Hindawi
International Journal of Computer Games Technology
Volume 2021, Article ID 9979657, 15 pages
https://doi.org/10.1155/2021/9979657

Review Article

Localization Tools in General Purpose Game Engines: A Systematic Mapping Study

Marcus Toftedahl

Division of Game Development, University of Skövde, Skövde, Sweden

Correspondence should be addressed to Marcus Toftedahl; marcus.toftedahl@his.se

Received 31 March 2021; Accepted 5 July 2021; Published 23 July 2021

Academic Editor: Cristian A. Rusu

This paper addresses localization from a game development perspective by studying the state of tool support for a localization work in general purpose game engines. Using a systematic mapping study, the most commonly used game engines and their official tool libraries are studied. The results indicate that even though localization tools exists for the game engines included in the study, the visibility, availability, and functionality differ. Localization tools that are user facing, i.e., used to create localization, are scarce while many are tool facing, i.e., used to import localization kits made outside the production pipeline.

1. Introduction

"The world is full of different markets and cultures and, to maximize profits™[sic], nowadays games are released in several languages. To solve this, internationalized text must be supported in any modern game engine." [1]

This introductory quote is a snippet from the "*Games and Internationalization*" section in the official documentation of the open source game engine *Godot* [2]. With the rise of digital storefronts and app stores, even small scale game developers have the opportunity to act on a global market. The availability of game creation tools and game engines has also been vital for a more diverse game development scene where game creation is accessible not only to professionals in the field [3]. The indie game development scene, where small-scale game development companies develop and release digital games, has been identified as a growing field [4]. The indie term can have different meanings, depending on the angle relating both to the "look-and-feel" of a certain game and to the specific production setting [5]. Freeman and McNeese [6] lift the differences between indie game production and larger production settings in the "mainstream" game industry, highlighting the differences in production and distribution structures, while a mainstream/AAA game development project consists of large teams, having big budgets and a structure including separate functions or specific corporate entities handling functions such as marketing or distribution. This is not always the case with indie game development, where Pereira and Bernardes [7] note that the structure of indie development is more flexible, including both the development process and the distribution.

Nicoll and Keogh [8] highlights *Unity* [9] as a game engine initially targeted to the "grassroots" of game development, including small teams and indie developers. *Unity*, as well as other game engines, provides a platform for developers to develop games using readily available game creation tools and other necessary components. A modern game engine contains a number of tools to create functionality and content in a game project. Game engines also provide functionality to deploy the produced games on a wide variety of platforms, including both multipurpose platforms (such as computers and smartphones) and specialized gaming platforms (i.e., game consoles). Connection to digital storefronts and distribution systems can also be provided through a game engine, where common distribution channels such as *Steam*, *Google Play*, and other digital storefronts can be used. As these digital channels are global, games can be released to an international audience; hence, localization is needed. Previous research has shown that independent developers with limited resources focus on localization late in the development process, in some cases even first when the game is released, before realizing the need of localization [10]. This

indicates that there could be a lack of awareness regarding localization as a part of the development process. The localization process described from a software perspective involves creating a localization-friendly product through planning for localization, having technical support for multiple languages and other variables such as measuring units and currencies [11]. From a media perspective, i.e., film and television, the focus on localization work often lies on subtitling and audio dubbing [12]. Localization from a software and a media perspective is relatively straight forward where a localization-friendly development process is defined [13].

Game localization practices have its root in these fields, but since games and game production are neither the same nor solely a combination of media and software production [14], this paper is a step towards defining a localization-friendly game development process. This is done by exploring the state of localization tools in modern game engines using a systematic mapping methodology. The focus is on general purpose game engines (such as *Unity*) and their connected tool libraries. The main research question is formulated as "How can a localization-friendly game production pipeline be constructed using a general purpose game engine and its connected tool library?" and is explored through three (3) subquestions: "Which commonly used general purpose game engines have tools included for game localization"; "How are the localization tools described in the corresponding official documentation"; and "In which ways can the production pipeline be expanded by adding additional tools to achieve a localization-friendly development process?"

2. Game Engines and Tools

Game engines have become a term commonly used to describe the software used to create a game. During the mid-1980s, game developers shifted towards creating tools that could be reused to create multiple titles, often within the same game genre instead of building each and every game from scratch [15]. An example of this is *SCUMM* (*Script Creation Utility for Maniac Mansion*) initially created to produce the game *Maniac Mansion* [16]. *SCUMM* was, after the release of *Maniac Mansion*, used to create a number of other game titles in the point and click adventure game genre during a period spanning over 20 years. As the time went and new games were made, updates were made to *SCUMM* to adhere to new hardware standards in the game industry (i.e., higher resolution graphics, digitized sound, and different media formats) as well as new tool's end features to enhance the workflow in production while keeping the basic structure of how the game assets are organized [17]. While *SCUMM* was an in-house game engine, not available to the public to use, there are numerous examples of game engines and game creation tools released to the public either commercially or as open source software. From an historical perspective, game creation tools such as *Garry Kitchen's Game Maker* [18] (depicted in Figure 1) or *Shoot'Em Up Construction Kit* [19] have allowed their users to create games using built-in tools for asset creation and game logic.

Game creation tools like these have been used by both professionals and hobbyists to create games, and while neither *SCUMM* nor *Garry Kitchen's Game Maker* or any of the other game creation tools mentioned used the term game engine, they allowed creators to leap ahead in the game production cycle by using readily available tools [20]. The term game engine was popularized in the early 1990s in connection to the rise of PC gaming, where the game *Doom* [21] was marketed with a specific focus on its technology and tools for creating content in the game, using the term *Doom Engine* to describe the technology behind the game [22]. The term game engine has been identified as ambiguous, used to describe a wide range of game creation software packages and related concepts, as well as being used in everyday game development lingo as the software used to create a game. This has led to misunderstandings and difficulties in communicating the processes around game development, where the term game engine is used both referring to the underlying technology running the game and creating the game [23].

In this paper, the focus lies on game engines as part of the production pipeline when developing a game. Toftedahl and Engström [24] have proposed a taxonomy to nuance the terminology of game engines as game production software. In the proposed taxonomy, game engines are divided into *general purpose game engines* (i.e., a game engine targeted at a broad range of game genres) and *special purpose game engines* (game engines targeted at specific game genres). In connection to the game engine, a number of tools are included in the production pipeline. The production pipeline is, based on O'Donnell [25] and Toftedahl and Engström [24], an important entity in the game development process, where the flow of production is handled by game creation software. The production pipeline and the game engine it contains are characterized by a very high degree of plasticity, i.e., tools can be added and altered in order to suit the specific needs for the development of a game. The altering of the structure of the production setting by using additional tools to enhance the workflow is addressed by O'Donnell [25] describing the production pipeline as "[…] the set of practices, tools, standards, aesthetics and themes that will carry the project into the production phase of game development". Figure 2 shows a model of a production pipeline containing a game engine and an arbitrary number of tools. The tools are divided into three categories: *user facing tools* (tools for content creation, typically with a GUI); *tool facing tools* (integration tools connecting different parts of the pipeline or middleware adding functionality); and *product facing tools* (in the engine core, highly optimized) [24].

The production pipeline can be specific to each and every project created, but when a pipeline is deemed effective enough, it can become a specific asset for the company, used for more than one project. One example of this is the *Unreal Engine* [26] originally developed internally during the creation of the game *Unreal series* [27] of games and later released as a commercial game engine [25]. During recent years, a plethora of game engines have become available for game developers of all sizes and levels of professionalism. Nicoll and Keogh [8] highlight how *Unity* [9] has become a widely used game engine and emphasize the importance of available tools to make game creation possible. Further, *Unity's* setup using a component oriented system, where the workflow

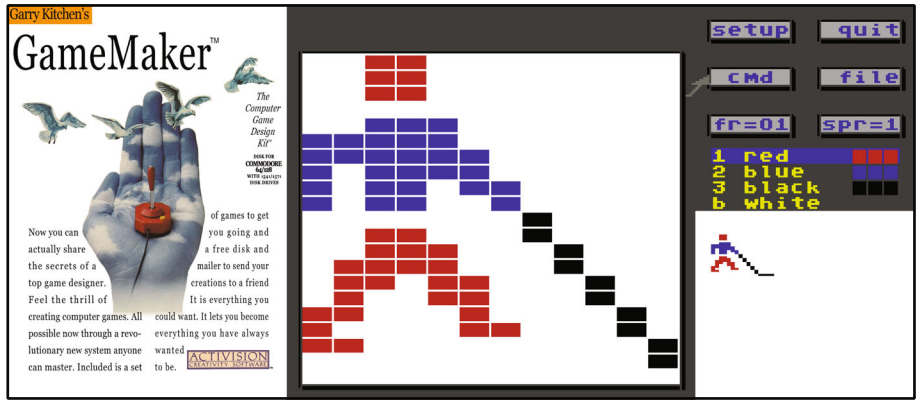

Figure 1: Cover and sprite editor tool from Garry Kitchen's Game Maker.

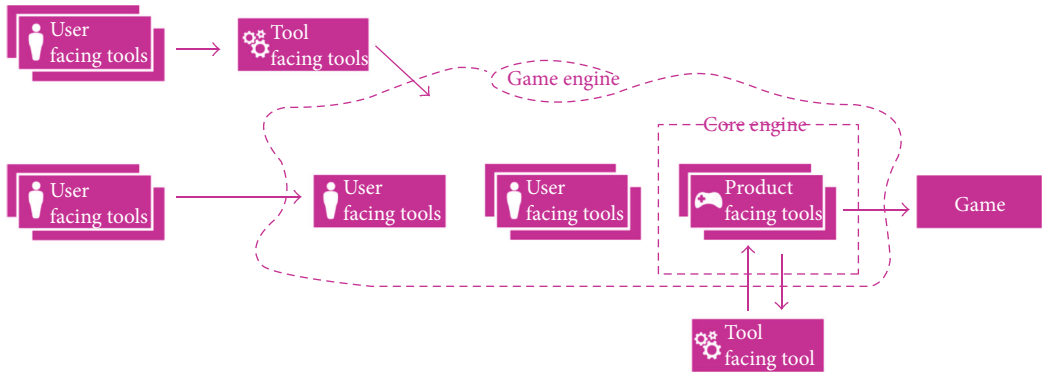

Figure 2: A model of a game development production pipeline.

can be altered based on the individual competences instead of relying on all implementation/coding to be conducted by programmers, is a feature added to both the creativity and flexibility in a game development project [8]. While *Unity* fits into the general purpose category of game engines, i.e., it can be used to create a wide variety of games for many platforms, it also contains a set of standard tools for game creation tasks as well as an official tool library, where the production pipeline can be expanded with additional tools by relatively simple means to fit the purpose of the development project in question. This is a model used by several publicly available general purpose game engines.

3. Game Localization Practices and Translation Management Systems

Game localization and its underlying processes are well-documented, based on its roots in software and media localization. The area of game localization has gained some attention from a translation studies research perspective where games as a medium for localization have been scrutinized from a linguistic and translation perspective [28]. From a translator's perspective, the book *Game Localization: Translating for the Global Digital Entertainment Industry* [29] gives an in-depth description of the field, including both overview of research conducted in the area and case descriptions from game localization projects. From a game development research perspective, there are few papers addressing game localization [28, 30]. While research is scarce, the area is somewhat covered by industry sources including best practice books, such as *The Game Localization Handbook* [31] describing the process from a practitioner's perspective.

Regardless the perspective, there is a common view regarding the importance of a localization-friendly development process. Bernal-Merino [32] highlights the industrial process of game localization and argues that localization aware procedures need to be integrated into the complex game development process on multiple levels and stages. Further, Bernal-Merino [32] emphasizes the variety of platforms and their inherent differences (i.e., ranging from mobile phones to dedicated game systems) making a universal game localization model difficult to design. Fry and Lommel [11] describes a global product development cycle, a general process of developing a product for a global market (see Figure 3). The development process is split into two main phases: internationalization and localization. The first phase in the process chronologically is internationalization, where the product is developed and implemented in such way as the second phase, localization, is possible. This means that the product is designed to be ready for localization, where necessary alterations in terms of language and other content can be made without redesigning the product. Localization is then the second phase, where translations and other adaptations to the product are made, using the system support created in the internationalization process.

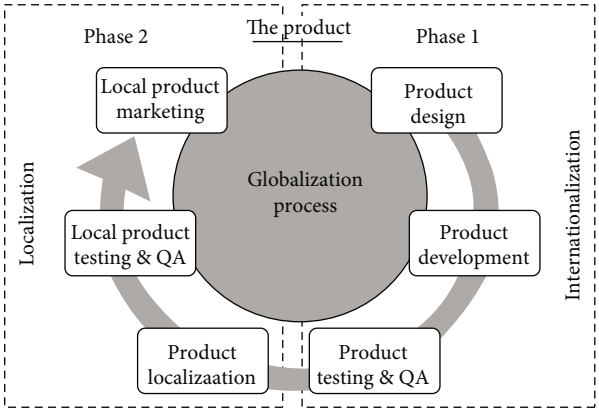

Figure 3: The global product development cycle, after Fry and Lommel [11].

Chandler and Deming [31] mean that the goal of internationalization is to create a project that easily can be localized with minimum effort and resources for the developer. When planned for, all game features can then be present in international versions of the game. If the game project is designed and planned for with internationalization as a guiding factor, the necessary alterations in the localization phase are easier to make. The planning aspect is also highlighted by O'Hagan and Mangiron [29], meaning that a localization-friendly development process planned for in advance saves considerable development time, compared to the ad hoc processes where localization is an afterthought and has to be implemented retrospectively.

To aid in the process of localizing a product, specific localization tools can be used. Esselink [33] describes the evolution of localization tools, where mainly early developers of office suites saw the need to adapt their software to be useful in different regions around the world. The localization-friendly development and the model depicted in Figure 3 have its origins in the software localization realm. The use of a localization kit is a part of the localization-friendly development setting. A localization kit contains information of the assets be localized in the process. It most basic form, a localization kit, is a text file with a structure where a number of attributes can be assigned to each text string; i.e., the file contains all localized text strings that should be displayed in the game, separated by language [32]. Each text string is provided with a specific identification number used to identify when and where the text line should be displayed at runtime. More advanced localization kits include the assets that should be localized in a folder structure. Esselink [13] means that a well-prepared localization kit will save considerable time when conducting the localization process. A well-prepared localization kit is especially valuable when working with several outsourced translators, i.e., when separate languages are handled by different external contacts. A localization kit is also essential when obtaining project quotations from external vendors, allowing for a more exact estimation of the costs for the work in question. Further, Esselink [13] highlights three components that a localization kit generally should contain, preferably in a well-organized folder structure: (1) localization guidelines and schedule information, including schedules and deliverables; (2) build environments and source files, i.e., information regarding the structure of the project and its technical aspects; and (3) reference materials to show and describe the product, including beta versions or other work material. To aid the help of creating and managing these localization kits, a translation management system (TMS) can be used. Figure 4 shows an example of a TMS, where the status of each translation can be managed and monitored.

Since localization relates to several types of assets in a game, the project structure of the localization kit is important. Localization is complex taking the three aspects of localization into consideration: linguistic issues (relating to the translation of the language in the product); content and cultural issues (including presentation of information through icons, graphics, or color); and technical issues (including the technicalities behind doing necessary alterations to a product without having to rebuild everything from scratch) [11]. Muñoz Sánchez and López Sánchez [35] have identified four good practices that a game development team should have in mind from the early stages of development to achieve a localization-friendly development project: (1) separation of the translatable text from the source code (i.e., not "hard coded" text strings that should be displayed in the game at runtime); (2) selection of appropriate fonts and coding for different languages (i.e., support for a wide variety of character sets, while maintaining both a coherent visual appearance and technical support); (3) interface design adaptable to the text shown or with plenty of space (i.e., accommodating for different layouts of text boxes shown in game due to varying lengths of translated text strings); and (4) use of special tags for gender, sex, or numbers (due to different pronouns, etc., in certain languages). A localization-friendly game production pipeline should be able to handle the aspects of localization, where linguistic and content and cultural issues are supported with a well-planned technical infrastructure.

4. Research Questions and Method

This paper has the aim to survey the state of localization tools in general purpose game engines and the possibility to have a localization-friendly development setting. The paper addresses both the localization tools included in the standard installation of the game engine and external tools that can be added to expand the production pipeline. This has been achieved through a systematic mapping study, focusing on general purpose game engines and the localization tools available in connection to the included engines. A systematic mapping study is described by Petersen et al. [36] as a method useful to get an understanding and overview of an area, in comparison to the systematic review used to scrutinize an area more in detail. Petersen et al. [36] highlights that the main differences between a systematic mapping study and a review study lie in the granularity of the results and the formulation and connection to the research questions.

The main research question guiding this paper is formulated as follows: "How can a localization-friendly game

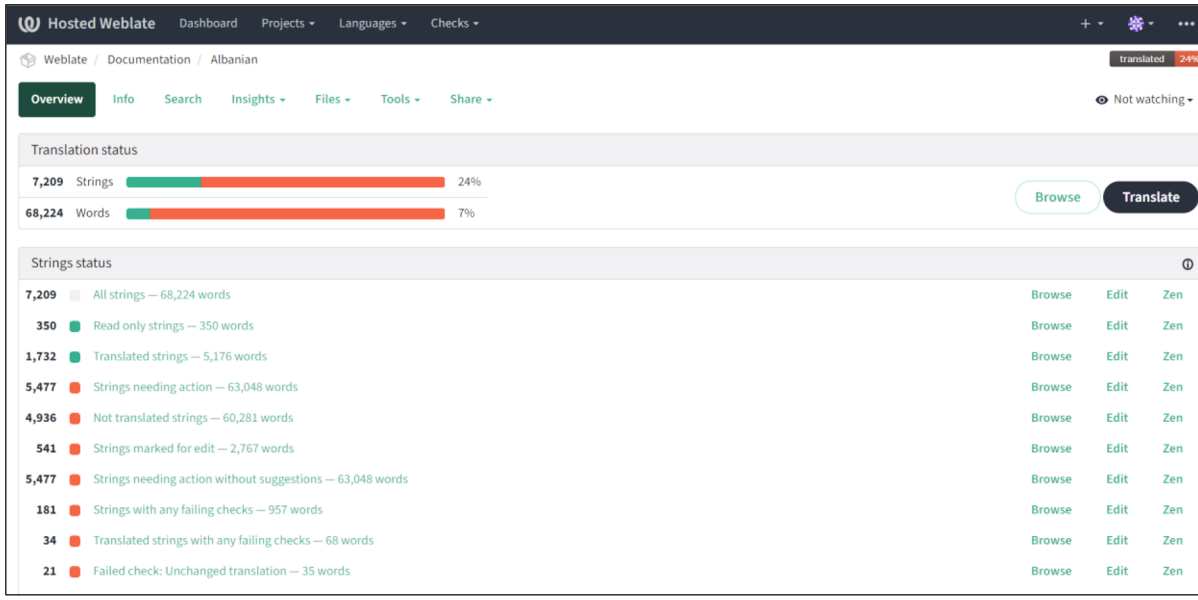

Figure 4: Screenshot of a TMS: Weblate [34], an open source web-based system for localization.

production pipeline be constructed using a general purpose game engine and its connected tool library?"

In order to explore this, three (3) subquestions are addressed:

(i) Which commonly used general purpose game engines have tools included for game localization?

(ii) How are the localization tools described in the corresponding official documentation?

(iii) In which ways can the production pipeline be expanded by adding additional tools to achieve a localization-friendly development process?

Through the subquestions, it will also be possible to explore the general area around game localization tools, to get a deeper understanding of the current state of game localization tools in connection to general purpose game engines.

5. Inclusion Criteria for the Game Engines in the Study

The inclusion criteria for the game engines in this study are twofold: (1) general purpose game engines that are (2) publicly available. The data regarding commonly used game engines by Toftedahl and Engström [24] is used to identify relevant game engines. In Table 1, an overview of the included game engines and their respective inclusion criteria is presented. The table is divided in two main sections where the ten most used game engines identified from the digital game stores *Steam* [37] and *Itch.io* [38] are listed.

Several of the game engines based on the data from the *Steam* game store are excluded due to their in-house nature, i.e., they are not available to the public and are either only used for in-house projects or available only by specific licensing agreements. Due to the closed nature of these game engines, it is also difficult to categorize them into the general purpose criteria, but since they are excluded based on the availability aspect, this is not a factor impacting this study. In contrast to the game engines identified from the *Steam* data, the game engines identified through the data from *Itch.io* are all available to the public. The main reason for exclusion in the *Itch.io* section is the general purpose criteria. Several identified game engines from Itch.io are special purpose game engines, i.e., only for producing games in a specific genre or on very specific target platforms. *Twine* [39], *Bitsy* [40], and *Ren'Py* [41] are used to create text driven games or visual novels; *RPG Maker* [42] is aimed at the RPG genre; and *PICO-8* [43] is used to create games for a retro-inspired "fantasy console" with strict limitations in terms of size and performance.

GameMaker [44] and *Construct* [45] are included in the general purpose category even though they have limited 3d graphics functionality. While they are mainly used for 2d games, both *GameMaker* and *Construct* can be used to create games in multiple genres and the games can be played on a multitude of platforms. Since game engines, tools, and related software are evolving and being updated at a rapid pace, the following list contains information regarding which version of each game engine the information in this study is based on. The most recent stable/long-term release of the game engines has been used and is listed (beta versions and similar excluded):

(i) Unity 2020.2.3

(ii) Unreal Engine 4.26

(iii) GameMaker 2.3.1.542

(iv) Godot 3.2.3

(v) CryEngine 5.6

(vi) Construct 3 r234.4

Table 1: Game engines and their inclusion criteria.

		General purpose?	Publicly available?	Included in study?
Data from steam	Unreal Engine	Yes	Yes	Yes
	Unity	Yes	Yes	Yes
	Source	?	No	No
	CryEngine	Yes	Yes	Yes
	Gamebryo	?	No	No
	IW	?	No	No
	Anvil	?	No	No
	id Tech	?	No	No
	Essence	?	No	No
	Clausewitz	?	No	No
Data from Itch.io	Unity	Yes	Yes	Yes
	Construct	Yes	Yes	Yes
	GameMaker	Yes	Yes	Yes
	Twine	No	Yes	No
	RPG Maker	No	Yes	No
	Bitsy	No	Yes	No
	PICO-8	No	Yes	No
	Unreal Engine	Yes	Yes	Yes
	Godot	Yes	Yes	Yes
	Ren'Py	No	Yes	No

6. Search Queries and Data Sources

To ensure that the search queries used would result in data relevant for the aim of the study, the terms used in localization processes presented by Fry and Lommel [11] and Esselink [13] among others were used as a starting point. The terms "globalization," "internationalization," and "localization" (including spelling variations) were used to search the official documentations of the included game engines. Searching the documentations using "globalization" gave no (0) hits related to the aim of the study. "Internationalization" gave results in three (3) of the official documentations. "Localization" gave most results: four (4) of the documentations contained results relevant to the aim of this study. All results from the "internationalization" query were also found when searching for "localization."

Based on this initial search, focus was put on "localization" as a search term. When browsing the results, other related key words were found ("translation" and "language") and incorporated in the final set of search queries. To catch as many instances of localization related information as possible, spelling variations and different forms of the search terms were used:

(i) Localization

(a) To catch different spellings (localization and localisation) as well as different inflections, the search has been carried out using locali∗ as search term

(ii) Translation

(a) The search query translat∗ has been used to catch variations of translate (translation, translate, translator, etc.)

(iii) Language

(a) The search query language∗ has been used to catch both singular and plural forms

The search queries were then used to find information related to the aim of the study from both the official documentation and the connected tool libraries available.

To get an overview of the state of localization tools in general purpose game engines, three main data sources related to the included game engines have been used: (1) the official documentation provided in with the game engine, (2) the standard production pipeline of the game engine and its plasticity, and (3) the official tool library connected to each game engine. Table 2 shows an overview of these areas, where each main area is divided into two subareas. The first subareas indicate whether the included game engines have the basic prerequisites to search the official documentation and expand the production pipeline and if an official tool library exists.

All included game engines have an official documentation available through the official websites. All official documentations have a search function making it possible to use the search queries to find documentation regarding localization. All included game engines have support for additional

Table 2: Overview of the general purpose game engines included in the study.

	Official documentation		Production pipeline		Official tool library	
	Searchable?	Addressing localization?	Expandable?	Localization tool included?	Connected tool library?	Localization tools available?
Construct	Yes	?	Yes	?	Yes	?
CryEngine	Yes	?	Yes	?	Yes	?
GameMaker	Yes	?	Yes	?	Yes	?
Godot	Yes	?	Yes	?	Yes	?
Unity	Yes	?	Yes	?	Yes	?
Unreal Engine	Yes	?	Yes	?	Yes	?

tools; i.e., the production pipeline can be expanded. All included game engines have official tool libraries, accessible either through the official website or from within the game engine itself.

7. An Overview of Localization Tools in General Purpose Game Engines

The following subchapters present the results from the tool review divided into the respective game engines. Due to the different structures of both the included official documentation and tool libraries in connection to the included game engines, the structure of the information relating to each game engine can differ slightly. Each subchapter is addressing the information found in the official documentation, the included localization tools, and the localization tools available in the tool libraries. Tables with more information regarding the localization tools found in the tool libraries are found in Supplementary Materials (available here).

It should be noted that information and descriptions (including historical information) regarding game engines are difficult to find from peer-reviewed or even industry sources. One of the most comprehensive sources containing relevant information of game engines is the English version of Wikipedia, hence the frequent use of Wikipedia sources in the subchapters.

7.1. Construct. Construct [45] is a game engine developed by Scirra Ltd., available since 2007. *Construct* has been released in three major versions: *Construct Classic* (rebranded as Classic with the release of Construct 2), *Construct 2*, and *Construct 3*. The first iteration, *Construct Classic*, was only available for development on the Windows operating system, using a visual programming coding interface only. The current iteration, *Construct 3*, has moved to run on multiple platforms, with the game engine running in a modern web browser of choice. Since 2019, the visual programming-only interface has been expanded, and *Construct 3* is now supporting programming in *JavaScript* [46].

Construct is limited to create 2d games, with the goal "[…] to make the best 2D game engine in the world" [47]. *Construct* is available to the public through a subscription model with Individual, Business, and Educational Organization licensing agreements. A free version of the Personal license is available with limitations in terms of how many assets allowed in a project, features in the editor, and publishing possibilities [48].

7.1.1. Localization Tools and Information in Construct and Its Documentation. There are no localization tools provided with *Construct*. There are no mentions of localization of game projects in the official documentation. The only reference in the documentation relating to the search queries is relating to how to change language in the *Construct* editor itself.

7.1.2. Localization Tools in Construct Tool Library. The official tool library connected to *Construct* is called *Construct Addons*, containing tools to be used for game creation and tools to alter the appearance of the *Construct* editor. *Construct Addons* are divided into four categories: *Plugins*, *Behaviours*, *Effects*, and *Themes*. There are 253 Addons in total available for *Construct*. Searching the *Construct Addons* for localization tools using the keywords results in two (2) tools: *c3-i18next* and *Google Translate*.

The two localization tools found in *Construct Addons* are both in the *Plugins* category of the tool library. Both tools have a similar functionality, acting as tool facing tools (i.e., tools that connects to other tools) between external frameworks for localization. The framework i18next, which the tool *c3-i18next* connects to, is an internationalization framework built-in *JavaScript*, providing a solution for several kinds of localization related tasks, scalable for the need of the specific product. The *i18next* framework can be connected to a TMS in order to structure the localization work. The tool with the name *Google Translate* connects to *MLKit*, a machine learning service toolkit provided by Google. *MLKit* has a number of localization and language-related features, including automated translation services through *the Google Translation API*.

The two (2) localization tools found in *Construct Addons* are tool facing tools.

7.2. CryEngine. CryEngine [49] is a game engine developed by *Crytek*, a German game development company. *CryEngine* was developed in 2002, as a tech demo to showcase the, at the time, latest graphics card technology. The tech demo gained attention from the community and was subsequently developed into a full game, *Far Cry* [50]. *CryEngine* has been used to develop a number of both internal games at *Crytek*

but has also been used by other companies, licensing the technology for their projects [51]. The latest iteration, *CryEngine V*, introduced a license model using a "pay what you want" licensing fee for the use of the game engine. If a commercial game is released using *CryEngine*, a royalty percentage has to be paid to *Crytek* [52].

7.2.1. Localization Tools and Information in CryEngine and Its Documentation. CryEngine has localization tools as a part of its standard toolset. The documentation describes the use of the localization system in relation to the different asset types that can be localized. The localization-related information in the documentation has separate chapters in relation to relevant categories. Examples of this include the following:

(1) User Interface (HUD/Menu)
    (i) UI Overview
        (A) Localization System
        (B) Texture Localization
(2) Dialog
    (i) Localization
(3) Audio
    (i) Audio Middleware
        (A) FMOD Studio Workflow
            (a) FMOD and Localization

The documentation covers localization of text (including dialogs), audio, graphical assets, and how external tools (as the FMOD audio middleware example above) can access the localization system and the localized assets. Each section has detailed descriptions on how to use the localization system for different kinds of assets. In the Dialog section an overview of the function is provided: "CRYENGINE is able to localize text and sound for different languages. All of the necessary data is stored inside pak files, stored in the <root>/Localization folder." Detailed descriptions of *PAK/Folder structure*, *Translating Spoken Text*, *Texture Localization*, *General Setup*, and *Test Localization Settings* are provided. Similar descriptions are found in the other localization-related sections, with both an overview of the use and detailed information on how to implement the localization into the game project.

The localization system in *CryEngine* is tool facing. The user needs to create the localization kits and then import them into the localization system using the described folder structure and process. This is further emphasized by the section in the official documentation regarding localization, where the exemplified with a screenshot of how a translation table should be formatted. The translation table, demonstrated created in *Excel*, is shown in Figure 5.

7.2.2. Localization Tools in CryEngine Tool Library. The official tool library connected to *CryEngine* is called *CryEngine Marketplace*, mostly containing assets for game creation (3d models, texture maps, sound effects, etc.) and also tools to expand the production pipeline. *CryEngine Marketplace* is divided into 11 categories, where *Plugins*, *Scrips*, *Full Project*, and *Training/Tutorials* are the categories not containing assets for game creation. In total, there are 1340 items in the *CryEngine Marketplace*.

The search queries did not result in any localization tools in the *CryEngine Marketplace*. Since the *CryEngine Marketplace* contains a limited number of tools (30 in total, divided into the categories listed above), a manual review of all available tools was made to ensure that any eventual localization tools are identified. The manual review did not reveal any localization tools.

7.3. GameMaker. GameMaker [44] is a game engine provided by *YoYo Games*. *GameMaker* was first developed and released by Mark Overmars in 1999 as a cross-platform game creation tool using a drag-and-drop visual programming interface as well as a proprietary scripting language called *GameMaker Language* (GML). *GameMaker* is intended to produce 2d games primarily but allows for rudimentary 3d game creation as well [54].

GameMaker is available through five different licenses: a Trial License providing access to *GameMaker* and its creation capabilities at no cost for 30 days with no publishing options; a Creator License for either Windows or MacOS which for a yearly fee gives access to the full functionality of *GameMaker* as a creation tool but limits its publishing options to publishing on the platform the game is developed on; a variety of Developer Licenses where the full functionality of the game engine is included and different publishing packages for mobile, desktop, or web are available for a one-time fee; Console Licenses including publishing on game consoles for a yearly fee; and an Education License for using *GameMaker* in an educational setting. The main differences between the licenses of *GameMaker* lie in which platforms to deploy to and its publishing possibilities; the functionality of *GameMaker* as a creation tool is similar [55].

7.3.1. Localization Tools and Information in GameMaker and Its Documentation. GameMaker does not have any built-in tools to support localization work. The documentation mentions localization and related terms with one reoccurring statement regarding how to handle the reading and writing of various text files, highlighting the importance of using the correct text encoding format (in this case, UTF8) to ensure compatibility with non-Roman letters. The same text is found in three separate categories/sections of the *GameMaker* documentation: Text Files (containing an overview of the use of text files); Ini Files (containing a brief overview of what an ini file is); and File Library (an introduction to the Files section of the visual scripting interface in *GameMaker*) [56].

No further information regarding localization of a project could be found in the *GameMaker* documentation.

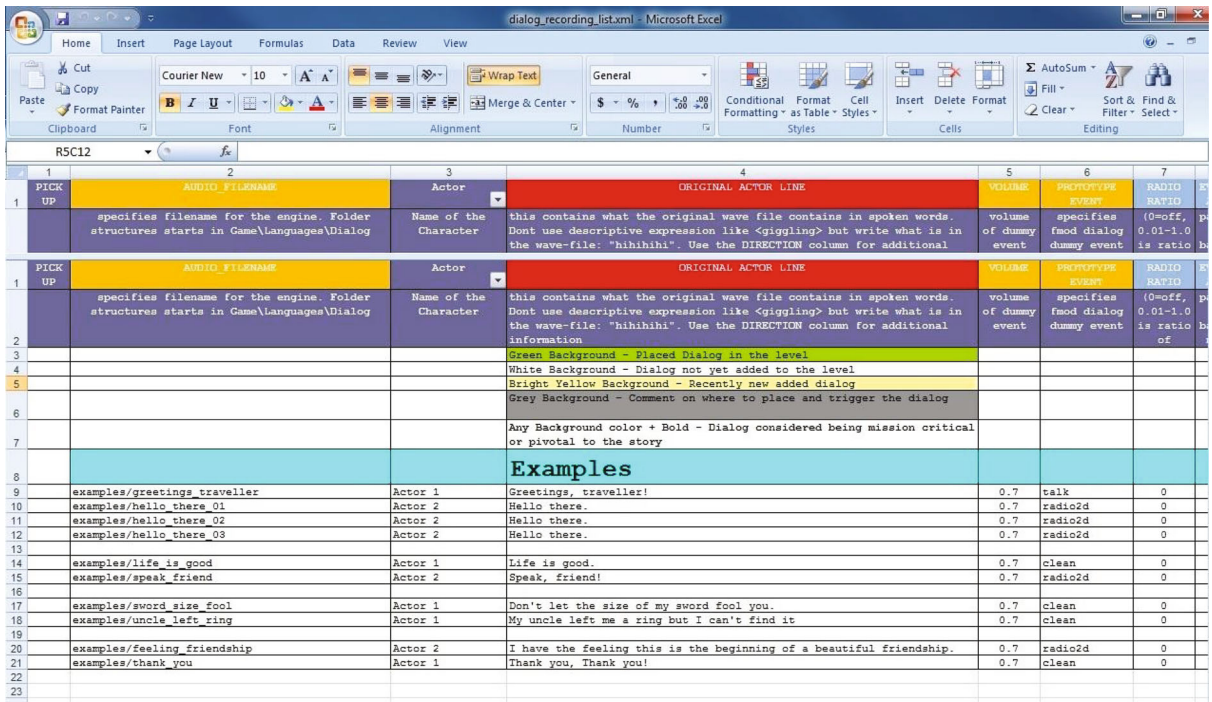

Figure 5: Screenshot of the Dialog System/Localization Manager section of *CryEngine* official documentation [53].

7.3.2. Localization Tools in GameMaker Tool Library. The official tool library connected to *GameMaker* is called *Marketplace*, containing both assets for game creation (i.e., graphical assets, sound, music, etc.) and tools and scripts to expand the functionality of the game engine. *GameMaker Marketplace* is divided into 18 categories, and there are 3499 items in total available in *GameMaker Marketplace*.

Searching the *GameMaker Marketplace* for localization tools using the keywords results in 7 tools. Five (5) of these are targeted to aid in the process of handling different languages in the game itself, i.e., providing functions to read an external localization kit containing source and target locales and to switch language during runtime by using specific commands. These 5 tools are not used to create the localization kit itself, but acting as a bridge between a localization kit created elsewhere and the game project, i.e., tool facing tools. One (1) tool, *Arabic Support*, is to aid the process of using Arabic letters in a game project, providing scripts to handle the issues related to having another character set. One (1) tool, *Multilanguage and Translate*, is a more comprehensive set of localization scripts connecting to an external localization manager where automatic or manual translation can be managed.

All 7 localization tools on *GameMaker Marketplace* are tool facing tools.

7.4. Godot. Godot [2] is a 2d and 3d game engine initially released in 2007 by the developers Juan Linietsky and Ariel Manzur. Linietzky and Manzur developed *Godot* as an in-house game engine used in several game companies in Latin America, before releasing *Godot* as an open source project under the *MIT License* in 2014. The *MIT License* grants users to use *Godot* for free, for any purpose, to make changes to the engine, and to distribute both unmodified and changed versions of the engine. The free nature of usage also applies to the games created with the engine. *Godot* can be run on several platforms and be used to create games targeting PC, mobile, and web [57].

7.4.1. Localization Tools and Information in Godot and Its Documentation. Godot comes with localization tools in the standard package, and the documentation has comprehensive information regarding localization and internationalization. The documentation has a *Project Workflow* section, where subsections are describing how to import different kinds of assets into the game project. The subsections cover the importing process and specific information on how to import images, audio samples, translations, and 3d scenes. The section describes the process in detail, using the built-in localization tool to handle different assets to localize.

Further, the documentation also covers Internationalization as a topic in the *Tutorials* section. This section gives a general overview of an internationalization and localization process, covering both implementation aspects and general aspects such as lists of locale codes and text formats and standards. The examples in the documentation regarding localization using *Godot* also refer to an official localization demo, which shows the localization examples from the documentation in a project form. The localization demo can be downloaded from the *Asset Library*.

The localization tool included as a standard tool in *Godot* is tool facing. The tool is used to import previously made localization kits through a GUI. The user can also get an overview of both the languages used and the connected text and assets to be localized, although not created and modified. Figure 6 is a screen capture of the *Godot* localization tool.

7.4.2. Localization Tools in Godot Tool Library. The official tool library connected to *Godot* is called *Asset Library*, containing both assets for game creation (graphical assets, sound, music, etc.) and tools and scripts to expand the functionality of the game engine. *Godot Asset Library* is divided into 10 categories. 813 items in total are available in the *Godot Asset Library*, and three (3) of these are localization tools.

Searching *Godot Asset Library* for localization tools revealed only three in total, and one of the results is the aforementioned translation demo called *GUI Translation Demo*. The link in the documentation mentioning the localization demo leads to this asset. It is worth noting that the *GUI Translation Demo* itself does not appear in the search results when using the locali∗ search query, only when using translat∗.

One (1) of the tools in the *Godot Asset Library* is tool facing (the aforementioned *GUI Translation Demo*, containing scripts to import external files), while two (2) are user facing (*Localization Editor* and *Translation Editor*) used to create and manage localization-related work.

7.5. Unity. Unity [9] is a game engine provided by *Unity Technologies*, released first in 2005. *Unity* was developed with the goal to make state-of-the-art game creation technology easily accessible to developers of all sizes. The first versions of *Unity* focused on 3d games, later shifting to support the creation of both 2d and 3d games for a wide range of platforms. *Unity* is available to the general public through either a number of free individual versions aimed at students and hobbyist developers or through subscriptions aimed at game developers with larger teams. *Unity* supports deployment to a large number of target platforms, including smart devices, computers, and game consoles [58].

7.5.1. Localization Tools and Information in Unity and Its Documentation. No results regarding localization or related concepts were found in the official *Unity* documentation using the search query.

7.5.2. Localization Tools in Unity Tool Library. The official tool library connected to *Unity* is called *Unity Asset Store*, containing both assets for game creation (graphical assets, sound, music, etc.) and tools and scripts to expand the functionality of the game engine. *Unity Asset Store* is divided into 3 main categories: *Assets* (containing 3d models, textures, sound, etc., for game creation); *Tools* (containing tools to expand the production pipeline in various ways); and *Services* (containing specific services such as monetization, in-app purchasing, and advertising). 65977 items in total are available in the *Unity Asset Store*, of which 8921 are in the *Tools* category. The *Tools* category of *Unity Asset Store* contains 21 subcategories, of which one is *Localization*. The *Localization* subcategory contains 80 tools in total. Since the *Unity Asset Store* has a specific category for localization tools, the 80 tools in that category have been used in the study. No further search has been conducted to find additional tools outside the *Localization* category.

An overview of the localization tools in *Unity Asset Store* is found in Supplementary Materials (available here). Of the tools in the *Localization* category, 59 tools are tool facing and 21 tools are user facing.

7.6. Unreal Engine. Unreal Engine [26] is a game engine provided by *Epic Games*. *Unreal Engine* has its origins in the AAA game development scene as the technology behind the *Unreal series* of games first released in 1998. Even though developed as an internal game engine at first, the *Unreal Engine* was early on available to external game development studios through licensing deals. In 2009 a version of *Unreal Engine 3*, named *Unreal Development Kit* (*UDK*), was released to the general public. *UDK* allowed game developers to access many of the features included in *Unreal Engine* without having to license the technology from *Epic Games*. The licensing requirement for *Unreal Engine* was further lifted in 2014, allowing schools and universities access to the full version of *Unreal Engine 4*. Since 2015, *Epic Games* has been providing *Unreal Engine* to the general public, to both professional and hobby game developers. Commercial products developed with *Unreal Engine* owe a royalty percentage to *Epic Games* if certain levels of revenues are reached by the product in question. *Unreal Engine* supports deployment to a large number of target platforms, including smart devices, computers, and game consoles [59].

7.6.1. Localization Tools and Information in Unreal Engine and Its Documentation. Unreal Engine provides localization tools in the standard installation. The official documentation gives a thorough explanation regarding how to use the localization system, covering several types of assets, and how to make the workflow more effective. In the section *Setting Up Your Production Pipeline*, the documentation has a subsection named *Localization*, covering the following topics: *Localization Overview*, *Text Localization*, *String Tables*, *Asset Localization*, *Localization Tools*, *Pipeline Optimization*, and *Managing the Active Culture at Runtime*. All information regarding localization is gathered under the *Localization* subsection. The *Localization Tools* topic goes into detail regarding the tools provided within the localization system of *Unreal Engine*, describing the workflow and the different components of the system. The *Localization Tools* in *Unreal Engine* is divided into *The Localization Dashboard*, describing the tool where the localization targets are managed; *Translation Editor*, where the text in the game can be checked and edited; *Translation Picker*, displaying information regarding text values used in the UI of the project; and *Automating Localization*, where parts of the localization pipeline can be automated using the *Unreal Automation Tool*.

The localization tool provided through the *Unreal Engine* standard installation is user facing, where the user can create and edit the localization kit.

7.6.2. Localization Tools in Unreal Engine Tool Library. The official tool library connected to *Unreal Engine* is called *Unreal Engine Marketplace*, containing both assets for game creation (graphical assets, sound, music, etc.) and tools and scripts to expand the functionality of the game engine.

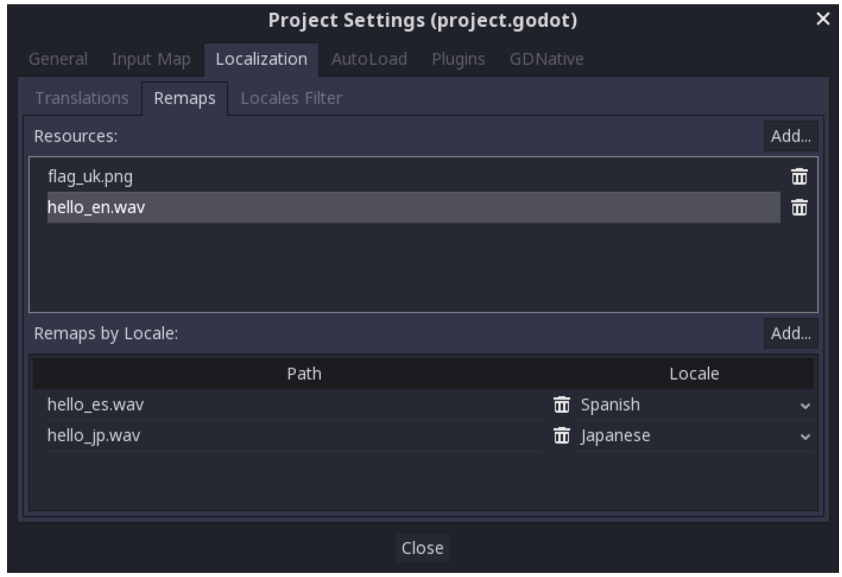

Figure 6: A screenshot of the localization tool in Godot.

Table 3: Summary of localization tools in general purpose game engines.

	Official documentation		Production pipeline		Official tool library	
	Searchable?	Addressing localization?	Expandable?	Localization tool included?	Connected tool library?	Localization tools available?
Construct	Yes	No	Yes	No	Yes	Yes
CryEngine	Yes	Yes	Yes	Yes	Yes	No
GameMaker	Yes	No *	Yes	No	Yes	Yes
Godot	Yes	Yes	Yes	Yes	Yes	Yes
Unity	Yes	No	Yes	No	Yes	Yes
Unreal Engine	Yes	Yes	Yes	Yes	Yes	Yes

Unreal Engine Marketplace is divided into 16 categories. 15,785 items in total are available at *Unreal Engine Marketplace*, of which three (3) are localization.

Two of the localization tools found when searching the *Unreal Engine Marketplace* are in the *Engine Plugin* category, both of them providing a solution for using *Amazon Web Services* (*AWS*) translation service with the *Blueprint* visual scripting system. The other tool, *MSL Toolkit*, is in the *Complete Project* section, providing several functions including menu, settings, and localization systems, promising to save time and simplify the workflow when building menu systems in a game project.

Unreal Engine Marketplace contains two (2) tool facing localization tools and one (1) user facing tool.

8. Summary of Localization Tool Overview

Table 3 shows an overview of the status of localization tools available for general purpose game engines. All game engines have official documentation available online. All of the official documentations are searchable and can be searched freely with no login required. While being not a part of the aim of the study, it can be noted that *Unity*, *Unreal Engine*, and *Godot* have their documentation translated into different languages than English, while *GameMaker* and *Construct* only have documentation in English. *CryEngine*, *Unreal Engine*, and *Godot* have documentation that addresses localization specifically, i.e., the same engines that have localization tools included. The *GameMaker* documentation has one reference to localization (described in the *GameMaker*

Table 4: Summary of types of tools in the official tool libraries per game engine.

	Tool facing tools	User facing tools	Total
Construct	2	0	2
CryEngine	0	0	0
GameMaker	7	0	7
Godot	1	2	3
Unity	59	21	80
Unreal Engine	2	1	3
Total	71	24	

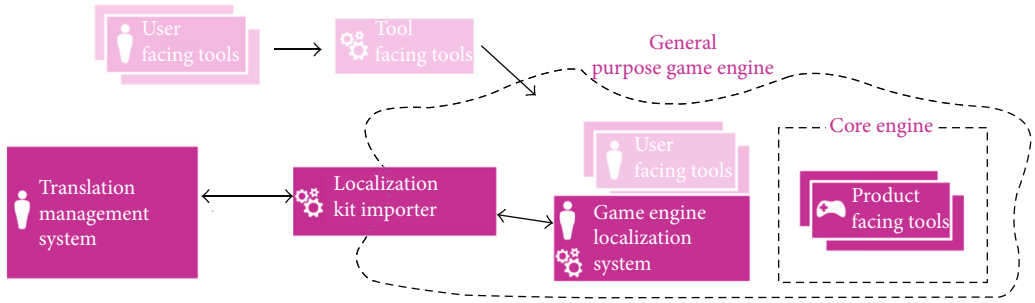

Figure 7: A proposed model of a localization-friendly game production pipeline.

chapter above), hence the asterisk in the table. All game engines in the study are expandable; i.e., the production pipelines can be expanded with additional tools. *Construct*, *GameMaker*, and *Unity* do not have a localization tool in their standard toolset and have no detailed localization-related information in their official documentation either. All included game engines have an official tool library connected to them, where additional functionality can be added through tools created by either community members or the providers of the engines. All official tool libraries are available online and are searchable. *CryEngine* is the only game engine in the study whose official tool library does not have any localization tools available.

Regarding user or tool facing tools, 71 of all identified tools are tool facing and 24 are user facing. Only *Unreal Engine* has a user facing tool included as a part of the standard installation, while *Godot* and *CryEngine* provide tool facing tools for localization. A summary of the types of tools in the official tool libraries is presented in Table 4.

Since a majority of the identified and classified localization tools are tool facing, additional tools are needed to conduct the localization work itself (i.e., using a TMS or other tool for creating a localization kit). All user facing tools identified in the study have a tool facing component; i.e., the localization work can be integrated using the tool itself.

9. Discussion and Conclusions

The market around game engines and tools are diverse, and several actors with different aims and scopes are involved. Ranging from multibillion dollar companies to smaller indie-like studios, the game engine market might be as diverse at the game market itself. As a conclusion to the paper, the research questions are addressed, starting with the subquestions leading to an answer of the main research question.

(a) Which commonly used general purpose game engines have tools included for game localization?

Three (3) of the included game engines have tools for localization provided as a part of the standard installation: *CryEngine*, *Godot*, and *Unreal Engine*. Two of them, *CryEngine* and *Unreal Engine*, have their roots in the AAA development section of the game industry, used to produce several AAA game titles during a long time before the technology was made available to the general development community. Both these game engines have localization systems as a part of the standard tool package and detailed information on how to use them in the official documentation. This might imply that localization has been an important factor in developing AAA games (such as *Unreal Tournament* or *Far Cry*) and that these tools have "survived" the transition from an in-house technology to a publicly available game engine.

The third example of a game engine having well-documented localization tools is *Godot*, an engine also having its roots in the game industry as an internal development tool. Although maybe not as well-known as the previous AAA examples, *Godot* was mainly used at game companies in Latin America before its turn to open source. Could its Latin America roots have influenced the inclusion of a localization tool, both due to a local market with diverse language preferences?

Unity, often hailed as the most wide-spread game engine, does not have any included game localization tools. Game localization is not mentioned in the official documentation either, meaning that a developer using *Unity* has to build a localization-friendly production pipeline using tools developed either by themselves or by using one or more of the multitude of localization tools available in the *Unity Asset Store*. Nicoll and Keogh [8] lift the democratization aspect of *Unity Technologies* marketing strategies. *Unity* is in its marketing hailed as the technology behind half of the mobile games in the world [60], thus having a large user group. The democratization aspects includes strong connections to its developer community, where sharing of knowledge is common. This is manifested in the localization tools area as, in comparison to the other included game engines, a large portion of localization tools of different kinds.

(b) How are the localization tools described in the corresponding official documentation?

Where included localization tools exists (i.e., *CryEngine*, *Godot*, and *Unreal Engine*), the tools are described in such detail that it would be possible to have localization-friendly development pipeline. The descriptions of the general process often relate to the proposed localization processes identified from previous research [11, 13, 29]. In the game engines where no localization tools are included as a part of the standard toolset, there are limited (or no) mentions of localization and related concepts.

Looking at the description of the tools available in the connected tool libraries, the documentation varies. Some have thorough documentation, explaining in detail how the localization process can be carried out, while some have only a few lines describing the general usage. The terminologies related to localization topics proposed by for instance Esselink [13] are in these scenarios often used scarcely, or there are similar terms giving search results relating to other aspects of game development. Examples of this are multiple spelling variants (localization vs. localisation, etc.) as well as terms with multiple meanings. Translation is, in relation to game engines at least, often used to express the mathematical meaning of translating a position of an object or point in a grid system, something that is commonly discussed in both 2d and 3d game developments. Alas, translation is also among the most common terms relating to localization processes.

(c) In which ways can the production pipeline be expanded by adding additional tools to achieve a localization-friendly development process?

The production pipeline can be expanded with localization tools in all of the game engines in this study. All game engines included in the study have support either to create tools to accommodate for localization or to use existing tools by expanding the production pipeline using the connected tool library available. All included game engines have an official tool library connected to them.

The majority of tool facing localization tools highlights the need of adding translation management systems (or similar) to the pipeline in order to create localization kits including the necessary localization data [13]. Figure 7 depicts a proposed model of a production pipeline where a *Translation Management System* (user facing, used to create and manage the localization kit) is connected to a *Localization Kit Importer* (tool facing, handling the connection and data transfer between the TMS and the game engine), which connects to a *Game Engine Localization System* (both tool and user facing, handling the connection to subsequent tools in the pipeline and connection to other game assets as well).

The proposed model of a localization-friendly game production pipeline allows for translation work, by working with the TMS, as well as game development tasks using the proposed *Game engine localization system* where the connections between the localization kit and the assets to be localized are made. *Godot* contains an example of a Game engine localization system (see Figure 6) where corresponding localization strings can be mapped to specific game assets.

Based on the subquestions, an answer to the main research question can be given:

(d) "*How can a localization-friendly game production pipeline be constructed using a general purpose game engine and its connected tool library?*"

The proposed model in Figure 7 provides a partial answer to the research question. The technical prerequisites for each project would still be project specific. As Bernal-Merino [32] points out, it is difficult to generalize a model suiting all game development projects. But with the proposed model of a production pipeline, the pieces of the pipeline can be identified and planned for. The state of game localization tools in contemporary general purpose game engines is good, where contemporary general purpose game engines have localization support either built-in or expandable by adding additional tools. In some cases, e.g., *GameMaker* and *Construct*, it can be considered unlikely to "stumble" upon localization processes either by using the game engine or by browsing the tool library or reading the official documentation due to the lack of visible localization related topics. Even though *Unity* has neither built-in support for localization nor addressing it in its documentation, its large community and official tool library with plenty of localization tools make it plausible that a localization-friendly development setting can be achieved.

Even when a game engine has localization support (as in the cases *CryEngine*, *Godot*, and *Unreal Engine*), the localization functions and tools are not in the immediate line of sight when using either of the game engines. This can of course be related to the actual need of localization. If the game is not planned to be released in multiple languages or territories, a localization work will be down prioritized. But it is much easier to have plans for it when localization is decided to be implemented.

The information regarding *Godot* and its development and evolution also highlights the importance of having reliable information sources regarding game production, game production tools, and related topics. Even though Wikipedia, where background information regarding the game engines in this paper to a large extent was gathered from, is very useful for such information, its reliability can (and should) be questioned. Until there is a larger body of knowledge with peer-reviewed and well-documented sources, Wikipedia is the best (and sometimes only) choice.

As an inspiration for a future work, the introductory quote from the *Godot* documentation can serve as an inspiration:

"The world is full of different markets and cultures and, to maximize profits™[sic], nowadays games are released in several languages. To solve this, internationalized text must be supported in any modern game engine"[1]

The quote as well as many of the tools found in this study focuses on text. While text is a large (and important) part of localization (whether the text is in graphical assets or in recorded dialogue is also an aspect), there is an aspect of games posing an interesting localization challenge not commonly addressed—game play or game mechanics. A suggestion for a future work is to research game production and game localization regarding regional differences in these areas, both from a cultural and technological perspective.

Data Availability

The details of the systematic mapping review derived from the corresponding official tool libraries used to support the findings of this study are included within the supplementary information file(s).

Conflicts of Interest

The author declares that there are no conflicts of interest.

Acknowledgments

This research has been funded by the Game Hub Scandinavia 2.0 project under the Interreg Öresund-Kattegat-Skagerrak European Regional Development Fund (Project ID NYPS 20201849).

Supplementary Materials

The data regarding localization tools found in the official tool libraries of the included game engines is provided as a separate file. All other data are presented within the paper. *(Supplementary Materials)*

References

[1] Godot Engine Team, "Godot documentation - Importing translations," 2021, https://docs.godotengine.org/en/stable/getting_started/workflow/assets/importing_translations.html/.

[2] Godot Engine Team, "Godot [general purpose game engine]," 2021.

[3] J. R. Whitson, "Voodoo software and boundary objects in game development: how developers collaborate and conflict with game engines and art tools," *New Media & Society*, vol. 20, no. 7, pp. 2315–2332, 2018.

[4] F. Parker, *Indie game studies year eleven*, DIGRA 2013-DeFragging Game Studies, 2013.

[5] M. B. Garda and P. Grabarczyk, "Is every indie game independent? Towards the concept of independent game," *Game Studies*, vol. 16, no. 1, 2016.

[6] G. Freeman and N. McNeese, "Exploring indie game development: team practices and social experiences in a creativity-centric technology community," *Computer Supported Cooperative Work (CSCW)*, vol. 28, no. 3-4, article 9348, pp. 723–748, 2019.

[7] L. S. Pereira and M. M. S. Bernardes, "Aspects of independent game Production," *Computers in Entertainment*, vol. 16, no. 4, pp. 1–16, 2018.

[8] B. Nicoll and B. Keogh, *The unity game engine and the circuits of cultural software*, Palgrave Pivot, 1 edition, 2019.

[9] Unity Technologies, "Unity [general purpose game engine]," 2021.

[10] M. Toftedahl, P. Backlund, and H. Engström, "Localization from an indie game production perspective: why, when and how?," in *DiGRA '18 - Proceedings of the 2018 DiGRA International Conference: The Game is the Message*, Torino, Italy, 2018.

[11] D. Fry and A. Lommel, *The localization industry primer*, LISA, 2003.

[12] C. M. Hevian, "Video games localisation: posing new challenges to the translator," *Perspectives*, vol. 14, no. 4, pp. 306–323, 2007.

[13] B. Esselink, "A Practical Guide to Localization," in *Language International World Directory*, vol. 4, John Benjamins Publishing Co, Amsterdam, 2000.

[14] C. O'Donnell, "Games are not convergence: the lost promise of digital production and convergence," *Convergence*, vol. 17, no. 3, pp. 271–286, 2011.

[15] A. Thorn, *Game Engine Design and Implementation*, Jones & Bartlett Publishers, Sudbury, MA, 2011.

[16] Lucasfilm Games, "Maniac Mansion [Computer Game]," 1987.

[17] M. L. Black, "Narrative and spatial form in digital media," *Games and Culture*, vol. 7, no. 3, pp. 209–237, 2012.

[18] Activision, Garry Kitchen's Game, "Maker [special purpose game engine]," 1985.

[19] Sensible Software, "Shoot'Em-Up Construction Kit [Special Purpose Game Engine]," 1987.

[20] J. Aycock and K. Biittner, "LeGACy code: studying how (amateur) game developers used graphic adventure creator," in *International Conference on the Foundations of Digital Games*, vol. 23, Bugibba, Malta, 2020.

[21] id Software, "Doom [computer game]," 1993.

[22] H. Lowood, "Game engines and game history," in *History of Games International Conference Proceedings*, p. 2014, Kinephanos, January 2014.

[23] E. F. Anderson, S. Engel, P. Comninos, and L. McLoughlin, "The case for research in game engine architecture," in *Proceedings of the 2008 Conference on Future Play: Research, Play, Share, Future Play 2008*, Toronto, Ontario, Canada, 2008.

[24] M. Toftedahl and H. Engström, "A taxonomy of game engines and the tools that drive the industry," in *DiGRA '19 - Proceedings of the 2019 DiGRA International Conference: Game, Play and the Emerging Ludo-Mix*, Kyoto, 2019.

[25] C. O'Donnell, *Developer's Dilemma: The Secret World of Videogame Creators*, The MIT Press, 2014.

[26] Epic Games, "Unreal Engine [general purpose game engine]," 2021.

[27] Epic Games, "Unreal [computer game]," 1998.

[28] C. Mangiron, "Research in game localisation," *The Journal of Internationalization and Localization*, vol. 4, no. 2, pp. 74–99, 2018.

[29] M. O'Hagan and C. Mangiron, *Game Localization: Translating for the Global Digital Entertainment Industry*, vol. 106, John Benjamins Publishing, 2013.

[30] M. Toftedahl, "Localization and regional aspects of game production - a research overview," in *13th International Conference on Game and Entertainment Technologies*, Zagreb, Croatia, 2020.

[31] H. M. Chandler and S. O. M. Deming, *The game localization handbook*, Jones & Bartlett Publishers, 2011.

[32] M. Á. Bernal-Merino, *Translation and localisation in video games: making entertainment software global*, Routledge, 2014.

[33] B. Esselink, "The evolution of localization," *The Guide from Multilingual Computing & Technology: Localization*, vol. 14, no. 5, pp. 4–7, 2003.

[34] Weblate Team, "Weblate [Computer Software]," 2021.

[35] P. Muñoz Sánchez and R. López Sánchez, *The ins and outs of the video game localization process for mobile devices*, vol. 14, no. 14, 2016Tradumàtica, 2016.

[36] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, Bari, Italy, 2008.

[37] Valve, "Steam [digital storefront]," 2021.

[38] itch corp, "itch.io [digital storefront]," 2021.

[39] Interactive Fiction Technology Foundation, "Twine [Special Purpose Game Engine]," 2021.

[40] A. Le Doux, "Bitsy [Special Purpose Game Engine]," 2021.

[41] T. Rothamel, "Ren'Py [Special Purpose Game Engine]," 2021.

[42] Enterbrain, "RPG Maker [special purpose game engine]," 2020.

[43] Lexaloffle Games, *PICO-8 [Special Purpose Game Engine]*, 2021.

[44] YoYo Games Ltd, "GameMaker [general purpose game engine]," 2021.

[45] Scirra Ltd, "Construct [General Purpose Game Engine]," 2021.

[46] Wikipedia, "Construct (game engine)," 2021, https://en.wikipedia.org/wiki/Construct_(game_engine)/.

[47] Scirra Ltd, "Construct FAQ," 2021, https://www.construct.net/en/make-games/faq/.

[48] Scirra Ltd, "Construct Licences," 2021, https://www.construct.net/en/make-games/buy-construct/.

[49] Crytek, "CryEngine," 2021, https://www.cryengine.com/.

[50] Crytek, "Far Cry," 2004.

[51] Wikipedia, "CryEngine," 2021, https://en.wikipedia.org/wiki/CryEngine/.

[52] Crytek, "CryEngine Licences," 2021, https://www.cryengine.com/ce-terms/.

[53] Crytek, "CryEngine Documentation - Localization Manager," 2021, https://docs.cryengine.com/display/SDKDOC2/Localization+Manager/.

[54] Wikipedia, "GameMaker," 2021, https://en.wikipedia.org/wiki/GameMaker/.

[55] YoYo Games, "GameMaker Licences," 2021, https://www.yoyogames.com/en/get/.

[56] YoYo Games, "GameMaker Documentation - Ini Files," 2021, https://manual.yoyogames.com/GameMaker_Language/GML_Reference/File_Handling/Ini_Files/Ini_Files.htm/.

[57] Wikipedia, "Godot (game engine)," 2021, https://en.wikipedia.org/wiki/Godot_(game_engine)/.

[58] Wikipedia, "Unity (game engine)," 2021, https://en.wikipedia.org/wiki/Unity_(game_engine)/.

[59] Wikipedia, "Unreal Engine," 2021, https://en.wikipedia.org/wiki/Unreal_Engine/.

[60] Unity Technologies, "Unity - the game engine of choice for mobile game developers," 2021, https://en.wikipedia.org/wiki/Https://unity.com/pages/unity-pro-mobile-gamesUnreal_Engine/.