Mathieu CHOURREAU - Léo FERME - Nathan PIEL
Université de Pau et des Pays Université de Pau et des Pays de l'Adour - IUT DE BAYONNE - PAYS-BASQUE année 2022-2023
Département Informatique saé 2.01

Saé 2.01 – Développement d'une application

Lecteur de diaporamas – Dossier d'Analyse et conception

SOMMAIRE :

1. Compléments de spécifications externes.

Rien à signaler dans cette partie.

2. Scénarios

Scénario nominal :

Dans le scénario nominal, on considère que l'utilisateur va faire toutes les actions possibles du lecteur diaporama. Il va donc charger le diaporama et sélectionner la catégorie qu'il veut et appuyer sur tous les boutons qu'il y a dans l'application donc "suivant", "reculer". L'utilisateur va ensuite appuyer sur des boutons dans la barre des menus pour finir par "quitter" qui fermera l'application.

Cas d'utilisation	**Scénario principale**	
Acteur primaire	Utilisateur	
Système	**Système du lecteur de diaporama**	
Acteur secondaire		
Niveau	**A DÉFINIR EN FONCTION DES DIAGRAMMES**	
Préconditions	L'utilisateur a lancé l'application	
Opérations	**Acteur**	**Système**
1		Le système affiche la fenêtre principale
2	L'utilisateur choisit l'onglet "Paramètre"	
3		Le système affiche le menu déroulant correspondant à l'action demandée
4	L'utilisateur choisit de charger un diaporama	
5		Le système charge un diaporama et affiche la première image associé à celui-ci
6	L'utilisateur choisit les catégories qu'il veut voir dans le diaporama	
7		Le système trie dans le diaporama les images en fonction des catégories choisi par l'utilisateur

8	L'utilisateur choisit d'avancer le diaporama	
9		Le système affiche l'image suivante
10	L'utilisateur choisit de reculer le diaporama	
11		Le système affiche l'image qui précède l'image actuelle
12	L'utilisateur choisit l'onglet "Aide"	
13		Le système affiche le menu déroulant correspondant à l'action demandée
14	L'utilisateur choisit l'onglet "A propos de"	
15		Le système affiche une nouvelle fenêtre avec les informations des créateurs et de l'application utilisée
16	L'utilisateur demande de fermer cette fenêtre en cliquant sur la croix	
17		Le système affiche la fenêtre principale
18	L'utilisateur choisit l'onglet "Fichier"	
19		Le système affiche le menu déroulant correspondant à l'action demandée
20	L'utilisateur choisit "Quitter"	
21		Le système ferme l'application

Scénario alternatif :

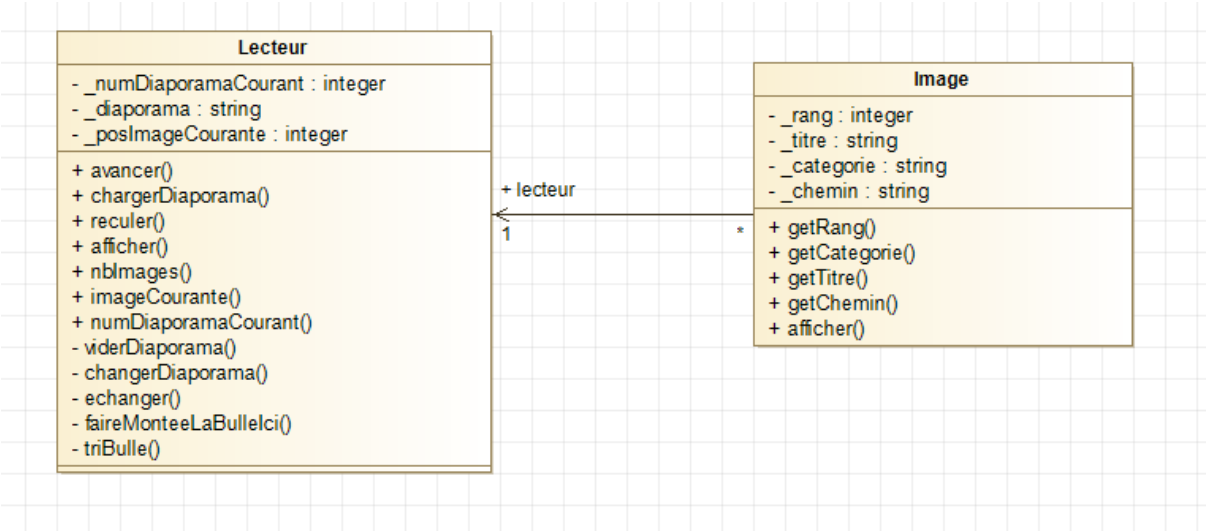
L'utilisateur va charger un diaporama mais va activer le mode automatique, le diaporama va donc défiler automatiquement jusqu'au dernier élément de celui-ci. L'utilisateur va ensuite fermer le diaporama.

Cas d'utilisation	**Scénario principale**
Acteur primaire	Utilisateur
Système	**Système du lecteur de diaporama**
Acteur secondaire	
Niveau	**A DÉFINIR EN FONCTION DES DIAGRAMMES**

Préconditions	L'utilisateur a lancé l'application	
Opérations	**Acteur**	**Système**
1		Le système affiche la fenêtre principale
2	L'utilisateur choisit l'onglet "Paramètre"	
3		Le système affiche le menu déroulant correspondant à l'action demandée
4	L'utilisateur choisit de charger un diaporama	
5		Le système charge un diaporama et affiche la première image associé à celui-ci
6	L'utilisateur choisit les catégories qu'il veut voir dans le diaporama	
7		Le système trie dans le diaporama les images en fonction des catégories choisi par l'utilisateur
8	L'utilisateur choisit de lancer le mode automatique	
9		Le système passe à l'image suivante et l'affiche
10		Une fois la dernière image atteinte, le mode automatique s'arrête
11	L'utilisateur choisit l'onglet "Fichier"	
12		Le système affiche le menu déroulant correspondant à l'action demandée
13	L'utilisateur choisit "Quitter"	
14		Le système ferme l'application

3. Diagramme de classe (UML)qt

a. Le diagramme de classes UML se focalise sur les classes **métier**, cad celles décrivant les éléments structurants de l'application, indépendamment des éléments d'interface.



A noter que dans ce diagramme de classe, _diaporama est de type Diaporama et non de type string.

b. Dictionnaire des éléments pour chaque classe

Classe Lecteur :

```
#ifndef LECTEUR_H
#define LECTEUR_H
#include "image.h"
#include "database.h"
#include <vector>
#include <QObject>
#include <QDebug>



typedef vector<Image*> Diaporama;   // Structure de données contenant les infos sur les images

class LecteurVue;
class Lecteur : public QObject
{
    Q_OBJECT

public:
    //getters et setters des attributs privés de LecteurVue
    LecteurVue* getLecteurVue();
    void setLecteurVue(LecteurVue* lv);

    //getters et setters des attributs privés de Image
    Image* getImage();
    void setImage(Image* i);

public:
    Lecteur();
    void avancer();             // incrémente _posImageCourante, modulo nbImages()
    void reculer();             // décrémente _posImageCourante, modulo nbImages()
    void afficher();            // affiche les informations sur lecteur-diaporama et image courante
    unsigned int nbImages();    // affiche la taille de _diaporama
    Image* imageCourante();     // retourne le pointeur vers l'image courante
    unsigned int numDiaporamaCourant();
    void viderDiaporama();      // vide _diaporama de tous ses objets image et les delete
    void changerDiaporama(unsigned int pNumDiaporama);    // permet de choisir un diaporama, 0 si aucun diaporama souhaité

private:
    unsigned int _numDiaporamaCourant;   // numéro du diaporama courant, par défaut 0
    Diaporama _diaporama;            // pointeurs vers les images du diaporama
    unsigned int _posImageCourante;  /* position, dans le diaporama,
                                        de l'image courante.
                                        Indéfini quand diaporama vide.
                                        Démarre à 0 quand diaporama non vide */
private:

    void chargerDiaporama();    // charge dans _diaporama les images du _numDiaporamaCourant
    void echanger(Image*, Image*);
    void faireMonteeLaBulleIci(Diaporama, unsigned short int, short unsigned int);
    void triBulle (Diaporama);

    Image *_image;   // pointeur sur le modèle
    LecteurVue *_leLecteurVue;  // pointeur sur la vue
    Database *db;

};

#endif // LECTEUR_H
```

Classe Image :

```cpp
#ifndef IMAGE_H
#define IMAGE_H
#include <iostream>
#include <QWidget>
#include <QObject>

using namespace std;

class Image
{
public:
    Image(unsigned int pRang=0, string pCategorie="", string pTitre="", string pChemin = "");
    unsigned int getRang();
    string getCategorie();
    string getTitre();
    string getChemin();
    void afficher();           // affiche tous les champs de l'image

private:
    unsigned int _rang;         /* rang de l'image au sein du diaporama
                                   auquel l'image est associée */
    string _titre;              // intitulé de l'image
    string _categorie;          // catégorie de l'image (personne, animal, objet)
    string _chemin;             // chemin complet vers le dossier où se trouve l'image
};

#endif // IMAGE_H
```

Classe LecteurVue :

```cpp
#ifndef LECTEURVUE_H
#define LECTEURVUE_H
#include "image.h"
#include "lecteur.h"
#include <QMainWindow>
#include <QString>
#include <QTimer>
#include <QLabel>
#include "dialogvitesse.h"

QT_BEGIN_NAMESPACE
namespace Ui { class LecteurVue; }
QT_END_NAMESPACE
class Lecteur;
class LecteurVue : public QMainWindow
{
    Q_OBJECT

public:
    LecteurVue(QWidget *parent = nullptr);
    ~LecteurVue();
    void mettreAJourVue();

    //getters et setters des attributs privés de Lecteur
    Lecteur* getLecteur();
    void setLecteur(Lecteur* l);

private slots:
    void lancerDiapo();               // Lance un timer qui appelle la méthode demandeAvancerDiapo() à intervalles régulières
    void arreterDiapo();              // Arrete le timer
    void demandeAvancerDiapo();       // Appel la méthode avancer() de Lecteur
    void demandeReculerDiapo();       // Appel la méthode reculer() de Lecteur
    void aProposDe();                 // Affiche un message qui donne des informations sur l'application
    void quitter();                   // Ferme la fenêtre
    void demandeChargerDiaporama();   // Appel la méthode chargerDiaporama() de Lecteur
    void demandeEnleverDiaporama();   // Appel la méthode viderDiaporama() de Lecteur
    void demandeChangerVitesseDiapo();          // Modifie la vitesse de défilement des images en changeant l'interevalle du timer
    void changerCategorie();          // Trie les images du diaporama en fonction des catégories choisies par l'utilisateur


private:
    QLabel *lBarreEtat;
    double vitesseDiapo;
    Ui::LecteurVue *ui;
    Lecteur *_leLecteur;
    QTimer *timer;
    DialogVitesse* maDlg;
};
#endif // LECTEURVUE_H
```

Classe DialogVitesse :

```h
#ifndef DIALOGVITESSE_H
#define DIALOGVITESSE_H
#include <QRadioButton>
#include <QDialog>

namespace Ui {
class DialogVitesse;
}

class DialogVitesse : public QDialog
{
    Q_OBJECT

public:
    explicit DialogVitesse(QWidget *parent = nullptr);
    ~DialogVitesse();
    double vitesseDiapo();                  // Retourne la vitesse de défilemenent voulue par l'utilisateur

private slots:
    void changerVitesseDiapoX0_5();         // Modifie la vitesse de défilement du diapo à 4 secondes pour défiler en X0.5
    void changerVitesseDiapoX1();           // Modifie la vitesse de défilement du diapo à 2 secondes pour défiler en X1
    void changerVitesseDiapoX2();           // Modifie la vitesse de défilement du diapo à 1 secondes pour défiler en X2

private:
    double valVitesseDiapo;                 // Initialisation de la vitesse de défilemenent voulue par l'utilisateur
    Ui::DialogVitesse *ui;
};

#endif // DIALOGVITESSE_H
```

Classe DialogVitesse :

```h
#include <QSqlDatabase>
#include <QDebug>


#ifndef DATABASE_H
#define DATABASE_H

#define DATABASE_NAME "SAE1"
#define CONNECT_TYPE "QODBC"

class Database
{
public:
    Database();
    bool openDataBase();
    void closeDataBase();

private:
    QSqlDatabase mybd;
    Database *db;
};



#endif // DATABASE_H
```

Remarques concernant le schéma de classes

4. On ne s'intéresse qu'aux attributs et méthodes métier. Notamment, on ne met pas, pour l'instant, ce qui relève de l'affichage car ce sont d'autres objets du programme (widgets) qui se chargeront de l'affichage. Par contre, on n'oublie pas les méthodes getXXX(), qui permettront aux objets métier de communiquer leur valeur aux objets graphiques pour que ceux-ci s'affichent.
5. On n'a mis ni le constructeur ni le destructeur, pour alléger le schéma.
6. D'autres attributs et méthodes pourront venir ultérieurement compléter cette première vision ANALYTIQUE de l'application. Il s'agira des attributs et méthodes dits DE CONCEPTION nécessaires au développement de l'application.

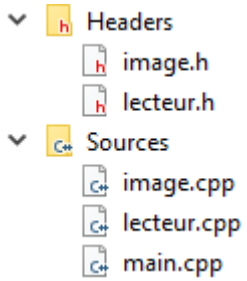
Version v0 – Version console seule

4. Implémentation et tests

4.1 Implémentation

Liste et rôle des fichiers de cette version :

lecteur.h	Spécification de la classe Lecteur
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image
image.cpp	Corps de la classe Image
main.cpp	Tester les méthodes de la classe Lecteur



4.2 Test

Test avec le programme fournit main.cpp
Valeur fournit :

```
Test avancer() : 4 fois
avancer() :
avancer() :
avancer() :
avancer() :

Test reculer() : 5 fois
reculer() :
reculer() :
reculer() :
reculer() :
reculer() :
```

Valeur obtenu :

```
Lecteur vide
Diaporama num. 1 selectionne.
4 images chargees dans le diaporama
Diaporama num. 1
image courante : image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)
```

```
Test avancer() : 4 fois
avancer() :
Diaporama num. 1
image courante : image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)

avancer() :
Diaporama num. 1
image courante : image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)

avancer() :
Diaporama num. 1
image courante : image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

avancer() :
Diaporama num. 1
image courante : image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)
```

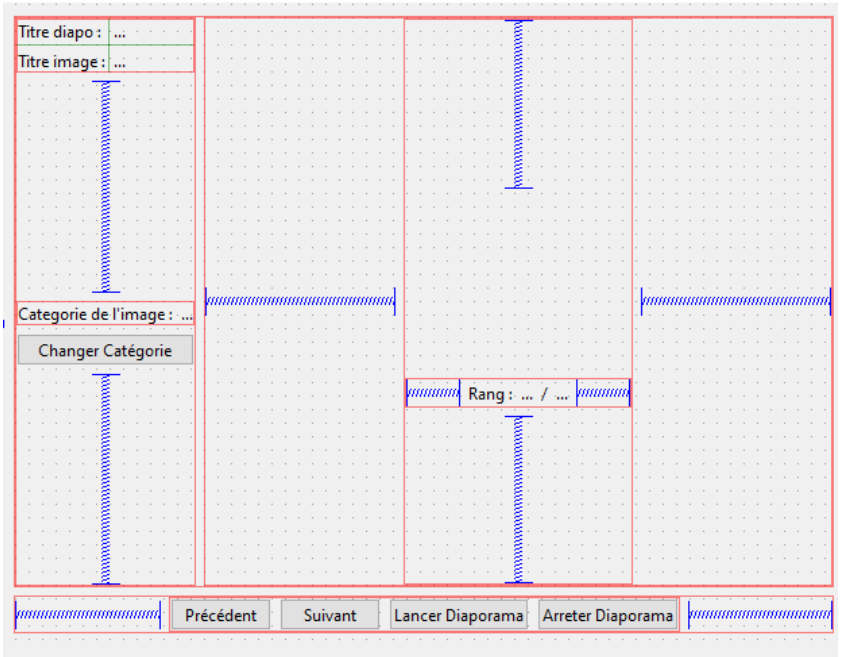
```
Test reculer() : 5 fois
reculer() :
Diaporama num. 1
image courante : image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

reculer() :
Diaporama num. 1
image courante : image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)

reculer() :
Diaporama num. 1
image courante : image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)

reculer() :
Diaporama num. 1
image courante : image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

reculer() :
Diaporama num. 1
image courante : image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)
```

Version v1 – projet Graphique seul

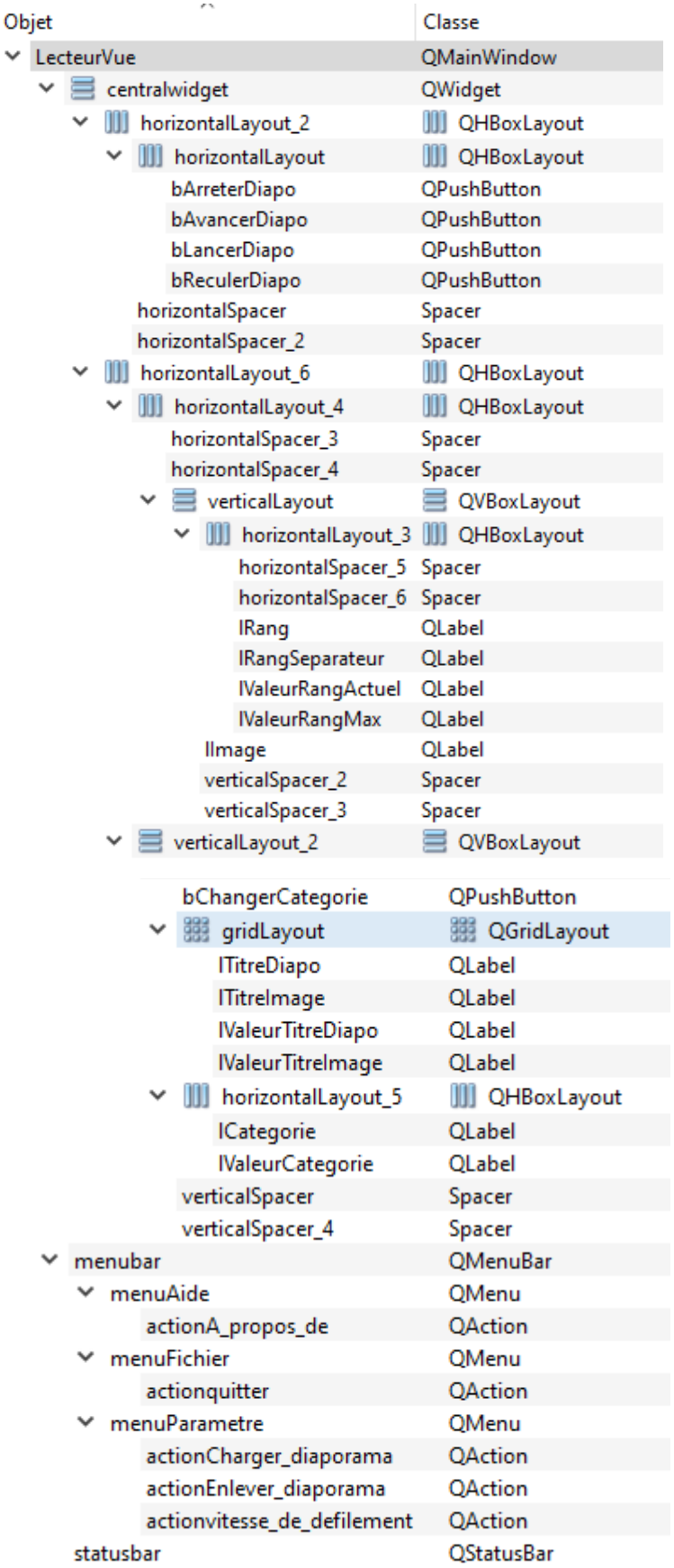
5. Éléments d'interface

Voici notre interface graphique ainsi que tous les noms des éléments présents dans l'interface.

Interface graphique



Les noms des éléments présents dans l'interface



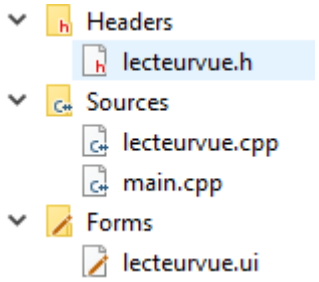
Objet	Classe
LecteurVue	QMainWindow
centralwidget	QWidget
horizontalLayout_2	QHBoxLayout
horizontalLayout	QHBoxLayout
bArreterDiapo	QPushButton
bAvancerDiapo	QPushButton
bLancerDiapo	QPushButton
bReculerDiapo	QPushButton
horizontalSpacer	Spacer
horizontalSpacer_2	Spacer
horizontalLayout_6	QHBoxLayout
horizontalLayout_4	QHBoxLayout
horizontalSpacer_3	Spacer
horizontalSpacer_4	Spacer
verticalLayout	QVBoxLayout
horizontalLayout_3	QHBoxLayout
horizontalSpacer_5	Spacer
horizontalSpacer_6	Spacer
lRang	QLabel
lRangSeparateur	QLabel
lValeurRangActuel	QLabel
lValeurRangMax	QLabel
lImage	QLabel
verticalSpacer_2	Spacer
verticalSpacer_3	Spacer
verticalLayout_2	QVBoxLayout
bChangerCategorie	QPushButton
gridLayout	QGridLayout
lTitreDiapo	QLabel
lTitreImage	QLabel
lValeurTitreDiapo	QLabel
lValeurTitreImage	QLabel
horizontalLayout_5	QHBoxLayout
lCategorie	QLabel
lValeurCategorie	QLabel
verticalSpacer	Spacer
verticalSpacer_4	Spacer
menubar	QMenuBar
menuAide	QMenu
actionA_propos_de	QAction
menuFichier	QMenu
actionquitter	QAction
menuParametre	QMenu
actionCharger_diaporama	QAction
actionEnlever_diaporama	QAction
actionvitesse_de_defilement	QAction
statusbar	QStatusBar

6. Implémentation et tests

6.1 Implémentation

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas
lecteurVue.cpp	Corps de la classe LecteurVue
lecteurvue.ui	Fichier du dessin de l'interface réalisé par QtDesigner
main.cpp	Tester les méthodes de la classe Lecteur



Remarques sur l'implémentation :

On a utilisé des connect pour chaque éléments qui peut être utilisé par l'utilisateur

```
connect(ui->bArreterDiapo,SIGNAL(clicked()),this,SLOT(ArreterDiapo()));
connect(ui->bLancerDiapo,SIGNAL(clicked()),this,SLOT(LancerDiapo()));
connect(ui->bAvancerDiapo,SIGNAL(clicked()),this,SLOT(AvancerDiapo()));
connect(ui->bReculerDiapo,SIGNAL(clicked()),this,SLOT(ReculerDiapo()));

connect(ui->actionA_propos_de,SIGNAL(clicked()),this,SLOT(AProposDe()));
connect(ui->actionquitter,SIGNAL(clicked()),this,SLOT(Quitter()));
connect(ui->actionCharger_diaporama,SIGNAL(clicked()),this,SLOT(ChargerDiaporama()));
connect(ui->actionEnlever_diaporama,SIGNAL(clicked()),this,SLOT(EnleverDiaporama()));
connect(ui->actionvitesse_de_defilement,SIGNAL(clicked()),this,SLOT(VitesseDefilement()));
connect(ui->bChangerCategorie,SIGNAL(clicked()),this,SLOT(ChangerCategorie()));
```

6.2 Test

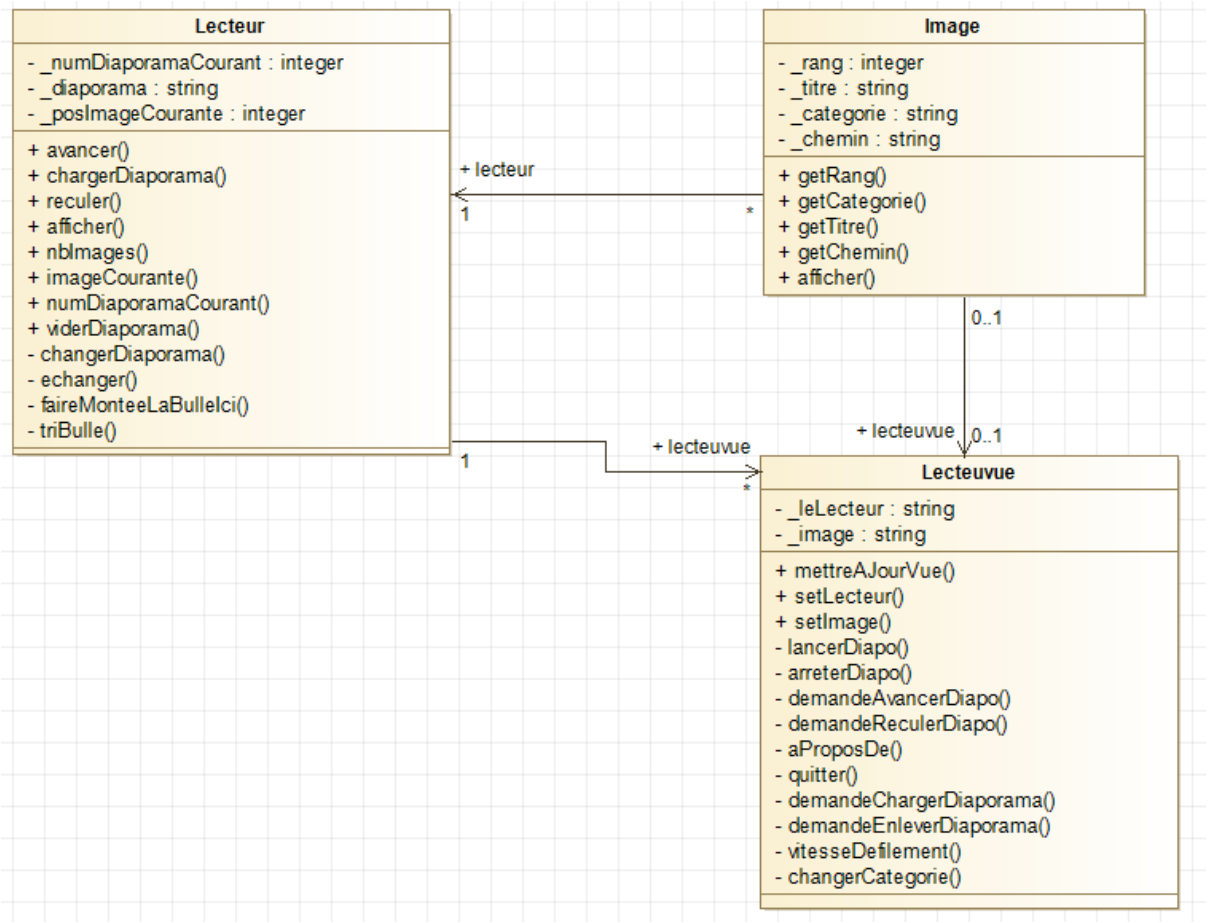
On vérifie les boutons qui marchent en affichant un message dans un qDebug quand un clique a été réalisé sur un bouton :

```
Clique sur ReculerDiapo
Clique sur AvancerDiapo
Clique sur LancerDiapo
Clique sur ArreterDiapo
Clique sur ChangerCategorie
```

Les boutons présents dans la barre de menu ne sont pas utilisables car elle n'est pas encore initialisée. L'interface ne change jamais car aucun des boutons ne sont fonctionnel, tout est affiché dans le shell.

Version v2 –

7. Diagramme de classes (UML)



A noter que dans ce schéma, certains types ne sont pas justes. Dans la classe Lecteur _diaporama est de type Diaporama. Dans la classe lecteurVue, _leLecteur est de type Lecteur et _image est de type Image.

8. Comportement de l'application

1. *Diagramme états-transitions-actions du lecteur de diaporamas (v2)*

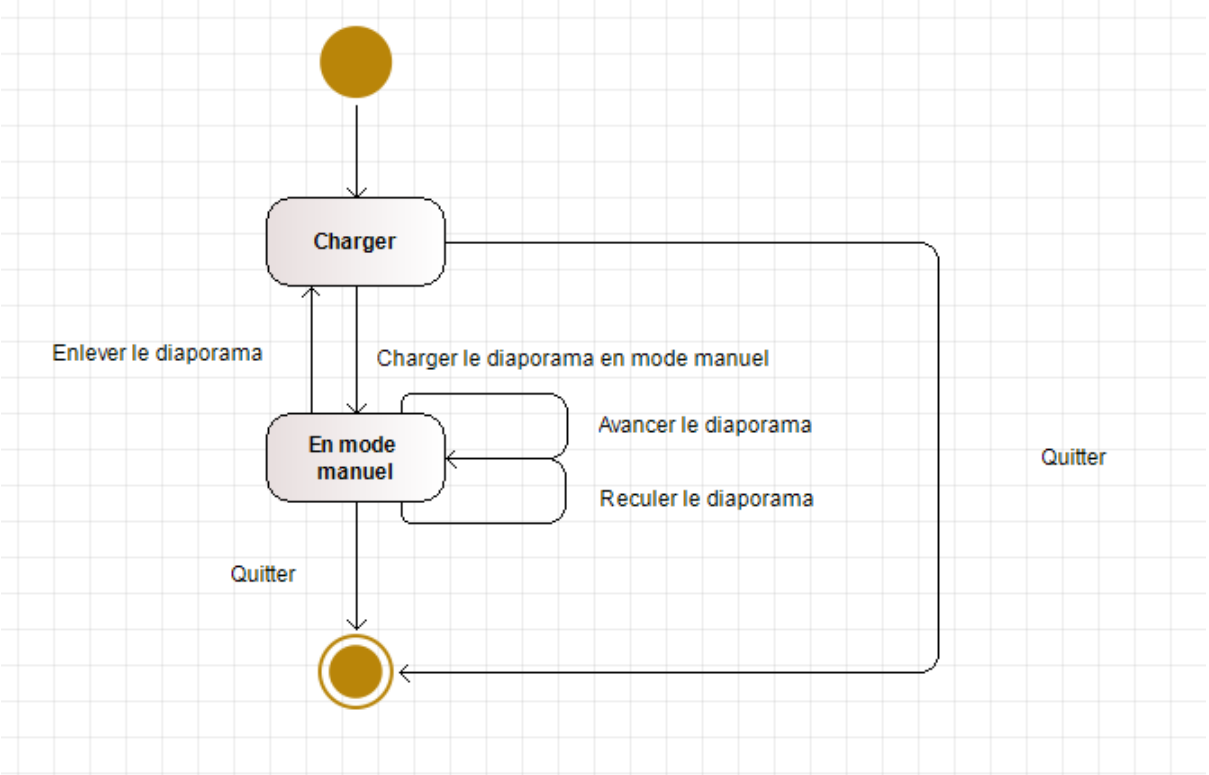


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v2

2. *Dictionnaire des états, événements et Actions (v2)*

Dictionnaire des états du diaporama

nomEtat	*Signification*
en mode manuel	Le diaporama est chargé et est par défaut en mode manuel. Les boutons accessibles sont : SUIVANT, PRÉCÉDENT, LANCER DIAP ET ARRÊTER DIAPO.

Tableau 2 : États du lecteur de diaporamas – v2

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
charger le diaporama en mode manuel	Se déclenche lorsque l'utilisateur clique sur "charger un diaporama". Le diaporama est par défaut en mode manuel. Toutes les images sont chargées dans le lecteur.
enlever le diaporama	Le diaporama est de nouveau vide quand cette action est réalisée lorsque l'utilisateur clique sur le bouton "enlever le diaporama". Les images sont enlevées du diaporama.

Tableau 3 : Événements faisant changer le diaporama d'état – v2

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
charger le diaporama en mode manuel	Se déclenche lorsque l'utilisateur clique sur "charger un diaporama". Le diaporama est par défaut en mode manuel. Toutes les images sont chargées dans le lecteur.
avancer le diaporama	Le lecteur est en mode manuel. Il décide d'avancer à l'image suivante en cliquant sur le bouton 'SUIVANT".
reculer le diaporama	Le lecteur est en mode manuel. Il décide de reculer d'une image en cliquant sur le bouton "PRECEDENT".
enlever le diaporama	Le diaporama est de nouveau vide quand cette action est réalisée lorsque l'utilisateur clique sur le bouton "enlever le diaporama". Les images sont enlevées du diaporama.
quitter le lecteur	Arrêt du lecteur (bouton correspondant "quitter").

Tableau 4 : Actions à réaliser lors des changements d'état – lecteur de diaporamas v2

3. ***Table T_EtatsEvenementsActions (v2)***

Correspondance matricielle du diagramme états-transitions de l'application :

- en *ligne : les **états*** du lecteur de diaporamas (éventuel état de départ d'une transition)
- en *colonne : les **événements*** faisant changer le lecteur d'état (déclencheur d'une transition)
- dans chaque cellule : l'état d'arrivée de la transition + action/traitement à faire + éventuellement garde accompagnant la transition

Élément graphique prenant en charge cet événement	actionCharger_diaporama	actionEnlever_diaporama
Événement -> nomEtat	charger le diaporama en mode manuel	enlever le diaporama
en mode manuel	X	

Tableau 5 : Matrice d'états-transitions du lecteur de diaporamas – v2

9. Implémentation et tests

1. *Implémentation (v2)*

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas Elle va gérer l'affichage des données et l'interaction avec l'utilisateur.
lecteurVue.cpp	Corps de la classe LecteurVue.
lecteurvue.ui	Fichier du dessin de l'interface réalisé par QtDesigner
lecteur.h	Spécification de la classe Lecteur. Ce fichier contient la spécification de la classe Lecteur.
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image Ce fichier contient la spécification de la classe Image.
image.cpp	Corps de la classe Image
main.cpp	Il contient le code principal pour initialiser l'application, créer des objets de classe, configurer l'interface utilisateur, et démarrer la boucle d'événements Qt.

Remarques sur l'implémentation :

Les signals ainsi que les slots ont été un peu modifié :

```
connect(ui->bArreterDiapo,SIGNAL(clicked()),this,SLOT(arreterDiapo()));
connect(ui->bLancerDiapo,SIGNAL(clicked()),this,SLOT(lancerDiapo()));
connect(ui->bAvancerDiapo,SIGNAL(clicked()),this,SLOT(demandeAvancerDiapo()));
connect(ui->bReculerDiapo,SIGNAL(clicked()),this,SLOT(demandeReculerDiapo()));

connect(ui->actionA_propos_de,SIGNAL(triggered()),this,SLOT(aProposDe()));
connect(ui->actionquitter,SIGNAL(triggered()),this,SLOT(quitter()));
connect(ui->actionCharger_diaporama,SIGNAL(triggered()),this,SLOT(demandeChargerDiaporama()));
connect(ui->actionEnlever_diaporama,SIGNAL(triggered()),this,SLOT(demandeEnleverDiaporama()));
connect(ui->actionvitesse_de_defilement,SIGNAL(triggered()),this,SLOT(vitesseDefilement()));
```

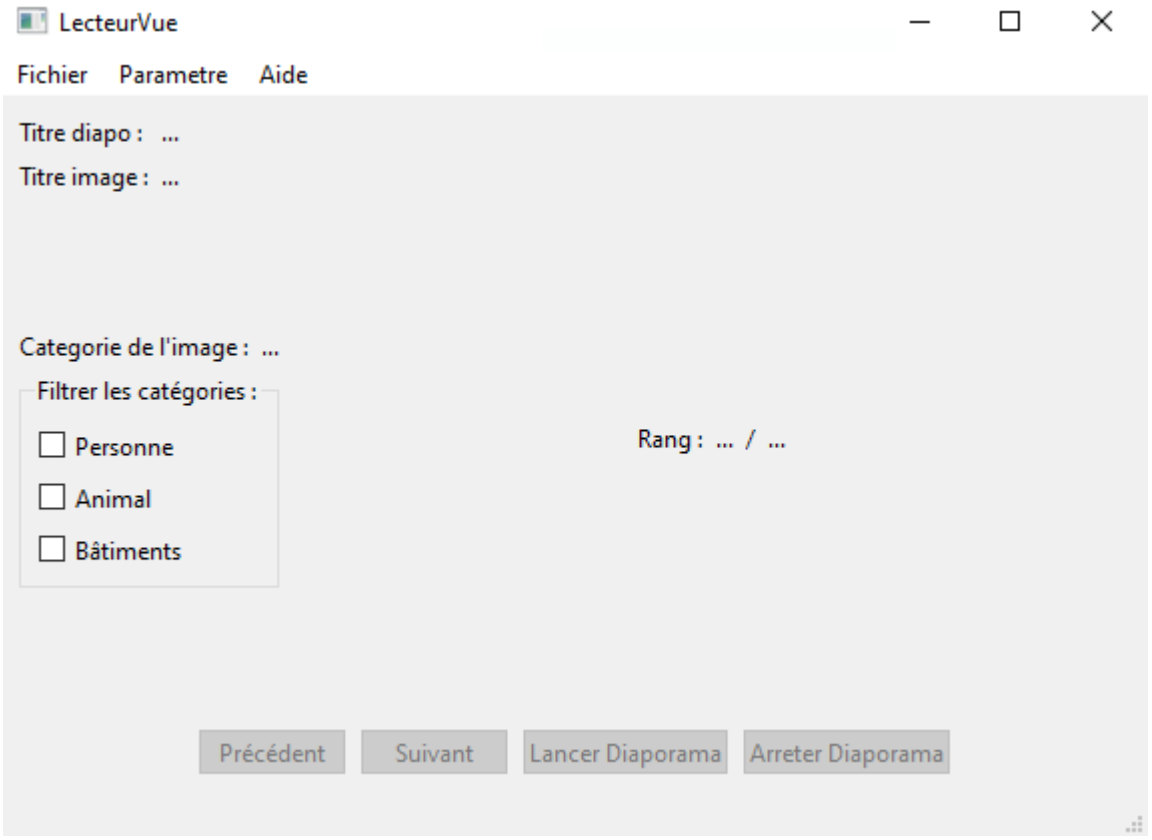
Pour les boutons situé sur la barre d'état, le signal n'est plus "clicked" mais "triggered"

2. ***Tests (v2)***

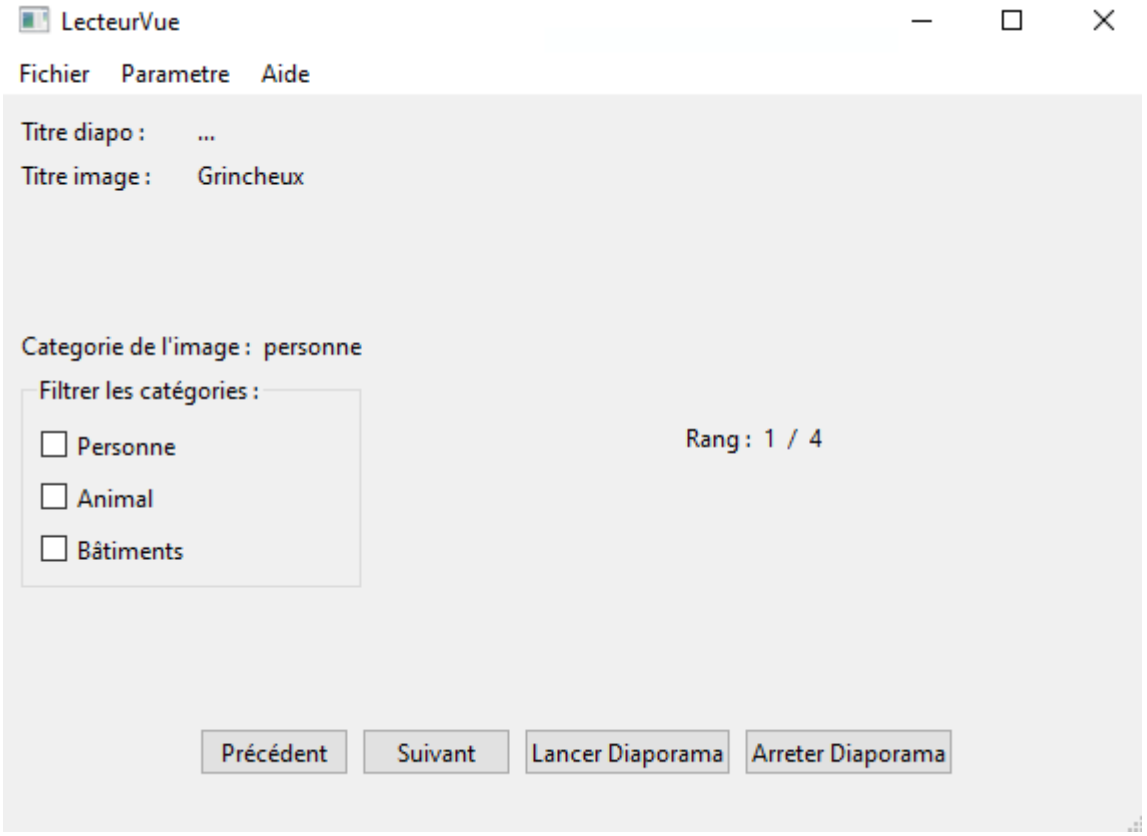
Les tests prévus : aucuns tests n'ont été prévus.

Charger le diaporama ainsi que suivant et reculer sont fonctionnel

fenêtre lors du lancement de l'application :

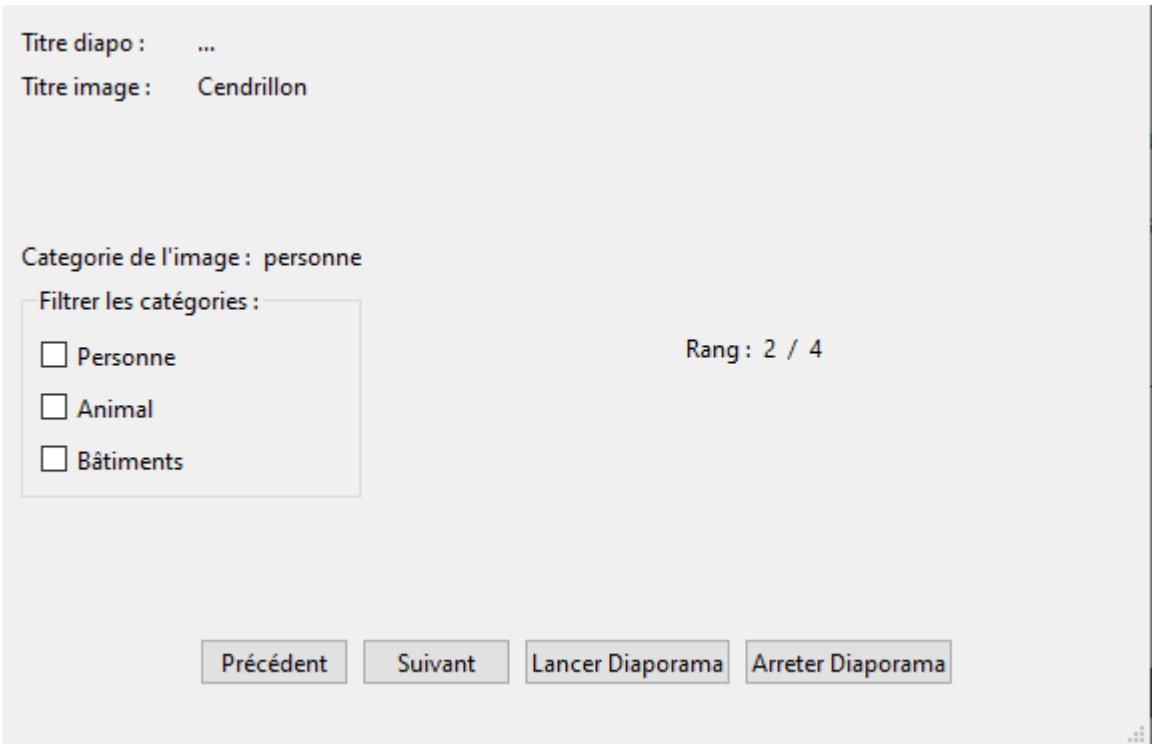


On clique sur "charger le diaporama" :

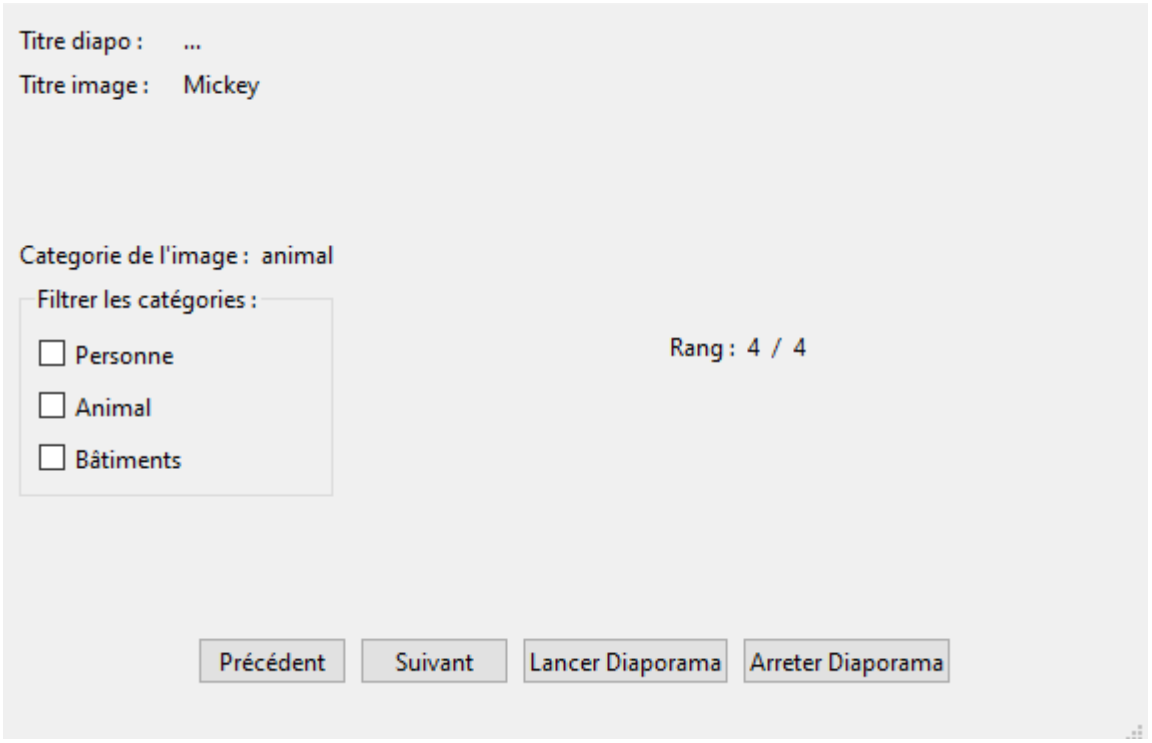


On voit bien que le diaporama est bien initialisé dans la fenêtre. Ensuite les bouton suivant et Précédent sont fonctionnel :

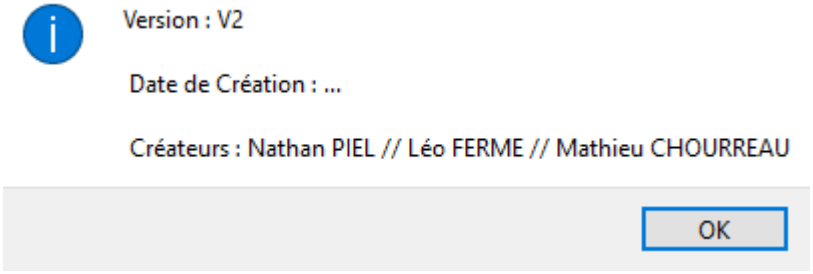
Suivant :



Précédent :



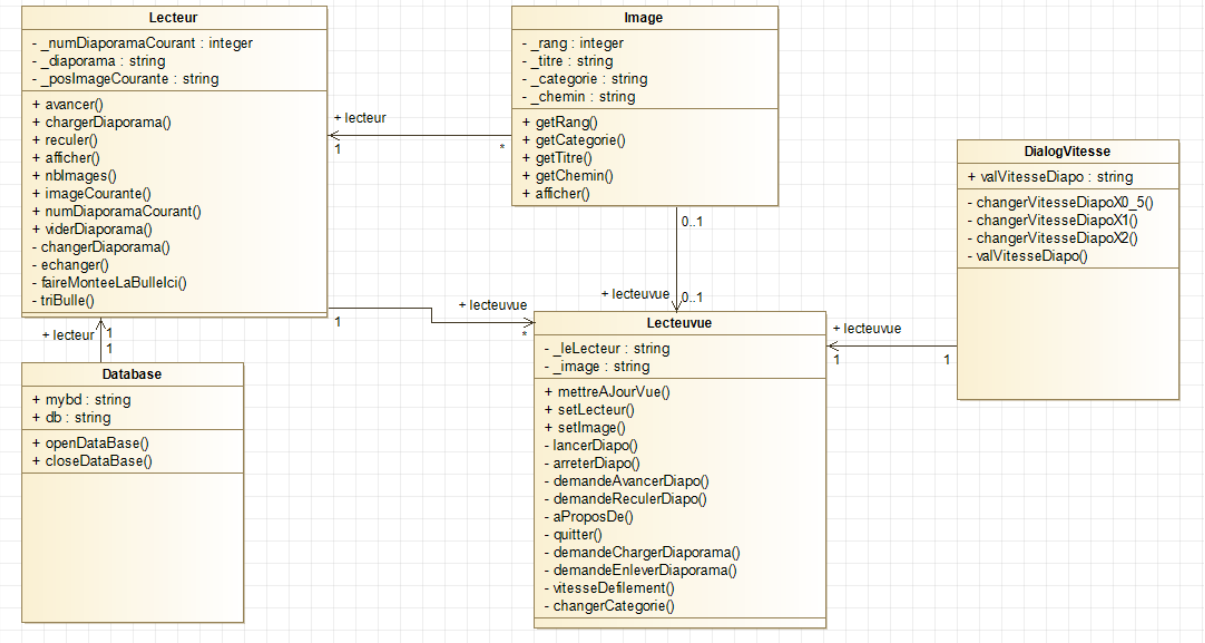
La catégorie “A propos de” est aussi fonctionnel , elle affiche ce message dans une QMessageBox :



Le bouton “quitter” marche aussi.

Version v5 –

10. Diagramme de classes (UML)



11. Comportement de l'application

1. *Diagramme états-transitions-actions du lecteur de diaporamas (v5)*

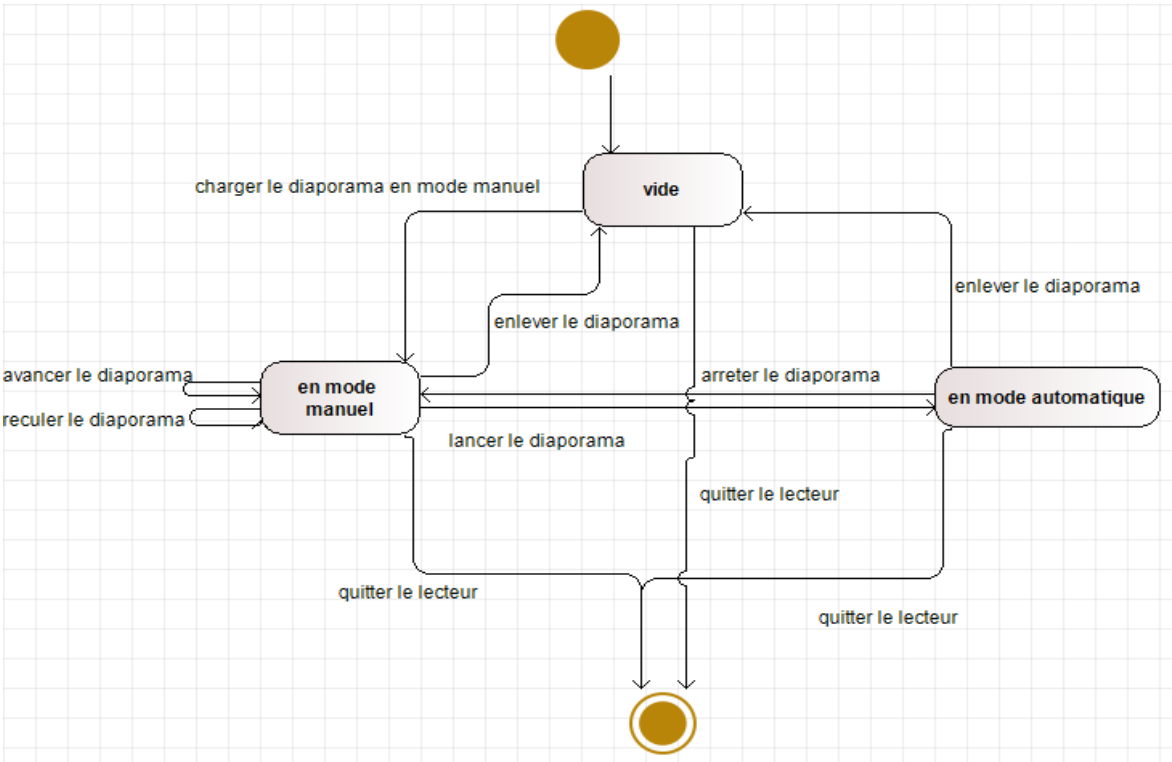


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v5

2. *Dictionnaire des états, événements et Actions (v5)*

Dictionnaire des états du diaporama

nomEtat	*Signification*
vide	Le diaporama ne contient aucune image. Aucun bouton n'est accessible.
en mode manuel	Le diaporama est chargé et est par défaut en mode manuel. Les boutons accessibles sont : SUIVANT, PRÉCÉDENT, LANCER DIAP ET ARRÊTER DIAPO.
en mode automatique	Le diaporama est en mode automatique quand l'utilisateur le choisit. Les boutons accessibles sont : SUIVANT, PRÉCÉDENT, LANCER DIAP ET ARRÊTER DIAPO.

Tableau 2 : États du lecteur de diaporamas – v5

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
charger le diaporama en mode manuel	Se déclenche lorsque l'utilisateur clique sur "charger un diaporama". Le diaporama est par défaut en mode manuel. Toutes les images sont chargées dans le lecteur.
enlever le diaporama	Le diaporama est de nouveau vide quand cette action est réalisée lorsque l'utilisateur clique sur le bouton "enlever le diaporama". Les images sont enlevées du diaporama.
lancer le diaporama	Fait en sorte que le diaporama passe de l'état de manuel à automatique (bouton correspondant "lancer le diaporama"). Les images défilent automatiquement les unes après les autres.

Tableau 3 : Evénements faisant changer le diaporama d'état – v5

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
charger le diaporama en mode manuel	Se déclenche lorsque l'utilisateur clique sur "charger un diaporama". Le diaporama est par défaut en mode manuel. Toutes les images sont chargées dans le lecteur.
avancer le diaporama	Le lecteur est en mode manuel. Il décide d'avancer à l'image suivante en cliquant sur le bouton 'SUIVANT".
reculer le diaporama	Le lecteur est en mode manuel. Il décide de reculer d'une image en cliquant sur le bouton "PRECEDENT".
enlever le diaporama	Le diaporama est de nouveau vide quand cette action est réalisée lorsque l'utilisateur clique sur le bouton "enlever le diaporama". Les images sont enlevées du diaporama.

lancer le diaporama	Fait en sorte que le diaporama passe de l'état de manuel à automatique (bouton correspondant "lancer le diaporama"). Les images défilent automatiquement les unes après les autres.
arrêter le diaporama	Fait en sorte que le diaporama passe de l'état d'automatique à manuel (bouton correspondant 'arrêter le diaporama'). Les images ne défilent plus toutes seules.
quitter le lecteur	Arrêt du lecteur (bouton correspondant "quitter").

Tableau 4 : Actions à réaliser lors des changements d'état – lecteur de diaporamas v5

3. *Table T_EtatsEvenementsActions (v5)*

Correspondance matricielle du diagramme états-transitions de l'application :

- en *ligne : les **états*** du lecteur de diaporamas (éventuel état de départ d'une transition)
- en *colonne : les **événements*** faisant changer le lecteur d'état (déclencheur d'une transition)
- dans chaque cellule : l'état d'arrivée de la transition + action/traitement à faire + éventuellement garde accompagnant la transition

Élément graphique prenant en charge cet événement	actionCharger_diaporama	actionEnlever_diaporama	bLancerDiapo
Événement -> nomEtat	charger le diaporama en mode manuel	enlever le diaporama	lancer le diaporama
vide		X	
en mode manuel	X		
en mode automatique			X

Tableau 5 : Matrice d'états-transitions du lecteur de diaporamas – v5

12. Implémentation et tests

1. *Implémentation (v5)*

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas Elle va gérer l'affichage des données et l'interaction avec l'utilisateur.
lecteurVue.cpp	Corps de la classe LecteurVue
lecteurvue.ui	Fichier du dessin de l'interface réalisé par QtDesigner
lecteur.h	Spécification de la classe Lecteur Ce fichier contient la spécification de la classe Lecteur.
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image Ce fichier contient la spécification de la classe Image.
image.cpp	Corps de la classe Image
main.cpp	Il contient le code principal pour initialiser l'application, créer des objets de classe, configurer l'interface utilisateur, et démarrer la boucle d'événements Qt.

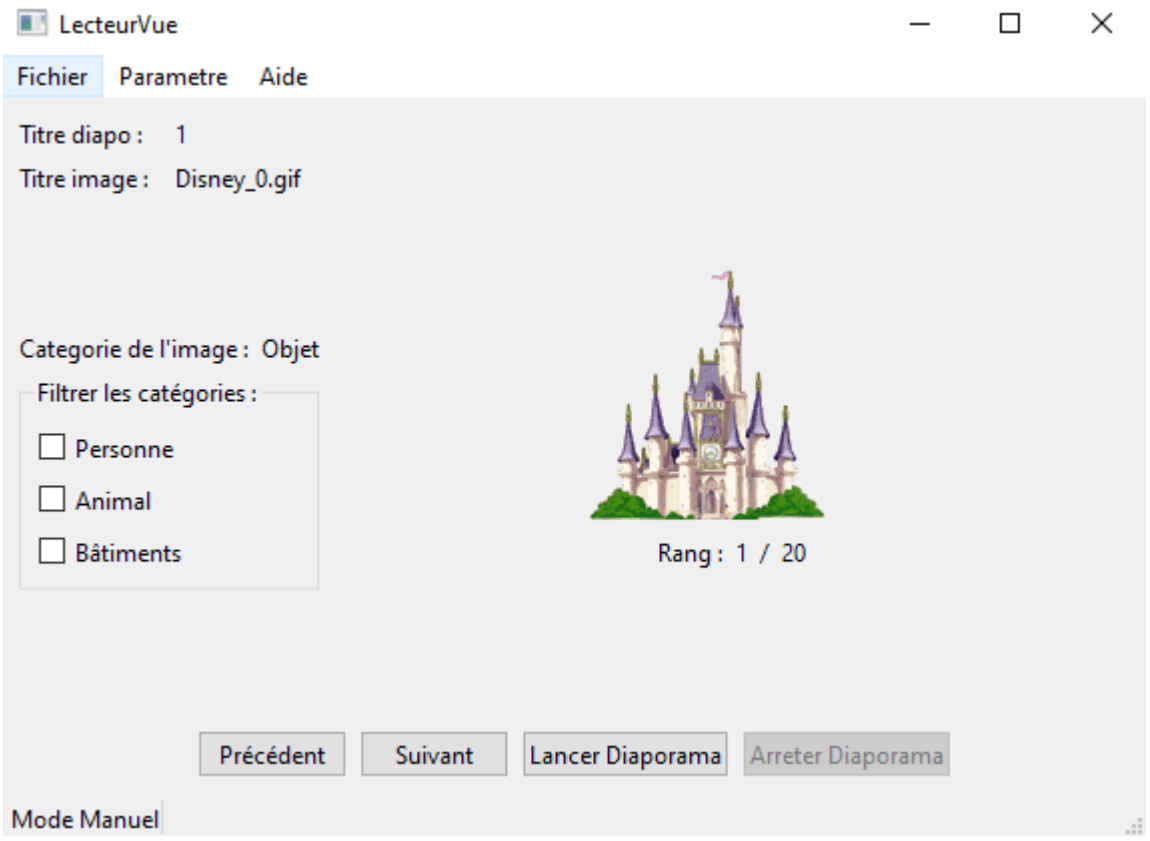
Remarques sur l'implémentation :

```
connect(ui->bArreterDiapo,SIGNAL(clicked()),this,SLOT(arreterDiapo()));
connect(ui->bLancerDiapo,SIGNAL(clicked()),this,SLOT(lancerDiapo()));
connect(ui->bAvancerDiapo,SIGNAL(clicked()),this,SLOT(demandeAvancerDiapo()));
connect(ui->bReculerDiapo,SIGNAL(clicked()),this,SLOT(demandeReculerDiapo()));

connect(ui->actionA_propos_de,SIGNAL(triggered()),this,SLOT(aProposDe()));
connect(ui->actionquitter,SIGNAL(triggered()),this,SLOT(quitter()));
connect(ui->actionCharger_diaporama,SIGNAL(triggered()),this,SLOT(demandeChargerDiaporama()));
connect(ui->actionEnlever_diaporama,SIGNAL(triggered()),this,SLOT(demandeEnleverDiaporama()));
connect(ui->actionVitesse_de_defilement,SIGNAL(triggered()),this,SLOT(demandeChangerVitesseDiapo()));
```

2. ***Tests (v5)***

Les tests prévus : aucuns tests n'ont été prévus.

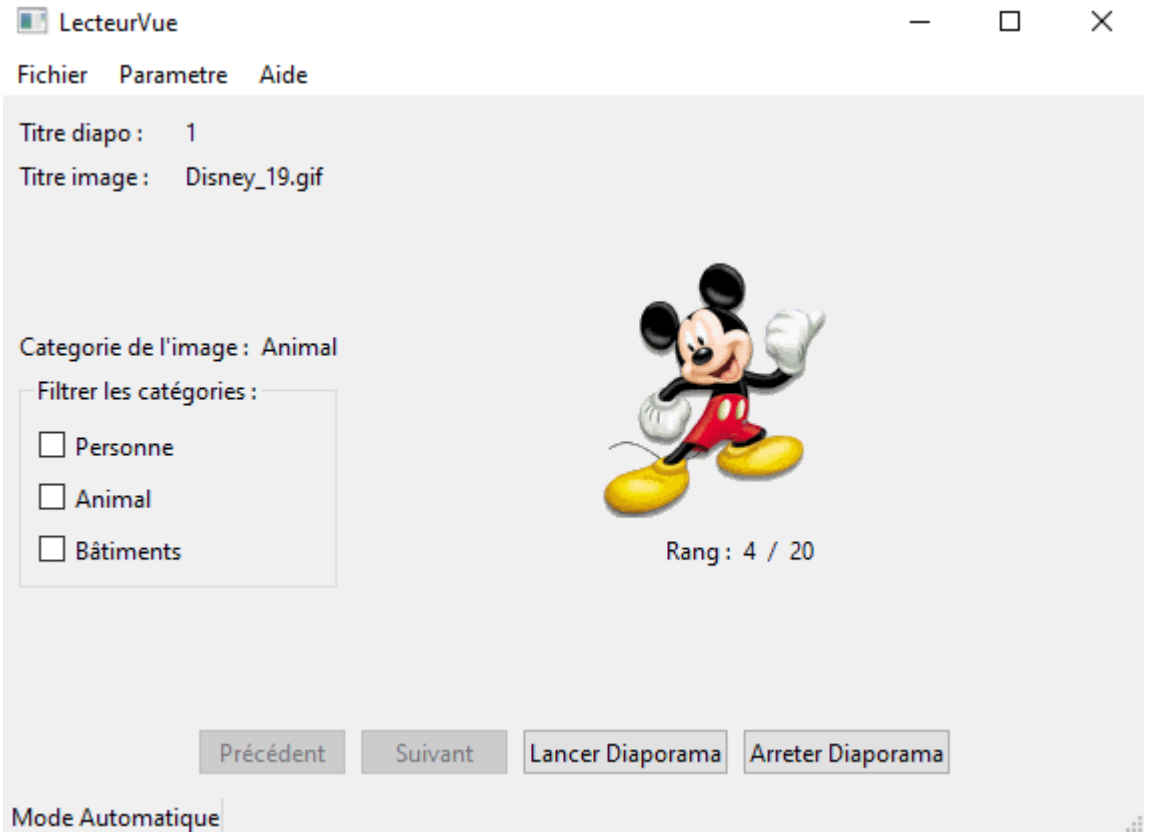
Mode manuel(charger diaporama) :



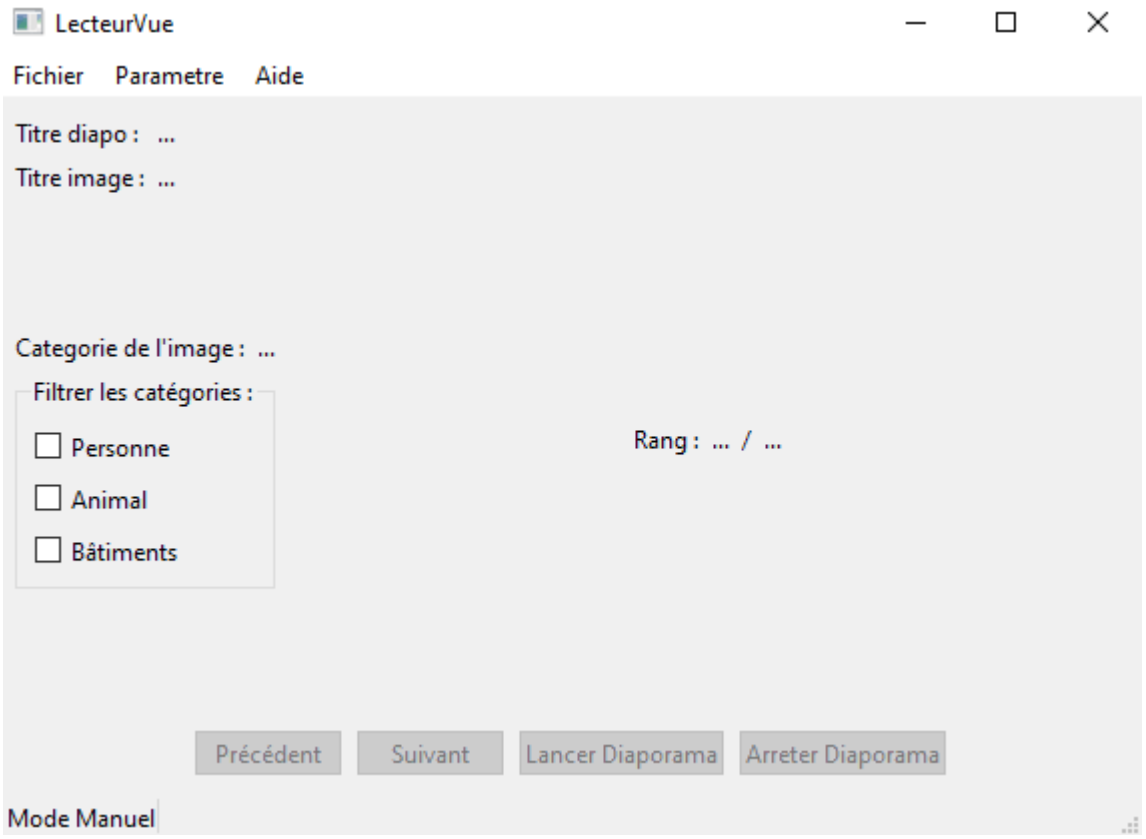
Mode manuel(suivant) :

Mode manuel(Precedent) :

Mode automatique (lancer Diaporama) :



Mode manuel(enlever diaporama) :



13. Bilan

Dépôt Git où trouver le projet complet (les versions réalisées) :

https://github.com/mathieu-Chourreau/projet_2.01

Temps global de travail (pour le groupe) :

Le temps global pour le groupe est de : 30h.

Apprentissages majeurs :

Les apprentissages majeurs ont été la mise en pratique des éléments appris lors des cours théoriques, des travaux dirigés (TD) et des travaux pratiques (TP). Les principaux éléments abordés étaient la manipulation de la classe Dialogue, l'application du modèle MVP, ainsi que l'utilisation des bases de données liées à QT.

Difficultés majeures :

Utilisation de la base de données :

- Effectuer une requête SQL fonctionnelle.
- Exploiter les valeurs retournées par la requête (dans le Qtimer)
- Filtrer le diaporama (qui est à l'heure actuelle la dernière fonctionnalité qu'il nous manque)

Points positifs / négatifs de l'activité :

Points positifs :

- Permet de mettre en pratique les éléments appris lors des cours théoriques.
- Travail en groupe.
- Permet de pratiquer le versionning sur GitHub.

Points négatifs :

- Les consignes sont parfois ambigu et les groupes reçoivent parfois différentes consignes lorsque l'enseignant vient les aider