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

Département Informatique SAÉ 20.1

Saé 2.01 – Développement d’une application

Lecteur de diaporamas – Dossier d’Analyse et conception

https://github.com/MaxMontouro/LecteurDiaporama

## Compléments de spécifications externes.

## Nous n’avons pas eu de points flous durant la grande partie de la SAÉ. Or, lorsque des petits problèmes sont survenus nous sommes directement partis voir un professeur afin qu’il nous vienne en aide.

## Scénarios

2.1 Scenario nominal:

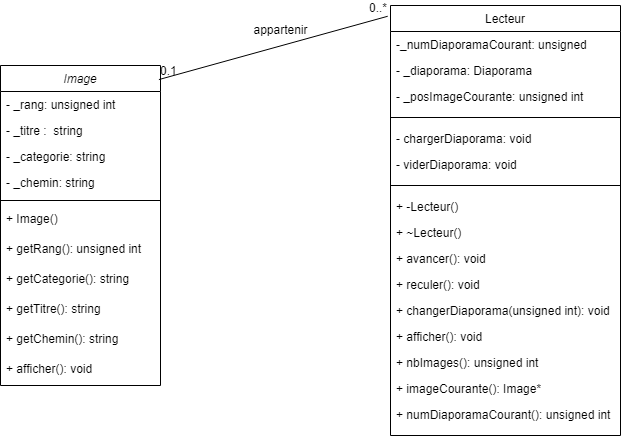
|  |  |  |
| --- | --- | --- |
| Cas Utilisation | LecteurDiaporama | |
| Action Primaire | Visionner un diaporama | |
| Système | Système de lecture de diaporama | |
| Précondition | L’utilisateur à lancer l’application et un diaporama | |
| Opérations | Acteur | Système de lecture de diaporama |
| 1. | L’utilisateur choisi un diaporama et le lance |  |
| 2. |  | Le système affiche le diaporama et l’affiche de base en mode manuel |
| 3. | L’utilisateur defile les differents transparents du diaporama |  |

2.2 Scenarios alternatifs:

|  |  |  |
| --- | --- | --- |
| Cas Utilisation | LecteurDiaporama | |
| Action Primaire | Visionner un diaporama | |
| Système | Système de lecture de diaporama | |
| Précondition | L’utilisateur à lancer l’application et un diaporama | |
| Opérations | Acteur | Système de lecture de diaporama |
| 1. | L’utilisateur choisi un diaporama et le lance |  |
| 2. |  | Le système affiche le diaporama et l’affiche de base en mode manuel |
| Extensions | | |
| 3.A.1 | L’utilisateur choisi un autre diaporama stocké sur la base de données |  |
| 3.A.2 |  | Le système affiche le diraporama souhaité par l’utilisateur |
| 3.B.1 | L’utilisateur choisi de passer en mode automatique de defilement des transparents |  |
| 3.B.2 |  | Le système affiche les transparents en mode automatique |

## 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 courant, par défaut 0 | unsigned int | 1 |
| \_diaporama | pointeurs vers les images du diaporama | Diaporama | grincheux |
| \_posImageCourante | Position, dans le diaporama, de l'image courante. Indéfini quand diaporama vide. Démarre à 0 quand diaporama non vide | unsigned int | 1 |

Tableau 1 : Dictionnaire des éléments - Classe Lecteur

|  |  |  |  |
| --- | --- | --- | --- |
| **Classe Image** | | | |
| **Nom attribut** | **Signification** | **Type** | **Exemple** |
| \_rang | Rang de l’image au sein du diaporama auquel l’iamge est associé | unsigned int | 1 |
| \_titre | Intitule de l’image | string | grincheux |
| \_\_categorie | Catégorie de l’image (personne, animal, objet) | string | animal |
| \_chemin | Chemin complet vers le dossier ou se trouve l’image | string | “C://xxx |

Tableau 3 : Dictionnaire des éléments - Classe Lecteur

1. 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

1. 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

## Implémentation et tests

## 4.1 Implémentation

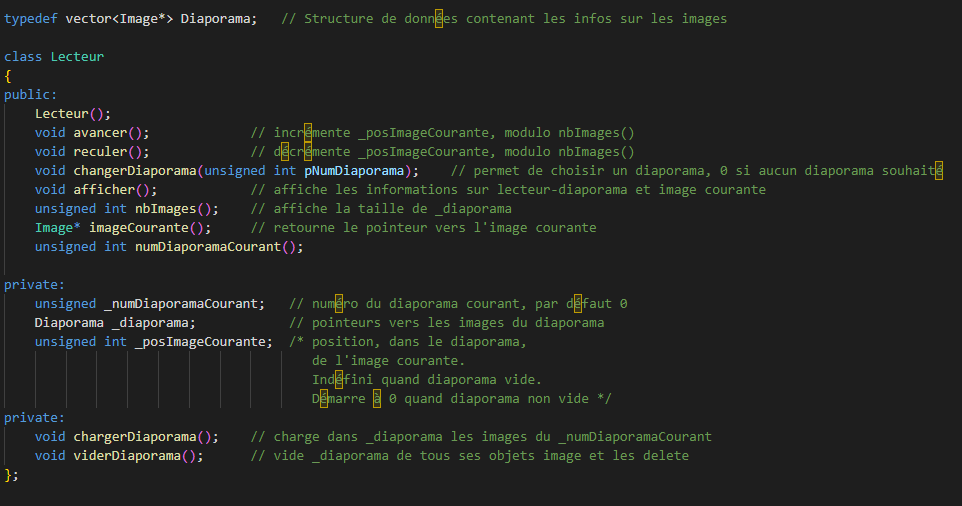
### Liste et rôle des fichiers de cette version :

|  |  |
| --- | --- |
| 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 |

### 4.1.1 Classe Lecteur :

### 

### Définition de la classe



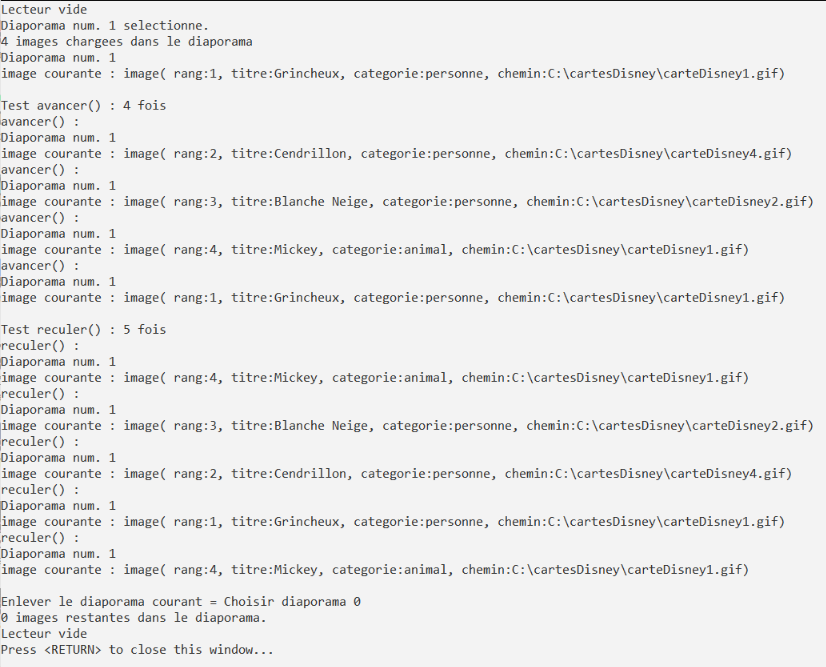
## Ici, nous retrouvons les données membres, les méthodes privées et publiques de la classe lecteur.

## 

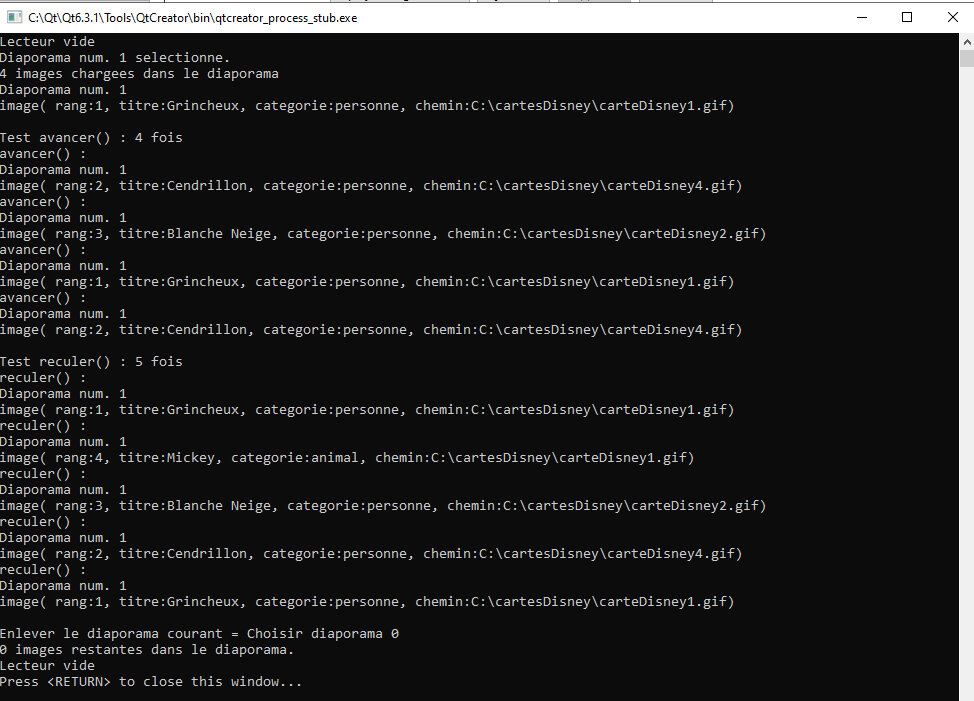
## 4.2 Test

Test avec le programme fourni main.cpp

*Résultat que nous devons obtenir*



*Résultat obtenu après compilation et éxecution :*



## 4.3 Temps Passé

## Pour la v0 de l’application, nous avons passé aux alentours d’une semaine avec les modifications apportées à la fin. En parallèle nous avancions la v1. De plus, la création de la v0 a été en parti finalisé pendant les vacances du Printemps donc le temps ne peut pas être quantifier.

# Version v1 – projet Graphique seul

## Éléments d’interface

## Implémentation et tests

### 6.1 Implémentation

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

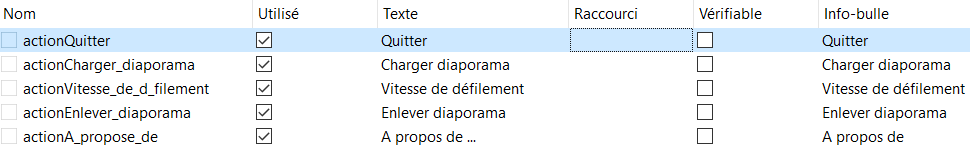
|  |  |
| --- | --- |
| lecteurVue.h | Spécification de la classe graphique Qt contenant l’interface du lecteur de diaporamas |
| lecteurVue.cpp | Corps de la classe LecteurVue |
| lecteurvue.ui | Fichier du dessin de l’interface réalisé par QtDesigner |
| main.cpp | Teste les méthodes de la classe Lecteur |

### Remarques sur l’implémentation :

### Nous n’avons pas implémenté de signal/slot pour cette version du projet.

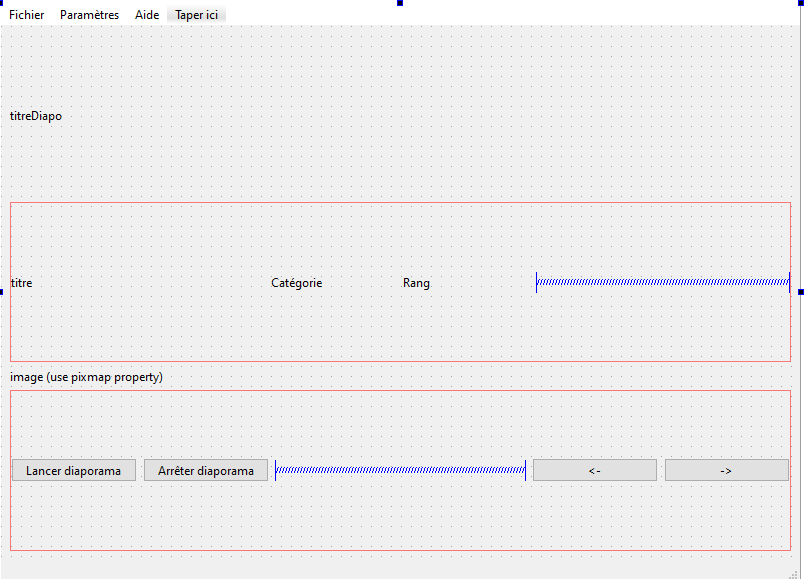
### 6.2 Test

### Nous pouvons voir ci-dessous, les différentes actions qui peuvent être réalisé sur le projet graphique.

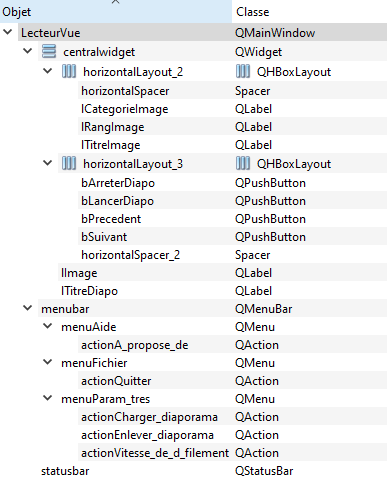


### 6.3 Temps passé

### Pour la v1, du fait que nous avancions en parallèle de la v0 nous avons passé aux alentours de deux semaines (en temps écoulé) celui-ci représente environ 3h/4h de travail. Pour le projet graphique, nous avons mis en place un bloc-notes avec les différents points importants qui doivent s’y trouver.

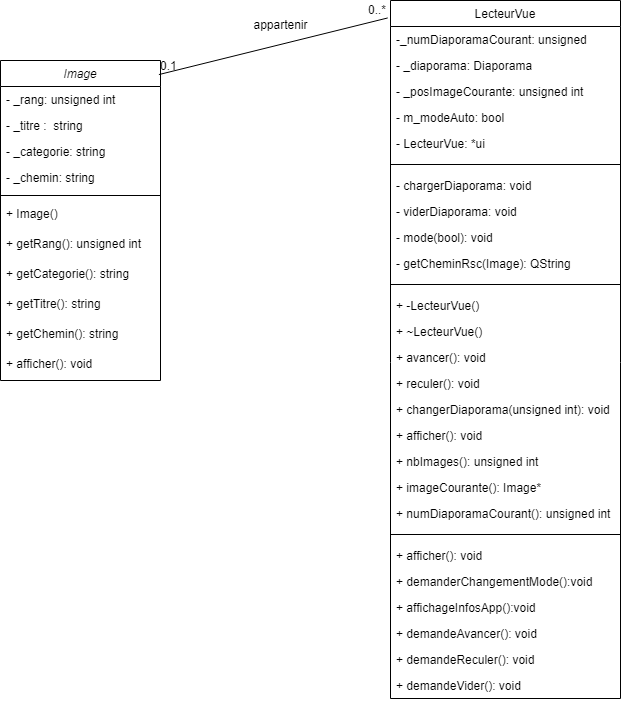


### Première aperçu du projet graphique (ui) avec notre choix d’organisation selon les éléments donnés.



# Version v2 –

## Diagramme de classes (UML)



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

*A faire*

Figure 2 : 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

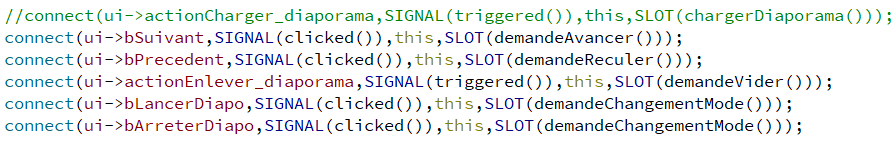
### Implémentation (v2)

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

|  |  |
| --- | --- |
| lecteurVue.h | Ici la classe LecteurVue contient tous les éléments qui permettent le bon fonctionnement de l’application à cette étape |
| lecteurVue.cpp | Corps de la classe LecteurVue. |
| lecteurvue.ui | Fichier du dessin de l’interface réalisé par QtDesigner |
| lecteur.h | Dans cette version nous n’avons plus la classe |
| lecteur.cpp | Corps de la classe Lecteur |
| image.h | Fonctionnement et mise à jour de l’image |
| image.cpp | Corps de la classe Image |
| main.cpp | Corps du fichier principal |

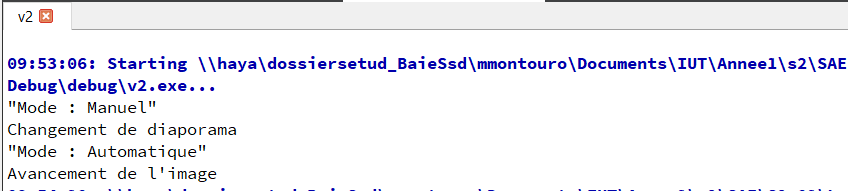
### Remarques sur l’implémentation :

### Pour les signal/slots nous avons implémenter les procédures qui sont présent dans le .h de la classe LecteurVue. Voir ci-dessous les différents connect.



### Tests (v2)

### Pour cette version, les tests se résume à vérifier si les connect se font bien. Par exemple si le mode automatique s’affiche bien etc…



Nous pouvons voir que le chargement du diaporama s’effectue correctement

Pour la v3, nous avons implémenter le mode automatique en utilisant notamment la bibliothèque QTimer.

Pour la v4, nous avons implémenter la fonction chargerDiaporama qui permet d’afficher les différentes images, enlever les différentes images mais aussi modifier la vitesse de défilement.

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

### 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

### 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.*

## 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

12.1 Dépôt Git se trouve à l’adresse suivante :

<https://github.com/MaxMontouro/LecteurDiaporama>

* 1. Temps Global :

Pour Clément :

Pour Maxime :

Toutes les heures de TD/TP misent à disposition, plus de 3h pendant les vacances de Pâques, plusieurs heures sur le temps extra-universitaire.

* 1. Apprentissages majeurs :

Pour Clément :

Pour Maxime :

Git et Github, codé en IHM, jeux de tests.

* 1. Difficultés majeurs :

Pour Clément :

Pour Maxime :

Je pense que ma difficulté majeure était de traiter le travail de 3 personnes à 2. En effet, le projet était conséquent, ainsi que la doc. De ce fait, il était difficile de tout gérer en même temps sur le nombre d’heure dédié à la SAÉ.

* 1. Points positifs/ négatifs de l’activité :

Pour Clément :

Pour Maxime :

J’ai appris pendant cette SAÉ à mieux gérer les applications avec IHM et notamment en utilisant QT mais aussi à me concentrer sur les outils pour versionner des codes comme git.

Or je n’ai pas trouvé le sujet intéressant comme ont pu être les sujets développés au cours du S1. De plus, étant donné que nous étions 2 dans le groupe le travail était dotant plus conséquent. Ainsi, ce fut un projet très lourd.