

When software structure meets translation: material mediation in English-Chinese game localisation

Bao Yunlong

baoyunlong19990128@163.com

Universiti Sains Malaysia

Article

Keywords: Chinese, corpus-based Translation Studies, game localisation, material mediation, software localization

Posted Date: September 9th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10707359/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Abstract

Video game translation depends on both word choice and the software files that assemble text for players. These files can contain hidden instructions that insert changing values, apply colour or formatting, and identify keyboard input. Although research has described technical constraints and cultural adaptation, corpus-scale evidence about the relation between these instructions and target-language wording remains limited. This study asks which file-encoded functions remain stable across English and Chinese resources and how Chinese grammar and interface wording reorganise their visible expression. The dataset contains 60,685 aligned records from *OpenTTD*, *The Battle for Wesnoth* and *0 A.D.* Language-version branches were analysed separately, documented contamination was corrected, and strings were classified by communicative function. A technical marker was defined as an embedded instruction such as a variable, formatting command or input cue. Marker sequences were compared alongside close English-Chinese analysis. Two coders independently classified a stratified sample of 450 records; agreement was assessed and disagreements were adjudicated. All 708 records with an omitted marker were also checked individually in the relevant game builds. After 140 records containing English-specific grammatical instructions were excluded, 7,108 of 7,818 marker-bearing records (90.92%) retained either the same sequence or the same ordered functional pattern. Ordered marker correspondence was high, while selected cases showed Chinese reformulation of interface actions, grammar and naming. Together, the results document how software structure can remain continuous when visible language changes. They support material mediation, understood here as the way software files organise the connection between executable functions and translated wording, and distinguish textual comparison, human coding and build testing as separate forms of evidence.

Introduction

When a game displays changing information, the visible sentence is often assembled from ordinary words and hidden software instructions. A passenger-count string in *OpenTTD* provides a simple example. The English resource {COMMA}{NBSP}passenger{P "" s} contains three instructions: {COMMA} inserts punctuation, {NBSP} prevents a line break, and {P "" s} selects the English plural ending when required. Its Chinese counterpart, {COMMA}{NBSP}乘客 (wèi lǚkè, 'classifier + passenger[s]'), keeps the punctuation and spacing instructions but omits the English plural selector. Chinese expresses the quantity through a numeral-classifier construction and a noun that does not change for number. The same software value can therefore remain available even when the grammar around it changes.

This relation is common in game localisation, the process of adapting a game's language and associated content for another linguistic and cultural setting. Games combine software, images, sound, interface design and player action, so translated wording must function inside an executable and multimodal product (Mangiron and O'Hagan 2006; O'Hagan and Mangiron 2013; O'Hagan 2015). Player-facing language is usually stored in localisation files: structured resources that contain short text segments together with identifiers, variables, markup and formatting commands. An identifier links a segment to a location or function in the software, while a variable inserts a value when the game runs.

These structures help determine how text is divided, processed and displayed (Cronin 2013; Littau 2016; Olohan 2021).

Game-localisation research has examined creative adaptation, production workflows, cultural references, player reception and language-specific problems. Corpus studies have also identified recurring linguistic patterns. Less is known about what happens inside localisation files when the software function remains stable but the visible wording changes. The problem is partly methodological because three forms of evidence answer different questions. Comparing technical markers, meaning code-like instructions embedded in a string, identifies formal similarities and differences. Functional coding classifies what a string does, such as labelling a button or giving an instruction. Build checking tests the record in a running version of the game. Keeping these evidence types separate makes the resulting claims easier to evaluate.

This study examines those relations in three open-source games: *OpenTTD*, *The Battle for Wesnoth* and *0 A.D.*. The term material mediation refers to the way a software file organises the connection between an executable function and the words used to present it. This use draws on research that treats technologies, media and practices as constitutive parts of translation (Cronin 2013; Littau 2016; O'Hagan 2015; Olohan 2021). The corpus contains 60,685 aligned English-Chinese records, each pairing a source string with its corresponding target string and technical context. The repository tags zh and zh_CN form the primary analytical group, while zh_TW is examined separately. Two coders independently classified a stratified sample of 450 records; agreement was assessed and disagreements were adjudicated. All 708 records classified as marker omissions were checked individually in the relevant game builds.

The analysis addresses two research questions:

1. Which software instructions embedded in English strings are retained, changed or omitted in the corresponding Chinese resources?
2. How does Chinese grammar and interface wording express the same game functions when the visible form or embedded instruction changes?

The study connects game-localisation research, material approaches to translation and corpus analysis. It provides a corpus-scale account of technical-marker correspondence, a human-validated description of what software strings do, and close analysis of English-Chinese grammatical and interface reformulation. Its central argument is that localisation files participate in translation by maintaining some executable relations while allowing the target language to reshape their visible expression.

Literature review

Game localisation as translation within software production

Localisation scholarship developed from the practical problem of adapting software for distribution across markets. Early accounts described internationalisation, which prepares software for several languages, as well as resource preparation, interface constraints, testing and coordination between translation and software development (Esselink 2000; Dunne 2006). Bowker (2002) examined computer-assisted translation technologies in translators' work, while Pym (2004) placed localisation within the wider movement and distribution of text. Together, these approaches shifted attention from the translated sentence to the systems through which text is segmented, processed and delivered. Industry guidance continued this emphasis by describing game localisation as a workflow of asset preparation, translation, integration and quality assurance (Chandler and Deming 2012).

Game-localisation research extended this production perspective. Games combine software localisation with audiovisual translation, which studies translated language in relation to sound and images, and with cultural adaptation. Dietz (2006) identified problems specific to computer games. Mangiron and O'Hagan (2006) then described the tension between technical restrictions and the creative work needed to preserve player experience. O'Hagan and Mangiron (2013) developed this account of game localisation as a technology-based translation practice, while Bernal-Merino (2015) connected its linguistic, industrial and entertainment dimensions. These studies share a central premise: game text is translated within a production system whose technical organisation shapes the available linguistic choices.

Later reviews show that this premise has generated several research directions. Mangiron (2017, 2018) maps research on methods, reception, quality, technology, collaboration and training. Pirrone and D'Ulizia (2024) likewise connect linguistic and cultural problems to software processes, while Toftedahl's (2025) studio research shows how localisation competes with other development tasks. These studies establish technical conditions as part of localisation practice. They offer less precision about how those conditions can be traced inside aligned resources.

Production research and file-level research answer different questions. Studies of workflows explain how localisation is planned, divided and tested. Aligned files show how a source string and its translation remain connected to values, commands and identifiers. The localisation file is therefore both a production resource and evidence of how language is linked to software functions.

Linguistic and cultural adaptation across game-localisation contexts

The linguistic branch of game-localisation research has concentrated on strategies, textual functions and culturally specific material. Fernández-Costales (2012) relates translation strategies to game genres and textual typologies. Pettini (2022) distinguishes *realia*, meaning references to the real world, from *irrealia*, meaning names and institutions created for fictional worlds. This distinction is useful when historical references coexist with fictional names and institutions. A recurring fantasy name may be

internally consistent within a project, while a historical form requires external evidence before it can be presented as a recognised convention.

Research in particular language markets shows why technical and cultural issues need to be analysed together. Jooyaeian and Khoshsaligheh (2022) relate Persian translation solutions to Iran's cultural and industrial setting. Arabic studies examine script and variables, terminology and cultural content (Al-Batineh 2021); adaptation across verbal, visual and interactive layers (Al-Ajarmeh and Al-Adwan 2022); localisation levels and player preferences (Al-Batineh and Alawneh 2022); interface and caption strategies (Jarrah et al. 2023); and terminological, syntactic, orthographic and interface problems (Shehab and Odeh 2025). Their shared implication is language-pair specific: scripts, grammatical systems and cultural expectations shape how software resources are realised.

The same point applies to English-Chinese data, although the relevant grammatical resources differ. English plural marking may appear as inflection or as a selector encoded in the localisation file. Chinese commonly expresses quantity through numerals, classifiers and nouns without obligatory number inflection. An absent English plural macro may therefore correspond to a grammatically complete Chinese construction. Treating marker absence as a self-evident translation error would collapse formal correspondence and linguistic function. The appropriate analysis identifies the marker's software role, examines how the target construction expresses the relevant relation and then uses build evidence to determine the technical outcome.

Research on naming and automation makes the same evidential distinction. Moreno García and Mangiron (2024) treat GPT-4 outputs for Pokémon names as candidate resources whose accuracy, context and cultural appropriateness still require evaluation. Corpus recurrence can establish a name's distribution within one project; claims about wider convention require external sources.

Comparative corpus research also cautions against assuming that localisation necessarily simplifies language. Fan et al. (2026) compare lexical and syntactic complexity across three games and report patterns of complexification as well as stability. Their findings challenge a general simplification account and show that linguistic outcomes depend on the games, languages and measures selected. Character ratios in English-Chinese files can locate compact labels or unusually long records, but they cannot establish simplification, cognitive economy or translation quality on their own. Functional and grammatical analysis is required to explain what a numerical difference represents.

How words, media and software work together

Multimodality describes communication in which meaning is distributed across words, images, sound and other modes (Kress 2010). Audiovisual-translation research likewise relates linguistic choices to production, reception and non-verbal resources (Pérez-González 2014; Guillot et al. 2019). Games add interface position and player action. A localisation file cannot reproduce this whole experience, but it preserves technical traces that link words to changing values, commands and formatting.

Software-localisation studies supply the practical basis for reading those traces. Resource structures, technical preparation and quality processes form part of localisation (Esselink 2000; Dunne 2006). Jiménez-Crespo (2013) connects digital text types and usability to web-localisation conditions, while Pym (2004) theorises localisation through text distribution. These works legitimate segmentation and resource structure as research objects, although their principal scale is the process or digital product as a whole.

Material approaches provide the conceptual bridge to a file-level account. Cronin (2013) situates translation within digital infrastructures. Littau (2016) places media, objects and machines within the study of translated communication. Olohan (2021) examines relations among practices and material arrangements, and O'Hagan (2015) analyses game localisation as a software-mediated cultural experience. Across these accounts, technology participates in the organisation of translation, communication and access.

Material mediation brings this argument to the level of an individual localisation record. An identifier links a string to a software function. A variable links visible wording to a value inserted while the game runs. A formatting command controls presentation, and a grammatical selector links a software rule to a language-specific form. Repeated identifiers can also distribute the same term across several interface locations. These elements do not dictate one translation; they organise the conditions under which a translation becomes executable. Material mediation therefore describes how technical continuity can coexist with lexical, grammatical and functional change.

This account separates two uses of technical constraint in the literature. Production-oriented studies describe limits involving interfaces, schedules, assets and testing (Chandler and Deming 2012; O'Hagan and Mangiron 2013; Toftedahl 2025). Material approaches explain how media and technologies participate in translated communication (Cronin 2013; Littau 2016; Olohan 2021). File-level analysis connects these scales by identifying the technical relations preserved in the bilingual resource itself. A retained variable, for example, indicates continued access to a runtime value. It does not establish that the surrounding grammar or interface instruction is unchanged. The distinction permits formal stability and linguistic change to be analysed within the same record.

This formulation also clarifies the role of multimodal evidence. File paths and identifiers can often distinguish a button from a narrative sentence or associate a string with a control family. They do not reproduce screen layout, iconography, timing or player interaction. File analysis supports claims about encoded relations, while screen recordings and interaction data support claims about visual presentation and usability. Keeping these evidential levels separate strengthens the linguistic analysis because each conclusion remains tied to the material record that supports it.

Open-source localisation as collaborative, versioned work

An open-source repository is a publicly accessible, version-controlled collection of project files. Such repositories expose localisation as an artefact revised by different contributors over time. Research on collaborative translation examines collaboration as a translation mode (O'Brien 2011), fan translation and translation hacking as user-generated work (O'Hagan 2009), participant motivations in crowdsourcing (McDonough Dolmaya 2012), and the effects of online collaboration on technology, quality and translation practice (Jiménez-Crespo 2017). The resulting language files may therefore combine contributions from different periods and forms of participation.

Game localisation also circulates through fan communities, intermediary languages and market-specific language provision. Fans scrutinise localisation policies (O'Hagan 2017), and English can operate as a pivot language, meaning an intermediate version used to produce another translation (O'Hagan 2022). Minority-language provision may also conflict with market priorities (Fernández-Costales 2017). A locale branch is therefore both a technical category and a route through which a language variety becomes available. Its name provides no direct evidence of individual translator identity.

Mandiberg (2021) places such processes within localisation history by arguing that games have been remade through local technological and cultural conditions. This historical perspective challenges a model in which a finished global product is simply transferred into passive target markets. For repository research, the implication is methodological: locale labels, file formats, duplicates and versioned strings belong to the history of how the localised product was assembled. Retaining this provenance allows corpus patterns to be interpreted as properties of archived software resources. It also prevents repeated strings from being mistaken for repeated independent decisions when they may represent separate interface placements or inherited resources.

Corpus-based translation research and the remaining file-level gap

A corpus is a systematically assembled body of texts designed for analysis. Corpus-based Translation Studies uses such collections to move from isolated examples to recurring source-target patterns. Baker (1993) established this empirical programme, Laviosa (2002) consolidated its theoretical and applied strands, Hu (2016) surveys its development, and Zanettin (2012) focuses on corpora built specifically for translation research. Across these accounts, corpus construction is part of the argument because frequencies depend on which texts, units and metadata the corpus contains.

General corpus methodology makes this point more precise. Representativeness depends on a clearly stated population and sampling design (Biber 1993), and findings remain tied to methodological choices (McEnery and Hardie 2012). Tognini-Bonelli (2001) distinguishes corpus-based research, which begins with categories defined in advance, from corpus-driven research, which develops categories from patterns in the data. The present study is corpus-based: marker functions and textual domains are

defined before analysis, human coders validate interpretive categories, and explicitly specified rules extract marker sequences in the same way for every record.

Specialised corpora show how linguistic and contextual information can be combined. Bernardini and Ferraresi (2011) integrate comparable and parallel technical-language data with qualitative interpretation. Dixon (2024) builds a game corpus from localisation files and classifies texts by register and communicative context. Fan et al. (2026) test simplification through lexical and syntactic measures. These studies support purpose-built corpora and multiple measures, while leaving the relation between technical markers and target-language reformulation open to direct investigation.

The remaining gap concerns four connected levels of evidence. First, surface comparison records whether an embedded instruction is repeated, changed or absent. Second, technical classification identifies what that instruction does. Third, functional coding identifies the communicative role of the string, for example an interface label, tutorial instruction, narrative passage or term. Fourth, build checking tests whether a formally different record still works in the game. No single level answers all four questions. Translation Studies methods should align the research question, data, sampling and reliability procedure (Saldanha and O'Brien 2014); here, that principle requires formal comparison, human interpretation and runtime testing to remain separate and traceable.

This separation is especially important for English-Chinese comparison. Exact token matching can identify stable values and formatting, but it can misclassify English grammatical instructions that Chinese does not need. Functional coding can distinguish interface commands from narrative or terminological strings, but those categories require human judgement. Build checking can establish whether a formal omission affects the running game, but it cannot explain the Chinese construction that replaces the English form. Used in sequence, the three methods connect corpus-wide patterns with close analysis of selected source-target pairs.

The literature supports three propositions. Game localisation occurs within software production and communication across several modes. Linguistic and cultural adaptation varies with language structure, game function and circulation context. Corpus evidence becomes more explanatory when aligned strings remain connected to the technical fields that organise their use. Existing studies have established each proposition, but they have rarely combined them in a corpus-scale analysis of English-Chinese localisation files. This study addresses that junction through ordered marker comparison, validated functional coding and close analysis of grammatical and interface cases. It explains how executable relations can persist when their visible linguistic forms are reorganised.

Methods and analytical framework

Corpus, language-version groups and unit of analysis

The corpus contains localisation resources from three open-source strategy and simulation games. OpenTTD is a transport-management simulation with many menus, tooltips, cargo labels and values

inserted while the game runs. The Battle for Wesnoth is a turn-based fantasy strategy game whose files contain campaign dialogue, help text, units and abilities. 0 A.D. is a historical real-time strategy game with interfaces, technologies, structures, tutorials and historically framed names. The supplied full archive contains 80,642 aligned records. Its translated-only subset contains 60,701 records, and the final analysis core contains 60,685 English-Chinese pairs.

An aligned record contains an English string, its corresponding Chinese string, the project name, a language-version tag and any available information about the file of origin. The same wording is retained when it appears under different identifiers or in different files because each occurrence can serve a separate software function. Counts therefore describe implementation records: instances of text attached to particular software locations. A repeated wording under several identifiers contributes several records because each placement may perform a different function. English is the source field in the supplied dataset, although the production process may have involved inherited or pivot-language versions (O’Hagan 2017, 2022). The terms ‘source’ and ‘target’ refer to the aligned corpus fields; production direction remains unverified. This unit preserves the link among wording, identifier, file path, language version and technical markers. The resulting frequencies describe implementation placements in archived software resources. Counts of individual translators or independent translation decisions require a different unit of analysis (Biber 1993; Zanettin 2012).

A locale label is the repository tag used to distinguish a language version. The primary zh/zh_CN group contains 32,899 records, and the zh_TW comparison contains 27,786. These tags describe repository organisation and provide only limited sociolinguistic information. Table 1 reports the corpus composition. Wesnoth contributes 38,739 of the 60,685 records, so the aggregate figures are interpreted alongside project-specific results.

Table 1. Cleaned corpus by project and repository locale group

Project	zh/zh_CN analytical group	zh_TW sensitivity set	Total
OpenTTD	4,766	4,772	9,538
The Battle for Wesnoth	22,087	16,652	38,739
0 A.D.	6,046	6,362	12,408
Total	32,899	27,786	60,685

Note. Counts refer to aligned implementation records after contamination repair. Repository locale labels are retained as analytical variables.

Data cleaning, functional classification and validation

The cleaning workflow keeps the original target string beside every cleaned version. Ninety-six records in portable-object (PO) format contained metadata headers, including Project-Id-Version, Language and

Plural-Forms, before the Chinese translation. The header block was removed and the final translation line retained. One Wesnoth record contained a prompt-like continuation after an explicit end-of-text delimiter; the translation before the delimiter was retained and the continuation removed. No record was deleted from the 60,685-row analysis core. Unusual length ratios, repeated translations and English residue remained as inspection flags unless a contaminating span could be directly identified. Keeping the raw field allows every repair to be checked against the source record and separates documented contamination from unusual but potentially valid translation data.

The final audit reconciled the 60,701-record translated-only file with the 60,685-record analysis core and retained the 16-record difference in the project audit trail. Repository versions, commit identifiers, retrieval dates, file types and licence conditions are recorded with the project archive for the final deposit.

Every record was assigned to a functional domain by ordered deterministic rules, meaning that the same project, path, identifier and source wording always produce the same category. The six domains are interface, system notification, instruction or tutorial, terminology and entity naming, narrative description, and residual technical material. File paths and identifiers take priority over individual words. A record in a tutorial file remains instructional when it mentions a unit, while a graphical-interface identifier remains an interface record when it contains a historical name. The residual category is used when the available context supports only a broad technical assignment. This ordering is necessary because words such as unit, attack or load can occur in a tutorial, a button, a narrative passage or a technical message. The rules remain recoverable from the data and match the research question (McEnery and Hardie 2012; Saldanha and O'Brien 2014).

Two coders independently applied the same codebook to a 450-record sample stratified by project, language-version group and rule-assigned domain. Agreement was calculated from their separate decisions, and disagreements were resolved through adjudication using the codebook. The validation record preserves the automated label, both coder decisions, the adjudicated decision and coding notes. The full corpus retains the deterministic classification, allowing every reported profile to be reproduced from explicit rules. The human validation tests whether those rules produce defensible functional interpretations. Stratification ensured that the sample covered the main projects, language versions and functional domains. Independent coding preceded adjudication, so the agreement assessment reflects genuinely separate judgements. The design therefore combines reproducibility across the full corpus with completed human validation (Saldanha and O'Brien 2014).

Comparing technical markers and selecting close-analysis cases

Technical markers are code-like instructions embedded in a localisation string. They were extracted from both sides of each English-Chinese pair and assigned to five operational functions defined in the coding

book. The label `value_variable` covers inserted values and indexed substitutions. `formatting_markup` covers colour, emphasis, line breaks and XML-like forms. `interaction_marker` covers keyboard keys and other input cues. `language_dependent_macro` covers selectors that implement source-language grammar. `other_technical_token` retains recognised code-adjacent forms outside those classes. These labels are coding terms used to make the comparison reproducible.

Marker correspondence was calculated after classification. An exact sequence match preserves the same marker forms in the same order. A same-category transformation changes the visible marker form but preserves the ordered sequence of functions. An omission records at least one source category without a target counterpart, apart from the separate rule for grammatical macros. A language-sensitive reconfiguration occurs when an English grammatical selector is reorganised through Chinese grammar. A cross-category change expresses one marker function through another category. These labels describe the text in the files; build testing supplies evidence about operation in the game. All 708 omissions, comprising 404 in the primary group and 304 in zh_TW, underwent item-level checking in the relevant builds. Exact and category-level comparison were both retained because different token forms can perform the same function, while order was retained because exchanging the positions of values or commands can change what a string does. The build audit records the tested outcome separately for every omission.

Formal compactness is a descriptive length ratio calculated after technical markers were removed and repeated whitespace was collapsed. The number of visible Chinese characters was divided by the number of visible source characters; records with no visible source text were treated as missing. Because English words and Chinese characters are different units, the ratio serves only to locate comparatively short or long strings. It provides no measure of mental effort, translation difficulty, available screen space or semantic simplification.

Close analysis examines selected records from files rich in interface text, instructions and terminology. Each example is analysed for embedded technical elements, the available interface or game-world function, and the Chinese linguistic resources used in the target. The codebook distinguishes direct terminological equivalence, functional reformulation, grammatical reconfiguration, genre-compatible formation and project-specific system mapping. These labels organise case-level interpretation. Full-corpus frequencies are reported only for categories applied across the complete dataset.

The qualitative cases were selected by explicit relevance criteria. Marker cases represent retention, transformation, omission or grammatical reconfiguration. Interface cases require an identifier or file path that links the string to a control, tooltip or display function. Terminology cases require an identifiable referent and relevant project context. The cases cover the principal relations examined in the article; prevalence is reported only for categories coded across the corpus.

The rule-based classifier and close analysis answer different questions. The classifier divides the full corpus through reproducible cues, while close analysis interprets relations that depend on meaning and knowledge of the game. A terminology record can contain either direct equivalence or project-specific

naming, and an interface record can contain either lexical equivalence or functional reformulation. Keeping these levels separate prevents a type of software text from being mistaken for a translation strategy.

OpenAI Codex supported language editing, structural revision and scripts used to audit the supplied manuscript and coding book. Researchers completed the human coding, agreement assessment, adjudication and build-validation outcomes; the system neither generated nor altered those decisions. The authors remain responsible for the analytical interpretations, references and final submission.

The evidence is organised by analytical level. Localisation files support claims about marker relations, repository labels, identifiers and aligned wording. Running builds provide context-specific runtime outcomes for tested records. Screens and interaction data support analysis of visual presentation and usability, while commits and interviews support production-history claims. The study analyses public software text and involves no human participants or personal data.

Results

Retention of embedded software instructions across language versions

The passenger example illustrates technical continuity combined with grammatical change. In {COMMA}{NBSP}passenger{P "" s} and {COMMA}{NBSP}乘客 (wèi lǚkè, 'classifier + passenger[s]'), the instructions for punctuation, spacing and the inserted quantity remain recognisable. Chinese supplies the classifier 位 (wèi) and leaves 乘客 (lǚkè, 'passenger[s]') unchanged for number. The file therefore preserves the changing value and its spacing while Chinese reorganises the surrounding noun phrase.

Technical markers occur in 7,958 source records. Table 2 reports their correspondence by repository language-version group. The primary data contain 2,448 exact matches, 1,258 same-category transformations, 404 omissions, 64 language-sensitive reconfigurations and one cross-category change. The zh_TW data contain 2,572, 843, 304, 63 and one, respectively. The 140 records containing language-dependent macros were excluded from the category-retention denominator because their grammatical function requires separate analysis. Among the remaining 7,818 marker-bearing records, 7,108 retain either the identical ordered sequence or the same ordered pattern of marker functions (90.92%). Thirteen records contain both an exact or same-category relation and a language-dependent macro, which is why the numerator differs from the raw sum of the first two categories. The 90.92% estimate measures correspondence in the localisation files at the level of ordered marker form or coded function. Every one of the 708 omissions retains a separate item-level build result.

Table 2
Source-marker correspondence by repository locale group

Correspondence	zh/zh_CN	zh_TW	Combined
Exact sequence match	2,448	2,572	5,020
Same-category transformation	1,258	843	2,101
Omission	404	304	708
Language-sensitive reconfiguration	64	63	127
Cross-category change	1	1	2
Marker-bearing source records	4,175	3,783	7,958
<i>Note.</i> Category retention is 7,108/7,818 after records containing a language-dependent macro are excluded. Omission is a file-level correspondence label; build-test outcomes derive from the completed item-level audit.			

Value and formatting markers supply the clearest retained relations. OpenTTD renders {DECIMAL}{NBSP}km/h as {DECIMAL}{NBSP}千米/小时 (qiānmǐ/xiǎoshí, 'kilometres per hour') and {DECIMAL}{NBSP}kW as {DECIMAL}{NBSP}千瓦 (qiānwǎ, 'kilowatts'). Runtime values and spacing survive while the unit labels are localised. Tooltips containing {BLACK} preserve the colour instruction around reorganised Chinese syntax. Exact identity is only one form of retention: <Ctrl> may appear in a target where the source uses an internally typed interaction token, leaving the interaction category stable across different surfaces.

Exceptions identify audit priorities. The two cross-category records mark cases in which the target represents one source function through another marker class. The completed item-level build audit records a separate outcome for each of the 708 omission records. Marker correspondence locates the formal difference, while build checking establishes its technical consequence in the tested context.

Project composition shapes these totals. OpenTTD contains dense values, colours, cargo quantities and input cues. Wesnoth distributes formatting across help and narrative resources. 0 A.D. contains fewer marker-bearing records relative to its terminology inventory. These are profiles of repository composition, not measures of translation quality.

Same-category transformations show why token spelling alone is an incomplete basis for comparison. A source may represent a runtime string as {RAW_STRING} while the Chinese resource uses {STRING}, or a project may represent a key through a literal angle-bracketed form. The tokens differ, but their ordered categories receive the same broad value or interaction code. Technical adequacy still depends on the syntax of the relevant game. Category matching therefore provides a reproducible triage step between exact comparison and build inspection.

Marker order also matters. Chinese constituent order can move a retained value or instruction to another position in the string. The ordered-sequence rule records that movement, and close analysis checks whether the new position matches the value's meaning. This is especially important for indexed values: a

set-based comparison would treat two strings as equivalent even if their runtime arguments had exchanged positions. Surface form, functional category and order are therefore preserved as separate fields.

Language-dependent macros show why technical markers must be classified by function. English plural selectors encode a contrast between singular and plural forms. Chinese can express quantity through a numeral, classifier and noun without changing the noun for number. The passenger record is one of 140 plural-macro cases. A quality check based only on identical source and target tokens would incorrectly flag this grammatically complete Chinese construction. Language-pair-aware checking must first identify what the macro does.

How Chinese reformulates interface functions

The functional classifier identifies the main kinds of text represented in each project. In the zh/zh_CN group, OpenTTD contains 3,340 interface records, 495 system notifications, 216 instructional records and 715 terminology records. Wesnoth contains 12,965 narrative records, 2,026 instructional records, 2,258 terminology records, 71 notifications and 4,767 residual records. The 0 A.D. data contain 2,615 interface records, 2,083 terminology records, 643 narrative records, 250 instructional records and 455 residual records. Table 3 reports these assignments. Their interpretation is supported by the completed independent double coding, agreement assessment and adjudication of the stratified 450-record sample.

Table 3
Functional profile of the zh/zh_CN analytical group

Functional domain	OpenTTD	The Battle for Wesnoth	0 A.D.	Total
Interface	3,340	0	2,615	5,955
System notification	495	71	0	566
Instruction or tutorial	216	2,026	250	2,492
Terminology and entity naming	715	2,258	2,083	5,056
Narrative description	0	12,965	643	13,608
Residual technical material	0	4,767	455	5,222
Total	4,766	22,087	6,046	32,899
<i>Note.</i> Domains derive from explicit classification rules. The 450-record stratified validation was independently double coded and adjudicated.				

The selected OpenTTD cases illustrate several interface reformulations. The English tooltips construe an operation through an imperative verb: *Open company finances information*, *Open list of company's stations*, *Select filtering criteria*, and *Toggle large/small window size*. Their Chinese targets are 显示公司财务信息 (xiǎnshì gōngsī cáiwù xìnxī, “display company financial information”), 显示公司车站 (xiǎnshì gōngsī chēzhàn

lièbiǎo, “display the company station list”), 选择过滤条件 (*xuǎnzé guòlǚ tiáojiàn*, “select filtering conditions”), and 切换窗口大小 (*qiēhuàn chuāngkǒu dàxiǎo*, “switch window size”). All four retain an action-object relation through compact Chinese verb-object constructions.

The mapping of open to 显示 (*xiǎnshì*, “display/show”) shifts attention from launching a window to the resulting display. Toggle coverage area display becomes 显示覆盖范围 (*xiǎnshì fùgài fànwéi*, “display coverage area”). The English string names alternation; the Chinese string names a salient state. The file pair supports a state-centred reading. Screen and interaction evidence would be needed to assess clarity across both toggle states.

Longer tooltips organise sequences of actions. Save, load or abandon game, exit program becomes 保存游戏, 加载游戏, 放弃游戏, 退出 (bǎocún yóuxì, dúqǔ yóuxì, fàngqì yóuxì, tuìchū, ‘save game, load game, abandon game, exit’). Open map, extra viewport, cargo flow or list of signs becomes 显示地图, 额外视图, 货物流, 或标志列表 (xiǎnshì dìtú, fùjiā shìdiǎn, kèhuòliú huò biāozhì lièbiǎo, ‘display map, extra viewport, cargo flow, or sign list’). The targets retain the selectable objects and distribute one verb across several coordinated items.

Instructional strings can add action detail around retained input cues. One tooltip reads {BLACK} Resize window to default size. Ctrl+Click to store current size as default. Double Ctrl+Click to reset stored default. The target uses {BLACK} 按Ctrl+单击以将当前大小存储为默认大小。按Ctrl+双击以重置存储的默认大小。It preserves the colour and Ctrl cues, while 按 (*ànzhù*, ‘hold down’) and 键 (*jiàn*, ‘key’) make the embodied input explicit. Formal retention and linguistic expansion occur together.

A tree-planting tooltip retains <Ctrl> and <Shift> but gives no full lexical counterpart to Click+Drag. The target still presents planting, diagonal selection and cost display. Marker comparison identifies the preserved keys, while semantic comparison identifies reduced gesture detail. The item-level build audit records the tested outcome for this string; interaction evidence would be needed to determine whether the reduced wording affects use during play.

Formal compactness has a narrow supporting role. Primary medians are 0.375 for OpenTTD, 0.400 for Wesnoth and 0.364 for 0 A.D.; zh_TW medians are 0.375, 0.404 and 0.400. Interface and terminology records have lower primary medians than narrative records under the rule-based classifier. These distributions locate short label-like strings; cognitive economy and simplification remain outside what a cross-script character ratio can measure.

The zh_TW comparison changes only part of this profile. OpenTTD has the same median and Wesnoth differs by 0.004, while the 0 A.D. median rises from 0.364 to 0.400. Orthography, lexical choice and file composition may all contribute. A paired design restricted to shared identifiers would be required to attribute the difference more precisely.

How Chinese grammar and naming choices shape the text

Terminology-rich resources show how Chinese lexical choices operate within recurring sets of software labels. OpenTTD cargo labels include Goods as 货物 (*huòwù*, ‘goods’), Iron Ore as 铁矿 (*tiěkuàngshí*, ‘iron

ore’) and Passengers as 乘客 (lǚkè, ‘passenger[s]’). Their recurrence across menus and reports gives them a technical distribution, while their lexical form follows ordinary Chinese category naming.

Wesnoth uses compact Chinese fantasy compounds. Necromancer becomes 亡灵法师 (sǐlíng fǎshī), combining elements meaning ‘dead spirit’ and ‘mage’. Other labels include 圣骑士 (shèng qíshì, ‘paladin’) and 箭雨 (jiànyǔ, ‘rain of arrows’). Within Wesnoth, the aligned strings and their recurrence support a project-level account of internal form and consistency.

The historical resources in 0 A.D. call for project-level interpretation. Chandragupta Maurya is rendered as 月轮王 (Yuèhùwáng, ‘King Moon Protector’) in the supplied corpus. The file pair establishes this project-specific rendering; its wider historiographic status falls outside the evidence assembled here.

Project descriptions can also align source wording with game terminology. Equip your workers with seed drills becomes 工人使用播种机, 提高农业效率 (gōngrén shǐyòng lóuchē, tígāo nóngyè xiàolǜ, ‘workers use seed drills, increasing agricultural efficiency’). Structures + 10% health and + 10% capture points becomes 结构健康 + 10%, 忠诚度 + 10% (jiànzhùwù shēngmìng + 10%, zhōngchéng + 10%, ‘structures health + 10%, loyalty + 10%’). These examples show how descriptive wording is mapped onto project terminology. Interpretation of the underlying mechanics relies on the relevant game system or documentation.

Internal consistency provides another file-level relation. A term can recur under different keys in menus, descriptions and notifications, linking one lexical choice to several interface functions. Exact recurrence checks can identify source terms with multiple Chinese targets, while file paths show whether that variation occurs across labels and running text. Because variation may be grammatically or contextually motivated, consistency review proceeds in two stages: mechanical detection followed by project-informed interpretation. Terminology can therefore be traced as a network of linked labels across the game.

The historical and fantasy cases rest on different evidence. 乘客 is reported as 0 A.D.’s Chinese form, whereas wider historical authority would require historiographic evidence. 亡灵法师 is analysed through morphology and recurrence within Wesnoth. The localisation files support project-level interpretations of both names.

The two research questions are answered through three related analytical strands. Technical analysis asks which embedded instructions remain stable. Interface analysis examines how actions and states are expressed. Linguistic analysis examines how Chinese grammar and naming choices shape the target. Table 4 identifies the evidence used for each strand.

Table 4
Evidence used for the three analytical strands

Analytical strand	Evidence	Claim supported	Additional evidence
Technical-marker continuity	Ordered marker sequences and categories across the corpus	Formal retention or difference in 7,958 marker-bearing records	Completed item-level build audit
Interface reformulation	Identifiers, paths and selected aligned strings	Case-level changes in actions, objects and states	Completed double coding; screen or interaction data
Chinese grammar and naming	Plural macros and selected terminology records	Case-level grammatical and project-specific naming choices	External historical or genre sources

The three strands can converge in one tooltip or label. A string may retain a colour marker, express an action through a resulting state and use a recurring Chinese system term. File structure connects the wording to a software function, while Chinese linguistic choices organise how that function appears to players.

Discussion

The clearest corpus-level result concerns ordered marker correspondence. Among 7,818 marker-bearing records outside the grammatical-macro set, 7,108 (90.92%) retain an identical sequence or the same ordered functional pattern. This provides corpus-scale evidence relevant to accounts of translation technology and materiality (Cronin 2013; Littau 2016; Olohan 2021). Values, formatting instructions, input cues and grammatical selectors make the software layer visible within the bilingual record. The measured stability lies in ordered marker forms and coded functions, even when the surrounding symbols and wording change. The completed build audit supplies a separate layer of evidence for the omission subset (O'Hagan 2015; Pirrone and D'Ulizia 2024).

Material mediation names a relation distinct from technical constraint. A constraint limits what can be done; material mediation describes how a file connects an executable function with a linguistic form. For example, an English plural macro encodes a grammatical choice, while a Chinese classifier construction reorganises that choice around the same changing value. This file-level account places the observed relation within scholarship on materiality and software-mediated translation (Cronin 2013; Littau 2016; O'Hagan 2015; Olohan 2021).

The three analytical strands rest on complementary evidence. Corpus-wide counts support conclusions about marker retention. Identifiers, paths and selected aligned strings support the analysis of interface reformulation. Plural-macro cases and selected terminology records support the analysis of Chinese grammar and naming. The strands describe related dimensions of the same records, not mutually exclusive categories for the whole corpus.

The 708 omission records show why formal comparison and runtime testing must remain separate. Marker comparison identifies an absent source category. The completed item-level build audit records the tested outcome of each case. Textual absence and software consequence are therefore separate variables linked at record level.

Grammatical macros show where file design meets language structure. English plural selectors expect a visible number contrast in the source string. Chinese can express quantity through a numeral, classifier and number-neutral noun. The absent suffix macro in 缺缺 is therefore a grammatical reorganisation around a value that remains technically anchored. Quality procedures need marker categories and language-pair rules before they assess surface absence.

The selected interface cases illustrate several forms of reformulation. English commands name actions such as opening, selecting or toggling. Chinese targets can foreground the displayed state, the object affected or a compact sequence of actions. For example, translating open as 开 presents the result of the operation, while coordinated Chinese structures can share one verb across several interface objects. The completed validation supports the functional categories, and the selected examples show how they operate in context. These case-level patterns are consistent with accounts of localisation as functional adaptation within software environments (Jiménez-Crespo 2013; O'Hagan and Mangiron 2013). The aligned records support case-level claims about the relation between wording and encoded function. Claims about screen layout or player interaction require those additional forms of evidence.

An identifier is a code-like name that links a text segment to a software function. Keys such as STR_TOOLTIP_CATCHMENT and STR_TOOLBAR_TOOLTIP_DISPLAY_MAP can connect a string to a control family even when the screen is unavailable, helping to distinguish a command from a descriptive sentence. Icons, neighbouring controls and toggle states belong to the screen and interaction layer, which is the relevant evidence for usability.

Interface reformulation also requires flexible units of comparison. A Chinese verb can shift emphasis from an action to its result, repeated verbal material can be shared across coordinated objects, and a brief input cue can be expanded into a local phrase. The relevant unit may therefore be a full string, a clause, a marker or a single lexical item. The unit is selected according to the relation under examination and the available evidence.

This evidence boundary complements usability-oriented accounts of localisation (Jiménez-Crespo 2013) by separating interface function from visible length. Formal compactness can locate short records, yet similar ratios may represent direct equivalents, reformulated states or reduced operational detail. Semantic comparison distinguishes these cases. Screen recordings and user testing would be needed to evaluate their effects during play.

Terminology adds a cultural dimension to the file-based relation. Pettini's (2022) distinction between real and fictional cultural references explains why historical and fantasy names require different forms of validation. Repetition within a project supports analysis of local consistency. Historical authority and

genre-wide convention, by contrast, depend on external historiographic, lexicographic or comparative evidence. The present corpus supports project-level description.

Repository language tags also have a sociolinguistic dimension. Fernández-Costales (2017) argues that game localisation concerns language availability and circulation as well as textual choice. Tags such as zh, zh_CN and zh_TW are traces of repository organisation and technical decisions, not complete descriptions of language communities. Analysing zh_TW separately avoids pooling distinct repository versions and keeps interpretation at the level supported by the data.

Three methodological implications follow. Identifiers, language-version tags, raw fields and classified markers should remain attached to aligned text. Automated comparison can locate formal differences before build testing. Language-version branches should be separated before projects are compared. Together, these practices make corpus analysis of software text traceable from the original record to the reported result.

The study also adapts familiar corpus principles to software text. Representativeness still depends on the relation between the stated population and the corpus design (Biber 1993), but repeated implementation records can be meaningful because the same wording may serve several software locations. Human reliability procedures remain necessary for categories based on communicative judgement, while deterministic marker extraction can be reproduced from stated rules. Parallel alignment remains central, but correspondence can involve executable function, interface action or naming as well as wording. The corpus must therefore retain the technical fields that make each segment interpretable, just as other specialised corpora retain register information (Dixon 2024). Quantitative summaries locate recurring formal relations, and close analysis explains the linguistic organisation of selected cases.

Different evidence types support different claims. Automated procedures identify file paths, marker sequences and formally defined measures (Baker 1993; Bernardini and Ferraresi 2011; Hu 2016; Zanettin 2012). Human validation supports categories that depend on communicative function or game knowledge. Running builds support runtime claims, screens support visual claims, interaction data support usability claims, and commits or interviews support production-history claims. This division gives each part of the coding package a specific methodological role.

Limitations

The corpus comprises three open-source strategy and simulation games. Its external validity is therefore limited to these archived repositories and comparable projects, not game localisation as a whole. Wesnoth contributes 38,739 of 60,685 records, which gives that project substantial weight in aggregate results. The counting unit is an implementation record, and repeated strings attached to different identifiers are retained. Frequencies describe software placements; claims about statistically independent translation decisions require a different sampling unit. Grouping zh and zh_CN as the

primary repository set also permits corpus-level comparison while potentially obscuring differences between those tags.

The 90.92% retention estimate concerns ordered marker identity or coded functional correspondence in localisation files. It does not by itself establish semantic equivalence, translation quality or runtime correctness. Human validation covers a stratified 450-record sample, while deterministic rules classify the full corpus of 60,685 records. The completed double-coded validation and item-level build audit address functional coding and runtime outcomes. The build audit covers the 708 records classified as omissions; the other correspondence categories were not exhaustively tested in running builds. Build outcomes also remain specific to the tested versions, configurations and interaction contexts. The character ratio is descriptive because English and Chinese are measured in different units.

The qualitative cases were selected to explain coded relations and support case-level interpretation. Corpus-wide prevalence is reported only for categories applied across the complete dataset. The corpus contains no screen-position, player-response, contributor-identity, commit-history or interview data. It therefore cannot identify translator intention, reconstruct production decisions, or evaluate visual prominence, usability and reception. Historical authority and genre-wide naming conventions also require external evidence beyond recurrence within these projects. These limits confine generalisation to the sampled repositories and the evidence types analysed here.

Conclusion

Localisation files matter analytically because translated software text is both linguistic and executable. In this corpus, values, formatting commands and input cues show high ordered correspondence. The selected Chinese cases illustrate how grammar and interface wording can reorganise the visible expression of those functions. Material mediation describes this connection between software structure and linguistic form: an executable relation can persist through a change in visible language.

Across the three repositories, marker analysis identifies high retention at the level of ordered technical form or coded function. The completed independent double coding, agreement assessment and adjudication support the functional profiles, while the item-level build audit distinguishes formal omission from tested software outcomes. The selected cases show how technical continuity can coexist with grammatical and functional reformulation. Taken together, the findings support the use of localisation files as one evidence source for relations between software structure and translation.

Declarations

Competing interests

The author declares no competing interests.

Ethical statement

Ethical approval

was not required because this study analysed publicly available software localisation materials and involved no human participants or personal data.

Author Contribution

Bao conceived the study, developed the methodology and analytical framework, curated the corpus, conducted the formal analysis and validation, and interpreted the results. Bao wrote the original draft and reviewed and edited the manuscript.

Data Availability

The data analysed in this study were derived from publicly available localisation resources from the OpenTTD, The Battle for Wesnoth, and 0 A.D. open-source game projects. The processed datasets and supporting materials generated for this study, including the coding framework, validation records, build-audit records, and analysis materials, have been deposited in a public repository and are available at related files.

References

1. Al-Ajarmeh O, Al-Adwan A (2022) Insights into blending game localisation in the Arab world: Arafiesta as a case study. *J Int Localization* 9(1):61–84. <https://doi.org/10.1075/jial.22002.aja>
2. Al-Batineh M (2021) Issues in Arabic video game localization: a descriptive study. *Transl Interpret* 13(2):45–64. <https://doi.org/10.12807/ti.113202.2021.a03>
3. Al-Batineh M, Alawneh R (2022) Current trends in localizing video games into Arabic: localization levels and gamers' preferences. *Perspectives* 30(2):323–342. <https://doi.org/10.1080/0907676X.2021.1926520>
4. Baker M (1993) Corpus linguistics and translation studies: implications and applications. In: Baker M, Francis G, Tognini-Bonelli E (eds) *Text and technology: in honour of John Sinclair*. John Benjamins, Amsterdam, pp 233–250. <https://doi.org/10.1075/z.64.15bak>
5. Bernal-Merino MÁ (2015) *Translation and localisation in video games: making entertainment software global*. Routledge, New York. <https://doi.org/10.4324/9781315752334>
6. Bernardini S, Ferraresi A (2011) Practice, description and theory come together: normalization or interference. *Italian Tech translation? Meta* 56(2):226–246. <https://doi.org/10.7202/1006174ar>
7. Biber D (1993) Representativeness in corpus design. *Lit Linguist Comput* 8(4):243–257. <https://doi.org/10.1093/lc/8.4.243>

8. Bowker L (2002) Computer-aided translation technology: a practical introduction. University of Ottawa, Ottawa
9. Chandler HM, Deming SO (2012) The game localization handbook, 2nd edn. Jones & Bartlett Learning, Sudbury
10. Cronin M (2013) Translation in the digital age. Routledge, London.
<https://doi.org/10.4324/9780203073599>
11. Dietz F (2006) Issues in localizing computer games. In: Dunne KJ (ed) Perspectives on localization. John Benjamins, Amsterdam, pp 121–134. <https://doi.org/10.1075/ata.xiii.10die>
12. Dixon DH (2024) Introducing the Single Player Offline Game Corpus (SPOC): a corpus of seven registers from digital role-playing games. *Corpora* 19(1):107–122.
<https://doi.org/10.3366/cor.2024.0300>
13. Dunne KJ (ed) (2006) Perspectives on localization. John Benjamins, Amsterdam.
<https://doi.org/10.1075/ata.xiii>
14. Esselink B (2000) A practical guide to localization. John Benjamins, Amsterdam.
<https://doi.org/10.1075/liwd.4>
15. Fan L, Yao Y, Xie R, Sio CI, Cheung AKF (2026) Testing the simplification universal in game localization: a quantitative comparison of linguistic complexity in Black Myth, Sekiro, and RDR2. *Humanit Soc Sci Commun* 13:120. <https://doi.org/10.1057/s41599-025-06424-0>
16. Fernández-Costales A (2012) Exploring translation strategies in video game localization. *MonTI* 4:385–408. <https://doi.org/10.6035/MonTI.2012.4.16>
17. Fernández-Costales A (2017) On the sociolinguistics of video games localisation. *J Int Localization* 4(2):120–140. <https://doi.org/10.1075/jial.00001.fer>
18. Guillot MN, Pavesi M, Desilla L (2019) AVT as intercultural mediation. *Multilingua* 38(5):495–504.
<https://doi.org/10.1515/multi-2018-0115>
19. Hu K (2016) Introducing corpus-based translation studies. Springer, Berlin.
<https://doi.org/10.1007/978-3-662-48218-6>
20. Jarrah S, Haider AS, Al-Salman S (2023) Strategies of localizing video games into Arabic: a case study of PUBG and Free Fire. *Open Cult Stud* 7:20220179. <https://doi.org/10.1515/culture-2022-0179>
21. Jiménez-Crespo MA (2013) Translation and web localization. Routledge, London.
<https://doi.org/10.4324/9780203520208>
22. Jiménez-Crespo MA (2017) Crowdsourcing and online collaborative translations: expanding the limits of Translation Studies. John Benjamins, Amsterdam. <https://doi.org/10.1075/btl.131>
23. Jooyaeian E, Khoshsaligheh M (2022) Translation solutions in professional video game localization in Iran. *Games Cult* 17(7–8):1054–1074. <https://doi.org/10.1177/15554120221077726>
24. Kress G (2010) Multimodality: a social semiotic approach to contemporary communication. Routledge, London. <https://doi.org/10.4324/9780203970034>

25. Laviosa S (2002) *Corpus-based Translation Studies: theory, findings, applications*. Rodopi, Amsterdam. <https://doi.org/10.1163/9789004485907>
26. Littau K (2016) Translation and the materialities of communication. *Transl Stud* 9(1):82–96. <https://doi.org/10.1080/14781700.2015.1063449>
27. Mandiberg S (2021) Video games have never been global: resituating video game localization history. In: Swalwell M (ed) *Game history and the local*. Palgrave Macmillan, Cham, pp 177–198. https://doi.org/10.1007/978-3-030-66422-0_10
28. Mangiron C (2017) Research in game localisation. *J Int Localization* 4(2):74–99. <https://doi.org/10.1075/jial.00003.man>
29. Mangiron C (2018) Game on! Burning issues in game localisation. *J Audiovis Transl* 1(1):122–138. <https://doi.org/10.47476/jat.v1i1.48>
30. Mangiron C, O'Hagan M (2006) Game localisation: unleashing imagination with 'restricted' translation. *J Spec Transl* 6:10–21. <https://doi.org/10.26034/cm.jostrans.2006.735>
31. McDonough Dolmaya J (2012) Analyzing the crowdsourcing model and its impact on public perceptions of translation. *Translator* 18(2):167–191. <https://doi.org/10.1080/13556509.2012.10799507>
32. McEnery T, Hardie A (2012) *Corpus linguistics: method, theory and practice*. Cambridge University Press, Cambridge. <https://doi.org/10.1017/CBO9780511981395>
33. Moreno García LD, Mangiron C (2024) Exploring the potential of GPT-4 as an interactive transcreation assistant in game localisation: a case study on the translation of Pokémon names. <https://doi.org/10.1080/0907676X.2024.2378346>. Perspectives
34. O'Brien S (2011) Collaborative translation. In: Gambier Y, van Doorslaer L (eds) *Handbook of Translation Studies*, vol 2. John Benjamins, Amsterdam, pp 17–20. <https://doi.org/10.1075/hts.2.col1>
35. O'Hagan M (2009) Evolution of user-generated translation: fansubs, translation hacking and crowdsourcing. *J Int Localization* 1(1):94–121. <https://doi.org/10.1075/jial.1.04hag>
36. O'Hagan M (2015) Game localisation as software-mediated cultural experience: shedding light on the changing role of translation in intercultural communication in the digital age. *Multilingua* 34(6):747–771. <https://doi.org/10.1515/multi-2014-0062>
37. O'Hagan M (2017) Seeking delocalization. *J Int Localization* 4(2):183–202. <https://doi.org/10.1075/jial.00004.oha>
38. O'Hagan M (2022) Indirect translation in game localization as a method of global circulation of digital artefacts. *Target* 34(3):441–464. <https://doi.org/10.1075/target.00007.oha>
39. O'Hagan M, Mangiron C (2013) *Game localization: translating for the global digital entertainment industry*. John Benjamins, Amsterdam. <https://doi.org/10.1075/btl.106>
40. Olohan M (2021) *Translation and practice theory*. Routledge, London. <https://doi.org/10.4324/9781315514772>

41. Pettini S (2022) The translation of realia and irrealia in game localization: culture-specificity between realism and fictionality. Routledge, London. <https://doi.org/10.4324/9781003001935>
42. Pérez-González L (2014) Audiovisual translation: theories, methods and issues. Routledge, London. <https://doi.org/10.4324/9781315762975>
43. Pirrone M, D'Ulizia A (2024) The localization of software and video games: current state and future perspectives. *Information* 15(10):648. <https://doi.org/10.3390/info15100648>
44. Pym A (2004) The moving text: localization, translation, and distribution. John Benjamins, Amsterdam. <https://doi.org/10.1075/btl.49>
45. Saldanha G, O'Brien S (2014) Research methodologies in Translation Studies. Routledge, London. <https://doi.org/10.4324/9781315760100>
46. Shehab E, Odeh S (2025) Linguistic issues in the localization of English video games into Arabic: a descriptive study. *Transl Spaces*. <https://doi.org/10.1075/ts.25045.she>
47. Toftedahl M (2025) Towards a deeper understanding of localization as a game development process: game studio studies in Sweden and China. *Entertain Comput* 54:100944. <https://doi.org/10.1016/j.entcom.2025.100944>
48. Tognini-Bonelli E (2001) Corpus linguistics at work. John Benjamins, Amsterdam. <https://doi.org/10.1075/scl.6>
49. Zanettin F (2012) Translation-driven corpora: corpus resources for descriptive and applied Translation Studies. Routledge, London. <https://doi.org/10.4324/9781315759661>

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [codingbook.xlsx](#)