Distances de pixels

Samuel Chatelier, Gabryelle Dutrey, Céleste Franke,
Swann Le Gall–Pérochon, Thibaut Le Gouëz,
Sterenn Rioual, Yoann Rose
classes de Seconde, Première et Terminale

Année 2024-2025

Établissement : Lycée de l'Harteloire, Brest
Enseignant : Jean-marie Gourmelon
Chercheur : Stéphane Rioual, Université de Bretagne Occidentale

1 Présentation du sujet

L'affichage d'un segment de droite reliant deux pixels sur un écran peut se faire suivant différents procédés, en voici un (qui n'est pas celui qui est le plus communément utilisé).

On considère un écran dans lequel tous les pixels sont des carrés contigus de côtés 1 et qui sont repérés par les coordonnées entières de leurs centres dans un repère orthonormé. On souhaite afficher le segment reliant les points $A(x_A\ ;\ y_A)$ et $B(x_B\ ;\ y_B)$. Pour cela on active tous les pixels qui sont traversés par le segment $[AB]$.

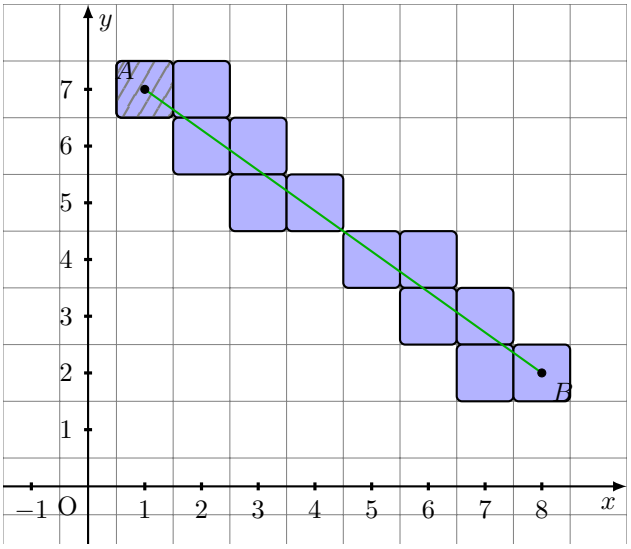

FIGURE 1 – Exemple de détermination d'une distance en pixels

La « distance entre deux pixels » peut alors être définie comme le nombre de pixels qu'il faut activer pour afficher le segment à l'écran, diminué de 1.

Ainsi par exemple la distance entre le pixel A et lui-même est égale à 0 $(= 1 - 1)$; la distance entre entre le pixel A et chacun de ses 8 pixels voisins est égale à 1 $(= 2 - 1)$; la distance entre les pixels A et B de la FIGURE 1 est égale à 11 $(= 12 - 1)$.

La définition de certains ensembles de points dans le plan euclidien fait intervenir des distances, par exemple le cercle, la médiatrice d'un segment, l'ellipse.

Quelle serait l'allure de ces ensembles de points sur un écran en utilisant cette « distance de pixels » ?

2 Démarche et résultats

Après une première séance de dessin et mesure de distances de pixels sur du papier quadrillé, nous avons immédiatement défini comme objectif d'obtenir des images de cercles de pixels en utilisant un langage de programmation comme Python. Pour cela nous avons divisé notre groupe en deux. Une partie travaillait à concevoir le programme et l'autre partie à la recherche d'une formule permettant de calculer la distance entre deux pixels sans compter les pixels sur un quadrillage.

Cette recherche nous a pris l'essentiel de notre temps mais nous avons réussi à obtenir un programme qui fonctionne pour le congrès d'Angers et nous avons pu finaliser les démonstrations validant la formule utilisée par la suite. Cependant nous n'avons pas eu le temps de valider ou d'invalider les conjectures émises suite aux premières observations, pour certaines d'elles nous avons simplement des pistes d'explication.

Lors du congrès d'Angers, un chercheur nous a demandé si cette distance de pixels était vraiment une distance et nous avons été surpris car nous n'avions pas réfléchi à cette question.

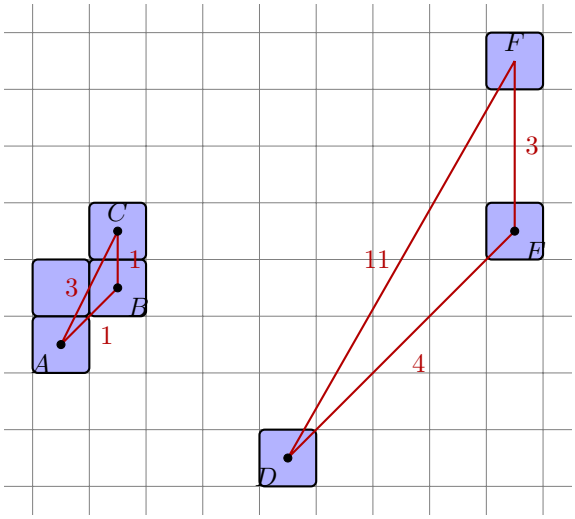

FIGURE 2 – Inégalité triangulaire non respectée

Nous avons ensuite cherché ce qu'est une distance et il est apparu que la caractérisation de la « distance en pixel » comme « distance » est un abus de langage, car une « distance » doit respecter les quatre axiomes suivants :

— *Axiome de séparation : La distance entre deux points est égale à 0 si et seulement si les deux points sont confondus.*

Le fait de ne pas compter le pixel d'origine du segment duquel on veut mesurer la longueur permet de respecter cet axiome.

— *Axiome de symétrie : La distance BA est égale à la distance AB.*

$[BA]$ et $[AB]$ sont confondus le même nombre de pixels sera donc activé.

— *Axiome de positivité : Une distance de deux points est toujours positive*

La plus petite distance possible (celle entre deux points confondus) est bien égale à 0.

— *Axiome de l'inégalité triangulaire : Dans un triangle ABC, $AB + BC \geqslant CA$.*

Or, on peut observer des triangles ABC tel que $AB + BC < CA$, deux exemples en sont donnés en FIGURE 2. Nous utiliserons malgré tout dans cet article l'expression « distance en pixels » ou « distance entre deux pixels » en ayant en tête que ceci ne désigne pas une distance mathématique.

3 Calcul de la distance entre deux pixels A et B

Dès les premières séances, nous avons décidé que nous travaillerons à partir des coordonnées d'un segment, que nous définirons clairement par la suite. L'idée était de pouvoir déterminer la distance en pixel sans compter manuellement chaque pixel traversé, mais uniquement grâce à ces coordonnées. Ainsi nous avons établi des tableaux comme ceux de la FIGURE 3.

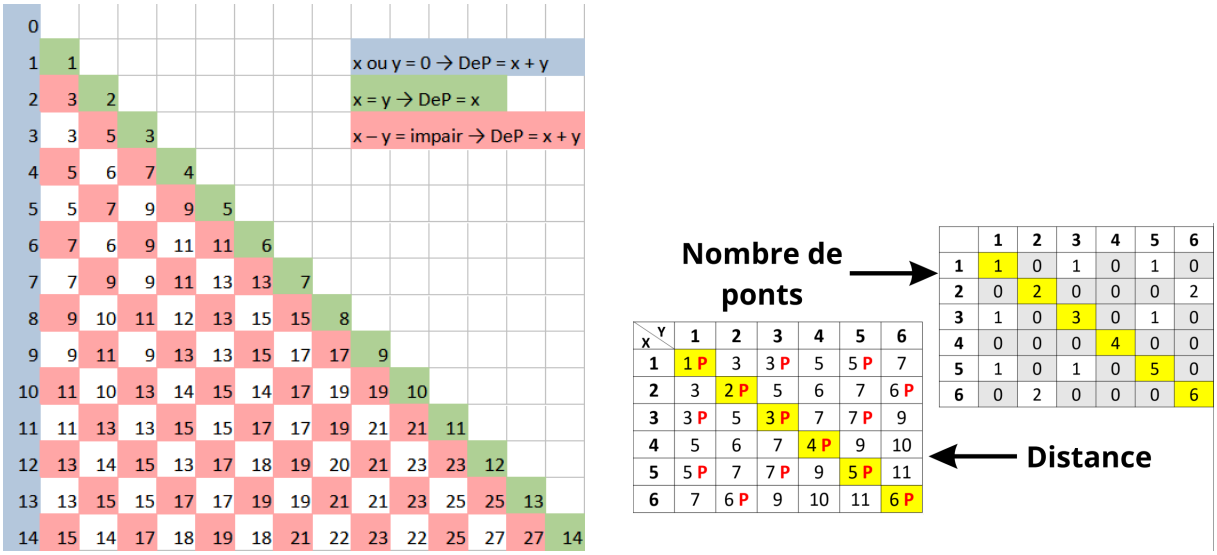

FIGURE 3 – Tableaux de distances obtenues « à la main »

À partir de ces tableaux, nous avons pu établir des conjectures notamment quant à la détermination de distance en pixel ou à la détermination du nombre de ponts (notion importante que nous verrons par la suite).

3.1 Définitions :

Pendant nos recherches nous avons utilisé entre nous des termes spécifiques pour désigner différents objets :

— le *plan pixelisé* est un plan dont chaque point de coordonnées entières est le centre d'un carré de côté 1 et dont ses côtés sont parallèles aux axes des abscisses et des ordonnées, appelés pixels. On associe naturellement les coordonnées du centre du pixel au pixel.
— un point M est considéré comme *appartenant à un pixel* si et seulement si il est situé à l'intérieur du pixel. Ainsi M n'est pas considéré comme appartenant au pixel s'il est situé sur un de ses côtés ou sommets.
— soient deux points de coordonnées entières $A(x_A; y_A)$ et $B(x_B; y_B)$. On appelle *coordonnées du segment* $[AB]$ les réels x et y tels que $x = |x_B - x_A|$ et $y = |y_B - y_A|$. Dans le cas où des coordonnées comme mesures algébrique sont nécessaires, comme pour les représentations paramétriques, on notera $\overline{x} = x_B - x_A$ et $\overline{y} = y_B - y_A$.
— un *passage* est un point d'un segment dont une des coordonnées est demi-entière, représentant le passage d'un pixel à un autre en parcourant le segment $[AB]$. Si n_{pass} est le nombre de passages et d la distance en pixel, $n_{\text{pass}} = d$.
— Si $x \geqslant y$, une *marche* est un point d'un segment d'ordonnée demi-entière ; si $x \leqslant y$, une marche est un point d'un segment d'abscisse demi-entière. Le nombre de marches d'un segment est égal à la plus petite des coordonnées du segment.
— Un *palier* est un passage qui n'est pas une marche.
— Un *pont* est un point d'un segment dont les deux coordonnées sont demi-entières. Un pont est donc toujours une marche.

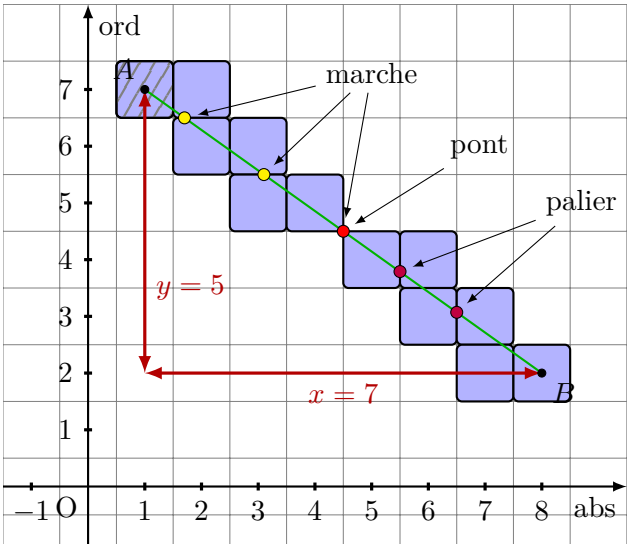

FIGURE 4 – Vocabulaire utilisé

3.2 Propriétés du segment

Propriété 1 : Soient deux points non confondus de coordonnées entières $A(x_A; y_A)$ et $B(x_B; y_B)$. $[AB]$ traverse au moins un pixel de chaque abscisse (*resp.* ordonnée) entière comprise entre x_A et x_B (*resp.* y_A et y_B).

Preuve :

On rappelle qu'un segment $[AB]$ est considéré comme traversant un pixel s'il existe au moins un point $M(x_M\ ;\ y_M)$ de $[AB]$ appartenant au pixel. Ainsi soit une représentation paramétrique

de $[AB]$:

$$\begin{cases} abs(t) = x_A + \overline{x}t \\ ord(t) = y_A + \overline{y}t \end{cases}, \ t \in [0;1]$$

$[AB]$ traverse le pixel de centre $M(x_M; y_M)$ si et seulement si :

$$\exists t \in [0;1] \text{ tel que } \begin{cases} x_M - 0.5 < abs(t) < x_M + 0.5 \\ y_M - 0.5 < ord(t) < y_M + 0.5 \end{cases}$$

abs est une fonction affine définie sur $[0;1]$ et donc continue sur cet intervalle. $abs(t)$ parcourt donc toutes les valeurs entre $x_A + 0\overline{x} = x_A$ et $x_A + \overline{x} = x_B$. $[AB]$ passe donc par au moins un pixel de chaque abscisse comprise entre x_A et x_B.

La démonstration s'applique également pour la fonction *ord*. □

Propriété 2 : Soit un segment de coordonnées x et y, de distance en pixel d et comportant p ponts. On a alors l'égalité

$$x + y - p = d.$$

Preuve :

On traitera le cas $x \geqslant y$; la preuve s'effectue de la même manière dans le cas contraire.

Soit n_{pass} le nombre de passages, n_m le nombre de marches et n_{pal} le nombre de paliers. $d = n_{\text{pass}}$ et $n_{\text{pass}} = n_m + n_{pal}$ donc $n_m + n_{\text{pal}} = d$. Or $n_m = y$ et il y a deux types de passages reliant deux pixels d'abscisses différentes, les paliers et les ponts, donc $n_{\text{pal}} + p = x$ soit $n_{\text{pal}} = x - p$.

On retrouve alors bien $x + y - p = d$. □

3.3 Nombre de ponts d'un segment en fonction de x et y

3.3.1 cas ou $x = 0$ ou $y = 0$

Dans ce cas il n'y a pas de marche et donc pas de pont.

3.3.2 cas ou $x \neq 0$ et $y \neq 0$

Soit $(x \ ; \ y) \in \mathbb{N}^{*2}$. On suppose deux points $A(x_A \ ; \ y_A)$ et $B(x_B \ ; \ y_B)$ de coordonnées entières telles que $|x_B - x_A| = x$ et $|y_B - y_A| = y$, définissant le segment $[AB]$ de coordonnées x et y et ayant pour représentation paramétrique :

$$\begin{cases} abs(t) = x_A + \overline{x}t \\ ord(t) = y_A + \overline{y}t \end{cases}, \ t \in [0;1]$$

Pour qu'un point $P(abs(t) \ ; \ ord(t))$ de $[AB]$ soit un pont, il faut et il suffit que $abs(t)$ et $ord(t)$ soient demi-entiers.

x_A et y_A étant entiers, $\overline{x}t$ et $\overline{y}t$ sont donc demi-entiers et il en va donc de même pour xt etyt.

On sait que $0 \leqslant t \leqslant 1 \Leftrightarrow 0 \leqslant xt \leqslant x$ car $x > 0$.

Or entre 0 et x il existe x demi-entiers, soit x valeurs de t possibles pour que xt soit demi-entier, notées t_{xn} avec n allant de 1 à x.

On a alors :

$$t_{xn} = \frac{n - 0,5}{x}, \quad \forall n \in \ [\![1 \ ; \ x \ ]\!]$$

De la même manière, les valeurs de t possibles pour que yt soit demi-entier sont :

$$t_{ym} = \frac{m - 0,5}{y}, \quad \forall m \in [\![1\ ;\ y\ ]\!].$$

Or pour qu'il y ait un pont il faut que les deux coordonnées soient demi-entières. Un pont correspond donc à une solution à l'équation $t_{xn} = t_{ym}$, d'inconnues n et m :

$$(E) : \qquad \frac{n - 0,5}{x} = \frac{m - 0,5}{y}$$

3.4 Détermination du nombre de ponts : résolution de (E)

[1] On cherche désormais à résoudre (E) dans $[\![1\ ;\ x\ ]\!] \times [\![1\ ;\ y\ ]\!]$, avec $(x\ ;\ y) \in \mathbb{N}^{*2}$:

$$\begin{aligned}(E) : \qquad & \frac{n - 0,5}{x} = \frac{m - 0,5}{y} \\ \Leftrightarrow\ & (2m-1)x = (2n-1)y \\ \Leftrightarrow\ & \frac{(2m-1)x}{PGCD(x\ ;\ y)} = \frac{(2n-1)y}{PGCD(x\ ;\ y)} \\ \Leftrightarrow\ & (2m-1)x' = (2n-1)y'\end{aligned}$$

avec $x' = \dfrac{x}{PGCD(x\ ;\ y)}$ et $y' = \dfrac{y}{PGCD(x\ ;\ y)}$; x' et y' sont donc premiers entre eux.

3.4.1 cas où x' et y' sont de parités différentes

$$(E) : \qquad (2m-1)x' = (2n-1)y'$$

$2n-1$ et $2m-1$ sont impairs et parmi x' et y', l'un des deux est pair et l'autre est impair. Il y a donc un des membres de l'équation qui est entier pair, et l'autre qui est impair, il n'y a donc pas de solution dans ce cas, et donc pas de pont. Ce cas survient lorsque la plus grande puissance de 2 divisant x est différente de la plus grande puissance de 2 divisant y, on a alors x' et y' de parités différentes.

3.4.2 cas où x' et y' sont tous les deux impairs

Ce cas survient lorsque la plus grande puissance de 2 divisant x est égale à la plus grande puissance de 2 divisant y : x' et y' ne comportent pas de facteurs 2 dans leur décomposition en facteurs premiers .

Propriété 3 : Lorsque la plus grande puissance de 2 divisant x est égale à la plus grande puissance de 2 divisant y, il y a $PGCD(x\ ;\ y)$ couples solution à (E) et donc autant de ponts .

Preuve :

Soit un couple d'entiers $(n; m)$ solution de (E) : $\qquad (2m-1)x' = (2n-1)y'$.

x' et y' sont premiers entre eux ; x' divise $(2n-1)y'$ et y' divise $(2m-1)x'$ donc d'après le théorème de Gauss, x' divise $2n-1$ et y' divise $2m-1$. On a alors :

$$2n - 1 = kx' \qquad \text{et} \qquad 2m - 1 = k'y'$$

avec k et k' deux entiers naturels impairs. Or :

$$\begin{aligned}(2m-1)x' &= (2n-1)y' \\ \Leftrightarrow k'y'x' &= kx'y' \\ \Leftrightarrow k' &= k\end{aligned}$$

On a donc $2n-1 = kx'$ et $2m-1 = ky' \Leftrightarrow n = \dfrac{kx'+1}{2}$ et $m = \dfrac{ky'+1}{2}$ avec k un entier naturel impair . Or $1 \leqslant n \leqslant x$. On en déduit donc :

$$\begin{aligned}&1 \leqslant n \leqslant x \\ \Leftrightarrow\ &1 \leqslant 2n-1 \leqslant 2PGCD(x\ ;\ y)x' - 1 \\ \Leftrightarrow\ &1 \leqslant kx' < 2PGCD(x\ ;\ y)x' \\ \Leftrightarrow\ &1 \leqslant k < 2PGCD(x\ ;\ y)\end{aligned}$$

(l'inégalité $1 \leqslant m \leqslant y$ conduit au même résultat)

Réciproquement, on va maintenant montrer que pour tout entier naturel impair k inférieur à $2PGCD(x\ ;\ y)$, le couple $\left(\dfrac{kx'+1}{2}\ ;\ \dfrac{ky'+1}{2}\right)$ est solution de (E).

$kx'+1$ et $ky'+1$ sont deux entiers naturels pairs donc $\dfrac{kx'+1}{2}$ et $\dfrac{ky'+1}{2}$ sont deux entiers naturels.
De plus on a déjà montré par équivalence $1 \leqslant n = \dfrac{kx'+1}{2} \leqslant x$ et par le même raisonnement on peut montrer $1 \leqslant m = \dfrac{ky'+1}{2} \leqslant y$. On a donc bien $\left(\dfrac{kx'+1}{2}\ ;\ \dfrac{ky'+1}{2}\right) \in [\![1\ ;\ x\ ]\!] \times [\![1\ ;\ y\ ]\!]$.

Enfin : $\left(2\dfrac{ky'+1}{2} - 1\right)x' = ky'x'$ et $\left(2\dfrac{kx'+1}{2} - 1\right)y' = kx'y'$

Donc pour tout entier naturel impair k inférieur à $2PGCD(x\ ;\ y)$, le couple $\left(\dfrac{kx'+1}{2}\ ;\ \dfrac{ky'+1}{2}\right)$ est solution de (E).
Or il existe $PGCD(x\ ;\ y)$ entiers naturels impairs compris entre 1 et $2PGCD(x\ ;\ y)$: il y a donc $PGCD(x\ ;\ y)$ couples solution à (E) et donc autant de ponts lorsque la plus grande puissance de 2 divisant x est égale à la plus grande puissance de 2 divisant y. □

3.4.3 Propriété de multiplicabilité et divisibilité d'un segment

Propriété 4 : Lorsqu'on multiplie les coordonnées x et y d'un segment par un même nombre rationnel positif $\dfrac{a}{b}$ tel que b divise les coordonnées, alors le nombre de ponts p et la distance d sont multipliés par ce même nombre.

Preuve :

Soient A et B les extrémités d'un segment $[AB]$ de coordonnées x et y , p et d respectivement le nombre de ponts et la distance en pixels de ce segment et un entier b tel que b divise x et y. b divise donc également $PGCD(x; y)$.

On considère un nombre rationnel positif $\frac{a}{b}$. La plus grande puissance de 2 divisant x est égale à la plus grande puissance de 2 divisant y si, et seulement si la plus grande puissance de 2 divisant $\frac{a}{b}x$ est égale à la plus grande puissance de 2 divisant $\frac{a}{b}y$.

Ainsi pour un deuxième segment $[A_1B_1]$ de coordonnées $x_1 = \frac{a}{b}x$ et $y_1 = \frac{a}{b}y$, le calcul de son nombre de ponts p_1 se fait donc de la même manière que le calcul de p.

Selon les cas :

— $p = 0$ donc $p_1 = 0$ on a alors $p_1 = \frac{a}{b}p$

— $p = PGCD(x\ ;\ y)$ donc $p_1 = PGCD(x_1\ ;\ y_1)$ on a alors :

$$p_1 = PGCD\left(\frac{a}{b}x\ ;\ \frac{a}{b}y\right)$$
$$\Leftrightarrow p_1 = \frac{a}{b}PGCD(x\ ;\ y) = \frac{a}{b}p$$

Dans tous les cas on a alors $p_1 = \frac{a}{b}p$.

Or $x + y - p = d$ et $x_1 + y_1 - p_1 = d_1$ donc $d_1 = \frac{a}{b}(x + y - p) = \frac{a}{b}d$.

Multiplier les coordonnées d'un segment par un même nombre rationnel $\frac{a}{b}$ tel que b divise les coordonnées revient donc à multiplier p et d par ce même nombre. □

3.5 Bilan : calcul de la distance suivant les coordonnées d'un segment

Pour déterminer la distance en pixel d d'un segment de coordonnées x et y, on a alors deux méthodes :

1. Une méthode générale :
   (a) Si $x = 0$ ou $y = 0$, alors $p = 0$.
   (b) Sinon on détermine si la plus grande puissance de 2 divisant x est égale à la plus grande puissance de 2 divisant y et on calcule p en conséquence
   (c) On calcule alors d à partir de x, y, et p.
2. Si on connait déjà les coordonnées x_0 et y_0 d'un segment $[A_0B_0]$, ainsi que sa distance en pixels d_0, et que les coordonnées x et y du segment que l'on recherche sont respectivement égales aux produits par un même nombre de x_0 et y_0, alors la distance en pixel recherchée d est égale à d_0 multipliée par ce même nombre.

On utilisera par la suite uniquement la première méthode car elle ne nécessite que la connaissance de x et de y.

4 Application à un programme de tracé de cercles

4.1 Fonctions de calcul de distances

Le calcul décrit précédemment se traduit par l'ensemble de fonctions de la Figure 5.

Les trois premières fonctions sont des sous-fonctions qui sont utilisées dans la fonction `distance` :

— la fonction `facteurs2` renvoie le nombre de facteurs 2 dans son argument : tant que l'on peut le diviser par 2 et que l'on obtient un reste nul, on ajoute 1 à la variable `nombre2` qui compte le nombre de fois où on a divisé par 2, donc le nombre de facteurs 2.

```python
def facteurs2(n): # recherche du nombre de facteur de 2 dans la decomposition en produit de facteur premiers
    Nombre2 = 0
    while n%2==0:
        n = n//2
        Nombre2+=1
    return Nombre2

def pontfinder(x, y):
# verification de la presence de ponts

    return facteurs2(x) == facteurs2(y) # si le nombre de facteur deux sont égales il y a un pont

def pgcd(x, y): # calcul du nombre de pont grace au pgcd
    while y !=0:
        q=x//y
        x,y=y,x-y*q
    return x

def distance(x,y): #calcul de la distance
    if x ==0 or y==0:
        return (x+y)
    if pontfinder(x, y): #calcul de la distance si il y a un pont
        return (x+y-pgcd(x, y))
    else:               # calcul de la distance si il n'y a pas de ponts
        return (x+y)
```

FIGURE 5 – Fonctions de calcul de distance

— la fonction `pontfinder` teste si ses arguments `x` et `y` ont le même nombre de facteurs 2.
— La fonction `pgcd` calcule le plus grand diviseur commun de ses deux arguments, par l'algorithme d'Euclide.

La fonction `distance` applique alors la méthode décrite dans le paragraphe **3.5**.

L'élaboration du programme nous a posé quelques problèmes mais plus particulièrement avec un cas précis : nous n'avions pas au départ traité séparément le cas où `x` ou `y` était égal à 0, et l'appel de la fonction `distance` avec `x` ou `y` égal à 0 faisait planter le programme. Nous avons recherché la cause de ce problème, en nous doutant qu'il s'agissait d'une boucle infinie, et nous avons vu que nous demandons dans la fonction `facteurs2` de diviser le nombre par deux jusqu'a ce que le reste ne soit pas un entier. Or si on divise zéro par deux, le quotient et le reste sont toujours 0 et le programme le répète donc à l'infini ; de ce fait on ajouté la première condition dans la fonction `distance` qui permet d'éviter ce problème.

La deuxième condition fait que si `pontfinder` renvoie `true` c'est à dire que `x` et `y` ont le même nombre de facteurs 2, alors la distance est égale à `x + y` − le plus grand diviseur commun de ces 2 nombres ; et si aucune de ces conditions n'est utilisée la distance est égale à `x + y`.

4.2 Fonctions de dessin

Assez tôt dans notre démarche de recherche, nous avions pour intention de coder un programme en Python qui nous permettrait de réaliser des représentations graphiques de cercles de pixels et d'extraire des données comme le nombre de pixels d'une grille qui correspondent à une même distance.

Nous avons commencé par programmer un algorithme qui pourrait demander à l'utilisateur un nombre représentant une distance et renvoyer une grille dont l'ensemble des carreaux dont la somme de l'abscisse et de l'ordonnée correspondrait à cette distance seraient colorés grâce à une

librairie nommé `BlockGrid` qui propose l'affichage d'une grille modifiable à souhait. Ce premier programme ne prenait pas en compte les ponts mais jetait déjà les fondations d'un futur code qui représenterait correctement des cercles de pixels.

Le programme final (Figure 6) quant à lui utilise la librairie `Matplotlib`, qui est plus performante que `BlockGrid`, et il prend également en compte les ponts : il calcule la distance en pixels de chaque pixel de la grille au centre de la grille en utilisant la fonction `distance` décrite dans la partie 4.1.

```
def Tracecercle(R):

    #Initialisation des variables

    Taille = R*2 +1 #On établit les dimensions de notre grille carrée de pixels
    MilieuX = MilieuY = Taille/2 -0.5 #On établit les coordonnées du centre de la grille grâce aux milieux horizontal et vertical

    ListeX, ListeY = Listing(Taille) #On remplit deux listes de tous les nombres entier compris dans l'intervalle [0; Taille]

    grille=[[[220,220,220] for i in range(Taille)] for i in range(Taille)]

    for x in ListeX:

        for y in ListeY:

            if distance(abs(MilieuX-x), abs(MilieuY-y))==R:

                grille[x][y] = (255,0,0)

    ## Affichage de l'image
    plt.clf()
    plt.imshow(grille)
    plt.axis('equal')
    plt.show()
```

Figure 6 – Programme final de tracé de cercle

Cet algorithme est fondé sur deux boucles `for` insérées l'une dans l'autre pour pouvoir appliquer la fonction `distance` sur chaque pixel de la grille, ligne par ligne, du pixel à l'abscisse et à l'ordonnée les plus élevées vers celui aux coordonnées les plus faibles : si le pixel parcouru est à la distance souhaitée du pixel central, il change de couleur sinon il garde la couleur du fond.

5 Figures obtenues et conjectures

Ce programme une fois réalisé a ainsi permis d'extraire des données et des représentations graphiques en grandes quantités, parmi lesquelles celles de la Figure 7.

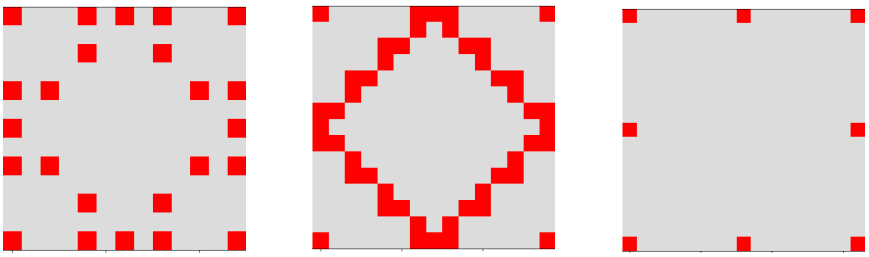

Figure 7 – Cercles de rayons 6, 7 et 8

5.1 Plusieurs types de cercles en pixels

La première étape dans ces observations a été de comparer l'allure de cercles de pixels de différents rayons et de leur trouver des points communs et des différences afin de les classer par catégories. Nous avons établi tous ces types de cercles par leurs points communs au niveau de la décomposition de leur rayon en facteurs premiers et leurs représentations graphiques qui parfois ont une allure similaire et de nombreux pixels en commun bien que le nombre de pixels qui les constituent ne soit pas toujours le même.

Nous avons ainsi pu classer ces cercles en quatre catégories :

— Les cercles dont le rayon est une puissance de 2 (distance de 2 ; 4 ; 8 ; 16 ...) de la FIGURE 8.

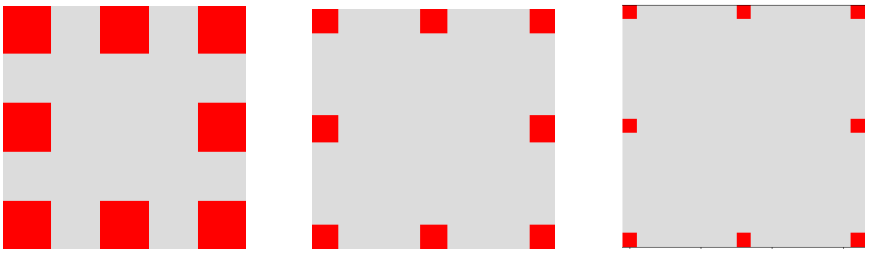

FIGURE 8 – Cercles de rayons 2, 4 et 8 (Puissances de 2)

— Les cercles dont le rayon est pair et non puissance de 2 (6 ; 10 ; 12...) de la FIGURE 9

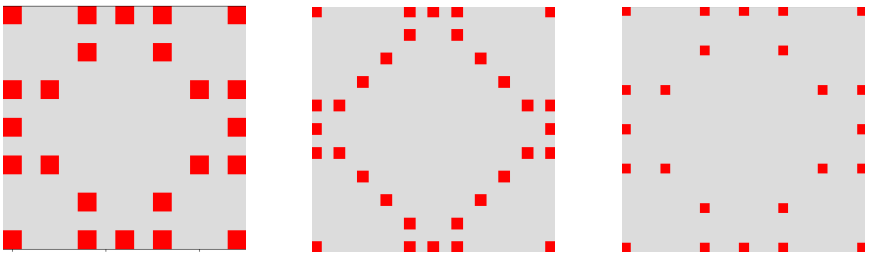

FIGURE 9 – Cercles de rayons 6, 10 et 12 (Nombres pairs non puissances de 2)

— Les cercles dont le rayon est un impair premier (3 ; 5 ; 7...) de la FIGURE 10

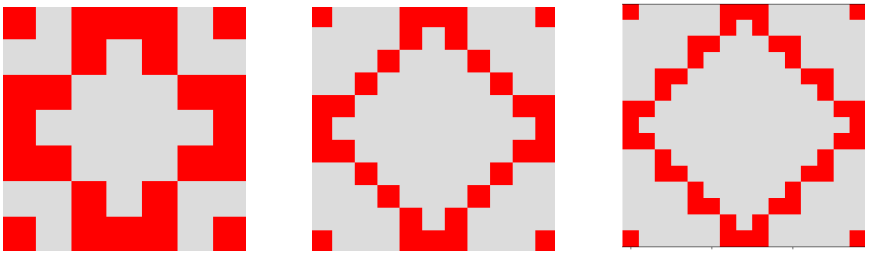

FIGURE 10 – Cercles de rayons 3, 5 et 7 (Nombres impairs premiers)

— Les cercles dont le rayon est un impair composé (9 ; 15 ; 21...) de la FIGURE 11

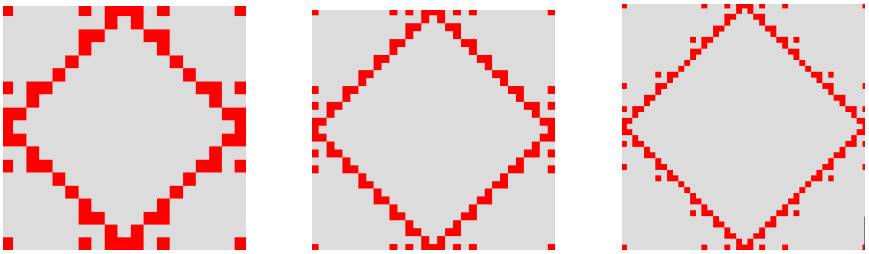

FIGURE 11 – Cercles de rayons 9, 15 et 21 (Nombres impairs composés)

On constate ainsi que toutes les représentations d'une même classe ont une allure semblable. Mais toutes les représentations ont pour point commun de contenir au moins un pixel dans chaque coin de la grille et un pixel au milieu de chaque bord de la grille.

5.2 Des propriétés observables

— Les cercles dont le rayon est une puissance de 2 sont formés de 8 pixels dont 4 sont situés sur les bords de la grille avec par conséquent leur abscisse et leur ordonnée qui sont égaux et un nombre de ponts égal à leur distance et 4 pixels situés au milieu de chaque bord de la grille avec un nombre de ponts nul. Il est important de noter que chaque représentation est une version plus "éclatée" avec des pixels plus espacés par rapport à un cercle de pixel de rayon 1.

— Lorsque l'on prend un cercle de pixel de n'importe quel rayon et que l'on multiplie ce même rayon par n'importe quelle puissance de 2, le cercle obtenu est constitué du même nombre de pixels et possède une allure très similaire mais plus "éclatée". Cela est en corrélation avec les cercles de rayon une puissance de 2 qui sont tous une version « agrandie » du cercle de pixels de rayon 2. Et cela fonctionne avec n'importe quel distance en pixels comme 3 par exemple : ici le premier cercle est de rayon 3, le deuxième de rayon 6 (3×2) et le dernier de rayon 12 (3×2^2). On constate alors que le même schéma semble se répéter jusqu'à l'infini et ce pour n'importe quel premier cercle d'une série tout comme les cercles de rayon égal à une puissance de 2.

— Toutes les représentations de cercles de pixels ont pour point commun de contenir au moins un pixel dans chaque coin de la grille et un pixel au milieu de chaque bord de la grille. La présence systématique de pixels au milieu de chaque bord de la grille s'explique par la taille de la grille qui mesure toujours $2 \times \text{rayon} + 1$ et celle de pixels dans chaque coin/sommet de la grille s'explique par l'égalité entre le rayon du cercle en pixels et le nombre de ponts qui se trouvent entre le centre de ce cercle et les pixels des coins.

— Les cercles de rayon impair se forment d'un carré orienté à 45° ainsi que de pixels isolés au niveau des bords et des coins de la grille, plus ou moins nombreux selon la distance de pixels qui leur correspond, mais si le rayon est un nombre premier, alors il n'y a pas d'autre points isolés que ceux qui sont aux quatre coins.

5.3 Des pistes d'explications ?

Nous n'avons malheureusement pas eu le temps de démontrer les observations et les conjectures émises, voici simplement pour conclure les quelques idées que nous avons à ce sujet :

— Lorsqu'on déplace d'un nombre pair de pixels l'un des sommets d'un segment, son centre se déplacera de la moitié de ce nombre de pixel (donc un nombre entier). Si ses coordonnées étaient entières elles le resteront. Autrement dit les ponts centraux sont conservés par ce ce déplacement d'un sommet. Or tout segment de longueur en distance en pixel pair, possède un milieu qui est confondu avec le centre d'un pixel. On peut donc le partager en deux segments mesurables avec la même distance en pixel. C'est pourquoi on observerait le même nombre de points pour un cercle de rayon x et un cercle de rayon $2x$.

— Chaque segment présentant plusieurs ponts peut se diviser en autant de segments qu'il a de ponts, avec une même distance en pixel. Chacun de ces segments aura un pont.

— La position des points isolés sur le bord pour des rayons impairs composés correspond aux facteurs de la décomposition : le segment qui joint un de ces pixels au centre possède un nombre de ponts égal à un des facteurs ce qui donne des simplifications dans la formule du calcul de la distance ; la distance de ce point isolé au centre est égale à la plus grande de ses coordonnées.

— Nous n'avons pas d'explication pour les points isolés qui ne sont pas sur les bords comme pour ceux du cercle de rayon 21 ...

NOTES DE L'ÉDITION

[1] Peut-être aurait-il été plus simple de supposer dès le départ que x et y sont premiers entre eux, c'est-à-dire : $x = x'$ et $y = y'$, puis d'utiliser la **Propriété 4** pour le cas général.

Avec le soutien de la Région Bretagne et de Brest Métropole