Contents

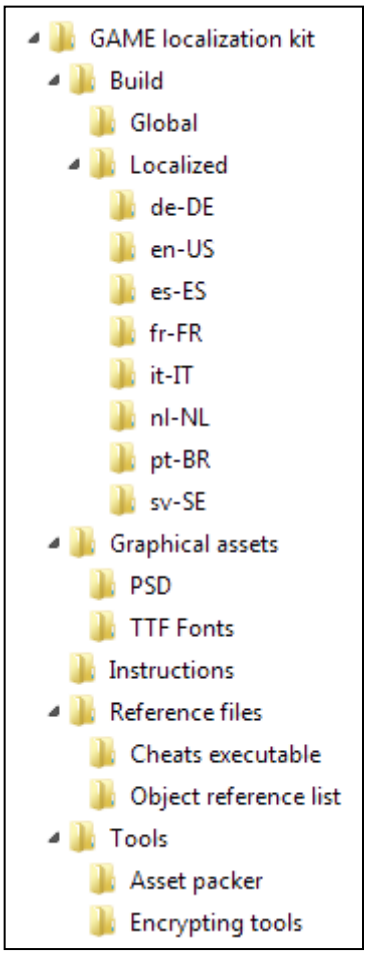
1 Introduction

Localization of casual games is a tricky process. This document is designed to make localization a little easier by describing how games should be prepared for localization, what works well, and what doesn't. Many problems that can be encountered are described in detail, and solutions for these problems are provided. Everything in this document is a result of best practice.

2 Game resources

2.1 Delivery of a localization kit

When delivering games to us, it's preferred to use a logical file structure so we can easily find all the materials that are required for localization. The directory tree on the right is a preferred example of how a typical game can be provided to us. Obviously we want the assets to be as complete as possible to prevent having to ask for extra or more materials. This shortens lead times for a project. A checklist can be found at the end of this document which should help you to prepare all of the assets we need. The combination of assets and game files required for localization is often referred to as **localization kit** or **lockit**.



Essential assets for a complete localization kit:

- **Build** folder containing two important subfolders
  - A **Global** folder which contains an unpacked, unencrypted English build of the game.
  - **Localized** folders that only contain language specific files. E.g: special latin-1 fonts, localized images, translated text files, etc. This is NOT the place for entire packed builds. If the game has not been localized into any languages yet, these folders are generally empty.
- **Graphical assets** should contain source files to edit images. PSD files with TTF fonts are ideal. More information in chapter 2.5
- **Instructions** should contain need-to-know information that we can use during localization. Cheats, packing information, which files need to be translated, usage of scripts and so on. More information in chapter 2.8
- **Reference files** are files that aid us in localizing the game more smoothly. An object reference list, cheats or background information for certain text strings can be useful things to put here. Genre specific examples can be found in chapter 2.9
- **Tools** that are required for localization, or to prepare a final build are stored in this folder. More information in chapter 2.7

The purpose of the **Global / Localized** approach is to make it easier to keep different versions under control, because in this approach we assume that there is only one generic English version of your game (**Global**). Localized files are isolated and stored in separate language folders (**Localized**) to be copied on top of the generic build to create a localized version. When working with other partners besides us, it's highly recommended to use this approach because it simplifies version control greatly.

Throughout the localization process, builds are sometimes sent back and forth between developers and the localization department. By providing all the files in this set structure from the get-go, and always using this structure from there on, changed files can be easily compared to older ones, which makes our lives much easier.

2.2 General

Described below are general issues that are encountered during the localization process.

2.2.1 Stitching

One of the worst things that can be encountered is stitching of words. In short this means that several pieces of text are "stitched" together to form a sentence. This generally works really well in English, but is a huge problem in many other languages. Nouns, pronouns and adjectives can have different endings depending on the words that are stitched together.

EXAMPLE: "I think %s is an expert in both lumbering and mining. You should talk to %s about repairing the train bridge"

Let's assume that the example above is a Spanish sentence. If the first %s, is replaced by a name, the sentence could become incorrect depending on whether the "stitched" word is a male or female character. The word Expert would become "experto" in case of a male character, and "experta" in case of a female character. What we often see

is that text files contain one Male sentence, and a Female sentence but this isn't an ideal solution. The example above illustrates that sentences with more than one stitch also occur, which would mean that there are 4 sentences possible.

A solution would be to write out all possibilities of a stitched sentence in a game's text file like this, without using any variables:

I think Jane is an expert in both lumbering and mining. You should talk to John about repairing the train bridge
I think Jack is an expert in both lumbering and mining. You should talk to Frank about repairing the train bridge

This often results in more text that needs to be translated, but it's the only way to be sure that the text is grammatically correct without understanding any of the grammar rules.

EXAMPLE 2: "You need %s %s and %s %s to build a %s"

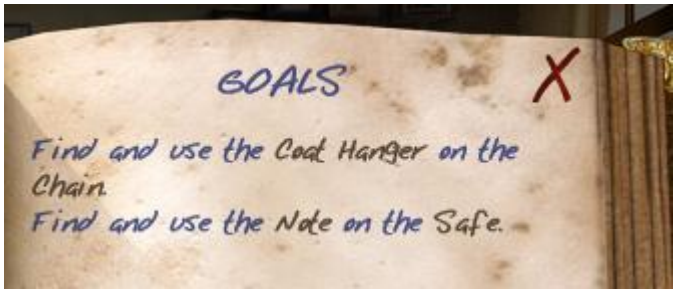
Example 2 illustrates that writing out sentences with 5 (!) variables would result in a huge amount of text. Often there are several clever ways to deal with this. Instead of text, images can be used to replace characters, or resource types, which avoids the use of male / female words. Something like:

[image.jpg] required for [image2.jpg]

Another problem with Example 2 is that the order of the variables can't be changed, which means that our translators can't come up with clever ways to change the order of the words to avoid different word exits. In German, possessive adjectives (my, your, mine), have many variations: mein, meine, meines, meiner, meinem, etc, and that's just for "my". If we know in advance which sentences or words are stitched together, and the order can be changed, many problems can be avoided.

EXAMPLE 3: “Find and use the” + <object> + “on the” + <object>

Whenever specific words are stitched together to form sentences, we can always guarantee that this will cause problems. In the example image displaying different goals you can see a few sentences that have been stitched together. If we break it down into how it’s written in the text file, it’s obvious that when different grammar rules apply, it will cause problems



EXAMPLE 4: "You have %s minute(s) left"

Stitching numbers to a sentence doesn't cause any problems at all. Ideally, for sentences like example 3, there is a singular and plural version.

2.2.2 Hard-coded texts

If text is hard-coded into the game's source code, it's impossible for us to edit this. This is a major problem and localization can't be continued at that point. A fix should first be provided where the text is externalized, before moving on.

Ideally all text is externalized into a single text file which contains all of the text in a game. However, sometimes a few bits of text remain in several other files, or it requires a lot of extra effort to externalize this. If this is the case, please mention this to us by providing a document in the localization pack. It requires quite a bit of extra time when the game is undergoing a linguistic test, to have these extra texts translated, put into the game, and tested again. The example picture displays that a word "Hint" is left behind in one of the script files (a LUA file in this case).

```
GVPopupHint {
    name="status_popup_hint",
    hinttextcontrol = "status_popup_hint_text",
    hint = "Hint",
    vx=175, vy=550,
    w=250,h=200,
    xml_anim = "invisible";
    xml_preloading = true,
    fade_time = 400,
    show_time = 3000,
    --wait_time = 1000,
    self_fade = true,
    states = {normal = "seed_popup"},
```

2.2.3 Splash screens external

When a game is distributed through or networks or portals, different types of splash screens are required. Sometimes when a game is put on another portal, the splash screen needs to be changed. To make this as easy as possible, the splash screen should always be changeable without rebuilding the game.

We would also like to change the order in which the splash screens appear, either by changing a script file, or simply using a splash1.jpg, splash2.jpg structure in which the order is defined.

Definite Don’ts are:

- Don't put the splash screen in the packed files
- No encryption for the splash screen
- No checksums for the splash screens

2.2.4 “30% rule”

Always keep in mind that translated texts can take up much more space than the original English text. The example image illustrates that there's plenty of space for English words in the list of hidden objects. However, especially in German and Italian, words can take up much more space. The word "skate" (5 letters) translates into Schlittschuh in German, which is remarkably longer. As a general rule, 30% more space than the longest English word, should always be reserved for translations.

Other solutions are enlarging text areas or reducing font sizes, but the latter is limited as it reduces readability, especially when players use windowed mode for games.

Depending on how the game is set up, individual text can sometimes be adjusted through script files. This works quite well if only a few texts in a game need some adjustments, but when lots of cropping issues occur, another solution should be pursued. Try to visualize how certain text labels will be displayed when they contain shorter or longer texts. There will always be situations that you cannot foresee. Modifications will need to be made to the game to make translations fit.

2.2.5 No linguistic data outside of game folder

Language specific texts shouldn't be stored in save game locations, unless it's in the game folder itself. Customers often play English (trial) versions of game first, before switching over to a localized version. Unfortunately this sometimes causes issues as certain texts are stored in saved games which are located in the profile data folder instead of the game folder and thus won't be overwritten when installing another language of a game.

2.2.6 Word mini-games

Mini-games in a game (like a puzzle) can often be difficult to localize or even impossible due to technical restrictions. The example image shows a word game where a player has to enter a word which is limited to the regular 26-letter alphabet.

Unfortunately it's quite hard or sometimes even impossible in some languages to come up with a word that has a connection to the game, has a specific length, and most importantly, doesn't contain any special characters. We can always try to localize these word games, but it's very important that these things are mentioned before we start translation. If it can't be localized properly, it's best to remove it from the game entirely to make sure non-English speaking players can solve the puzzle. Provide

as much information as possible in the document describing the localization pack. When these issues are discovered later on in the localization process, they can cause delays.

2.2.7 No translation tools

We often receive specific translation tools when starting up a new project, unfortunately we can't / won't use these. It's very frustrating to need to learn with new tools every time we pick up a new project, so we're only working with regular text files. Luckily we've developed our own set of tools to easily work with pretty much every format we come across so this shouldn't be a problem at all.

2.3 Text files

The vast majority of games use text files for storing text, and an occasional image here and there. To process these files easily, there are several basic rules that you should adhere to which are described in the following paragraphs.

2.3.1 Format

There are many different ways to put the linguistic content of a game into a text file. XML files are usually the easiest to handle, especially when it has a simple and clean "Key / Value" as can be seen in the example.

Strings.xml

```
...
<Text id="TT_X_QUIT">Click here to quit</Text>
<Text id="TT_RESTORE">Click here to play in a window</Text>
<Text id="TT_SAVEGAME_YES">Start a previously saved game</Text>
...
```

We are able to handle pretty much every file, however there are a few less than ideal formats where issues can occur. An example is where words are separated by spaces or tabs, and especially when this hasn't been done consistently, this is not convenient to work with. Not only doesn’t it have a clear structure, it’s also very easy to break the game, and then it becomes very hard to find out where something goes wrong.

We often see Excel XML files (with XLS file extension, and in XML format) and it's not a problem to work with these.

2.3.2 Amount of files, consistency and file encoding

Especially with hidden object games, the amount of files containing text can be quite high. Preferably we have two text files for these games: one with all the hidden objects and one with the rest of the texts. For regular games only one text file should suffice. If there is a reason to use multiple files, please make sure that they all have the same file encoding (preferably UTF-8, or Windows-1252). Often there are many different file encoding used, and some of those don't support all special characters properly when saved in a different encoding, or the encoding isn't supported by the game.

Also important is to make sure that all of the files have the same format. If one file has a simple Key / Value structure, please use the same format in the other text files so that there is no confusion, Keep it simple, clean, and easy to understand.

2.3.3 Settings in text files

Localization is not only about translating texts in a game, another important factor are a country/language's Regional Options. Often games are delivered in English and there is no support for a custom thousand separator which looks quite weird in some languages, as a comma, which is the thousand separator in English, is the decimal symbol in Dutch. If there is no way to have a custom thousand separator, it's best to just leave it out entirely. Regional settings that should be considered are (next page):

	en-US	nl-NL	de-DE	sv-SE	it-IT	es-ES	fr-FR	Pt-BR
Numbers								
Decimal Symbol	.	,	,	,	,	,	,	,
Digit Grouping Symbol	,	.	. (see note*)	[space]	.	.	[space]	.
Time								
Time format	h:mm:ss tt	H:mm:ss	HH:mm:ss	HH:mm:ss	H.mm.ss	H:mm:ss	HH:mm:ss	HH:mm:ss
Time seperator	:	:	:	:	:	:	:	:
AM symbol	AM							
PM symbol	PM							
Seconds	s	sec	Sek. or s	s	s	seg.	**s**	**s** or **"**
Minutes	min	min	Min. or m	min	min	min.	min	m or '
Hours	h	u or uur	Std. or St., very rarely h (for Latin "Hora")	tim or t	h.	h.	h	h or :
Days	d	dagen	t	d	g	d	j	d
Weeks	w	wk	W (not very common)	v	set	sem	sem.	sem.
Date								
short date	M/d/yyyy	d-M-yyyy	dd.MM.yyyy	yyyy-MM-dd	dd/MM/yyyy	dd-MM-yyyy	dd/MM/yyyy	dd/MM/yy
example	7/6/2005	6-7-2005	06.07.2005	2005-07-06	06/07/2005	06-07-2005	06/07/2005	06/07/05
date seperator	/	-	.	-	/	/		/
long date	dddd, MMMM dd, yyyy	dddd d MMMM yyyy	dddd, d. MMMM yyyy	den 'd MMMM yyyy	dddd d MMMM yyyy	dddd, dd' de 'MMMM' de 'yyyy	dddd d MMMM yyyy	dddd, d' de 'MMMM' de 'yyyy
Example	Wednesday, July 06, 2005	woensdag 6 juli 2005	Mittwoch, 6. Juli 2005	den 6 juli 2005	mercoledì 6 luglio 2005	miércoles, 06 de julio de 2005	mercredi 6 juillet 2005	terça-feira, 6 de maio de 2008
Currency								
Positive	$1,234.56	€ 1.234,56	1234,56 €* (see note*)	1 234,56 kr	€ 1.234,56	1.234,56 €	1 234,56 €	$ 1.234,00
Negative	-$1,234.56	€ 1.234,56-	-1234,56 €* (see note*)	-1 234,56 kr	€ -1.234,56	-1.234,56 €	-1 234,56 €	($ 1.234,00)

* Separating dot only for numbers higher than 9999 (or 9999,99 €)

2.4 Script files

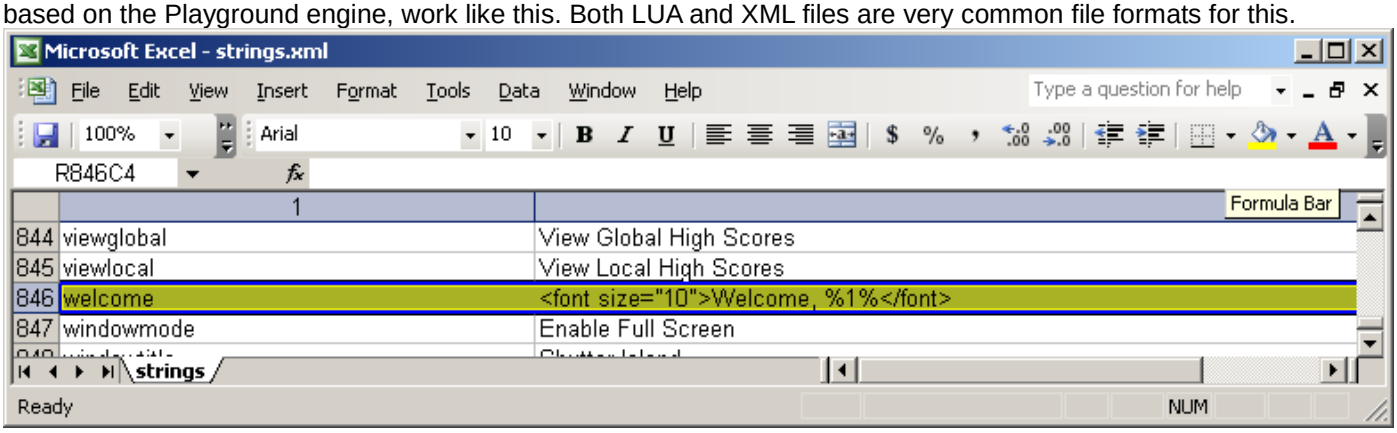
Script files like LUA, XML, CFG and INI can be used to make all kinds of adjustments to all kinds of aspects in the game.

2.4.1 Adjusting font sizes

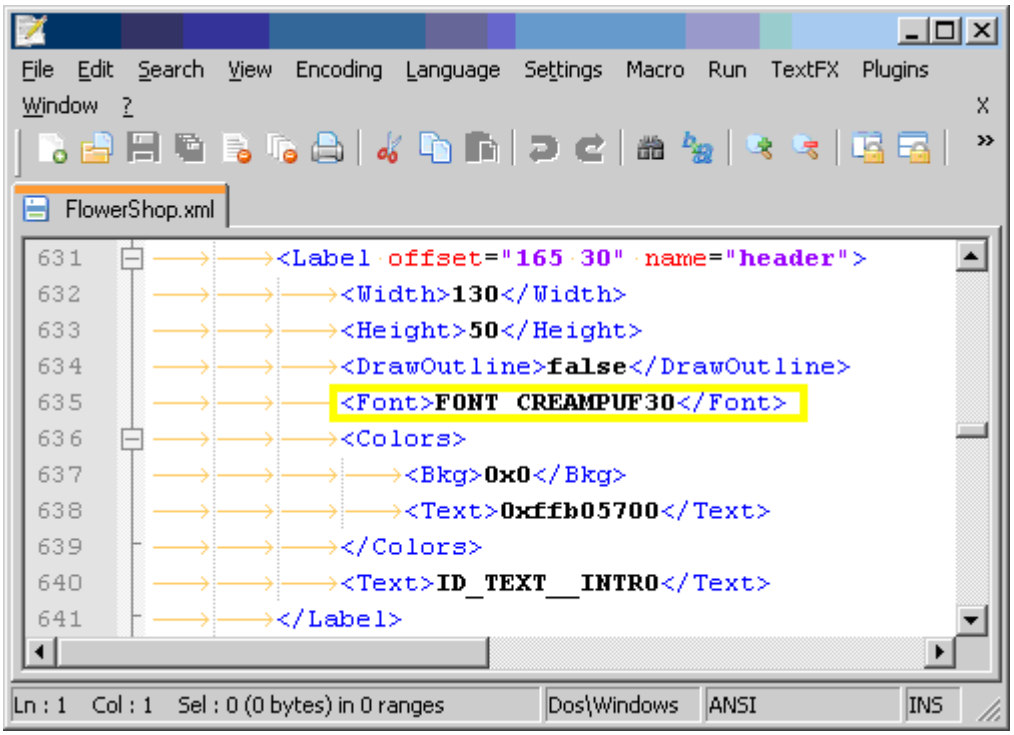
Often translated texts don't fit properly in certain areas like buttons or they overlap on other places where there shouldn't be any text. A good way to prevent/fix this problem, is to allow the font size to be adjustable.

This is something that should be adjusted either in the text file itself, the file which contains all of the text strings from the game (see Excel image below). Or this can be done in an additional configuration file. A good example of the first, are games based on the Playground engine. Pretty much every font in the game can be adjusted by using simple HTML tags as can be seen in the Excel screenshot below.

Solution 2, where font sizes can be defined in external files are much more common. Pretty much every game where fonts are allowed to be changed, except for games based on the Playground engine, work like this. Both LUA and XML files are very common file formats for this.



For a big amount of games graphic, or bitmap fonts are used, and usually they can't be scaled properly. When this happens, it's often required to have a new font file created, which has smaller characters.



2.4.2 Adjusting text positions

When text strings become smaller or longer, the alignment can become a problem. When buttons are supposed to be center aligned, but the coordinates of a text label is aligned to the left (in the center of a button), it looks bad. The following picture visualizes the problem and should make clear what should be done, which is basically to always keep in mind what happens to a text when it changes size.

ORIGINAL

NOT ALIGNED PROPERLY

LABEL TOO SMALL

ENOUGH SPACE. ALIGNMENT GOOD

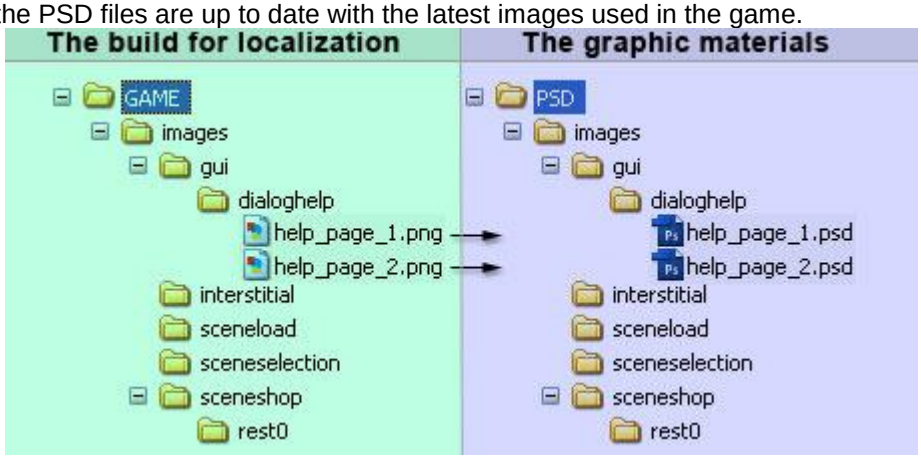
2.5 Images

Sometimes it's unavoidable to have text in image files. While this isn't very convenient, we can perfectly handle these types of files.

2.5.1 Proper file structure

In order for us to localize image files as quickly and easily as possible without having to search through everything, it's recommended to deliver the PSD files in the same structure as the actual game, as displayed in the image below. If possible, please make sure the game doesn't use any uncommon image formats. Ideally the format should be JPG or PNG, but other formats are also workable, like: TGA or TTF. We prefer to avoid: GIF, JPE, PVR, DDS and JP2

- Do not put every image of the game in a single PSD file, this is very inconvenient, and editing these massive files is very time consuming.
- Make sure the PSD files only display English language, and not Russian, Japanese or any other language.
- Prevent having two image files for a single image (avoid having a separate transparency image, please use a format that has alpha layer support like PNG) Example images are on the next page.
- Make sure the PSD files are up to date with the latest images used in the game.



2.5.2 Text labels

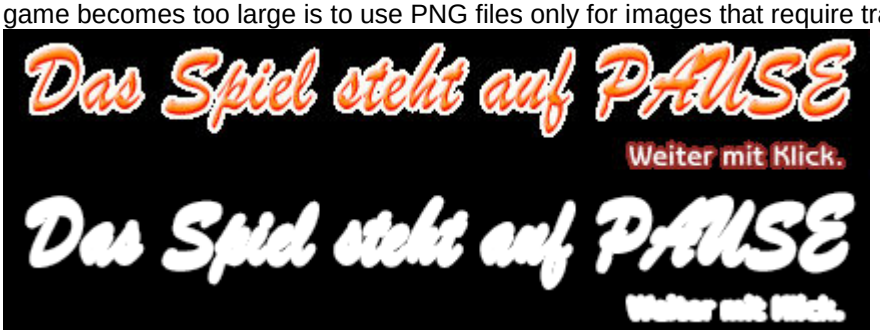
In order for us to efficiently localize the PSD files, text should always be put in a "text label" layer, instead of a rasterized layer that can't be edited. When this is not possible, e.g. in case of 'designer text', which needs to be replaced by another font, please put the text in a separate layer, so that it can easily be removed, and replaced by other text.

What also helps is to have the text in "Paragraph" text layer format instead of "Point" text layer, to make sure the width, height and location of the text box have been set properly, and also the alignment shouldn't cause any issues when the text becomes bigger. Please see the image in "Adjusting text positions" for clarification. For more information on paragraph and point text, please see the URL in the reference list.

2.5.3 Multiple files with same image and text

For some games it cannot be avoided that multiple image files have to be used to display the same graphic in-game. A good example is multiple 'states' for buttons. A button can be in rest position, a cursor can hover over it, or it can be pressed. The image below shows these three forms. In rare cases, there's also a 'release' state'. For these cases, it's imperative that the precautions described in the previous chapter are followed for best result. That way the text won't 'jump' or move erratically.

Alpha layers, which are used for transparency should always be saved in the image file itself when possible, to prevent having separate alpha layer files as demonstrated in the image below. We prefer to work with PNG files whenever possible, as this is a losless format, and it supports transparency perfectly. A good compromise when the game becomes too large is to use PNG files only for images that require transparency, and JPG for the rest.



2.6 Fonts

Problems with fonts are one of the main reasons why a localization project gets delayed. Below you can find several tips and advice to avoid problems.

2.6.1 Special characters

Throughout this document the term "Special characters" is often mentioned. By this we mean the characters outside of the alphabet (abc...xyz). Most Western European languages require more letters than just the regular alphabet. All of these extra characters are contained within the Latin-1 character set, also called: ISO/IEC 8859-1. Please note that for the Cyrillic character set (which is used for Eastern European languages), Latin-1 won't suffice. Two things are very important for these characters when localizing a game into other languages.

- The encoding needs to support the characters. If for example the files are provided in ANSI format, and this is the required format for the game, it's going to be a problem when the game needs to be translated into an eastern European language as the encoding doesn't support Cyrillic characters.
- Font files must contain all of the required characters. Sometimes we receive games with a note that they support Unicode, and while that's all fine and dandy, it doesn't mean that the actual characters are within the font file. Always make sure that the characters are present in every font file.

2.6.2 Missing characters in fonts

Some TTF fonts are 'designer fonts' and we can't easily create 'special characters' for them. In these cases we have a couple of options.

- The developer needs to create the missing characters for these fonts. Usually only Latin-1 support is required, unless the game needs to be localized into other than Western European languages.

- A substitute font needs to be provided. Preferably, a font that is somewhat similar to the original would be best as it's bad practice to use Arial everywhere.

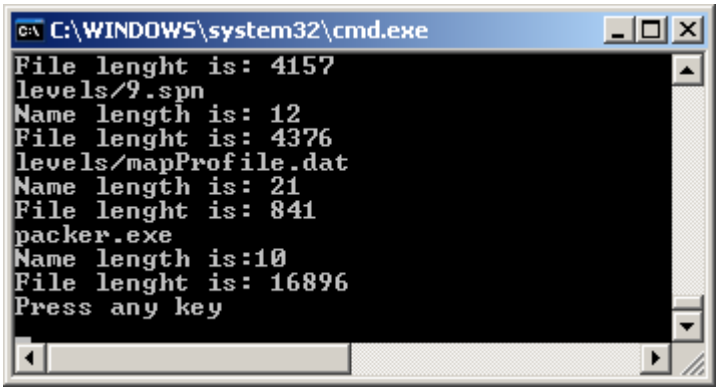
Bitmap fonts are even trickier to handle as we usually don't have any tools to edit them. Every developer uses his own format and therefore it's their responsibility to supply us with fonts that support the required characters.

2.7 Tools

We come across all kinds of different tools, below you can find the most common ones and which requirements they should adhere to.

2.7.1 Building and Packing tools

One of our requirements is that we can create localized builds locally and without developer help. This means that sometimes game files have to be converted to another format when a game hasn't been designed with localization in mind and thus doesn't have any native support for pretty XML files. When this is the case, a build script or converter tool (whatever works) is required to do this ourselves. Important for these tools is:



- Command line support, and no user input. We usually need to create 7 builds several times during the localization process and we don't want to manually perform all kinds of steps during that process. So please don't put any time and effort in creating fancy GUIs, we won't be using them.
- Relative path support for command line parameters
- Please provide command line help or a small text file with instructions

Good examples of command line packing tools are PFPACK (often used for games based on Playground engine) or POPPAK which are quite common. In the example above, you can see an example of a tool which isn't very convenient as it requires user input *"press any key"*.

We encourage developers to use packing tools because this protects game assets, but it also reduces installation times for customers.

2.7.2 Encryption

Some games have an encryption tool instead of a packing tool. While in essence they do the same thing, which is protecting game files, the latter is preferred.

2.7.3 Clean builds

Please don't use any exotic file or folder names in the game, and make sure the game is as clean as possible:

- No dots in folder names
- No hidden thumbs.db files in folders that contain images

- No hidden MacOS folders in Windows builds
- When save files are stored in the game folder, please remove them before sending them over to us
- Scan the game for viruses

2.8 Documentation

In order to speed up the localization process, and have as little delays as possible because of questions and answers going back and forth, it's encouraged to provide a document explaining the localization steps. Also please go through the checklist at the end of this document and supply as with as much information as possible.

2.8.1 Cheats

During the localization process, there are several points where the game has to be tested. The linguistic test is the first time that translators play through the game to see if everything looks right. No matter how fun the game is, we need to be able to move through the game quickly as sometimes this test has to be performed multiple times. Often only a small segment of the game has to be checked or rechecked, and it that case it must be possible to move to a specific area, without playing the game from start.

General

- Win level
- Win Expert level
- Lose level (especially when there is different text each time you lose, or if it takes 20 minutes to lose a level)
- Unlock map
- Unlock map level by level
- Skip to level
- Skip/win mini-game – particularly nice for puzzle or word games (this cheat is often forgotten)
- Win trophies (per trophy)
- Skip introduction cheat, particularly for a game with a long introduction scenes
- Scroll through cut-scenes
- Load a specific cutscene
- Reduce cooldown time, increase power-ups
- Cheat to allow cycling through any tutorial messages that can be found in a given level. It’s very frustrating when a message pops up randomly, you correct a mistake and then you are unable to trigger it again
- Access secret-levels and/or alternate story-sequences if any

Simulation games

- Add money (and substract money)
- Add time (and substract time)
- Add resources

Hidden Object games

- Show all items to be found, either highlighted in boxes, or with sparkles – We do however still want the option to click one item at a time in order to check for singular/plural strings
- Force all items, instead of just a selection – It’s common to have levels with 20+ objects while only 10 or so are displayed. We want to be sure we’ve seen all the possible items that could appear in that level
- Find all items and win level cheat
- Unlimited hints without a cooldown
- The fast forward skip button

Word games

- Reload/shuffle puzzle to completely refresh it with new content

2.9 Requirements

Depending on the game, some games will need some extra information to get localization started. These are typical things to put into a document in the localization pack. If any of the issues described below occur, please provide information on how to tackle the problem.

2.9.1 General

- UPPERCASE or lowercase text required for some words? Some languages have certain words always written in uppercase, while other languages have words only written in lowercase
- No special characters allowed in certain text strings?
- Objects / items / themes aren't always well known in all countries. Certain specific food, brands or traffic signs might not exist everywhere

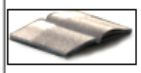
2.9.2 Word games

- No dynamic letter distribution. Like in English, not all characters are evenly distributed. The letter distribution differs in each level. Please make sure this is either dynamically determined by the game, or make sure that there is a configuration file where we can change the letter distribution per language. If this can not be changed, the characters will be distributed according to the English language and will not be correctly balanced
- Support accented or flattened characters. It is not always possible or logical to allow users to input words with accented characters. A mechanism to solve this is to make use of “display words” or “look-up words”. In this scenario all words do contain accented characters inside the dictionary files. The user, however, will put in words without accents. The game will look up the word and matches it with the most likely accented word inside the dictionary file. That word will eventually be displayed. Be aware that there are a few combined characters instead of accented characters, like Æ and ß
- Word list (encryption) tools. Please make sure to include instructions and (command line) tools so that we can create localized versions

2.9.3 Hidden Object (I-spy)

- Object reference list. We need some sort of visual reference for our translators to localize the object names. We would really appreciate it if you could provide such a list in HTML or XLS format. We will always try to create one on our end based on the language file and the available (image) resource files, by creating a small script to loop through the resource files and to output an HTML file

beachhouse *(scene_beachhouse.txt)*

Sailboat **OBJECT: obj23** *ITEMID: sc_beachhouse_boat*	
Book **OBJECT: obj47** *ITEMID: sc_beachhouse_book*	
Drink Mixer **OBJECT: obj78** *ITEMID: sc_beachhouse_cocktailshaker*	
Coffee Mug **OBJECT: obj69** *ITEMID: sc_beachhouse_coffeecup*	

- Stackable objects with multiple meanings. When there seems to be grouping / plurals of certain hidden objects, this causes two problems. 1: For some objects the same word is used ("Bat" can be a baseball bat or an animal), but this doesn't work in other languages. 2: Sometimes the plural version of a word isn't stored separately, but an extra "s" is (hard-coded) added, which doesn't work in other languages due to other grammar rules
- Unknown or limited space for object names. The length of translated objects will be different. On average, translations require more space. What happens when it goes out of the bounds of the list? Is there a maximum word length?
- Objects that refer to text, for example when the player has to look for “a cat” can refer to the text “cat” written on a wall or piece of paper
- Object images that contain text and require localization require extra attention. The image with a poison bottle works just as well if the word poison was left out.

2.9.4 Simulation

- Many different currency occurrences. There are various currency symbols scattered throughout the game, also on animated objects (like coins). Is it possible to edit all of them in a generic way?

2.9.5 Adventure

- When an adventure has multiple storylines, and/or different endings, a flowchart would be a great way to illustrate where certain choices in the game are made to affect the story. This reduces linguistic and QA testing times

3 Super cool features

We sometimes come across some features that developers have built in to help us localize the game more easily. These aren't requirements, but they were so helpful that we'll list them here in case you want to go the extra mile to help us.

- In-game text-label adjustment. Text overlap issues are the most common issues that we encounter. There are several ways to fix or workaround this. A really cool feature to have is to have the ability to see and change the boundaries (or font size) of labels in-game, by activating a certain mode with cheats.
- Auto-scaling text. A very interesting feature we've seen, are "clever" buttons or labels that will automatically reduce the font size when the texts will exceed the boundaries. This is a really nice way to help avoid text overlap issues.
- Auto-scaling dialogs. It is preferred to have text bubbles that expand automatically to fit the text. If text bubbles are fixed in size then it is important to provide 30% to 50% extra space for translations.
- In-game reload of strings file. When we make adjustments to a text file, it's great to instantly reload text files while you're in the game so that you won’t have to restart the game and go to the same scene again to check.

4 Checklist

Text	Yes / No / Comment / Details
Are all game texts in a single language file? (and NO translation tools)? Is there a character limit for certain words?	
Does text still fit if it becomes 30% larger after translation?	
Is text adjustable or scalable to make longer translations fit?	
No linguistic data in application data or savegame folder? (E.g. hidden object names)	
Are there word mini-games? If so, are there textual restrictions?	
No hard-coded text in source code?	
No stitching of words in the language file? If so, what is stitched?	
Images	
Layered PSD files for Images with text?	
Accented characters (Latin-1) in ALL fonts, including TTF files for PSDs?	
Misc.	
Cheats that allow moving through the game quickly?	
Branding screens configurable and not packed/encrypted in data file?	
All tools included (packing/encrypting/building)	
Are regional settings configurable and not hard-coded? (Currency, thousand seperator, etc)	
No files that aren't needed in the game folder? (e.g. thumbs.db)	
Has the game already been localized in some languages?	
Is there an exclusivity period for already localized languages?If so, how long and when does it expire?	
Is there a localization manual document?	

5 Reference list

ISO/IEC 8859-1 (Latin-1) character encoding
http://en.wikipedia.org/wiki/ISO/IEC_8859-1

Cyrillic alphabet
http://en.wikipedia.org/wiki/Cyrillic_alphabet

UTF-8 character encoding
http://en.wikipedia.org/wiki/UTF-8

LUA programming language
http://www.lua.org/

XML standard
http://www.w3.org/TR/2008/REC-xml-20081126/

Photoshop paragraph text
http://medialab.com/sitegrinder/support/paragraphvspointtext.html