Befehle Grafik

Die C++ Grafiken des Typs 'Script xxxx' stellen den Scripten neben den Turtlebefehle auch die folgenden Grafikbefehle zur Verfügung.

Hilfedateien		
	Scripte:	2_Scripte.pdf
	Befehle Turtle:	3_Turtle.pdf
	Farben:	5_Farben.pdf

Befehle mit optionalen Parametern		
	Befehl:	t.drawRaster(Einheiten=10, Marken=0.6);
	Mögliche Aufrufe:	t.drawRaster( );
		t.drawRaster(15.5);
		t.drawRaster(15.5, 0.8);

Basisbefehle für Scriptgrafiken

Aufruf von Scriptgrafiken

---

var Grafik="Script Name"; Aufruf der C++ Grafik "Script Name". Die möglichen Grafiknamen findet man in der Glalerie unter 'Grafiken für Scripte'. Die Grafik ruft dann die Scriptfunktionen init() und draw() auf.

"Script Turtle"	Basisbefehle. Siehe Galerie 'Grafiken für Scripte'.
"Script Funktionen"	Basisbefehle und Zeichenbefehle für Funktionsgrafen.
"Script Escher"	Basisbefehle und Morphing für 'Eigene Befehle'
"Script Lindenmayer"	Basisbefehle und Unterstützung für Lindenmayer-Systeme.

---

var Grafik="präfix:pfad/Name#Marke; präfix:pfad/Name#Marke; präfix:pfad/Name#Marke; ...";

Aufrufliste für Scripte mit Grafiknamen. Bei mehr als einem Script, wird die Rundgang-Steuerung aufgerufen.
Beim Aufruf einer Aufrufliste wird die laufende Aufrufliste durch die neue ersetzt.

Grafikname	präfix:pfad/Name#Marke;	
	präfix:	Optional. Abkürzung für bestimmte Ordner. Siehe Galerie und Einstellungen.
		'eigene:' eigene Scripte.
		'demo:' schreibgeschützte Scripte.
		'c++:' C++ Grafik.
		'pdf:' PDF-Verzeichnis.
		'C:' Laufwerk.
	pfad/	Optional. Relative- oder absolute Pfadangaben.
	Name	Dateiname oder PDF-Datei.
	#Marke	Optional. Scriptmarke, siehe Scripte.
	;	Strichpunkt obligatorisch!

Beispiele:		
	"eigene:escher/muster.txt#4;"	Script #4 in Datei muster.txt im Unterordner escher/ von eigene:
	"./escher/muster.txt#4;"	Angabe relativ wie oben, wenn das Script in eigene: liegt.
	"C:/Bilder/muster.txt;"	Absolute Angabe. Erstes Script.
	"c++:Farbmischung;"	C++ Grafik aufrufen. Die Namen findet man in der Galerie.

Koordinatensystem

---

t.drawKoordSystem(Beschriftung=true, Einheiten=10, Marken=0.6); Koordinatensystem zeichnen.
t.drawRaster(Einheiten=10, Marken=0.6); Raster zeichnen.

Eigene Befehle

Die turtle kann Punkte, Polygon-Objekte und Funktionen unter 'Eigene Befehle' abspeichern.

Verwendung:		
	Funktionen:	Werden so ausgeführt. wie sie definiert wurden.
	Punkte, Polygon-Objekte:	Verwenden immer die Abbildungsmatix drawMatrix. Danach wird auf die Turtleposition verschoben und in Turtlerichtung gedreht. Position und Richtung der Turtle werden nicht gändert.

Der Recorder zeichnet Befehlsfolgen als Funktion auf. Er kann auch zum Debuggen verwendet werden.

t.recDraw("Name"); "Name": Recorder einschalten. Alle folgenden Befehle werden als function Name() { ...} in die Befehlsliste 'Eigene Befehle' eingetragen.
"": Recorder auschalten. Wiedergabe mit t.draw("Name");

Mir defDraw können verschachtelte Polygon-Objekte definiert werden.

t.defDraw("Name", mode=0); Die Turtledaten Poygon, Pen, Brush, FillMode unter "Name" in 'Eigene Befehle' eintragen.

mode=0: Befehl "Name" neu anlegen. Turtledaten speichern.
mode=1: Turtledaten an bestehenden Befehl "Name" anhängen.
mode=2: Turtledaten an bestehenden Befehl "Name" anhängen. DasTurtlepolygon wird vor dem Speichern am ersten Befehlspolygon geklippt.
mode=3: Offene Polygone: Das Turtlepolygon ins das erste Befehlspolygon einpassen. Die Anfangs- und Endpunkte der Polygone bestimmen die Abbildung. Geschlossene Polygone werden nur verschoben.

---

t.draw("Name", UseStyle=true); Befehl "Name" zeichnen.
recDraw-Befehl werden als Funktion ausgeführt.

defDraw-Befehle: Verwenden immer zuerst die Abbildungsmatix drawMatrix.
Danach eine Schiebung auf Last() und eine Drehung mit lastW().
Position und Richtung der Turtle werden nicht gändert.
UseStyle=true: Style von "Name" verwenden
UseStyle=false: Style von Turtle verwenden

t.drawName("Name", dx=0,dy=0); Beschriftung "Name" ausgeben. Die Position ist der erste Punkt von draw("Name") mit der Verschiebung (dx,dy).

---

t.defPoint("Name"); Turtleposition Last() als Befehlspunkt "Name" merken.
t.defPoint("Name", Index); Punkt Index des Turtlepoygons als Punkt "Name" speichern.
Index ist 0,1, bis n oder -1,-2, bis -(n+1). 0 ist erster Punkt, -1 letzter Punkt.
t.defPoint("Name", "P1", "P2", t); Teilungspunkt "P1" + t*("P2"-"P1") als "Name" speichern.
t.defVector("Name", "P1", "P2",t=1); Ortsvektor t*("P2"-"P1") als Punkt "Name" speichern.
t.defLine("Name", l); Strecke der Länge l von Last() in Richtung LastW() als Polygon "Name" speichern.
t.defLine("Name", "P1", "P2"); Strecke "P2" "P1" als Polygon "Name" speichern.

---

bool t.defScript("Flag","Grafikname"); "Flag" ist in der Befehlsliste: Rückgabe true.
"Flag" ist nicht in der Befehlsliste: Rückgabe false.
1. Script "Grafikname" ausführen
2."Flag" in die Befehlsliste eintragen.
3. Aufrufendes Script neu ausführen.
Die in "Grafikname" definierten Befehle stehen dann im Script zur Verfügung.

t.defCall("Name","Definition", mode=0); Funktionsdefinition "Definition" unter "Name" ist in der Befehlsliste eintragen.
mode=0: Befehl "Name" neu anlegen.
mode=1: "Definition" an bestehenden Befehl anhängen.

t.call("Name"); Funktion "Name" ausführen. Alias für t.draw("name").

---

t.msgDraw(Befehl=""); "": Alle Befehle aus 'Eigene Befehle' anzeigen.
"Name": Daten zum Befehl "Name" anzeigen.
bool t.exists("Name"); Rückgabe true: Befehl "Name" existiert.

---

t.clrDraw("Name"); Befehl "Name" aus 'Eigene Befehle' löschen.
t.clrDraw(); Alle 'Eigenen Befehle' löschen.
t.clrDraw("0"); Datenbank 'Eigenen Befehle' bereinigen.

---

t.setPolygon(Start=0, Anzahl=-1); Turtle-Polygons durch eine Abbildung mit drawMatrix ersetzen.
Start = 0, Anzahl =-1 : Alle Punkte verwenden.
Start = -1, Anzahl =-1: Punkte in umgekehrter Reihenfolge verwenden.
Start >= 0, Anzahl>=0: Anzahl Punkte ab Start verwenden. Der erste Punkt hat den Index 0.
Start <-1, Anzahl>=0: Anzahl Punkte ab Start in umgekehrter Reihenfolge verwenden. Der letzte Punkt hat den Index -1.

t.addPolygon("Name", mode=0); Turtle-Polygon erweitern. Das erste Polygon von "Name" mit drawMatrix abbilden und zum Endpunkt des Turtle-Polygons verschieben.
Punkte aus "Name" in gegebener Reihenfolge ans Turtle-Polygon anhängen.

mode -1: Punkte in umgekehrter Reihenfolge anhängen.

t.setPolygon("Name", mode=0,);
Turtle-Polygon aus dem ersten Polygon von "Name" erzeugen.
"Name" wird zuerst mit drawMatrix abgebildet.
mode = 0: Turtle-Polygon durch "Name" ersetzen.
mode = -1: Turtle-Polygon durch "Name" ersetzen. Punkte in umgekehrter Reihenfolge.
mode = 1: Turtle-Polygon Duchschnitt "Name".
mode = 2: Turtle-Polygon Vereinigung "Name".
mode = 3: Turtle-Polygon geschlossen: Turtle-Polygon ohne "Name".
Turtle-Polygon offen: Turtle-Polygon mit "Name" abschneiden.

t.setPolygon("Name", 4, dt=0.1);
mode = 4: "Name" liefert die Stützpunkte eines quadratischen Bezier Splines.
epsilon<dt<1 Bestimmt die Punktedichte pro Spline-Segment.

---

drawMatrix: Die Matrix legt die aktuelle Abbildung für 'Eigene Befehle' fest.
Achtung: Die Einstellungen verändern die bestehende Matrix kumulativ und sind nicht kommutativ!.

Mit t.setIdentity(); wird bekommt man wieder die Einheitsmatrix (reset).

reset=true : Abbildung auf die Einheitsmatrix anwenden.
reset=false : Abbildung auf drawMatrix anwenden.

t.setRotate(Winkel, reset=true);	Drehung um Winkel in Grad.
t.setRotate("P", Winkel, reset=true);	Drehung um "P" mit Winkel in Grad.
t.setMirror("P1", "P2", reset=true);	Spiegelung an der Geraden "P1" "P2".
t.setScale(sx, sy, reset=true);	Skalierung.
t.setTranslate(dx, dy, reset=true);	Translation.
t.setShear(sh, sv, reset=true);	Scherung.
t.setIdentity();	Einheitsmatrix (reset).

Turtle-Befehle ohne Abbildung mit drawMatrix:

t.shift("Name", t=1); Ein moveTo auf (lastX(), lastY()) + t*(Vektor "Name")) mit Drehung um lastW().

t.moveTo("P1", Index=0); Turtle auf Punkt oder Polygon "P1" mit Pen bewegen.
goTo("P1", Index=0); Turtle auf Punkt oder Polygon "P1" ohne Pen bewegen.
Bei einem Polygon bestimmt der Index den Punkt.
Gültig sind 0,1, bis n oder -1,-2, bis -(n+1). 0 ist erster Punkt, -1 letzter Punkt.

Messagebox

t.msg(Meldung); Text oder Zahlen (Variant) in der MsgBox anzeigen..
t.clrMsg(); MsgBox löschen.

t.msgPolygon(); Polygon als Funktion in der MsgBox anzeigen.

t.msgScript(showWerte=true); true: Script-Objekte und Werte in der MsgBox anzeigen.
false: Nur die Script-Objekte anzeigen.

Diashow

DiaRestwert = t.showDia(ms, Startwert, Decrement=1);

Grafik in ms Millisekunden neu zeichnen. Startwert mit Decrement herunterzählen.
Rückgabe: DiaRestwert in ms.

if (DiaRestwert<=0)	DiaRestwert = Startwert - Decrement;
else	DiaRestwert = DiaRestwert - Decrement;
if (DiaRestwert>0)	Neuzeichnen in ms.
else	Stop

"Script Funktionen" Basisbefehle und Zeichenbefehle für Funktionsgrafen.

var min = Blattrand	Startwert von t.
var max = Blattrand	Endwert von t.
var dt = 0.2	Für den Umlaufsinn: min<max: dt ist Increment (t1= t0 + dt).

min>max: dt ist Decrement (t1= t0 - dt).

t.drawGraf("Fnkt", dx=0, dy=0);

Funktionsgraf vom Typ f(t) zeichnen.
"Fnkt" beliebiger Name einer Scriptfunktion f(t) mit einem Parameter.
dx,dy: Verschiebung des Ursprungs

t.drawGraf("Fnkt1", "Fnkt2", dx=0, dy=0);

Parameterfunktion vom Typ fx(t) und fx(t) zeichnen.
"Fnkt1" beliebiger Name der ersten Scriptfunktion fx(t)
"Fnkt2" beliebiger Name der ersten Scriptfunktion fy(t)
dx,dy: Verschiebung des Ursprungs

"Script Escher" Basisbefehle und Morphing für 'Eigene Befehle'

Im Script werden zwei Morphingfunktionen erwartet:

```
function morphX(x,y)
   { ...
     return x-Koordinate;
   }
```
Morphingfunktion für die x-Koordinate
Funktionscode

```
function morphY(x,y)
   { ...
     return y-Koordinate;
   }
```
Morphingfunktion für die y-Koordinate
Funktionscode

`t.morphDraw("Befehl", UseStyle=true);`

Ruft den Befehl t.draw("Befehl") auf. Nach dem Anwenden der drawMatrix werden alle Punkte P(x,y) durch P(morphX(x,y) , morphY(x,y)) ersetzt.
UseStyle==true: Style von "Name" verwenden.
UseStyle==false: Style von Turtle verwenden.

"Script Lindenmayer" Basisbefehle und Unterstützung für Lindenmayer-Systeme.

`t.makeLSystem();` L-System Definitionen aus dem Script lesen und Modulstring erzeugen
`t.drawLSystem();` Modulstring zeichnen

`string t.showLInfo();` Rückgabe: Definitionen des L-Systems
`string showLModule();` Rückgabe: Modulstring