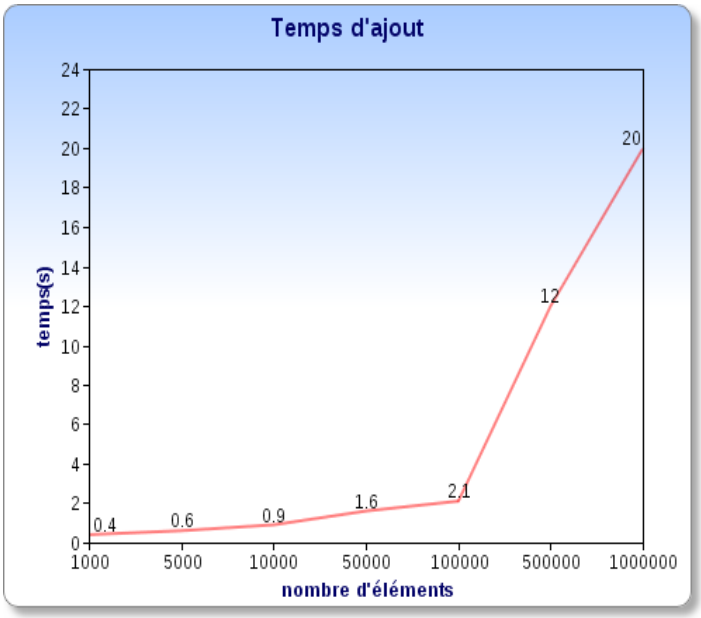
Document d'évaluation :

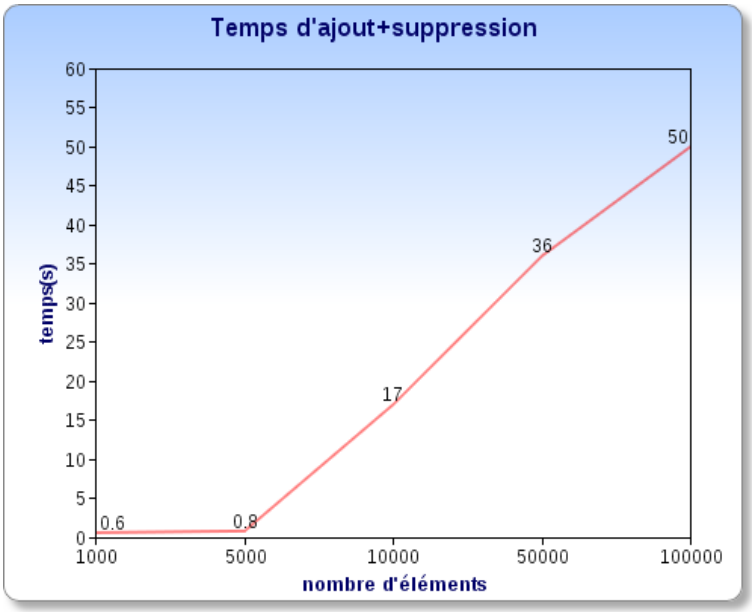
Pour tester les performances de notre application, nous avons choisi de calculer les temps d'exécution des opérations qui demandent le plus de temps d'exécution: L'ajout, la suppression des éléments un par un. Ces manipulation sont également représentatives du reste des manipulations des éléments géométriques que permet notre application(déplacement, chargement du modèle dans un fichier, chargement d'un modèle depuis un fichier) puisque il s'agit des mêmes opérations :

- Recherche dans un dictionnaire avant d'ajouter un élément pour s'assurer qu'il n'est pas déjà présent.
- Recherche dans un dictionnaire pour vérifier que l'élément qu'on veut supprimer existe pour le supprimer.

La recherche dans le dictionnaire étant logarithmique, les optimisations que nous avons essayé de mettre en place consistaient à :

- Eviter de transmettre les paramètre par valeur pour éviter les copies inutiles(pour les commandes qui prenaient un pointeur sur le modèle en paramètre, pour les vecteurs contenant les différentes informations à transmettre aux constructeurs commandes puis aux constructeurs des différents éléments géométriques puis aux différentes méthodes .
- Essayer de rendre les recherches dans nos collections les plus profitables possibles, pour ne pas faire les même recherches plusieurs fois. Notre logique était de regrouper toutes les vérifications syntaxiques sur les commandes dans l'application, car elle étaient sans relation avec le modèle. Dans les commandes, nous avons regroupé les vérifications liées au modèle (présence de l'élément, appartenance à des objets agrégés...) . Mais il y a certaines recherches qui sont redondantes, soit parce que nous nous étions rendu compte qu'une vérification manquait après, soit parce que nous ne voyions pas une autre manière de faire. (particulièrement en ce qui concerne les éléments appartenant aux objets agrégés, ou la suppression requiert de les supprimer dans leur objets agrégés, puis le undo requiert de les ajouter dans le modèle, puis dans les objets agrégés auxquels ils appartenaient. Cette redondance pourrait expliquer les résultats de nos tests de performance.





Les deux tests ci-dessus ont été réalisés sur plusieurs éléments incluant des objets agrégés. Pour le deuxième tests, tous les ajouts ont d'abord été faits, avant de faire toutes les suppressions, afin d'augmenter la taille du modèle. On remarque que la complexité est en $o(n^2)$, ce qui veut dire que toutes les vérifications annexes ont influencé les performances de ces deux opérations.