Infographie : CM1

Segment

Programme pour tracer un segment en plaçant point par point en évitant les problèmes posés par la symétrie :

```
#include <stdio.h>
#include <stdlib.h>

void segment(int x1, int y1, int x2, int y2)
{
  int dx = x2-x1;
  int dy = y2-y1;

  float a = dy/dx;
  float b = y1-a*x1;

  if (abs(dx)>=abs(dy))
  {
    for(int x=x1 ; x<=x2 ; x++)
    {
      int y = floor(a*x+b);
      point(x,y);
    }
  }

  else
  {
    a = dx/dy;
    b = x1-a*y1;

    for(int y=y1 ; y<=y2 ; y++)
    {
      int x = floor(a*y+b);
      point(x,y);
    }
  }
}
```

Cumul d'erreur : La droite n'est pas retransmise à 100% car nous travaillons avec des pixels. Comme c'est une approximation à l'arrondi, l'erreur est tantôt positive, tantôt négative. Dès que la valeur absolue de l'erreur dépasse les $\frac{1}{2}$, nous changeons de ligne (ou de colonne) de pixel.

Algorithme de Gupta & Sproul permet de mettre la même intensité lumineuse sur toutes les lignes. (Ajoute des pixels un peu plus clairs pour lisser (Antialiasing))

Cercle

Problématique ? Comment va-t-on tracer un cercle ?
En utilisant la symétrie ! Par rapport à l'axe horizontal, vertical et aux diagonales. On va avoir besoin d'une mesure d'erreur et pour ça on utilise la fonction implicite du cercle :
$x^2 + y^2 - r^2 = 0$

Programme :

```
#include <stdio.h>
#include <stdlib.h>

void cercle(int r)
```

```
{
  int x = r;
  int y = 0
  point(x,y);
  while(x != y)
  {
    int dh = (x+1)*(x+1)+(y*y)-(r*r);
    int dd = (x+1)*(x+1)+(y-1)*(y-1)-r*r;
    if (abs(dh)<=abs(dd))
    {
      x = x+1;
    }
  }
}
```

Algorithmes élémentaires

- pixel ops
- tramage
- tracé
- remplissage
    - triangles
    - contours quelconque
- découpage

Remplissage

```
triangle_simple(x0, x1, x2, y0, y1 : entier)
  calculer ymin, ymax
  calculer les couef des deux côtés est et ouest
    (ae, be, ao, bo) -> x = ae.y + be (est)
                     -> x = ao.y + bo (ouest)
  pour y de y0 à y1
    calculer xe et xo
    tracer un segment horizontal entre (xo, y) et (xe, y)
  fin
fin
```

remplissage de polygone par balayage :

- sur chaque ligne :
    - calculer les points d'intersection
    - tri en x (croissant)
    - segment horizontal entre chaque paire d'intersection (impair-pair)

Infographie : Les transformations géométriques.

1. Définition

Modifier un objet dans son espace de définition OU transporter un objet dans un autre espace de travail.

2. Transformations affines : équations idividuelles simples

2.1 Translation *(En. translation)*

- Transformation élémentaire
- Déplacer un objet.
- Transformation isométrique (pas de déformation).

```
translation(O : objet, T : vecteur)
    Pour chaque point Pi de O
        Pi <- Pi +T
    Fin Pour
Fin
```

- Matrice de translation :
- $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix}$ Dans ce sens : $\begin{pmatrix} x & y & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix}$

2.2 Rotations *(En. rotation)*

- Transformation élémentaire

2.2.1 Rotation en 2D

- Rotation d'angle θ autour de l'origine.
- Transformation isométrique.

```
rotation(O : objet)
    Pour chaque point Pi de O
        Pi.x <- Pi.x*cos(θ) - Pi.y*sin(θ)
        Pi.y <- Pi.x*sin(θ) + Pi.y*cos(θ)
    Fin Pour
Fin
```

- Matrice de rotation : $\begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

2.2.2 Rotations en 3D

- Rotation autour d'un axe.

```
rotation_autour_de_x(O : objet)
    Pour chaque point Pi de O
        Pi.x <- x
        Pi.y <- Pi.y*cos(θ) - Pi.z*sin(θ)
        Pi.z <- Pi.y*sin(θ) + Pi.z*cos(θ)
    Fin Pour
Fin
```

- En pratique nous utilisons les matrices.

2.3 Homothétie *(En. scaling)*

- Transformation élémentaire
- Changement d'échelle / changement de taille.
- Déformation.

```
scaling(O : objet, Vx : coeff, Vy : coeff, Vz : coeff)
    Pour chaque point Pi de O
        Pi.x <- x*Vx
        Pi.y <- y*Vy
        Pi.z <- z*Vz
    Fin Pour
Fin
```

- $Vx < 1 \Rightarrow$ Contraction de l'objet le long de x.
- $Vx > 1 \Rightarrow$ Elongation de l'objet le long de x.

- $Facteurd'échelle = 0 \Rightarrow$ projections
  - Exemple : Projection droite : $x' = x$, $y' = y$, $z' = z$

2.3.1 Cas particulier de la symétrie *(En. symmetry)*

- $Facteurd'échelle < 0 \Rightarrow$ symétries
  - $Vx < 0 \Rightarrow$ symétrie sur x.

2.4 Cisaillements *(En. shearing)*

2.4.1 Cisaillements 2D

- "mise en italique"
- Cisaillement horizontal :

```
shearing_x(O : objet, a : coeff)    //Transforme un | en /
    Pour chaque point Pi de O
        Pi.x <- x + a*y
        Pi.y <- y
    Fin Pour
Fin
```

- Cisaillement vertical :

```
shearing_y(O : objet, b : coeff)
    Pour chaque point Pi de O
        Pi.x <- x
        Pi.y <- b*x + y
    Fin Pour
Fin
```

- Cisaillement total :

```
shearing(O : objet, a, b : coeff)
    Pour chaque point Pi de O
        Pi.x <- x + a*y
        Pi.y <- b*x + y
    Fin Pour
Fin
```

2.4.2 Cisaillements 3D

- Cisaillement total :

```
shearing(O : objet, a, b, c, d, e, f : coeff)
    Pour chaque point Pi de O
        Pi.x <- x + a*y + b*z
        Pi.y <- c*x + y + d*z
        Pi.z <- e*x + f*y + z
    Fin Pour
Fin
```

Infographie : Courbes et surfaces.

1. Définition

- **Fonctions explicites**
  - **Courbe 2D :** $y = f(x)$ ou $y_i := t[x_i]$
  - **Surface 3D :** $z = f(x, y)$ ou $z_i := t[x_i][y_i]$

- **Fonctions implicites**
  - **1D :** $f(x) = 0$
  - **2D :** $f(x, y) = 0$
    - **Cercle :** $x^2 + y^2 - r^2 = 0$
  - **3D :** $f(x, y, z) = 0$
    - **Sphère :** $x^2 + y^2 + z^2 - r^2 = 0$

- **Fonctions paramétriques**
  - **Courbe 2D :** $\begin{cases} x = f_x(t) \\ y = f_y(t) \end{cases}$
  - **Courbe 3D :** $\begin{cases} x = f_x(t) \\ y = f_y(t) \\ z = f_z(t) \end{cases}$
  - **Surface 3D :** $\begin{cases} x = f_x(u, v) \\ y = f_y(u, v) \\ z = f_z(u, v) \end{cases}$

Fonctions polynomiales

- De **degré 1 (ligne droite) :**
  - 2 points de contrôle.
  - $\begin{cases} f_x(t) = a_x t + b_x \\ f_y(t) = a_y t + b_y \end{cases}$
  - $\begin{cases} A(x) = b_x \\ A(y) = b_y \end{cases}$
  - $\begin{cases} B(x) = a_x + b_x \\ B(y) = a_y + b_y \end{cases}$
    - $\Rightarrow \begin{cases} x = A_x + t(B_x - A_x) \\ y = A_y + t(B_y - A_y) \end{cases}$
- Avec des fonctions polynomiales de **degré 2** nous pouvons tracer des **paraboles** ("avec une bosse") :

- 3 points de contrôle.
- $\begin{cases} f_x(t) = a_x t^2 + b_x + c_x \\ f_y(t) = a_y t^2 + b_y + c_y \end{cases}$
- ...

- Avec des fonctions polynomiales de **degré 3** nous pouvons tracer des courbes avec **2 bosses** (tangente) :
  - 4 points de contrôle.
  - ...
- **Degré** n : $n-1$ bosses
  - $n+1$ points de contrôle.
- Le bon compromis : **degré** 3

Courbes d'Hermitte

- 2 points de contrôle et 2 tangentes.
- $Q(t) = \begin{array}{l} (2t^3 - 3t^2 + 1)P_1 \\ +(-2t^3 + 3t^2)P_4 \\ +(t^3 - 2t^2 + t)\overrightarrow{R_1} \\ (t^3 - t^2)\overrightarrow{R_2} \end{array}$

Courbes de Bézier

- 4 points de contrôle ($=$ 2 extrémités + 2 attracteur)
- $Q(t) = \begin{array}{l} (1-t)^3 P_1 \\ +3t(1-t)^2 P_2 \\ +3t^2(1-t)P_3 \\ +3t^3 P_4 \end{array}$
  - Somme pondérée des points de contrôle. Fonction de pondération : **polynôme de Bernstein :**
    - $B_{0,3}(t) \qquad B_{1,3}(t) \qquad B_{2,3}(t) \qquad B_{3,3}(t)$
    - $\sum B_{i,3}(t) \leq 1 \Rightarrow$ La courbe reste à l'intérieur de l'enveloppe convexe des points de contrôle.

Continuité : assemblage de segments de courbes

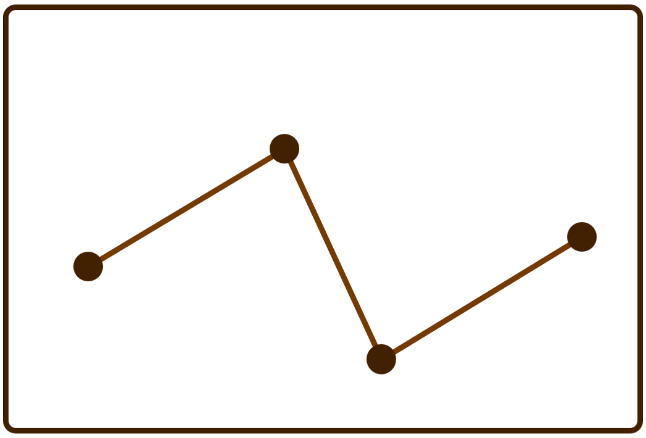
- À l'ordre $0 \rightarrow C^0 \Rightarrow 1$ point de contact. Par exemple $Q_1(1) = Q_2(0)$
- À l'ordre 1
  - Tangentes identiques $\Rightarrow C^1$
  - Tangentes parallèles $\Rightarrow G^1$

Interpolation d'une courbe passant par un ensemble de points P_i

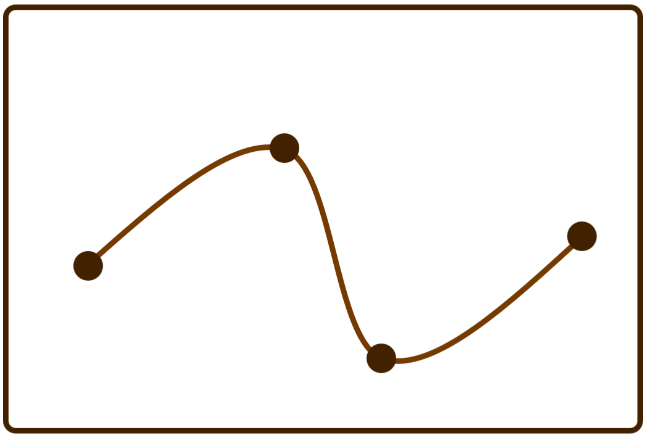
- Utilise la courbe d'Hermitte
- $Q_i(t) = h_{00}(t)P_i + h_{10}(t)\overrightarrow{R_i} + h_{20}(t)P_{i+1} + h_{30}(t)\overrightarrow{R_{i+1}}$
- Différence finie : $m_i = \dfrac{1}{2} \times (\dfrac{p_{i+1} - p_i}{x_{i+1} - x_i} + \dfrac{p_i - p_{i+1}}{x_i - x_{i+1}})$
- Définition de la courbe :
  - P_i (points de passage)
  - t_i (abscisse paramétrique **globale** des points de passage)
  - Vecteurs paramétriques de valeur croissante : $[t_0, t_1, ..., t_n], t_{i+1} \geq t_i$
    - Exemple : [0, 1, 2, ..., n] courbes lisses.
    - Calcul de la valeur paramétrique **locale** : $t_l = \dfrac{t_g - t_i}{t_{i+1} - t_i} \in [0;1]$

- Courbe cardinale :
    - $\overrightarrow{R_i} = (1-c)\dfrac{P_{i+1}-P_i}{t_{i+1}-t_i}$

$c = 1 \Rightarrow$ **tension max :**



$c = 0,5 \Rightarrow$ **tension neutre :**



$c = 0,5 \Rightarrow$ **tension min :**