Rapport de stage

Développement de SkyTouch

Application permettant le contrôle à distance de l'Atlas 3D du CDS, Aladin

Stagiaire: HECKEL Maxime
Maître de stage: SCHAAFF André

Remerciements

Je tiens tout d'abord à remercier André Schaaff , mon maître de stage pour m'avoir accueilli au sein du département informatique de l'Observatoire Astronomique de Strasbourg.

Je souhaite également remercier Mark Taylor, Astronome et développeur à l'université de Bristol au Royaume-Uni et un des concepteur du protocole SAMP, pour son aide plus que précieuse m'ayant permis de mener à bien mon projet.

Enfin je tiens à remercier Thomas Boch pour m'avoir fourni les versions bêtas de Aladin me permettant ainsi de compléter mon application.

INTRODUCTION

Recherchant un stage tourné vers le développement d'application, j'étais très satisfait du sujet de stage que m'a proposé André Schaaff. Il recherchait un moyen d'utiliser les terminaux tactiles munis d'écrans dit «multi-touch» afin de proposer une interface tactile pour l'utilisation d'un logiciel de cartographie du ciel: Aladin.C'est ainsi qu'est né le projet SkyTouch, dont le développement et les fonctionnalités sont détaillés dans ce rapport.

Ainsi dans un premier temps, je présenterai l'Observatoire Astronomique de Strasbourg avec en particulier les services du CDS, la branche de l'observatoire ou j'ai eu la chance de travailler. Je reviendrai ensuite plus en détail sur SkyTouch, l'objet de mon stage et comment ce dernier s'est déroulé.

I. Présentation de l'Observatoire

I.1 Observatoire Astronomique de Strasbourg

I.1.1 Présentation générale

L' Observatoire Astronomique de Strasbourg est un observatoire des sciences de l'Univers et une Unité Mixte de Recherche du CNRS et de l'Université de Strasbourg.
Il a été fondé en 1881 sur le site du campus historique de l'Université de Strasbourg. Ce dernier est constitué de trois bâtiments ainsi que du planétarium de la ville de Strasbourg destiné au grand public.

L'Observatoire abrite des activités de recherche et d'enseignement diverse, mais également les service du Centre de Données de Strasbourg (CDS).

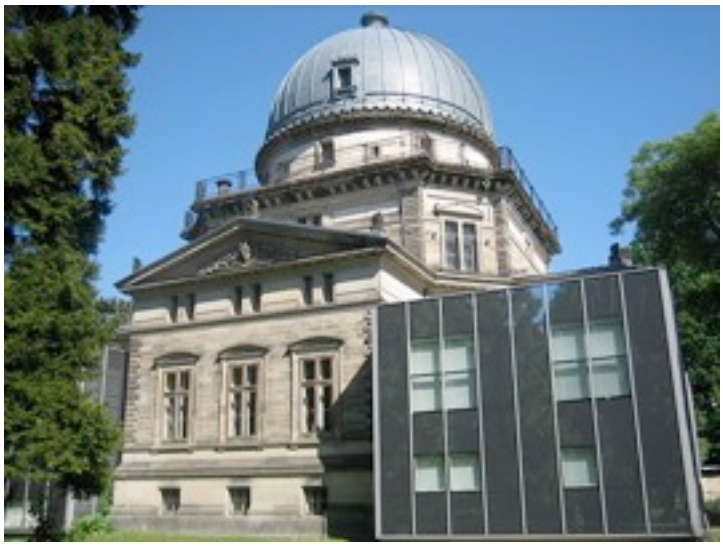


Figure 1: L'Observatoire Astronomique de Strasbourg

I.1.2 Les domaines de recherche

Quatre équipes permanentes de recherche sont présentes à l'Observatoire : les Galaxies, les Hautes Energies et le Centre de Données astronomiques.

Galaxies

Les recherches et activités de l'équipe Galaxie couvrent des problèmes et des concepts variés concernant la formation des galaxies ainsi que l'étude et le recensement des populations stellaires qui constituent ces galaxies.

Plus particulièrement, cette équipe se concentre sur l'étude des propriétés physiques de notre galaxie et des galaxies voisines, notamment du Groupe Local. Ils observent non seulement les populations stellaires mais analyse aussi la dynamique gravitationnelle qui régit les mouvements des étoiles et des diverses matériaux les composants. Leur objectif est de réunir suffisamment d'information afin de pouvoir établir une histoire précise de l'évolution d'un système stellaire et ainsi reconstituer les évènement clés de la vie d'une galaxie.

Hautes Energies

L'équipe Hautes Energies consacre leurs travaux sur l'astrophysique des hautes énergies, notamment sur l'étude d'objets stellaires jeunes et d'objets compacts de notre galaxie .
Les chercheurs de cette équipe utilisent des instruments performants d'observation en rayons X présent a bord des satellites Chandra et XMM-Newton. ainsi que des télescopes terrestres munis d'instruments de haute qualité.

I.2 Le Centre de Données astronomiques de Strasbourg

I.2.1 Présentation

Le Centre de Données astronomiques de Strasbourg (CDS) créé en 1972 a pour objectif de collecter, homogénéiser, distribuer et préserver l'information astronomique pour le bénéfice de l'ensemble de la communauté scientifique internationale.

Ce service de l'Observatoire Astronomique est composé de 31 personnes parmi un effectif total de 80 employés. On compte 2 administratifs, 8 assistants ingénieur et ingénieurs d'études documentalistes, 3 CDD ingénieurs de recherche en informatique, 8 ingénieurs d'étude et ingénieurs de recherche en informatique, 10 enseignants chercheurs et chercheurs. Le CDS possède également sa propre infrastructure réseau et tous les postes de travail sont basés sur la distribution Linux Ubuntu.

Les services du CDS sont largement utilisés par la communauté astronomique internationale. La mission du CDS se décline en fait sur plusieurs types d'activités qui sont :

- Construction de services de référence à forte valeur ajoutée
- Définition de standards et d'outils génériques
- Participation à des projets
- Recherche & développement
- Diffusion des connaissances

C'est durant la période 2000 - 2003 qu'est apparu très rapidement le concept d'Observatoire Virtuel. Le CDS joue un rôle moteur dans la grande majorité des projets européens (AVO, Euro VOTECH, EuroVO DCA, EuroVO AIDA, EuroVO ICE, CoSADIE) liés à ce concept, mais aussi dans l'organisation de la communauté nationale autour de ce projet (actions spécifiques OV France).

I.2.2 Les services du CDS

Le développement de services de références constitue la base de l'activité du Centre de Données astronomique de Strasbourg.

En effet, le CDS développe et maintient trois services principaux de référence dans le milieu de l'astronomie. Bien que très différents à première vue , ces services sont étroitement liés et indispensables pour le travail des chercheurs de part leur utilisation simple et pratique.

a. Simbad

Simbad est la base de données de référence pour la nomenclature et la bibliographie des objets célestes. Plus de 7,3 millions d'objets et plus de 18 millions d'identificateurs y sont référencés.
Il s'agit aujourd'hui d'une référence au niveau mondial, des centaines de milliers de requêtes y sont effectuées chaque jours et cette base de donnée est sans équivalent à ce jour.

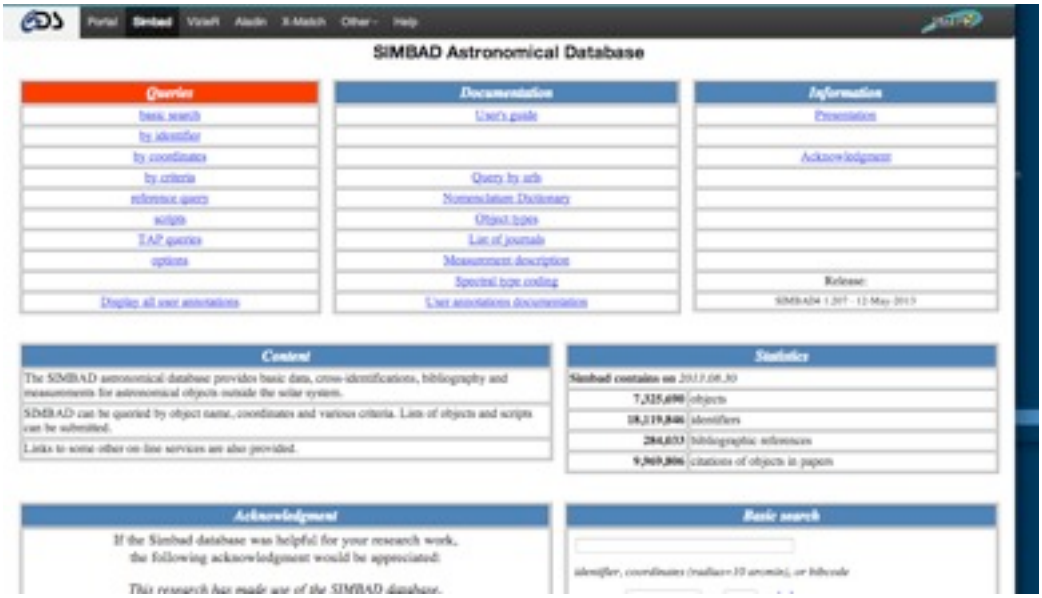


Figure 2: Interface de SIMBAD

Ce service permet, à partir du nom d'un objet astronomique, de retrouver des informations comme la position de celui-ci son type ou encore les mesures/études effectuées ainsi que l'ensemble des articles où il est référencé. D'autres méthode pour interroger la base sont également possible en faisant une requête sur une position bien précise ou ses références.

b. VizieR

VizieR est une base de données qui regroupe environ 9000 catalogue d'objets astronomiques. Ces catalogues sont des tables relevées durant les diverses observations et rentrées dans VizieR par les documentalites.

Ce service permet notamment à l'utilisateur d'accéder de manière homogène à des données hétérogènes de catalogue, de les exporter sous différents formats et de les comparer.
Les requêtes réalisées sur ce cette base de données peuvent porter sur des critères tel que la longueur d'onde avec laquelle les objets ont été observé ou encore la mission durant laquelle ils ont pu être détecté.

Figure 3: Interface de VizieR

c. Aladin

Aladin est un atlas du ciel interactif. Il fait parti des services les plus récent et est développé et maintenu par Pierre Fernique, Thomas Boch et Francois Bonnarel. Ce logiciel permet de visualiser des images de n'importe quelle partie du ciel, de charger et de superposer des objets issus des différents catalogues astronomiques ainsi que d'accéder aux informations des objets visualisé. Les catalogues d'images proviennent des bases de données du CDS mais également de certains catalogues internationaux.

Le logiciel est entièrement écrit en Java et est disponible en version dite «desktop», c'est à dire en version pour ordinateur de bureau» mais également sous forme «d'applet», petite application, sur le site du CDS et sur version mobile (Tablette) sous le nom de SkySurvey. Aladin est disponible librement et est distribué sous licence GPL.

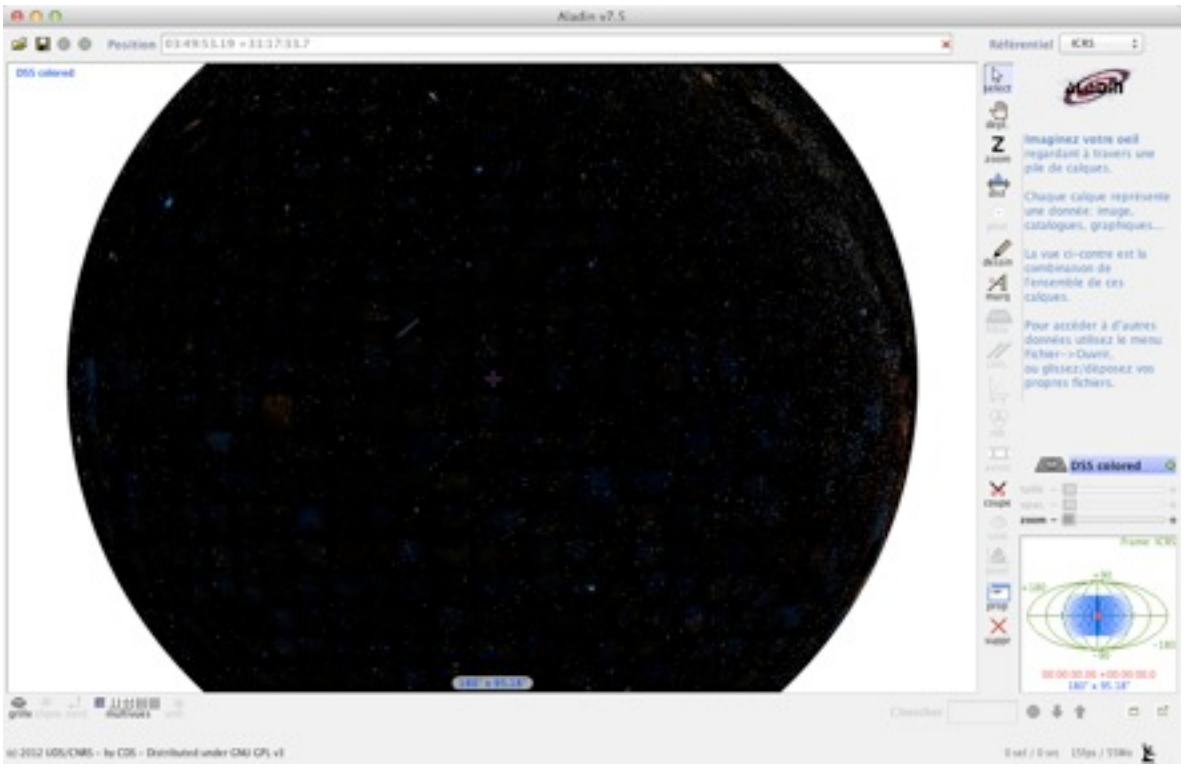


Figure 4: Interface du logiciel Aladin

I.3 International Virtual Observatory Allicance

I.3.1 Présentation

L'IVOA est une organisation internationale ayant pour but de faciliter les échanges et la collaboration entre chercheurs, nécessaire à l'élaboration de standards d'interoperabilité et au déploiement d'outils, de systèmes et de structures organisées afin de permettre une utilisation des archives astronomiques (images, catalogues, ...)

Figure 5: Logo de l'IVOA

L'IVOA compte a l'heure actuelle une vingtaine de participant parmi lesquels on peut citer l'Australie, la France, l'Allemagne, la Hongrie, l'Inde, le Brésil, l'Espagne, le Royaume Uni, le Japon, la Russie, l'Italie, la Chine ainsi que les Etats Unis. A noter également que l'ESA participe activement au développement de l'IVOA a l'internationale.

I.3.2 Le CDS au sein de l'IVOA

Le CDS est une infrastructure très reconnue qui joue un rôle important dans les collaborations internationales et au sein de l'IVOA. Il participe activement à la standardisation des infrastructures mais également à de nombreux travaux.

L'organisation de l'IVOA se fait par groupe de travail, chacun constitué de membres de plusieurs partenaires dont le but est de discuter des standards à mettre en place. Le CDS y joue un rôle primordial car il participe et co-dirige certains de ces groupes de travail :

- André Schaaff pour le groupe de travail Grid and Web Services.
-Mireille Louys est «vice-chair» du groupe de travail Semantics
- Pierre Fernique est «vice-chaire» du groupe de travail Application
- François Bonnarel est «vice-chair» du groupe de travail DAL

II. Déroulement du stage

II.1 Initialisation du stage

II.1.1 Sujet de stage

Le Centre de Données astronomique de Strasbourg a une forte activité de recherche et développement dans le secteur informatique depuis de nombreuses années afin de suivre les évolutions techniques en matière de hardware comme de software. Actuellement, il évalue, entre autres, l'utilisation des écrans multitouch sur terminaux mobiles (smartphone et tablette dont le nombre ne fait qu'augmenter depuis maintenant plus de 5 ans), avec les services qu'il propose.

C'est dans ce cadre que s'inscrit ce stage. L'objectif fixé par André, mon maître de stage, était d'évaluer les possibilités d'utilisation d'appareils dit multitouch (tactiles), smartphone ou tablette, pour amélioré l'utilisation, et donc l'expérience de l'utilisateur (UX) du logiciel Aladin, l'atlas interactif du ciel développé par le CDS. L'idée principale de ce projet était d'utiliser un protocole de communication déjà existant, permettant entre autre d'établir une connexion entre les différents services du CDS, afin de pouvoir piloter le logiciel à partir d'un appareil mobile et ainsi profiter de son écran multitouch pour réaliser diverses manipulations.
Le protocole en question se nomme SAMP : Simple Application Message Protocol et est un des projets résultant du programme de développement et de standardisation de l'IVOA.

Une fois l'appareil tactile connecté à l'ordinateur hôte, l'utilisateur doit être en mesure d'effectuer toutes les opérations de bases sur Aladin, sans avoir à basculer sur la version desktop.
Le développement d'une application mobile s'est tout de suite positionné comme solution pour répondre à cette demande. Devant la diversité des terminaux disponibles, en terme de systèmes d'exploitation, cette application se devait d'être multi-OS, «cross-platform» afin de permettre au plus grand nombre de personnes d'utiliser cette application.

Le sujet du stage s'est donc tourné vers le développement d'une application cross-platform reprenant les différents outils d'Aladin adaptés aux différents terminaux mobiles possédant le système d'exploitation Android ou iOS.

II.2.2 Moyens mis à disposition

Le travail réalisé lors des premiers jour de stage était axé sur l'analyse, la compréhension des besoins des utilisateurs des outils développés par le CDS. André Schaaff m'a tout de suite donné les différents éléments nécessaires pour «baliser» mon travail. Ainsi il a fallu dans un premier temps savoir utiliser Aladin, et faire une liste des différentes fonctionnalités de l'application. De plus, il était également nécessaire de prendre connaissance de SAMP, le protocole de communication utilisé par les différentes applications développées par le CDS, un outil clé dans le développement de l'application.

a. Codes Sources

Les code sources des logiciels tel que Aladin, mais aussi JSAMP, le hub SAMP permettant de gérer les connexions entre programme, étaient à ma disposition afin de mener à bien mon travail. L'accès a la console d'Aladin m'a permis de bien cibler les fonctionnalités clés de ce dernier ainsi que de réfléchir à la façon dont j'allais les implémenter dans l'application.

L'utilisation du hub JSAMP m'a permis de mieux comprendre le fonctionnement de SAMP lui même ce que ne me permettait pas le hub SAMP intégré nativement dans Aladin. J'ai pu ainsi faire une liste des possibilités mais également des limitations du hub.

b. Moyen de documentation

Le site du CDS regorge de documentations et de manuels d'utilisation concernant les différents services et logiciels qu'il propose. Une page de ce site est entièrement consacré à Aladin ce qui m'a permis de prendre connaissance avec les différentes fonctionnalités et possibilités de ce logiciel. De plus un document pdf (cf Annexes) contenait une description bien précise de l'API et de son utilisation.

Concernant SAMP, il s'agit d'un protocole interne à l'IVOA d'où une très faible quantité d'information disponible sur les forum d'entraides, mais j'ai tout de même pu trouver quelques présentations qui m'ont été d'une grande aide afin de maîtriser au mieux l'établissement de connexion entre logiciel via ce protocole. C'est également à ce moment là que j'ai pu me rendre compte de certaines limitations relativement handicapantes. En effet la documentation précisait que la connexion de terminaux externes au localhost était impossible du aux diverses sécurités mis en place par les développeurs.

Le site de PhoneGap contient toute la documentation nécessaire pour bien commencer à développer une application mobile, que ce soit sur iOS ou Android. De plus PhoneGap appartenant désormais a Adobe, j'ai pu trouvé un bon nombre de présentations faites par les ingénieurs de la société.Malheureusement la documentation officielle étant parfois dépréciée, j'ai du me rediriger vers des forum pour avoir des solutions à certains problèmes comme désactiver la possibilité de faire défiler une page ou encore changer les icônes et «splashscreens» par défaut.

c. Contacts et aides

Les différentes limitations du protocole SAMP pouvant mettre en péril le projet, André a pu me mettre en relation avec Mark Taylor, Astronome et développeur de l'Université de Bristol, à l'origine du projet SAMP. Mark m'a été d'une grande aide notamment concernant la sécurité mise en place par le hub. J'ai également eu droit, après lui avoir expliqué mon problème, à une version améliorée du hub permettant l'ajout de clients externes au localhost. Ainsi voyant la nécessité d'assouplir les normes de sécurité du hub, Mark sortira une nouvelle version d'ici peu de temps, permettant la connexion d'appareils externes pour tous.

L’équipe en charge d’Aladin a également pu m’aider. En effet certaines fonctionnalités qui n’étaient pas disponible nativement, comme l’envoi d’une position de Aladin vers un client SAMP, pouvant être un vrai handicap lors de l’utilisation de l’application, un des développeur du logiciel, Thomas Boch, a pu les ajouter afin de les intégrer à l’API.
Moyens matériels

Dans le cadre du développement de l’application, on m’a fournit dans un premier temps une tablette Nexus 10 équipée d’Android 4.2 afin de développer au mieux la version Android de l’application. Du à l'indisponibilité des services développeur Apple j’ai malheureusement du me contenter du simulateur iOS disponible sous OSX dans un premier temps, je n’ai pu mettre la version iOS sur mon téléphone qu’à la toute fin de mon stage.

II.2 Eléments techniques

Cette partie a pour but de détailler les différents éléments techniques clés de ce stage.

II.2.1 Le besoin

Les terminaux tactiles possédant des écrans de plus en plus précis et performants, leur utilisation pour le contrôle d’un logiciel tel que Aladin semble apporter un certain nombre d’avantages. Le déplacement sur la carte du ciel se ferait au doigt ce qui présente une plus grande flexibilité, les déplacements à la souris sur la version bureau sont souvent laborieux. De plus, la possibilité de contrôler Aladin à distance aurait une grande utilité lors des réunions et conférences de l’observatoire, et permettrait ainsi d’afficher la carte du ciel sur un écran de plus grande taille.

II.2.2 Le protocole SAMP (Simple Application Messaging Protocol)

Le protocol SAMP, développé dans le cadre de l’IVOA est un successeur de PLASTIC (Platform for Astronomy Tool InterConnection) le premier protocole de connexion entre les différents services du CDS.

a. Concept clé: le hub

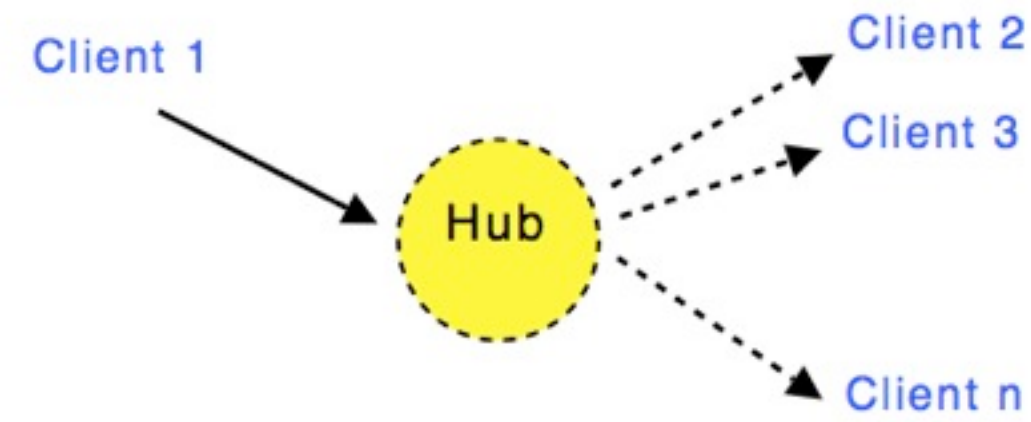


Figure 6: Schéma représentant le fonctionnement du Hub

Le «Hub» peut être intégré à l'application utilisée ou peut être une application à part entière. Il réalise la jonction entre les différents services et permet la transmission et la réception de messages entre les différents clients ainsi que la connexion de plusieurs clients (cf Figure 6).
Chaque client se voit ainsi assigner, une fois connecté au hub, un ID unique, ils déclarent eux même leur metadata (informations personnelles), tel que le nom de l'application souhaitant se connecter ou encore une rapide description.

Cycle de vie d'un client:

1. Vérifie si le hub est fonctionnel.
2. Si oui, établissement d'une communication avec le hub.
3. Connexion au hub.
4. Envoi des metadata.
5. Interroge le hub pour savoir quel autre client y est connecté.
6. Envoi/Réception des messages via le hub.
7. Déconnexion.

Cycle de vie du hub:

1. Vérifie si il n'y a pas d'autres hubs lancés sur la machine.
2. Se lance si aucun autre hub est détecté.
3. Attend la connexion d'un client.
4. Lors de la connexion d'un client, le hub réceptionne les metadata.
5. Lorsque le hub reçoit un message il le transfert au bon destinataire.
6. Attend la déconnexion des clients.
7. Si le hub n'est plus en mesure de communiquer avec un client, il le déconnecte.

8. Fermeture du hub.

Le hub comprend également différents modes appelés «Profiles». Le «Standard Profile» est le mode de base du hub permettant la communication entre les différents services du CDS sur une même machine. Or notre but étant de connecter un appareil distant via SAMP, je me suis plus intéressé au «Web Profile», permettant de connecter des petites web apps à la gamme de logiciels compatibles SAMP.

Les «Messages» sont des chaines de caractères envoyées par l'utilisateur. Il contient l'information que l'on veut envoyer à une autre application. Un message SAMP est composé des éléments suivants :

samp.mtype: Un mType est une chaine de caractères définissant le type de message à envoyer au client. Chaque message contient exactement un mType, et le message est délivré qu'aux client abonnées à ce mType.

samp.params: La valeur params contient les paramètres du message envoyé à l'application.

Exemple de message SAMP :

samp.mtype: script.aladin.send
samp.params: script : reticle off

Ce message désactive le réticule dans Aladin.

Une liste des différents mTypes, ainsi que leur paramètres respectifs, inclus dans l'API d'Aladin est disponible en fin de rapport.

b. Envoie et réception de message

SAMP présente la possibilité de communiquer avec les diverses applications compatibles de plusieurs manières différentes. En effet chaque pattern de messagerie correspond a un moyen de communication bien particulier et doit être utilisé en fonction de la réponse que l'on veut obtenir.

Pour le développement de l'application je me suis concentré sur 2 types de messages : «notification» et «call».

Notification: Il s'agit d'un moyen de communication unidirectionnel dont le but est d'envoyer une instruction à l'application sans attendre de réponse.

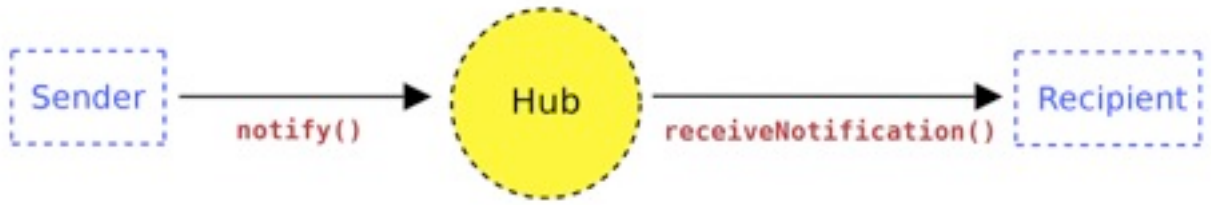


Figure 9: Schéma représentant le fonctionnement d'un message de notification

1. L'envoyeur envoie un message au hub dans le but de le délivrer à un ou plusieurs récepteurs.
2. Le hub transfert le message aux récepteurs
3. Aucune réponse n'est envoyée à l'envoyeur.

Call: Il s'agit d'un moyen de communication asynchrone, permettant la réception d'une réponse.

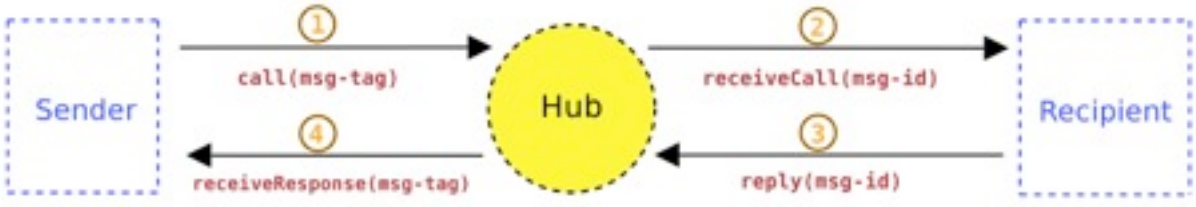


Figure 10: Schéma représentant le fonctionnement d'un message d'appel

1.L'envoyeur envoie un message au hub dans le but de le délivrer à un ou plusieurs récepteurs, prenant en argument un «msg-tag». Le message retour contient donc un unique identifier associé au «msg-tag» appelé «msg-id».

2. Le hub transfert le message aux récepteurs lui envoyant en paramètre le «msg-id» associé au «msg-tag» du message entrant.

3. Chaque récepteur, envoie à son tour un message de réponse au hub avec en argument le «msg-id».

4. En utilisant un callback le hub envoie la réponse à l'envoyeur à l'origine du message.

Comme l'application que je développais ne devait communiquer qu'avec un seul récepteur, Aladin, j'ai utilisé pour plus de confort et de clarté dans le code les protocoles «notifyAll» et «callAll» ne nécessitant pas de préciser l'identificateur du récepteur.

II.2.3 La librairie SAMPJS

SAMPJS est une librairie Javascript créée par une équipe de développeur de l'IVOA. Cette librairie permet de faire la jonction entre les technologies du web, HTML5 Javascript, et le protocole SAMP que je devais utiliser. Mark Taylor faisant également parti de ce projet je n'ai pas hésité à lui poser plusieurs questions concernant l'utilisation des diverses fonctionnalités qu'offrait SAMPJS.

La librairie étant relativement simple d'utilisation, et le javascript un langage que j'avais déjà rencontré au cours de certains projets personnels, je n'ai pas eu de mal à développer les fonctions de base nécessaires au bon fonctionnement de l'application tel que la connexion au hub via le «Web Profile», l'envoi d'instruction simple à Aladin, etc.

Voici la structure d'une fonction permettant l'envoi d'un message au hub SAMP écrite en javascript:

```
var allsky = function(){
    var message = new samp.Message("script.aladin.send",
                                   {"script": "get allsky"});
    connector.connection.callAll(["script.aladin.send", message]);
};
```

Figure 11: Extrait du code source de SkyTouch / Fonction allsky

On reconnait la structure d'un message décrite en II.2.1.b, le mType : script.aladin.send et le params : script: get allsky. Enfin, on envoie notre message par un callAll,nécessaire lors de l'envoi de scripts.
(Cette fonction est utilisée dans l'application pour afficher l'ensemble du ciel sur Aladin depuis le téléphone)

La variable «connector», appelant la méthode «connection» dans le message précédent, est déclarée lors de la connexion de l'appareil mobile au hub SAMP de la façon suivante:

```
var meta = {
    "samp.name": "SkyTouch",
    "samp.description": "Sky position touch controller"
};
var subs = cc.calculateSubscriptions();
 var connector;
    $('#registerBtn').click(function() {
        var ipnew = document.getElementById("ipenter").value;
        ip = ipnew;
        connector = new samp.Connector("SkyTouch", meta, subs);
        samp.register(connector.name, function(conn) {
            connector.setConnection(conn);
            $('#registerBtn').attr('disabled', 'disabled');
        }, null);
    });

var ip = "localhost";
```

Figure 12: Extrait du code source de SkyTouch / Metadata et fonction de connexion au hub

Comme décrit dans la section II.2.1.a, les metadata sont gérées au sein de l'application. Ici on les déclare dans une variable «meta», contenant a la fois le nom de l'application et une description rapide. La variable connector va quant à elle initialisé une nouvelle connexion samp pour l'application SkyTouch, contenant les metadata «meta» et cette connexion sera de type «subscription», c'est à dire inscription au hub SAMP lancé sur la machine hôte.

II.3 Premiers prototypes

II.3.1 Premiers tests sur navigateur

Prise en main rapide

Etant donné le fait que le développement de l'application se concentrait uniquement sur des technologies du web, j'ai pu réaliser les premiers tests et aussi prendre mes marques avec SAMP et SAMPJS à partir d'un navigateur internet. J'ai beaucoup utilisé Chrome, qui, par expérience, est une valeur sûr concernant la compatibilité des différents standards du web et qui de plus, intègre WebKit, le moteur de rendu de page Web utilisé également par Phonegap.

Afin de pouvoir faire mes tests dans les meilleurs conditions, j'ai développé un petit serveur http en Node.js, un framework évènementiel permettant d'exécuter du Javascript coté serveur.

Ci dessous une des premières ébauches fonctionnelles de SkyTouch:

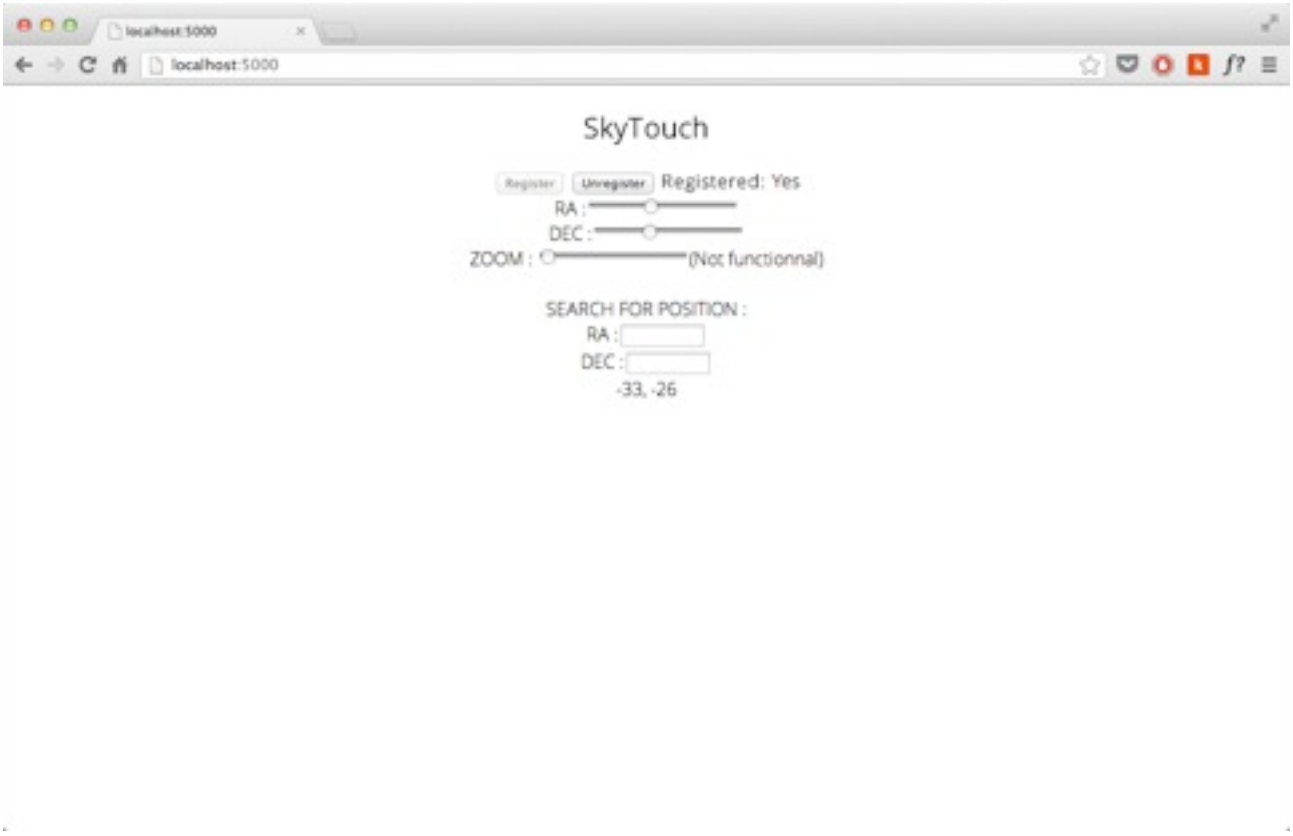


Figure 13: Première version test de SkyTouch

Cette première version permettait une connexion locale au hub SAMP, la modification manuelle, via des réglettes, de la position ainsi que la recherche d'élément en entrant leur position. Le zoom n'était pas fonctionnel en raisons de limitations de l'API liées à la sécurité.

II.3.2 Utilisation complète de l'api et ajout du support tactile

a.Trackpad tactile

Un des principaux enjeux du développement de l'application SkyTouch était d'apporter un support tactile à l'utilisation d'Aladin. En effet, le multi-touch apporte une expérience utilisateur bien plus agréable que l'utilisation d'une souris ou d'un Trackpad classique lorsqu'il s'agit d'effectuer des mouvements ou des déplacements sur le logiciel.
Afin d'apporter à mon application cette surcouche tactile, j'ai décidé de chercher sur GitHub, le très célèbre réseau social pour développeurs, si il existait une librairie gérant la gestuelle pour des web-app. C'est alors que j'ai découvert Hammer.js, une librairie javascript, répondant exactement à la problématique et ayant également une documentation bien fournis.
Chaque geste effectué à l'écran est catégorisé : tap, swipe right, swipe left, drag up, drag down, hold, rotate, pinch, etc, et est considéré comme un évènement. Ci-dessous un exemple tiré de la documentation de Hammer.js :

```
var element = document.getElementById('test_el');
var hammertime = Hammer(element).on("tap", function(event) {
    alert('hello!');
});
```

Figure 14: Extrait de la documentation de Hammer.js / Fonction basique affichant un message lors d'un geste

On déclare une zone tactile «test_el» que l'on active avec «Hammer(element).on» prenant en argument un des gestes disponibles et un évènement, ici il s'agit d'un simple «alert». Chaque évènement contient plusieurs propriétés liées au geste réalisé. Cette liste de propriétés étant très bien fournie, elle permet d'obtenir simplement toutes les informations nécessaire lors d'un évènement (cf Figure 15).

```
timestamp    {Number}        time the event occurred
target       {HTMLElement}   target element
touches      {Array}         touches (fingers, mouse) on the screen
pointerType  {String}        kind of pointer that was used. matches Hammer.POINTER_MOUSE|TOUCH
center       {Object}        center position of the touches. contains pageX and pageY
deltaTime    {Number}        the total time of the touches in the screen
deltaX       {Number}        the delta on x axis we haved moved
deltaY       {Number}        the delta on y axis we haved moved
velocityX    {Number}        the velocity on the x
velocityY    {Number}        the velocity on y
angle        {Number}        the angle we are moving
direction    {String}        the direction we are moving. matches Hammer.DIRECTION_UP|DOWN|LEFT|RIGHT
distance     {Number}        the distance we haved moved
scale        {Number}        scaling of the touches, needs 2 touches
rotation     {Number}        rotation of the touches, needs 2 touches *
eventType    {String}        matches Hammer.EVENT_START|MOVE|END
srcEvent     {Object}        the source event, like TouchStart or MouseDown *
startEvent   {Object}        contains the same properties as above,
                             but from the first touch. this is used to calculate
                             distances, deltaTime, scaling etc
```

Figure 15: Extrait de la documentation de Hammer.js / Liste des informations obtenues lors d'un geste

Suite à la prise en main de cet outil, j'ai décidé d'utiliser Hammer.js afin de développer un trackpad tactile en HTML5 et Javascript permettant ainsi de déplacer le curseur de Aladin au doigt depuis l'appareil distant. La fonction est très simple grâce à l'utilisation des informations deltaX et deltaY permettant d'obtenir une valeur représentant le déplacement du doigts sur l'écran selon les axes x et y. On envoie cette valeur dans un message SAMP contenant le mType coord.pointAt.Sky prenant en argument l'ascension droite ra (abscisse) et la déclinaison de (ordonnée).

b. Déblocage de l'API

Comme signalé lors de la présentation de SAMP, le protocole affiche un certain niveau de restrictions parfois élevées concernant l'envoi de certains messages. En effet certains mTypes étaient impossible à utiliser pour des raisons de sécurité, notamment le mType script.aladin.send permettant l'envoi de script via SAMP, bloquant ainsi un bon nombre de fonctionnalités.

Afin d'y remédier, il est nécessaire de désactiver toutes les restrictions de type mType pour le «Web Profile», la mise à jour du hub SAMP par Mark Taylor devrait retirer les restrictions par défaut. Ainsi, j'ai été capable d'implémenter de nombreuses fonctionnalités suite au déblocage des diverses restrictions. Voici la liste de ces dernières :

- Trackpad tactile
- Zoom
- Gestion des vues
- Gestion de la grille
- Barre de recherche
- Gestion de la précision
- Chargement de tableau
- Affichage du ciel

Figure 16: Version test de SkyTouch suite au déblocage de l'API

II.4 Développement sur de l'application

II.4.1 Petite présentation de PhoneGap

PhoneGap est un framework gratuit et open source permettant de créer des applications mobiles en utilisant les différents standards du web que sont HTML5 CSS3 Javascript/

Jquery. Ce framework présente également l'avantage d'être «cross-platform», c'est à dire qu'il présente la possibilité de compiler l'application à la fois pour Android, iOS, Windows Phone 8, etc.

Figure 17: Schéma représentant le fonctionnement de PhoneGap

L'utilisation de ce framework, à la fois sur Android et sur iOS m'a permis de développer facilement une application adaptée sur tablette et téléphone.

Problèmes rencontrés et astuces :

Bien que simple d'utilisation, et permettant une réduction du temps de développement non négligeable, le choix de technologies web peut poser certains problèmes. En effet, l'une des principales critiques que l'on peut apporter à PhoneGap est la lenteur des applications produites. En effet l'utilisation abusive d'animation JQuery peut grandement alourdir l'application et ainsi la ralentir lors de son utilisation. Une des solutions et de ne pas utiliser JQuery et d'opter plutôt pour des animations CSS, beaucoup plus fluides et surtout moins énergivore (Je n'ai pas eu ce problème lors du développement, l'application ne comporte, pour le moment, aucune animations car elle se devait avant tout d'être fonctionnelle).
Un autre soucis majeur a été la réactivité des boutons. La solution que j'ai trouvé afin d'évité ce délai est de modifier tous les évènements «onclick» en évènement «ontouchstart». En effet, un évènement «onclick» comprend un délai de 300ms alors qu'un évènement «ontouchstart» est quasi instantané.

Mis à part ces deux problèmes majeurs, mon expérience avec PhoneGap a été très agréable de part la facilité de la mise en place de l'environnement de développement mais aussi grâce aux différents tutoriaux disponibles en ligne qui m'ont été d'une très grande aide.

II.4.2 Développement sur Android

a. Adaptation rapide

N'ayant jamais eu l'occasion de développer sur Android par le passé, j'ai dû dans un premier temps mettre en place mon environnement de développement Android sous Eclipse. Là ou Apple exige

d’utiliser son propre système d’exploitation pour développer sur ses produits , Android présente l’avantage d’être utilisable sur n’importe quel OS grâce à un environnement incontournable, Eclipse.On peut également mentionner la sortie assez récente d’Android Studio, un IDE développé par Google entièrement consacré au développement d'applications Android. Malheureusement ce dernier était trop jeune pour être utilisable en production, je me suis donc contenter de Eclipse.

Il faut d’abord télécharger et installer le SDK d’Android (http://developer.android.com/sdk). Une fois ceci fait on peut importer le projet PhoneGap dans Eclipse en se rendant dans le menu d’importation de projet et en sélectionnant un projet de type application Android.

b. Compilation et premiers tests

Sur Eclipse, la procédure pour compiler un projet Android est presque identique à la compilation d’un projet Java. Il suffit dans un premier temps de choisir l’appareil ciblé, l’appareil sur lequel on va lancer l’application.

Le mieux quand on développe sur Android est de disposer d’une tablette ou d’un smartphone pour réaliser les tests. Si ce n’est pas le cas on peut toujours créer et configurer un simulateur, ou machine virtuelle, sous Android. Malheureusement, celle-ci est extrêmement lente, et cela même quand il faut virtualiser une ancienne version du système. Avec le SDK, on a la possibilité de configurer son ou ses simulateurs grâce à l’éditeur de machines virtuelles. On peut y choisir exactement le matériel que l’on utilise ainsi que la taille de l’écran, ce qui permet d’être sur du bon rendu de l’interface de l’utilisateur.
Je n’ai, heureusement, pas eu besoin d’utiliser le simulateur Android bien longtemps, une Nexus 10 neuve sous Android 4.2 a été livré a temps pour que je puisse tester SkyTouch directement sur le terminal tactile. Il m’a suffit de connecter la tablette en USB, la tablette a été immédiatement reconnu par Eclipse, je n’ai donc eu aucun soucis.

b. Premiers résultats

A ma grande surprise, l’application s’est lancée sans problème dés la première compilation sous Eclipse. L’interface était néanmoins très petite, les dimensions de la tablette étant très généreuses (2560x1600), il a donc fallu que je la remanie et que je revois aussi l’ergonomie de l’application, certains boutons étant difficilement accessibles, les champs de texte trop petits et l’interface peu intuitive. Quant à la connexion à la machine hôte et l’interaction avec Aladin, je n’ai pas été déçu, bien au contraire. Les messages étaient bien transmis à la console d’Aladin et étaient exécutés quasi instantanément.

II.4.3 Développement sous iOS

a. Un travail en grande partie sur simulateur

Le site de développeur Apple ayant été indisponible pendant plus d'un mois, j'ai été dans l'impossibilité d'enregistrer mon appareil sur le compte développeur du CDS. Il m'était ainsi impossible d'utiliser mon propre téléphone pour réaliser les différents tests sur l'application. Je me suis donc contenté du simulateur iOS livré avec la dernière version de XCode, l'environnement de développement disponible sur OSX. Contrairement à Eclipse, le simulateur de XCode est beaucoup plus performant et réaliste. Il ne présente aucune lenteur et la gestuelle à la souris est très bien interprétée par le logiciel. Ceci m'a permis de réaliser les différents tests et ajustements nécessaires au bon fonctionnement de l'application tout en étant sûr du résultat. En effet, contrairement à Eclipse, l'application compilée sur XCode est identique à celle testée sur simulateur, tant sur la partie visuelle que sur la partie fonctionnalité. Ainsi, une fois les résultats sur simulateurs acceptables, il m'a fallu attendre quelques semaines pour tester SkyTouch sur un vrai iPhone.

b. Quelques difficultés

XCode est un environnement de développement agréable voire plus agréable qu'Eclipse notamment au niveau du simulateur iOS, mais il présente néanmoins un très gros défaut lorsqu'il s'agit d'enregistrer un appareil sur un compte développeur. En effet la procédure est très compliquée. La où sur Eclipse il suffisait de connecter sa tablette en USB et de compiler, la procédure sur XCode comprend un bon nombres d'étapes intermédiaires:

- Enregistrement du compte développeur
- Connexion au compte développeur
- Aller dans le menu du compte développeur
- Enregistrer son appareil sur le compte développeur
- Attribution d'un certificat pour l'appareil
- Attribution d'un certificat pour l'application
- Compilation

N'ayant jamais réalisé la procédure auparavant, André et moi même avons eu beaucoup de mal à transférer l'application sur son iPhone. Concernant mon téléphone la difficulté fut encore plus grande dans la mesure ou je possédais une version béta du futur système d'exploitation mobile d'Apple iOS7 dont le SDK était uniquement compatible sur la dernière version béta de XCode (Xcode 5.0 béta). Il a donc fallut refaire toute la procédure d'enregistrement sur cette version.

c. Premiers résultats

Comme pour la version Android, j'ai été surpris par le fonctionnement de l'application sur iOS. En effet celle-ci, bien que tournant sur un appareil bien moins puissant que la Nexus 10, semblait plus rapide et beaucoup plus performante en terme d'envoi de message a Aladin. Quelques remaniements mineures étaient nécessaires, notamment refaire les «checkboxes» (boites à cocher) dans les options de l'application, qui visiblement ne fonctionnaient pas sous iOS alors qu'elles ne présentaient aucun problème sur Android.

III.Bilan

III.1 L'application finale : SkyTouch v0.9

III.1.1 Version actuelle

N'étant au départ qu'une simple idée de mon maître de stage, André Schaaff, j'ai été très heureux de voir que, petit à petit au cours du développement, SkyTouch prenait forme. Au final j'ai réussi à produire une application fonctionnelle remplissant la grande majorité des objectifs tout en la rendant disponible sur les deux systèmes d'exploitation mobile dominant le marché actuellement. Au final voici une liste des diverses fonctionnalités que permet SkyTouch dans sa version actuelle :

- Connexion au hub SAMP et a Aladin (nécessite pour le moment une version spéciale du hub développée par Mark Taylor)
- Affichage du ciel
- Recherche d'éléments: en utilisant leur nom courant, leurs différents identificateurs ou encore leur position
- Trackpad tactile permettant de se déplacer dans le ciel
- 6 niveaux de zoom avec adaptation de la précision du Trackpad en fonction du niveau de zoom
- Gestion des vues: possibilité de diviser l'écran d'Aladin jusqu'à 9 vue indépendantes et possibilité de choisir sur quelle vue travailler.
- Afficher/Enlever la grille
- Afficher/Enlever le réticule
- Chargement de base de données: j'ai pu intégrer les bases de donnée SIMBAD du CDS et NED de la NASA à l'application afin de charger les différents éléments de la base lors d'une recherche
- Gestion des colorations: 11 colorations sont disponibles avec la possibilité de les gérer (afficher la coloration, effacer la coloration)
- gestion des Macros: possibilité de créer/effacer ses propres boutons a partir des commandes de l'API Aladin (dans le but de préparer une présentation par exemple)

Des captures d'écran de l'application dans sa version actuelle sont disponibles en annexe de ce rapport.

III.1.2 Les futures versions

Comme certaines fonctionnalités, implémentées lors du développement mais retirées de la version 0.9, nécessitaient l'utilisation de la version bêta du futur Aladin encore légèrement instable, j'ai décidé d'attendre un peu avant de les rendre accessible dans l'application. En effet, la version 1.0 de l'application devrait comporter plusieurs fonctionnalités supplémentaires. Elle sera disponible le jour de la sortie de la prochaine version d'Aladin mais également lorsque la prochaine version du hub SAMP, comportant moins de restrictions que la version actuelle, sera disponible. Ainsi l'application sera beaucoup plus facile d'utilisation notamment en ce qui concerne la mise en place du hub.

Voici une liste de quelques fonctionnalités qui seront implémenter pour la version 1.0 ou pour des versions ultérieures:

- Calibration automatique du Trackpad tactile suite à une recherche (1.0)
- Possibilité de prendre des captures d'écran de Aladin depuis l'application (1.0)
- Sauvegarde automatique des captures d'écran dans l'album photo du téléphone
- Ajout de nouvelles base de données (1.0)
- Meilleurs contrôle des colorations (1.0)
- Mise en place d'un «local storage» afin de garder en mémoire les informations et les paramètres de l'utilisateur
- Possibilité de charger des tableaux de données depuis l'application

Il me fallait en moyenne 3 à 4 heures pour implémenter une fonctionnalités. Certaines ne nécessitaient que quelques minutes de réflexion, d'autres pouvaient me prendre plusieurs journée à implémenter.

Les fonctionnalités de la version 1.0 sont déjà développées et présentes au sein du code source de l'application.

III.2 Conclusion et perspectives

L'objet du stage était de déterminer comment adapter les différentes fonctionnalités du logiciel Aladin à la fois sur iOS et Android afin de créer une interface tactile permettant de le contrôler à distance. Afin de répondre à cette problématique, je me suis tout d'abord intéressé au protocole de communication développé par l'IVOA : SAMP, pour tenter dans un premier temps de comprendre son fonctionnement puis de lister les différentes possibilités qu'il offrait. Après une étude en détail de l'API d'Aladin et de la documentation de SAMPJS, une API Javascript, j'avais tous les éléments en main pour développer une application fonctionnelle.

Malgré l'utilisation de technologies web pour permettre un développement cross-plateform rapide, les performances et la réactivité des commandes envoyées à distance de l'application vers Aladin

via le hub SAMP sont bien là, grâce à diverses astuces trouvées dans la documentation et quelques présentations des ingénieurs de Adobe.

La version actuelle de SkyTouch est toujours une bêta, mais une version 1.0 distribuable est envisageable une fois que les autres logiciels, le hub SAMP et Aladin, seront mis à jour afin de faciliter l'utilisation de l'application, notamment en ce qui concerne la phase d'établissement de la connexion de l'appareil mobile vers le hub, qui pour le moment n'est pas «user-friendly» (un lancement du hub par ligne de commande est nécessaire). De plus, l'utilisation qui sera faite de SkyTouch est toujours en discussion, sera t-elle uniquement réservé au personnel de l'Observatoire pour des présentations sur Aladin ? Ou pourra-t-elle être déployée à plus grande échelle dans des domaines tels que l'éducation ?

Je suis très satisfait de mon stage. Ce dernier s'est très bien déroulé, j'ai pu travaillé dans un domaine qui m'intéressait beaucoup tout en ayant l'occasion de me perfectionner dans les différents langages de programmation que j'ai pu utiliser pour réaliser cette application. J'avais également la chance d'être au sein d'une équipe qui a su m'aider lorsque j'avais des problèmes mais également traiter mes demandes lorsque j'avais besoin d'une fonctionnalité particulière que ce soit pour Aladin ou pour le hub SAMP.

De plus j'ai eu le privilège de faire une présentation, en anglais, de SkyTouch lors de la conférence CoSADIE tenu le 2 septembre à l'Observatoire Astronomique de Strasbourg. L'application a visiblement intéressé le public et à soulever des interrogations similaires à celle évoquer précédemment. L'application sera également présentée par André lors de l'InterOp, une conférence de l'IVOA tenu cette année à Hawaii regroupant des développeurs et chercheurs de plusieurs pays.

Ce stage fut l'une de mes premières expériences dans le monde du travail, mais aussi dans le monde de la recherche au sein d'un laboratoire, un domaine qui m'était jusqu'alors inconnue. Il m'a permis de prendre conscience de l'importance de l'outil informatique au sein de la recherche, celui-ci devient un outil indispensable et se doit d'assister au mieux les scientifiques ou dans le cas de l'Observatoire Astronomique de Strasbourg , les astronomes.

Glossaire

Aladin: Application développé en Java, atlas 3D, interactif du ciel étoilé.

Android: Système d'exploitation open source pour smartphones et tablettes, conçu par Android, une start-up rachetée par Google.

Bêta: Se dit d'un logiciel en production juste avant la fin du développement de la première version.

CDS: Centre de Données astronomiques de Strasbourg, collecte homogénéise, distribue et préserve l'information astronomique pour le bénéfice de l'ensemble de la communauté astronomique.

Chandra: Télescope à rayon X lancé en 1999 par la navette spatiale Columbia.

CNRS: Centre National de la Recherche Scientifique.

Code source: Ensemble d'instructions écrites dans un langage de programmation informatique de haut niveau, compréhensible par un être humain entraîné permettant d'obtenir un programme pour un ordinateur.

Compilation: Travail réalisé par un compilateur qui consiste à transformer un code source lisible par un humain en un fichier binaire exécutable par une machine.

Eclipse: Environnement de développement principalement utilisé pour développer en Java.

ESA: European Space Agency (Agence Spatiale Européenne).

GPL: General Public License

iOS: Système d'exploitation pour smartphones et tablettes conçu par Apple.

Local Storage: méthode consistant à stocker des informations d'utilisateurs depuis HTML5.

Multi-Touch: Un dispositif multi-touch est une technique d'interaction Homme-Machine par le biais de plusieurs points de contact, souvent avec plusieurs doigts

OSX: Système d'exploitation pour ordinateur développé par Apple.

SAMP: Simple Application Message Protocol, protocole de communication conçu dans le cadre de l'IVOA dont le but est de permettre à tous les logiciels du CDS de communiquer ente eux.

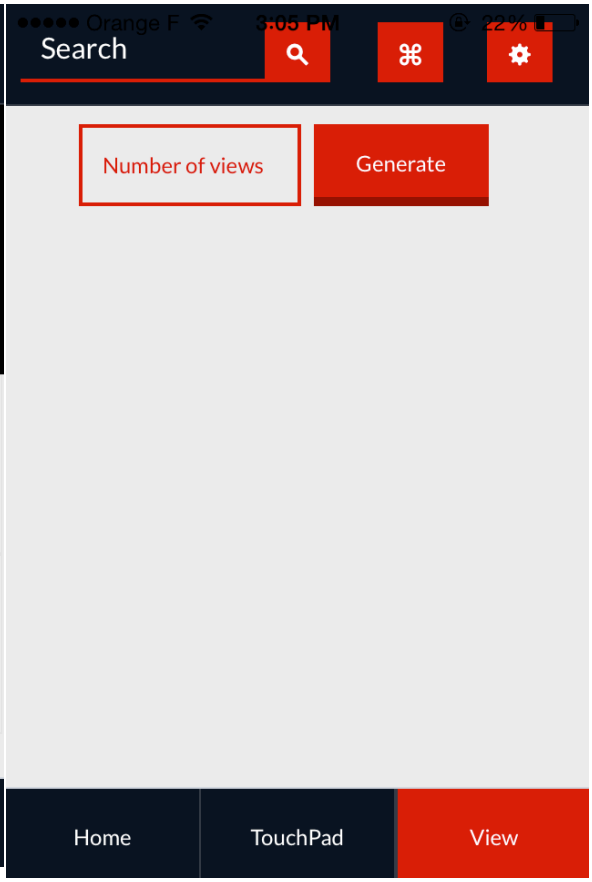
SkyTouch: Application Android et iOS permettant le contrôle a distance de Aladin.

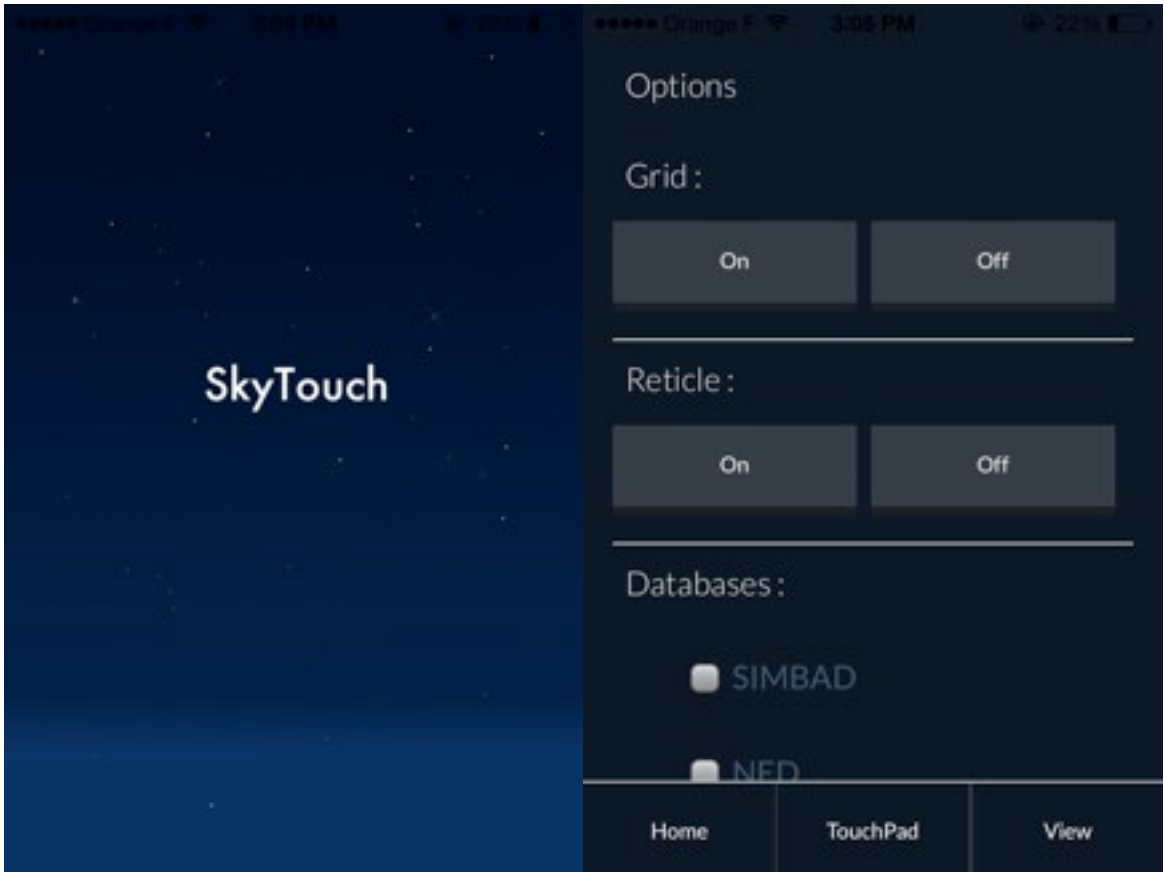
XCode: Environnement de développement créé par Apple et intégré à OSX, principalement utilisé pour développer des applications iOS ou OSX.

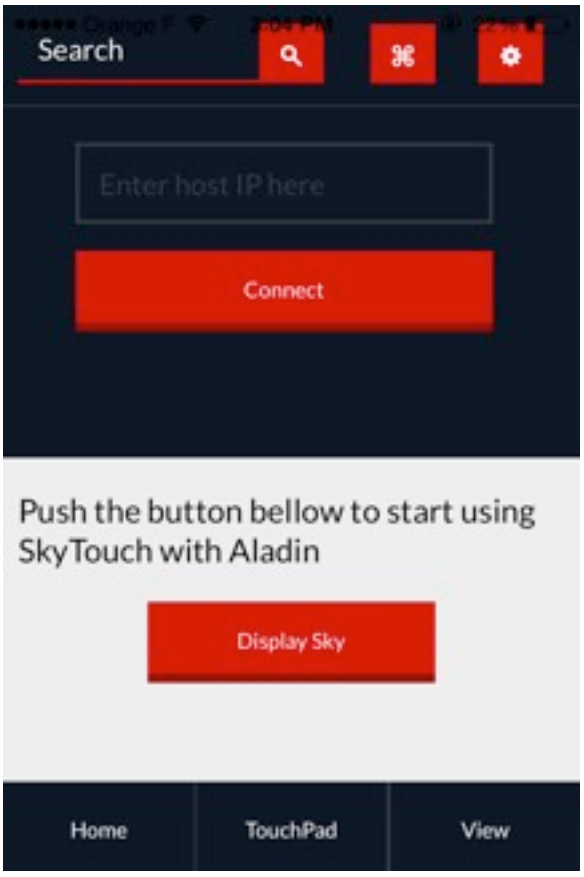
XMM-Newton: Satellite artificiel d'observation des rayons X lancé par l'ESA en 1999.

Annexes

Captures d’écran de SkyTouch




SkyTouch
Options
Grid :
On
Off
Reticle :
On
Off
Databases :
SIMBAD
NED
Home
TouchPad
View


Search
Enter host IP here
Connect
Push the button bellow to start using SkyTouch with Aladin
Display Sky
Home
TouchPad
View

Extrait de la notice de Aladin: l'API de Aladin

4.6. Script console and script mode

Open the Aladin script console from the Tools... menu: all ac- tions or commands sent and correctly executed since the begin- ning of your Aladin session are printed in the Aladin console (the echoed lines are bracketed).

In order to use Aladin in script mode, Aladin can be controlled via in-line commands. These commands have to be submitted to the Command> in the console window (see Fig. 21). The detailed help on script commands is obtained from the Help menu, and is also reported in the section 4.6 and Appendix B. Use the keyboard arrows to call back or mod- ify any previously executed command. Files containing a list of command are submitted to the console by using the script command load *file*.

The main basic script command is:

Get ServerName[,ServerName] Target

1 – **ServerName** can be *Simbad, NED, Aladin* or *VizieR*. For this last one, the catalogue specification is required (in parentheses). For Aladin, the survey, colour, or resolution can be specified as well between parentheses.

2 – **Target** can be a Simbad identifier or J2000 coordinates in sexagesimal syntax.

If the *Target* is omitted, Aladin takes into account the last specified target. If, for an initial query, there is only a *Target* and no *ServerName*, Aladin will load a default set (currently: DSS-I high resolution image, SIMBAD and GSC1.2).

You will find in the FAQ of the interactive HELP menu detailed possibilities for the syntax of shortcut commands.

Examples:

get Simbad,VizieR(GSC1.2) M 81
get Simbad 00 42 44.10 +41 16 08.8
get Aladin(DSS2),VizieR(USNO2) NGC 7436
get Aladin(DSS2),NED 18 19 -13 50
get Skyview(Rosat) M 1
get SSS.img(UKST) M 2
Typing a Simbad object name or J2000 coordinates, the reticle shifts to this position.

Script commands are submitted to the Command> in the console window. Script files can be be submitted typ- ing the commands Aladin filescript or Aladin < filescript Additionally, the window interface can be removed by launching Aladin with the -script parameter.

The list of script commands is detailed below: see also the Help menu, this help is also reported in the Appendix B.

With "*x*" a number plane identifier (number 1 is the bottom of the Aladin plane stack) or a label plane identifier (allowing use of the wildcard "*"), with "*v*" a view (in multiview mode, they are labelled *A1, A2... B1...*), the in-line commands are:

PLANE:

– get servers [target] [radiusUnit]: execute a command to call image and data servers (ex: 'get aladin(DSS2),VizieR(GSC) M99')

– load *file*: create a new plane with the file (image or data)

– select *x*: select the plane *x*
– rename [*x*] *newname*: name or rename a plane
– hide [*x*] [*x1*] [*x2...*]: hide planes
– show [*x1*] [*x2...*]: show planes
– mv *x1 x2*: move plane *x1* below plane *x2* (or into if *x2* is

a folder)

- `rm [x]|all`: delete planes
- `export nn file [votable]`: save images or tabu- lar data from plane number *nn* (FITS formats for image, txt or VOTable otherwise))

`VIEW:`

1 – `modeview [1|2|4|9|16]`: the multiview controller opens *1,2..16* views

2 – `createview [x [v]]`: only in multiview mode, al- lows to create a view at position *v*, from the data plane at position *x* in Aladin stack

3 – `select [v1] [v2...]`: select views in multimode- view, views *v1, v2...* are labelled *A1, A2... B1...*

4 – `zoom fct`: change zoom factor on the current view (`1/2x`, `4x`, ...)

5 – `attach|detach [v1] [v2...]`: view attachment to the current target (in multiview mode), the attach view al- ways try to display the current target (reticle position)

6 – `lock|unlock [v1] [v2...]`: lock a view to keep it in the same place in multiview mode, even scrolling the view panels

7 – `mv|copy v1 v2`: move or copy views (multiview mode)

8 – rm *[v1] [v2...]* | *ROI* : delete views

9 – save *file*: save the current view, image and overlay (BMP format)

10 – coord|object: show a specified position

IMAGE:

1 – flipflop [V|H]: vertical (resp. horizontal) image in- version

2 – reverse [on/off]: (un)reverse the current view

3 – cm {gray|BB|A|stern}: select the colour map

4 – RGB [*x1*|*v1 x2*|*v2 x3*|*v3*]: create a RGB image from planes *x1*, *x2*, *x3* or view *v1*, *v2*, *v3*

5 – blink [*x1*|*v1*] [*x2*|*v2...*]: create a blink sequence of images –resamp[*x1x2*][8|Full][Cl|bil]: create a resampled image

– contour [*nn* [[no]smooth] [zoom]]: draw *nn*

isocontours on the selected plane (default: 4 levels) (zoom: draw the contours only on the zoom area).

CATALOGUE:

1 – flipflop [V|H]: vertical (resp. horizontal) pixel im- age inversion

2 – filter [*name*] {*filter-content*}: create a fil- ter, the *filter-content* must follow the filter instruction rules (see section 8.3.7) (ex: $[PHOT*]<16 {draw})

3 – filter fied filter

4 – addcol

[*name*] [on/off]: switch on/off the speci-

([*x*],[*name*],[UCD],[Unit],expr): column generator for a tabular data plane

1 – xmatch [*x1*] [*x2*] [dist]: positional cross match tool between tabular data planes x1 and x2

2 – creatplane [*name*]: create a new catalogue plane with the current selected sources

– `createROI [npix|radius"]`: create zoomed views around the selected sources

`GRAPHIC TOOL:`
– `draw` *`fct(param)`*: add graphical overlays on views with three functions: string, tag, line

– `grid [on/off]`: (un)activate the coordinate grid
– `reticle [on/off]`: (un)activate the reticle
– `scale [on/off]`: (un)activate the scale line and other

view overlays information

`FOLDER:`

1 – `md [`*`name`*`]`: create a new folder

2 – `mv` *`x1 [x2...] name`* : move data planes `x1` and `x2` within folder `name`

3 – `rm [`*`name`*`]`: delete a folder

4 – `show [`*`name`*`]`: show and display the views of all the data planes disposed in a folder

5 – `hide [`*`name`*`]`: hide the views accociated to the content of a folder

`MISCELLANEOUS:`

6 – `backup` *`file`*: generate a backup (proprietary .AJ for- mat) of the Aladin plane stack

7 – `print`: print the current views

8 – `trace`: turn on and off the debug verbose mode

9 – `reset`: reset the Aladin plane stack and views

10 – `info` *`msg`*: print a message in the status window

11 – `hist`: display the script command history in the Aladin console

12 – `status`: display stack plane status

13 – `mem`: display the current memory size

14 – `help [cmd|off]`: display this help

- `demo`: demonstration mode
- `sync`: wait until all planes are ready
- `pause [`*`nn`*`]`: wait *nn* seconds (default: one)
- `timeout` *`nn`*`|off`: set *nn* minutes as timeout (default: 15)
- `quit`: stop Aladin.

Script example

Suppose that the file foo contains the following lines:

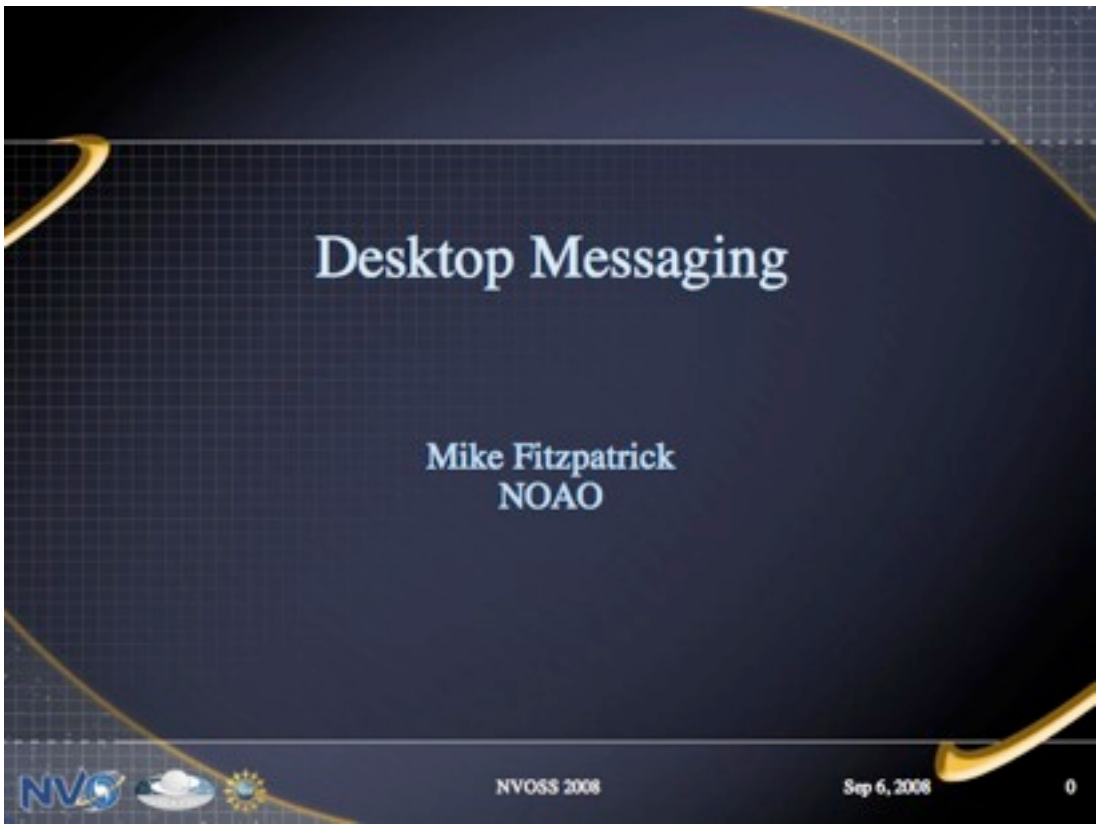
```
grid

get aladin(dss2),vizier(usno2) m99
sync
zoom 2x
cm BB
save m99.bmp
quit

```

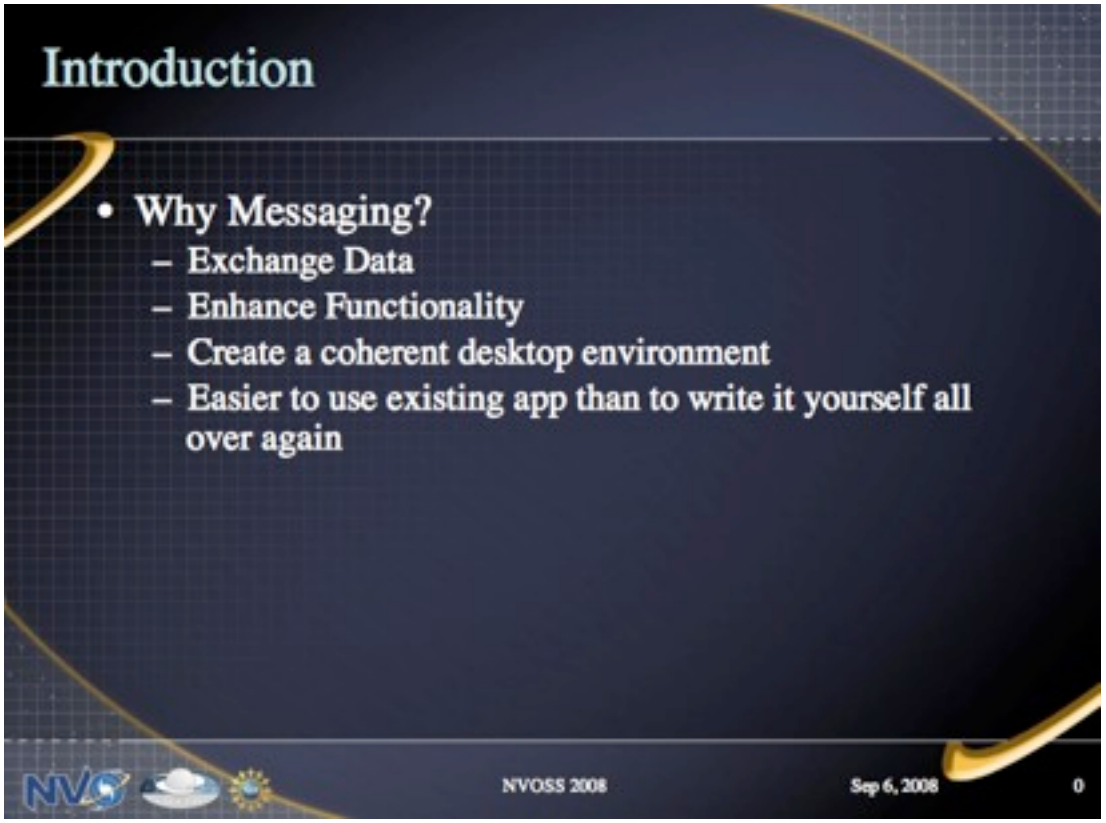
By redirecting the Aladin standard input as below:

```
java -jar Aladin.jar -script < foo
```

Aladin produces transparently an image file called m99.bmp with a DSS2 image and the USNO2 catalogue overlay.

Présentation de SAMP par Mike Fitzpatrick





PLASTIC

PLASTIC (Platform for Astronomy Tool InterConnection) was a first cut at tool interoperability.

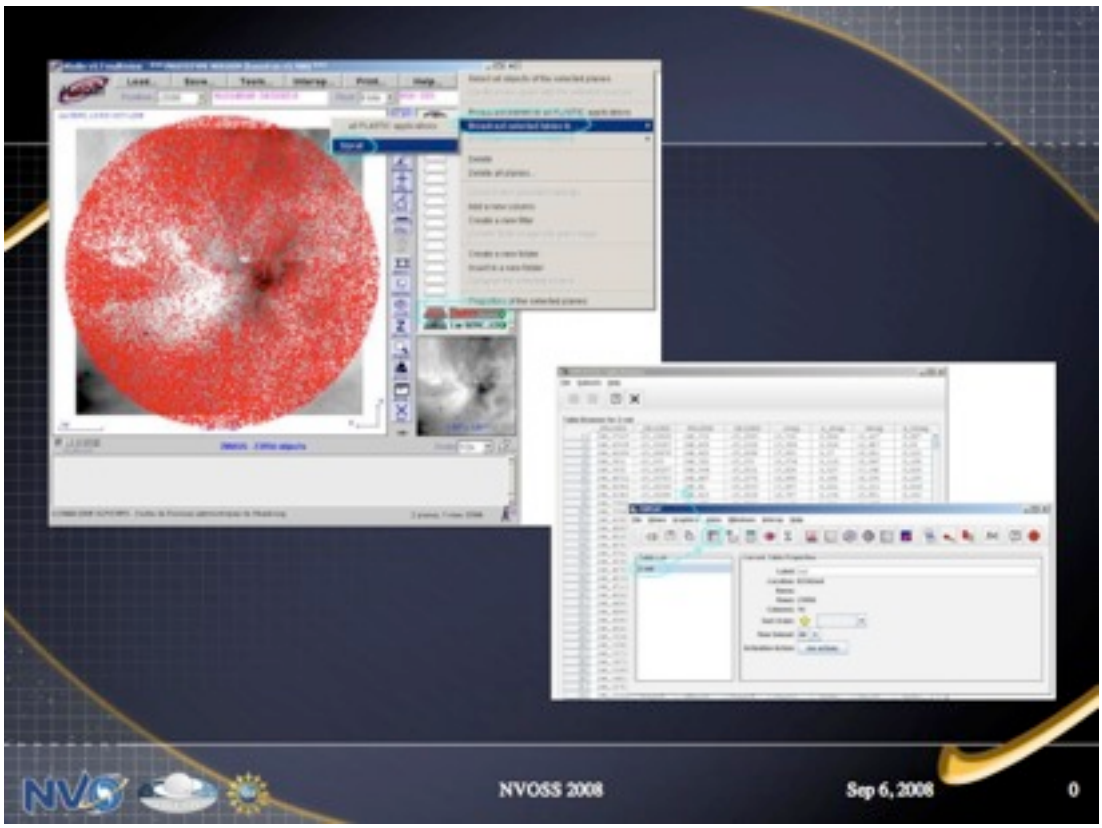
- Language Neutral, implementations in many languages
- XML-RPC as the underlying protocol (Java RMI also supported)
- Extensible simply by creating new messages
- Intentionally simple to foster quick adoption by others

PLASTIC

- Key concepts are the Hub and Messages
- Messages consist of strings agreed to by the developers
  - Same form as an ivorn
  - No clear method for how these are created

    ivo://votech.org/votable/loadFromURL
    ivo://votech.org/votable/showObjects
    ivo://votech.org/fits/image/loadFromURL

- Hub either standalone or as part of an app



SAMP

- Simple Applications Messaging Protocol
  - Successor to PLASTIC
  - Currently an IVOA Working Draft specification
  - Implementations in progress
- Design Goals
  - Want to keep it simple to get quick adoption
  - Don't *require* backward compatibility
  - Want to formalize message types
  - Want to extend it for more complex messaging in later versions

SAMP Message Models

- Publish/Subscribe messaging
  - Clients can subscribe only to messages they support
- Broadcast & Point-to-Point messaging
  - Clients can send a message to *anyone*, or a specific client it knows is connected
- Event-based messaging
  - Some message types only announce an event vs. a request for some action
- Synchronous/Asynchronous

SAMP Architecture

- Hub-based
- All clients required to register with the Hub
- XML-RPC underlying protocol
- Delivery Patterns
  - Notification
  - Asynch Call/Response
  - Synch Call/Response

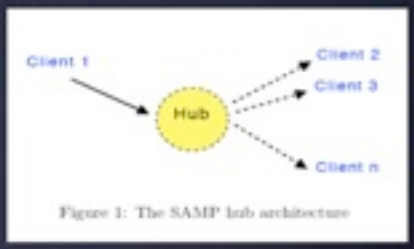


Figure 1: The SAMP hub architecture

SAMP Architecture

- SAMP Profiles
  - Allows configuration of SAMP (e.g. alternates to XML-RPC, language-specific interfaces)
- Hub Registration
  - Clients and Hub each have unique ID assigned
  - Clients declare metadata about themselves (e.g. name of app, a description, etc)
  - Clients declare which messages they're interested in

Abstract API

```
          reg-info = register  ()
                   unregister  ()
               declareMetadata  (metadata)
        map meta = getMetadata  (client-id)
           declareSubscriptions  (map subscriptions)
 subscriptions = getSubscriptions  (client-id)
client-ids = getRegisteredClients  ()
client-subs = getSubscribedClients  (mtype)

                       notify  (recipient-id, message)
                    notifyAll  (message)
         string msg-id = call  (recipient-id, msg-tag, message)
      string msg-id = callAll  (msg-tag, message)
   map response = callAndWait  (recipient-id, message, timeout)
                        reply  (msg-id, response)
```

Admin Messages

- Use *samp* namespace (e.g. *samp.hub.stopping*) ?

```
hub
      event
          stopping                          hub is stopping
  app
      event
          register      id                  app has registered
          unregister    id                  app has unregistered
          starting      id                  app starts processing
          stopping      id                  app stops processing
          mtype         mtypes              app declares new mtype
                        id
          metadata      meta                app declares new metadata
                        id
      echo              str                 app echo string
      isAlive           id                  app is alive
```

Application Messages

```
set
      param         param           set param to
value
                    value
get
      param         param     value   get param
value
```

General set/get messages

```
file
     event
          load      filename        the 'filename' was
loaded
          save      filename        the 'filename' was
saved
     load           filename        load this file 'filename'
     save           filename        save to 'filename'
```

What about *rename* and/or *delete* ?

```
url
     event
          load      url             'url' was loaded
          save      url             'url' saved to
'filename'
                    filename
     load           url             load url at 'url'
     save           url             save 'url' to
'filename'
                    filename
```

Others?

Application Messages

```
image
        event
            load        imname          the 'imname' was loaded
            save        imname          the 'imname' was saved
        load            imname          load image 'imname'
        save            imname          save image to 'imname'

        display         imname          display image in 'imname'
        panTo            x, y           pan display (arb coords)
            pixel        x, y           pan display to pixel coords
            sky          ra, dec        pan display to sky coords
        zoom            level           zoom to given level (+/-N level)
        highlight
            pixel        x, y           highlight point at pixel coords
            sky          ra, dec        highlight point at sky coords
```


Application Messages

```
table
        event
            load        tblname         the 'tblname' was loaded
            save        tblname         the 'tblname' was saved
        load            tblname         load table 'tblname'
        save            tblname         save table to 'tblname'

        highlight
            row         row             highlight specified row
            col         col             highlight specified column
            cell        row, col        highlight cell at position
        select
            row         row             select (subset) named row
            col         col             select (subset) named column
            rowList     rows            select (subset) named rows
            colList     cols            select (subset) named columns
```

Application Messages

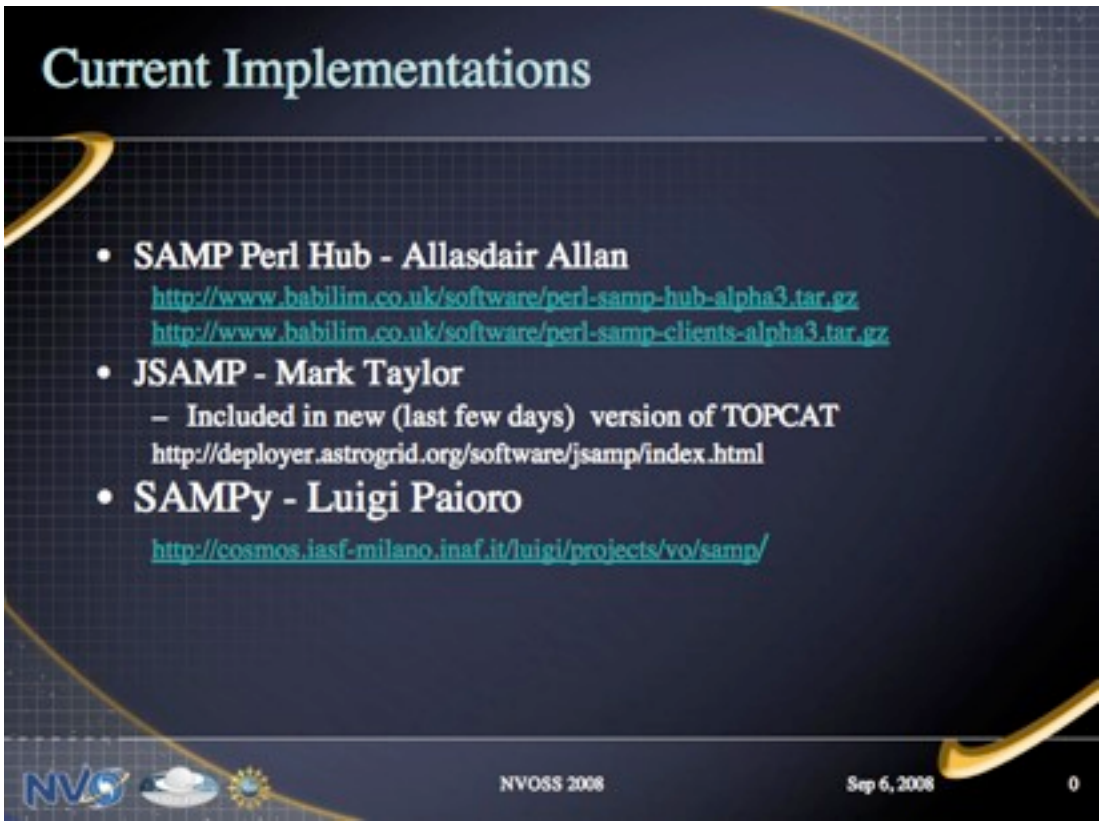
```
spectrum
        event
            load        name        the 'name' was loaded
            save        name        the 'name' was saved
        load            name        load table 'name' (no format)
            fits        name        load spectrum in FITS file
            votable     name        load spectrum in VOTable
                :        :                  :
        save            name        save spectrum to 'name'
            fits        name        save spectrum as FITS file
            votable     name        save spectrum as VOTable
                :        :                  :
```

- How do we handle spectral formats (e.g. echelle vs 1-D spectra)?
  - Suggest this be an optional parameter to the mtype

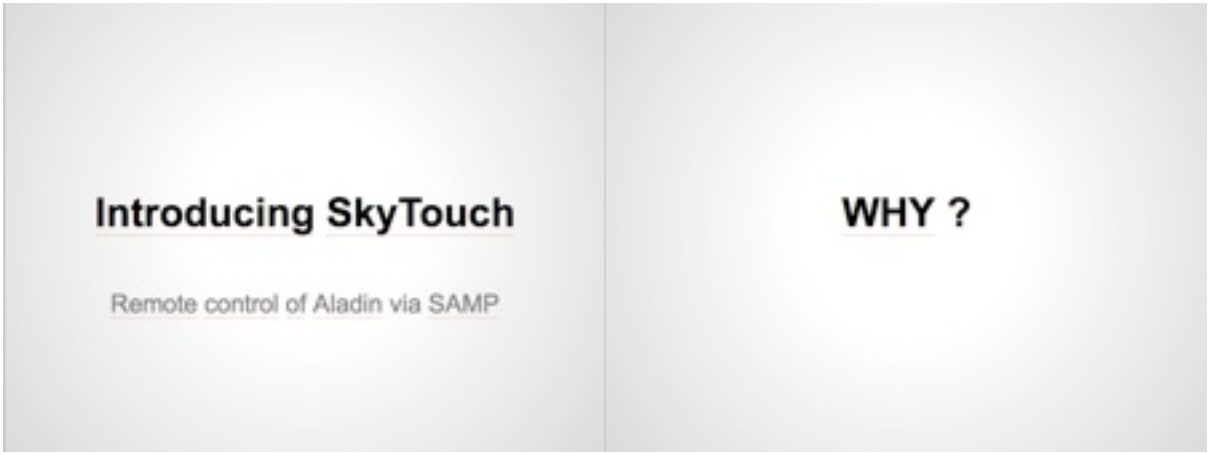
Application Messages

```
coord
    pointAt         x, y                point at the given coord
        sky         ra, dec             point at given sky position
```

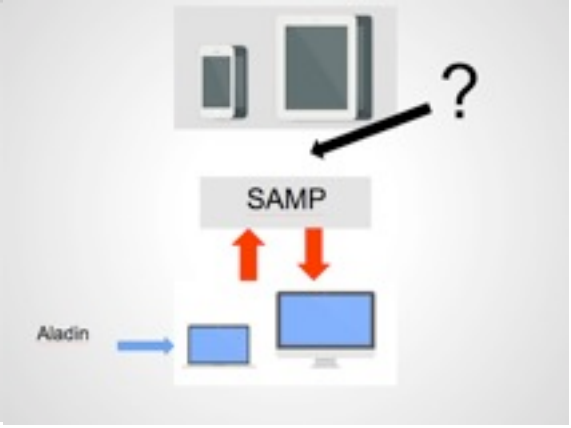
- What about coordinate transformations? STC shows us this is a can of worms, but we might want simple transforms like equatorial to galactic
- Does this confuse e.g. *image.panTo* Mtype?
- What others are needed??



Slides de ma présentation CoSADIE du 2 Septembre 2013



HOW ?



TECHNOLOGIES

- SAMP (Simple Application Message Protocol)
- SAMPJS
- HammerJS
- PhoneGap

SAMPJS

- "**Sampjs** is a small JavaScript library for using the SAMP Web Profile from within web pages"
- http://github.com/astrojs/sampjs/
- Developed by Mark Taylor

EXAMPLE

```
connector = new samp.Connector("SkyTouch", meta, subs);

var select = function(item){
    var id = $(item).attr("id");
    var message = new samp.Message("script.aladin.send",
                                   {"script": "select " + id.toString()});
    connector.connection.callAll(["script.aladin.send", message]);
};
```

MTYPE

Params

Initiate a connection

Message type

HammerJS

- HammerJS is a small JavaScript library for multi-touch gestures
- Multi-touch Trackpad in SkyTouch
- Maybe more multi-touch gestures in further releases

```
timeStamp     (Number)        time the event occured
target        (HTMLElement)   target element
touches       (Array)         touches (fingers, mouse) on the screen
pointerType   (String)        kind of pointer that was used. matches Hammer.POINTER_MOUSE|TOUCH
center        (Object)        center position of the touches. contains pageX and pageY
deltaTime     (Number)        the total time of the touches in the screen
deltaX        (Number)        the delta on x axis we have moved
deltaY        (Number)        the delta on y axis we have moved
velocityX     (Number)        the velocity on the x
velocityY     (Number)        the velocity on y
angle         (Number)        the angle we are moving
direction     (String)        the direction we are moving. matches Hammer.DIRECTION_UP|DOWN|LEFT|RIGHT
distance      (Number)        the distance we haved moved
scale         (Number)        scaling of the touches, needs 2 touches
rotation      (Number)        rotation of the touches, needs 2 touches *
eventType     (String)        matches Hammer.EVENT_START|MOVE|END
srcEvent      (Object)        the source event, like TouchStart or MouseDown *
startEvent    (Object)        contains the same properties as above,
                              but from the first touch. this is used to calculate
                              distances, deltaTime, scaling etc
```

PhoneGap

- Use Web standards to develop native applications
- Building applications for each device : iPhone Android Phone, Windows Phone

Difficulties

- Web-native apps are slow

- SAMP hub restrictions (mType, non local device)

Features

- Connection to the hub
- Display Sky
- Manage views (up to 9 views)
- Multi-touch trackpad
- Zoom
- Search bar
- Customizable buttons (Macro)
- Manage Colorations

DEMO

Thank you for your attention