Using Globalization in Applications

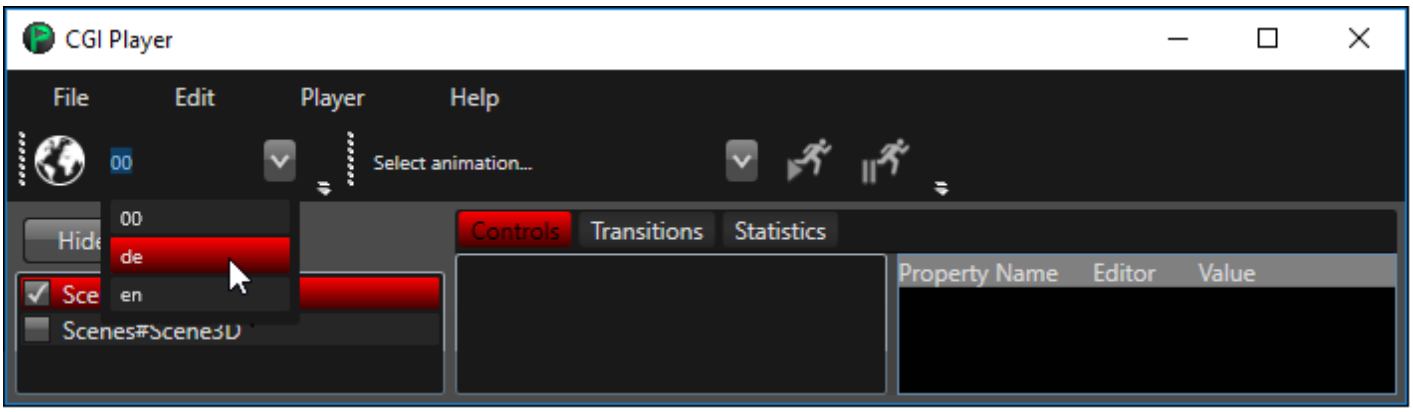
Requirements for Multi Language Support

- CMake setting CANDERA_GLOBALIZATION_ENABLED is enabled
- Sources of CanderaGlobalization available

Example: Set Culture in the Player

The Player provides, if Globalization is enabled in Candera, a checkbox to select all cultures, which are available as language packs in the loaded asset.

1. Open the asset containing language packs (MultiLanguagePack_Asset_Host.bin) and load the scene in the Player.
2. Select the desired language in the drop down list next to the globe icon in the top left corner



Candera Globalization API

The following table explains the usage of Globalization interfaces and how they are used in application code.

Candera::Globalization::Culture	Represents a language, region, and text direction • Locale: ASCII encoded character array, e.g. de_AT. Use Candera::Globalization::Culture::GetLocale() • Display name: Platform-encoded character array (usually UTF-8), e.g. "German (Austria)". Use Candera::Globalization::Culture::GetDisplayName() • Text direction: Enum describing text direction for culture. Use Candera::Globalization::Culture::GetTextDirection()

Candera::Globalization::CultureManager	Access to cultures and changing of current culture
	Singleton:
	Candera::Globalization::CultureManager::GetInstance()
	Retrieve available cultures:
	`return CultureManager::GetInstance().GetCu`
	Get current culture:
	`Culture::SharedPointer culture = CultureMa`
	Get culture name:
	`Culture::SharedPointer culture = CultureMa`
	`return culture->GetDisplayName();`
	Set new culture:
	`Culture::SharedPointer culture = CultureMa`
	`FEATSTD_LOG_INFO(" -> Select culture '%s'"`
	`rc = CultureManager::GetInstance().SetCurr`
	• Candera::Globalization::CultureManager::GetCulture(const Char* locale) const - Retrieval of specific culture by locale

Candera::Globalization::CultureChangeListener	Base class for listeners to culture change events
	Notification of CultureChangeEvents
	• Listener has to derive from Candera::Globalization::CultureChangeListener
	  ◦ override Candera::Globalization::CultureChangeListener::OnCultureChanged(const Culture& culture) {...} for handler
	• Registration as listener
	  ◦ Candera::Globalization::CultureManager::AddCultureChangeListener(CultureChangeListener* listener)
	• Deregistration of listener
	  ◦ Candera::Globalization::CultureManager::RemoveCultureChangeListener(CultureChangeListener* listener)

FeatStd::String and FeatStd::TextId	Point of access to a text in currently active language
	Non-translatable String
	• Will not change with language
	• Use constructor FeatStd::String::String(const TChar*);
	
	Translatable String
	• Changes with language
	• Text-Id from string value
	  ◦ Use FeatStd::TextId::TextId(const Char* id) to get the computed hash id from a given character id
	  ◦ Use FeatStd::String::String(const TextId& id) to access the text for a given Text-Id
	`FeatStd::String text( FeatStd::TextId ("textField_1_3"));`
	• Direct Text-Id: "#\<HEX_HASH_VALUE\>"
	  ◦ Use FeatStd::String::String(UInt32 id) to access the text for a given <HEX_HASH_VALUE>
	`FeatStd::String text(0x13); // for accessing text refe`

Revision #4
Created 2023-02-22 01:48:18 UTC by Tsuyoshi.Kato
Updated 2023-06-19 01:50:20 UTC by Tsuyoshi.Kato