UNIVERSITÉ PARIS SUD

Examen du 20 Decembre 2013

Licence MPI. Info 112 (IGOR)

F. Vernier, C Perin & C. Jacquemin

1. Dessin & Interaction (10 pts)

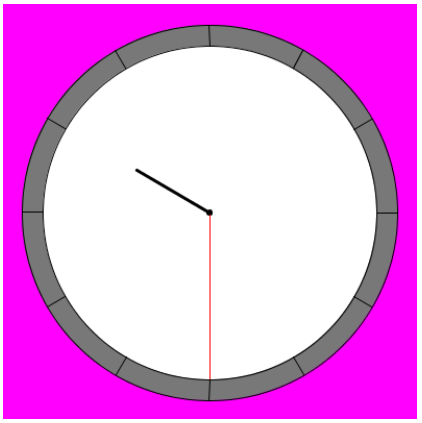


Figure 1 : L'horloge complète

Dans cet exercice, il vous est demandé de donner le code en langage Processing permettant de réaliser l'horloge ci-contre. Pour cela, vous disposez des fonctions Processing suivantes :

- `line(x1,y1,x2,y2)`
- `ellipse(x,y,w,h)`
- `stroke(r,g,b,a)`
- `strokeWeight(t)`
- `fill(r,g,b,a)`

Ainsi que les autres fonctions standards du langage Processing (cos, atan2, …).

Le squelette suivant vous est fourni :

```
int r_int = 160; //rayon interieur
int r_ext = 180; //rayon exterieur
int heures = 10;
int minutes = 30;
void setup() {
      size(400, 400);
      smooth();
      noLoop();
}
void draw() {
      background( ____ , ____ , ____ ); //question 1.1
}
```

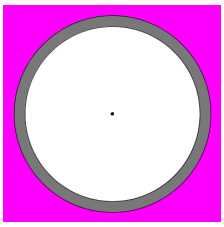
1.1 (0.5pt) Compléter l'appel à la fonction `background` dans le squelette en remplissant les valeurs des 3 paramètres afin d'obtenir la couleur **magenta**.

1.2 (0.5pt) Quel est le code **hexadécimal** du magenta?

_____

_____

1.3 (2pts) Donner le code dans la fonction `draw()` pour dessiner les éléments de l'horloge suivants, en vous servant des variables fournies dans le squelette et en évitant d'utiliser des constantes comme 200 ou 400 !:



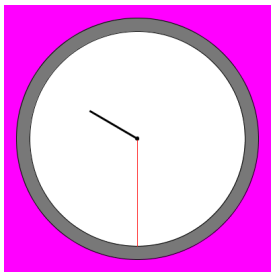
1.3.1 Le point central, de rayon 5 pixels et de couleur noire

void draw {_____

_____

1.3.2 Le **cercle** extérieur et le **cercle** intérieur, d'épaisseur 1 pixel et de couleur noire, avec **l'anneau** externe (le cadre) de couleur grise et le **disque** interne de couleur blanche.

1.4 (1,5pt) Continuer le code dans la fonction `draw()` pour dessiner la grande et la petite aiguille de manière à ce que l'horloge indique 10h30 (en utilisant les variables `heures` et `minutes` pour définir les angles des aiguilles par rapport à la verticale), avec :



- Aiguille des minutes en rouge, d'épaisseur 1 pixel, de longueur `r_int`
- Aiguille des heures en noir, épaisseur 3 pixels, de longueur égale à la moitié de la longueur de l'aiguille des minutes
  ATTENTION : à 10h30 la petite aiguille est sur le 10 **exactement** !!!

1.5 (1.5pt) Continuer le code dans la fonction draw() pour dessiner les 12 marqueurs indiquant chacune des 12 heures, en noir, d'une épaisseur de 1 pixel comme illustrés à la Figure 1 .

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

1.6 (2pt) On ajoute au programme la fonction `mouseClicked()`. Compléter cette fonction afin que l'heure (et les minutes si possible) change en fonction de l'angle entre l'horizontale et l'axe centre de l'horloge <-> la position du click (si on clique à droite du point central on doit mettre l'horloge à 3h)

```
void mouseClicked() {
```

_____

_____

_____

_____

_____

```
}
```

1.7 (2pts) Que modifier au programme pour que la variable des minutes s'incrémente (de 1) toutes les 1/20 secondes et donc que celle des heures s'incrémente (de 1) toutes les 3 secondes :

_____

_____

_____

_____

_____

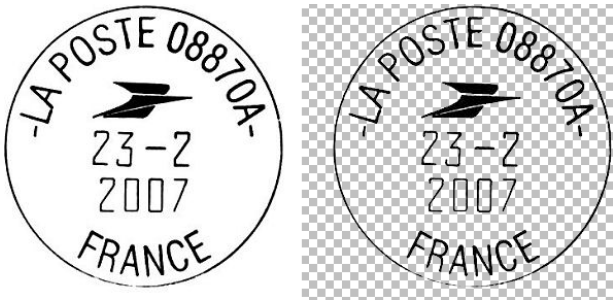
2 Traitement d'image (5 pts)

Soit la fonction suivante qui copie une image (src) sur une autre (dst) en remplaçant la transparence de l'image src par une opacité totale:

```
void blitimg(PImage dst, PImage src) {
  dst.loadPixels();
  for (int j=0; j<src.height; j++) {
    for (int i=0; i<src.width; i++) {
      int c = src.pixels[i+j*src.width];
      float r = red(c);
      float g = green(c);
      float b = blue(c);
      dst.pixels[i+j*dst.width] = color(r,g,b, 255) ;
    }
  }
  dst.updatePixels();
}
```

2.1 (2pt) Compléter le code suivant afin de transformer une image en noir & blanc et image noir& transparent (ou les pixels blancs deviennent transparents). Les pixels gris (50%) doivent devenir noir à demi-transparent), les pixels gris (25%) doivent devenir noir à un quart transparent et ainsi de suite.

```
void blitimg(PImage dst, PImage src) {
  dst.loadPixels();
  for (int j=0; j<src.h; j++) {
    for (int i=0; i<src.w; i++) {
      int c = src.pixels[i+j*src.width];
      float r = red(c);
      float g = green(c);
      float b = blue(c);
```



_____

_____

_____

_____

```
      dst.pixels[i+j*dst.width] = _____________________________
    }
  }
  dst.updatePixels();
}
```

2.1 (3pt) Compléter le code suivant afin de transformer une image couleur en image couleur avec de la transparence calculée en fonction de la saturation et de la luminance de chacun des pixels. Pour ce faire, vous devrez utiliser la fonction saturation(c) qui retourne la saturation d'une couleur et vous devrez calculer la luminance de la couleur de chaque pixel (son niveau de gris). La saturation du blanc, du gris ou du noir est de 0 et la saturation d'une couleur vive est de 1. On souhaite qu'un pixel dont la luminance vaut respectivement 0%, 25%, 50%, 75% et 100% devienne opaque à respectivement 100%, 75%, 50%, 25%, 0%. Par ailleurs un pixel dont la saturation vaut respectivement 0%, 25%, 50%, 75% et 100% devient opaque à 0%, 25%, 50%, 75% et 100%. Enfin on souhaite que ces deux effets se multiplient pour un pixel avec une luminosité et une saturation donnée.

```
void blitimg(PImage dst, PImage src) {
  dst.loadPixels();
  for (int j=0; j<src.height; j++) {
    for (int i=0; i<src.width; i++) {
      int c = src.pixels[i+j*src.width];
      float r = red(c);
      float g = green(c);
      float b = blue(c);
```

_____

_____

_____

_____

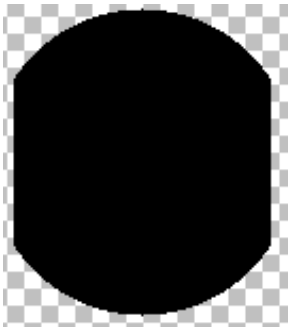
```
      dst.pixel[i+j*dst.width] = _____
    }
  }
  dst.updatePixels();
}
```

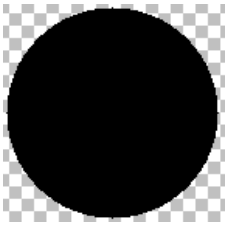
3 Création d'image (2 pts)

Soit la fonction suivante qui crée une image de cercle remplie

```
PImage createImg(int w, int h, int r, int g, int b) {
  PImage img = createImage(w, h, ARGB) ;
  img.loadPixels();
  for (int j=0; j<img.height; j++) {
    for (int i=0; i<img.width; i++) {
      float  d = dist(i,j,w/2, h/2) ;
      if (d<=min(w/2, h/2))
        img.pixels[i+j* img.width] = color(r,g,b, 255);
      else
        img.pixels[i+j* img.width] = color(0,0,0,0);
    }
  }
  img.updatePixels();
return img;
}
```

3.1 (1pt) Si les dimensions (w,h) ne forment pas un carré, le cercle est-il coupé su le bord le plus court OU BIEN est-il rétréci pour que le rayon corresponde à la plus petite des deux dimensions ?





_____

_____

3.2 (1pt) Que faut-il modifier dans le code pour obtenir le comportement décrit ci-dessus qui ne correspond pas à celui du code fourni ?

_____

_____

4 Affichage (3 pts)

Soit la fonction draw() suivante qui affiche une image transparente img2 telle que celle produite a la question 2.1 ou 3.1

```
void draw(){
  background(255);
  fill(192);
  noStroke();
  for (int j=0; j<height/8; j++)
    for (int i=0; i<width/8; i++)
      if ((i+j)%2==0)
        rect(i*8, j*8, 8, 8);

  image(img2, width-img2.width, img2.height/8);
}
```

4.1 A quoi servent les deux boucles **for**, le **if** et le **rect** dans cette fonction ?

_____

_____

_____

_____

_____

_____