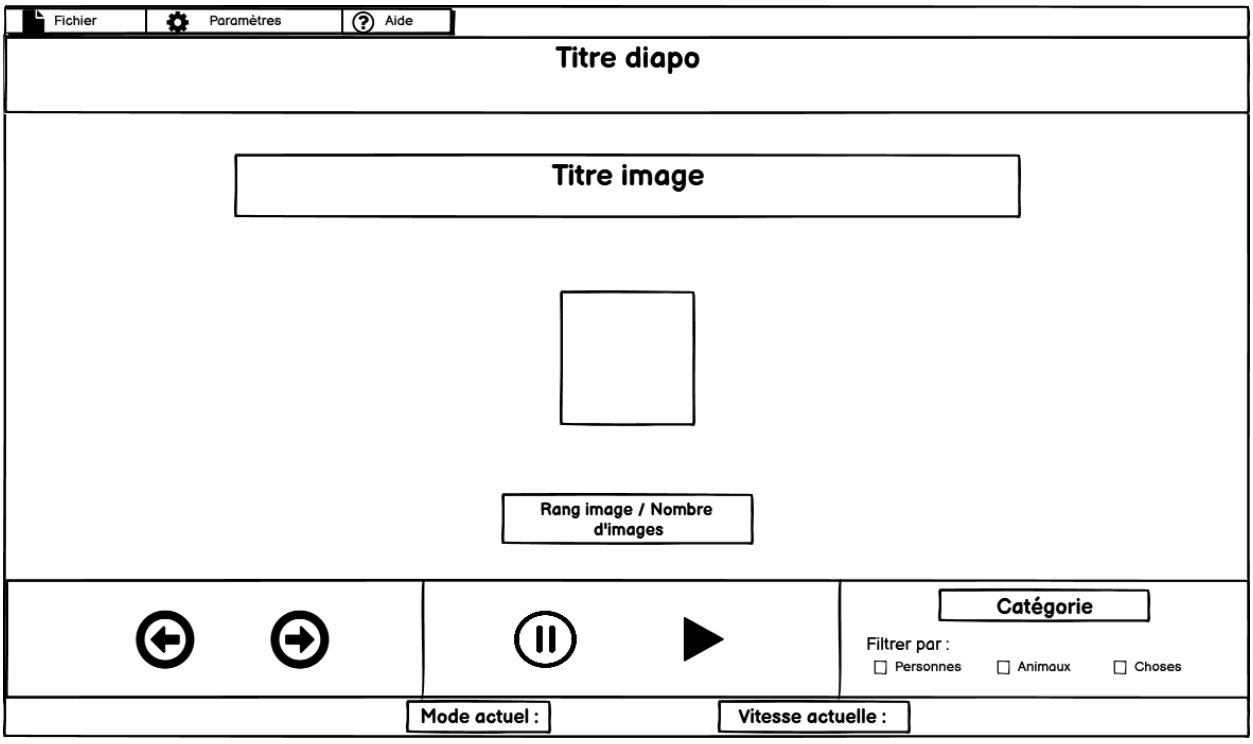
Saé 2.01 – Développement d'une application

Lecteur de diaporamas – Dossier d'Analyse et conception

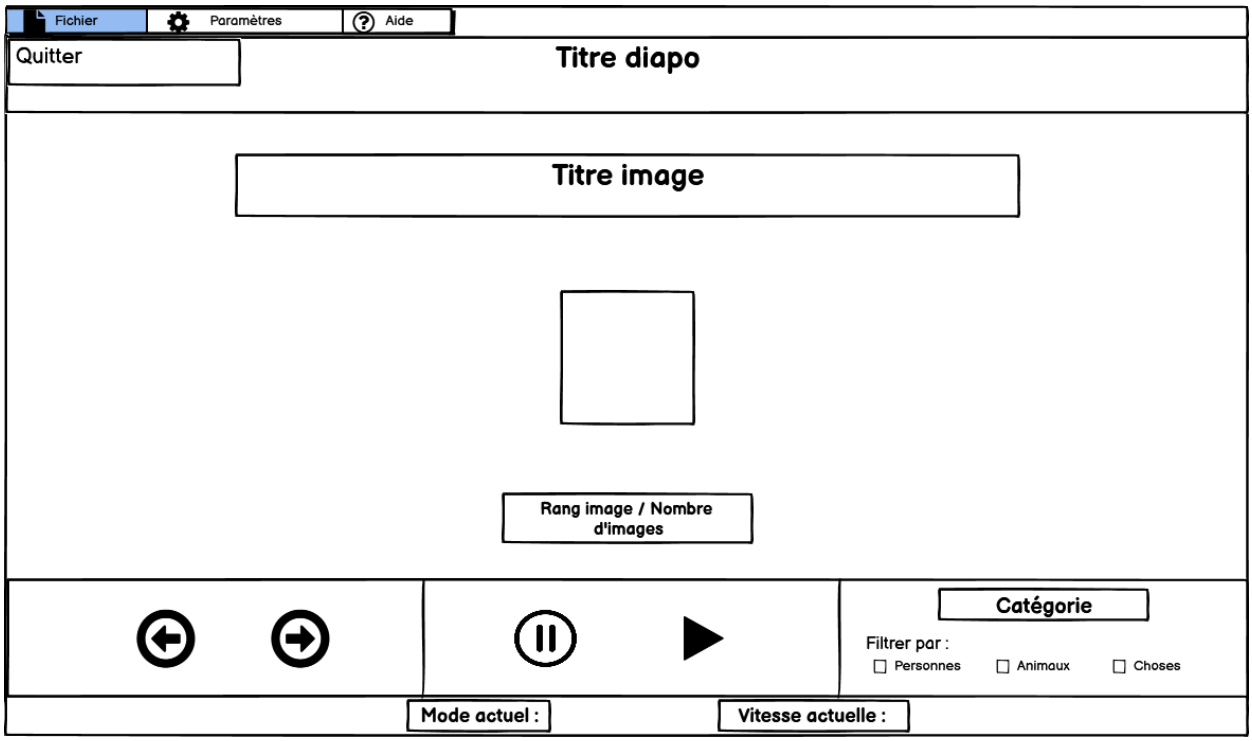
1. Compléments de spécifications externes.

1.1 Maquettes

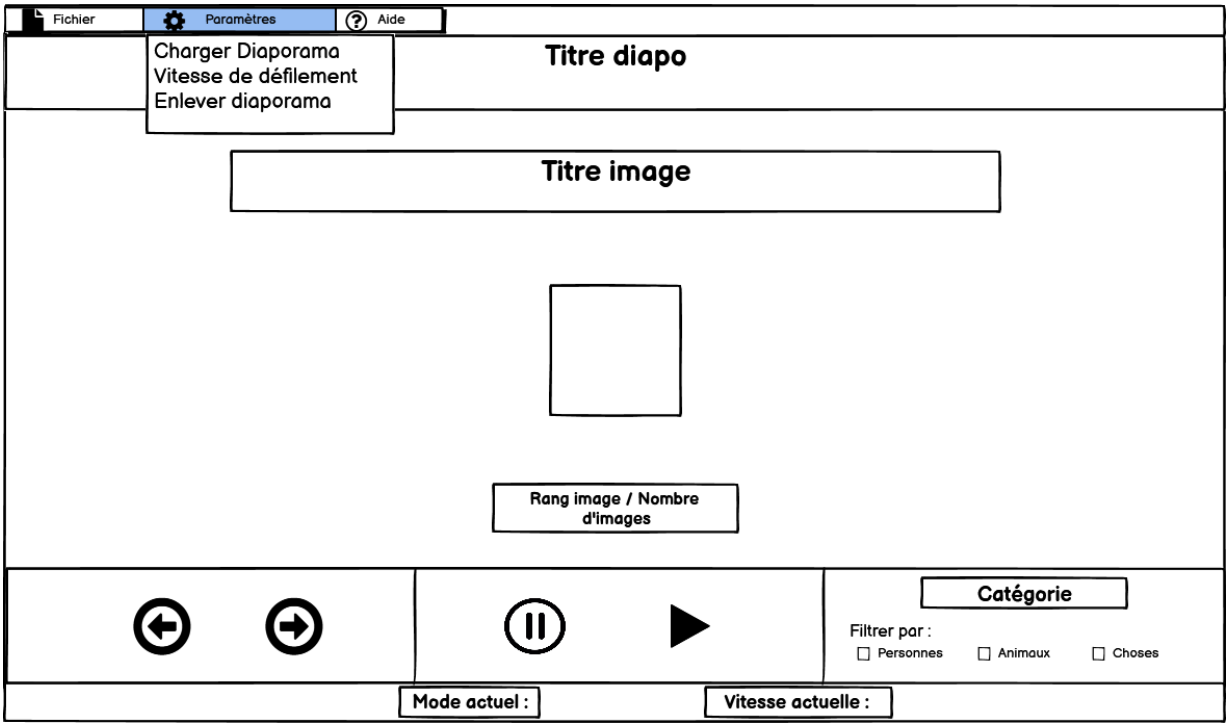
Des maquettes ont été réalisées pour clarifier l'organisation des différents éléments et des différentes informations qui doivent figurer dans les écrans du diaporama :



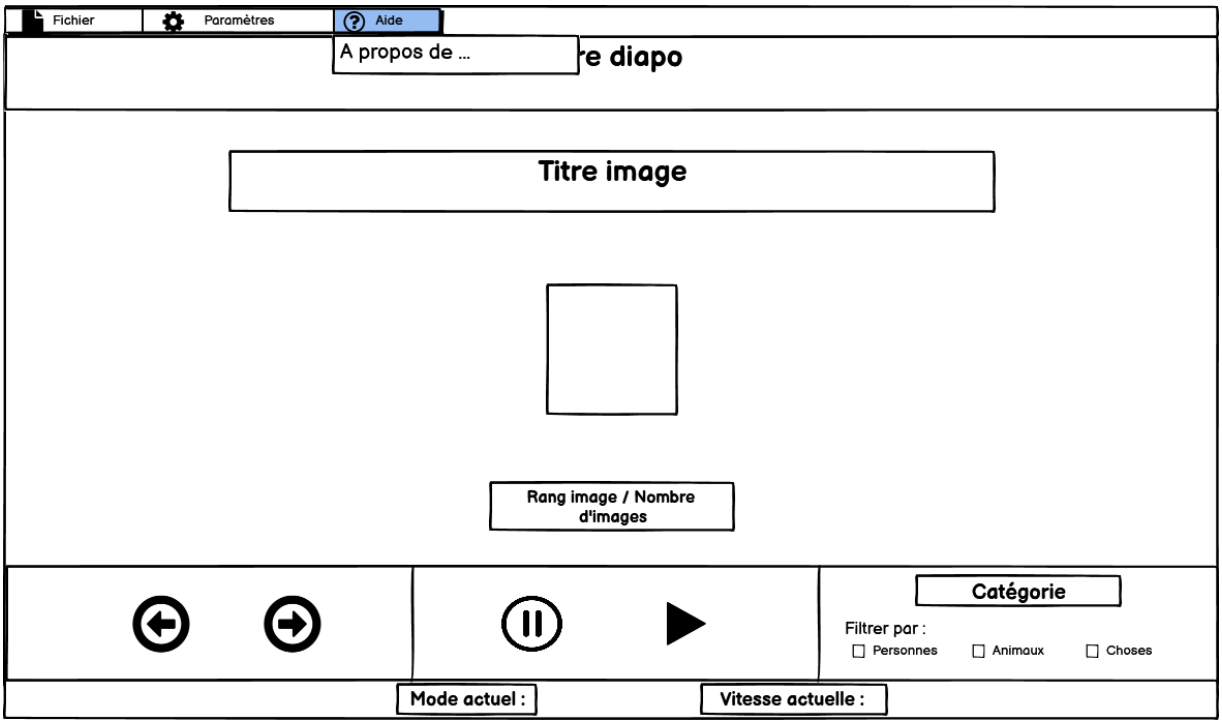
MaquetteMain



MaquetteFichier

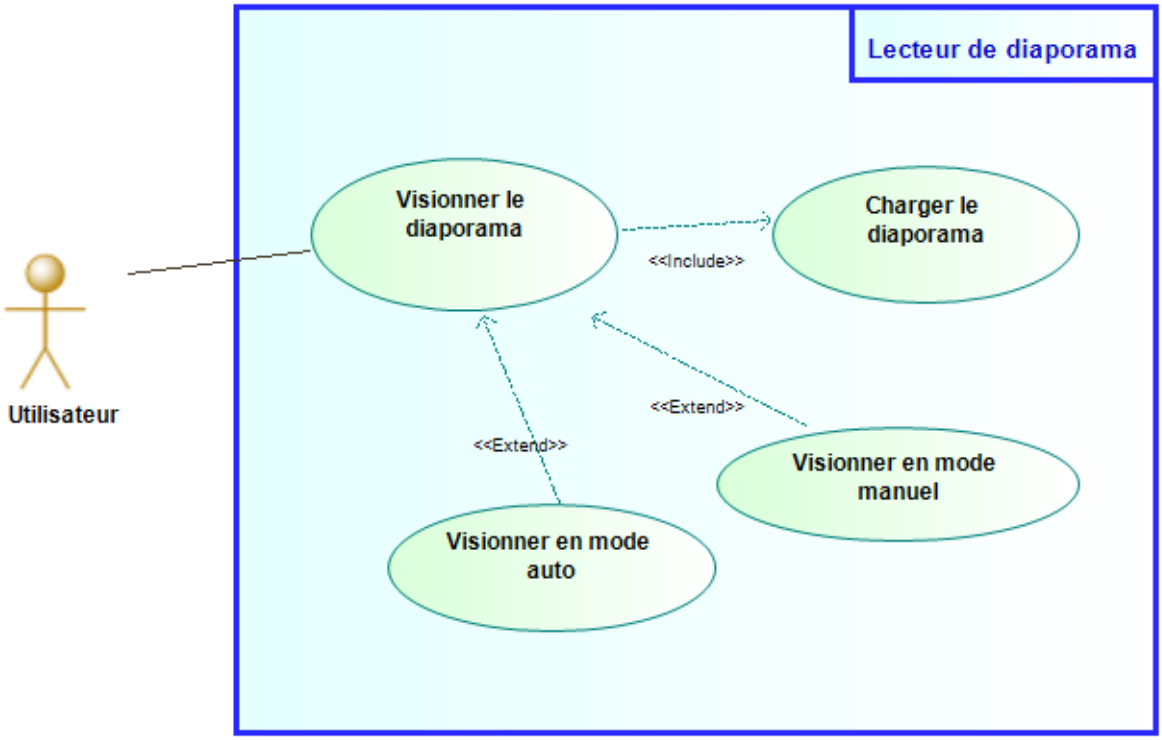


MaquetteParam



MaquetteAide

1.2 Diagramme des cas d'utilisations



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

Titre : L'utilisateur lance son diaporama

Acteurs : Utilisateur

Résumé : Un utilisateur souhaite lire son diaporama

Fiche : Description Réelle

Métadonnées :

Création : 16/05/2023

Responsable : AMBROISE Axel

Modifiée le : 16/05/2023 Version : 1

Enchainements

Préconditions :

1. L'utilisateur a lancé l'application.

Postconditions :

1. Le diaporama est lu

Enchaînement Nominal :

Acteur :	Système :	Maquettes
L'utilisateur va dans les paramètres		
	Affichage du menu déroulant	MaquetteFichier
L'utilisateur demande à charger son diaporama		
	Affichage du formulaire de chargement du diaporama	
L'utilisateur sélectionne son diaporama		
	Récupération puis affichage du diaporama	MaquetteMain
L'utilisateur navigue à l'aide des boutons disponible		

Enchaînement Alternatif :

L'utilisateur lance le diaporama en mode automatique puis change la vitesse de défilement

Acteur :	Système :	Maquettes
L'utilisateur va dans les parametres		
	Affichage du menu déroulant	MaquetteParam
L'utilisateur demande à charger son diaporama		
	Affichage du formulaire de chargement du diaporama	
L'utilisateur sélectionne son diaporama		

	Récupération puis affichage du diaporama	
L'utilisateur appuie sur le bouton de lancement de diaporama		
	Changement de mode manuel à mode automatique	
L'utilisateur va dans les parametres		
	Affichage du menu déroulant	MaquetteParam
L'utilisateur selectionne l'option de changement de vitesse de defilement		
	Affichage du formulaire de changement de vitesse	
L'utilisateur entre la vitesse qu'il souhaite puis valide		
	Modification de la vitesse de défilement	

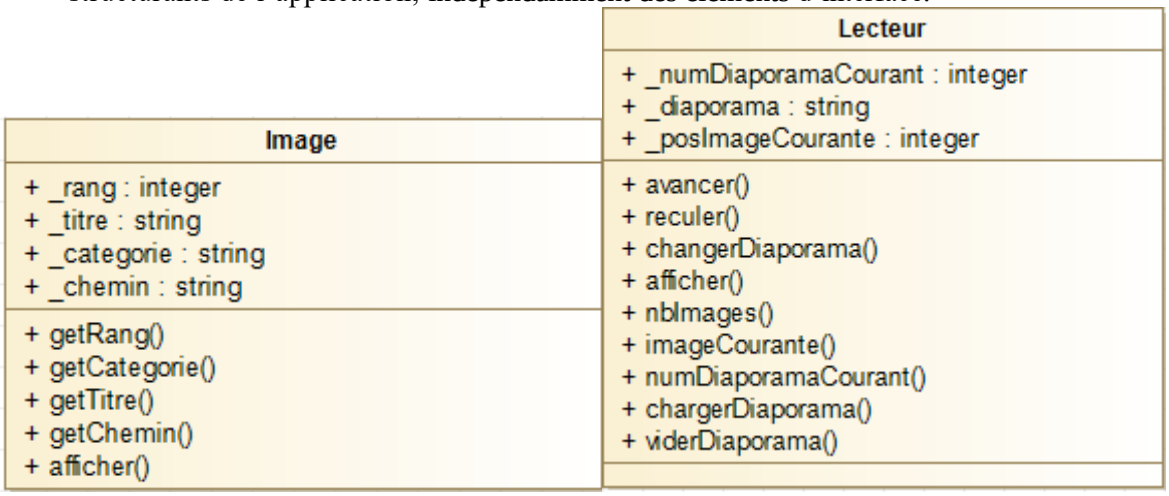
Enchaînement Alternatif :

L'utilisateur enlève son diaporama

Acteur :	**Système :**	**Maquettes**
L'utilisateur va dans les parametres		
	Affichage du menu déroulant	MaquetteParam
L'utilisateur demande enlevé son diaporama		
	Dechargement du diaporama	
	Affichage du menu	

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	Le rang de l'image au sein du diaporama auquel l'image est associée	unsigned int	4
_titre	L'intitulé de l'image	string	"Mickey"
_categorie	La catégorie de l'image (personne, animal, objet)	string	"Animal"
_chemin	Le chemin complet vers le dossier où se trouve l'image	string	"C:\\cartesDisney\\carteDisney1.gif"

Tableau 1 : Dictionnaire des éléments - Classe Image

Classe Lecteur			
Nom Attribut	**Signification**	**Type**	**Exemple**
_numDiaporamaCourant	Le numéro du diaporama courant, par défaut 0	unsigned int	1
_diaporama	Le pointeur vers les images du diaporama	Diaporama	monDiapo
_posImageCourante	La position, dans le diaporama, de l'image courante. Indéfini quand diaporama vide. Démarre à 0 quand diaporama non vide	Unsigned int	6

(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();
    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 XXX

(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 fourni main.cpp

Testeur :	M.Guiheneuf	**Date :**	
Element Testé :	LecteurDiaporama : v-0	**Version :**	1

Classe	Description	Valeur(s) en entrée	Résultat(s) attendu(s)	Conforme	
Valide n°1	Sélection du diaporama : Statut du lecteur avant la sélection, message de sélection, nombre d'images chargées et affichage de l'image courante	pNumDiaporama = 1	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)		
Valide n°2	Test avancer (): 4 fois ==> 1ère fois	-	avancer() : Diaporama num. 1 Image courante: image (rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)		
Valide n°3	Test avancer (): 4 fois ==> 2ème fois	-	avancer() : Diaporama num. 1 Image courante: image (rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)		

Valide n°4	Test avancer (): 4 fois ==> 3ème fois	-	avancer() : Diaporama num. 1 Image courante: image (rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)		
Valide n°5	Test avancer (): 4 fois ==> 4ème fois	-	avancer() : Diaporama num. 1 Image courante: image (rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)		
Valide n°6	Test reculer (): 5 fois ==> 1ère fois	-	reculer() : Diaporama num. 1 Image courante: image (rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)		
Valide n°7	Test reculer (): 5 fois ==> 2ème fois	-	reculer() : Diaporama num. 1 Image courante: image (rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)		
Valide n°8	Test reculer (): 5 fois ==> 3ème fois	-	reculer() : Diaporama num. 1 Image courante: image (rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)		
Valide n°9	Test reculer (): 5 fois ==> 4ème fois	-	reculer() : Diaporama num. 1 Image courante: image (rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)		
Valide n°10	Test reculer (): 5 fois ==> 5ème fois	-	reculer() : Diaporama num. 1 Image courante: image (rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)		

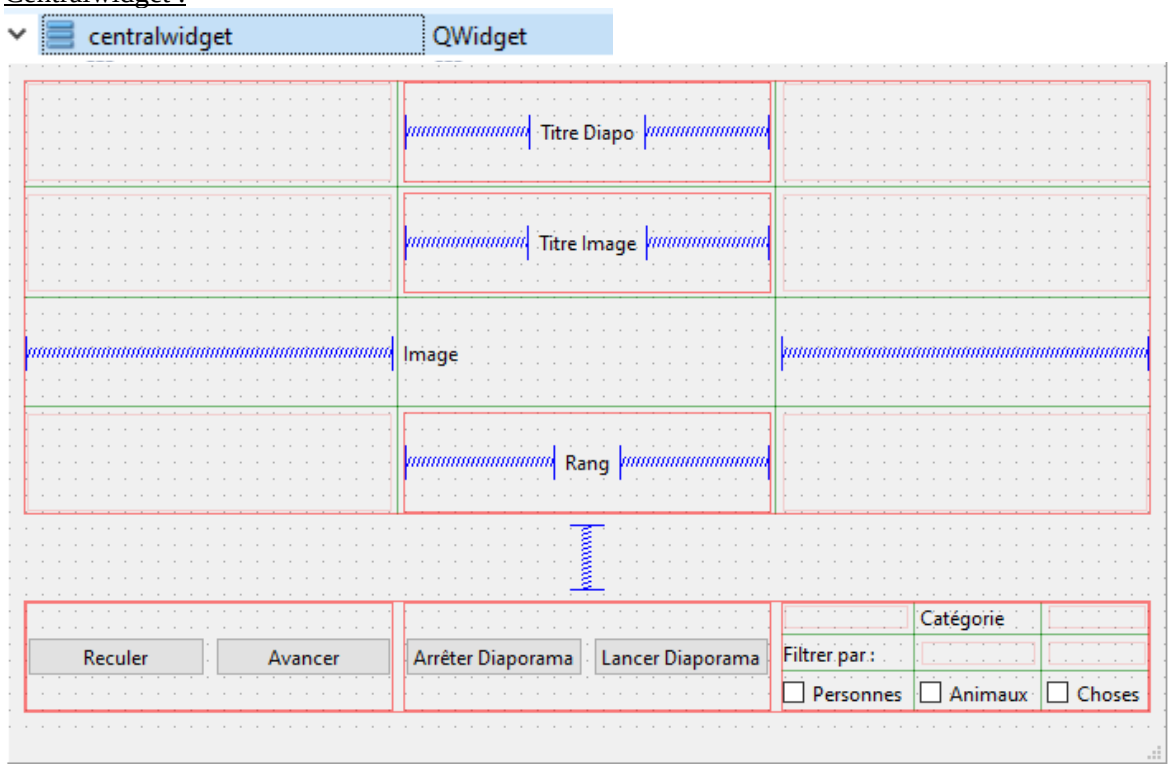
Valide n°11	Enlever le diaporama : Choix du diaporama 0, nombre d'images dans le lecteur et statut du lecteur	pNumDiaporama = 0	Enlever le diaporama courant = Choisir diaporama 0 0 images restantes dans le diaporama. Lecteur vide		

Version v1 – projet Graphique seul

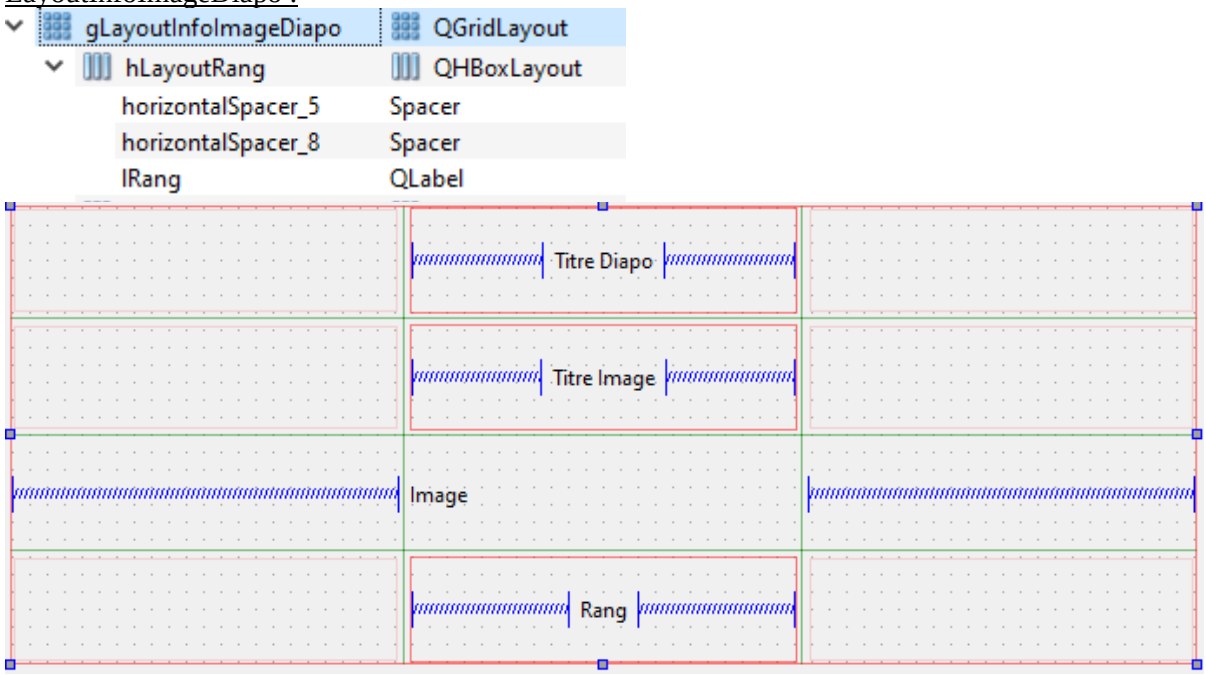
5. Éléments d'interface

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.

Centralwidget :



LayoutInfoImageDiapo :



LayoutRang :

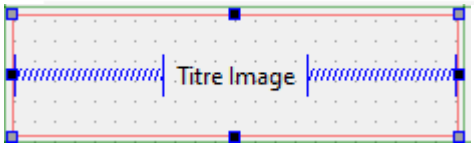
hLayoutRang	QHBoxLayout
horizontalSpacer_5	Spacer
horizontalSpacer_8	Spacer
lRang	QLabel

LayoutTitreDiapo :

hLayoutTitreDiapo	QHBoxLayout
horizontalSpacer	Spacer
horizontalSpacer_2	Spacer
lTitreDiapo	QLabel

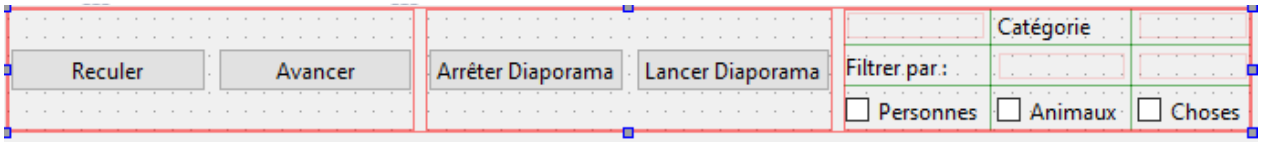
LayoutTitreImage :

hLayoutTitreImage	QHBoxLayout
horizontalSpacer_3	Spacer
horizontalSpacer_4	Spacer
lTitreImage	QLabel



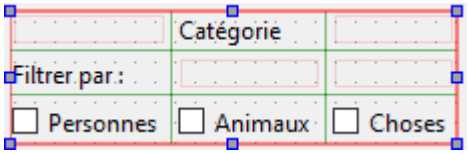
LayoutMenuInter :

hLayoutMenuInter	QHBoxLayout
hLayoutChangerImage	QHBoxLayout
bAvancer	QPushButton
bReculer	QPushButton

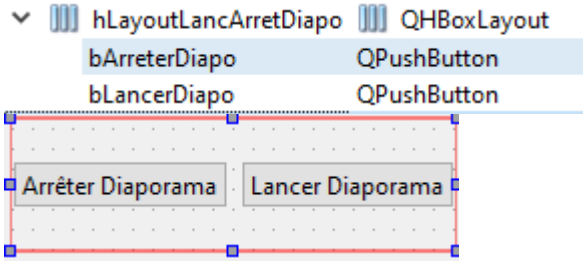


LayoutFiltrage :

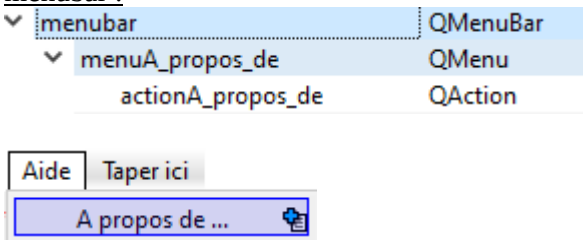
hLayoutFiltrage	QGridLayout
cbAnimal	QCheckBox
cbChose	QCheckBox
cbPersonne	QCheckBox
lCategorie	QLabel
lFiltrage	QLabel



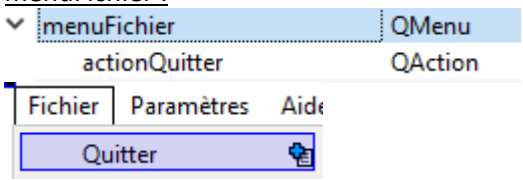
LayoutLancArretDiapo :



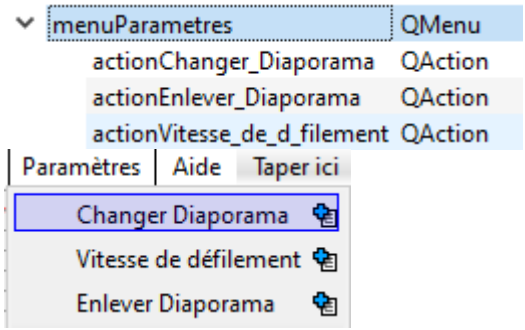
menubar :



menuFichier :



menuParametres :



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***

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*

Version v2 –

7. Diagramme de classes (UML)

Idem que V0

8. Comportement de l'application

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

En cours...

Figure 2 : Diagramme états-transitions du lecteur de diaporamas – v2

7.2 Dictionnaire des états, événements et Actions (v2)

Dictionnaire des états du diaporama

nomEtat	*Signification*

Tableau 2 : États du lecteur de diaporamas – v2

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*

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

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*

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 pregnant en charge cet événement →			
Événement → nomEtat			

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.

9. 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 *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	??

Remarques sur l'implémentation :

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

8.2 Tests (v2)

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***

Version v5 –

10. Diagramme de classes (UML)

A faire – s'il y a des changements - sinon indiquer que idem vXX

11. Comportement de l'application

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

A faire

Figure 3 : Diagramme états-transitions du lecteur de diaporamas – v5

11.2 Dictionnaire des états, événements et Actions (v5)

Dictionnaire des états du diaporama

nomEtat	*Signification*

Tableau 6 : États du lecteur de diaporamas – v5

Dictionnaire des événements faisant changer le diaporama d'état

nomEvénement	*Signification*

Tableau 7 : Evénements faisant changer le diaporama d'état – v5

Description des actions réalisées lors de la traversée des transitions

nomAction	*Signification*

Tableau 8 : 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 pregnant en charge cet événement →			
Événement → nomEtat			

Tableau 9 : 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.

12. 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 *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	??

Remarques sur l'implémentation :

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

12.2 Tests (v5)

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***

13. Bilan

Dépôt Git où trouver le projet complet (les versions réalisées)
Temps global de travail (pour le groupe)
Apprentissages majeurs
Difficultés majeures
Points positifs / négatifs de l'activité