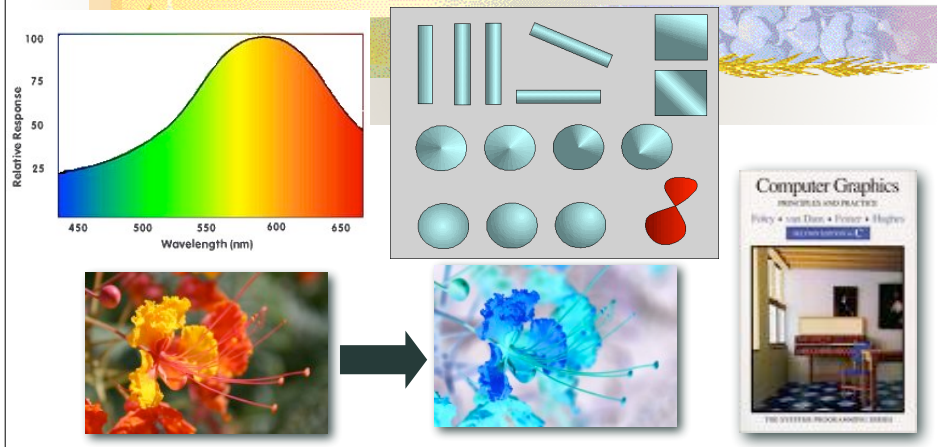
Introduction au Graphisme des ORdinateur (**Info112** ou I.G.OR.)

Frédéric VERNIER



Introduction au Graphisme des ORdinateur (**Info112** ou I.G.OR.)

Frédéric VERNIER

Graphisme en 2 dimensions : Art of no-Z

Sources :

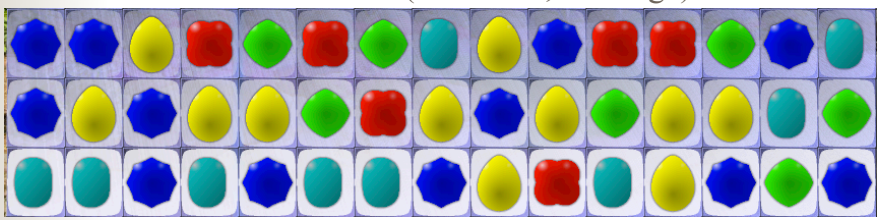
http://wikipedia.org/
http://student.kuleuven.be/~m0216922/CG
http://www.gnurou.org/writing/linuxmag/sdl/
http://sol.gfxile.net/gp/
http://developer.apple.com
Livre Foley & Van Dam

Ressources

- http://www.cs.dartmouth.edu/~fabio/teaching/programming10/notes/1.php
- https://www.lri.fr/~mbl/ENS/IG1/plan.html
- http://lodev.org/cgtutor/
- http://trolen.polytech.unice.fr/cours/maje/cours/
- **https://processing.org/reference/**
- **https://ecampus.paris-saclay.fr/course/view.php?id=34872**

Programme

- Graphisme matriciel : Les pixels et la transparence
- Graphisme vectoriel : Les primitives de dessin
- Double buffering, manipulation d'images et d'écrans
- Hasard : le sel et le poivre du graphisme
- Temps (évolution des pixels) : Animations
- Couplage graphisme-utilisateur : Interaction
- Effets spéciaux (ombres, dégradés, inverse-video, etc.)
- Fausse 3D : le 2D et demi (occlusion, ombrage)



Un bref retour sur l'histoire

- Les ordinateurs ont besoins …
  - De programmes
  - De données
  - D'un utilisateur (ne serais-ce pour dire que programme faire tourner sur quelles données)
- Comment faire communiquer humain et ordinateur ?
  - Des boutons (on/off), des manettes, des poignées, des bouts de papier, des oscilloscopes ?
- Quel est le problème en fait ?
  - Ordinateur rapide, infaillible, borné, sans initiative
  - Humain lent, faillible, infini, imaginatif
  - Langages très très différents

Communication

- Entités internes
  - Humain : phrases, langage dit naturel, idées, valeurs comparées sur une échelle, ordres de grandeur, préférences, envies, etc.
  - Ordinateur :
    - 0 et 1
    - Chiffres (voir comment coder un chiffre avec des 0 et 1)
    - Lettres (voir comment coder des lettres avec des chiffres)
    - Instructions simples (addition, test d'égalité, redémarrer le programme a partir de la ligne 43, etc.)
- Organes de communication
  - Humain : ouie, vue, odorat, goût et toucher
  - Ordi : aucun par défaut mais tout ce qui peut se transformer en électricité est utilisable pour faire un sens de communication !
    - ATTENTION: l'électricité doit être quantifiée (4v ⇒ 01101)

Problème

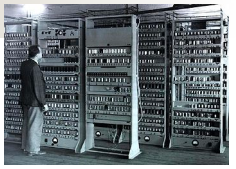
- Les mots ne sont qu'une succession de lettres pour un ordinateur
- Les lettres manuscrites ne sont que des gribouillis
- Aucun sens humain ne perçoit l'électricité ni n'en produit (… sans chirurgie ;-)
- L'ordinateur n'a pas d'idées à lui, pas d'initiative

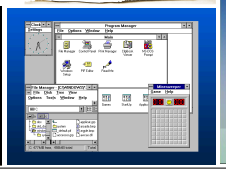
On a fait des programmes qui simulent des neurones mais aucune idée n'en est sorti !

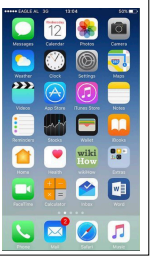
Les ordinateurs sont déterministes par NATURE

Evolution des paradigmes

- Le mode carte perforée - imprimante
  - Papier cartonné **imprimé** et perforé
- Le mode console
  - Un clavier = boutons poussoirs avec dessin dessus
  - Un écran = un oscilloscope ++
- Le mode graphique « fenêtré »
- Le mode touch full screen

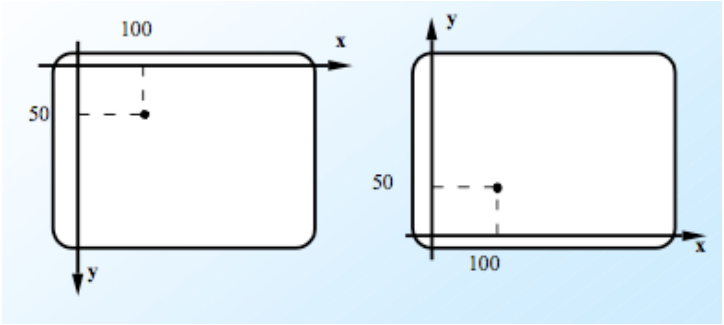






Références à l'écran

- Certains systèmes graphiques utilisent la convention des repères direct en mathématique



…. D'autres pas (la majorité même)

- Héritage des balayages par oscilloscope

Rappel de Maths/Géométrie

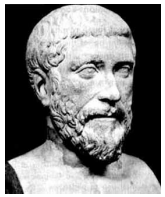
- Point(x,y),Vecteur (dx, dy), droites (ax+by+c=0)
- Trigonométrie (cos, sin, atan, etc.)

Maths

Coordonnées normalisées
coord. polaires
Equations implicites
3x+2y-1=0

Geom. points
$\overrightarrow{AB}$ *= (3,2)*

Matrices



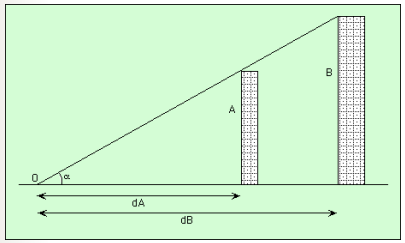
Info

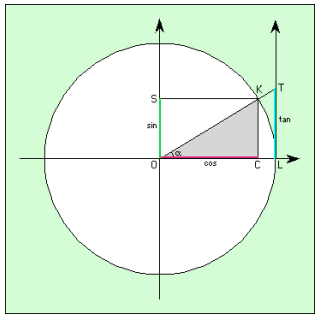
Coordonnées réels de l'informatique
320+cos(x/180*PI), 20+sin(x/180*PI)
Fonctions discrètes

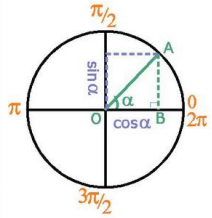
```
float f(float x){
  return (1-3x)/2;
}
```

2 variables
ABx=3;
ABy=2;
Tableaux de nombres

Rappels (Attention intero QCM)



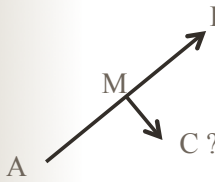




Masse en kg	Prix en €
2	10
1,5	x

Coordonnées : points et vecteurs

- Coordonnées du point C(xC, yC) ?



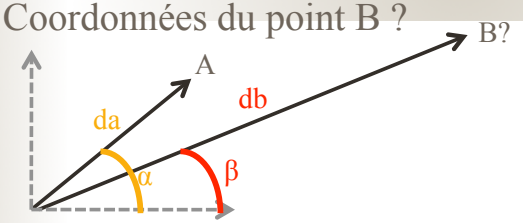
Réponse

A(xa,ya)
En info A n'existe PAS. Il est modélisé par 2 variables xa et ya
B(xb,yb)
$\overrightarrow{AB}$ = (xb-xa, yb-ya) $\overrightarrow{AB}^{\perp}$ = (yb-ya, xa-xb)
$B = A + \overrightarrow{AB}$ => $M=A+\overrightarrow{AB}/2$ => $C=A+\overrightarrow{AB}/2 + \overrightarrow{AB}^{\perp}/4$

xC= xa+ (xb-xa)/2 + (yb-ya)/4 **yC**=ya+(yb-ya)/2+(xa-xb)/4

Coordonnées : points et vecteurs

- Coordonnées du point B ?



Réponse

A(xa,ya)

da=$\sqrt{xa\times xa+ya\times ya}$ α=atan(ya/xa) … ou atan2(ya, xa)

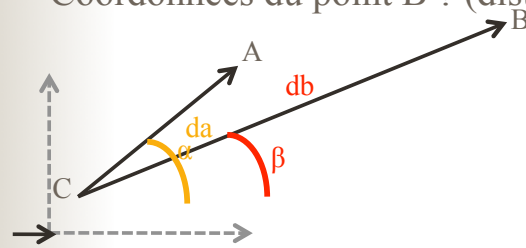
β=α/2

db = da*2

xb= cos(β)*db **yb**= sin(β)*db

Changement de repère, décalage

- Coordonnées du point B ? (dist × 2, angle/2)



Réponse

CA(xa-xc, ya-yc)

α=atan2(ya-yc, xa-xc)

dca=$\sqrt{(xa-xc)\times(xa-xc)+(ya-yc)\times(ya-yc)}$

β=α/2

dcb = dcs × 2

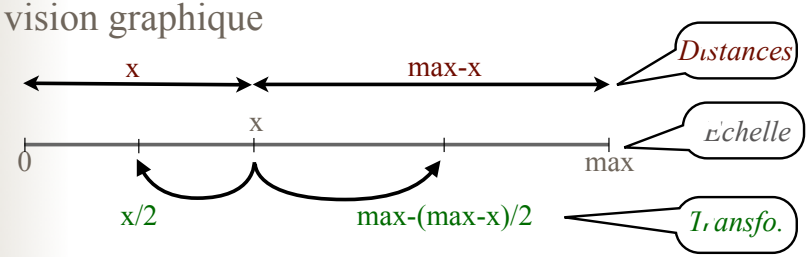
xb= xc+cos(β)*db **yb**=yc+ sin(β)*db

Principe des complémentaires

- Un étudiant à 12/20 s'estime 2 fois meilleur qu'un à 6/20
  - 12/2 = 6
- Un enseignant souhaite "booster" les notes car la moyenne n'est que de 5/20.
  - Il ne PEUT PAS multiplier les notes par deux (12=>24) !
  - 20/20 doit rester 20/20
  - Solution : La moyenne des écarts à 20 ? ... 15
  - 15=>10 .... x2/3
  - 20 - (20-note)*2/3
  - 0/20 => 6.66
- 12/20 = l'écart à 0 est de 12 => Math sur le nombre
  - Moy, coeff mult, etc. => Ne pas oublier de re-normaliser !

Principe des complémentaires

- 0 n'est pas le seul point de référence
- En informatique toute variable a un maximum
  - pas d'infini
- x = (max-(max-x)*1)
- vision graphique



Règle de trois

- Une caisse de 5 kg de cerises coûte 30 €. Combien coûtent 3 kg?

5kg	3 kg
€30	?

- Ma voiture consomme 9 litres d'essence aux 100km. Combien consomme-t-elle pour parcourir 425 km?
- Un pixel passe du noir (0) au gris foncé (64) en 100ms. Quand atteint-il le blanc (255) ?

truc	truc
5kg	3kg
€30	?

Résultat bon truc (bonne unité) = (bon truc (mauvaise unité) / autre truc (mauvaise unité)) x autre truc (bonne unité)

Règle de trois + décalage

- prime agriculture d'altitude
  - 0m => 0euros
  - 999m => 0 euros
  - 1000m => 50 euros
  - 2000m => 300 euros

1000	1400	2000
€50	?	€300

- Combien de prime à 1400m ?
  - 50+400*x (400=**1400-1000**)
  - combien rapporte 1 m ?
    - x=250/1000 (250=**300-50**)

2000-1000	1400-1000
250	?-50

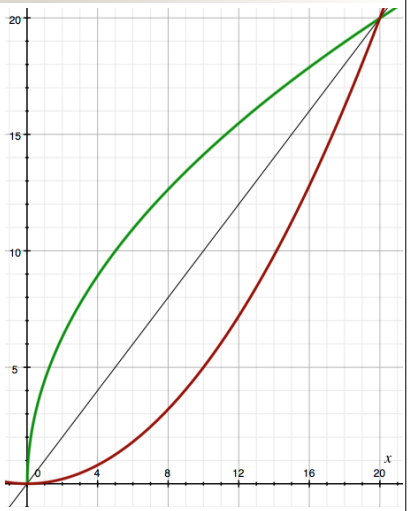
- dans la moitié basse de l'écran faire un dégradé du orange(255, 128, 0) au jaune (255, 0, 0)

Moyenne pondérée

- au bac
  - 12/20 en math coeff 3
  - 7/20 en physique coeff 2
  - moyenne = (12*3+7*2)/(3+2)
- Formule
  - (val1 . coeff1+val2 . coeff2)(coeff1+coeff2)
- Normalisée (0<=coeff<=1)
  - v1*coeff+v2*(1-coeff) / 1
- Variable
  - for (int i=0; i<=N; i++)
    - x = (val1*i + val2*(N-i))/N;

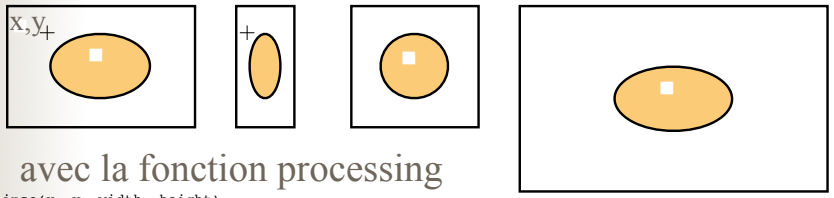
Exercice

- Un prof sympa applique une formule $y=20\cdot\frac{x}{20}^{0.5}$
- Un prof sadique applique une formule $y=20\cdot\frac{x}{20}^{2}$
- Trouvez une formule mixte
  - qui est sadique entre 0 et 10
  - qui est sympa entre 10 et 20
  - qui est de + en + sympa
  - 0 reste 0 et 20 reste 20 !



Exercice

- dessinez un ovale centré de hauteur 100 pixels
  - proportionnel à la fenêtre width x height



- avec la fonction processing

```
ellipse(x, y, width, height)
```

- Solution

```
void draw() {
  ellipse(width/2-?, height/2-?, ?, 100);
}
```

```
void draw() {
  ellipse(width/2-?, height/2-100/2, ?, 100);
}
```

height	100
width	?

```
void draw() {
  ellipse(width/2-?, height/2-100/2, 100*width/height, 100);
}
```

```
void draw() {
  ellipse(width/2-100*width/height/2, height/2-100/2, 100*width/height, 100);
}
```


Programme de révision

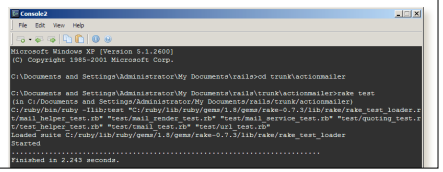
- Trigonométrie (cos, sin, atan)
- Changement de repère
- Règle de trois / proportionnalité
- Thalès / Pythagore
- Points et Vecteurs
- Principe des complémentaires
- Moyennes Pondérées

7

- Pas de par coeur ! Il faut comprendre l'utilité de chaque outil pour l'utiliser au bon moment !

Le mode console

- Texte de la console
  - Si les points lumineux (pixels)sont judicieusement placés, un œil humain interprète la forme comme un caractère. Par exemple « c »
  - Les touches (boutons poussoirs imprimés) du clavier font apparaîtrent le caractère correspondant à l'écran, c'est le principe du retour d'information (feedback)
  - Le feedback apparaît immédiatement à l'écran au niveau d'un caractère spécial clignotant appellé **curseur**
  - Caractères spéciaux
    - Tab, CR, &^%#$... EOF



Exemple

```
Bienvenu dans le programme de jeu d'Echec
  A B C D E F G H
1 T C F r R F C T
2 p p p p p p p p
3 _ _ _ _ _ _ _ _
4 _ _ _ _ _ _ _ _
5 _ _ _ _ _ _ _ _
6 _ _ _ _ _ _ _ _
7 p p p p p p p p
8 T C F r R F C T
Jouez votre coup sur l'échiquier (ex: G7-G5)
```

A retenir

- L'affichage de la touche pressée est le plus **rapide** possible
  - L'humain a du mal à se rappeler
  - L'humain fait des erreurs de frappe qu'il doit corriger
- Le curseur **clignote**
  - Le regard de l'humain est attiré, la recherche est moins longue
  - Le regard de l'humain passe du clavier à l'écran (surtout quand il apprend à se servir du clavier)
  - L'humain s'interrompt parfois dans son travail avant de revenir à ce qu'il faisait
- L'écran défile (vers le bas ligne par ligne) ou est remplacé par un écran quasi identique à chaque rafraîchissement
  - L'humain a du mal à assimiler trop de nouvelles informations
  - Des informations qui apparaissent légèrement décalées par rapport à l'instant d'avant ne sont pas interprétées comme nouvelles !

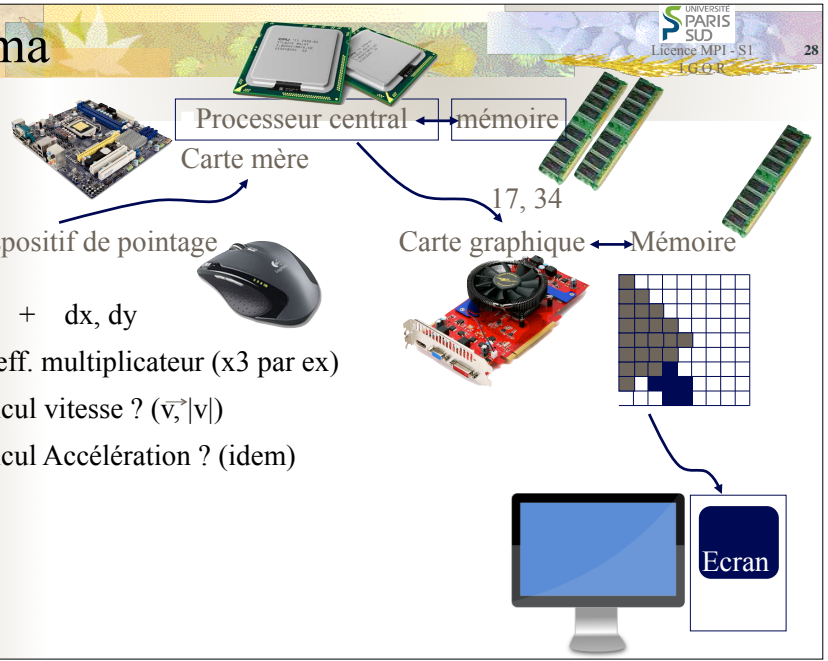
Et maintenant

- Les écrans ont évolués
  - Ils peuvent afficher des dessins géométriques
  - Ils peuvent afficher des images pré-enregistrées (bitmap)
  - Ils peuvent afficher des pixels de couleur
    - En jouant sur trois petits points rouge, vert et bleu très proches
  - Ils sont très rapides, plus grand, etc.
  - Ils peuvent afficher en temps réel le pointeur d'un dispositif de pointage (souris, touchpad ou autre)
    - Ce dernier ne clignote pas … pourquoi ?
- En fait …
  - Matrice de pixels, traitements des graphismes et couplage avec d'autres dispositifs sont dissociés

Exemple

- Comment fonctionne une souris ?
  - schéma général
    - passage de l'électricité
    - Principales fonctions de l'ordinateur
  - Pseudo-code
    - Algorithme
      - Variables
      - Traitements
    - Quand et par qui est appelé ce code
      - 1 fois / n fois / jusqu'à ce que ...
      - à la demande ou n fois par seconde ?
      - ...

Schéma



TD No 1

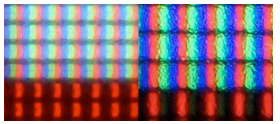
- Explication mémoire / registres (a, b, c, d) et ←

```
a ← souris.dx // en mm
b ← souris.dy
c ← memoire.x
d ← memoire.y
a ← a+c
b ←b+d
memoire.x ←a
memoire.y ←b
…dessine_pointeur_souris()
```

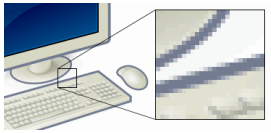
120 fois par seconde

Pixels

- 3 lumières (rouge, verte et bleue) couvrant un carré
  - couvrant un carré (ou presque)
  - en Matrice les uns a coté des autres
  - intensité des 3 lumières contrôlable



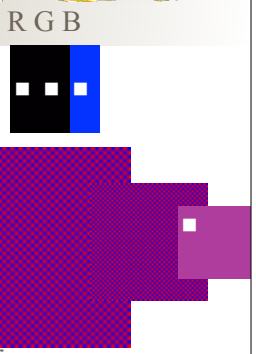
- Suffisamment petites
  - pour ne pas voir leur forme carrée
  - pour ne pas distinguer les lumières



- Attention
  - les lumières ne sont pas de l'encre
  - abrégé p ou px

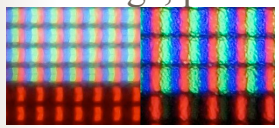
Matrice de pixels

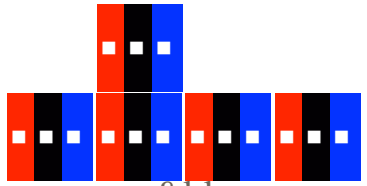
- noir+noir+bleu = bleu foncée?
  - Mélange des couleurs
  - Aussi vrai pour pixels adjacents
- Rouge+Vert+Bleu = Blanc !
- Technologies
  - LCD=LEDs+masque
  - CRT= faisceau d'électron
- Conséquences
  - LCD=meilleur contraste local, noir moins noir
  - CRT= meilleur noir, plus lumineux… mais moins bon contraste
- Contraste (papier + encre noir=10000:1)
  - Ex 400:1 différence de luminosité entre plus sombre et plus clair
  - Si blanc 2x plus blanc = 800:1 mais est-ce les blancs le pb ?



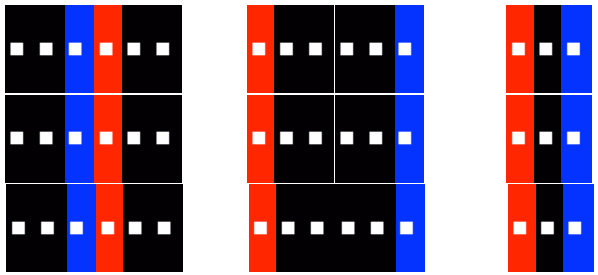
Couleur des pixels

- Pixel rouge, pixel blanc, pixel magenta

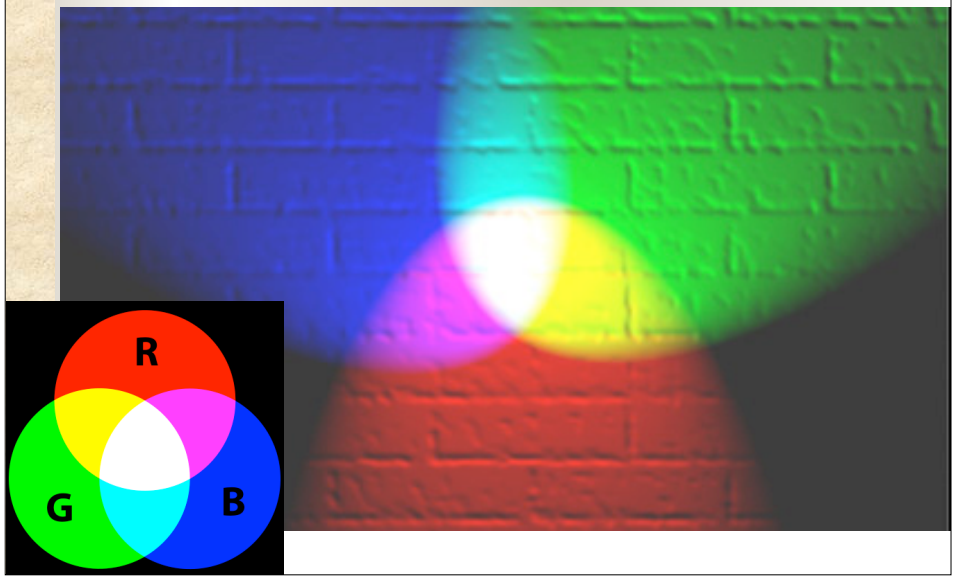




- Horizontal magenta :
- Verticales: bleu&rouge ou rouge&bleu ou magenta



Couleurs additives RGB



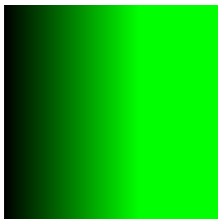
Exemple

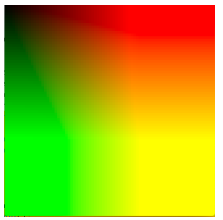
- Pixel par pixel

```
void draw() {
  background(0, 0, 128);
  loadPixels();
  for (int j = 0; j < height; j++) {
    for (int i = 0; i < width; i++) {
      pixels[j*width+i] = color(0, i, 0);
    }
  }
  updatePixels();
}
```

```
void setup() {
   size(400, 400);
}
```

```
R de 3
Bilinéaire
Compl.
Moy pond
/ entière
Pal coul
```

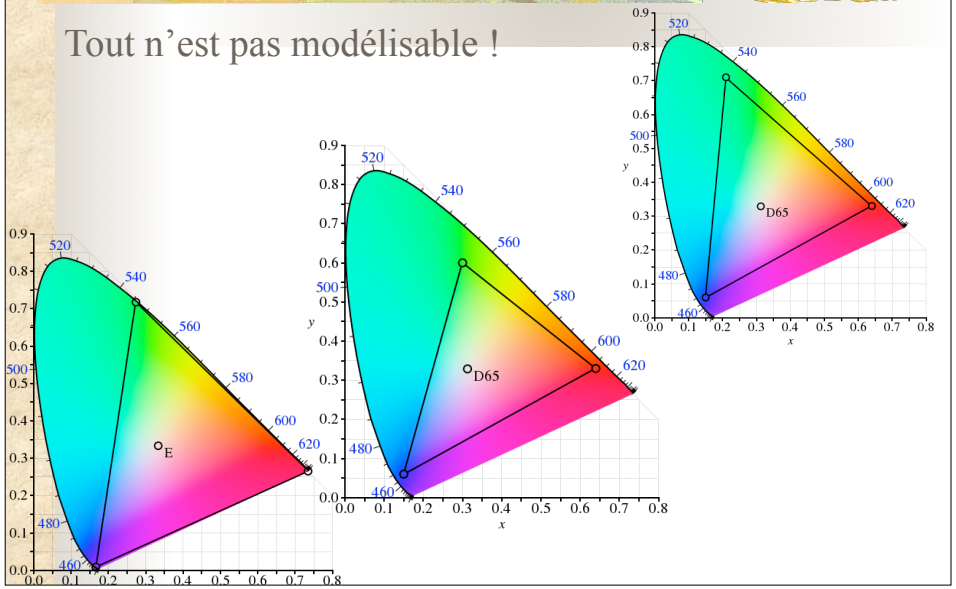




- Vous ne comprenez rien à ce code ?
  - C'est normal !!!
- Essayons-le ensemble
  - posons nous les bonnes questions
- Tout ne sera pas révélé aujourd'hui !

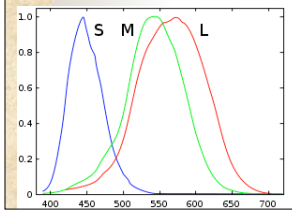
Modèle RGB

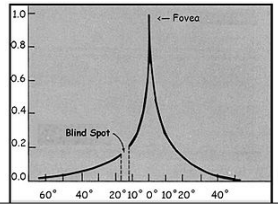
Tout n'est pas modélisable !

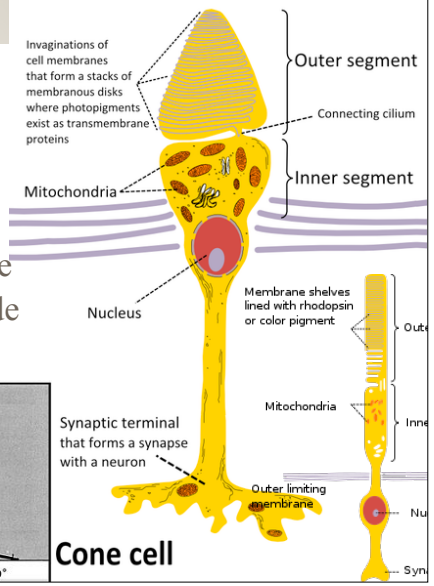


Œil humain

- Bâtonnets
  - 90 M
  - Sensibles (vue obs.)
- Cônes
  - 4.5 M
  - Moins sensible à la lumière
  - Sensible changement rapide
  - 3 types (R, V, et B)







Graphique à bas niveau

- Pourquoi ne pas parler de graphique à haut niveau
  - Peindre la Joconde sur une forme de théière vue de dessus…
  - Mieux comprendre = mieux utiliser
  - Jeux et Applications exigeantes sont construites sur les niveaux bas
  - Sens historique
  - Il reste des choses nouvelles et prometteuses à faire
- Instructions et Données de base
  - Instruction non décomposable
  - Données sous la forme la plus proche de sa forme matérielle

Palette de couleurs

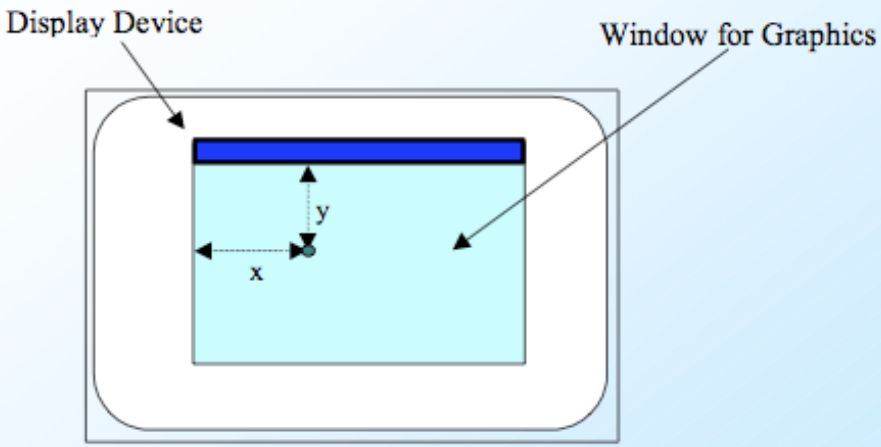
256 couleurs de la palette VGA

Clairs / foncés

Tons

Couleurs vives et pastels

Si dessin à l'écran ou dans une fenêtre

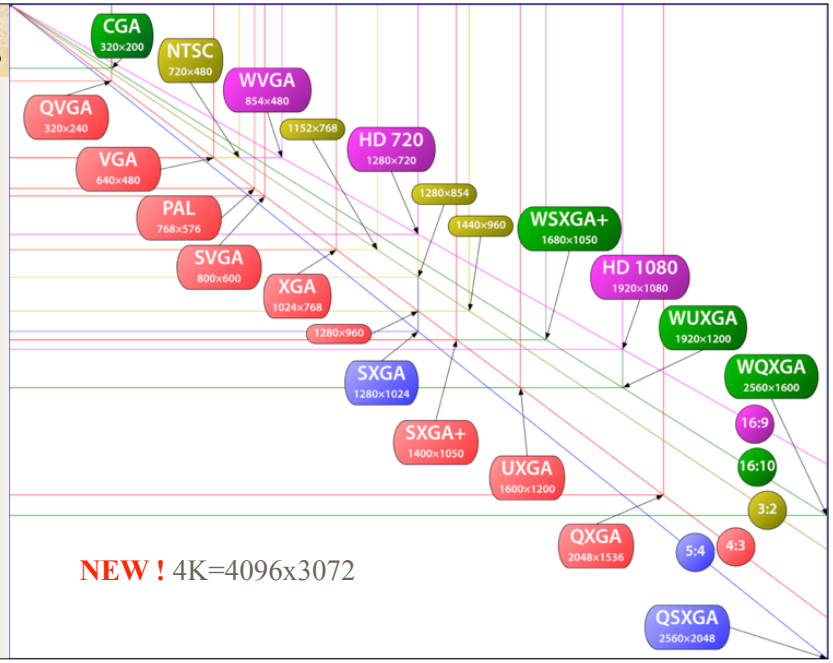


Normal meaning for *SetPixel(x,y,green)*

Taille de l'écran

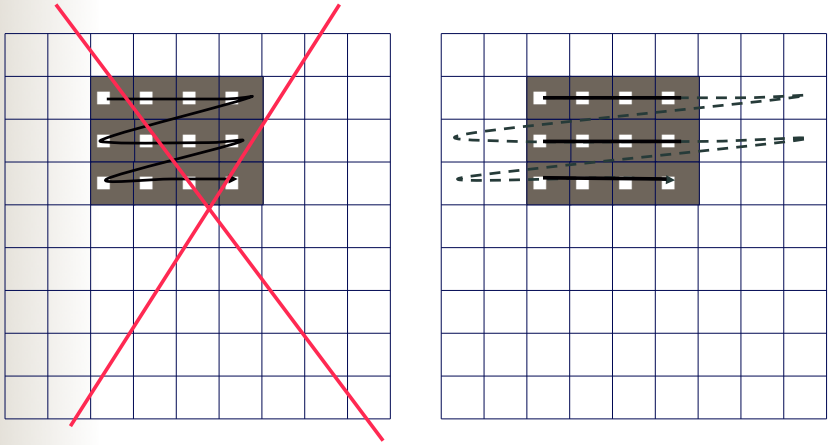
- Tailles standards
  - Mode portrait (hauteur>largeur)
  - Mode Paysage (hauteur<largeur)
    - 640*480 VGA (Video Graphics Array)
    - 948 Ko (24=3*8 bit par pixel)
    - 1,2 Mo (32 bits par pixel)
    - à 72dpi = 22,58cm * 16,93cm
    - Rapport 4/3 entre largeur et hauteur
      - 5/4 ou 4/3 ou 3/2 ou 16/10 ou 16/9

Autres



Autre instruction de base : fillrect

- Rempli un rectangle d'une couleur uniforme
  - fillRect(x, y, w, h, color) /fillRect(x1, y1, x2, y2, color)?

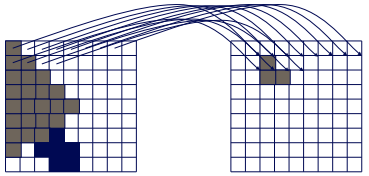


Balayage de la mémoire

- Mémoire linéaire
  - On ne balaie jamais de haut en bas !
  - Mémoire et processeur = 2 entités
- Performances
  - dépendent de la largeur de écran, jamais de la hauteur
  - Mise en cache des images et du contenu de l'écran
    - Registres (i.e. 8x64 bits)
    - L1 (i.e 64k)
    - L2 (i.e 1M) <= 1 petite image mais pas un écran entier !
    - Memoire vive =RAM (i.e 8G)
    - Disque dur (i.e 1T)

Autre instruction de base : copie

- Copie (Blit surface)
  - Copy (adresse1, x1, y1, w1, h1, addresse2, x2, y2)



  - La copie efface les pixels de la destination
  - La copie ignore les débordements
    - si (x2+w1>largeurE) copy (... largeurE-x2...)
  - La copie ne transforme pas les pixels selon leur encodage
  - La copie utilise 2 fois plus de cache !

Encodage des pixels

- RGB (pour RedGreenBlue)
  - La composante en rouge est encodée sur les bits de poids **fort** (l'équivalent des centaines en quelques sorte)
  - La composante en bleu est encodée sur les bits de poids **faible** (l'équivalent des unités en quelques sorte)
  - La composante en verte est au **milieu** (dizaine)
- chaque valeur est comprise entre 0 et 255
- La couleur qui **ajoute** du rouge et du bleu
  - = un violet/rose !
- Les couleurs sont additives (plus on en met plus c'est lumineux) contrairement à la peinture (plus on en mélange plus on a un gris foncé)
  - (r=0, g=0, b=0) => noir
  - (r=255, g=255, b=255) => blanc

Rappel

- Division entière
  - 13/5 = 2
  - pas de chiffre après la virgule
  - combien de paquets entiers
- Modulo
  - 13 mod 5 = 3 (13%5 est l'écriture informatique)
  - reste de la division entiere
  - 13 = **(13/5)***5 + **13 mod 5**
- Exemple : 5 frères héritent de 13 appartements
  - combien on donne d'appartements à chacun ? (div-ent)
  - combien on en vend pour partager ? (modulo)

Opérations spéciales

- Masque noir : & AND

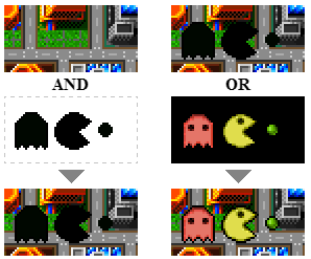
```
int pixel  = 0x20CF0200;
pixel = pixel&0x00FF0000;
println("pixel="+hex(pixel)); // pixel=0x00CF0000
```

Pixels

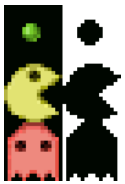
```
00100000110011110000001000000000
00000000111111110000000000000000
00000000110011110000000000000000
```

- Masque + décalage = extraction !

```
int pixel = 0x00CF0000;
pixel = pixel>>16; //dec. droite
println(...); // pixel=0xCF
```



- Sprites …
  - fond imagée



Exercice

```
int r = 255;
int g = 0;
int b = 128;
int p1 = r<<24 + g<<16 + b<<8;
int p2 = (r<<24) + (g<<16) + (b<<8);
color p3 = color(255, 204, 0);
```

Oups ! Priorité ?

Que valent p1, p2 ?

```
int p = 0x8040B000;
int r, g, b;
```

Comment récupérer les valeur de r, g, et b ?(80, 40 et B0)

Exercice

```
int r = 255;
int g = 0;
int b = 128;
int p1 = r<<24  +  g<<16  +  b<<8;
int p2 = (r<<24) + (g<<16) + (b<<8);
color p3 = color(255, 204, 0);
```

Oups ! Priorité ?

Que valent p1, p2 ?

```
int p = 0x8040B000;
int r, g, b;
```

Comment récupérer les valeur de r, g, et b ?

```
int r = red(p);
```

ou …

```
int r = (p&0x00FF0000)>>24;
```

Merci les fonctions !

Exercice

```
int r = 255;
int g = 0;
int b = 128;
int p1 = r<<24  +  g<<16  +  b<<8;
int p2 = (r<<24) + (g<<16) + (b<<8);
```

Oups ! Priorité ?

Que valent p1, p2 ?

```
int p = 0x8040B000;
int r, g, b;
```

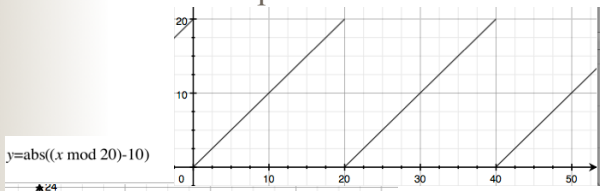
Comment récupérer les valeur de r, g, et b ?

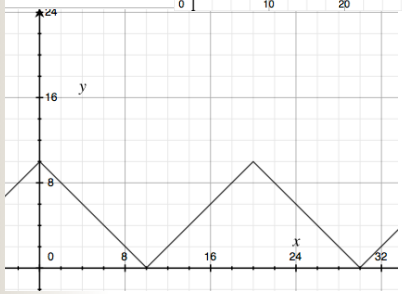
```
int r = (p & 0xFF000000) >> 24;
int g = (p & 0x00FF0000) >> 16;
int b = (p & 0x0000FF00) >> 8;
```

Regardez comme ce code est bien aligné !

Modulo et graphisme

- Modulo = reste de la division entière
  - effet de coupure



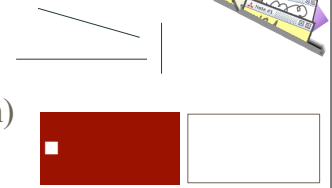


Dessin

Poser (plein de) pixels d'une couleur uniforme les uns à coté des autres

Primitives de dessin x paramètres

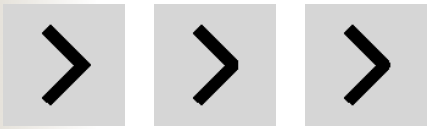
- Dessiner un Dessin sur un Dessin !!!
  - (un dessin, un endroit)
- Dessiner un segment H / V / diagonal
  - (x1, y1, x2, y2)
- Dessiner un Rectangle (rempli ou non)
  - (x, y, w, h)
- Dessiner un Cercle/Ellipse/Ovale
  - (Centre, r1, r2)
- Dessiner un texte (Font, gras, italique, etc.)
  - (Texte, position, attributs)
- Dessiner un triangle, un polygone, une courbe
  - (liste de points)



Du **texte**

x Paramètre d'affichage (attributs)

- Epaisseur du trait
- Couleur de trait + couleur de remplissage
- Anti-aliasing on/off
- Embout des lignes
- Jointure des lignes



- Ombres, halo, etc.

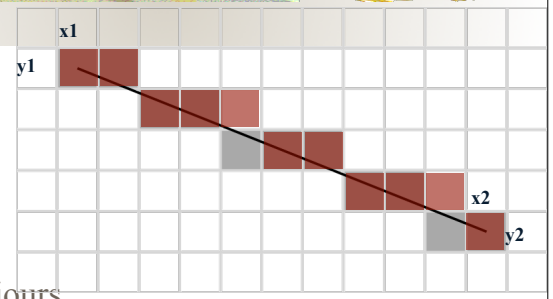
Dessiner une image

- Fonctions de blit (blitImage)
  - Recopie des pixels un à un (source⇒destination)
  - Double boucle imbriquée (ligne par ligne)
- Variantes
  - miroirs horizontal/vertical
  - changement d'encodage (ie. RGB⇒BGRA)
  - filtres ...

Mosaïque

§

- Tracé par escalier
  - Bresemham
  - Point-médian
- Algorithme
  - itératif
  - Soit x avance toujours
    - y avance ssi un accumulateur atteint un seuil (l'accu est vidé)
  - Soit y avance toujours
    - x avance ssi un accumulateur atteint un seuil (accu vidé)
  - L'accumulateur dépend de x/y ou de y/x (ie 5/11 seuil=1)
    - 0+ 5/11 + 5/11 +5/11 (!!! **-11/11**)+5/11



La fonction dessineSegment()

- Plein de ifs
  - Si un des points sort de l'écran
  - Si c'est x ou y qui avance toujours
  - Si droite⇒gauche ou le contraire (resp. haut⇒bas)
- Algo Bresenham
  - sur des entiers plutôt que des réels (id +5 et seuil à 11)
- Modification du pixels
  - En utilisant une autre fonction setPixel()
  - En colorant légèrement le pixel voisin (selon acc)

Exercice

- A partir de drawSegment(...) ou drawPixel(...)
  - Fonction Sinus remplie dans carré de cote 100px

```
void draw() {
  for (int i=0; i<width; i++)
    line(i, height, i, height-sin(i));
}
```

```
void draw() {
  for (int i=0; i<width; i++)
    line(i, height, i, height-30*sin(i));
}
```

```
void draw() {
  for (int i=0; i<width; i++)
    line(i, height, i, height/2-30*sin(i));
}
```

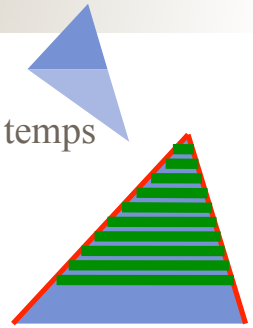
```
void draw() {
  for (int i=0; i<width; i++)
    line(i, height, i, height/2-30*sin(i*5));
}
```

```
void draw() {
  for (int i=0; i<width; i++)
    line(i, height, i, height/2-30*sin(i/5));
}
```

```
void draw() {
  for (float i=0; i<width; i++)
    line(i, height, i, height/2-30*sin(i/5));
}
```


Cas particulier : le triangle

- Le triangle est coupée en deux
  - Selon le sommet du milieu en y
- Les 2 cotés sont parcourus en même temps
  - Quand le y a avancé de chaque coté
    - On remplit la mémoire linéaire !
- Le début et la fin de chaque ligne
  - Peuvent avoir de couleurs différentes
    - Et on remplit la mémoire avec des pixels en dégradé
  - Peuvent être précédé/suivi de pixels transparent(s)
- Les pixels du contenu
  - peuvent être récupéré dans un image (texture)

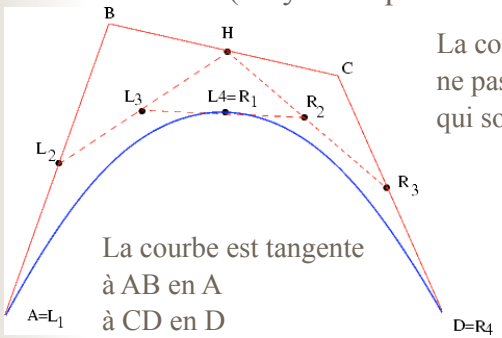


Exercice

- A partir de la primitive suivante :
  - `// dessine un triangle`
  - `void triangle(int x1, int y1, int x2, int y2, int x3, int y3)`
- Dessinez une étoile à 8 branches au centre d'un écran 640x480 qui occupe la moitié de la hauteur de l'écran
- Solution : choisir les ingrédients avant d'écrire la recette
  - Maths ...
  - Boucle
  - Itérateur
  - Variable intermédiaire
  - Accumulateur

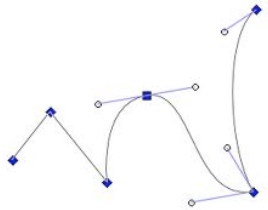
Dessin de courbes

- Plusieurs points consécutifs (i.e. 15)
  - Dessin de beaucoup de petits segments (i.e. 280)
  - Jointures parfaites
  - Ex: Bézier (moyennes pondérés des pts intermédiaires)



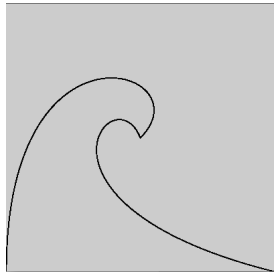
La courbe va de A à D
ne passe pas par B et C
qui sont des points de contrôle !

La courbe est tangente
à AB en A
à CD en D
ne passe pas par B et C



Exercice

- Tracé d'une vague comme ci-dessous avec
  - `bezier(x1, y1, x2, y2, x3, y3, x4, y4)`



```
void draw() {
  noFill();
  bezier(0, height, 0, 0, 3*width/4, height/4, width/2, height/2);
  bezier(width, height, 0, 3*height/4, 2*width/5, height/4, width/2, height/2);
}
```


dessin de texte

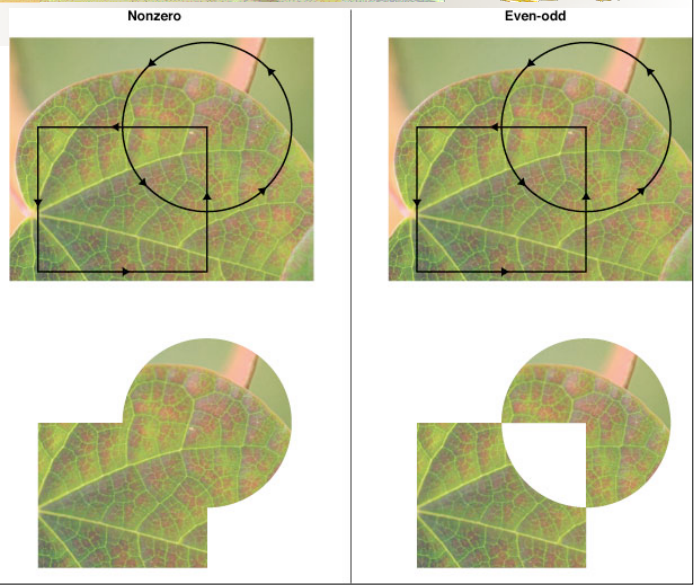
- Solution #1
  - 1 symbole/lettre = 1 image
  - texte de n mots = n×drawImage(L[Mot[i]], x+a, y)
  - 1 jeu de 200 images ×Font × Tailles (8, 10, 12, 14...)
  - A la volée ou en stock ?
- Solution #2
  - Une forme (courbes + droites) par lettres
- Besoin de connaître la taille du texte dessiné
  - hauteur=param. largeur=calcul

Clipping :

Dessin sélectif

Masque vectoriel

Encore plus de if !!!!



Primitives de dessin

- Pas toutes au même niveau/coût
  - drawRect() ⇒ 4×drawLine() ⇒ drawPixel()
  - fillRect() ⇒ n×drawLine() ⇒ drawPixel()
  - drawTriangle()⇒ n×drawLine() ⇒ drawPixel()
  - drawImage() ⇒ n×m×drawPixel()
  - drawCurve() ⇒ N×drawLine() ⇒ drawPixel()
  - drawText() ⇒ N×drawChar() ⇒
    - ⇒ drawImage()
    - ⇒ m×drawCurve()⇒ p×drawLine()⇒ k×drawPixel()
  - etc.

Double boucle imbriquée

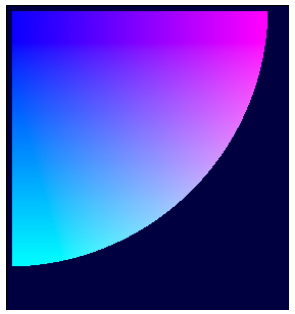
- Pixel par pixel

```
void draw() {
  background(0, 0, 128);
  loadPixels();
  for (int j = 0; j < height; j++) {
    for (int i = 0; i < width; i++) {
      pixels[j*width+i] = color(0, 255, 0);
    }
  }
  updatePixels();
}
```

Pour se mettre d'accord avec la boite à outil qui accède aussi aux pixels

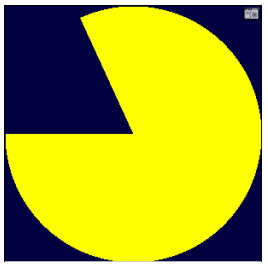
- Opportunités

```
void draw() {
  background(0, 0, 64);
  loadPixels();
  for (int j = 5; j < height; j++) {
    for (int i = 5; i < width; i++) {
      if (i*i+j*j<255*255)
        pixels[j*width+i] = color(i, j, 255);
    }
  }
  updatePixels();
}
```

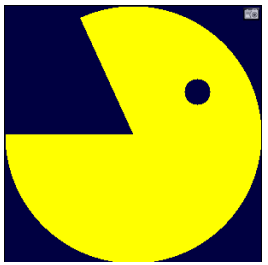


Exercice

- Faire un PacMan



- ... avec un oeil



Exercice

- Faire un PacMan

```
void draw() {
  background(0, 0, 64);
  loadPixels();
  for (int j = 0; j < height; j++) {
    for (int i = 0; i < width; i++) {
      float d = dist(i,j, width/2, height/2);
      float a = atan2(j-height/2, i-width/2);
      if (d<width/2 && a>0.5-PI)
        pixels[j*width+i] = color(255, 255, 0);
    }
  }
  updatePixels();
}
```

- ... avec un oeil

```
...
for (int j = 0; j < height; j++) {
  for (int i = 0; i < width; i++) {
    float d = dist(i,j, width/2, height/2);
    float a = atan2(j-height/2, i-width/2);
    float do = dist(i,j, 3*width/4, height/3);
    if (d<width/2 && a>0.5-PI && do>20)
      pixels[j*width+i] = color(255, 255, 0);
  }
}
...
```

Exercices

- ... et la bouche qui s'ouvre et se ferme
- dégradé de couleur (jaune + effet ombrage)
- oeil + sourcil
- fond de la bouche pas au centre
- pacman ovale

Exercice

- ... et la bouche qui s'ouvre et se ferme

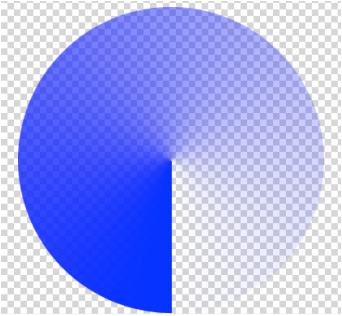
```
float ab = PI/4;
float da = 0.02;
void draw() {
  if (ab>=PI/2 || ab<=PI/8) da = da*-1;
  ab = ab+da;
  background(0, 0, 64);
  loadPixels();
  for (int j = 0; j < height; j++) {
    for (int i = 0; i < width; i++) {
      float d  = dist(i,j, width/2,   height/2);
      float d0 = dist(i,j, 3*width/4, height/3);
      float a = atan2(j-height/2, i-width/2);
      if (d<width/2 && a>ab-PI && d0>20)
        pixels[j*width+i] = color(255, 255, 0);
    }
  }
  updatePixels();
}
```

- ... on en reparlera quand on verra l'animation !

Transparence

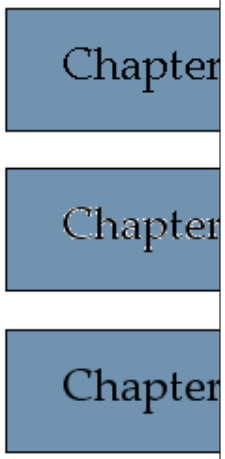
Transparence

- Jusqu'à maintenant quand on dessinait ... on effaçait !
- Propriété d'un graphisme laissant penser à l'utilisateur qu'il perçoit 2 parties l'une par dessus l'autre
- Propriété d'un pixel définissant comment il s'ajoute à celui du dessous … lorsqu'il **sera** copié

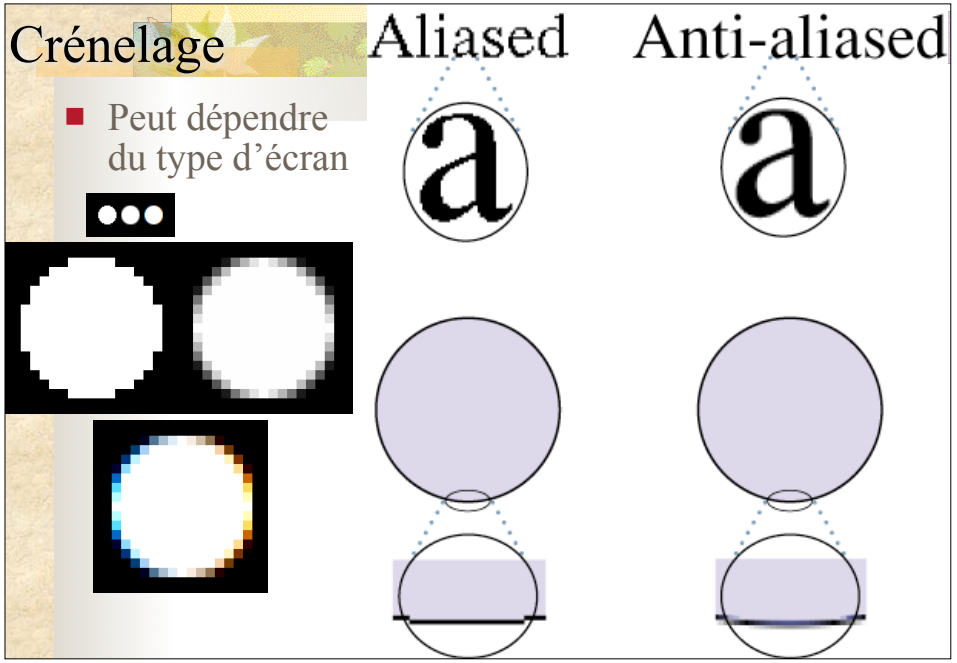


Transparence

- Implicite dans le dessin vectoriel
  - Pas de forme = transparence
  - + couleurs translucides
- EXPLICITE dans le dessin matriciel
- Transparence totale
  - Le pixel laisse voir ce qu'il y a derrière
- Transparence partielle
  - Le pixel laisse voir ce qu'il y a derrière mais le teinte un peu d'une autre couleur
- Transparence de bord
  - Eviter les effets d'escalier (aliasing)



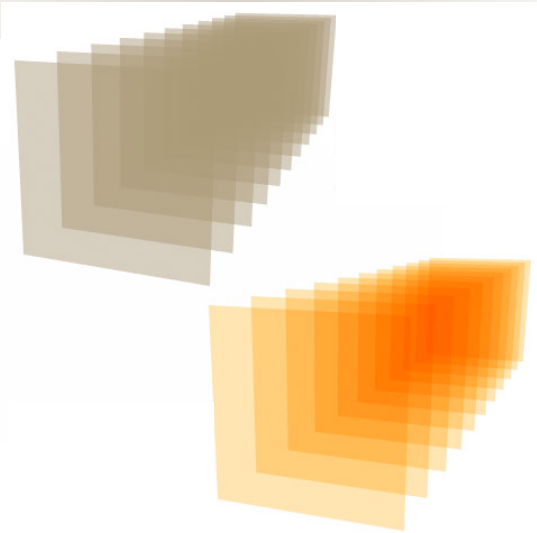
Crénelage



- Peut dépendre du type d'écran

Transparence

- Transparence Brouillard
- Transparence cellophane



Composition de niveaux de couleur

- G1 et G2 deux niveaux de gris (ex : 0x00 et 0x88)
  - Couleur finale = (G1+G2)/2 (donc 50%-50%)
    - Si G1=G2 ... Couleur finale = G1 = G2
    - Commutatif : (G1+G2)/2 = (G2+G1)/2
      - Pas d'importance de l'ordre. Si G1 par dessus G2 ou G2 sur G1
    - Pas associatif ... ((G1+G2)/2+G3)/2 ≠ (G1+(G2+G3)/2)/2
      - Importance de l'ordre au delà de 2 filtres transparents !
- Cas d'une transparence inégale : (G1+3*G2)/4
- Et en couleur … (R1G1B1 + R2G2B2)/2
  - (R1+R2)/2 (G1+G2)/2 (B1+B2)/2
  - (3*R1+R2)/4 (G1+G2)/2 (B1+7*B2)/8 … rare
- Si le coefficient est toujours le même pour les 3 channels
  - Coder avec le pixel transparent son coefficient
  - 0xC04080 (fond) + 0x8000FF40 (<- le deuxième pixel à 25%)

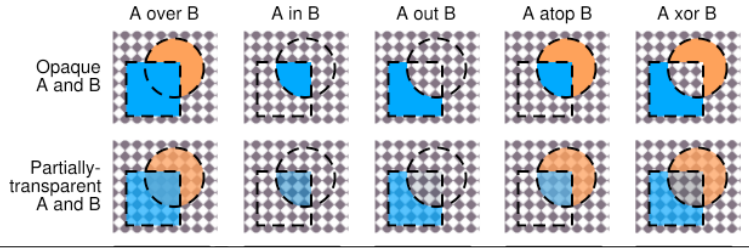
RGBA A=Alpha Channel = Opacity ≠transparency

Composition alpha

- Duff et Porter : 4 autres compositions
  - α=0 Complètement transparent
  - α=255=0xFF Complètement opaque
  - OVER, IN,OUT, ATOP et XOR

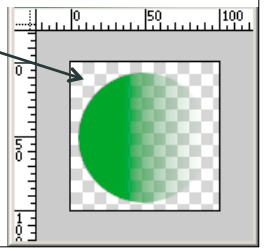
$$C_o = C_a\alpha_a + C_b\alpha_b\,(1-\alpha_a)$$
$$\alpha_o = \alpha_a + \alpha_b\,(1-\alpha_a)$$

Le pixel résultat a lui aussi son opacité !



Transparence et format de fichier

- Bmp : RGB 16M pas compressé, pas de transparence !
- Gif : 256 couleurs dont 1 peut être la transparence totale
- Jpg : 16 millions de couleurs mais pas de transparence
- Png et Tiff: 16 millions de couleurs et 256 niveaux de transparence : RGBA
  - 32 bits au lieu de 24
  - A fort niveau de transparence toutes les couleurs se ressemblent
  - La transparence ne se voit qu'avec quelque chose derrière
  - Quadrillage blanc-gris par convention
- Le quatrième canal est bien plus compréhensible lorsqu'on le mélange aux autres … ne pourrait-on pas avoir les trois premiers canaux tout aussi intuitifs ?
-



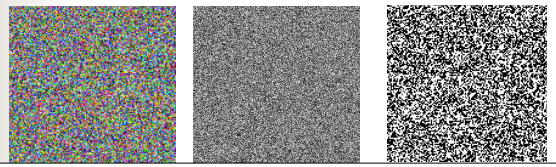
Hasard et graphisme

```
for (int i=0; i<22; i++) {
  int v = int(random(0, 255));
  println(v);
}
```

12 77 124 11 39 206 45 4 16 167 148 62 250 174 225 196 10 193 68 123 165 94

Nombres aléatoires

- Les ordinateurs ne peuvent pas produire des nombres au hasards car ils sont déterministes (prévisibles)
- Les ordinateurs peuvent sortir des nombres pseudo-aléatoires 3235646545, 1905468903, 2658963165…
  - Des nombres qui ONT L'AIR d'être aléatoires
  - En fait une suite mathématique TRES compliqué
  - Si l'ordinateur commence par 3235646545 =>1905468903
- A quoi ressemble une image de nombres aléatoires



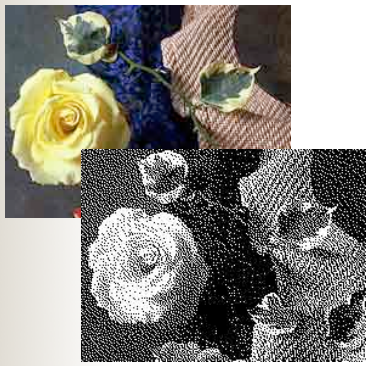
Essayons sous sketchpad !

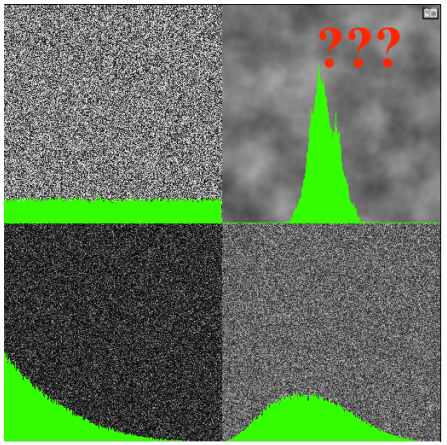
Bruit

- L'affichage de nombre aléatoire
  - Nécessite un modulo pour retailler les nombres
  - Ressemble à de la neige de TV
  - Forme des paquets
  - Permet de rajouter du réalisme à certain graphiques
    - Affichage d'étoiles dans le ciel …
- Se combine aux propriétés des pixels
  - r, g, b, luminosité, teinte, posx, posy, tps d'affichage, etc
- Cela ressemble t-il à la nature ?
  - A la surface de l'eau ?
  - A la répartition des arbres dans un foret ?

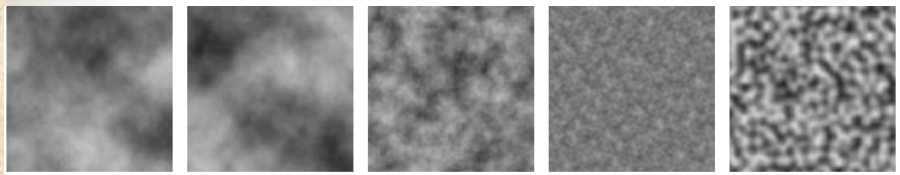
Autres bruits

- Modulation des fréquences d'apparition des valeurs
  - Un 0 n'a pas la même chance de tomber qu'un 255 !





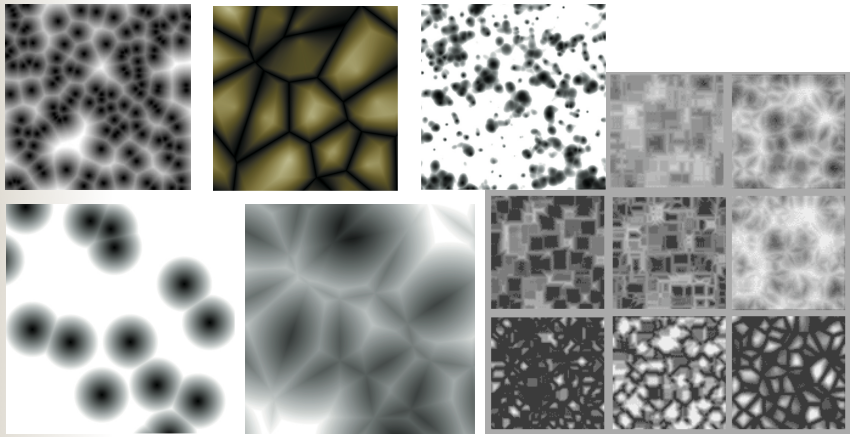
- Probabilité d'apparition en fonction des voisins !
  - Forte notion de graphisme !
  - Certaines valeurs ont plus de chance d'apparaître à coté de certaines valeurs
  - Le générateur aléatoire doit savoir OU vous affichez !



  - noise (i, j) ≠ noise (i/10, j/10) => effet de zoom
  - On ne peut pas zoomer à l'infini (nombre d'octave)
  - On peut ajouter des bruits à différents niveau de zoom

Bruit de Worley

- Valeur = Distance au point le plus proches
  - parmi n points pris au hasard !

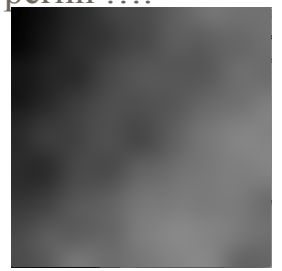


Et la beauté fut !

Exercice

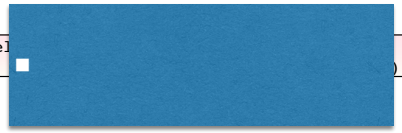
- A partir de l'affichage dun bruit de perlin ….

```
public void setup(){
  size(512, 512);
  noLoop();
}

public void draw(){
  loadPixels();
    for (int j=0; j<height; j++)
      for (int i=0; i<width; i++)
      pixels[j*height+i] = color(
        255*noise(i/512.0,j/512.0));
  updatePixels();
}
```

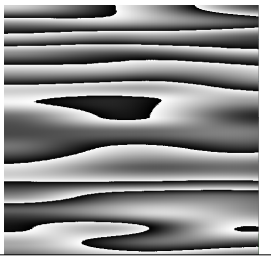


- Dessiner des veine

```
pixel
```

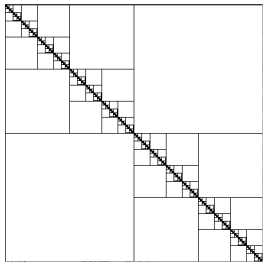


```
);
```



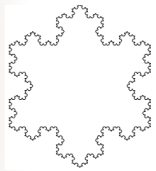
Dessin récursif

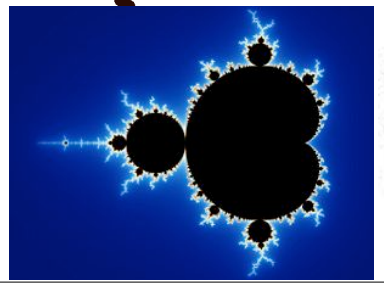
```
void drawR(int x, int y, int w, int h) {
  rect(x,y,w,h);
  if (w>1 && h>1){
    drawR( x,  y, w/2,  h/2);
    drawR(x+w/2,  y+h/2, w/2,  h/2);
  }
}

void draw() {
  drawR( 0,  0, width,  height);
}
```



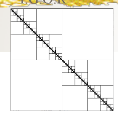
Récursivité

- Beau … mais est-ce très utile ?
  - Fractales





Ingrédients



- Une fonction qui s'appelle elle même !
- Des paramètres qui évoluent
- Un test pour s'arrêter
- Un dessin simple
- Un appel de départ

```
void drawR(int x, int y, int w, int h) {
  rect(x,y,w,h);
  if (w>1 && h>1){
    drawR( x,   y, w/2,  h/2);
    drawR(x+w/2,  y+h/2, w/2,  h/2);
  }
}

void draw() {
  drawR( 0,  0, width,  height);
}
```

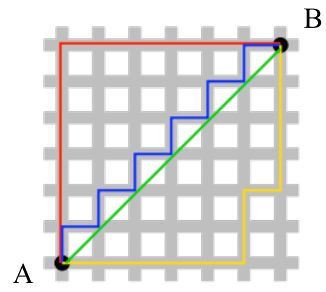
void drawR(int x, int y, int w, int h) {
rect(x,y,w,h);
if (w>1 && h>1){
drawR(x, y, w/2, h/2);
drawR(x+w/2, y+h/2, w/2, h/2);
}
}
void draw() {
drawR(0, 0, width, height);
}

void drawR(int x, int y, int w, int h) {
rect(x,y,w,h);
if (w>1 && h>1){
drawR(x, y, w/2, h/2);
drawR(x+w/2, y+h/2, w/2, h/2);
}
}
void draw() {
drawR(0, 0, width, height);
}

Distances

Quel est le plus court chemin d'un point A à un point B ?

Quelles sont tous les points équidistants d'un point A ?



4 distances importantes

- s'appliquent à n dimensions

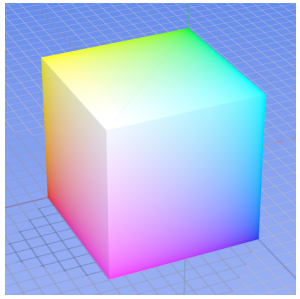
Nom	Paramètre	Fonction
distance de Manhattan	1-distance	$\sum_{i=1}^{n} \lvert x_i - y_i \rvert$
distance euclidienne	2-distance	$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
distance de Minkowski	p-distance	$\sqrt[p]{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p}$
distance de Tchebychev (en)	∞-distance	$\lim_{p\to\infty} \sqrt[p]{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p} = \sup_{1\leq i\leq n} \lvert x_i - y_i \rvert$

- distances à O(0,0) mais d(x-xA, y-yA) = distance à A

en 2 dimensions ...

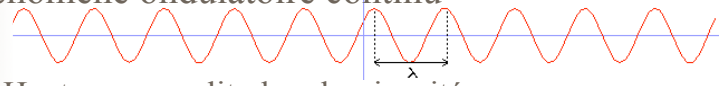
- Distance de Manhattan (NYC)
  - d = |xA-xB|+|yA-yB|
- Distance euclidienne
  - d = $\sqrt{(xA-xB)^2+(yA-yB)^2}$
- Distance de MINKOWSKI... pour p=5
  - d = $\sqrt[5]{|xA-xB|^5+|yA-yB|^5}$
- Distance de TCHEBYCHEV (roi du plateau d'échec)
  - d = max(|xA-xB|, |yA-yB|) p=

Espaces de couleurs

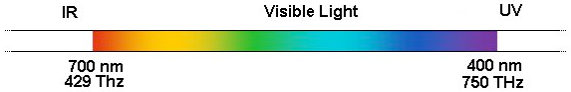


Couleurs

- Phénomène ondulatoire continu

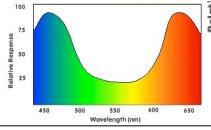


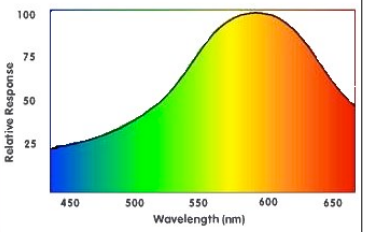
  - Hauteur = amplitude = luminosité
  - Largeur de la sin. = longueur d'onde = teinte



- Perception d'un mélange de plusieurs signaux par des capteurs R G et B

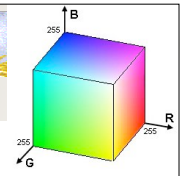
Violet =

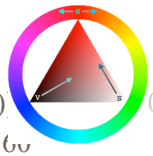


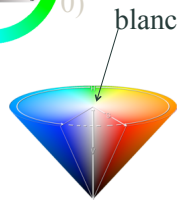


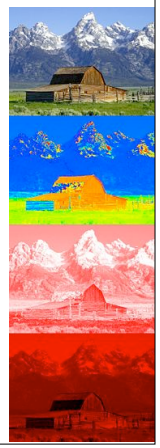
RGB et HSV

- RGB mélange rouge vert et bleu de façon additive … pas très naturel
- HSV décompose l'espace 3D des couleurs selon 3 autres axes
  - Hue : teinte 0-255 décrit tout l'arc en ciel
  - Saturation : couleur <=> vive-couleur <=>pastel-gris
  - Value(≈luminosité)
- Transformation
  - Value = max (r, g, b)
  - S = 1-(min(r,g,b)/max(r,g,b)) 0)
  - H= dépend du max … 0°- 360









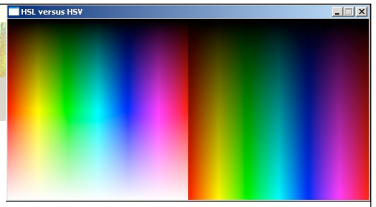
Teinte (H) de HSV et HSL

- HSV

$$S = \begin{cases} 0, & \text{if } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{otherwise} \end{cases}$$

$$V = MAX$$

$$h = \begin{cases} \text{undefined} & \text{if } max = min \\ 60° \times \frac{g-b}{max-min} + 0°, & \text{if } max = r \text{ and } g \geq b \\ 60° \times \frac{g-b}{max-min} + 360°, & \text{if } max = r \text{ and } g < b \\ 60° \times \frac{b-r}{max-min} + 120°, & \text{if } max = g \\ 60° \times \frac{r-g}{max-min} + 240°, & \text{if } max = b \end{cases}$$



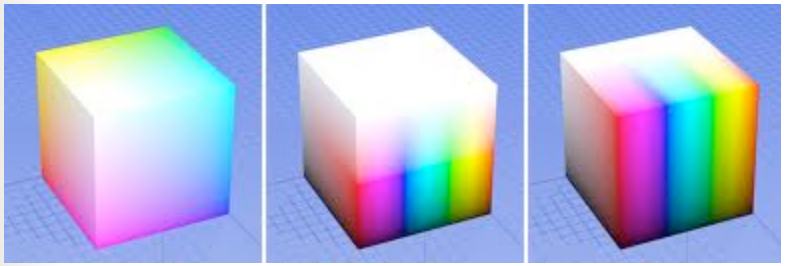
- HSL

$$l = \tfrac{1}{2}(max + min)$$

$$s = \begin{cases} 0 & \text{if } l = 0 \text{ or } max = min \\ \frac{max-min}{max+min} = \frac{max-min}{2l}, & \text{if } 0 < l \leq \frac{1}{2} \\ \frac{max-min}{2-(max+min)} = \frac{max-min}{2-2l}, & \text{if } l > \frac{1}{2} \end{cases}$$

Codage

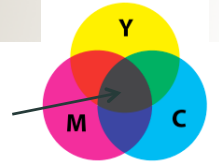
- Au delà du modèle
  - Un codage sur 8+8+8 bits fera toujours un cube !

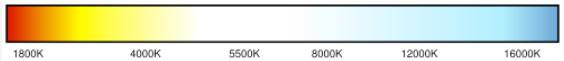


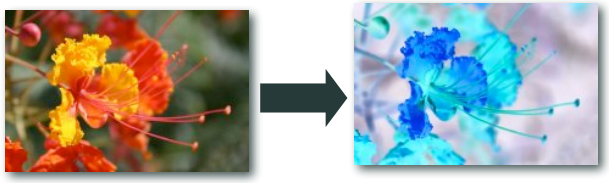
RGB HSL HSV

Et encore

- CMJK : couleurs soustractives pour les encres des imprimantes
- Couleurs complémentaires
  - Mélange des deux = niveau de gris
- Couleurs **additives** RGB
- Couleurs chaudes/froides
- Inverse vidéo
- Filtres…





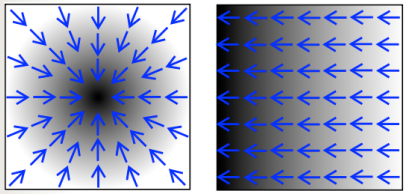


Ombres, Floues et Halos

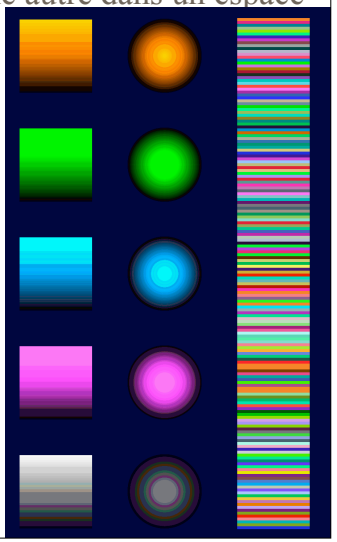
Dégradés, Ombres, Floues, Halos et Réflexion

Dégradé de couleur (gradient)

- Passage progressif d'une couleur à une autre dans un espace à 2 dimensions
  - Gradient linéaire
  - Gradient circulaire

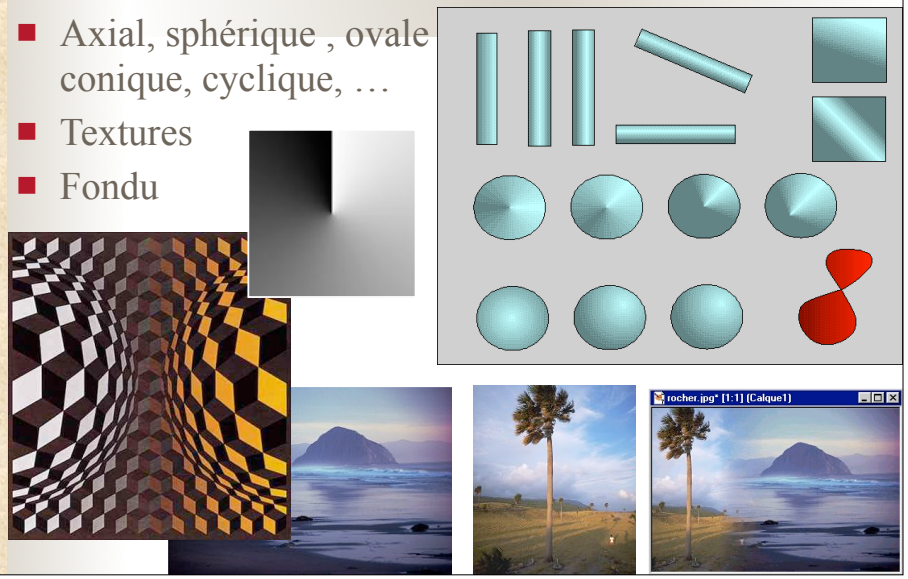


- Gradient bi-linéaire
- Gradient multiple



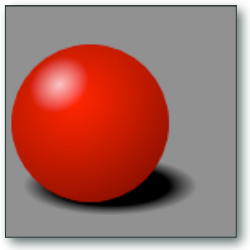
Autres dégradés

- Axial, sphérique , ovale conique, cyclique, …
- Textures
- Fondu



Ombres portées, ombrage et reflet

- La 3D n'est qu'une illusion de l'œil …
  - Décomposer
  - Comprendre
  - Modéliser
  - Coder 1 à 1
  - Mélanger

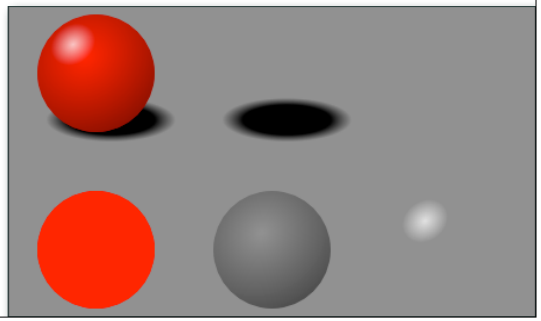


- Quelles primitives sont nécessaires ?
  - `// dessine un cercle de centre cx,cy rempli de la couleur r,v,b ….`
  - `void fillCircle(int cx, int cy, int rayon, int r, int v, int b)`
- Quelles autres primitives sont nécessaires ?

Exercice


```
// ou bien
void fillCircle(PImage i, int cx, int cy, int rayon, int color)

// ou bien
void fillCircle(PImage, int cx, int cy, int rx, int ry, int color)

// et donc
PImage *blur(PImage i, int fx, int fy)

Sphere degradée de couleur ?

Reflet ?
```

- Pensez à
  - la généricité
  - La re-utilisation



Halo, ombres, flou

- Agrandissement, masque et flou gaussien,

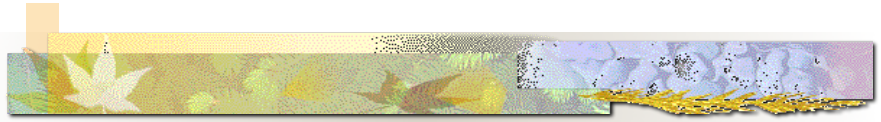
Flous

- Flou de vitesse (cinétique) =>
- Flou Gaussien

- Matrice :

0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067
0.00002292	0.00078633	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00038771	0.01330373	0.11098164	**0.22508352**	0.11098164	0.01330373	0.00038771
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00002292	0.00078633	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067

Composition



Couches d'affichage, Transformation, double buffering et dessin caché

Couches d'affichage (layers)

- Un affichage produit par un ordinateur est presque systématiquement un assemblage de couches
- L'ordre des couches et l'inverse de leur affichage
- La dernière couche est supposée opaque
- Par défaut une couche est transparente
- 2 couches non-recouvrantes peuvent être interverties sans modification visible
- Plusieurs couches peuvent être pré-calculées en une couche plus rapide à afficher
- En mémoire centrale / vidéo
- Couches Reliées / Dissociées (glissement)
- Affichage dégradé pour certaines optimisations

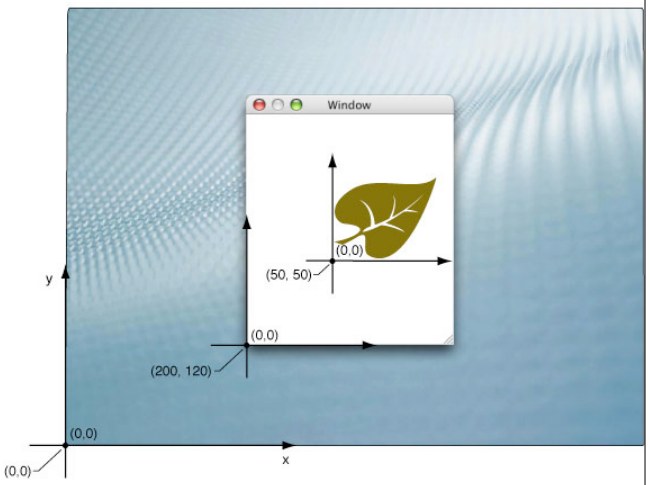
Boites à Outils graphiques

- Utilisation des primitives gardées
  - Pas de dessin dans les zones hors-image
  - Plus de if = moins de performances
- Utilisation de variables globales
  - Pour éviter la répétition d'arguments
    - Couleurs(fond et dessin), épaisseur, clipping, police, etc.
- Dessin à l'écran ou dans une image cachée
- Permettent l'affichage plein écran (fullscreen)
  - Ou l'affichage dans des fenêtres de l'OS
- Gèrent le dessin transformé
  - …

Transformation géométriques

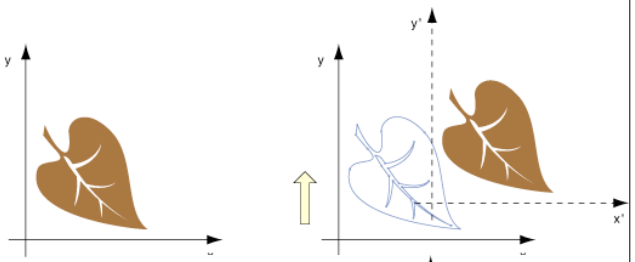
- Transformer puis dessiner !

sans transformation

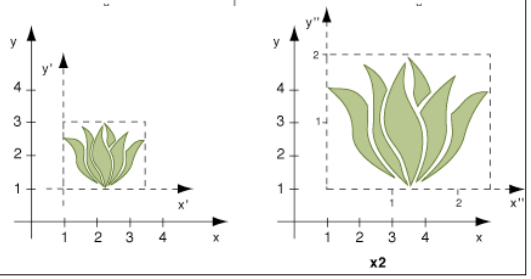


Translation & Mise à l'échelle

- Décalage de tous les dessin futurs

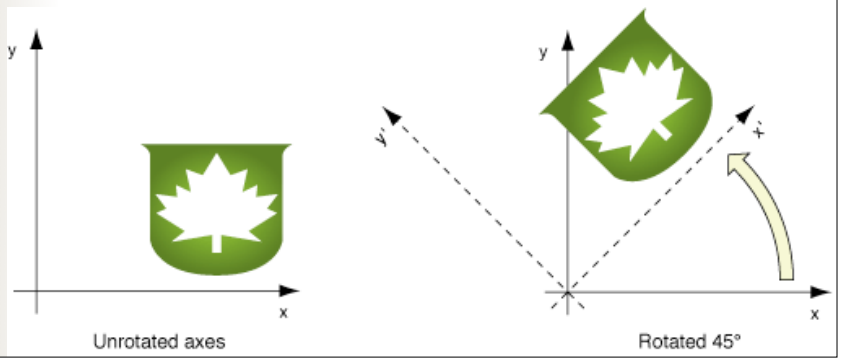


- Scale = zoom de tous les dessins futurs



Rotations

- S'applique autour de ce qui aurait été le point 0,0
- Attention au effets d'escalier sur le texte
- Possibilité de concaténer des transformations …



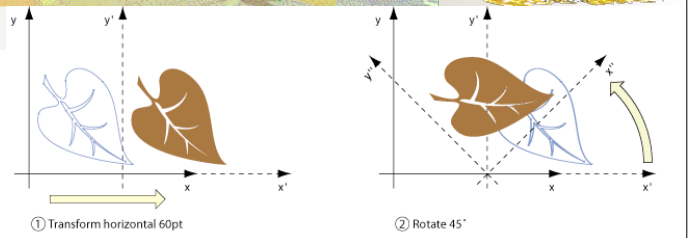
Rotation et Translation

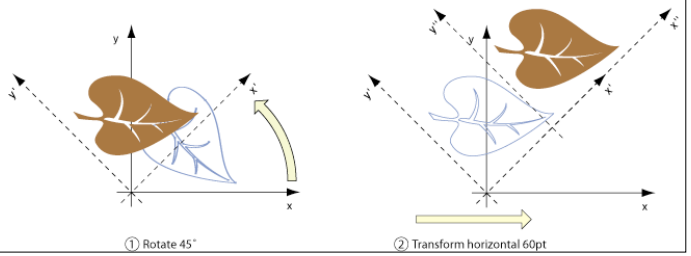
Importance de l'ordre

Trans+Rot

≠

Rot+Trans





Animation

- Modifications successives du graphisme
  - Déplacement, zoom, rotation
  - Propriété(s) d'un effet variant dans le temps (taille du halo, propriété du dégradé)
  - Images ressemblantes successives (explosions, marche...)
- Organisation temporelle de l'animation
  - Attente pour éviter de dessiner plus que 25x / sec
  - Echelle de progression (linéaire, accélérante, par à coup)
  - Gestion spéciale du début et de la fin
  - Eviter l'effet d'aveuglement dû au clignotement

....

http://fr.wikipedia.org/wiki/Animation

Double buffering 1/2

- Eviter ce clignotement


```
Tant que (NON(FINI)) Faire
  t ← tempsActuelEnMs();
  dessinerRectangle (screen, 0, 0, largeur, hauteur, 0x00000000);

  dessinerCoucheDuFond(screen, …);

  dessinerCoucheDuDessus(screen, …);

  dessinerElementActif(screen, …);

  tpsEcoule  = tempsActuelEnMs()-t;

  si (tpsEcoule <20) alors
    attendre(20-tpsEcoule);
  fin si

Fin Tant Que
```

Découpage de la fonction de dessin en « couches »

Attente pour garantir le rythme de l'animation

Double buffering 2/2

- Principe
  - Dessiner dans une image aussi grande que l'écran qui n'est pas l'écran.
  - Quand le dessin est fini et que l'im age ainsi préparée ressemble le plus possible à l'écran visible on « bascule » l'image invisible sur l'écran visible

```
Buffer ← allocationImage(ecran ->largeur, ecran->hauteur);
Tant que (NON(FINI)) Faire
  t ← tempsActuelEnMs();
  dessinerRectangle(buffer, 0, 0, largeur, hauteur, 0x00000000);
  dessinerCoucheDuFond(buffer, …);

  dessinerImage(ecran, buffer, …);
  si (tpsEcoule <20) alors
    attendre(20-tpsEcoule);
  fin si
Fin Tant Que
```


Dessin caché (offscreen)

- Créer un contexte graphique
- Activer ce contexte graphique
- Dessiner dessus au lieu de l'écran
- Afficher le résultat comme une image

```
PGraphics pg;
void setup(){
  size(800, 600);
  pg = createGraphics(300,200);
  pg.beginDraw();
    pg.stroke(255, 128, 0);
    pg.line(0,0,300, 200);
    pg.line(0,200,300, 0);
    pg.ellipse(150, 100, 30, 20);
  pg.endDraw();
}


void draw(){
  image(pg, width/2, height/2);
}
```

Dessin caché (opportunité et limites)

- Faire un dessin couteux (lent)
  - Afficher rapidement l'image
- Faire une dessin droit (ie rect)
  - Afficher en tournant (rect au lieu de polygone)
- Dessiner à haute résolution
  - afficher avec un scale => antialiasing
- Limitation No1 = pas de pos des elems au dessin
  - Pas de if => descr. logique des elems de dessin
- Faire une image noir et blanc
  - Tinter l'image
  - Ne marche pas à cause d'un bug processing !

Solution


```
PImage    pi;
PGraphics pg;
float     t=0;

void setup(){
  size(800, 600);
  pg = createGraphics(300,200);
  pi = createImage(300,200, ARGB);
  pg.beginDraw();
   pg.strokeWeight(3);
   pg.fill(128);
   pg.stroke(192);
   pg.line(0,0,300, 200);
   pg.line(0,200,300, 0);
   pg.ellipse(150, 100, 90, 60);
  pg.endDraw();
  pi.copy(pg, 0,0,pg.width, pg.height,
              0,0,pi.width, pi.height);
}
```

```
void draw(){
  background(255);
  imageMode(CENTER);
  colorMode(HSB);
  t=t+0.1;
  pushMatrix();
   translate(width/2,
             height/2);
   rotate(t);
   tint(t, 255, 255);
   image(pi, 0,0);
  popMatrix();
}
```

Interaction

Evénement, callbacks et machines à état

Interaction Humain-Graphique

- Dispositif de pointage (x,y)
  - Souris
  - Pointing stick
  - TouchPad
  - Molette 1D / 2D avec/sans cran
  - Boutons
  - Evt=Move / Press / Release / Drag / Scroll / Click / Over
  - 120 Move ou Drag à la seconde
- Claviers
  - Boutons poussoirs (caps lock avec LED feedback)
  - Evt = Press, Release, Type

Gestion des événements

- Méthodes de callback
  - `mouseClicked()`
  - `mouseDragged()`
  - `mouseMoved()`
  - `mousePressed()`
  - `mousePressed`
  - `mouseReleased()`
  - `keyPressed()`
  - `keyReleased()`
  - `keyTyped()`

Appel à redraw() explicite…

Possibilité de démarrer/arrêter l'animation

```
noLoop();
loop();
```

- Utilisation de variables globales dans draw()
  - mouseButton
  - mouseX
  - mouseY
  - pmouseX
  - pmouseY
  - key
  - keyCode
  - keyPressed

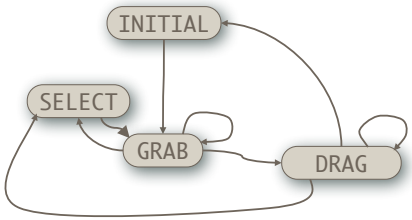
Utilisables dans les callbacks aussi !

Un peu de méthode

- Création
  - mousePressed() et mouseReleased() en même temps
  - Testez le fonctionnement avec println("toto")
  - Testez l'évolution des variables println("mx: "+mousex);
- Débogage
  - Traces (println) indentées : evol. d'une variable
  - Dessins intermédiaires (boite englobante, diagonale, repère)
  - Découpage du dessin en fonctions
    - Factorisation
    - Versions du graphismes
- Optimisation : millis() + traces

Conception d'un système interactif

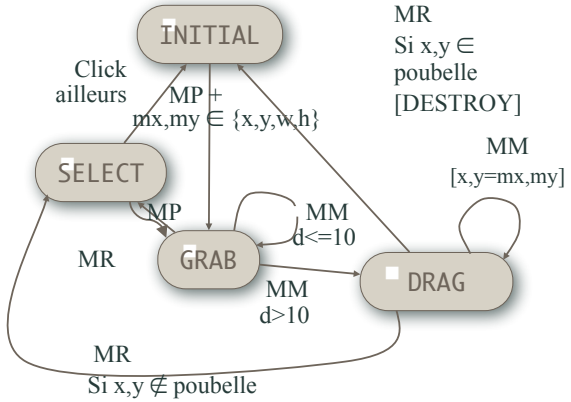
- Réfléchir avant de coder
  - Concevoir = prévoir !
  - Besoin d'outils pour s'abstraire du code
- Machine à état finis
  - Voir le logiciel graphique comme un robot
  - Est dans un certain état a tout moment
  - Change d'état à cause de certains évènements
    - Etat = affichage différent
    - Transitions = code callbacks



Système interactif

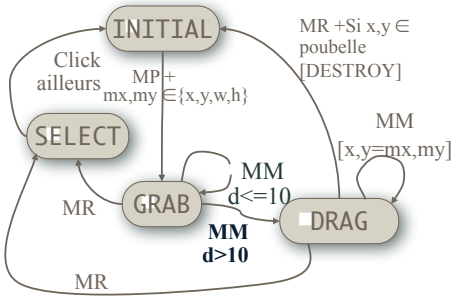
- Décrit le comportement de l'application par un ensemble de transitions

```
int INITIAL  = 0;
int SELECT   = 1;
int GRAB     = 2;
int DRAG     = 3;

etat = INITIAL;
```



Codage d'une transition

```
void mouseDragged(){
  int dx  = mouseX-pmouseX;
  int dy  = mouseY-pmouseY;
  float d = sqrt(dx*dx+dy*dy);
  if (state==GRAB && d>10) {
    state = DRAG;
  }
}
```



ERREUR !
L'image ne décolle pas !
… sauf à donner un grand coup de souris

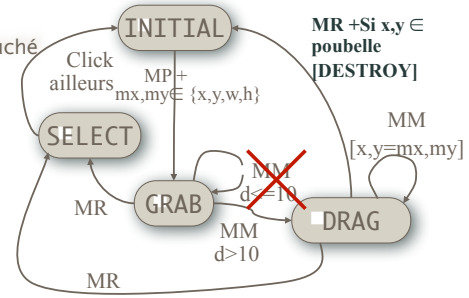
Solution

- Utilisation hasardeuse de pmouseX
  - Doc dit pmouseX = version précédente de la souris
  - On doit **créer sa propre variable globale**

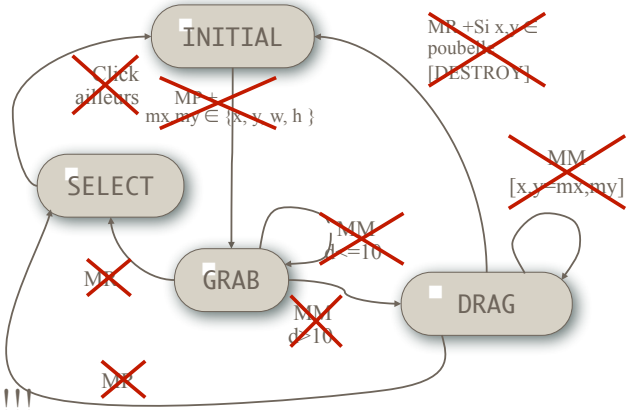
```
int mousePressX, mousePressY;

 void mousePressed(){
    mousePressX = mouseX;
    mousePressY = mouseY;
  }
  void mouseDragged(){
    int dx  = mouseX-mousePressX;
    int dy  = mouseY-mousePressY;
    float d = sqrt(dx*dx+dy*dy);
    if (state==GRAB && d>10) {
      state = DRAG;
    }
}
```


Codage d'une transition-Action

```
int done = 0;
while (!done) {
  // message processing loop
  SDL_Event event;
  while (SDL_WaitEvent(&event)) {
     // check for messages
     switch (event.type) {
        // evt du bouton souris relaché
        case SDL_MOUSEBUTTONUP:
          if (x>px && x<px+pw &&
              y>py && y<py+ph &&
              state==DRAG) {
            state = INITIAL;
            // [DESTROY]
          }
          break;
        ...
     } // end switch
  } // end of message processing
  ...
```



@



ET C'EST FINI !!!

On ne code rien pour les états : Juste les transitions

Format des événements

- event.type
  - SDL_MOUSEBUTTONDOWN
  - SDL_MOUSEMOTION
  - SDL_MOUSEBUTTONUP
  - SDL_MOUSEKEYDOWN
  - SDL_MOUSEKEYUP
  - SDL_USEREVENT

event.button

```
typedef struct{
  Uint8 type;
  Uint8 button;
  Uint8 state;
  Uint16 x, y;
} SDL_MouseButtonEvent;
```

```
if (event.button.button==
    SDL_BUTTON_LEFT) {…}
```

event.motion

```
typedef struct{
  Uint8 type;
  Uint8 state;
  Uint16 x, y;
  Sint16 xrel, yrel;
} SDL_MouseMotionEvent;
```

```
Posx = event.motion.x;
```

event.key

```
typedef struct{
  Uint8 type;
  Uint8 state;
  SDL_keysym keysym;
} SDL_KeyboardEvent;
```

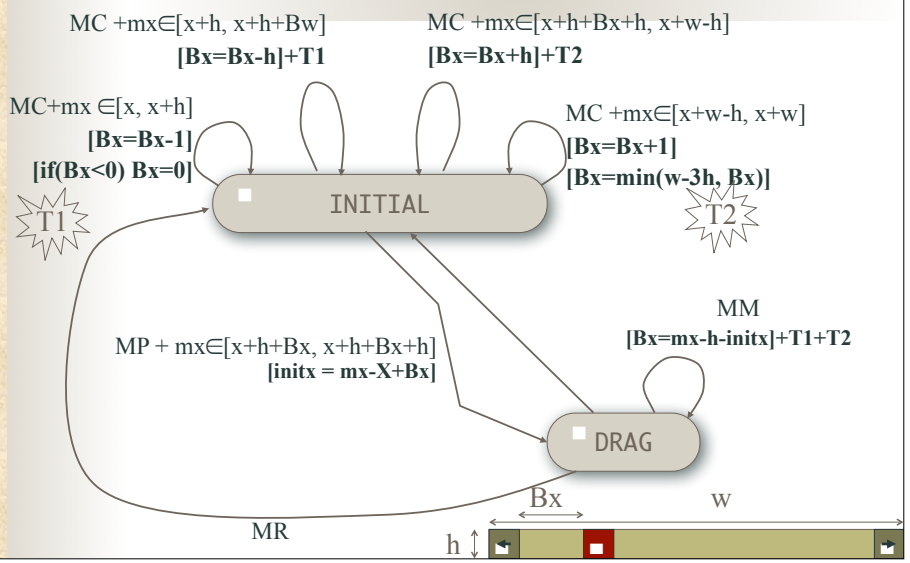
```
if(event.key.keysym.sym==
   SDLK_LEFT) {…}
```

event.user

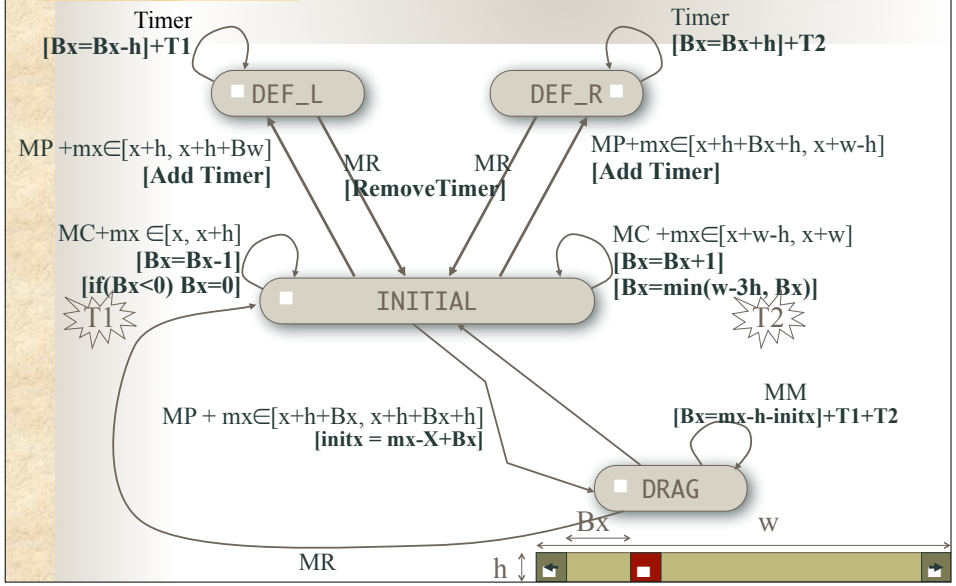
```
typedef struct{
  Uint8 type;
  int code;
  void *data1;
  void *data2;
} SDL_UserEvent;
```

```
SDL_Event event;
event.type=SDL_USEREVENT;
event.user.code=my_code;
event.user.data1 = my_data;
event.user.data2 = 0;
```


Exemple : la barre de défilement v1



Exemple : la barre de défilement v2



Exemple : Codage

- Une structure représentant la barre

```
class BoiteBoite{
  int x;
  int y;
  int w;
  int h;
  int Bx;
  int state;
  int timer_id;
}
```

- Une fonction + Tests pour toutes les modifs

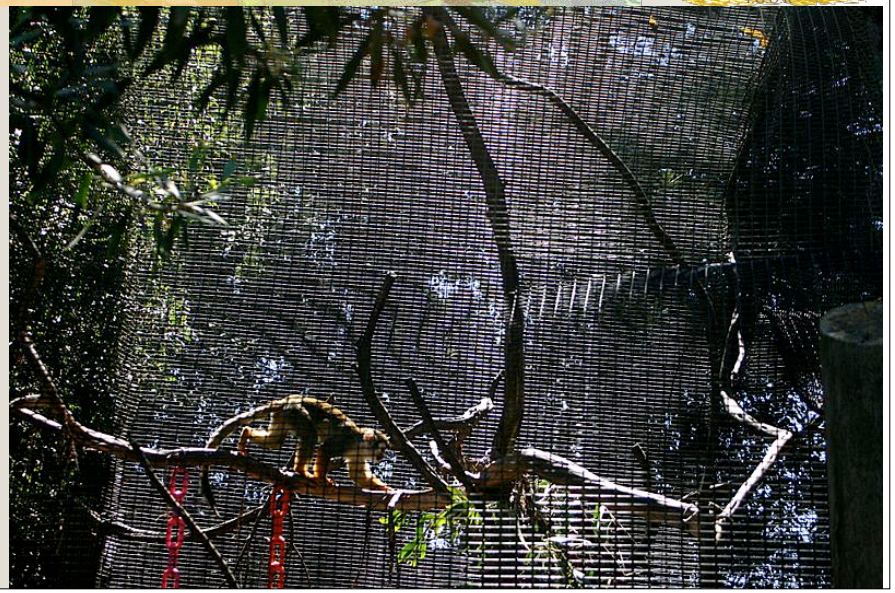
```
void setBx(int newBx){
  if (newBx>x+h && newBx<x+w-h)
    Bx=newBx;
}
```

- Just Do It !
- Barre Proportionnelle ? Intervalle ? ScrollArea ?
- Inertie ? Histogramme ? Graduation ?

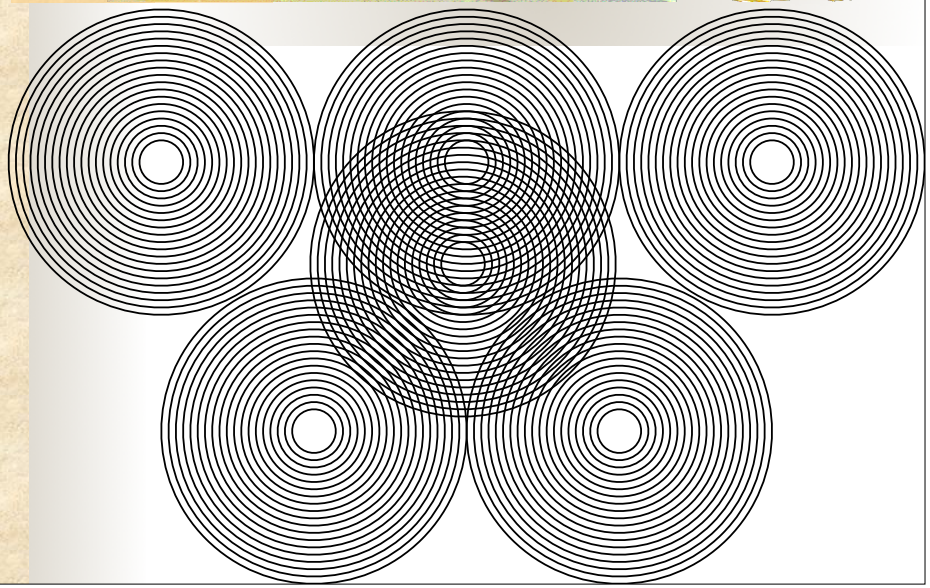
Introduction au Graphisme des ORdinateurs

Quelques trucs divers à connaître

Effet de Moiré

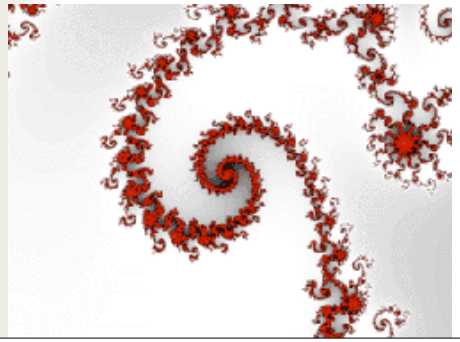


… Vous êtes en mon pouvoir !



Dessin Récursif

- Fractals
- Végétation



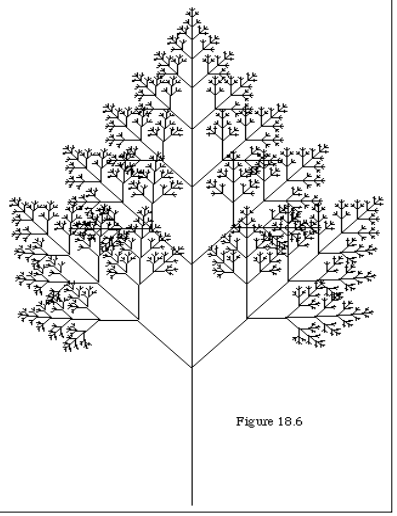
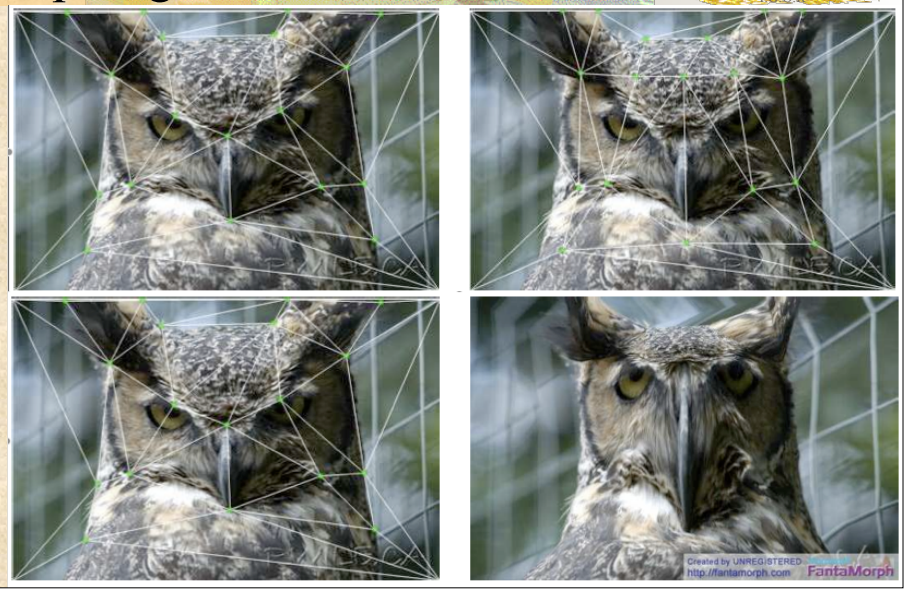


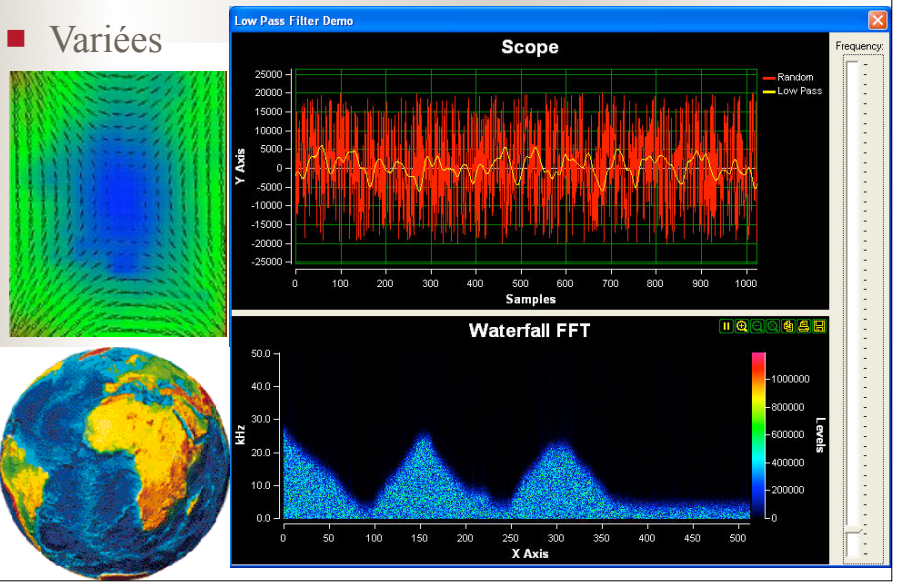
Figure 18.6

Morphing

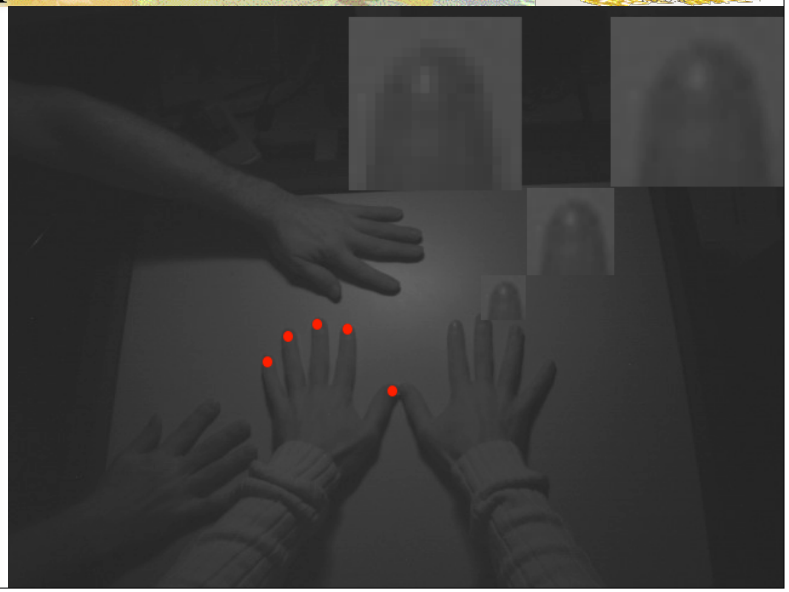


Visualisation

- Variées



Vision par ordinateur

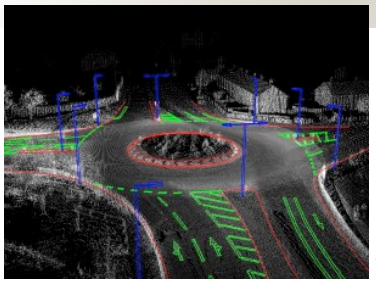


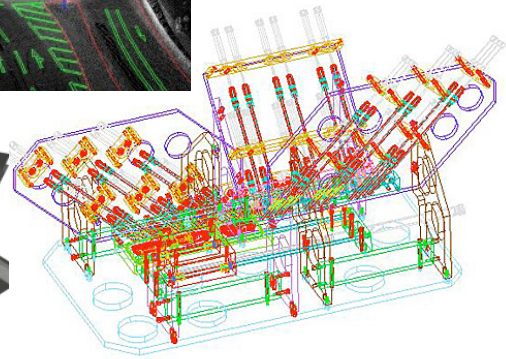
Dessin en 3 dimensions

Effet Spéciaux

- Mélange de plusieurs aspects
  - Vision pour détecter / calibrer
  - Graphisme 2D / 3D
  - Animation

C.A.O / C.A.D.





Introduction au Graphisme des ORdinateurs

Comment intégrer le graphisme dans un programme ?

Dessin à partir de modèles

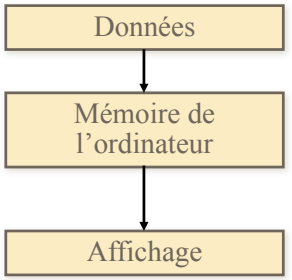
- Quel sont les différences / points communs entre vison par ordinateur et graphisme ?
  - Graphisme = Dessiner à partir de modèle(s)
  - Vision par ordinateur = retrouver le(s) modèle(s) a partir du dessin
- Modèle de quoi ?
  - Organisation spatiale de nombreux éléments similaires
    - Vision=comptage, Graphisme = Stockage/Boucles/Affichage
  - Organisation d'éléments hétérogènes
    - Vision=détection, Graphisme = Stokage/cas par cas/Affichage
  - Couleurs
    - Vision=calibrage, Graphisme = Visualisation/Effets graphiques

Modèle de données

- Données bruts
  - dans un fichier (au format spécifique),
  - sur le réseau (protocole spécifique) ou
  - dans une base de données (schéma spécifique)
- CHARGEMENT en mémoire (au travers d'une API d'entrée-sortie)
- Conversion au format adéquat
  - Ou création d'un format standard ?
- Stockage d'une STRUCTURE de données
  - Liste d'éléments, arborescence, graphe, texte structuré ...
  - Variables globales + fcts utilitaires

Architecture

- Construiriez vous une maison sans d'abord dessiner un plan ?



- Contraintes :
  - Structure de la mémoire adaptée aux données
  - Structure de la mémoire adaptée à l'affichage
    - Rapidité + Animation + Multi-écran + Matériel Hétérogène
  - Structure de la mémoire adaptée aux modifications
  - Structure de la mémoire adaptée à l'interaction