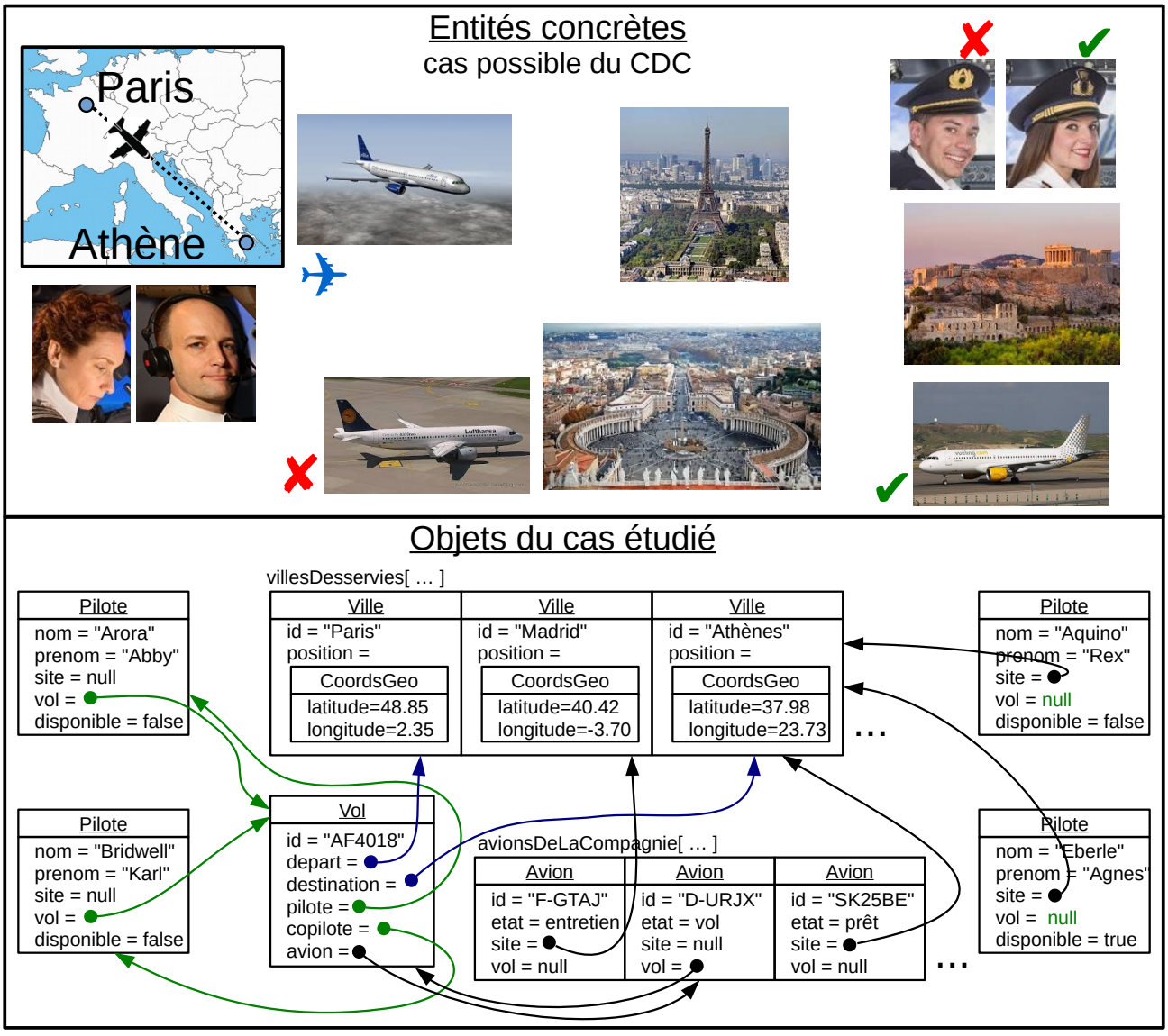
TD/TP 1

Données structurées, Diagrammes objets, Sémantique par valeur / par référence

Objectifs, méthodes

À partir de l'analyse des besoins spécifiés par un CDC (Cahier Des Charges) la découverte des classes qui vont constituer les types d'une solution informatique est au cœur de la programmation orientée objet. Les objets de ces classes entretiennent des relations : les objets sont comme des « paquets d'informations » et un certain type de paquet peut contenir des données élémentaires et/ou des sous-paquets d'informations (composition) et/ou des pointeurs vers d'autres paquets (référence). Cette analyse du CDC peut commencer par la description de situations concrètes (souvent simplifiées) qui devront être traitées par la solution informatique : on y retrouvera donc des **instances** avec des valeurs spécifiques représentatives de cas.



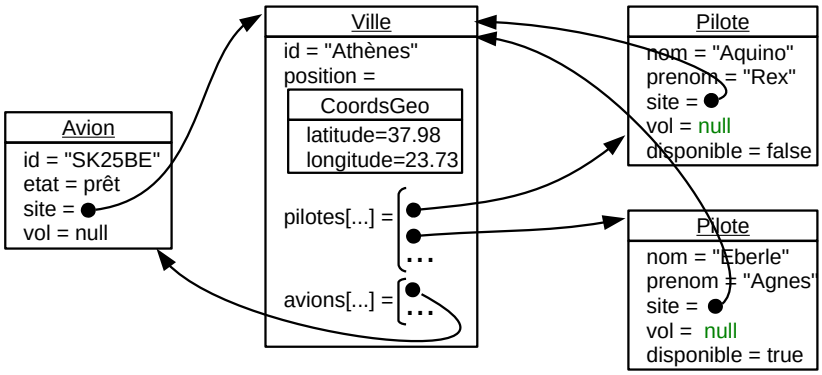
Pour l'instant on ne s'intéresse pas encore aux méthodes des objets (traitements associés aux données) mais seulement aux données et à leur structure. Les choix qui ont conduit à ce schéma particulier dépendent de nombreux aspects rédigés du CDC, les objectifs, le périmètre, les cas d'utilisation de la solution informatique à développer :

- Les objectifs : la mission à remplir est de développer un système d'information (logiciel, base de données...) permettant à une compagnie aérienne de planifier/adapter des vols en fonction des pilotes et avions disponibles. Le système devra permettre de ne pas proposer des vols qui seraient impossibles et de minimiser les transports inutiles de pilotes non affectés à des vols, ainsi que d'optimiser la gestion des avions (transferts et maintenance)
- Le périmètre : sont concernés par le système : les pilotes, les avions, les destinations desservies, les planificateurs de la compagnie et le logiciel de planification existant. Les pilotes interagissent avec le système en confirmant l'arrivée à destination et en indiquant leur disponibilité (arrêts maladie, repos, grèves...). Les planificateurs interagissent avec le système en renseignant les listes d'avions et leur données (état, site…) les listes de pilotes et la liste des vols prévus par le système de planification existant.
- Les cas d'utilisation/fonctionnalités : en concertation avec les parties prenantes concernées (clients du système, décideurs, techniciens, futurs utilisateurs) on a établi une longue liste de cas d'utilisation :
    1. Je suis dans le service maintenance de la compagnie, je veux savoir où envoyer la pièce de rechange nécessaire pour réparer un avion qui est en entretien.
    2. Je suis pilote, hier c'était mon anniversaire et j'ai trop bu : je me suis signalé comme indisponible. Aujourd'hui ça va mieux, je veux indiquer que je suis de nouveau dispo. Un objet informatique **représente** une entité concrète (un véhicule, un vol, un pilote), il faut décider des façons de synchroniser l'information (entrées/sorties)
    3. Je suis planificateur, je veux vérifier la possibilité d'un vol au départ d'une ville, doivent se trouver dans cette ville : un avion prêt et 2 pilotes disponibles.
    4. etc... (probablement des centaines de pages passionnantes !)

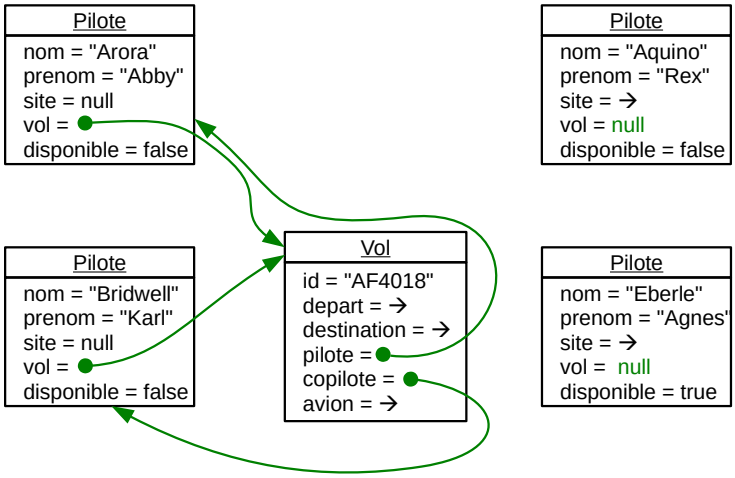
Les **diagrammes objets** permettent de vérifier qu'on a bien compris ce qu'il fallait modéliser et que le modèle répond bien aux contraintes du CDC (ni plus ni moins) et permettra de réaliser de façon efficace les fonctionnalités. Les classes s'identifient aisément, ainsi que les attributs. Les oublis ou maladresses éventuels peuvent être corrigés. Par exemple le modèle objet **proposé** à la page précédente permet de répondre aux fonctions 1. et 2. du CDC mais difficilement au 3.

1. L'avion à réparer est le F-GTAJ : il suffit de suivre la référence de son attribut *site* pour trouver un objet qui m'indique la ville où envoyer la pièce de rechange : Madrid.
2. Je suis le pilote Aquino Rex, je me connecte sur le portail Web sécurisé de la compagnie et j'accède à l'objet qui me représente informatiquement pour changer l'attribut *disponible*.
3. Au départ d'Athènes on a bien les informations qu'il y a un avion disponible SK25BE, et 2 pilotes mais un seul disponible et donc on peut en conclure qu'en l'état un nouveau vol départ Athènes ne peut pas être programmé. Mais à partir de l'objet Ville on ne peut pas accéder à ces informations. Pour l'avion il faut parcourir la liste avionsDeLaCompagnie, en se demandant à chaque fois si il est dans la ville de départ qui nous intéresse (F-GTAJ à Athènes ? Non. D-URJX à Athènes ? Non. SK25BE à Athènes ? Oui ! → on vérifie qu'il est prêt etc...). Ça marche mais ce n'est pas efficace : c'est acceptable ou pas selon qu'on a quelques dizaines d'entités ou 10^3 ou 10^6 et selon la fréquence de cette requête. Par contre dans le modèle proposé les objets pilotes ne sont pas regroupés dans une liste ou un tableau, on ne peut donc pas parcourir tous les pilotes ! Il s'agit d'un oubli ou d'une omission à corriger, évidemment l'ensemble des pilotes doivent être référencés. Reste que ce cas

d'usage serait non performant. Si c'est trop impactant il faudra modifier le modèle des objets Ville en permettant au système d'avoir directement l'information suivante : quels sont les pilotes et les avions dans une ville ? Ceci revient à ajouter aux objets de la classe Ville un attribut « liste de références vers pilotes » et un attribut « liste de références vers avions » :



Il n'est pas nécessaire de reprendre la totalité du diagramme précédent. En fait on constate que même un « cas d'usage » relativement simple peut conduire à un diagramme objet embrouillé et peu lisible : ce n'est pas choquant, les situations réelles modélisées correspondent souvent à des réseaux d'entités en interactions complexes. On s'efforcera d'éviter les croisements des flèches dans la mesure du possible. On peut extraire une partie du réseau pour se concentrer sur un aspect, par exemple on peut extraire du diagramme initialement proposé seulement ce qui concerne les relations entre pilotes et vols (en vert) :



Ceci permet de voir qu'un pilote peut référencer un site ou référencer un vol mais pas les 2 à la fois. Et on ne peut pas avoir une seule référence « situation » qui pointe des fois sur un Vol des fois sur une Ville, pour la même raison qu'on ne peut avoir un pointeur sur 2 types distincts : concrètement les références seront implémentées sous formes de pointeurs. Une référence qui n'a pas d'objet à pointer peut être indiquée par null (le pointeur NULL en C, nullptr en C++).

Les diagrammes d'objets sont donc des croquis « jetables »: ils présentent des vues partielles, sur des cas concrets, et permettent de valider ou de rejeter un certain nombre d'hypothèses sur la façon de structurer les données en objets et en relations entre objets. Il ne faut pas hésiter à les corriger, les étendre, multiplier les cas particuliers jusqu'à vérifier l'ensemble des fonctionnalités du CDC en matière d'informations nécessaires et d'accessibilité à ces informations.

Finalement on pourra extraire de ce travail préliminaire un **diagramme de classe** stable qui servira de référence pour l'implémentation en langage de programmation (C++, Java, C# …). Les diagrammes de classe sont abordés en détail au cours 2 et TD/TP 2.

Consignes, conventions

Dans ce TD/TP nous allons **travailler sur papier** ou éventuellement sur **tablette avec stylet** mais en tout cas **à main levée**. Un diagramme dessiné en 3 minutes avec un crayon prend facilement 30 minutes à faire "au propre" avec des outils point & click… On veut aller vite pour explorer des schémas et de nombreuses alternatives : privilégiez la productivité et la fluidité, rayez, corrigez, refaites en tout ou partie, c'est un travail de type croquis préparatoires. Les exercices proposés ici s'inscrivent dans une continuité, les diagrammes objets du TD/TP1 serviront de base aux diagrammes de classe du TD/TP2 qui seront utilisés lors du passage au code C++ (implémentation) lors des TD/TP 6 et 7 : **conservez soigneusement vos études papiers ou fichiers graphiques tablettes et n'oubliez pas de les rapporter de séance en séance.**

Les diagrammes d'objets demandés seront réalisés selon les conventions illustrées précédemment. Dès qu'elles seront identifiées les classes correspondantes seront indiquées dans le cadre haut de chaque objet. La convention UML indique que sur un diagramme d'objets le nom de la classe est souligné. *La convention en UML comme en C++ est que les noms de classe sont toujours en majuscule (Avion, Pilote …). Les noms des attributs commencent toujours par une minuscule, on peut utiliser le camelCase pour les noms composés (monAttributSuperImportant).*

→ Les données élémentaires sont les nombres réels (à virgule), entiers, booléens (valeur *false/true*), ou énumérations (1 valeur parmi n, par exemple *prêt* / *vol* / *entretien*) et chaînes de caractères.

→ Les données composées sont des attributs qui correspondent à une « sous-structure » (en C : champ de type structure dans une structure). On les représentera comme étant une boite objet **dans** la boite objet qui les utilise (voir l'attribut *position* de type CoordsGeo sur l'exemple).

→ Les données référencées sont des attributs qui correspondent à des pointeurs (en C : champ de type pointeur sur structure dans une structure). On les représentera par des **flèches** partant **précisément de l'attribut** et arrivant sur **l'extérieur de la boite englobant l'objet cible**. Le point d'arrivée sur l'objet cible n'a aucune importance (on ne pointe pas un attribut de la cible).

→ Enfin les collections de données (listes et/ou tableaux à une dimension) seront schématisées par un alignement d'objets de même type, collés côte à côte, et désignés par un nom de collection ou un nom d'attribut avec […] (voir le « tableau » villesDesservies[...] sur l'exemple) et … pour indiquer que la liste d'objet peut grandir. On ne fera pas de distinction entre tableau et liste à ce stade (pour l'essentiel les conteneurs du C++ feront abstraction de ce « détail »). Si le nombre d'élément est fixe et connu on peut le préciser (par exemple attribut notes[3] = 12.5 17 14.5) Pour une collection de références à d'autres objets (en C: tableau de pointeurs) on pourra aligner une séquence de départs de flèches (voir objet « Athènes » figure en haut page précédente).

Avertissement : La distinction entre diagramme de classe et diagramme d'objets est aussi importante que la distinction entre classe et objet, et vous comprenez bien qu'il est aussi important de distinguer entre « humain » et « Charles Darwin ». Ne confondez pas ! Mais ce n'est pas toujours aussi simple, par exemple « scientifique » peut être une valeur d'attribut énuméré pour une enquête d'opinion, ou un objet de la classe Profession si on modélise les profils de rémunération dans la société, ou une classe spécialisée si on modélise l'histoire des grands hommes. Tout dépend du CDC, des objectifs, du périmètre, des fonctionnalités.
Il est d'usage dans un cours d'introduction à la conception orientée objet de commencer tout de suite à travailler avec des diagrammes de classe. Il est certain qu'avec l'habitude on passe « directement » de l'énoncé (CDC) au diagramme de classe sans l'étape diagramme d'objets. Les diagrammes de classe sont plus lisibles et ne dépendent pas de valeurs illustratives arbitraires mais sont aussi plus abstraits. Je souhaite commencer ce module par une approche plus concrète afin de donner des bases solides aux discussions.

Afin que vous sachiez où nous allons : les conventions demandées au TD/TP 1 sont un mix expérimental entre la norme UML officielle pour les diagrammes d'objets et les conventions que nous avons pris l'habitude d'utiliser en ING1 pour représenter des « schémas mémoire » de données structurées, en particulier l'usage des flèches pour indiquer des pointeurs. Nous pourrons continuer d'utiliser cette notation hybride lors des prochaines séances en fonction de son mérite, notamment si un diagramme de classe paraît trop abstrait et qu'on veut le concrétiser par un exemple. Pour les diagrammes de classes nous essaierons de coller au plus près à la norme UML 2. Finalement je tiens à vous féliciter pour votre persévérance et vous promets de ne pas vous demander de lire 6 pages d'intro technique avant d'attaquer le 1er exo du TD/TP à chaque séance !

Fil conducteur : Maillage 2D triangulé

Le fil conducteur est une thématique que vous retrouverez aux TD/TP 1, 2, 6, 7. Du code auxiliaire sera fourni pour exporter des résultats sous forme de fichiers graphiques au format SVG visualisables dans un navigateur ou éditables dans un éditeur de dessin vectoriel. Le fil conducteur part du CDC initial suivant :

On souhaite mettre en place une bibliothèque de représentation et de manipulation de maillages 2D triangulés. La bibliothèque sera écrite en C++ et permettra de travailler avec 1 ou plusieurs maillages, chaque maillage étant un ensemble de triangles dans un repère 2D cartésien. Cette bibliothèque pourra être utilisée comme base de manipulation pour des logiciels C++ de dessin vectoriel à vocation créative (figures de gauche et du centre) aussi bien que comme auxiliaire d'édition de maillages pour des logiciels techniques, en particulier des simulateurs par « éléments finis » et des systèmes d'information géographique (figure de droite).

Quelques objets typiques qu'on manipulera sont donnés à titre indicatif sur les 3 images ci-dessus : nous souhaitons en général travailler avec des maillages compacts et « connexes », c'est à dire qu'un maillage avec des triangles disjoints comme sur le bas du cou du perroquet (figure du milieu, un seul objet maillage) sont exceptionnels : la plupart des triangles d'un même maillage partagent entre eux des sommets et des arêtes communes. Le modèle objet proposé devra permettre de représenter efficacement ces maillages, d'exporter/importer ces maillages vers/depuis un format de fichier (à concevoir, plus tard). Pour l'instant on considérera un seul objet maillage à la fois.

La bibliothèque devra permettre de travailler avec des sélections de sommets : un nombre arbitraire de sommets du maillage sont dans la sélection. À un moment donné il n'y a qu'une seule sélection, la sélection peut être « vide » (aucun sommet sélectionné). On peut ajouter/enlever un/des sommet(s) de la sélection en le(s) désignant avec un cadre englobant dans le repère cartésien. Le groupe de sommets sélectionnés peut être translaté, dilaté/contracté, tourné, retourné en miroir (toutes les transformations affines du plan).

Il est possible d'ajouter de nouveaux sommets, un par un. Les sommets ont des coordonnées réelles, le repère utilisé sera un repère infographique orthonormal ***indirect*** *avec l'axe des abscisses de gauche à droite et l'axe des ordonnées de haut en bas. À la création d'un nouveau sommet on spécifie ses coordonnées. À un moment donné il peut exister 0, 1 ou n sommets qui ne sont pas (ou pas encore) utilisés par des triangles : sommets isolés. Ajouter un nouveau sommet au maillage commence donc toujours par ajouter un sommet isolé. Quand exactement 3 sommets sont dans la sélection et si ces 3 sommets ne correspondent pas déjà à un triangle alors on dispose d'une opération d'ajout d'un nouveau triangle au maillage. A l'inverse une opération permet de supprimer le(s) triangle(s) dont les 3 sommets se trouve dans la sélection, cette suppression de triangles ne supprime pas les sommets (éventuellement certains sommets se retrouvent isolés). Enfin on a aussi une opération pour supprimer les sommets de la sélection, les triangles qui ont au moins un sommet supprimé sont supprimés, et la sélection se retrouve vidée.*

Chaque triangle a une couleur caractérisée par un triplet d'entiers (rouge, vert, bleu, chacun dans l'intervalle 0 .. 255). On pourra préciser la couleur du(des) triangles lors de leur création, ou avoir une couleur par défaut (255, 255, 255) et donner une nouvelle couleur aux triangles de la sélection (dont les 3 sommets sont dans la sélection). Le fait que le maillage est représenté avec ou sans les arêtes surlignées en segments noirs ou que les sommets sont représentés ou pas par un petit disque noir sont des paramètres des méthodes de dessin : ils ne font pas partie des données du modèle et il ne sera pas demandé d'avoir certains triangles surlignés et d'autres pas. En revanche les méthodes de dessin, qui doivent pouvoir afficher (ou exporter en format SVG) les triangles (en parcourant tous les triangles d'un maillage) doivent pouvoir surligner ou afficher d'une façon particulière les sommets de la sélection.

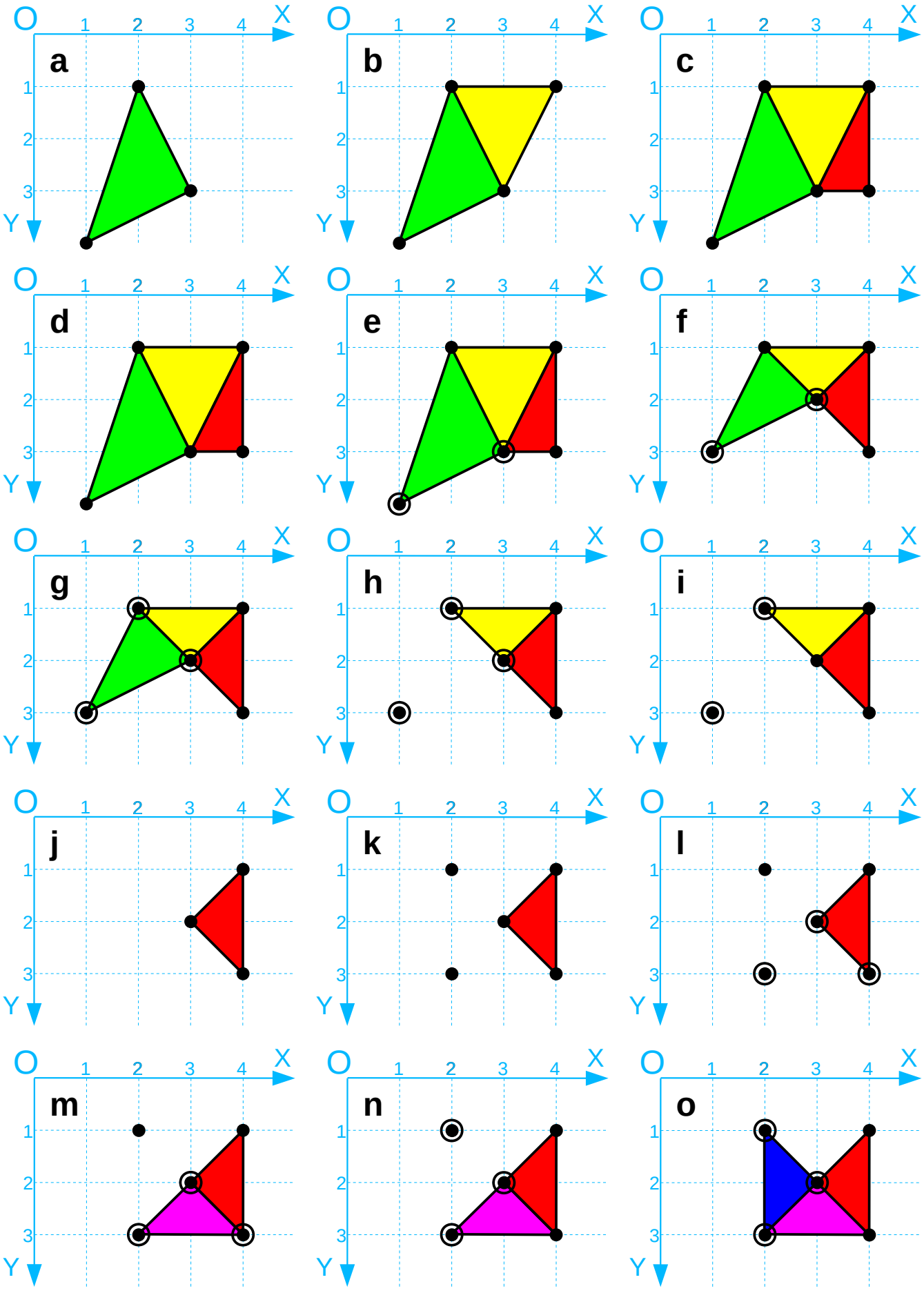
Le code C++ de la bibliothèque sera utilisable par des appels de méthode aux objets concernés. On prévoira des protocoles de test pour valider les fonctionnalités sous forme de séquences d'utilisation des objets (tests codés en dur en tant que code client). Dans un 2ème temps une interface expérimentale permettra de jouer interactivement avec les fonctionnalités de la bibliothèque uniquement à partir des flots standards de la ***console*** *cin/cout. Pour simplifier les tests en mode console il sera possible de désigner les sommets (sélections) en entrant des indices. La réalisation d'une interface interactive graphique/souris ne fait pas partie du contrat de ce CDC : cette tâche est déléguée aux futurs développeurs intégrant la bibliothèque dans leurs propres projets. Dès que des fonctions de chargement seront disponibles ont testera des modèles comportant jusqu'à plusieurs millions de triangles, et on comparera les performance avec une bibliothèque concurrente, tant sur les temps de traitement que sur la quantité de mémoire utilisée.*
[il s'agit d'un énoncé fictif, nous n'allons pas réellement nous mesurer à la concurrence!]

1. Défricher les classes, les attributs, les objets ou collections d'objets

À partir de la lecture du CDC repérez ce qui vous semble devoir être des classes et leurs attributs. Organisez vos classes avec un cadre en haut indiquant le nom que vous donnerez à la classe (1ère lettre en majuscule) et un cadre en dessous avec les attributs des objets de cette classe (1ère lettre en minuscule). A ce stade vous pouvez encore douter, c'est une phase préliminaire, ajoutez des points d'interrogation à coté des éléments dont vous n'êtes pas sûr en précisant vos doutes par écrit.

Par exemple, on peut indiquer un attribut booléen sélectionné (oui/non) dans une classe Sommet. Possible, mais est-il obligatoire que sélectionné soit un attribut, peut être qu'on a plutôt une collection de références aux sommets qui sont dans la sélection ? Qu'en pensez-vous ? À ce stade on manque encore de visibilité, inutile de figer des décisions trop tôt. Mais on peut déjà être à peu près sûrs qu'il y aura au moins une classe Sommet et une classe Triangle !

Cas concrets à étudier (voir énoncé page suivante)



2. Diagrammes d'objets, cas simple a b c

Testez vos hypothèses de structuration des données sur les cas concrets proposés à la page précédente. Commencez par représenter vos objets selon les conventions demandées pour la situation **a** puis augmentez le diagramme avec les objets supplémentaires qui arrivent dans les situations **b** et **c**.

*Testez cette séquence **a b c** d'évolution des objets selon que les triangles sont **composés** de sommets ou que les triangles **référencent** des sommets : au moins 2 diagrammes à dessiner*

- **Composition** : les objets sommets sont représentés par des boites de données **dans** la boite des objets triangle. Comme il y aura 3 sommets par triangles on aura donc trois boites sommets dans chaque boite triangle. On dira qu'on met en place une **sémantique par valeur** puisque c'est l'objet triangle qui héberge **les valeurs des objets sommets**. Concrètement le poids en octets d'un objet triangle sera 3 fois le poids de chaque objet sommet qu'il contient (plus le poids de ses autres attributs). Un objet qui n'utilise que des attributs par valeur est facile à séparer/isoler du reste du modèle, il n'est pas « rattaché par des fils », il est facile à dupliquer, facile à sauver/charger sur un fichier.
- **Référence** : les objets sommets sont représentés par des boites de données **à l'extérieur** de la boite des objets triangle. Comme il y aura 3 sommets par triangles on aura donc dans chaque boite triangle trois départs de flèches vers 3 boites sommets à l'extérieur. On dira qu'on met en place une **sémantique par référence** puisque l'objet triangle **n'héberge pas les valeurs des objets sommets, seulement leur adresse**. Concrètement le poids en octets d'un objet triangle sera 3 fois le poids d'un pointeur (plus le poids de ses autres attributs). Un pointeur pèsera toujours 4 octets (compilation 32 bits). Un même objet peut être référencé par plusieurs autres objets : **plusieurs triangles pourront référencer un même sommet**. Un objet qui utilise au moins un attribut par référence n'est pas facile à séparer/isoler du reste du modèle, il est « rattaché par des fils », il n'est pas facile à dupliquer (faut il dupliquer les objets référencés ?), et pas facile à sauver/charger sur un fichier : on ne peut pas restaurer un pointeur directement en partant d'une adresse sauvée car les adresses de stockage en mémoire vive sont décidées par le système d'exploitation et changent à chaque nouvelle exécution.

La distinction entre ces 2 façons d'associer des objets est essentielle, prenez le temps de bien comprendre les conséquences de ce choix en comparant les 2 schémas distincts. Combien d'octets pèse la situation **c** dans chaque approche ? Quels sont les autres éléments du CDC qui vont nous faire préférer l'un à l'autre ? Peut-on trancher à ce stade ? *Discutez.*

3. Diagramme d'objets, le « point de vue mathématique »

Un collègue du département mathématiques assiste par curiosité à la réunion où l'équipe discute des mérites de tel ou tel modèle objet de représentation de nos maillages. Il suggère alors d'aller « au plus simple de son point de vue » : un maillage est composé de N triangles, un triangle est composé de 3 segments, un segment est composé de 2 sommets, un sommet est composé de 2 coordonnées réelles. *Pensez vous que ce soit une bonne idée ? Pourquoi ? Dessiner & rédiger !*

4. Diagrammes d'objets, séquence d'utilisation du modèle d e f … o

Choisir le modèle qui semble le plus adapté compte tenu des analyses précédentes. On va vérifier qu'il est compatible avec les exigences fonctionnelles du CDC. On repart de la situation **d** (identique à la **c**) et on déroule des opérations qui sont possibles d'après le CDC :

e → sélection de 2 sommets
f → translation vers le haut des sommets sélectionnés
g → ajout d'un sommet dans la sélection
h → suppression de triangle(s) qui ont 3 sommets sélectionnés (ici il n'y a que le vert)
ici on obtient un sommet « isolé », la sélection reste la même
i → enlever un sommet de la sélection
j → suppression des sommets de la sélection : les sommets disparaissent et les triangles qui utilisaient au moins un des sommets supprimé disparaissent. La sélection est vide.
k → en 2 opérations élémentaires on crée successivement 2 nouveaux sommets
les nouveaux sommets sont initialement des sommets « isolés »
l → sélection de 3 sommets
m → création d'un nouveau triangle sur la base des 3 sommets de la sélection
n → enlever un sommet de la sélection, ajout d'un sommet dans la sélection
o → création d'un nouveau triangle sur la base des 3 sommets de la sélection

Les triplets (rouge, vert, bleu) de couleur sont vert (0, 255, 0) jaune (255, 255, 0) rouge (255,0,0) magenta (255,0,255) et bleu (0,0,255)

Dessiner les diagrammes d'objets compte tenu de vos choix. Il n'est pas nécessaire de faire un nouveau schéma à chaque étape mais repartez sur un nouveau schéma quand ça devient illisible. Vérifiez qu'à chaque étape on dispose bien des informations nécessaires aux opérations, qu'on navigue bien les références dans le bon sens (on accède bien aux données en suivant les flèches à l'endroit, pas à l'envers), que la méthode de dessin disposera des informations nécessaires pour dessiner la figure (y compris la sélection) en parcourant la collection des triangles. Corriger le modèle si il ne convient pas, recommencer la séquence avec les corrections si nécessaire. Comparer votre modèle obtenu finalement avec celui des autres. Un consensus peut-il se dégager ?

5. Format de fichier (facultatif au TD/TP 1)

Si vous avez encore un peu d'énergie vous pouvez commencer à réfléchir à la façon d'enregistrer un maillage dans un fichier. *A quoi ressemblerait le fichier décrivant le maillage **c** ? Ce format est-il directement compatible avec le modèle objet des données en cours d'utilisation par l'application ?*

Fait avec les outils de la suite bureautique LibreOffice
(application écrite en C++/Java/Python, que de l'objet !)