# LIFAMI – TP : Nombres complexes et transformations du plan

Ce TP va chercher à illustrer les différents aspects des nombres complexes au travers de différentes applications.

## Les nombres complexes sont super cools !

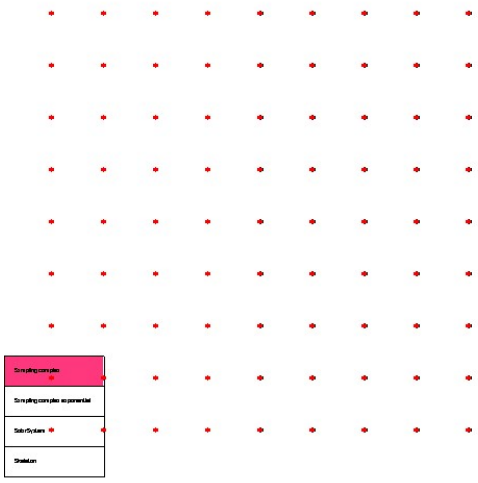
1. Codez les fonctions et procédures du TD questions de 1 à 8

## Le pavé dans le plan complexe

En TD nous avons vu deux manières de construire un nombre complexe.

- Représentation algébrique : $C = x + i.y$
- Représentation exponentielle : $C = r(\cos(\theta) + i \sin(\theta)) = r.e^{i\theta}$

2. Ecrivez une procédure *draw* qui va effectuer une double boucle affichant une grille régulière sur l'espace des complexes, c'est-à-dire des points espacés de 10 unités entre 0 et 500 (si 500 est la taille de votre fenêtre) en utilisant *make_complex*.



3. Faites de même mais en explorant les paramètres $r$ et $\theta$ en utilisant *make_complex_exp*.
   - La boucle sur $r$ ira de …… à ……. par pas de …….
   - La boucle sur $\theta$ ira de …… à ……. par pas de …….

   Quelle figure allez-vous obtenir ?

## Battre des ailes pour voler !

L'objectif de cet exercice est de guider au clavier un oiseau qui bat des ailes.

4. Un oiseau est défini par une position représentée par un nombre complexe et un angle d'ouverture des ailes. Définissez la structure.

5. Ecrivez la fonction qui affiche l'oiseau.

   Pour faire une animation la fonction main va ressembler à ceci :

```
int main(int , char** )
{
    bool stop=false;
    winInit("Birds", DIMW, DIMW);
    backgroundColor( 100, 50, 200 );
    Bird bi;
    init(bi);
    while( !stop )
    {
            winClear();
            draw(bi);
            update(bi);
            stop = winDisplay();
    }
    winQuit();
    return 0;
}
```

6. A chaque itération de la boucle principale, l'oiseau bat des ailes en changeant l'angle. Ceci va se faire dans la fonction *update*. On pourrait aussi faire le changement d'état dans la procédure *draw*, mais pour bien structurer le code la procédure *draw* affiche et la procédure *update* met à jour les variables.

   Indication : l'angle va varier entre -20 et +20 degrés. Basez-vous sur une fonction oscillante et périodique que vous connaissez bien.

7. Ajoutez des comportements :
   - l'oiseau tombe ;
   - si on appuie sur la flèche haut, l'oiseau bat des ailes et monte ;
   - si on appuie sur flèche gauche/droite l'oiseau va à gauche/droite.

   Pour tester une touche :

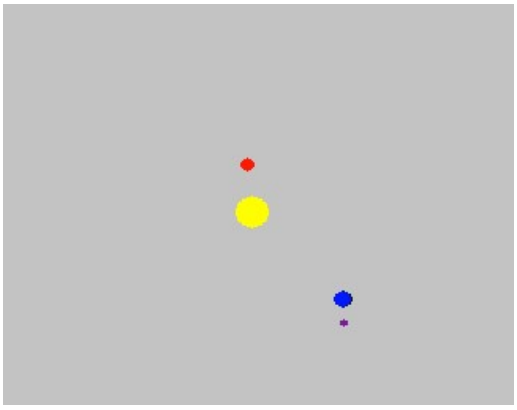
```
if (isKeyPressed(SDLK_UP))
{
```

## Saturne, ça tourne !

8. Reprenez la question du système solaire du TD et codez le.

9. Vous ajouterez 4 images dans votre structure pour afficher une image de la planète à la place d'un cercle.

10. Vous pourez également faire tourner le soleil autour d'un centre de galaxie placé au milieu de la fenêtre.

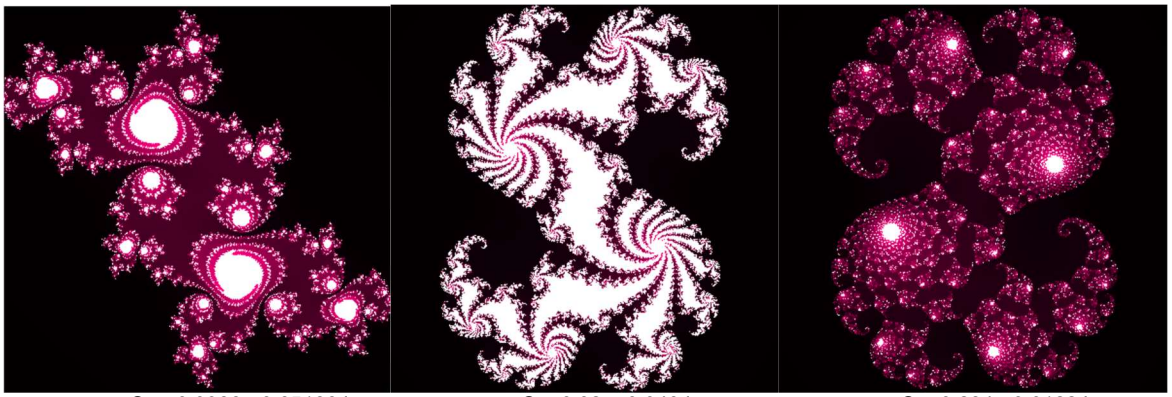


## Julia

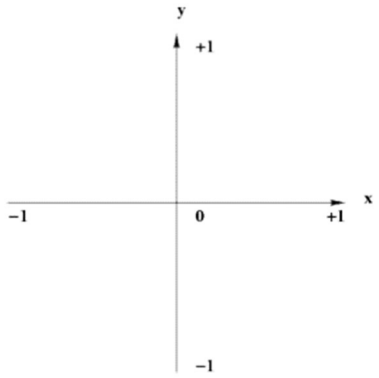
Définition de l'ensemble de Julia (cf. http ://fr.wikipedia.org/wiki/Ensemble de Julia). Etant données deux nombres complexes, C et $Z_0$, définissons la suite $(Z_n)$ par la relation suivante :

$$Z_{n+1} = Z^2_n + C$$

Pour une valeur donnée de C, l'ensemble de Julia est la frontière de l'ensemble des valeurs initiales $Z_0$ pour lesquelles la suite est bornée. Dans notre cas, nous allons faire correspondre une valeur de $Z_0$ pour chaque pixel de l'image.



C = -0.0986 - 0.65186.i C = 0.32 + 0.043.i C = 0.284 - 0.0122.i




Chaque pixel de l'image va correspondre à un point complexe entre [-1 ;1]

11. Ecrire une fonction qui implémente la suite ($Z_n$) et qui retourne le numéro du terme dont le module (la norme) est supérieur à une borne, si le module reste inférieur à cette borne, la fonction renverra le nombre maximal d'itérations. La borne d'arrêt, le nombre d'itération maximal, le terme $Z_0$ et la constante C sont des paramètres de la fonction. Cette fonction renverra le nombre d'itérations avant que la série dépasse la borne.

12. Ecrire un sous-programme qui « renvoie » une couleur en prenant en entrée le nombre d'itérations qui a permis l'arrêt du calcul de la suite de la question précédente. Pour les couleurs, choisissez celles que vous voulez ! En divisant le nombre d'itérations par le nombre d'itérations maximal vous obtenez un réel entre 0 et 1. Cette fonction pourra par exemple modifier 3 champs *r,g,b* passés en paramètre.

13. Ecrire la procédure *draw_julia()* qui permette d'afficher un ensemble de Julia. Cette procédure utilisera les 2 procédures précédentes pour chacun des pixels de l'image. Vous utiliserez la fonction *put_pixel* de Grapic pour colorier un pixel de la fenêtre.

    Convertissez les coordonnées (i,j) de chaque pixel en Complexe de coordonnées de l'espace de la fonction entre [-1,1]. Passez ce complexe en paramètre $Z_0$. Essayez C = 0.32 + 0.043.i

14. Testez votre programme, en changeant les dimensions de l'image, le nombre d'itérations maximal, la valeur de la constante C, les couleurs, la plage [-X ;X], ...