Travaux d'Études et de Recherche
Rapport Intermédiaire

Julien Henry
Nicolas Lacourte-Barbadaux
Alexandre Masson
Léo Rousseau

14 Janvier 2013

Table des matières

1 Résumé du projet

R est un logiciel de statistiques open source. De notre côté, nous développons depuis plusieurs années un logiciel de visualisation de données en java doté d'une interface graphique 3D interactive.

Il s'agit ici d'étudier le portage de cette application vers R, en se focalisant notamment sur les capacités d'interaction des librairies 3D proposées par ce système.

2 Introduction du domaine

La classification La classification est une forme d'apprentissage automatique. Elle consiste, étant donné un ensemble d'éléments, à les répartir dans différentes catégories. Il existe deux courants majoritaires de classification.

La classification supervisée Dans la classification supervisée on cherche à déterminer la classe (l'étiquette) d'un objet étant-donné un ensemble d'apprentissage dont on connaît déjà la classification. Par exemple si on classe des animaux on doit pouvoir déterminer que le cobra est un reptile parce qu'il a une description proche des reptiles déjà présents dans l'ensemble initial.

La classification non-supervisée La classification non-supervisée, ou encore clustering, a un but différent. On cherche à classer un ensemble de données en plusieurs classes sans les connaître à l'avance. On calcule une répartition pour l'ensemble. C'est ce que fait par exemple l'algorithme des K-Moyennes.

3 Analyse de l'existant

3.1 Le logiciel R

Le logiciel R est à la fois un environnement interactif de calcul, en lignes de commandes, et un langage de programmation. Il existe sur différentes plateformes (Linux, MacOS et Windows). Il peut être utilisé pour des traitements de données ou des analyses statistiques. R est un logiciel libre, ouvert à la participation de développeurs extérieurs. En effet, l'environnement s'enrichit par l'intégration de packages provenant de la communauté ajoutant des modules et des fonctionnalités complémentaires au logiciel initial.

Le logiciel R

La communauté R est très active et de nombreux modules sont régulièrement proposés. R permet facilement de charger des données, de sortir des informations statistiques, de visualiser diverses courbes extraites des données, etc.

Par défaut, R se présente sous la forme d'une ligne de commande. Il est cependant possible de développer des interfaces utilisateur pour contrôler l'application ainsi que pour visualiser des résultats de traitement.

```
R version 2.11.1
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

>
```

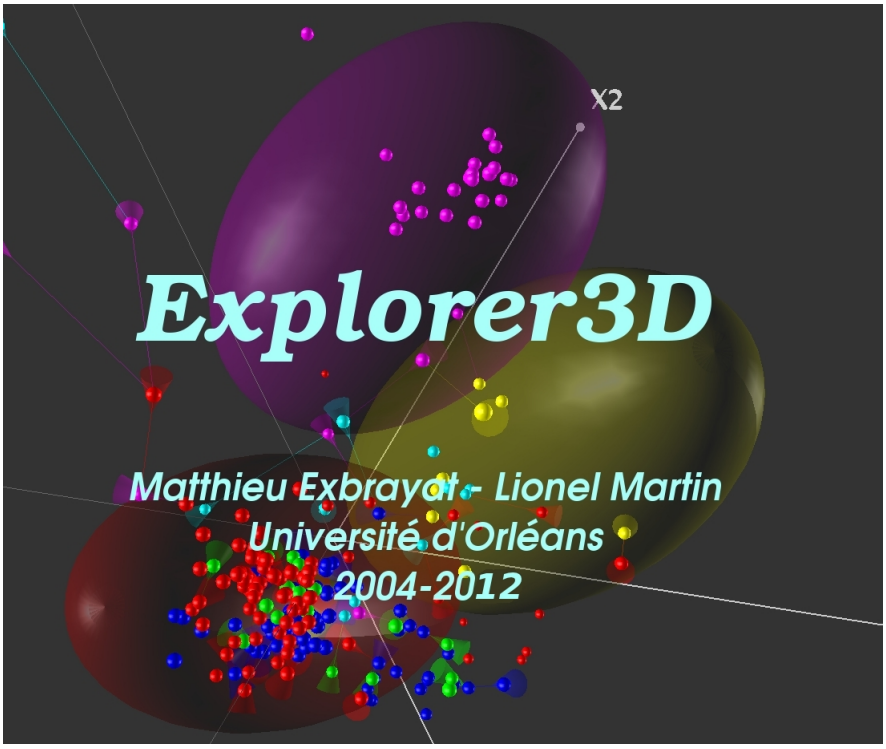
Aperçu de la console R

Le logiciel R nous permet de traiter des données tabulaires, sous forme de vecteurs, ou de data-frame (matrice contenant les instances de objets en ligne, avec un attribut par colonne). Il propose ensuite des opérations sur les vecteurs, tout en discriminant les objets par leurs attributs non numériques. Ce qui permet d'étiqueter les objets avant de faire des calculs et pouvoir ensuite différencier les résultats des calculs en fonction des étiquettes des objets.

On utilisera sans doute cette particularité pour colorer les objets dans la fenêtre 3D, en fonction de leurs propriétés.

3.2 Le logiciel Explorer3D

De leur coté, Matthieu Exbrayat et Lionel Martin, maîtres de conférences au Laboratoire d'informatique Fondamentale d'Orléans, développent depuis plusieurs années un logiciel de visualisation de données doté d'une interface graphique 3D interactive. Ce logiciel est écrit en Java.
http ://www.univ-orleans.fr/lifo/software/Explorer3D/



Explorer3D

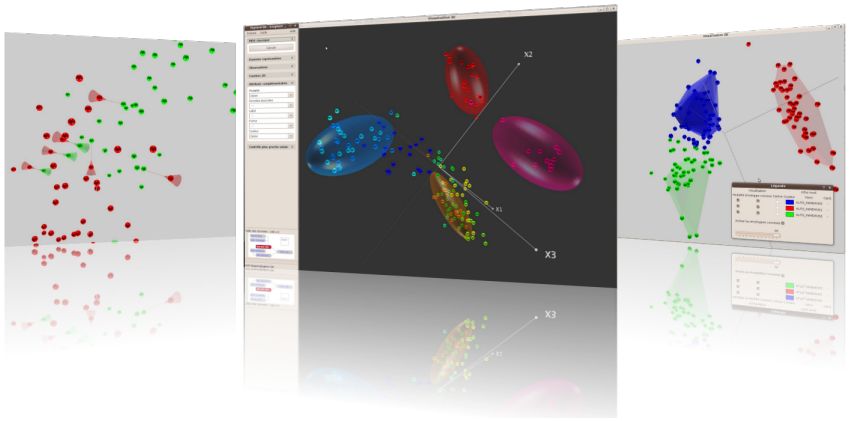
Ils ont acquis une bonne maîtrise de la structuration d'un tel outil afin de le rendre évolutif. Ils ont également mis en place un certain nombre de fonctionnalités interactives.

La plateforme R est devenu un standard du domaine de la classification. Mais le logiciel Explorer3D, propose une interface inédite pour la visualisation des résultats de classification. C'est pourquoi ses concepteurs souhaiteraient porter leur travail sous R, car ils sera ainsi plus facile d'accès et s'inscrira dans le standard du domaine. Cette mise aux normes du standard devrait permettre une diffusion plus large du projet Explorer3D, puisque cela le rendra utilisable par tous les utilisateurs de R.

Nous étudions maintenant les possibilités d'interactions entre ce logiciel et la plateforme R, afin d'intégrer rapidement divers outils existant sous R, mais également de diffuser notre logiciel vers cette communauté.
Au moins deux approches sont possibles :

- intégrer des invocations de R depuis java.
- migrer notre logiciel vers R.

Concernant le premier point, l'invocation de R depuis java est relativement simple. Elle ne sera pas abordée dans ce TER. Concernant le second point, R propose la création de fenêtres de contrôle et de fenêtres graphiques 2D et 3D (voir par exemple http ://rgl.neoscientists.org/). Toutefois, nous ne connaissons pas les possibilités d'interaction réelles de l'interface 3D.



Les possibilités d'Explorer3D

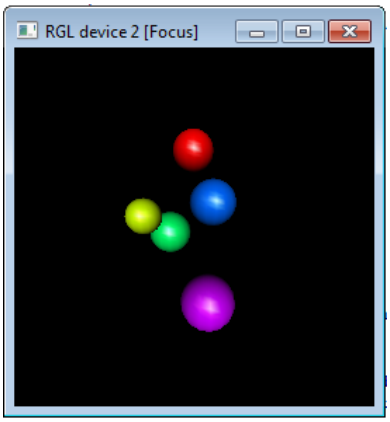
3.3 Packages R utilisables

Après quelques recherches sur internet, il apparaît qu'il existe des packages R, bibliothèques libres mise au point par la communauté, qui pourraient répondre à certains besoins du portage d'Explorer3D sous R. Nous en avons retenu deux pour le moment, RGL et Rgtk, qui semblent répondre respectivement aux attentes en termes de : fenêtre 3D et interactions avec la scène 3D pour le premier, et interface utilisateur pour le second. Nous allons présenter plus en détails ces deux packages.

3.3.1 RGL

RGL est un package pour le langage de programmation R. Il étend les possibilités de R avec l'ajout d'outils de visualisation 3D en temps réel.

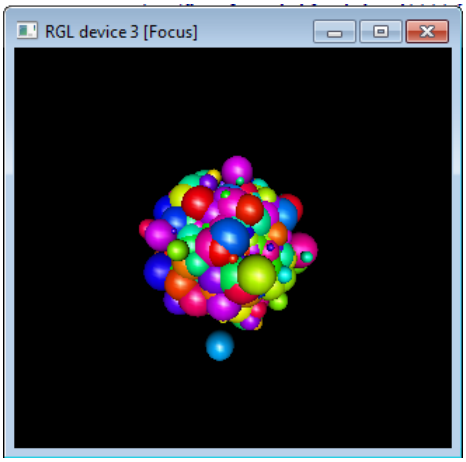
Les univers 3D ont besoin d'être projetés sur des images 2D affichées à l'écran, c'est pourquoi il est nécessaire d'avoir un outils qui effectue cette étape afin d'afficher des objets tri-dimensionnel à l'écran. Cet outils doit simuler la lumière, la structure des objets et leurs textures afin de donner l'illusion de 3D sur les images 2D projetées.



Des sphères dans RGL

Le coeur de RGL est codé en c++ et utilise OpenGL. RGL est ainsi une interface entre R et OpenGL. OpenGL est un standard des applications graphiques utilisé dans un grand nombre d'applications et langages. Il résout notamment le problème d'affichage en 2D d'univers 3D cité plus tôt.

Plusieurs fonctionnalités sont accessibles simultanément via RGL, telles que l'intégration drag/drop dans la fenêtre de visualisation et la réception de nouvelles instructions depuis la console R. Ainsi les objets peuvent être examinés en trois-dimension grâce au zoom et la scène peut pivoter sur elle-même.



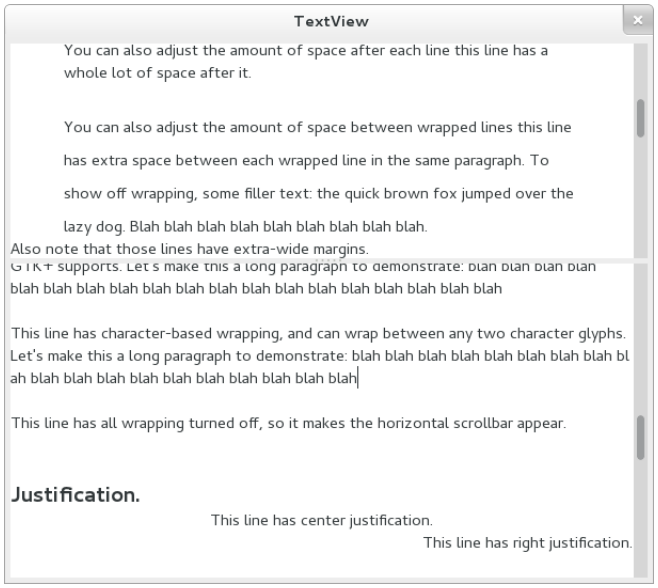
Plus de sphères dans RGL

Le but de RGL est donc de surcoucher OpenGL pour le langage R. Il permet ainsi via des instructions de générer des scènes 3D et d'interagir avec celles-ci via de nombreux outils.

3.3.2 RGTK

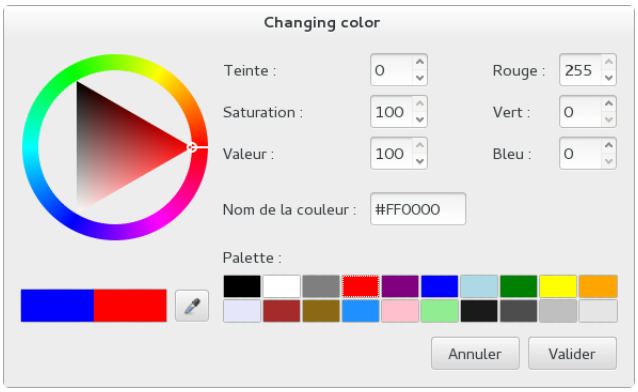
Le package RGtk2 s'ajoute en tant que librairie externe pour inclure dans R la possibilité de créer des interfaces graphiques.

Comme dans Explorer3d, on a besoin de gérer les données grâce à l'interface graphique. Cette extension de R est essentielle dans le développement de l'application.



fenêtre divisée en plusieurs parties

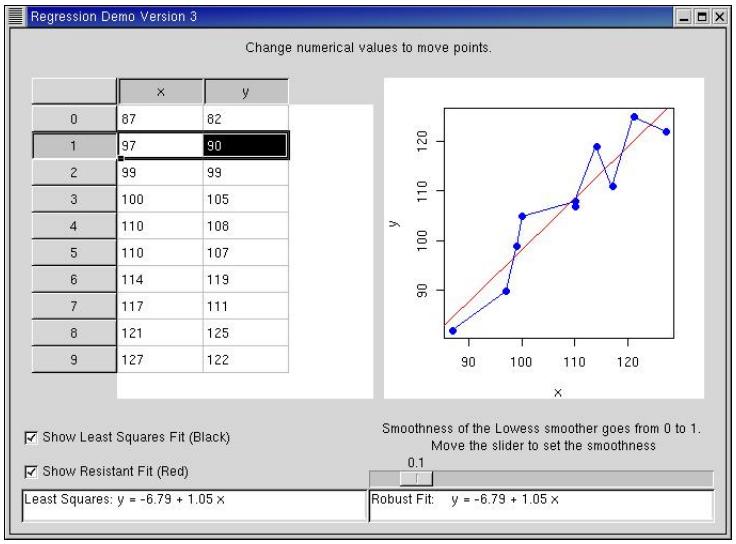
RGtk est un package dérivé de GTK2. Il est écrit en c++. GTK est un standard pour la création d'interface utilisateurs en c++ et bien d'autres langages. Grâce à RGtk2 nous pouvons résoudre le problème de création d'interface graphique. Comme dans la capture d'écran précédente nous pouvons créer une fenêtre avec plusieurs compartiments à l'intérieur de celle ci. Nous pouvons aussi avoir une fenêtre pour choisir la couleur d'un objet affiché dans une vue RG.



Sélectionneur de couleur RGtk

Le but de RGtk2 est de surcoucher GTK. Avec ce package on pourra interagir avec le logiciel.

Ce package nous offre la possibilité de créer des interfaces graphiques pour l'utilisateur, ce qui lui permettra de faire les demandes de calculs, sans passer par la console.



Un autre exemple d'interface graphique avec RGtk

4 Besoins non fonctionnels

- Montée en charge
- Documentation des technologies à utiliser
- MVC

4.1 Montée en charge

Comme expliqué plus haut, nous allons différer de la méthode utilisée dans Explorer3D pour toute la partie affichage. Pour savoir si les solutions choisies sont acceptables, il sera nécessaire de faire des tests de montée en charge.

Pour cela nous allons tester la réactivité du système aux différents événements, que ce soit des demandes de calculs,ou des interactions avec la scène 3D (rotation, zoom, déplacement, etc...).

Pour réaliser ces tests , nous procéderons à la main aux différentes manipulations, et nous nous baserons sur notre jugement en terme de vitesse de réponse, car il n'est pas facile de chronométrer la réactivité de la scène 3D, nous allons bien entendu refaire ces test avec un nombre de plus en plus grand d'objet à afficher, pour pouvoir déterminer une limite en terme de quantité d'objets représentables.

4.2 Documentation des technologies

Dans le cadre cette étude nous allons être amenés à utiliser certaines technologies qui ne seront pas forcément facile à mettre en oeuvre. Cette étude devant se placer dans la continuité d'un projet plus large, nous allons aussi devoir produire et introduire dans notre rendu, une documentation sur les technologies utilisées.

Mais plus qu'une documentation technique, nous souhaitons produire plusieurs petits tutoriaux, qui permettrons à nos successeurs de gagner du temps, et de prendre en main plus facilement le résultat de notre travail, ainsi que les technologies utilisées pour mener à bien ce projet.

4.3 Modèle Vue Controleur

Comme expliqué dans l'analyse de l'existant, Explorer3D a été développé pour être extensible et modulable. Ses concepteurs ont en effet veillé à ce que son architecture permette l'ajout de nouveaux modules, sans empiéter sur les modules existants.

Il serait donc souhaitable de garder la même optique lors de ce TER, et si possible mettre en place un modèle propre, de type Modèle Vue Contrôleur, pour permettre à notre travail d'être évolutif. Nous allons donc essayer de séparer les parties calculatoires des parties d'affichages dans la conception et le développement.

5 Besoins fonctionnels

Les principaux besoins fonctionnels sont les fonctionnalités d'Explorer3D qu'il est nécessaire de porter sous R, du moins autant qu'il sera possible d'en porter. Nous allons lister les différentes fonctionnalités souhaitées et nous les commenterons.

Nous souhaitons pouvoir fournir a l'utilisateur une interface graphique lui permettant d'agir sur les données, offrant diverses possibilités, telles que le chargement de données, ou la demande de calcul. Il doit aussi avoir la possibilité d'ouvrir une nouvelle fenêtre 3D interactive.

Nous allons donc déterminer si il est possible de créer ces différentes fenêtres, aussi nous chercherons à savoir comment pourrions nous agir depuis cette interface utilisateur sur la ou les fenêtres 3D.

Nous allons maintenant préciser ce que nous attendons des fenêtres 3D.

Nous l'avons évoqué dans les besoins non fonctionnels, nous voulons avoir une fenêtre qui nous permet d'afficher une très grand nombre d'objets, nous ne donnerons pas d'ordre de grandeur pour le moment, car nous en saurons plus après les premiers tests.

Nous souhaitons aussi avoir la possibilité de modifier la scène 3D, plus précisément nous souhaitons connaître les possibilités de zoom et de déplacement de la scène dans la fenêtre 3D. Nous nous intéressons aussi à la rotation de la scène, nous aimerions pouvoir faire tourner la scène et ainsi avoir un autre point de vue de la répartition des objets ;

Comme dans Explorer3D, nous voulons offrir la possibilité de sélectionner certain objets, par un clic sur l'objet dans la fenêtre 3D. Il sera donc nécessaire de pouvoir connaître l'état d'un objet, s'il est sélectionné ou non. Plus qu'un coté algorithmique, nous cherchons ici à savoir comment pourrions nous si possible implémenter ça avec RGL.

Nous souhaiterions aussi avoir la possibilité de changer les propriétés des objets, par exemple la couleur, afin de pouvoir mettre en évidence les objets sélectionné, et ceux de manière la plus fluide et rapide possible.

Nous souhaitons aussi offrir la possibilité de repositionner les objets dans l'espace, et nous souhaitons pouvoir afficher dynamiquement ce repositionnement, nous chercherons ici a afficher une animation, et pas seulement rafraîchir la scène avec la nouvelle position de l'objet déplacé.

Nous aimerons aussi pouvoir rajouter de façon dynamique des objets dans la scène tels que des ellipses ou ces enveloppes connexes, encore une fois , sans avoir a rouvrir une nouvelle fenêtre, mais juste en mettant a jour la fenêtre courante.

Nous souhaiterions avoir la possibilité de pouvoir faire cohabiter plusieurs fenêtres 3D, par exemple pour comparer plusieurs représentations des objets.

Explorer3D propose aussi un système de loupe , qui en plus d'un zoom optique sur une partie de la scène où se trouve le pointeur de la souris, sélectionne tous les objets présents sous la loupe, ce qui permet par exemple de mettre en évidence , la présence d'un groupe d'objets dans une vue (une fenêtre) mais ce même "groupe" serait dispersé dans une autre vue.

- AJOUT : possibilité de se mettre en écoute sur un port pour réception de commandes de type jeu de données + affichage.

6 Description des prototypes

6.1 Utiliser RGL

Prérequis Vous devez avoir une version de R récente déjà présente sur votre ordinateur ; le mieux est de récupérer les sources les plus à jour et de les compiler directement pour votre système.

Le package RGL se trouve dans la liste des package disponnibles pour R à cet adresse : http ://cran.r-project.org/ section "Packages", "sorted by Names", "rgl". Le fichier à récupérer est le "package source" en tar.tgz. Une fois que ce fichier est téléchargé, lancez une invite de commande R et saisissez :

```
> install.packages("$HOME/Downloads/rgl_0.93.928.tar.gz")
```

Le chemin passé en paramètre est un exemple, donnez le chemin vers le fichier que vous avez téléchargé. Celà devrait prendre un peu de temps.

Une fois l'installation terminée vous devez linker le package dans l'environnement courrant (et vous devrez le faire à chaque début de session R où vous aurez besoin de RGL) :

```
> library(rgl)
```

6.2 Première Session d'affichage 3D en R

Voici un code court qui va afficher N sphères de tailles, coordonnées et rayons aléatoires :

```
premier.test <- function(N)
{
  rgl.open()
  rgl.bg(color="black")
  rgl.spheres(x=rnorm(N), y=rnorm(N), z=rnorm(N),
              radius=runif(N), color=rainbow(N))
}

> premier.test(10)
```

Maintenant on va expliquer chaque lignes de ce script. La première commande, **rgl.open()** va permettre d'ouvrir une feneêtre openGL. Pour l'instant cette fenêtre est vide, on n'a pas encore demandé d'affichages. Ensuite la seconde ligne **rgl.bg(color="black")** va permetre de changer la couleur d'arrière plan.

Enfin, la dernière commande, la plus importante va demander au moteur OpenGL d'afficher N sphères. **rgl.spheres(x=rnorm(N), y=rnorm(N), z=rnorm(N), radius=runif(N), color=rainbow(N))**. Nous allons l'expliquer pour chaque paramètres.

- *x,y,z* : Trois vecteurs de taille identique représentant pour chaque sphère les coordonnées de son centre
- *radius* : Un vecteur de la même taille que x,y et z qui contient pour chaque sphère son rayon
- *color* : Un vecteur de la même taille que x,y et z contenant les couleurs pour chaque sphères (ou alors une seule couleur pour toute les sphères)

6.3 Primitives de dessin

6.3.1 Points et Lignes

Dessiner un point ou un ensemble de points. Le paramètre *color* est disponible pour l'ensemble des primitives

```
> rgl.open()
> rgl.bg(color="black")
> rgl.points(x, y, z, color="red" )
> rgl.lines(x, y, z)
```

- *x,y,z* : Trois vecteurs de taille identique représentant pour chaque point les coordonnées de son centre
- *color* : Un vecteur de la même taille que x,y et z contenant les couleurs pour chaque points (ou alors une seule couleur pour tous les points)

6.3.2 triangles

Dessiner un ensemble de triangles

```
> rgl.open()
> rgl.bg(color="black")
> rgl.triangles(x, y, z, normals=NULL, texcoord=NULL )
```

- *x,y,z* : Trois vecteurs de taille identique représentant pour chaque point les coordonnées de son centre
- *Normals* : Un vecteur contenant pour chaque sommet du reseau de triangle
- *texcoords* : Un vecteur contenant pour chaque sommet du reseau de triangle ses coordonnées de texture.

6.3.3 Sphères

Dessiner une sphère ou un ensemble de sphères. Le paramètre *color* est disponible pour l'ensemble des primitives

```
> rgl.spheres(x, y, z,r color="red" )
```

- *x,y,z* : Trois vecteurs de taille identique représentant pour chaque point les coordonnées de son centre
- *r* : Un vecteur de la même taille que x, y et z contenant le rayon pour chaque sphères
- *color* : Un vecteur de la même taille que x,y et z contenant les couleurs pour chaque sphères(ou alors une seule couleur pour toutes les sphères)

6.4 recapitulatif rapide des fonctions de base de RGL

gestion de l'outils

rgl.open()/open3d()	Ouvre une nouvelle fenêtre
rgl.close()	Ferme la fenêtre actuelle
rgl.cur()	Retourne le nombre de fenêtres ouvertes
rgl.set(idFenetre)	Selectionne la fenêtre corespondante comme active
rgl.quit()	Ferme toute les fenêtres ouvertes et décharge la console courante de RGL

gestion de la scène

rgl.clear()	Nettoie la scène en fonction du paramètre passé (ex "shapes" ou "lights")
rgl.pop(type="shapes")	Supprime le dernier objet ajouté à la scène

fonctions d'export

rgl.snapshot(nameFile)	Sauvegarde un screenshot de la scène courante au format PNG

gestion des objets

rgl.points(x,y,z,..)/points3d()	Dessine un point aux coordonnées x,y et z (voir plus haut)
rgl.lines(x,y,z,..)	Dessine une/des lignes sur l'axe (voir détails plus haut)
rgl.triangles(x,y,z,..)	Dessine un/des triangles (voir détails plus haut)
rgl.quads(x,y,z,..)	Dessine un/des carrés
rgl.spheres(x,y,z,r,..)	Dessine une/des sphères (voir détails plus haut)
rgl.texts(x,y,z,text,..)	Ajoute une texture à la scène
rgl.surface(x,y,z)	Ajoute a la surface ..

gestion de l'environnement

rgl.viewpoint(theta,phi,fov,zoom,interactive)	Définie le point de vue (theta, phi) dans des coordonnées polaires avec un angle de vue fov et un facteur zoom. Le marqueur logique interactive définie si ou non la navigation est autorisée
rgl.light(theta,phi,..)	Ajoute une source de lumière à la scène
rgl.bg(..)	Attribue le background de la scène
rgl.bbox(..)/bbox3d	Attribue un support visuel dynamique aux objets ainsi qu'une échelle visible de leurs coordonnées

6.5 Utiliser RGtk2

Prérequis Vous devez avoir une version de R récente déjà présente sur votre ordinateur ; le mieux est de récupérer les sources les plus à jour et de les compiler directement pour votre système.

Le package RGL se trouve dans la liste des package disponnibles pour R à cet adresse : http ://cran.r-project.org/ section "Packages", "sorted by Names", "rgtk". Le fichier à récupérer est le "package source" en tar.tgz. Une fois que ce fichier est téléchargé, lancez une invite de commande R et saisissez :

```
> install.packages("$HOME/Downloads/RGtk2_2.20.25.tar.gz")
```

Le chemin passé en paramètre est un exemple, donnez le chemin vers le fichier que vous avez téléchargé. Celà devrait prendre un peu de temps.

Une fois l'installation terminée vous devez linker le package dans l'environnement courrant (et vous devrez le faire à chaque début de session R où vous aurez besoin de RGL) :

```
> library(RGtk2)
```

6.6 Première fenêtre avec RGtk2

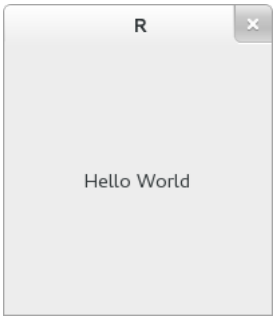
Voici un code court qui va nous permettre d'afficher une fenêtre avec le label "hello world".

```
>library(RGtk2)

createWindow <- function()
{
    window <- gtkWindow()

    label <- gtkLabel("Hello World")
    window$add(label)
}

>createWindow()
```



Résultat du code ci-dessus

Maintenant on va expliquer chacune des lignes de code. La première ligne **window <- gtkWindow()** nous permet de créer une fenêtre vide. Ensuite l'appel à la fonction **gtkLabel()** permet de créer un label qu'il faudra ensuite ajouter à la vue car pour le moment la vue est vide même si les éléments à ajouter sont déclarés. Enfin grâce à la dernière ligne, **window$add(label)** on ajoute le label à la vue.

6.7 Quelques fenêtres utiles

6.7.1 Une fenêtre en plein écran

Pour afficher une fenêtre en plein écran, il faut créer la vue et ensuite lui demander de passer en plein écran.

```
> win <- gtkWindow()
> gtkWindowFullscreen(win)
```

La fonction **gtkWindowFullscreen(win)** prend en paramètre un objet de type gtkWindow.

6.7.2 Des onglets dans la vue

Une fenêtre avec des onglets est une chose intéressante à avoir pour l'interface utilisateur.

```
> window.master <- gtkWindow("toplevel",show=FALSE)
>
> forms.notebook <- gtkNotebook()
> forms.notebook$setTabPos("top")
>
> form1.notebook <- gtkNotebook()
> form1.notebook$setTabPos("top")
>
> form1.boxp1.y3 <- gtkVBox(FALSE,3)
> form1.boxp2.ud <- gtkVBox(FALSE,2)
> forms.notebook$add(form1.notebook)
> window.master$add(forms.notebook)
>
> window.master$show()
```

- **gtkNotebook** est une conteneur de vue avec des onglets.

- **form1.boxp1.y3 <- gtkVBox(FALSE,3)** permet de créer une gtkVBox qui sera notre premier onglet.

- **forms.notebook$add(form1.notebook)** : on ajoute le formulaire form1 au formulaire principal forms qui sera ensuite ajouté à la vue via la commande **window.master$add(forms.notebook)**.

- **window.master$show()** affiche la fenêtre complète. L'appel de cette fonction est obligatoire car on avait choisi de ne pas afficher la fenêtre lors de sa création.

7 Planning, affectation des taches

Nous nous sommes réunis pour discuter de la façon dont nous allions structurer notre travail, mais nous avons aussi mis en place une méthodologie afin d'avancer tous sur les même bases. Nous avons aussi mis en place un planning et un répartition du travail.

7.1 Méthodologie

Concernant la voie à suivre pour notre travail de groupe ou individuel, nous nous sommes mis d'accord pour se concerter en début de chaque semaine, afin de faire le point sur l'avancement du travail jusqu'à la date de la réunion. Ce rendez-vous hebdomadaire a aussi pour but de fixer la répartition des taches pour la semaine qui suit, afin que chacun sache ce qu'il a à faire, et comment le faire au mieux.

Le but de ces rencontres et aussi de pouvoir faire semaine par semaine un compte-rendu de l'avancement au client (afin qu'il puisse suivre le développement du projet). Ce compte-rendu aura aussi pour objectif, du fait que notre client est aussi notre encadrant, de lui permettre de nous donner son avis sur notre organisation, étape par étape, ainsi que de voir avec lui au fur et à mesure que l'on avance, si nous répondons aux besoins, et de pouvoir lui poser des questions si nous rencontrons des difficultés.

7.2 Répartition des taches

Nous avons constaté que deux grandes parties se distinguent dans ce TER. Comme expliqué dans l'analyse des besoins, il apparaît qu'il est facile de traiter chacune de leur coté, la partie concernant l'interface utilisateur, et la partie de gestion de la fenêtre 3D.

Le groupe étant formé de quatre personnes, nous allons travailler par binôme, travaillant deux par deux sur chacune des parties du TER. Nous avons déjà travaillé avec OpenGL, et certain d'entre nous ont déjà travaillé avec RGL et Explorer3D.

7.3 Planning Général

8 Bibliographie