Tutorial zu Blitzplus I

Einzige Voraussetzung sind 2D Grundkenntnisse.

Vorab einiges

Blitzplus ist eine Weiterentwicklung des Blitz2D. Blitz2D wurde schnell abgelöst von verbesserten Blitz Sprachen (Blitz3D,Blitzplus,BlitzMAX), da 2D nicht annähernd soviel konnte wie die neueren, so kann man nun auch programmunabhänige Dynamic Link Library's, kurz Dll's, verwenden. Außerdem war es mit Blitz 2D nur möglich mit 2 Dimensionen zuarbeiten. Blitz 3D ging den entscheidenden Weg in die Dritte Dimension, Blitzplus ermöglichte das verwenden des windowstypischen Design und BlitzMAX erlaubt auch unter Linux zu programmieren. Blitz2D besitzt nur eine Hand voll Befehle,Blitzplus hingegen besitzt über 500 Befehle, Blitz3D sogar über 600. Das einzige was jetzt noch fehlt, wäre eine Kombination von Blitzplus und Blitz 3D. Dies wäre dann eine Allroundsprache mit 740 Befehlen die 2D,3D und GUI beherrschen könnte. Doch kommen wir zurück zu den Tatsachen, Blitzplus ist ein weiterer Schritt in Richtung Professionalität.

Grundlegendes: Fenster & Button

Der wohl wichtigste und grundlegendste Befehl in Blitzplus, ist der Befehl mit dem man ein Fenster erstellt. Als Parameter übergibt man den Namen, die X- und Y-Startposition des Fensters , die Breite, die Höhe und gegebenenfalls die Zugehörigkeit einer Gruppe und einen Parameter zu der äußerlichen Erscheinung.

```
Fenster = CreateWindow("Test-Fenster",0,0,300,300)
```

Damit hat man schon ein einfaches Fenster erstellt, doch von dem Fenster sieht man noch nicht viel da wir dem Programm noch nicht gesagt haben das es immer angezeigt werden soll. Dies lässt sich anhand einer Schleife beheben. Am besten nimmt man hier eine `Repeat - Until/Forever` Schleife. Das Beispielprogramm lässt sich mit der taste Escape schließen, die den Scancode 1 hat (`Until Keyhit(`***`1`***`)`).

```
Fenster = CreateWindow("Test-Fenster",0,0,300,300)
Repeat

Until Keyhit(1)
```

Da es aber nicht sonderlich ausgereift wirkt das Fenster nur mit Escape zuschließen, verwenden wir das von Blitzplus neueingeführte Event-System. Mit `WaitEvent()` wartet das Programm solange bis ein Event ausgelöst wird. Zum Beispiel ein Klick auf einen Button. `WaitEvent()` liefert einen Hexadezimalen Code zurück. Wenn auf das Kreuz oben rechts geklickt wird, so liefert `WaitEvent()` den Hexadezimalen Wert 803. Um ein Hexadezimalen Wert zudefinieren wird ein Dollarzeichen vor den Wert geschrieben.

Diese Event-System ist äußerst nützlich. Also modifizieren wir den Quellcode so, dass man nur auf das Kreuz klicken muss um das Fenster zuschließen und das Programm zu beenden.

```
Fenster = CreateWindow("Test-Fenster",0,0,300,300)
Repeat
      Event_ID = WaitEvent()

      If Event_ID = $803 Then End

Forever
```

Wenn nun ein Event bei `Waitevent()` eingetroffen ist gibt es noch einige andere Möglichkeiten dies zuverwenden. Mit `EventSource()` ermittelt man die Herkunft des letzten Events, zum Beispiel die ID von einem Button. Mit dem Befehl `EventData()` gelangt man an noch weitere Information. (genaueres siehe Onlinehilfe auf www.blitzbase.de).

Da das Fenster noch nicht wirklich was kann, und auch sonderlich nicht sehr spannend ist kommen wir zum Thema Buttons. `CreateButton()` ist der Befehl, mit dem man einen Button erstellt. Die Parameter die dieser Button erwartet ist, der Inhalt des Buttons, die X- und Y-Startposition, die Breite und Höhe, die Zugehörigkeit zu einem vorhandenen Fenster, und schließlich die Art des Buttons. Es gibt 3 verschiedene Arten von Buttons: [1] Normaler Button , [2] Checkbox , [3] Radiobutton. Außerdem, später bei Grafikausgaben bremst das allein stehende `WaitEvent()` das Programm ab, also sollte man lieber eine If Abfrage machen und in die Klammer von `WaitEvent()` eine 1 setzten. Dieser Parameter gibt an wie lange das Programm in Millisekunden angehalten werden soll und auf ein Event gewartet werden soll.

```
Fenster = CreateWindow("Test-Fenster",0,0,300,300)
normal_button = CreateButton ("Normal",10,10,120,25,Fenster,1)
checkbox_button = CreateButton ("Checkbox",10,40,120,25,Fenster,2)
radio_button = CreateButton ("Radiobutton",10,70,120,25,Fenster,3)

Repeat

 If WaitEvent(1) = $ 803 Then End

Forever
```

Nun müssen wir noch ermitteln ob der Button angeklickt/ausgefüllt ist. Dies machen wir mit dem Befehl `ButtonState()`. Ist der zurück gegebene Wert 1 , wurde er angeklickt/ausgefüllt. Bei 0 ist nichts passiert.
Hierbei können wir gleich den Befehl `SetGadgetText()` benutzten, der den vorher vereinbarten Buttontext, wieder verändern kann. So können wir prüfen ob der Status vom Button gleich 1 ist. Die geforderten Parameter sind der zuändernde Button, und der Neue Text.

```
Fenster = CreateWindow("Test-Fenster",0,0,300,300)
normal_button = CreateButton ("Normal",10,10,120,25,Fenster,1)
checkbox_button = CreateButton ("Checkbox",10,40,120,25,Fenster,2)
radio_button = CreateButton ("Radiobutton",10,70,120,25,Fenster,3)

Repeat

If ButtonState(checkbox_button) = 1 Then SetGadgetText checkbox_button,
"Checkbox[EIN]"
If ButtonState(radio_button) = 1 Then SetGadgetText radio_button,
"Radiobutton[EIN]"

If ButtonState(checkbox_button) = 0 Then SetGadgetText checkbox_button,
"Checkbox[AUS]"
If ButtonState(radio_button) = 0 Then SetGadgetText radio_button,
"Radiobutton[AUS]"


If WaitEvent(1) = $ 803 Then End

Forever
```

Grafik: Texte & Canvas

Im Moment können wir noch keinen freistehenden Text ausgeben oder eine Grafik zeichnen. Um dies in einem Fenster zu ermöglichen, müssen wir Canvas erzeugen. Das sind Bereiche in denen man zeichnen kann. Es gibt dafür 3 Befehle, die unumgänglich bei Canvas sind: `CreateCanvas()`,`Setbuffer CanvasBuffer()` und `FlipCanvas()`.

```
Canvas = CreateCanvas(X-Startposition,Y-
Startposition,Breite,Höhe,Zugehörigkeit)
```

Erstellt den Canvas.

```
Setbuffer CanvasBuffer(Canvas)
```

Aktiviert den Canvasbuffer für den jeweiligen Canvas.

```
FlipCanvas(Canvas)
```

Ist der Flipbefehl, für den jeweiligen Canvas.

In diesem Canvasbereich, kann man nun alle Standardmäßigen 2D Befehle benutzen. Zum Anfang, um nicht mit den Traditionen zubrechen, drucken wir den alt bewährten Text: „Hallo Welt“. Wer jetzt denkt, das geht doch einfacher mit `Print`, der liegt diesmal falsch. In Blitzplus wurden die Befehle `Print` und `Locate` entfernt, da der Befehl Text sie sowieso vereint.

```
Fenster = CreateWindow("Test-Fenster",0,0,300,300)
checkbox_button = CreateButton ("Text darstellen",10,40,120,25,Fenster,2)
radio_button = CreateButton ("Rot",10,70,120,25,Fenster,3)
radio_button2 = CreateButton ("Grün",10,90,120,25,Fenster,3)
radio_button3 = CreateButton ("Blau",10,110,120,25,Fenster,3)

canvas = CreateCanvas(140,0,140,100,Fenster)

SetBuffer CanvasBuffer(canvas)

Repeat
Cls


If ButtonState(checkbox_button) = 1 Then Text 10,10," Hallo Welt"

If ButtonState(radio_button) = 1 Then Color 255,0,0
If ButtonState(radio_button2) = 1 Then Color 0,255,0
If ButtonState(radio_button3) = 1 Then Color 0,0,255


    If WaitEvent(1) = $803 Then End

FlipCanvas canvas
Forever
```

Das Menü

Bei Anwendungen ist es oft hilfreich ein Menü zuerstellen. Da man das nicht neu

programmieren muss, kann man auf das alt bewährte Windowsmenü zurückgreifen. `CreateMenu()` erzeugt dieses Menü, als Parameter müssen nur der Name des Menüs, eine Identitätsnummer (wird später nochmals genauer behandelt), und das Fenstermenü in dem es erscheinen soll, das jeweilige Fenstermenü wird Mit `WindowMenu()` und dem jeweiligen Fenster ermittelt. Das erste Menü das erschaffen wird ist das Menü das ganz links steht. Man sollte sich die Reihenfolge schon einmal vorher klar machen. Gut haben wir ein Menü, nur wo sind die Auswahlmöglichkeiten? Nutzen wir den gleichen Befehl noch einmal, nun setzten wir aber statt des dazugehörigen Fenster den Variablenname des Menüs ein. Dies ist aber nur eine Art des Menüs, es gibt noch die sogenannte Check-Variante, diese definieren wir wiefolgt mit `CheckMenu` (der gegen Befehl lautet `UncheckMenu`) und dem Menü Punkt der umgewandelt werden soll. Wenn man möchte das ein Menüeintrag nicht sofort oder nicht immer verfügbar ist kann man ihn, sozusagen ,deaktivieren mit DisableMenu. Um den Menüeintrag wieder zu aktivieren benutzen wir den Befehl `EnableMenu`. Nachträglich kann der Menütext mit `SetMenuText`, dem Menü und dem neuen Text geändert werden. Zu guter letzt müssen wir das Menü aktualisieren mit `UpdateWindowMenu`, als Parameter übergeben wir die Identität des Fensters. Das aktualisieren muss immer erfolgen wenn etwas verändert wurde, sonst wird es nicht dargestellt.

```
fenster = CreateWindow("Menü-Fenster",100,100,150,150,0,1)
hauptmenu1 = CreateMenu("Datei",0,WindowMenu(fenster))
hauptmenu2 = CreateMenu("Bearbeiten",1,WindowMenu(fenster))

untermenu1_1 = CreateMenu("Neu",2,hauptmenu1)
untermenu1_2 = CreateMenu("Öffnen",3,hauptmenu1)
untermenu1_3 = CreateMenu("Beenden",4,hauptmenu1)

untermenu2_1 = CreateMenu("Filter 1",2,hauptmenu2)
untermenu2_2 = CreateMenu("Filter 2",2,hauptmenu2)
untermenu2_3 = CreateMenu("Rendern",3,hauptmenu2)

CheckMenu untermenu2_1
UncheckMenu untermenu2_2
DisableMenu untermenu2_3

UpdateWindowMenu(fenster)

Repeat

If WaitEvent(1) = $803 Then End

Forever
```

Textfelder, Listen & Labels

Fangen wir an mit den Textfeldern, diese sind recht einfach handzuhaben. Mit dem Befehl `CreateTextField(X-Startposition, Y-Startposition,Breite, Höhe, Fenster)` erstellt man eine einfache Eingabezeile. Diese kann nun im Programm mit Textgefüllt werden. Soll diese Feld schon vorher mit Text gefüllt werden so kann man das mit dem Befehl `SetGadgetText(Textfeld,Text)`. Wenn man nun aber einen mehrzeiligen Text, wie zum Beispiel für längere Texte, kann man mit `CreateTextArea(X-Startposition, Y-Startposition,Breite, Höhe, Fenster)` ein Textbereich erstellen. Um nun den Text auch verwerten zukönnen, müssen wir das eingegebene auch auslesen. Dazu speichern wir den Text mit `TextFieldText(Textfeld)`in eine Variable. Dieser Befehl gilt für `CreateTextfield()`, sowie für `CreateTextArea()`. Da man aber nicht immer weiß, was man in die Textfelder eintragen soll, kann man sogenannte Labels erstellen, da es zu umständlich wäre immer neue Canvas zuerstellen. Der Befehl dazu lautet `CreateLabel(Text,X-Pos,Y-Pos,Breite,Höhe,Fenster)`. Nun machen wir uns an die

Listen, es gibt zwei Arten, einmal die ComboBox und die ListBox. Der Unterschied ist schwer zuerklären, aber ich versuch es. Nun, in einer Listbox sieht man mehrere Möglichkeiten auf einmal, in einer ComboBox ist der erste Eintrag leer, nun man öffnet die Auswahlliste mit einem Klick auf den Aufklapp-Pfeil und selektiert den gewünschten Eintrag. Wenn man eine ComboBox mit (`CreateComboBox(X-Pos, Y-Pos, Breite, Höhe, Fenster`) erstellt hat, kann man anhand des Befehls `AddGadgetItem(Liste, Text, selektiert? [1=JA][0=NEIN], Icon [falls vorhanden])` ein Item der Liste hinzufügen. Das gleiche macht man mit `CreateListBox(X-Pos, Y-Pos, Breite, Höhe, Fenster).` Nun gibt es eine Reihe an Befehlen die man nutzen kann:

`item=SelectedGadget(Gadget)`
Liefert die Nummer des selektierten Items/Eintrags.

`SelectGadgetItem(Gadget,item)`
Selektiert ein Item/Eintrag in der Liste.

`anzahl=CountGadgetItem(Gadget)`
Liefert die Anzahl der Items/Einträge in der Liste.

`ModifyGadgetItem(Gadget, Nummer des Gadgets, Text, Icon [falls vorhanden]`
Modifiziert ein Item/Eintrag in der Liste.

`InsertGadgetItem(Gadget, Stelle des Einfügen, Text, Icon [falls vorhanden]`
Fügt ein Item/Eintrag an der gewünschten Position ein.

`ClearGadgetItems Gadget`
Löscht alle Items/Einträge in der Liste (! Liste bleibt bestehen !)

Einen Befehl der auch wichtig ist, ist der Befehl `FreeGadgetItem Gadget`.
Dieser Befehl löscht die Liste. Dies funktioniert auch mit Buttons, Textfelder, Fenster und anderen Gadget-Objekten.

So das war der erste Teil des Tutorials. Geplant sind noch zwei weitere. Eins zum Thema Toolbars, Tabbers, Verzweigungen(TREE-View), HTML und Slider. Im zweiten werden dann die letzten Themen wie Request-Funktionen, Fortschrittleisten, Panels und Befehle, die keinen großen Bereiche zugeordnet werden können.

Ich hoffe das Tutorial ist hilfreich zum erlernen der Sprache BlitzPlus