

## Saé 2.01 – Développement d'une application

### Lecteur de diaporamas – Dossier d'Analyse et conception

Lien Github: <https://github.com/CRUSSIÈRE/SAE2.1/tree/main>

CRUSSIÈRE Lucas - GRATCHEV Grigori TD3/TP5

---

1. Compléments de spécifications externes.	2
2. Scénarios	2
3. Diagramme de classe (UML)	3
Version v0 – Version console seule	6
4. Implémentation et tests	6
4.1 Implémentation	6
4.2 Test	6
Version v1 – projet Graphique seul	8
5. Éléments d'interface	8
6. Implémentation et tests	10
Version v2 –	12
7. Diagramme de classes (UML)	12
8. Comportement de l'application	13
9. Implémentation et tests	16
Version v3 –	17
10. Diagramme de classes (UML)	17
11. Comportement de l'application	18
12. Implémentation et tests	22

## 1. Compléments de spécifications externes.

*Pas de spécifications externes en plus*

## 2. Scénarios

*Description du scénario nominal et de un / deux scénarios alternatifs afin de mettre en évidence les interactions entre le système et l'utilisateur*

- |   |
|---|
| <p>Lecture du diaporama<br/>Pré-conditions:<br/>- l'utilisateur a lancé l'application<br/>- Le diaporama est chargé</p> |
|---|

1. L'utilisateur clique sur le bouton "suivant"	
	2. Le lecteur de diaporama affiche l'image suivante dans le diaporama
3. L'utilisateur clique sur le bouton "précédent"	
	4. Le lecteur de diaporama affiche l'image précédente dans le diaporama
5. L'utilisateur clique sur le bouton "Démarrer le mode auto"	
	6. Le lecteur défile toutes les images automatiquement
7. L'utilisateur clique sur le bouton "Arrêter le mode auto"	
	7. Le lecteur arrête de faire défiler les images automatiquement

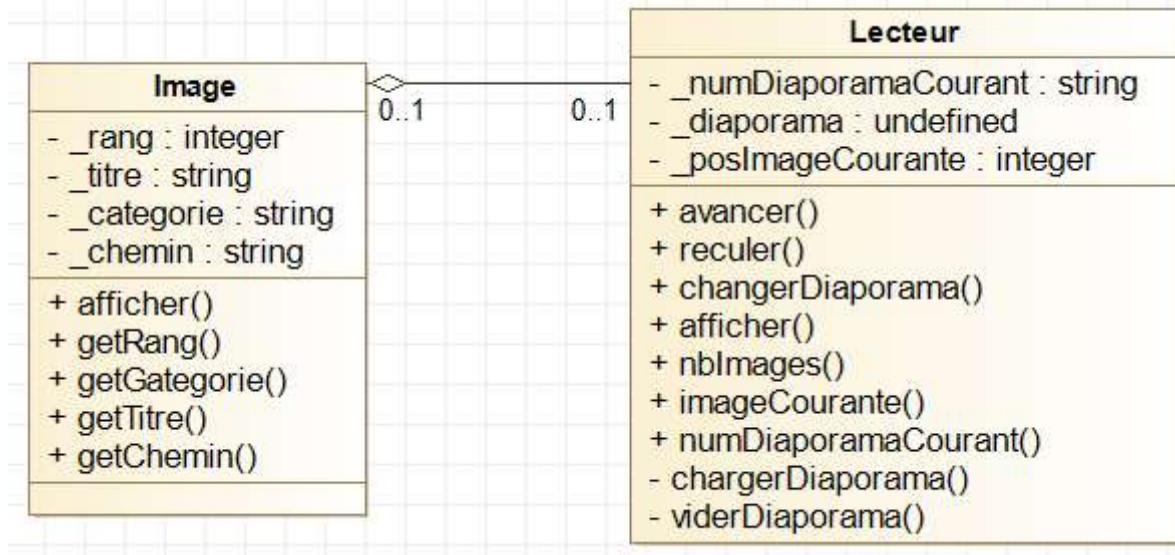
#### Scénarios alternatifs:

<p style="text-align: center;"><b>Lecture du diaporama</b></p> <p style="text-align: center;">Pré-conditions:</p> <ul style="list-style-type: none"> <li>- l'utilisateur a lancé l'application</li> <li>- Le diaporama est démarré</li> </ul>	
1. L'utilisateur clique sur le bouton suivant	
	2. Le diaporama est vide et affiche "Impossible d'avance: fin du diaporama atteinte"

<p style="text-align: center;"><b>Lecture du diaporama</b></p> <p style="text-align: center;">Pré-conditions:</p> <ul style="list-style-type: none"> <li>- l'utilisateur a lancé l'application</li> <li>- Le diaporama est sur la dernière diapositive</li> </ul>	
1. L'utilisateur clique sur le bouton suivant	
	2. Le diaporama recommence à la diapositive 1

### 3. Diagramme de classe (UML)

(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.



*Nous avons mis le type undefined pour “\_diaporama” car on ne peut pas créer de type sur modelio, le type est normalement Diaporama.*

(b) Dictionnaire des éléments pour chaque classe

Classe Image			
Nom attribut	Signification	Type	Exemple
rang	rang de l'image au sein du diaporama auquel l'image est associée	integer	1
Titre	Intitulé de l'image	string	Donald
Categorie	Catégorie de l'image	string	personne, animal, objet
Chemin	Chemin complet vers le dossier où se trouve l'image	string	\haya\dossiersetud_BaieSsd\ggratchev\Downloads\cartesDisney\Disney_2.gif

Tableau : Dictionnaire des éléments - Classe Image

Classe Lecteur			
Nom attribut	Signification	Type	Exemple
numDiaporamaCourant	Numéro du diaporama courant, par défaut 0	integer	1
Diaporama	Pointeurs vers les images du diaporama	vector	image( rang:2, titre:Cendrillon, categorie:personne, chemin:\haya\dossiersetud_BaieSsd\ggratchev\Downloads\cartesDisney\Disney_2.gif)
posImageCourante	position, dans le diaporama, l'image courante. Indéfini quand diaporama vide.	integer	0

	Démarre à 0 quand diaporama non vide	
--	--------------------------------------	--

Tableau : Dictionnaire des éléments - Classe Lecteur

(c) Dictionnaire des méthodes : vous pouvez fournir directement le fichier entête de chaque classe.

Exemple (classe lecteur de la version Console) :

```
#ifndef LECTEUR_H
#define LECTEUR_H
#include "image.h"
#include <iostream>
#include <vector>

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

class Lecteur
{
public:
    Lecteur();
    void avancer(); // incrémente _posImageCourante, modulo nbImages()
    void reculer(); // décrémente _posImageCourante, modulo nbImages()
    void changerDiaporama(unsigned int pNumDiaporama); // permet de choisir un diaporama, 0 si aucun diaporama souhaité
    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();

private:
    unsigned _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 */
    void chargerDiaporama(); // charge dans _diaporama les images du _numDiaporamaCourant
    void viderDiaporama(); // vide _diaporama de tous ses objets image et les delete
};

#endif // LECTEUR_H
```

Figure : Schéma de classes = Classe Lecteur

```
#ifndef IMAGE_H
#define IMAGE_H
#include <iostream>
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
```

Figure : Schéma de classes = Classe Image

**(d) Remarques concernant le schéma de classes**

1. 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.
2. On n'a mis ni le constructeur ni le destructeur, pour alléger le schéma.
3. 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	Teste les méthodes de la classe Lecteur

### 4.2 Test

Test avec le programme fournit main.cpp

*Valeurs fournies / attendues... comme montré dans la ressource R2.03 (partie tests)*

#### a) Résultat attendu

```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
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)

Enlever le diaporama courant = Choisir diaporama 0
0 images restantes dans le diaporama.
Lecteur vide
Press <RETURN> to close this window...
```

b) Résultat obtenu

lecteurDiaporama

lecteurDiaporama.pro

Headers

image.h  
lecteur.h

Sources

image.cpp  
lecteur.cpp  
main.cpp

sae\_v2

#include "lecteur.h"

Lecteur::Lecteur()

{

\_numDiaporamaCourant = 0; // = le lecteur est vide

}

void Lecteur::avancer()

{

if (numDiaporamaCourant() > 0 && \_posImageCourante < nbImages() - 1)

{

\_posImageCourante++;

Sortie de l'application

lecteurDiaporama sae\_v2

Lecteur vide.

16:14:13: F:\TP\R202DevApplisAvecIHM\build-lecteurDiaporama-Desktop\_Qt\_6\_3\_1\_MinGW\_64\_bit-Debug\lecteurDiaporama.exe

16:33:08: Starting F:\TP\R202DevApplisAvecIHM\build-lecteurDiaporama-Desktop\_Qt\_6\_3\_1\_MinGW\_64\_bit-Debug\lecteurDiaporama.exe

Lecteur vide.

Diaporama num. 1 selectionné.

4 images chargées dans le diaporama

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:1, titre:Grincheux, catégorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test avancer() : 4 fois

Image suivante

image( rang:2, titre:Cendrillon, catégorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)

avancer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:2, titre:Cendrillon, catégorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)

Image suivante

image( rang:3, titre:Blanche Neige, catégorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)

avancer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:3, titre:Blanche Neige, catégorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)

Image suivante

image( rang:4, titre:Mickey, catégorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

avancer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:4, titre:Mickey, catégorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

Impossible d'avancer : fin du diaporama atteinte

avancer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:4, titre:Mickey, catégorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

Test reculer() : 5 fois

Image 2 affichée.

reculer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:3, titre:Blanche Neige, catégorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)

Image 1 affichée.

reculer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:2, titre:Cendrillon, catégorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)

Image 0 affichée.

reculer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:1, titre:Grincheux, catégorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Image 3 affichée.

reculer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:4, titre:Mickey, catégorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)

Image 2 affichée.

reculer() :

Diaporama numéro 1 est en cours de lecture.

Image courante : image( rang:3, titre:Blanche Neige, catégorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)

Enlever le diaporama courant = Choisir diaporama 0

0 images restantes dans le diaporama.

non chargé

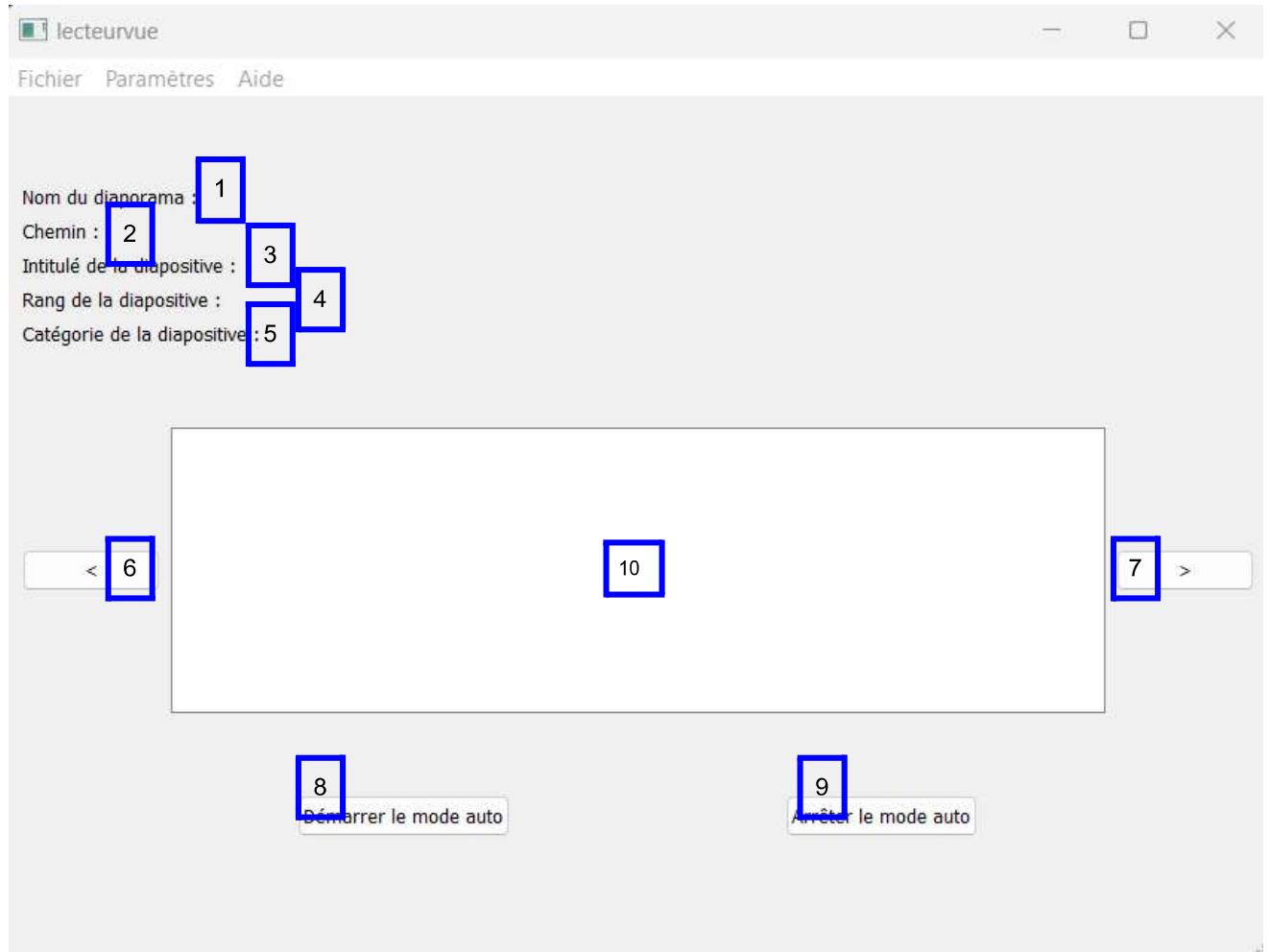
Lecteur vide.

16:33:08: F:\TP\R202DevApplisAvecIHM\build-lecteurDiaporama-Desktop\_Qt\_6\_3\_1\_MinGW\_64\_bit-Debug\lecteurDiaporama.exe exited with code 0

# Version v1 – projet Graphique seul

## 5. Éléments d'interface

*A faire ici : description sommaire des éléments de l'interface, par exemple, avec une copie d'écran sur laquelle sont nommés les variables/objets graphiques et où les layouts sont positionnés et nommés.  
Vérifier que tous les éléments graphiques qui seront manipulés par l'application ont des noms pertinents et bien formés.*



Objet	Classe
lecteurvue	QMainWindow
statusbar	QStatusBar
menubar	QMenuBar
menuParam_tres	QMenu
menuFichier	QMenu
menuAide	QMenu
centralwidget	QWidget
verticalLayout_2	QVBoxLayout
label_5	QLabel
label_4	QLabel
label_3	QLabel
label_2	QLabel
label	QLabel
horizontalLayout_2	QHBoxLayout
graphicsView	QGraphicsView
bSuivant	QPushButton
bPrecedent	QPushButton
horizontalLayout	QHBoxLayout
horizontalSpacer_3	Spacer
horizontalSpacer_2	Spacer
horizontalSpacer	Spacer
bDemarrerModeAuto	QPushButton
bArreterModeAuto	QPushButton

	Intitulé	Nom objet	Description
1	Nom Diaporama	label	Label qui permet d'afficher le nom du diaporama
2	Chemin	label_2	Label qui permet d'afficher le chemin de l'image actuelle
3	Intitulé de la diapositive	label_3	Label qui permet d'afficher l'intitulé de la diapositive actuelle
4	Rang de la diapositive	label_4	Label qui permet d'afficher le rang de l'image actuelle
5	Catégorie de la diapositive	label_5	Label qui permet d'afficher la catégorie de l'image actuelle

6	Bouton précédent	bPrecedent	Bouton qui permet de passer à l'image précédente
7	Bouton suivant	bSuivant	Bouton qui permet de passer à l'image suivante
8	Bouton démarrer le mode automatique	bDemarrerModeAuto	Bouton qui permet de démarrer le mode automatique du diaporama
9	Bouton arrêter le mode automatique	bArreterModeAuto	Bouton qui permet d'arrêter le mode automatique du diaporama
10	Emplacement d'affichage de l'image	graphicsView	Emplacement qui permet d'afficher les images du diaporama

## 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	Teste les méthodes de la classe Lecteur

Remarques sur l'implémentation :

*Commenter brièvement les choix importants d'implémentation réalisés, comme par exemple, les signals/slots*

Nous avons fait le choix de faire 4 signals/slots pour faire le déclenchement des 4 différents boutons que nous possédons.

```
connect(ui->bPrecedent, SIGNAL(clicked()), this, SLOT(precedent()));
connect(ui->bSuivant, SIGNAL(clicked()), this, SLOT(suivant()));
connect(ui->bDemarrerModeAuto, SIGNAL(clicked()), this, SLOT(demarrerModeAuto()));
connect(ui->bArreterModeAuto, SIGNAL(clicked()), this, SLOT(arreterModeAuto()));
```

### 6.2 Test

*A faire :*

*Décrire les tests prévus / réalisés pour montrer :*

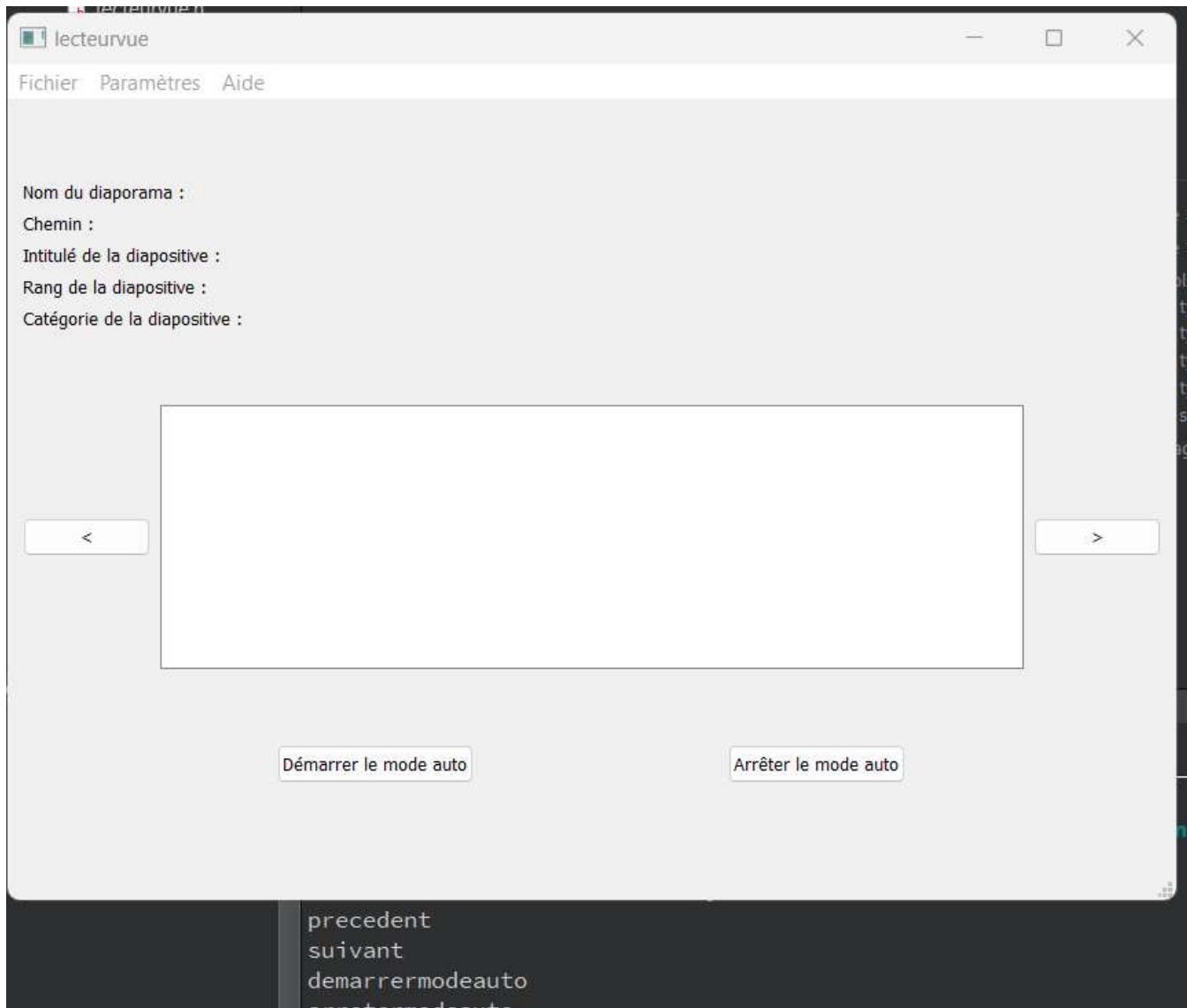
- Le comportement de l'interface non lié aux aspects fonctionnels du programme
- Le comportement de l'interface liée aux aspects fonctionnels du programme

Lié aux aspects fonctionnels du programme:

Opération testé	Description	Comportement attendu	Comportement obtenu
Clic sur la flèche “suivant”	On appuie sur la flèche suivant qui servira à passer à la diapositive suivante	Message debug qui dit qu'on a appuyé sur le bouton	<b>suivant</b>
Clic sur la flèche “précédent”	On appuie sur la flèche précédente qui servira à passer à la diapositive précédente	Message debug qui dit qu'on a appuyé sur le bouton	<b>precedent</b>
Clic sur le bouton “Démarrer le mode auto”	On appuie sur le bouton démarrer le mode auto qui servira à lancer le mode automatique	Message debug qui dit qu'on a appuyé sur le bouton	<b>demarrermodeauto</b>
Clic sur le bouton “Arrêter le mode auto”	On appuie sur le bouton démarrer le mode auto qui servira à arrêter le mode automatique	Message debug qui dit qu'on a appuyé sur le bouton	<b>arretermodeauto</b>

Non lié aux aspects fonctionnels du programme:

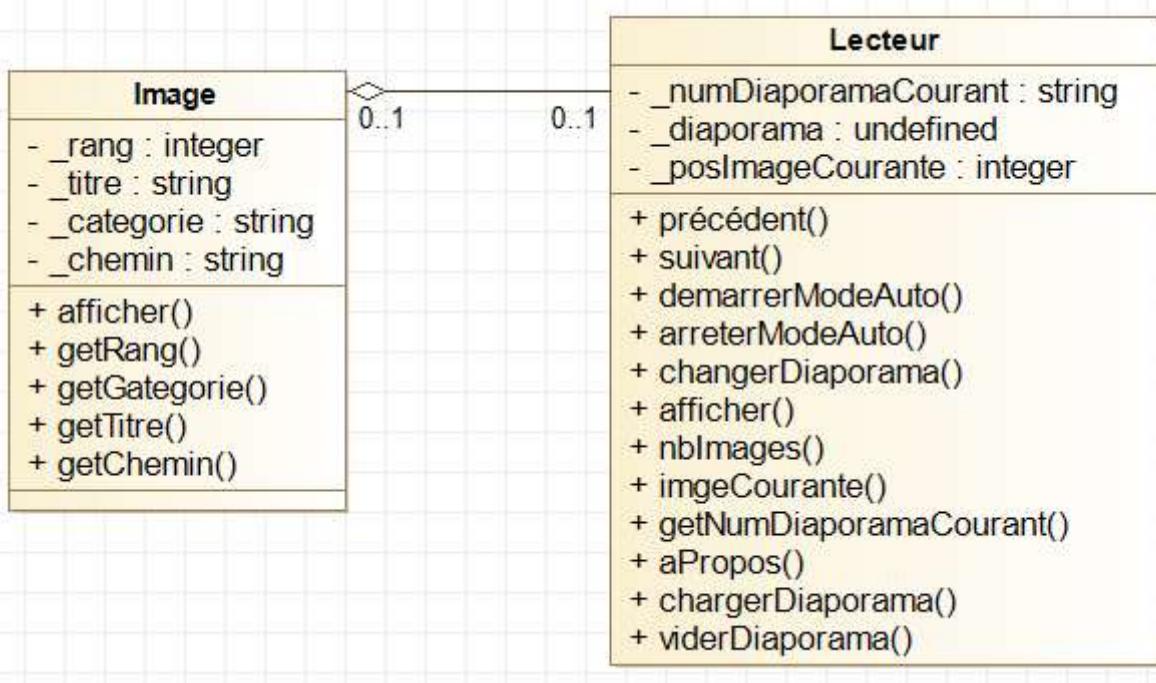
Opération testé	Description	Comportement attendu	Comportement obtenu
Agrandissement	Nous essayons d'agrandir la fenêtre pour vérifier la responsivité de la fenêtre	Tous les composants de la fenêtre s'agrandissent en même temps que la fenêtre	Le comportement obtenu est le comportement attendu
Rétrécissement	Nous essayons de rétrécir la fenêtre pour vérifier la responsivité de la fenêtre	Tous les composants de la fenêtre se retrécissent en même temps que la fenêtre	Le comportement obtenu est le comportement attendu



Dans cette capture d'écran, on peut voir notre interface et les différents messages qui s'affichent quand on clique sur les différents boutons.

## Version v2 –

### 7. Diagramme de classes (UML)



Nous avons mis le type `undefined` pour “`_diaporama`” car on ne peut pas créer de type sur modelio, le type est normalement `Diaporama`.

## 8. Comportement de l'application

### 7.1 Diagramme états-transitions-actions du lecteur de diaporamas (v2)

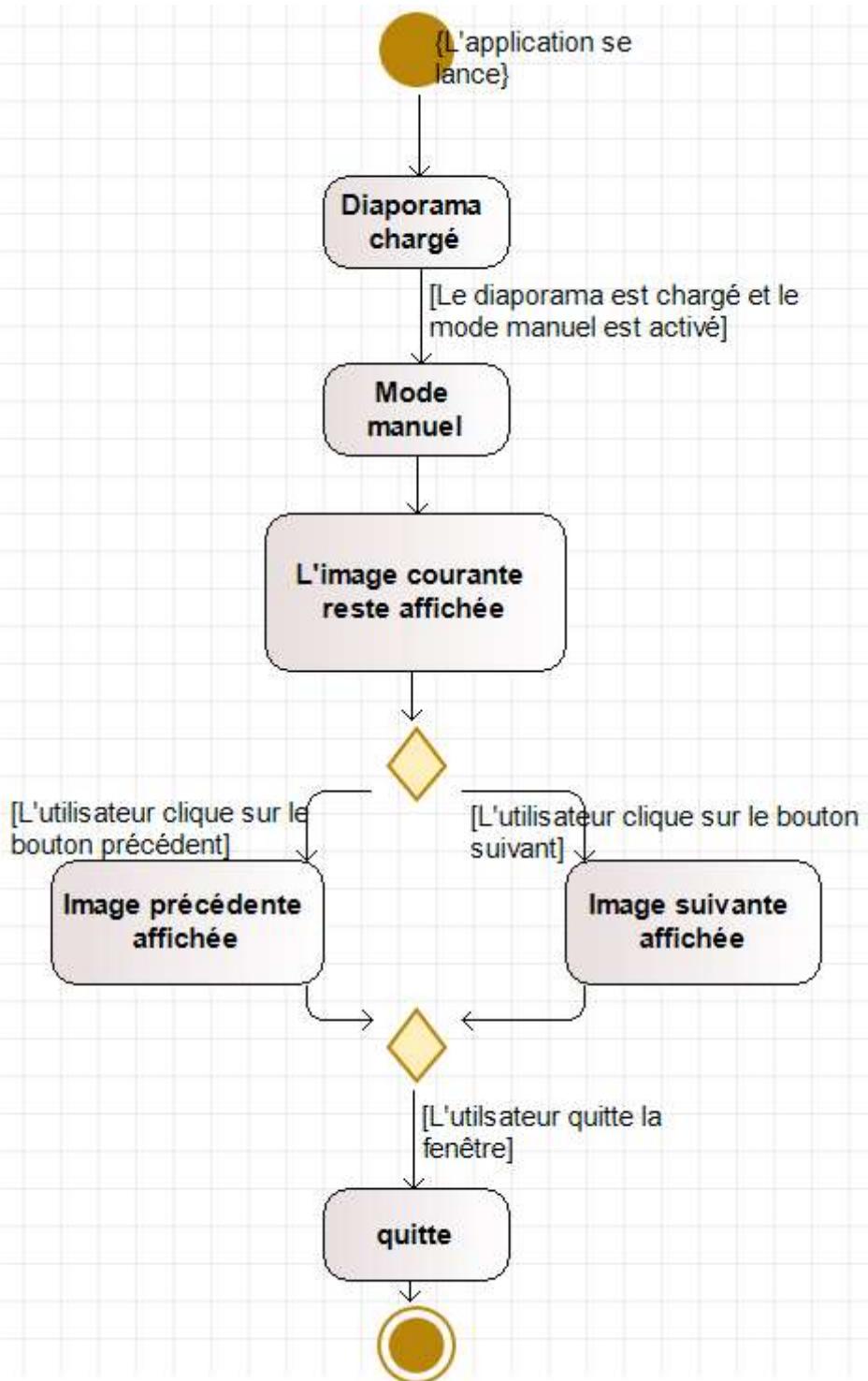


Figure : Diagramme états-transitions du lecteur de diaporamas – v2

## 7.2 Dictionnaire des états, événements et Actions (v2)

### Dictionnaire des états du diaporama

nomEtat	Signification
mode manuel	Le diaporama est en mode manuel

Tableau : États du lecteur de diaporamas – v2

### Dictionnaire des événements faisant changer le diaporama d'état

<i>nomEvénement</i>	<i>Signification</i>
Clic sur le bouton suivant	L'utilisateur clique sur le bouton qui lui permet de passer à l'image suivante
Clic sur le bouton précédent	L'utilisateur clique sur le bouton qui lui permet de passer à l'image précédente.

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

### Description des actions réalisées lors de la traversée des transitions

<i>nomAction</i>	<i>Signification</i>
L'application diaporama s'ouvre	L'application démarre
Le mode manuel est activé	L'utilisateur passe les diapositives
Le diaporama est quitté	L'utilisateur quitte le diaporama

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

### 7.3 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 →		
Événement → nomEtat	Clic sur le bouton suivant	Clic sur le bouton précédent
Mode manuel		

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

L'intérêt de cette vue matricielle est qu'elle permet une préparation naturelle et aisée de l'étape suivante de programmation.

## 9. Implémentation et tests

### 8.1 Implémentation (v2)

Liste et rôle des fichiers de cette version :

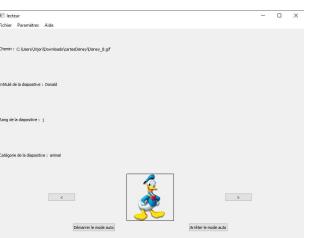
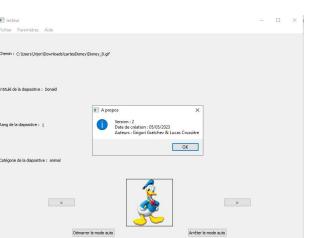
lecteur.ui	Fichier du dessin de l'interface réalisé par QtDesigner
lecteur.h	Spécification de la classe Lecteur. Ce fichier contient la déclaration de la classe "lecteur" qui représente un lecteur de diaporama d'images.
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image Ce fichier contient la déclaration de la classe "Image" qui représente une image avec des informations telles que son rang, sa catégorie, son titre et son chemin.
image.cpp	Corps de la classe Image
main.cpp	Permet de lancer l'application et ouvre la fenêtre Lecteur

On a fusionné la classe Lecteur et LecteurVue comme c'était demandé dans le sujet donc on a 2 fichiers .h.

Remarques sur l'implémentation :

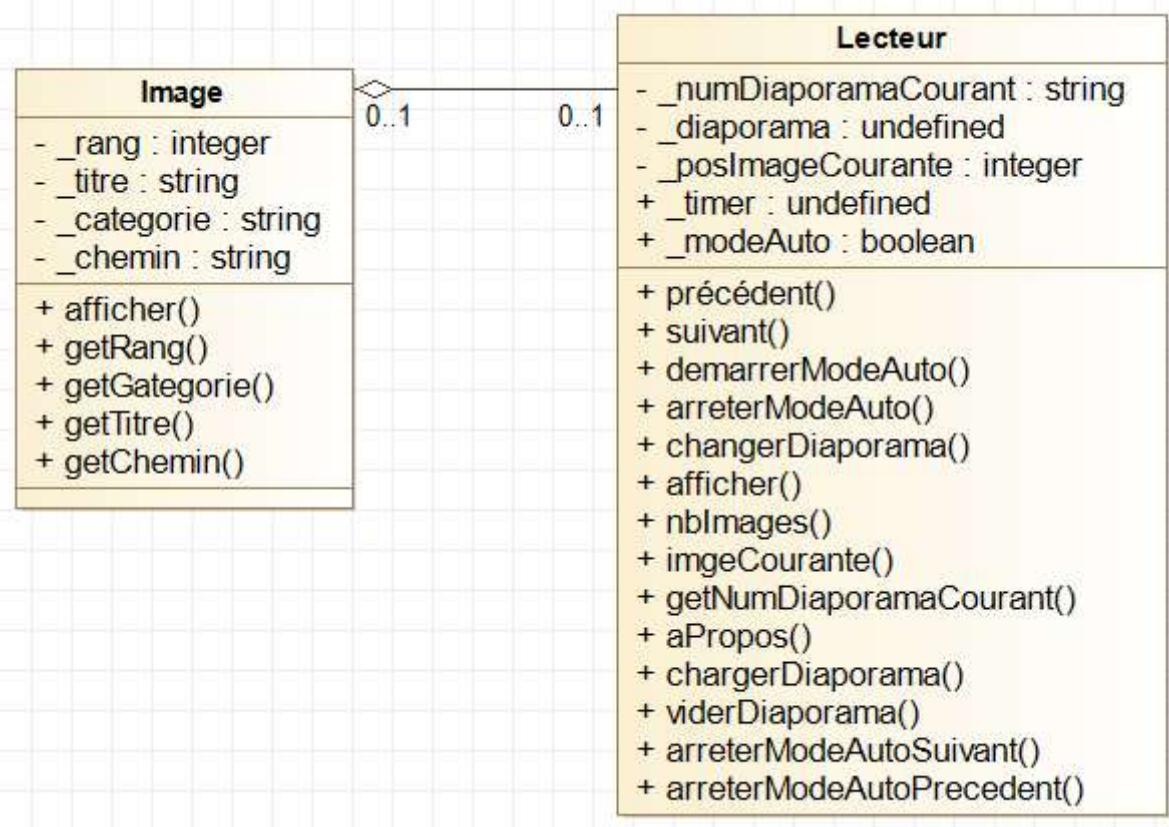
Chaque bouton a un slot dédié pour son signal clicked() (ou triggered() s'il s'agit d'un bouton dans le menu "fichier").

### 8.2 Tests (v2)

Opération testé	Description	Comportement attendu	Comportement obtenu
L'application est lancée	L'utilisateur lance l'application	La fenêtre de l'application s'ouvre	
Passage à l'image suivante	En faisant cette opération, ça passe de l'image actuelle à l'image d'après	Affichage de l'image suivante	
Passage à l'image précédente	En faisant cette opération, ça passe de l'image actuelle à l'image d'avant	Affichage de l'image précédente	
Le menu Aide >> À propos de... ouvre une Boîte de Message	En faisant cette opération, en cliquant dans le menu Aide et ensuite À propos de, cela nous permet d'avoir certaines informations sur l'application, comme la version de l'application, la date de création ou encore les auteurs de l'application	Fait apparaître une boîte de dialogue avec les informations attendus	
Le menu Fichier >> Quitter permet d'arrêter l'application	En faisant cette opération, en cliquant dans le menu sur Fichier, on peut quitter l'application en cliquant sur Quitter	Ferme l'application	Le comportement obtenu est le comportement attendu.

## Version v3 –

### 10. Diagramme de classes (UML)



*Nous avons mis le type `undefined` pour “`_diaporama`” car on ne peut pas créer de type sur modelio, le type est normalement `Diaporama`. Et pareil pour “`_timer`” qui est de type `QTimer`.*

## 11. Comportement de l'application

### 7.4 Diagramme états-transitions-actions du lecteur de diaporamas (v3)

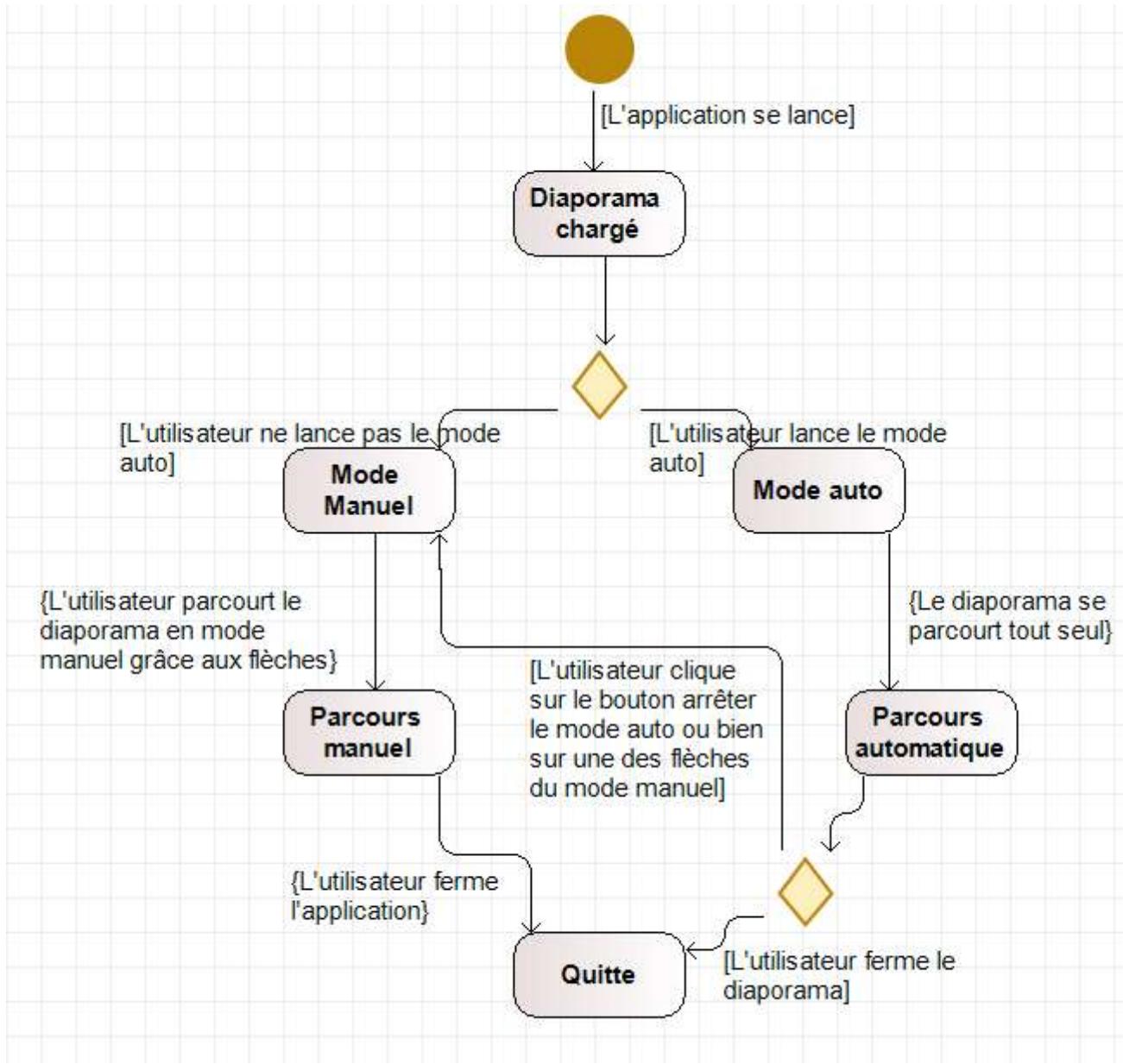


Figure : Diagramme états-transitions du lecteur de diaporamas – v3

## 7.5 Dictionnaire des états, événements et Actions (v3)

### Dictionnaire des états du diaporama

<i>nomEtat</i>	<i>Signification</i>
Diaporama Chargé	Le diaporama est chargé
Mode Manuel	Le diaporama est en mode manuel
Mode automatique	Le diaporama est en mode automatique

Parcours manuel	L'utilisateur parcourt le diaporama grâce aux flèches
Parcours auto	Le diaporama se parcourt tout seul
Diaporama quitté	L'utilisateur quitte le diaporama

Tableau : États du lecteur de diaporamas – v3

#### Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	Signification
Lancement de l'application	L'utilisateur lance l'application
L'utilisateur ne lance pas le mode auto	L'utilisateur déroule le diaporama manuellement
L'utilisateur clique sur le bouton mode automatique	Le diaporama se déroule tout seul
L'utilisateur utilise les flèches	L'utilisateur utilise les flèches précédent et suivant pour se balader dans le diaporama
L'utilisateur a activé le mode auto et utilise les flèches	L'utilisateur utilise les flèches pendant le mode automatique donc le diaporama passe en manuel
L'utilisateur quitte le diaporama	L'utilisateur va dans le menu et quitte le diaporama

Tableau : Evénements faisant changer le diaporama d'état – v3

#### Description des actions réalisées lors de la traversée des transitions

nomAction	Signification
Lancement de l'application	L'application s'ouvre et le diaporama se charge
passage en mode manuel	Le diaporama est en mode manuel
Passage en mode automatique	L'utilisateur passe le diaporama en mode automatique
L'utilisateur utilise ou la flèche suivante ou la flèche précédente	Passe à la diapo suivant ou précédente, et si le mode automatique est annulé, cette action l'enlève
Le diaporama se parcourt automatiquement	Le diaporama se parcourt automatiquement

L'utilisateur quitte le diaporama	L'utilisateur va dans le menu et quitte le diaporama
-----------------------------------	--

Tableau : Actions à réaliser lors des changements d'état – lecteur de diaporamas v3

### 7.6 Table T\_EtatsEvenementsActions (v3)

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 →			Démarrer le mode auto	<		
Événement → nomEtat	Lancement de l'application	L'utilisateur ne lance pas le mode auto	L'utilisateur clique sur le bouton mode automatique	L'utilisateur utilise les flèches	L'utilisateur a activé le mode auto et utilise les flèches	L'utilisateur quitte le diaporama
Charge	L'application s'ouvre et le diaporama se charge					
Mode Manuel		passage en mode manuel				
Mode Automatique			Passage en mode automatique			
Parcours manuel				L'utilisateur utilise ou la flèche suivante ou la flèche précédente		
Parcours auto					Le diaporama se parcourt automatiquement	
Diaporama quitté						L'utilisateur quitte le diaporama

Tableau : Matrice d'états-transitions du lecteur de diaporamas – v3

L'intérêt de cette vue matricielle est qu'elle permet une préparation naturelle et aisée de l'étape suivante de

programmation.

## 12. Implémentation et tests

### 8.3 Implémentation (v3)

Liste et rôle des fichiers de cette version :

lecteur.vue.ui	Fichier du dessin de l'interface réalisé par QtDesigner
lecteur.h	Spécification de la classe Lecteur. Ce fichier contient la déclaration de la classe "lecteur" qui représente un lecteur de diaporama d'images.
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image Ce fichier contient la déclaration de la classe "Image" qui représente une image avec des informations telles que son rang, sa catégorie, son titre et son chemin.
image.cpp	Corps de la classe Image
main.cpp	Permet de lancer l'application et ouvre la fenêtre Lecteur

On a fusionné la classe Lecteur et LecteurVue comme c'était demandé dans le sujet donc on a 2 fichiers .h.

Remarques sur l'implémentation :

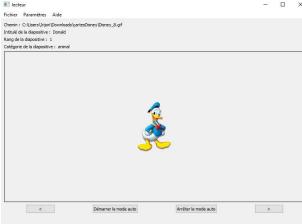
Les boutons suivant et précédent ont maintenant 2 slots chacun pour le signal clicked(). Les slots suivant() et precedent(), qui permettent d'afficher l'image suivante ou précédente ainsi que les slots arreterModeAutoSuivant() et arreterModeAutoPrecedent() qui servent à arrêter le mode auto quand le bouton suivant ou précédent est cliqué et qui ne s'exécutent que si le lecteur est en mode auto.

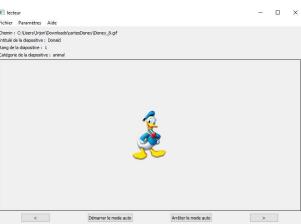
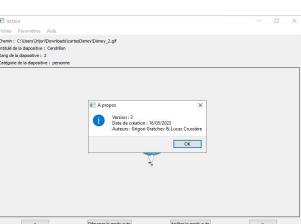
### 8.4 Tests (v3)

A faire :

Décrire les tests prévus / réalisés pour montrer :

- Le comportement de l'interface non lié aux aspects fonctionnels du programme
- Le comportement de l'interface liée aux aspects fonctionnels du programme
- Le comportement fonctionnel de l'application

Opération testé	Description	Comportement attendu	Comportement obtenu
L'application est lancée	L'utilisateur lance l'application	La fenêtre de l'application s'ouvre	
Passage à l'image suivante	En faisant cette opération, ça passe de l'image actuelle à l'image d'après	Affichage de l'image suivante	

Passage à l'image précédente	En faisant cette opération, ça passe de l'image actuelle à l'image d'avant	Affichage de l'image précédente	
Le menu <b>Aide &gt;&gt; A propos de...</b> ouvre une Boîte de Message	En faisant cette opération, en cliquant dans le menu Aide et ensuite À propos de, cela nous permet d'avoir certaines informations sur l'application, comme la version de l'application, la date de création ou encore les auteurs de l'application	Fait apparaître une boîte de dialogue avec les informations attendues	
Le menu <b>Fichier &gt;&gt; Quitter</b> permet d'arrêter l'application	En faisant cette opération, en cliquant dans le menu sur Fichier, on peut quitter l'application en cliquant sur Quitter	Permet de quitter l'application grâce à un bouton dans le menu	Le comportement obtenu est le comportement attendu.
L'utilisateur clique sur le bouton activer le mode automatique	En faisant cette opération, ça passe le diaporama en mode automatique	Permet d'activer le mode automatique du diaporama et donc de défiler les images toutes seules	Le comportement obtenu est le comportement attendu.
L'utilisateur clique sur le bouton arrêter le mode automatique ou clique sur l'une des flèches	En faisant une de ces opérations, ça enlève le mode automatique du diaporama	Permet de repasser le diaporama en mode manuel	Le comportement obtenu est le comportement attendu.