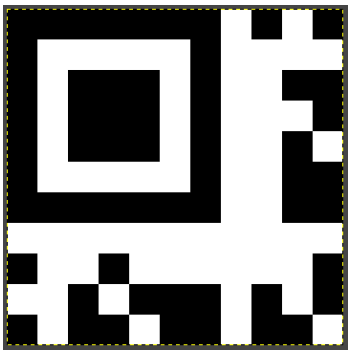


Projet QRCODE

Partie 2 :

Dans cette partie je devais réaliser la partie masquage du QRCODE avec le meilleur choix de masque.

I. Génération des masks :

Tout d'abord je me suis penché sur la réalisation des différents masques à l'aide de la formule donner par la norme ISO 18004/2015

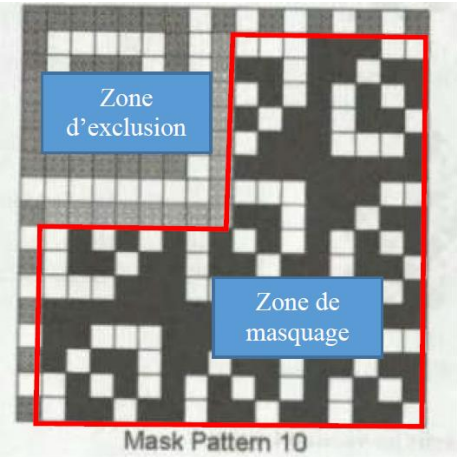
Table 10 — Data mask pattern generation conditions

Data mask pattern reference for QR Code symbols	Data mask pattern reference for Micro QR Code symbols	Condition
000		$(i + j) \bmod 2 = 0$
001	00	$i \bmod 2 = 0$
010		$j \bmod 3 = 0$
011		$(i + j) \bmod 3 = 0$
100	01	$((i \operatorname{div} 2) + (j \operatorname{div} 3)) \bmod 2 = 0$
101		$(ij) \bmod 2 + (ij) \bmod 3 = 0$
110	10	$((ij) \bmod 2 + (ij) \bmod 3) \bmod 2 = 0$
111	11	$((i+j) \bmod 2 + (ij) \bmod 3) \bmod 2 = 0$

Figure 21 shows all data mask patterns, illustrated in a version 1 symbol. Figure 23 simulates the effects of data masking using data mask pattern references 000 to 111.

```
*************Affichage QRCODE sur la console****************
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . * *
. . . . . . . . . . *
. . . . . . . . . * .
. . . . . . . . . . *
. . . . . . . . . * *
. . . . . . . . . . .
. . . . . . . . . * *
. . * . * . * . * . *
. . * * . * * . * * .
```

En vérifiant bien que le cahier des charges à bien été respecter sur la génération du mask. Pour rappel le masque ne doit pas être généré sur le finder pattern, timing pattern et Version Information



Dans notre cas le cahier des charges est bien respecté.

```c
void genere_QRmask(unsigned char qrmask[NB_MODULE][NB_MODULE],int no_masque)
{


  printf("\n donnee Utile : \tNumero du masque :%d \t Nombre de module : %d \n", no_masque, NB_MODULE);
    //Afin d'avoir un aperçu sur la console des valeurs transmissent dans la boucle.

    int i,j, finder =9;
    //Envoie de donnée dans un tableaux, avec la formules
    //des masks
    for(i=1; i<NB_MODULE; i++)
    {
        for(j=1; j<NB_MODULE; j++)
        {

            switch (no_masque)
            {
            case 0: //Si le masque 0 est sélectionner
                if(i%2==0)
                    qrmask[i][j]=NOIR;
                if(i%2!=0)
                    qrmask[i][j]=BLANC;
                break;

            case 1: //Si le masque 1 est sélectionner
                if((i/2+j/3)%2==0)
                    qrmask[i][j]=NOIR;
                if((i/2+j/3)%2==0)
                    qrmask[i][j]=BLANC;
                break;

            case 2: //Si le masque 2 est sélectionner
                if(((i*j)%2+(i*j)%3)%2==0)
                    qrmask[i][j]=NOIR;
                if(((i*j)%2+(i*j)%3)%2!=0)
                    qrmask[i][j]=BLANC;
                break;

            case 3: //Si le masque 3 est sélectionner
                if(((i+j)%2+(i*j)%3)%2==0)
                    qrmask[i][j]=NOIR;
                if(((i+j)%2+(i*j)%3)%2!=0)
                    qrmask[i][j]=BLANC;
                break;
            }
            if(i< finder && j<finder ||j==0 && i<NB_MODULE ||i==0 && j<NB_MODULE)
                qrmask[i][j]=BLANC;
        }

    }


}
```

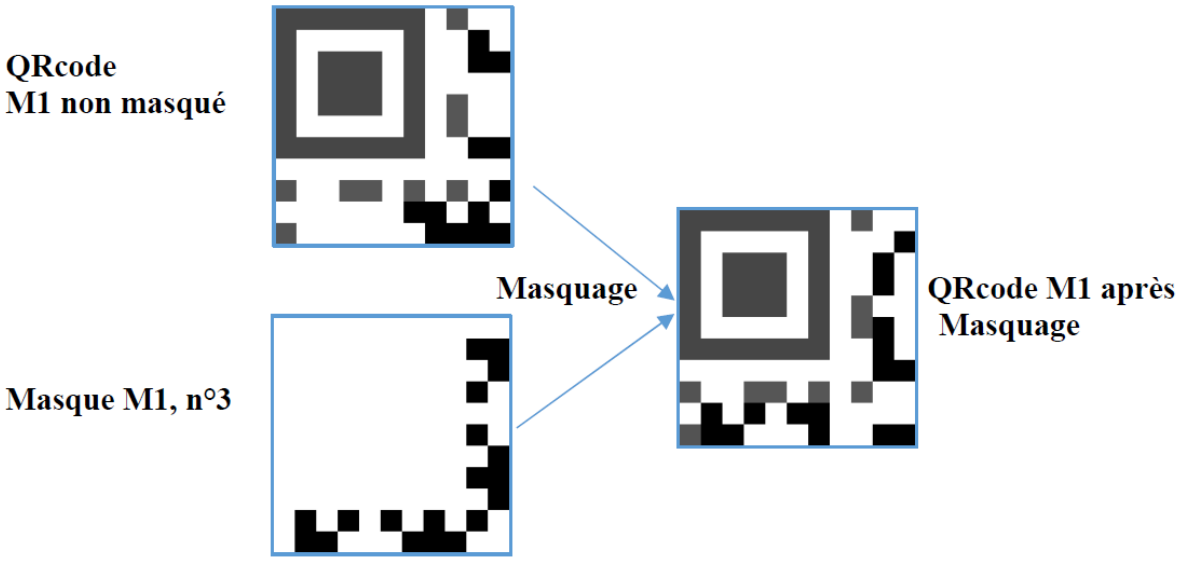
Voici le code nous permettant

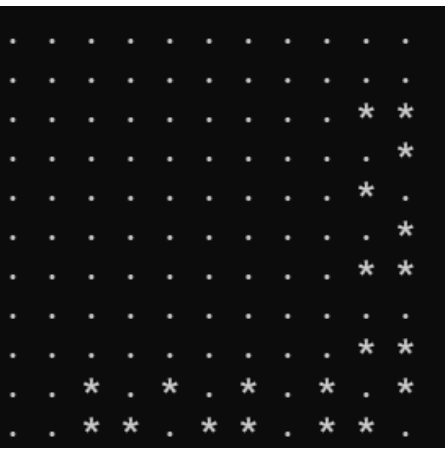
De générer les différents masques

II. Réalisation du XOR

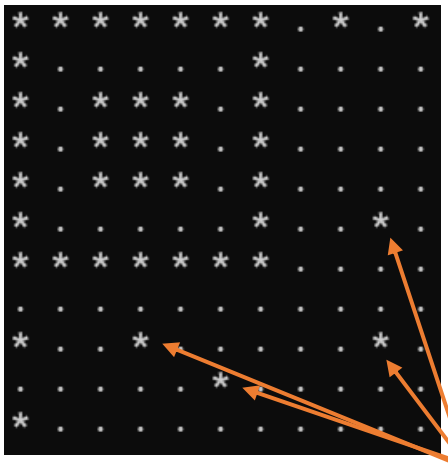
Maintenant nous devons réaliser un XOR afin de combiner le masque généré avec le qrcode.

On devait respecté comme contrainte que si le module du masque est noir et que celui du qrcode est blanc, alors le qrcode final prendra comme module la couleur noir. En Revanche si le masque et le qrcode sont tout deux de couleur noir, alors le qrcode final aura pour module la couleur blanc.





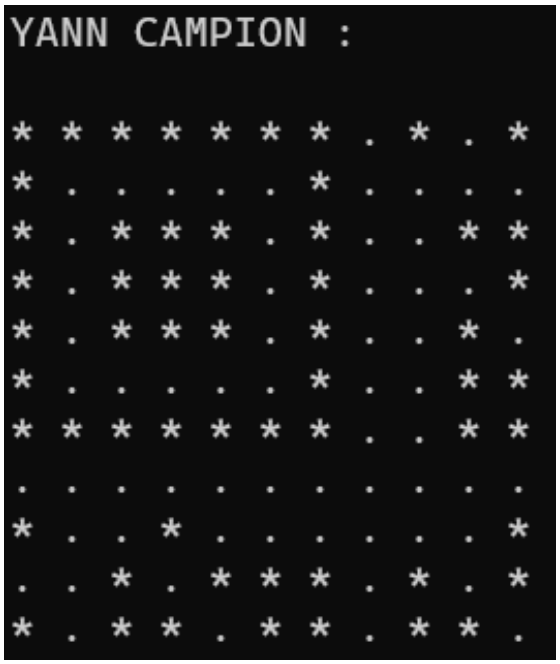
Masque généré



Pattern généré depuis la partie 0.

Quelques valeurs de test afin de voir si le xor fonctionne bien correctement lors de son exécution

Une fois la fonction du XOR effectuer nous obtenons ainsi le résultat suivant :



Le Xor réalise donc bien le cahier des charges.

```
1  void xor_QRcode_QRmask(unsigned char qrcode[NB_MODULE][NB_MODULE],const unsigned char
 2 qrmask[NB_MODULE][NB_MODULE])
 3 {
 4     int i,j;
 5 //appel des deux tableaux
 6     for(i=0; i<NB_MODULE; i++)
 7     {
 8         for(j=0; j<NB_MODULE; j++)
 9         {
10             //mettre les cases noirs du masque sur les cases blanches du qrcode
11             if(qrmask[i][j]==NOIR && qrcode[i][j]==BLANC)
12                 qrcode[i][j]=NOIR;
13             //sinon si le masque a les mêmes cases que le qrcode en noir mettre dans le
14             // qrcode final la case en blanc
15             else if (qrmask[i][j]==NOIR && qrcode[i][j]==NOIR)
16                 qrcode[i][j]=BLANC;
17         }
18     }
   }
```

III. Génération du score pour le masque

Je devais faire un score, permettant de noter le masque choisi après exécution du XOR. Pour cela il devait compter le nombre de case noir étant sur la dernière ligne, et la dernière colonne du qrcode, puis calculer le score du masque à l'aide de la formule de la norme ISO18004/2015 (p54)

If $SUM_1 \leq SUM_2$

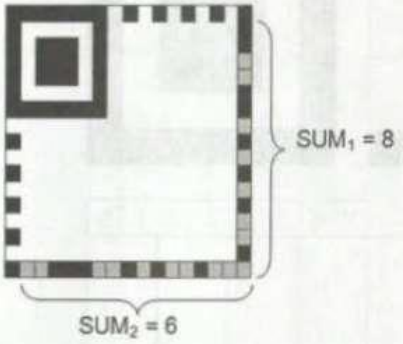
Evaluation score = $SUM_1 \times 16 + SUM_2$

If $SUM_1 > SUM_2$

Evaluation score = $SUM_2 \times 16 + SUM_1$

SUM_1 number of dark modules in right side edge

SUM_2 number of dark modules in lower side edge



$SUM_1 > SUM_2$

Evaluation point = $SUM_2 \times 16 + SUM_1$

= (6 × 16) + 8

= 104

Figure 24 — Evaluation of masking results in Micro QR Code symbol

```
YANN CAMPION :
**********************Score masque num├®ro 0******************


score = 107
score du masque 107 :
```

Notre score est donc correct avec le masque choisi.

```c
int score_masquage_QRcode(const unsigned char qrcode[NB_MODULE][NB_MODULE])
{
    int som_1=0, som_2=0, i,j;
    int score=0;
    for(i=0; i<NB_MODULE; i++)
    {

        for(j=0; j<NB_MODULE; j++)
        {
            //j==10 Dernière colone du qrcode+Comptage case noir
            //dans cette colone
            if(j==10 && qrcode[i][j]==NOIR )
                som_1=som_1+1;
            //i==10 Dernière ligne du QRcode+comptage case noir
            if(i==10 && qrcode[i][j]==NOIR)
                som_2=som_2+1;
            //Calcule du score final
            if(som_1<=som_2)
                score=som_1*16+som_2;

            if(som_1>som_2)
                score=som_2*16+som_1;
        }
    }
    printf("score = %d",score);
    //Renvoie dans le test unitaire 2 la valeur du score
    return score;
}
```

IV. Choix du meilleur masque pour une génération optimal du qrcode

Pour cela on devait utiliser le masque nous donnant le score le plus important. J'ai généré les 4 masque directement dans la fonction test-unitaire2 afin de générer les 4 masques successivement, avec enregistrement du score

```
YANN CAMPION :
********************Score masque num├®ro 1*****************


score = 54
score du masque 54 :

********************Meilleur score pour le masque le plus adapt├®

score max = 107 et meilleur mask = 0



 donnee Utile :         Numero du masque :2      Nombre de module : 11
reinitialisation en cour
Reinitialisation fini


YANN CAMPION :
********************Score masque num├®ro 2*****************


score = 119
score du masque 119 :

********************Meilleur score pour le masque le plus adapt├®

score max = 119 et meilleur mask = 2
```

Nous pouvons maintenant récupérer la valeur de notre score, et le masque utiliser nous permettant d'obtenir se score, pour générer le qrcode avec.

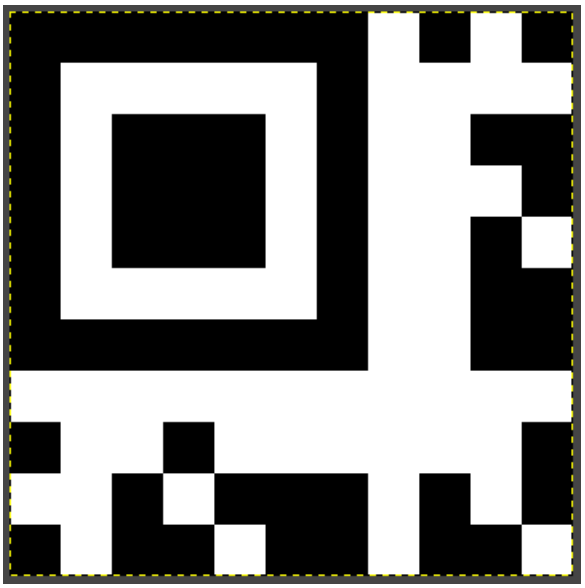
Voici le résultat final avec le masque optimal.

```
YANN CAMPION :

* * * * * * * . * . *
* . . . . . * . . . .
* . * * * . * . . * *
* . * * * . * . . . *
* . * * * . * . . * .
* . . . . . * . . * *
* * * * * * * . . * *
. . . . . . . . . . .
* . . * . . . . . . *
. . * . * * * . * . *
* . * * . * * . * * .


Process returned 0 (0x0)   execution time : 1.114 s
Press any key to continue.
```

Et voici le rendu final après exportation en image



```c
void test_unitaire_sujet2(void)
{
    printf("Void test_unitaire_sujet2\n\n\n");
    unsigned char MicroQRcode[NB_MODULE][NB_MODULE];  // le microQRcode sans la Quiet ZONE
    unsigned char MicroQRmask[NB_MODULE][NB_MODULE];  // le mask
    unsigned short int mode_microQRcode = M4_L ;     // choix du mode
    unsigned short int mask_number = 0;              // choix du n° de MASK
    int score, score_max =0, mask; //Initialisation des
    //fonctions score, score_max et mask

    efface_QRcode(MicroQRmask); //Effacement du MicroQRmask afin
    //de pouvoir en généré un sans problème de génération
    printf("\n\n\n**********Selection du meilleur mask**********\n\n\n\n");

    for(mask_number=0; mask_number<=3; mask_number++) //Boucle permettant de générer
    {
        //tout les masques avec leurs QRCODES afin de sélectionner le mask meilleur mask.
        genere_QRmask(MicroQRmask,mask_number);
        efface_QRcode(MicroQRcode);
        initialise_QRcode(MicroQRcode);
        xor_QRcode_QRmask(MicroQRcode, MicroQRmask);

        printf("**********Score masque numéro %d**********\n\n\n", mask_number);
        //Affichage des différents scores de chaque masque.
        score = score_masquage_QRcode(MicroQRcode);
        printf("\nscore du masque %d : \n", score);
        //Maintenant réalisons une boucle mémorisant le score le plus grand
        if(score>=score_max)
            score_max = score;
        //Réalisons une boucle mémorisant le numéro de mask en fonction
        //du score
        if(score_max == score)
            mask = mask_number;
        printf("\n**********Meilleur score pour le masque le plus adapté");
        printf("\n\nscore max = %d et meilleur mask = %d", score_max, mask);

        printf("\n\n\n\n");

    }
    //Maintenant génération du QRCODE avec le meilleur mask
    printf("\n\n\n**********Selection du meilleur mask**********");
    printf("\n\n**********Donnee Utile**********");
    printf("\nMeilleur mask : %d \nScore atteint : %d", mask, score_max);

    printf("\n\n\n**********Génération QRCODE final**********");
    genere_QRmask(MicroQRmask,mask);
    printf("\n\n\n**********Affichage QRCODE sur la console**********\n");
    QRcode_to_console(MicroQRmask);
    efface_QRcode(MicroQRcode);
    initialise_QRcode(MicroQRcode);
```