

Civilization 5 UI Tutorial

Part 4, Other Input Controls

Version: 1.0

Status: Draft

Date: 18 May 2012

Author: William Howard

Contents

Overview	3
Create an Input Mod	3
Input 1 - Push Buttons	3
Input 2 - Edit Boxes	9
Input 3 - Sliders.....	12
Summary	19

Overview

The objective of this tutorial is to cover the remaining Civ 5 User Interface (UI) input elements and controls - namely Check Boxes, Radio Buttons, Edit Boxes and Sliders.

The tutorial takes the practical approach - first we will write some "code", then we will look at what it does. At each step we will have working UI dialogs - which may or may not in themselves be useful - but which can be used as a working starting point for your own UI mods.

This tutorial assumes you have used ModBuddy and FireTuner to create your own simple non-UI mods, and also that you have a basic understanding of XML. Some Lua will be needed, but that will be explained in each step.

All the code in this tutorial can be downloaded from the in-game ModHub as "Test - UI Tutorial - 4 Other Input Controls" found under the "Other" category.

So, to start we need a ModBuddy project ...

Create an Input Mod

Using ModBuddy, create a new mod called "Test - Input" (or some such).

To this mod add the two folders "UI" and "XML". In the UI folder create the two files "Input.xml" and "Input.lua" (delete the standard content added to these files). In the XML folder create the file "InputText.xml" (you can leave the standard content in this file).

In the mod's properties, on the "Mod Info" tab, uncheck "Affects Saved Games". On the "Actions" tab, add an "On Mod Activated - Update Database" entry for "XML/InputText.xml". On the "Content" tab, add an "InGameUIAddin" for "UI/Input.xml".

Save the project.

Input 1 - Push Buttons

In "Tutorial 1 - The Basics", we developed a dialog to show a greeting in one of three languages (Dialog 5) and also to show/hide the dialog box "decoration" (Dialog 6). We are going to merge these two dialogs and change them to use "push buttons" - check boxes and radio buttons.

Add the following to the "UI/Input.xml" file

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

  <Grid Size="600,400" Anchor="C,C" Style="Grid9DetailFive140"
  ConsumeMouse="1">
    <Image ID="TopDecoration" Anchor="C,T" AnchorSide="I,O"
    Offset="0,-27" Size="256,64" Texture="DecTop256x64.dds">
```

UI Tutorial - Other Input Controls

```
<Image Anchor="C,C" Size="80,80"
Texture="NotificationFrameBase.dds">
    <Image Anchor="C,C" Size="80,80"
Texture="NotificationGeneric.dds" />
</Image>
</Image>

<Container ID="SideDecoration" 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_INPUT_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" Offset="-30,0" StackGrowth="Right"
Padding="50">
            <RadioButton ID="LangEN" RadioGroup="LangGroup"
IsChecked="1" Size="120,24" TextAnchor="R,C"
String="TXT_KEY_TEST_INPUT_BUTTON_ENGLISH" Font="TwCenMT16"
ColorSet="Beige_Black_Alpha" FontStyle="Shadow"/>
            <RadioButton ID="LangFR" RadioGroup="LangGroup"
IsChecked="0" Size="120,24" TextAnchor="R,C"
String="TXT_KEY_TEST_INPUT_BUTTON_FRENCH" Font="TwCenMT16"
ColorSet="Beige_Black_Alpha" FontStyle="Shadow"/>
            <RadioButton ID="LangDE" RadioGroup="LangGroup"
IsChecked="0" Size="120,24" TextAnchor="R,C"
String="TXT_KEY_TEST_INPUT_BUTTON_GERMAN" Font="TwCenMT16"
ColorSet="Beige_Black_Alpha" FontStyle="Shadow"/>
        </Stack>
    </Box>
</Box>

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

UI Tutorial - Other Input Controls

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

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

<CheckBox ID="ShowDecoration" Style="SquareCheck" Anchor="L,B"
Offset="14,40" IsChecked="1" TextAnchor="R,C"
String="TXT_KEY_TEST_INPUT_DECORATED" />
</Grid>
</Context>
```

Add the following to the "UI/Input.lua" file

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

function OnShowDecoration(bIsChecked)
    Controls.TopDecoration:SetHide(not bIsChecked)
    Controls.SideDecoration:SetHide(not bIsChecked)
end
Controls.ShowDecoration:RegisterCheckHandler(OnShowDecoration)

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

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

function OnLangDE()
```

UI Tutorial - Other Input Controls

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

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

Add the following to the "XML/InputText.xml" file (this is for all examples in this section of the UI Tutorial)

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
    <Language_en_US>
        <Row Tag="TXT_KEY_TEST_INPUT_BUTTON_OK">
            <Text>OK</Text>
        </Row>
        <Row Tag="TXT_KEY_TEST_INPUT_BUTTON_OK_TT">
            <Text>Left click to dismiss the popup</Text>
        </Row>

        <Row Tag="TXT_KEY_TEST_INPUT_DECORATED">
            <Text>Decorated</Text>
        </Row>

        <Row Tag="TXT_KEY_TEST_INPUT_HELLO_LEADER">
            <Text>Hello {1_LeaderName}</Text>
        </Row>
        <Row Tag="TXT_KEY_TEST_INPUT_BONJOUR_LEADER">
            <Text>Bonjour {1_LeaderName}</Text>
        </Row>
        <Row Tag="TXT_KEY_TEST_INPUT_GUTENTAG_LEADER">
            <Text>Guten Tag {1_LeaderName}</Text>
        </Row>

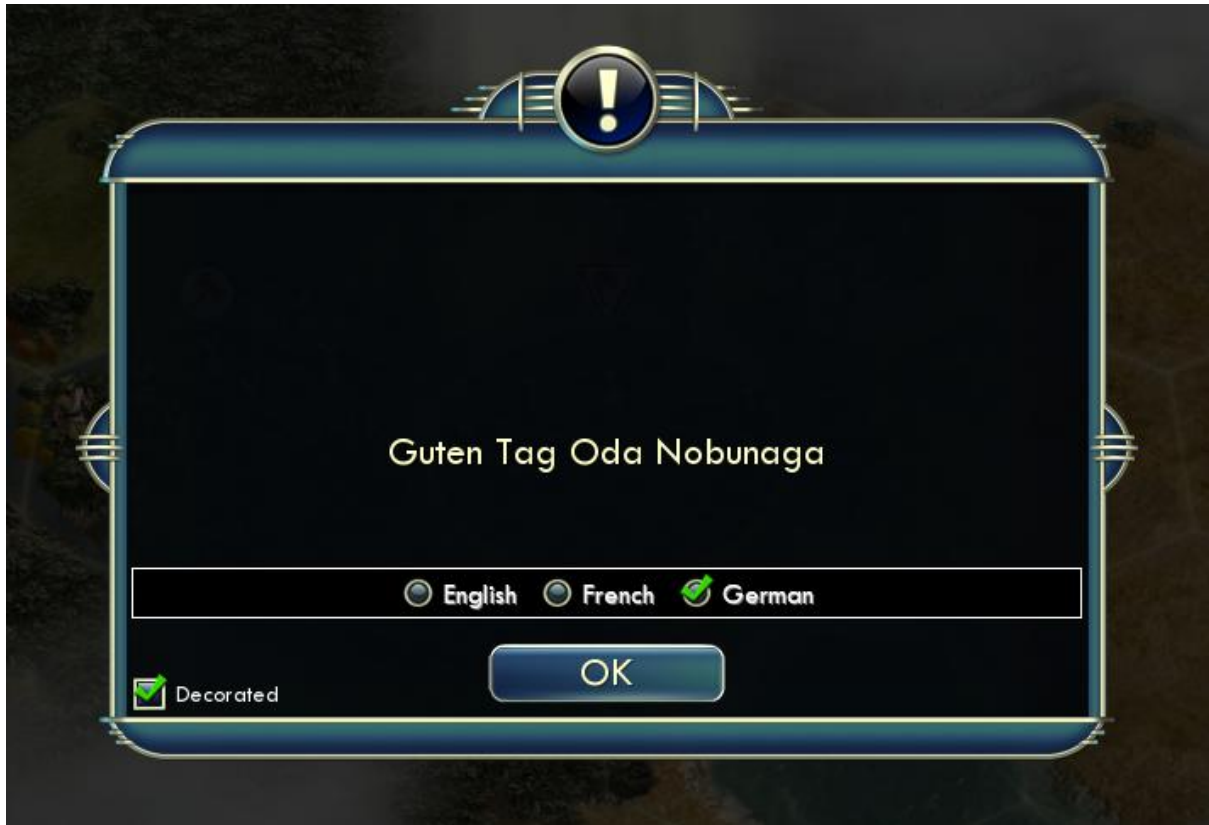
        <Row Tag="TXT_KEY_TEST_INPUT_BUTTON_ENGLISH">
            <Text>English</Text>
        </Row>
        <Row Tag="TXT_KEY_TEST_INPUT_BUTTON_FRENCH">
            <Text>French</Text>
        </Row>
        <Row Tag="TXT_KEY_TEST_INPUT_BUTTON_GERMAN">
            <Text>German</Text>
        </Row>
```

UI Tutorial - Other Input Controls

```
<Row Tag="TXT_KEY_TEST_INPUT_BUTTON_NOTIFY">
  <Text>Notify</Text>
</Row>
<Row Tag="TXT_KEY_TEST_INPUT_BUTTON_NOTIFY_TT">
  <Text>Left click to add the message to the notification
list</Text>
</Row>
<Row Tag="TXT_KEY_TEST_INPUT_NOTIFY_HEADING">
  <Text>Title</Text>
</Row>
<Row Tag="TXT_KEY_TEST_INPUT_NOTIFY_TEXT">
  <Text>Content</Text>
</Row>

<Row Tag="TXT_KEY_TEST_INPUT_RED">
  <Text>Red</Text>
</Row>
<Row Tag="TXT_KEY_TEST_INPUT_GREEN">
  <Text>Green</Text>
</Row>
<Row Tag="TXT_KEY_TEST_INPUT_BLUE">
  <Text>Blue</Text>
</Row>
<Row Tag="TXT_KEY_TEST_INPUT_ALPHA">
  <Text>Alpha</Text>
</Row>
</Language_en_US>
</GameData>
```

Save the files and build the mod. Start Civ 5, enable the mod, and start a new game. In the middle of the screen you should see the greeting dialog.



The three standard buttons across the bottom of the screen have been replaced with "radio buttons" (for those not old enough to remember non-digital radios, only one of a group of radio buttons may be selected (checked) at any time - selecting a different one in the group automatically deselects the current one)

```
<RadioButton ID="LangEN" RadioGroup="LangGroup" IsChecked="1" .../>
<RadioButton ID="LangFR" RadioGroup="LangGroup" IsChecked="0" .../>
<RadioButton ID="LangDE" RadioGroup="LangGroup" IsChecked="0" .../>
```

The RadioButton element is pretty much the same as a standard GridButton (hence why I've removed some of the attributes above) with the addition of the RadioGroup and IsChecked attributes. As you would expect, all RadioButtons (no matter where they are in the context) with the same RadioGroup value ("LangGroup" in this case) are part of the same grouping, and only one will be checked (selected) at any time. The IsChecked attribute provides a way to indicate which button should be checked by default - only one RadioButton in a group should have the value "1". Which RadioButton is checked can also be set from Lua with the :SetChecked(bChecked) method.

The Lua code to handle the RadioButtons is the same as for the GridButtons.

The decoration is shown/hidden via a CheckBox element

```
<CheckBox ID="ShowDecoration" Style="SquareCheck" IsChecked="1"
TextAnchor="R,C" ... />
```

A CheckBox is either checked or unchecked, clicking the box changes the "state" - if it was checked it becomes unchecked and vice-versa.

CheckBoxes have two standard styles - round (they look identical to RadioButtons) and square (as shown in the screen grab) - controlled via the Style attribute. The default is round, but the preferred style for check boxes is square, so the Style="SquareCheck" attribute is almost always used. (You can create your own custom checkboxes but that requires you to create two multi-image textures - one for the box and one for the mark - and that is way beyond the scope of this tutorial!)

The TextAnchor attribute determines which side of the box any text label is displayed, and the IsChecked attribute determines the initial checked (1) or unchecked (0) state - as per RadioButtons the checked state can be changed from Lua via the :SetChecked(bChecked) method.

The Lua to handle the CheckBox is slightly different to other buttons

```
function OnShowDecoration(bIsChecked)
    Controls.TopDecoration:SetHide(not bIsChecked)
    Controls.SideDecoration:SetHide(not bIsChecked)
end
Controls.ShowDecoration:RegisterCheckHandler(OnShowDecoration)
```

in that we need to use RegisterCheckHandler and not RegisterCallback and the callback method receives a parameter to indicate if the button is checked (true) or unchecked (false).

Input 2 - Edit Boxes

On very rare occasions we need the user to be able to enter text. (Of the over 100 mods I've written only one has ever needed an EditBox!)

Change the "UI/Input.xml" file to

```
<?xml version="1.0" encoding="utf-8" ?>
<Context ID="Input2">
    <Grid Size="400,230" Anchor="C,C" Style="Grid9DetailFive140"
    ConsumeMouse="1">
        <Label Anchor="L,T" Offset="20,60" Font="TwCenMT22"
        ColorSet="Beige_Black_Alpha" FontStyle="Shadow"
        String="TXT_KEY_TEST_INPUT_NOTIFY_HEADING" />
        <Box Anchor="L,T" Offset="90,60" Size="290,24" Color="Beige,255">
            <Box Anchor="C,C" Size="288,22" Color="Black,255" >
                <EditBox CallOnChar="1" EditMode="1" Size="284,22"
                Anchor="C,C" ID="NotifyHeading" Font="TwCenMT20" MaxLength="40"
                FocusStop="0"/>
            </Box>
        </Box>

        <Label Anchor="L,T" Offset="20,100" Font="TwCenMT22"
        ColorSet="Beige_Black_Alpha" FontStyle="Shadow"
        String="TXT_KEY_TEST_INPUT_NOTIFY_TEXT" />
        <Box Anchor="L,T" Offset="90,100" Size="290,24"
        Color="Beige,255">
            <Box Anchor="C,C" Size="288,22" Color="Black,255" >
```

UI Tutorial - Other Input Controls

```
<TextBox CallOnChar="1" EditMode="1" Size="284,22"
Anchor="C,C" ID="NotifyText" Font="TwCenMT20" FocusStop="1" />
</Box>
</Box>

<GridButton ID="Notify" Disabled="1" Size="140,36" Anchor="C,B"
Offset="-75,50" Style="BaseButton"
ToolTip="TXT_KEY_TEST_INPUT_BUTTON_NOTIFY_TT">
  <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_INPUT_BUTTON_NOTIFY" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
</GridButton>

<GridButton ID="OK" Size="140,36" Anchor="C,B" Offset="75,50"
Style="BaseButton" ToolTip="TXT_KEY_TEST_INPUT_BUTTON_OK_TT">
  <Label Anchor="C,C" Offset="0,-2"
String="TXT_KEY_TEST_INPUT_BUTTON_OK" Font="TwCenMT24"
FontStyle="Shadow" ColorSet="Beige_Black_Alpha" />
</GridButton>
</Grid>
</Context>
```

and change the "UI/Input.lua" file to

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

function OnNotify()

Players[Game.GetActivePlayer()]:AddNotification(NotificationTypes.NOTI
FICATION_GENERIC, Controls.NotifyText:GetText(),
Controls.NotifyHeading:GetText())
end
Controls.Notify:RegisterCallback(Mouse.eLClick, OnNotify)

function Validate(sValue, control, bFire)
  local sText = string.gsub(Controls.NotifyText:GetText(), " ", "")
  local sHeading = string.gsub(Controls.NotifyHeading:GetText(), " ",
  "")

  Controls.Notify:SetDisabled(sText:len() == 0 or sHeading:len() == 0)
end
Controls.NotifyText:RegisterCallback(Validate)
Controls.NotifyHeading:RegisterCallback(Validate)

Controls.Notify:SetDisabled(true)
```

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



As you can see we have a dialog with two text input areas, filling these in and clicking the Notify button adds an entry to the Notification Panel.

```
<Box Anchor="L,T" Offset="90,60" Size="290,24" Color="Beige,255">
  <Box Anchor="C,C" Size="288,22" Color="Black,255" >
    <EditBox CallOnChar="1" EditMode="1" Size="284,22" Anchor="C,C"
    ID="NotifyHeading" Font="TwCenMT20" MaxLength="40" FocusStop="0"/>
  </Box>
</Box>
```

The EditBoxes are similar, so we'll only look at one. An EditBox, other than a thin editing cursor, has no appearance - it is a blank bit of screen a user can type into - so we need to draw a border around it (by using the box-on-box technique).

The CallOnChar="1" and EditMode="1" attributes are a bit complex to explain (and I'm not sure I fully understand them anyway!) - just assume you need them with these values and always add them to your EditBox elements!

The MaxLength attribute controls the maximum number of characters a user can type into the box - if you omit this, the user can just keep typing, so it's advisable to always include a value, say 255.

The FocusStop attribute determines the order that EditBox controls will be activated when the user presses the Tab key - the NotifyHeading edit box has a value of "0" while the NotifyText edit box has a value of "1", enabling the user to tab from one edit area to the other.

Because we set CallOnChar="1", every time the user types into the edit box the core game engine will generate an event which we can listen for with the following Lua. (This is not quite true as some user actions, such as paste or "select all and delete" do not appear to generate events.)

```
function Validate(sValue, control, bFire)
    local sText = string.gsub(Controls.NotifyText:GetText(), " ", "")
    local sHeading = string.gsub(Controls.NotifyHeading:GetText(), " ",
    "")

    Controls.Notify:SetDisabled(sText:len() == 0 or sHeading:len() == 0)
end
Controls.NotifyText:RegisterCallback(Validate)
Controls.NotifyHeading:RegisterCallback(Validate)
```

Confusingly, like standard buttons we use the :RegisterCallback() method to listen for these events, but unlike standard buttons, our callback function receives three parameters

- sValue - the current contents of the control
- control - the control itself
- bFire - true if the last key was Enter/Return, false otherwise

So if the user types "Hello" and then presses the Enter key, we will get the following 6 calls to callback method

1. Validate("H", Controls.NotifyText, false)
2. Validate("He", Controls.NotifyText, false)
3. Validate("Hel", Controls.NotifyText, false)
4. Validate("Hell", Controls.NotifyText, false)
5. Validate("Hello", Controls.NotifyText, false)
6. Validate("Hello", Controls.NotifyText, true)

Usually we are only interested in the "bFire=true" case, but to do something different the sample code tests the contents of both controls and if either are empty, disables the Notify button. (The next section shows how to use the bFire parameter.)

Input 3 - Sliders

On even rarer occasions, we need the user to enter a value within a range, and want to do this with a "slider" - examples of sliders in the standard game interface are the volume controls on the Audio Options screen and the number of City States on the Advanced Setup screen.

Change the "UI/Input.xml" file to

```
<?xml version="1.0" encoding="utf-8" ?>
<Context ID="Input3">
    <Box Style="BGBlock_ClearTopBar" />

    <Grid Size="600,540" Anchor="C,C" Style="Grid9DetailFive140"
    ConsumeMouse="1">
        <Stack Anchor="C,T" Offset="0,50" StackGrowth="Right" Padding="0">
            <Image Size="180,220" Texture="AdvisorForeign001.dds" />
            <Image Size="180,220" Texture="AdvisorEconomic001.dds" />
```

UI Tutorial - Other Input Controls

```
<Image Size="180,220" Texture="AdvisorMilitary001.dds" />
</Stack>

<Box ID="ColourBlob" Size="400,180" Offset="0,70" Anchor="C,T"
Color="Black,255" />

<Container Size="500,150" Offset="0,290" Anchor="C,T">
    <Label Anchor="L,T" Color0="Red" Color1="Black,200"
Offset="0,10" Font="TwCenMT20" FontStyle="Shadow"
String="TXT_KEY_TEST_INPUT_RED"/>
    <Slider ID="RedSlider" Style="Slider" Length="350" Anchor="C,T"
Offset="0,0" />
    <Box Anchor="R,T" Offset="0,4" Size="40,22" Color="Red,255" >
        <Box Anchor="C,C" Size="38,20" Color="Black,255" >
            <EditBox ID="RedValue" EditMode="1" CallOnChar="1"
Size="34,18" Anchor="C,C" Font="TwCenMT18" NumberInput="1"
MaxLength="3" />
        </Box>
    </Box>

    <Label Anchor="L,T" Color0="Green" Color1="Black,200"
Offset="0,50" Font="TwCenMT20" FontStyle="Shadow"
String="TXT_KEY_TEST_INPUT_GREEN"/>
    <Slider ID="GreenSlider" Style="Slider" Length="350"
Anchor="C,T" Offset="0,40" />
    <Box Anchor="R,T" Offset="0,44" Size="40,22" Color="Green,255" >
        <Box Anchor="C,C" Size="38,20" Color="Black,255" >
            <EditBox ID="GreenValue" EditMode="1" Size="34,18"
Anchor="C,C" Font="TwCenMT18" NumberInput="1" MaxLength="3" />
        </Box>
    </Box>

    <Label Anchor="L,T" Color0="Blue" Color1="Black,200"
Offset="0,90" Font="TwCenMT20" FontStyle="Shadow"
String="TXT_KEY_TEST_INPUT_BLUE"/>
    <Slider ID="BlueSlider" Style="Slider" Length="350" Anchor="C,T"
Offset="0,80" />
    <Box Anchor="R,T" Offset="0,84" Size="40,22" Color="Blue,255" >
        <Box Anchor="C,C" Size="38,20" Color="Black,255" >
            <EditBox ID="BlueValue" EditMode="1" Size="34,18"
Anchor="C,C" Font="TwCenMT18" NumberInput="1" MaxLength="3" />
        </Box>
    </Box>

    <Label Anchor="L,T" Color0="White" Color1="Black,200"
Offset="0,130" Font="TwCenMT20" FontStyle="Shadow"
String="TXT_KEY_TEST_INPUT_ALPHA"/>
    <Slider ID="AlphaSlider" Style="Slider" Length="350"
Anchor="C,T" Offset="0,120" />
```

UI Tutorial - Other Input Controls

```
<Box Anchor="R,T" Offset="0,124" Size="40,22" Color="White,255"
>
    <Box Anchor="C,C" Size="38,20" Color="Black,255" >
        <EditBox ID="AlphaValue" EditMode="1" Size="34,18"
Anchor="C,C" Font="TwCenMT18" NumberInput="1" MaxLength="3" />
    </Box>
</Box>
</Container>

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

and change the "UI/Input.lua" file to

```
local fRed = (200/255)
local fGreen = (200/255)
local fBlue = (200/255)
local fAlpha = (160/255)

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

function OnRedSliderValueChanged(fValue)
    local iRed = math.floor(fValue * 255)
    Controls.RedValue:SetText(iRed)

    fRed = fValue
    SetBlob()
end
Controls.RedSlider:RegisterSliderCallback(OnRedSliderValueChanged)

function OnGreenSliderValueChanged(fValue)
    local iGreen = math.floor(fValue * 255)
    Controls.GreenValue:SetText(iGreen)

    fGreen = fValue
    SetBlob()
end
Controls.GreenSlider:RegisterSliderCallback(OnGreenSliderValueChanged)

function OnBlueSliderValueChanged(fValue)
```

UI Tutorial - Other Input Controls

```
    local iBlue = math.floor(fValue * 255)
    Controls.BlueValue:SetText(iBlue)

    fBlue = fValue
    SetBlob()
end
Controls.BlueSlider:RegisterSliderCallback(OnBlueSliderValueChanged)

function OnAlphaSliderValueChanged(fValue)
    local iAlpha = math.floor(fValue * 255)
    Controls.AlphaValue:SetText(iAlpha)

    fAlpha = fValue
    SetBlob()
end
Controls.AlphaSlider:RegisterSliderCallback(OnAlphaSliderValueChanged)

function OnRedValueChanged(sValue, control, bFire)
    if (bFire) then
        local iValue = tonumber(sValue)

        if (iValue < 0) then
            iValue = 0
        elseif (iValue > 255) then
            iValue = 255
        end

        fRed = iValue/255

        Controls.RedValue:SetText(tostring(iValue))
        Controls.RedSlider:SetValue(fRed)

        SetBlob()
    end
end
Controls.RedValue:RegisterCallback(OnRedValueChanged)

function OnGreenValueChanged(sValue, control, bFire)
    if (bFire) then
        local iValue = tonumber(sValue)

        if (iValue < 0) then
            iValue = 0
        elseif (iValue > 255) then
            iValue = 255
        end

        fGreen = iValue/255
```

UI Tutorial - Other Input Controls

```
Controls.GreenValue:SetText(tostring(iValue))
Controls.GreenSlider:SetValue(fGreen)

    SetBlob()
end
end
Controls.GreenValue:RegisterCallback(OnGreenValueChanged)

function OnBlueValueChanged(sValue, control, bFire)
    if (bFire) then
        local iValue = tonumber(sValue)

        if (iValue < 0) then
            iValue = 0
        elseif (iValue > 255) then
            iValue = 255
        end

        fBlue = iValue/255

        Controls.BlueValue:SetText(tostring(iValue))
        Controls.BlueSlider:SetValue(fBlue)

        SetBlob()
    end
end
Controls.BlueValue:RegisterCallback(OnBlueValueChanged)

function OnAlphaValueChanged(sValue, control, bFire)
    if (bFire) then
        local iValue = tonumber(sValue)

        if (iValue < 0) then
            iValue = 0
        elseif (iValue > 255) then
            iValue = 255
        end

        fAlpha = iValue/255

        Controls.AlphaValue:SetText(tostring(iValue))
        Controls.AlphaSlider:SetValue(fAlpha)

        SetBlob()
    end
end
Controls.AlphaValue:RegisterCallback(OnAlphaValueChanged)

function SetBlob()
```


UI Tutorial - Other Input Controls

```
Controls.ColourBlob:SetColor({x=fRed, y=fGreen, z=fBlue, w=fAlpha})  
end
```

```
Controls.RedValue:SetText(fRed*255)  
Controls.GreenValue:SetText(fGreen*255)  
Controls.BlueValue:SetText(fBlue*255)  
Controls.AlphaValue:SetText(fAlpha*255)
```

```
Controls.RedSlider:SetValue(fRed)  
Controls.GreenSlider:SetValue(fGreen)  
Controls.BlueSlider:SetValue(fBlue)  
Controls.AlphaSlider:SetValue(fAlpha)
```

```
SetBlob()
```

save the changes, rebuild the mod, restart Civ 5, re-enable the mod and start a new game, you will be presented with the dialog below.



You can change the colour (Red, Green and Blue values) and the opacity (Alpha value) of the "blob" over the three advisor images either by using the sliders (0-255) or by entering a value directly.

The four slider / edit box combinations are basically identical

UI Tutorial - Other Input Controls

```
<Slider ID="RedSlider" Style="Slider" Length="350" Anchor="C,T"
Offset="0,0" />
<Box Anchor="R,T" Offset="0,4" Size="40,22" Color="Red,255" >
  <Box Anchor="C,C" Size="38,20" Color="Black,255" >
    <EditBox ID="RedValue" EditMode="1" CallOnChar="1" Size="34,18"
Anchor="C,C" Font="TwCenMT18" NumberInput="1" MaxLength="3" />
  </Box>
</Box>
```

Once again the EditBox gets a surrounding box-in-box border, but this time has a NumberInput="1" attribute which restricts the keys that the control will accept to the numbers 0 through 9.

The Slider has a Style (which is always "Slider") and a Length (equivalent to the Size attribute for most controls), note that it is not possible to set the width of a slider - they are fixed.

The Lua code is mainly concerned with making sure that a number in the range 0 to 255 has been entered and setting the other control to match the one that was changed

```
function OnRedSliderValueChanged(fValue)
  local iRed = math.floor(fValue * 255)
  Controls.RedValue:SetText(iRed)

  fRed = fValue
  SetBlob()
end
Controls.RedSlider:RegisterSliderCallback(OnRedSliderValueChanged)

function OnRedValueChanged(sValue, control, bFire)
  if (bFire) then
    local iValue = tonumber(sValue)

    if (iValue < 0) then
      iValue = 0
    elseif (iValue > 255) then
      iValue = 255
    end

    fRed = iValue/255

    Controls.RedValue:SetText(tostring(iValue))
    Controls.RedSlider:SetValue(fRed)

    SetBlob()
  end
end
Controls.RedValue:RegisterCallback(OnRedValueChanged)
```

finally calling SetBlob() to change the colour of the box over the advisor images.

Summary

This tutorial has covered the remaining Civ 5 User Interface (UI) elements for obtaining input from the user, and they are

- Selection Buttons
 - CheckBox
 - RadioButton
- User Input
 - EditBox
- Sliders
 - Slider

All of the XML and Lua code for the examples in this tutorial can be downloaded from the Mod Hub as "Test - UI Tutorial - 4 Other Input Controls" (in the Other category). Each input step is included separately and can be displayed from the FireTuner Lua Console tab by selecting the Input context and then entering "ShowN()" in the command line (where N is the step to display, eg "Show1()", "Show2()", etc)

Part 5 of this series of UI Tutorials will cover how to provide feedback to the user - active tooltip, progress meters, etc.