ARES Slideshow

Bedienungsanleitung V1.3

ARES Video bietet die Möglichkeit selbst erstellte "intelligente" Diashows für das Laserschießtraining zu benutzen. In diesen können nicht nur verschiedene Bilder angezeigt werden, sondern auch der Übergang zwischen diesen Bildern kann auf viele Weisen definiert und sogar interaktiv gestaltet werden. Das heißt die Diashows können auf die Schüsse reagieren. Die Shows werden dafür in einer einfachen Scriptsprache geschrieben, die zwar keine Programmierkenntnisse erfordert bei Bedarf aber auch die Möglichkeit für komplexe Abläufe bietet. Um das Erstellen der Diashows zu erleichtern gibt es zudem einen Editor, der es unter anderem ermöglicht, die Diashows schrittweise abzuspielen. Zudem bietet er eine farbliche Hervorhebung von Schlüsselwörtern und zeigt beim Ausführen gegebenenfalls Fehlermeldungen an.

Prinzipieller Aufbau und Funktionsweise der Diashows

Die Diashow-Dateien haben die Endung ".dia". Es handelt sich um einfache Textdateien, die mit jedem Texteditor geöffnet werden können. Beim Abspielen wird die Datei zeilenweise abgearbeitet. Leere Zeilen sind zulässig und werden einfach übergangen. Zudem ist es möglich Kommentare zu schreiben. Diese beginnen mit einem Raute-Symbol (#) welches dafür sorgt, dass aller Text bis zum Ende der jeweiligen Zeile ignoriert wird. In jeder Zeile steht genau ein Befehl wobei dieser beliebig lang sein kann. Das Zeilenende markiert somit auch das Ende eines Befehls. Ein explizites Zeichen, wie es bei manchen anderen Programmiersprachen üblich ist, ist nicht nötig. Die Diashow hört automatisch auf, wenn die letzte Zeile verarbeitet wurde. Ebenso kann Sie jederzeit durch Drücken der Escape-Taste beendet werden.

Grundelemente der Scripte: Befehle, Variablen, und Logik

Die Fähigkeiten der Diashow-Scripte kann man grob in drei Bereiche unterteilen. Es gibt Befehle, Variablen und Logik.

Befehle sind die Kommandos, die das eigentliche Aussehen der Diashow bestimmen. Mit ihnen lassen sich Bilder anzeigen, Text ausgeben, Geräusche abspielen oder Wartezeiten definieren.

Variablen dienen dazu sich Werte zu speichern und damit zu rechnen oder abhängig von den Werten Entscheidungen zu fällen. Man kann selbst beliebig viele eigene Variablen definieren und benutzen. Das System selbst stellt aber auch viele Informationen über Variablen bereit. Zum Beispiel die Trefferposition des letzten Schusses, die Größe verschiedener Elemente oder die seit Beginn vergangene Zeit.

Logik ist der Teil der den Ablauf der Diashows bestimmt und somit die "Intelligenz" liefert. Es gibt Verzweigungen, Schleifen, Sprünge und die logischen Verknüpfungen UND und ODER. Damit ist es möglich abhängig von Variablenwerten verschiedene Teile der Diashow zu durchlaufen.

Übersicht über die Befehle

Alle Befehle sollten am Anfang eine Zeile stehen, da sie zwar eine Aktion bewirken, aber selbst nicht von anderen Befehlen verarbeitet werden können. Die einzige Ausnahme ist der RAND-Befehl dieser kann überall da eingesetzt werden wo auch eine Variable oder eine Zahl stehen könnte.

Befehl	Funktion
OPEN "%Datei%"	Öffnet eine Bilddatei und zeigt sie auf dem Bildschirm an. Die Datei muss sich im selben Ordner wie die Diashow befinden. Es werden nur folgende Formate unterstützt (jpg, png, bmp). Der Dateiname steht in Anführungszeichen. Der Dateiname kann auch aus einer Kombination von Text und Variablen generiert werden. Beispiel: **OPEN "Bild" + $Var + ".jpg"** Wenn die Variable $Var den Wert 1 hat wird die Bilddatei "Bild1.jpg" geöffnet, hat sie den Wert 2 die Datei "Bild2.jpg"
OPEN_SCORE "%Datei%"	Ähnlich wie OPEN jedoch wird die Bilddatei nicht angezeigt sondern im Hintergrund als "Punkteschablone" geöffnet. Idealerweise sollte die Schablone die gleiche Größe haben wie das angezeigte Bild. Nach einem Schuss steht die Farbe der Punkteschablone an der getroffenen Stelle über 3 Variablen zur Verfügung. Somit lassen sich beliebige Trefferzonen definieren, die jedoch für den Schützen nicht sichtbar sein müssen.
OPEN_DECAL "%Datei%"	Wählt die Datei für das Decal-Objekt. Dieses ist ein Bild, das über der Diashow angezeigt werden kann, wobei die Position und die Größe über Variablen frei einstellbar ist. Auch Bewegungen lassen sich durch fortwährendes Ändern der Position realisieren.
PRINT "%Text%"	Setzt den Text des Text-Objekts auf den angegeben Text. Das Textobjekt kann z.B. zum Anzeigen von Meldungen, aber auch Punktezahlen und Zeiten benutzt werden. Wie auch bei den OPEN-Befehlen kann der Text aus einer Kombination von Textstücken und Werten zusammengesetzt werden. Beispiel **PRINT "Zeit: " + $RUNTIME +" Sekunden"** Setzt den Text auf "Zeit: 31,415 Sekunden" wobei die Zahl der aktuelle Wert der Variablen $RUNTIME ist, die die bisherige Laufzeit der Diashow enthält.
SOUND "%Datei.wav%"	Spielt eine Sounddatei im wav-Format ab. Dieser Befehl sollte nicht zu oft hintereinander verwendet werden, sondern es sollte gewartet werden, bis der vorherige Sound abgespielt wurde, bevor der nächste gespielt wird.
WAIT %ZEIT%	Wartet für die angegebene Zeit in Sekunden bevor die nächste Zeile ausgeführt wird. Beispiel: **WAIT 1,5**
RAND %Zahl%	Erzeugt eine Zufallszahl zwischen 0 und der eingegeben Zahl minus 1. Es werden nur ganze Zahlen ohne Kommateil erzeugt. Möchte man Kommazahlen, muss man das Ergebnis entsprechend teilen. Beispiel: **RAND 5** Liefert eine Zufallszahl die 0,1,2,3 oder 4 sein kann. **(RAND 100)/10** Liefert eine Zufallszahl zwischen 0,0 und 9,9

VARIABLEN UND RECHENOPERATIONEN

Eine Variable hat einen Namen und ihr kann ein Wert zugewiesen werden. Der Name der Variablen muss mit einem Dollarzeichen ($) anfangen und darf dann aus beliebig vielen Groß- und Kleinbuschstaben oder Ziffern bestehen. Umlaute sind nicht zulässig und das einzige erlaubte Sonderzeichen ist der Unterstrich (_). Jede Variable wird automatisch angelegt sobald ihr Name das erste Mal auftaucht. Wird ihr dabei nicht bereits ein Wert zugewiesen wird der Wert 0 angenommen. Der Wert ist eine Festkommazahl mit 3 Nachkommastellen. Der kleinste Wert größer

als 0 ist also 0,001. Der größte mögliche Wert ist 2.000.000 (2 Millionen). Wird einer Variablen ein größerer Wert zugweisen, wird dieser automatisch auf diese 2 Mio. begrenzt. Auch negative Werte sind möglich, so dass der gesamte Wertebereich von -2.000.000 bis 2.000.000 geht. Bei Werten ist sowohl der Punkt als auch das Komma als Dezimaltrennzeichen zulässig.

Eine Variable behält ihren Wert bis zum Ende der Diashow bei, sofern ihr nicht explizit ein neuer Wert zugewiesen wird. Die Zuweisung eines Wertes erfolgt über ein einfaches Gleichheitszeichen (=), wobei der Wert rechts des Gleichheitszeichens der Variablen links davon zugewiesen wird. Der Wert rechts kann dabei selbst auch eine Variable sein. Auch Kettenzuweisungen funktionieren.

```
$varA = 5            #Variable $varA hat jetzt den Wert 5
$varB = $varA        #Variable $varB hat jetzt auch den Wert 5
$varA = $varB = 7    #Variable $varA und $varB haben beide den Wert 7
```

Mit Variablen und festen Werten lassen sich die vier Grundrechenarten ausführen. Die Ergebnisse dieser Rechnungen kann man als Eingabe für Befehle nutzen oder sie wiederum Variablen zuweisen um sie so zu speichern. Bei diesen Rechnungen wird die Punkt-vor-Strich-Regel beachtet, wobei auch runde Klammern verwendet werden können um die Reihenfolge der Abarbeitung zu beeinflussen.

```
$varA = 2 + 1,5                #Variable $varA hat den Wert 3,5
$varB = $varA * 2              #Variable $varB hat den Wert 7
$varC = $varB/3                #Variable $varC hat den Wert 2,333
$varD = ($varA - $varC)*2      #Variable $varD hat den Wert 3,334
```

Häufig kommt es vor, dass man den Wert einer Variablen mit einem anderen Wert verrechnen und das Ergebnis wieder derselben Variable zuweisen will. Zu diesem Zweck gibt es eine Kurzschreibweise bei der auf das Symbol für die Rechenoperation ein Gleichheitszeichen folgt.

```
$varA = $varA + 5    #Variable $varA wird um 5 erhöht
$varA += 5           #Variable $varA wird auch um 5 erhöht
$varB -= $varA       #$varB wird um den Wert von $varA verringert
```

Logik

Mit Logikbefehlen lässt sich der Ablauf der Slideshow beeinflussen, indem sie die Ausführung von Befehlen mit Bedingen verknüpfen.

Bedingungen

sind Ausdrücke bei denen Variablen oder Werte miteinander verglichen werden. Das Ergebnis dieses Vergleichs kann wahr oder falsch sein. Folgende Vergleiche sind möglich:

`$varA == $varB`	Gleichheit
`$varA != $varB`	Ungleichheit
`$varA < $varB`	Kleiner
`$varA > $varB`	Größer
`$varA <= $varB`	Kleiner oder Gleich
`$varA >= $varB`	Größer oder Gleich

Man beachte, dass für den Vergleich auf Gleichheit ein doppeltes Gleichheitszeichen verwendet werden muss. Das einfache Gleichheitszeichen wird für die Zuweisung verwendet.

Werte bzw. Variablen können für sich genommen auch eine Bedingung sein. Ist an der Stelle, an der eine Bedingung stehen müsste, nur eine Variable angegeben, so wird die Bedingung als nicht erfüllt gewertet, wenn diese Variable genau 0 ist, ansonsten gilt sie als erfüllt.

Mehrere einzelne Vergleiche können mit den Schlüsselwörtern **AND** und **OR** miteinander verknüpft werden, wobei auch runde Klammern verwendet werden können um die Reihenfolge der Abarbeitung (von innen nach außen) zu bestimmen.

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

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

IF [BEDINGUNG]...ELSE...ENDIF

Mit der **IF**-Konstruktion können Teile des Scriptes ausgeführt werden je nachdem ob eine bestimme Bedingung erfüllt ist. Über den optionalen **ELSE**-Teil kann dann auch ein anderer Teil ausgeführt werden, wenn die Bedingung nicht erfüllt ist. Zur besseren Lesbarkeit sind die Teile in den einzelnen Zweigen im Beispiel eingerückt. Dies ist zulässig und zu empfehlen, aber nicht notwendig.

```
#Beispiel IF ohne ELSE-Zweig
IF $varA < 5
  PRINT "A ist kleiner 5"          #passiert nur wenn A kleiner 5
ENDIF

#Beispiel IF mit ELSE-Zweig
IF $varB == 3
  PRINT "B ist 3"
ELSE
  PRINT "B ist nicht 3"
ENDIF
```

WHILE [BEDINGUNG]...ENDWHILE

Eine While-Schleife wird dazu benutzt einen bestimmten Abschnitt immer wieder zu durchlaufen so lange eine Bedingung erfüllt ist. Wenn der Programmfluss die Zeile mit dem **WHILE** erreicht, wird geprüft ob die Bedingung erfüllt ist. Wenn dies der Fall ist, wird der folgende Programmteil ausgeführt, ansonsten springt der Programmfluss auf die Zeile hinter dem dazugehörigen **ENDWHILE**-Befehl. Wenn das Programm nach dem Ausführen der Befehle die Zeile mit dem **ENDWHILE** erreicht, prüft es, ob die Bedingung immer noch erfüllt ist. Ist das der Fall springt das Programm wieder nach oben und wiederholt den gesamten Teil. Wann immer die Diashow einen solchen Rücksprung an den Anfang einer Schleife vollzieht, macht es automatische eine Pause von 0,01 Sekunden. Dadurch ist es vollkommen unbedenklich Endlosschleifen zu erstellen, also Schleifen deren Bedingung immer erfüllt ist. In vielen anderen "Programmiersprachen" würden solche Endlosschleifen zu schweren Fehlern führen.

```
#Ausgabe eines Countdowns
$Countdown = 3
WHILE $Countdown > 0      #wiederholen bis die Zahl 0 erreicht
  PRINT $Countdown        #Aktuelle Zahl ausgaben
  $Countdown -= 1         #Zahl um 1 reduzieren
  WAIT 1                  #1 Sekunde Warten
ENDWHILE
PRINT ""                  #Text Leeren
```

BRAKE

Brake ist kein eigenes Konstrukt, sondern ein Befehl, der erlaubt eine **WHILE**-Schleife zu verlassen, auch wenn die Bedingung nach ihrem **WHILE**-Befehl noch erfüllt ist. Wenn der Programmablauf innerhalb einer **WHILE**-Schleife auf ein **BREAK** trifft, springt er sofort auf die Zeile nach dem dazugehörigen **ENDWHILE**. Somit lassen sich Schleifen mit mehreren Abbruchbedingungen schaffen. Man kann z.B. eine Endlosschleife erstellen und diese trotzdem wieder verlassen. Je nach bevorzugtem Schreibstil kann das zu kurzen und übersichtlichen Programmen führen.

```
WHILE 1               #Endlosschleife: "1" gilt immer als erfüllt
  IF $RUNTIME > 10
     BREAK            #Nach 10 Sekunden wird die Schleife verlassen
  ENDIF
ENDWHILE
```

Schleifen lassen sich auch ineinander verschachteln wobei ein **BREAK**-Befehl immer nur die innerste der gerade aktiven Schleifen verlässt.

```
#Verschachtelte Schleifen. Es wird ein DECAL-Bild geladen und dieses
#dann zeilenweise von links nach rechts bewegt. Am Ende jeder Zeile
#springt es ein Stück nach unten
OPEN_DECAL "Target.png"
$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 :MARKER

Bei einem **GOTO**-Befehl springt der Programmablauf direkt zu der Zeile mit dem angegebenen Marker. Dieser Marker muss existieren und er muss einmalig sein. Ein Marker muss alleine in einer Zeile stehen. Für ihn gelten dieselben Namensregeln wie für Variablen, jedoch beginnt er nicht mit einem Dollarzeichen sondern mit einem Doppelpunkt.

```
GOTO :END       #Sprung zum Marker ":END"
$varA = 7       #Diese Zeile wird nicht ausgeführt
:END
```

Der Position des Zielmarkers sind keine Grenzen gesetzt. Mit einem **GOTO**-Befehl kann aus Schleifen heraus oder auch an beliebiger Stelle in sie hinein gesprungen werden. Es wird empfohlen den **GOTO**-Befehl nur sehr sparsam zu benutzten, da er die Gefahr birgt der Programmablauf sehr unübersichtlich zu machen.

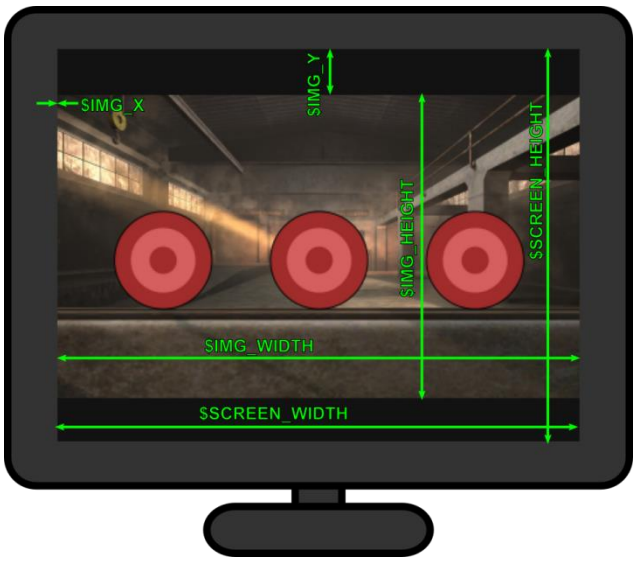
Systemvariablen, Decals und Text

Systemvariablen sind Variablen über die das System Informationen bereitstellt oder über die man Objekte manipulieren kann. Sie lassen sich wie ganz normale Variablen benutzen jedoch behalten sie nicht zwingend dauerhaft ihren Wert, sondern werden ohne Zutun des Skriptes stets mit den aktuellen Werten gefüllt. Sie lassen sich grob in 5 Kategorien einteilen. Jede dieser Kategorien lässt sich im Slideshow Editor ein- bzw. ausblenden, um die Liste der angezeigten Variablen übersichtlich zu halten.

Allgemeine Systemvariablen enthalten Informationen über die Bildschirmgröße und die Zeit

Variable	Beschreibung
$RUNTIME	Zeit, die seit dem Start der Diashow vergangen ist Kann bei Bedarf auch geändert werden wonach die Zeit von dem neuen Wert weitergezählt wird
$SCREEN_WIDTH	Breite des Anzeigebereichs in Pixeln (Bildschirmbreite)
$SCREEN_HEIGHT	Höhe des Anzeigebereichs in Pixeln (Bildschirmhöhe)
$IMG_WIDTH	Breite des Angezeigten Bildes in Pixeln
$IMG_HEIGHT	Höhe des Angezeigten Bildes in Pixeln
$IMG_X	X-Position des Angezeigten Bildes in Pixeln bezogen auf die linke Bildschirmkante
$IMG_Y	Y-Position des Angezeigten Bildes in Pixeln bezogen auf die obere Bildschirmkante

Beim Öffnen einer Bilddatei wird diese automatisch so skaliert, dass sie den Anzeigebereich so weit wie möglich ausfüllt, jedoch ihr Seitenverhältnis nicht verzerrt wird. Ist das Bild also genau so groß wie der Bildschirm oder hat zumindest das selbe Seitenverhältnis, dann wird es vollflächig angezeigt. Die Bildhöhe- und breite sind dann gleich der Bildschirmhöhe- bzw. breite und die Bildpositionen sind 0. Wenn das Seitenverhältnis nicht passt, sind entweder links und rechts oder oben und unten schwarze Balken. Die beiden Positionswerte geben dann die Position der oberen linken Bildecke bezogen auf die obere linke Bildschirmecke an. Die Variablen, die mit "$IMG_" beginnen werden automatisch bei jedem **OPEN**-Befehl auf die Werte gesetzt wie sie im folgenden Bild dargestellt sind.



Diese Variablen sind vor allem dann nützlich, wenn man eine Diashow so schreiben will, dass sie bei beliebigen Bildschirmauflösungen funktioniert. Dann sollte man nämlich Objekte nicht anhand absoluter Werte positionieren, sondern in Abhängigkeit der Bildschirmauflösung. So ist ein Objekt dessen X-Position die Hälfte von

$SCREEN_WIDTH ist und dessen Y-Position die Hälfte von **$SCREEN_HEIGHT** ist immer in der Mitte des Bildschirms zentriert.

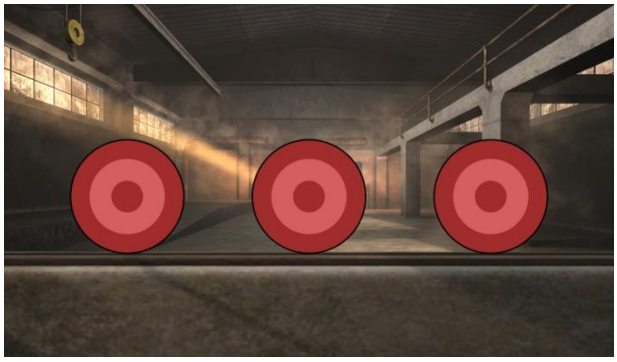
EINGABEVARIABLEN enthalten den aktuellen Zustand von Tastaturtasten. Die Variablen haben den Wert von 1, wenn die Taste gerade gedrückt ist und den Wert 0, wenn sie es nicht ist. Somit ist es möglich die Diashow auch über Tastatureingaben zu beeinflussen. Zur Verfügung stehen die Buchstabentasten (ohne Sonderzeichen), die Zahlentasten 0 bis 9, die F-Tasten F1 bis F12 und die Enter-Taste. Die Escape- und die Leertaste sind schon seitens des Programms mit Sonderfunktionen belegt. Wie auch beim Abspielen von Videos kann eine Diashow mit der Leertaste jederzeit pausiert und mit der Escape-Taste beendet werden.

Variable	Beschreibung
$KEY_0 … **$KEY_9**	Zustand der entsprechenden Zahlentaste (1 = gedrückt) / (0 = nicht gedrückt)
$KEY_A … **$KEY_Z**	Zustand der entsprechenden Buchstabentaste (1 = gedrückt) / (0 = nicht gedrückt)
$KEY_F1 … **$KEY_F12**	Zustand der entsprechenden F-Taste (1 = gedrückt) / (0 = nicht gedrückt)
$KEY_ENTER	Zustand der Enter-Taste (1 = gedrückt) / (0 = nicht gedrückt)

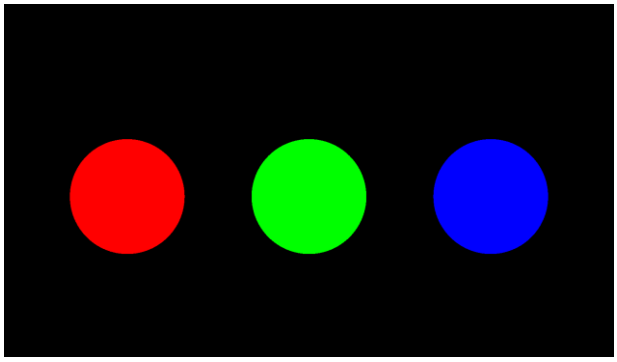
TREFFERVARIABLEN werden bei jedem neuen Schuss aktualisiert und enthalten Information über diesen. Es wird sowohl die Zahl der abgegebenen Schüsse gezählt als auch die Position des letzten Schusses zur Verfügung gestellt. Zusätzlich werden noch die Farbinformationen des getroffenen Pixels ausgelesen, und zwar sowohl im normalen, angezeigten Bild als auch im Score-Bild. Letzteres bietet somit eine Möglichkeit, Trefferzonen auszuwerten.

Variable	Beschreibung
$SHOT_COUNT	Anzahl der bisher abgegeben Schüsse. Kann bei Bedarf auch geändert werden wonach die Zeit von dem neuen Wert weitergezählt wird.
$HIT_TIME	Breite des Anzeigebereichs in Pixeln (Bildschirmbreite)
$HIT_X	X-Position des Treffers in Pixeln bezogen auf die linke Bildschirmkante
$HIT_Y	Y-Position des Treffers in Pixeln bezogen auf die obere Bildschirmkante
$HIT_RED **$HIT_GREEN** **$HIT_BLUE**	Farbwerte des getroffenen Pixels im vorher per "OPEN"-Befehl geöffneten Bild. Die 3 Variablen stehen für die 3 Farbkanäle Rot, Grün und Blau. Wenn das Bild nicht getroffen wurde oder noch kein Schuss abgegeben wurde, sind alle 3 Werte 0.
$HIT_SCORE_RED **$HIT_SCORE_GREEN** **$HIT_SCORE_BLUE**	Farbwerte des getroffenen Pixels im vorher per "OPEN_SCORE"-Befehl geöffneten Bild. Die 3 Variablen stehen für die 3 Farbkanäle Rot, Grün und Blau. Wenn das Bild nicht getroffen wurde oder noch kein Schuss abgegeben wurde sind alle 3 Werte 0.

Die Position im Score-Bild wird aus derjenigen im angezeigten Bild berechnet. Wenn der Treffer im angezeigten Bild also z.B. ein Drittel der Bildbreite vom linken Rand und ein Viertel vom oberen Rand entfernt, dann gilt das auch für die ausgewertete Position im Score-Bild. Das heißt angezeigtes Bild und Score-Bild müssen nicht gleich groß sein und auch nicht dasselbe Seitenverhältnis haben. Allerdings wird empfohlen trotzdem gleich große Bilder zu nehmen, da dies die Dinge einfacher macht. Die Bilder unten zeigen anhand eines Fallscheibenspiels wie die Trefferauswertung über ein Score-Bild funktionieren kann. Anhand der drei farbigen Kreise kann nicht nur erkannt werden, ob eine Scheibe getroffen wurde, sondern sogar welche.



angezeigtes Bild



Score-Bild

Wenn man Trefferzonen durch farbige Flächen auswerten möchte, sollte man beim Erstellen der Bilder darauf achten keine unerwünschten Antialiasing-Effekte zu haben. Antialiasing ist eine Technik die viele Malprogramme nutzen um Kanten glatter aussehen zu lassen. Anstatt einen harten Übergang zu zeichnen wird ein fließender Farbübergang an der Kante erzeugt. Dies ist natürlich störend, wenn man anhand der Farbe eine klare Entscheidung treffen möchte, ob ein Treffer innerhalb oder außerhalb der Fläche liegt. Daher sollte man sich des Effekts bewusst sein und herausfinden wie man ihn beim jeweils benutzen Malprogramm deaktivieren kann bzw. ein entsprechendes Programm benutzen.

ohne Antialiasing

mit Antialiasing

Das DECAL-Objekt

Es ist möglich eine einzelne Bilddatei frei positionier- und skalierbar vor dem Hintergrund anzuzeigen. Dies geht mit dem sogenannte Decal-Objekt (Decal = englisch für Aufkleber/Abziehbild). Solange keine Bilddatei geladen wurde, ist das Decal nicht sichtbar. Das Laden der Bilddatei erfolgt über den Befehl "**OPEN_DECAL**" und funktioniert nach denselben Gesetzmäßigkeiten wie die anderen beiden Open-Befehle. Das Bild muss im selben Ordner wie die Diashow sein und darf die Formate *.png, *.jpg oder *.bmp haben. Bei PNG-Bildern wird auch der Transparenzkanal berücksichtigt, so dass diese das Mittel der Wahl sind, wenn man nicht-rechteckige Decals anzeigen möchte. Das Decal-Objekt wird mit einer Reihe von Variablen konfiguriert. Die Variablen **$DECAL_X** und **$DECAL_Y** bestimmen die Position bezogen auf die obere linke Bildschirmecke und die Variablen **$DECAL_WIDTH** und **$DECAL_HEIGHT** die Größe. Die letzteren Beiden werden allerdings auch vom System verändert. Nach jedem OPEN_DECAL-Befehl enthalten sie Höhe und Breite der geladenen Bilddatei, welche daher unverzerrt in Originalgröße angezeigt wird. Werden diese Variablen allerdings nach dem Öffnen einer Bilddatei vom Script verändert, sorgt das dafür, dass das Decal-Bild auf die entsprechende Größe skaliert wird. Dabei kann auch das Seitenverhältnis verändert werden, wodurch das Bild verzerrt wird. Die Werte können natürlich auch anhand der Bildschirmauflösung berechnet werden, so dass das Decal unabhängig von dieser immer gleich aussieht. Bei der Positionierung des Decals gilt immer die obere linke Ecke des Decals als Referenzpunkt welcher durch setzen von **$DECAL_X** und **$DECAL_Y** positioniert wird. Um das Decal-Objekt wieder unsichtbar zu machen, kann man einfach den OPEN_DECAL-Befehl mit einem nicht vorhandenen (oder leeren) Dateinamen aufrufen.

Variable	Beschreibung
$DECAL_X	X-Position des DECAL-Objektes in Pixeln bezogen auf die linke Bildschirmkante
$DECAL_Y	Y-Position des DECAL-Objektes in Pixeln bezogen auf die obere Bildschirmkante
$DECAL_WIDTH **$DECAL_HEIGHT**	Breite/Höhe des Decals. Wird beim **OPEN_DECAL**-Befehl auf die Breite der Bilddatei gesetzt und kann anschließend geändert werden um das Decal zu skalieren. Werden hier negative Werte verwendet wird das Decal um die entsprechende Achse gespiegelt.

DAS TEXTOBJEKT

Neben dem Decal gibt es auch die Möglichkeit ein Textobjekt zu erstellen und frei zu positionieren. Der Text wird dabei über den **PRINT**-Befehl festgelegt. Wie bei den anderen Befehlen auch kann nach dem PRINT-Schlüsselwort einfach ein Text in Anführungszeichen stehen oder auch eine Kombination mehrerer Textstücke und Variablen die über Pluszeichen zusammengefügt werden. Die Werte der Variablen werden dabei in den Text eingefügt. Dabei gelten folgende Regeln:

- Hat ein Wert keine Nachkommastellen wird nur der reine Vorkomma-Wert ausgegeben (z.B. "3")
- Hat ein Wert Nachkommastellen so wird er immer mit allen 3 Nachkommastellen ausgegeben (z.B. "12,300")
- Das Trennzeichen ist ein Punkt
- Ist ein Wert negativ wird ein Minuszeichen vorangestellt. Positive Werte haben kein Vorzeichen.

Es ist auch zulässig den **PRINT**-Befehl ausschließlich mit einer Variablen oder einem festen Wert zu schreiben, ohne diese mit einem Textstück zu verknüpfen.

Um den Text ganz unsichtbar zu machen kann einfach der **PRINT**-Befehl mit einem leeren Text ausgeführt werden.

Ändern des Textes oder der Schriftgröße führt dazu, dass die Größe des Textobjekts sich ändert. Daher werden die entsprechenden Variablen dabei aktualisiert.

Die Schriftart ist Arial und kann nicht geändert werden, jedoch kann die Textfarbe über 3 Variablen gesetzt werden.

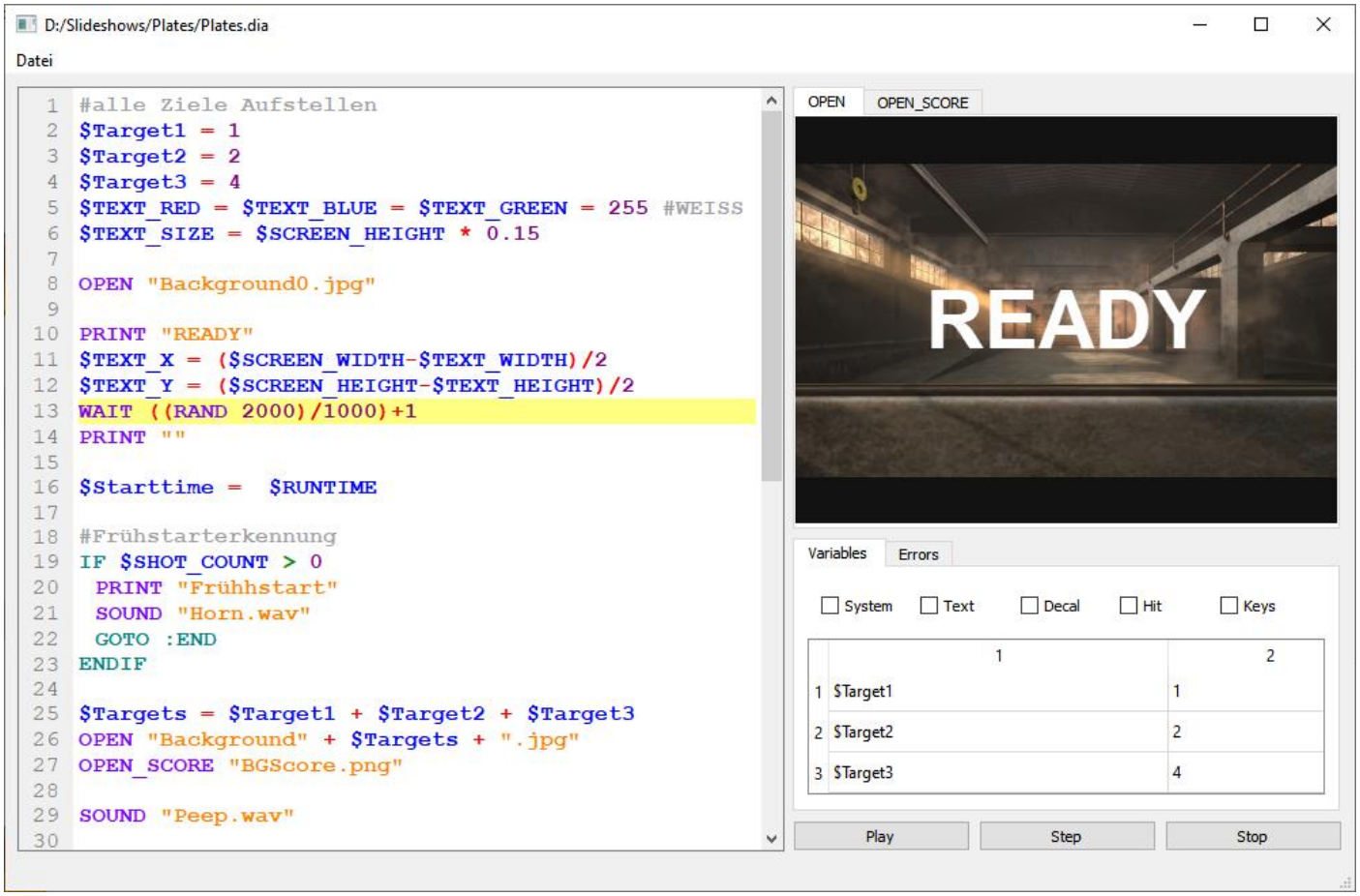
Um mehrzeilige Texte zu erstellen, kann mit der Zeichenkombination "\n" ein Zeilenumbuch erstellt werden. Dennoch muss der ganze **PRINT**-Befehl am Stück in einer Zeile des Scriptes stehen.

Variable	Beschreibung
$SHOT_COUNT	Zeit, die seit dem Start der Diashow vergangen ist Kann bei Bedarf auch geändert werden wonach die Zeit von dem neuen Wert weitergezählt wird
$TEXT_X	X-Position des Text-Objektes in Pixeln bezogen auf die linke Bildschirmkante
$TEXT_Y	Y-Position des Text -Objektes in Pixeln bezogen auf die obere Bildschirmkante
$TEXT_SIZE	Schriftgröße des Textobjektes. Die Höhe der Schrift ist ungefähr der Wert dieser Variablen in Pixeln
$TEXT_WIDTH **$TEXT_HEIGHT**	Breite und Höhe des Textobjekts. Sie werden automatisch aktualisiert, wenn der Text oder die Schriftgröße verändert werden. Die Werte können zum Beispiel benutzt werden um das Textobjekt an einem bestimmt Ort zentriert zu positionieren.
$TEXT_RED **$TEXT_GREEN** **$TEXT_BLUE**	Bestimmen zusammen die Farbe des Textobjekts. Es sollten für jeden Farbkanal nur Werte zwischen 0 und 255 verwendet werden.

```
$TEXT_RED = $TEXT_BLUE = $TEXT_GREEN = 255   #Textfarbe Weiss
$TEXT_SIZE = $SCREEN_HEIGHT*0.06             #Textgröße
PRINT $RUNTIME + " Sek.\n" + $SHOT_COUNT + " Schuss"
$TEXT_X = ($SCREEN_WIDTH-$TEXT_WIDTH)/2      #Vertikal Zentrieren
$TEXT_Y = $SCREEN_HEIGHT*0.75                #Horizontal 75%

#Der oben stehende Code schreibt einen weißen Text der in zwei
#Zeilen die Zeit und die abgegebenen Schüsse darstellt
#
#    12.865 Sek.
#    14 Schuss
```

Arbeiten mit dem Editor



Der Editor hilft beim Schreiben der Diashowscripte. Über das Dateimenü können .dia Dateien geladen und gespeichert werden. Man kann sie aber auch einfach per Drag and Drop ins Editorfenster ziehen. Mit dem Button "Play/Pause" kann die Diashow im Vorschaufenster abgespielt werden. Durch Klicken ins Vorschaufenster werden dabei Schüsse simuliert. Neben dem für den Spieler sichtbaren Hintergrundbild, welches mit einem OPEN-Befehl geladen wird kann auch die Punkteschablone (OPEN-SCORE) angezeigt werden. Es ist zudem möglich die Diashow Zeilenweise abzuarbeiten. In diesem Fall zeigt eine gelbe Markierung die als nächstes ausgeführte Zeile an. Während der Ausführung werden im Variablenfenster alle verwendeten Variablen und ihre momentanen Werte angezeigt wobei die Systemvariablen der verschiedenen Kategorien ein- und ausgeblendet werden können.

Wenn es beim Ausführen zu einem Fehler kommt, hält die Diashow an und die entsprechende Zeile wird markiert. In der Registerkarte "Errors" wird zudem eine kurze Beschreibung angezeigt die hilft die Ursache des Problems zu finden.

Fehlermeldung	Beschreibung
Unknown Token(s):"xyz"	Der Teil "xyz" in der Zeile kann weder einer Variable, einem Befehl oder einem sonstigen Schlüsselwort zugewiesen werden.
Number Of Brackets	Die Anzahl an öffnenden und schließenden Klammern in der Zeile stimmt nicht überein
Wrong Paramters Types: "xyz"	Im Teil "xyz" wird versucht eine Operation mit nicht zusammen passenden Komponenten durchzuführen. Zum Beispiel einer Variable einen Text zuzuweisen (**`$VAR = "text"`**)
OPEN - File Not Found: "xyz" **OPEN_SCORE - File Not Found: "xyz"**	Die Datei "xyz" die mit einem OPEN-Befehl bzw. OPEN_SCORE-Befehl geladen werden sollte, wurde nicht gefunden. Bei diesem Fehler hält die Diashow nicht an, sondern lädt stattdessen ein Standartbild