Civilization 5 UI Tutorial

Part 6, World Placement

Version: 1.0

Status: Draft

Date: 12 May 2012

Author: William Howard

Contents

Overview

The objective of this tutorial is to illustrate how to place Civ 5 User Interface (UI) controls on the World Map.

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 - 6 World Placement" found under the "Other" category.

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

Create a World Mod

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

To this mod add the two folders "UI" and "XML". In the UI folder create the two files "World.xml" and "World.lua" (delete the standard content added to these files). In the XML folder create the file "WorldText.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/WorldText.xml". On the "Content" tab, add an "InGameUIAddin" for "UI/World.xml".

Save the project.

World 1 - Hello World!

In "Tutorial 1 - The Basics", the very first UI mod we created displayed the text "Hello World!" in the centre of the screen. We are going to start this tutorial with something very similar.

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

```
<?xml version="1.0" encoding="utf-8" ?>
<Context>
  <WorldAnchor ID="Anchor">
    <Label Anchor="C,C" Font="TwCenMT24" FontStyle="Shadow"
ColorSet="Beige_Black_Alpha" String="TXT_KEY_TEST_WORLD_MESSAGE"/>
  </WorldAnchor>
</Context>
```

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

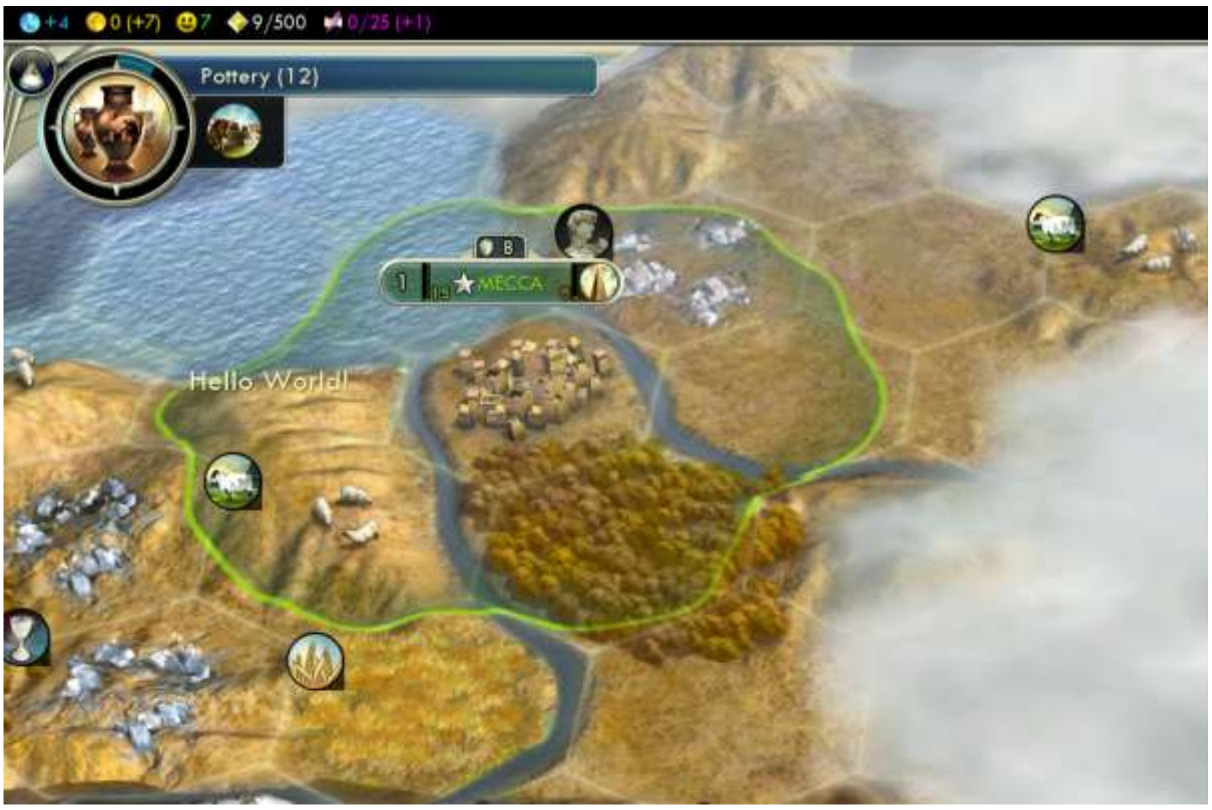
```
function GetWorldPos(pPlot)
  return HexToWorld(ToHexFromGrid({x=pPlot:GetX(), y=pPlot:GetY()}))
end

Controls.Anchor:SetWorldPosition( GetWorldPos(
Players[0]:GetStartingPlot()))
```

Add the following to the "XML/WorldText.xml" file

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <Language_en_US>
    <Row Tag="TXT_KEY_TEST_WORLD_MESSAGE">
      <Text>Hello World!</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 "Hello World!" text. Move the map around - whereas before the text stayed in the centre of the screen, now it stays over the same place on the World Map.



In the context, our "Hello World!" Label is surrounded by a <WorldAnchor> element

```
  <WorldAnchor ID="Anchor">
    <Label Anchor="C,C" ... String="TXT_KEY_TEST_WORLD_MESSAGE"/>
  </WorldAnchor>
```

This permits us to anchor the Label to the World Map and not the screen. In addition to this tag, we also have to tell the game engine whereabouts on the World Map to place the Label, and we do that with Lua

```
Controls.Anchor:SetWorldPosition( GetWorldPos(
Players[0]:GetStartingPlot()))
```

which anchors the label control at our starting plot.

But what's that "GetWorldPos()" function all about? I'm glad you asked - but you probably won't be in a minute!

Civ 5 has THREE co-ordinate systems - grid, hex and world.

The **grid co-ordinate system** has X and Y values and is what the Plot:GetX() and Plot:GetY() methods return, and also how we get plots from the Map object via its GetPlot(x, y) method. It attempts to map the plots onto a standard X-Y co-ordinate system, with the origin (0,0) in the bottom left corner, positive X increasing from left-to-right and positive Y increasing from bottom-to-top. It's how most people think of graph paper.

However, it doesn't work for hex based grids, as we can't calculate the shortest distance between two tiles using that format. To do that, we need to specify the location of a "hex" as three values X, Y and Z. (More information can be found at http://keekerdc.com/2011/03/hexagon-grids-coordinate-systems-and-distance-calculations/) Therefore, underlying the plot grid there is a **hex co-ordinate system** that has X, Y and Z values. Some events give us hex co-ordinates, so the game engine kindly provides two functions, one to convert from hex to grid "ToGridFromHex()" and one to convert the other way "ToHexFromGrid()"

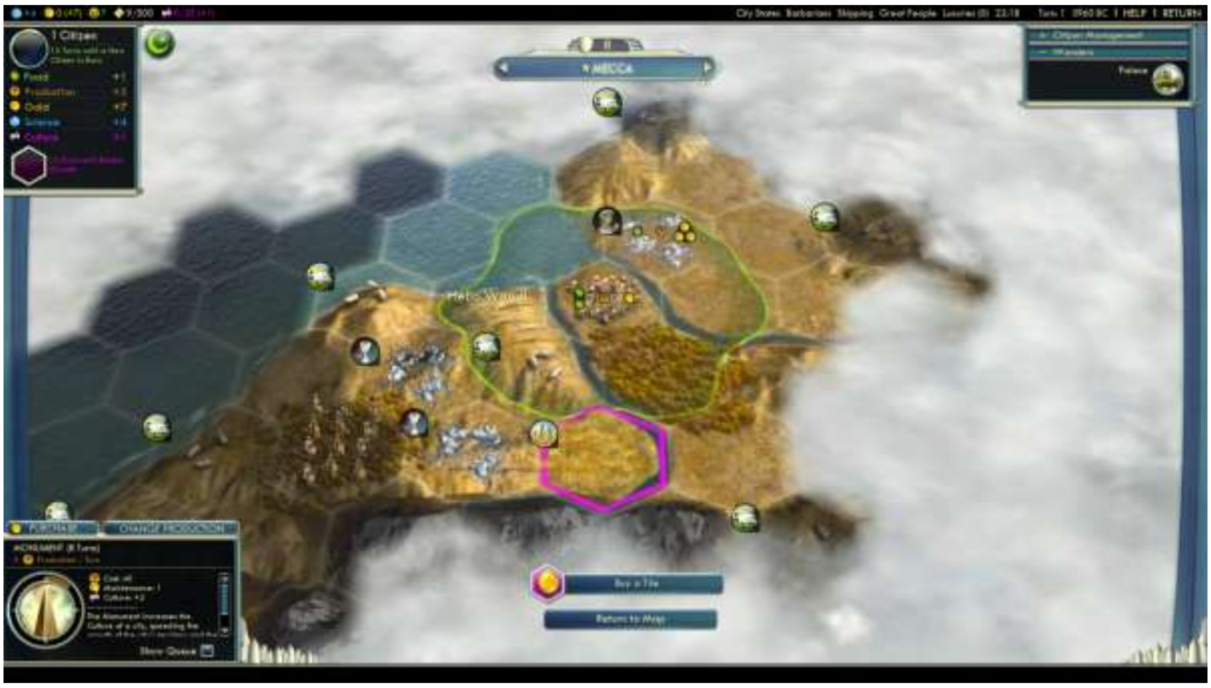
The hex co-ordinate system deals with tiles, but each tile is drawn over many screen pixels, so ultimately we need something with a higher resolution than a tile, and that's where the **world co-ordinate system** comes in. This is also an X, Y, Z based system, but we actually need to know very little about it, other than there is a core game function to convert from the hex co-ordinate system to the world system "HexToWorld()".

So, to convert a Plot to something that SetWorldPosition() can understand we need to do

```
HexToWorld(ToHexFromGrid({x=pPlot:GetX(), y=pPlot:GetY()}))
```

which I really don't want to have to type more than once, so I placed it in the "GetWorldPos()" function!

We have a slight issue in that when we change into City View our label remains on the map, and usually we don't want this.



Continued in Part 2