POUR CONTINUER AVEC APDE (PROCESSING POUR ANDROID)

1. Animation simple : un point se déplace horizontalement sur un fond noir

- on commence par déclarer une variable de type nombre entier (int) à laquelle on donne le nom que l'on souhaite (ici, « x »),
- on crée le « void setup », c'est un groupement d'instructions qui ne seront lues qu'une seule fois, au lancement de l'animation. (Les caractères spéciaux « { » et « } » se trouvent dans la barre grise au-dessus du clavier virtuel),
- « orientation » est utile pour les tablettes : même si elles changent de position au cours de l'animation, celle-ci ne sera pas arrêtée et continuera de s'effectuer dans le sens prédéterminé (LANDSCAPE ou PORTRAIT),
- on crée ensuite le « void draw », groupement d'instructions qui seront lues par défaut 60 fois par seconde, aussi longtemps que l'animation est en cours,
- x=x+1; veut dire que la nouvelle valeur de « x » sera égale à l'ancienne valeur de « x » à laquelle on ajoute 1. Ainsi, au bout de 1 seconde d'animation, « x » vaudra 60,
- on utilise la variable « x » pour déterminer la position horizontale de notre point.

```
int x;

void setup() {
    orientation(LANDSCAPE);
    background(0);
}

void draw() {
    background(0);
    stroke(255);
    strokeWeight(5);
    x=x+1;
    point(x,100);
}
```

2. **Faire revenir le point à sa position initiale quand il sort de l'écran**

- admettons que la longueur de la tablette fasse 1024 pixels, on introduit une condition : si « x » est supérieur à 1024, alors « x » devient 0.

```
int x;

void setup() {
    orientation(LANDSCAPE);
    background(0);
}

void draw() {
    background(0);
    stroke(255);
    strokeWeight(5);
    x=x+1;
    if(x>1024) {
        x=0 ;
    }
    point(x,100);
}
```

3. **Faire changer la direction du point quand il arrive à l'une des deux extrémités de l'écran**

    - on déclare une nouvelle variable que l'on nomme « vitesse »,
    - dans le « void setup », on lui donne la valeur initiale de 1. Si l'on ne spécifie pas de valeur initiale (comme on l'a fait pour « x »), cette valeur sera de zéro,
    - on modifie la condition qui devient : si « x » est supérieur à 1024 ou si « x » est inférieur à 0, alors la nouvelle valeur de la vitesse devient l'inverse de la précédente. (Le caractère spécial « | » se trouve dans la barre grise au-dessus du clavier virtuel).

```
int x, vitesse;

void setup() {
    orientation(LANDSCAPE);
    background(0);
    vitesse=1 ;
}

void draw() {
    background(0);
    stroke(255);
    strokeWeight(5);
    x=x+vitesse;
    if(x>1024 || x<0) {
        vitesse=-vitesse ;
    }
    point(x,100);
}
```

4. **Faisons se déplacer le point dans les deux dimensions, de façon aléatoire**

- transformons la nature de nos variables pour qu'elles acceptent les décimales (« float » désigne une variable de type décimal),
- créons « y » et « vitessey », et transformons « vitesse » en « vitessex »,
- demandons à l'ordinateur, une seule fois au début de l'animation, de donner à « vitessex » une valeur comprise entre 1 et 2,
- faisons de même avec « vitessey »,
- appliquons les mêmes évolutions à « x » et à « y » dans le « void draw »,
- appliquons les mêmes conditions à « x » et à « y » dans le « void draw »,
- utilisons la variable « y » pour déterminer la position verticale du point.

```
float x, y, vitessex, vitessey;

void setup() {
    orientation(LANDSCAPE);
    background(0);
    vitessex=random(1,2) ;
    vitessey=random(1,2) ;
}

void draw() {
    background(0);
    stroke(255);
    strokeWeight(5);
    x=x+vitessex;
    y=y+vitessey ;
    if(x>1024 || x<0) {
        vitessex=-vitessex ;
    }
    if(y>768 || y<0) {
        vitessey=-vitessey ;
    }
    point(x,y);
}
```

A présent, on supprime le « background(0); » du « void draw » et on obtient une première machine à dessiner en laissant la trace du point.

5. Perfectionnons notre machine à dessiner en appliquant une rotation

- appliquons un « translate », instruction qui permet de modifier la position du point de référence. Le point (0, 0) n'est plus en haut à gauche de l'écran mais en son centre. Cette opération va nous permettre de faire tourner l'animation autour de ce point central,
- changeons les nombres dans les conditions. En effet, maintenant, le côté gauche de l'écran n'est plus à 0 mais à -512, le côté droit à 512, le haut à -384 et le bas à 384,
- demandons au logiciel de faire tourner l'animation autour du nouveau point de référence à raison d'un millième de la valeur de « x » (ou de toute autre variable que l'on créera à cet effet).

```
float x, y, vitessex, vitessey;

void setup() {
    orientation(LANDSCAPE);
    background(0);
    vitessex=random(1,2);
    vitessey=random(1,2);
}

void draw() {
    stroke(255);
    strokeWeight(5);
    x=x+vitessex;
    y=y+vitessey ;
    translate(512, 384);
    if(x>512 || x<-512) {
        vitessex=-vitessex ;
    }
    if(y>384 || y<-384) {
        vitessey=-vitessey ;
    }
    rotate(x/1000);
    point(x,y);
}
```

6. Place à la créativité

- il est temps de devenir créatif et de jouer avec les couleurs, la taille du point, les valeurs de « vitessex » et de « vitessey », et de moduler la vitesse de la rotation,
- on pourra aussi remplacer le point par toute autre forme, ou combiner plusieurs formes,
- dans l'exemple suivant, on réduit la grosseur du point à 2 pixels et on crée une ligne très fine (0.1 pixel) qui relie le point au centre de l'image,
- on se sert des valeurs absolues de « x » et de « y » pour déterminer la couleur de la ligne.

```
float x, y, vitessex, vitessey;

void setup() {
    orientation(LANDSCAPE);
    background(0);
    vitessex=random(1,2);
    vitessey=random(1,2);
}

void draw() {
    stroke(255);
    strokeWeight(2);
    x=x+vitessex;
    y=y+vitessey ;
    translate(512, 384);
    if(x>512 || x<-512) {
       vitessex=-vitessex ;
    }
    if(y>384 || y<-384) {
       vitessey=-vitessey ;
    }
    rotate(x/1000);
    point(x,y);
    strokeWeight(0.1);
    stroke(abs(x+y),abs(x),abs(y));
    line(x,y,0,0);
}
```