Continued from Part 1

Dialog 3 - Adding an OK Button

Change the "UI/Dialog.xml" file to be

```
<?xml version="1.0" encoding="utf-8" ?>
<Context>
  <Box Style="BGBlock_ClearTopBar" />

  <Grid Size="400,270" Anchor="C,C" Style="Grid9DetailFive140"
ConsumeMouse="1">
    <Label ID="Message" Anchor="C,C" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha"
String="TXT_KEY_TEST_DIALOG_MESSAGE"/>

    <GridButton ID="OK" Size="140,36" Anchor="C,B" Offset="0,50"
Style="BaseButton" ToolTip="TXT_KEY_TEST_DIALOG_BUTTON_OK_TT">
      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_OK" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>
  </Grid>
</Context>
```

and add the following to the end of "UI/Dialog.lua" file (after the Show() and Hide() stuff)

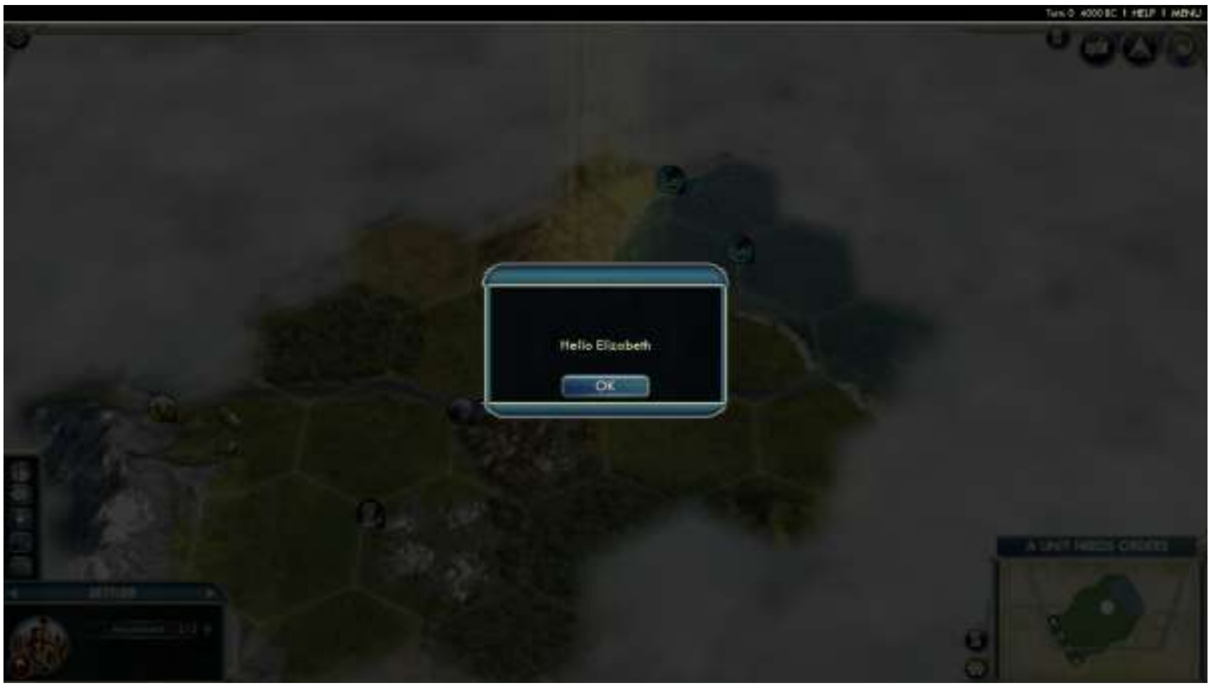
```
function OnOK()
  Hide()
end
Controls.OK:RegisterCallback(Mouse.eLClick, OnOK)

local leaderName =
GameInfo.Leaders[Players[Game.GetActivePlayer()]:GetLeaderType()].Desc
ription
Controls.Message:LocalizeAndSetText("TXT_KEY_TEST_DIALOG_HELLO_LEADER"
, leaderName)
```

and add the following into "XML/DialogText.xml"

```
    <Row Tag="TXT_KEY_TEST_DIALOG_BUTTON_OK">
      <Text>OK</Text>
    </Row>
    <Row Tag="TXT_KEY_TEST_DIALOG_BUTTON_OK_TT">
      <Text>Left click to dismiss the popup</Text>
    </Row>

    <Row Tag="TXT_KEY_TEST_DIALOG_HELLO_LEADER">
      <Text>Hello {1_LeaderName}</Text>
```

```
        </Row>
```

Save the changes, rebuild the mod, restart Civ 5, re-enable the mod and start a new game. Not only does the dialog box contain an OK button, the message now says "Hello {your leader name}" and also the rest of the screen has been "greyed out"



The message in the centre of the dialog was changed with the following Lua code

```
local leaderName =
GameInfo.Leaders[Players[Game.GetActivePlayer()]:GetLeaderType()].Desc
ription
Controls.Message:LocalizeAndSetText("TXT_KEY_TEST_DIALOG_HELLO_LEADER"
, leaderName)
```

All you really need to know about the first line is that "leaderName" ends up with the text key that is the name of the leader of the civilization that you are playing as - in my case TXT_KEY_LEADER_ELIZABETH

The second line is much more interesting. We can give names to our UI elements - via the ID attribute - and I have named the <Label> element that displayed "Hello World!" as "Message".

Once we have named the UI element, we can get to it from Lua via the "Controls" set - so in this case "Controls.Message" - and once we can get to the UI element, we can manipulate it. To distinguish between the XML UI element and the Lua UI element, we will refer to the latter as the control.

One of the more useful things we can change for a label control is the text that it displays, and we do this with the SetText() method. So we could write (in Lua)

```
Controls.Message:SetText("Hello World!")
```

which would of course be bad, as we should be using text keys (you can enter the above in the FireTuner Lua console and the text displayed in the centre of the dialog box will change back)

If we use

```
Controls.Message:SetText("TXT_KEY_TEST_DIALOG_MESSAGE")
```

which is good, we don't actually get "Hello World!" displayed but (somewhat surprisingly) "TXT_KEY_TEST_DIALOG_MESSAGE". This is because Lua doesn't perform the magic of translating from text keys to text automatically - we have to tell it to do it like this

```
Controls.Message:SetText(Locale.ConvertTextKey("TXT_KEY_TEST_DIALOG_ME
SSAGE"))
```

The pattern ":SetText(Locale.ConvertTextKey(something))" is so common that there's a shortcut form

```
Controls.Message:LocalizeAndSetText("TXT_KEY_TEST_DIALOG_MESSAGE")
```

Except we don't want the message to say "Hello World!" but "Hello Elizabeth".

We could mess about with trying to stick the text key for "Hello" in front of the leader name, but there is a better way. Text keys can contain "place holders", eg

```
<Row Tag="TXT_KEY_TEST_DIALOG_HELLO_LEADER">
  <Text>Hello {1_LeaderName}</Text>
</Row>
```

The {} bit is the place holder. A place holder is a number followed by an optional underscore (_) and a description of what the placeholder is used for. We could use just {1} but {1_LeaderName} tells anyone translating the text keys what to expect in the place holder - "Hello {1}" could be "Hello England" or "Hello Elizabeth", but "Hello {1_LeaderName}" is going to be "Hello Elizabeth" - and it may make a difference in other languages.

The number refers to the order that the extra bits of text are supplied in. Place holders don't have to occur in order in the text key, both of these are valid and can be generated from the same Lua

```
"You have {1_GoldTotal} in your treasury and spent {2_GoldTurn} this
turn"
"Your bills amounted to {2_GoldTurn} and you have {1_GoldTotal} left
under the bed"
```

Phew!

So what about the OK button?

```
    <GridButton ID="OK" Size="140,36" Anchor="C,B" Offset="0,50"
Style="BaseButton" ToolTip="TXT_KEY_TEST_DIALOG_BUTTON_OK_TT">
      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_OK" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>
```

You should know the meaning of the inner <Label> element by now, the Offset="0,-2" attribute just nudges the text 2 pixels up the screen (to allow for the fact that there is slightly more space above the letters in the font than below) and it defines the text that appears on the button - "OK" in this case.

The <GridButton> element defines the button - its size, anchor (relative to the grid it's in) and offset. We need the Offset="0,50" attribute so that it doesn't sit in the border but above it.

Hang on, you said "Offset="0,**-2**" attribute just nudges the text 2 pixels **up** the screen" and " Offset="0,**50**" attribute so that it doesn't sit in the border but **above** it", so is "negative y" up or down the screen?

Like most things in life, the answer is "it depends". In this case it depends on where the anchor is. Because the label has an anchor of "C,C" "negative y" is up the screen, whereas the gridbutton has an anchor of "C,B" so "negative y" is down the screen.

- "anything,T" or "anything,C" and positive y is down the screen
- "anything,B" and positive y is up the screen
- "L, anything" or "C, anything" and positive x is from left-to-right
- "R, anything" and positive x is from right-to-left

The ToolTip attribute defines the text key that will be displayed as the mouse hovers over the button and the Style attribute defines what the button will look like (like grids there are various pre-defined styles - BaseButton is the standard one for buttons in a dialog box).

Finally the ID attribute gives the element a name and allows us to access if from Lua (as "Controls.OK")

With the OK button defined in the context it will be drawn, but without some Lua code it won't actually do anything!

When the user clicks our OK button the core game engine detects the mouse click, works out which component(s) the mouse was over at the time and generates an "event". Our Lua code has to tell the core game engine that we are interested in this "event" and also what code should be executed. The code to be executed is technically known as a "call-back" (it gets called back from the core engine) and we "register" our interest in the event, so we need the Lua code line

```
Controls.OK:RegisterCallback(Mouse.eLClick, OnOK)
```

to tell the core game engine "when the user left clicks (Mouse.eLClick) on our OK button (Controls.OK) call our function OnOK"

And we also need an "OnOK" function

```
function OnOK()
  Hide()
end
```

which simply hides the dialog box.

Almost there!

All UI systems have two types of dialog boxes - those which you can only click the buttons/controls in them, and those where you can click other buttons/controls on the screen (technically known as "modal" and "modeless" dialogs respectively - but who really cares!) Civ 5 is no different, but the way it handles them is fairly unique. By default, all dialogs in Civ 5 are "modeless". If we want a "modal" dialog we just draw a box over the entire screen and set the ConsumeMouse="1" attribute on it. To indicate to the user that they can't click on anything else it is customary to "grey out" the rest of the screen. As this is such a common task, the Civ UI has a short-cut for doing this

```
<Box Style="BGBlock" />
```

There is a disadvantage to this, in that it also disables the Help and Menu buttons in the top-right hand corner of the screen. It's polite to leave the Menu button active (so the user can exit) so there is another short-cut that excludes the top panel (and hence leaves the Help and Menu buttons active)

```
<Box Style="BGBlock_ClearTopBar" />
```

Which explains what that's doing in the context and why the screen is greyed-out.

Important Note: Only one of "BGBlock" or "BGBlock_ClearTopBar" should be present and if one or the other is present it must be contained only within the <Context> element and be the first thing in the context.

Dialog 4 - Static and Dynamic Decoration

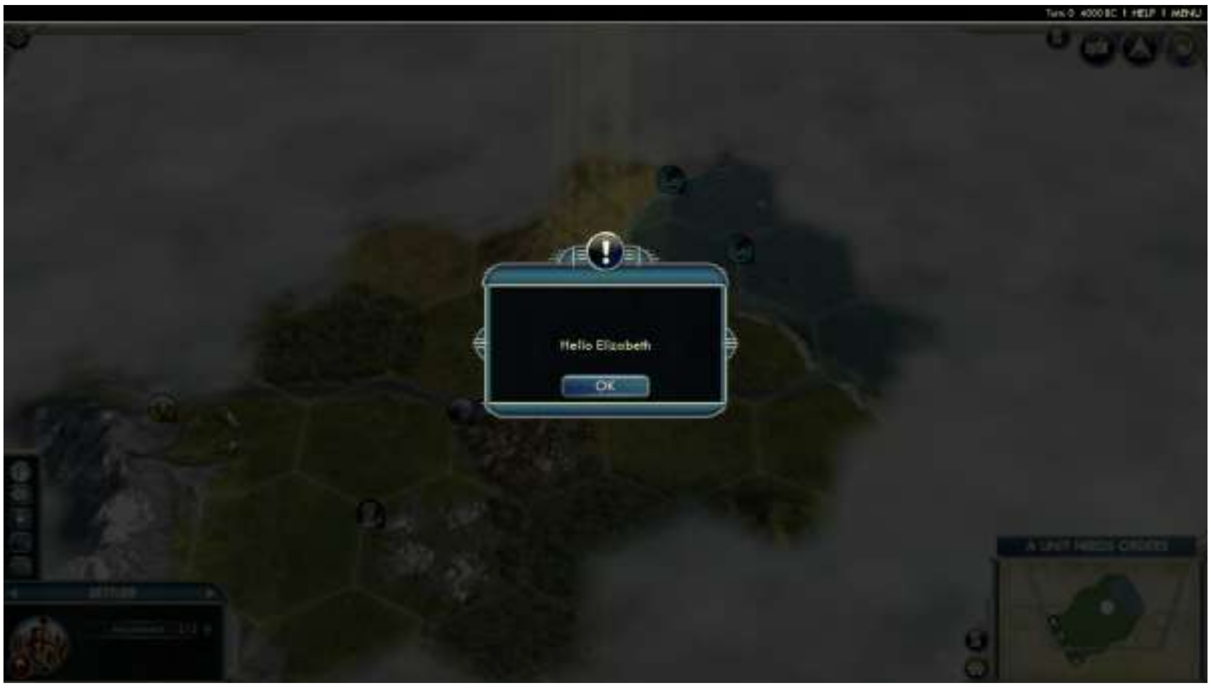
And now for a light interlude! Compared to the standard Civ 5 dialogs, our dialog box is a bit bland, so let's add some decoration!

Change the "UI/Dialog.xml" file to be (the new bits are in bold)

```
<?xml version="1.0" encoding="utf-8" ?>
<Context>
  <Box Style="BGBlock_ClearTopBar" />

  <Grid Size="400,270" Anchor="C,C" Style="Grid9DetailFive140"
ConsumeMouse="1">
    <Image Anchor="C,T" AnchorSide="I,O" Offset="0,-27" Size="256,64"
Texture="DecTop256x64.dds">
      <Image Anchor="C,C" Offset="0,-6" Size="80,80"
Texture="NotificationFrameBase.dds">
        <Image Anchor="C,C" Offset="0,0" Size="80,80"
Texture="NotificationGeneric.dds" />
      </Image>
    </Image>

    <Image Anchor="L,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Left.dds" />
```

```
    <Image Anchor="R,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Right.dds" />

    <Label ID="Message" Anchor="C,C" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha"
String="TXT_KEY_TEST_DIALOG_MESSAGE"/>

    <GridButton ID="OK" Size="140,36" Anchor="C,B" Offset="0,50"
Style="BaseButton" ToolTip="TXT_KEY_TEST_DIALOG_BUTTON_OK_TT">
      <ShowOnMouseOver>
        <AlphaAnim Anchor="L,C" AnchorSide="O,O" Size="16,32"
TextureOffset="0,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
        <Image     Anchor="L,C" AnchorSide="O,O" Size="8,16"
TextureOffset="0,0" Texture="buttonsides.dds" />

        <AlphaAnim Anchor="R,C" AnchorSide="O,O" Size="16,32"
TextureOffset="16,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
        <Image     Anchor="R,C" AnchorSide="O,O" Size="8,16"
TextureOffset="8,0"  Texture="buttonsides.dds" />
      </ShowOnMouseOver>

      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_OK" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>
  </Grid>
</Context>
```

Save the changes, rebuild the mod, restart Civ 5, re-enable the mod and start a new game.



Now that looks more like a standard Civ 5 dialog.

The top graphic is drawn by

```
<Image Anchor="C,T" AnchorSide="I,O" Offset="0,-27" Size="256,64"
Texture="DecTop256x64.dds">
  <Image Anchor="C,C" Offset="0,-6" Size="80,80"
Texture="NotificationFrameBase.dds">
    <Image ID="DialogTopIcon" Anchor="C,C" Offset="0,0" Size="80,80"
Texture="NotificationGeneric.dds" />
  </Image>
</Image>
```

The <Image> element places a graphic into the context as determined by the Texture attribute. Most of the attributes should be familiar by now; the AnchorSide attribute determines if the image is placed on the "inside" or "outside" of the grid (and further confuses the notion of "up", "down", "left" and "right"). For most decoration all we really need to do is find a standard dialog that has the decoration we want and copy it - understanding is not a requirement!

The side graphics are drawn by

```
<Image Anchor="L,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Left.dds" />
<Image Anchor="R,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Right.dds" />
```

which were copied from another dialog!

The button animation is provided by the following

```
<ShowOnMouseOver>
```

```
  <AlphaAnim Anchor="L,C" AnchorSide="O,O" Size="16,32"
TextureOffset="0,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
  <Image     Anchor="L,C" AnchorSide="O,O" Size="8,16"
TextureOffset="0,0" Texture="buttonsides.dds" />

  <AlphaAnim Anchor="R,C" AnchorSide="O,O" Size="16,32"
TextureOffset="16,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
  <Image     Anchor="R,C" AnchorSide="O,O" Size="8,16"
TextureOffset="8,0"  Texture="buttonsides.dds" />
</ShowOnMouseOver>
```

The <ShowOnMouseOver> tag tells the UI to hide all the contained elements until the mouse moves over the button, at which point they are revealed. When the mouse moves out of the button they are hidden again. (There is also a <HideOnMouseOver> tag which works the other way around.)

The elements which are revealed give positive feedback to the user that the mouse is over the button and if they click the mouse the button will be actioned. In this case the button gets "side handles" (the two Image elements) and the side handles "glow" (the two AlphaAnim elements) - for more information about the AlphaAnim elements see http://forums.civfanatics.com/showthread.php?t=460482

Dialog 4b - Changing the Icon

The icon at the top of the dialog frequently indicates what the dialog is about - a unit, building, city state, etc. Civ 5 does not store icons as separate graphics but in "icon atlases" - where up to 64 icons are held in one image and you have to "cut out" the required icon. Provided we can identify which icon in which atlas we want to use, Civ 5 provides a mechanism to perform the "cutting out" for us.

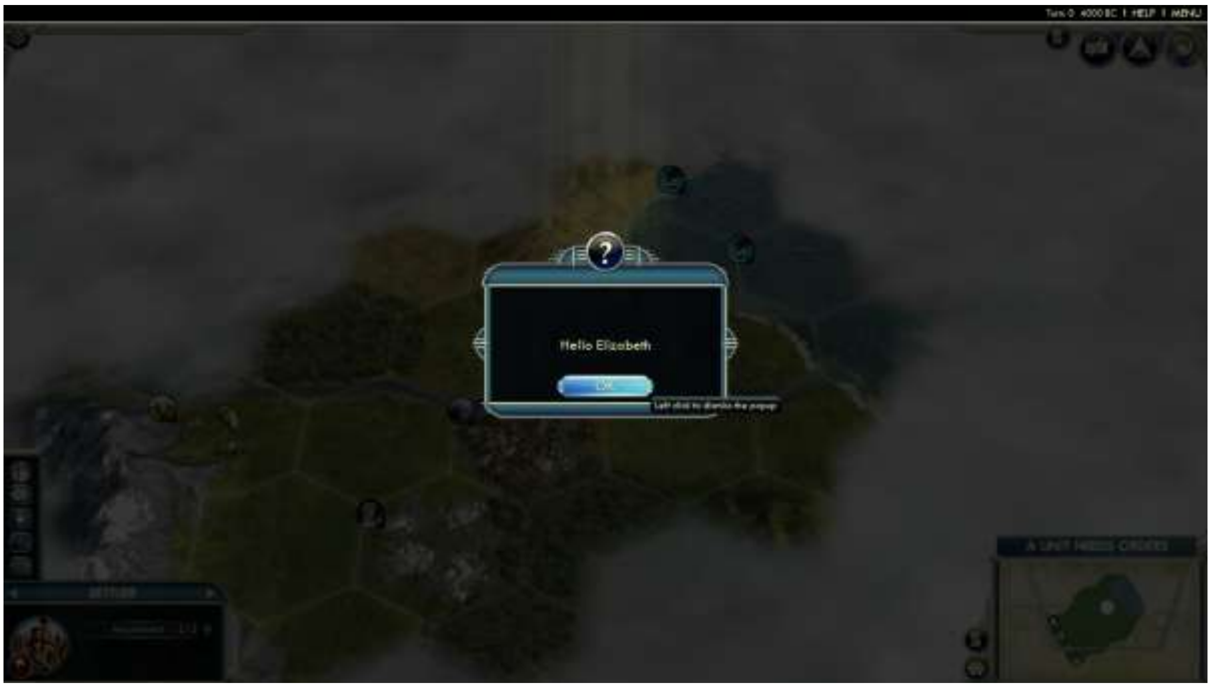
To the very top of the "UI/Dialog.lua" file add

```
include("IconSupport")
```

and to the very end add

```
IconHookup(23, 80, "CIV_COLOR_ATLAS", Controls.DialogTopIcon)
```

Save the changes, rebuild the mod, restart Civ 5, re-enable the mod and start a new game.



The top icon has changed from an exclamation mark (!) to a question mark (?)

The line

```
include("IconSupport")
```

tells Lua that we want to use the icon support code (the stuff that does the "cutting out") and

```
IconHookup(23, 80, "CIV_COLOR_ATLAS", Controls.DialogTopIcon)
```

tells the "cutting out" code (which is called "IconHookup") which icon (number 23) in which atlas (CIV_COLOR_ATLAS) and at what size (80) we want "cut out" and where we would like it put (into the DialogTopIcon control - which must be an Image element).

For more information on creating icons and the way Civ 5 handles icon atlases see http://forums.civfanatics.com/showthread.php?t=459392

We now have something that looks like a standard Civ 5 dialog box. We can change the text in it, either via a text key or from Lua, and the user can tell us that they have finished reading it (by pressing the OK button). It's a start, but usually you need the user to make a decision, and for that we need more buttons.

Dialog 5 - More Buttons

Change the "UI/Dialog.xml" file to be (the new bits are in bold)

```
<?xml version="1.0" encoding="utf-8" ?>
<Context>
  <Box Style="BGBlock_ClearTopBar" />
  <Grid Size="600,400" Anchor="C,C" Style="Grid9DetailFive140"
ConsumeMouse="1">
    <Image Anchor="C,T" AnchorSide="I,O" Offset="0,-27" Size="256,64"
Texture="DecTop256x64.dds">
      <Image Anchor="C,C" Offset="0,-6" Size="80,80"
Texture="NotificationFrameBase.dds">
        <Image ID="DialogTopIcon" Anchor="C,C" Offset="0,0"
Size="80,80" Texture="NotificationGeneric.dds" />
      </Image>
    </Image>

    <Image Anchor="L,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Left.dds" />
    <Image Anchor="R,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Right.dds" />

    <Label ID="Message" Anchor="C,C" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha"
String="TXT_KEY_TEST_DIALOG_MESSAGE"/>

    <GridButton ID="LangEN" Size="140,36" Anchor="C,B" Offset="-
150,100" Style="BaseButton">
      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_ENGLISH" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>

    <GridButton ID="LangFR" Size="140,36" Anchor="C,B" Offset="0,100"
Style="BaseButton">
      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_FRENCH" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>

    <GridButton ID="LangDE" Size="140,36" Anchor="C,B"
Offset="150,100" Style="BaseButton">
      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_GERMAN" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>

    <GridButton ID="OK" Size="140,36" Anchor="C,B" Offset="0,50"
Style="BaseButton" ToolTip="TXT_KEY_TEST_DIALOG_BUTTON_OK_TT">
      <ShowOnMouseOver>
        <AlphaAnim Anchor="L,C" AnchorSide="O,O" Size="16,32"
TextureOffset="0,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
        <Image     Anchor="L,C" AnchorSide="O,O" Size="8,16"
TextureOffset="0,0" Texture="buttonsides.dds" />
```

```
        <AlphaAnim Anchor="R,C" AnchorSide="O,O" Size="16,32"
TextureOffset="16,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
        <Image     Anchor="R,C" AnchorSide="O,O" Size="8,16"
TextureOffset="8,0"  Texture="buttonsides.dds" />
      </ShowOnMouseOver>

      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_OK" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>
  </Grid>
</Context>
```

and "UI/Dialog.lua" to be

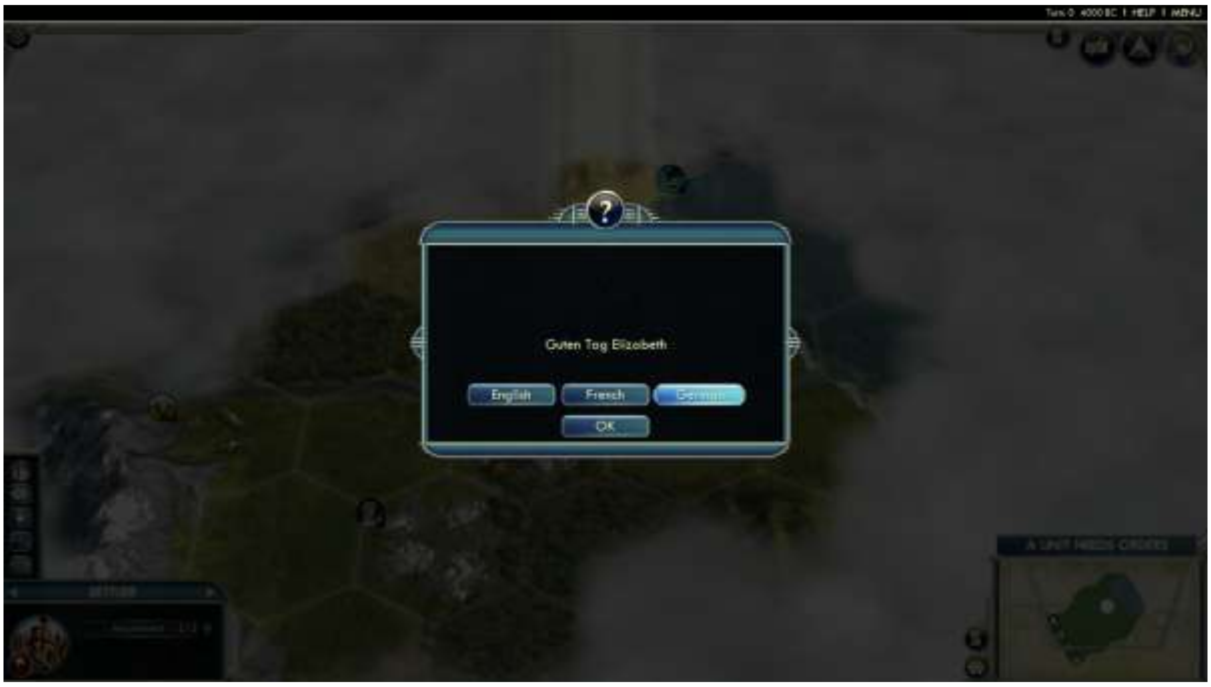
```
include("IconSupport")

function OnOK()
  ContextPtr:SetHide(true)
end
Controls.OK:RegisterCallback(Mouse.eLClick, OnOK)

function OnLangEN()
  Controls.Message:LocalizeAndSetText(
"TXT_KEY_TEST_DIALOG_HELLO_LEADER",
Players[Game.GetActivePlayer()]:GetName())
end
Controls.LangEN:RegisterCallback(Mouse.eLClick, OnLangEN)

function OnLangFR()
  Controls.Message:LocalizeAndSetText(
"TXT_KEY_TEST_DIALOG_BONJOUR_LEADER",
Players[Game.GetActivePlayer()]:GetName())
end
Controls.LangFR:RegisterCallback(Mouse.eLClick, OnLangFR)

function OnLangDE()
  Controls.Message:LocalizeAndSetText(
"TXT_KEY_TEST_DIALOG_GUTENTAG_LEADER",
Players[Game.GetActivePlayer()]:GetName())
end
Controls.LangDE:RegisterCallback(Mouse.eLClick, OnLangDE)

local leaderName =
GameInfo.Leaders[Players[Game.GetActivePlayer()]:GetLeaderType()].Desc
ription
```

```
    Controls.Message:LocalizeAndSetText("TXT_KEY_TEST_DIALOG_HELLO_LEADER"
    , leaderName)

    IconHookup(23, 80, "CIV_COLOR_ATLAS", Controls.DialogTopIcon)
```

and add the following to "XML/DialogText.xml"

```
    <Row Tag="TXT_KEY_TEST_DIALOG_BONJOUR_LEADER">
      <Text>Bonjour {1_LeaderName}</Text>
    </Row>
    <Row Tag="TXT_KEY_TEST_DIALOG_GUTENTAG_LEADER">
      <Text>Guten Tag {1_LeaderName}</Text>
    </Row>

    <Row Tag="TXT_KEY_TEST_DIALOG_BUTTON_ENGLISH">
      <Text>English</Text>
    </Row>
    <Row Tag="TXT_KEY_TEST_DIALOG_BUTTON_FRENCH">
      <Text>French</Text>
    </Row>
    <Row Tag="TXT_KEY_TEST_DIALOG_BUTTON_GERMAN">
      <Text>German</Text>
    </Row>
```

Save the changes, rebuild the mod, restart Civ 5, re-enable the mod and start a new game.



The first thing to notice is that our dialog is bigger. This was achieved by simply changing the Size attribute on the Grid element from "400,270" to "600,400", and as every other component is anchored relative to the grid they have moved appropriately.

Our dialog also has three extra buttons, and clicking on one of these changes the greeting displayed.

The code added to the "UI/Dialog.xml" file added these three buttons - they are basically the same as the OK button and should be becoming familiar. The only attribute of interest are the Offsets

```
<GridButton ID="LangEN" Size="140,36" Offset="-150,100" ...
<GridButton ID="LangFR" Size="140,36" Offset="0,100" ...
<GridButton ID="LangDE" Size="140,36" Offset="150,100" ...
```

Where did the "magic numbers" come from?

The French (middle) button is offset "0,100" or 100 pixels above the centre of the bottom edge of the grid - very similar to the OK button.

The English (left) button is offset "-150,100", the ",100" part puts it in the same horizontal alignment as the French button and the "-150" bit places it to the left - but why "150"? The buttons are 140 pixels wide and we want a 10 pixel gap between the buttons, for a total of 150 (Which is obvious now, but won't be in a weeks time when we come to change the button size to 120 and forget to change the offset to 130 and end up with a 30 pixel gap between the buttons!)

The German (right) button is the same as the English button, just to the right of the French button.

So what if we added a 4th button? We'd need to add another <GridButton> XML element and associated Lua callback function, but we'd also need to shuffle all the existing buttons around, and as four 140 pixel wide buttons won't fit, we'd also need to adjust all the sizes which would have a consequence for the offsets - which makes the likelihood of getting something wrong (or "missing a bit") very high.

There must be a better way? There is!

Dialog 5b - Easier More Buttons

In "UI/Dialog.xml", replace the three <GridButtons> (in bold above) with

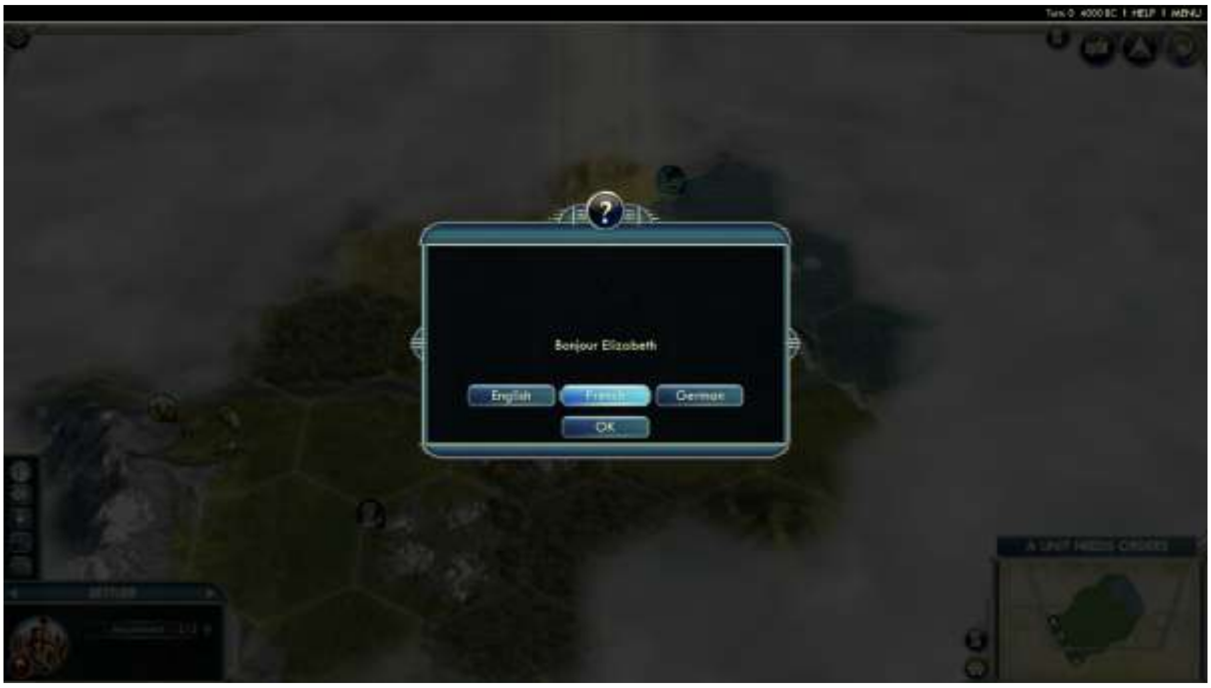
```
<Stack Anchor="C,B" Offset="0,100" StackGrowth="Right" Padding="10">
  <GridButton ID="LangEN" Size="140,36" Style="BaseButton">
    <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_ENGLISH" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
  </GridButton>

  <GridButton ID="LangFR" Size="140,36" Style="BaseButton">
    <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_FRENCH" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
  </GridButton>

  <GridButton ID="LangDE" Size="140,36" Style="BaseButton">
    <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_GERMAN" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
```

```
    </GridButton>
  </Stack>
```

(That is, **remove the Offset and Anchor attributes from each of the <GridButton> elements** and wrap them with the <Stack> element.)

No changes are needed to the "UI/Dialog.lua" file.

Save the changes, rebuild the mod, restart Civ 5, re-enable the mod and start a new game.



Nothing changed! That's good, the layout has stayed the same, but it's now a lot easier for us to add/remove language buttons.

The <Stack> element arranges whatever it contains in order either left-to-right or top-to-bottom. Our stack arranges the three language buttons left-to-right

```
  <Stack Anchor="C,B" Offset="0,100" StackGrowth="Right" Padding="10">
```

The stack is located within the dialog the same as the central (French) button was, ie, Anchor="C,B" Offset="0,100". As items are placed into it, it grows to the right (ie the items are aligned left-to-right) and the gap (padding) between each item is 10 pixels.

The attributes on the buttons are now concerned solely with how the buttons appear, and not how they are laid out. To add a new language button we simply add it to the stack, and all the other buttons will be shuffled about automatically. If we want to adjust the gap between buttons, we just change one value (padding) and not many values (the offset of each button in the previous code).

In general, if you have lots of things to place in a dialog side-by-side or one above another, "Think Stack". All of the initial game selection menus are stacks of buttons, the unit action buttons, promotion buttons, building buttons in the city view, etc, etc, etc are stacks of buttons - stacks are one of the most common UI elements used.

Dialog 6 - Odds and Ends

That's pretty much it for the basic UI elements, how they are used and what they do. It just remains to highlight a few other common elements.

Change the "UI/Dialog.xml" file to be (the new bits are in bold)

```
<?xml version="1.0" encoding="utf-8" ?>
<Context ID="Dialog6">
  <Box Style="BGBlock_ClearTopBar" />

  <Grid Size="600,400" Anchor="C,C" Style="Grid9DetailFive140"
ConsumeMouse="1">
    <Image Anchor="C,T" AnchorSide="I,O" Offset="0,-27" Size="256,64"
Texture="DecTop256x64.dds">
      <Button ID="IconButton" Anchor="C,C" Offset="0,-6" Size="80,80"
ToolTip="TXT_KEY_TEST_DIALOG_BUTTON_ICON_TT">
        <Image Anchor="C,C" Size="80,80"
Texture="NotificationFrameBase.dds">
          <Image Anchor="C,C" Size="80,80"
Texture="NotificationGeneric.dds" />
        </Image>
      </Button>
    </Image>

    <Container ID="SideHandles" Size="600,400" Anchor="C,C">
      <Image Anchor="L,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Left.dds" />
      <Image Anchor="R,C" AnchorSide="O,I" Offset="-17,0" Size="32,64"
Texture="Dec32x64Right.dds" />
    </Container>

    <Label ID="Message" Anchor="C,C" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha"
String="TXT_KEY_TEST_DIALOG_MESSAGE"/>

    <Box Anchor="C,B" Offset="0,100" Size="560,30" Color="White,255">
      <Box Anchor="C,C" Size="558,28" Color="Black,255">
        <Stack Anchor="C,C" StackGrowth="Right" Padding="10">
          <TextButton ID="LangEN" Size="120,24"
String="TXT_KEY_TEST_DIALOG_BUTTON_ENGLISH" Font="TwCenMT16"
ColorSet="Beige_Black_Alpha" FontStyle="Shadow"
MouseOverStyle="SoftShadow" />
          <TextButton ID="LangFR" Size="120,24"
String="TXT_KEY_TEST_DIALOG_BUTTON_FRENCH"  Font="TwCenMT16"
ColorSet="Beige_Black_Alpha" FontStyle="Shadow"
MouseOverStyle="SoftShadow" />
```

```
          <TextButton ID="LangDE" Size="120,24"
String="TXT_KEY_TEST_DIALOG_BUTTON_GERMAN"  Font="TwCenMT16"
ColorSet="Beige_Black_Alpha" FontStyle="Shadow"
MouseOverStyle="SoftShadow" />
        </Stack>
      </Box>
    </Box>

    <GridButton ID="OK" Size="140,36" Anchor="C,B" Offset="0,50"
Style="BaseButton" ToolTip="TXT_KEY_TEST_DIALOG_BUTTON_OK_TT">
      <ShowOnMouseOver>
        <AlphaAnim Anchor="L,C" AnchorSide="O,O" Size="16,32"
TextureOffset="0,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
        <Image     Anchor="L,C" AnchorSide="O,O" Size="8,16"
TextureOffset="0,0" Texture="buttonsides.dds" />

        <AlphaAnim Anchor="R,C" AnchorSide="O,O" Size="16,32"
TextureOffset="16,0" Texture="buttonsidesglow.dds" Cycle="Bounce"
Speed="1" AlphaStart=".99" AlphaEnd=".25"/>
        <Image     Anchor="R,C" AnchorSide="O,O" Size="8,16"
TextureOffset="8,0"  Texture="buttonsides.dds" />
      </ShowOnMouseOver>

      <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_DIALOG_BUTTON_OK" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
    </GridButton>
  </Grid>
</Context>
```

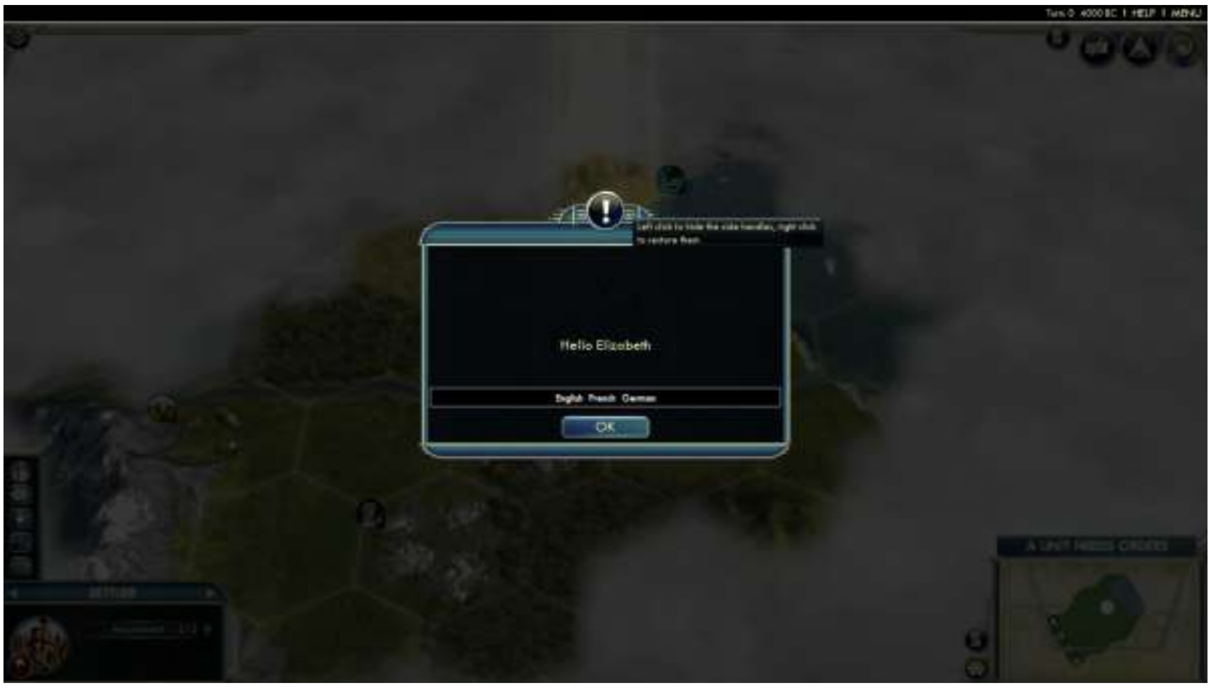
and add the following to "UI/Dialog.lua"

```
function OnLeftIconButton()
  Controls.SideHandles:SetHide(true)
end
Controls.IconButton:RegisterCallback(Mouse.eLClick, OnLeftIconButton)

function OnRightIconButton()
  Controls.SideHandles:SetHide(false)
end
Controls.IconButton:RegisterCallback(Mouse.eRClick, OnRightIconButton)
```

and add the following to "XML/DialogText.xml"

```
<Row Tag="TXT_KEY_TEST_DIALOG_BUTTON_ICON_TT">
  <Text>Left click to hide the side handles, right click to restore
them</Text>
</Row>
```

Save the changes, rebuild the mod, restart Civ 5, re-enable the mod and start a new game.



Sometimes we don't want the full graphical effect of a GridButton, we just want the text to be clickable. For this there is the TextButton

```
<TextButton ID="LangEN" Size="120,24"
String="TXT_KEY_TEST_DIALOG_BUTTON_ENGLISH" Font="TwCenMT16"
ColorSet="Beige_Black_Alpha" FontStyle="Shadow"
MouseOverStyle="SoftShadow" />
```

which is almost identical to the <Label> element except we can register a callback for it and specify how the text is to change as the mouse passes over it (the MouseOverStyle attribute) - the options are "no change" (ie omit the attribute) or "SoftShadow" (so take it or leave it!)

We can also pretty much make anything clickable, by surrounding it with the <Button> element. Usually we want to make an icon (image) clickable.

```
<Button ID="IconButton" Anchor="C,C" Offset="0,-6" Size="80,80"
ToolTip="TXT_KEY_TEST_DIALOG_BUTTON_ICON_TT">
  <Image Anchor="C,C" Size="80,80"
Texture="NotificationFrameBase.dds">
    <Image Anchor="C,C" Size="80,80" Texture="NotificationGeneric.dds"
/>
  </Image>
</Button>
```

The Button element has no visual appearance itself, it just serves to make whatever it contains clickable (and provided a mouse-over tooltip)

To group things together and provide a background for them (usually black) we can use the <Box> element. Using two nested box elements - the outer one white and the inner one black, we can draw a border around elements.

```
<Box Anchor="C,B" Offset="0,100" Size="560,30" Color="White,255">
  <Box Anchor="C,C" Size="558,28" Color="Black,255">
    ...
  </Box>
</Box>
```

The outer box is located at our usual place just above the OK button (Anchor="C,B" Offset="0,100") and draws a solid white (Color="White,255") block 560 pixels wide by 30 pixels high.

The inner box is centred within the outer box (Anchor="C,C") and draws a solid black (Color="Black,255") block 558 pixels wide by 28 pixels high - which leaves a single pixel width white border showing.

Sometimes we just want to group things so we can refer to them by a single name (usually for the purposes of hiding/showing them all together) and for that we can use the Container element - a container is just a transparent box

```
<Container ID="SideHandles" Size="600,400" Anchor="C,C">
  ...
</Container>
```

Summary

This tutorial has covered the basic Civ 5 User Interface (UI) elements (and any Lua code needed to make them work), and they are

- Contexts
    - Context
- Groupings
    - Grid
    - Box
    - Container
- Visual Elements
    - Label
    - Image
- Buttons
    - GridButton
    - TextButton
    - Button
- Mouse Over Events
    - ShowOnMouseOver (also HideOnMouseOver)
    - AlphaAnim
- Stacks
    - Stack

Between them, these comprise around 80% of the elements used within the core game user interface.

All of the XML and Lua code for the examples in this tutorial can be downloaded from the Mod Hub as " Test - UI Tutorial - 1 Basics" (in the Other category). Each dialog step is included separately and can be displayed from the FireTuner Lua Console tab by selecting the Dialog context and then entering "ShowN()" in the command line (where N is the step to display, eg "Show2()", "Show4b()", etc)

There are other references for the Civ 5 UI elements available

- http://www.weplayciv.com/forums/entry.php?45-GUI-Buttons-and-Popups-The-Right-Way
- http://forums.civfanatics.com/showthread.php?t=399743
- http://forums.civfanatics.com/showthread.php?t=408628

I make no apologies for "re-inventing part of the wheel" with Part 1 of this series of UI Tutorials. Part 2 will cover creating buttons dynamically from Lua.