### 

Lucine.Helpers.TextManager

**A simple TextManager for Lucine Apps**



Lucine.Helpers.TextManager

The role of the text manager is to keep trace of all text of the application.

All texts are defined in an xml file using Ids and Text with the following format

<?xml version="1.0" encoding="utf-16"?>

<TextDatabase xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<Texts>

<Entry Id="ID\_SHOWPANEL1" Text="Show panel 1" />

<Entry Id="ID\_SHOWPANEL2" Text="Show panel 2" />

<Entry Id="ID\_TITLEPANEL1" Text="Panel 1" />

<Entry Id="ID\_TITLEPANEL2" Text="Panel 2" />

<Entry Id="ID\_CLOSEPANEL1" Text="Close Panel 1" />

<Entry Id="ID\_CLOSEPANEL2" Text="Close Panel 2" />

<Entry Id="ID\_QUIT" Text="Quit" />

<Entry Id="ID\_QUITMSG" Text="Ok to quit ?" />

<Entry Id="ID\_OK" Text="OK" />

<Entry Id="ID\_CANCEL" Text="Cancel" />

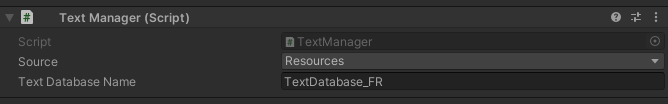
<Entry Id="ID\_ENTERTEXT" Text="type in text..." />

</Texts>

</TextDatabase>

This xmlfile can loaded by the TextManager from a file in StreamingAssets folder, or from a Text resource in a Resources directory (in which case the extension of the file has to be .txt even if xml)

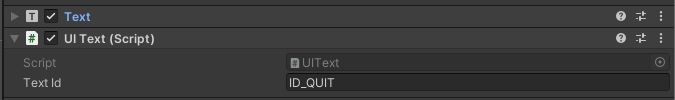
TextManager is a singleton that can be place where you want. In the sample it is located on the Starter GameObject.



You can select the TextDatabase name (no extension has to be given when using resources) and where to load it from. Resources => Resources folder, StreamingAssets, streaming assets folder (in this case you have to include the extension of the file). In the sample the file is present both in streamingassets and resources, in order you can test both methods.

When the file is loaded it fires an OnTextDabaseChanged event, you can register to this event to be notified when something changed in the texts.

That’s what does UIText component



This component requires that the gameobject on which it is, has also a TextComponent.

The only information to set on it is the id of the text in the textdatabase that need to be set in the text component.

On startup the TextComponent is cached and UIText register itself to the OnTextDataBaseChanged event. That way when the text is ready, it is alerted and it can just ask to the TextManager the text that correspond to the id it has and set it in the text component.

*/// <summary>  
/// Change the text when event fired  
/// </summary>*public void OnTextChanged()  
{  
 m\_Text.text = TextManager.Instance.GetText(m\_TextId);  
}

The TextManager is useful to avoid having text in the application. All texts are outside the application.

This way translation are made easier.