



AOZ
STUDIO



Magic Book - Livre de Magie



Ce livre de Magie décrit des instructions souvent utilisées, les ingrédients pour devenir magicien et donner vie à vos créations. Pour en savoir plus vous pouvez aussi consulter le Guide utilisateur ou l'aide interactive d'AOZ Studio. Il y a près de 1000 instructions dans AOZ, pour faire tellement de choses !



Les écrans

L'écran utilisé par AOZ à une taille par défaut de 1920 x 1080 pixels pour les graphismes et de 80 x 25 caractères pour les textes.

Il est possible de programmer dans plusieurs écrans.

Imaginez les écrans comme une pile de feuilles de papier, vous les placez les uns sur les autres. Si l'un a des couleurs transparentes, vous verrez des parties de l'écran en dessous. Sauf indication contraire, toutes les instructions et fonctions ont un effet sur l'écran sélectionné. Vous devez dire à AOZ sur quel écran vous travaillez avec l'instruction Screen <s> . Par défaut, vous n'êtes pas obligé d'ouvrir l'écran 0.

Screen s

Active l'écran s (le screen 0 est activé par défaut). **Ex: Screen 1**

#DisplayWidth: w / #DisplayHeight: h (2 instructions ici)

Définit largeur ou hauteur de l'affichage. **Ex: #DisplayWidth: 1000**

=Display Width / =Display Height. **Ex: DW=Display Width**

Renvoie la valeur de la largeur ou de la hauteur de l'affichage AOZ.

Screen Open s, width, height, colours, mode, lines, border, tags

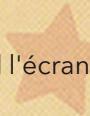
Ouvre un écran s, spécifiant la largeur et la hauteur en pixels, le Nb de couleurs de la palette. Facultatif : le mode (lowres, hires...), le Nb de lignes de la fenêtre de texte, la bordure, les balises pour les propriétés spéciales.

Screen To Front s / Screen To Back s

Déplace les écrans vers le haut ou le bas de la pile pour qu'ils recouvrent les autres. (Si s est omis, tire l'écran actuel vers le haut.)

Screen Show s / Screen Hide s

Rend l'écran visible ou non (Si s est omis, rend l'écran actuel visible.)



Cls / Cls c

Efface l'écran actuel **Ex: Cls.**

Efface l'écran actuel indexé par la couleur c. **Ex: Cls 0**



Affichez des textes

Vous pouvez utiliser les instructions Print ou Text pour afficher des textes.

Print " "

Imprimera le texte entre " ", ou le nombre si il n'y a pas de "

Ex : Print "Bonjour" Affiche Bonjour en tant que texte

Ex : Print 2 Affiche 2 en tant que chiffre

Ex : Print 2+2 Affiche en faisant l'addition (ce sont des nombres)

Print x, y\$, z# (etc...)

Affiche chaque variable dans l'ordre indiqué, en commençant à la position actuelle du curseur de texte.

Ex : Print "Bonjour", "AOZ"; "Studio" Notez les séparateurs , et ;

-Lorsqu'une virgule est utilisée comme séparateur, le curseur sera positionné sur tabulation suivante. (modifiable avec Set Tab n).

-Lorsqu'un point-virgule est utilisé, le curseur sera juste à droite.

Print Using f\$; x, y\$, z# (and so on)

Affiche chaque variable formatée en utilisant la chaîne de format f\$.

Certains caractères seront remplacés par le contenu des variables listées. **Ex : Print "### ~~~ #,###.#" ; 1,"ab",1234.56**

Résultat : + 1 ab 1 234,5

Curs On / Curs Off

Affiche ou masque le curseur de texte clignotant.

Locate x,y

Défini la position de l'affichage du Print en x, y. Ce sont des positions de caractères pas de pixels. (x=0, y=0 est en haut à gauche de l'écran). **Ex : Locate 10,10 : Print "Bonjour"**

Pen x

Définit la couleur de l'encre (pour le texte). Par défaut on écrit en blanc sur fond noir. Pen 1 pour blanc, 0 pour le noir, 4 pour rouge, 5 pour vert, 6 pour bleu, etc... **Ex : Pen 4 : Print "bonjour"**

Paper x

Définit la couleur du fond (arrière-plan). **Ex : Paper 1**

=Pen

Donne la valeur de la couleur de Pen utilisée.

Ex : x = Pen met dans la variable x la valeur de Pen.

=Paper

Donne la valeur de la couleur du fond utilisée.

Ex : Print Paper affiche directement la valeur de Paper

Text x,y,t\$

Affiche t\$ aux coordonnées graphiques x,y sur l'écran actuel, en utilisant les paramètres de police graphique actuels. Si les coordonnées x et/ou y sont omises, les coordonnées seront lues à partir du curseur graphique.

Ex d'affichage en utilisant des écrans texte et graphique :

```
#googleFont:"syncopate"
// Creation d'un écran 0 :
Screen Open 0,1920,1080,32,0 : Curs Off
// Creation d'un écran 1, transparent, paper noir et Ink blanc
Screen Open 1,1920,1080,32,0 : Curs Off : Set Transparent 0 :
Paper 0 : Ink 1
Set font "syncopate", 70

Screen 0 // Switch sur l'écran 0
Actor "magic", image$="magic", X= 0, Y=10
Screen 1 // Switch sur l'écran 1
Text 5,90, "WELCOME"
Print "AOZ"
```



Commandes de dessin

Note : AOZ se souviendra du dernier point d'affichage des graphismes, appelé curseur graphique.

Plot x,y

Dessine un pixel aux coordonnées graphiques x, y

Draw x1,y1 To x2,y2

Trace une ligne entre x1,y1 et x2,y2 (avec la couleur d'encre).

Ex : Ink 1 : Draw 50,50 To 800, 50

Draw To x1,y1

Trace une ligne du curseur graphique vers x1,y1.

Polyline x1,y1 To x2,y2 To x3,y3 (et ainsi de suite)

Trace une ligne du premier au point suivant, puis au suivant, etc. Si x1, y1 sont omis, le dessin commencera au niveau du curseur.

Ex : Ink 1 : Polyline 100, 100 To 100, 400 To 400, 400

Polygon x1,y1 To x2,y2 To x3,y3 (et ainsi de suite)

Trace des lignes entre chacun des points spécifiés pour former un polygone.

Ex: Ink 4 : Polygon 100,100 To 100, 400 To 400, 400

Box x1,y1 To x2,y2

Dessine une boîte définie par les coins x1,y1 et x2,y2.

Bar x1,y1 To x2,y2

Dessine une zone remplie définie par les coins x1,y1 et x2,y2.

Circle x,y,r

Dessine un cercle centré sur x, y, avec un rayon de r.

Disc x,y,r

Dessine un cercle centré sur x, y, avec un rayon de r.

Ellipse x,y,rh,rv, rot, fill

Dessine une ellipse centrée en x, y, avec un rayon horizontal de rh et un rayon vertical de rv, tourné de rot, remplie de couleur fill.

Filled Ellipse x,y,rh,rv, rot, fillIndex, lineIndex

Dessine une ellipse centrée à x, y, avec un rayon horizontal de rh et un rayon vertical de rv, tournée de rot, de la couleur indexée par lineIndex, et remplie de la couleur indexée par fillIndex.

Ex: Ink 4 : Filled Ellipse 960,540,200,100,90

Star x,y,r1,r2,points, rot, fill

Dessine une étoile pointue centrée sur x, y, avec un rayon intérieur de r1 et un rayon extérieur de r2, tourné de rot...

Filled Star x,y,r1,r2,points,rot,fillIndex,lineIndex

Dessine une étoile remplie, pointue ...

Ex: Ink 6 : Filled Star 960,540,700,50,20

Ink c

Définit la couleur de l'encre graphique (indexée par c dans la palette de l'écran actuel).

=Ink

Renvoie l'index de couleur de l'encre graphique.

AOZ comprend 1400 instructions, pour en savoir plus utilisez le Guide Utilisateur et la barre de recherche dans AOZ Studio.



Définissez la scène, affichez l'acteur

Actor est une instruction sophistiquée permettant de facilement manipuler des images 2D, 3D, des vidéos, sur l'écran. Vous pouvez vérifier et contrôler le comportement de vos Actors, gérer les collisions et bien plus encore. Avec Actor vous pouvez faire des jeux, d'excellentes interfaces, des portails, des applications,...

Voici les principaux paramètres d'Actor (il y en a plus) :

- **Nom ou index:** Définissez l'actor en lui donnant un nom.

Exemple : Actor "A", Actor "magie", ou un index.

Exemple utilisant Actor avec un index (une valeur entière) :

For J=0 to 4

 Actor J, Image\$="magic", X=10, Y=J*200,
 EndX=1800, duration=1000*J

 Next J

 Wait Input

- **Type\$:** Type de l'acteur. « 2d » par défaut, ou « 3d »
- **X :** Position horizontale de l'acteur à l'écran. 0 par défaut.
- **Y :** Position verticale de l'acteur à l'écran. 0 par défaut.
- **Z :** (3D) Position en profondeur de l'acteur à l'écran. 0 par défaut.
- **StartX :** Position horizontale de départ du mouvement de l'actor.
- **StartY :** Position verticale de départ du mouvement de l'actor.
- **StartZ :** (3D) La position de départ en profondeur du mouvement de l'actor.
- **EndX :** Position horizontale à la fin du mouvement de l'actor.
- **EndY :** La position verticale à la fin du mouvement de l'actor.
- **EndZ :** (3D) La position de profondeur à la fin du mouvement de l'actor.

- **Duration :** Durée du mouvement de l'actor en milli secs.
Ex : Actor "magic", Image\$="magic", X=0, EndX=2000, duration=4000 : Wait input
- **Image / Image\$:** Index ou nom de l'image de l'actor. Ex: Image=1 ou Image\$="magie ». Les images sont situées dans votre dossier app/resources/images, ou sont déjà dans le dossier AOZ Drive /AOZ Drive/resources/images, ou avec l'instruction Load Asset spécifiant là où il se trouve.
- **Video\$:** Nom de la vidéo dans votre dossier app/resources/images ou chargée par Load Asset.
Ex : Actor "mycam", Video\$="@webcam" affiche la webcam1 par défaut (pas dans l'AOZ Viewer, uniquement dans les navigateurs car il faut l'autorisation d'utiliser la caméra)
- **Videoplay :** lis ou stop la vidéo. True par défaut.
- **Videoloop :** lis la vidéo en boucle. True par défaut.
- **Control\$:** Dirige l'actor ("keyboard", "mouse", "joystick").
Ex : Actor "m", Image\$="magic", control\$="keyboard"
Par défaut c'est avec les touches fléchées et les touches Q D Z S (il est possible de changer les touches).
- **Transition\$:** Assigne un déplacement à l'actor, par défaut "linear". AOZ utilise la puissante API CREATE.JS, avec ces effets : https://www.createjs.com/demos/tweenjs/tween_sparktable
- **Scale :** valeur décimale de zoom de l'actor. Valeur par défaut 1.0 (taille normale). **Ex : Scale=0.5** (50% en X et en Y)
- **ScaleX / ScaleY :** valeur décimale de zoom en X / Y
- **StartScale / EndScale:** valeur décimale des début / fin de zoom.
- **StartScaleX / EndScaleX / StartScaleY / EndScaleY :** valeur décimale des début / fin de zoom en X / Y.
- **Angle / StartAngle / EndAngle :** valeur de l'angle / début / fin de rotation de l'actor en degrés. Par défaut 0.

- **Anglez / Startanglez / Endanglez :** (3D) Modification de l'angle Z / des début / fin / de rotation de l'actor 3D. Si pas défini le dernier angle est utilisé.
- **Alpha / StartAlpha / EndAlpha:** Valeur décimale d'opacité / Début / Fin de l'actor. Par défaut 1.0 (total visibilité).
- **SkewX / StartSkewX / EndSkewX:** Valeur décimale de la distorsion verticale / Début / Fin de l'actor. Par défaut 1.0.
- **SkewY / StartSkewY / EndSkewY:** Valeur décimale de la distorsion horizontale / Début / Fin de l'actor. Par défaut 1.0.
- **Visible :** True/False affiche ou masque l'actor.
- **Enable :** Active ou désactive l'actor. Si la valeur est False, l'actor sera toujours affiché à l'écran, mais les contrôles, les animations et les actions de la souris seront désactivés.
- **Collision :** active ou désactive toutes les collisions avec cet actor. Si False, aucun effet de collision ne sera appliqué.
- **LoopMove :** True/False, joue le mouvement en boucle.
- **ActionMove\$:** "play" ou "pause" le mouvement.
- **Spritesheet\$:** Nom de la sprite-sheet d'animation de l'actor.
- **Anim\$:** Nom de l'animation à jouer telle que définie par la spritesheet ou par l'instruction Actor Animation.
- **Actor Animation :** Instruction pour définir une animation sans Spritesheet et la jouer avec Actor.
 - Sequence\$ est une chaîne avec chaque nom ou numéro d'image séparé par une virgule.
 - Les images peuvent être situées dans le dossier resources/image ou chargées d'ailleurs avec Load Asset.
 - La dernière valeur peut être "L" pour boucler, "E" pour terminer, "R" pour inverser et boucler.
 - Vous pouvez régler un délai en millisecondes entre 2 images comme ceci : "1:1000, 2,..." pour faire une pause de 1 sec. entre les images 1 et 2.
 - Actor "Magic", AnimPlay=False (ou True) pour jouer ou arrêter l'animation

Exemple:

Actor Animation "gorun", Sequence\$="1,2,3,4,L"
Actor "magic", Anim\$="gorun"

- **LoopAnim** : True/False, joue l'animation en boucle
- **Hotspot\$ / HotspotX / HotspotY** : Valeur décimale ou texte déterminant la position du point chaud "hot spot" (voir le chapitre Hotspot). **Ex Hotspot\$="middle"**
- **HRev / VRev** : True/False, effet miroir horizontal ou vertical.
- **LeftLimit ou RightLimit** : Valeur en pixels du bord gauche ou droit (en X) de la limite de déplacement de l'actor.
- **TopLimit ou BottomLimit** : valeur en pixels du bord en Y de la limite de déplacement de l'actor.
Ex : Actor "magic", Image\$="magic", control\$="mouse", TopLimit=100, BottomLimit= 600
- **LookAt\$** : Crée un mouvement de l'actor qui va suivre automatique un objet, un actor ou un point sur l'écran.
- **Auto\$** : Définit un mouvement automatique de l'actor. Il utilise les mêmes paramètres que les propriétés de Control\$: offsetX, offsetY, angle, forward et backward.
- **Group\$**: Assigne l'actor à un groupe, qui porte un nom. Le groupe peut ensuite être manipulé comme un actor.
Exemple pour un groupe d'ennemis dans un jeu vidéo.
- **OnChange\$** : Nom de la procédure appelée lorsque l'actor subit un changement (mouvement, forme, transparence...).
- **OnAnimChange\$**: Nom de la procédure appelée lorsque l'animation de l'actor subit un changement.
- **OnCollision\$** : Nom de la procédure appelée lorsque l'actor entre en collision avec un autre.
Note : Une autre façon pour gérer simplement les collisions est la fonction Actor Col() qui devient vraie (True) si collision.
Ex : If Actor Col ("magic", "lucie")=True Then ...

- **OnLimit\$** : Nom de la procédure appelée lorsque l'actor atteint l'un des bords.
- **Onmouse\$**: Nom de la procédure appelée à chaque action de souris sur l'actor.
- **Onmouseenter\$** : Nom de la procédure appelée lorsque le pointeur de la souris passe sur l'actor.
- **Onmouseclick\$**: Nom de la procédure appelée à chaque clic sur la souris.
- **Onkeypress\$**: Nom de la procédure appelée à chaque appui sur une touche.
- **Actor X / Actor Y** : donne la position de l'actor sur l'écran
Ex : PosX = Actor X ("magic")
- **Actor Del ""** : Instruction pour effacer des actors.
Ex : Actor Del "magic" détruit l'actor magic
Ex : Actor Del "*" détruit tous les actors créés

Exemple (déplacez Magic avec le clavier) :

```

Actor "Magic", X=10, Y=200, Image$="magic", OnMouse$="MOUSEEVENT",
Control$="keyboard", Oncollision$="MOUSEEVENT"
Actor "Lucie", X=400, Y=200, Image$="lucie", OnMouse$="MOUSEEVENT",
OnMouseEnter$="MOUSEEVENT"

Do
  Refresh // synchro de l'affichage, a utiliser avec des graphismes
Loop

Procedure MOUSEEVENT [EVENT$, BUTTON, INDEX$, INDEX1$, INDEX2$]
  Cls : Print INDEX$;" : "; INDEX1$ ; " "; INDEX2$ + " "
  EVENT$:"+EVENT$ + " "
  BUTTON:";BUTTON
End Proc

```

Note : il y a de nombreuses autres fonctionnalités pour Actor, décrite dans l'aide intégrée à la barre de recherche d'AOZ Studio.



Boucles et reboucle



Do...Loop

Entre Do et Loop le code est continuallement traité en boucle.

While [condition] ...Wend

Exécute le code entre While et Wend tant que la condition est vraie (True). Lorsque la condition devient False, AOZ se branche sur le code immédiatement après l'instruction Wend.

While J<5

 Inc J : Print J // Inc J augmente J de 1, idem à J=J+1
Wend

Repeat...Until [condition]

Comme While Wend, répète le code entre Repeat et Until tant que la condition, testée cette fois à la fin de la boucle, est vraie.

Repeat

 Inc J : Print J
Until J>4



For ... To ... Step ... Next

La fameuse boucle For Next, très utile.

Step est une option. Dans la boucle l'indice (en dessous I) peut être utilisé comme s'il s'agissait d'une variable normale.

For J=1 To 10 Step 2

 Print J // en boucle, de J égal 1 à 10, par pas de 2
Next J

If ... Then ... Else

Est une autre fameuse instruction, pour faire les tests. Else est une option. Il y a plusieurs façons de l'utiliser dont la plus simple : If <condition est vraie> Then <fait ca>

Ex : If A=1 Then Print "A est égal à 1" Else Print "pas égal"



Musique et Sons

Shoot / Boom / Bell / Explode

Joue des sons simples prédéfinis. **Ex : Boom**

SamPlay

Sfxr extension Interface of the Sfxr library, synthesizer of game

Sound Effect:

- * Sfx Play // play a sound effect without fuss
- * Sfx Create // create a new sound effect
- * Sfx Del // Delete a sound effect
- * Sfx Mutate // Slightly change the sound effect
- * Sfx To // Assign a sound effect to a voice
- * Sfx Set // Set a property of a sound effect
- * =Sfx Json\$ // Returns the JSON definition of a sound effect
- * =Sfx Get // Return the value of one of its properties
- * =Sfx Get\$ // Return the value of one of its string properties

Clavier

Input x\$

Retourne dans la variable (ici x\$) le texte tapé au clavier. Il est possible d'afficher un texte avant un point virgule comme ceci :

Ex : Input "nom ?";nom\$

=Inkey\$

Retourne la touche appuyée sinon une chaîne vide "". Pour les touches de fonction, AltGr,... vous devrez utiliser la fonction ScanCode.

=ScanCode

Renvoie le code (standard ASCII) de la dernière touche de Inkey\$.

Do

A\$ = Inkey\$: B= ScanCode

If A\$<>"" then Print A\$,B // Si A\$ est vide la boucle continue

Loop

Analog/Digital Gamepad/Joystick

=Gamepad Axis (g, axis)

Position de l'axe sur la manette analogique/numérique "g", entre : -1 (-100%) et 1 (100%) avec 0 = centre (0%). (Utilisez l'application Joystick Tester).

=Gamepad Button (g, b)

Retourne True (vrai) quand le bouton b du joystick g est appuyé.

=Joy (j)

Renvoie l'état du joystick j sous forme binaire. Chaque chiffre (bit) représente une fonction du joystick numérique, comme suit :

Bit :	Signifie :
1	Joystick est en haut
2	Joystick est en bas
4	Joystick est à gauche
8	Joystick est à droite
16	Bouton Fire est appuyé

Note : il existe plusieurs autres instructions pour utiliser les joysticks.

Et la souris !

=X Mouse =Y Mouse

Renvoie la position horizontale et verticale de la souris.

=Mouse Key

Donne un nombre indiquant l'état des boutons de souris :

1 = le gauche est enfoncé, 2 le droit, 3 les deux, 4 le bouton central.

Do // Essayez d'appuyer sur plusieurs boutons de souris

Print Mouse Key, X Mouse, Y Mouse

Loop



Limit Mouse x1,y1 To x2,y2

Limite le déplacement de la souris au rectangle x1,y1 à x2,y2

Change Mouse <nom de l'image> (sans extension)

Vous pouvez utiliser une image comme pointeur de souris. PNG uniquement, avec une taille maximale de 128x128px.

Placez l'image dans votre dossier d'application :

ressources/ressources/souris **Ex : Change Mouse "01mouse"**

Utilisation de la souris avec des exemples :

Exemple pour savoir dans quelles zones se trouve le pointeur souris :

Reserve Zone 2 : Line Width 10 //réserve 2 zones, épaisseur de ligne

Ink 1 : Pen 1 // Dessine et définit les 2 zones :

Box 80,400,1800,200 : Set Zone 1, 80,400,1800,200

Box 80,700,1800,200 : Set Zone 2, 80,700,1800,200

Do // Boucle pour situer la souris

A= Mouse Zone // Retourne le N° de zone (0= hors zones)

Locate 1,5 : Print A // et l'affiche.

Loop

Exemple pour savoir si l'utilisateur clique et où (sur un grand actor) :

*Les Procédures sont expliquées plus loin :

```
Actor "forest", Image$="forest", Y=50, OnMouse$="MOUSEEVENT"
Wait Key // pour que le programme ne se termine pas
```

```
Procedure MOUSEEVENT [INDEX$, EVENT$, X, Y]
    Cls : Print "ACTOR:"; INDEX$ ; "  EVENT$:" ; EVENT$ ;"  "; X;Y
End Proc
```



Attends-moi !

Wait Click

Met l'application en pause jusqu'à l'appui d'un bouton de la souris, ou que l'écran ait été touché.

Wait Input

Met l'application en pause jusqu'à l'appui d'un bouton de la souris, ou que l'écran ait été touché, ou qu'une touche du clavier ait été enfoncee.

Wait x

Suspend l'application pendant x secondes. **Ex Wait 2**

Refresh (or Wait Vbl)

Synchronise le code avec l'affichage, **il est nécessaire de mettre cette instruction dans les boucles d'affichage.**

Exemple de boucle avec 2 affichages, essayez sans le Refresh

```
Do
    score=score+1
    UI Progress "barre", x=10, y=200, width=1900, height=50, value=score/20
    Actor "magic", Image$="magic", X=score
    Refresh
Loop
```



Le Code spaghetti !

Goto et Gosub

L'abus d'utilisation de l'instruction Goto va rendre votre code difficile à suivre et à maintenir, il est préférable d'utiliser Gosub ou les procédures.

D'abord les labels

Une "étiquette" (label en Anglais) identifie un « point d'arrivée » dans le programme. C'est une position à laquelle le programme va pouvoir aller en utilisant les instructions : Goto, Gosub, Then, Else, On, etc.

Les étiquettes peuvent contenir des lettres et des chiffres, mais doivent commencer par une lettre. En option, les étiquettes peuvent être calculées. Une définition d'étiquette doit être le premier élément d'une ligne de code et avoir un deux-points : à la fin de celle-ci. Lorsque vous faites le saut à l'étiquette là le : est omis. Ce sera plus facile avec un ex :

MyLabel: ' une fois défini, vous pouvez l'utiliser comme ceci:

```
Goto MyLabel ' Passer à MyLabel via Goto (non : )  
Gosub MyLabel ' Passer à MyLabel via Gosub (non : )  
' Étiquettes dynamiques « calculées »:  
Gosub Chr$(Asc("A") + n) ' Si n=0, label="A", Si n=1, label="B"
```



Numéro de Ligne

Si un label débute par un N° de ligne cela fonctionne de même mais il n'est pas utile de mettre les : (2 points)

100 Print "Hello."	' avec N° de ligner
Gosub x*10	' On peut calculer le saut à la ligne
Goto 350	' Et bien sur aller à une ligne spécifique

Les instructions **Goto** et **Gosub** se branchent sur l'étiquette ou le N° de ligne spécifié et continuent l'exécution du code.

Gosub diffère : si le code rencontre ensuite l'instruction **Return** il retourne juste après le Gosub. Gosub a donc toujours un Return.



Procédure

Une procédure est un bloc de code que vous pouvez exécuter en l'appelant par un nom que vous choisissez, le programme ne rentre dans la procédure que si il y a un appel :

```
Myproc [3] // appel de la procédure  
Myproc [10] // appeler à nouveau la même procédure
```

```
Procedure Myproc [A] // avec le paramètre passé dans [], ici A  
    Print A*2 // autant de lignes de code avant le End Proc  
End Procedure // déclare la fin et retourne après l'appel
```

Autre exemple :

```
Print Myproc$[3]
```

```
Procedure Myproc$[a] // entrée de la valeur a dans la procédure  
    S$=str$(a*2) // Str$() converti un chiffre en texte  
End Procedure[S$] // sortie de la valeur dans S$.
```



Pop Proc

Quitter la procédure sans exécuter le code restant

Exit App

Quitte une application et ferme sa fenêtre.

Variables, constantes et types de données

AOZ est une extension du célèbre langage BASIC, comme Ada, Pascal, SQL,... il est insensible aux majuscules/minuscules, c'est-à-dire que la variable FLAG est la même que flag. L'instruction Print et idem que print.

Les noms de variables peuvent contenir des lettres, des chiffres et des _
Par défaut, une variable est un entier, sinon vous devez déclarer le type de la variable en ajoutant un symbole comme ceci :

- **Integer** (nombres entiers) : X=68000 (par défaut)
- **Float** (nombres réels) : Y#=123,45 (notez le #)
Ex : Temp=19.5 : pas bon Temp#=19.5 : bon
- **Strings** (alphanumériques) : Z\$="Fred@3" (notez le \$)
Les chaînes doivent être entourées de guillemets doubles.
Ex : NOM="AOZ" : pas bon NOM\$="AOZ" : bon
- **Hexadécimal** X = \$03D0 (notez le \$)
- **Binary** Y=%0110 (notez le %)
- **Dim** Avec dimensions Manuels *Ex : Dim A(10,20)*
Automatiques *Ex : Dim A\$() : A\$(1000) = "Hello!" : Print A\$(1000)*
Ou les mixer les deux *Ex : Dim TOTO@(, 100)*

SAUVEGARDE LOCALE ET BASE DE DONNEES

La mémoire locale des navigateurs (exemple Chrome) peut conserver des données sous forme de Cookies, mais attention, ce tant que la mémoire cache du navigateur n'est pas effacée.

Sauver un fichier ou des données dans la mémoire cache.

Open In / Open Out / Close

1. Ouvrir en écriture avec : Open Out N° de canal, nom de fichier
2. Sauvegarder avec : Print #N° de canal ouvert, donnée
3. Fermer un canal ouvert avec un : Close N° de canal ouvert

Exemple:

```
Filename$="MyFile.txt"  
Input SA$      // texte à sauvegarder
```

```
Open Out 1, Filename$  
Print #1,SA$  
Close 1
```

```
Open In 1, Filename$  
Input #1, RD$  
Print "READ ";RD$  
Close 1
```

Utiliser une base de données.

AOZ comprends les instructions pour Firebase (hébergé chez Google) et pour SQL (hébergé chez AOZ Studio où ailleurs). Résumé pour SQL :

- **DB Database** : sélectionne une base de données
- **DB Table** : sélectionne une table
- **DB Read** : copie les valeurs de l'enregistrement dans vos variables AOZ
- **DB Write** : copie le contenu de vos variables AOZ dans l'enregistrement sélectionné
- **DB New** : sélectionne un nouvel enregistrement vide. (A utiliser avant un nouveau DB Write.)

- **DB Search All** : recherche tous les enregistrements et sélectionne le premier
- **DB Search** : recherche les enregistrements suivant un critère de recherche simple et sélectionne le premier
- **DB Search Sql** : recherche les enregistrements suivant des critères de recherche écrit en langage SQL et sélectionne le 1er
- **DB First** : sélectionne le premier enregistrement résultant de la recherche
- **DB Last** : sélectionne le dernier enregistrement résultant de la recherche
- **DB Next** : sélectionne l'enregistrement suivant résultant de la recherche
- **DB Previous** : sélectionne l'enregistrement précédent résultant de la recherche
- **DB Pointer** : récupère l'identifiant de l'enregistrement sélectionné
- **DB Point On** : sélectionne un enregistrement dont on connaît l'identifiant

Exemple de programme qui écrit et lit :

```
name$="AOZ" : score=987654  
DB Write "name$, score"  
  
DB Read "name$, score"  
Print name$, score
```



Note : si on ne défini pas le nom de la base et/ou de la table, AOZ les créés par défaut. Voir le Guide Utilisateur.

AOZ comprends 1400 instructions, ce livre de Magie vous donne quelques tours, pour le reste il existe le livre de Magie plus complet sur : doc.aoz.com.

Et le support sur DISCORD : <https://fr.aoz.studio/faq>





AOZ STUDIO

V1.0 Dec.22