Continued from Part 1

Feedback 2b - Texture Progress Bars

Being able to see graphically how much we've "done" and how much further we have "to go" is good, but sometimes we'd also like to know how much further we'll be next turn.

We could do this by overlaying two Bar elements, the lower one showing where we'll be next turn and the upper one showing where we are this turn (which will need a BGColor="Black,0" (ie transparent) attribute) - but there is a better way.

Change the "UI/Feedback.xml" file to

```
<?xml version="1.0" encoding="utf-8" ?>
<Context>
  <Grid ID="ProgressGrid" Anchor="R,T" Offset="5,75" Size="255,240"
Style="Grid9DetailSix140" ConsumeMouse="1">
    <Label ID="CultureName" String="TXT_KEY_TEST_FEEDBACK_CULTURE"
Anchor="L,T" Offset="20,50" ColorSet="Beige_Black_Alpha"
Font="TwCenMT20"/>
    <Image ID="CultureFrame" Anchor="L,T" Offset="20,75" Size="214,29"
Texture="MeterBarFrame.dds">
      <TextureBar ID="CultureBar" Anchor="L,T" Offset="3,4"
Size="196,21" Direction="Right" Texture="MeterBarUtopia.dds"/>
    </Image>

    <Label ID="ScienceName" String="TXT_KEY_TEST_FEEDBACK_SCIENCE"
Anchor="L,T" Offset="20,115" ColorSet="Beige_Black_Alpha"
Font="TwCenMT20"/>
    <Image ID="ScienceFrame" Anchor="L,T" Offset="20,140"
Size="214,29" Texture="MeterBarFrame.dds">
      <TextureBar ID="ScienceBar" Anchor="L,T" Offset="15,7"
Size="184,15" Direction="Right" Texture="XPmeter.dds"/>
    </Image>
  </Grid>
</Context>
```

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

```
function UpdateCulture(pPlayer)
  if (Game.IsOption(GameOptionTypes.GAMEOPTION_NO_POLICIES)) then
    Controls.CultureBar:SetHide(true)
 else
    Controls.CultureBar:SetHide(false)

    local fComplete, fCompleteNextTurn
    if (pPlayer:GetNextPolicyCost() > 0) then
      fComplete = pPlayer:GetJONSCulture() /
pPlayer:GetNextPolicyCost()
```

```
      fCompleteNextTurn = (pPlayer:GetJONSCulture() +
pPlayer:GetTotalJONSCulturePerTurn()) / pPlayer:GetNextPolicyCost()
    else
             fComplete = 1
      fCompleteNextTurn = 1
    end

    -- We have to allow for the first 12 pixels being black!
    fComplete = ((fComplete * 184) + 12) / 196
    fCompleteNextTurn = ((fCompleteNextTurn * 184) + 12) / 196

    Controls.CultureBar:SetPercents(fComplete, fCompleteNextTurn)
  end
end

function UpdateScience(pPlayer)
  if (Game.IsOption(GameOptionTypes.GAMEOPTION_NO_SCIENCE)) then
    Controls.ScienceBar:SetHide(true)
 else
    Controls.ScienceBar:SetHide(false)

    local fComplete, fCompleteNextTurn
    local iCurrentTech = pPlayer:GetCurrentResearch()

    if (iCurrentTech == -1) then
      fComplete = 0
      fCompleteNextTurn = 0
    else
      fComplete = pPlayer:GetResearchProgress(iCurrentTech) /
pPlayer:GetResearchCost(iCurrentTech)
      fCompleteNextTurn = (pPlayer:GetResearchProgress(iCurrentTech) +
pPlayer:GetScience()) / pPlayer:GetResearchCost(iCurrentTech)
    end

    Controls.ScienceBar:SetPercents(fComplete, fCompleteNextTurn)
  end
end

function OnChangeEvent()
  local pPlayer = Players[Game.GetActivePlayer()]

  UpdateCulture(pPlayer)
  UpdateScience(pPlayer)
end
Events.GameplaySetActivePlayer.Add(OnChangeEvent)
Events.SerialEventGameDataDirty.Add(OnChangeEvent)
```

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

The panel now displays bars that have both "filled" (solid) and "will be filled" (translucent) parts.

```
<Image ID="CultureFrame" Anchor="L,T" Offset="20,75" Size="214,29"
Texture="MeterBarFrame.dds">
  <TextureBar ID="CultureBar" Anchor="L,T" Offset="3,4" Size="196,21"
Direction="Right" Texture="MeterBarUtopia.dds"/>
</Image>
```

The Image element provides the outer frame (we could have drawn a plain border with the box-in-box technique) and the Bar element has been replaced with a TextureBar element.

The TextureBar element, like the Bar element, has a Direction attribute but you will need to make sure that the dds texture is suitable for the chosen direction.

The TextureBar element doesn't have "filled" and "un-filled" colours, but a single Texture="" attribute. The texture will be "squashed" or "stretched" to fill the size of the TextureBar, so you'll need to make sure the chosen dds texture is either the correct size or can be scaled appropriately.

To set how much of the bar is filled, we need to use some Lua, we could use

```
Controls.CultureBar:SetPercent(fComplete)
```

like before, that that will only set the "filled" part, to set both the "filled" and "will be filled" parts we use

```
Controls.CultureBar:SetPercents(fComplete, fCompleteNextTurn)
```

again our actual Lua code allows for scenarios where Policies are turned off, and our update code is called from events.

Feedback 3 - Meters

Straight lines are good, curves are better!

Change the "UI/Feedback.xml" file to

```
<?xml version="1.0" encoding="utf-8" ?>
<Context>
  <Image ID="GAMeter" Anchor="R,T" Offset="5,100" Size="128,128"
Texture="TechTreeMeterBack.dds">
    <Image ID="GAMeterIcon"  Anchor="C,C" Size="128,128"/>
    <Meter ID="GAMeterDial"  Anchor="C,C" Size="128,128"
Texture="ProductionPanelMeter128.dds" HasShadow="1" Color="White,255"
ShadowColor="White,100"/>
    <Image ID="GAMeterFrame" Anchor="C,C" Size="128,128"
Texture="TechTreeMeterFrame.dds"/>
  </Image>
</Context>
```

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

```
include("IconSupport")

function UpdateGoldenAge(pPlayer)
  if (Game.IsOption(GameOptionTypes.GAMEOPTION_NO_HAPPINESS)) then
    Controls.GAMeter:SetHide(true)
 else
    Controls.GAMeter:SetHide(false)

    local fComplete, fCompleteNextTurn
    local iCurrentTech = pPlayer:GetCurrentResearch()

    if (pPlayer:GetGoldenAgeTurns() > 0) then
      fComplete = 1
      fCompleteNextTurn = 1
    else
      fComplete = pPlayer:GetGoldenAgeProgressMeter() /
pPlayer:GetGoldenAgeProgressThreshold()
      fCompleteNextTurn = (pPlayer:GetGoldenAgeProgressMeter() +
pPlayer:GetExcessHappiness()) /
pPlayer:GetGoldenAgeProgressThreshold()
    end

    Controls.GAMeterDial:SetPercents(fComplete, fCompleteNextTurn)
```

```
  end
end

function OnChangeEvent()
  local pPlayer = Players[Game.GetActivePlayer()]

  UpdateGoldenAge(pPlayer)
end
Events.GameplaySetActivePlayer.Add(OnChangeEvent)
Events.SerialEventGameDataDirty.Add(OnChangeEvent)

IconHookup(18, 128, "BW_ATLAS_2", Controls.GAMeterIcon)
```

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

Most of the "meter" is comprised of static images - the background (GAMeter), the inner icon (GAMeterIcon) and the frame/bezel (GAMeterFrame)

```
<Image ID="GAMeter" Anchor="R,T" Offset="5,100" Size="128,128"
Texture="TechTreeMeterBack.dds">
  <Image ID="GAMeterIcon"  Anchor="C,C" Size="128,128"/>
```

```
  <Meter ID="GAMeterDial"  Anchor="C,C" Size="128,128"
Texture="ProductionPanelMeter128.dds" HasShadow="1" Color="White,255"
ShadowColor="White,100"/>
  <Image ID="GAMeterFrame" Anchor="C,C" Size="128,128"
Texture="TechTreeMeterFrame.dds"/>
</Image>
```

the interesting part is the Meter element (in bold).

A Meter is a circular TextureBar. The opacity of the "filled" portion is taken from the Color attribute (which will almost always be "White,255") and the translucency of the "to be filled" portion is taken from the ShadowColor attribute ("White,100" seems to be the de-facto standard). If the Meter is to have a "to be filled" portion, you must specify the HasShadow="1" attribute, otherwise it won't be displayed.

The same :SetPercent(fComplete) and :SetPercents(fComplete, fCompleteNextTurn) methods are used with the Meter control as they were with the TextureBar control.

Continued in Part 3