Windows Phone Localization for Games

XNA Game Studio provides a robust platform to enable localization of game projects. It is critical for the developer to keep in mind exactly what steps are required to ensure a localizable game. This document provides a step-by-step guide and recommended practices to ensure a game project is localizable.

# Catapult Wars Localized Sample



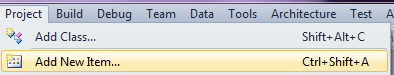
This content details the localized version of [**Catapult Wars Lab**](http://create.msdn.com/education/catalog/lab/catapult_wars)sample from [App Hub](http://create.msdn.com/en-US/education/roadmap).

**IMPORTANT NOTES**

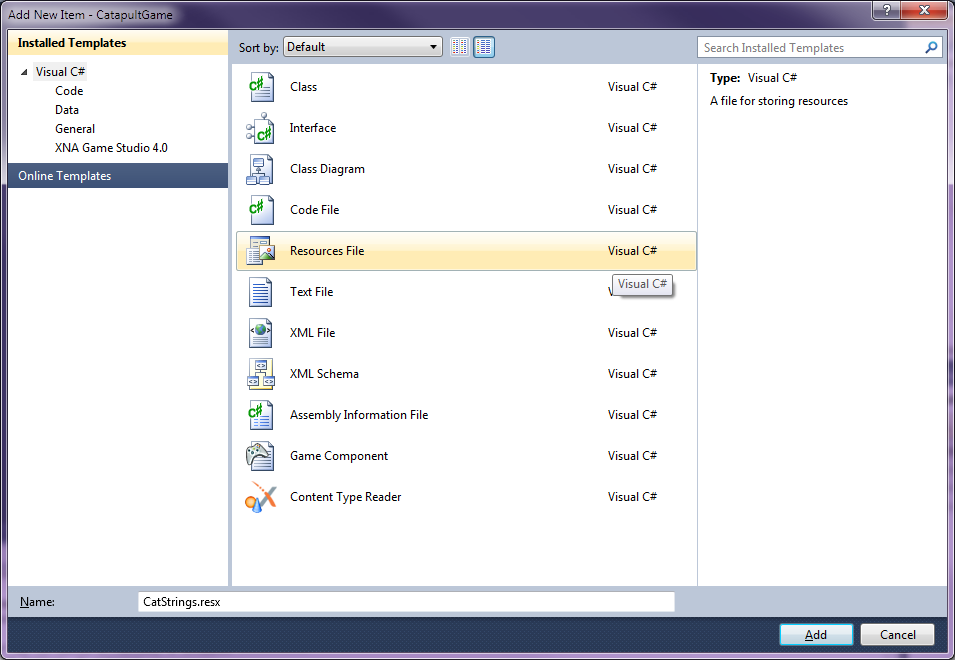
* Machine translation was used to translate English strings.

## Ensuring strings are localizable - Create resource files (RESX)

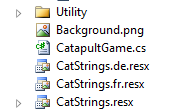
1. In Visual Studio main menu, click **Project >> Add New Item…**



1. Select **Resources File** and name file as appropriate



1. Repeat steps 1 & 2 and create language-specific resources files (e.g., CatStrings.de.resx for German).
2. Below screenshot shows English resource file, **CatStrings.resx**, and German and French resource files – **CatStrings.de.resx** and **CatStrings.fr.resx**, respectively.



## Using strings from resource files

Avoid hard-coding strings in code. Place any strings viewable by user in resource file (RESX). The following example code in **MainMenuScreen.cs** shows how to reference strings placed in a resource file.

**MainMenuScreen.cs *Not ready for localization***

Line 29 // Create our menu entries.

Line 30 MenuEntry startGameMenuEntry = new MenuEntry("Play");

Line 31 MenuEntry exitMenuEntry = new MenuEntry("Exit");

**MainMenuScreen.cs *Ready for localization***

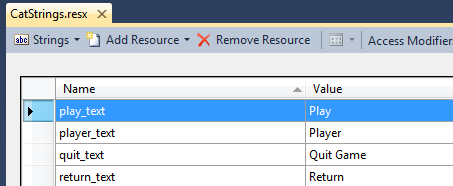
Line 32 // Create our menu entries.

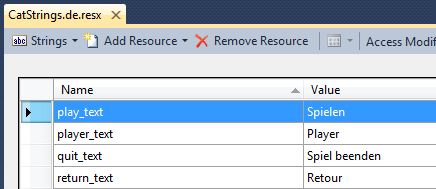
Line 33 // LOCALIZATION

Line 34 // string in string table

Line 35 MenuEntry startGameMenuEntry = new MenuEntry(CatStrings.play\_text);

Line 36 MenuEntry exitMenuEntry = new MenuEntry(CatStrings.exit\_text);





## Working with fonts

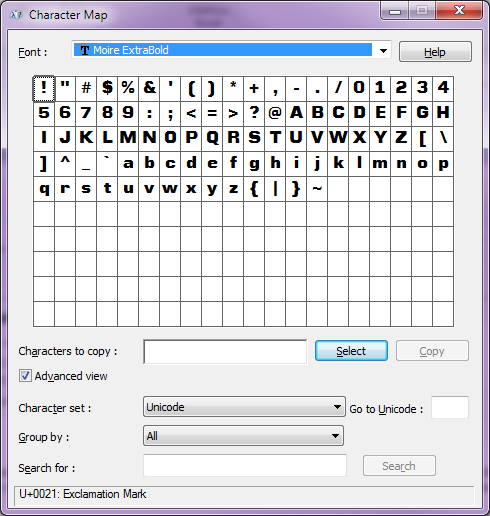
### Glyph support in spritefont

Font must include glyph support for target languages. For example, if the game title will be localized for Western European languages – French, Italian, German, and Spanish – support for glyphs such as â,ë,î,ø,ú is required.

The example code below from **MenuFont.spritefont** details replacing *Moire ExtraBold* font with *Impact* font.

**MenuFont.spritefont *Not ready for localization***

Line 14 <FontName>Moire ExtraBold</FontName>



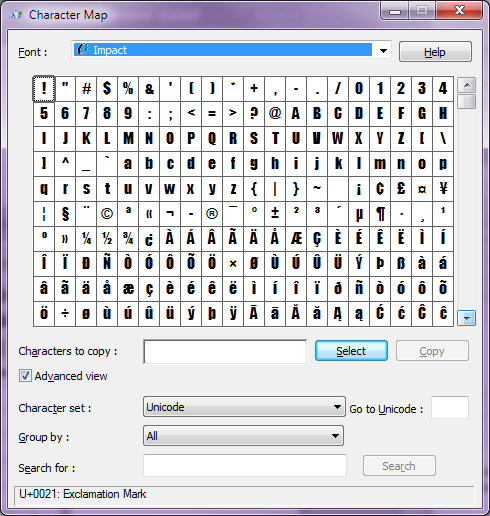
Missing support for extended glyphs



Without support for all required glyphs, the game application will experience a runtime crash.

**MenuFont.spritefont *Ready for localization***

Line 20 <FontName>Impact</FontName>



Proper support for extended glyphs



### Spritefont settings

When using spritefont, it is important to use the proper character range for all required glyphs. Source code from **MenuFont.spritefont** below illustrates a change in glyph range to support target localization languages.

Character range only includes Latin glyphs ***Not ready for localization***

**MenuFont.spritefont**

Line 54 <CharacterRegion>

Line 55 <Start>&#32;</Start>

Line 56 <End>&#126;</End>

Line 57 </CharacterRegion>

Character range includes extended Latin glyphs  ***Ready for localization***

**MenuFont.spritefont**

Line 60 <CharacterRegion>

Line 61 <Start>&#32;</Start>

Line 62 <!--

Line 63 LOCALIZATION

Line 64 Update range to 255 to include glyphs for Western European glyph support

Line 65 -->

Line 66 <End>&#255;</End>

…

Line 71 </CharacterRegion>

### Glyph support in bitmap fonts

When using a tool to create bitmap fonts such as the FontMaker tool, ensure proper range is included for Western glyph support. This tool is available on Microsoft’s Game Developer Network. Use up to 255 for Western European glyph support.

## Working with textures

The image below has hard-coded strings in the image.

**Instructions.png *Not ready for localization***



Generic background where strings can be rendered.

**Instructions.png *Ready for localization***



The source code below from **InstructionsScreen.cs** details how to render text onto the screen during runtime.

**InstructionsScreen.cs**

Line 122 Vector2 textpos = new Vector2(20f,25f);

Line 123 spriteBatch.DrawString(font, CatStrings.instructions\_text1, textpos,

Color.DarkOrange);

Line 124 textpos.Y += font.MeasureString(CatStrings.instructions\_text1).Y;

Line 125 spriteBatch.DrawString(font, CatStrings.instructions\_text2, textpos,

Color.DarkOrange);

Line 126 textpos.Y += font.MeasureString(CatStrings.instructions\_text2).Y;

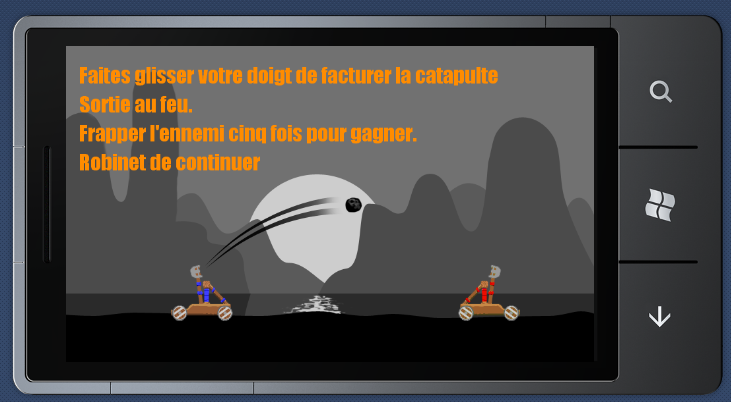
Line 127 spriteBatch.DrawString(font, CatStrings.instructions\_text3, textpos,

Color.DarkOrange);

Line 128 textpos.Y += font.MeasureString(CatStrings.instructions\_text3).Y;

Line 129 spriteBatch.DrawString(font, CatStrings.instructions\_text4, textpos,

Color.DarkOrange);



## Demo code to change UI Culture

At the main menu, the localized sample game includes a third menu option that displays the current UI Culture for the running thread. Clicking on this text (**English (United States)**) will change the UI Culture to the next culture which is French (France).



The language change works in a continuous cycle. After loading French, tapping on **French (France)** will load German (Germany). Tapping **German (Germany)** will cycle back to English (United States) assets.

**MainMenuScreen.cs**

Line 107 void changeLanguage(object sender, EventArgs e)

Line 108 {

Line 109 // Tell the resource manager what language to use when loading strings.

Line 110 if (Thread.CurrentThread.CurrentUICulture.Name == "en-US")

Line 111 {

Line 112 // Changes the CurrentCulture of the current thread to de-DE.

Line 113 Thread.CurrentThread.CurrentUICulture = new CultureInfo("fr-FR");

Line 114 ScreenManager.AddScreen(new InstructionsScreen("fr-FR"), null);

Line 115 }

Line 116 else if (Thread.CurrentThread.CurrentUICulture.Name == "fr-FR")

Line 117 {

Line 118 // Changes the CurrentCulture of the current thread to de-DE.

Line 119 Thread.CurrentThread.CurrentUICulture = new CultureInfo("de-DE");

Line 120 ScreenManager.AddScreen(new InstructionsScreen("de-DE"), null);

Line 121 }

Line 122 else if (Thread.CurrentThread.CurrentUICulture.Name == "de-DE")

Line 123 {

Line 124 // Changes the CurrentCulture of the current thread to fr-FR.

Line 125 Thread.CurrentThread.CurrentUICulture = new CultureInfo("EN-us");

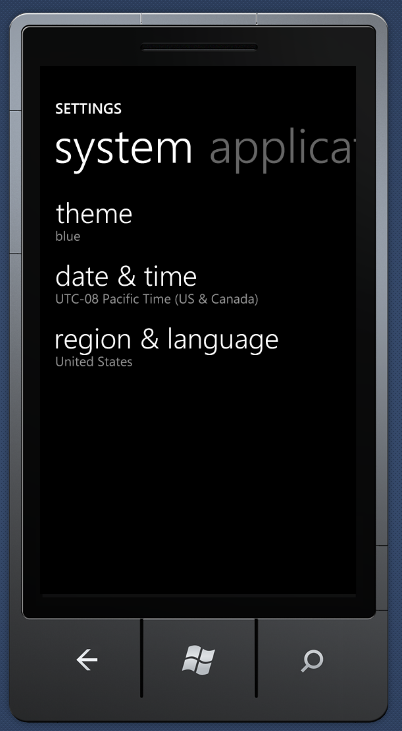
Line 126 ScreenManager.AddScreen(new InstructionsScreen("EN-us"), null);

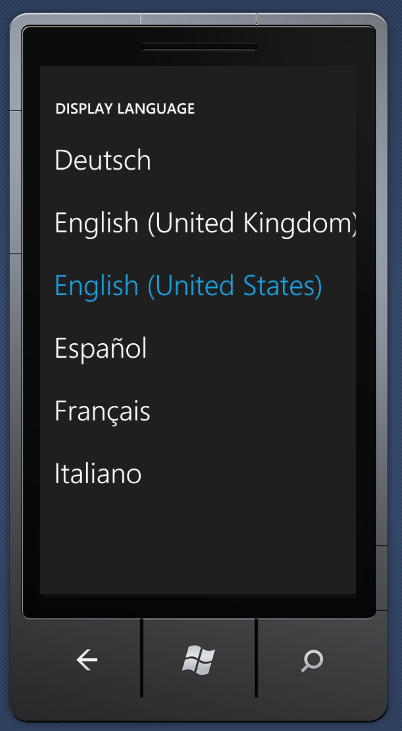
Line 127 }

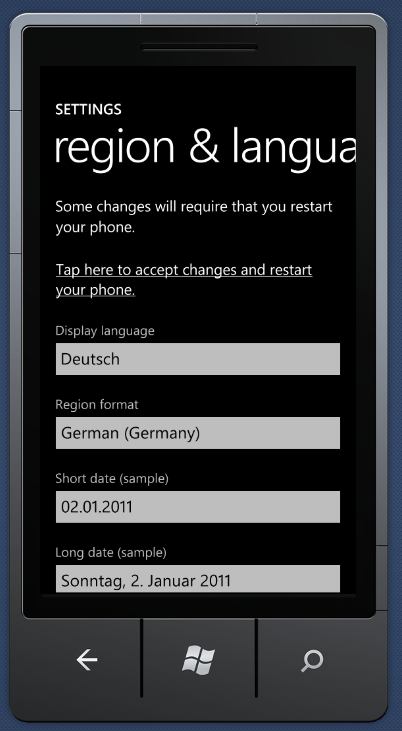
Line 128 }

Note, this code is only for convenience in displaying different language assets without requiring a phone reboot. XNA Platform will automatically load localized assets based on the Windows Phone 7 language settings.

Language settings can be changed in **Settings >> region & language >> Display language** and choosing a different language setting. The phone must be restarted for language changes to enable. If **display language** setting cannot be changed on WP7 device, please refer to Microsoft kb article id #2413345 - <http://support.microsoft.com/kb/2413345>







This document is provided “as-is” for informational purposes only.  Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.  You bear the risk of using it.  Microsoft gives no express warranties, guarantees or conditions.

You may copy and use this document for your internal, reference purposes. This document is confidential and proprietary to Microsoft. It is disclosed and can be used only pursuant to a non-disclosure agreement.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

[© 2011 Microsoft Corporation.  All rights reserved.](http://www.microsoft.com/info/cpyright.mspx)

Microsoft, Xbox, Xbox 360, Zune and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.