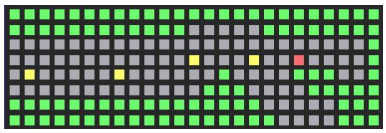
https://clx.asso.fr/spip/?Le-concours-du-reporter

Concours de programmation

Le concours du reporter

- L'Association - Nos actions - 2003 : Concours de programmation -



Date de mise en ligne : mercredi 15 janvier 2003

Pour les vacances 1995, période propice à la méditation, la réflexion et le jeu, le Reporter lançait un concours de programmation auprÚs de ses lecteurs.

Cette période avait été choisie spécialement en pensant aux étudiants qui auraient ainsi plus de temps à y consacrer qu'en période scolaire (s'ils n'ont pas autre chose a penser bien sur :-).

Aujourd'hui, CLX se propose de faire revivre ce concours sous Linux, en adaptant les rÚgles.

Introduction

L'objet du concours en lui-même est très simple. Il va s'agir de créer un pilote logiciel de formule 1.

Ce concours se veut rester dans l'esprit qui anime le REPORTER depuis sa création, à savoir rester accessible à tous quelque soit son niveau en programmation et être un moyen de partager une passion via le plaisir du "jeu" et non pas pour l'appât du gain.

Ce concours n'est donc pas primé selon le concept traditionnel Faire les choses comme tout le monde n'est pas notre sport favori :-). Ce qui ne veut pas dire que les participants n'y trouveront pas leur bonheur.

Outre le plaisir de participer que procure un jeu collectif, les 50 premiers d'entre-vous qui nous auront fait parvenir le résultat de leurs cogitations avant la clôture du concours recevront tous un cadeau.

Bien entendu cela ne concerne que les 50 premiers qui nous auront envoyé un programme parfaitement opérationnel, c'est-à-dire capable de traiter dans le respect des règles et sans planter au moins 70% des circuits qu'il aura à parcourir. Voilà qui va décevoir les astucieux qui espèraient envoyer un p'tit programme se contentant de s'exécuter rien que pour obtenir un cadeau :-)

Ici seul le mérite compte, il n'est pas nécessaire d'être vainqueur pour avoir droit à son cadeau.

A l'époque, celui-ci se présentait sous la forme d'un abonnement gratuit d'un an (4 numéros) au Reporter.

Les participants auraient reçu donc dans leur boite à lettre dès sa parution et en priorité le REPORTER sur disquette en version Dos et Windows. Et comme il restait de la place sur ces supports, Le reporter n'aurait pas manqué de la combler avec quelques petits bonus.

Aujourd'hui, nous proposons des cadeaux de valeur comparable, c'est à dire un abonnement gratuit à la liste de diffusion du CLX, un accès privilégié au site WEB de CLX ave la possibilité de publier des articles, et peut-être d'autres bonus. Si d'aimables mécènes souhaitent apporter leur contribution à la renaissance de ce concours en faisant bénéficier ces mêmes 50 premiers candidats de petits cadeaux, ils seront les bienvenus.

Nous ne manquerons pas de les signaler sur le site.

Dernière petite chose, tous les mois précédant la clôture du concours, nous ferons paraître les résultats des programmes reçus obtenus sur un circuit de référence (avec le nom de leur auteur bien sûr).

Il est temps maintenant que vous preniez connaissance des règles du jeu, mais n'oubliez pas que si *l'important est de participer* (Pierre de Coubertin), *l'essentiel est de s'amuser* (Le Reporter).

Présentation

A la base, il s'agit d'un concours de programmation : chaque participant doit écrire un programme qui permet de piloter au plus vite une formule 1 sur un circuit quelconque dont il ne connait pas a priori le parcours.

Il faut donc écrire un programme de pilote automatique de formule 1.

En fait ce concours s'apparente à celui organisé à sa grande époque, par la revue Micro-Système. Ce concours avait passionné une bonne partie de la France, mais limitait fortement la participation car il fallait réaliser un vrai modèle réduit de voiture.

La différence c'est qu'ici le concours est uniquement logiciel et chacun peut facilement participer : il suffit de posséder un ordinateur compatible PC. Peu de moyens sont donc nécessaires (c'est ouvert à tous ou presque) et seule l'astuce du programmeur lui permettra de faire la différence.

Pour cela, des règles très simples définissent le comportement 'physique' de la voiture sur la piste. Le programme doit respecter ces règles, tout en suivant le tracé de la piste qui constitue le circuit, et joindre au plus vite la ligne d'arrivée.

L'originalité de ce concours réside dans le fait que même les organisateurs du concours (nous-même) ignorent sur quelles pistes se dérouleront les courses !

En effet chaque participant doit aussi fournir un fichier 'circuit' contenant une piste dont le tracé est laissé à sa discrétion, (et qui a priori favorisera son propre pilote).

Chacun des participants fera tourner sa voiture sur son propre circuit, mais aussi sur les circuits de ses adversaires.

Bref pour gagner, il faut jouer sur les deux tableaux :

[-] 1) Développer un pilote logiciel qui puisse tourner vite sur n'importe quel circuit sans en connaitre a priori le parcours.
[-] 2) Décrire un circuit sélectif, qui tendra à pieger et ralentir au maximum les pilotes adversaires par sa complexité tout en favorisant le sien.

Bien entendu il n'est pas question que vous perdiez du temps à créer un circuit ou un programme de chronométrage.

Sont donc fournis gratuitement avec ce numéro un programme vous permettant de dessiner graphiquement un circuit conforme ainsi qu'un programme de chronométrage et de validation qui utilise le fichier des déplacements généré par votre pilote logiciel pour évaluer ses performances.

L'éditeur de circuits originel nécessite une carte vga, mais n'est pas indispensable pour créer un circuit. Un simple éditeur de texte suffit comme expliqué plus loin, ce afin de rendre le concours accessible indépendamment du matériel PC utilisé. Par contre, le programme tourne sous DOS.

De même, ce n'est pas parce que l'on parle de 'tourner vite' ou de 'chronométrage' qu'il s'agit d'un concours favorisant un langage rapide (ASM par exemple). Comme vous le comprendrez en lisant cet article en détail, les temps en question sont des temps relatifs.

Plus que le langage et sa rapidité, c'est l'astuce du programmeur qui lui permettra de faire la différence. Tout le monde a ses chances dans ce concours, que ce soit un habitué du C ou du basic.

Règles d'évolution de la F1

Les règles qui régissent le déplacement d'une voiture sur la piste d'un circuit quelconque sont assez simples.

Elles sont basées sur un petit jeu qui se pratiquait (et se pratique sûrement encore !) alors que nous usions nos fonds de culottes sur les bancs du lycée...

Imaginez une feuille de papier quadrillée sur laquelle on a délimité une piste à l'aide de deux tracés continus formant un circuit bouclé sur lui-même (vous voyez, ce n'est que du matériel classique pour les cancres du fond de la classe : juste des stylos et du papier !).

Ça se jouait à deux (voire plus), et chacun utilisait un stylo-bille de couleur différente, qui correspondait à la couleur de sa voiture fétiche.

La position de la voiture - qui devait bien sûr toujours rester sur la piste - était indiquée par un petit

point, à l'intersection d'une ligne horizontale et d'une ligne verticale du quadrillage de la feuille.

Chacun, à tour de rôle, déplaçait sa voiture, en respectant la règle simple récurrente suivante.

La nouvelle position de la voiture s'obtenait en reportant le déplacement précédent de la voiture, et en choisissant soit le point obtenu, soit l'un de ses 8 voisins immédiats.

GIF - 1.5Â ko
Dans l'exemple représenté ci dessus, la voiture, dont le déplacement précédent était de un point vers le haut et de 4 vers la droite, peut maintenant choisir l'un des 9 points indiqué par ' ?' comme nouvelle position.

On constate que par ce simple mécanisme, on pouvait accélérer (augmenter le déplacement d'un point), freiner (diminuer d'un point) ou rester à vitesse constante indépendamment dans les deux directions.

On obtenait ainsi un comportement similaire à celui d'une voiture (essayez et vous verrez qu'il ne faut pas prendre des virages tendus à vitesse trop élevée !).

Bien entendu, il fallait aussi s'évertuer à éviter les collisions, soit avec les voitures adverses, soit avec la bord de la piste.

Les différences entre ce jeu scolaire et le concours sont peu nombreuses :

[-] Pour des raisons de simplification, la position de la voiture est au centre d'un carré (appelons-le "pixel", même si ce n'est pas une dénomination parfaite). Les bords de piste ne sont pas une ligne continue, mais sont aussi définis par des pixels carrés.

Voir ci-dessous un 'zoom' sur une portion de circuit :

JPEG - 3.8Â ko
[-] Le programme que vous avez à écrire (le pilote logiciel) doit décrire l'évolution d'une seule voiture sur la piste.

En fait, il s'agit plutôt d'essais de qualification que de la course en elle même. Chaque nouveau déplacement de la voiture correspond à un temps factice d'une seconde. La taille d'un 'pixel' correspond de façon arbitraire à celle d'un carré de 10 mètres par 10 mètres. Ces conventions permettent d'obtenir les vitesses de notre voiture virtuelle, exprimées en km/h, de façon assez réalistes.

Le chronomètrage de votre voiture correspond donc à un chiffre tout à fait identique quelque soit l'ordinateur, car il correspond au nombre de coups nécessaires pour boucler un circuit donné.

Le but, donc, c'est d'écrire un programme qui définit les positions successives de la voiture, en respectant les règles des déplacements, et suivant un tracé donné, afin de boucler un tour en moins de coups que les autres programmes.

Pour ce faire, le circuit est représenté dans un fichier à l'aide de caractères Ascii disposés dans une matrice rectangulaire de 100 lignes par 150 colonnes. Un caractère espace représente la piste tandis que tout autre représente le décor. Les règles définissant le format de ce fichier sont décrites plus loin.

Le point de départ est fixé (c'est le point de coordonnées x=40 y=4), ainsi que le sens dans lequel il faut faire le tour du circuit (Départ vers la droite, soit un sens qui suit globalement celui des aiguilles d'une montre).

La vitesse au départ est nulle, ce qui signifie que le point d'arrivé du premier déplacement est un des 8 voisins du point de départ (ou plus exactement un des trois placés à sa droite, sinon le départ se ferait en marche arrière :-)

Le but du programme est donc d'écrire un fichier résultat donnant les différentes positions de la voiture calculées pour un circuit donné jusqu'à la ligne d'arrivée (de même coordonnées que le point de départ).

Il est, évidemment, interdit d'entrer en collision avec les bords de la piste, faute de quoi votre formule 1 sera considérée comme détruite et votre tour ne sera pas pris en compte ! (Ce n'est pas très solide ces bolides...)

Le fichier résultat est ensuite passé en paramètre au programme TESTF1.EXE qui se charge de vérifier que le circuit est bien bouclé, qu'il n'y a pas collisions, que les règles de déplacement sont respectées, et en final d'afficher les temps.

Il convient de préciser en détail la manière dont s'effectue les tests de collisions [1], qui vous empêchent de 'mordre' sur les bas cotés pour couper un virage.

La règle est simple : on trace la ligne entre le point de départ et d'arrivée de la voiture, et si l'un des points intermédiaires n'est pas sur la piste, il y a collision !

Mais, me diront à juste titre certains d'entre vous, cette règle apparemment simple peut poser un problème, très connu par ceux qui ont déjà écrit des routines de tracé de lignes...

En effet certains cas particuliers peuvent devenir des cas litigieux, comme illustré par l'exemple suivant :
JPEG - 828Â octets
Le déplacement de la voiture est de 5 pixels vers la droite et de 1 pixels vers le haut (joignant les deux points jaunes. Pour les deux premiers et les deux derniers pixels, il n'y a pas de problème : si l'un des points gris n'est pas sur la piste, il y a collision.

Par contre pour un décalage de 5 pixels vers la droite, la voiture passe juste au milieu de deux pixels : ceux représentés en rouge. Il n'a pas de raison de privilégier a priori plus une solution que l'autre ! (alors que les routines de tracé de lignes feront un choix ou l'autre selon la manière dont elles sont programmées)

Nous avons donc décidé d'utiliser une routine de test de collision souple, à savoir de déclarer qu'il n'y a pas collision si l'un OU l'autre des deux pixels reste sur la piste.

Bien entendu si les deux pixels (l'un ET l'autre) sont hors de la piste, il y a collision.

Ainsi, quel que soit la routine de 'tracé' que vous utiliserez, le déplacement sera conforme, évitant toute ambiguïté.

Pour éviter toute discussion concernant la routine de *test de collision* un exemple de cet algorithme est donné, en langage Pascal (adaptation facile à tout autre langage).

Pour bien comprendre tout ce mécanisme très simple (bien que difficile à expliquer), vous trouverez ci-après quelques exemples illustrant les principes d'évolution.

Notez qu'il s'agit à chaque fois de portions de circuits.

Dans les exemples présentés, les positions précédentes de la voiture sont représentées en jaune

JPEG - 9.3Â ko
Pas de problème pour la voiture, elle contrôle bien sa vitesse, ce qui lui permet d'aborder le virage en toute quiétude...

JPEG - 25.7Â ko
Sur cet exemple, la voiture va trop vite.. Bien que restant toujours sur la piste, la trajectoire d'un point à l'autre coupe le bord, amenant la routine de test de collision à constater un accident.

JPEG - 9.2Â ko
Sur cet exemple la voiture ne va pas trop vite.. Elle reste
toujours sur la piste, et sur le point litigieux clignotant vert, la routine de test de collision, de par sa souplesse,
accepte la trajectoire, considérant que la voiture est passée
juste à gauche.

JPEG - 27Â ko
Dans ce cas, le programme déclarera la position de la voiture comme illicite. En effet, le dernier déplacement était de trois pixels vers la droite et la voiture se 'permet' brusquement un
déplacement que de un pixel.. La position permise de la voiture est l'un des points voisins du point blanc clignotant, ce qui signifie que le crash est inévitable...
JPEG - 22.7Â ko
La voiture va trop vite, mais heureusement une impasse lui permet de décélérer.. Il ne lui reste plus qu'à freiner à fond (jusqu'à un déplacement nul en X) pour reprendre le bon chemin, dans
l'autre sens.

Règles de description du circuit

Le fichier de description du circuit, d'extension CIR par convention, a pour but de délimiter la piste sur laquelle doit évoluer la voiture par rapport au reste du dessin.

Par convention, le circuit est compris dans un rectangle de longueur 150 et de hauteur 100 (limites extérieures).

Le fichier est donc simplement constitué d'un ensemble de 100 lignes de chaînes de caractères ASCII. Chaque chaîne de caractères est constituée de 150 caractères utiles, plus les classiques "carriage return" et "line feed" éventuels qui séparent les lignes.

On pourra donc relire aisément ce fichier et mettre les données dans un tableau de caractères de type (par exemple)

```
.
 donnee: array[0..149,0..99]
 0   1   2   3       146 147 148 149
 +--------------     -----------------+
 0  ¦   ¦   ¦   ¦        ¦   ¦   ¦   ¦   ¦
 +---+---+---+--     -+---+---+---+---¦
 1  ¦   ¦   ¦   ¦        ¦   ¦   ¦   ¦   ¦

 +---+---+---+--     -+---+---+---+---¦
 99 ¦   ¦   ¦   ¦        ¦   ¦   ¦   ¦   ¦
 +--------------     -----------------+
```

La première valeur du tableau, comprise entre 0 et 149, comporte les coordonnées en X (longueur) du dessin, avec la valeur 0 représentée classiquement à gauche sur l'écran.

La deuxième valeur du tableau, comprise entre 0 et 99, comporte les coordonnées en Y (hauteur) du dessin, avec la valeur 0 représentée classiquement en haut sur l'écran et correspondant à la première ligne du fichier CIR.

Pour distinguer la piste du reste du circuit, une convention simple : un point de la piste est indiqué dans le fichier par le caractère ' ' (espace, code ASCII 32).

Tout autre caractère ASCII doit être considéré par le programme pilote comme ne faisant pas partie de la piste, donc inaccessible à la voiture (rappel : une voiture hors piste provoque un accident qui l'élimine).

Néanmoins, pour permettre une représentation plus avenante du circuit, j'ai décidé d'utiliser une convention faisant correspondre une couleur à certains caractères ASCII.
Cette convention utilise les couleurs typiques de Windows, plutôt que celles de la carte VGA, pour permettre une conversion facile à partir d'un fichier "BMP" non compressé à 16 couleurs.

Cette convention est la suivante :

caractère	couleur
A	Noir
B	Rouge foncé
C	Vert foncé
D	Jaune foncé

E	Bleu foncé
F	Magenta foncé
G	Cyan foncé
H	Gris clair
J	Rouge vif
K	Vert vif
L	Jaune vif
M	Bleu vif
N	Magenta vif
O	Cyan vif
P	Blanc

Il faut noter que la couleur gris foncé est attribuée par convention au caractère espace (la piste) et non pas à la lettre 'I'.

Tous les autres caractères ASCII sont dessinés, dans les outils fournis, avec la couleur 'Vert vif'.

Rappelons cependant que ces codes de couleurs n'existent que pour faire joli, et qu'un circuit peut très bien être défini uniquement à l'aide de 'X' et ' ' via un éditeur de textes !

Mais rassurez-vous, vous pouvez complètement ignorer et oublier ces conventions de couleurs : un programme de dessin, CREECIR, est fourni ! Il est documenté plus loin, et va vous aider à créer votre circuit, ainsi que le colorier à votre (bon) goût.

Pour qu'un circuit soit considéré comme conforme, il faut qu'il vérifie les différentes règles suivantes :

[-] **(1)** La surface totale occupée par le circuit (piste et décors) doit être de taille 150*100, à savoir 100 lignes de 150 caractères (rappel)

[-] **(2)** La piste ne peut pas affleurer les bords du dessin. Plus précisément, on ne pas mettre de caractère ' ' en ligne 0 ou 99, ou en colonne 0 et 149. Le but de cette règle est de délimiter facilement le terrain sans avoir à y rajouter un bord fictif (et ainsi faciliter les tests !).

[-] **(3)** Une zone de piste est imposée : c'est le rectangle délimité par les points x=20, y=2 et x=59, y=6. C'est au milieu de cette zone que se situe le point de départ (x=40, y=4).

Après avoir bouclé un tour complet, en partant vers la droite (sens x croissant), la ligne d'arrivée est une ligne verticale, de la largeur de la piste, et située en x=40.

Le but de cette règle est, bien entendu, de fixer précisément les points de départ et la ligne d'arrivée, ainsi qu'une largeur et longueur de piste suffisante pour bien démarrer et passer la la ligne d'arrivée en accélération.

Le sens de rotation est fixé (globalement celui des aiguilles d'une montre), car la voiture doit se déplacer au départ dans le sens des x croissant (droite). Elle coupera la ligne d'arrivée lorsqu'elle aura atteint ou dépassé la coordonnée x=40, en venant bien entendu d'une position x plus faible (pas question de couper la ligne d'arrivée dans le mauvais sens !!!) :

JPEG - 6.2Â ko La vitesse de départ est nulle, ce qui signifie, en pratique, que le premier point après la position (40,4) est l'un des voisins du point de départ.

[-] **(4)** Un seul chemin doit mener du départ vers l'arrivée. Le circuit doit donc être évidemment bouclé sur lui-même, mais il ne peut pas comporter d'ilôts au milieu de la piste. Par ilôt, j'entend un ou plusieurs caractères au beau milieu de la piste.

Par exemple ce genre de configurationest interdit :

JPEG - 2.4Â ko
Attention, ceci ne vous interdit pas certains 'pièges' amusants dans votre circuit. Par exemple la création d'impasses, ayant pour but de tromper les voitures adverses est permise.

JPEG - 2.2Â ko
[-] **(5)** La largeur de la piste est au minimum de 2. Le but de ce concours est d'écrire un pilote de Formule 1, et pas d'auto cross ! Cette règle peut s'énoncer plus précisément comme suit :
aucun point de la piste ne peut avoir pour "voisin" (i.e. une des 8 cases qui l'entoure) à la fois un bord de piste intérieur et un bord de piste extérieur. Il faut donc toujours au minimum deux caractères de piste (pixels) pour séparer les deux bords, y compris en diagonale.

Dans tous les cas de figure, si vous avez un doute sur la conformité de votre piste au règlement, le programme CREECIR est là pour vous aider en effectuant toutes les vérifications sur votre fichier CIR, et en indiquant, si nécessaire, pour quelle(s) raison(s) un circuit donné n'est pas conforme.

Post-scriptum :

Notez-bien qu'actuellement CLX n'a pas redéveloppé sous Linux, les outils initiaux, fournis par le Reporter et programmés par Luc RiviÚre. Par contre, si vous avez installé FreeDOS (un DOS sous GPL) sur votre machine ou Dosemu (un émulateur DOS sous linux), vous pourrez les utiliser.

Nous allons réécrire des versions minimales de ces outils sous linux et les mettre sous licence GPL. Nous les publieront ici dans un avenir plus ou moins proche (à moins que vous ne vous sentiez le courage de le faire vous-même et d'en faire profiter la communauté entiÚre, bien sûr).

Notez également que la participation au présent concours implique que votre pilote soit publié sur le site de CLX à l'issue du concours sous la license libre de votre choix.

Vous trouverez tous les numéros du Reporter sur le site de l'un des ancien R.O.R.

[1] Ci-joint un exemple en Pascal de l'algorithme de test de collision.

C'est une fonction *surPiste*, qui retourne vraie (TRUE) si la voiture reste sur la piste et faux (FALSE) lorsqu'il y a une collision dans le déplacement.

On suppose au départ que le joueur a placé son circuit dans un tableau donnee[x,y] et que le test donnee[x,y]=piste consiste à savoir si le point concerné est sur la piste.

Facile à adapter si on utilise une autre structure de données.

```
function surPiste(x,y,xf,yf:integer):boolean;
 { Teste si une ligne joignant x,y à xf,yf est sur le circuit.
 algorithme de Bresenham légèrement modifié pour les cas litigieux.
 retourne vrai (true) si on reste sur la piste,
 retourne faux (false) si on casse la voiture sur un bord.
 L. Rivière }
 var dx,dy,total,xi,yi:integer; ok:boolean;
 begin
 surPiste:=false;

 { Commençons par éliminer les cas triviaux }
 if donnee[x,y]<>piste then exit;
 { Retourne FAUX si on est pas déjà sur la piste! }
 if (x=xf) and (y=yf) then begin
 { Si on est sur la piste et que l'on bouge pas.. }
 surPiste:=true; exit;
 end;
 { Le cas général.. }
 ok:=true;
 { On commence, comme dans la plupart des algo de tracé, à se
 mettre dans le premier quadrant, pour raisonner avec des
 déplacement relatifs positifs }
 { D'abord en X.. dx toujours positif, xi prend le signe (±1) }
 if (x
 xi:=1; dx:=xf-x;
 end else begin
 xi:=-1; dx:=x-xf;
 end;
 { Ensuite en Y.. dy toujours positif, yi prend le signe (±1) }
 if (y
 yi:=1; dy:=yf-y;
 end else begin
 yi:=-1; dy:=y-yf;
 end;
 { on se déplace toujours d'un pixel à la fois selon la direction qui
 nécessite le plus de pixel. C'est pourquoi on fait les tests pour
 savoir quel déplacement est le plus grand.. }
 if (dx>dy) then begin
 { Comme le déplacement en X est plus grand, on va y aller de x à xf
 avec l'incrémentant d'un xi à chaque fois.
 Le reste c'est quasi du Bresenham classique, à savoir
 que l'on incrémente un compteur (ici total) d'une valeur égale
 à deux fois le déplacement y. }
 total:=0;
 repeat
 x:=x+xi;
 total:=total+2*dy;
 if (total=dx) then ok:=(donnee[x,y]=piste) else ok:=false;
 { Le cas litigieux signalé pour le test de collision se repère
 au fait que le compteur tombe pile sur la valeur de dx.
 Dans ce cas on met une valeur booléenne (ok) à jour selon le
 fait que l'on soit sur la piste ou non, ceci avant la modification
 de Y. Si on est pas sur le cas litigieux, on met OK à false
 (comme si on était pas sur la piste), car par la suite on fera un
 OU logique... }
 if (total>=dx) then begin
 y:=y+yi; total:=total-2*dx;
 end;
 { Comme tout Bresenham, lorsque l'on atteint ou dépasse dx, on
 incrémente la valeur d'un cran }
 ok:=(ok or (donnee[x,y]=piste));
 { Toute l'astuce du ok précédent. Si on était dans un cas litigieux
 on fait un 'ou' entre avant et après la décrémentation Y, donc
 on est sur la piste lorsque l'un ou l'autre des pixels l'est..
 Sinon (cas non litigieux) on teste bien le seul pixel concerné
 car on fait un 'ou' avec une valeur initialement false }
 until (x=xf) or not(ok);
 { On arrête quand on est plus sur la piste ou que l'on a atteint
 l'objectif x=xf }
 end else begin
 { Idem mais avec un déplacement de base suivant Y }
 total:=0;
 repeat
 y:=y+yi;
```

```
total:=total+2*dx;
if (total=dy) then ok:=(donnee[x,y]=piste) else ok:=false;
if (total>=dy) then begin
x:=x+xi; total:=total-2*dy;
end;
ok:=(ok or (donnee[x,y]=piste));
until(y=yf) or not(ok);
end;
surPiste:=ok;
end;
```