

Mieux développer avec

Qt-Designer

&

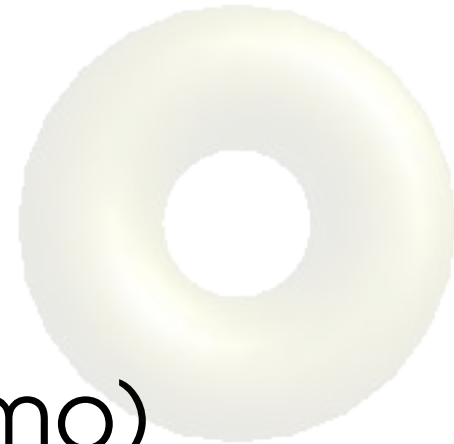
Présenté par

Yassine CHAUCHE
&
Tarik ALLA




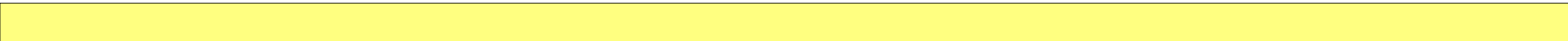
KDevelop

Sommaire

- [K'est-ce que c'est ? \(intro\)](#)
- Tour d'horizon (démono)
- Signaux et récépteurs
- Première application (démono)
- Conclusion



K'est-ce que c'est ?

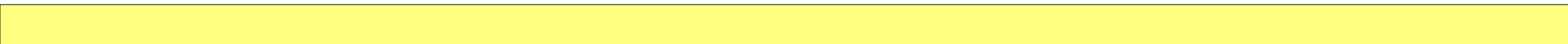
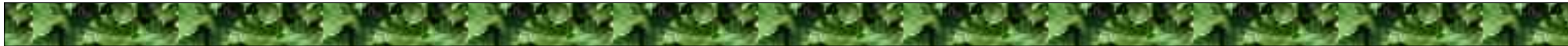
- 
- A quoi ça sert ?
 - Pourquoi Kdevelop ?
 - Pourquoi Qt-Designer ?
 - KDE ?
- 
- 
- 

A quoi ça sert ?

The logo for KDevelop is a dark blue rounded rectangle with a starry night sky pattern. The word "KDevelop" is written in a white serif font in the center.

KDevelop

- C'est un IDE
- Il regroupe plusieurs outils
- Tout ce qu'il faut pour programmer
- Supporte plusieurs langages



A quoi ça sert ?

The QT logo is displayed in white serif font on a dark blue background with a pattern of small white stars, resembling a night sky. The logo is centered within a rounded rectangular frame.

QT

- C'est une bibliothèque
- Orientée Objet (C++)
- Portable
- Moderne
- Complexe mais pas compliquée
- Complète
- Gratuite (sous conditions)
- Une suite d'outils qui l'accompagnent

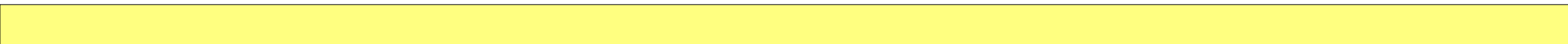
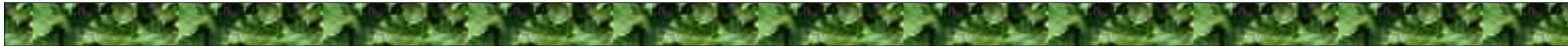
A quoi ça sert ?

QT-Designer

- Constructeur de GUI pour QT
- Ce N'EST PAS un IDE
- Ne produit aucun code
- Données au format libre (XML)
- Gratuit

K'est-ce que c'est ?

- A quoi ça sert ?
- Pourquoi KDevelop ?
- Pourquoi Qt-Designer ?
- KDE ?



Pourquoi KDevelop ?

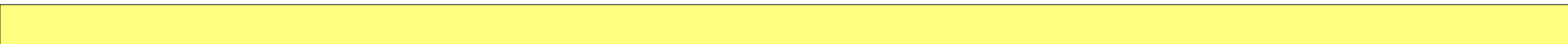
The logo for KDevelop, featuring the word "KDevelop" in a white serif font centered on a dark blue rectangular background with a pattern of small white stars.

KDevelop

- Puissant éditeur de texte
- Supporte la complétion de code
- Puissant debugger
- Makefile, autoconf, config, install etc.
- Repérage directe des erreurs dans le code source.
- Compatible KDE(Développement de plug-ins)
- Supporte beaucoup de langages
- Intégration de tous les outils nécessaires
- Ergonomique et pratique

K'est-ce que c'est ?

- A quoi ça sert ?
- Pourquoi Kdevelop ?
- Pourquoi Qt-Designer ?
- KDE ?



Pourquoi QT ?

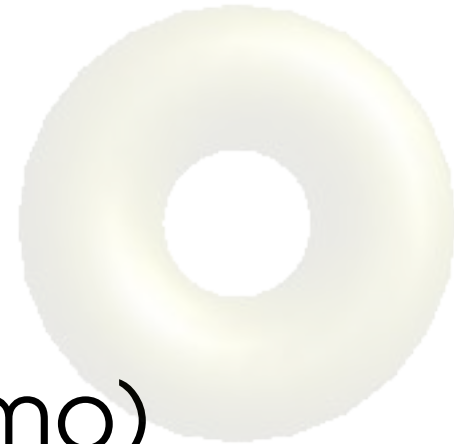


QT

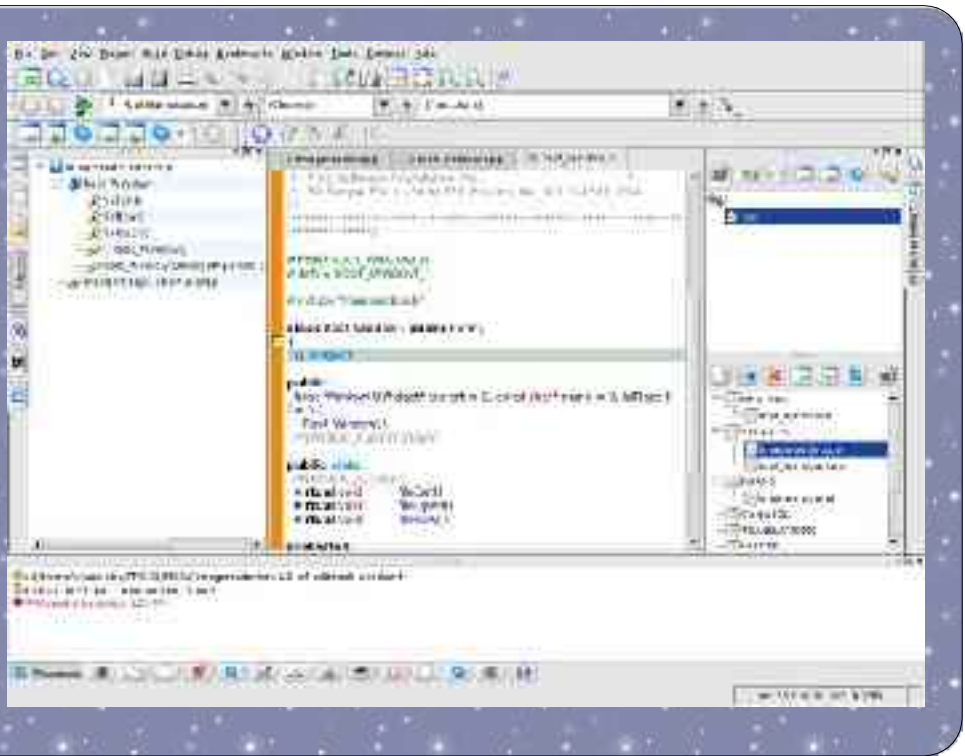
- Librairie très riche, complète, et performante
- Portable sur les systèmes Unix-like, Windows, et MacOs
- Est à la base du projet KDE
- Documentation détaillé, tutoriaux et manuels d'utilisation en-ligne
- Plusieurs modules dispos (SQL, Réseau, OpenGL, XML...)
- Outils de construction d'interface graphique (Qt-Designer, uic, qmake, etc.)
- Possibilité de traduire en plusieurs langues

Sommaire

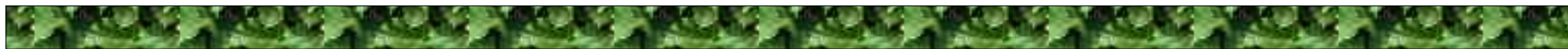
- K'est-ce que c'est ? (intro)
- [Tour d'horizon \(démono\)](#)
- Signaux et récépteurs
- Première application (démono)
- Conclusion



Tour d'horizon

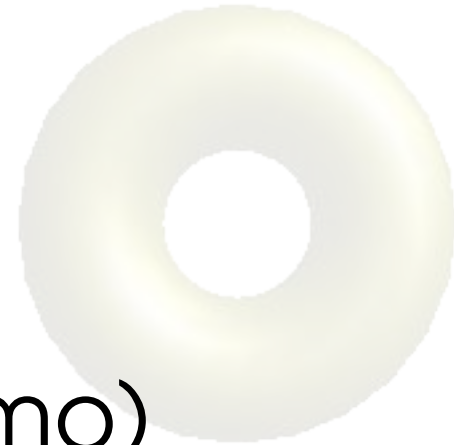


(Demo -- 15 minutes)



Sommaire

- K'est-ce que c'est ? (intro)
- Tour d'horizon (démono)
- [Signaux et récepteurs](#)
- Première application (démono)
- Conclusion



Signaux et récepteurs

- Widgets
- Signals (signaux)
- Slots (récepteurs)
- QObject
- MOC


Widgets



Widgets

- Composant de base d'une interface graphique QT (classe racine)
- Votre widget doit en hériter
- Plusieurs QWidgets sous-jacents spécialisés, à vous de trouver celui qui vous conviens.
- Toutes les opérations de dessins doivent se trouver dans la méthode paintEvent()
- Chaque QWidget est fils d'un autre QWidget
- Si le père d'un QWidget est NULL, alors QWidget est indépendant

Signaux et récepteurs

- 
- Widgets
 - Signals (signaux)
 - Slots (récepteurs)
 - QObject
 - MOC
- 
- 
- 

Signals (Signaux)

Signaux

- Un QWidget peut interagir avec d'autres en leur envoyant un signal
- Ce signal provoque chez les Widgets intéressés la déclenchement d'une action
- Un signal peut être émis explicitement à travers la méthode emit(), ou implicitement en implémentant une méthodes qui capture un évènement utilisateur (évènements souris, clavier, changement de focus etc.)
- Un signal doit être décalré dans un bloc de méthodes, contenant le mot clé **signals**
- Usuellement, les signaux sont déclarés **protected**
- Un signal peut être émis vers différents Widgets

Signaux et récepteurs

- 
- Widgets
 - Signals (signaux)
 - Slots (récepteurs)
 - QObject
 - MOC
- 
- 
- 

Slots (Récepteurs)

Récepteurs

- Un QWidget peut implémenter une méthode, qu'il déclare comme étant un slot, pour intercepter les signaux d'autres QWidget
- Le mot clé **slots** doit précéder la déclaration d'un slot.
- Les récepteurs doivent être déclaré **protected** si on veut que les QWidget héritiers puissent les exploiter
- Un récepteur peut intercepter plusieurs signaux

Signaux et récepteurs

- Widgets
- Signals (signaux)
- Slots (récepteurs)
- QObject
- MOC

QObject

QObject

- Un QWidget peut ne pas avoir besoin d'implémenter de signal, ni de récepteur.
- Un QWidget implémentant un signal et/ou un slot devra être précédé avant toute déclaration du mot clé **QObject**
- Pour que le signal du QWidget A puisse exciter le récepteur du QWidget B, il faut créer une connexion entre les deux par la méthode `QObject::connect(A,nom_signalA,B,nom_slotB)`
- Répéter autant de fois que nécessaire les connexions entre un signal et tous les slots à connecter

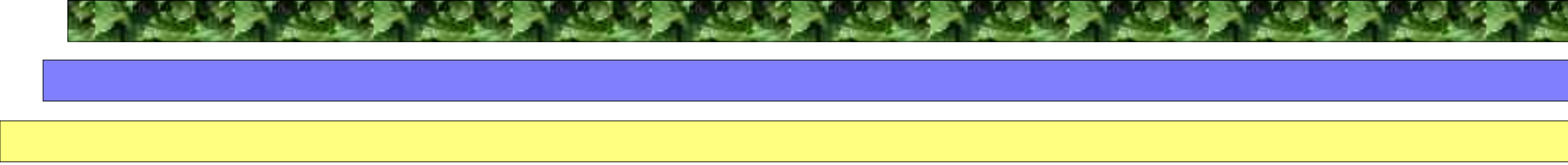
Signaux et récepteurs

- Widgets
- Signals (signaux)
- Slots (récepteurs)
- QObject
- MOC

Meta Object Compiler

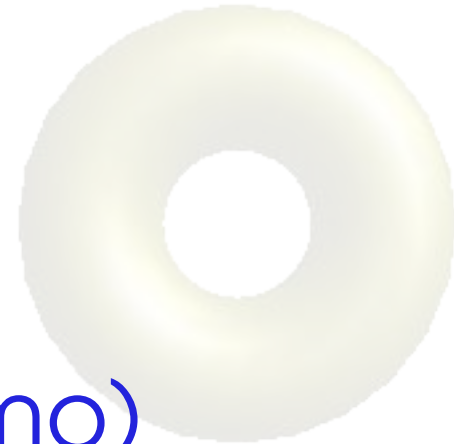


MOC

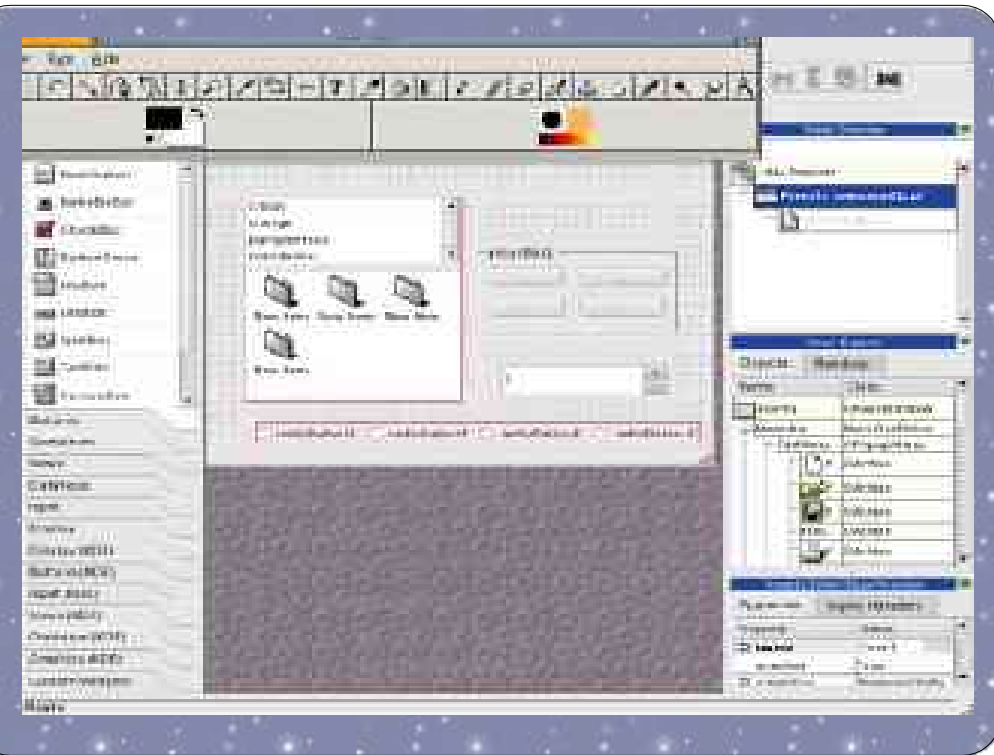
- Les mots clés : **QObject**, **slots** et **signals** ne sont pas des mots clés C++ standards.
 - Les fichiers sources C++ contenant ces mots clés doivent être traités avec moc
 - Des fichiers `moc_nomfichier.cpp` et `moc_nomfichier.h` sont générés
 - KDevelop prends en charge toutes les opérations de compilation nécessaires.
- 

Sommaire

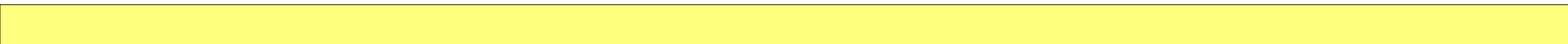
- K'est-ce que c'est ? (intro)
- Tour d'horizon (démono)
- Signaux et récepteurs
- Première application (démono)
- Conclusion



Première Application

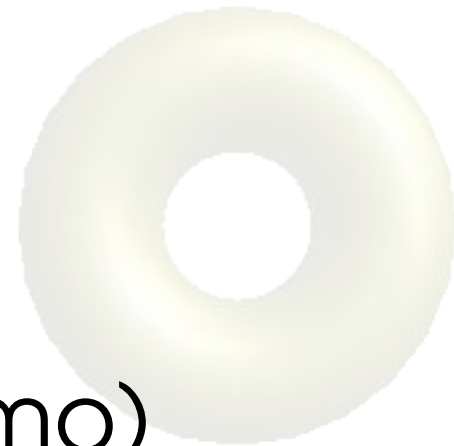


Démo (30 min)



Sommaire

- K'est-ce que c'est ? (intro)
- Tour d'horizon (d emo)
- Signaux et r ecepteurs
- Premi ere application (d emo)
- [Conclusion](#)



Conclusion

- Méthodologie
- Dérivation ou génération de code ?
- Les développements à venir
- Critiques

Méthodologie

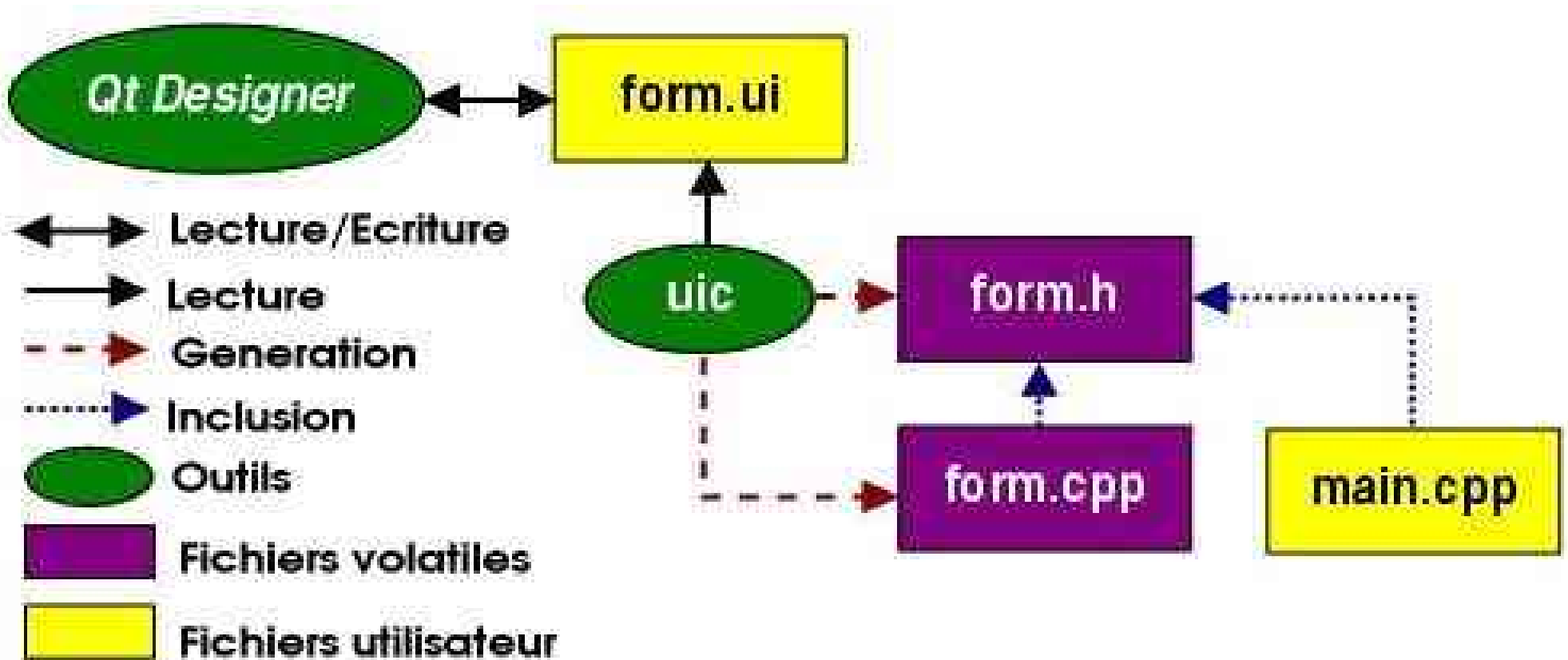
- Ouvrir QT-Designer
- Construire une interface graphique
- Sauvegarder sous le format .ui
- Ouvrir KDevelop
- Construire un nouveau projet QT
- Dans le QManager :
 - Inclure les sources dans le répertoire SOURCES
 - Inclure les headers dans le répertoire HEADER
 - Inclure l'interface graphique (fichier.ui) dans le répertoire FORMS
- Implémenter vos signaux et récepteurs dans une nouvelle classe
Héritée de la classe parente de votre GUI
- Compiler en appelant BUILD PROJECT
- KDevelop se charge de tout le reste

Conclusion

- Méthodologie
- Dérivation ou génération de code ?
- Les développements à venir
- Critiques

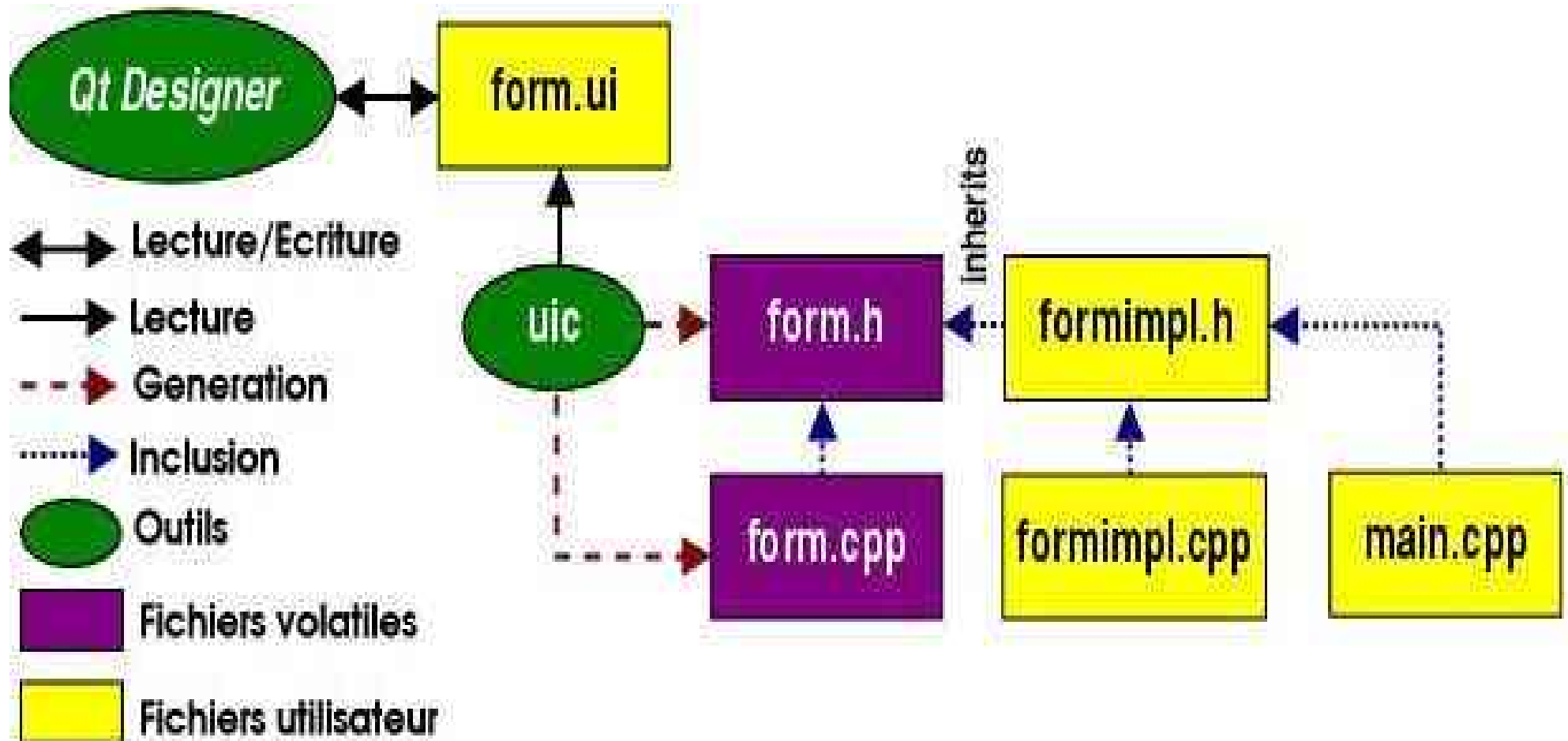
Dérivation / Génération de code

- Toute modification apportée au code généré automatiquement à partir de l'interface graphique (fichier .ui) sera perdue à chaque recompilation



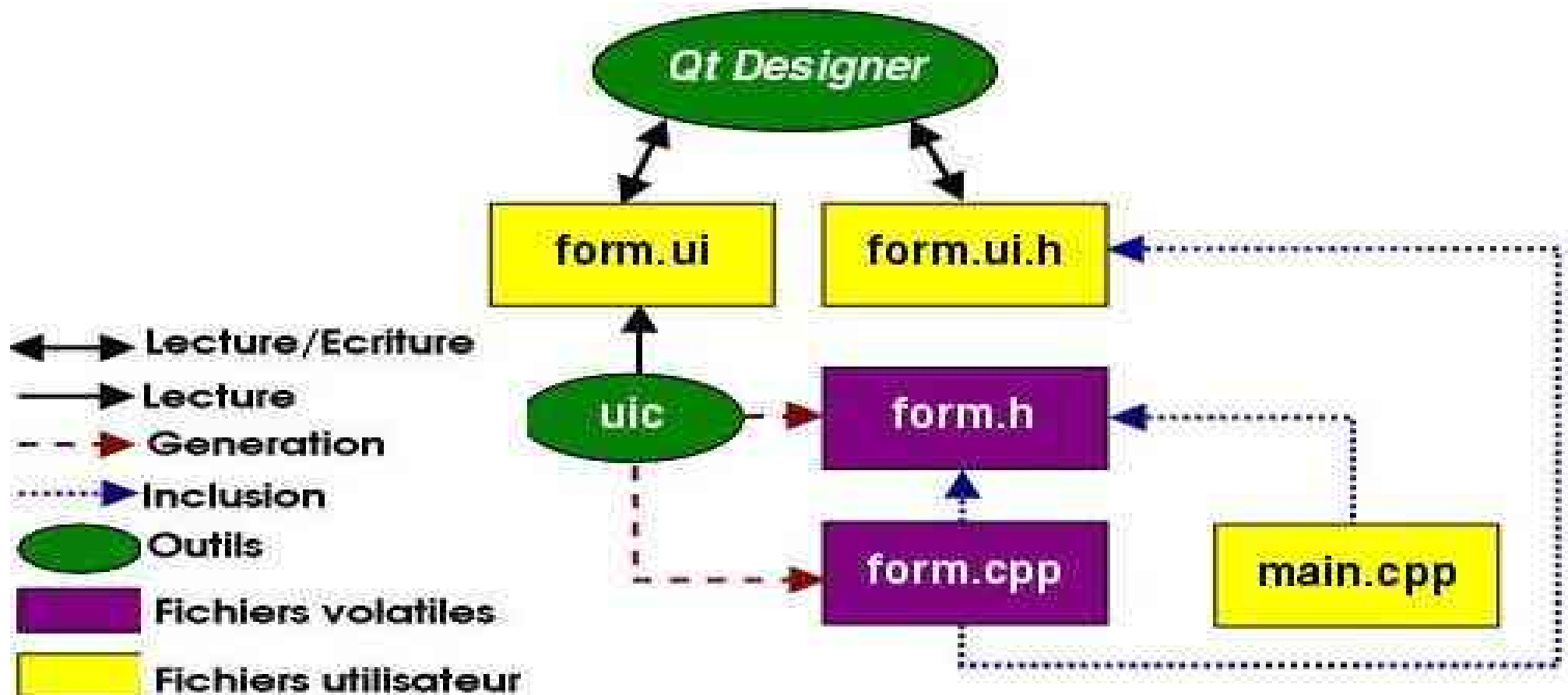
Dérivation / Génération de code

- Pour éviter ce désagrément, il faudra écrire une classe qui dérive de la classe principale de votre interface et y implémenter vos slots et signaux.



Dérivation / Génération de code

- Une autre méthode consiste à généré du code cpp embarqué dans l'interface graphique. Cette méthode est à déconseiller car elle disparaîtra dans les prochaines versions de QT.





Conclusion

- 
- Méthodologie
 - Dérivation ou génération de code ?
 - Les développements à venir
 - Critiques
- 
- 
- 

Développements à venir

- Développement 1
- Développement 2
- Développement 3

Conclusion

- 
- Méthodologie
 - Dérivation ou génération de code ?
 - Les développements à venir
 - Critiques
- 

Critiques



LES MOINS

- Complétion partiel du code (ne reconnaît pas les méthodes des bibliothèques)
- Obligation de recourir à un constructeur externe d'interface graphique
- La dérivation de classe est une solution non intuitive pour l'implémentation des slots et signaux
- KDevelop reste encore truffés de bugs (multiples erreurs de segmentations)
- QT-Designer ne gère pas encore un projet complet (interface + implémentation)

Critiques

LES PLUS

- Interface intuitive et ergonomique pour les deux produits
- Palette d'outils appréciable sous KDevelop et accompagnant QT
- Des centaines de widgets prêt à l'emploi
- Très bonne documentation riche en exemples ainsi que des tutoriels bien formés
- Large communauté d'utilisateurs (forums, Usenet, IRC, sites web dédiés...)

En-Ligne

- Qt-Assistant (commande « assistant»)
- Documentation sous KDevelop (panneau doc)
- Pages de manuels (commande man)

Sur IRC

Serveur : irc.kde.org

Canaux : #kdevelop

#qt

#kde-devel

Sur Le WEB :

- page d'accueil du site des créateurs :

<http://www.trolltech.com/>

- livre en ligne "C++ GUI PROGRAMMING WITH QT" :

http://phptr.com/content/images/0131240722/downloads/blanchette_book.pdf

- Page d'accueil des vidéos de démo :

<http://www.trolltech.com/video/index.html>

- Vidéo d'introduction à QT :

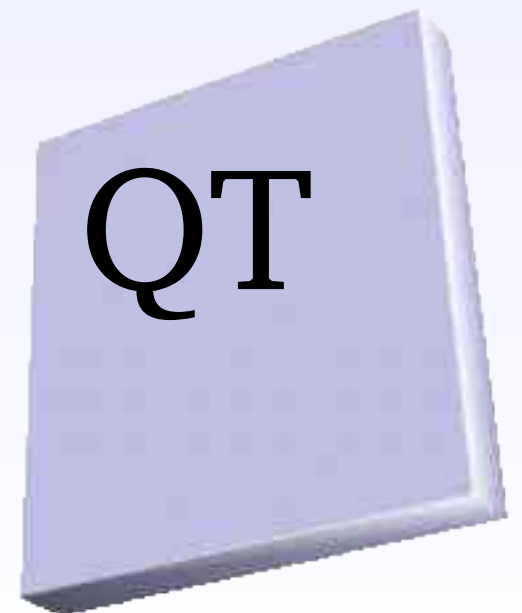
<http://www.trolltech.com/video/overview.html>

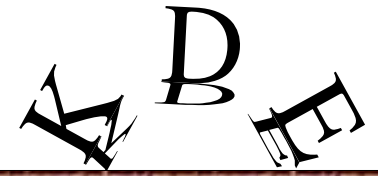
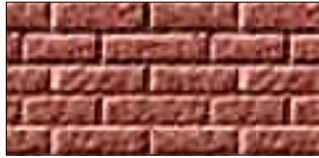
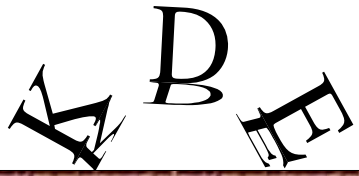
- Vidéo d'explication des signaux et slots :

<http://www.trolltech.com/video/signalsslots.html>

- Site francophone de la communauté QT:

<http://prog.qt.free.fr/portal.php>





KDEVELOP

- Page d'accueil :

<http://www.kdevelop.org/>

KDEVELOP - QT :

- Tutoriel qt et kdevelop

<http://women.kde.org/articles/tutorials/kdevelop3/fr/>

KDE:

- En savoir plus sur KDE, en français :

http://www.kde-france.org/article.php3?id_article=50

LIVRES

"Programmer avec QT, 2nd édition" chez oreilly & associates