20.4 Clipboard

Die Klasse *Clipboard (gb.qt4)* ist statisch und kann für die Manipulation des Clipboards des Systems verwendet werden. Das Clipboard oder die Zwischenablage ist ein Speicherbereich für den einfachen Austausch von Daten (Text und Bilder) zwischen Programmen. Zuerst werden die Daten kopiert oder ausgeschnitten und im Clipboard gespeichert. Dann kann man die im Clipboard gespeicherten Daten in geeigneten Gambas-Komponenten für Text oder Bilder einfügen und realisiert so das Kopieren oder Verschieben von Daten. Mit dem Ende des Gambas-Programms, dass das Clipboard nutzt, wird auch das Clipboard-Objekt zerstört und damit dessen Inhalt. Abhilfe schafft die Installation eines Clipboard-Managers:

- http://en.wikipedia.org/wiki/Clipboard_manager
- http://freedesktop.org/wiki/ClipboardManager/

20.4.1 Konstanten

Die Klasse *Clipboard* verwendet 3 Konstanten für die Kennzeichnung des Typs des Inhalts des Clipboards:

Konstante	Konstante	Beschreibung
None	0	Der Clipboard-Inhalt konnte weder als Text noch als Bild interpretiert werden.
Text	1	Der Clipboard-Inhalt wurde als Text interpretiert.
Image	2	Der Clipboard-Inhalt wurde als Bild interpretiert.

Tabelle 20.4.1.1 : Konstanten der Klasse Clipboard

20.4.2 Eigenschaften

Die Klasse *Clipboard* verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Format	String	Gibt das Text-Format des im Clipboard gespeicherten Textes zurück oder NULL, wenn das Clipboard leer ist.
Formats	String[]	Zurückgegeben wird ein String-Array mit den MIME-Typen der im Clipboard gespeicherten Daten.
Type	Integer	Gibt den Typ des Inhalts des Clipboards zurück. Das ist entweder Clipboard.Text (=1) für Text oder Clipboard.Image (=2) für ein Bild oder Clipboard.None (=0), wenn das Clipboard leer ist.

Tabelle 20.4.2.1 : Eigenschaften der Klasse Clipboard

20.4.3 Methoden

Die Klasse *Clipboard* besitzt drei Methoden, deren Beschreibung Sie in der nächsten Tabelle finden:

Methode	Beschreibung
Clear	Der Aufruf der Methode *Clipboard.Clear* löscht den Inhalt des Clipboards.
Copy (Data AS Variant [, Format AS String])	Kopiert Daten in das Clipboard. Sind die Daten vom Typ 'String', dann können Sie über den Parameter 'Format' zum Beispiel mit "text/plain" den MIME-Typ vorgeben.
Function Paste ([Format As String]) As Variant	Fügt den Inhalt des Clipboards an der vorgesehenen Stelle ein. Sie können den MIME-Typ über den optionalen Format-Parameter angeben. Es wird NULL zurückgegeben, wenn der Inhalt des Clipboards leer ist oder nicht als Text oder Bild interpretiert werden kann.

Tabelle 20.4.3.1 : Methoden der Klasse Clipboard

20.4.4 Mime-Typen

Daten, die aus einem Programm in das Clipboard kopiert wurden, existieren dort meist in mehreren Formaten.

- Sie haben die Möglichkeit, die Formate der im Clipboard gespeicherten Daten – angezeigt mit dem entsprechenden MIME-Typ – über den Wert der Eigenschaft *Clipboard.Formats* auszulesen und anzuzeigen.
- Ein Screenshot zum Beispiel – aufgenommen mit der Tasten-Kombination CTRL+Drucken-Taste – liegt in 7 Formaten im Clipboard: *image/x-MS-bmp, image/x-bmp, image/bmp, image/tiff, image/jpeg, image/png* und *application/x-qt-image* (→ Ubuntu).
- Ein kopierter Text aus einer Webseite – angezeigt mit Firefox – hatte dagegen diese Text-Formate: *text/html, text/_moz_htmlcontext, text/_moz_htmlinfo, text/plain* und *text/x-moz-url-priv*.

Kennen Sie die Formate Ihrer bevorzugten (Quell-)Anwendungen, dann können Sie später die Texte oder Bilder schon in einem geeigneten Format in das Clipboard legen oder aus diesem im vorgegebenen Format in die (Ziel-)Anwendung einfügen, welche diese Daten weiter verarbeitet.

20.4.5 Daten in das Clipboard einfügen

Es gibt mehrere Möglichkeiten, Daten in die Zwischenablage oder in das Clipboard zu legen. Gemeinsam ist allen Daten, dass Sie unter Gambas nur Text oder Bilder in das Clipboard einfügen können und von dort in eine (Ziel-)Anwendung. Das gilt nicht für Text- oder Bild-*Dateien*! Deren Inhalt müssen Sie erst einer Eigenschaft eines verwendbaren Gambas-Objekts mit dem Datentyp *Picture* oder *Image* zuweisen.

Beispiel 1 – Bild aus einer *Datei* in das Clipboard (Datentyp Image) kopieren:

```
Clipboard.Copy(Image.Load("Pictures/fraktal_1.jpg"))
Clipboard.Copy(Picture.Load("Pictures/fraktal_1.jpg").Image)
```

Beispiel 2 – Bild aus einer *Picturebox* in die Zwischenablage legen:

```
If PictureBox1.Picture <> Null Then
   Clipboard.Copy(PictureBox1.Picture.Image)
Endif
```

Beispiel 3 – Desktop-Screenshot anzeigen und speichern

Die folgende Prozedur gibt eine Bildschirm-Kopie zurück, die sofort in einer PictureBox angezeigt und dann in einem bestimmten Verzeichnis als png-Grafik mit höchster Qualität (→ 100 %) abgespeichert wird:

```
[1]  Public Sub btnGetScreenShot_Click()
[2]    FMain.Hide
[3]      Wait 0.1 ' In der Praxis bis auf 3 Sekunden erhöhen!
[4]      PictureBox1.Picture = Desktop.Screenshot()
[5]      PictureBox1.Picture.Save(User.Home &/ "Bilder/current_screenshot.png", 100)
[6]    FMain.Show
[7]
[8]  ' Alternative ohne direkte Anzeige - Option Speichern-Dialog einfügen
[9]  ' FMain.Hide
[10] ' Wait 0.05
[11] '   Desktop.Screenshot().Save(User.Home &/ "Bilder/current_screenshot.png", 100)
[12] ' FMain.Show
[13]
[14] End ' btnGetScreenShot_Click()
```

Unter KDE und Gnome können Screenshots durch einfaches Drücken der Drucken-Taste erstellt werden. Mit der Tastenkombination Alt+Drucken-Taste legen Sie ein Abbild des aktiven Fensters in das Clipboard, den gesamten Desktop im Unterschied dazu mit CTRL+Drucken-Taste. Wenn Sie für das Anfertigen eines Screenshots ein spezielles Programme verwenden, so kann es passieren, dass deren Programm-Optionen die o.a. Wirkungen der Tasten oder Tasten-Kombinationen überschreiben. Zu diesen Programmen gehören *Shutter* und *KSnapshot* – aber auch *Gimp* (Datei> Erstellen> Bildschirmfoto).

Interessant ist das Konsolen-Programm 'Gnome-Screenshot' im Zusammenhang mit den Instruktionen SHELL und EXEC:

```
gnome-screenshot -w -b -c --delay=2
gnome-screenshot -i

-c, --clipboard                Aufnahme direkt an die Zwischenablage senden
-w, --window                   Nur ein Fenster anstatt des gesamten Bildschirms aufnehmen
-b, --include-border           Den Fensterrahmen mit in das Bildschirmfoto aufnehmen
-d, --delay=Sekunden           Ein Bildschirmfoto erst nach der angegebenen Zeit aufnehmen [in Sekunden]
-i, --interactive              Optionen interaktiv festlegen
```

Beispiel 4 – Screenshot des aktiven Top-Level-Fensters

```
SHELL "gnome-screenshot -c -w -b --delay=1" Wait
PictureBox2.Picture = Clipboard.Paste().Picture
```

Die Bildschirmkopie des Top-Level-Fensters wird in einer PictureBox angezeigt.