Info112

Rappels de mathématiques

Pour l'informatique graphique à deux dimensions

Orsay - 4 novembre 2024

Trigonométrie (cos, sin, atan)

Les fonctions *cos* et *sin* permettent de calculer les coordonnées cartésiennes(x,y) à partir des coordonnées polaires(r, alpha) grâce à la formule générale :

x = x0 + r***cos*(alpha)

y = y0 + r***sin*(alpha)

La fonction atan(y/x) et « Pythagore » r=sqrt(x*x+y*y) ou r=dist(x,y,0,0). permettent de calculer les coordonnées polaires à partir des coordonnées cartésiennes mais on préférera utiliser **alpha=atan2(y,x)** qui évite de s'occuper des cas où x=0. Par ailleurs Pythagore sert aussi à calculer la distance entre deux points sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)). On peut donc s'en servir pour calculer la distance au centre d'un repère polaire qu'on a choisi. De même on peut utiliser atan2((y1-y2), (x1-x2)) pour calculer l'angle par rapport à un point/centre x2,y2 quelconque (enfin l'angle par rapport à l'horizontale qui passe par ce point)

Changement de repère

En remplaçant dans un calcul ou une formule xa par xb=xa-dx et ya par yb=ya-dy on peut dire que xb,yb sont les coordonnées du même point que xa,ya mais dans un autre repère.

ATTENTION : cos, sin et atan2 manipulent des radians. Une règle de trois permet d'afficher des degrés si nécessaire … Processing a même des fonction radians() et degrees() pour convertir plus facilement.

ATTENTION : La plupart des langages de programmation ont une variable constante PI.

CONSEIL : Faites toujours un schéma

CONSEIL 2 (pour les plus forts) : On a le droit de faire la moyenne de 2 points (pour obtenir le milieu) mais pas la moyenne de 2 angles car ça ne marche que dans certains cas. Pour la faire la moyenne de 2 angles, on fait la moyenne des sinus, la moyenne des cosinus et avec l'arc-tangente des deux on retrouve un angle !

Règle de trois / proportionnalité

Lorsqu'on a deux éléments qui varient proportionnellement l'un à l'autre et qu'on connait un point de correspondance entre les 2 éléments.

Par exemple

H1 : le prix des pommes varie en fonction du poids **proportionnellement**.

H2 : 3 kilos coutent 7 euros (1 correspondance)

Alors on peut calculer la valeur d'un éléments si on connait l'autre avec la règle de trois.

Par exemple :

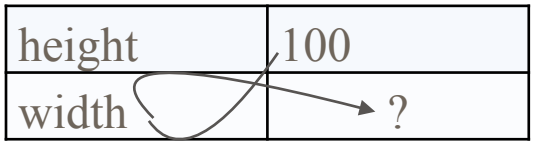
« Combien coutent 5 kilos de pomme ? => 5*7/3

« Combien pèsent 25 euros de pomme ? => 25*3/7

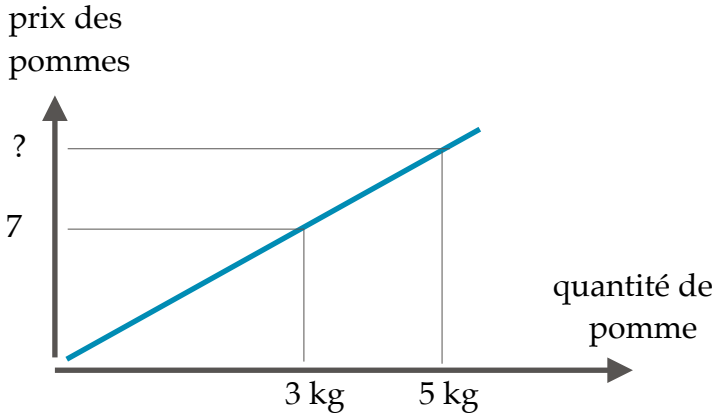
« Combien coutent x kilos de pomme ? => x*7/3

« Combien pèsent y kilos de pomme ? => y*3/7

? = 100*width/height



En géométrie la règle de trois s'appelle aussi Thalès



Règle de trois sur le graphique d'une fonction linéaire

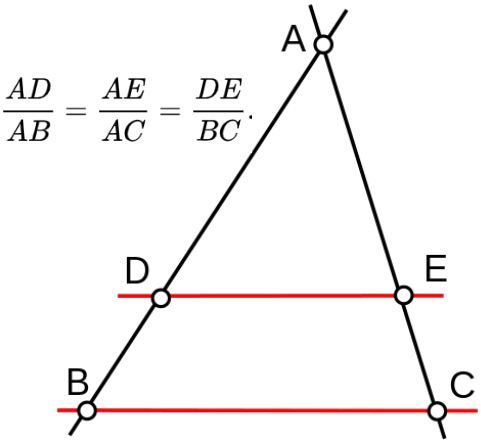


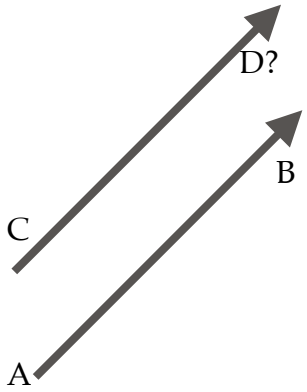
Illustration du Théorème de Thalès sur Wikipedia

ATTENTION : En informatique on ne calculera par forcement le résultat. On peut très bien écrire dans un code **int prix = 7*5/3;** et on laisse l'ordinateur faire le calcul !

ATTENTION 2 : En informatique quand tous les composants d'un calcul sont des **entiers** le langage de programmation peut choisir de faire une division entière (sans virgule). 7*5/3 ≠ 7/3*5 car 7/3*5 calcule d'abord 7/3 et donne comme résultat 2 et 2*5 =10 alors que 7*5/3 calcule d'abord 7*35 = 35 et 35/3 =11.

On préférera donc sans doute écrire **float prix = 7*5.0/3**

Points et Vecteurs



La plupart des fonctions de dessin attendent des coordonnées de **POINT** en paramètre. Pour calculer les coordonnées des points on peut être amené à calculer des coordonnées de **VECTEUR** mais on ne peut pas donner des coordonnées de vecteur à une fonction qui attend des coordonnées de point. Lorsqu'on réfléchit au point qu'on veut donner en paramètre d'une fonction il peut s'agir d'un point qui est au bout d'un vecteur mais ce vecteur part toujours d'un autre point.

Si on connait les coordonnées des points A et B, le vecteur AB a pour coordonnées (xB-xA, yB-yA). Si on cherche les coordonnées du point D au bout du vecteur AB mais qui part du point C on calcule deux additions : xD = xC+(xB-xA) et yD = yC+(yB-yA). On dit qu'on ajoute un point ET un vecteur. Quand on ajoute un point et un vecteur on obtient toujours un autre point. Pour obtenir un vecteur à partir de deux points on fait deux soustractions (point d'arrivée MOINS point de départ).

On peut multiplier les deux coordonnées d'un vecteur par un même scalaire (*0.5 pour le diminuer de moitié, *3 pour le rendre 3 fois plus long) MAIS on ne multiple généralement pas les coordonnées d'un point. On peut diviser les coordonnées d'un vecteur par sa norme(longueur) « sqrt(dx*dx+dy*dy) » avant de le multiplier par un autre scalaire (*18.5) pour obtenir un vecteur dont la longueur est exactement de 18,5.

Si on connait les coordonnées d'un vecteur V(dx, dy) on peut très facilement obtenir deux vecteurs orthogonaux en INVERSANT les x et les y et en rajoutant un signe moins à l'une des coordonnées. On trouve facilement que les coordonnées des vecteurs orthogonaux sont (dy, -dx) et (-dy, dx).

ATTENTION : Pour savoir lequel des vecteur orthogonaux « tourne vers la droite » on peut faire un schéma avec un vecteur V où dx et dy sont positifs et dessiner les vecteur orthogonaux MAIS en informatique graphique, l'axe des y est presque toujours orienté vers le bas ce qui inverse tout !

La somme de deux vecteurs (les sommes des coordonnées de deux vecteurs) est un vecteur

CONSEIL : Faites un schéma pour bien comprendre quel point doit être ajouter à quel(S) vecteur(S) pour obtenir le point désiré.

CONSEIL : Ecrivez consciencieusement la formule pour l'axe des x, faites un copier-coller et dans la formule collée remplacez tous les x par des y et les y par des x.

Principe des complémentaires

Le complémentaire d'une valeur est la valeur maximale que peut prendre cette valeur MOINS la valeur.

Par exemple une coordonnées x à l'écran a pour complémentaire « width-x » car width est la largeur de l'écran ou de la fenêtre.

Par exemple une valeur de vert g a pour complémentaire « 255-g » car 255 est la quantité de vert maximal d'un pixel.

Par exemple un angle an a pour complémentaire « 2*PI-an » car 2*PI est l'angle maximal qui fait un tour complet.

Par exemple une note no a pour complémentaire « 20-no » car 20 est la note maximale que vous pouvez obtenir.

Un complémentaire est une valeur très similaire à l'original et on peut souvent lui appliquer les mêmes mathématiques (addition, soustraction, multiplication, racine carrée, etc.) mais avec un effet « inversé »

ATTENTION : une fois le complémentaire « modifié » par ces mathématique il faut TRES SOUVENT en reprendre le complémentaire pour utiliser le résultat de la même façon que la valeur initiale.

Par exemple, le complémentaire d'une note est le nombre de points qu'il manque pour obtenir 20/20. Ue note de 12/20 a pour complémentaire 8. Diviser une note par deux est une sanction très lourde (12/20 => 6/20). Diviser le complémentaire par deux est une récompense énorme (12/20=manque 8 points devient manque 4 points=16/20) mais il faut reprendre le complémentaire (20-4) pour calculer à quelle note correspond cette récompense (16/20). La formule pour une note quelconque est donc 20 - ((20-no)/2) où vous reconnaitrez deux fois le symbole 20- car il y a bien deux fois la formule du complémentaire.

CONSEIL : En informatique toute variable a un maximum mais il faut réfléchir pour savoir si le complémentaire signifie vraiment quelque chose.

CONSEIL : On est pas forcé de faire le complémentaire au maximum. Comme le complémentaire a un effet miroir on peut vouloir faire le complémentaire à width/2 pour que l'effet miroir soit autour de l'axe vertical width/4

Moyennes Pondérées

La moyenne pondérée de plusieurs valeurs est le calcul de la somme des valeur, chacune multipliée par son poids respectif, le tout divisé par la somme des poids.

On fait généralement la moyenne pondérée de plusieurs valeurs ayant la même signification (moyennes de plusieurs longueurs, moyenne de plusieurs luminosités, etc.). On obtient une valeur intermédiaire entre les valeurs utilisées avec toujours la même signification (une moyenne de longueurs est une longueur, …).

Dans un algorithme, les poids peuvent être constants mais les valeurs peuvent être des variables. Par exemple les poids peuvent permettre une moyenne pondérée entre la position de la souris et le centre de l'écran où le centre de l'écran a trois plus de poids : (mouseX*1 + width/2*3) /4. Si on fait une moyenne pondérée sur les x on a surement envie de faire la même sur les y. Dans d'autres algorithmes les valeurs peuvent être constantes mais les poids variables. Pour une valeur de t (du temps) entre 0 et 10 on pourra afficher quelque chose en
(posX2 * t + posX1* (10-t)) /10. Cette moyenne pondérée entre posX1 et posX2 offrira un grand nombre de valeurs intermédiaires entre posX1 et posX2 au cours du temps.
Si on fait une moyenne pondérée avec les poids i et (10-i) on trouvera 11 positions intermédiaires entre la position de la souris et le centre de l'écran.

```
for (int i=0; i<=10; i++){
   x = (mouseX*i + width/2*(10-i)) /10
   y = (mouseY*i + height/2*(10-i)) /10
}
```

CONSEIL : Il est même possible de faire une moyenne pondérée avec des poids négatifs … tant que la somme des poids ne fait pas 0 bien sûr.

CONSEIL : Si on a que 2 valeurs et qu'on prend comme poids une variable et son complémentaire, la somme des poids est facile et constante.

CONSEIL : On dit qu'on normalise une moyenne pondérée lorsqu'on choisit les poids pour que leur somme fassent 1 (ce qui évite la division finale).

CONSEIL : On ne peut pas faire la moyenne d'angles. On est obligé de passer en « coordonnées cartésiennes » et prendre l'atan2(somme des sinus, somme des cosinus)

Rappel Processing

```
void setup() {
  // executé une seule fois
}
void draw() {
  // executé 20 fois par seconde
}
```

On peut accéder au pixel i,j d'une image img avec la formule

`img.pixels[i+j*img.width]`

On peut accéder à TOUS les pixels d'une image avec le code générique

```
img.loadPixels();
for (int j=0; j<=img.height; j++){
  for (int i=0; i<=img.width; i++){
    int r = red(img.pixels[i+j*img.width]); …
    img.pixels[i+j*img.width] = color(…);
  }
}
img.updatePixels();
```

On peut changer la couleur de dessin stroke(…) et de remplissage fill(…) en donnant 3 valeurs : rg et b ou hab selon colorMode(RGB) ou colorMode(HSB).

ellipse(x,y, w,h); dessine une ellipse en x,y de taille w h (un cercle si w=h)
rect(x,y, w,h); dessine une rectangle en x,y de taille w h (un carré si w=h)
line(x1,y1, x2,y2); dessine une ligne entre les points x1,y1 et x2,y2

random (min, max) retourne un/des nombres pseudo-aléatoires entre min et max toujours les mêmes après randomSeed(x)
noise(x), noise(x, y), noise(x, y, z) retourne une valeur entre 0 et 1 (plutôt proche de 0,5) qui est continue / proche des valeurs de noise(x+dx), noise(x+dx, y+dy), noise(x+dx, y+dy, z+dz) quand dx, dy, dz sont petits (bien inférieurs à 1).

frameCount est un entier qui dit combien de fois on a re-dessiné la fenêtre depuis le lancement
mouseX, mouseY valent la dernière position connue de la souris
frameRate(r) change le nombre d'affichage par seconde
mouseMoved(), mouseDragged(), mousePressed(), mouseReleased() sont des foncions callback qu'il suffit de remplir et qui sont appellees automatiquement selon l'utilisation de la souris.

pushMatrix() et restoreMatrix() sauvent et resteront le repère de dessin. Entre les deux on peux faire des changement de repère avec translate(dx, dy),rotate(ang_radian) ou scale(zooms, zoomy)

bezier (x1, y1, x2, y2, x3, y3, x4, y4) dessine une courbe de bezier de x1, y1 vers x4, y4 avec les points de contrôles qui font que la courbe est tangente en x1,y1 vers x2, y2