Prémilinaires

- Organisez-vous en binômes. Maximum un(e) seule(e) isolée(e) dans toute la promo ! Les binômes peuvent rester identiques à ceux de la semaine dernière. Ou pas, mais on doit commencer à converger vers ceux du projet, qui se déroulera sans monôme.
- Tous les nouveaux programmes « en cascade » ci-dessous peuvent être écrits dans un même fichier .c (et son compagnon .h) et appelés par une option -1, -2, -3, de la ligne de commande *via* la même et unique routine `main`. Les routines écrites pour une question doivent pouvoir être réutilisées par les suivantes.
- Commencez à vous entraîner à doxygen-commenter le code et voir ce que ça donne.

1. Finir le TP1.

2. Génération/stockage. Écrivez un programme pour créer un polygone dont les sommets sont définis à l'aide de la souris (fin de saisie quand on pointe à nouveau sur le premier sommet, à quelques pixels près), et qui enregistre ceux-ci, sans répétition, dans l'ordre de saisie, dans un fichier de nom indiqué par une option sur la ligne de commande, p. ex. : '`tpga2 -1 -opoly.txt`'. [*o pour output*]. Format du fichier = nbpoints de la frontière (entier) suivi de la liste des couples de coordonnées séparés par une virgule, en double. Remarque : sans aller jusqu'à afficher en continu le segment courant suivant le déplacement de la souris, affichez le segment courant une fois défini (ainsi que tous les précédents), c'est suffisamment satisfaisant pour l'utilisateur.
3. Dégénérescence/simplicité. Écrivez un programme qui ouvre un des fichiers créés précédemment (par exemple '`tpga2 -2 –ipoly.txt`' sur la ligne de commande [*i pour input*]), vérifie qu'il a plus de deux sommets distincts et non tous alignés, affiche le polygone correspondant puis teste s'il est simple (pas d'auto-intersections[1]) ou non, en affichant dans une couleur différente les couples d'arêtes fautives. Cerise sur le polygone : affichage particulier des points d'intersection ? (si pas plus de 5mn de travail). Retour de la fonction : 1 si le polygone est non-dégénéré (a strictement plus de 2 points, non tous alignés) et simple (pas d'auto-intersections), 0 sinon.
4. Convexité locale. Écrivez un programme qui, après avoir testé que le polygone est non-dégénéré et simple, l'affiche et détermine s'il est donné dans le sens trigonométrique et le réoriente dans ce sens si besoin, puis donne une couleur différente aux couples d'arêtes successives sur la frontière ne formant pas un angle convexe. Retour : 1 si convexe, 0 sinon.
5. Inclusion convexe. Écrivez une routine qui détermine en temps logarithmique si un point requête donné (à la souris au début de a recherche de solution, lecture par fichier ensuite) est intérieur ou extérieur à un polygone convexe donné (affichage comme dans le TP1). Évaluez le temps de réponse moyen sur plusieurs milliers de requêtes après le pré-traitement initial (aussi chiffré).

[1] Pour tester si deux arêtes AB et CD s'intersectent ou non : utiliser uniquement le prédicat orientation. Comment faire ? Réponse en séance…