

TRACÉS DE CERCLES SUR μC

Nous avons abordé le mois dernier le tracé de droites en utilisant l'algorithme de Bresenham. Ce dernier a le mérite de ne nécessiter que des calculs utilisant des nombres entiers et est donc plus rapide que ceux qui utilisent des nombres flottants. Bresenham a élaboré un algorithme similaire pour le tracé des cercles que nous allons détailler.

Considérons un cercle de centre (0, 0) et de rayon R. Nous ne nous intéresserons qu'au premier quadrant ($X \geq 0, Y \geq 0$). On peut facilement déduire les trois autres quadrants grâce aux symétries suivant les axes des X et des Y. L'algorithme de Bresenham est un algorithme itératif qui calcule les nouvelles coordonnées X_i, Y_i du nouveau point à tracer en fonction des positions du point précédent. Les observations suivantes simplifient la conception de l'algorithme : l'algorithme commence par le point $Y = R, X = 0$, Y est une fonction monotone décroissante de X.

Pour tout point x_i, y_i du cercle, le point suivant généré ne peut avoir que l'une des trois positions suivantes : immédiatement à droite aux coordonnées $x_i + 1, y_i$, diagonalement aux coordonnées $x_i + 1, y_i - 1$, immédiatement en dessous aux coordonnées $x_i, y_i - 1$.

Le point à choisir doit minimiser les distances (ou les carrés de distances) entre une de ces trois positions et le cercle réel soit :

$$mh = |(x_i + 1)^2 + (y_i)^2 - R^2|$$

$$md = |(x_i + 1)^2 + (y_i - 1)^2 - R^2|$$

$$mv = |x_i^2 + (y_i - 1)^2 - R^2|$$

Soit $\Delta_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2$, la différence entre le carré de la distance depuis le centre du cercle au pixel diagonal $(x_i + 1), (y_i - 1)$ et la distance à un point du cercle.

Si $\Delta_i < 0$, le point diagonal est à l'intérieur du cercle, aussi le point choisi doit être le point à droite (mh) ou le point diagonal (md). En effet, le point vertical (mv) serait encore plus éloigné à l'intérieur du cercle. Pour choisir le point, il faut calculer la différence entre mh et md.

$$\delta = mh - md$$

$$\delta = |(x_i + 1)^2 + (y_i)^2 - R^2|$$

$$- |(x_i + 1)^2 + (y_i - 1)^2 - R^2|$$

Si $\delta < 0$ la distance du cercle au pixel diagonal (md) est plus grande que la distance au point horizontal (mh).

Aussi, si :

$$\Delta_i < 0 \text{ et } \delta \leq 0, \text{ choisir mh}$$

$$\Delta_i < 0 \text{ et } \delta > 0, \text{ choisir md}$$

```
; TRACE DE CERCLE. Algorithme de Bresenham
; Trace le cercle de Rayon R4 et de centre X = R6 et Y = R7 dans un espace
; ou X et Y sont compris entre 0 et 255.
; Le rayon maximum du cercle est de 255. La fonction PLOT duplique le point
; calculé par CERCLE suivant des axes de symétries passant par le centre du
; cercle.
; Les quatre points ainsi calculés sont tracés par la fonction PIXEL.
; La fonction PLOT réalise du «clipping» si les points calculés sortent de
; l'espace (les points qui sortent de l'espace affichable ne sont pas affichés)
; Si l'afficheur LCD a une résolution inférieure à 256 X 256, il faut modifier
; PLOT de manière à ce que le clipping fonctionne correctement.

SEGMENT_DATA SEGMENT DATA
SEGMENT_CODE SEGMENT CODE
RSEG SEGMENT_DATA

; Utilisé par l'algorithme de tracé de cercle pour un quadrant
XI: DS 1 ; pixel dans le premier quadrant
YI: DS 1
DELTA: DS 2 ; carré de l'erreur
; Utilisé pour tracer le cercle complet
CENTREX: DS 1 ; centre du cercle
CENTREY: DS 1

MSB EQU 0
LSB EQU 1

RSEG SEGMENT_CODE
; PIXEL doit être écrit en fonction de l'afficheur graphique
; X est passé dans R6, Y dans R7.
PIXEL: RET
; Affiche les 4 points symétriques. Les axes de symétrie passent par le
; centre du cercle (CENTREX, CENTREY).
PLOT: MOV A,CENTREX ; position du centre
ADD A,XI ; translate la position X du cercle
JC PLOT2 ; dépasse le bord, sort
MOV R6,A
MOV A,CENTREY ; idem pour Y
ADD A,YI
JC PLOT3 ; n'essaie même pas le deuxième
MOV R7,A
ACALL PIXEL ; trace le premier quadrant
; deuxième quadrant -X, Y
PLOT2: CLR C
MOV A,CENTREX ; position du centre
SUBB A,XI ; translate la position X du cercle
JC PLOT4 ; dépasse le bord, essay quatrième
MOV R6,A
MOV A,CENTREY ; Y idem premier quadrant
ADD A,YI
JC PLOT3 ; dépasse essay le 3ième quadrant
MOV R7,A
ACALL PIXEL ; trace le deuxième quadrant
; troisième quadrant -X, -Y
PLOT3: CLR C
MOV A,CENTREX ; position du centre
SUBB A,XI ; translate la position X du cercle
JC PLOT4 ; dépasse le bord, sort
MOV R6,A
CLR C
MOV A,CENTREY
SUBB A,YI ; Y depasse, sort
JC PLOT4
MOV R7,A
ACALL PIXEL ; trace le troisième quadrant
; quatrième quadrant X, -Y
PLOT4: MOV A,CENTREX ; position du centre
ADD A,XI ; translate la position X du cercle
JC PLOT5 ; dépasse le bord, sort
MOV R6,A
CLR C
MOV A,CENTREY ; Y idem troisième quadrant
SUBB A,YI
JC PLOT5
MOV R7,A
ACALL PIXEL ; trace le quatrième quadrant
PLOT5: RET
; Transfère DELTA dans R6:R7
GETERR: MOV A,DELTA+MSB ; DELTA+MSB
MOV R6,A ; DELTA+MSB
MOV A,DELTA+LSB ; DELTA+LSB
MOV R7,A
RET
```

suite page suivante

Comme le point diagonal est à l'intérieur du cercle, le calcul de δ se simplifie :

$$\begin{aligned}\delta &= (x_i + 1)^2 + (y_i)^2 - R^2 \\ &- (x_i + 1)^2 + (y_i - 1)^2 - R^2 \\ \delta &= 2 [(x_i + 1)^2 + (y_i - 1)^2 - R^2] + 2 y_i - 1 \\ \delta &= 2 (\Delta_i + y_i) - 1\end{aligned}$$

Un raisonnement similaire pour le cas $\Delta_i > 0$ aboutit à un calcul de δ' suivant :

$$\delta' = 2 (\Delta_i - x_i) - 1$$

$\Delta_i > 0$ et $\delta' \leq 0$, choisir md

$\Delta_i > 0$ et $\delta' > 0$, choisir mv

$\Delta_i = 0$ choisir md

Le calcul de Δ_i peut être effectué de manière récursive pour les trois cas possibles. Si le nouveau point est le point horizontal (mh) :

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i$$

$$\Delta_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2$$

$$\Delta_{i+1} = \Delta_i + 2x_{i+1} + 1$$

De même si le nouveau point est le point diagonal :

$$\Delta_{i+1} = \Delta_i + 2x_{i+1} - 2y_{i+1} + 1$$

Si c'est le point vertical :

$$\Delta_{i+1} = \Delta_i - 2y_{i+1} + 1$$

L'algorithme de Bresenham complet pour le tracé de cercle dans le premier quadrant s'écrit :

$$x_i = 0$$

$$y_i = R$$

$$\Delta_i = 2(1-R)$$

boucle :

Plot (x_i , y_i)

si $y_i \leq 0$ va à fin

si $\Delta_i < 0$ alors va à intérieur

si $\Delta_i > 0$ alors va à extérieur

si $\Delta_i = 0$ alors va à diagonal

intérieur :

$$\delta = 2 (\Delta_i + y_i) - 1$$

si $\delta \leq 0$ alors va à horizontal

si $\delta > 0$ alors va à diagonal

extérieur :

$$\delta' = 2 (\Delta_i - x_i) - 1$$

si $\delta' \leq 0$ alors va à diagonal

si $\delta' > 0$ alors va à vertical

horizontal :

$$x_i = x_i + 1$$

$$\Delta_i = \Delta_i + 2x_{i+1}$$

va à boucle

diagonal :

$$x_i = x_i + 1$$

$$y_i = y_i - 1$$

$$\Delta_i = \Delta_i + 2x_i - 2y_i + 1$$

va à boucle

vertical :

$$y_i = y_i - 1$$

$$\Delta_i = \Delta_i - 2y_i + 1$$

va à boucle

fin :

Le listing 1 reprend l'algorithme ci-dessus traduit en assembleur 8051. La fonction CERCLE calcule les points X_i et Y_i correspondant au cercle de rayon R4 dans le premier quadrant. CERCLE appelle la fonction PLOT qui trace les points correspondants aux quatre quadrants du cercle du centre CENTREX et CENTREY. Pour faciliter les calculs, l'opposé de Δ_i est calculé à la place de Δ_i . Les signes de tests sont inversés par rapport à la description de l'algorithme. La fonction PLOT appelle la fonction PIXEL qui trace le point sur l'organe graphique choisi. Cette fonction doit être écrite par l'utilisateur. Le rayon doit être supérieur à zéro et inférieur à 256. Le listing est disponible sur le serveur ERP.

```
; R6:R7 = R6:R7 - R4:R5
SUBINT: CLR C
        MOV A,R7
        SUBB A,R5
        MOV R7,A
        MOV A,R6
        SUBB A,R4
        MOV R6,A
        RET

; R6:R7 = R6:R7 + R4:R5
ADDINT: MOV A,R7
        ADD A,R5
        MOV R7,A
        MOV A,R6
        ADDC A,R4
        MOV R6,A
        RET

; R4:R5 = R6:R7
TFRINT: MOV A,R7
        MOV R5,A
        MOV A,R6
        MOV R4,A
        RET

; R6:R7 = 2 * (R6:R7)
DBLINT: MOV A,R7
        ADD A,R7
        MOV R7,A
        MOV A,R6
        ADDC A,R6
        MOV R6,A
        RET

; Etend l'addition 8 bits sur 16 bits dans R4:R5
EXTADD: MOV R5,A ; sauve le LSB
        CLR A
        ADDC A,#0 ; étend sur 16 bits
        MOV R4,A ; sauve le MSB
        RET

; CERCLE utilise l'algorithme de Bresenham pour tracer un cercle.
; CERCLE ne trace que les points du premier quadrant. Le centre du cercle est
; X = 0, Y = 0. CERCLE est initialisé avec Y = Rayon, X = 0 et s'arrête quant
; le premier quadrant est tracé, c'est à dire quant Y = 0. Le rayon maximum
; est de 255.
; CERCLE appelle la fonction PLOT qui trace les quatre points symétriques pour
; avoir un cercle complet. PLOT utilise les paramètres X1 et Y1 qui sont la
; position calculée de l'arc de cercle pour le premier quadrant.
; CERCLE modifie A, R4, R5, R6, R7. (R0, R1, R2, R3 inchangés)
CERCLE: MOV Y1,R4 ; Y1 = Rayon
        DEC R4 ; R - 1
        MOV A,R4
        ADD A,R4 ; 2 * (R - 1)
        MOV DELTA+LSB,A
        CLR A
        MOV X1,A ; X1 = 0
        ADDC A,#0
        MOV DELTA+MSB,A
CER1: ACALL ACALL ; trace X1, Y1 et les points symétriques
        MOV A,Y1 ; si Y1 = 0, fin
        JZ CER50
        MOV A,DELTA+MSB
        ORL A,DELTA+LSB
        JZ CER20
        ACALL GETERR ; R6:R7 = DELTA
        ACALL DBLINT ; 2*DELTA, retour dans A le MSB
        JB ACC.7,CER3 ; teste le signe de 2*DELTA
; signe > 0, calcule 2*DELTA - (2*Y1 - 1)
CER2: MOV A,Y1 ; Y1
        DEC A ; Y1 - 1
        ADD A,Y1 ; 2*Y1 - 1 (Y1 toujours >= 1)
        ACALL EXTADD ; étend sur 16 bits dans R4:R5
        ACALL SUBINT ; 2*DELTA - (2*Y1 - 1)
        JNB ACC.7,CER10 ; >= 0
        AJMP CER20 ; < 0
; signe < 0, calcule 2*DELTA + 2*X1 + 1
CER3: MOV A,X1 ; 2*X1
        ADD A,X1 ; 2*X1 + 1 pas de retenue (2*X1 pair)
        INC A ; étend sur 16 bits dans R4:R5
        ACALL EXTADD ; 2*DELTA + 2*X1 + 1
        JNB ACC.7,CER20 ; >= 0
        AJMP CER30 ; < 0
; calcule le nouveau pixel
CER10: INC X1 ; DEPLACEMENT HORIZONTAL
        MOV A,X1 ; calcule Di = Di - (2*X1 + 1)
        ADD A,X1 ; 2*X1
        INC A ; 2*X1 + 1
        ACALL EXTADD ; étend sur 16 bits dans R4:R5
        AJMP CER40 ; va soustraire de DELTA
CER20: INC X1 ; DEPLACEMENT EN BIAIS
        DEC Y1 ; X1 - Y1
        CLR C ; sauve le LSB dans le MSB
        MOV A,X1
        SUBB A,Y1
        MOV R6,A
        CLR A
        SUBB A,#0 ; sauve le MSB et récupère le LSB
        XCH A,R6 ; X1 - Y1 + 1
        ADD A,#1 ; étend le résultat sur 16 bits
        MOV R7,A
        ADDC A,#0
        MOV R6,A ; dans R6:R7
        ACALL DBLINT ; 2* (X1 - Y1 + 1)
        ACALL TFRINT ; dans R4:R5
        AJMP CER40 ; va soustraire de DELTA
; DEPLACEMENT VERTICAL
CER30: DEC Y1 ; si Y1 = 0...
        JZ CER31 ; Y1 - 1 positif ou nul
        DEC A ; 2*Y1 - 1 > 0
        ACALL EXTADD ; étend sur 16 bits dans R4:R5
        ACALL GETERR ; prend DELTA
        ACALL ADDINT ; additionne a DELTA
        AJMP CER41 ; et transfère
CER31: MOV R4,A ; R4:R5 = 1
        INC A
        MOV R5,A
; recalcule la nouvelle DELTA
CER40: ACALL GETERR ; R6:R7 = DELTA
        ACALL SUBINT ; DELTA = DELTA - R4:R5
        MOV DELTA+MSB,A
CER41: MOV A,R7
        MOV DELTA+LSB,A
        AJMP CER1 ; boucle jusqu'à Y1 = 0 (et R1 = RAYON)
CER50: RET
END
```

■ Listing 1.