

Projet *Monofin*

28/05/09


Encadreur Patrice LEGUESDRON Laurent MONIER Fulgence RAZAFIMAHERY	Rapporteur Mireille DUCASSÉ
---	---------------------------------------

CHAUDET Yoann, GARCIA Paul, GAUTIER Quentin, LE SQUER Nicolas, MUSSET Nicolas, VILLOING Xavier

1

Introduction

Introduction - Monopalme



2

Introduction

Plan de l'exposé

- Introduction
- Application
 - Interface graphique
 - Editeur de dessin
 - Extraction de contours
- Structure interne
 - Structure de données
 - Interface COMSOL
- Organisation
 - Gestion de projet
 - Planification
- Conclusion

3

Introduction

Contexte du projet - besoins

- Simuler le comportement d'une monopalme dans l'eau
- Tester modèles mathématiques
- Rechercher les formes les plus « efficaces »

4

Introduction

Contexte du projet - conséquences

- Utilisation de logiciels de simulation complexes
- Temps pour pouvoir lancer une simulation long
 - Dessin de la palme
 - Utilisation de modeleurs 3D
 - Dessin avec COMSOL
 - Configuration de la simulation

5

Introduction

Contexte du projet - problématique

- Tester différentes formes, long
 - Modeleur 3D : temps de dessin, long et compliqué
 - COMSOL : long et peu puissant
- Mettre au point, long
- Logiciels complexes pas forcément adaptés à des utilisateurs non-informaticiens

6

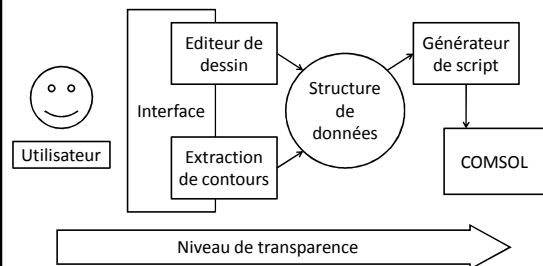
Introduction

Objectifs du projet

- Proposer:
 - Une interface claire et intuitive
 - Un fonctionnement simple et puissant
- Permettre:
 - Modéliser une monopalme en 3D facilement
 - Créer des projets COMSOL sans intervention humaine

7

Introduction

Objectif - Mise en œuvre

8

Interface graphique

9

Interface graphique

Interface graphique (1/4)

- Première version : assistant
 - caractéristiques
 - étapes balisées et progressives
 - avantages / inconvénients
 - facilité d'utilisation
 - peu flexible, peu de liberté pour l'utilisateur
- Deuxième version : mode projet
 - caractéristiques
 - espace de travail
 - multi-fenêtré
 - avantages / inconvénients
 - utilisation un peu plus experte
 - flexibilité

10

Interface graphique

Interface graphique (2/4)

- Architecture
 - utilisation de la librairie Qt
 - nombreuses classes disponibles
 - structures de donnée
 - classes de disposition
 - boutons, menus, zones de saisie, etc.

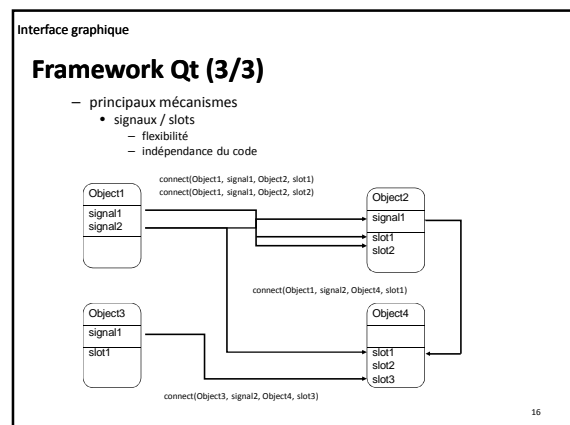
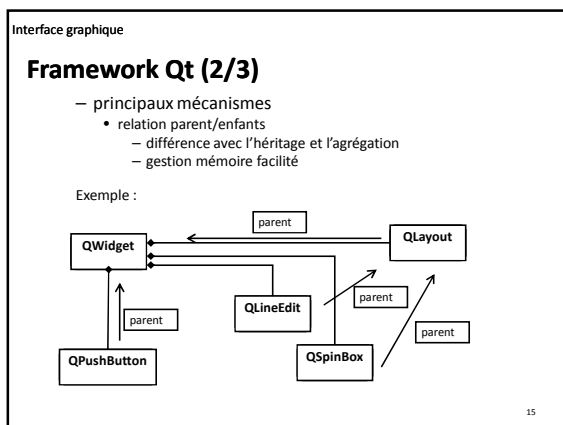
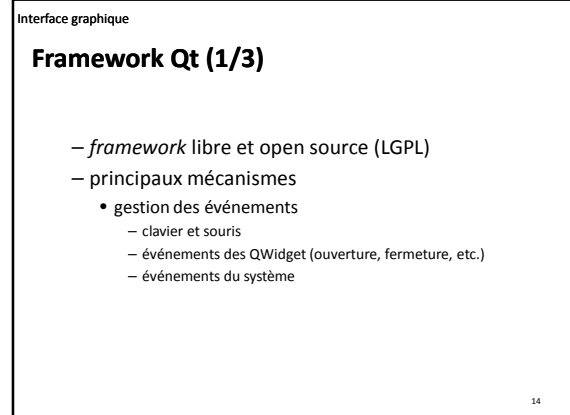
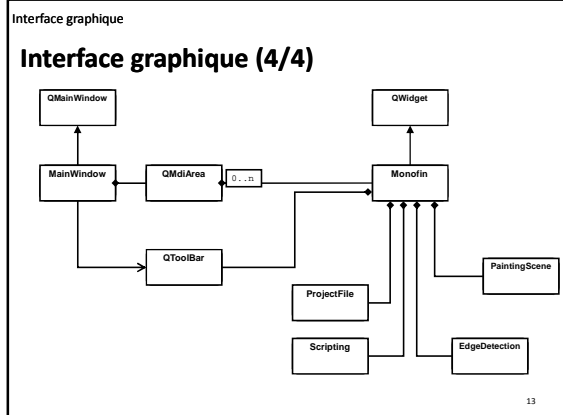
11

Interface graphique

Interface graphique (3/4)

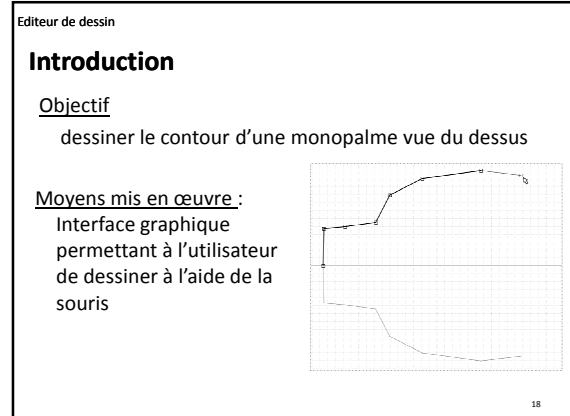
- Architecture
 - deux classes principales
 - MainWindow (hérite QMainWindow)
 - gère la zone MDI
 - gère les menus
 - affiche les barres d'outils
 - Monofin (hérite QWidget)
 - gère la zone de dessin
 - gère les barres d'outils

12



Fonctionnalités et structure de données du module de dessin de la surface

17



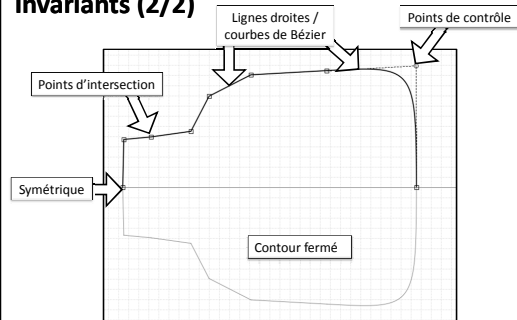
Editeur de dessin

Invariants (1/2)

- Dessiner une palme symétrique
- Dessiner à l'aide de trois éléments principaux :
 - Points d'intersections
 - Lignes (droites ou courbes de Bézier)
 - Points de contrôle pour les courbes de Bézier
- Créer un contour fermé

19

Editeur de dessin

Invariants (2/2)

20

Editeur de dessin

Fonctionnalités

- Fonctionnalités d'un éditeur « basique » de dessin : ajout/suppression de points...
- Très adaptées au problème (intérêt de ne pas reprendre un éditeur existant)
- Respectent les invariants

21

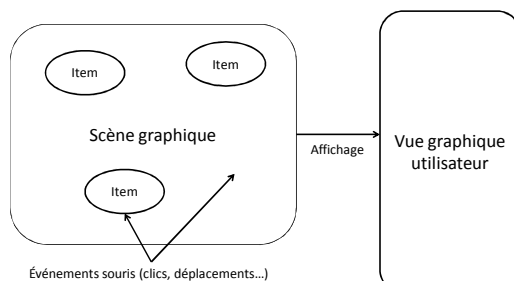
Editeur de dessin

Implémentation

- Interface uniquement basée sur la bibliothèque graphique Qt
- Modèle de Qt parfaitement adapté à ce problème : Scène – Items – Vue (graphiques)
- Modèle représenté par 3 classes dont dérivent celles du module de dessin

22

Editeur de dessin

Modèle Scène - Items – Vue (1/2)

23

Editeur de dessin

Modèle Scène - Items – Vue (2/2)

- Scène :
 - Contient un ensemble d'items graphiques
 - Gère cet ensemble d'items (ajout, suppression...)
 - Est le « moteur » de l'interface de dessin
- Items :
 - Sont des points d'intersection, lignes, points de contrôle
 - Possèdent leur propre fonction d'affichage
 - Peuvent gérer les événements souris
- Vue :
 - Affiche la scène (tous les items)
 - Permet de changer l'échelle (zoom)
 - Permet de changer les axes (inversion de l'axe y)

24

Editeur de dessin

Intégration avec les autres modules

- Intégration par la scène
- Ensemble de « *slots* » pour :
 - Conserver une liste des fonctionnalités
 - Faire un lien avec l'interface générale via les « signaux »
- Lien avec la façade de la structure interne pour :
 - Enregistrer chaque modification de l'utilisateur
 - Pouvoir récupérer l'intégralité de la structure pour l'afficher et la modifier

25

Extraction de contours

26

Extraction de contours

Introduction (1/2)

- **Objectif :** Modéliser automatiquement une forme de monopalmes à partir d'une image pour pouvoir l'intégrer dans le module de dessin
- **Solution :**
 - Créer une interface utilisateur permettant le chargement d'une image et le placement de cette image par rapport à un axe
 - Extraire la forme prédominante de l'image à l'aide de différents algorithmes et intégrer cette forme dans le module dessin

27

Extraction de contours

Introduction (2/2)

- Utilisation de deux algorithmes :
 - Pour la détection de contours : Algorithme des contours actifs (*Snakes*)
 - Pour la vectorisation de contours : **Potrace** – un algorithme de *tracing* basé sur des polygones

28

Extraction de contours

Algorithme détection de contours (1/2)

- Détection de contour :
 - Basé sur l'algorithme des contours actifs (*Snakes*) décrit par Kass et Witkin (1987)
 - Algorithme :
 - En entrée : une image (bitmap)
 - En sortie : une chaîne fermée de pixels représentant le contour de la forme à détecter

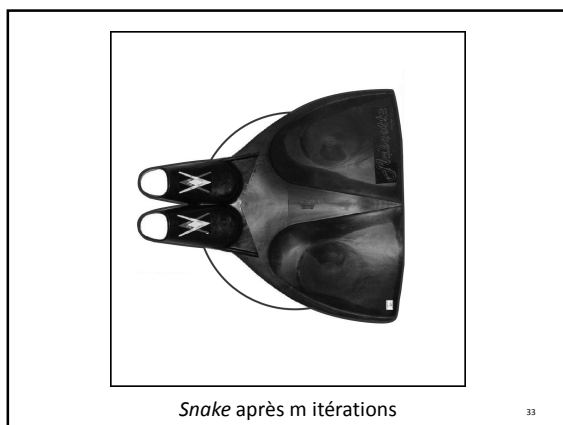
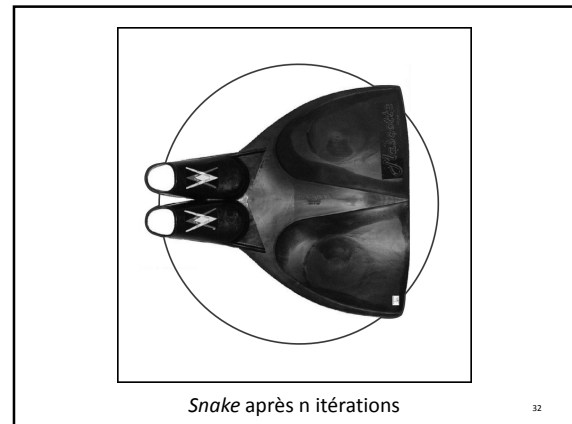
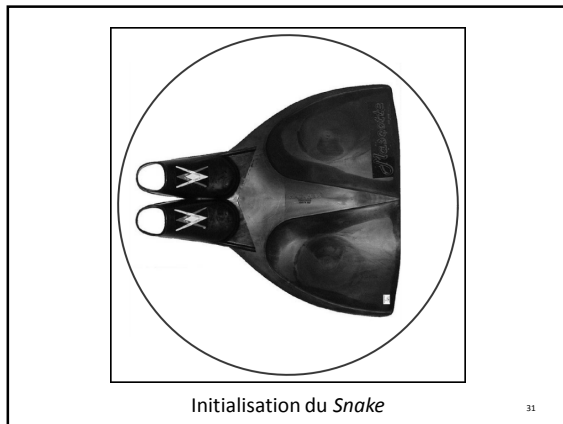
29

Extraction de contours

Algorithme détection de contours (2/2)

- Fonctionnement de l'algorithme :
 - Snake composé d'une multitude de points chacun rattaché à un pixel de l'image
 - Initialisation :
 - Tous les points sont situés sur un cercle englobant entièrement l'image
 - Déroulement :
 - A chaque itération, les points du Snake se rapprochent du centre de l'image
 - Si un point détecte un changement de luminosité il se fige sur le pixel auquel il est rattaché
 - Fin lorsque tous les points sont figés ou atteignent le centre de l'image

30



Extraction de contours

Algorithme de vectorisation de contours (1/2)

- Vectorisation de contours :
 - Adaptation de l'algorithme Potrace décrit par Peter Selinger (2003)
<http://potrace.sourceforge.net/>
 - Algorithme :
 - En entrée : une chaîne fermée de pixels (*Snake*)
 - En sortie : approximation vectorielle de la chaîne de pixels sous forme de courbes de Bézier

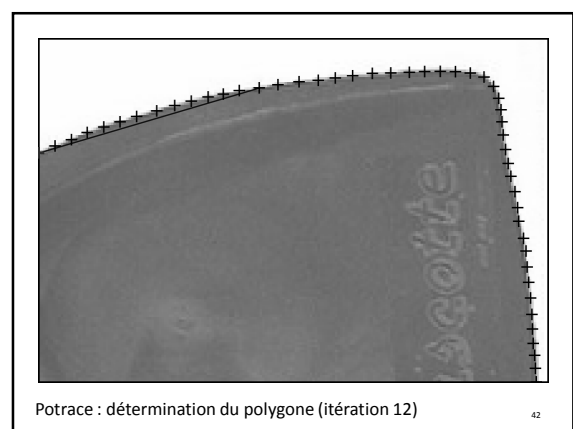
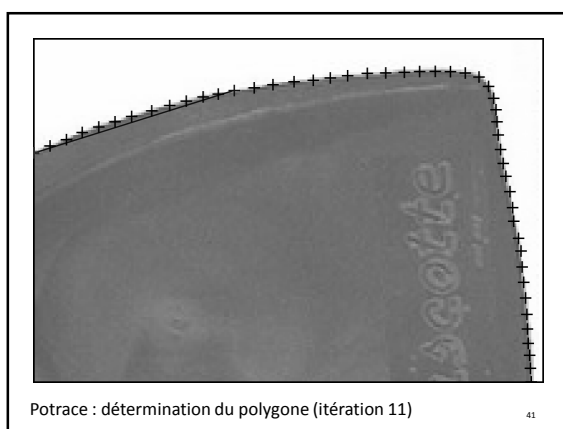
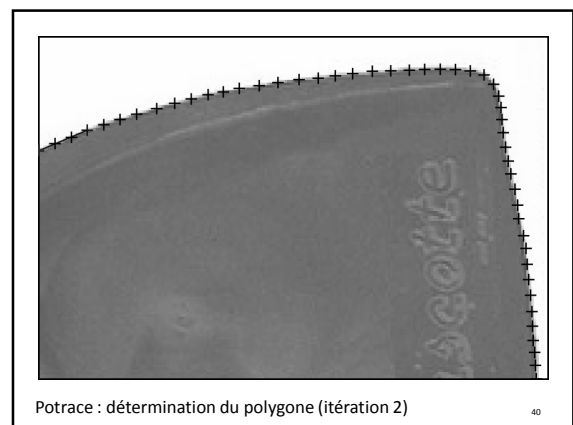
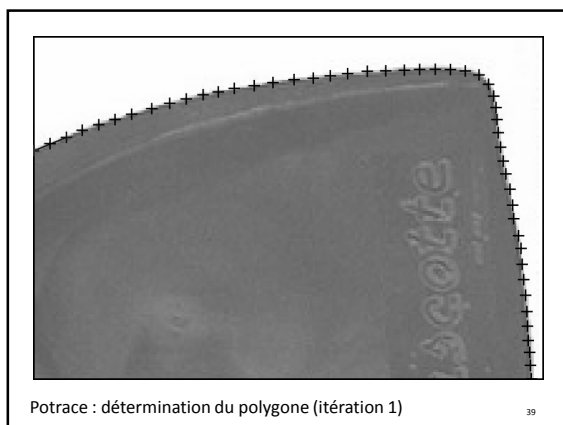
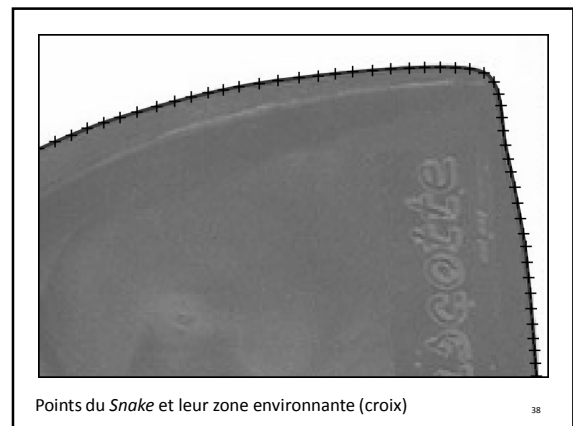
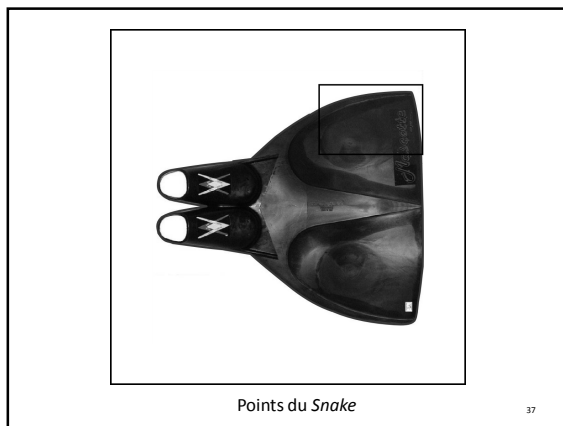
35

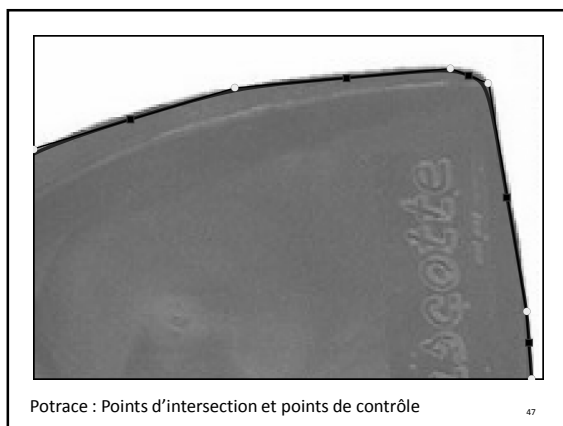
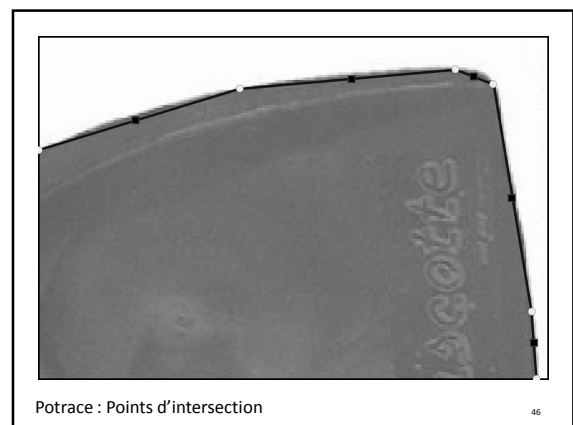
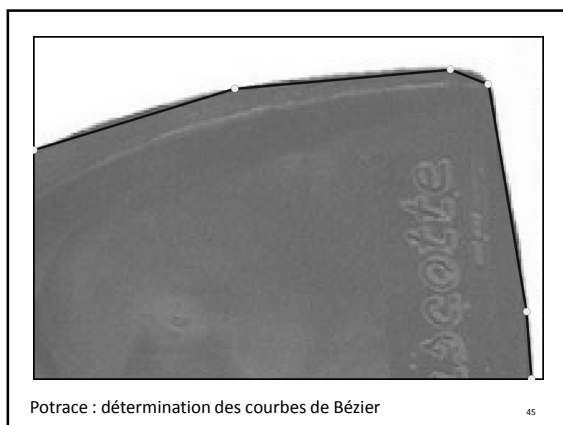
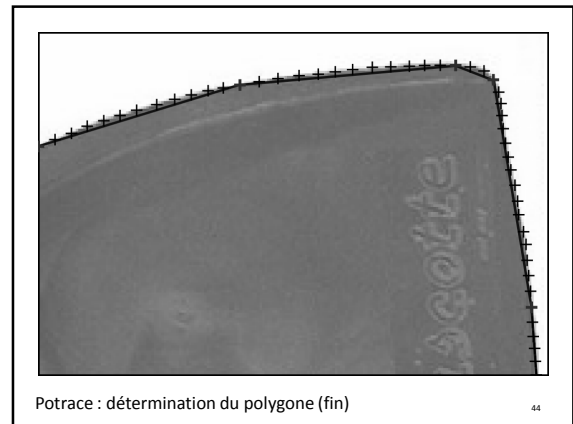
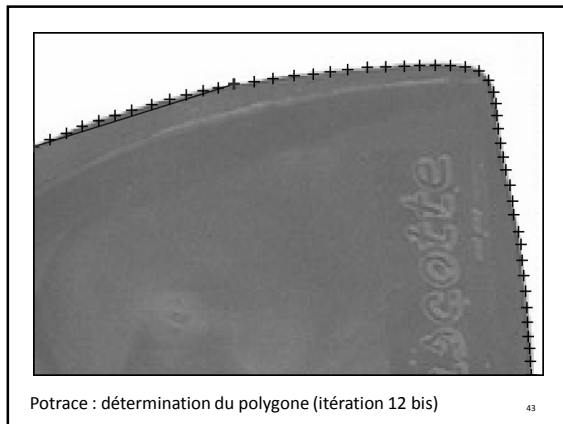
Extraction de contours

Algorithme de vectorisation de contours (2/2)

- Fonctionnement de l'algorithme :
 - Initialisation :
 - Initialisation d'une liste vide de sommets d'un polygone
 - Positionnement d'un pointeur sur un pixel donné du *Snake* et ajout de ce pixel à la liste de sommets
 - Déroulement (en deux étapes)
 - Etape 1 : Création d'un polygone approximant le tracé du *Snake*
 - Etape 2 : Détermination des points d'intersection et de contrôle de la courbe de Bézier

36



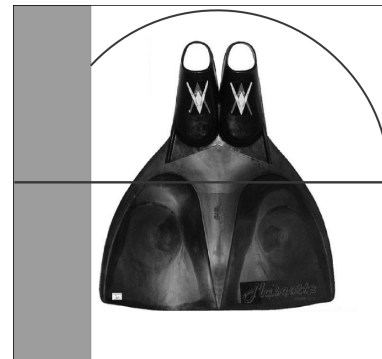


Extraction de contours

Algorithmes dans l'application

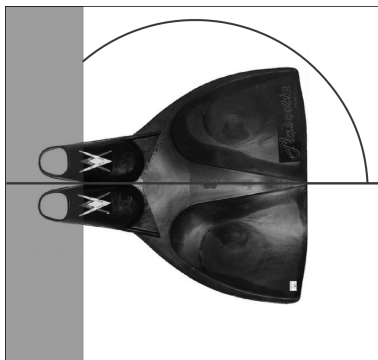
- Etapes d'extraction d'une forme :
 - Chargement de l'image
 - Positionnement de l'image par rapport à un axe
 - Lancement des algorithmes
 - Création de la forme extraite et intégration dans le module dessin

49



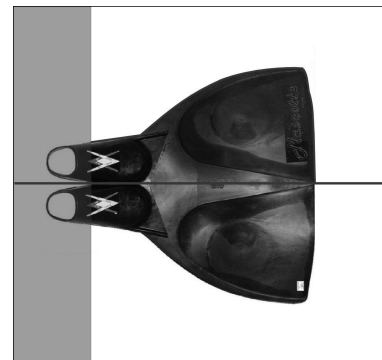
Chargement de l'image

50



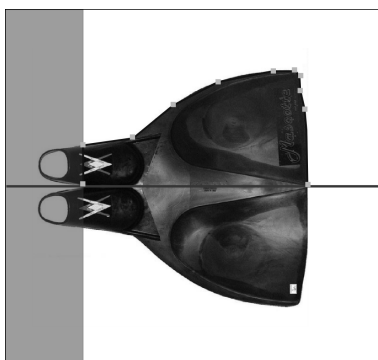
Positionnement de l'image par rapport à l'axe

51



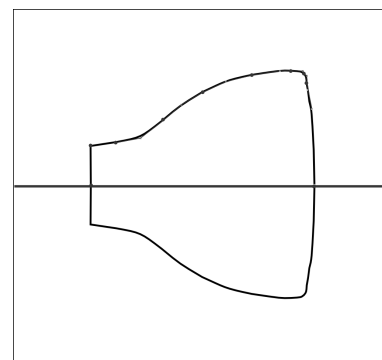
Algorithme de *Snake*

52



Algorithme : *Potrace*

53



Intégration de la forme dans le module dessin

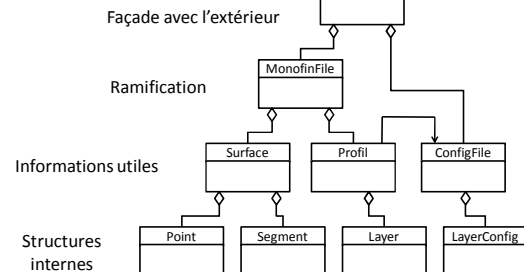
54

Structure de données

55

Structure de données

Structure Générale (1/2)



56

Structure de données

Structure Générale (2/2)

Profil Vs ConfigFile

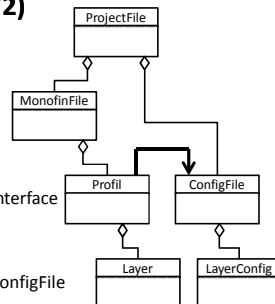
Profil pour la taille des strates

ConfigFile pour leur composition

Emplacements différents dans l'interface

Observateur

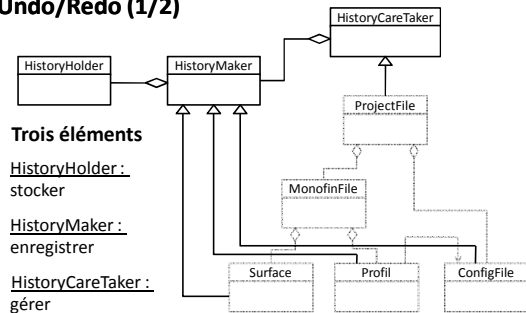
Profil notifie les changements à ConfigFile



57

Structure de données

Undo/Redo (1/2)

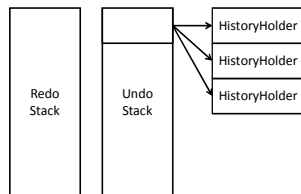


58

Structure de données

Undo/Redo (2/2)

HistoryCareTaker



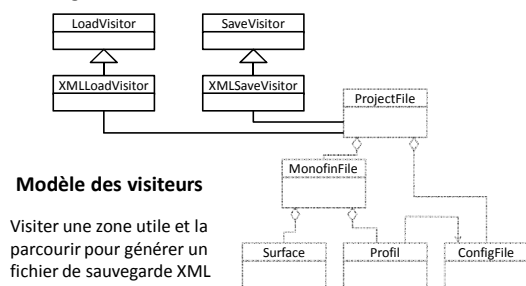
séquence :

- Dépiler la liste de Holders en sommet
- Envoyer chaque Holder à son Maker attribué
- Appeler les fonctions pour défaire
- Construire de nouveaux Holders pour le Redo

59

Structure de données

Sauvegarde (1/2)



60

Structure de données

Sauvegarde (2/2)**Format XML**

Stocke la liste des points, des segments, des strates et de leurs paramètres

```
<monofin>
  <segments>
    <segment number="0">
      <intersectionpoint number="0"/>
      <intersectionpoint number="1"/>
    </segment>
  </segments>
  <points>
    <intersection>
      <point number="0" X="0.000000" Y="0.000000"/>
      <point number="1" X="10.000000" Y="10.000000"/>
    </intersection>
  </points> ...
</monofin>
```

61

Interface COMSOL

62

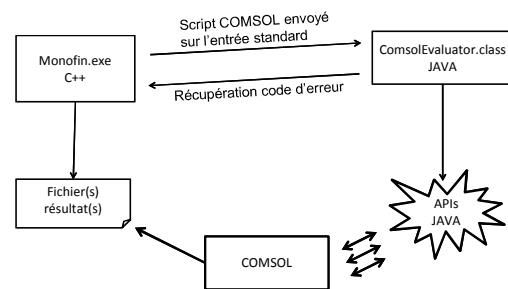
Interface COMSOL

Présentation COMSOL

- COMSOL : Environnement de simulation
 - COMSOL Multiphysics
 - Interface graphique
 - Familière de nos utilisateurs
 - COMSOL Script
 - Langage de script
 - Accès à toutes les fonctionnalités
 - APIs exposées à travers JAVA

63

Interface COMSOL

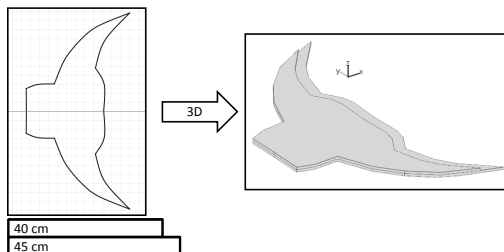
Processus de communication

64

Interface COMSOL

Script de visualisation

- Génère une image représentant la palme en 3D
- Apprécier le passage 2D à 3D

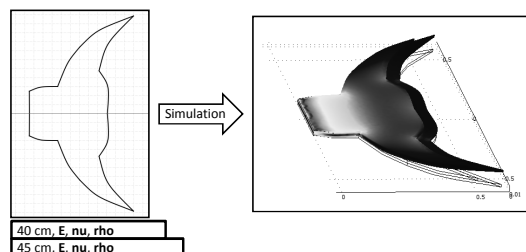


65

Interface COMSOL

Script de simulation

- Génère un fichier .MPH binaire et portable
- Simulation modale : *Eigenfrequency*



66

Interface COMSOL

Structure d'un script

```
clear monofin;
```

67

Interface COMSOL

Structure d'un script

```
clear monofin;
monofin.length = 45 * 1e-2;
```

68

Interface COMSOL

Structure d'un script

```
clear monofin;
monofin.length = 45 * 1e-2;
monofin.segments = [ ...
    struct('x', [430.72 430.56 430.4], 'y', [0 24.8 49.6]), ...
    struct('x', [430.4 453.6 492.8], 'y', [49.6 60 60]), ...
    struct('x', [492.8 514.4 541.44], 'y', [60 115.2 147.84]), ...
    struct('x', [541.44 578.4 660], 'y', [147.84 185.6 217]), ...
    struct('x', [660 599 584], 'y', [217 156 95]), ...
    struct('x', [584 607 602.4], 'y', [95 62 0]) ];
monofin.metaSegments = ...
    struct('absoluteLength', 171.68, 'dX', 430.72, 'dY', 0);
```

69

Interface COMSOL

Structure d'un script

```
clear monofin;
monofin.length = 45 * 1e-2;
monofin.segments = [ ... ];
monofin.metaSegments = ...;
monofin.layers = [ ...
    struct('thickness', 1 * 1e-2, 'length', 1, ...
        'E', 2000 * 1e6, 'nu', 0.33, 'rho', 7850), ...
    struct('thickness', 1 * 1e-2, 'length', 0.868889, ...
        'E', 2000 * 1e6, 'nu', 0.33, 'rho', 7850) ];
```

70

Interface COMSOL

Structure d'un script

```
clear monofin;
monofin.length = 45 * 1e-2;
monofin.segments = [ ... ];
monofin.metaSegments = ...;
monofin.layers = [ ... ];
monofin.settings = ...;
```

71

Interface COMSOL

Structure d'un script

```
clear monofin;
monofin.length = 45 * 1e-2;
monofin.segments = [ ... ];
monofin.metaSegments = ...;
monofin.layers = [ ... ];
monofin.settings = ...;
path('C:\Monofin\scripts', path);
main_default(monofin);
```

72

Interface COMSOL

Structure d'un script

```
clear monofin;
monofin.length = 45 * 1e-2;
monofin.segments = [ ... ];
monofin.metaSegments = ...;
monofin.layers = [ ... ];
monofin.settings = ...;
path('C:\Monofin\scripts', path);
main_default(monofin);
```

```
graph LR
    main_viewer.m --> build_geometry.m
    main_default.m --> build_geometry.m
    main_default.m --> build_fem.m
```

73

Gestion du projet

74

Organisation du projet

Organisation (1/2)

- Réunions hebdomadaires
 - bilan,
 - répartition des tâches.
- Répartition en équipes
 - un module = une équipe

75

Organisation du projet

Organisation (2/2)

- Itérations
 - réalisation d'un prototype
 - utilisation de SVN
- Tests
 - unitaires : au sein de chaque équipe
 - d'intégration : à l'ajout des modules dans l'interface

76

Planification

77

Planification

Suivi de planification

- Organisation des tâches différente de la planification prévue
 - Apparue lors de la conception
- « Equipes » de une à deux personnes
 - Réunions de mise en commun
- Décompte à la semaine
 - Beaucoup de variation dans les horaires

78

Planification

Comparaison	
Planification	Résultat
<u>Conception : 540 heures</u> Architecture générale (60h) Interface générale (100h) Editeur de dessin (260h) Interface COMSOL (120h)	<u>Conception : 343 heures</u> Structure de données (75h) Fenêtrage (60h) Extraction de contour (50h) Editeur de dessin (68h) Interface COMSOL (48h) Mise en commun (42h)
<u>Réalisation : 820 heures</u> Mêmes rubriques + tests unitaires	<u>Réalisation : 915 heures</u> Mêmes rubriques + tests unitaires + intégration

79

Planification

Etat d'avancement
<ul style="list-style-type: none"> Projet : Un noyau avec beaucoup de fonctionnalités <ul style="list-style-type: none"> Fonctions principales : OK <ul style="list-style-type: none"> Éditeur de dessin, strates, extraction de contours, sauvegarde, script, simulation Fonctions secondaires <ul style="list-style-type: none"> Bibliothèque de formes, aperçu avant simulation : OK Plusieurs autres fonctions en attente

80

Planification

Enseignements
<ul style="list-style-type: none"> Difficile de prévoir les tâches à réaliser de façon précise avant la conception Limite entre conception et construction floue Beaucoup de temps pour les tests d'intégration et le débogage

81

Conclusion

<ul style="list-style-type: none"> Projet initié cette année <ul style="list-style-type: none"> Grande liberté dans les choix techniques : Qt, C++ Réelles attentes des utilisateurs Volonté de poser des bases solides Pistes de poursuite <ul style="list-style-type: none"> Simulation dynamique Post-traitements à l'aide de MATLAB Comparaison des résultats avec le robot Permettre de créer de meilleurs monopalmes !

82

