Architecture d'un programme interactif graphique

Organiser un projet selon **Model-View-Controller (MVC)**

Plan:

- Les 3 types de dialogue utilisateur
- Approche Model-View-Controller
- Illustration dans le context du projet

Les 3 styles de dialogues humain-machine

Point essentiel

- **ligne de commande** (commandes UNIX-LINUX)
  - passage de paramètres au programme avec **argc** et **argv**

Une seule «lecture»

- **Conversationnel** (programmes semestre d'automne)
  - **cout**, **cin**, etc disponible avec la bibliothèque standard C++

Lecture bloquante

- **Interface Graphique Utilisateur** (*Graphic User Interface* / *GUI*)
  - widgets (boutons, etc...) disponible dans ***bibliothèques***

Lecture non-bloquante

Bibliothèque (*Library*) : regroupe le code objet de plusieurs modules réalisant une tâche bien définie (exemple: interface graphique, dessin, etc...)

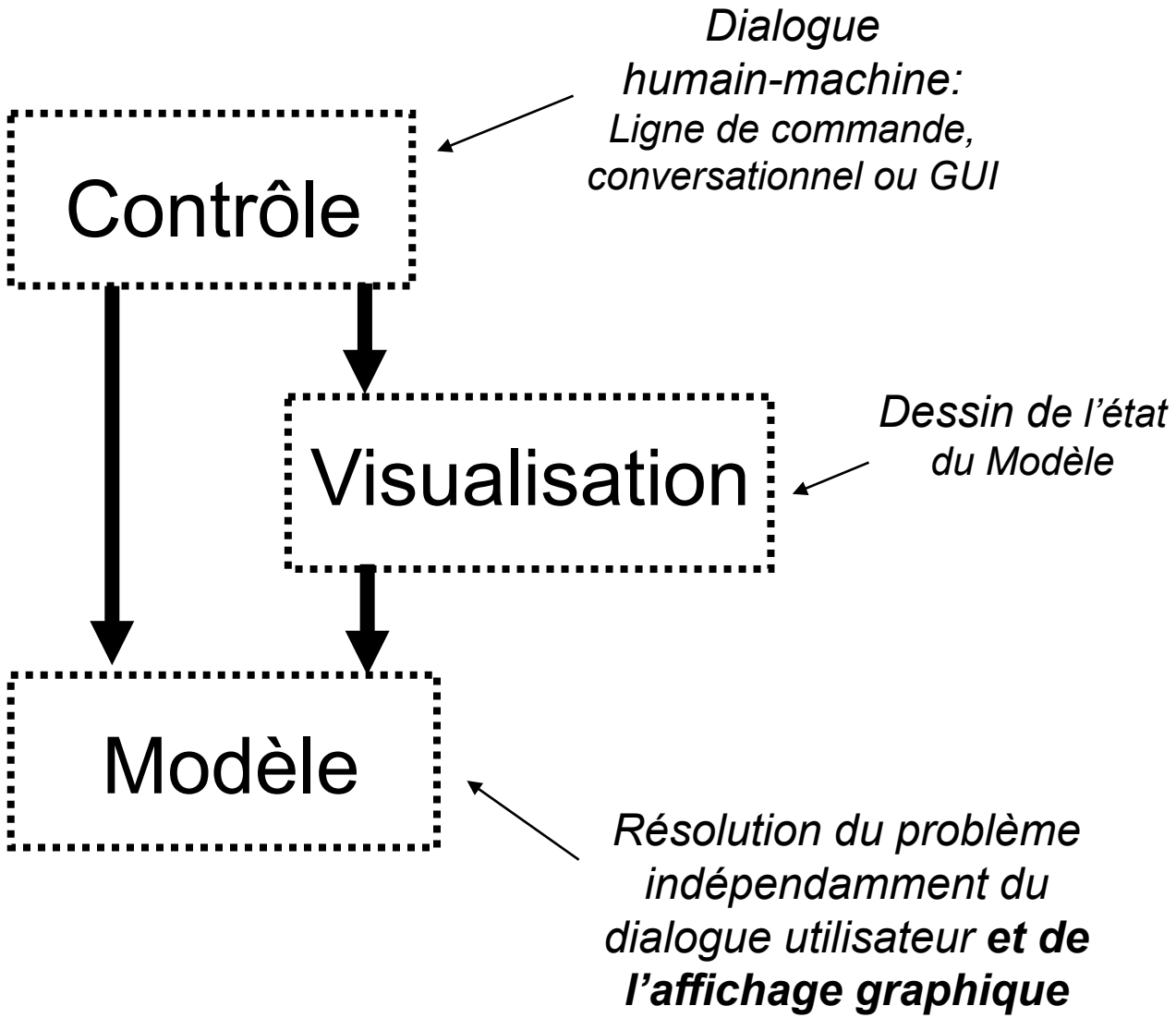
API (*Application Programming Interface*) : prototypes des fonctions exportées par une bibliothèque dans un fichier en-tête (exemple: **gtkmm.h**) = fichier d'interface de la bibliothèque

L'architecture MVC: Model-View-Controller (version stricte)

Principe : Separation of concerns

Objectifs:

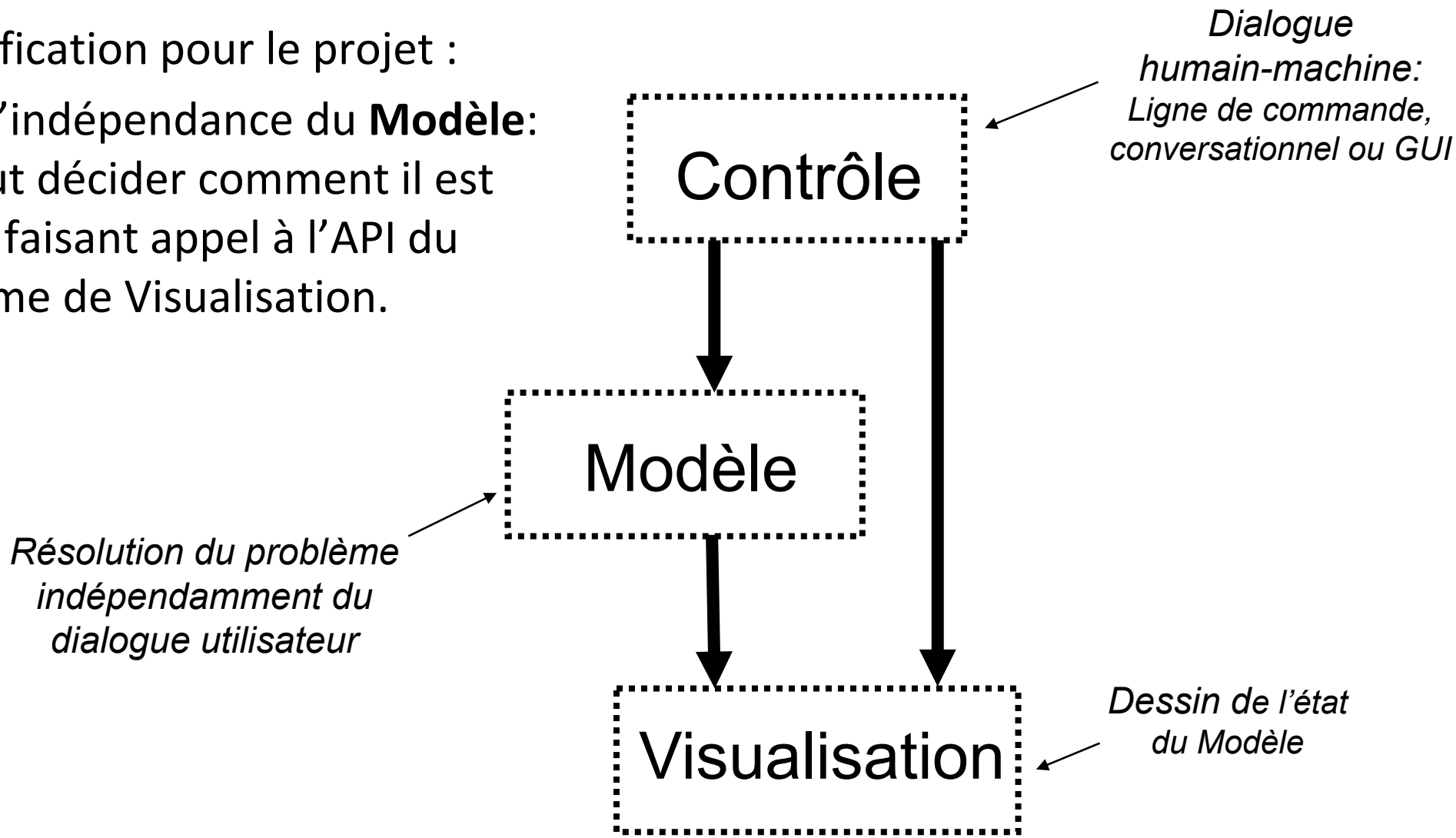
- Permettre plusieurs styles de Visualisations et de Contrôle/Dialogue sur un Modèle.
- S'adapter facilement à l'évolution rapide de la technologie des parties "Visualisation" et "Contrôle utilisateur" en gardant stable la partie "Modèle métier"
- Dans l'industrie, les compétences sont souvent focalisées sur un sous-système, rarement sur l'ensemble des trois.



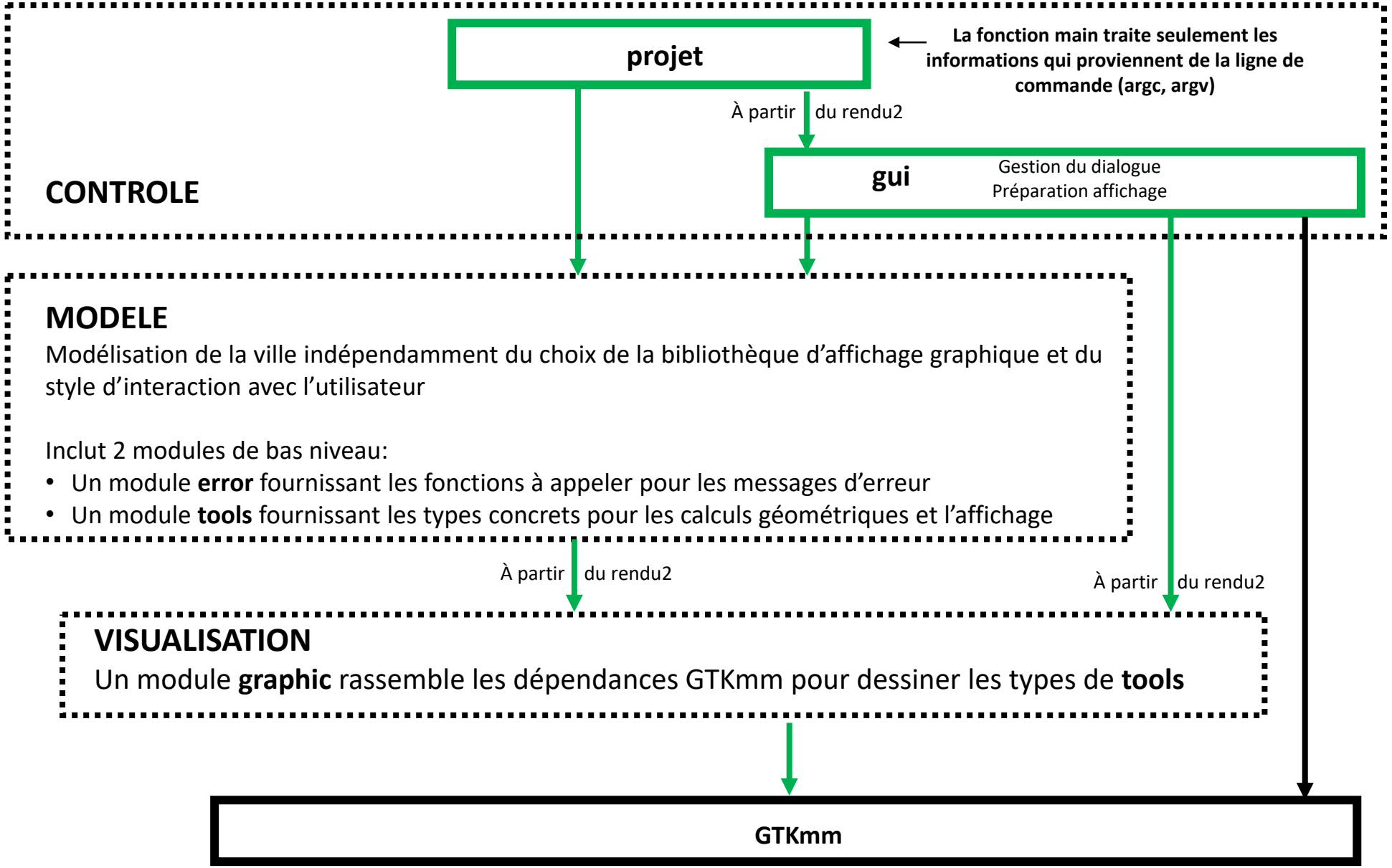
L'architecture MVC: Model-View-Controller

Simplification pour le projet :

On relaxe l'indépendance du **Modèle**: celui-ci peut décider comment il est dessiné en faisant appel à l'API du sous-système de Visualisation.



L'architecture MVC: Model-View-Controller (Fig 10 projet)



Résumé

- **En vertu du principe de séparation des fonctionalités nous adoptons l'approche Model-View-Controller pour l'architecture d'une application interactive.**
- **Le sous-système de Contrôle pilote celui de Visualisation et celui du Modèle du problème traité.**
- **GTKmm offre une hiérarchie de classes C++ pour construire une interface utilisateur**