TP 5 - TP noté

Ce TP noté est divisé en deux parties. Pour des raisons pratiques vous travaillerez uniquement sous linux (et de préférence sur les machines de l'IUT). Une archive contenant le code à compléter est disponible sur ma page à l'adresse suivante :

`http://www.lmpa.univ-littoral.fr/~fromentin/teaching/2019/dut/tp-note.tar.gz`

Renommer le répertoire `nom-prenom` avec vos nom et prenom. C'est ce répertoire complété que vous me donnerez à l'issu du TP noté.

Attention. N'oubliez pas de faire des copies de sauvegardes régulièrement avant de ne pas tout perdre suite à une fausse manipultation.

1. Courbes de Bézier

Le but de cette partie est de tracer une courbe de Bézier cubique à l'aide de la librairie graphique `SDL2`. Elle est à réaliser dans le répertoire `bezier`.

Exercice 1. Nous allons commencer par créer une classe `Point` permettant de représenter des points à coefficients entiers. Cette classe sera codée dans les fichiers `point.hpp` et `point.cpp`.

a. Coder un constructeur prenant deux entiers x et y en paramètre et créant le point de coordonnées (x, y).

b. Coder deux accesseurs `lire_x` et `lire_y` retournant l'abscisse (resp. l'ordonnée) du point courant

Les fonctions qui suivent ne sont pas des fonctions membres de la classe `Point` et sont donc à coder à l'extérieur de la class `Point`.

c. Surcharger l'opérateur `operator<<` pour permettre d'afficher des objets de la classe `Point`.

d. Créer une fonction `milieu` qui étant donnés deux points A et B retourne le milieu du segment $[AB]$

e. Créer une fonctions `distance` qui étant donnés deux points A et B retourne la distance $d(A, B)$ entre A et B.

À partir de maintenant vous n'aurez plus que le fichier `main.cpp` à modifier.

Exercice 2. Les courbes de Bézier cubiques sont créer à l'aide de quatre points quelconques. Ces points seront directement positionnés à l'aide la souris. Vous devez modifier le code de la fonction `main` pour obtenir le comportement suivant :

- Dès que l'utilisateur clique dans la fenêtre on enregistre la position de la souris dans une variable de type `Point`.
- Si quatre points ont été créés de cette façon afficher le message "Dessin courbe de Bézier"
- Si l'utilisateur clique une cinquième fois afficher le message "Efface courbe de Bézier" et considérer ce nouveau point comme le premier d'une nouvelle courbe de Bézier

Exercice 3. Nous allons maintenant dessiné les points définissant la courbe de Bézier.

a. Écrire une fonction permettant de dessiner un `point` A sous la forme d'un carré de centre A. Les arrêtes du carré seront de longueur $2c + 1$ où c sera un entier positif qu'on déterminera pour avoir un rendu satisfaisant. Vous pourrez tracer le carré de la couleur que vous souhaitez.

b. Afficher les points déjà saisis lors de la construction de la courbe de Bézier à l'aide de la souris

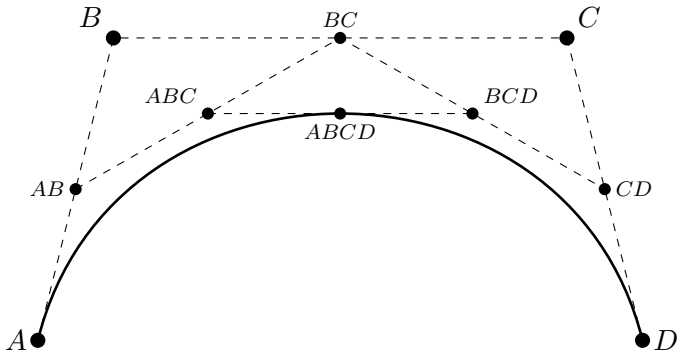
Exercice 4.

a. Ecrire une fonction qui étant donné quatre points dessine une ligne brisée reliant ces points.

b. Modifier le code de la fonction `main` pour que la ligne brisée reliant les 4 points soit afficher lors de l'affichage du message "Dessin courbe de Bézier".

c. Dès que le message "Efface courbe de Bézier" est affiché il faut que la ligne brisée ne soit plus affichée à l'écran.

Nous disposons maintenant de tous les outils nécessires pour tracer la courbe de Bézier à l'aide de l'algorithme de De Casteljau.



Les points AB, BC, CD, ABC, BCD et $ABCD$ sont placés au milieu de leur segement respectif.

On dira qu'une courbe de Bézier définie par les points A, B, C et D est négligeable si toutes les distances $d(A, B)$, $d(B, C)$, $d(C, D)$ et $d(A, D)$ sont inférieurs à 0.1. Le tracer de la courbe de Bézier définie par les points A, B, C, D est

- la ligne reliant A à D si la courbe est négligeable
- obtenu en traçant la courbe de Bézier définie par le points $A, AB, ABC, ABCD$ puis celle définie par les points $ABCD, BCD, CD, D$.

Exercice 5. Ecrire des fonctions permettant de :

a. Décider si quatre points définissent une courbe de Bézier négligeable.

b. Tracer la courbe de Bézier définie par quatre points donnés.

Exercice 6. Finir de tout mettre en place.

2. Pendule oscillant

Dans cette section nous souhaitons modéliser un pendule oscillant à l'aide des méthodes de Runge-Kutta d'ordre 1 et 2 (RK1 et RK2). Rappelons que ces méthode permettent de résoudres numériquement des équations différentielles du type :

$$y'(t) = f(t, y(t)) \quad \text{et} \quad y(0) = y_0 \tag{1}$$

où $y : \mathbb{R}^+ \to \mathbb{R}^d$ est une fonction inconnue, t est une variable parcourant $\mathbb{R}^+$, le vecteur y_0 de $\mathbb{R}^d$ est la condition initiale et $f : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d$ est une fonction connue dépendant du problème à étudier.

Nous rappelons aussi que pour tout intervalle I, une fonction $g : I \to \mathbb{R}^d$ est donnée par

$$g(t) = \begin{bmatrix} g_1(t) \\ \vdots \\ g_d(t) \end{bmatrix} \quad \text{pour } t \in I,$$

où $g_1, \ldots, g_d$ sont des fonctions de I dans $\mathbb{R}$. De plus si g est dérivable sur I alors nous avons

$$g'(t) = \begin{bmatrix} g_1'(t) \\ \vdots \\ g_d'(t) \end{bmatrix} \quad \text{pour } t \in I.$$

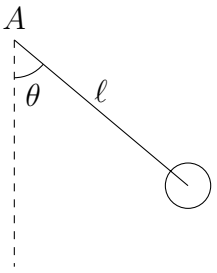
Les méthodes de Runge-Kutta ne donneront pas la fonction y solution de (1) mais donneront des valeurs approchée de $y(t)$ pour certaines valeurs de t. Commençons par fixé un réél strictement positif h supposé petit. Pour tout $n \in \mathbb{N}$, on pose $t_n = n \times h$. Nous souhaitons alors obtenir une valeur approchée a_n de $y(t_n)$ pour tout $n \in \mathbb{N}$.

Pour les deux méthodes, la suite (a_n) est définie par récurrence à partir de $a_n = y_0$ et les formules de récurrence suivantes :

- $a_{n+1} = a_n + hf(t_n, a_n)$ pour la méthode RK1,
- $a_{n+1} = a_n + hf\left(t_n + \frac{h}{2}, a_n + \frac{h}{2} f(t_n, a_n)\right)$ pour la méthode RK2.

Le code demandé sera à écrire dans le fichier `main.cpp` du répertoire `pendule1`

Exercice 7. On considère un pendule oscillant de masse m accroché en A à l'aide d'une tige de longueur ℓ. On note θ l'angle que forme la tige du pendule avec la verticale.



a. Compléter la fonction `dessine_pendule(double theta)` permettant de dessiner le pendule (et sa tige) pour un angle θ donné. Le point A sera le milieu de la fenêtre, la longueur de la tige sera de 200 pixels et le rayon du cercle sera de 30 pixels.

b. Tester votre fonction pour différentes valeurs de θ.

Pour la suite on suppose que l'angle θ vaut $\pi/8$ au temps $t = 0$.

Exercice 8. Dans le cas **sans frottement** l'angle $\theta(t)$ du pendule satisfait l'équation différentielle :

$$\theta''(t) = -\frac{g}{\ell}\sin(\theta(t)) \quad \text{avec} \quad \theta(0) = \frac{\pi}{8} \quad \text{et} \quad \theta'(0) = 0,$$

où g est la constante d'accélération gravitationnelle 9.81 m.s^{-2}. On pose $\omega = \theta'$. L'équation précédente devient alors :

$$\begin{cases} \theta'(t) = \omega(t) \\ \omega'(t) = -\frac{g}{\ell}\sin(\theta(t)) \end{cases} \quad \text{avec} \quad \begin{cases} \theta(0) = \frac{\pi}{8} \\ \omega(0) = 0 \end{cases} \tag{2}$$

Cette équation est la même que celle donnée en (1) en posant $y(t) = \begin{bmatrix} \theta(t) \\ \omega(t) \end{bmatrix}$.

a. Compléter la fonction

```
void f(double t,double theta,double omega,double& dtheta,double& domega)
```

qui étant données `theta` $= \theta(t)$ et `omega` $= \omega(t)$ stocke les valeurs de $\theta'(t)$ et $\omega'(t)$ dans `dtheta` et `domega` respectivement (voir équation (2)).

b. Compléter la fonction

```
void rk1(double t,double& theta,double& omega)
```

qui met à jour les valeurs de θ et ω à l'aide de la méthode RK1.

c. L'animation obtenue est-elle conforme à la réalité ? On pourra tester différentes valeurs de ℓ.

d. Compléter la fonction

```
void rk2(double t,double& theta,double& omega)
```

qui met à jour les valeurs de θ et ω à l'aide de la méthode RK2.

e. Modifier la boucle événementielle afin d'utiliser RK2 à la place de RK1 pour l'animation du pendule.

f. L'animation obtenue est-elle conforme à la réalité ? Quelle méthode vous semble la plus adaptée ? On pourra tester différentes valeurs de ℓ.

Dupliquer votre fichier `main.cpp` contenu dans le répertoire `pendule1` dans le répertoire `pendule2`. À partir de maintenant vous travaillerez dans le répertoire `pendule2`.

Exercice 9. Dans le cas **avec frottement** l'angle $\theta(t)$ du pendule satisfait l'équation différentielle :

$$\theta''(t) = -\frac{g}{\ell}\sin(\theta(t)) - \frac{k}{m\ell^2}\theta'(t) \quad \text{avec} \quad \theta(0) = \frac{\pi}{8} \quad \text{et} \quad \theta'(0) = 0,$$

où g est la constante d'accélération gravitationnelle 9.81 m.s^{-2} et k une constante sans unité correspondant au coefficient de frottement visqueux. La valeur $k = 0$ correspond au cas sans frotement.

a. Que devient le système d'équation (2) dans ce cas.

b. Définir une variable globale k valant 0.5.

c. Modifier la fonction `f` en fonction des nouvelles equations obtenues.

d. Faire des simulations pour différentes valeurs de k.