My Gui Documentation

Index

Getting Started

Installation

Unzip the downloaded file and copy the ‘mygui’ folder (located in the module folder) into your monkey modules folder.
Make sure that you also have fontmachine module in there
Fontmachine is free - http://www.jungleide.com/fontmachine/fontmachine-14-01-14-A.zip

Integration

At the beginning of your code:

Import mygui

In your Update Method

Gui.Update()

In Your Render Method:

Gui.Draw()

```
Import mojo
Import mygui

Function Main()
	New Game
End Function

Class Game Extends App
	Method OnCreate()
		SetUpdateRate(60)
	End Method

	Method OnUpdate()
		Gui.Update()
	End Method

	Method OnRender()
		Cls(200,200,200)
		Gui.Draw()
	End Method
End Class
```

Please Note

MyGui works by drawing elements onto the ‘Gui.CurrentScreen’.
This means that in order to create anything you will first have to create a screen to put it on.
You can have multiple screens and use effects to transition between them.

How to Create Elements

Declaring Elements

All Element types use the same definition type (*Gui*)

Global MyScreen:Gui
Global MyButton:Gui

Creating Elements

After Declaring you element you can create it, either in your OnCreate Method or on the fly anywhere in your program.

Remember you will first need to create a screen to place them on.

MyScreen Gui.CreateScreen()
MyButton = Gui.CreateButton(X, Y, W, H, Text$, Parent:Gui, Style=0)

All of the creation functions are listed o the next page along with a description of their parameters. But for now let me show you a simple working example.

(You may have to manually change the (“) as they copy/paste wrong)

```
Import mojo
Import mygui

Global MainScreen:Gui
Global MyButton:Gui

Function Main()
        New Game
End Function

Class Game Extends App
        Method OnCreate()
                SetUpdateRate(60)
                MainScreen = Gui.CreateScreen()
                MyButton = Gui.CreateButton(10, 10, 120, 25, “MyButton”, MainScreen)
        End Method

        Method OnUpdate()
                Gui.Update()
                If MyButton.IsClicked Then Gui.MsgBox(“MsgBox”, “MyButton Clicked”)
        End Method

        Method OnRender()
                Cls(200,200,200)
                Gui.Draw()
        End Method
End Class
```

Creation Functions 1

Unless otherwise stated:

X and Y relate to the location within the parent i.e position of button within its parent window.
Parent must be a screen, window or tab.

Most text will allow you to use ~n as a new line

Screens

Gui.CreateScreen()

The current screen is automaticaly set to the last screen you create, you can set the current screen manualy or transition to different screen - this is covered later on.

Windows

Gui.CreateWindow(X, Y, W, H, Title:String, Parent:Gui, Order = 1, Closeable = 1, Minimisable = 1, Maximisable = 1, Sizeable = 1, Moveable = 1, Statusbar = 1)

Order: 0=Always behind 1=Variable 2=Always ontop
If the title is “” then the window wil have no titlebar

Buttons

Gui.CreateButton(X, Y, W, H, Text:String, Parent:Gui. Style=0)

Style can either be 0/1, in the default theme 0=grey/black 1=blue/white

Img Button

Gui.CreateImgButton(X, Y, W, H, Img:Image, Parent:Gui, ImgX = 0, ImgY = 0, ImgW = 0, ImgH = 0)

The Image must contain 4 states (Normal,over,down,inactive) in a row in that order.
Image can be part of an atlas (where you would enter ImgX / ImgY)
When part of an atlas ImgW and ImgH are the width/height of the a single state

Slider Button

Gui.CreateSlideButton(X, Y, W, H, Text1:String, Text2:String, Parent:Gui, Value = 0)

Text1 = Left Text2=Right.   Value0 = Left revealed Value1 = Right revealed

Tickbox

Gui.CreateTickbox(X, Y, H, Text:String, Parent:Gui, StartValue = 0)

0=Unchecked   1=Checked

Radio

Gui.CreateRadio(X, Y, H, Text:String, Parent:Gui, Group = 0, StartValue = 0)

0=Unchecked   1=Checked
Group lets you set a number to have different groups of radio boxes within the same parent.
Group number is only valid in a single parent. i.e two groups with different parents can have the same number and be seperate.

DropDowns

Gui.CreateDropDown(X, Y, W, H, Parent:Gui, StartText:String = “Please Select...”)

DropDown Items

Gui.CreateDropDownItem(Text:String, Parent:Gui, Value = 0)

Parent must be a Dropdown
Value and text gets assigned to the parent when selected.

Creation Functions 2

Unless otherwise stated:

X and Y relate to the location within the parent i.e position of button within its parent window.

Parent must be a screen, window or tab.

Most text will allow you to use ~n as a new line

Listboxes

Gui.CreateListbox(X, Y, W, H, Parent:Gui)

Listbox Items

Gui.CreateListboxItem(Text:String, Parent:Gui, Value = 0)

Parent must be a Listbox
Value and text gets assigned to the parent when selected.

Vertical Scrollbars

Gui.CreateVScrollbar(X, Y, W, H, Minimum, Maximum, Parent:Gui, Value = 0, Stp = 0)

Stp is the increment amount of the scrollbar i.e. it goes between 0 and 100 in steps of 10.

Horizontal Scrollbars

Gui.CreateHScrollbar(X, Y, W, H, Minimum, Maximum, Parent:Gui, Value = 0, Stp = 0)

Stp is the increment amount of the scrollbar i.e. it goes between 0 and 100 in steps of 10.

Sliders

Gui.CreateSlider(X, Y, W, H, Minimum, Maximum, Parent:Gui, Value = 0, Stp = 0)

Stp is the increment amount of the scrollbar i.e. it goes between 0 and 100 in steps of 10.

Menus

Gui.CreateMenu(Text:String, Parent:Gui)

Parent must be a window or a screen

Menu Items

Gui.CreateMenuItem(Text:String, Parent:Gui, Tickbox = 0, Value = 0)

Parent must be a Menu or another MenuItem
When Tickbox is 1 the menu's value will return the value of the tickbox

Tabs

Gui.CreateTab(Text:String, Parent:Gui)

Parent must be a Screen or a Window
Parent elements to tabs as you would a window or a screen

Labels

Gui.CreateLabel(X, Y, Text:String, Parent:Gui, Align = 0, Boarder = 0)

Align aligns multiline text (~n), X,Y remains the top left.
Boarder adds a 1px line around the label

Creation Functions 3

Unless otherwise stated:

X and Y relate to the location within the parent i.e position of button within its parent window.

Parent must be a screen, window or tab.

Most text will allow you to use ~n as a new line

Textfields

Gui.CreateTextField(X, Y, W, H, Parent:Gui, Text:String, AllowNumbers = 1, AllowLetters = 1, AllowSymbols = 1, MaxLength = 0)

Textfields are single line text entry
MaxLength limits the amout of characters that can be typed. 0=unlimited

Textboxes

Gui.CreateTextBox(X, Y, W, H, Parent:Gui, Text:String, AllowNumbers = 1, AllowLetters = 1, AllowSymbols = 1, Wordwrap = 0)

Textfields are single line text entry
Wordwrap=1 keeps the text formatted to within the width of the box.

Tables

Gui.CreateTable(X, Y, W, H, Parent, Rows, Colomns, CellWidth, CellHeight)

This creates an empty table. see table notes on how to add headers, textfields,dropdowns and ticks.

Charts

Gui.CreateChart(X,Y,W,H, Parent:Gui, Type, Data:String,Title:String, XLabel:String, YLabel:String, HasKey)

Type: 0=Bar 1=Line 2=Scatter 3=Pie
Data: See table notes
HasKey: Displays a key from within the data string

Screen Transitions

You can either set the current screen or transition to a new screen by doing the following:
(Where screen = destination screen)

Gui.SetScreen(Screen:Gui)
Gui.FadeToScreen(Screen:Gui)
Gui.SlideToScreen(Screen:Gui)
Gui.TurnToScreen(Screen:Gui)
Gui.ZoomToScreen(Screen:Gui)

Setting and Returning Element Properties

Setting an elements properties

Once an element has been created you can still alter its properies, location, size, position, value etc by simply doing the following:

```
MyElement.X = 100
MyElement.Value = 1
MyElement.Text = “New Text”
MyListBox.Selected = MyListBoxItem
```

Returning an elements properties

```
Local BX = MyElement.X
If MyTickbox.Value = 1 Then ...
```

List of changable / Returnable properties

X,Y,W,H - The position and size of most elements
Value - The value of most elements i.e. tickbox, radio
Text:String - The text of most elements, this is also used as the title text of a window
Active - 0=Inactive 1=Active
Minimisable, Maximisable, Closeable, Sizeable, Moveable, Statusbar - Windows only
StatusText:String - Windows only
Minimum, Maximum, Stp - Scrollbars and sliders only
Text1, Text2 - Slider buttons only
SelectedTab:Gui - Use to set window/screen current tab
Wordwrapped - Used in Textboxes
Selected:Gui - Used in Listboxes

Special Cases

Listboxes and dropdowns will contain the text and the value of their selected items.
Menus if they have tickbox will have the value 0/1
A windows text is its titlebar text - if “” the window will have no titlebar

To Change a window or a menuItems Icon

```
MyWindow.AddIcon(Image:Image, ImgX = 0, ImgY = 0, ImgW = 0, ImgH = 0)
```

If part of an atlas you will need to enter ImgX, ImgY, ImgW, ImgH

Tooltips

Tooltips are super easy:

```
MyElement.Tooltip = “This is a tooltip”
MyElement.Tooltip = “”
```

Gui Interaction

This page covers how to use the elements when they are clicked etc.
There are two ways to do this:

Gui.Clicked - Returns the gui element that has been clicked (Valid for 1 loop)
Gui.Over - Returns the gui element that is currently moused over
Gui.OverTime - Returns the start millisecs the current element was first mouse over
Gui.Down - Returns the gui element that is currently mouse held down
Gui.DownTime - Returns the start millisecs the current element was first mouse down

Alternatly you can use the element itself. i.e.

MyElement.IsClicked - Returns 1 if the element has been clicked (Valid for 1 loop)
MyElement.IsDoubleClicked - Returns 1 if the element has been clicked (Valid for 1 loop)
MyElement.IsOver - Returns 1 if the element is mouse over
MyElement.IsDown- Returns 1 if the element is mouse held down

```
Import mojo
Import mygui

Global MainScreen:Gui
Global MyButton:Gui
Global StatusText:String

Function Main()
	New Game
End Function

Class Game Extends App
	Method OnCreate()
		SetUpdateRate(60)
		MainScreen = Gui.CreateScreen()
		MyButton = Gui.CreateButton(10, 100, 120, 25, “MyButton”, MainScreen)
	End Method

	Method OnUpdate()
		Gui.Update()
		StatusText = “”
		If MyButton.IsOver Then StatusText = “Button Over”
		If MyButton.IsDown Then StatusText = “Button Down”
		If MyButton.IsClicked Then StatusText = “Button Clicked”
	End Method

	Method OnRender()
		Cls(200,200,200)
		Gui.Draw()
		DrawText(StatusText, 0, 0)
		If Gui.Over <> Null Then DrawText(“OverTime: ” + Gui.OverTime, 100, 0)
	End Method
End Class
```

MsgBox

Message boxes can be used as a simple notification and can be used to return an 'ok / cancel / close' user input.

Display a message box

Gui.MsgBox(Title:String, Message:String, Buttons, CloseButton=1, Reference:String="")
Buttons: 1=Ok 2=Ok & Cancel
Close button refers to the close button on the msgbox window
Reference is needed if you inted to capture what button the user clicked

Get User input from a MsgBox

Checks to see if the chosen button of a chosed msgbox is clicked

Gui.CheckMsgBox(Reference:String, Button)
Reference is the reference:String used when displaying the msgBox
Button: 0=Close   1=Ok   2=Cancel

```
Import mojo
Import mygui

Global MainScreen:Gui
Global MyButton:Gui

Function Main()
        New Game
End Function

Class Game Extends App
        Method OnCreate()
                SetUpdateRate(60)
                MainScreen = Gui.CreateScreen()
                MyButton = Gui.CreateButton(10, 100, 120, 25, "MsgBox", MainScreen)
        End Method

        Method OnUpdate()
                Gui.Update()
                If MyButton.IsClicked Then Gui.MsgBox("Title", "Message", 2, 1, "Ref1")
                If Gui.CheckMsgBox("Ref1", 0) Then Gui.MsgBox("Result", "Close button clicked")
                If Gui.CheckMsgBox("Ref1", 1) Then Gui.MsgBox("Result", "Ok button clicked")
                If Gui.CheckMsgBox("Ref1", 2) Then Gui.MsgBox("Result", "Cancel button clicked")
        End Method

        Method OnRender()
                Cls(200,200,200)
                Gui.Draw()
        End Method
End Class
```

Custom Canvas

You can create a custom canvas for a window or a screen. This will let you draw into a winodw/screen behind other elements.

First create a custom class that extends ‘MyGui_Canvas’

Your custom class should have two methods - Draw() and Update().

```
Class MyCanvas Extends MyGui_Canvas
    Method Draw()
        ‘Your drawing commands
    End Method

    Method Update()
        ‘Your update commands
    End Method
End Class
```

Attach your custom class to the desired window / screen

```
MyWindow.Canvas = New MyCanvas
```

Returning the canvas X,Y,W,H

In order for you to use the correct drawing positions for your custom canvas, the MyGui_Canvas has 4 fields (X, Y, W, H) that you reference to directly. (Or by using Self.X etc)

```
Class MyCanvas Extends MyGui_Canvas
    Method Draw()
        Local CanvasX = X
        Local CanvasY = Y
        Local CanvasW= W
        Local CanvasH = H
        DrawText(X+”,”+Y+”,”+W+”,”+H,X+10,Y+10)
    End Method

    Method Update()

    End Method
End Class
```

File Operations

Import

This will only work with targets that support monkeys os module
Due to the file operations needing the os module I have implemented it as a seperate import.

```
Import mygui.fileoperations
```

Opening the file operations window

```
Gui_FileOperation.LoadFile(Extension:String, Reference_Id:String)
Gui_FileOperation.SaveFile(Extension:String, Reference_Id:String)
```

Extension: This is the file type to be opened/saved. The Window will filter the file types.
Reference_Id: This reference be unique. It enables you to have multiple Load/save operations.

Returning User File Path

```
Gui_FileOperation.CheckLoadReturn(Reference_Id:String)
Gui_FileOperation.CheckSaveReturn(Reference_Id:String)
```

Reference_Id: This is the reference used in the above commands.

Example on next page

```
'File Operations Example

Import mojo
Import mygui
Import os
Import mygui.fileoperations

Global MainScreen:Gui
Global Button1:Gui
Global Button2:Gui
Global Textbox:Gui

Function Main()
	New Game()
End Function

Class Game Extends App
	Method OnCreate()
		SetUpdateRate(60)
		MainScreen = Gui.CreateScreen()
		Button1 = Gui.CreateButton(20, 20, 120, 24, "Load File", MainScreen)
		Button2 = Gui.CreateButton(150, 20, 120, 24, "Save File", MainScreen)
		Textbox = Gui.CreateTextBox(20, 55, 270, 200, MainScreen, "Text to be saved")
	End Method

	Method OnUpdate()
		Gui.Update()

		If Button1.IsClicked Then Gui_FileOperation.LoadFile("txt", "LoadFile")
		If Button2.IsClicked Then Gui_FileOperation.SaveFile(".txt", "SaveFile")

		Local LoadReturn:String = Gui_FileOperation.CheckLoadReturn("LoadFile")
		Local SaveReturn:String = Gui_FileOperation.CheckSaveReturn("SaveFile")

		'Load File
		If LoadReturn <> "" Then
			Textbox.Text = os.LoadString(LoadReturn)
		EndIf
		'SaveFile
		If SaveReturn <> "" Then
			os.SaveString(Textbox.Text, SaveReturn)
		End If
	End Method


	Method OnRender()
		Cls(240, 240, 240)
		SetColor(255, 255, 255)
		Gui.Draw()
	End Method
End Class
```

Tables

Adding Elements

Once you have created an empty table you can start filling it with elements by doing the following:

```
MyTable.TableAddHeader(CellX, CellY, Label:String)
MyTable.TableAddText(CellX, CellY,  Text:String, Numbers, Letters, Symbols, MaxLength)
MyTable.TableAddDropDown(CellX, CellY, Text:String)
MyTable.TableAddDropDownItem(CellX, CellY, Text:String)
MyTable.TableAddTick(CellX, CellY, Height, Label:String, Value)
```

Getting Data From a Table

```
MyTable.TableGetText(CellX, CellY)
MyTable.TableGetValue(CellX, CellY)
```

Remember even if a textfield within a table is displaying a number you will still need to use the TableGetText() Method

Charts

ChartData

Data for the chart is read from the ChartData field.
This string is seperated by ";" And ","

Example PieChart Data:
"100;200;100" - will draw a piechart with 3 segments

Example BarChart Data:
"10,10;5,9;5,10" - will draw 3 bars that are each split into 2

Example LineChart Data:
"0,0,10,10,20,20,30,30;0,10,10,20,20,30,30,40" - will draw2 Lines (x,y,x,y,x,y,x,y,x,y ; x,y,x,y,x,y,x,y)

Chart Key

If The charts 'ContainsKey' field is set to 1 then the system will read the first section of the data as chart key data, seperated by ',' until it reads ';'

Example Pie with key data
"Key1,Key2,Key3;100;200;100" - the order of the keys corresponds to the order of the data

Tab Control and SetFocus

Tab Control

In MyGui you can navigate/cycle through the elements using the tab and arrow keys.
Hitting enter or space will simulate a click of the current onFocus element.
Shift Tab will take the onFocus up a level. (e.g. from a window to a tab/menu/windowbuttons/parent)
Use arrow keys to navigate menus,dropdowns,listboxes, tabs and window buttons

Setting the OnFocus element manualy

```
Gui.SetFocus(Element:Gui)
```

Gui Settings and Themes

Gui Settings

There are a few settings you can adjust to make MyGui suit you!
These are listed below:

Gui.ShowTooltips - 0/1 If 0 no tooltips will be shown
Gui.Animated - 0/1 Animates the closing and minimising of windows
Gui.AnimatedSpeed - Sets the animated frame delay in millisecs (default 15)
Gui.Shadows - 0/1 Shows a shadow under windows, menus and dropdowns
Gui.UseFastDraw - 0/1 This is designed to reduce operation lag on slow computers,
if =1 (defualt) loading the screen takes longer, but once loaded will run faster.
if =0 then screen will load faster but will run slower.
(Fastdraw works by prerendering the elements)

Gui Theme

You can change the way MyGui looks and even the fonts it uses.
Note: To create a new font you will need fontmachine editor.
(I have included some themes and fonts to get you started - in the themes foder)

Gui.SetTheme(Img:Image)
Image must be the same layout as the 'Atlas.png' in the data folder

Gui.SetFonts(TitleFont:String = "", NormalFont:String = "", NormalInv:String = "")
The parameters refer to the '.txt' file of a fontmachine font