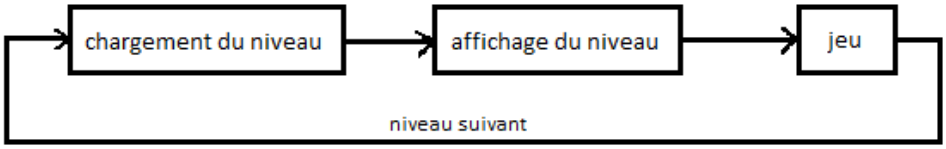
Un jeu de mini golf

Bonjour, aujourd'hui nous allons réaliser ensemble un jeu de mini golf. Même pour un programmeur expérimenté ce n'est pas évident, mais nous allons montrer qu'avec un peu d'organisation et de patience vous pouvez parvenir à de très beaux résultats. Dans cet exemple nous avons volontairement allégé le code pour simplifier son explication, mais libre à vous de rajouter des détails par la suite pour vous approprier le jeu.

Structure du Programme

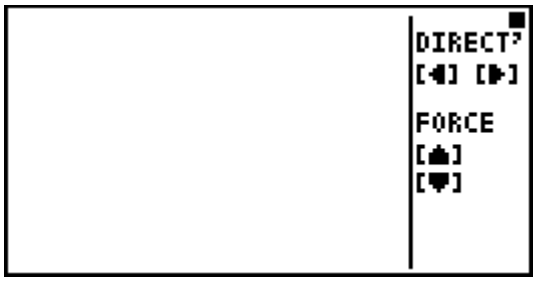
Voilà la structure principale que nous vous proposons pour ce jeu. C'est une structure très classique des jeux possédant plusieurs niveaux.



Nous allons donc voir dans un premier temps les graphismes et la gestion des niveaux, puis dans un second temps le moteur de jeu.

Mise en place des graphismes

Nous avons choisi de diviser l'écran en 2 zones : la première pour le jeu en lui-même et l'autre pour les commandes (il y en aura 2 nous les verrons dans la prochaine partie).

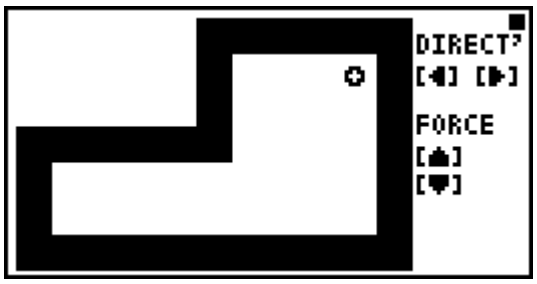


Voici le code que nous pouvons utiliser pour cela :

```
ViewWindow 1,127,0,1,63,0          // ne pas oublier d'initialiser la WiewWindow
Text 5,101,"DIRECT'"
Text 13,101,"[◀][▶]"
Text 25,101,"FORCE"
Text 33,101,"[▲]"
Text 40,101,"[▼]"
F-Line 99,1,99,63
StoPict 1                          // on enregistre cette image pour la réutiliser ensuite
```

Pour simplifier l'affichage des niveaux, la zone de jeu peut être quadrillée par des carrés de 9x9 pixels. On a donc un quadrillage de 11x7 carrés. A chaque carré on va associer un décor au choix : rien, un mur, le départ, l'arrivée.

Pour illustrer nos propos nous allons utiliser le niveau suivant :



Une des méthodes les plus simples pour gérer de tels niveaux consiste à utiliser les matrices. On code donc le niveau dans une matrice avec le code suivant :

- 0 = rien
- 1 = départ
- 2 = arrivé
- 3 = mur

Puis on va utiliser deux boucles pour parcourir la matrice et tracer le décor. Voilà le code pour charger et afficher un niveau.

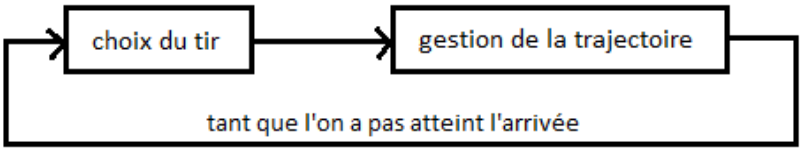
```
1→Z                 // Z indique le niveau en cours, on l'initialise au début du programme

If Z=1              // niveau 1
  Then [[3,3,3,3,3,3,3,3,3,3,3][3,1,0,0,0,0,0,0,0,0,3][3,0,0,0,0,0,0,0,0,0,3][3,3,3,3,3,3,0,0,0,0,3]
[0,0,0,0,0,3,0,0,0,0,3] [0,0,0,0,0,3,0,0,0,2,3][0,0,0,0,0,3,3,3,3,3,3]]→Mat A
IfEnd
If Z=2              // niveau 2
  Then              // à vous d'inventer vos propres niveaux
IfEnd

Cls                 // chargement des décors
RclPict 1           // on réaffiche l'image de fond
For 1→I To 11               // on parcourt toute la matrice
  For 1→J To 7
    If Mat A[J,I]=1  // on en profite pour enregistrer les coordonnées du départ et de l'arrivée
      Then I→A:J→B
    IfEnd
    If Mat A[J,I]=2
      Then I→C:J→D
      Circle 9(I-1)+4,9(J-1)+4,2            // on trace un cercle pour l'arrivée
    IfEnd
    If Mat A[J,I]=3         // on trace un carré noir si c'est un mur
      Then For 1→K To 9
        F-Line 9(I-1)+K,9(J-1)+1,9(I-1)+K,9J
      Next
    IfEnd
Next:Next
StoPict 2           // on enregistre le décor
```

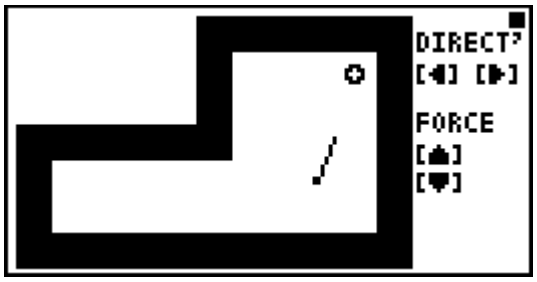
Le jeu

Le jeu peut se décomposer en deux étapes que l'on répète tant que l'on n'a pas atteint l'arrivée : la frappe de la balle, puis la gestion de la trajectoire.



Interface pour choisir la direction et la force du tir

On va utiliser ici un programme similaire à celui du cours sur le Getkey. Les commandes seront ◀ et ▶ pour gérer la direction et ▲ et ▼ pour gérer la force du tir.




```
9(A-1)+4→E    // on initialise les paramètres tels que la position de la balle, l'angle et la vitesse
9(B-1)+4→F
2→V:0→W                    // V correspond à la vitesse et W à l'angle
Do
  Cls
  RclPict 2                // on réaffiche le décor
  SketchThickF-Line E,F,E,F
  F-Line E,F,E+3Vcos W,F+3Vsin W   // on affiche la ligne indiquant la direction et la force du tir
  Do
    Getkey→G
  LpWhile G=0
  If G=28 And V<4          // on augmente la force
    Then V+1→V
  IfEnd
  If G=37 And V>1          // on diminue la force
    Then V-1→V
  IfEnd
  If G=38          // on augmente l'angle
    Then W+10→W
  IfEnd
  If G=27          // on diminue l'angle
    Then W-10→W
  IfEnd
LpWhile G ≠31
```

Trajectoire de la balle

La trajectoire est rectiligne ralentie. Il suffit donc d'avancer la balle dans la direction (cos W, sin W) avec des écarts de plus en plus faibles. Comme ceci par exemple :

```
0→G
cos W→M          // on enregistre la direction selon laquelle avance la balle
sin W→N
Do               // à chaque itération de cette boucle on va faire avancer la balle
  E+VM→E
  F+VN→F
  SketchThickF-Line E,F,E,F          // on affiche la nouvelle position de la balle
  G+1→G
  If G=2V        // cette partie permet de gérer la vitesse de la balle
    Then V-1→V   // c'est une bidouille pour rendre la trajectoire plus ou moins réaliste
    0→G
  IfEnd
  For 0→H To 200:Next                // petite pause pour bien voir la balle avancer
LpWhile V>0                  // on boucle tant que la balle avance
```

Gérer les collisions et les rebonds

La dernière étape, qui n'est pas la plus simple, consiste à gérer les collisions avec les murs. Détecter les murs revient à connaitre la position de la balle dans la matrice dont on avait parlé plus haut. Pour cela on utilise une simple division.

Ensuite pour gérer les rebonds, on va inverser « l'avancée » de la balle selon l'axe x si la collision se fait par la droite ou la gauche, et on va inverser « l'avancée » de la balle selon l'axe y si la collision se fait par le haut ou le bas. Voilà ce que cela peut donner :

```
0→G
cos W→M
sin W→N
Do               // à chaque itération de cette boucle on va faire avancer la balle
  E+VM→E
  F+VN→F
  SketchThickF-Line E,F,E,F          // on affiche la nouvelle position de la balle
  G+1→G
  If G=2V        // cette partie permet de gérer la vitesse de la balle
    Then V-1→V
    0→G
  IfEnd

  Int (E/9)+1→I      // dans ce qui suit on gère les collisions, si la balle change de case dans la
  Int (F/9)+1→J        matrice et rencontre un mur
  If Mat A[J,I]=3
    Then If I ≠ A              // si le mur est à droite ou à gauche
      Then -M→M
    IfEnd
    If J ≠ B                   // si le mur est au dessus ou en dessous
      Then -N→N
    IfEnd
  IfEnd
  I→A:J→B

  For 0→H To 200:Next                // petite pause pour bien voir la balle avancer
LpWhile V>0
```

Combinons toutes nos boucles pour obtenir le code final

Cette partie rassemble simplement les morceaux de code expliqués précédemment, elle rajoute juste la boucle liée à la gestion des différents niveaux et celle qui fait retirer la balle tant que l'arrivée n'est pas atteinte.

J'ai rajouté quelques commandes au début du programme pour paramétrer correctement la calculatrice et éviter d'éventuels problèmes, vous les trouverez en faisant [SHIFT] [MENU] (SET UP).

```
Deg:CoordOff:GridOff:AxesOff:LabelOff                    // initialisation des parametres

ViewWindow 1,127,0,1,63,0
Text 5,101,"DIRECT'"                    // chargement de l'image de fond avec les commandes
Text 13,101,"[ ◀ ] [ ▶ ]"
Text 25,101,"FORCE"
Text 33,101,"[ ▲ ]"
Text 40,101,"[ ▼ ]"
F-Line 99,1,99,63
StoPict 1

1→Z
Do                    // la boucle des niveaux
If Z=1                // chargement des niveaux
  Then [[3,3,3,3,3,3,3,3,3,3,3][3,1,0,0,0,0,0,0,0,0,3][3,0,0,0,0,0,0,0,0,0,3][3,3,3,3,3,3,0,0,0,0,3]
[0,0,0,0,0,3,0,0,0,0,3] [0,0,0,0,0,3,0,0,0,2,3][0,0,0,0,0,3,3,3,3,3,3]]→Mat A
IfEnd
If Z=2
  Then                // à vous d'inventer vos propres niveaux
IfEnd

Cls                   // chargement des décors
RclPict 1
For 1→I To 11
  For 1→J To 7
    If Mat A[J,I]=1   // on en profite pour enregistrer les coordonnées du départ et de l'arrivée
      Then I→A:J→B
    IfEnd
    If Mat A[J,I]=2
      Then I→C:J→D
      Circle 9(I-1)+4,9(J-1)+4,2
    IfEnd
    If Mat A[J,I]=3
      Then For 1→K To 9
        F-Line 9(I-1)+K,9(J-1)+1,9(I-1)+K,9J
      Next
    IfEnd
Next:Next
StoPict 2             // on enregistre le décor
```

```
 While A≠C Or B≠D              //  la boucle qui détecte l'arrivée
9(A-1)+4→E
9(B-1)+4→F
2→V:0→W
Do                          // interface pour la direction et la force du tir
  Cls
  RclPict 2
  SketchThickF-Line E,F,E,F
  F-Line E,F,E+3Vcos W,F+3Vsin W                  // on affiche la direction et la force du tir
  Do
    Getkey→G
  LpWhile G=0
  If G=28 And V<4             // on augmente la force
    Then V+1→V
  IfEnd
  If G=37 And V>1             // on diminue la force
    Then V-1→V
  IfEnd
  If G=38             // on augmente l'angle
    Then W+10→W
  IfEnd
  If G=27             // on diminue l'angle
    Then W-10→W
  IfEnd
LpWhile G ≠31

0→G
cos W→M
sin W→N
Do                          // à chaque itération de cette boucle on va faire avancer la balle
  E+VM→E
  F+VN→F
  SketchThickF-Line E,F,E,F                 // on affiche la nouvelle position de la balle
  G+1→G
  If G=2V             // cette partie permet de gérer la vitesse de la balle
    Then V-1→V
    0→G
   IfEnd
  Int (E/9)+1→I       // dans ce qui suit on gère les collisions
  Int (F/9)+1→J
  If Mat A[J,I]=3
    Then If I ≠A                // si le mur est à droite ou à gauche
        Then -M→M
      IfEnd
      If J ≠B                   // si le mur est au dessus ou en dessous
        Then -N→N
      IfEnd
  IfEnd
  I→A:J→B
  For 0→H To 200:Next                   // petite pause pour bien voir la balle avancer
LpWhile V>0
WhileEnd            // quand l'arrivée est atteinte

Z+1→Z               // niveau suivant
LpWhile Z ≤2                    // parce qu'il n'y a que 2 niveaux pour l'instant
```