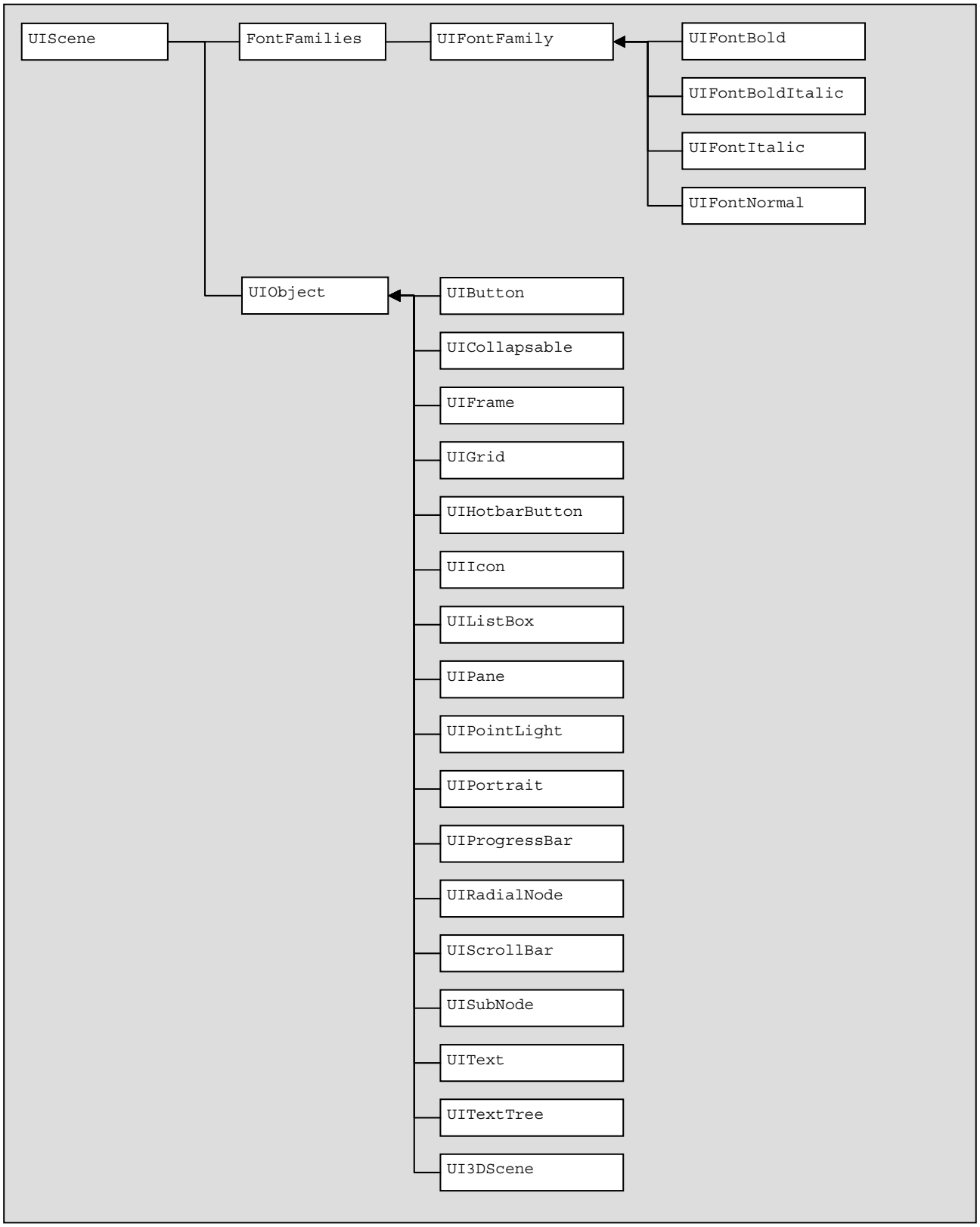
A Rough Guide To GUIs

Table of Contents

UI DOM

The Document Object Model

The following diagram represents the UI DOM (as currently understood by Sunjammer):



UI Objects

UIScene

About

The UIScene tag at the top of the XML files defines certain global parameters about the entire GUI window being defined in the file.

A scene doesn't have anything to render by default, but rather contains items that will get rendered. So a file that defines a UIScene and nothing else will not appear as anything in the game.

Some of the attributes for UIScene overlap with those of UIObject, even though UIScene does not inherit from UIObject.

Attributes

priority
This determines where scenes fall in the render order, in terms of which scenes should cover up other scenes. The valid arguments to this attribute in order of highest priority to lowest are:

- SCENE_TOOLTIP - Appears on top of any other GUI
- SCENE_GLOBAL - Used for messageboxes generally.
- SCENE_FE_FULLSCREEN - Pre-game full screen GUIs
- SCENE_INGAME_FULLSCREEN - Full screen GUIs within the game.
- SCENE_INGAME_SYSTEM - Escape menu and options screens.
- SCENE_SCRIPT - GUIs brought up by script
- SCENE_NWN1_DIALOG - The NWN1 style dialog box
- SCENE_INGAME_MENU - The popup player menu
- SCENE_INGAME - Most of the windows in-game
- SCENE_INGAME_TARGET - The target box

updaterate
The time between updates, in seconds. If no value is specified, the scene will get an update every frame.

draggable
This determines if the entire scene is draggable. This is different from dragging a single UI Object within a scene as it moves the entire window. The default setting is false.

dragregion_x
dragregion_y
dragregion_width
dragregion_height
These 4 attributes are used to define the space on the scene that can be used to drag the scene. The numbers are in pixels. If a drag region is defined using these 4 attributes, the user can only drag the scene by clicking and dragging in the defined location. If no drag region is defined, the user can drag the scene by clicking anywhere on the scene where the click will not be used by a child object of the scene.

dragresizable
Determines if this scene can be resized by dragging the edges of it. This is used in the chat window for resizing, for example. The default setting is false.

dragresizeborder
This attribute defines the size, in pixels, of the border of the scene that can be clicked on in order to resize the window.

capturemouseclicks
If set to false, then mouse clicks should pass right through the background of this scene. Useful for semi-transparent overlays and other situations where it is expected that the user be able to click through the UI Object. The default setting is true.

capturemousevents
This is pretty much a redundant attribute. It does the exact same thing as capturemouseclicks.

fullscreen
This attribute can be true or false. By default it is false. It controls what happens to this scene when the resolution of the game changes. Scenes that have this attribute set to 'true' will not have their origins moved but their dimenions will be adjusted to the new resolution. Scenes that have this attribute set to 'false' will not be resized when the resolution changes but their origin will change to stay roughly in the same part of the screen that the window appeared in before.

width
Width of the scene in pixels. The argument 'SCREEN_WIDTH' can also be used to make this scene always match the width of the full screen.

height
Height of the scene in pixels. The argument 'SCREEN_HEIGHT' can also be used to make this scene always match the height of the full screen.

x
The x origin for the window. Defaults to 0. The following string arguments are also valid:

- ALIGN_CENTER
- ALIGN_LEFT
- ALIGN_RIGHT

y
The y origin for the window. Defaults to 0. The following string arguments are also valid:

- ALIGN_CENTER
- ALIGN_LEFT
- ALIGN_RIGHT

expiretime
This is the amount of time in seconds a scene should be up before it is automatically removed from view and unloaded from memory. If 0 or not set, then the scene will never be automatically removed from view.

idleexpiretime
This is the amount of time in seconds a scene should not be visible before it should be unloaded from memory. This can be used to unload scenes that were shown once that will not be needed again anytime soon. The next time the scene is requested, it will be loaded from disk. This makes it possible to sometimes do quick testing of GUI changes by setting a short idleexpiretime, closing the window, then reopening it to see what it looks like with the new changes on disk.

autolayout
This attribute is only of interest in message-box style GUI screens. If enabled, the engine will attempt to automatically position the text, the 'okay', and the 'cancel' button in a classic message box layout. If left false, then the engine will not attempt to arrange any of the contents of the messagebox when the message box is requested. It is false by default.

modal
If set to true, then this UI scene will block access to the game or any underlying GUI screens until the window is closed. It is false by default. Note that scripts that bring up GUIs can override this parameter based on the parameters passed in to the script function.

fadein
The time in seconds this scene should take to fade in completely when opened.

fadeout
The time in seconds this scene should take to fade out completely when closed.

minwidth
This attribute is defunct and no longer does anything.

minheight
This attribute is defunct and no longer does anything.

scriptloadable
In order for a server script to have any control over this GUI at all, this attribute must exist in the UIScene tag and must be set to true. It is false by default. If false, than all script functions that attempt to manipulate a user's GUI will fail.

backoutkey
If this attribute is set true, then use of the backout key ('Esc' key by default) will cause this window to close. The backout key follows a defined set of logic for determining what to do when it is used. First it closes any open 'windows' in the reverse order that they were opened (Most recent window opened is closed first). If there are no windows to close, it clears the player's target. If the player has nothing targetted, it brings up the in-game options screen.

Callbacks

OnAdd
This callback gets executed any time the scene is made visible.

OnBackout
This callback gets executed when the user hits their Esc key and this scene is going to be closed as a result. If the backoutkey attribute is not true, then this callback will do nothing.

OnCreate
This callback gets executed when the scene is loaded from the XML file. This is not necessarily a one time thing, as it is possible for scenes to get unloaded from memory and then reloaded from the file the next time they are needed. Additional detail about this will be included later in this article.

OnDestroy
This callback gets executed when the scene is unloaded from memory.

OnRemove
This callback gets executed any time the scene is closed. Note that scenes can be closed and yet remain in memory. This in fact the default behavior. See idleexpiretime below for more information.

OnUnhandledMouseClick
This callback gets executed if a user clicks on the scene, but none of the contents of the scene did anything with the user's click.

OnUpdate
This is the callback to execute every frame.

UIObject

About

UIObject is the 'base' GUI object. You can't actually define it in XML, but it contains attributes that are common to many different UI Objects, so I'm starting here.

The following are the attributes that will be loaded for every UI Object in the XML file except for the attributes. << this makes no sense

Some attributes will be ignored or overridden depending on the situation.

Note that attributes are case sensitive.

Attributes

name
String value for the name of the UI Object. This is used internally and will be necessary for scripting to interact with this object in the future.

scalewithscene
If this is set to true, then this UI Object will have its dimensions scaled to match its parent's dimensions if the parent object gets resized. Handy for things like background images.

usescaler
This tells the code to try and use some logic when this object gets resized in determining where its new X,Y origin should be. For example, if there is a lot of space to the right of the object, it will assume this object was left-anchored and keep the object to the left side with its new position. It's kind of a confusing attribute, all I can say is try it out and see if it does what you want, and don't use it if it doesn't. :)

width
The width of the UI Object. Many things override this, for example if the object will be a box in a grid, the grid will control the width, or if the box is in a list box, the list box may override it, depending on the list box attributes, etc. This attribute can take pixel count arguments, or the following strings:

- SCREEN_WIDTH - Width will be full screen no matter what the resolution is.
- PARENT_WIDTH - Width will be the full width of its parent object, no matter what the size of the parent object is.

height
Same as the width attribute, except substitute height for width in all cases. Note that if no height or width attributes are set, the object will take on the height and width of its parent object.

x
The x origin of the object with respect to its parent UI object. This can take a pixel count, or the following strings as arguments:

- ALIGN_CENTER - Center this Object to the screen
- ALIGN_PARENT - Center this object within its parent UI Object.
- ALIGN_LEFT - Keep this object oriented to the left of its parent (x = 0).
- ALIGN_RIGHT - Push this object against the right side of the parent object.

y
The y origin of the object with respect to its parent UI object. This can take a pixel count, or the following strings as arguments:

- ALIGN_CENTER - Center this object to the screen

- ALIGN_PARENT - Center this object within its parent UI Object.
- ALIGN_TOP - Keep this object at the top of its parent object (y=0)
- ALIGN_BOTTOM - Push this object against the bottom of the parent object.

focusable
In general, this means that the object can't be clicked on. For example, if it is an object in a UIGrid, it means that object in the grid can't be clicked on. This will keep the object of being the target of an actiontarget as well (When the mouse cursor changes to indicate the user can click somewhere to perform an action). Default value is true.

ignoreevents
This is somewhat identical to setting focusable=false, but it also means that the UI Object will ignore objects being dragged on it, among some other UI events besides just mouse clicking. It defaults to false.

handleactiontarget
Setting this to true means that action targets can be used on this GUI object. Action targets actions that need to be targeted on something, indicated by the mouse cursor changing appearance. For example, clicking on a spell button, then having the cursor change to indicate you need to click on what target you want the spell cast. If this attribute is true, then the GUI object will be treated as a valid target for actions to be performed on. It defaults to false.

draggable
This indicates that the UI Object can be dragged and dropped, such as hotbar buttons. Defaults to false.

DefaultTooltip
This is a default tooltip STRREF that is used in case a custom tooltip callback is not really necessary.

hidden
Whether or not this object is visible. Defaults to true..

disabled
Whether or not this object is enabled. Applies mostly to buttons, but for the most part, no UI Object that is disabled will accept user input. Defaults to false.

capturemouseclicks
If set to false, then mouse clicks should pass right through this object. Useful for semi-transparent overlays and other situations where it is expected that the user be able to click through the UI Object. This also applies to MouseEnter and MouseLeave. If an object is set to not capture mouse clicks, then objects underneath of it cannot be moused over.

update
This determines if this UI object receives calls to an OnUpdate callback. Defaults to false. If set to true, then this UIObject will have its OnUpdate callback called on every frame, or slower if an updaterate is defined.

UpdateRate
This attribute can be used to tune the update rate of a UI Object. If a particular object doesn't need to be updated every frame, then set an update rate here to slow it down. The number value is in seconds and floating points are okay (0.5 for half second, for example).

hotbartype
This is an advanced attribute that is used for determining how the hotbar should treat another UI Object being dragged onto it. The currently valid values for this attribute are:
- HOTBAR_NONE - The hotbar should ignore this.

- HOTBAR_ITEM - This is an item icon being dragged, such as from inventory.
- HOTBAR_KNOWNSPELL - This item being dragged is something from the spellbook.
- HOTBAR_SPELL - This item is a spell from somewhere else, such as another hotbar slot.
- HOTBAR_FEAT - This item is a feat from somewhere, such as the feat listing or another hotbar slot.
- HOTBAR_BUTTON - This is an empty hotbar button.
- HOTBAR_SKILL - This is an activateable skill, such as from the skills pane or another hotbar slot.
- HOTBAR_TOGGLEMODE_BUTTON - This is a button used to toggle some kind of mode.
- HOTBAR_BARTER - This is used to flag barter grid items as not being able to be dragged to the hotbar.
- HOTBAR_DM_COMMAND - The item being dragged is a DM Client command.
- HOTBAR_VM_COMMAND - The item being dragged is an emote command.
- HOTBAR_DM_CREATOR - The item being dragged is an entry from the DM Client Creator.

Scale*ToParent
This was a set of attributes that was going to allow more control over how an object behaved when its parent got scaled. But in looking at the code, it looks like they've been gutted and no longer do anything.

MouseOverSFX
The sound effect to play when this UI Object gets moused over.

alpha
How transparent this object should be by default.

dontrendermousegrab
'Mouse Grab' means item being currently dragged. Normally the mouse grab is rendered last of all in order to keep it on top. There are situations where this behavior wasn't desired, though I can't remember what. When set to true, it means that this object needs to not get rendered last like normal dragged objects, but rather is rendered when it would normally be.

SetDataInt
This stores an 'int' value in the UI Object's data element at int index 0.

SetDataFloat
This stores a 'float' value in the UI Object's data element at float index 0.

SetDataString
This stores a 'string' value in the UI Object's data element at string index 0.

hideoverride
If true, this attribute sets a UI Object hidden and keeps it that way until a script says otherwise via the SetGuiObjectHidden() script function.

This can be used to keep an item hidden that the GUI may try to make visible regardless what a script tries to do, such as a row in a list box.

As a result, objects that are set hidden via SetGuiObjectHidden() will STAY hidden no matter what, unless SetGuiObjectHidden() is later used to make it visible.

For example, if a new icon is added to a screen and `hideoverride=true` in the XML, it will remain hidden on that screen no matter what the GUI might try to do with it until I call SetGuiObjectHidden() with a false parameter to make it visible again, at which point the GUI once again has control over making that object hidden or unhidden.

Callbacks

OnGainedFocus
This callback is executed when an object gains focus. This applies mostly to buttons that got clicked on or editable text fields that got clicked in.

OnLostFocus
This callback is executed when an object loses focus. This applies mostly to buttons that had been previously clicked on or editable text fields that had been previously clicked in.

OnMouseDrop
This callback is executed on a UI Object when it gets dragged and then dropped. Upon being dropped, this callback will be executed.

OnMouseDropFailed
This callback is executed if a UI Object is dragged and then dropped but not dropped over another UI Object.

OnMouseDropReceived
This is the UI Callback to execute when a draggable UI object gets dropped on this UI object. This is not executed by the object being dropped, but rather the object that got something dropped ON it.

OnMouseEnter
This callback is executed when the mouse cursor is moved over this object.

OnMouseLeave
This callback is executed when the mouse cursor is moved off of the object after having been moved onto it.

OnRadialRequest
This is the callback that gets executed if the user brings up the context sensitive menu on this UI object.

OnResize
This callback is executed any time the UI Object gets resized for any reason.

OnTooltip
This callback gets executed if the mouse hovers over this object long enough based on the user's preference for the tooltip delay.

OnUpdate
The callback to be executed when this object receives its update.

UIPane

About

UIPane is the generic 'container' for other UI Objects. Panes can't be rendered, but they can contain 'child' objects which can be renderable objects.

Every scene has a 'rootpane', which is automatically defined by the engine when a new UIScene is loaded from a XML file.

Child objects within a pane will be positioned with their x,y origin relative to the origin of the UI Pane that contains the child. UIPanes can also simulate buttons with the tupple attribute, so many of the attributes for UIPane are similar to those found under UIButton.

Attributes

MouseDownSFX
Sound effect to be played when the Pane is clicked on. This only means anything if the Pane is a tupple style pane.

MouseDragSFX
Sound effect for dragging a tupple style UI Pane object.

MouseDropSFX
Sound effect when releasing a dragged tupple style UI Pane object.

MouseUpSFX
Similar to a button up sound effect, this only means anything if the Pane is a tupple style pane.

tupple
Setting this attribute to true makes the entire UI Pane behave like a button rather than just a container. This makes it possible to design more complicated buttons than normal, default UIButton tag allows for such as having a defined layout or multiple icons within the same button, etc. Once a Pane becomes a tupple, it can be clicked on, gain focus, lose focus, become enabled, become disabled, etc., just like a normal button. Note that it is still possible to place buttons within a tupple style pane that should work fine. So you could have a listbox row that is a tupple style pane that also has a toggle button on every row, for example.

Callbacks

OnLeftClick
The callback to be executed when a tupple style UI Pane is left clicked on.

OnLeftDoubleClick
The callback to be executed when a tupple style UI Pane is double left clicked on.

OnRightClick
The callback to be executed when a tupple style UI Pane is right clicked on.

OnRightDoubleClick
The callback to be executed when a tupple style UI pane is double right clicked on.

UIButton

About

UIButton is the generic, clickable object one expects to find on any PC game interface. It also acts as a 'restricted' container of sorts, as it can contain specific UI objects within it.

A button is made up of several child objects contained in a single pane. These pieces can be defined explicitly or left up to the engine to define.

The child objects that a UIButton can contain are:

- Up to 1 UIText field. If a UIText object is not defined as a child object in the XML, the engine will create one automatically when creating the UIButton.

- Up to 11 UIFrames. A UIFrame contained within a button must have a state attribute that only applies when specifically contained within a button. The following arguments are valid for the state attribute:
    - up
    - down
    - disabled
    - focused
    - hilited
    - hifocus
    - header
    - hiheader
    - downheader
    - base

  These states can mean different things depending on the context of the button, so I'll just make some notes about a few of them. 'header', 'hiheader', and 'downheader' are the states for a button that is to be the header of a UICollapsable object, which will be explained further in a later article. 'base' is useful for when using state frames that leave some of the 'base' button visible, such as hotbar button icons.

- An unlimited number of UIIcons that will be added to the 'overlay' list for that button. Overlays are extra icons that can be made visible by the engine that stay with the button. At this time, there is no way to manipulate these overlays via script.

Attributes

style
This is a 'style' name from the stylesheet.xml file that this button should adopt for its default parameters. At this time there is not support for custom styles.

repeatcallback
If this attribute is set to true, the OnLeftClick callback will get executed every frame as long as the left mouse button is held down on it. On the first update the OnLeftClick will execute on the mousedown. There will then be a 0.5 second pause, then after that the OnLeftClick callback will be executed every frame until the left mouse button is released.

groupid
This is a group number for radio buttons. Group IDs are global for the entire XML file that contains the button. That is to say, there can only be one group of radio buttons with the group id of '1', or '2', etc. All buttons that share the same group id form a 'set' of radio buttons. Only one button in that set can be selected at a time, all the rest are non-selected. Selecting any one button in the set will unselect the previously selected radio button.

groupmemberid
This gives the UI Button a unique within a set of radio buttons. There should only be one button with a specific groupmemberid in a set of buttons. Ideally these IDs should start with 1 and increment from there, but that is not completely necessary.

buttontype
This attribute accepts only 'radio' or 'check' as arguments, defining the button as a radio button or checkbox style button.

color
This attribute accepts color strings. These colors will blend with any other colors found in the textures of the button. So setting this to grey, for example, will make a grey-scale style icon for the button.

disabledcolor
This attribute accepts color strings. This will be the blending color for when the button is put into a disabled state.

disabledtextcolor
This attribute accepts color strings. This color will change the color of any text contained in the button's text field if the button is put into a disabled state.

strref
This is a STRREF that represents the string that should appear in the button.

text
This is hard coded text that should appear in the button. It overrides the strref attribute if it is present.

Callbacks

OnSelected
This callback is applicable to radio buttons and toggle boxes. If the button is a toggle box, this callback is executed when the toggle box is activated. If the button is a radio button, this callback is executed when the button becomes the selected radio button.

OnUnselected
This callback is applicable to radio buttons and toggle boxes. If the button is a toggle box, this callback is executed when the toggle box is de-activated. If the button is a radio button, this callback is executed if thi sbutton was the selected radio button, but now another radio button has been selected.

UIFrame

About

UI Frame is used for borders and backgrounds for other UI Objects. UIFrames are imbedded in buttons and a few of the container style UI Objects that will be discussed later.

Attributes

bottom
bottomleft
bottomright
fill
left
right
top
topleft
topright
The texture pieces that make up a frame. They take a texture name as an argument.

mhtop
Stands for MirrorHorizontalTop. Defaults to false. If true, the Top Left Corner will be mirrored to the Top Right Corner. If combined with mvright, then the Top Left Corner will be mirrored to the Right Bottom Corner.

mhbottom
Stands for MirrorHorizontalBottom. Defaults to false. If true AND mvright is true, then the Top Left Corner will get mirrored to the Bottom Right Corner. If true AND mvright is false, then the Bottom Left Corner will get mirrored to the Bottom Right Corner.

mvleft
Stands for MirrorVerticalLeft. Defaults to false. If true, the Top Left Corner will get mirrored to the Bottom Left Corner. If true AND mhbottom is true, the Top Left Corner will get mirrored to the Bottom Right Corner.

mvright
Stands for MirrorVerticalRight. Defaults to false. If true AND mhtop is true, the Top Left Corner will get mirrored to the Bottom Right Corner. If true AND mhbottom is false, the Top Right Corner will get mirrored to the Bottom Right Corner.

mhside
Stands for MirrorHorizontalSide. Defaults to false. If true, the Left Side will get mirrored to the Right Side.

mvside
Stands for MirrorVeritcalSide. Defaults to false. If true, the Top Side will very mirrored to the Bottom Side.

maside
Stands for MirrorAllSides. Defaults to false. If true, the Left Side will get mirrored to the Top Side, Right Side, and Bottom Side.

fillstyle
This tells the engine to fill in the center space of the frame with the fill texture using different algorithms. This attribute takes the following string arguments:

- **center** - Center the 'fill' texture.

- **stretch** - Stretch the 'fill' texture to take up the full center.
- **tile** - Repeat the 'fill' texture to fill up the center.

border
This is the thickness of the border pieces in pixels. Note that it defaults to 0. If no border thickness is set, then all the textures besides 'fill' won't show up when the UIFrame gets rendered.

color
This takes color strings to define the color that should be blended with the frame textures.

Notes

If a UIFrame doesn't have a height or width defined by its attributes, it automatically inherits the height and width of whatever object contains it.

UIText

About

UIText is the GUI object used for containing and rendering text. It is one of the more complicated UI Objects to work with.

Attributes

fontfamily
This defines which font should be used via a string argument. The fonts are defined in fontfamily.xml. The 'name' attribute for the different UIFontFamily elements in fontfamily.xml is how they are identified here. If this attribute is not set, then the text field will use the 'Default' font family defined in fontfamily.xml.

nextcontrol
Which control should have focus if the user hits tab while editing this UIText field. It only really makes sense to put UIText field names as the arguments to this attribute, as focus doesn't have any significant meaning for any other type of UIObject. Note that the nextcontrol must be in the same UIScene as this text field.

prevcontrol
Which control should have focus if the user hits shift+tab while editing this UIText field. It only really makes sense to put UIText field names as the arguments to this attribute, as focus doesn't have any significant meaning for any other type of UIObject. Note that the prevcontrol must be in the same UIScene as this text field.

filter
This attribute can be used to limit what text the user can enter into this text field when editing it. By default, any input character is valid. It takes strings as its arguments. The following strings are handled:

- **alpha** - Only letters can be entered in this field.
- **numeric** - Only numbers can be entered in this field.
- **alphanumeric** - Only letters and numbers can be entered in this field.
- **signed_numeric** - Only numbers in this field, except the '-' character, which may only be entered at the front of the string.

maxnumber
This number limits the max number that a user can enter when editing this text field. It only means anything if the text field's filter is set to numeric or signed_numeric.

allowspace
If this attribute is false, then any space characters will not be allowed in this field, including tabs or newlines. It defaults to false.

allowpunc
This attribute only means anything if the filter type is set to alpha or alphanumeric. It allows graphical punctuation characters to be entered in the text field. If no filter type is specified, then this attribute doesn't do anything. It defaults to false, so must be set to true if the intent is to allow users to enter punctuation style characters into this text field.

password
Defaults to false. If set to true, then the characters in this string will be replaced with * marks when this text field gets rendered.

sanitized
Defaults to false. If this attribute is set to true, then any string entered in this text field will be run through the built in swear-filter.

style
This is a modifier to the font that is being used. It takes a string or number as its argument. The following arguments are supported:

- **style 1** or **normal** - as defined by the UIFontNormal element in fontfamily.xml
- **style 2** or **bold** - as defined by the UIFontBold element in fontfamily.xml
- **style 3** or **italic** - as defined by the UIFontItalic element in fontfamily.xml
- **style 4** or **bolditalic** - as defined by the UIFontBoldItalic element in fontfamily.xml

indent
This is the number of pixels to indent the first line of any text. It defaults to 0.

hangingindent
This is the number of pixels to indent all lines besides the first line in a multi-line text field. It defaults to 0.

align
This is how the text should be aligned in the text field. The following string arguments are handled: left, right, center. Defaults to left.

hilite
Defaults to false. If set true, then the text in this text field will hilight when moused over.

valign
This is how the text should be aligned vertically within the text field. The following string arguments are handled: top, bottom, middle. Defaults to top.

poscenter
This attribute is defunct and no longer does anything.

sizetofit
Defaults to false. If set to true, then the text field will extend its height downward as far as possible in order to contain the length of text within it. This is an old attribute and should be replaced by just setting the height of the text field to DYNAMIC.

buttonoverlay
This only means anything if the text field is a child element of a UIButton element. If left to the default of false, then this text field will become the normal text field that shows up in buttons. If set to true, it becomes Overlay Text. At this time, Overlay Text can only be manipulated by the engine.

editable
Defaults to false. If set to true, then the user can edit the text within this text field by clicking it to place their cursor and typing like normal.

selectable
This attribute does nothing at this time.

maxlines
The max number of lines this text field can contain. Setting this attribute to '1' will keep the engine from trying to break the text into multiple lines. Text that will not fit into the text field after the maximum number of lines has been reached will not get rendered. If not defined, the

engine will try to wrap a string and fit it into the space available no matter what, which can sometimes result in no text at all appearing due to spacing errors.

multiline
This attribute toggles whether or not a user can hit enter to start a new line of text within a editable text field. It requires that allowspaces be also set to true or else it will seem to do nothing. It defaults to false. When false, the OnReturn callback will get called when the user hits enter, instead of a new line being inserted.

returnrestricted
This attribute can be a bit confusing. It defaults to false and only means anything when used with multiline=true. What happens is that if this attribute is set to 'true', the text field will accept multiple lines of input and wrap them automatically, but if the user hits their return key, the OnReturn callback gets called instead of inserting a newline. An example of this behavior can be found in the chat input box.

uppercase
Defaults to false. If true, all text rendered in this text field will be rendered in uppercase.

maxlength
This limits the number of characters that can be typed into a text field by the user. Once the user has entered the max limit of characters, no additional characters can be input. If no maxlength is set, the user can keep typing indefinitely.

color
A color to blend with all of the characters within the text field. This can sometimes conflict with the use of the color tags in text fields.

highlightcolor
This attribute defines the color that should be blended with the text when doing a mouse-over style highlight.

selectioncolor
This attribute does nothing at this time.

highlightonmouseover
This is just a clone of the hilite attribute mentioned above.

text
This is for hard coding the text that should show up in this text field. Overrides strref.

strref
This is for coding which string ref should show up in the text field. It is overridden by text.

Callbacks

OnLostFocus
This callback is executed when the cursor is taken out of the text field due to the user clicking elsewhere.

OnReturn
This callback gets executed when the user hits Enter while the cursor is in this text field. Note that if the text field is set with multiline=true and restrictedreturn=false, then this callback will not get executed.

UIListBox

About

The **UIListBox** is a container similar to UIPane and anything listed under UIPane that makes sense for UIListBox probably does the same thing under UIListBox. So I'm going to stick to discussing the attributes that are unique to UIListBox. Refer to UIPane for the basic attributes (Like X, Y, Height, Width, etc).

There's a couple different ways list boxes are used:

- A list box can be used for displaying multiple rows of objects. Things like the Feat List or the lines of text in the chat windows use it this way. This is accomplished by just adding multiple objects to the list box. Typically these objects are the same size, but that is not a requirement.

- A list box can also be used for containing a single large text field and being able to scroll over that text field. This can be seen in places like the class and race descriptions on character creation. This approach is accomplished by having a single text field within the list box, making the height of the text field dynamic, and changing the text of the text field. The listbox will allow for scrolling over the large text field.

Another note, I may use the term 'control' a lot in this write up. 'Controls' in this context are the contents of the list box. These are different than 'children' of the list box, which would include every object contained by the listbox, like the scrollbar. I will try to call controls 'rows' to make it more clear, but if I slip up and call it a 'control,' I'm talking about a row.

Attributes

xPadding
The amount of padding that the listbox should put between the edges of the listbox and the contents of the listbox. This padding is applied equally to both sides. So if the padding is 5, and the ListBox is 100 pixels wide, then the ListBox will constrain the rows to a max width of 90.

yPadding
The amoutn of padding that the list box puts between each row. This includes padding between the top row and the top of the list box and the bottom row and the bottom of the list box as well.

snaptobottom
Setting this to 'true' will make it so that any time the text field in the listbox gets resized, the listbox will snap to the bottom to show the bottom of the text. This is used mostly for situations where text keeps getting added to a text field and you want the list box to keep scrolling to the bottom to display all the text. This is not how the chat window works, BTW.

scrollsegmentsize
This is how many pixels you want the listbox to scroll for each 'scroll click' on the part of the user. The parameter is a number, the default vaule is 1.

unequalcontrols
This is a somewhat complicated attribute. If you don't have any preference, it's usually best to just set it to true. The purpose of it is to speed up performance on list boxes where each row is the same height. So if your list box will definitely have each row with the same height, then set this to false to speed up scrolling with that list box.

showpartialchild
This tells the list box if it should just clip the bottom row if it can't show the entire row. Defaults to true, parameters are 'true' or 'false.'

scrollbaronright
Setting this to true moves the scrollbar over to the right side of the list box. Setting it to false will move the scroll bar to the left side of the list box. The default behavior is 'true'.

selectonleftclick
This tells the game that you want the user to be able to click on the rows of the list box. This means that the list box is intended to contain a number of rows and you want the user to pick a row. It is false by default. If false, then clicking on a row will perform the 'LeftClick' callback if you click on a button row, or will set the text field as the focus if there is just a single large text field in the list box. If by true, clicking on the row will still execute the OnLeftClick callback for the button if there is one. It will also set that row as selected.

hidescrollbar
If this is set to true, then the scrollbar will be hidden no matter what. This is for the rare case you may want text to scroll by but don't care to have the user being able to scroll back up. Defaults to false.

hidescrollbarwhennotneeded
If this is set to true, the scrollbar will be hidden unless there are enough contents within the list box to require scrolling to see them all. Defaults to false.

Notes

If the engine finds a UIFrame nested within a ListBox while loading the XML, it treats that Frame as a border for the full size of the list box. I'm not sure if this ever saw any actual testing though, as I believe we usually just added a separate UIFrame to the Scene to act as the border.

If the engine finds a UIScrollBar nested within a ListBox while loading the XML, that scrollbar will become the scrollbar for the ListBox and be attached to the left or right depending on the attributes.

A listbox can contain the following types of UIObjects as its rows:

- Button, Label,
- Hotbarbutton,
- Text,
- Collapsable,
- Pane,
- Grid,
- Icon,
- TextTree

If the object found has an attribute called 'prototype=true', then that object becomes the ProtoType for the list box. The prototype defines all the attributes, styles, etc., that each row that is added dynamically to the listbox should have. Unfortunately, this is something better experimented with to understand rather than me trying to explain. :) But in brief, find a listbox that has a prototype object in the game, and see what effect changing the prototype's attributes have on the way the listbox rows look in the game.

UI Functions

UIButton_OnUpdate_ControlSelected

This callback takes a single parameter. It's only purpose is to enable or disable a button based on whether or not the user has selected a row in a List Box.

The single parameter is:
sListBoxName = The name of the list box we want to check for selection.

If the user has a row selected in the listbox named in the parameter, then the UIButton will be set to Enabled.

If the user does not have a row selected in the listbox named in the parameter, then the UIButton will be disabled.

UIObject_Input_ActionTargetScript

This callback will one of the most powerful new tools when it comes to creating custom actions in the game.

UIObject_Input_ActionTargetScript(), new in 1.06, will let you make 'GUI actions' similar to the 'actions' that are created when you click on a spell or feat button to cast it, or on a button in the DM Choser that then has you select a target to do the action to.

Once the user clicks on a valid target, a bunch of data will be sent to the server along with a request to execute a script associated with the click.

Anyway, enough summary, here's the details:

```
UIObject_Input_ActionTargetScript("sValidTargetTypes", nValidCursor, nInvalidCursor,
nSpellTargetIndex, bUseHostile,"sScriptName", script parameter list...)
```

sValidTargetTypes
This parameter is a string that will list the types of objects that will be valid to click on. If you enter a "", that means all targets are valid for this action. Otherwise, you can enter a list, seperated by commas or any other deliminator you want to use. The valid target names are:

- self
- creature
- ground
- item
- door
- placeable
- trigger

For example: "creature,placeable" would mean that valid targets would include clicking on placeables and clicking on creatures.

nValidCursor
This is an integer constant to point to one of the cursors that the engine understands. You can only really use the cursors compiled into the game's executable. I will make a list of those constants at the end of this post. This is the cursor that you want to use when the cursor is over a valid target.

nInvalidCursor
Just like the nValidCursor parameter, except this is the cursor you want displayed when the mouse is over an invalid target. For example, if my valid targets were creatures and placeables, but the mouse was over open ground, this is the cursor that would be shown.

nSpellTargetIndex
This is a reference into the spelltarget.2da to indicate that you want to use one of the spell targeting projected textures like the ones used when targeting Area of Effect spells. If you do not want an AoE texture around the cursor, just enter -1 for this parameter.

bIsHostile
This tells the engine if you want to use the Hostile Texture column from spelltarget.2da or not. If you entered -1 for nSpellTargetIndex, then this parameter doesn't mean anything. You must enter either "true" or "false", even if you entered -1 for nSpellTargetIndex.

sScriptName
The name of the script you want to follow. This follows the same restrictions that UIObject_Misc_ExecuteServerScript() usese, in that the script name must start with 'gui_', or else the server will reject the request.

Parameter List
This is an unlimited list of parameters just like ExecuteServerScript() supports. However, there are a few extra parameters that this function can support:

- target.object - This will replace the parameter with the OBJECT_ID of the object that the user clicked on.

- target.x - This will replace the parameter with the X value of the position that the user clicked on as a float. If the user clicked on a specific object, instead of open ground, then this X will be the position of that object as far as the client knows. Since the client could be a few milliseconds out of synch with the server, the server may consider this object at a slightly different position than the client does.

- target.y - Same as X, except for the Y value of the position vector.

- target.z - Same as X, except for the Z value of the position vector.

For example, if I wanted to create a DM Action button that deleted whatever placeable the DM targetted as well as all placeables within 15 yards of the target, I would create a UIButton with the following callback:

```
OnLeftClick=UIObject_Input_ActionTargetScript("placeable",51,53,17,"true","gui_eraseplace
ables","target:object")
```

The number 51 would be for the CURSOR_KILL cursor, the number 53 would be for the CURSOR_NOKILL cursor, the number 17 would be for row 17 in spelltarget.2da whcih is a large circle with a 30 yard width (so 15 yard radius), and the "true" is because I want it to use the hostile texture out of that 2DA. I want to use the clicked on placeable as my starting point, so I go ahead and pass that along as a parameter as well. It will be passed into gui_eraseplaceables as either a string or an int, depending on what parameter I have that script expecting.

Anyway, the list of cursor constants is:

1	MOUSECURSOR_DEFAULT
2	MOUSECURSOR_DEFAULT_DOWN
3	MOUSECURSOR_WALK
4	MOUSECURSOR_WALK_DOWN
5	MOUSECURSOR_NOWALK
6	MOUSECURSOR_NOWALK_DOWN
7	MOUSECURSOR_ATTACK
8	MOUSECURSOR_ATTACK_DOWN
9	MOUSECURSOR_NOATTACK
10	MOUSECURSOR_NOATTACK_DOWN
11	MOUSECURSOR_TALK
12	MOUSECURSOR_TALK_DOWN
13	MOUSECURSOR_NOTALK
14	MOUSECURSOR_NOTALK_DOWN
15	MOUSECURSOR_FOLLOW
16	MOUSECURSOR_FOLLOW_DOWN
17	MOUSECURSOR_EXAMINE
18	MOUSECURSOR_EXAMINE_DOWN
19	MOUSECURSOR_NOEXAMINE
20	MOUSECURSOR_NOEXAMINE_DOWN
21	MOUSECURSOR_TRANSITION
22	MOUSECURSOR_TRANSITION_DOWN
23	MOUSECURSOR_DOOR
24	MOUSECURSOR_DOOR_DOWN

25	MOUSECURSOR_USE
26	MOUSECURSOR_USE_DOWN
27	MOUSECURSOR_NOUSE
28	MOUSECURSOR_NOUSE_DOWN
29	MOUSECURSOR_MAGIC
30	MOUSECURSOR_MAGIC_DOWN
31	MOUSECURSOR_NOMAGIC
32	MOUSECURSOR_NOMAGIC_DOWN
33	MOUSECURSOR_DISARM
34	MOUSECURSOR_DISARM_DOWN
35	MOUSECURSOR_NODISARM
36	MOUSECURSOR_NODISARM_DOWN
37	MOUSECURSOR_ACTION
38	MOUSECURSOR_ACTION_DOWN
39	MOUSECURSOR_NOACTION
40	MOUSECURSOR_NOACTION_DOWN
41	MOUSECURSOR_LOCK
42	MOUSECURSOR_LOCK_DOWN
43	MOUSECURSOR_NOLOCK
44	MOUSECURSOR_NOLOCK_DOWN
45	MOUSECURSOR_PUSHPIN
46	MOUSECURSOR_PUSHPIN_DOWN
47	MOUSECURSOR_CREATE
48	MOUSECURSOR_CREATE_DOWN
49	MOUSECURSOR_NOCREATE
50	MOUSECURSOR_NOCREATE_DOWN
51	MOUSECURSOR_KILL
52	MOUSECURSOR_KILL_DOWN
53	MOUSECURSOR_NOKILL
54	MOUSECURSOR_NOKILL_DOWN
55	MOUSECURSOR_HEAL
56	MOUSECURSOR_HEAL_DOWN
57	MOUSECURSOR_NOHEAL
58	MOUSECURSOR_NOHEAL_DOWN
59	MOUSECURSOR_RUNARROW
75	MOUSECURSOR_WALKARROW
91	MOUSECURSOR_PICKUP
92	MOUSECURSOR_PICKUP_DOWN
99	MOUSECURSOR_CHATBOX_SIZING
125	MOUSECURSOR_NODEFAULT
126	MOUSECURSOR_NODEFAULT_DOWN
128	MOUSECURSOR_MAP
127	MOUSECURSOR_MAP_DOWN
130	MOUSECURSOR_NOMAP
129	MOUSECURSOR_NOMAP_DOWN
131	MOUSECURSOR_WAIT_DOWN
132	MOUSECURSOR_WAIT

UIObject_Misc_ExtractData

```
UIObject_Misc_ExtractData(sourceobject, datatype, index, destinationvar)
```

This callback can be used to extract the data imbedded by the engine within UI Objects.

sourceobject
This parameter is used to define which UI Object we are extracting data from. It accepts the following options:

- **selected:*listboxname*** - By replacing 'listboxname' with the name of the listbox you're interested in, you can tell this callback to use the selected row within the list box as the source of the data.

- **self:** - By using 'self:', you are telling the callback to use the object that invoked the callback as the source from which to extract the data.

- **context:uiobject** - By using 'context:uiobject:' you are telling the callback to use the UI Object that invoked the context menu. This obviously only means anything if the callback is being called from a context menu node. If called from anywhere else, the callback will do nothing.

- ***objectname*** - If you enter any string besides the above 3 options, the callback will search for a UIObject within the same scene as the object executing this callback that has that name.

datatype
The datatype tells the callback which data type it should be extracting from the UIObject. Note that regardless of what the original data type is, the data type that gets stored in the GUI Variable at the end will always be a string. The data type values are:

- bool ('true' or 'false' are the strings that will get stored in the GUI variable)
- int
- float
- string
- strref
- objectid

Again, it's important to remember that no matter what the originating data type is, the result will be stored as a string variable within the GUI variable specified.

index
This is the index of the data to extract. If the index is invalid for the UIObject, a default value will get stored in the GUI variable (Either "" or "0" or "false" depending on what the originating data type was).

destinationvar
This can be a local:# or global:#. The type of the stored data will always be a string.

Example 1

The grid icons in the inventory grid have the following data imbedded in them:

OBJECT_ID(0) = The object currently displayed in that icon
OBJECT_ID(1) = The object that was dispalyed in that icon on the previous frame
int(0) = The stacksize for that object on the previous update frame
int(1) = Indicates if the item is identified or not
int(2) = Indicates if the item is usable or not
int(3) = Indicates if the item is ghosted or not

It would be possible for me to change the inventory screen to drop items if I double right click on them.

I would edit the 'InvPrototypeButton' inside of inventory.xml and add:

```
OnRightDoubleClick0=UIObject_Misc_ExtractData("self:","objectid",0,local:3)
OnRightDoubleClick1=UIObject_Misc_ExecuteServerScript("gui_dropitem",local:3)
```

When I right double click that item, the first callback will extract the Object ID of the item and place it in local:3 as a string that looks like: "32144" or whatever the object ID is.

Then the 2nd callback would execute, asking the server to fire off the "gui_dropitem" script, and it would pass the 32144 value in as a parameter to the script (Either as a string "32144" or an integer 32144, depending on what the parameter type the script is expecting).

Now I will be adding an IntToObject() script funciton that will make it possible to take that int parameter and turn it into an object parameter. At that point, you could use the DropItem action and have the player drop that object.

Note that in this example, I would've had to remove the OnRightClick call to the context menu for the item button or it would end up blocking the double right click (Due to the context menu popping up and blocking the second click).

Example 2

This example is a lot more complicated:

The spell buttons in the game usually have the following pieces of data:

int(0) = The spell ID that the button represents
int(1) = The multi-class index that can cast that spell
int(2) = Not used anymore, so nothing.
int(3) = The meta magic feat, if any
int(4) = Whether or not this is a spontanously cast spell
int(5) = Whether or not this is a domain spell
int(6) = Whether or not this is a cheat-command spell cast

The values for int(3), the meta magic feat, are:

1 = Empower
2 = Extend
4 = Maximize
8 = Quicken
16 = Silent
32 = Still
64 = Persistent
128 = Permanent

The warlock invocations are also meta-magic feats, as follows (Note, I'm not sure how many of these are actually in the game, I'm just going down the list of constants in the code):

256 = Draining Blast
512 = Eldritch Spear
1024 = Frightful Blast
2048 = Hideous Blow
4096 = Beshadowed Blast
8192 = Brimstone Blast
16384 = Eldritch Chain

32768 = Hellrime Blast
65536 = Bewitching Blast
13102 = Eldritch Cone
262144 = Noxious Blast
524288 = Vitrioloic Blast
1048576 = Eldritch Doom
2097152 = Utterdark Blast

Let's say I want to make my own custom Examine Spell command for the spells_known.xml listing of spells.

First off, let's add our own custom radial menu (Another new feature in 1.06).

In spells_known.xml, I would go to the SPELLPANE_PROTO pane and add a

```
OnRightClick=UIObject_OnRadial_DisplayCustomRadial("root-customspell")
```

The 'root-customspell' will be the name of a new root node that I would then add to the contextmenu.xml

For now, that root node will only have 1 sub node called: node-spellexamine

node-spellexamine will be the name of a new radial node that I would make with the following callbacks:

```
OnLeftClickExpanded0=UIObject_Misc_ExtractData("context:uiobject","int",0,local:5)
OnLeftClickExpanded1=UIObject_Misc_ExecuteServerScript("gui_spellexamine",local:5)
OnLeftClickCollapsed0=UIObject_Misc_ExtractData("context:uiobject","int",0,local:5)
OnLeftClickCollapsed1=UIObject_Misc_ExecuteServerScript("gui_spellexamine",local:5)
```

Now when I go into the game and bring up the Spells Known screen and right click on the spell icon for a spell in the list, I will get a context menu with 1 item called 'Examine'. Clicking on that will first call: UIObject_Misc_ExtractData() which will extract the spell ID from int(0) on the button I brought the context menu up on and place that ID in local:5

Then it will fire `UIObject_Misc_ExecuteServerScript()` and pass along the spell ID (as stored in local:5) as the parameter to the script: gui_spellexamine

UIObject_Misc_StoreObjectData

The only valid parameter to this callback is a local or global GUI variable index. localguivars are expressed as local:#, where # is the index for the variable.

globalguivars are expressed as global:#, where # is the index for the variable.

This callback takes the engine-level data in a UIObject and stores it in a global or local gui variable. The engine-level data will vary from UIObject to UIObject. It is also impossible to specify which data item you want to set the GUI variable to, if the UIObject has multiple items of data.

This is a really old callback and is of limited use to custom GUIs at this time. My intent is to write a new callback that accomplishes what this callback does, but that takes more parameters in order to more accurately specify what data you are interested in from the UIObject

UIObject_Tooltip_DisplayTooltipString

```
UIObject_Tooltip_DisplayTooltipString
```

Of the 9 parameters only the first 4 parameters are required:

Message
The literal string to display in the tooltip

Xloc
The X origin for where the tooltip should appear. This can be a pixel location, or you can use the following two strings to make it relative: MOUSE_X will place the X origin of the tooltip near where the mouse is. OBJECT_X will place the tooltip next to the UIObject that is being moused over, if any. At first, it will try to place the tooltip to the right side of the UIObject. But if there is not enough screen space to fit the tooltip there, it will place it on the left side of the UIObject.

YLoc
The Y origin for where the tooltip should appear. This can be a pixel location, or you can use the following two strings to make it relative: MOUSE_Y will place the Y origin of the tooltip near where the mouse is. OBJECT_Y will place the tooltip next to the UIObject that is being moused over, if any. At first, it will try to place the tooltip on top of the UIObject. But if there is not enough screen space to fit the tooltip there, it will place it on the bottom of the UIObject.

Screen Tag
The Screen Tag for the XML file to be used for the tooltip

Xalignment
This parameter can be used to quickly position a tooltip relative to the full-screen space. The valid arguments are:

- ALIGN_NONE (default)
- ALIGN_CENTER
- ALIGN_LEFT
- ALIGN_RIGHT

Yalignment
This parameter can be used to quickly position a tooltip relative to the full-screen space. The valid arguments are:

- ALIGN_NONE (default)
- ALIGN_CENTER
- ALIGN_TOP
- ALIGN_BOTTOM

Xpadding
This is used along with XAlign to quickly position the tooltip. It controls how far in the tooltip should be pushed from the left or right border of the screen space. For example, setting XAlignment to ALIGN_RIGHT and XPadding to 10 will result in the tooltip appearing on the right side of the screen, with a space of 10 pixels between the right edge of the tooltip scene and the edge of the screen.

Ypadding
This is used along with YAlign to quickly position the tooltip. It controls how far in the tootip should be pushed from the top or bottom border of the screen space. For example, setting YAlignment to ALIGN_BOTTOM and YPadding to 10 will result in the tooltip appearing on the bottom of the screen with a space of 10 pixels between the bottom edge of the tooltip scene and the edge of the screen.

TextAlignment
How the text should be aligned in the tooltip. The options are:

- ALIGN_NONE (default),
- ALIGN_CENTER
- ALIGN_LEFT
- ALIGN_RIGHT
- ALIGN_TOP
- ALIGN_BOTTOM

UI Misc

Special Screens

Most XML files are loaded as default UI Scene, which implies no special behaviors. There are, however, several XML files that will get special handling in code and most conform to specific rules in order to even be loaded. All of these scenes are distinguished by their identifying name in the INI file or the screen tag given to them when being loaded via script. The actual XML file name doesn't matter.

Note that where a name ends with a * it means that only the text up to the * is required and that the name may be unique after that mark.

If a Screen Tag doesn't match to any of the above screens, then a normal UI Scene gets created with all the default behaviors associated with it.

Screens

The following are special GUI screens that are loaded differently than most XML files:

SCREEN_FADE
The full screen 'fade' effect. It gets loaded like a normal XML file at first. After loaded, the first UIIcon object found (The one highest up in the file) becomes the engine's 'Fading' icon. The rest of the contents of the file will behave normally. In our default SCREEN_FADE, we include just the full screen icon. But there may be other things one would want to include on the fade, such as custom images, etc.

SCREEN_QUICKCHAT
The NWN1 style dialog box. First it is loaded like a normal XML file, then the following ui objects are searched for in order for the engine to use them:

- npclistbox - Listbox of NPC spoken text
- npctext - Text field for NPC spoken text (Contained by npclistbox generally)
- replieslistbox - Listbox of Player Reply options
- skipdialogbutton - Button for skipping through the NPC spoken nodes
- speakername - Text field for containing the speaker's name
- portrait - UIPortrait object

All but the speakername and portrait fields are necessary or the window will not be loaded.

SCREEN_CUTSCENE
The fill screen cutscene view with the black bars on the top and bottom. This window is loaded like any other, then the following objects are searched for by the engine:

- topbar - UIFrame for the top black bar
- bottombar - UIFrame for the bottom black bar
- FULLSCREEN_IMAGE - UIIcon used for full screen images.
- toplistbox - Listbox to contain the text shown on top
- toplistboxtext - Textfield contained by toplistbox
- bottomlistbox - Listbox to contain the text shown on the bottom. Spoken by NPCs
- bottomlistboxtext - Text field contained by bottomlistbox
- replieslistbox - Listbox to contain the replies available to a player.
- skipdialogbutton - Fullscreen button for clicking through the dialog

Failure to locate any of the above objects will result in the screen not loading.

SCREEN_CONTEXTMENU
The rightclick menu system. Technically, this gets loaded like any other XML file. I'll make another post later about the syntax and format of the contextmenu.xml file.

SCREEN_MINIMAP
The in-game minimap GUI. I don't know a lot about how the Minimap works. If further documentation is requested for it, I can research it further later.

SCREEN_AREAMAP
The in-game area map that can be brought up. I don't know a lot about how the Area Map works. If further documentation is requested for it, I can research it further later.

SCREEN_MESSAGEBOX_SPLITSTACK*
Scene used for splitting stacks of items in inventory. The only gui element required is 'inputbox'.

SCREEN_MESSAGEBOX_SPLITSTACKSTORE*
Scene used for splitting stacks of items for stores. It's actually loaded identically to the SCREEN_MESSAGEBOX_SPLITSTACK and only requires the 'inputbox' object to exist. I'm not sure why it got made into a seperate entry.

SCREEN_MESSAGEBOX
This is used for the generic message box popups. They can be used effectively by scripts as well. The required elements for these are:

- messagetext = Text field that contains the message.
- okbutton = Button that will execute the OK callback
- cancelbutton = Button that will execute the Cancel callback
- messageboxlb = Listbox for containing the message text in case the text gets long.
- MSGBOX_BACKGROUND = Frame used for the background, this one is optional.

SCREEN_STRINGINPUT_MESSAGEBOX*
Message boxes that prompt for user string input. They are identical to normal Message boxes, except that they also require a 'inputbox' text field.

SCREEN_MESSAGE*
Chat boxes, pretty much. Note that currently the engine only supports loading the hard coded ones in ingamegui.ini. I hope to change this to be a lot more flexible down the line. The required elements are:

- messagelistbox - Listbox containing the scrollback text. If it is not found, the engine searches for messagelistbox2 instead. If neither is found, the screen is not loaded.
- inputbox - Text field used for user input.
- IMEREadingWindow - Used for IME support.
- IMEReadingWindowBG - Used for IME support.
- IMEComposeWindow - Used for IME support.
- IMEComposeWindow - Used for IME support.
- IMECandidateWindow - Used for IME support.
- IMECandidateWindowBG - Used for IME support.
- INPUT_CONTAINER - Used by the engine for easily hiding/unhiding the input related objects.

SCREEN_HOTBAR
SCREEN_HOTBAR_2
SCREEN_HOTBAR_V1
SCREEN_HOTBAR_V2
All the hotbars built into the game. There are no required or special UI elements needed to load these.

New UI Callback Parameters

In 1.06, callbacks can take 2 new 'variable' parameters in addition to **global:#** and **local:#**.

listboxrow:listboxname

This will insert the row # that is currently selected in the listbox indicated by listboxname

For example, if I had a callback that looked like:

```
OnLeftClick=UIObject_Misc_ExecuteServerScript("myguiscript",listboxrow:mylb)
```

On the same UIScene I have a listbox called 'mylb' where I have selected the 3rd row.

When the OnLeftClick callback above gets executed, the 1 parameter that gets sent to that script will be an integer with the value of 2. (The first row in the LB would send a value of 0).

listboxtext:listboxname[.textfieldname]

This will insert the text contained by that row. If the listbox rows are buttons or text fields, then the textfieldname parameter above is not necessary. If instead they are complex UI Panes, then the textfieldname will be used to specify which object within the pane contains the text we want.

For example, if I had a callback that looked like:

```
OnLeftClick=UIObject_Misc_ExecuteServerScript("myguiscript",listboxtext:mylb.mytext)
```

In the same UIScene, I have a Listbox named 'mylb'. The rows of that lb are complex UIPanes that contain a text field in them called 'mytext'.

When that callback executes, the engine will look at 'mylb', find the currently selected row, then look for the object named 'mytext' within that row and insert the text within that object as a parameter to the script on the server.

Now obviously the real power of these new variable parameters will be when coupled with the ability for scripts to populate a gui listbox. That functionality isn't in yet, but is something else I intend to have in for 1.06.

Multiple Callbacks Per Event

In 1.06, it will be possible to set up multiple GUI callbacks on a single UI event in XML. The syntax will be to use the event name and append a number to it. The engine will read in the callbacks until it fails to find the next sequential number.

For example, I could set up a UI Button to with the following callbacks:

```
OnUpdate=UIObject_OnUpdate_DisableIfLocalVarEquals(local:3,"true")
OnLeftClick=UIObject_Input_ScreenOpen("SCREEN_OPTIONS","false")
OnLeftClick0=UIObject_Input_SetLocalVarString(local:3,"true")
```

When I click that button, it would open the Options screen, then set the local var to "true". On its next update, it would become disabled because of the OnUpdate check.

Note that the callback without a number will be handled before the 0'th callback.
So OnLeftClick will be handled before OnLeftClick0.

Also note that the callback without the number is not required but is mostly just supported for backwards compatibility.

For example, it would be valid to have the following callbacks:

```
OnLeftClick0=
OnLeftClick1=
OnLeftClick2=
```

That will work fine, executing 0, then 1, then 2.

It is also valid to have:

```
OnLeftClick=
OnLeftClick0=
OnLeftClick1=
OnLeftClick2=
```

This will execute them in order.

It is not valid to start with any other number besides 0 however.

```
OnLeftClick1=
OnLeftClick2=
OnLeftClick3=
```

will result in no callbacks being loaded for that event for that UIObject.

Sometimes the engine assigns callbacks to UIObjects internally. The engine will only ever override the first callback, so if you are having a problem with the engine overriding your callback for some UI Object, you might need to insert a dummy callback first, before the callbacks you actually want to execute. This doesn't happen too often though, so I doubt this will be an issue for very many.

Note at this time, the only event that doesn't support multiple callbacks is OnTooltip, due to how the engine has to handle this event internally. I may or may not be able to fix OnTooltip by the time 1.06 comes out.

Console Commands

1.06 also added a couple console commands to help with GUI debugging

- **openuiscreen []**: TODO
- **closeuiscreen**: TODO
- **unloaduiscreen**: TODO

and:

- **guidebug [MOUSEINPUT] [MOUSEOVER] [CALLBACKS]**: This brings up an extra 'chat' window and will provide engine-level debugging info about mouse-overs, mouseinput, or callback execution.

UI Scripting

Functions

```
//This function will close a specific GUI panel on the client.
//The panel must be located within the [ScriptGUI] section of the ingamegui.ini
//in order to let this script close it.
void CloseGUIScreen(
       object oPlayer,
       string sScreenName );
```

```
// This function allows the script to display a GUI on the player's client.
// The first parameter is the object ID owned by the player you wish to
// display the GUI on.
// The second parameter is the name of the GUI screen to display. Note
// that only screens located in the [GuiScreen] section of ingamegui.ini
// will be accessible.
// The 3rd parameter indicates if the displayed GUI should be modal when
// it pops up.
// 4th parameter. This defines the resource that should be used for this screen if the
// screenName is not already found in the ingamegui.ini or pregamegui.ini. If left
// blank, then no gui will be loaded if the ScreenName doesn't already exist. If the
// sScreenName is *already* in use, then the 4th parameter will be ignored
void DisplayGuiScreen(
       object oPlayer,
       string sScreenName,
       int bModal,
       string sFileName="");
```

```
void SetGUIObjectDisabled(
       object oPlayer,
       string sScreenName,
       string sUIObjectName,
       int bDisabled );
```

```
// This function will set a GUI object as hidden or visible on a GUI panel on
// the client.
// The panel must be located within the [ScriptGUI] section of the ingamegui.ini
// in order to let this script function have any effect on it.
// Also, the panel must be in memory. Which means the panel should probably not have
// any idle expiration times set in the <UIScene> tag that would cause the panel to
// unload
void SetGUIObjectHidden(
       object oPlayer,
       string sScreenName,
       string sUIObjectName,
       int bHidden );
```

```
// If StrRef is -1, then sText is used instead
// Passing in -1 and "" will clear the text
void SetGUIObjectText(
       object oPlayer,
       string sScreenName,
       string sUIObjectName,
       int nStrRef,
       string sText );
```

```
// This script function sets the position of a progress bar on a client's GUI.
// The progress can be a percentage between 0.0 and 1.0 (empty and full).
// In order for this script function to work, the UIScene that contains the
// progress bar must have a scriptloadable=true attribute in it.
void SetGUIProgressBarPosition(
       object oPlayer,
       string sScreenName,
       string sUIObjectName,
       float fPosition );
// This script function sets the texture for a Icon, Button, or Frame.
```

```
// If the object is an icon, the texture is set as the icon's image.
// If the object is a frame, the texture is set as the frame's FILL texture
// If the object is a button that has a base frame, the texture is set
// as the BASE frame's FILL texture.
// If the object is a button that has no base frame, the texture is set
// as the UP state's FILL texture.
// Texture names should include the extention (*.tga for example).
// The UIScene that contains the UIObject must have a scriptloadable=true
// attribute in it.
void SetGUITexture(
       object oPlayer,
       string sScreenName,
       string sUIObjectName,
       string sTexture );
```

```
// This function sets the string value of a local GUI variable on the client of
// the indicated player
void SetLocalGUIVariable(
       object oPlayer,
       string sScreenName,
       int nVarIndex,
       string sVarValue );
```

```
// This script function will turn off a player's GUI, except for the FADE screen.
// Floating text will still be rendered.
// NOTE: that if the player hits ESC, their GUI will come back up.
// - oPlayer    object ID of the player to turn off the GUI for
// - bHide      if TRUE, turns off the player's GUI, of FALSE, shows the player's GUI.
void SetPlayerGUIHidden(
       object oPlayer,
       int bHidden );
```

```
//Convert a script integer to an 'object'
//Note that this doesn't guarentee that the 'object' is anything valid
object IntToObject( int nInt );
```

```
//Convert an 'object' to an integer data type
int ObjectToInt( object oObj );
```

```
// Convert oObject into a hexadecimal string.
string ObjectToString(object oObject);
```

```
//Convert a 'string' to an object data type.
//This does not guarentee that the 'object' is anything valid
object StringToObject( string sString );
```

```
//This script function displays a text input box popup on the client of the
//player passed in as the first parameter.
//
// oPC           - The player object of the player to show this message box to
// nMessageStrRef- The STRREF for the Message Box message.
// sMessage      - The text to display in the message box. Overrides anything
//               - indicated by the nMessageStrRef
// sOkCB         - The callback script to call if the user clicks OK, defaults
//               - to none. The script name MUST start with 'gui'
// sCancelCB     - The callback script to call if the user clicks Cancel, defaults
//               - to none. The script name MUST start with 'gui'
// bShowCancel   - If TRUE, Cancel Button will appear on the message box.
// sScreenName   - The GUI SCREEN NAME to use in place of the default message box.
//               - The default is SCREEN_STRINGINPUT_MESSAGEBOX
// nOkStrRef     - The STRREF to display in the OK button, defaults to OK
// sOkString     - The string to show in the OK button. Overrides anything that
//               - nOkStrRef indicates if it is not an empty string
// nCancelStrRef - The STRREF to dispaly in the Cancel button, defaults to Cancel.
```

```
// sCancelString - The string to display in the Cancel button. Overrides anything
//                          - that nCancelStrRef indicates if it is anything besides
empty string
// sDefaultString- The text that gets copied into the input area,
//                          - used as a default answer
void DisplayInputBox(
       object oPC,
       int nMessageStrRef,
       string sMessage,
       string sOkCB="",
       string sCancelCB="",
       int bShowCancel=FALSE,
       string sScreenName="",
       int nOkStrRef=0,
       string sOkString="",
       int nCancelStrRef=0,
       string sCancelString="",
       string sDefaultString="",
       string sUnusedString="" );
```

```
//This script function displays a message box popup on the client of the
//player passed in as the first parameter.
//
// oPC           - The player object of the player to show this message box to
// nMessageStrRef- The STRREF for the Message Box message.
// sMessage      - The text to display in the message box. Overrides anything
//                 indicated by the nMessageStrRef
// sOkCB         - The callback script to call if the user clicks OK, defaults
//                 to none. The script name MUST start with 'gui'
// sCancelCB     - The callback script to call if the user clicks Cancel, defaults
//                 to none. The script name MUST start with 'gui'
// bShowCancel   - If TRUE, Cancel Button will appear on the message box.
// sScreenName   - The GUI SCREEN NAME to use in place of the default message box.
//                 The default is SCREEN_MESSAGEBOX_DEFAULT
// nOkStrRef     - The STRREF to display in the OK button, defaults to OK
// sOkString     - The string to show in the OK button. Overrides anything that
//                 nOkStrRef indicates if it is not an empty string
// nCancelStrRef - The STRREF to dispaly in the Cancel button, defaults to Cancel.
// sCancelString - The string to display in the Cancel button. Overrides anything
//                 that nCancelStrRef indicates if it is anything besides empty string
void DisplayMessageBox(
       object oPC,
       int nMessageStrRef,
       string sMessage,
       string sOkCB="",
       string sCancelCB="",
       int bShowCancel=FALSE,
       string sScreenName="",
       int nOkStrRef=0,
       string sOkString="",
       int nCancelStrRef=0,
       string sCancelString="" );
```

UI Fonts

Fonts

UIFontFamilies

- Default
- NWN1_Dialog
- NWN2_Dialog
- Floating_Text_Default
- Floating_Text_Special
- International
- Title_Font
- Body_Font
- Special_Font
- Special_Font_2

Default

- The name of this font family CANNOT change
- This is also the font used by the chat window (US/EN)

Default	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_Main.ttf	12	false	false
UIFontBold	NWN2_MainB.ttf	12	false	false
UIFontItalic	NWN2_MainI.ttf	12	false	false
UIFontBoldItalic	NWN2_MainBI.ttf	12	false	false

NWN1_Dialog

- This is the font used by the NWN1 Dialog system and various other screens as well

NWN1_Dialog	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_Main.ttf	14	false	false
UIFontBold	NWN2_MainB.ttf	14	false	false
UIFontItalic	NWN2_MainI.ttf	14	false	false
UIFontBoldItalic	NWN2_MainBI.ttf	14	false	false

NWN2_Dialog

- This is the font used by the NWN2 Dialog screen

NWN2_Dialog	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_Main.ttf	19	false	false
UIFontBold	NWN2_MainB.ttf	19	false	false
UIFontItalic	NWN2_MainI.ttf	19	false	false
UIFontBoldItalic	NWN2_MainBI.ttf	19	false	false

Floating_Text_Default & Floating_Text_Special

- These next two font families are used by the floating text system.
- They can be accessed through the ftext_styles.2da

Floating_Text_Default	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_Main.ttf	14	true	true
UIFontBold	NWN2_MainB.ttf	14	true	true
UIFontItalic	NWN2_MainI.ttf	14	true	true
UIFontBoldItalic	NWN2_MainBI.ttf	14	true	true

Floating_Text_Special	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_Main.ttf	15	true	true
UIFontBold	NWN2_MainB.ttf	15	true	true
UIFontItalic	NWN2_MainI.ttf	15	true	true
UIFontBoldItalic	NWN2_MainBI.ttf	15	true	true

International

- The name of this font family CANNOT change
- This is the font used by the chat window (C/J/K and F/I/G)

International	font	pointsize	dropshadows	outline
UIFontNormal	ARIALUNI.TTF	12	false	false
UIFontBold	ARIALUNI.TTF	12	false	false
UIFontItalic	ARIALUNI.TTF	12	false	false
UIFontBoldItalic	ARIALUNI.TTF	12	false	false

Title_Font

- This font family is used for Buttons and Titles throughout the UI

Title_Font	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_TitleB.ttf	12	true	true
UIFontBold	NWN2_TitleB.ttf	14	true	true
UIFontItalic	NWN2_TitleB.ttf	12	true	false
UIFontBoldItalic	NWN2_TitleB.ttf	14	true	false

Body_Font

- This font family is used for numeric data for the UI

Body_Font	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_MainB.ttf	12	false	true
UIFontBold	NWN2_MainB.ttf	14	false	true
UIFontItalic	NWN2_MainB.ttf	12	true	false
UIFontBoldItalic	NWN2_MainB.ttf	14	true	false

Special_Font

- UIFontNormal = quick spell font
- UIFontItalic = mini map and others
- UIFontBoldItalic = large page headers (24 bold)

Special_Font	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_Main.ttf	10	false	true
UIFontBold	NWN2_MainB.ttf	16	false	true
UIFontItalic	NWN2_Main.ttf	11	false	false
UIFontBoldItalic	NWN2_TitleB.ttf	24	false	true

Special_Font_2

- UIFontNormal = used for concentration
- UIFontBold = target frames
- UIFontItalic = UNUSED
- UIFontBoldItalic = UNUSED

Special_Font_2	font	pointsize	dropshadows	outline
UIFontNormal	NWN2_TitleB.ttf	11	true	false
UIFontBold	NWN2_Main.ttf	12	false	true
UIFontItalic	NWN2_Main.ttf	11	false	false
UIFontBoldItalic	NWN2_TitleB.ttf	24	false	true

UI Styles

Styles

TODO: convert these to tables and remove extraneous info …
TODO: provide image library

GENERIC BUTTONS

```
    <!-- round radio button -->
    <UIButton name="ROUND_RADIO_BUTTON" buttontype=radio width=20 height=20
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIFrame state=up        fill="b_radio_off.tga" />
      <UIFrame state=down      fill="b_radio_on.tga" />
      <UIFrame state=focused   fill="b_radio_off.tga" />
      <UIFrame state=hilited   fill="b_radio_off.tga" />
      <UIFrame state=hifocus   fill="b_radio_on.tga" />
      <UIFrame state=disabled  fill="b_radio_off.tga" />
    </UIButton>

    <!-- round checkbox button -->
    <UIButton name="ROUND_CHECKBOX_BUTTON" buttontype=check width=20 height=20
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIFrame state=up        fill="b_radio_off.tga" />
      <UIFrame state=down      fill="b_radio_on.tga" />
      <UIFrame state=focused   fill="b_radio_off.tga" />
      <UIFrame state=hilited   fill="b_radio_off.tga" />
      <UIFrame state=hifocus   fill="b_radio_on.tga" />
      <UIFrame state=disabled  fill="b_radio_off.tga" />
    </UIButton>

    <!-- square checkbox button -->
    <UIButton name="SQUARE_CHECKBOX_BUTTON" buttontype=check width=20 height=20
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIFrame state=up        fill="b_checkbox_off.tga" />
      <UIFrame state=down      fill="b_checkbox_on.tga" />
      <UIFrame state=focused   fill="b_checkbox_off.tga" />
      <UIFrame state=hilited   fill="b_checkbox_off.tga" />
      <UIFrame state=hifocus   fill="b_checkbox_on.tga" />
      <UIFrame state=disabled  fill="b_checkbox_off.tga" />
    </UIButton>

    <!-- square radio button -->
    <UIButton name="SQUARE_RADIO_BUTTON" buttontype=radio width=20 height=20
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIFrame state=up        fill="b_checkbox_off.tga" />
      <UIFrame state=down      fill="b_checkbox_on.tga" />
      <UIFrame state=focused   fill="b_checkbox_off.tga" />
      <UIFrame state=hilited   fill="b_checkbox_off.tga" />
      <UIFrame state=hifocus   fill="b_checkbox_on.tga" />
      <UIFrame state=disabled  fill="b_checkbox_off.tga" />
    </UIButton>

    <!-- small tab button -->
    <UIButton name="STYLE_SMALL_TAB" width=82 height=33 buttontype=radio
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Title_Font" style="1"/>
      <UIFrame state=up        fill="tab_sm_normal.tga" />
      <UIFrame state=down      fill="tab_sm_pressed.tga" />
      <UIFrame state=focused   fill="tab_sm_normal.tga" />
      <UIFrame state=hilited   fill="tab_sm_normal.tga" />
      <UIFrame state=hifocus   fill="tab_sm_pressed.tga" />
      <UIFrame state=disabled  fill="tab_sm_normal.tga" />
    </UIButton>

    <!-- big tab button -->
    <UIButton name="STYLE_BIG_TAB" width=122 height=33 buttontype=radio
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Title_Font" style="1"/>
```

```
      <UIFrame state=up        fill="tab_big_normal.tga" />
      <UIFrame state=down      fill="tab_big_pressed.tga" />
      <UIFrame state=focused   fill="tab_big_normal.tga" />
      <UIFrame state=hilited   fill="tab_big_normal.tga" />
      <UIFrame state=hifocus   fill="tab_big_pressed.tga" />
      <UIFrame state=disabled  fill="tab_big_normal.tga" />
    </UIButton>

    <!-- small bordered button -->
    <UIButton name="STYLE_SMALL_BUTTON" width=124 height=28 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText  align=center valign=middle fontfamily="Title_Font" style="1" />
      <UIFrame state=up        fill="b_sm_normal.tga" />
      <UIFrame state=down      fill="b_sm_pressed.tga" />
      <UIFrame state=focused   fill="b_sm_normal.tga" />
      <UIFrame state=hilited   fill="b_sm_hover.tga" />
      <UIFrame state=hifocus   fill="b_sm_hover.tga" />
      <UIFrame state=disabled  fill="b_sm_normal.tga" />
    </UIButton>

    <!-- larger bordered button -->
    <UIButton name="STYLE_LARGE_BUTTON" width=194 height=28 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Title_Font" style="1" />
      <UIFrame state=up        fill="b_lg_normal.tga" />
      <UIFrame state=down      fill="b_lg_hover_pressed.tga" />
      <UIFrame state=focused   fill="b_lg_normal.tga" />
      <UIFrame state=hilited   fill="b_lg_hover.tga" />
      <UIFrame state=hifocus   fill="b_lg_hover.tga" />
      <UIFrame state=disabled  fill="b_lg_normal.tga" />
    </UIButton>

    <!-- plus button -->
    <UIButton name="STYLE_PLUS_BUTTON" width=23 height=23 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIFrame state=up        fill="b_plus_normal.tga" />
      <UIFrame state=down      fill="b_plus_pressed.tga" />
      <UIFrame state=focused   fill="b_plus_hover.tga" />
      <UIFrame state=hilited   fill="b_plus_hover.tga" />
      <UIFrame state=hifocus   fill="b_plus_hover.tga" />
      <UIFrame state=disabled  fill="b_plus_normal.tga" />
    </UIButton>

    <!-- minus button -->
    <UIButton name="STYLE_MINUS_BUTTON" width=23 height=23 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIFrame state=up        fill="b_minus_normal.tga" />
      <UIFrame state=down      fill="b_minus_pressed.tga" />
      <UIFrame state=focused   fill="b_minus_hover.tga" />
      <UIFrame state=hilited   fill="b_minus_hover.tga" />
      <UIFrame state=hifocus   fill="b_minus_hover.tga" />
      <UIFrame state=disabled  fill="b_minus_normal.tga" />
    </UIButton>

    <!-- empty button -->
    <UIButton name="STYLE_EMPTY_BUTTON" width=40 height=40 >
      <UIText align=center valign=middle fontfamily="Title_Font" style=1 />
      <UIFrame state=base      fill="b_empty.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_empty.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="b_empty.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!--TextTreeButton-->
    <UIButton name="STYLE_TEXTTREE_BUTTON" width=40 height=40 >
      <UIText align=center valign=middle fontfamily="Default" style=1 />
```

```
      <UIFrame state=base      fill="b_empty.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_empty.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="b_empty.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- Buff Button -->
    <UIButton name="STYLE_BUFF_BUTTON" width=24 height=24 >
      <UIText align=center valign=middle fontfamily="Title_Font" style=1 />
      <UIFrame state=base      fill="b_empty.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_empty.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="b_empty.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- options selection button -->
    <UIButton name="OPTIONS_BUTTON" width=100 height=23 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="selection_fill2.tga" />
      <UIFrame state=focused   fill="selection_fill2.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="selection_fill2.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- options selection button -->
    <UIButton name="OPTIONS_BUTTON_2" width=100 height=23 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Default" style="1" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="selection_fill2.tga" />
      <UIFrame state=focused   fill="selection_fill2.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="selection_fill2.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- multiplayer game selection button -->
    <UIButton name="MP_GAME_BUTTON" width=100 height=23 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="International" style="1" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="selection_fill.tga" />
      <UIFrame state=focused   fill="selection_fill.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="selection_fill.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- Mode Bar Button -->
    <UIButton name="STYLE_MODEBAR_BUTTON" width="77" height="28" buttontype="radio"
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIText align="center" valign="middle" fontfamily="Title_Font" style="1" />
      <UIFrame state="up"         fill="b_mb_normal.tga" />
      <UIFrame state="down"       fill="b_mb_pressed.tga" />
      <UIFrame state="focused"    fill="b_mb_normal.tga" />
      <UIFrame state="hilited"    fill="b_mb_normal.tga" />
      <UIFrame state="hifocus"    fill="b_mb_pressed.tga" />
      <UIFrame state="disabled"   fill="b_mb_normal.tga" />
    </UIButton>
```

CHARACTER GENERATION AND LEVEL UP BUTTONS

```
    <!-- CharGen Tab Style-->
    <UIButton name="STYLE_CHARGEN_TAB" width="124" height="28" buttontype="radio"
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIText align="center" valign="middle" fontfamily="Title_Font" style="4" />
      <UIFrame state="up"        fill="b_ctab_normal.tga" />
      <UIFrame state="down"      fill="b_ctab_pressed.tga" />
      <UIFrame state="focused"   fill="b_ctab_normal.tga" />
      <UIFrame state="hilited"   fill="b_ctab_hover.tga" />
      <UIFrame state="hifocus"   fill="b_ctab_pressed.tga" />
      <UIFrame state="disabled"  fill="b_ctab_normal.tga" />
    </UIButton>

    <!-- CharGen Group Style -->
    <UIButton name="STYLE_GROUP_BOX" width="230" height="40" >
      <UIText align="left" valign="top" fontfamily="Title_Font" style="3" />
      <UIFrame state="up"         fill="cont_cust_hh.tga" />
      <UIFrame state="down"       fill="cont_cust_hh.tga" />
      <UIFrame state="focused"    fill="cont_cust_hh.tga" />
      <UIFrame state="hilited"    fill="cont_cust_hh.tga" />
      <UIFrame state="hifocus"    fill="cont_cust_hh.tga" />
      <UIFrame state="disabled"   fill="cont_cust_hh.tga" />
    </UIButton>

    <!-- CharGen: Finalize Screen Listbox button -->
    <UIButton name="STYLE_CHARGEN_LIST_BUTTON" width="234" height="40"
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIText align="center" valign="middle" fontfamily="Title_Font" style="3" />
      <UIFrame state="up"        fill="b_g_lg03_normal.tga" />
      <UIFrame state="down"      fill="b_g_lg03_pressed.tga" />
      <UIFrame state="focused"   fill="b_g_lg03_hover.tga" />
      <UIFrame state="hilited"   fill="b_g_lg03_hover.tga" />
      <UIFrame state="hifocus"   fill="b_g_lg03_pressed.tga" />
      <UIFrame state="disabled"  fill="b_g_lg03_disabled.tga" />
    </UIButton>

    <!-- Skill Image Button for CharGen -->
    <UIButton name="STYLE_SKILL_ICON" x=0 y=0 width=40 height=40 prototype=true
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIFrame state=base      fill="skill_points_container.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_overlay.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="b_overlay.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- Skill Text Button for CharGen -->
    <UIButton name="STYLE_SKILL_TEXT" width=120 height=40 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText name="SKILL_TEXTFIELD" indent=5 hangingindent=5 align=left valign=middle
fontfamily="Title_Font" style=3 />
      <UIFrame state=up        fill="b_g_lg01_normal.tga" />
      <UIFrame state=down      fill="b_g_lg01_pressed.tga" />
      <UIFrame state=focused   fill="b_g_lg01_pressed.tga" />
      <UIFrame state=hilited   fill="b_g_lg01_normal.tga" />
      <UIFrame state=hifocus   fill="b_g_lg01_pressed.tga" />
      <UIFrame state=disabled  fill="b_g_lg01_normal.tga" />
    </UIButton>
```

FRONT END BUTTONS

```
    <!-- Front End Button used on Main Menu, Title 14pt. w/ Goldish Text  -->
    <UIButton name="STYLE_MENU_BUTTON" width=194 height=43 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
```

```
      <UIText align=center valign=middle fontfamily="Title_Font" style="2" color="DBE7F2"
/>
      <UIFrame state=up        fill="b_main_normal.tga" />
      <UIFrame state=down      fill="b_main_hover_pressed.tga" />
      <UIFrame state=focused   fill="b_main_normal.tga" />
      <UIFrame state=hilited   fill="b_main_hover.tga" />
      <UIFrame state=hifocus   fill="b_main_hover_pressed.tga" />
      <UIFrame state=disabled  fill="b_main_normal.tga" />
    </UIButton>

    <!-- Front End Button used on Main Menu, Title 14pt. w/ Goldish Text  -->
    <UIButton name="STYLE_MENU_BUTTON_SML" width=164 height=35 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align="center" valign="middle" fontfamily="Title_Font" style="2"
color="DBE7F2" />
      <UIFrame state="up"    fill="b_main_sm_normal.tga" />
      <UIFrame state="down"    fill="b_main_sm_hover_pressed.tga" />
      <UIFrame state="focused"  fill="b_main_sm_normal.tga" />
      <UIFrame state="hilited"  fill="b_main_sm_hover.tga" />
      <UIFrame state="hifocus"  fill="b_main_sm_hover_pressed.tga" />
      <UIFrame state="disabled"  fill="b_main_sm_normal.tga" />
    </UIButton>

    <!-- Listbox Button used on Module and Campaign Select Screen -->
    <UIButton name="STYLE_LIST_BUTTON" width=570 height=40 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Title_Font" style="2"
multiline=false />
      <UIFrame state=up        fill="b_g_lg04_normal.tga" />
      <UIFrame state=down      fill="b_g_lg04_pressed.tga" />
      <UIFrame state=focused   fill="b_g_lg04_hover.tga" />
      <UIFrame state=hilited   fill="b_g_lg04_hover.tga" />
      <UIFrame state=hifocus   fill="b_g_lg04_pressed.tga" />
      <UIFrame state=disabled  fill="b_g_lg04_normal.tga" />
    </UIButton>

    <!-- Multiplayer Browser Tab 102 x 27 -->
    <UIButton name="MP_BROWSER_TAB_102" width=102 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="column_tab_102_normal.tga" />
      <UIFrame state=down      fill="column_tab_102_pressed.tga" />
      <UIFrame state=focused   fill="column_tab_102_normal.tga" />
      <UIFrame state=hilited   fill="column_tab_102_normal.tga" />
      <UIFrame state=hifocus   fill="column_tab_102_pressed.tga" />
      <UIFrame state=disabled  fill="column_tab_102_normal.tga" />
    </UIButton>

    <!-- Multiplayer Browser Title Tab 165x28 -->
    <UIButton name="MP_BROWSER_TAB_TITLE" width=165 height=28 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="b_mpt_normal.tga" />
      <UIFrame state=down      fill="b_mpt_pressed.tga" />
      <UIFrame state=focused   fill="b_mpt_normal.tga" />
      <UIFrame state=hilited   fill="b_mpt_normal.tga" />
      <UIFrame state=hifocus   fill="b_mpt_pressed.tga" />
      <UIFrame state=disabled  fill="b_mpt_normal.tga" />
    </UIButton>

    <!-- Multiplayer Browser Tab 202 x 27 -->
    <UIButton name="MP_BROWSER_TAB_202" width=202 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="column_tab_202_normal.tga" />
      <UIFrame state=down      fill="column_tab_202_pressed.tga" />
      <UIFrame state=focused   fill="column_tab_202_normal.tga" />
      <UIFrame state=hilited   fill="column_tab_202_normal.tga" />
      <UIFrame state=hifocus   fill="column_tab_202_pressed.tga" />
      <UIFrame state=disabled  fill="column_tab_202_normal.tga" />
```

```
    </UIButton>

    <!-- Multiplayer Browser Tab 232 x 27 -->
    <UIButton name="MP_BROWSER_TAB_232" width=232 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="column_tab_232_normal.tga" />
      <UIFrame state=down      fill="column_tab_232_pressed.tga" />
      <UIFrame state=focused   fill="column_tab_232_normal.tga" />
      <UIFrame state=hilited   fill="column_tab_232_normal.tga" />
      <UIFrame state=hifocus   fill="column_tab_232_pressed.tga" />
      <UIFrame state=disabled  fill="column_tab_232_normal.tga" />
    </UIButton>

    <!-- Multiplayer Browser Tab 250 x 27 -->
    <UIButton name="MP_BROWSER_TAB_250" width=250 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="column_tab_250_normal.tga" />
      <UIFrame state=down      fill="column_tab_250_pressed.tga" />
      <UIFrame state=focused   fill="column_tab_250_normal.tga" />
      <UIFrame state=hilited   fill="column_tab_250_normal.tga" />
      <UIFrame state=hifocus   fill="column_tab_250_pressed.tga" />
      <UIFrame state=disabled  fill="column_tab_250_normal.tga" />
    </UIButton>

    <!-- Multiplayer Browser Tab 269 x 27 -->
    <UIButton name="MP_BROWSER_TAB_269" width=269 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="column_tab_269_normal.tga" />
      <UIFrame state=down      fill="column_tab_269_pressed.tga" />
      <UIFrame state=focused   fill="column_tab_269_normal.tga" />
      <UIFrame state=hilited   fill="column_tab_269_normal.tga" />
      <UIFrame state=hifocus   fill="column_tab_269_pressed.tga" />
      <UIFrame state=disabled  fill="column_tab_269_normal.tga" />
    </UIButton>

    <!-- Multiplayer Browser Tab 302 x 27 -->
    <UIButton name="MP_BROWSER_TAB_302" width=302 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="column_tab_302_normal.tga" />
      <UIFrame state=down      fill="column_tab_302_pressed.tga" />
      <UIFrame state=focused   fill="column_tab_302_normal.tga" />
      <UIFrame state=hilited   fill="column_tab_302_normal.tga" />
      <UIFrame state=hifocus   fill="column_tab_302_pressed.tga" />
      <UIFrame state=disabled  fill="column_tab_302_normal.tga" />
    </UIButton>

    <!-- Multiplayer Browser Tab 383 x 27 -->
    <UIButton name="MP_BROWSER_TAB_383" width=383 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Body_Font" style="1"/>
      <UIFrame state=up        fill="column_tab_383_normal.tga" />
      <UIFrame state=down      fill="column_tab_383_pressed.tga" />
      <UIFrame state=focused   fill="column_tab_383_normal.tga" />
      <UIFrame state=hilited   fill="column_tab_383_normal.tga" />
      <UIFrame state=hifocus   fill="column_tab_383_pressed.tga" />
      <UIFrame state=disabled  fill="column_tab_383_normal.tga" />
    </UIButton>

    <!-- Medium Listbox Button used on Load GameScreen -->
    <UIButton name="STYLE_LIST_BUTTON_MED" width=393 height=40 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Title_Font" style="2" />
      <UIFrame state=up        fill="b_g_lg05_normal.tga" />
      <UIFrame state=down      fill="b_g_lg05_pressed.tga" />
      <UIFrame state=focused   fill="b_g_lg05_hover.tga" />
      <UIFrame state=hilited   fill="b_g_lg05_hover.tga" />
```

```
      <UIFrame state=hifocus   fill="b_g_lg05_pressed.tga" />
      <UIFrame state=disabled  fill="b_g_lg05_disabled.tga" />
    </UIButton>

    <!-- Small Listbox Button used on Load GameScreen -->
    <UIButton name="STYLE_LIST_BUTTON_SML" width=173 height=40 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=center valign=middle fontfamily="Title_Font" style="2" />
      <UIFrame state=up        fill="b_g_sm02_normal.tga" />
      <UIFrame state=down      fill="b_g_sm02_pressed.tga" />
      <UIFrame state=focused   fill="b_g_sm02_hover.tga" />
      <UIFrame state=hilited   fill="b_g_sm02_hover.tga" />
      <UIFrame state=hifocus   fill="b_g_sm02_pressed.tga" />
      <UIFrame state=disabled  fill="b_g_sm02_disabled.tga" />
    </UIButton>
```

IN GAME BUTTONS

```
    <!-- Close Button -->
    <UIButton name="STYLE_CLOSE_BUTTON" width=27 height=27 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIFrame state=up        fill="b_close_normal.tga" />
      <UIFrame state=down      fill="b_close_hover_pressed.tga" />
      <UIFrame state=focused   fill="b_close_normal.tga" />
      <UIFrame state=hilited   fill="b_close_hover.tga" />
      <UIFrame state=hifocus   fill="b_close_hover.tga" />
      <UIFrame state=disabled  fill="b_close_normal.tga" />
    </UIButton>

    <!-- Player Menu Toggle Button -->
    <UIButton name="PM_TOGGLE_BUTTON" width=155 height=24 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button"
OnToolTip=UIObject_Tooltip_DisplayObject(OBJECT_X,OBJECT_Y,SCREEN_TOOLTIP_2,ALIGN_NONE,AL
IGN_NONE,0,0,ALIGN_LEFT) >
      <UIText  align=left valign=middle fontfamily="Body_Font" style="1" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="menu_highlight.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="menu_highlight.tga" />
      <UIFrame state=hifocus   fill="b_empty.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <UIButton name="CHAT_MODE_BUTTON" height=18 width=18 buttontype=radio
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIFrame state=base    fill="b_checkbox_off.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_overlay_active.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_overlay.tga" />
      <UIFrame state=hifocus   fill="b_overlay.tga" />
      <UIFrame state=disabled fill="b_emptyframe.tga" />
    </UIButton>

    <!-- equipment slot button -->
    <UIButton name="STYLE_EQUIP_BUTTON" width=36 height=36 draggable=true
handleactiontarget=true MouseDragSFX="gui_drag" MouseDropSFX="gui_drop"
    OnMouseDropFailed=UIButton_OnDropFailed_DropInventoryItem()

OnToolTip=UIObject_Tooltip_DisplayObject(OBJECT_X,OBJECT_Y,SCREEN_TOOLTIP_2,ALIGN_NONE,AL
IGN_NONE,0,0,ALIGN_LEFT)
    OnRadialRequest=UIObject_OnRadial_DisplayInventoryRadial() >
      <UIText align=right valign=bottom fontfamily="Special_Font" style="1"/>
      <UIFrame state=base      fill="b_empty.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_overlay.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_overlay.tga" />
```

```
      <UIFrame state=hifocus   fill="b_overlay.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- Context Menu Node-->
    <UIRadialNode name="BASE_RADIAL_NODE" height=32 width=32 x=0 y=0
    OnLeftClickExpanded=UIRadialNode_Mouse_CloseMenu()
    OnLeftClickCollapsed=UIRadialNode_Mouse_CloseMenu()>
      <UIFrame state=base      fill="b_empty.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_overlay.tga" />
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_overlay.tga" />
      <UIFrame state=hifocus   fill="b_overlay.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIRadialNode>

      <UIButton name="BEHAVIOR_COLLAPSABLE_HEADER" width=400 height=22  >
        <UIText align=center valign=middle fontfamily="Title_Font" style=1 />
        <UIFrame state=up          fill="b_feat_hover_pressed.tga" />
        <UIFrame state=down        fill="b_feat_hover_pressed.tga" />
        <UIFrame state=hilited     fill="b_feat_hover_pressed.tga" />
        <UIFrame state=focused     fill="b_feat_hover_pressed.tga" />
        <UIFrame state=hifocus     fill="b_feat_hover_pressed.tga" />
        <UIFrame state=disabled    fill="b_feat_hover_pressed.tga" />
        <UIFrame state=header      fill="b_feat_normal.tga" />
        <UIFrame state=hiheader    fill="b_feat_normal.tga" />
        <UIFrame state=downheader  fill="b_feat_normal.tga" />
      </UIButton>

      <UIButton name="BEHAVIOR_STATE_BUTTON" disabledcolor=FFFFFF height=20 width=20
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
        <UIFrame state=up        fill="b_radio_off.tga" />
        <UIFrame state=down      fill="b_radio_off.tga" />
        <UIFrame state=focused   fill="b_radio_off.tga" />
        <UIFrame state=hilited   fill="b_radio_off.tga" />
        <UIFrame state=hifocus   fill="b_radio_off.tga" />
        <UIFrame state=disabled  fill="b_radio_on.tga" />
      </UIButton>

      <UIButton name="BEHAVIOR_TEXT_BUTTON"  width=280 height=29
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
        <UIText name="BEHAVIOR_TEXT_BUTTON_TEXT" indent=15 align=left valign=middle
fontfamily="Title_Font" style=1 />
        <UIFrame state=up        fill="stats_container4.tga" />
        <UIFrame state=down      fill="stats_container4.tga" />
        <UIFrame state=focused   fill="stats_container4.tga" />
        <UIFrame state=hilited   fill="stats_container4.tga" />
        <UIFrame state=hifocus   fill="stats_container4.tga" />
        <UIFrame state=disabled  fill="stats_container4.tga" />
      </UIButton>

      <UIButton name="BEHAVIOR_ALL_BUTTON" width=60 height=28 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
        <UIText  align=center valign=middle fontfamily="Title_Font" style="1" />
        <UIFrame state=up        fill="b_sm_normal.tga" />
        <UIFrame state=down      fill="b_sm_pressed.tga" />
        <UIFrame state=focused   fill="b_sm_normal.tga" />
        <UIFrame state=hilited   fill="b_sm_hover.tga" />
        <UIFrame state=hifocus   fill="b_sm_hover.tga" />
        <UIFrame state=disabled  fill="b_sm_normal.tga" />
      </UIButton>

    <!-- Image Button for Loot Screen -->
    <UIButton name="STYLE_LOOT_ICON" x=0 y=0 width=40 height=40 prototype=true
MouseDownSFX="gui_m_down" MouseUpSFX="gui_button" >
      <UIText  align=right valign=bottom fontfamily="Special_Font" style="1" />
      <UIFrame state=base    fill="b_empty.tga" />
      <UIFrame state=up        fill="b_empty.tga" />
      <UIFrame state=down      fill="b_overlay.tga" />
```

```
      <UIFrame state=focused   fill="b_empty.tga" />
      <UIFrame state=hilited   fill="b_empty.tga" />
      <UIFrame state=hifocus   fill="b_overlay.tga" />
      <UIFrame state=disabled  fill="b_empty.tga" />
    </UIButton>

    <!-- Text Button for Loot Screen -->
    <UIButton name="STYLE_LOOT_TEXT" width=173 height=40 MouseDownSFX="gui_m_down"
MouseUpSFX="gui_button" >
      <UIText align=left valign=middle indent=10 hangingindent=10 fontfamily="Title_Font"
style=1 />
      <UIFrame state=up        fill="b_g_sm02_normal.tga" />
      <UIFrame state=down      fill="b_g_sm02_pressed.tga" />
      <UIFrame state=focused   fill="b_g_sm02_hover.tga" />
      <UIFrame state=hilited   fill="b_g_sm02_hover.tga" />
      <UIFrame state=hifocus   fill="b_g_sm02_pressed.tga" />
      <UIFrame state=disabled  fill="b_g_sm02_disabled.tga" />
    </UIButton>
```

SCROLLBARS

```
    <!-- standard scrollbar without a gutter/background -->
    <UIScrollBar name="STYLE_SB_THIN" width=15 height=PARENT_HEIGHT >
      <UIButton name=up x=0 y=0 width=15 height=22 >
        <UIFrame state=up        fill="sb_u_normal.tga" />
        <UIFrame state=down      fill="sb_u_hover_pressed.tga" />
        <UIFrame state=focused   fill="sb_u_hover.tga" />
        <UIFrame state=hilited   fill="sb_u_hover.tga" />
        <UIFrame state=hifocus   fill="sb_u_hover_pressed.tga" />
        <UIFrame state=disabled  fill="sb_u_disabled.tga" />
      </UIButton>
      <UIButton name=down x=0 width=15 height=22 >
        <UIFrame state=up        fill="sb_d_normal.tga" />
        <UIFrame state=down      fill="sb_d_hover_pressed.tga" />
        <UIFrame state=focused   fill="sb_d_hover.tga" />
        <UIFrame state=hilited   fill="sb_d_hover.tga" />
        <UIFrame state=hifocus   fill="sb_d_hover_pressed.tga" />
        <UIFrame state=disabled  fill="sb_d_disabled.tga" />
      </UIButton>
      <UIButton name=slider x=0 width=15 height=18 >
        <UIFrame state=up        fill="sb_handle.tga" />
        <UIFrame state=down      fill="sb_handle.tga" />
        <UIFrame state=focused   fill="sb_handle.tga" />
        <UIFrame state=hilited   fill="sb_handle.tga" />
        <UIFrame state=hifocus   fill="sb_handle.tga" />
        <UIFrame state=disabled  fill="sb_handle.tga" />
      </UIButton>
      <UIButton name=back x=0 width=15 height=PARENT_HEIGHT dontrendermousegrab=true >
        <UIFrame state=up        fill="sb_gutter.tga" />
        <UIFrame state=down      fill="sb_gutter.tga" />
        <UIFrame state=focused   fill="sb_gutter.tga" />
        <UIFrame state=hilited   fill="sb_gutter.tga" />
        <UIFrame state=hifocus   fill="sb_gutter.tga" />
        <UIFrame state=disabled  fill="sb_gutter.tga" />
      </UIButton>
    </UIScrollBar>

    <UIScrollBar name="STYLE_SB_ULTRA_THIN" width=8 height=PARENT_HEIGHT >
      <UIButton name=up x=0 y=0 width=8 height=8 >
        <UIFrame state=up        fill="sb_u_normal.tga" />
        <UIFrame state=down      fill="sb_u_hover_pressed.tga" />
        <UIFrame state=focused   fill="sb_u_hover.tga" />
        <UIFrame state=hilited   fill="sb_u_hover.tga" />
        <UIFrame state=hifocus   fill="sb_u_hover_pressed.tga" />
        <UIFrame state=disabled  fill="sb_u_disabled.tga" />
      </UIButton>
      <UIButton name=down x=0 width=8 height=8 >
```

```
        <UIFrame state=up        fill="sb_d_normal.tga" />
        <UIFrame state=down      fill="sb_d_hover_pressed.tga" />
        <UIFrame state=focused   fill="sb_d_hover.tga" />
        <UIFrame state=hilited   fill="sb_d_hover.tga" />
        <UIFrame state=hifocus   fill="sb_d_hover_pressed.tga" />
        <UIFrame state=disabled  fill="sb_d_disabled.tga" />
      </UIButton>
      <UIButton name=slider x=0 width=8 height=4 >
        <UIFrame state=up        fill="sb_handle.tga" />
        <UIFrame state=down      fill="sb_handle.tga" />
        <UIFrame state=focused   fill="sb_handle.tga" />
        <UIFrame state=hilited   fill="sb_handle.tga" />
        <UIFrame state=hifocus   fill="sb_handle.tga" />
        <UIFrame state=disabled  fill="sb_handle.tga" />
      </UIButton>
      <UIButton name=back x=0 width=8 height=PARENT_HEIGHT dontrendermousegrab=true >
        <UIFrame state=up        fill="sb_gutter.tga" />
        <UIFrame state=down      fill="sb_gutter.tga" />
        <UIFrame state=focused   fill="sb_gutter.tga" />
        <UIFrame state=hilited   fill="sb_gutter.tga" />
        <UIFrame state=hifocus   fill="sb_gutter.tga" />
        <UIFrame state=disabled  fill="sb_gutter.tga" />
      </UIButton>
    </UIScrollBar>

    <!-- horizontal slider bar used for settings and options -->
    <UIScrollBar name="STYLE_SLIDER" width=200 height=23 horizontal=true >
      <UIButton name=up x=0 y=0 width=22 height=23 >
        <UIFrame state=up        fill="b_larw_normal.tga" />
        <UIFrame state=down      fill="b_larw_pressed.tga" />
        <UIFrame state=focused   fill="b_larw_hover.tga" />
        <UIFrame state=hilited   fill="b_larw_hover.tga" />
        <UIFrame state=hifocus   fill="b_larw_hover_pressed.tga" />
        <UIFrame state=disabled  fill="b_larw_normal.tga" />
      </UIButton>
      <UIButton name=down x=0 y=290 width=22 height=23 >
        <UIFrame state=up        fill="b_rarw_normal.tga" />
        <UIFrame state=down      fill="b_rarw_pressed.tga" />
        <UIFrame state=focused   fill="b_rarw_hover.tga" />
        <UIFrame state=hilited   fill="b_rarw_hover.tga" />
        <UIFrame state=hifocus   fill="b_rarw_hover_pressed.tga" />
        <UIFrame state=disabled  fill="b_rarw_normal.tga" />
      </UIButton>
      <UIButton name=slider x=1 y=150 width=10 height=23>
        <UIFrame state=up        fill="slider_handle_normal.tga" />
        <UIFrame state=down      fill="slider_handle_pressed.tga" />
        <UIFrame state=focused   fill="slider_handle_pressed.tga" />
        <UIFrame state=hilited   fill="slider_handle_pressed.tga" />
        <UIFrame state=hifocus   fill="slider_handle_pressed.tga" />
        <UIFrame state=disabled  fill="slider_handle_normal.tga" />
      </UIButton>
      <UIButton name=back x=0 y=0 height=PARENT_HEIGHT width=PARENT_WIDTH
OnLeftClick=UIScrollBar_Input_JumpSlider()>
        <UIFrame state=up        fill="slider_background.tga" />
        <UIFrame state=down      fill="slider_background.tga" />
        <UIFrame state=focused   fill="slider_background.tga" />
        <UIFrame state=hilited   fill="slider_background.tga" />
        <UIFrame state=hifocus   fill="slider_background.tga" />
        <UIFrame state=disabled  fill="slider_background.tga" />
      </UIButton>
    </UIScrollBar>

    <!-- horizontal slider bar used for settings and options -->
    <UIScrollBar name="STYLE_OPTION_SLIDER" width=200 height=20 horizontal=true >
      <UIButton name=up x=0 y=0 height=20 width=20 >
        <UIFrame state=up        fill="b_minus_normal.tga" />
        <UIFrame state=down      fill="b_minus_hover_pressed.tga" />
        <UIFrame state=focused   fill="b_minus_normal.tga" />
        <UIFrame state=hilited   fill="b_minus_hover.tga" />
        <UIFrame state=hifocus   fill="b_minus_hover_pressed.tga" />
```

```
        <UIFrame state=disabled  fill="b_minus_normal.tga" />
      </UIButton>
      <UIButton name=down x=0 y=290 width=20 height=20 >
        <UIFrame state=up        fill="b_plus_normal.tga" />
        <UIFrame state=down      fill="b_plus_hover_pressed.tga" />
        <UIFrame state=focused   fill="b_plus_normal.tga" />
        <UIFrame state=hilited   fill="b_plus_hover.tga" />
        <UIFrame state=hifocus   fill="b_plus_hover_pressed.tga" />
        <UIFrame state=disabled  fill="b_plus_normal.tga" />
      </UIButton>
      <UIButton name=slider x=1 y=150 width=10 height=20 >
        <UIFrame state=up        fill="slider_handle_normal.tga" />
        <UIFrame state=down      fill="slider_handle_pressed.tga" />
        <UIFrame state=focused   fill="slider_handle_pressed.tga" />
        <UIFrame state=hilited   fill="slider_handle_pressed.tga" />
        <UIFrame state=hifocus   fill="slider_handle_pressed.tga" />
        <UIFrame state=disabled  fill="slider_handle_normal.tga" />
      </UIButton>
      <UIButton name=back x=0 y=0 width=PARENT_WIDTH height=PARENT_HEIGHT
OnLeftClick=UIScrollBar_Input_JumpSlider()>
        <UIFrame state=up        fill="slider_background.tga" />
        <UIFrame state=down      fill="slider_background.tga" />
        <UIFrame state=focused   fill="slider_background.tga" />
        <UIFrame state=hilited   fill="slider_background.tga" />
        <UIFrame state=hifocus   fill="slider_background.tga" />
        <UIFrame state=disabled  fill="slider_background.tga" />
      </UIButton>
    </UIScrollBar>
```