LIFAMI

Seconde Chance (Durée : 1h30min)

NOM :

……
……

PRENOM :

……
……

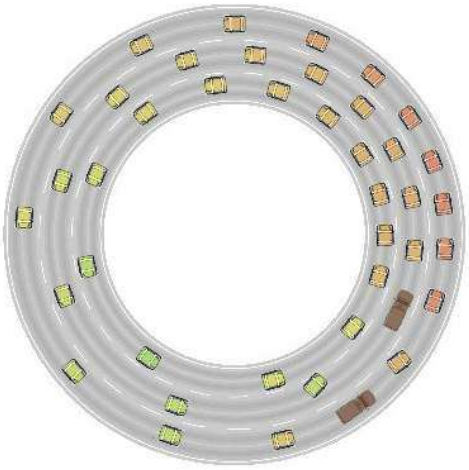
Numéro Etudiant :

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.

Pour la suite, vous disposez de la structure *Complex* et des fonctions/procédures suivantes.

```
struct Complex { float x,y ; } ;
Complex make_complex(float x, float y);
Complex make_complex_exp(float r, float angle_radian);
Complex operator+(Complex a, Complex b);
Complex operator*(float a, Complex b);
Complex operator*(Complexe a, Complex b);
Complex operator/(Complex a, float b);
```

Nous cherchons à faire une simulation du trafic routier sur des routes circulaires comme ceci :



1. Une voiture est définie par deux réels : un angle *ang* (degrés) et une vitesse angulaire *va* (degrés par seconde). Définissez la structure *Voiture*. Ici il n'y a pas besoin de vecteur à 2 dimensions car la voiture est contrainte sur une ligne circulaire qui peut se représenter en 1D, par l'angle.

…/1

2. Une route est définie par un tableau *tab* de voitures de taille MAX_VOITURES, un entier *nv* représentant le nombre de voitures réellement actives dans le tableau et un rayon *ray* qui donne le rayon de la route. Définissez la structure *Route*.

… / 1

3. Écrivez la procédure *init_route* qui initialise une route à partir d'un nombre de voitures et d'un rayon. Cette procédure placera les voitures dans le tableau de manière régulière sur un tour complet. Nous contraignons les routes à ce que l'ordre du tableau représente l'ordre des voitures sur le cercle. Une voiture de la case i aura un angle plus petit que la voiture de la case i+1. La voiture se trouvant devant la dernière voiture est la voiture de la case 0. La vitesse angulaire *va* sera tirée au hasard entre 10 et 30 degrés par seconde.

… / 3

4. Écrivez la fonction *rotate* qui prend en paramètre un nombre complexe à faire tourner, le centre de la rotation (*Complex*) et l'angle en degrés (réel) de la rotation.

… / 3

5. Écrivez la procédure *dessine_route* qui affiche une route et toutes les voitures. La route sera affichée avec deux cercles centrée à l'écran de rayon plus ou moins 5 par rapport au rayon stocké dans la structure route. Une voiture sera affichée par un cercle d'un rayon que vous devez déterminer pour que le cercle soit tangent aux bords de la route. Pour placer la voiture vous aurez besoin de la fonction *rotate*.

… / 3

6. A chaque pas de temps *dt*, une voiture avance en fonction de sa vitesse angulaire *va*, mais une voiture ne peut pas dépasser les voitures devant elle. Il faut donc faire les déplacements en deux phases bien distinctes.
    a. Faire avancer toutes les voitures dans une structure *Route* temporaire, sans tenir compte des dépassements éventuels. L'angle dépassera à un moment 360 degrés pour se souvenir combien de tour ont été effectué.
    b. Gérer les dépassements interdits : afin que les voitures se trouvant devant aient déjà résolu leurs collisions éventuelles, il faut parcourir le tableau temporaire de la fin vers le début. Pour les voitures ayant dépassées la suivante, il faut la faire revenir au même niveau que la voiture de devant moins un angle de 3 degrés et changer le vecteur vitesse pour prendre en compte le freinage.

Ecrivez le sous-programme *update_route* qui prend une route et un réel *dt* en paramètre et qui met à jour les voitures en les faisant avancer.

… / 4

7. Écrivez une fonction *verifOrdre* qui prend en paramètre une route et qui renvoie vrai quand toute les voitures sont dans le bon ordre et faux sinon. Attention la dernière voiture du tableau doit se comparer à la première voiture plus 360 degrés.

… / 3

8. Écrivez la fonction principale *main* qui va créer la route avec 10 voitures et un rayon de 100 pixels. Puis cette fonction affiche et déplace les voitures en boucle.

… / 2

```
int main(int argc, char **)
{
        winInit("Coraly", DIMW, DIMW) ;









        return 0 ;
}
```