

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement 2016-2017

Flux vidéo et anamorphose cylindrique

Tuteur académique
Christophe LENTE

Étudiant
Alexis SALUDO (DI5)

3 avril 2017



Liste des intervenants

Nom	Email	Qualité
Alexis SALUDO	alexis.saludo@etu.univ-tours.fr	Étudiant DI5
Christophe LENTE	christophe.lente@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Alexis Saludo susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Christophe Lente susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Alexis Saludo, *Flux vidéo et anamorphose cylindrique*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2016-2017.

```
@mastersthesis{
  author={Saludo, Alexis},
  title={Flux vidéo et anamorphose cylindrique},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2016-2017}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	vi
Introduction	1
I Objectif et État de l’art	2
1 Contexte de la réalisation	3
1 Contexte	3
2 Objectif	3
2.1 Anamorphose Cylindrique sur une image	3
2.2 Anamorphose Cylindrique sur un flux vidéo	4
2.3 Anamorphose Cylindrique d’un objet en 3D	4
2 Description générale	5
1 Méthode de transformation.....	5
2 Caractéristiques des utilisateurs	6
3 Fonctionnalités du système	6
4 Contraintes de développement	7

5	Contraintes d'exploitation.....	7
6	Structure générale du système	7
3	Etat de l'art	8
1	Recherche bibliographique	8
1.1	Jean Louis Vaulezard (1630).....	8
1.2	Anamorph Me ! (2001).....	8
1.3	Jonty Hurwitz (2008).....	9
1.4	Reactable (2008)	9
1.5	Anamorphose et capture caméra (2008)	9
1.6	Anamorphicon (2008).....	10
2	Positionnement du projet	10
3	Veille Technologique	10
3.1	Librairie Webcam.....	10
3.2	Librairie Media Vidéo.....	11
II	Développement	12
4	Analyse	13
1	Schéma général de l'application.....	13
2	Diagramme de classe	14
3	Amélioration de l'application	16
3.1	Amélioration de l'utilisation des Threads.....	16
3.2	Amélioration de l'utilisation en ressource	16
3.3	Amélioration de la fonction de transformation	17
4	Ajout de nouvelles fonctionnalités.....	17
4.1	Déplacement du miroir par contact tactile	17
4.2	Modification de l'affichage	17
4.3	Paramétrage en direct	18
4.4	Aide à l'impression et l'export d'image	18
5	Qualité du code	19
1	Documentation	19
1.1	Guide utilisateur.....	19
1.2	Documentation développeur.....	19
1.3	Commentaire dans le code	20
2	Tests	20
2.1	Test de bon fonctionnement	20
2.2	Test unitaire	20
3	Métriques.....	21

3.1	Sonar	21
3.2	Java Mission Controle	23
6	Bilan	28
1	Etat actuel du projet	28
2	Conclusion	28
	Bibliographie	29
	Annexes	32
	Spécification	33
1	Spécifications fonctionnelles.....	33
2	Interface matériel / logiciel	33
3	Interface homme / machine	34
4	Interface logiciel/logiciel.....	35
	Gestion de projet	36
1	Tâches	36
1.1	Anamorphose image	36
1.2	anamorphose Vidéo	37
1.3	Optionnel	38
2	Démarche projet	38
3	Livrable	38
4	Planning Prévisionnel	39
	Bilan Gestion de Projet	40
1	Présentation du projet	40
2	Livrables et Taches	40
2.1	Analyse de faisabilité.....	41
2.2	Analyse de risque.....	42
2.3	Suivie de projet	42
3	Outils de gestion de projet	44
	Documentation Utilisateur	45
1	Comment exécuter le programme.....	45
2	L'interface	45
2.1	Fenêtre Source.....	46
2.2	Fenêtre Destination	49
3	Actions possibles	50
3.1	Démarrer l'anamorphose.....	50

3.2	Redéfinir les paramètres.....	50
3.3	Déplacer la transformation.....	50
3.4	Changer de source	50
3.5	Enregistrement d'image et aide à l'impression.....	50
Documentation Développeur		52
1	Mise en place de l'environnement de développement	52
1.1	Prérequis	52
1.2	Installation.....	52
2	Implémentation du diagramme de classe	53
2.1	Diagramme complet	53
2.2	Choix d'implémentation.....	55
2.2.1	Implémentation général	55
2.2.2	Le panneau d'image source	55
2.2.3	La classe Anamorphose	56
2.2.4	Le panneau d'image transformé.....	58
2.2.5	Le panneau de configuration.....	59
2.3	Possibilité d'amélioration	60
2.3.1	Utilisation de Raster	60
2.3.2	Un panneau d'aide à l'impression amélioré.....	60
2.3.3	Gérer le cas d'un changement de transformation.....	61
3	Qualité Logiciel	61
3.1	Fichiers de Log.....	61
3.2	Tests Unitaires.....	61
3.3	Métrique Sonar	62
3.4	Métrique JMC	62
4	Versionning.....	63
Cahier de tests		64
1	Panneau de configuration.....	64
1.1	Conditions de test.....	64
1.1.1	Paramètres général aux anamorphoses	64
1.1.2	Paramètres liés à l'anamorphose cylindrique	65
1.2	Cas de tests.....	65
1.2.1	Liste des cas	65
1.2.2	Résultat	65
2	Fichier de configuration	66
2.1	Conditions de test.....	66
2.2	Cas de tests.....	67
2.2.1	Liste des cas	67

2.2.2	Résultat	67
3	Panneau d'aide à l'impression	68
3.1	Conditions de test	68
3.2	Cas de tests.....	69
3.2.1	Liste des cas	69
3.2.2	Résultat	69
Comptes rendus hebdomadaires		70

Table des figures

2 Description générale

1	Schéma de l'anamorphose	5
2	Cas d'utilisation de l'application	6

3 Etat de l'art

1	Sculpture de grenouille après anamorphose	9
2	Schéma du Projet Anamorphicon	10

4 Analyse

1	Schéma général du projet	13
2	Diagramme de classe du projet	14
3	Diagramme de classe complet du projet	15

5 Qualité du code

1	Metrique Sonar Avant correction	21
2	Liste de bug trouvé par Sonar Avant correction	22
3	Metrique Sonar Après correction	23
4	Metrique JMC avant correction	24
5	Metrique JMC avant correction	25
6	Metrique JMC avant correction	25
7	Metrique JMC après correction	26
8	Metrique JMC après correction	26
9	Metrique JMC après correction	27

Spécification

1	Interface du projet	34
---	---------------------------	----

Gestion de projet

1	Planning prévisionnel réalisé au début du projet.....	39
---	---	----

Bilan Gestion de Projet

1	Planning S10 previsionnel	41
2	Planning S10 réalisé	43

Documentation Utilisateur

1	Fenêtre Source	46
2	Barre de Menu	46
3	Ecran Source	47
4	Bouton de chargement d'un fichier	47
5	Ecran de Caméra.....	48
6	Panneau de paramètre et d'informations.....	48
7	Panneau d'image transformée	49
8	Panneau d'aide à l'impression	51
9	Fenetre d'erreur	51

Documentation Développeur

1	Dossier libs	52
2	Dossier vlcj.....	53
3	Dossier config	53
4	Diagramme de classe du projet	54
5	Schéma général de l'application.....	55
6	Diagramme de Classes du panneau d'image source.....	55
7	Diagramme de classes de l'anamorphose.....	56
8	Diagramme de Classes du panneau destination.....	58
9	Diagramme de Classes du panneau de configuration	59
10	Fenetre Java Mission Control	62

Cahier de tests

1	Panneau de configuration.....	64
2	Fichier de configuration	66
3	Panneau d'aide à l'impression	68

Introduction

Ce projet de Recherche et développement, réalisé en 5ème année, s'inscrit dans la formation effectuée à l'Ecole Polytechnique de Tours et constitue une réelle expérience en termes de conduite de projet et de génie logiciel. Le Projet se déroule du 22 septembre 2016 à mars 2017 à raison de 16 heures par semaine. Il donne lieu à la rédaction d'un cahier de spécifications, de deux rapports et deux soutenances orales pour présenter le rendu du travail. Ce document est donc le cahier de spécification du projet "Trompe l'œil et Anamorphose Cyindrique". Il définit les besoins, l'environnement du projet et les objectifs à réaliser.

Les acteurs sont :

- La maitrise d'œuvre (MOE) : A. Saludo étudiant en 5ème année de l'Ecole d'Ingénieur Polytech'Tours au sein du département Informatique, dans le cadre du Projet Recherche et développement.
- La maitrise d'ouvrage (MOA) : Christophe Lenté, chercheur dans l'équipe ROOT à Polytech.

Dans le but de présenter un stand interactif lors des Portes Ouvertes de Polytech, M. Lente a pensé à faire découvrir au public un projet lié aux trompes l'œil et plus particulièrement l'anamorphose cylindrique. Le projet doit répondre à ce besoin en créant un outil permettant d'utiliser le principe de l'anamorphose de façon ludique et attrayante.

Ce document a été rédigé par A. Saludo et sera validé par les différents acteurs du projet.

Première partie

Objectif et État de l'art

1

Contexte de la réalisation

1 Contexte

De nos jours, l'anamorphose est une technique de déformation d'images ou d'objets utilisée à des fins essentiellement artistiques. Cette technique datant du XVII^{ème} siècle possède encore son intérêt aujourd'hui dans la représentation d'objets en trois dimensions à partir d'images planes.

Ce projet s'inscrit dans un contexte d'exposition. Prévue pour les journées portes ouvertes de l'école Polytech en mars 2017, l'application est destinée à être présentée à un public afin de montrer la réalisation de transformation d'images grâce à des formules mathématiques.

2 Objectif

L'anamorphose est une technique de déformation réversible d'image à l'aide d'un procédé optique (miroir courbe) ou par transformation mathématique. L'objectif de ce projet est de créer une application permettant de réaliser l'anamorphose cylindrique sur un flux vidéo. Ce projet a alors été séparé en plusieurs étapes qui seront réalisées au fur et à mesure de l'avancement du projet.

2.1 Anamorphose Cylindrique sur une image

La première étape de ce projet est de pouvoir effectuer cette transformation sur une image. L'utilisateur choisit une image et saisit les paramètres nécessaires à la création d'une image anamorphique. Celle-ci s'affichera alors à l'écran dans une seconde fenêtre. Il pourra ainsi lancer l'impression de l'image obtenue au format convenu.

2.2 Anamorphose Cylindrique sur un flux vidéo

La seconde partie du projet sera alors d'adapter la première application pour qu'elle puisse être utilisée avec une vidéo ou un flux en continue (comme une caméra) en entrée. L'utilisateur devra alors sélectionner la source afin de lancer le processus d'anamorphose. A tout moment l'utilisateur pourrait mettre le flux en pause pour effectuer l'impression de l'image déformée de ce flux à un instant donné.

2.3 Anamorphose Cylindrique d'un objet en 3D

Enfin, en fonction du temps restant, il sera envisagé de travailler sur la modélisation de l'anamorphose d'objet 3D. L'utilisateur devra alors spécifier un objet 3D dans un format accepté par le programme. Le programme créera alors un nouveau modèle 3D déformé qui pourrait être imprimé avec n'importe quelle imprimante 3D.

2

Description générale

Dans cette partie, nous aborderons de façon sommaire la méthode de transformation utilisée pour l'anamorphose d'une image. Puis, nous spécifierons les caractéristiques de l'utilisateur avant de définir les actions qu'il pourra réaliser sur notre application. Nous verrons ensuite les contraintes de développement et d'exploitation pour terminer sur l'architecture générale du projet.

1 Méthode de transformation

Afin d'effectuer notre transformation anamorphique, nous avons défini plusieurs paramètres à savoir le rayon du cylindre réfléchissant et la position de la personne observatrice (V sur **Figure 1**) par rapport à ce cylindre.

Une fois ces paramètres recueillis, on commence par redimensionner l'image source à déformer afin qu'elle tienne à l'intérieur du cylindre. Cela permet d'avoir une image entièrement visible dans la partie réfléchissante que regarde l'observateur.

Pour chaque pixel de l'image, on va alors chercher le point I appartenant au cylindre et correspondant à l'intersection entre ce pixel et la position V de l'observateur.

Grâce à la normale au cylindre en ce point, nous pouvons alors trouver les coordonnées réfléchies de notre image.

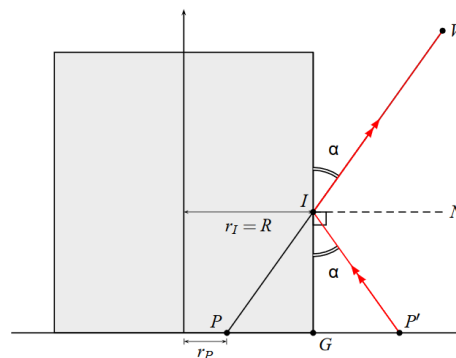


Figure 1 – Schéma de l'anamorphose

2 Caractéristiques des utilisateurs

Le système est destiné à être exposé dans des salons tels que les portes ouvertes de Polytech, mais les visiteurs n'auront pas de possibilité d'interaction directe avec l'application. Nous aurons donc un unique type d'acteur qui sera le seul à accéder à l'application :

- L'Administrateur :
 - Configure les paramètres pour l'anamorphose
 - Choisit la source qui sera utilisée

3 Fonctionnalités du système

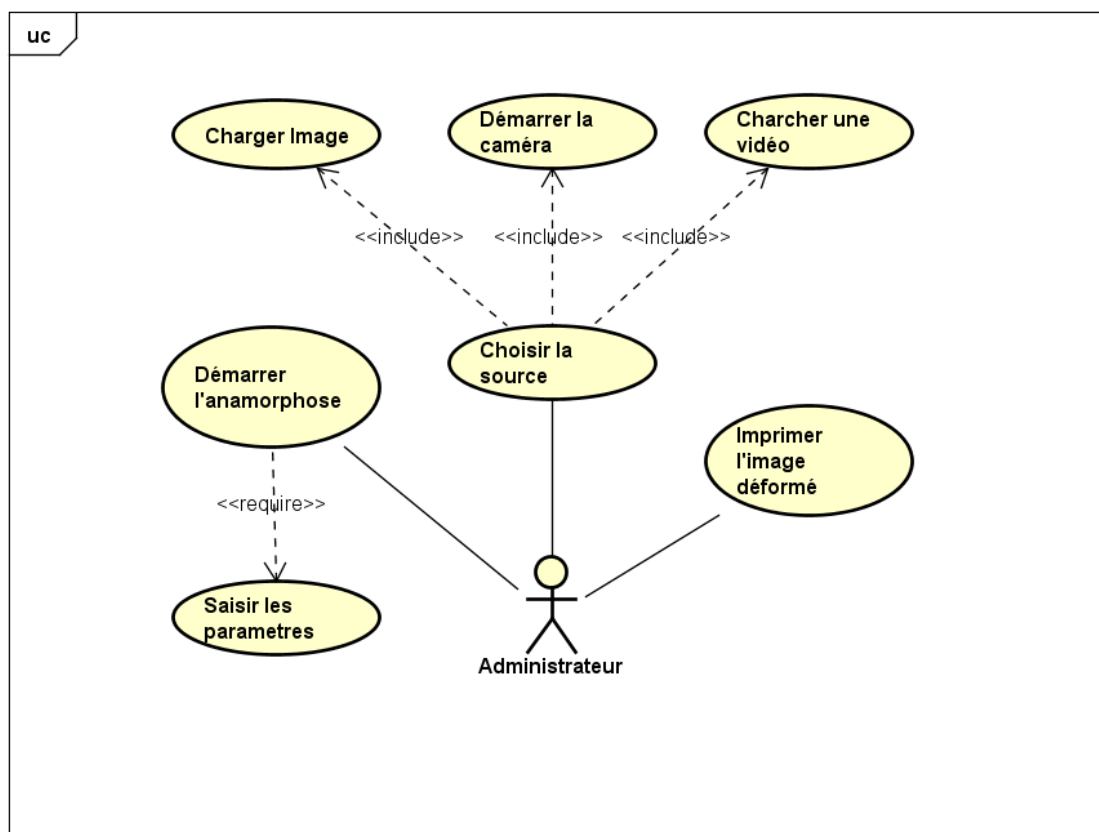


Figure 2 – Cas d'utilisation de l'application

L'utilisateur aura alors trois interactions différentes avec l'application.

Fonction 1 : Choisir la source

Afin d'effectuer l'anamorphose, on doit choisir la caméra ou le fichier sur lequel on va appliquer la déformation.

Fonction 2 : Démarrer l'anamorphose

Une fois le fichier chargé, on démarre le processus d'anamorphose avec les paramètres entrés par l'utilisateur.

Fonction 3 : Imprimer l'image déformée

Cette fonction permettra d'enregistrer l'image en cours d'affichage pour l'imprimer au format A4, si c'est possible, en fonction des paramètres saisis.

4 Contraintes de développement

Les différentes contraintes de développement auxquelles nous serons confrontés sont :

- Le langage de programmation : Java
- L'IDE utilisé : Eclipse Mars
- Délais de réalisation : Mars 2017.

5 Contraintes d'exploitation

Notre application travaille sur une version réduite de l'image car elle doit tenir dans le cylindre. Les contraintes sont alors sur le temps de déformation d'une telle image par rapport au nombre d'images par seconde affichées par la vidéo. Celles-ci sont liées à la puissance de calcul de la machine.

6 Structure générale du système

La solution sera divisée en plusieurs parties en suivant le modèle MVC :

- Une partie Modèle :
 - Les classes métiers en Java qui modéliseront le programme sous forme d'entités.
- Une partie Vue :
 - Elle va afficher les rendus de l'anamorphose et les formulaires pour l'utilisateur
 - Elle sera représentée par deux écrans. Le premier permettra le chargement du fichier de base ainsi que la saisie des paramètres. Le second quant à lui, affichera l'image de l'anamorphose seule. Tout ceci permet ainsi la projection d'image anamorphique sur un grand écran.
- Une partie Contrôleur :
 - Elle s'occupe de traiter les informations.
 - Elle s'interface avec la partie Modèle qui fournira les informations et la partie Vue pour l'affichage.
 - C'est dans cette partie que s'exécutera l'algorithme de l'anamorphose. Cet algorithme aura une interface pour pouvoir implémenter facilement si le fichier est une vidéo ou un fichier 3D

3

Etat de l'art

Dans ce chapitre, nous verrons tout d'abord les applications existantes autour de l'anamorphose cylindrique au cours du temps, puis nous aborderons le positionnement de ce projet par rapport à ces derniers. Enfin, nous terminerons sur la veille technologique effectuée dans le choix des librairies utilisées.

1 Recherche bibliographique

1.1 Jean Louis Vaulezard (1630)

Jean Louis Vaulezard est un géomètre français qui a écrit le livre « Perspective cylindrique et conique, concave et convexe ou traité des apparences vues par le moyen des miroirs, Paris ». C'est le premier à décrire les techniques et des explications sur les anamorphoses.

1.2 Anamorph Me ! (2001)

Anamorph Me ! est un logiciel développé en 2001 en C++ en utilisant une ancienne librairie d'images « paintlib ». Le logiciel fonctionnait sous les versions XP et Vista de Windows mais ne fonctionne plus à ce jour. Il permettait de réaliser les anamorphoses obliques, cylindriques et coniques.

1.3 Jonty Hurwitz (2008)

En 2008, Jonty Hurwitz a repris le principe de l'anamorphose cylindrique afin cette fois de réaliser la sculpture d'une grenouille.



Figure 1 – Sculpture de grenouille après anamorphose

1.4 Reactable (2008)

En 2008, le centre d'innovation numérique de Lyon amorce un projet nommé Erasme. Celui-ci utiliserait l'anamorphose cylindrique en lien avec l'utilisation d'une Reactable. La Reactable est une table qui détecte les objets qui lui sont posés dessus. Le but du projet était de détecter le cylindre afin d'afficher l'image déformée au bon endroit. Il était aussi envisagé d'afficher des images en 3D en tournant le cylindre. Cependant aucun rapport n'a été publié concernant l'avancé de ce projet.

1.5 Anamorphose et capture caméra (2008)

En 2008, M. Blossier, enseignant secondaire à Rouen, a développé une application Java à l'aide d'une librairie Java Media Framework, permettant de réaliser des anamorphoses à partir d'images ou de captures de caméra.

1.6 Anamorphicon (2008)

Une équipe de chercheuses d'université au Japon a développé une application sur Ipad mettant en œuvre l'anamorphose cylindrique en lien avec le multi-touch de l'appareil. L'application permet d'afficher un objet sous tous ces angles à partir de plusieurs photos (jusqu'à 70).

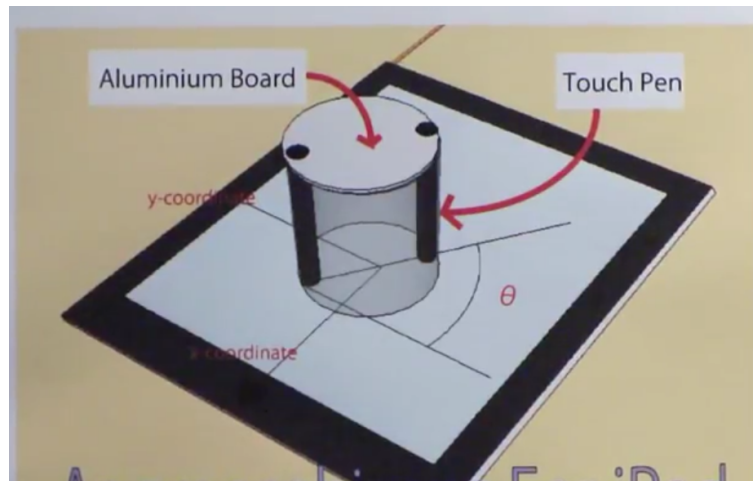


Figure 2 – Schéma du Projet Anamorphicon

Comme le montre la Figure 2, l'application utilise deux stylets à l'intérieur du miroir cylindrique. A partir des coordonnées calculées par les positions des stylets, le programme déduit les coordonnées du cylindre pour afficher l'image déformée au bon endroit. En pivotant le cylindre, l'application va alors déterminer quelle image afficher en fonction de l'angle (θ).

2 Positionnement du projet

Mon projet va consister à développer une nouvelle application Java en s'inspirant du programme écrit par M. Blossier. Tout d'abord, nous réaliserons une application similaire travaillant sur une image fixe. Ensuite, nous travaillerons sur un flux d'image en continu, pour terminer sur des fichiers 3D. Il n'existe à ce jour aucune application prenant en compte le traitement de flux d'images ou d'objets 3D.

3 Veille Technologique

Afin de réaliser ce projet, nous avons besoin d'intégrer des bibliothèques externes pour utiliser une caméra et des fichiers de type média.

3.1 Librairie Webcam

Plusieurs choix de bibliothèques pour Webcam étaient disponibles entre Java CV, OpenIMAJ, JMF. Mais une bibliothèque était basée sur tous les Framework cités précédemment. Cette bibliothèque s'appelle « Webcam Capture » et va permettre d'utiliser des caméras sur n'importe quelle plateforme ou architecture. Elle permet aussi de manipuler le flux d'images renvoyé par la caméra, ce qui va rendre faisable l'anamorphose sur ce flux.

3.2 Librairie Media Vidéo

Concernant le traitement des vidéos, la bibliothèque VLCj basé sur le logiciel VLC a été choisie car elle supporte un grand nombre de types de médias. Cette librairie permet aussi de décomposer la vidéo en un flux d'image afin de pouvoir la retravailler.

Deuxième partie

Développement

Lors de la phase de recherche, j'ai été amené à travailler sur un premier prototype de l'application qui devait être capable de transformer une image. Ce second semestre a donc été consacré à la mise en place d'amélioration de la fonction de transformation ainsi qu'à l'amélioration des performances et de l'utilisation des ressources.

4

Analyse

Notre application va donc permettre de charger un flux et de le transformer afin de l'afficher. On a alors trois tâches qui interagissent entre elles. Afin de mieux visualiser le projet, nous allons tout d'abord regarder un schéma général du fonctionnement de l'application. Puis, nous détaillerons ce schéma grâce à un diagramme de classe.

1 Schéma général de l'application



Figure 1 – Schéma général du projet

Tout d'abord, la première étape est de récupérer les images de notre flux. On commence donc par le décomposer en images successives que l'on stockera. On en profitera aussi pour afficher ce flux en direct afin de vérifier son contenu.

La seconde étape de notre application est de transformer les images sources enregistrées. Cette étape est certainement celle qui prendra le plus de temps dans notre application car on applique une formule mathématique sur chaque pixel de notre image.

Cependant, afin de visualiser notre image avec une bonne résolution dans notre cylindre, elle est redimensionnée selon le rayon du cylindre entré en paramètres. Une fois redimensionnée et transformée, l'image est alors stockée.

Enfin, la dernière étape de ce processus est la récupération des images déformées. Lors de cette étape, on peut aussi être amené à redimensionner notre image selon le zoom de notre application.

2 Diagramme de classe

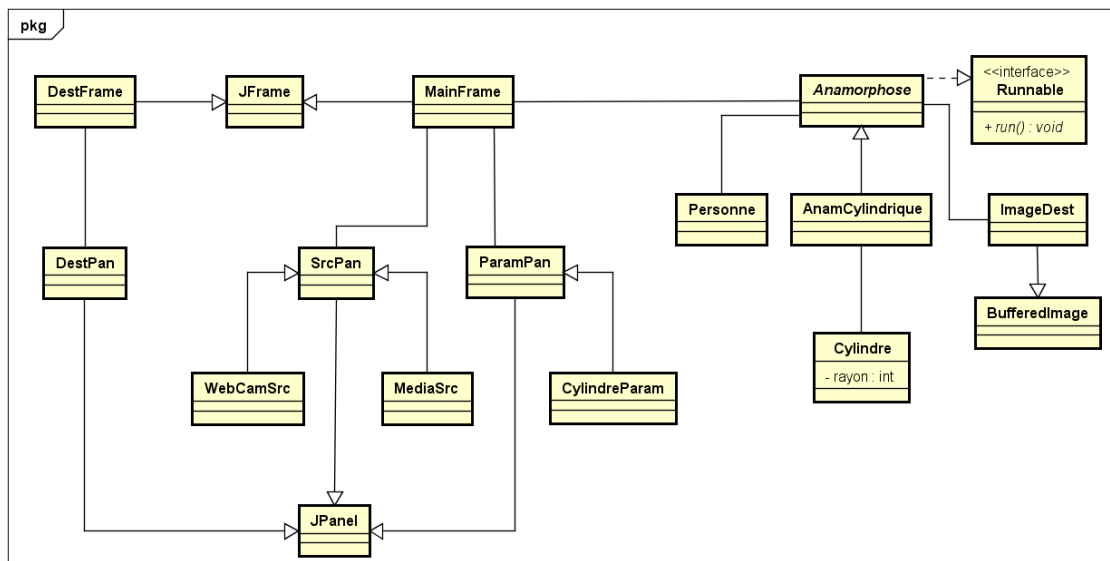


Figure 2 – Diagramme de classe du projet

Voici le diagramme de classe simplifié associé à notre application. Ce diagramme fût réalisé après une première analyse du prototype de l'application que j'ai réalisé. On retrouve alors nos trois parties.

La première, représentée par la classe **MainFrame**, est une fenêtre comportant deux **JPanel** à savoir celui de notre flux média **SrcPan** et celui qui permet de saisir les paramètres de notre anamorphose "**ParamPan**". Ces deux classes sont déclarées **Abstract** et il faudra donc passer par des classes en héritage pour les utiliser.

La seconde partie de notre application est représentée par la classe **Anamorphose**, contenant nos algorithmes de transformation d'images ainsi que les paramètres utile à celle-ci. Cette classe est aussi définie **Abstract** et ne pourra pas être instanciée. On utilisera alors une de ses classes filles : dans notre cas la classe **AnamCylindrique**.

La dernière partie est celle de l'affichage de notre image déformée. Elle est représentée par notre classe **DestFrame**. En plus d'afficher l'image déformée et la position de notre cylindre sur l'image, elle donnera la possibilité de zoomer ou dézoomer notre affichage.

Le fait de passer par des classes **Abstraite** permet d'introduire une certaine modularité dans notre application ce qui permettra entre autre de pouvoir ajouter d'autres algorithmes de déformation ou flux d'entrée.

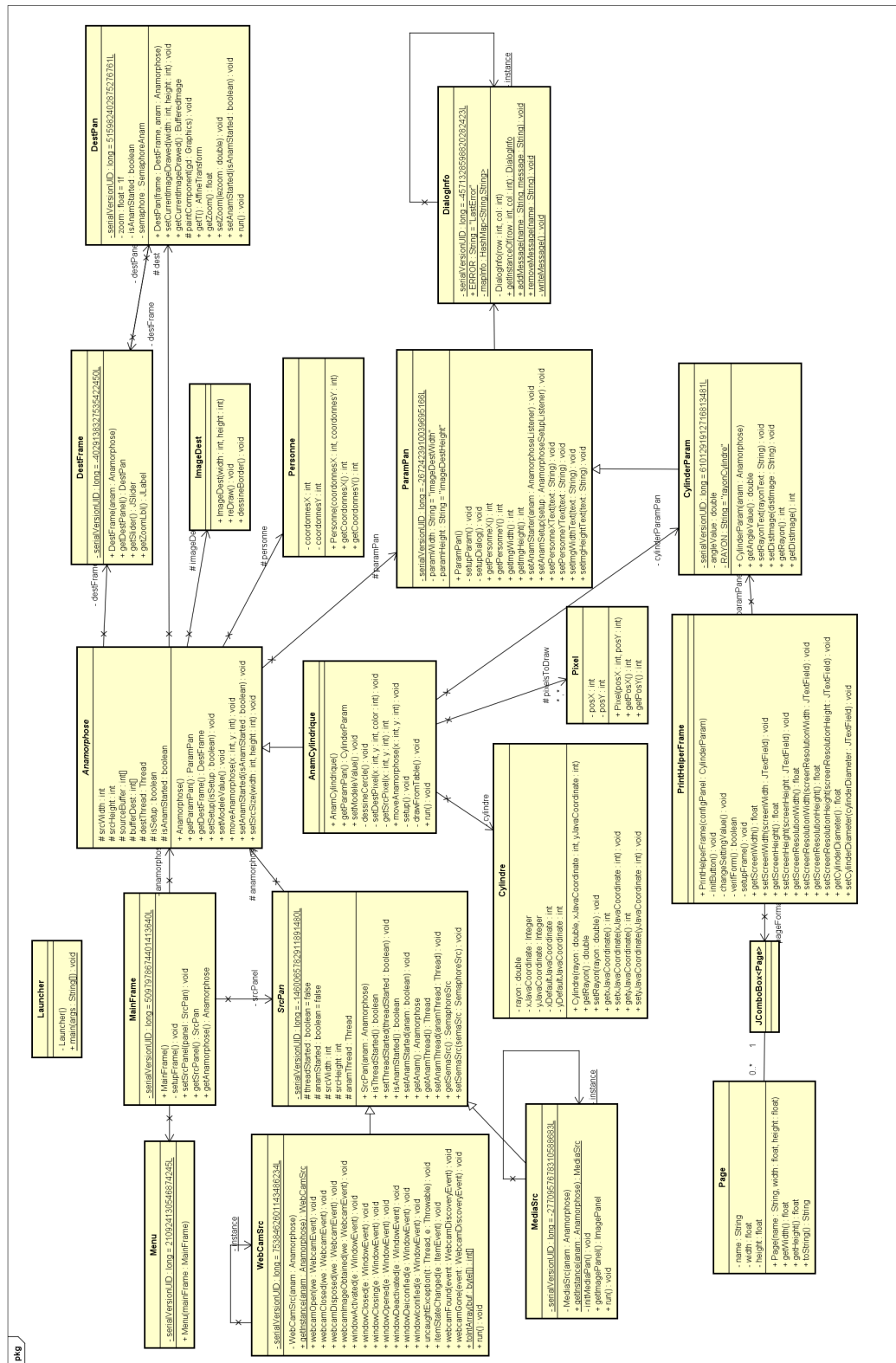


Figure 3 – Diagramme de classe complet du projet

Voici le diagramme complet de mon application. Ce diagramme fut généré à partir du code écrit pendant la phase de développement. On retrouve toujours nos trois parties mais on remarque que certaines classes telles que l'aide à l'impression ou le panneau d'information sont apparus. Ces classes n'avaient pas été définies dans la première partie de l'analyse car elles dépendaient essentiellement de la mise en place du cœur de l'application.

Enfin certaines classes ont été modifiées par rapport au diagramme réalisé après le prototype. C'est le cas du panneau de configuration qui est maintenant fourni par l'anamorphose cylindrique car c'est elle qui définit son implémentation de la classe Abstraite et qui va avoir besoin des valeurs de ce dernier.

3 Amélioration de l'application

Après la mise en place du premier prototype, j'ai donc réalisé un premier diagramme de classe permettant la création du projet. Ce diagramme m'a permis d'établir les bases de l'application concernant les classes principales à mettre en œuvre sans pour autant couvrir toutes les méthodes nécessaires à son fonctionnement. Après avoir implémenté ce nouveau diagramme de classe tout en gardant la même gestion des flux définie dans le prototype, nous avons pu avec mon encadrant tester l'application sur le matériel.

Il s'est avéré que les ressources matérielles n'étaient pas suffisantes en terme de capacité CPU et mémoire vive pour faire fonctionner l'application. Il a donc fallu effectuer certains changements pour améliorer cette première version afin qu'elle puisse convenir au dimensionnement de notre appareil.

3.1 Amélioration de l'utilisation des Threads

Mon projet utilise plusieurs Thread pour l'affichage des images sources et des images transformées mais aussi pour la transformation. Pour ma première version de l'application, j'ai alors mis en place des files d'attente d'images avec plusieurs Thread. Cette solution a alors été testée sur le matériel mis à disposition pour les tests et ne donnée pas satisfaction. L'application souffrait de ralentissement car les spécification de la machine de test n'était pas connu au moment des spécifications.

La solution qui a été retenue pour gérer au mieux ce problème fut de mettre en place des Sémaphores. Cette solution permet entre autre de mettre en place des sauts d'images si la transformation est trop lente. Le Thread permettant de réaliser l'affichage des images sources et de transformé étant le même géré nativement par Java, l'application était alors fait de 2 Threads.

La seconde solution qui fut mise en place fut d'abandonner la parallélisation de la fonction de transformation. La machine permettant de réaliser les tests n'ayant que très peu de ressource CPU, il était donc inutile de tenter de paralléliser cette tâche d'autant plus que mon application fonctionne déjà en parallèle avec le Thread de Java AWT qui est utilisé pour les interfaces.

3.2 Amélioration de l'utilisation en ressource

Le prototype et la première version du projet travaillaient avec des instances de `BufferedImage`. Pour chaque images de mon flux que cela soit un média ou de la caméra, il y avait donc autant d'images créées que d'images dans la source. Sur la machine de développement cela ne posait aucun problème mais sur la machine de tests cela ralentissait considérablement l'application. La machine n'ayant pas assez de ressources mémoire devait créer de l'espace sur les disques durs ralentissant ainsi l'application mais aussi le système d'exploitation tout entier.

La solution trouvée à ce problème fut de travailler sur des tableaux d'entier au lieu de travailler sur des instances d'images. La recopie d'image était une opération extrêmement coûteuse qui est à utiliser de façon ponctuelle mais pas dans le cas du traitement d'une vidéo. En passant par une recopie de tableaux de nombres entiers, le programme a fini par consommer moins de ressources permettant ainsi d'être fonctionnel sur la machine de test.

3.3 Amélioration de la fonction de transformation

Lors de la création du prototype, il a été défini que nous travaillerons sur des flux vidéo. Pour cela nous récupérerons donc chaque image sur laquelle nous appliquons la transformation d'anamorphose cylindrique.

Or toutes nos images ont la même taille dans notre flux vidéo et donc la transformation fournie les mêmes résultats en termes de coordonnées pour chaque images. Afin de réduire les temps de calculs, j'ai alors mis en place un tableau de coordonnées afin de stocker toutes les valeurs des résultats pour la première image transformée et pour les images suivantes il n'y a plus de calculs à effectuer mais simplement la lecture du tableau.

Cette technique a permis de gagner du temps pour la transformation de l'image rendant ainsi la phase de transformation plus courte.

La seconde amélioration que j'ai apporté à l'application fut de modifier la fonction de transformation afin de travailler sur l'image source au lieu de travailler sur une image réduite. Cela a permis de résoudre en partie le problème de pixel manquant en sur-échantillonnant la transformation car l'image transformée doit normalement être contenue dans le cylindre.

Or avec cette technique on transforme une image plus grande mais on applique la réduction sur les résultats de notre transformation. Ainsi il est possible que certain pixel se superposent.

4 Ajout de nouvelles fonctionnalités

Après la création de la première version de l'application, plusieurs fonctionnalités ont été réfléchies afin d'améliorer le système et de le rendre plus interactif.

4.1 Déplacement du miroir par contact tactile

Cette première fonctionnalité a été proposée par Mr. Venturini après la première soutenance de ce projet. Afin de rendre l'application interactive avec les visiteurs, il a été décidé d'implémenter une fonction utilisant la partie tactile de notre écran. Le principe est le suivant, lors d'un déplacement du cylindre sur l'écran, la transformation suit le cylindre.

Le tactile de l'écran permet la simulation du clic gauche par contact avec l'écran. J'ai alors réalisé une fonction qui réagit au clic de la souris sur l'écran afin de déplacer le centre du cylindre servant à la transformation sur l'image.

4.2 Modification de l'affichage

Cette fonctionnalité a été mise en place en lien avec la recherche de solution pour combler les pixels manquants après la transformation. Afin de pouvoir réduire l'apparition de ces trous dans la déformation, j'ai réalisé une fonction capable de réduire l'angle d'affichage de la transformation sur le cylindre entre 0 et 180 degrés.

Pour cela j'ai tout d'abord rajouté un curseur qui permet de choisir l'angle désiré sur l'interface de configuration . Ainsi en le bougeant, j'effectue un ratio sur la transformation afin de la réduire.

Ainsi en réduisant suffisamment la transformation par rapport au sur-échantillonnage, nous arrivons à empêcher l'apparition de trous dans l'image.

4.3 Paramétrage en direct

Cette fonctionnalité découle de la fonction précédente. Lors du déplacement du curseur pour modifier l'angle d'affichage, c'est tout le processus de transformation qui est effectué de nouveau.

Or il semblait illogique de pouvoir effectuer ce processus en pleine transformation seulement en bougeant le curseur d'affichage ou le cylindre. Ainsi, j'ai ajouté un bouton qui notifie au processus de transformation qu'il doit refaire la mise en place des configurations par rapport aux nouvelles valeurs du panneau prévu à cet effet.

4.4 Aide à l'impression et l'export d'image

Une fonctionnalité d'impression de l'image transformée avait été définie au début du projet mais fut reportée car je n'ai jamais eu le matériel à disposition pour la réaliser. Après avoir réalisé toutes les fonctionnalités prévues ainsi que celles mentionnées ci-dessus, j'ai réalisé une fonction d'export de l'image courante affichée sur la fenêtre d'images transformées. Cette fonction copie alors l'image affichée et l'enregistre dans un dossier qui est défini dans le fichier de configuration.

Cette fonction devait alors remplacer celle d'impression comme alternative mais il restait un problème avec la résolution de l'image. Selon les dimensions entrées par l'utilisateur, l'image exportée pouvait ne pas être de la dimension du format d'une page et ainsi lors de l'impression ne plus convenir au cylindre utilisé. C'est pourquoi je fus amené à réaliser un formulaire d'aide pour trouver les valeurs adaptées à l'impression au format A4.

Pour cela l'utilisateur doit renseigner certaines informations quant à la résolution de l'écran et ses dimensions réelles. Ainsi en validant le formulaire, le panneau de configuration est mis à jour afin que l'utilisateur puisse réaliser à nouveau sa transformation.

5

Qualité du code

1 Documentation

Une partie de cette phase de développement fut consacrée à la documentation de l'application d'anamorphose Cylindrique.

1.1 Guide utilisateur

Afin que le programme puisse être utilisé par une personne n'ayant aucune connaissance en informatique, j'ai réalisé une documentation utilisateur. Elle contient plusieurs parties en commençant par la manière d'exécuter le programme, suivi d'une description détaillée de l'interface utilisateur, et enfin les différentes actions possibles avec cette application.

Cette documentation permet alors à n'importe quel utilisateur possédant le fichier exécutable de pouvoir utiliser l'application sur sa machine.

1.2 Documentation développeur

Cependant, dans le but de pouvoir continuer le développement et comprendre le fonctionnement de celui-ci, un cahier de développeur fut réalisé. Ce document comprend alors plusieurs parties.

Tout d'abord, j'explique comment reprendre le projet en commençant par énoncer les prérequis, tel que la version de Java ou l'IDE que j'ai utilisé. Je parle ensuite de l'installation du projet en détaillant le contenu des différents dossiers qu'il comporte.

Dans une deuxième partie, je parle de l'implémentation de mon application en commentant mon diagramme de classe complet. C'est ici que je détaille alors mes choix d'implémentation pour chacune de mes classes et que j'introduis les différents éléments à améliorer en vue d'un prochain développement.

Ensuite, je décris les éléments de qualité logiciel que j'ai mis en place ainsi que la façon de les réutiliser. Cette partie permettra aux futurs développeurs de mettre plus facilement en place la partie qualité logiciel.

Enfin, je décris la façon dont j'ai disposé le projet en plusieurs versions ainsi que le moyen d'accéder au dossier les contenant.

1.3 Commentaire dans le code

Après avoir mis en place la version définitive de l'application, j'en ai commenté entièrement le code. L'application étant créée entièrement pour ce projet, il était important de générer une telle documentation.

En plus, elle contient de nombreuses classes complexes telles que celle comportant la fonction de transformation par anamorphose cylindrique, ainsi que celles définissant nos deux IHM principales. J'ai donc généré la JavaDoc qui est un document web contenant toutes les classes de mon projet avec les commentaires que j'ai écrit.

2 Tests

Après avoir terminé l'application avec toutes ses fonctionnalités, j'ai été amené à travailler sur les tests. Certains d'entre eux ont tout de même été effectués tout au long du développement afin de vérifier la non-régression de l'application.

2.1 Test de bon fonctionnement

Tout au long du développement de l'application, des tests succins et non répertoriés ont été réalisés visant à vérifier le fonctionnement des fonctionnalités qui avaient été mises en place. Le projet ayant été divisé en plusieurs phases de prototypage, certaines fonctionnalités ont subi de grosses modifications, en particulier celle de la transformation.

A chaque modification, les tests suivants ont été effectués :

- Ouverture de l'application avec le lecteur vidéo en source et un fichier de configuration
- Test de chargement d'une vidéo
- Test de la transformation par clic sur le bouton « Transformé »
- Test de la fonction de déplacement de la transformation par clique sur l'affichage
- Répéter les tests précédents avec le panel de la caméra

Certains de ces tests ont été effectués en parallèle de l'utilisation de l'application Java Mission Control afin d'évaluer les performances de l'application.

2.2 Test unitaire

Lors de la conception de l'application plusieurs tests unitaires ont été pensés, essentiellement sur les différentes parties comportant des entrées utilisateur ; Par Exemple le panneau de configuration qui est la principale source d'entrées.

La seconde fonction qui fut testée est l'utilisation du fichier de configuration, qui est important pour la mise en place de l'interface utilisateur. Enfin j'ai testé la fenêtre d'aide à l'impression. L'ensemble de ces tests sont répertoriés en annexes dans le « Cahiers de Tests ».

3 Métriques

3.1 Sonar

Lors de mon Projet de Recherche et développement, j'ai été amené à mettre en place un outil de contrôle de qualité de mon logiciel. Cet outil va alors analyser tous mes fichiers et m'afficher sur une page web les différents problèmes majeurs de mon application, des soucis de sécurité et des soucis mineurs tels que des erreurs de syntaxes.

Voici l'écran présentant un résumé de la qualité de mon code après une seconde analyse :

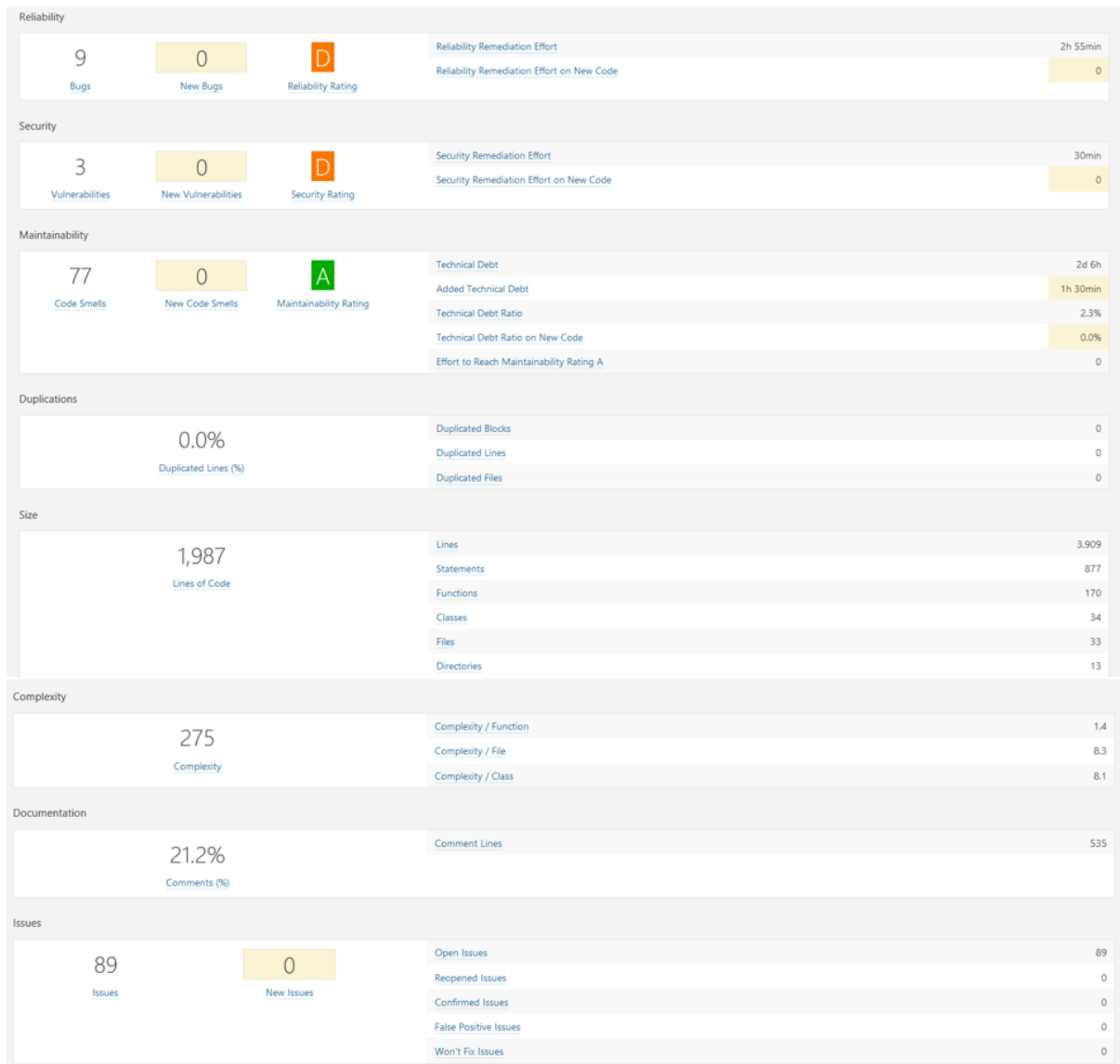


Figure 1 – Metrique Sonar Avant correction

Ce rapport nous indique donc à droite le temps que l'outil estime pour corriger les différents problèmes. En cliquant sur le lien « Bugs », vous êtes redirigé vers la liste des bugs.

The screenshot displays the SonarQube interface with the following details:

- Top Bar:** Shows "1 / 20" items and buttons for "Reload" and "New Search".
- Project Navigation:** A sidebar on the left shows the project structure: "JavaProject" and "src/polytech/projet/prd/anamorphose/lock/SemaphoreAnam.java".
- Bug List:**
 - Bug 1:** "Either re-interrupt this method or rethrow the 'InterruptedException'." (Critical, Closed/Fixed, 15min effort, CWE: cwe, multi-threading).
 - Bug 2:** "Either re-interrupt this method or rethrow the 'InterruptedException'." (Critical, Closed/Fixed, 15min effort, CWE: cwe, multi-threading).
 - Bug 3:** "Make 'currentImageDrawed' transient or serializable." (Critical, Closed/Fixed, 30min effort, CWE: cwe, serialization).
 - Bug 4:** "Cast one of the operands of this division operation to a 'double'." (Critical, Closed/Fixed, 5min effort, CWE: cwe, sans-top25-risky).
 - Bug 5:** "Cast one of the operands of this division operation to a 'double'." (Critical, Closed/Fixed, 5min effort, CWE: cwe, sans-top25-risky).
 - Bug 6:** "Use a StringBuilder instead." (Minor, Closed/Fixed, 10min effort, CWE: performance).

Figure 2 – Liste de bug trouvé par Sonar Avant correction

Sur cette fenêtre, nous avons alors plusieurs informations tel que l'importance du bug son état, son assignation et l'évaluation du temps à passer pour la corriger. Afin de corriger l'erreur, il suffit de cliquer sur la flèche à droite pour que Sonar nous redirige sur le fichier et la ligne de code qui pose problème. En cliquant sur les « ... » à droite du titre de l'erreur, Sonar nous affiche plus de détails quant à la manière de résoudre le problème.

Après plusieurs versions de mon code, je suis arrivé à corriger les erreurs mais j'ai aussi augmenté le pourcentage de commentaire.

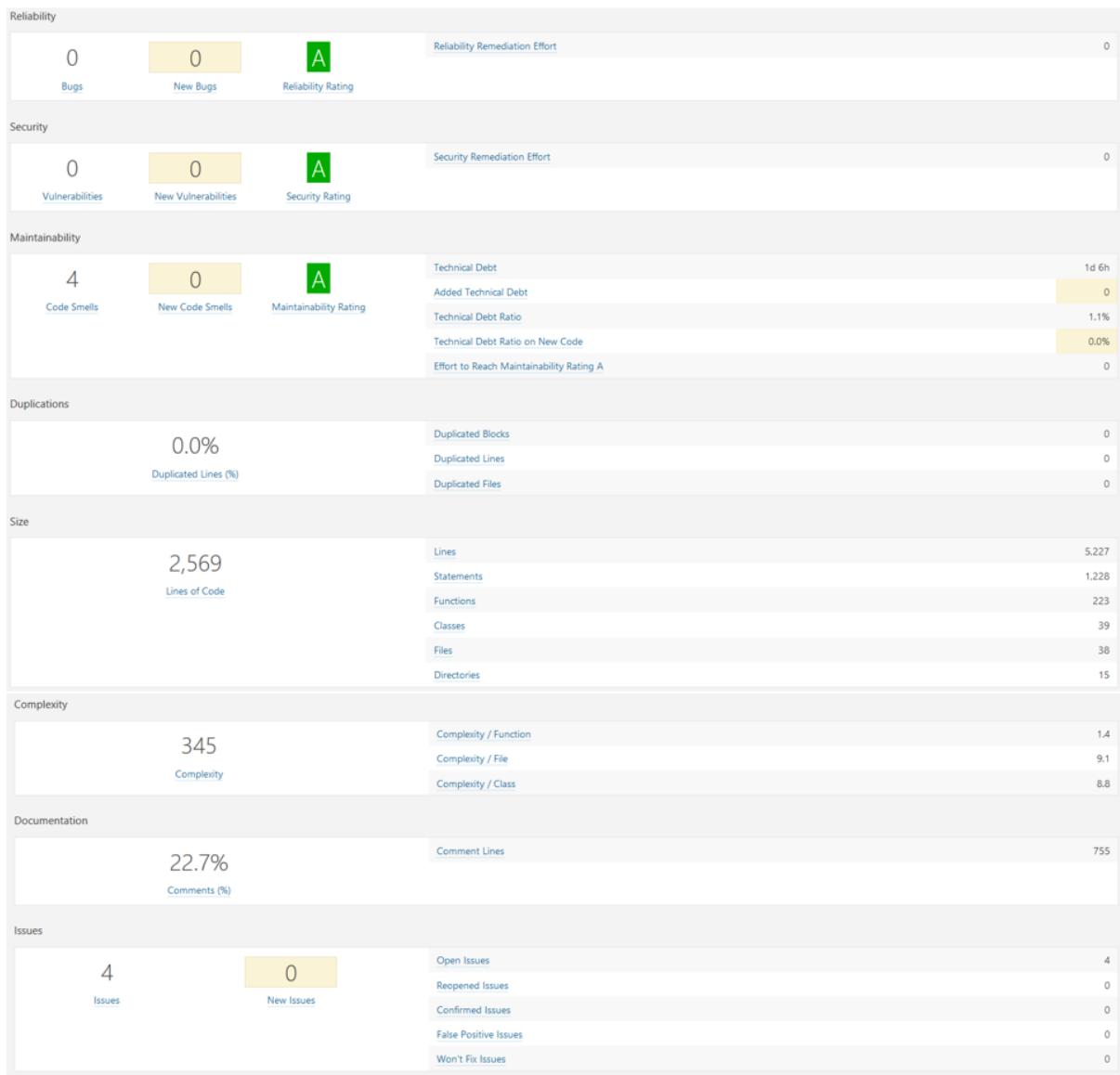


Figure 3 – Metrique Sonar Après correction

On peut tout de même voir qu'il reste 4 « Code Smell » qui sont des erreurs qui n'impactent pas la sécurité de l'application mais qui, dans mon cas, impacte sa complexité. Les erreurs sont que je devrais remplacer mes héritages des classes JPanel et JFrame par l'utilisation d'une variable à l'intérieur de celle-ci.

3.2 Java Mission Controle

Pendant le projet, j'ai été confronté à des problèmes de performance de mon application. Dans le but de fonctionner sur une machine avec peu de ressources, le projet devait être un minimum performant.

Afin de faire cette analyse, j'ai utilisé l'application Java Mission Contrôle qui permet d'avoir des informations quant à l'utilisation mémoire et CPU de notre application Java.

De plus, ce logiciel nous permet d'avoir des informations sur les parties du code qui ont généré des ressources mémoire. Grâce à ces informations il m'a donc été facile de voir ce qui n'allait pas et de facilement corriger mes erreurs.

Pour lancer une analyse avec Java Mission Contrôle il faut tout d'abord lancer l'application Java Mission Contrôle puis lancer votre programme Java et démarrer un enregistrement. Il est alors demandé d'entrer une durée d'enregistrement.

Voici alors comment je vais réaliser mes tests de performance :

- 30 secondes d'enregistrement de l'application à vide
- 30 secondes d'enregistrement avec la lecture d'une vidéo
- 30 secondes d'enregistrement avec la lecture d'une vidéo et la transformation
- 30 secondes d'enregistrement avec la lecture d'une vidéo et la transformation en bougeant la position de la transformation
- 30 secondes d'enregistrement en coupant la transformation
- 30 secondes d'enregistrement avec la caméra après la vidéo
- 30 secondes d'enregistrement avec la caméra et la transformation
- 30 secondes d'enregistrement avec la caméra et la transformation en bougeant la position de la transformation
- 30 secondes d'enregistrement en coupant la transformation

Soit 4 minutes 30 secondes d'enregistrement

Le premier enregistrement va être effectué sur la version 1 de mon code datant du 25 Janvier 2017. Pour une raison qui m'est inconnu je n'ai aucun moyen d'avoir les informations sur l'utilisation CPU de l'application.

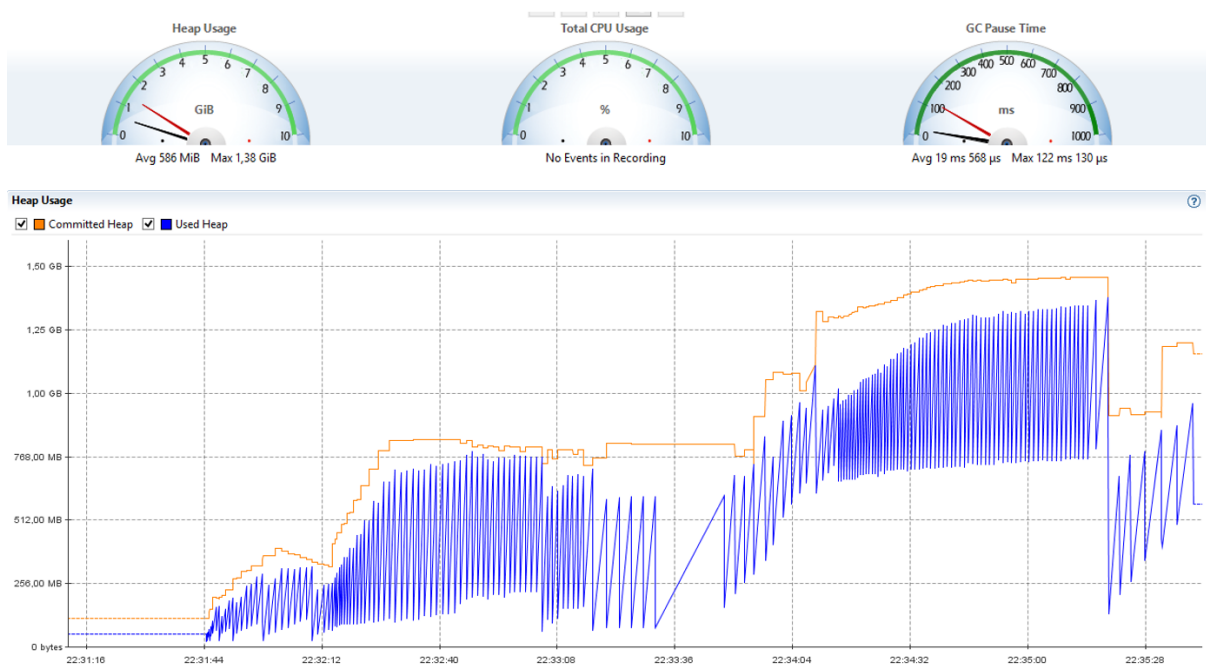


Figure 4 – Metrique JMC avant correction

Voici le premier graphique d'utilisation de la mémoire de l'application. On voit bien que l'application consomme énormément de mémoire, c'est d'ailleurs pourquoi le message suivant s'est affichée dans la console après l'enregistrement :

"Caused by : java.lang.OutOfMemoryError : Java heap space"

En moyenne l'application a donc utilisé 586 Mb de mémoire et avec une utilisation maximal de 1.3 Gb ce qui est très important pour ce genre d'applications.

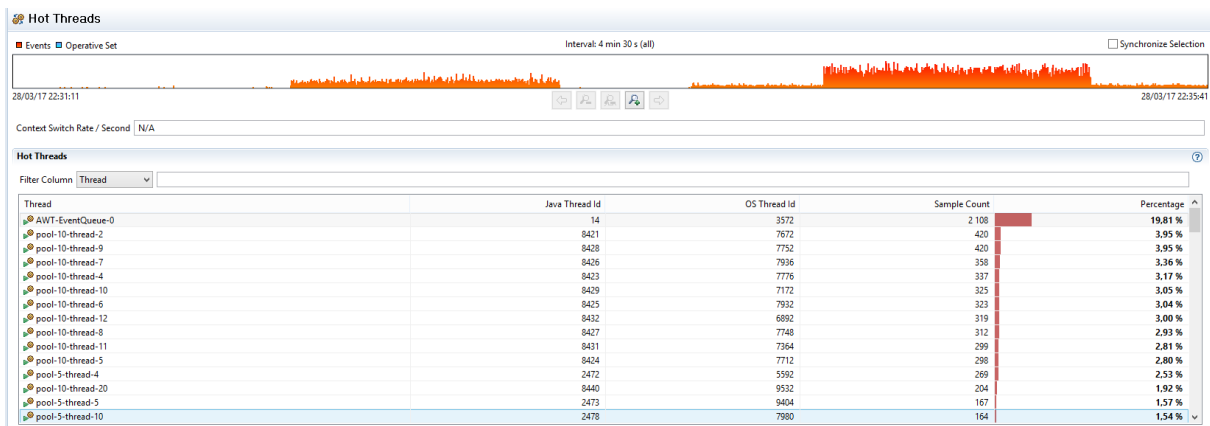


Figure 5 – Metrique JMC avant correction

On peut voir sur cet écran, que mon programme utilise la parallélisation avec un pool de thread. Mais que néanmoins le thread qui va travailler le plus est celui de l’affichage Java.

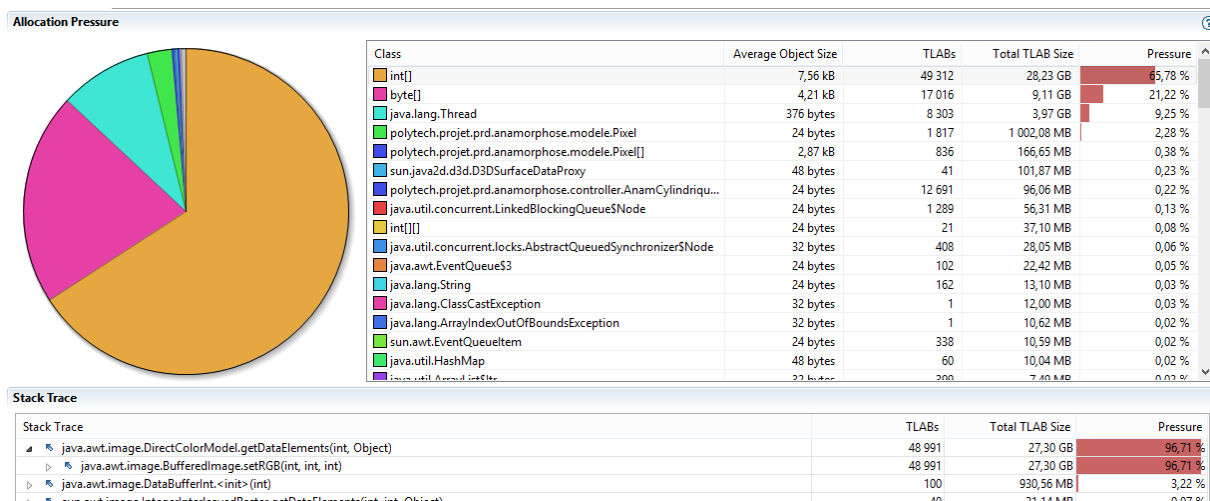


Figure 6 – Metrique JMC avant correction

Cet écran permet de voir quels types de ressource ont été générés ainsi que la méthode qui les a créés. Dans notre cas, on voit que c’est la méthode setRGB de BufferedImage qui génère énormément de donnée à savoir 27GB en 4 min 30s.

C’est donc à partir de ces informations mais aussi celle du CPU en regardant en continue l’information de l’utilisation courante de l’application qui montait tout de même à 100 % de mon CPU que j’ai commencé à travailler à l’amélioration des performances.

Voici les résultats de la dernière version de mon programme avec le même enregistrement :



Figure 7 – Metrique JMC après correction

On voit déjà que l'application utilise beaucoup moins de ressources avec le lecteur média mais que c'est toujours avec la caméra que la consommation est plus importante.

Cependant, celle-ci utilise en moyenne 182 Mb et au maximum 800Mb. En comparaison avec la première version, elle consomme en moyenne 3 fois moins et au maximum 2 fois moins.

De plus, on remarque que le nombre de passages du Garbage Collector en Bleu est beaucoup moins élevé dans cette deuxième version de l'application pour la partie avec l'utilisation de la caméra mais qu'elle l'est plus avec l'utilisation de VLC.

Hot Threads					
Filter Column Thread					
Thread	Java Thread Id	OS Thread Id	Sample Count	Percentage	
Thread-9602	9639	10452	4 583	<div></div>	50,04 %
AWT-EventQueue-0	14	10940	3 090	<div></div>	33,74 %
Thread-2438	2471	8012	1 273	<div></div>	13,90 %
Thread-9605	9646	4384	143	<div></div>	1,56 %
pool-2-thread-1	22	7560	16	<div></div>	0,17 %
webcam-panel-scheduled-executor-1	9644	1744	13	<div></div>	0,14 %
...

Figure 8 – Metrique JMC après correction

Après avoir pris connaissance des caractéristiques de la machine qui allait être utilisée pour tester l'application, j'ai tout de suite changé d'approche et arrêté la parallélisation de la transformation. Nous avons donc avec cet écran l'affichage en premier, du Thread qui s'est occupé de gérer la caméra, puis le thread Java qui affiche les images dans les panneaux, et enfin le Thread qui gère l'anamorphose en 3e position.

On constate donc que le thread de transformation est utilisé moins souvent que celui de la caméra.

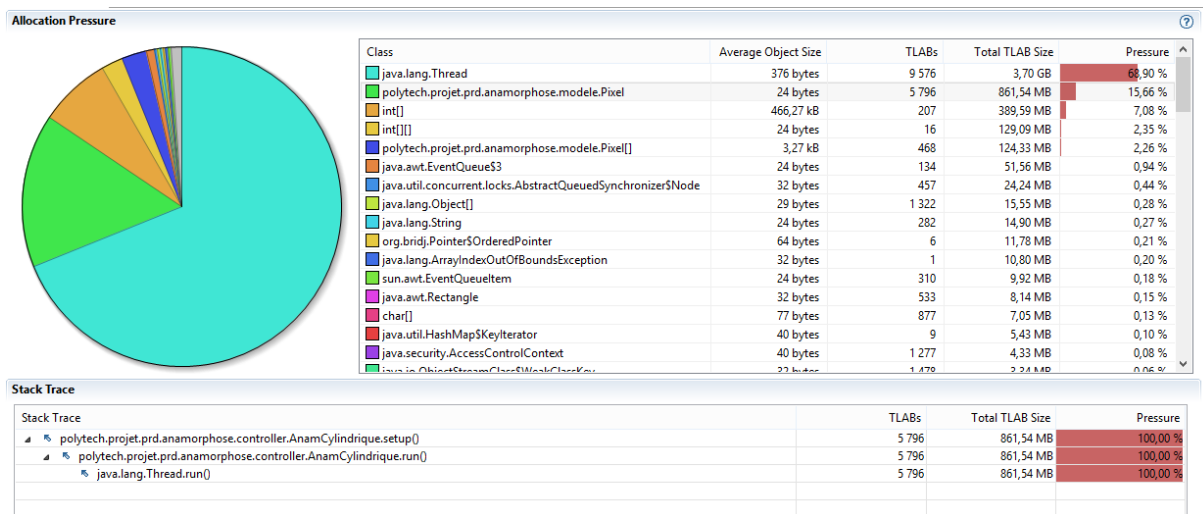


Figure 9 – Metrique JMC après correction

Enfin, voici le diagramme présentant les ressources générées par l'application. On remarque avec surprise que la quantité totale d'éléments générés est d'environ 4 GB au total et que c'est en grande partie à cause de la création de Thread. Nous ne pouvons néanmoins pas savoir quelle méthode a créé ces Threads mais grâce à d'autres analyses, j'ai pu déterminer qu'il s'agissait de la librairie VLC qui crée un thread pour chaque image de notre vidéo afin de pouvoir la récupérer dans le fichier.

En seconde position se trouve notre classe Pixel qui a été générée 900 Mb de données : c'est autant que dans la version 1. Ces données sont générées à chaque fois que nous modifions la position du miroir car entre temps la source d'image aurait pu changer.

6

Bilan

1 Etat actuel du projet

En l'état actuel, le projet fut mené à terme. Toutes les fonctionnalités demandées ont été réalisées. Cependant certaines améliorations peuvent tout de même être réalisées afin de corriger certains problèmes d'affichage et à aider l'ajout de fonctionnalités. Ces améliorations ont été listées dans le cahier de développeur.

En ce qui concerne la qualité du projet, l'application a passé plusieurs tests de qualités tant sur le code que ces performances. Néanmoins, certaines améliorations peuvent être apportées sur ce sujet.

Ce projet contient 5 classes avec une profondeur d'héritage trop importante à cause de l'héritage d'une classe Java. Il serait alors possible de réduire cette profondeur en passant d'un héritage à l'implémentation en attribut.

D'autre part, en ce qui concerne les performances, elles peuvent encore être améliorées en travaillant cette fois sur une utilisation plus approfondie des bibliothèques car ce sont elles qui impactent le plus sur les performances.

Concernant la gestion de projet, elle fut correctement menée à quelques erreurs près dans le planning qui furent d'ailleurs plutôt favorable. Certaines tâches ont été réalisées plus rapidement que prévues permettant ainsi de rajouter des fonctionnalités supplémentaires. Un bilan plus détaillé de la gestion de projet se trouve en annexe.

2 Conclusion

En conclusion, ce projet m'a permis d'approfondir mes connaissances en programmation Java mais surtout dans la mise en place de contrôle qualité avec l'utilisation de Sonar et Java Mission Control. J'ai alors pu travailler sur la mise en place d'une application de l'écriture des spécifications jusqu'à sa mise en place aux portes ouvertes de Polytech durant lesquelles j'ai pu présenter le projet au public.

Ce projet pourra être utilisé à nouveau pour d'autres portes ouvertes, mais est aussi exploitable afin d'ajouter de nouvelles fonctionnalités tel que de nouvelles transformations.



Bibliographie

Formule mathématique

Auteur : Jürgen Gilg, Manuel Luque, Jean-Michel Sarlat

Titre : Documentation sur les anamorphoses cylindriques

Année : 2011

URL : <http://melusine.eu.org/syracuse/G/pstricks/Anamorphoses/anamorphose02/pst-anamorphosis-doc.pdf>

Langue : Français

Organisation : Syracuse

Date de visite : 2016-12-07

Etat de l'art

Auteur : Jean-Louis Vaulezard

Titre : Perspective cylindrique et conique, concave et convexe ou traité des apparences vues par le moyen des miroirs, Paris

Année : 1630

URL : https://fr.wikipedia.org/wiki/Jean-Louis_Vaulezard

Langue : Français

Date de visite : 2016-12-07

Auteur : Philip Kent

Titre : Anamorph Me! Software

Année : 2001

URL : <http://www.anamorphosis.com/software.html>

Langue : English

Date de visite : 2016-12-07

Auteur : Jonty Hurwitz

Titre : Sculpture d'une grenouille après anamorphose cylindrique

Année : 2008

URL : <http://www.jontyhurwitz.com/anamorphosis>

Date de visite : 2016-12-07

Auteur : Yann

Titre : Les anamorphoses : le projet

Année : 2008

URL : <http://www.erasme.org/Les-anamorphoses-le-projet>

Langue : Français

Organisation : Erasme

Date de visite : 2016-12-07

Auteur : Mathieu Blossier

Titre : L'illusion conique

Année : 2008

URL : <http://numerisation.irem.univ-mrs.fr/R0/IR008001/IR008001.pdf>

Langue : Français

Date de visite : 2016-12-07

Auteur : Nancy Owano

Titre : Researchers showcase cylindrical mirror on iPad

Année : 2011

URL : <http://phys.org/news/2011-10-showcase-cylindrical-mirror-ipad.html>

Langue : English

Organisation : Phys.org

Date de visite : 2016-12-07

Librairies

Auteur : Bartosz Firyn

Titre : Webcam Capture Librairies Caméra

Année : 2015

URL : <http://webcam-capture.sarxos.pl/>

Langue : Anglais

Date de visite : 2016-12-07

Auteur : Mark Lee

Titre : VLCj Librairies vidéo

Année : 2016

URL : <http://capricasoftware.co.uk/#/projects/vlcj>

Langue : Anglais

Date de visite : 2016-12-07

Annexes

Spécification

1 Spécifications fonctionnelles

Fonction 1 : Choisir la source pour l'anamorphose

Entrée : un fichier image ou vidéo ou un flux continue

Sortie : chargement du fichier dans l'application

Contraintes :

Référence :

Fonction 2 : Démarrer l'algorithme d'anamorphose

Entrée : le fichier chargé dans l'application

Sortie : le fichier modifié par l'anamorphose

Contraintes :

Référence :

Fonction 3 : Imprimé l'image déformée

Entrée : le fichier modifié par l'anamorphose

Sortie : un fichier imprimable

Contraintes : le fichier de base devra tenir sur une feuille de type A4

Référence

2 Interface matériel / logiciel

Il est possible que le projet intègre l'utilisation de moyens de capture d'image comme une caméra. L'application pourra s'interfacer avec un écran tactile branché en HDMI pour projeter l'image déformée.

3 Interface homme / machine

Nous créerons nous-même l'interface homme / machine qui conviendra à l'utilisation de l'application.

Les vues ci-dessous sont proposées à titre indicatif et ne représentent pas en soit l'aperçu du produit final mais seulement pour exposer des idées d'interface.

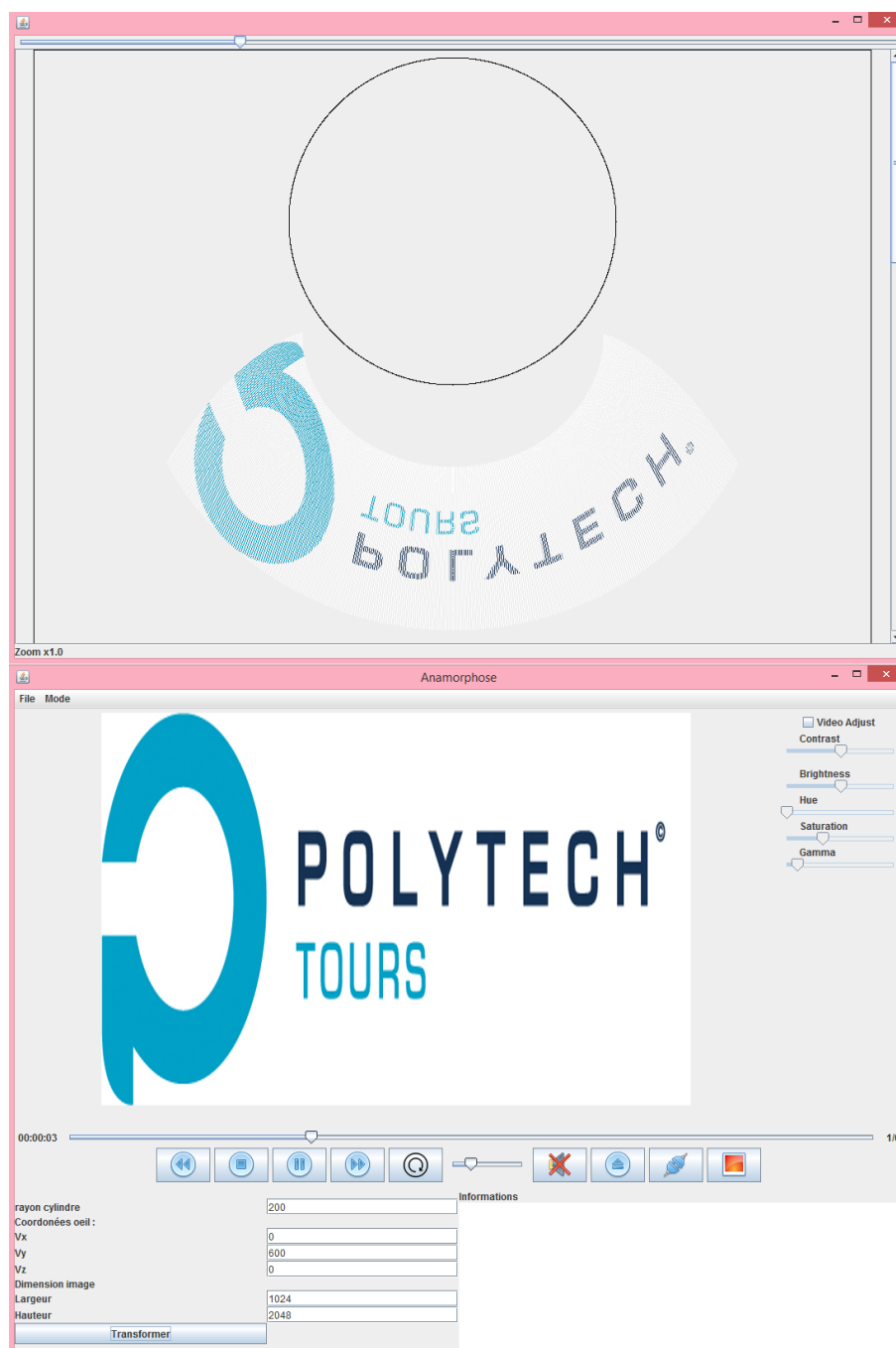


Figure 1 – Interface du projet

4 Interface logiciel/logiciel

Pour réaliser cette application, nous utiliserons deux librairies Java pour le traitement de la caméra et des fichiers Vidéo

Pour l'utilisation de la caméra dans notre application Java, la décision s'est portée sur la librairie : « webcam capture ». Ce choix a été fait car elle prend en compte une très large variété de caméra notamment grâce à l'intégration de multiple Framework de capture.

Concernant les fichiers médias, le choix s'est porté sur la librairie VLCj qui est basée sur le lecteur multimédia VLC. Le lecteur multimédia prenant en compte de multiples fichiers, que cela soit audio / image / vidéo, cela fut un critère décisif dans le choix de la librairie de vidéo.

Gestion de projet

1 Tâches

Tache 0 : Recherche sur le sujet

Priorité : haute

Entrée : aucune

Sortie : Document sur l'anamorphose cylindrique

Durée estimée : 24h

1.1 Anamorphose image

Tache 1 : Mettre en place l'environnement

Priorité : haute

Entrée : aucune

Sortie : SVN

Durée estimée : 3h

Tache 2 : Tester les formules mathématiques

Priorité : haute

Entrée : Recherche internet

Sortie : Code Java de transformation

Durée estimée : 24h

Tache 3 : Développer les classes Entité

Priorité : haute

Entrée : SVN v1

Sortie : Code Java Classe Modèle

Durée estimée : 5h

Tache 4 : Mise en place du contrôleur

Priorité : haute

Entrée : Code Java Classe Modèle

Sortie : les classes contrôleurs

Durée estimée : 16h

Tache 5 : Mise en place des vues

Priorité : haute

Entrée : les classes contrôleurs

Sortie : les vues en Java

Durée estimée : 8h

Tache 6 : Mise en place des tests unitaires

Priorité : moyen

Entrée : Cahier de Test

Sortie : Code JUnit

Durée estimée : 5h

1.2 anamorphose Vidéo**Tache 7 : Recherche de la librairie adaptée**

Priorité : haute

Entrée : Projet Anamorphose Image terminée

Sortie : Librairie vidéo intégrée au projet

Durée estimée : 32h

Tache 8 : Développer les classes entités vidéo

Priorité : haute

Entrée : Librairie Vidéo

Sortie : Code Java Classe Modèle

Durée estimée : 16h

Tache 9 : Mise en place du contrôleur pour la vidéo

Priorité : haute

Entrée : Code Java Classe Modèle de vidéo

Sortie : les classes contrôleurs

Durée estimée : 24h

Tache 10 : Adapter les vues pour la vidéo

Priorité : haute

Entrée : les classes contrôleurs de vidéo

Sortie : les vues en Java

Durée estimée : 16h

1.3 Optionnel

Tache 11 : Optimisation et multithreading

Priorité : Moyen

Entrée : Application Java Terminé

Sortie : Application de la parallélisation du code

Durée estimée : 24h

Tache 12 : Adapter l'application pour la réalisation de modèle 3D

Priorité : Faible

Entrée : Application Java Vidéo

Sortie : Application qui traite les modèles 3D

Durée estimée : Non évaluée

2 Démarche projet

Le projet se déroule suivant une méthode Agile afin de proposer plusieurs livrables et de voir l'évolution du projet au cours du temps.

Une réunion avec la MOA pourra se faire régulièrement. Cette réunion portera sur un bilan de l'avancée du projet et des éventuels changements de fonctionnalités. En plus de cette réunion, la MOE s'engage à fournir un compte-rendu hebdomadaire sur les tâches effectuées durant cette semaine.

3 Livrable

Ce projet consistera en la production de plusieurs livrables dont un minimum de deux à savoir :

- Application avec anamorphose d'image
- Application traitant l'anamorphose de flux vidéo

4 Planning Prévisionnel

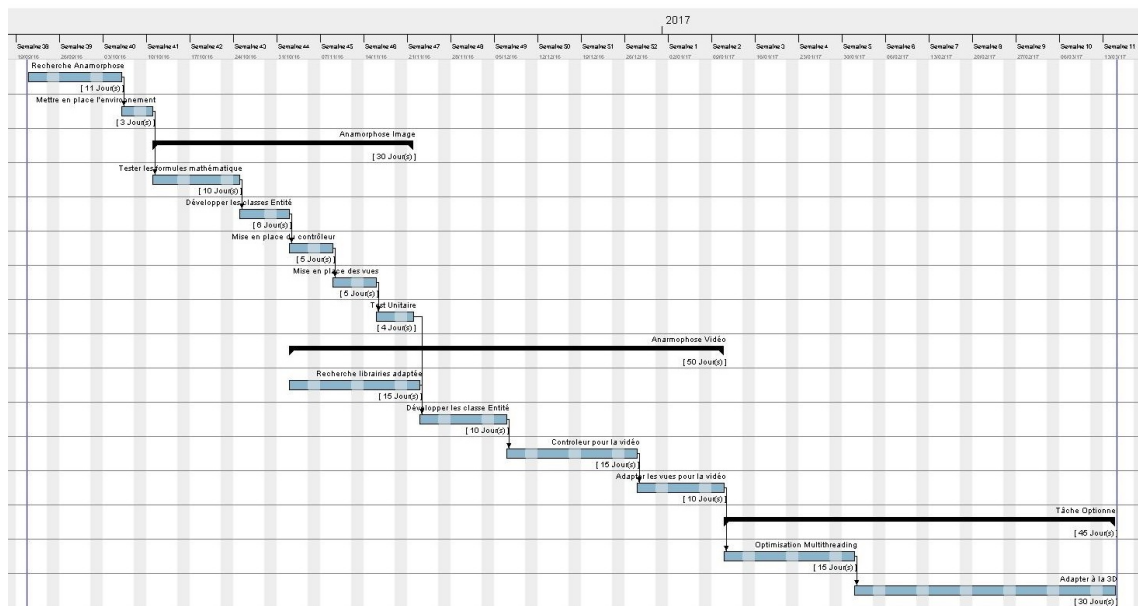


Figure 1 – Planning prévisionnel réalisé au début du projet

Bilan Gestion de Projet

1 Présentation du projet

Le projet « Anamorphose Cylindrique » est un projet de développement d'une application capable de réaliser une transformation anamorphique cylindrique sur un flux vidéo. Le principe est alors de déformer un flux d'images afin qu'il puisse être vu correctement par réflexion dans un cylindre.

L'objectif principal de ce projet fut de proposer pour les portes ouvertes de Polytech un projet mettant en avant l'utilisation de formules mathématiques dans les transformations d'images.

Le livrable imposé est une application simple permettant de réaliser une anamorphose cylindrique sur le flux d'une vidéo ou d'une caméra.

Le projet fut alors géré en mode agile avec la création de plusieurs lots faisant étapes dans la création du livrable final. Ce choix fut fait avant tout pour permettre à mon encadrant de voir l'application évoluer au cours du développement afin de rectifier certains aspects de l'interface si nécessaire.

2 Livrables et Taches

Au début du projet, plusieurs livrables intermédiaires ont été définis à savoir un prototype mettant en place la transformation sur une image puis sur une vidéo et enfin sur une caméra. A partir de ces étapes, j'ai alors découpé mon projet en plusieurs tâches.

Tout d'abord, lors de la phase de recherche, j'ai été amené à travailler sur l'élaboration de ce prototype. Il visait à tester la compatibilité des bibliothèques trouvées, avec mon utilisation de celles-ci pour la transformation d'image. J'ai ainsi pu effectuer une première phase d'analyse de la solution finale afin de prévoir les différentes tâches à réaliser pour développer la version finale de l'application.

J'ai donc commencé par créer un diagramme de classe épuré comportant seulement les classes importantes de l'application, afin que je puisse tester différentes améliorations. A partir de ce diagramme, j'ai donc conçu mon planning prévisionnel jusqu'à la date des Portes ouvertes de Polytech prévue le 4 Mars.

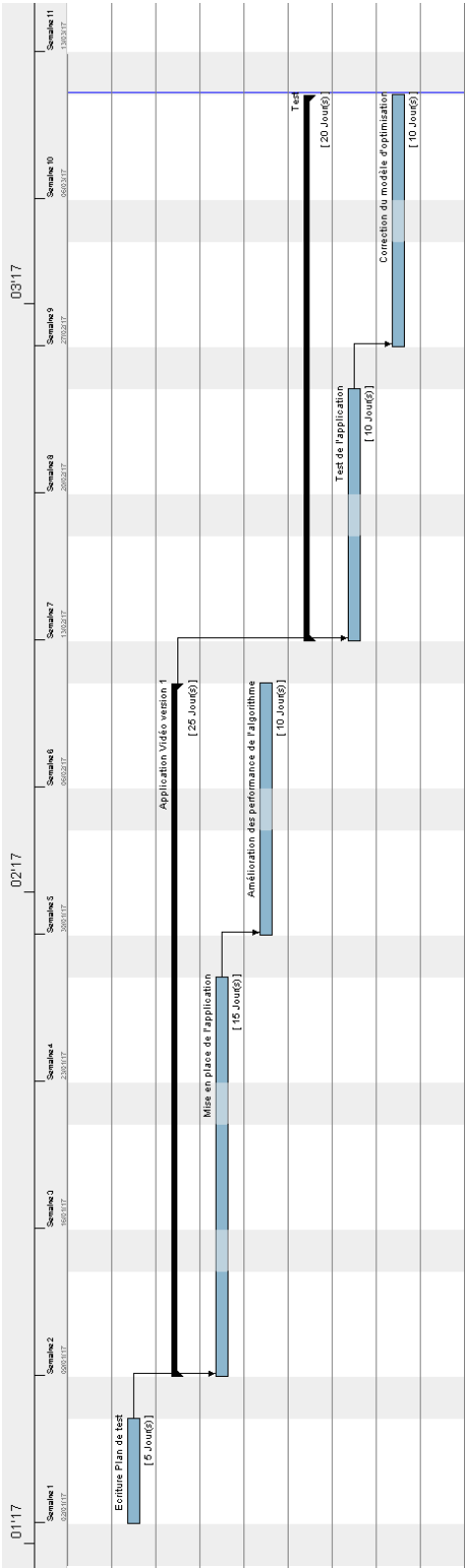


Figure 1 – Planning S10 prévisionnel

2.1 Analyse de faisabilité

Ce projet avait une contrainte temporelle forte qui était les Portes Ouvertes de Polytech. Cependant l’objectif n’était pas d’avoir entièrement terminé l’application pour cette date mais que la fonctionnalité principale de déformation fonctionne au minimum sur une image.

Cette contrainte ne fut alors pas un problème car le prototype permettait déjà ce résultat, mais uniquement sur la machine de développement. Il était nécessaire alors de la tester sur le matériel qui sera utilisé au Portes Ouvertes.

Après les premiers tests, il s’est avéré que le prototype ne convenait pas : des améliorations dans la gestion des ressources étaient nécessaires. Toutefois, cette phase fut prévue dans le planning prévisionnel et n’a donc eu aucun impact sur la mise en place de l’application sur le matériel pour les portes ouvertes.

Seule la fonctionnalité d’impression d’image déformée fut mise de côté pour cette date car le matériel n’était pas disponible pour le développement. Cette fonctionnalité fut donc reportée pour la fin du projet et a été correctement réalisée.

2.2 Analyse de risque

La tâche d'amélioration des performances fut critique pour la réalisation de ce projet avant les Portes Ouvertes, mais elle avait déjà été identifiée au premier semestre et dupliquée. L'application devant être testée après son optimisation, une seconde tâche de correction fut alors mise en place.

En cas d'échec, cette tâche aurait empêché l'utilisation de toutes les fonctionnalités le jour de la présentation au public, ce qui était déjà prévu par mon encadrant. Au final, le temps prévu pour l'optimisation du projet fut suffisant pour corriger le programme grâce à l'aide d'un professeur.

En fin de compte, même si cette tâche n'avait pas été réalisée dans le temps imparti, nous aurions pu faire fonctionner le programme avec moins de fonctionnalités et aurions reporté sa mise en œuvre sur le mois restant.

L'application devait être opérationnelle pour les portes ouvertes. Le dernier mois fut donc laissé libre pour la rédaction des rapports et la mise en place des différentes modifications, visant à corriger certaines erreurs (qui impactaient la qualité du programme).

2.3 Suivre de projet

Tout d'abord, le projet a été entièrement suivi par mon encadrant, notamment grâce aux rapports hebdomadaires qui ont été écrits. De plus, il est venu faire le point avec moi à de nombreuses reprises afin de voir l'avancement du projet mais aussi pour valider les tests fonctionnels.

Les différentes étapes clés ont donc été vérifiées par mon encadrant M. Christophe Lenté ce qui a conduit à plusieurs rectifications sur l'application via l'ajout de fonctionnalités supplémentaires.

Ces fonctionnalités ont alors ajouté un certain temps de décalage dans le planning, mais n'ont pas impacté les délais prévus. En effet, elles étaient relativement simples à mettre en place notamment grâce à l'analyse du prototype qui avait été faite.

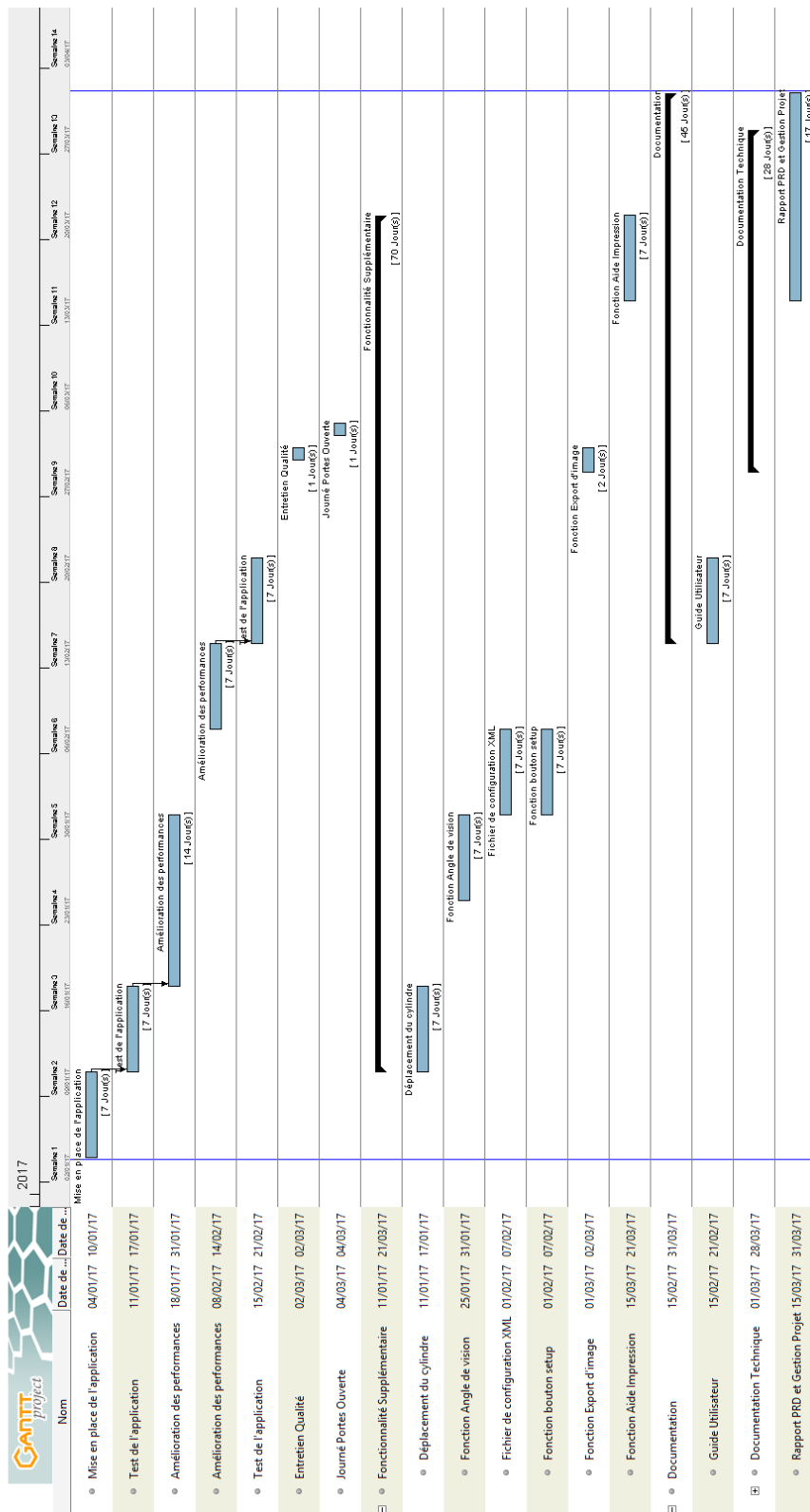


Figure 2 – Planning S10 réalisé

Voici mon planning réalisé au semestre 10. On remarque qu'il ne correspond pas tout à fait à celui prévue à la fin du semestre 9. La tâche d'écriture du plan de test a par exemple été déplacée au 1er Mars alors qu'elle était planifiée au début du semestre. A la place, j'ai réalisé plus tôt la tâche de mise en place de l'application qui fut d'ailleurs terminée plus rapidement que prévue.

Ainsi, les tâches d'amélioration des performances et de tests ont pu être avancées de deux semaines. C'est grâce à cette avance que j'ai pu ajouter certaines tâches de développement supplémentaire. La tâche d'amélioration des performances fut donc coupée par cet ajout de fonctionnalités.

L'application fut testée et validée une semaine avant les portes ouvertes de Polytech, me permettant ainsi de consacrer le reste de mon temps à la qualité logiciel et la rédaction de la documentation. Le guide utilisateur fut le premier document réalisé afin que mon encadrant puisse présenter le projet au public dans le cas où un imprévu m'aurait empêché de le faire.

Enfin, la fin du projet fut entièrement consacrée à la rédaction de documentation et la mise en place d'une fonction d'aide à l'impression.

3 Outils de gestion de projet

Afin de mettre en place mes plannings réalisés et prévisionnels, j'ai été amené à utiliser l'application GanttProject qui est gratuite et qui permet de réaliser des diagrammes de Gantt.

Dans le but de pouvoir gérer les sources de mon projet, j'ai mis en place un système de version à l'aide de SVN. Cela m'a permis d'effectuer des retours dans le temps lorsqu'une modification posait problème.

Documentation Utilisateur

1 Comment exécuter le programme

Prérequis :

- Java 1.8
- VLC (32 ou 64 bits selon Java)

Avant d'exécuter le programme, il faut vérifier tout d'abord la présence des éléments suivants :

- Un fichier "PRD.jar"
- Un fichier "Launcher.bat"
- Un dossier "config"
- Un fichier « defaultParam.xml » dans le dossier config
- Un fichier « logback.xml » peut aussi être dans le dossier config et permet l'affichage des erreurs dans des fichiers de logs plutôt que dans la console

Il vous suffit alors de double cliquer sur le fichier « Launcher.bat » afin d'exécuter le programme

2 L'interface

Au lancement de l'application vous devriez voir apparaître deux nouvelles fenêtres.

2.1 Fenêtre Source

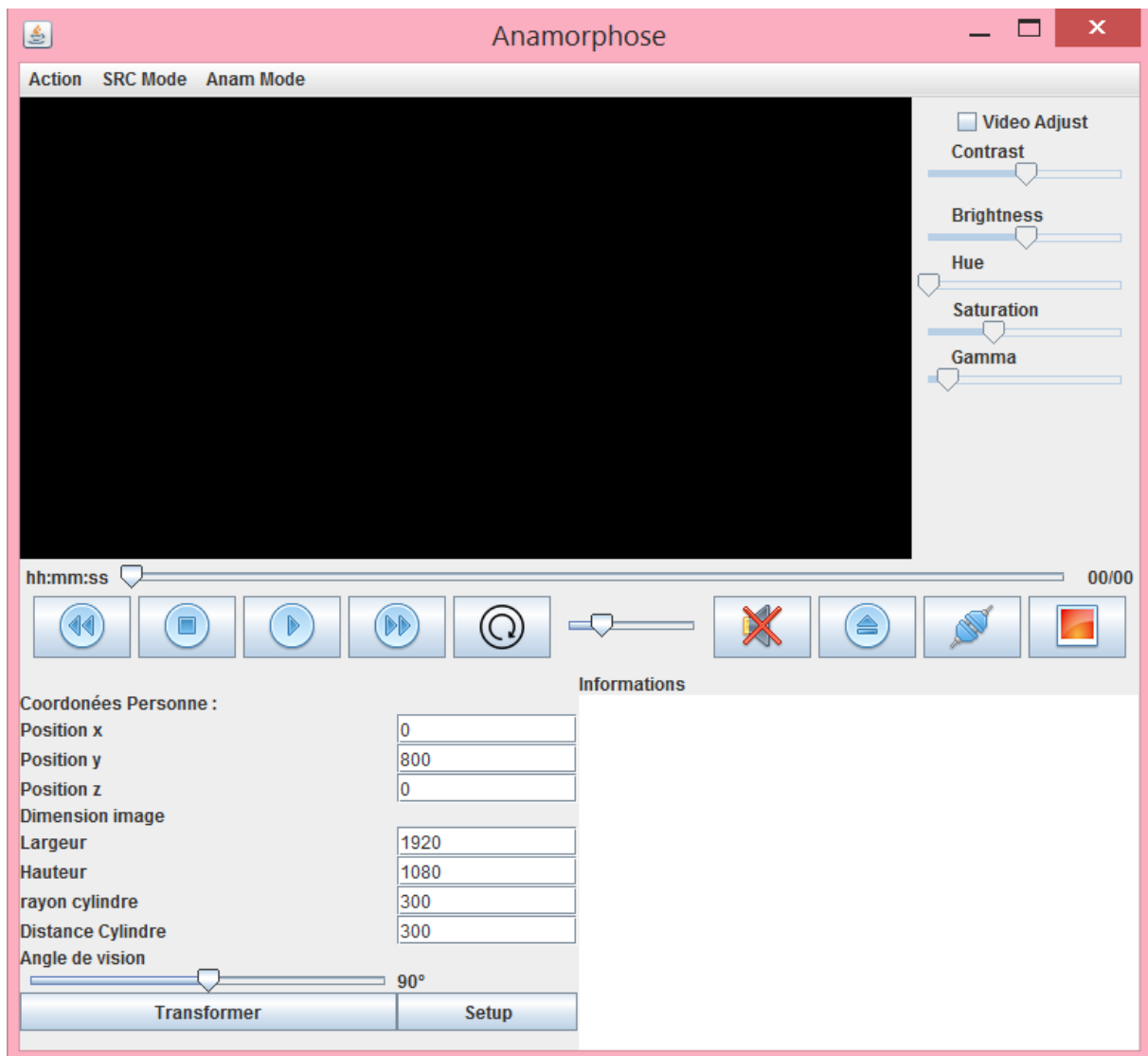


Figure 1 – Fenêtre Source

Voici la fenêtre source qui permet de réaliser la configuration de la source de l'application. Cette fenêtre comporte plusieurs parties.

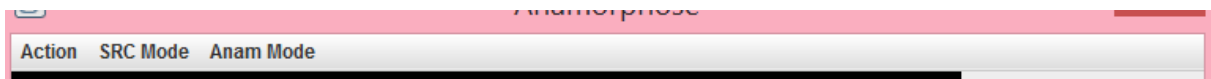


Figure 2 – Barre de Menu

Notre barre de Menu située en haut de la fenêtre comporte plusieurs menus déroulants, permettant de faire des changements dans notre application.

Le menu Action contient deux boutons qui permettent d'enregistrer l'image transformer et de quitter l'application.

Le menu SRC Mode contient deux boutons radio permettant de changer la source d'image soit par une vidéo soit par une caméra.

Le menu Anam Mode contient un bouton radio avec la seule transformation disponible : l'anamorphose cylindrique

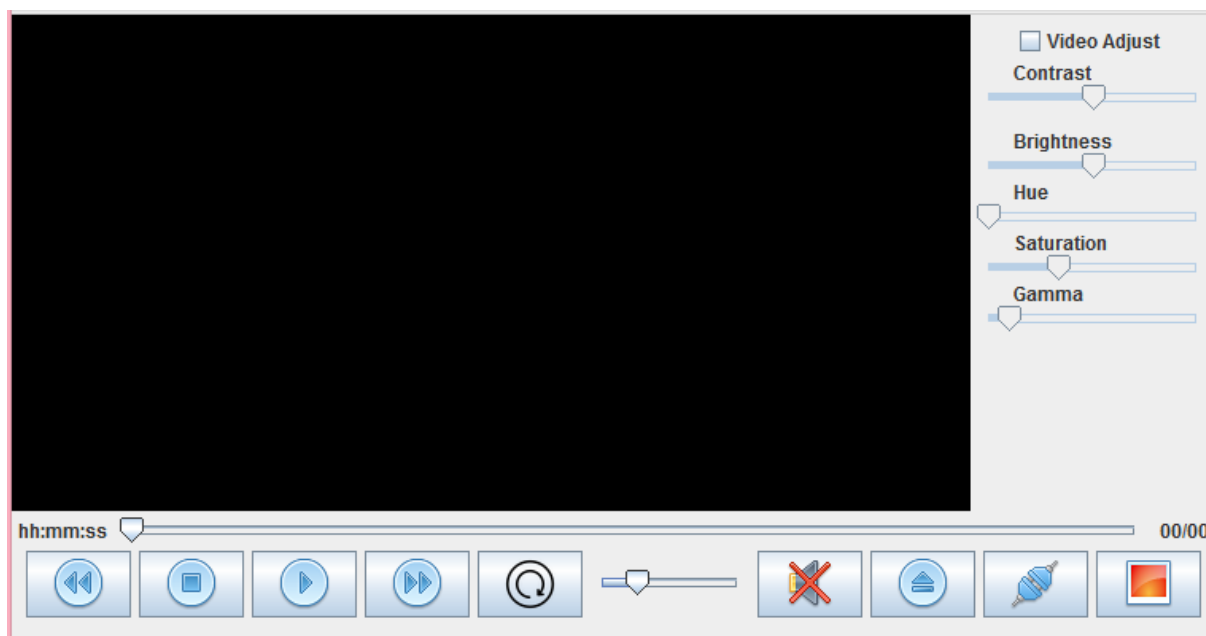


Figure 3 – Ecran Source

Notre écran source où seront affichés la caméra ou le lecteur vidéo, selon notre choix.

Le lecteur vidéo contient une barre de menu permettant diverse actions sur la vidéo.



Figure 4 – Bouton de chargement d'un fichier

Le bouton de ce panneau que l'on utilisera le plus souvent dans ce panneau est celui de chargement de fichier.

Si vous choisissez le mode avec utilisation de caméra vous devriez voir l'écran suivant :

Sur ce panneau il est possible de sélectionner la caméra que l'on souhaite utiliser comme source d'images à notre anamorphose cylindrique.

Enfin la dernière partie de cette fenêtre est le panneau de configuration et d'affichage d'informations :

Ce panneau contient plusieurs champs texte permettant de modifier les paramètres de l'anamorphose ainsi qu'un panneau d'informations contenant les messages du programme.

Vous retrouverez alors divers paramètres. Certains concernent l'application en général, comme la position du spectateur par rapport à la position du cylindre et la dimension de l'image transformée qui devra être égale à la résolution de l'écran pour un meilleur affichage.

En-dessous se trouvent les paramètres propres à la transformation par anamorphose :

- Le rayon du cylindre
- La distance cylindre qui représente la distance de la transformation par rapport au cylindre
- L'angle de vision qui représente l'angle sur lequel sera affichée l'image une fois réfléchie dans le cylindre

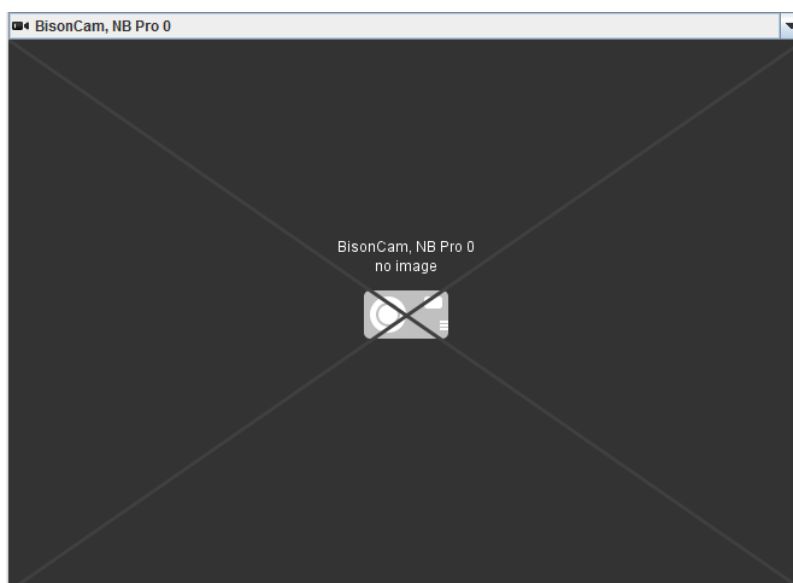


Figure 5 – Ecran de Caméra

Coordonnées Personne :		Informations
Position x	<input type="text" value="0"/>	
Position y	<input type="text" value="800"/>	
Position z	<input type="text" value="0"/>	
Dimension image		
Largeur	<input type="text" value="1920"/>	
Hauteur	<input type="text" value="1080"/>	
rayon cylindre	<input type="text" value="300"/>	
Distance Cylindre	<input type="text" value="300"/>	
Angle de vision	<input type="range" value="90"/> 90°	
<input type="button" value="Transformer"/> <input type="button" value="Setup"/>		

Figure 6 – Panneau de paramètre et d'informations

A droite se situe le panneau d'informations. Certaines seront marquées sur ce panneau, comme l'état de l'anamorphose ou encore l'affichage de la dernière erreur.

2.2 Fenêtre Destination

Sur le deuxième écran vous trouverez l'image déformée comme cela :

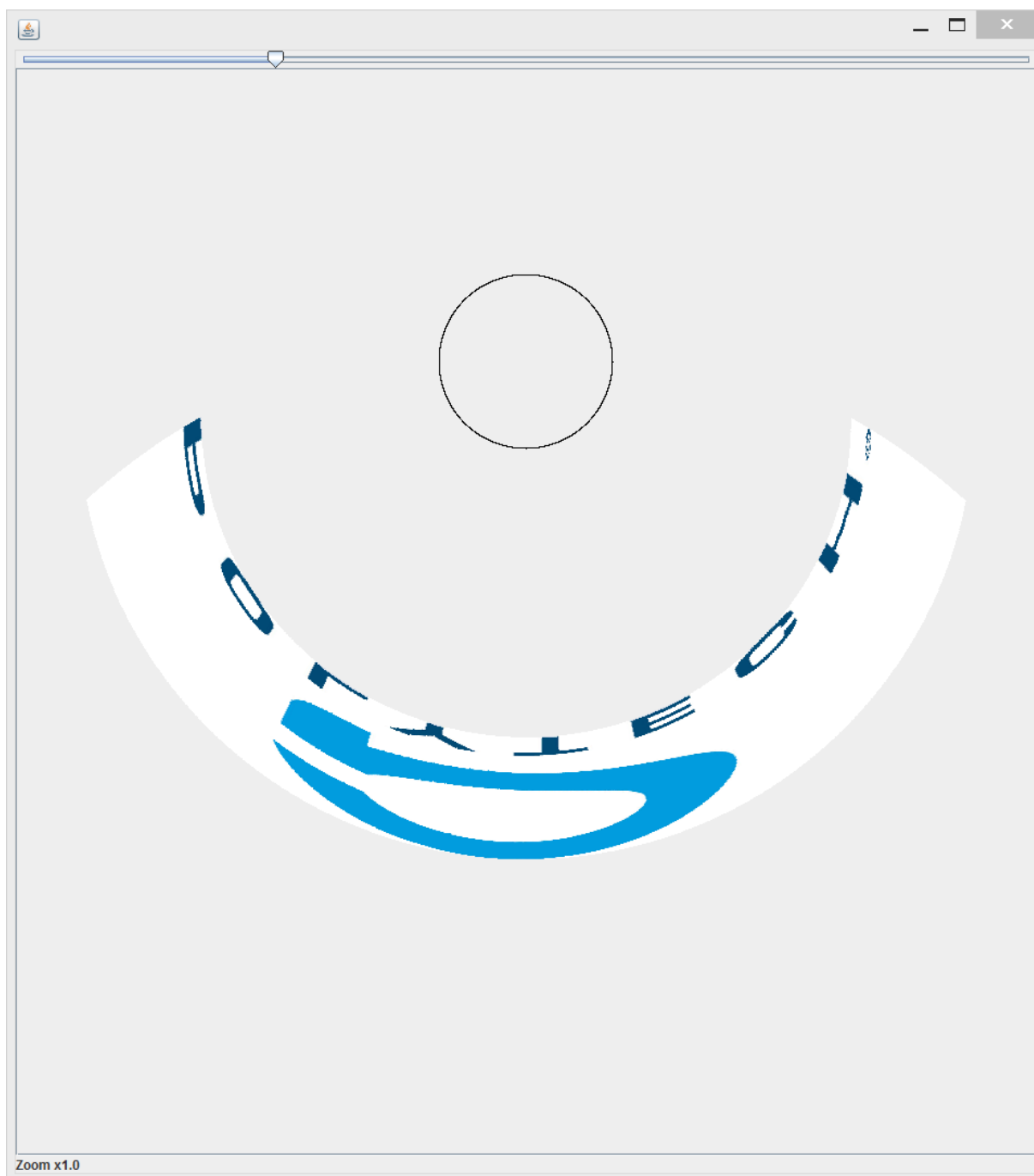


Figure 7 – *Panneau d'image transformée*

Sur cet écran on peut voir en haut un curseur permettant de zoomer / dé-zoomer. Cette fonctionnalité est aussi présente via l'utilisation de la molette de la souris. On peut alors voir la valeur du zoom affichée en bas à gauche de l'écran.

Une autre fonctionnalité présente sur cet écran est le fait de pouvoir déplacer la position du cylindre via un clic sur l'écran.

3 Actions possibles

3.1 Démarrer l'anamorphose

Avant de démarrer l'anamorphose, vous devez vérifier que tous les champs soient corrects. Si un des champs ne l'est pas, alors il sera remis à sa valeur par défaut, définie dans le fichier de configuration.

Une fois les paramètres vérifiés, il vous suffit de cliquer sur le bouton « Transformer » se trouvant en bas à gauche de la Fenêtre source. Si tout se passe correctement, l'anamorphose doit démarrer et un message apparait dans la fenêtre d'informations, avec l'état de la transformation.

3.2 Redéfinir les paramètres

A tout moment, vous pouvez demander à l'application de s'actualiser pour prendre en compte les derniers paramètres entrés. Pour cela, il vous suffit de cliquer sur le bouton « Setup » situé à droite du bouton « Transformer ».

En cliquant dessus, l'application va recharger tous les paramètres du panneau de configurations et comme pour le démarrage si une valeur n'est pas conforme il la remplacera par la valeur par défaut.

3.3 Déplacer la transformation

Quand la transformation est en cours, il est possible de déplacer la transformation sur l'écran affichant la transformation. Afin de réaliser cette action, il vous suffit de passer votre curseur sur la fenêtre qui affiche l'image transformée. Ensuite, il faut effectuer un clic gauche à l'intérieur de la zone défini par un cadre noir, qui définit en fait les bordures de l'image transformée.

3.4 Changer de source

Afin de changer la source d'images, il suffit de cliquer sur le menu déroulant « Src Mode » en haut de la fenêtre source. Vous trouverez alors deux boutons radio, un pour la source vidéo et le second pour la source caméra.

En cliquant sur l'un d'eux vous allez changer la source utilisée pour la transformation. Si vous cliquez sur l'élément qui est déjà la source d'image alors rien ne va changer.

3.5 Enregistrement d'image et aide à l'impression

L'application possède une fonction qui permet d'enregistrer l'image en cours de transformation. Cette fonction se trouve dans le menu déroulant « Action » et s'appelle « Export as PNG ». On rôle est d'enregistrer l'image en cours avec un nom généré aléatoirement afin de ne pas écraser les images précédentes.

Lors de l'appel à cette fonction, un dossier « images » se crée à côté de l'exécutable et alors l'image sera enregistrée à l'intérieur.

Cette image peut alors être imprimée, mais il se peut que le format défini dans les paramètres ne soit pas optimal pour tenir sur une feuille A4. C'est pour cela qu'une deuxième fonctionnalité a été ajoutée au programme permettant d'aider l'utilisateur à avoir les paramètres optimaux pour avoir l'image au format A4.

Cette fonction, nommée « Setting Helper to print » se trouve dans le menu Action en dessous de la fonction d'export d'image. En cliquant dessus, une fenêtre doit apparaître avec un formulaire.

 Une fenêtre de dialogue intitulée "Aide à l'impression" avec une barre de titre rose. Elle contient un formulaire avec les champs suivants : "Format Page:" avec un menu déroulant sélectionnant "Format A4 Paysage", "Taille Ecran:" (sous-titre), "Largeur Ecran (Cm) :", "Hauteur Ecran (Cm) :", "Résolution Ecran:" (sous-titre), "Largeur (px) :", "Hauteur (px) :", "Mirroir:" (case à cocher), et "Diametre Cylindre (Cm) :". Un bouton "Ajuster" est situé en bas à droite.

Figure 8 – Panneau d'aide à l'impression

Cette fenêtre contient un formulaire avec tous les paramètres nécessaires à la réalisation du paramétrage pour un format de page donné. Pour notre application, seuls les formats A4 en paysage et en portrait nous intéressent.

Le formulaire contient plusieurs parties. La première est le choix du format de la page parmi les formats disponibles. Ensuite, il faut remplir certaines informations sur l'écran à savoir sa largeur et sa hauteur réelles en cm puis la résolution courante de l'écran en pixel. Enfin, il vous est demandé le diamètre du cylindre en Cm.

Une fois le formulaire rempli, il vous suffit de cliquer sur le bouton « Ajuster » afin que les nouveaux paramètres se mettent automatiquement dans le panneau de configuration. Par contre si une erreur subsiste dans le formulaire, la fenêtre d'erreur suivante apparaît.

 Une fenêtre de dialogue intitulée "Form Error" avec une barre de titre rose. Elle contient un message d'erreur "Le formulaire est mal remplie" précédé d'un pictogramme d'erreur (un X rouge dans un octagone). Un bouton "OK" est situé en bas au centre.

Figure 9 – Fenetre d'erreur

L'erreur est alors affichée sur le panneau d'informations. Il est possible qu'il y ait plusieurs erreurs dans le formulaire mais le panneau ne recense que la dernière erreur qui est apparue.

Documentation Développeur

1 Mise en place de l'environnement de développement

1.1 Prérequis

Afin de pouvoir développer sur l'application « Anamorphose Cylindrique » vous avez besoin des éléments suivants :

- Un IDE (Le projet fut développé sous Eclipse Luna et Néon)
- Java 1.8
- JUnit 4 (Prévue de base avec Eclipse)
- VLC 32 ou 64 bits selon la version Java (La version actuelle qui fonctionne est 2.2.4 Weatherwax)

1.2 Installation

Afin pouvoir travailler sur l'application, il vous faut tout d'abord vérifier la présence des éléments suivants :

- dossier libs avec l'architecture suivante :

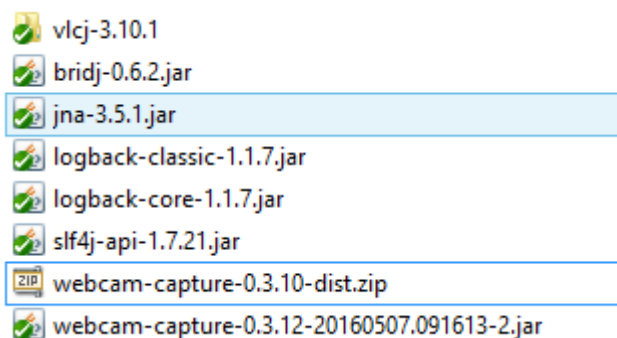


Figure 1 – Dossier libs

Le fichier « webcam-capture-0.3.10-dist.zip » contient les fichiers sources du jar situé en dessous.

Le dossier vlcj contient quant à lui les éléments suivant :

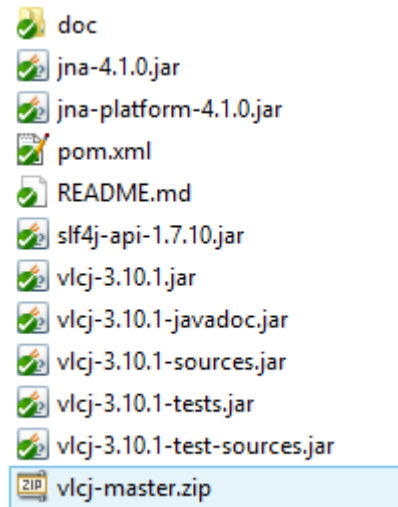


Figure 2 – Dossier vlcj

Le fichier « vlcj-master.zip » contient les fichiers sources des différents fichiers « .jar » situés au-dessus.

- dossier « config » contenant les fichiers suivants :

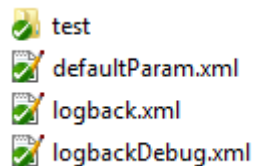


Figure 3 – Dossier config

Le dossier "config" contient un fichier de configuration pour l'application et aussi des fichiers de gestion des Logs, dont un pour le programme en usage normal et un pour l'usage en mode debug. Enfin il comporte un sous-dossier avec les configurations nécessaire aux tests unitaires.

Afin de déployer l'application dans votre environnement de développement, il vous suffit de faire un import du fichier Zip contenant toutes les sources. Une fois cela fait, il faut vérifier que toutes les librairies du dossier libs sont présentes.

2 Implémentation du diagramme de classe

2.1 Diagramme complet

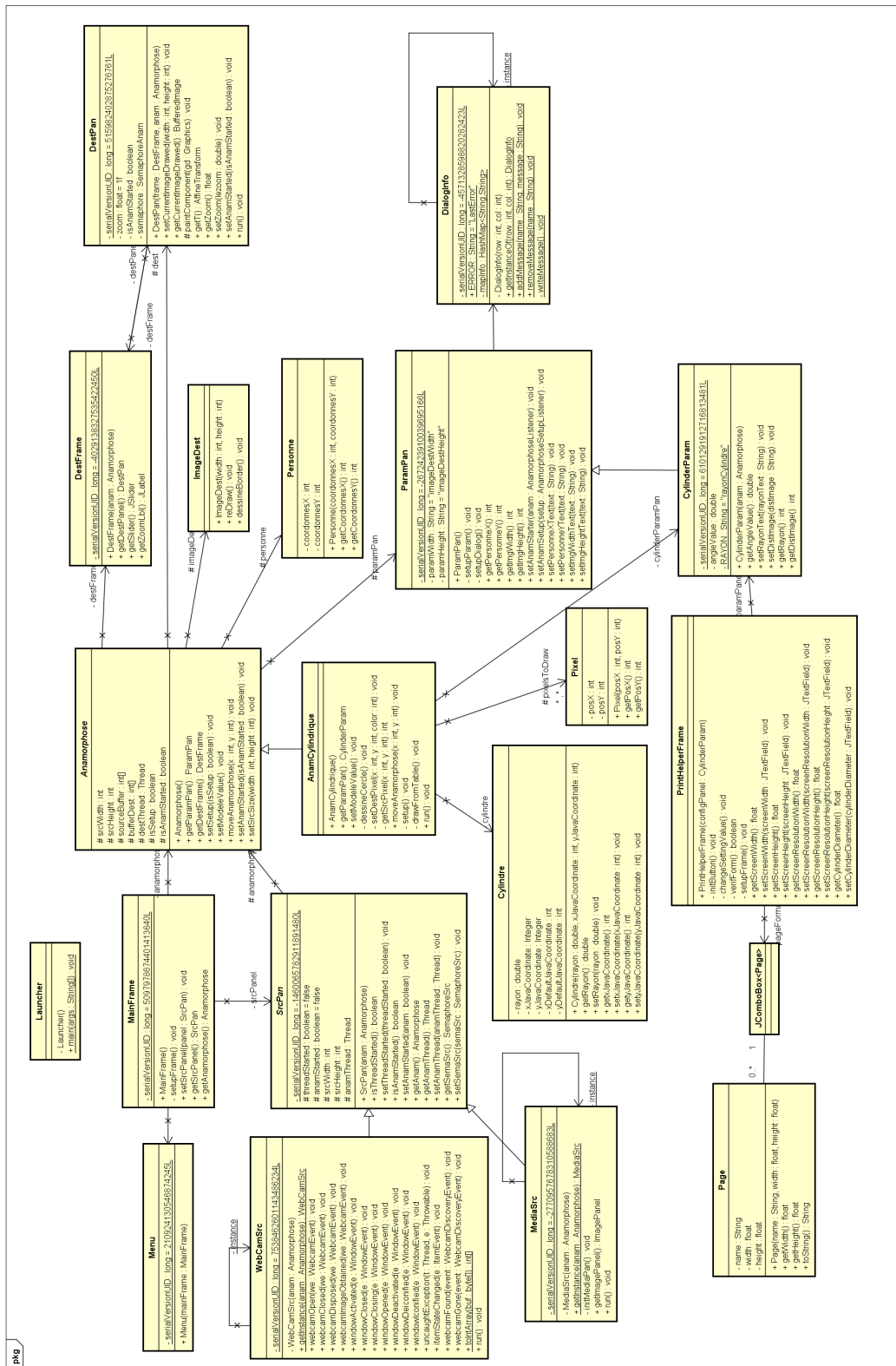


Figure 4 – Diagramme de classe du projet

2.2 Choix d'implémentation

Lors de la mise en place de l'application, j'ai mis en place différents patterns afin que l'application puisse être réutilisable et améliorable.

2.2.1 Implémentation général

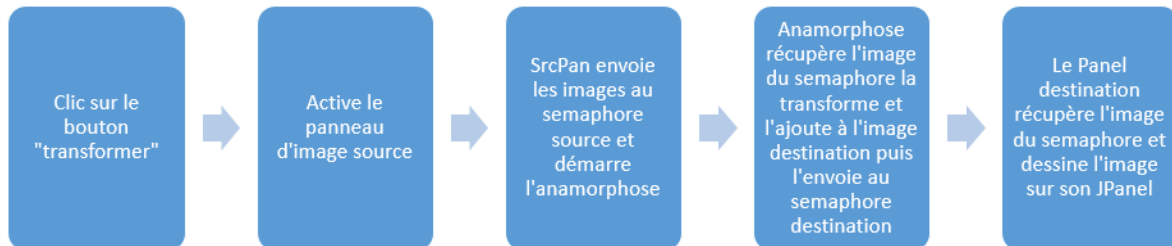


Figure 5 – Schéma général de l'application

Ce diagramme schématise le déroulement des actions qui s'enchainent à partir du clic sur le bouton "transformer".

2.2.2 Le panneau d'image source

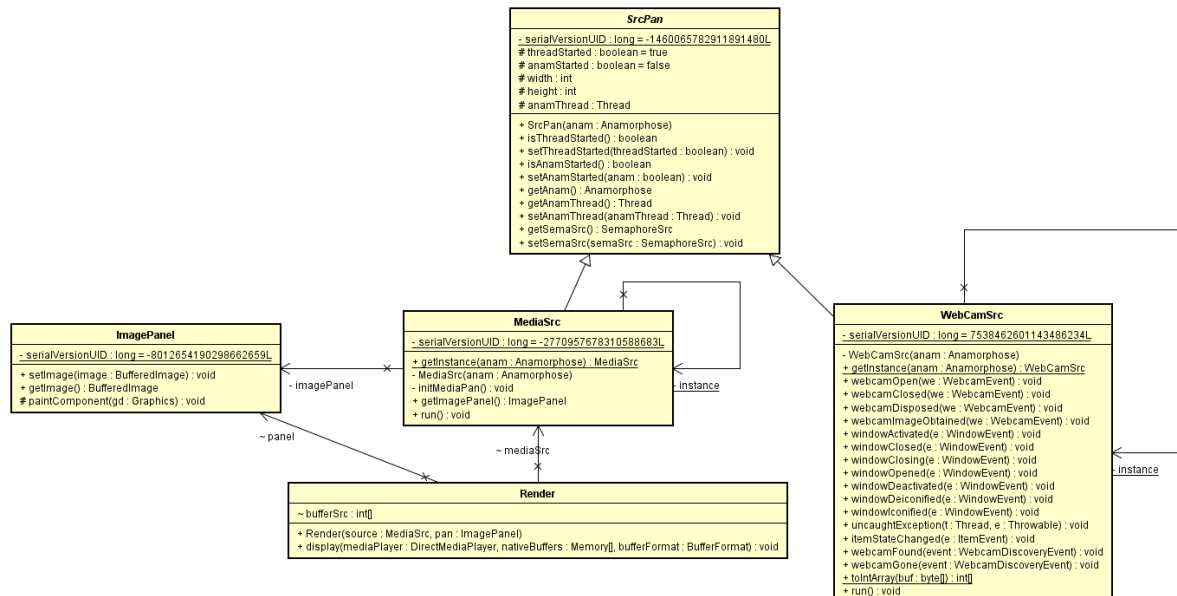


Figure 6 – Diagramme de Classes du panneau d'image source

Le panneau d'image source représenté par la Classe abstraite « SrcPan » est une classe qui hérite de JPanel. Ce choix a été fait car notre fenêtre principale est une JFrame.

Cette classe implémente l'interface « Runnable » car c'est grâce à la fonction « Run » que nous récupérons les images. Toutefois, il est possible que certaines bibliothèques possèdent leur propre classe pour récupérer les images, comme c'est le cas pour la bibliothèque VLC qui les récupère grâce à la classe « Render ».

La classe SrcPan comporte de nombreux éléments tels que :

- Deux booléens « anamStarted » et « threadStarted » qui notifient si la transformation est lancée et si le panneau doit être actif.
- l'Anamorphose en cours car c'est le panneau d'images source qui va lancer le processus de transformation grâce au Thread « anamThread » qu'il stock aussi. Ce thread représente l'instance d'Anamorphose en cours lancé par le JPanel.
- Les dimensions de l'image source qui ont normalement été définies dans le fichier de configuration.
- Un Semaphore contenant la copie de la dernière image source qui sera utilisée par la transformation d'image.

Pour ce projet, nous avons donc deux classes qui héritent de « SrcPan » ; A savoir « MediaSrc » pour la bibliothèque VLC et « WebCamSrc » pour la bibliothèque de la caméra. Ces deux classes implémentent la stratégie Singleton afin de ne garder qu'une seule instance de la fenêtre et donc éviter que des Thread soient créés inutilement.

Le Panneau source de VLC est décomposé en plusieurs classes par rapport à celui de la Webcam. Le panel comporte plusieurs menus de VLC qui ont été créés dans des classes séparées. De plus, les fonctions de récupération (Classe « Render.java ») et de définition du format de l'image (Classe « BufferFormatCallBackImpl ») sont des classes à implémenter afin de pouvoir récupérer les images de la vidéo.

2.2.3 La classe Anamorphose

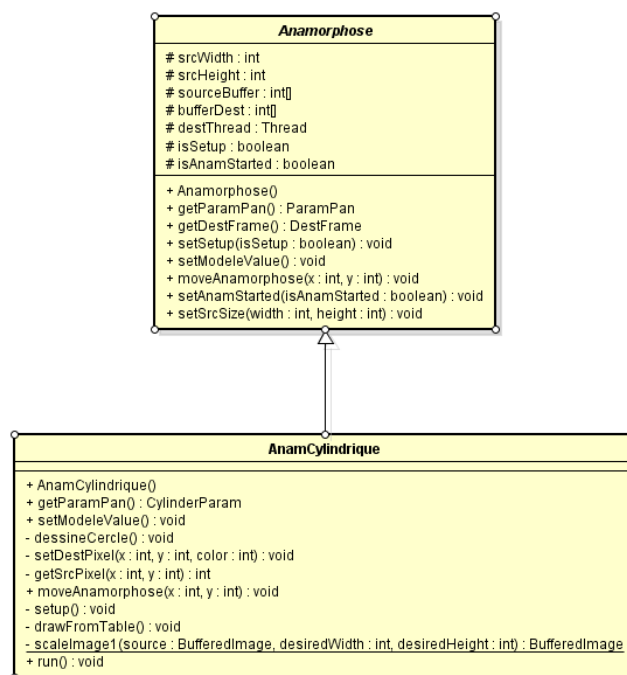


Figure 7 – Diagramme de classes de l'anamorphose

Le but de ce projet était de pouvoir avoir une application permettant de réaliser une anamorphose Cylindrique. Néanmoins, dans une optique d'amélioration ou d'ajout de fonctionnalités au programme tel que d'autres types d'anamorphose, j'ai mis en place une classe abstraite « Anamorphose » et donc une classe « AnamCylindrique » qui en hérite.

La classe Anamorphose implémente la classe Runnable car on va exécuter la fonction de transformation en parallèle à la lecture d'image source. Cette classe contient deux booléens, « isSetup » qui va définir l'état de la transformation comme étant prête et « isAnamStarted » qui va définir si l'utilisateur a mis en route la transformation d'images.

La classe contient aussi divers attributs relatifs aux images telles que les dimensions des images sources et un buffer d'entiers destiné à contenir les informations de la dernière image du Sémaphore source. Nous avons besoin de ces deux informations car elles nous permettent de mieux parcourir le buffer quand il faut effectuer la transformation. La classe contient aussi un buffer pour l'image destination ainsi qu'un Thread correspondant à l'instance du panneau de destination.

La classe AnamCylindrique hérite donc de Anamorphose afin d'implémenter la fonction run. Pour la transformation par anamorphose cylindrique j'ai séparé la fonction en deux parties.

La première partie est la partie Setup. On va faire en sorte que tous les éléments dont on a besoin pour la transformation soient correctement initialisés. Pour cela, on appelle la fonction setModelValue qui met à jour toute les instances des classes modèles pour l'anamorphose cylindrique et appelle la fonction de la classe mère qui est censée mettre à jour celles des anamorphoses en général.

Ensuite, on va appliquer notre fonction de transformation sur tous les pixels de notre image source. Cette transformation va alors me rendre un tableau avec la nouvelle position pour chaque pixel de l'image. Une fois ce tableau enregistré, on met à jour la variable isSetup afin que l'algorithme ne retourne plus dans le setup que si on le lui redemande.

C'est alors que la seconde partie prend le relais avec une méthode bien plus rapide. Tant qu'il n'y a pas eu de nouveau setup, le format de l'image n'est pas censé bouger. Ainsi, on peut réutiliser le tableau de coordonnées pour afficher les nouveaux pixels. Le gain de temps est donc considérable car tous les calculs de transformation sont exécutés une seule fois puis on ne parcourt plus qu'un tableau pour récupérer la valeur de couleur des nouveaux pixels.

2.2.4 Le panneau d'image transformé

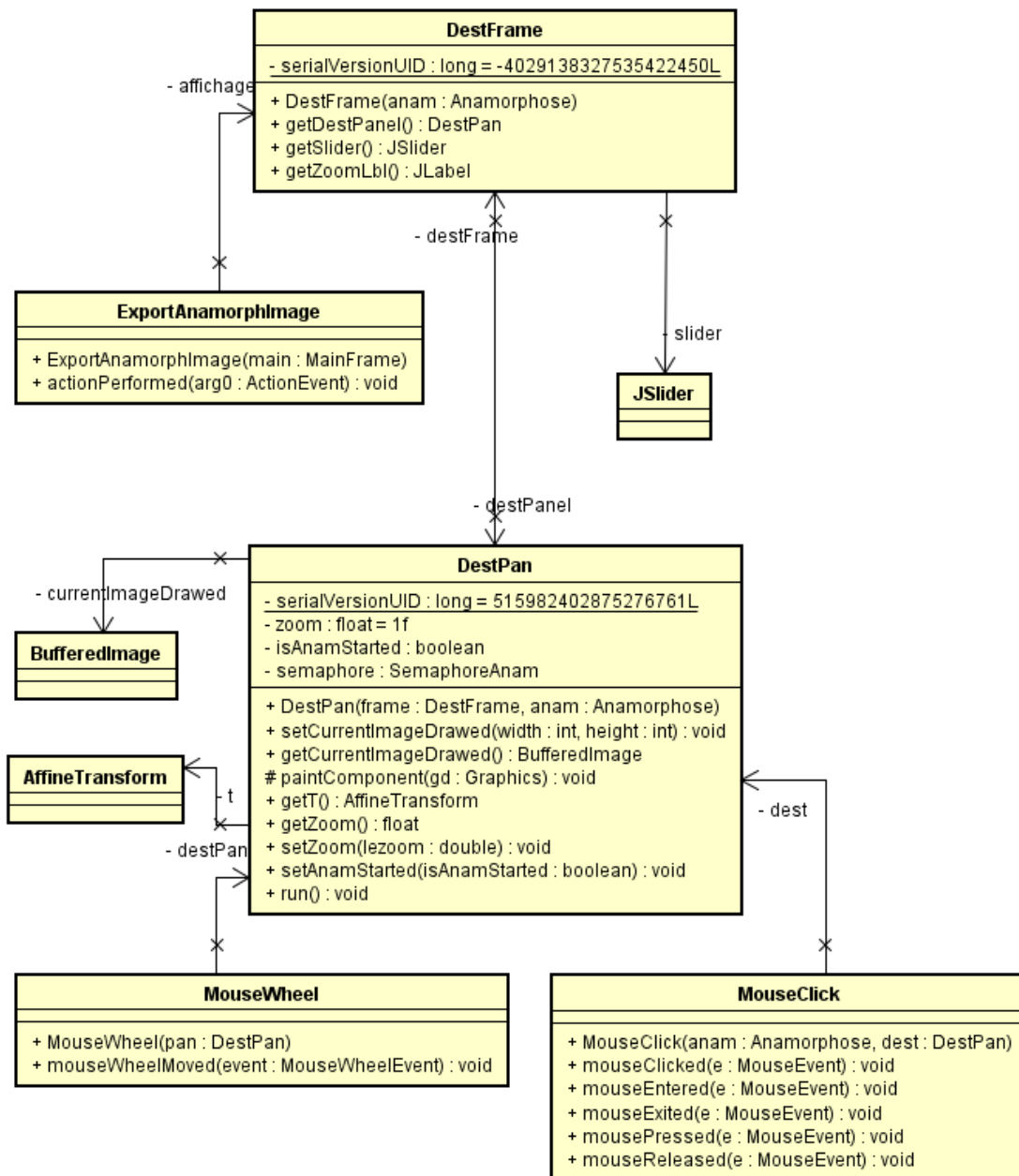


Figure 8 – Diagramme de Classes du panneau destination

Le panneau d’affichage d’image transformé est la seule classe de la chaîne de traitement ou je n’ai pas implémenté d’héritage. Tout d’abord, le projet est centré exclusivement sur l’affichage de vidéo ou de photo. J’ai donc mis en place plusieurs fonctionnalités permettant l’affichage de vidéo dans les meilleures conditions possibles.

Pour cela, j’ai séparé en deux parties cette fenêtre avec d’une part la fenêtre contenant un slider et d’autre part un JLabel permettant de modifier le zoom et d’en afficher la valeur. Cette fenêtre est en lien avec la fonction d’export qui va demander l’image en cours d’affichage

contenu dans son JPanel.

Ce JPanel représente la deuxième partie de la fenêtre qui garde en mémoire la dernière image transformée et qui est lui-même inclus dans un JScrollPane permettant un effet de zoom tout en conservant la même taille de fenêtre.

C'est ce JPanel qui implémente la classe Runnable et qui sera exécuté à partir de la classe Anamorphose. Cette fonction run est simple. C'est une boucle qui va utiliser le sémaphore contenant la dernière image transformée à chaque fois qu'une nouvelle image arrivera, et l'affichera dans son JPanel en appliquant une transformation affine dans le but de garder l'image et son zoom au centre de la fenêtre.

Ce panel comporte deux actions implémenté par deux Listener « MouseWheel » et « MouseClic ». MouseWheel permet à l'utilisateur de zoomer grâce à l'utilisation de la molette de sa souris. L'action de zoom étant aussi active sur le Slider de la JFrame. Et l'action MouseClic elle va permettre de déplacer le cylindre en prenant en compte les coordonnées X et Y du clic ainsi que la transformation affine pour savoir où se situe l'image.

2.2.5 Le panneau de configuration

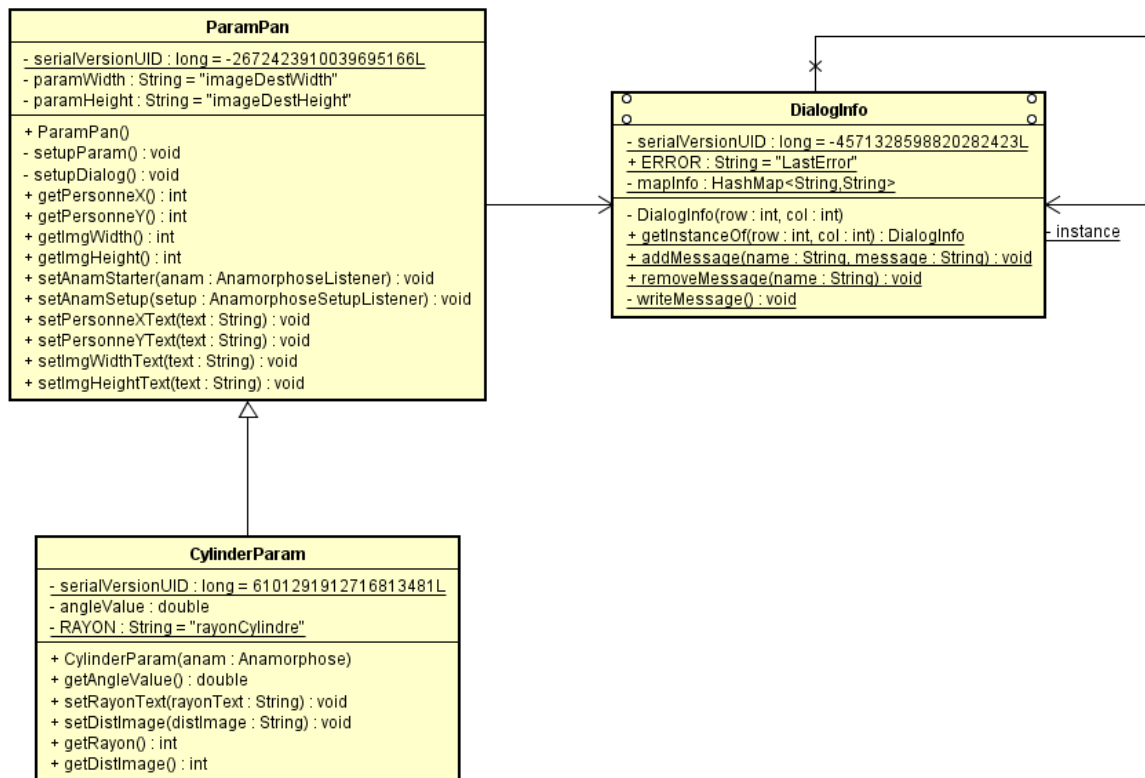


Figure 9 – Diagramme de Classes du panneau de configuration

Le panneau de configuration définit tous les paramètres nécessaires à la transformation anamorphique. C'est pourquoi elle est fournie par la classe Anamorphose car chaque Anamorphose doit posséder son propre panneau de configuration pour récupérer les paramètres nécessaires.

Panneau de configuration

Le panneau de configurations est une classe abstraite qui hérite de la classe JPanel. Il définit tous les champs nécessaires à toutes les anamorphoses qui sont alors disposés dans un GridBagLayout. J'ai utilisé ce layout car il permet de gérer la position X et Y sur la grille de chaque champs peu importe l'ordre d'ajout.

Le panneau contient deux boutons permettant de lancer l'anamorphose et de refaire le setup de l'anamorphose, mais ils ne sont pas ajoutés à la grille. Chaque classe fille doit ajouter les boutons qu'elle souhaite à la fin du GridBagLayout. Cela permet ainsi à certaines anamorphoses de ne pas forcément avoir un bouton pour le setup si elle ne le permet pas.

Panneau d'informations

Le panneau d'informations « DialogInfo.java » est contenu dans le panneau de configuration mais la classe implémente le pattern Singleton car il doit être intégré dans chaque panneau de configuration.

Afin de gérer l'affichage des messages de n'importe quelle partie du programme, la classe « DialogInfo » comporte une « Hashmap » accessible de façon statique par les méthodes d'ajout ou de suppression d'entrée dans la map. A chaque modification, on appelle la méthode « writeMessage() » qui va actualiser le panneau avec les informations que la map possède.

2.3 Possibilité d'amélioration

Ce projet a été mis en place avant tout pour être utilisé avec l'anamorphose cylindrique mais a été développé de sorte qu'il puisse être amélioré par l'ajout de nouveaux modes d'entrée ou de nouvelles transformations. Ceci est une liste d'améliorations qui pourraient y être apportées.

2.3.1 Utilisation de Raster

Lors de la présentation du projet aux Portes Ouverte, le professeur M. Aupetit m'as suggéré de mettre en place un système utilisant la "rastérisation" d'image afin de pouvoir combler les trous correspondant aux pixels manquant. Je n'ai pas eu le temps de regarder son principe de fonctionnement mais certains éléments sont présents dans ces anciens cours au lien suivant dans l'option réalité virtuelle.

<https://wiki.projectsforge.org/xwiki/wiki/teaching/view/DI5/>

2.3.2 Un panneau d'aide à l'impression amélioré

Au début du projet, il m'a été demandé de mettre en place une solution permettant à l'utilisateur de pouvoir imprimer facilement la transformation au format A4. N'ayant jamais eu accès à une imprimante pour faire des tests, j'ai donc ajouté une fenêtre d'aide à l'impression qui reprend le principe de mise à l'échelle de l'image destination avec comme référence la taille de l'écran et sa résolution.

D'une part, ce panneau n'implémente pas d'héritage et donc ne fonctionne que pour une anamorphose cylindrique. Il faudrait donc mettre en place une nouvelle classe générale à toutes les anamorphoses. Cependant, la meilleure solution serait de réaliser une vraie interface permettant l'impression à partir de l'application.

2.3.3 Gérer le cas d'un changement de transformation

L'application permet à l'heure actuelle de réaliser seulement la transformation par anamorphose cylindrique mais ne convient pas pour une utilisation avec plusieurs transformations. Afin de mettre en place d'autres transformations voici les points suivants à mettre en place :

- Si l'anamorphose change, Faire que en sorte que les panneaux d'images sources et destination soit aussi affectés par ce changement. Il faut donc aussi prendre en compte tous les panneaux sources.
- Implémenter le Pattern Singleton pour l'anamorphose afin de ne pas initialiser plusieurs threads exécutant la même transformation.
- Gérer l'activité des Threads de transformation comme ce fut le cas pour les Threads d'images source avec un booléen définissant si l'anamorphose est active ou non.

3 Qualité Logiciel

3.1 Fichiers de Log

Afin de respecter les métriques de qualités, j'ai remplacé les affichages d'erreur dans la console par un affichage via Logger.

Le logger SLF4J est simple à mettre en place car il suffit d'utiliser la fonction suivante :

- `LoggerFactory.getLogger(NomClass.class)`

Cette méthode vous donne accès au logger répertorié pour cette classe. Ainsi, tous les messages d'information, de debug et d'erreur doivent être écrits avec l'utilisation de ce logger. Au début du programme dans la classe « `Launcher.java` », je définis une variable système afin que la classe `Logger` puisse être paramétré par un fichier de configuration.

Le fichier de configuration « `logback.xml` » permet entre autre de définir quel niveau de message doit être affiché dans la console mais aussi facilite l'enregistrement de ces messages dans des fichiers. Pour ma part, j'ai défini plusieurs niveaux d'affichage :

- La console n'affiche que les erreurs de type `WARN` ou supérieur
- Le fichier `log/Error.log` enregistre les erreurs de type `WARN` ou supérieur
- Le fichier `log/Debug.log` n'enregistre que les affichages de type `DEBUG`
- Le fichier `log/Info.log` n'enregistre que les affichages du type `INFO`

Un second fichier de log est utilisé pour l'affichage en mode debug et permet d'afficher toute erreurs dans la console.

3.2 Tests Unitaires

Pour réaliser mes tests unitaires, j'ai utilisé JUnit 4 qui met en place les annotations de test. Certains de mes tests consistent à vérifier qu'une fonction renvoie bien une exception. Pour cela, j'utilise la classe « `ExpectedException` ». La fonction « `expect` » permet de préciser quelle exception le programme doit envoyer validant ainsi le test si l'erreur est bien récupérée.

Afin de ne pas polluer les fichiers de logs lors de la réalisation des tests, j'ai mis en place un fichier « `logbackTest.xml` ». Il permet de récupérer toutes les erreurs qui ont été enregistrées par le `Logger` et les enregistre dans un dossier `logTest`.

3.3 Métrique Sonar

Afin de vérifier la qualité de mon application, j'ai utilisé SonarQube. Ce logiciel met en place une grande liste de métriques permettant de vérifier la qualité du code. Il m'a alors permis de corriger certaines erreurs dans mon code mais aussi de vérifier que mon code respectait les conventions Java.

Pour mettre en place Sonar dans votre environnement de travail il vous faut télécharger deux applications :

- Le serveur SonarQube (<https://www.sonarqube.org/downloads/>)
- Le Scanner Sonar Runner (<https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>)

Il vous suffit alors de télécharger ces deux utilitaires et de suivre la documentation Sonar Scanner afin de pouvoir scanner le projet Java. Les résultats seront alors directement enregistrés sur le serveur.

3.4 Métrique JMC

Java Mission Control est un utilitaire permettant de scanner l'activité de votre programme quand il est en cours d'exécution. Pour cela, il faut vous procurer l'application sur le site d'Oracle à l'adresse suivante : <http://www.oracle.com/technetwork/java/javaseproducts/mission-control/java-mission-control-1998576.html>

Après installation de l'application sur votre ordinateur, il vous suffit de lancer le programme Java Mission Contrôle et de lancer l'application d'anamorphose. Une ligne devrait apparaître dans le JVM Browser mentionnant la classe Launcher de notre application. Il vous suffit alors d'effectuer un clic droit sur cette ligne et de sélectionner « Start Flight Recording ». La fenêtre suivante doit apparaître vous demandant d'activer la fonctionnalité.

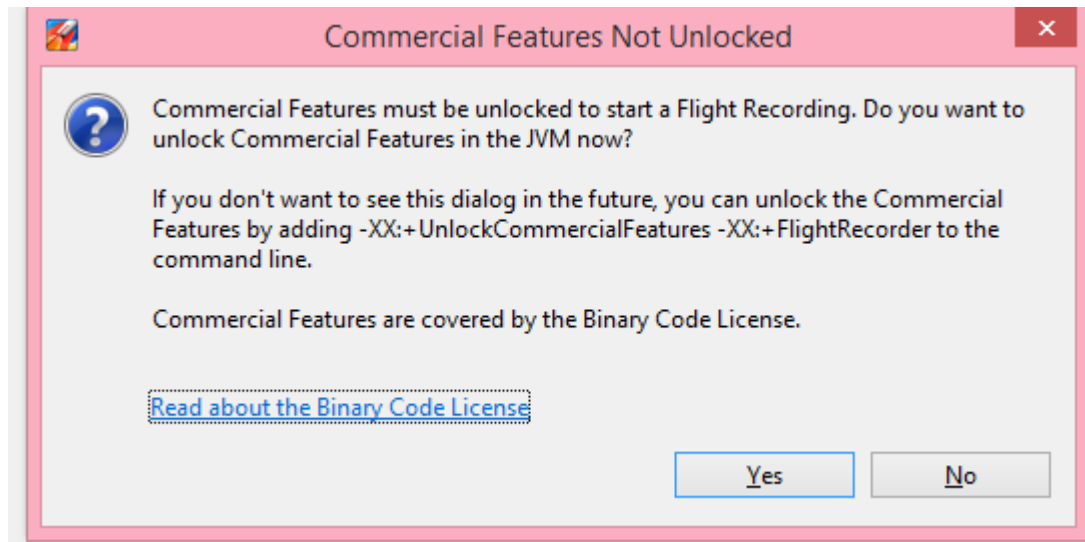


Figure 10 – Fenêtre Java Mission Control

Cliquez sur « Yes » et un formulaire vous demandera des informations quant à la durée de l'enregistrement à effectuer.

Une fois l'enregistrement terminé, le rapport doit alors apparaître donnant ainsi les informations sur les éléments qui ont été alloués, la mémoire utilisée et les fonctions les plus souvent appelées.

4 Versionning

Lors de ce projet, j'ai utilisé un repository SVN pour la gestion de mes fichiers sources. Ce repository est disposé sur un serveur publique « XP-Dev ».

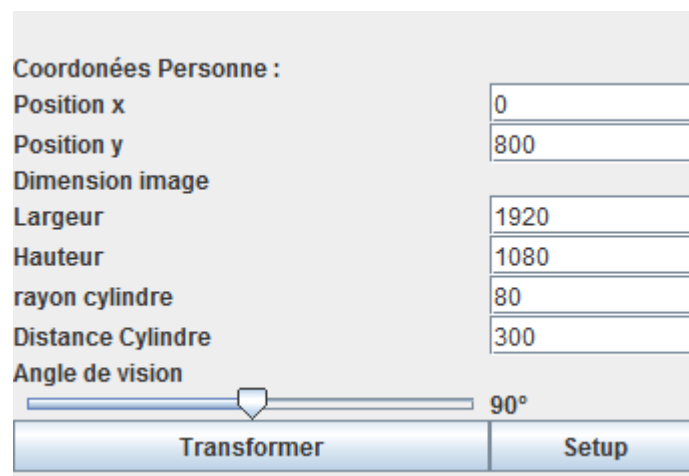
<https://xp-dev.com/summary/238503>

Si vous voulez les accès à ce SVN, veuillez me contacter à cette adresse alexis.j.saludo@hotmail.fr

Cahier de tests

1 Panneau de configuration

L'application comporte un panneau de configuration pour la transformation comportant diverses champs texte.



The configuration panel is titled 'Coordonnées Personne :'. It contains several input fields and a slider. The fields are: 'Position x' with value 0, 'Position y' with value 800, 'Dimension image' with value 1920, 'Largeur' with value 1080, 'Hauteur' with value 80, 'rayon cylindre' with value 300, and 'Distance Cylindre' with value 90°. Below these fields is a slider for 'Angle de vision' ranging from 0 to 90°. At the bottom are two buttons: 'Transformer' and 'Setup'.

Paramètre	Valeur
Position x	0
Position y	800
Dimension image	1920
Largeur	1080
Hauteur	80
rayon cylindre	300
Distance Cylindre	90°

Figure 1 – Panneau de configuration

1.1 Conditions de test

1.1.1 Paramètres général aux anamorphoses

Test du champ Position X de l'Observateur

Condition :

- Ce champ doit être au format numérique

Test du champ Position Y de l'Observateur

Condition :

- Ce champ doit être au format numérique

Test du champ Largeur de la Dimension de l'image destination

Condition :

- Ce champ doit être un nombre entier positif

Test du champ Hauteur de la Dimension de l'image destination

Condition :

- Ce champ doit être un nombre entier positif

Condition supplémentaire :

- La multiplication de la largeur par la hauteur de l'image destination ne doit pas dépasser 16 000 000 afin de ne pas saturer l'espace mémoire.

1.1.2 Paramètres liés à l'anamorphose cylindrique

Test du champ Rayon du cylindre

Condition :

- Ce champ doit être un nombre entier positif

Test du champ Distance Cylindre

Condition :

- Ce champ doit être un nombre entier positif

1.2 Cas de tests

1.2.1 Liste des cas

Voici une liste de cas à tester afin de vérifier que le formulaire fonctionne correctement :

- Le formulaire est bon toutes les valeurs sont correctes
- Le formulaire contient un champ incorrectement rempli
- Le formulaire contient un champ vide
- Le formulaire contient une valeur négative qui ne devrait pas l'être
- Le formulaire contient des valeurs trop grandes pour la hauteur et largeur destination

1.2.2 Résultat

Formulaire correct

Quand le formulaire est correct alors le programme passe à la suite de l'exécution

Formulaire incorrect – champ mal rempli

Quand un champ du formulaire est incorrect, le formulaire renvoie une erreur « InvalidFormatException ». La valeur est alors remplacée par celle par défaut du fichier de configuration.

Formulaire incorrect – champ vide

Quand un champ du formulaire est vide, le formulaire renvoie une erreur « `InvalidFormatException` ». La valeur est alors remplacée par celle par défaut du fichier de configuration.

Formulaire incorrect – champ négatif

Quand le formulaire possède un champ négatif alors qu'il doit forcément être positif (exemple : « Largeur Image Destination »), alors le programme renvoie une erreur « `IllegalArgumentException` ». La valeur est alors remplacée par celle par défaut du fichier de configuration.

Formulaire incorrect – valeur largeur ou hauteur image destination trop grande

Si la valeur de la Largeur ou de la Hauteur est trop grande alors le programme renvoie `IllegalArgumentException`. Les valeurs sont alors restaurées par celle dans le fichier de configuration.

2 Fichier de configuration

Notre application utilise un fichier de configuration permettant d'initialiser certaines valeurs par défaut du programme. Toutes ces valeurs sont nécessaires au fonctionnement du programme.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>

  <entry key="personneX">0</entry>
  <entry key="personneY">800</entry>

  <entry key="imageDestWidth">1920</entry>
  <entry key="imageDestHeight">1080</entry>

  <entry key="rayonCylindre">80</entry>
  <entry key="distImage">300</entry>

  <entry key="widthSrc">1280</entry>
  <entry key="heightSrc">960</entry>

  <entry key="pathToSavedImage">./images/</entry>
</properties>
```

Figure 2 – Fichier de configuration

2.1 Conditions de test

Les tests à réaliser sont :

Test de la présence de toutes les valeurs

Conditions :

- Les valeurs sont présentes dans le fichier

Test du format des valeurs

Conditions :

- Les valeurs numérique doivent pouvoir être reconnues par la fonction Integer.parse() de Java
- Les valeurs doivent être positives pour les valeurs numériques, sauf coordonnées Observateur et Distance Image
- Les multiplications des valeurs « imageDestWidth » par « imageDestHeight » et « widthSrc » par « heightSrc » doivent être inférieurs à 16 000 000 permettant ainsi de ne pas dépasser des limites de mémoire

2.2 Cas de tests

2.2.1 Liste des cas

Voici une liste des cas à traiter afin de vérifier que le fichier de configuration est correct :

- Le fichier de configuration est correct et la fonction setFile() renvoie vrai
- Le fichier de configuration est incorrect car un champ numérique est incorrect
- Le fichier de configuration est incorrect car un champ de caractères contient des caractères spéciaux
- Le fichier de configuration est incorrect car un champ est manquant

2.2.2 Résultat

Fichier de configuration Valide

Si le fichier est correct, la fonction LoadFromXML.setFile() renvoie vrai.

Fichier de configuration incorrect – Champ numérique incorrect

Si un champ numérique est incorrect la fonction LoadFromXML.setFile() renvoie faux et une erreur NumberFormatException est enregistrée en Log.

Fichier de configuration incorrect – Champ caractère invalide

Si le champ de caractères pour le fichier contient des caractères non valides pour la création d'un répertoire, alors une exception « InvalidPathException » est enregistrée en log et la fonction LoadFromXML.setFile() renvoie faux.

Fichier de configuration incorrect – Champ manquant

Si un champ du fichier de configuration est manquant alors la vérification de celui-ci échoue et la fonction LoadFromXML.setFile() renvoie faux.

3 Panneau d'aide à l'impression

Afin de pouvoir imprimer une image, une fonction a été implémentée afin de définir les bons paramètres. Cette fenêtre doit donc être testée.

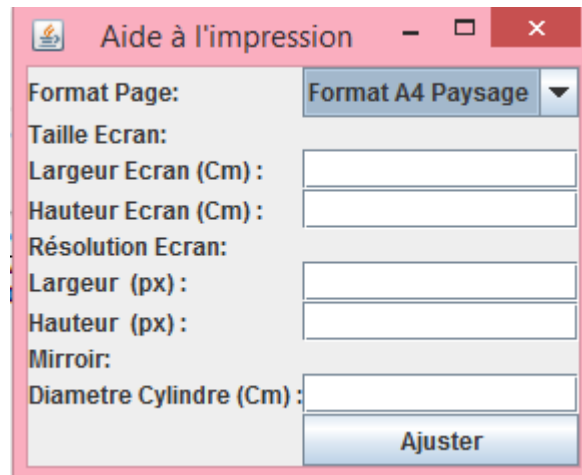


Figure 3 – Panneau d'aide à l'impression

3.1 Conditions de test

Test du champ Largeur Ecran

Conditions :

- Ce champ doit être un Réel positif reconnu par la fonction « Float.parse() » de Java

Test du champ Hauteur Ecran

Conditions :

- Ce champ doit être un Réel positif reconnu par la fonction « Float.parse() » de Java

Test du Champ Largeur Résolution

Conditions :

- Ce champ doit être un Entier positif reconnu par la fonction « Integer.parse() » de Java

Test du champ Hauteur Résolution

Conditions :

- Ce champ doit être un Entier positif reconnu par la fonction « Integer.parse() » de Java

Test du Champ Diamètre Cylindre

Conditions :

- Ce champ doit être un Réel positif reconnu par la fonction « Float.parse() » de Java

3.2 Cas de tests

3.2.1 Liste des cas

Voici une liste des cas à traiter afin de valider que le formulaire d'aide à l'impression soit correct :

- Le formulaire est correctement rempli
- Le formulaire contient une valeur négative
- Le formulaire contient une valeur incorrecte
- Le formulaire contient un champ vide

3.2.2 Résultat

Formulaire Valide

Si le formulaire est valide alors `printHelper.valideForm()` renvoie vrai et l'exécution du programme continue

Formulaire Invalide – valeur négative

Si le formulaire contient une valeur négative alors la fonction `validForm()` renvoie faux et l'erreur « valeur négative » est logger dans les fichiers d'erreur.

Formulaire Invalide – valeur incorrecte

Si le formulaire contient une valeur incorrecte alors la fonction `validForm()` renvoie faux et l'erreur `NumberFormatException` est logger dans les fichiers d'erreur.

Formulaire Invalide – champ vide

Si le formulaire contient une valeur vide alors la fonction `validForm()` renvoie faux et l'erreur `NumberFormatException` est logger dans les fichiers d'erreur.



Comptes rendus hebdomadaires

Compte rendu n°1 du 21/09/2016

Prise de contact avec l'encadrant afin d'avoir plus d'informations sur le sujet.

Recherche effectuée sur les « anamorphoses cylindriques ».

Recensement de différentes sources dans un fichier Word.

Etude d'une application faite en java avec la librairie Open GL mais qui n'a pas l'air de fonctionner.

Ecriture dans un fichier Word des différentes formules mathématiques nécessaires à la réalisation d'une anamorphose.

Envoie d'un mail à un enseignant secondaire Mathieu Blossier afin de savoir s'il était possible d'avoir l'application et les sources de son programme Java sur les anamorphoses, la brochure spécifiant que ces logiciels sont libres de droit pour un usage non commercial [lien](#) page 9.

Compte rendu n°2 du 28/09/2016

Recherche de documents en anglais sur les anamorphoses.

Récupération du code de M. Blossier sur les anamorphoses et début de tests et correction.

Ajout de formules mathématiques trouvées sur des sources trouvées en anglais.

Synthèse dans un document des projets existants sur les anamorphoses afin de situer le projet par rapport à ce qui a déjà été fait par le passé.

Création d'un cahier de spécification.

Compte rendu n°3 du 05/10/2016

Ecriture du cahier de spécification. Il manque les IHM et le planning prévisionnel.

Début de l'écriture du Rapport de projet (Introduction + Etat de l'art).

Compte rendu n°4 du 12/10/2016

Rédaction du rapport sur l'état de l'art.

Modification des tâches dans le cahier de spécifications.

Début de la programmation de la 1ère partie de l'application sur les anamorphoses cylindrique sur une image.

Compte rendu n°5 du 19/10/2016

J'ai commencé à travailler sur la formule et son application dans un prototype Java.

Pour cela, j'ai utilisé une des formules que j'ai trouvé sur internet. D'ailleurs, plusieurs d'entre elles semblent erronées et c'est peut-être le cas aussi pour celles que j'utilise.

Compte rendu n°6 du 26/10/2016

Développement de l'interface avec la saisie de paramètres dans l'application en cours de réalisation.

Choix du modèle mathématique à tester en concertation avec mon encadrant.

Modification du cahier de spécification.

Compte rendu n°7 du 02/11/2016

Ajout de la fonctionnalité de Zoom pour avoir un affichage à l'échelle d'une très grande feuille pour vérification.

Ajout de la fonctionnalité de saisie des paramètres par l'utilisateur.

Recherche en cours sur la conversion cm en pixel qui pose problème car dépend du DPI de l'image.

Séparation de l'affichage anamorphose de l'interface de paramétrage.

Prise de rendez-vous avec M. Raveau pour parler du Cahier de Spécification

Compte rendu n°8 du 09/11/2016

Mise en place du nouveau repère pour l'anamorphose.

Mise en place d'un mode test pour travailler sur un carré de dimension la taille du rayon.

Correction de la formule mathématique utilisée.

Modification du cahier de spécification ajout du détail des fonctionnalités et des contraintes d'exploitation.

Modification du rapport d'état de l'art afin d'ajouter la méthode de transformation que l'on va utiliser.

A faire pour les séances à venir :

Effectuer des recherches sur les librairies de webcam afin de fixer une librairie pour les flux vidéo.

Etudier le code de programme de M. Blossier sur la fonction de remplissage car notre image déformée est plus grande que notre image source. Donc il faut remplir entre chaque colonne de pixel.

Compte rendu n°9 du 16/11/2016

Ajout de l'utilisation d'une webcam avec la librairie « webcam recording ».

Ajout de l'affichage de la webcam dans l'application avec la capture d'une image.

Choix de la librairie « Vlcj » pour les fichiers média supporte les images et les fichiers vidéo.

Recherche d'un moyen de récupérer les images frame par frame d'une vidéo.

Recherche d'un moyen d'intégrer le lecteur dans l'application.

A faire pour les séances à venir :

Fonction de remplissage.

Compléter le rapport avec les recherches effectuées sur les librairies vidéo et préparer la soutenance.

Compte rendu n°10 du 23/11/2016

Travail sur le cahier de spécification et le rapport d'état de l'art.

Ajout de la partie Analyse au rapport.

Compte rendu n°11 du 30/11/2016

Travail sur l'analyse des prototype et sur le rapport du PRD.

Compte rendu n°12 du 7/12/2016

Correction du rapport et préparation de la soutenance de mi-parcours

Compte rendu n°13 du 4/1/2017

Mise en place de l'application en suivant le Diagramme de Classe

Correction des problèmes d'arrêt de la transformation quand il n'y a pas assez d'images

Mise en place de la possibilité d'arrêter l'anamorphose

Ajout de l'utilisation d'un tableau de coordonnées de pixels pour accélérer la transformation

Problème subsistant :

Les threads s'arrêtent au bout d'un certain temps sans activité

L'image n'est pas remplie

A faire la semaine prochaine :

Tester l'application sur l'écran tactile

Implémenter le déplacement du cylindre par un clic

Corriger le problème de thread

Chercher une solution pour remplir l'image à l'aide du tableau de coordonnées

Compte rendu n°14 du 11/1/2017

Correction temporaire du problème de thread inactif par l'affichage de message dans le terminal

Test de l'application sur Ecran tactile (Les capacités du PC derrière l'écran permettent à peine de déformé une vidéo)

Ajout de la fonctionnalité de déplacement du cylindre à l'aide d'un clic sur l'écran.

Compte rendu n°15 du 18/1/2017

Correction de l'utilisation du thread en continue par affichage de message par une solution créant plusieurs threads pour chaque utilisation

Entretien avec l'encadrant :

Tout semble correct.

Une solution pour le remplissage serait de faire soit même le scale avec des valeurs en double et de passer soit même toute les valeurs dans la fonction d'anamorphose pour travailler sur des valeurs réel et les transformer en entier pour l'affichage finale.

Mise en place de la solution proposée avec succès.

Soucis cela ralentit l'application.

Travail sur l'amélioration des performances.

Compte rendu n°16 du 25/1/2017

Amélioration des performances de calcul en parallélisant la fonction de remplissage.

Correction de la fonction setup qui doit passer en priorité par rapport aux autres opérations

Correction de la vitesse de traitement pour la caméra.

Mise en place de Pool de Thread permettant de gérer les exécutable.

Ajout d'un JSlider permettant en temps réel de changer l'angle de vision de 0 à 180°

Refonte du code et mise en place d'une version 3

Fonctionnalité supplémentaire :

Ajout du Menu de changement de type d'anamorphose

Le panneau de configuration est maintenant propre à l'anamorphose choisit

Le bouton de démarrage de l'anamorphose est maintenant découplé de la classe Anamorphose car elle a maintenant effet sur la source.

Reste à faire :

Gérer le souci d'affichage des images qui s'entrelace (problème de thread)

Ajouter un bouton pour changer à chaud les paramètres pour l'anamorphose

Mettre dans le fichier de configuration un paramètre pour la taille désiré du flux d'entrées

Utiliser le panneau d'affichage de message pour afficher certaines informations

Commenter le code

Compte rendu n°17 du 1/2/2017

Ajout d'un bouton pour changer à chaud les paramètres pour l'anamorphose

Ajout d'une classe pour la gestion des paramètres du fichier XML et ajout des attributs de l'image source

Recherche sur l'utilisation de la classe Future pour appeler toute les taches en même temps

Demande de modification de l'écran au Service informatique afin de pouvoir faire tourner l'application

Après entretiens avec M. Aupetit, il a été convenu que la gestion des images et le passage dans des listes n'était pas optimale et que je devrais passer par des buffers

Reste à faire :

Implémenter les modifications conseillées par M. aupetit

Ajouter des informations au panneau d'affichage

Finir de commenter le code

Compte rendu n°18 du 8/2/2017

Test de l'application sans les modifications conseillé par M. Aupetit.

L'application tourne trop au ralentit et consomme trop de ressource

La machine ne possédant que 1go de RAM c'est trop insuffisant pour faire marcher ce programme

Après un second entretien avec M. Aupetit pour demander conseil à propos des changements qu'il m'avait demandé d'effectuer, la méthode la plus simple de gérer le problème ne serait pas d'utiliser des buffers mais des sémaphores.

Afin d'éviter une trop grosse utilisation de la mémoire il serait aussi conseiller d'utiliser des tableaux d'entiers au lieu des images.

L'utilisation d'Executor Service pour paralléliser l'application deviens inutile avec les sémaphores de plus que la machine n'as que deux cœurs et donc paralléliser la transformation est inutile.

Application de toutes les modifications et test sur la machine.

L'application fonctionne plus rapidement avec quelques ralentissements certainement à cause des paramètres de la JVM.

Reste à faire :

Commenter le nouveau code

Modifier la distance d'affichage de l'image par rapport au cylindre

Demander l'obtention d'une caméra pour tester l'application avec une caméra sur l'appareil test

Compte rendu n°19 du 15/2/2017

Ajout d'un champ pour la distance de l'image par rapport au cylindre

Code commenté

Test avec une caméra

Création d'un guide utilisateur

Reste à faire :

Afficher une deuxième fenêtre avec une image fixe déformée ou imprimer une image déformée

Compte rendu n°20 du 1/3/2017

Mise en place d'une fonction d'export de l'image déformé

Ajout d'une boîte de dialogue si le fichier de configuration n'est pas présent

Ajout d'un fichier de configuration pour supprimer les messages de Log de la librairie de WebCam

Recherche sur la possibilité de basculer sur le mode caméra sans caméra de connecter. Sans résultat pour le moment.

Création de la JavaDoc

Création du nouveau diagramme de classe à partir des fichiers Java

Entretiens sur la qualité du code avec Carl Esswein

Reste à faire

Commenté plus exhaustivement le code
 Mettre en place des tests unitaires
 Mettre en place des outils de mesures de métrique (SonarQube)
 Automatisé l'export au format d'une feuille A4 (introduire les paramètres nécessaire taille écran en cm / résolution écran en pixel et les différents formats de feuille en cm / pixel)
 Implémenté des fonctions de Rastérisation grâce aux anciens cours de M.Aupetit

Compte rendu n°21 du 8/3/2017

Ajout de commentaire sur les variables
 Mise en place de SonarQube et analyse du code
 Application de nombreux correctif pour corriger 118 améliorations proposées par Sonar
 Ecriture de certains tests unitaire

 Reste à faire
 Ecrire les rapports
 Finir de mettre en place des tests unitaires
 Automatisé l'export au format d'une feuille A4 (introduire les paramètres nécessaire taille écran en cm / résolution écran en pixel et les différents formats de feuille en cm / pixel)
 Implémenté des fonctions de Rastérisation grâce au ancien cours de M.Aupetit

Compte rendu n°22 du 15/3/2017

Test unitaire
 Correction de bug et optimisation mémoire
 Rédaction des rapports
 Mise en place d'une fenêtre d'aide au paramétrage pour un type de feuille donnée.

 Reste à faire
 Ecrire les rapports de PRD et de conduite de projet
 Ecrire les documentations Utilisateurs et Développeur
 Finir de mettre en place des tests unitaires pour le fichier de configuration et pour la nouvelle fenêtre d'aide
 Implémenté des fonctions de Rastérisation grâce aux ancien scours de M.Aupetit pour combler définitivement les trous dans l'image

Compte rendu n°23 du 22/3/2017

Test unitaire sur le fichier de log et sur la nouvelle fenetre d'aide à l'impression
 Ecriture des documents suivant :
 - Cahier développeur
 - Cahier de test
 - Analyse Sonar et JMC
 Génération de la documentation complète

 Reste à faire :

Rapport Gestion de Projet

Rapport PRD

Compte rendu n°24 du 29/3/2017

Ecriture du Rapport Gestion de Projet

Ecriture du Rapport PRD

Flux vidéo et anamorphose cylindrique

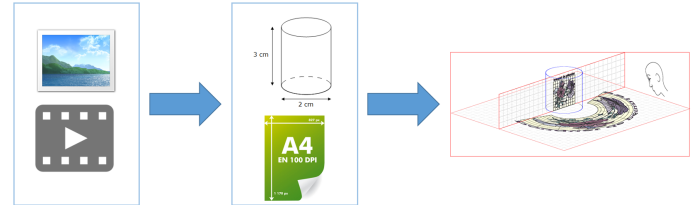
Alexis Saludo

Encadrement : Christophe Lente

Objectifs

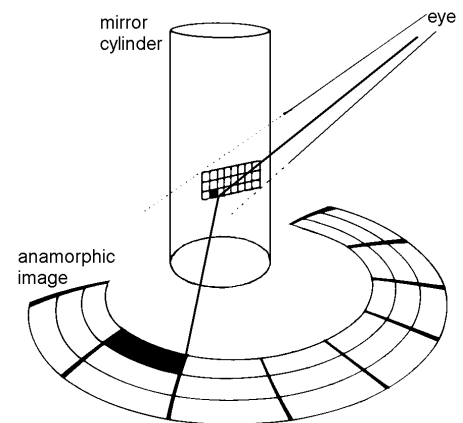
Notre application doit permettre de :

- Choisir une source de flux
- Configurer notre anamorphose cylindrique
- Afficher en continue la transformation anamorphique du flux
- Imprimer une image du flux transformée à un instant t



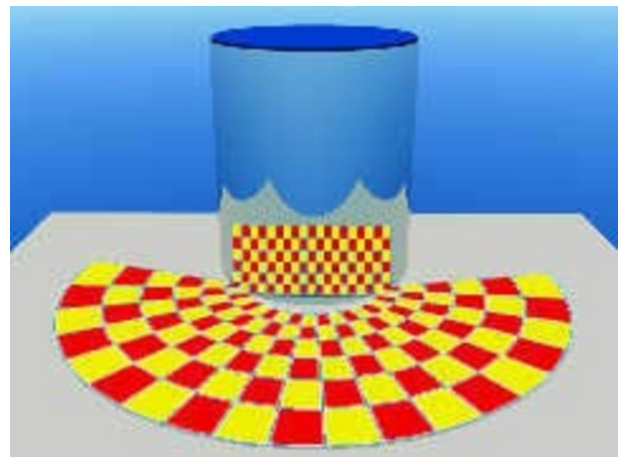
Mise en œuvre

- Développement d'un logiciel pour effectuer une anamorphose cylindrique
- Utilisation de formule mathématique de transformation
- Utilisation de bibliothèques de traitement de flux vidéo



Résultats attendus

- Utilisation de divers types de **flux vidéo**
- Transformation anamorphique **réalisable** d'un flux vidéo
- Application **modulaire**



Flux vidéo et anamorphose cylindrique

Alexis Saludo

Encadrement : Christophe Lente

Objectifs

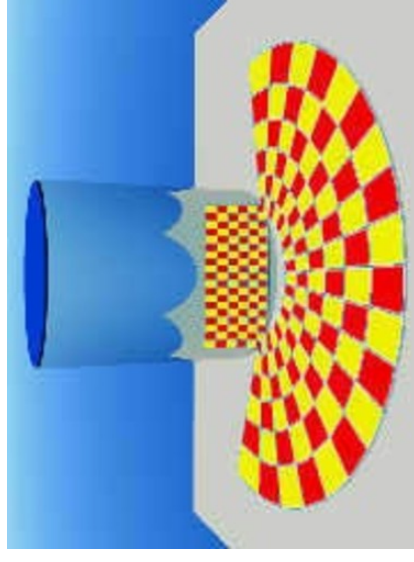
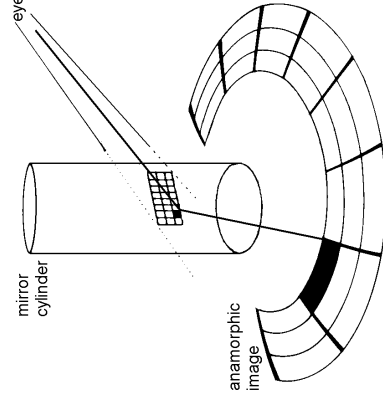
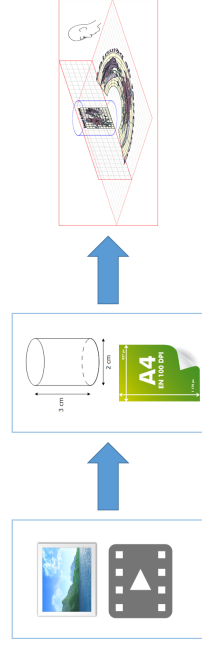
- Notre application doit permettre de :
- Choisir une source de flux
 - Configurer notre anamorphose cylindrique
 - Afficher en continu la transformation anamorphique du flux
 - Imprimer une image du flux transformée à un instant t

Mise en œuvre

- Développement d'un logiciel pour effectuer une anamorphose cylindrique
- Utilisation de formule mathématique de transformation
- Utilisation de bibliothèques de traitement de flux vidéo

Résultats attendus

- Utilisation de divers types de **flux vidéo**
- Transformation anamorphique **réalisable** d'un flux vidéo
- Application **modulaire**



Flux vidéo et anamorphose cylindrique

Résumé

Ce projet destiné à être exposé dans des salons permet de présenter la transformation de flux vidéo par anamorphose cylindrique grâce à une formule mathématique.

Mots-clés

Image, Anamorphose, Vidéo

Abstract

This project intended to be exhibited in live room is to show the transformation of video streams by cylindrical anamorphosis with a mathematical formula.

Keywords

Picture, Anamorphosis, Video