 Perspectives

Il est prévu dans les semaines à venir d'améliorer les performances de notre application et d'enrichir ses fonctionnalités. En effet, nous souhaitons exploiter davantage les possibilités offertes par HTML5.

Par exemple, au démarrage de l'application, le programme doit créer la sphère HEALPix avec ses 768 faces. Il serait intéressant de garder en mémoire cet objet JavaScript grâce au LocalStorage (en le sérialisant) afin d'économiser les calculs effectués à l'initialisation de la sphère.

De plus, les moteurs JavaScript actuels sont « mono-thread* », autrement dit les navigateurs ne peuvent pas effectuer plusieurs traitements en parallèle. Ceci implique par exemple, dans le cas concret de la mise à jour de la lentille aux niveaux de résolution supérieurs à 3, que les 25 images nécessaires sont chargées séquentiellement. Selon les connections Internet, ce chargement peut être plus ou moins long. Afin de réduire ce temps, nous souhaiterions expérimenter les Web Workers. Ces derniers permettent d'effectuer des traitements en arrière-plan et donc de récupérer les 25 images en parallèle. Nous pourrions donc théoriquement diviser le temps de chargements de ces textures par 25.

Enfin, il serait pratique de pouvoir localiser et d'identifier des objets astronomiques sur la sphère et d'afficher un certain nombre d'informations les concernant.

CHAPITRE 6

CONCLUSION

6.1 Points abordés

L'objet de ce stage concernait l'étude et le prototypage d'une application web pour l'astronomie. L'intérêt de ce prototype est d'évaluer les possibilités offertes par HTML5 et de proposer une expérience intuitive et réactive manipulant d'importants volumes de données très spécifiques. L'application devant être multiplateforme et les objets 3D manipulés nécessitant une grande flexibilité de modélisation, nous nous sommes orientés vers la libraire JavaScript Three.js.

Dans ce document qui s'articule en quatre parties, nous avons présenté dans un premier temps l'Observatoire astronomique de Strasbourg. Nous avons abordé ses activités scientifiques portées par trois grandes équipes de recherche et deux services d'observation de premier plan. Nous avons vu ensuite que les services et outils du Centre de Données astronomiques de Strasbourg mondialement reconnus contribuaient fortement au rayonnement international de l'Observatoire. Puis nous nous sommes intéressés plus particulièrement aux missions du CDS et aux besoins justifiant l'existence de ce stage. Enfin, nous avons décrit précisément la mission qui nous a été confiée.

Dans une seconde partie, nous avons réalisé un état de l'art des différents concepts appliqués à la représentation 3D de relevés astronomiques en ligne. Tout d'abord nous avons introduit le modèle de pixélisation HEALPix en expliquant ses principales caractéristiques et en soulignant l'absence de librairie en JavaScript. Puis nous avons présenté le futur standard HTML5 en évoquant pourquoi il est l'initiateur de l'informatique virtualisée multiplateforme. Nous avons fini ce chapitre en établissant une liste des méthodes de représentation 3D au sein du navigateur et en avançant les arguments qui ont orienté notre choix vers la librairie Three.js au détriment de CSS3 et WebGL.

C'est ainsi que dans une troisième partie nous avons élaboré un premier modèle de sphère HEALPix en JavaScript. Il existe différents procédés permettant la représentation d'environnements 3D mais ils ne répondent pas tous aux contraintes de l'astronomie. En effet, les astronomes sont habitués à observer les relevés cartographiques de ciel selon un modèle sphérique et d'un point de vue externe. La méthode « skybox » n'était donc pas applicable.

De plus les formats d'images de permettaient pas de reconstruire une unique texture sous la forme d'un planisphère, qui plus est présente des déformations aux pôles. C'est pourquoi nous nous sommes orientés vers une modèle de géométrie particulière nous permettant de créer et de manipuler de multiples faces en 3D à la manière d'un seul et unique objet. Nous avons également détaillé comment nous avons pallié à l'absence de librairie HEALPix en JavaScript. Nous avons ensuite abordé le problème de la texture multirésolution. Three.js propose une méthode de représentation d'objet à plusieurs niveaux de détail. Nous avons vu que cette technique n'était pas compatible avec les volumes de données mis en jeux. C'est pourquoi nous avons opté une gestion dynamique d'une section de la sphère qui s'adapte selon un étalonnage bien défini en restructurant sa géométrie et en chargeant les textures correspondant aux différents niveaux de granularité. Enfin, nous nous sommes intéressés aux problèmes liés à la mémoire limitée des appareils informatiques, mobiles ou non. Nous avons effectivement constaté qu'à partir d'un certain temps d'utilisation, l'application chargeait assez d'images en mémoire pour planter le navigateur. Nous avons contourné cette impasse en gérant nous-même la mémoire à la place du Garbage Collector de JavaScript. Recycler les objets et limiter la représentation de la sphère à sa partie visible a été décisif dans la viabilité de notre prototype.

Dans une dernière et quatrième partie, nous avons dressé un bilan du support et des performances de notre application sur différentes plateformes et navigateurs. Il en ressort de cette comparaison que Chrome offre la meilleure expérience en termes d'aspect et de fluidité. Contrairement aux attentes, nous nous sommes aperçus qu'Internet Explorer 9 intégrait aussi bien le *canvas* que Firefox. De plus, le *canvas* souffre d'un défaut d'implantation sur Safari dans sa version Mac et iOS ce qui rend inutilisable le prototype sur iPhone et iPad. Concernant les autres terminaux mobiles, seules les tablettes Android à partir de Ice Cream Sandwich[8] avec l'application Chrome bêta, donnaient des résultats relativement acceptables. Nous avons conclu ce chapitre en abordant le futur de l'application et les améliorations que nous devrions apporter dans les semaines à venir. Nous avons évoqué l'optimisation des performances grâce aux autres fonctionnalités de HTML5 dont le LocalStorage et les Web Workers ainsi que l'intégration du pointage d'objets ce qui permettrait de localiser des objets astronomiques directement sur la sphère.

[8] Version 4 du système d'exploitation Android.

6.2 Apports personnels et professionnels

Ces stage a été l'objet d'une expérience très enrichissante tant sur le plan professionnel qu'humain. Les personnes rencontrées, le cadre de travail unique, le sujet passionnant sur lequel nous avons travaillé et les responsabilités qui nous ont été confiées sont autant d'éléments qui marqueront ce stage. De l'apprentissage de la librairie Three.js, aux conférences astronomiques données en anglais par des scientifiques de premier plan en passant par le perfectionnement en JavaScript, ces dix semaines de recherche et de développement resteront comme une expérience inoubliable. Sans compter la gratification de notre travail qui a été présenté par M. André SCHAAFF lors de la conférence Interop de l'IVOA à Urbana-Champaign, IL près de Chicago en mai dernier.

Bibliographie

Acko : http://acko.net/blog/making-worlds-1-of-spheres-and-cubes/

Aerotwist : http://aerotwist.com/

Centre de Données astronomiques de Strasbourg : http://cdsweb.u-strasbg.fr/CDS-f.gml

Console.log() : http://www.benvanik.tumblr.com/post/14225281899/simple-image-pyramid-demo-in-500-lines-of-javascript/

Creative JS : http://creativejs.com/tutorials/three-js-part-1-make-a-star-field/

Design HTML5 : http://www.design-html5.com/2012/04/06/webgl-et-three-js/

Fhtr : http://fhtr.org/BasicsOfThreeJS/

GitHub : https://github.com/mrdoob/Three.js/

Google Sky : http://www.google.com/sky/

Html5 rocks : http://slides.html5rocks.com/

Jet Propulsion Laboratory : http://healpix.jpl.nasa.gov/

Krpano : http://www.krpano.com/tours/kuchlerhaus/html5only.html

Laugharne : http://laugharne.me/post/11781335852/debuter-avec-three-js

Le Jardin des Sciences : http://jardin-sciences.unistra.fr/planetarium/

Nimphio : http://www.nimphio.us/three.js/examples/

Nokia Maps 3D WebGL : http://maps3d.svc.nokia.com/webgl/index.html

Observatoire astronomique de Strasbourg : http://astro.unistra.fr/

Stackoverflow : http://www.stackoverflow.com/questions/7544957/memory-leak-in-javascript-when-using-new-image

SpiderGL : http://www.spidergl.org/example.php?id=7/

Three.js – JavaScript 3D library : http://mrdoob.github.com/Three.js/

Wikipédia : http://en.wikipedia.org/wiki/HEALPix

http://fr.wikipedia.org/wiki/Harmonique_sphérique

http://en.wikipedia.org/wiki/Ondelette

Index des figures

Glossaire

Applet : Logiciel qui s'exécute dans la fenêtre d'un navigateur web.

ASOV : Action Spécifique Observatoire Virtuel France

Cloud computing : Concept qui consiste à déporter sur des serveurs distants des stockages et des traitements informatiques traditionnellement localisés sur des serveurs locaux ou sur le poste de l'utilisateur.

Curvelet : Terme anglais que nous pourrions traduire par « courbelette » désignant une extension du concept d'ondelette, outil mathématique utilisé dans l'analyse multirésolution de signaux tridimensionnels.

Curvilinéaire : Qui est composé de lignes et de courbes.

** ** : Abréviation de « division », balise HTML utilisée pour diviser un document en plusieurs sections.

Euro-VO : European Virtual Observatory

Framework : Kit de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.

HEALPix : Hierarchical Equal Area iso-Latitude Pixelization.

Iso-latitude : Caractéristique se traduisant par une répartition selon des lignes à latitude constante.

IVOA : International Virtual Observatory Alliance

Magnétohydrodynamique : Discipline scientifique qui décrit le comportement d'un fluide conducteur du courant électrique (liquide ou gaz ionisé appelé plasma) en présence de champs électromagnétiques.

Multiplateforme : Propriété d'un logiciel ou d'une application à fonctionner sur plusieurs appareils informatiques indépendamment du système d'exploitation.

Multirésolution : Présentant différents niveaux de granularité.

Pixélisation : Francisation du terme anglais « pixelization » traduisant le processus de quadrillage ou division d'un volume.

Plugin : Module d'extension qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

Relevé astronomique : Recensement (*survey* en anglais) d'objets astronomiques réalisé à l'aide de télescopes et de satellites.

Tread : Fil d'exécution d'un ensemble d'instructions du langage machine d'un processeur.

US-VAO : US Virtual Astronomical Observatory

Virtualisation : Ici, concept décrivant le portage de logiciels nécessitant une installation sous la forme d'applications web utilisables directement depuis l'Internet.

WebKit : Moteur de rendu de pages Web. Il est notamment utilisé par les navigateurs Safari et Chrome.

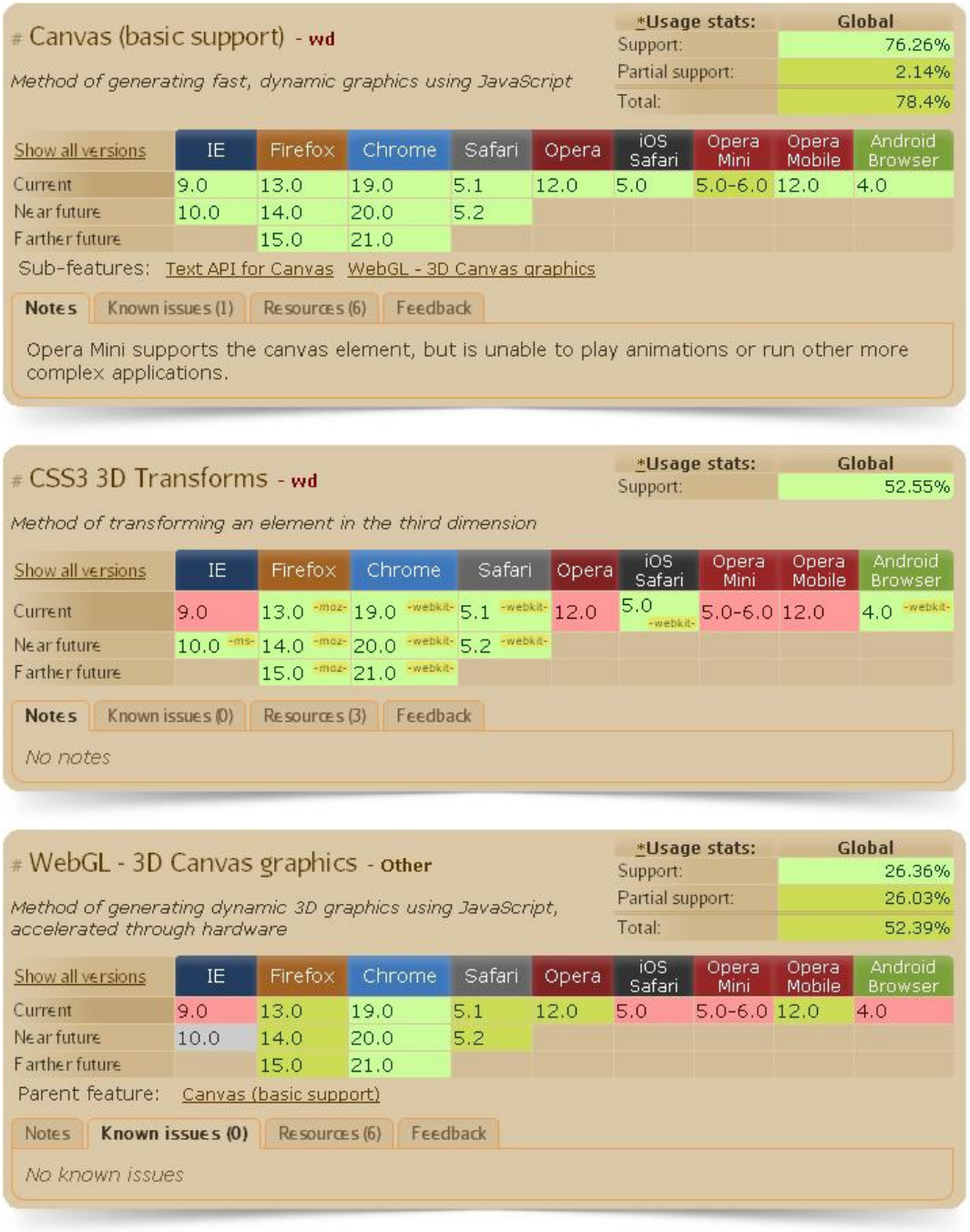
Annexe A

État des standards de l'IVOA

Groupe de travail	**Champ d'application**	**Standards** (*REC – Recommandé, PR – proposé à la recommandation, WD – document de travail*)
APPS – Applications	Concerne les échanges entre applications	SAMP-Simple Application Messaging Protocol - REC avril 2009
DAL – Data Access Layer	Gère les accès distants aux données	SIA – Image Access – PR oct 2009 SSA – Spectral Data Access – REC fév 2008 SLA – Atomic Line Access – PR jul 2009 CS – Catalog Cone Search –REC fév. 2008 TAP – Table Access Protocol – PR oct 2009
DM -Data Model	Décrit les modèles de données	STC – Space Time Coordinate – REC oct 2007 Charac – Dataset Characterisation – REC mars 2008 SDM – Spectral Data Model – REC oct 2007
GWS – Grid& Web services	Décrit l'usage des grilles de calcul et des services Web	SSO – Single Sign On Profile – REC jan 2008 VOSpace specification – REC jan 2008 CDP - Credential Delegation Protocol – PR aout 2009 UWS – Universal Worker Servide – PR sept 2009 BP – Service Web Basic Profile - WD sept 2008
Resource Registry	Gère l'annuaire des ressources	RM – Ressource Metadata – REC avril 2004 ID – IVOA identifier – REC mars 2007 RI – Registry Interface – PR sept 2009 VOResource – XML EncodingSchema – REC fév 2008 VODataService – VOResource Extension – PR sept 2009
Semantics	Décrit la signification des données	UCD – Uniform Content Descriptor – REC aout 2005 UCD words maintenance – REC mai 2006 UCD1+ Controlled Vocabulary – REC déc 2005 Vocabulary – REC oct 2009
VOE – VO Event	Gère les évènements transitoires	VOEvent – Sky Event Reporting Metadata – REC nov 2006
VQL – VO Query Language	Décrit les langages d'interrogation des ressources	ADQL – Astronomical Data Query Language – REC oct 2008 SkyNodeinteface – WD juin 2005
VOT – VOTable	Décrit le format des tables astronomiques	VOTable Format Definition- REC 2004, 1.2 – PR sept 2009

Tableaux comparatifs du support des technologies CSS3, WebGL et Canvas selon les navigateurs

= Supporté = En partie supporté = Non supporté = Support inconnu



Canvas (basic support) - wd

Method of generating fast, dynamic graphics using JavaScript

*Usage stats:	Global
Support:	76.26%
Partial support:	2.14%
Total:	78.4%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
Current	9.0	13.0	19.0	5.1	12.0	5.0	5.0-6.0	12.0	4.0
Near future	10.0	14.0	20.0	5.2					
Farther future		15.0	21.0						

Sub-features: Text API for Canvas WebGL - 3D Canvas graphics

Notes | Known issues (1) | Resources (6) | Feedback

Opera Mini supports the canvas element, but is unable to play animations or run other more complex applications.

CSS3 3D Transforms - wd

Method of transforming an element in the third dimension

*Usage stats:	Global
Support:	52.55%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
Current	9.0	13.0 -moz-	19.0 -webkit-	5.1 -webkit-	12.0	5.0 -webkit-	5.0-6.0	12.0	4.0 -webkit-
Near future	10.0 -ms-	14.0 -moz-	20.0 -webkit-	5.2 -webkit-					
Farther future		15.0 -moz-	21.0 -webkit-						

Notes | Known issues (0) | Resources (3) | Feedback

No notes

WebGL - 3D Canvas graphics - Other

Method of generating dynamic 3D graphics using JavaScript, accelerated through hardware

*Usage stats:	Global
Support:	26.36%
Partial support:	26.03%
Total:	52.39%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
Current	9.0	13.0	19.0	5.1	12.0	5.0	5.0-6.0	12.0	4.0
Near future	10.0	14.0	20.0	5.2					
Farther future		15.0	21.0						

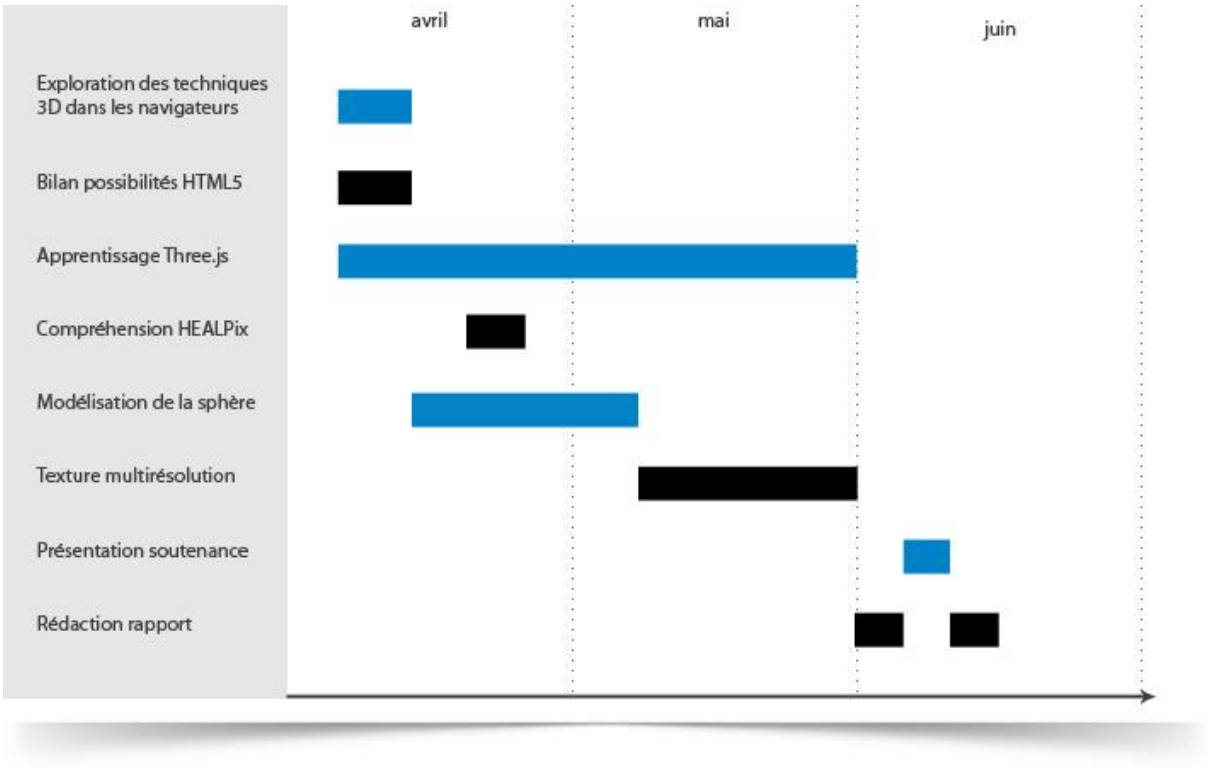
Parent feature: Canvas (basic support)

Notes | Known issues (0) | Resources (6) | Feedback

No known issues

Statistiques réalisées par le site *caniuse.com*, mises à jour le 27 mai 2012.

Etapes et phases du stage (Gantt)



Captures d'écran du prototype