Kamelia Hamchouche
Imane Mahssini
Marylou Lohier

Saé 2.01 – Développement d'une application

Lecteur de diaporamas – Dossier d'Analyse et conception

---

1. **Compléments de spécifications externes.**

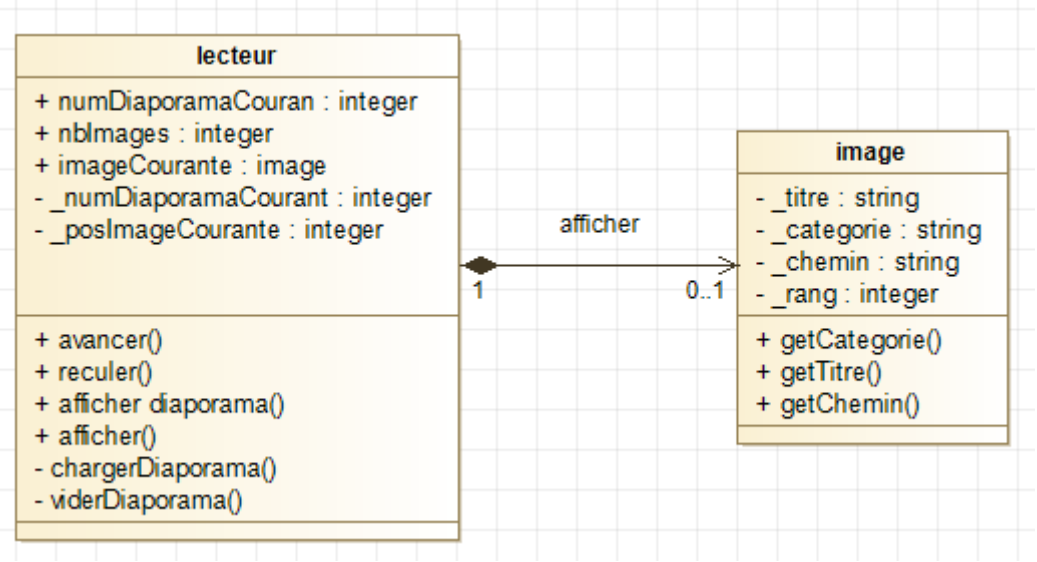
Pour le Timer nous ne savions pas qu'il fallait récupérer la saisie de l'utilisateur au lieu de lui faire choisir une vitesse parmi d'autres. Pour la boîte de dialogue on ne savait pas au début qu'il fallait prendre une boîte de dialogue prédéfinie comme le TP sur le questionnaire du nombre d'enfants.

2. **Scénarios**

Cas d'utilisation	**mode manuel**	
Acteur primaire	utilisateur du diaporama	
secteur	application de diaporama	
Niveau	Objectif utilisateur	
Préconditions	L'utilisteur a ouvert l'application et a choisis son diporama avec les filtres	
Opérations	**Acteur**	**Diaporama**
1	l'utilisateur choisit le mode manuel par default	
2		Le système affiche la première image du diaporama
3	l'utilisateur demande à afficher l'image suivante	
4		Le système affiche l'image suivante du diaporama et met à jour les nouvelles informations de l'image
5	l'utilisateur demande à voir l'image précédente	
6		Le système affiche l'image précédente et met à jour les nouvelles informations de l'image
extension		
3.A	**demande de passer au mode automatique**	
3.A.1	l'utilisateur demande de passer en mode automatique	
3.A.2		le système lance le diaporama en mode automatique
3.A.3		le système demande à l'utilisateur de saisir une vitesse de défilement
3.A.4	l'utilisateur saisie une vitesse	
3.A.5		le système défile toutes les images une par une selon la vitesse de défilement saisie

3. **Diagramme de classe (UML)**

a)



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	`unsigned int`	`1`
`_titre`	intitulé de l'image	`string`	`chateau`
`_categorie`	catégorie de l'image	`string`	`animal`
`_chemin`	chemin complet vers le dossier où se trouve l'image	`string`	`c:/PERSONNAGE/DISNEY`

Classe *Lecteur*			
Nom attribut	**Signification**	**Type**	**Exemple**
`numDiapoCouran`	numéro du diaporama courant, par défaut 0	`unsigned int`	`1`
`nbImages`	pointeurs vers les images du diaporama	`Diaporama`	`chateau`
`ImageCourante`	position dans le diaporama de l'image courante. Indéfini quand le diaporama est vide. Démarre à 0 quand le diaporama est vide	`unsigned int`	`1`
`_numDiapoCourant`	numéro du diaporama courant, par défaut 0	`integer`	`2`
_posImageCourante	position, dans le diaporama,de l'image courante. Indéfini quand diaporama vide. Démarre à 0 quand diaporama non vide	`integer`	`2`

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 <vector>

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

class Lecteur
{
public:
   ~Lecteur();
   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 */
private:
   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 1 : Schéma de classes = Classe Lecteur

```
#ifndef IMAGE_H
#define IMAGE_H
#include <iostream>
using namespace std;

class Image
{
public:
   ~Image();
   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 2 : Schéma de classes = Classe Image

Remarques concernant le schéma de classes

On a pas encore mis d'elements grahiques car on se cncentre dans cette version sur elements qu'on trouvera dans la fenêtre.Par contre, on a mis les méthodes getxxx(), qui permettent aux objets métier de communiquer leur valeur aux objets graphiques pour que ceux-ci s'affichent.

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

C'est censé afficher dans la console des lignes de texte précisant pour chaque diaporama son rang, son nom, son URL, son numéro de diapositive et sa catégorie.

Test: ChangerDiaporama()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
`numDiaporamaCourant`				lecteur vide
	changerDiaporama()	numDiaporamaCourant=1 posImageCourante=1	numDiaporamaCourant=1 posImageCourante=1	Diaporama num1 selectionne 4image chargées dans le diaporama Diaporama image courant: image(rang:1, titre:Cendrillon, categorie: personne, chemin:C:\carte_dysnei…

Test: Avancer()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
posImageCourante				image courante
	Avancer()	_posImageCourant	_posImageCourant	avancer() :

		e = (_posImageCourante + 1) % nbImages();	e = (_posImageCourante + 1) % nbImages();	Diaporama num. 1 image(rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif) avancer() : Diaporama num. 1 image(rang:2, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test: Reculer()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
posImageCourante				image courante
	Reculer()	_posImageCourante = (_posImageCourante + nbImages() - 1) % nbImages();	_posImageCourante = (_posImageCourante + nbImages() - 1) % nbImages();	Reculer() Diaporama num. 1 image(rang:4, titre:Mickey Mouse, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif) Reculer() : Diaporama num. 1 image(rang:3, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test: chargerDiaporama()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
imageACharger				numero titre et categorie de l'image courante
	chargerDiaporama()	_diaporama.push_back(*imageACharger*);	_diaporama.push_back(*imageACharger*);	numero titre et categorie de l'image courante

Test: Afficher()

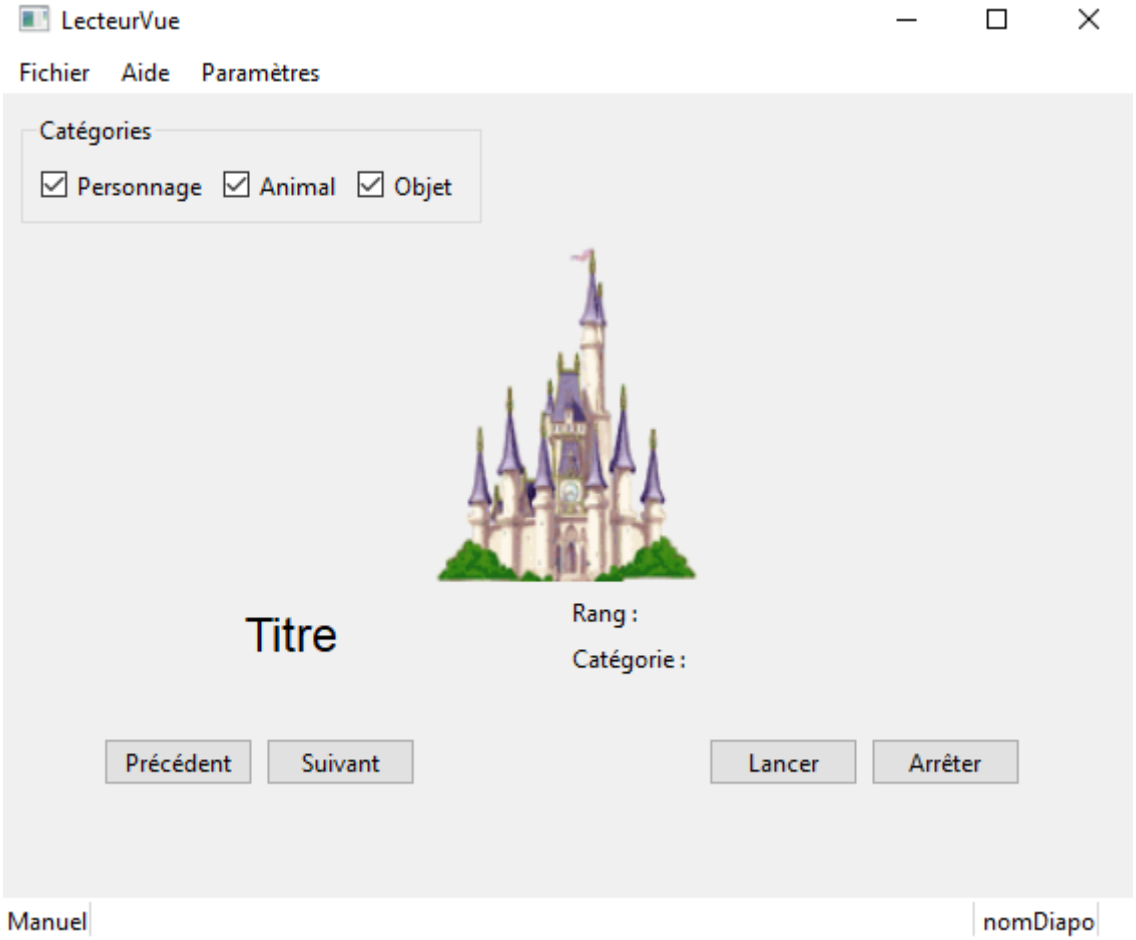
Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
numDiaporamaCourant(				diaporama vide
	Afficher()	_diaporama[0]->afficher();	_diaporama[0]->afficher();	affiche les informations sur le lecteur

Test: viderDiaporama()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
numDiaporamaCourant(				enlever le diaporama courant
	viderDiaporama()	_posImageCourante = 0;	_posImageCourante = 0;	0 images restantes dans le diaporama

Version v1 – projet Graphique seul

5. Éléments d'interface



Nous avons décidé de faire un diaporama responsive seulement vers le bas car l'image ne peut pas s'agrandir donc cela ne servait à rien, différents boutons servent pour la navigation du diaporama, les catégories que l'utilisateur choisit, les informations sur l'image courante et les 3 onglets Fichier, Aide et Paramètre. Nous n'avons pas fait de boutons pause car ça n'existe pas dans ce diaporama. Il y a deux modes: manuel ou automatique. Enfin il y a également des informations sur l'image qui est visionnée et de cases qui peuvent être cochées ou pas selon le filtre que l'on veut ajouté à notre 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 avec les éléments graphiques de la fenetre principal
main.cpp	Teste les méthodes de la classe Lecteur

Remarques sur l'implémentation :

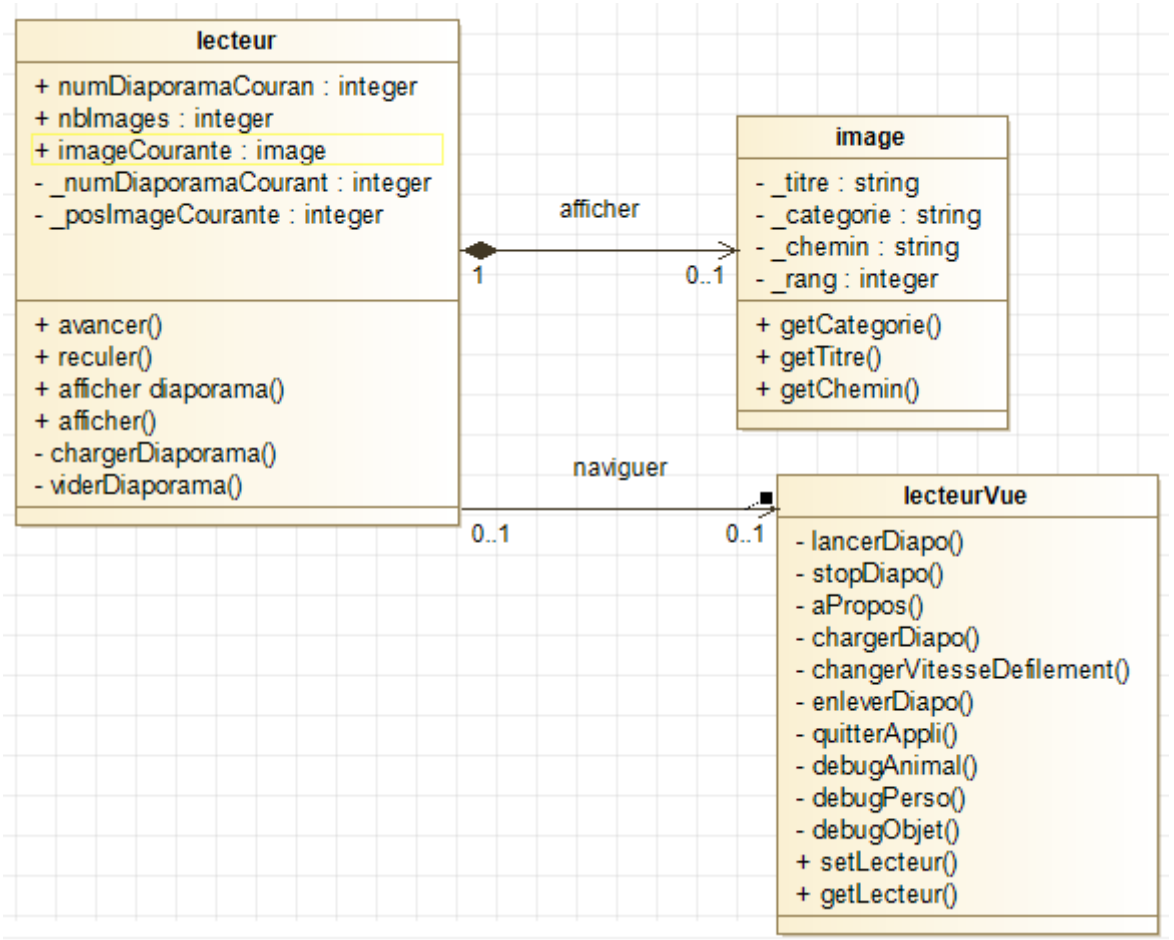
Nous avons attribué pour chaque bouton et actions des QDebug afin de vérifier le code et voir si toutes les interactions fonctionnes

6.2 Test

éléments testés	action testés	résultats obtenue	affichage à l'écran
case animal	case cochée	ok	filtrer les image animal
case animal	case décochée	ok	pas de filtre pour animal
case personne	case cochée	ok	filtrer les image personne
case personne	case décochée	ok	pas de filtre pour personne
case objet	case cochée	ok	filtrer les image objet
case objet	case décochée	ok	pas de filtre pour objet
bouton précédent	clic	ok	diaporama précédent
bouton suivant	clic	ok	diaporama suivant
bouton lancer	clic	ok	diaporama lancé
bouton arrêter	clic	ok	diaporama arrêté
Fichier: Quitter	sélectionné	l'application se ferme	rien
Paramètre: vitesse défilement	sélectionné	saisie de l'utilisateur	rien ca ne s'affiche pas à l'écran
Paramètre: charger diaporama	sélectionné	diaporama chargé	les images se charge et s'affichent.
Aide: à propos de	sélectionné	message affiché	informations sur les développeurs
Paramètre : enlever diaporama	sélectionné	diaporama retiré	plus d'image car il n'y a plus de diaporama

Version v2 –

7. Diagramme de classes (UML)



8. Comportement de l'application

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

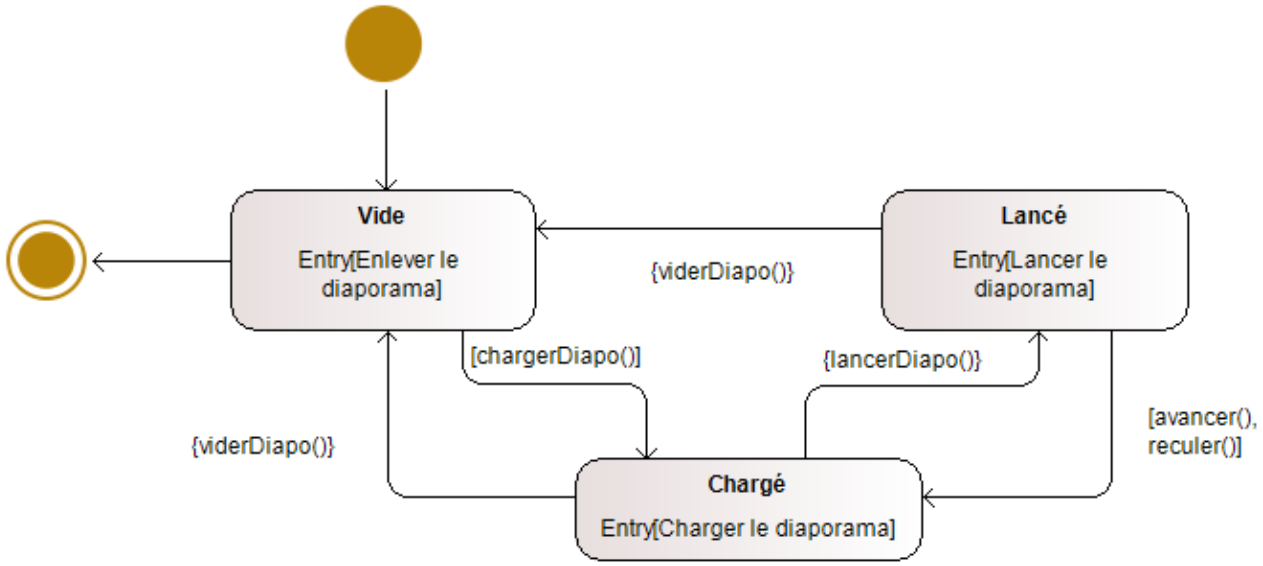


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v2

7.2 Dictionnaire des états, événements et Actions (v2)

Dictionnaire des états du diaporama

nomEtat	*Signification*
`vide`	Le diaporama contient aucune image
`chargé`	Le diaporama est chargé
`lancé`	Le diaporama a été lancé en mode auto

Tableau 2 : États du lecteur de diaporamas – v2

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
`viderDiapo`	Le diaporama est vidé : chargé -> vidé
`chargerDiapo`	Le diaporama est chargé : vidé -> chargé
`lancerDiapo`	Le diaporama est lancé : chargé -> lancé

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

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
`Vider le diaporama`	Les images sont retirées du diaporama pour le vider
`Charger un diaporama`	Les images sont chargées dans le diaporama
`Lancer le diaporama`	Le diaporama est lancé

Tableau 4 : 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 □	`viderDiapo`	`chargerDiapo`	`lancerDiapo`
Événement □ *nomEtat*	`Vider le diaporama`	`Charger le diaporama`	`Lancer le diaporama`
vide	x		
chargé		x	x
lancé			x

Tableau 5 : 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.

8.Implémentation et tests

8.1 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 définit toutes les méthodes qui permettent d'agir sur le diaporama*
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. *Elle définit toutes les variables centrées autour du diaporama et les méthodes initiales qui agissent sur le diaporama*
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image *Elle permet de définir les différentes informations d'une image et de définir les méthodes qui donnent ces informations.*

image.cpp	Corps de la classe Image
main.cpp	Met en place l'application, le lecteur, la fenêtre et affiche l'ensemble où le lecteur est intégré dans la fenêtre.

Remarques sur l'implémentation :

Nous avons choisi de connecter les boutons aux procédures qui leur correspondent.
Nous avons choisi de séparer le code en plusieurs fichiers distincts pour nous y retrouver.

8.2 Tests (v2)

idem v1 +

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
debugAnimal	**case filtrage animal**	**qDebug() << "Animal sélectionné"<< Qt::endl;**	**qDebug() << "Animal sélectionné"<< Qt::endl;**	**Animal sélectionné**
debugPerso	**debugAnimal case filtrage personne**	**qDebug() << "Personnage sélectionné"<< Qt::endl;**	**qDebug() << "Personnage sélectionné"<< Qt::endl;**	**Personnage sélectionné**
debugObjet	**debugAnimal case filtrage objet**	**qDebug() << "Objet sélectionné"<< Qt::endl;**	**qDebug() << "Objet sélectionné"<< Qt::endl;**	**Objet sélectionné**

Version v3–

9 Diagramme de classes (UML)

idem que v2

10.Comportement de l'application

10.1Diagramme états-transitions-actions du lecteur de diaporamas (v3)

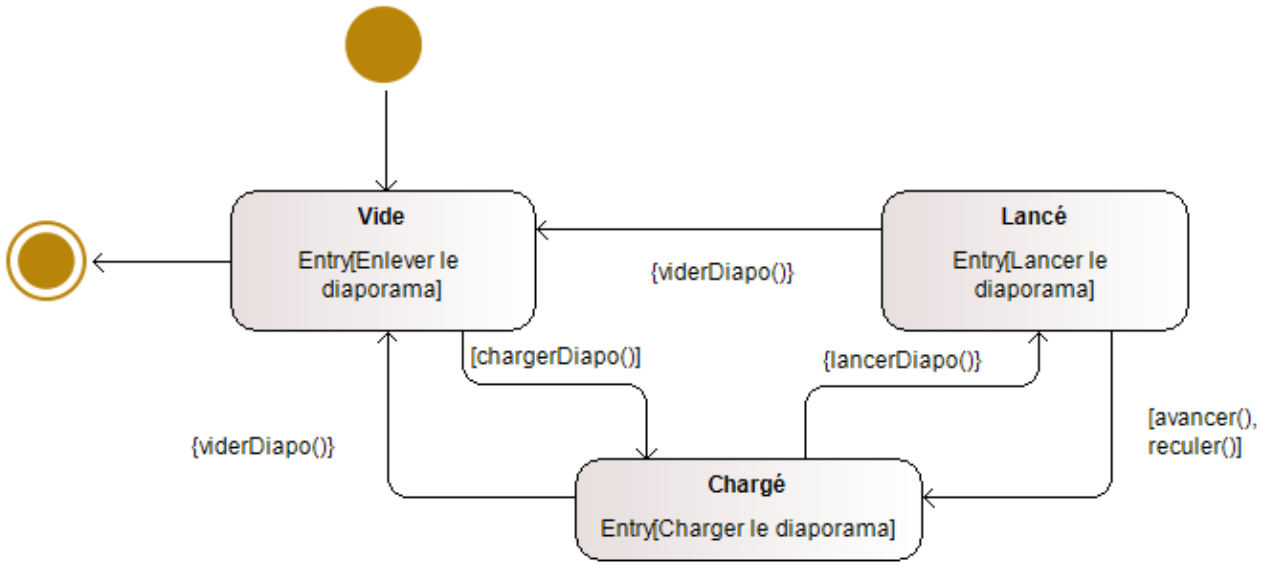


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v3

10.2Dictionnaire des états, événements et Actions (v3) (à vérifier)

Dictionnaire des états du diaporama

nomEtat	*Signification*
`vide`	Le diaporama contient aucune image
`chargé`	Le diaporama est chargé
`lancé`	Le diaporama a été lancé en mode auto

Tableau 2 : États du lecteur de diaporamas – v3

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*

viderDiapo	Le diaporama est vidé : chargé -> vidé
chargerDiapo	Le diaporama est chargé : vidé -> chargé
lancerDiapo	Le diaporama est lancé : chargé -> lancé

Tableau 3 : Événements faisant changer le diaporama d'état – v3

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
Vider le diaporama	Les images sont retirées du diaporama pour le vider
Charger un diaporama	Les images sont chargées dans le diaporama
Lancer le diaporama	Le diaporama est lancé

Tableau 4 : Actions à réaliser lors des changements d'état – lecteur de diaporamas v3

Table T_EtatsEvenementsActions (v3) (à vérifier)

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*	viderDiapo	chargerDiapo	lancerDiapo
	Vider le diaporama	Charger le diaporama	Lancer le diaporama
vide	x		
chargé		x	x
lancé			x

Tableau 5 : 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.

11Implémentation et tests

11.1 Implémentation (v3)

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas *Elle définit toutes les méthodes qui permettent d'agir sur le diaporama*
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. *Elle définit toutes les variables centrées autour du diaporama et les méthodes initiales qui agissent sur le diaporama*
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image *Elle permet de définir les différentes informations d'une image et de définir les méthodes qui donnent ces informations.*
image.cpp	Corps de la classe Image
main.cpp	Met en place l'application, le lecteur, la fenêtre et affiche l'ensemble où le lecteur est intégré dans la fenêtre.

Remarques sur l'implémentation :

La majorité de nos méthodes se fon par appel par signal

11.2 Tests (v3)

Test: ChangerDiaporama()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
`numDiaporamaCourant`				lecteur vide
	changerDiaporama()	numDiaporamaCourant=1 posImageCourante=1	numDiaporamaCourant=1 posImageCourante=1	Diaporama num1 selectionne 4image chargées dans le diaporama Diaporama image courant: image(rang:1, titre:Cendrillon, categorie: personne, chemin:C:\carte_dysnei…

Test: Avancer()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
posImageCourante				image courante
	Avancer()	_posImageCourante =	_posImageCourante =	avancer() : Diaporama num. 1

		(_posImageCourante + 1) % nbImages();	(_posImageCourante + 1) % nbImages();	image(rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif) avancer() : Diaporama num. 1 image(rang:2, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test: Reculer()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
posImageCourante				image courante
	Reculer()	_posImageCourante = (_posImageCourante + nbImages() - 1) % nbImages();	_posImageCourante = (_posImageCourante + nbImages() - 1) % nbImages();	Reculer() Diaporama num. 1 image(rang:4, titre:Mickey Mouse, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif) Reculer() : Diaporama num. 1 image(rang:3, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)

Test: chargerDiaporama()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
imageACharger				numero titre et categorie de l'image courante
	chargerDiaporama()	_diaporama.push_back(*imageACharger*);	_diaporama.push_back(*imageACharger*);	numero titre et categorie de l'image courante

Test: Afficher()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
numDiaporamaCourant(				diaporama vide
	Afficher()	_diaporama[0]->afficher();	_diaporama[0]->afficher();	affiche les informations sur le lecteur

Test: viderDiaporama()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
numDiaporamaCourant(				enlever le diaporama courant
	viderDiaporama()	_posImageCourante = 0;	_posImageCourante = 0;	0 images restantes dans le diaporama

Version v4 –

12 Diagramme de classes (UML)

idem que v2

13 Comportement de l'application

Diagramme états-transitions-actions du lecteur de diaporamas (v4)

idem v2

Dictionnaire des états, événements et Actions (v4)

Dictionnaire des états du diaporama

nomEtat	*Signification*
`vide`	Le diaporama contient aucune image
`chargé`	Le diaporama est chargé
`lancé`	Le diaporama a été lancé en mode auto

Tableau 2 : États du lecteur de diaporamas – v4

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
`viderDiapo`	Le diaporama est vidé : chargé -> vidé
`chargerDiapo`	Le diaporama est chargé : vidé -> chargé
`lancerDiapo`	Le diaporama est lancé : chargé -> lancé

Tableau 3 : Événements faisant changer le diaporama d'état – v4

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
`Vider le diaporama`	Les images sont retirées du diaporama pour le vider
`Charger un diaporama`	Les images sont chargées dans le diaporama
`Lancer le diaporama`	Le diaporama est lancé

Tableau 4 : Actions à réaliser lors des changements d'état – lecteur de diaporamas v4

Table T_EtatsEvenementsActions (v4)

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 □	`viderDiapo`	`chargerDiapo`	`lancerDiapo`
Événement □ *nomEtat*	`Vider le`	`Charger le`	`Lancer le`

	diaporama	diaporama	diaporama
vide	x		
chargé		x	x
lancé			x

Tableau 5 : Matrice d'états-transitions du lecteur de diaporamas – v4

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

12.1 Implémentation (v3) Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas *Elle définit toutes les méthodes qui permettent d'agir sur le diaporama*
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. *Elle définit toutes les variables centrées autour du diaporama et les méthodes initiales qui agissent sur le diaporama*
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image *Elle permet de définir les différentes informations d'une image et de définir les méthodes qui donnent ces informations.*
image.cpp	Corps de la classe Image
main.cpp	Met en place l'application, le lecteur, la fenêtre et affiche l'ensemble où le lecteur est intégré dans la fenêtre.

Tableau 5 : Matrice d'états-transitions du lecteur de diaporamas – v4

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

10. Implémentation et tests

13.1 Implémentation (v4)

Liste et rôle des fichiers de cette version :

lecteurVue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas *Préciser le rôle*
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 *Préciser le rôle*
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image *Préciser le rôle*
image.cpp	Corps de la classe Image
main.cpp	??

13.2 Tests (v4)

idem v3

Version v5 –

11. Diagramme de classes (UML)

idem v2

12. Comportement de l'application

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

idem v2

Dictionnaire des états du diaporama

nomEtat	*Signification*
`vide`	Le diaporama contient aucune image
`chargé`	Le diaporama est chargé
`lancé`	Le diaporama a été lancé en mode auto

Tableau 2 : États du lecteur de diaporamas – v5

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
`viderDiapo`	Le diaporama est vidé : chargé -> vidé
`chargerDiapo`	Le diaporama est chargé : vidé -> chargé
`lancerDiapo`	Le diaporama est lancé : chargé -> lancé

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

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
`Vider le diaporama`	Les images sont retirées du diaporama pour le vider

`Charger un diaporama`	Les images sont chargées dans le diaporama
`Lancer le diaporama`	Le diaporama est lancé

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

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 → / *Événement* ↓ *nomEtat*	`viderDiapo`	`chargerDiapo`	`lancerDiapo`
	`Vider le diaporama`	`Charger le diaporama`	`Lancer le diaporama`
vide	x		
chargé		x	x
lancé			x

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

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

13. Implémentation et tests

12.1 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 *Vue*
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 *Présentation*
lecteur.cpp	Corps de la classe Lecteur
image.h	Spécification de la classe Image *Modèle*
image.cpp	Corps de la classe Image
main.cpp	Lance l'application et le lecteur se trouvant dans l'application

Remarques sur l'implémentation :

idem

12.2 Tests (v5)

idem sauf avancer()

Etat_initial	Action testée	Résultat Attendus	Résultat obtenue	Affichage à l'écran
imageACharger				
	chargerDiaporama ()	récupération des données OK		

14. Bilan

Dépôt Git où trouver le projet complet (les versions réalisées):
https://github.com/lgmarylou/S2.01

Temps global de travail (pour le groupe):
Nous avons passer globalement 5 heures par semaine

Apprentissages majeurs:
- gérer l'affichage d'images
- optimisation du code et organisation du code
- utilisation d'une boîte de dialogue
- utilisation d'un timer
- importation d'une base de données et son utilisation

Difficulté majeure : codage des fonctions avancées et précédent en mode automatique

Points positifs / négatifs de l'activité
- On ne nous a pas assez aidées dans les consignes
- Il manque quelques séances accompagnés
- Pour le nombre de versions et les attentes le temps est un peu juste une ou deux semaine de plus aurait été utile

Le sujet de la SAE était intéressant il nous a permis d'appréhender les fonctionnalité QT principal.
Le fait que le sujet soit divisés en plusieurs partie est une bonne idée.
La v0 nous a permis de mieux comprendre le sujet et ce qui était demandé.