NAME

akfavatar-graphic − Modul für die Anzeige von Grafiken unter Lua-AKFAvatar

SYNTAX

local graphic = require "akfavatar-graphic"

BESCHREIBUNG

Modul für Lua-AKFAvatar um Grafiken anzuzeigen.

Hinweise:

- Koordinaten fangen mit 1, 1 in der oberen linken Ecke an.
- Man kann es auch nach Art der Turtle-Grafik verwenden.
- Die Stiftbreite, Position, Ausrichtung, Stiftfarbe und Hintergrundfarbe sind für jede Grafik einzeln festgelegt.
- Der Code ist nicht auf Geschwindigkeit optimiert. Man kann zwar Animationen machen, aber es werden keine fließenden Bewegungen.

Funktionen / Methoden

graphic.new(*[Breite, Höhe] [, Hintergrundfarbe]***)**

Erzeugt eine neue Grafik (eine Leinwand).

Wenn die *Breite* und *Höhe* nicht angegeben wird, wird das gesamte Fenster bzw. Bildschirm verwendet.
Es wird ein dünner, schwarzer Stift ausgewählt und die Position ist in der Mitte, nach oben ausgerichtet.
Wenn die *Hintergrundfarbe* nicht angegeben ist, wird diese von AKFAvatar übernommen.

Gibt die Grafik, die Breite und Höhe zurück (dh. drei Werte).

Beispiel:

```
local gr, width, height = graphic.new()
```

graphic.fullsize()

Gibt die erforderliche Breite und Höhe zurück, so dass das Bild das gesamte Fenster füllt.

graphic.set_resize_key(*Taste***)**

Setzt einen Tasten-Code (als Zahl), die ausgegeben werden soll, wenn die Fenstergröße verändert wurde. Gibt den vorherigen Tasten-Code zurück.

Hinweis: Man kann einen Wert zwischen 0xE000 und 0xE9FF verwenden, um Konflikte mit echten Tasten-Codes zu vermeiden.

graphic.set_pointer_buttons_key(*Taste***)**

Setzt einen Tasten-Code (als Zahl), die ausgegeben werden soll, wenn eine Maustaste gedrückt wurde. Gibt den vorherigen Tasten-Code zurück.

Hinweis: Man kann einen Wert zwischen 0xE000 und 0xE9FF verwenden, um Konflikte mit echten Tasten-Codes zu vermeiden.

graphic.set_pointer_motion_key(*Taste***)**

Setzt eine Tasten-Code (als Zahl), die ausgegeben werden soll, wenn die Maus bewegt wurde. Gibt den vorherigen Tasten-Code zurück.

graphic.get_pointer_position()

Gibt die x und y Koordinaten der Maus-Position zurück, relativ zur zuletzt gezeigten Grafik.

*gr***:show()**

Zeigt die Grafik als Bild nach dem Zeichnen.

Hinweis: Man kann auch Zwischenschritte zeigen, dann weiter zeichnen, eine Weile warten und dann den nächsten Schritt zeigen und so weiter. Dadurch erzeugt man eine kleine Animation. Man sollte die Schritte aber nicht zu klein wählen, sonst wird es zu langsam auf langsamen

Geräten.

gr:**size()**
Gibt die Breite und Höhe der Grafik *gr* zurück.

gr:**width()**
Gibt die Breite der Grafik *gr* zurück.

gr:**height()**
Gibt die Höhe der Grafik *gr* zurück.

gr:**color(***Farbname***)**
Legt die Zeichenfarbe fest.
Die Farbe kann entweder ein vordefinierter englischsprachiger Name sein, oder eine hexadezimale Angabe, wie zum Beispiel *0xFFFF00* oder *"#FFFF00"*.

gr:**rgb(***rot, grün, blau***)**
Legt die Zeichenfarbe anhand von RGB-Werten fest.
Die Werte müssen im Bereich von 0 bis 255 inklusive sein.
Diese Methode ist schneller, als **gr:color()** zu verwenden.

gr:**eraser()**
Setzt die Zeichenfarbe auf die Hintergrundfarbe (Radiergummi).

gr:**thickness(***Wert***)**
Legt die Dicke des Stiftes fest. Der *Wert* 1 ist der dünnste.

gr:**clear(***[Farbe]***)**
Löscht die Grafik.
Wenn eine *Farbe* angegeben ist, wird diese zur neuen Hintergrundfarbe.
Die Stift-Position wird nicht verändert.

gr:**border3d(***[gedrückt]***)**
Zeichnet einen 3D-Rahmen um die Grafik. Die Farbe basiert auf der Hintergrundfarbe der Grafik.
Die Einstellung der Stift-Dicke wird ignoriert, der Rahmen ist immer 3 Pixel dick.

Wenn *gedrückt* auf **true** gesetzt wird, wird der Rahmen nach innen gedrückt dargestellt.

Die Stift-Position wird nicht verändert.

gr:**putpixel(***[x, y]***)**
Setzt einen Pixel an den angegebenen Koordinaten, oder bei der aktuellen Stift-Position.
Die Stift-Position wird nicht verändert.

gr:**getpixel(***[x, y]***)**
Gibt die Pixelfarbe an den angegebenen Koordinaten oder bei der aktuellen Stift-Position zurück.
Die Farbe wird als String in hexadezimaler RGB-Schreibweise zurückgegeben. Im Fehlerfall wird *nil* und eine Fehlermeldung zurückgegeben.
Die Stift-Position wird nicht verändert.

gr:**getpixelrgb(***[x, y]***)**
Gibt die Pixelfarbe an den angegebenen Koordinaten oder bei der aktuellen Stift-Position zurück.
Es werden drei Ganzzahlen für rot, grün und blau, im Bereich von 0 bis 255 zurückgegeben.
Im Fehlerfall wird *nil* und eine Fehlermeldung zurückgegeben.
Die Stift-Position wird nicht verändert.

gr:**putdot(***[x, y]***)**
Setzt einen Punkt an den angegebenen Koordinaten, oder bei der aktuellen Stift-Position.
Wenn der Stift dünn ist, ist es das gleiche wie **gr:putpixel()**.
Die Stift-Position wird nicht verändert.

gr:**pen_position()**
Gibt die x- und y-Position des Stiftes zurück (dh. es werden zwei Werte zurückgegeben).

gr:**center()**
gr:**home()**
Setzt den Stift auf die Mitte der Grafik, nach oben ausgerichtet.

gr:**moveto(***x, y***)**
Bewegt den Stift nach *x, y* ohne zu zeichnen.

gr:**moverel(***x, y***)**
Bewegt den Stift ohne zu zeichnen relativ zu seiner aktuellen Position.
Ein postitiver *x*-Wert bewegt ihn nach rechts,
ein negativer *x*-Wert bewegt ihn nach links.
Ein postitiver *y*-Wert bewegt ihn nach unten,
ein negativer *y*-Wert bewegt ihn nach oben.

gr:**lineto(***x, y***)**
Zeichnet eine Linie von der aktuellen Stift-Position zu den absoluten Koordinaten.
Der Stift wird zu den neuen Koordinaten bewegt.

gr:**linerel(***x, y***)**
Zeichnet eine Linie relativ zur aktuellen Stift-Position.
Ein postitiver *x*-Wert zeichnet nach rechts,
ein negativer *x*-Wert zeichnet nach links.
Ein postitiver *y*-Wert zeichnet nach unten,
ein negativer *y*-Wert zeichnet nach oben.
Der Stift wird dabei zu den neuen Koordinaten bewegt.

gr:**line(***x1, y1, x2, y2***)**
Zeichnet eine Linie von *x1, y1* nach *x2, y2*.
Der Stift wird auf *x2, y2* gesetzt.

gr:**bar(***x1, y1, x2, y2***)**
Malt einen ausgefüllten Balken mit *x1, y1* als obere linke Ecke und *x2, y2* als untere rechte Ecke.
Die Stift-Position wird nicht verändert.

gr:**rectangle(***x1, y1, x2, y2***)**
Zeichnet ein Rechteck mit *x1, y1* als obere linke Ecke und *x2, y2* als untere rechte Ecke.
Die Stift-Position wird nicht verändert.

gr:**arc(***Radius [, Winkel1] [, Winkel2]***)**
gr:**circle(***Radius [, Winkel1] [, Winkel2]***)**
Zeichnet einen Kreis oder Bogen mit dem angegebenen *Radius*.
Die Stift-Position markiert den Mittelpunkt.

Man kann einen Teil des Kreises (einen Bogen) zeichnen, indem man einen oder zwei Winkel in Grad angibt. Wenn zwei Winkel angegeben sind, dann wird im Uhrzeigersinn vom ersten Winkel bis zum zweiten gezeichnet. Wenn nur ein Winkel angegebene ist, verwendet es die Stift-Ausrichtung als Anfangswinkel (siehe unten bei **Turtle-Grafik**).

gr:**disc(***Radius [, x, y]***)**
Zeichnet eine Scheibe, dh. einen gefüllten Kreis mit dem angegebenen *Radius* und den angegebenen Koordinaten als Mittelpunkt. Sind keine Koordinaten angegeben, wird die aktuelle Stift-Position als Mittelpunkt verwendet.
Die Stift-Position wird nicht verändert.

gr:**text(***Text [, x, y]***)**
Schreibt einen Text, ausgerichtet an der angegebenen Position oder der Stift-Position.

Standardmäßig wird der Text an der Postion zentriert. Aber das kann man mit **gr:textalign()** ändern.

Die Kodierung wird von den AKFAvatar-Einstellungen übernommen. Andererseits werden keine anderen der dortigen Einstellungen verwendet. Die Farbe ist die Zeichenfarbe für die Grafik. Es

gibt momentan keine einfache Möglichkeit für Fettdruck, Unterstreichen oder inverse Dastellung.

Man kann sämtliche darstellbaren Zeichen verwenden, aber Steuerzeichen werden nicht unterstützt, nichtmal ein Zeilenumbruch.
Die Stift-Position wird nicht verändert.

*gr***:textalign(***[horizontal] [, vertikal]***)**
Legt die Ausrichtung für **gr:text()** fest.

Die horizontale Ausrichtung kann eines von "left", "center" oder "right" sein. Die Vorgabe ist "center".

Die vertikale Ausrichtung kann eines von "top", "center" oder "bottom" sein. Die Vorgabe ist "center".

Die Ausrichtung bedeutet, wo der Bezugspunkt ist, zum Beispiel wenn man angibt, dass es links ausgerichtet sein soll, ist der Bezugspunkt links, aber der Text läuft nach rechts.

graphic.font_size()
*gr***:font_size()**
Gibt die Breite, Höhe und die Grundlinie eines Zeichens zurück. Es handelt sich um einen Zeichensatz mit fester Breite, jedes Zeichen hat die selbe Breite.

*gr***:put(***Grafik [, x, y]***)**
Kopiert eine Grafik auf die Grafik *gr* bei der angegebenen Position (obere linke Ecke). Wenn keine Position angegeben ist, wird es in die obere linke Ecke gesetzt. Der vorherige Inhalt wird überschrieben (keine Transparenz).

Eine Grafik mit der selben Breite und ohne Angabe einer Position zu kopieren ist äußerst effizient. Das selbe gilt wenn die Grafik die selbe Breite hat und die *x*-Position auf 1 gesetzt ist.

*gr***:put_transparency(***Grafik [, x, y]***)**
Legt eine Grafik über die Grafik *gr* bei der angegebenen Position (obere linke Ecke).
Wenn keine Position angegeben ist, wird es in die obere linke Ecke gesetzt.
Pixel mit der Hintergrundfarbe werden nicht kopiert, die sind transparent.
Dies ist wesentlich langsamer als **gr:put()**.

*gr***:put_file(***Dateiname [, x, y]***)**
Kopiert eine Grafik aus einer Datei auf die Grafik *gr* bei der angegebenen Position (obere linke Ecke).
Wenn keine Position angegeben ist, wird es in die obere linke Ecke gesetzt.

*gr***:put_image(***Daten [, x, y]***)**
Kopiert eine Grafik aus den *Daten* auf die Grafik *gr* bei der angegebenen Position (obere linke Ecke). *Daten* kann ein String mit den Bilddaten sein, oder eine Tabelle mit Strings von XPM-Daten.
Wenn keine Position angegeben ist, wird es in die obere linke Ecke gesetzt.

*gr***:get(***x1, y1, x2, y2***)**
Gibt einen Bereich der Grafik *gr* als neue Grafik zurück.
Die meisten Einstellungen werden ebenfalls kopiert, mit Ausnahme der Größe und der Stift-Position.
Der Stift wird auf die Mitte der neuen Grafik gesetzt, nach oben ausgerichtet.
Alle Werte müssen im gültigen Bereich liegen.

*gr***:duplicate()**
Gibt ein exaktes Duplikat (eine Kopie) der Grafik *gr* zurück.
Die Grafik-spezifischen Einstellungen werden ebenfalls kopiert.
Das geht schneller als mit **gr:get()**.

Man kann dies zum Beispiel verwenden, um erstmal einen festen Hintergrund zu zeichnen und dann ein Duplikat zu erstellen. Darauf zeichnet man dann einen Vordergrund. Dann kann man den Hintergrund wieder darauf kopieren (**gr:put**) und einen neuen Vordergrund zeichnen.

gr:**shift_vertically(***Zeilen***)**
Verschiebt die Grafik vertikal.
Ein positiver Wert für *Zeilen* schiebt sie nach unten.
Ein negativer Wert für *Zeilen* schiebt sie nach oben.
Der Stift wird mitbewegt.

gr:**shift_horizontally(***Spalten***)**
Verschiebt die Grafik horizontal.
Ein positiver Wert für *Spalten* schiebt sie nach rechts.
Ein negativer Wert für *Spalten* schiebt sie nach links.
Der Stift wird mitbewegt.

gr:**export_ppm(** *filename***)**
Exportiert die Grafik als Portable Pixmap (PPM) Datei.

Das PPM-Format ist simpel zu implementieren, aber nicht sehr effizient. Man kann aber die „netpbm"-Werkzeuge oder „ImageMagick" verwenden, um es in ein anderes Format umzuwandeln.

Das folgende Beispiel zeigt, wie man das macht:

```
function export(graphic, name)
  graphic:export_ppm(name..".ppm")
  if os.execute("pnmtopng "..name..".ppm > "..name..".png")
     or os.execute("convert "..name..".ppm "..name..".png") then
    os.remove(name..".ppm")
  end
end
```

Zunächst exportiert es die Grafik ins PPM-Format. Dann versucht es dieses ins PNG-Format umzuwandeln. Falls das gelingt, wird die PPM-Datei gelöscht. Wenn der Anwender „netpbm" oder „ImageMagick" aber nicht installiert hat, hat er am Ende immer noch die PPM-Datei.

Turtle-Grafik
Um Turtle-Grafik („Schildkröten-Grafik", manchmal auch „Igel-Grafik" genannt) zu verstehen, muss man sich eine Schildkröte verstellen, die einen Stift trägt. Man kann der Schildkröte dann Befehle erteilen in welche Richtung sie sich drehen soll und wie weit sie sich zu bewegen hat.

gr:**heading(***Ausrichtung***)**
Legt die Ausrichtung der Schildkröte fest. Der Wert muss in Grad angegeben werden und die Schildkröte dreht sich im Uhrzeigersinn. Der Wert 0 bedeutet, sie ist nach oben ausgerichtet, 90 bedeutet nach rechts.

gr:**get_heading()**
Gibt die Ausrichtung der Schildkröte zurück (siehe **gr:heading()**).

gr:**right(***Winkel***)**
Drehe die Schildkröte im Uhrzeigersinn um den angegebenen *Winkel* in Grad.

gr:**left(***Winkel***)**
Drehe die Schildkröte gegen den Uhrzeigersinn um den angegebenen *Winkel* in Grad.

gr:**draw(***Schritte***)**
Zeichne eine Linie in die Richtung, in die die Schildköte ausgerichtet ist.

gr:**move(***Schritte***)**
Bewege die Schildkröte in die Richtung in der sie ausgerichtet ist, ohne zu zeichnen.

gr:**home()**
Setzt die Schildkröte in die Mitte der Grafik, nach oben ausgerichtet.

SIEHE AUCH

lua-akfavatar(1) **lua**(1) **lua-akfavatar-ref**(3) **akfavatar-term**(3) **akfavatar.utf8**(3)