Université de Pau et des Pays de l'Adour - IUT DE BAYONNE - PAYS-BASQUE année 2022-2023

Département Informatique saé 2.01

Saé 2.01 – Développement d’une application

Lecteur de diaporamas – Dossier d’Analyse et conception



## Compléments de spécifications externes.

*On précise uniquement les points qui vous ont semblé flous ou bien incomplets. Rien de plus à signaler dans cette étude.*

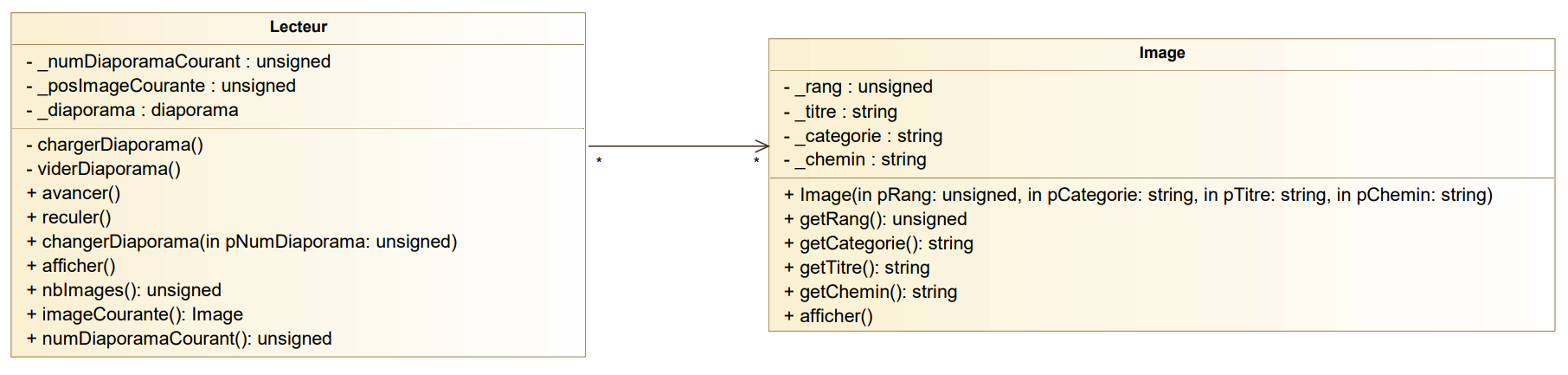
## Scénarios

|  | **Enchaînement Nominal** | |
| --- | --- | --- |
| **Messages** | **Acteur : Utilisateur** | **Système : Lecteur Diaporama** |
| 1 | L'utilisateur demande de charger le diaporama |  |
| 2 |  | Le système charge le diaporama |
| 3 |  | Le système affiche le diaporama |
| 4 | L'utilisateur demande à passer à l'image suivante |  |
| 5 |  | Le système affiche l'image suivante |
| 6 | L'utilisateur demande à passer à l'image précédente |  |
| 7 |  | Le système affiche l'image précédente |
|  | **Enchaînements Alternatifs** | |
|  | **Acteur : Utilisateur** | **Système : Lecteur Diaporama** |
| 4.A | **Mode Auto** | |
| 4.A.1 | L'utilisateur demande à déclencher le mode automatique |  |
| 4.A.2 |  | Le système déclenche le mode automatique |
|  | **Acteur : Utilisateur** | **Système : Lecteur Diaporama** |
| 4.B | **Modifier vitesse d'affichage** | |
| 4.B.1 | L'utilisateur demande à modifier la vitesse d'affichage du diaporama |  |
| 4.B.2 |  | Le système modifie la vitesse d'affichage du diaporama |

*Tableau 1 : Scénarios*

## Diagramme de classe (UML)

1. 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.



1. Dictionnaire des éléments pour chaque classe

| **Classe Lecteur** | | | |
| --- | --- | --- | --- |
| **Nom attribut** | **Signification** | **Type** | **Exemple** |
| \_numDiaporamaCourant | numéro du diaporama actuel | unsigned int | 0 |
| \_posImageCourante | position dans le diaporama de l'image courante | unsigned int | 1 |
| \_diaporama | pointeur vers l’ensemble des images du diaporama | Diaporama |  |

*Tableau 2 : Dictionnaire des éléments - Classe Lecteur*

| **Classe Image** | | | |
| --- | --- | --- | --- |
| **Nom attribut** | **Signification** | **Type** | **Exemple** |
| \_rang | rang de l'image au sein du diaporama auquel l'image est associée | unsigned int | 15 |
| \_title | intitulé de l'image | string | Dingo |
| \_categorie | catégorie de l'image (personne, animal, objet) | string | Personne |
| \_chemin | chemin complet vers le dossier où se trouve l'image | string | F:\cartesDisney\Disney\_15.gif |

*Tableau 3 : Dictionnaire des éléments - Classe Image*

1. Dictionnaire des méthodes : vous pouvez fournir directement le fichier entête de chaque classe.

Image.h :

#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*

Lecteur.h :

#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*

# Version v0 – Version console seule

## 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

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

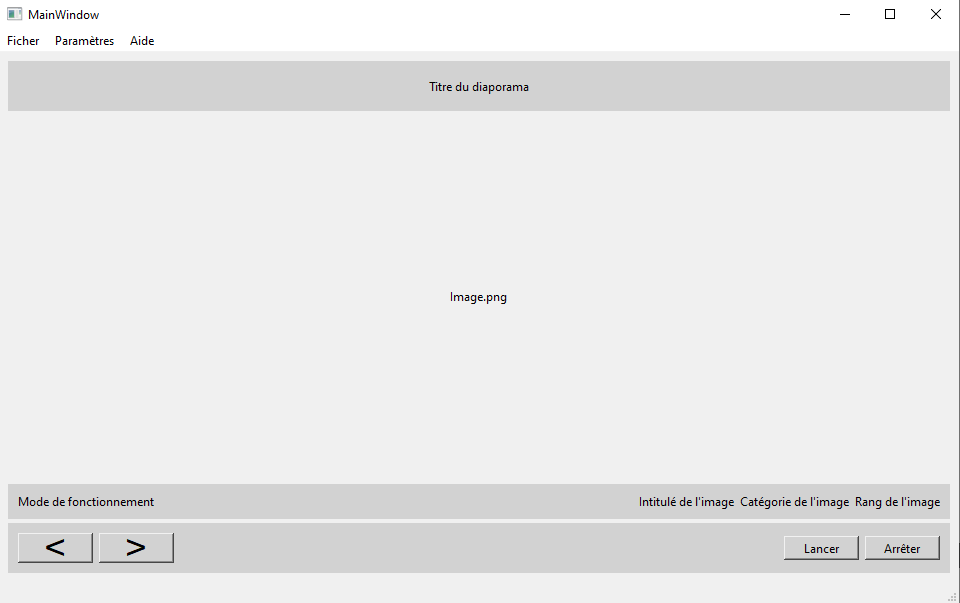
| **Valeurs fournies** | **Valeurs attendues** |
| --- | --- |
| afficher() | “Lecteur vide” |
| changerDiaporama(1) | “Diaporama num. 1 selectionne.  4 images chargees dans le diaporama” |
| afficher() | “Diaporama num. 1 :  Image courante :  image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)” |
| 4 fois : avancer()  afficher() | “Diaporama num. 1 :  Image courante :  image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)  Diaporama num. 1 :  Image courante :  image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)  Diaporama num. 1 :  Image courante :  image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)  Diaporama num. 1 :  Image courante :  image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)” |
| 5 fois : reculer()  afficher() | “Diaporama num. 1 :  Image courante :  image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)  Diaporama num. 1 :  Image courante :  image( rang:3, titre:Blanche Neige, categorie:personne, chemin:C:\cartesDisney\carteDisney2.gif)  Diaporama num. 1 :  Image courante :  image( rang:2, titre:Cendrillon, categorie:personne, chemin:C:\cartesDisney\carteDisney4.gif)  Diaporama num. 1 :  Image courante :  image( rang:1, titre:Grincheux, categorie:personne, chemin:C:\cartesDisney\carteDisney1.gif)  Diaporama num. 1 :  Image courante :  image( rang:4, titre:Mickey, categorie:animal, chemin:C:\cartesDisney\carteDisney1.gif)” |
| changerDiaporama(0) | “0 images restantes dans le diaporama.” |
| afficher() | Lecteur vide. |

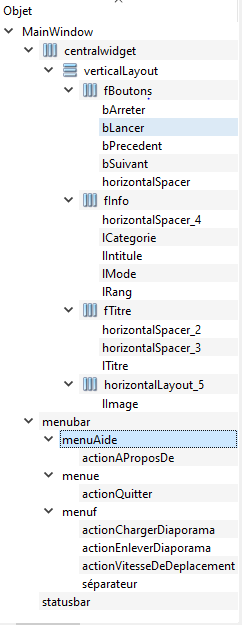
# Version v1 – projet Graphique seul

## 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.*





**Description sommaire des éléments de l’interface**

* “<” → bPrecedent : bouton qui permet de passer à l’image précédente de l’image courante du diaporama.
* “>” → bSuivant : bouton qui permet de passer à l’image suivante de l’image courante du diaporama.
* “Lancer” → bLancer : bouton qui permet de lancer le mode auto lorsque celui-ci est désactivé.
* “Arrêter” → bArreter : bouton qui permet d’arrêter le mode auto lorsque celui-ci est activé.
* “Mode de fonctionnement” → lMode : label qui permet de savoir dans quel mode de fonctionnement (auto ou manuel) on est.
* “Intitulé de l’image” → lIntitule : label qui permet de connaître l’intitulé de l’image courante.
* “Catégorie de l’image” → lCategorie : label qui permet de connaître la catégorie de l’image courante.
* “Rang de l’image” → lRang : label qui permet de connaître le rang de l’image courante dans le diaporama courant.
* “label.png” → lImage : label qui permet d’afficher l’image courante.
* “Titre du diaporama” → label qui affiche le titre du diaporama courant.
* actionAProposDe → affiche

## 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 :

### Nous avons créé 9 public slots, qui sont :

### void quitter()

### void charger()

### void enlever()

### void vitesse()

### void apropos()

### void precedent()

### void suivant()

### void lancer()

### void arreter()

Nous les avons connecté avec les différent bouton de l’interface graphique et avec différent signals :

| **Éléments d’intéraction** | **SIGNALS** | **SLOTS** |
| --- | --- | --- |
| actionQuitter | triggered() | quitter() |
| actionChargerDiaporama | triggered() | charger() |
| actionEnleverDiaporama | triggered() | enlever() |
| actionVitesseDeDeplacement | triggered() | vitesse() |
| actionAProposDe | triggered() | apropos() |
| bSuivant | clicked() | suivant() |
| bPrecedent | clicked() | precedent() |
| bArreter | clicked() | arreter() |
| bLancer | clicked() | lancer() |

### 6.2 Test

| **Nom de l’action effectuée** | **Comportement attendu** | **Comportement observé** |
| --- | --- | --- |
| Quitter le programme | Message QDebug “Bouton quitter” | Test validé |
| Charger le diaporama | Message QDebug “Bouton charger” | Test validé |
| Enlever le diaporama | Message QDebug “Bouton enlever” | Test validé |
| Modifier la vitesse de défilement | Message QDebug “Bouton vitesse” | Test validé |
| Afficher la fenêtre A propos | Message QDebug “Bouton apropos” | Test validé |
| Aller à l’image suivante | Message QDebug “Bouton suivant” | Test validé |
| Aller à l’image précédente | Message QDebug “Bouton precedent” | Test validé |
| Arrêter le mode automatique | Message QDebug “Bouton lancer” | Test validé |
| Démarrer le mode automatique | Message QDebug “Bouton arreter” | Test validé |

# Version v2 –

## Diagramme de classes (UML)

*A faire – s’il y a des changements - sinon indiquer que idem v0*

## Comportement de l’application

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

*A faire*

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

### 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*

### 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.*

## Implémentation et tests

### Implémentation (v2)

### Liste et rôle des fichiers de cette version :

| lecteurVue.h | Spécification de la classe graphique Qt contenant l’interface du lecteur de diaporamasPréciser 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 ImagePréciser 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

### 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 –

## Diagramme de classes (UML)

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

## Comportement de l’application

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

*A faire*

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

### Dictionnaire des états, événements et Actions (v5)

**Dictionnaire** **des états du diaporama**

| *nomEtat* | *Signification* |
| --- | --- |
|  |  |
|  |  |

*Tableau 2 : États du lecteur de diaporamas – v5*

**Dictionnaire des événements faisant changer le diaporama d’état**

| *nomEvénement* | *Signification* |
| --- | --- |
|  |  |
|  |  |

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

**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 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 pregnant en charge cet événement 🡪* |  |  |  |
| --- | --- | --- | --- |
| *Événement 🡪*  *nomEtat* |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*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.*

## Implémentation et tests

### Implémentation (v5)

### Liste et rôle des fichiers de cette version :

| lecteurVue.h | Spécification de la classe graphique Qt contenant l’interface du lecteur de diaporamasPréciser 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 LecteurPréciser le rôle |
| lecteur.cpp | Corps de la classe Lecteur |
| image.h | Spécification de la classe ImagePréciser 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

### 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

## 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