Saé 2.01 – Développement d'une application

Lecteur de diaporamas – Dossier d'Analyse et conception

HERRMANN Anthony - GOUMEAUX Gauthier - MARTIN Solène
TD2 - TP3

1. **Compléments de spécifications externes.**

Lors de la lecture du sujet, il est dit que le mode de fonctionnement en cours est toujours affiché dans la barre de statut, or il n'est pas marqué qu'il faille la manipuler. Nous avons alors demandé à M. Nodenot puis à Mme. Dagorret si nous devions le faire, suite à quoi nous l'avons géré dans la v1.

Lors de la programmation de la v2, nous avons été confronté à un problème de numéro de diaporama qui était tout le temps à 0 et donc affichait "Diaporama Vide" dans la console.
Dans la méthode "Charger diaporama", l'initialisation du paramètre "_numDiaporamaCourant" n'était pas faite donc il avait pour valeur 0. Or, si ce paramètre est à 0, même si les images sont chargées le lecteur ne pourra pas faire la lecture.
Nous avons donc rajouté "_numDiaporamaCourant +=1" pour incrémenter.

2. **Scénarios**

Scénario nominal : l'utilisateur déroule le diaporama en mode manuel

1. L'utilisateur ouvre la section "Paramètres" du menu
2. Il charge un diaporama
3. Il parcourt le diaporama à l'aide des boutons de l'interface principale
4. Il ouvre la section "Paramètres" du menu
5. Il enlève le diaporama
6. Il ouvre la section "Fichier" du menu
7. Il quitte l'application

Scénario alternatif 1 : l'utilisateur déroule le diaporama en mode automatique

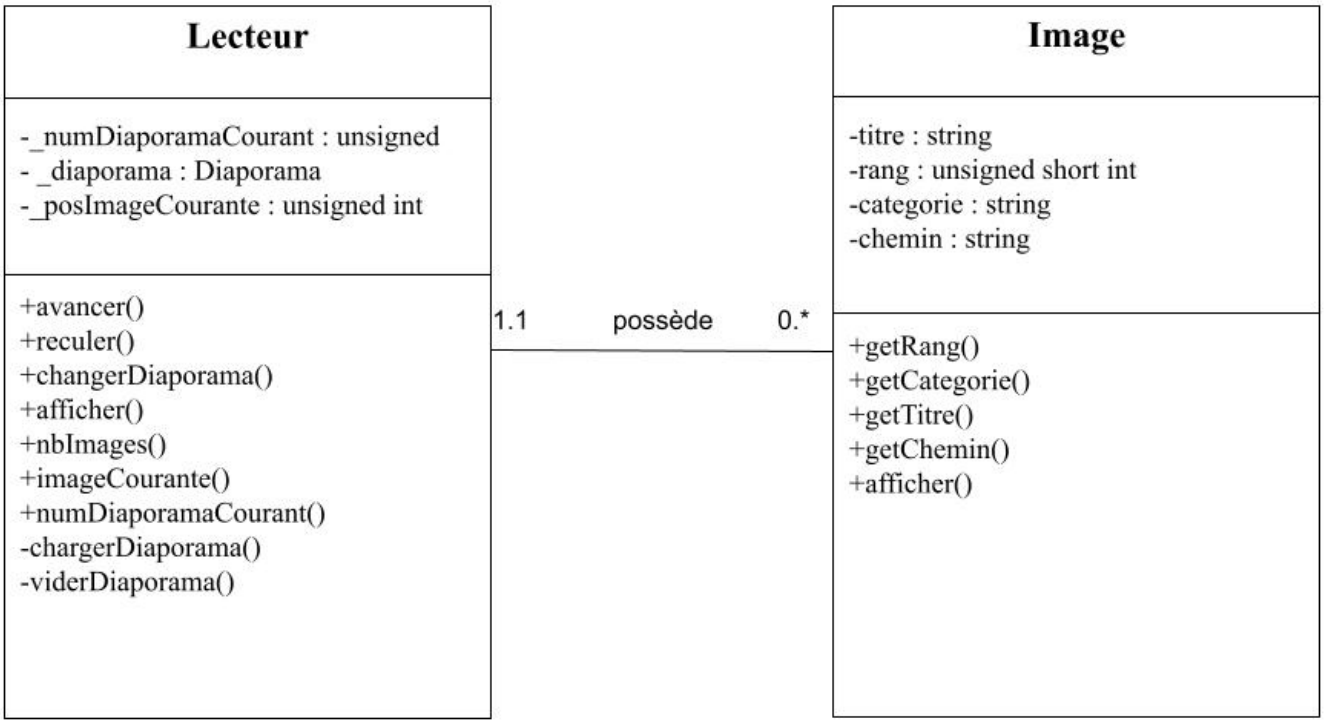
3.A. L'utilisateur passe le mode de déroulement en automatique en cliquant sur "Lancer Diaporama"
3.B. Il arrête le diaporama en cliquant sur "Arrêter Diaporama"
3.C. Retour en 4.

Scénario alternatif 2 : l'utilisateur change la vitesse de défilement

3.B.1. L'utilisateur ouvre la section "Paramètres" du menu
3.B.2. Il change la vitesse de défilement des images
3.B.3. Retour en 3.B.

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.



(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`	`"Pumba"`
`_categorie`	catégorie de l'image (personne, animal, objet)	`string`	`"personne"`
`_chemin`	chemin complet vers le dossier où se trouve l'image	`string`	`"C:\...\image.gif"`

Tableau 2 : Dictionnaire des éléments - Classe Image

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

Tableau 3 : Dictionnaire des éléments - Classe LecteurVue

(c) Dictionnaire des méthodes :

```cpp
#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 4 : Schéma de classes = Classe "image.h"

```cpp
#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();
    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 5 : Schéma de classes = Classe "lecteur.h"

(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

Classe	Description	Valeur(s) en entrée	Résultat(s) attendu(s)	Résultat(s) obtenu(s)
valide n°1	Le chemin d'accès correspond à une image d'extension .gif	"C:\\cartesDisney\\carteDisney_0.gif"	C:\\cartesDisney\\carteDisney_0.gif	C:\\cartesDisney\\carteDisney_0.gif
valide n°2	La catégorie de l'image est contenu dans les valeurs possibles ("personne", "animal" ou "objet")	"animal"	animal	animal
valide n°3	Le chemin d'accès correspond à une image d'extension .png	"C:\\cartesDisney\\carteDisney_0.png"	C:\\cartesDisney\\carteDisney_0.png	C:\\cartesDisney\\carteDisney_0.png
invalide n°1	Le chemin d'accès ne correspond à aucune image	"C:\\cartesDisney\\carteDisney_70.gif"	Echec	Echec
invalide n°2	Le chemin d'accès correspond à un fichier texte	"C:\\cartesDisney\\carteDisney_0.txt"	Echec	Echec
invalide n°3	Le chemin d'accès est vide	""	Echec	Echec
invalide n°4	La catégorie de l'image n'est pas contenue dans les valeurs possibles	"nourriture"	Echec	Echec
invalide n°5	La catégorie de l'image est vide	""	Echec	Echec
invalide n°6	Le titre de l'image est vide	""	Echec	Echec

Version v1 – projet Graphique seul

5. Éléments d'interface

L'interface de notre v0 est composée tout d'abord d'un horizontal layout nommé "centralwidget". Il correspond à la disposition de base des éléments dans notre fenêtre.

A l'intérieur de ce layout nous avons placé un autre horizontal layout de même nom ainsi que deux spacer nommés "horizontalSpacer" et "horizontalSpacer_2" afin que nos éléments graphiques soient centrés dans notre fenêtre.

Nous avons donc ensuite ajouter un vertical layout, possédant le même nom, dans lequel nous avons tout d'abord intégrer un texte browser nommé "AffichageTextePhoto" affichant des informations complémentaires aux images des diaporamas ainsi que l'image qui l'accompagne dans un QFrame nommé "frame".

Ensuite, d'autres éléments comme les labels "lCategorie" et "lTitrePhoto" ont été ajoutés afin d'afficher les informations principales de chaque image.

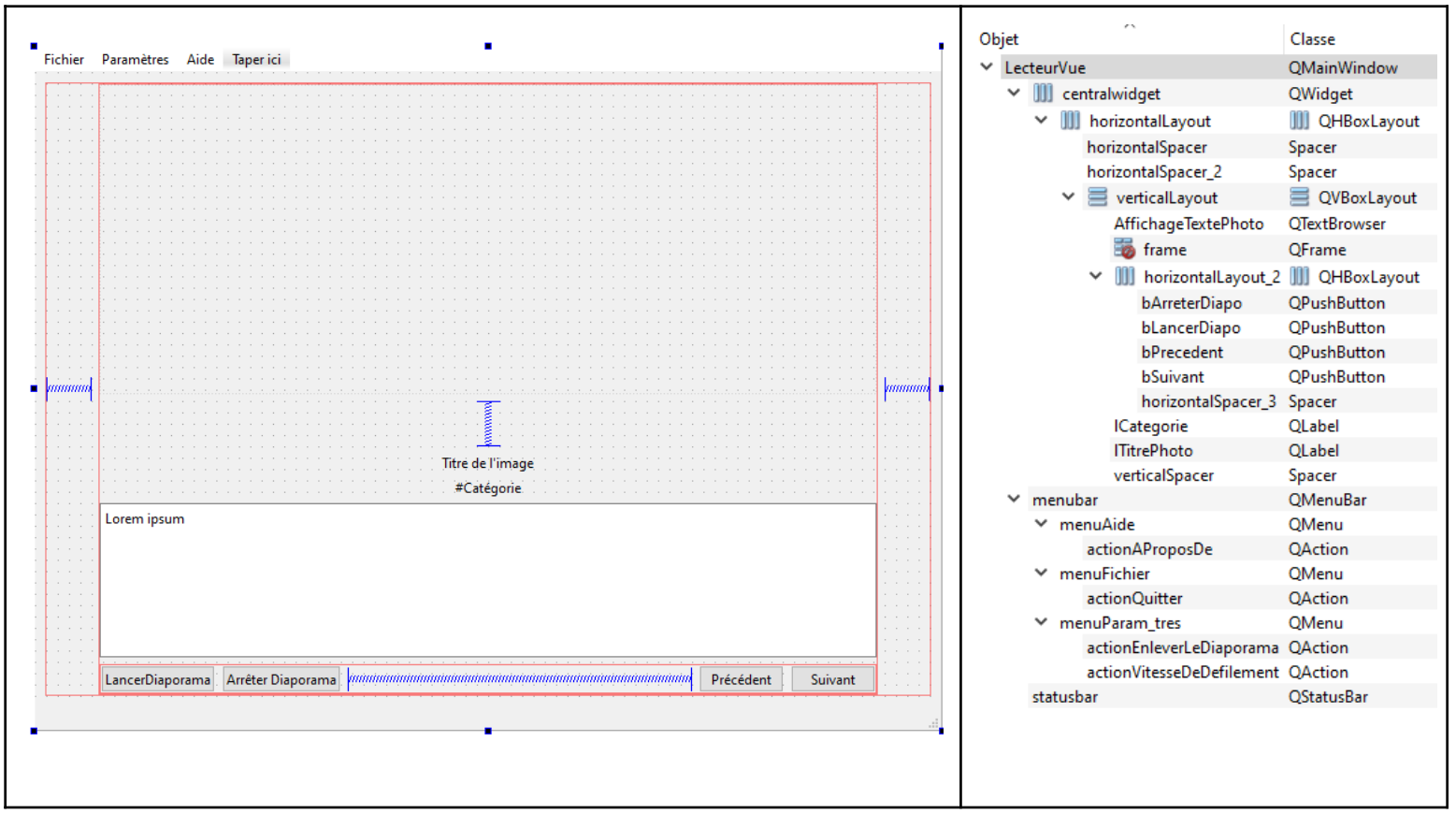
un autre spacer, de nom "verticalSpacer", a été ajouté entre notre QFrame et nos labels pour la lisibilité. Un autre horizontal layout, de nom "horizontalLayout_2", a été positionné ensuite afin d'y placer nos boutons.

Ces boutons appelés "bArreterDiapo", "bLancerDiapo", "bPrecedent" et 'bSuivant" ont ensuite été séparés en deux parties par un horizontal spacer nommé "horizontalSpacer_3".

Enfin, dans notre QMenuBar nommé "menubar", nous avons ajouté 3 QMenu différents appelés "menuAide", "menuFichier" et "menuParametres".

Dans notre "menuAide", nous avons ajouté l'action "actionAProposDe". Dans "menuFichier", on retrouve l'action "actionQuitter" et enfin dans le menu "menuParametres" il y a les actions "actionEnleverDiaporama" et "actionVitesseDeDefilement".

On obtient donc une interface graphique ressemblant à ceci :



6. Implémentation et tests

6.1 Implémentation

Liste et rôle des fichiers de cette version :

lecteurvue.h	Spécification de la classe Lecteur Contient la description des méthodes de la classe LecteurVue
lecteurvue.cpp	Corps de la classe Lecteur
main.cpp	Teste les méthodes de la classe Lecteur

Il n'y a pas de nouveaux éléments, mais les fichiers image.h et .cpp ont été fusionnés au lecteurvue.h et .cpp

Remarques sur l'implémentation :

Nous avons implémenté un slot pour chaque élément d'interface qui s'active lors de la réception des signaux de base intégrés dans la classe QObject (clicked, triggered). Ces slots affichent un message dans la console lorsque l'on interagit avec les éléments graphiques tels que les boutons et la barre de menu.

6.2 Test

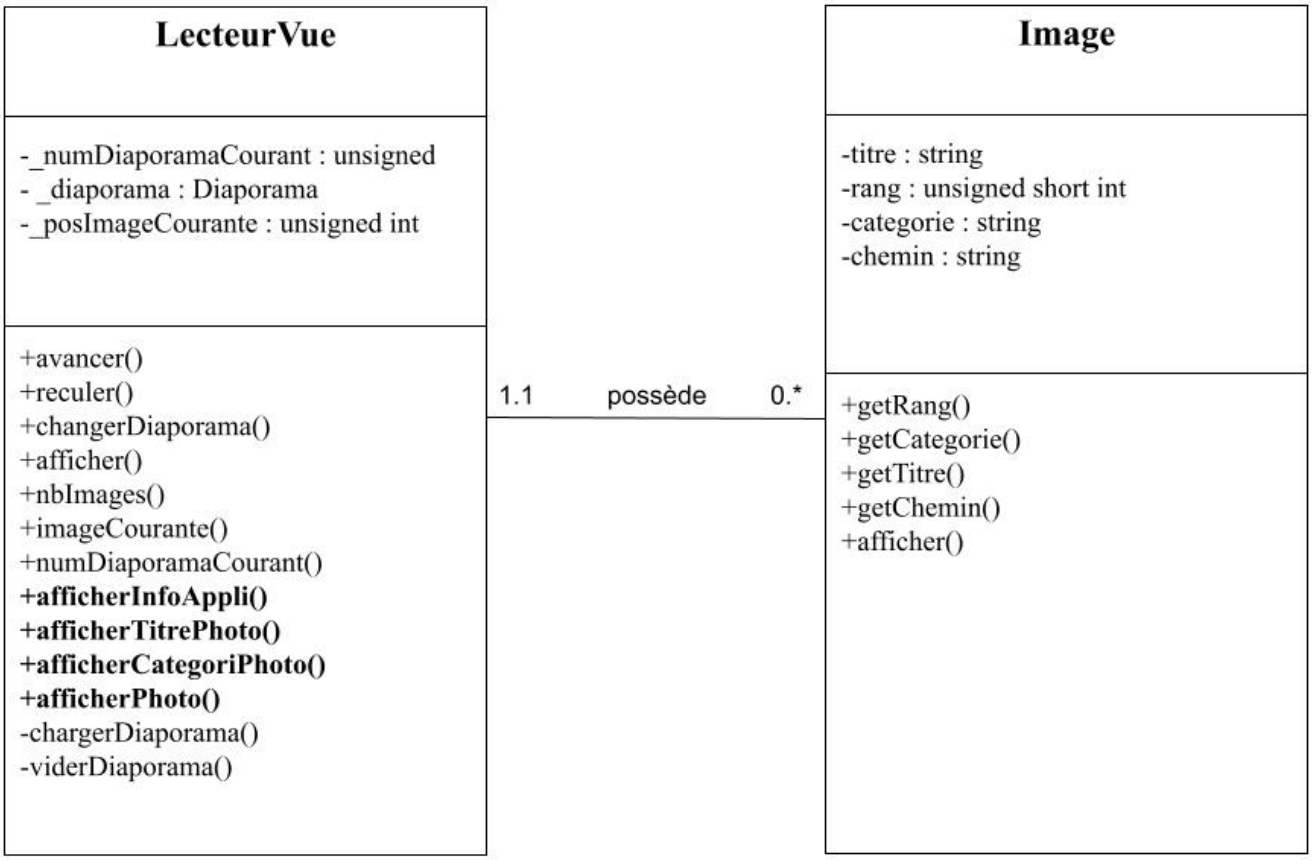
Test avec le programme main.cpp

Slots	Description	Résultat(s) attendu(s)	Résultat(s) obtenu(s)
suivant()	Affiche un message lorsque le bouton *Suivant* est cliqué	"J'affiche l'image suivant"	"J'affiche l'image suivant"
precedent()	Affiche un message lorsque le bouton *Précédent* est cliqué	"J'affiche l'image précédente"	"J'affiche l'image précédente"
demarrerDiapo()	Affiche un message lorsque le bouton *Démarrer Diaporama* est cliqué	"Je démarre le diaporama"	"Je démarre le diaporama"
arreterDiapo()	Affiche un message lorsque le bouton *Arrêter Diaporama* est cliqué	"J'arrete le diaporama"	"J'arrete le diaporama"
fermerFenetre()	Affiche un message lorsque le bouton *Quitter* est cliqué	"Je ferme la fenêtre"	"Je ferme la fenêtre"
aProposDe()	Affiche un message lorsque le bouton *A propos de* est cliqué	"J'affiche les informations de l'application"	"J'affiche les informations de l'application"
vitesseDefilement()	Affiche un message lorsque le bouton *Vitesse de défilement* est cliqué	"Je modifie la vitesse de défilement du diaporama"	"Je modifie la vitesse de défilement du diaporama"
chargerDiapo()	Affiche un message lorsque le bouton *Charger le diaporama* est cliqué	"Je charge le diaporama"	"Je charge le diaporama"
enleverDiapo()	Affiche un message lorsque le bouton *Enlever le diaporama* est cliqué	"J'enlève le diaporama"	"J'enlève le diaporama"

Version v2 –

7. Diagramme de classes (UML)

Les changements par rapport à la v0 ont été mis en **gras**.



8. Comportement de l'application

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

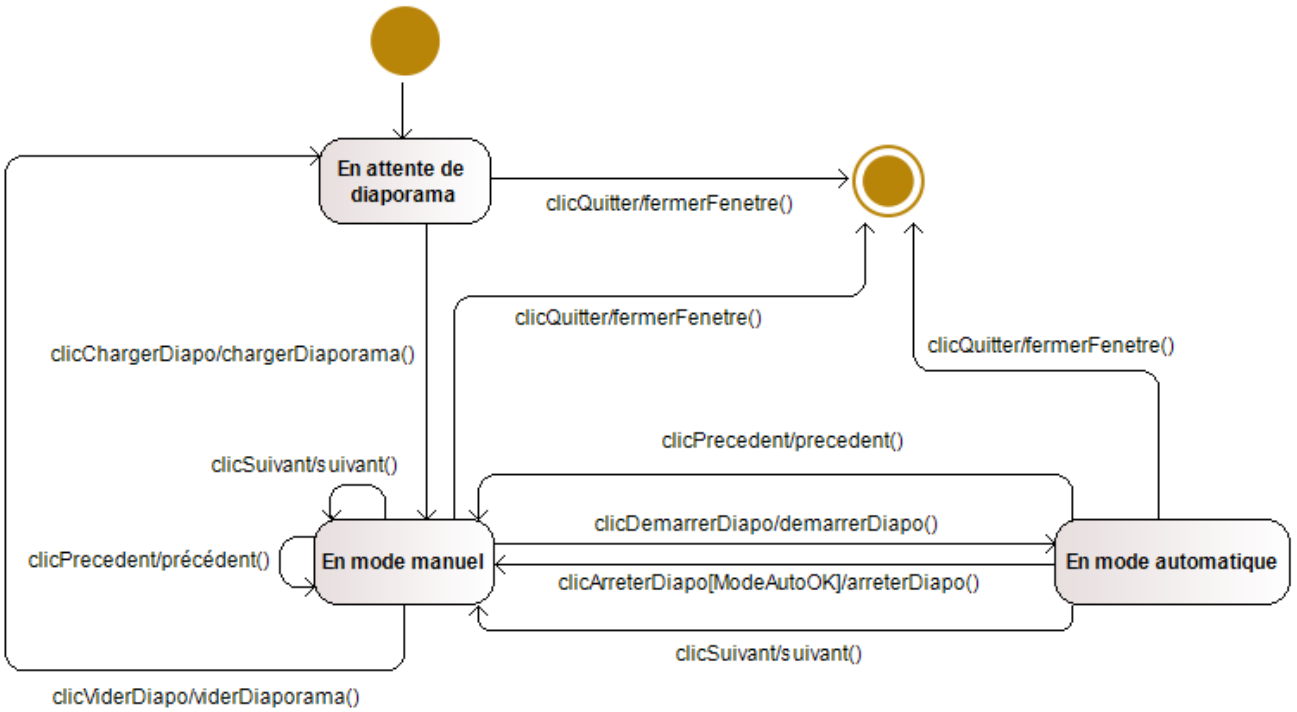


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v2

8.2 Dictionnaire des états, événements et Actions (v2)

Dictionnaire des états du diaporama

nomEtat	*Signification*
`En attente diaporama`	Le lecteur attend que l'utilisateur charge un diaporama ou ferme la fenêtre
`En mode manuel`	Le lecteur défile les images du diaporama en mode manuel, c'est-à-dire à chaque action de l'utilisateur sur les boutons *suivant* et *précédent*
`En mode automatique`	Le lecteur défile les images du diaporama en mode automatique, c'est-à-dire qu'il change l'image affichée toutes les x secondes

Tableau 2 : États du lecteur de diaporamas – v2

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
`clicChargerDiapo`	L'utilisateur clique sur le bouton *Charger le diaporama* avec le clic gauche de la souris
`clicSuivant`	L'utilisateur clique sur le bouton *Suivant* avec le clic gauche de la souris
`clicPrecedent`	L'utilisateur clique sur le bouton *Précédent* avec le clic gauche de la souris
`clicDemarrerDiapo`	L'utilisateur clique sur le bouton *Démarrer le diaporama* avec le clic gauche de la souris
`clicArreterDiapo`	L'utilisateur clique sur le bouton *Arrêter le diaporama* avec le clic gauche de la souris
`clicViderDiapo`	L'utilisateur clique sur le bouton *Vider le diaporama* avec le clic gauche de la souris
`clicQuitter`	L'utilisateur clique sur le bouton *Quitter* avec le clic gauche de la souris

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

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
`chargerDiaporama()`	Le lecteur charge le diaporama sélectionné
`suivant()`	Le lecteur défile l'image et affiche la suivante
`précédent()`	Le lecteur défile l'image et affiche la précédente
`demarrerDiapo()`	Le lecteur commence à défiler les images en mode automatique
`arreterDiapo()`	Le lecteur arrête de défiler les images en mode automatique
`viderDiaporama()`	Le lecteur retire le diaporama chargé
`fermerFenetre()`	La fenêtre du lecteur se ferme

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

8.3 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_le_diaporama	bSuivant	bPrecedent	bLancerDiapo	bArreterDiapo	actionEnleverLeDiaporama	actionQuitter
Événement □ *nomEtat*	clicChargerDiapo	clicSuivant	clicPrecedent	clicDemarrerDiapo	clicArreterDiapo	clicViderDiapo	clicQuitter
En attente diaporama	En mode manuel	//	//	//	//	//	Fermé
En mode manuel	//	En mode manuel	En mode manuel	En mode automatique	//	En attente de diaporama	Fermé
En mode automatique	//	En mode manuel	En mode manuel	//	[ModeAutoOK] En mode manuel	//	Fermé

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

L'état *Fermé* correspond à la fermeture de la fenêtre, nous avons donc choisi de ne pas la représenter car il s'agit de l'état final.

9. Implémentation et tests

9.1 Implémentation (v2)

Liste et rôle des fichiers de cette version, les nouveaux éléments ont été mis en gras :

lecteurvue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas Contient la description des méthodes de la classe LecteurVue
lecteurvue.cpp	Corps de la classe LecteurVue
lecteurvue.ui	**Fichier du dessin de l'interface du lecteur réalisé par QtDesigner**
image.h	**Spécification de la classe Image** **Contient la description des méthodes de la classe Image**
image.cpp	**Corps de la classe Image**
main.cpp	Test des méthodes de la classe LecteurVue

Remarques sur l'implémentation :

Nous avons implémenté un slot pour chacun des éléments d'interface afin que l'utilisateur puisse interagir avec eux. Ces slots affichent à la fois un message dans la console décrivant l'action déclenchée par celui-ci et mettent également à jour les éléments d'interfaces correspondant. Les signaux que nous avons utilisés proviennent de la classe QObject proposée par QT.

Liste des slots de la classe LecteurVue :

```
void precedent();
void suivant();
void demarrerDiapo();
void arreterDiapo();
void fermerFenetre();
void aProposDe();
void vitesseDefilement();
void enleverDiapo();
void chargementDiaporama();
```

9.2 Tests (v2)

Test avec le programme main.cpp

Classe	Description	Valeur(s) en entrée	Résultat(s) attendu(s)	Résultat(s) obtenu(s)
valide n°1	Le chemin d'accès correspond à une image d'extension .gif	"C:\\cartesDisney\\carteDisney_0.gif"	Affichage de l'image dans l'interface	Affichage de l'image dans l'interface
valide n°2	La catégorie de l'image est contenu dans les valeurs possibles ("personne", "animal" ou "objet")	"animal"	Affichage de la catégorie dans l'interface	Affichage de la catégorie dans l'interface
valide n°3	Le chemin d'accès correspond à une image d'extension .png	"C:\\cartesDisney\\carteDisney_0.png"	Affichage de l'image dans l'interface	Affichage de l'image dans l'interface

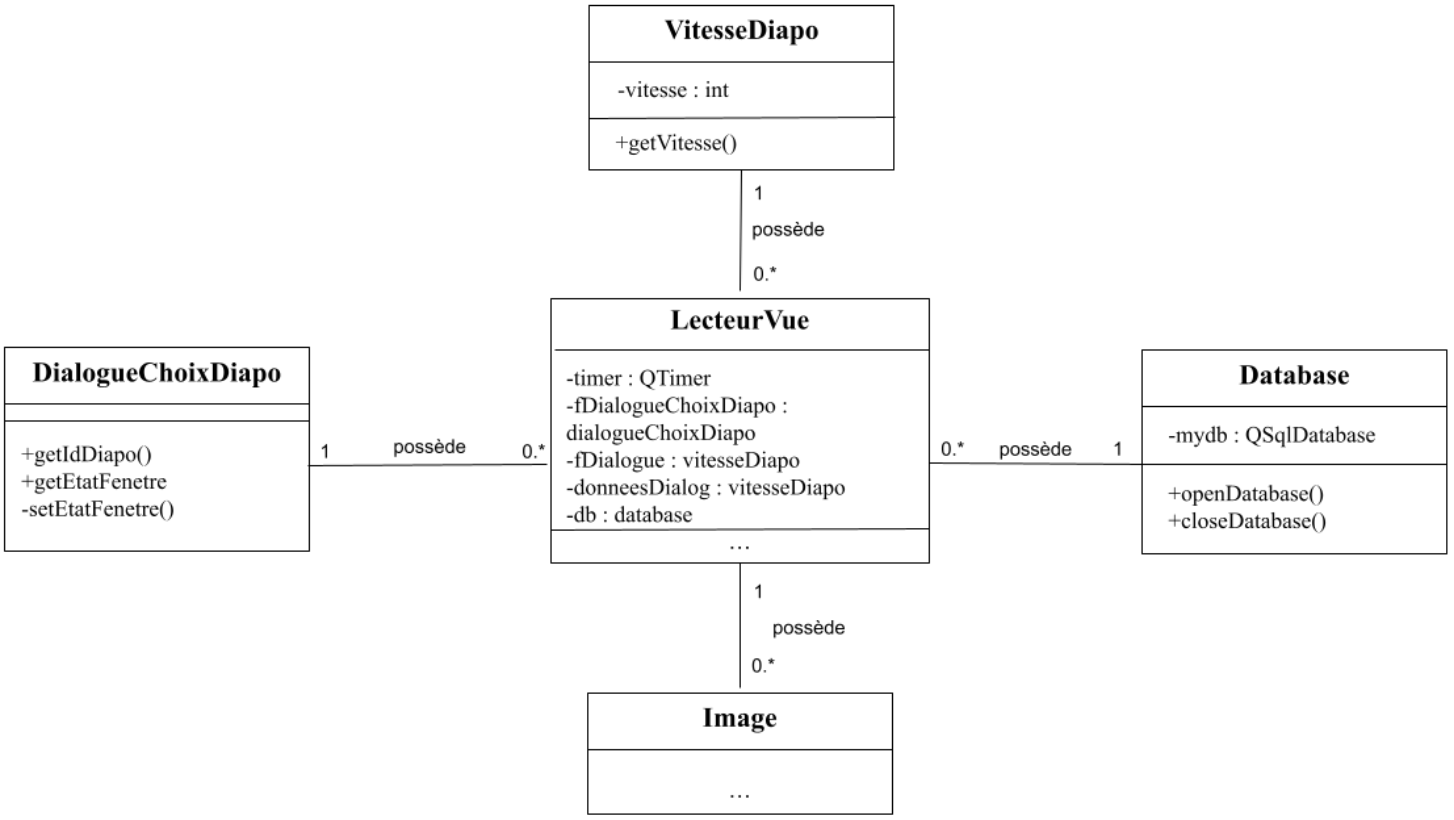
valide n°4	L'utilisateur clique sur le bouton *Suivant*	Le bouton *bSuivant* est activé	Défilement à l'image suivante et affichage des informations dans la console	Défilement à l'image suivante et affichage des informations dans la console
valide n°5	L'utilisateur clique sur le bouton *Précédent*	Le bouton *bPrecedent* est activé	Retour à l'image précédente et affichage des informations dans la console	Retour à l'image précédente et affichage des informations dans la console
valide n°6	L'utilisateur clique sur le bouton *Démarrer Diaporama*	Le bouton *bLancerDiapo* est activé	Changement du statut en mode automatique et affichage d'un message dans la console	Changement du statut en mode automatique et affichage d'un message dans la console
valide n°7	L'utilisateur clique sur le bouton *Arrêter Diaporama*	Le bouton *bArreterDiapo* est activé	Changement du statut du diaporama en mode manuel et affichage d'un message dans la console	Changement du statut du diaporama en mode manuel et affichage d'un message dans la console
valide n°8	L'utilisateur clique sur le bouton *Quitter*	L'action *actionQuitter* est appelée	Affichage d'un message dans la console et fermeture de la fenêtre	Affichage d'un message dans la console et fermeture de la fenêtre
valide n°9	L'utilisateur clique sur le bouton *A propos de*	L'action *actionAProposDe* est appelée	Affichage d'un message contenant la version de l'application, la date de création et les auteurs	Affichage d'un message contenant la version de l'application, la date de création et les auteurs
valide n°10	L'utilisateur clique sur le bouton *Vitesse de défilement*	L'action *actionVitesseDeDefilement* est appelée	Affichage d'un message dans la console	Affichage d'un message dans la console
valide n°11	L'utilisateur clique sur le bouton *Enlever le diaporama*	L'action *actionEnleverLeDiaporama* est appelée	Affichage d'un message dans la console et vide le diaporama	Affichage d'un message dans la console et vide le diaporama
valide n°12	L'utilisateur clique sur le bouton *Charger le diaporama*	L'action *actionCharger_le_diaporama* est appelée	Affichage d'un message dans la console et changement du diaporama	Affichage d'un message dans la console et changement du diaporama
invalide n°1	Le chemin d'accès ne correspond à aucune image	"C:\\cartesDisney\\carteDisney_70.gif"	Echec	Echec

invalide n°2	Le chemin d’accès correspond à un fichier texte	"C:\\cartesDisney\\carteDisney_0.txt"	Echec	Echec
invalide n°3	Le chemin d’accès est vide	“”	Echec	Echec
invalide n°4	La catégorie de l’image n’est pas contenue dans les valeurs possibles	“nourriture”	Echec	Echec
invalide n°5	La catégorie de l’image est vide	“”	Echec	Echec
invalide n°6	Le titre de l’image est vide	“”	Echec	Echec

Version v5 –

10. Diagramme de classes (UML)

La classe *Image* est identique à la v2 et la classe *LecteurVue* possède les mêmes méthodes de la v2 avec en plus celles rajoutées sur ce diagramme.



11. Comportement de l'application

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

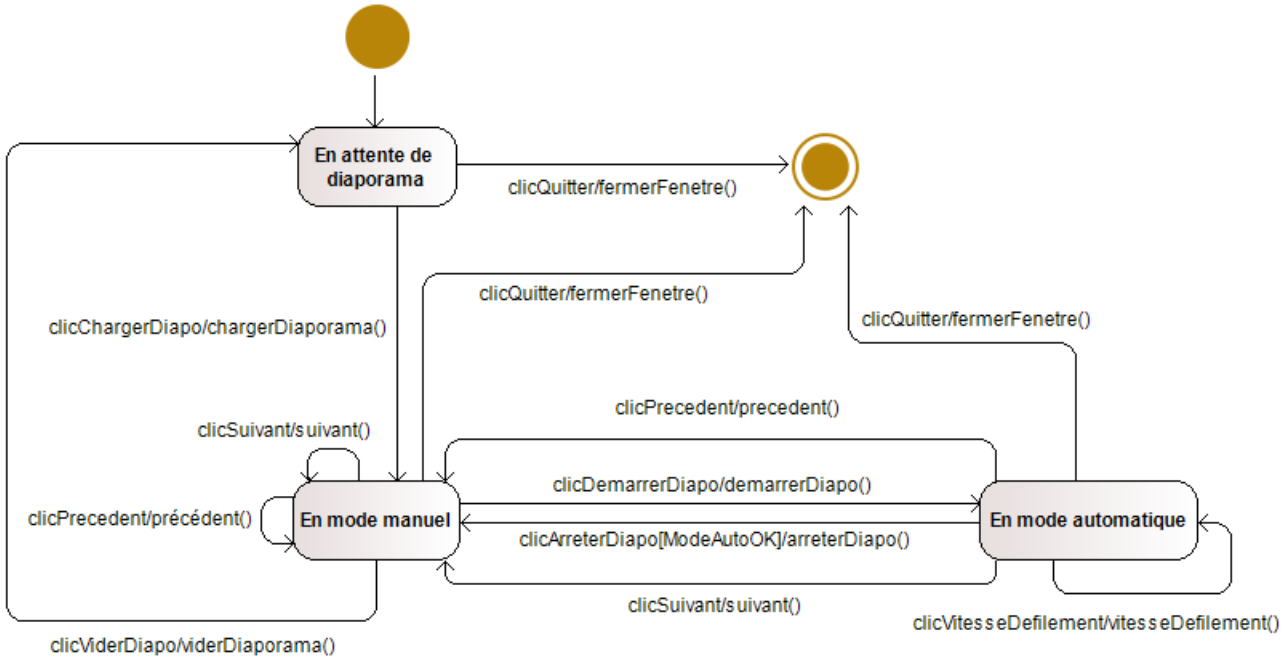


Figure 9 : Diagramme états-transitions du lecteur de diaporamas – v5

11.2 Dictionnaire des états, événements et Actions (v5)

Dictionnaire des états du diaporama

nomEtat	*Signification*
`En attente diaporama`	Le lecteur attend que l'utilisateur charge un diaporama ou ferme la fenêtre
`En mode manuel`	Le lecteur défile les images du diaporama en mode manuel, c'est-à-dire à chaque action de l'utilisateur sur les boutons *suivant* et *précédent*
`En mode automatique`	Le lecteur défile les images du diaporama en mode automatique, c'est-à-dire qu'il change l'image affichée toutes les x secondes

Tableau 2 : États du lecteur de diaporamas – v5

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*
`clicChargerDiapo`	L'utilisateur clique sur le bouton *Charger le diaporama* avec le clic gauche de la souris
`clicSuivant`	L'utilisateur clique sur le bouton *Suivant* avec le clic gauche de la souris
`clicPrecedent`	L'utilisateur clique sur le bouton *Précédent* avec le clic gauche de la souris
`clicDemarrerDiapo`	L'utilisateur clique sur le bouton *Démarrer le diaporama* avec le clic gauche de la souris
`clicArreterDiapo`	L'utilisateur clique sur le bouton *Arrêter le diaporama* avec le clic gauche de la souris
`clicViderDiapo`	L'utilisateur clique sur le bouton *Vider le diaporama* avec le clic gauche de la souris
`clicQuitter`	L'utilisateur clique sur le bouton *Quitter* avec le clic gauche de la souris
`clicVitesseDefilement`	L'utilisateur clique sur le bouton *Vitesse de défilement* avec le clic gauche de la souris

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

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*
`chargerDiaporama()`	Le lecteur charge le diaporama sélectionné
`suivant()`	Le lecteur défile l'image et affiche la suivante
`précédent()`	Le lecteur défile l'image et affiche la précédente
`demarrerDiapo()`	Le lecteur commence à défiler les images en mode automatique
`arreterDiapo()`	Le lecteur arrête de défiler les images en mode automatique
`viderDiaporama()`	Le lecteur retire le diaporama chargé
`fermerFenetre()`	La fenêtre du lecteur se ferme
`vitesseDefilement()`	La fenêtre de dialogue du choix de la vitesse de défilement s'ouvre

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

11.3 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_le_diaporama	bSuivant	bPrecedent	bLancerDiapo	bArreterDiapo	actionEnleverLeDiaporama	actionVitesseDefilement	actionQuitter
Événement nomEtat	clicChargerDiapo	clicSuivant	clicPrecedent	clicDemarrerDiapo	clicArreterDiapo	clicViderDiapo	clicVitesseDefilement	clicQuitter
En attente diaporama	En mode manuel	//	//	//	//	//	//	Fermé
En mode manuel	//	En mode manuel	En mode manuel	En mode automatique	//	En attente de diaporama	//	Fermé
En mode automatique	//	En mode manuel	En mode manuel	//	[ModeAutoOK] En mode manuel	//	En mode automatique	Fermé

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

12. Implémentation et tests

12.1 Implémentation (v5)

Liste et rôle des fichiers de cette version, les nouveaux éléments ont été mis en gras :

lecteurvue.h	Spécification de la classe graphique Qt contenant l'interface du lecteur de diaporamas Contient la description des méthodes de la classe LecteurVue
lecteurvue.cpp	Corps de la classe LecteurVue.
lecteurvue.ui	Fichier du dessin de l'interface du lecteur réalisé par QtDesigner
image.h	Spécification de la classe Image Contient la description des méthodes de la classe Image
image.cpp	Corps de la classe Image
database.h	**Spécification de la classe database** **Contient la description des méthodes de la classe database**
database.cpp	**Corps de la classe database**
dialoguechoixdiapo.h	**Spécification de la classe dialogueChoixDiapo** **Contient la description des méthodes de la classe dialogueChoixDiapo**
dialoguechoixdiapo.cpp	**Corps de la classe dialogueChoixDiapo**
dialoguechoixdiapo.ui	**Fichier du dessin de l'interface pour le choix de diaporama réalisé par QtDesigner**
vitessediapo.h	**Spécification de la classe vitesseDiapo** **Contient la description des méthodes de la classe vitesseDiapo**
vitessediapo.cpp	**Corps de la classe vitesseDiapo**
vitessediapo.ui	**Fichier du dessin de l'interface pour le choix de la vitesse de diaporama réalisé par QtDesigner**
main.cpp	Programme principal

Remarques sur l'implémentation :

En plus des signaux et slots de la version précédente, nous avons implémenté des nouveaux slots pour les classes LecteurVue et dialoguechoixDiapo.

Liste des slots de la classe LecteurVue, les nouveaux éléments ont été mis en gras :

```
void precedent();
void suivant();
void demarrerDiapo();
void arreterDiapo();
void fermerFenetre();
void aProposDe();
void vitesseDefilement();
void enleverDiapo();
void chargementDiaporama();
void finTimer();
```

Liste des slots de la classe dialogueChoixDiapo :

```
void setFenetreConfirme();
void setFenetreRejete();
```

12.2 Tests (v5)

Test avec le programme main.cpp

Classe	Description	Valeur(s) en entrée	Résultat(s) attendu(s)	Résultat(s) obtenu(s)
valide n°1	Le chemin d'accès correspond à une image d'extension .gif	"C:\\cartesDisney\\carteDisney_0.gif"	Affichage de l'image dans l'interface	Affichage de l'image dans l'interface
valide n°2	La catégorie de l'image est contenu dans les valeurs possibles ("personne", "animal" ou "objet")	"animal"	Affichage de la catégorie dans l'interface	Affichage de la catégorie dans l'interface
valide n°3	Le chemin d'accès correspond à une image d'extension .png	"C:\\cartesDisney\\carteDisney_0.png"	Affichage de l'image dans l'interface	Affichage de l'image dans l'interface
valide n°4	L'utilisateur clique sur le bouton *Suivant*	Le bouton *bSuivant* est activé	Défilement à l'image suivante et affichage des informations dans la console	Défilement à l'image suivante et affichage des informations dans la console
valide n°5	L'utilisateur clique sur le bouton *Précédent*	Le bouton *bPrecedent* est activé	Retour à l'image précédente et affichage des informations dans la console	Retour à l'image précédente et affichage des informations dans la console
valide n°6	L'utilisateur clique sur le bouton *Démarrer Diaporama*	Le bouton *bLancerDiapo* est activé	Changement du statut en mode automatique et affichage d'un message dans la console	Changement du statut en mode automatique et affichage d'un message dans la console
valide n°7	L'utilisateur clique sur le bouton *Arrêter Diaporama*	Le bouton *bArreterDiapo* est activé	Changement du statut du diaporama en mode manuel et affichage d'un message dans la console	Changement du statut du diaporama en mode manuel et affichage d'un message dans la console
valide n°8	L'utilisateur clique sur le bouton *Quitter*	L'action *actionQuitter* est appelée	Affichage d'un message dans la console et fermeture de la fenêtre	Affichage d'un message dans la console et fermeture de la fenêtre
valide n°9	L'utilisateur clique sur le bouton *A propos de*	L'action *actionAProposDe* est appelée	Affichage d'un message contenant la version de l'application, la date de création et les auteurs	Affichage d'un message contenant la version de l'application, la date de création et les auteurs
valide n°10	L'utilisateur clique sur le bouton *Vitesse de*	L'action *actionVitesseDeDefi*	Ouverture d'une fenêtre modale pour	Ouverture d'une fenêtre modale pour

	défilement	*lement* est appelée	saisir la valeur et affichage d'un message dans la console	saisir la valeur et affichage d'un message dans la console
valide n°11	L'utilisateur clique sur le bouton *Enlever le diaporama*	L'action *actionEnleverLeDiaporama* est appelée	Affichage d'un message dans la console et vide le diaporama	Affichage d'un message dans la console et vide le diaporama
valide n°12	L'utilisateur clique sur le bouton *Charger le diaporama*	L'action *actionCharger_le_diaporama* est appelée	Affichage d'un message dans la console et changement du diaporama	Affichage d'un message dans la console et changement du diaporama
valide n°13	La valeur de la vitesse de défilement		Les images défilent toutes les	Les images défilent toutes les
invalide n°1	Le chemin d'accès ne correspond à aucune image	"C:\\cartesDisney\\carteDisney_70.gif"	Echec	Echec
invalide n°2	Le chemin d'accès correspond à un fichier texte	"C:\\cartesDisney\\carteDisney_0.txt"	Echec	Echec
invalide n°3	Le chemin d'accès est vide	""	Echec	Echec
invalide n°4	La catégorie de l'image n'est pas contenue dans les valeurs possibles	"nourriture"	Echec	Echec
invalide n°5	La catégorie de l'image est vide	""	Echec	Echec
invalide n°6	Le titre de l'image est vide	""	Echec	Echec
invalide n°7	La valeur de la vitesse de défilement est une chaîne de caractère	"trois"	Echec	Echec
invalide n°8	La vitesse de défilement est une valeur négative	"-25"	Echec	Echec
invalide n°9	La vitesse de défilement est un nombre à virgule	"2.5"	Echec	Echec

13. Bilan

Nous avons passé chacun 25 heures en séance et 3 heures chez nous, ce qui fait un total de 28 heures par personne, soit 84 heures au total sur cette SAE.

Pendant ce temps, nous avons approfondi notre apprentissage de Qt, travaillé sur la réalisation des jeux d'essais, élaboré le diagramme d'état-transition et les dictionnaires associés, ainsi que conçu les diagrammes de classes. Nous avons rencontré des difficultés pour comprendre certaines parties du sujet, notamment l'objectif des différentes versions, mais nous avons pu éclaircir ces points en posant des questions aux professeurs lors des séances encadrées. Il a également été difficile de s'adapter au nouveau modèle du diagramme d'état-transition avec les dictionnaires, mais nous nous sommes appuyés sur la documentation distribuée pour le réaliser.

Malgré quelques explications parfois ambiguës, nous trouvons que le sujet de la SAE est intéressant car il nous permet d'appliquer nos connaissances à un projet concret et ludique.

Le lien du Git pour retrouver le projet complet et les versions réalisées :
https://github.com/Solenelina/S2.01-DevApp

Tableau d'analyse

Versions	Personnes	Temps passé sur...		J'ai appris à...	J'ai aimé/pas aimé...	J'ai trouvé difficile de...	J'aurai pu mieux faire sur...	Je peux améliorer...
		conception	code					
v0	Anthony	1h	1h30	utiliser des vecteurs	élaborer les classes	proggrammer le tri des images	le temps de programmation de cette version	-
	Gauthier	0	0	-	-	-	-	-
	Solène	1h	1h	utiliser le type vecteur	J'ai aimé utiliser des classes et découvrir le type vecteur	programmer le tri des images du diaporama	le temps passé à programmer cette version	-
Versions	**Personnes**	**Temps passé sur...**		**J'ai appris à...**	**J'ai aimé/pas aimé...**	**J'ai trouvé difficile de...**	**J'aurai pu mieux faire sur...**	**Je peux améliorer...**
		conception	**code**					
v1	Anthony	30min	15min	-	J'ai réfléchir sur la conception de l'interface de l'application	-	-	-
	Gauthier	1h30	30min	à utiliser les signals de la barre de menu	J'ai faire la conception de l'interface	mettre les layout correctement	le versionning des versions (erreur sur le tire ou oublie de fonctionnalités	changer bouton Catégorie pour les trier et afficher message dans la barre de statut
	Solène	30min	15min	à utiliser les signaux et slots	J'ai aimé réfléchir sur la conception de l'interface de l'application et faire du brainstorming	penser à tous les éléments présents et tout détailler pour faciliter la conception graphique	-	La disposition des différents éléments d'interface

Versions	Personnes	Temps passé sur...		J'ai appris à...	J'ai aimé/pas aimé...	J'ai trouvé difficile de...	J'aurai pu mieux faire sur...	Je peux améliorer...
		conception	code					
v2	Anthony	2h	30min	-	Je n'ai pas aimé faire les tests	-	le diagramme d'état-transition	la gestion du temps
	Gauthier	30min	4h30	à utiliser la barre de statut	J'ai aimé coder cette version et la voir fonctionner	garder les idées claires sur ce qu'il y avait à faire	la gestion de mes émotions : le fait de ne pas réussir m'a énervé et j'ai passé plus de temps que j'aurai dû comme j'étais perdu	-
	Solène	1h30	1h	à réaliser les dictionnaires d'états/évènements, et utiliser les actions du menu	J'ai pas aimé réaliser les jeux d'essais, mais j'ai aimé implémenter les signaux/slots	-	le diagramme d'état-transition	-
Versions	**Personnes**	**Temps passé sur...**		**J'ai appris à...**	**J'ai aimé/pas aimé...**	**J'ai trouvé difficile de...**	**J'aurai pu mieux faire sur...**	**Je peux améliorer...**
		conception	**code**					
v3	Anthony	30min	15min	utiliser un timer	-	-	-	-
	Gauthier	30min	1h	utiliser un timer	Utiliser un timer et le gérer pour son arrêt	-	L'attention porté aux chemins des images	-
	Solène	20min	30min	utiliser un timer et automatiser le défilement des images	J'ai aimé voir le défilement automatique fonctionner	-	-	-
Versions	**Personnes**	**Temps passé sur...**		**J'ai appris à...**	**J'ai aimé/pas aimé...**	**J'ai trouvé difficile de...**	**J'aurai pu mieux faire sur...**	**Je peux améliorer...**
		conception	**code**					

v4	Anthony	30min	30min	implémenter la fenêtre de dialogue	Je n'ai pas aimé implémenter la fenêtre de dialogue	-	-	-
	Gauthier	30min	1h15	a verifier que l'utilisateur a effectué une saisie dans une fenetre de dialogue	-	De comprendre comment fonctionner la vérification d'une saisie d'une fenetre de dialogue	Les relectures et vérifications pour vérifier que rien ne bug ou tout soit fonctionnel	-
	Solène	30min	15min	créer une fenêtre de dialogue et mettre en place une vérification de saisie	-	-	la vérification de la vitesse de défilement	-
Versions	**Personnes**	**Temps passé sur...**		**J'ai appris à...**	**J'ai aimé/pas aimé...**	**J'ai trouvé difficile de...**	**J'aurai pu mieux faire sur...**	**Je peux améliorer...**
		conception	**code**					
v5	Anthony	1h30	30min	-	Je n'ai pas aimé faire l'état-transition	faire les tests	-	-
	Gauthier	30min	4h30	A vérifier qu'une fenetre de dialogue est bien fermé avec des getter et des setter		peupler un tableau à partir des informations issues de la base de données	La façon de choisir le diapo au premier coup, car cela m'a pris plus de temps	-
	Solène	2h	30min		J'ai pas aimé réaliser les jeux d'essais	conceptualiser le diagramme des classes avec les nombreux fichiers		-