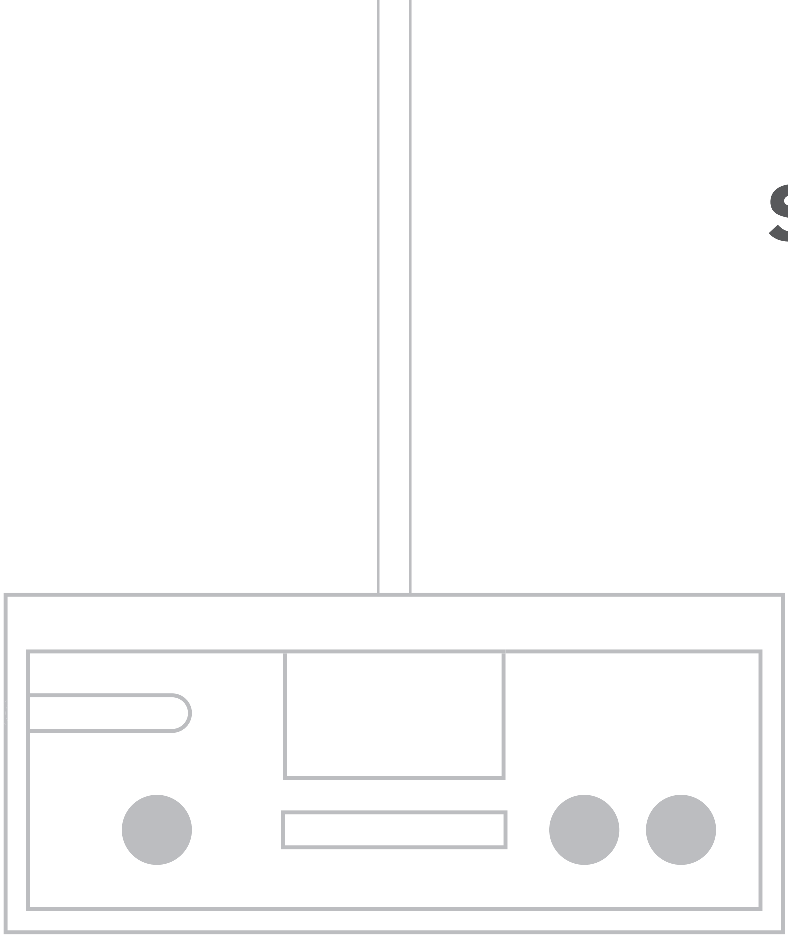

Getting Started with Gaming Localization

18 Tips

for Game Developers and Game Publishers

By Loek van Kooten

1 Choose your translation provider with care

You deserve the best quality. So do your users. How do you ensure that you obtain excellent quality? By focusing on more than just translation costs. Awarding your project to the lowest bidder is a recipe for a major gaming localization fail.

Focus on a high-quality UX first and foremost

Have you ever played a game that had a typo or some other error? Chances are that it ruined your experience as a user. On the other hand, a flawless, high-quality experience makes people want to keep playing a game and can make it engaging and even addictive.

Spelling and grammar mistakes are common in low-quality translations, but so are actual errors in meaning. In general, the lower the rate of translation, the higher the risk of poor quality. If there's no budget for proper localization, allocate the budget you do have toward something else that will benefit your company more.

Always use humans, whether they are freelancers or agencies

As a buyer of localization services, you have many choices, but the best option is always professional localization by human beings. Should you work with freelancers or with agencies? It depends on many factors. Per-word costs with agencies are typically 20%-100% more than what you would pay by going directly to freelance translators.

However, if you don't have time to recruit, test, and select translators yourself, or if you are translating a large amount of content into multiple languages and need someone to manage the project for you, an agency may be a good option to consider.

"Awarding your project to the lowest bidder is a recipe for a major gaming localization fail."

If you work with an agency, you will need to add 20%-100% to standard translation rates. Ask the agency how much they are paying their translators. Ask if you can speak to one of their translators. If you use an agency that charges 0.20 euro per word, but only pays their translators 0.01 euro per word, you are paying outside of the standard rates – consider this a red flag for quality.

2 Set realistic deadlines for your translation projects

A translator for a gaming localization project can deliver about 2,000 words per day on average, depending on the content type. Translation is a much more complex and time-consuming process than writing in one's native language. A writer only has to worry about the process of creation, not the process of conversion.

So, the first thing you should ask yourself is this: "How long did it actually take us to write this manual?" Chances are, translating it will take at least as much time.

Let's look at the possibility of translating a 100,000-word manual in three days. To translate such a manual in three days, we need to reserve two days for translation and one day for proofreading.

That leaves two days for 100,000 words, which equates to 50,000 words per day. To translate 50,000 words in a single day, you need to produce at least 2,000 words per day per translator, which means you would need 25 translators.

Avoid using too many translators on the same project

Imagine having 25 people writing your manual simultaneously. The result would be an inconsistent end product with 25 different writing styles. The same applies to translation. If you wanted to carry out proofreading on the third day, this would merely mean checking grammar and spelling. To actually unify 25 different writing styles, you need one extra person (not more, as that would still result in multiple styles). It will take at least three extra working days to have one person unify 100,000 words. When you need this "unifier" to start working to make your deadline, the translators haven't even started working yet.

In other words, translating a 100,000-word manual in three days is nearly impossible to do with any sort of quality guarantee. Be wary of any translation agency that says they can cater to your needs anyway and still deliver a good translation.

"How long did it take you to write the manual? Chances are, translating it will take at least as much time."

Plan ahead to prevent a last-minute translation time crunch

How do you prevent getting into a precarious situation with impossible deadlines? Try to plan things in advance by providing translators pieces of text as soon as they become available. Don't assume that you need to wait until you provide a final draft. Even if some of these pieces change later on, modern translation tools, also known as computer-assisted translation (CAT) tools, should be able to tell translators exactly what changed where. This ensures that even if you later provide the final text with updates, you'll never pay twice for the very same sentence.

Also, time your project wisely. The months of April and May are extremely busy when it comes to game translations. Most game translators are fully booked during these seasons. Try to have your texts translated earlier or later if you want to avoid working with less experienced translators who are not specialized in games, and who therefore may offer inferior quality.

3 Use XML or separate text files and avoid embedding text directly into the code

Most professional translation tools offer support for languages like *C++* and can directly extract translatable text from several kinds of code. It is not generally a good idea to directly embed your text into your scripts. For a translator, it doesn't matter whether the text is displayed when John Doe enters the room or if it should scroll from left to right only after the player has found the *BFG*. Embedding text directly into the code instead of keeping text in separate files will make your life harder for several reasons.

First, the number of words counted will increase tremendously (often up to twice the actual amount) if there are no filters available that can separate the text from the code. Why should you pay more? Reduce your translation cost significantly by ensuring this separation of text from code. Second, having text embedded in code will slow down the localization process tremendously, as the translator needs to actually read the code to determine which segments need translation and which ones do not. Third, the consistency of the translation will be difficult if not impossible to maintain.

"Embedding text into code can slow down the localization process and make it more costly than necessary."

Take the following example:

```
show_data(samples, countone, "Original values");
show_data(samples, counttwo, "Original values");
```

The translator's translation tool will count these as two separate strings. If there are thousands of other strings between the above-mentioned two strings, you can be sure the translator will have forgotten how she or he translated them to begin with.

Here is an alternative:

```
Original values
Original values
```

Had the code looked like this, the translation tool would have automatically propagated the first instance to the second instance. That saves you money, saves time, and leads to greater consistency and translation quality. So, use separate text files, or even better yet, use XML.

4 Remember that not every language has different grammatical structures

Programmers like efficiency, for good reasons. Why code this...

```
Brian takes the gun
Brian drops the gun
```

...if you can use this instead?

```
Brian %s1% the gun
takes
drops
```

Hard-coding around one language's grammar is a really bad idea. Using Dutch as an example (the bold words are equivalents), the translation of the full sentences is:

```
Brian pakt het pistool
Brian laat het pistool vallen
```

How will we fit the Dutch translation ***laat...vallen*** in %s1%? It's not possible.

Drop in Dutch becomes *laat vallen* (literally: *let fall*) and the verb clause is split around the object of the sentence (the gun). This is a typically Dutch phenomenon. But there are many languages with similar challenges. The Japanese, for example, can put adjectives in a past or future tense. So if you say *this is gad*, it could mean something like *this was good*. Every language has its own interesting phenomena, which developers simply cannot take into account when constructing "concatenated" sentences like these.

Avoid concatenated sentences when writing code

If you need to construct concatenated sentences, you should consult your translators in advance. Once you have started localizing your games regularly, you should have a rough idea of the languages you normally localize into. So use your team and ask your translators for help. A translator will not usually charge you for some simple advice on these matters.

The best solution? Avoid concatenated constructions entirely. If you do include them, make very clear which sentences can be combined with which sentences (all possible combinations) and be prepared for some unpleasant surprises that require some changes to the code.

Using the concatenated approach might result in fewer words, but it does not lead to less expensive or faster localization. The number of words saved will often amount to a very tiny number – not significant enough to result in any real cost savings. On top of that, the translator will need to really slow down when encountering segments like these, trying to think of workarounds and solutions that work in the target language. This means that the translator will have fewer linguistic choices available to provide the best quality.

The most important thing to remember is this: Never glue different linguistic segments together in your code without first consulting your translators.

5 Remember that not every language takes up the same amount of space

Consider the following example:

Up (maximum string length: two characters)

The word for *up* in Dutch is *omhoog*. How are we going to fit this into two characters? There is no way the word "omhoog" can ever be abbreviated to two characters. *Om, Oh,* whatever you try, no one will ever understand the original meaning. In a best-case scenario, you need at least four characters here: *omh*. Even then, the result does not sound pretty in Dutch.

Let's reverse the situation

Try to come up with a two-character translation in English for the Dutch word *ar,* which means *sleigh*. *Sl* is probably not going to cut it, is it?

To ensure that your translation can display properly, you need to allow at least 50% more space for your translations (as you can see, in this particular example even 50% is not enough). Even better: Use a dynamic user interface that adjusts on the fly (i.e., don't hard-code it) so no unnecessary sacrifices need to be made. It is more work, but it will save you lots of time during the localization process itself.

6 Remember that every word has many possible translations

Never, ever cut corners by copying and pasting previous translations from your translator. This can have absolutely disastrous results.

I once did a translation about a spaceship simulator–in this context space is translated as *ruimte* (as in universe). The client copied and pasted the word *ruimte* to a message popping up every three seconds, instructing the user to press *SPACE* (as in space bar) to jump, without consulting me beforehand. The result was that the user was instructed to press the universe every so many seconds.

The end result, reminiscent of Zero Wing (All Your Base Are Belong To Us), was run into the ground by all review sites.

Instruct your programmers to **never** touch translations if they don't speak the target language and **always** consult your translator.

The second way this can go wrong is by having only one instance for a string called *space*, while using it in two different contexts. The first context is a screen on which players can input their names.

In this case, *space* refers to the space bar (Dutch: *spatie*). The second context is the title of the first level, which is set in space (Dutch: *de ruimte*).

Of course, there is no way the two different translations can be put in the very same string. So you need two different strings here, for two different contexts, even though in English, the source text of the string is the same.

7 Provide context and answer questions from your translator

It's often a very good idea to provide context for as many strings as possible. Consider the following:

Great, you did it!

I can't believe that worked.

Run, run, before they get us!

I'm glad we made it.

These are some random strings taken from a chase scene. And I'm sure you interpreted *I'm glad we made it* as *I'm glad we didn't get caught.* However, in this particular case it meant *I'm glad we made this bomb instead of them; at least we know how to make a good bomb.*

The two different meanings lead to two very different translations. This situation, known by translators as "ambiguity," happens more often than you think. For a good translation, the source text should be unequivocal. This isn't always possible, so to prevent any problems, it's always a good idea to provide context. Some developers do this by explaining what the string is about in a separate comment.

Translation software can warn the translator of possible pitfalls like these, in which case a good translator will send you some questions. But if both the software and the translator overlook these kinds of pitfalls, the result will be an incorrect translation. An extra explanation about context or actually playtesting the game are great ways to get rid of these kinds of context-sensitive errors. Do you think playtesting is normal? The vast majority of developers never playtest localized versions of their games. It is therefore no wonder that so many localized games receive poor reviews on review sites.

Some translation agencies have your games playtested by foreign exchange students they have never met or interviewed before. Your game too might have been tested by an inexperienced user who knows everything about biotechnology, but nothing about games or linguistics. Make sure to ask your translation agency about their testing process before you work with them.

Also, in dialogues, it is extremely important that you indicate who says what to whom. This is because in other languages, there are different levels of politeness. In English, everybody calls everybody *you*, but in Dutch, we use *u* toward superiors and elderly people and *jij* toward friends or younger people. Japanese holds the absolute record when it comes to politeness levels. Verbs, grammar, pronouns, everything changes:

Genki? (How are you?, blunt style)
Gokigen wa ikaga de gozaimasu ka? (How are you?, a bit more polite)

In Japanese, there are also phenomena like male and female languages, each with its own grammar.

8 Avoid using hard enters in the middle of strings

Japanese developers and developers of mobile phone games love doing this.

Many translators use computer-assisted translation tools for their translations. This software automatically copies repetitions in the text, enabling the translator to give you a discount on similar strings, making sure the style of the translation is consistent and greatly speeding up delivery. However, the smallest building blocks most tools can work with are full segments.

For example:

The
sword dropped

The
axe dropped

In Dutch, this becomes:

Het
zwaard viel

De
bijl viel

As you can see, automatically copying the string ***the***, which occurs twice, is never going to work here, as Dutch uses different articles for *sword* and *axe*. The same applies to longer segments.

A translated segment can only be copied reliably if the segment is complete. If a segment is split into several parts, you will see that these depend on each other, meaning that the translation of a certain part of a segment can change depending on what comes before or after it.

Even if *English (A+B)=Dutch (a+b)*, *English (A+C)* is not necessarily *Dutch (a+c)*. It often is *Dutch (a'+c)*.

Since it's impossible to determine beforehand which segments are split in the middle and which are not, even the occasional use of hard enters in your text can render translators' tools totally useless.

What does this mean in practice? On average, localization of soft-coded strings is about 12% less expensive than localization of hard-coded strings. For every language you localize into, you can save money and speed up the process.

By the way, there's a linguistic reason why Japanese developers love hard-coding their strings. In Japan, it's customary to wrap text by hand. Japanese has no spaces between words, and it wouldn't be user-friendly to have the text automatically wrapped in the middle of words. Languages like English and Dutch do feature spaces, and can therefore automatically be wrapped on the fly.

Because of this, and the Japanese developers mostly not wanting to code a separate word-wrapping routine for languages with spaces (which is not that hard; a language like PHP takes just one extra line in your code), translators instead need to spend hours and hours on wrapping the text by hand, line by line, string by string, which is very counterproductive. You will have to pay more because the localization process will take more time and manual labor.

To speed up the process, I wrote my own PHP software, which extracts the strings from Excel, unwraps them, and after translation, rewraps them back into Excel. However, even then, hard-coded strings are still much harder to translate. Also, not all your translators can write their own software.

Suppose a string is spread over several lines and one word has to be changed: The change will make certain lines longer and shorter, so that all lines need to be rearranged and rewrapped.

Before change:

This string is not

wrapped on the fly.

Its maximum width is

20 characters.

After change:

No, no, this string	*<= two extra words, line changed*
is not wrapped on	*<= no word changes, but line changed nonetheless*
the fly. Its maximum	*<= no word changes, but line changed nonetheless*
width is 20	*<= no word changes, but line changed nonetheless*
characters.	*<= extra line!*

So, if you want to speed up your translation process and make it less expensive, repeat this mantra:

Thou shalt not hard-code! Thou shalt not hard-code!

9 Remember that translation, like programming, is a profession

At birthday parties, people often ask me what I do for a living. When they find out I'm a translator, they mostly reply that their English is absolutely perfect and that they can't understand that I make a living from doing something so "easy." It is very tempting to have a translation done by someone you know well, who claims that he or she will be able to deliver a perfect product.

Don't fall for it. Outsourcing translations to non-translators is as bad as outsourcing programming to non-programmers. Just like someone might dabble in coding, people might dabble in language, but that does not make them a professional.

Take a look at your own user forums. Do a quick grammar and spelling check. Would you trust your users to write on behalf of your company? They might speak the language perfectly, but it does not mean you can trust them to represent your brand.

Quite often, game publishers and developers have their own websites translated (partially) by internal staff, often non-linguists. The fact that someone speaks another language does not mean he or she is actually able to come up with a good translation in that language.

Let the programmers program, let the marketers market, and let the translators translate.

10 Keep the translator informed of development updates

This problem can lead to all kinds of disasters. Let me give just one example. Recently I spent two full days on playtesting a high-profile game. The goal was to catch any strings that didn't fit in dialogue boxes. After two full days were spent, the developer mentioned that a new, lower resolution had been implemented into the game, resulting in a bigger font with fewer characters fitting in a dialogue box. The developer might as well have poured the 500 euro spent on playtesting down the drain, as the whole game needed to be retested from scratch in the lower resolution.

11 Recognize that numbers, units, dates, and times require localization

Aside from the fact that Europe uses a metric system, as opposed to the non-metric system used in the UK and the U.S., we also write numbers in a different way.

The numbers *3.0*, *1,000*, and *10,000* are rendered respectively as *3,0*, *1000*, and *10.000* in Dutch. So yes, numbers need to be localized too.

And if your game uses units like *mph* (racing games) or *lbs*, you will need to implement a unit conversion routine that converts these to respectively *km/h* and *kg*. This applies to any non-metric units used in your game.

Many countries also use different formats for dates and times. In Dutch, for example, we use the *DD-MM-YYYY* format for dates and the 24-hour *HH:MM* format for times (AM and PM are not used). Even the symbols used in these formats (- and *:*, respectively) can change.

12 Refrain from using automatic punctuation handling

One of the translations I localized had to be entered via a web interface. The punctuation was handled automatically. Periods at the end of every sentence were placed automatically, so that inserting a full stop in the actual translation would eventually result in sentences ending with two full stops, *like this..*

Once I understood what was going on, this was easily solved, but even then, some of the Dutch sentences ended up with weird punctuation.

Even such a simple thing as a comma is used differently in every language. In English, this sentence looks perfectly normal. In Dutch, however, we'd never put a comma behind "English." Also, *¿did you ever consider that Spanish phrases questions like this?* I will spare you the details about Japanese, a language that doesn't use spaces *andiswrittenlikethis*.

13 Use the same translation providers repeatedly to boost consistency and reduce costs

If you switch translators or agencies with every single project, your product will use different terms and writing styles. Also, if you do not stick with consistent translation providers, you lose a great deal of money because you can't leverage legacy translations.

There are alternate ways to ensure consistency, but these often cost more than simply sticking with the same agency or translator in the first place.

It's often very tempting to go to another agency because your previous agency cannot deliver fast enough or because you have found someone else who is cheaper. And of course, sometimes there is no other possibility. However, think twice before switching providers, because you may end up taking a hit on quality and costs in the process.

Many agencies do not have a policy to ensure consistency. This means that the same project often ends up at five to six different agencies or translators.

Programmer A calls agency A, programmer B calls agency B, the secretary calls agency C, and the development manager uses the services of freelancer D. Imagine the chaos.

Some agencies give you the impression that your translation is handled by only one translator, while very often, this is not the case. They often use multiple translators (or even other agencies) for one project without ever telling you, once more resulting in a less consistent, lower-quality translation.

If you work with an agency, ask for a guarantee that it will use the same translators repeatedly.

14 Make sure to take advantage of "translation memory"

Computer-assisted translation tools are tools that store every sentence a translator has ever translated for you in what is called a "translation memory" file. This can be quite useful, so that when you update a manual, re-used sentences are translated automatically.

This is a secret that you can use to get ahead of your competition. Even the biggest game translation agencies in the world almost never use computer-assisted translation tools. Out of eight game translation agencies I have been working with on a full-time basis for 12 years, only one has asked me once if I used these tools. Indeed, typing translations manually in Excel still seems to be the de facto way to localize games for most companies.

How many new words are there in your latest update? Developers try to come up with the right answer by color-coding every new segment while adding it. Agencies try to find the answer by comparing all sentences one by one (charging you for every minute spent). With the right tools, however, you can calculate the answer (and hence your discount) within just a matter of a few minutes.

It always amazes me how agencies ask translators without computer-assisted translation tools to use the same terminology as in previous translations. There is no way you can guarantee that without using these tools. You cannot possibly expect people to look up every single noun in existing translations using *Edit > Search* in Excel to see how said word has been translated before (and remember it while translating the rest of the text!). To ensure 100% consistency, you'd need to repeat this procedure about 2,000 times for a simple 25-page manual (a 25-page manual has about 2,000 unique potential terms).

So what often happens is that translators just make a wild guess, look up what they think could have been translated before, and just get on with it. Meanwhile, the agency tells you your translation is 100% consistent with previous translations. Indeed, it's what they asked their translators to ensure, but translators without computer-assisted translation tools have no choice but to ignore these instructions and play dumb. After all, being honest means they won't get the job, which will eventually go to someone less honest, anyway. By the time you get complaints, they have already moved on to the next project for another client.

15 Just say "no" to ASCII

This isn't really an issue for localizations to Dutch, but it can have tremendous consequences if you ever decide to localize your game to say Japanese or Chinese. Japanese uses about 8,000 characters (of which about 2,000 are used regularly); Chinese uses 40,000 (of which about 8,000 are used regularly). As you can see, the one-byte ASCII set is by no means sufficient to address all characters used.

ASCII routines should be totally forbidden and replaced by Unicode routines, even if your game is only in English. You never know what you will do in the future, and Unicode offers support for all languages. Of course, you will still need to integrate different font sets for less common languages, but that's a lot easier than rewriting your code from scratch.

16 Let players control text speed

Not all players read as fast as you do. It is very frustrating to see dialogue texts disappear while you haven't even finished reading them.

This becomes an even more important issue if you have your game translated, as translations often become longer than the original source text. So, let people control text speed themselves and implement a button the player must press to move to the next dialogue.

And if this is not possible, for example, in cutscenes, either use voice-overs or err on the side of caution and have the reading speed tested by normal people (not game testers, who read all day long and can read much faster than normal people).

17 Make sure your fonts are legible in other languages

Not all players have an HDTV. Do what professional studios do: Mix the music on the worst loudspeakers you can find (often transistor radios). If music sounds good on those speakers, it will sound good everywhere. So grab that old, dusty CRT from the attic. If that monitor displays the text neatly, the text will be legible no matter what system your players use.

18 Make translators want to do business with you by paying them on time

EU law says that standard payment terms are 30 days net. That means 30 days net, not 31 days net, not 41 days net, and certainly not 51 days net. Ask yourself how long it would take you to quit your job if your boss suddenly decided to pay your wages one month later just because that's his new policy.

Many translators have dozens if not hundreds of clients. If you decide to pay your translators 30 days late, they will direct attention to other clients that actually pay on time. The best translators are always in high demand; make sure that you treat them respectfully and professionally so that they will want to work with you not just on a single project, but in the long term.