The MaxGUI Beginner Tutorial Series - Tutorial 18: The Treeview Gadget - Part 1

The treeview gadget is another common gadget used in all sorts of applications. For example the MaxIDE uses this gadget to display the function lists and debug information.

The thing about a treeview gadget is that it always requires a root to start. You can then start to add nodes to this root. The nodes can have other nodes attached to it. Imagine how the treeview in the Windows File Explorer behaves.

In fact I am going to use the File Explorer as the teaching metaphor to introduce you to the wonders of the treeview gadget.

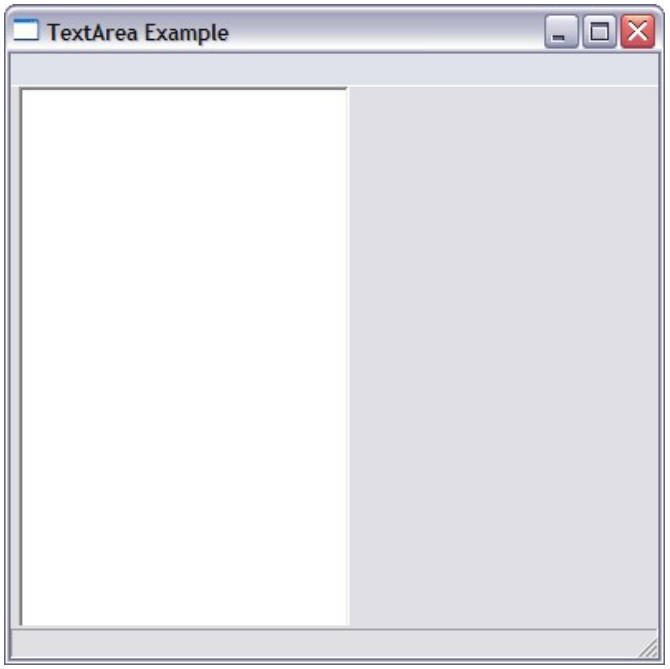
Creating a TextView Gadget using CreateTreeView

```
Function CreateTreeView:TGadget(x,y,w,h,group:TGadget,style=0)
```

A very simple start to this Gadget will be as follows:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("TreeView Example", 40,40,400,400)
Global MyTreeView:TGadget=CreateTreeView(5,0,200,360,MyWindow)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  End Select
Forever
End
```

Unlike some other CreateGadget function, this one does not do anything much:)

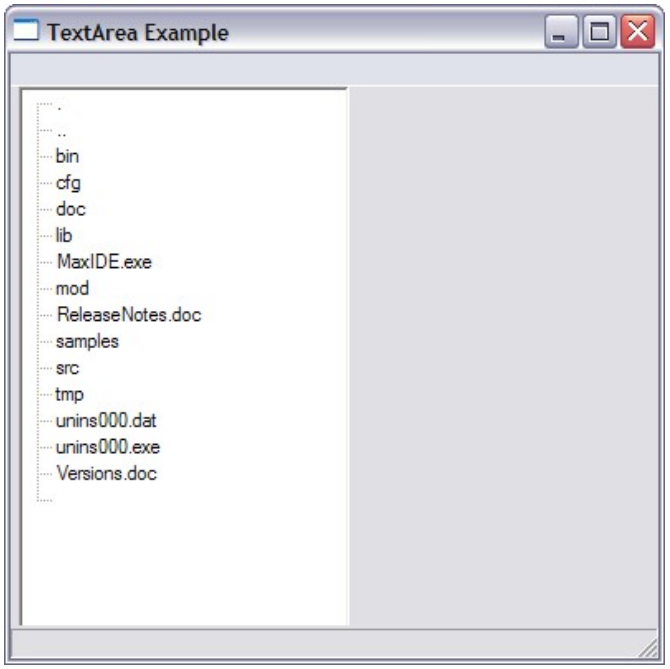


Adding Nodes to the TreeView Gadget using AddTreeViewNode

What we are going to do now is to read in the files (or folders) in the BlitzMax installed directory and add those to our TreeView via the **AddTreeViewNode** function.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("TreeView Example", 40,40,400,400)
Global MyTreeView:TGadget=CreateTreeView(5,0,200,360,MyWindow)

Local Folder:int=ReadDir(BlitzMaxPath())
Local File:String
Repeat
   File=NextFile(Folder)
   AddTreeViewNode(file,MyTreeView)
Until File=Null

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  End Select
Forever
End
```

The TreeView is now populated with some items. Not yet recognisable ala File Explorer but already we can see some similarities.



Let us explain a little bit how we are reading the files/folder in. We first need to start with a **ReadDir** function which actually will open a directory for reading, in this case the folder where BlitzMax was installed to.

```
Local Folder:int=ReadDir(BlitzMaxPath())
```

The **NextFile** function will allow us to loop through every file/folder in this directory until we hit a null which denotes the end of the directory (no more file to read) so we stop

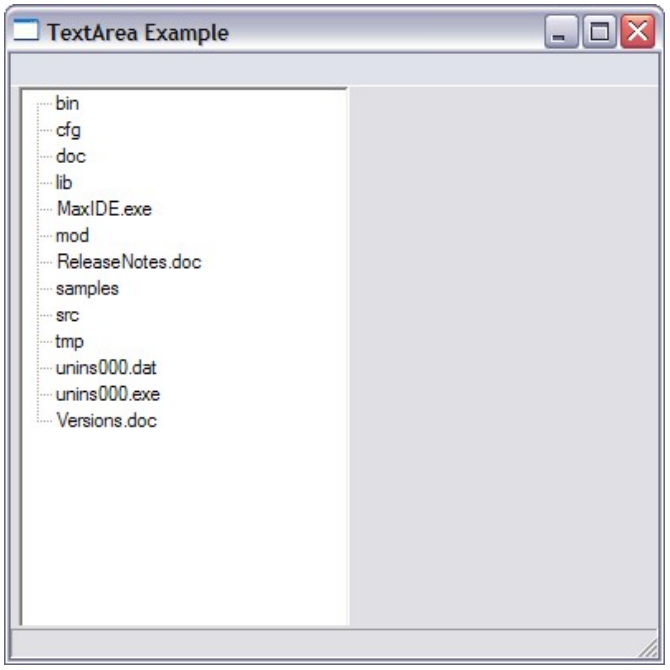
```
Repeat
   File=NextFile(Folder)
   AddTreeViewNode(file,MyTreeView)
```

```
    Until File=Null
```

You will notice however that the items in the TreeView contains items which we never see in a real File Explorer so we need to filter them out as follows:-

```
  Repeat
     File=NextFile(Folder)
     If File=".." Or File="." Or File=Null Then
        'Do Nothing
     Else
        AddTreeViewNode(file,MyTreeView)
     EndIf
  Until File=Null
```

We now have a much cleaner look.



If we look at the syntax for **AddTreeViewNode**, we can see that it has an icon parameter.

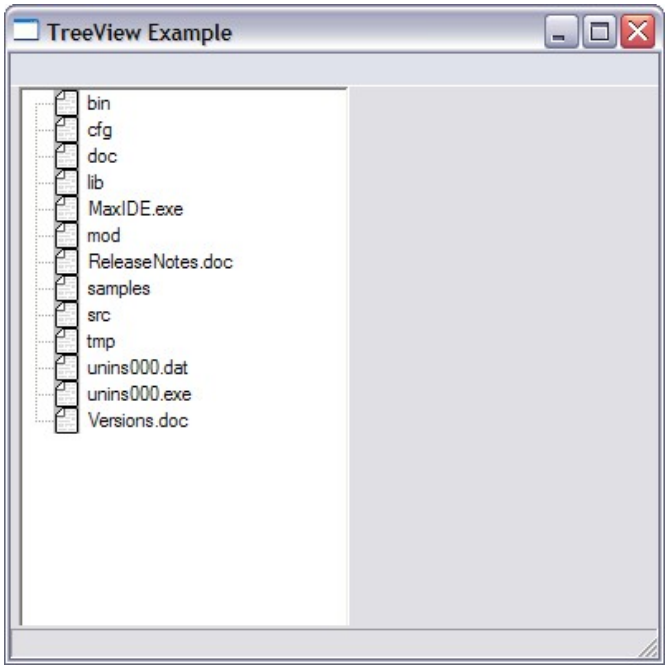
```
Function AddTreeViewNode:TGadget( text$,node:TGadget,icon=-1 )
```

In our previous encounters with icons in gadget, we know that we first have to load an iconstrip and then assign that to the relevant gadget as follows:-
** Download this bmp and place it in an appropriate location for loading in our program below

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("TreeView Example", 40,40,400,400)
Global MyTreeView:TGadget=CreateTreeView(5,0,200,360,MyWindow)

Local Folder:int=ReadDir(BlitzMaxPath())
Local File:String

Local IconStrip:TIconStrip=LoadIconStrip("D:\My Documents on E\_Tutorials
\toolbar.bmp")
SetGadgetIconStrip(MyTreeView, IconStrip)

Repeat
   File=NextFile(Folder)
```

```
    If File=".." Or File="." Or File=Null Then
        'Do Nothing
    Else
        AddTreeViewNode(file,MyTreeView)
    EndIf
Until File=Null

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  End Select
Forever
End
```

Notice that the first image from our bitmap is becomes the default icon as the iconstrip has now been assigned to the gadget.

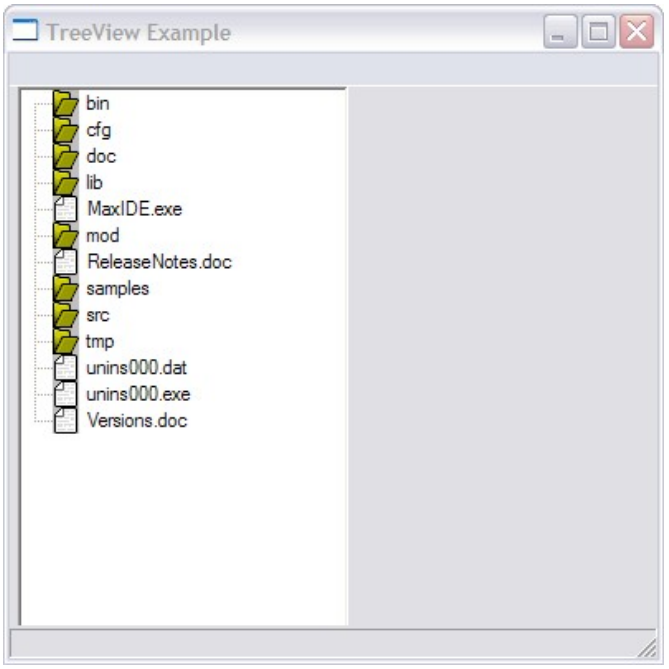


To be really clever, we can try and mimic the File Explorer look more by doing something like this:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("TreeView Example", 40,40,400,400)
Global MyTreeView:TGadget=CreateTreeView(5,0,200,360,MyWindow)

Local Folder:int=ReadDir(BlitzMaxPath())
Local File:String
Local FullPath:String

Local IconStrip:TIconStrip=LoadIconStrip("D:\My Documents on E\_Tutorials\toolbar.bmp")
SetGadgetIconStrip(MyTreeView, IconStrip)

Repeat
    File=NextFile(Folder)
    If File=".." Or File="." Or File=Null Then
        'Do Nothing
    Else
        fullPath = RealPath(BlitzMaxPath()+"/"+file)
        If FileType(FullPath)=FILETYPE_DIR Then
            AddTreeViewNode(file,MyTreeView,1)
        Else
```

```
            AddTreeViewNode(file,MyTreeView,0)
        EndIf
    EndIf
Until File=Null

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  End Select
Forever
End
```

Using the **FileType** function we can check to see whether the file is a folder or a file. We then attach the correct icon to the treeview node. Voila, we now have something looking a bit more like file explorer :)



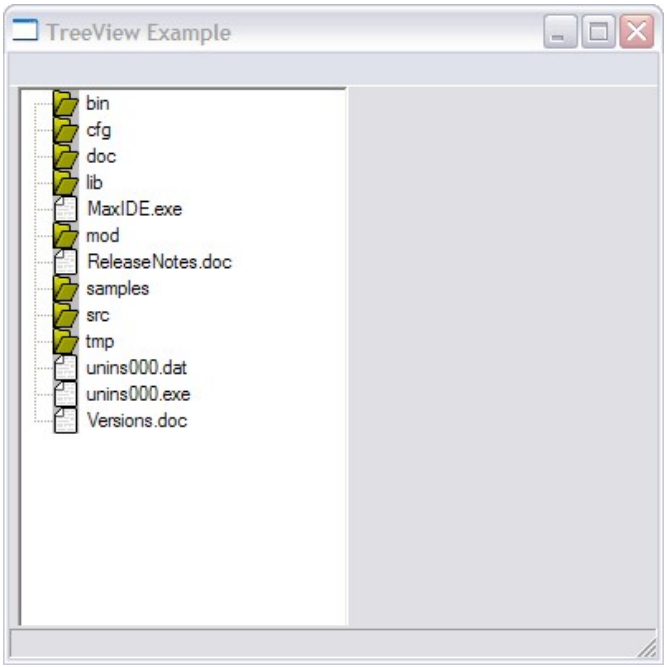
Expanding and Collapsing TreeView Nodes

We can repackage our above program a little bit differently by moving some codes into function to position ourselves for the next step.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("TreeView Example", 40,40,400,400)
Global MyTreeView:TGadget=CreateTreeView(5,0,200,360,MyWindow)

Local IconStrip:TIconStrip=LoadIconStrip("D:\My Documents on E\_Tutorials\toolbar.bmp")
SetGadgetIconStrip(MyTreeView, IconStrip)

EnumFiles(BlitzMaxPath(),MyTreeView)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  End Select
Forever
```

```
End

Function EnumFiles:Int(Dir:String, Parent:TGadget)

  Local Folder:int=ReadDir(Dir)
  Local File:String
  Local FullPath:String

  Repeat
     File=NextFile(Folder)
     If File=".." Or File="." Or File=Null Then
        'Do Nothing
     Else
        fullPath = RealPath(BlitzMaxPath()+"/"+file)
        If FileType(FullPath)=FILETYPE_DIR Then
           AddTreeViewNode(file,MyTreeView,1)
        Else
           AddTreeViewNode(file,MyTreeView,0)
        EndIf
     EndIf
  Until File=Null

End Function
```

Running the above gave us exactly the same answer.

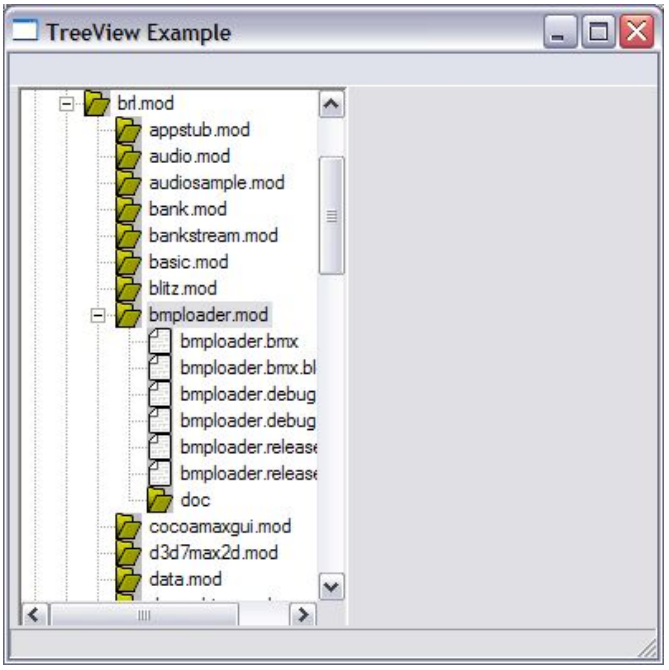


Now lets add some code (see text in bold) where, by clicking on the folders we can explore its content just like File Explorer.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("TreeView Example", 40,40,400,400)
Global MyTreeView:TGadget=CreateTreeView(5,0,200,330,MyWindow)

Local IconStrip:TIconStrip=LoadIconStrip("D:\My Documents on E\_Tutorials\toolbar.bmp")
SetGadgetIconStrip(MyTreeView, IconStrip)

EnumFiles(BlitzMaxPath(),MyTreeView)

Repeat
  WaitEvent()
```

```
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End

  Case EVENT_GADGETACTION
      Local Node:TGadget = SelectedTreeViewNode(MyTreeView)
      Local s:String=String(Node.Context)
      If s>""
         EnumFiles(s,node)
         node.context=""
      EndIf
      ExpandTreeViewNode(node)

  End Select
Forever
End

Function EnumFiles:Int(Dir:String, Parent:TGadget)

  Local Folder:int=ReadDir(Dir)
  Local File:String
  Local FullPath:String

  Repeat
     File=NextFile(Folder)
     If File=".." Or File="." Or File=Null Then
        'Do Nothing
     Else
        fullPath = RealPath(Dir+"/"+file)
        If FileType(FullPath)=FILETYPE_DIR Then
          Local handle:TGadget=AddTreeViewNode(file,Parent,1)
          handle.context=FullPath
        Else
           AddTreeViewNode(file,Parent,0)
        EndIf
     EndIf
  Until File=Null

End Function
```

With the above code, we can now explore our Blitzmax folder to our hearts content. Notice how the scrollbars automagically appear when the content of our treeview grows longer the the gadget height and/or width.



Now lets take a look at how we manage to achieve this.

```
Function EnumFiles:Int(Dir:String, Parent:TGadget)

  Local Folder:int=ReadDir(Dir)
  Local File:String
  Local FullPath:String

  Repeat
     File=NextFile(Folder)
     If File=".." Or File="." Or File=Null Then
        'Do Nothing
     Else
        fullPath = RealPath(Dir+"/"+file)
        If FileType(FullPath)=FILETYPE_DIR Then
          Local handle:TGadget=AddTreeViewNode(file,Parent,1)
          handle.context=FullPath
        Else
           AddTreeViewNode(file,Parent,0)
        EndIf
     EndIf
  Until File=Null

End Function
```

The **AddTreeViewNode** function returns a TGadget type which we have named as **handle**. The TGadget type has a field call **context** which we can use to store the full path of our folder for later use.

```
  Case EVENT_GADGETACTION
       Local Node:TGadget = SelectedTreeViewNode(MyTreeView)
       Local s:String=String(Node.Context)
       If s>""
          EnumFiles(s,node)
          node.context=""
       EndIf
       ExpandTreeViewNode(node)
```

When we double clicked on the treeview item, a GadgetAction event will be generated. We capture the selected node using the **SelectedTreeViewNode** function and then checks its context field. Note that the context field holds a generic Object type so we have to cast it into a string before we can use it as a string variable.

Remember earlier we had stored the folder path which we can now use to explore the next level using our EnumFiles function. The context field is then nulled to signify that we are done exploring this node.

The **ExpandTreeVewNode** function is then called to expand that particular node.

Summary

There are more Treeview related functions to cover which we will do in the next tutorial but for this tutorial we learnt how to

- create a treeview gadget using the **CreateTreeView** function
- add items (nodes) to the treeview gadget using **AddTreeViewNode**
- pretty up our treeview with icons using the icon parameter of **AddTreeViewNode**
- expand nodes using the **ExpandTreeVewNode** (note that a similar function called **CollapseTreeViewNode** exists as well)
- Use the EVENT_GADGETACTION eventID to capture mouseclicks on the TreeView Gagdet and the **SelectedTreeViewNode** function to retrieve the node the user clicked on.

As you can see from the above example, using these functions already allow us to create a rudimentary file explorer example.

That's all for now. Return to Main Index.