# IMN430 - Visualisation

## Chapitre 1
## Introduction

Michaël Bernier & Olivier Godin

Université de Sherbrooke

5 janvier 2017

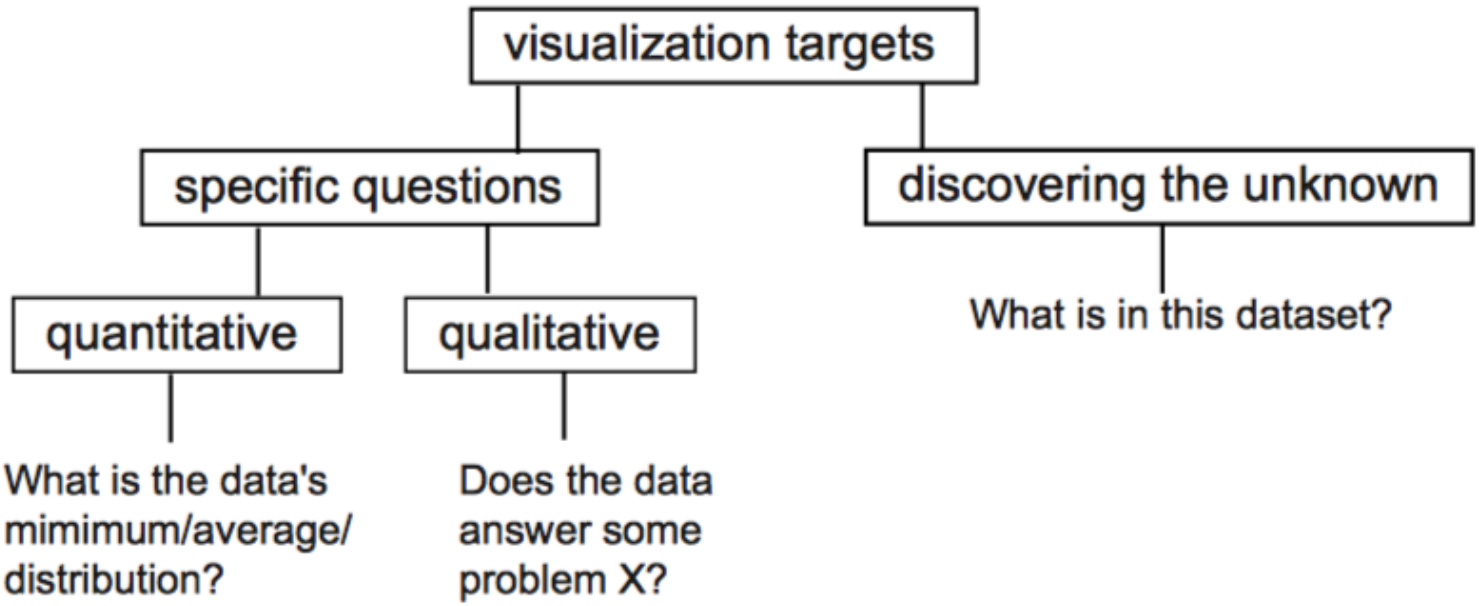
# Plan du chapitre

# Définition et objectifs

# Qu'est-ce que la visualisation ?

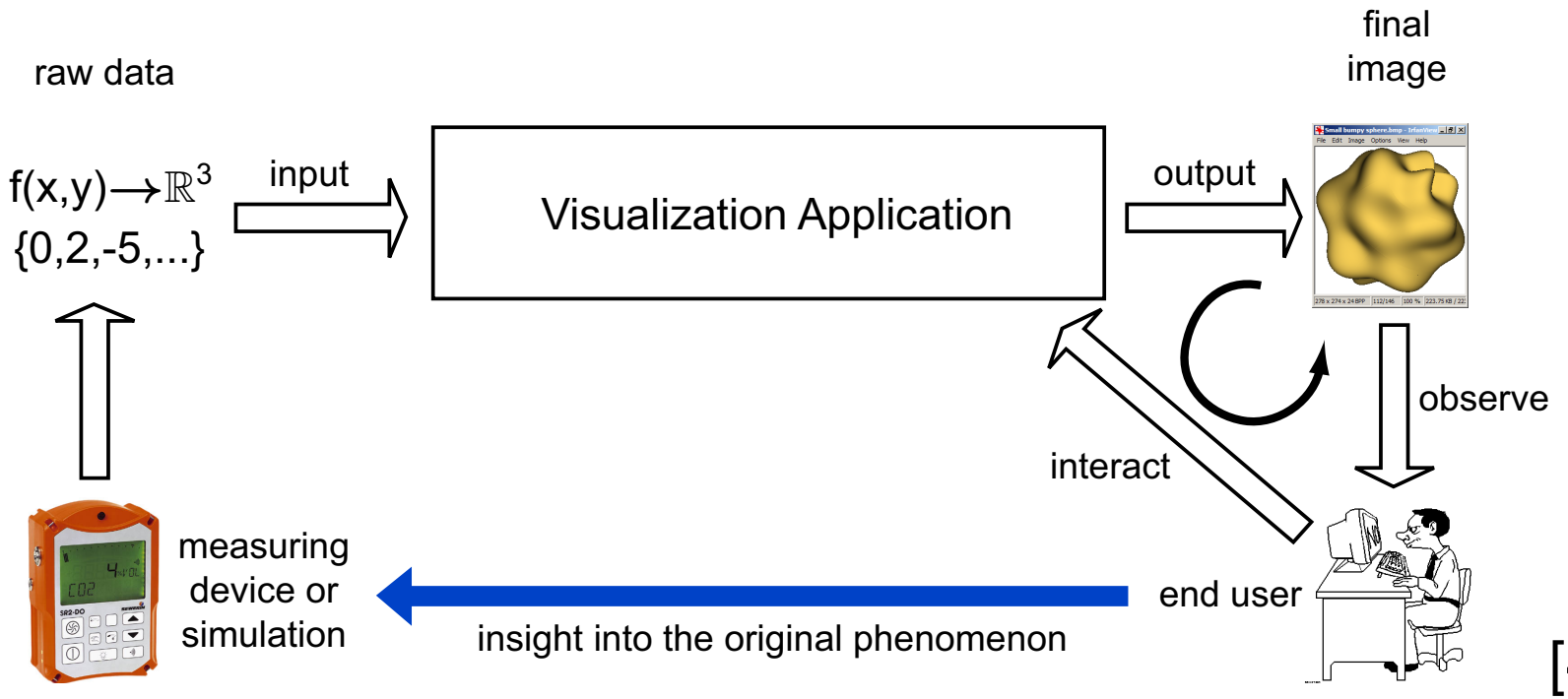
La visualisation est le processus d'**interprétation visuelle** ou de **représentation graphique** d'un ensemble de données.

Elle répond au besoin **répondre à une question spécifique** ou **explorer l'inconnu**.




[4]

# Qu'est-ce que la visualisation ?




[4]

# Qu'est-ce que la visualisation ?

La visualisation est une branche de l'informatique regroupant le **traitement**, l'**analyse** et la **représentation graphique** de données provenant de divers domaines : les finances, la médecine, les sciences sociales, le divertissement, etc.

Deux champs de compétence sont particulièrement sollicités en visualisation : l'**infographie** et les **statistiques**.

# Qu'est-ce que la visualisation ?

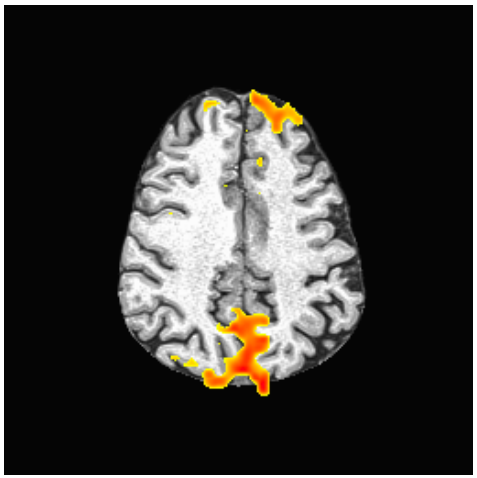
Au fil des années, les techniques de visualisation de données se sont démocratisées jusqu'à faire partie intégrante de nos vies.

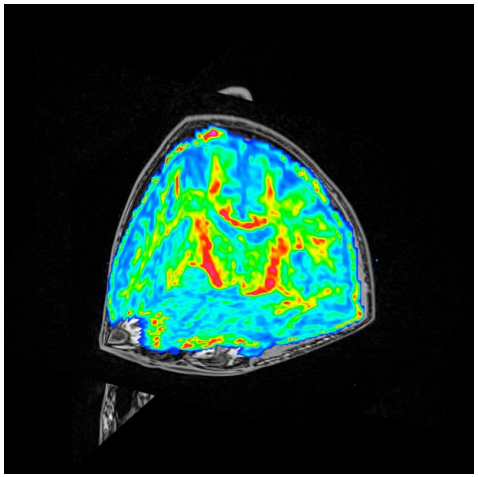
Ce faisant, nous ne remarquons plus nécessairement la présence et l'importance de ces résultats dans notre quotidien.

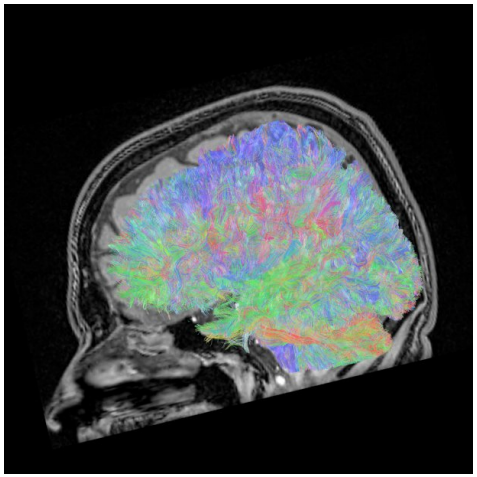
# Qu'est-ce que la visualisation ?

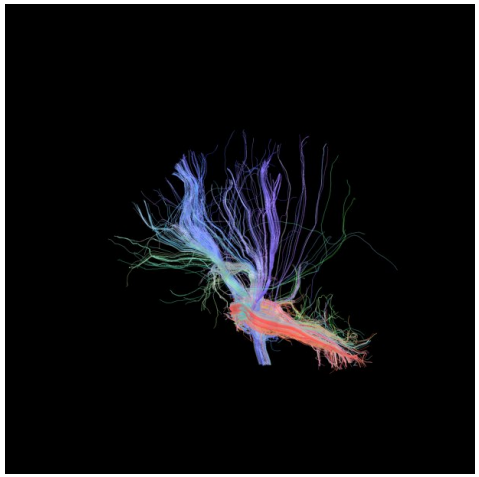
```
Date,Open,High,Low,Close,Volume
16-Mar-12,584.72,589.20,578.00,585.57,29481697
15-Mar-12,599.61,600.01,578.55,585.56,41416824
14-Mar-12,578.05,594.72,575.40,589.58,50672821
13-Mar-12,557.54,568.18,555.75,568.10,24673237
12-Mar-12,548.98,552.00,547.00,552.00,14545717
9-Mar-12,544.21,547.74,543.11,545.17,14960676
8-Mar-12,534.69,542.99,532.12,541.99,18444389
7-Mar-12,536.80,537.78,523.30,530.69,28518507
6-Mar-12,523.66,533.69,516.22,530.26,28937093
5-Mar-12,545.42,547.48,526.00,533.16,28897169
2-Mar-12,544.24,546.80,542.52,545.18,15418227
1-Mar-12,548.17,548.21,538.77,544.47,24401884
29-Feb-12,541.56,547.61,535.70,542.44,34000054
28-Feb-12,527.96,535.41,525.85,535.41,21442354
27-Feb-12,521.31,528.50,516.28,525.76,19556472
24-Feb-12,519.67,522.90,518.64,522.41,14831415
23-Feb-12,515.08,517.83,509.50,516.39,20286616
22-Feb-12,513.08,515.49,509.07,513.04,17260544
21-Feb-12,506.88,514.85,504.12,514.85,21628057
17-Feb-12,502.12,502.12,502.12,502.12,0
16-Feb-12,491.50,504.89,486.63,502.21,33733915
15-Feb-12,514.26,526.29,496.89,497.67,53789720
14-Feb-12,504.66,509.56,502.00,509.46,16497573
13-Feb-12,499.53,503.83,497.09,502.60,18471658
10-Feb-12,490.96,497.62,488.55,493.42,22546425
9-Feb-12,480.76,496.75,480.56,493.17,31579086
8-Feb-12,470.50,476.79,469.70,476.68,14567040
7-Feb-12,465.25,469.75,464.58,468.83,11293609
6-Feb-12,458.38,464.98,458.20,463.97,8917228
3-Feb-12,457.30,460.00,455.56,459.68,10245287
2-Feb-12,455.90,457.17,453.98,455.12,6671221
1-Feb-12,458.41,458.99,455.55,456.19,9644366
```

# Qu'est-ce que la visualisation ?





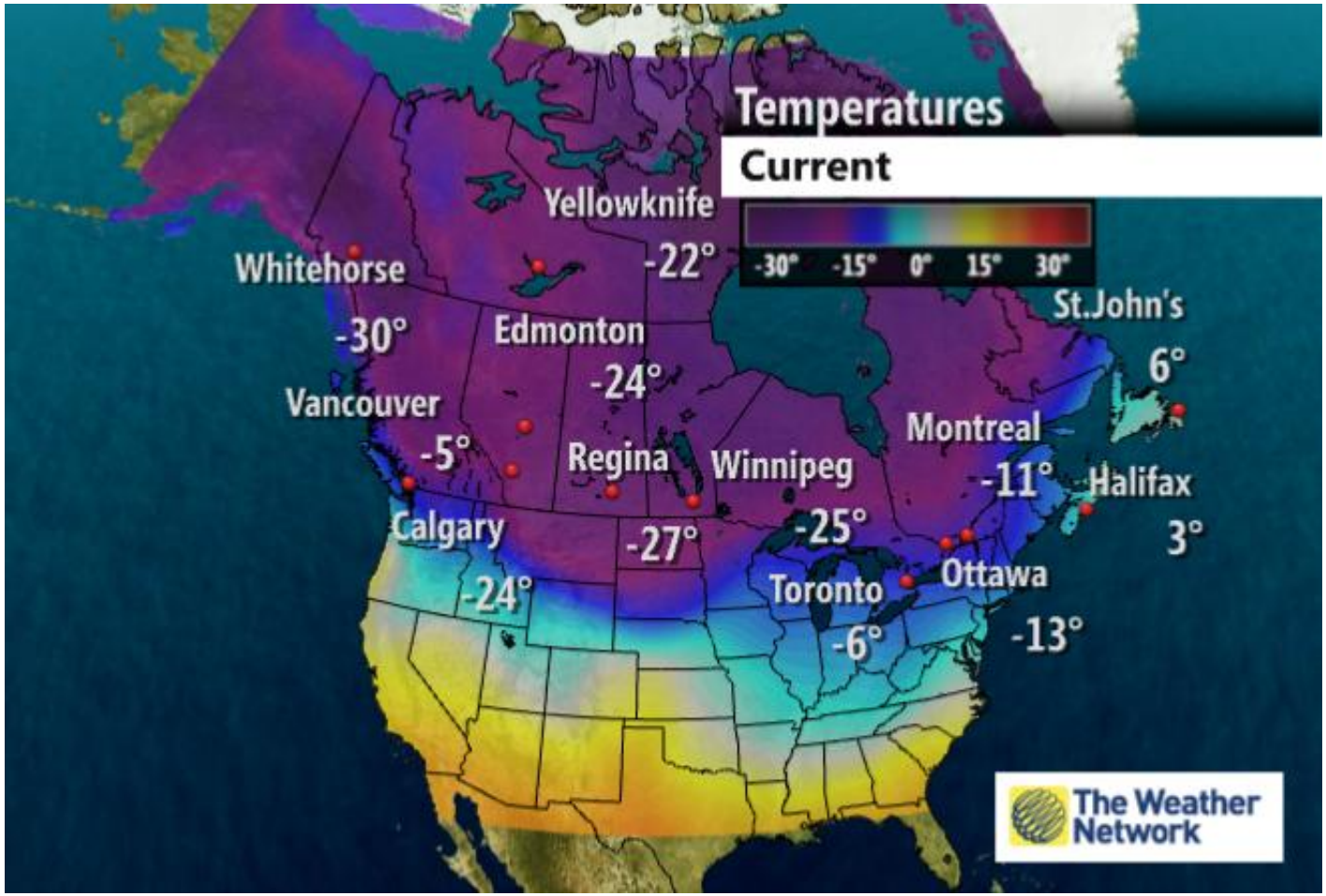




# Qu'est-ce que la visualisation ?


# Qu'est-ce que la visualisation ?

# Qu'est-ce que la visualisation ?



# Qu'est-ce que la visualisation ?

Il est important de savoir distinguer les domaines du traitement d'image, de l'infographie et de la visualisation.

Le **traitement d'image** est l'étude des images 2D pour en extraire de l'information ou pour en modifier les caractéristiques.

L'**infographie** permet de créer des images de toute pièce à l'aide d'un ordinateur, qu'il s'agisse d'images 2D dessinées par un artiste ou de complexes scènes 3D obtenues par des opérations de rendu.

La **visualisation** a quant à elle pour objectif de permettre l'exploration de données représentées sous une forme visuelle afin d'aider notre compréhension du phénomène illustré.

# Pipeline de visualisation

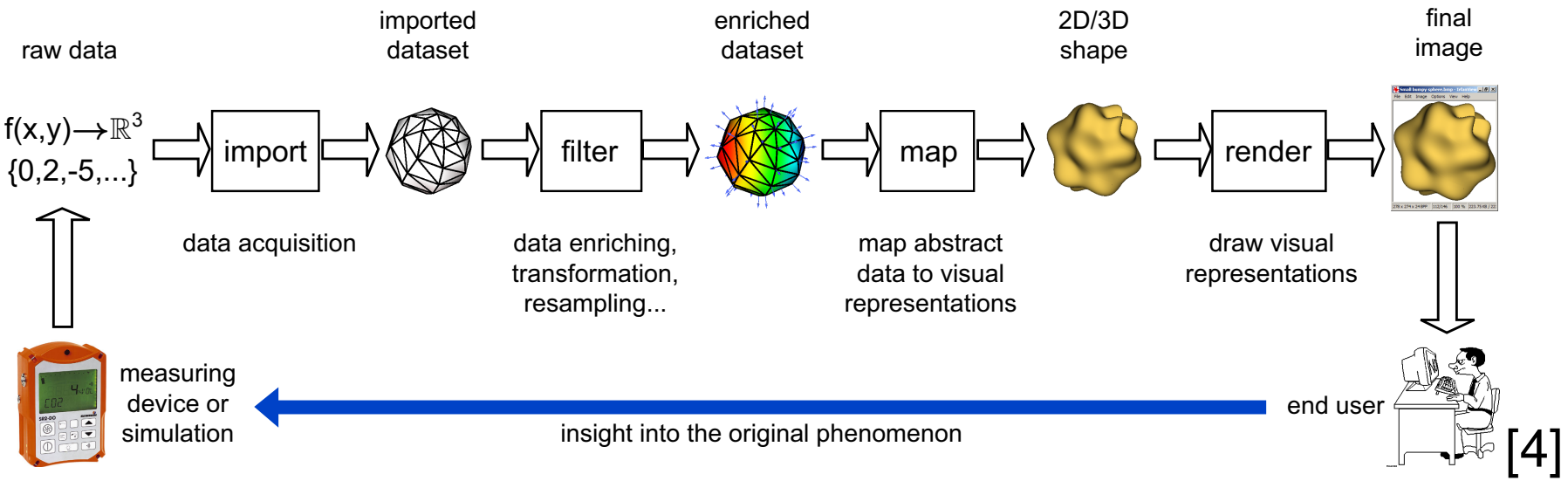
# Pipeline de visualisation

L'objectif de la visualisation est de représenter visuellement des données qui ne possèdent pas nécessairement d'**interprétation géométrique naturelle**.

On s'intéresse ici à la structure complète d'un logiciel de visualisation. Ce processus porte le nom de **pipeline de visualisation** et est composé de quatre grandes étapes :

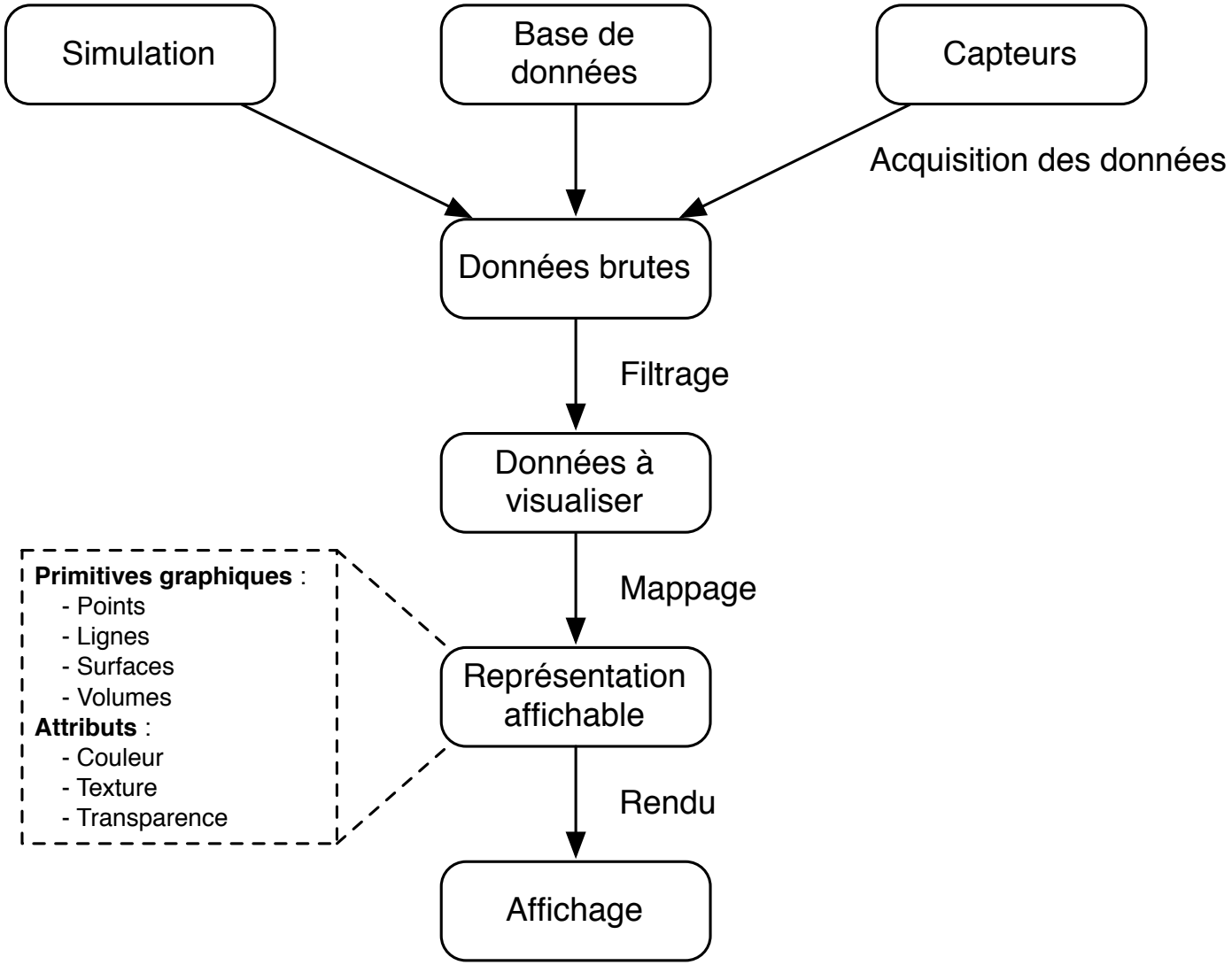
1. l'**acquisition des données**
2. le **filtrage**
3. le **mappage**
4. le **rendu**

# Pipeline de visualisation

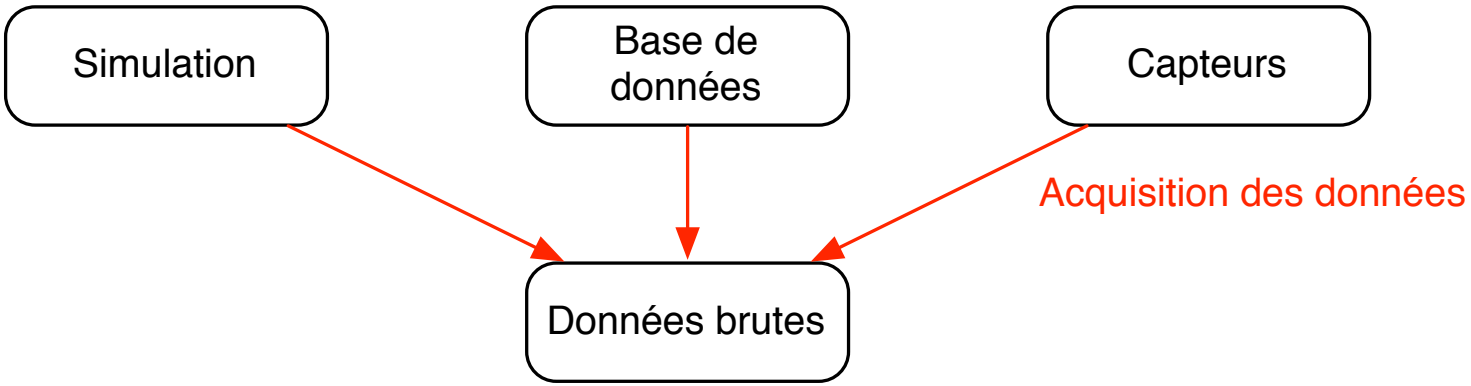



[4]

# Pipeline de visualisation



# Acquisition




L'**acquisition des données** brutes peut être faite de différentes façons : par des simulations (calculs informatiques), des enquêtes statistiques, des bases de données historiques, des capteurs de mesures réelles, etc.

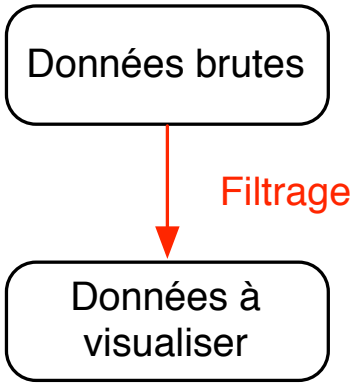
# Acquisition

En pratique, l'acquisition peut inclure différentes **modifications aux données** originales :

- Modification de l'étendue ($[\min, \max] \longrightarrow [0, 1]$)
- Échantillonnage (Continu $\longrightarrow$ Discret)
- Rééchantillonnage (Discret $\longrightarrow$ Discret)

Il faut garder à l'esprit que les choix faits lors de l'acquisition **détermineront la qualité de la représentation graphique** que l'on sera en mesure d'obtenir.

# Filtrage

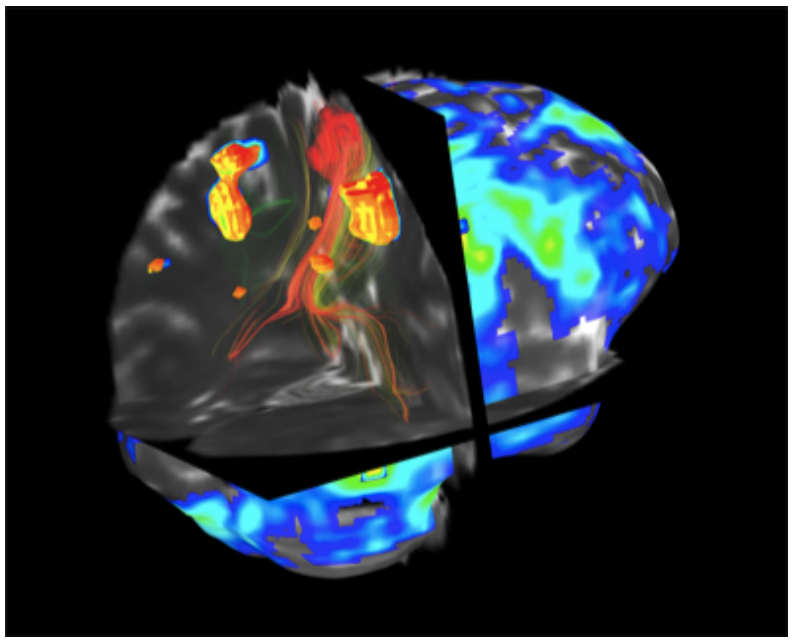



Le **filtrage** prépare les données brutes pour la suite du traitement. On peut par exemple éliminer des données inutiles, faire du débruitage ou faire de la segmentation.

# Filtrage

Attardons-nous à la pertinence de cette étape en s'intéressant à deux exemples, provenant de deux domaines différents :

- l'**imagerie médicale**
- la **finance**



# Filtrage

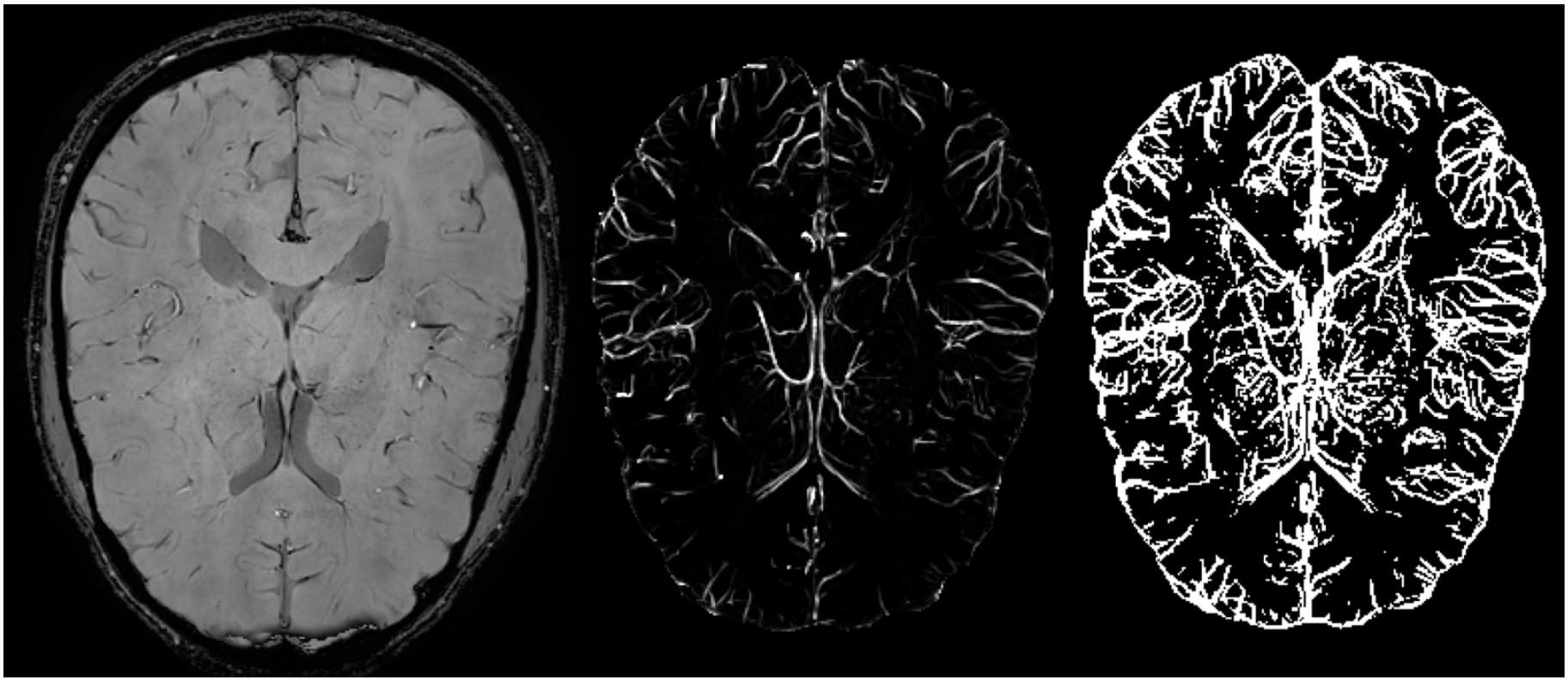

PLEASE STARE INTO MY EYES
FOR ONE MINUTE
YOUR CAT-SCAN IS NOW COMPLETE

# Filtrage

En **imagerie médicale**, un scanner IRM ou CT fournit énormément d'information... **Souvent beaucoup plus qu'il n'en faut**.

À la suite d'un AVC, par exemple, les médecins s'intéresseront avant tout au **système sanguin**, tandis que suite à une fracture, c'est l'information sur la **structure osseuse** qui sera considérée comme pertinente.

# Filtrage

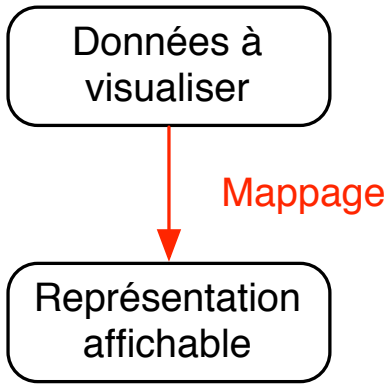


# Filtrage

En **finances**, une analyste reçoit de l'information sur **le cours de l'ensemble des actions** à la bourse.

Il peut toutefois n'être qu'intéressé par la valeur de celles d'**une seule compagnie**, de toutes les entreprises dans **un domaine particulier**, ou encore par **les titres que détient un client**.

# Mappage

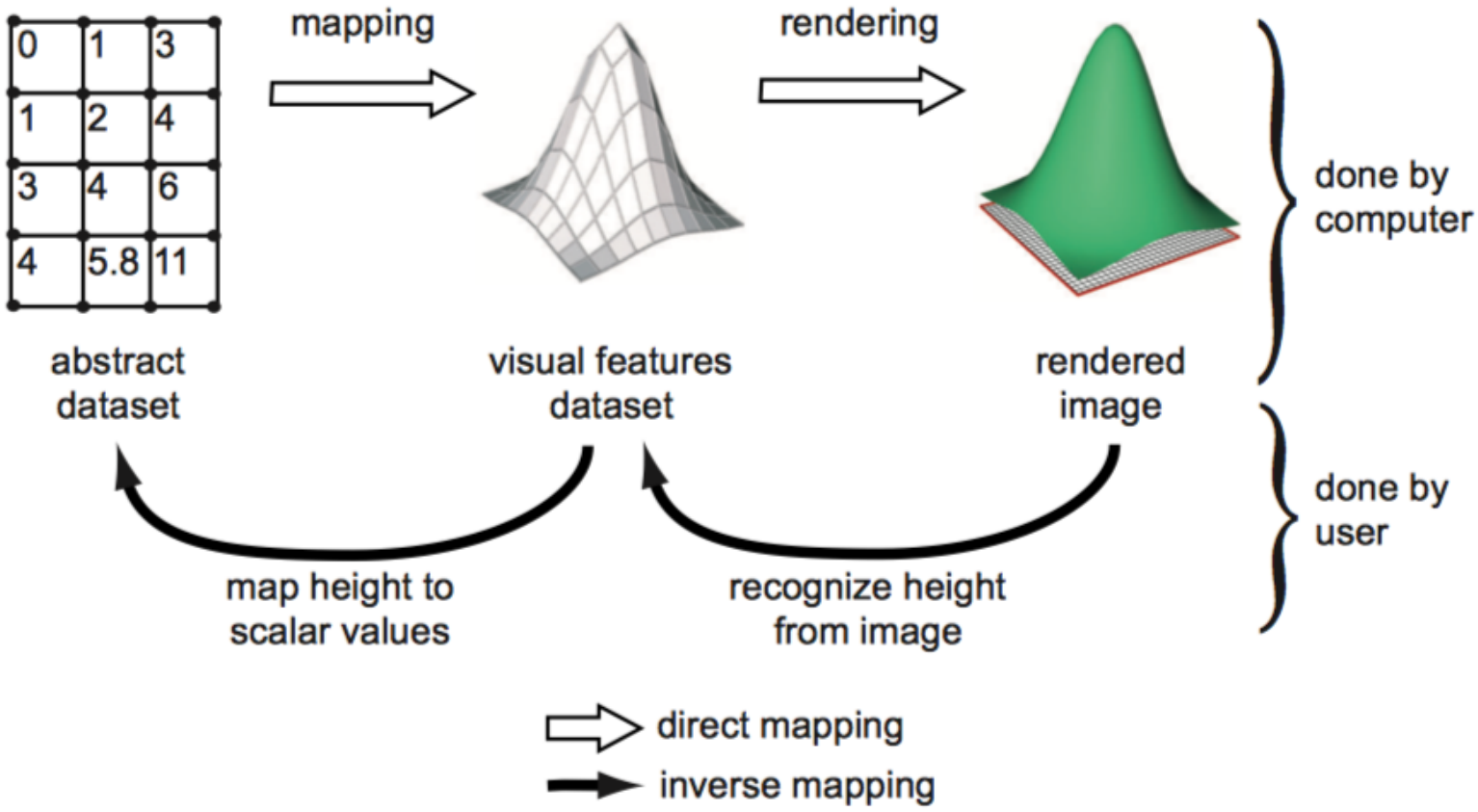



Le **mappage** est l'étape centrale du processus de visualisation. On cherche à mettre en correspondance les données avec des primitives graphiques connues.

On cherche ici à répondre au **quoi** et **comment** qu'on veut visualiser.

# Mappage

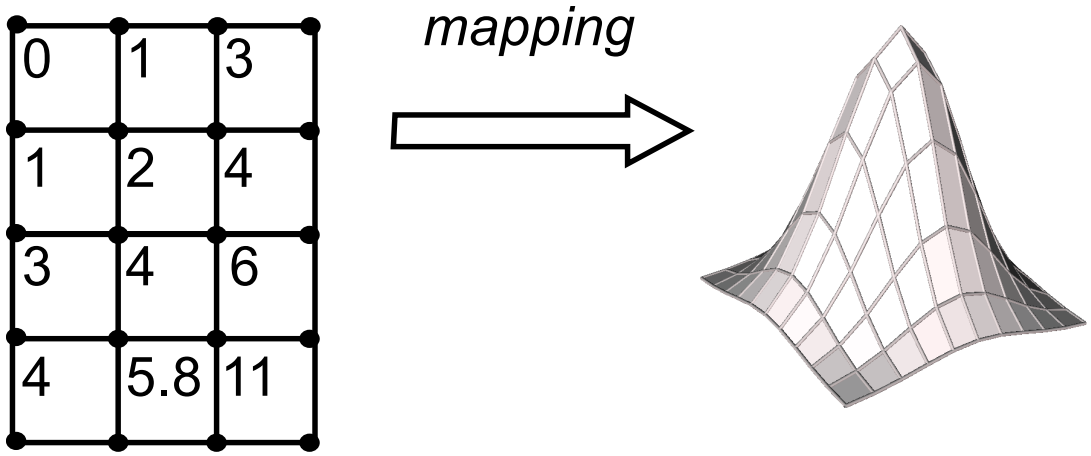
Le mappage est différent du rendu ; celui-ci a pour but de simuler une représentation graphique (scène) afin de facilité l'examination des données.




[4]

# Mappage

En plus des **propriétés géométriques** qui dépendent de la nature des données, on peut faire correspondre des **attributs visuels** comme la couleur et la texture aux données.

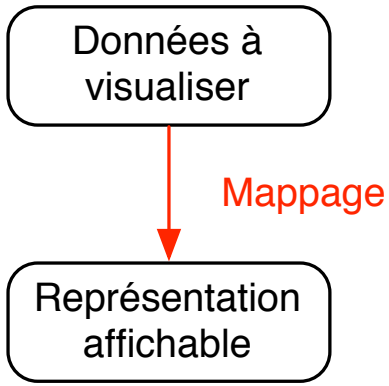



abstract dataset

visual features dataset

[4]

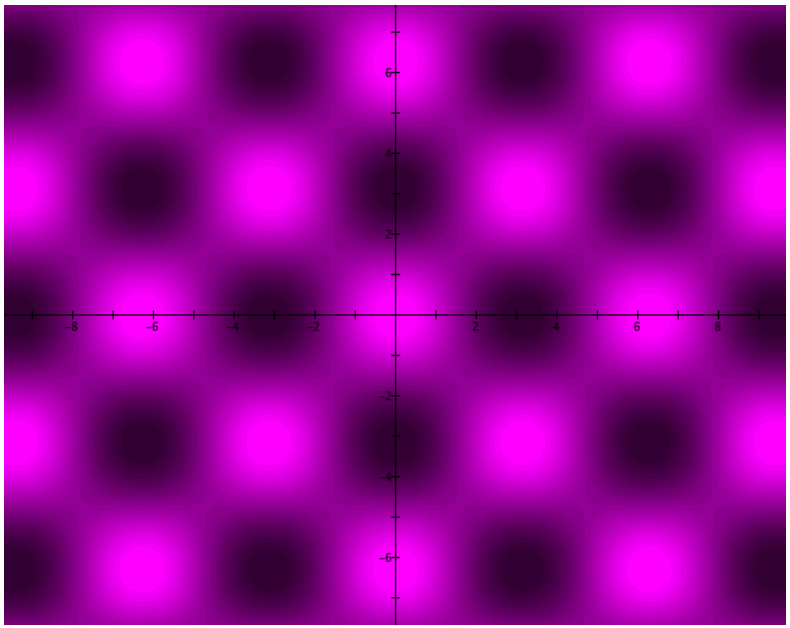
# Mappage

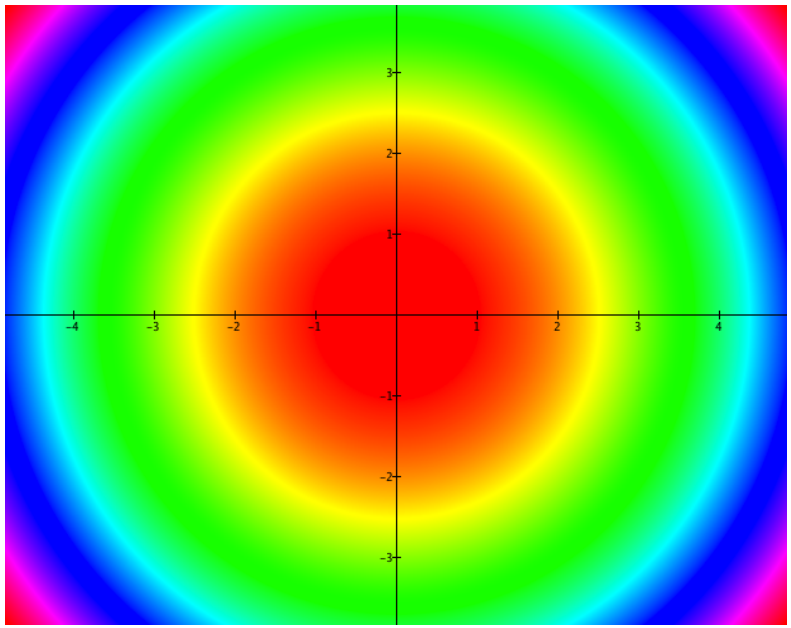



| Données | Primitive graphique |
|---|---|
| Scalaires 2D | Isocontour |
| Scalaires 2D | Carte de hauteurs |
| Scalaires 3D | Isosurface |
| Vecteurs 2D/3D | Champ vectoriel |
| Tenseurs 2D/3D | Champ de glyphes |

# Mappage

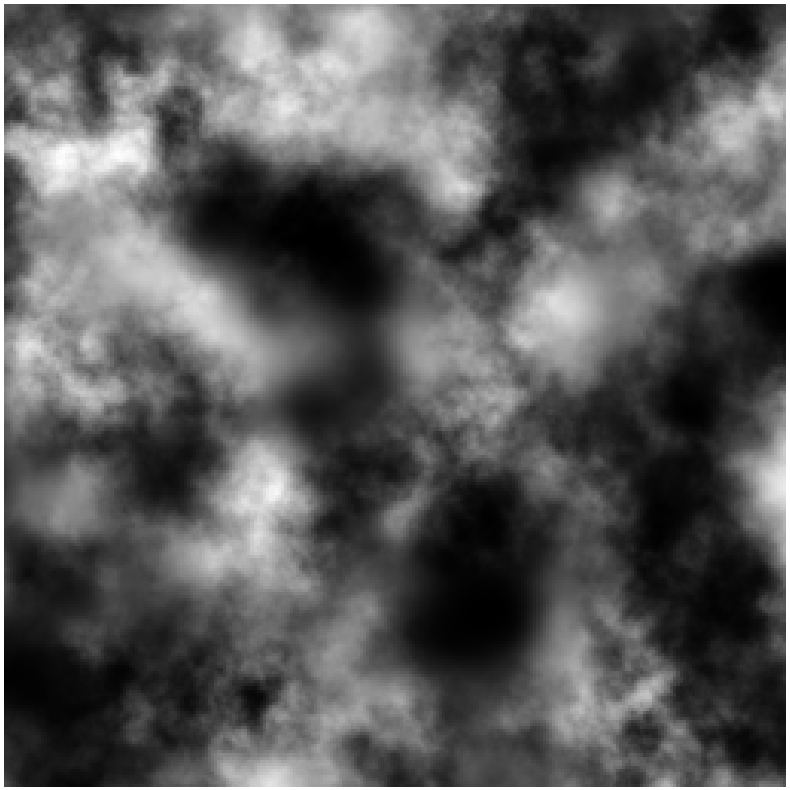
## Scalaires 2D : isocontours et code de couleurs

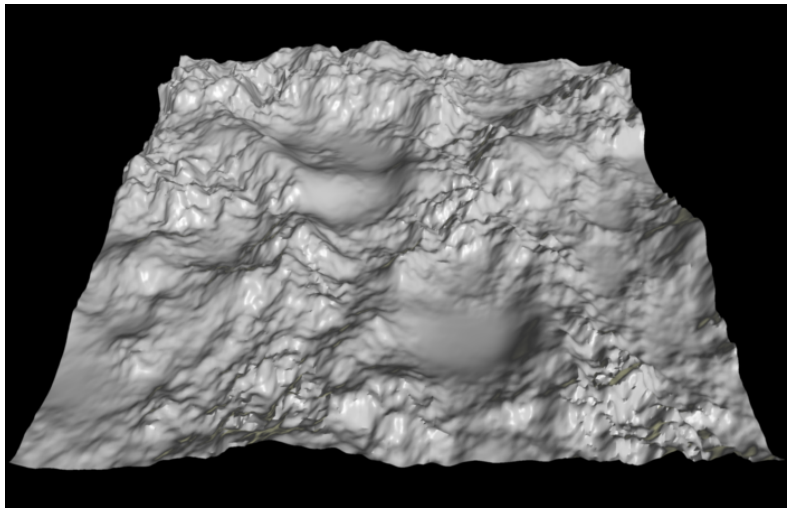




# Mappage

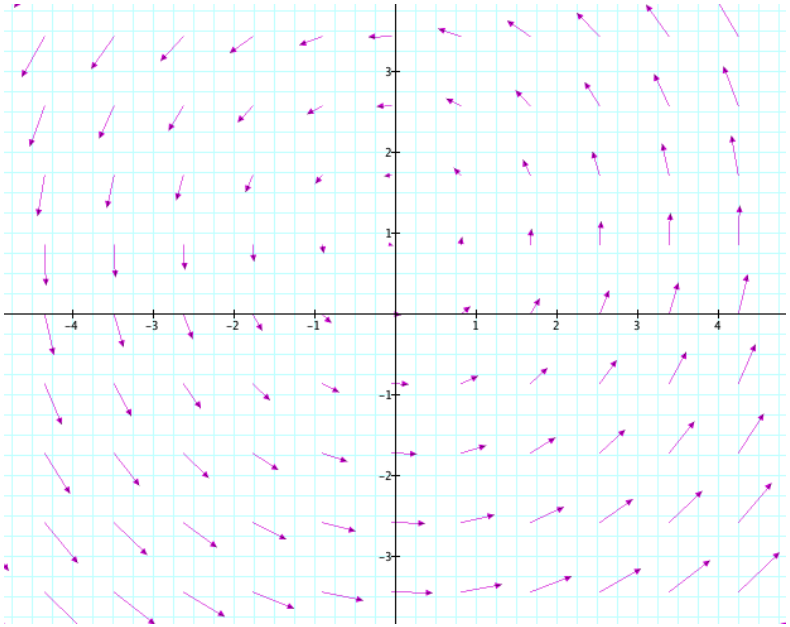
## Scalaires 2D : carte de hauteurs

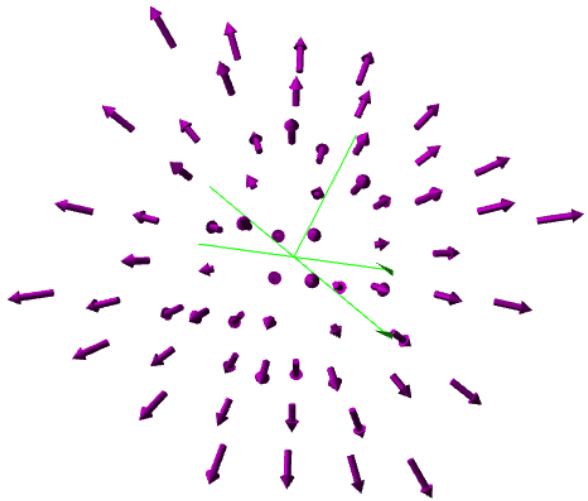




# Mappage

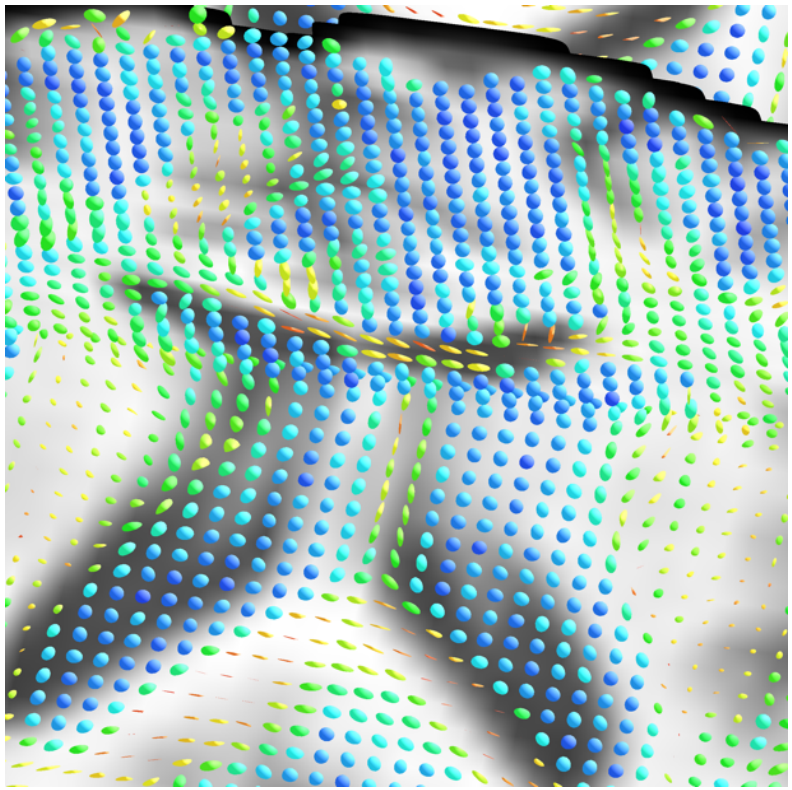
## Vecteurs 2D/3D : champ vectoriel



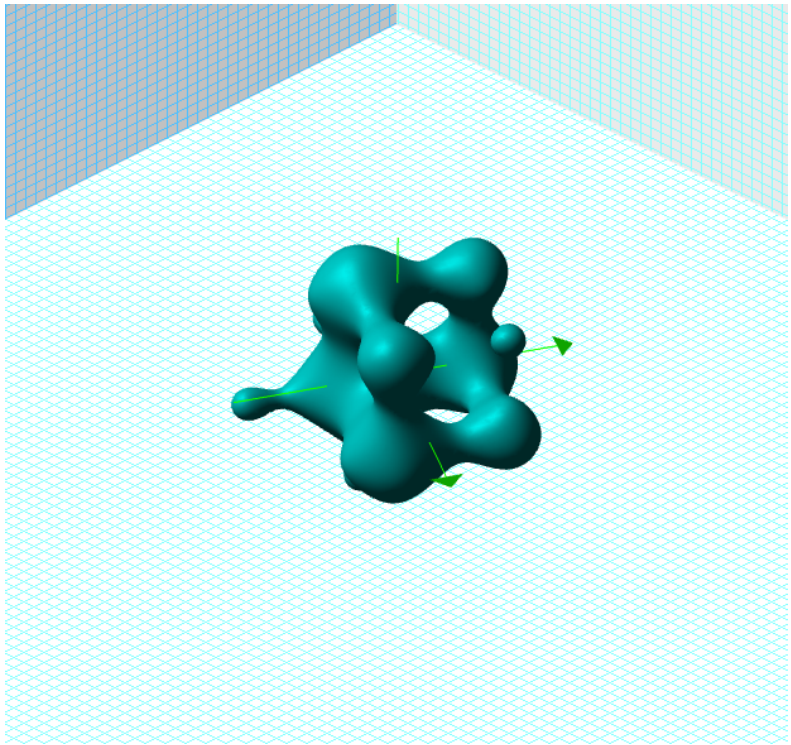
# Mappage

## Tenseurs 3D : champ de glyphes



# Mappage

## Champ 3D et volume



# Mappage

Tel que mentionné précédemment, le mappage et l'**étape centrale du processus de visualisation**. Les autres opérations du pipeline font essentiellement appel à des techniques provenant d'autres disciplines telles que les **statistiques**, le **traitement de signal** et l'**infographie**.

Ce n'est donc pas un hasard si les chapitres suivants porteront essentiellement sur cette portion du processus...

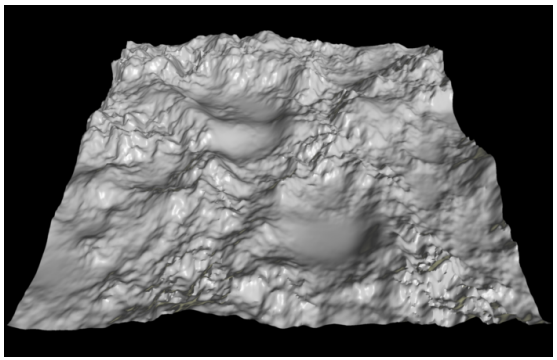
# Mappage

Parmi les propriétés que doit avoir le mappage, on retrouve l'**injectivité**.

C'est donc dire que **deux données différentes** $x_1 \neq x_2$ doivent correspondre à des **valeurs d'attributs visuels différents**.

# Mappage

À titre d'exemple, considérons la carte de hauteur d'une fonction $z = f(x, y)$.



On doit être en mesure d'**inverser mentalement la représentation graphique** pour estimer les valeurs relatives de $z$. Si le mappage n'est pas injectif, deux valeurs de $z$ différentes pourraient avoir la même hauteur dans la carte.

# Mappage

Une autre propriété fort utile pour le mappage est celle de **conservation des distances**.

Si deux points sont séparés par une distance $d$ dans l'espace des données, alors leur distance $d'$ dans l'espace de visualisation devrait **refléter la distance originale**.

La façon la plus simple d'y arriver est de maintenir une relation de **proportionnalité directe entre les distances**.

# Mappage

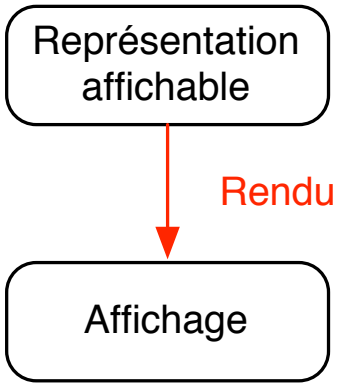
Soient quatre points $x_1$, $x_2$, $x_3$ et $x_4$ tels que

$$d(x_1, x_2) = d_i$$
$$d(x_3, x_4) = d_j$$

dans l'espace des données. Si dans l'espace de visualisation ces distances deviennent $d'_i$ et $d'_j$, respectivement, on devra avoir que

$$\frac{d_i}{d_j} = \frac{d'_i}{d'_j}.$$

# Rendu




Dans l'étape de **rendu**, les données géométriques sont converties en information visuelle. Pour atteindre un bon niveau de détails, on applique des techniques appartenant au domaine de l'infographie (illumination, textures, projection, etc.)

# Sources d'erreur

Chaque étape du pipeline **peut être la source de plusieurs erreurs**. Celles-ci peuvent grandement altérer les résultats jusqu'à les rendre complètement faux.

Il est donc crucial de savoir comment minimiser les erreurs en évitant certains pièges.

# Sources d'erreur

1. **Acquisition des données**
   - L'échantillonnage est-il assez précis pour qu'on soit en mesure d'obtenir l'information souhaitée ? Inversement, est-il trop fin ? Il ne faut pas considérer des données inutiles qui ne feraient qu'alourdir les calculs.
   - Est-ce que la quantification se fait avec assez de précision pour être en mesure de faire ressortir les caractéristiques souhaitées ?

# Sources d'erreur

2. **Filtrage**

   - Conserve-t-on les données importantes et significatives ? Au contraire, élimine-t-on les données non pertinentes à l'extraction des caractéristiques souhaitées ?
   - S'il est nécessaire d'ajouter des données par interpolation, quelle forme doivent prendre celles-ci ? À partir de quels points doit-on faire l'interpolation ? Les données ajoutées sont elles représentatives du reste ?

# Sources d'erreur

3. **Mappage**
   - Le choix de la primitive graphique est il approprié pour représenter le type d'information que l'on souhaite obtenir des données ?

# Sources d'erreur

4. **Rendu**

   - Cherche-t-on à avoir un rendu interactif ? Si oui, des compromis devront être faits sur les techniques utilisées pour permettre une navigation rapide à travers les données.
   - Il est important de savoir où seront affichées les images et de considérer les limitations techniques pour ne pas inutilement faire de rendus complexes.
   - L'ajout de réalisme (illumination, transparence, etc.) n'ajoute pas nécessairement de valeur informative à vos images. Plus beau ne rime pas toujours avec plus significatif.

# Introduction à VTK

# Visualization Toolkit

La librairie **Visualization Toolkit** (VTK) est devenue un standard dans le domaine du traitement d'images et de la visualisation.

Pour en maîtriser le fonctionnement, il faut

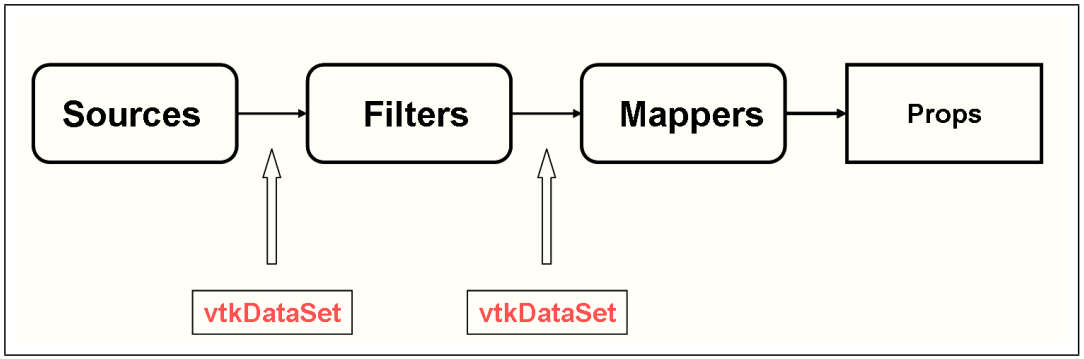
1. comprendre la **structure des objets** dans la librairie ; et
2. comprendre l'**organisation en pipeline** de la librairie.

# Visualization Toolkit

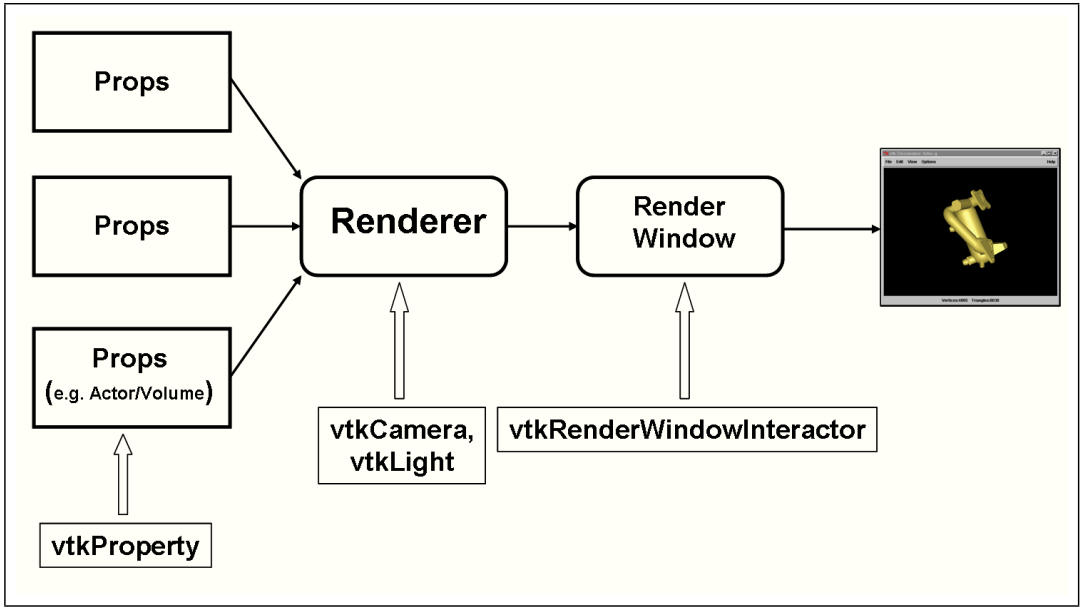
VTK est une **immense librairie** dont la structure de classes peut faire peur. Une fois qu'on a saisi quelques concepts de base, elle peut toutefois devenir **un outil très puissant**.

La plupart des opérations dans VTK sont faites en respectant un **pipeline de traitement** où différents éléments sont attachés ensembles pour effectuer des tâches complexes.

# Visualization Toolkit




[1]




[1]

# Visualization Toolkit

- ***Sources*** : Ce sont les classes de VTK qui permettent d'obtenir les données. À titre d'exemple, `vtkJPEGReader` lit le contenu d'un fichier *.jpg*.
- ***Filters*** : Ce sont les classes de VTK qui permettent de modifier les données originales pour les rendre plus adaptées à l'analyse. À titre d'exemple, `vtkImageGaussianSmooth` applique un filtre Gaussien sur une image.
- ***Mappers*** : Ce sont les classes de VTK qui permettent de définir les liens entre les données et les primitives graphiques. Plusieurs *Mappers* peuvent partager les mêmes données et permettre leur affichage de différentes façons.

# Visualization Toolkit

- ***Props/Actors*** : Ces classes de VTK prennent en entrée les données produites par un *Mapper* et génèrent la représentation visuelle 3D des données. Un *Mapper* ne doit pas être associé à plus d'un *Props*.
- ***Renderer*** : Ce sont les classes de VTK qui génèrent les images 2D à partir de données 3D. On leur associe des *Actors*, mais aussi des informations sur l'éclairage et la caméra virtuelle.
- ***Render Window*** : Ce sont les classes de VTK qui définissent l'espace occupé à l'écran par le résultat du *Renderer*.

# Fonctionnement

# Fonctionnement

```
#include <QApplication>

#include <vtkSmartPointer.h>
#include <vtkSphereSource.h>
#include <vtkPolyDataMapper.h>
#include <vtkActor.h>
#include <vtkImageViewer.h>
#include <vtkRenderWindowInteractor.h>
#include <vtkInteractorStyleImage.h>
#include <vtkRenderer.h>
#include <vtkJPEGReader.h>

#include <QVTKWidget.h>
```

# Fonctionnement

```
int main(int argc, char** argv)
{
  QApplication app(argc, argv);

  QVTKWidget widget;
  widget.resize(256,256);

  // ... VTK STUFF (Page suivante)

  widget.SetRenderWindow(renderWindow);
  widget.show();

  app.exec();

  return EXIT_SUCCESS;
}
```

# Fonctionnement

```
//setup sphere
vtkSmartPointer<vtkSphereSource> sphereSource = vtkSmartPointer<vtkSphereSource>::New();
sphereSource->Update();

vtkSmartPointer<vtkPolyDataMapper> sphereMapper = vtkSmartPointer<vtkPolyDataMapper>::New();
sphereMapper->SetInputConnection(sphereSource->GetOutputPort());

vtkSmartPointer<vtkActor> sphereActor = vtkSmartPointer<vtkActor>::New();
sphereActor->SetMapper(sphereMapper);

//setup window
vtkSmartPointer<vtkRenderWindow> renderWindow = vtkSmartPointer<vtkRenderWindow>::New();

//setup renderer
vtkSmartPointer<vtkRenderer> renderer = vtkSmartPointer<vtkRenderer>::New();
renderWindow->AddRenderer(renderer);
renderer->AddActor(sphereActor);
renderer->ResetCamera();
```

# Installation de Qt + VTK (Windows)

1. Télécharger et installer **CMake**

   `http://www.cmake.org/cmake/resources/software.html`

2. Télécharger et installer **Qt Library** pour Visual Studio 2010

   `http://qt.nokia.com/downloads`

3. Ajouter **un lien vers la librairie Qt** dans la variable d'environnement `PATH`.

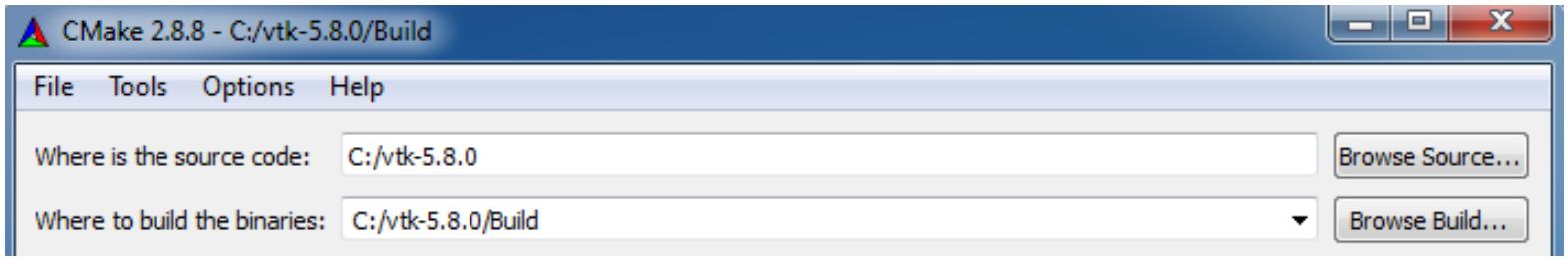
   `[Chemin d'installation de Qt]\Qt\4.8.1\bin`

# Installation de Qt + VTK (Windows)

4. Télécharger le code source de la librairie **VTK** et le décompresser

   `http://www.vtk.org/VTK/resources/software.html`

5. Dans CMake, indiquer **l'emplacement du dossier** où se trouve le code source de la librairie VTK et fournir un **dossier de Build**.

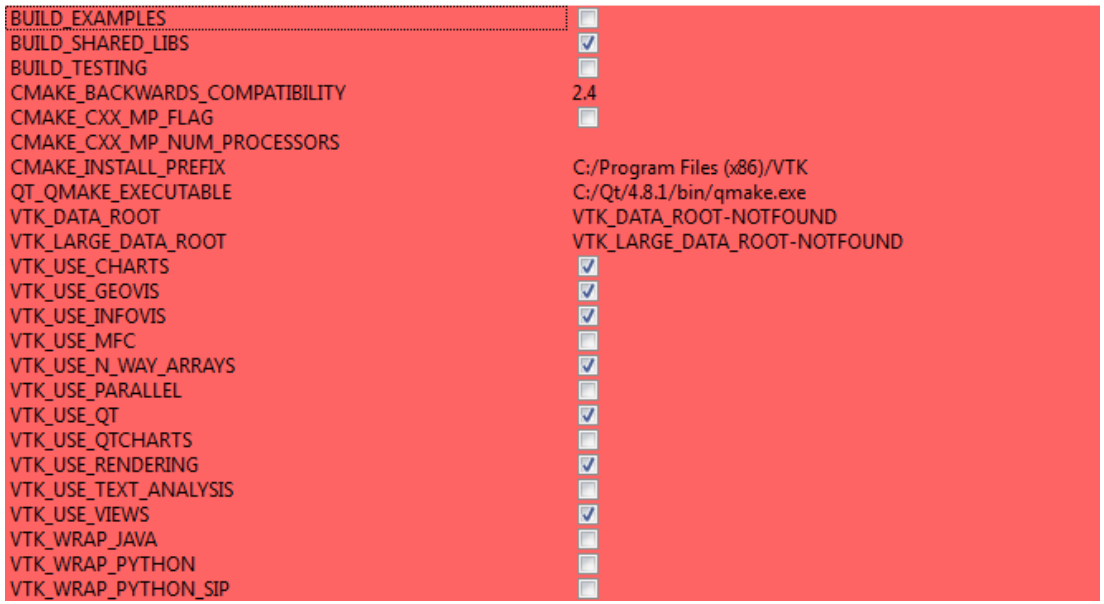



6. Appuyer sur **Configure**.

# Installation de Qt + VTK (Windows)

7. Une fois la configuration terminée, assurez-vous que les options `BUILD_SHARED_LIBS` et `VTK_USE_QT` soient cochées et que `BUILD_EXAMPLES` soit décochée.

8. Vérifier que l'option `QT_QMAKE_EXECUTABLE` soit bien définie.



# Installation de Qt + VTK (Windows)

9. Appuyer sur **Configure**.

10. S'il n'y a pas d'erreurs supplémentaires, appuyer sur **Generate**.

11. Dans le dossier de Build défini dans CMake, vous trouverez une solution Visual Studio 2010 pour compiler la librairie VTK. **Effectuer la compilation en Debug, puis en Release**. Cette étape prendra un certain temps.

12. Ajouter **un lien vers le dossier de Build** dans la variable d'environnement `PATH`.

# Références

# Références I


X. Papademetris and A. Joshi.
Introduction to programming for image analysis with vtk, 2009.


W. Schroeder, K. Martin, and B. Lorensen.
The visualization toolkit : An object-oriented approach to 3d graphics, 2006.


C. Taras, T. Ertl, R. Botchen, and I. Entina.
Online course for scientific visualization, 2007.


A. C. Telea.
Data visualization : Principles and practice, 2008.