Université du Littoral **Année 2011-2012**
Licence 2 ème année **Informatique**

Courbes et Surfaces

Courbes de Bézier

Ce devoir consiste à coder un programme permettant de tracer de manière intéractive une courbe de Bézier à l'aide de la librairie SDL.

1. Ce qu'il faut faire

Ce projet est à faire en binôme ou trinôme et à rendre au plus tard le dimanche 6 mai à 23h59.

Vous trouverez sur mon site une archive `bezier.tar.gaz`. Elle contient
- un fichier `Enonce.pdf` qui n'est autre que cet énoncé ;
- un répertoire `Librairie` contentant les sous-répertoires `Linux`, `Mac` et `Windows`. Le code source contenu dans ces répertoires est votre trousse à outils, plusieurs classes et fonctions y sont définies. La première étape technique consiste à compiler et à faire fonctionner le programme contenu dans ce répertoire en fonction de votre plateforme. Les fichiers `Readme.txt` contiennent quelques instructions.
- un répertoire `CopiesEcran` qui contient des copies d'écran illustrant le rendu à obtenir à la section portant sur l'algorithme de De Casteljau

Vous me rendrez avant la date limite, une archive `.zip` ou `.tar.gz` contenant
- un fichier `info.txt` précisant les noms, prénoms et adresses email des membres
- un répertoire `TesteDeCasteljau` qui contiendra plusieurs sous-répertoires. Dans chacun de ces sous-répertoires, vous mettrez le code source d'un programme dessinant une courbe de Bézier ainsi qu'une copie d'écran de ce que vous avez obtenu chez vous. Evidemment les courbes de Bézier seront différentes.
- un répertoire `Interactions` qui contiendra le code source des différents programmes demandé à la dernière section.
- un fichier `rapport.pdf` contenant la réponse aux différents exercices ainsi qu'une description des problèmes rencontrés et des solutions trouvées et/ou testées.

Les différentes fonctions demandées sont à mettre au choix dans les fichiers

`main.cpp`, `bezier.h` et `bezier.cpp`.

Pour rappel, un fichier `.h` ne doit pas contenir le code de la fonction mais juste sa déclaration sous forme de prototype.

Si vous avez des questions ou que vous souhaitez communiquer entre vous, j'ai mis un forum à votre disposition.

2. Primitives de dessin

Nous rappelons que le y de l'écran SDL est inversé par rapport à la direction mathématique. Ainsi, si l'écran est de largeur 640 et de hauteur 480 alors le point en bas à gauche a pour coordonnées $(0, 480)$ tandis que celui en haut à droite a pour coordonnées $(640, 0)$.

Pour cette section, vous aurez besoin des classes `Point`, `Couleur`, `Ecran_SDL` définies dans la librairie.

La classe `Point` permet de représenter les points. L'instruction `Point P(x,y)` crée le point `P` d'abscisse `x` et d'ordonnée `y`. Si `P` est une variable de type `Point` alors `P.x` désigne l'abscisse de `P` tandis que `P.y` désigne son ordonnée. La commande `cout<<P<<endl` affiche les cordonnées du `Point P`.

La classe `Couleur` permet de représenter les couleurs. L'instruction `Couleur c(n)` avec `n` compris entre `0` et `255` crée une couleur grise. La valeur `n=0` donne un noir tandis que `n=255` donne un blanc. L'instruction

`Couleur c(r,g,b)` où `r`, `g` et `b` sont compris entre `0` et `255` crée une couleur ayant une composante rouge égale à `r`, une composante verte égale à `g` et une composante bleue égale à `b`. N'hésitez pas à tester différentes valeurs de `r,g,b` pour comprendre le principe.

La classe `Ecran_SDL` permet de gérer l'affichage SDL. L'instruction `Ecran_SDL(w,h)` crée un écran de largeur `w` et de hauteur `h`. Si `écran` est une variable de type `Ecran_SDL` alors la commande
– `ecran.effacer()` efface l'écran ;
– `ecran.rafraichir()` rafraichit l'écran et provoque l'affichage des modifications ;
– `ecran.dessine_point(P,c)` colorie le pixel correspondant au `Point P` avec la `Couleur C`.

Exercice 1. Ecrire une fonction

```
void dessine_ligne_horizontale(Ecran_SDL ecran, Point P,usint l,Couleur c)
```

qui dessine la ligne horizontale de longueur `l` et de couleur `c` ayant le point `P` comme extrémité gauche.

Exercice 2. Ecrire une fonction

```
void dessine_ligne_verticale(Ecran_SDL ecran, Point P,usint l,Couleur c)
```

qui dessine la ligne verticale de longueur `l` et de couleur `c` ayant le point `P` comme extrémité basse.

Dans la librairie vous trouverez une classe `Carre`. Cette classe permet évidemment de représenter des carrés. Un carré est donné par son centre et la longueur de ces arrêtes. L'instruction `Carre c(Point P,6)` crée le carré centré en `P` et de longueur `6`. Si `carre` est une variable de type `Carre` alors la commande
– `carre.C` retourne le `Point` centre de `carre`,
– `carre.l` retourne la longueur des arrêtes de `carre`.

Exercice 3. A l'aide des fonctions permettant de tracer des lignes horizontales et des lignes verticales, écrire une fonction

```
void dessine_carre(Ecran_SDL ecran,Carre carre,Couleur c)
```

qui dessine en couleur `c` le contour du carré `carre`.

3. Polynômes de Bernstein

Soit n un entier. Pour tout $k \in \{0, ..., n\}$, on appelle coefficient binomial et on note $\binom{n}{k}$ l'entier $\frac{n!}{k!(n-k)!}$.
Il existe $n+1$ polynômes de Bernstein de degré n, noté $B_0^n, ..., B_n^n$ et définie sur $[0,1]$ par

$$B_k^n(x) = \binom{n}{k} x^k (1-x)^{n-k} \quad \text{pour } k = 0, ..., n$$

Exercice 4. Calculer, à la main, les polynômes de Bernstein de degré 3.

Exercice 5. A partir de la relation

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

montrer que l'on a

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i+1}^{n-1}(t), \quad \forall t \in [0,1].$$

Exercice 6. Soient n un entier $\geqslant 1$. Montrer, à l'aide de l'exercice précédent, que pour tout $\lambda_0, ..., \lambda_n$ on a

$$\sum_{k=0}^{n} \lambda_k B_k^n \left(\frac{1}{2}\right) = \sum_{k=0}^{n-1} \lambda'_k B_k^{n-1} \left(\frac{1}{2}\right)$$

où $\lambda'_k = \frac{1}{2}\lambda_k + \frac{1}{2}\lambda_{k+1}$.

4. Courbe de Bézier de degré 3

Soient $A = (x_A, y_A)$, $B = (x_B, y_B)$ deux points du plan et λ, μ deux réels, alors $\lambda A + \mu B$ désigne le point de coordonnées $(\lambda x_A + \mu x_B, \lambda y_A + \mu y_B)$.

Definition 1. Soient A, B, C et D quatre points du plan. On définit la courbe de Bézier $\mathcal{C}_{A,B,C,D}$ comme l'ensemble des points

$$\mathcal{C}_{A,B,C,D}(t) = B_0^3(t)A + B_1^3(t)B + B_2^3(t)C + B_3^3(t)D$$

pour $t \in [0, 1]$.

Exercice 7. Soient A, B, C et D quatre points du plan. Calculer les points de $\mathcal{C}_{A,B,C,D}$ associés aux valeurs 0 et 1 du paramètre t. En déduire que les points A et D appartiennent à $\mathcal{C}_{A,B,C,D}$.

Soient A, B, C et D quatre points du plan. On note AB le point $\frac{1}{2}A + \frac{1}{2}B$. De cette manière, on pose
- $AB = \frac{1}{2}A + \frac{1}{2}B$;
- $BC = \frac{1}{2}B + \frac{1}{2}C$;
- $CD = \frac{1}{2}C + \frac{1}{2}D$;
- $ABC = \frac{1}{2}AB + \frac{1}{2}BC$;
- $BCD = \frac{1}{2}BC + \frac{1}{2}CD$;
- $ABCD = \frac{1}{2}ABC + \frac{1}{2}BCD$.

Exercice 8. Représenter les points AB, BC, CD, ABC, BCD, $ABCD$ sur un dessin lorsque les points A, B, C et D décrivent un quadrilatère quelconque non croisé. Avant de dessiner le point AB, par exemple, on tracera le segment $[A, B]$. De même pour tous les autres points.

Exercice 9. Montrer à l'aide de l'exercice 6 qu'on a

$$\mathcal{C}_{A,B,C,D}\left(\frac{1}{2}\right) = ABCD$$

5. Algorithme de De Casteljau

La méthode de De Casteljau pour tracer des courbes de Bézier de degré 3 revient à dire que la courbe de Bézier $\mathcal{C}_{A,B,C,D}$ est réunion de la courbe de Bézier $\mathcal{C}_{A,AB,ABC,ABCD}$ et de la courbe de Bézier $\mathcal{C}_{ABCD,BCD,CD,D}$. C'est donc un algorithme récursif. L'algorithme est le suivant :

```
DeCasteljau(A,B,C,D) :
  Si la condition d'arrêt n'est pas vérifiée faire
    Calculer AB, BC, CD, ABC, BCD et ABCD
    Dessiner le point ABCD
    DeCasteljau(A,AB,ABC,ABCD)
    DeCasteljau(ABCD,BCD,CD,D)
```

Le test d'arrêt que nous allons utiliser porte sur la distance entre les points de contrôle.

Exercice 10. Ecrire une fonction

```
float carre_distance(Point A,Point B)
```

qui retourne le carré de la distance entre les points `A` et `B`.

Le test d'arrêt consistera par exemple à arrêter l'appel récursif lorsque le maximum des carrés des distances entre A et les autres points est au plus 0.1.

Exercice 11. Ecrire une fonction

```
Point milieu(Point A,Point B)
```

qui retourne le `Point` milieu du segment `[A,B]`.

Dans la librairie est définie une classe `CourbeBezier`, dans les fichiers `bezier.h` et `bezier.cpp`. L'instruction `CourbeBezier courbe(A,B,C,D)` où `A`,`B`,`C` et `D` sont des variables de type `Point` représente la courbe $\mathcal{C}_{A,B,C,D}$. Si `courbe` est une variable de type `CourbeBezier` alors les commandes `courbe.A`, `courbe.B`, `courbe.C` et `courbe.D` retourne respectivement les points `A`, `B`, `C` et `D`.

Exercice 12. Ecrire, en utilisant l'algorithme de `DeCasteljau` et les deux fonctions précédentes, une fonction

```
void dessine_courbe_bezier(Ecran_SDL,CourbeBezier courbe,Couleur c)
```

qui dessine en couleur `c` la courbe de Bézier `courbe`.

Exercice 13. A l'aide de votre programme dessiner au moins 6 courbes de Bézier en blanc avec les points de contrôle repérés à l'aide de petits carrés rouges. Pour chaque dessin de courbe, vous mettrez le code source et la copie d'écran associée dans un sous-répertoire du répertoire `TesteDeCasteljau` de l'archive que vous me rendrez. Je dois pouvoir en compilant et lancant chacun des codes retombés sur le même dessin. Un exemple de copie d'écran est disponible dans l'archive que vous avez téléchargée.

6. INTERACTIONS

Compiler et lancer le programme contenu dans l'archive `bezier.tar.gz` disponible sur mon site et que vous avez déjà dû télécharger, compiler et lancer.

Exercice 14. Expliquer ce que fait ce programme ainsi que son fonctionnement, surtout la partie contenue dans le fichier `main.cpp`

Exercice 15. Ecrire une fonction

```
bool est_dans_carre(Point P,Carre C)
```

qui teste si un le `Point P` est contenu dans le `Carre C`.

Exercice 16. Ecrire un programme qui affiche un carré à l'écran et permet ensuite de le déplacer. Pour déplacer le carré, on clique à l'intérieur puis on déplace la souris tout en maintenant le bouton enfoncé. Le déplacement s'arrête lorsque le bouton est relâché. Vous mettrez le code source de ce programme dans un sous-répertoire `Deplacement` du répertoire `Interactions` de l'archive que vous me rendrez.

Exercice 17. Ecrire un programme qui affiche une courbe de Bézier de degré 3, qui a donc 4 points de contrôle. Chaque point de contrôle sera représenté par un petit carré. L'utilisateur devra pouvoir déplacer chacun de ces petits carrés de la même manière qu'à l'exercice précédent. La courbe de Bézier sera alors retracée en fonction de la nouvelle position des points de contrôle.