



# Modules systèmes 8.1 et 8.2

Durée : 2x24h de TP en 2<sup>ème</sup> année BTS-SNIR

## Programmation Orientée Objet : conception Traitement d'images



### Objectifs :

Traiter, en mode console, un fichier Bitmap afin d'en extraire l'entête et l'ensemble des pixels. Créer une classe image permettant d'effectuer les traitements les plus courants. En utilisant l'héritage, mettre en œuvre une conception orientée objet permettant de traiter l'image d'une cible provenant d'une séance de tir au pistolet. Utiliser les techniques de seuillage et de filtrage afin de détecter l'emplacement de la cible dans l'image brute, puis les coordonnées des impacts dans la cible. Calculer le score du tireur et l'afficher sur un journal lumineux. Ajouter la prise automatique de photo de la cible à l'aide d'une Web-Cam.

### Contraintes :

Le logiciel fonctionnera sous Windows.  
Une conception UML est exigée.  
Le langage C++ orienté objet est imposé.

### Matériel :

PC sous Windows.  
Caméra Heden.  
Ensemble de cibles cartonnées avec impacts.  
Journal lumineux.

### Logiciels :

C++ Builder XE6 ou QT creator.  
Logiciels constructeurs de l'afficheur.  
Logiciels constructeurs de la caméra.

### Documentation :

Documentation constructeur de l'afficheur.

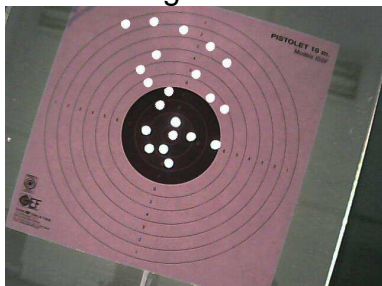
### Annexes :

Format d'un fichier BMP.  
Classe SNImage, son constructeur et la méthode de chargement d'une image.  
Méthodes de dessin et de recherche sur l'image.  
Exemples de traitements d'images.  
Principe du traitement de la cible.  
Normalisation cible pour pistolet à air à 10m.  
Format binaire des images BMP JPG et PNG.

**Un compte rendu écrit -et manuscrit- de votre travail est à rendre à la fin du module.**

Évaluation écrite-éclair à chaque fin de séance

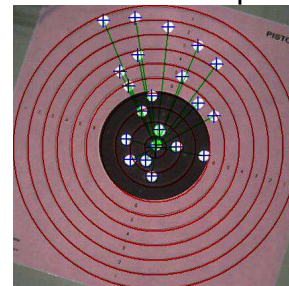
Image brute



Centrage



Détection des impacts





## MISE EN SITUATION

L'entreprise ADS désire remplacer les anciennes machines optiques à compter les points DISAG par un système de caméras moins coûteuses et offrant une analyse plus rapide de la cible. En temps réel, le nom et le score du tireur seront affichés sur un journal lumineux dans le stand de tir. Le nouveau système informatisé offrira la possibilité de suivre les performances des tireurs depuis un site WEB sécurisé.





## A/ Analyse UML

-  A.1 Donner les diagrammes UML de cas d'utilisation et de déploiement du système complet.
-  A.2 Donner le diagramme UML de séquence de la séance de tir.

## MS8.1 CODAGE : lecture d'un fichier BMP, conception et test d'une classe Image

## B/ Lecture et écriture d'un fichier BMP


 B.1 Une classe **SNImage** son constructeur, ainsi que sa méthode de chargement d'un fichier BMP est donnée en annexe 2 : donner la valeur des attributs de la classe (de type **unsigned long**) lors du chargement du fichier image proposé en exemple dans l'annexe 1. Coder et tester (en écrivant le **main()** adéquat) la classe **SNImage**, son constructeur et la méthode **Chargement** : les structures **Coordonnees** (composée de 2 entiers **ligne** et **colonne**) et **Pixel** (composée de 3 octets (**unsigned char**) (**bleu, vert, rouge**)) doivent être créés.

 B.2 En s'aidant d'Internet et du code de la méthode **Chargement**, proposer le code du destructeur de la classe **SNImage** permettant de libérer la mémoire allouée à l'image dans le cas où **dimensionMax** n'est pas nul.

B.3 Coder et tester le destructeur de la classe **SNImage**.

B.4 Coder et tester la méthode de sauvegarde d'une image.

Le code suivant permet d'ouvrir une image sous **Paint**, il permettra de visualiser les résultats :  
`WinExec("mspaint.exe Resultats\\Resultat.bmp" , 1) ;`

 B.5 Ecrire la structure **Coordonnee** composée des deux entiers **ligne** et **colonne**. Ecrire et tester le programme **main()** permettant de tester, les une après les autres, les méthodes suivantes (données en annexe 3).

```
void Colorie(Coordonnee coord,int l,int h,Pixel couleur);
void Detoure(int largeurGauche,int largeurDroite,int largeurHaut,int largeurBas,Pixel couleur);
void Recadre(Coordonnee coord,int l,int h);
Coordonnee RechercheZone(int l,int h,Pixel couleur);
void DessineCroix(Coordonnee coord,int taille,int epaisseur,Pixel couleur);
void DessineCarre(Coordonnee coord,int taille,int epaisseur,Pixel couleur);
void Dessine7Segments(Coordonnee coord,int taille,int epaisseur,Pixel couleur,string message);
Afficher les coordonnées lorsque c'est nécessaire.
```

## C/ Traitements d'images (L'annexe 4 donne des exemples de ces traitements)

C.1 Coder et tester les méthodes (traitements simples) suivantes :

```
void Negatif();
void RetourneHorizontal();
void RetourneVertical();
void NiveauGris();
```

L'annexe 4 donne des exemples de ces traitements.

C.2 Coder et tester les méthodes (traitements complexes) suivantes :

```
void SeuilleNoirBlanc(int niveau);
void Tourne90Droite();
void Tourne90Gauche();
void Tourne180();
```

C.3 Optionnel : Coder et tester les méthodes suivantes :

`void Eclaircit(int niveau);`

`void Assombrit(int niveau);`

L'annexe 4 donne des exemples de ces traitements.

## MS8.2 CODAGE : analyse d'une cible

### D/ conception de la classe cible



D.1 La classe **Cible** héritera de la classe **SNImage**. En étudiant les annexes 5 et 6, déterminer les attributs et les méthodes nécessaires au traitement de la cible.



D.2 Proposer un diagramme de classe complet contenant les 2 structures (**Coordonnee**, **Pixel**), les classes **Cible**, **SNImage** mais aussi une classe **Tireur**.

### E/ traitement de la cible



E.1 Quelles méthodes de la classe **SNImage** doivent être utilisées pour détecter la cible et recadrer l'image sur la cible.

E.2 Coder et tester la méthode permettant le recadrage de l'image de la cible.

On donne :

```
int Cible :: Arrondi(float r)
{
    int e=r;
    if(r-e>=0.5) e+=1;
    return e;
}
```



E.3 Quelles méthodes de la classe **SNImage** doivent être utilisées pour détecter les impacts ?

E.4 Coder et tester la méthode permettant la détection des impacts : les coordonnées des centres des impacts, et leurs tailles seront stockés dans des tableaux.



E.5 Dessiner le triangle rectangle dont l'hypoténuse est la distance entre le centre de l'impact et le centre de la cible. En déduire l'équation donnant la distance minimale entre le centre de la cible et l'impact (il faut tenir compte de la taille de l'impact).

E.6 Coder et tester la méthode permettant de déterminer la distance minimale de chaque impact du centre de la cible : en déduire le nombre de points de chaque impacte.

E.7 Afficher directement sur la cible le nombre de points de chaque impact ainsi que le score et le nom du tireur.

On donne :

```
string Cible ::IntToString(int n)
{
    char chaine[100];
    string schaine;
    sprintf(chaine,"%d",n);
    schaine=chaine;
    return schaine;
}
```



## F/ intégration des classes Cible, SNImage et Tireur

F.1 Concevoir sous C++ Builder ou QT creator l'interface homme-machine suivante :

F.2 Intégrer les classes **Cible**, **SNImage** et **Tireur**, puis coder entièrement l'interface.

## G/ Prise de la photo [ voir MS3 ]

Intégrer le code permettant de se connecter en HTTP à la caméra Heden afin d'obtenir et d'afficher l'image de la cible dans l'IHM.



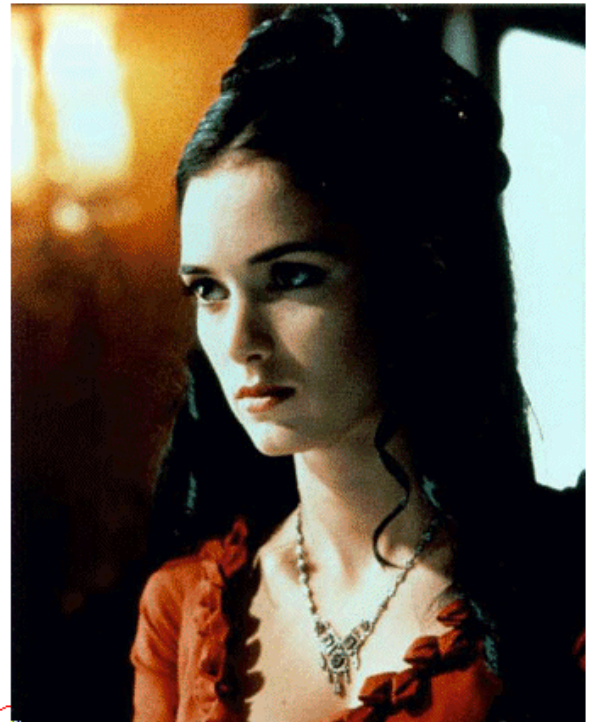
## H/ Affichage du score sur un journal lumineux [ voir MS1 ]

Intégrer les classes Afficheur et Ligne au projet et coder l'envoi du nom du joueur et du score au serveur TCP-IP connecté en USB au journal lumineux.





	largeur	69D92=433554	hauteur	=54 offset (début image)	nb de bits par pixel	-bitmap
0	42	4d	92	9d	06	00
10	00	00	53	01	00	00
20	00	00	5c	9d	06	00
30	00	00	00	00	00	00
40	ff	ff	80	80	80	ff
50	03	02	02	02	01	01
60	05	00	01	05	00	01
70	00	01	05	00	01	05
80	01	05	00	01	05	00
90	05	00	01	05	00	01
a0	00	01	05	00	01	05
b0	01	05	00	01	05	00
c0	05	00	01	05	00	01
d0	00	01	05	00	01	05
e0	01	03	01	01	03	01
f0	00	01	01	00	01	01
100	06	01	04	05	00	05
110	ac	5a	82	d4	55	88
120	53	6d	d7	50	72	d0
130	63	d3	33	5d	ce	2a
140	b6	0b	36	af	07	2c
150	19	18	8c	17	11	80
160	9a	d7	70	92	ce	65
170	c2	33	66	c7	31	68
180	1f	5a	be	27	5f	c4
190	21	7e	00	14	6f	09
1a0	67	0e	08	5f	15	13
1b0	05	03	3e	01	00	35
1c0	0a	39	06	0c	3b	06
1d0	2c	04	0a	33	02	19
1e0	97	b7	ce	9e	b6	cc
1f0	b9	d4	9a	b7	d2	96
200	d5	98	b6	d1	97	b5
210	90	ac	cf	97	b2	d7
410	01	01	01	01	01	01
420	01	01	01	01	01	01
430	00	00	00	ff	ff	80
440	00	c0	c0	c0	03	01
450	03	01	01	05	00	01
460	00	01	05	00	01	05
470	01	05	00	01	05	00
480	05	00	01	05	00	01
490	00	01	05	00	01	05
4a0	01	05	00	01	05	00
4b0	05	00	01	05	00	01
4c0	00	01	05	00	01	05
4d0	01	05	00	01	05	00
4e0	01	01	01	01	01	01
4f0	01	01	05	00	00	02
500	00	01	00	02	00	02
510	4c	84	d5	46	7e	cb
520	79	d3	43	78	c8	39
530	c0	20	5e	b8	18	57
540	03	35	ab	02	2e	a3



Taille sur le disque : 433 554 octets  
Résolution : 96 x 96 points par pouce  
Largeur : 339 Hauteur : 425

- \* taille image =(339\*3+k)\*425
- k choisi pour que 339\*3+k soit multiple de 4 (ici k=3)
- \* premier pixel
- \* résolution pour l'impression pixel/m (largeur)

Attention en fin de ligne k octets à 0 sont ajoutés dans le fichier

## ANNEXE 2 : classe SNImage, constructeur, et méthode Chargement

```
class SNImage
{ private: unsigned long dimensionMax;
  unsigned long tailleFile,offset;
  unsigned long tailleImage,tailleTete,zero,format,resoLarg,resoHaut;
  void ChangeCouleurSegment(Coordonnee coord,int taille,int epaisseur,Pixel couleur,char segment); //segment='a' 'b' 'c' 'd' 'e' 'f'
protected:
  Pixel **image;
  unsigned short sign;
  unsigned long largeur,hauteur;
public:
  SNImage();
  ~SNImage();
  inline unsigned short Signature(){return sign;}
  inline unsigned long Largeur(){return largeur;}
  inline unsigned long Hauteur(){return hauteur;}
  inline unsigned long TailleFichier(){return tailleFile;}
  inline unsigned long Offset(){return offset;}
  inline unsigned long TailleImage(){return tailleImage;}
  inline unsigned long TailleEntete(){return tailleTete;}
  inline unsigned long Format(){return format;}
  inline unsigned long ResolutionLargeur(){return resoLarg;}
  inline unsigned long ResolutionHauteur(){return resoHaut;}
  void Chargement(string nomFichier);
  void Sauvegarde(string nomFichier);
  void Tourne90Droite();
  void Tourne90Gauche();
  void Tourne180();
  void Negatif();
  void RetourneHorizontal();
  void RetourneVertical();
  void NiveauGris();
  void Eclaircit(int niveau);
  void Assombrit(int niveau);
  void SeuilleNoirBlanc(int niveau);
  void Colorie(Coordonnee coord,int l,int h,Pixel couleur);
  void Detoure(int largeurGauche,int largeurDroite,int largeurHaut,int largeurBas,Pixel couleur);
  void Recadre(Coordonnee coord,int l,int h);
  Coordonnee RechercheZone(int l,int h,Pixel couleur);
  void DessineCroix(Coordonnee coord,int taille,int epaisseur,Pixel couleur);
  void DessineCarre(Coordonnee coord,int taille,int epaisseur,Pixel couleur);
  void Dessine7Segments(Coordonnee coord,int taille,int epaisseur,Pixel couleur,string message);};
```

```
SNImage::SNImage() { sign=0;largeur=0;hauteur=0;tailleFile=0;offset=0;tailleImage=0; tailleTete=0;format=0;resoLarg=0;resoHaut=0;dimensionMax=0;}
```

```
void SNImage::Chargement(string nomFichier)
{
  ifstream entree;      long i,j;char k; long h;
  unsigned long dimensionMaxPrecedente=dimensionMax;
  entree.open(nomFichier.c_str(),fstream::binary);
  //lecture entête image:
  entree.read((char*)&sign,2*sizeof(char));//signature (BM)=424D
  entree.read((char*)&tailleFile,sizeof(long));
  entree.read((char*)&zero,sizeof(long));
  entree.read((char*)&offset,sizeof(long));
  entree.read((char*)&tailleTete,sizeof(long));
  entree.read((char*)&largeur,sizeof(long));
  entree.read((char*)&hauteur,sizeof(long));
  entree.read((char*)&format,sizeof(long));
  entree.read((char*)&zero,sizeof(long));
  entree.read((char*)&tailleImage,sizeof(long));
  entree.read((char*)&resoLarg,sizeof(long));
  entree.read((char*)&resoHaut,sizeof(long));
  entree.read((char*)&zero,sizeof(long));
  entree.read((char*)&zero,sizeof(long));
  //calcul de k
  k=(4-(largeur*3)%4)%4;
  //remplissage tableau 2 dimensions
  i=hauteur-1;j=0;//i indice des lignes
  if(sign!=0x4D42) cout<<"Vous devez choisir un bitmap .bmp";
  else if(format!=0x180001) cout<<"Vous devez choisir un bitmap 24 bits";
  else
  { //libération éventuelle de la mémoire occupée par l'ancienne image
    if(dimensionMaxPrecedente)
    {
      for(h=0;h<dimensionMaxPrecedente;h++) delete []image[h];
      delete []image;
    }
    if(hauteur>largeur) dimensionMax=hauteur; else dimensionMax=largeur;
    //réservation de la mémoire
    image=new Pixel*[dimensionMax];//i
    for(h=0;h<dimensionMax;h++) image[h]=new Pixel[dimensionMax];//j
    while(!entree.eof())
    {
      entree.read((char*)&image[i][j].bleu,sizeof(char));
      entree.read((char*)&image[i][j].vert,sizeof(char));
      entree.read((char*)&image[i][j].rouge,sizeof(char));      j++;
      if(j==largeur)
      {
        entree.seekg(k,ios_base::cur);
        j=0;i--;
      }
    }
  }
  entree.close();
}
```

## ANNEXE 3 : méthodes de dessin et de recherche sur l'image

```
void SNImage::Colorie(Coordonnee coord,int l,int h,Pixel couleur)
{
    if((coord.ligne+h<=hauteur)&&(coord.colonne+l<=largeur))
        for(int i=coord.ligne;i<coord.ligne+h;i++) for(int j=coord.colonne;j<coord.colonne+l;j++)
            {
                image[i][j].bleu=couleur.bleu;
                image[i][j].vert=couleur.vert;
                image[i][j].rouge=couleur.rouge;
            }
}
```

```
void SNImage::Detoure(int largeurGauche,int largeurDroite,int largeurHaut,int largeurBas,Pixel couleur)
{
    Coordonnee coor;
    coor.ligne=0;coor.colonne=0;
    Colorie(coor,largeurGauche,hauteur,couleur);
    Colorie(coor,largeur,largeurHaut,couleur);
    coor.ligne=0;coor.colonne=largeur-largeurDroite;
    Colorie(coor,largeurDroite,hauteur,couleur);
    coor.ligne=hauteur-largeurBas;coor.colonne=0;
    Colorie(coor,largeur,largeurBas,couleur);
}
```

```
void SNImage::Recadre(Coordonnee coord,int l,int h)
{
    if((coord.ligne+h<=hauteur)&&(coord.colonne+l<=largeur))
    {
        for(int i=0;i<h;i++)
            for(int j=0;j<l;j++)
                {
                    image[i][j]=image[i+coord.ligne][j+coord.colonne];
                }
        largeur=l;hauteur=h;
        int k=(4-(largeur*3)%4)%4;
        tailleImage=(largeur+k)*(hauteur)*3;
        tailleFile=tailleImage+offset;
    }
}
```

```
Coordonnee SNImage::RechercheZone(int l,int h,Pixel couleur)
{
    Coordonnee coord={-1,-1};
    int i,j,m,n;
    bool zoneTrouve;
    for(int i=0;i<hauteur;i++)
    {
        for(int j=0;j<largeur;j++)
        {
            if(image[i][j].bleu==couleur.bleu && image[i][j].vert==couleur.vert && image[i][j].rouge==couleur.rouge)
            {
                zoneTrouve=true;
                for(int m=0;m<h;m++) for(int n=0;n<l;n++)
                {
                    if(image[i+m][j+n].bleu!=couleur.bleu || image[i+m][j+n].vert!=couleur.vert || image[i+m][j+n].rouge!=couleur.rouge)
                        zoneTrouve=false;
                }
            }
            if(zoneTrouve)
            {
                coord.ligne=i; coord.colonne=j;
                i=hauteur; j=largeur;
            }
        }
    }
    return coord;
}
```

```
void SNImage::DessineCroix(Coordonnee coord,int taille,int epaisseur,Pixel couleur)
{
    Coordonnee debut;
    debut.ligne=coord.ligne-taille/2;debut.colonne=coord.colonne-epaisseur/2;
    if(debut.ligne<0) debut.ligne=0;
    if(debut.colonne<0) debut.colonne=0;
    Colorie(debut,epaisseur,taille,couleur);
    debut.ligne=coord.ligne+epaisseur/2;debut.colonne=coord.colonne-taille/2;
    if(debut.ligne<0) debut.ligne=0;
    if(debut.colonne<0) debut.colonne=0;
    Colorie(debut,taille,epaisseur,couleur);
}
```

```
void SNImage::DessineCarre(Coordonnee coord,int taille,int epaisseur,Pixel couleur)
{
    Coordonnee debut=coord;
    debut.ligne==epaisseur/2; debut.colonne==epaisseur/2;
    Colorie(debut,epaisseur,taille,couleur); //a ou e
    debut.colonne+=taille;
    Colorie(debut,epaisseur,taille,couleur); //b ou c
    debut=coord;
    debut.ligne==epaisseur/2; debut.colonne==epaisseur/2;
    debut.ligne+=taille;
    Colorie(debut,taille+epaisseur,epaisseur,couleur); //g ou d*/
}
```








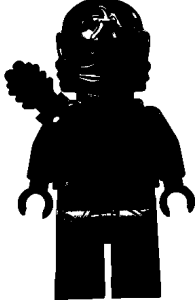










```
void SNImage::ChangeCouleurSegment(Coordonnee coord,int taille,int epaisseur,Pixel couleur,char segment)//segment='a' 'b' 'c' 'd' 'e' 'f'
{
    Coordonnee debut;
    switch (segment)
    {
        case 'a': debut=coord; debut.ligne==epaisseur/2; debut.colonne==epaisseur/2; Colorie(debut,taille/2+epaisseur,epaisseur,couleur); break;
        case 'b': debut=coord; debut.ligne==epaisseur/2; debut.colonne==epaisseur/2; debut.colonne+=taille/2; Colorie(debut,epaisseur,taille/2+epaisseur,couleur); break;
        case 'f': debut=coord; debut.ligne==epaisseur/2; debut.colonne==epaisseur/2; Colorie(debut,epaisseur,taille/2+epaisseur,couleur); break;
        case 'g': debut=coord; debut.ligne==epaisseur/2; debut.colonne==epaisseur/2; debut.ligne+=taille/2; Colorie(debut,taille/2+epaisseur,epaisseur,couleur); break;
        case 'c': debut=coord; debut.ligne==epaisseur/2-taille/2; debut.colonne==epaisseur/2; debut.colonne+=taille/2; Colorie(debut,epaisseur,taille/2+epaisseur,couleur); break;
        case 'e': debut=coord; debut.ligne==epaisseur/2-taille/2; debut.colonne==epaisseur/2; Colorie(debut,epaisseur,taille/2+epaisseur,couleur); break;
        case 'd': debut=coord; debut.ligne==epaisseur/2-taille/2; debut.colonne==epaisseur/2; debut.ligne+=taille/2; Colorie(debut,taille/2+epaisseur,epaisseur,couleur); break;
    }
}
```

```


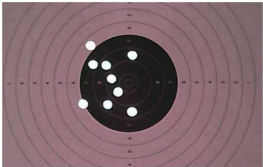
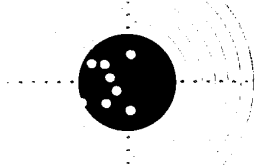
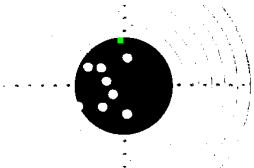
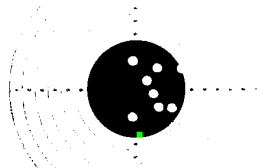
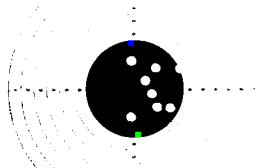
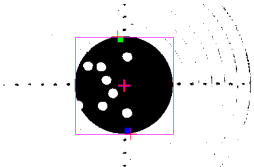
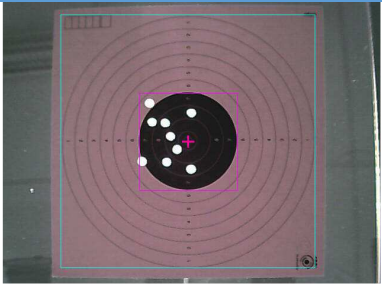

void SNImage::Dessine7Segments(Coordonnee coord,int taille,int epaisseur,Pixel couleur,string message)
{
    int ligneDepart=coord.ligne;
    for(int i=0;i<message.length();i++)
    {
        coord.ligne=ligneDepart;
        switch (message.c_str()[i])
        {
            case '0': case 'O':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f'); break;
            case '1': case 'l':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c'); break;
            case '2':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case '3':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case '4':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case '5': case 's': case 'S':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case '6':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case '7':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c'); break;
            case '8': case 'B':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case '9':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case 'p': case 'P':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'a');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'b');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case 'o':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case 'i':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c'); break;
            case 'n':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'c');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
            case 't':
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'d');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'e');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'f');
                ChangeCouleurSegment(coord,taille,epaisseur,couleur,'g'); break;
        }
        coord.colonne+=2*(taille/2);
        coord.ligne=ligneDepart;
    }
}

```

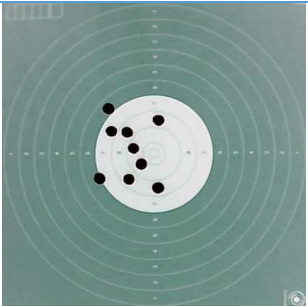





			
Originale	Négatif	Eclaircissement	Assombrissement
			
Niveaux de gris	Seuillage 50	Seuillage 128	Seuillage 200
			
Coloration	Détourage	Détection de zones	Détection de zones
			
Miroir horizontal	Miroir vertical	Recadrage	
			
Rotation 90°	Rotation 180°	Rotation 270°	



## 1] Détection du centre noir : calcul des coordonnées du bord de la cible

		
Originale	Détourage	Seuillage
		
Détection de zone	Rotation 180°	Détection de zone
		
Calcul du diamètre et du centre	Calcul du bord de la cible	Recadrage

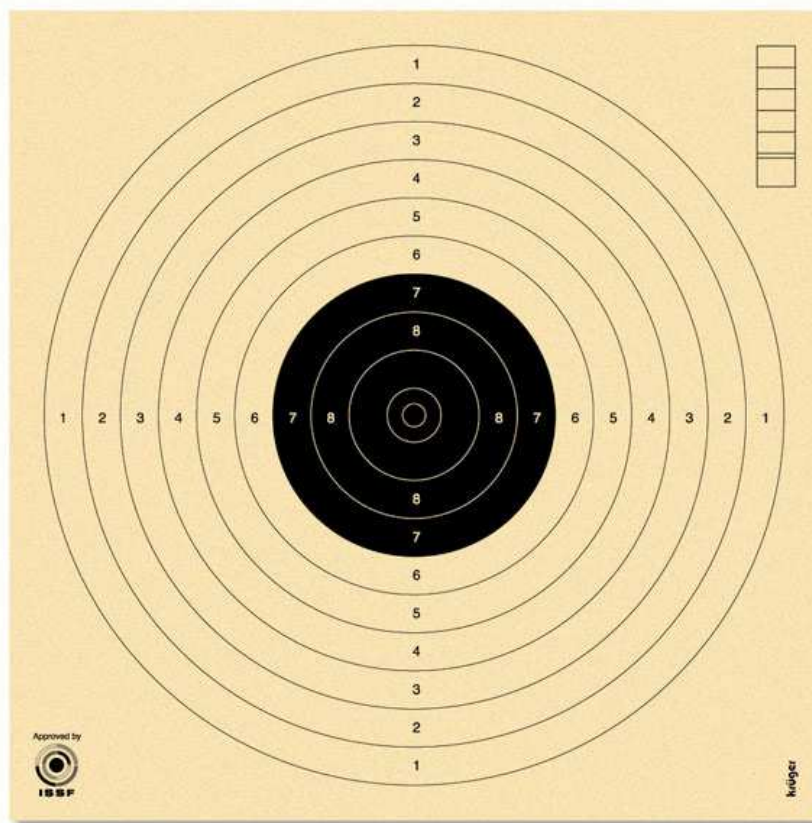
## 2] Création d'une nouvelle image centrée sur la cible et détection des impacts

			
Négatif	Seuillage	Détection de zone et coloration	Détection et coloration de zone à 180°

## 3] Calcul du score

	
Calcul du centre et de la taille des impacts, calcul de la position des cercles concentriques	Calcul et affichage du nombre de points de chaque impact et calcul du score

## ANNEXE 6 : normalisation de la cible pour pistolet à air à 10 m



Diamètre en mm des zones et approximation des mesures.

10 9 8 7 6 5 4 3 2 1

11,5 27,5 43,5 59,5 75,5 91,5 107,5 123,5 139,5 155,5

+/-0,1 +/-0,2 +/-0,2 +/-0,5 +/-0,5 +/-0,5 +/-0,5 +/-0,5 +/-0,5 +/-0,5

Mouche: 5 mm (+/- 0,1)

Diamètre du visuel noir: 59,5 mm de la zone 7 à la zone 10.

Épaisseur des cordons: 0,1 mm à 0,2 mm.

Dimensions minimales visibles du carton : 17 x 17 cm.

La valeur des zones 1 à 8 est imprimée dans les plans horizontaux et verticaux et les zones 9 et 10 ne portent pas de numéro.

Les chiffres ne doivent pas avoir une hauteur supérieure à 2 mm.







tortueGeniale.bmp



tortueGeniale.jpg



tortueGeniale.png

HexEdit - C:\Users\diervese\Desktop\Xe6imageTest2\tortueGeniale.bmp

File	Edit	Find	View	About	
0	42	4d	2a	8f 04 00 00 00 00 36 00 00 00 28 00	BM*.....6....(.
10	00	00	04	01 00 00 7f 01 00 00 01 00 18 00 00 00	.....
20	00	00	f4	8e 04 00 c4 0e 00 00 c4 0e 00 00 00 00	.....
30	00	00	00	00 00 00 ff ff ff ff ff ff ff ff ff	.....
40	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
50	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
60	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
70	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
80	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
90	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
a0	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
b0	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
c0	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....
d0	ff	ff	ff	ff ff ff ff ff ff ff ff ff ff ff ff ff	.....

HexEdit - C:\Users\diervese\Desktop\Xe6imageTest2\tortueGeniale.jpg

File	Edit	Find	View	About	
0	ff	d8	ff	e0 10 4a 46 49 46 00 01 01 00 00 01	.....JFIF.....
10	00	01	00	00 ff db 00 43 00 08 06 06 07 06 05 08	.....C.....
20	07	07	07	09 09 08 0a 0c 14 0d 0c 0b 0b 0c 19 12	.....
30	13	0f	14	1d 1a 1f 1e 1d 1a 1c 1c 20 24 2e 27 20	.....\$. '
40	22	2c	23	1c 1c 28 37 29 2c 30 31 34 34 34 1f 27	.....",#..(7),01444.'
50	39	3d	38	32 3c 2e 33 34 32 ff db 00 43 01 09 09	.....9=82<.342...C...
60	09	0c	0b	0c 18 0d 0d 18 32 21 1c 21 32 32 32 32	.....2!.12222
70	32	32	32	32 32 32 32 32 32 32 32 32 32 32 32 32	.....2222222222222222
80	32	32	32	32 32 32 32 32 32 32 32 32 32 32 32 32	.....2222222222222222
90	32	32	32	32 32 32 32 32 32 32 32 32 32 32 32 32	.....2222222222222222
a0	00	11	08	01 7f 01 04 03 01 22 00 02 11 01 03 11	....."
b0	01	ff	c4	00 1f 00 00 01 05 01 01 01 01 01 01 00	.....
c0	00	00	00	00 00 00 01 02 03 04 05 06 07 08 09	.....
d0	0a	0b	ff	c4 00 b5 10 00 02 01 03 03 02 04 03 05	.....
e0	05	04	04	00 00 01 7d 01 02 03 00 04 11 05 12 21	.....}.....!
f0	31	41	06	13 51 61 07 22 71 14 32 81 91 a1 08 23	.....1A...Qa.."q.2....#

HexEdit - C:\Users\diervese\Desktop\Xe6imageTest2\tortueGeniale.png

File	Edit	Find	View	About	
0	89	50	4e	47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52	..PNG.....IHDR
10	00	00	01	04 00 00 01 7f 08 06 00 00 00 af ed b7	.....
20	08	00	00	00 01 73 52 47 42 00 ae ce 1c e9 00 00	.....sRGB.....
30	00	04	67	41 4d 41 00 00 b1 8f 0b fc 61 05 00 00	.....gAMA.....a...
40	00	09	70	48 59 73 00 00 0e c4 00 00 0e c4 01 95	.....pHYs.....
50	2b	0e	1b	00 00 ff a5 49 44 41 54 78 5e ec bd 07	.....+.....IDATx^...
60	60	54	d5	d6 b7 1f 7b 6f 28 45 7a ef 48 15 51 41	.....T....{o(Ez.H.QA
70	a5	89	48	af 02 02 a2 34 29 16 50 40 7a ef 35 b4	.....H....4).P@z.5.
80	d0	02	21	21 84 90 0a 09 09 90 42 42 4d 23 bd f7	.....!!.....BBM#..
90	de	7b	af	3c ff b5 4f 08 a2 d7 fb be ef f7 7a ff	.....{.<...O.....z.
a0	9f	de	fb	cd d2 c5 99 39 33 99 39 73 ce 5e cf fe	.....93.9s.~..
b0	ad	7d	76	d1 43 67 3a d3 99 ce 1e 9a 0e 08 3a d3	.....}v.Cg:.....
c0	99	ce	1e	99 0e 08 3a d3 99 ce 1e 99 0e 08 3a d3	.....
d0	99	ce	1e	99 0e 08 3a d3 99 ce 1e 99 0e 08 3a d3	.....
e0	99	ce	1e	99 0e 08 3a d3 99 ce 1e 99 0e 08 3a d3	.....
f0	99	ce	1e	99 0e 08 3a d3 99 ce 1e 99 0e 08 3a d3	.....