UNIVERSITE SIDI MOHAMED BEN ABDELLAH
FACULTE DES SCIENCES ET TECHNIQUES FES
DEPARTEMENT D'INFORMATIQUE

Projet de Fin d'Etudes

Licence Sciences et Techniques Génie Informatique

Etude et réalisation d'une application Logiciel Interface graphique (CAO), Simulation numérique et Visualisation

---

Lieu de stage :Ecole Nationale des Sciences Appliquées d'Oujda

Réalisé par :

MEJDOUBI Abderrahmane

Encadré par :

Pr. RAHMOUNE Mohamed (ENSAO)

Pr. MAJDA Aicha (FSTF)

Pr. BENABBOU Abderrahim (FSTF)

Soutenu le 15/06/2012 devant le jury composé de :

Pr. BENABBOU Rachid

Pr. ZAHIAzzedine

Pr. MAJDA Aicha

Année Universitaire 2011-2012

Etude et réalisation d’une application Logiciel : Interface graphique (CAO), Simulation numérique et Visualisation

Chute libre

Distance de sécurité

Etude de crash

Dédicace

AuDIEU tout puissant mon créateur.

À ma mère, en témoignage de ma profonde gratitude et de mon incontestable reconnaissance, pour ses prières et son soutien toutau long de mon cursus, toute la confiance qu'elle m'accorde et tout l'amour dont elle m'entoure,

À monpère, celui qui m'a toujours indiqué la bonne voie,en signe d'amour, de gratitude pour tous les soutiens et les sacrifices dont il a fait preuve à mon égard,

Nul mot ne saurait exprimer à sa juste valeur le dévouement et le profond respect que je porte envers vous.Je remercie le bon DIEU qui a illuminé ma vie par votre présence. Que DIEU vous préserve et vous accorde, santé, bonheur et prospérité,

À mes deux chers frères, que je leur souhaite une vie pleine de bonheur et de succès,

À toute ma famille,

Àmes amis, et à tous mes proches,

À tous ceux qui m'aiment, je dédie ce modeste travail…

Abderrahmane

Remerciements

Je tiens à exprimer ma profonde gratitude et mes sincères remerciements au professeur Mr. RAHMOUNE Mohamed, mon encadrant de stage, pour m'avoir intégré rapidement au sein de l'Equipe de Modélisation et Simulation Numérique de l'ENSA d'Oujda, et m'avoir accordé toute sa confiance; pour le temps qu'il m'a consacré tout au long de cette période, sachant répondre à toutes mes interrogations; sans oublier sa participation au cheminement de ce rapport.

Ainsi, je tiens à présenter mes reconnaissances et mes remerciements à mon encadrant académique Mme.MAJDAAicha, pour le temps consacré aux réunions qui ont rythmées les différentes étapes de mon stageet à la lecture de ce rapport. Les discussions que nous avons partagées ont permis d'orienter mon travail d'une manière pertinente. Je la remercie aussi pour sa disponibilité à encadrer ce travail à travers ses critiques et ses propositions d'amélioration.

Je saisis aussi l'occasion pour remercier Mr.BERRICH Jamal, ingénieur en informatique à l'ENSAO, pour son support technique.

Il m'est agréable de m'acquitter d'une dette de reconnaissance auprès de tous mes enseignants à l'FST de Fès qui ont si bien menés leur noble quête d'enseignement. Je les remerciedu fond de mon cœur, non seulement pour le savoir qu'ils m'ont transmis, mais aussi pour la fierté et l'ambition que leurs personnes m'inspirent.

Je remercie égalementPr.BENABBOU Rachid, Pr. ZAHI Azzedine et Pr. MAJDAAicha,les membres du jury pour m'avoir honoré en acceptant d'évaluer et de juger ce travail.

Enfin je remercie toutes les personnes qui ont contribuées de près ou de loin à la réalisation de ce travail, ainsi qu'au bon déroulement du stage, et dont les noms nefigurent pas dans ce document.

Résumé

La simulation numérique du comportement des pièces mécaniques et la simulation des procédés de leurs fabrications deviennent des outils stratégiques d'aide à la décision et à l'amélioration du cyclede vie des projets industriels. Un gain considérable en temps de productionest remarqué depuis l'intégration de la simulation numériquesous forme de prologiciels. Le temps de productiond'un véhiculequi était de deux à trois années, a été réduit àune année voire moins actuellement.

Ce projetvient d'être initié à l'équipe de recherche (EMSN : Equipe de Modélisation et Simulation Numérique) de l'ENSAO ; Mais le temps de stage alloué ne permet pas d'aller au bout de toutes les ambitions de l'équipe. Néanmoins, un plan et un planning ontété établis avec le responsable du projetafin de répondre au cahier des charges et respecterles délais prévus.

Dans cette perspective, nous nous intéressons d'une manière globale, à la simulation numérique par les éléments finis qui nous permet d'introduire la physique à la géométrie des pièces pour mieux cerner les problèmes de service ou de production.

Cet intérêt est concrétisé dans ce projet, par notre contribution à la réalisation d'uneapplicationopérationnelle; de l'interface graphiquepour la réalisationde lagéométrie (conception de pièce), en passant par le solveur poursimuler le mouvement et/ou le comportement mécanique ainsi que le discrétiseur pour générer le maillage d'une surface, à la visualisation des résultats du calcul.

Nous traitons en particulier à titre applicatif et de manière itérative : lasimulation d'un corps en chute libre, ce cas représente le test pilote pour la mise en placed'une architecturecaractérisée par son évolutivité, son indépendance ainsi que son extensibilité ; le mouvement cinématique,et, l'étude de crashd'un véhicule contre un obstacle afin de montrer le potentiel de l'application.

Mots clés : simulation numérique - géométrie - dessin- graphique - discrétisation - maillage - cinématique - chute libre - distance de sécurité- crash

Table des matières

Liste des figures

Liste des tableaux

Glossaire

C	
CAO	Conception Assistée par Ordinateur
D	
DOM	Document Object Model
E	
ENSAO	Ecole Nationale des Sciences Appliquées - OUJDA
EMSN	Equipe de Modélisation et Simulation Numérique
G	
GUI	Graphical User Interface
H	
HTTP	Hypertext Transfer Protocol
HTML	HypertextMarkupLanguage
I	
IHM	Interface Homme Machine
M	
MVC	ModèleVueContrôleur
R	
RSS	Rich Site Summary
S	
SGML	Standard GeneralizedMarkupLanguage
SQL	StructuredQueryLanguage
T	
2TUP	TwoTracksUnifiedProcess
X	
XML	eXtensibleMarkupLanguage
XSLT	Extensible StylesheetLanguage Transformations
XHTML	eXtensible HyperText MarkupLanguage

Introduction

Dans leur quête d'une meilleure satisfaction de leurs clients, les grandes sociétés s'orientent de plus en plus vers les tests numériques dont le but d'améliorer la qualité de leurs produits. La simulation numérique est un choix stratégique opté par les entreprises pour accompagner leurs produits pendant toutes les étapes de leurs cycles de vie. Dans une chaîne de valeurs ajoutées, s'intègre des prologiciels qui les aident nonseulement dans la conception, mais aussi dans le procédé de fabrication sous forme d'itérations d'amélioration jusqu'à l'obtention d'un meilleur produit de point de vue qualité et coût. L'intégration de la simulation numérique dans les tests de validation réduit considérablement les risques de production, et les quantités de rejet.

En effet, le cycle de vie et le processus de validation du produit lui-même, passe par plusieurs étapes qui font appel à des outils qui nécessitent un temps et un coût considérables. Le génie de l'industrie ne cesse de créer de nouveaux produits qui demandent d'adopter de nouveaux outils logiciels pour assurer leur validation. C'est dans cette tendance que les laboratoires et équipes de recherche universitaires ne cessentde fournirent un soutien et une base d'innovation dans le domaine des matériaux et outils numériques,en mettant leurs compétences scientifiques et informatiques au service des entreprises.

C'est dans ce cadre, que j'ai effectuémon projet de fin d'études à l'Ecole National de Sciences Appliquées d'Oujda (ENSAO), au sein de l'équipe de recherche, Equipe de Modélisation et Simulation Numérique (EMSN). Il s'intègredans le programme de la licence ST option génie informatique, préparé à la Faculté des Sciences et Techniques de Fès.

Ce projet vise la simulation numérique et l'étude du crash d'un véhicule face à un obstacle, ainsi que le calcul de la distance de sécurité recommandée à un conducteur qui roule avec une vitesse donnée. Pour arriver à ce stade, nous débutons par la simulation d'un objet en chute libre, afin de s'initier avec l'environnement, les outils et langages de travail, et précisément, pour la mise en place de l'architecture adoptée dans toute l'application.

Chapitre I

Contexte générale du projet

Introduction

Au début de ce rapport, nous allons présenter dans ce premier chapitre, l'organisme d'accueil au sein duquel s'est déroulé le stage, par la suite nous décortiquons la problématique et le cadre du projet, aussi nous montrons l'objectif de cette étude et en fin le plan de structuration de ce document.

1. Organisme d'accueil

1.1. Présentation de l'ENSAO

L'ENSAO, première grande école d'ingénieurs d'état dans la région de l'oriental a été créée par le décret ministériel N° 2-99-1007 du 1er rajeb 1422 (19 septembre 2001).

L'établissement est localisé sur le campus universitaire. Il s'inscrit dans le cadre de la diversification des formations dispensées au sein de l'Université et de la dynamisation de l'environnement socio-économique et industriel régional et national.

Mission :

La vocation de l'ENSAO est de former des ingénieurs d'état, hautement qualifiés et de faire accroître leur nombre dans les spécialités les plus demandées.

La mission de l'école est :

- La formation d'Ingénieurs d'état hautement qualifiés,
- La recherche scientifique et technologique,
- La recherche et développement R&D,
- La formation continue.

L'ENSAO a vocation pour tout ce qui concerne l'enseignement supérieur, la recherche scientifique et technique et la formation initiale et continue des ingénieurs et des cadres, notamment dans les domaines suivants:

- Génie Informatique;
- Génie Electrique;
- Génie Industriel;
- Génie Télécommunications et Réseaux;
- Génie Mécanique et Productique;
- Génie Civil.

Elle assure la préparation et la délivrance du diplôme d'ingénieur d'état;

1.2. Les études

La formation d'ingénieur se déroule en deux phases:

- Un cycle préparatoire d'une durée de deux années (ou quatre semestres) qui constitue le tronc commun;
- Un cycle ingénieur de trois ans (ou six semestres) dans de nombreux domaines de spécialités.

Durant ce cursus, l'accent est mis sur les enseignements scientifiques et techniques, les sciences humaines, la gestion et aussi sur la connaissance de l'entreprise. Les travaux pratiques prennent une part très importante dans la scolarité. La formation est complétée alors par des projets et des stages en entreprise.

Le cycle préparatoire :

Objectif

L'objectif du cycle préparatoire est de donner aux élèves une solide formation de base scientifique, technique, ainsi qu'en communication orale et écrite. A cela s'ajoute l'élaboration de mini-projets et stages. Le caractère généraliste de la formation du cycle préparatoire est indispensable pour donner, à tout ingénieur, les connaissances et les compétences qui lui permettent, quelle que soit sa spécialité, de se réorienter en cours de carrière. Le rôle de ce cycle est aussi d'aider l'élève à acquérir les méthodes et les techniques de travail qui lui seront indispensables pour la suite de ses études et dans sa carrière professionnelle.

Le cycle ingénieur :

Objectif

L'objectif de ce cycle est de donner au futur ingénieur une solide formation dans le domaine de la spécialité choisie ainsi qu'en enseignement de langues, de communication orale et écrite, de connaissance de l'entreprise et aussi de préparer l'élève ingénieur à son futur métier par des stages et des projets.

Formation et organisation pédagogique :

A l'instar du cycle préparatoire, la formation dans le cycle ingénieur est organisée au sein du département Cycle Ingénieur qui regroupe les filières de spécialités. L'organisation administrative de ce département est similaire à celle du département du cycle préparatoire. Le cursus est organisé sur trois années réparties en six semestres dans l'une des filières de spécialité suivantes:

- Génie Informatique;
- Génie Télécommunications et Réseaux;
- Génie Electrique;
- Génie Industriel;
- Génie Electronique et informatique industrielle;
- Génie Civil.

A noter que d'autres spécialités peuvent être créées (par arrêté de l'autorité gouvernementale chargée de l'enseignement supérieur).

Le mode de gestion pédagogique est similaire à celui du cycle préparatoire.

La présence à toutes les formes d'enseignement est obligatoire. La formation est complétée par des stages en entreprises ou dans des laboratoires. Les stages préparent l'élève ingénieur à sa future profession par l'acquisition d'expériences pratiques. Il est donc capital qu'ils correspondent à une situation réelle de travail et qu'ils s'inscrivent à l'intérieur des activités normales de l'entreprise. Le nombre, la durée et l'emplacement de ces stages ainsi que le mode d'évaluation seront étudiés par les différents intervenants (direction, départements etc.). Le sixième semestre est consacré entièrement à un stage en entreprise ou dans des laboratoires de recherche dont le projet donne lieu à la rédaction d'un mémoire et à une soutenance devant un jury d'industriels et d'universitaires. L'école œuvrera à établir des liens structurels avec des établissements d'autres pays travaillant dans des domaines connexes, de façon à organiser des échanges d'étudiants pour y effectuer des stages, des projets voire même des études pendant une année complète à l'étranger.

1.3. Equipes de recherches

L'ENSA d'Oujda intègre de nombreuses équipes de recherches scientifiques spécialisées dans différents domaines :

Equipe	Axes de recherche
Modélisation et simulation numérique	▪ Modulation de la turbulence dans les écoulements à surfaces libres ▪ Matériaux composites et intelligents
Mécanique et calcul scientifique	▪ Lois de comportement des matériaux ▪ Matériaux, structures et endommagement
Systèmes et dispositifs micro et nanoélectronique	▪ Modélisation de capteurs chimiques destinés à l'analyse de l'eau ▪ Conception de composants et circuits intégrés utilisés dans la haute fréquence et optoélectronique
Electronique et télécommunication	▪ Conception et réalisation de capteurs thermoélectriques de détection des rayonnements infrarouges ▪ Contrôle industriel par vision artificielle
Systèmes de production	▪ Maintenance, sécurité des systèmes industriels et environnement ▪ Conception et contrôle des produits
Mathématique, informatique et mécanique	▪ Cryptographie et sécurité informatique ▪ Script de la langue arabe

Tableau 1 : Equipes de recherches scientifiques

L'Equipe de Modélisation et Simulation Numérique a été créée en 2011 avantla visite royale à l'ENSAO. Elle s'oriente vers la modélisation mathématique et la simulation numérique des phénomènes physiques, en particulier les structures intelligentes, les composites et l'écoulement de fluide. Plusieurs travaux dans ses différentes disciplines ont été déjà initiés par les différents membres permanents et associés de l'équipe. La simulation numérique par des éléments finis ou volumes finis est l'axe principal de notre recherche. Le développement des applications-logiciels, en particulier les interfaces pour la conception assistée par ordinateur et la réalisation du code de calcul, ainsi que les interfaces de visualisation, présentent un intérêt particulier dans notre équipe ; d'où le lancement d'un grand projet de réalisation d'une

interface qui va rassembler les différents codes existants et unifier les interfaces utilisées en se basant sur une architecture MVC qui va permettre, l'indépendance des applications en forçant une communication transversale avec les différents produits.

2. Cadre du projet et problématique :

Dans la plupart des secteurs de production industriels, le test matériel et fonctionnel occupe une partie très importante dans le processus de production, dont le but est d'étudier le bon fonctionnement du produit et aussi de s'assurer de la sécurité de son utilisation avant de procéder à sa commercialisation.

Notre projet fait partie des logiciels d'étude dans le domaine d'industrie de l'automobile. C'est une application scientifique basée sur des notions physiques, en particulier la simulation numérique du comportement d'un véhicule en mouvement après un freinage, ainsi que les dégâts provoqués après un éventuel choc frontal avec un obstacle.

Ce projet entre dans la catégorie des projets de recherche scientifique, effectués au sein des équipes de recherche universitaire (EMSN). Ce type de projet demande un temps considérable, des bases scientifiques solides et d'importantes connaissances techniques en termes de langages de programmations. La réalisation d'un projet de recherche est souvent faite par la collaboration de toute une équipe et un groupe de travail où chacun de ses membres est spécialisé dans des taches bien précises. Dans ce contexte, l'organisation et la configuration du projet, ont une importance primordiale, dont le but est d'aboutir à un résultat intéressant tout en respectant les délais prévus.

Mais, dans le cas de mon projet, n'ayant pas la possibilité d'avoir une équipe, m'a exigé de réaliser l'ensemble des taches du projet séparément, et procéder par la suite à leur intégration afin d'arriver à mettre en place une application opérationnelle et intéressante.

3. Objectif de l'étude

Un grand nombre de logiciels de simulation numérique soit en CAO, soit en calcul de structure, sontdisponibles dans le marché.Chaque logiciel, apprécié pour certaines de sescaractéristiques,répond à un besoin particulier.Par exemple, si on prendCatia qui est le produit utilisé par excellence dans le domaine de la CAO, est apprécié pour sa solution complète. Il n'empêche que d'autres logiciels, comme SolidWorks,continuentd'exister sur le marché et d'améliorer leurs applications pour atteindre un grand public.Dans la simulation de crash et emboutissage, on trouve Pam-Crash qui présente un grandpotentiel par sa solution multi-physique. Dans ce travail, nous proposons une application qui s'inspire de la solution Pam-Crash, mais en cherchant à proposer des solutions complémentaires et non concurrentiels.Nous nous intéressons donc à un objectif bien précis et orienté vers l'industrie automobile. Pour cela nous étudions le casd'un véhicule en mouvement, et nous déterminons sa distance de sécurité, afin d'estimer rapidement les dégâts humains causés par un crash contre un obstacle, en évitant un calcul lourd et coûteux par rapport à un calcul de crash complet fait par Pam-Crash ou un autre logiciel équivalent.

L'application à réaliser fait l'objet de deux parties :

- Une application GUI, réalisée avec le framework orienté objet Qt 4.7 : Elle est composée d'un outil de dessin pour la partie géométrie, un générateur de maillage pour la partie discrétisation, un ensemble de formulaires pour la partie cinématique la partie devançant le lancement de la simulation, et, finalement, un visualisateur de résultat. Toutes ces fonctionnalités sont intégrées dans une seule interface simple d'utilisation et intuitive.

- Un solveur développé en C++, son rôle se focalise sur la génération du résultat de chaque simulation selon le traitement adéquat, en se basant sur les données d'entrées fournies par des fichiers XML (input) créés par l'interface lors du lancement du calcul. C'est à base de ces fichiers que d'autres sont créés par le solveur contenant le résultat à visualiser (output). Ce programme de traitement est lancé en arrière-plan à l'aide d'un processus afin de ne pas interrompre la continuité du travail de l'utilisateur.

L'adoption de cette architecture qui sépare l'interface, du traitement et des données, est un choix stratégique et crucial pour la mise en œuvre d'une application qui s'exécute dans un temps relativement court et demande des ressources systèmes réduites.Cette architecture est basée sur l'MVC, le patron d'architecture et la méthode de conception qui organise l'IHM d'une application logiciel.

4. Plan du document

Le premier chapitre de ce travail fait l'objet d'une introduction générale sur le contexte du projet à réaliser, une présentation de l'organisme d'accueil, l'Ecole National de Sciences Appliqués d'Oujda, la problématique et le cadre du projet, l'objectif de cette étude et finalement une description du plan de ce document.

Quant au deuxième chapitre, il sera consacré à l'étude générale du projet en commençant par une étude préliminaire où nous allons spécifier les différents besoins fonctionnels et techniques sollicités par le projet. Par la suite nous effectuerons un rappel sur quelques notions en physique utilisées dans ce projet,et,finalement, captureret décrire de façon détaillée lesbesoins principaux exprimés dans ce projet, et réaliser enfin un diagramme de cas d'utilisation regroupant l'ensemble des cas d'utilisation traités par le projet.

Dans le troisième chapitre, on trouve deux parties :Primo, la partie analyse oùnous identifions les classes candidates des cas d'utilisation capturés dans l'étape étude, après nous développons le modèle statique et, par la suite le diagramme des composants qui va nous permettre de montrer les différentes parties de notre application. Secondo, la partie conception, qui permettra de mettre en relief l'architecture mise en place dans l'application, ainsi que le diagramme de classes final et terminer par le modèle de données.

La quatrième et dernière partie sera consacrée à la réalisation et la mise en œuvredu projet. Elle comporte trois axes : premièrement, une description des différents outils et langages utilisés tout au long de la réalisation du projet, deuxièmement nous allons définir la méthode MVC et le gain apporté par son utilisation dans notre projet, et, troisièmement, les différents composants et étapes

de réalisation de cette application. Ces derniers occuperont bien évidemment une partie très importante dans ce rapport afin de mettre en évidence l'ensemble des difficultés techniques rencontrées durant le cycle de réalisation de ce projet.

Conclusion :

Dans ce premier chapitre, nous avons exposés le contexte général de ce projet, l'organisme d'accueil, la problématique, le cadre général, l'objectif et le plan du document. Par la suite, dans le deuxième chapitre, nous étudions l'ensemble des besoins et attentes de la réalisation de ce projet.

Chapitre II

Etude générale du projet

Introduction

La phase d'étude et analyse des besoinsd'un projet est l'étape avant-réalisation, elle permet d'établir un cahier des charges clair et explicatif, qui assure l'entente entre le client et le développeur, afin de mener le projet à ses fins et ses besoins souhaités.

Dans ce chapitre nous allons étudier les différents besoins et attentes de ce projet selon ses deux aspects, fonctionnel et technique.

1. Etude préliminaire

L'étude préliminaire est la toute première étape du processus de développement en Y (2TUP). Elle consiste à effectuer un premier repérage des besoins fonctionnels et opérationnels, en utilisant principalement le texte, ou des diagrammes très simples. Elle prépare les activités plus formelles de capture des besoins fonctionnels et de capture des besoins techniques.

1.1. Recueil des besoins fonctionnels

Suite aux différentes réunions avec mon encadrant de projet,dont leurs but est de cerner et mieux identifier les besoins et attentes de l'application,nous nous sommes fixés sur un ensemble de besoins détaillés dans le cahier des charges préliminaires suivant :

Simulation d'une chute libre :

La chute libre est une notion physique basique (*voir: 2 - Rappel de quelques notions en physique*), elle représente un corps en mouvement sous la pesanteur, sans frottement.

Cette phase représente la première itération de ce projet, elle a été introduite pour mettre en place l'architecture à adopter dans toute l'application, pour montrer les différents avantages et inconvénients de cette architecture, et, aussi, pour faire face aux difficultés techniquessuite à sonutilisation.

Nous simulons un corps en chute libre, représenté par unpoint-centre de gravité, ayant un poidsdonné. Dans l'absence de frottement, ce corps doit rebondir sans arrêt après le contact avec un obstacle.

Calcul de la distance de sécurité :

La distance de sécurité est un facteur très important dans le secteur du transport routier. Elle représente la distance minimale recommandée pour un conducteur à respecter entre son véhicule et celui qui le précède.

La simulation d'une voiture en mouvement, qui a un poids connu et qui roule à une vitesse donnée, nous permet de calculer la distance minimale que l'automobiliste doit respecter pour être en sécurité. Cette simulation est composée de deux parties : le mouvement avant freinage, où la voiture

roule à une vitesse constante, et le mouvement après freinage, où la vitesse de la voiture décélère jusqu'à son arrêt complet.

Etude et simulation du crash d'un véhicule :

Ce besoin représente le but principal de cette application. Il se concentre sur la modélisation d'un véhiculeobjectivé par une barre qui vient heurter un obstacle.

Le résultat de cette étude sera sous forme d'un maillage coloré en se basant sur les parties endommagées du véhicule. Cela nous permettra de déterminer la gravité de l'accident et par conséquence l'état du conducteur.

Tous d'abord, la simulation passe par la réalisation de la barre qui représente le véhicule. Cette barre est illustrée par une surface, laquelle est dessinée avec l'outil-géométrie intégré dans l'application. Par la suite vient la notion de discrétisation, elle sert à transformer la surface en mailles afin d'étudier l'impact de l'accident sur chaque partie du véhicule.

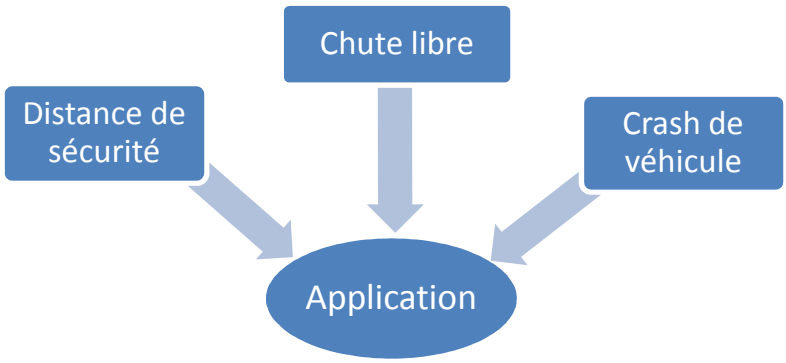


Figure 1 : Les besoins principaux de l'application

1.2. Choix techniques

Afin de mener ce projet à son but, et pour mieux étudier, analyser, et, réaliser ses besoins fonctionnels et techniques, il a fallu, avant tout, choisir le bon processus et la bonne méthode de développement logiciel. Pour cela, nousavons opté pour le processus de développement logiciel 2TUPqui implémente le Processus Unifié.Le 2TUP propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels.

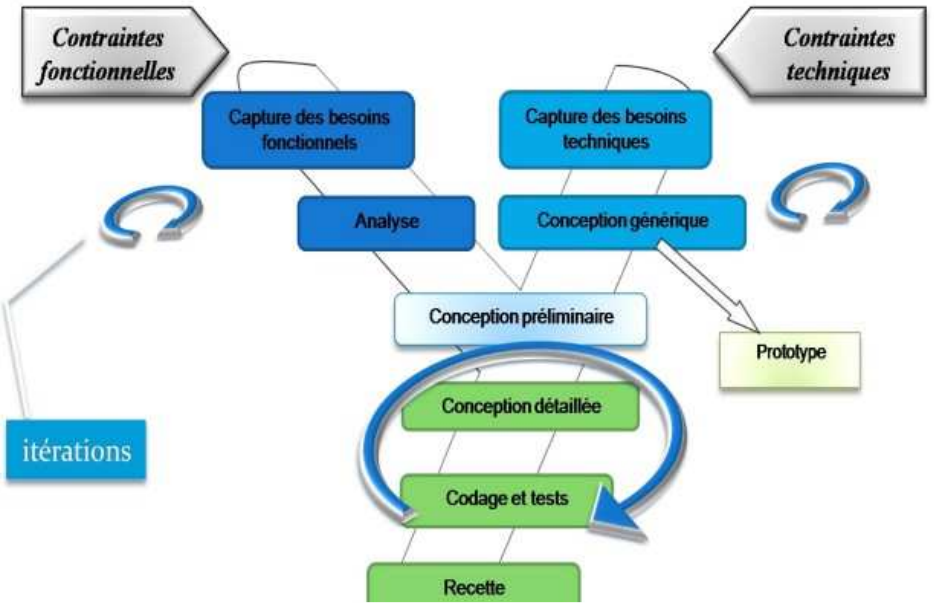


Figure 2 :2TUP, Le processus de développement en Y

Le point fort qui nous acontraint à choisir ce processusest la possibilité de pouvoir l'utiliser dans chaque itération évolutive du projet.

Pour réaliser cette application, des outils et des langages de programmation ont été choisis par le responsable du projet afin d'unifier les choix techniques :

- Le langage de modélisation UML
- L'architecture MVC
- Le langage de Programmation Orienté Objet C++
- Le framework graphiqueQt
- Le langage de balisage XML
- L'IDE Visual C++ 2008
- L'outil Qt Creator

1.3. Planification du projet

La réalisation de ce projet s'est déroulée selon un planning bien précis dont le but est de respecter les délais prédéfinis pour le stage.Ce planning consiste à spécifier et prévoir les différentes phases et étapes à suivre tout au long du cycle de développement du projet.

Pour schématiser le planning du projet, le choix a été porté sur l'outil *GanttProject* pour sa réputation parmi les logiciels libres de gestion de projet. Il permet la planification d'un projet à travers la réalisation d'un diagramme de Gantt.

Ci-dessous, le planningde réalisation du projet :

Nom	Date de début	Date de fin
Projet de fin d'études : Application physia	16/04/12	04/06/12
Première réunion	16/04/12	16/04/12
Etude préliminaire	17/04/12	22/04/12
Etablissement du cahier des charges	17/04/12	21/04/12
Confirmation et validation des besoins	22/04/12	22/04/12
Phase 1 : Chute libre	23/04/12	03/05/12
Analyse et conception	23/04/12	24/04/12
Réalisation de l'interface graphique et du solveur (version 1.0)	25/04/12	29/04/12
Mise en place de l'architecture	30/04/12	01/05/12
Réalisation du visualisateur	01/05/12	02/05/12
Validation de l'avancement	03/05/12	03/05/12
Phase 2 : Distance de sécurité	03/05/12	14/05/12
Analyse et conception	03/05/12	05/05/12
Réalisation et codage	06/05/12	08/05/12
Intégration des changements à l'interface et au solveur (version 2.0)	09/05/12	10/05/12
Mise à jour du visualisateur	10/05/12	11/05/12
Test et Validation de l'avancement	11/05/12	14/05/12
Phase 3 : Crash de véhicule	14/05/12	04/06/12
Analyse et conception	14/05/12	17/05/12
Elaboration d'un algorithme de maillage	18/05/12	18/05/12
Réalisation du discrétiseur	19/05/12	20/05/12
Integration du discrétiseur dans l'architecture global de l'application (version 3.0)	21/05/12	23/05/12
Mise à jour et finalisation du visualisateur		
Mise à jour et finalisation de l'outil de géométrie		
Mise à jour et finalisation de l'outil de cinématique		
Validation de l'avancement		
Intégration du solveur crash	04/06/12	04/06/12
Test d'intégration	08/06/12	08/06/12
Finalisation du rapport & Préparation de la soutenance	09/06/12	15/06/12

Figure 3 : Le planning du projet

1.4. Recueildes besoins opérationnels

Le développement des différents composants de l'application doit être optimisé, en tenant compte de l'efficacité del'architectureetde la facilité deson extension, ainsi que de l'ergonomie et du temps de calcul.

L'application doit intégrer la notion de projet ; la création d'un nouveau projet, saréouverture et la réutilisabilité de ses résultats.

Le projet est concrétisé par un répertoire qui porte le nom du projet, et qui contient un fichier XML précisant des informations sur le projet.

L'application est à l'état embryonnaire, dont l'accès n'est pas réglementé. Par contre ce qui peut être une solution de gestion de licence,est de suivre les techniques de gestion de licences proposées par le FLEXLM.

L'application doit être portable sur toutes les plates-formes (Unix, Linux, Windows…).Un travail concernant la compilation des exécutables sur les différentes plates-formes doit être préparé.

1.5. Identification des acteurs et messages

Acteurs :

L'application est susceptible d'être utilisé par un grand nombre d'utilisateurs, sans contraintesspécifiques de sécurité. Elle est spécialement conçue pour des scientifiques et des étudiants, pour des fins d'étude et de recherche.

Les acteurs prévus pour l'utilisation de l'application sont :

- Les scientifiques
- Les étudiants
- Les chercheurs
- Les industriels
- Etc.

Messages :

Des messages sont reçus et émis par l'application afin de traiter un cas donné. Un message peut prendre différentes formes selon son contexte.

Les messages émis par le système :

- Création de formes géométriques.
- Création de maillages.
- La visualisation du mouvement d'une balle en chute libre.
- La visualisation du déplacement d'un véhicule avant et après freinage.
- La simulation du crash d'un véhicule avec un obstacle.
- Création du répertoire et fichiers du projet.

Les messages reçus par le système :

- La création d'un nouveau projet.
- L'ouverture d'un projet.
- Modification des paramètres de l'application.
- Zoomer et dé-zoomer la zone de dessin.
- Le dessin et la modification d'un point.
- L'insertion d'une ligne.
- La réalisation d'une surface.
- L'ajout d'une voiture.
- Effacer la zone de dessin.

2. Rappel de quelques notions en physique

2.1. Chute libre

Nous rappelons que l'étude ne vise pas à faire la chute libre en soit. Mais c'est un exemple pilote qui permet de mettre en évidence les difficultés techniques de l'architectureet permet aussi de bâtir un concept d'une architecture efficace qui recueil toutes les différentes applications actuelles et futures.

La phase de descente du corps pendant la chute libre est définie par l'équation suivante :

$$x = \frac{1}{2} g * t^2 + x_0 \ ;$$

x est la position du corps en chaque instant.

g est la pesanteur (vaut 10).

$x_0$est la position initiale.

t est le temps de la chute.

Après le rebond, le corps remonte avec une vitesse donnée suite à la réaction du sol. Sans tenir compte du frottement d'air, l'équation du mouvement est :

$$x = -\frac{1}{2} g * t^2 + v_0 * t + x_0 \ ;$$

$v_0$est la vitesse initiale.

t est le temps de la montée dans l'aire.

La position d'arrêt $x_a$du corps dansl'aire est la suivante :

$$x_a = \frac{v_0^2}{2 * g}$$

En absence du frottement, nous assistons à un rebondissement sans arrêt du corps.

2.2. Distance d'arrêt

Cet exemple est réalisé pour mettre en évidence l'usage utile d'un véhicule par tout conducteur averti. Il permet de déterminer la distance de sécurité recommandée pour un automobiliste qui rouleà une vitesse donnée. Nous considérons que l'automobiliste freine après une vitesse de croisière donnée *V* (vitesseuniforme). Ce freinage, représenté par une décélération, dépend proportionnellement du frottement, et inversement proportionnelle, de la vitesseet du poids du véhicule.

$$\gamma = \gamma_{fr} + \gamma_v + \gamma_p = \left[k_1 * fr + \frac{k_2}{V} + \frac{k_3}{mg} \right]$$

$\gamma_{fr} = k_1 * fr$ *est la décélération provoquée par le frottement fr.*

$\gamma_v = \frac{k_2}{V}$ *est la décélération due à la vitesse V.*

$\gamma_p = \frac{k_3}{mg}$ *est la décélération due au poids mg.*

La distance de sécurité est par définition la somme de la distance de réaction x_a et la distance de freinage x_f.

La distance de réaction est liée au temps de réaction de l'automobiliste avant le freinage est estiméeàune à deux secondes.

$$x_a = V * t \quad avec \quad 1s < t < 2s$$

La distance de freinage est la distance qui correspond d'un passage de la vitesse initiale *V* à une vitesse nulle.

$$x_f = \frac{V^2}{2 * \gamma}$$

La position du véhicule avant l'arrêt total (décélération) est :

$$x = -\frac{1}{2}\gamma * t^2 + V * t + x_a$$

2.3. La simulation numérique du crash d'un véhicule

L'étude de crash se base sur un comportement non linéaire en déformation et déplacement, plus une gestion non linéaire du contact du véhicule contre l'obstacle. Le problème a été difficile de le prendre en charge dans un stage de deux mois. Malgré ces contraintes, nous avons cherché à faire une modélisation simplifiée où le comportement est linéaire, et où le contact est géré d'une manière simplifiée. Ce qui nous ramène à l'étude d'un mouvement purement élastique d'une barre qui représente vulgairement le véhicule. Néanmoins, nous considérons qu'une déformation de 20 cm représente un danger réel à l'automobiliste. Cette distance correspond à peu près à la distance entre le volant et le conducteur.

2.3.1. Notions sur la mécanique des milieux continues

L'équation qui gouverne l'état de déformation d'un corps est défini par :

$$\begin{cases} div\sigma = f \quad / \quad \Omega \\ \sigma.n = F \quad / \quad \partial_1\Omega \\ u = u_d \quad / \quad \partial_2\Omega \end{cases}$$

σ *est le tenseur de contrainte.*

f est la densité de force volumique.

F est la densité de force appliquée au bord du domaine Ω.

Le tenseur de contrainte σ est lié au tenseur de la déformation ε par :

$$\{\sigma\} = [E]\{\varepsilon\}$$

E est la matrice qui caractérisel'élasticité du matériau du corps Ω.

La déformation liée au déplacement *u* des différents points du corps suiteà leurs déformations élastiques est définie par :

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

L'équation des travaux virtuels correspond à :

$$\int_\Omega \sigma(u).\varepsilon(v) = \int_\Omega f.vf + \int_{\partial_1\Omega} F.v$$

2.3.2. Maillage et discrétisation de la géométrie

Un maillage est la discrétisation spatiale d'un milieu continu, ou aussi, une modélisation géométrique d'un domaine par des éléments proportionnés finis et bien définis. L'objet d'un maillage est de procéder à une simplification d'un système par un modèle représentant ce système et, éventuellement, son environnement (le milieu), dans l'optique de simulations de calculs ou de représentations graphiques.

Un maillage est défini par :

- son repère ;
- les points le constituant, caractérisés par leurs coordonnées ;
- les cellules, constituant des polytopes reliant *n* de ces points ;

et peut-être caractérisé notamment par :

- sa dimension : typiquement 2D ou 3D ;
- son volume (dimension totale couverte) ;
- sa finesse : surface ou volume moyen des cellules composant le maillage ;
- la géométrie des cellules : triangles, polygones, carrés, etc en 2D, polyèdres, parallélépipèdes, cubes en 3D.

3. Capture des besoins

3.1. Déterminer les cas d'utilisation

Un cas d'utilisation représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système. Il exprime les interactions acteurs/système et apporte une valeur ajoutée « notable » à l'acteur concerné.

L'identification des cas d'utilisation, nous donne un aperçu des fonctionnalités futures que doit implémenter le système. Cependant, il nous faut plusieurs itérations pour arriver à constituer des cas d'utilisation complets. D'autres cas d'utilisation vont apparaître au fur et à mesure de leur description, et de l'avancement dans le « recueil des besoins fonctionnels ». Pour constituer les cas d'utilisation, il faut considérer l'intention fonctionnelle de l'acteur par rapport au système dans le cadre de l'émission ou de la réception de chaque message. En regroupant les intentions fonctionnelles en unités cohérentes, on obtient les cas d'utilisationsdétaillés dans le tableau suivant :

Cas d'utilisation	Messages émis/reçus
Dessiner des formes géométriques	Emet : ▪ Dessiner un point ▪ Dessiner une ligne ▪ Dessiner une surface Reçoit : ▪ Les coordonnées de points ▪ Couleur, diamètre, épaisseur, etc.
Générer le maillage d'une surface	Emet : ▪ Générer et afficher le maillage d'une surface ▪ Enregistrer le maillage Reçoit : ▪ Coordonnées des quatre points de la surface ▪ Le pas en X et Y du discrétisation ▪ Couleur du maillage
Visualiser les simulations	Emet : ▪ La visualisation des mouvements et déplacements de la simulation Reçoit : ▪ Fichier d'entrée des coordonnées de déplacements des points du mouvement
Simuler une chute libre	Emet : ▪ Créer un point (balle) ▪ Ajouter un sol ▪ Visualiser le mouvement de chute Reçoit : ▪ Les coordonnées du point ▪ L'emplacement du sol
Calculer la distance de sécurité	Emet : ▪ Ajouter une voiture ▪ Visualiser le mouvement de déplacement et décélération de la voiture ▪ Résultat de l'étude : la distance de sécurité recommandée Reçoit : ▪ La vitesse ▪ Le poids ▪ L'emplacement du freinage
Etudier le crash d'un véhicule	Emet : ▪ Créer une surface ▪ Discrétiser une surface ▪ Corriger le maillage ▪ Visualiser le crash Reçoit : ▪ Les coordonnées des quatre points de la surface ▪ La vitesse ▪ Le poids ▪ L'emplacement de l'obstacle

Tableau 2 : Détails des cas d'utilisations

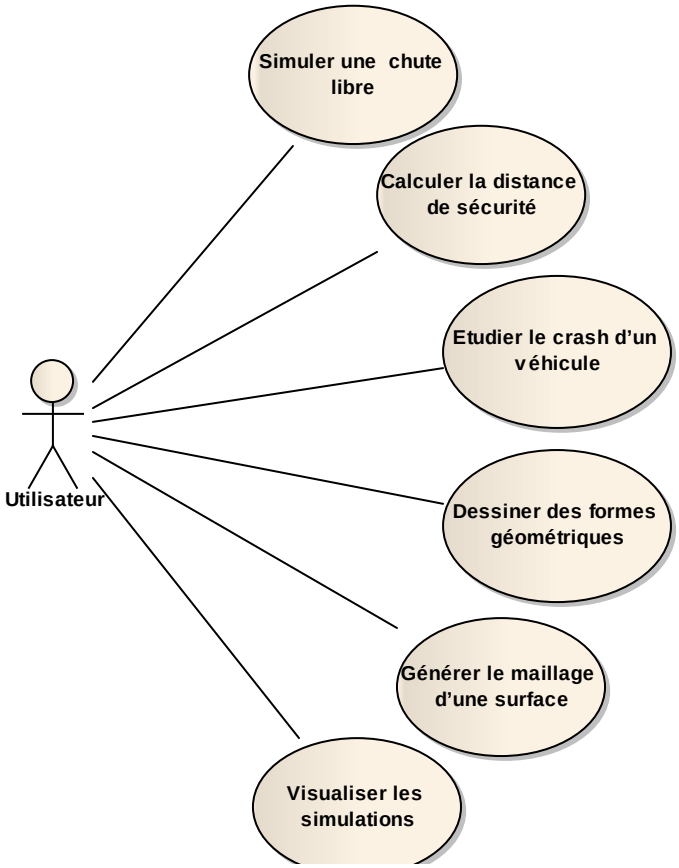


Figure 4 : Diagramme des cas d'utilisations

3.2. Description préliminaire des cas d'utilisations

Voici une description préliminaire des cas d'utilisations énumérés précédemment :

Dessiner des formes géométriques :

*Intention :*Préparer la base géométrique pour la cinématique des mouvements à simuler.

Actions : Dessiner, modifier, déplacer, sélectionner et, supprimer des points, des lignes et des surfaces ; effacer la zone de dessin ; zoomer la zone de dessin.

Générer le maillage d'une surface :

*Intention :*Générer et afficher le maillage de la discrétisation d'une surface.

Actions : Introduire les informations sur le maillage, l'afficher, le modifier, le déplacer et le supprimer.

Visualiser les simulations :

*Intention :*Visualiser les simulations.

Actions : Lancer la visualisation du mouvement d'une simulation, arrêter la visualisation.

Simuler une chute libre :

*Intention :*Simuler une balle en chute libre.

Actions : Dessiner une balle, la modifier et la déplacer ;enregistrer la cinématique de la chute libre d'une balle et calculer le mouvement de la chute.

Calculer la distance de sécurité :

*Intention :*Simuler le mouvement d'un véhicule pour calculer sa distance de sécurité recommandée.

Actions : Insérer une voiture, enregistrer la cinématique du déplacement d'un véhicule, calculer la distance de sécurité pour une vitesse donnée.

Etudier le crash d'un véhicule :

*Intention :*Etudier et simuler le crash frontal d'un véhicule contre un obstacle.

*Actions :*Dessiner des points, des lignes, une surface ; discrétiser une surface, enregistrer la cinématique du crash, générer le fichier de calcul des déformations de la surface après le choc.

3.3. Description détaillée des cas d'utilisation

Nous allons maintenant décrire en détails chaque cas d'utilisation qui doit faire l'objet d'une définition, qui décrit l'intention de l'acteur lorsqu'il utilisele système et les séquences d'actions principales qu'il est susceptible d'effectuer. Ces définitions servent à fixer les idées et non pas pour spécifier un fonctionnement complet et irréversible.

Dessiner des formes géométriques

Préconditions :

- Un nouveau projet est créé.

Scénario nominal :

- Dessiner un point.
- Dessiner une ligne sur la base de deux points.
- Réaliser une surface sur la base de quatre lignes.
- Modifier, déplacer, supprimer une forme géométrique.
- Effacer la zone de dessin.

Scénarios alternatifs :

- Le projet n'existe pas.

Post conditions :

- Le dessin des formes géométrique est effectué avec succès.

Diagramme de séquence :

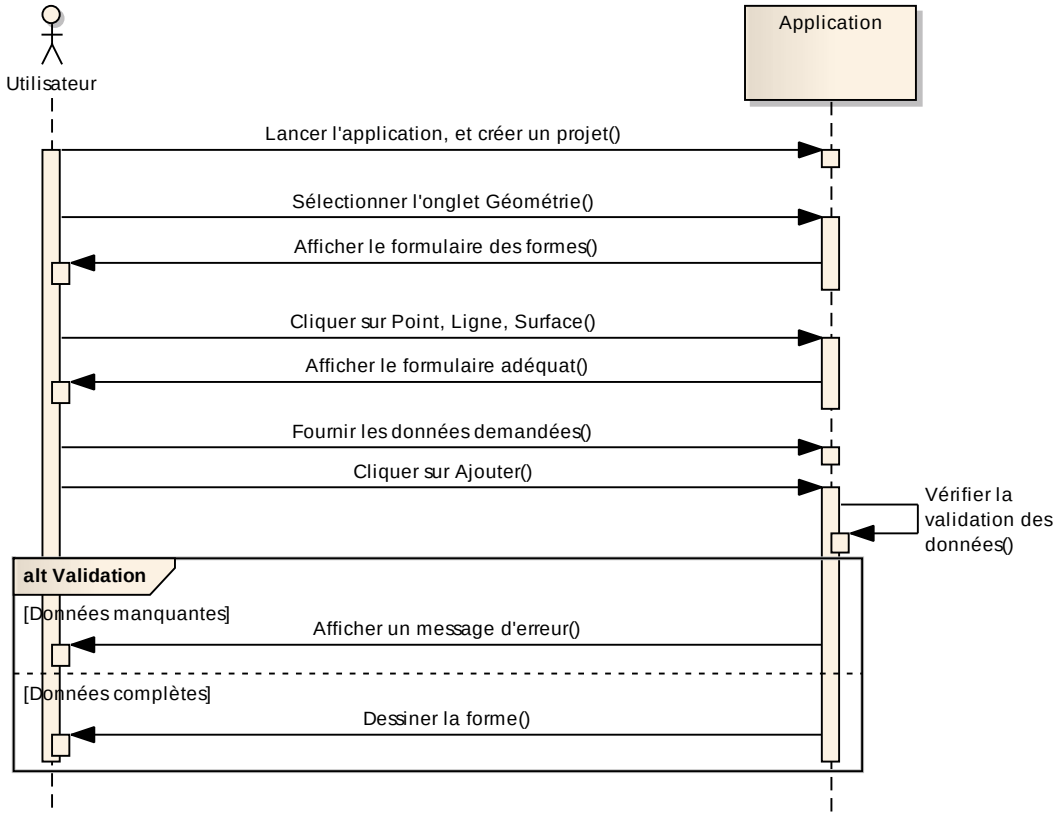


Figure 5 : Diagramme de séquence - Dessiner des formes géométriques -

Générer le maillage d'une surface

Préconditions :

- Un nouveau projet est créé.
- Au moins une surface existe sur la zone de dessin.

Scénario nominal :

- Sélectionner la surface.
- Fournir le pas de discrétisation en X et en Y.
- Choisir la couleur du maillage.
- Enregistrer et lancer la discrétisation.

Scénarios alternatifs :

- Le projet n'existe pas.
- Aucune surface n'est disponible sur la zone de dessin.

Post conditions :

- Le maillage est généré et affiché avec succès.

Diagramme de séquence :

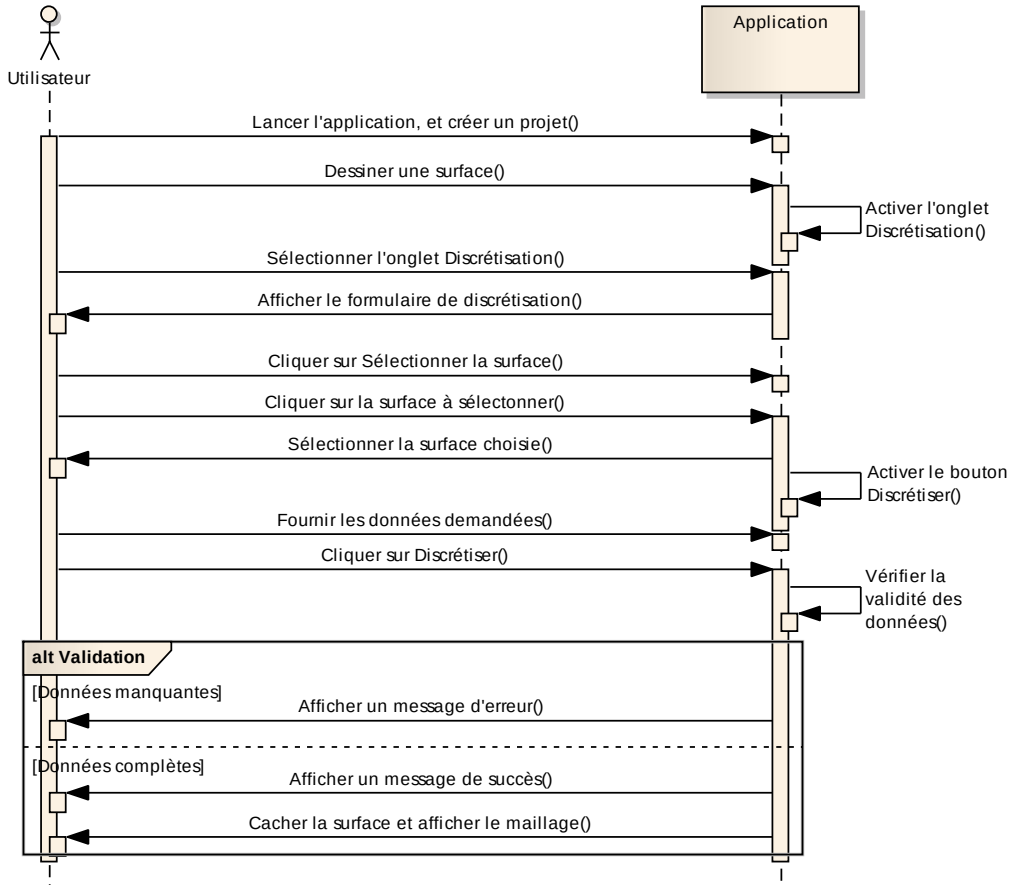


Figure 6 : Diagramme de séquence - Générer le maillage d'une surface -

Visualiser les simulations

Préconditions :

- Un nouveau projet est créé.
- Le fichier de données de sorties existe.

Scénario nominal :

- Lancer la visualisation.
- Arrêter la visualisation.

Scénarios alternatifs :

- Le projet n'existe pas.
- Le projet est vide, aucun fichier de données de sorties n'est disponible.

Post conditions :

- La visualisation du mouvement de la simulation est effectuée avec succès.

Diagramme de séquence :

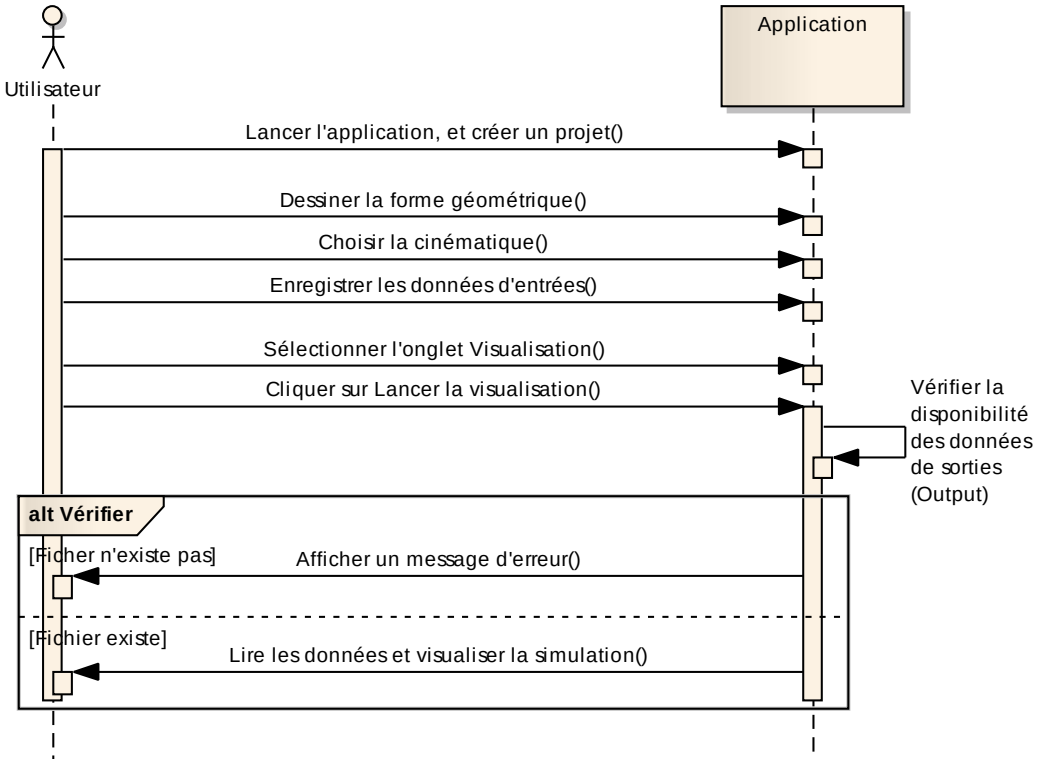


Figure 7 : Diagramme de séquence - Visualiser les simulations -

Simuler une chute libre

Préconditions :

- Un nouveau projet est créé.

Scénario nominal :

- Création du point qui représente la balle.
- Sélectionner le point de la simulation.
- Associer un sol à la balle.
- Le sol est bien au-dessous de la balle.
- Enregistrement des données d'entrées et lancement du calcul.
- Le visualisateur récupère le fichier de données de sorties.
- La visualisation est lancée avec succès.

Scénarios alternatifs :

- Aucun sol n'est trouvé au-dessous de la balle après la projection, affichage d'un message d'erreur.
- Le visualisateur ne trouve pas le fichier de données de sorties, affichage d'un message d'erreur.
- Le projet n'existe pas.

Post conditions :

- La visualisation de la simulation d'une balle en chute libre est effectuée avec succès.

Calculer la distance de sécurité

Préconditions :

- Un nouveau projet est créé.

Scénario nominal :

- Insertion de la voiture.
- Fournir la vitesse, le poids et l'emplacement du freinage.
- Fichier de données d'entrées enregistré, et calcul lancé.
- Le visualisateur récupère le fichier de données de sorties.
- La visualisation est lancée avec succès.

Scénarios alternatifs :

- Une des valeurs de la vitesse, le poids ou l'emplacement de freinage est manquant, affichage d'un message d'erreur.
- Le visualisateur ne trouve pas le fichier de données de sorties, affichage d'un message d'erreur.
- Le projet n'existe pas.

Post conditions :

- La visualisationdu déplacement de la voiture, et le calcul de sa distance de sécurité recommandée est effectuée avec succès.

Etudier le crash d'un véhicule

Préconditions :

- Un nouveau projet est créé.

Scénario nominal :

- Dessin d'une surface qui représente le véhicule en vue de haut.
- Discrétisation de la surface, le fichier de maillage est enregistré.
- Sélection de la surface et insertion de la vitesse et le poids.
- Ajout de l'obstacle.
- Fichier de données d'entrées enregistré, et calcul lancé.
- Le visualisateur récupère le fichier de données de sorties.
- La visualisation est lancée avec succès.

Scénarios alternatifs :

- Aucune surface n'est disponible.

- Le fichier de maillage n'existe pas.
- La valeurde la vitesse ou le poids n'est pas fourni, affichage d'un message d'erreur.
- Le visualisateur ne trouve pas le fichier de données de sorties, affichage d'un message d'erreur.
- Le projet n'existe pas.

Post conditions :

- Visualisation du crash de la surface (le véhicule) contre l'obstacle, et affichage de la déformation finale des mailles (les éléments) de la surface après le crash.

Conclusion :

Dans ce chapitre nous avons étudié les différents besoins de ce projet, selon une description préliminaire, et une autre détaillée, de chaque cas d'utilisation. Un rappel, enquelques lignes, de quelques notions en physique pour mieux suivre le projet. Le chapitre suivant portera sur la phase analyse et conception, où nous allons effectuer une conception détaillée du projet.

Chapitre III

Analyse et conception du projet

Introduction

Les étapes, analyse et conception, sont des étapes clés dans le processusde développement logiciel. Elles nous permettent d'ébaucherla structure objet du projet constituée de différents diagrammes de conception, de classe et de composant, qui montrent l'architecturede l'application.

1. Analyse du projet

1.1. Identification des classes candidates

Dans cette étape nous allons préparer la modélisation orientée objet en commençant par trouver les classes principales du futur modèle statique d'analyse.

Ci-dessous le diagramme des classes candidates participantes des cas d'utilisation :

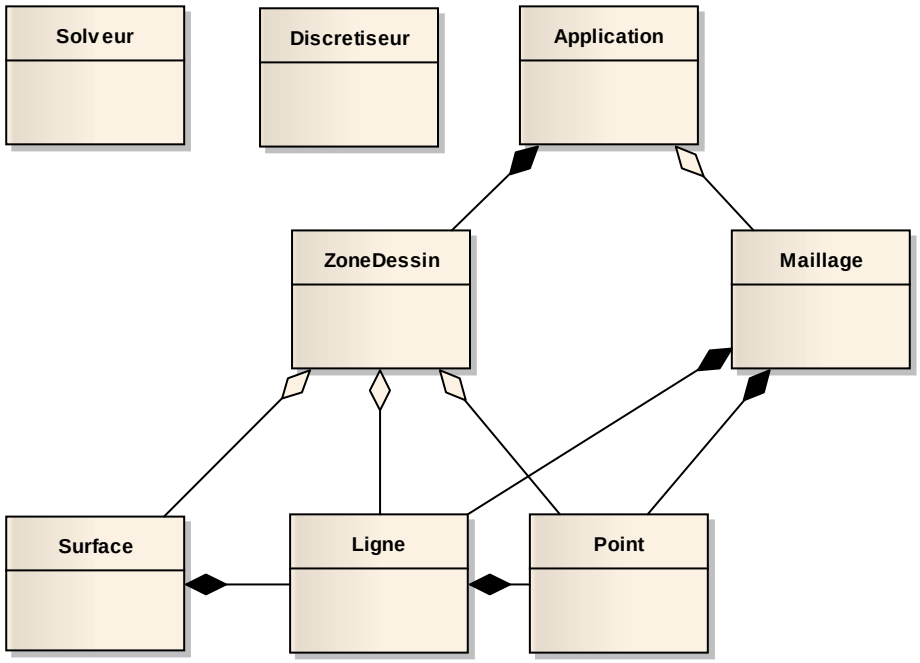


Figure 8 : Diagramme de classes préliminaires

1.2. Développement du modèle statique

Le développement du modèle statique constitue la deuxième étape d'analyse. Il propose une description de chacune des classes déduites précédemment, comme le montre le tableau ci-dessous :

Classe	Description
Application	C'est la classe qui représente l'interface graphique. *Attributs :* ▪ Maillage ▪ Nom du projet ▪ Chemin du projet ▪ Chemin du solveur ▪ Chemin du discrétiseur ▪ Scène ▪ Timer ▪ Pointeurs des composants graphiques
Maillage	Permet de générer et afficher le maillage. *Attributs :* ▪ Type ▪ Couleur ▪ Liste des nœuds ▪ Liste des éléments ▪ Liste des lignes
ZoneDessin	C'est la classe qui définit la scène de dessin des formes géométriques. *Attributs :* ▪ Mode ▪ Couleur du point ▪ Couleur de la ligne ▪ Diamètre du point ▪ Epaisseur de la ligne ▪ Nombres des points ▪ Nombres des lignes ▪ Nombres des surfaces
Point	La classe qui représente un point. *Attributs :* ▪ Liste des lignes qui sont liées au point ▪ Couleur du point ▪ Diamètre du point
Ligne	La classe qui représente une ligne. *Attributs :* ▪ Pointeur sur le point de début de la ligne ▪ Pointeur sur le point de fin de la ligne ▪ Couleur de la ligne ▪ Epaisseur de la ligne
Surface	La classe qui représente une surface.
Solveur	C'est la classe principale du composant « Solveur ». *Attributs :* ▪ Liste des points Output ▪ Nom du fichier Input ▪ Type de la simulation
Discretiseur	C'est la classe principale du composant « Discrétiseur ». *Attributs :*

	▪ Nombre de pas en X ▪ Nombre de pas en Y ▪ Coordonnées des quatre points de la surface ▪ Longueur ▪ Largeur ▪ Liste des points ▪ Liste des éléments ▪ Nom du fichier Input ▪ Type du maillage ▪ La taille du pas en X ▪ La taille du pas en Y

Tableau 3 : Modèle statique

1.3. Diagramme des composants

La réutilisabilité et l'extensibilité de cette application, ont étéles plus importantes contraintes à prendre en considération au cours de la réalisation de ce projet.

La séparation de l'application en composants est la manière la plus adéquatepourrépondre à ces différentes contraintes, elle permet de mieux gérer la possibilité de l'extensibilité de l'application pour répondre à de nouveaux besoins futures.

Dans notre cas, l'application est constituée de quatre composants (*figure 9*):

- L'interface graphique
- Le visualisateur
- Le solveur
- Le discrétiseur

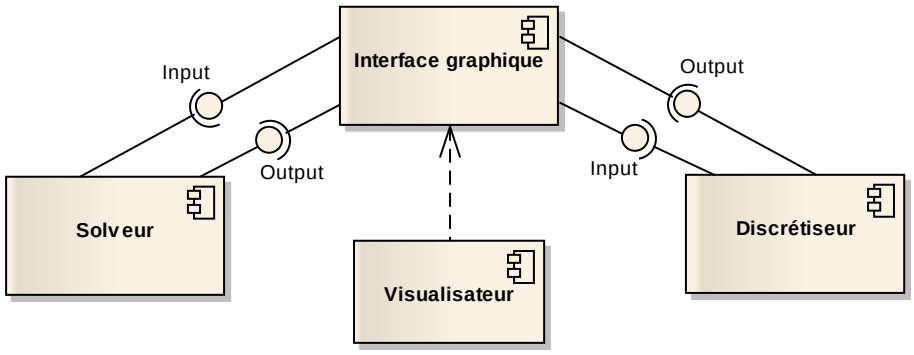


Figure 9 : Diagramme de composants

2. Conception du projet

2.1. L'architecture mise en place

Le patron d'architecture et la méthode de conception MVC (*voir : Chapitre 4, partie 2-L'architecture MVC*), sont la base de l'architecture de notre application.

Cette architecture a été choisie pourle cadre quelle offre pour structurer une application logiciel, dont le but est d'organiser globalement une interface graphique.

Ce patron d'architecture impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.

Les trois parties fondamentales de l'application selon la méthode MVC sont :

- *Vue :* L'interface graphique, qui regroupe tous les composants graphiques, ainsi que toutes les fenêtres et les boites de dialogue.
- *Contrôleur :* C'est le Solveur, il effectue le traitement et le calcul de chaque cas de simulation. Il est techniquement complétement séparé de l'interface graphique. Le discrétiseur fait aussi partie du contrôleur.
- *Modèle :*Est l'ensemble des données d'entrées et de sorties, communiquées entre le contrôleur et la vue, stockées dans des fichiers XML, avec une structure unifiée.

Ci-dessous le schéma global de l'architecture de notre application :

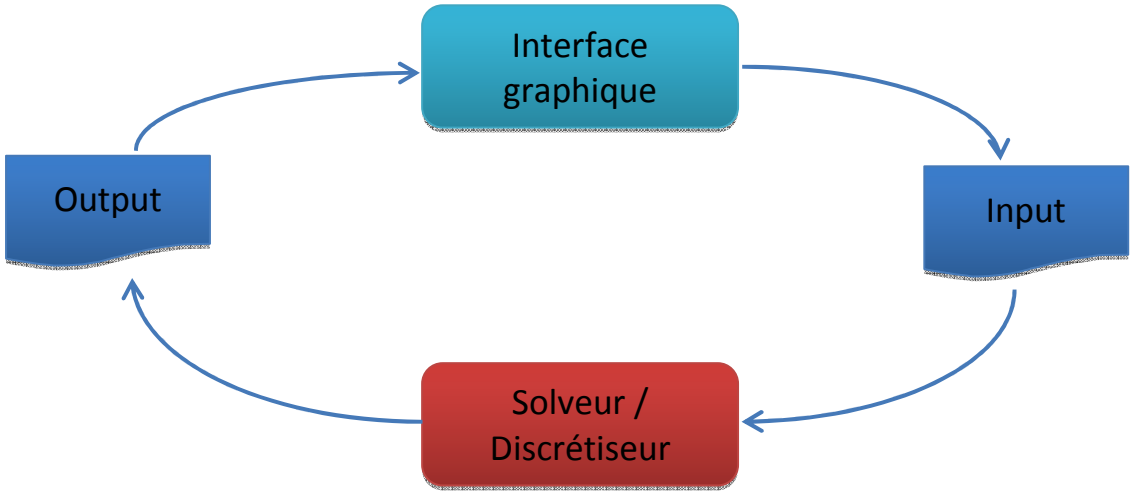


Figure 10 : Schéma global de l'architecture de l'application

2.2. Diagramme de classes détaillées

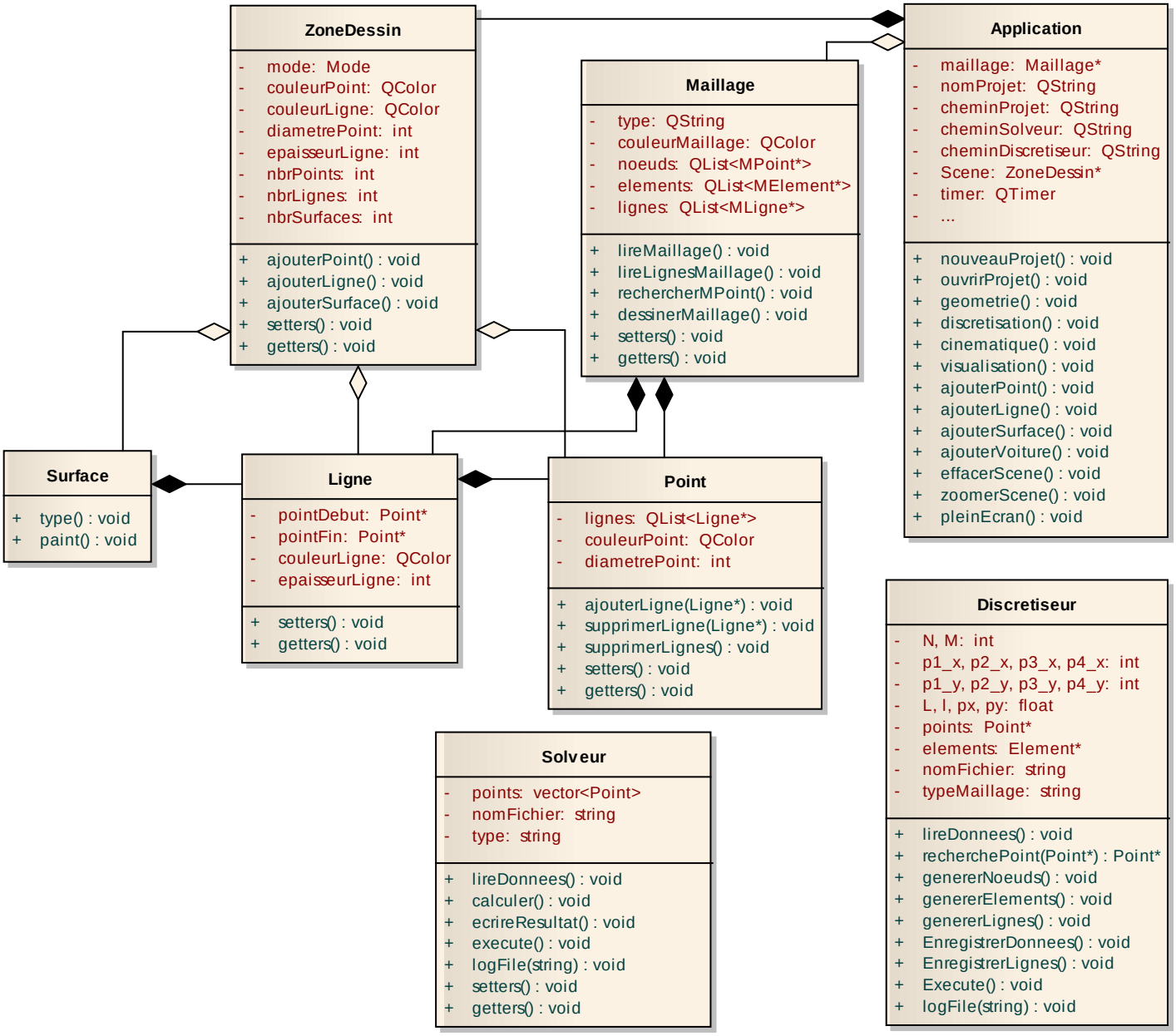


Figure 11 : Diagramme de classes détaillées

2.3. Modèle de données

Pour stocker les données dans un support permanant, on trouve plusieurs solutions et technologies qui répondent parfaitement à ce besoin, et que chacune de ces solutions diffère des autres par un point ou des points forts qui recommandent son utilisation. Comme c'est le cas pour l'XML le langage de balisage extensible, la technologie que nous avons choisie pour notre application. Elle se caractérise par *son universalité, sa gratuité, sa portabilité* ainsi que *son extensibilité*.

Afin d'assurer la communication entre l'interface et/ou le solveur, les fichiers de données XML jouent un rôle primordiale. Les modèles de données pour structurer le contenu desdifférendsfichiers,

s'intègrent dans le but de synchroniser l'accès et la lecture des données de chacun de ces fichiers par les différents composants.

Dans notre application, chaque fichier des données d'entrées ou de sorties,créépar l'interface graphique et/ou le contrôleur (le solveur ou le discrétiseur), a une structure et un modèle de données spécifique. Ci-dessous quelques exemples :

- Fichier de projet :

```
<projet heure="13:40" nom="a" date="02/06/2012">
    <fichier nom="a.xml"/>
    <fichier nom="maillage.xml"/>
    <fichier nom="maillage_tmp.xml"/>
</projet>
```

Figure 12 : Modèle de données du Fichier de projet

- Fichier de maillage :

```
<maillage projet="a" type="Q4" couleur="#a0a0a4">
    <input n="10" m="5">
        <p1 x="0" y="0" />
        <p2 x="200" y="0" />
        <p3 x="200" y="100" />
        <p4 x="0" y="100" />
    </input>
    <output>
        <points>
            <p num="66" x="200" y="100" z="0" />
            ....
            <p num="1" x="0" y="0" z="0" />
        </points>
        <elements>
            <e num="50" s1="54" s2="55" s3="66" s4="65" />
            ....
            <e num="1" s1="1" s2="2" s3="13" s4="12" />
        </elements>
    </output>
</maillage>
```

Figure 13 : Modèle de données du Fichier de maillage

- Fichier de chute libre :

```
<chute projet="a" distance="133">
    <input>
        <balle x="0" y="0" couleur="#0000ff" diametre="30" />
    </input>
    <output>
        <p x="0" y="0" />
        ....
        <p x="0" y="-0.651793" />
    </output>
</chute>
```

Figure 14 : Modèle de données du Fichier de chute libre

Conclusion :

Ce chapitre décrit les différentes étapes d'analyse et de conception qui nous ont permis de faire un choix stratégique pour mettre une architecture efficace et extensible, capable d'accueillir le développement de l'interface et du solveur d'une manière indépendante. Cette architecture permet de gérer,d'une manière indépendante,la validation et la maintenance des trois composants de l'application: interface, solveur, visualisateur. Le quatrième et dernier chapitre, complète notre vision, et porte les détailsdu développement et réalisation de l'application.

Chapitre IV

Mise en œuvre du projet

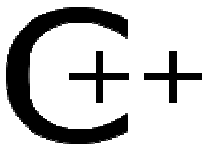
Introduction

Ce chapitre nous permet de s'initier avec les outils et langages utilisés pour la réalisation de cette application, ainsi que l'architecture MVC que nous avons adoptédans cette application. Une deuxième partie sera consacrée aux différentes étapes de réalisation et une description de l'ensemble des composants.

1. Outils et langages utilisés

1.1. C++ / Visual-C++

C++ :



Le C++ est un langage de programmation permettant la programmation sous de multiples paradigmes comme la programmation procédurale, la programmation orientée objet et la programmation générique. C++ est actuellement le 3e langage le plus utilisé au monde. Le langage C++ n'appartient à personne et par conséquent n'importe qui peut l'utiliser sans besoin d'une autorisation ou obligation de payer pour avoir le droit d'utilisation.

Bjarne Stroustrup a développé C++ au cours des années 1980, alors qu'il travaillait dans le laboratoire de recherche Bell d'AT&T. Il s'agissait en l'occurrence d'améliorer le langage C. Il l'avait d'ailleurs nommé *C with classes* (« *C avec des classes* »). Les premières améliorations se concrétisèrent donc par la prise en charge des classes, ainsi que par de nombreuses autres fonctionnalités comme les fonctions virtuelles, la surcharge des opérateurs, l'héritage (simple ou multiple), les « *templates* », la gestion des exceptions, etc.

Le langage C++ est normalisé par l'ISO. Sa première normalisation date de 1998 (ISO/CEI 14882:1998). Le standard a ensuite été amendé par l'erratum technique de 2003 ISO/CEI 14882:2003. Le standard actuel a été ratifié et publié par ISO en septembre 2011 sous le nom de ISO/IEC 14882:2011. (aussi appelé C++11).

En langage C, ++ est l'opérateur d'incrémentation, c'est-à-dire l'augmentation de la valeur d'une variable de 1. C'est pourquoi C++ porte ce nom : cela signifie que C++ est un niveau au-dessus du C. Il existe de nombreuses bibliothèques C++ en plus de la bibliothèque standard de C++ (C++ Standard Library) qui est incluse dans la norme. Par ailleurs, C++ permet l'utilisation de l'ensemble des bibliothèques C existantes.

Visual C++ :

Visual C++ est un environnement de développement intégré pour Windows, conçu par Microsoft pour les langages de programmation C et C++ et intégrant différents outils pour développer,compiler et déboguer un programme en C++ s'exécutant sur Windows, ainsi que des bibliothèques comme les MFC.

Il a par la suite été intégré au Framework Visual Studio, qui constitue ainsi un cadre unique aux divers environnements de développements de Microsoft. Le terme de Visual C++ est toutefois toujours employé pour désigner l'ensemble constitué par Visual Studio configuré pour C et C++.

1.2. Qt / Qt Creator

Qt :

Qt est un framework orienté objet, développé en C++ par Qt DevelopmentFrameworks, filiale de Nokia. Il offre des composants d'interface graphique (widgets), d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc. Qt est par certains aspects un framework lorsqu'on l'utilise pour concevoir des interfaces graphiques ou que l'on architecture son application en utilisant les mécanismes des signaux et slots par exemple.

Qtpermet la portabilité des applications qui n'utilisent que ses composants par simple recompilation du code source. Les environnements supportés sont les Unix (dont Linux) qui utilisent le système graphique X Window System, Windows, Mac OS X et également Tizen. Le fait d'être une bibliothèque logicielle multi plate-forme attire un grand nombre de personnes qui ont donc l'occasion de diffuser leurs programmes sur les principaux OS existants.

Qt est notamment connu pour être la bibliothèque sur laquelle repose l'environnement graphique KDE, l'un des environnements de bureau les plus utilisés dans le monde Linux.

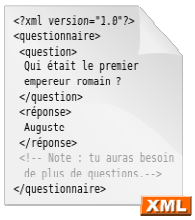
Qt Creator :

Qt Creator est un environnement de développement intégré multi plate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++.

Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercurial ainsi que la documentation Qt. L'éditeur de texte intégré permet l'auto complétion ainsi que la coloration syntaxique. Qt Creator utilise sous Linux le compilateur gcc et MinGW par défaut sous Windows.

Qt Creator a été traduit en français par l'équipe Qt de Developpez.com.

1.3. XML



Le XML (Extensible MarkupLanguage, « langage de balisage extensible ») est un langage informatique de balisage générique qui dérive du SGML. Cette syntaxe est dite extensible car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS… Elle est reconnaissable par son usage des chevrons (<>) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche…) entre systèmes d'informations hétérogènes (interopérabilité). Avec ses outils et langages associés une application XML respecte généralement certains principes :

- La structure d'un document XML est définie et validable par un schéma,
- Un document XML est entièrement transformable dans un autre document XML.

Voici les principaux atouts de XML :

- La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML ;
- Auto descriptif et extensible ;
- Une structure arborescente : permettant de modéliser la majorité des problèmes informatiques ;
- Universalité et portabilité : les différents jeux de caractères sont pris en compte ;
- Déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP ;
- Intégrabilité : un document XML est utilisable par toute application pourvue d'un parser (c'est-à-dire un logiciel permettant d'analyser un code XML) ;
- Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.

Ainsi, XML est particulièrement adapté à l'échange de données et de documents.

2. L'architecture MVC

Le modèle-vue-contrôleur est un patron d'architecture et une méthode de conception qui organise l'interface homme-machine (IHM) d'une application logicielle. Ce paradigme divise l'IHM en un modèle (modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements, synchronisation), chacun ayant un rôle précis dans l'interface. Cette méthode a été mise au point en 1979 par TrygveReenskaug, qui travaillait alors sur Smalltalk dans les laboratoires de recherche Xerox PARC.

Architecture modèle/vue/contrôleur :

L'organisation globale d'une interface graphique est souvent délicate. L'architecture MVC ne résout pas tous les problèmes. Elle fournit souvent une première approche qui peut ensuite être adaptée. Elle offre aussi un cadre pour structurer une application.

Ce patron d'architecture impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.

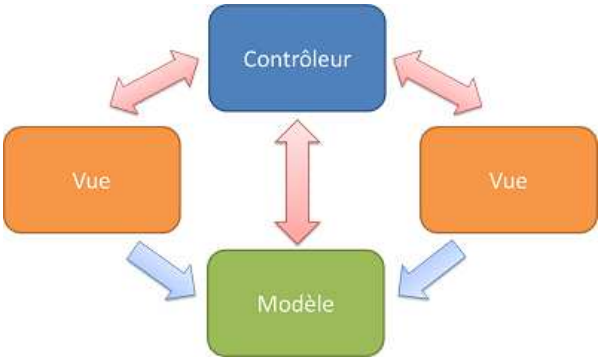


Figure 15 : Architecture modèle/vue/contrôleur

Le modèle :

Le modèle représente le comportement de l'application : traitements des données, interactions avec la base de données, etc. Il décrit ou contient les données manipulées par l'application. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient. Le modèle offre des méthodes pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle sont dénués de toute présentation.

La vue :

La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, etc.). Ces différents événements sont envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

Plusieurs vues, partielles ou non, peuvent afficher des informations d'un même modèle. Par exemple, une application de conversion de bases a un entier comme unique donnée. Ce même entier peut être affiché de multiples façons (en texte dans différentes bases, bit par bit avec des boutons à cocher, avec des curseurs). La vue peut aussi offrir la possibilité à l'utilisateur de changer de vue.

Elle peut être conçue en HTML ou tout autre marquage de présentation.

Le contrôleur :

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle, et ce dernier notifie la vue que les données ont changée pour qu'elle se mette à jour. D'après le patron de conception observateur/observable, la vue est un "observateur" du modèle qui est lui "observable". Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier. Le contrôleur n'effectue aucun traitement, ne modifie aucune donnée. Il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande.

Par exemple, dans le cas d'une base de données gérant les emplois du temps des professeurs d'une école, une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours. Le contrôleur ajoute

ce cours au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci ne modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

Quand un même objet contrôleur reçoit les événements de tous les composants, il lui faut déterminer quelle est l'origine de chaque événement. Ce tri des événements peut s'avérer fastidieux et peut conduire à un code pas très élégant (un énorme Switch). C'est pourquoi le contrôleur est souvent scindé en plusieurs parties dont chacune reçoit les événements d'une partie des composants.

Avantages et inconvénients :

Un avantage apporté par ce modèle est la clarté de l'architecture qu'il impose. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration indépendante de chaque composant de l'application. En effet, la modification des traitements ne change en rien la vue. Par exemple on peut passer d'une base de données de type SQL à XML en changeant simplement les traitements d'interaction avec la base, et les vues ne sont pas affectées.

Le MVC montre ses limites dans le cadre des applications utilisant les technologies du web, bâties à partir de serveurs d'applications. Des couches supplémentaires sont alors introduites ainsi que les mécanismes d'inversion de contrôle et d'injection de dépendance.

3. Réalisation

3.1. Informations statistiques sur le code

Pour réaliser cette application, nous avons développé et écris des centaines de lignes de codes et des dizaines de fonctions. Ci-dessous un tableau qui regroupe quelques chiffres statistiques approximatifs sur le code des différentes classes des différends composants de l'application (*tableau 4*) :

	Nombre de fonctions	Nombre des lignes du .h	Nombre des lignes du .cpp
Application	64	230	2240
ZoneDessin	23	70	200
Maillage	12	55	150
Point	12	35	60
Ligne	9	35	45
Surface	3	25	30
Solveur	8	40	290
Discretiseur	12	65	390
8	**143**	**555**	**3405**

Tableau 4 : Informations statistiques sur le code

- Total - à peu près **-4000** lignes de codes.

3.2. L'interface graphique

L'interface grahique de notre application regroupe quatre outils principaux. Dans un premier temps, on trouve l'outil géométrie qui sert à dessiner les différentes formes géométriques et de préparer la base géométrique nécessaire à une simulation. Par la suite vient l'outil de discrétisation, son rôle est de discrétiser une surface en collaboration avec le discrétiseur. L'outil cinématique, qui spécifie le type de mouvement et les données nécessaires à la simulation, constitut l'étape avant le lancement des calculs effectués par le solveur suite à une communication en arrière plan à l'aide d'un processus.

La partie visualisationest un composant totalement indépendant de l'application, elle est intégrée dans l'interface graphique afin de regrouper les différends outils graphiques dans une seule interface simple et érgonomique, où la plupart des fonctionnalités sont faciles d'accès et souples d'utilisation, soit,à l'aide des racourcis clavier,soit à l'aide de la sourie.

Au lancement de l'application, l'interface graphique est entièrement désactivée (*figure 16*), ellene sera activée et prête à l'utilisation qu'après la création d'un nouveau projet, ou à l'ouverture d'un ancien :

Figure 16 : Interface graphiqueentièrement désactivée

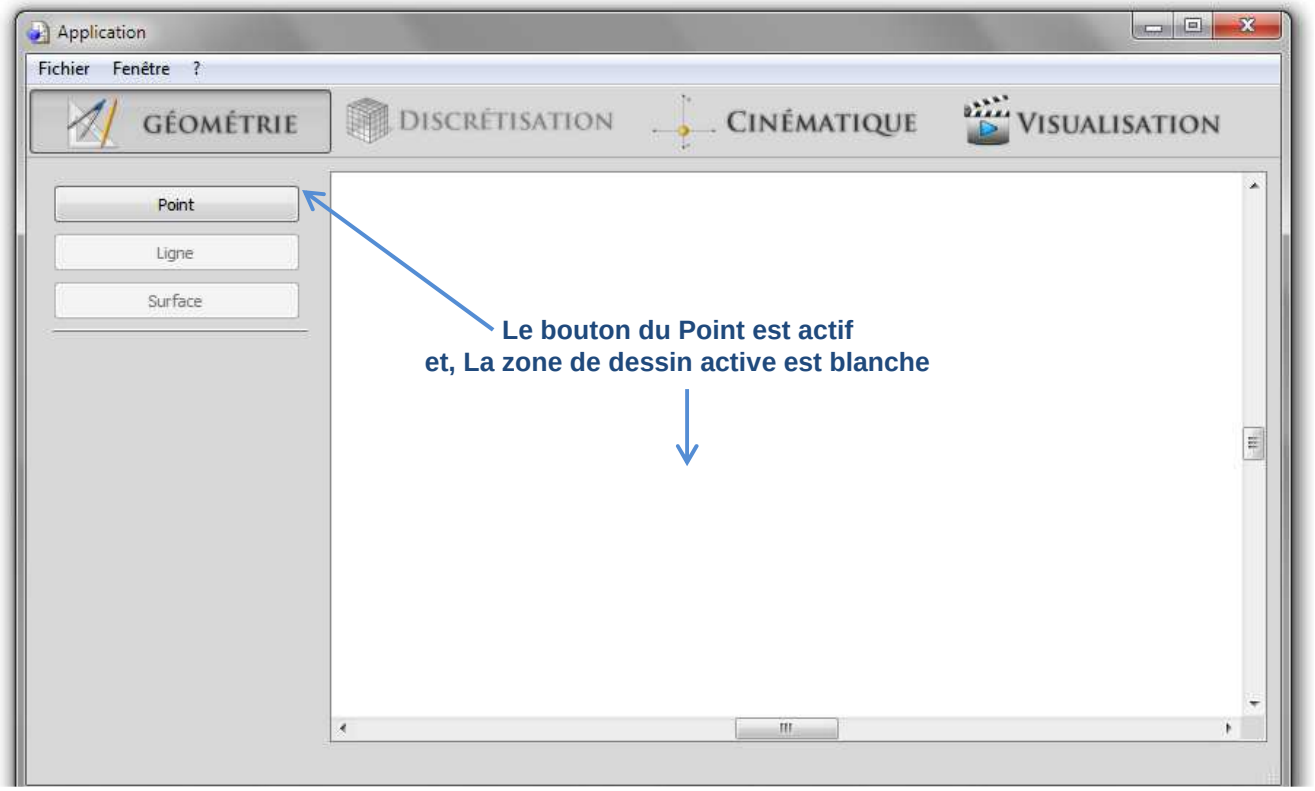


Figure 17 : Interface graphique activée

Avant tout, nous découpons l'interface graphique (*figure 18*)en cinq parties essentielles, chaque partie est dédiéeà des fonctionnalités spécifiques :

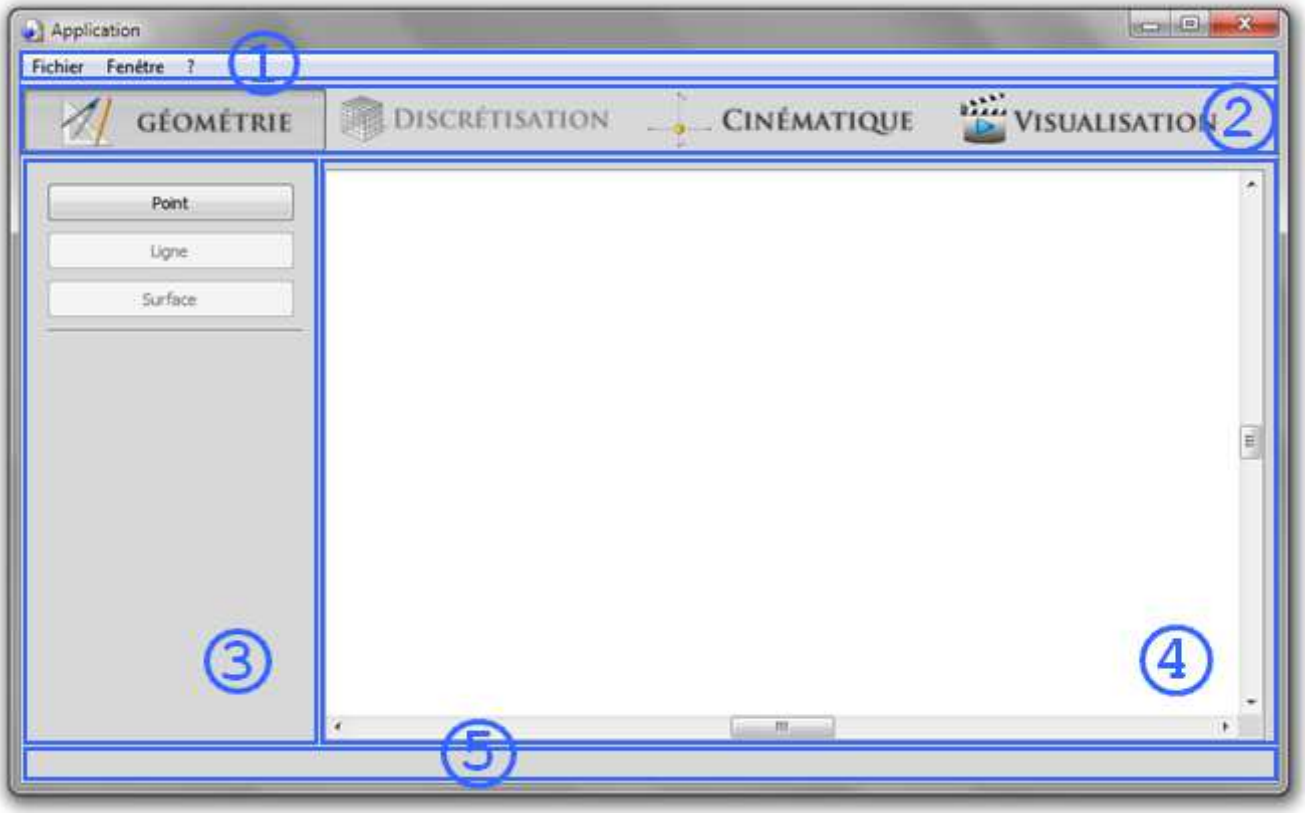


Figure 18 : Interface graphique divisée

[1] : La barre de menu, elle contient le menu Fichier, le menu Fenêtre et le menu aide « ? ».

[2] : Le menu Horizontal, regroupe les quatre outils.

[3] : Le menu Vertical est la partie dynamique de l'interface, il change vis-à-vis de l'outil choisi.

[4] : La zone de dessin représente la scène où nous dessinons les formes géométriques.

[5] : La barre d'état affiche l'état actuel de l'interface.

5.1.1. La barre de menu

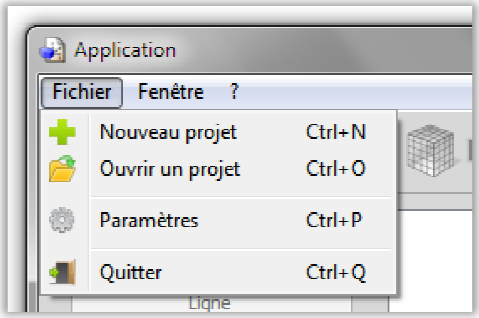


Figure 19 : Menu fichier

Le menu Fichier(*figure 19*) permet de créer un nouveau projet (raccourcis clavier Ctrl+N), d'ouvrir un projet (raccourcis clavier Ctrl+O), de paramétrer l'interface graphique en spécifiant l'emplacement du solveur et du discrétiseur ainsi que d'autre informations nécessaires au bon fonctionnement de l'application (raccourcis clavier Ctrl+P), et, finalement, la possibilité de Quitter l'application (raccourcis clavier Ctrl+Q).

Le menu Fenêtre (*figure 20*) offre la possibilité de zoomer la scène ou la zone de dessin, de la nettoyer en supprimant tous les objets graphiques insérés dans la scène, et aussi de mettre la fenêtre en mode plein écran (raccourcis clavier Ctrl+F11).

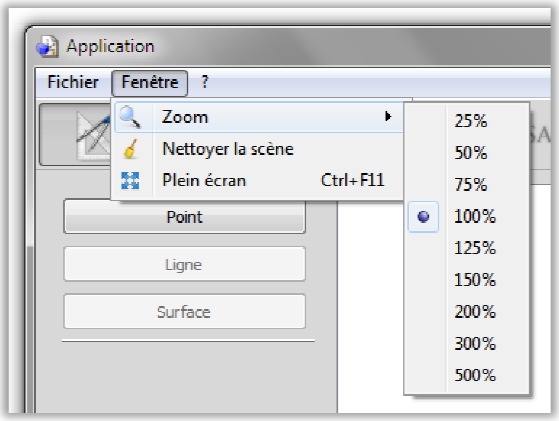


Figure 20: Menu fenêtre

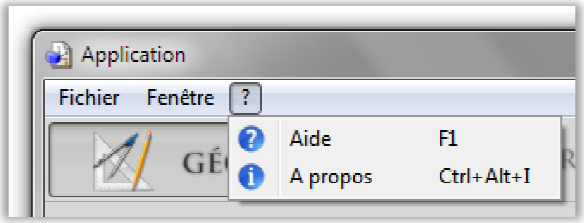


Figure 21 : Menu « ? »

Le menu aide « ? »(*figure 21*) contient une aide sur l'utilisation de l'application (raccourcis clavier F1), ainsi qu'un « A propos », qui présente quelque informations sur l'application (raccourcis clavier Ctrl+Alt+I).

La *figure 22*, représente la boite de dialogue de création d'un nouveau projet :

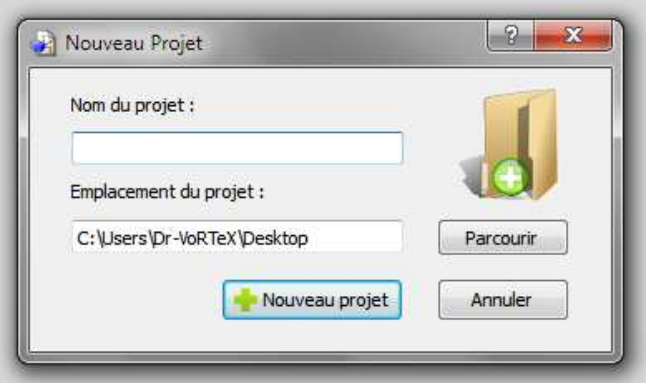


Figure 22 : Ajouter un nouveau Projet

La *figure 23*illustre la boite d'ouverture d'un ancien projet :

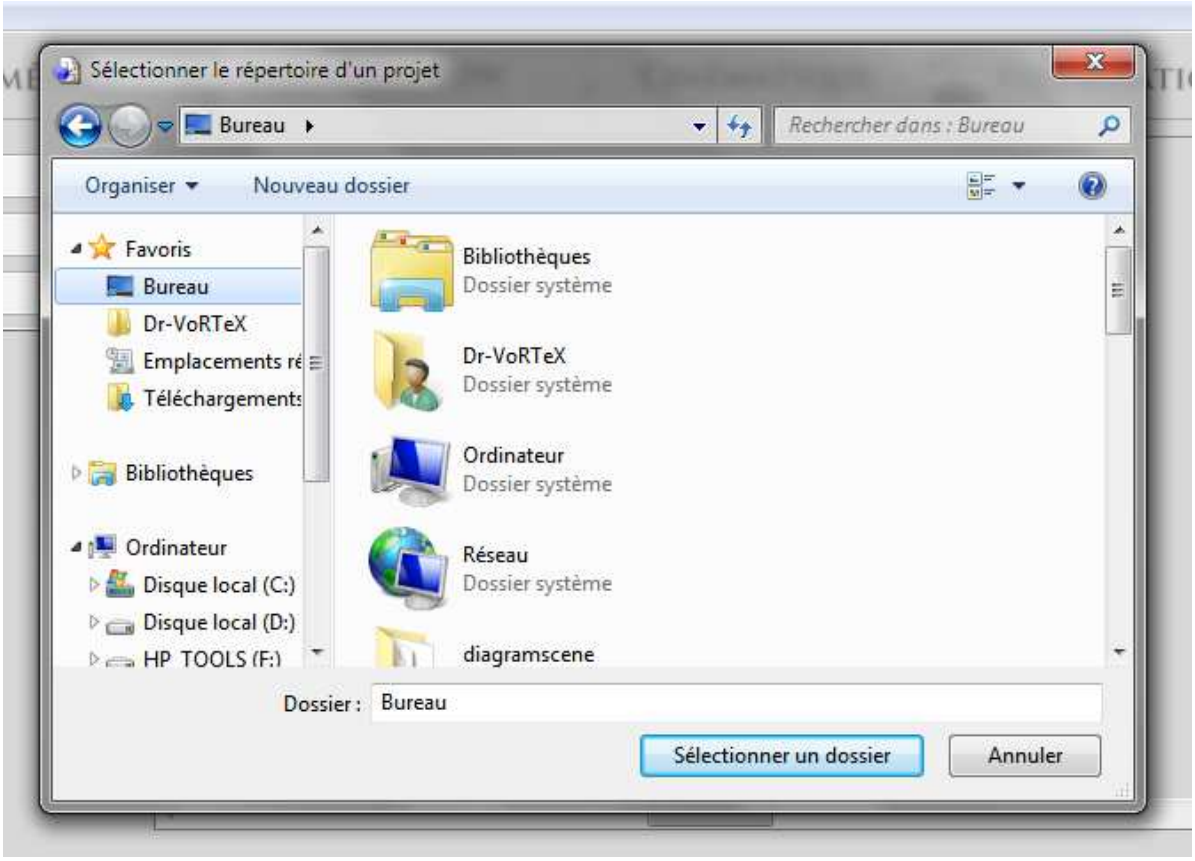


Figure 23 : Ouvrir un Projet

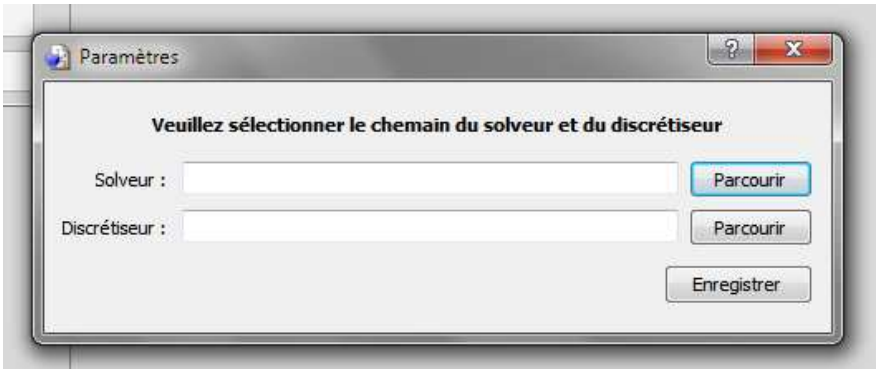


Figure 24 : Boite de dialogue paramètres

5.1.2. Le menu Horizontal

Le menu Horizontal permet de basculer entre les quatreoutils de l'interface graphique, il est composé de boutons cochables dont chacun est caractérisé par une icone indicative et un titre (*figure 25*).

Figure 25 : Menu Horizontal

5.1.3. Le menu Vertical

Le menu Vertical est le menu principalde l'interface graphique, il regroupe l'ensemble des formulaires et composants graphiques de chacun des outils cités précédemment.

L'outil géométrie a son propre menu vertical, dont une partie est fixeet contient trois sous-menus qui changent selon le choix de la forme géométrique (Point, Ligne, Surface).

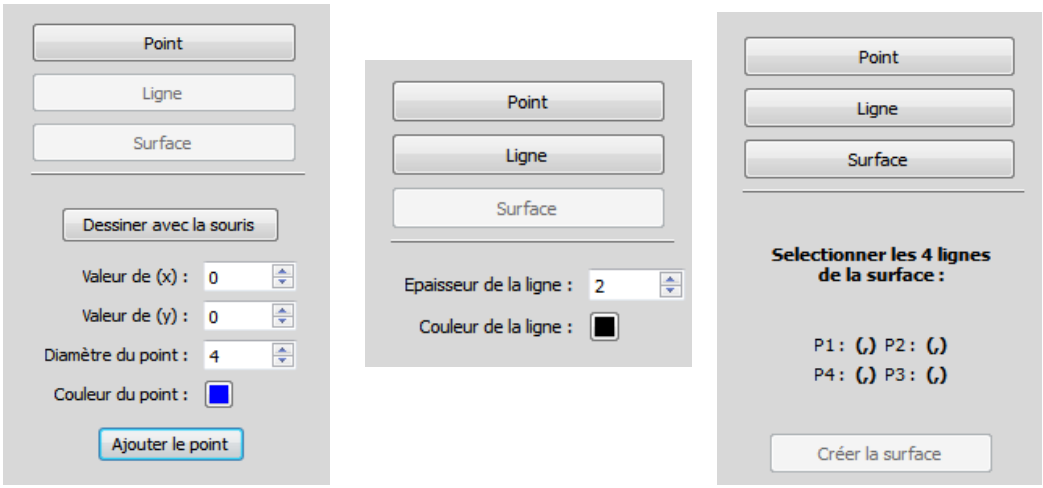


Figure 26 : Menu Vertical - Géométrie-

La*figure 27*présente le menu de discrétisation :

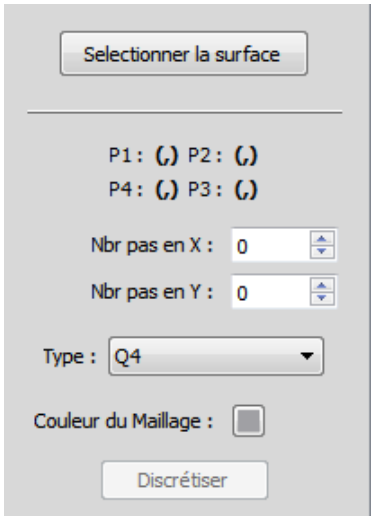


Figure 27 : Menu Vertical - Discrétisation-

L'outil cinématique contient trois sous-menus (*figure 28*), un, pour le mouvement de la chute libre, un, pour la distance de sécurité et, un autre pour l'étude de crash.

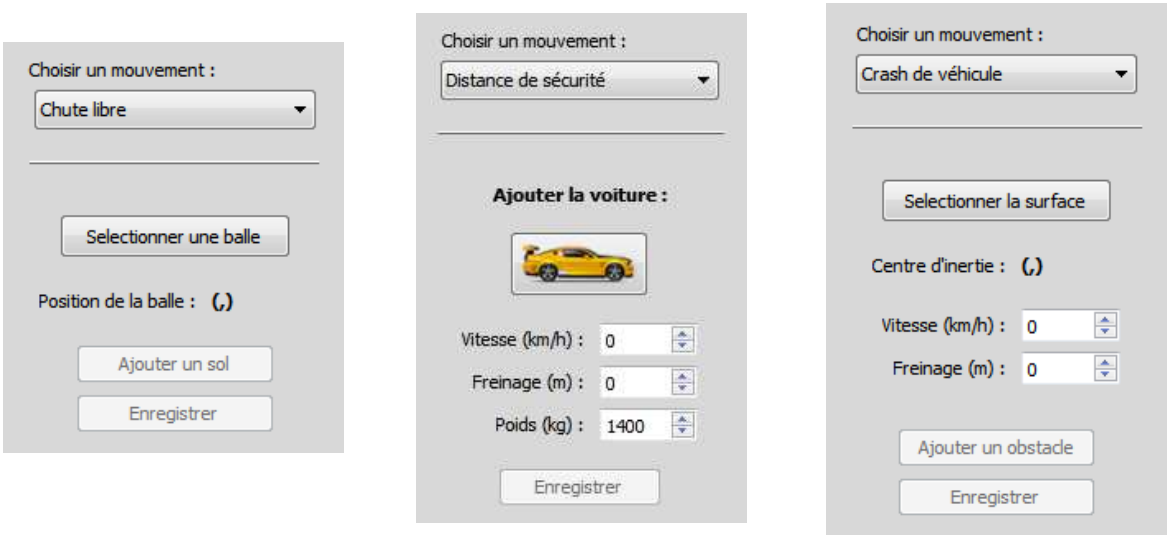


Figure 28: Menu Vertical - Cinématique -

Le menu Visualisation (*figure 29*) sert à visualiser le mouvement d'une simulation, il est composé de deux boutons, un pour lancer la visualisation et un autre pour l'arrêter.

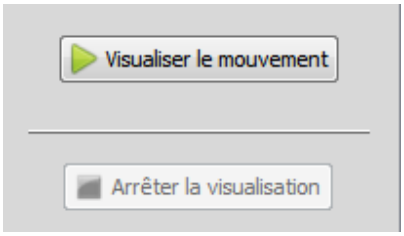


Figure 29 : Menu Vertical - Visualisation-

5.1.4. La zone de dessin

La zone de dessin ou la scène graphique(*figure 30*), c'est l'emplacement réservé au dessin des formes graphiques. Elle est sous forme d'un rectangle avec un arrière-plan blanc, qui peut être agrandi suite au redimensionnementde la fenêtre. Comme elle peut être aussi zoomer ou dé-zoomer grâce à la fonction zoom du menu fenêtre. Il existe aussi une fonction dans le menu fenêtre pour la nettoyer.

La zone de dessin est dotée aussi de deux barres de défilements, verticale et horizontale, qui permette de se déplacer dans la totalité de la scène.

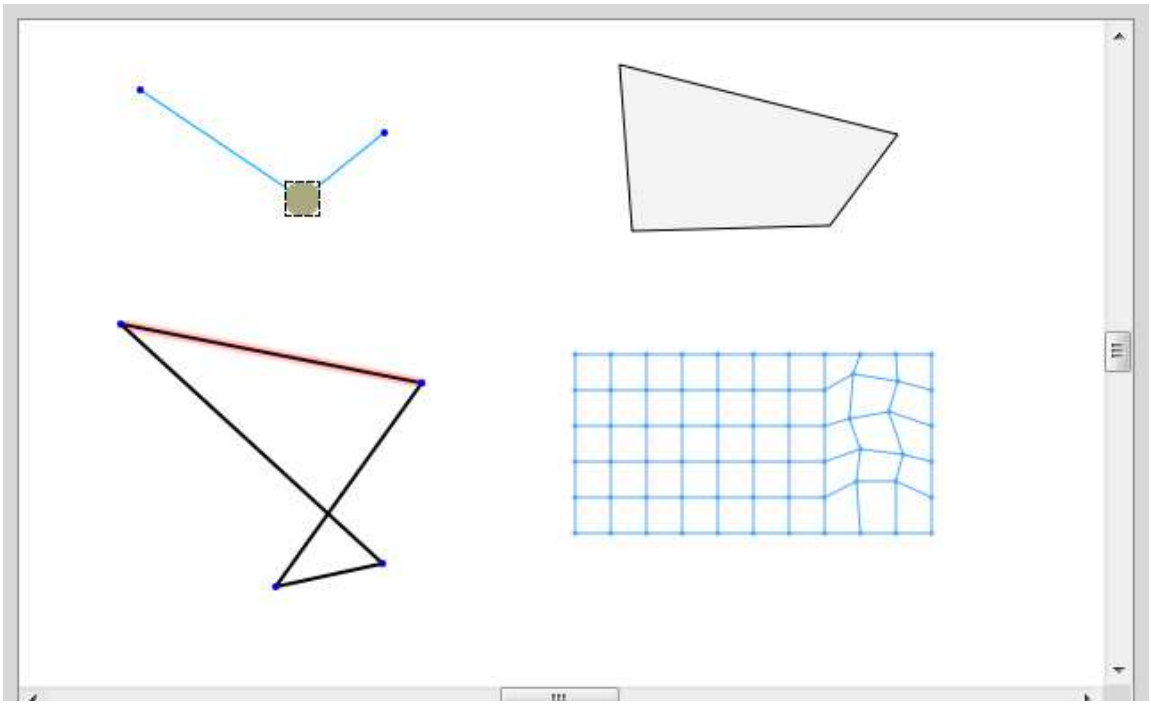


Figure 30 : La zone de dessin

3.3. Le solveur

Suite à l'architecture adoptée dans ce projet, le solveur est le composant qui joue le rôle du contrôleur dans la méthode MVC. Il permet d'effectuer l'ensemble des traitements et calculs des différentes simulations réalisables par notre application.

C'est un programme indépendant, développé en C++ par l'IDE Visual C++, il contient la classe principale Solveur, et d'autres classes secondaires comme « Point », « Element »...Etc.

La classe Solveur traite chaque cas de simulation de manière séparée. Elle reçoit en premier temps, le chemin du fichier des données d'entrées en paramètre de la fonction *main* (*figure 31*), qui est par la suite passécomme paramètre à son constructeur.

```
#include <iostream>
#include "solveur.h"

using namespace std;

int main(int argc, char *argv[])
{
    if(argc == 2)
    {
        Solveur S(argv[1]);
        S.Execute();
    }
    else
        cout << "Erreur : Parametre manquant" << endl;

    return 0;
}
```

Figure 31 : Code de la fonction main du Solveur

```
class Point
{
public:
    Point(int N=0){Num = N;}
    int Num;
    float x,y,z;
    Point *suivant;
};

class Solveur
{
    vector<Point> points;
    string fichier;
    Point *balle;
    Point *voiture;
    int num;
    string type;
    int distance;
    float vitesse;
    int freinage;
    int poids;

public:
    Solveur(string);
    void lireDonnees();
    void calculer();
    void ecrireResultat();
    void Execute();
    void logFile(string);
    void setFichier(string f) { fichier = f; }
    string getFichier() { return fichier; }
};
```

Figure 32 : Code de la classeSolveur

Pour traiter les fichiers XML, nous avions besoin d'un parseur XML, notre choix a étéfixé sur le parseur TinyXML. C'est un parseur XML pour le langage C++. Il est distribué sous la licence *zlib*. Il est capable d'analyser un document XML pour générer un arbre DOM, il est aussi capable de lire et d'écrire des fichiers XML.

TinyXML est caractérisé par sa petite taille, sa simplicité et aussi par son code léger.

La *figure 33* montre un extrait du code d'ouverture d'un fichier XML et sa transformation en un arbre DOM.

```
TiXmlDocument doc(fichier);
if(!doc.LoadFile())
{
    logFile("Solveur@lireDonnees : erreur lors du chargement du fichier \"chute.xml\"");
    return;
}
else
{
    TiXmlHandle hdl(&doc);
    TiXmlElement *element = hdl.FirstChildElement().Element();
    element->QueryIntAttribute("distance", &this->distance);

    element = hdl.FirstChildElement().FirstChildElement().FirstChildElement().Element();
    balle = new Point();
    element->QueryFloatAttribute("x", &balle->x);
    element->QueryFloatAttribute("y", &balle->y);
}
```

Figure 33 : Code de lecture d'un fichier XML

3.4. Le discrétiseur : générateur du maillage

Le discrétiseur joue le même rôle d'un sous solveur au sein de l'architecture de l'application, car il est aussi un composant et un programme indépendant. Il reçoit en paramètre de sa fonction *main* le chemin du fichier des données d'entrées, qui est passécomme paramètre à son constructeur.

Le discrétiseur permet de générer le maillage d'une surface. Il est basé sur les coordonnées des quatre coins de la surface, ainsi que le nombre de pas en X et en Y. Premièrement, il lit les données d'entrées enregistrées dans le fichier « *maillage.xml* », par la suite il génère l'ensemble des nœuds et des éléments et aussi les lignes qui vont permettre d'afficher le maillage.

```
#include <iostream>
#include "discretiseur.h"

using namespace std;

int main(int argc, char *argv[])
{
    if(argc == 2)
    {
        Maillage M(argv[1]);
        M.Execute();
    }
    else
        cout << "Erreur : Parametre manquant" << endl;

    return 0;
}
```

Figure 34 : Code de la fonction main du Discrétiseur

```
#ifndef MAILLAGE_H
#define MAILLAGE_H

#include <iostream>

using namespace std;

class Point
{
    public:
    Point(int N=0){Num = N;}
    int Num;
    float x,y,z;
    Point *suivant;
};

class Element
{
    public:
    int Num;
    Point *somet1, *somet2, *somet3, *somet4;
    Element *suivant;
};

class Lignes
{
    public:
    int Num;
    Point *somet1, *somet2, *somet3, *somet4, *somet5;
    Lignes *suivant;
};

class Maillage
{
    int N, M;
    int p1_x, p1_y;
    int p2_x, p2_y;
    int p3_x, p3_y;
    int p4_x, p4_y;
    float L, l, px, py;
    Point *points;
    Element *elements;
    Lignes *lignes;
    string nomFichier;
    string type;
    const char *nomProjet;
    bool erreurFichier;
    public:
    Maillage(string);
    ~Maillage();
    void AfficherDonnees();
    void saisirDonnees();
    void lireDonnees();
    Point* recherche(Point*, int);
    void genererNoeuds();
    void genererElements();
    void InverserListe();
    void genererLignes();
    void EnregistrerDonnees();
    void EnregistrerLignes();
    void Execute();
    void logFile(string);
};

#endif // MAILLAGE_H
```

Figure 35 : Code de la classe Discrétiseur

3.5. Points à améliorer

Vu les contraintes du temps alloué au stage, les exigences ducahier des charges, durapport et surtout des délais, il nous a été indispensable de reporterles correctionsd'amélioration de chaque fonctionnalité. Néanmoins nous avons pu recenser une liste des points à améliorer.

Liste des points à améliorer:

- Enregistrement des fichiers de donnéesgéométriques.
- La réouverture d'un projet existant.
- Doter l'outildedessin d'une marge d'erreur au niveau de la sélection des formes, et surtout des lignes.
- Avoir la possibilité de garder la relation entre la surface et son maillage.
- Améliorer le discrétiseur afin qu'il puisse traiter toutes les formes d'une surface.
- Intégrer le paramètre des unités de mesure.
- Permettre la coloration des éléments du maillage.

3.6. Liste des bugs

Dans tout projet professionnel, il faut s'attendre à l'apparition de quelques bugs durant son cycle de développement. Dans ce projet nous avons rencontré quelques bugs, ci-dessous décrits :

- Problème de mémoire après la suppression et le déplacement d'un des deux points qui compose la ligne.
- La ligne ne change pas d'emplacement après la modification en saisissant les coordonnées d'un de ses points.
- Problème mémoire apparu lors de la création de deux surfaces en même temps.
- Les traces des formes sélectionnées restent sur la scène après leurs déplacements.

Conclusion :

Nous avons utilisé un ensemble d'outils et de langages qui nous ont été nécessaires pour réaliser notre projet. Ce dernier se base sur une architecture MVC qui permet de faciliter le développement des différends composants, d'une manière indépendante ;elle facilite aussi la maintenance et la validation de manièreséparée. La structure modulaire de chaque composant, rend l'organisation du travail en équipe plus efficace,elle assure une communication transversale par un responsable de projet.

Conclusion et perspectives

Ce stage de fin d'études, effectué au sein de l'équipe de recherche EMSN à l'ENSAO, m'a permis, dans un premier temps, de mettre en pratique les connaissances théoriques et techniques acquises au court de ma formation en licence sciences et techniques de l'FSTF, et aussi j'ai eu l'occasion d'intégrer une véritable équipe de recherche scientifique et universitaire, en bénéficiant d'une nouvelle approche à traiter les sujets, une approche de recherche, une approche scientifique et technique qui ouvre de nouveaux horizons et de nombreuses ambitions.

Scientifiquement, ce stage m'a permis de s'initier avec le domaine de la modélisation et simulation numérique, c'est un domaine très prometteur dans le secteur de l'industrie. Il représente un maillant primordiale dans la chaine de production de l'industrie moderne.

Techniquement, j'ai pu pratiquer, de manière plus approfondie, mes connaissances sur le langage de programmation orienté objet C++, ainsi que son framework graphique le Qt, en faisant face aux différentes difficultés techniques rencontrées tout au long du cycle de développement de cette application. La documentation officielle de Qt était la source principale pour essayer de résoudre et d'implémenter les solutions techniques sollicitées par l'ensemble des outils graphiques.

L'adoption de l'architecture MVC dans cette application, m'a permis de découvrir l'apport très intéressant de cette méthode de conception dans une application logiciel, en terme d'extensibilité, de séparation modulaire, de facilité de maintenance ainsi que la réutilisabilité de ses composants.

Cette expérience a été un véritable changement dans ma vie professionnel, elle m'a aidé à découvrir un domaine que j'ignorais auparavant, et de participer au développement et à la réalisation d'un projet de taille, en respectant un cahier des charges et en tenant compte des délais prévus. Cela n'a pu être possible sans l'encadrement fort précieux et l'assistance pertinente de Mr RAHMOUNE Mohamed, mon encadrant de stage à l'ENSAO.

Enfin, je suis sûr que la conception effectuée et l'architecture mise en place garantira l'extensibilité de l'application développée et permettra à l'équipe de recherche EMSN, par le biais d'autres intervenants et collaborateurs, d'intégrer d'autres modules pour réponde aux futurs besoins.

A titre d'exemple et de perspectives, l'application peut être modifiée afin de prendre en considération la sauvegarde de la géométrie des formes ; d'y intégrer la notion de version pour éviter les éventuels problèmes d'incompatibilité entre les projets de différentes versions de l'application ; d'améliorer la performance de l'outilde dessin de la géométrie, en le dotant d'une intelligence et des marges d'erreurs dans le traitement des évènements effectués par la souris et aussi de déployer et mettre en place un installable de l'application pour les différentes plateformes afin de faciliter son utilisation.

Index

Bibliographie

Livres

Programmation en C++ et Génie logiciel - Vincent T'kindt| DUNOD

Qt4 et C++ - Programmation d'interfaces GUI - Jasmin Blanchette, Mark Summerfield | PEARSON

Adresses internet

http://www.cplusplus.com - La référence officiel de C++

http://qt.nokia.com - Le site officiel du framework Qt

http://doc.qt.nokia.com - La documentation officielle de Qt

http://www.w3.org/XML - La partie XML de la W3

http://www.grinninglizard.com/tinyxml - Le site officiel du parseur TinyXML

http://www.developpez.com - Communauté francophone dédiée au développement informatique

http://www.siteduzero.com - Site communautaire de tutoriels gratuits en programmation et développement web

http://qt-project.org - The official Qt community site

http://www.commentcamarche.net - Comment Ça Marche (CCM) - Communauté informatique

http://fr.wikipedia.org - Wikipédia, l'encyclopédie libre

http://www.qtcentre.org - Qt Centre Forum - Qt Centre Community Portal

http://www.w3schools.com/xml - XML Tutorial

http://stackoverflow.com - A language-independentcollaborativelyedited question and answersite for programmers

Annexes

Annexe I : Processus Y

Le processus en « Y » est un processus de développement logiciel qui se caractérise par :

1. Processus incrémental piloté par les risques

Définition d'incrément

Un incrément constitue un ensemble d'étapes de développement qui aboutit à la construction de tout ou partie du système. Le contenu d'un incrément est porteur d'améliorations ou d'évolutions du système et il peut être évalué par les utilisateurs.

Le processus de développement en « Y » se reproduit à différents niveaux d'avancement en se terminant sur la livraison d'un nouvel incrément.

Il a été conçu pour gérer en priorité et en parallèle les risques de nature fonctionnelle et technique:

- d'une part, les risques d'imprécision fonctionnelle, et d'inadéquation aux besoins sur la branche gauche.

- d'autre part les risques d'incapacité d'intégrer les technologies, et d'inadéquation technique sur la branche droite.

2. Processus piloté par les exigences des utilisateurs

La majorité des risques proviennent du non adéquation technique et fonctionnelle du système aux besoins des utilisateurs. Les exigences des utilisateurs sont donc prioritairement traitées dans les deux branche du processus en Y.

L'enjeu du processus en Y est donc de développer le point de vue utilisateur et de construire la spécification puis la conception objet à partir des concepts maniés par les acteurs du système.

3. Processus de modélisation avec UML

Il apparaît difficile d'envisager le processus 2TUP sans recourir à UML comme support. Le processus en Y utilise UML dans toutes les phases par exemple : les cas d'utilisation et les diagrammes de séquences dans la phase de spécification fonctionnelle, diagramme de classes dans la phase de structuration etc.

Annexe II : UML

Introduction

La description de la programmation par objets a fait ressortir l'étendue du travail conceptuel nécessaire : définition des classes, de leurs relations, des attributs et méthodes, des interfaces etc.

Pour programmer une application, il ne convient pas de se lancer tête baissée dans l'écriture du code : Il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle modélisation; son produit est un modèle.

Les spécifications fournies par la maîtrise d'ouvrage en programmation impérative étaient souvent floues: les articulations conceptuelles (structures de données, algorithmes de traitement) s'exprimant dans le vocabulaire de l'informatique, le modèle devait souvent être élaboré par celle-ci.

L'approche objet permet en principe à la maîtrise d'ouvrage des exprimer de façon précise selon un vocabulaire qui, tout en transcrivant les besoins du métier, pourra être immédiatement compris par les informaticiens. En principe seulement, car la modélisation demande aux maîtrises d'ouvrage une compétence, un professionnalisme qui ne sont pas aujourd'hui répandus.

UML en œuvre

UML n'est pas une méthode (i.e. une description normative des étapes de la modélisation): ses auteurs ont en effet estimé qu'il n'était pas opportun de définir une méthode en raison de la diversité des cas particuliers. Ils ont préféré se borner à définir un langage graphique qui permet de représenter, de communiquer les divers aspects d'un système d'information (aux graphiques sont bien sûr associés des textes qui expliquent leur contenu). UML est donc un métalangage car il fournit les éléments permettant de construire le modèle qui, lui, sera le langage du projet.

Il est impossible de donner une représentation graphique complète d'un logiciel, ou de tout autre système complexe, de même qu'il est impossible de représenter entièrement une statue (à trois dimensions) par des photographies (à deux dimensions). Mais il est possible de donner sur un tel système des vues partielles, analogues chacune à une photographie d'une statue, et dont la juxtaposition donnera une idée utilisable en pratique sans risque d'erreur grave.

Les diagrammes d'UML

UML 2.0 comporte ainsi treize types de diagrammes représentant autant de vues distinctes pourreprésenter des concepts particuliers du système d'information. Ils se répartissent en deux grands groupes :

Diagrammes structurels ou diagrammes statiques (UML Structure)

- diagramme de classes (Class diagram)
- diagramme d'objets (Object diagram)

- diagramme de composants (Component diagram)
- diagramme de déploiement (Deploymentdiagram)
- diagramme de paquetages (Package diagram)
- diagramme de structures composites (Composite structure diagram)

Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)

- diagramme de cas d'utilisation (Use case diagram)
- diagramme d'activités (Activitydiagram)
- diagramme d'états-transitions (State machine diagram)
- diagrammed'interaction(Interactiondiagram)
- diagramme de séquence (Sequencediagram)
- diagramme de communication (Communication diagram)
- diagramme global d'interaction (Interaction overviewdiagram)
- diagramme de temps (Timing diagram)

Ces diagrammes, d'une utilité variable selon les cas, ne sont pas nécessairement tous produits à l'occasion d'une modélisation. Les plus utiles pour la maîtrise d'ouvrage sont les diagrammes d'activités, de cas d'utilisation, de classes, d'objets, de séquence et d'états-transitions. Les diagrammes de composants, de déploiement et de communication sont surtout utiles pour la maîtrise d'œuvre à qui ils permettent de formaliser les contraintes de la réalisation et la solution technique.

Annexe III : La simulation numérique

Sans prétendre à une description exhaustive, on peut dire que les simulations numériques peuvent permettre de comprendre (recherche fondamentale ou appliquée), prédire (météorologie, climatologie, épidémiologie, …) ou concevoir (automobile, aéronautique, génie civil,..).

Pourquoi la simulation numérique ?

La simulation numérique est une approche qui permet au chercheur et à l'ingénieur d'analyser des phénomènes qui par leur complexité échappent au calcul « traditionnel ».

Cette complexité peut être de nature très différente.

- **Elle peut être liée au nombre d'objets à prendre en compte.**Ainsi, en utilisant les lois de la gravitation, le physicien sait calculer depuis longtemps le mouvement d'une planète autour d'une étoile. Le mouvement des quelques millions d'étoiles à l'intérieur d'une galaxie est régi par les mêmes lois, mais seule la simulation numérique peut permettre de l'étudier.

- on a aussi recours à la simulation numérique **lorsqu'un très grand nombre de paramètres doit être incorporé dans un calcul**. La propagation d'une vague dans l'océan est gouvernée par la dynamique des fluides et il est possible de calculer sa vitesse dans des situations simples, idéalisées. Si, dans le cas d'un tsunami, on veut prédire avec une précision suffisante la hauteur de la vague en chaque point du littoral, il faut tenir compte dans les équations de la morphologie des fonds marins et du rivage sur toute la zone concernée.

- **la complexité d'un problème peut aussi provenir du nombre de phénomènes qui interviennent.** Ainsi l'évolution du climat sur le long terme doit intégrer la modélisation de phénomènes thermiques, géologiques, biologiques et chimiques … De plus, souvent, les sources de « complexité » que nous avons évoquées peuvent se cumuler. C'est en particulier le cas d'une arme thermonucléaire où un grand nombre de phénomènes physiques s'imbriquent et ce, dans des géométries complexes.

A quoi peuvent servir les simulations numériques ?

Sans prétendre à une description exhaustive, on peut dire que les simulations numériques peuvent permettre de :

- comprendre (recherche fondamentale ou appliquée);
- prédire (météorologie, climatologie, épidémiologie, …);
- concevoir (automobile, aéronautique, génie civil, …).

Dans les domaines qui ont été évoqués, les outils traditionnels que sont devenus l'expérimentation, les tests, les maquettages, etc, sont devenus très coûteux en temps ou en argent (exemple les crashs tests), parfois insuffisamment représentatifs (une maquette ne permettra pas à un ingénieur de choisir le béton le mieux adapté à un pont ou à un barrage, ni comment il vieillira) ou tout simplement impossible pour diverses raisons (les essais nucléaires par exemple !).

Quels sont les ingrédients d'une simulation numérique ?

Au-delà des ordinateurs, les outils de la simulation numériques sont :

- des modèles mathématiques (ensemble d'équations) qui décrivent les phénomènes étudiés (physiques ou autres),
- des méthodes ou algorithmes qui permettent à la machine de résoudre ces équations,
- des données sur les composants utilisés (par exemple pour chaque pièce d'une automobile sa densité, son coefficient de dilatation, son élasticité, etc …).

Les outils de la simulation numérique

A ces ingrédients, il faut ajouter les moyens d'acquérir ces données, de valider les modèles mathématiques utilisés et lorsque c'est possible de vérifier les résultats des simulations.

En amont d'une simulation numérique, les physiciens établissent les équations décrivant les phénomènes à analyser. Ces équations ne pouvant être résolues « à la main », des numériciens les transcrivent sous forme adaptée au calcul par ordinateur. Le plus souvent, cela consiste à découper le domaine de calcul en un grand nombre de petites zones qu'on appelle des mailles. Les équations mises en forme doivent permettre de calculer comment chaque maille va évoluer sous l'influence de ses voisines durant un court intervalle de temps (ou pas en temps). Le problème est alors dit « discrétisé ». La précision du calcul augmente en général avec le nombre de mailles et donc avec la puissance de la machine.

Une simulation menée à son terme génère ainsi une très grande quantité de données (par exemple la densité, la température, … à chaque intervalle de temps et dans chaque maille), qu'il ne serait pas possible d'exploiter sous la forme de simples colonnes de chiffres. Il faut donc également disposer d'outils spécifiques permettant de mettre en forme et de visualiser les résultats pour pouvoir les interpréter.

La simulation numérique : un outil idéal ?

La qualité d'une simulation numérique est bien sûr liée à la qualité des « ingrédients » évoqués plus haut.

Sa fiabilité dépend de la qualité des moyens de validation, globale ou par parties. Du fait de la complexité des problèmes traités, il est assez difficile d'évaluer le degré de précision d'une simulation. C'est un domaine qui fait l'objet de recherches tout comme l'amélioration des modèles et des algorithmes.

La composante humaine a aussi une importance. Un risque pourrait être que les utilisateurs, chercheurs ou ingénieurs, considèrent que les résultats de leurs simulations « sont » la réalité. Leur « savoir-faire » est donc aussi un aspect essentiel.

L'enjeu de la simulation numérique

La simulation numérique est aujourd'hui indispensable :

- pour la recherche par l'apport à la compréhension des phénomènes complexes ;
- pour les entreprises par le maintien de leur compétitivité (des produits conçus plus vite et moins cher).

Les progrès dans le développement des superordinateurs (HPC) sont en adéquation avec ces demandes et permettent d'y répondre (la puissance des machines est multipliée par 10 tous les quatre ans, pour le même coût).

La création d'un pôle de compétence en simulation haute performance, Teratec, apporte les spécialistes nécessaires à la modélisation des phénomènes et au fonctionnement efficace des superordinateurs.

Source :http://www.guideinformatique.com/fiche-simulation_numerique-779.htm (07/06/2012)