ARES Slideshow

Mode d'emploi V1.3

ARES Video offre la possibilité d'utiliser des diaporamas "intelligents" créés par l'utilisateur pour l'entraînement au tir au laser. Dans ces derniers, il est non seulement possible d'afficher différentes images, mais aussi de définir de nombreuses façons la transition entre ces images et même de les rendre interactives. Cela signifie que les diaporamas peuvent réagir aux tirs. Les diaporamas sont écrits dans un langage de script simple qui ne nécessite pas de connaissances en programmation, mais qui offre la possibilité de créer des processus complexes. Pour faciliter la création des diaporamas, il existe également un éditeur qui permet, entre autres, de lire les diaporamas pas à pas. Il offre en outre une mise en évidence en couleur des mots-clés et affiche éventuellement des messages d'erreur lors de l'exécution.

Structure de principe et fonctionnement des diaporamas

Les fichiers de diaporama ont l'extension ".dia". Il s'agit de simples fichiers texte qui peuvent être ouverts avec n'importe quel éditeur de texte. Lors de la lecture, le fichier est traité ligne par ligne. Les lignes vides sont autorisées et sont tout simplement ignorées. Il est en outre possible d'écrire des commentaires. Ceux-ci commencent par un symbole dièse (#) qui fait en sorte que tout le texte soit ignoré jusqu'à la fin de la ligne correspondante. Chaque ligne contient une seule instruction, qui peut être de n'importe quelle longueur. La fin de la ligne marque donc également la fin d'une commande. Un signe explicite, comme c'est le cas dans d'autres langages de programmation, n'est pas nécessaire. Le diaporama s'arrête automatiquement lorsque la dernière ligne a été traitée. De même, il peut être arrêté à tout moment en appuyant sur la touche d'échappement.

Éléments de base des scripts : Commandes, variables et logique

Les capacités des scripts de diaporama peuvent être grossièrement divisées en trois domaines. Il y a les commandes, les variables et la logique.

Les commandes sont les commandes qui déterminent l'aspect réel du diaporama. Elles permettent d'afficher des images, d'éditer du texte, de diffuser des sons ou de définir des temps d'attente.

Les variables servent à enregistrer des valeurs et à calculer ou à prendre des décisions en fonction de ces valeurs. On peut définir et utiliser autant de variables que l'on veut. Le système lui-même met également à disposition de nombreuses informations sur les variables. Par exemple, la position de l'impact du dernier tir, la taille de différents éléments ou le temps écoulé depuis le début.

La logique est la partie qui détermine le déroulement des diaporamas et fournit ainsi "l'intelligence". Il y a des ramifications, des boucles, des sauts et les liens logiques ET et OU. Il est ainsi possible de parcourir différentes parties du diaporama en fonction des valeurs des variables.

Aperçu des commandes

Toutes les commandes doivent être placées au début d'une ligne, car elles provoquent certes une action, mais ne peuvent pas être traitées par d'autres commandes. La seule exception est la commande RAND qui peut être utilisée partout où il y a une variable ou un nombre.

Commande	Fonction
OPEN "%Fichier%"	Ouvre un fichier image et l'affiche à l'écran. Le fichier doit se trouver dans le même dossier que le diaporama. Seuls les formats suivants sont supportés (jpg, png, bmp). Le nom du fichier est entre guillemets. Le nom de fichier peut également être généré à partir d'une combinaison de texte et de variables. exemple : **OPEN "Image" + $Var + ".jpg"** Si la variable $Var a la valeur 1, le fichier image "Image1.jpg" est ouvert, si elle a la valeur 2, le fichier "Image2.jpg" est ouvert.
OPEN_SCORE "%Fichier%"	Similaire à OPEN, mais le fichier image n'est pas affiché, il est ouvert en arrière-plan comme "gabarit de points". Idéalement, le pochoir devrait avoir la même taille que l'image affichée. Après un tir, la couleur du gabarit de points à l'endroit touché est disponible via 3 variables. Il est ainsi possible de définir des zones d'impact au choix, qui ne doivent toutefois pas être visibles pour le tireur.
OPEN_DECAL "%Fichier%"	Sélectionne le fichier pour l'objet decal. Il s'agit d'une image qui peut être affichée au-dessus du diaporama, dont la position et la taille sont librement réglables via des variables. Il est également possible de réaliser des mouvements en changeant continuellement la position.
PRINT "%Text%"	Définit le texte de l'objet texte sur le texte spécifié. L'objet texte peut par exemple être utilisé pour afficher des messages, mais aussi des scores et des temps. Comme pour les commandes OPEN, le texte peut être composé d'une combinaison de morceaux de texte et de valeurs. Exemple **PRINT "Heure : " + $RUNTIME +" secondes"** Définit le texte comme "Temps : 31,415 secondes" où le nombre est la valeur actuelle de la variable $RUNTIME, qui contient la durée de fonctionnement du diaporama jusqu'à présent.
SON "%Fichier.wav%"	Joue un fichier son au format wav. Cette commande ne doit pas être utilisée trop souvent à la suite, mais il faut attendre que le son précédent ait été joué avant de passer au suivant.
WAIT %TEMPS%	Attend pendant le temps indiqué en secondes avant d'exécuter la ligne suivante. exemple : **WAIT 1,5**
BORD %Chiffre%	Génère un nombre aléatoire compris entre 0 et le nombre saisi moins 1. Seuls des nombres entiers sans partie décimale sont générés. Si l'on souhaite obtenir des nombres à virgule, il faut diviser le résultat en conséquence. exemple : **BORD 5** Donne un nombre aléatoire qui peut être 0,1,2,3 ou 4. **(RAND 100)/10** Donne un nombre aléatoire entre 0,0 et 9,9

VARIABLES ET OPÉRATIONS ARITHMÉTIQUES

Une variable a un nom et une valeur peut lui être attribuée. Le nom de la variable doit commencer par un signe dollar ($) et peut ensuite être composé d'un nombre quelconque de lettres majuscules et minuscules ou de chiffres. Les trémas ne sont pas autorisés et le seul caractère spécial autorisé est le trait de soulignement (_). Chaque variable est créée automatiquement dès que son nom apparaît pour la première fois. Si une valeur ne lui est pas déjà attribuée, elle prend la valeur 0. La valeur est un nombre à virgule fixe avec 3 chiffres après la virgule. La plus petite valeur supérieure à 0 est donc 0,001. La plus grande valeur possible est 2.000.000 (2 millions). Si une valeur supérieure est attribuée à une variable, elle est automatiquement limitée à ces 2 millions. Des valeurs négatives sont

également possibles, de sorte que l'ensemble de la plage de valeurs va de -2.000.000 à 2.000.000. Pour les valeurs, le point et la virgule sont autorisés comme séparateurs décimaux.

Une variable conserve sa valeur jusqu'à la fin du diaporama, à moins qu'une nouvelle valeur ne lui soit explicitement attribuée. L'affectation d'une valeur se fait par un simple signe égal (=), la valeur à droite du signe égal étant affectée à la variable à gauche. La valeur à droite peut également être une variable. Les affectations en chaîne fonctionnent également.

```
$varA = 5            #La variable $varA a ont la valeur 5
$varB = $varA        #La variable $varB a ont aussi la valeur5
$varA = $varB = 7    #Les variables $varA et $varB ont la valeur 7
```

Les variables et les valeurs fixes permettent d'effectuer les quatre opérations de base. Les résultats de ces calculs peuvent être utilisés comme entrée pour des commandes ou être affectés à des variables pour être stockés. Ces calculs respectent la règle du point avant le tiret, mais il est également possible d'utiliser des parenthèses rondes pour influencer l'ordre de traitement.

```
$varA = 2 + 1,5                #La variable $varA a la valeur 3,5
$varB = $varA * 2              #La variable $varB a la valeur 7
$varC = $varB/3                #La variable $varC a la valeur 2,333
$varD = ($varA - $varC)*2      #La variable $varD a la valeur 3,334
```

Il arrive souvent que l'on veuille calculer la valeur d'une variable avec une autre valeur et réaffecter le résultat à la même variable. Pour ce faire, il existe une notation abrégée dans laquelle le symbole de l'opération arithmétique est suivi du signe égal.

```
$varA = $varA + 5    #La variable $varA est augmentée de 5
$varA += 5           #La variable $varA est aussi augmentée de 5
$varB -= $varA       #$varB est diminué de la valeur de $varA
```

LOGIQUE

Les commandes logiques permettent d'influencer le déroulement du diaporama en associant l'exécution des commandes à des conditions.

CONDITIONS

sont des expressions dans lesquelles des variables ou des valeurs sont comparées entre elles. Le résultat de cette comparaison peut être vrai ou faux. Les comparaisons suivantes sont possibles :

$varA == $varB	Égalité
$varA != $varB	Inégalité
$varA < $varB	Petit
$varA > $varB	Plus grand
$varA <= $varB	Plus petit ou égal
$varA >= $varB	Plus grand ou égal

Notez qu'un signe d'égalité double doit être utilisé pour la comparaison à l'égalité. Le signe d'égalité simple est utilisé pour l'affectation.

Les valeurs ou les variables peuvent également constituer une condition en soi. Si une seule variable est indiquée à l'endroit où devrait se trouver une condition, la condition est considérée comme non remplie si cette variable vaut exactement 0, sinon elle est considérée comme remplie.

Plusieurs comparaisons individuelles peuvent être reliées entre elles à l'aide des mots-clés **AND** et **OR**, des parenthèses rondes pouvant également être utilisées pour déterminer l'ordre de traitement (de l'intérieur vers l'extérieur).

```
IF $varA < 5 OR $varA == 8
#...
ENDIF

IF ($varA >= 10 AND $varB != 7) OR $varC == $varD
#...
ENDIF
```

IF [CONDITION]...ELSE...ENDIF

La **construction IF** permet d'exécuter des parties du script selon qu'une certaine condition est remplie ou non. La **partie ELSE** optionnelle permet d'exécuter une autre partie si la condition n'est pas remplie. Pour une meilleure lisibilité, les parties des différentes branches sont mises en retrait dans l'exemple. Cela est autorisé et recommandé, mais pas nécessaire.

```
#Exemple IF sans branche ELSE
IF $varA < 5
  PRINT "A est inférieur à 5"      #Seulement si A inférieur à 5
ENDIF

#Exemple IF avec branche ELSE
IF $varB == 3
  PRINT "B est 3"
ELSE
  PRINT "B n'est pas 3"
ENDIF
```

WHILE [CONDITION]...ENDWHILE

Une boucle While est utilisée pour parcourir une certaine section encore et encore tant qu'une condition est remplie. Lorsque le flux de programme atteint la ligne avec le **WHILE**, il est vérifié si la condition est remplie. Si c'est le cas, la partie suivante du programme est exécutée, sinon le flux de programme saute à la ligne après l'**instruction ENDWHILE** correspondante. Lorsque le programme atteint la ligne avec **ENDWHILE** après l'exécution des instructions, il vérifie si la condition est toujours remplie. Si c'est le cas, le programme remonte et répète toute la partie. Chaque fois que le diaporama effectue un tel retour au début d'une boucle, il fait automatiquement une pause de 0,01 seconde. Il est donc parfaitement possible de créer des boucles infinies, c'est-à-dire des boucles dont la condition est toujours remplie. Dans de nombreux autres "langages de programmation", de telles boucles infinies entraîneraient de graves erreurs.

```
#Émission d'un compte à rebours
$Countdown = 3
WHILE $Countdown > 0      # répéter jusqu'à ce qu'on atteigne 0
  PRINT $Countdown        #Sortir le nombre actuel
  $Countdown -= 1         #Réduire le nombre de 1
  WAIT 1                  #1 seconde d'attente
ENDWHILE
PRINT ""                  #Texte Vider
```

BRAKE

Brake n'est pas une construction en soi, mais une commande qui permet de quitter une **boucle WHILE**, même si la condition après sa **commande WHILE** est encore remplie. Si le déroulement du programme rencontre un **BREAK** à l'intérieur d'une **boucle WHILE**, il passe immédiatement à la ligne suivant l'**ENDWHILE** correspondant. Il est ainsi possible de créer des boucles avec plusieurs conditions d'interruption. On peut par exemple créer une boucle sans fin et la quitter quand même. Selon le style d'écriture que l'on préfère, cela peut conduire à des programmes courts et clairs.

```
WHILE 1               #Boucle sans fin : "1" est toujours vrai
  IF $RUNTIME > 10
       BREAK          #Après 10 secondes, on quitte la boucle
  ENDIF
ENDWHILE
```

Les boucles peuvent également être imbriquées les unes dans les autres, une **commande BREAK ne** quittant que la plus interne des boucles actuellement actives.

```
#Boucle imbriquée. Une image DECAL est chargée et celle-ci
#puis se déplace ligne par ligne de gauche à droite. A la fin de
#chaque ligne il saute un peu vers le bas
OPEN_DECAL "Target.png" (cible)
$DECAL_X = $DECAL_Y = 0
WHILE $DECAL_Y < 100
     WHILE $DECAL_X < 100
          $DECAL_X += 10
          WAIT 0.02
     ENDWHILE
     $DECAL_X = 0
     $DECAL_Y += 10
ENDWHILE
```

GOTO :MARQUEUR

Lors d'une **commande GOTO**, le programme passe directement à la ligne contenant le marqueur indiqué. Ce marqueur doit exister et il doit être unique. Un marqueur doit être seul sur une ligne. Il est soumis aux mêmes règles de nommage que les variables, sauf qu'il ne commence pas par un signe dollar mais par deux points.

```
GOTO :END       #Saut au marqueur ":END"
$varA = 7       #Cette ligne ne sera pas exécutée
:END
```

Il n'y a pas de limite à la position du marqueur de destination. Avec une **commande GOTO, il** est possible de sortir des boucles ou d'y entrer à n'importe quel endroit. Il est recommandé de n'utiliser la **commande GOTO** qu'avec parcimonie, car elle risque de rendre le déroulement du programme très confus.

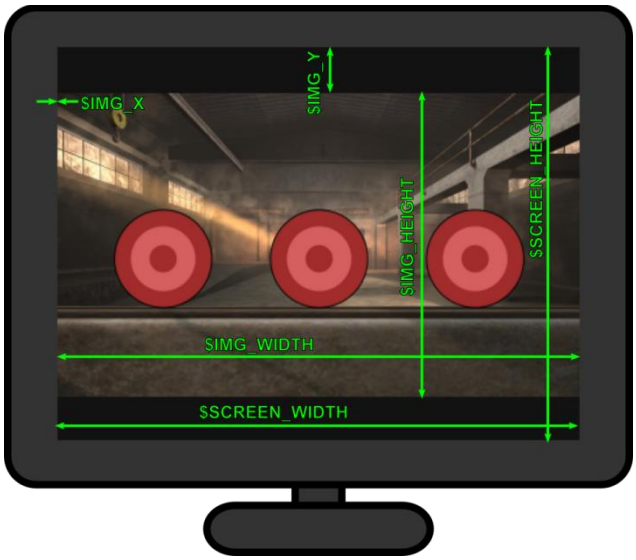
VARIABLES SYSTÈME, DÉCALS ET TEXTE

Les variables système sont des variables par lesquelles le système met à disposition des informations ou par lesquelles on peut manipuler des objets. Elles peuvent être utilisées comme des variables normales, mais elles ne conservent pas nécessairement leur valeur de manière permanente, mais sont toujours remplies avec les valeurs actuelles sans intervention du script. Elles se répartissent grossièrement en 5 catégories. Chacune de ces catégories peut être affichée ou masquée dans l'éditeur de diaporama afin que la liste des variables affichées reste claire.

LES VARIABLES SYSTÈME GÉNÉRALES contiennent des informations sur la taille de l'écran et l'heure.

Variable	Description
$RUNTIME	Temps écoulé depuis le début du diaporama Peut être modifié si nécessaire, puis le temps est décompté à partir de la nouvelle valeur.
$SCREEN_WIDTH	Largeur de la zone d'affichage en pixels (largeur de l'écran)
$SCREEN_HEIGHT	Hauteur de la zone d'affichage en pixels (hauteur de l'écran)
$IMG_WIDTH	Largeur de l'image affichée en pixels
$IMG_HEIGHT	Hauteur de l'image affichée en pixels
$IMG_X	Position X de l'image affichée en pixels par rapport au bord gauche de l'écran
$IMG_Y	Position Y de l'image affichée en pixels par rapport au bord supérieur de l'écran

Lorsqu'un fichier image est ouvert, il est automatiquement mis à l'échelle de manière à remplir le plus possible la zone d'affichage, sans toutefois que son rapport hauteur/largeur ne soit déformé. Si l'image est aussi grande que l'écran ou a au moins le même rapport largeur/hauteur, elle est affichée en plein écran. La hauteur et la largeur de l'image sont alors égales à la hauteur et à la largeur de l'écran et les positions de l'image sont égales à 0. Si le rapport hauteur/largeur ne convient pas, il y a des barres noires soit à gauche et à droite, soit en haut et en bas. Les deux valeurs de position indiquent alors la position du coin supérieur gauche de l'image par rapport au coin supérieur gauche de l'écran. Les variables commençant par "$IMG_" sont automatiquement définies à chaque **commande** OPEN sur les valeurs représentées dans l'image suivante.



Ces variables sont surtout utiles lorsque l'on veut écrire un diaporama de manière à ce qu'il fonctionne avec n'importe quelle résolution d'écran. Dans ce cas, il ne faut pas positionner les objets en fonction de valeurs absolues,

mais en fonction de la résolution d'écran. Ainsi, un objet dont la position X est la moitié de **$SCREEN_WIDTH** et dont la position Y est la moitié de **$SCREEN_HEIGHT** est toujours centré au centre de l'écran.

LES VARIABLES D'ENTRÉE contiennent l'état actuel des touches du clavier. Les variables ont une valeur de 1 lorsque la touche est enfoncée et une valeur de 0 lorsqu'elle ne l'est pas. Il est ainsi possible d'influencer le diaporama également via les saisies clavier. Les touches disponibles sont les touches alphabétiques (sans caractères spéciaux), les touches numériques 0 à 9, les touches F F1 à F12 et la touche Enter. Les touches Echap et Espace sont déjà affectées à des fonctions spéciales par le programme. Comme pour la lecture de vidéos, un diaporama peut être mis en pause à tout moment avec la barre d'espacement et terminé avec la touche d'échappement.

Variable	Description
$KEY_0 ... $KEY_9	État de la touche numérique correspondante (1 = enfoncée) / (0 = non enfoncée)
$KEY_A ... $KEY_Z	État de la touche de lettre correspondante (1 = enfoncée) / (0 = non enfoncée)
$KEY_F1 ... $KEY_F12	État de la touche F correspondante (1 = enfoncée) / (0 = non enfoncée)
$KEY_ENTER	État de la touche Enter (1 = enfoncée) / (0 = non enfoncée)

LES VARIABLES D'IMPACT sont mises à jour à chaque nouveau tir et contiennent des informations sur celui-ci. Le nombre de tirs effectués est compté et la position du dernier tir est mise à disposition. De plus, les informations sur la couleur du pixel touché sont lues, aussi bien dans l'image normale affichée que dans l'image de score. Cette dernière offre ainsi la possibilité d'évaluer les zones d'impact.

Variable	Description
$SHOT_COUNT	Nombre de tirs effectués jusqu'à présent. Peut également être modifié si nécessaire, après quoi le temps est décompté à partir de la nouvelle valeur.
$HIT_TIME	Largeur de la zone d'affichage en pixels (largeur de l'écran)
$HIT_X	Position X de l'impact en pixels par rapport au bord gauche de l'écran
$HIT_Y	Position Y de l'impact en pixels par rapport au bord supérieur de l'écran
$HIT_RED **$HIT_GREEN** **$HIT_BLUE**	Valeurs de couleur du pixel touché dans l'image préalablement ouverte par la commande "OPEN". Les 3 variables représentent les 3 canaux de couleur rouge, vert et bleu. Si l'image n'a pas été touchée ou si aucun tir n'a encore été effectué, les 3 valeurs sont à 0.
$HIT_SCORE_RED **$HIT_SCORE_GREEN** **$HIT_SCORE_BLUE**	Valeurs de couleur du pixel touché dans l'image préalablement ouverte par la commande "OPEN_SCORE". Les 3 variables représentent les 3 canaux de couleur rouge, vert et bleu. Si l'image n'a pas été touchée ou si aucun tir n'a encore été effectué, les 3 valeurs sont à 0.

La position dans l'image de score est calculée à partir de celle dans l'image affichée. Si le résultat dans l'image affichée se trouve par exemple à un tiers de la largeur de l'image du bord gauche et à un quart du bord supérieur, cela vaut également pour la position évaluée dans l'image du score. Cela signifie que l'image affichée et l'image de score ne doivent pas nécessairement être de la même taille ni avoir le même rapport hauteur/largeur. Il est toutefois recommandé d'utiliser des images de même taille, car cela facilite les choses. Les images ci-dessous montrent, à l'aide d'un jeu de cibles tombantes, comment l'évaluation des résultats peut fonctionner via une image de score. Les trois cercles de couleur permettent non seulement de savoir si une cible a été touchée, mais aussi de savoir laquelle.

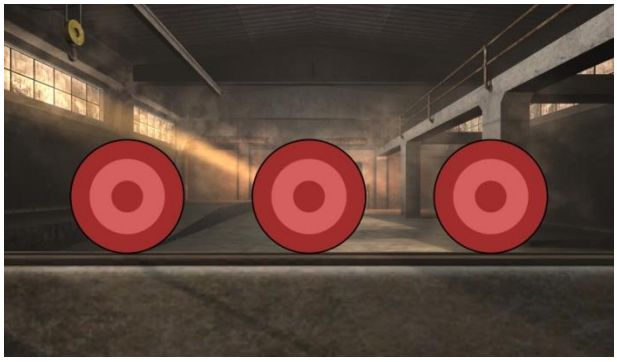


image affichée

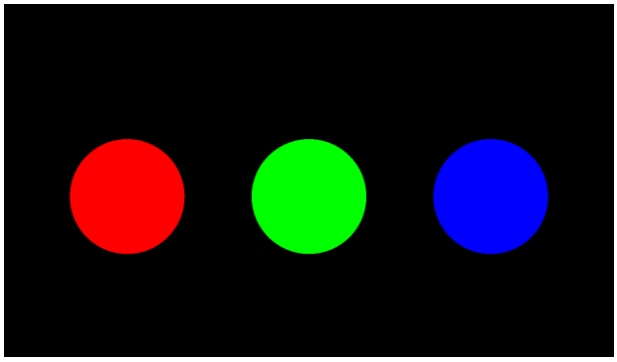


Image du score

Si l'on souhaite évaluer les zones d'impact par des surfaces colorées, il faut veiller à ne pas avoir d'effets d'anticrénelage indésirables lors de la création des images. L'anticrénelage est une technique utilisée par de nombreux programmes de peinture pour rendre les bords plus lisses. Au lieu de dessiner une transition dure, une transition de couleur fluide est créée sur le bord. Cela est bien sûr gênant si l'on veut décider clairement, à partir de la couleur, si un impact se situe à l'intérieur ou à l'extérieur de la surface. Il faut donc être conscient de cet effet et trouver comment le désactiver dans le programme de peinture utilisé ou utiliser un programme correspondant.

sans antialiasing

avec antialiasing

L'OBJET DECAL

Il est possible d'afficher un seul fichier image sur l'arrière-plan en le positionnant et en le redimensionnant librement. Cela est possible grâce à l'objet Decal (Decal = autocollant/décalque en anglais). Tant qu'aucun fichier image n'a été chargé, la décalcomanie n'est pas visible. Le chargement du fichier image s'effectue via la commande "**OPEN_DECAL**" et fonctionne selon les mêmes règles que les deux autres commandes Open. L'image doit se trouver dans le même dossier que le diaporama et peut avoir les formats *.png, *.jpg ou *.bmp. Pour les images PNG, le canal de transparence est également pris en compte, de sorte qu'elles sont le moyen de choix si l'on souhaite afficher des décals non rectangulaires. L'objet Decal est configuré à l'aide d'une série de variables. Les variables **$DECAL_X** et **$DECAL_Y** déterminent la position par rapport au coin supérieur gauche de l'écran et les variables **$DECAL_WIDTH** et **$DECAL_HEIGHT** la taille. Ces deux dernières sont toutefois également modifiées par le système. Après chaque commande OPEN_DECAL, elles contiennent la hauteur et la largeur du fichier image chargé, qui est donc affiché sans distorsion dans sa taille originale. Si ces variables sont modifiées par le script après l'ouverture d'un fichier image, l'image decal est mise à l'échelle à la taille correspondante. Le rapport hauteur/largeur peut également être modifié, ce qui entraîne une déformation de l'image. Les valeurs peuvent bien sûr être calculées en fonction de la résolution de l'écran, de sorte que la décalcomanie ait toujours le même aspect, quelle que soit la résolution. Lors du positionnement de la décalcomanie, le coin supérieur gauche de la décalcomanie sert toujours de point de référence et est positionné par le biais de **$DECAL_X** et **$DECAL_Y**. Pour rendre l'objet decal à nouveau invisible, il suffit d'appeler la commande OPEN_DECAL avec un nom de fichier inexistant (ou vide).

Variable	Description
$DECAL_X	Position X de l'objet DECAL en pixels par rapport au bord gauche de l'écran
$DECAL_Y	Position Y de l'objet DECAL en pixels par rapport au bord supérieur de l'écran
$DECAL_WIDTH **$DECAL_HEIGHT**	Largeur/hauteur du decal. Est fixé à la largeur du fichier image lors de la **commande OPEN_DECAL** et peut ensuite être modifié pour mettre la décalcomanie à l'échelle. Si des valeurs négatives sont utilisées ici, le decal est retourné autour de l'axe correspondant.

L'OBJET TEXTE

Outre la décalcomanie, il est également possible de créer un objet texte et de le positionner librement. Le texte est alors défini par la **commande PRINT**. Comme pour les autres commandes, le mot-clé PRINT peut être suivi d'un simple texte entre guillemets ou d'une combinaison de plusieurs morceaux de texte et de variables qui sont reliés par des signes plus. Les valeurs des variables sont alors insérées dans le texte. Les règles suivantes s'appliquent :

- Si une valeur n'a pas de décimales, seule la valeur avant la virgule est affichée (p. ex. "3").
- Si une valeur a des décimales, elle est toujours éditée avec les trois décimales (p. ex. "12,300").
- Le séparateur est un point
- Si une valeur est négative, elle est précédée du signe moins. Les valeurs positives ne sont pas précédées d'un signe.

Il est également permis d'écrire la **commande PRINT** exclusivement avec une variable ou une valeur fixe, sans les associer à un morceau de texte.

Pour rendre le texte totalement invisible, il suffit d'exécuter la **commande PRINT** avec un texte vide.

La modification du texte ou de la taille de la police entraîne un changement de taille de l'objet texte. C'est pourquoi les variables correspondantes sont alors mises à jour.

La police de caractères est Arial et ne peut pas être modifiée, mais la couleur du texte peut être définie à l'aide de 3 variables.

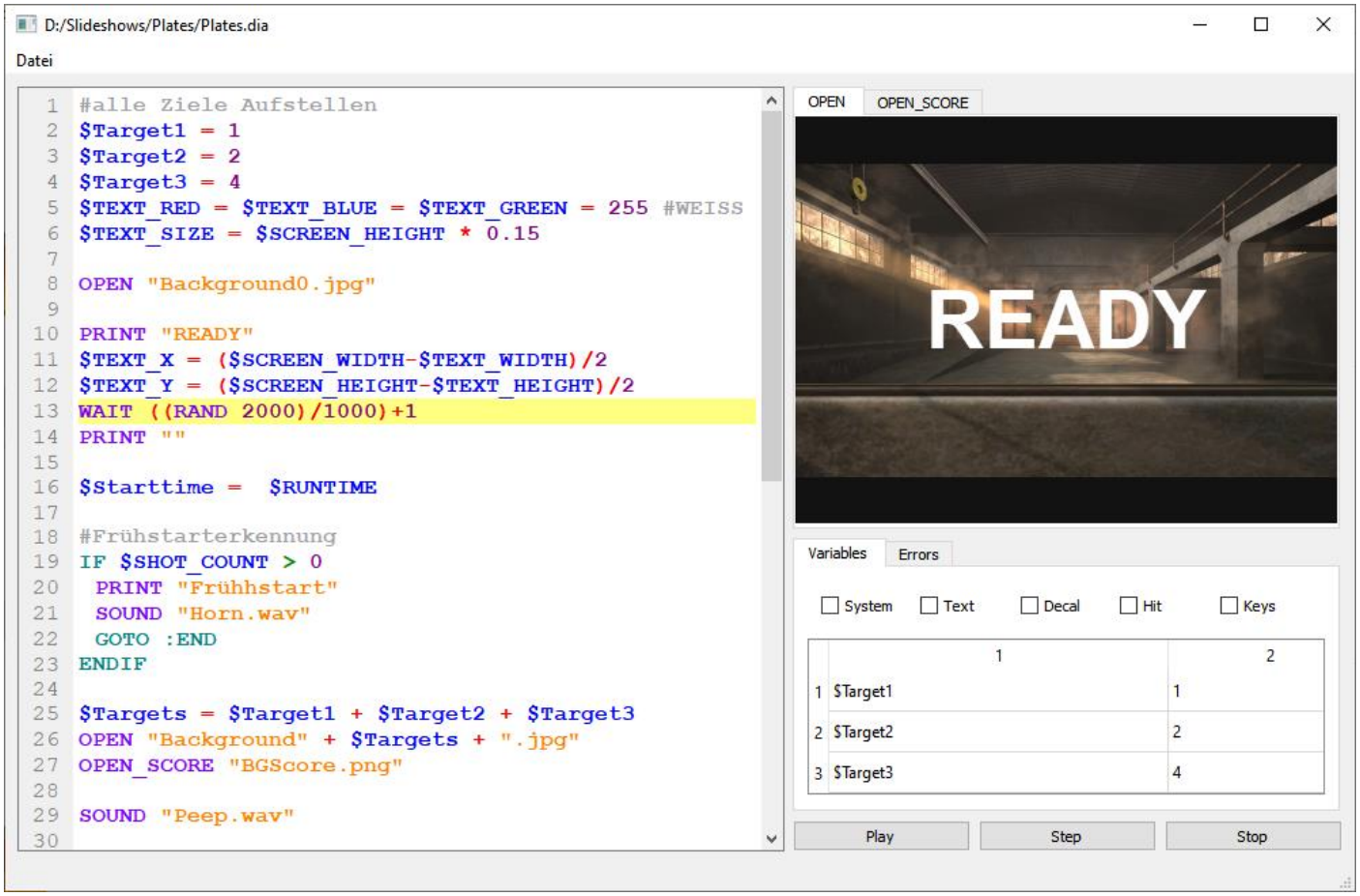
Pour créer des textes de plusieurs lignes, il est possible de créer un saut de ligne avec la combinaison de caractères "\n". Néanmoins, l'ensemble de la **commande PRINT** doit se trouver d'un seul tenant sur une seule ligne du script.

Variable	Description
$SHOT_COUNT	Temps écoulé depuis le début du diaporama Peut être modifié si nécessaire, puis le temps est décompté à partir de la nouvelle valeur.
$TEXT_X	Position X de l'objet texte en pixels par rapport au bord gauche de l'écran
$TEXT_Y	Position Y de l'objet texte en pixels par rapport au bord supérieur de l'écran
$TEXT_SIZE	Taille de la police de l'objet texte. La hauteur de la police est approximativement la valeur de cette variable en pixels
$TEXT_WIDTH **$TEXT_HEIGHT**	Largeur et hauteur de l'objet texte. Elles sont automatiquement mises à jour lorsque le texte ou la taille de la police sont modifiés. Les valeurs peuvent par exemple être utilisées pour positionner l'objet texte de manière centrée à un endroit précis.
$TEXT_RED **$TEXT_GREEN** **$TEXT_BLUE**	Déterminent ensemble la couleur de l'objet texte. Seules les valeurs comprises entre 0 et 255 doivent être utilisées pour chaque canal de couleur.

```
$TEXT_RED = $TEXT_BLUE = $TEXT_GREEN = 255   #Couleur du texte blanc
$TEXT_SIZE = $SCREEN_HEIGHT*0.06             #Taille du texte
PRINT $RUNTIME + " Sec.\N" + $SHOT_COUNT + " Tir"
$TEXT_X = ($SCREEN_WIDTH-$TEXT_WIDTH)/2      #Centrage vertical
$TEXT_Y = $SCREEN_HEIGHT*0.75                #Horizontale 75

#Le code ci-dessus écrit un texte blanc divisé en deux parties.
#Lignes représentant le temps et les tirs effectués
#
# 12.865 sec.
# 14 coup
```

TRAVAILLER AVEC L'ÉDITEUR



L'éditeur aide à écrire les scripts de diaporama. Les fichiers .dia peuvent être chargés et enregistrés via le menu Fichier. On peut également les glisser-déposer dans la fenêtre de l'éditeur. Le bouton "Play/Pause" permet de lire le diaporama dans la fenêtre d'aperçu. En cliquant dans la fenêtre d'aperçu, des tirs sont simulés. En plus de l'image d'arrière-plan visible pour le joueur, qui est chargée par une commande OPEN, le gabarit de points (OPEN-SCORE) peut également être affiché. Il est également possible d'exécuter le diaporama ligne par ligne. Dans ce cas, une marque jaune indique la prochaine ligne à exécuter. Pendant l'exécution, toutes les variables utilisées et leurs valeurs momentanées sont affichées dans la fenêtre des variables, les variables système des différentes catégories pouvant être affichées ou masquées.

Si une erreur se produit lors de l'exécution, le diaporama s'arrête et la ligne correspondante est mise en évidence. Dans l'onglet "Erreurs", une brève description est également affichée pour aider à trouver la cause du problème.

Message d'erreur	Description
Jeton(s) inconnu(s) : "xyz"	La partie "xyz" de la ligne ne peut pas être attribuée à une variable, à une commande ou à un autre mot-clé.
Number Of Brackets	Le nombre de parenthèses ouvrantes et fermantes dans la ligne ne correspond pas
Wrong Paramters Types : "xyz" (mauvais types de parametres)	Dans la partie "xyz", on essaie d'effectuer une opération avec des composants qui ne vont pas ensemble. Par exemple, assigner un texte à une variable (`$VAR = "text"`)
OPEN - File Not Found : "xyz" (fichier non trouvé) **OPEN_SCORE - File Not Found : "xyz" (fichier non trouvé)**	Le fichier "xyz" qui devrait être chargé avec une commande OPEN ou une commande OPEN_SCORE n'a pas été trouvé. Dans cette erreur, le diaporama ne s'arrête pas, mais charge une image standard à la place.