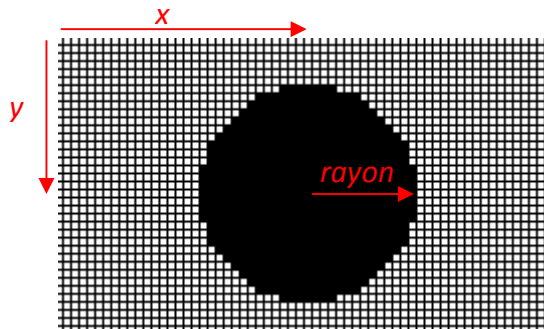


## Documentation sur la librairie ncLib.h version 1.1

La librairie contient 3 fonctions graphiques.

- int disque(int *position\_dans\_x*, int *position\_dans\_y*, int *rayon*, int *couleur*)

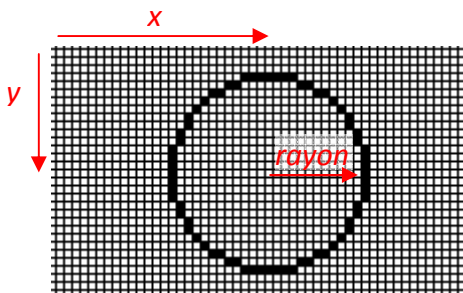
Il s'agit de dessiner un cercle plein parfait dans la VRAM, sans utiliser la trigonométrie, à partir de l'algorithme de Bresenham, un ingénieur ayant travaillé chez IBM dans les années 60. Pour plus d'informations sur Bresenham et ses algorithmes consultez wikipedia.



Si *couleur* = 1, le disque est tracé en noir.  
Si *couleur* = 0, le disque est tracé en blanc.

- int cercle(int *position\_dans\_x*, int *position\_dans\_y*, int *rayon*, int *couleur*)

En utilisant toujours l'algorithme de Bresenham, il s'agit de tracer un cercle dans la VRAM à fond transparent.



Si *couleur* = 1, le cercle est tracé en noir.  
Si *couleur* = 0, le cercle est tracé en blanc.

- int affichebitmap(int *position\_dans\_x*,int *position\_dans\_y*,int *longueur\_x*,int *longueur\_y*, unsigned char *\*imagedata*)

Il s'agit d'afficher une image bitmap dans la VRAM

- Une image bitmap noir et blanc



|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Pour obtenir une image bitmap, on prend une image en noir et blanc, on prend les pixels 8 par 8 en partant du haut à gauche. On remplace les pixels noirs par des 1 et les blancs par des 0 de manière à obtenir un nombre binaire de 1 octet. On convertit ensuite ce nombre en hexadécimal. Ici, b(11010011)=h(D3).

Avec le SDK, cette suite de nombres hexadécimaux se met dans un tableau du type 'const unsigned char *tampon*[] = {0xXX,...}'

'XX' représente vos nombres en hexadécimal, '0x' signifie qu'ils sont en hexadécimal.

- Une question de place

Alors pourquoi s'embêter à utiliser une bitmap plutôt que d'utiliser des fonctions telles que 'Bdisp\_DrawLineVRAM(*x,y,x,y*)' ou 'Bdisp\_SetPoint\_VRAM(*x,y,coul*)' ?

Tous simplement parce que la méthode des bitmap prend beaucoup moins de place dans votre add-in.

Après avoir déclaré votre tableau bitmap, utilisez affichebitmap.

*position\_dans\_x* : là où sera mis la bitmap dans x. (à la gauche de l'image)

*position\_dans\_y* : là où sera mis la bitmap dans y. (en haut de l'image)

*longueur\_x* : la longueur de votre image en pixels sur x.

*longueur\_y* : la hauteur de votre image en pixels sur y.

*\*imagedata* : ici, votre tableau (ex : '(unsigned char\*)tampon')