LIFAMI

Contrôle Continu mi-parcours (Durée : 1h00)

Lundi 14 mars 2022

Recommandations : Les documents, calculatrice, téléphone portable sont interdits. La qualité de l'écriture et de la présentation sera prise en compte dans la note finale.

NOM : ……

PRENOM : ……..

Numéro Etudiant : ……

Groupe ou Chargé de TD…..……

Questions de cours (5 pts)

1. En utilisant la structure `Complex` telle qu'elle a été présentée dans les TD / TP, écrivez en C++ l'`operator*` qui prend en paramètres 2 `Complex` et retourne un `Complex` calculé comme le produit de ces deux `Complex`.

2. Ecrivez la fonction `make_complex_exp` qui à partir d'un rayon `r` et d'un angle `theta_deg` **donné en degrés**, retourne un `Complex`.

On suppose les fonctions / opérateurs suivants écrits.

```
Complex make_complex(float x, float y);
Complex make_ complex_exp(float r, float theta_rad); // en radian
Complex operator+(Complex a, Complex b);
Complex operator-(Complex a, Complex b);
Complex operator*(Complex a, float lambda);
```

3. En utilisant les fonctions à votre disposition, écrivez en C++ la fonction `rotate` qui prend en paramètre un `Complex p` à faire tourner, un `Complex c` qui est le centre de rotation, et un réel `angle` **en degrés** définissant l'angle de rotation. La fonction calcule et renvoie la rotation du `Complex p` tourné d'un angle `angle` autour du `Complex c`.

Interpolation forme et couleur… (15 points)

Dans tout l'exercice on supposera la structure `Point` et la structure `Color` définies comme dans les TD/TP et on pourra utiliser les opérateurs / fonctions suivants :

```
Point make_point (float x, float y)
Point make_point_exp(float r, float theta_deg)
Point operator+(Point a, Point b)
Point operator-(Point a, Point b)
Point operator*(float a, Point b)
Color make_color (unsigned char r, unsigned char g, unsigned char b)
Color operator+(Color a, Color b)
Color operator*(float a, Color b)
Color interpolation_couleur (Color A, Color B, float t)
```

Dans cette partie, nous allons effectuer une interpolation entre 2 cercles colorés (*Fig. 1*) : entre le cercle rouge à l'intérieur et le cercle bleu à l'extérieur. **Nous bannirons l'utilisation de la fonction `circle`**. Un cercle sera représenté par un tableau de points, comme pour l'exercice sur les polygones. L'interpolation portera à la fois sur la position des différents points qui constituent le cercle et sur la couleur. Les figures 2, 3 et 4 représentent différentes interpolations pour des valeurs de t. La couleur du cercle interpolé passe par toutes les nuances entre le rouge et le bleu.

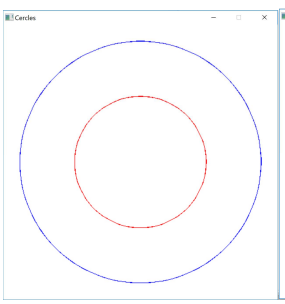

Fig. 1, les 2 cercles.

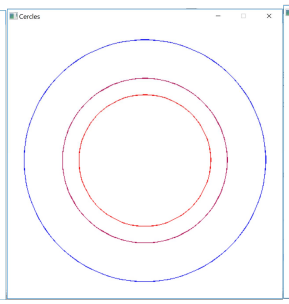

Fig. 2, t=0.1

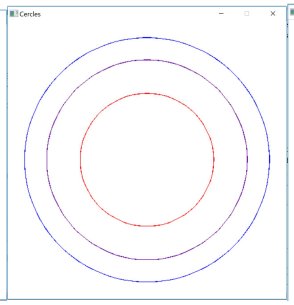

Fig. 3, t=0.6

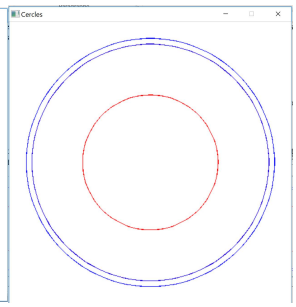

Fig. 4, t=0.9

1. Ecrivez en C++ la fonction `Interpolation_Point` permettant à partir de 2 `Points` et d'un paramètre `t` réel de calculer et retourner un nouveau `Point` résultat de l'interpolation par `t`.

2. Un `Cercle` sera défini par son rayon `r`, le nombre de `Point n` qui le décrivent, ainsi qu'un tableau de `Point p`. Le premier `point` du tableau correspondra aux coordonnées du centre du cercle et les suivants aux différents `Point` du cercle. On pourra utiliser une constante `MAX` pour la taille du tableau. Définissez la structure `Cercle`.

3. Ecrivez en C++ la procédure `Init_Cercle` qui remplit un `Cercle`. Le rayon sera passé en paramètre, le centre sera placé au milieu de la fenêtre de taille `DIMW` et on positionnera de manière régulière `n = 180 Points` sur le `Cercle`.

4. Ecrivez en C++ la procédure `Draw_Cercle` qui affiche un `Cercle` passé en paramètre. Deux `Point` successifs seront reliés par une `line`. Attention, le premier `point` correspond au centre du `Cercle`.

5. Ecrivez en C++ la fonction `Interpolation_Cercle` qui retourne un `Cercle`, résultat de l'interpolation de deux `Cercle A` et `B` par le paramètre `t`, passés en paramètres. On suppose que les 2 cercles ont le même nombre de points.

6. On dispose d'une structure `Color` contenant les 3 composantes `r`, `g`, et `b` entier, de tous les opérateurs sur les `Color`, d'une fonction `make_color` qui créé une `Color` à partir de 3 entiers, ainsi que d'une fonction `interpolation_color` qui à partir de 2 `Color` retourne le résultat de l'interpolation par un paramètre `t`. Complétez le programme principal suivant pour qu'il affiche le `Cercle C1` en rouge, le `Cercle C2` en bleu et qu'il calcule un cercle intermédiaire `C3` comme étant l'interpolation entre `C1` et `C2` par un paramètre `t` qui évoluera au cours du temps. Pour l'évolution de `t` vous pourrez utiliser par exemple `elapsedTime`. La `Color` du cercle interpolé sera également interpolée entre `C1` et `C2`. Rappel : `color (r,g,b)` permet de changer la couleur du pinceau dans `grapic` avant de dessiner.

```
int main(int argc, char** argv)
{
    bool stop=false;
    winInit("Cercles", DIMW, DIMW);// Créer une fenêtre (taille DIMWxDIMW)
    backgroundColor(255,255,255);
    Cercle C1, C2, C3;
    Color Col1, Col2, Col3;
    InitCercle(C1,120);
    InitCercle(C2,220);



    while (!stop)
    {
         winClear();

         stop= winDisplay();
    }
    pressSpace();
    winQuit();
    return 0;
}
```